



3A HiISP

## Development Reference

Issue	03
Date	2014-02-26

**Copyright © HiSilicon Technologies Co., Ltd. 2013-2014. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

## **Trademarks and Permissions**



**HISILICON**, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **HiSilicon Technologies Co., Ltd.**

Address: Huawei Industrial Base  
Bantian, Longgang  
Shenzhen 518129  
People's Republic of China

Website: <http://www.hisilicon.com>

Email: [support@hisilicon.com](mailto:support@hisilicon.com)



# About This Document

## Purpose

This document describes the submodules of the image signal processor (ISP) and their application programming interfaces (APIs), data structures, and error codes.

## Related Version

The following table lists the product version related to this document.

Product Name	Version
Hi3516	V100
Hi3518	V100



## Intended Audience

This document is intended for:




- Technical support personnel
- Software development engineers

## Symbol Conventions

The symbols that may be found in this document are defined as follows:

Symbol	Description
 <b>DANGER</b>	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 <b>WARNING</b>	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.



Symbol	Description
 <b>CAUTION</b>	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 <b>TIP</b>	Provides a tip that may help you solve a problem or save time.
 <b>NOTE</b>	Provides additional information to emphasize or supplement important points in the main text.

## Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

### Issue 03 (2014-02-26)

This issue is the third draft release, which incorporates the following changes:

#### Chapter 3 AE

In section 3.4.2, the MPIs `HI_MPI_ISP_SetAEDelayAttr` and `HI_MPI_ISP_GetAEDelayAttr` are added.

In section 3.5, **s16HistError** is added to the **Syntax** and **Member** fields of `ISP_EXP_STA_INFO_S`.

In section 3.5.1, `AE_SENSOR_GAININFO_S` is added. The **pfn\_cmos\_again\_calc\_table** and **pfn\_cmos\_dgain\_calc\_table** parameters are added to the **Member** field of `AE_SENSOR_EXP_FUNC_S`. The description in the **Note** field of `AE_ACCURACY_S` is updated.

In section 3.5.2, the description in the **Note** field of `ISP_AE_MODE_E` is updated, and the description of the maximum and minimum exposure time is updated in the **Note** field of `ISP_AE_ATTR_S`. The data structures `ISP_AE_ROUTE_NODE_S`, `ISP_AE_ROUTE_S`, and `ISP_AE_DELAY_S` are added.

#### Chapter 4 AWB

In section 4.4.2, the MPIs `HI_MPI_ISP_SetAWBAlgType`, `HI_MPI_ISP_GetAWBAlgType`, `HI_MPI_ISP_SetAdvAWBAttr`, `HI_MPI_ISP_GetAdvAWBAttr`, `HI_MPI_ISP_SetLightSource`, and `HI_MPI_ISP_GetLightSource` are added.

In section 4.5.2, the data structures `ISP_AWB_ALG_TYPE_E`, `ISP_ADV_AWB_ATTR_S`, `ISP_AWB_LIGHTSOURCE_INFO_S`, and `ISP_AWB_ADD_LIGHTSOURCE_S` are added, and the **Syntax**, **Member**, and **Difference** fields of `ISP_AWB_ATTR_S` are modified.

#### Chapter 6 IMP

In section 6.3.3, **u32VarianceSpace** and **u32VarianceIntensity** are added to the **Syntax** and **Member** fields of `ISP_DRC_ATTR_S`. The description of the **u32SlopeMax**, **u32SlopeMin**, **u32WhiteLevel**, and **u32BlackLevel** parameters is updated in the **Member** field of `ISP_DRC_ATTR_S`.



In section 6.6.4, the **Syntax**, **Member**, and **Note** fields of ISP\_CR\_ATTR\_S are updated, and Table 6-4 is added.

In section 6.8.4, the descriptions in the **Member** field of ISP\_DIS\_INFO\_S are updated.

In section 6.11.3, **u8LumThresh** and **u8NpOffset** are added to the **Syntax** and **Member** fields of ISP\_DEMOSAIC\_ATTR\_S, and Table 6-6 and Table 6-7 are added.

Section 6.13 is added.

#### **Chapter 10 Proc Debugging Information**

In section 10.1, the loading method is updated.

### **Issue 02 (2013-09-25)**

This issue is the second draft release, which incorporates the following changes:

#### **Chapter 3 AE**

The description of the **u32SystemGainMax** parameter of ISP\_AE\_ATTR\_EX\_S is updated.

The **u8ExpHistTarget[5]** parameter and its description are added to ISP\_EXP\_STA\_INFO\_S. In addition, **[Note]** is added.

The description of the analog gain and digital gain of the high-precision sensor and the digital gain of the high-precision ISP is removed from **[Note]** in ISP\_INNER\_STATE\_INFO\_S.

#### **Chapter 4 AWB**

The **u32Rgain**, **u32Ggain**, and **u32Bgain** parameters and their description are added to ISP\_WB\_STA\_INFO\_S.

#### **Chapter 8 AF**

The **u8MetricsShift** and **u8NpOffset** parameters and their description are added to ISP\_FOCUS\_STA\_INFO\_S.

#### **Chapter 10 Proc Debugging Information**

This chapter is added.

### **Issue 01 (2013-07-16)**

This issue is first draft release.



# Contents

<b>About This Document.....</b>	<b>i</b>
<b>1 ISP .....</b>	<b>1</b>
1.1 Overview .....	1
1.2 Function Description .....	1
1.2.2 Design Methodology.....	2
1.2.3 File Structure.....	2
1.2.4 Development Mode.....	3
1.2.5 Internal Process .....	4
1.2.6 Software Process .....	5
<b>2 System Control .....</b>	<b>7</b>
2.1 Function Description .....	7
2.2 API Reference .....	7
2.3 Data Structures .....	33
<b>3 AE .....</b>	<b>62</b>
3.1 Overview .....	62
3.2 Important Concepts .....	62
3.3 Function Description .....	63
3.4 API Reference .....	64
3.4.1 AE Algorithm Library MPIS .....	64
3.4.2 AE Control MPIS.....	67
3.4.3 AI Control MPIS.....	85
3.5 Data Structures .....	89
3.5.1 Registration .....	89
3.5.2 AE .....	94
3.5.3 AI .....	105
<b>4 AWB .....</b>	<b>110</b>
4.1 Overview .....	110
4.2 Important Concepts .....	110
4.3 Function Description .....	110
4.4 API Reference .....	112
4.4.1 AWB Algorithm Library MPIS.....	112



4.4.2 AWB Control MPIs .....	114
4.4.3 WB Statistics MPIs .....	126
4.5 Data Structures .....	131
4.5.1 Registration .....	131
4.5.2 WB .....	134
<b>5 CCM.....</b>	<b>144</b>
5.1 Overview .....	144
5.2 Important Concepts .....	144
5.3 Function Description .....	144
5.4 API Reference .....	145
5.5 Data Structures .....	150
<b>6 IMP.....</b>	<b>154</b>
6.1 Sharpen .....	154
6.1.1 Function Description .....	154
6.1.2 API Reference .....	154
6.1.3 Data Structure .....	156
6.2 Gamma .....	158
6.2.1 Function Description .....	158
6.2.2 API Reference .....	158
6.2.3 Data Structures .....	162
6.3 DRC .....	165
6.3.1 Function Description .....	165
6.3.2 API Reference .....	165
6.3.3 Data Structure .....	167
6.4 Lens Shading Correction .....	169
6.4.1 Overview .....	169
6.4.2 Function Description .....	169
6.4.3 API Reference .....	170
6.4.4 Data Structures .....	174
6.5 Defect Pixel .....	179
6.5.1 Overview .....	179
6.5.2 Function Description .....	179
6.5.3 API Reference .....	180
6.5.4 Data Structures .....	183
6.6 Crosstalk Removal .....	185
6.6.1 Overview .....	185
6.6.2 Function Description .....	185
6.6.3 API Reference .....	186
6.6.4 Data Structure .....	187
6.7 Denoise .....	189
6.7.1 Overview .....	189



6.7.2 Function Description.....	189
6.7.3 API Reference .....	189
6.7.4 Data Structure .....	191
6.8 DIS .....	193
6.8.1 Overview.....	193
6.8.2 Function Description.....	193
6.8.3 API Reference .....	193
6.8.4 Data Structures.....	198
6.9 Anti-fog .....	200
6.9.1 Function Description.....	200
6.9.2 API Reference .....	200
6.9.3 Data Structure .....	202
6.10 Anti-False Color.....	203
6.10.1 Overview.....	203
6.10.2 Function Description.....	203
6.10.3 API Reference .....	203
6.10.4 Data Structure .....	205
6.11 Demosaic.....	206
6.11.1 Function Description.....	206
6.11.2 API Reference .....	206
6.11.3 Data Structure.....	208
6.12 Black Level .....	211
6.12.1 Overview.....	211
6.12.2 API Reference .....	211
6.12.3 Data Structures.....	213
<b>7 ISP Debugging Information .....</b>	<b>214</b>
7.1 Overview .....	214
7.2 Function Description.....	214
7.3 API Reference .....	214
7.4 Data Structure.....	217
<b>8 AF .....</b>	<b>220</b>
8.1 Overview .....	220
8.2 Function Description.....	220
8.3 API Reference .....	220
8.4 Data Structures .....	222
<b>9 Error Codes.....</b>	<b>224</b>
<b>10 Proc Debugging Information.....</b>	<b>225</b>
10.1 Overview .....	225
10.2 ISP.....	225





## Figures

<b>Figure 1-1</b> Operating mode of the ISP.....	1
<b>Figure 1-2</b> Design methodology of the ISP firmware.....	2
<b>Figure 1-3</b> File structure of the ISP firmware.....	3
<b>Figure 1-4</b> Internal process of the ISP firmware.....	4
<b>Figure 1-5</b> Architecture of the ISP firmware .....	5
<b>Figure 1-6</b> Flowchart of the ISP firmware.....	6
<b>Figure 2-1</b> Interfaces between the ISP library and the sensor library .....	25
<b>Figure 2-2</b> Interfaces between the ISP library and the AE algorithm library.....	27
<b>Figure 2-3</b> Interfaces between the ISP library and the AWB algorithm library .....	28
<b>Figure 2-4</b> Interfaces between the ISP library and the AF algorithm library.....	29
<b>Figure 3-1</b> Workflow of the AE module .....	63
<b>Figure 3-2</b> 5-segment AE statistics histogram .....	64
<b>Figure 3-3</b> 256-segment AE statistics histogram .....	65
<b>Figure 3-4</b> AE working principle.....	65
<b>Figure 3-5</b> Interfaces between the AE algorithm library and the sensor library .....	67
<b>Figure 3-6</b> AE allocation routes .....	75
<b>Figure 4-1</b> Schematic diagram of the AWB module.....	112
<b>Figure 4-2</b> Interface between the AWB algorithm library and the sensor library .....	114
<b>Figure 5-1</b> CCM .....	146
<b>Figure 6-1</b> Crosstalk removal threshold .....	187
<b>Figure 6-2</b> Offset schematic diagram of the DIS module.....	195
<b>Figure 6-3</b> DIS horizontal offset.....	201
<b>Figure 6-4</b> DIS vertical offset.....	202



## Tables

<b>Table 2-1</b> Bits of u32ModFlag.....	16
<b>Table 5-1</b> Mapping between the values of au8Sat[8] and gains.....	153
<b>Table 6-1</b> Values of u8SharpenAltD[8] based on the sensor gain.....	159
<b>Table 6-2</b> Values of u8SharpenAltUd based on the sensor gain .....	159
<b>Table 6-3</b> Mesh_Scale parameter.....	171
<b>Table 6-4</b> Mapping between the values of u8Strength and gain .....	190
<b>Table 6-5</b> Values of u8SnrThresh[8] based on the sensor gain .....	194
<b>Table 6-6</b> Mapping between the values of u8LumThresh and gain .....	212
<b>Table 6-7</b> Mapping between the values of u8NpOffset and gain.....	212
<b>Table 9-1</b> Error codes for ISP APIs.....	226

# 1 ISP

## 1.1 Overview

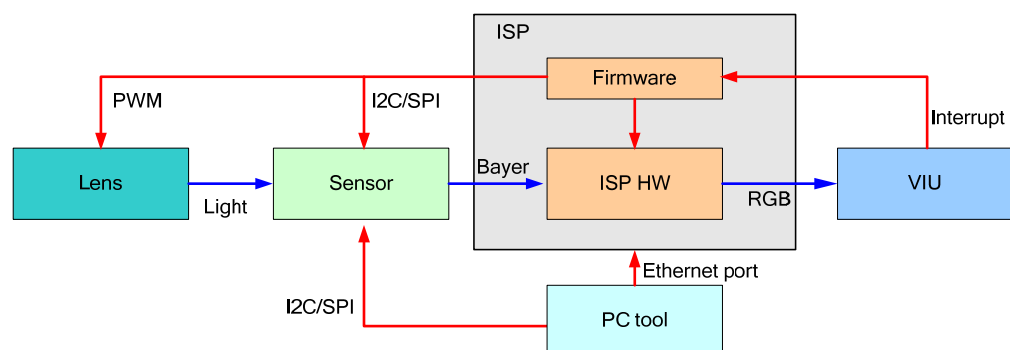
The image signal processor (ISP) processes digital images by using a series of digital image processing algorithms. The ISP supports the following functions: 3A algorithm, defect pixel correction, denoising, highlight compensation, backlight compensation, color enhancement, and lens shading correction. The ISP consists of the logic and firmware running on the logic. This chapter describes the user APIs related to the ISP.

## 1.2 Function Description

Figure 1-1 shows the operating mode of the ISP. The lens projects light signals onto the photosensitive area of the sensor. After photoelectric conversion, the ISP transmits raw Bayer images to the ISP. The ISP processes the images based on related algorithms, and outputs the images in the RGB spatial domain to the backend video capture (VICAP) module. During this process, the ISP controls the lens and sensor by using the firmware, implementing the functions including automatic iris (AI), automatic exposure (AE), and automatic white balance (AWB). The firmware is driven by the interrupts of the VICAP module. The PC tool adjusts the quality of the online images of the ISP over the Ethernet port or serial port.

The logic of the ISP processes images based on algorithms, and provides real-time information about current images. The firmware obtains the image information, recalculates related parameter values such as the exposure time and gain, and estimates the parameter values required by the next frame to control the lens, sensor, and ISP logic. This ensures that the image quality is automatically adjusted.

**Figure 1-1** Operating mode of the ISP

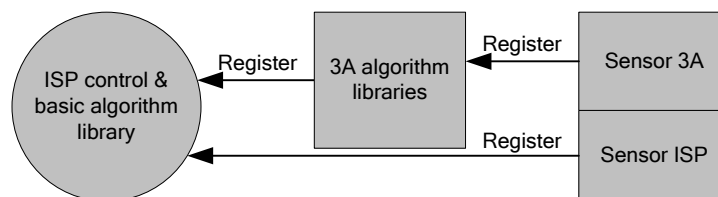


For details about related concepts and functions, see the *Hi3516 Full-HD IP Camera SoC Data Sheet* and *Hi3518 720p IP Camera SoC Data Sheet*.

## 1.2.2 Design Methodology

The ISP firmware consists of the ISP control unit, basic algorithm unit, AE/AWB/AF algorithm libraries (3A algorithm libraries), and sensor library. According to the firmware design methodology, separate 3A algorithm libraries are provided. The ISP control unit schedules the basic algorithm unit and 3A algorithm libraries, and the sensor library registers callback functions with the ISP basic algorithm unit and 3A algorithm libraries to support various sensors. See [Figure 1-2](#).

**Figure 1-2** Design methodology of the ISP firmware

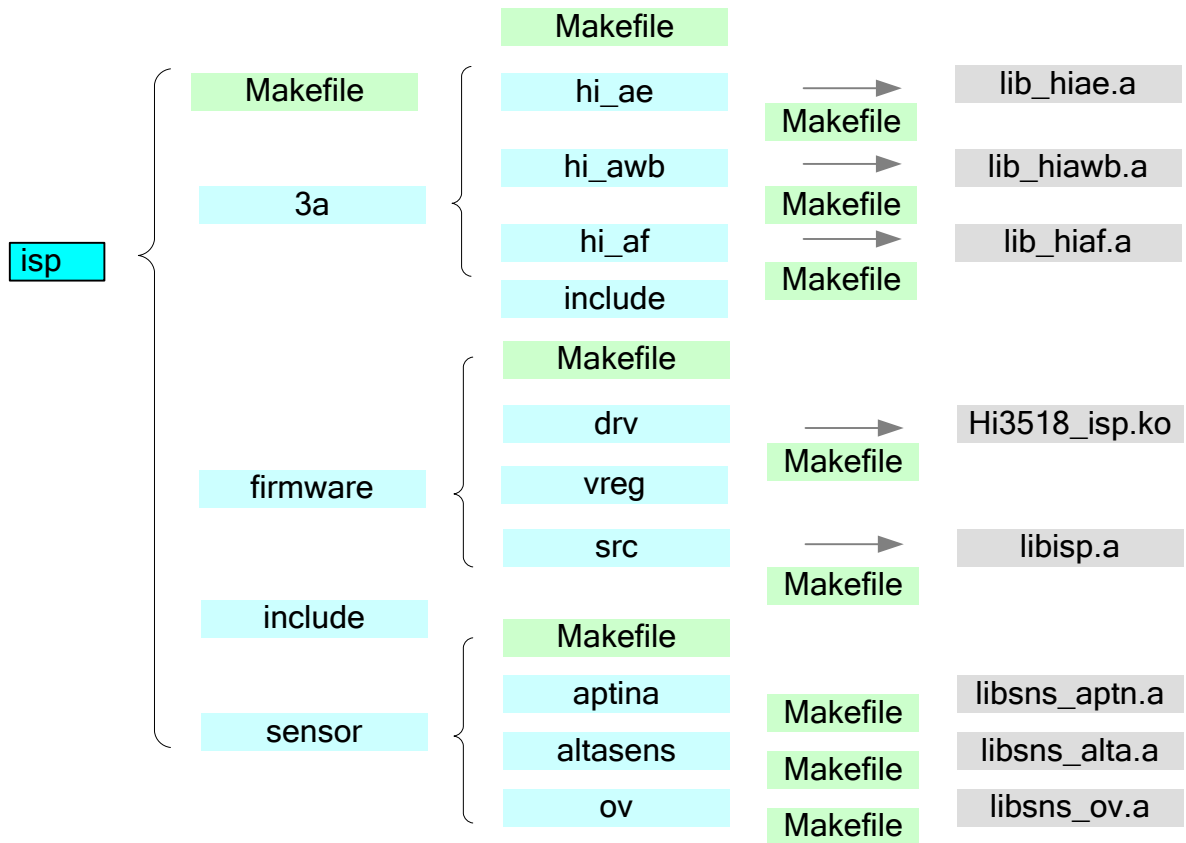


Different sensors register callback control functions with the ISP basic algorithm unit and 3A algorithm libraries. When the ISP control unit schedules the basic algorithm unit and 3A algorithm libraries, the callback functions are called to obtain initialization parameters and control sensors, for example, adjusting the exposure time, analog gain, and digital gain, controlling lens step focus, and rotating the iris.

## 1.2.3 File Structure

As shown in [Figure 1-3](#), the files of the ISP library, 3A algorithm library, and sensor library are stored in different folders. The driver generated in the **drv** folder of the firmware reports the ISP interrupt. The interrupt is used to drive the ISP control unit. The ISP control unit obtains statistics from the driver, schedules the basic algorithm unit and 3A algorithm library, and notifies the driver of the registers to be configured. The **src** folder includes the code of the ISP control unit and basic algorithm unit. The ISP library **libisp.a** is generated after the code in the **src** folder is compiled. The **3a** folder includes the AE/AWB/AF algorithm library. You can develop your own 3A algorithms based on a unified interface. The **sensor** folder stores the drivers of all sensors as open source code. The driver code of each sensor is compiled as a library to facilitate application compilation and connection between the ISP firmware and sensors. You can call related callback functions based on the features of the used sensor.

**Figure 1-3** File structure of the ISP firmware



## 1.2.4 Development Mode

The SDK supports various development modes:

- Uses only the HiSilicon 3A algorithm libraries.  
You need to call the sensor adaptation interfaces provided in the ISP basic algorithm unit and HiSilicon 3A algorithm libraries to adapt to various sensors. The **sensor** folder includes the following two key files:
  - **sensor\_cmos.c**  
This file is used to implement the callback functions required by the ISP. The callback functions include the adaptation algorithms of sensors and vary according to sensors.
  - **sensor\_ctrl.c**  
This file is the underlying driver of the sensor, and is used to read, write to, and initialize sensors. You can develop these two files based on the data sheet of sensors and seek for help from the sensor vendor.
- Develops 3A algorithm libraries based on the 3A algorithm registration interfaces provided in the HiSilicon ISP library. You need to call the sensor adaptation interfaces provided in the ISP basic algorithm unit and your own 3A algorithm libraries to adapt to various sensors.



- Uses both the HiSilicon 3A algorithm libraries and your own 3A algorithm libraries. For example, you can use **lib\_hiae.a** to implement AE, and use your own 3A algorithm libraries to implement AWB.

[Note]

The senior engineers who are familiar with the ISP logic and capable of algorithm development can develop algorithm libraries based on ISP registers.

## 1.2.5 Internal Process

Figure 1-4 shows the internal process of the ISP firmware. The process includes initialization and dynamic adjustment tasks. In the initialization task, the ISP control unit, basic algorithm unit, and 3A algorithm libraries are initialized, and the sensor callback function is called to obtain sensor initialization parameters. In the dynamic adjustment task, the ISP control unit schedules the basic algorithm unit and 3A algorithm libraries to perform real-time calculation and control.

**Figure 1-4** Internal process of the ISP firmware

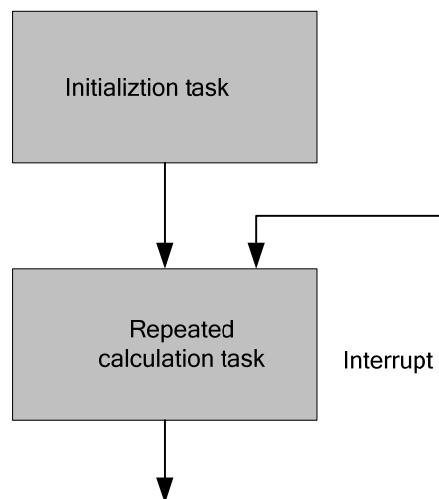
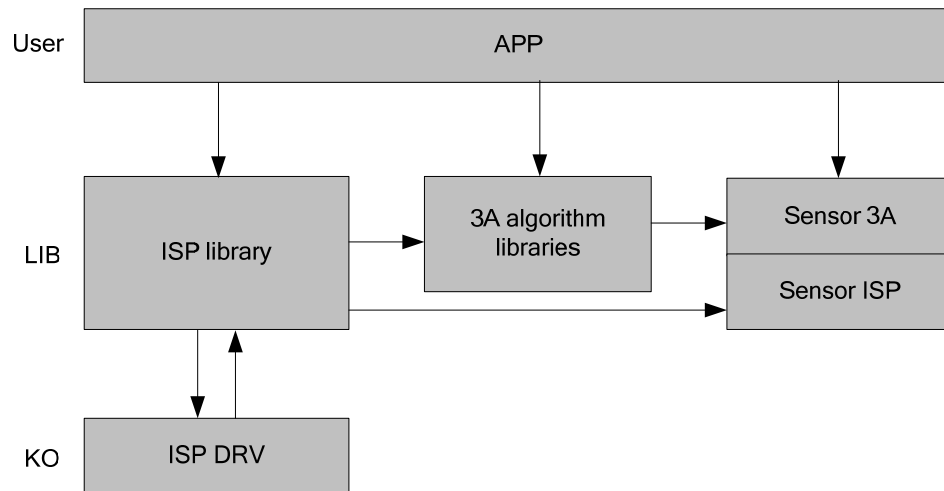


Figure 1-5 shows the architecture of the ISP firmware.

**Figure 1-5** Architecture of the ISP firmware

## 1.2.6 Software Process

As the front-end capture unit, the ISP must work with the VIU. After the ISP is initialized and configured, the interface timings of the VIU must be configured. This ensures that the input timings of different sensors are compatible, and the input timings of the ISP are correct. After timings are configured, the ISP can be started to dynamically adjust the image quality. The VIU captures the output images and stores them in the DDR. Then the images are transmitted for displaying or encoding. [Figure 1-6](#) shows the software process.

The PC tuning tool dynamically adjusts the image quality at the PC end by tuning the values of related parameters such as denoise strength, color conversion matrix, and saturation. If the PC tuning tool is not provided in the product release phase, you can call the image quality adjustment MPI to bring corresponding image effect.

**Figure 1-6** Flowchart of the ISP firmware



After adjusting images, you can save settings by using the configuration file of the PC tuning tool. Then the configured image parameters can be loaded when the ISP is started next time.





# 2 System Control

---

## 2.1 Function Description

The system control part provides the following functions:

- Configures the ISP initialization timing.
- Configures the ISP image attributes.
- Initializes the ISP firmware.
- Runs the ISP firmware.
- Configures ISP modules.

## 2.2 API Reference

The following are system control MPIs:

- [HI\\_MPI\\_ISP\\_SetInputTiming](#): Sets an ISP input timing.
- [HI\\_MPI\\_ISP\\_GetInputTiming](#): Obtains an ISP input timing.
- [HI\\_MPI\\_ISP\\_SetImageAttr](#): Sets the input image attribute.
- [HI\\_MPI\\_ISP\\_GetImageAttr](#): Obtains the input image attribute.
- [HI\\_MPI\\_ISP\\_Init](#): Initializes the ISP firmware.
- [HI\\_MPI\\_ISP\\_Run](#): Runs the ISP firmware.
- [HI\\_MPI\\_ISP\\_Exit](#): Closes the ISP firmware.
- [HI\\_MPI\\_ISP\\_FreezeFmw](#): Freezes the ISP firmware.
- [HI\\_MPI\\_ISP\\_SetModuleControl](#): Controls ISP modules.
- [HI\\_MPI\\_ISP\\_GetModuleControl](#): Obtains the control status of ISP modules.
- [HI\\_MPI\\_ISP\\_SetSlowFrameRate](#): Sets the multiple for decreasing the frame rate of the ISP.
- [HI\\_MPI\\_ISP\\_GetSlowFrameRate](#): Obtains the multiple for decreasing the frame rate of the ISP.
- [HI\\_MPI\\_ISP\\_SetAntiFlickerAttr](#): Sets the anti-flicker frequency of the ISP.
- [HI\\_MPI\\_ISP\\_GetAntiFlickerAttr](#): Obtains the anti-flicker frequency of the ISP.
- [HI\\_MPI\\_ISP\\_GetVDTimeOut](#): Obtains ISP interrupt information.
- [HI\\_MPI\\_ISP\\_SetWdrAttr](#): Obtains the wide dynamic range (WDR) attributes of the ISP.



- [HI\\_MPI\\_ISP\\_GetWdrAttr](#): Sets the WDR attributes of the ISP.
- [HI\\_MPI\\_ISP\\_SensorRegCallBack](#): Registers a sensor.
- [HI\\_MPI\\_ISP\\_AeLibRegCallBack](#): Registers the AE algorithm library.
- [HI\\_MPI\\_ISP\\_AwbLibRegCallBack](#): Registers the AWB algorithm library.
- [HI\\_MPI\\_ISP\\_AfLibRegCallBack](#): Registers the AF algorithm library.
- [HI\\_MPI\\_ISP\\_SetBindAttr](#): Binds the ISP library to 3A algorithm libraries and sensors.
- [HI\\_MPI\\_ISP\\_GetBindAttr](#): Obtains the binding relationship between the ISP library and 3A algorithm libraries /sensors.
- [HI\\_MPI\\_ISP\\_VRegInit](#): Initializes the external registers in a specified address range.
- [HI\\_MPI\\_ISP\\_VRegExit](#): Destroys the external registers in a specified address range.
- [HI\\_MPI\\_ISP\\_GetVRegAddr](#): Obtains the physical addresses for the external registers in a specified address range.

## HI\_MPI\_ISP\_SetInputTiming

[Description]

Sets an ISP input timing.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetInputTiming(const ISP\_INPUT\_TIMING\_S *pstInputTiming);
```

[Parameter]

Parameter	Description	Input/Output
pstInputTiming	Input timing attribute Static attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: [hi\\_comm\\_isp.h](#), [mpi\\_isp.h](#)



- Library file: libisp.a

[Note]

- By calling this MPI, you can obtain valid images required by the ISP from the images input by a sensor. If the output images have black borders, that is, optical black (OB) regions of the sensor, you must call this MPI to crop the black borders. If the images input by the sensor are valid images, set the interface mode to ISP\_WIND\_NONE. This indicates that the configuration of the cropping area is invalid.
- Before calling this MPI, you must initialize the ISP.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetInputTiming](#)

## HI\_MPI\_ISP\_GetInputTiming

[Description]

Obtains an ISP input timing.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetInputTiming(ISP\_INPUT\_TIMING\_S *pstInputTiming);
```

[Parameter]

Parameter	Description	Input/Output
pstInputTiming	Input timing attribute Static attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h



- Library file: libisp.a

[Note]

Before calling this MPI, you must set a timing.

[Example]

None

[See Also]

HI\_MPI\_ISP\_SetInputTiming

## HI\_MPI\_ISP\_SetImageAttr

[Description]

Sets the input image attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetImageAttr(const ISP_IMAGE_ATTR_S *pstImageAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstImageAttr	Input image attribute Static attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.
HI_ERR_ISP_ILLEGAL_PARAM	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]



- The image attribute is the capture attribute of a sensor.
- Before calling this MPI, you must initialize the ISP.

[Example]

None

[See Also]

HI\_MPI\_ISP\_GetImageAttr

## HI\_MPI\_ISP\_GetImageAttr

[Description]

Obtains the input image attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetImageAttr(ISP\_IMAGE\_ATTR\_S *pstImageAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstImageAttr	Input image attribute Static attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

Before calling this MPI, you must set the input image attribute.

[Example]

None



[See Also]

[HI\\_MPI\\_ISP\\_SetImageAttr](#)

## HI\_MPI\_ISP\_Init

[Description]

Initializes the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_Init(HI_VOID);
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_SNS_UNREGISTER</a>	The sensor is not registered.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

Before initializing the firmware, you must initialize the sensor and register related callback functions.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_Exit](#)

## HI\_MPI\_ISP\_Run

[Description]

Runs the ISP firmware.

[Syntax]



```
HI_S32 HI_MPI_ISP_Run (HI_VOID) ;
```

[Parameter]

None

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_SNS_UNREGISTER</a>	The sensor is not registered.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- Before running the firmware, you must initialize the sensor and register related callback functions.
- Before running the firmware, you must configure the timing and image attribute.
- This MPI is a block interface. You are advised to call it by using a real-time thread.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_Init](#)

## HI\_MPI\_ISP\_Exit

[Description]

Closes the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_Exit (HI_VOID) ;
```

[Parameter]

None

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]


None

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_Init](#)

## HI\_MPI\_ISP\_FreezeFmw

[Description]

Freezes the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_FreezeFmw(HI_BOOL bFreeze);
```

[Parameter]

Parameter	Description	Input/Output
bFreeze	ISP firmware freeze enable	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.





[Error Code]



None

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

When **bFreeze** is **True**, the 3A algorithm, DRC algorithm, and noise reduction (NR) algorithm of the ISP firmware are disabled. The sensor registers remain the values obtained before the firmware is frozen until **bFreeze** is **False** by calling this MPI.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetModuleControl

[Description]

Controls ISP modules.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetModuleControl(HI_U32 u32ModFlag);
```

[Parameter]

Parameter	Description	Input/Output
u32ModFlag	Module control value	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]




None

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- This MPI can be used to enable or disable the modules of the ISP.
- Each bit of **u32ModFlag** controls one module of the ISP. The value **0** indicates that the corresponding module is enabled, and the value **1** indicates that the corresponding module is disabled. [Table 2-1](#) describes the bits of **u32ModFlag**.

**Table 2-1** Bits of u32ModFlag

Bit	Description
[31:27]	Reserved
[26]	The output end directly connects to the input end.
[25:24]	00: All requests are processed. 01: All ISP processing is bypassed (the VI port still connects to the VO end), and the sensor raw data is output. 01: All ISP processing is bypassed (the VI port still connects to the VO end), and the sensor raw data of the MSBs of channel 1 and channel 2 is output. 11: The output end connects to GND.
[23:15]	Reserved
[14]	Gamma table bypass
[13]	Color matrix bypass
[12]	Demosaic module bypass (raw data output)
[11]	DRC bypass
[10]	Reserved
[9]	Shading correction bypass
[8:7]	Reserved
[6]	Black level and gain bypass
[5]	Denoise bypass
[4]	Defect pixel correction bypass
[3]	Green balance bypass
[2]	WDR compression frontend lookup bypass



Bit	Description
[1]	Frontend black level adjustment bypass
[0]	Video test generator bypass

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetModuleControl](#)

## HI\_MPI\_ISP\_GetModuleControl

[Description]

Obtains the control status of ISP modules.

[Syntax]

```
HI_MPI_ISP_GetModuleControl (HI_U32 *pu32ModFlag);
```

[Parameter]

Parameter	Description	Input/Output
pu32ModFlag	Module control value	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None



[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetModuleControl](#)

## HI\_MPI\_ISP\_SetSlowFrameRate

[Description]

Sets the multiple for decreasing the frame rate of the ISP.

[Syntax]

```
HI_MPI_ISP_SetSlowFrameRate(HI_U8 u8Value);
```

[Parameter]

Parameter	Description	Input/Output
u8Value	Parameter for decreasing the frame rate	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

The current frame rate is calculated as follows: Current frame rate = Original frame rate/(u8Value >> 4). The minimum value of u8Value is 0x10. If **u8Value** is **0x10**, the current frame rate is the same as the original frame rate; if **u8Value** is **0x20**, the current frame rate is half of the original frame rate; if **u8Value** is **0x30**, the current frame rate is 1/3 of the original frame rate, and so on.

[Example]

None



[See Also]

[HI\\_MPI\\_ISP\\_GetSlowFrameRate](#)

## HI\_MPI\_ISP\_GetSlowFrameRate

[Description]

Obtains the multiple for decreasing the frame rate of the ISP.

[Syntax]

```
HI_MPI_ISP_GetSlowFrameRate (HI_U8 *pu8Value);
```

[Parameter]

Parameter	Description	Input/Output
pu8Value	Parameter for decreasing the frame rate	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetSlowFrameRate](#)

## HI\_MPI\_ISP\_SetAntiFlickerAttr

[Description]



Sets the anti-flicker frequency of the ISP.

[Syntax]

```
HI_MPI_ISP_SetAntiFlickerAttr(const ISP\_ANTIFLICKER\_S *pstAntiflicker);
```

[Parameter]

Parameter	Description	Input/Output
pstAntiflicker	Anti-flicker frequency	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

If the power supply frequency is 50 Hz, the anti-flicker frequency is 50. If the power supply frequency is 60 Hz, the anti-flicker frequency is 60.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetAntiFlickerAttr](#)

## HI\_MPI\_ISP\_GetAntiFlickerAttr

[Description]

Obtains the anti-flicker frequency of the ISP.

[Syntax]

```
HI_MPI_ISP_GetAntiFlickerAttr(ISP\_ANTIFLICKER\_S *pstAntiflicker);
```



[Parameter]

Parameter	Description	Input/Output
pstAntiFlicker	Anti-flicker frequency	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetAntiFlickerAttr](#)

## HI\_MPI\_ISP\_GetVDTimeOut

[Description]

Obtains ISP interrupt information.

[Syntax]

```
HI_MPI_ISP_GetVDTimeOut(ISP\_VD\_INFO\_S *pstIspVdInfo, HI_U32 u32MilliSec);
```

[Parameter]

Parameter	Description	Input/Output
pstIspVdInfo	Pointer to the structure of the ISP frame information	Output
u32MilliSec	Timeout period, in the unit of ms	Input



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- This MPI is used to obtain the information about the interrupts generated by the ISP, including whether interrupts are generated and the information about the current ISP frame when interrupts are generated.
- The **u32MilliSec** parameter indicates the timeout period and its unit is ms. The MPI is returned if no ISP interrupts are obtained within the period defined by **u32MilliSec**. If **u32MilliSec** is set to **0**, the block mode is used. In this case, the MPI is returned only after ISP interrupts are obtained.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetWdrAttr

[Description]

Obtains the WDR attributes of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWdrAttr(const ISP_WDR_ATTR_S *pstWdrAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstWdrAttr	WDR attributes	Input





[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI\_MPI\_ISP\_GetWdrAttr

## HI\_MPI\_ISP\_GetWdrAttr

[Description]

Sets the WDR attributes of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWdrAttr (ISP_WDR_ATTR_S *pstWdrAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstWdrAttr	WDR attributes	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h



- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI\_MPI\_ISP\_SetWdrAttr

## HI\_MPI\_ISP\_SensorRegCallBack

[Description]

Registers a sensor.

[Syntax]

```
HI_S32 HI_MPI_ISP_SensorRegCallBack(SENSOR_ID SensorId,  
ISP_SENSOR_REGISTER_S *pstRegister);
```

[Parameter]

Parameter	Description	Input/Output
SensorId	ID of the sensor that is registered with the ISP library	Input
pstRegister	Pointer to the data structure for registering a sensor	Input

[Return Value]

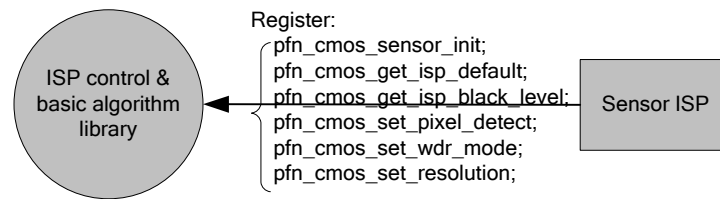
Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- **SensorId** can be configured in the sensor library and is used to check whether the sensor registered with the ISP library is the same as that registered with the 3A algorithm library.
- The ISP obtains differentiated initialization parameters and controls sensors by using the callback interfaces registered by sensors.

**Figure 2-1** Interfaces between the ISP library and the sensor library


#### [Example]

```
ISP_SENSOR_REGISTER_S stIspRegister;
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIspRegister.stSnsExp;

memset(pstSensorExpFunc, 0, sizeof(ISP_SENSOR_EXP_FUNC_S));
pstSensorExpFunc->pfn_cmos_sensor_init = sensor_init;
pstSensorExpFunc->pfn_cmos_get_isp_default = cmos_get_isp_default;
pstSensorExpFunc->pfn_cmos_get_isp_black_level = cmos_get_isp_black_level;
pstSensorExpFunc->pfn_cmos_set_pixel_detect = cmos_set_pixel_detect;
pstSensorExpFunc->pfn_cmos_set_wdr_mode = cmos_set_wdr_mode;
s32Ret = HI_MPI_ISP_SensorRegCallBack(IMX104_ID, &stIspRegister);
if (s32Ret)
{
    printf("sensor register callback function failed!\n");
    return s32Ret;
}
```

#### [See Also]

None [HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_AeLibRegCallBack

#### [Description]

Registers the AE algorithm library.

#### [Syntax]

```
HI_S32 HI_MPI_ISP_AeLibRegCallBack(ALG_LIB_S *pstAeLib,
    ISP_AE_REGISTER_S *pstRegister);
```

#### [Parameter]

Parameter	Description	Input/Output
pstAeLib	Pointer to the data structure of the AE algorithm library	Input
pstRegister	Pointer to the data structure for registering the AE algorithm library	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

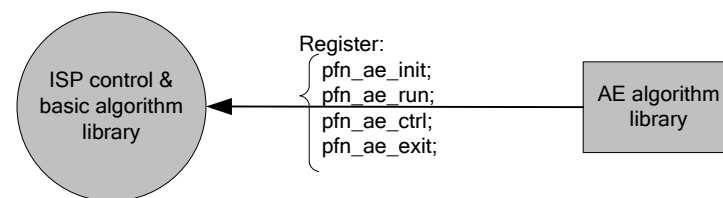
### [Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

### [Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AE algorithm library.

**Figure 2-2** Interfaces between the ISP library and the AE algorithm library



### [Example]

```

ISP_AE_REGISTER_S stRegister;

HI_S32 s32Ret = HI_SUCCESS;

AE_CHECK_POINTER(pstAeLib);
AE_CHECK_HANDLE_ID(pstAeLib->s32Id);
AE_CHECK_LIB_NAME(pstAeLib->acLibName);

stRegister.stAeExpFunc.pfn_ae_init = AeInit;
stRegister.stAeExpFunc.pfn_ae_run = AeRun;
stRegister.stAeExpFunc.pfn_ae_ctrl = AeCtrl;
stRegister.stAeExpFunc.pfn_ae_exit = AeExit;
s32Ret = HI_MPI_ISP_AeLibRegCallBack(pstAeLib, &stRegister);
if (HI_SUCCESS != s32Ret)
{
    printf("Hi_ae register failed!\n");
}
  
```



}

[See Also]

None [HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_AwbLibRegCallBack

[Description]

Registers the AWB algorithm library.

[Syntax]

```
HI_S32 HI_MPI_ISP_AwbLibRegCallBack(ALG_LIB_S *pstAwbLib,  
                                     ISP_AWB_REGISTER_S *pstRegister);
```

[Parameter]

Parameter	Description	Input/Output
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input
pstRegister	Pointer to the data structure for registering the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

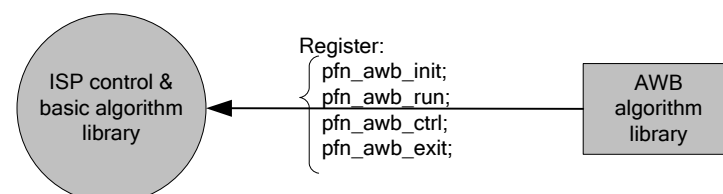
[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AWB algorithm library.

**Figure 2-3** Interfaces between the ISP library and the AWB algorithm library





[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_AfLibRegCallBack

[Description]

Registers the AF algorithm library.

[Syntax]

```
HI_S32 HI_MPI_ISP_AfLibRegCallBack(ALG_LIB_S *pstAfLib,  
    ISP_AF_REGISTER_S *pstRegister);
```

[Parameter]

Parameter	Description	Input/Output
pstAfLib	Pointer to the data structure of the AF algorithm library	Input
pstRegister	Pointer to the data structure for registering the AF algorithm library	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

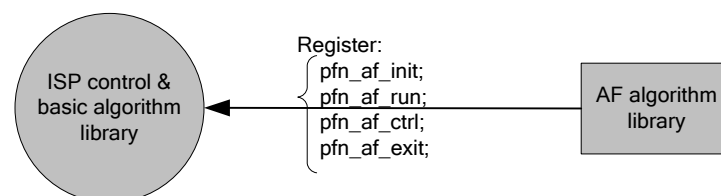
[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AF algorithm library.

**Figure 2-4** Interfaces between the ISP library and the AF algorithm library





[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetBindAttr

[Description]

Binds the ISP library to 3A algorithm libraries and sensors.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetBindAttr(const ISP_BIND_ATTR_S *pstBindAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstBindAttr	Pointer to the binding data structure	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI is called only when you register multiple AE, AWB, and AF libraries and want to switch the libraries. If multiple AE, AWB, and AF libraries are registered, the last registered AE, AWB, and AF libraries are bound by default.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetBindAttr

[Description]



Obtains the binding relationship between the ISP library and 3A algorithm libraries/sensors.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetBindAttr(ISP_BIND_ATTR_S *pstBindAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstBindAttr	Pointer to the binding data structure	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_VRegInit

[Description]

Initializes the external registers in a specified address range.

[Syntax]

```
HI_S32 HI_MPI_ISP_VRegInit(HI_U32 u32BaseAddr, HI_U32 u32Size);
```

[Parameter]

Parameter	Description	Input/Output
u32BaseAddr	Base addresses for external registers	Input
u32Size	Size of external registers	Input





[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- External registers are used in the ISP library and algorithm libraries. Similar to the shared memory mechanism, external registers are used to control the internal status of the ISP library and algorithm libraries under multiple threads.
- In the external register mechanism, a range of virtual register addresses (**u32BaseAddr** to **u32BaseAddr + u32Size**) are defined and a range of physical addresses with the same size are allocated. If the virtual register addresses are the same, the data at the corresponding physical addresses can be read or written through any thread. In **isp/firmware/vreg** of the ISP firmware, the MPIs such as **HI\_MPI\_VRegInit**, **HI\_MPI\_VRegExit**, and **HI\_MPI\_GetVRegAddr** are called to encapsulate the external registers. The following are the defined mappings between address ranges and external registers:
  - The address range 0–0x10000 corresponds to the actual hardware registers of the ISP.
  - The address range 0x10000–0x20000 corresponds to the ISP external registers.
  - The address range 0x20000–0x30000 corresponds to the external registers of a maximum of 16 AE algorithm libraries.
  - The address range 0x30000–0x40000 corresponds to the external registers of a maximum of 16 AWB algorithm libraries.
  - The address range 0x40000–0x50000 corresponds to the external registers of a maximum of 16 AF algorithm libraries.The external registers in each address range need to be separately created and initialized by calling the encapsulated MPIs through the ISP library or algorithm libraries. You can also implement the multi-thread control mechanism or customize external registers.
- When **HI\_MPI\_ISP\_VRegInit** is called, a range of physical addresses are allocated, and the mapping between external register addresses and actual physical addresses is created and recorded. The address range 0–0x10000 corresponds to the physical addresses for actual hardware registers by default.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_VRegExit

[Description]



Destroys the external registers in a specified address range.

[Syntax]

```
HI_S32 HI_MPI_ISP_VRegExit (HI_U32 u32BaseAddr);
```

[Parameter]

Parameter	Description	Input/Output
u32BaseAddr	Base addresses for external registers	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

When this MPI is called, a range of physical addresses are destroyed, and the recorded mapping between external register addresses and actual physical addresses is deleted.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetVRegAddr

[Description]

Obtains the physical addresses for the external registers in a specified address range.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetVRegAddr (HI_U32 u32BaseAddr, HI_U32 *pu32PhyAddr);
```

[Parameter]

Parameter	Description	Input/Output
u32BaseAddr	Base addresses for external registers	Input
pu32PhyAddr	Pointer to the physical addresses for external registers	Output



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- When this MPI is called, actual physical addresses are obtained based on external register addresses.
- The addresses are base addresses without offset addresses.

[Example]

None

[See Also]

None

## 2.3 Data Structures

The following are ISP data structures:

- [ISP\\_WIND\\_MODE\\_E](#): Defines the type of the ISP input timing window.
- [ISP\\_INPUT\\_TIMING\\_S](#): Defines the attribute of the ISP input timing window.
- [ISP\\_BAYER\\_FORMAT\\_E](#): Defines the format of the input Bayer image.
- [ISP\\_IMAGE\\_ATTR\\_S](#): Defines the attribute of ISP input image.
- [ISP\\_ANTIFLICKER\\_MODE\\_E](#): Defines the anti-flicker mode of the ISP.
- [ISP\\_ANTIFLICKER\\_S](#): Defines the anti-flicker attribute of the ISP.
- [ISP\\_VD\\_INFO\\_S](#): Defines ISP frame information.
- [ISP\\_WDR\\_ATTR\\_S](#): Defines the WDR attributes of the ISP.
- [ISP\\_SENSOR\\_REGISTER\\_S](#): Defines sensor registration information.
- [ISP\\_SENSOR\\_EXP\\_FUNC\\_S](#): Defines the sensor callback function.
- [ISP\\_CMOS\\_DEFAULT\\_S](#): Defines the initialization parameter for the ISP basic algorithm unit.
- [ISP\\_CMOS\\_BLACK\\_LEVEL\\_S](#): Defines the sensor black level.
- [ALG\\_LIB\\_S](#): Defines AE, AWB, and AF algorithm libraries.
- [ISP\\_BIND\\_ATTR\\_S](#): Defines the binding relationship between the ISP library and sensors/3A algorithm libraries.
- [ISP\\_AE\\_REGISTER\\_S](#): Defines AE registration information.



- [ISP\\_AE\\_EXP\\_FUNC\\_S](#): Defines the AE callback function.
- [ISP\\_AE\\_PARAM\\_S](#): Defines the initialization parameter provided by the ISP for the AE algorithm library.
- [ISP\\_AE\\_INFO\\_S](#): Defines the statistics provided by the ISP for the AE algorithm library.
- [ISP\\_AE\\_RESULT\\_S](#): Defines the results returned by the AE algorithm library to the ISP for configuring registers.
- [ISP\\_AWB\\_REGISTER\\_S](#): Defines AWB registration information.
- [ISP\\_AWB\\_EXP\\_FUNC\\_S](#): Defines the AWB callback function.
- [ISP\\_AWB\\_PARAM\\_S](#): Defines the initialization parameter provided by the ISP for the AWB algorithm library.
- [ISP\\_AWB\\_INFO\\_S](#): Defines the statistics provided by the ISP for the AWB algorithm library.
- [ISP\\_AWB\\_RESULT\\_S](#): Defines the results returned by the AWB algorithm library to the ISP for configuring registers.
- [ISP\\_AF\\_REGISTER\\_S](#): Defines AF registration information.
- [ISP\\_AF\\_EXP\\_FUNC\\_S](#): Defines the AF callback function.
- [ISP\\_AF\\_PARAM\\_S](#): Defines the initialization parameter provided by the ISP for the AF algorithm library.
- [ISP\\_AF\\_INFO\\_S](#): Defines the statistics provided by the ISP for the AF algorithm library.
- [ISP\\_AF\\_RESULT\\_S](#): Defines the results returned by the AF algorithm library to the ISP for configuring registers.

## ISP\_WIND\_MODE\_E

[Description]

Defines the type of the ISP input timing window.

[Syntax]

```
typedef enum hiISP_WIND_MODE_E
{
    ISP_WIND_NONE        = 0,
    ISP_WIND_HOR          = 1,
    ISP_WIND_VER          = 2,
    ISP_WIND_ALL          = 3,
    ISP_WIND_BUTT
} ISP_WIND_MODE_E;
```

[Member]

Member	Description
ISP_WIND_NONE	Window that cannot be cropped
ISP_WIND_HOR	Window that can be cropped horizontally only
ISP_WIND_VER	Window that can be cropped vertically only



Member	Description
ISP_WIND_ALL	Window that can be cropped horizontally and vertically

[Note]

None

[See Also]

[ISP\\_INPUT\\_TIMING\\_S](#)

## ISP\_INPUT\_TIMING\_S

[Description]

Defines the attribute of the ISP input timing window.

[Syntax]

```
typedef struct hiISP_INPUT_TIMING_S
{
    ISP_WIND_MODE_E enWndMode;
    HI_U16 u16HorWndStart;
    HI_U16 u16HorWndLength;
    HI_U16 u16VerWndStart;
    HI_U16 u16VerWndLength;

} ISP_INPUT_TIMING_S;
```

[Member]

Member	Description
enWndMode	Window crop mode
u16HorWndStart	Horizontal start position Value range: [0x0, 0x780]
u16HorWndLength	Horizontal window length Value range: [0x0, 0x780]
u16VerWndStart	Vertical start position Value range: [0x0, 0x4B0]
u16VerWndLength	Vertical window height Value range: [0x0, 0x4B0]

[Note]

If an image output from a sensor has OB regions, you must call HI\_MPI\_ISP\_SetInputTiming to crop the OB regions.



[See Also]

[ISP\\_WIND\\_MODE\\_E](#)

## ISP\_BAYER\_FORMAT\_E

[Description]

Defines the format of the input Bayer image.

[Syntax]

```
typedef enum hiISP_BAYER_FORMAT_E
{
    BAYER_RGGB = 0,
    BAYER_GRBG = 1,
    BAYER_GBRG = 2,
    BAYER_BGGR = 3,
    BAYER_BUTT

} ISP_BAYER_FORMAT_E;
```

[Member]

Member	Description
BAYER_RGGB	RGGB format
BAYER_GRBG	GRGB format
BAYER_GBRG	GBRG format
BAYER_BGGR	BGGR format

[Note]

You can obtain the used format from the data sheet related to the sensor.

[See Also]

[ISP\\_IMAGE\\_ATTR\\_S](#)

## ISP\_IMAGE\_ATTR\_S

[Description]

Defines the attribute of ISP input image.

[Syntax]

```
typedef struct hiISP_IMAGE_ATTR_S
{
    HI_U16 u16Width;
    HI_U16 u16Height;
```



```
HI_U16 u16FrameRate;  
ISP\_BAYER\_FORMAT\_E enBayer;  
  
} ISP_IMAGE_ATTR_S;
```

[Member]

Member	Description
u16Width	Input image width Value range: [0x0, 0x780]
u16Height	Input image height Value range: [0x0, 0x4B0]
u16FrameRate	Frame rate of the input image Value range: [0x0, 0xFF]
enBayer	Bayer data format

[Note]

None

[See Also]

[ISP\\_BAYER\\_FORMAT\\_E](#)

## ISP\_ANTIFLICKER\_MODE\_E

[Description]

Defines the anti-flicker mode of the ISP.

[Syntax]

```
typedef enum hiISP_ANTIFLICKER_MODE_E  
{  
    ISP_ANTIFLICKER_MODE_0 = 0x0,  
    ISP_ANTIFLICKER_MODE_1 = 0x1,  
    ISP_ANTIFLICKER_MODE_BUTT  
}ISP_ANTIFLICKER_MODE_E;
```

[Member]

Member	Description
ISP_ANTIFLICKER_MODE_0	Anti-flicker mode 0
ISP_ANTIFLICKER_MODE_1	Anti-flicker mode 1

[Note]



- **ISP\_ANTIFLICKER\_MODE\_0** is anti-flicker mode 0. The exposure time can be adjusted based on luminance. The minimum exposure time is fixed at 1/120s (60 Hz) or 1/100s (50 Hz).
  - When there are lights, the exposure time can match the light source frequency, which avoids picture flicker.
  - In high-luminance environments, the higher luminance indicates shorter exposure time. However, the minimum exposure time in anti-flicker mode 0 does not match the light source frequency, which results in overexposure.
- **ISP\_ANTIFLICKER\_MODE\_1** is anti-flicker mode 1. The exposure time can be adjusted based on the luminance. The minimum exposure time can be the minimum exposure time of the sensor. Anti-flicker mode 1 differs from anti-flicker mode 0 in high-luminance environments.
  - In high-luminance environments, the minimum exposure time can be the minimum exposure time of the sensor. This avoids overexposure, but anti-flicker does not work.

[See Also]

None

## ISP\_ANTIFLICKER\_S

[Description]

Defines the anti-flicker attribute of the ISP.

[Syntax]

```
typedef struct hiISP_ANTIFLICKER_S
{
    HI_BOOL bEnable;
    HI_U8 u8Frequency;
    ISP_ANTIFLICKER_MODE_E enMode;
} ISP_ANTIFLICKER_S;
```

[Member]

Member	Description
bEnable	If <b>bEnable</b> is <b>HI_TRUE</b> , anti-flicker is enabled. If <b>bEnable</b> is <b>HI_FALSE</b> , anti-flicker is disabled.
u8Frequency	Anti-flicker frequency
enMode	Anti-flicker mode

[Note]

None

[See Also]

None





## ISP\_VD\_INFO\_S

### [Description]

Defines ISP frame information.

### [Syntax]

```
typedef struct hiISP_VD_INFO_S
{
    HI_U32 u32Reserved;
} ISP_VD_INFO_S;
```

### [Member]

Member	Description
u32Reserved	Reserved bytes

### [Note]

This data structure has only a reserved variable. That is, no ISP frame information is provided currently.

### [See Also]

None

## ISP\_WDR\_ATTR\_S

### [Description]

Defines the WDR attributes of the ISP.

### [Syntax]

```
typedef struct hiISP_WDR_ATTR_S
{
    ISP_WDR_MODE_E enWdrMode;
} ISP_WDR_ATTR_S;
```

### [Member]

Member		Description
enWdrMode	ISP_SENSOR_LINEAR_MODE	Linear mode
	ISP_SENSOR_WDR_MODE	WDR mode

### [Note]

None

### [See Also]



None

## ISP\_SENSOR\_REGISTER\_S

[Description]

Defines sensor registration information.

[Syntax]

```
typedef struct hiISP_SENSOR_REGISTER_S
{
    ISP_SENSOR_EXP_FUNC_S stSnsExp;
} ISP_SENSOR_REGISTER_S;
```

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]

ISP\_SENSOR\_EXP\_FUNC\_S

## ISP\_SENSOR\_EXP\_FUNC\_S

[Description]

Defines the sensor callback function.

[Syntax]

```
typedef struct hiISP_SENSOR_EXP_FUNC_S
{
    HI_VOID(*pfn_cmos_sensor_init)(HI_VOID);

    HI_U32(*pfn_cmos_get_isp_default)(ISP_CMOS_DEFAULT_S *pstDef);

    HI_U32(*pfn_cmos_get_isp_black_level)(ISP_CMOS_BLACK_LEVEL_S
    *pstBlackLevel);

    HI_VOID(*pfn_cmos_set_pixel_detect)(HI_BOOL bEnable);

    HI_VOID(*pfn_cmos_set_wdr_mode)(HI_U8 u8Mode);

    HI_VOID(*pfn_cmos_set_resolution)(HI_U32 u32ResolutionMode);
} ISP_SENSOR_EXP_FUNC_S;
```

[Member]



Member	Description
pfn_cmos_sensor_init	Pointer to the callback function for initializing sensors
pfn_cmos_get_isp_default	Pointer to the callback function for obtaining the initial value of the ISP basic algorithm
pfn_cmos_get_isp_black_level	Pointer to the callback function for obtaining the sensor black level
pfn_cmos_set_pixel_detect	Pointer to the callback function for enabling or disabling defect pixel correction
pfn_cmos_set_wdr_mode	Pointer to the callback function for switching between the WDR mode and linear mode
pfn_cmos_set_resolution	Pointer to the callback function for setting the resolution

[Note]

- **pfn\_cmos\_get\_isp\_black\_level** is provided, because the black level of some sensors varies according to the sensor gain.
- If a callback function pointer is not required, set it to **NULL**. For example, **pfn\_cmos\_set\_resolution** needs to set to **NULL** if a sensor that does not allow you to switch the resolution is used.

[See Also]

ISP\_SENSOR\_REGISTER\_S

## ISP\_CMOS\_DEFAULT\_S

[Description]

Defines the initialization parameter for the ISP basic algorithm unit.

[Syntax]

```
typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_COMM_S      stComm;

    ISP_CMOS_DENOISE_S    stDenoise;

    ISP_CMOS_DRC_S        stDrc;

    ISP_CMOS_AGC_TABLE_S  stAgcTbl;

    ISP_CMOS_NOISE_TABLE_S stNoiseTbl;

    ISP_CMOS_DEMOSAIC_S    stDemosaic;

    ISP_CMOS_GAMMAFE_S     stGammafe;

    ISP_CMOS_SHADING_S     stShading;
} ISP_CMOS_DEFAULT_S;
```



[Member]

Member	Sub Member	Description
stComm	u8Rggb	RGrGbB output sequence of the sensor. The value range is [0, 3].
	u8BalanceFe	The default value is 0x1 is recommended. You are advised not to change the default value.
stDenoise	u8SinterThresh	The default value is 0x15 is recommended. You are advised not to change the default value.
	u8NoiseProfile	Noise profile. The default value 0 indicates the default ISP noise profile. You are advised to use the default value 0.
	u16Nr0	The default value is 0x0. You are advised not to change the default value.
	u16Nr1	The default value is 0x0. You are advised not to change the default value.
stDrc	u8DrcBlack	The default value is 0x0. You are advised not to change the default value.
	u8DrcVs	The default value is 0x04 in linear mode or 0x08 in WDR mode. You are advised not to change the default value.
	u8DrcVi	The default value is 0x08 in linear mode or 0x01 in WDR mode. You are advised not to change the default value.
	u8DrcSm	The default value is 0xA0 in linear mode or 0x3C in WDR mode. You are advised not to change the default value.
	u16DrcWl	The default value is 0x4FF in linear mode or 0xFFF in WDR mode. You are advised not to change the default value.
stAgcTbl	bValid	Data validity identifier of the data structure. The value is 0 or 1.
	au8SharpenAltD	Interpolation array for dynamically adjusting the sharpness of the large edges of images based on the gain. The value range is [0, 255].
	au8SharpenAltUd	Interpolation array for dynamically adjusting the sharpness of the small textures of images based on the gain. The value range is [0, 255].
	au8SnrThresh	Interpolation array for dynamically setting the image denoising strength based on the



Member	Sub Member	Description
		gain. The value range is [0, 255].
	au8DemosaicLumThresh	Array for setting the luminance threshold for the sharpness of large edges of images. The default value is recommended, and the value range is [0, 255].
	au8DemosaicNpOffset	Array for setting image noise parameters. The default value is recommended, and the value range is [0, 255].
	au8GeStrength	Array for setting the parameters of the green equalization parameter. The default value is recommended, and the value range is [0, 255].
stNoiseTbl	bValid	Data validity identifier of the data structure. The value is 0 or 1.
	au8NoiseProfileWeightLut	Array for setting the noise profile related to sensor features. The array value is used as the input of the denoise module. The default value is recommended, and the value range is [0, 255].
	au8DemosaicWeightLut	Array for setting the noise profile related to sensor features. The array value is used as the input of the demosaic module. The default value is recommended, and the value range is [0, 255].
stDemosaic	bValid	Data validity identifier of the data structure. The value is 0 or 1.
	u8VhSlope	Vertical/Horizontal slope threshold. The default value is recommended, and the value range is [0, 255].
	u8AaSlope	Angle slope threshold. The default value is recommended, and the value range is [0, 255].
	u8VaSlope	VH-AA slope threshold. The default value is recommended, and the value range is [0, 255].
	u8UuSlope	Undefined slope threshold. The default value is recommended, and the value range is [0, 255].
	u8SatSlope	Saturation slope threshold. The default value is recommended, and the value range is [0, 255].
	u8AcSlope	High-frequency component filtering slope threshold. The default value is recommended, and the value range is [0,



Member	Sub Member	Description
		255].
	u16VhThresh	Vertical/Horizontal threshold. The default value is recommended, and the value range is [0, 0xFFFF].
	u16AaThresh	Angle threshold. The default value is recommended, and the value range is [0, 0xFFFF].
	u16VaThresh	VA threshold. The default value is recommended, and the value range is [0, 0xFFFF].
	u16UuThresh	Undefined threshold. The default value is recommended, and the value range is [0, 0xFFFF].
	u16SatThresh	Saturation threshold. The default value is recommended, and the value range is [0, 0xFFFF].
	u16AcThresh	High-frequency component filtering threshold. The default value is recommended, and the value range is [0, 0xFFFF].
stGammafe	bValid	Data validity identifier of the data structure. The value is 0 or 1. This member needs to be configured when the sensor supports the WDR mode.
	au16Gammafe	GammaFe table. The value range is [0, 0xFFFF].
stShading	bValid	Data validity identifier of the data structure. The value is 0 or 1. If shading correction is not required, you can set this member to make data invalid.
	u16RCenterX	Horizontal coordinate of the R component center. The value range is [0x0, 0xFFFF].
	u16RCenterY	Vertical coordinate of the R component center. The value range is [0x0, 0xFFFF].
	u16GCenterX	Horizontal coordinate of the G component center. The value range is [0x0, 0xFFFF].
	u16GCenterY	Vertical coordinate of the G component center. The value range is [0x0, 0xFFFF].



Member	Sub Member	Description
	u16BCenterX	Horizontal coordinate of the B component center. The value range is [0x0, 0xFFFF].
	u16BCenterY	Vertical coordinate of the B component center. The value range is [0x0, 0xFFFF].
	au16RShadingTbl	Correction table of the R component. The value range is [0x0, 0xFFFF].
	au16GShadingTbl	Correction table of the G component. The value range is [0x0, 0xFFFF].
	au16BShadingTbl	Correction table of the B component. The value range is [0x0, 0xFFFF].
	u16ROffCenter	Distance between the R component center and the farthest angle. A longer distance indicates a smaller value. The value range is [0x0, 0xFFFF].
	u16GOffCenter	Distance between the G component center and the farthest angle. A longer distance indicates a smaller value. The value range is [0x0, 0xFFFF].
	u16BOffCenter	Distance between the B component center and the farthest angle. A longer distance indicates a smaller value. The value range is [0x0, 0xFFFF].
	u16TblNodeNum	Number of used nodes in each component correction table. The value range is [0x0, 0x81], and the default value is 0x81.

[Note]

The default values of this data structure are stored in **sensor\_cmos.c**. If you want to change the default values, modify the corresponding parameters in **sensor\_cmos.c**. If you want to connect a new sensor, refer to the default values of the provided sensors.

[See Also]

ISP\_SENSOR\_EXP\_FUNC\_S

## ISP\_CMOS\_BLACK\_LEVEL\_S

[Description]

Defines the sensor black level.



[Syntax]

```
typedef struct hiISP_CMOS_BLACK_LEVEL_S
{
    HI_BOOL bUpdate;

    HI_U8  au8BlackLevel[4];
} ISP_CMOS_BLACK_LEVEL_S;
```

[Member]

Member	Description
bUpdate	Whether the sensor black level varies according to the gain Value range: [0, 1]
au8BlackLevel	Sensor black level array Value range: [0, 255]

[Note]

If the sensor black level does not vary according to the gain, set **bUpdate** to **HI\_FALSE**.

[See Also]

ISP\_SENSOR\_EXP\_FUNC\_S

## ALG\_LIB\_S

[Description]

Defines AE, AWB, and AF algorithm libraries.

[Syntax]

```
typedef struct hiALG_LIB_S
{
    HI_S32  s32Id;
    HI_CHAR acLibName[20];
} ALG_LIB_S;
```

[Member]

Member	Description
s32Id	ID of the algorithm library instance
acLibName	Character array for identifying the algorithm library name

[Note]





**acLibName** is used to distinguish algorithm libraries, and **s32Id** is used to support multiple instances in an algorithm library.

[See Also]

None

## ISP\_BIND\_ATTR\_S

[Description]

Defines the binding relationship between the ISP library and sensors/3A algorithm libraries.

[Syntax]

```
typedef struct hiISP_BIND_ATTR_S
{
    SENSOR_ID    SensorId;
    ALG_LIB_S    stAeLib;
    ALG_LIB_S    stAfLib;
    ALG_LIB_S    stAwbLib;
} ISP_BIND_ATTR_S;
```

[Member]

Member	Description
SensorId	Sensor ID.
stAeLib	Data structure of the AE algorithm library
stAfLib	Data structure of the AF algorithm library
stAwbLib	Data structure of the AWB algorithm library

[Note]

None

[See Also]

None

## ISP\_AE\_REGISTER\_S

[Description]

Defines AE registration information.

[Syntax]

```
typedef struct hiISP_AE_REGISTER_S
{
    AE_EXP_FUNC_S stAeExpFunc;
} ISP_AE_REGISTER_S;
```



[Member]

Member	Description
stAeExpFunc	Callback function for registering the AE algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

AE\_EXP\_FUNC\_S

## ISP\_AE\_EXP\_FUNC\_S

[Description]

Defines the AE callback function.

[Syntax]

```
typedef struct hiISP_AE_EXP_FUNC_S
{
    HI_S32 (*pfn_ae_init)(HI_S32 s32Handle, const ISP_AE_PARAM_S
*pstAeParam);

    HI_S32 (*pfn_ae_run)(HI_S32 s32Handle,
        const ISP_AE_INFO_S *pstAeInfo,
        ISP_AE_RESULT_S *pstAeResult,
        HI_S32 s32Rsv);

    HI_S32 (*pfn_ae_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);

    HI_S32 (*pfn_ae_exit)(HI_S32 s32Handle);
} ISP_AE_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_ae_init	Pointer to the callback function for initializing the AE algorithm library
pfn_ae_run	Pointer to the callback function for running the AE algorithm library
pfn_ae_ctrl	Pointer to the callback function for controlling the internal status of the AE algorithm library
pfn_ae_exit	Pointer to the callback function for destroying the AE algorithm library



[Note]

- `pfn_ae_init` is called when `HI_MPI_ISP_Init` is called to initialize the AE algorithm library.
- `pfn_ae_run` is called when `CHI_MPI_ISP_Run` is called to run the AE algorithm library and calculate the exposure time and gain of the sensor and the digital gain of the ISP.
- According to the design methodology, a ctrl interface is implemented in the AE algorithm library to change the internal running status. The ctrl interface provides a parameter transfer command and a VOID data transfer pointer. The ctrl interface is registered with the ISP library as a callback function pointer. The ISP control unit inexplicitly calls commands to control the internal running status of the AE algorithm library. The ctrl interface can also be called as a user interface to change the internal running status of the AE algorithm status. See the following instance:

```
HI_S32 AeCtrlCmd(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue)
{
    AE_CHECK_POINTER(pValue);

    switch (u32Cmd)
    {
        case ISP_WDR_MODE_SET :
            .....
            break;
        .....
    }

    return HI_SUCCESS;
}
```

- When the ISP runs, the ISP control unit inexplicitly calls `pfn_ae_ctrl` to prompt the AE algorithm library to switch the WDR or linear mode, set the frame rate, and configure the sensor.

The following is the current ctrl command defined by the firmware:

```
typedef enum hiISP_CTRL_CMD_E
{
    ISP_WDR_MODE_SET = 8000,
    ISP_AE_FPS_BASE_SET,
    ISP_AWB_ISO_SET, /*Set the ISO and change the saturation when the ISO
changes.*/

    ISP_CTRL_CMD_BUTT,
} ISP_CTRL_CMD_E;
```

- `pfn_ae_exit` is called when `HI_MPI_ISP_Exit` is called to destroy the AE algorithm library.
- Multiple instances can be initialized and run in an algorithm library. The **s32Handle** parameter is used to distinguish instances. If you want to support multiple instances, use different **stAlgLib**. **s32Id** to register instances with the algorithm library for multiple times. See the following instance:



```
ALG_LIB_S stAeLib;  
  
stAeLib.s32Id = 0;  
  
strcpy(stAeLib.acLibName, HI_AE_LIB_NAME);  
  
HI_MPI_AE_Register(&stAeLib);  
  
stAeLib.s32Id = 1;  
  
HI_MPI_AE_Register(&stAeLib);
```

[See Also]

ISP\_AE\_REGISTER\_S

## ISP\_AE\_PARAM\_S

[Description]

Defines the initialization parameter provided by the ISP for the AE algorithm library.

[Syntax]

```
typedef struct hiISP_AE_PARAM_S  
{  
  
    SENSOR_ID SensorId;  
  
  
    HI_U32 u32MaxIspDgain;  
  
    HI_U32 u32MinIspDgain;  
  
    HI_U32 u32IspDgainShift;  
  
} ISP_AE_PARAM_S;
```

[Member]

Member	Description
SensorId	ID of the sensor that is registered with the ISP library. This ID is used to check whether the sensor registered with the ISP library is the same as the one registered with the AE algorithm library.
u32MaxIspDgain	Maximum digital gain of the ISP with the accuracy of <b>u32IspDgainShift</b>
u32MinIspDgain	Minimum digital gain of the ISP with the accuracy of <b>u32IspDgainShift</b>
u32IspDgainShift	ISP digital gain accuracy

[Note]



The gain accuracy meets the following condition:  $1 \ll \mathbf{u32IspDgainShift}$ . If  $\mathbf{u32MaxIspDgain}$  is **512**,  $\mathbf{u32IspDgainShift}$  is **4**, and the gain accuracy is **16** ( $1 \ll 4$ ), the maximum digital gain supported by the ISP is  $32\times$  ( $512/16$ ).

[See Also]

ISP\_AE\_EXP\_FUNC\_S

## ISP\_AE\_INFO\_S

[Description]

Defines the statistics provided by the ISP for the AE algorithm library.

[Syntax]

```
typedef struct hiISP_AE_INFO_S
{
    HI_U32  u32FrameCnt;    /* the counting of frame */

    ISP_AE_STAT_1_S *pstAeStat1;
    ISP_AE_STAT_2_S *pstAeStat2;
    ISP_AE_STAT_3_S *pstAeStat3;
} ISP_AE_INFO_S;
```

[Member]

Member	Sub Member	Description
u32FrameCnt	None	Total number of frames. The value range is [0, 0xFFFFFFFF].
pstAeStat1	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram. The value range is [0, 255].
	au16MeteringHist	Statistics array of the 5-segment histogram. The value range is [0, 0xFFFF].
pstAeStat2	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram. The value range is [0, 255].
	au16MeteringMemArray	Zone statistics array of the 5-segment histogram. The value range is [0, 0xFFFF].
pstAeStat3	au16HistogramMemArray	Statistics array of the 256-segment histogram. The value range is [0, 0xFFFF].

[Note]

- The operation frequency (for example, once per two frames) of the AE algorithm library can be controlled by setting **u32FrameCnt**.



- For details about the definitions of the 5-segment histogram and 256-segment histogram, see section 3.3 "Function Description."
- Statistics are displayed as a global 5-segment histogram, a 5-segment histogram with 15 x 17 zones (separate statistics for each zone), and a 256-segment histogram. When a data structure pointer is not null, the statistics are valid. An MPI for setting the mode of displaying statistics will be provided.
- The statistics are normalized to 0xFFFF.

[See Also]

ISP\_AE\_EXP\_FUNC\_S

## ISP\_AE\_RESULT\_S

[Description]

Defines the results returned by the AE algorithm library to the ISP for configuring registers.

[Syntax]

```
typedef struct hiISP_AE_RESULT_S
{
    HI_U32  u32IspDgain;

    HI_U32  u32IspDgainShift;

    HI_U32  u32Iso;

    ISP_AE_STAT_ATTR_S stStatAttr;
} ISP_AE_RESULT_S;
```

[Member]

Member	Sub Member	Description
u32IspDgain	None	ISP digital gain.
u32IspDgainShift	None	ISP digital gain accuracy.
u32Iso	None	Total gain calculated by the AE algorithm library.
stStatAttr	bChange	Whether the sub members of <b>stStatAttr</b> need to be configured again.
	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram. The value range is [0, 255].
	au8WeightTable	AE weight table of 15 x 17 zones. The value range is [0, 255].

[Note]

- The ISP basic algorithm unit adjusts the parameters such as the sharpness and denoise parameters based on the total gain calculated by the AE algorithm library.



- There are default segmentation thresholds of the 5-segment histogram and AE weight table in the ISP library. You are advised to configure the segmentation threshold of the 5-segment histogram based on the used sensor. The related registers do not need to be configured frequently because the **bChange** parameter specifies whether the returned values need to be configured again.

[See Also]

ISP\_AE\_EXP\_FUNC\_S

## ISP\_AWB\_REGISTER\_S

[Description]

Defines AWB registration information.

[Syntax]

```
typedef struct hiISP_AWB_REGISTER_S
{
    AWB_EXP_FUNC_S stAwbExpFunc;
} ISP_AWB_REGISTER_S;
```

[Member]

Member	Description
stAwbExpFunc	Callback function for registering the AWB algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

AWB\_EXP\_FUNC\_S

## ISP\_AWB\_EXP\_FUNC\_S

[Description]

Defines the AWB callback function.

[Syntax]

```
typedef struct hiISP_AWB_EXP_FUNC_S
{
    HI_S32 (*pfn_awb_init)(HI_S32 s32Handle, const ISP_AWB_PARAM_S
        *pstAwbParam);

    HI_S32 (*pfn_awb_run)(HI_S32 s32Handle,
        const ISP_AWB_INFO_S *pstAwbInfo,
        ISP_AWB_RESULT_S *pstAwbResult,
```



```
HI_S32 s32Rsv);  
  
HI_S32 (*pfn_awb_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);  
  
HI_S32 (*pfn_awb_exit)(HI_S32 s32Handle);  
  
} ISP_AWB_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_awb_init	Pointer to the callback function for initializing the AWB algorithm library
pfn_awb_run	Pointer to the callback function for running the AWB algorithm library
pfn_awb_ctrl	Pointer to the callback function for controlling the internal status of the AWB algorithm library
pfn_awb_exit	Pointer to the callback function for destroying the AWB algorithm library

[Note]

- pfn\_awb\_init is called when HI\_MPI\_ISP\_Init is called to initialize the AWB algorithm library.
- pfn\_awb\_run is called when HI\_MPI\_ISP\_Run is called to run the AWB algorithm library and calculate the white balance gain and color correction matrix (CCM).
- When the ISP runs, the control unit inexplicitly calls pfn\_awb\_ctrl to prompt the AWB algorithm library to switch the WDR or linear mode and set the ISO (sensor gain). The ISO is related to the saturation. The chrominance noises are large when the gain is large. The saturation needs to be adjusted to set the ISO.

The following is the current ctrl command defined by the firmware:

```
typedef enum hiISP_CTRL_CMD_E  
{  
    ISP_WDR_MODE_SET = 8000,  
    ISP_AE_FPS_BASE_SET,  
    ISP_AWB_ISO_SET, /*Set the ISO and change the saturation when the ISO  
changes.*/  
  
    ISP_CTRL_CMD_BUTT,  
} ISP_CTRL_CMD_E;
```

- pfn\_awb\_exit is called when HI\_MPI\_ISP\_Exit is called to destroy the AWB algorithm library.

[See Also]

ISP\_AWB\_REGISTER\_S

## ISP\_AWB\_PARAM\_S

[Description]

Defines the initialization parameter provided by the ISP for the AWB algorithm library.





[Syntax]

```
typedef struct hiISP_AWB_PARAM_S
{
    SENSOR_ID SensorId;

    HI_S32 s32Rsv;
} ISP_AWB_PARAM_S;
```

[Member]

Member	Description
SensorId	ID of the sensor that is registered with the ISP library. This ID is used to check whether the sensor registered with the ISP library is the same as the one registered with the AWB algorithm library.
s32Rsv	Reserved

[Note]

None

[See Also]

ISP\_AWB\_EXP\_FUNC\_S

## ISP\_AWB\_INFO\_S

[Description]

Defines the statistics provided by the ISP for the AWB algorithm library.

[Syntax]

```
typedef struct hiISP_AWB_INFO_S
{
    HI_U32 u32FrameCnt;    /* the counting of frame */

    ISP_AWB_STAT_1_S *pstAwbStat1;
    ISP_AWB_STAT_2_S *pstAwbStat2;
} ISP_AWB_INFO_S;
```

[Member]

Member	Sub Member	Description
u32FrameCnt	None	Total number of frames. The value range is [0, 0xFFFFFFFF].



Member	Sub Member	Description
pstAwbStat1	u16MeteringAwbRg	Ratio of the average R value to the average G value of white points in statistics. The value range is [0, 0xFFFF].
	u16MeteringAwbBg	Ratio of the average B value to the average G value of white points in statistics. The value range is [0, 0xFFFF].
	u32MeteringAwbSum	Number of white points in the statistics. The value range is [0, 0xFFFFFFFF].
pstAwbStat2	au16MeteringMemArrayRg	Ratio of the average R value to the average G value of white points in statistics by zone. The value range is [0, 0xFFFF].
	au16MeteringMemArrayBg	Ratio of the average B value to the average G value of white points in statistics by zone. The value range is [0, 0xFFFF].
	au16MeteringMemArraySum	Number of white points in the statistics by zone. The value range is [0, 0xFFFFFFFF].

[Note]

- The operation frequency (for example, once per two frames) of the AWB algorithm library can be controlled by setting **u32FrameCnt**.
- For details about the definitions of **Rg**, **Bg**, and **Sum**, see chapter 4 "AWB."
- Statistics are provided as global statistics and statistics by 15 x17 zones (separate statistics for each zone).

[See Also]

ISP\_AWB\_EXP\_FUNC\_S

## ISP\_AWB\_RESULT\_S

[Description]

Defines the results returned by the AWB algorithm library to the ISP for configuring registers.

[Syntax]

```
typedef struct hiISP_AWB_RESULT_S
{
    HI_U32  au32WhiteBalanceGain[4];

    HI_U16  au16ColorMatrix[9];

    ISP_AWB_STAT_ATTR_S stStatAttr;
} ISP_AWB_RESULT_S;
```



[Member]

Member	Sub Member	Description
au32WhiteBalanceGain	None	Gain of the R, Gr, Gb, and B color channels obtained by using the AWB algorithm (16-bit accuracy).
au16ColorMatrix	None	CCM (8-bit accuracy).
stStatAttr	bChange	Whether the sub members of <b>stStatAttr</b> need to be configured again.
	u16MeteringWhiteLevelAwb	Upper limit for searching for white points during white point statistics. The default value is 0x3ac.
	u16MeteringBlackLevelAwb	Lower limit for searching for white points during white point statistics. The default value is 0x40.
	u16MeteringCrRefMaxAwb	Maximum R/G value of the white point area during white point statistics. The default value is 512.
	u16MeteringCbRefMaxAwb	Maximum B/G value of the white point area during white point statistics. The default value is 512.
	u16MeteringCrRefMinAwb	Minimum R/G value of the white point area during white point statistics. The default value is 128.
	u16MeteringCbRefMinAwb	Minimum B/G value of the white point area during white point statistics. The default value is 128.

[Note]

- The AWB algorithm library calculates the gains of the R, Gr, Gb, and B color channels to obtain the corrected white color. The 16-bit accuracy indicates the last 16 bits are decimals.
- In WDR mode, images are placed in a non-linear space because of GammaFe. Therefore, the square root is extracted from the return value of the AWB algorithm library and then configured to registers. If you develop a new AWB algorithm, the ISP control unit extracts the square root when configuring registers. The AWB algorithm library only needs to return correct 16-bit gains of four color channels.



- The ISP digital gain is added to the gains of four color channels. The ISP control unit performs the addition operation when configuring registers. The AWB algorithm library only needs to return correct 16-bit gains of four components.
- The AWB algorithm library also calculates a 3 x 3 CCM to reproduce actual colors. The 8-bit accuracy indicates that the last eight bits are decimals.
- The information in the **stStatAttr** data structure determines the pixels that are considered as white points for statistics. You can use the default values of **stStatAttr** or customize **stStatAttr** when you develop a new AWB algorithm. The **bChange** parameter indicates that whether the values of **stStatAttr** need to be configured to registers in the current frame.

[See Also]

ISP\_AWB\_EXP\_FUNC\_S

## ISP\_AF\_REGISTER\_S

[Description]

Defines AF registration information.

[Syntax]

```
typedef struct hiISP_AF_REGISTER_S
{
    AF_EXP_FUNC_S stAfExpFunc;
} ISP_AF_REGISTER_S;
```

[Member]

Member	Description
stAfExpFunc	Callback function for registering the AF algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

AF\_EXP\_FUNC\_S

## ISP\_AF\_EXP\_FUNC\_S

[Description]

Defines the AF callback function.

[Syntax]

```
typedef struct hiISP_AF_EXP_FUNC_S
{
```



```
HI_S32 (*pfn_af_init)(HI_S32 s32Handle, const ISP_AF_PARAM_S
*pstAfParam);

HI_S32 (*pfn_af_run)(HI_S32 s32Handle,
const ISP_AF_INFO_S *pstAfInfo,
ISP_AF_RESULT_S *pstAfResult,
HI_S32 s32Rsv);

HI_S32 (*pfn_af_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);
HI_S32 (*pfn_af_exit)(HI_S32 s32Handle);
} ISP_AF_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_af_init	Pointer to the callback function for initializing the AF algorithm library
pfn_af_run	Pointer to the callback function for running the AF algorithm library
pfn_af_ctrl	Pointer to the callback function for controlling the internal status of the AF algorithm library
pfn_af_exit	Pointer to the callback function for destroying the AF algorithm library

[Note]

The AF algorithm library is not implemented currently.

[See Also]

ISP\_AF\_REGISTER\_S

## ISP\_AF\_PARAM\_S

[Description]

Defines the initialization parameter provided by the ISP for the AF algorithm library.

[Syntax]

```
typedef struct hiISP_AF_PARAM_S
{
    SENSOR_ID SensorId;

    HI_S32 s32Rsv;
} ISP_AF_PARAM_S;
```

[Member]



Member	Description
SensorId	ID of the sensor that is registered with the ISP library. This ID is used to check whether the sensor registered with the ISP library is the same as the one registered with the AF algorithm library.
s32Rsv	Reserved

[Note]

None

[See Also]

ISP\_AF\_EXP\_FUNC\_S

## ISP\_AF\_INFO\_S

[Description]

Defines the statistics provided by the ISP for the AF algorithm library.

[Syntax]

```
typedef struct hiISP_AF_INFO_S
{
    HI_U32  u32FrameCnt;    /* the counting of frame */

    ISP_AF_STAT_S *pstAfStat;
} ISP_AF_INFO_S;
```

[Member]

Member	Description
u32FrameCnt	Total number of frames. The value range is [0, 0xFFFFFFFF]
pstAfStat	Pointer to AF statistics

[Note]

The AF algorithm library is not implemented currently.

[See Also]

ISP\_AF\_EXP\_FUNC\_S

## ISP\_AF\_RESULT\_S

[Description]



Defines the results returned by the AF algorithm library to the ISP for configuring registers.

[Syntax]

```
typedef struct hiISP_AF_RESULT_S  
{  
    HI_S32 s32Rsv;  
} ISP_AF_RESULT_S;
```

[Member]

Member	Description
s32Rsv	Reserved

[Note]

The AF algorithm library is not implemented currently.

[See Also]

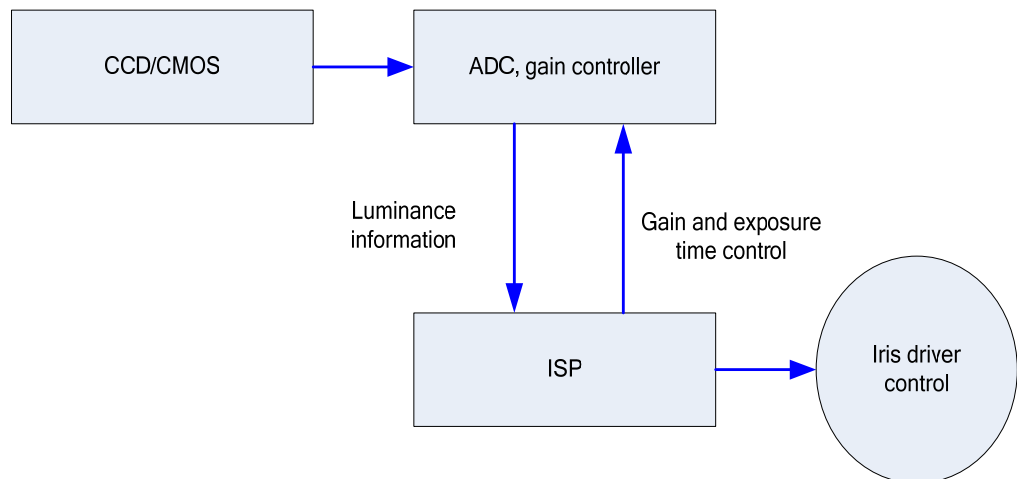
ISP\_AF\_EXP\_FUNC\_S

# 3 AE

## 3.1 Overview

The HiISP AE module obtains the current image exposure based on the automatic photometric system, and automatically sets the iris, sensor shutter, and gain to ensure optimal image quality. The AE algorithms include the iris first algorithm, shutter first algorithm, and gain first algorithm. The shutter first algorithm is used to limit the exposure time first and applies to the scenario of taking pictures of moving objects. The gain first algorithm is used to limit the sensor gain first, ensuring low image noise. [Figure 3-1](#) shows the workflow of the AE module.

**Figure 3-1** Workflow of the AE module



## 3.2 Important Concepts

The following describes the concepts related to the AE module:

- **Exposure time:** It is the period for the sensor to accumulate electric charge. That is, it is the period that starts from sensor pixel exposure and ends when the amount of electricity is read.

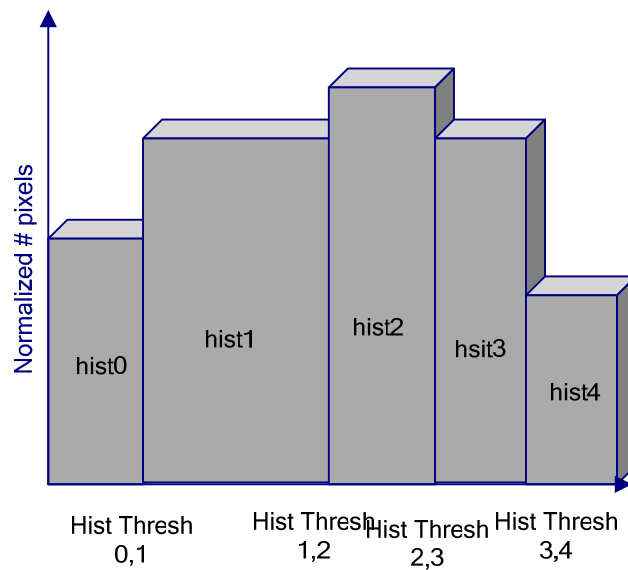


- Exposure gain: It is the total amplification coefficient for the output electric charge of the sensor. The exposure gains include the digital gain and analog gain. As the noise caused by analog gain is low, analog gain is preferred.
- Iris and mechanical shutter: The iris changes the central hole size of the lens, and the mechanical shutter controls the exposure time. The iris works with the mechanical shutter to control the amount of input light.
- Anti-flicker: The screen flickers when the power working frequency of the electric light does not match the sensor frame rate. Anti-flicker is implemented by specifying the exposure time and changing the sensor frame rate.

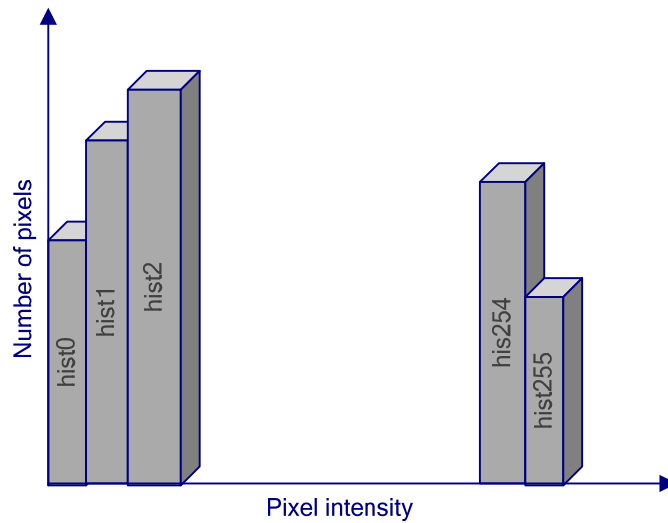
### 3.3 Function Description

The AE module consists of the AE statistics module and AE algorithm firmware (providing the AE control policy). The AE statistics module collects statistics on the luminance of the sensor input data. The statistics are displayed as histograms. The AE module can provide a 5-segment histogram or 256-segment histogram for the entire image or divide the entire image into M x N zones and then provide the histogram of each zone. See [Figure 3-2](#) and [Figure 3-3](#)

**Figure 3-2** 5-segment AE statistics histogram

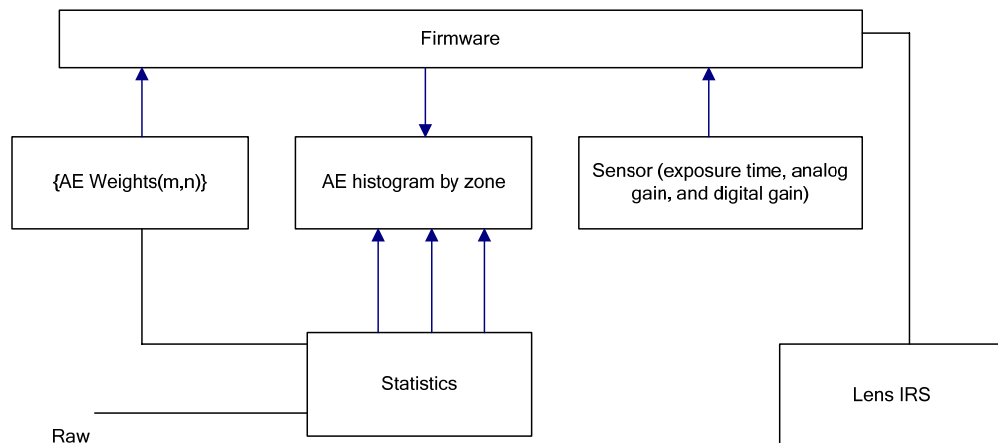


**Figure 3-3** 256-segment AE statistics histogram



The AE algorithm principle is that the input image statistics are obtained in real time and compared with the target luminance. The sensor exposure time, gain, and lens iris are dynamically adjusted to ensure that the actual luminance is close to the target luminance. See [Figure 3-4](#).

**Figure 3-4** AE working principle



## 3.4 API Reference

### 3.4.1 AE Algorithm Library MPIS

The following are AE algorithm library MPIS:

- [HI\\_MPI\\_AE\\_Register](#): Registers the AE algorithm library with the ISP.



- [HI\\_MPI\\_AE\\_SensorRegCallBack](#): Registers a sensor. This callback function is provided by the AE algorithm library.

## HI\_MPI\_AE\_Register

[Description]

Registers the AE algorithm library with the ISP.

[Syntax]

```
HI_S32 HI_MPI_AE_Register(ALG_LIB_S *pstAeLib);
```

[Parameter]

Parameter	Description	Input/Output
pstAeLib	Pointer to the data structure of the AE algorithm library	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_ae.h
- Library file: libisp.a

[Note]

- This MPI calls the AE registration callback function HI\_MPI\_ISP\_AeLibRegCallBack provided by the ISP library to register the AE algorithm library with the ISP library.
- You can call this MPI to register the AE algorithm library with the ISP library.
- Multiple instances can be registered with the AE algorithm library.

[Example]

```
stAeLib.s32Id = 0;

strcpy(stAeLib.acLibName, HI_AE_LIB_NAME);

HI_MPI_AE_Register(&stAeLib);

stAeLib.s32Id = 1;

HI_MPI_AE_Register(&stAeLib);
```

[See Also]

[HI\\_MPI\\_ISP\\_AeLibRegCallBack](#)

## HI\_MPI\_AE\_SensorRegCallBack

### [Description]

Registers a sensor. This callback function is provided by the AE algorithm library.

### [Syntax]

```
HI_S32 HI_MPI_AE_SensorRegCallBack(ALG_LIB_S *pstAeLib, SENSOR_ID SensorId,
AE_SENSOR_REGISTER_S *pstRegister);
```

### [Parameter]

Parameter	Description	Input/Output
pstAeLib	Pointer to the data structure of the AE algorithm library	Input
SensorId	ID of the sensor that is registered with the AE algorithm library	Input
pstRegister	Pointer to the data structure for registering a sensor	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

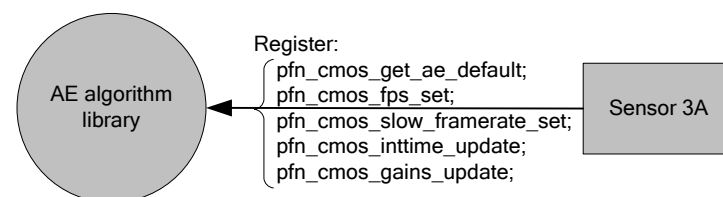
### [Requirement]

- Header files: hi\_comm\_isp.h, mpi\_ae.h
- Library file: libisp.a

### [Requirement]

- **SensorId** can be configured in the sensor library and is used to ensure that the sensor registered with the ISP library is the same as that registered with the 3A algorithm library.
- The AE algorithm library obtains differentiated initialization parameters and controls sensors by using the callback interfaces registered by sensors.

**Figure 3-5** Interfaces between the AE algorithm library and the sensor library





[Example]

```
ALG_LIB_S stLib;
AE_SENSOR_REGISTER_S stAeRegister;
AE_SENSOR_EXP_FUNC_S *pstExpFuncs = &stAeRegister.stSnsExp;

memset(pstExpFuncs, 0, sizeof(AE_SENSOR_EXP_FUNC_S));
pstExpFuncs->pfn_cmos_get_ae_default = cmos_get_ae_default;
pstExpFuncs->pfn_cmos_fps_set = cmos_fps_set;
pstExpFuncs->pfn_cmos_slow_framerate_set = cmos_slow_framerate_set;
pstExpFuncs->pfn_cmos_inttime_update = cmos_inttime_update;
pstExpFuncs->pfn_cmos_gains_update = cmos_gains_update;

stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AE_LIB_NAME);
s32Ret = HI_MPI_AE_SensorRegCallBack(&stLib, IMX104_ID, &stAeRegister);
if (s32Ret)
{
    printf("sensor register callback function to ae lib failed!\n");
    return s32Ret;
}
```

[See Also]

None

## 3.4.2 AE Control MPIS

The following are AE control MPIS:

- [HI\\_MPI\\_ISP\\_SetExposureType](#): Sets the AE control type.
- [HI\\_MPI\\_ISP\\_GetExposureType](#): Obtains the AE control type.
- [HI\\_MPI\\_ISP\\_SetAEAttr](#): Sets AE attributes.
- [HI\\_MPI\\_ISP\\_GetAEAttr](#): Obtains AE attributes.
- [HI\\_MPI\\_ISP\\_SetAEAttrEx](#): Sets extended AE attributes.
- [HI\\_MPI\\_ISP\\_GetAEAttrEx](#): Obtains extended AE attributes.
- [HI\\_MPI\\_ISP\\_SetAEDelayAttr](#): Sets AE delay attributes.
- [HI\\_MPI\\_ISP\\_GetAEDelayAttr](#): Obtains AE delay attributes.
- [HI\\_MPI\\_ISP\\_SetAERouteAttr](#): Sets the AE allocation policy.
- [HI\\_MPI\\_ISP\\_GetAERouteAttr](#): Obtains the AE allocation policy.
- [HI\\_MPI\\_ISP\\_GetAEAttrEx](#): Obtains extended AE attributes.
- [HI\\_MPI\\_ISP\\_SetMEAttr](#): Sets manual exposure (ME) attributes.
- [HI\\_MPI\\_ISP\\_GetMEAttr](#): Obtains ME attributes.
- [HI\\_MPI\\_ISP\\_SetMEAttrEx](#): Sets extended ME attributes.
- [HI\\_MPI\\_ISP\\_GetMEAttrEx](#): Obtains extended ME attributes.
- [HI\\_MPI\\_ISP\\_SetExpStaInfo](#): Sets the AE statistics.



- [HI\\_MPI\\_ISP\\_GetExpStaInfo](#): Obtains the AE statistics.
- [HI\\_MPI\\_ISP\\_QueryInnerStateInfo](#): Obtains the internal status information.
- [HI\\_MPI\\_ISP\\_QueryInnerStateInfoEx](#): Obtains the internal status information.

## HI\_MPI\_ISP\_SetExposureType

### [Description]

Sets the AE control type.

### [Syntax]

```
HI_S32 HI_MPI_ISP_SetExposureType(ISP\_OP\_TYPE\_E enExpType);
```

### [Parameter]

Parameter	Description	Input/Output
enExpType	AE control type	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

### [Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

### [Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

### [Note]

- If the AE control type is set to automatic mode, the exposure time and exposure gain are automatically controlled based on the AE algorithm.
- If the AE control type is set to automatic mode, ME attributes must be set. The ME attributes include enable parameters (exposure time enable, exposure analog gain enable, and exposure digital gain enable) and exposure parameters (exposure time, exposure analog gain, and exposure digital gain).

### [Example]

```
{  
    ISP\_OP\_TYPE\_E enExpType;
```



```
ISP_ME_ATTR_S stMEAttr;  
enExpType = OP_TYPE_MANUAL;  
stMEAttr.bManualExpLineEnable = 1;  
stMEAttr.bManualAGainEnable = 0;  
stMEAttr.bManualDGainEnable = 0;  
stMEAttr.u32ExpLine = 200;  
  
ISP_MST_StartIsp();  
PAUSE;  
CHECK_RET(HI_MPI_ISP_SetExposureType(enExpType), "set exposure type");  
CHECK_RET(HI_MPI_ISP_SetMEAttr(&stMEAttr), "set ME Attr");  
PAUSE;  
enExpType = OP_TYPE_AUTO;  
CHECK_RET(HI_MPI_ISP_SetExposureType(enExpType), "set exposure type");  
PAUSE;  
ISP_MST_StopIsp();  
  
ISP_MST_PASS();  
}
```

[See Also]

[HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_GetExposureType

[Description]

Obtains the AE control type.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetExposureType(ISP\_OP\_TYPE\_E *penExpType);
```

[Parameter]

Parameter	Description	Input/Output
penExpType	AE control type	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]



Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetExposureType](#)

## HI\_MPI\_ISP\_SetAEAttr

[Description]

Sets AE attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAEAttr(const ISP\_AE\_ATTR\_S*pstAEAttr);
```

[Parameter]

Parameter	Description	Input/Output
<code>pstAEAttr</code>	AE attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.





[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- This MPI is used to set the exposure time, and the minimum values and maximum values for digital gain and analog gain.
- The exposure adjustment step attribute is used to adjust the exposure convergence speed. A larger step indicates a higher exposure convergence speed but more flickered effect.
- The exposure tolerance attribute is used to adjust the sensitivity to the environment during exposure. A larger exposure tolerance indicates more insensitive effect and greater luminance error that is caused by adjusting the same target luminance for multiple times. Therefore, the exposure tolerance cannot be too large.
- The exposure luminance compensation attribute is used to adjust the exposure luminance. A larger exposure luminance compensation value indicates higher image luminance.
- The exposure weight table attribute is used to adjust the exposure weight of the region of interest (ROI).

[Example]

```
{  
    ISP_AE_ATTR_S stAEAttr;  
    HI_MPI_ISP_GetAEAttr(&stAEAttr);  
  
    stAEAttr.u16ExpTimeMax = 4000;  
    stAEAttr.u16ExpTimeMin = 2;  
    stAEAttr.u16AGainMax   = 28;  
    stAEAttr.u16AGainMin   = 0;  
    stAEAttr.u16DGainMax   = 38;  
    stAEAttr.u16DGainMin   = 0;  
    stAEAttr.u8ExpStep     = 8;  
    stAEAttr.s16ExpTolerance = 4;  
    stAEAttr.u8ExpCompensation = 0x20;  
  
    HI_MPI_ISP_SetAEAttr(&stAEAttr);  
}
```

[See Also]

[HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_GetAEAttr

[Description]

Obtains AE attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAEAttr(const ISP_AE_ATTR_S *pstAEAttr);
```



[Parameter]

Parameter	Description	Input/Output
pstAEAttr	AE attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_SetAERouteAttr

[Description]

Sets the AE allocation policy.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAERouteAttr(const ISP_AE_ROUTE_S *pstAERouteAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstAERouteAttr	AE allocation policy	Input



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

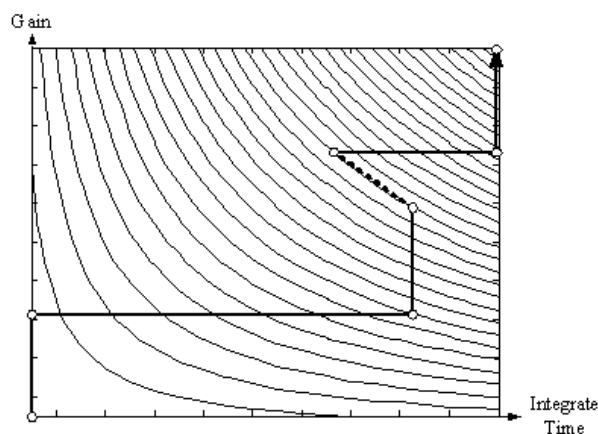
- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

- This MPI is used to set AE allocation routes. The exposure calculated by the AE module is allocated based on the configured route. You can also set the exposure first, gain first, and shutter first modes.
- [Figure 3-6](#) shows the allocated AE routes. Note the following limitations when allocating AE routes:
  - A maximum of eight nodes are supported. Each node has the exposure time, gain, and iris components. The gain includes analog gain, digital gain, and ISP digital gain.
  - Only P-iris is supported. The DC-iris is not supported, because it cannot be accurately controlled.
  - The node exposure is the product of the exposure time, gain, and iris. The node exposure is monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest.
  - If the exposure of an adjacent node increases, the value of a component should increase while the values of other components are retained. The component with increased value determines the allocation policy of the route. For example, if the value of the gain component increases, the allocation policy of the route is gain first.
  - The exposure of adjacent nodes can be the same. In this case, an allocation mode can be switched to another one.
  - You can set routes based on the application scenario. The allocation route can be dynamically switched.
  - The exposure time and then gain are allocated by default. If the current exposure does not fall within the configured route range, the default allocation policy is used.



**Figure 3-6** AE allocation routes



[Example]

```
{
    ISP_AE_ROUTE_S stRoute;
    HI_U32 au32RouteNode[ISP_AE_ROUTE_MAX_NODES][3] =
    {{2,1024,1},{100,1024,1},{100,5120,1},{300,5120,1},{300,102400,1},{748,10
    2400,1},{748,786432,1}};

    HI_MPI_ISP_GetAERouteAttr(&stRoute);
    stRoute.u32TotalNum = 7;
    memcpy(stRoute.astRouteNode, au32RouteNode, sizeof(au32RouteNode));
    HI_MPI_ISP_SetAERouteAttr(&stRoute);
}
```

[See Also]

[HI\\_MPI\\_ISP\\_GetAERouteAttr](#)

## HI\_MPI\_ISP\_GetAERouteAttr

[Description]

Obtains the AE allocation policy.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAERouteAttr(ISP_AE_ROUTE_S *pstAERouteAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstAERouteAttr	AE allocation policy	Output

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetAERouteAttr](#)

## HI\_MPI\_ISP\_SetAEAttrEx

[Description]

Sets extended AE attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAEAttrEx(const ISP_AE_ATTR_S_EX *pstAEAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
pstAEAttrEx	Extended AE attributes	Input

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- HI\_MPI\_ISP\_SetAEAttrEx is an extended AE attribute MPI. Compared with HI\_MPI\_ISP\_SetAEAttr, HI\_MPI\_ISP\_SetAEAttrEx supports 10-bit analog gain or digital gain and allows you to set maximum ISP digital gain and the maximum system gain.
- HI\_MPI\_ISP\_SetAEAttrEx is used to set the exposure time, maximum sensor digital gain and sensor analog gain, minimum sensor digital gain and sensor analog gain, maximum ISP digital gain, and the maximum system gain.
- The exposure adjustment step attribute is used to adjust the exposure convergence speed. A larger step indicates a higher exposure convergence speed but more flickered effect.
- The exposure tolerance attribute is used to adjust the sensitivity to the environment during exposure. A larger exposure tolerance indicates more insensitive effect and greater luminance error that is caused by adjusting the same target luminance for multiple times. Therefore, the exposure tolerance cannot be too large.
- The exposure luminance attribute is used to adjust the exposure luminance. A larger exposure luminance compensation value indicates higher image luminance.
- The exposure weight table attribute is used to adjust the exposure weight of the ROI.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_GetAEAttrEx

[Description]

Obtains extended AE attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAEAttrEx(const ISP_AE_ATTR_EX_S *pstAEAttrEx);
```



[Parameter]

Parameter	Description	Input/Output
pstAEAttrEx	Extended AE attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetExposureType](#)

## HI\_MPI\_ISP\_SetAEDelayAttr

[Description]

Sets AE delay attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAEDelayAttr(const ISP_AE_DELAY_S *pstAEDelayAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAEDelayAttr	AE delay attributes	Input



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_ae.h`
- Library file: `lib_hiae.a`

[Note]

- AE adjustment starts only when the duration in which the picture luminance is less than or greater than the target luminance exceeds the configured value.
- When the AE adjustment step is the same, larger values of **u16BlackDelayFrame** and **u16WhiteDelayFrame** indicate longer time for adjusting the luminance to the target value. Therefore, if you want the same AE convergence speed before and after setting the AE delay, you need to change the AE adjustment step.
- Human eyes are sensitive to overexposure. It is recommended that the value of **u16WhiteDelayFrame** be less than **u16BlackDelayFrame**.

[Example]

None

[See Also]

`HI_MPI_ISP_SetAEAttrEx`

## HI\_MPI\_ISP\_GetAEDelayAttr

[Description]

Obtains AE delay attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAEDelayAttr(ISP_AE_DELAY_S *pstAEDelayAttr);
```

[Parameter]

Parameter	Description	Input/Output
<code>pstAEDelayAttr</code>	AE delay attributes	Output





[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_ae.h
- Library file: lib\_hiae.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetMEAttr

[Description]

Sets ME attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetMEAttr(const ISP_ME_ATTR_S *pstMEAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstMEAttr	Attributes of ME parameters and ME enable parameters	Input

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- ME enable parameters include exposure time enable, analog gain enable, and digital gain enable.
- ME parameters include exposure time, analog gain, and digital gain.
- When ME enable parameters are valid, the corresponding exposure parameters must be configured.
- The ME time is measured by the number of scanned lines of the sensor.
- The ME analog gain and digital gain are measured by multiples.
- If the value of an ME parameter is greater than the maximum value supported by the sensor, the maximum value is used; if the value of an ME parameter is smaller than the minimum value supported by the sensor, the minimum value is used.

[Example]

None

[See Also]

HI\_MPI\_ISP\_ISP\_GetMEAttrEx

## HI\_MPI\_ISP\_GetMEAttr

[Description]

Obtains ME attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetMEAttr(ISP\_ME\_ATTR\_S*pstMEAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstMEAttr	Attributes of ME parameters and ME enable	Output



Parameter	Description	Input/Output
	parameters	

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

If the configured ME parameter value does not fall within the value range, a value within the value range is obtained.

[Example]

None

[See Also]

- [HI\\_MPI\\_ISP\\_SetMEAttr](#)
- [HI\\_MPI\\_ISP\\_SetExposureType](#)

## HI\_MPI\_ISP\_SetMEAttrEx

[Description]

Sets extended ME attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetMEAttrEx(const ISP_ME_ATTR_EX_S *pstMEAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
pstMEAttrEx	Extended attributes of ME parameters and ME enable parameters	Input



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- Compared with HI\_MPI\_ISP\_SetMEAttrm, HI\_MPI\_ISP\_SetMEAttrEx allows you to set 10-bit gains.
- ME enable parameters include exposure time enable, analog gain enable, and digital gain enable.
- Because the ISP digital gain is not supported, the ISP digital gain needs to be set to 1x in the **XX\_CMOS.C** driver.
- ME parameters include exposure time, analog gain, and digital gain.
- When ME enable parameters are valid, the corresponding exposure parameters must be set.
- The ME time is measured by the number of scanned lines of the sensor.
- The ME analog gain and digital gain are measured by multiples.
- If the value of an ME parameter is greater than the maximum value supported by the sensor, the maximum value is used; if the value of an ME parameter is smaller than the minimum value supported by the sensor, the minimum value is used.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetMEAttrEx](#)

## HI\_MPI\_ISP\_GetMEAttrEx

[Description]

Obtains extended ME attributes.

[Syntax]



```
HI_S32 HI_MPI_ISP_GetMEAttrEx(ISP_ME_ATTR_S_EX *pstMEAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
pstMEAttrEx	Extended attributes of ME parameters and ME enable parameters	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

If the configured ME parameter value does not fall within the value range, a value within the value range is obtained.

[Example]

None

[See Also]

- [HI\\_MPI\\_ISP\\_SetExposureType](#)
- [HI\\_MPI\\_ISP\\_SetMEAttrEx](#)

## HI\_MPI\_ISP\_SetExpStaInfo

[Description]

Sets the AE statistics.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetExpStaInfo(ISP\_EXP\_STA\_INFO\_S *pstExpStatistic);
```

[Parameter]



Parameter	Description	Input/Output
pstExpStatistic	Exposure statistics	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI is used to set the segment positions for the 5-segment histogram.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetExpStaInfo](#)

## HI\_MPI\_ISP\_GetExpStaInfo

[Description]

Obtains AE statistics such as the information about the 256-segment histogram, 5-segment histogram, block histogram, and average luminance.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetExpStaInfo(ISP\_EXP\_STA\_INFO\_S *pstExpStatistic);
```

[Parameter]

Parameter	Description	Input/Output
pstExpStatistic	Exposure statistics	Output



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetExpStaInfo](#)

## HI\_MPI\_ISP\_QueryInnerStateInfo

[Description]

Obtains the internal status information.

[Syntax]

```
HI_S32 HI_MPI_ISP_QueryInnerStateInfo(ISP_INNER_STATE_INFO_S  
*pstInnerStateInfo);
```

[Parameter]

Parameter	Description	Input/Output
pstInnerStateInfo	Internal status information	Output

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_QueryInnerStateInfoEx

[Description]

Obtains the internal status information.

[Syntax]

```
HI_S32 HI_MPI_ISP_QueryInnerStateInfoEx(ISP_INNER_STATE_INFO_EX_S  
*pstInnerStateInfoEx);
```

[Parameter]

Parameter	Description	Input/Output
pstInnerStateInfoEx	Internal status information	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.





[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

None

[Example]

None

[See Also]

None

### 3.4.3 AI Control MPIs

The following are AI control MPIs:

- [HI\\_MPI\\_ISP\\_SetIrisType](#): Sets the iris control type.
- [HI\\_MPI\\_ISP\\_GetIrisType](#): Obtains the iris control type.
- [HI\\_MPI\\_ISP\\_SetAIAttr](#): Sets the AI control attribute.
- [HI\\_MPI\\_ISP\\_GetAIAttr](#): Obtains the AI control attribute.

#### HI\_MPI\_ISP\_SetIrisType

[Description]

Sets the iris control type.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetIrisType(ISP\_OP\_TYPE\_E enIrisType);
```

[Parameter]

Parameter	Description	Input/Output
<code>enIrisType</code>	Iris control type	Input

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
HI_SUCCESS	Success.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This function is not supported currently.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetIrisType](#)

## HI\_MPI\_ISP\_GetIrisType

[Description]

Obtains the iris control type.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetIrisType(ISP\_OP\_TYPE\_E *penIrisType);
```

[Parameter]

Parameter	Description	Input/Output
penIrisType	Iris control type	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]



Error Code	Description
HI_SUCCESS	Success.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetIrisType](#)

## HI\_MPI\_ISP\_SetAIAttr

[Description]

Sets the AI control attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAIAttr(const ISP\_AI\_ATTR\_S*pstAIAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAIAttr	AI attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]



- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI is used to enable AI control function and implement iris correction. Iris correction allows you to obtain the board voltage for stopping the iris. The ambient luminance must be stable when the correction program starts for successful iris correction.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetIrisType](#)

## HI\_MPI\_ISP\_GetAIAttr

[Description]

Obtains the AI control attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAIAttr(ISP\_AI\_ATTR\_S*pstAIAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAIAttr	AI attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]



None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetAIAAttr](#)

## 3.5 Data Structures

### 3.5.1 Registration

The following are registration-related data structures:

- [AE\\_SENSOR\\_REGISTER\\_S](#): Defines sensor registration information.
- [AE\\_SENSOR\\_GAININFO\\_S](#): Defines the sensor gain table.
- [AE\\_SENSOR\\_EXP\\_FUNC\\_S](#): Defines the sensor callback function.
- [AE\\_SENSOR\\_DEFAULT\\_S](#): Defines the initialization parameter for the AE algorithm library.
- [AE\\_ACCURACY\\_E](#): Defines the accuracy type of the exposure time and gain.
- [AE\\_ACCURACY\\_S](#): Defines the accuracy of the exposure time and gain.

#### AE\_SENSOR\_REGISTER\_S

[Description]

Defines sensor registration information.

[Syntax]

```
typedef struct hiAE_SENSOR_REGISTER_S
{
    AE_SENSOR_EXP_FUNC_S stSnsExp;
} AE_SENSOR_REGISTER_S;
```

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]

[AE\\_SENSOR\\_EXP\\_FUNC\\_S](#)



## AE\_SENSOR\_GAININFO\_S

### [Description]

Defines the sensor gain table.

### [Syntax]

```
typedef struct hiAE_SENSOR_GAININFO_S
{
    HI_U32  u32SnsTimes; //10bit precision
    HI_U32  u32GainDb;   // gain step in table
} AE_SENSOR_GAININFO_S;
```

### [Member]

Member	Description
u32SnsTimes	Gain multiple
u32GainDb	Index of the value in the gain table

### [Note]

- **u32SnsTimes** indicates the sensor multiple and its precision is 10 bits. That is, the value **1024** indicates 1x gain.
- **u32GainDb** indicates the index of the value queried in the gain table.

### [See Also]

AE\_SENSOR\_GAININFO\_S

## AE\_SENSOR\_EXP\_FUNC\_S

### [Description]

Defines the sensor callback function.

### [Syntax]

```
typedef struct hiAE_SENSOR_EXP_FUNC_S
{
    typedef struct hiAE_SENSOR_EXP_FUNC_S
    {
        HI_S32(*pfn_cmos_get_ae_default)(AE_SENSOR_DEFAULT_S *pstAeSnsDft);
        HI_VOID(*pfn_cmos_fps_set)(HI_U8 u8Fps, AE_SENSOR_DEFAULT_S
        *pstAeSnsDft);
        HI_VOID(*pfn_cmos_slow_framerate_set)(HI_U8 u8SlowFrameRate,
        AE_SENSOR_DEFAULT_S *pstAeSnsDft);
        HI_VOID(*pfn_cmos_inttime_update)(HI_U32 u32IntTime);
        HI_VOID(*pfn_cmos_gains_update)(HI_U32 u32Again, HI_U32 u32Dgain);
        HI_VOID(*pfn_cmos_again_calc_table)(HI_U32 u32InTimes,
```



```
AE_SENSOR_GAININFO_S *pstAeSnsGainInfo);  
    HI_VOID (*pfn_cmos_dgain_calc_table)(HI_U32 u32InTimes,  
AE_SENSOR_GAININFO_S *pstAeSnsGainInfo);} AE_SENSOR_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_cmos_get_ae_default	Pointer to the callback function for obtaining the initial value of the AE algorithm library
pfn_cmos_fps_set	Pointer to the callback function for setting the frame rate of a sensor
pfn_cmos_slow_framerate_set	Pointer to the callback function for reducing the frame rate of a sensor
pfn_cmos_inttime_update	Pointer to the callback function for setting the exposure time of a sensor
pfn_cmos_gains_update	Pointer to the callback function for setting the analog and digital gains of a sensor
pfn_cmos_again_calc_table	Pointer of the callback function for setting the analog gain table of a sensor
pfn_cmos_dgain_calc_table	Pointer of the callback function for setting the digital gain table of a sensor

[Note]

- If a callback function pointer is not required, set it to **NULL**.
- For details about frame rate reduction, see HI\_MPI\_ISP\_SetSlowFrameRate.
- The accuracy of the exposure time and gain is defined AE\_SENSOR\_DEFAULT\_S. The exposure time and gain configured in pfn\_cmos\_inttime\_update and pfn\_cmos\_gains\_update respectively are the values with accuracy.

[See Also]

ISP\_SENSOR\_REGISTER\_S

## AE\_SENSOR\_DEFAULT\_S

[Description]

Defines the initialization parameter for the AE algorithm library.

[Syntax]

```
typedef struct hiAE_SENSOR_DEFAULT_S  
{  
    HI_U8    au8HistThresh[4];  
    HI_U8    u8AeCompensation;  
  
    HI_U32    u32LinesPer500ms;
```



```

HI_U32  u32FlickerFreq;

HI_U32  u32MaxIntTime;    /* unit is line */
HI_U32  u32MinIntTime;
HI_U32  u32MaxIntTimeTarget;
HI_U32  u32MinIntTimeTarget;
AE_ACCURACY_S  stIntTimeAccu;

HI_U32  u32MaxAgain;
HI_U32  u32MinAgain;
HI_U32  u32MaxAgainTarget;
HI_U32  u32MinAgainTarget;
AE_ACCURACY_S  stAgainAccu;

HI_U32  u32MaxDgain;
HI_U32  u32MinDgain;
HI_U32  u32MaxDgainTarget;
HI_U32  u32MinDgainTarget;
AE_ACCURACY_S  stDgainAccu;
} AE_SENSOR_DEFAULT_S;

```

[Member]

Member	Description
au8HistThresh	Segmentation threshold array of the 5-segment histogram. The value range is [0, 255]. The default value is {0xd, 0x28, 0x60, 0x80} in linear mode or {0x20, 0x40, 0x60, 0x80} in WDR mode. The default value is recommended.
u8AeCompensation	Target AE luminance. The value range is [0, 255], and the value 0x80 is recommended.
u32LinesPer500ms	Total number of lines per 500 ms
u32FlickerFreq	Anti-flicker frequency, which is 256 times the current power frequency
u32MaxIntTime	Maximum exposure time (in line)
u32MinIntTime	Minimum exposure time (in line)
u32MaxIntTimeTarget	Target maximum exposure time (in line)
u32MinIntTimeTarget	Target minimum exposure time (in line)
stIntTimeAccu	Exposure time accuracy
u32MaxAgain	Maximum analog gain (measured by multiple)
u32MinAgain	Minimum analog gain (measured by multiple)
u32MaxAgainTarget	Target maximum analog gain (measured by multiple)





u32MinAgainTarget	Target minimum analog gain (measured by multiple)
stAgainAccu	Analog gain accuracy
u32MaxDgain	Maximum digital gain (measured by multiple)
u32MinDgain	Minimum digital gain (measured by multiple)
u32MaxDgainTarget	Target maximum digital gain (measured by multiple)
u32MinDgainTarget	Target minimum digital gain (measured by multiple)
stDgainAccu	Digital gain accuracy

[Note]

None

[See Also]

ISP\_SENSOR\_EXP\_FUNC\_S

## AE\_ACCURACY\_E

[Description]

Defines the accuracy type of the exposure time and gain.

[Syntax]

```
typedef enum hiAE_ACCURACY_E
{
    AE_ACCURACY_DB = 0,
    AE_ACCURACY_LINEAR,
    AE_ACCURACY_TABLE,
    AE_ACCURACY_BUTT,
} AE_ACCURACY_E;
```

[Member]

Member	Description
AE_ACCURACY_DB	DB accuracy
AE_ACCURACY_LINEAR	Linear accuracy
AE_ACCURACY_TABLE	Table type

[Note]

None

[See Also]

AE\_ACCURACY\_S



## AE\_ACCURACY\_S

### [Description]

Defines the accuracy of the exposure time and gain.

### [Syntax]

```
typedef struct hiAE_ACCURACY_S
{
    AE_ACCURACY_E enAccuType;
    float    f32Accuracy;
} AE_ACCURACY_S;
```

### [Member]

Member	Description
enAccuType	Accuracy type, including linear accuracy, DB accuracy and TABLE accuracy.
f32Accuracy	Accuracy

### [Note]

- The accuracy of most gains is classified into linear accuracy, DB accuracy and TABLE accuracy.
- The linear accuracy indicates that the gain multiple increases evenly. For example, if **again** supported by the sensor is 1x, 1 (1 + 1/16)x, 1 (1 + 2/16)x, ..., and 1 (1 + N/16)x, the accuracy type can be set to **AE\_ACCURACY\_LINEAR**, and the accuracy can be set to **0.0625**. In this case, if **again** is **32**, the gain is 2x (32 x 0.0625).
- The DB accuracy indicates that gain multiple is doubled. For example, if **again** supported by the sensor is 0 dB, 0.3 dB, 0.6 dB, ..., and (0.3 x N) dB, the accuracy type can be set to **AE\_ACCURACY\_DB**, and the accuracy can be set to **0.3**. In this case, if **again** is **80**, the gain is 16x (80 x 0.3 = 24 dB). If the **again** supported by the sensor is 1x, 2x, 4x, 8x, ..., and 2<sup>N</sup>x (corresponding to 0 dB, 6 dB, 12 dB, 18 dB, ..., and 6N dB), the accuracy type is set to **AE\_ACCURACY\_DB**, and the accuracy can be set to **6**.
- The table precision indicates the gain multiple and is obtained by querying the gain table. The table precision is fixed at 10 bits, that is, the value **1024** indicates 1x gain. When the gains of some sensors increase in non-linear mode, the required analog gain or digital gain can be calculated by using the AE algorithm. The value that is queried from the sensor gain table and closest to the calculated value is used as the digital gain or analog gain.
- When the gain table is used, the callback function in the initialization callback data structure **AE\_SENSOR\_EXP\_FUCN\_S** is required.
- Typically, the accuracy of the number of exposure lines is linear accuracy and its unit is line.
- After the accuracy is defined, most sensors can be connected by using unified interfaces.

### [See Also]

ISP\_SENSOR\_REGISTER\_S



## 3.5.2 AE

The following are AE data structures:

- [ISP\\_AE\\_MODE\\_E](#): Defines the AE mode.
- [ISP\\_AE\\_ATTR\\_S](#): Defines the AE attributes of the ISP.
- [ISP\\_AE\\_ATTR\\_EX\\_S](#): Defines the extended AE attributes of the ISP (sensor gain with 10-bit precision).
- [ISP\\_ME\\_ATTR\\_S](#): Defines the ME attributes of the ISP.
- [ISP\\_ME\\_ATTR\\_EX\\_S](#): Defines the extended ME attributes of the ISP (sensor gain with 10-bit precision).
- [ISP\\_EXP\\_STA\\_INFO\\_S](#): Defines the ISP exposure statistics.
- [ISP\\_INNER\\_STATE\\_INFO\\_S](#): Defines the internal status information about the ISP.
- [ISP\\_INNER\\_STATE\\_INFO\\_EX\\_S](#): Defines the extended internal status information about the ISP (sensor gain with 10-bit precision).
- [ISP\\_AE\\_ROUTE\\_NODE\\_S](#): Defines the attributes of the AE route node.
- [ISP\\_AE\\_ROUTE\\_S](#): Defines the attributes of the ISP exposure allocation policy.
- [ISP\\_AE\\_DELAY\\_S](#): Defines the ISP exposure delay attributes.

### ISP\_AE\_MODE\_E

[Description]

Defines the AE mode.

[Syntax]

```
typedef enum hiISP_AE_MODE_E
{
    AE_MODE_LOW_NOISE          = 0,
    AE_MODE_FRAME_RATE         = 1,
    AE_MODE_BUTT
} ISP_AE_MODE_E;
```

[Member]

Member	Description
AE_MODE_LOW_NOISE	Noise preference mode
AE_MODE_FRAME_RATE	Frame rate preference mode

[Note]

- The noise preference mode indicates that the priority is given to prolong the exposure time to reduce the gain value during AE adjustment. When the sensor gain is reached to the configured maximum value, the frame rate gradually decreases and the exposure time is prolonged during AE adjustment until the exposure time is the maximum AE time. In low-illumination scenarios, the noises are small but the frame rate decreases. When anti-flicker is enabled in noise preference mode, the exposure time continuously



increases but does not increase by a multiple such as 1/100 ms or 1/120 ms after the frame rate decreases.

- Frame rate preference mode indicates that the frame rate is ensured during AE adjustment. In low-illumination scenarios, the noises are large.

[See Also]

None

## ISP\_AE\_ATTR\_S

[Description]

Defines the AE attributes of the ISP.

[Syntax]

```
typedef struct hiISP_AE_ATTR_S
{
    ISP_AE_MODE_E enAEMode;
    HI_U16 u16ExpTimeMax;
    HI_U16 u16ExpTimeMin;
    HI_U16 u16DGainMax;
    HI_U16 u16DGainMin;
    HI_U16 u16AGainMax;
    HI_U16 u16AGainMin;
    HI_U8 u8ExpStep;
    HI_S16 s16ExpTolerance;
    HI_U8 u8ExpCompensation;
    ISP_AE_FRAME_END_UPDATE_E enFrameEndUpdateMode;
    HI_BOOL bByPassAE;
    HI_U8 u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_AE_ATTR_S;
```

[Member]

Member	Description
enAEMode	Preference mode for AE, for example, frame rate preference mode or noise preference mode
u16ExpTimeMax	Maximum exposure time for AE. This member limits the maximum exposure time for AE. If an object is moving, you can set the member to a smaller value. The value range is [0x2, 0xFFFF], which depends on the sensor.
u16ExpTimeMin	Minimum exposure time for AE The value range is [0x2, 0xFFFF], which depends on the sensor.
u16DGainMax	Maximum digital gain value for AE



Member	Description
	The value range is [0x1, 0xFF], which depends on the sensor.
u16DGainMin	Minimum digital gain value for AE The value range is [0x1, 0xFF], which depends on the sensor.
u16SystemGainMax	Maximum system gain including the maximum digital gain and analog gain of the sensor and the maximum digital gain of the ISP. <b>u16SystemGainMax</b> is used with <b>u16SystemGainShift</b> .
u16SystemGainShift	Accuracy of the maximum system gain
u8ExpStep	Initial step during AE adjustment
s16ExpTolerance	Exposure tolerance during AE adjustment The value range is [0x0, 0xFFFF].
u8ExpCompensation	Exposure compensation during AE adjustment
enFrameEndUpdateMode	Mode of updating exposure variables
bByPassAE	Whether to use the AE algorithm
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	Weight table for AE

[Note]

- Maximum and minimum exposure time and gain value for AE  
You can specify the exposure time and gain value based on the application scenario. If objects move rapidly, you can set a short exposure time. This improves picture smearing. Currently, the minimum digital gain cannot be set.
  - If **enAEMode** is **LowNoise**, the minimum exposure time can be set to the maximum exposure time of a frame at the normal frame rate.
  - If **enAEMode** is **FrameRate**, the minimum exposure time must be less than or equal to the maximum exposure time of a frame at the normal frame rate. The system limits the minimum exposure time to the time of a frame even the minimum exposure time is set to the maximum exposure time of more than one frame.
- Initial step during AE adjustment  
A larger step indicates that a higher exposure convergence speed and greater change of luminance during automatic exposure.
- Exposure tolerance during AE adjustment  
A larger tolerance indicates lower sensitivity to ambient luminance changes.
- Exposure compensation during AE adjustment  
Larger compensation indicates brighter object to be exposed and brighter picture.
- Mode of updating exposure variables



To prevent the picture from flickering during AE adjustment when the anti-flicker function is enabled, set **enFrameEndUpdateMode** to **ISP\_AE\_FRAME\_END\_UPDATE\_1** for the Pana34041 and IMX104 sensors, **ISP\_AE\_FRAME\_END\_UPDATE\_2** for the OV9712 and MT9P006 sensors, and **ISP\_AE\_FRAME\_END\_UPDATE\_0** (default value) for other sensors.

- Weight table for AE

The static AE statistics are divided into 7 x 9 zones. You can change the weight of each zone by setting the weight table. For example, increasing the weight of the central zone changes the luminance of the central zone, which results in the change of picture statistics.

[See Also]

None

## ISP\_AE\_ATTR\_EX\_S

[Description]

Defines the AE attributes of the ISP (sensor gain with 10-bit precision).

[Syntax]

```
typedef struct hiISP_AE_ATTR_EX_S
{
    ISP_AE_MODE_E enAEMode;
    HI_U32 u32ExpTimeMax;
    HI_U32 u32ExpTimeMin;
    HI_U32 u32DGainMax;
    HI_U32 u32DGainMin;
    HI_U32 u32AGainMax;
    HI_U32 u32AGainMin;
    HI_U32 u32ISPDGainMax;
    HI_U32 u32SystemGainMax;
    HI_U8 u8ExpStep;
    HI_S16 s16ExpTolerance;
    HI_U8 u8ExpCompensation;
    ISP_AE_FRAME_END_UPDATE_E enFrameEndUpdateMode;
    HI_BOOL bByPassAE;
    HI_U8 u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_AE_ATTR_EX_S;
```

[Member]

Member	Description
enAEMode	Preference mode for AE, for example, frame rate preference mode or noise preference mode.
u32ExpTimeMax	Maximum exposure time for AE. This member limits the maximum exposure time for AE. If an object is moving, you can set the member to a smaller value.



Member	Description
	The value range is [0x2, 0xFFFF], which depends on the sensor.
u32ExpTimeMin	Minimum exposure time for AE The value range is [0x2, 0xFFFF], which depends on the sensor.
u32DGainMax	Maximum digital gain for AE (10-bit decimal accuracy) The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32DGainMin	Minimum digital gain for AE (10-bit decimal accuracy) The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32AGainMax	Maximum analog gain for AE (10-bit decimal accuracy) The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32AGainMin	Minimum analog gain for AE (10-bit decimal accuracy) The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32SystemGainMax	Maximum system gain (10-bit decimal precision) obtained by multiplying the maximum digital gain of the sensor, maximum analog gain of the sensor, and the maximum digital gain of the ISP. The value range is [0x400, 0xFFFFFFFF].
u8ExpStep	Initial step during AE adjustment
s16ExpTolerance	Exposure tolerance during AE adjustment The value range is [0x0, 0xFFFF].
u8ExpCompensation	Exposure compensation during AE adjustment
enFrameEndUpdateMode	Mode of updating exposure variables
bByPassAE	AE algorithm bypass
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	Weight table for AE

[Note]

- Maximum and minimum exposure time and gain value for AE  
You can specify the exposure time and gain value based on the application scenario. If objects move fast, you can set a short exposure time. This improves picture smearing. The minimum digital gain cannot be set currently.
- Initial step during AE adjustment



A larger step indicates that a higher exposure convergence speed and greater change of luminance during automatic exposure.

- Exposure tolerance during AE adjustment

A larger tolerance indicates lower sensitivity to ambient luminance changes.

- Exposure compensation during AE adjustment

Larger compensation indicates brighter object to be exposed and brighter picture.

- Mode of updating exposure variables

To prevent the picture from flickering the anti-flicker function is enabled, set **enFrameEndUpdateMode** to **ISP\_AE\_FRAME\_END\_UPDATE\_1** for the Pana34041 and IMX104 sensors, **ISP\_AE\_FRAME\_END\_UPDATE\_2** for the OV9712 and MT9P006 sensors, and **ISP\_AE\_FRAME\_END\_UPDATE\_0** (default value) for other sensors.

- Weight table for AE

The static AE statistics are divided into 7 x 9 zones. You can change the weight of each zone by setting the weight table. For example, increasing the weight of the central zone changes the luminance of the central zone, which results in the change of picture statistics.

[See Also]

None

## ISP\_ME\_ATTR\_S

[Description]

Defines the ME attributes of the ISP.

[Syntax]

```
typedef struct hiISP_ME_ATTR_S
{
    HI_S32 s32AGain;
    HI_S32 s32DGain;
    HI_U32 u32ExpLine;
    HI_BOOL bManualExpLineEnable;
    HI_BOOL bManualAGainEnable;
    HI_BOOL bManualDGainEnable;
} ISP_ME_ATTR_S;
```

[Member]

Member	Description
s32AGain	Manual analog gain The value range is [0x0, 0xFF], which depends on the sensor.
s32DGain	Manual digital gain The value range is [0x0, 0xFF], which depends on the sensor.
u32ExpLine	ME time





Member	Description
	The value range is [0x0, 0xFFFF], which depends on the sensor.
bManualExpLineEnable	ME time enable
bManualAGainEnable	Manual analog gain enable
bManualDGainEnable	Manual digital gain enable

[Note]

- When ME enable parameters are valid, the corresponding ME parameters must be configured.
- The ME time is measured by the number of scanned lines of the sensor.
- The ME analog gain and digital gain are measured by multiples.
- If the value of an ME parameter is greater than the maximum value supported by the sensor, the maximum value is used; if the value of an ME parameter is smaller than the minimum value supported by the sensor, the minimum value is used.

[See Also]

None

## ISP\_ME\_ATTR\_EX\_S

[Description]

Defines the extended ME attributes of the ISP (sensor gain with 10-bit precision).

[Syntax]

```
typedef struct hiISP_ME_ATTR_EX_S
{
    HI_S32 s32AGain;
    HI_S32 s32DGain;
    HI_U32 u32ExpLine;
    HI_BOOL bManualExpLineEnable;
    HI_BOOL bManualAGainEnable;
    HI_BOOL bManualDGainEnable;
} ISP_ME_ATTR_EX_S;
```

[Member]

Member	Description
s32AGain	Manual analog gain (10-bit precision). The value range is [0x400, 0x7FFFFFFF], which depends on the sensor.
s32DGain	Manual digital gain (10-bit precision). The value range is [0x400, 0x7FFFFFFF], which depends on the sensor.



Member	Description
u32ExpLine	ME time. The value range is [0x0, 0xFFFF], which depends on the sensor.
bManualExpLineEnable	ME time enable.
bManualAGainEnable	Manual analog gain enable.
bManualDGainEnable	Manual digital gain enable.

[Note]

- When ME enable parameters are valid, the corresponding ME parameters must be configured.
- The ME time is measured by the number of scanned lines of the sensor.
- The ME analog gain and digital gain are measured by multiples.
- If the value of an ME parameter is greater than the maximum value supported by the sensor, the maximum value is used; if the value of an ME parameter is smaller than the minimum value supported by the sensor, the minimum value is used.

[See Also]

None

## ISP\_EXP\_STA\_INFO\_S

[Description]

Defines the ISP exposure statistics.

[Syntax]

```
typedef struct hiISP_EXP_STA_INFO_S
{
    HI_U8  u8ExpHistThresh[4];
    HI_U16 u16ExpStatistic[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN][5];
    HI_U16 u16Exp_Hist256Value[256];
    HI_U16 u16Exp_Hist5Value[5];
    HI_U8  u8AveLum;
    HI_U8  u8ExpHistTarget[5];
    HI_S16 s16HistError;
} ISP_EXP_STA_INFO_S;
```

[Member]

Member	Description
u8ExpHistThresh[4]	Segment points of a 5-segment histogram Value range: [0x0, 0xFF]



Member	Description
u16ExpStatistic[ ][5]	Region-based statistics Value range: [0x0, 0xFFFF]
u16Exp_Hist256Value[256]	256-segment histogram statistics Value range: [0x0, 0xFFFF]
u16Exp_Hist5Value[5]	5-segment histogram statistics Value range: [0x0, 0xFFFF]
u8AveLum	Average luminance Value range: [0x0, 0xFF]
u8ExpHistTarget[5]	Segmented target value of the 5-segment histogram Value range: [0x0, 0xFF]
s16HistError	Difference (read-only) between the target AE luminance and the actual luminance. If it is a positive value, the target luminance is greater than the actual luminance; if it is a negative value, the target luminance is less than the actual luminance. Value range: [-0xFFFF, +0xFFFF]

[Note]

Based on the weight of the error between the value of **u8ExpHistTarget** and the pixel value of the actual image in the 5-segment histogram, the AE algorithm adjusts the exposure and gain to make the pixel value of the actual image in the 5-segment histogram be close to the value of **u8ExpHistTarget**. For example, if the values of **u8ExpHistTarget[0]** and **u8ExpHistTarget[1]** are large, indicating that a large number of pixels fall on the relatively dark area finally, the final brightness is low.

The speed of convergence from bright to dark or from dark to bright of the AE algorithm relies on the segmentation method of the 5-segment histogram. If the central value of the third segment differs greatly from 0x80, the speed of convergence from bright to dark is inconsistent with that from dark to bright. When adjusting the value of **u8ExpHistThresh** to adjust the 5-segment histogram, you also need to adjust the value of **u8ExpHistTarget** to ensure the expected pixel target value of each segment of histogram, thereby ensuring proper brightness.

[See Also]

None

## ISP\_INNER\_STATE\_INFO\_S

[Description]

Defines the internal status information about the ISP.

[Syntax]

```
typedef struct hiISP_EXP_STA_INFO_S
{
```



```
HI_U8 u8ExpHistThresh[4];  
HI_U16 u16ExpStatistic[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN][5];  
HI_U16 u16Exp_Hist256Value[256];  
HI_U16 u16Exp_Hist5Value[5];  
HI_U8 u8AveLum;  
HI_U8 u8ExpHistTarget[5];  
HI_S16 s16HistError;  
  
} ISP_EXP_STA_INFO_S;
```

[Member]

Member	Description
u32ExposureTime	Sensor exposure time. Value range: [0x0, 0xFFFF]
u32AnalogGain	Sensor analog gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32DigitalGain	Sensor digital gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32ISPDigitalGain	ISP digital gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32Exposure	Sensor exposure. Value range: [0x0, 0xFFFFFFFF]
u16AE_Hist256Value[256]	256-segment histogram. Value range: [0x0, 0xFFFF].
u16AE_Hist5Value[5]	5-segment histogram after region weighting. Value range: [0x0, 0xFFFF].
u8AveLum	Average picture luminance. Value range: [0x00, 0xFF].

[Note]

- The exposure time is measured by the number of scanned lines. The exposure is the product of the sensor exposure time, sensor analog gain, sensor digital gain, and ISP digital gain.
- The sensor digital gain and analog gain are measured by multiples.
- The average picture luminance is normalized, and the value range is 0–255.
- The iris status is not taken into account for checking whether the ISP reaches the maximum exposure performance.

[See Also]

None



## ISP\_INNER\_STATE\_INFO\_EX\_S

### [Description]

Defines the extended internal status information about the ISP (sensor gain with 10-bit precision).

### [Syntax]

```
typedef struct hiISP_INNER_STATE_INFO_EX_S
{
    HI_U32 u32ExposureTime;
    HI_U32 u32AnalogGain;
    HI_U32 u32DigitalGain;
    HI_U32 u32ISPDigitalGain;
    HI_U32 u32Exposure;
    HI_U16 u16AE_Hist256Value[256];
    HI_U16 u16AE_Hist5Value[5];
    HI_U8 u8AveLum;
    HI_BOOL bExposureIsMAX;
} ISP_INNER_STATE_INFO_S;
```

### [Member]

Member	Description
u32ExposureTime	Sensor exposure time. Value range: [0x0, 0xFFFF]
u32AnalogGain	Sensor analog gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32DigitalGain	Sensor digital gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32ISPDigitalGain	ISP digital gain (10-bit precision). Value range: [0x400, 0xFFFFFFFF]
u32Exposure	Sensor exposure. Value range: [0x0, 0xFFFFFFFF]
u16AE_Hist256Value[256]	256-segment histogram. Value range: [0x0, 0xFFFF].
u16AE_Hist5Value[5]	5-segment histogram after region weighting. Value range: [0x0, 0xFFFF].
u8AveLum	Average picture luminance. Value range: [0x00, 0xFF].
bExposureIsMAX	0: The ISP does not reach the maximum exposure performance.



Member	Description
	1: The ISP reaches the maximum exposure performance.

[Note]

- The exposure time is measured by the number of scanned lines. The exposure is the product of the sensor exposure time, sensor analog gain, sensor digital gain, and ISP digital gain.
- The sensor digital gain and analog gain are measured by multiples.
- The highly accurate sensor analog gain, sensor digital gain, and ISP digital gain are measured by multiples. They are used with the corresponding offset parameter. For example, in **u32AnalogGainFine**, the lower **u8AnalogGainFineShift** bits are the decimal part of the sensor analog gain, and the others are the integral part of the sensor analog gain.
- The 256-segment and 5-segment histograms are normalized, and their value range is 0–65535.
- The average picture luminance is normalized, and the value range is 0–255.
- The iris status is not taken into account for checking whether the ISP reaches the maximum exposure performance.

[See Also]

None

## ISP\_AE\_ROUTE\_NODE\_S

[Description]

Defines the attributes of the AE route node.

[Syntax]

```
typedef struct hiISP_AE_ROUTE_NODE_S
{
    HI_U32  u32IntTime;
    HI_U32  u32SysGain;
    HI_U32  u32ApePercent;
} ISP_AE_ROUTE_NODE_S;
```

[Member]

Member	Description
u32IntTime	Node exposure time. Value range: [0x0, 0xFFFF]
u32SysGain	Node gain (10-bit precision), including the sensor analog gain, sensor digital gain, and ISP digital gain. Value range: [0x400, 0xFFFFFFFF]
u32ApePercent	Node iris size. Only the P-iris is supported.



Member	Description
	Value range: [0x0, 0x64]

[Note]

- The node exposure is the product of the exposure time, gain, and iris. The node exposure is monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest.
- If two or more values of adjacent nodes change, the nodes must have the same exposure, that is, the product of the exposure time, gain, and iris of a node must be the same as those of another node.
- If the exposure of adjacent nodes increases, the value of a component should increase while the values of other components are retained. The component with increased value determines the allocation policy of the route. For example, if the value of the gain component increases, the allocation policy of the route is gain first.
- Only the P-iris is supported. The DC-iris is not supported, because it cannot be accurately controlled.

[See Also]

[HI\\_MPI\\_ISP\\_SetAERouteAttr](#)

## ISP\_AE\_ROUTE\_S

[Description]

Defines the attributes of the ISP exposure allocation policy.

[Syntax]

```
typedef struct hiISP_AE_ROUTE_S
{
    HI_U32 u32TotalNum;
    ISP_AE_ROUTE_NODE_S astRouteNode[ISP_AE_ROUTE_MAX_NODES];
} ISP_AE_ROUTE_S;
```

[Member]

Member	Description
u32TotalNum	Number of nodes on the exposure allocation route. A maximum of eight nodes are supported currently.
astRouteNode [ISP_AE_ROUTE_MAX_NODES]	Attributes of the nodes on the exposure allocation route

[Note]



- A maximum of eight nodes are supported. Each node has the exposure time, gain, and iris components. The gain includes the sensor analog gain, sensor digital gain, and ISP digital gain.
- You can set routes based on the application scenario. The allocation route can be dynamically switched.
- The exposure time and then gain are allocated by default. If the current exposure does not fall within the configured route range, the default allocation policy is used.

[See Also]

- [ISP\\_AE\\_ROUTE\\_NODE\\_S](#)
- [HI\\_MPI\\_ISP\\_SetAERouteAttr](#)

## ISP\_AE\_DELAY\_S

[Description]

Defines the ISP exposure delay attributes.

[Syntax]

```
typedef struct hiISP_AE_DELAY_S
{
    HI_U16 u16BlackDelayFrame;
    HI_U16 u16WhiteDelayFrame;
} ISP_AE_DELAY_S;
```

[Member]

Member	Description
u16BlackDelayFrame	AE adjustment starts when the picture luminance is less than the target luminance for <b>u16BlackDelayFrame</b> frames.
u16WhiteDelayFrame	AE adjustment starts when the picture luminance is greater than the target luminance for <b>u16WhiteDelayFrame</b> frames.

[Note]

- When the AE adjustment step is the same, larger values of **u16BlackDelayFrame** and **u16WhiteDelayFrame** indicate longer time for adjusting the luminance to the target value.
- Human eyes are sensitive to overexposure. It is recommended that the value of **u16WhiteDelayFrame** be less than **u16BlackDelayFrame**.

[See Also]

[HI\\_MPI\\_ISP\\_SetAEDelayAttr](#)

## 3.5.3 AI

The following are AI data structures:





- [ISP\\_OP\\_TYPE\\_E](#): Defines the AI mode of the ISP.
- [ISP\\_IRIS\\_STATUS\\_E](#): Defines the ISP iris status.
- [ISP\\_TRIGGER\\_STATUS\\_E](#): Defines the ISP correction or detection status.
- [ISP\\_AI\\_ATTR\\_S](#): Defines the AI attributes of the ISP.

## ISP\_OP\_TYPE\_E

### [Description]

Defines the AI mode of the ISP.

### [Syntax]

```
typedef struct hiISP_OP_TYPE_E
{
    OP_TYPE_AUTO = 0;
    OP_TYPE_MANUAL = 1;
    OP_TYPE_BUTT
} ISP_OP_TYPE_E;
```

### [Member]

Member	Description
OP_TYPE_AUTO	Automatic exposure mode
OP_TYPE_MANUAL	Manual exposure mode

### [Note]

Currently, the AI manual exposure mode is not supported. When the AE manual mode is used, the AI automatic exposure mode does not take effect.

### [See Also]

None

## ISP\_IRIS\_STATUS\_E

### [Description]

Defines the ISP iris status.

### [Syntax]

```
typedef enum hiISP_IRIS_STATUS_E
{
    ISP_IRIS_KEEP = 0,
    ISP_IRIS_OPEN = 1,
    ISP_IRIS_CLOSE = 2,
    ISP_IRIS_BUTT
} ISP_IRIS_STATUS_E;
```



[Member]

Member	Description
ISP_IRIS_KEEP	The current status of the iris is retained.
ISP_IRIS_OPEN	The iris is opened.
ISP_IRIS_CLOSE	The iris is closed.

[Note]

None

[See Also]

None

## ISP\_TRIGGER\_STATUS\_E

[Description]

Defines the ISP correction or detection status.

[Syntax]

```
typedef enum hiISP_TRIGGER_STATUS_E
{
    ISP_TRIGGER_INIT      = 0,
    ISP_TRIGGER_SUCCESS   = 1,
    ISP_TRIGGER_TIMEOUT   = 2,
    ISP_TRIGGER_BUTT
} ISP_TRIGGER_STATUS_E;
```

[Member]

Member	Description
ISP_TRIGGER_INIT	The ISP is initialized and no correction is performed.
ISP_TRIGGER_SUCCESS	Correction is successful.
ISP_TRIGGER_TIMEOUT	Correction ends due to timeout.

[Note]

None

[See Also]

None



## ISP\_AI\_ATTR\_S

### [Description]

Defines the AI attributes of the ISP.

### [Syntax]

```
typedef struct hiISP_AI_ATTR_S
{
    HI_BOOL bIrisEnable;
    HI_BOOL bIrisCalEnable;
    HI_U32  u32IrisHoldValue;
    ISP_IRIS_STATUS_E enIrisStatus;
    ISP_TRIGGER_STATUS_E enTriggerStatus;
    HI_U16 u16IrisStopValue;
    HI_U16 u16IrisCloseDrive;
    HI_U16 u16IrisTriggerTime;
    HI_U8  u8IrisInertiaValue;
} ISP_AI_ATTR_S;
```

### [Member]

Member	Description
bIrisEnable	AI enable
bIrisCalEnable	AI correction enable
u32IrisHoldValue	AI correction value Value range: [0x0, 0x3E8]
enIrisStatus	Iris status
enTriggerStatus	Result and status of iris correction
u16IrisStopValue	Initial value of the AI correction value Value range: [0x0, 0x3E8]
u16IrisCloseDrive	Drive value for closing the iris Value range: [0x0, 0x3E8]
u16IrisTriggerTime	AI correction timeout, in frame Value range: [0x0, 0xFFF]
u8IrisInertiaValue	Inertia time during AI reverse, in frame

### [Note]

- AI correction allows you to obtain the board voltage for stopping the iris. The ambient luminance must be stable for successful iris correction.
- The proper value of u16IrisStopValue increases the speed of AI correction.



- The recommended value range of u16IrisCloseDrive is [700, 900]. The greater the value is, the faster the iris is closed.
- If the time for AI correction is greater than or equal to u16IrisTriggerTime, AI correction ends due to timeout.
- AI does not enter the ON status immediately when the AI status is changed from OFF to ON. That is, AI retains the OFF status for a period of time, and then enters the ON status because of inertia. The default value of u8IrisInertiaValue is 5, and the recommended value range is [5, 10]. The greater the value is, the longer time required for AI correction.

[See Also]

- [ISP\\_IRIS\\_STATUS\\_E](#)
- [ISP\\_TRIGGER\\_STATUS\\_E](#)



# 4 AWB

## 4.1 Overview

The spectral components of visible light vary according to the color temperature. The white objects have a red cast at a low color temperature or have a blue cast at a high color temperature. Human eyes can identify the actual object color based on brain reflections. The AWB algorithm is used to reduce the impacts on the actual object color exerted by external light sources. This ensures that the captured color information is converted into the information without color cast under an ideal light source.

## 4.2 Important Concepts

The following are the concepts related to the AWB module:

- Color temperature: It is defined based on the absolute black body. When the radiation of a light source is the same as that of the absolute black body in the visible area, the temperature of the absolute black body is the color temperature.
- White balance: The white objects have a blue or red cast at different color temperatures. The white balance algorithm is used to adjust the strength of R, G, and B channels to obtain the actual white color.

## 4.3 Function Description

This section describes the function implementation of the AWB Module

The AWB module consists of the WB statistics module and AWB control policy algorithm firmware. The WB statistics module collects statistics on the average ratios of R, G, and B channel information output by the sensor. The WB statistics module can provide weighted ratio of the entire image or the ratio of each zone. The WB statistics module can also divide an image into M x N zones (M rows and N columns), and collect statistics on the average G/R and G/B values and number of white points for each zone.

$$awb\_rg_{mn} = \sum_{p \in \Omega_{mn}} \frac{G_p}{R_p} * \theta_p$$

$$awb_{mn} = \sum_{p \in \Omega_{mn}} \theta_p$$

In the preceding formulas,  $\theta$  indicates whether the current point is a white point;  $R$  indicates red component value of a white point;  $G$  indicates the green component value of a white point;  $awb$  is the number of white points;  $awb\_rg$  is the average G/R value.

The zone weights can be set and the weights of all zones are the same by default. The weighted global statistics is output.

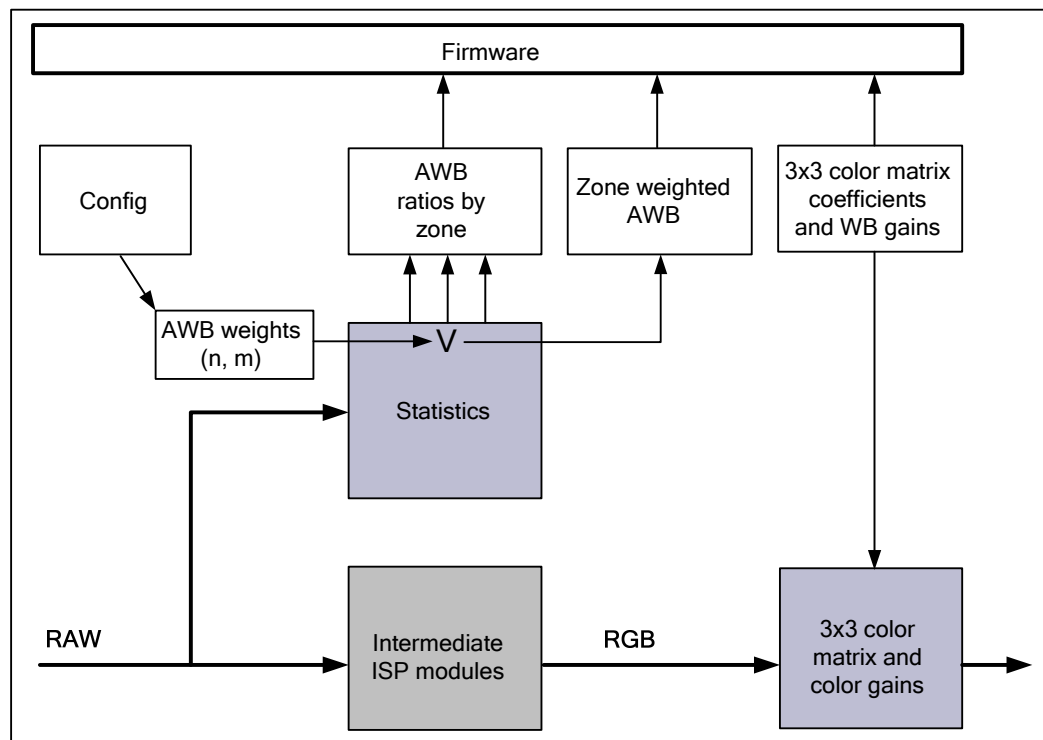
The statistics provided by the AWB module mainly include R/G and B/G values. The R/G and B/G values are average values. The average values are calculated based on the total R/G and B/G values of white points in each zone and AWB weight coefficient by using the following formula:

$$AWB\_RG = \frac{\sum_{mn} w_{mn} * awb\_rg_{mn}}{\sum_{mn} w_{mn} * awb_{mn}}$$

where  $awb\_rg$  is the average R/G value of white points in  $[m][n]$  zone, and  $awb$  is the total number of white points in  $[m][n]$  zone.

Figure 4-1 shows the schematic diagram of the AWB module.

**Figure 4-1** Schematic diagram of the AWB module





## 4.4 API Reference

### 4.4.1 AWB Algorithm Library MPIs

The following are AWB algorithm library MPIs:

- [HI\\_MPI\\_AWB\\_Register](#): Registers the AWB algorithm library with the ISP.
- [HI\\_MPI\\_AWB\\_SensorRegCallBack](#): Registers a sensor. This callback function is provided by the AWB algorithm library.

#### HI\_MPI\_AWB\_Register

[Description]

Registers the AWB algorithm library with the ISP.

[Syntax]

```
HI_S32 HI_MPI_AWB_Register(ALG_LIB_S *pstAwbLib);
```

[Parameter]

Parameter	Description	Input/Output
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_awb.h
- Library file: libisp.a

[Note]

- This MPI calls the AWB registration callback function `HI_MPI_ISP_AwbLibRegCallBack` provided by the ISP library to register the AWB algorithm library with the ISP library.
- You can call this MPI to register the AWB algorithm library with the ISP library.
- Multiple instances can be registered with the AWB algorithm library.

[Example]

None

[See Also]

`HI_MPI_ISP_AwbLibRegCallBack`

## HI\_MPI\_AWB\_SensorRegCallBack

### [Description]

Registers a sensor. This callback function is provided by the AWB algorithm library.

### [Syntax]

```
HI_S32 HI_MPI_AWB_SensorRegCallBack(ALG_LIB_S *pstAwbLib, SENSOR_ID
SensorId, AWB_SENSOR_REGISTER_S *pstRegister);
```

### [Parameter]

Parameter	Description	Input/Output
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input
SensorId	ID of the sensor that is registered with the AWB algorithm library	Input
pstRegister	Pointer to the data structure for registering a sensor	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

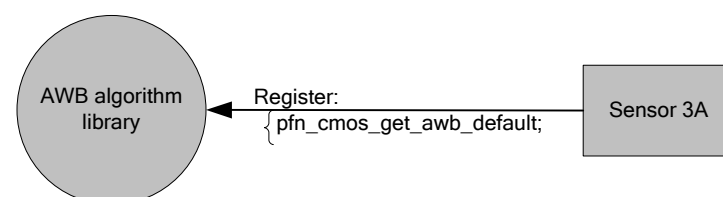
### [Requirement]

- Header files: hi\_comm\_isp.h, mpi\_awb.h
- Library file: libisp.a

### [Requirement]

- **SensorId** can be configured in the sensor library and is used to ensure that the sensor registered with the ISP library is the same as that registered with the 3A algorithm library.
- The AWB algorithm library obtains differentiated initialization parameters and controls sensors by using the callback interfaces registered by sensors.

**Figure 4-2** Interface between the AWB algorithm library and the sensor library







[Example]

```
ALG_LIB_S stLib;
AWB_SENSOR_REGISTER_S stAwbRegister;
AWB_SENSOR_EXP_FUNC_S *pstExpFuncs = &stAwbRegister.stSnsExp;

memset(pstExpFuncs, 0, sizeof(AWB_SENSOR_EXP_FUNC_S));
pstExpFuncs->pfn_cmos_get_awb_default = cmos_get_awb_default;

stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AWB_LIB_NAME);
s32Ret = HI_MPI_AWB_SensorRegCallBack(&stLib, IMX104_ID, &stAwbRegister);
if (s32Ret)
{
    printf("sensor register callback function to awb lib failed!\n");
    return s32Ret;
}
```

[See Also]

None

## 4.4.2 AWB Control MPIs

The following are AWB control MPIs:

- [HI\\_MPI\\_ISP\\_SetWBType](#): Sets the white balance (WB) control type.
- [HI\\_MPI\\_ISP\\_GetWBType](#): Obtains the WB control type.
- [HI\\_MPI\\_ISP\\_SetAWBAttr](#): Sets automatic AWB attributes.
- [HI\\_MPI\\_ISP\\_GetAWBAttr](#): Obtains automatic AWB attributes.
- [HI\\_MPI\\_ISP\\_SetMWBAAttr](#): Sets manual AWB attributes.
- [HI\\_MPI\\_ISP\\_GetMWBAAttr](#): Obtains manual AWB attributes.
- [HI\\_MPI\\_ISP\\_SetAWBAlgType](#): Sets the AWB algorithm type.
- [HI\\_MPI\\_ISP\\_GetAWBAlgType](#): Obtains the AWB algorithm type.
- [HI\\_MPI\\_ISP\\_SetAdvAWBAttr](#): Sets the advanced AWB algorithm attributes.
- [HI\\_MPI\\_ISP\\_GetAdvAWBAttr](#): Obtains the advanced AWB algorithm attributes.
- [HI\\_MPI\\_ISP\\_SetLightSource](#): Sets the attributes of separate illuminant points.
- [HI\\_MPI\\_ISP\\_GetLightSource](#): Obtains the attributes of separate illuminant points.

### HI\_MPI\_ISP\_SetWBType

[Description]

Sets the WB control type.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWBType(ISP\_OP\_TYPE\_E enWBType);
```

[Parameter]



Parameter	Description	Input/Output
enWBType	WB control type	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- If the WB control type is set to automatic mode, the white balance is adjusted automatically based on the WB algorithm.
- If the WB control type is set to manual mode, the AWB algorithm is invalid. In this case, you must set the values of Rgain and Bgain.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetWBType](#)

## HI\_MPI\_ISP\_GetWBType

[Description]

Obtains the WB control type.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWBType (ISP_OP_TYPE_E *penWBType);
```

[Parameter]

Parameter	Description	Input/Output
penWBType	WB control type	Output



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetWBType](#)

## HI\_MPI\_ISP\_SetAWBAttr

[Description]

Sets automatic AWB attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAWBAttr(const ISP\_AWB\_ATTR\_S*pstAWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAWBAttr	Automatic AWB attributes	Input

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI can be used to configure the AWB weight table. By setting the values of **u8RGStrength** and **u8BGStrength**, configure the upper and lower color temperature limits for the AWB algorithm, change the picture hue, or select the global AWB algorithm or zoned AWB algorithm.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetWBType](#)

## HI\_MPI\_ISP\_GetAWBAttr

[Description]

Obtains automatic AWB attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAWBAttr(ISP\_MWB\_ATTR\_S*pstAWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAWBAttr	Automatic AWB attributes	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.



[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetWBType](#)

## HI\_MPI\_ISP\_SetMWBAAttr

[Description]

Sets manual AWB attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetMWBAAttr(const ISP\_MWB\_ATTR\_S*pstMWBAAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstMWBAAttr	Manual AWB attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.



[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

When the AWB mode is set to manual, you can call this MPI to manually adjust AWB.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetWBType](#)

## HI\_MPI\_ISP\_GetMWBAAttr

[Description]

Obtains manual AWB attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetMWBAAttr(ISP\_MWB\_ATTR\_S*pstMWBAAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstMWBAAttr	Manual AWB attributes	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a



[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetWBType](#)

## HI\_MPI\_ISP\_SetAWBAlgType

[Description]

Sets the AWB algorithm type.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAWBAlgType(ISP_AWB_ALG_TYPE_E enALGType);
```

[Parameter]

Parameter	Description	Input/Output
enALGType	AWB algorithm type.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]



- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- If **enALGType** is **AWB\_ALG\_DEFAULT**, the AWB effect is the same as that of the SDK SPC070. If **enALGType** is **AWB\_ALG\_ADVANCE**, the AWB algorithm is the optimized one.
- If **enALGType** is **AWB\_ALG\_ADVANCE**, you are advised to adjust algorithm parameters by calling **HI\_MPI\_ISP\_SetAdvAWBAttr** to achieve the optimal effect.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetAWBAlgType](#)

## HI\_MPI\_ISP\_GetAWBAlgType

[Description]

Obtains the AWB algorithm type.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAWBAlgType (ISP_AWB_ALG_TYPE_E *penALGType);
```

[Parameter]

Parameter	Description	Input/Output
penALGType	AWB algorithm type.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Difference]





Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetAWBAlgType](#)

## HI\_MPI\_ISP\_SetAdvAWBAttr

[Description]

Sets the advanced AWB algorithm attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAdvAWBAttr(ISP_ADV_AWB_ATTR_S *pstAdvAWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAdvAWBAttr	Advanced AWB attributes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.



[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- This MPI takes effect only when **enALGType** is **AWB\_ALG\_ADVANCE**.
- The AWB tolerance and limitations on the color temperature curve can be set by calling this MPI.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetAdvAWBAttr](#)

## HI\_MPI\_ISP\_GetAdvAWBAttr

[Description]

Obtains the advanced AWB algorithm attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAdvAWBAttr(ISP_ADV_AWB_ATTR_S *pstAdvAWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAdvAWBAttr	Advanced AWB attributes.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]



Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetAdvAWBAttr](#)

## HI\_MPI\_ISP\_SetLightSource

[Description]

Sets the attributes of separate illuminant points.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetLightSource (ISP_AWB_ADD_LIGHTSOURCE_S  
*pstLightSource);
```

[Parameter]

Parameter	Description	Input/Output
pstLightSource	Attributes of separate illuminant points.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.



[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI takes effect only when **enALGType** is **AWB\_ALG\_ADVANCE**. A maximum of four separate illuminant points can be set.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetLightSource](#)

## HI\_MPI\_ISP\_GetLightSource

[Description]

Obtains the attributes of separate illuminant points.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetLightSource(ISP_AWB_ADD_LIGHTSOURCE_S  
*pstLightSource);
```

[Parameter]

Parameter	Description	Input/Output
pstLightSource	Attributes of separate illuminant points.	Output

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this MPI.
Hi3518	The Hi3518 supports this MPI.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetLightSource](#)

## 4.4.3 WB Statistics MPIs

The following are WB statistics MPIs:

- [HI\\_MPI\\_ISP\\_SetColorTemp](#): Sets the AWB color temperature.
- [HI\\_MPI\\_ISP\\_GetColorTemp](#): Obtains the AWB color temperature.
- [HI\\_MPI\\_ISP\\_SetWBStaInfo](#): Sets the white balance statistical information.
- [HI\\_MPI\\_ISP\\_GetWBStaInfo](#): Obtains the white balance statistical information.

### HI\_MPI\_ISP\_SetColorTemp

[Description]

Sets the AWB color temperature.

[Syntax]



```
HI_S32 HI_MPI_ISP_SetColorTemp(HI_U16 u16ColorTemp);
```

[Parameter]

Parameter	Description	Input/Output
u16ColorTemp	Color temperature, in °K	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This function is not supported currently.

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetColorTemp

[Description]

Obtains the AWB color temperature.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetColorTemp(HI_U16 *pu16ColorTemp);
```

[Parameter]

Parameter	Description	Input/Output
pu16ColorTemp	Current AWB color temperature, in °K	Output



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetWBStaInfo

[Description]

Sets the white balance statistical information.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWBStaInfo(ISP_WB_STA_INFO_S *pstWBStatistic);
```

[Parameter]

Parameter	Description	Input/Output
pstWBStatistic	White balance statistical information	Input

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetWBStaInfo](#)

## HI\_MPI\_ISP\_GetWBStaInfo

[Description]

Obtains the white balance statistical information.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWBStaInfo(ISP_WB_STA_INFO_S *pstWBStatistic);
```

[Parameter]

Parameter	Description	Input/Output
pstWBStatistic	White balance statistical information	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]





Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

The following is an example that demonstrates how to customize the AWB algorithm:

```
ISP_AWB_ATTR_S stAWBAttr;
ISP_MWB_ATTR_S stMWBAttr;
ISP_WB_STA_INFO_S stWBSTAInfo;
int i, j;

//Set WB to Manual mode, disable AWB
HI_MPI_ISP_SetWBType(OP_TYPE_MANUAL);

//Set WB gain to 1
stMWBAttr.ul6Rgain = 0x100;
stMWBAttr.ul6Bgain = 0x100;
stMWBAttr.ul6Ggain = 0x100;
HI_MPI_ISP_SetMWBAttr(&stMWBAttr);

//define white point range
stWBSTAInfo.ul6BlackLevel = 0x40;
stWBSTAInfo.ul6WhiteLevel = 0x3a0;
stWBSTAInfo.ul6CrMax = 0x200;
stWBSTAInfo.ul6CrMin = 0x80;
stWBSTAInfo.ul6CbMax = 0x200;
stWBSTAInfo.ul6CbMin = 0x80;
HI_MPI_ISP_SetWBStaInfo(&stWBSTAInfo);

while (1)
{
    //Get statistics of AWB, include global awb info and zoned awb info
    HI_MPI_ISP_GetWBStaInfo(&stWBSTAInfo);

    stMWBAttr.ul6Rgain = stWBSTAInfo.ul6GRgain;
    stMWBAttr.ul6Bgain = stWBSTAInfo.ul6GBgain;
```



```
stMWBAAttr.u16Ggain = 0x100;

//Set gain to WB registers if the statistics is reasonable
if (stWBSTAInfo.u32GSum > 0x1000)
{
    HI_MPI_ISP_SetMWBAAttr(&stMWBAAttr);
}
//Sleep 1 frame
usleep(40000);
}
```

[See Also]

[ISP\\_WB\\_ZONE\\_STA\\_INFO\\_S](#)

## 4.5 Data Structures

### 4.5.1 Registration

The following are registration-related data structures:

- [AWB\\_SENSOR\\_REGISTER\\_S](#): Defines sensor registration information.
- [AWB\\_SENSOR\\_EXP\\_FUNC\\_S](#): Defines the sensor callback function.
- [AWB\\_SENSOR\\_DEFAULT\\_S](#): Defines the initialization parameter for the AWB algorithm library.

#### AWB\_SENSOR\_REGISTER\_S

[Description]

Defines sensor registration information.

[Syntax]

```
typedef struct hiAWB_SENSOR_REGISTER_S
{
    AWB_SENSOR_EXP_FUNC_S stSnsExp;
} AWB_SENSOR_REGISTER_S;
```

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]



## AWB\_SENSOR\_EXP\_FUNC\_S

### AWB\_SENSOR\_EXP\_FUNC\_S

#### [Description]

Defines the sensor callback function.

#### [Syntax]

```
typedef struct hiAWB_SENSOR_EXP_FUNC_S
{
    HI_S32 (*pfn_cmos_get_awb_default) (AWB_SENSOR_DEFAULT_S *pstAwbSnsDft);
} AWB_SENSOR_EXP_FUNC_S;
```

#### [Member]

Member	Description
pfn_cmos_get_awb_default	Pointer to the callback function for obtaining the initial value of the AWB algorithm library

#### [Note]

None

#### [See Also]

ISP\_SENSOR\_REGISTER\_S

### AWB\_SENSOR\_DEFAULT\_S

#### [Description]

Defines the initialization parameter for the AWB algorithm library.

#### [Syntax]

```
typedef struct hiAWB_SENSOR_DEFAULT_S
{
    HI_U16  u16WbRefTemp;          /* reference color temperature for WB */
    HI_U16  aul6GainOffset[4];     /* gain offset for white balance */
    HI_S32  as32WbPara[6];        /* parameter for wb curve */
    AWB_AGC_TABLE_S  stAgcTbl;
    AWB_CCM_S  stCcm;
} AWB_SENSOR_DEFAULT_S;
```

#### [Member]



Member	Sub Member	Description
u16WbRefTemp	None	Color temperature for correcting the static white balance. The value range is [0, 0xFFFF].
au16GainOffset	None	Gain of the R, Gr, Gb, and B color channels for static white balance. The value range is [0, 0xFFFF].
as32WbPara	None	White balance parameter provided by the correction tool. The value range is [0, 0xFFFFFFFF].
stAgcTbl	bValid	Data validity identifier of the data structure. The value is 0 or 1.
	au8Saturation	Interpolation array for dynamically adjusting the saturation based on the gain. The value range is [0, 255].
stCcm	u16HighColorTemp	High color temperature. The value range is [0, 0xFFFF].
	au16HighCCM	High color temperature CCM. The value range is [0, 0xFFFF].
	u16MidColorTemp	Middle color temperature. The value range is [0, 0xFFFF].
	au16MidCCM	Middle color temperature CCM. The value range is [0, 0xFFFF].
	u16LowColorTemp	Low color temperature. The value range is [0, 0xFFFF].
	au16LowCCM	Low color temperature CCM. The value range is [0, 0xFFFF].

[Note]

None

[See Also]

ISP\_SENSOR\_EXP\_FUNC\_S

## 4.5.2 WB

The following are the data structures related to the AWB module:

- [ISP\\_AWB\\_ATTR\\_S](#): Defines the AWB attributes of the ISP.
- [ISP\\_AWB\\_CALIBRATION\\_S](#): Stores AWB calibration parameters for the ISP.
- [ISP\\_MWB\\_ATTR\\_S](#): Defines the manual AWB attributes of the ISP.
- [ISP\\_WB\\_ZONE\\_STA\\_INFO\\_S](#): Defines the white balance statistical information about zones.
- [ISP\\_WB\\_STA\\_INFO\\_S](#): Defines the white balance statistical information.



- [ISP\\_AWB\\_ALG\\_TYPE\\_E](#): Defines the AWB algorithm type.
- [ISP\\_ADV\\_AWB\\_ATTR\\_S](#): Defines the advanced AWB attributes.
- [ISP\\_AWB\\_LIGHTSOURCE\\_INFO\\_S](#): Defines the attributes of an illuminant point.
- [ISP\\_AWB\\_ADD\\_LIGHTSOURCE\\_S](#): Defines the attributes of separate illuminant points.

## ISP\_AWB\_ATTR\_S

### [Description]

Defines the AWB attributes of the ISP.

### [Syntax]

```
typedef struct hiISP_AWB_ATTR_S
{
    ISP_AWB_CALIBRATION_S stAWBCalibration;
    HI_U16 u16Speed;
    HI_U8 u8Speed;
    HI_U8 u8RGStrength;
    HI_U8 u8BGStrength;
    HI_U8 u8ZoneSel;
    HI_U8 u8HighColorTemp;
    HI_U8 u8LowColorTemp;
    HI_U8 u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_AWB_ATTR_S;
```

### [Member]

Member	Description
stAWBCalibration	AWB calibration
u16Speed	AWB convergence speed The default value is 0x40.
u8RGStrength	Red-green (RG) strength for AWB
u8BGStrength	Blue-green (BG) strength for AWB
u8ZoneSel	Automatic AWB algorithm selection
u8HighColorTemp	Upper color temperature limit for the AWB algorithm
u8LowColorTemp	Lower color temperature limit for the AWB algorithm
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	AWB weight table Value range: [0, 15]

### [Difference]



Chip	Description
Hi3516	<p>#define WEIGHT_ZONE_ROW 7</p> <p>#define WEIGHT_ZONE_COLUMN 9</p> <p>The AWB statistics module can divide an image into 7 x 9 zones, and output the AWB statistics of each zone.</p> <p>If <b>u8ZoneSel</b> is 0 or greater than or equal to 63, the global AWB algorithm is selected. If <b>u8ZoneSel</b> ranges from 1 to 62, the zoned AWB algorithm is selected. The statistical results of only first <b>u8ZoneSel</b> sorted zones involve AWB correction. The specific value is not affected.</p>
Hi3518	<p>#define WEIGHT_ZONE_ROW 15</p> <p>#define WEIGHT_ZONE_COLUMN 17</p> <p>The AWB statistics module can divide an image into 15 x 17 zones, and output the AWB statistics of each zone.</p> <p>If <b>u8ZoneSel</b> is 0 or 255, the global AWB algorithm is selected. If <b>u8ZoneSel</b> ranges from 1 to 254, the zoned AWB algorithm is selected. The statistical results of only first <b>u8ZoneSel</b> sorted zones involve AWB correction. The specific value is not affected.</p>

[Note]

- RG strength and BG strength for AWB  
You can adjust the strength of the R component and B component by adjusting the RG strength and BG strength. In this way, cool hue or warm hue is obtained.
- The automatic AWB algorithm is selected as follows:
  - If **u8ZoneSel** is 0 or is greater than or equal to 0x3F, select the global automatic AWB algorithm. The 7 x 9 zones are involved in the AWB algorithm statistics.
  - If **u8ZoneSel** ranges from 1 to 0x3E, select the zoned AWB algorithm. The value 0x10 is recommended. In this case, the AWB effect is good, but the CPU usage is high.
- Upper color temperature limit for the AWB algorithm  
The highest or lowest color temperature supported by the AWB algorithm. If the color temperature does not fall within the valid color temperature range, the picture has obvious color casts.
- AWB weight table  
The static AWB statistics are divided into 7 x 9 zones after the weighted average operation. By setting the weight table, you can change the impact on the statistics caused by each zone. For example, if you focus on the white balance effect of the central zone, you can increase its weighted value.

[See Also]

None

## ISP\_AWB\_CALIBRATION\_S

[Description]



Stores AWB calibration parameters for the ISP.

[Syntax]

```
typedef struct hiISP_AWB_CALIBRATION_S
{
    HI_S32 as32CurvePara[6];
    HI_U16 au16StaticWB[4];
    HI_U16 u16RefTemp;
} ISP_AWB_CALIBRATION_S;
```

[Member]

Member	Description
as32CurvePara[6]	AWB curve calibration Value range: [-0x80000000, +0x7FFFFFFF] The default value is defined by the <b>wb_para[6]</b> member of <b>cmos_isp_default_t</b> in the <b>cmos.c</b> file.
au16StaticWB[4]	Static white balance calibration Value range: [0x0, 0xFFFF] The default value is defined by the <b>gain_offset [4]</b> member of <b>cmos_isp_default_t</b> in the <b>cmos.c</b> file.
u16RefTemp	Reference AWB color temperature Value range: [0x0, 0xFFFF] The default value is defined by the <b>wb_ref_temp</b> member of <b>cmos_isp_default_t</b> in the <b>cmos.c</b> file.

## ISP\_MWB\_ATTR\_S

[Description]

Defines the manual AWB attributes of the ISP.

[Syntax]

```
typedef struct hiISP_MWB_ATTR_S
{
    HI_U16 u16Rgain;
    HI_U16 u16Ggain;
    HI_U16 u16Bgain;
} ISP_MWB_ATTR_S;
```

[Member]

Member	Description
u16Rgain	Red gain for manual AWB



Member	Description
	Value range: [0x0, 0xFFFF]
u16Ggain	Green gain for manual AWB Value range: [0x0, 0xFFFF]
u16Bgain	Blue gain for manual AWB Value range: [0x0, 0xFFFF]

[Note]

None

[See Also]

None

## ISP\_WB\_ZONE\_STA\_INFO\_S

[Description]

Defines the white balance statistical information about zones.

[Syntax]

```
typedef struct hiISP_WB_ZONE_STA_INFO_S
{
    HI_U16 u16Rg;
    HI_U16 u16Bg;
    HI_U32 u32Sum;
} ISP_WB_ZONE_STA_INFO_S;
```

[Member]

Member	Description
u16Rg	Average G/R value of the white points in zones. The data format is 4.8-bit fixed-point Value range: [0x0, 0xFFFF]
u16Bg	Average G/B value of the white points in zones. The data format is 4.8-bit fixed-point Value range: [0x0, 0xFFFF]
u32Sum	Number of white points in zones Value range: [0x0, 0xFFFFFFFF]

[Note]

A picture is divided into M x N zones. The statistical information about the white points in zones is output.





[See Also]

None

## ISP\_WB\_STA\_INFO\_S

[Description]

Defines the white balance statistical information.

[Syntax]

```
typedef struct hiISP_WB_STA_INFO_S
{
    HI_U16 u16WhiteLevel;
    HI_U16 u16BlackLevel;
    HI_U16 u16CbMax;
    HI_U16 u16CbMin;
    HI_U16 u16CrMax;
    HI_U16 u16CrMin;
    HI_U16 u16GRgain;
    HI_U16 u16GBgain;
    HI_U32 u32GSum;
    HI_U32 u32Rgain;
    HI_U32 u32Ggain;
    HI_U32 u32Bgain;
    ISP_WB_ZONE_STA_INFO_S stZoneSta[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_WB_STA_INFO_S;
```

[Member]

Member	Description
u16WhiteLevel	Upper limit of the white point luminance Value range: (u16BlackLevel, 0x3FF]
u16BlackLevel	Lower limit of the white point luminance Value range: [0, u16WhiteLevel)
u16CbMax	B/G upper limit of the white point. The data format is 4.8-bit fixed-point. Value range: [0x0, 0xFFF]
u16CbMin	B/G lower limit of the white point. The data format is 4.8-bit fixed-point. The spots in [u16CbMin, u16CbMax] are involved in white balance statistics. Value range: [0x0, u16CbMax)
u16CrMax	R/G upper limit of the white point. The data format is 4.8-bit fixed-point.



Member	Description
	Value range: [0x0, 0xFFFF]
u16CrMin	R/G lower limit of the white point. The data format is 4.8-bit fixed-point. The spots in [u16CrMin, u16CrMax] are involved in white balance statistics. Value range: [0x0, u16CrMax)
u16GRgain	Weighted global white balance statistical information, G/R Value range: [0x0, 0xFFFF]
u16GBgain	Weighted global white balance statistical information, G/B Value range: [0x0, 0xFFFF]
u32GSum	Number of white points involved in white balance statistics Value range: [0x0, 0xFFFF]
u32Rgain	Real-time gain value of channel R, which can only be obtained and cannot be set
u32Ggain	Real-time gain value of channel G, which can only be obtained and cannot be set
u32Bgain	Real-time gain value of channel B, which can only be obtained and cannot be set
stZoneSta[WEIGHT_ZONE_ROW] [WEIGHT_ZONE_COLUMN]	White balance statistical information about zones

[Note]

None

[See Also]

[ISP\\_WB\\_ZONE\\_STA\\_INFO\\_S](#)

## ISP\_AWB\_ALG\_TYPE\_E

[Description]

Defines the AWB algorithm type.

[Syntax]

```
typedef enum hiISP_AWB_ALG_TYPE_E
{
    AWB_ALG_DEFAULT = 0,
    AWB_ALG_ADVANCE = 1,
    AWB_ALG_BUTT
} ISP_AWB_ALG_TYPE_E;
```



[Member]

Member	Description
AWB_ALG_DEFAULT	AWB algorithm of the SDK SPC070.
AWB_ALG_ADVANCE	Optimized AWB algorithm.

[Note]

The AWB stability, hybrid illuminant, and precision of the high and low color temperatures are improved for the AWB\_ALG\_ADVANCE algorithm.

[See Also]

None

## ISP\_ADV\_AWB\_ATTR\_S

[Description]

Defines the advanced AWB attributes.

[Syntax]

```
typedef struct hiISP_ADV_AWB_ATTR_S
{
    HI_BOOL bAccuPrior;

    HI_U8 u8Tolerance;

    HI_U16 u16CurveLLimit;

    HI_U16 u16CurveRLimit;
} ISP_ADV_AWB_ATTR_S;
```

[Member]

Member	Description
bAccuPrior	Enabling this member improves the AWB precision in the common indoor scenario.  You are advised to disable this member in the hybrid illuminant, pure color in a large region, and outdoor scenarios.
u8Tolerance	AWB tolerance.  The color temperature of the natural illuminant outdoors gradually changes; therefore, the value 4 or less is recommended.  If an indoor artificial illuminant is used, the tolerance can be increased to avoid interference of the pure-color moving object.



Member	Description
u16CurveLLimit	Limitation on the width of the left part of the color temperature curve. The value range is [0x0, 0xFF]. A smaller value indicates a wider illuminant range but lower AWB precision.
u16CurveRLimit	Limitation on the width of the right part of the color temperature curve. The value range is [0x0, 0xFF]. A larger value indicates a wider illuminant range but lower AWB precision.

[Note]

None

[See Also]

None

## ISP\_AWB\_LIGHTSOURCE\_INFO\_S

[Description]

Defines the attributes of an illuminant point.

[Syntax]

```
typedef struct hiISP_AWB_LIGHTSOURCE_INFO_S
{
    HI_U16 u16WhiteRgain;
    HI_U16 u16WhiteBgain;
    HI_U16 u16ExpQuant;
    HI_BOOL bLightStatus;
} ISP_AWB_LIGHTSOURCE_INFO_S;
```

[Member]

Member	Description
u16WhiteRgain	Rgain that is obtained by calibrating the ColorChecker Raw data captured in an illuminant using the <b>Static WB</b> option of the calibration tool.
u16WhiteBgain	Bgain that is obtained by calibrating the ColorChecker Raw data captured in an illuminant using the <b>WB</b> option of the calibration tool.
u16ExpQuant	Illuminant luminance (not supported currently).



Member	Description
bLightStatus	Illuminant point status. 0: disabled 1: enabled

[Note]

None

[See Also]

None

## ISP\_AWB\_ADD\_LIGHTSOURCE\_S

[Description]

Defines the attributes of separate illuminant points.

[Syntax]

```
typedef struct hiISP_AWB_ADD_LIGHTSOURCE_S
{
    HI_BOOL  bLightEnable;
    ISP_AWB_LIGHTSOURCE_INFO_S stLightInfo[LIGHTSOURCE_NUM];
}ISP_AWB_ADD_LIGHTSOURCE_S;
```

[Member]

Member	Description
bLightEnable	Separate illuminant point correction enable. White points of some illuminants such as the cool white fluorescent (CWF) are beyond the pre-calibrated color temperature curve. Adding separate illuminant points improves the AWB effect under such an illuminant.
stLightInfo[LIGHTSOURCE_NUM]	Separate illuminant point information.

[Note]

#define LIGHTSOURCE\_NUM 4

[See Also]

[ISP\\_AWB\\_LIGHTSOURCE\\_INFO\\_S](#)

# 5 CCM

## 5.1 Overview

The responses to the spectrum (R, G, and B components) are different between the sensor and the human eyes. A color correction matrix (CCM) is used to correct the spectrum response cross effect and spectral responsivity, ensuring that the colors of captured images are the same as visual colors.

## 5.2 Important Concepts

The following describes the concepts related to the CCM:

- Color reproduction: A CCM is used to correct the spectrum response cross effect and spectral responsivity, ensuring that the colors of images processed by the ISP are the same as visual colors
- Saturation: It is also called color purity. The saturation depends on the ratio of the colorization component to the decolorization component (gray). A larger colorization component ratio indicates the higher saturation, and a larger decolorization component ratio indicates the lower saturation.

## 5.3 Function Description

The offline calibration tool supports precorrection by using a 3 x 3 CCM. When the ISP is working, the firmware adjusts the saturation based on the current illumination to dynamically adjust the CCM coefficients. [Figure 5-1](#) shows a CCM.

**Figure 5-1** CCM

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} m_{RR} & m_{RG} & m_{RB} \\ m_{GR} & m_{GG} & m_{GB} \\ m_{BR} & m_{BG} & m_{BB} \end{pmatrix} \bullet \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$



## 5.4 API Reference

The following are CCM MPIs:

- [HI\\_MPI\\_ISP\\_SetSaturationAttr](#): Sets the color saturation attribute.
- [HI\\_MPI\\_ISP\\_GetSaturationAttr](#): Obtains the color saturation attribute.
- [HI\\_MPI\\_ISP\\_SetSaturation](#): Sets the expected color saturation.
- [HI\\_MPI\\_ISP\\_GetSaturation](#): Obtains the expected color saturation.
- [HI\\_MPI\\_ISP\\_SetCCM](#): Sets the CCM.
- [HI\\_MPI\\_ISP\\_GetCCM](#): Obtains the CCM.

### HI\_MPI\_ISP\_SetSaturationAttr

[Description]

Sets the color saturation attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetSaturationAttr(const ISP\_SATURATION\_ATTR\_S
*pstSatAttr);
```

[Parameter]

Parameter	Description	Input/Output
ISP_SATURATION_ATTR_S	Color saturation attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]



None

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

None



[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetSaturationAttr](#)

## HI\_MPI\_ISP\_GetSaturationAttr

[Description]

Obtains the color saturation attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetSaturationAttr(ISP\_SATURATION\_ATTR\_S *pstSatAttr);
```

[Parameter]

Parameter	Description	Input/Output
ISP_SATURATION_ATTR_S	Color saturation attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetSaturationAttr](#)





## HI\_MPI\_ISP\_SetSaturation

### [Description]

Sets the expected color saturation.

### [Syntax]

```
HI_S32 HI_MPI_ISP_SetSaturation(HI_U8 u8Value);
```

### [Parameter]

Parameter	Description	Input/Output
u8Value	Expected color saturation	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

### [Error Code]

None

### [Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

### [Note]

None

### [Example]

None

### [See Also]

None

## HI\_MPI\_ISP\_GetSaturation

### [Description]

Obtains the expected color saturation.

### [Syntax]

```
HI_S32 HI_MPI_ISP_GetSaturation(HI_U32 *pu32Value);
```

### [Parameter]



Parameter	Description	Input/Output
pu32Value	Expected color saturation	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_SetCCM

[Description]

Sets the CCM.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetCCM(const ISP\_COLORMATRIX\_S*pstColorMatrix);
```

[Parameter]

Parameter	Description	Input/Output
pstColorMatrix	Color matrix	Input

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- The data format of the CCM must be the same as that of the correction tool.
- The CCM can be configured based on the current color temperature, implementing better color reproduction at both high and low color temperatures.
- This MPI supports high, medium, and low CCMs. You need to correct a group of CCMs at high, medium, and low color temperatures respectively.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetCCM](#)

## HI\_MPI\_ISP\_GetCCM

[Description]

Obtains the CCM.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetCCM(ISP\_COLORMATRIX\_S*pstColorMatrix);
```

[Parameter]

Parameter	Description	Input/Output
pstColorMatrix	Color matrix	Output

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetCCM](#)

## 5.5 Data Structures

- [ISP\\_SATURATION\\_ATTR\\_S](#): Defines the ISP color saturation attribute.
- [ISP\\_COLORMATRIX\\_S](#): Defines the ISP color matrix attribute.

### ISP\_SATURATION\_ATTR\_S

[Description]

Defines the ISP color saturation attribute.

[Syntax]

```
typedef struct hiISP_SATURATION_ATTR_S
typedef struct hiISP_SATURATION_ATTR_S
{
    HI_BOOL bSatManualEnable;
    HI_U8   u8SatTarget;
    HI_U8   au8Sat[8];
}ISP_SATURATION_ATTR_S;
```

[Member]



Member	Description
bSatManualEnable	Manual color saturation enable. HI_FALSE: disabled HI_TRUE: enabled The default value is <b>HI_FALSE</b> .
u8SatTarget	Expected saturation. Value range: [0x00, 0xFF]. The default value is <b>0x80</b> .
au8Sat[8]	Configured picture saturation. The eight values in this array correspond to eight saturation values based on the sensor gain. A larger gain corresponds to a smaller saturation value. For details about the mapping between the values of au8Sat[8] and gains, see <a href="#">Table 5-1</a> .

**Table 5-1** Mapping between the values of au8Sat[8] and gains

au8Sat	Again*Dgian*IspDgain (times)
au8Sat [0]	1
au8Sat [1]	2
au8Sat [2]	4
au8Sat [3]	8
au8Sat [4]	16
au8Sat [5]	32
au8Sat [6]	64
au8Sat [7]	128

[Note]

Saturation can be automatically or manually adjusted.

- If bSatManualEnable is HI\_FALSE, saturation is automatically adjusted based on the system gain. For details about the mapping between saturation values and gains, see [Table 5-1](#). u8SatTarget is the expected saturation, and it is calculated as follows:  
Actual saturation = Saturation automatically adjusted based on the gain x u8SatTarget/0x80
- If bSatManualEnable is HI\_TRUE, saturation is manually adjusted.  
In this case, the actual saturation is the expected saturation (u8SatTarget), and the variable u8Sat[8] is invalid.

[See Also]

None



## ISP\_COLORMATRIX\_S

### [Description]

Defines the ISP color matrix attribute.

### [Syntax]

```
typedef struct hiISP_COLORMATRIX_S
{
    HI_U16 u16HighColorTemp;
    HI_U16 au16HighCCM[9];
    HI_U16 u16MidColorTemp;
    HI_U16 au16MidCCM[9];
    HI_U16 u16LowColorTemp;
    HI_U16 au16LowCCM[9];
} ISP_COLORMATRIX_S;
```

### [Member]

Member	Description
u16HighColorTemp	High color temperature. Value range: [2000, 10000]
au16HighCCM[9]	CCM at a high color temperature. Value range: [0x0, 0xFFFF]
u16MidColorTemp	Medium color temperature. Value range: [2000, u16HighColorTemp – 400]
au16MidCCM[9]	CCM at a medium color temperature. Value range: [0x0, 0xFFFF]
u16LowColorTemp	Low color temperature. Value range: [2000, u16MidColorTemp – 400]
au16LowCCM[9]	CCM at a low color temperature. Value range: [0x0, 0xFFFF]

### [Note]

- The data format of the CCM must be the same as that of the correction tool.
- u16HighColorTemp, u16MidColorTemp, and u16LowColorTemp must meet the following conditions:
  - $u16HighColorTemp - u16MidColorTemp \geq 400$
  - $u16MidColorTemp - u16LowColorTemp \geq 400$

### [See Also]

None





# 6 IMP

## 6.1 Sharpen

### 6.1.1 Function Description

The sharpen module is used to adjust the sharpen attribute of image edges. The sharpen strength is used to increase the horizontal and vertical edge strength of images.

### 6.1.2 API Reference

The following are sharpen MPIs:

- [HI\\_MPI\\_ISP\\_SetSharpenAttr](#): Sets the edge sharpen attribute.
- [HI\\_MPI\\_ISP\\_GetSharpenAttr](#): Obtains the edge sharpen attribute.

#### HI\_MPI\_ISP\_SetSharpenAttr

[Description]

Sets the edge sharpen attribute.

[Syntax]

```
HI_MPI_ISP_SetSharpenAttr(const ISP\_SHARPEN\_ATTR\_S*pstSharpenAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstSharpenAttr	Attribute for edge sharpen	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.





[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetSharpenAttr](#)

## HI\_MPI\_ISP\_GetSharpenAttr

[Description]

Obtains the edge sharpen attribute.

[Syntax]

```
HI_MPI_ISP_GetSharpenAttr(ISP\_SHARPEN\_ATTR\_S*pstSharpenAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstSharpenAttr	Attribute for edge sharpen	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.



Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetSharpenAttr](#)

## 6.1.3 Data Structure

### ISP\_SHARPEN\_ATTR\_S

[Description]

Defines the attribute for ISP sharpen.

[Syntax]

```
typedef struct hiISP_SHARPEN_ATTR_S
{
    HI_BOOL bEnable;
    HI_BOOL bManualEnable;
    HI_U8 u8StrengthTarget;
    HI_U8 u8StrengthMin;
    HI_U8 u8SharpenAltD[8];
    HI_U8 u8SharpenAltUd[8];
} ISP_SHARPEN_ATTR_S;
```

[Member]

Member	Description
bEnable	Sharpen enable HI_FALSE: disabled HI_TRUE: enabled The default value is HI_TRUE.
bManualEnable	Manual sharpen enable HI_FALSE: disabled



Member	Description
	HI_TRUE: enabled The default value is HI_FALSE.
u32StrengthTarget	Target sharpen strength when manual sharpen is enabled Value range: [0x00, 0xFF] The default value is 0x80.
u8StrengthMin	Minimum sharpen strength Value range: [0, 0xFF] The default value is 0x28.
u8SharpenAltD[8]	Sharpness of the large edge of the picture For details about the mapping between the eight values in this array and the values of the sensor based on the gain, see <a href="#">Table 6-1</a> .
u8SharpenAltUd[8]	Sharpness of the small texture of the picture For details about the mapping between the eight values in this array and the values of the sensor based on the gain, see <a href="#">Table 6-2</a> .

**Table 6-1** Values of u8SharpenAltD[8] based on the sensor gain

u8SharpenAltD	Again*Dgian*IspDgain(times)
u8SharpenAltD [0]	1
u8SharpenAltD [1]	2
u8SharpenAltD [2]	4
u8SharpenAltD [3]	8
u8SharpenAltD [4]	16
u8SharpenAltD [5]	32
u8SharpenAltD [6]	64
u8SharpenAltD [7]	128

**Table 6-2** Values of u8SharpenAltUd based on the sensor gain

u8SharpenAltUd	Again*Dgian*IspDgain(times)
u8SharpenAltUd [0]	1
u8SharpenAltUd [1]	2
u8SharpenAltUd [2]	4
u8SharpenAltUd [3]	8



u8SharpenAltUd	Again*Dgian*IspDgain(times)
u8SharpenAltUd [4]	16
u8SharpenAltUd [5]	32
u8SharpenAltUd [6]	64
u8SharpenAltUd [7]	128

[Note]

- When the manual sharpen function is enabled, a larger **u8Strength** value indicates higher sharpen strength.
- Automatic sharpen and manual sharpen are supported.
  - When **bEnable** is **HI\_TRUE** and **bManualEnable** is **HI\_FALSE**, automatic sharpen is used.  
For details about the relationship between the sharpen strength and the system gain, see descriptions of **u8SharpenAltD[8]** and **u8SharpenAltUd[8]**.
  - When **bEnable** and **bManualEnable** are **HI\_TRUE**, manual sharpen is used.
- The sharpen strength is automatically adjusted between the minimum strength and the expected strength based on the sensor gain.
- When the function of manually adjusting the sharpen strength is enabled, the actual sharpen strength is the expected value defined by **u32StrengthTarget**.

[See Also]

None

## 6.2 Gamma

### 6.2.1 Function Description

The gamma module performs non-linear conversion on the luminance space to adapt to the output device. The gamma module corrects R, G, and B components by using the gamma tables from the same group. The image pixels between gamma tables are generated by using linear interpolations.

### 6.2.2 API Reference

The following are gamma MPIs:

- [HI\\_MPI\\_ISP\\_SetGammaAttr](#): Sets the gamma attribute.
- [HI\\_MPI\\_ISP\\_SetGammaTable](#): Sets the gamma table attribute.
- [HI\\_MPI\\_ISP\\_GetGammaTable](#): Obtains the gamma table attribute.

#### HI\_MPI\_ISP\_SetGammaAttr

[Description]



Sets the gamma attribute.

[Syntax]

```
HI_MPI_ISP_SetGammaAttr(const ISP_GAMMA_ATTR_S* pstGammaAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstGammaAttr	Gamma attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetGammaAttr](#)

## HI\_MPI\_ISP\_GetGammaAttr

[Description]

Obtains the gamma attribute.

[Syntax]

```
HI_MPI_ISP_GetGammaAttr(ISP_GAMMA_ATTR_S* pstGammaAttr);
```

[Parameter]



Parameter	Description	Input/Output
pstGammaAttr	Gamma attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetGammaAttr](#)

## HI\_MPI\_ISP\_SetGammaTable

[Description]

Sets the gamma table attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetGammaTable(const ISP\_GAMMA\_TABLE\_S\* pstGammaTable);
```

[Parameter]

Parameter	Description	Input/Output
pstGammaTable	Gamma table attribute	Input

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: `hi_comm_isp.h`, `mpi_isp.h`
- Library file: `libisp.a`

[Note]

- Before setting a gamma table, you must set the gamma attribute.
- When you select the 1.6, 1.8, 2.0, or 2.2 gamma curve supported by the ISP, you do not need to set `u16Gamma`.
- If you use a user-defined gamma curve, you must set `u16Gamma`.

[Example]

None

[See Also]

- [HI\\_MPI\\_ISP\\_SetGammaAttr](#)
- [HI\\_MPI\\_ISP\\_GetGammaAttr](#)

## HI\_MPI\_ISP\_GetGammaTable

[Description]

Obtains the gamma table attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetGammaTable(ISP\_GAMMA\_TABLE\_S\* pstGammaTable);
```

[Parameter]

Parameter	Description	Input/Output
<code>pstGammaTable</code>	Gamma table attribute	Output

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
HI_SUCCESS	Success.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

Only the gamma table can be read, that is, ISP\_GAMMA\_CURVE\_E cannot be read.

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetGammaAttr](#)

## 6.2.3 Data Structures

The following are gamma data structures:

- [ISP\\_GAMMA\\_ATTR\\_S](#): Defines the attribute for ISP gamma correction.
- [ISP\\_GAMMA\\_CURVE\\_E](#): Defines the type of the ISP gamma curve.
- [ISP\\_GAMMA\\_TABLE\\_S](#): Defines the attribute of the ISP gamma table.

### ISP\_GAMMA\_ATTR\_S

[Description]

Defines the attribute for ISP gamma correction.

[Syntax]

```
typedef struct hiISP_GAMMA_ATTR_S
{
    HI_BOOL bEnable;

} ISP_GAMMA_ATTR_S;
```

[Member]





Member	Description
bEnable	Gamma correction enable HI_FALSE: disabled HI_TRUE: enabled The default value is HI_TRUE.

[Note]

The same gamma table is used when the components R, G, and B are corrected. A gamma table includes 65 elements.

[See Also]

None

## ISP\_GAMMA\_CURVE\_E

[Description]

Defines the type of the ISP gamma curve.

[Syntax]

```
typedef enum hiISP_GAMMA_CURVE_E
{
    ISP_GAMMA_CURVE_1_6 = 0x0,
    ISP_GAMMA_CURVE_1_8 = 0x1,
    ISP_GAMMA_CURVE_2_0 = 0x2,
    ISP_GAMMA_CURVE_2_2 = 0x3,
    ISP_GAMMA_CURVE_DEFAULT = 0x4,
    ISP_GAMMA_CURVE_SRGB = 0x5,
    ISP_GAMMA_CURVE_USER_DEFINE = 0x6,
    ISP_GAMMA_CURVE_BUTT
} ISP_GAMMA_CURVE_E;
```

[Member]

Member	Description
ISP_GAMMA_CURVE_1_6	1.6 gamma curve
ISP_GAMMA_CURVE_1_8	1.8 gamma curve
ISP_GAMMA_CURVE_2_0	2.0 gamma curve
ISP_GAMMA_CURVE_2_2	2.2 gamma curve
ISP_GAMMA_CURVE_SRGB	sRGB gamma curve
ISP_GAMMA_CURVE_DEFAULT	Default gamma curve
ISP_GAMMA_CURVE_USER_DEFINE	User-defined gamma curve



[Note]

When a user-defined gamma curve is used, ensure that the gamma table is properly configured.

[See Also]

None

## ISP\_GAMMA\_TABLE\_S

[Description]

Defines the attribute of the ISP gamma table.

[Syntax]

```
typedef struct hiISP_GAMMA_TABLE_S
{
    ISP_GAMMA_CURVE_E enGammaCurve;
    HI_U16 u16Gamma[GAMMA_LUT_SIZE];
    HI_U16 u16Gamma_FE[GAMMA_FE_LUT_SIZE];
} ISP_GAMMA_TABLE_S;
```

[Member]

Member	Description
enGammaCurve	Gamma curve selection The default value is ISP_GAMMA_CURVE_DEFAULT.
u16Gamma[GAMMA_LUT_SIZE]	Gamma table Value range: <ul style="list-style-type: none"><li>• [0, 0xFFFF] for the Hi3516</li><li>• [0, 0xFFF] for the Hi3518</li></ul>
u16Gamma_FE[GAMMA_FE_LUT_SIZE]	Gamma_fe table of the wide dynamic range (WDR) sensor Value range: <ul style="list-style-type: none"><li>• [0, 0xFFFF] for the Hi3516</li><li>• [0, 0xFFF] for the Hi3518</li></ul>

[Difference]

Chip	GAMMA_LUT_SIZE Value	GAMMA_FE_LUT_SIZE Value
Hi3516	65	129
Hi3518	257	257



[Note]

- When a user-defined gamma curve is used, ensure that the gamma table is properly configured.
- Call `HI_MPI_ISP_SetGammaTable` to configure the gamma table and ignore the `u16Gamma_FE` variable. Call `HI_MPI_ISP_SetGammaFETable` to configure the `Gamma_fe` table of the WDR sensor and ignore the `enGammaCurve` and `u16Gamma` variables.

[See Also]

[ISP\\_GAMMA\\_CURVE\\_E](#)

## 6.3 DRC

### 6.3.1 Function Description

The dynamic range compression (DRC) module adjusts the image dynamic range to display more image details. It is an advanced local tone mapping engine based on the features of the human visual system. The DRC module supports multi-space dynamic range compression.

### 6.3.2 API Reference

The following are DRC MPIs:

- [HI\\_MPI\\_ISP\\_SetDRCAttr](#): Sets DRC attributes.
- [HI\\_MPI\\_ISP\\_GetDRCAttr](#): Obtains DRC attributes.

#### HI\_MPI\_ISP\_SetDRCAttr

[Description]

Sets DRC attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDRCAttr(const ISP\_DRC\_ATTR\_S*pstDRCAttr);
```

[Parameter]

Parameter	Description	Input/Output
<code>pstDRCAttr</code>	DRC attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.



[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetDRCAAttr](#)

## HI\_MPI\_ISP\_GetDRCAAttr

[Description]

Obtains DRC attributes.

[Syntax]

```
HI_MPI_ISP_GetDRCAAttr(ISP\_DRC\_ATTR\_S*pstDRCAAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstDRCAAttr	DRC attributes	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.



[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetDRCAttr](#)

## 6.3.3 Data Structure

### ISP\_DRC\_ATTR\_S

[Description]

Defines DRC attributes.

[Syntax]

```
typedef struct hiISP_DENOISE_ATTR_S
{
    HI_BOOL bDRCEnable;
    HI_BOOL bDRCManualEnable;
    HI_U32  u32StrengthTarget;
    HI_U32  u32SlopeMax;
    HI_U32  u32SlopeMin;
    HI_U32  u32WhiteLevel;
    HI_U32  u32BlackLevel;
    HI_U32  u32VarianceSpace;
    HI_U32  u32VarianceIntensity;
} ISP_DENOISE_ATTR_S;
```

[Member]

Member	Description
bDRCEnable	DRC enable HI_FALSE: disabled HI_TRUE: enabled The default value is HI_FALSE for common sensors or is HI_TRUE for WDR sensors.
bDRCManualEnable	Manual DRC enable HI_FALSE: disabled



Member	Description
	HI_TRUE: enabled The default value is HI_FALSE.
u32StrengthTarget	Target DRC strength Value range: [0, 0xFF] The default value is 0x80.
u32SlopeMax	DRC enhancement control parameter for controlling the maximum slope of the curve of the current pixel. Value range: [0, 0xFF] The default value is defined by the <b>iridix_sm</b> member of <b>cmos_isp_default_t</b> in the <b>cmos.c</b> file. Recommended value: [0x20, 0x60]
u32SlopeMin	DRC enhancement control parameter for controlling the minimum slope of the curve of the current pixel. Value range: [0, 0xFF] The default value is 0x40 for common sensors or is 0x10 for WDR sensors. Recommended value range: [0x00, 0x30]
u32WhiteLevel	Maximum pixel value for DRC enhancement, that is, pixel value of the bayer data domain. The pixels greater than <b>u32Whitelevel</b> are not involved in the DRC calculation. The default value is defined by the <b>iridix_wl</b> member of <b>cmos_isp_default_t</b> in the <b>cmos.c</b> file. Value range: [0, 0xFFFF]
u32BlackLevel	Minimum pixel value for DRC enhancement, that is, pixel value of the bayer data domain. The pixels less than <b>u32Blacklevel</b> are not involved in the DRC calculation. Value range: [0, 0xFFFF] The default value is 0.
u32VarianceSpace	A larger value indicates that the pixel window when tone curves are generated is larger. Value range: [0x0, 0xF] The default value is 2.
u32VarianceIntensity	A larger value indicates that the DRC strength variance between each pixel and surrounding pixels is slighter. Value range: [0x0, 0xF] The default value is 1.

[Note]

- After the DRC is started, the larger the **u32StrengthTarget** value, the lighter the dark area, and the larger the noise is.



- When the manual DRC function is enabled, the actual DRC strength is the expected value **u32StrengthTarget**.
- Other DRC parameters are advanced parameters. You are advised not to modify them.

[See Also]

None

## 6.4 Lens Shading Correction

### 6.4.1 Overview

The lens structure determines that a sensor's center absorbs more light than its surroundings and the surroundings look more like shading. This phenomenon is called vignetting. The lens shading correction function is used to compensate and correct the vignettes on images. For R, G, B components, the same parameter can be used during luminance correction, and respective correction parameters are used during color correction.

### 6.4.2 Function Description

#### Hi3516

The Hi3516 uses mesh correction to divide images. An image is divided into 64 x 64 zones by default. Each zone has an 8-bit correction coefficient (gain). The actual gain for each pixel is generated by using bilinear interpolations. The correction coefficient type is defined by the global parameter Mesh\_Scale. Correction coefficients are generated by the offline calibration tool. [Table 6-3](#) describes the Mesh\_Scale parameter.

**Table 6-3** Mesh\_Scale parameter

Mesh Scale	Correction Coefficient Format	Maximum Gain
0	Unsigned decimal, 1.7-bit fixed-point	X2
1	Unsigned decimal, 2.6-bit fixed-point	X4
2	Unsigned decimal, 3.5-bit fixed-point	X8
3	Unsigned decimal, 4.4-bit fixed-point	X16

#### Hi3518

The Hi3518 uses radial correction to set centers and correction coefficients for R, G, and B components. Correction coefficients define the gain from the center to the farthest corner in the form of a concentric ring. The correction coefficients are expressed by the lookup table with the size of 129, the data format is unsigned decimal, fixed-point 4.12. Theoretically, a maximum of X16 gain is supported. Correction coefficients are generated by the offline calibration tool.



## 6.4.3 API Reference

The following are the MPIs related to lens shading correction:

- [HI\\_MPI\\_ISP\\_SetShadingAttr](#): Sets the attribute for shading correction.
- [HI\\_MPI\\_ISP\\_GetShadingAttr](#): Obtains the attribute for shading correction.
- [HI\\_MPI\\_ISP\\_SetShadingTable](#): Sets the attribute of the shading correction lookup table.
- [HI\\_MPI\\_ISP\\_GetShadingTable](#): Obtains the attribute of the shading correction lookup table.

### HI\_MPI\_ISP\_SetShadingAttr

[Description]

Sets the attribute for shading correction.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetShadingAttr(const ISP\_SHADING\_ATTR\_S*pstShadingAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstShadingAttr	Attribute for shading correction	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The parameter is invalid.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

This MPI is used to enable lens shading correction.

[Example]

None





[See Also]

[HI\\_MPI\\_ISP\\_GetShadingAttr](#)

## HI\_MPI\_ISP\_GetShadingAttr

[Description]

Obtains the attribute for shading correction.

[Syntax]

```
HI_MPI_ISP_GetShadingAttr(ISP\_SHADING\_ATTR\_S*pstShadingAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstShadingAttr	Attribute for shading correction	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetShadingAttr](#)

## HI\_MPI\_ISP\_SetShadingTable

[Description]



Sets the attribute of the shading correction lookup table.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetShadingTable(const ISP_SHADING_ATTR_S *pstShadingTab);
```

[Parameter]

Parameter	Description	Input/Output
pstShadingTab	Attribute of the shading correction lookup table	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	The input parameter is invalid.

[Request]

- Header files: hi\_comm\_isp.h, mpi\_isp.h.
- Library file: libisp.a

[Note]

This MPI is used to set a shading correction lookup table. Only the first u16ShadingTableNodeNumber shading correction points in the table need to be set for the Hi3518.

[Example]

None

[See Also]

- [HI\\_MPI\\_ISP\\_GetShadingTable](#)
- [HI\\_MPI\\_ISP\\_GetShadingAttr](#)

## HI\_MPI\_ISP\_GetShadingTable

[Description]

Obtains the attribute of the shading correction lookup table.



[Syntax]

```
HI_S32 HI_MPI_ISP_GetShadingTable(ISP\_SHADING\_TAB\_E*pstShadingTab);
```

[Parameter]

Parameter	Description	Input/Output
pstShadingTab	Attribute of the shading correction lookup table	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. The value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Request]

- Header files: [hi\\_comm\\_isp.h](#), [mpi\\_isp.h](#).
- Library file: [libisp.a](#)

[Note]

None

[Example]

None

[See Also]

- [HI\\_MPI\\_ISP\\_SetShadingTable](#)
- [HI\\_MPI\\_ISP\\_SetShadingAttr](#)

## 6.4.4 Data Structures

### 6.4.4.1 Hi3516 Data Structures

The following are Hi3516 data structures related to lens shading correction:

- [ISP\\_SHADING\\_SCALE\\_E](#): Defines the format of the ISP lens shading correction value.
- [ISP\\_SHADING\\_TAB\\_E](#): Defines the type of the ISP lens shading table.
- [ISP\\_SHADING\\_ATTR\\_S](#): Defines the attribute for ISP lens shading correction.
- [ISP\\_SHADINGTAB\\_S](#): Defines the attribute of the ISP lens shading table.



## ISP\_SHADING\_SCALE\_E

### [Description]

Defines the format of the ISP lens shading correction value.

### [Syntax]

```
typedef enum hiISP_SHADING_SCALE_E
{
    ISP_SHADING_SCALE_2 = 0x0,
    ISP_SHADING_SCALE_4 = 0x1,
    ISP_SHADING_SCALE_8 = 0x2,
    ISP_SHADING_SCALE_16 = 0x3,
    ISP_SHADING_SCALE_BUTT
} ISP_SHADING_SCALE_E;
```

### [Member]

Member	Description
ISP_SHADING_SCALE_2	The format of the lens shading correction value is 1.7-bit fix-point.
ISP_SHADING_SCALE_4	The format of the lens shading correction value is 2.6-bit fix-point.
ISP_SHADING_SCALE_8	The format of the lens shading correction value is 3.5-bit fix-point.
ISP_SHADING_SCALE_16	The format of the lens shading correction value is 4.4-bit fix-point.

### [Note]

None

### [See Also]

None

## ISP\_SHADING\_TAB\_E

### [Description]

Defines the type of the ISP lens shading table.

### [Syntax]

```
typedef enum hiISP_SHADING_TAB_E
{
    SHADING_TAB_R = 0,
    SHADING_TAB_G = 1,
    SHADING_TAB_B = 2,
}
```



```
SHADING_TAB_BUTT  
} ISP_SHADING_TAB_E;
```

[Member]

Member	Description
SHADING_TAB_R	Lens shading table R
SHADING_TAB_G	Lens shading table G
SHADING_TAB_B	Lens shading table B

[Note]

None

[See Also]

None

## ISP\_SHADING\_ATTR\_S

[Description]

Defines the attribute for ISP lens shading correction.

[Syntax]

```
typedef struct hiISP_SHADING_ATTR_S  
{  
    HI_BOOL Enable;  
} ISP_SHADING_ATTR_S;
```

[Member]

Member	Description
bEnable	Lens shading correction enable HI_FALSE: disabled HI_TRUE: enabled The default value is HI_FALSE.

[Note]

Corresponding shading tables are used for lens shading correction.

[See Also]

None



## ISP\_SHADINGTAB\_S

### [Description]

Defines the attribute of the ISP lens shading table.

### [Syntax]

```
typedef struct hiISP_SHADINGTAB_S
{
    ISP_SHADING_SCALE_E enScale;
    ISP_SHADING_TAB_E enMesh_R;
    ISP_SHADING_TAB_E enMesh_G;
    ISP_SHADING_TAB_E enMesh_B;
    HI_U8 u8ShadingTable_R[64*64];
    HI_U8 u8ShadingTable_G[64*64];
    HI_U8 u8ShadingTable_B[64*64];
} ISP_SHADINGTAB_S;
```

### [Member]

Member	Description
enScale	Format of the lens shading correction value
enMesh_R	Lens shading table selected for the R component
enMesh_G	Lens shading table selected for the G component
enMesh_B	Lens shading table selected for the B component
u8ShadingTable_R[64*64]	Lens shading table R
u8ShadingTable_G[64*64]	Lens shading table G
u8ShadingTable_B[64*64]	Lens shading table B

### [Note]

- enScale determines the value formats of the lens shading tables R, G, and B.
- Typically, you only need to configure one lens shading table (for example, u8ShadingTable\_R[64\*64]), and set enMesh\_R, enMesh\_G, and enMesh\_B to the same value (for example, SHADING\_TAB\_R). In this way, the lens shading table R is selected for components R, G, B, and lens shading correction is implemented.
- In some scenarios (for example, a poor lens is used), color shading correction is required. You need to configure three lens shading tables, and set enMesh\_R, enMesh\_G, and enMesh\_B to select corresponding tables. Color shading correction reduces color inconsistency.

### [See Also]

- [ISP\\_SHADING\\_SCALE\\_E](#)
- [ISP\\_SHADING\\_TAB\\_E](#)



## 6.4.4.2 Hi3518 Data Structures

The following are Hi3518 data structures related to lens shading correction:

- [ISP\\_SHADING\\_ATTR\\_S](#): Defines the attribute for ISP lens shading correction.
- [ISP\\_SHADINGTAB\\_S](#): Defines the attribute of the ISP lens shading table.

### ISP\_SHADING\_ATTR\_S

[Description]

Defines the attribute for ISP lens shading correction.

[Syntax]

```
typedef struct hiISP_SHADING_ATTR_S
{
    HI_BOOL Enable;
} ISP_SHADING_ATTR_S;
```

[Member]

Member	Description
bEnable	Lens shading correction enable HI_FALSE: disabled HI_TRUE: enabled The default value is HI_FALSE.

[Note]

Respective shading tables can be used for different lens shading correction operations.

[See Also]

None

### ISP\_SHADINGTAB\_S

[Description]

Defines the attribute of the ISP lens shading table.

[Syntax]

```
typedef struct hiISP_SHADINGTAB_S
{
    HI_U16 u16ShadingCenterR_X;
    HI_U16 u16ShadingCenterR_Y;
    HI_U16 u16ShadingCenterG_X;
    HI_U16 u16ShadingCenterG_Y;
    HI_U16 u16ShadingCenterB_X;
    HI_U16 u16ShadingCenterB_Y;
```



```

HI_U16 u16ShadingTable_R[129];
HI_U16 u16ShadingTable_G[129];
HI_U16 u16ShadingTable_B[129];

HI_U16 u16ShadingOffCenter_R;
HI_U16 u16ShadingOffCenter_G;
HI_U16 u16ShadingOffCenter_B;

HI_U16 u16ShadingTableNodeNumber;

} ISP_SHADINGTAB_S;

```

[Member]

Member	Description
u16ShadingCenterR_X	Horizontal coordinate of the R component center Value range: [0x0, 0xFFFF]
u16ShadingCenterR_Y	Vertical coordinate of the R component center Value range: [0x0, 0xFFFF]
u16ShadingCenterG_X	Horizontal coordinate of the G component center Value range: [0x0, 0xFFFF]
u16ShadingCenterG_Y	Vertical coordinate of the G component center Value range: [0x0, 0xFFFF]
u16ShadingCenterB_X	Horizontal coordinate of the B component center Value range: [0x0, 0xFFFF]
u16ShadingCenterB_Y	Vertical coordinate of the B component center Value range: [0x0, 0xFFFF]
u16ShadingTable_R	Correction table of the R component Value range: [0x0, 0xFFFF]
u16ShadingTable_G	Correction table of the G component Value range: [0x0, 0xFFFF]
u16ShadingTable_B	Correction table of the B component Value range: [0x0, 0xFFFF]
u16ShadingOffCenter_R	The maximum radial distance from the centre point of the R component to the frame edge. A larger distance indicates a smaller value. Value range: [0x0, 0xFFFF]
u16ShadingOffCenter_G	The maximum radial distance from the centre point of the G component to the frame edge. A larger distance indicates a smaller value.





Member	Description
	Value range: [0x0, 0xFFFF]
u16ShadingOffCenter_B	The maximum radial distance from the centre point of the B component to the frame edge. A larger distance indicates a smaller value. Value range: [0x0, 0xFFFF]
u16ShadingTableNodeNumber	Number of shading correction points in each component correction table. Value range: The default value is <b>0x81</b> .

[Note]

The upper left corner of the image is the origin. The coordinate unit is pixel.

[See Also]

None

## 6.5 Defect Pixel

### 6.5.1 Overview

The defect pixel correction module detects defect pixels by using the internal defect pixel correction logic. The coordinate information about the detected defect pixels is stored in the flash memory. If the ISP restarts, the defect pixel coordinates are loaded and such defect pixels do not need to be detected again.

### 6.5.2 Function Description

The defect pixel correction module searches for the defect pixels that are quite different from their adjacent pixels by using a 5 x 5 window.

The defect pixel correction module supports the following modes:

- Static defect pixel detection and correction  
The iris is disabled. The defect pixel detection program is started to obtain defect pixels coordinates. The total number of pixel pixels to be corrected depends on the memory of the defect pixel detection module. The detected defect pixels are corrected by using median filtering for adjacent pixels.
- Dynamic defect pixel detection and correction  
The defect pixel correction module dynamically detects and corrects defect pixels. There is limitation on the number of defect pixels to be corrected. The defect pixel correction module uses a median filter. Though the dynamic defect pixel detection and correction mode is less reliable than the static defect pixel detection and correction mode, the dynamic mode is strongly recommended in the low-illumination scenario.

### 6.5.3 API Reference

The following are MPIs related to defect pixel correction:



- [HI\\_MPI\\_ISP\\_SetDefectPixelAttr](#): Sets the attribute for defect pixel correction.
- [HI\\_MPI\\_ISP\\_GetDefectPixelAttr](#): Obtains the attribute for defect pixel correction.

## HI\_MPI\_ISP\_SetDefectPixelAttr

### [Description]

Sets the attribute for defect pixel correction.

### [Syntax]

```
HI_S32 HI_MPI_ISP_SetDefectPixelAttr(const ISP_DP_ATTR_S *pstDPAttr);
```

### [Parameter]

Parameter	Description	Input/Output
pstDPAttr	Attribute for defect pixel correction	Input

### [Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

### [Error Code]

Error Code	Description
HI_SUCCESS	Success.

### [Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

### [Note]

This MPI provides the functions of correcting defect pixels and detecting defect pixels. After defect pixel correction is enabled, the ISP automatically processes the pixels of each frame. You need to enable defect pixel detection once only. After obtaining the coordinates of defect pixels, you can disable the defect pixel detection. Before defect pixel detection, you must cover the lens or disable the shutter function, set the analog gain and digital gain to minimum values, and decrease the frame rate to 5–6 frame/s. This ensures that the exposure duration is 200 ms.

### [Example]

```
ISP_DP_ATTR_S stDPAttr;  
HI_U16 i;
```



```
HI_U32 u32Table[1024] = {0};

HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
stDPAttr.bEnableStatic = HI_TRUE;
stDPAttr.bEnableDynamic = HI_TRUE;
stDPAttr.bEnableDetect = HI_TRUE;
stDPAttr.u16BadPixelCountMax = 0x200;
stDPAttr.u16BadPixelCountMin = 0x40;

for(i=0; i< 1024;i++)
{
    stDPAttr.u32BadPixelTable[i] = 0;
}
HI_MPI_ISP_SetDefectPixelAttr(&stDPAttr);
PAUSE;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
PAUSE;
```

[See Also]

[HI\\_MPI\\_ISP\\_GetDefectPixelAttr](#)

## HI\_MPI\_ISP\_GetDefectPixelAttr

[Description]

Obtains the attribute for defect pixel correction.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDefectPixelAttr(ISP\_DP\_ATTR\_S*pstDPAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstDPAttr	Attribute for defect pixel correction	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Requirement]



- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

```
{
ISP_DP_ATTR_S  stDPAttr;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
stDPAttr.bEnableStatic = HI_TRUE;
stDPAttr.bEnableDynamic= HI_FALSE;
stDPAttr.bEnableDetect = HI_TRUE;
stDPAttr.u16BadPixelCountMax = 0x200;
stDPAttr.u16BadPixelCountMin = 0x40;
for(i=0; i< 1024;i++)
{
stDPAttr.u32BadPixelTable[i] = 0;
}

HI_MPI_ISP_SetDefectPixelAttr(&stDPAttr);
PAUSE;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
PAUSE;

}
```

[See Also]

[HI\\_MPI\\_ISP\\_SetDefectPixelAttr](#)

## 6.5.4 Data Structures

The following are the data structures related to defect pixel correction:

- [ISP\\_DP\\_ATTR\\_S](#): Defines the attribute for ISP defect pixel correction.
- [ISP\\_TRIGGER\\_STATUS\\_E](#): Defines the ISP correction or detection status.

### ISP\_DP\_ATTR\_S

[Description]

Defines the attribute for ISP defect pixel correction.

[Syntax]

```
typedef struct hiISP_DP_ATTR_S
{
    HI_BOOL bEnableDynamic;
    HI_U16 u16DynamicBadPixelSlope;
```



```

HI_U16 u16DynamicBadPixelThresh;
HI_BOOL bEnableStatic;
HI_BOOL bEnableDetect;
ISP_TRIGGER_STATUS_E enTriggerStatus;
HI_U8 u8BadPixelStartThresh;
HI_U8 u8BadPixelFinishThresh;
HI_U16 u16BadPixelCountMax;
HI_U16 u16BadPixelCountMin;
HI_U16 u16BadPixelCount;
HI_U16 u16BadPixelTriggerTime;
HI_U32 u32BadPixelTable[1024];
} ISP_DP_ATTR_S;

```

[Member]

Member	Description
bEnableDynamic	Dynamic defect pixel correction enable
u16DynamicBadPixelSlope	Dynamic defect pixel correction strength Value range: [0, 0xFFFF]
u16DynamicBadPixelThresh	Dynamic defect pixel detection threshold Value range: [0, 0xFFFF]
bEnableStatic	Static defect pixel correction enable
bEnableDetect	Static defect pixel detection enable
enTriggerStatus	Result and status of static defect pixel detection
u8BadPixelStartThresh	Threshold for starting static defect pixel detection
u8BadPixelFinishThresh	Threshold for stopping static defect pixel detection
u16BadPixelCountMax	Maximum number of allowed static defect pixels Value range: [0, 0x3FF]
u16BadPixelCountMin	Minimum number of allowed static defect pixels Value range: [0, 0x3FF]
u16BadPixelCount	Number of static defect pixels Value range: [0, 0x3FF]
u16BadPixelTriggerTime	Timeout of static defect pixel detection or correction, in frame Value range: [0, 0x640]
u32BadPixelTable[1024]	Coordinates of a defect pixel. First 22 bits are valid. The lower 11 bits indicate the horizontal coordinates; and bits 12–22 indicate the vertical coordinates.



[Note]

- The ISP defect pixel detection is successful if the number of detected defect pixels ranges from u16BadPixelCountMin to u16BadPixelCountMax. You need to change the values of u16BadPixelCountMin and u16BadPixelCountMax when different sensors are used.
- The value of u8BadPixelFinishThresh is an output value. If the sensors of the same type are used, you are advised to set u8BadPixelStartThresh based on the value of u8BadPixelFinishThresh. This increases the speed of static defect pixel correction.

[See Also]

[ISP\\_TRIGGER\\_STATUS\\_E](#)

## ISP\_TRIGGER\_STATUS\_E

[Description]

Defines the ISP correction or detection status.

[Syntax]

```
typedef enum hiISP_TRIGGER_STATUS_E
{
    ISP_TRIGGER_INIT      = 0,
    ISP_TRIGGER_SUCCESS   = 1,
    ISP_TRIGGER_TIMEOUT   = 2,
    ISP_TRIGGER_BUTT
} ISP_TRIGGER_STATUS_E;
```

[Member]

Member	Description
ISP_TRIGGER_INIT	The ISP is initialized and no correction is performed.
ISP_TRIGGER_SUCCESS	Correction is successful.
ISP_TRIGGER_TIMEOUT	Correction ends due to timeout.

[Note]

None

[See Also]

None

## 6.6 Crosstalk Removal

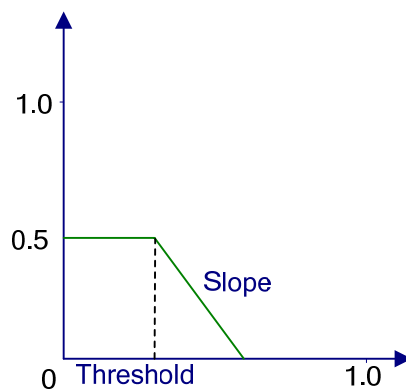
### 6.6.1 Overview

The crosstalk removal module balances the differences between the Gr and Gb values of adjacent pixels in raw data. This function avoids squares or similar patterns that occur when the demosaic interpolation algorithm is used. When light comes from special angles, crosstalk occurs in the sensor. As a result, patterns are generated due to the inconsistency of Gr and Gb values of adjacent pixels.

### 6.6.2 Function Description

As shown in [Figure 6-1](#), the horizontal coordinate indicates the difference between Gr and Gb values ( $|Gr - Gb|$ ), and the vertical coordinate indicates the processing strength. When the difference between Gr and Gb values is below the threshold, the maximum strength value 0.5 is used. When the difference is above the threshold, the strength value decreases gradually. A larger threshold indicates a larger processing strength value. A larger slope value indicates severer fluctuations between the minimum and maximum processing strength values.

**Figure 6-1** Crosstalk removal threshold



### 6.6.3 API Reference

The following are the MPIs related to crosstalk removal:

- [HI\\_MPI\\_ISP\\_SetCrosstalkAttr](#): Sets the crosstalk removal attribute.
- [HI\\_MPI\\_ISP\\_GetCrosstalkAttr](#): Obtains the crosstalk removal attribute.

#### HI\_MPI\_ISP\_SetCrosstalkAttr

[Description]

Sets the crosstalk removal attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetCrosstalkAttr(const ISP\_CR\_ATTR\_S *pstCRAAttr)
```

[Parameter]



Parameter	Description	Input/Output
pstCRAttr	Crosstalk attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetCrosstalkAttr](#)

## HI\_MPI\_ISP\_GetCrosstalkAttr

[Description]

Obtains the crosstalk removal attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetCrosstalkAttr(const ISP\_CR\_ATTR\_S *pstCRAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstCRAttr	Crosstalk attribute	Output

[Return Value]





Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetCrosstalkAttr](#)

## 6.6.4 Data Structure

### ISP\_CR\_ATTR\_S

[Description]

Defines the ISP crosstalk attribute.

[Syntax]

```
typedef struct hiISP_CR_ATTR_S
{
    HI_BOOL  bEnable;
    HI_U8    u8Sensitivity;
    HI_U16   u16Threshold;
    HI_U16   u16Slope;
    HI_U8    u8Strength[8];
}ISP_CR_ATTR_S;
```

[Member]



Member	Description
bEnable	Crosstalk removal enable
u8Sensitivity	Crosstalk removal sensitivity
u16Threshold	Crosstalk removal threshold Value range: [0, 0xFFFF]
u16Slope	Crosstalk removal slope Value range: [0, 0xFFFF]
u8Strength	Crosstalk removal strength. The eight values of this array correspond to different ISO values. For details, see <a href="#">Table 6-4</a> .

**Table 6-4** Mapping between the values of u8Strength and gain

u8Strength	Again*Dgian*IspDgain (times)
u8Strength [0]	1
u8Strength [1]	2
u8Strength [2]	4
u8Strength [3]	8
u8Strength [4]	16
u8Strength [5]	32
u8Strength [6]	64
u8Strength [7]	128

[Note]

- Typically, the **u8Strength** value decreases when the ISO value increases.
- A larger **u8Sensitivity** value indicates that the edge is less sensitive during R component correction.
- A larger **u16Threshold** value indicates a larger processing strength value.
- A larger **u16Slope** value indicates that the processing strength changes more significantly when the difference between Gr and Gb values changes.

[See Also]

None



## 6.7 Denoise

### 6.7.1 Overview

The denoise algorithm is used in the spatial domain for processing the raw data. The edges and textures are retained during denoise.

### 6.7.2 Function Description

The denoise algorithm supports the following modes:

- Automatic mode  
The denoise strength is in non-linear proportion to the system gain. The denoise strength automatically changes according to environment. When the ambient environment is dark, the system analog gain and digital gain increase at the same time. When the ambient environment is bright, the system analog gain and digital gain are small. For details about the mapping between the denoise strength and the system gain, see descriptions of member variables in `ISP_DENOISE_ATTR_S`.
- Manual mode  
The actual denoise strength is the same as the target value.

### 6.7.3 API Reference

The following are denoise MPIs:

- [HI\\_MPI\\_ISP\\_SetDenoiseAttr](#): Sets the denoise attribute.
- [HI\\_MPI\\_ISP\\_GetDenoiseAttr](#): Obtains the denoise attribute.

#### HI\_MPI\_ISP\_SetDenoiseAttr

[Description]

Sets the denoise attribute.

[Syntax]

```
HI_MPI_ISP_SetDenoiseAttr(const *pstDenoiseAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstDenoiseAttr	Denoise attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.



[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetDenoiseAttr](#)

## HI\_MPI\_ISP\_GetDenoiseAttr

[Description]

Obtains the denoise attribute.

[Syntax]

```
HI_MPI_ISP_GetDenoiseAttr(const ISP\_DENOISE\_ATTR\_S*pstDenoiseAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstDenoiseAttr	Denoise attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.



[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetDenoiseAttr](#)

## 6.7.4 Data Structure

### ISP\_DENOISE\_ATTR\_S

[Description]

Defines the ISP denoise attribute.

[Syntax]

```
typedef struct hiISP_DENOISE_ATTR_S
{
    HI_BOOL bEnable;
    HI_BOOL bManualEnable;
    HI_U8 u8ThreshTarget;
    HI_U8 u8ThreshMax;
    HI_U8 u8SnrThresh[8]
} ISP_DENOISE_ATTR_S;
```

[Member]

Member	Description
bEnable	Denoise enable
bManualEnable	Manual denoise enable
u8ThreshTarget	Target denoise threshold when manual denoise is enabled
u8ThreshMax	Maximum denoise threshold
u8SnrThresh[8]	Image denoise strength. The eight values in this array correspond to eight image denoise strength levels based on the sensor gain. A larger gain corresponds to higher denoise strength. The relationship between the image denoise strength and the sensor gain is described in <a href="#">Table 6-5</a> .



**Table 6-5** Values of u8SnrThresh[8] based on the sensor gain

Snr_thresh	Again*Dgian*IspDgain(times)
u8SnrThresh [0]	1
u8SnrThresh [1]	2
u8SnrThresh [2]	4
u8SnrThresh [3]	8
u8SnrThresh [4]	16
u8SnrThresh [5]	32
u8SnrThresh [6]	64
u8SnrThresh [7]	128

[Note]

- A larger u8ThreshTarget value indicates the greater denoise effect when manual denoise is enabled.
- When manual denoise is enabled, the actual denoise strength is the expected value u8ThreshTarget.

[See Also]

None

## 6.8 DIS

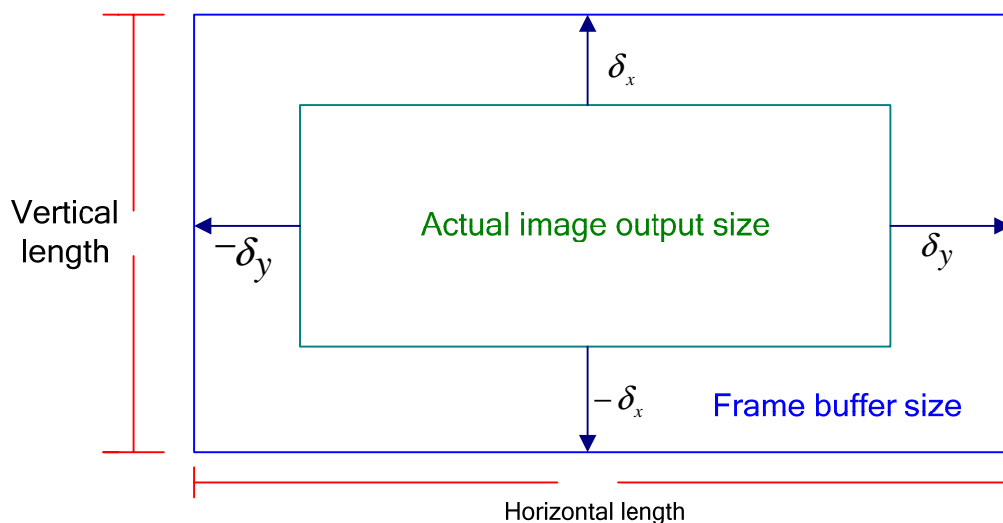
### 6.8.1 Overview

The digital image stabilization (DIS) module compares the current image with the previous frame to determine the image flicker degree. Then the DIS module adjusts the images output from the frame buffer based on the image offset value to stabilize the images.

### 6.8.2 Function Description

The DIS determines the horizontal offset  $\delta_x$  and vertical offset  $\delta_y$  for stabilizing each image. The offset range of horizontal or vertical output pixels is  $[-128, +128]$  by default. [Figure 6-2](#) shows the offset schematic diagram of the DIS module.

**Figure 6-2** Offset schematic diagram of the DIS module



As shown in [Figure 6-2](#), the green rectangle is the size of the image output by the ISP, and the blue rectangle is the size of the image stored in the frame buffer. The horizontal offset and

vertical offset for the image output by the DIS module are  $\delta_x$  and  $\delta_y$  respectively when an image flickers. You can perform operations based on the offset values to stabilize the image.

## 6.8.3 API Reference

The following are DIS APIs:

- [HI\\_MPI\\_ISP\\_SetDISAttr](#): Sets the digital image stabilizer (DIS) attribute.
- [HI\\_MPI\\_ISP\\_GetDISAttr](#): Gets the DIS attribute.
- [HI\\_MPI\\_ISP\\_GetDISInfo](#): Obtains DIS statistics.

### HI\_MPI\_ISP\_SetDISAttr

[Description]

Sets the DIS attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDISAttr(const ISP_DIS_ATTR_S *pstDISAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstDISAttr	DIS attribute	Input

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

## HI\_MPI\_ISP\_GetDISAttr

[Description]

Obtains the DIS attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDISAttr(ISP\_DIS\_ATTR\_S*pstDISAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstDISAttr	DIS attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]





Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetDISInfo

[Description]

Obtains DIS statistics.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDISInfo(ISP\_DIS\_INFO\_S*pstDISInfo);
```

[Parameter]

Parameter	Description	Input/Output
pstDISInfo	DIS attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]



- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

```
HI_U32 ChnId;

    VI_DRV_CHN_STORE_INFO* pstStoreCfg;
    static RECT_S      stCapRect[VICAP_CHANNEL_NUM] = {{0, 0, 1280, 720}};
    static HI_U32      OFFSET_X,OFFSET_Y,Multiplier;
    HI_U32              OFFSET_Tmp_X,OFFSET_Tmp_Y;
    HI_U32              compare_value_x,compare_value_y;
    static HI_U32      Previous_Value_X = 0;
    static HI_U32      Previous_Value_Y = 0;
    ISP_DIS_INFO      pstDISInfo;
    ISP_DIS_ATTR_S    pstDISAttr;
    pstDISAttr.

HI_MPI_ISP_SetDISAttr(const ISP_DIS_ATTR_S *pstDISAttr);

HI_MPI_ISP_GetDISInfo(&pstDISInfo);
OFFSET_X = pstDISInfo.s8Xoffset;
OFFSET_Y = pstDISInfo.s8Yoffset;
if(1 == OFFSET_X%2)
    OFFSET_X = OFFSET_X -1;
if(1 == OFFSET_Y%2)
    OFFSET_Y = OFFSET_Y -1;
Multiplier = 1;
    OFFSET_Tmp_X = OFFSET_X;
OFFSET_Tmp_Y = OFFSET_Y;
    compare_value_x = (OFFSET_X > Previous_Value_X)? (OFFSET_X -
Previous_Value_X):(Previous_Value_X - OFFSET_X);
    compare_value_y = (OFFSET_Y > Previous_Value_Y)? (OFFSET_Y -
Previous_Value_Y):(Previous_Value_Y - OFFSET_Y);
    if(0x80 < OFFSET_X){

if((compare_value_x<2)&&((0x100 - OFFSET_X) < 0xf))    /*****Filter out the
values that are less than 2 and whose amplitude is less than 0xf.*****/
OFFSET_X = 0x100;

stCapRect[ChnId].s32X = 128 + (0x100 - OFFSET_X) * Multiplier;
//Information input to vi_crop. stCapRect[ChnId].s32X is the start position
where the VIU crops frames.
```



```
stCapRect[ChnId].u32Width = 1280 + 128 + (0x100 - OFFSET_X) *
Multiplier; //Information input to vi_crop. stCapRect[ChnId].s32X is the start
position where the VIU crops frames.
}
else if(0x80 > OFFSET_X)
{
if((compare_value_x<2)&&(OFFSET_X < 0xf))
OFFSET_X = 0;
stCapRect[ChnId].s32X = 128 - OFFSET_X * Multiplier;
stCapRect[ChnId].u32Width = 1280 + 128 - OFFSET_X * Multiplier;
}
else if(0x80 == OFFSET_X)
{
stCapRect[ChnId].s32X = 128;
stCapRect[ChnId].u32Width = 1280 + 128;
}

if(0x80 < OFFSET_Y)
{
if((compare_value_y<2)&&((0x100 - OFFSET_Y) < 0x10))
OFFSET_Y = 0x100;
stCapRect[ChnId].s32Y = 128 + (0x100 - OFFSET_Y) * Multiplier;
stCapRect[ChnId].u32Height = 720 + 128 + (0x100 - OFFSET_Y) * Multiplier;
}
if(0x80 > OFFSET_Y)
{
if((compare_value_y<2)&&(OFFSET_Y < 0x10))
OFFSET_Y = 0;
stCapRect[ChnId].s32Y = 128 - OFFSET_Y*Multiplier;
stCapRect[ChnId].u32Height = 720 + 128 - OFFSET_Y * Multiplier;
}
else if(0x80 == OFFSET_Y)
{
stCapRect[ChnId].s32Y = 128;
stCapRect[ChnId].u32Height = 720 + 128;
}

Previous_Value_X = OFFSET_Tmp_X;
Previous_Value_Y = OFFSET_Tmp_Y;
VI_DRV_SetChCrop(0, &stCapRect[ChnId]);
pstStoreCfg->u32Width = stCapRect[ChnId].u32Width;
pstStoreCfg->u32Height = stCapRect[ChnId].u32Height;
VI_DRV_SetChDes(ChnId, pstStoreCfg);
md_set_vdp_rect(HAL_DISP_LAYER_VSD1, 0, 0,
```



```
stCapRect [ChnId] .u32Width, stCapRect [ChnId] .u32Height);
```

[See Also]

None

## 6.8.4 Data Structures

The following are the data structures related to the DIS:

- [ISP\\_DIS\\_ATTR\\_S](#): Defines the DIS attribute.
- [ISP\\_DIS\\_INFO\\_S](#): Defines the information output by the DIS module.

### ISP\_DIS\_ATTR\_S

[Description]

Defines the DIS attribute.

[Syntax]

```
typedef struct hiISP_DIS_ATTR_S
{
    HI_BOOL bEnable;
} ISP_DIS_ATTR_S;
```

[Member]

Member	Description
bEnable	DIS enable

[Note]

None

[See Also]

None

### ISP\_DIS\_INFO\_S

[Description]

Defines the information output by the DIS module.

[Syntax]

```
typedef struct hiISP_DIS_INFO_S
{
    HI_S8 s8Xoffset;
    HI_S8 s8Yoffset;
} ISP_DIS_INFO_S;
```

[Member]

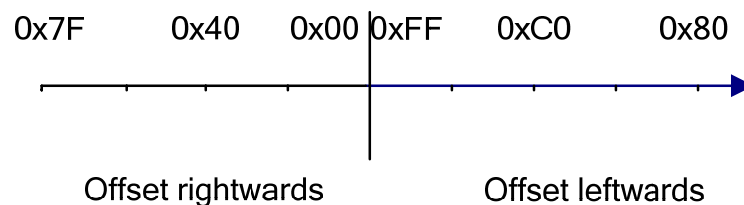


Member	Description
s8Xoffset	Horizontal offset for the image output by the DIS module Value range: [0x00, 0xFF]
s8Yoffset	Vertical offset for the image output by the DIS module Value range: [0x00, 0xFF]

[Note]

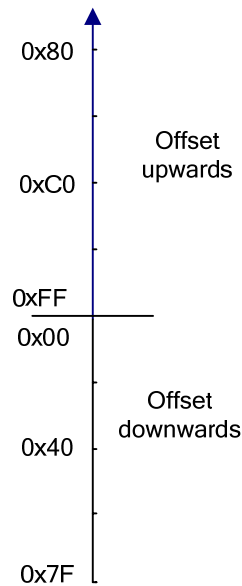
s8Xoffset is a signed number. When an image is offset leftwards, the output range is [0x80, 0xFF]. The value is expressed by ones-complement code. The value 0xFF indicates that the image is offset leftwards by one pixel, and the value 0xFE indicates that the image is offset leftwards by two pixels. When an image is offset rightwards, the output range is [0x00, 0x80]. The value 0x1 indicates that the image is offset rightwards by one pixel, and so on. See [Figure 6-3](#).

**Figure 6-3** DIS horizontal offset



s8Yoffset is a signed number. When an image is offset upwards, the output range is [0x80, 0xFF]. The value is expressed by ones-complement code. The value 0xFF indicates that the image is offset upwards by one pixel, and the value 0xFE indicates that the image is offset upwards by two pixels. When an image is offset downwards, the output range is [0x00, 0x80]. The value 0x1 indicates that the image is offset downwards by one pixel, and so on. See [Figure 6-4](#).

**Figure 6-4** DIS vertical offset



[See Also]

None

## 6.9 Anti-fog

### 6.9.1 Function Description

The anti-fog function is implemented by dynamically changing the image contrast and luminance. The anti-fog module estimates the fog thickness based on statistics on frames. The anti-fog algorithm takes effect only when the fog is heavy and is full of the entire frame. Otherwise, the anti-fog module considers that there is no fog.

### 6.9.2 API Reference

The following are anti-fog MPIs:

- [HI\\_MPI\\_ISP\\_SetAntiFogAttr](#): Sets the anti-fog attribute.
- [HI\\_MPI\\_ISP\\_GetAntiFogAttr](#): Obtains the anti-fog attribute.

#### HI\_MPI\_ISP\_SetAntiFogAttr

[Description]

Set the anti-fog attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAntiFogAttr(const ISP\_ANTIFOG\_S*pstAntiFog)
```

[Parameter]



Parameter	Description	Input/Output
pstAntiFog	Anti-fog attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetDenoiseAttr](#)

## HI\_MPI\_ISP\_GetAntiFogAttr

[Description]

Obtains the anti-fog attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAntiFogAttr(ISP\_ANTIFOG\_S *pstAntiFog)
```

[Parameter]

Parameter	Description	Input/Output
pstAntiFog	Anti-fog attribute	Output

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetDenoiseAttr](#)

## 6.9.3 Data Structure

### ISP\_ANTIFOG\_S

[Description]

Defines the ISP anti-fog attribute.

[Syntax]

```
typedef struct hiISP_ANTIFOG_S
{
    HI_BOOL bEnable;
    HI_U8  u8Strength;
} ISP_ANTIFOG_S;
```

[Member]

Member	Description
bEnable	Anti-fog enable
u8Strength	Anti-fog strength, ranging from [0, 255]





[Note]

None

[See Also]

None

## 6.10 Anti-False Color

### 6.10.1 Overview

The anti-false color function is used to remove the moires in the high-frequency parts of images.

### 6.10.2 Function Description

High-frequency aliasing occurs in high-frequency components when image interpolation is used. When a lens focuses on a resolution test card, the high-frequency part has false colors if there is no optical low-pass filter (OLPF) on the sensor surface.

### 6.10.3 API Reference

The following are MPIs related to anti-false color:

- [HI\\_MPI\\_ISP\\_SetAntiFalseColorAttr](#): Sets the anti-false color attribute.
- [HI\\_MPI\\_ISP\\_GetAntiFalseColorAttr](#): Obtains the anti-false color attribute.

#### HI\_MPI\_ISP\_SetAntiFalseColorAttr

[Description]

Sets the anti-false color attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAntiFalseColorAttr(const ISP_ANTI_FALSECOLOR_S  
*pstAntiFC)
```

[Parameter]

Parameter	Description	Input/Output
pstAntiFC	Anti-false color attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.



[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetAntiFalseColorAttr

[Description]

Obtains the anti-false color attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAntiFalseColorAttr(const ISP\_ANTI\_FALSECOLOR\_S  
*pstAntiFC)
```

[Parameter]

Parameter	Description	Input/Output
pstAntiFC	Anti-false color attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]



Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None

## 6.10.4 Data Structure

### ISP\_ANTI\_FALSECOLOR\_S

[Description]

Defines the ISP anti-false color attribute.

[Syntax]

```
typedef struct hiISP_ANTI_FALSECOLOR_S
{
    HI_U8  u8Strength;
} ISP_ANTI_FALSECOLOR_S;
```

[Member]

Member	Description
u8Strength	Anti-false color strength Value range: [0x0, 0x95]

[Note]

If u8Strength is 0, the anti-false color function is unavailable.

[See Also]

None



## 6.11 Demosaic

### 6.11.1 Function Description

Demosaic indicates that the input Bayer data is converted into RGB data.

### 6.11.2 API Reference

- [HI\\_MPI\\_ISP\\_SetDemosaicAttr](#): Sets the demosaic attribute.
- [HI\\_MPI\\_ISP\\_GetDemosaicAttr](#): Obtains the demosaic attribute.

#### HI\_MPI\_ISP\_SetDemosaicAttr

[Description]

Sets the demosaic attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDemosaicAttr(ISP_DEMOSAIC_ATTR_S *pstDemosaicAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstDemosaicAttr	Demosaic attribute	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]



None

[See Also]

None

## HI\_MPI\_ISP\_GetDemosaicAttr

[Description]

Obtains the demosaic attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDemosaicAttr(ISP_DEMOSAIC_ATTR_S *pstDemosaicAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstDemosaicAttr	Demosaic attribute	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. Its value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

None



## 6.11.3 Data Structure

### ISP\_DEMOSAIC\_ATTR\_S

#### [Description]

Defines the ISP demosaic attribute.

#### [Syntax]

```
typedef struct hiISP_DEMOSAIC_ATTR_S
{
    HI_U8    u8VhSlope;    /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8AaSlope;    /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8VaSlope;    /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8UuSlope;    /*RW, Range: [0x0, 0xFF] */
    HI_U16   u16VhThresh; /*RW, Range: [0x0, 0xFFFF] */
    HI_U16   u16AaThresh; /*RW, Range: [0x0, 0xFFFF] */
    HI_U16   u16VaThresh; /*RW, Range: [0x0, 0xFFFF] */
    HI_U16   u16UuThresh; /*RW, Range: [0x0, 0xFFFF] */
    HI_U8    u8DemosaicConfig; /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8LumThresh[8]; /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8NpOffset[8]; /*RW, Range: [0x0, 0xFF] */
} ISP_DEMOSAIC_ATTR_S;
```

#### [Member]

Member	Description
u8VhSlope	Slope for the vertical and horizontal edge thresholds. Increasing this parameter value increases the horizontal resolution, vertical resolution, and noise. Value range: [0x00, 0xFF]. The default value is defined by the <b>vh_slope</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u8AaSlope	Slope for the 45° and 135° edge thresholds. Increasing this parameter value increases the slant resolution and noise. Value range: [0x00, 0xFF]. The default value is defined by the <b>aa_slope</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u8VaSlope	Slope for the vertical, horizontal, 45°, and 135° edge thresholds. Increasing this parameter value increases the horizontal resolution, vertical resolution, slant resolution, and noise. Value range: [0x00, 0xFF]. The default value is defined by the <b>va_slope</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u8UuSlope	Slope for all edge thresholds. Increasing this parameter value



Member	Description
	increases the resolution, sharpness, and noise and causes false color. Value range: [0x00, 0xFF]. The default value is defined by the <b>uu_slope</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u16VhThresh	Threshold for vertical and horizontal edges. Increasing this parameter value decreases the false color, noise, vertical resolution, and horizontal resolution. Value range: [0x0, 0xFFFF]. The default value is defined by the <b>vh_thresh</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u16AaThresh	Threshold for 45° and 135° edges. Increasing this parameter value decreases the false color, noise, and slant resolution. Value range: [0x0, 0xFFFF]. The default value is defined by the <b>aa_thresh</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u16VaThresh	Threshold for vertical, horizontal, 45°, and 135° edges. Increasing this parameter value decreases the false color, noise, vertical resolution, horizontal resolution, and slant resolution. Value range: [0x00, 0xFF]. The default value is defined by the <b>va_thresh</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u16UuThresh	Threshold for all edges. Increasing this parameter value decreases the false color, noise, resolution, and sharpness. Value range: [0x00, 0xFF]. The default value is defined by the <b>uu_thresh</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.
u8DemosaicConfig	Debugging mode of the demosaic module. Value range: [0x00, 0xFF]. 0: normal output 4: UU debugging mode 17: AA debugging mode 18: VA debugging mode 19: VH debugging mode Other values: reserved
u8LumThresh	A larger value indicates the more obvious picture edge. The eight values of this array correspond to different ISO values. For details, see <a href="#">Table 6-6</a> .
u8NpOffset	Offset of the demosaic noise profile. The eight values of this array correspond to different ISO values. For details, see <a href="#">Table 6-7</a> .



**Table 6-6** Mapping between the values of u8LumThresh and gain

u8LumThresh	Again*Dgian*IspDgain (times)
u8LumThresh [0]	1
u8LumThresh [1]	2
u8LumThresh [2]	4
u8LumThresh [3]	8
u8LumThresh [4]	16
u8LumThresh [5]	32
u8LumThresh [6]	64
u8LumThresh [7]	128

**Table 6-7** Mapping between the values of u8NpOffset and gain

u8NpOffset	Again*Dgian*IspDgain (times)
u8NpOffset [0]	1
u8NpOffset [1]	2
u8NpOffset [2]	4
u8NpOffset [3]	8
u8NpOffset [4]	16
u8NpOffset [5]	32
u8NpOffset [6]	64
u8NpOffset [7]	128

[Note]

- A smaller slope indicates that fewer edges at some angles are involved in mixing.
- A larger thresh value indicates a narrower angle judgment range. When the thresh value is too large, only the vertical, horizontal, 45°, and 135° edges are taken into account.
- The parameters of this data structure are advanced parameters. You are advised not to change the parameter values.

[See Also]

None





## 6.12 Black Level

### 6.12.1 Overview

The black level indicates the output luminance of the sensor when there is no light source. The ISP needs to subtract the luminance for processing colors.

### 6.12.2 API Reference

The following are black level MPIS:

- [HI\\_MPI\\_ISP\\_SetBlackLevelAttr](#): Sets black level attributes.
- [HI\\_MPI\\_ISP\\_GetBlackLevelAttr](#): Obtains black level attributes.

#### HI\_MPI\_ISP\_SetBlackLevelAttr

[Description]

Sets black level attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetBlackLevelAttr(const ISP_BLACK_LEVEL_S *pstBlackLevel)
```

[Parameter]

Parameter	Description	Input/Output
pstBlackLevel	Black level attributes	Input

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]



None

[Example]

None

[See Also]

None

## HI\_MPI\_ISP\_GetBlackLevelAttr

[Description]

Obtains black level attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetBlackLevelAttr(ISP_BLACK_LEVEL_S *pstBlackLevel)
```

[Parameter]

Parameter	Description	Input/Output
pstBlackLevel	Black level attributes	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]



None

## 6.12.3 Data Structures

### ISP\_BLACK\_LEVEL\_S

[Description]

Defines the attributes of the ISP black level.

[Syntax]

```
typedef struct hiISP_BLACK_LEVEL_S
{
    HI_U16 au16BlackLevel[4]; /*RW, Range: [0x0, 0xFFFF]*/
} ISP_BLACK_LEVEL_S;
```

[Member]

Member	Description
au16BlackLevel[4]	Black level values that correspond to the black levels of the R, Gr, Gb, and B components respectively. Value range: [0x0, 0xFFFF] The default values are defined by the <b>black_level[4]</b> member of <b>cmos_isp_demosaic_t</b> in the <b>cmos.c</b> file.

[Note]

None

[See Also]

None

## 6.13 Obtaining ISP Virtual Addresses

### 6.13.1 Function Description

The ISP consists of the AE library module, AWB library module, AF library module, ISP physical register module, and other module. Each module has a start address.

### 6.13.2 API Reference

#### HI\_MPI\_GetISPRegAttr

[Description]

Obtains the attributes of ISP virtual addresses.



[Syntax]

```
HI_S32 HI_MPI_ISP_GetISPRegAttr(ISP_REG_ATTR_S *pstIspRegAttr)
```

[Parameter]

Parameter	Description	Input/Output
pstIspRegAttr	Attributes of ISP virtual addresses.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

The virtual address for the AF library module is not provided currently.

[Example]

None

[See Also]

None

## 6.13.3 Data Structure

### ISP\_REG\_ATTR\_S

[Description]

Defines the attributes of the virtual addresses for the registers of ISP submodules.

[Syntax]

```
typedef struct hiISP_REG_ATTR_S  
{
```



```
HI_U32 u32IspRegAddr;  
HI_U32 u32IspRegSize;  
HI_U32 u32IspExtRegAddr;  
HI_U32 u32IspExtRegSize;  
HI_U32 u32AeExtRegAddr;  
HI_U32 u32AeExtRegSize;  
HI_U32 u32AwbExtRegAddr;  
HI_U32 u32AwbExtRegSize;  
} ISP_REG_ATTR_S;
```

[Member]

Member	Description
u32IspRegAddr	Start virtual address for the ISP internal physical registers.
u32IspRegSize	Size of the ISP internal physical registers.
u32IspExtRegAddr	Start virtual address for the ISP external virtual registers.
u32IspExtRegSize	Size of the ISP external virtual registers.
u32AeExtRegAddr	Start virtual address for the AE library module.
u32AeExtRegSize	Size of the AE library module.
u32AwbExtRegAddr	Start virtual address for the AWB library module.
u32AwbExtRegSize	Size of the AWB library module.

[Note]

None

[See Also]

None



# 7 ISP Debugging Information

## 7.1 Overview

Two MPIs are provided for debugging some modules in the ISP, helping customers to locate picture quality issues.

## 7.2 Function Description

Enable a debugging module such as the AE, AWB, and SYS to obtain the module status when the ISP runs. You can view the debugging function to record a desired module status when the ISP runs, helping to locate exceptions. You can also specify the number of recorded frames.

## 7.3 API Reference

- [HI\\_MPI\\_ISP\\_SetDebug](#): Sets an ISP debugging MPI.
- [HI\\_MPI\\_ISP\\_GetDebug](#): Obtains an ISP debugging MPI.

### HI\_MPI\_ISP\_SetDebug

[Description]

Sets attributes of the MPI for debugging a desired module.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDebug(ISP\_DEBUG\_INFO\_S *pstIspDebug)
```

[Parameter]

Parameter	Description	Input/Output
pstIspDebug	Debugging information	Input

[Return Value]



Return Value	Description
0	Success.
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

- You can enable any of the AE, AWB, and SYS separately in the debugging information.
- You can allocate the memory space for storing corresponding debugging information, and specify the memory address for the corresponding data structure as an input parameter.

[Example]

```
HI_S32 s32Ret;
HI_U32 u32PhyAddr;
HI_VOID *pVitAddr;
FILE *fp;
ISP_DEBUG_INFO_S stIspDebug;
PAUSE;

CHECK_RET(HI_MPI_ISP_GetDebug(&stIspDebug), "ISP debug get");

stIspDebug.u32DebugDepth = 30; /* */

HI_U32 u32SizeHi = (stIspDebug.u32AWBSize & 0xFFFF0000) >> 16; /*status*/
HI_U32 u32SizeLo = stIspDebug.u32AWBSize & 0xFFFF; /*cfg*/
HI_U32 u32MemSize = u32SizeLo + u32SizeHi * stIspDebug.u32DebugDepth;
s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&u32PhyAddr, &pVitAddr, NULL, NULL,
u32MemSize);

stIspDebug.u32AWBAddr = u32PhyAddr;
stIspDebug.bAWBDebugEnable = 1;

CHECK_RET(HI_MPI_ISP_SetDebug(&stIspDebug), "ISP debug set");
```



```
    PAUSE;

    HI_U32 *pu32VirAddr = (HI_U32 *)HI_MPI_SYS_Mmap(stIspDebug.u32AWBAddr,
u32MemSize);

    fp=fopen("mst_30000.dat","wb");
    if(fp==NULL)
    {
        printf("open file mst_30000.dat error \n");
        return -1;
    }

    fwrite(pu32VirAddr,1,u32MemSize,fp);
    printf("write file\n");
    PAUSE;
fclose(fp);

    stIspDebug.bAWBDebugEnable = 0;
    CHECK_RET(HI_MPI_ISP_SetDebug(&stIspDebug), "ISP debug set");

    HI_MPI_SYS_Munmap(pu32VirAddr, u32MemSize);
    HI_MPI_SYS_MmzFree(u32PhyAddr, pVitAddr);
```

[See Also]

None

## HI\_MPI\_ISP\_GetDebug

[Description]

Obtains attributes of the MPI for debugging a module.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDebug(ISP\_DEBUG\_INFO\_S*pstIspDebug)
```

[Parameter]

Parameter	Description	Input/Output
pstIspDebug	Debugging information	Output

[Return Value]

Return Value	Description
0	Success.





Return Value	Description
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

## 7.4 Data Structure

### ISP\_DEBUG\_INFO\_S

[Description]

Defines ISP debugging information attributes.

[Syntax]

```
typedef struct hiISP_DEBUG_INFO_S
{
    HI_BOOL  bAEDebugEnable ;
    HI_U32   u32AEAddr ;
    HI_U32   u32AESize ;
    HI_BOOL  bAWBDebugEnable ;
    HI_U32   u32AWBAddr ;
    HI_U32   u32AWBSize ;
    HI_U32   u32SysDebugEnable ;
    HI_U32   u32SysAddr ;
    HI_U32   u32SysSize ;
    HI_U32   u32DebugDepth ;
}
```

[Member]



Member	Description
bAEDebugEnable	AE debugging enable. You can obtain AE status after enabling this member.
u32AEAddr	AE debugging address. AE debugging information is written to the allocated memory space specified by this AE debugging address.
u32AESize	<p>Memory space for storing AE debugging information. The lower 16 bits indicate the memory space for storing fixed information, and the upper 16 bits indicate the memory space for storing module status information of a frame.</p> <p>Memory space for storing AE debugging information is calculated as follows: Number of total frames x Memory space for storing module status information of a frame + Memory space for storing fixed information.</p> <p>This member is read-only and cannot be written by using HI_MPI_ISP_SetDebug.</p>
bAWBDebugEnable	AWB debugging enable. You can obtain AWB status after enabling this member.
u32AWBAddr	AWB debugging address. AWB debugging information will be written to the allocated memory space specified by the AWB debugging address.
u32AWBSize	<p>Memory space for storing AWB debugging information. The lower 16 bits indicate the memory space for storing fixed information, and the upper 16 bits indicate the memory space for storing module status information of a frame.</p> <p>Memory space for storing AWB debugging information is calculated as follows: Number of total frames x Memory space for storing module status information of a frame + Memory space for storing fixed information.</p> <p>This member is read-only and cannot be written by using HI_MPI_ISP_SetDebug.</p>
u32SysDebugEnable	SYS debugging enable. You can obtain SYS debugging information after enabling this member.
u32SysAddr	SYS debugging address. SYS debugging information is written to the allocated memory space specified by the SYS debugging address.
u32SysSize	<p>Memory space for storing SYS debugging information. The lower 16 bits indicate the memory space for storing fixed information, and the upper 16 bits indicate the memory space for storing module status information of a frame.</p> <p>Memory space for storing SYS debugging information is calculated as follows: Number of total frames x Memory space for storing module status information of a frame + Memory space for storing fixed information.</p> <p>This member is read-only and cannot be written by using HI_MPI_ISP_SetDebug.</p>



Member	Description
u32DebugDepth	Debugging depth that indicates the number of frames for obtaining debugging information

[Note]

Debugging information indicates status information during ISP running. Determine the debugging module, specify the number of frames in debugging information, allocate the memory space for storing corresponding debugging information, and specify the address for the corresponding data structure as an input parameter. For example, if you want to obtain AE debugging status information of 30 frames (`u32DebugDepth = 30`), the memory space for storing AE debugging information is calculated as follows:  $[(u32AESize \& 0xFFFF0000) \gg 16] * u32DebugDepth + u32AESize \& 0xFFFF$ .

[See Also]

None



# 8 AF

## 8.1 Overview

By using the image processing technology, the quality of imaging is analyzed to obtain the current focus status of the system, and then the camera is driven to adjust the focal length of lens. This process is called automatic focus (AF). In the image processing technology, the gray scale difference and quantity of high frequency components are analyzed to check whether an image is a clear image in focus state.

## 8.2 Function Description

The AF module provides AF statistics including the normalized contract of an entire frame and normalized contract by zone.

## 8.3 API Reference

The following are AF MPIs:

- [HI\\_MPI\\_ISP\\_SetFocusStaInfo](#): Sets AF statistics.
- [HI\\_MPI\\_ISP\\_GetFocusStaInfo](#): Obtains AF statistics.

### HI\_MPI\_ISP\_SetFocusStaInfo

[Description]

Sets AF statistics.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetFocusStaInfo (const ISP_FOCUS_STA_INFO_S  
*pstFocusStatistics)
```

[Parameter]

Parameter	Description	Input/Output
pstFocusStatistic	AF statistics	Input



[Return Value]

Return Value	Description
0	Success.
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_GetFocusStaInfo](#)

## HI\_MPI\_ISP\_GetFocusStaInfo

[Description]

Obtains AF statistics.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetFocusStaInfo(ISP\_FOCUS\_STA\_INFO\_S *pstFocusStatistic)
```

[Parameter]

Parameter	Description	Input/Output
pstFocusStatistic	AF statistics	Output

[Return Value]

Return Value	Description
0	Success.



Return Value	Description
Other values	Failure. The return value is an error code.

[Error Code]

Error Code	Description
<a href="#">HI_ERR_ISP_NULL_PTR</a>	The pointer is null.

[Requirement]

- Header files: hi\_comm\_isp.h, mpi\_isp.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

[HI\\_MPI\\_ISP\\_SetFocusStaInfo](#)

## 8.4 Data Structures

### ISP\_FOCUS\_STA\_INFO\_S

[Description]

Defines AF statistics of the ISP.

[Syntax]

```
typedef struct hiISP_FOCUS_STA_INFO_S
{
    HI_U16 u16FocusMetrics;
    HI_U16 u16ThresholdRead;
    HI_U16 u16ThresholdWrite;
    HI_U16 u16FocusIntensity;
    HI_U8 u8MetricsShift;
    HI_U8 u8NpOffset;
    HI_U16 u16ZoneMetrics[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_FOCUS_STA_INFO_S;
```

[Member]



Member	Description
u16FocusMetrics	Normalized contrast of an entire frame Value range: [0x0, 0xFFFF]
u16ThresholdRead	For debugging
u16ThresholdWrite	For debugging
u16FocusIntensity	For debugging
u8MetricsShift	Proportion factor of statistics information value, with a default value of <b>3</b>
u8NpOffset	Statistics information value and Noise Profile
u16ZoneMetrics[WEIGHT_ZONE_ROW] [WEIGHT_ZONE_COLUMN]	Normalized contrast by zone Value range: [0x0, 0xFFFF]

[Note]

None

[See Also]

None



# 9 Error Codes

Table 9-1 describes the error codes for ISP APIs.

**Table 9-1** Error codes for ISP APIs

Error Code	Macro Definition	Description
0xA00A8006	HI_ERR_ISP_NULL_PTR	The input pointer is null.
0xA00A8003	HI_ERR_ISP_ILLEGAL_PARAM	The input parameter is invalid.
0xA00A8043	HI_ERR_ISP_SNS_UNREGISTER	The sensor is not registered.
0xA00A0040	HI_ERR_ISP_NOT_INIT	The ISP is not initialized.
0xA00A0041	HI_ERR_ISP_TM_NOT_CFG	The timing is not configured.
0xA00A0044	HI_ERR_ISP_INVALID_ADDR	The address is invalid.
0xA00A0042	HI_ERR_ISP_ATTR_NOT_CFG	The attribute is not configured.





# 10 Proc Debugging Information

## 10.1 Overview

The debugging information uses the proc file system of Linux. It reflects the running status of the current system in real time. The information recorded can be used for troubleshooting and analysis.

[File directory]

/proc/umap/isp

[Loading method]

- When loading .ko files of the ISP, add the module parameter **proc\_param=n**. If **n** is **0**, the proc information is disabled; if **n** is not **0**, the proc information is updated every **n** frames. For example, "**insmod hi3518\_isp.ko proc\_param=30**" indicates that the proc information is updated every 30 frames
- The CPU resource is consumed when proc information is enabled. You are advised to update the proc information once every 30 frames or enable proc information only during debugging.

[Method of viewing information]

- You can run a cat command to view information on the console, for example, cat /proc/umap/isp. You can also use other common file operation commands to copy the **proc** file of the ISP to the current directory, for example, **cp /proc/umap/isp ./-rf**.
- You can regard the preceding file as a common read-only file to perform read operations in an application, for example, fopen and fread.



### NOTE

Note the following two conditions when describing parameters:

- For a parameter whose value range is {0, 1}, if the mapping between the actual value and description of this parameter is not listed, **1** indicates yes and **0** indicates no.
- For a parameter whose value range is {aaa, bbb, ccc}, if the mapping between the actual value and description of this parameter is not listed, judge the parameter description directly based on the values **aaa**, **bbb**, and **ccc**.

## 10.2 ISP

[Debugging information]



```
# cat /proc/umap/isp
```

```
[ISP] Version: [Hi3518_ISP_V1.0.0.0 Release], Build Time[Aug 12 2013,  
14:17:34]
```

```
-----AE INFO-----  
Again  AShift  Dgain  DShift  IspDg  IShift  SysGain  SShift  Iso  Line  
162      4      16      4      16      4      648      6    1000  1122
```

```
Comp    Step    Tole    Error    Fps  SlowFrt  MaxLine  MaxAg  MaxDg  MaxIDg  
128      2     10     61     30     16    65535    256   128    64
```

```
Target0 Target1 Target2 Target3 Target4 Thresh0 Thresh1 Thresh2 Thresh3  
0x 4000 0x 4000 0x 3000 0x 4000 0x 2000      13      40      96     128
```

```
ManuEn  MaLine  MaAg  MaDg  WdrMode  AuFlick  SlowMod  UpMode  
0        0        0        0    LINE      0        1        0
```

```
-----AWB INFO-----  
Gain0  Gain1  Gain2  Gain3  CoTemp  
0x1c3  0x116  0x116  0x2a4  4098
```

```
Color00 Color01 Color02 Color10 Color11 Color12 Color20 Color21 Color22  
0x 217  0x80fb  0x801e  0x8063  0x 1c4  0x8063  0x 14  0x80d3  0x 1bd
```

```
ManuEn  MaSatEn  Sat  SatTg  Zones  Speed  
0        0     122  128    32     25
```

```
-----DRC INFO-----  
En  ManuEn  DrcSt  DrcStTg  
0    0     66    128
```

```
-----DEMOSAIC INFO-----  
VhSlope  VhTh  AaSlope  AaTh  VaSlope  VaTh  UuSlope  UuTh  
0xf5     0x 32  0x98     0x 5b    0xe6   0x 52    0x90    0x 40
```

```
SatSlope  SatTh  AcSlope  AcTh  
0x5d     0x171  0xcf     0x1b3
```

```
-----NR INFO-----  
En  ManuEn  Thresh  Offset  
1    0     25     26
```

```
-----SHARPEN INFO-----
```



En	ManuEn	SpD	SpUd	LumTh	GeSt
1	0	122	174	128	85

-----SHADING INFO-----

En  
0

#

[Debugging information analysis]

Record the usage of the current ISP module.

[Parameter description]

Parameter		Description
AE INFO	Again	Analog gain of the sensor
	AShift	Analog gain precision of the sensor
	Dgain	Digital gain of the sensor
	DShift	Digital gain precision of the sensor
	IspDg	Digital gain of the ISP
	IShift	Digital gain precision of the ISP
	SysGain	System gain
	SShift	System gain precision
	Iso	ISO value
	Line	Exposure duration of the sensor, which is calculated by rows
	Comp	Exposure compensation during automatic exposure adjustment
	Step	Initial step during automatic exposure adjustment
	Tole	Exposure tolerance during automatic exposure adjustment
	Error	Exposure error during automatic exposure adjustment
	Fps	Frame rate
	SlowFrt	Frame rate reduction times
	MaxLine	Maximum exposure duration
	MaxAg	Maximum analog gain of the sensor
	MaxDg	Maximum digital gain of the sensor
	MaxIDg	Maximum digital gain of the ISP
	Target0	Target value of the first segment of the 5-segment histogram during automatic exposure adjustment



Parameter		Description
	Target1	Target value of the second segment of the 5-segment histogram during automatic exposure adjustment
	Target2	Target value of the third segment of the 5-segment histogram during automatic exposure adjustment
	Target3	Target value of the fourth segment of the 5-segment histogram during automatic exposure adjustment
	Target4	Target value of the fifth segment of the 5-segment histogram during automatic exposure adjustment
	Thresh0	Separation point between the first and second segments of the 5-segment histogram during automatic exposure adjustment
	Thresh1	Separation point between the second and third segments of the 5-segment histogram during automatic exposure adjustment
	Thresh2	Separation point between the third and fourth segments of the 5-segment histogram during automatic exposure adjustment
	Thresh3	Separation point between the fourth and fifth segments of the 5-segment histogram during automatic exposure adjustment
	ManuEn	Manual exposure switch
	MaLine	Manual exposure duration
	MaAg	Analog gain of the sensor exposed manually
	MaDg	Digital gain of the sensor exposed manually
	WdrMode	WDR mode
	AuFlick	Automatic anti-twinkle switch
	SlowMod	Automatic frame rate reduction mode
	UpMode	Exposure variable update mode
AWB INFO	Gain0	Gain of channel R in white balance mode
	Gain1	Gain of channel Gr in white balance mode
	Gain2	Gain of channel Gb in white balance mode
	Gain3	Gain of channel B in white balance mode
	CoTemp	Color temperature detected currently
	Color00	RR value of color restoration matrix
	Color01	RG value of color restoration matrix
	Color02	RB value of color restoration matrix
	Color10	GR value of color restoration matrix
	Color11	GG value of color restoration matrix



Parameter		Description
	Color12	GB value of color restoration matrix
	Color20	BR value of color restoration matrix
	Color21	BG value of color restoration matrix
	Color22	BB value of color restoration matrix
	ManuEn	Manual white balance switch
	MaSatEn	Manual saturation switch
	Sat	Saturation value
	SatTg	Target saturation value
	Zones	Automatic white balance algorithm selection
	Speed	Convergence speed of automatic white balance
DRC INFO	En	DRC switch
	ManuEn	Manual DRC switch
	DrcSt	DRC strength value
	DrcStTg	Target DRC strength value
DEMOSA IC INFO	VhSlope	Slope of the mixed threshold of the vertical and horizontal edges
	VhTh	Threshold of the mixed range of the vertical and horizontal edges
	AaSlope	Slope of the mixed threshold of the 45° and 135° edges
	AaTh	Threshold of the mixed range of the 45° and 135° edges
	VaSlope	Slope of the mixed threshold of the vertical, horizontal, 45°, and 135° edges
	VaTh	Threshold of the mixed range of the vertical, horizontal, 45°, and 135° edges
	UuSlope	Slope of the mixed threshold of all edges
	UuTh	Threshold of the mixed range of all edges
	SatSlope	Slope of the mixed threshold of saturation
	SatTh	Threshold of the mixed range of saturation
	AcSlope	Slope of the mixed threshold of DC component
	AcTh	Threshold of the mixed range of DC component
NR INFO	En	Spatial noise reduction switch
	ManuEn	Manual spatial noise reduction switch



Parameter		Description
	Thresh	Target threshold of noise processing
	Offset	Target offset of noise processing
SHARPEN INFO	En	Sharpen switch
	ManuEn	Manual sharpen switch
	SpD	Acuity of the large edge of an image
	SpUd	Acuity of the small texture of an image
	LumTh	Luma Thresh
	GeSt	Ge Strength
SHADING INFO	En	Lens shading correction switch