HiISP

Development Reference

Issue 06

Date 2016-10-28

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.

NOTE

Unless otherwise specified, this document applies to the Hi3516A and Hi3516D.

Related Versions

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

Product Name	Version
Hi3516A	V100
Hi3516D	V100

Intended Audience

This document is intended for:

- Technical support engineers
- 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.

Symbol	Description
WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
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 06 (2016-10-28)

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

Chapter 2 System Control

In section 2.2, HI_MPI_ISP_GetVDTimeOut is modified, and HI_MPI_ISP_SetDCFInfo and HI_MPI_ISP_GetDCFInfo are added.

In section 2.3, ISP_DCF_INFO_S is added.

Chapter 3 AE

In section 3.5.1, AE_FSWDR_ATTR_S is added and AE_SENSOR_DEFAULT_S is modified.

In section 3.5.2, ISP_AE_ROUTE_NODE_S and ISP_AE_ROUTE_EX_NODE_S are modified.

In section 3.5.3, ISP_PIRIS_ATTR_S and ISP_IRIS_ATTR_S are modified.

Chapter 4 AWB

In section 4.5.2, ISP_AWB_ATTR_EX_S and ISP_AWB_IN_OUT_ATTR_S are modified.

Chapter 6 IMP

In section 6.5.2, figure 6-3 and figure 6-4 are added.

In section 6.13.2, the description in the **Note** field of HI_MPI_ISP_FPNCalibrate is modified.

Chapter 10 Error Codes

Chapter 10, table 10-1 is modified.

Appendix "Acronyms and Abbreviations" is added.

Issue 05 (2016-01-29)

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

Chapter 2 System Control

In section 2.2, HI_MPI_ISP_SetModParam and HI_MPI_ISP_GetModParam are added. The content related to Huawei LiteOS is added.

In section 2.3, ISP_AE_RESULT_S is modified and ISP_MOD_PARAM_S is added.

Chapter 3 AE

In section 3.5.1, AE_SENSOR_EXP_FUNC_S, AE_SENSOR_DEFAULT_S, ISP_AE_ATTR_S, and ISP_EXP_INFO_S are modified.

In section 3.5.2, the description in the **Note** field of ISP_SUBFLICKER_S is modified. ISP_FSWDR_MODE_E is added. The descriptions in the **Syntax**, **Member**, and **Note** fields of ISP_AE_ATTR_S are modified.

In section 3.5.3, the description in the **Note** field of ISP_IRIS_ATTR_S is modified.

Chapter 4 AWB

In section 4.4.1, the descriptions in the **Requirement** fields of HI_MPI_AWB_UnRegister, HI_MPI_AWB_SensorRegCallBack, and HI_MPI_AWB_SensorUnRegCallBack are modified.

In section 4.4.2, the descriptions in the **Requirement** fields of the MPIs from HI_MPI_ISP_SetWBAttr to HI_MPI_ISP_GetAWBAttrEx are modified.

In section 4.4.3, HI_MPI_ISP_CalGainByTemp is added.

In section 4.5.2, ISP_AWB_MULTI_LS_TYPE_E is added. ISP_AWB_ATTR_EX_S and ISP_WB_INFO_S are modified. Figure 4-4 is added.

Chapter 6 IMP

In section 6.7.4, ISP_NR_AUTO_ATTR_S and ISP_NP_TABLE_S are modified.

In section 6.9.3, the description in the **Member** field of ISP_DEFOG_ATTR_S is modified.

Chapter 11 Proc Debugging Information

Section 11.1 and section 11.2 are modified.

Issue 04 (2015-10-31)

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

Chapter 2 System Control

In section 2.2, HI_MPI_ISP_QueryInnerStateInfo is added.

In section 2.3, the descriptions in the **Syntax** and **Member** fields of ISP_AE_RESULT_S are modified. The description in the **Note** field of ISP_AWB_EXP_FUNC_S is modified. The description in the **Member** field of ISP_PUB_ATTR_S is modified. ISP_INNER_STATE_INFO_S is added.

Chapter 3 AE

In section 3.5.1, the descriptions in the **Syntax** and **Member** fields of AE_SENSOR_DEFAULT_S are modified.

In section 3.5.2, the description in the **Note** field of ISP_AE_MODE_E is modified. The descriptions in the **Syntax** and **Member** fields of ISP_WDR_EXPOSURE_ATTR_S are modified.

In section 3.5.3, the descriptions in the **Note** fields of ISP_IRIS_TYPE_E, ISP_MI_ATTR_S, and ISP_IRIS_ATTR_S are modified.

Chapter 4 AWB

In section 4.5.2, ISP_AWB_ATTR_S and ISP_AWB_CT_LIMIT_ATTR_S are modified.

Chapter 5 CCM

In section 5.5, table 5-1 is modified.

Chapter 7 Statistics

In section 7.3, ISP_FOCUS_STATISTICS_S is added.

Chapter 11 Proc Debugging Information

Section 11.2 is modified.

Issue 03 (2015-06-15)

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

Chapter 2 System Control

In section 2.3, the descriptions of ISP_AE_RESULT_S and ISP_AWB_RESULT_S are updated.

Chapter 3 AE

In section 3.4.2, the descriptions of HI_MPI_ISP_SetAERouteAttr and HI_MPI_ISP_QueryExposureInfo are updated.

In section 3.4.3, the description of HI_MPI_ISP_SetIrisAttr is updated. HI_MPI_ISP_SetDcirisAttr, HI_MPI_ISP_GetDcirisAttr, HI_MPI_ISP_SetPirisAttr, and HI_MPI_ISP_GetPirisAttr are added.

In section 3.5.2, the descriptions of ISP_EXPOSURE_ATTR_S, ISP_AE_ROUTE_NODE_S, ISP_AE_ROUTE_S, and ISP_EXP_INFO_S are updated.

In section 3.5.3, ISP_DCIRIS_ATTR_S, ISP_PIRIS_ATTR_S, table 3-1, and table 3-2 are added.

Chapter 4 AWB

In section 4.5.2, ISP_AWB_CBCR_TRACK_ATTR_S, ISP_AWB_LUM_HISTGRAM_ATTR_S, and table 4-1 to table 4-4 are added.

Chapter 6 IMP

In section 6.1.3, ISP_RGBSHARPEN_ATTR_S is added.

In section 6.7.3, HI_MPI_ISP_SetNPTable and HI_MPI_ISP_GetNPTable are added.

In section 6.7.4, ISP_NP_TABLE_S is added.

In section 6.14.3, the descriptions in the **Note** fields of ISP_ACM_ATTR_S and ISP_ACM_LUT_S are updated.

Chapter 11 Proc Debugging Information

The description in section 11.2 is updated.

Issue 02 (2015-02-10)

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

Chapter 2 System Control

In section 2.2, the description in the **Note** field of HI_MPI_ISP_SetPubAttr is updated.

In section 2.3, the descriptions of ISP_AE_INFO_S, ISP_SNS_REGS_INFO_S and ISP_AE_RESULT_S are updated.

Chapter 3 AE

In section 3.5.2, the descriptions of ISP_EXPOSURE_ATTR_S and ISP_WDR_EXPOSURE_ATTR_S are updated. ISP_SUBFLICKER_S is added.

In section 3.5.3, the descriptions of ISP_IRIS_STATUS_E, ISP_IRIS_ATTR_S, and ISP_IRIS_TYPE_E are updated.

Chapter 4 AWB

In section 4.5.2, the description of ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S is updated.

Chapter 5 CCM

In section 5.4, HI_MPI_ISP_SetColorToneAttr and HI_MPI_ISP_GetColorToneAttr are deleted.

In section 5.5, ISP_COLOR_TONE_ATTR_S is deleted.

Chapter 6 IMP

Section 6.15 is added.

Chapter 7 Statistics

In section 7.3, the descriptions of ISP_AF_CFG_S and ISP_AF_SQU_MODE_E are updated.

Chapter 11 Proc Debugging Information

The description in section 11.2 is updated.

Issue 01 (2014-12-20)

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

The contents related to the Hi3516D are added.

Chapter 2 System Control

In section 2.3, ISP_CTRL_PROC_WRITE_S and ISP_CTRL_CMD_E are added.

Chapter 3 AE

In section 3.4.2, HI_MPI_ISP_SetWDRExposureAttr and HI_MPI_ISP_GetWDRExposureAttr are added.

In section 3.4.3, HI_MPI_ISP_SetAICalibrate and HI_MPI_ISP_GetAICalibrate are deleted.

In section 3.5.2, ISP_WDR_EXPOSURE_ATTR_S is deleted.

In section 3.5.3, ISP_IRIS_TYPE_E and ISP_IRIS_F_NO_E are added. ISP_AI_ATTR_S, ISP_STATUS_E, and ISP_AI_CALIBRATE_S are deleted.

Chapter 4 AWB

In section 4.5.2, figure 4-3 is added, and the descriptions of ISP_AWB_ATTR_EX_S and ISP_AWB_IN_OUT_ATTR_S are updated.

Chapter 6 IMP

In section 6.3.3, the description of ISP_DRC_ATTR_S is updated.

In section 6.15.1, the function description is updated.

Chapter 7 Statistics

In section 7.3, ISP_FOCUS_STATISTICS_S is deleted.

Chapter 8 Default Cmos Parameter Configuration

In section 8.1, the procedure for parsing the .ini file is added.

Issue 00B05 (2014-11-10)

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

Chapter 2 System Control

In section 2.2, HI_MPI_ISP_SetRegister and HI_MPI_ISP_GetRegister are added. The descriptions in the **Note** fields of HI_MPI_ISP_SetPubAttr and HI_MPI_ISP_SetWDRMode are updated.

In section 2.3, ISP_CMOS_SENSOR_IMAGE_MOD is added. ISP_CMOS_WDR_ATTR_S, ISP_CMOS_SENSOR_MAX_RESOLUTION, and ISP_CMOS_SENSOR_IMAGE_MODE are deleted. The descriptions in the **Member** fields of RECT_S, ISP_SENSOR_EXP_FUNC_S, ISP_CMOS_DEFAULT_S, and ISP_SNS_REGS_INFO_S are updated.

Chapter 5 CCM

In section 5.4, the description in the **Note** field of HI_MPI_ISP_SetCCMAttr is updated.

Chapter 6 IMP

In section 6.3.3, the description in the **Member** field of ISP_DRC_ATTR_S is updated. Figure 6-1 and figure 6-2 are added.

In section 6.13.3, the descriptions of ISP_FPN_ATTR_S, ISP_FPN_MANUAL_ATTR_S, and ISP_FPN_AUTO_ATTR_S are updated.

In section 6.15.1, figure 6-8 and figure 6-9 are added.

Chapter 10 Error Codes

Table 10-1 is updated.

Issue 00B04 (2014-10-19)

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

Chapter 2 System Control

In section 2.3, the description in the **Member** field of ISP_AE_RESULT_S is updated.

Chapter 3 AE

In section 3.4.2, the description in the **Note** field of HI_MPI_ISP_SetAERouteAttr is updated.

In section 3.5.2, the description in the **Note** field of ISP_AE_ATTR_S is updated.

Chapter 4 AWB

In section 4.5.2, the description of ISP_AWB_ATTR_S and ISP_WB_ATTR_S is updated.

Chapter 6 IMP

In section 6.13.3, the description of ISP_FPN_CALIBRATE_ATTR_S is updated.

In section 6.15.3, the description in the **Note** field of ISP_WDR_FS_ATTR_S is updated.

Chapter 7 Statistics

In section 7.3, the descriptions in the **Member** and **Note** fields of ISP_AE_STATISTICS_CFG_S are updated.

Chapter 8 Default Cmos Parameter Configuration

Chapter 8 "Default Cmos Parameter Configuration" is added.

Issue 00B03 (2014-09-25)

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

Chapter 1 ISP

In section 1.2, "File Structure" is deleted.

Chapter 2 System Control

In section 2.3, figures 2-5 to 2-8 are added.

Chapter 3 AE

In section 3.3, figure 3-2 and figure 3-4 are modified.

The description in sections 3.4.1, 3.4.2, 3.5.2, and 3.5.3 is updated.

Chapter 4 AWB

In section 4.4.1, HI_MPI_ISP_AWBLibRegCallBack is deleted.

In section 4.5.2, figure 4-3 is modified.

Chapter 5 CCM

The description in section 5.4 is updated.

Chapter 6 IMP

The description in section 6.1 is updated.

In section 6.5.3, the descriptions of HI_MPI_ISP_ SetDPCalibrate and HI_MPI_ISP_ GetDPCalibrate are updated.

In section 6.5.4, figure 6-1 is added.

In section 6.6.4, the description of ISP_CR_ATTR_S is updated.

In section 6.9.3, the description of ISP_DEFOG_AUTO_ATTR_S is updated.

In section 6.13.3, the description of ISP_FPN_CALIBRATE_ATTR_S is updated.

In section 6.15.3, the descriptions in the **Member** fields of ISP_GAMMAFE_ATTR_S and ISP_WDR_FS_ATTR_S are updated.

Chapter 7 Statistics

In section 7.3, the descriptions in the **Member** and **Note** fields of ISP_AE_STATISTICS_CFG_S are updated. The descriptions in the **Member** fields of ISP_WB_STATISTICS_CFG_PARA_S, ISP_WB_BAYER_STATISTICS_INFO_S, and ISP_FOCUS_ZONE_S are updated.

Chapter 10 Proc Debugging Information

In section 10.2, AE INFO parameters are modified.

Issue 00B02 (2014-09-14)

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

Chapter 1 ISP

The description in section 1.2.2 is updated.

Chapter 2 System Control

The description in section 2.2 is updated.

In section 2.3, ISP_AWB_PARAM_S and ISP_AE_PARAM_S are modified.

Chapter 3 AE

In section 3.4.2, HI_MPI_ISP_SetExposureAttr is modified.

In section 3.5.2, ISP_AE_MODE_E and ISP_AE_ATTR_S are modified.

Chapter 4 AWB

In section 4.5.2, ISP_AWB_ALG_TYPE_E, ISP_AWB_CT_LIMIT_ATTR_S, and ISP_AWB_IN_OUT_ATTR_S are modified.

Chapter 6 IMP

In section 6.5.3, HI_MPI_ISP_SetDPCalibrate is modified.

In section 6.5.4, ISP_DP_STATIC_CALIBRATE_S is modified.

Chapter 7 Statistics

In section 7.3, ISP_AF_CFG_S, ISP_AF_PEAK_MODE_E, ISP_AF_SQU_MODE_E, ISP_AF_H_PARAM_S, ISP_AF_FV_PARAM_S, and FOCUS_STATISTICS_CFG_S are added.

Chapter 9

This chapter is deleted.

Issue 00B01 (2014-07-25)

This issue is the first draft release.

Contents

About This Document i

1 ISP 1

1.1 Overview 1

1.2 Function Description 1

1.2.1 Architecture 2

1.2.2 Development Mode 2

1.2.3 Internal Process 3

1.2.4 Software Process 4

2 System Control 6

2.1 Function Description 6

2.2 API Reference 6

2.3 Data Structures 40

3 AE 91

3.1 Overview 91

3.2 Important Concepts 91

3.3 Function Description 92

3.4 API Reference 93

3.4.1 AE Algorithm Library MPIs 93

3.4.2 AE Control MPIs 99

3.4.3 AI Control MPIs 113

3.5 Data Structures 121

3.5.1 Registration 121

3.5.2 AE 130

3.5.3 AI 156

4 AWB 170

4.1 Overview 170

4.2 Important Concepts 170

4.3 Function Description 170

4.4 API Reference 172

4.4.1 AWB Algorithm Library MPIs 172

4.4.2 AWB Control MPIs 176

4.4.3 WB Consult MPI 180
4.5 Data Structures 183
4.5.1 Registration 183
4.5.2 WB 185
5 CCM 208
5.1 Overview 208
5.2 Important Concepts 208
5.3 Function Description 208
5.4 API Reference 209
5.5 Data Structures 213
6 IMP 219
6.1 Sharpen 219
6.1.1 Function Description 219
6.1.2 API Reference 219
6.1.3 Data Structures 221
6.2 Gamma 227
6.2.1 Function Description 227
6.2.2 API Reference 227
6.2.3 Data Structures 229
6.3 DRC 230
6.3.1 Function Description 230
6.3.2 API Reference 230
6.3.3 Data Structure 233
6.4 Lens Shading Correction 237
6.4.1 Overview 237
6.4.2 Function Description 238
6.4.3 API Reference 238
6.4.4 Data Structures 240
6.5 Defect Pixel 242
6.5.1 Overview 242
6.5.2 Function Description 242
6.5.3 API Reference 246
6.5.4 Data Structures 250
6.6 Crosstalk Removal 256
6.6.1 Overview 256
6.6.2 Function Description 256
6.6.3 API Reference 257
6.6.4 Data Structure 259
6.7 Noise Reduction 261
6.7.1 Overview 261
6.7.2 Function Description 261

6.7.3 API Reference 261
6.7.4 Data Structures 265
6.8 DIS 269
6.8.1 Overview 269
6.8.2 Function Description 269
6.8.3 API Reference 270
6.8.4 Data Structure 274
6.9 Anti-Fog 274
6.9.1 Function Description 274
6.9.2 API Reference 274
6.9.3 Data Structures 276
6.10 Anti-False Color 278
6.10.1 Overview 278
6.10.2 Function Description 279
6.10.3 API Reference 279
6.10.4 Data Structure 281
6.11 Demosaic 281
6.11.1 Function Description 281
6.11.2 API Reference 281
6.11.3 Data Structures 283
6.12 Black Level 288
6.12.1 Overview 288
6.12.2 API Reference 288
6.12.3 Data Structures 290
6.13 FPN Removal 291
6.13.1 Overview 291
6.13.2 API Reference 292
6.13.3 Data Structures 296
6.14 ACM 300
6.14.1 Overview 300
6.14.2 API Reference 300
6.14.3 Data Structures 304
6.15 ColorTone 307
6.15.1 Overview 307
6.15.2 API Reference 308
6.15.3 Data Structures 309
6.16 WDR 310
6.16.1 Overview 310
6.16.2 API Reference 315
6.16.3 Data Structures 319
6.17 Obtaining ISP Virtual Addresses 321
6.17.1 Function Description 321

6.17.2 API Reference 321
6.17.3 Data Structure 322
7 Statistics 324
7.1 Overview 324
7.2 API Reference 324
7.3 Data Structures 327
8 Default Cmos Parameter Configuration 347
8.1 Overview 347
8.2 Cmos Structure 348
8.3 INI File Usage Description 348
8.3.1 AE 348
8.3.2 AWB 349
8.3.3 ISP 349
8.4 Precautions 352
9 Debug 354
9.1 Overview 354
9.2 Function Description 354
9.3 API Reference 354
9.4 Data Structure 357
10 Error Codes 358
11 Proc Debugging Information 359
11.1 Overview 359
11.2 ISP 360
A Acronyms and Abbreviations 368

Figures

Figure 1-1 Operating mode of the ISP 2
Figure 1-2 Architecture of the ISP firmware 2
Figure 1-3 Internal process of the ISP firmware 3
Figure 1-4 Architecture of the ISP firmware 4
Figure 1-5 Flowchart of the ISP firmware 5
Figure 2-1 Interfaces between the ISP library and the sensor library 24
Figure 2-2 Interfaces between the ISP library and the AE algorithm library 26
Figure 2-3 Interfaces between the ISP library and the AWB algorithm library 29
Figure 2-4 Interfaces between the ISP library and the AF algorithm library 31
Figure 2-5 Impact on the sharpen curve exerted by the Core when Mag is 8 and Strength is 127 56
Figure 2-6 Impact on the sharpen curve exerted by the Mag when Core is 255 and Strength is 127 57
Figure 2-7 Impact on the sharpen curve exerted by the Strength when Core is 255 and Mag is 8 57
Figure 2-8 Description of the register with the address of 0x205a06c0 68
Figure 2-9 Parameters related to white point area selection 80
Figure 3-1 Workflow of the AE module 91
Figure 3-2 5-segment AE statistics histogram 92
Figure 3-3 256-segment AE statistics histogram 93
Figure 3-4 AE working principle 93
Figure 3-5 Interfaces between the AE algorithm library and the sensor library 97
Figure 3-6 AE allocation routes 107
Figure 4-1 Schematic diagram of the AWB module 171
Figure 4-2 Interface between the AWB algorithm library and the sensor library 175
Figure 4-3 Parameters of color temperature curves 199
Figure 4-4 Configuration example of the color temperature weight in the multi-illuminant scenario (*n* indicates the number of segmentation points for the color temperature) 200
Figure 4-5 Parameters for outdoor color temperature range 205
Figure 5-1 CCM 208

Figure 6-1 Impact on the DRC tone curve exerted by the Asymmetry when BrightEnhance is 255 236
Figure 6-2 Impact on the DRC tone curve exerted by the BrightEnhance when Asymmetry is 20 237
Figure 6-3 Correctable static defect pixels 243
Figure 6-4 Defect pixel cluster that cannot be completely corrected 244
Figure 6-5 Dynamic defect pixel correction 252
Figure 6-6 Crosstalk removal threshold 257
Figure 6-7 Offset schematic diagram of the DIS module 270
Figure 6-8 FPN calibration 292
Figure 6-9 FPN correction 292
Figure 6-10 Handling process in sensor built-in WDR mode 312
Figure 6-11 Handling process in multi-frame combination WDR mode 314
Figure 7-1 Square mode 338
Figure 8-1 Cmos structure 348

Tables

Table 3-1 Mapping table for the position of the P iris stepper motor and the iris F value (taking the Forecam NV03105P as an example) 165

Table 3-2 Parameters related to the P iris (taking the Forecam NV03105P as an example) 166

Table 4-1 Mapping between the values of au16CrMax[16] and gains (only for reference) 189

Table 4-2 Mapping between the values of au16CrMin[16] and gains (only for reference) 190

Table 4-3 Mapping between the values of au16CbMax[16] and gains (only for reference) 191

Table 4-4 Mapping between the values of au16CbMin[16] and gains (only for reference) 191

Table 5-1 Mapping between the values of au8Sat[16] and gains (using the MN34220 sensor as an example). 215

Table 6-1 Mapping between the configured values of u8SharpenD[ISP_AUTO_STENGTH_NUM] and gains 223

Table 6-2 Mapping between the configured values of u8SharpenUd[ISP_AUTO_STENGTH_NUM] and gains 223

Table 6-3 Mapping between the configured values of u8SharpenRGB[ISP_AUTO_STENGTH_NUM] and gains 224

Table 6-4 Mapping between the values of u8Strength[ISP_AUTO_STENGTH_NUM] and gains 260

Table 6-5 Mapping between the values of au8Thresh [ISP_AUTO_STENGTH_NUM] and gains 267

Table 6-6 Mapping between the values of au8LumThresh[ISP_AUTO_STENGTH_NUM] and gains 286

Table 6-7 Mapping between the values of au8NpOffset[ISP_AUTO_STENGTH_NUM] and gains 286

Table 8-1 Parameters in the [AE] field 348

Table 8-2 Parameters in the [AWB] field 349

Table 8-3 Parameters in the [ISP] field 350

Table 10-1 Error codes for ISP APIs 358

1 ISP

1.1 Overview

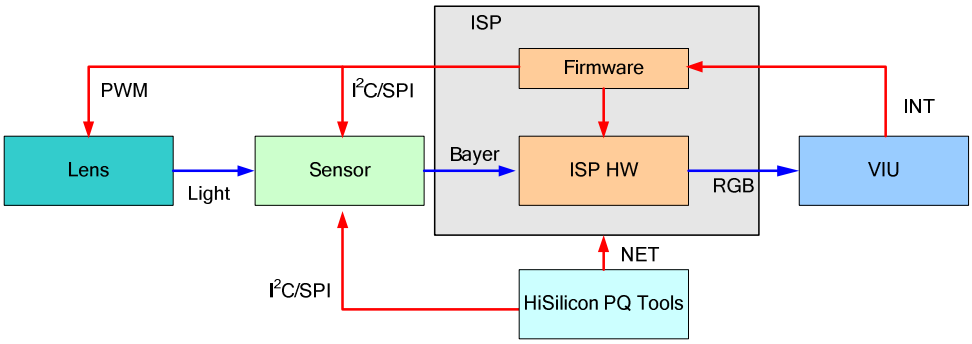
The image signal processor (ISP) processes digital images by using a series of digital image processing algorithms. The ISP supports the following functions: 3A algorithm, defect pixel correction, noise reduction (NR), highlight compensation, backlight compensation, color enhancement, and lens shading 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 back-end 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 HiSilicon PQ Tools 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

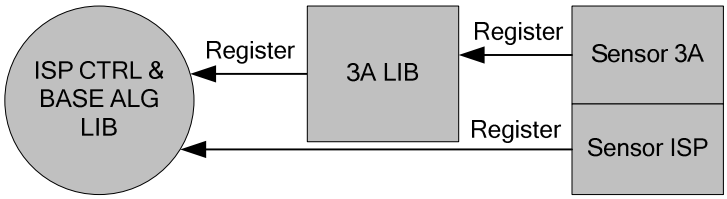


For details about related concepts and functions, see the *Hi3516A/Hi3516D HD IP Camera SoC Data Sheet*.

1.2.1 Architecture

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

Figure 1-2 Architecture of the ISP firmware



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

1.2.2 Development Mode

The SDK supports various development modes:

- You can use only the HiSilicon 3A algorithm libraries. You need to call the sensor adaptation interfaces provided in the ISP basic algorithm library and HiSilicon 3A algorithm libraries to adapt to various sensors. Each sensor corresponds to a folder and the sensor folder includes the following two key files:
  - **sensor_cmos.c**

This file is used to implement the callback functions required by the ISP. The callback functions include the adaptation algorithms of sensors and vary according to sensors.

- **sensor_ctrl.c**

  This file is the underlying driver of the sensor, and is used to read, write to, and initialize sensors. You can develop these two files based on the data sheet of sensors and seek for help from the sensor vendor.

- You can develop your own 3A algorithm libraries based on the 3A algorithm registration interfaces provided in the HiSilicon ISP library. You need to call the sensor adaptation interfaces provided in the ISP basic algorithm unit and your own 3A algorithm libraries to adapt to various sensors.
- You can use both the HiSilicon 3A algorithm libraries and your own 3A algorithm libraries. For example, you can use **lib_hiae.a** to implement AE, and use your own 3A algorithm libraries to implement AWB.

NOTE

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

1.2.3 Internal Process

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

Figure 1-3 Internal process of the ISP firmware

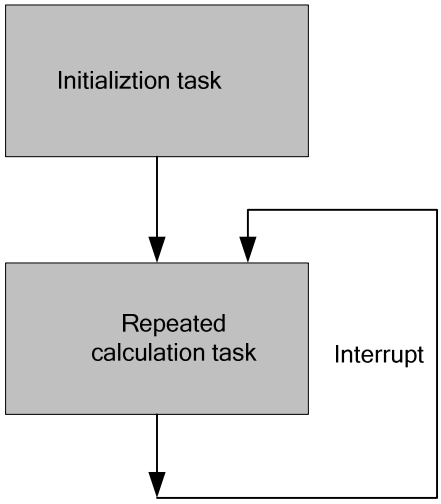
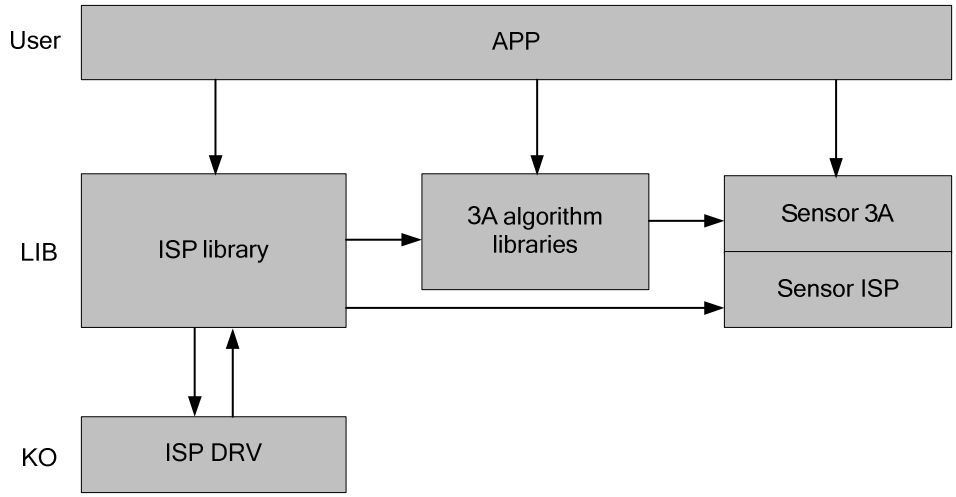


Figure 1-4 shows the architecture of the ISP firmware.

Figure 1-4 Architecture of the ISP firmware

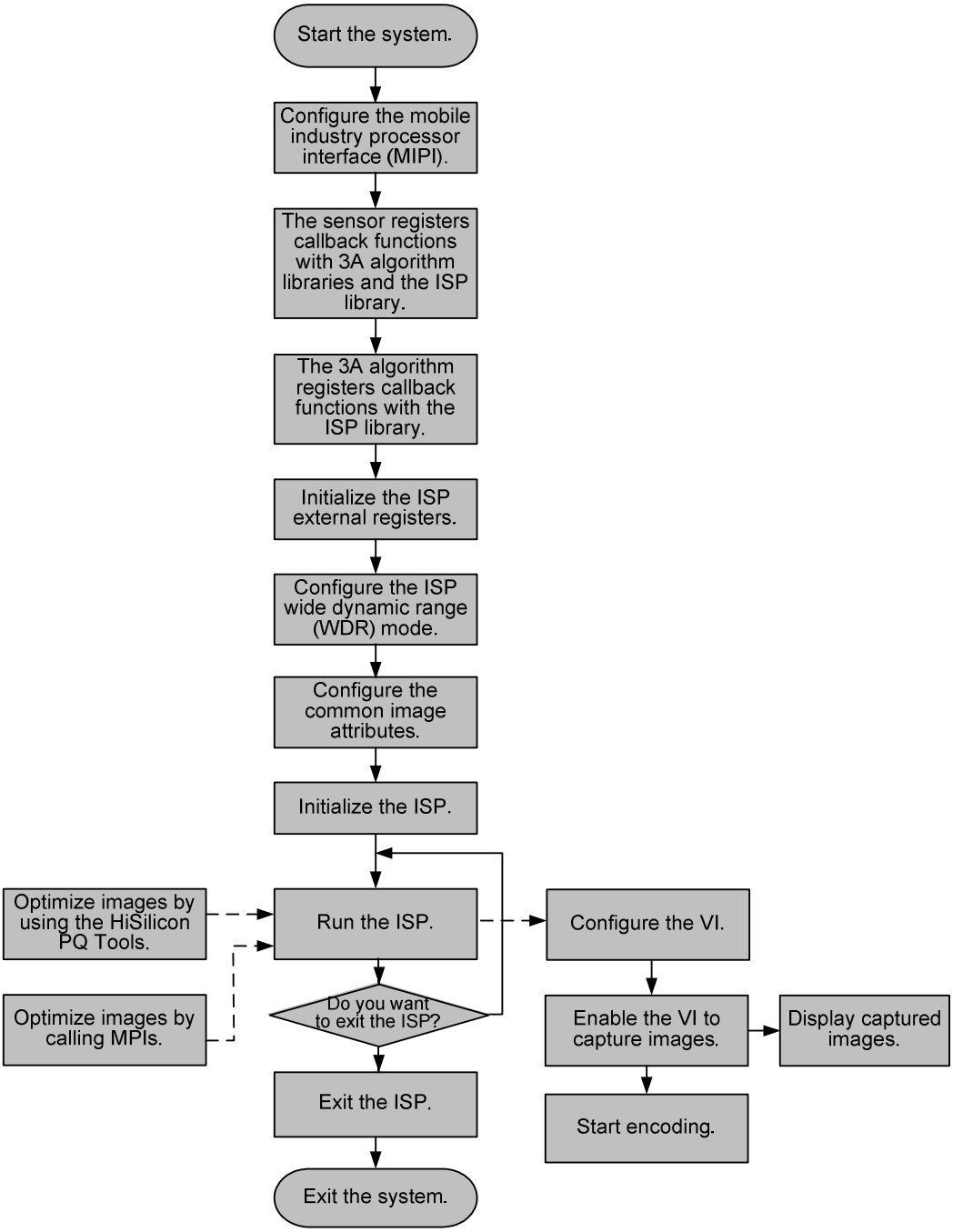


1.2.4 Software Process

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 transmits them for displaying or encoding. Then the images are transmitted for displaying or encoding. Figure 1-5 shows the software process.

The HiSilicon PQ Tools dynamically adjusts the image quality at the PC end by tuning the values of related parameters such as NR strength, color conversion matrix, and saturation.

Figure 1-5 Flowchart of the ISP firmware



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

2 System Control

2.1 Function Description

The system control part provides the following functions:

- Configures common ISP attributes.
- Initializes the ISP firmware.
- Runs the ISP firmware.
- Exits the ISP firmware.
- Configures ISP modules.

2.2 API Reference

Unless otherwise specified, the MPIs in this document support multiple processes.

- HI_MPI_ISP_MemInit: Initializes the ISP external registers.
- HI_MPI_ISP_Init: Initializes the ISP firmware.
- HI_MPI_ISP_Run: Runs the ISP firmware.
- HI_MPI_ISP_Exit: Exits the ISP firmware.
- HI_MPI_ISP_SetPubAttr: Sets the ISP public attributes.
- HI_MPI_ISP_GetPubAttr: Obtains the ISP public attributes.
- HI_MPI_ISP_SetFMWState: Sets the ISP firmware status.
- HI_MPI_ISP_GetFMWState: Obtains the ISP firmware status.
- HI_MPI_ISP_SetWDRMode: Sets the wide dynamic range (WDR) mode of the ISP.
- HI_MPI_ISP_GetWDRMode: Obtains the WDR mode of the ISP.
- HI_MPI_ISP_SetModuleControl: Controls the modules of the ISP.
- HI_MPI_ISP_GetModuleControl: Obtains the control status of the modules of the ISP.
- HI_MPI_ISP_SetRegister: Sets the register value.
- HI_MPI_ISP_GetRegister: Obtains the register value.
- HI_MPI_ISP_GetVDTimeOut: Obtains ISP interrupt information.
- HI_MPI_ISP_SensorRegCallBack: Registers a sensor.
- HI_MPI_ISP_SensorUnRegCallBack: Deregisters a sensor.

- HI_MPI_ISP_AELibRegCallBack: Registers the AE algorithm library.
- HI_MPI_ISP_AELibUnRegCallBack: Deregisters the AE algorithm library.
- HI_MPI_ISP_AWBLibRegCallBack: Registers the AWB algorithm library.
- HI_MPI_ISP_AWBLibUnRegCallBack: Deregisters the AWB algorithm library.
- HI_MPI_ISP_AFLibRegCallBack: Registers the AF algorithm library.
- HI_MPI_ISP_AFLibUnRegCallBack: Deregisters the AF algorithm library.
- HI_MPI_ISP_SetBindAttr: Binds the ISP library to 3A algorithm libraries/sensors.
- HI_MPI_ISP_GetBindAttr: Obtains the binding relationship between the ISP library and 3A algorithm libraries/sensors.
- HI_MPI_ISP_QueryInnerStateInfo: Obtains the ISP status information.
- HI_MPI_ISP_SetDCFInfo: Sets DCF parameters.
- HI_MPI_ISP_GetDCFInfo: Obtains DCF parameters.
- HI_MPI_ISP_SetModParam: Sets the ISP module parameters.
- HI_MPI_ISP_GetModParam: Obtains the ISP module parameters.

HI_MPI_ISP_MemInit

[Description]

Initializes the ISP external registers.

[Syntax]

```
HI_S32 HI_MPI_ISP_MemInit(ISP_DEV IspDev);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_MEM_NOT_INIT	The external registers are not initialized.
HI_ERR_ISP_SNS_UNREGISTER	The sensor is not registered.
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 initializing the ISP external registers, ensure that .ko drivers are loaded and related callback functions are registered by the sensor.
- This MPI must be called before HI_MPI_ISP_SetWDRMode and HI_MPI_ISP_SetPubAttr are called to set the wide dynamic range (WDR) mode and common image attributes respectively.
- This MPI does not support multiple processes and it must be called with sensor_register_callback, HI_MPI_AE_Register, HI_MPI_AWB_Register, HI_MPI_AF_Register, HI_MPI_ISP_Init, HI_MPI_ISP_Run, and HI_MPI_ISP_Exit in the same process.
- This MPI cannot be called repeatedly.
- You are advised to call this MPI to initialize the ISP external registers again after calling HI_MPI_ISP_Exit.
- Huawei LiteOS does not support the kernel module loading mechanism. To achieve the effect similar to loading the .ko drivers in Linux, run **sdk_init.c** under **Huawei LiteOS release/ko**.

[Example]

None

[See Also]

HI_MPI_ISP_Exit

HI_MPI_ISP_Init

[Description]

Initializes the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_Init(ISP_DEV IspDev);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_MEM_NOT_INIT	The external registers are not initialized.
HI_ERR_ISP_NOT_INIT	The ISP is not initialized.
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 ISP firmware, ensure that .ko drivers are loaded and related callback functions are registered by the sensor with the ISP library.
- Before initializing the ISP firmware, ensure that HI_MPI_ISP_MemInit is called to initialize the ISP external registers.
- Before initializing the ISP firmware, ensure that HI_MPI_ISP_SetWDRMode and HI_MPI_ISP_SetPubAttr are called to set the WDR mode and common image attributes respectively.
- This MPI does not support multiple processes and it must be called with sensor_register_callback, HI_MPI_AE_Register, HI_MPI_AWB_Register, HI_MPI_AF_Register, HI_MPI_ISP_MemInit, HI_MPI_ISP_Run, and HI_MPI_ISP_Exit in the same process.
- This MPI cannot be called repeatedly.
- You are advised to call this MPI to initialize the ISP firmware again after calling HI_MPI_ISP_Exit.
- If the JPEG DCF function is enabled, HI_MPI_VB_SetSupplementConf must be called before this API is called (For details, see section 2.2 "Functions" in the *HiMPP IPC V2.0 Media Processing Software Development Reference*.) to set stSupplementConf to VB_SUPPLEMENT_JPEG_MASK.
- Huawei LiteOS does not support the kernel module loading mechanism. To achieve the effect similar to loading the .ko drivers in Linux, run **sdk_init.c** under **Huawei LiteOS release/ko**.

[Example]

None

[See Also]

HI_MPI_ISP_Exit

HI_MPI_ISP_Run

[Description]

Runs the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_Run(ISP_DEV IspDev);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_SNS_UNREGISTER	The sensor is not registered.
HI_ERR_ISP_MEM_NOT_INIT	The external registers are not initialized.
HI_ERR_ISP_NOT_INIT	The ISP is not initialized.

[Requirement]

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

[Note]

- Before running the firmware, ensure that the sensor is initialized and related callback functions are registered with the ISP library.
- Before running the firmware, ensure that HI_MPI_ISP_Init is called to initialize the ISP.
- This MPI does not support multiple processes and it must be called with sensor_register_callback, HI_MPI_AE_Register, HI_MPI_AWB_Register, HI_MPI_AF_Register, HI_MPI_ISP_MemInit, HI_MPI_ISP_Init, and HI_MPI_ISP_Exit in the same process.
- This MPI is a block interface. You are advised to call it using a real-time thread.
- When the .ko drivers of ISP are loaded, the interrupt processing mechanism can be configured by using the module parameter **int_bottomhalf**. The default value is 0. When

int_bottomhalf is set to **0**, the operations, for example obtaining 3A statistics, are stored in the interrupt service routine (ISR) of hardware interrupts and the time for ISP hardware interrupt processing is about 1 ms. This ensures that 3A statistics are obtained in a timely manner. However, other hardware interrupts may not be responded in a timely manner. When **int_bottomhalf** is set to a non-zero value, the operations, for example obtaining 3A statistics, are moved to the bottom half of the interrupts. In this case, the time for the hardware interrupt processing is short, which ensures that hardware interrupts are responded in a timely manner. However, the 3A statistics may not be obtained in a timely manner. Huawei LiteOS does not support the bottom half function of interrupts. When this parameter is enabled in Huawei LiteOS, the message "Do not support in Lite OS" is displayed.

- Huawei LiteOS does not support the kernel module loading mechanism. To achieve the effect similar to loading the .ko drivers in Linux, run the ISP_init function in **sdk_init.c** under **Huawei LiteOS release/ko**.

[Example]

None

[See Also]

HI_MPI_ISP_Init

HI_MPI_ISP_Exit

[Description]

Exits the ISP firmware.

[Syntax]

```
HI_S32 HI_MPI_ISP_Exit(ISP_DEV IspDev);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

- Header files: hi_comm_isp.h, mpi_isp.h

- Library file: libisp.a

[Note]

- Before calling this MPI to exit the ISP firmware, you must call HI_MPI_ISP_Init and HI_MPI_ISP_Run.
- This MPI does not support multiple processes and it must be called with sensor_register_callback, HI_MPI_AE_Register, HI_MPI_AWB_Register, HI_MPI_AF_Register, HI_MPI_ISP_MemInit, HI_MPI_ISP_Init, and HI_MPI_ISP_Run in the same process.
- This MPI cannot be called repeatedly.

[Example]

None

[See Also]

HI_MPI_ISP_Init

HI_MPI_ISP_SetPubAttr

[Description]

Sets the ISP public attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetPubAttr(ISP_DEV IspDev, const ISP_PUB_ATTR_S
*pstPubAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstPubAttr	ISP public attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 the sensor.
- Before starting the ISP, ensure that HI_MPI_ISP_MemInit.
- This MPI is called to initialize the ISP external registers.
- The resolution and frame rate can be dynamically switched by calling this MPI after the ISP runs.
- After this MPI is called, the handling process in the ISP is as follows:
  - The ISP firmware determines whether the image resolution and frame rate are changed. If no, the ISP exits the switching process. If yes, the ISP firmware calls the **cmos_set_image_mode** function in **sensor cmos.c** to change the sensor mode.
  - If the sensor mode is changed (the return value is 0), the ISP firmware calls the sensor_init function to reconfigure the sensor.
  - The ISP firmware transfers the frame rate information to the HiSilicon AE algorithm library and determines whether to change the frame rate.
- If the sensor mode is changed when this MPI is called to dynamically switch the resolution and the frame rate, see the switching process in the sample code in the SDK (stop the VI device, perform the switching, and start the VI device). During dynamic resolution and frame rate switching, the target resolution or frame rate must be different from the current resolution or frame rate. Otherwise, exceptions may occur because the sensor may not be initialized again.
- When the cropping function provided by the VI device and ISP is used, calling this MPI switches the sensor to the corresponding mode if the resolution and frame rate after cropping are less than those in another sensor mode.
- You can modify the **cmos_set_image_mode** function in **sensor cmos.c** to adjust the switching sequence of the sensor modes. For sensors that provide only the initialization sequences of 5M30fps and 1080P60fps, the 1080P30fps can be obtained by modifying the **cmos_set_image_mode** function to crop the 5M30fps or decrease the frame rate of 1080P60fps.

[Example]

None

[See Also]

HI_MPI_ISP_GetPubAttr

HI_MPI_ISP_GetPubAttr

[Description]

Obtains the ISP public attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetPubAttr(ISP_DEV IspDev, ISP_PUB_ATTR_S *pstPubAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstImageAttr	ISP public attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 running the ISP and calling this MPI, you must set the ISP public attributes.

[Example]

None

[See Also]

HI_MPI_ISP_SetPubAttr

HI_MPI_ISP_SetFMWState

[Description]

Sets the ISP firmware status.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetFMWState(ISP_DEV IspDev, const ISP_FMW_STATE_E
enState);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
enState	ISP firmware status	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

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

[Note]

When **enState** is **ISP_FMW_STATE_FREEZE**, the following algorithms are frozen, and the sensor registers are not configured any more and their values before freezing are retained: 3A algorithms, sharpen algorithm, dynamic range compression (DRC) algorithm, crosstalk removal algorithm, noise reduction (NR) algorithm, anti-fog algorithm, demosaic algorithm, black level algorithm, fixed pattern noise (FPN) removal algorithm, automatic color management (ACM) algorithm, and WDR algorithm. The ISP firmware runs properly only when **enState** is **ISP_FMW_STATE_RUN**.

[Example]

None

[See Also]

HI_MPI_ISP_GetFMWState

HI_MPI_ISP_GetFMWState

[Description]

Obtains the ISP firmware status.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetFMWState(ISP_DEV IspDev, ISP_FMW_STATE_E *penState);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
penState	ISP firmware status	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetFMWState

HI_MPI_ISP_SetWDRMode

[Description]

Sets the WDR mode of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWDRMode(ISP_DEV IspDev, const ISP_WDR_MODE_S
*pstWDRMode);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstWDRMode	WDR mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

- Before starting the ISP, ensure that HI_MPI_ISP_MemInit.
- This MPI is called to initialize the ISP external registers.
- The WDR can be switched by calling this MPI after the ISP runs.
- If the original WDR mode to be switched and the target WDR mode are the same, it is recommended that the upper-layer applications do not reconfigure the mobile industry processor interface (MIPI). Otherwise, the image may fail to be captured. It is recommended that the upper-layer applications determine whether the current WDR mode is the same as the WDR mode to be switched to. If yes, exit and do not perform the switching.

[Example]

None

[See Also]

HI_MPI_ISP_GetWDRMode

HI_MPI_ISP_GetWDRMode

[Description]

Obtains the WDR mode of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWDRMode(ISP_DEV IspDev, ISP_WDR_MODE_S *pstWDRMode);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstWDRMode	Obtained WDR mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetWDRMode

HI_MPI_ISP_SetModuleControl

[Description]

Controls the modules of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetModuleControl(ISP_DEV IspDev, const ISP_MODULE_CTRL_U
*punModCtrl);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
punModCtrl	Module control value	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

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

[Note]

- This MPI can be used to enable or disable the modules of the ISP.
- Each bit of **punModCtrl** controls a 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.

[Example]

None

[See Also]

HI_MPI_ISP_GetModuleControl

HI_MPI_ISP_GetModuleControl

[Description]

Obtains the control status of the modules of the ISP.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetModuleControl(ISP_DEV IspDev, ISP_MODULE_CTRL_U
*punModCtrl);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
punModCtrl	Module control value	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_SetRegister

[Description]

Sets the register value.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetRegister(ISP_DEV IspDev, HI_U32 u32Addr, HI_U32 u32Value);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
u32Addr	Register address	Input
u32Value	Register value	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

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

[Note]

The register address consists of the base address and the offset address. The base addresses for ISP logic registers, ISP external registers, HiSilicon AE external registers, HiSilicon AWB external registers, and HiSilicon AF external registers are 0x205A0000, 0x10000, 0x20000, 0x30000, and 0x40000 respectively.

[Example]

None

[See Also]

HI_MPI_ISP_GetRegister

HI_MPI_ISP_GetRegister

[Description]

Obtains the register value.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetRegister(ISP_DEV IspDev, HI_U32 u32Addr, HI_U32 *pu32Value);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
u32Addr	Register address	Input
pu32Value	Register value	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 register address consists of the base address and the offset address. The base addresses for ISP logic registers, ISP external registers, HiSilicon AE external registers, HiSilicon AWB external registers, and HiSilicon AF external registers are 0x205A0000, 0x10000, 0x20000, 0x30000, and 0x40000 respectively.

[Example]

None

[See Also]

HI_MPI_ISP_SetRegister

HI_MPI_ISP_GetVDTimeOut

[Description]

Obtains ISP interrupt information.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetVDTimeOut(ISP_DEV IspDev, ISP_VD_INFO_S *pstIspVdInfo,
HI_U32 u32MilliSec);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspVdInfo	Pointer to the structure of ISP frame information	Output
u32MilliSec	Timeout period (in ms)	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.
HI_ERR_ISP_NO_INT	The ISP module has no interrupt.

[Requirement]

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

[Note]

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

[Example]

None

[See Also]

None

HI_MPI_ISP_SensorRegCallBack

[Description]

Registers a sensor. This MPI is a callback interface.

[Syntax]

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

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

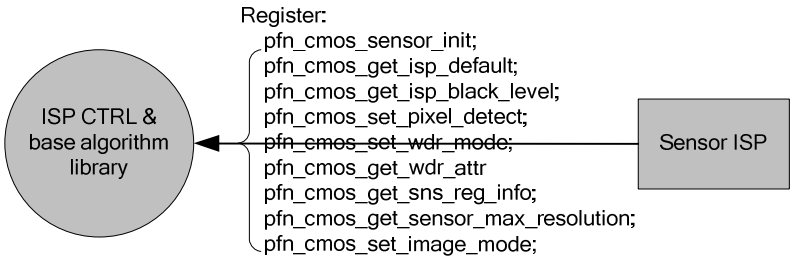
- Header files: hi_comm_isp.h, mpi_isp.h

- Library file: libisp.a

[Note]

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

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



[Example]

```
ISP_DEV IspDev = 0;
HI_S32 s32Ret;
ISP_SENSOR_REGISTER_S stIspRegister;
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIspRegister.stSnsExp;

memset(pstSensorExpFunc, 0, sizeof(ISP_SENSOR_EXP_FUNC_S));
pstSensorExpFunc->pfn_cmos_sensor_init = sensor_init;
pstSensorExpFunc->pfn_cmos_get_isp_default = cmos_get_isp_default;
pstSensorExpFunc->pfn_cmos_get_isp_black_level = cmos_get_isp_black_level;
pstSensorExpFunc->pfn_cmos_set_pixel_detect = cmos_set_pixel_detect;
pstSensorExpFunc->pfn_cmos_set_wdr_mode = cmos_set_wdr_mode;
pstSensorExpFunc->pfn_cmos_get_wdr_attr = cmos_get_wdr_attr;
pstSensorExpFunc->pfn_cmos_get_sns_reg_info = cmos_get_sns_regs_info;
pstSensorExpFunc->pfn_cmos_get_sensor_max_resolution =
cmos_get_sensor_max_resolution;
pstSensorExpFunc->pfn_cmos_set_image_mode = cmos_set_image_mode;
s32Ret = HI_MPI_ISP_SensorRegCallBack(IspDev, IMX178_ID, &stIspRegister);
if (s32Ret)
{
   printf("sensor register callback function failed!\n");
   return s32Ret;
}
```

[See Also]

HI_MPI_ISP_SensorUnRegCallBack

HI_MPI_ISP_SensorUnRegCallBack

[Description]

Deregisters a sensor. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_SensorUnRegCallBack(ISP_DEV IspDev, SENSOR_ID SensorId);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
SensorId	ID of the sensor that is registered with the ISP library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

SensorId can be configured in the sensor library and is used to ensure that the sensor deregistered from the ISP library is the same as that deregistered from the 3A algorithm library.

[Example]

```
ISP_DEV IspDev = 0;
    s32Ret = HI_MPI_ISP_SensorUnRegCallBack(IspDev, IMX178_ID);
    if (s32Ret)
    {
        printf("sensor unregister callback function failed!\n");
        return s32Ret;
    }
```

[See Also]

HI_MPI_ISP_SensorRegCallBack

HI_MPI_ISP_AELibRegCallBack

[Description]

Registers the AE algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AELibRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAeLib,
ISP_AE_REGISTER_S *pstRegister);
```

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

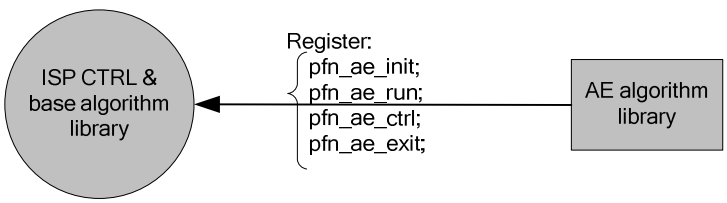
[Requirement]

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

[Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AE algorithm library. If you use the HiSilicon AE algorithm library, this MPI can be ignored. If you use your own AE algorithm library, you need to call this MPI to register callback functions with the ISP library.

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



[Example]

```
ISP_AE_REGISTER_S stRegister;
HI_S32 s32Ret = HI_SUCCESS;
stRegister.stAeExpFunc.pfn_ae_init  = AeInit;
stRegister.stAeExpFunc.pfn_ae_run   = AeRun;
stRegister.stAeExpFunc.pfn_ae_ctrl  = AeCtrl;
stRegister.stAeExpFunc.pfn_ae_exit  = AeExit;
s32Ret = HI_MPI_ISP_AeLibRegCallBack(IspDev, pstAeLib, &stRegister);
if (HI_SUCCESS != s32Ret)
{
printf("Hi_ae register failed!\n");
}
```

[See Also]

HI_MPI_ISP_AELibUnRegCallBack

HI_MPI_ISP_AELibUnRegCallBack

[Description]

Deregisters the AE algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AELibUnRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAeLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAeLib	Pointer to the data structure of the AE algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

If you use the HiSilicon AE algorithm library, this MPI can be ignored. If you use your own AE algorithm library, you need to call this MPI to deregister callback functions from the ISP library.

[Example]

```
HI_S32 s32Ret = HI_SUCCESS;
s32Ret = HI_MPI_ISP_AELibUnRegCallBack(IspDev, pstAeLib);
if (HI_SUCCESS != s32Ret)
{
printf("Hi_ae unregister failed!\n");
}
return s32Ret;
```

[See Also]

HI_MPI_ISP_AELibRegCallBack

HI_MPI_ISP_AWBLibRegCallBack

[Description]

Registers the AWB algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AWBLibRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAwbLib,
ISP_AWB_REGISTER_S *pstRegister);
```

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

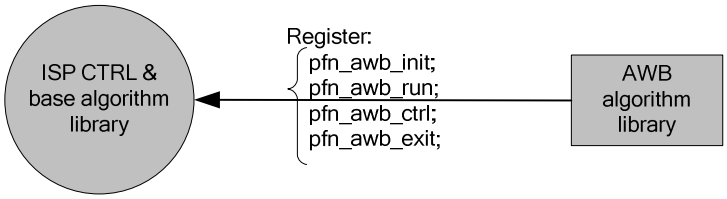
- Header files: hi_comm_isp.h, mpi_isp.h

- Library file: libisp.a

[Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AWB algorithm library. If you use the HiSilicon AWB algorithm library, this MPI can be ignored. If you use your own AWB algorithm library, you need to call this MPI to register callback functions with the ISP library.

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



[Example]

None

[See Also]

HI_MPI_ISP_AWBLibUnRegCallBack

HI_MPI_ISP_AWBLibUnRegCallBack

[Description]

Deregisters the AWB algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AWBLibUnRegCallBack(ISP_DEV IspDev, ALG_LIB_S
*pstAwbLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

None

[Example]

If you use the HiSilicon AWB algorithm library, this MPI can be ignored. If you use your own AWB algorithm library, you need to call this MPI to deregister callback functions from the ISP library.

[See Also]

HI_MPI_ISP_AWBLibRegCallBack

HI_MPI_ISP_AFLibRegCallBack

[Description]

Registers the AF algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AFLibRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAfLib,
ISP_AF_REGISTER_S *pstRegister);
```

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

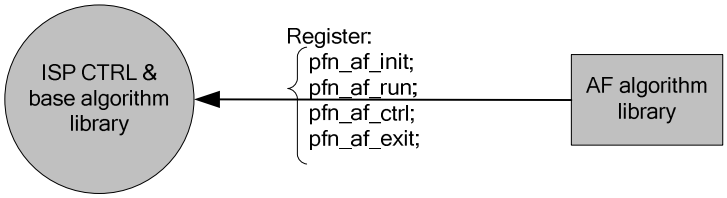
[Requirement]

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

[Note]

The ISP provides unified interfaces for initializing, running, controlling, and destroying the AF algorithm library. If you use the HiSilicon AF algorithm library, this MPI can be ignored. If you use your own AF algorithm library, you need to call this MPI to register callback functions with the ISP library.

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



[Example]

None

[See Also]

HI_MPI_ISP_AFLibUnRegCallBack

HI_MPI_ISP_AFLibUnRegCallBack

[Description]

Deregisters the AF algorithm library. This MPI is a callback interface.

[Syntax]

```
HI_S32 HI_MPI_ISP_AFLibUnRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAfLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAfLib	Pointer to the data structure of the AF algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

- Header files: hi_comm_isp.h, mpi_isp.h

- Library file: libisp.a

[Note]

If you use the HiSilicon AF algorithm library, this MPI can be ignored. If you use your own AF algorithm library, you need to call this MPI to deregister callback functions from the ISP library.

[Example]

None

[See Also]

HI_MPI_ISP_AFLibRegCallBack

HI_MPI_ISP_SetBindAttr

[Description]

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

[Syntax]

```
HI_S32 HI_MPI_ISP_SetBindAttr(ISP_DEV IspDev, const ISP_BIND_ATTR_S
*pstBindAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstBindAttr	Pointer to the binding data structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

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

[Example]

None

[See Also]

HI_MPI_ISP_GetBindAttr

HI_MPI_ISP_GetBindAttr

[Description]

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

[Syntax]

```
HI_S32 HI_MPI_ISP_GetBindAttr(ISP_DEV IspDev, ISP_BIND_ATTR_S
*pstBindAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstBindAttr	Pointer to the binding data structure	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetBindAttr

HI_MPI_ISP_QueryInnerStateInfo

[Description]

Obtains the ISP status information.

[Syntax]

```
HI_S32 HI_MPI_ISP_QueryInnerStateInfo(ISP_DEV IspDev,
ISP_INNER_STATE_INFO_S *pstInnerStateInfo);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstInnerStateInfo	Pointer to the structure of the ISP status information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_ISP_SetDCFInfo

[Description]

Sets DCF parameters.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDCFInfo(ISP_DEV IspDev, const ISP_DCF_INFO_S
*pstIspDCF)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspDCF	Pointer to DCF parameters	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

HI_MPI_VB_SetSupplementConf must be called before this API is called (For details, see section 2.2 "Functions" in the *HiMPP IPC V2.0 Media Processing Software Development Reference*.) to set stSupplementConf to VB_SUPPLEMENT_JPEG_MASK.

[Example]

```
VB_SUPPLEMENT_CONF_S stSupplementConf;

stSupplementConf.u32SupplementConf = VB_SUPPLEMENT_JPEG_MASK;
s32Ret=HI_MPI_VB_SetSupplementConf(&stSupplementConf);
if(HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VB_SetSupplementConf err 0x%x\n",s32Ret);
}

......

s32Ret=HI_MPI_VB_Init();
if(HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_VB_Init err 0x%x\n",s32Ret);
}

......

ISP_DEV IspDev;
```

```
s32Ret=HI_MPI_ISP_Init(IspDev);

......

ISP_DCF_INFO_S stIspDCF;
//will:119 105 108 108
stIspDCF.au8ImageDescription[0]=119;
stIspDCF.au8ImageDescription[1]=105;
stIspDCF.au8ImageDescription[2]=108;
stIspDCF.au8ImageDescription[3]=108;
stIspDCF.au8ImageDescription[4]=0;
//hisi:104 105 115 105
stIspDCF.au8Make[0]=104;
stIspDCF.au8Make[1]=105;
stIspDCF.au8Make[2]=115;
stIspDCF.au8Make[3]=105;
stIspDCF.au8Make[4]=0;
//funy:102 117 110 121
stIspDCF.au8Model[0]=102;
stIspDCF.au8Model[1]=117;
stIspDCF.au8Model[2]=110;
stIspDCF.au8Model[3]=121;
stIspDCF.au8Model[4]=0;
//v.1.1.0:118 46 49 46 49 46 48
stIspDCF.au8Software[0] = 118;
stIspDCF.au8Software[1] = 46;
stIspDCF.au8Software[2] = 49;
stIspDCF.au8Software[3] = 46;
stIspDCF.au8Software[4] = 49;
stIspDCF.au8Software[5] = 46;
stIspDCF.au8Software[6] = 48;
stIspDCF.au8Software[7] = 0;

stIspDCF.u16ISOSpeedRatings = 500;
stIspDCF.u32ExposureBiasValue = 5;
stIspDCF.u32ExposureTime      = 0x00010004;
stIspDCF.u32FNumber           = 0x0001000f;
stIspDCF.u32FocalLength       = 0x00640001;
stIspDCF.u32MaxApertureValue = 0x00010001;
stIspDCF.u8Contrast          =5;
stIspDCF.u8CustomRendered    = 0;
stIspDCF.u8ExposureMode    = 0;
stIspDCF.u8ExposureProgram  = 1;
stIspDCF.u8FocalLengthIn35mmFilm = 0;
```

```
stIspDCF.u8GainControl = 1;
stIspDCF.u8LightSource = 1;
stIspDCF.u8MeteringMode = 1;
stIspDCF.u8Saturation  = 1;
stIspDCF.u8SceneCaptureType = 1;
stIspDCF.u8SceneType = 0;
stIspDCF.u8Sharpness =5;
stIspDCF.u8WhiteBalance = 0;

HI_MPI_ISP_SetDCFInfo(IspDev,&stIspDCF);
```

[See Also]

HI_MPI_ISP_GetDCFInfo

HI_MPI_ISP_GetDCFInfo

[Description]

Obtains DCF parameters.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDCFInfo(ISP_DEV IspDev, ISP_DCF_INFO_S *pstIspDCF)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspDCF	Pointer to DCF parameters	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

HI_MPI_VB_SetSupplementConf must be called before this API is called (For details, see section 2.2 "Functions" in the *HiMPP IPC V2.0 Media Processing Software Development*

Reference.) to set stSupplementConf to VB_SUPPLEMENT_JPEG_MASK, because the DCF information must be stored in the DCF buffer allocated when the ISP is initialized.

[Example]

None

[See Also]

HI_MPI_ISP_SetDCFInfo

HI_MPI_ISP_SetModParam

[Description]

Sets the ISP module parameters.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetModParam (ISP_MOD_PARAM_S *pstIspModParam);
```

[Parameter]

Parameter	Description	Input/Output
pstIspModParam	Pointer to the ISP module parameter structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.
HI_ERR_ISP_NOT_SUPPORT	This function is not supported by the ISP.

[Requirement]

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

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_GetModParam

HI_MPI_ISP_GetModParam

[Description]

Obtains the ISP module parameters.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetModParam (ISP_MOD_PARAM_S *pstIspModParam);
```

[Parameter]

Parameter	Description	Input/Output
pstIspModParam	Pointer to the ISP module parameter structure	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetModParam

2.3 Data Structures

Unless otherwise specified, the default value range of variables is the value range of the corresponding data type, and the default data precision of variables is 1. For example, the value range of the variables of the HI_U8 type is [0, 255].

The following are ISP data structures:

- RECT_S: Defines the start position of the cropping window, image width, and image height.
- ISP_BAYER_FORMAT_E: Defines the format of the input Bayer image.
- ISP_PUB_ATTR_S: Defines the common attributes of ISP.
- ISP_FMW_STATE_E: Defines the ISP firmware status.
- WDR_MODE_E: Defines the WDR mode enumeration.
- ISP_WDR_MODE_S: Defines the WDR mode of the ISP.
- ISP_MODULE_CTRL_U: Defines the control for ISP modules.
- ISP_VD_INFO_S: Defines ISP frame information.
- ISP_SENSOR_REGISTER_S: Defines sensor registration information.
- ISP_SENSOR_EXP_FUNC_S: Defines the sensor callback function.
- ISP_CMOS_SENSOR_IMAGE_MODE: Defines the width, height, and frame rate of the sensor output.
- ISP_CMOS_DEFAULT_S: Defines the initialization parameter for the ISP basic algorithm unit.
- ISP_CMOS_BLACK_LEVEL_S: Defines the sensor black level.
- ISP_SNS_REGS_INFO_S: Defines sensor register information.
- ALG_LIB_S: Defines AE, AWB, and AF algorithm libraries.
- ISP_BIND_ATTR_S: Defines the binding relationship between the ISP library and sensors/3A algorithm libraries.
- ISP_CTRL_PROC_WRITE_S: Defines ISP proc information.
- ISP_CTRL_CMD_E: Defines 3A control commands.
- ISP_AE_REGISTER_S: Defines AE registration information.
- ISP_AE_EXP_FUNC_S: Defines the AE callback function.
- ISP_AE_PARAM_S: Defines the initialization parameter provided by the ISP for the AE algorithm library.
- ISP_AE_INFO_S: Defines the statistics provided by the ISP for the AE algorithm library.
- ISP_AE_RESULT_S: Defines the results returned by the AE algorithm library to the ISP for configuring registers.
- ISP_AWB_REGISTER_S: Defines AWB registration information.
- ISP_AWB_EXP_FUNC_S: Defines the AWB callback function.
- ISP_AWB_PARAM_S: Defines the initialization parameter provided by the ISP for the AWB algorithm library.
- ISP_AWB_INFO_S: Defines the statistics provided by the ISP for the AWB algorithm library.
- ISP_AWB_RESULT_S: Defines the results returned by the AWB algorithm library to the ISP for configuring registers.

- ISP_AF_REGISTER_S: Defines AF registration information.
- ISP_AF_EXP_FUNC_S: Defines the AF callback function.
- ISP_AF_PARAM_S: Defines the initialization parameter provided by the ISP for the AF algorithm library.
- ISP_AF_INFO_S: Defines the statistics provided by the ISP for the AF algorithm library.
- ISP_AF_RESULT_S: Defines the results returned by the AF algorithm library to the ISP for configuring registers.
- ISP_INNER_STATE_INFO_S: Defines the structure of the ISP status information.
- ISP_DCF_INFO_S: Defines the DCF information.
- ISP_MOD_PARAM_S: Defines the structure of the ISP module parameters.

RECT_S

[Description]

Defines the start position of the cropping window, image width, and image height.

[Syntax]

```
typedef struct hiRECT_S
{
    HI_S32 s32X;
    HI_S32 s32Y;
    HI_U32 u32Width;
    HI_U32 u32Height;
}RECT_S;
```

[Member]

Member	Description
s32X	Horizontal start position Value range: [0, u32Width − 480]
s32Y	Vertical start position Value range: [0, u32Height − 240]
u32Width	Image width Value range: [480, 2592]
u32Height	Image height Value range: [240, 2200]

[Note]

The sum of the horizontal start position and image width must be less than the width of the image output from the sensor, and the sum of the vertical start position and image height must be less than the height of the image output from the sensor.

[See Also]

None

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]

None.

ISP_PUB_ATTR_S

[Description]

Defines the common attributes of ISP.

[Syntax]

```
typedef struct hiISP_PUB_ATTR_S
{
    ISP_BAYER_FORMAT_E   stWndRect;
    HI_FLOAT             f32FrameRate;
    ISP_BAYER_FORMAT_E   enBayer;
} ISP_PUB_ATTR_S;
```

[Member]

Member	Description
stWndRect	Start position of the cropping window, image width, and image height
f32FrameRate	Frame rate of the input image Value range: (0.00, 65535.00]
enBayer	Bayer data format

[Note]

None

[See Also]

None

ISP_FMW_STATE_E

[Description]

Defines the ISP firmware status.

[Syntax]

```
typedef enum hiISP_FMW_STATE_E
{
    ISP_FMW_STATE_RUN = 0,
    ISP_FMW_STATE_FREEZE,
    ISP_FMW_STATE_BUTT
} ISP_FMW_STATE_E;
```

[Member]

Member	Description
ISP_FMW_STATE_RUN	Running status
ISP_FMW_STATE_FREEZE	Frozen status

[Note]

None

[See Also]

None

WDR_MODE_E

[Description]

Defines the WDR mode.

[Syntax]

```
typedef enum hiWDR_MODE_E
{
    WDR_MODE_NONE = 0,
    WDR_MODE_BUILT_IN,
    WDR_MODE_2To1_LINE,
    WDR_MODE_2To1_FRAME,
    WDR_MODE_2To1_FRAME_FULL_RATE,
    WDR_MODE_3To1_LINE,
    WDR_MODE_3To1_FRAME,
    WDR_MODE_3To1_FRAME_FULL_RATE,
    WDR_MODE_4To1_LINE,
    WDR_MODE_4To1_FRAME,
    WDR_MODE_4To1_FRAME_FULL_RATE,
    WDR_MODE_BUTT,
} WDR_MODE_E;
```

[Member]

Member	Description
WDR_MODE_NONE	Linear mode
WDR_MODE_BUILT_IN	Sensor combination WDR mode
WDR_MODE_2To1_LINE	WDR mode in which two frames are combined and output as crossed lines
WDR_MODE_2To1_FRAME	WDR mode in which two frames are combined and output as a frame
WDR_MODE_2To1_FRAME_FULL_RATE	WDR mode in which two frames are combined and output as a frame at full frame rate
WDR_MODE_3To1_LINE	WDR mode in which three frames are combined and output as crossed lines
WDR_MODE_3To1_FRAME	WDR mode in which three frames are combined and output as a frame
WDR_MODE_3To1_FRAME_FULL_RATE	WDR mode in which three frames are combined and output as a frame at full frame rate
WDR_MODE_4To1_LINE	WDR mode in which four frames are combined and output as crossed lines
WDR_MODE_4To1_FRAME	WDR mode in which four frames are combined and output as a frame
WDR_MODE_4To1_FRAME_FULL_RATE	WDR mode in which four frames are

Member	Description
	combined and output as a frame at full frame rate

[Note]

- Currently, only the **WDR_MODE_NONE**, **WDR_MODE_BUILT_IN**, **WDR_MODE_2To1_LINE**, **WDR_MODE_2To1_FRAME**, and **WDR_MODE_2To1_FRAME_FULL_RATE** modes are supported.
- **WDR_MODE_BUILT_IN** and **WDR_MODE_2To1_LINE** are supported only by specific sensors.

[See Also]

None

ISP_WDR_MODE_S

[Description]

Defines the WDR mode of the ISP.

[Syntax]

```
typedef struct hiISP_WDR_MODE_S
{
    WDR_MODE_E  enWDRMode;
} ISP_WDR_MODE_S;
```

[Member]

Member	Description
enWDRMode	WDR mode

[Note]

None

[See Also]

None

ISP_MODULE_CTRL_U

[Description]

Defines the control for ISP modules.

[Syntax]

```
typedef union hiISP_MODULE_CTRL_U
{
```

```
    HI_U32  u32Key;
    struct
    {
        HI_U32  bitBypassVideoTest  : 1 ;   /* [0] */
        HI_U32  bitBypassBalanceFe  : 1 ;   /* [1] */
        HI_U32  bitBypassISPDGain     : 1 ;   /* [2] */
        HI_U32  bitBypassGammaFe    : 1 ;   /* [3] */
        HI_U32  bitBypassCrosstalkR : 1 ;   /* [4] */
        HI_U32  bitBypassDPC        : 1 ;   /* [5] */
        HI_U32  bitBypassNR         : 1 ;   /* [6] */
        HI_U32  bitRsv1             : 1 ;   /* [7] */
        HI_U32  bitBypassWBGain     : 1 ;   /* [8] */
        HI_U32  bitBypassShading    : 1 ;   /* [9] */
        HI_U32  bitRsv2             : 1 ;   /* [10] */
        HI_U32  bitBypassDRC        : 1 ;   /* [11] */
        HI_U32  bitBypassDemosaic   : 1 ;   /* [12] */
        HI_U32  bitBypassColorMatrix: 1 ;   /* [13] */
        HI_U32  bitBypassGamma      : 1 ;   /* [14] */
        HI_U32  bitBypassFSWDR      : 1 ;   /* [15] */
        HI_U32  bitGammaFePosition  : 1 ;   /* [16] */
        HI_U32  bit2Rsv3            : 2 ;   /* [17:18] */
        HI_U32  bitBypassCsConv     : 1 ;   /* [19] */
        HI_U32  bit2Rsv4            : 2 ;   /* [20:21] */
        HI_U32  bitBypassSharpen    : 1 ;   /* [22] */
        HI_U32  bitChnSwitch        : 1 ;   /* [23] */
        HI_U32  bit2BypassMode      : 2 ;   /* [24:25] */
        HI_U32  bitBypassAll        : 1 ;   /* [26] */
        HI_U32  bit5Rsv5            : 5 ;   /* [27:31] */
    };
}ISP_MODULE_CTRL_U;
```

[Member]

Member	Description
bitBypassVideoTest	Video test generator bypass
bitBypassBalanceFe	Front-end black level adjustment bypass
bitBypassISPDGain	Digital gain bypass
bitBypassGammaFe	GammaFe table bypass
bitBypassCrosstalkR	Crosstalk removal bypass
bitBypassDPC	Defect pixel correction bypass
bitBypassNR	NR bypass

Member	Description
bitBypassWBGain	White balance gain and offset bypass
bitBypassShading	Lens shading correction bypass
bitBypassDRC	DRC bypass
bitBypassDemosaic	Demosaic module bypass
bitBypassColorMatrix	Color matrix bypass
bitBypassGamma	Gamma table bypass
bitBypassFSWDR	Multi-frame combination WDR bypass
bitGammaFePosition	0: The GammaFe table is placed between the front-end black level and GE and applies to the linear mode and the sensor built-in WDR mode. 1: The GammaFe table is placed after the FSWDR and applies to the multi-frame combination WDR mode.
bitBypassCsConv	CSC bypass
bitBypassSharpen	Sharpen RGB bypass
bitChnSwitch	Input port 1 and port 2 switch
bit2BypassMode	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. 10: 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 is set to 0.
bitBypassAll	The output end directly connects to the input end.

[Note]

None

[See Also]

None

ISP_VD_INFO_S

[Description]

Defines ISP frame information.

[Syntax]

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

```
} ISP_VD_INFO_S;
```

[Member]

Member	Description
u32Reserved	Reserved bytes

[Note]

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

[See Also]

None

ISP_SENSOR_REGISTER_S

[Description]

Defines sensor registration information.

[Syntax]

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

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]

ISP_SENSOR_EXP_FUNC_S

ISP_SENSOR_EXP_FUNC_S

[Description]

Defines the sensor callback function.

[Syntax]

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

```
    HI_VOID(*pfn_cmos_sensor_global_init)(HI_VOID);
    HI_S32(*pfn_cmos_set_image_mode)(ISP_CMOS_SENSOR_IMAGE_MODE_S
*pstSensorImageMode);
    HI_VOID(*pfn_cmos_set_wdr_mode)(HI_U8 u8Mode);
    HI_U32(*pfn_cmos_get_isp_default)(ISP_CMOS_DEFAULT_S *pstDef);
    HI_U32(*pfn_cmos_get_isp_black_level)(ISP_CMOS_BLACK_LEVEL_S
*pstBlackLevel);
    HI_U32(*pfn_cmos_get_sns_reg_info)(ISP_SNS_REGS_INFO_S
*pstSnsRegsInfo);
    HI_VOID(*pfn_cmos_set_pixel_detect)(HI_BOOL bEnable);
} ISP_SENSOR_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_cmos_sensor_init	Pointer to the callback function for initializing sensors
pfn_cmos_sensor_global_init	Pointer to the callback function for initializing global variables
pfn_cmos_set_image_mode	Pointer to the callback function for configuring the switching of the resolution and the frame rate. If 0 is returned, the sensor mode is changed, and the ISP calls **pfn_cmos_sensor_init** to reconfigure the sensor. If other values are returned, the sensor mode remains unchanged, and the ISP does not reconfigure the sensor.
pfn_cmos_set_wdr_mode	Pointer to the callback function for setting the WDR mode
pfn_cmos_get_isp_default	Pointer to the callback function for obtaining the initial value of the ISP basic algorithm
pfn_cmos_get_isp_black_level	Pointer to the callback function for obtaining the sensor black level, which allows the black level to be dynamically adjusted based on the sensor gain. If the black level is dynamically adjusted, it cannot be configured by calling HI_MPI_ISP_SetBlackLevelAttr.
pfn_cmos_get_sns_reg_info	Pointer to the callback function for obtaining sensor register information, which is used to configure the AE information in kernel mode.
pfn_cmos_set_pixel_detect	Pointer to the callback function for enabling or disabling defect pixel correction

[Note]

Values must be assigned to members **pfn_cmos_sensor_init**, **pfn_cmos_get_isp_default**, **pfn_cmos_get_isp_black_level**, **pfn_cmos_set_pixel_detect**, and

pfn_cmos_get_sns_reg_info. If no values need to be assigned to other callback function pointers, set these pointers to **NULL**. For example, **pfn_cmos_set_image_mode** needs to be set to **NULL** if a sensor that does not support resolution switching is used.

[See Also]

ISP_SENSOR_REGISTER_S

ISP_CMOS_SENSOR_IMAGE_MODE

[Description]

Defines the width, height, and frame rate of the sensor output.

[Syntax]

```
typedef struct hiISP_CMOS_SENSOR_IMAGE_MODE_S
{
    HI_U16 u16Width;
    HI_U16 u16Height;
    HI_U16 u16Fps;
}ISP_CMOS_SENSOR_IMAGE_MODE;
```

[Member]

Member	Description
u16Width	Width of the sensor output
u16Height	Height of the sensor output
u16Fps	Frame rate of the sensor output

[Note]

None

[See Also]

ISP_SENSOR_EXP_FUNC_S

ISP_CMOS_DEFAULT_S

[Description]

Defines the initialization parameter for the ISP basic algorithm unit.

[Syntax]

```
typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_DRC_S         stDrc;
    ISP_CMOS_AGC_TABLE_S    stAgcTbl;
    ISP_CMOS_NOISE_TABLE_S  stNoiseTbl;
    ISP_CMOS_DEMOSAIC_S     stDemosaic;
```

```
    ISP_CMOS_GAMMAFE_S      stGammafe;
    ISP_CMOS_GAMMA_S        stGamma;
    ISP_CMOS_SHADING_S      stShading;
    ISP_CMOS_RGBSHARPEN_S   stRgbSharpen;
    ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
} ISP_CMOS_DEFAULT_S;
```

[Member]

Member	Sub Member	Description
stDrc	bEnable	HI_FALSE: The DRC is disabled. HI_TRUE: The DRC is enabled. In linear mode, the default value is HI_FALSE. In WDR mode, the default value is HI_TRUE and it must be set to **HI_TRUE**.
	u32BlackLevel	Lower limit of the DRC algorithm. Pixels whose values are less than **u32Blacklevel** are not involved in the DRC calculation. Value range: [0, 0xFFF] The default value is 0.
	u32WhiteLevel	Upper limit of the DRC algorithm. Pixels whose values are greater than **u32Whitelevel** are not involved in the DRC calculation. Value range: [0, 0xFFF] In linear mode, the default value is 0x4FF. In WDR mode, the default value is 0xFFF.
	u32SlopeMax	Control parameter for DRC tone curves, which is used to limit the maximum slope (gain) for the dark regions on the DRC curve Value range: [0, 0xFF] In linear mode, the default value is 0x30. In WDR mode, the default value is 0x38.
	u32SlopeMin	Control parameter for DRC tone curves, which is used to limit the minimum slope (gain) for the bright regions on the DRC curve Value range: [0, 0xFF] In linear mode, the default value is 0x0. In WDR mode, the default value is 0xC0.
	u32VarianceSpace	Spatial domain sensitivity of the DRC algorithm. A larger value indicates that more surrounding pixels are referenced when tone_curve is generated. Value range: [0x0, 0xF]

Member	Sub Member	Description
		In linear mode, the default value is 0x4. In WDR mode, the default value is 0xA.
	u32VarianceIntensity	Luminance domain sensitivity of the DRC algorithm. A larger value indicates that the difference between the tone_curve of each pixel and that of its surrounding pixels is smaller. Value range: [0x0, 0xF] In linear mode, the default value is 0x1. In WDR mode, the default value is 0x4.
stAgcTbl	bValid	Data validity identifier of the data structure Value: 0 or 1
	au8SharpenAltD	Interpolation array for dynamically adjusting the sharpness of the large edges of images based on the gain Value range: [0, 255]
	au8SharpenAltUd	Interpolation array for dynamically adjusting the sharpness of the small textures of images based on the gain Value range: [0, 255]
	au8SnrThresh	Interpolation array for dynamically setting the image NR strength based on the gain Value range: [0, 255]
	au8DemosaicLumThresh	Array for setting the luminance threshold for the sharpness of large edges of images The default value is recommended. Value range: [0, 255]
	au8DemosaicNpOffset	Array for setting image noise parameters The default value is recommended. Value range: [0, 255]
	au8GeStrength	Array for setting the parameters of the green equalization parameter The default value is recommended. Value range: [0, 255]
	au8SharpenRGB	Interpolation array for dynamically adjusting the overall image sharpness based on the gain Value range: [0, 255]
stNoiseTbl	bValid	Data validity identifier of the data structure Value: 0 or 1
	au8NoiseProfileWeightLut	Array for setting the noise profile related to sensor features. The array value is used as the

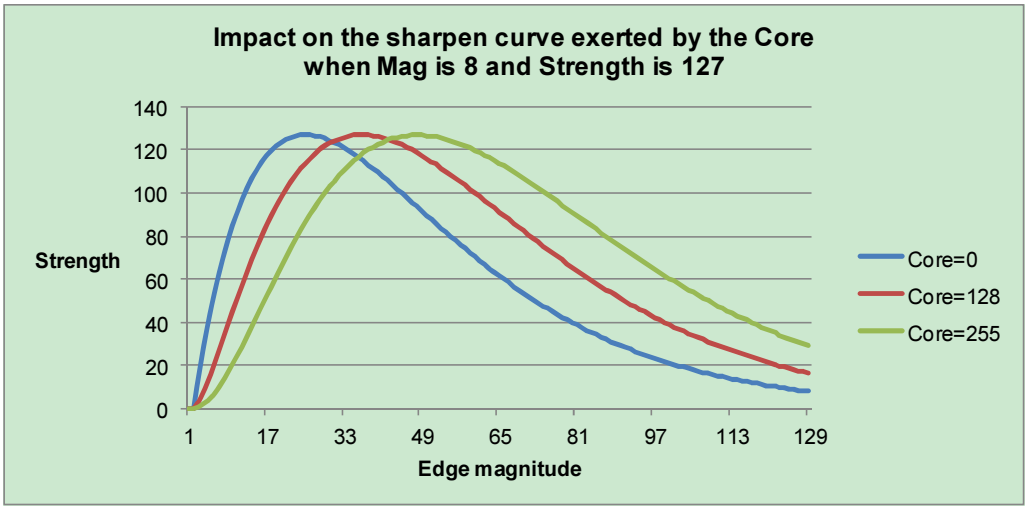
Member	Sub Member	Description
		input of the NR module. The default value is recommended. Value range: [0, 255]
	au8DemosaicWeightLut	Array for setting the noise profile related to sensor features. The array value is used as the input of the demosaic module. The default value is recommended. Value range: [0, 255]
stDemosaic	bValid	Data validity identifier of the data structure Value: 0 or 1
	u8VhSlope	Vertical/Horizontal slope threshold The default value is recommended. Value range: [0, 255]
	u8AaSlope	Angle slope threshold The default value is recommended. Value range: [0, 255]
	u8VaSlope	VH-AA slope threshold The default value is recommended. Value range: [0, 255]
	u8UuSlope	Undefined slope threshold The default value is recommended. Value range: [0, 255]
	u8SatSlope	Saturation slope threshold The default value is recommended. Value range: [0, 255]
	u8AcSlope	High-frequency component filtering slope threshold The default value is recommended. Value range: [0, 255]
	u8FcSlope	Anti-false color slope threshold The default value is recommended. Value range: [0, 255]
	u16VhThresh	Vertical/Horizontal threshold The default value is recommended. Value range: [0, 0xFFFF]
	u16AaThresh	Angle threshold The default value is recommended.

Member	Sub Member	Description
		Value range: [0, 0xFFFF]
	u16VaThresh	VA threshold The default value is recommended. Value range: [0, 0xFFFF]
	u16UuThresh	Undefined threshold The default value is recommended. Value range: [0, 0xFFFF]
	u16SatThresh	Saturation threshold The default value is recommended. Value range: [0, 0xFFFF]
	u16AcThresh	High-frequency component filtering threshold The default value is recommended. Value range: [0, 0xFFFF]
stGammafe	bValid	Data validity identifier of the data structure Value: 0 or 1 The value must be configured in WDR mode.
	au16Gammafe0	GammaFe0 table. It is generated by the correction tool, and is used to reduce the error during the interpolation of the lookup table and implement the image compression function by working with **au16Gammafe1**. Value range: [0, 0xFFFF]
	au16Gammafe1	GammaFe1 table. It is generated by the correction tool, and is used to implement the image compression function by working with **au16Gammafe0**. Value range: [0, 0xFFFF]
stGamma	bValid	Data validity identifier of the data structure. If the default gamma curve is used, the member can be set to an invalid value. Value: 0 or 1
	au16Gamma	Gamma table Value range: [0, 0xFFFF]
stShading	bValid	Data validity identifier of the data structure. The value is 0 or 1. If shading correction is not required, you can set this member to make data invalid.
	u16RCenterX	Horizontal coordinate of the R component center Value range: [0x0, 0xFFFF]

Member	Sub Member	Description
	u16RCenterY	Vertical coordinate of the R component center Value range: [0x0, 0xFFFF]
	u16GCenterX	Horizontal coordinate of the G component center Value range: [0x0, 0xFFFF]
	u16GCenterY	Vertical coordinate of the G component center Value range: [0x0, 0xFFFF]
	u16BCenterX	Horizontal coordinate of the B component center Value range: [0x0, 0xFFFF]
	u16BCenterY	Vertical coordinate of the B component center Value range: [0x0, 0xFFFF]
	au16RShadingTbl	Correction table of the R component Value range: [0x0, 0xFFFF]
	au16GShadingTbl	Correction table of the G component Value range: [0x0, 0xFFFF]
	au16BShadingTbl	Correction table of the B component Value range: [0x0, 0xFFFF]
	u16ROffCenter	Distance between the R component center and the farthest angle. A longer distance indicates a smaller value. Value range: [0x0, 0xFFFF]
	u16GOffCenter	Distance between the G component center and the farthest angle. A longer distance indicates a smaller value. Value range: [0x0, 0xFFFF]
	u16BOffCenter	Distance between the B component center and the farthest angle. A longer distance indicates a smaller value. Value range: [0x0, 0xFFFF]
	u16TblNodeNum	Number of used nodes in each component correction table. Value range: [0x0, 0x81] Default value: 0x81
stRgbSharpen	bValid	Data validity identifier of the data structure. The value is 0 or 1. If the default SharpenRGB curve is used, the member can be set to an invalid value.
	u8LutCore	It is used to generate sharpen RGB curves. The

Member	Sub Member	Description
		parameter exerts more impact on the rising slope of the sharpen RGB curves than on the falling slope. See Figure 2-5. Value range: [0, 255]
	u8LutStrength	It is used to generate sharpen RGB curves. The parameter affects the strength of the sharpen RGB curves. See Figure 2-6. Value range: [0, 127]
	u8LutMagnitude	It is used to generate sharpen RGB curves. The parameter exerts more impact on the falling slope of the sharpen RGB curves than on the rising slope. See Figure 2-7. Value range: [0, 31]
stSensorMaxResolution	u32MaxWidth	Maximum width supported by the sensor, ensuring that the configured width does not exceed the maximum width when the resolution is switched
	u32MaxHeight	Maximum height supported by the sensor, ensuring that the configured height does not exceed the maximum height when the resolution is switched

Figure 2-5 Impact on the sharpen curve exerted by the Core when Mag is 8 and Strength is 127



NOTE

The sharpen curves indicate the RGB sharpen adjustment strength with different edge magnitudes.

Figure 2-6 Impact on the sharpen curve exerted by the Mag when Core is 255 and Strength is 127

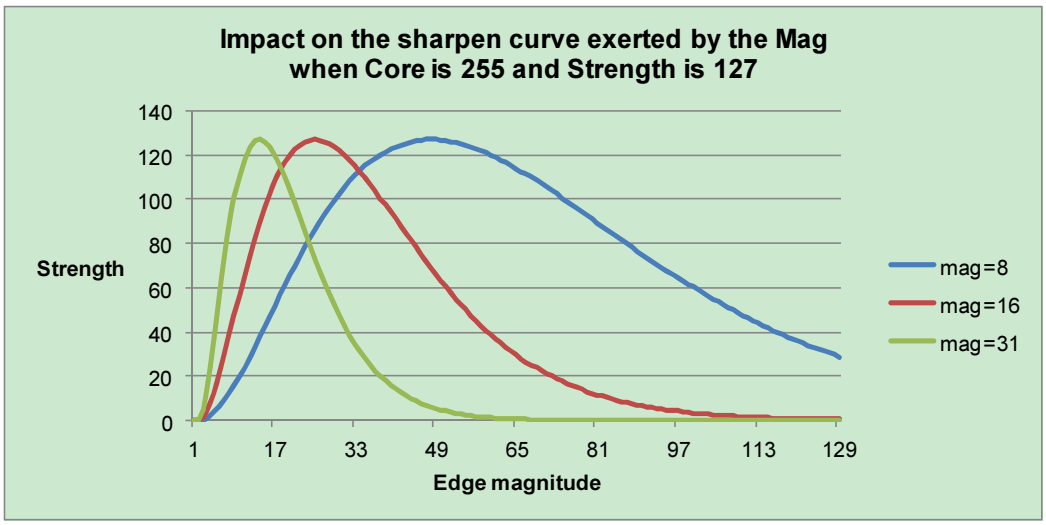
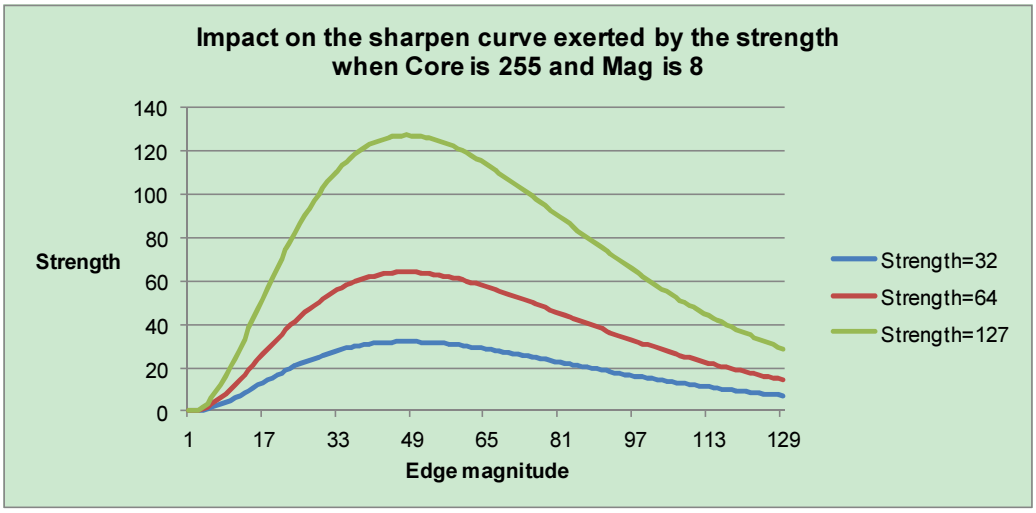


Figure 2-7 Impact on the sharpen curve exerted by the Strength when Core is 255 and Mag is 8



[Note]

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

[See Also]

ISP_SENSOR_EXP_FUNC_S

ISP_CMOS_BLACK_LEVEL_S

[Description]

Defines the sensor black level.

[Syntax]

```
typedef struct hiISP_CMOS_BLACK_LEVEL_S
{
    HI_BOOL bUpdate;
    HI_U16   au16BlackLevel[4];
} ISP_CMOS_BLACK_LEVEL_S;
```

[Member]

Member	Description
bUpdate	Whether the sensor black level varies according to the gain Value range: [0, 1] If the value is set to **HI_TRUE**, the black level value cannot be set by calling HI_MPI_ISP_SetBlackLevelAttr.
au16BlackLevel	Sensor black level array Value range: [0, 65535]

[Note]

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

[See Also]

ISP_SENSOR_EXP_FUNC_S

ISP_SNS_REGS_INFO_S

[Description]

Defines sensor register information.

[Syntax]

```
typedef struct hiISP_SNS_REGS_INFO_S
{
    ISP_SNS_TYPE_E enSnsType;
    HI_U32  u32RegNum;
    HI_U8   u8Cfg2ValidDelayMax;
    union
    {
        ISP_I2C_DATA_S astI2cData[ISP_MAX_SNS_REGS];
        ISP_SSP_DATA_S astSspData[ISP_MAX_SNS_REGS];
    };
} ISP_SNS_REGS_INFO_S;
```

[Member]

Member	Sub Member	Description
enSnsType	ISP_SNS_I2C_TYPE	Communication between the sensor and the ISP over the I^2C interface
	ISP_SNS_SSP_TYPE	Communication between the sensor and the ISP over the SSP interface
u32RegNum	None	Number of registers required when exposure results are written to the sensor. The member value cannot be dynamically changed.
u8Cfg2ValidDelayMax	None	Maximum number of delayed frames from the time when all sensor registers are configured to the time when configurations take effect, which is used to ensure the synchronization between sensor registers and ISP registers Typically, the exposure time register of the CMOS sensor has the longest delay, which is 1−2 frames. Therefore, the value is typically set to 1 or 2.
astI2cData	bUpdate	HI_TRUE: The sensor registers are written. HI_FALSE: The sensor registers are not written.
	u8DelayFrmNum	Number of delayed frames for the sensor register This member is configured to ensure that the exposure time and gain take effect at the same time.
	u8DevAddr	Sensor device address
	u32RegAddr	Sensor register address
	u32AddrByteNum	Bit width of the sensor register address
	u32Data	Sensor register data
	u32DataByteNum	Bit width of sensor register data
astSspData	bUpdate	HI_TRUE: The sensor registers are written. HI_FALSE: The sensor registers are not written.
	u8DelayFrmNum	Number of delayed frames for the sensor register This member is configured to ensure that the exposure time and gain take effect at the same time.
	u32DevAddr	Sensor device address
	u32DevAddrByteNum	Bit width of the sensor device address
	u32RegAddr	Sensor register address
	u32AddrByteNum	Bit width of the sensor register address
	u32Data	Sensor register data

Member	Sub Member	Description
	u32DataByteNum	Bit width of sensor register data

[Note]

None

[See Also]

ISP_SENSOR_EXP_FUNC_S

ALG_LIB_S

[Description]

Defines AE, AWB, and AF algorithm libraries.

[Syntax]

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

[Member]

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

[Note]

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

[See Also]

None

ISP_BIND_ATTR_S

[Description]

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

[Syntax]

```
typedef struct hiISP_BIND_ATTR_S
{
    SENSOR_ID   SensorId;
```

```
    ALG_LIB_S   stAeLib;
    ALG_LIB_S   stAfLib;
    ALG_LIB_S   stAwbLib;
} ISP_BIND_ATTR_S;
```

[Member]

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

[Note]

None

[See Also]

None

ISP_CTRL_PROC_WRITE_S

[Description]

Defines ISP proc information.

[Syntax]

```
typedef struct hiISP_CTRL_PROC_WRITE_S
{
    HI_CHAR *pcProcBuff;
    HI_U32   u32BuffLen;
    HI_U32   u32WriteLen;
} ISP_CTRL_PROC_WRITE_S;
```

[Member]

Member	Description
pcProcBuff	Pointer to the buffer for storing the proc information transferred from the ISP to the current algorithm
u32BuffLen	Size (in byte) of the remaining space of the buffer for storing the proc information transferred from the ISP to the current algorithm. The total size of the buffer is 10 KB.
u32WriteLen	Length (in byte) of the proc information transferred from the current algorithm to the ISP

[Note]

Pay attention to this data structure only when you use a customized 3A algorithm and require the proc information that supports the 3A algorithm.

[See Also]

None

ISP_CTRL_CMD_E

[Description]

Defines 3A control commands.

[Syntax]

```
typedef enum hiISP_CTRL_CMD_E
{
    ISP_WDR_MODE_SET = 8000,
    ISP_PROC_WRITE,
    ISP_AE_FPS_BASE_SET,
    ISP_AWB_ISO_SET,  /* set iso, change saturation when iso change */
    ISP_CHANGE_IMAGE_MODE_SET,
    ISP_DCFINFO_GET,
    ISP_AWB_INTTIME_SET,
    ISP_CTRL_CMD_BUTT,
} ISP_CTRL_CMD_E;
```

[Member]

Member	Description
ISP_WDR_MODE_SET	WDR mode. This command is used to configure the WDR mode of the ISP control unit to the algorithm module. The parameter data structure corresponding to this command is WDR_MODE_E.
ISP_PROC_WRITE	RPOC write information. This command is used to configure the proc information of the algorithm module to the ISP control unit. The parameter data structure corresponding to this command is ISP_CTRL_PROC_WRITE_S.
ISP_AE_FPS_BASE_SET	Frame rate. This command is used to configure the frame rate of the ISP control unit to the AE module. The parameter in this command is the same as **f32FrameRate** in ISP_PUB_ATTR_S.
ISP_AWB_ISO_SET	ISO value. This command is used to configure the current ISO value of the AE module to the AWB module for automatically adjusting the saturation. The parameter in this command is the same as **u32Iso** in ISP_AE_RESULT_S.

Member	Description
ISP_CHANGE_IMAGE_MODE_SET	Image resolution switching identifier. This command is used to configure the image resolution switching identifier to the algorithm module. The parameter data type corresponding to the command is HI_U8. The value 0 indicates that the image resolution is not switched, and other values indicate that the image resolution has been switched.
ISP_DCFINFO_GET	DCF information. This command is used to configure the DCF information of the algorithm module to the ISP control unit. The parameter data structure corresponding to this command is ISP_DCF_INFO_S in **hi_comm_video.h**.
ISP_AWB_INTTIME_SET	Exposure. This command is used to configure the current exposure value of the AE module to the AWB module for outdoor and indoor detection. The parameter in this command is the same as **u32IntTimeUs** in ISP_AE_RESULT_S.

[Note]

None

[See Also]

None

ISP_AE_REGISTER_S

[Description]

Defines AE registration information.

[Syntax]

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

[Member]

Member	Description
stAeExpFunc	Callback function for registering the AE algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

None

ISP_AE_EXP_FUNC_S

[Description]

Defines the AE callback function.

[Syntax]

```
typedef struct hiISP_AE_EXP_FUNC_S
{
    HI_S32 (*pfn_ae_init)(HI_S32 s32Handle, const ISP_AE_PARAM_S
*pstAeParam);
    HI_S32 (*pfn_ae_run)(HI_S32 s32Handle, const ISP_AE_INFO_S *pstAeInfo,
ISP_AE_RESULT_S *pstAeResult, HI_S32 s32Rsv);
    HI_S32 (*pfn_ae_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);
    HI_S32 (*pfn_ae_exit)(HI_S32 s32Handle);
} ISP_AE_EXP_FUNC_S;
```

[Member]

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

[Note]

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

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

```
    AE_CHECK_POINTER(pValue);
    switch (u32Cmd)
    {
        case ISP_WDR_MODE_SET :
            ...
            break;
        ...
    }
    return HI_SUCCESS;
}
```

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

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

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

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

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

[See Also]

ISP_AE_REGISTER_S

ISP_AE_PARAM_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AE_PARAM_S
```

```
{
    SENSOR_ID SensorId;
    HI_U8   u8WDRMode;
    HI_FLOAT f32Fps;
    HI_S32 s32Rsv;
} ISP_AE_PARAM_S;
```

[Member]

Member	Description
SensorId	ID of the sensor that is registered with the ISP library. This ID is used to check whether the sensor registered with the ISP library is the same as the one registered with the AE algorithm library.
u8WDRMode	WDR mode. The ISP provides the WDR mode information for the AE.
f32Fps	Frame rate. The ISP provides the frame rate information for the AE.

[Note]

None

[See Also]

ISP_AE_EXP_FUNC_S

ISP_AE_INFO_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AE_INFO_S
{
   HI_U32  u32FrameCnt;    /* the counting of frame */
   ISP_AE_STAT_1_S *pstAeStat1;
   ISP_AE_STAT_2_S *pstAeStat2;
   ISP_AE_STAT_3_S *pstAeStat3;
   ISP_AE_STAT_4_S *pstAeStat4;
   ISP_AE_STAT_5_S *pstAeStat5;
} ISP_AE_INFO_S;
```

[Member]

Member	Sub Member	Description
u32FrameCnt	None	Total number of frames Value range: [0, 0xFFFFFFFF]
pstAeStat1	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram Value range: [0, 255]
	au16MeteringHist	Statistics array of the 5-segment histogram Value range: [0, 0xFFFF] **au16MeteringHist[0]** indicates the first segment, **au16MeteringHist[1]** indicates the second segment, **au16MeteringHist[2]** indicates the fourth segment, and **au16MeteringHist[3]** indicates the fifth segment. The value of the third segment is calculated as follows: 0xFFFF − (**au16MeteringHist[0]** + **au16MeteringHist[1]** + **au16MeteringHist[2]** + **auMeteringHist[3]**).
pstAeStat2	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram Value range: [0, 255]
	au16MeteringMemArrary	Zone statistics array of the 5-segment histogram Value range: [0, 0xFFFF]
pstAeStat3	au32HistogramMemArray	Statistics array of the 256-segment histogram Value range: [0, 0xFFFFFFFF]
pstAeStat4	u16GlobalAvgR	Average value of R component in global statistics Value range: [0, 0xFFFF]
	u16GlobalAvgGr	Average value of Gr component in global statistics Value range: [0, 0xFFFF]
	u16GlobalAvgGb	Average value of Gb component in global statistics Value range: [0, 0xFFFF]
	u16GlobalAvgB	Average value of B component in global statistics Value range: [0, 0xFFFF]
pstAeStat5	au16ZoneAvg	Average values of components R, Gr, Gb, and B in zoned statistics

Member	Sub Member	Description
		Value range: [0, 0xFFFF]

[Note]

- The operation frequency (for example, once per two frames) of the AE algorithm library can be controlled by setting **u32FrameCnt**.
- **pstAeStat1** and **pstAeStat2** indicate the normalized global and zoned 5-segment histogram statistics respectively obtained according to the segmentation thresholds of the histogram. The value ranges of the two parameters are [0, 0xFFFF]. Taking the global 5-segment histogram for example, if the values of all the pixels in the picture are greater than the maximum threshold, the data of the fifth segment in the histogram is 0xFFFF and the data of the other four segments in the histogram is 0. The global 5-segment histogram is affected by the zone weights and is irrelevant to the position of the statistics module in the ISP pipeline.
- **pstAeStat3** indicates the global 256-segment histogram statistics. The statistics is obtained by collecting the upper eight bits of the data in the input data stream. The data of each segment indicates the number of pixel points corresponding to a specific gray scale. The sum of the data of the 256 segments is the number of pixel points involved in the statistics. The sum is determined by the register with the address of 0x205a06c0, as shown in Figure 3-2. Once the value of the register with the address of 0x205a06c0 is determined, the sum of the data of the 256 segments is determined. For the HiSilicon AE algorithm, the statistics on the Gr channel is used by default, the statistics on the R and Gb channels is used in scenarios with a large red area, and the statistics on the B and Gr channels is used in scenarios with a large blue area. If the configured position of the statistics module for the 256-segment histogram is the same as the position of the statistics module for the 5-segment histogram in the ISP pipeline, the 256-segment histogram will be affected by the zone weights.

Figure 2-8 Description of the register with the address of 0x205a06c0

Addr	Mode	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Default
0x06C0	R/W	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0x00000000
										offset y	skip y			offset x	skip x			

skip x [2:0]	Histogram decimation in horizontal direction: 0=every 2nd pixel; 1=every 3rd pixel; 2=every 4th pixel; 3=every 5th pixel; 4=every 8th pixel ; 5+=every 9th pixel
offset x	0= start from the first column; 1=start from second column
skip y [2:0]	Histogram decimation in vertical direction: 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel ; 6+=every 9th pixel
offset y	0= start from the first row; 1= start from second row

- **pstAeStat4** indicates the normalized average values of the global components R, Gr, Gb, and B. The value range of **pstAeStat4** is [0, 0xFFFF]. If the statistics is a 12-bit number, the value 0xFFFF indicates that the average value of a component is 4095 (maximum value of a 12-bit number). The average values of the global four components are affected by the zone weights and are irrelevant to the position of the statistics module in the ISP pipeline.

- **pstAeStat5** indicates the normalized average values of the components R, Gr, Gb, and B for each zone among the 15 x 17 zones. The value range of **pstAeStat5** is [0, 0xFFFF], and the value meaning is the same as that of **pstAeStat4**.
- If the preceding statistics modules are after GammaFE in the ISP pipeline in WDR mode, data after root extraction is collected for statistics. For details about how to change the positions of the preceding statistics modules in the ISP pipeline, see chapter 7 "Statistics."

[See Also]

ISP_AE_EXP_FUNC_S

ISP_AE_RESULT_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AE_RESULT_S
{
    HI_U32  au32IntTime[4];
    HI_U32  u32IspDgain;
    HI_U32  u32Iso;
    HI_U8   u8AERunInterval;
    HI_BOOL bPirisValid;
    HI_S32  s32PirisPos;
    HI_U32  u32PirisGain;
    ISP_FSWDR_MODE_E enFSWDRMode;
    ISP_AE_STAT_ATTR_S stStatAttr;
} ISP_AE_RESULT_S;
```

[Member]

Member	Sub Member	Description
au32IntTime	N/A	Exposure time calculated by the AE (in μs). **f32Offset** in **cmos.c** must be considered when the exposure time in the unit of line is converted into that in the unit of μs. In linear mode and sensor built-in WDR mode, only **au32IntTime[0]** is valid. It is recommended that **au32IntTime[1]** to **au32IntTime[3]** be set to the same value as **au32IntTime[0]**. In *N*-frame combination WDR mode, **au32IntTime[0]** to **au32IntTime[*N* − 1]** are valid and their configured values are in ascending order, indicating the shortest exposure time to the longest exposure time. The values are used for calculating the ratio of the long frame exposure to the short frame exposure. It is recommended that **au32IntTime[*N* − 1]** to **au32IntTime[3]** be set to the same value as **au32IntTime[*N* − 1]**.

Member	Sub Member	Description
		The value of **au32IntTime[0]** needs to be transferred to other modules for associated control related to the exposure time, which affects the HiSilicon AWB effect. This data structure must be configured when the HiSilicon AWB algorithms and multi-frame combination WDR mode are used.
u32IspDgain	N/A	ISP digital gain (8-bit precision). This member must be configured when the ISP digital gain is used and is set to **0x100** when the ISP digital gain is not used.
u32Iso	N/A	Total gain calculated by the AE algorithm library. The ISO indicates the system gain and is multiplied by 100. For example, if the system sensor gain is 2x and the ISP gain is 1x, the ISO of the entire system is 200 (2 x 1 x 100). This calculation method applies to all the ISO in this document. This member affects the adaptation (such as denoising and sharpening) effect and therefore must be configured.
u8AERunInterval	N/A	Running interval of the AE algorithm. The value range is [1, 255]. The value 1 indicates that the AE algorithm runs for every frame, the value 2 indicates that the AE algorithm runs for every two frames, and so on. The recommended maximum value is 2; otherwise, the AE adjustment speed is affected. In WDR mode, the recommended value is 1, which ensures smooth AE convergence. This member determines the number of interval frames during the time the calculation results of the AE algorithms are configured to the sensor registers and ISP registers. Therefore, this member must be configured.
bPirisValid	-	P iris validity flag. When **bPirisValid** is set to **HI_TRUE**, the P iris drive is called back to configure the position of the stepper motor in kernel mode. When **bPirisValid** is set to **HI_FALSE**, the P iris drive is not called back. When the HiSilicon AE algorithms are used, **bPirisValid** must be set to **HI_TRUE** if the HiSilicon P iris drive connects to the P-iris lens, and it must be set to **HI_FALSE** if the HiSilicon P iris drive connects to a non-P-iris lens.
s32PirisPos	-	Position of the P iris stepper motor. The value range depends on the specifications of the P-iris lens. **s32PirisPos** must be configured when the HiSilicon P iris drive connects to the P-iris lens.
u32PirisGain	-	Equivalent gain of the P iris. The value range

Member	Sub Member	Description
		depends on the specifications of the P-iris lens. This parameter is used to calculate the equivalent exposure when the P iris is working. The equivalent exposure serves as a reference for other modules. The value range is [0, 1024]. When the AE algorithm developed by customers connects to a non-P-iris lens, this value can be set to 512.
enFSWDRMode	-	FSWDR mode. It contains common WDR mode and long frame mode, which are referenced by other modules. When long frame mode is enabled, the optimization parameters can be set. For the AE algorithms developed by the customers, this member can be set to **ISP_FSWDR_NORMAL_MODE** if the long frame mode is not supported. This member can be set to **ISP_FSWDR_LONG_FRAME_MODE** if the FSWDR module outputs only long frame data. Note that the long frame mode only takes effect in the line WDR mode.
stStatAttr	bChange	Whether the sub members of **stStatAttr** need to be configured again
	au8MeteringHistThresh	Segmentation threshold array of the 5-segment histogram Value range: [0, 255]
	au8WeightTable	AE weight table of 15 x 17 zones Value range: [0, 255]

[Note]

- The ISP basic algorithm unit adjusts the parameters such as the sharpness and NR parameters based on the total gain calculated by the AE algorithm library.
- There are default segmentation thresholds of the 5-segment histogram and AE weight table in the ISP library. You are advised to configure the segmentation threshold of the 5-segment histogram based on the used sensor. The related registers do not need to be configured frequently because the **bChange** parameter specifies whether the returned values need to be configured again.
- The exposure time (**au32IntTime**) in the unit of line can be converted into that in the unit of μs by using **u32LinesPer500ms** in **cmos.c**. The conversion relationship is as follows:

  au32IntTime[0] = {[(HI_U64)au32IntTimeRst[0] x 1024 − u32Offset] x 500000/pstAeSnsDft->u32LinesPer500ms} >> 10

  In the preceding formula, **au32IntTimeRst[0]** indicates the exposure time in the unit of line. **u32Offset** is calculated as follows: u32Offset = f32Offset x 1024. **f32Offset** indicates the exposure time offset. For details, see the description of AE_ACCURACY_S.

[See Also]

ISP_AE_EXP_FUNC_S

ISP_AWB_REGISTER_S

[Description]

Defines AWB registration information.

[Syntax]

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

[Member]

Member	Description
stAwbExpFunc	Callback function for registering the AWB algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

ISP_AWB_EXP_FUNC_S

ISP_AWB_EXP_FUNC_S

[Description]

Defines the AWB callback function.

[Syntax]

```
typedef struct hiISP_AWB_EXP_FUNC_S
{
    HI_S32 (*pfn_awb_init)(HI_S32 s32Handle, const ISP_AWB_PARAM_S
*pstAwbParam);
    HI_S32 (*pfn_awb_run)(HI_S32 s32Handle,
        const ISP_AWB_INFO_S *pstAwbInfo,
        ISP_AWB_RESULT_S *pstAwbResult,
        HI_S32 s32Rsv);
    HI_S32 (*pfn_awb_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);
    HI_S32 (*pfn_awb_exit)(HI_S32 s32Handle);
} ISP_AWB_EXP_FUNC_S;
```

[Member]

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

[Note]

- **pfn_awb_init** is called when HI_MPI_ISP_Init is called to initialize the AWB algorithm library.
- **pfn_awb_run** is called when HI_MPI_ISP_Run is called to run the AWB algorithm library and calculate the white balance gain and color correction matrix (CCM).
- When the ISP runs, the control unit implicitly calls **pfn_awb_ctrl** to prompt the AWB algorithm library to switch the WDR or linear mode and set the ISO and exposure time (in μs). The ISO is related to the saturation. The chrominance noises are large when the gain is large. The saturation needs to be adjusted to set the ISO. The exposure time is set to assist indoor/outdoor judgment.
  The following is the current ctrl command defined by the firmware:

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

- **pfn_awb_exit** is called when HI_MPI_ISP_Exit is called to destroy the AWB algorithm library.

[See Also]

ISP_AWB_REGISTER_S

ISP_AWB_PARAM_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AWB_PARAM_S
{
```

```
    SENSOR_ID SensorId;
    HI_U8 u8WDRMode;
    HI_S32 s32Rsv;
} ISP_AWB_PARAM_S;
```

[Member]

Member	Description
SensorId	ID of the sensor that is registered with the ISP library. This ID is used to check whether the sensor registered with the ISP library is the same as the one registered with the AWB algorithm library.
u8WDRMode	WDR mode. The ISP provides the WDR mode information for the AWB.
s32Rsv	Reserved

[Note]

None

[See Also]

ISP_AWB_EXP_FUNC_S

ISP_AWB_INFO_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AWB_INFO_S
{
    HI_U32  u32FrameCnt;    /* the counting of frame */
    ISP_AWB_STAT_1_S *pstAwbStat1;
    ISP_AWB_STAT_2_S *pstAwbStat2;
    ISP_AWB_STAT_3_S *pstAwbStat3;
    ISP_AWB_STAT_4_S *pstAwbStat4;
} ISP_AWB_INFO_S;
```

[Member]

Member	Sub Member	Description
u32FrameCnt	None	Total number of frames Value range: [0, 0xFFFFFFFF]
pstAwbStat1	u16MeteringAwbRg	Ratio of the average G value to the average R value of white points in RGB field statistics

Member	Sub Member	Description
		Value range: [0, 0xFFFF]
	u16MeteringAwbBg	Ratio of the average G value to the average B value of white points in RGB field statistics Value range: [0, 0xFFFF]
	u32MeteringAwbSum	Number of white points in RGB field statistics Value range: [0, 0xFFFFFFFF]
pstAwbStat2	au16MeteringMemArrayRg	Ratio of the average G value to the average R value of white points in zoned statistics for the RGB field Value range: [0, 0xFFFF]
	au16MeteringMemArrayBg	Ratio of the average G value to the average B value of white points in zoned statistics for the RGB field Value range: [0, 0xFFFF]
	au16MeteringMemArraySum	Number of white points in zoned statistics for the RGB field Value range: [0, 0xFFFFFFFF]
pstAwbStat3	u16MeteringAwbAvgR	Average R value of white points in global statistics for the Bayer field Value range: [0, 0xFFFF]
	u16MeteringAwbAvgG	Average G value of white points in global statistics for the Bayer field Value range: [0, 0xFFFF]
	u16MeteringAwbAvgB	Average B value of white points in global statistics for the Bayer field Value range: [0, 0xFFFF]
	u16MeteringAwbCountAll	Number of white points in global statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]
	u16MeteringAwbCountMin	Number of pixels whose values are less than **BlackLevel** in global statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]
	u16MeteringAwbCountMax	Number of pixels whose values are greater than **WhiteLevel** in global statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]
pstAwbStat4	au16MeteringMemArrayAv	Average R value of white points in zoned

Member	Sub Member	Description
	gR	statistics for the Bayer field Value range: [0, 0xFFFF]
	au16MeteringMemArrayAvgG	Average G value of white points in zoned statistics for the Bayer field Value range: [0, 0xFFFF]
	au16MeteringMemArrayAvgB	Average B value of white points in zoned statistics for the Bayer field Value range: [0, 0xFFFF]
	au16MeteringMemArrayCountAll	Number of white points in zoned statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]
	au16MeteringMemArrayCountMin	Number of pixels whose values are less than **BlackLevel** in zoned statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]
	au16MeteringMemArrayCountMax	Number of pixels whose values are greater than **WhiteLevel** in zoned statistics for the Bayer field, which has been normalized Value range: [0, 0xFFFF]

[Note]

- The operation frequency (for example, once per two frames) of the AWB algorithm library can be controlled by setting **u32FrameCnt**.
- For details about the definitions of **Rg**, **Bg**, and **Sum**, see chapter 4 "AWB."
- Statistics are provided in four ways: global statistics for the RGB field, statistics with 15 x 17 zones for the RGB field (separate statistics for each zone), global statistics for the Bayer field, and statistics with 15 x 17 zones for the Bayer field (separate statistics for each zone).

[See Also]

ISP_AWB_EXP_FUNC_S

ISP_AWB_RESULT_S

[Description]

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

[Syntax]

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

```
    HI_U16  au16ColorMatrix[9];
    ISP_AWB_STAT_ATTR_S stStatAttr;
    ISP_AWB_RAW_STAT_ATTR_S stRawStatAttr;
} ISP_AWB_RESULT_S;
```

[Member]

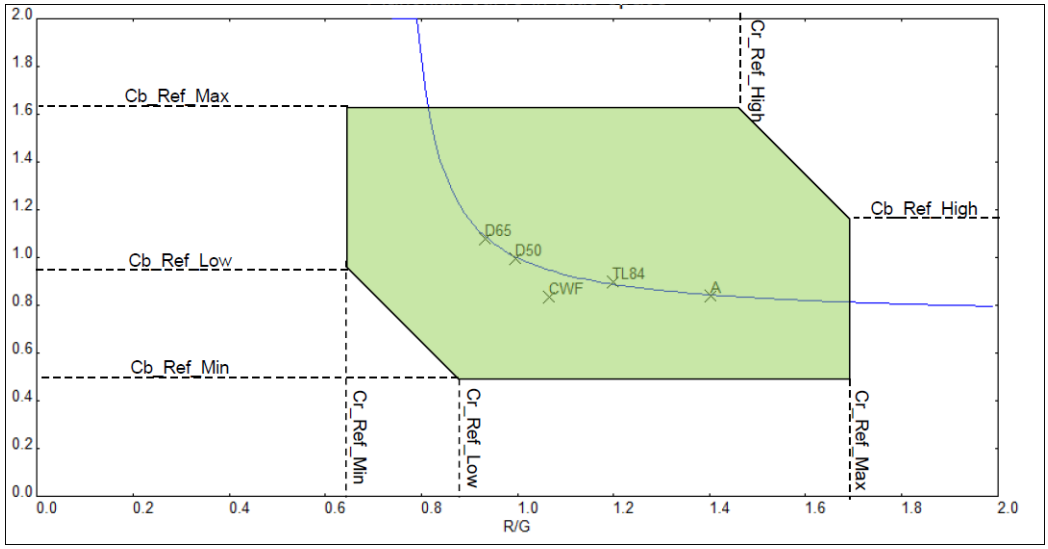
Member	Sub Member	Description
au32WhiteBalanceGain	None	Gain of the R, Gr, Gb, and B color channels obtained by using the AWB algorithms (16-bit precision)
au16ColorMatrix	None	CCM (8-bit precision)
stStatAttr	bChange	Whether the sub members of **stStatAttr** need to be configured again
	u16MeteringWhiteLevelAwb	Upper luminance limit for searching for white points during white point statistics for the RGB field Value range: [0x0, 0x3FF] Default value: 0x3ac
	u16MeteringBlackLevelAwb	Lower luminance limit for searching for white points during white point statistics for the RGB field Value range: [0x0, 0x3FF] Default value: 0x40
	u16MeteringCrRefMaxAwb	Maximum R/G color difference during white point statistics for the RGB field (8-bit precision) Default value: 512
	u16MeteringCbRefMaxAwb	Maximum B/G color difference during white point statistics for the RGB field (8-bit precision) Default value: 512
	u16MeteringCrRefMinAwb	Minimum R/G color difference during white point statistics for the RGB field (8-bit precision) Default value: 128
	u16MeteringCbRefMinAwb	Minimum B/G color difference during white point

Member	Sub Member	Description
		statistics for the RGB field (8-bit precision) Default value: 128
	u16MeteringCrRefHighAwb	Cr value (8-bit precision) corresponding to CbMax limited in the white point area in the hexagon during white point statistics for the RGB field Default value: 512
	u16MeteringCrRefLowAwb	Cr value (8-bit precision) corresponding to CbMin limited in the white point area in the hexagon during white point statistics for the RGB field Default value: 128
	u16MeteringCbRefHighAwb	Cb value (8-bit precision) corresponding to CrMax limited in the white point area in the hexagon during white point statistics for the RGB field Default value: 512
	u16MeteringCbRefLowAwb	Cb value (8-bit precision) corresponding to CrMin limited in the white point area in the hexagon during white point statistics for the RGB field Default value: 128
stRawStatAttr	bChange	Whether the sub members of **stRawStatAttr** need to be configured again
	u16MeteringWhiteLevelAwb	Upper luminance limit for searching for white points during white point statistics for the Bayer field Value range: [0x0, 0xFFF] Default value: 0xFFF
	u16MeteringBlackLevelAwb	Lower luminance limit for searching for white points during white point statistics for the Bayer field Value range: [0x0, 0xFFF]

Member	Sub Member	Description
		Default value: 0x0
	u16MeteringCrRefMaxAwb	Maximum R/G color difference during white point statistics for the Bayer field (8-bit precision) Default value: 512
	u16MeteringCbRefMaxAwb	Maximum B/G color difference during white point statistics for the Bayer field (8-bit precision) Default value: 512
	u16MeteringCrRefMinAwb	Minimum R/G color difference during white point statistics for the Bayer field (8-bit precision) Default value: 128
	u16MeteringCbRefMinAwb	Minimum B/G color difference during white point statistics for the Bayer field (8-bit precision) Default value: 128
	u16MeteringCrRefHighAwb	Cr value (8-bit precision) corresponding to CbMax limited in the white point area in the hexagon during white point statistics for the Bayer field Default value: 512
	u16MeteringCrRefLowAwb	Cr value (8-bit precision) corresponding to CbMin limited in the white point area in the hexagon during white point statistics for the Bayer field Default value: 128
	u16MeteringCbRefHighAwb	Cb value (8-bit precision) corresponding to CrMax limited in the white point area in the hexagon during white point statistics for the Bayer field Default value: 512
	u16MeteringCbRefLowAwb	Cb value (8-bit precision) corresponding to CrMin limited in the white point area

Member	Sub Member	Description
		in the hexagon during white point statistics for the Bayer field Default value: 128

Figure 2-9 Parameters related to white point area selection



[Note]

- The AWB algorithm library calculates the gains of the R, Gr, Gb, and B color channels to obtain the corrected white color. The 16-bit precision indicates the last 16 bits are decimals.
- In WDR mode, images are placed in a non-linear space because of GammaFe. Therefore, the square root is extracted from the return value of the AWB algorithm library and then configured to registers. If you develop a new AWB algorithm, the ISP control unit extracts the square root when configuring registers. The AWB algorithm library only needs to return correct 16-bit gains of four color channels.
- The AWB algorithm library also calculates a 3 x 3 CCM to reproduce actual colors. The 8-bit precision indicates that the last eight bits are decimals.
- The information in the **stStatAttr** data structure determines the pixels that are considered as white points for statistics. You can use the default values of **stStatAttr** or customize **stStatAttr** when you develop a new AWB algorithm. The **bChange** parameter indicates that whether the values of **stStatAttr** need to be configured to registers in the current frame.
- The ambient color temperature and illumination affect the distribution range of white points. When the HiSilicon AWB algorithms are running, the white point parameters in the AWB statistics for the Bayer field are automatically updated based on the environment parameters. If you want to modify the AWB statistics parameters when the HiSilicon AWB algorithms are running, you need to call HI_MPI_ISP_SetWBAttr to disable the automatic update function of the statistics parameters.

[See Also]

ISP_AWB_EXP_FUNC_S

ISP_AF_REGISTER_S

[Description]

Defines AF registration information.

[Syntax]

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

[Member]

Member	Description
stAfExpFunc	Callback function for registering the AF algorithm library

[Note]

This data structure is encapsulated for extension.

[See Also]

ISP_AF_EXP_FUNC_S

ISP_AF_EXP_FUNC_S

[Description]

Defines the AF callback function.

[Syntax]

```
typedef struct hiISP_AF_EXP_FUNC_S
{
   HI_S32 (*pfn_af_init)(HI_S32 s32Handle, const ISP_AF_PARAM_S
*pstAfParam);
   HI_S32 (*pfn_af_run)(HI_S32 s32Handle,
       const ISP_AF_INFO_S *pstAfInfo,
       ISP_AF_RESULT_S *pstAfResult,
       HI_S32 s32Rsv);
   HI_S32 (*pfn_af_ctrl)(HI_S32 s32Handle, HI_U32 u32Cmd, HI_VOID *pValue);
   HI_S32 (*pfn_af_exit)(HI_S32 s32Handle);
} ISP_AF_EXP_FUNC_S;
```

[Member]

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

[Note]

The AF algorithm library is not implemented currently.

[See Also]

ISP_AF_REGISTER_S

ISP_AF_PARAM_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AF_PARAM_S
{
    SENSOR_ID SensorId;
    HI_S32 s32Rsv;
} ISP_AF_PARAM_S;
```

[Member]

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

[Note]

None

[See Also]

ISP_AF_EXP_FUNC_S

ISP_AF_INFO_S

[Description]

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

[Syntax]

```
typedef struct hiISP_AF_INFO_S
{
    HI_U32  u32FrameCnt;    /* the counting of frame */
    ISP_AF_STAT_S *pstAfStat;
} ISP_AF_INFO_S;
```

[Member]

Member	Sub Member	Description
u32FrameCnt	None	Total number of frames Value range: [0, 0xFFFFFFFF]
stAfStat	stZoneMetrics. u16v1	Statistics of the FIR filter for the AF odd columns in zoned statistics
	stZoneMetrics. u16h1	Statistics of the IIR filter for the AF odd rows in zoned statistics
	stZoneMetrics. u16v2	Statistics of the FIR filter for the AF even columns in zoned statistics
	stZoneMetrics. u16h2	Statistics of the IIR filter for the AF even rows in zoned statistics
	stZoneMetrics. u16y	AF luminance statistics in zoned statistics (accumulated luminance of the pixels in zones)

[Note]

None

[See Also]

ISP_AF_EXP_FUNC_S

ISP_AF_RESULT_S

[Description]

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

[Syntax]

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

```
} ISP_AF_RESULT_S;
```

[Member]

Member	Description
s32Rsv	Reserved

[Note]

None

[See Also]

ISP_AF_EXP_FUNC_S

ISP_INNER_STATE_INFO_S

[Description]

Defines the structure of the ISP status information.

[Syntax]

```
typedef struct hiISP_INNER_STATE_INFO_S
{
    HI_U32 u32DRCStrengthActual;
    HI_U32 u32NRStrengthActual;
    HI_U32 u32SharpenStrengthDActual;
    HI_U32 u32SharpenStrengthUdActual;
    HI_U32 u32SharpenStrengthRGBActual;
    HI_U32 u32DefogStrengthActual;
    HI_U32 u32WDRExpRatioActual;
    HI_U32 au32BLActual[4];
    HI_BOOL bWDRSwitchFinish;
    HI_BOOL bResSwitchFinish;
} ISP_INNER_STATE_INFO_S;
```

[Member]

Member	Description
u32DRCStrengthActual	DRC strength that takes effect
u32NRStrengthActual	2DNR strength that takes effect
u32SharpenStrengthDActual	SharpenD strength that takes effect
u32SharpenStrengthUdActual	SharpenUd strength that takes effect
u32SharpenStrengthRGBActual	RGB sharpen strength that takes effect
u32DefogStrengthActual	Anti-fog strength that takes effect

Member	Description
u32WDRExpRatioActual	Exposure ratio that takes effect
au32BLActual[4]	Black level that takes effect
bWDRSwitchFinish	Completion identifier for switching between the WDR mode and the linear mode. The value HI_TRUE indicates that the switching is complete.
bResSwitchFinish	Completion identifier for resolution switching. The value HI_TRUE indicates that the switching is complete.

[Note]

None

[See Also]

None

ISP_DCF_INFO_S

[Description]

Defines the DCF information.

[Syntax]

```
#define DCF_DRSCRIPTION_LENGTH     32
typedef struct hiISP_DCF_INFO_S
{
    HI_U8       au8ImageDescription[DCF_DRSCRIPTION_LENGTH];
    HI_U8       au8Make[DCF_DRSCRIPTION_LENGTH];
    HI_U8       au8Model[DCF_DRSCRIPTION_LENGTH];
    HI_U8       au8Software[DCF_DRSCRIPTION_LENGTH];
    HI_U16      u16ISOSpeedRatings;
    HI_U32      u32FNumber;
    HI_U32      u32MaxApertureValue;
    HI_U32      u32ExposureTime;
    HI_U32      u32ExposureBiasValue;
    HI_U8       u8ExposureProgram;
    HI_U8       u8MeteringMode;
    HI_U8       u8LightSource;
    HI_U32      u32FocalLength;
    HI_U8       u8SceneType;
    HI_U8       u8CustomRendered;
    HI_U8       u8ExposureMode;
    HI_U8       u8WhiteBalance;
    HI_U8       u8FocalLengthIn35mmFilm;
```

```
    HI_U8       u8SceneCaptureType;
    HI_U8       u8GainControl;
    HI_U8       u8Contrast;
    HI_U8       u8Saturation;
    HI_U8       u8Sharpness;
}
```

[Member]

Member	Description
au8ImageDescription	Image description and source, that is, the tool used to generate the image The data format is ASCII strings and the maximum length is 32.
au8Make	Manufacturer that produces the product The data format is ASCII strings and the maximum length is 32.
au8Model	Device model The data format is ASCII strings and the maximum length is 32.
au8Software	Software that shows the firmware version The data format is ASCII strings and the maximum length is 32.
u16ISOSpeedRatings	Photosensibility The data format is unsigned short. This member is read-only.
u32FNumber	Aperture coefficient The data format is unsigned rational. The upper 16 bits are the numerator, and the lower 16 bits are the denominator. This member is read-only.
u32MaxApertureValue	Maximum aperture value of the lens The data format is unsigned rational. The upper 16 bits are the numerator, and the lower 16 bits are the denominator. This member is read-only.
u32ExposureTime	Exposure time (reciprocal of the shutter speed). The unit is second. The data format is unsigned rational. The upper 16 bits are the numerator, and the lower 16 bits are the denominator. This member is read-only.
u32ExposureBiasValue	Exposure compensation during photographing. The

Member	Description
	unit is APEX (EV) The data format is unsigned rational. The upper 16 bits are the numerator, and the lower 16 bits are the denominator. This member is read-only.
u8ExposureProgram	Exposure program used by the camera during photographing 1: manual exposure 2: normal program exposure 3: aperture-first exposure 4: shutter-first exposure 5: creative program (low-speed program) 6: motion program (high-speed program) 7: portrait mode 8: landscape mode The data format is unsigned short. This member is read-only.
u8MeteringMode	Light metering mode of exposure 0: unknown 1: average metering 2: center-weighted metering 3: spot metering 4: multi-spot metering 5: multi-zone metering 6: partial metering 255: others The data format is unsigned short. This member is read-only.
u8LightSource	Light source, indicating the white balance setting 0: unknown 1: sunlight 2: fluorescent lamp 3: incandescent lamp (tungsten filament) 10: flash lamp 17: standard light A 18: standard light B 19: standard light C 20: D55 21: D65 22: D75

Member	Description
	255: others The data format is unsigned short.
u32FocalLength	Focal length of the lens during photographing. The unit is mm. The data format is unsigned rational. The upper 16 bits are the numerator, and the lower 16 bits are the denominator.
u8SceneType	Type of the photographing scenario. The value 0x01 indicates that the picture is directly taken by using the camera. The data format is unsigned short. Not supported currently
u8CustomRendered	Customized picture processing 0: standard 1: customized picture processing
u8ExposureMode	Exposure mode 0: automatic exposure 1: manual exposure 2: automatic exposure bracketing (AEB) The data format is unsigned short. This member is read-only.
u8WhiteBalance	White balance 0: AWB 1: manual white balance The data format is unsigned short. This member is read-only.
u8FocalLengthIn35mmFilm	35 mm film focal length. The value 0 indicates that the focal length does not exist. The data format is unsigned short.
u8SceneCaptureType	Scenario photographing type 0: standard mode 1: landscape mode 2: portrait mode 3: night mode The data format is unsigned short.
u8GainControl	Gain control 0: none 1: low gain up 2: high gain up 3: low gain down

Member	Description
	4: high gain down The data format is unsigned short.
u8Contrast	Contrast The data format is unsigned short.
u8Saturation	Saturation 0: none 1: low 2: high The data format is unsigned short.
u8Sharpness	Sharpness The data format is unsigned short.

[Note]

None

[See Also]

- HI_MPI_ISP_SetDCFInfo
- HI_MPI_ISP_GetDCFInfo

ISP_MOD_PARAM_S

[Description]

Defines the structure of the ISP module parameters.

[Syntax]

```
typedef struct hiISP_MOD_PARAM_S
{
    HI_U32      proc_param;
}ISP_MOD_PARAM_S;
```

[Member]

Member	Description
proc_param	Update frequency of the proc information for ISP Value range: [0, +∞) When **proc_param** is *n*, the proc information of ISP is updated once every *n* frames.

[Note]

- When the .ko files are loaded, if the ISP module parameter **proc_param** is set to **0**, the memory for storing the proc information of ISP is not allocated. If **proc_param** is set to a non-zero value, the memory for storing the proc information of ISP is allocated.
- When the .ko files are loaded and the ISP module parameter **proc_param** is set to **0**, **proc_param** set by calling HI_MPI_ISP_SetModParam must be 0.
- When the .ko files are loaded and the ISP module parameter **proc_param** is set to a non-zero value, **proc_param** set by calling HI_MPI_ISP_SetModParam can only be dynamically switched between non-zero values.
- Frequent update of the proc information for ISP occupies the CPU resource. You are advised to update the proc information once every 30 frames or enable proc information only during debugging.
- Huawei LiteOS does not support the kernel module loading mechanism. To achieve the effect similar to loading the .ko drivers in Linux, run **sdk_init.c** under **Huawei LiteOS release/ko**. In the ISP_init function in **sdk_init.c**, add **stIsp_Param.u32ProcParam=n** to ISP_ModInit to set **proc_param**.

[See Also]

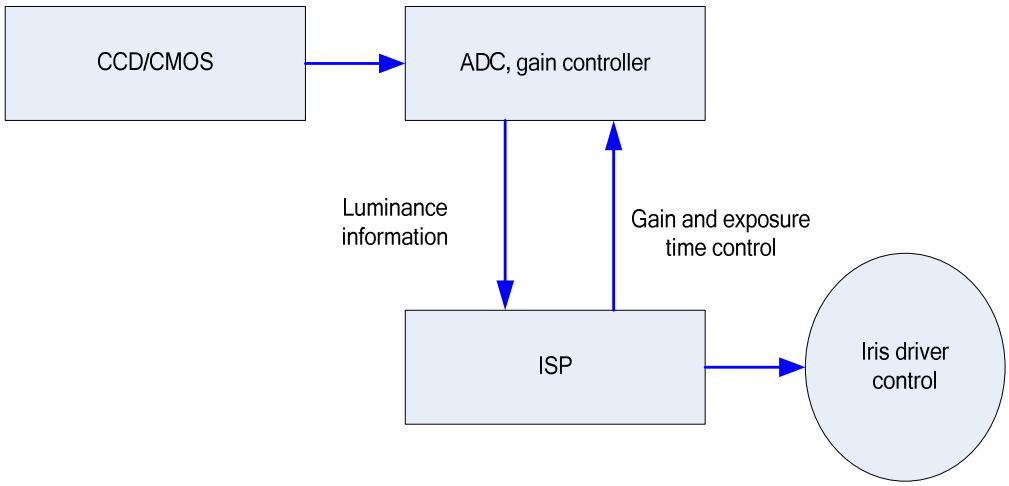
None

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 iris first algorithm adjusts the iris to the proper position first and then allocates the exposure time and gains, which applies only to the P-iris lens and balances the noise and depth of field. The shutter first algorithm allocates the exposure time first and then the sensor gain and ISP gain, which minimizes the noises in the taken images. The gain first algorithm allocates the sensor gain and ISP gain first and then the exposure time, which applies to the scenario of taking images of moving objects. You can also customize the exposure allocation policies of AE algorithms. 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: The iris changes the central hole size of the lens, and the mechanical shutter controls the exposure time.
- 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 include histograms and average values. The AE module can provide a 5-segment histogram, a 256-segment histogram, or average statistics values of the R, Gr, Gb, and B components for the entire image or divide the entire image into M x N zones and then provide a 5-segment histogram or average statistics values of the R, Gr, Gb, and B components for each zone. See Figure 3-2 and Figure 3-3.

Figure 3-2 5-segment AE statistics histogram

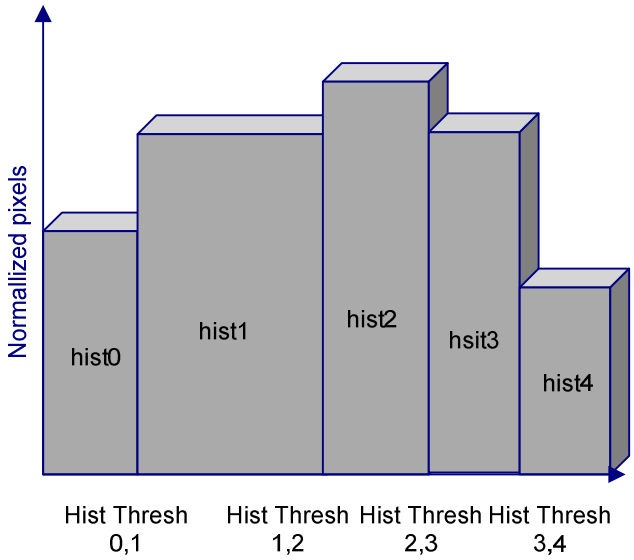
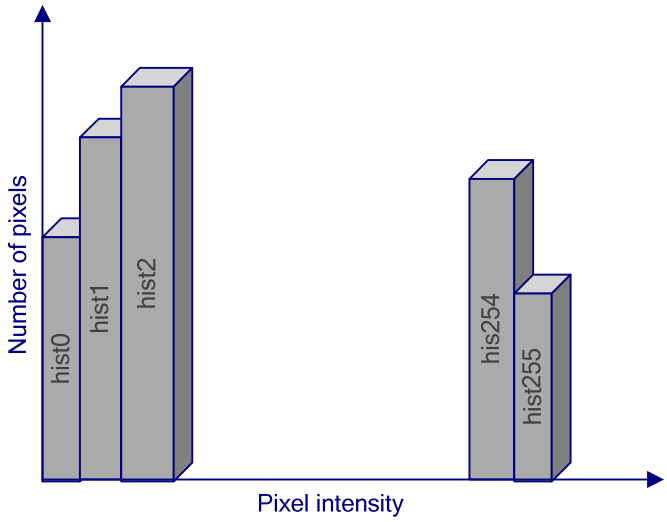
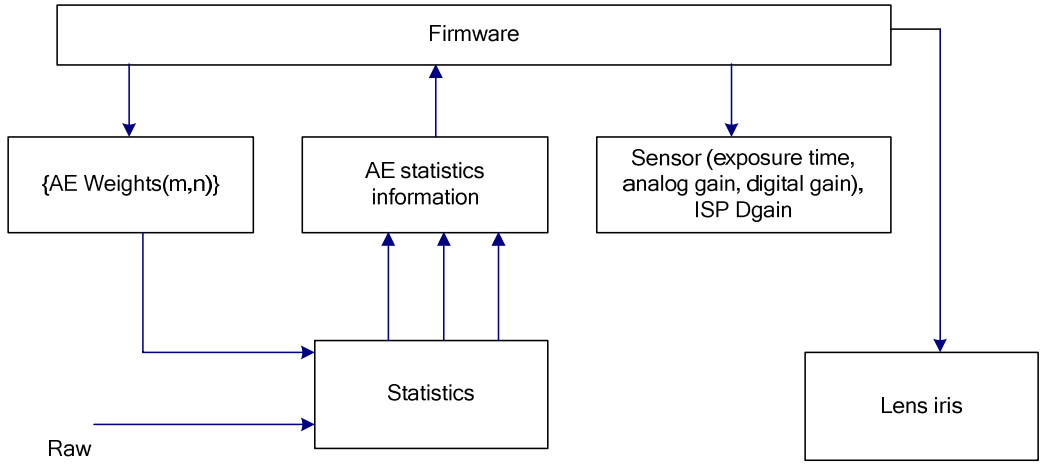


Figure 3-3 256-segment AE statistics histogram



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

Figure 3-4 AE working principle



3.4 API Reference

3.4.1 AE Algorithm Library MPIs

All the AE algorithm library MPIs are based on the HiSilicon AE algorithm library. These MPIs can be ignored and are unavailable if you use your own AE algorithm library.

- HI_MPI_AE_Register: Registers the AE algorithm library with the ISP.
- HI_MPI_AE_UnRegister: Deregisters the AE algorithm library from the ISP.
- HI_MPI_AE_SensorRegCallBack: Registers a sensor. This callback function is provided by the AE algorithm library.
- HI_MPI_AE_SensorUnRegCallBack: Deregisters a sensor. This callback function is provided by the AE algorithm library.

HI_MPI_AE_Register

[Description]

Registers the AE algorithm library with the ISP.

[Syntax]

```
HI_S32 HI_MPI_AE_Register(ISP_DEV IspDev, ALG_LIB_S *pstAeLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAeLib	Pointer to the data structure of the AE algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

- This MPI calls the AE registration callback function HI_MPI_ISP_AeLibRegCallBack provided by the ISP library to register the AE algorithm library with the ISP library.
- Multiple instances can be registered with the AE algorithm library.

[Example]

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

```
HI_MPI_AE_Register(IspDev, &stAeLib);
```

[See Also]

None

HI_MPI_AE_UnRegister

[Description]

Deregisters the AE algorithm library from the ISP.

[Syntax]

```
HI_S32 HI_MPI_AE_UnRegister(ISP_DEV IspDev, ALG_LIB_S *pstAeLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAeLib	Pointer to the data structure of the AE algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

This MPI calls the AE deregistration callback function **HI_MPI_ISP_AELibUnRegCallBack** provided by the ISP library to deregister the AE algorithm library from the ISP library.

[Example]

```
ISP_DEV IspDev = 0;
stAeLib.s32Id = 0;
strcpy(stAeLib.acLibName, HI_AE_LIB_NAME);
HI_MPI_AE_UnRegister(IspDev, &stAeLib);
stAeLib.s32Id = 1;
HI_MPI_AE_UnRegister(IspDev, &stAeLib);
```

[See Also]

None

HI_MPI_AE_SensorRegCallBack

[Description]

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

[Syntax]

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

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

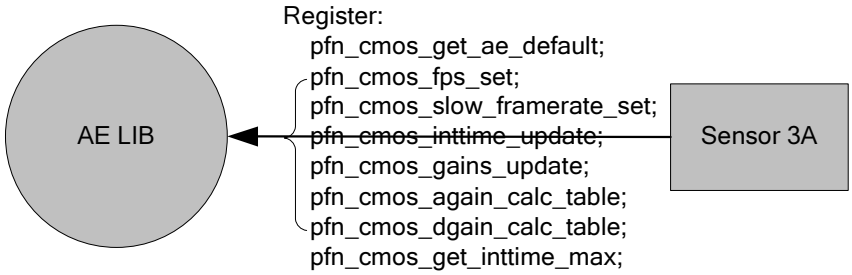
[Requirement]

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

[Requirement]

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

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



[Example]

```
ALG_LIB_S stLib;
AE_SENSOR_REGISTER_S  stAeRegister;
AE_SENSOR_EXP_FUNC_S *pstExpFuncs = &stAeRegister.stSnsExp;
memset(pstExpFuncs, 0, sizeof(AE_SENSOR_EXP_FUNC_S));
pstExpFuncs->pfn_cmos_get_ae_default    = cmos_get_ae_default;
pstExpFuncs->pfn_cmos_fps_set          = cmos_fps_set;
pstExpFuncs->pfn_cmos_slow_framerate_set= cmos_slow_framerate_set;
pstExpFuncs->pfn_cmos_inttime_update    = cmos_inttime_update;
pstExpFuncs->pfn_cmos_gains_update     = cmos_gains_update;
pstExpFuncs->pfn_cmos_again_calc_table  = cmos_again_calc_table;
pstExpFuncs->pfn_cmos_dgain_calc_table  = cmos_dgain_calc_table;
pstExpFuncs->pfn_cmos_get_inttime_max   = cmos_get_inttime_max;
pstExpFuncs->pfn_cmos_ae_fswdr_attr_set = cmos_ae_fswdr_attr_set;
ISP_DEV IspDev = 0;
stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AE_LIB_NAME);
s32Ret = HI_MPI_AE_SensorRegCallBack(IspDev, &stLib, IMX104_ID,
&stAeRegister);
if (s32Ret)
{
    printf("sensor register callback function to ae lib failed!\n");
    return s32Ret;
}
```

[See Also]

None

HI_MPI_AE_SensorUnRegCallBack

[Description]

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

[Syntax]

```
HI_S32 HI_MPI_AE_SensorUnRegCallBack(ISP_DEV IspDev, ALG_LIB_S *pstAeLib,
SENSOR_ID SensorId);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAeLib	Pointer to the data structure of the AE algorithm library	Input
SensorId	ID of the sensor that is deregistered from the AE algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

SensorId can be configured in the sensor library and is used to ensure that the sensor deregistered from the ISP library is the same as that deregistered from the 3A algorithm library.

[Example]

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

[See Also]

None

3.4.2 AE Control MPIs

The following are AE control MPIs:

- HI_MPI_ISP_SetExposureAttr: Sets the AE attribute.
- HI_MPI_ISP_GetExposureAttr: Obtains the AE attribute.
- HI_MPI_ISP_SetWDRExposureAttr: Sets the AE exposure attribute in WDR mode.
- HI_MPI_ISP_GetWDRExposureAttr: Obtains the AE exposure attribute in WDR mode.
- HI_MPI_ISP_SetAERouteAttr: Sets the AE route policy.
- HI_MPI_ISP_GetAERouteAttr: Obtains the AE route policy.
- HI_MPI_ISP_SetAERouteAttrEx: Sets the extended attribute of the AE allocation policy.
- HI_MPI_ISP_GetAERouteAttrEx: Obtains the extended attribute of the AE allocation policy.
- HI_MPI_ISP_QueryExposureInfo: Obtains the internal status information.

HI_MPI_ISP_SetExposureAttr

[Description]

Sets the AE attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetExposureAttr(ISP_DEV IspDev, const ISP_EXPOSURE_ATTR_S
*pstExpAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstExpAttr	Pointer to the AE attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

- When the exposure control mode is automatic mode, the exposure time and exposure gain are automatically controlled by the AE algorithm. The exposure effect can be customized by configuring the parameters in the AE attribute data structure stAuto.
- When the exposure control mode is manual mode, the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain can be enabled, and exposure parameters (including the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain) can be set by configuring the manual exposure (ME) attribute data structure stManual.
- When the exposure control mode is automatic mode, configuring ME attribute parameters has no effect. When the exposure control mode is manual mode, configuring AE attribute parameters has no effect.
- When the exposure control mode is manual mode, if the value of an exposure parameter is greater than the maximum value or less than the minimum value, the maximum value or the minimum value supported by the sensor is used.
- In both automatic and manual exposure control modes, the unit of the exposure time is μs, and the exposure gain is a multiple of the 10-bit precision. For example, the value 1024 indicates 1x gain, and the value 2048 indicates 2x gain.

[Example]

Configure AE attributes:

```
ISP_DEV IspDev = 0;
ISP_EXPOSURE_ATTR_S stExpAttr;

HI_MPI_ISP_GetExposureAttr(IspDev, &stExpAttr);

stExpAttr.bByPass = HI_FALSE;
stExpAttr.enOpType = OP_TYPE_AUTO;
stExpAttr.stAuto.stExpTimeRange.u32Max = 40000;
stExpAttr.stAuto.stExpTimeRange.u32Min = 10;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);

stExpAttr.stAuto.u8Speed = 0x80;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);

stExpAttr.stAuto.enAEStrategyMode = AE_EXP_HIGHLIGHT_PRIOR;
stExpAttr.stAuto.u16HistRatioSlope = 0x100;
stExpAttr.stAuto.u8MaxHistOffset = 0x40;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);

stExpAttr.stAuto.stAntiflicker.bEnable = HI_TRUE;
stExpAttr.stAuto.stAntiflicker.u8Frequency = 50;
```

```
stExpAttr.stAuto.stAntiflicker.enMode = ISP_ANTIFLICKER_NORMAL_MODE;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);

stExpAttr.stAuto.stAEDelayAttr.u16BlackDelayFrame = 10;
stExpAttr.stAuto.stAEDelayAttr.u16WhiteDelayFrame = 0;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);

HI_U8 i,j;
HI_U8 u8Weighttable[15][17]={{1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                             };
 for (i = 0; i < AE_ZONE_ROW; i++)
 {
     for (j = 0; j < AE_ZONE_COLUMN; j++)
     {
         stExpAttr.stAuto.au8Weight[i][j] = u8Weighttable[i][j];
     }
}
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);
```

Configure ME attributes:

```
ISP_DEV IspDev = 0;
ISP_EXPOSURE_ATTR_S stExpAttr;

HI_MPI_ISP_GetExposureAttr(IspDev, &stExpAttr);
stExpAttr.bByPass = HI_FALSE;
stExpAttr.enOpType = OP_TYPE_MANUAL;
stExpAttr.stManual.enAGainOpType = OP_TYPE_MANUAL;
stExpAttr.stManual.enDGainOpType = OP_TYPE_MANUAL;
stExpAttr.stManual.enISPDGainOpType = OP_TYPE_MANUAL;
```

```
stExpAttr.stManual.enExpTimeOpType = OP_TYPE_MANUAL;
stExpAttr.stManual.u32AGain = 0x400;
stExpAttr.stManual.u32DGain = 0x400;
stExpAttr.stManual.u32ISPDGain = 0x400;
stExpAttr.stManual.u32ExpTime= 0x40000;
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);
```

[See Also]

HI_MPI_ISP_GetExposureAttr

HI_MPI_ISP_GetExposureAttr

[Description]

Obtains the AE attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetExposureAttr(ISP_DEV IspDev, ISP_EXPOSURE_ATTR_S
*pstExpAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstExpAttr	Pointer to the AE attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetExposureAttr

HI_MPI_ISP_SetWDRExposureAttr

[Description]

Sets the AE exposure attribute in WDR mode.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWDRExposureAttr(ISP_DEV IspDev, const
ISP_WDR_EXPOSURE_ATTR_S *pstWDRExpAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstWDRExpAttr	Pointer to the AE exposure attribute in WDR mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_GetWDRExposureAttr

HI_MPI_ISP_GetWDRExposureAttr

[Description]

Obtains the AE exposure attribute in WDR mode.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWDRExposureAttr(ISP_DEV IspDev,
ISP_WDR_EXPOSURE_ATTR_S *pstWDRExpAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstWDRExpAttr	Pointer to the AE exposure attribute in WDR mode	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetWDRExposureAttr

HI_MPI_ISP_SetAERouteAttr

[Description]

Sets the AE route attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAERouteAttr(ISP_DEV IspDev, const ISP_AE_ROUTE_S
*pstAERouteAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAERouteAttr	Pointer to the data structure of the AE route attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

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

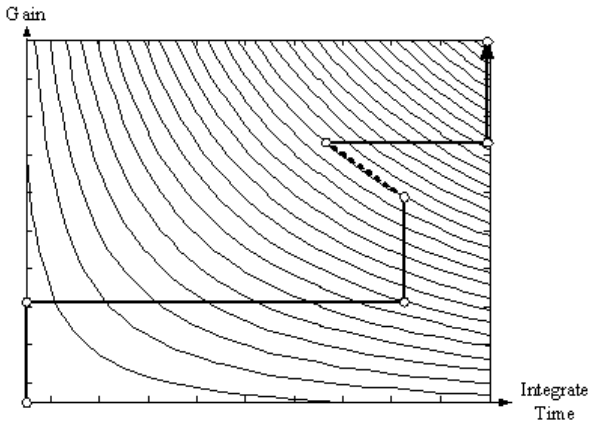
[Requirement]

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

[Note]

- This MPI is used to set AE allocation routes. The exposure calculated by the AE module is allocated based on the configured route. You can also set the exposure first, gain first, and shutter first modes.
- − shows the allocated AE routes. Note the following limitations when allocating AE routes:
  - A maximum of 16 nodes are supported. Each node has the exposure time, gain, and iris components. The gain includes analog gain, digital gain, and ISP digital gain.
  - Only the P iris is supported. The DC iris is not supported because it cannot be accurately controlled. As a result, the iris components of the DC iris and manual iris are invalid. That is, when the iris type is DC iris, the node iris component has no effect on exposure allocation.
  - The node exposure is the product of the exposure time, gain, and iris. The node exposure is monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest.
  - If the exposure of an adjacent node increases, the value of a component should increase while the values of other components are retained. The component with increased value determines the allocation policy of the route. For example, if the value of the gain component increases, the allocation policy of the route is gain first.
  - The exposure of adjacent nodes can be the same. In this case, an allocation mode can be switched to another one. It is recommended that the nodes have different exposure.
  - You can set routes based on the application scenario. The allocation route can be dynamically switched.
  - According to the default AE allocation policy for the DC iris and manual iris, the exposure time is allocated first, and then the gain is allocated. According to the default AE allocation policy for the P iris, the iris is adjusted to the maximum size, and then the exposure time and gain are allocated in sequence. If the current exposure does not fall within the configured route range, the default allocation policy is used.
  - When the DC iris and P iris are switched in online mode, the AE route is reset to the default allocation policy that matches the iris type. You can configure the AE route as required when switching the iris type.
  - If the maximum exposure time is changed, the changed value is updated in the AE allocation routes during automatic frame rate reduction.
  - During the switchover between the linear mode and the WDR mode, if the AE route is configured in **cmos.c**, the route in **cmos.c** is used after the switching. Otherwise, the default AE route is used.
  - During the frame rate switching or resolution switching, if the configured maximum target exposure time is greater than the maximum exposure time allowed by one frame after the switching, the maximum exposure time for the allocation route is changed to the maximum exposure time allowed by one frame after the switching.
  - During automatic frame rate reduction, switchover between the liner mode and WDR mode, frame rate or resolution switching, or when the maximum/minimum value of the exposure time or gain is limited, the actual AE route that takes effect may be different from that configured in the MPI. In this case, you can call HI_MPI_ISP_QueryExposureInfo to obtain the actual AE route that takes effect.

Figure 3-6 AE allocation routes



[Example]

```
ISP_DEV IspDev = 0;
ISP_AE_ROUTE_S stRoute;
HI_U32 au32RouteNode[3][3]
       = {{100,1024,1},{40000,1024,1},{40000,16384,1}};
HI_MPI_ISP_GetAERouteAttr(IspDev, &stRoute);
stRoute.u32TotalNum = 3;
memcpy(stRoute.astRouteNode, au32RouteNode, sizeof(au32RouteNode));
HI_MPI_ISP_SetAERouteAttr(IspDev, &stRoute);
```

[See Also]

- HI_MPI_ISP_GetAERouteAttr
- ISP_AE_ROUTE_S

HI_MPI_ISP_GetAERouteAttr

[Description]

Obtains the AE route attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAERouteAttr(ISP_DEV IspDev, ISP_AE_ROUTE_S
*pstAERouteAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAERouteAttr	Pointer to the data structure of the AE route attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetAERouteAttr

HI_MPI_ISP_SetAERouteAttrEx

[Description]

Sets the extended attribute of the AE allocation policy. The sensor analog gain, sensor digital gain, and ISP digital gain in the AE allocation policy can be separately configured.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAERouteAttrEx(ISP_DEV IspDev, const ISP_AE_ROUTE_EX_S
*pstAERouteAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAERouteAttrEx	Pointer to the extended attribute of AE allocation policy	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 files: libisp.a, lib_hiae.a

[Note]

- This MPI is used to set the extended attribute of the AE allocation policy. The exposure calculated by the AE algorithm is allocated based on the configured route. You can set the allocation priorities of the exposure time, sensor analog gain, sensor digital gain, ISP digital gain, and iris as required (for example, the exposure time is allocated first in exposure time first mode). This MPI can be used to configure the exposure allocation route in WDR mode to alleviate flicker at the power frequency in the multi-frame combination WDR mode under normal indoor illumination and optimize picture effect in WDR mode.
- You can choose whether to validate the extended attribute of the AE allocation policy by configuring **bAERouteExValid** of HI_MPI_ISP_SetExposureAttr. When **bAERouteExValid** is set to **HI_TRUE**, the extended AE route is used; otherwise, the common AE route is used.
- Note the following limitations when allocating the extended AE route:
  - At most 16 nodes are supported. Each node has five components, including the exposure time, sensor analog gain, sensor digital gain, ISP digital gain, and iris.
  - The unit of the exposure time for a node is μs. The exposure time cannot be set to 0. The exposure time cannot be too small; otherwise, the number of actual corresponding exposure lines is 0 and exceptions may occur.
  - The iris component is supported only when the P iris is used. The iris component is invalid when the DC iris or the manual iris is used because the DC iris cannot be precisely controlled. That is, when the iris type is DC iris, the node iris component has no effect on exposure allocation.
  - The node exposure is the product of the exposure time, sensor analog gain, sensor digital gain, ISP digital gain, and iris. The node exposure is monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest.
  - If the exposure of an adjacent node increases, the value of a component should increase while the values of other components remain unchanged. The

component with an increased value determines the allocation policy of the route. For example, if the value of the sensor analog gain component increases, the allocation policy of the route is sensor analog gain first (the sensor analog gain is allocated first).

- The exposure of adjacent nodes can be the same. In this case, an allocation mode can be switched to another one. It is recommended that the nodes have different exposure.
- You can set the route based on the application scenario. The allocation route can be dynamically switched.
- According to the default extended AE allocation policy for the DC iris and manual iris, the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain are allocated in sequence. According to the default extended AE allocation policy for the P iris, the iris is adjusted to the maximum size, and then the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain are allocated in sequence. If the current exposure does not fall within the configured route range, the default allocation policy is used.
- When the DC iris and P iris are switched in online mode, the extended AE route is reset to the default allocation policy that matches the iris type. You can configure the extended AE route as required when switching the iris type.
- If the maximum exposure time is changed during automatic frame rate reduction, the changed value is updated to the allocation route.
- During the switchover between the linear mode and WDR mode, if the extended AE route is configured in **cmos.c** and the identifier of the extended AE route (**bAERouteExValid**) in **cmos.c** is set to **HI_TRUE**, the extended AE route in **cmos.c** is used after mode switching. Otherwise, the default AE route is used.
- During the frame rate switching or resolution switching, if the configured maximum target exposure time is greater than the maximum exposure time allowed by one frame after switching, the maximum exposure time for the allocation route is changed to the maximum exposure time allowed by one frame after switching.
- During automatic frame rate reduction, switchover between the liner mode and WDR mode, frame rate or resolution switching, or when the maximum/minimum value of the exposure time or gain is limited, the actual extended AE route that takes effect may be different from that configured in the MPI. In this case, you can call HI_MPI_ISP_QueryExposureInfo to obtain the actual extended AE route that takes effect.

[Example]

```
ISP_DEV IspDev = 0;
ISP_EXPOSURE_ATTR_S stExpAttr;
ISP_AE_ROUTE_EX_S stRouteEx;
HI_U32 au32RouteExNode1[6][5]
       = {{   30,  1024,  1024, 1024, 0},
          {   30,  1024,  1024, 1024, 10},
          {   30, 16384,  1024, 1024, 10},
          {1000000, 16384,  1024, 1024, 10},
          {1000000, 16384, 16384, 1024, 10},
          {1000000, 16384, 16384, 4096, 10}};
HI_MPI_ISP_GetAERouteAttrEx(IspDev, &stRouteEx);
HI_MPI_ISP_GetExposureAttr(IspDev, &stExpAttr);
stExpAttr.bAERouteExValid = HI_TRUE;
```

```
stRouteEx.u32TotalNum = 6;
memcpy(stRouteEx.astRouteExNode, au32RouteExNode1,
sizeof(au32RouteExNode1));
HI_MPI_ISP_SetAERouteAttrEx(IspDev, &stRouteEx);
HI_MPI_ISP_SetExposureAttr(IspDev, &stExpAttr);
```

[See Also]

- HI_MPI_ISP_GetAERouteAttrEx
- HI_MPI_ISP_SetExposureAttr
- ISP_AE_ROUTE_S

HI_MPI_ISP_GetAERouteAttrEx

[Description]

Obtains the extended attribute of the AE allocation policy.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAERouteAttrEx(ISP_DEV IspDev, ISP_AE_ROUTE_EX_S
*pstAERouteAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAERouteAttrEx	Pointer to the extended attribute of AE allocation policy	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetAERouteAttrEx

HI_MPI_ISP_QueryExposureInfo

[Description]

Obtains AE internal status information (including the global 5-segment histogram, 256-segment histogram, and average luminance) and AE running status information (including the exposure time, gain, light exposure and the actual AE route that takes effect).

[Syntax]

```
HI_S32 HI_MPI_ISP_QueryExposureInfo(ISP_DEV IspDev, ISP_EXP_INFO_S
*pstExpInfo);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstExpInfo	Pointer to the data structure of AE internal status information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

- The unit of the obtained exposure time is μs. The obtained sensor analog gain, sensor digital gain, and ISP digital gain are measured by multiples, and their precision is 10 bits.
- The obtained light exposure is calculated as follows: Obtained light exposure = Exposure time x Exposure gain. The iris status is not considered. The exposure time is in the unit of line. The exposure gain includes the sensor analog gain, sensor digital gain, and ISP digital gain, and it is a multiple with 6-bit decimal precision. If the precision of the obtained light exposure fails to meet the requirements, you can recalculate an exposure based on the preceding high-precision exposure time (in μs) and gain (10-bit decimal precision).
- You can check whether AE is stable by querying **s16HistError**. If the absolute value of **s16HistError** is less than the exposure tolerance, AE is not performed currently.
- Both the AE route or extended AE route obtained by calling this MPI and the one in the proc information are the AE route or extended AE route that takes effect. The differences are as follows: the node exposure time obtained by calling this MPI is in the unit of μs, whereas the exposure time in the proc information is in the unit of line; the iris component obtained by calling this MPI is the F value and the value range is [0, 10], whereas the iris component in the proc information is the equivalent gain of the F value, and the value range is (1<<F value).

[Example]

```
ISP_DEV IspDev = 0;
ISP_EXP_INFO_S stExpInfo;
HI_MPI_ISP_QueryExposureInfo(IspDev, &stExpInfo);
printf("Sensor exposure time: %d\n",stExpInfo.u32ExpTime);
printf("Analog Gain: %d\n",stExpInfo.u32AGain);
printf("Digital Gain: %d\n",stExpInfo.u32DGain);
printf("ISP Gain: %d\n",stExpInfo.u32ISPDGain);
printf("Exposure: %d\n",stExpInfo.u32Exposure);
printf("Average Luminance: %d\n",stExpInfo.u8AveLum);
printf("Hist error: %d\n",stExpInfo.s16HistError);
stExpInfo.bExposureIsMAX ? printf("Exposure is MAX!\n") : printf("Exposure
is NOT MAX!\n");
for(i = 0; i < 5; i++)
{
printf("Hist5Value[%d]: %d\n",i,stExpInfo.u16AE_Hist5Value[i]);
}
```

[See Also]

None

3.4.3 AI Control MPIs

The following are AI control MPIs:

- HI_MPI_ISP_SetIrisAttr: Sets the iris control attribute.
- HI_MPI_ISP_GetIrisAttr: Obtains the iris control attribute.
- HI_MPI_ISP_SetDcirisAttr: Sets the AI control attribute of the DC iris.
- HI_MPI_ISP_GetDcirisAttr: Obtains the AI control attribute of the DC iris.
- HI_MPI_ISP_SetPirisAttr: Sets the AI control attribute of the P iris.
- HI_MPI_ISP_GetPirisAttr: Obtains the AI control attribute of the P iris.

HI_MPI_ISP_SetIrisAttr

[Description]

Sets the iris control attribute. This MPI can be used to set manual iris and iris type parameters.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetIrisAttr(ISP_DEV IspDev, const ISP_IRIS_ATTR_S
*pstIrisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIrisAttr	Pointer to the iris control attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

- Before testing the iris by using the AI algorithms, you are advised to check whether the AI circuit characteristics meet the requirements of the HiSilicon Internet Protocol camera (IPC).
- When AI is disabled, the iris type cannot be changed. Therefore the iris type retains the previous value.
- You are advised to set the iris type attribute based on the iris type of the actual connected lens, and then set the corresponding control attribute of the DC iris or P iris. If the connected lens uses the manual iris, the iris type can be set to **ISP_IRIS_DC_TYPE**, and you are advised to disable AI.
- The manual iris attribute is used for debugging, and can be configured by using this MPI. For the P iris, the manual iris parameter **enIrisFNO** is affected by the maximum and minimum iris target values. More AI parameters can be configured by calling HI_MPI_ISP_SetDcirisAttr and HI_MPI_ISP_SetPirisAttr.

[Example]

None

[See Also]

- ISP_IRIS_ATTR_S
- HI_MPI_ISP_SetDcirisAttr
- HI_MPI_ISP_SetPirisAttr

HI_MPI_ISP_GetIrisAttr

[Description]

Obtains the iris control attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetIrisAttr(ISP_DEV IspDev, ISP_IRIS_ATTR_S
*pstIrisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIrisAttr	Pointer to the iris control attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_ISP_SetDcirisAttr

[Description]

Sets the AI control attribute of the DC iris. This MPI can be used to set the AI parameters of the DC iris.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDcirisAttr(ISP_DEV IspDev, const ISP_DCIRIS_ATTR_S
*pstDcirisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDcirisAttr	Pointer to the AI control attribute of the DC iris	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

- The proportional-integral-derivative (PID) algorithm is used for controlling the DC iris. This algorithm controls the iris size by adjusting the pulse-width modulation (PWM) duty cycle based on the picture luminance. When the exposure time and gain reach the minimum target values, the algorithms start iris control. If the iris control meets the requirements of the target luminance, the AE exits directly and the exposure time and gain remain unchanged. After the picture luminance becomes stable and the PWM duty cycle retains the value when the iris is opened for a while, the AI algorithms consider that the iris has been opened to the maximum size, end the iris control, and hand over the control to the AE algorithms. During iris control, if AE algorithm parameters that need to take effect immediately (for example, the maximum/minimum exposure time, maximum/minimum gain, and anti-flicker parameter) are changed, the AE algorithm responds instantaneously. Then the AI algorithm determines whether to start iris control based on the configured parameters and ambient luminance. Starting or ending iris control takes a short period of time. Therefore, you are advised to disable AI when the manual iris is used; otherwise, the AE adjustment speed will be affected. It is recommended that AI be always enabled for the DC iris because disabling and enabling AI may result in iris control exceptions. For some telephoto DC-iris lenses, the default parameter values may cause the iris to be enabled or disabled too quickly. This issue can be solved by adjusting related parameters. For details, see the description of ISP_DCIRIS_ATTR_S.
- If the AI function is disabled, the DC iris will be opened to the maximum size.

[Example]

None

[See Also]

- ISP_IRIS_ATTR_S
- ISP_DCIRIS_ATTR_S

HI_MPI_ISP_GetDcirisAttr

[Description]

Obtains the AI control attribute of the DC iris.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDcirisAttr(ISP_DEV IspDev, ISP_DCIRIS_ATTR_S
*pstDcirisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDcirisAttr	Pointer to the AI control attribute of the DC iris	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_ISP_SetPirisAttr

[Description]

Sets the AI control attribute of the P iris.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetPirisAttr(ISP_DEV IspDev, const ISP_PIRIS_ATTR_S
*pstPirisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstPirisAttr	Pointer to the AI control attribute of the P iris	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

- The AI control attribute of the P iris contains a write-only parameter **bStepFNOTableChange**. You are advised to set the AI control attribute of the P iris before obtaining it. Otherwise, an error may occur.
- The P iris is controlled by using the AE allocation route. To ensure that the P iris drive connects to the corresponding lens and the P iris works properly, the parameters related to the lens and the AE allocation route must be correctly configured. For details, see ISP_PIRIS_ATTR_S and HI_MPI_ISP_SetAERouteAttr. The drive mode may differ according to the specifications of the P iris. You can modify the P iris drive to adapt to different lenses.
- If the AI function is disabled, the P iris will be opened to the maximum iris target value, which corresponds to the position of the stepper motor.

[Example]

```
ISP_DEV IspDev = 0;
    ISP_PIRIS_ATTR_S stPirisAttr,stPirisAttrDef;
    HI_U16 u16TotalStepDef = 93;
    HI_U16 u16StepCountDef = 62;
    HI_U16 au16StepFNOTableDef[1024] =
{30,35,40,45,50,56,61,67,73,79,85,92,98,105,112,120,127,135,143,150,158,1
```

```
66,174,183,191,200,208,217,225,234,243,252,261,270,279,289,298,307,316,32
5,335,344,353,362,372,381,390,399,408,417,426,435,444,453,462,470,478,486
,493,500,506,512};
    ISP_IRIS_F_NO_E  enMaxIrisFNOTargetDef = 9;
    ISP_IRIS_F_NO_E  enMinIrisFNOTargetDef = 5;
    stPirisAttrDef.bStepFNOTableChange = HI_TRUE;
    stPirisAttrDef.bZeroIsMax = HI_TRUE;
    stPirisAttrDef.u16StepCount = u16StepCountDef;
    stPirisAttrDef.u16TotalStep = u16TotalStepDef;
    stPirisAttrDef.enMaxIrisFNOTarget = enMaxIrisFNOTargetDef;
    stPirisAttrDef.enMinIrisFNOTarget = enMinIrisFNOTargetDef;
    memcpy(stPirisAttrDef.au16StepFNOTable, au16StepFNOTableDef,
sizeof(stPirisAttrDef.au16StepFNOTable));
HI_MPI_ISP_SetPirisAttr(IspDev, &stPirisAttrDef);
HI_MPI_ISP_GetPirisAttr(IspDev, &stPirisAttr);
```

[See Also]

- ISP_IRIS_ATTR_S
- ISP_PIRIS_ATTR_S

HI_MPI_ISP_GetPirisAttr

[Description]

Obtains the AI control attribute of the P iris.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetPirisAttr(ISP_DEV IspDev, ISP_PIRIS_ATTR_S
*pstPirisAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstPirisAttr	Pointer to the AI control attribute of the P iris	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 files: libisp.a, lib_hiae.a

[Note]

None

[Example]

None

[See Also]

None

3.5 Data Structures

3.5.1 Registration

The following are registration-related data structures:

- AE_SENSOR_REGISTER_S: Defines sensor registration information.
- AE_SENSOR_EXP_FUNC_S: Defines the sensor callback function.
- AE_SENSOR_DEFAULT_S: Defines the initialization parameter for the AE algorithm library.
- AE_ACCURACY_E: Defines the precision type of the exposure time and gain.
- AE_ACCURACY_S: Defines the precision of the exposure time and gain.
- AE_FSWDR_ATTR_S: Defines the structure of the ISP FSWDR running mode attribute.

AE_SENSOR_REGISTER_S

[Description]

Defines sensor registration information.

[Syntax]

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

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]

AE_SENSOR_EXP_FUNC_S

AE_SENSOR_EXP_FUNC_S

[Description]

Defines the sensor callback function.

[Syntax]

```
typedef struct hiAE_SENSOR_EXP_FUNC_S
{
  HI_S32(*pfn_cmos_get_ae_default)(AE_SENSOR_DEFAULT_S *pstAeSnsDft);
  HI_VOID(*pfn_cmos_fps_set)(HI_U8 u8Fps, AE_SENSOR_DEFAULT_S
  *pstAeSnsDft);
  HI_VOID(*pfn_cmos_slow_framerate_set)(HI_U16 u16SlowFrameRate,
  AE_SENSOR_DEFAULT_S *pstAeSnsDft);
  HI_VOID(*pfn_cmos_inttime_update)(HI_U32 u32IntTime);
  HI_VOID(*pfn_cmos_gains_update)(HI_U32 u32Again, HI_U32 u32Dgain);

  HI_VOID (*pfn_cmos_again_calc_table)(HI_U32 *pu32AgainLin, HI_U32
  *pu32AgainDb);
  HI_VOID (*pfn_cmos_dgain_calc_table)(HI_U32 *pu32DgainLin, HI_U32
  *pu32DgainDb);
  HI_VOID (*pfn_cmos_get_inttime_max)(HI_U32 u32Ratio, HI_U32
  *pu32IntTimeMax);
  HI_VOID(*pfn_cmos_ae_fswdr_attr_set)(AE_FSWDR_ATTR_S *pstAeFSWDRAttr);
} AE_SENSOR_EXP_FUNC_S;
```

[Member]

Member	Description
pfn_cmos_get_ae_default	Pointer to the callback function for obtaining the initial value of the AE algorithm library
pfn_cmos_fps_set	Pointer to the callback function for setting the frame rate of a sensor
pfn_cmos_slow_framerate_set	Pointer to the callback function for reducing the frame rate of a sensor

Member	Description
pfn_cmos_inttime_update	Pointer to the callback function for setting the exposure time of a sensor
pfn_cmos_gains_update	Pointer to the callback function for setting the analog and digital gains of a sensor
pfn_cmos_again_calc_table	Pointer to the callback function for setting the analog gain table of a sensor
pfn_cmos_dgain_calc_table	Pointer to the callback function for setting the digital gain table of a sensor
pfn_cmos_get_inttime_max	Pointer to the callback function for obtaining the allowed maximum short exposure time under different exposure multiples for long and short frames in multi-frame combination WDR mode. **pfn_cmos_get_inttime_max** is closely related to the sensor.
pfn_cmos_ae_fswdr_attr_set	Pointer to the callback function for setting the FSWDR mode

[Note]

- If a callback function pointer is not required, set it to **null**.
- The precision of the exposure time and gain is defined in AE_SENSOR_DEFAULT_S. The exposure time and gain configured in pfn_cmos_inttime_update and pfn_cmos_gains_update respectively are the values with precision. The method of converting the exposure time and gain into the configured values of the sensor is closely related to the sensor. For details, see corresponding sensor manuals.

[See Also]

ISP_SENSOR_REGISTER_S

AE_SENSOR_DEFAULT_S

[Description]

Defines the initialization parameter for the AE algorithm library.

[Syntax]

```
typedef struct hiAE_SENSOR_DEFAULT_S
{
    HI_U8   au8HistThresh[4];
    HI_U8   u8AeCompensation;
    HI_U32  u32LinesPer500ms;
    HI_U32  u32FlickerFreq;
    HI_FLOAT f32Fps;
    HI_U32  u32InitExposure;
    HI_U32  u32InitAESpeed;
    HI_U32  u32InitAETolerance;
```

```
    HI_U32 u32FullLinesStd;
    HI_U32  u32FullLinesMax;
    HI_U32  u32FullLines;
    HI_U32  u32MaxIntTime;
    HI_U32  u32MinIntTime;
    HI_U32  u32MaxIntTimeTarget;
    HI_U32  u32MinIntTimeTarget;
    AE_ACCURACY_S stIntTimeAccu;
    HI_U32  u32MaxAgain;
    HI_U32  u32MinAgain;
    HI_U32  u32MaxAgainTarget;
    HI_U32  u32MinAgainTarget;
    AE_ACCURACY_S stAgainAccu;
    HI_U32  u32MaxDgain;
    HI_U32  u32MinDgain;
    HI_U32  u32MaxDgainTarget;
    HI_U32  u32MinDgainTarget;
    AE_ACCURACY_S stDgainAccu;
    HI_U32  u32MaxISPDgainTarget;
    HI_U32  u32MinISPDgainTarget;
    HI_U32  u32ISPDgainShift;
    ISP_AE_ROUTE_S stAERouteAttr;
    HI_BOOL bAERouteExValid;
    ISP_AE_ROUTE_EX_S stAERouteAttrEx;
    HI_U16  u16ManRatioEnable;
    HI_U32 u32Ratio;
    ISP_IRIS_TYPE_E  enIrisType;
    ISP_PIRIS_ATTR_S stPirisAttr;
    ISP_IRIS_F_NO_E  enMaxIrisFNO;
    ISP_IRIS_F_NO_E  enMinIrisFNO;
} AE_SENSOR_DEFAULT_S;
```

[Member]

Member	Description
au8HistThresh	Segmentation threshold array of the 5-segment histogram. The value range is [0, 255]. The default value is {0xD, 0x28, 0x60, 0x80} in linear mode, {0x20, 0x40, 0x60, 0x80} in BUILT_IN WDR mode, or {0xC, 0x18, 0x60, 0x80} in multi-frame combination WDR mode. The default value is recommended.
u8AeCompensation	Target AE luminance. The value range is [0, 255], and the value 0x38−0x40 is recommended.
u32LinesPer500ms	Total number of lines per 500 ms

Member	Description
u32FlickerFreq	Anti-flicker frequency, which is 256 times the current power frequency
f32Fps	Reference frame rate. When the HiSilicon AE algorithms and the callback function cmos_fps_set are used, this member must be assigned to return the actual frame rate.
u32InitExposure	Default initial exposure. It is the exposure time (in line) multiplied by the system gain (6-bit decimal precision). The HiSilicon AE algorithm uses **u32InitExposure** as the exposure of the first five frames, which can be used for accelerating the startup of the motion DV. For the products that require fast boot, it is recommended that the initial exposure be configured based on common scenarios to reach the fast AE convergence effect. During switching between the linear mode and WDR mode, this value is used as the exposure value of the first frame after switching. Modify the exposure time and gain of the sensor initialization sequence, and multiply the exposure time (number of lines) by the gain (multiple, 6-bit decimal precision) to obtain the exposure value. Then assign the obtained exposure value to this variable to make the switching smoother. The default value 0 is used if this member is not configured.
u32InitAESpeed	Default initial AE adjustment speed. The HiSilicon AE algorithm uses this value as the adjustment speed of the first 100 frames, which can be used for accelerating the startup of the motion DV. For the products that require fast boot, it is recommended that the value of this member is configured to **128**. Otherwise, the default number of this member is 0. The value range will be limited to [64, 255] by the internal AE algorithm.
u32InitAETolerance	Default initial AE tolerance. The HiSilicon AE algorithm uses this value as the exposure tolerance of the first 100 frames to obtain the luminance range for the flag indicating that the AE convergence becomes stable for the first time. When the AE statistical luminance falls within [Target luminance – u32InitAETolerance, Target luminance + u32InitAETolerance] for the first time, the flag indicating that the AE convergence becomes stable for the first time is obtained. The default value 0 is used if this member is not configured. If this member is not set or is set to **0** in **cmos.c**, the AE algorithm internally sets it to **u8AeCompensation**/10. The AE algorithm limits the value of this member to [0, 255] internally.
u32FullLinesStd	Number of valid lines in a frame at the reference frame rate
u32FullLinesMax	Maximum number of rows in one frame after the frame rate of the sensor is reduced. **u32FullLinesMax** is generally set to the maximum number of rows supported by the sensor.

Member	Description
u32FullLines	Total number of actual rows in one frame for the sensor. When the HiSilicon AE algorithms and the callback functions cmos_fps_set and cmos_slow_framerate_set are used, this member must be assigned to return the total number of actual rows in one frame.
u32MaxIntTime	Maximum exposure time (in line)
u32MinIntTime	Minimum exposure time (in line)
u32MaxIntTimeTarget	Maximum target exposure time (in line)
u32MinIntTimeTarget	Minimum target exposure time (in line)
stIntTimeAccu	Exposure time precision
u32MaxAgain	Maximum analog gain (measured by multiple)
u32MinAgain	Minimum analog gain (measured by multiple)
u32MaxAgainTarget	Maximum target analog gain (measured by multiple)
u32MinAgainTarget	Minimum target analog gain (measured by multiple)
stAgainAccu	Analog gain precision
u32MaxDgain	Maximum digital gain (measured by multiple)
u32MinDgain	Minimum digital gain (measured by multiple)
u32MaxDgainTarget	Maximum target digital gain (measured by multiple)
u32MinDgainTarget	Minimum target digital gain (measured by multiple)
stDgainAccu	Digital gain precision
u32MaxISPDgainTarget	Maximum target ISP digital gain (measured by multiple)
u32MinISPDgainTarget	Minimum target ISP digital gain (measured by multiple)
u32ISPDgainShift	Precision of the ISP digital gain
stAERouteAttr	Default AE route
bAERouteExValid	Whether the extended AE allocation route is valid. When **bAERouteExValid** is set to **HI_TRUE**, the extended AE allocation route is used; otherwise, the common AE allocation route is used.
stAERouteAttrEx	Default extended AE allocation route. If this member is set to an appropriate value, the picture effect in WDR mode can be optimized.
u16ManRatioEnable	Manual exposure ration enable in 2-frame combination WDR mode. When **u16ManRatioEnable** is set to **HI_TRUE**, the exposure ratio is fixed and is not automatically updated based on the scenario.

Member	Description
u32Ratio	Default exposure ration in 2-frame combination WDR mode (6-bit decimal precision). When **u16ManRatioEnable** is set to **HI_TRUE**, **u32Ratio** is used as the default exposure ratio. Value range: [0x40, 0xFFF]
enIrisType	Default iris type (DC iris or P iris). The AI algorithms fail to work properly when the default iris type is inconsistent with the iris type of the connected lens. When the DC iris and P iris are switched in online mode, do not set this member in **cmos.c**. Otherwise, the AE route may not match the iris type.
stPirisAttr	P iris parameter, which is related to the lens. This parameter is required only when the default iris type is P iris.
enMaxIrisFNO	Maximum F value that can be used by the P iris, which is related to the lens and must be configured when the P iris is used. **enMaxIrisFNO** is used to limit the values of **enMaxIrisFNOTarget** and **enMinIrisFNOTarget** that actually take effect to avoid AE allocation exceptions caused when the iris target value falls within an inappropriate range. Value range: [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
enMinIrisFNO	Minimum F value that can be used by the P iris, which is related to the lens and must be configured when the P iris is used. **enMinIrisFNO** is used to limit the values of **enMaxIrisFNOTarget** and **enMinIrisFNOTarget** that actually take effect to avoid AE allocation exceptions caused when the iris target value falls within an inappropriate range. Value range: [ISP_IRIS_F_NO_32_0, enMaxIrisFNO]

[Note]

- During the switching between the linear mode and WDR mode, the callback function pfn_cmos_get_ae_default is used to update the related AE default parameters. If the extended AE allocation route is used in WDR mode and not used in linear mode, you are advised to clear the AE route in the cmos_get_ae_default function as follows: **bAERouteExValid** = **HI_FALSE**, **stAERouteAttr.u32TotalNum** = **0**, **stAERouteAttrEx.u32TotalNum** = **0**, and then configure the WDR branch as required.
- The HiSilicon AE algorithm uses **u32InitExposure** as the exposure of the first five frames, which can be used for accelerating the startup of the motion DV. For the products that require fast boot, it is recommended that the initial exposure be configured based on common scenarios to achieve the fast AE convergence effect. In FSWDR mode, **u32InitExposure** indicates the short frame exposure. To rapidly achieve normal AE in FSWDR mode, the value should be set to the fixed exposure ratio in **cmos.c** to shorten the time for adjusting the exposure ratio. When the AE becomes stable, **u32InitExposure** can be set to automatic exposure ratio as required. If **u32InitExposure** is not set or set to **0** in **cmos.c**, the AE algorithm internally calculates the exposure from 1024.
- During frame rate switching, note the frame rate (**f32Fps**) and number of valid rows (**u32FullLines**) that take effect no matter whether the frame rate changes.

[See Also]

ISP_SENSOR_EXP_FUNC_S

AE_ACCURACY_E

[Description]

Defines the precision type of the exposure time and gain.

[Syntax]

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

[Member]

Member	Description
AE_ACCURACY_DB	DB precision
AE_ACCURACY_LINEAR	Linear precision
AE_ACCURACY_TABLE	Table type

[Note]

None

[See Also]

AE_ACCURACY_S

AE_ACCURACY_S

[Description]

Defines the precision of the exposure time and gain.

[Syntax]

```
typedef struct hiAE_ACCURACY_S
{
  AE_ACCURACY_E enAccuType;
  float   f32Accuracy;
  float   f32Offset;
} AE_ACCURACY_S;
```

[Member]

Member	Description
enAccuType	Precision type, including linear precision, DB precision and table precision.
f32Accuracy	Precision
f32Offset	Exposure time offset (in line), which is closely related to the sensor

[Note]

- The precision of most gains is classified into linear precision, DB precision and TABLE precision.
- The linear precision indicates that the gain multiple increases evenly. For example, if **again** supported by the sensor is 1x, 1 (1 + 1/16)x, 1 (1 + 2/16)x, …, and 1 (1 + *N*/16)x, the precision type can be set to **AE_ACCURACY_LINEAR**, and the precision can be set to **0.0625**. In this case, if **again** is **32**, the gain is 2x (32 x 0.0625).
- The DB precision indicates that gain multiple is doubled. For example, if **again** supported by the sensor is 0 dB, 0.3 dB, 0.6 dB, …, and (0.3 x N) dB, the precision type can be set to **AE_ACCURACY_DB**, and the precision can be set to **0.3**. In this case, if **again** is **80**, the gain is 16x (80 x 0.3 = 24 dB). If the **again** supported by the sensor is 1x, 2x, 4x, 8x, …, and 2^nx (corresponding to 0 dB, 6 dB, 12 dB, 18 dB, …, and 6*N* dB), the precision type is set to **AE_ACCURACY_DB**, and the precision can be set to **6**. Because there is an error when the AE algorithm calculates the precision for converting the linear precision into DB precision, the table precision is recommended if the DB precision is less than 1.
- The table precision indicates the gain multiple and is obtained by querying the gain table. The table precision is fixed at 10 bits, that is, the value **1024** indicates 1x gain. When the gains of some sensors increase in non-linear mode, the required analog gain or digital gain can be calculated by using the AE algorithm. The value that is queried from the sensor gain table and closest to the calculated value is used as the digital gain or analog gain.
- When the gain table is used, the callback function in the initialization callback data structure AE_SENSOR_EXP_FUCN_S is required.
- Typically, the exposure time is in linear precision and its unit is line. If the exposure time is in linear precision and the precision (**f32Accuracy**) is an integer greater than 1, the number of exposure lines calculated by the AE algorithm can only be an integral multiple of **f32Accuracy**. Based on this rule, the long/short frame exposure time of some sensors in WDR mode can be calculated. For example, when the Sony IMX185 sensor works in built-in WDR mode, if the ratio of the long frame exposure to the short frame exposure is 16, the AE algorithm calculates the long frame exposure time based on the statistics, and obtains the short frame exposure time by dividing the long frame exposure time by 16 in **cmos.c**. To ensure that the number of exposure lines for a short frame is a value obtained after exact division, the number of exposure lines for a long frame must be a multiple of the exposure ratio (16). Therefore, **f32Accuracy** can be set to **16**. The maximum value of the long frame exposure time is greater than 10 ms, which meets the requirement of enabling anti-flicker at 50 Hz/60 Hz. In this case, enabling anti-flicker can optimize the picture effect in WDR mode under the normal indoor illumination.
- Before AE light exposure is allocated, ensure that there is a linear relationship between the exposure time and the sensor gain. That is, when the light exposure is fixed, the image luminance is retained when the exposure time and sensor gain are changed. For

example, if the light exposure is 4096, the image luminance should be retained when the exposure time is 2 and gain is 2048 or the exposure time is 4 and gain is 1024. If there is no linear relationship between the exposure time and the sensor gain, the screen flickers even the exposure time of one line changes in a highlight environment because the exposure time is too short. When some sensors such as Panasonic MN34220 use the 1080p@30 fps initialization sequence, there is a linear relationship between the exposure time and the sensor gain only after 0.8018 line is shifted. Therefore, the **f32Offset** variable is added. The unit of this floating variable is line. When **f32Offset** is set to **0.8018**, the required offset is implemented in the AE algorithm. The offset is not required for other sensors. In this case, **f32Offset** must be set to **0** in **cmos.c**.

- The preceding three precision types are defined to connect most sensors over unified interfaces.

[See Also]

ISP_SENSOR_REGISTER_S

AE_FSWDR_ATTR_S

[Description]

Defines the structure of the ISP FSWDR running mode attribute.

[Syntax]

```
typedef struct hiAE_FSWDR_ATTR_S
{
    ISP_FSWDR_MODE_E enFSWDRMode;
} AE_FSWDR_ATTR_S;
```

[Member]

Member	Description
enFSWDRMode	FS WDR running mode, including the normal WDR mode and long frame mode

[Note]

The long frame mode is valid only in row WDR mode. If the exposure ratio is set to 1:1 in frame WDR mode, the effect is the same as that in row WDR long frame mode.

[See Also]

AE_SENSOR_EXP_FUNC_S

3.5.2 AE

The following are AE data structures:

- ISP_OP_TYPE_E: Defines the exposure mode.
- ISP_AE_MODE_E: Defines the AE mode.
- ISP_AE_STRATEGY_E: Defines the AE exposure policy mode.
- ISP_AE_DELAY_S: Defines the ISP exposure delay attribute.

- ISP_AE_RANGE_S: Defines the maximum exposure time or gain and minimum exposure time or gain.
- ISP_ANTIFLICKER_MODE_E: Defines the anti-flicker mode.
- ISP_ANTIFLICKER_S: Defines the anti-flicker attribute.
- ISP_SUBFLICKER_S: Defines the sub-anti-flicker attribute of the ISP.
- ISP_FSWDR_MODE_E: Defines the FSWDR mode of the ISP.
- ISP_AE_ATTR_S: Defines the AE attributes of the ISP.
- ISP_ME_ATTR_S: Defines the ME attribute.
- ISP_EXPOSURE_ATTR_S: Defines the ISP exposure attribute.
- ISP_WDR_EXPOSURE_ATTR_S: Defines the AE exposure attribute in WDR mode.
- ISP_AE_ROUTE_NODE_S: Defines the attributes of the AE route node.
- ISP_AE_ROUTE_S: Defines the attributes of the ISP exposure allocation policy.
- ISP_AE_ROUTE_EX_NODE_S: Defines the attributes of the node on the extended AE route.
- ISP_AE_ROUTE_EX_S: Defines the extended attribute of AE allocation policy.
- ISP_EXP_INFO_S: Defines the internal exposure status information about the ISP.

ISP_OP_TYPE_E

[Description]

Defines the exposure mode.

[Syntax]

```
typedef enum 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 mode
OP_TYPE_MANUAL	Manual mode

[Note]

In automatic mode, changing the ME attribute has no effect; in manual mode, the ME time and gain are limited by the maximum/minimum exposure time and gain in automatic mode.

[See Also]

None

ISP_AE_MODE_E

[Description]

Defines the AE mode.

[Syntax]

```
typedef enum hiISP_AE_MODE_E
{
    AE_MODE_SLOW_SHUTTER = 0,
    AE_MODE_FIX_FRAME_RATE  = 1,
    AE_MODE_BUTT
} ISP_AE_MODE_E;
```

[Member]

Member	Description
AE_MODE_ SLOW_SHUTTER	Automatic frame rate reduction mode (slow shutter mode)
AE_MODE_FIX_FRAME_RATE	Constant frame rate mode

[Note]

- In automatic frame rate reduction mode, the exposure time is increased in priority to minimize the gain during AE adjustment. When the sensor gain reaches 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 is low.
- In constant frame rate mode, the frame rate is retained during AE adjustment. In low-illumination scenarios, the noises are large.
- The built-in WDR mode and multi-frame combination WDR mode support frame rate reduction. If the line-WDR sensor is connected to the customer's sensor, **cmos.c** should be modified to automatically reduce the frame rate. For details about the modification method, see the **cmos.c** file in **mn34220/imx123/ov4689**.

[See Also]

None

ISP_AE_STRATEGY_E

[Description]

Defines the AE exposure policy mode.

[Syntax]

```
typedef enum hiISP_AE_STRATEGY_E
{
    AE_EXP_HIGHLIGHT_PRIOR = 0,
    AE_EXP_LOWLIGHT_PRIOR  = 1,
```

```
    AE_STRATEGY_MODE_BUTT
} ISP_AE_STRATEGY_E;
```

[Member]

Member	Description
AE_EXP_HIGHLIGHT_PRIOR	Highlight first exposure mode
AE_EXP_LOWLIGHT_PRIOR	Lowlight first exposure mode

[Note]

The default exposure policy mode is highlight first exposure mode. In this mode, overexposure is avoided. In WDR scenarios, the luminance in dark regions is low. If dark regions need to be concerned, the lowlight first exposure mode can be used. However, overexposure occurs in bright regions.

[See Also]

None

ISP_AE_DELAY_S

[Description]

Defines the ISP exposure delay attribute.

[Syntax]

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

[Member]

Member	Description
u16BlackDelayFrame	AE adjustment starts when the image luminance is less than the target luminance for **u16BlackDelayFrame** frames.
u16WhiteDelayFrame	AE adjustment starts when the image luminance is greater than the target luminance for **u16WhiteDelayFrame** frames.

[Note]

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

- When a constant frame rate is decreased, if the frame rate is too low, the values of **u16BlackDelayFrame** and **u16WhiteDelayFrame** must be adjusted. You are advised not to set **u16BlackDelayFrame** and **u16WhiteDelayFrame** to too large values. Otherwise, the AE convergence time is long.

[See Also]

None

ISP_AE_RANGE_S

[Description]

Defines the maximum exposure time or gain and minimum exposure time or gain.

[Syntax]

```
typedef struct hiISP_AE_RANGE_S
{
    HI_U32 u32Max;
    HI_U32 u32Min;
} ISP_AE_RANGE_S;
```

[Member]

Member	Description
u32Max	Maximum value
u32Min	Minimum value

[Note]

The minimum value must be less than or equal to the maximum value.

[See Also]

None

ISP_ANTIFLICKER_MODE_E

[Description]

Defines the anti-flicker mode.

[Syntax]

```
typedef enum hiISP_ANTIFLICKER_MODE_E
{
    ISP_ANTIFLICKER_NORMAL_MODE = 0x0,
    ISP_ANTIFLICKER_AUTO_MODE = 0x1,
    ISP_ANTIFLICKER_MODE_BUTT
}ISP_ANTIFLICKER_MODE_E;
```

[Member]

Member	Description
ISP_ANTIFLICKER_NORMAL_MODE	Normal anti-flicker mode
ISP_ANTIFLICKER_AUTO_MODE	Automatic anti-flicker mode

[Note]

- In normal anti-flicker mode, the exposure time can be adjusted based on the luminance. The minimum exposure time is fixed at 1/120s (60 Hz) or 1/100s (50 Hz). The exposure time is not limited by the minimum exposure time in this mode.
  - When there are lights, the exposure time can match the illuminant frequency, which avoids image flicker.
  - In high-luminance environments, higher luminance requires shorter exposure time. However, the minimum exposure time in normal anti-flicker mode cannot match the illuminant frequency, which results in overexposure.
- In automatic anti-flicker mode, the exposure time can be adjusted based on the luminance. The minimum exposure time can be the minimum exposure time of the sensor. The normal anti-flicker mode differs from the automatic anti-flicker mode 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.

[Syntax]

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

[Member]

Member	Description
bEnable	Anti-flicker enable HI_TRUE: enabled HI_FALSE: disabled

Member	Description
u8Frequency	Anti-flicker frequency Value range: [0x0, 0xFF] Default value: 0x0
enMode	Anti-flicker mode (normal anti-flicker mode or automatic anti-flicker mode)

[Note]

After anti-flicker is enabled, the actual exposure time is limited by the maximum/minimum exposure time. If the minimum anti-flicker time is greater than the maximum exposure time, the actual exposure time is limited to the maximum exposure time after anti-flicker is enabled.

[See Also]

ISP_ANTIFLICKER_MODE_E

ISP_SUBFLICKER_S

[Description]

Defines the sub-anti-flicker attribute of the ISP.

[Syntax]

```
typedef struct hiISP_SUBFLICKER_S
{
    HI_BOOL bEnable;
    HI_U8   u8LumaDiff;
} ISP_SUBFLICKER_S;
```

[Member]

Member	Description
bEnable	Sub-anti-flicker enable HI_TRUE: enabled HI_FALSE: disabled
u8LumaDiff	Anti-flicker level Value range: [0x0, 0x64] When the sub-anti-flicker function takes effect, a higher anti-flicker level indicates that the sub-anti-flicker function is closer to the anti-flicker function.

[Note]

- In forcible anti-flicker mode, the minimum exposure time is fixed at 1/120 second (60 Hz) or 1/100 second (50 Hz). In some scenarios, for example, the indoor backlight

scenario (the lens is focused on the window), overexposure may be quite obvious. However, picture flicker at the power frequency is obvious if anti-flicker is not enabled. In this case, the sub-anti-flicker mode is adopted to balance overexposure with flicker. When the sub-anti-flicker function takes effect in forcible anti-flicker mode, if the picture luminance is less than (**AeCompensation** + **u8LumaDiff**), the exposure time is still fixed at the minimum anti-flicker time of 1/120 second (60 Hz) or 1/100 second (50 Hz) to avoid picture flicker. If the picture luminance is greater than (**AeCompensation** + **u8LumaDiff**), the anti-flicker is not enabled and the target picture luminance is adjusted to (**AeCompensation** + **u8LumaDiff**). Overexposure is mitigated due to the tolerance of certain amount of picture flicker. Note that in the non-backlight scenario, when **u8LumaDiff** is small (for example, less than 10), the exposure time may change consistently around the minimal anti-flicker value, which causes the slight picture flicker. It is recommended that this value be greater than or equal to 20. The reason is that when **u8LumaDiff** is set to a small value, the anti-flicker function does not take effect in the backlight scenario and the effect in the backlight scenario is similar to that in automatic anti-flicker mode. The sub-anti-flicker effect cannot be achieved. Therefore, this member should be set to a large value.

- The sub-anti-flicker function takes effect only when anti-flicker is enabled in forcible anti-flicker mode and the anti-flicker frequency is not 0.

[See Also]

None

ISP_FSWDR_MODE_E

[Description]

Defines the FSWDR mode of the ISP.

[Syntax]

```
typedef enum hiISP_FSWDR_MODE_E
{
    ISP_FSWDR_NORMAL_MODE = 0x0,
    ISP_FSWDR_LONG_FRAME_MODE = 0x1,
    ISP_FSWDR_MODE_BUTT
}ISP_FSWDR_MODE_E;
```

[Member]

Member	Description
ISP_FSWDR_NORMAL_MODE	Normal mode. In this mode, the AE and the combination module work based on the automatic/manual exposure ratio.
ISP_FSWDR_LONG_FRAME_MODE	Long frame mode. In this mode, the AE sets the short frame exposure time to the minimum value, the long frame exposure time is close to the maximum value that a frame allows, and the combination module outputs only the long frame data. This mode can be used to optimize the picture quality in weak WDR or low illumination scenarios.

[Note]

The long frame mode takes effect only in the line WDR mode. To achieve the same effect in the frame WDR mode, set the exposure ratio to 1:1.

[See Also]

None

ISP_AE_ATTR_S

[Description]

Defines the AE attributes of the ISP.

[Syntax]

```
typedef struct hiISP_AE_ATTR_S
{
   ISP_AE_RANGE_S stExpTimeRange;
   ISP_AE_RANGE_S stAGainRange;
   ISP_AE_RANGE_S stDGainRange;
   ISP_AE_RANGE_S stISPDGainRange;
   ISP_AE_RANGE_S stSysGainRange;
   HI_U32 u32GainThreshold;
   HI_U8  u8Speed;
   HI_U16  u16BlackSpeedBias;
   HI_U8  u8Tolerance;
   HI_U8  u8Compensation;
   HI_U16  u16EVBias;
   ISP_AE_STRATEGY_E enAEStrategyMode;
   HI_U16  u16HistRatioSlope;
   HI_U8   u8MaxHistOffset;
   ISP_AE_MODE_E  enAEMode;
   ISP_ANTIFLICKER_S stAntiflicker;
   ISP_SUBFLICKER_S stSubflicker;
   ISP_AE_DELAY_S stAEDelayAttr;
   HI_BOOL bManualExpValue;
   HI_U32 u32ExpValue;
   ISP_FSWDR_MODE_E enFSWDRMode;
   HI_BOOL bWDRQuick;
   HI_U8 au8Weight[AE_ZONE_ROW][AE_ZONE_COLUMN];
} ISP_AE_ATTR_S;
```

[Member]

Member	Description
stExpTimeRange	Exposure time range (in μs) Value range: [0x0, 0xFFFFFFFF] The specific range is related to the sensor.
stAGainRange	Range of the sensor analog gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.
stDGainRange	Range of the sensor digital gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.
stISPDGainRange	Range of the ISP digital gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF]
stSysGainRange	System gain range (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.
u32GainThreshold;	System gain threshold during automatic frame rate reduction (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF]
u8Speed	Speed during AE adjustment Value range: [0x0, 0xFF] Default value: 0x40
u16BlackSpeedBias	Offset of the AE adjustment speed for adjusting the picture from dark to bright. A larger value indicates a faster speed of adjusting the picture from dark to bright. Value range: [0x0, 0xFFFF] Default value: 0x90
u8Tolerance	Luminance tolerance during AE adjustment Value range: [0x0, 0xFF] Default value: 0x2
u8Compensation	Target luminance for AE adjustment Value range: [0x0, 0xFF] Default value: 0x38
u16EVBias	Exposure bias during AE adjustment (10-bit decimal precision) Value range: [0x0, 0xFFFF] Default value: 0x400
enAEStrategyMode	AE policy mode (highlight first or lowlight first)

Member	Description
u16HistRatioSlope	Region of interest (ROI) weight Value range: [0x0, 0xFFFF] Default value: 0x80
u8MaxHistOffset	Maximum influence on the average statistics exerted by ROIs Value range: [0x0, 0xFF] Default value: 0x10
enAEMode	AE mode (automatic frame rate reduction mode or constant frame rate mode)
stAntiflicker	Anti-flicker attribute. Anti-flicker is disabled by default.
stSubflicker	Sub-anti-flicker attribute. Sub-anti-flicker is disabled by default.
stAEDelayAttr	Delay attribute. The default value of **u16BlackDelayFrame** is 8, and the default value of **u16WhiteDelayFrame** is 0.
bManualExpValue	Manual exposure enable. When **bManualExpValue** is set to **HI_TRUE**, the AE algorithm uses **u32ExpValue** as the current exposure for the allocation of exposure time and gain. When **bManualExpValue** is set to **HI_FALSE**, the automatically calculated exposure is used for allocation. Default value: HI_FALSE
u32ExpValue	Manual exposure. It is the exposure time (in line) multiplied by the system gain (6-bit decimal precision). Value range: (0x0, 0xFFFFFFFF]
enFSWDRMode	FSWDR mode Default value: ISP_FSWDR_NORMAL_MODE
bWDRQuick	Disable for temporal filtering performed on the first 50 frames. In WDR mode, when the stable status of the AE algorithm is readjusted (the luminance error is less than or equal to the error tolerance **u8Tolerance**), the temporal filtering is performed on the first 50 frames, and therefore the AE adjustment is smoother. When **bWDRQuick** is **HI_TRUE**, the temporal filtering performed on the 50 frames is disabled and the AE convergence speed is faster. Default value: HI_FALSE
au8Weight[AE_ZONE_ROW][AE_ZONE_COLUMN]	AE zone weight table with 15 x 17 zones. Only the 5-segment histogram and 256-segment histogram are affected.

[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 alleviates image smearing.

- AE system gain

  If the product of the minimum sensor analog gain, minimum sensor digital gain, and minimum ISP digital gain is less than the minimum system gain, the minimum system gain is used for AE algorithm calculation. If the product of the maximum sensor analog gain, maximum sensor digital gain, and maximum ISP digital gain is greater than the maximum system gain, the maximum system gain is used for AE algorithm calculation. You are advised to limit gain values by setting the maximum and minimum system gains. When the sensor analog gain, sensor digital gain, and ISP digital gain are limited separately, if the high-precision ISP digital gain is limited to 1x, the screen may flicker. The AE algorithm internally calculates the maximum/minimum exposure by using the system gain, instead of directly limiting a certain gain value. For example, the limitation on the maximum system gain is 1x and the limitation on the minimum sensor analog gain is 2x. The result which actually takes effect is based on the limitation on the sensor analog gain. This rule applies to the sensor digital gain and the ISP digital gain.

- System gain threshold during automatic frame rate reduction

  In slow shutter mode, when the system gain reaches the threshold, the system enters the slow shutter mode automatically.

- **u8Speed** is used to set the AE convergence speed. A larger value indicates faster AE convergence but more frequent change of luminance during convergence. If the Panasonic MN34220 sensor is used, it is recommended that the configured value of **u8Speed** be less than 64 in WDR mode in which two frames are combined due to the performance restrictions of the Panasonic MN34220 sensor. Otherwise, flicker may occur when the light changes significantly.

- **u16BlackSpeedBias** is an offset value of the AE adjustment speed for adjusting the picture from dark to bright In default **u8Speed**, the AE adjustment speed from bright to dark is faster than that from dark to bright. To equalize the speeds in both directions, increase **u16BlackSpeedBias**.

- The luminance compensation attribute parameter **u8Compensation** is used to adjust the target exposure luminance. A larger exposure luminance compensation value indicates higher image luminance.

- The exposure bias attribute parameter **u16EVBias** is used to fine-tune the target luminance of the image in special scenarios and it can be considered as high-precision luminance compensation. The actual target AE luminance is calculated as follows: **u8Compensation** x **u16EVBias**/1024. When **u8Compensation** remains unchanged, a larger exposure bias indicates higher image luminance.

- The exposure tolerance attribute parameter **u8Tolerance** 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. Especially when the target luminance value of the image is relatively small and the exposure tolerance value is relatively large, the final luminance after the AE convergence when the luminance changes from bright to dark and the final image luminance after the AE convergence when the luminance changes from dark to bright may differ greatly. When the target luminance of the image is fine-tuned by changing the value of **u16EVBias**, if the exposure tolerance value is too large and the change of **u16EVBias** is too small, the adjusting effect may not be obvious.

- The exposure policy attribute parameter **enAEStrategyMode** is used to select the highlight first or lowlight first exposure policy. Highlight first indicates that the ISP is sensitive to the highlight. In this case, overexposure should be avoided. Lowlight first indicates that the ISP is sensitive to the lowlight. In this case, the dark regions are as clear as possible no matter whether overexposure occurs. The default exposure policy is highlight first, which can be changed based on the application scenario.

- **u16HistRatioSlope** is used to set the ROI weight. If the highlight first exposure policy is selected, the weight of highlight regions is configured. A larger weight indicates greater sensitivity to highlight regions. If the lowlight first exposure policy is selected, the weight of lowlight regions is configured. A larger weight indicates greater sensitivity to dark regions. It is recommended that the value of **u16HistRatioSlope** not be greater than 0x100. If the highlight first exposure policy is selected and the value of **u16HistRatioSlope** is too large, the AE algorithm considers that the picture luminance is much greater than the target luminance as long as there is a small overexposure region in the picture. Therefore, the exposure value is continuously decreased. As a result, the picture becomes dark. In addition, the overexposure region in the picture quickly becomes smaller and to a certain extent the AE algorithm considers that the picture luminance is less than the target luminance. Therefore, the exposure value is increased. In this way, the exposure value is decreased and increased repeatedly, and picture flicker occurs. If the lowlight first exposure policy is selected and the AE target luminance is low, the picture luminance is greatly affected when black objects move back and forth in the picture. This issue can be alleviated by decreasing the exposure adjustment step (**u8ExpStep**) and increasing the tolerance (**s16ExpTolerance**).
- In AE mode, the greatest impact (**u8MaxHistOffset**) on the average statistics exerted by ROIs can be configured. Configuring **u8MaxHistOffset** is equivalent to limiting the increased weight when **u16HistRatioSlope** is increased. If **u8MaxHistOffset** is **0**, there is no special processing in highlight or lowlight regions regardless of the value of **u16HistRatioSlope**. In this case, the average statistics are the original values. If **u8MaxHistOffset** is set to an appropriate value, the average luminance of the image after the AE is stable in any scenario can be limited to a certain range. If the highlight first exposure policy is selected and the value of **u8MaxHistOffset** is large, the luminance of the entire picture may be low in the scenarios with high contrast such as the outdoor scenario with the sky and trees in sunny days because the effect of sky in bright regions takes priority. This issue can be solved by decreasing the value of **u8MaxHistOffset**.
- When the exposure policy is switched, the values of **u16HistRatioSlope** and **u8MaxHistOffset** should be updated. Otherwise, the values of **u16HistRatioSlope** and **u8MaxHistOffset** for the previous exposure policy are used, and the result may be unexpected.
- You are advised to suppress the highlight by decreasing the AE target luminance and assigning appropriate values to **u16HistRatioSlope** and **u8MaxHistOffset** in highlight first exposure mode. You can enable the DRC function to make the dark regions clear. The lowlight first exposure policy can be used to implement backlight compensation in non-specified regions.
- When the exposure control mode is automatic mode, the exposure mode **enAEMode** can be set to slow shutter mode or constant frame rate mode. The slow shutter mode is used to automatically decrease the frame rate to reduce noises in the low-illuminant scenario.
- The anti-flicker attribute **stAntiflicker** can be used to set anti-flicker enable status, anti-flicker frequency, and anti-flicker mode. In FSWDR mode, **stAntiflicker** takes effect only on long frames. You are advised to use the automatic anti-flicker mode in FSWDR mode. The reason is that the maximum short frame exposure time is limited according to the change of the exposure ratio.
- The sub-anti-flicker attribute **stSubflicker** can be used to enable or disable the sub-anti-flicker function and set the anti-flicker level. If the AI function is used, you are advised to disable sub-anti-flicker.
- When the exposure control mode is automatic mode, the AE delay attribute **stAEDelayAttr** can be configured. An appropriate delay ensures stable luminance, preventing luminance variance when objects move quickly. The delay value can be

increased properly when the bit rate is low to avoid blocking artifact during AE adjustment.

- You can set **bManualExpValue** to **HI_TRUE** and then manually configure the exposure (**u32ExpValue**) to mask the adjusted exposure and use only the allocated exposure of the HiSilicon AE algorithms. **u32ExpValue** is limited by the maximum and minimum exposure. The maximum/minimum exposure is the maximum/minimum exposure time multiplied by the gain.
- When the FSWDR mode is changed, to ensure that the long/short frame exposure time falls within an appropriate range, the following callback functions in cmos.c need to be modified: cmos_long_frame_mode_set, cmos_get_ae_default, cmos_fps_set, and cmos_get_inttime_max. During the switching between the common WDR mode and the long frame mode, the callback function cmos_get_ae_default is used, and the maximum/minimum exposure time, maximum/minimum gain, maximum/minimum iris, and AE compensation are updated. In addition, the AE internally updates u16HistRatioSlope and u8MaxHistOffset to default values. Note that the settings related to the exposure ratio are not updated and retain the previous values. Besides, when long frame mode is enabled, the DRC strength gradually converges to the auto-target value. You can call the corresponding MPI to change the value of the parameter after switching is complete. Note that, to switch smoothly, the short frame exposure must be equal to the long frame exposure before/after the switching. When the exposure is large, the gains are limited in the common WDR mode. In long frame mode, to ensure the low illumination effect, the gains are not limited. However, the short frame exposure may not be equal to the long frame exposure before/after the switching. In this case, several frames suddenly become bright or dark. To avoid this case, the limit on gains is removed after the switching is complete. Note that the long frame mode takes effect only in line WDR mode. To achieve the same effect in frame WDR mode, set the exposure ratio to 1:1. It is recommended that **enFSWDRMode** be set to ISP_FSWDR_NORMAL_MODE in non-line WDR mode.
- Weight table for AE

  The static AE statistics are divided into 15 x 17 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 the global histogram statistics for the image. This attribute can be used to adjust the exposure weight of the ROI and implement backlight compensation for specified regions. If the position of the ROI for a specific product form is determined, the weight of the position can be increased. For example, the ROI is typically in the middle and lower part of the picture for the motion DV and event data recorder (EDR). Therefore, the weight of the middle and lower area can be properly increased to ensure that the exposure of the ROI is appropriate.

[See Also]

HI_MPI_ISP_SetExposureAttr

ISP_ME_ATTR_S

[Description]

Defines the ME attribute.

[Syntax]

```
typedef struct hiISP_ME_ATTR_S
{
    ISP_OP_TYPE_E enExpTimeOpType;
```

```
    ISP_OP_TYPE_E enAGainOpType;
    ISP_OP_TYPE_E enDGainOpType;
    ISP_OP_TYPE_E enISPDGainOpType;
    HI_U32 u32ExpTime;
    HI_U32 u32AGain;
    HI_U32 u32DGain;
    HI_U32 u32ISPDGain;
} ISP_ME_ATTR_S;
```

[Member]

Member	Description
enExpTimeOpType	ME time enable Default value: OP_TYPE_AUTO
enAGainOpType	Manual sensor analog gain enable Default value: OP_TYPE_AUTO
enDGainOpType	Manual sensor digital gain enable Default value: OP_TYPE_AUTO
enISPDGainOpType	Manual ISP digital gain enable Default value: OP_TYPE_AUTO
u32ExpTime	ME time (in μs) Default value: 0x4000 Value range: [0x0, 0xFFFFFFFF] The specific range is related to the sensor.
u32AGain	Manual sensor analog gain (10-bit decimal precision) Default value: 0x400 Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.
u32DGain	Manual sensor digital gain (10-bit decimal precision) Default value: 0x400 Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.
u32ISPDGain	Manual ISP digital gain (10-bit decimal precision) Default value: 0x400 Value range: [0x400, 0xFFFFFFFF] The specific range is related to the sensor.

[Note]

- If ME is enabled, ME parameters must be configured. Otherwise, the default values are used.

- The gain is measured by a 10-bit decimal precision, that is, the value 1024 indicates 1x gain.
- 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 less than the minimum value supported by the sensor, the minimum value is used.

[See Also]

HI_MPI_ISP_SetExposureAttr

ISP_EXPOSURE_ATTR_S

[Description]

Defines the ISP exposure attribute.

[Syntax]

```
typedef struct hiISP_EXPOSURE_ATTR_S
{
    HI_BOOL        bByPass;
    ISP_OP_TYPE_E   enOpType;
    HI_U8     u8AERunInterval;
    HI_BOOL   bHistStatAdjust;
    HI_BOOL         bAERouteExValid;
    ISP_ME_ATTR_S   stManual;
    ISP_AE_ATTR_S  stAuto;
} ISP_EXPOSURE_ATTR_S;
```

[Member]

Member	Description
bByPass	AE bypass enable Default value: HI_FALSE
enOpType	ME or AE mode Default value: OP_TYPE_AUTO
u8AERunInterval	Running interval of the AE algorithm Value range: [1, 255] Default value: 1 The value 1 indicates that the AE algorithm runs for every frame, the value 2 indicates that the AE algorithm runs for every two frames, and so on. The recommended maximum value is 2; otherwise, the AE adjustment speed is affected. In WDR mode, the recommended value is 1, which ensures smooth AE convergence.
bHistStatAdjust	Enable for adjusting the statistical mode of the 256-segment histogram according to the scenario Default value: HI_TRUE

Member	Description
bAERouteExValid	Whether the extended AE allocation route is valid. When **bAERouteExValid** is set to **HI_TRUE**, the extended AE allocation route is used; otherwise, the common AE allocation route is used. Default value: HI_FALSE
stManual	ME attribute
stAuto	AE attribute

[Note]

- When AE **ByPass** is **HI_TRUE**, the AE module is bypassed, and configuring the AE module has no effect on the picture luminance. ISP_AE_RESULT_S retains the value of the last frame before the AE module is bypassed.
- The HiSilicon AE algorithm is used to adjust the luminance based on the global 256-segment histogram. In normal scenarios, the 256-segment histogram collects only the Gb component or Gr component for statistics. If **bHistStatAdjust** is set to **HI_TRUE** in the scenario where there is a large region with a single color (for example, red or blue), the AE algorithm adjusts the statistical mode of the 256-segment histogram based on the global average value (the R component or B component is considered). In this way, the picture luminance will not be too high in the scenario with a large red or blue region. If **bHistStatAdjust** is set to **HI_FALSE**, the AE algorithm does not adjust the statistical mode of the 256-segment histogram. You are advised to set **bHistStatAdjust** to **HI_FALSE** in the infrared (IR) scenario.
- During the switchover between the WDR mode and linear mode, **u8AERunInterval** is reset to **1**. You can change the parameter value as required after switching is complete.

[See Also]

- HI_MPI_ISP_SetExposureAttr
- HI_MPI_ISP_GetExposureAttr

ISP_WDR_EXPOSURE_ATTR_S

[Description]

Defines the AE exposure attribute in WDR mode.

[Syntax]

```
typedef struct hiISP_WDR_EXPOSURE_ATTR_S
{
    ISP_OP_TYPE_E enExpRatioType;
    HI_U32 u32ExpRatio;
    HI_U32 u32ExpRatioMax;
    HI_U32 u32ExpRatioMin;
    HI_U16 u16Tolerance;
    HI_U16 u16Speed;
    HI_U16 u16RatioBias;
} ISP_WDR_EXPOSURE_ATTR_S;
```

[Member]

Member	Description
enExpRatioType	This member is valid only in the WDR mode in which two frames are combined and output as a frame. OP_TYPE_AUTO: The ratio of long frame exposure to short frame exposure is automatically calculated based on the application scenario. OP_TYPE_MANUAL: The ratio of long frame exposure to short frame exposure is manually configured.
u32ExpRatio	This member is valid only in the WDR mode in which two frames are combined and output as a frame. When **enExpRatioType** is **OP_TYPE_AUTO**, **u32ExpRatio** is invalid. When **enExpRatioType** is **OP_TYPE_MANUAL**, **u32ExpRatio** can be read and written and indicates the expected ratio of the long frame exposure time to the short frame exposure time. The format is unsigned 6.6-bit fixed point. The value 0x40 indicates that the ratio of the long frame exposure time to the short frame exposure time is 1. Value range: [0x40, 0xFFF]
u32ExpRatioMax	This member is valid only in the WDR mode in which two frames are combined and output as a frame. When **enExpRatioType** is **OP_TYPE_AUTO**, **u32ExpRatioMax** indicates the maximum ratio of the long frame exposure time to the short frame exposure time. When **enExpRatioType** is **OP_TYPE_MANUAL**, **u32ExpRatioMax** is invalid. The format is unsigned 6.6-bit fixed point. The value 0x40 indicates that the ratio of the long frame exposure time to the short frame exposure time is 1. The default value is 0x400. Value range: [0x40, 0xFFF]
u32ExpRatioMin	This member is valid only in the WDR mode in which two frames are combined and output as a frame. When **enExpRatioType** is **OP_TYPE_AUTO**, **u32ExpRatioMin** indicates the minimum ratio of the long frame exposure time to the short frame exposure time. When **enExpRatioType** is **OP_TYPE_MANUAL**, **u32ExpRatioMin** is invalid. The format is unsigned 6.6-bit fixed point. The value 0x40 indicates that the ratio of the long frame exposure time to the short frame exposure time is 1. The default value is 0x40**.** Value range: [0x40, u32ExpRatioMax]

Member	Description
u16Tolerance	Exposure ratio tolerance value. This member is valid only in 2-frame combination WDR mode. When **enExpRatioType** is **OP_TYPE_AUTO**, a larger value indicates lower sensitivity to the scenario changes, that is, the exposure ratio changes slightly. When the dynamic change of the scenario falls within a certain range, the exposure ratio remains unchanged. The default value is 0xC. Value range: [0x0, 0xFF]
u16Speed	Speed of the automatic exposure ratio adjustment. This member is valid only in 2-frame combination WDR mode. When **enExpRatioType** is **OP_TYPE_AUTO**, a larger value indicates a higher speed of automatic exposure ratio adjustment. The default value is 0x20**.** Value range: [0x0, 0xFF]
u16RatioBias	Exposure ratio tolerance value. This member is valid only in 2-frame combination WDR mode. When **enExpRatioType** is **OP_TYPE_AUTO**, a larger value indicates that the automatic exposure ratio is greater. The default value is 0x400, which indicates that the calculation result of the automatic exposure ratio algorithm is not adjusted. The exposure ratio adjusted by **u16RatioBias** is limited by the maximum/minimum exposure ratio. Value range: [0x0, 0xFFFF]

[Note]

- It is recommended that the **u16Tolerance** should not be set to a too large value, so that the case that exposure ratio cannot correctly reflect the change of dynamic scenario range can be avoided.
- It is recommended that the **u16Speed** should not be set smaller than 0x8, so the case that exposure ratio adjustment is slow or fails to work due to the low calculation precision in some scenarios can be avoided. When **u16Speed** is too large, the exposure ratio may change too rapidly. As a result, the picture luminance changes significantly.

[See Also]

- HI_MPI_ISP_SetWDRExposureAttr
- HI_MPI_ISP_GetWDRExposureAttr

ISP_AE_ROUTE_NODE_S

[Description]

Defines the attributes of the AE route node.

[Syntax]

```
typedef struct hiISP_AE_ROUTE_NODE_S
{
    HI_U32  u32IntTime;
```

```
    HI_U32  u32SysGain;
    ISP_IRIS_F_NO_E  enIrisFNO;
    HI_U32  u32IrisFNOLin;
} ISP_AE_ROUTE_NODE_S;
```

[Member]

Member	Description
u32IntTime	Node exposure time Value range: (0x0, 0xFFFFFFFF]
u32SysGain	Node gain (10-bit precision), including the sensor analog gain, sensor digital gain, and ISP digital gain Value range: [0x400, 0xFFFFFFFF]
enIrisFNO	F value of the node iris. Only the P iris is supported. Value range: [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
u32IrisFNOLin	Equivalent gain of the node iris F value. Only the P-iris is supported. Value range: [1, 1024]

[Note]

- The node exposure is the product of the exposure time, gain, and iris. The node exposure must be monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest. During exposure calculation, the iris F value needs to be converted into the equivalent gain as follows: Equivalent gain FNo = 1 << ISP_IRIS_F_NO_*XX_XX*. Therefore, F32.0 corresponds to gain 1, F22.0 corresponds to gain 2, F16.0 corresponds to gain 4, …, F1.0 corresponds to gain 1024.
- **bFNOExValid** in the ISP_PIRIS_ATTR_S can be configured to determine whether **enIrisFNO** or **u32IrisFNOLin** is used as the AE route node iris value that actually takes effect. If **bFNOExValid** is **HI_TRUE**, the high-precision **u32IrisFNOLin** will be used. **enIrisFNO** is used by default. **enIrisFNO** is restricted by **enMaxIrisFNOTarget/enMinIrisFNOTarget** in ISP_PIRIS_ATTR_S. **u32IrisFNOLin** is restricted by **u32MaxIrisFNOTarget/u32MinIrisFNOTarget** in ISP_PIRIS_ATTR_S. In addition, **enIrisFNO** and **u32IrisFNOLin** are restricted by **enMaxIrisFNO/enMinIrisFNO** in AE_SENSOR_DEFAULT_S. Therefore, a proper value needs to be assigned to **enMaxIrisFNO/enMinIrisFNO** in **cmos.c** when the P iris is connected.
- The nodes cannot be configured to have the same exposure value.
- To ensure the exposure of the exposure node, when one component of the exposure node is restricted, other components that have not reached the maximum value will be adjusted. The route that takes effect may be inconsistent with the configured route. In addition, to avoid exposure overflow, the maximum product value of the exposure time and gain of a node cannot exceed 0x1FFFFFFFFFFFFF if the iris component is enabled, and cannot exceed 0x7FFFFFFFFFFFFFFF if the iris component is disabled.
- If the exposure of adjacent nodes increases, the value of a component should increase while the values of other components are retained. The component with

increased value determines the allocation policy of the route. For example, if the value of the gain component increases, the allocation policy of the route is gain first.

- Only the P iris is supported. The DC iris is not supported because it cannot be accurately controlled.
- For the P iris, you are advised to set the exposure time, gain, and iris F value of the first node to the corresponding minimum target values, and set those of the last node to the corresponding maximum target values.

[See Also]

HI_MPI_ISP_SetAERouteAttr

ISP_AE_ROUTE_S

[Description]

Defines the attributes of the ISP exposure allocation policy.

[Syntax]

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

[Member]

Member	Description
u32TotalNum	Number of nodes on the exposure allocation route. A maximum of 16 nodes are supported currently.
astRouteNode [ISP_AE_ROUTE_MAX_NODES]	Attributes of the nodes on the exposure allocation route

[Note]

- A maximum of 16 nodes are supported. Each node has the exposure time, gain, and iris components. The gain includes the sensor analog gain, sensor digital gain, and ISP digital gain.
- You can set routes based on the application scenario. The allocation route can be dynamically switched.
- According to the default AE allocation policy for the DC iris and manual iris, the exposure time is allocated first, and then the gain is allocated. If the current exposure does not fall within the configured route range, the default allocation policy is used.
- According to the default AE allocation policy for the P iris, the iris is adjusted to the maximum size, and then the exposure time and gain are allocated in sequence. If the current exposure does not fall within the configured route range, the default allocation policy is used.
- When the DC iris and P iris are switched in online mode, the AE route is reset to the default allocation policy that matches the iris type. You can configure the AE route as

required when switching the iris type. During the switching, the AE route parameter of the default allocation policy matches the maximum/minimum value of the AE time, gain and iris component. Note that the AE allocation route can also be set in **cmos.c**. When the DC iris and P iris are switched in online mode, do not set **enIrisType** in **cmos.c**. In this case, the iris type configured by the AE algorithm is sent to **cmos.c** and then cmos_get_ae_default or cmos_fps_set is called to obtain the AE route that matches the iris type.

[See Also]

HI_MPI_ISP_SetAERouteAttr

ISP_AE_ROUTE_EX_NODE_S

[Description]

Defines the attributes of the node on the extended AE route.

[Syntax]

```
typedef struct hiISP_AE_ROUTE_EX_NODE_S
{
    HI_U32  u32IntTime;
    HI_U32  u32Again;
    HI_U32  u32Dgain;
    HI_U32  u32IspDgain;
    ISP_IRIS_F_NO_E  enIrisFNO;
    HI_U32  u32IrisFNOLin;
} ISP_AE_ROUTE_EX_NODE_S;
```

[Member]

Member	Description
u32IntTime	Node exposure time (in μs) Value range: (0x0, 0xFFFFFFFF]
u32Again	Sensor analog gain (10-bit precision) Value range: [0x400, 0x3FFFFF]
u32Dgain	Sensor digital gain (10-bit precision) Value range: [0x400, 0x3FFFFF]
u32IspDgain	ISP digital gain (10-bit precision) Value range: [0x400, 0x40000]
enIrisFNO	F value of the node iris. Only the P iris is supported. Value range: [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
u32IrisFNOLin	Equivalent gain of the node iris F value. Only the P-iris is supported. Value range: [1, 1024]

[Note]

- The node exposure is the product of the exposure time, sensor analog gain, sensor digital gain, ISP digital gain, and iris. The node exposure must be monotonically increasing. To be specific, the exposure of a node is greater than or equal to that of the previous node. The exposure of the first node is the lowest, and the exposure of the last one is the highest. During exposure calculation, the F value of the iris needs to be converted into the equivalent gain as follows: Equivalent gain FNo = 1 << ISP_IRIS_F_NO_*XX_XX*. Therefore, F32.0 corresponds to gain 1, F22.0 corresponds to gain 2, F16.0 corresponds to gain 4, …, F1.0 corresponds to gain 1024.
- **bFNOExValid** in the ISP_PIRIS_ATTR_S structure can be configured to determine whether **enIrisFNO** or **u32IrisFNOLin** is used as the extended AE route node iris value that actually takes effect. If **bFNOExValid** is **HI_TRUE**, the high-precision **u32IrisFNOLin** will be used. **enIrisFNO** is used by default. **enIrisFNO** is restricted by **enMaxIrisFNOTarget/enMinIrisFNOTarget** in ISP_PIRIS_ATTR_S. **u32IrisFNOLin** is restricted by **u32MaxIrisFNOTarget/u32MinIrisFNOTarget** in ISP_PIRIS_ATTR_S. In addition, **enIrisFNO** and **u32IrisFNOLin** are restricted by **enMaxIrisFNO/enMinIrisFNO** in AE_SENSOR_DEFAULT_S. Therefore, a proper value needs to be assigned to **enMaxIrisFNO/enMinIrisFNO** in **cmos.c** when the P iris is connected.
- The nodes cannot be configured to have the same exposure value.
- To ensure the exposure of the exposure node, when one component of the exposure node is restricted, other components that have not reached the maximum value will be adjusted. The route that takes effect may be inconsistent with the configured route.
- If the exposure of adjacent nodes increases, the value of a component should increase while the values of other components remain unchanged. The component with an increased value determines the allocation policy of the route. For example, if the value of the sensor analog gain component increases, the allocation policy of the route is sensor analog gain first (the sensor analog gain is allocated first).
- Only the P iris is supported. The DC-iris is not supported because it cannot be precisely controlled.
- For the P iris, you are advised to set the exposure time, gain, and iris F value of the first node to the corresponding minimum target values, and set those of the last node to the corresponding maximum target values.
- The node exposure time cannot be set to 0. The node exposure time cannot be too small; otherwise, the number of exposure lines corresponding to the exposure time (in μs) is 0 and exceptions may occur.
- The node gain range for the extended AE route is smaller than that for the common AE route. In addition, to avoid exposure overflow, the 10-bit system gain equivalent to the product of the sensor analog gain, sensor digital gain, and ISP digital gain cannot be greater than 0xFFFFFFFF. The maximum product value of the exposure time and the 10-bit system gain of a node cannot exceed 0x1FFFFFFFFFFFF if the iris component is enabled, and cannot exceed 0x7FFFFFFFFFFFFFF if the iris component is disabled.
- If the allocation policy of some routes needs to be set to ISP digital gain first, you are advised not to set the ISP digital gain to the maximum value because the ISP digital gain needs to be used to compensate for allocation precision. It is recommended that a margin of at least 1/2 of the maximum ISP digital gain be reserved for precision compensation. To be specific, **ISPDgain** of the node on the allocation route cannot be greater than **MaxISPDgain**/2.

[See Also]

HI_MPI_ISP_SetAERouteAttrEx

ISP_AE_ROUTE_EX_S

[Description]

Defines the extended attribute of AE allocation policy.

[Syntax]

```
typedef struct hiISP_AE_ROUTE_EX_S
{
    HI_U32 u32TotalNum;
    ISP_AE_ROUTE_EX_NODE_S astRouteExNode[ISP_AE_ROUTE_EX_MAX_NODES];
} ISP_AE_ROUTE_EX_S;
```

[Member]

Member	Description
u32TotalNum	Number of nodes on the extended exposure allocation route. A maximum of 16 nodes are supported currently.
astRouteExNode [ISP_AE_ROUTE_EX_MAX_NODES]	Attributes of the node on the extended exposure allocation route

[Note]

- At most 16 nodes are supported. Each node has five components, including the exposure time, sensor analog gain, sensor digital gain, ISP digital gain, and iris.
- You can set the route based on the application scenario. The allocation route can be dynamically switched.
- According to the default extended AE allocation policy for the DC iris and manual iris, the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain are allocated in sequence. If the current exposure does not fall within the configured route range, the default allocation policy is used.
- According to the default extended AE allocation policy for the P iris, the iris is adjusted to the maximum size, and then the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain are allocated in sequence. If the current exposure does not fall within the configured route range, the default allocation policy is used.
- When the DC iris and P iris are switched in online mode, the extended AE route is reset to the default allocation policy that matches the iris type. You can configure the extended AE route as required when switching the iris type. During the switching, the AE route parameter of the default allocation policy matches the maximum/minimum value of the AE time, the gain and the iris component. Note that the AE allocation route can also be set in **cmos.c**. When the DC iris and P iris are switched in online mode, do not set **enIrisType** in **cmos.c**. In this case, the iris type configured by the AE algorithm is sent to **cmos.c** and then cmos_get_ae_default or cmos_fps_set is called back to obtain the extended AE route that matches the iris type.

[See Also]

HI_MPI_ISP_SetAERouteAttrEx

ISP_EXP_INFO_S

[Description]

Defines the internal exposure status information about the ISP.

[Syntax]

```
typedef struct hiISP_EXP_INFO_S
{
    HI_U32 u32ExpTime;
    HI_U32 u32LongExpTime;
    HI_U32 u32AGain;
    HI_U32 u32DGain;
    HI_U32 u32ISPDGain;
    HI_U32 u32Exposure;
    HI_BOOL bExposureIsMAX;
    HI_S16 s16HistError;
    HI_U32 u32AE_Hist256Value[256];
    HI_U16 u16AE_Hist5Value[5];
    HI_U8  u8AveLum;
    HI_U32 u32LinesPer500ms;
    HI_U32 u32PirisFNO;
    HI_U32 u32Fps;
    HI_U32 u32ISO;
    HI_U32 u32RefExpRatio;
    HI_U32 u32FirstStableTime;
    ISP_AE_ROUTE_S stAERoute;
    ISP_AE_ROUTE_EX_S stAERouteEx;
}ISP_EXP_INFO_S;
```

[Member]

Member	Description
u32ExpTime	Current exposure time (in μs) Value range: (0x0, 0xFFFFFFFF]
u32LongExpTime	Current long-frame exposure time in FSWDR mode (in μs) Value range: (0x0, 0xFFFFFFFF]
u32AGain	Current sensor analog gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF]
u32DGain	Current sensor digital gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF]
u32ISPDGain	Current ISP digital gain (10-bit decimal precision) Value range: [0x400, 0xFFFFFFFF]

Member	Description
u32Exposure	Current light exposure. It is the exposure time (in line) multiplied by the exposure gain (6-bit decimal precision). Value range: [0x40, 0xFFFFFFFF]
bExposureIsMAX	0: The ISP does not reach the maximum exposure performance. 1: The ISP reaches the maximum exposure performance.
s16HistError	Difference 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: [−0x8000, +0x7FFF]
u32AE_Hist256Value[256]	Global 256-segment histogram Value range: [0x0, 0xFFFFFFFF]
u16AE_Hist5Value[5]	Global 5-segment histogram Value range: [0x0, 0xFFFF]
u8AveLum	Average luminance Value range: [0x0, 0xFF]
u32LinesPer500ms	Number of exposure lines within every 500 ms. This member is used for converting the exposure time in the unit of μs into that in the unit of line. Value range: [0x0, 0xFFFFFFFF]
u32PirisFNO	Equivalent gain corresponding to the current F value of the P iris Value range: [0x0, 0x400]
u32Fps	Actual picture frame rate multiplied by 100
u32ISO	Product of the sensor analog gain, sensor digital gain, ISP digital gain, and 100. The gains are in 0-bit precision. Value range: [0x64, 0xFFFFFFFF]
u32RefExpRatio	Reference exposure ratio. This member is used to estimate the dynamic range of the current scenario and affected by the values of **Tolerance** and **Speed** in ISP_WDR_EXPOSURE_ATTR_S. Value range: [0x40, 0xFFF]
u32FirstStableTime	First time when AE convergence is stable (in μs)
stAERoute	Actual AE route that takes effect. The exposure time in each node is in the unit of μs, and the gain is in 10-bit precision. The value range of the iris is [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]. When the iris type is DC iris, the node iris value has no effect on exposure allocation.

Member	Description
stAERouteEx	Actual extended AE route that takes effect. The exposure time in each node is in the unit of μs, and the gain is in 10-bit precision. The value range of the iris is [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]. When the iris type is DC iris, the node iris value has no effect on exposure allocation.

[Note]

- The iris status is not considered during exposure calculation. The exposure is the product of the exposure time, sensor analog gain, sensor digital gain, and ISP digital gain. The exposure time is in the unit of exposure line, and the exposure gain is in 6-bit decimal precision. If the precision of the exposure fails to meet the requirements, you can recalculate an exposure based on the high-precision exposure time (in μs) and gain (10-bit decimal precision).
- The iris status is not considered for checking whether the ISP reaches the maximum exposure performance. If the current exposure time is greater than or equal to the maximum target exposure time and the current gain is greater than or equal to the maximum target gain, the ISP is considered to reach the maximum exposure performance. Otherwise, the ISP does not reach the maximum exposure performance.
- The 5-segment histogram is normalized, and its value range is 0–65535. The 256-segment histogram is not normalized.
- The average image luminance is normalized, and the value range is 0–255.
- **u32Fps** is used for querying the actual frame rate during automatic frame rate reduction.
- **u32ISO** and **u32RefExpRatio** can be considered as the reference values when the long frame mode is changed to in FSWDR mode. Note that if **u32ISO** is the reference value, the effect of the exposure ratio on the switching threshold should be considered to avoid frequent switching between the long frame mode and the common WDR mode. If **u32RefExpRatio** is the reference value, the bright region may be overexposed in long frame mode and **u32RefExpRatio** cannot correctly indicate the dynamic scenario range, which affects the switching from the long frame mode to the common WDR mode.

[See Also]

HI_MPI_ISP_QueryExposureInfo

3.5.3 AI

The following are AI data structures:

- ISP_OP_TYPE_E: Defines the AI mode.
- ISP_IRIS_STATUS_E: Defines the ISP iris status.
- ISP_IRIS_TYPE_E: Defines the ISP iris type.
- ISP_IRIS_F_NO_E: Defines the F value of the ISP iris.
- ISP_MI_ATTR_S: Defines the MI attribute.
- ISP_DCIRIS_ATTR_S: Defines the attributes of the AI algorithms for the DC iris.
- ISP_PIRIS_ATTR_S: Defines the attributes of the P iris.
- ISP_IRIS_ATTR_S: Defines the iris attribute.

ISP_OP_TYPE_E

[Description]

Defines the AI mode.

[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 used for debugging.

[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.

Member	Description
ISP_IRIS_CLOSE	The iris is closed.

[Note]

When the iris status is set to **ISP_IRIS_OPEN** or **ISP_IRIS_CLOSE**, the iris is completely opened or closed. In this case, you can check whether the AI circuit and driver are correct. The configurations of **ISP_IRIS_OPEN** and **ISP_IRIS_CLOSE** takes priority over the configurations of the AI enable and AI mode (manual or automatic). When the AI algorithms are running, the ISP iris status must be set to **ISP_IRIS_KEEP** to ensure that the iris works properly.

[See Also]

None

ISP_IRIS_TYPE_E

[Description]

Defines the ISP iris type.

[Syntax]

```
typedef enum hiISP_IRIS_TYPE_E
{
    ISP_IRIS_DC_TYPE = 0,
    ISP_IRIS_P_TYPE,
    ISP_IRIS_TYPE_BUTT,
} ISP_IRIS_TYPE_E;
```

[Member]

Member	Description
ISP_IRIS_DC_TYPE	DC iris
ISP_IRIS_P_TYPE	P iris

[Note]

- The AI algorithms work properly only after the iris type is correctly configured.
- If the connected lens uses the manual iris, ISP_IRIS_TYPE_E can be set to **ISP_IRIS_DC_TYPE**, and you are advised to disable AI.
- Before the iris is switched from a DC iris to a P iris, parameters related to the P iris need to be configured and the iris status needs to be set to **ISP_IRIS_KEEP**.

[See Also]

None

ISP_IRIS_F_NO_E

[Description]

Defines the F value of the ISP iris.

[Syntax]

```
typedef enum hiISP_IRIS_F_NO_E
{
    ISP_IRIS_F_NO_32_0 = 0,
    ISP_IRIS_F_NO_22_0,
    ISP_IRIS_F_NO_16_0,
    ISP_IRIS_F_NO_11_0,
    ISP_IRIS_F_NO_8_0,
    ISP_IRIS_F_NO_5_6,
    ISP_IRIS_F_NO_4_0,
    ISP_IRIS_F_NO_2_8,
    ISP_IRIS_F_NO_2_0,
    ISP_IRIS_F_NO_1_4,
    ISP_IRIS_F_NO_1_0,
    ISP_IRIS_F_NO_BUTT,
} ISP_IRIS_F_NO_E;
```

[Member]

Member	Description
ISP_IRIS_F_NO_32_0	F32.0 iris
ISP_IRIS_F_NO_22_0	F22.0 iris
ISP_IRIS_F_NO_16_0	F16.0 iris
ISP_IRIS_F_NO_11_0	F11.0 iris
ISP_IRIS_F_NO_8_0	F8.0 iris
ISP_IRIS_F_NO_5_6	F5.6 iris
ISP_IRIS_F_NO_4_0	F4.0 iris
ISP_IRIS_F_NO_2_8	F2.8 iris
ISP_IRIS_F_NO_2_0	F2.0 iris
ISP_IRIS_F_NO_1_4	F1.4 iris
ISP_IRIS_F_NO_1_0	F1.0 iris

[Note]

For the P iris, when the AE algorithm calculates the exposure based on the allocation route, the iris F value needs to be converted into the equivalent gain as follows: Equivalent gain FNo

= 1 << ISP_IRIS_F_NO_*XX_XX*. Therefore, F32.0 corresponds to gain 1, F22.0 corresponds to gain 2, F16.0 corresponds to gain 4, …, F1.0 corresponds to gain 1024.

[See Also]

None

ISP_MI_ATTR_S

[Description]

Defines the MI attribute.

[Syntax]

```
typedef struct hiISP_MI_ATTR_S
{
   HI_U32  u32HoldValue;
   ISP_IRIS_F_NO_E  enIrisFNO;
} ISP_MI_ATTR_S;
```

[Member]

Member	Description
u32HoldValue	AI correction value, for debugging the DC iris Value range: [0x0, 0x3E8]
enIrisFNO	Manual iris size. The irises are distinguished based on the F value. Only the P iris is supported. Value range: [0x0, 0x64]

[Note]

- When the connected lens uses the DC iris, if ISP_IRIS_STATUS_E is set to **ISP_IRIS_KEEP**, the manual iris is enabled and **u32HoldValue** can be used for debugging the DC iris. In this case, the PWM duty cycle is **u32HoldValue**.
- When the connected lens uses the P iris, if ISP_IRIS_STATUS_E is set to **ISP_IRIS_KEEP**, the manual exposure mode and the manual iris are enabled and **enIrisFNO** can be used for debugging the P iris. In this case, the stepper motor of the P iris is controlled to move to the position where the iris F value is closest to **enIrisFNO**. In AE mode, the manual P iris does not take effect. To fix the iris F value at a certain value, set **enMaxIrisFNOTarget** and **enMinIrisFNOTarget** to the same value.

[See Also]

None

ISP_DCIRIS_ATTR_S

[Description]

Defines the attributes of the AI algorithms for the DC iris.

[Syntax]

```
typedef struct hiISP_DCIRIS_ATTR_S
{
    HI_S32 s32Kp;
    HI_S32 s32Ki;
    HI_S32 s32Kd;
    HI_U32 u32MinPwmDuty;
    HI_U32 u32MaxPwmDuty;
    HI_U32 u32OpenPwmDuty;
} ISP_DCIRIS_ATTR_S;
```

[Member]

Member	Description
s32Kp	Proportional gain. This member is used for adjusting the speed of enabling/disabling the iris. A larger value indicates a higher speed. If the value is too small, the iris is repeatedly enabled and disabled during the convergence when the light greatly varies. If the value is too large, the iris is repeatedly enabled and disabled due to over-regulation. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 7000 Value range: [0, 100000]
s32Ki	Integral gain. This member is used for adjusting the speed of enabling/disabling the iris. A larger value indicates a higher speed. If the value is too small, the picture luminance is low after convergence due to the highlight. If the value is too large, the iris may fail to be disabled in the highlight scenario. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 100 (the value does not to be changed) Value range: [0, 1000]
s32Kd	Derivative gain. This member is used for limiting the speed of enabling/disabling the iris when the light greatly varies. A larger value indicates a lower speed. If the value is too large, the sensitivity to the instantaneous luminance change is too high. As a result, the picture luminance oscillates when the luminance in the scenario changes rapidly. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 3000 Value range: [0, 100000]
u32MinPwmDuty	Minimum PWM duty cycle. A smaller value indicates a higher speed of disabling the iris when overexposure occurs. However, the iris may be enabled and disabled repeatedly in this case. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 250 Value range: [0, 1000]

Member	Description
u32MaxPwmDuty	Maximum PWM duty cycle. A larger value indicates a higher speed of enabling the iris when the entire picture is dark. If the value is too small, the iris size may not reach the maximum value when iris control ends. As a result, picture noises may be obvious. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 950 Value range: [0, 1000]
u32OpenPwmDuty	PWM duty cycle when the iris is enabled. Iris control ends a period after the picture luminance becomes stable and the PWM duty cycle is greater than **u32OpenPwmDuty**. Therefore, the value cannot be too small. Otherwise, iris control ends before the iris size reaches the maximum value. As a result, picture noises may be obvious. This member needs to be set to an appropriate value based on the circuit characteristics and lens. Recommended value: 800 Value range: [0, 1000]

[Note]

- If the iris is disabled and enabled repeatedly, the speed of disabling the iris is too high. You can solve this issue by properly decreasing **s32Kp** and increasing **u32MinPwmDuty**.
- The value of **u32OpenPwmDuty** must be between **u32MinPwmDuty** and **u32MaxPwmDuty**. You need to set **u32OpenPwmDuty** to an appropriate value to ensure that the iris can be quickly enabled.

[See Also]

- HI_MPI_ISP_SetDcirisAttr
- HI_MPI_ISP_GetDcirisAttr

ISP_PIRIS_ATTR_S

[Description]

Defines the attributes of the P iris.

[Syntax]

```
typedef struct hiISP_PIRIS_ATTR_S
{
    HI_BOOL bStepFNOTableChange;
    HI_BOOL bZeroIsMax;
    HI_U16 u16TotalStep;
    HI_U16 u16StepCount;
    HI_U16 au16StepFNOTable[AI_MAX_STEP_FNO_NUM];
    ISP_IRIS_F_NO_E  enMaxIrisFNOTarget;
```

```
    ISP_IRIS_F_NO_E  enMinIrisFNOTarget;
    HI_BOOL bFNOExValid;
    HI_U32 u32MaxIrisFNOTarget;
    HI_U32 u32MinIrisFNOTarget;
} ISP_PIRIS_ATTR_S;
```

[Member]

Member	Description
bStepFNOTableChange	Flag indicating whether the mapping table for the position of the P iris stepper motor and the iris F value is updated HI_TRUE: updated HI_FALSE: not updated
bZeroIsMax	Flag indicating whether the iris is opened to the maximum size when the P iris stepper motor is in step 0. The value is related to the P-iris lens. HI_TRUE: The iris is opened to the maximum size when the stepper motor is in step 0. HI_FALSE: The iris is disabled when the stepper motor is in step 0.
u16TotalStep	Total number of steps for the P iris stepper motor. The value is related to the P-iris lens. Value range: [1, 1024]
u16StepCount	Number of available steps for the P iris stepper motor. The value is related to the P-iris lens. Value range: [1, 1024]
au16StepFNOTable	Mapping table for the position of the P iris stepper motor and the F value. The value is related to the P-iris lens. Value range: [0, 1024]
enMaxIrisFNOTarget	Maximum iris target value. This member is used for controlling the AE allocation route. The size of the iris that takes effect is related to the P-iris lens. Value range: [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
enMinIrisFNOTarget	Minimum iris target value. This member is used for controlling the AE allocation route. The size of the iris that takes effect is related to the P-iris lens. Value range: [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
bFNOExValid	Flag indicating whether the AE allocation route adopts the higher-precision equivalent gain of the iris F value when the P iris is connected. If the value is HI_TRUE, the AE allocation route adopts the higher-precision equivalent gain of the iris F value. Default value: HI_FALSE

Member	Description
u32MaxIrisFNOTarget	Target value of the equivalent gain of the maximum iris F value. This member is used for controlling the AE allocation route. The size of the iris that takes effect is related to the P-iris lens. Value range: [1, 1024]
u32MinIrisFNOTarget	Target value of the equivalent gain of the minimum iris F value. This member is used for controlling the AE allocation route. The size of the iris that takes effect is related to the P-iris lens. Value range: [1, 1024]

[Note]

- When the P-iris lens is changed in online mode, this data structure can be configured by using the corresponding MPI to transfer the parameters of the new lens to the AE algorithms and P iris drive.
- **bStepFNOTableChange** is write-only. The value of this parameter obtained by calling the MPI is always **HI_FALSE**.
- **u16TotalStep** indicates the total number of steps for the P iris stepper motor. It can be used for calibrating the start position of the iris. **u16StepCount** indicates the number of available steps for the P iris stepper motor, and it is generally smaller than **u16TotalStep**. For the positions near the end at which the iris is disabled, the error is large when the iris aperture is converted into the F value, and oscillation occurs during iris adjustment. Therefore, these positions are generally not used. If the precision of the P iris that supports at most 1024 steps is high and the number of adjustable steps exceeds 1024, you are advised not to use the positions near the end at which the iris is disabled.
- The mapping table (**au16StepFNOTable**) for the position of the P iris stepper motor and the iris F value is configured based on the mapping relationship between the stepper motor position and iris aperture, which is provided by the lens vendor. The P iris is controlled by using the AE allocation route, which requires a good linear relationship between the iris F value and exposure time as well as gain. Therefore, high precision is required for the mapping table. The value 1024 indicates F1.0, the value 512 indicates F1.4, ..., the value 1 indicates F32.0. During the preparation of the mapping table, the maximum value of the mapping table can be specified based on the actual F value corresponding to the maximum iris aperture, or the maximum iris aperture can be mapped to 1024 regardless of the actual F value when the iris is opened to the maximum size. When the focal length of a lens is changed, the value of **au16StepFNOTable** needs to be changed accordingly.
- The following describes how to configure the parameters related to the lens by taking the Forecam NV03105P P-iris lens as an example. Table 3-1 describes the mapping relationship between the stepper motor position and the aperture area, which is provided by the lens vendor. The total number of steps for the stepper motor of this lens is 93. The iris aperture area is the largest when the stepper motor is in step 0. The nominal maximum relative aperture is F1.4. Therefore, the maximum F value is 512. After the maximum F value is determined, the F value corresponding to other aperture area can be calculated based on the linear relationship. For example, the aperture area is 48.835 when the stepper motor is in step 1. The F value is 506 ((48.835/49.366) x 512). In this way, the F values corresponding to all valid stepper motor positions can be obtained. As shown in Table 3-1, the aperture area is very small (the maximum aperture

area is thousands of times this value) when the lens is close to the end at which this lens is disabled. The F value is not precise enough to reflect the change in the aperture area so that the corresponding F values are all 0. Actually, even if the precision of the mapping table is increased, the F values are inaccurate due to the huge difference between the nominal aperture area near the end at which this lens is disabled and the actual aperture area. You are advised not to use these F values; otherwise, the iris may be enabled and disabled repeatedly during adjustment. You are advised to use only the first 62 steps during the preparation of the mapping table. Table 3-2 describes the parameters related to the lens.

- Only the first **u16StepCount** elements of **au16StepFNOTable** are valid and used for the calculation of the AE algorithms. The values must be monotonically increasing until the array coordinate reaches **u16StepCount**. The maximum iris F value must be reached when the array coordinate is (**u16StepCount − 1**). When the corresponding MPI is called to configure **au16StepFNOTable**, only the first **u16StepCount** elements are written to the register. When the corresponding MPI is called to obtain **au16StepFNOTable**, the read values of only the first **u16StepCount** elements are valid, and other elements are read as 0.
- When the iris type is P iris, the HiSilicon AE algorithm calculates the maximum or minimum exposure by referring to **enMaxIrisFNOTarget** or **enMinIrisFNOTarget** (**u32MaxIrisFNOTarget** or **u32MinIrisFNOTarget** if **bFNOExValid** is **true**). Therefore, the values of these parameters should match the value of **au16StepFNOTable**. That is, the maximum or minimum value should fall within the value range of **au16StepFNOTable**.

Table 3-1 Mapping table for the position of the P iris stepper motor and the iris F value (taking the Forecam NV03105P as an example)

Stepper Motor Position	Aperture Area	F Value	Stepper Motor Position	Aperture Area	F Value	Stepper Motor Position	Aperture Area	F Value
0	49.366	512	35	20.07	208	70	0.136	1
1	48.835	506	36	19.241	200	71	0.095	1
2	48.234	500	37	18.42	191	72	0.067	1
3	47.571	493	38	17.609	183	73	0.045	0
4	46.856	486	39	16.808	174	74	0.028	0
5	46.11	478	40	16.017	166	75	0.016	0
6	45.324	470	41	15.237	158	76	0.008	0
7	44.511	462	42	14.469	150	77	0.004	0
8	43.674	453	43	13.788	143	78	0.003	0
9	42.822	444	44	12.972	135	79	0.003	0
10	41.963	435	45	12.254	127	80	0.002	0
11	41.099	426	46	11.541	120	81	0.002	0
12	40.231	417	47	10.843	112	82	0.001	0
13	39.357	408	48	10.162	105	83	0.001	0

Stepper Motor Position	Aperture Area	F Value	Stepper Motor Position	Aperture Area	F Value	Stepper Motor Position	Aperture Area	F Value
14	38.478	399	49	9.497	98	84	0.001	0
15	37.608	390	50	8.851	92	85	close	0
16	36.721	381	51	8.222	85	86	close	0
17	35.832	372	52	7.611	79	87	close	0
18	34.94	362	53	7.018	73	88	close	0
19	34.047	353	54	6.443	67	89	close	0
20	33.153	344	55	5.893	61	90	close	0
21	32.259	335	56	5.354	56	91	close	0
22	31.365	325	57	4.832	50	92	Close	0
23	30.473	316	58	4.329	45	93	M-stop	0
24	29.582	307	59	3.843	40			
25	28.706	298	60	3.376	35			
26	27.82	289	61	2.926	30			
27	26.937	279	62	2.494	26			
28	26.059	270	63	2.08	22			
29	25.184	261	64	1.684	17			
30	24.315	252	65	1.305	14			
31	23.451	243	66	0.949	10			
32	22.593	234	67	0.607	6			
33	21.741	225	68	0.374	4			
34	20.896	217	69	0.225	2			

Table 3-2 Parameters related to the P iris (taking the Forecam NV03105P as an example)

Parameter	Value	Description
bZeroIsMax	HI_TRUE	Flag indicating whether the iris aperture is the largest when the stepper motor is in step 0. Therefore, the value is HI_TRUE.
u16TotalStep	93	Total number of steps for the stepper motor

Parameter	Value	Description
u16StepCount	62	Number of available steps for the stepper motor
au16StepFNOTable	{30, 35, 40, 45, 50, 56, 61, 67, 73, 79, 85, 92, 98, 105, 112, 120, 127, 135, 143, 150, 158, 166, 174, 183, 191, 200, 208, 217, 225, 234, 243, 252, 261, 270, 279, 289, 298, 307, 316, 325, 335, 344, 353, 362, 372, 381, 390, 399, 408, 417, 426, 435, 444, 453, 462, 470, 478, 486, 493, 500, 506, 512}	Mapping table for the position of the P iris stepper motor and the iris F value. According to Table 3-1, the F values corresponding to the first 62 steps of the stepper motor are arranged in ascending order. Only the first **u16StepCount** elements of **au16StepFNOTable** are used for the calculation of the AE algorithms. The values must be monotonically increasing until the array coordinate reaches **u16StepCount**. The maximum iris F value (512) must be reached when the array coordinate is (**u16StepCount − 1**).
enMaxIrisFNOTarget	ISP_IRIS_F_NO_1_4	Maximum value of the mapping table (512), corresponding to F1.4
enMinIrisFNOTarget	ISP_IRIS_F_NO_5_6	Minimum value of the mapping table (30), corresponding to F5.6
bFNOExValid	HI_FALSE	The high-precision equivalent gain of the iris F value is not used by default.
u32MaxIrisFNOTarget	512	The maximum value 512 in the value range of **au16StepFNOTable** is used.
u32MinIrisFNOTarget	30	The minimum value 30 in the value range of **au16StepFNOTable** is used.

[See Also]

- HI_MPI_ISP_SetPirisAttr
- HI_MPI_ISP_GetPirisAttr
- HI_MPI_ISP_SetAERouteAttr
- HI_MPI_ISP_SetAERouteAttrEx

ISP_IRIS_ATTR_S

[Description]

Defines the iris attribute.

[Syntax]

```
typedef struct hiISP_IRIS_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E   enOpType;
```

```
    ISP_IRIS_TYPE_E enIrisType;
    ISP_IRIS_STATUS_E enIrisStatus;
    ISP_MI_ATTR_S   stMIAttr;
} ISP_IRIS_ATTR_S;
```

[Member]

Member	Description
bEnable	AI enable
enOpType	Iris mode (AI mode or MI mode)
enIrisType	Iris type (DC iris or P iris)
enIrisStatus	Iris status
stMIAttr	MI attribute

[Note]

- Before testing the iris by using the AI algorithms, you are advised to check whether the AI circuit characteristics meet the requirements of the HiSilicon IPC.
- When **bEnable** is **HI_FALSE**, the iris type cannot be changed. Therefore the iris type retains the previous value.
- For the DC iris, the AI algorithms control the iris by adjusting the PWM duty cycle based on the picture luminance. When the exposure time and gain reach the minimum target values, the algorithms start iris control. If the iris control meets the requirements of the target luminance, the AE exits directly and the exposure time and gain remain unchanged. After the picture luminance becomes stable and the PWM duty cycle retains the value when the iris is opened for a while, the AI algorithms consider that the iris has been opened to the maximum size, end the iris control, and hand over the control to the AE algorithms. During iris control, if AE algorithm parameters that need to take effect immediately (for example, the maximum/minimum exposure time, maximum/minimum gain, and anti-flicker parameter) are changed, the AE algorithm responds instantaneously. Then the AI algorithm determines whether to start iris control based on the configured parameters and ambient luminance. Starting or ending iris control takes a short period of time. Therefore, you are advised to disable AI when the manual iris is used; otherwise, the AE adjustment speed will be affected. It is recommended that AI be always enabled for the DC iris because disabling and enabling AI may result in iris control exceptions. For some telephoto DC-iris lenses, the default parameter values may cause the iris to be enabled or disabled too quickly. This issue can be solved by adjusting related parameters. For details, see the description of ISP_DCIRIS_ATTR_S.
- The P iris is controlled by using the AE allocation route. To ensure that the P iris drive connects to the corresponding lens and the P iris works properly, the parameters related to the lens and the AE allocation route must be correctly configured. For details, see ISP_PIRIS_ATTR_S and HI_MPI_ISP_SetAERouteAttr. The drive mode may differ according to the specifications of the P iris. You can modify the P iris drive to adapt to different lenses.
- If the AI function is disabled, the DC iris will be opened to the maximum size, whereas the P iris will be opened to the maximum iris target value, which corresponds to the position of the stepper motor. However, the exposure allocation still refers to the AE route, which may cause picture luminance exceptions. Therefore, when the P-iris lens is

connected, if you do not want to enable AI, switch the iris type to ISP_IRIS_DC_TYPE to ensure that the exposure is normal.

- Before the iris is switched from a DC iris to a P iris, parameters related to the P iris need to be configured and the iris status needs to be set to **ISP_IRIS_KEEP**.
- If the DC iris is tested on the HiSilicon demo board, you need to run **himm 0x200F00A4 0x2**, **himm 0x20030038 0x6** and **himm 0x2021007c 0x0** to enable three registers when loading the .ko driver. When the DC iris is tested on the HiSilicon reference board, you need to run **himm 0x200F00DC 0x1** and **himm 0x20030038 0x6** to enable two registers when loading the .ko driver. When the DC iris is connected, it is recommended that the preceding registers be enabled when the .ko drivers are loaded. When the ISP is running, the preceding register can be enabled only after HI_MPI_ISP_SetIrisAttr is called to disable AI. In this way, the DC iris can work normally.
- When the DC iris is tested on the HiSilicon demo board or reference board, the PWM number must be specified when the .ko driver of the ISP is loaded. The values of **pwm_num** for the demo board and reference board are **4** and **5** respectively. **pwm_num** is set to **5** by default in the code.
- When the P iris is tested on the HiSilicon demo board, you need to run **himm 0x20210400 0x1f** and **himm 0x2021007C 0x10** to enable two registers when loading the .ko driver.
- Huawei LiteOS does not support the kernel module loading mechanism. To achieve the effect similar to loading the **.ko** drivers in Linux, run **sdk_init.c** under **Huawei LiteOS release/ko**. In the ISP_init function in **sdk_init.c** under **release/ko**, add **stIsp_Param.u32PwmNum=n** to ISP_ModInit.

[See Also]

- HI_MPI_ISP_SetIrisAttr
- HI_MPI_ISP_GetIrisAttr

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 algorithms are 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 response to the white color in the sensor has a blue or red cast under the illuminants with 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 values or ratios of R, G, and B color channels 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.

$$aw\_rg_{mn} = \sum_{p \epsilon \Omega_{mn}} \frac{G_p}{R_p} * \theta_p$$

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

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 weighted average global statistics are output.

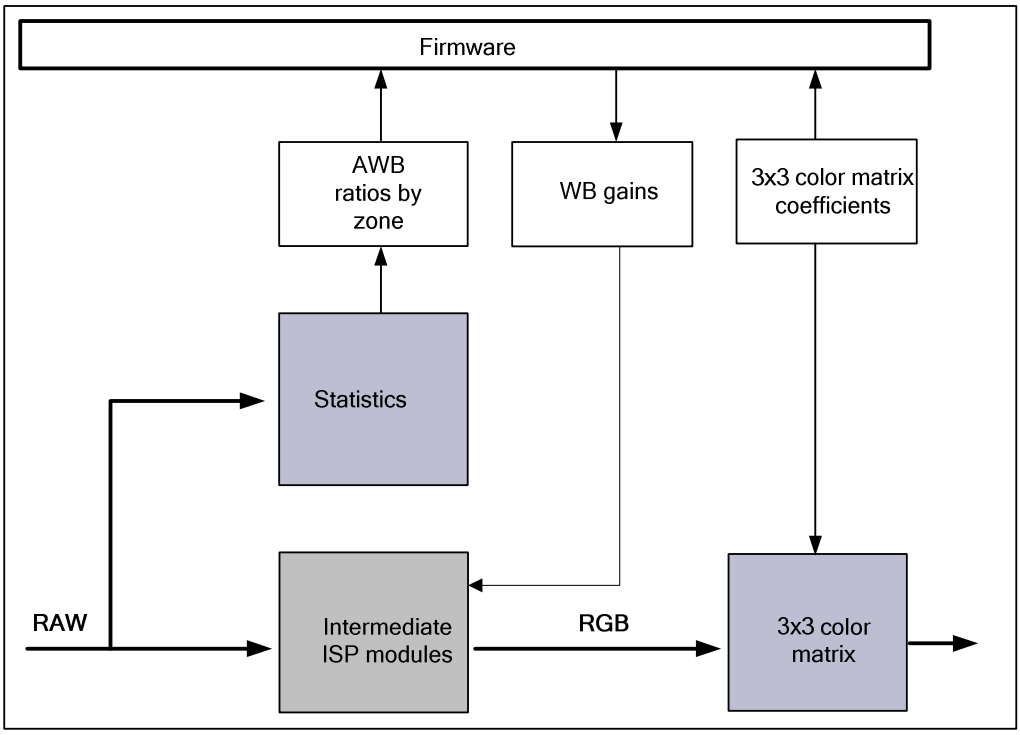
The statistics provided by the AWB module mainly include R/G and B/G values. The R/G and B/G values are average values. The average values are calculated based on the total R/G and B/G values of white points in each zone and AWB weight coefficient by using the following formula:

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

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

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

Figure 4-1 Schematic diagram of the AWB module



4.4 API Reference

4.4.1 AWB Algorithm Library MPIs

All the AWB algorithm library MPIs are based on the HiSilicon AWB algorithm library. These MPIs can be ignored and are unavailable if you use your own AWB algorithm library.

- HI_MPI_AWB_Register: Registers the AWB algorithm library with the ISP.
- HI_MPI_AWB_UnRegister: Deregisters the AWB algorithm library from the ISP.
- HI_MPI_AWB_SensorRegCallBack: Registers a sensor. This callback function is provided by the AWB algorithm library.
- HI_MPI_AWB_SensorUnRegCallBack: Deregisters a sensor. This callback function is provided by the AWB algorithm library.

HI_MPI_AWB_Register

[Description]

Registers the AWB algorithm library with the ISP.

[Syntax]

```
HI_S32 HI_MPI_AWB_Register(ISP_DEV IspDev, ALG_LIB_S *pstAwbLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

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

[Note]

- This MPI calls the AWB registration callback function HI_MPI_ISP_AWBLibRegCallBack provided by the ISP library to register the AWB algorithm library with the ISP library.

- You can call this MPI to register the HiSilicon AWB algorithm library with the ISP library.
- Multiple instances can be registered with the AWB algorithm library.

[Example]

None

[See Also]

HI_MPI_ISP_AWBLibRegCallBack

HI_MPI_AWB_UnRegister

[Description]

Deregisters the AWB algorithm library from the ISP.

[Syntax]

```
HI_S32 HI_MPI_AWB_UnRegister(ISP_DEV IspDev, ALG_LIB_S *pstAwbLib);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

- Header files: hi_comm_isp.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

- This MPI calls the AWB deregistration callback function HI_MPI_ISP_AWBLibRegCallBack provided by the ISP library to deregister the AWB algorithm library from the ISP library.
- You can call this MPI to deregister the HiSilicon AWB algorithm library from the ISP library.

[Example]

None

[See Also]

HI_MPI_ISP_AWBLibRegCallBack

HI_MPI_AWB_SensorRegCallBack

[Description]

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

[Syntax]

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

[Parameter]

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

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

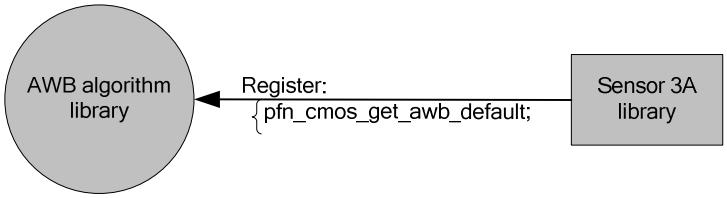
[Requirement]

- Header files: hi_comm_isp.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Requirement]

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

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



[Example]

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

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

ISP_DEV IspDev = 0;
stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AWB_LIB_NAME);
s32Ret = HI_MPI_AWB_SensorRegCallBack(&stLib, IMX178_ID, &stAwbRegister);
if (s32Ret)
{
   printf("sensor register callback function to awb lib failed!\n");
   return s32Ret;
}
```

[See Also]

None

HI_MPI_AWB_SensorUnRegCallBack

[Description]

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

[Syntax]

```
HI_S32 HI_MPI_AWB_SensorUnRegCallBack (ISP_DEV IspDev, ALG_LIB_S *pstAwbLib,
SENSOR_ID SensorId);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAwbLib	Pointer to the data structure of the AWB algorithm library	Input

Parameter	Description	Input/Output
SensorId	ID of the sensor that is registered with the AWB algorithm library	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Requirement]

- Header files: hi_comm_isp.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

None

[See Also]

None

4.4.2 AWB Control MPIs

The following are AWB control MPIs:

- HI_MPI_ISP_SetWBAttr: Sets the white balance attribute.
- HI_MPI_ISP_GetWBAttr: Obtains the white balance attribute.
- HI_MPI_ISP_SetAWBAttrEx: Sets the extended AWB attribute.
- HI_MPI_ISP_GetAWBAttrEx: Obtains the extended AWB attribute.

HI_MPI_ISP_SetWBAttr

[Description]

Sets the white balance attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetWBAttr(ISP_DEV IspDev, const ISP_WB_ATTR_S *pstWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstWBAttr	White balance attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_ILLEGAL_PARAM	The parameter is invalid.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

- If the white balance control mode is set to automatic mode, the white balance coefficient is automatically adjusted based on the white balance algorithm.
- If the white balance control mode is set to manual mode, the AWB algorithms are invalid. In this case, you must set **Rgain**, **Ggain**, and **Bgain**.
- When the white balance bypass function is enabled, the AWB algorithms do not work. In this case, **Rgain**, **Ggain**, and **Bgain** are **0x100** (8-bit decimal).

[Example]

None

[See Also]

HI_MPI_ISP_GetWBAttr

HI_MPI_ISP_GetWBAttr

[Description]

Obtains the white balance attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetWBAttr(ISP_DEV IspDev, ISP_WB_ATTR_S *pstWBAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

Parameter	Description	Input/Output
pstWBAttr	White balance attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetWBAttr

HI_MPI_ISP_SetAWBAttrEx

[Description]

Sets the extended AWB attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAWBAttrEx(ISP_DEV IspDev, ISP_AWB_ATTR_EX_S
*pstAWBAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

Parameter	Description	Input/Output
pstAWBAttrEx	Advanced AWB attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

This MPI is valid only when the **pstWBAttr->enAlgType** member in HI_MPI_ISP_SetWBAttr is **AWB_ALG_ADVANCE**.

[Example]

None

[See Also]

HI_MPI_ISP_GetAWBAttrEx

HI_MPI_ISP_GetAWBAttrEx

[Description]

Obtains the extended AWB attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAWBAttrEx(ISP_DEV IspDev, ISP_AWB_ATTR_EX_S
*pstAWBAttrEx);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input

Parameter	Description	Input/Output
pstAWBAttrEx	Advanced AWB attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetAWBAttrEx

4.4.3 WB Consult MPI

- HI_MPI_ISP_QueryWBInfo: Obtains the current white balance gain coefficient, color temperature, saturation, and CCM coefficient.
- HI_MPI_ISP_CalGainByTemp: Calculates the gain coefficient of the white balance at a specific color temperature.

HI_MPI_ISP_QueryWBInfo

[Description]

Obtains the current white balance gain coefficient, color temperature, saturation, and CCM coefficient.

[Syntax]

```
HI_S32 HI_MPI_ISP_QueryWBInfo(ISP_DEV IspDev, ISP_WB_INFO_S *pstWBInfo);
```

[Parameter]

Parameter	Description	Input/Output
pstWBInfo	Color description parameter	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

None

[Example]

None

[See Also]

None

HI_MPI_ISP_CalGainByTemp

[Description]

Calculates the gain coefficient of the white balance at a specific color temperature.

[Syntax]

```
HI_S32 HI_MPI_ISP_CalGainByTemp(const ISP_WB_ATTR_S *pstWBAttr, HI_U16
u16ColorTemp, HI_S16 s16Shift, HI_U16 *pu16AWBGain);
```

[Parameter]

Parameter	Description	Input/Output
pstWBAttr	WB attributes	Input
u16ColorTemp	Color temperature (in °K) Value range: [1500, 15000]	Input
s16Shift	Position and distance between the white point and the Planckian curve Value range: [−64, +64]	Input
pu16AWBGain	Gains of R, Gr, Gb, and B channels at the preset color temperature	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.
HI_ERR_ISP_ILLEGAL_PARAM	This parameter is invalid.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a, lib_hiawb.a

[Note]

- The calibrated WB parameters determine the accuracy of the calculation on preset color temperature gains. Therefore, the WB parameters must be calibrated and configured to ISP before this MPI is called. HI_MPI_ISP_GetWBAttr is called to obtain the calibrated result and then HI_MPI_ISP_CalGainByTemp is called to calculate the preset color temperature. During this process, the calibrated result cannot be modified.
- **bGainNorm** of ISP_AWB_ATTR_S affects the WB gains. When **bGainNorm** is 0, the minimum gain is 0x100. When **bGainNorm** is 1, the minimum gain is relevant to the black level of the sensor.
- **s16Shift** determines the position relationship between the illuminant and the Planckian curve. When **s16Shift** is a negative value, the illuminant is on the left of the Planckian curve, and the calculated WB gain has a red cast at the preset color temperature. When **s16Shift** is a positive number, the illuminant is on the right of the Planckian curve and the calculated WB gain has a green cast at the preset color temperature.

[Example]

None

[See Also]

HI_MPI_ISP_GetWBAttr

4.5 Data Structures

4.5.1 Registration

The following are registration-related data structures:

- AWB_SENSOR_REGISTER_S: Defines sensor registration information.
- AWB_SENSOR_EXP_FUNC_S: Defines the sensor callback function.
- AWB_SENSOR_DEFAULT_S: Defines the initialization parameter for the AWB algorithm library.

AWB_SENSOR_REGISTER_S

[Description]

Defines sensor registration information.

[Syntax]

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

[Member]

Member	Description
stSnsExp	Callback function for registering sensors

[Note]

This data structure is encapsulated for extension.

[See Also]

AWB_SENSOR_EXP_FUNC_S

AWB_SENSOR_EXP_FUNC_S

[Description]

Defines the sensor callback function.

[Syntax]

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

[Member]

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

[Note]

None

[See Also]

ISP_SENSOR_REGISTER_S

AWB_SENSOR_DEFAULT_S

[Description]

Defines the initialization parameter for the AWB algorithm library.

[Syntax]

```
typedef struct hiAWB_SENSOR_DEFAULT_S
{
    HI_U16  u16WbRefTemp;      /* reference color temperature for WB  */
    HI_U16  au16GainOffset[4];  /* gain offset for white balance */
    HI_S32  as32WbPara[6];     /* parameter for wb curve */
    AWB_AGC_TABLE_S stAgcTbl;
    AWB_CCM_S stCcm;
} AWB_SENSOR_DEFAULT_S;
```

[Member]

Member	Sub Member	Description
u16WbRefTemp	None	Color temperature for correcting the static white balance Value range: [0, 0xFFFF]
au16GainOffset	None	Gain of the R, Gr, Gb, and B color channels for static white balance Value range: [0, 0xFFFF]
as32WbPara	None	White balance parameter provided by the correction tool Value range: [0, 0xFFFFFFFF]

Member	Sub Member	Description
stAgcTbl	bValid	Data validity identifier of the data structure Value: 0 or 1
	au8Saturation	Interpolation array for dynamically adjusting the saturation based on the gain Value range: [0, 255]
stCcm	u16HighColorTemp	High color temperature Value range: [0, 0xFFFF]
	au16HighCCM	High color temperature CCM Value range: [0, 0xFFFF]
	u16MidColorTemp	Middle color temperature Value range: [0, 0xFFFF]
	au16MidCCM	Middle color temperature CCM Value range: [0, 0xFFFF]
	u16LowColorTemp	Low color temperature Value range: [0, 0xFFFF]
	au16LowCCM	Low color temperature CCM Value range: [0, 0xFFFF]

[Note]

The reference color temperature is environmental color temperature after static white balance correction and the actual value measured by the colorimeter needs to be provided.

[See Also]

ISP_SENSOR_EXP_FUNC_S

4.5.2 WB

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

- ISP_AWB_ATTR_S: Defines the AWB attributes of the ISP.
- ISP_AWB_CBCR_TRACK_ATTR_S: Defines the associated parameters of the Bayer field statistics.
- ISP_AWB_LUM_HISTGRAM_ATTR_S: Defines the parameters of the luminance histogram statistics for white balance.
- ISP_AWB_ALG_TYPE_E: Defines the AWB algorithm type.
- ISP_AWB_CT_LIMIT_ATTR_S: Defines the AWB gain range.
- ISP_MWB_ATTR_S: Defines the manual AWB attributes of the ISP.
- ISP_WB_ATTR_S: Defines AWB attributes.
- ISP_AWB_MULTI_LS_TYPE_E: Defines the AWB policy in the multi-illuminant scenario.

- ISP_AWB_ATTR_EX_S: Defines extended AWB attributes.
- ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S: Defines the information about a separate illuminant.
- ISP_AWB_IN_OUT_ATTR_S: Defines the parameters for determining the outdoor or indoor scenario.
- ISP_WB_INFO_S: Defines white balance, saturation, color correction information.

ISP_AWB_ATTR_S

[Description]

Defines the AWB attributes of the ISP.

[Syntax]

```
typedef struct hiISP_AWB_ATTR_S
{
   HI_BOOL bEnable;
   HI_U16 u16RefColorTemp;
   HI_U16 au16StaticWB[4];
   HI_S32 as32CurvePara[6];
   ISP_AWB_ALG_TYPE_E enAlgType;
   HI_U8 u8RGStrength;
   HI_U8 u8BGStrength;
   HI_U16 u16Speed;
   HI_U16 u16ZoneSel;
   HI_U16 u16HighColorTemp;
   HI_U16 u16LowColorTemp;
   ISP_AWB_CT_LIMIT_ATTR_S stCTLimit;
   HI_BOOL bShiftLimitEn;
   HI_U8 u8ShiftLimit;
   HI_BOOL bGainNormEn;
   ISP_AWB_CBCR_TRACK_ATTR_S stCbCrTrack;
   ISP_AWB_LUM_HISTGRAM_ATTR_S stLumaHist;
} ISP_AWB_ATTR_S;
```

[Member]

Member	Description
bEnable	AWB enable Default value: HI_TRUE
u16RefColorTemp	Color temperature of the illuminant for correcting the static white balance coefficient
au16StaticWB[4]	Static WB coefficient
as32CurvePara[6]	Adjustment curve coefficient
enAlgType	AWB algorithm type selection

Member	Description
u8RGStrength	AWB adjustment strength for the R channel
u8BGStrength	AWB adjustment strength for the B channel
u16Speed	AWB convergence speed
u16ZoneSel	AWB global or zoned algorithm select Value range: [0, 255]
U16HighColorTemp	Upper color temperature limit for the AWB algorithm
U16LowColorTemp	Lower color temperature limit for the AWB algorithm
stCTLimit	Specified gain calculation mode and gain value in manual mode when the color temperature for the AWB algorithm exceeds the color temperature range
bShiftLimitEn	Whether to enable the function of mapping the points beyond the color temperature curve to the range of the color temperature curve for the AWB algorithm
u8ShiftLimit	Specified range of the color temperature curve beyond which the points are mapped to the valid range of the color temperature curve
bGainNormEn	Gain normalization enable for the AWB algorithm
stCbCrTrack	Associated parameters of the Bayer field statistics
stLumaHist	Parameters of the luminance histogram statistics for white balance

[Difference]

None

[Note]

- When **bEnable** is **HI_TRUE**, the AWB works properly and the RGB channel gain coefficient is calculated by the AWB based on the environment color temperature. When **bEnable** is **HI_FALSE**, the AWB does not work and the RGB channel gain coefficient is fixed at the corrected static white balance coefficient.
- The values of the AWB parameters **16RefColorTemp**, **au16StaticWB[4]**, and **as32CurvePara[6]** are obtained by using the correction tool. They are the prerequisite of accurate AWB. After an optical device is replaced, it is recommended that the three groups of parameters be corrected again. **u16RefColorTemp** is the actual color temperature of the illuminant for static white balance correction. You are advised to perform static white balance correction under the D50 illuminant. **as32CurvePara[0]**, **as32CurvePara[1]**, and **as32CurvePara[2]** determine the Planckian curve. **as32CurvePara[3]**, **as32CurvePara[4]**, and **as32CurvePara[5]** determine the color temperature curve, and **as32CurvePara[4]** is fixed at 128.
- You are advised not to adjust the RG strength and BG strength for AWB in normal cases.

The AWB calibration strength can be configured by adjusting the calibration strength of the R/B channel. 0x80 is the standard strength value. You are advised to set **u8RGStrength** and **u8BGStrength** to the same value that is less than or equal to 0x80. At a low color temperature, the image has a red cast when the calibration strength is less than 0x80, and the image has a blue cast when the calibration strength is greater than 0x80. At a high color temperature, the image has a blue cast when the calibration strength is less than 0x80, and the image has a red cast when the calibration strength is greater than 0x80.

- **u16Speed** is used to control the convergence speed of the AWB gain. A larger **u16Speed** value indicates a faster AWB convergence speed and a wider range of fluctuation between frames when the illuminant is switched. A smaller **u16Speed** value indicates a slower AWB convergence speed, a narrower range of fluctuation between frames, and greater stability when the illuminant is switched.
- When **u16ZoneSel** is **0** or **255**, an algorithm similar to the gray world algorithm is used. When **16ZoneSel** is other values, the AWB algorithm filters all blocks by classification to improve the AWB precision.
- Upper and lower limits (in ºK) of the color temperature for the AWB algorithm

  The upper and lower limits are the maximum and minimum color temperatures supported by the AWB algorithm. If the actual color temperature is greater than the upper limit or less than the lower limit, the AWB cannot be completely restored, and the image color is close to the illuminant color (that is, the image has a yellow cast at a low color temperature and has a blue cast at a high color temperature).

  bShiftLimitEn determines whether the AWB gain is mapped on the Planckian curve. **u8ShiftLimit** is used to control the white block range. A smaller value indicates a narrower range of the supported illuminants and higher precision, and vice versa.
- When **bGainNormEn** is enabled, the RGB channel gain is limited to increase the signal-to-noise ratio (SNR) at a low color temperature in a low-illumination scenario.

[See Also]

- ISP_WB_ATTR_S
- ISP_AWB_CBCR_TRACK_ATTR_S
- ISP_AWB_LUM_HISTGRAM_ATTR_S

ISP_AWB_CBCR_TRACK_ATTR_S

[Description]

Defines the associated parameters of the Bayer field statistics.

[Syntax]

```
typedef struct hiISP_AWB_CBCR_TRACK_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  au16CrMax[ISP_AUTO_STENGTH_NUM];
    HI_U16  au16CrMin[ISP_AUTO_STENGTH_NUM];
    HI_U16  au16CbMax[ISP_AUTO_STENGTH_NUM];
    HI_U16  au16CbMin[ISP_AUTO_STENGTH_NUM];
} ISP_AWB_CBCR_TRACK_ATTR_S;
```

[Member]

Member	Description
bEnable	Enable for associating the parameters of the Bayer field statistics with the ambient illumination and color temperature
au16CrMax[ISP_AUTO_STENGTH_NUM]	**CrMax** values under various illuminations
au16CrMin[ISP_AUTO_STENGTH_NUM]	**CrMin** values under various illuminations
au16CbMax[ISP_AUTO_STENGTH_NUM]	**CbMax** values under various illuminations
au16CbMin[ISP_AUTO_STENGTH_NUM]	**CbMin** values under various illuminations

[Note]

- After the statistics parameters are associated with the ambient illumination and color temperature, the AWB algorithms calculate **CrMax**, **CrMin**, **CbMax**, and **CbMin** in real time based on the ambient illumination, color temperature, and the configured arrays such as **au16CrMax**, and configure the corresponding logic registers. In this case, the configurations of the preceding four parameters by using the HiSilicon PQ Tools do not take effect.
- After the statistics parameters are associated with the ambient illumination and color temperature, **CrMax** and other statistics parameters calculated by the AWB algorithms are related to the ambient color temperature. At the low color temperature, the range of the white points is wide; at the middle and high color temperatures, the range of the white points is narrow.
- You need to disable the association function before configuring **CrMax**, **CrMin**, **CbMax**, and **CbMin**.
- The values of **au16CrMax[0]**, **au16CrMin[0]**, **au16CbMax[0]**, and **au16CbMin[0]** can be determined after the AWB parameters are calibrated. After the supported color temperature range is determined, the raw pictures at high and low color temperatures are captured, and the R/G and B/G values of the white area are calculated. **au16CrMax[0]** and **au16CbMin[0]** correspond to the R/G value and the B/G value at the low color temperature respectively. **au16CrMin[0]** and **au16CbMax[0]** correspond to the R/G value and the B/G value at the high color temperature respectively. It is recommended that the value ranges of Cr and Cb be slightly greater than those of R/G and B/G in the raw picture statistics.
- You are advised to calibrate the **au16CrMax** and **au16CbMin** arrays in the low color temperature scenario (for example, under the sodium lamp). You need to the collect the R/G and B/G values of the white area under various illuminations to configure the **au16CrMax** and **au16CbMin** arrays. It is recommended that the value ranges of Cr and Cb be slightly greater than those of R/G and B/G in the raw picture statistics.
- The **au16CrMin** and **au16CbMax** arrays can be set to constants because the ambient color temperature is generally under 5000°K in the low illumination scenario.

Table 4-1 Mapping between the values of au16CrMax[16] and gains (only for reference)

au16CrMax	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CrMax [0]	1	0x130

au16CrMax	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CrMax [1]	2	0x130
au16CrMax [2]	4	0x130
au16CrMax [3]	8	0x130
au16CrMax [4]	16	0x130
au16CrMax [5]	32	0x130
au16CrMax [6]	64	0x130
au16CrMax [7]	128	0x14D
au16CrMax [8]	256	0x162
au16CrMax [9]	512	0x177
au16CrMax [10]	1024	0x189
au16CrMax [11]	2048	0x1A1
au16CrMax [12]	4096	0x1A6
au16CrMax [13]	8192	0x1A6
au16CrMax [14]	16384	0x1A6
au16CrMax [15]	32768	0x1A6

Table 4-2 Mapping between the values of au16CrMin[16] and gains (only for reference)

au16CrMin	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CrMin [0]	1	0x40
au16CrMin [1]	2	0x40
au16CrMin [2]	4	0x40
au16CrMin [3]	8	0x40
au16CrMin [4]	16	0x40
au16CrMin [5]	32	0x40
au16CrMin [6]	64	0x40
au16CrMin [7]	128	0x40
au16CrMin [8]	256	0x40
au16CrMin [9]	512	0x40
au16CrMin [10]	1024	0x40
au16CrMin [11]	2048	0x40

au16CrMin	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CrMin [12]	4096	0x40
au16CrMin [13]	8192	0x40
au16CrMin [14]	16384	0x40
au16CrMin [15]	32768	0x40

Table 4-3 Mapping between the values of au16CbMax[16] and gains (only for reference)

au16CbMax	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CbMax [0]	1	0x120
au16CbMax [1]	2	0x120
au16CbMax [2]	4	0x120
au16CbMax [3]	8	0x120
au16CbMax [4]	16	0x120
au16CbMax [5]	32	0x120
au16CbMax [6]	64	0x120
au16CbMax [7]	128	0x120
au16CbMax [8]	256	0x120
au16CbMax [9]	512	0x120
au16CbMax [10]	1024	0x120
au16CbMax [11]	2048	0x120
au16CbMax [12]	4096	0x120
au16CbMax [13]	8192	0x120
au16CbMax [14]	16384	0x120
au16CbMax [15]	32768	0x120

Table 4-4 Mapping between the values of au16CbMin[16] and gains (only for reference)

au16CbMin	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CbMin [0]	1	0x40
au16CbMin [1]	2	0x40
au16CbMin [2]	4	0x40
au16CbMin [3]	8	0x40

au16CbMin	Again x Dgain x ISPDgain (Multiple)	Configured Value
au16CbMin [4]	16	0x40
au16CbMin [5]	32	0x40
au16CbMin [6]	64	0x40
au16CbMin [7]	128	0x3C
au16CbMin [8]	256	0x38
au16CbMin [9]	512	0x34
au16CbMin [10]	1024	0x30
au16CbMin [11]	2048	0x2C
au16CbMin [12]	4096	0x2A
au16CbMin [13]	8192	0x28
au16CbMin [14]	16384	0x28
au16CbMin [15]	32768	0x28

[See Also]

ISP_AWB_ATTR_S

ISP_AWB_LUM_HISTGRAM_ATTR_S

[Description]

Defines the parameters of the luminance histogram statistics for white balance.

[Syntax]

```
typedef struct hiISP_AWB_LUM_HISTGRAM_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E  enOpType;
    HI_U8   au8HistThresh[6];
    HI_U16  au16HistWt[6];
} ISP_AWB_LUM_HISTGRAM_ATTR_S;
```

[Member]

Member	Description
bEnable	Whether the luminance affects the block weight

Member	Description
enOpType	Mode. In automatic mode, the AWB algorithms implement luminance histogram statistics on the block statistical result and automatically allocate the luminance weight. In manual mode, you need to configure the threshold and weight of the luminance histogram.
au8HistThresh[6]	Configured threshold of the luminance histogram, valid only in manual mode
au16HistWt[6]	Configured weight of the luminance histogram (8-bit decimal precision). This member is valid in both the autoamtic mode and manual mode.

[Note]

- The output AWB statistics for the RGB field do not contain the luminance information.
- **au8HistThresh[0]** is fixed at **0**, and **au8HistThresh[5]** is fixed at **0xFF**. **au8HistThresh[*i* + 1]** must be greater than **au8HistThresh[*i*]**.
- **au16HistWt** is used to configure the weight of the luminance histogram. You can set a high priority to the bright region or the dark region by configuring **au16HistWt**.

[See Also]

ISP_AWB_ATTR_S

ISP_AWB_ALG_TYPE_E

[Description]

Defines the AWB algorithm type.

[Syntax]

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

[Member]

Member	Description
AWB_ALG_LOWCOST	Improved gray world algorithm. It is an AWB algorithm that automatically calculates zone weights based on the statistics.
AWB_ALG_ADVANCE	AWB algorithm that classifies the statistics and re-filters the white blocks
AWB_ALG_BUTT	Invalid

[Note]

- If the AWB algorithm is set to **AWB_ALG_LOWCOST**, the CPU usage is low, and the AWB is stable in various illumination scenarios.
- If the AWB algorithm is set to **AWB_ALG_ADVANCE**, the CPU usage is high. In addition, in the extremely low illumination scenario at a high or low color temperature, slight illuminant color is retained, which improves the AWB algorithm precision when the captured image has a large pure color block (such as a large green block outdoors or a large complexion block indoors).

[See Also]

ISP_AWB_ATTR_S

ISP_AWB_CT_LIMIT_ATTR_S

[Description]

Defines the AWB gain range.

[Syntax]

```
typedef struct hiISP_AWB_CT_LIMIT_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    HI_U16 u16HighRgLimit;
    HI_U16 u16HighBgLimit;
    HI_U16 u16LowRgLimit;
    HI_U16 u16LowBgLimit;
} ISP_AWB_CT_LIMIT_ATTR_S;
```

[Member]

Member	Description
bEnable	WB gain range enable
enOpType	WB gain range configured in automatic or manual mode
u16HighRgLimit	Maximum R gain at a high color temperature in manual mode (8-bit decimal precision). Value range: [0x0, 0xFFF]
u16HighBgLimit	Minimum B gain at a high color temperature in manual mode (8-bit decimal precision). Value range: [0x0, 0xFFF]
u16LowRgLimit	Minimum R gain at a low color temperature in manual mode (8-bit decimal precision). Value range: [0x0, 0xFFF]

Member	Description
u16LowBgLimit	Maximum B gain at a low color temperature in manual mode (8-bit decimal precision). Value range: [0x0, 0xFFF]

[Note]

- ISP_AWB_CT_LIMIT_ATTR_S defines the action to be taken by the AWB module when the ambient color temperature does not fall within the configured color temperature range. The related parameters take effect only when the AWB module detects that the ambient color temperature is greater than the upper limit or less than the lower limit.
- The automatic mode or manual mode can be selected. In automatic mode, the AWB module calculates the AWB gains at the maximum and minimum color temperatures based on the corrected color temperature curve to limit the gains of R and B channels. In manual mode, the AWB module configures the **u16HighRgLimit** and **u16HighBgLimit** parameters to limit the gains of R and B channels at a high color temperature, and configures the **u16LowRgLimit** and **u16LowBgLimit** parameters to limit the gains of R and B channels at a high color temperature.
- You are advised to determine the range of the working color temperature before configuring **u16HighRgLimit**, **u16HighBgLimit**, **u16LowRgLimit**, and **u16LowBgLimit** when you correct the AWB color temperature curve. Rg is the horizontal coordinate of the Planckian curve, and Bg is the vertical coordinate of the Planckian curve.

[See Also]

ISP_AWB_ATTR_S

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 u16Grgain;
    HI_U16 u16Gbgain;
    HI_U16 u16Bgain;
} ISP_MWB_ATTR_S;
```

[Member]

Member	Description
u16Rgain	Red channel gain for manual AWB (8-bit decimal precision) Value range: [0x0, 0xFFF]

Member	Description
u16Grgain	Green channel gain for manual AWB (8-bit decimal precision) Value range: [0x0, 0xFFF]
u16Gbgain	Green channel gain for manual AWB (8-bit decimal precision) Value range: [0x0, 0xFFF]
u16Bgain	Blue channel gain for manual AWB (8-bit decimal precision) Value range: [0x0, 0xFFF]

[Note]

None

[See Also]

ISP_WB_ATTR_S

ISP_WB_ATTR_S

[Description]

Defines AWB attributes.

[Syntax]

```
typedef struct hiISP_WB_ATTR_S
{
  HI_BOOL bByPass;
  ISP_OP_TYPE_E enOpType;
  ISP_MWB_ATTR_S stManual;
  ISP_AWB_ATTR_S stAuto;
} ISP_WB_ATTR_S;
```

[Member]

Member	Description
bByPass	Bypass enable for the white balance module Default value: HI_FALSE
enOpType	Automatic/Manual WB switch
stManual	Manual AWB parameter
stAuto	AWB parameter

[Note]

When **bByPass** is **HI_TRUE**, the configuration of other WB parameters does not take effect and the RGB channel gain coefficient is fixed at 0x100.

[See Also]

- ISP_MWB_ATTR_S
- ISP_AWB_ATTR_S

ISP_AWB_MULTI_LS_TYPE_E

[Description]

Defines the AWB policy in the multi-illuminant scenario.

[Syntax]

```
typedef enum hiISP_AWB_MULTI_LS_TYPE_E
{
    AWB_MULTI_LS_SAT = 0,
    AWB_MULTI_LS_CCM = 1,
    AWB_MULTI_LS_BUTT
} ISP_AWB_MULTI_LS_TYPE_E;
```

[Member]

Member	Description
AWB_MULTI_LS_SAT	Automatic adjustment for saturation in the multi-illuminant scenario
AWB_MULTI_LS_CCM	Automatic adjustment for CCM in the multi-illuminant scenario
AWB_MULTI_LS_BUTT	Invalid member

[Note]

AWB_MULTI_LS_CCM is recommended because the saturation loss of the picture is less when **AWB_MULTI_LS_CCM** is used.

[See Also]

ISP_AWB_ATTR_S

ISP_AWB_ATTR_EX_S

[Description]

Defines extended AWB attributes.

[Syntax]

```
typedef struct hiISP_AWB_ATTR_EX_S
{
    HI_U8 u8Tolerance;
    HI_U8 u8ZoneRadius;
    HI_U16 u16CurveLLimit;
```

```
    HI_U16 u16CurveRLimit;
    HI_BOOL bExtraLightEn;
    ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S stLightInfo[4];
    ISP_AWB_IN_OUT_ATTR_S stInOrOut;
    HI_BOOL bMultiLightSourceEn;
    ISP_AWB_MULTI_LS_TYPE_E enMultiLSType;
    HI_U16  u16MultiLSScaler;
    HI_U16  au16MultiCTBin[8];
    HI_U16  au16MultiCTWt[8];
    HI_BOOL bFineTunEn;
    HI_U8    u8FineTunStrength;
} ISP_AWB_ATTR_EX_S;
```

[Member]

Member	Description
u8Tolerance	Tolerance of AWB adjustment. The AWB performs no operation when the detection error falls within the threshold.
u8ZoneRadius	Zone radius for classifying pixels during AWB statistics. A smaller value indicates higher AWB precision but lower AWB algorithm stability.
u16CurveLLimit	Left border of the AWB color temperature curve
u16CurveRLimit	Right border of the AWB color temperature curve
bExtraLightEn	Separate illuminant (outside the color temperature curve) enable during AWB calculation
stLightInfo[4]	Information about the separate illuminants outside the color temperature curve. A maximum of four illuminants can be added.
stInOrOut	Parameter for determining the AWB outdoor or indoor scenario
bMultiLightSourceEn	Enable for the AWB multi-illuminant detection. This member is used to check whether the current scenario is a multi-illuminant scenario and adjust the saturation or CCM based on the degree of the multiple illuminants.
enMultiLSType	Multi-illuminant adjustment policy select. It supports the adjustment of saturation or CCM.
u16MultiLSScaler	Maximum adjustment range for saturation or CCM in the multi-illuminant scenario. The actual adjustment range is also relevant to the degree of multiple illuminants in the scenario. Value range: [0x0, 0x100]
au16MultiCTBin[8]	Color temperature segment parameter in the multi-illuminant scenario The value of **au16MultiCTBin[4]** must be a monotonically increasing sequence.

Member	Description
au16MultiCTWt[8]	Color temperature weight parameter in the multi-illuminant scenario Value range: [0x0, 0x400]
bFineTunEn	Enable for the AWB special color detection, including the complexion detection
u8FineTunStrength	Strength of detecting a single color, such as complexion or blue. This member is valid only when **bFineTunEn** is enabled. Value range: [0x0, 0xFF]

Figure 4-3 Parameters of color temperature curves

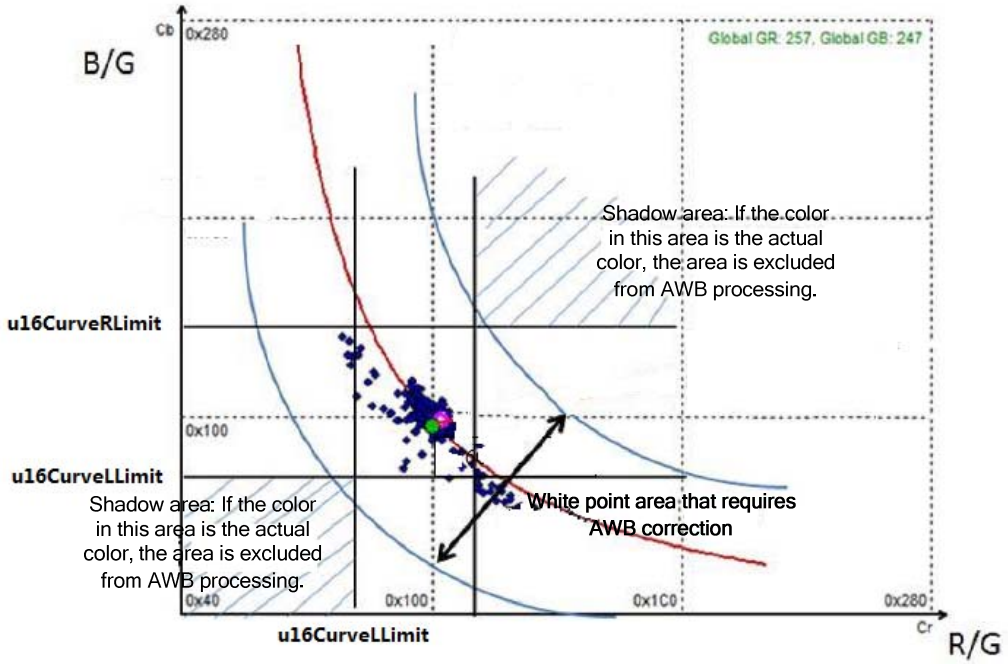
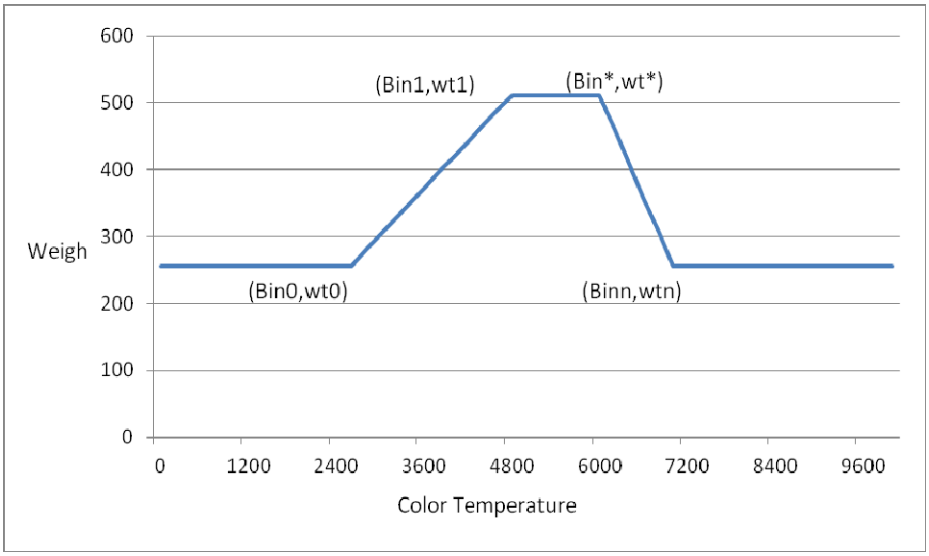


Figure 4-4 Configuration example of the color temperature weight in the multi-illuminant scenario (*n* indicates the number of segmentation points for the color temperature)



[Note]

- **u8Tolerance** is an AWB sensitivity parameter. If **u8Tolerance** is **0**, AWB coefficients are updated in real time, and the CPU usage is high. If **u8Tolerance** is large, the AWB module updates AWB coefficients only when the AWB module detects that the ambient color temperature changes, and therefore the CPU usage is low. When the ambient color temperature is fine-tuned, the image may have a slight color cast. You are advised to set **u8Tolerance** to **0** in the outdoor scenario and **2** in the indoor scenario.
- **u8ZoneRadius** indicates the color difference radius of white blocks in the illuminant environment. A larger radius indicates that the restrictions on white blocks are looser, more white blocks are found, the stability is better, but the AWB precision is lower, and vice versa. It is recommended that the radius in the hybrid illuminant scenario be greater than that in the common scenario.
- The value of **u16CurveLLimit** is less than or equal to **0x100**, and the value of **u16CurveRLimit** is greater than or equal to **0x100**. **u16CurveLLimit** and **u16CurveRLimit** are used to exclude color blocks. As shown in Figure 4-3, **u16CurveLLimit** is used to exclude the left shadow area, and **u16CurveRLimit** is used to exclude the right shadow area.
- The AWB algorithm allows you to add the information about separate illuminants outside the color temperature curve to improve the AWB effect under a special illuminant.
- After **bMultiLightSourceEn** is enabled, the AWB algorithms determine whether the current scenario is a multi-illuminant scenario. In the multi-illuminant scenario, the saturation is automatically decreased or the CCM is adjusted to reduce the color cast effect. The maximum adjustment range is determined by **u16MultiLSScaler**. When the CCM is adjusted, the hue will be changed.
- When **enMultiLSType** is **AWB_MULTI_LS_SAT**, saturation is adjusted to reduce the color cast effect in the multi-illuminant scenario. **u16MultiLSScaler/0x100** indicates the gain coefficient of the final saturation. A larger value indicates higher image saturation. If **u16MultiLSScaler** is **0xC0**, the gain coefficient of the saturation is 0.75 (0xC0/0x100). If **u16MultiLSScaler** is **0x0**, the gain coefficient of the saturation is 0, and the picture may become white and black in the multi-illuminant scenario.

- When **enMultiLSType** is **AWB_MULTI_LS_CCM**, CCM is adjusted to reduce the color cast effect in the multi-illuminant scenario. **u16MultiLSScaler/0x100** indicates the adjustment range of the CCM. A larger value indicates lower image saturation. If **u16MultiLSScaler** is **0xC0**, the maximum adjustment range of the CCM is 0xC0. If **u16MultiLSScaler** is **0x0**, the maximum adjustment range of the CCM is 0, which is equivalent to setting **bMultiLightSourceEn** to **0**.
- In WDR mode, the HiSilicon AWB algorithm automatically disables the function of judging whether the current scenario is a multi-illuminant scenario.
- The **au16MultiCTBin** and **au16MultiCTWt** parameters can be properly configured to improve the color effect in the high color temperature or low color temperature zone in the multi-illuminant scenario. In the non-multi-illuminant scenario, the parameter configuration does not take effect. Figure 4-4 describes the mapping between the color temperature and the weight configuration.
- When **bFineTunEn** is enabled, the AWB automatically detects special colors, such as complexion, to improve the AWB effect in the complexion scenario and the AWB accuracy. However, errors may occur in the blue background+3500ºK illuminant scenario, which results in a slight yellow cast of the picture.
- **u8FineTunStrength** is used to adjust the complexion detection strength. A larger value indicates that the AWB effect is better in the complexion scenario but the side effect is more obvious when misjudgment occurs. The default value 0x80 is recommended.

[See Also]

- ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S
- ISP_AWB_IN_OUT_ATTR_S

ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S

[Description]

Defines the information about a separate illuminant.

[Syntax]

```
typedef struct hiISP_AWB_LIGHTSOURCE_INFO_S
{
    HI_U16 u16WhiteRgain;
    HI_U16 u16WhiteBgain;
    HI_U16 u16ExpQuant;
    HI_BOOL bLightStatus;
    HI_U8   u8LightStatus;
    HI_U8   u8Radius
} ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S;
```

[Member]

Member	Description
u16WhiteRgain	Coordinates of the Cr component center of the region to be added or deleted (8-bit decimal precision) Value range: [0x0, 0xFFF]

Member	Description
u16WhiteBgain	Coordinates of the Cb component center of the region to be added or deleted (8-bit decimal precision) Value range: [0x0, 0xFFF]
u16ExpQuant	External environment luminance. This member is not supported and does not need to be configured.
u8LightStatus	Illuminant status 0: disabled 1: Add the region corresponding to the illuminant. 2: Delete the region corresponding to an interference color.
u8Radius	Radius of the region to be added or deleted

[Note]

- This data structure is used to specify a circular region with the center of [**u16WhiteRgain**, **u16WhiteBgain**] and radius of **u8Radius**. This region is added as the region corresponding to an independent illuminant if it is specified as a region corresponding to the white color under a specific illuminant. This region is deleted if it is specified as the region corresponding to an interference color under a specific illuminant.
- You can add or delete a region by setting **u8LightStatus**. When **u8LightStatus** is set to **1**, this region is added as the region corresponding to an independent illuminant outside the corrected Planckian curve, which optimizes the color effect under this illuminant while exerting no effect on the color effect under other illuminants. When **u8LightStatus** is set to **2**, the region corresponding to an interference color (complexion color, green, or sky blue) under a specific illuminant can be deleted.
- You are advised to delete the region corresponding to an interference color only to optimize the color effect under a specific illuminant. Otherwise, the color effect under other illuminants will be affected. In the entire region determined by the CbCr coordinates, the region corresponding to the complexion color under the 5500ºK illuminant overlaps the regions corresponding to the white color under the 3500ºK illuminant. If the region corresponding to the complexion color is deleted under the 5500ºK illuminant, a color cast occurs under the 3500ºK illuminant.
- The coordinate values (**u16WhiteRgain** and **u16WhiteBgain**) are Rgain and Bgain that are obtained by calibrating the ColorChecker Raw data captured under an illuminant using the **Static WB** option of the calibration tool. If you want to delete a region corresponding to an interference color, you can determine the values of **u16WhiteRgain** and **u16WhiteBgain** by performing static white balance correction on this region.
- If the region is to be added as the region corresponding to an independent illuminant, the recommended value range of **u8Radius** is [0x8, 0x10]. If the region is specified as the region corresponding to an interference color, you are advised to set **u8Radius** to an appropriate value to prevent the circular region with the center of [**u16WhiteRgain**, **u16WhiteBgain**] and radius of **u8Radius** from being overlapped the Planckian curve.
- Currently at most four regions corresponding to four independent illuminants are supported. The four regions must be completely independent of each other. Note that a region cannot be specified as the region corresponding to an independent illuminant and the region corresponding to an interference color at the same time.

[See Also]

ISP_AWB_ATTR_EX_S

ISP_AWB_IN_OUT_ATTR_S

[Description]

Defines the parameters for determining the outdoor or indoor scenario.

[Syntax]

```
typedef struct hiISP_AWB_IN_OUT_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    HI_BOOL bOutdoorStatus;
    HI_U32 u32OutThresh;
    HI_U16 u16LowStart;
    HI_U16 u16LowStop;
    HI_U16 u16HighStart;
    HI_U16 u16HighStop;
    HI_BOOL bGreenEnhanceEn;
    HI_U8   u8OutShiftLimit;
} ISP_AWB_IN_OUT_ATTR_S;
```

[Member]

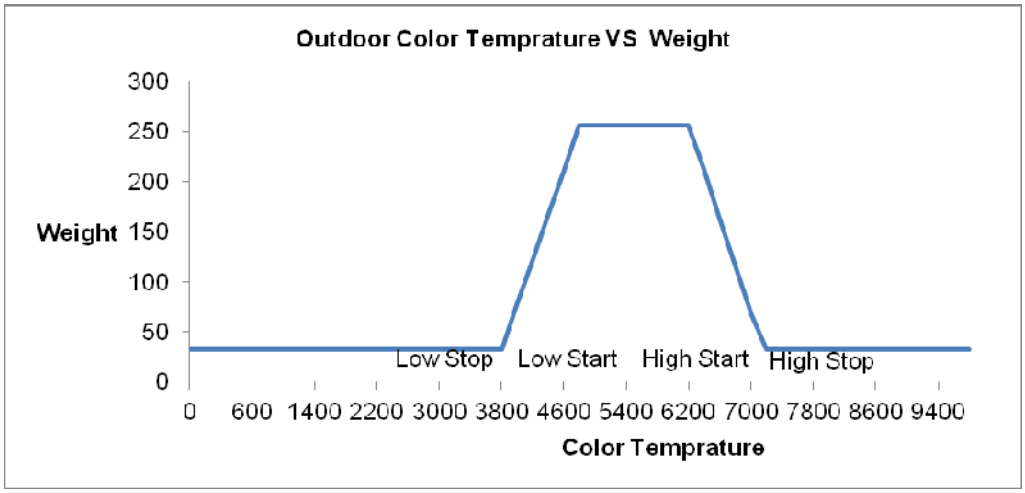
Member	Description
bEnable	Outdoor/Indoor scenario determination enable
enOpType	Operation type of determining the outdoor/indoor scenario (automatic or manual)
bOutdoorStatus	Current outdoor/indoor scenario status determined 1: outdoor 0: indoor
u32OutThresh	Threshold for determining the outdoor/indoor scenario (exposure time in μs). When the value is less than the threshold, the parameter indicates the outdoor scenario.
u16LowStart	Start value of the low temperature zone for decreasing the weight of the pixels in the low color temperature range. 5000ºK is recommended.
u16LowStop	End value of the low temperature zone for decreasing the weight of the pixels in the low color temperature range. 4500ºK is recommended.
u16HighStart	Start value of the high temperature zone for decreasing the weight of the pixels in the low color temperature range. 6500ºK is recommended.

Member	Description
u16HighStop	End value of the high temperature zone for decreasing the weight of the pixels in the low color temperature range. 8000ºK is recommended.
bGreenEnhanceEn	Whether to enhance the green channel in the green plant scenario.
u8OutShiftLimit	White point range limit (Shift) for the AWB algorithm in the outdoor scenario Value range: [0x0, 0xFF]

[Note]

- When the indoor or outdoor scenario is determined automatically, **bOutdoorStatus** is read-only. When the indoor or outdoor scenario is determined manually, **bOutdoorStatus** is write-only, and you can specify whether the current scenario is indoor or outdoor.
- **u32OutThresh** needs to be adjusted based on the sensitometric characteristic of the sensor.
- When you use your own AE algorithm library and the indoor or outdoor scenario is determined automatically, ensure that the current exposure time (in μs) and ISO are transferred to the AWB algorithm library.
- The four parameters **u16LowStart**, **u16LowStop**, **u16HighStart**, and **u16HighStop** used to set the color temperature range and are related to the sensor correction feature. You are advised to fine-tune the sensor. Requirements on the values of the four parameters are as follows: **u16LowStop < u16LowStart < u16HighStart < u16HighStop**. If the outdoor color temperature range [**u16LowStart**, **u16HighStart**] is wide, the color effect may not be good if the captured image has a large pure color block in the outdoor scenario. If the range [**u16LowStart**, **u16HighStart**] is narrow, AWB cannot be completely restored in the high and low color temperature scenarios such as the sunset scenario. As a result, the image has a yellow cast at low color temperature and has a blue cast at high color temperature.
- After AWB correction parameters are adjusted, **u16LowStart**, **u16LowStop**, **u16HighStart**, and **u16HighStop** need to be adjusted again to achieve the optimal effect.
- The shift value affects the performance of the AWB algorithm. The values of **u8ShiftLimit** and **u8OutShiftLimit** have little impact on the AWB result in command scenarios, but have great impact on the AWB result in the special illuminant and large-area single-color scenarios. When the values of the two parameters are small, the single-color effect is good but color cast easily occurs under the special illuminant. When the values of the two parameters are large, the AWB performance is stable but color cast may occur in the large-area green or yellow scenario.

Figure 4-5 Parameters for outdoor color temperature range



[See Also]

ISP_AWB_ATTR_EX_S

ISP_WB_INFO_S

[Description]

Defines white balance, saturation, color correction information.

[Syntax]

```
typedef struct hiISP_WB_INFO_S
{
   HI_U16 u16Rgain;
   HI_U16 u16Grgain;
   HI_U16 u16Gbgain;
   HI_U16 u16Bgain;
   HI_U16 u16Saturation;
   HI_U16 u16ColorTemp;
   HI_U16 au16CCM[9];
  HI_U16 u16LS0CT;
  HI_U16 u16LS1CT;
  HI_U16 u16LS0Area;
  HI_U16 u16LS1Area;
   HI_U8  u8MultiDegree;
} ISP_WB_INFO_S;
```

[Member]

Member	Description
u16Rgain	Current gain of the R channel (8-bit decimal precision)

Member	Description
u16Grgain	Current gain of the Gr channel (8-bit decimal precision)
u16Gbgain	Current gain of the Gb channel (8-bit decimal precision)
u16Bgain	Current gain of the B channel (8-bit decimal precision)
u16Saturation	Current saturation Value range: [0, 255]
u16ColorTemp	Current color temperature
au16CCM[9]	Current CCM value (8-bit decimal precision) Bit 15 is a sign bit. The value 0 indicates positive, and the value 1 indicates negative. For example, the value 0x8010 indicates −16.
u16LS0CT	Color temperature of the primary illuminant in the multi-illuminant scenario
u16LS1CT	Color temperature of the secondary illuminant in the multi-illuminant scenario
u16LS0Area	Size of the primary illuminant in the multi-illuminant scenario Value range: [0x0, 0xFF]
u16LS1Area	Size of the secondary illuminant in the multi-illuminant scenario Value range: [0x0, 0xFF]
u8MultiDegree	Probability that the current scenario is a multi-illuminant scenario

[Note]

- When **u8MultiDegree** is a non-zero value, the current scenario is a multi-illuminant scenario. A larger value indicates a higher probability that the current scenario is a multi-illuminant scenario.
- The query results of **u16LS0CT**, **u16LS1CT**, **u16LS0Area**, and **u16LS1Area** take effect only when **u8MultiDegree** is a non-zero value.
- The multi-illuminant detection function is enabled only in the indoor scenario. When the AWB determines that the current scenario is an outdoor scenario, the multi-illuminant detection function is automatically disabled to avoid saturation decrease.
- The luminance, color temperature difference, and other conditions of the primary and secondary illuminants affect the calculation of the probability of the multi-illuminant scenario. If the primary and secondary illuminants are detected in the scenario, a larger color temperature difference and a smaller luminance difference between the primary and secondary illuminants indicates higher probability of the multi-illuminant scenario.
- The sum of **u16LS0Area** and **u16LS1Area** may be greater than the number of white balance zones. The reason is that some zones may be affected by the primary and secondary illuminants and the algorithm cannot clearly identify the overlapped zone. Therefore, the overlapped zone is overlapped to the primary and secondary illuminants separately.

[See Also]

None

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 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 saturation, hue, and CCM MPIs are based on the HiSilicon AWB algorithm library. These MPIs can be ignored and are unavailable if you use your own AWB algorithm library.

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_SetCCMAttr: Sets the CCM.
- HI_MPI_ISP_GetCCMAttr: Obtains the CCM.

HI_MPI_ISP_SetSaturationAttr

[Description]

Sets the color saturation attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetSaturationAttr(ISP_DEV IspDev, const
ISP_SATURATION_ATTR_S *pstSatAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstSatAttr	Color saturation attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.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_DEV IspDev, ISP_SATURATION_ATTR_S
*pstSatAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstSatAttr	Color saturation attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetSaturationAttr

HI_MPI_ISP_SetCCMAttr

[Description]

Sets the CCM.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetCCMAttr(ISP_DEV IspDev, const ISP_COLORMATRIX_ATTR_S
*pstCCMAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCCMAttr	CCM attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.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.
- In manual mode, the CCM that takes effect in the ISP system is the manually configured CCM, which facilitates the fine adjustment of the CCM. Therefore, it is recommended that the CCM matrices in high, intermediate, and low illuminants be corrected in manual

mode. After three groups of corrected CCM matrices are obtained and written by using the MPI, the combination result of CCMs in multiple illuminants can be verified.

- The saturation adjustment is valid in manual CCM mode. Before fine-tuning the CCM under a specific illuminant, check whether the current saturation value is equal to the expected saturation value.

[Example]

None

[See Also]

HI_MPI_ISP_GetCCMAttr

HI_MPI_ISP_GetCCMAttr

[Description]

Obtains the CCM.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetCCMAttr(ISP_DEV IspDev, ISP_COLORMATRIX_ATTR_S
*pstCCMAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCCMAttr	CCM attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Description
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetCCMAttr

5.5 Data Structures

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

- ISP_SATURATION_ATTR_S: Defines the ISP saturation attribute.
- ISP_SATURATION_MANUAL_S: Defines the manual saturation attribute.
- ISP_SATURATION_AUTO_S: Defines the automatic saturation attribute.
- ISP_COLORMATRIX_ATTR_S: Defines the ISP color matrix attribute.
- ISP_COLORMATRIX_AUTO_S: Defines the automatic CCM attribute.
- ISP_COLORMATRIX_MANUAL_S: Defines the manual CCM attribute.

ISP_SATURATION_ATTR_S

[Description]

Defines the ISP saturation attribute.

[Syntax]

```
typedef struct hiISP_SATURATION_ATTR_S
{
    ISP_OP_TYPE_E enOpType;
    ISP_SATURATION_MANUAL_S stManual;
    ISP_SATURATION_AUTO_S stAuto;
}ISP_SATURATION_ATTR_S;
```

[Member]

Member	Description
enOpType	Saturation adjustment type (manual or automatic)
stManual	Manual saturation parameter
stAuto	Automatic saturation parameter

[Note]

Saturation can be automatically or manually adjusted.

- If **enOpType** is **OP_TYPE_AUTO**, saturation is automatically adjusted based on the system gain. For details about the mapping between the saturation and gains, see the description of **au8Sat[16]**.
- If **enOpType** is **OP_TYPE_MANUAL**, saturation is manually adjusted, and **au8Sat[16]** is invalid.

[See Also]

- ISP_SATURATION_AUTO_S
- ISP_COLORMATRIX_MANUAL_S

ISP_SATURATION_MANUAL_S

[Description]

Defines the manual saturation attribute.

[Syntax]

```
typedef struct hiISP_SATURATION_MANUAL_S
{
    HI_U8 u8Saturation;
} ISP_SATURATION_MANUAL_S;
```

[Member]

Member	Description
u8Saturation	Manual saturation

[Note]

None

[See Also]

ISP_SATURATION_AUTO_S

ISP_SATURATION_AUTO_S

[Description]

Defines the automatic saturation attribute.

[Syntax]

```
typedef struct hiISP_SATURATION_AUTO_S
{
    HI_U8 au8Sat[ISP_AUTO_STENGTH_NUM];
} ISP_SATURATION_AUTO_S;
```

[Member]

Member	Description
au8Sat[ISP_AUTO_STENGTH_NUM]	Automatic saturation

Table 5-1 Mapping between the values of au8Sat[16] and gains (using the MN34220 sensor as an example)

au8Sat	Again x Dgain x ISPDgain (Multiple)	Recommended Configured Value
au8Sat [0]	1	0x74
au8Sat [1]	2	0x74
au8Sat [2]	4	0x70
au8Sat [3]	8	0x6b
au8Sat [4]	16	0x64
au8Sat [5]	32	0x5b
au8Sat [6]	64	0x52
au8Sat [7]	128	0x44
au8Sat [8]	256	0x3b
au8Sat [9]	512	0x38
au8Sat [10]	1024	0x38
au8Sat [11]	2048	0x38
au8Sat [12]	4096	0x38
au8Sat [13]	8192	0x38
au8Sat [14]	16384	0x38
au8Sat [15]	32768	0x38

[Note]

None

[See Also]

ISP_SATURATION_MANUAL_S

ISP_COLORMATRIX_ATTR_S

[Description]

Defines the ISP color matrix attribute.

[Syntax]

```
typedef struct hiISP_COLORMATRIX_ATTR_S
{
  ISP_OP_TYPE_E enOpType;
  ISP_COLORMATRIX_MANUAL_S stManual;
  ISP_COLORMATRIX_AUTO_S stAuto;
} ISP_COLORMATRIX_ATTR_S;
```

[Member]

Member	Description
enOpType	CCM type (manual or automatic)
stManual	Manual CCM
stAuto	Automatic CCM

[Note]

None

[See Also]

- ISP_COLORMATRIX_AUTO_S
- ISP_COLORMATRIX_MANUAL_S

ISP_COLORMATRIX_AUTO_S

[Description]

Defines the automatic CCM attribute.

[Syntax]

```
typedef struct hiISP_COLORMATRIX_AUTO_S
{
  HI_BOOL bISOActEn;
  HI_BOOL bTempActEn;
  HI_U16 u16HighColorTemp;
  HI_U16 au16HighCCM[9];
  HI_U16 u16MidColorTemp;
  HI_U16 au16MidCCM[9];
  HI_U16 u16LowColorTemp;
  HI_U16 au16LowCCM[9];
} ISP_COLORMATRIX_AUTO_S;
```

[Member]

Member	Description
bISOActEn	Whether to enable the CCM bypass function under low illuminations

Member	Description
bTempActEn	Whether to enable the CCM bypass function at the high and low color temperatures
u16HighColorTemp	High color temperature Value range: [2800, 10000]
au16HighCCM[9]	CCM at a high color temperature (8-bit decimal precision) Bit 15 is a sign bit. The value 0 indicates positive, and the value 1 indicates negative. For example, 0x8010 indicates −16. Value range: [0x0, 0xFFFF]
u16MidColorTemp	Medium color temperature Value range: [2400, u16HighColorTemp − 400]
au16MidCCM[9]	CCM at a medium color temperature (8-bit decimal precision) Bit 15 is a sign bit. The value 0 indicates positive, and the value 1 indicates negative. For example, the value 0x8010 indicates −16. Value range: [0x0, 0xFFFF]
u16LowColorTemp	Low color temperature Value range: [2000, u16MidColorTemp − 400]
au16LowCCM[9]	CCM at a low color temperature (8-bit decimal precision) Bit 15 is a sign bit. The value 0 indicates positive, and the value 1 indicates negative. For example, 0x8010 indicates −16. 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** ≥ 400
  - **u16MidColorTemp** – **u16LowColorTemp** ≥ 400
- When the illumination is extremely low, the object colors are more natural if the CCM is bypassed. After the CCM bypass function is disabled, the saturation array still takes effect when the illumination is extremely low.
- In the high color temperature (higher than 10000ºK) and low color temperature (lower than 2300ºK) scenarios, the AWB may not be completed restored. If a high-saturation CCM is used in this case, the AWB color cast will be more obvious. For example, in the sodium lamp scenario, the effect of the blue color is not good enough in the white area after AWB correction. If a high-saturation CCM is used, the B component of the white area may be 0 and the entire picture has a yellow cast. After **bTempActEn** is enabled, the CCM is automatically bypassed at the high and low color temperatures, and the color effect in these scenarios is optimized.

[See Also]

ISP_COLORMATRIX_MANUAL_S

ISP_COLORMATRIX_MANUAL_S

[Description]

Defines the manual CCM attribute.

[Syntax]

```
typedef struct hiISP_COLORMATRIX_MANUAL_S
{
  HI_BOOL bSatEn;
  HI_U16 au16CCM[9];
} ISP_COLORMATRIX_MANUAL_S;
```

[Member]

Member	Description
bSatEn	Whether the saturation takes effect in manual CCM mode
au16CCM[9]	Manual CCM (8-bit decimal precision) Bit 15 is a sign bit. The value 0 indicates positive, and the value 1 indicates negative. For example, the value 0x8010 indicates −16. Value range: [0x0, 0xFFFF]

[Note]

You can choose whether to validate the saturation in manual CCM mode by setting **bSatEn**. If **bSatEn** is set to **1**, the CCM that takes effect is the manual CCM multiplied by the saturation CCM. You can determine the saturation arrays under various illuminations based on this formula. If **bSatEn** is set to **0**, the CCM that takes effect is the manual CCM.

[See Also]

ISP_COLORMATRIX_AUTO_S

6 IMP

The IMP is a module that affects the picture effect. The corresponding APIs must be called after HI_MPI_ISP_Init is called.

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_S32 HI_MPI_ISP_SetSharpenAttr(ISP_DEV IspDev const ISP_SHARPEN_ATTR_S
*pstSharpenAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstSharpenAttr	Attribute for edge sharpen	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_32 HI_MPI_ISP_GetSharpenAttr(ISP_DEV_IspDev, ISP_SHARPEN_ATTR_S
*pstSharpenAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstSharpenAttr	Attribute for edge sharpen	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetSharpenAttr

6.1.3 Data Structures

The following are the data structures related to sharpening:

- ISP_SHARPEN_MANUAL_ATTR_S: Defines the manual ISP sharpen attribute.
- ISP_SHARPEN_AUTO_ATTR_S: Defines the automatic ISP sharpen attribute.
- ISP_RGBSHARPEN_ATTR_S: Defines the ISP RGB sharpen attribute.
- ISP_SHARPEN_ATTR_S: Defines the ISP sharpen attribute.

ISP_SHARPEN_MANUAL_ATTR_S

[Description]

Defines the manual attribute for ISP sharpen.

[Syntax]

```
typedef struct hiISP_SHARPEN_MANUAL_ATTR_S
{
    HI_U8 u8SharpenD;
    HI_U8 u8SharpenUd;
```

```
    HI_U8 u8SharpenRGB;
} ISP_SHARPEN_MANUAL_ATTR_S;
```

[Member]

Member	Description
u8SharpenD	Sharpness of the large edge Value range: [0, 0xFF]
u8SharpenUd	Sharpness of the small texture Value range: [0, 0xFF]
u8SharpenRGB	Overall image sharpness Value range: [0, 0xFF]

ISP_SHARPEN_AUTO_ATTR_S

[Description]

Defines the automatic ISP sharpen attribute.

[Syntax]

```
typedef struct hiISP_SHARPEN_AUTO_ATTR_S
{
    HI_U8 au8SharpenD[ISP_AUTO_STENGTH_NUM];
    HI_U8 au8SharpenUd[ISP_AUTO_STENGTH_NUM];
    HI_U8 au8SharpenRGB[ISP_AUTO_STENGTH_NUM];
} ISP_SHARPEN_AUTO_ATTR_S;
```

[Member]

Member	Description
u8SharpenD[ISP_AUTO_STENGTH_NUM]	Large edge sharpness. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-1.
u8SharpenUd[ISP_AUTO_STENGTH_NUM]	Small texture sharpness. It is recommended **u8SharpenUd** be less than **u8SharpenD** under the same gain. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-2.

Member	Description
u8SharpenRGB[ISP_AUTO_STENGTH_NUM]	Overall image sharpness. You are advised not to set a large value because white edges are obvious when the value is too large. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-3.

Table 6-1 Mapping between the configured values of u8SharpenD[ISP_AUTO_STENGTH_NUM] and gains

u8SharpenD	Again x Dgain x ISPDgain (Multiple)
u8SharpenD [0]	1
u8SharpenD [1]	2
u8SharpenD [2]	4
u8SharpenD [3]	8
u8SharpenD [4]	16
u8SharpenD [5]	32
u8SharpenD [6]	64
u8SharpenD [7]	128
u8SharpenD [8]	256
u8SharpenD [9]	512
u8SharpenD [10]	1024
u8SharpenD [11]	2048
u8SharpenD [12]	4096
u8SharpenD [13]	8192
u8SharpenD [14]	16384
u8SharpenD [15]	32768

Table 6-2 Mapping between the configured values of u8SharpenUd[ISP_AUTO_STENGTH_NUM] and gains

u8SharpenUd	Again x Dgain x ISPDgain (Multiple)
u8SharpenUd [0]	1

u8SharpenUd	Again x Dgain x ISPDgain (Multiple)
u8SharpenUd [1]	2
u8SharpenUd [2]	4
u8SharpenUd [3]	8
u8SharpenUd [4]	16
u8SharpenUd [5]	32
u8SharpenUd [6]	64
u8SharpenUd [7]	128
u8SharpenUd [8]	256
u8SharpenUd [9]	512
u8SharpenUd [10]	1024
u8SharpenUd [11]	2048
u8SharpenUd [12]	4096
u8SharpenUd [13]	8192
u8SharpenUd [14]	16384
u8SharpenUd [15]	32768

Table 6-3 Mapping between the configured values of u8SharpenRGB[ISP_AUTO_STENGTH_NUM] and gains

u8SharpenRGB	Again x Dgain x ISPDgain (Multiple)
u8SharpenRGB [0]	1
u8SharpenRGB [1]	2
u8SharpenRGB [2]	4
u8SharpenRGB [3]	8
u8SharpenRGB [4]	16
u8SharpenRGB [5]	32
u8SharpenRGB [6]	64
u8SharpenRGB [7]	128
u8SharpenRGB [8]	256
u8SharpenRGB [9]	512
u8SharpenRGB [10]	1024
u8SharpenRGB [11]	2048

u8SharpenRGB	Again x Dgain x ISPDgain (Multiple)
u8SharpenRGB [12]	4096
u8SharpenRGB [13]	8192
u8SharpenRGB [14]	16384
u8SharpenRGB [15]	32768

ISP_RGBSHARPEN_ATTR_S

[Description]

Defines the ISP RGB sharpen attribute.

[Syntax]

```
typedef struct hiISP_RGBSHARPEN_ATTR_S
{
    HI_U8   u8LutCore;
    HI_U8   u8LutStrength;
    HI_U8   u8LutMagnitude;
} ISP_RGBSHARPEN_ATTR_S;
```

[Member]

Member	Description
u8LutCore	Parameter used for generating the sharpen RGB curve. This parameter mainly affects the rising slope of the sharpen RGB curve, and it also affects the falling slope of the sharpen RGB curve. See Figure 2-5. Value range: [0, 255]
u8LutStrength	Parameter used for generating the sharpen RGB curve. This parameter affects the strength of the sharpen RGB curve. See Figure 2-6. Value range: [0, 127]
u8LutMagnitude	Parameter used for generating the sharpen RGB curve. This parameter mainly affects the falling slope of the sharpen RGB curve, and it also affects the rising slope of the sharpen RGB curve. See Figure 2-7. Value range: [0, 31]

[Note]

None

[See Also]

None

ISP_SHARPEN_ATTR_S

[Description]

Defines the ISP sharpen attribute.

[Syntax]

```
typedef struct hiISP_SHARPEN_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_SHARPEN_MANUAL_ATTR_S stManual;
    ISP_SHARPEN_AUTO_ATTR_S   stAuto;
    ISP_RGBSHARPEN_ATTR_S     stRGBSharpenAttr;
} ISP_SHARPEN_ATTR_S;
```

[Member]

Member	Description
bEnable	Sharpen enhancement enable HI_FALSE: disabled HI_TRUE (default): enabled
enOpType	Sharpen operation type OP_TYPE_AUTO (default): automatic OP_TYPE_MANUAL: manual
stManual	Manual sharpen parameter. For details, see ISP_SHARPEN_MANUAL_ATTR_S.
stAuto	Automatic sharpen parameter. For details, see ISP_SHARPEN_AUTO_ATTR_S.
stRGBSharpenAttr	ISP RGB sharpen attribute

[Note]

Automatic sharpen and manual sharpen are supported.

- When **bEnable** is **HI_TRUE** and **enOpType** is **OP_TYPE_AUTO**, automatic sharpen is used. For details about the mapping between the sharpen strength and the sensor gain, see descriptions of **u8SharpenD[ISP_AUTO_STENGTH_NUM]**, **u8SharpenUd[ISP_AUTO_STENGTH_NUM]**, and **au8SharpenRGB[ISP_AUTO_STENGTH_NUM]**.
- When **bEnable** is **HI_TRUE** and **enOpType** is **OP_TYPE_MANUAL**, manual sharpen is used.

[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 same gamma tables. The spacing between gamma table nodes is the same. The image pixel values between nodes 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_SetGammaAttr

[Description]

Sets the gamma attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetGammaAttr(ISP_DEV IspDev, const ISP_GAMMA_ATTR_S*
pstGammaAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstGammaAttr	Gamma attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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]

After HI_MPI_ISP_SetGammaAttr is configured, the setting takes effect two frames later.

[Example]

None

[See Also]

HI_MPI_ISP_GetGammaAttr

HI_MPI_ISP_GetGammaAttr

[Description]

Obtains the gamma attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetGammaAttr(ISP_DEV IspDev, ISP_GAMMA_ATTR_S*
pstGammaAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstGammaAttr	Gamma attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

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_TYPE_E: Defines the type of the ISP gamma curve.

ISP_GAMMA_ATTR_S

[Description]

Defines the attribute for ISP gamma correction.

[Syntax]

```
typedef struct hiISP_GAMMA_ATTR_S
{
    HI_BOOL bEnable;
    ISP_GAMMA_CURVE_TYPE_E enCurveType;
    HI_U16 u16Table[GAMMA_NODE_NUM];
} ISP_GAMMA_ATTR_S;
```

[Member]

Member	Description
bEnable	Gamma correction enable HI_FALSE: disabled HI_TRUE: enabled Default value: HI_TRUE
enCurveType	Gamma curve select Default value: ISP_GAMMA_CURVE_DEFAULT
u16Table[GAMMA_NODE_NUM]	Gamma table Value range: [0, 0xFFF]

[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_TYPE_E

[Description]

Defines the type of the ISP gamma curve.

[Syntax]

```
typedef enum hiISP_GAMMA_CURVE_TYPE_E
{
   ISP_GAMMA_CURVE_DEFAULT = 0x0,
   ISP_GAMMA_CURVE_SRGB,
   ISP_GAMMA_CURVE_USER_DEFINE,
   ISP_GAMMA_CURVE_BUTT
} ISP_GAMMA_CURVE_TYPE_E;
```

[Member]

Member	Description
ISP_GAMMA_CURVE_DEFAULT	Default gamma curve
ISP_GAMMA_CURVE_SRGB	sRGB 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. The gamma curve in WDR mode is different from that in linear mode and can be generated by the HiSilicon PQ Tools.

[See Also]

ISP_GAMMA_ATTR_S

6.3 DRC

6.3.1 Function Description

The 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(ISP_DEV IspDev, const ISP_DRC_ATTR_S*pstDRC);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDRC	DRC attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_GetDRCAttr

HI_MPI_ISP_GetDRCAttr

[Description]

Obtains DRC attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDRCAttr(ISP_DEV IspDev, ISP_DRC_ATTR_S*pstDRC);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDRC	DRC attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

[Description]

Defines DRC attributes.

[Syntax]

```
typedef struct hiISP_DRC_ATTR_S
{
   HI_BOOL bEnable;
   HI_U32  u32SlopeMax;
   HI_U32  u32SlopeMin;
   HI_U32  u32WhiteLevel;
   HI_U32  u32BlackLevel;
   HI_U32  u32VarianceSpace;
   HI_U32  u32VarianceIntensity;
   HI_U32  u32Asymmetry;
   HI_U32  u32BrightEnhance;
   ISP_OP_TYPE_E enOpType;
   ISP_DRC_MANUAL_ATTR_S stManual;
   ISP_DRC_AUTO_ATTR_S   stAuto;
} ISP_DERC_ATTR_S;
```

[Member]

Member	Sub Member	Description
bEnable	None	DRC enable HI_FALSE: disabled HI_TRUE: enabled In linear mode, the default value is HI_FALSE. In WDR mode, the default value is HI_TRUE and it must be set to **HI_TRUE**.
u32SlopeMax	None	DRC curve control parameter used to limit the maximum slope (gain) for the dark regions on the DRC curve. A larger value indicates a brighter image. **u32SlopeMax** can be less than **u32SlopeMin**. Value range: [0, 0xFF] The default value is defined by the **stDrc.u32SlopeMax** member of cmos_isp_default_t in the **cmos.c** file. Recommended value range: [0x20, 0x60]

Member	Sub Member	Description
u32SlopeMin	None	DRC curve control parameter used to limit the minimum slope (gain) for the bright regions on the DRC curve. A larger value indicates a darker image. Value range: [0, 0xFF] The default value is specified by the **stDrc.u32SlopeMin** member of cmos_isp_default_t in the **cmos.c** file.
u32WhiteLevel	None	Upper limit of the DRC algorithm. Pixels whose values are greater than **u32Whitelevel** are not involved in the DRC calculation. The default value is defined by the **stDrc.u32WhiteLevel** member of cmos_isp_default_t in the **cmos.c** file. Value range: [0, 0xFFF]
u32BlackLevel	None	Lower limit of the DRC algorithm. Pixels whose values are less than **u32Blacklevel** are not involved in the DRC calculation. Value range: [0, 0xFFF] The default value is specified by the **stDrc.u32BlackLevel** member of cmos_isp_default_t in the **cmos.c** file.
u32VarianceSpace	None	Spatial domain sensitivity of the DRC algorithm. A larger value indicates that more surrounding pixels are referenced when the DRC curve is generated. Value range: [0x0, 0xF] The default value is defined by the **stDrc.u32VarianceSpace** member of cmos_isp_default_t in the **cmos.c** file.
u32VarianceIntensity	None	Luminance domain sensitivity of the DRC algorithm. A larger value indicates that the difference between the DRC curve of each pixel and that of its surrounding pixels is smaller. Value range: [0x0, 0xF] The default value is defined by the **stDrc.u32VarianceIntensity** member of cmos_isp_default_t in the **cmos.c** file.
u32Asymmetry	None	DRC curve control parameter used to generate the DRC curve. The DRC curve is used for non-linear mapping of image data. The default value is recommended. See Figure 6-1. Value range: [0, 0xFF] Default value: 0x14

Member	Sub Member	Description
u32BrightEnhance	None	DRC curve control parameter used to generate the DRC curve. The DRC curve is used for non-linear mapping of image data. The default value is recommended. See Figure 6-2. Value range: [0, 0xFF] Default value: 0xA0
enOpType	None	OP_TYPE_AUTO (default): automatic DRC strength mode OP_TYPE_MANUAL: manual DRC strength mode
stManual	u32Strength	DRC strength in manual mode. In this mode, the actual strength is the same as the configured value. Value range: [0, 0xFF] Default value: 0x80
stAuto	u32Strength	DRC strength in automatic mode. In linear mode, the actual strength is related to the configured value, histograms, and ISO. In multi-frame combination WDR mode and sensor WDR mode, the actual strength is related to the configured value, the ratio of the long exposure time to the short exposure time and the ISO. A larger value indicates a greater actual strength. In long frame mode, the DRC strength gradually converges to this value. Value range: [0, 0xFF] Default value: 0x80

Figure 6-1 Impact on the DRC tone curve exerted by the Asymmetry when BrightEnhance is 255

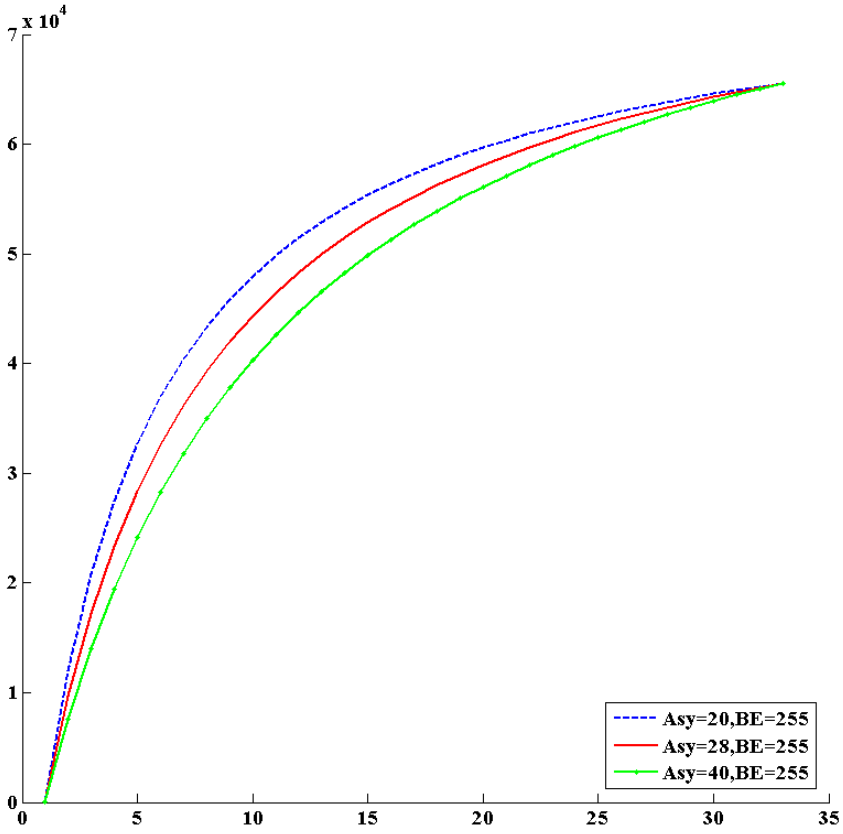
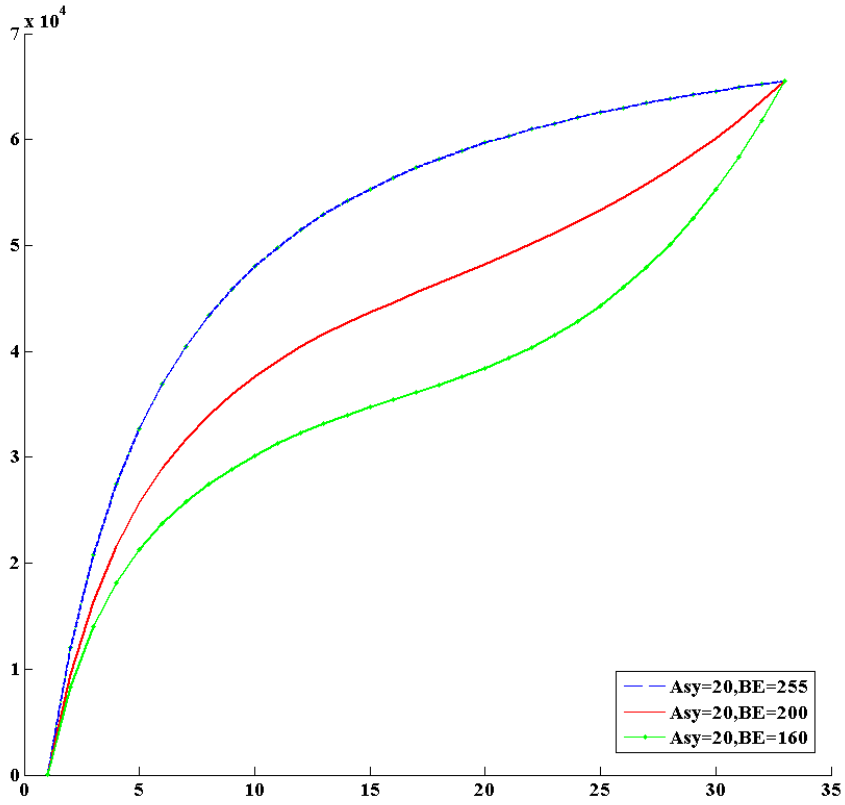


Figure 6-2 Impact on the DRC tone curve exerted by the BrightEnhance when Asymmetry is 20



[Note]

- After the DRC is started, the larger the **u32Strength** value, the lighter the dark area, and the larger the noise is.
- The DRC function must be enabled in WDR mode.
- The variables **u32Asymmetry** and **u32BrightEnhance** cannot be dynamically modified. Otherwise, bad frames occur. After the variables **enOpType**, **stManual.u32Strength**, and **stAuto.u32Strength** are configured, the setting takes effect two frames later. For other variables, the setting takes effect in the next frame.

[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

6.4.2.1 Hi3516A

Radial correction is adopted 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 and 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_SetShadingAttr

[Description]

Sets the attribute for shading correction.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetShadingAttr(ISP_DEV IspDev, const ISP_SHADING_ATTR_S
*pstShadingAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstShadingAttr	Attribute for shading correction	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_GetShadingAttr

HI_MPI_ISP_GetShadingAttr

[Description]

Obtains the attribute for shading correction.

[Syntax]

```
HI_MPI_ISP_GetShadingAttr(ISP_DEV IspDev, ISP_SHADING_ATTR_S
*pstShadingAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstShadingAttr	Attribute for shading correction	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

6.4.4 Data Structures

The following are data structures related to lens shading correction for the Hi3516A:

- ISP_SHADING_ATTR_S: Defines the ISP lens shading correction attribute.
- ISP_SHADING_RADIAL_S: Defines the coordinates, center distances, and correction table attributes for the R, G, and B components.
- POINT_S: Defines the horizontal and vertical coordinates for the R, G, and B components.

ISP_SHADING_ATTR_S

[Description]

Defines the ISP lens shading correction attribute.

[Syntax]

```
typedef struct hiISP_SHADING_ATTR_S
{
   HI_BOOL bEnable;
   HI_U16  u16TableNodeNum;
   ISP_SHADING_RADIAL_S astRadialShading[3];
} ISP_SHADING_ATTR_S;
```

[Member]

Member	Description
bEnable	Lens shading correction enable HI_FALSE (default): disabled HI_TRUE: enabled

Member	Description
u16TableNodeNum	Number of used nodes in each component correction table Value range: [0x0, 0x81] If shading is configured in **cmos.c**, the default value depends on **cmos.c**. If shading is not configured in **cmos.c**, the default value is 0x81.
astRadialShading	Coordinates, center distances, and correction table attributes for the R, G, and B components

[Note]

During lens shading correction, the shading tables of the R, G, and B components are separately called.

[See Also]

ISP_SHADING_RADIAL_S

ISP_SHADING_RADIAL_S

[Description]

Defines the coordinates, center distances, and correction table attributes for the R, G, and B components.

[Syntax]

```
typedef struct hiISP_SHADING_RADIAL_S
{
    HI_U16      u16OffCenter;
    POINT_S     stCenter;
    HI_U32      u32Table[SHADING_NODE_NUM_MAX];
} ISP_SHADING_RADIAL_S;
```

[Member]

Member	Description
u16OffCenter;	Scaling coefficient of the distance from the center of each component to the farthest angle. A longer distance indicates a smaller value. **u16OffCenter** is calculated as follows: u16OffCenter = 2^31/r^2. **r** indicates the distance from the center to the farthest angle and its unit is pixel. Value range: [0x0, 0xFFFF]
stCenter	Coordinates of the center of each component
u32Table	Correction table of each component Value range: [0x0, 0xFFFF]

[Note]

None

[See Also]

POINT_S

POINT_S

[Description]

Defines the horizontal and vertical coordinates for the R, G, and B components.

[Syntax]

```
typedef struct hiPOINT_S
{
    HI_S32 s32X;
    HI_S32 s32Y;
}POINT_S;
```

[Member]

Member	Description
s32X	Horizontal coordinate of the center of each component Value range: [0x0, 0xFFFF]
s32Y	Vertical coordinate of the center of each component Value range: [0x0, 0xFFFF]

[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. In this mode, components R, Gr, Gb, and B are processed separately. If the defect pixels in Figure 6-3 appear in the Bayer picture, the defect pixels can be corrected. The defect pixels in Figure 6-4 cannot be completely corrected. The black blocks indicate the defect pixels in a color channel (R, Gr, Gb, or B). The maximum number of defect pixels (including bright defect pixels and dark defect pixels) that can be detected and corrected is 4095.

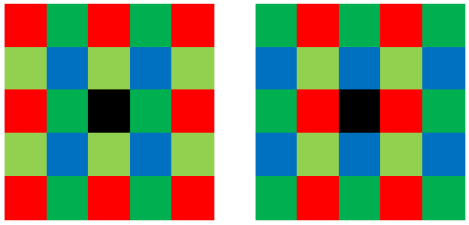
  To detect bright defect pixels, enable the iris. The defect pixel detection program is started to obtain coordinates for defect pixels. The detected defect pixels are corrected by using median filtering for adjacent pixels.

  To detect dark defect pixels, enable the iris, select the flat background, use a fixed illuminant such as a luminance box, and set the average luminance of the entire image to 50% of the maximum luminance or set the B channel luminance in the Bayer format to 20% of the maximum luminance. The defect pixel detection program is started to obtain coordinates for defect pixels. The detected defect pixels are corrected by using median filtering for adjacent pixels.
- Dynamic defect pixel detection and correction

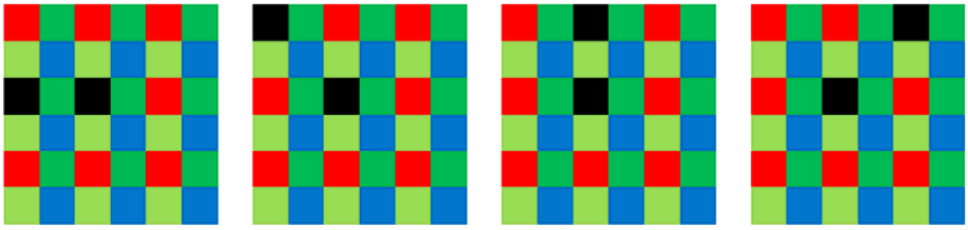
  In this mode, the defect pixel correction module dynamically detects and corrects defect pixels. Only one defect pixel can be processed at a time and there is no limit on the number of defect pixels to be corrected. In normal illuminant scenarios, you can choose whether to use the dynamic defect pixel correction function or not. In low illuminant scenarios, you are advised to use the dynamic defect pixel correction function because weak defect pixels are amplified to dazzling white points and obvious black points.

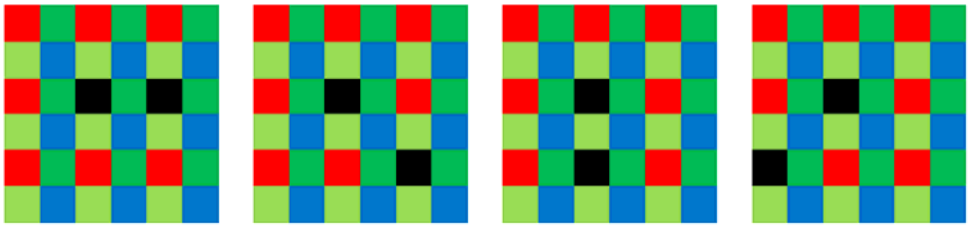
Figure 6-3 Correctable static defect pixels

1. Single defect pixel



2. Two defect pixels (single channel)





3. Defect pixel cluster of multiple color channels (at most two defect pixels for a single channel)

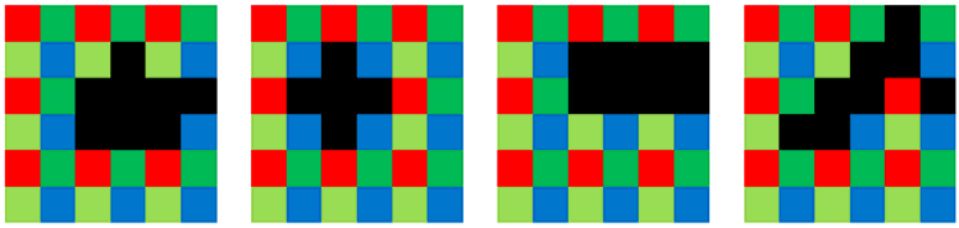
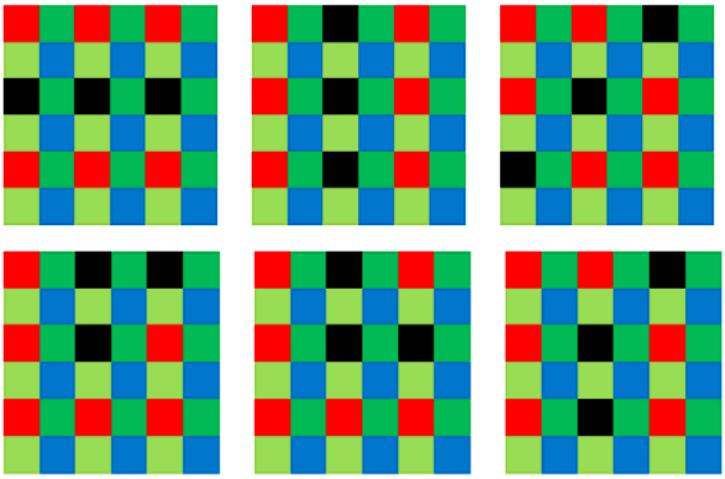
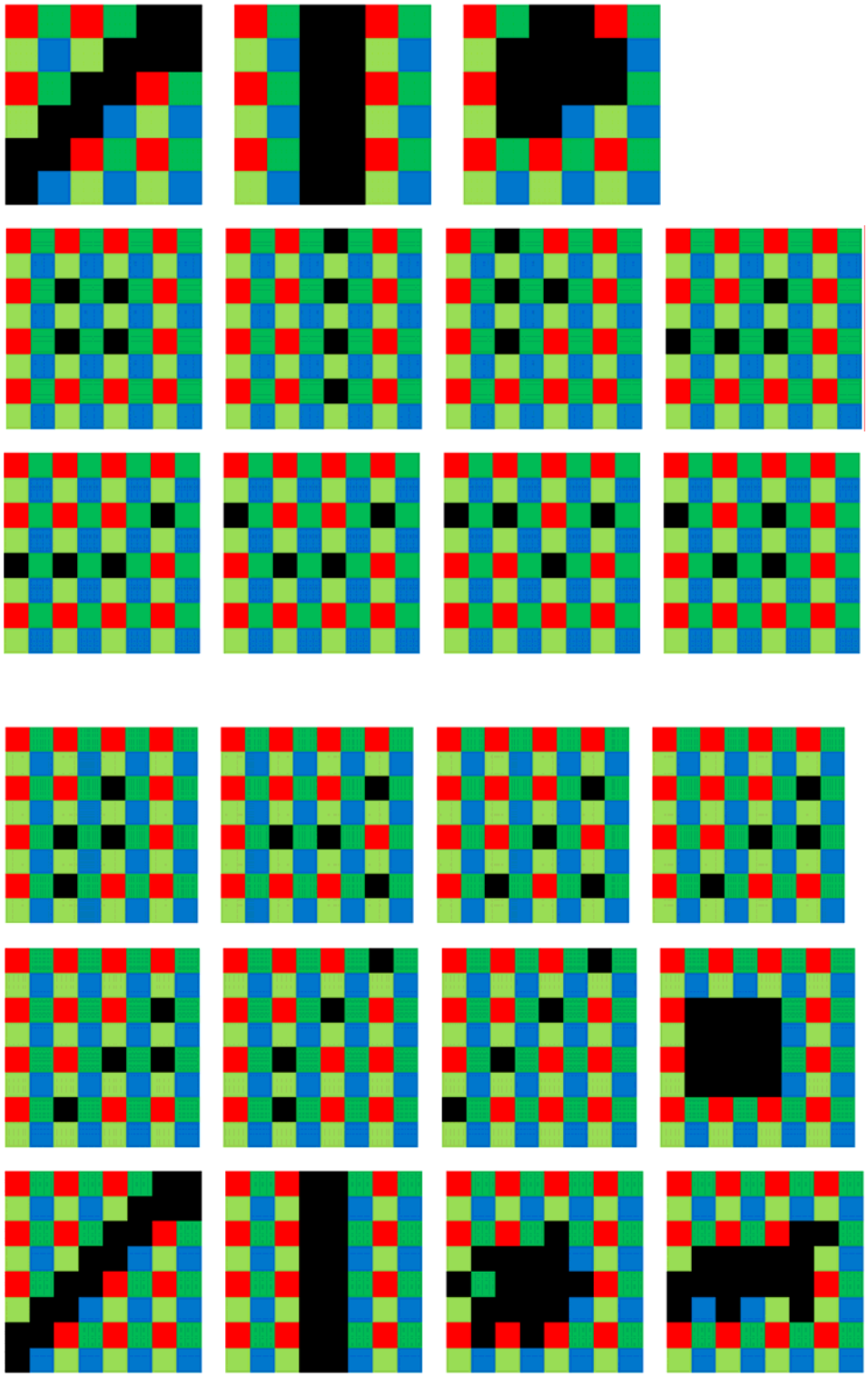


Figure 6-4 Defect pixel cluster that cannot be completely corrected

1. Three defect pixels (single channel)



2. Defect pixel cluster of multiple color channels (three defect pixels for a single channel)



NOTE

- As the memory for storing the defect pixel information is limited, static DPC may be unable to completely correct the defect pixels in the sensor. Therefore, static DPC should correct the defect pixels that have the most impact on the picture quality, and dynamic DPC should correct the remaining defect pixels.

- Static DPC can correct at most 4095 static defect pixels.

6.5.3 API Reference

The following are MPIs related to defect pixel correction:

- HI_MPI_ISP_SetDPAttr: Sets the attribute for defect pixel correction.
- HI_MPI_ISP_GetDPAttr: Obtains the attribute for defect pixel correction.
- HI_MPI_ISP_SetDPCalibrate: Sets the defect pixel correction process.
- HI_MPI_ISP_GetDPCalibrate: Obtains the defect pixel correction process.

HI_MPI_ISP_SetDPAttr

[Description]

Sets the attribute for defect pixel correction.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDPAttr(ISP_DEV IspDev, const ISP_DP_ATTR_S*pstDPAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDPAttr	Attribute for defect pixel correction	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

HI_MPI_ISP_GetDPAttr

[Description]

Obtains the attribute for defect pixel correction.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDefectPixelAttr(ISP_DEV IspDev,
ISP_DP_ATTR_S*pstDPAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDPAttr	Attribute for defect pixel correction	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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]

None

HI_MPI_ISP_SetDPCalibrate

[Description]

Sets the defect pixel correction process.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDPCalibrate(ISP_DEV IspDev, const
ISP_DP_STATIC_CALIBRATE_S *pstDPCalibrate);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDPCalibrate	Attribute for static defect pixel correction	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 calibrate the static defect pixels. The device needs to be calibrated during the production and the calibration result is stored in the flash memory. When the device is powered on, you can read the calibration result from the flash memory and import the result to the system by calling HI_MPI_ISP_SetDPAttr. This MPI is used to calibrate both bright and dark defect pixels.
- To calibrate bright defect pixels, cover the lens or close the shutter, set appropriate parameter values and call this MPI to start the bright defect pixel calibration. Ensure that the frame rate is about 5 fps, the sensor exposure time is about 200 ms, and the gain value is the minimum value during the calibration. Call HI_MPI_ISP_GetDPCalibrate to query the value of **enStatus**. If the value is queried successfully, the coordinate lookup table of defect pixels is obtained and the calibration is complete. Otherwise, the calibration program exits due to timeout. In this case, adjust the parameters and call this MPI again.
- To calibrate dark defect pixels, open the lens and shutter, use a fixed illuminant such as a luminance box or other uniform illuminants, adjust the AE to set the luminance of the entire image to 50% of the maximum luminance or set the B component luminance in the raw data format to 20% of the maximum luminance. Set appropriate parameter values and call this MPI to start the dark defect pixel calibration. Call HI_MPI_ISP_GetDPCalibrate to query the value of **enStatus**. If the value is queried successfully, the coordinate lookup table of defect pixels is obtained and the calibration

is complete. Otherwise, the calibration program exits due to timeout. In this case, adjust the parameters and call this MPI again.

- During the production, if the device is calibrated successfully once, the value of **u8FinishThresh** can be obtained by calling HI_MPI_ISP_GetDPCalibrate and the value of **u8FinishThresh** can be assigned to **u8StartThresh** as the initial value. In this way, the subsequent calibration can be accelerated.

[Example]

None

[See Also]

HI_MPI_ISP_GetDPAttr

HI_MPI_ISP_GetDPCalibrate

[Description]

Obtains the defect pixel correction process.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDPCalibrate(ISP_DEV IspDev, const
ISP_DP_STATIC_CALIBRATE_S *pstDPCalibrate);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDPCalibrate	Attribute for static defect pixel correction	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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]

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_DP_DYNAMIC_ATTR_S: Defines the attribute for dynamic defect pixel correction.
- ISP_DP_STATIC_ATTR_S: Defines the attribute for static defect pixel correction.
- ISP_DP_STATIC_CALIBRATE_S: Defines the attribute for the static defect pixel correction process.
- ISP_STATIC_DP_TYPE_E: Defines the type of static defect pixel correction.
- ISP_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
{
    ISP_DP_DYNAMIC_ATTR_S stDynamicAttr;
    ISP_DP_STATIC_ATTR_S  stStaticAttr;
} ISP_DP_ATTR_S;
```

[Member]

Member	Description
stDynamicAttr	Dynamic correction attribute
stStaticAttr	Static correction attribute

[Note]

None

[See Also]

- ISP_DP_DYNAMIC_ATTR_S
- ISP_DP_STATIC_ATTR_S

ISP_DP_DYNAMIC_ATTR_S

[Description]

Defines the attribute for dynamic defect pixel correction.

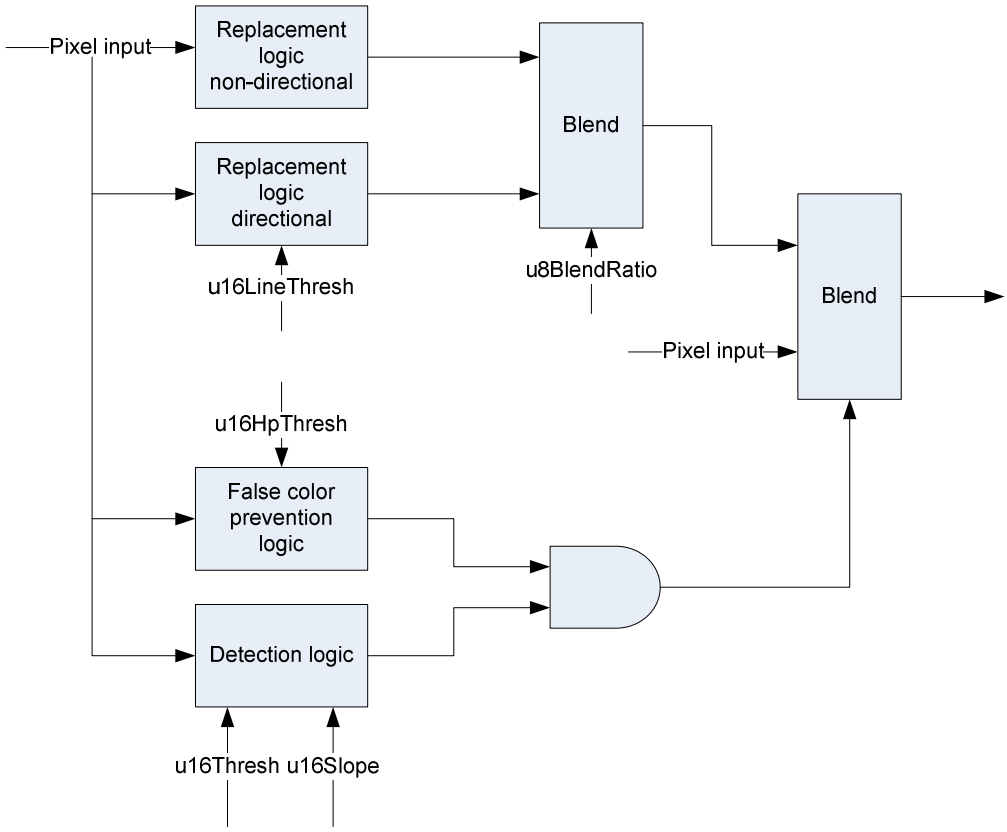
[Syntax]

```
typedef struct hiISP_DP_DYNAMIC_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16Slope;
    HI_U16  u16Thresh;
    HI_U16  u16LineThresh;
    HI_U16  u16HpThresh;
    HI_U8   u8BlendRatio;
} ISP_DP_DYNAMIC_ATTR_S;
```

[Member]

Member	Description
bEnable	Dynamic defect pixel correction enable HI_TRUE (default): enabled HI_FALSE: disabled
u16Slope	Strength of dynamic defect pixel correction Value range: [0, 0xFFF]
u16Thresh	Threshold for dynamic defect pixel detection Value range: [0, 0xFFF]
u16LineThresh	Selected threshold with the direction replacement value during defect pixel correction. A larger value indicates fewer values with directions selected. Value range: [0, 0xFFF]
u16HpThresh	Threshold for edge false color due to over-correction. A smaller value indicates less edge false color due to over-correction. Value range: [0, 0xFFF]
u8BlendRatio	Weight with direction replacement values during defect pixel correction. The weight without direction replacement is (0xFF − u8BlendRatio). Value range: [0, 0xFF]

Figure 6-5 Dynamic defect pixel correction



[Note]

None

[See Also]

None

ISP_DP_STATIC_ATTR_S

[Description]

Defines the attribute for static defect pixel correction.

[Syntax]

```
typedef struct hiISP_DP_STATIC_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16BrightCount;
    HI_U16  u16DarkCount;
    HI_U32  au32BrightTable[STATIC_DP_COUNT_MAX];
    HI_U32  au32DarkTable[STATIC_DP_COUNT_MAX];
    HI_BOOL bShow;
} ISP_DP_STATIC_ATTR_S;
```

[Member]

Member	Description
bEnable	Static defect pixel correction enable HI_TRUE (default): enabled HI_FALSE: disabled
u16BrightCount	Number of bright defect pixels Value range: [0, 0xFFF] Note: When this member acts as the input, it indicates the number of bright defect pixels. When this member acts as the output, it indicates the total number of bright and dark defect pixels.
u16DarkCount	Number of dark defect pixels Value range: [0, 0xFFF] Note: When this member acts as the input, it indicates the number of dark defect pixels. When this member acts as the output, it is **0**.
au32BrightTable	Coordinates of bright defect pixels. Lower 24 bits are valid. Bits 0−11 indicate the horizontal coordinates; and bits 12–23 indicate the vertical coordinates. Note: When this member acts as the input, it indicates the coordinate lookup table for bright defect pixels. When this member acts as the output, it indicates the coordinate lookup table for bright and dark defect pixels.
au32DarkTable	Coordinates of dark defect pixels. Lower 24 bits are valid. Bits 0−11 indicate the horizontal coordinates; and bits 12–23 indicate the vertical coordinates. Note: When this member acts as the input, it indicates the number of dark defect pixels. When this member acts as the output, it is invalid.
bShow	Defect pixel coordinate enable

[Note]

None

[See Also]

None

ISP_DP_STATIC_CALIBRATE_S

[Description]

Defines the attribute for the static defect pixel correction process.

[Syntax]

```
typedef struct hiISP_DP_STATIC_CALIBRATE_S
{
    HI_BOOL bEnable;
```

```
    HI_BOOL bEnableDetect;
    ISP_STATIC_DP_TYPE_E enStaticDPType;
    HI_U8   u8StartThresh;
    HI_U16  u16CountMax;
    HI_U16  u16CountMin;
    HI_U16  u16TimeLimit;
    HI_U32  u32Table[STATIC_DP_COUNT_MAX];
    HI_U8   u8FinishThresh;
    HI_U16  u16Count;
    ISP_STATUS_E enStatus;
} ISP_DP_STATIC_CALIBRATE_S;
```

[Member]

Member	Description
bEnable	Static defect pixel correction enable HI_TRUE (default): enabled HI_FALSE: disabled
bEnableDetect	Static defect pixel detection enable HI_TRUE: enabled HI_FALSE (default): disabled
enStaticDPType	Type of static defect pixel correction
u8StartThresh	Threshold for starting static defect pixel detection Value range: [1, 0xFF]
u16CountMax	Maximum number of allowed static defect pixels Value range: [0, 0xFFF]
u16CountMin	Minimum number of allowed static defect pixels Value range: [0, CountMax]
u16TimeLimit	Correction timeout threshold Value range: [0, 0x640]
u32Table	Coordinate lookup table for bright and dark defect pixels, read-only. Lower 24 bits are valid. Bits 0−11 indicate the horizontal coordinates; and bits 12–23 indicate the vertical coordinates.
u8FinishThresh	Read-only, threshold for stopping static defect pixel detection
u16Count	Read-only, number of detected defect pixels
enStatus	Read-only, result and status of static defect pixel detection

[Note]

- The ISP defect pixel detection is successful if the number of detected defect pixels ranges from **u16CountMax** to **u16CountMin**. You need to adjust the values of **u16CountMax** and **u16CountMin** when different sensors are used.
- **u8FinishThresh** is used only as output. If the sensors of the same type are used, you can set an appropriate **u8StartThresh** value based on the value of **u8FinishThresh** for accelerating static defect pixel correction.
- **u16Slope** and **u16Thresh** are used during static defect pixel correction, and they affect the correction result. Typically, **u16Slope** is set to **0x200**, and **u16Thresh** is set to **0x40**. If **u16Slope** and **u16Thresh** are set to default values and the number of static defect pixels is much greater than the number of allowed defect pixels to be corrected, you can increase the values of **u16Slope** and **u16Thresh**.

[See Also]

- ISP_STATIC_DP_TYPE_E
- ISP_STATUS_E

ISP_STATIC_DP_TYPE_E

[Description]

Defines the type of static defect pixel correction.

[Syntax]

```
typedef enum hiISP_STATIC_DP_TYPE_E{
    ISP_STATIC_DP_BRIGHT = 0x0,
    ISP_STATIC_DP_DARK,
    ISP_STATIC_DP_BUTT
} ISP_STATIC_DP_TYPE_E;
```

[Member]

Member	Description
ISP_STATIC_DP_BRIGHT	0x0: bright defect pixel correction
ISP_STATIC_DP_DARK	0x1: dark defect pixel correction

[Note]

None

[See Also]

None

ISP_STATUS_E

[Description]

Defines the ISP correction or detection status.

[Syntax]

```
typedef enum hiISP_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

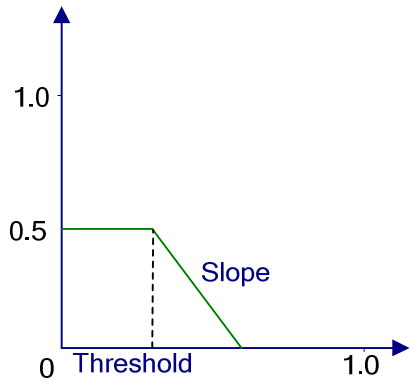
6.6.1 Overview

The crosstalk removal module balances the differences between the Gr and Gr 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 6-6, 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 6-6 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(ISP_DEV IspDev, const ISP_CR_ATTR_S
*pstCRAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCRAttr	Crosstalk attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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(ISP_DEV IspDev, ISP_CR_ATTR_S
*pstCRAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCRAttr	Crosstalk attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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    au8Strength[ISP_AUTO_STENGTH_NUM];
    HI_U8    u8Sensitivity;
    HI_U16   u16Threshold;
    HI_U16   u16Slope;
}ISP_CR_ATTR_S;
```

[Member]

Member	Description
bEnable	Crosstalk removal enable
u8Strength[ISP_AUTO_STENGTH_NUM]	Used for correction. Typically, the default value is used. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-4.
u8Sensitivity	Crosstalk removal sensitivity

Member	Description
u16Threshold	Crosstalk removal threshold Value range: [0, 0xFFF]
u16Slope	Crosstalk removal slope Value range: [0, 0xFFF]

Table 6-4 Mapping between the values of u8Strength[ISP_AUTO_STENGTH_NUM] and gains

u8Strength	Again x Dgain x ISPDgain (Multiple)
u8Strength [0]	0x55
u8Strength [1]	0x55
u8Strength [2]	0x55
u8Strength [3]	0x55
u8Strength [4]	0x55
u8Strength [5]	0x55
u8Strength [6]	0x37
u8Strength [7]	0x37
u8Strength [8]	0x37
u8Strength [9]	0x37
u8Strength [10]	0x37
u8Strength [11]	0x37
u8Strength [12]	0x37
u8Strength [13]	0x37
u8Strength [14]	0x37
u8Strength [15]	0x37

[Note]

- A larger **u8Sensitivity** value indicates stronger green balance on the detected edge.
- 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 Noise Reduction

6.7.1 Overview

The NR algorithm is used in the spatial domain for processing the raw data. The edges and textures are retained during noise reduction. The NR algorithm processes data based on the noise features of the sensor, which are obtained by using the offline correction tool.

6.7.2 Function Description

The NR algorithm supports the following modes:

- Automatic mode

  The NR threshold is in non-linear proportion to the system gain. The NR threshold 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 NR threshold and the system gain, see descriptions of member variables in ISP_NR_ATTR_S.

- Manual mode

  The actual NR threshold is the same as the target value.

  If the denoising (two-dimensional NR of the ISP module and three-dimensional NR of the VPSS module) strength is strong in low illumination scenarios, the dark regions in the picture become darker and the bright regions in the picture become brighter. There are more bright noises than dark noises in dark regions because the pixel values of the dark noises are restricted to 0. After denoising, more bright noises are reduced than dark noises. As a result, the dark regions become darker. There are more dark noises than bright noises in bright regions because the pixel values of the bright noises are restricted to the maximum value. After denoising, more dark noises are reduced than bright noises. As a result, the bright regions become brighter. To maintain the luminance in dark and bright regions, you are advised to adjust the gamma or dynamic contrast improvement (DCI) module.

  If the strength of spatial-domain denoising (two-dimensional NR of the ISP module) is strong in low illumination scenarios, the picture may have a green cast. The reason is that low-frequency color noises cannot be removed by spatial-domain denoising. When many high-frequency noises are removed, the remaining low-frequency color noises result in the green cast. You are advised not to set the strength of the spatial domain denoising to a large value. It is recommended that noises be removed by using the three-dimensional NR function of the VPSS module and increase the chrominance denoising strength of the three-dimensional NR function.

6.7.3 API Reference

The following are NR MPIs:

- HI_MPI_ISP_SetNRAttr: Sets the NR attribute.
- HI_MPI_ISP_GetNRAttr: Obtains the NR attribute.
- HI_MPI_ISP_SetNPTable: Sets the noise features.
- HI_MPI_ISP_GetNPTable: Obtains the noise features.

HI_MPI_ISP_SetNRAttr

[Description]

Sets the NR attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetNRAttr(ISP_DEV IspDev, const ISP_NR_ATTR_S *pstNRAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstNRAttr	NR attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_GetNRAttr

HI_MPI_ISP_GetNRAttr

[Description]

Obtains the NR attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetNRAttr(ISP_DEV IspDev, ISP_NR_ATTR_S *pstNRAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstNRAttr	NR attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetNRAttr

HI_MPI_ISP_SetNPTable

[Description]

Sets the noise features.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetNPTable(ISP_DEV IspDev, const ISP_NP_TABLE_S
```

```
*pstNPTable);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstNPTable	Noise features	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_GetNPTable

HI_MPI_ISP_GetNPTable

[Description]

Obtains the noise features.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetNPTable(ISP_DEV IspDev, ISP_NP_TABLE_S *pstNPTable);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstNPTable	Noise features	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetNPTable

6.7.4 Data Structures

The following are the data structures related to NR:

- ISP_NR_MANUAL_ATTR_S: Defines the manual ISP NR attribute.
- ISP_NR_AUTO_ATTR_S: Defines the automatic ISP NR attribute.
- ISP_NR_ATTR_S: Defines the ISP NR attribute.
- ISP_NP_TABLE_S: Defines the ISP noise features.

ISP_NR_MANUAL_ATTR_S

[Description]

Defines the manual ISP NR attribute.

[Syntax]

```
typedef struct hiISP_NR_MANUAL_ATTR_S
{
    HI_U8 u8Thresh;
    HI_U8 u8ThreshLong;
} ISP_NR_MANUAL_ATTR_S;
```

[Member]

Member	Description
u8Thresh	Picture NR threshold in linear and sensor built-in WDR modes; NR threshold of the short-exposure picture in FS WDR mode Value range: [0, 0xFF]
u8ThreshLong	Invalid in linear and sensor built-in WDR modes; NR threshold of the long-exposure picture in FS WDR mode Value range: [0, 0xFF]

[Note]

A larger **u8ThreshTarget** value indicates greater NR effect when manual NR is enabled.

[See Also]

None

ISP_NR_AUTO_ATTR_S

[Description]

Defines the automatic ISP NR attribute.

[Syntax]

```
typedef struct hiISP_NR_AUTO_ATTR_S
{
    HI_U8 au8Thresh[ISP_AUTO_STENGTH_NUM];
} ISP_NR_AUTO_ATTR_S;
```

[Member]

Member	Description
au8Thresh[ISP_AUTO_STENGTH_NUM]	Image NR strength in automatic mode. The 16 values in this array correspond to the NR strength under different sensor gains. A larger gain corresponds to larger NR strength. For details about the mapping, see Table 6-5.

Table 6-5 Mapping between the values of au8Thresh [ISP_AUTO_STENGTH_NUM] and gains

au8Thresh	Again x Dgain x ISPDgain (Multiple)
au8Thresh [0]	1
au8Thresh [1]	2
au8Thresh [2]	4
au8Thresh [3]	8
au8Thresh [4]	16
au8Thresh [5]	32
au8Thresh [6]	64
au8Thresh [7]	128
au8Thresh [8]	256
au8Thresh [9]	512
au8Thresh [10]	1024
au8Thresh [11]	2048
au8Thresh [12]	4096
au8Thresh [13]	8192
au8Thresh [14]	16384
au8Thresh [15]	32768

[Note]

- In two-frame combination WDR mode, the actual denoising threshold for the short frames is calculated as follows: au8Thresh[*i*] + au8NPTable[127]. The actual denoising threshold for the long frames is calculated as follows: au8Thresh[*i*] + au8NPTable[127]/ExpRatio. **ExpRatio** indicates the ratio of the long frame exposure to the short frame exposure.
- In two-frame combination WDR mode, the denoising strength is strong when the default corrected **au8NPTable** is used and **au8Thresh** is set to the minimum value **0**. In this case, **au8NPTable[127]** should be decreased.

[See Also]

None

ISP_NR_ATTR_S

[Description]

Defines the ISP NR attribute.

[Syntax]

```
typedef struct hiISP_NR_ATTR_S
```

```
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_NR_MANUAL_ATTR_S stManual;
    ISP_NR_AUTO_ATTR_S   stAuto;
} ISP_NR_ATTR_S;
```

[Member]

Member	Description
bEnable	NR enable
enOpType	NR mode
stManual	Image NR threshold in manual mode
stAuto	Image NR threshold in automatic mode

[Note]

None

[See Also]

None

ISP_NP_TABLE_S

[Description]

Defines the ISP noise features.

[Syntax]

```
typedef struct hiISP_NP_TABLE_S
{
    HI_U8 au8NPTable[NP_NODE_NUM_MAX];
} ISP_NP_TABLE_S;
```

[Member]

Member	Description
au8NPTable	Noise features. The default value is specified by **au8NoiseProfileWeightLut** of the ISP_CMOS_NOISE_TABLE_S structure in **cmos.c**.

[Note]

- **au8NPTable** describes the baseline denoising threshold for the luminance of different pictures. Subscripts correspond to the normalized picture luminance (the subscript 0 corresponds to the pixel value 0 of the 12-bit raw data, and the subscript 127 corresponds

to the pixel value 4095 of the 12-bit raw data), and elements correspond to baseline denoising thresholds. The final denoising threshold of the picture is the sum of the baseline denoising threshold determined by **au8NPTable** and the denoising threshold determined by ISP_NR_ATTR_S.

- The denoising module processes the pictures with black level. Therefore, the first several elements of **au8NPTable** are set to **0**.
- **au8NPTable** is generated by the offline correction tool and does not need to be modified. To adjust the denoising strengths in some luminance regions of the picture, the corresponding luminance values of **au8NPTable** can be changed.
- In two-frame combination WDR mode, **au8NPTable** indicates the noise feature of the long frame pictures. In two-frame combination WDR mode, after the root extraction of the picture, the first 126 elements of **au8NPTable** are set to **0** while the one-hundred and twenty-seventh element is set to a non-zero value.

[See Also]

None

6.8 DIS

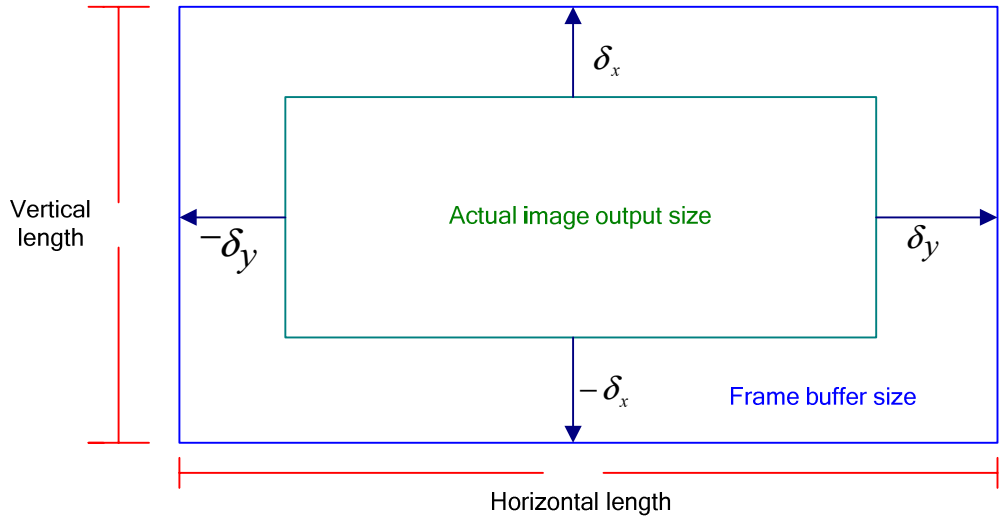
6.8.1 Overview

The digital image stabilization (DIS) module calculates the offset of the current image by comparing it with the previous frame, and transfers the result to downstream modules. Then the output image range is adjusted, and the image becomes stable. After each frame is processed by the DIS module, a group of crop start points (X, Y) is obtained. The images become stable after being cropped by the back-end VPSS.

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 6-7 shows the offset schematic diagram of the DIS module.

Figure 6-7 Offset schematic diagram of the DIS module



As shown in Figure 6-7, the green rectangle is the size of the image output by the VPSS, 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. The VPSS crops the image based on the two offsets and outputs a stable image.

The DIS function is available only in VI-VPSS offline mode. In addition, the DIS function must be set to absolute mode (VPSS_CROP_ABS_COOR) in VPSS crop mode. Otherwise, there is no DIS effect.

6.8.3 API Reference

The following are DIS MPIs:

- HI_MPI_ISP_SetDISAttr: Sets the DIS attribute.
- HI_MPI_ISP_GetDISAttr: Gets the DIS attribute.

HI_MPI_ISP_SetDISAttr

[Description]

Sets the DIS attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDISAttr(ISP_DEV IspDev, const ISP_DIS_ATTR_S
*pstDISAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDISAttr	DIS attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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]

```
/* SET & ENABLE  VI  DEV */
stChnAttr.stCapRect.s32X = 0;
stChnAttr.stCapRect.s32Y = 0;
stChnAttr.stCapRect.u32Width = 1280+100;
stChnAttr.stCapRect.u32Height = 720+100;
stChnAttr.stDestSize.u32Width = stChnAttr.stCapRect.u32Width;
stChnAttr.stDestSize.u32Height = stChnAttr.stCapRect.u32Height;
s32Ret  = HI_MPI_VI_SetChnAttr(ViChn, &stChnAttr);
if (s32Ret != HI_SUCCESS)
{
 SAMPLE_PRT("HI_MPI_VI_SetChnAttr failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
s32Ret  = HI_MPI_VI_EnableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
     SAMPLE_PRT("HI_MPI_VI_EnableChn failed with %#x\n", s32Ret);
     return HI_FAILURE;
 }
s32Ret = HI_MPI_VPSS_CreateGrp(VpssGrp, &stVpssGrpAttr);
if (s32Ret != HI_SUCCESS)
```

```
{
    SAMPLE_PRT("HI_MPI_VPSS_CreateGrp failed with %#x\n", s32Ret);
    return HI_FAILURE;
 }
s32Ret = HI_MPI_VPSS_StartGrp(VpssGrp);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_StartGrp failed with %#x\n", s32Ret);
    return HI_FAILURE;
 }
VPSS_CROP_INFO_S stVpssCropInfo;

stVpssCropInfo.bEnable = HI_TRUE;
stVpssCropInfo.enCropCoordinate = VPSS_CROP_ABS_COOR;
stVpssCropInfo.stCropRect.s32X = 50;
stVpssCropInfo.stCropRect.s32Y = 50;
stVpssCropInfo.stCropRect.u32Width  = 1280;
stVpssCropInfo.stCropRect.u32Height = 720;
s32Ret = HI_MPI_VPSS_SetGrpCrop(VpssGrp, &stVpssCropInfo);
if (s32Ret != HI_SUCCESS)
{
   SAMPLE_PRT("HI_MPI_VPSS_SetGrpCrop failed with %#x\n", s32Ret);
   return HI_FAILURE;
 }
s32Ret = HI_MPI_VPSS_SetChnAttr(VpssGrp, VpssChn, &stVpssChnAttr);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_SetChnAttr failed with %#x\n", s32Ret);
    return HI_FAILURE;
 }
 s32Ret = HI_MPI_VPSS_EnableChn(VpssGrp, VpssChn);
 if (s32Ret != HI_SUCCESS)
 {
    SAMPLE_PRT("HI_MPI_VPSS_EnableChn failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
ISP_DIS_ATTR_S stDisAttr;
stDisAttr.bEnable = HI_TRUE;
int IspDev = 0;
s32Ret =  HI_MPI_ISP_SetDISAttr(IspDev, &stDisAttr);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_ISP_SetDISAttr failed with %#x\n", s32Ret);
    return HI_FAILURE;
```

```
}
...
```

[See Also]

None.

HI_MPI_ISP_GetDISAttr

[Description]

Obtains the DIS attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDISAttr(ISP_DEV IspDev, ISP_DIS_ATTR_S *pstDISAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDISAttr	DIS attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

6.8.4 Data Structure

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

6.9 Anti-Fog

6.9.1 Function Description

The anti-fog 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_SetDeFogAttr: Sets the anti-fog attribute.
- HI_MPI_ISP_GetDefogAttr: Obtains the anti-fog attribute.

HI_MPI_ISP_SetDeFogAttr

[Description]

Set the anti-fog attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetDeFogAttr(ISP_DEV IspDev, const ISP_DEFOG_ATTR_S
*pstDefogAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDefogAttr	Anti-fog attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_GetDefogAttr

HI_MPI_ISP_GetDefogAttr

[Description]

Obtains the anti-fog attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetDeFogAttr(ISP_DEV IspDev, ISP_DEFOG_ATTR_S
```

```
*pstDefogAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAntiFog	Anti-fog attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetDeFogAttr

6.9.3 Data Structures

The following are the data structures related to anti-fog:

- ISP_DEFOG_MANUAL_ATTR_S: Defines the manual anti-fog attribute.
- ISP_DEFOG_AUTO_ATTR_S: Defines the automatic anti-fog attribute.
- ISP_DEFOG_ATTR_S: Defines the ISP anti-fog attribute.

ISP_DEFOG_MANUAL_ATTR_S

[Description]

Defines the manual anti-fog attribute.

[Syntax]

```
typedef struct hiISP_DEFOG_MANUAL_ATTR_S
{
    HI_U8   u8strength;
}ISP_DEFOG_MANUAL_ATTR_S;
```

[Member]

Member	Description
u8Strength	Manual anti-fog strength Value range: [0, 255]

[Note]

None

[See Also]

None

ISP_DEFOG_AUTO_ATTR_S

[Description]

Defines the automatic anti-fog attribute.

[Syntax]

```
typedef struct hiISP_DEFOG_AUTO_ATTR_S
{
    HI_U8            u8strength;
}ISP_DEFOG_AUTO_ATTR_S;
```

[Member]

Member	Description
u8strength	Weighted coefficient for automatic anti-fog strength. The final anti-fog strength is the anti-fog strength of each block multiplied by the weighted coefficient. Value range: [0, 255]

ISP_DEFOG_ATTR_S

[Description]

Defines the ISP anti-fog attribute.

[Syntax]

```
typedef struct hiISP_DEFOG_ATTR_S
{
    HI_BOOL          bEnable;
    HI_U8            u8HorizontalBlock;
    HI_U8            u8VerticalBlock;
    ISP_OP_TYPE_E     enOpType;
    ISP_DEFOG_MANUAL_ATTR_S stManual;
    ISP_DEFOG_AUTO_ATTR_S   stAuto;
}ISP_DEFOG_ATTR_S;
```

[Member]

Member	Description
bEnable	Anti-fog enable
u8HorizontalBlock	Number of horizontal blocks. A smaller value indicates a larger possibility of global anti-flog and lower CPU usage. Value range: [1, 16] The maximum value 16 is recommended.
u8VerticalBlock	Number of vertical blocks. A smaller value indicates a larger possibility of global anti-flog and lower CPU usage. Value range: [1, 15] The maximum value 15 is recommended.
enOpType	Operating mode
stManual	Manual mode
stAuto	Automatic mode

[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(ISP_DEV IspDev, const
ISP_ANTI_FALSECOLOR_S *pstAntiFC)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAntiFC	Anti-false color attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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(ISP_DEV IspDev,
ISP_ANTI_FALSECOLOR_S *pstAntiFC)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAntiFC	Anti-false color attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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_SetAntiFalseColorAttr(ISP_DEV IspDev, const
ISP_ANTI_FALSECOLOR_S *pstAntiFC);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstDemosaicAttr	Demosaic attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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
IspDev	ISP device ID	Input
pstDemosaicAttr	Demosaic attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 Structures

The following are the data structures related to demosaic:

- ISP_DEMOSAIC_ATTR_S: Defines the ISP demosaic attribute.
- ISP_DEMOSAIC_CFG_E: Defines the ISP demosaic operating mode.

ISP_DEMOSAIC_ATTR_S

[Description]

Defines the ISP demosaic attribute.

[Syntax]

```
typedef struct hiISP_DEMOSAIC_ATTR_S
{
    HI_U8   u8VhSlope;
    HI_U8   u8AaSlope;
    HI_U8   u8VaSlope;
    HI_U8   u8UuSlope;
    HI_U16  u16VhThresh;
    HI_U16  u16AaThresh;
    HI_U16  u16VaThresh;
    HI_U16  u16UuThresh;
    ISP_DEMOSAIC_CFG_E   enCfgType;
    HI_U8   au8LumThresh[ISP_AUTO_STENGTH_NUM];
    HI_U8   au8NpOffset[ISP_AUTO_STENGTH_NUM];
}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 **u8VhSlope** member of ISP_CMOS_DEMOSAIC_S 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 **u8AaSlope** member of ISP_CMOS_DEMOSAIC_S 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 **u8VaSlope** member of ISP_CMOS_DEMOSAIC_S in the **cmos.c** file.
u8UuSlope	Slope for all edge thresholds. Increasing this parameter value increases the resolution, sharpness, and noise and causes false color. Value range: [0x00, 0xFF] The default value is defined by the **u8UuSlope** member of ISP_CMOS_DEMOSAIC_S in the **cmos.c** file.

Member	Description
u16VhThresh	Threshold for vertical and horizontal edges. Increasing this parameter value decreases the false color, noise, vertical resolution, and horizontal resolution. Value range: [0x0, 0xFFF] 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, 0xFFF] 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, 0xFFF] 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, 0xFFF] The default value is defined by the **uu_thresh** member of cmos_isp_demosaic_t in the **cmos.c** file.
ISP_DEMOSAIC_CFG_E	Debugging mode of the demosaic module. Value range: [0x00, 0xFF] ISP_DEMOSAIC_CFG_DEFAULT: normal output ISP_DEMOSAIC_CFG_VH: VH debugging mode ISP_DEMOSAIC_CFG_AA: AA debugging mode ISP_DEMOSAIC_CFG_VA: VA debugging mode ISP_DEMOSAIC_CFG_UU: UU debugging mode Other values: reserved
au8LumThresh[ISP_AUTO_STENGTH_NUM]	Array for setting the luminance threshold for the sharpness of large edges of images. The value range is [0, 0xFF]. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-6.
au8NpOffset[ISP_AUTO_STENGTH_NUM]	Array for setting image noise parameters. The value range is [0, 0xFF]. The 16 values in this array correspond to different configured values under different sensor gains. For details about the mapping, see Table 6-7.

Table 6-6 Mapping between the values of au8LumThresh[ISP_AUTO_STENGTH_NUM] and gains

u8LumThresh	Again x Dgain x ISPDgain (Multiple)
u8LumThresh [0]	1
u8LumThresh [1]	2
u8LumThresh [2]	4
u8LumThresh [3]	8
u8LumThresh [4]	16
u8LumThresh [5]	32
u8LumThresh [6]	64
u8LumThresh [7]	128
au8LumThresh [8]	256
au8LumThresh [9]	512
au8LumThresh [10]	1024
au8LumThresh [11]	2048
au8LumThresh [12]	4096
au8LumThresh [13]	8192
au8LumThresh [14]	16384
au8LumThresh [15]	32768

Table 6-7 Mapping between the values of au8NpOffset[ISP_AUTO_STENGTH_NUM] and gains

u8NpOffset	Again x Dgain x ISPDgain (Multiple)
u8NpOffset [0]	1
u8NpOffset [1]	2
u8NpOffset [2]	4
u8NpOffset [3]	8
u8NpOffset [4]	16
u8NpOffset [5]	32
u8NpOffset [6]	64
u8NpOffset [7]	128
au8NpOffset [8]	256

u8NpOffset	Again x Dgain x ISPDgain (Multiple)
au8NpOffset [9]	512
au8NpOffset [10]	1024
au8NpOffset [11]	2048
au8NpOffset [12]	4096
au8NpOffset [13]	8192
au8NpOffset [14]	16384
au8NpOffset [15]	32768

[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

ISP_DEMOSAIC_CFG_E

[Description]

Defines the ISP demosaic operating mode.

[Syntax]

```
typedef enum hiISP_DEMOSAIC_CFG_E
{
    ISP_DEMOSAIC_CFG_DEFAULT = 0,
    ISP_DEMOSAIC_CFG_VH,
    ISP_DEMOSAIC_CFG_AA,
    ISP_DEMOSAIC_CFG_VA,
    ISP_DEMOSAIC_CFG_UU,
    ISP_DEMOSAIC_CFG_BUTT,
} ISP_DEMOSAIC_CFG_E;
```

[Member]

Member	Description
ISP_DEMOSAIC_CFG_DEFAULT	Normal operating mode
ISP_DEMOSAIC_CFG_VH	VH debugging mode

Member	Description
ISP_DEMOSAIC_CFG_AA	AA debugging mode
ISP_DEMOSAIC_CFG_VA	VA debugging mode
ISP_DEMOSAIC_CFG_UU	UU debugging mode

[Note]

None

[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(ISP_DEV IspDev, const ISP_BLACK_LEVEL_S
*pstBlackLevel);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstBlackLevel	Black level attributes	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 **bUpdate** member of ISP_CMOS_BLACK_LEVEL_S in **cmos.c** is set to **HI_TRUE**, the black level configuration in **cmos.c** is always used. In this case, this MPI is invalid.

[Example]

None

[See Also]

None

HI_MPI_ISP_GetBlackLevelAttr

[Description]

Obtains black level attributes.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetBlackLevelAttr(ISP_DEV IspDev, ISP_BLACK_LEVEL_S
*pstBlackLevel);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstBlackLevel	Black level attributes	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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, 0xFFF]*/
} 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, 0xFFF] 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

6.13 FPN Removal

6.13.1 Overview

The sensor converts optical signals into electrical signals, transmits the signals to millions of analog-to-digital converters (ADCs), and then outputs images. The pixel output signals from the ADCs vary according to the photodiode size, doping density, contamination during production, and parameter deviation of the metal-oxide-semiconductor field-effect-transistor (MOSFET). The noises caused by the deviations are called fixed pattern noise because the noise is fixed for each pixel.

The FPN removal module is used to remove the FPNs.

The FPN calibration process is as follows:

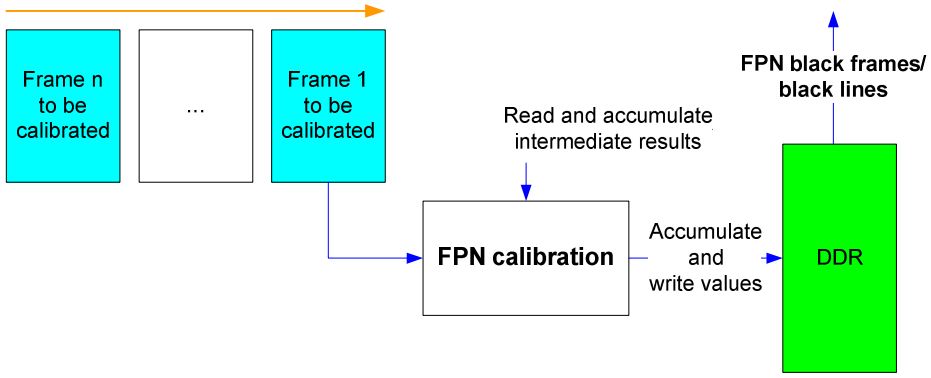
Step 1 Close the lens iris and use the sensor to collect black frames.

Step 2 Enable the calibration program. Write the raw data that meets the calibration requirements to the memory when the first frame is received. When the second frame is received, read the black frame from the memory over a read port, add the second frame with the black frame, and write the accumulated value to the memory over a write port. Repeat the operation for subsequent frames. However, when the last frame is received, the average value of the accumulated black frames is written to the memory.

Step 3 The calibration ends when the average value is written to the memory. In this way, the memory stores the FPNs in low illuminant scenarios. Then you can store the black frames to an external memory.

----End

Figure 6-8 FPN calibration



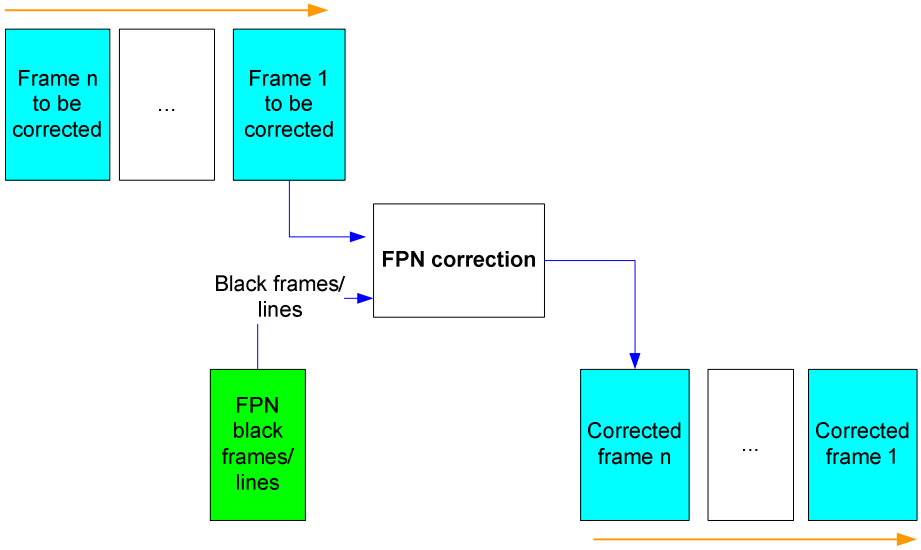
The FPN correction process is as follows:

Step 1 Load the calibrated black frames from the external memory to the internal memory.

Step 2 When the normal images with the FPN collected by the sensor are received, black frames in the memory are removed for each frame to obtain the corrected frames without FPNs.

----End

Figure 6-9 FPN correction



6.13.2 API Reference

The following are FPN removal MPIs:

- HI_MPI_ISP_FPNCalibrate: Sets the calibration attribute of the FPN removal module.
- HI_MPI_ISP_SetFPNAttr: Sets the attribute of the FPN removal module.
- HI_MPI_ISP_GetFPNAttr: Obtains the attribute of the FPN removal module.

HI_MPI_ISP_FPNCalibrate

[Description]

Sets the calibration attribute of the FPN removal module.

[Syntax]

```
HI_S32 HI_MPI_ISP_FPNCalibrate(ISP_DEV IspDev, ISP_FPN_CALIBRATE_ATTR_S
*pstCalibrateAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCalibrateAttr	Pointer to the calibration attribute of the FPN removal module	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

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

[Note]

- Note that the memory used for calibration must be allocated by 16 bits. After the calibration is complete, the black frames or lines, ISO, length of black frames, offsets, and compression flag need to be stored in external storage medium, which is used during the correction.
- In line mode, only the 16-bit non-compressed black line data is supported.
- In frame mode, the 16-bit non-compressed black frame data, and 8/10/12-bit compressed as well as non-compressed black frame data are supported.
- When the front-end frame rate is less than 12 frames, the calibration may fail. Therefore, do not perform FPN calibration at low frame rate.

[Example]

For details about the calibration process, see samples.

[See Also]

HI_MPI_ISP_SetFPNAttr

HI_MPI_ISP_SetFPNAttr

[Description]

Sets the attribute of the FPN removal module.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetFPNAttr(ISP_DEV IspDev, const ISP_FPN_ATTR_S
*pstFPNAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstFPNAttr	Pointer to the attribute of the FPN removal module	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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 information about offsets and ISO is read based on the black frame information stored during the calibration, and then the black frame information is read and stored in the memory. The information is required during the correction. In automatic mode, the correction strength

is adjusted automatically based on the ISO during correction. In manual mode, the correction strength is adjusted based on the input correction strength.

[Example]

```
VI_CHN ViChn = 0;
VI_CHN_ATTR_S stTempChnAttr;
ISP_FPN_ATTR_S stFPNAttr;
SAMPLE_VI_FRAME_INFO_S stVMstFrame;
ISP_DEV IspDev = 0;
ISP_FPN_FRAME_INFO_S *pstFpnFrmInfo;
HI_S32 s32Ret;
HI_U32 u32Iso;


...
/* start&enable vi dev  */
/* start&enable vi chn  */
...
...
/* read dark frame file:dark frame, offset, iso */
...
pstFpnFrmInfo = &stFPNAttr.stFpnFrmInfo;
pstFpnFrmInfo->u32FrmSize = stVMstFrame.u32FrmSize;
memcpy(&pstFpnFrmInfo->stFpnFrame,
       &stVMstFrame.stVideoFrame,
       sizeof(VIDEO_FRAME_INFO_S));
stFPNAttr.bEnable  = HI_TRUE;
stFPNAttr.enOpType = OP_TYPE_AUTO;
stFPNAttr.enFpnType = ISP_FPN_TYPE_FRAME;
stFPNAttr.stFpnFrmInfo.u32Iso = u32Iso;
memcpy(&stFPNAttr.stFpnFrmInfo.stFpnFrame, &stVMstFrame.stVideoFrame,
sizeof(VIDEO_FRAME_INFO_S));
/* start correction  */
HI_MPI_ISP_SetFPNAttr(IspDev, &stFPNAttr);
…
```

[See Also]

HI_MPI_ISP_GetFPNAttr

HI_MPI_ISP_GetFPNAttr

[Description]

Obtains the attribute of the FPN removal module.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetFPNAttr(ISP_DEV IspDev, ISP_FPN_ATTR_S *pstFPNAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstFPNAttr	Pointer to the attribute of the FPN removal module	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_SetFPNAttr

6.13.3 Data Structures

The following are the data structures related to FPN removal:

- ISP_FPN_FRAME_INFO_S: Defines the information about calibrated black frames.
- ISP_FPN_CALIBRATE_ATTR_S: Defines the calibration attribute of the FPN removal module.
- ISP_FPN_ATTR_S: Defines the correction attribute of the FPN removal module.
- ISP_FPN_MANUAL_ATTR_S: Defines the manual correction attribute of the FPN removal module.

- ISP_FPN_AUTO_ATTR_S: Defines the automatic correction attribute of the FPN removal module.

ISP_FPN_FRAME_INFO_S

[Description]

Defines the information about calibrated black frames.

[Syntax]

```
typedef struct hiISP_FPN_FRAME_INFO_S
{
    HI_U32              u32Iso;
    HI_U32             u32Offset;
    HI_U32             u32FrmSize;
    VIDEO_FRAME_INFO_S  stFpnFrame;
}ISP_FPN_FRAME_INFO_S;
```

[Member]

Member	Description
u32Iso	ISO when the black frame is calibrated. The source is the ISO stored during the calibration. The value of the ISO must be greater than 0.
u32Offset	Average value of all the pixels of the black frame. The source is the offset stored during the calibration. Value range: [0, 0xFFF]
u32FrmSize	Frame size. The output black frames include compressed frames and uncompressed frames. The source is the black frame size stored during the calibration.
stFpnFrame	Black frame information. Allocate the memory, read the calibrated black frames from the external memory to the internal memory, and assign a value to this member.

[Note]

None

[See Also]

ISP_FPN_CALIBRATE_ATTR_S

ISP_FPN_CALIBRATE_ATTR_S

[Description]

Defines the calibration attribute of the FPN removal module.

[Syntax]

```
typedef struct hiISP_FPN_CALIBRATE_ATTR_S
{
   HI_U32                      u32Threshold;
   HI_U32                      u32FrameNum;
   ISP_FPN_TYPE_E                enFpnType;
   ISP_FPN_FRAME_INFO_S          stFpnCaliFrame;
}ISP_FPN_CALIBRATE_ATTR_S;
```

[Member]

Member	Description
u32Threshold	Calibration threshold. If the pixel value is above the threshold during calibration, this pixel is considered as a defect pixel and is excluded from calibration.
u32FrameNum	Number of frames to be calibrated. The value range is {1, 2, 4, 8, 16}, that is, the integer exponentiation of 2.
enFpnType	Calibration mode (frame mode or line mode). The correction effect in frame mode is better than that in line mode. However, the memory consumption of the line mode is lower than that of the frame mode.
stFpnCaliFrame	Information about calibrated black frames

[Note]

The threshold value is related to the number of bits of the stored black frames during the calibration. Assume that the number of bits of the stored black frames during the calibration is N, the threshold is (2^N)>>2.

[See Also]

- ISP_FPN_FRAME_INFO_S
- ISP_FPN_ATTR_S

ISP_FPN_ATTR_S

[Description]

Defines the correction attribute of the FPN removal module.

[Syntax]

```
typedef struct hiISP_FPN_ATTR_S
{
    HI_BOOL           bEnable;
    ISP_OP_TYPE_E     enOpType;
    ISP_FPN_TYPE_E        enFpnType;
    ISP_FPN_FRAME_INFO_S  stFpnFrmInfo;
    ISP_FPN_MANUAL_ATTR_S stManual;
    ISP_FPN_AUTO_ATTR_S   stAuto;
```

```
}ISP_FPN_ATTR_S;
```

[Member]

Member	Description
bEnable	Correction enable
enOpType	Correction mode (manual mode or automatic mode)
enFpnType	Correction mode (frame mode or line mode)
stFpnFrmInfo	Black frame information
stManual	Manual correction attribute
stAuto	Automatic correction attribute

[Note]

The manual correction mode and automatic correction mode are supported. In manual mode, you need to configure the correction strength. In automatic mode, the system automatically calculates the correction strength based on the ISO value of the current image.

[See Also]

- HI_MPI_ISP_SetFPNAttr
- HI_MPI_ISP_GetFPNAttr
- ISP_FPN_MANUAL_ATTR_S
- ISP_FPN_AUTO_ATTR_S

ISP_FPN_MANUAL_ATTR_S

[Description]

Defines the manual correction attribute of the FPN removal module.

[Syntax]

```
typedef struct hiISP_FPN_MANUAL_ATTR_S
{
   HI_U32            u32Strength;
}ISP_FPN_MANUAL_ATTR_S;
```

[Member]

Member	Description
u32Strength	Manual correction strength

[Note]

None

[See Also]

None

ISP_FPN_AUTO_ATTR_S

[Description]

Defines the automatic correction attribute of the FPN removal module.

[Syntax]

```
typedef struct hiISP_FPN_AUTO_ATTR_S
{
    HI_U32              u32Strength;
}ISP_FPN_AUTO_ATTR_S;
```

[Member]

Member	Description
u32Strength	Real-time correction strength in automatic mode, read-only (The ISP FWM automatically adjusts the strength according to the ISO)

[Note]

None

[See Also]

None

6.14 ACM

6.14.1 Overview

The ACM can be used to adjust the favorite colors such as green, blue, and complexion by adjusting the luminance, hue, and saturation in a specific zone. The ACM module provides five default coefficient modes.

6.14.2 API Reference

The following are ACM MPIs:

- HI_MPI_ISP_SetAcmAttr: Sets the ACM attribute.
- HI_MPI_ISP_GetAcmAttr: Obtains the ACM attribute.
- HI_MPI_ISP_SetAcmCoeff: Sets the ACM coefficients in a mode and enables the ACM coefficient mode to take effect immediately.
- HI_MPI_ISP_GetAcmCoeff: Obtains the ACM coefficients in a mode.

HI_MPI_ISP_SetAcmAttr

[Description]

Sets the ACM attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAcmAttr(ISP_DEV IspDev, ISP_ACM_ATTR_S *pstAcmAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAcmAttr	ACM attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_GetAcmAttr

[Description]

Obtains the ACM attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAcmAttr(ISP_DEV IspDev, ISP_ACM_ATTR_S *pstAcmAttr)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstAcmAttr	ACM attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_SetAcmCoeff

[Description]

Sets the ACM coefficients in a mode and enables the ACM coefficient mode to take effect immediately.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetAcmCoeff(ISP_DEV IspDev, ISP_ACM_LUT_S *pstAcmCoef,
ISP_ACM_MODE_E enMode)
```

[Parameter]

Parameter	Description	Input/Output
pstAcmCoef	ACM coefficient	Input
enMode	ACM coefficient mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_GetAcmCoeff

[Description]

Obtains the ACM coefficients in a mode.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetAcmCoeff(ISP_DEV IspDev, ISP_ACM_LUT_S *pstAcmCoef,
ISP_ACM_MODE_E enMode)
```

[Parameter]

Parameter	Description	Input/Output
pstAcmCoef	ACM coefficient	Output
enMode	ACM coefficient mode	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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

6.14.3 Data Structures

The following are the data structures related to ACM:

- ISP_ACM_MODE_E: Defines the ACM coefficient mode.
- ISP_ACM_ATTR_S: Defines the ACM attribute.
- ISP_ACM_LUT_S: Defines the ACM coefficient lookup table.

ISP_ACM_MODE_E

[Description]

Defines the ACM coefficient mode.

[Syntax]

```
typedef enum hiISP_ACM_MODE_E
{
    ISP_ACM_MODE_BLUE         = 0,
    ISP_ACM_MODE_GREEN          ,
    ISP_ACM_MODE_BG             ,
    ISP_ACM_MODE_SKIN           ,
    ISP_ACM_MODE_VIVID          ,
    ISP_ACM_MODE_BUTT
} ISP_ACM_MODE_E;
```

[Member]

Member	Description
ISP_ACM_MODE_BLUE	Blue mode
ISP_ACM_MODE_GREEN	Green mode
ISP_ACM_MODE_BG	Blue and green mode
ISP_ACM_MODE_SKIN	Complexion mode
ISP_ACM_MODE_VIVID	Combination mode

[Note]

None

[See Also]

None

ISP_ACM_ATTR_S

[Description]

Defines the ACM attribute.

[Syntax]

```
typedef struct hi_ISP_ACM_ATTR_S
{
    HI_BOOL        bEnable        ;
    HI_BOOL        bDemoEnable    ;
    ISP_ACM_MODE_E  enMode        ;
    HI_U32         u32Stretch     ;
    HI_U32         u32ClipRange   ;
    HI_U32         u32AcmClipOrWrap;
    HI_U32         u32CbcrThr     ;
    HI_U32         u32GainLuma    ;
```

```
    HI_U32          u32GainHue      ;
    HI_U32          u32GainSat      ;
}ISP_ACM_ATTR_S;
```

[Member]

Member	Description
bEnable	ACM enable HI_TRUE: enabled HI_FALSE: disabled
bDemoEnable	Demonstration mode enable. ACM is performed only on the right part.
enMode	ACM coefficient mode select
u32Stretch	Input data range 0: limit range 1: full range
u32ClipRange	Output data range 0: limit range 1: full range
u32AcmClipOrWrap	Decimal processing mode 0: round off 1: truncation
u32CbcrThr	Chrominance adjustment threshold Value range: [0, 255]
u32GainLuma	Luminance gain Value range: [0, 512] Precision: 1/64
u32GainHue	Hue gain Value range: [0, 512] Precision: 1/64
u32GainSat	Saturation gain Value range: [0, 512] Precision: 1/64

[Note]

- You can preset at most five groups of ACM tables (corresponding to five groups of color styles) by configuring **enMode**.
- When the data width is eight bits, the value range of the limited-range Y component is [16, 235], the value range of the Cb/Cr component is [16, 240], and the value range of the full-range Y/Cb/Cr component is [0, 255].

- ACM is implemented to protect the colors similar to gray only when the Cb/Cr component of the pixel is greater than or equal to **u32CbcrThr**. You are advised to set **u32CbcrThr** to **0** to avoid noise amplification.
- You are advised to associate the luminance, hue, and saturation gains with the ISO to avoid noise amplification caused by ACM under low illuminations.

[See Also]

None

ISP_ACM_LUT_S

[Description]

Defines the ACM coefficient lookup table.

[Syntax]

```
typedef struct hi_ISP_ACM_LUT_S
{
    HI_S16 as16Y[ACM_Y_NUM][ACM_S_NUM][ACM_H_NUM];
    HI_S16 as16H[ACM_Y_NUM][ACM_S_NUM][ACM_H_NUM];
    HI_S16 as16S[ACM_Y_NUM][ACM_S_NUM][ACM_H_NUM];
}ISP_ACM_LUT_S;
```

[Member]

Member	Description
as16Y	Luminance lookup table Value range: [−256, +256]
as16H	Hue lookup table Value range: [−64, +64]
as16S	Saturation lookup table Value range: [−256, +256]

[Note]

You can obtain the ACM lookup table and default gain by using the ACM calibration tool.

[See Also]

None

6.15 ColorTone

6.15.1 Overview

The ColorTone module provides the interfaces for hue adjustment. The picture can be adjusted to have a red, green, or blue cast based on user preference by calling the interfaces.

6.15.2 API Reference

The following are ColorTone MPIs:

- HI_MPI_ISP_SetColorToneAttr: Sets the hue attribute.
- HI_MPI_ISP_GetColorToneAttr: Obtains the hue attribute.

HI_MPI_ISP_SetColorToneAttr

[Description]

Sets the hue attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetColorToneAttr(ISP_DEV IspDev, const
ISP_COLOR_TONE_ATTR_S *pstCTAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCTAttr	Hue attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

None

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_GetColorToneAttr

HI_MPI_ISP_GetColorToneAttr

[Description]

Obtains the hue attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetColorToneAttr(ISP_DEV IspDev,ISP_COLOR_TONE_ATTR_S
*pstCTAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstCTAttr	Hue attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

- Header files: hi_awb_comm.h, mpi_awb.h
- Library file: libisp.a

[Note]

None

[Example]

None

[See Also]

HI_MPI_ISP_SetColorToneAttr

6.15.3 Data Structures

The following is the data structure related to ColorTone:

ISP_COLOR_TONE_ATTR_S: Defines the hue adjustment attribute

ISP_COLOR_TONE_ATTR_S

[Description]

Defines the hue adjustment attribute.

[Syntax]

```
typedef struct hiISP_COLOR_TONE_ATTR_S
{
   HI_U16 u16RedCastGain;
   HI_U16 u16GreenCastGain;
   HI_U16 u16BlueCastGain;
} ISP_COLOR_TONE_ATTR_S;
```

[Member]

Member	Description
u16RedCastGain	Gain of the R channel (8-bit decimal precision)
u16GreenCastGain	Gain of the G channel (8-bit decimal precision)
u16BlueCastGain	Gain of the B channel (8-bit decimal precision)

[Note]

You can specify the color style by using the ColorTone function regardless of the color temperature.

[See Also]

None

6.16 WDR

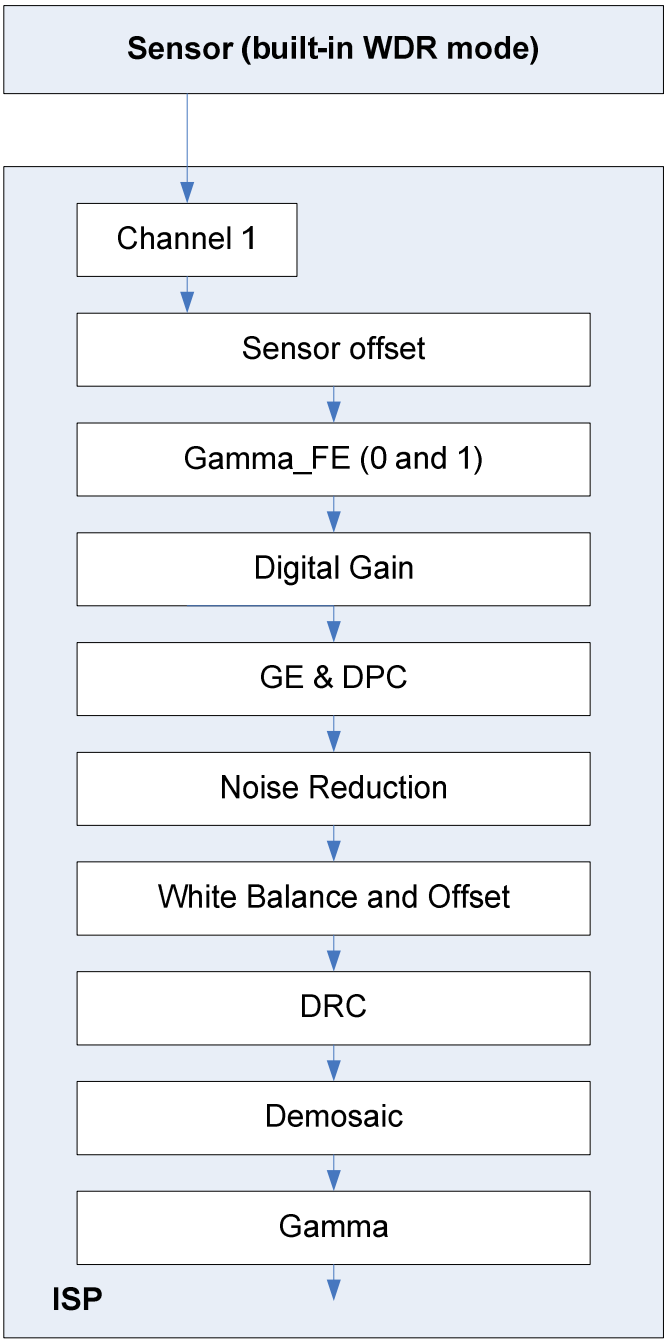
6.16.1 Overview

The WDR-related modules require extra or special configurations for supporting the sensor WDR mode and the multi-frame combination WDR mode. In the ISP pipeline, the GammaFE module extracts a root from the image data and compresses the data as non-linear data. Therefore, the AE and AWB statistics must be squared and converted into linear data before being used. The AE and AWB statistics as well as configuration information such as the ISP digital gain and white balance gain can be configured to ISP registers only after root extraction. When the HiSilicon AE or AWB library is used, the ISP firmware automatically processes data. Pay attention to these differences when you develop your own AE or AWB library.

- As shown in 4, the handling process in sensor built-in WDR mode is as follows:

1. The output image data of the sensor is transferred to the subsequent module by using channel 1.
2. The GammaFE module decompresses the image data which is compressed by segment, and obtains the image data which is greater than 12 bits. Then the 12-bit image data is obtained after the black level processing and non-linear mapping. The processing is implemented by using the lookup table. The GammaFE module must be enabled and placed after the sensor offset module (**bitGammaFePosition** of ISP_MODULE_CTRL_U must be set to 0).
3. The DRC module is used to improve the image dynamic range and must be enabled.
4. The gamma module squares the non-linear image data to convert it into linear data and then maps the gamma curve in linear mode. The gamma curve in WDR mode is different from that in linear mode. The gamma curve in linear mode can be converted into the gamma curve in WDR mode by using the HiSilicon PQ Tools.

Figure 6-10 Handling process in sensor built-in WDR mode

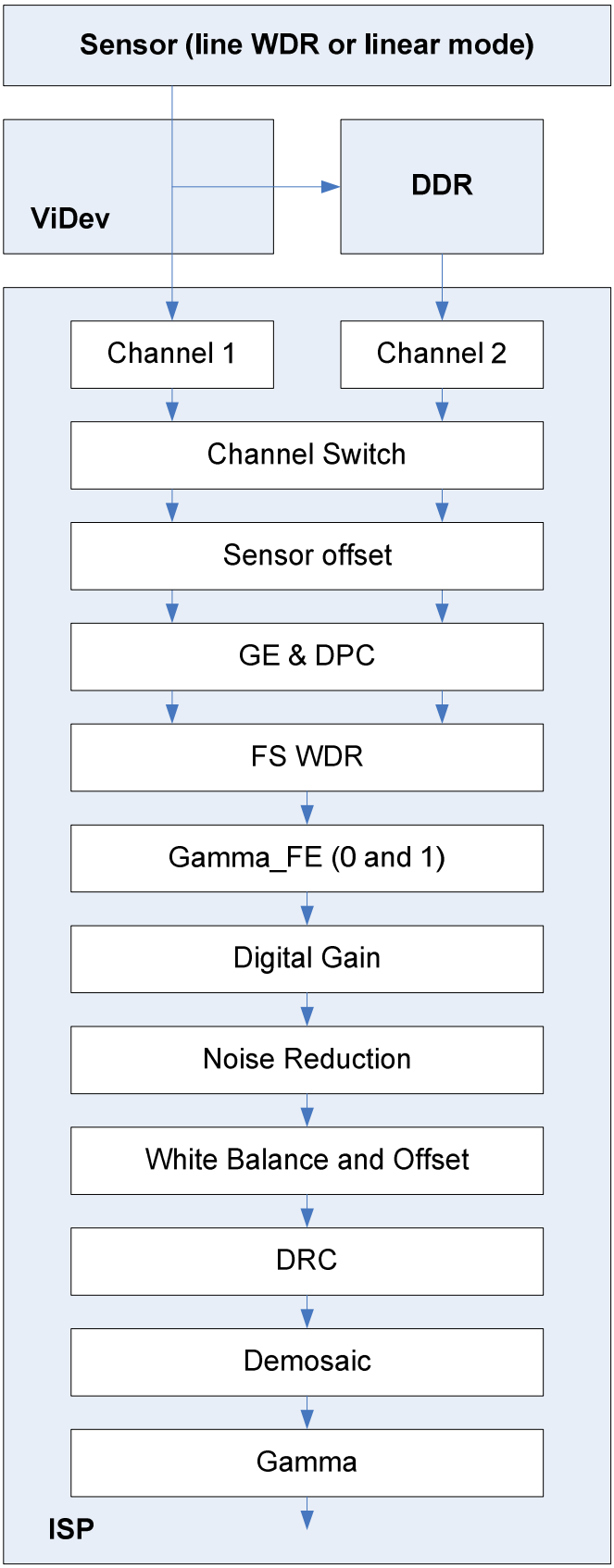


- As shown in Figure 6-11, the handling process in multi-frame combination WDR mode is as follows:

1. The long (short) exposure image data output by the sensor is sent to the DDR, and read from the DDR and sent to channel 2 when the short (long) exposure data is ready. The short (long) exposure image data is directly sent to channel 1.
2. The channel switch module exchanges the image data in channel 1 and channel 2, and generally does not need to be configured.
3. The FS WDR module combines the received long exposure data and short exposure data to generate the 16-bit image data.

4. The GammaFE module obtains the 12-bit image data through non-linear mapping. The GammaFE module must be enabled and placed after the FS WDR module (**bitGammaFePosition** of ISP_MODULE_CTRL_U must be set to **1**).
5. The DRC module is used to improve the image dynamic range and must be enabled.
6. The gamma module squares the non-linear image data to convert it into linear data and then maps the gamma curve in linear mode. The gamma curve in WDR mode is different from that in linear mode. The gamma curve in linear mode can be converted into the gamma curve in WDR mode by using the HiSilicon PQ Tools.

Figure 6-11 Handling process in multi-frame combination WDR mode



6.16.2 API Reference

The following are WDR MPIs:

- HI_MPI_ISP_SetGammaFEAttr: Sets the GammaFE attribute.
- HI_MPI_ISP_GetGammaFEAttr: Obtains the GammaFE attribute.
- HI_MPI_ISP_SetFSWDRAttr: Sets the multi-frame combination WDR attribute.
- HI_MPI_ISP_GetFSWDRAttr: Obtains the multi-frame combination WDR attribute.

HI_MPI_ISP_SetGammaFEAttr

[Description]

Sets the GammaFE attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetGammaFEAttr(ISP_DEV IspDev, const ISP_GAMMAFE_ATTR_S
*pstGammaFEAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstGammaFEAttr	GammaFE attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_GetGammaFEAttr

[Description]

Obtains the GammaFE attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetGammaFEAttr(ISP_DEV IspDev, ISP_GAMMAFE_ATTR_S
*pstGammaFEAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstGammaFEAttr	GammaFE attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_SetFSWDRAttr

[Description]

Sets the multi-frame combination WDR attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetFSWDRAttr(ISP_DEV IspDev, const ISP_WDR_FS_ATTR_S
*pstFSWDRAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstFSWDRAttr	Multi-frame combination WDR attribute	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_GetFSWDRAttr

[Description]

Obtains the multi-frame combination WDR attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetFSWDRAttr(ISP_DEV IspDev, ISP_WDR_FS_ATTR_S
*pstFSWDRAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstFSWDRAttr	Multi-frame combination WDR attribute	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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

6.16.3 Data Structures

The following are the data structures related to WDR:

- ISP_GAMMAFE_ATTR_S: Defines the ISP GammaFE attribute.
- ISP_WDR_FS_ATTR_S: Defines the multi-frame combination WDR attribute of the ISP.

ISP_GAMMAFE_ATTR_S

[Description]

Defines the ISP GammaFE attribute.

[Syntax]

```
typedef struct hiISP_GAMMAFE_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16 u16Table[GAMMA_FE0_NODE_NUM + GAMMA_FE1_NODE_NUM];
} ISP_GAMMAFE_ATTR_S;
```

[Member]

Member	Description
bEnable	GammaFE table enable HI_TRUE: enabled HI_FALSE: disabled
u16Table	GammaFE lookup table Linear mode: The 12-bit sensor can ignore the GammaFE lookup table, and the 14-bit sensor needs to generate the GammaFE lookup table by using an offline calibration tool. Sensor built-in WDR mode: For details, see corresponding sensor manuals. The GammaFE lookup table is generated by using an offline calibration tool. Multi-frame combination WDR mode: The default value is used.

[Note]

None

[See Also]

None

ISP_WDR_FS_ATTR_S

[Description]

Defines the multi-frame combination WDR attribute of the ISP.

[Syntax]

```
typedef struct hiISP_WDR_FS_ATTR_S
{
    HI_BOOL bMotionComp;
    HI_U16  u16ShortThresh;
    HI_U16  u16LongThresh;
    ISP_COMBINE_MODE_E enFSWDRComMode;
} ISP_WDR_FS_ATTR_S;
```

[Member]

Member	Description
bMotionComp	WDR motion compensation enable HI_TRUE: enabled. The smearing effect is mitigated. This is the default value. HI_FALSE: disabled. Under indoor fluorescent lamps, the motion compensation may cause more stripes due to flicker. In this case, the motion compensation can be disabled.
u16ShortThresh	Short exposure threshold. Only short exposure data is selected if the image data is above the threshold. The default value 0xF00 is recommended. Value range: [0x0, 0xFFF]
u16LongThresh	Long exposure threshold. Only long exposure data is selected if the image data is below the threshold. The default value 0xC00 is recommended. Value range: [0x0, u16ShortThresh]
enFSWDRComMode	FS_WDR_COMBINE_SHORT_FIRST: Short exposure data is selected first in the combined zone, where the smearing effect is alleviated. FS_WDR_COMBINE_LONG_FIRST: Long exposure data is selected first in the combined zone, where the SNR is high.

[Note]

This data structure is used for adjusting the image effect only in multi-frame combination WDR mode. If the Panasonic MN34220 sensor is used, it is recommended that the configured value of **u8Speed** be less than 64 in WDR mode in which two frames are combined due to the performance restrictions of the Panasonic MN34220 sensor. Otherwise, flicker may occur when the light changes significantly.

[See Also]

None

6.17 Obtaining ISP Virtual Addresses

6.17.1 Function Description

The ISP consists of the AE library module, AWB library module, AF library module, ISP physical register module, and other module. Each module has a start address.

6.17.2 API Reference

HI_MPI_GetISPRegAttr

[Description]

Obtains the attributes of ISP virtual addresses.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetISPRegAttr(ISP_DEV IspDev, ISP_REG_ATTR_S *
pstIspRegAttr);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspRegAttr	Attributes of ISP virtual addresses.	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 virtual address for the AF library module is not provided currently.

[Example]

None

[See Also]

None

6.17.3 Data Structure

ISP_REG_ATTR_S

[Description]

Defines the attributes of the virtual addresses for the registers of ISP submodules.

[Syntax]

```
typedef struct hiISP_REG_ATTR_S
{
    HI_U32 u32IspRegAddr;
    HI_U32 u32IspRegSize;
    HI_U32 u32IspExtRegAddr;
    HI_U32 u32IspExtRegSize;
    HI_U32 u32AeExtRegAddr;
    HI_U32 u32AeExtRegSize;
    HI_U32 u32AwbExtRegAddr;
    HI_U32 u32AwbExtRegSize;
} ISP_REG_ATTR_S;
```

[Member]

Member	Description
u32IspRegAddr	Start virtual address for the ISP internal physical registers.
u32IspRegSize	Size of the ISP internal physical registers.
u32IspExtRegAddr	Start virtual address for the ISP external virtual registers.
u32IspExtRegSize	Size of the ISP external virtual registers.
u32AeExtRegAddr	Start virtual address for the AE library module.
u32AeExtRegSize	Size of the AE library module.
u32AwbExtRegAddr	Start virtual address for the AWB library module.
u32AwbExtRegSize	Size of the AWB library module.

[Note]

None

[See Also]

None

7 Statistics

7.1 Overview

The ISP provides 3A statistics and related configurations.

7.2 API Reference

The statistics MPIs must be called after HI_MPI_ISP_Init is called. The following are statistics MPIs:

- HI_MPI_ISP_SetStatisticsConfig: Sets the AF statistics attribute.
- HI_MPI_ISP_GetStatisticsConfig: Obtains the AF statistics configuration.
- HI_MPI_ISP_GetStatistics: Obtains ISP statistics.

HI_MPI_ISP_SetStatisticsConfig

[Description]

Sets the AF statistics attribute.

[Syntax]

```
HI_S32 HI_MPI_ISP_SetStatisticsConfig(ISP_DEV IspDev, const
ISP_STATISTICS_CFG_S *pstStatCfg);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstStatCfg	ISP statistics configuration	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
HI_ERR_ISP_NULL_PTR	The pointer is null.

[Requirement]

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

[Note]

Unnecessary statistics can be disabled to improve system performance.

[Example]

None

[See Also]

HI_MPI_ISP_GetStatisticsConfig

HI_MPI_ISP_GetStatisticsConfig

[Description]

Obtains the AF statistics configuration.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetStatisticsConfig(ISP_DEV IspDev, ISP_STATISTICS_CFG_S
*pstStatCfg);
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstStatCfg	ISP statistics configuration	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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_SetStatisticsConfig

HI_MPI_ISP_GetStatistics

[Description]

Obtains ISP statistics.

[Syntax]

```
HI_S32 HI_MPI_ISP_GetStatistics(ISP_DEV IspDev, ISP_STATISTICS_S *pstStat)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstStat	ISP statistics	Output

[Return Value]

Return Value	Description
0	Success

Return Value	Description
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[Error Code]

Error Code	Definition
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

7.3 Data Structures

The following are the data structures related to statistics:

- ISP_STATISTICS_CFG_S: Defines the ISP statistics configuration.
- ISP_STATISTICS_CTRL_U: Defines the enable status of ISP statistics.
- ISP_AE_STATISTICS_CFG_S: Defines the AE statistics configuration.
- ISP_WB_STATISTICS_CFG_S: Defines the AWB statistics configuration.
- ISP_WB_STATISTICS_CFG_PARA_S: Defines AWB statistics parameters.
- ISP_FOCUS_STATISTICS_CFG_S: Defines the AF statistics configuration.
- ISP_AF_CFG_S: Defines AF statistics configuration parameters.
- ISP_AF_PEAK_MODE_E: Defines whether the peak value of the zoned IIR statistics (**u16h1** and **u16h2**) is calculated.
- ISP_AF_SQU_MODE_E: Defines whether the zoned statistics are squared in square mode.
- ISP_AF_H_PARAM_S: Defines the IIR parameter configuration of horizontal filters for AF statistics.
- ISP_AF_V_PARAM_S: Defines the FIR parameter configuration of vertical filters for AF statistics.
- ISP_AF_FV_PARAM_S: Defines parameters for the AF statistics result output format.

- ISP_STATISTICS_S: Defines statistics.
- ISP_AE_STATISTICS_S: Defines AE statistics.
- ISP_WB_STATISTICS_S: Defines AWB statistics.
- ISP_WB_BAYER_STATISTICS_INFO_S: Defines AWB statistics in the Bayer field.
- ISP_WB_RGB_STATISTICS_INFO_S: Defines AWB statistics in the RGB field.
- ISP_FOCUS_STATISTICS_S: Defines AF statistics.
- ISP_FOCUS_ZONE_S: Defines AF statistics parameters.

ISP_STATISTICS_CFG_S

[Description]

Defines the ISP statistics configuration.

[Syntax]

```
typedef struct hiISP_STATISTICS_CFG_S
{
    ISP_STATISTICS_CTRL_U       unKey;
    ISP_AE_STATISTICS_CFG_S     stAECfg;
    ISP_WB_STATISTICS_CFG_S     stWBCfg;
    ISP_FOCUS_STATISTICS_CFG_S  stFocusCfg;
} ISP_STATISTICS_CFG_S;
```

[Member]

Member	Description
unKey	Statistics enable
stAECfg	AE statistics configuration
stWBCfg	AWB statistics configuration
stFocusCfg	AF statistics configuration

[Note]

None

[See Also]

None

ISP_STATISTICS_CTRL_U

[Description]

Defines the enable status of ISP statistics.

[Syntax]

```
typedef union hiISP_STATISTICS_CTRL_U
```

```
{
    HI_U32  u32Key;
    struct
    {
        HI_U32  bit1AeStat1    : 1 ;
        HI_U32  bit1AeStat2    : 1 ;
        HI_U32  bit1AeStat3    : 1 ;
        HI_U32  bit1AeStat4    : 1 ;
        HI_U32  bit1AeStat5    : 1 ;
        HI_U32  bit1AwbStat1   : 1 ;
        HI_U32  bit1AwbStat2   : 1 ;
        HI_U32  bit1AwbStat3   : 1 ;
        HI_U32  bit1AwbStat4   : 1 ;
        HI_U32  bit1AfStat     : 1 ;
        HI_U32  bit1Defog      : 1 ;
        HI_U32  bit22Rsv       : 21;
    };
}ISP_STATISTICS_CTRL_U;
```

[Member]

Member	Description
bit1AeStat1	Global 5-segment histogram enable
bit1AeStat2	5-segment histogram enable for each zone
bit1AeStat3	Global 256-segment histogram enable
bit1AeStat4	Global statistics averaging enable
bit1AeStat5	Statistics averaging enable for each zone
bit1AwbStat1	AWB global statistics enable for the RGB field
bit1AwbStat2	AWB zoned statistics enable for the RGB field
bit1AwbStat3	AWB global statistics enable for the Bayer field
bit1AwbStat4	AWB zoned statistics enable for the Bayer field
bit1AfStat	AF statistics enable

[Note]

You can disable unused statistics to improve the system performance because the statistics are not read during the interrupt.

[See Also]

None

ISP_AE_STATISTICS_CFG_S

[Description]

Defines the AE statistics configuration.

[Syntax]

```
typedef struct hiISP_AE_STATISTICS_CFG_S
{
    HI_U8  au8HistThresh[4];
    ISP_AE_SWITCH_E  enAESwitch;
    ISP_AE_HIST_SWITCH_E  enHistSwitch;
    ISP_AE_SWITCH_E  enAESumSwitch;
} ISP_AE_STATISTICS_CFG_S;
```

[Member]

Member	Description
au8HistThresh	Threshold for the 5-segment histogram
enAESwitch	Positions of the statistics modules for the global and zoned 5-segment histograms in the ISP pipeline. The default value is 0. 0: after static WB 1: immediately from sensor, channel 1 (for WDR modes) 2: after shading 3: after GammaFE 4: After DRC 5: immediately from sensor, channel 2 (for WDR modes) 6: after WDR stitch 7: after BLC, channel 2 (for WDR modes), only when GammaFE is after BLC of channel 1
enHistSwitch	Position of the statistics module for the global 256-segment histogram in the ISP pipeline. The default value is 0. 0: same as AE 1: immediately from sensor, channel 1 (for WDR modes) 2: after shading 3: after GammaFE 4: after DRC 5: immediately from sensor, channel 2 (for WDR modes) 6: after WDR stitch 7: after BLC, channel 2 (for WDR modes), only when GammaFE is after BLC of channel 1

Member	Description
enAESumSwitch	Positions of the global and zoned average value statistics modules in the ISP pipeline. The default value is 0. 0: after static WB 1: immediately from sensor, channel 1 (for WDR modes) 2: after shading 3: after GammaFE 4: After DRC 5: immediately from sensor, channel 2 (for WDR modes) 6: after WDR stitch 7: after BLC, channel 2 (for WDR modes), only when GammaFE is after BLC of channel 1

[Note]

- The 15 x 17 weight table configuration takes effect for the 256-segment histogram only when the member indicating the position of the global 256-segment histogram in the ISP pipeline is set to 0.
- When the position of the AE statistics module in the ISP pipeline is set to 3, the statistics module is always after GammaFE. Now GammaFE can be after BLC or WDR stitch, depending on the configured value of GammaFePosition.
- In linear mode or sensor built-in WDR mode, only the statistics of channel 1 is valid by default.
- The average value of the global four-component statistics is affected by the 15 x 17 weight table, and is irrelevant to the configured value of **enAESumSwitch**.

[See Also]

None

ISP_WB_STATISTICS_CFG_S

[Description]

Defines the AWB statistics configuration.

[Syntax]

```
typedef struct hiISP_WB_STATISTICS_CFG_S
{
    ISP_WB_STATISTICS_CFG_PARA_S stBayerCfg;
    ISP_WB_STATISTICS_CFG_PARA_S stRGBCfg;
} ISP_WB_STATISTICS_CFG_S;
```

[Member]

Member	Description
stBayerCfg	AWB statistics configuration of the Bayer field

Member	Description
stRGBCfg	AWB statistics configuration of the RGB field

[Note]

None

[See Also]

None

ISP_WB_STATISTICS_CFG_PARA_S

[Description]

Defines AWB statistics parameters.

[Syntax]

```
typedef struct hiISP_WB_STATIST
{
    HI_U16 u16WhiteLevel;
    HI_U16 u16BlackLevel;
    HI_U16 u16CbMax;
    HI_U16 u16CbMin;
    HI_U16 u16CrMax;
    HI_U16 u16CrMin;
    HI_U16 u16CbHigh;
    HI_U16 u16CbLow;
    HI_U16 u16CrHigh;
    HI_U16 u16CrLow;
} ISP_WB_STATISTICS_CFG_PARA_S;
```

[Member]

Member	Description
u16WhiteLevel	Upper luminance limit for searching for white points during white point statistics Value range: [0x0, 0xFFF] Default value: 0xFFF
u16BlackLevel	Lower luminance limit for searching for white points during white point statistics Value range: [0x0, 0xFFF] Default value: 0x0
u16CbMax	Maximum B/G color difference during white point statistics (8-bit precision) Default value: 512

Member	Description
u16CbMin	Minimum B/G color difference during white point statistics (8-bit precision) Default value: 128
u16CrMax	Maximum R/G color difference during white point statistics (8-bit precision) Default value: 512
u16CrMin	Minimum R/G color difference during white point statistics (8-bit precision) Default value: 128
u16CbHigh	Cb value (8-bit precision) corresponding to CrMax limited in the white point area in the hexagon during white point statistics Default value: 512
u16CbLow	Cb value (8-bit precision) corresponding to CrMin limited in the white point area in the hexagon during white point statistics Default value: 128
u16CrHigh	Cr value (8-bit precision) corresponding to CbMax limited in the white point area in the hexagon during white point statistics Default value: 512
u16CrLow	Cr value (8-bit precision) corresponding to CbMin limited in the white point area in the hexagon during white point statistics Default value: 128

For details, see Figure 2-9.

[Note]

None

[See Also]

None

ISP_FOCUS_STATISTICS_CFG_S

[Description]

Defines the AF statistics configuration.

[Syntax]

```
typedef struct hiISP_FOCUS_STATISTICS_CFG_S
{
    ISP_AF_CFG_S        stConfig;
    ISP_AF_H_PARAM_S    stHParam_IIR0;
    ISP_AF_H_PARAM_S    stHParam_IIR1;
    ISP_AF_V_PARAM_S    stVParam_FIR0;
```

```
    ISP_AF_V_PARAM_S    stVParam_FIR1;
    ISP_AF_FV_PARAM_S   stFVParam;
} ISP_FOCUS_STATISTICS_CFG_S;
```

[Member]

Member	Description
stConfig	AF global configuration parameters
stHParam_IIR0	Horizontal filter IIR parameters in the odd row
stHParam_IIR1	Horizontal filter IIR parameters in the even row
stVParam_FIR0	Vertical filter FIR parameters in the odd row
stVParam_FIR1	Vertical filter FIR parameters in the even row
stFVParam	Parameter configuration for statistics result output format

[Note]

None

[See Also]

None

ISP_AF_CFG_S

[Description]

Defines AF statistics configuration parameters.

[Syntax]

```
typedef struct hiISP_AF_CFG_S
{
    HI_BOOL           bEnable;
    HI_U16            u16Hwnd;
    HI_U16            u16Vwnd;
    HI_U16            u16Hsize;
    HI_U16            u16Vsize;
    ISP_AF_PEAK_MODE_E  enPeakMode;
    ISP_AF_SQU_MODE_E   enSquMode;
}
```

[Member]

Member	Description
bEnable	AF enable

Member	Description
u16Hwnd	Number of horizontal AF statistics blocks Value range: [1, 17]
u16Vwnd	Number of vertical AF statistics blocks Value range: [1, 15]
u16Hsize	Input image width for AF statistics Value range: [1, 0xFFF]
u16Vsize	Input image height for AF statistics Value range: [1, 0xFFF]
enPeakMode	Whether the peak value of the zoned IIR statistics (**u16h1** and **u16h2**) is calculated. **enPeakMode** and **enSquMode** affect the zoned statistics.
enSquMode	Whether the zoned horizontal statistics and vertical statistics are squared in square mode. **enSquMode** and **enPeakMode** affect the zoned statistics.

[Note]

- Calculation method of the zoned IIR statistics (**u16h1** and **u16h2**)
  - When **enPeakMode** is **ISP_AF_STA_NORM** and **enSquMode** is **ISP_AF_STA_SUM_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the IIR filter output values of each pixel in zones.
  - When **enPeakMode** is **ISP_AF_STA_NORM** and **enSquMode** is **ISP_AF_STA_SUM_SQU**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of squares of the IIR filter output value of each pixel in zones.
  - When **enPeakMode** is **ISP_AF_STA_PEAK** and **enSquMode** is **ISP_AF_STA_SUM_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the maximum IIR filter output value of the pixels in each row in zones.
  - When **enPeakMode** is **ISP_AF_STA_PEAK** and **enSquMode** is **ISP_AF_STA_SUM_SQU**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of squares of the maximum IIR filter output value of the pixels in each row in zones.
- Calculation method of the zoned FIR statistics (**u16v1** and **u16v2**)

  When **enSquMode** is **ISP_AF_STA_SUM_NORM**, the sum of the FIR filter output value of each pixel in zones is the zoned FIR statistics (**u16v1** and **u16v2**).

  When **enSquMode** is **ISP_AF_STA_SUM_SQU**, the sum of the squared FIR filter output value of each pixel in zones is the zoned FIR statistics (**u16v1** and **u16v2**).
- Calculation method of the zoned luminance statistics (**u16y**)

  The zoned luminance statistics (**u16y**) are the sum of the luminance of each pixel in zones.

[See Also]

None

ISP_AF_PEAK_MODE_E

[Description]

Defines whether the peak value of the zoned IIR statistics (**u16h1** and **u16h2**) is calculated.

[Syntax]

```
typedef enum hiISP_AF_PEAK_MODE_E
{
    ISP_AF_STA_NORM        = 0,
    ISP_AF_STA_PEAK          ,
    ISP_AF_STA_BUTT
}
```

[Member]

Member	Description
ISP_AF_STA_NORM	When **enSquMode** is **ISP_AF_STA_SUM_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the IIR filter output value of each pixel in zones. When **enSquMode** is **ISP_AF_STA_SUM_SQU**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the squares of the IIR filter output value of each pixel in zones.
ISP_AF_STA_PEAK	When **enSquMode** is **ISP_AF_STA_SUM_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the maximum IIR filter output value of the pixels in each row in zones. When **enSquMode** is **ISP_AF_STA_SUM_SQU**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the squares of the maximum IIR filter output value of the pixels in each row in zones.

[Note]

The statistics of the vertical FIR filter are not affected by ISP_AF_PEAK_MODE_E.

[See Also]

None

ISP_AF_SQU_MODE_E

[Description]

Defines whether the zoned statistics are squared in square mode.

[Syntax]

```
typedef enum hiISP_AF_SQU_MODE_E
{
    ISP_AF_STA_SUM_NORM      = 0,
    ISP_AF_STA_SUM_SQU         ,
    ISP_AF_STA_SUM_BUTT
}
```

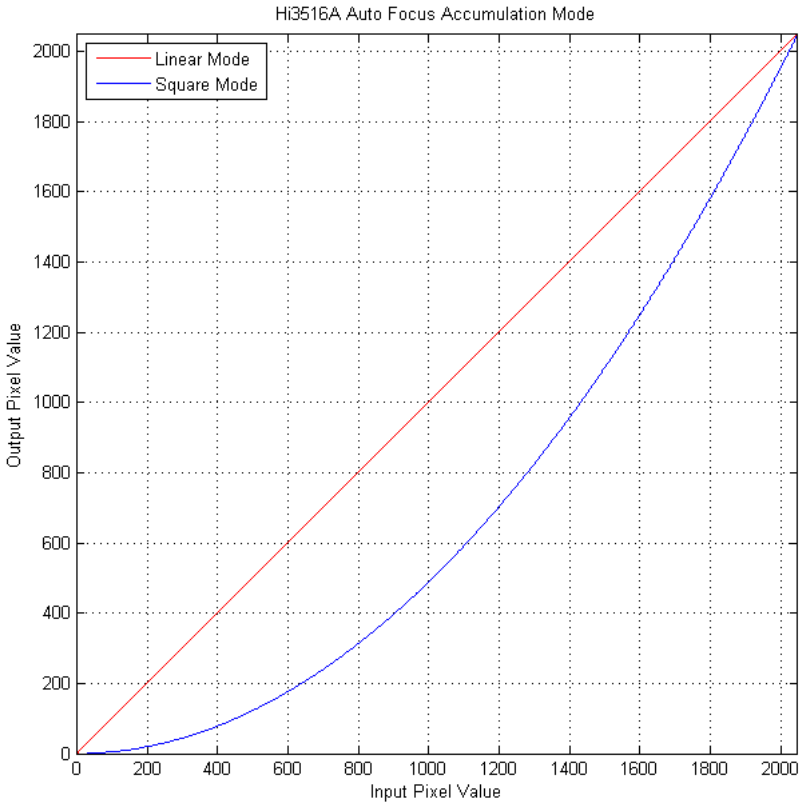
[Member]

Member	Description
ISP_AF_STA_SUM_NORM	Horizontal statistics When **enPeakMode** is **ISP_AF_STA_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the IIR filter output value of each pixel in zones. When **enPeakMode** is **ISP_AF_STA_PEAK**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the maximum IIR filter output value of the pixels in each row in zones. Vertical statistics The sum of the FIR filter output value of each pixel in zones is the zoned FIR statistics (**u16v1** and **u16v2**).
ISP_AF_STA_SUM_SQU	Horizontal statistics When **enPeakMode** is **ISP_AF_STA_NORM**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the squares of the IIR filter output value of each pixel in zones. When **enPeakMode** is **ISP_AF_STA_PEAK**, the zoned IIR statistics (**u16h1** and **u16h2**) are the sum of the squares of the maximum IIR filter output value of the pixels in each row in zones. Vertical statistics The sum of the squared FIR filter output value of each pixel in zones is the zoned FIR statistics (**u16v1** and **u16v2**).

[Note]

In square mode, the filter outputs are normalized, squared, and then collected for statistics. As shown in Figure 7-1, the input indicates the output pixel value after filtering, and the output indicates the output value after processing in square mode. Compared with the normal (linear) mode, the FV curve is steeper near the focus in square mode. In addition, noise signals with small amplitudes are suppressed, and flat regions on the FV curve are more flat. You need to set **enSquMode** based on the application scenario.

Figure 7-1 Square mode



[See Also]

None

ISP_AF_H_PARAM_S

[Description]

Defines the IIR parameter configuration of horizontal filters for AF statistics.

[Syntax]

```
typedef enum hiISP_AF_H_PARAM_S
{
    HI_BOOL     abIIREn[3];
    HI_S16      as16IIRGain[7];
    HI_U16      au16IIRShift[4];
    HI_U16      u16IIRThd;
}
```

[Member]

Member	Description
abIIREn[3]	Cascaded IIR enable
as16IIRGain[7]	IIR filter coefficient, for controlling the frequency response of the IIR filter
au16IIRShift[4]	IIR filter shift adjustment Value range: [0, 0xF]
u16IIRThd	IIR filter statistics threshold Value range: [0, 0x7FF] The value is involved in the FV value accumulation of the output statistics blocks only when the FV value of each pixel is above the threshold.

[Note]

as16IIRGain[0] is an 8-bit unsigned number. **as16IIRGain[1]** to **as16IIRGain[6]** are 10-bit signed numbers and cannot be −512. The IIR enable and filter coefficients and **IIRShift** can be generated by using the HiSilicon PQ Tools. You can adjust **u16IIRThd** to obtain the optimal FV statistics as required.

[See Also]

None

ISP_AF_V_PARAM_S

[Description]

Defines the FIR parameter configuration of vertical filters for AF statistics.

[Syntax]

```
typedef enum hiISP_AF_V_PARAM_S
{
    HI_S16 as16FIRH[5];
    HI_U16 u16FIRThd;
}
```

[Member]

Member	Description
as16FIRH[5]	FIR filter coefficient, for controlling the frequency response of the FIR filter Value range: [−31, +31]

Member	Description
u16FIRThd	FIR filter statistics threshold Value range: [0, 0x7FF] The value is involved in the FV value accumulation of the output statistics blocks only when the FV value of each pixel is above the threshold.

[Note]

The FIR filter coefficient can be generated by using the HiSilicon PQ Tools. You can adjust **u16FIRThd** to obtain the optimal vertical FV statistics as required.

[See Also]

None

ISP_AF_FV_PARAM_S

[Description]

Defines parameters for the AF statistics result output format.

[Syntax]

```
typedef enum hiISP_AF_H_PARAM_S
{
    HI_U16 u16AccShiftY;
    HI_U16 au16AccShiftH[2];
    HI_U16 au16AccShiftV[2];
}
```

[Member]

Member	Description
u16AccShiftY	Shift of luminance Y statistics Value range: [0, 0xF]
au16AccShiftH[2]	Shift of horizontal IIR filtering statistics Value range: [0, 0xF]
au16AccShiftV[2]	Shift of vertical FIR filtering statistics Value range: [0, 0xF]

[Note]

The FV value is a 16-bit unsigned number. When there are many details or noises in a scenario, the FV value may overflow. In this case, the output needs to be shifted rightwards to ensure that the output falls within the valid range. You can adjust the shift value as required.

[See Also]

None

ISP_STATISTICS_S

[Description]

Defines statistics.

[Syntax]

```
typedef struct hiISP_STATISTICS_S
{
    ISP_AE_STATISTICS_S        stAEStat;
    ISP_WB_STATISTICS_S        stWBStat;
    ISP_FOCUS_STATISTICS_S     stFocusStat;
} ISP_STATISTICS_S;
```

[Member]

Member	Description
stAEStat	AE statistics
stWBStat	AWB statistics
stFocusStat	AF statistics

[Note]

None

[See Also]

None

ISP_AE_STATISTICS_S

[Description]

Defines AE statistics.

[Syntax]

```
typedef struct hiISP_AE_STATISTICS_S
{
    HI_U16 au16Hist5Value[5];
    HI_U16 au16ZoneHist5Value[AE_ZONE_ROW][AE_ZONE_COLUMN][5];
    HI_U32 au32Hist256Value[256];
    HI_U16 au16GlobalAvg[4];
    HI_U16 au16ZoneAvg[AE_ZONE_ROW][AE_ZONE_COLUMN][4];
}ISP_AE_STATISTICS_S;
```

[Member]

Member	Description
au16Hist5Value	Global 5-segment histogram value
au16ZoneHist5Value	Zoned 5-segment histogram value
au32Hist256Value	Global 256-segment histogram value
au16GlobalAvg	Average values of global statistics, corresponding to the average values of the R, Gr, Gb, and B components respectively
au16ZoneAvg	Averages value of zoned statistics, corresponding to the average values of the R, Gr, Gb, and B components respectively

[Note]

None

[See Also]

None

ISP_WB_STATISTICS_S

[Description]

Defines AWB statistics.

[Syntax]

```
typedef struct hiISP_WB_STATISTICS_S
{
    ISP_WB_BAYER_STATISTICS_INFO_S stBayerStatistics;
    ISP_WB_RGB_STATISTICS_INFO_S   stRGBStatistics;
} ISP_WB_STATISTICS_S;
```

[Member]

Member	Description
stBayerStatistics	AWB statistics of the Bayer field
stRGBStatistics	AWB statistics of the RGB field

[Note]

None

[See Also]

None

ISP_WB_BAYER_STATISTICS_INFO_S

[Description]

Defines AWB statistics in the Bayer field.

[Syntax]

```
typedef struct hiISP_WB_BAYER_STATISTICS_S
{
    HI_U16 u16GlobalR;
    HI_U16 u16GlobalG;
    HI_U16 u16GlobalB;
    HI_U16 u16CountAll;
    HI_U16 u16CountMin;
    HI_U16 u16CountMax;
    HI_U16 au16ZoneAvgR[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneAvgG[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneAvgB[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneCountAll[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneCountMin[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneCountMax[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
} ISP_WB_BAYER_STATISTICS_INFO_S;
```

[Member]

Member	Description
u16GlobalR	Average value of R component global statistics for the Bayer field Value range: [0x0, 0xFFFF]
u16GlobalG	Average value of G component global statistics for the Bayer field Value range: [0x0, 0xFFFF]
u16GlobalB	Average value of B component global statistics for the Bayer field Value range: [0x0, 0xFFFF]
u16CountAll	Number of pixels in the gray region of the hexagon (obtained after global statistics), which has been normalized Value range: [0x0, 0xFFFF]
u16CountMin	Number of pixels whose values are less than **BlackLevel** (obtained after global statistics), which has been normalized Value range: [0x0, 0xFFFF]
u16CountMax	Number of pixels whose values are greater than **WhiteLevel** (obtained after global statistics), which has been normalized Value range: [0x0, 0xFFFF]
au16ZoneAvgR	Average value of R component zoned statistics for the Bayer field Value range: [0x0, 0xFFFF]
au16ZoneAvgG	Average value of G component zoned statistics for the Bayer field Value range: [0x0, 0xFFFF]

Member	Description
au16ZoneAvgB	Average value of B component zoned statistics for the Bayer field Value range: [0x0, 0xFFFF]
au16ZoneCountAll	Number of pixels in the gray region of the hexagon (obtained after zoned statistics), which has been normalized Value range: [0x0, 0xFFFF]
au16ZoneCountMin	Number of pixels whose values are less than **BlackLevel** (obtained after zoned statistics), which has been normalized Value range: [0x0, 0xFFFF]
au16ZoneCountMax	Number of pixels whose values are greater than **WhiteLevel** (obtained after zoned statistics), which has been normalized Value range: [0x0, 0xFFFF]

[Note]

None

[See Also]

None

ISP_WB_RGB_STATISTICS_INFO_S

[Description]

Defines AWB statistics in the RGB field.

[Syntax]

```
typedef struct hiISP_WB_RGB_STATISTICS_S
{
    HI_U16 u16GlobalGR;
    HI_U16 u16GlobalGB;
    HI_U32 u32GlobalSum;
    HI_U16 au16ZoneGR[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U16 au16ZoneGB[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
    HI_U32 au32ZoneSum[AWB_ZONE_ROW][AWB_ZONE_COLUMN];
} ISP_WB_RGB_STATISTICS_INFO_S;
```

[Member]

Member	Description
u16GlobalGR	G/R value of the RGB field obtained after global statistics
u16GlobalGB	G/B value of the RGB field obtained after global statistics
u32GlobalSum	Number of gray points of the RGB field obtained after global statistics

Member	Description
au16ZoneGR	G/R value of the RGB field obtained after zoned statistics
au16ZoneGB	G/B value of the RGB field obtained after zoned statistics
au32ZoneSum	Number of gray points of the RGB field obtained after zoned statistics

[Note]

None

[See Also]

None

ISP_FOCUS_STATISTICS_S

[Description]

Defines AF statistics.

[Syntax]

```
typedef struct hiISP_FOCUS_STATISTICS_S
{
    ISP_FOCUS_ZONE_S stZoneMetrics[AF_ZONE_ROW][AF_ZONE_COLUMN];
} ISP_FOCUS_STATISTICS_S;
```

[Member]

Member	Description
stZoneMetrics	Parameter of AF statistics

[Note]

None

[See Also]

ISP_FOCUS_ZONE_S

ISP_FOCUS_ZONE_S

[Description]

Defines AF statistics parameters.

[Syntax]

```
typedef struct hiISP_FOCUS_ZONE_S
{
    HI_U16  u16v1;
    HI_U16  u16h1;
```

```
    HI_U16  u16v2;
    HI_U16  u16h2;
    HI_U16  u16y;
} ISP_FOCUS_ZONE_S;
```

[Member]

Member	Description
u16v1	Statistics of the FIR filter for the AF odd columns in zoned statistics
u16h1	Statistics of the IIR filter for the AF odd rows in zoned statistics
u16v2	Statistics of the FIR filter for the AF even columns in zoned statistics
u16h2	Statistics of the IIR filter for the AF even rows in zoned statistics
u16y	AF luminance statistics in zoned statistics (accumulated luminance of the pixels in zones)

[Note]

u16y is the accumulated value of the Y components of each block. It can be used to work with the AF algorithm. For example, when some blocks face highlight illuminants, the FV value is easily affected. In this case, the impact on the global FV exerted by the block FV can be alleviated based on the luminance. In addition, these blocks can be used to implement a motion detection algorithm to detect the motion interference during focusing. You can use the preceding statistics based on the requirements of the AF algorithm.

[See Also]

None

8 Default Cmos Parameter Configuration

8.1 Overview

Each sensor corresponds to an **xxxx_cmos.c** file, which consists of the system control part (see chapter 2 "System Control") and the default parameter configuration part. The default parameters can be configured by parsing the **xxxx_cfg.ini** file or be initialized directly by functions. The configuration mode is specified by modifying the Makefile (such as the **ISP_INI_CONFIG** variable in the **Makefile.param** file in the **mpp** folder of the SDK). When the default parameters are configured by parsing the **xxxx_cfg.ini** file, the path of the .ini file can be configured by calling sensor_set_inifile_path (const char *pcPath).

The .ini file is parsed as follows:

Step 1 Set **ISP_INI_CONFIG** to **y** in **Makefile.param** in the **mpp** directory.

Step 2 Compile the sensor library in the **ISP** directory.

Step 3 Copy the compiled sensor library and library files related to the .ini file from the **ISP** directory to a specified directory.

Step 4 Call sensor_set_inifile_path(const char *pcPath) in the service program to configure the .ini file path (see the sample).

Step 5 Add the sensor and library files related to the .ini file (**lib_iniparser** and **lib_cmoscfg**) to the **makefile** file of the service program and compile the files (see the sample).

----End

8.2 Cmos Structure

Figure 8-1 Cmos structure

8.3 INI File Usage Description

Each sensor corresponds to an **xxxx_cfg.ini** file such as **panasonic_mn34220/mn34220_cfg.ini**. For details, see the .ini files in the **sensor** folder.

The **mn34220_cfg.ini** file contains three fields ([AE], [AWB], and [ISP]). Each field contains the number of parameter groups and the parameter list in each group.

8.3.1 AE

Table 8-1 describes parameters in the [AE] field in the .ini file.

Table 8-1 Parameters in the [AE] field

Parameter	Description
AEModeNumber	Number of AE parameter groups
AeCompensation_0	Target AE luminance
MaxIntTimeTarget_0	Maximum target exposure time
MinIntTimeTarget_0	Minimum target exposure time

Parameter	Description
MaxAgainTarget_0	Maximum target sensor analog gain
MinAgainTarget_0	Minimum target sensor analog gain
MaxDgainTarget_0	Maximum target sensor digital gain
MinDgainTarget_0	Minimum target sensor digital gain
ISPDgainShift_0	Precision of the ISP digital gain
MinISPDgainTarget_0	Minimum target ISP digital gain
MaxISPDgainTarget_0	Maximum target ISP digital gain

Note that parameters with the suffix _0 indicate parameters in parameter group 1. The meaning of the suffix applies to parameters in the [AWB] and [ISP] fields.

8.3.2 AWB

Table 8-2 describes parameters in the [AWB] field in the .ini file.

Table 8-2 Parameters in the [AWB] field

Parameter	Description
AWBModeNumber	Number of AWB parameter groups
HighColorTemp_0	High color temperature
HighCCM_0	High color temperature CCM
MidColorTemp_0	Medium color temperature
MidCCM_0	Medium color temperature CCM
LowColorTemp_0	Low color temperature
LowCCM_0	Low color temperature CCM
WbRefTemp_0	Corrected color temperature for static white balance
GainOffset_0	Gains of the R, Gr, Gb, and B channels for the static white balance
WbPara_0	White balance parameters provided by the correction tool
SatValid_0	Sat data validity enable
Saturation_0	Interpolation array for dynamically adjusting the saturation based on the gain

8.3.3 ISP

Table 8-3 describes parameters in the [ISP] field in the .ini file.

Table 8-3 Parameters in the [ISP] field

Parameter	Description
ISPModeNumber	Number of ISP parameter mode groups
AgcValid_0	Automatic gain control (AGC) data validity enable
SharpenAltD_0	Interpolation array for dynamically adjusting the sharpness of the large edges of images based on the gain
SharpenAltUd_0	Interpolation array for dynamically adjusting the sharpness of the small textures of images based on the gain
SnrThresh_0	Interpolation array for dynamically setting the image denoising strength based on the gain
DemosaicLumThresh_0	Array for setting the luminance threshold for the sharpness of large edges of images
DemosaicNpOffset_0	Array for setting image noise parameters
GeStrength_0	Array for setting the parameters of the green equalization parameter
SharpenRGB_0	Interpolation array for dynamically adjusting the overall image sharpness based on the gain
WeightValid_0	Weight data validity enable
NoiseProfileWeight_0	Array for setting the noise profile related to sensor features. The array value is used as the input of the denoising module.
DemosaicWeight_0	Array for setting the noise profile related to sensor features. The array value is used as the input of the demosaic module.
demosaicValid_0	Demosaic data validity enable
VhSlope_0	Vertical/Horizontal slope threshold
AaSlope_0	Angle slope threshold
VaSlope_0	VH-AA slope threshold
UuSlope_0	Slope threshold in all directions
SatSlope_0	Saturation slope threshold
AcSlope_0	High-frequency component filtering slope threshold
FcSlope_0	Anti-false color slope threshold
VhThresh_0	Vertical/Horizontal threshold
AaThresh_0	Angle threshold
VaThresh_0	VH-AA threshold
UuThresh_0	Threshold in all directions
SatThresh_0	Saturation threshold

Parameter	Description
AcThresh_0	High-frequency component filtering threshold
gammaRGBValid_0	Gamma data validity enable
gammaRGB0_0	Segment 1 of the RGBGamma table
gammaRGB1_0	Segment 2 of the RGBGamma table
gammaRGB2_0	Segment 3 of the RGBGamma table
gammafevalid_0	GammaFe data validity enable
gammafe0_0	GammaFe0 table
gammafe1.0_0	Segment 1 of the GammaFe1 table
gammafe1.1_0	Segment 2 of the GammaFe1 table
gammafe1.2_0	Segment 3 of the GammaFe1 table
RGBsharpenLutValid_0	RGBsharpen curve validity enable
LutCore_0	Core parameter of the SharpenRGB curve
LutStrength_0	Strength parameter of the SharpenRGB curve
LutMagnitude_0	Magnitude parameter of the SharpenRGB curve
DrcEnable_0	DRC data validity enable
BlackLevel_0	The default value is recommended. You are advised not to change the default value.
WhiteLevel_0	The default value is recommended. You are advised not to change the default value.
SlopeMax_0	The default value is recommended. You are advised not to change the default value.
SlopeMin_0	The default value is recommended. You are advised not to change the default value.
VarianceSpace_0	The default value is recommended. You are advised not to change the default value.
VarianceIntensity_0	The default value is recommended. You are advised not to change the default value.
ShadingValid_0	Shading data validity enable
RCenterX_0	Horizontal coordinate of the R component center
RCenterY_0	Vertical coordinate of the R component center
GCenterX_0	Horizontal coordinate of the G component center
GCenterY_0	Vertical coordinate of the G component center
BCenterX_0	Horizontal coordinate of the B component center

Parameter	Description
BCenterY_0	Vertical coordinate of the B component center
RShadingTbl_0	Correction table of the R component
GShadingTbl_0	Correction table of the G component
BShadingTbl_0	Correction table of the B component
ROffCenter_0	Distance between the R component center and the farthest angle
GOffCenter_0	Distance between the G component center and the farthest angle
BOffCenter_0	Distance between the B component center and the farthest angle
TblNodeNum_0	Number of used nodes in each component correction table

8.4 Precautions

The preceding **mn34220_cfg.ini** file contains a group of ISP cmos parameters. You can crop each subgroup (separated by a blank row in the preceding tables) in each parameter mode group as required. You can delete the following two items as required:

SatValid_0 = 1

Saturation_0 = xxxxxxx

Only parameters with the "Valid" character can be deleted as required. Other parameters are mandatory. The number of parameter groups is configured according to the number of parameters groups in each field. If the number is set to 3, three groups of parameter configuration are supported. For details about the format, see **mn34220_cfg.ini**. The [AE/AWB/ISP] field supports at most six groups of parameters. The number of parameter groups for each field can be increased or decreased within the range. Note that the suffix is required in the parameter name, for example, xxxx_0 to xxxx_5.

The parameter name in each parameter mode group must strictly correspond to the parameter name in the file such as **mn34220_cfg.ini**. The parameter name cannot be changed arbitrarily. Otherwise, the parameter configuration may fail. If the parameter name needs to be changed, the corresponding name in **hi_cmoscfg/hi_cmos_cfg.c** must be changed synchronously.

For each parameter group such as Saturation_0 = 0x80|0x80|0x80|0x78|0x70|0x68|0x58|0x48|0x40|0x38|0x38|0x38|0x38|0x38|0x38|0x38|, two parameter values can be separated by using a vertical bar (|) or a comma. The parameter value can be in hexadecimal format or decimal format.

The format of the gamma table needs to be noted in the .ini configuration file. It is recommended that the gamma table be divided into three segments and configured separately so that it can be parsed correctly. For details, see **mn34220_cfg.ini**.

Note the following when using the .ini configuration file:

- Avoid identical parameter configurations. Otherwise, a parsing error may occur.

- When writing the field name of "[AE/AWB/ISP]", ensure that "[" and "]" are in the same line.
- Only the space is valid before the field name of "[AE/AWB/ISP]". If there are other characters before the field name, the field name and its sub-items cannot be identified.
- The parameter name and the parameter value must be in the same line. Otherwise, the parameter cannot be identified.
- Do not use special characters such as "\r", "\n", "[',']", ";", and ", because the combination of these special characters may result in unpredictable exceptions.
- You are advised not to use ';' in the .ini file, because ';' is the comment sign. All the characters after ';' are regarded as comments.

9 Debug

9.1 Overview

Two MPIs are provided for debugging some modules in the ISP, helping customers to locate image quality issues.

9.2 Function Description

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.

9.3 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_DEV IspDev, const ISP_DEBUG_INFO_S
*pstIspDebug)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspDebug	Debugging information	Input

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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 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 = 0;
  HI_U32  au32Depth   = 10;
  HI_U32  au32Size    = 0;
  ISP_DEBUG_INFO_S  stDebug;
  HI_VOID *apVitAddr   = HI_NULL;    /* virt addr malloc memory */
  HI_VOID *apVirtAddr  = HI_NULL;    /* virt addr mmap */
  au32Size = sizeof(ISP_DBG_ATTR_S) + sizeof(ISP_DBG_STATUS_S) * au32Depth;
  s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&au32PhyAddr, &apVitAddr, HI_NULL,
HI_NULL, au32Size);
        if (HI_SUCCESS != s32Ret)
        {
            printf("Buf not enough!\n");
            return HI_FAILURE;
        }
        stDebug.bDebugEn = HI_TRUE;
        stDebug.u32Depth = au32Depth;
        stDebug.u32PhyAddr = au32PhyAddr;
        s32Ret = HI_MPI_ISP_SetDebug(0, &stDebug);
        if (HI_SUCCESS != s32Ret)
```

```
{
    printf("HI_MPI_ISP_SetDebug failed 0x%x!\n", s32Ret);
    return HI_FAILURE;
}
```

[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_DEV IspDev, ISP_DEBUG_INFO_S*pstIspDebug)
```

[Parameter]

Parameter	Description	Input/Output
IspDev	ISP device ID	Input
pstIspDebug	Debugging information	Output

[Return Value]

Return Value	Description
0	Success
Other values	Failure. Its value is an error code. For details, see chapter 10 "Error Codes."

[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

9.4 Data Structure

ISP_DEBUG_INFO_S

[Description]

Defines ISP debugging information attributes.

[Syntax]

```
typedef struct hiISP_DEBUG_INFO_S
{
    HI_BOOL bDebugEn;
    HI_U32  u32PhyAddr;
    HI_U32  u32Depth;
} ISP_DEBUG_INFO_S;
```

[Member]

Member	Description
bDebugEn	Debugging enable
u32PhyAddr	Physical address for debugging information
u32Depth	Debugging depth (number of frames for obtaining debugging information)

[Note]

None

[See Also]

None

10 Error Codes

Table 10-1 describes the error codes for ISP APIs.

Table 10-1 Error codes for ISP APIs

Error Code	Macro Definition	Description
0xA01C8006	HI_ERR_ISP_NULL_PTR	The input pointer is null.
0xA01C8003	HI_ERR_ISP_ILLEGAL_PARAM	The input parameter is invalid.
0xA01C8008	HI_ERR_ISP_NOT_SUPPORT	This function is not supported by the ISP.
0xA01C8043	HI_ERR_ISP_SNS_UNREGISTER	The sensor is not registered.
0xA01C8041	HI_ERR_ISP_MEM_NOT_INIT	The external registers are not initialized.
0xA01C8040	HI_ERR_ISP_NOT_INIT	The ISP is not initialized.
0xA01C8044	HI_ERR_ISP_INVALID_ADDR	The address is invalid.
0xA01C8042	HI_ERR_ISP_ATTR_NOT_CFG	The attribute is not configured.
0xA01C8045	HI_ERR_ISP_NOMEM	The memory is insufficient.
0xA01C8046	HI_ERR_ISP_NO_INT	The ISP module has no interrupt.

11 Proc Debugging Information

11.1 Overview

The debugging information uses the proc file system. 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]

- Linux

  When loading .ko files of the ISP, add the module parameter **proc_param=n**. If **n** is **0**, the proc information is disabled and the memory for storing the proc information is not allocated. If **n** is not **0**, the proc information is updated every n frames and the memory for storing the proc information is allocated. For example, "**insmod hi3516A_isp.ko proc_param=30**" indicates that the proc information is updated every 30 frames.

- Huawei LiteOS

  In the ISP_init function in **sdk_init.c** under **release/ko**, **stIsp_Param.u32ProcParam= n** is added to ISP_ModInit. If **n** is **0**, the proc is disabled and the memory for storing the proc information is not allocated. If **n** is not **0**, the proc information is updated every *n* frames and the memory for storing the proc information is allocated. For example, if **u32ProcParam** is **30**, the proc information is updated every 30 frames.

NOTE

The CPU resource is consumed when proc information is enabled. You are advised to update the proc information once every 30 frames or enable proc information only during debugging.

[Method of viewing information]

- You can run a **cat** command to view information on the console, for example, **cat /proc/umap/isp**. You can also use other common file operation commands to copy the **proc** file of the ISP to the current directory, for example, **cp /proc/umap/isp ./ -rf**. Huawei LiteOS only supports viewing information by running a **cat** command and does not support using other common file operation commands to copy files.
- 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.

11.2 ISP

[Debugging information]

```
# cat /proc/umap/isp
[ISP] Version: [Hi3516A_ISP_V1.0.0.0  Debug], Build Time[Jul  3 2014,
15:15:19]
-----MODULE PARAM----------------------------------------------------
proc_param    int_bottomhalf
30            0

-----DRV INFO--------------------------------------------------------------
IspDev     IntCnt       IntT     MaxIntT     IntGapT     MaxGapT      IntRat
IspResetCnt
0     1629244       9271       9416      39973  2048461462        26
2

PtIntCnt      PtIntT     PtMaxIntT   PtIntGapT   PtMaxGapT    PtIntRat
SensorCfgT   SensorMaxT
1629044         22         27        39973      2048578125         25
166        166
-------------------PubAttr INFO----------------------------------------
      WndX       WndY      WndW       WndH      Bayer
       0          0        1920       1080       GRBG
[AE] Version: [Hi3516A_ISP_V1.0.0.0 B010 Debug], Build Time[Feb 23 2016,
15:38:34]
------------------AE INFO-------------------------------------------------
Again   Dgain   IspDg SysGain     Iso    Line AEInter Incrmnt        Exp
1stTime
7168    1024    1028    7196     702    1550      1     257    11141563
150000
Comp   EVbias  OriAve  Offset   Speed    Tole   Error     Fps RealFpsBDelay
WDelay
56     1024     51      5      64       6      0     25.00     2500      8
0
MaxLineMaxLineT  MaxAgT  MaxDgT MaxIDgT  MaxSgT Thresh0 Thresh1 Thresh2
Thresh3
1348    1348   32382   32382   1638016384280     13     40      96     128
ManuEn  MaLine    MaAg    MaDg MaIspDg   WdrMode ExpRatio AnFlick SlowMod
```

```
GainTh
0       0       0       0       0      LINE          64      0       1  16380280

NodeId IntTime SysGain IrisApeUpStgy  DwStgy         Mltply
 0       2      64       1       0       4            128
 1    1348      64       1       1       0          86272
 2    1348 1023767       1       4       1     1380037916
AuIrEn  IrType  MaIrEn DbgIrSt
      1  DCIris       0       0
[AWB] Version: [Hi3516A_ISP_V1.0.0.0 B010 Debug], Build Time[Feb 23 2016,
15:38:34]
-----AWB INFO--------------------------------------------------------------
Gain0   Gain1   Gain2   Gain3  CoTemp
0x1b0   0x110   0x110   0x25b    3745
Color00 Color01 Color02 Color10 Color11 Color12 Color20 Color21 Color22
0x 19f  0x8059  0x8046  0x806e  0x 1c7  0x8059  0x  11  0x80cf  0x 1be

ManuEn     Sat   Zones   Speed
      0     116      32     128
-----DRC INFO--------------------------------------------------------------
     En  ManuEn   DrcSt DrcStTg
      0       0       0     128
-----DEMOSAIC INFO---------------------------------------------------------
VhSlope    VhTh AaSlope    AaTh VaSlope    VaTh UuSlope    UuTh
0xdc   0x  0    0xc8   0x  0    0xb9   0x  0    0xa8   0x  8

SatSlope   SatTh AcSlope    AcTh FcSlope
   0x5d   0x  0    0xa0   0x1b3    0x90
-----NR INFO---------------------------------------------------------------
     En  ManuEn ThreshS ThreshL  Offset
      1       0      11      11       0
-----SHARPEN INFO----------------------------------------------------------
     En  ManuEn     SpD    SpUd   SpRGB   LumTh    GeSt
      1       0      56      96      96      64      85
-----FPN INFO--------------------------------------------------------------
     En OpType Strength Offset
      0     150x      00x     0
-----ACM INFO--------------------------------------------------------------
En   DemoEn   Mode  Stretch    Clip    Wrap CbcrThr GainLuma GainHue GainSta
0       0       4         1       1       0       0           0       0       0

-----SHADING INFO----------------------------------------------------------
En
0
```

[Debugging information analysis]

Records the usage of the current ISP module.

[Parameter description]

Parameter		Description
MODULE PARAMque	proc_param	Update interval for the proc information. When this value is 0, the proc is disabled and the memory for storing the proc information is not allocated. When this value is a non-zero value, the proc information is updated every *n* frames and the memory for storing the proc information is allocated.
	int_bottomhalf	ISP interrupt bottom half enable 0: The operations, for example, obtaining the 3A statistics, are stored in the ISR of hardware interrupts. 1: The operations, for example, obtaining the 3A statistics, are stored in the bottom half of the interrupt. In this case, the time for the hardware interrupt processing is short, but the 3A statistics may not be obtained in a timely manner. Huawei LiteOS does not support the bottom half function of interrupts. When this parameter is enabled in Huawei LiteOS, the message "Do not support in Lite OS" is displayed.
DRV INFO	IspDev	ISP device ID
	IntCnt	Number of ISP channel interrupts
	IntT	Time used to process ISP interrupts
	MaxIntT	Maximum time used to process ISP interrupts
	IntGapT	Interval between two adjacent ISP interrupts
	MaxGapT	Maximum interval between two adjacent ISP interrupts
	IntRat	Number of ISP interrupts per second
	IspResetCnt	Number of times of ISP reset
	PtIntCnt	Number of interrupts for the port channel
	PtIntT	Time used to process port interrupts
	PtMaxIntT	Maximum time used to process port interrupts
	PtIntGapT	Interval between two adjacent port interrupts
	PtMaxGapT	Maximum interval between two adjacent port interrupts
	PtIntRat	Number of port interrupts per second
	SensorCfgT	Sensor configuration time
	SensorMaxT	Maximum sensor configuration time
PubAttr	WndX	Horizontal start position

Parameter		Description
INFO	WndY	Vertical start position
	WndW	Picture width
	WndH	Picture height
	Bayer	Format of the input Bayer image
AE INFO	Again	Analog gain of the sensor (10-bit precision)
	Dgain	Digital gain of the sensor (10-bit precision)
	IspDg	Digital gain of the ISP (10-bit precision)
	SysGain	System gain (10-bit precision)
	ISO	The ISO indicates the system gain and is multiplied by 100. For example, if the system sensor gain is 2x and the ISP gain is 1x, the ISO of the entire system is 200 (2 x 1 x 100). This calculation method applies to all the ISO in this document.
	Line	Exposure duration of the sensor, which is calculated by rows
	AEInter	Running interval of the AE algorithm. The value 2 indicates that the AE algorithm runs every two frames.
	Incrmnt	Increment of the current exposure value relative to the exposure value of the previous frame (8-bit precision)
	Exp	Current exposure value, which is the product of the shutter, gain, and iris values
	1stTime	First time when AE is stable (in μs)
	Comp	Exposure compensation during automatic exposure adjustment
	EVbias	Exposure bias during AE adjustment
	OriAve	Average luminance of the image calculated according to the 256-segment histogram
	Offset	Average luminance offset based on AE statistics
	Speed	Initial step during AE adjustment
	Tole	Exposure tolerance during automatic exposure adjustment
	Error	Exposure error during automatic exposure adjustment
	Fps	Reference frame rate
	RealFps	Actual picture frame rate
	BDelay	Waiting time for AE adjustment when the image luminance is lower than the target luminance (unit: frame)

Parameter		Description
	WDelay	Waiting time for AE adjustment when the image luminance is higher than the target luminance (unit: frame)
	MaxLine	Maximum exposure duration
	MaxLineT	Configured maximum exposure time
	MaxAgT	Configured maximum sensor analog gain
	MaxDgT	Configured maximum sensor digital gain
	MaxIDgT	Configured maximum ISP digital gain
	MaxSgT	Configured maximum system gain
	Thresh0	Segment point of the first and second segments for the 5-segment histogram during AE adjustment
	Thresh1	Segment point of the second and third segments for the 5-segment histogram during AE adjustment
	Thresh2	Segment point of the third and fourth segments for the 5-segment histogram during AE adjustment
	Thresh3	Segment point of the fourth and fifth segments for the 5-segment histogram during AE 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
	MaIspDg	ISP digital gain in ME mode
	WdrMode	WDR mode
	ExpRatio	Ratio of the long-frame exposure time to the short-frame exposure time, 6-bit precision. In long frame mode, this parameter can be ignored.
	AnFlick	Automatic anti-twinkle switch
	SlowMod	Automatic frame rate reduction mode
	GainTh	System gain in automatic frame reduction mode
	NodeId	Node ID
	IntTime	Node exposure time (in μs)
	SysGain	Node gain, including the sensor analog gain, sensor digital gain, and ISP gain
	IrisApe	Node iris size. Only the P iris is supported.
	UpStgy	Allocation policy for the AE uplink routes

Parameter		Description
	DwStgy	Allocation policy for the AE downlink routes
	Mltply	Product of the node shutter, gain, and iris
	AuIrisEn	AI enable
	IrType	Iris type (DC iris or P iris)
	MaIrisEn	Manual iris enable
	DbgIrSt	Iris debugging status. The value 0 indicates that the current status of the iris is retained. The value 1 indicates that the iris is opened. The value 2 indicates that the iris is closed.
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
	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
	Sat	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

Parameter		Description
DEMOSAIC 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
	FcSlope	Slope of false color threshold
NR INFO	En	Spatial noise reduction switch
	ManuEn	Manual spatial noise reduction switch
	ThreshS	Target threshold for WDR short exposure denoising
	ThreshL	Target threshold for WDR long exposure denoising
	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
	SpRGB	Sharpness of RGB sharpen
	LumTh	Luma thresh
	GeSt	GE strength
FPN INFO	En	FPN correction enable
	OpType	Type select (manual mode or automatic mode)
	Strength	Manual FPN correction strength

Parameter		Description
	Offset	Average pixel value for black frames
ACM INFO	En	ACM switch
	DemoEn	Demonstration mode enable (ACM is performed only on the right part)
	Mode	ACM coefficient mode select
	Stretch	Input data range
	Clip	Output data range
	Wrap	Decimal processing mode
	CbcrThr	Chrominance adjustment threshold
	GainLuma	Luminance gain
	GainHue	Hue gain
	GainSta	Saturation gain
SHADING INFO	En	Shading switch

A Acronyms and Abbreviations

Acronym or Abbreviation	Full Name
3A	AE, AWB and AF
ACM	Auto Color Management
AE	Auto Exposure
AF	Auto Focus
AG	Analog Gain
AGC	Auto Gain Control
API	Application Programming Interface
AWB	Auto White Balance Correction
CCM	Color Correction Matrix
CPU	Central Processing Unit
DG	Digital Gain
DIS	Digital Image Stabilization
DPC	Defect Pixel Correction
DRC	Dynamic Range Compression
FPN	Fixed Pattern Noise
FW	Firmware
IMP	Image Processing
ISO	ISO standard 12232:2006. A measure of photographic sensitivity to light
ISP	Image Signal Processor
LUT	Lookup Table
NP	Noise Profile

Acronym or Abbreviation	Full Name
NR	Noise Reduction
WB	White Balance
WDR	Wide Dynamic Range
YCbCr	Method of encoding RGB color space according to the ITU-R BT.601 or ITU- R BT 709 standards
YUV	A method of encoding RGB color space in practice equivalent to YCbCr