



HiISP

Development Reference

Issue	03
Date	2014-02-26

Copyright © HiSilicon Technologies Co., Ltd. 2012-2014. 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



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 Versions

The following table lists the product versions 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
 DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.



Symbol	Description
 CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	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 official release, which incorporates the following changes:

Chapter 3 AE

In section 3.5.1, the description in **Note** field of ISP_AE_MODE_E is updated, and [ISP_EXP_STA_INFO_S](#) is deleted. The description of the maximum and minimum exposure time is updated in the **Note** field of ISP_AE_ATTR_S.

In section 3.5.2, **s16HistError** is added to the **Syntax** and **Member** fields of ISP_EXP_STA_INFO_S.

Chapter 4 AWB

In section 4.4.1, 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. The note "The Hi3516 does not support this MPI" is added to the **Note** field of HI_MPI_ISP_SetAWBAlgType

In section 4.5.1, 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 6.2.2, the descriptions in the **Note** field of HI_MPI_ISP_GetGammaTable are deleted.

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.

Chapter 10 Proc Debugging Information

In section 10.1, the loading method is updated.

Issue 02 (2013-09-25)

This issue is the second official 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 as well as the **Note** field are added to `ISP_EXP_STA_INFO_S`.

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 (2013-06-30)

This issue is the first official release, which incorporates the following changes:

Chapter 2 System Control

In section 2.2, `HI_MPI_ISP_GetVDTimeOut` is added.

In section 2.3, `ISP_VD_INFO_S` is added.

Chapter 3 AE

In section 3.4.1, the MPIS `HI_MPI_ISP_SetAEAttrEx`, `HI_MPI_ISP_GetAEAttrEx`, `HI_MPI_ISP_SetMEAttrEx`, `HI_MPI_ISP_GetMEAttrEx`, `HI_MPI_ISP_QueryInnerStateInfo`, and `HI_MPI_ISP_QueryInnerStateInfoEx` are added.

In section 3.5.1, **u16SystemGainMax** and **u16SystemGainShift** are added to the **Syntax** and **Member** fields of `ISP_AE_ATTR_S`, and the description of the mode of updating exposure variables is updated in the **Note** field of `ISP_AE_ATTR_S`.

The data structures `ISP_AE_ATTR_EX_S`, `ISP_ME_ATTR_EX_S`, `ISP_INNER_STATE_INFO_S`, and `ISP_INNER_STATE_INFO_EX_S` are added.

Chapter 4 AWB

In section 4.5.1, **tAWBCalibration** and **u8Speed** are added to the **Syntax** and **Member** fields of `ISP_AWB_ATTR_S`, and `ISP_AWB_CALIBRATION_S` is added.



Chapter 6 IMP

In section 6.1.3, the description of **u8SharpenAltUd[8]** is updated in the **Member** field of ISP_SHARPEN_ATTR_S.

In section 6.5.4, **u16DynamicBadPixelSlope** and **u16DynamicBadPixelThresh** are added to the **Syntax** and **Member** fields of ISP_DP_ATTR_S.

Section 6.12 "Black Level" is added.

Chapter 8 AF

This chapter is added.

Issue 00B08 (2013-04-02)

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

Chapter 3 AE

In section 3.5.1, the descriptions in the **Note** fields of ISP_AE_MODE_E and ISP_AE_ATTR_S are updated.

Chapter 6 IMP

In section 6.9, the descriptions of the anti-fog functions are updated, members are added to ISP_ANTIFOG_S, and the descriptions of ISP_ANTIFOG_S are updated.

Issue 00B07 (2013-02-28)

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

Chapter 2 System Control

In section 2.2, [Table 2-1](#) is added to the **Note** field of HI_MPI_ISP_SetModuleControl.

In section 2.3, ISP_ANTIFLICKER_MODE_E is added, and the **enMode** member is added to the **Member** field of ISP_ANTIFLICKER_S.

Chapter 3 AE

In section 3.5.2, the descriptions in the **Note** field of ISP_OP_TYPE_E are updated.

Chapter 4 AWB

In section 4.5.1, the descriptions in the **Difference** field of ISP_AWB_ATTR_S are updated.

Chapter 5 CCM

In section 5.4, HI_MPI_ISP_SetSaturation and HI_MPI_ISP_GetSaturation are added, and the descriptions in the **Note** field of HI_MPI_ISP_SetSaturationAttr are updated.

In section 5.5, the **au8Sat[8]** member and [Table 5-1](#) are added to the **Member** field of ISP_SATURATION_ATTR_S, and the descriptions in the **Note** field of ISP_SATURATION_ATTR_S are updated. In addition, the value ranges of **u16HighColorTemp**, **u16MidColorTemp**, and **u16LowColorTemp** of ISP_COLORMATRIX_S are changed.

Chapter 6 IMP

Section 6.11 "Demosaic" is added.



Issue 00B06 (2013-02-05)

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

Chapter 5 CCM

In section 5.4, descriptions are modified in the **Syntax** and **Parameter** fields of HI_MPI_ISP_SetSaturationAttr and HI_MPI_ISP_GetSaturationAttr. Descriptions are added to the **Note** field of HI_MPI_ISP_SetSaturationAttr.

In section 5.5, ISP_SATURATION_ATTR_S is added.

Chapter 6 IMP

In section 6.4.3, the HI_ERR_ISP_ILLEGAL_PARAM error code is added to HI_MPI_ISP_SetShadingTable. Descriptions for the Hi3518 configurations are added to the **Note** field.

In section 6.4.4, the members **u16ShadingOffCenter_R**, **u16ShadingOffCenter_G**, **u16ShadingOffCenter_B**, and **u16ShadingTableNodeNumber** are added to the **Member** field of ISP_SHADINGTAB_S for the Hi3518.

Issue 00B05 (2012-12-26)

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

Chapter 3 AE

In section 3.5.2, the descriptions are modified in the **Member** and **Note** fields of ISP_OP_TYPE_E.

Chapter 4 AWB

In section 4.4.1, the description "Configure the upper and lower color temperature limits for the AWB algorithm" is added in the **Note** field of HI_MPI_ISP_SetAWBAttr.

In section 4.5, the members **u8HighColorTemp** and **HI_U8 u8LowColorTemp** are added in the **Syntax** and **Member** fields of ISP_AWB_ATTR_S. The description "Configure the upper and low color temperature limits for the AWB algorithm" is added to the **Note** field.

Chapter 6 IMP

In section 6.1.3, **HI_U8 u8SharpenAltD[8]** and **HI_U8 u8SharpenAltUd[8]**, and their descriptions are added to ISP_SHARPEN_ATTR_S. Descriptions and the minimum value are changed for **u32StrengthTarget**. The default value for **u8StrengthMin** is changed. The reference content is changed in the **Note** field for the relationship between the sharpen strength and the sensor gain.

In section 6.7.2, descriptions of the denoise strength are updated.

In section 6.7.4, **HI_U8 u8SnrThresh[8]** and its descriptions are added to ISP_DENOISE_ATTR_S.

Chapter 7 Debug

This is a new chapter.

Issue 00B04 (2012-11-25)

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

Chapter 2 System Control



In section 2.2, HI_MPI_ISP_SetAntiFlickerAttr and HI_MPI_ISP_GetAntiFlickerAttr are added.

In section 2.3, ISP_ANTIFLICKER_S is added.

Chapter 3 AE

In section 3.5, the value range is added to the **Member** field of ISP_EXP_STA_INFO_S.

Chapter 4 AWB

In section 4.5, values ranges are added to the **Member** field of ISP_MWB_ATTR_S. The value ranges of **u16CbMin** and **u16CrMin** are changed in the **Member** field of ISP_WB_STA_INFO_S.

Chapter 5 CCM

In section 5.4, the descriptions of three CCMs are added to the **Note** field of HI_MPI_ISP_SetCCM.

In section 5.5, the descriptions are updated in the **Syntax**, **Member**, and **Note** fields of ISP_COLORMATRIX_S.

Chapter 6 IMP

In section 6.1.3, the relationship between the sharpen strength and the system gain is added to the **Note** field of ISP_SHARPEN_ATTR_S.

In section 6.2.3, value ranges are added to the **Member** field of ISP_GAMMA_TABLE_S for the Hi3518. The values for the Hi3518 are changed in the **Difference** field of ISP_GAMMA_TABLE_S. The descriptions are updated in the **Syntax** and **Member** fields of ISP_GAMMA_CURVE_E.

In section 6.7.2, the relationship between the denoise strength and the system gain is added.

In section 6.7.4, the descriptions are updated in the **Note** field of ISP_DENOISE_ATTR_S.

Issue 00B03 (2012-10-30)

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

Chapter 2 System Control

In section 2.2, the MPIs HI_MPI_ISP_SetSlowFrameRate and HI_MPI_ISP_GetSlowFrameRate are added.

Chapter 3 AE

In section 3.4.1, the MPIs HI_MPI_ISP_SetExpStaInfo and HI_MPI_ISP_GetExpStaInfo are added.

In section 3.5.1, ISP_EXP_STA_INFO_S is added. The value ranges of the **u16ExpTimeMax**, **u16ExpTimeMin**, **u16DgainMax**, **u16DgainMin**, **u16AgainMax**, and **u16AgainMin** parameters are changed in the **Member** field of ISP_AE_ATTR_S.

In section 3.5.2, ISP_OP_TYPE_E is added.

Chapter 4 AWB

In section 4.5, the value range of the **u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]** parameter is changed



in the **Member** field of ISP_AWB_ATTR_S. The value ranges of the parameters are changed in the **Member** field of ISP_WB_ZONE_STA_INFO_S.

Chapter 5 CCM

In section 5.4, the description of the **pu8Value** parameter is updated in the **Parameter** field of HI_MPI_ISP_GetSaturationAttr. The value range of the **u16CorrMatrix[9]** parameter is changed in the **Member** field of ISP_COLORMATRIX_S.

Chapter 6 IMP

In section 6.1.3, a note is added to the **Note** field of ISP_SHARPEN_ATTR_S.

In section 6.5.3, the descriptions are updated in the **Example** fields of HI_MPI_ISP_SetDefectPixelAttr and HI_MPI_ISP_GetDefectPixelAttr.

In section 6.10.2, the descriptions are updated.

Chapter 7 Error Codes

In [Table 9-1](#), four error codes are added.

Issue 00B02 (2012-09-20)

This issue is first draft release.



Contents

About This Document.....	i
1 ISP	1
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	3
1.2.6 Software Process	4
2 System Control	6
2.1 Function Description	6
2.2 API Reference	6
2.3 Data Structures	21
3 AE	28
3.1 Overview	28
3.2 Important Concepts	28
3.3 Function Description	29
3.4 API Reference	31
3.4.1 AE Control MPIS	31
3.4.2 AI Control MPIS	46
3.5 Data Structures	50
3.5.1 AE	50
3.5.2 AI	61
4 AWB	65
4.1 Overview	65
4.2 Important Concepts	65
4.3 Function Description	65
4.4 API Reference	67
4.4.1 AWB Control MPIS	67
4.4.2 WB Statistics MPIS	79
4.5 Data Structures	84



5 CCM	95
5.1 Overview	95
5.2 Important Concepts	95
5.3 Function Description	95
5.4 API Reference	96
5.5 Data Structures	101
6 IMP	104
6.1 Sharpen	104
6.1.1 Function Description	104
6.1.2 API Reference	104
6.1.3 Data Structure	106
6.2 Gamma	108
6.2.1 Function Description	108
6.2.2 API Reference	108
6.2.3 Data Structures	112
6.3 DRC	115
6.3.1 Function Description	115
6.3.2 API Reference	115
6.3.3 Data Structure	117
6.4 Lens Shading Correction	119
6.4.1 Overview	119
6.4.2 Function Description	119
6.4.3 API Reference	120
6.4.4 Data Structures	124
6.5 Defect Pixel	129
6.5.1 Overview	129
6.5.2 Function Description	129
6.5.3 API Reference	130
6.5.4 Data Structures	133
6.6 Crosstalk Removal	135
6.6.1 Overview	135
6.6.2 Function Description	135
6.6.3 API Reference	136
6.6.4 Data Structure	138
6.7 Denoise	139
6.7.1 Overview	139
6.7.2 Function Description	139
6.7.3 API Reference	139
6.7.4 Data Structure	141
6.8 DIS	143
6.8.1 Overview	143



6.8.2 Function Description.....	143
6.8.3 API Reference	143
6.8.4 Data Structures.....	148
6.9 Anti-fog	150
6.9.1 Function Description.....	150
6.9.2 API Reference	150
6.9.3 Data Structure	152
6.10 Anti-False Color	153
6.10.1 Overview.....	153
6.10.2 Function Description.....	153
6.10.3 API Reference	153
6.10.4 Data Structure	155
6.11 Demosaic.....	156
6.11.1 Function Description.....	156
6.11.2 API Reference	156
6.11.3 Data Structure.....	158
6.12 Black Level	161
6.12.1 Overview.....	161
6.12.2 API Reference	161
6.12.3 Data Structures.....	163
7 ISP Debugging Information	164
7.1 Overview	164
7.2 Function Description.....	164
7.3 API Reference	164
7.4 Data Structure.....	167
8 AF	170
8.1 Overview	170
8.2 Function Description.....	170
8.3 API Reference	170
8.4 Data Structures	172
9 Error Codes	174
10 Proc Debugging Information.....	175
10.1 Overview	175
10.2 ISP	175



Figures

Figure 1-1 Operating mode of the ISP.....	1
Figure 1-2 Design methodology of the ISP firmware.....	2
Figure 1-3 File structure of the ISP firmware.....	3
Figure 1-4 Internal process of the ISP firmware.....	4
Figure 1-5 Flowchart of the ISP firmware.....	5
Figure 3-1 Workflow of the AE module	28
Figure 3-2 5-segment AE statistics histogram	29
Figure 3-3 256-segment AE statistics histogram	30
Figure 3-4 AE working principle.....	30
Figure 4-1 Schematic diagram of the AWB module.....	66
Figure 5-1 CCM	95
Figure 6-1 Crosstalk removal threshold	135
Figure 6-2 Offset schematic diagram of the DIS module.....	143
Figure 6-3 DIS horizontal offset.....	149
Figure 6-4 DIS vertical offset.....	150



Tables

Table 2-1 Bits of u32ModFlag.....	15
Table 5-1 Mapping between the values of au8Sat[8] and gains.....	102
Table 6-1 Values of u8SharpenAltD[8] based on the sensor gain.....	107
Table 6-2 Values of u8SharpenAltUd based on the sensor gain	107
Table 6-3 Mesh_Scale parameter.....	119
Table 6-4 Mapping between the values of u8Strength and gain	138
Table 6-5 Values of u8SnrThresh[8] based on the sensor gain	142
Table 6-6 Mapping between the values of u8LumThresh and gain	160
Table 6-7 Mapping between the values of u8NpOffset and gain.....	160
Table 9-1 Error codes for ISP APIs.....	174

1 ISP

1.1 Overview

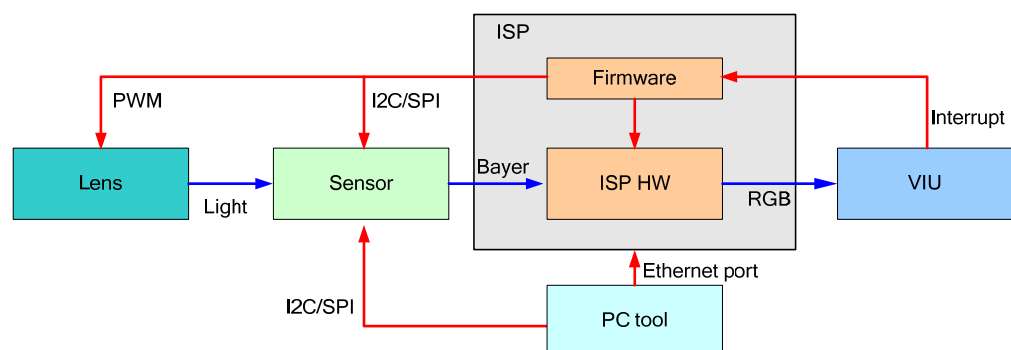
The image signal processor (ISP) processes digital images by using a series of digital image processing algorithms. The ISP supports the following functions: 3A algorithm, defect pixel correction, denoise, highlight compensation, backlight compensation, color enhancement, and lens shading correction.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



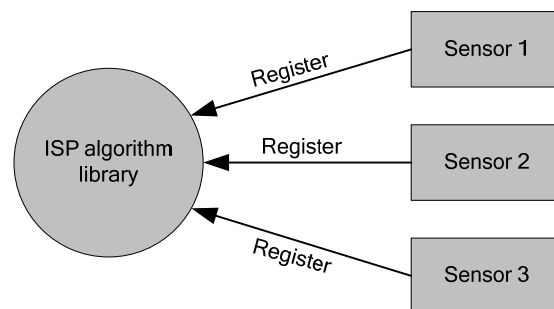


For details about related concepts and functions, see the *Hi3516 Full-HD IP Camera SoC Data Sheet* and *Hi3518 720p IP Camera SoC Data Sheet*.

1.2.2 Design Methodology

The ISP firmware consists of the algorithm control unit and sensor control unit. The firmware is designed based on the methodology of separating the algorithm control unit from the sensor control unit. This ensures that only the sensor control unit needs to be modified when a new sensor needs to be configured.

Figure 1-2 Design methodology of the ISP firmware

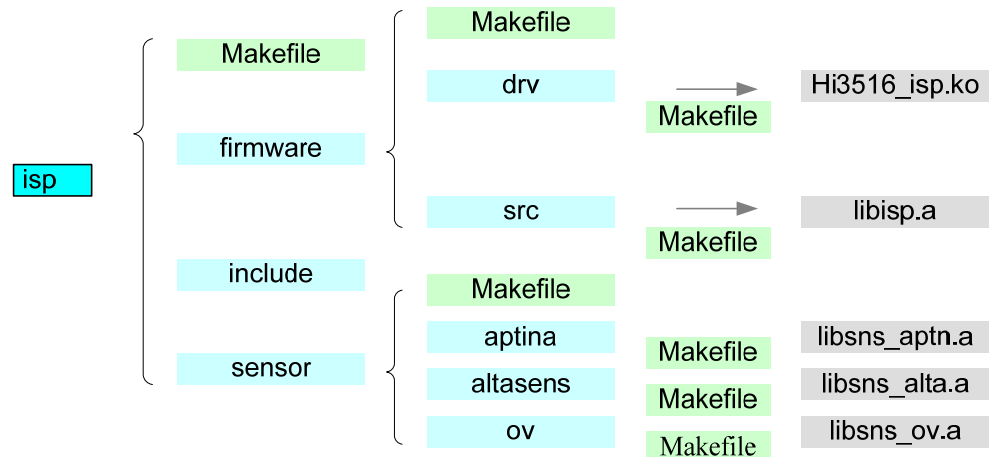


Sensors register to the ISP algorithm library for control functions. These functions are callback function. By calling these callback functions, the firmware controls sensors to implement the functions including exposure time adjustment, analog gain, digital gain, lens step focus, and iris rotation.

1.2.3 File Structure

[Figure 1-3](#) shows the file structure of the ISP firmware. The files of the firmware and sensor are stored in tow folders. The driver in the **drv** folder of the firmware is used to report the ISP interrupt. The running of the firmware depends on the ISP interrupt. The **src** folder stores the key algorithms of the firmware. After the source code is compiled, the basic algorithm library **libisp.a** is generated. The **sensor** folder stores the drivers of all sensors as open source code. The sensor driver code is compiled as a library to facilitate application compilation and connection between the ISP firmware and sensor. 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 provided SDK complies with the common development mode, and the algorithm library is retained. This enables you to use different sensors. The **sensor** folder includes the following two key files:

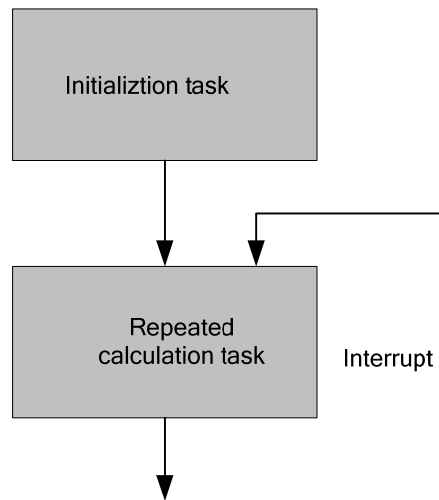
- **sensor_cmos.c**
This file is used to implement the callback functions required by the ISP. The callback functions include the adaptation algorithms of sensors and vary according to sensors.
- **sensor_ctrl.c**
This file is the bottom-layer driver of the sensor, and is used to read, write to, and initialize sensors. You can develop these two files based on the data sheet of sensors and seek for help from the sensor vendor.

[Note]

The senior engineers that are familiar with the ISP logic and algorithm development can develop the algorithm library based on ISP registers.

1.2.5 Internal Process

The internal process of the firmware consists of the initialization process and dynamic adjustment process. During the initialization process, the entire ISP is initialized. The sensor is initialized by calling related callback functions each time an interrupt is generated. During the dynamic adjustment process, the firmware performs real-time calculation, and controls the parameters of the ISP such as the exposure time and gain of the sensor.

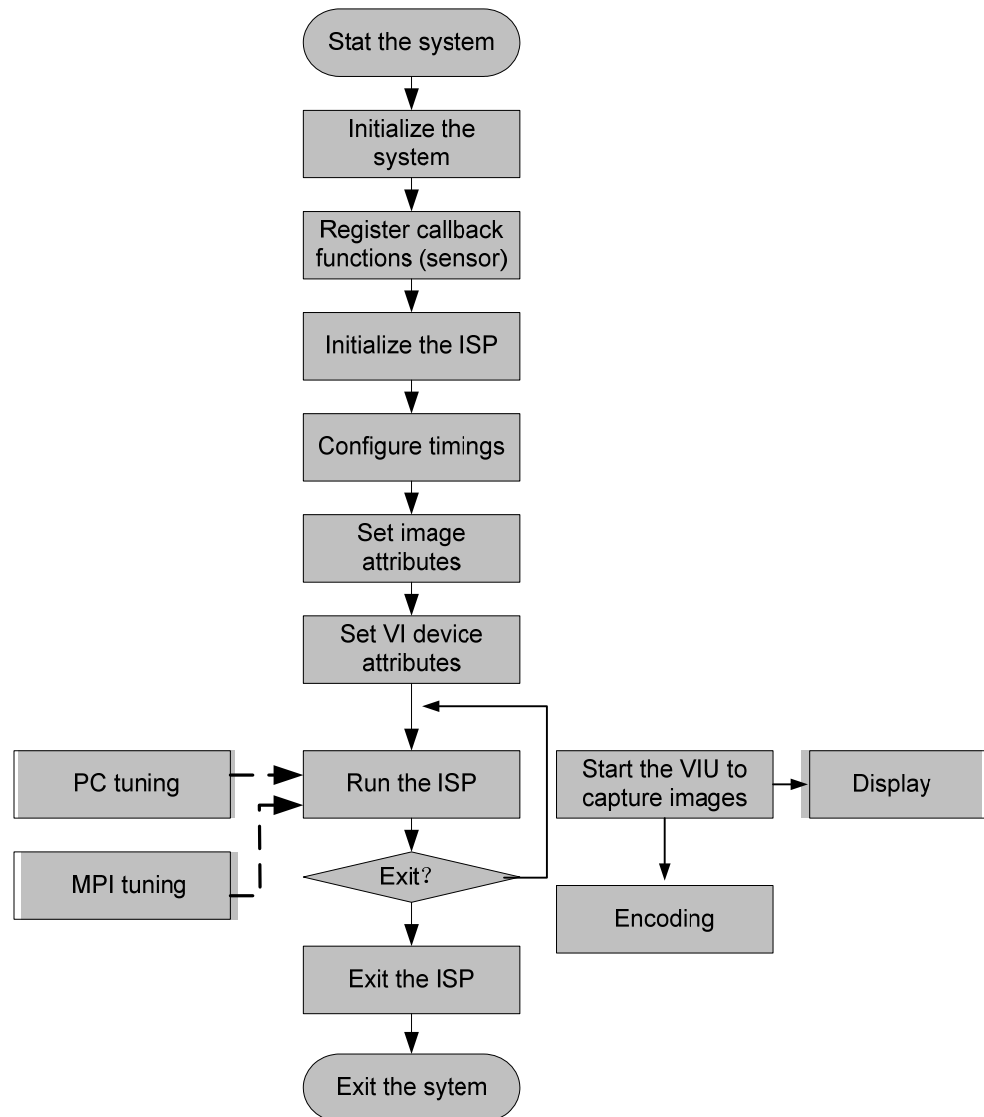
Figure 1-4 Internal process of the ISP firmware

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 1-5](#) 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 1-5 Flowchart of the ISP firmware



After adjusting images, you can save setting by using the configuration file of the PC tuning tool. This enables you to load configured parameters when starting the ISP 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_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
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]

- 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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_SNS_UNREGISTER	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
HI_ERR_ISP_SNS_UNREGISTER	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]

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_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]

Error Code	Description
HI_SUCCESS	Success.

[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.
[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
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]

[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
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 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
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]

[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
HI_ERR_ISP_ILLEGAL_PARAM	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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.
- **u32MilliSec** 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

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_S](#): Defines the anti-flicker attribute of the ISP.
- [ISP_VD_INFO_S](#): Defines ISP frame information.

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.
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.



Member	Description
	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 illuminant 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 illuminant 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 bEnable is HI_TRUE , anti-flicker is enabled. If bEnable is HI_FALSE , 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]

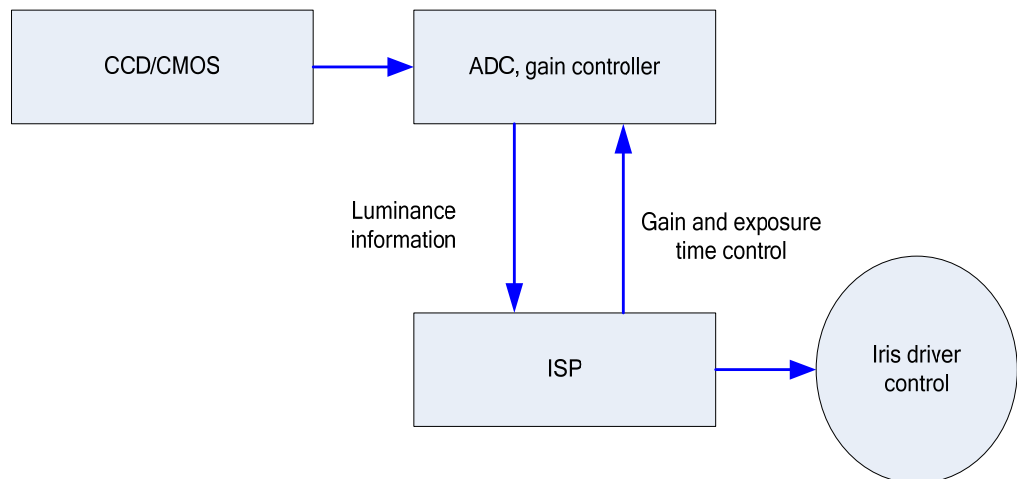
None

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

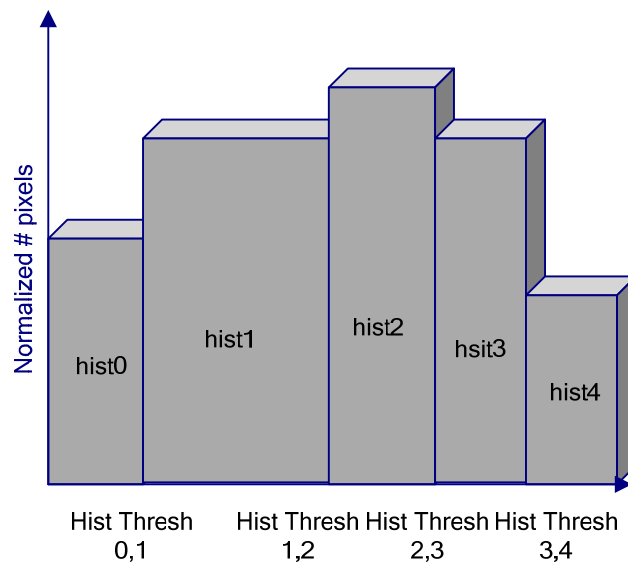
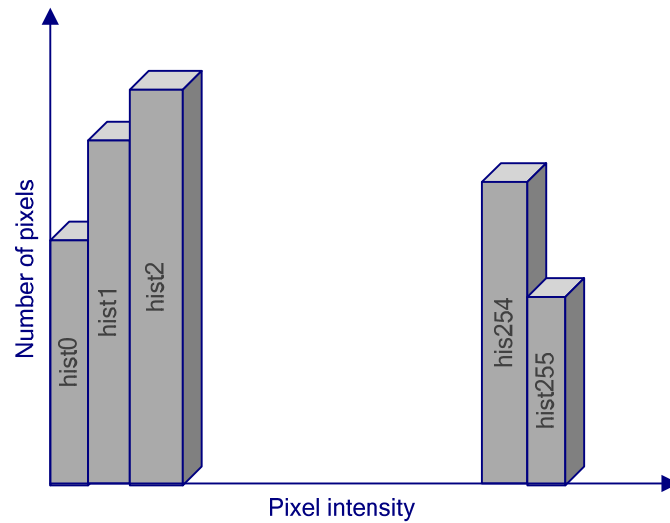
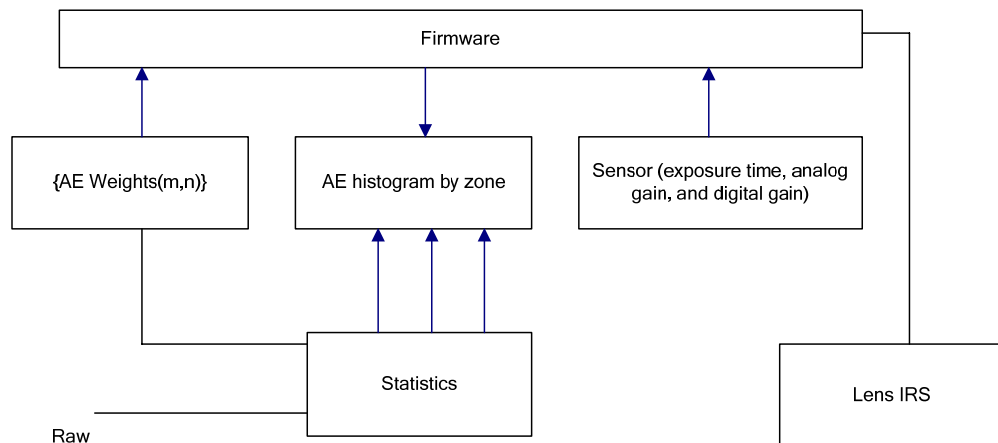


Figure 3-3 256-segment AE statistics histogram



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 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_SetMEAttr](#): Sets manual exposure (ME) attributes.
- [HI_MPI_ISP_GetMEAttr](#): Obtains 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
HI_ERR_ISP_ILLEGAL_PARAM	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
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]

[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
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
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]

- 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.ul6ExpTimeMax = 4000;
    stAEAttr.ul6ExpTimeMin = 2;
    stAEAttr.ul6AGainMax   = 28;
    stAEAttr.ul6AGainMin   = 0;
    stAEAttr.ul6DGainMax   = 38;
    stAEAttr.ul6DGainMin   = 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
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]



HI_MPI_ISP_GetExposureType

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.
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]

- [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
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]

[HI_MPI_ISP_GetExposureType](#)

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
HI_ERR_ISP_NULL_PTR	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_GetAEAttr](#)

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 parameters	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]

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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
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]

[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.
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

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
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

3.4.2 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
enIrisType	Iris control type.	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 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]

None

[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
HI_ERR_ISP_NULL_PTR	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
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]

[HI_MPI_ISP_SetAIAAttr](#)

3.5 Data Structures

3.5.1 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_ME_ATTR_S](#): Defines the ME attributes of the ISP.

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_DP_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. 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. u16SystemGainMax is used with u16SystemGainShift .
u16SystemGainShift	Offset 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.

[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. 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 precision). The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32DGainMin	Minimum digital gain for AE (10-bit decimal precision). The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32AGainMax	Maximum analog gain for AE (10-bit decimal precision). The value range is [0x400, 0xFFFFFFFF], which depends on the sensor.
u32AGainMin	Minimum analog gain for AE (10-bit decimal precision). 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.



Member	Description
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. 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 ME attributes of the ISP.

[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.
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]
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_INNER_STATE_INFO_S
{
    HI_U32 u32ExposureTime;
    HI_U32 u32AnalogGain;
    HI_U32 u32DigitalGain;
    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, 0xFFFFFFFF]
u32AnalogGain	Sensor analog gain. Value range: [0x0, 0xFFFFFFFF]
u32DigitalGain	Sensor digital gain. Value range: [0x0, 0xFFFFFFFF]
u32Exposure	Sensor exposure. Value range: [0x0, 0xFFFFFFFF]
u16AE_Hist256Value[256]	256-segment histogram. Value range: [0x0, 0xFFFFFFFF]
u16AE_Hist5Value[5]	5-segment histogram after region weighting.
u8AveLum	Average picture luminance. Value range: [0x0, 0xFF]
bExposureIsMAX	0: The ISP does not reach the maximum exposure performance. 1: The ISP reaches the maximum exposure performance.

[Note]

- 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 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_INNER_STATE_INFO_EX_S

[Description]

Defines the internal status information about the ISP.

[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].



Member	Description
u8AveLum	Average picture luminance. Value range: [0x00, 0xFF].
bExposureIsMAX	0: The ISP does not reach the maximum exposure performance. 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

3.5.2 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, 0xFFFF]
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 illuminants. This ensures that the captured color information is converted into the information without color cast under an ideal illuminant.

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 an illuminant 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, θ 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.

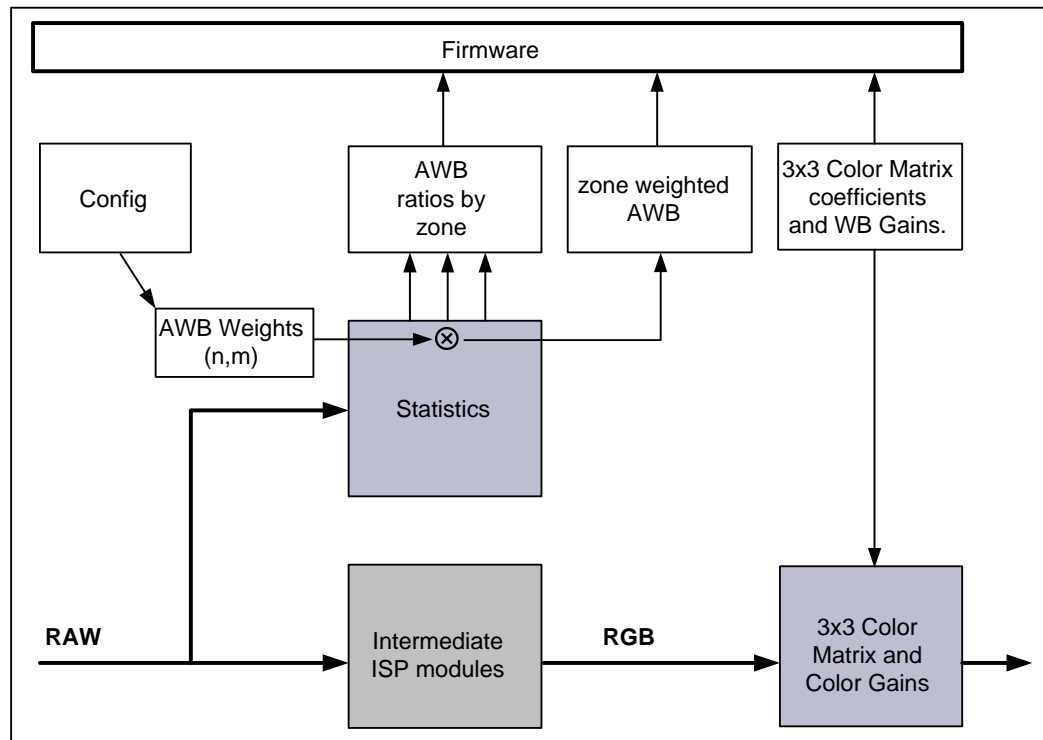
The statistics provided by the AWB module mainly include R/G and B/G values. The R/G and B/G values are average values. The average values are calculated based on the total R/G and B/G values of white points in each zone and AWB weight coefficient by using the following formula:

$$AWB_RG = \frac{\sum_{mn} w_{mn} * awb_rg_{mn}}{\sum_{mn} w_{mn} * awb_{mn}}$$

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 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
HI_ERR_ISP_ILLEGAL_PARAM	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
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]

[HI_MPI_ISP_GetWBType](#)

HI_MPI_ISP_SetAWBAttr

[Description]

Sets automatic AWB attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAWBAttr(const ISP_AWB_ADD_LIGHTSOURCE_S: Defines the  
attributes of separate illuminant points.  
*pstAWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstAWBAttr	Automatic AWB attributes.	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.

[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
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]

[HI_MPI_ISP_GetWBType](#)

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
HI_ERR_ISP_NULL_PTR	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_GetMWBAttr( ISP\_MWB\_ATTR\_S*pstMWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
pstMWBAttr	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
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]

[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
HI_ERR_ISP_ILLEGAL_PARAM	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_SetAdvAWBAAttr** to achieve the optimal effect.
- The Hi3516 does not support this MPI.

[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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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
HI_ERR_ISP_NULL_PTR	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](#)

4.4.2 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
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]

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
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

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.
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]

[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\_ADD\_LIGHTSOURCE\_S: Defines the attributes of separate illuminant points.  
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_GetLightSource: Obtains the attributes of separate illuminant
points.
(OP_TYPE_MANUAL);

//Set WB gain to 1
stMWBAttr.u16Rgain = 0x100;
stMWBAttr.u16Bgain = 0x100;
stMWBAttr.u16Ggain = 0x100;
HI_MPI_ISP_SetMWBAttr(&stMWBAttr);

//define white point range
stWBSTAInfo.u16BlackLevel = 0x40;
stWBSTAInfo.u16WhiteLevel = 0x3a0;
stWBSTAInfo.u16CrMax = 0x200;
stWBSTAInfo.u16CrMin = 0x80;
stWBSTAInfo.u16CbMax = 0x200;
stWBSTAInfo.u16CbMin = 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.u16Rgain = stWBSTAInfo.u16GRgain;
    stMWBAttr.u16Bgain = stWBSTAInfo.u16GBgain;
    stMWBAttr.u16Ggain = 0x100;

    //Set gain to WB registers if the statistics is reasonable
    if (stWBSTAInfo.u32GSum > 0x1000)
    {
        HI_MPI_ISP_SetMWBAttr(&stMWBAttr);
    }

    //Sleep 1 frame
    usleep(40000);
}
```

[See Also]

[ISP_WB_ZONE_STA_INFO_S](#)



4.5 Data Structures

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 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 value range is [0x1, 0xFFFF], and the default value is 0x40. A larger value indicates a faster convergence speed.
u8RGStrength	Red-green (RG) strength for AWB.
u8BGStrength	Blue-green (BG) strength for AWB.



Member	Description
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	<pre>#define WEIGHT_ZONE_ROW 7 #define WEIGHT_ZONE_COLUMN 9</pre> <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 u8ZoneSel is 0 or greater than or equal to 63, the global AWB algorithm is selected. If u8ZoneSel ranges from 1 to 62, the zoned AWB algorithm is selected. The statistical results of only first u8ZoneSel sorted zones involve AWB correction. The specific value is not affected.</p>
Hi3518	<pre>#define WEIGHT_ZONE_ROW 15 #define WEIGHT_ZONE_COLUMN 17</pre> <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 u8ZoneSel is 0 or 255, the global AWB algorithm is selected. If u8ZoneSel ranges from 1 to 254, the zoned AWB algorithm is selected. The statistical results of only first u8ZoneSel 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 wb_para[6] member of cmos_isp_default_t in the cmos.c file.
au16StaticWB[4]	Static white balance calibration. Value range: [0x0, 0xFFFF] The default value is defined by the gain_offset [4] member of cmos_isp_default_t in the cmos.c file.
u16RefTemp	Reference AWB color temperature. Value range: [0x0, 0xFFFF] The default value is defined by the wb_ref_temp member of cmos_isp_default_t in the cmos.c 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. 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]



Member	Description
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]



Member	Description
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, 0xFFFF]
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. 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.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this data structure.
Hi3518	The Hi3518 supports this data structure.

[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.
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.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this data structure.
Hi3518	The Hi3518 supports this data structure.

[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 Static WB option of the calibration tool.
u16WhiteBgain	Bgain that is obtained by calibrating the ColorChecker Raw data captured in an illuminant using the WB option of the calibration tool.
u16ExpQuant	Illuminant luminance (not supported currently).
bLightStatus	Illuminant point status. 0: disabled 1: enabled

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this data structure.
Hi3518	The Hi3518 supports this data structure.

[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.

[Difference]

Chip	Description
Hi3516	The Hi3516 does not support this data structure.
Hi3518	The Hi3518 supports this data structure.

[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
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]

[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
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

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
HI_ERR_ISP_NULL_PTR	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
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]

[HI_MPI_ISP_SetCCM](#)

5.5 Data Structures

- [ISP_SATURATION_ATTR_S](#): Defines the ISP color saturation attribute.
- [ISP_COLOMATRIX_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
{
    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 HI_FALSE .
u8SatTarget	Expected saturation. Value range: [0x00, 0xFF]. The default value is 0x80 .
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 Table 5-1 .

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

au8Sat	Again*Dgain*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
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]

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
HI_ERR_ISP_NULL_PTR	The pointer is null.



Error Code	Description
HI_ERR_ISP_ILLEGAL_PARAM	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](#): Defines the attribute for ISP sharpen.

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.



Member	Description
	HI_FALSE: disabled 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 Table 6-1 .
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 Table 6-2 .

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	Again*Dgian*IspDgain (times)
u8SharpenAltUd [3]	8
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_GetGammaAttr](#): Obtains 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
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]

[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
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]

[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
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]

- 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">• [0, 0xFFFF] for the Hi3516• [0, 0xFFF] for the Hi3518
u16Gamma_FE[GAMMA_FE_LUT_SIZE]	Gamma_fe table of the wide dynamic range (WDR) sensor. Value range: <ul style="list-style-type: none">• [0, 0xFFFF] for the Hi3516• [0, 0xFFF] for the Hi3518

[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
pstDRCAttr	DRC attributes.	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.

[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
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]

[HI_MPI_ISP_SetDRCAttr](#)

6.3.3 Data Structure

[ISP_DRC_ATTR_S](#): Defines DRC attributes.

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.



Member	Description
bDRCManualEnable	Manual DRC enable. HI_FALSE: disabled 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 iridix_sm member of cmos_isp_default_t in the cmos.c 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 u32Whitelevel are not involved in the DRC calculation. The default value is defined by the iridix_wl member of cmos_isp_default_t in the cmos.c 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 u32Blacklevel 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
HI_ERR_ISP_ILLEGAL_PARAM	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_SetShadingAttr](#)

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
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]

[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
HI_ERR_ISP_NULL_PTR	The pointer is null.
HI_ERR_ISP_ILLEGAL_PARAM	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
HI_ERR_ISP_NULL_PTR	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

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](#)



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 center 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 center 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 center 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 0x81 .

[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]

None

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

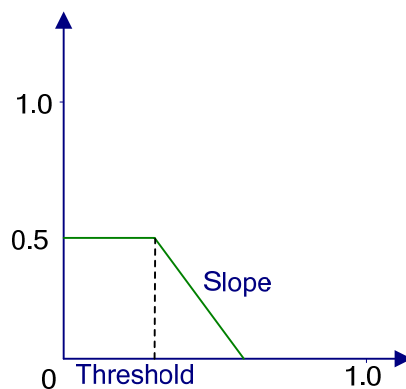
6.6.1 Overview

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
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]

[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
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]

[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 Table 6-4 .

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
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]

[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
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]

[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 Table 6-5 .



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

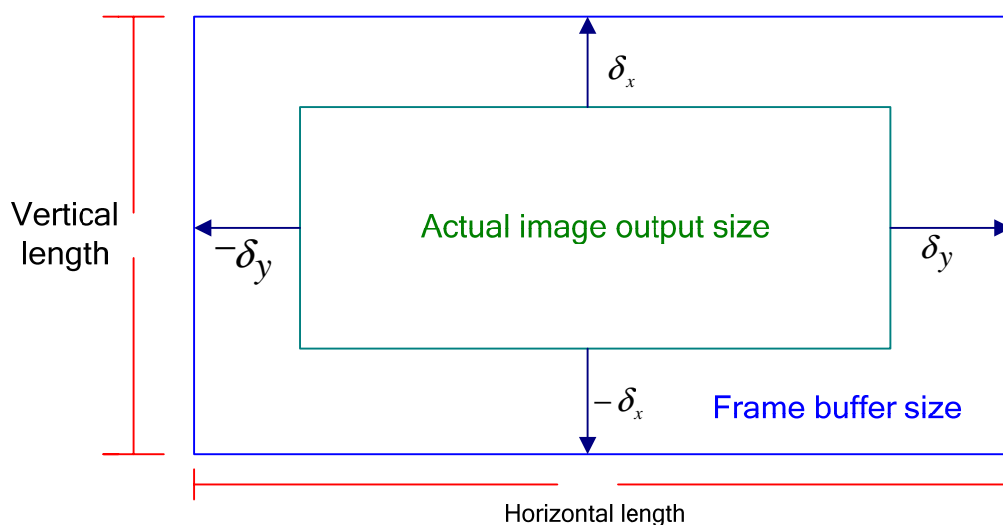
6.8.1 Overview

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 δ_x and vertical offset δ_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 δ_x and δ_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
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]

See [HI_MPI_ISP_GetDISInfo](#).

[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
<code>pstDISAttr</code>	DIS 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]

[HI_MPI_ISP_SetDISAttr](#)

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
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]

```
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;
```



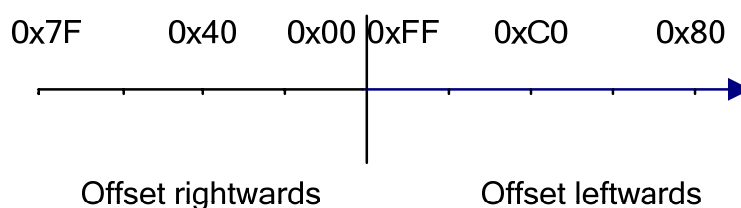
[Member]

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]

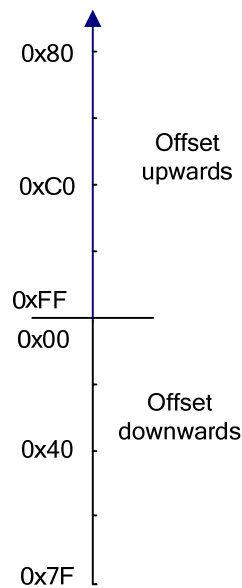
s8Xoffset is a signed number. When an image is offset leftwards, the output range is [0x80, 0xFF]. The value is expressed by ones-complement code. The value 0xFF indicates that the image is offset leftwards by one pixel, and the value 0xFE indicates that the image is offset leftwards by two pixels. When 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
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]

[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
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]

[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
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

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
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

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
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



6.11.3 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 vh_slope member of cmos_isp_demosaic_t in the cmos.c 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 aa_slope member of cmos_isp_demosaic_t in the cmos.c 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 va_slope member of cmos_isp_demosaic_t in the cmos.c 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 uu_slope member of cmos_isp_demosaic_t in the cmos.c 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 vh_thresh member of cmos_isp_demosaic_t in the cmos.c 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 aa_thresh member of cmos_isp_demosaic_t in the cmos.c 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 va_thresh member of cmos_isp_demosaic_t in the cmos.c 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 uu_thresh member of cmos_isp_demosaic_t in the cmos.c 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 Table 6-6 .
u8NpOffset	Offset of the demosaic noise profile. The eight values of this array correspond to different ISO values. For details, see Table 6-7 .



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 illuminant. 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
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

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
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

6.12.3 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 black_level[4] member of cmos_isp_demosaic_t in the cmos.c file.

[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
HI_ERR_ISP_NULL_PTR	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
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

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
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]

[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
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]

[HI_MPI_ISP_SetFocusStaInfo](#)

8.4 Data Structures

ISP_FOCUS_STA_INFO_S

[Description]

Defines AF statistics of the ISP.

[Syntax]

```
typedef struct hiISP_FOUCS_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 3
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 IPS 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