



HiISP

## 开发参考

文档版本 02

发布日期 2017-06-12

**版权所有 © 深圳市海思半导体有限公司 2015-2017。保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## **商标声明**



**HISILICON**、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## **注意**

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

## **深圳市海思半导体有限公司**

地址：                    深圳市龙岗区坂田华为基地华为电气生产中心                    邮编：518129

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

客户服务电话：          +86-755-28788858

客户服务传真：          +86-755-28357515

客户服务邮箱：          [support@hisilicon.com](mailto:support@hisilicon.com)



# 前 言

## 概述

本文为使用 HiSP 开发的程序员而写，目的是为您在开发过程中遇到的问题提供解决办法和帮助。



说明

本文以 Hi3518EV200 描述为例，本文未做特殊说明，Hi3518EV201、Hi3516CV200 与 Hi3518EV200 完全一致。

## 产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200

## 读者对象






本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

## 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。



符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

## 修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

## 文档版本 02 (2017-06-12)

### 第 2 章 系统控制

2.3 小节，新增 ISP\_MODULE\_PARAM\_S 和 ISP\_OP\_TYPE\_E

### 第 3 章 AE

3.5 小节涉及修改

### 第 4 章 AWB

4.5.2 小节，ISP\_AWB\_ATTR\_S【注意事项】涉及修改

### 第 6 章 IMP

6.8.3 小节，ISP\_UVNR\_ATTR\_S 的【成员】涉及修改

6.3.3、6.5.3 和 6.14.2 小节涉及修改

### 第 7 章 RGBIR

7.3 小节涉及修改

### 第 12 章 Proc 调试信息说明

12.2 小节，添加 MODULE\_PARAM 中的参数

## 文档版本 01 (2016-10-28)

### 第 2 章 系统控制



2.2 小节, HI\_MPI\_ISP\_GetVDTimeOut 的【错误码】涉及修改, 新增 HI\_MPI\_ISP\_SetDCFInfo 和 HI\_MPI\_ISP\_GetDCFInfo

2.3 小节, 新增 ISP\_DCF\_INFO\_S

### 第 3 章 AE

3.5.1 AE\_SENSOR\_EXP\_FUNC\_S、AE\_SENSOR\_DEFAULT\_S 有修改, 新增 AE\_FSWDR\_ATTR\_S

3.5.2 ISP\_AE\_ROUTE\_NODE\_S、ISP\_AE\_ROUTE\_EX\_NODE\_S 有修改, 新增 ISP\_FSWDR\_MODE\_E

3.5.3 ISP\_PIRIS\_ATTR\_S 涉及修改

### 第 4 章 AWB

4.5.2 ISP\_AWB\_ATTR\_EX\_S、ISP\_AWB\_IN\_OUT\_ATTR\_S 涉及修改

6.3.3 ISP\_DRC\_ATTR\_S【成员】和【注意事项】涉及修改; ISP\_DRC\_AUTO\_ATTR\_S【成员】有修改

### 第 6 章 IMP

6.14.2 小节, 修改 HI\_MPI\_ISP\_FPNCalibrate 的【注意】

### 第 11 章 错误码

表 11-1 涉及修改

### 第 12 章 Proc 调试信息说明

12.2 小节涉及修改

## 文档版本 00B06 (2016-05-20)

2.2 小节, 新增 HI\_MPI\_ISP\_SetModParam 和 HI\_MPI\_ISP\_GetModParam

2.3 小节, ISP\_AE\_RESULT\_S、ISP\_AWB\_RESULT\_S 涉及修改, 新增 ISP\_MOD\_PARAM\_S

3.5.2 小节, ISP\_AE\_ATTR\_S、ISP\_EXP\_INFO\_S、ISP\_AE\_ROUTE\_NODE\_S、ISP\_AE\_ROUTE\_EX\_NODE\_S 和 ISP\_PIRIS\_ATTR\_S 涉及修改

3.5.3 小节, ISP\_IRIS\_ATTR\_S 涉及修改

4.4.3 小节, 新增 HI\_MPI\_ISP\_CalGainByTemp

4.5.2 小节, ISP\_AWB\_LUM\_HISTGRAM\_ATTR\_S 和 ISP\_WB\_INFO\_S 涉及修改, 新增 ISP\_AWB\_MULTI\_LS\_TYPE\_E, 新增图 4-4; 表 4-1 到表 4-4 都涉及修改

12.2 小节涉及修改

## 文档版本 00B05 (2016-03-14)

添加 Huawei LiteOS 的相关内容



## 文档版本 00B04 (2015-12-18)

2.3 节，修改 WDR\_MODE\_E 中的【注意事项】

修改 ISP\_CMOS\_DEFAULT\_S 和 ISP\_AWB\_RESULT\_S 中的【成员】

3.5.1 修改 AE\_SENSOR\_DEFAULT\_S 的【定义】、【成员】和【注意事项】

3.5.2 修改 ISP\_AE\_ATTR\_S【成员】和【注意事项】

4.5.1 修改 AWB\_SENSOR\_DEFAULT\_S 的【成员】

5.5 ISP\_SATURATION\_MANUAL\_S 和 ISP\_SATURATION\_AUTO\_S 的【成员】有修改

6.3.3 修改 ISP\_DRC\_MANUAL\_ATTR\_S、ISP\_DRC\_AUTO\_ATTR\_S 和  
ISP\_DRC\_ATTR\_S【定义】和【成员】

6.4.3 修改 ISP\_SHADING\_ATTR\_S 的【定义】、【成员】和【注意事项】

新增 6.19 小节

6.5.3 修改 ISP\_DP\_DYNAMIC\_AUTO\_ATTR\_S 中的【成员】

12.2 小节有修改

## 文档版本 00B03 (2015-09-20)

2.2 HI\_MPI\_ISP\_SetModuleControl 和 ISP\_AWB\_EXP\_FUNC\_S 中的【注意】有修改

2.3 ISP\_AWB\_INFO\_S 和 ISP\_AWB\_RESULT\_S 中的【成员】和【注意事项】有修改

3.4.2 新增 HI\_MPI\_ISP\_SetAERouteAttrEx 和 HI\_MPI\_ISP\_GetAERouteAttrEx

3.4.3 新增 HI\_MPI\_ISP\_SetDcirisAttr, HI\_MPI\_ISP\_GetDcirisAttr,  
HI\_MPI\_ISP\_SetPirisAttr 和 HI\_MPI\_ISP\_GetPirisAttr

3.5.2 新增 ISP\_AE\_ROUTE\_EX\_NODE\_S 和 ISP\_AE\_ROUTE\_EX\_S

3.5.3 添加 ISP\_DCIRIS\_ATTR\_S 和 ISP\_PIRIS\_ATTR\_S

4.5.2 ISP\_AWB\_ATTR\_S、ISP\_AWB\_CT\_LIMIT\_ATTR\_S 和 ISP\_AWB\_ATTR\_EX\_S 有  
修改；新增 ISP\_AWB\_CBCR\_TRACK\_ATTR\_S 和  
ISP\_AWB\_LUM\_HISTGRAM\_ATTR\_S

5.5 ISP\_COLORMATRIX\_AUTO\_S 和 ISP\_COLORMATRIX\_MANUAL\_S 有修改

6.3.3 ISP\_DRC\_ATTR\_S 涉及更改

6.14.2 HI\_MPI\_ISP\_FPNCalibrate 中的【注意】有修改。

12.2 小节涉及修改

## 文档版本 00B02 (2015-08-20)

第二次临时版本发布。

6.17.1 小节有修改；6.17.3 小节，ISP\_GAMMAFE\_ATTR\_S 中描述有修改



### 9.3.3 节有删除部分内容

## 文档版本 00B01 (2015-05-27)

第一次临时版本发布。



# 目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 功能描述.....	1
1.2.1 架构 .....	2
1.2.2 开发模式 .....	2
1.2.3 内部流程 .....	2
1.2.4 软件流程 .....	3
2 系统控制.....	5
2.1 功能概述.....	5
2.2 API 参考 .....	5
2.3 数据类型.....	38
3 AE .....	85
3.1 概述.....	85
3.2 重要概念.....	85
3.3 功能描述.....	86
3.4 API 参考 .....	87
3.4.1 AE 库接口 .....	87
3.4.2 AE 控制模块 .....	92
3.4.3 AI 控制模块 .....	105
3.5 数据类型.....	112
3.5.1 Register .....	112
3.5.2 AE .....	121
3.5.3 AI .....	140
4 AWB.....	153
4.1 概述.....	153
4.2 重要概念.....	153
4.3 功能描述.....	153
4.4 API 参考 .....	154





4.4.1 AWB 库接口 .....	154
4.4.2 AWB 控制模块 .....	158
4.4.3 WB 统计信息 .....	163
4.5 数据类型 .....	165
4.5.1 Register .....	165
4.5.2 WB .....	168
<b>5 CCM .....</b>	<b>189</b>
5.1 概述 .....	189
5.2 重要概念 .....	189
5.3 功能描述 .....	189
5.4 API 参考 .....	190
5.5 数据类型 .....	194
<b>6 IMP .....</b>	<b>200</b>
6.1 Sharpen .....	200
6.1.1 功能描述 .....	200
6.1.2 API 参考 .....	200
6.1.3 数据类型 .....	202
6.2 Gamma .....	210
6.2.1 功能描述 .....	210
6.2.2 API 参考 .....	210
6.2.3 数据类型 .....	212
6.3 DRC .....	214
6.3.1 功能描述 .....	214
6.3.2 API 参考 .....	214
6.3.3 数据类型 .....	217
6.4 镜头阴影校正 .....	222
6.4.1 功能描述 .....	222
6.4.2 API 参考 .....	222
6.4.3 数据类型 .....	225
6.5 Defect Pixel .....	226
6.5.1 功能描述 .....	226
6.5.2 API 参考 .....	227
6.5.3 数据类型 .....	235
6.6 Crosstalk Removal .....	242
6.6.1 概述 .....	242
6.6.2 功能描述 .....	242
6.6.3 API 参考 .....	243
6.6.4 数据类型 .....	245
6.7 去噪算法 .....	247



6.7.1 概述 .....	247
6.7.2 功能描述 .....	247
6.7.3 API 参考 .....	247
6.7.4 数据类型 .....	249
6.8 UVNR .....	253
6.8.1 功能描述 .....	253
6.8.2 API 参考 .....	253
6.8.3 数据类型 .....	255
6.9 DIS .....	260
6.9.1 概述 .....	260
6.9.2 功能描述 .....	260
6.9.3 API 参考 .....	260
6.9.4 数据类型 .....	264
6.10 Defog.....	265
6.10.1 功能描述 .....	265
6.10.2 API 参考 .....	265
6.10.3 数据类型 .....	267
6.11 去伪彩.....	269
6.11.1 功能描述.....	269
6.11.2 API 参考 .....	269
6.11.3 数据类型.....	271
6.12 去马赛克.....	274
6.12.1 功能描述 .....	274
6.12.2 API 参考 .....	274
6.12.3 数据类型 .....	276
6.13 黑电平.....	280
6.13.1 功能描述 .....	280
6.13.2 API 参考 .....	280
6.13.3 数据类型 .....	282
6.14 去 FPN .....	283
6.14.1 功能描述 .....	283
6.14.2 API 参考 .....	284
6.14.3 数据类型 .....	288
6.15 ACM.....	292
6.15.1 功能描述 .....	292
6.15.2 API 参考 .....	292
6.15.3 数据类型 .....	297
6.16 ColorTone.....	299
6.16.1 功能描述 .....	299



6.16.2 API 参考 .....	299
6.16.3 数据类型 .....	301
6.17 GAMMAFE .....	302
6.17.1 功能描述 .....	302
6.17.2 API 参考 .....	302
6.17.3 数据类型 .....	304
6.18 获取 ISP 模块虚拟地址 .....	305
6.18.1 功能描述 .....	305
6.18.2 API 参考 .....	305
6.18.3 数据类型 .....	306
6.19 查询内部状态信息 .....	307
6.19.1 功能描述 .....	307
6.19.2 API 参考 .....	308
6.19.3 数据类型 .....	309
<b>7 RGBIR .....</b>	<b>311</b>
7.1 功能描述 .....	311
7.2 MPI 参考 .....	311
7.3 数据类型 .....	316
<b>8 统计信息 .....</b>	<b>319</b>
8.1 概述 .....	319
8.2 API 参考 .....	319
8.3 数据类型 .....	322
<b>9 Cmos 默认参数配置 .....</b>	<b>342</b>
9.1 概述 .....	342
9.2 Cmos 结构图示意 .....	343
9.3 INI 文件使用说明 .....	343
9.3.1 AE .....	343
9.3.2 AWB .....	344
9.3.3 ISP .....	345
9.4 注意事项 .....	346
<b>10 Debug .....</b>	<b>348</b>
10.1 概述 .....	348
10.2 功能描述 .....	348
10.3 API 参考 .....	348
10.4 数据类型 .....	351
<b>11 错误码 .....</b>	<b>352</b>
<b>12 Proc 调试信息说明 .....</b>	<b>353</b>



---

12.1 概述.....	353
12.2 ISP.....	354



## 图目录

图 1-1 ISP 控制结构示意图.....	1
图 1-2 ISP firmware 架构 .....	2
图 1-3 ISP firmware 内部流程 .....	3
图 1-4 ISP firmware 软件结构 .....	3
图 1-5 ISP firmware 使用流程 .....	4
图 2-1 ISP 库与 sensor 库间的接口 .....	23
图 2-2 ISP 库与 AE 库间的接口 .....	25
图 2-3 ISP 库与 AWB 库间的接口 .....	28
图 2-4 ISP 库与 AF 库间的接口 .....	30
图 2-5 寄存器 0x205a2014 的地址描述 .....	62
图 2-6 白色区域选择相关参数 .....	73
图 3-1 AE 模块工作流程图 .....	85
图 3-2 AE 256 段统计信息直方图 .....	86
图 3-3 AE 工作原理图 .....	87
图 3-4 AE 库与 sensor 库间的接口 .....	90
图 3-5 AE 分配路线示意图 .....	98
图 4-1 AWB 工作原理图 .....	154
图 4-2 AWB 库 与 sensor 库间的接口 .....	157
图 4-3 色温曲线的参数示意 .....	182
图 4-4 混合光源场景色温权重设置示例说明（n 为色温分段点个数） .....	182
图 4-5 室外色温范围参数的意义 .....	187
图 5-1 CCM 矩阵 .....	189
图 6-1 u8Asymmetry 的变化对曲线的影响趋势 .....	221
图 6-2 u8SecondPole 的变化对曲线的影响趋势 .....	221
图 6-3 u8Stretch 的变化对曲线的影响趋势 .....	222



图 6-4 CrossTalk Remove 门限 .....	243
图 6-5 DIS 偏移示意图 .....	260
图 6-6 去马赛克处理图 .....	274
图 6-7 FPN 标定示意图 .....	283
图 6-8 FPN 校正示意图 .....	284
图 8-1 Square 模式 .....	333
图 9-1 Cmos 结构示意图 .....	343



## 表目录

表 3-1 P-Iris 步进电机位置与 F 值映射表，以福光 NV03105P 为例 .....	148
表 3-2 P-Iris 镜头相关参数，以福光 NV03105P 为例 .....	149
表 4-1 au16CrMax [16]在不同的增益情况下的设置值（仅供参考） .....	173
表 4-2 au16CrMin [16]在不同的增益情况下的设置值（仅供参考） .....	173
表 4-3 au16CbMax [16]在不同的增益情况下的设置值（仅供参考） .....	174
表 4-4 au16CbMin [16]在不同的增益情况下的设置值（仅供参考） .....	175
表 5-1 au8Sat[16]在不同的增益情况下的设置值，以 mn34220 为例 .....	196
表 6-1 abEnLowLumaShoot [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	204
表 6-2 au8SharpenD [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	205
表 6-3 au8SharpenUd[ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	206
表 6-4 au8OverShoot [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	206
表 6-5 au8UnderShoot [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	207
表 6-6 au8TextureNoiseThd [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	208
表 6-7 au8EdgeNoiseThd [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	208
表 6-8 au16Slope[16]在不同增益情况下对应的设置值 .....	241
表 6-9 u16Strength [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	246
表 6-10 在不同增益情况下对应的设置值 .....	251
表 6-11 au8ColorCast [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	257
表 6-12 au8UvnrThreshold [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	257
表 6-13 au8UvnrStrength [ISP_AUTO_STENGTH_NUM]在不同的增益情况下的设置值 .....	258
表 6-14 au16NpOffset[ISP_AUTO_STRENGTH_NUM]在不同增益情况下对应的设置值 .....	278
表 11-1 ISP API 错误码 .....	352



# 1 概述

## 1.1 概述

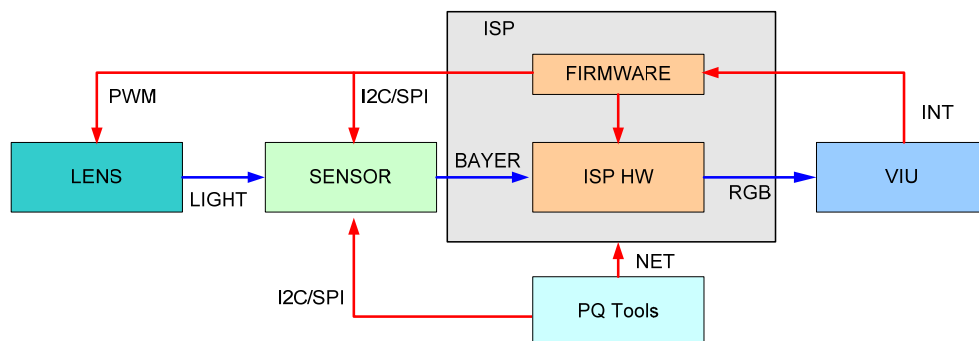
ISP 通过一系列数字图像处理算法完成对数字图像的效果处理。主要包括 3A、坏点校正、去噪、强光抑制、背光补偿、色彩增强、镜头阴影校正等处理。ISP 包括逻辑部分以及运行在其上的 firmware。这里主要介绍 ISP 的用户接口。

## 1.2 功能描述

ISP 的控制结构如图 1-1 所示，lens 将光信号投射到 sensor 的感光区域后，sensor 经过光电转换，将 Bayer 格式的原始图像送给 ISP，ISP 经过算法处理，输出 RGB 空间域的图像给后端的视频采集单元。在这个过程中，ISP 通过运行在其上的 firmware 对 ISP 逻辑，lens 和 sensor 进行相应控制，进而完成自动光圈、自动曝光、自动白平衡等功能。其中，firmware 的运转靠视频采集单元的中断驱动。PQ Tools 工具通过网口或者串口完成对 ISP 的在线图像质量调节。

ISP 由 ISP 逻辑及运行在其上的 Firmware 组成，逻辑单元除了完成一部分算法处理外，还可以统计出当前图像的实时信息。Firmware 通过获取 ISP 逻辑的图像统计信息，重新计算，反馈控制 lens、sensor 和 ISP 逻辑，以达到自动调节图像质量的目的。

图1-1 ISP 控制结构示意图



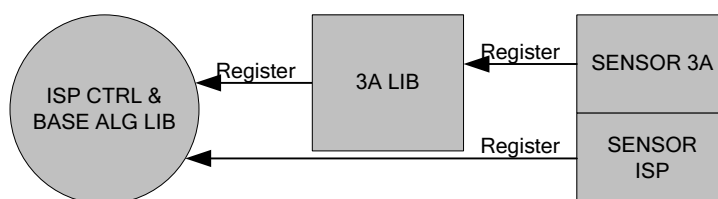
ISP 逻辑主要流程、具体概念和功能点请参见芯片手册。



## 1.2.1 架构

ISP 的 Firmware 包含三部分，一部分是 ISP 控制单元和基础算法库，一部分是 AE/AWB/AF 算法库，一部分是 sensor 库。Firmware 设计的基本思想是单独提供 3A 算法库，由 ISP 控制单元调度基础算法库和 3A 算法库，同时 sensor 库分别向 ISP 基础算法库和 3A 算法库注册函数回调，以实现差异化的 sensor 适配。ISP firmware 架构如图 1-2 所示。

图1-2 ISP firmware 架构



不同的 sensor 都以回调函数的形式，向 ISP 算法库注册控制函数。ISP 控制单元调度基础算法库和 3A 算法库时，将通过这些回调函数获取初始化参数，并控制 sensor，如调节曝光时间、模拟增益、数字增益，控制 lens 步进聚焦或旋转光圈等。

## 1.2.2 开发模式

SDK 支持用户使用多种开发模式：

- a. 用户使用海思的 3A 算法库。这时用户需要根据 ISP 基础算法库和 3A 算法库给出的 sensor 适配接口去适配不同的 sensor。每款 sensor 对应一个文件夹，文件夹中包含两个主要文件：
  - sensor\_cmos.c  
该文件中主要实现 ISP 需要的回调函数，这些回调函数中包含了 sensor 的适配算法，不同的 sensor 可能有所不同。
  - sensor\_ctrl.c  
sensor 的底层控制驱动，主要实现 sensor 的读写和初始化动作。用户可以根据 sensor 的 datasheet 进行这两个文件的开发，必要的时候可以向 sensor 厂家寻求支持。
- b. 用户根据 ISP 库提供的 3A 算法注册接口，实现自己的 3A 算法库开发。这时用户需要根据 ISP 基础算法库和用户的 3A 算法库给出的 sensor 适配接口去适配不同的 sensor。
- c. 用户部分使用海思 3A 算法库，部分实现自己的 3A 算法库。例如 AE 使用 lib\_hiae.a，AWB 使用自己的 3A 算法库。SDK 提供了灵活多变的支持方式。

## 1.2.3 内部流程

Firmware 内部流程分两部分，如图 1-3 所示。一部分是初始化任务，主要完成 ISP 控制单元的初始化、ISP 基础算法库的初始化、3A 算法库的初始化，包括调用 sensor 的回调获取 sensor 差异化的初始化参数；另一部分是动态调节过程，在这个过程中，

firmware 中的 ISP 控制单元调度 ISP 基础算法库和 3A 算法库，实时计算并进行相应控制。Firmware 的软件结构如图 1-4 所示。

图1-3 ISP firmware 内部流程

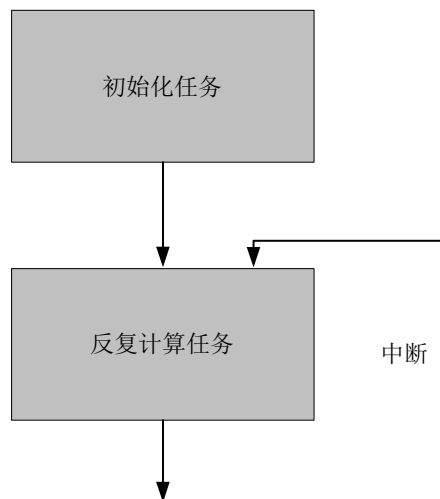
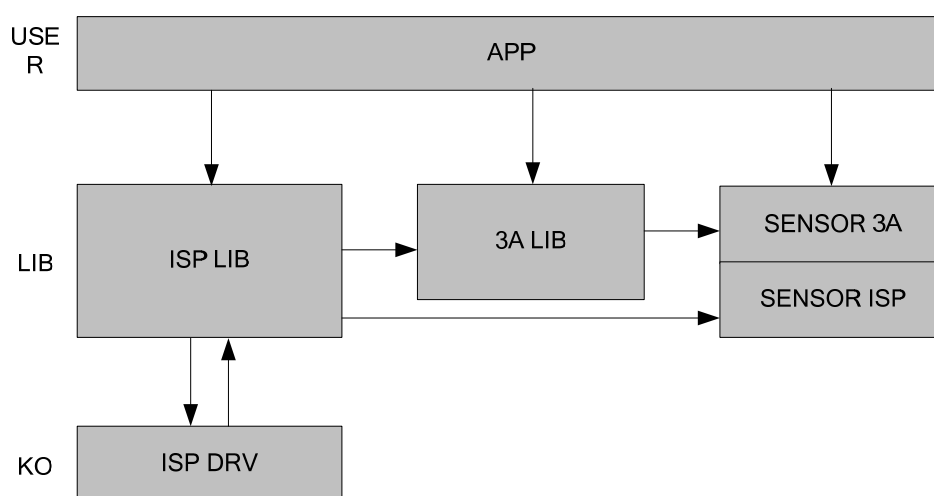


图1-4 ISP firmware 软件结构

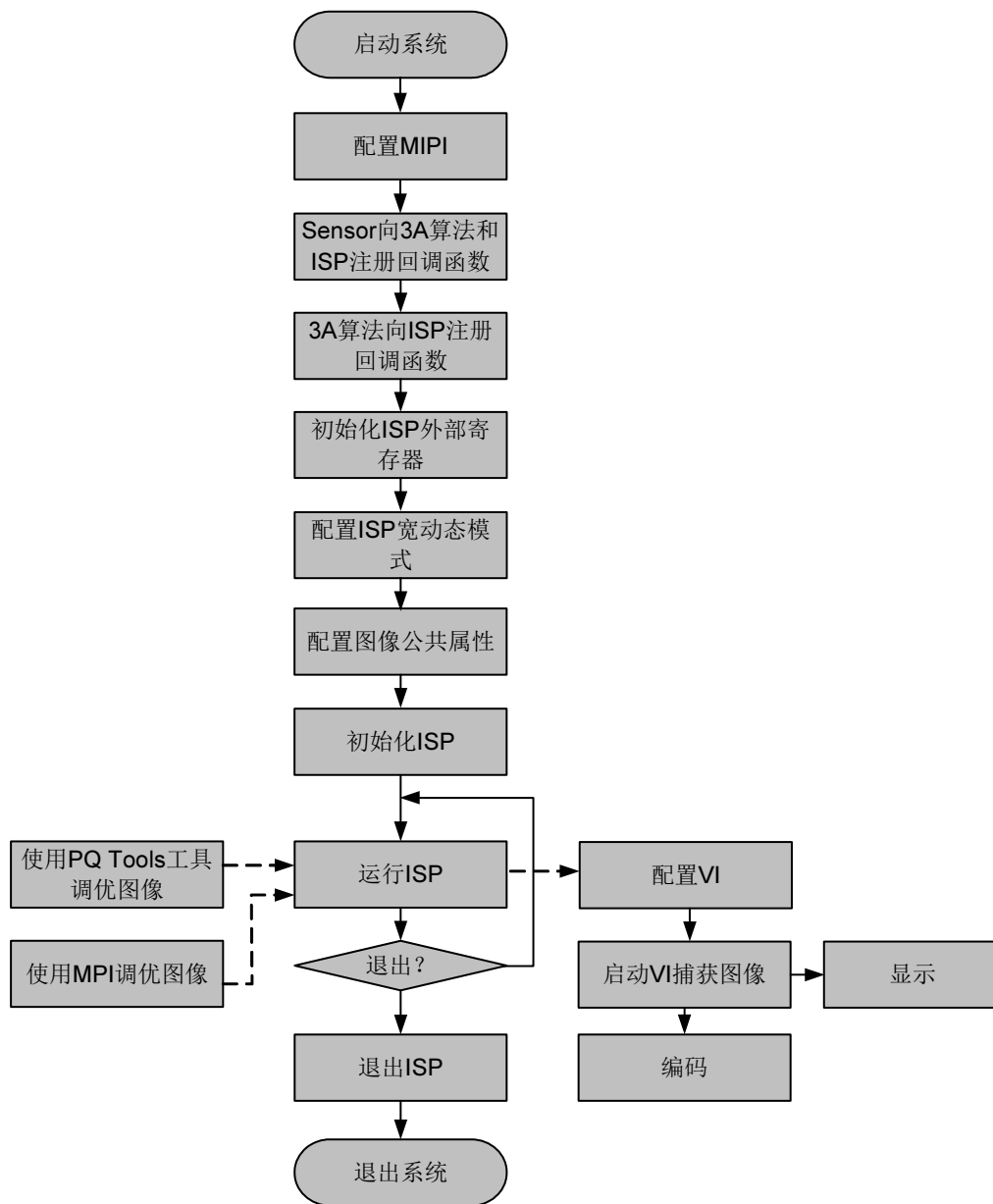


## 1.2.4 软件流程

ISP 作为前端采集部分，需要和视频采集单元（VIU）协同工作。ISP 初始化和基本配置完成后，需要 VIU 进行接口时序匹配。一是为了匹配不同 sensor 的输入时序，二是为 ISP 配置正确的输入时序。待时序配置完成后，ISP 就可以启动 Run 来进行动态图像质量调节。此时输出的图像被 VIU 采集，进而送去显示或编码。软件使用流程如图 1-5 示。

PQ Tools 工具主要完成在 PC 端进行动态图像质量调节，可以调节多个影响图像质量的因子，如去噪强度、色彩转换矩阵、饱和度等。

图1-5 ISP firmware 使用流程



如果用户调试好图像效果后，可以使用 PQ Tools 工具提供的配置文件保存功能进行配置参数保存。在下次启动时系统可以使用 PQ Tools 工具提供的配置文件加载功能加载已经调节好的图像参数。



# 2 系统控制

## 2.1 功能概述

系统控制部分包含了 ISP 公共属性配置，初始化 ISP Firmware、运行 ISP firmware、退出 ISP firmware，设置 ISP 各模块等功能。

## 2.2 API 参考

本文档中接口，如无特殊说明，支持多进程。

- [HI\\_MPI\\_ISP\\_MemInit](#): 初始化 ISP 外部寄存器。
- [HI\\_MPI\\_ISP\\_Init](#): 初始化 ISP firmware。
- [HI\\_MPI\\_ISP\\_Run](#): 运行 ISP firmware。
- [HI\\_MPI\\_ISP\\_Exit](#): 退出 ISP firmware。
- [HI\\_MPI\\_ISP\\_SetPubAttr](#): 设置 ISP 公共属性。
- [HI\\_MPI\\_ISP\\_GetPubAttr](#): 获取 ISP 公共属性。
- [HI\\_MPI\\_ISP\\_SetFMWState](#): 设置 ISP firmware 状态。
- [HI\\_MPI\\_ISP\\_GetFMWState](#): 获取 ISP firmware 状态。
- [HI\\_MPI\\_ISP\\_SetWDRMode](#): 设置 ISP 宽动态模式。
- [HI\\_MPI\\_ISP\\_GetWDRMode](#): 获取 ISP 宽动态模式。
- [HI\\_MPI\\_ISP\\_SetModuleControl](#): 设定 ISP 功能模块的控制。
- [HI\\_MPI\\_ISP\\_GetModuleControl](#): 获取 ISP 功能模块的控制。
- [HI\\_MPI\\_ISP\\_SetRegister](#): 设置寄存器值。
- [HI\\_MPI\\_ISP\\_GetRegister](#): 获取寄存器值。
- [HI\\_MPI\\_ISP\\_GetVDTimeOut](#): 获取 ISP 中断信息。
- [HI\\_MPI\\_ISP\\_SensorRegCallBack](#): ISP 提供的 sensor 注册的回调接口。
- [HI\\_MPI\\_ISP\\_SensorUnRegCallBack](#): ISP 提供的 sensor 反注册的回调接口。
- [HI\\_MPI\\_ISP\\_AELibRegCallBack](#): ISP 提供的 AE 库注册的回调接口。
- [HI\\_MPI\\_ISP\\_AELibUnRegCallBack](#): ISP 提供的 AE 库反注册的回调接口。



- [HI\\_MPI\\_ISP\\_AWBLibRegCallBack](#): ISP 提供的 AWB 库注册的回调接口。
- [HI\\_MPI\\_ISP\\_AWBLibUnRegCallBack](#): ISP 提供的 AWB 库反注册的回调接口。
- [HI\\_MPI\\_ISP\\_AFLibRegCallBack](#): ISP 提供的 AF 库注册的回调接口。
- [HI\\_MPI\\_ISP\\_AFLibUnRegCallBack](#): ISP 提供的 AF 库反注册的回调接口。
- [HI\\_MPI\\_ISP\\_SetBindAttr](#): 设置 ISP 库与 3A 库、sensor 的绑定关系。
- [HI\\_MPI\\_ISP\\_GetBindAttr](#): 获取 ISP 库与 3A 库、sensor 的绑定关系。
- [HI\\_MPI\\_ISP\\_SetDCFInfo](#): 设置 DCF 参数。
- [HI\\_MPI\\_ISP\\_GetDCFInfo](#): 获取 DCF 参数。
- [HI\\_MPI\\_ISP\\_SetModParam](#): 设置 ISP 的模块参数。
- [HI\\_MPI\\_ISP\\_GetModParam](#): 获取 ISP 的模块参数。

## HI\_MPI\_ISP\_MemInit

### 【描述】

初始化 ISP 外部寄存器。

### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
<a href="#">HI_ERR_ISP_SNS_UNREGISTER</a>	Sensor 未注册。
HI_ERR_ISP_ILLEGAL_PARAM	参数无效。

### 【需求】



- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 外部寄存器初始化前需要确保 ko 已加载，sensor 向 ISP 注册了回调函数。
- 调用本接口后，才能调用 [HI\\_MPI\\_ISP\\_SetWDRMode](#) 和 [HI\\_MPI\\_ISP\\_SetPubAttr](#) 分别配置 WDR 模式和图像公共属性。
- 不支持多进程，必须要与 [sensor\\_register\\_callback](#)、[HI\\_MPI\\_AE\\_Register](#)、[HI\\_MPI\\_AWB\\_Register](#)、[HI\\_MPI\\_ISP\\_Init](#)、[HI\\_MPI\\_ISP\\_Run](#)、[HI\\_MPI\\_ISP\\_Exit](#) 接口在同一个进程调用。
- 不支持重复调用本接口。
- 推荐调用 [HI\\_MPI\\_ISP\\_Exit](#) 后，再调用本接口重新初始化。
- Huawei LiteOS 没有内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk\_init.c 中执行的相关过程。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_Init

#### 【描述】

初始化 ISP firmware。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】



接口返回值	含义
HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
HI_ERR_ISP_NOT_INIT	没有初始化
<a href="#">HI_ERR_ISP_SNS_UNREGISTER</a>	Sensor 未注册。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 初始化前需要确保 ko 已加载，sensor 向 ISP 注册了回调函数。
- 初始化前需要确保已调用 [HI\\_MPI\\_ISP\\_MemInit](#) 初始化 ISP 外部寄存器。
- 初始化前需要确保已调用 [HI\\_MPI\\_ISP\\_SetWDRMode](#) 和 [HI\\_MPI\\_ISP\\_SetPubAttr](#) 分别配置 WDR 模式和图像公共属性。
- 不支持多进程，必须要与 sensor\_register\_callback、[HI\\_MPI\\_AE\\_Register](#)、[HI\\_MPI\\_AWB\\_Register](#)、[HI\\_MPI\\_ISP\\_MemInit](#)、[HI\\_MPI\\_ISP\\_Run](#)、[HI\\_MPI\\_ISP\\_Exit](#) 接口在同一个进程调用。
- 不支持重复调用本接口。
- 推荐调用 [HI\\_MPI\\_ISP\\_Exit](#) 后，再调用本接口重新初始化。
- 如果使能 JPEG DCF 功能，调用本接口前须调用 [HI\\_MPI\\_VB\\_SetSupplementConf](#)（请参考《HiMPP IPC V2.0 媒体处理软件开发参考》的系统控制章节 2.2 小节），将 stSupplementConf 配置为 VB\_SUPPLEMENT\_JPEG\_MASK。
- Huawei LiteOS 没内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk\_init.c 中执行的相关过程。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_Run

#### 【描述】

运行 ISP firmware。

#### 【语法】

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

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_SNS_UNREGISTER</a>	Sensor 未注册。
HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
HI_ERR_ISP_NOT_INIT	没有初始化

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

- 运行前需要确保 sensor 已经初始化，并且向 ISP 注册了回调函数。
- 运行前需要确保已调用 [HI\\_MPI\\_ISP\\_Init](#) 初始化 ISP。
- 不支持多进程，必须要与 sensor\_register\_callback、[HI\\_MPI\\_AE\\_Register](#)、[HI\\_MPI\\_AWB\\_Register](#)、[HI\\_MPI\\_ISP\\_MemInit](#)、[HI\\_MPI\\_ISP\\_Init](#)、[HI\\_MPI\\_ISP\\_Exit](#) 接口在同一个进程调用。
- 该接口是阻塞接口，建议用户采用实时线程处理。

【举例】

无

【相关主题】

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

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

【描述】

退出 ISP firmware。





#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

无

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 调用 [HI\\_MPI\\_ISP\\_Init](#) 和 [HI\\_MPI\\_ISP\\_Run](#) 之后，再调用本接口退出 ISP firmware。
- 不支持多进程，必须要与 [sensor\\_register\\_callback](#)、[HI\\_MPI\\_AE\\_Register](#)、[HI\\_MPI\\_AWB\\_Register](#)、[HI\\_MPI\\_ISP\\_MemInit](#)、[HI\\_MPI\\_ISP\\_Init](#)、[HI\\_MPI\\_ISP\\_Run](#) 接口在同一个进程调用。
- 支持重复调用本接口。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_SetPubAttr

#### 【描述】

设置 ISP 公共属性。

#### 【语法】



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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstPubAttr	ISP 公共属性。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 图像属性即对应的 sensor 的采集属性。
- ISP 启动时，需要确保已调用 [HI\\_MPI\\_ISP\\_MemInit](#) 初始化 ISP 外部寄存器。
- 支持在 ISP 运行之后，调用本接口实现动态分辨率和帧率切换。
- 调用本接口后 ISP 内的处理流程：a) ISP firmware 判断图像分辨率和帧率是否变化，若都不变则直接返回；否则，ISP firmware 会调用 sensor cmos.c 里面的 [cmos\\_set\\_image\\_mode](#) 函数改变 sensor 模式；b) 若 sensor 模式改变（返回值为 0），则 ISP firmware 会调用 [sensor\\_init](#) 函数重新配置 sensor；c) ISP firmware 将帧率信息传给海思 AE 库，并决定是否更改帧率。
- 若调用本接口实现动态分辨率和帧率切换时 sensor 模式发生了改变，请参照 [sample](#) 提供的切换流程操作（先停掉 Vi 设备，再切换，最后启动 Vi 设备）。另外，动态分辨率和帧率切换时，切换的分辨率和帧率必须有一项要不同（即不能切换到自己本身），否则，sensor 可能不会重新初始化而导致异常。



- 使用 Vi Dev 和 ISP 提供的裁剪功能时，需要注意：若裁剪后的分辨率和帧率，小于另一组 sensor 模式的分辨率和帧率，则调用本接口会先切换到对应的 sensor 模式。
- 用户可以更改 sensor cmos.c 里面的 `cmos_set_image_mode` 函数调整 sensor 模式切换的顺序。如只提供了 5M30fps 和 1080P60fps 初始化序列的 sensor，若要运行 1080P30fps，可以从 5M30fps 裁剪得到，也可以从 1080P60fps 降帧得到，修改 `cmos_set_image_mode` 函数实现即可。

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetPubAttr

【描述】

获取 ISP 公共属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstImageAttr	ISP 公共属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】



- 头文件: hi\_comm\_isp.h、mpi\_isp.h
- 库文件: libisp.a

【注意】

ISP 运行之前, 需要先设置 ISP 公共属性才能调用本接口。

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_SetFMWState

【描述】

设置 ISP firmware 状态。

【语法】

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

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
enState	ISP firmware 状态。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 <a href="#">错误码</a> 。

【错误码】

无

【需求】

- 头文件: hi\_comm\_isp.h、mpi\_isp.h
- 库文件: libisp.a

【注意】



当 enState 值为 ISP\_FMW\_STATE\_FREEZE 后，ISP Firmware 的 3A 算法，Sharpen 算法，DRC 算法，Crosstalk removal 算法，NR 算法，去雾算法，去马赛克算法，黑电平算法，去 FPN 算法，ACM 算法，WDR 算法等会冻结，Sensor 的寄存器也会停止配置，并保持冻结前的值。当 enState 值为 ISP\_FMW\_STATE\_RUN 后，ISP firmware 正常运行。

**【举例】**

无

**【相关主题】**

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

## HI\_MPI\_ISP\_GetFMWState

**【描述】**

获取 ISP firmware 状态。

**【语法】**

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

**【参数】**

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
penState	ISP firmware 状态。	输出

**【返回值】**

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

**【错误码】**

无

**【需求】**

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

**【注意】**

无



【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_SetWDRMode

【描述】

设置 ISP 宽动态模式。

【语法】

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

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstWDRMode	宽动态模式。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

- ISP 启动时，需要确保已调用 [HI\\_MPI\\_ISP\\_MemInit](#) 初始化 ISP 外部寄存器。
- 支持在 ISP 运行之后，调用本接口实现宽动态切换。
- 如果在相同的 WDR 模式之间切换时，请上层应用程序不要重新设置 MIPI 接口，否则会导致采集不到图像。在相同的 WDR 模式之间进行切换时，建议上层应用程序判断当前的 WDR 模式与要切换的 WDR 模式是否相同，如果相同，则可以直接退出即可，不用进行切换。

【举例】

无



【相关主题】

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

## HI\_MPI\_ISP\_GetWDRMode

【描述】

获取 ISP 宽动态模式。

【语法】

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

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstWDRMode	获取宽动态模式。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_SetModuleControl

【描述】

设定 ISP 功能模块的控制。



#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
punModCtrl	模块控制值。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

无

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 该接口可控制 ISP 各功能模块的使能。
- 该接口对应的寄存器与各模块的使能寄存器复用。
- punModCtrl 中每个比特位控制着 ISP 中的一个功能模块的使能，0 表示开启该模块；1 表示关闭该模块。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetModuleControl

#### 【描述】

获取 ISP 功能模块的控制。

#### 【语法】





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

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
punModCtrl	模块控制值。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

头文件：hi\_comm\_isp.h、mpi\_isp.h

库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_SetRegister

【描述】

设置寄存器值。

【语法】

```
HI_S32 HI_MPI_ISP_SetRegister(ISP_DEV IspDev, HI_U32 u32Addr, HI_U32
```



```
u32Value);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
u32Addr	寄存器地址。	输入
u32Value	寄存器值。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

无

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

寄存器地址由 base 和 offset 组成，其中 base 的格式：ISP 逻辑寄存器为 0x205A0000；ISP 外部寄存器为 0x10000；海思 AE 外部寄存器为 0x20000；海思 AWB 外部寄存器为 0x30000；海思 AF 外部寄存器为 0x40000。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetRegister

#### 【描述】

获取寄存器值。

#### 【语法】

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



### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
u32Addr	寄存器地址。	输入
pu32Value	寄存器值。	输出

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

寄存器地址由 base 和 offset 组成，其中 base 的格式：ISP 逻辑寄存器为 0x205A0000；ISP 外部寄存器为 0x10000；海思 AE 外部寄存器为 0x20000；海思 AWB 外部寄存器为 0x30000；海思 AF 外部寄存器为 0x40000。

### 【举例】

无

### 【相关主题】

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

## HI\_MPI\_ISP\_GetVDTimeOut

### 【描述】

获取 ISP 中断信息。

### 【语法】



```
HI_S32 HI_MPI_ISP_GetVdTimeOut (ISP_DEV IspDev, ISP\_VD\_INFO\_S  
*pstIspVdInfo, HI_U32 u32MilliSec);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspVdInfo	ISP 帧信息结构指针	输出
u32MilliSec	超时时间，单位 ms	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
HI_ERR_ISP_NO_INT	ISP 无中断。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 该接口表示获取 ISP 产生中断的相关信息，包括是否产生了中断，中断产生时的当前 ISP 帧信息。
- u32MilliSec 参数的单位是毫秒，指超时时间。即在 u32MilliSec 毫秒内，如果获取不到 ISP 中断，则函数返回。当 u32MilliSec 设为 0 时，表示阻塞模式，程序一直等待，直到获取到 ISP 中断才返回。

#### 【举例】

无

#### 【相关主题】

无



## HI\_MPI\_ISP\_SensorRegCallBack

### 【描述】

ISP 提供的 sensor 注册的回调接口。

### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
SensorId	向 ISP 注册的 Sensor 的 Id。	输入
pstRegister	Sensor 注册结构体指针。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

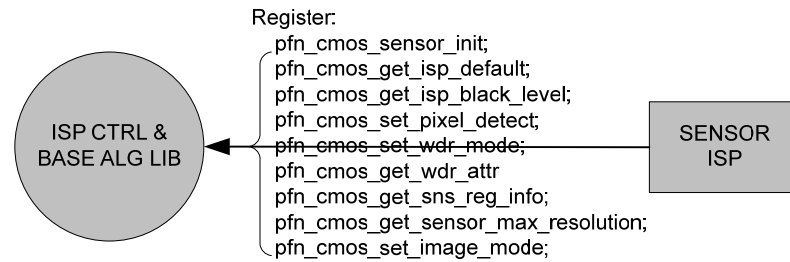
### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

- SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 注册的 sensor 和向 3A 注册的 sensor 是否为同一个 sensor。
- ISP 通过 sensor 注册的一系列回调接口，获取差异化的初始化参数，并控制 sensor。

图2-1 ISP 库与 sensor 库间的接口



### 【举例】

```
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;
}
```

### 【相关主题】

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

## HI\_MPI\_ISP\_SensorUnRegCallBack

### 【描述】

ISP 提供的 sensor 反注册的回调接口。

### 【语法】

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



#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
SensorId	向 ISP 注册的 Sensor 的 Id。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 反注册的 sensor 和向 3A 反注册的 sensor 是否为同一个 sensor。

#### 【举例】

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

#### 【相关主题】

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

## HI\_MPI\_ISP\_AELibRegCallBack

#### 【描述】

ISP 提供的 AE 库注册的回调接口。

#### 【语法】

```
HI_S32 HI_MPI_ISP_AELibRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S *pstAeLib,
ISP\_AE\_REGISTER\_S *pstRegister);
```



【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAeLib	AE 库结构体指针。	输入
pstRegister	AE 库注册结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

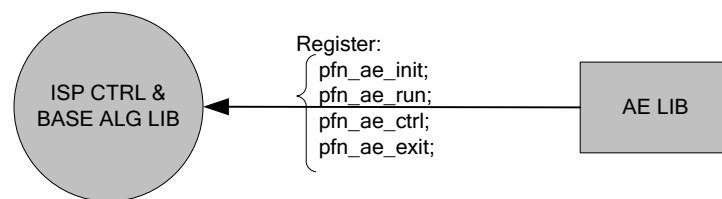
【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

ISP 提供统一的 AE 算法库接口，初始化、运行、控制、销毁 AE 算法库。使用海思 AE 算法库时，不需要关注此接口；使用用户自己的 AE 算法库时，需要调用此接口向 ISP 注册回调函数。

图2-2 ISP 库与 AE 库间的接口



【举例】

```
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");  
}
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_AELibUnRegCallBack

#### 【描述】

ISP 提供的 AE 库反注册的回调接口。

#### 【语法】

```
HI_S32 HI_MPI_ISP_AELibUnRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S *pstAeLib);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAeLib	AE 库结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

使用海思 AE 算法库时，不需要关注此接口；使用用户自己的 AE 算法库时，需要调用此接口向 ISP 反注册回调函数。

#### 【举例】

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



```
}  
return s32Ret;
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_AWBLibRegCallBack

#### 【描述】

ISP 提供的 AWB 库注册的回调接口。

#### 【语法】

```
HI_S32 HI_MPI_ISP_AWBLibRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S *pstAwbLib,  
ISP\_AWB\_REGISTER\_S *pstRegister);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAwbLib	AWB 库结构体指针。	输入
pstRegister	AWB 库注册结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

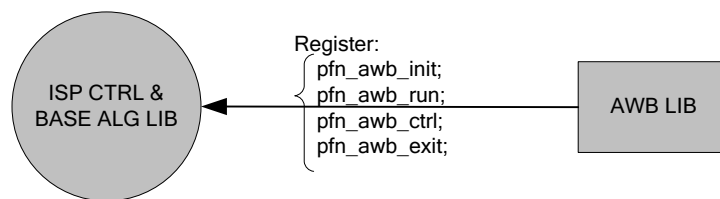
- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

ISP 提供统一的 AWB 算法库接口，初始化、运行、控制、销毁 AWB 算法库。使用海思 AWB 算法库时，不需要关注此接口；使用用户自己的 AWB 算法库时，需要调用此接口向 ISP 注册回调函数。



图2-3 ISP 库与 AWB 库间的接口



【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_AWBLibUnRegCallBack

【描述】

ISP 提供的 AWB 库反注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_AWBLibUnRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S  
*pstAwbLib);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAwbLib	AWB 库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】



无

#### 【举例】

使用海思 AWB 算法库时，不需要关注此接口；使用用户自己的 AWB 算法库时，需要调用此接口向 ISP 反注册回调函数。

#### 【相关主题】

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

## HI\_MPI\_ISP\_AFLibRegCallBack

#### 【描述】

ISP 提供的 AF 库注册的回调接口。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAfLib	AF 库结构体指针。	输入
pstRegister	AF 库注册结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

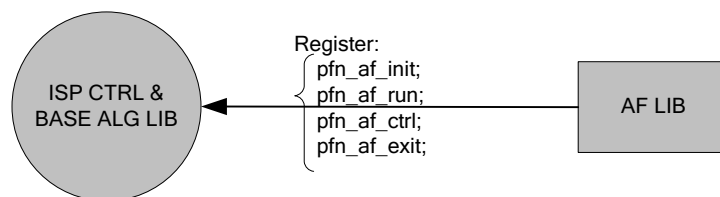
- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

ISP 提供统一的 AF 算法库接口，初始化、运行、控制、销毁 AF 算法库。使用海思 AF 算法库时，不需要关注此接口；使用用户自己的 AF 算法库时，需要调用此接口向 ISP 注册回调函数。



图2-4 ISP 库与 AF 库间的接口



【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_AFLibUnRegCallBack

【描述】

ISP 提供的 AF 库反注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_AFLibUnRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S *pstAfLib);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAfLib	AF 库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】



使用海思 AF 算法库时，不需要关注此接口；使用用户自己的 AF 算法库时，需要调用此接口向 ISP 反注册回调函数。

**【举例】**

无

**【相关主题】**

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

## HI\_MPI\_ISP\_SetBindAttr

**【描述】**

设置 ISP 库与 3A 库、sensor 的绑定关系。

**【语法】**

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

**【参数】**

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstBindAttr	绑定结构体指针。	输入

**【返回值】**

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

**【注意】**

不是必须调用的接口，仅当注册多个 AE/AWB/AF 库，并希望切换算法库时才需要调用。当注册多个 AE/AWB/AF 库时，默认绑定的为最后一个注册的 AE 库、AWB 库、AF 库。

**【举例】**

无



【相关主题】

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

## HI\_MPI\_ISP\_GetBindAttr

【描述】

获取 ISP 库与 3A 库、sensor 的绑定关系。

【语法】

```
HI_S32 HI_MPI_ISP_GetBindAttr(ISP_DEV IspDev, ISP\_BIND\_ATTR\_S
*pstBindAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstBindAttr	绑定结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_SetDCFInfo

【描述】

设置 DCF 参数。



### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspDCF	DCF 参数结构体指针。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

调用该接口前须调用 HI\_MPI\_VB\_SetSupplementConf（请参考《HiMPP IPC V2.0 媒体处理软件开发参考》的系统控制章节 2.2 小节），将 stSupplementConf 配置为 VB\_SUPPLEMENT\_JPEG\_MASK。

### 【举例】

```
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);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetDCFInfo

#### 【描述】

获取 DCF 参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDCFInfo(ISP_DEV IspDev, ISP\_DCF\_INFO\_S *pstIspDCF)
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspDCF	DCF 参数结构体指针。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】



- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

调用该接口前须调用 HI\_MPI\_VB\_SetSupplementConf（请参考《HiMPP IPC V2.0 媒体处理软件开发参考》的系统控制章节 2.2 小节），将 stSupplementConf 配置为 VB\_SUPPLEMENT\_JPEG\_MASK。因为 DCF 信息需要存储在 ISP Init 时分配 DCF Buffer 中。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_SetModParam

#### 【描述】

设置 ISP 模块参数。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
pstIspModParam	ISP 模块参数结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
HI_ERR_ISP_NOT_SUPPORT	不支持的操作。

#### 【需求】



- 头文件: hi\_comm\_isp.h、mpi\_isp.h
- 库文件: libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetModParam

【描述】

获取 ISP 模块参数。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstIspModParam	ISP 模块参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件: hi\_comm\_isp.h、mpi\_isp.h
- 库文件: libisp.a

【注意】



无

【举例】

无

【相关主题】

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

## 2.3 数据类型

本文档中变量，如未明确指定取值范围，则默认是数据类型对应的取值范围。例如 HI\_U8 数据类型的变量取值范围为[0, 255]。本文档中变量，如未明确指定数据精度，则默认是 1。

- [RECT\\_S](#): 定义裁剪窗口起始位置和图像宽高。
- [ISP\\_BAYER\\_FORMAT\\_E](#): 定义输入 Bayer 图像数据格式。
- [ISP\\_PUB\\_ATTR\\_S](#): 定义 ISP 公共属性。
- [ISP\\_FMW\\_STATE\\_E](#): 定义 ISPfirmware 状态。
- [WDR\\_MODE\\_E](#): 定义宽动态模式。
- [ISP\\_WDR\\_MODE\\_S](#): 定义 ISP 宽动态模式。
- [ISP\\_MODULE\\_CTRL\\_U](#): 定义 ISP 功能模块的控制。
- [ISP\\_VD\\_INFO\\_S](#): 定义 ISP 帧信息。
- [ISP\\_SENSOR\\_REGISTER\\_S](#): 定义 sensor 注册结构体。
- [ISP\\_SENSOR\\_EXP\\_FUNC\\_S](#): 定义 sensor 回调函数结构体。
- [ISP\\_CMOS\\_SENSOR\\_IMAGE\\_MODE](#): 定义 sensor 输出的宽高和帧率属性。
- [ISP\\_CMOS\\_DEFAULT\\_S](#): 定义 ISP 基础算法库的初始化参数结构体。
- [ISP\\_CMOS\\_BLACK\\_LEVEL\\_S](#): 定义 sensor 的黑电平结构体。
- [ISP\\_SNS\\_REGS\\_INFO\\_S](#): 定义 sensor 的寄存器信息。
- [ALG\\_LIB\\_S](#): 定义 AE/AWB/AF 算法库结构体。
- [ISP\\_BIND\\_ATTR\\_S](#): 定义 ISP 库与 Sensor、3A 库之间绑定关系的结构体。
- [ISP\\_CTRL\\_PROC\\_WRITE\\_S](#): 定义 ISP 的 PROC 信息。
- [ISP\\_CTRL\\_CMD\\_E](#): 定义 ISP 对 3A 的控制命令。
- [ISP\\_AE\\_REGISTER\\_S](#): 定义 AE 注册结构体。
- [ISP\\_AE\\_EXP\\_FUNC\\_S](#): 定义 AE 回调函数结构体。
- [ISP\\_AE\\_PARAM\\_S](#): 定义 ISP 提供给 AE 的初始化参数结构体。
- [ISP\\_AE\\_INFO\\_S](#): 定义 ISP 提供给 AE 的统计信息结构体。
- [ISP\\_AE\\_RESULT\\_S](#): 定义 AE 库返回给 ISP 的配置寄存器结构体。
- [ISP\\_AWB\\_REGISTER\\_S](#): 定义 AWB 注册结构体。
- [ISP\\_AWB\\_EXP\\_FUNC\\_S](#): 定义 AWB 回调函数结构体。
- [ISP\\_AWB\\_PARAM\\_S](#): 定义 ISP 提供给 AWB 的初始化参数结构体。



- [ISP\\_AWB\\_INFO\\_S](#): 定义 ISP 提供给 AWB 的统计信息结构体。
- [ISP\\_AWB\\_RESULT\\_S](#): 定义 AWB 库返回给 ISP 的配置寄存器结构体。
- [ISP\\_AF\\_REGISTER\\_S](#): 定义 AF 注册结构体。
- [ISP\\_AF\\_EXP\\_FUNC\\_S](#): 定义 AF 回调函数结构体。
- [ISP\\_AF\\_PARAM\\_S](#): 定义 ISP 提供给 AF 的初始化参数结构体。
- [ISP\\_AF\\_INFO\\_S](#): 定义 ISP 提供给 AF 的统计信息结构体。
- [ISP\\_AF\\_RESULT\\_S](#): 定义 AF 库返回给 ISP 的配置寄存器结构体。
- [ISP\\_DCF\\_INFO\\_S](#): 定义 DCF 信息参数结构体。
- [ISP\\_MOD\\_PARAM\\_S](#): 定义 ISP 模块参数结构体。
- [ISP\\_MODULE\\_PARAM\\_S](#): 定义 ISP 模块参数结构体。
- [ISP\\_OP\\_TYPE\\_E](#): 定义模块运行状态。

## RECT\_S

### 【说明】

定义裁剪窗口起始位置和图像宽高。

### 【定义】

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

### 【成员】

成员名称	描述
s32X	水平方向起始位置，取值范围[0, u32Width - 480]。
s32Y	垂直方向起始位置，取值范围[0, u32Height - 240]。
u32Width	图像宽度，取值范围[480, 2592]。
u32Height	图像高度，取值范围[240, 2200]。

### 【注意事项】

水平方向起始位置与图像宽度之和应小于 sensor 输出的图像宽度；垂直方向起始位置与图像高度之和应小于 sensor 输出的图像高度。

### 【相关数据类型及接口】

无



## ISP\_BAYER\_FORMAT\_E

### 【说明】

定义输入 Bayer 图像数据格式。

### 【定义】

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

### 【成员】

成员名称	描述
BAYER_RGGB	RGGB 排列方式。
BAYER_GRBG	GRGB 排列方式。
BAYER_GBRG	GBRG 排列方式。
BAYER_BGGR	BGGR 排列方式。

### 【注意事项】

该格式可以从所使用 sensor 的 DataSheet 上获取，并和裁剪起始位置相关。

### 【相关数据类型及接口】

无

## ISP\_PUB\_ATTR\_S

### 【说明】

定义 ISP 公共属性。

### 【定义】

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



【成员】

成员名称	描述
stWndRect	裁剪窗口起始位置和图像宽高。
f32FrameRate	输入图像帧率，取值范围为(0.00,255.00]。
enBayer	Bayer 数据格式。

【注意事项】

无

【相关数据类型及接口】

无

## ISP\_FMW\_STATE\_E

【说明】

定义 ISPfirmware 状态。

【定义】

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

【成员】

成员名称	描述
ISP_FMW_STATE_RUN	Firmware 正常运行状态。
ISP_FMW_STATE_FREEZE	Firmware 冻结状态。

【注意事项】

无

【相关数据类型及接口】

无

## WDR\_MODE\_E

【说明】





定义宽动态模式。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
WDR_MODE_NONE	线性模式。
WDR_MODE_BUILT_IN	Sensor 合成 WDR 模式。
WDR_MODE_2To1_LINE	2 帧合成行 WDR 模式。
WDR_MODE_2To1_FRAME	2 帧合成帧 WDR 模式。
WDR_MODE_2To1_FRAME_FULL_RATE	2 帧合成帧 WDR 全帧率模式。
WDR_MODE_3To1_LINE	3 帧合成行 WDR 模式。
WDR_MODE_3To1_FRAME	3 帧合成帧 WDR 模式。
WDR_MODE_3To1_FRAME_FULL_RATE	3 帧合成帧 WDR 全帧率模式。
WDR_MODE_4To1_LINE	4 帧合成行 WDR 模式。
WDR_MODE_4To1_FRAME	4 帧合成帧 WDR 模式。
WDR_MODE_4To1_FRAME_FULL_RATE	4 帧合成帧 WDR 全帧率模式。

### 【注意事项】

- 目前仅支持 WDR\_MODE\_NONE、WDR\_MODE\_BUILT\_IN 模式。
- WDR\_MODE\_BUILT\_IN 需要 sensor 支持。



【相关数据类型及接口】

无

## ISP\_WDR\_MODE\_S

【说明】

定义 ISP 宽动态模式。

【定义】

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

【成员】

成员名称	描述
enWDRMode	宽动态模式。

【注意事项】

无

【相关数据类型及接口】

无

## ISP\_MODULE\_CTRL\_U

【说明】

定义 ISP 功能模块的控制。

【定义】

```
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 bitBypassDehaze      : 1 ; /* [7] */
HI_U32 bitBypassWBGain      : 1 ; /* [8] */
HI_U32 bitBypassShading     : 1 ; /* [9] */
HI_U32 bitBypassACM         : 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 bitBypassUVNR        : 1 ; /* [23] */
HI_U32 bitChnSwitch         : 1 ; /* [24] */
HI_U32 bit2BypassMode       : 2 ; /* [25:26] */
HI_U32 bitBypassRGBIR       : 1 ; /* [27] */
HI_U32 bitBypassAll         : 1 ; /* [28] */
HI_U32 bit5Rsv5             : 3 ; /* [29:31] */

};

}ISP_MODULE_CTRL_U;

```

**【成员】**

成员名称	描述
bitBypassVideoTest	旁路视频测试生成器。
bitBypassBalanceFe	旁路前端黑电平调节
bitBypassISPDGain	旁路数字增益。
bitBypassGammaFe	旁路 GammaFe 表。
bitBypassCrosstalkR	旁路 Crosstalk Removal。
bitBypassDPC	旁路坏点校正。
bitBypassNR	旁路去噪。
bitBypassDehaze	旁路去雾。
bitBypassWBGain	旁路白平衡增益和偏移量。
bitBypassShading	旁路镜头阴影校正。
bitBypassACM	旁路 ACM。
bitBypassDRC	旁路 DRC。



成员名称	描述
bitBypassDemosaic	旁路去马赛克模块。
bitBypassColorMatrix	旁路颜色矩阵。
bitBypassGamma	旁路 Gamma 表。
bitBypassFSWDR	旁路多帧合成 WDR，Hi3518EV200 不支持多帧合成 WDR。
bitGammaFePosition	Hi3518EV200 不支持。
bitBypassCsConv	旁路颜色空间转换。
bitBypassSharpen	旁路 SharpenRGB。
bitBypassUVNR	旁路 UVNR。
bitChnSwitch	Hi3518EV200 不支持。
bit2BypassMode	Hi3518EV200 不支持。
bitBypassRGBIR	旁路 RGBIR 模块。
bitBypassAll	输出直接与输入相连。Hi3518EV200 不支持
bit5Rsv5	保留位。

【注意事项】

无

【相关数据类型及接口】

无

## ISP\_VD\_INFO\_S

【说明】

定义 ISP 帧信息。

【定义】

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

【成员】

成员名称	描述
u32Reserved	保留字节



#### 【注意事项】

当前 ISP 帧信息结构体只有保留变量，即当前没有提供 ISP 帧相关信息。

#### 【相关数据类型及接口】

无

## ISP\_SENSOR\_REGISTER\_S

#### 【说明】

定义 sensor 注册结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
stSnsExp	Sensor 注册的回调函数结构体。

#### 【注意事项】

封装的目的是为了扩展。

#### 【相关数据类型及接口】

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

## ISP\_SENSOR\_EXP\_FUNC\_S

#### 【说明】

定义 sensor 回调函数结构体。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
pfn_cmos_sensor_init	初始化 sensor 的回调函数指针。
pfn_cmos_sensor_global_init	初始化全局变量的回调函数指针。
pfn_cmos_set_image_mode	设置分辨率和帧率切换的回调函数指针。返回值 0 表示 sensor 模式发生改变，ISP 会调用 pfn_cmos_sensor_init 重新配置 sensor；其他返回值表示 sensor 模式没有变化，ISP 不会重新配置 sensor。
pfn_cmos_set_wdr_mode	设置 wdr 模式的回调函数指针。
pfn_cmos_get_isp_default	获取 ISP 基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取 sensor 的黑电平值的回调函数指针，支持根据 sensor 增益动态调整黑电平值。若此处动态调整黑电平值，则无法通过接口 <a href="#">HI_MPI_ISP_SetBlackLevelAttr</a> 设置黑电平。
pfn_cmos_get_sns_reg_info	获取 sensor 寄存器信息的回调函数指针，用于实现内核态配置 AE 信息。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。

#### 【注意事项】

pfn\_cmos\_sensor\_init, pfn\_cmos\_get\_isp\_default, pfn\_cmos\_get\_isp\_black\_level, pfn\_cmos\_set\_pixel\_detect 和 pfn\_cmos\_get\_sns\_reg\_info 必须赋值，其他回调函数指针如果不需要赋值，应置为 NULL。例如有的 sensor 不支持切换分辨率，那么 pfn\_cmos\_set\_image\_mode 需要置为 NULL。

#### 【相关数据类型及接口】

[ISP\\_SENSOR\\_REGISTER\\_S](#)

## ISP\_CMOS\_SENSOR\_IMAGE\_MODE

#### 【说明】

定义 sensor 输出的宽高和帧率属性。

#### 【定义】



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

#### 【成员】

成员名称	描述
u16Width	Sensor 输出的宽度。
u16Height	Sensor 输出的高度。
u16Fps	Sensor 输出的帧率。

#### 【注意事项】

无

#### 【相关数据类型及接口】

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

## ISP\_CMOS\_DEFAULT\_S

#### 【说明】

定义 ISP 基础算法库的初始化参数结构体。

#### 【定义】

```
typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_NOISE_TABLE_S    stNoiseTbl;
    ISP_CMOS_DEMOSAIC_S       stDemosaic;
    ISP_CMOS_GAMMAFE_S        stGammafe;
    ISP_CMOS_GAMMA_S          stGamma;
    ISP_CMOS_RGBSHARPEN_S     stRgbSharpen;
    ISP_CMOS_UVNR_S           stUvnr;
    ISP_CMOS_DPC_S            stDpc;
    ISP_CMOS_LSC_S            stLsc;
    ISP_CMOS_RGBIR_S          stRgbir;
    ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
} ISP_CMOS_DEFAULT_S;
```

#### 【成员】



成员名称	子成员名称	描述
stNoiseTbl	u8SensorIndex	Sensor 类型标记。
	stNrCaliPara	2DNR 校正相关的参数，由校正工具计算得到。
	stIsoParaTable[HI_ISP_NR_ISO_LEVEL_MAX]	2DNR 与 ISO 联动的参数。
stDemosaic	bEnable	该模块是否使能，取值范围为[0,1]。
	u8VhSlope	垂直、水平边缘混合阈值的斜率，调低此参数会提高解析度，加大噪声。 取值范围：[0x0, 0xFF]。一般小于 64。
	u8VhLimit	垂直、水平弱边缘基限值，调高此参数会降低弱边缘锐度，减少噪声。 取值范围：[0x0, 0xFF]。一般大于 16 且小于 64。
	u8VhOffset	垂直、水平弱边缘偏差值，调高此参数会降低弱边缘锐度，减少噪声。 取值范围：[0x0, 0xFF]。
	u8UuSlope	全部边缘混合阈值的斜率，调高此参数会提高解析度、锐度，加大噪声。 取值范围：[0x0, 0x3FF]。
	bFcrEnable	高频去伪彩使能，取值范围为[0,1]。
	u8FcrStrength	去伪彩强度值。值越大，去伪彩强度越强。 取值范围：[0, 0xFF]。
stGammafe	au16Gammafe0	GammaFe 表 0，用于减小查找表插值时出现的误差，与 au16Gammafe1 共同完成图像压缩的功能，由校正工具生成取值范围为 [0,0xFFFF]。
	au16Gammafe1	GammaFe 表 1，与 au16Gammafe0 共同完成图像压缩的功能，由校正工具生成，取值范围为[0,0xFFFF]。
	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果使用默认 gammafe 曲线，可以配置为无效。
stGamma	au16Gamma	Gamma 表，取值范围为[0,0xFFFF]。
	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果使用默认 gamma 曲线，可以配置为无效。





成员名称	子成员名称	描述
stRgbSharpen	abEnPixSel [ISP_AUTO_ISO_STENGTH_NUM]	低照度环境下，图像锐化后的 shoot 的严格控制。正常照度噪声小时，建议设为 0；低照度噪声大时，建议设为 1。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	au8MaxSharpAmt1 [ISP_AUTO_ISO_STENGTH_NUM]	无方向的锐化，设置图像纹理的锐度，通常情况建议在相同增益情形下设置的 u8SharpenD 的值小于 u8SharpenUd 的值，该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。
	au8MaxEdgeAmt [ISP_AUTO_ISO_STENGTH_NUM]	有方向的锐化，设置图像边缘和小细节的锐度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	au8SharpThd2 [ISP_AUTO_ISO_STENGTH_NUM]	图像纹理上噪声的控制阈值。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	au8EdgeThd2 [ISP_AUTO_ISO_STENGTH_NUM]	图像边缘上噪声的控制阈值。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	au8OvershootAmt [ISP_AUTO_ISO_STENGTH_NUM]	设置图像的 overshoot 的强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	au8UndershootAmt [ISP_AUTO_ISO_STENGTH_NUM]	设置图像的 undershoot 强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
stUvnr	UVNR_lutSigma [ISP_AUTO_ISO_STENGTH_NUM]	设置图像的色度降噪的细调参数，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。
	Coring_lutLimit [ISP_AUTO_ISO_STENGTH_NUM]	设置图像在整体偏色情况下的偏色微调强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值。取值范围：[0x0, 0x2]
	UVNR_blendRatio [ISP_AUTO_ISO_STENGTH_NUM]	设置图像的色度降噪的粗调参数，该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。
stDpc	au16Slope[ISP_AUTO_ISO_STENGTH_NUM]	DPC 动态坏点处理强度，取值越大，DPC 处理强度越强，取值范围：[0x0, 0xFF]
	au16BlendRatio[ISP_AUTO_ISO_STENGTH_NUM]	坏点校正过程所需的融合比率，取值越大，图像越绿，取值范围：[0x0, 0x100]



成员名称	子成员名称	描述
	NGTH_NUM]	
stLsc	au32R_Gain[HI_ISP_LSC_GRID_POINTS]	用来储存 LSC 所用 R 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 R 分量阴影矫正增益数据值。
	au32Gr_Gain[HI_ISP_LSC_GRID_POINTS]	用来储存 LSC 所用 Gr 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 Gr 分量阴影矫正增益数据值。
	au32Gb_Gain[HI_ISP_LSC_GRID_POINTS]	用来储存 LSC 所用 Gb 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 Gb 分量阴影矫正增益数据值。
	au32B_Gain[HI_ISP_LSC_GRID_POINTS]	用来储存 LSC 所用 B 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 B 分量阴影矫正增益数据值。
stRgbir	bValid	该结构体的数据是否有效，取值范围为[0, 1]。如果为 0，该结构体的其他数据将不会被写入寄存器，因此非 RGBIR sensor 可以不设置此结构体。
	bEnable	RGBIR 模块使能，取值范围：[0, 1] 0：禁止； 1：使能。
	enIrPosType	Sensor IR 分量位置，取值范围： [ISP_IRPOS_TYPE_GR, ISP_IRPOS_TYPE_GB]
	u16OverExpThresh	过曝值门限，取值范围：[0, 0xFFFF]
	bIrOutEn	IR 图像输出使能，取值范围：[0,1] 0：禁止； 1：使能
	bIrFilterEn	IR 通道 G 插值滤波使能，取值范围：[0,1] 0：禁止； 1：使能
	bRemovelEn	去除红外使能，取值范围：[0,1] 0：禁止； 1：使能



成员名称	子成员名称	描述
	enCompType	增益补偿类型，取值范围： [OP_TYPE_AUTO, OP_TYPE_MANUAL]
	u16ManuGain	手动增益补偿增益大小，取值范围： [0x100,0x3ff] 其中 2bit 整数，8bit 小数
	as16ScaleCoef[15]	颜色矫正系数（矫正获取），取值范围：[-512,511] 有符号数，8bit 精度
stSensorMaxResolution	u32MaxHeight	Sensor 支持的最大高度，防止分辨率切换时配置超过 sensor 支持的高度。
	u32MaxWidth	Sensor 支持的最大宽度，防止分辨率切换时配置超过 sensor 支持的宽度。

#### 【注意事项】

该结构体中的默认值均在 sensor\_cmos.c 中，如果用户需要修改默认值，请修改相应参数，如果用户需要对接新的 sensor，请参考已经提供的其他 sensor 的默认值。

#### 【相关数据类型及接口】

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

## ISP\_CMOS\_BLACK\_LEVEL\_S

#### 【说明】

定义 sensor 的黑电平结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
bUpdate	Sensor 的黑电平是否会动态根据增益改变，取值范围[0,1]。若设置为 HI_TRUE，则无法通过 <a href="#">HI_MPI_ISP_SetBlackLevelAttr</a> 设置黑电平。
au16BlackLevel	Sensor 的黑电平数组，取值范围[0, 65535]。

#### 【注意事项】



如果 sensor 的黑电平不会动态根据增益改变，bUpdate 配置为 HI\_FALSE 即可。

#### 【相关数据类型及接口】

#### ISP\_SENSOR\_EXP\_FUNC\_S

### ISP\_SNS\_REGS\_INFO\_S

#### 【说明】

定义 sensor 的寄存器信息。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	子成员名称	描述
enSnsType	ISP_SNS_I2C_TYPE	Sensor 与 ISP 使用 I2C 接口通信。
	ISP_SNS_SSP_TYPE	Sensor 与 ISP 使用 SSP 接口通信。
u32RegNum	-	曝光结果写到 sensor 时需要配置的寄存器个数，不支持动态修改。
u8Cfg2ValidDelayMax	-	所有 Sensor 寄存器从配置到生效延迟的帧数的最大值，单位为帧，用于保证 sensor 寄存器和 ISP 寄存器的同步。一般情况下，cmos sensor 的曝光时间寄存器的延迟最大，为 1~2 帧，因此配置一般为 1 或 2。
astI2cData	bUpdate	HI_TRUE：数据会配置 sensor 寄存器；HI_FALSE：数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。



成员名称	子成员名称	描述
	u8DevAddr	Sensor 设备地址。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。
astSspData	bUpdate	HI_TRUE: 数据会配置 sensor 寄存器; HI_FALSE: 数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u32DevAddr	Sensor 设备地址。
	u32DevAddrByteNum	Sensor 设备地址位宽。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。

【注意事项】

无

【相关数据类型及接口】

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

## ALG\_LIB\_S

【说明】

定义 AE/AWB/AF 算法库结构体。

【定义】

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



#### 【成员】

成员名称	描述
s32Id	算法库实例的 Id。
acLibName	标识算法库名称的字符数组。

#### 【注意事项】

库的名字 acLibName，用以区分不同的算法库；库的 s32Id，用以支持运行同一个算法库的多个实例。

#### 【相关数据类型及接口】

无

## ISP\_BIND\_ATTR\_S

#### 【说明】

定义 ISP 库与 Sensor、3A 库之间绑定关系的结构体。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
SensorId	Sensor 的 Id。
stAeLib	AE 库结构体。
stAfLib	AF 库结构体。
stAwbLib	AWB 库结构体。

#### 【注意事项】

无

#### 【相关数据类型及接口】

无



## ISP\_CTRL\_PROC\_WRITE\_S

### 【说明】

定义 ISP 的 PROC 信息。

### 【定义】

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

### 【成员】

成员名称	描述
pcProcBuff	ISP 传给当前算法的 Proc 信息 Buffer 指针。
u32BuffLen	ISP 传给当前算法的 Proc 信息 Buffer 当前剩余字节数。Buffer 总大小为 10K 字节。
u32WriteLen	当前算法传给 ISP 的 Proc 信息字节数。

### 【注意事项】

用户使用自己的 3A 算法，并需要支持 3A 算法的 proc 信息功能时，才需要关注此接口。

### 【相关数据类型及接口】

无

## ISP\_CTRL\_CMD\_E

### 【说明】

定义 ISP 对 3A 的控制命令。

### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
ISP_WDR_MODE_SET	设置 WDR 模式，将 ISP 控制单元的 WDR 模式配置到算法模块，此命令对应的参数数据类型是 <a href="#">WDR_MODE_E</a> 。
ISP_PROC_WRITE	设置写 PROC 信息，将算法模块的 PROC 信息配置到 ISP 控制单元，此命令对应的参数数据类型是 <a href="#">ISP_CTRL_PROC_WRITE_S</a> 。
ISP_AE_FPS_BASE_SET	设置帧率，将 ISP 控制单元的帧率信息配置到 AE 算法模块，此命令对应的参数与 <a href="#">ISP_PUB_ATTR_S</a> 里面的 f32FrameRate 一样。
ISP_AWB_ISO_SET	设置 ISO 值，将 AE 当前的 ISO 值配置到 AWB 模块，用于自动调整饱和度，此命令对应的参数与 <a href="#">ISP_AE_RESULT_S</a> 里面的 u32Iso 一样。
ISP_CHANGE_IMAGE_MODE_SET	设置图像分辨率切换标识，将 ISP 控制单元的图像分辨率标识配置到算法模块，此命令对应的参数数据类型为 HI_U8，参数值为 0 表示图像分辨率未切换，其他值表示图像分辨率已切换。
ISP_DCFINFO_GET	设置 DCF 信息，将算法模块的 DCF 信息配置到 ISP 控制单元，此命令对应的参数数据类型是 hi_comm_video.h 文件里面的 <a href="#">ISP_DCF_INFO_S</a> 。
ISP_AWB_INTTIME_SET	设置曝光量值，将 AE 当前的曝光量值配置到 AWB 模块，用于室内外检测，此命令对应的参数与 <a href="#">ISP_AE_RESULT_S</a> 里面的 u32IntTimeUs 一样。

#### 【注意事项】

无

#### 【相关数据类型及接口】

无

## ISP\_AE\_REGISTER\_S

#### 【说明】





定义 AE 注册结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
stAeExpFunc	AE 注册的回调函数结构体。

#### 【注意事项】

封装的目的是为了扩展。

#### 【相关数据类型及接口】

无

## ISP\_AE\_EXP\_FUNC\_S

#### 【说明】

定义 AE 回调函数结构体。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
pfn_ae_init	初始化 AE 的回调函数指针。
pfn_ae_run	运行 AE 的回调函数指针。



成员名称	描述
pfn_ae_ctrl	控制 AE 内部状态的回调函数指针。
pfn_ae_exit	销毁 AE 的回调函数指针。

#### 【注意事项】

- 调用 [HI\\_MPI\\_ISP\\_Init](#) 时将调用 pfn\_ae\_init 回调函数，以初始化 AE 算法库。
- 调用 [HI\\_MPI\\_ISP\\_Run](#) 时将调用 pfn\_ae\_run 回调函数，以运行 AE 算法库，计算得到 sensor 的曝光时间和增益、ISP 的数字增益。
- 设计思路中，算法库实现 ctrl 接口用以改变内部运行状态，ctrl 接口提供一个参数传输命令，提供一个 VOID 类型的指针传输数据。ctrl 接口一方面以回调函数指针的形式注册给 ISP 库，ISP 控制单元隐式调用一些命令控制算法库内部运行状态，另一方面，以算法库的用户接口的形式，从而用户可以改变算法库内部运行状态。示例：

```
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;
}
```

- 运行时 ISP 控制单元会隐式调用 pfn\_ae\_ctrl 回调函数，通知 AE 算法库切换 WDR 和线性模式、设置 FPS、通知配置 sensor。

当前 Firmware 定义的 ctrl 命令有：

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

- 调用 [HI\\_MPI\\_ISP\\_Exit](#) 时将调用 pfn\_ae\_exit 回调函数，以销毁 AE 算法库。
- 一个算法库支持初始化和运行多个实例，参数 s32Handle 以区分不同的算法库实例。如果需要在支持多个实例，可以用不同的 stAlgLib.s32Id 注册多次算法库。例如：

```
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);
```

#### 【相关数据类型及接口】

#### ISP\_AE\_REGISTER\_S

### ISP\_AE\_PARAM\_S

#### 【说明】

定义 ISP 提供给 AE 的初始化参数结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
SensorId	向 ISP 注册的 sensor 的 id，用以检查向 ISP 注册的 sensor 和向 AE 注册的 sensor 是否一致。
u8WDRMode	宽动态模式，ISP 向 AE 提供宽动态模式信息。
f32Fps	帧率，ISP 向 AE 提供帧率信息。

#### 【注意事项】

无

#### 【相关数据类型及接口】

#### ISP\_AE\_EXP\_FUNC\_S

### ISP\_AE\_INFO\_S

#### 【说明】

定义 ISP 提供给 AE 的统计信息结构体。

#### 【定义】

```
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;

```

**【成员】**

成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAeStat1	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。 Hi3518EV200 不支持。
	au16MeteringHist	五段直方图的统计信息数组，取值范围为[0, 0xFFFF], au16MeteringHist[0]表示第一段，au16MeteringHist[1]表示第二段，au16MeteringHist[2]表示第四段，au16MeteringHist[3]表示第五段。其中第三段统计信息值为 0xFFFF-(au16MeteringHist[0]+au16MeteringHist[1]+au16MeteringHist[2]+au16MeteringHist[3]) Hi3518EV200 不支持。
pstAeStat2	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。 Hi3518EV200 不支持。
	au16MeteringMemArray	五段直方图的分区间的统计信息数组，取值范围为[0, 0xFFFF]。 Hi3518EV200 不支持。
pstAeStat3	au32HistogramMemArray	256 段直方图的统计信息数组，取值范围为[0, 0xFFFFFFFF]。
	u32PixelCount	不计权重的 256 段直方图统计像素总数。
pstAeStat4	u16GlobalAvgR	全局 R 分量平均值。 取值范围为[0, 0xFF]。
	u16GlobalAvgGr	全局 Gr 分量平均值。 取值范围为[0, 0xFF]。
	u16GlobalAvgGb	全局 Gb 分量平均值。 取值范围为[0, 0xFF]。



成员名称	子成员名称	描述
	u16GlobalAvgB	全局 B 分量平均值。 取值范围为[0, 0xFF]。
pstAeStat5	au16ZoneAvg	分区间 R、Gr、Gb、B 分量平均值。 取值范围为[0, 0xFF]。

## 【注意事项】

- AE 库可以根据 u32FrameCnt 控制运算频率，例如两帧运算一次。
- pstAeStat1 和 pstAeStat2 分别表示根据直方图分割门限得到的归一化全局和分区间五段直方图统计信息，取值范围[0, 0xFFFF]。以全局五段直方图为例，如图像中的所有像素都大于最大门限值，那么第五段直方图数据即为 0xFFFF，其他 4 段直方图数据为 0。全局五段直方图会受分区间权重影响，与该模块在 ISP pipeline 的位置无关。Hi3518EV200 不支持全局和分区间五段直方图统计。
- pstAeStat3 表示全局 256 段直方图统计信息。该统计信息是取输入数据流中的高 8bit 数据统计得到的，每个 bin 中数据表示该灰度值对应的像素个数。256 个 bin 的数据之和即为参与统计的像素点个数，由寄存器 0x205a2014 决定，如图 2-5 所示。一旦寄存器 0x205a2014 的值是确定的，那么 256 个 bin 的数据之和也是确定的。目前，海思 AE 算法默认只用了 Gr 通道的统计信息，在大面积红色时，会采用 R 和 Gb 通道的统计信息，在大面积蓝色时，会采用 B 和 Gr 通道的统计信息。全局 256 段直方图会受到分区间权重的影响。

图2-5 寄存器 0x205a2014 的地址描述

Addr <sup>①</sup>	Mode <sup>②</sup>	31~8 <sup>③</sup>	7 <sup>④</sup>	6 <sup>⑤</sup>	5 <sup>⑥</sup>	4 <sup>⑦</sup>	3 <sup>⑧</sup>	2 <sup>⑨</sup>	1 <sup>⑩</sup>	0 <sup>⑪</sup>	Default <sup>⑫</sup>
0x2014 <sup>⑬</sup>	R/W <sup>⑭</sup>		Offset y <sup>⑮</sup>	Skip y <sup>⑯</sup>			Offset x <sup>⑰</sup>	Skip x <sup>⑱</sup>			0x00000000 <sup>⑲</sup>

Skip x[2:0] Histogram decimation in horizontal direction: 0=every 2nd pixel; 1=every 3rd pixel; ①  
2=every 4th pixel; 3=every 5th pixel; 4=every 8th pixel; 5+=every 9th pixel;②

Offset x 0=start from the first column; 1=start from second column;③

Skip y[2:0] Histogram decimation in vertical direction: 0=every pixel; 1=every 2nd pixel; ④  
2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; ⑤  
6+=every 9th pixel;⑥

Offset y 0=start from the first row; 1=start from second row;⑦

- pstAeStat4 表示全局 R/Gr/Gb/B 4 分量取高 8bit 统计得到的均值，取值范围[0, 0xFF]。全局 4 分量平均值会受分区间权重影响。
- pstAeStat5 表示 15\*17 区间每个区间 R/Gr/Gb/B 4 分量取高 8bit 统计得到的均值，取值范围[0, 0xFF]。
- AE 统计模块可将输入数据开方后再进行统计，通过配置寄存器 0x205a2040 第 8bit 值可选择对输入数据开方与否。所谓数据开方，指的是对输入数据归一化至 1 后做开方处理。以 256 段直方图统计为例，若输入数据为 12bit，某个像素值为 2048，若数据开方关闭，直接取高 8bit 数据进行统计，为灰度值 128 对应的 bin



像素个数加 1；若数据开方便能，像素值 2048 归一化至 1 后为 0.5，0.5 开方为 0.707，用 8bit 表示 0.707 为 181，此时在直方图上表现为灰度值 181 对应的 bin 像素个数加 1。由此可见，像素值较小的数据开方处理后像素值明显变大，相当于统计信息通过压缩亮区的统计精度来提升暗区的统计精度。建议在 WDR 模式下数据开方便能，线性模式下数据开方关闭。此外，AE 统计模块在 ISP pipeline 的位置可以变化，具体可参考第八章统计信息部分相关描述。

#### 【相关数据类型及接口】

#### ISP\_AE\_EXP\_FUNC\_S

### ISP\_AE\_RESULT\_S

#### 【说明】

定义 AE 库返回给 ISP 的配置寄存器结构体。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	子成员名称	描述
au32IntTime	-	AE 计算得出的曝光时间，以 us 为单位，将曝光时间由行数转换成 us 时需要考虑 cmos.c 中 f32Offset 的影响。线性模式和 sensor built-in WDR 模式下，只有 au32IntTime[0]有效，au32IntTime[1:3]建议配置等于 au32IntTime[0]；N 帧合成 WDR 模式下，au32IntTime[0:(N-1)]有效，配置值由小到大，依次表示最短到最长的曝光时间，用于计算长短帧曝光比，au32IntTime[(N-1):3]建议配置等于 au32IntTime[(N-1)]。au32IntTime[0]还需传递至其他模块用于与曝光时间相关的联动控制，会影响海思 AWB 效果。使用海思 AWB 算法和多帧 WDR 模式时必须配置该结构体。
u32IspDgain	-	ISP 的数字增益，8bit 精度。使用 ISP 数字增益时必须配置；不使用时配置为 0x100。



成员名称	子成员名称	描述
u32Iso	-	AE 计算得出的总增益值，ISO 表示系统增益，以常数 100 乘以倍数为单位,例如系统中 sensor 的增益为 2 倍，ISP 的增益为 1 倍，那么整个系统的 ISO 值计算方式为： $2*1*100=200$ ，即系统 ISO 为 200，本文档中涉及到的 ISO 都是采用这种计算方法。此变量影响 ISP 的去噪，sharpen 等自适应效果，必须配置。
u8AERunInterval	-	AE 算法运行的间隔，取值范围为[1,255]，取值为 1 时表示每帧都运行 AE 算法，取值为 2 时表示每 2 帧运行 1 次 AE 算法，依此类推。建议该值设置不要大于 2，否则 AE 调节速度会受到影响。WDR 模式时，该值建议设置为 1，这样 AE 收敛会更加平滑。此变量决定 AE 计算结果配置到 sensor 和 ISP 寄存器的间隔帧数，必须配置。
bPirisValid	-	Piris 是否有效的标志，取值为 HI_TRUE 时在内核态回调 Piris 驱动配置步进电机的位置，取值为 HI_FALSE 时不回调。使用海思 AE 算法和海思 Piris 驱动对接 Piris 镜头时，该值须置为 HI_TRUE，对接非 Piris 镜头时该值须置为 HI_FALSE。
s32PirisPos	-	Piris 步进电机的位置，取值范围与具体 Piris 镜头相关。使用海思 Piris 驱动对接 Piris 镜头时，该值必须配置。
u32PirisGain	-	Piris 光圈等效增益，取值范围与具体 Piris 镜头相关。可用于计算 Piris 工作时的等效曝光量，供其他模块参考。取值范围为[0, 1024]。对接非 Piris 镜头时建议将该值配置为 512。
enFSWDRMode	-	WDR 合成模式，0 表示普通多帧合成 WDR 模式，1 表示长帧模式。 Hi3518EV200 不支持。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。 Hi3518EV200 不支持。
	au8WeightTable	15x17 个区间的 AE 权重表，取值范围为[0, 255]。

## 【注意事项】



- ISP 基本算法模块将会根据 AE 计算得出的总增益值调节配置参数，例如锐化、去噪等。
- 五段直方图分割门限值和 AE 权重表在 ISP 库中有默认值，建议用户可以根据实际 sensor 配置最佳的五段直方图分割门限值。这些寄存器并不需要频繁配置，bChange 参数标识本次计算返回的值是否需要配置寄存器。Hi3518EV200 不支持五段直方图。
- 将曝光时间 au32IntTime 由行转换成 us 时，可以通过 cmos.c 中的 u32LinesPer500ms 进行，转换关系如下：  
$$\text{au32IntTime}[0] = (((\text{HI\_U64})\text{au32IntTimeRst}[0] * 1024 - \text{u32Offset}) * 500000 / \text{pstAeSnsDft} \rightarrow \text{u32LinesPer500ms}) \gg 10$$
  
上式中 au32IntTimeRst[0] 是以行数为单位的曝光时间，u32Offset=f32Offset \* 1024，f32Offset 即为曝光时间的偏移量，详见 [AE\\_ACCURACY\\_S](#) 的描述。

#### 【相关数据类型及接口】

[ISP\\_AE\\_EXP\\_FUNC\\_S](#)

## ISP\_AWB\_REGISTER\_S

#### 【说明】

定义 AWB 注册结构体。

#### 【定义】

```
typedef struct hiISP_AWB_REGISTER_S
{
    ISP\_AWB\_EXP\_FUNC\_S stAwbExpFunc;
} ISP_AWB_REGISTER_S;
```

#### 【成员】

成员名称	描述
stAwbExpFunc	AWB 注册的回调函数结构体。

#### 【注意事项】

封装的目的是为了扩展。

#### 【相关数据类型及接口】

[ISP\\_AWB\\_EXP\\_FUNC\\_S](#)

## ISP\_AWB\_EXP\_FUNC\_S

#### 【说明】

定义 AWB 回调函数结构体。

#### 【定义】





```
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;
```

#### 【成员】

成员名称	描述
pfn_awb_init	初始化 AWB 的回调函数指针。
pfn_awb_run	运行 AWB 的回调函数指针。
pfn_awb_ctrl	控制 AWB 内部状态的回调函数指针。
pfn_awb_exit	销毁 AWB 的回调函数指针。

#### 【注意事项】

- 调用 [HI\\_MPI\\_ISP\\_Init](#) 时将调用 pfn\_awb\_init 回调函数，以初始化 AWB 算法库。
- 调用 [HI\\_MPI\\_ISP\\_Run](#) 时将调用 pfn\_awb\_run 回调函数，以运行 AWB 算法库，计算得到白平衡增益、色彩校正矩阵。
- 运行时 ISP 控制单元会隐式调用 pfn\_awb\_ctrl 回调函数，通知 AWB 算法库切换 WDR 和线性模式、设置 ISO 和曝光时间（曝光时间单位是 us）。设置 ISO 的目的是为了实现 ISO 与饱和度的联动，增益大时色度噪声也会比较大，所以需要调节饱和度。设置曝光时间是为了辅助室内外判断。

当前 Firmware 定义的 ctrl 命令有：

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

- 调用 [HI\\_MPI\\_ISP\\_Exit](#) 时将调用 pfn\_awb\_exit 回调函数，以销毁 AWB 算法库。

#### 【相关数据类型及接口】

[ISP\\_AWB\\_REGISTER\\_S](#)



## ISP\_AWB\_PARAM\_S

### 【说明】

定义 ISP 提供给 AWB 的初始化参数结构体。

### 【定义】

```
typedef struct hiISP_AWB_PARAM_S
{
    SENSOR_ID SensorId;
    HI_U8 u8WDRMode;
    HI_S32 s32Rsv;
} ISP_AWB_PARAM_S;
```

### 【成员】

成员名称	描述
SensorId	向 ISP 注册的 sensor 的 id，用以检查向 ISP 注册的 sensor 和向 AWB 注册的 sensor 是否一致。
u8WDRMode	宽动态模式，ISP 向 AWB 提供宽动态模式信息。
s32Rsv	保留参数。

### 【注意事项】

无

### 【相关数据类型及接口】

[ISP\\_AWB\\_EXP\\_FUNC\\_S](#)

## ISP\_AWB\_INFO\_S

### 【说明】

定义 ISP 提供给 AWB 的统计信息结构体。

### 【定义】

```
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;
```

### 【成员】



成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat1	u16MeteringAwbRg	RGB 域统计信息中白点的 G 和 R 的平均值的比值，取值范围为[0, 0xFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringAwbBg	RGB 域统计信息中白点的 G 和 B 的平均值的比值，取值范围为[0, 0xFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
	u32MeteringAwbSum	RGB 域统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
pstAwbStat2	au16MeteringMemArrayRg	RGB 域分区间的统计信息中白点的 G 和 R 的平均值的比值，取值范围为[0, 0xFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
	au16MeteringMemArrayBg	RGB 域分区间的统计信息中白点的 G 和 B 的平均值的比值，取值范围为[0, 0xFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
	au16MeteringMemArraySum	RGB 域分区间的统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。Hi3518E 不支持 RGB 域 AWB 统计。
pstAwbStat3	u16MeteringAwbAvgR	Bayer 域全局统计信息中白点的 R 分量平均值。取值范围为[0, 0xFFF]。
	u16MeteringAwbAvgG	Bayer 域全局统计信息中白点的 G 分量平均值。取值范围为[0, 0xFFF]。
	u16MeteringAwbAvgB	Bayer 域全局统计信息中白点的 B 分量平均值。取值范围为[0, 0xFFF]。
	u16MeteringAwbCountAll	Bayer 域全局统计信息中白点的个数。已做归一化，取值范围为[0, 0xFFFF]。
	u16MeteringAwbCountMin	Bayer 域全局统计信息中小于 BlackLevel 的像素个数。已做归一



成员名称	子成员名称	描述
		化，取值范围为[0, 0xFFFF]。
	u16MeteringAwbCountMax	Bayer 域全局统计信息中大于 WhiteLevel 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。
pstAwbStat4	au16MeteringMemArrayAvgR	Bayer 域分区间统计信息中白点的 R 分量平均值。取值范围为[0, 0xFFF]。
	au16MeteringMemArrayAvgG	Bayer 域分区间统计信息中白点的 G 分量平均值。取值范围为[0, 0xFFF]。
	au16MeteringMemArrayAvgB	Bayer 域分区间统计信息中白点的 B 分量平均值。取值范围为[0, 0xFFF]。
	au16MeteringMemArrayCountAll	Bayer 域分区间统计信息中白点的个数。已做归一化，取值范围为 [0, 0xFFFF]。
	au16MeteringMemArrayCountMin	Bayer 域分区间统计信息中小于 BlackLevel 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。
	au16MeteringMemArrayCountMax	Bayer 域分区间统计信息中大于 WhiteLevel 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。

#### 【注意事项】

- AWB 库可以根据 u32FrameCnt 控制运算频率，例如两帧运算一次。
- Rg、Bg、Sum 的意义请参考 AWB 章节的描述。
- Hi3518EV200 只支持 Bayer 域统计信息。统计信息分两种形式提供，第一种是 Bayer 域全局的统计信息，第二种是 Bayer 域 15x17 个区间的统计信息。
- Hi3518EV200 AWB 统计模块在 BLC 模块后，因此输出的统计结果不带黑电平，RGB 均值输出的位宽是 12bit。

#### 【相关数据类型及接口】

[ISP\\_AWB\\_EXP\\_FUNC\\_S](#)

## ISP\_AWB\_RESULT\_S

#### 【说明】

定义 AWB 库返回给 ISP 的配置寄存器结构体。

#### 【定义】



```
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;
```

**【成员】**

成员名称	子成员名称	描述
au32WhiteBalanceGain	-	白平衡算法得出的 R、Gr、Gb、B 颜色通道的增益，16bit 精度表示。
au16ColorMatrix	-	色彩还原矩阵，8bit 精度表示。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringWhiteLevelAwb	RGB 域统计白点信息时，找白点的亮度上限。取值范围[0x0, 0x3FF]，默认值 0x3ac。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringBlackLevelAwb	RGB 域统计白点信息时，找白点的亮度下限。取值范围 [0x0, 0x3FF]，默认值 0x40。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCrRefMaxAwb	RGB 域统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCbRefMaxAwb	RGB 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCrRefMinAwb	RGB 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCbRefMinAwb	RGB 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默

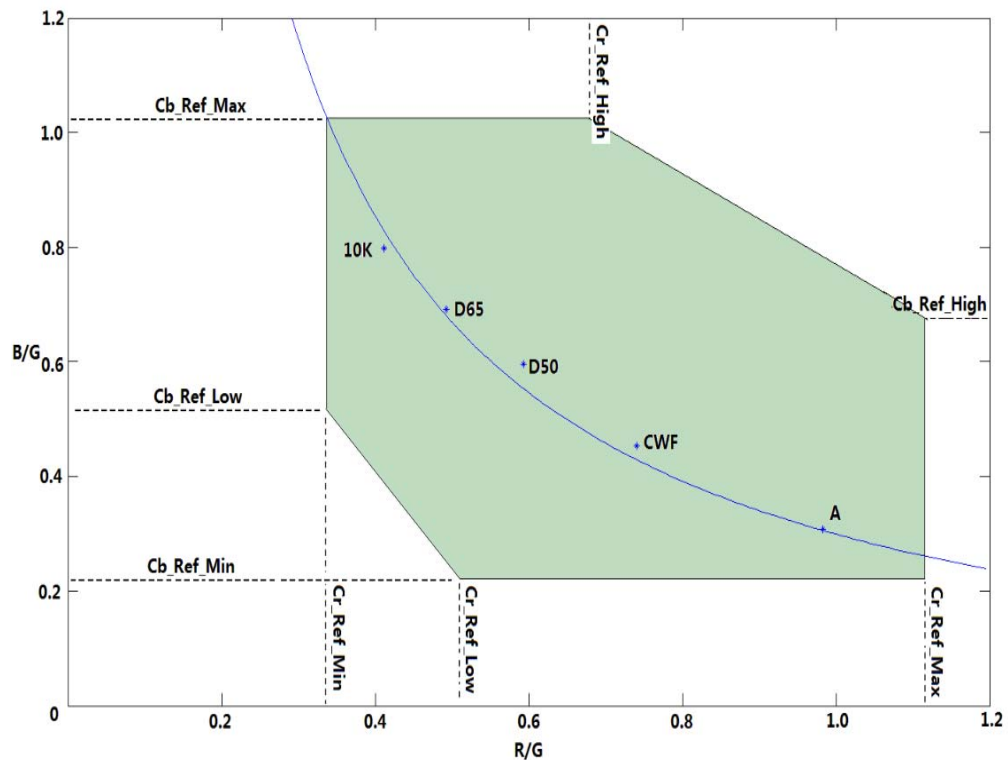


成员名称	子成员名称	描述
		认值 128。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCrRefHighAwb	RGB 域统计白点信息时，六边形白点区域限制 CbMax 对应的 Cr 值，8bit 精度，默认值 512。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCrRefLowAwb	RGB 域统计白点信息时，六边形白点区域限制 CbMin 对应的 Cr 值，8bit 精度，默认值 128。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCbRefHighAwb	RGB 域统计白点信息时，六边形白点区域限制 CrMax 对应的 Cb 值，8bit 精度，默认值 512。Hi3518E 不支持 RGB 域 AWB 统计。
	u16MeteringCbRefLowAwb	RGB 域统计白点信息时，六边形白点区域限制 CrMin 对应的 Cb 值，8bit 精度，默认值 128。Hi3518E 不支持 RGB 域 AWB 统计。
stRawStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	bAboveWhiteLevelClip	大于亮度上限的点是否参与白点统计。Hi3518E 不支持该参数配置。
	bBelowBlackLevelClip	小于亮度下限的点是否参与白点统计。Hi3518E 不支持该参数配置。
	u16MeteringWhiteLevelAwb	Bayer 域统计白点信息时，找白点的亮度上限。取值范围 [0x0, 0xFFFF]，默认值 0xFFFF。
	u16MeteringBlackLevelAwb	Bayer 域统计白点信息时，找白点的亮度下限。取值范围 [0x0, 0xFFFF]，默认值 0x0。
	u16MeteringCrRefMaxAwb	Bayer 域统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。



成员名称	子成员名称	描述
	u16MeteringCbRefMaxAwb	Bayer 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。
	u16MeteringCrRefMinAwb	Bayer 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。
	u16MeteringCbRefMinAwb	Bayer 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。
	u16MeteringCrRefHighAwb	Bayer 域统计白点信息时，六边形白点区域限制 CbMax 对应的 Cr 值，8bit 精度，默认值 512。
	u16MeteringCrRefLowAwb	Bayer 域统计白点信息时，六边形白点区域限制 CbMin 对应的 Cr 值，8bit 精度，默认值 128。
	u16MeteringCbRefHighAwb	Bayer 域统计白点信息时，六边形白点区域限制 CrMax 对应的 Cb 值，8bit 精度，默认值 512。
	u16MeteringCbRefLowAwb	Bayer 域统计白点信息时，六边形白点区域限制 CrMin 对应的 Cb 值，8bit 精度，默认值 128。

图2-6 白色区域选择相关参数



#### 【注意事项】

- AWB 算法首先将会计算出 R、Gr、Gb、B 颜色通道的增益，以校正出白色，16bit 精度表明最后 16 位为小数。
- Hi3518EV200 WDR 模式，AWB 统计结果和增益不需要做特别处理，和 Linear 模式一致。
- AWB 算法其次会计算一个 3x3 的颜色校正矩阵，以还原出真实的色彩，8bit 精度表明最后 8 位为小数。
- stStatAttr 结构体中的信息决定什么样的像素点被认为是白点，从而参与统计。用户开发新的 AWB 算法时可以使用默认值，也可以自定义配置，bChange 标识表明运行时当前帧是否需要配置 stStatAttr 结构体中的值到寄存器。
- 环境色温、照度会影响白点的分布范围，海思 AWB 算法在运行过程中，会根据环境参数自动刷新 AWB Byaer 域统计信息的白点参数。用户希望在海思 AWB 算法运行时修改 AWB 统计参数，需要调用 [HI\\_MPI\\_ISP\\_SetWBAttr\(\)](#) 接口，关闭统计参数自动刷新功能。用户关闭统计参数自动刷新功能，到 ISP 收到配置并响应，有 2 帧的延时。因此，用户设置后需要等待 2 帧时间再修改 AWB 统计参数。
- Hi3518EV200 只支持 Bayer 域统计信息。
- u16MeteringCrRefHighAwb、u16MeteringCrRefLowAwb、u16MeteringCbRefHighAwb、u16MeteringCbRefLowAwb 这四个参数 18EV200 不支持。

#### 【相关数据类型及接口】





## ISP\_AWB\_EXP\_FUNC\_S

### ISP\_AF\_REGISTER\_S

#### 【说明】

定义 AF 注册结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
stAfExpFunc	AF 注册的回调函数结构体。

#### 【注意事项】

封装的目的是为了扩展。

#### 【相关数据类型及接口】

## ISP\_AF\_EXP\_FUNC\_S

### ISP\_AF\_EXP\_FUNC\_S

#### 【说明】

定义 AF 回调函数结构体。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
pfn_af_init	初始化 AF 的回调函数指针。
pfn_af_run	运行 AF 的回调函数指针。
pfn_af_ctrl	控制 AF 内部状态的回调函数指针。
pfn_af_exit	销毁 AF 的回调函数指针。

【注意事项】

AF 库暂未实现。

【相关数据类型及接口】

[ISP\\_AF\\_REGISTER\\_S](#)

## ISP\_AF\_PARAM\_S

【说明】

定义 ISP 提供给 AF 的初始化参数结构体。

【定义】

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

【成员】

成员名称	描述
SensorId	向 ISP 注册的 sensor 的 id，用以检查向 ISP 注册的 sensor 和向 AF 注册的 sensor 是否一致。
s32Rsv	保留参数。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_AF\\_EXP\\_FUNC\\_S](#)

## ISP\_AF\_INFO\_S

【说明】



定义 ISP 提供给 AF 的统计信息结构体。

【定义】

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

【成员】

成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
stAfStat	stZoneMetrics. u16v1	分区间统计的 AF 垂直方向奇数列 FIR 滤波器的统计值。
	stZoneMetrics. u16h1	分区间统计的 AF 水平方向奇数行 IIR 滤波器的统计值。
	stZoneMetrics. u16v2	分区间统计的 AF 垂直方向偶数列 FIR 滤波器的统计值。
	stZoneMetrics. u16h2	分区间统计的 AF 水平方向偶数行 IIR 滤波器的统计值。
	stZoneMetrics. u16y	分区间统计的 AF 亮度统计值，区块中每个像素点的亮度累加值。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_AF\\_EXP\\_FUNC\\_S](#)

## ISP\_AF\_RESULT\_S

【说明】

定义 AF 库返回给 ISP 的配置寄存器结构体。

【定义】

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



【成员】

成员名称	描述
s32Rsv	保留参数。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_AF\\_EXP\\_FUNC\\_S](#)

## ISP\_DCF\_INFO\_S

【说明】

定义 DCF 信息参数结构体。

【定义】

```
#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;
```



}

## 【成员】

成员名称	描述
au8ImageDescription	图像描述、来源，指生成图像的工具。 数据格式：ascii strings，长度最大 32。
au8Make	生产者，指产品生产厂家。 数据格式：ascii strings，长度最大 32。
au8Model	型号，指设备型号。 数据格式：ascii strings，长度最大 32。
au8Software	软件，显示固件 Firmware 版本。 数据格式：ascii strings，长度最大 32。
u16ISOSpeedRatings	感光度。 数据格式：unsigned short。 只读。
u32FNumber	光圈系数。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32MaxApertureValue	镜头的最大光圈值。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32ExposureTime	曝光时间 (快门速度的倒数)，单位是秒。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32ExposureBiasValue	照片拍摄时的曝光补偿，单位是 APEX(EV)。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母，Hi3518EV200 不支持。 只读。
u8ExposureProgram	拍照时相机使用的曝光程序。 1：手动曝光； 2：正常程序曝光； 3：光圈优先曝光； 4：快门优先曝光；



成员名称	描述
	5: 创意程序(慢速程序); 6: 动作程序(高速程序); 7: 肖像模式; 8: 风景模式。 数据格式: unsigned short。 只读。
u8MeteringMode	曝光的测光模式。 0: 未知; 1: 平均测光; 2: 中央重点测光; 3: 点测光; 4: 多点测光; 5: 多区域测光; 6: 部分测光; 255: 是其他。 数据格式: unsigned short。 只读。
u8LightSource	光源, 表示白平衡设置。 0: 未知。 1: 日光; 2: 荧光灯; 3: 白炽灯(钨丝); 10: 闪光灯; 17: 标准光 A; 18: 标准光 B; 19: 标准光 C; 20: D55; 21: D65; 22: D75; 255: 其他。 数据格式: unsigned short。
u32FocalLength	拍摄照片时的镜头的焦距长度, 单位是毫米。数据格式: unsigned rational, 高 16 位为分子, 低 16 位为分母。
u8SceneType	表示拍摄场景的类型, 值 '0x01' 表示图像是指通过相机直接拍摄。



成员名称	描述
	数据格式: unsigned short。 暂不支持。
u8CustomRendered	自定义图像处理。0: 标准; 1: 自定义。
u8ExposureMode	曝光模式。 0: 自动曝光; 1: 手动曝光; 2: 自动包围曝光。 数据格式: unsigned short。 只读。
u8WhiteBalance	白平衡。 0: 自动白平衡; 1: 手动白平衡。 数据格式: unsigned short。 只读。
u8FocalLengthIn35mmFilm	35 毫米胶片焦距, 0: 表示这个焦距不存在。 数据格式: unsigned short。
u8SceneCaptureType	场景拍摄类型。 0: 标准; 1: 风景模式; 2: 肖像模式; 3: 夜间模式。 数据格式: unsigned short。
u8GainControl	增益控制。 0: None; 1: Low gain up; 2 : High gain up; 3: Low gain down; 4: High gain down。 数据格式: unsigned short。
u8Contrast	对比度。 数据格式: unsigned short。
u8Saturation	饱和度。 0: 无; 1: 低;



成员名称	描述
	2: 高。 数据格式: unsigned short。
u8Sharpness	锐度。 数据格式: unsigned short。

【注意事项】

无

【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetDCFInfo](#)
- [HI\\_MPI\\_ISP\\_GetDCFInfo](#)

## ISP\_MOD\_PARAM\_S

【说明】

定义 ISP 模块参数结构体。

【定义】

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

【成员】

成员名称	描述
proc_param	ISP 的 PROC 信息更新频率。 最小值为 0，无上限。 proc_param 为 n 表示每隔 n 帧更新一次 ISP 的 PROC 信息。

【注意事项】

- 加载 ko 时设置的 ISP 模块参数 proc\_param 如果为 0，则不分配存储 ISP Proc 信息的内存；如果为非 0，则分配存储 ISP Proc 信息的内存。
- 当加载 ko 时设置的 ISP 模块参数 proc\_param=0 时，通过接口 [HI\\_MPI\\_ISP\\_SetModParam](#) 设置的 proc\_param 不能设置为非 0 值。
- 当加载 ko 时设置的 ISP 模块参数 proc\_param 为非 0 值时，通过接口 [HI\\_MPI\\_ISP\\_SetModParam](#) 设置的 proc\_param 只能在非 0 值之间动态切换，不能设置为 0。





- ISP Proc 信息频繁更新会消耗 CPU 资源。推荐设为 30 帧更新一次，或仅 Debug 时开启。
- Huawei LiteOS 没内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk\_init.c 中执行的相关过程。在 sdk\_init.c 中 ISP\_init 函数中，添加参数 stIsp\_Param.u32ProcParam = n 传入 ISP\_ModInit 中，即实现了 proc\_param 参数的设置。

#### 【相关数据类型及接口】

无

## ISP\_MODULE\_PARAM\_S

#### 【说明】

定义 ISP 模块参数结构体。该结构体只在 Huawei LiteOS 版本中存在。

#### 【定义】

```
typedef struct hiISP_MODULE_PARAM_S
{
    HI_U32  u32PwmNum;
    HI_U32  u32ProcParam;
    HI_U32  u32UpdatePos;
    HI_U32  u32LscUpdateMode;
} ISP_MODULE_PARAM_S;
```

#### 【成员】

成员名称	描述
u32PwmNum	ISP 的使用 PWM 的序号，默认为 2。 取值范围：[0x0,0x3]
u32ProcParam	ISP 的 PROC 信息更新频率。 取值范围：[0x0, 0xFFFFFFFF] u32procparam 为 n 表示每隔 n 帧更新一次 ISP 的 PROC 信息。
u32UpdatePos	表示 ISP 中断配置寄存器的位置。 u32UpdatePos=0 表示默认位置，为帧起始中断； u32UpdatePos=1 表示中断配置为 AF 统计信息完成中断。
u32LscUpdateMode	表示 lsc 表项配置位置。 0 表示在用户态配置； 1 表示在内核态 AF 统计信息完成中断配置。

#### 【注意事项】

- 本结构体只在 Huawei LiteOS 版本下使用，在 sdk\_init.c 中传入相应参数。



- Huawei LiteOS 没内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk\_init.c 中执行的相关过程。在 sdk\_init.c 中 ISP\_init 函数中，添加参数 stIsp\_Param.u32ProcParam = n 传入 ISP\_ModInit 中，即实现了 proc\_param 参数的设置。

【相关数据类型及接口】

无

## ISP\_OP\_TYPE\_E

【说明】

定义模块运行状态。

【定义】

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

【成员】

成员名称	描述
OP_TYPE_AUTO	0x0: 运行在自动模式下
OP_TYPE_MANUAL	0x1: 运行在手动模式下
OP_TYPE_BUTT	无效值

【注意事项】

无

【相关数据类型及接口】

- [ISP\\_NR\\_ATTR\\_S](#)
- [HI\\_MPI\\_ISP\\_SetExposureAttr](#)
- [HI\\_MPI\\_ISP\\_SetWBAttr](#)
- [HI\\_MPI\\_ISP\\_SetIrisAttr](#)
- [HI\\_MPI\\_ISP\\_SetSaturationAttr](#)
- [HI\\_MPI\\_ISP\\_SetCCMArr](#)
- [HI\\_MPI\\_ISP\\_SetSharpenAttr](#)
- [HI\\_MPI\\_ISP\\_SetDRCArr](#)
- [HI\\_MPI\\_ISP\\_SetMeshShadingAttr](#)
- [HI\\_MPI\\_ISP\\_SetDeFogAttr](#)



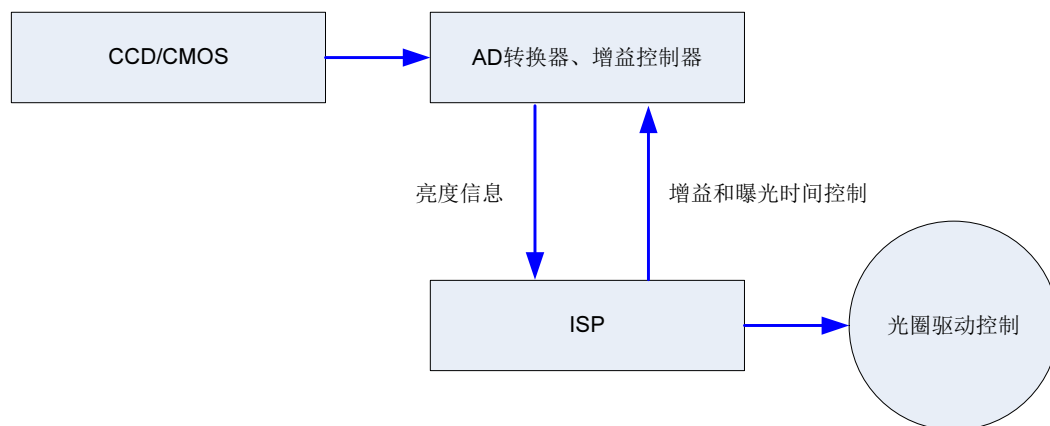
- [HI\\_MPI\\_ISP\\_SetAntiFalseColorAttr](#)
- [HI\\_MPI\\_ISP\\_SetDemosaicAttr](#)
- [HI\\_MPI\\_ISP\\_SetFPNAttr](#)
- [HI\\_MPI\\_ISP\\_SetDPDynamicAttr](#)

# 3 AE

## 3.1 概述

HiISP AE 模块实现的功能是：根据自动测光系统获得当前图像的曝光量，再自动配置镜头光圈、sensor 快门及增益来获得最佳的图像质量。自动曝光的算法主要分光圈优先、快门优先、增益优先。光圈优先时算法会优先调整光圈到合适的位置，再分配曝光时间和增益，只适合 p-iris 镜头，这样能均衡噪声和景深。快门优先时算法会优先分配曝光时间，再分配 sensor 增益和 ISP 增益，这样拍摄的图像噪声会比较小。增益优先则是优先分配 sensor 增益和 ISP 增益，再分配曝光时间，适合拍摄运动物体的场景。当前 AE 算法也支持客户设定更灵活的曝光分配策略，AE 模块的工作流程如图 3-1 所示。

图3-1 AE 模块工作流程图



## 3.2 重要概念

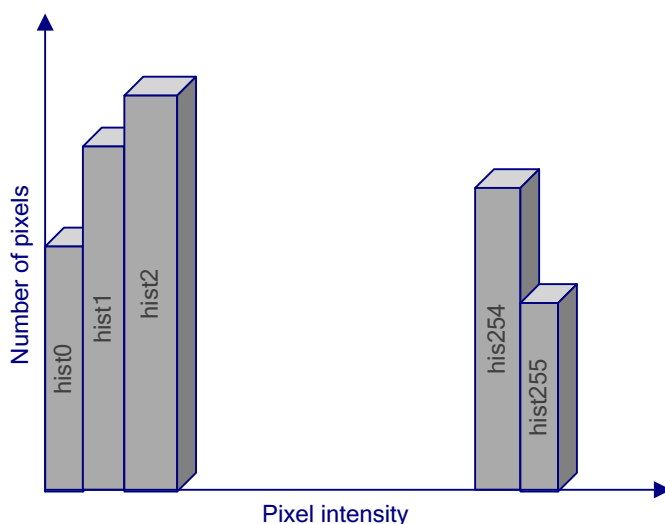
- 曝光时间：sensor 积累电荷的时间，是 sensor pixel 从开始曝光到电量被读出的这段时间。
- 曝光增益：对 sensor 的输出电荷的总的放大系数，一般有数字增益和模拟增益，模拟增益引入的噪声会稍小，所以一般优先用模拟增益。

- 光圈：光圈是镜头中可以改变中间孔大小的机械装置。
- 抗闪烁：由于电灯电源工频与 sensor 的帧率不匹配而导致的画面闪烁，一般通过限定曝光时间和修改 sensor 的帧率来达到抗闪烁的效果。

### 3.3 功能描述

AE 模块主要有 ISP 的 AE 统计信息模块及 AE 控制策略的 AE 算法 Firmware 两部分组成。ISP 的 AE 统计信息模块主要是提供 sensor 输入数据的亮度信息统计。其提供的统计信息包括直方图和平均值，可同时提供整幅图像的 256 段直方图和 R/Gr/Gb/B 四分量平均值统计信息，还可提供将整幅图像分成 MxN 区块的每个区块的 R/Gr/Gb/B 四分量平均值统计信息，具体如图 3-2 所示。

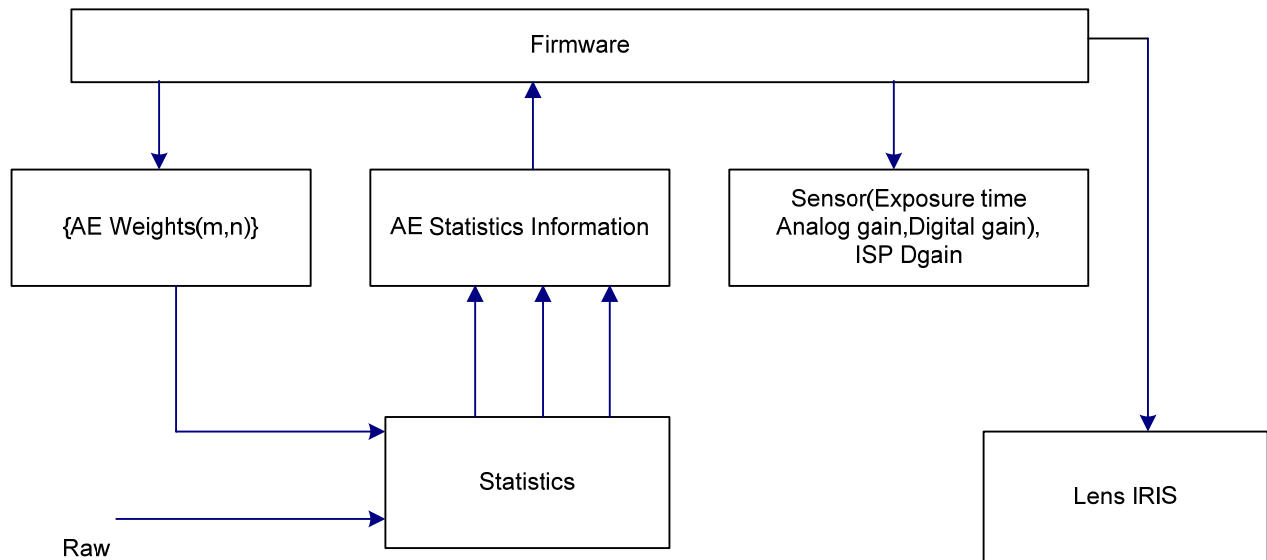
图3-2 AE 256 段统计信息直方图



AE 算法的主要工作原理是实时获取输入图像的统计信息并与设定目标亮度进行比较，而动态调节 sensor 的曝光时间和增益以及镜头光圈大小以达到实际亮度与设定目标亮度接近。其工作原理如图 3-3 所示。



图3-3 AE 工作原理图



## 3.4 API 参考

### 3.4.1 AE 库接口

所有 AE 库接口都只是针对海思 AE 库，如果客户自己实现 AE 库，不需要关注这些接口，且无法使用这些接口。

- [HI\\_MPI\\_AE\\_Register](#): 向 ISP 注册 AE 库。
- [HI\\_MPI\\_AE\\_UnRegister](#): 向 ISP 反注册 AE 库。
- [HI\\_MPI\\_AE\\_SensorRegCallBack](#): AE 库提供的 sensor 注册的回调接口。
- [HI\\_MPI\\_AE\\_SensorUnRegCallBack](#): AE 库提供的 sensor 反注册的回调接口。

#### HI\_MPI\_AE\_Register

##### 【描述】

向 ISP 注册 AE 库。

##### 【语法】

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

##### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入



参数名称	描述	输入/输出
pstAeLib	AE 算法库结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_ae.h
- 库文件：libisp.a、lib\_hiae.a

#### 【注意】

- 该接口调用了 ISP 库提供的 AE 注册回调接口 [HI\\_MPI\\_ISP\\_AELibRegCallBack](#)，以实现海思 AE 库向 ISP 库注册的功能。
- AE 库可以注册多个实例。

#### 【举例】

```
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);
```

#### 【相关主题】

无

## HI\_MPI\_AE\_UnRegister

#### 【描述】

向 ISP 反注册 AE 库。

#### 【语法】

```
HI_S32 HI_MPI_AE_UnRegister(ISP_DEV IspDev, ALG\_LIB\_S *pstAeLib);
```

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAeLib	AE 算法库结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_ae.h
- 库文件：libisp.a、lib\_hiae.a

#### 【注意】

该接口调用了 ISP 库提供的 AE 反注册回调接口 [HI\\_MPI\\_ISP\\_AELibUnRegCallBack](#)，以实现 AE 向 ISP 库反注册的功能。

#### 【举例】

```
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);
```

#### 【相关主题】

无

## HI\_MPI\_AE\_SensorRegCallBack

#### 【描述】

AE 库提供的 sensor 注册的回调接口。

#### 【语法】

```
HI_S32 HI_MPI_AE_SensorRegCallBack(ISP_DEV IspDev, ALG\_LIB\_S *pstAeLib,
SENSOR_ID SensorId, AE\_SENSOR\_REGISTER\_S *pstRegister);
```

#### 【参数】





参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAeLib	AE 算法库结构体指针。	输入
SensorId	向 AE 注册的 Sensor 的 Id。	输入
pstRegister	Sensor 注册结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

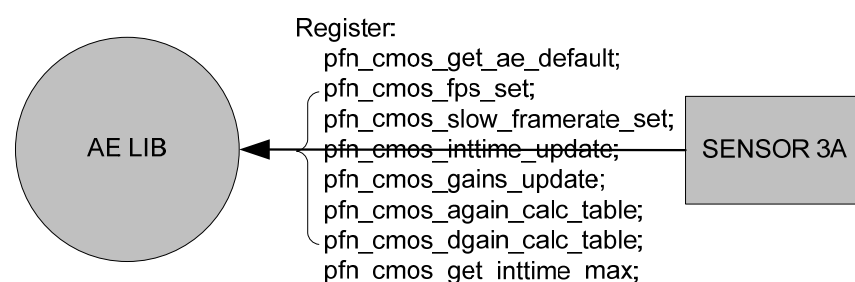
#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_ae.h
- 库文件：libisp.a、lib\_hiaa.a

#### 【注意】

- SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 注册的 sensor 和向 3A 注册的 sensor 是否为同一个 sensor。
- AE 通过 sensor 注册的一系列回调接口，获取差异化的初始化参数，并控制 sensor。

图3-4 AE 库与 sensor 库间的接口



#### 【举例】

```
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;

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;
}
```

#### 【相关主题】

无

## HI\_MPI\_AE\_SensorUnRegCallBack

#### 【描述】

AE 库提供的 sensor 反注册的回调接口。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAeLib	AE 算法库结构体指针。	输入
SensorId	向 AE 反注册的 Sensor 的 Id。	输入

#### 【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_ae.h
- 库文件：libisp.a、lib\_hiae.a

#### 【注意】

SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 反注册的 sensor 和向 3A 反注册的 sensor 是否为同一个 sensor。

#### 【举例】

```
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;
}
```

#### 【相关主题】

无

## 3.4.2 AE 控制模块

曝光控制接口：

- [HI\\_MPI\\_ISP\\_SetExposureAttr](#)：设置 AE 曝光属性。
- [HI\\_MPI\\_ISP\\_GetExposureAttr](#)：获取 AE 曝光属性。
- [HI\\_MPI\\_ISP\\_SetAERouteAttr](#)：设置 AE 路线属性。
- [HI\\_MPI\\_ISP\\_GetAERouteAttr](#)：获取 AE 路线属性。
- [HI\\_MPI\\_ISP\\_SetAERouteAttrEx](#)：设置 AE 曝光分配扩展属性，支持分别设置 AE 分配策略中的 sensor 模拟增益，sensor 数字增益和 ISP 数字增益。
- [HI\\_MPI\\_ISP\\_GetAERouteAttrEx](#)：获取 AE 曝光分配策略扩展属性。
- [HI\\_MPI\\_ISP\\_QueryExposureInfo](#)：获取 AE 内部状态信息。

### HI\_MPI\_ISP\_SetExposureAttr

#### 【描述】



设定 AE 曝光属性。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstExpAttr	AE 曝光属性结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

#### 【注意】

- AE 曝光控制类型为自动时，曝光时间，曝光增益都由 AE 算法自动控制，可以通过配置自动曝光属性结构体 stAuto 里面的参数得到不同的曝光效果。
- AE 曝光控制类型为手动时，可以通过配置手动曝光属性结构体 stManual 控制使能类型（曝光时间使能、sensor 模拟增益使能、sensor 数字增益使能、ISP 数字增益使能）及相应的曝光参数（曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益）。
- AE 曝光控制类型为自动时，配置手动曝光属性的参数无效。同理，AE 曝光控制类型为手动时，配置自动曝光属性的参数无效。
- AE 曝光控制类型为手动时，若曝光参数设置超出最大（小）值，将使用 sensor 支持的最大（小）值代替。



- 无论是自动曝光还是手动曝光，曝光时间的单位为微秒(us)，曝光增益的单位为10bit精度的倍数，即1024代表1倍，2048代表2倍等。

#### 【举例】

自动曝光属性设置：

```
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]={1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0},
                                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},
```



```
        {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);
```

手动曝光属性设置:

```
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)
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetExposureAttr

#### 【描述】

获取 AE 曝光属性。

#### 【语法】

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

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstExpAttr	AE 曝光属性结构体指针。	输出

**【返回值】**

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

**【错误码】**

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

**【需求】**

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

**【注意】**

无

**【举例】**

无

**【相关主题】**

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

## HI\_MPI\_ISP\_SetAERouteAttr

**【描述】**

设置 AE 曝光分配策略。

**【语法】**

```
HI_S32 HI_MPI_ISP_SetAERouteAttr(ISP_DEV IspDev, const ISP\_AE\_ROUTE\_S
*pstAERouteAttr);
```

**【参数】**



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAERouteAttr	AE 曝光分配策略结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

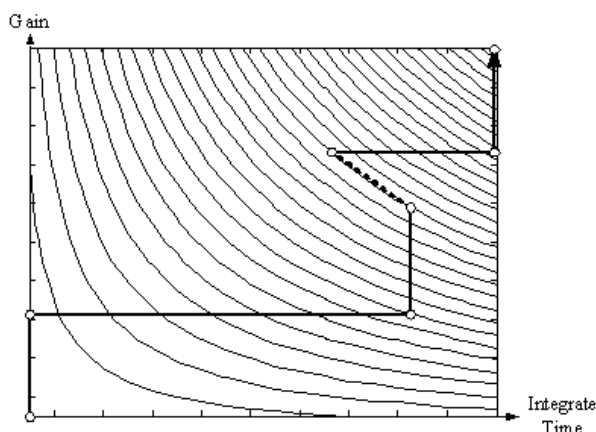
#### 【注意】

- 此接口用于设定 AE 曝光分配路线，AE 计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光优先、增益优先、光圈优先。
- AE 分配路线示意图如[图 3-5](#) 所示。AE 分配路线遵循以下限定：
  - 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含模拟增益、数字增益、ISP 数字增益。
  - 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。
  - 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
  - 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
  - 相邻节点的曝光量允许相同，这时可以由一种分配方式切换到另一种分配方式。



- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 分配策略是首先分配曝光时间，其次分配增益。针对 P-Iris 镜头，默认 AE 分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设置的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置 AE route。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了 AE route，则在切换后会采用 cmos.c 中的 route，否则采用默认的 AE route。
- 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与 WDR 模式切换、帧率或分辨率切换、限制曝光时间或增益的最大最小值等情况时，实际生效的 AE route 可能与 MPI 设置的不一致，此时可以通过 HI\_MPI\_ISP\_QueryExposureInfo 获取实际生效的 AE route。

图3-5 AE 分配路线示意图



### 【举例】

```
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);
```

### 【相关主题】



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

## HI\_MPI\_ISP\_GetAERouteAttr

### 【描述】

获取 AE 曝光分配策略。

### 【语法】

```
HI_S32 HI_MPI_ISP_GetAERouteAttr(ISP_DEV IspDev, ISP\_AE\_ROUTE\_S  
*pstAERouteAttr);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAERouteAttr	AE 曝光分配策略结构体指针。	输出

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

### 【注意】

无

### 【举例】

无



#### 【相关主题】

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

## HI\_MPI\_ISP\_SetAERouteAttrEx

#### 【描述】

设置 AE 曝光分配扩展属性，支持分别设置 AE 分配策略中的 sensor 模拟增益，sensor 数字增益和 ISP 数字增益。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetAERouteAttrEx(ISP_DEV IspDev, const  
ISP\_AE\_ROUTE\_EX\_S *pstAERouteAttrEx);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAERouteAttrEx	AE 曝光分配策略扩展属性结构体指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

#### 【注意】

- 此接口用于设定 AE 曝光分配扩展属性，AE 计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor 模拟增益优先、sensor 数字增益优先、ISP 数字增益优先和光圈优先。该接口可用于设置 WDR 模



式下的曝光分配路线，减轻正常室内照度多帧合成 WDR 产生的工频闪现象，优化 WDR 模式图像效果。

- AE 曝光分配扩展属性是否生效可通过配置 HI\_MPI\_ISP\_SetExposureAttr 接口中的 bAERouteExValid 来实现。bAERouteExValid 为 HI\_TRUE 时使用扩展 AE route，否则使用正常 AE route。
- AE 扩展分配路线遵循以下限定：
  - 最大支持 16 个节点，每个节点有曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈五个分量。
  - 节点中曝光时间的单位为 us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
  - 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。
  - 节点的曝光量是曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
  - 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如 sensor 模拟增益分量增加，那么该段路线的分配策略是 sensor 模拟增益优先。
  - 相邻节点的曝光量允许相同，这时可以由一种分配方式切换到另一种分配方式。建议不要设置等曝光量节点。
  - 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
  - 针对 DC-Iris 和手动光圈镜头，默认 AE 扩展分配策略是先分配曝光时间，再分配 sensor 模拟增益、sensor 数字增益，最后分配 ISP 数字增益。针对 P-Iris 镜头，默认 AE 扩展分配策略是先调节光圈，将光圈调至最大后调节曝光时间，最后再分配 sensor 模拟增益、sensor 数字增益和 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
  - 在线进行 DC-Iris 和 P-Iris 切换，扩展 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展 AE route。
  - 自动降帧时，最大曝光时间的改变会更新到分配路线中。
  - 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了扩展 AE route，且 cmos.c 中的扩展 AE route 标识符 bAERouteExValid 为 HI\_TRUE，则在切换后会采用 cmos.c 中的扩展 AE route，否则采用默认的 AE route。
  - 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
  - 发生自动降帧、线性与 WDR 模式切换、帧率或分辨率切换、限制曝光时间或增益的最大最小值等情况时，实际生效的扩展 AE route 可能与 MPI 设置的不一致，此时可以通过 HI\_MPI\_ISP\_QueryExposureInfo 获取实际生效的扩展 AE route。

#### 【举例】

```
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);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetAERouteAttrEx

#### 【描述】

获取 AE 曝光分配策略扩展属性。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAERouteAttrEx	AE 曝光分配策略扩展属性结构体指针。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。



【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

【注意】

无。

【举例】

无。

【相关主题】

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

## HI\_MPI\_ISP\_QueryExposureInfo

【描述】

获取 AE 内部状态信息，包括 256 段直方图和平均亮度等统计信息，同时还可获取 AE 运行状态中的曝光时间、增益和曝光量等信息。

【语法】

```
HI_S32 HI_MPI_ISP_QueryExposureInfo(ISP_DEV IspDev, ISP\_EXP\_INFO\_S
*pstExpInfo);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstExpInfo	曝光内部状态信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。



#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

#### 【注意】

- 获取的曝光时间以微秒(us)为单位，获取的 sensor 模拟增益、sensor 数字增益和 ISP 数字增益以倍数为单位，精度是 10bit。
- 获取的曝光量=(曝光时间\*曝光增益)，其中曝光时间以(ms)为单位，曝光增益包括 sensor 模拟增益、sensor 数字增益和 ISP 数字增益，以倍数为单位，6bit 小数精度。若该值的精度不足以满足需求，可以根据上面高精度的曝光时间(us)和增益(10bit 小数精度)重新计算一个曝光量。
- 可通过查询 s16HistError 来获取 AE 是否稳定的信息，若 s16HistError 的绝对值小于曝光容忍偏差值，意味着当前 AE 不会动作。
- 通过该接口获取的 AE route 与 Proc 信息中的 AE route 都是实际生效的值。只不过该接口的节点曝光时间以 us 为单位，而 Proc 信息中的曝光时间以行数为单位；该接口的光圈分量为 F 值大小，取值范围为[0,10]，而 Proc 信息中的光圈分量为 F 值等效增益，大小为(1<=F 值)。扩展 AE route 也是同样的对应关系。

#### 【举例】

```
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.ul6AE_Hist5Value[i]);  
}
```

#### 【相关主题】

无

### 3.4.3 AI 控制模块

光圈控制接口:

- [HI\\_MPI\\_ISP\\_SetIrisAttr](#): 设置光圈的属性。
- [HI\\_MPI\\_ISP\\_GetIrisAttr](#): 获取光圈的属性。
- [HI\\_MPI\\_ISP\\_SetDcirisAttr](#): 设置 DC-Iris 自动光圈控制属性。
- [HI\\_MPI\\_ISP\\_GetDcirisAttr](#): 获取 DC-Iris 自动光圈控制属性。
- [HI\\_MPI\\_ISP\\_SetPirisAttr](#): 设置 P-Iris 自动光圈控制属性。
- [HI\\_MPI\\_ISP\\_GetPirisAttr](#): 获取 P-Iris 自动光圈控制属性。

#### HI\_MPI\_ISP\_SetIrisAttr

##### 【描述】

设定光圈的属性，该函数可实现手动光圈属性和光圈类型等参数的设置。

##### 【语法】

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

##### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIrisAttr	光圈属性结构体指针。	输入

##### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

##### 【错误码】





接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

**【需求】**

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

**【注意】**

- 进行 AI 算法测试前，建议确认 AI 电路特性是否符合海思 IPC 要求。
- 根据实际对接镜头光圈类型，设置正确的光圈类型属性，由此再去设置相关的 DC-Iris/P-Iris 控制属性。若对接的是手动光圈镜头，可将光圈类型设置为 ISP\_IRIS\_DC\_TYPE，建议此时关闭 AI 使能。
- 手动光圈属性主要用于调试，可通过该 MPI 进行设置。对于 P-Iris 镜头，手动 enIrisFNO 值会受到最大、最小光圈目标值的影响。自动光圈属性的更多参数需要调用 [HI\\_MPI\\_ISP\\_SetDcirisAttr](#) 和 [HI\\_MPI\\_ISP\\_SetPirisAttr](#) 进行设置。

**【举例】**

无

**【相关主题】**

- [HI\\_MPI\\_ISP\\_GetIrisAttr](#)
- [HI\\_MPI\\_ISP\\_SetDcirisAttr](#)
- [HI\\_MPI\\_ISP\\_SetPirisAttr](#)

## HI\_MPI\_ISP\_GetIrisAttr

**【描述】**

获取光圈的控制属性。

**【语法】**

```
HI_S32 HI_MPI_ISP_GetIrisAttr(ISP_DEV IspDev, ISP\_IRIS\_ATTR\_S
*pstIrisAttr);
```

**【参数】**

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIrisAttr	光圈控制属性结构体指针。	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

【注意】

无

【举例】

无

【相关主题】

无

## HI\_MPI\_ISP\_SetDcirisAttr

【描述】

设定 DC-Iris AI 算法的控制属性，该函数可实现 DC-Iris 自动光圈的参数设置。

【语法】

```
HI_S32 HI_MPI_ISP_SetDcirisAttr(ISP_DEV IspDev, const ISP\_DCIRIS\_ATTR\_S
*pstDcirisAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstDcirisAttr	DC-Iris 自动光圈控制属性结构体指针。	输入



#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

#### 【注意】

- DC-Iris 光圈控制采用 PID 算法，算法根据画面亮度，调节 PWM 占空比对光圈大小进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE 直接返回，保持曝光时间和增益不变。当画面亮度稳定且 PWM 占空比维持在打开值一段时间后，AI 算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给 AE。处于光圈控制区时，更改 AE 算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE 会即时响应，根据新设定的参数和环境亮度，AI 算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭 AI 功能，否则 AE 的调节速度会受到一点影响。针对 DC-Iris 镜头建议一直打开 AI 功能，随意开关 AI 容易导致光圈控制出现异常。针对某些长焦的 DC-Iris 镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见 [ISP\\_DCIRIS\\_ATTR\\_S](#) 部分描述。
- 关闭 AI 功能，对于 DC-Iris 镜头，光圈会打开到最大。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetDcirisAttr

#### 【描述】

获取 DC-Iris 自动光圈的属性。



#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstDcirisAttr	DC-Iris 自动光圈控制属性结构体指针。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## HI\_MPI\_ISP\_SetPirisAttr

#### 【描述】

设定 P-Iris 自动光圈的控制属性。



### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstPirisAttr	P-Iris 自动光圈控制属性结构体指针。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiae.a

### 【注意】

- P-Iris 自动光圈控制属性包含一个只写参数 bStepFNOTableChange，建议先对该结构体赋值，set 一次之后再 get，直接 get 再 set MPI 可能会报错。
- P-Iris 镜头光圈控制通过 AE 分配路线进行。P-Iris 对接重点在于正确设置镜头相关参数和合理设置 AE 分配路线，详见 [ISP\\_PIRIS\\_ATTR\\_S](#) 和 [HI\\_MPI\\_ISP\\_SetAERouteAttr](#) 部分描述，才能保证 P-Iris 正常工作。由于不同 P-Iris 的驱动方式可能会有差别，用户可以自行修改 P-Iris 驱动以适配不同镜头。
- 关闭 AI 功能，对于 P-Iris 镜头，光圈会打开到最大光圈目标值对应步进电机位置。

### 【举例】

```
ISP_DEV IspDev = 0;  
ISP_PIRIS_ATTR_S stPirisAttr, stPirisAttrDef;
```



```
HI_U16 u16TotalStepDef = 94;
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);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetPirisAttr

#### 【描述】

获取 P-Iris 自动光圈的属性。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstPirisAttr	P-Iris 自动光圈控制属性结构体指针。	输出

#### 【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a、lib\_hiaa.a

【注意】

无

【举例】

无

【相关主题】

无

## 3.5 数据类型

### 3.5.1 Register

- [AE\\_SENSOR\\_REGISTER\\_S](#)：定义 sensor 注册结构体。
- [AE\\_SENSOR\\_EXP\\_FUNC\\_S](#)：定义 sensor 回调函数结构体。
- [AE\\_SENSOR\\_DEFAULT\\_S](#)：定义 AE 算法库的初始化参数结构体。
- [AE\\_ACCURACY\\_E](#)：定义曝光时间、增益的精度类型的枚举。
- [AE\\_ACCURACY\\_S](#)：定义曝光时间、增益的精度结构体。
- [AE\\_FSWDR\\_ATTR\\_S](#)：定义 ISP FSWDR 运行模式属性结构体。

#### AE\_SENSOR\_REGISTER\_S

【说明】

定义 sensor 注册结构体。

【定义】



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

#### 【成员】

成员名称	描述
stSnsExp	Sensor 注册的回调函数结构体。

#### 【注意事项】

封装的目的是为了扩展。

#### 【相关数据类型及接口】

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

## AE\_SENSOR\_EXP\_FUNC\_S

#### 【说明】

定义 sensor 回调函数结构体。

#### 【定义】

```
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_FLOAT f32Fps, AE_SENSOR_DEFAULT_S
    *pstAeSnsDft);
    HI_VOID(*pfn_cmos_slow_framerate_set)(HI_U32 u32FullLines,
    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;
```

#### 【成员】





成员名称	描述
pfn_cmos_get_ae_default	获取 AE 算法库的初始值的回调函数指针。
pfn_cmos_fps_set	设置 sensor 的帧率。
pfn_cmos_slow_framerate_set	设置 sensor 的降帧。
pfn_cmos_inttime_update	设置 sensor 的曝光时间。
pfn_cmos_gains_update	设置 sensor 的模拟增益和数字增益。
pfn_cmos_again_calc_table	计算 TABLE 类型 sensor 模拟增益。
pfn_cmos_dgain_calc_table	计算 TABLE 类型 sensor 数字增益。
pfn_cmos_get_inttime_max	帧合成 WDR 模式下，获取长、短帧不同曝光倍数时所允许的短曝光时间最大值，与 sensor 强相关。(Hi3518EV200 不支持)。
pfn_cmos_ae_fswdr_attr_set	设置 FSWDR 运行模式。(Hi3518EV200 不支持)。

#### 【注意事项】

- 如果回调函数指针不需要赋值，需要置为 NULL。
- 在 [AE\\_SENSOR\\_DEFAULT\\_S](#) 中定义了曝光时间和增益的精度，pfn\_cmos\_inttime\_update 和 pfn\_cmos\_gains\_update 中设置的曝光时间和增益都是带精度的值，如何转换成 sensor 的配置值与 sensor 强相关，请参阅 sensor 手册。

#### 【相关数据类型及接口】

[ISP\\_SENSOR\\_REGISTER\\_S](#)

## AE\_SENSOR\_DEFAULT\_S

#### 【说明】

定义 AE 算法库的初始化参数结构体。

#### 【定义】

```
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_U8  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;
HI_U8  u8AERunInterval;
} AE_SENSOR_DEFAULT_S;

```

## 【成员】

成员名称	描述
au8HistThresh	五段直方图的分割门限值数组，取值范围为[0,255]。推荐使用默认值，线性模式为{0xd,0x28,0x60,0x80}，BUILT_IN WDR 模式为{0x20,0x40,0x60,0x80}，帧合成 WDR 模式为{0xc,0x18,0x60,0x80}。(Hi3518EV200 不支持)
u8AeCompensation	AE 亮度目标值，取值范围为[0,255]，建议用 0x38~0x40。
u32LinesPer500ms	每 500ms 的总行数。
u32FlickerFreq	抗闪频率，数值为当前电源频率的 256 倍。



成员名称	描述
f32Fps	基准帧率。
u32InitExposure	默认初始曝光量，等于曝光时间(行数)*系统增益(6bit 小数精度)。AE 算法采用该值作为初始 5 帧的曝光量，可用于运动 DV 加速启动。建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到 AE 快速收敛。线性/WDR 模式切换时，也会采用该值作为切换后第一帧的曝光量。合理修改 sensor 初始化序列的曝光时间和增益，将曝光时间(行数)*增益(倍数，6bit 小数精度)计算得到曝光量，并赋值给该变量，可使得切换更加平滑。若不设置，该值默认为 0。
u32InitAESpeed	默认初始 AE 调节速度，海思 AE 算法采用该值作为初始 100 帧的调节速度，可用于运动 DV 加速启动。建议关注快速启动的产品形态可将该值配置为 128。若不设置，该值默认为 0。AE 算法内部会将该值钳位至[64, 255]。
u32InitAETolerance	默认初始 AE 曝光容忍偏差，海思 AE 算法采用该值作为初始 100 帧的曝光容忍偏差值，可用于设置得到首次 AE 收敛稳定标志的亮度范围。AE 统计亮度第一次落在[目标亮度-u32InitAETolerance, 目标亮度+u32InitAETolerance]范围内时，得到首次 AE 收敛稳定标志。若不设置，该值默认为 0。若在 cmos.c 不对该值赋值或赋值为 0，则 AE 算法内部将其置为 u8AeCompensation/10。AE 算法内部会将该值钳位至(0, 255]。
u32FullLinesStd	基准帧率时 1 帧的有效行数。
u32FullLinesMax	sensor 降帧后 1 帧所能达到的最大行数，一般设为 sensor 所支持的最大行数。
u32FullLines	当前实际生效的 1 帧的有效行数。
u32MaxIntTime	最大曝光时间，以行为单位。
u32MinIntTime	最小曝光时间，以行为单位。
u32MaxIntTimeTarget	最大曝光时间目标值，以行为单位。
u32MinIntTimeTarget	最小曝光时间目标值，以行为单位。
stIntTimeAccu	曝光时间精度。
u32MaxAgain	最大 sensor 模拟增益，以倍为单位。
u32MinAgain	最小 sensor 模拟增益，以倍为单位。
u32MaxAgainTarget	最大 sensor 模拟增益目标值，以倍为单位。
u32MinAgainTarget	最小 sensor 模拟增益目标值，以倍为单位。
stAgainAccu	sensor 模拟增益精度。



成员名称	描述
u32MaxDgain	最大 sensor 数字增益，以倍为单位。
u32MinDgain	最小 sensor 数字增益，以倍为单位。
u32MaxDgainTarget	最大 sensor 数字增益目标值，以倍为单位。
u32MinDgainTarget	最小 sensor 数字增益目标值，以倍为单位。
stDgainAccu	sensor 数字增益精度。
u32MaxISPDgainTarget	最大 ISP 数字增益目标值，以倍为单位。
u32MinISPDgainTarget	最小 ISP 数字增益目标值，以倍为单位。
u32ISPDgainShift	ISP 数字增益精度。
stAERouteAttr	AE 默认分配路线。
bAERouteExValid	AE 扩展分配路线是否生效开关，该值为 HI_TRUE 时使用扩展分配路线，否则使用普通 AE 分配路线。
stAERouteAttrEx	AE 默认扩展分配路线，合理配置可用于优化 WDR 模式图像效果。
u16ManRatioEnable	两帧合成 WDR 手动曝光比使能，该值为 HI_TRUE 时，曝光比固定，不会根据场景自动更新。(Hi3518EV200 不支持)
u32Ratio	两帧合成 WDR 默认曝光比。当 u16ManRatioEnable 为 HI_TRUE 时，采用 u32Ratio 作为默认曝光比。6bit 小数精度。(Hi3518EV200 不支持) 取值范围：[0x40, 0xFFFF]。
enIrisType	默认镜头类型，DC-Iris 或 P-Iris。默认镜头类型与实际对接镜头类型不一致时，AI 算法无法正常工作。
stPirisAttr	P-Iris 参数，与具体镜头相关。只有默认镜头类型是 P-Iris 时，才需要用该结构体参数。
enMaxIrisFNO	P-Iris 可用的最大 F 值，与具体使用镜头相关，使用 P-Iris 时必须设置，用于限制实际生效的 enMaxIrisFNOTarget 和 enMinIrisFNOTarget，避免光圈目标值落到不合理的范围导致 AE 分配异常。 取值范围为[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。
enMinIrisFNO	P-Iris 可用的最小 F 值，与具体使用镜头相关，使用 P-Iris 时必须设置，用于限制实际生效的 enMaxIrisFNOTarget 和 enMinIrisFNOTarget，避免光圈目标值落到不合理的范围导致 AE 分配异常。 取值范围为[ISP_IRIS_F_NO_32_0, enMaxIrisFNO]。
u8AERunInterval	默认 AE 算法运行间隔，以帧为单位。若不设置，则默认



成员名称	描述
	AE 每帧执行一次。 取值范围：(0x0, 0xFF]。

#### 【注意事项】

- 线性/WDR 模式切换时，会回调 `pfn_cmos_get_ae_default` 函数更新 AE 相关默认参数。若 WDR 模式要使用 AE 扩展分配路线而线性模式不需要，建议在 `pfn_cmos_get_ae_default` 函数里面先对 AE 路线清零：`baERouteExValid = HI_FALSE`，`stAERouteAttr.u32TotalNum = 0`，`stAERouteAttrEx.u32TotalNum = 0`，然后视需要在 WDR 分支赋值。
- 海思 AE 算法采用 `u32InitExposure` 作为初始 5 帧的曝光量，可用于运动 DV 加速启动。建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到 AE 快速收敛。FSWDR 模式该值对应的是短帧曝光量，FSWDR 模式若要快速启动，最好在 `cmos.c` 设置为固定曝光比，以减少曝光比调整的时间，待 AE 稳定后再根据需要设置为自动曝光比。若在 `cmos.c` 未给 `u32InitExposure` 赋值或将 `u32InitExposure` 赋值为 0，则 AE 算法内部按起始曝光量为 1024 开始计算。
- 在切换帧率时，无论帧率是否发生变化，注意返回实际生效的帧率 `f32Fps` 及有效行数 `u32FullLines`。

#### 【相关数据类型及接口】

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

## AE\_ACCURACY\_E

#### 【说明】

定义曝光时间、增益的精度类型的枚举。

#### 【定义】

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

#### 【成员】

成员名称	描述
AE_ACCURACY_DB	DB 精度类型。
AE_ACCURACY_LINEAR	线性精度类型。
AE_ACCURACY_TABLE	表格类型。



#### 【注意事项】

无

#### 【相关数据类型及接口】

[AE\\_ACCURACY\\_S](#)

## AE\_ACCURACY\_S

#### 【说明】

定义曝光时间、增益的精度结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
enAccuType	精度类型，包括线性类型、DB 类型和 TABLE 类型。
f32Accuracy	精度值。
f32Offset	曝光时间的偏移量，以行为单位，该值配置与 sensor 强相关。

#### 【注意事项】

- 大部分增益的精度均可归结为线性精度、DB 精度和 TABLE 精度这几类。
- 线性精度指的是增益的倍数均匀增加。例如 sensor 能支持的 again 为 1 倍，1(1+1/16)倍，1(1+2/16)倍……这种情况下，精度类型可以设定为 AE\_ACCURACY\_LINEAR，精度值设置为 0.0625，在这种精度下，again 为 32 则表示  $32 \times 0.0625 = 2$  倍增益。
- DB 精度指的是增益的倍数以倍增的形式增加。例如 sensor 的芯片手册说明能支持的 again 为 0db，0.3db，0.6db……这种情况下，精度类型可以设定为 AE\_ACCURACY\_DB，精度值设置为 0.3，在这种精度下，again 为 80 则表示  $80 \times 0.3\text{db} = 24\text{db}$ ，24db 是 16 倍增益。再例如 sensor 能支持的 again 为 1 倍，2 倍，4 倍，8 倍……这种情况下，对应为 db 则为 0db，6db，12db，18db，所以精度类型设定为 AE\_ACCURACY\_DB，精度值设置为 6。由于 AE 算法内部实现线性转 DB 时计算精度存在误差，当 DB 精度较高(小于 1)时，建议采用 TABLE 精度来实现。



- TABLE 精度指增益的倍数是通过查表的形式获得，表格统一使用 10bit 精度，即 1024 表示 1 倍增益。当某些 sensor 的增益值的增加规律非线性或者 DB 精度较高时，可以使用 TABLE 方式，AE 算法计算出需要的模拟增益/数字增益的数值，用查询 sensor 增益表格中最接近的值作为数字增益/模拟增益的值。
- 使用 TABLE 模式时，需要相应的初始化回调结构体 AE\_SENSOR\_EXP\_FUNC\_S 中的回调函数。
- 曝光行数的精度通常都是线性的，都是以行为单位。
- 曝光时间与 sensor 增益保持线性关系是 AE 曝光量分配的前提，即认为曝光量一定的情况下，曝光时间和 sensor 增益可以相互转换而保持图像亮度基本不变。比如说曝光量为 4096，那么分配为曝光时间(2)\*增益(2048)，也可以分配为曝光时间(4)\*增益(1024)，这 2 种情况下图像亮度应该基本不变。若不满足这种线性关系，在高亮环境下，由于曝光时间很短，1 行曝光时间的变化也容易使画面发生闪烁。某些 sensor，如 Panasonic MN34220 采用 1080p@30fps 的初始化序列配置时，曝光时间必须偏移 0.8018 行之后才与 sensor 增益有线性关系，因此增加了 f32Offset 这个变量，它为 float 型变量，以行为单位，将 f32Offset 设置成 0.8018，AE 算法内部即可完成偏移处理。一般 sensor 的曝光时间和增益不存在这种偏移关系，需在 cmos.c 中将该值配为 0。
- 规定这 3 种精度类型，可以以统一的接口形式对接绝大部分 sensor。

#### 【相关数据类型及接口】

[ISP\\_SENSOR\\_REGISTER\\_S](#)

## AE\_FSWDR\_ATTR\_S

#### 【说明】

定义 ISP FSWDR 运行模式属性结构体。

#### 【定义】

```
typedef struct hiAE_FSWDR_ATTR_S
{
    ISP\_FSWDR\_MODE\_E enFSWDRMode;
} AE_FSWDR_ATTR_S;
```

#### 【成员】

成员名称	描述
enFSWDRMode	FSWDR 运行模式，目前包括正常 WDR 模式和长帧模式。

#### 【注意事项】

长帧模式只在行模式 WDR 下才有效，帧模式 WDR 把曝光比设置为 1: 1 即可达到与行 WDR 长帧模式一样的效果。

#### 【相关数据类型及接口】

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



## 3.5.2 AE

- [ISP\\_AE\\_MODE\\_E](#): 定义自动曝光的模式。
- [ISP\\_AE\\_STRATEGY\\_E](#): 定义 AE 曝光策略模式。
- [ISP\\_AE\\_DELAY\\_S](#): 定义 AE 延时属性。
- [ISP\\_AE\\_RANGE\\_S](#): 定义曝光时间或增益的最大值和最小值。
- [ISP\\_ANTIFLICKER\\_MODE\\_E](#): 定义抗闪模式。
- [ISP\\_ANTIFLICKER\\_S](#): 定义抗闪属性。
- [ISP\\_SUBFLICKER\\_S](#): 定义 ISP 图像亚抗闪属性。
- [ISP\\_FSWDR\\_MODE\\_E](#): 定义 ISP FSWDR 运行模式。
- [ISP\\_AE\\_ATTR\\_S](#): 定义自动曝光属性。
- [ISP\\_ME\\_ATTR\\_S](#): 定义手动曝光属性。
- [ISP\\_EXPOSURE\\_ATTR\\_S](#): 定义 ISP 曝光属性。
- [ISP\\_AE\\_ROUTE\\_NODE\\_S](#): 定义 AE 分配路线节点属性。
- [ISP\\_AE\\_ROUTE\\_S](#): 定义 AE 曝光分配策略属性。
- [ISP\\_AE\\_ROUTE\\_EX\\_NODE\\_S](#): 定义 AE 扩展分配路线节点属性。
- [ISP\\_AE\\_ROUTE\\_EX\\_S](#): 定义 AE 曝光分配策略扩展属性。
- [ISP\\_EXP\\_INFO\\_S](#): 定义 ISP 曝光内部状态信息。

### ISP\_AE\_MODE\_E

#### 【说明】

定义自动曝光的模式。

#### 【定义】

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

#### 【成员】

成员名称	描述
AE_MODE_SLOW_SHUTTER	自动降帧模式，即 SLOW_SHUTTER 模式。
AE_MODE_FIX_FRAME_RATE	固定帧率模式。

#### 【注意事项】





- 自动降帧模式是指自动曝光调节时会优先增大曝光时间来尽量减小增益。当 sensor 增益达到用户设置的最大值时，自动曝光调节会逐渐降帧率并延长曝光时间，持续到曝光时间等于自动曝光的最大时间为止。低照度环境下噪声较小但帧率会降低。
- 固定帧率模式是指自动曝光调节时保持帧率不变，低照度环境下噪声会较大。
- Built-In WDR 模式和多帧合成 WDR 模式不建议使用降帧模式，降帧会导致运动拖尾。

【相关数据类型及接口】

无

## ISP\_AE\_STRATEGY\_E

【说明】

定义 AE 曝光策略模式。

【定义】

```
typedef enum hiISP_AE_STRATEGY_E
{
    AE_EXP_HIGHLIGHT_PRIOR = 0,
    AE_EXP_LOWLIGHT_PRIOR = 1,
    AE_STRATEGY_MODE_BUTT
} ISP_AE_STRATEGY_E;
```

【成员】

成员名称	描述
AE_EXP_HIGHLIGHT_PRIOR	高光优先曝光模式。
AE_EXP_LOWLIGHT_PRIOR	低光优先曝光模式。

【注意事项】

默认曝光策略为高光优先，尽量避免画面过曝，但在宽动态场景时，暗处亮度较低。此时若主要关注暗处区域，可采用低光优先模式，但亮处容易过曝。

【相关数据类型及接口】

无

## ISP\_AE\_DELAY\_S

【说明】

定义 AE 延时属性。

【定义】



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

#### 【成员】

成员名称	描述
u16BlackDelayFrame	图像亮度小于目标亮度时间超过 u16BlackDelayFrame 帧时，AE 开始调节。
u16WhiteDelayFrame	图像亮度大于目标亮度时间超过 u16WhiteDelayFrame 帧时，AE 开始调节。

#### 【注意事项】

- u16BlackDelayFrame/u16WhiteDelayFrame 设置越大，在 AE 调节步长相同的情况下，调节至目标亮度需要越长时间。
- 人眼对于过曝比较敏感，建议 u16WhiteDelayFrame 要设置得比 u16BlackDelayFrame 小一些。
- 固定降帧时，若帧率较低，u16BlackDelayFrame/u16WhiteDelayFrame 要做出相应调整，建议不要设置过大，否则 AE 收敛时间会较长。

#### 【相关数据类型及接口】

无

## ISP\_AE\_RANGE\_S

#### 【说明】

定义曝光时间或增益的最大值和最小值。

#### 【定义】

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

#### 【成员】

成员名称	描述
u32Max	最大值。
u32Min	最小值。



#### 【注意事项】

最小值不能大于最大值。

#### 【相关数据类型及接口】

无

## ISP\_ANTIFLICKER\_MODE\_E

#### 【说明】

定义抗闪模式。

#### 【定义】

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

#### 【成员】

成员名称	描述
ISP_ANTIFLICKER_NORMAL_MODE	普通抗闪模式。
ISP_ANTIFLICKER_AUTO_MODE	自动抗闪模式。

#### 【注意事项】

- ISP\_ANTIFLICKER\_NORMAL\_MODE 为普通抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间固定为 1/120 sec（60Hz）或 1/100 sec(50Hz)，不受曝光时间最小值的限制。
  - 有灯光的环境：曝光时间可与光源频率相匹配，能够防止图像闪烁。
  - 高亮度的环境：亮度越高，要求曝光时间就最短。而普通抗闪模式的最小曝光时间不能匹配光源频率，产生过曝。
- ISP\_ANTIFLICKER\_AUTO\_MODE 为自动抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间可达到 sensor 的最小曝光时间。与普通抗闪模式的差别主要在高亮度的环境体现。
  - 高亮度的环境：最小曝光时间可以达到 sensor 的最小曝光时间，能够有效抑制过曝，但此时抗闪失效。

#### 【相关数据类型及接口】

无



## ISP\_ANTIFLICKER\_S

### 【说明】

定义抗闪属性。

### 【定义】

```
typedef struct hiISP_ANTIFLICKER_S
{
    HI_BOOL bEnable;
    HI_U8   u8Frequency;
    ISP\_ANTIFLICKER\_MODE\_E enMode;
} ISP_ANTIFLICKER_S;
```

### 【成员】

成员名称	描述
bEnable	bEnable 为 HI_TRUE 时使能图像抗闪，为 HI_FALSE 时不使能图像抗闪。
u8Frequency	抗闪频率值。 取值范围：[0x0, 0xFF]，默认值为 0x0。
enMode	抗闪模式，普通抗闪或自动抗闪。

### 【注意事项】

无

### 【相关数据类型及接口】

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

## ISP\_SUBFLICKER\_S

### 【说明】

定义 ISP 图像亚抗闪属性。

### 【定义】

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

### 【成员】



成员名称	描述
bEnable	bEnable 为 HI_TRUE 时使能图像亚抗闪功能，为 HI_FALSE 时不使能图像亚抗闪。
u8LumaDiff	抗闪程度设置，范围 [0x0, 0x64]。亚抗闪功能生效时，该值越大越接近于抗闪。

#### 【注意事项】

- 强制抗闪模式时，最小曝光时间固定为 1/120 sec（60Hz）或 1/100 sec(50Hz)，在某些场景（如室内对准室外窗户的背光场景）画面可能会过曝严重，但不抗闪画面工频闪又比较厉害。在这种情况下引入亚抗闪模式是为了取得过曝与闪烁之间的平衡。在强制抗闪模式下，当亚抗闪功能生效时，若画面亮度小于 (AeCompensation + u8LumaDiff)，那么曝光时间仍会固定为最小抗闪时间 1/120 sec（60Hz）或 1/100 sec(50Hz)以防止图像闪烁。若画面亮度大于(AeCompensation + u8LumaDiff)，则曝光时间会减小，使得画面最终亮度接近(AeCompensation + u8LumaDiff)，通过引入一定程度的画面闪烁来避免画面过曝严重。
- 只有在满足打开抗闪、强制抗闪模式且抗闪频率值不等于 0 的前提下，亚抗闪功能才能生效。

#### 【相关数据类型及接口】

无

## ISP\_FSWDR\_MODE\_E

#### 【说明】

定义 ISP FSWDR 运行模式。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
ISP_FSWDR_NORMAL_MODE	正常 FSWDR 模式，此时 AE 和合成模块按照自动/手动曝光比进行工作。
ISP_FSWDR_LONG_FRAME_MODE	长帧模式，此时 AE 将短帧曝光时间设置为最小值，长帧曝光时间接近 1 帧所允许的最大值，合成模块只输出长帧数据，可用于优



	化弱宽动态场景或低照度时的图像质量。
--	--------------------

#### 【注意事项】

长帧模式只在行模式 WDR 下才有效，帧模式 WDR 把曝光比设置为 1: 1 即可达到与行 WDR 长帧模式一样的效果。

#### 【相关数据类型及接口】

无

## ISP\_AE\_ATTR\_S

#### 【说明】

定义自动曝光属性。

#### 【定义】

```
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;
    HI_U8 au8Weight[AE_ZONE_ROW][AE_ZONE_COLUMN];
} ISP_AE_ATTR_S;
```

#### 【成员】



成员名称	描述
stExpTimeRange	曝光时间范围，设置最大值和最小值，以微秒(us)为单位。 取值范围：[0x0, 0xFFFFFFFF]，具体范围与 sensor 相关。
stAGainRange	Sensor 模拟增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
stDGainRange	Sensor 数字增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
stISPDGainRange	ISP 数字增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0x40000]。
stSysGainRange	系统增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32GainThreshold	自动降帧时的系统增益门限值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]。
u8Speed	自动曝光调整时的速度。 取值范围：[0x0, 0xFF]，默认值为 0x40。
u16BlackSpeedBias	从较亮场景到较暗场景自动曝光调整时的速度系数。 取值范围：[0x0, 0xFFFF]，默认值为 0x90。
u8Tolerance	自动曝光调整时对画面亮度的容忍偏差。 取值范围：[0x0, 0xFF]，默认值为 0x2。
u8Compensation	自动曝光调整时的目标亮度。 取值范围：[0x0, 0xFF]，默认值为 0x38。
u16EVBias	自动曝光调整时的曝光量偏差值，10bit 小数精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x400。
enAESTrategyMode	自动曝光策略，高光优先或低光优先。
u16HistRatioSlope	感兴趣区域的权重。 取值范围：[0x0, 0xFFFF]，默认值为 0x80。
u8MaxHistOffset	感兴趣区域对统计平均值影响的最大程度。 取值范围：[0x0, 0xFF]，默认值为 0x10。
enAEMode	自动曝光模式，自动降帧模式或固定帧率模式。
stAntiflicker	抗闪属性设置。默认抗闪不使能。
stSubflicker	亚抗闪属性设置。默认亚抗闪不使能。
stAEDelayAttr	延时属性设置。默认 u16BlackDelayFrame=8， u16WhiteDelayFrame=0。



成员名称	描述
bManualExpValue	手动曝光量使能，该值为 HI_TRUE 时，AE 算法采用 u32ExpValue 作为当前曝光量进行曝光时间和增益等的分配，为 HI_FALSE 时采用自动计算的曝光量进行分配。默认为 HI_FALSE。
u32ExpValue	手动曝光量值，等于曝光时间(行数)*系统增益(6bit 小数精度)。 取值范围：(0x0, 0xFFFFFFFF]。
au8Weight[AE_ZONE_ROW][AE_ZONE_COLUMN]	自动曝光 15*17 的区域权重表，只影响全局五段直方图和 256 段直方图统计信息。

#### 【注意事项】

- 自动曝光的最大最小时间及增益  
可根据不同的场景对曝光时间及增益进行限定，如有高速运动物体场景可限定最大曝光时间值为较小值，这样可减轻运动物体拖影现象。ISP 数字增益会引入噪声，为了控制噪声驱动文件会对 ISP 数字增益最大值进行限制，不会达到理论最大值（256 倍）。
- 自动曝光的系统增益  
若(sensor 模拟增益最小值\*sensor 数字增益最小值\*ISP 数字增益最小值)小于系统增益最小值，则 AE 算法内部计算时最小增益会被限制到系统增益的最小值。若 (sensor 模拟增益最大值\*sensor 数字增益最大值\*ISP 数字增益最大值)大于系统增益最大值，则 AE 算法内部计算时最大增益会被限制到系统增益的最大值。推荐通过设置系统增益的最大、最小值进行增益限制，分别限制 sensor 模拟增益、sensor 数字增益和 ISP 数字增益时，若把较高精度的 ISP 数字增益限制到 1 倍，容易导致闪烁。实际上，AE 算法内部利用系统增益来计算最大/最小曝光量，而并不是直接限制某项增益的值。比如说系统增益最大值限制为 1 倍，但 sensor 模拟增益的最小值限制为 2 倍，则实际生效的结果以 sensor 模拟增益的限制为准，sensor 数字增益及 ISP 数字增益的限制也有同理。
- 自动降帧时的系统增益门限值  
在 SLOW\_SHUTTER 模式下，当系统增益达到所设置的门限值时，系统将自动进入 SLOW\_SHUTTER 模式。
- u8Speed 用于设定自动曝光时的收敛速度，该值越大曝光的收敛速度越快，u16BlackSpeedBias 用于设定从较亮场景到较暗场景自动曝光时的收敛速度系数，该值越大从较亮场景到较暗场景曝光的收敛速度越快，收敛速度过快会导致收敛过程中出现反复震荡。对于 Panasonic mn34220 这款 sensor，受 sensor 性能限制，在 2 帧合成 WDR 模式下，建议 u8Speed 设置不能超过 64，否则在光线剧烈变化场景时可能出现闪烁。
- 亮度补偿属性 u8Compensation 用于调节曝光的目标亮度。曝光亮度补偿值越大则图像亮度越高。
- 曝光偏差属性 u16EVBias 用于在特殊场景下对画面的目标亮度进行微调，也可认为是更高精度的亮度补偿值，通过调节该值来改变画面目标亮度，真实生效的 AE



目标亮度为  $u8Compensation * u16EVBias / 1024$ 。u8Compensation 不变时，该值越大则图像亮度越高。

- 曝光容忍偏差属性 u8Tolerance 用于调节曝光对环境的灵敏度，曝光容忍偏差值越大则曝光越不敏感，且可能导致同一目标亮度值多次调节得到的亮度差异越大，所以该属性推荐不能设定过大。尤其当目标亮度较低时，Tolerance 较大，最终画面收敛亮度可能会有明显差异。通过改变 EVBias 对画面目标亮度进行微调时，若 Tolerance 较大，过小的 EVBias 变化量可能会看不出调整效果。
- 曝光策略属性 enAESTrategyMode 用于选择对高光优先或低光优先的曝光策略。高光优先意味着对高光敏感，尽量避免画面过曝。低光优先意味着对低光敏感，尽量看清楚暗处区域，不管画面是否过曝。默认的曝光策略是高光优先，用户可根据场景需要调整。
- u16HistRatioSlope 用于设定感兴趣区域的权重。若是高光优先曝光模式，则该值设置的是高光区域的权重，该值越大，则意味着对高光区域越敏感。反之，若是低光优先模式，则该值设置的是低光区域的权重，该值越大，则意味着对暗处区域越敏感。建议 u16HistRatioSlope 的值设置不要超过 0x100。高光优先曝光模式下，如果该值过大，只要画面中有一小部分过曝区域，AE 算法就会认为画面亮度远大于目标亮度，这样曝光量就会不断往下调，导致画面变得很黑，同时画面中的过曝区域也在快速变小，达到某个程度的话 AE 算法又会认为画面亮度比目标亮度小了，曝光量又会往上调，如此反复出现来回调的现象，看起来像是画面闪烁。类似的，在低光优先模式下，如果 AE 目标亮度设置得比较低，若画面中有黑色物体来回移动，对画面亮度也会有较大影响，此时调低曝光调整步长 u8ExpStep 和提高容忍偏差 s16ExpTolerance 能够缓解该问题。
- 自动曝光时，可设定感兴趣区域对统计平均值影响的最大程度 u8MaxHistOffset。该值相当于对提高 u16HistRatioSlope 时增加的权重做限制，若该值为 0，无论 u16HistRatioSlope 多大，也不会对高光或低光区域做特殊处理，此时的统计平均值就是原始值。通过合理设置该值，可以保证任何场景 AE 稳定后画面平均亮度都在一定范围内。在高光优先曝光模式下，如果该值设置较大，在对比度稍高一些的场景，如晴天室外场景，有天空有树木，则可能导致整体画面亮度偏低，因为此时优先保证了亮区天空的效果，通过限制该值从而限制对亮区的倾斜程度可以解决该问题。
- 曝光策略切换时，最好同时更新 u16HistRatioSlope 和 u8MaxHistOffset 的值，否则这 2 个值会采用上一种策略下的配置，效果可能与预期不符。
- 在做强光抑制方案时，建议在高光优先曝光模式下，通过降低 AE 目标亮度，同时合理设置 u16HistRatioSlope 和 u8MaxHistOffset 来抑制强光，暗区则可以通过使能 DRC 来看清楚。低光优先则可以用于实现非指定区域的背光补偿。
- AE 曝光控制类型为自动时，可设定曝光模式 enAEMode。该值可设定为慢快门模式(SLOW\_SHUTTER)或固定帧率模式。慢快门模式通常用于低照度场景下进行自动降帧，以减少画面噪声。Built-In WDR 模式和多帧合成 WDR 模式不建议使用降帧模式。
- 抗闪属性结构体 stAntiflicker 可用于设定抗闪使能，抗闪频率和抗闪模式等属性。
- 亚抗闪属性结构体 stSubflicker 可用于设定亚抗闪使能及抗闪程度属性。若有自动光圈，建议关闭亚抗闪功能。
- AE 曝光控制类型为自动时，可设定 AE 延时生效属性结构体 stAEDelayAttr。该值的合理设置可提高画面亮度的稳定性，防止快速运动物体经过导致画面亮度发生变化。低码率设置时可适当增加该值，以避免 AE 调节时出现块效应。



- 用户可以通过将 bManualExpValue 置为 HI\_TRUE，手动设置曝光量 u32ExpValue 来屏蔽海思 AE 算法的曝光量调整部分，只用到曝光量分配部分。u32ExpValue 的值会受到最大、最小曝光量的限制。最大、最小曝光量分别对应最大、最小曝光时间与增益的乘积。
- 自动曝光的权重表  
自动曝光的静态统计信息分为 15\*17 个区域，可通过设定权重表改变每个区域的权重。如可使中心区域的权重加大，则中心区域的亮度变化会使图像的全局直方图统计信息产生更多的变化。该属性可调节感兴趣区域的曝光权重，可用于实现指定区域的背光补偿。

#### 【相关数据类型及接口】

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

## ISP\_ME\_ATTR\_S

#### 【说明】

定义手动曝光属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
enExpTimeOpType	手动曝光时间使能，默认值为 OP_TYPE_AUTO
enAGainOpType	手动 sensor 模拟增益使能，默认值为 OP_TYPE_AUTO
enDGainOpType	手动 sensor 数字增益使能，默认值为 OP_TYPE_AUTO
enISPDGainOpType	手动 ISP 数字增益使能，默认值为 OP_TYPE_AUTO
u32ExpTime	手动曝光时间，以微秒(us)为单位，默认值为 0x4000。 取值范围：[0x0, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32AGain	手动 sensor 模拟增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。



成员名称	描述
u32DGain	手动 sensor 数字增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32ISPDGain	手动 ISP 数字增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0x40000]，具体范围与 sensor 相关。

#### 【注意事项】

- 手动曝光使能参数有效时，必须设置相应的手动曝光参数，若不设置，则采用系统默认值。
- 自动模式下，更改手动曝光属性的值不会生效，手动模式下，手动曝光时间和增益的值会受到自动模式下最大/最小曝光时间和增益的限制。
- 增益单位为 10bit 小数精度的倍数，即 1024 代表 1 倍。
- 若曝光参数设置超出最大（小）值，将使用 sensor 支持的最大（小）值代替。
- 手动曝光模式下，P-Iris 功能不生效。若要固定曝光时间或者增益来调节 P-Iris，可在自动模式下，将曝光时间或增益的最大/最小值设置成一样来实现。

#### 【相关数据类型及接口】

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

## ISP\_EXPOSURE\_ATTR\_S

#### 【说明】

定义 ISP 曝光属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
bByPass	AE 模块 bypass 功能使能，默认为 HI_FALSE。



成员名称	描述
enOpType	自动曝光或手动曝光开关，默认为 OP_TYPE_AUTO。
u8AERunInterval	AE 算法运行的间隔，取值范围为[1,255]，取值为 1 时表示每帧都运行 AE 算法，取值为 2 时表示每 2 帧运行 1 次 AE 算法，依此类推。建议该值设置不要大于 2，否则 AE 调节速度会受到影响。WDR 模式时，该值建议设置为 1，这样 AE 收敛会更加平滑。
bHistStatAdjust	256 段直方图统计方式是否根据场景进行调整开关，默认为 HI_TRUE。
bAERouteExValid	AE 扩展分配路线是否生效开关，HI_TRUE 时使用 AE 扩展分配路线，否则使用普通 AE 分配路线。默认为 HI_FALSE。
stManual	手动曝光属性结构体
stAuto	自动曝光属性结构体

#### 【注意事项】

- AE ByPass 为 HI\_TRUE 时，AE 模块被 bypass，任何 AE 配置都不会对图像亮度产生影响。ISP\_AE\_RESULT\_S 保持为 AE bypass 前一帧的值。
- 海思 AE 主要基于全局 256 段直方图信息进行亮度调整，正常场景 256 段直方图只统计 Gb 或 Gr 分量。bHistStatAdjust 为 HI\_TRUE 时，若进入了大面积单色场景（如大面积红色或蓝色），AE 算法会根据全局平均值信息对 256 段直方图统计方式进行调整，把 R 分量或 B 分量也考虑进来，如此可以防止大面积红色或蓝色场景画面亮度偏高。bHistStatAdjust 为 HI\_FALSE 时，AE 算法不会调整 256 段直方图统计方式，红外场景建议将 bHistStatAdjust 设置为 HI\_FALSE。
- WDR/线性模式切换时，u8AERunInterval 会重置为 1，用户可以在切换完成后，根据需要修改该值。

#### 【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetExposureAttr](#)
- [HI\\_MPI\\_ISP\\_GetExposureAttr](#)

## ISP\_AE\_ROUTE\_NODE\_S

#### 【说明】

定义 AE 分配路线节点属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
u32IntTime	节点曝光时间，单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]。
u32SysGain	节点增益，包括 Sensor 模拟增益，Sensor 数字增益和 ISP 数字增益，10bit 精度。 取值范围：[0x400, 0xFFFFFFFF]。
enIrisFNO	节点光圈 F 值大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。
u32IrisFNOLin	节点光圈 F 值等效增益大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[1, 1024]。

#### 【注意事项】

- 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。在计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益 FNO =  $1 \ll \text{ISP\_IRIS\_F\_NO\_XX\_XX}$ 。由此可知 F32.0 对应增益 1，F22.0 对应增益 2，F16.0 对应增益 4，以此类推，F1.0 对应增益 1024。
- 可以通过设置结构体 [ISP\\_PIRIS\\_ATTR\\_S](#) 中的 bFNOExValid 来决定实际生效的 AE route 节点光圈值采用 enIrisFNO 或 u32IrisFNOLin。bFNOExValid 为 HI\_TRUE 时采用高精度的 u32IrisFNOLin，默认采用 enIrisFNO。enIrisFNO 会受到 [ISP\\_PIRIS\\_ATTR\\_S](#) 中 enMaxIrisFNOTarget/ enMinIrisFNOTarget 的限制，u32IrisFNOLin 会受到 [ISP\\_PIRIS\\_ATTR\\_S](#) 中 u32MaxIrisFNOTarget/ u32MinIrisFNOTarget 的限制。另外，enIrisFNO 和 u32IrisFNOLin 还会受到 [AE\\_SENSOR\\_DEFAULT\\_S](#) 中 enMaxIrisFNO/ enMinIrisFNO 的限制，所以对接 P-Iris 时要在 cms.c 中给 enMaxIrisFNO/ enMinIrisFNO 赋合适的值。
- 不支持设置等曝光量节点。
- 为了保证曝光节点的曝光量，曝光节点的一个分量发生限制时，会对其他未达到最大值的分量进行调整，实际生效路径可能与设定路径不一致。为了防止曝光量溢出，如果光圈分量使能，则同一节点的曝光时间与增益的乘积最大值不能超过 0x1FFFFFFFFFFFFFFF，如果光圈分量不使能，则同一节点的曝光时间与增益的乘积最大值不能超过 0x7FFFFFFFFFFFFFFF。
- 如果相邻节点的曝光量增加，则应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
- 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制。



- 针对 P-Iris，建议将第一个节点的曝光时间、增益和光圈 F 值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈 F 值都设置为相应的最大目标值。

#### 【相关数据类型及接口】

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

## ISP\_AE\_ROUTE\_S

#### 【说明】

定义 AE 曝光分配策略属性。

#### 【定义】

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

#### 【成员】

成员名称	描述
u32TotalNum	曝光分配路线节点数目，目前最大为 16。
astRouteNode [ISP_AE_ROUTE_MAX_NODES]	曝光分配路线节点属性。

#### 【注意事项】

- 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含 Sensor 模拟增益、Sensor 数字增益、ISP 数字增益。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 分配策略是曝光时间优先，其次分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 针对 P-Iris，默认 AE 分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置 AE route。

#### 【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetExposureAttr](#)
- [HI\\_MPI\\_ISP\\_GetExposureAttr](#)



## ISP\_AE\_ROUTE\_EX\_NODE\_S

### 【说明】

定义 AE 扩展分配路线节点属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
u32IntTime	节点曝光时间，单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]。
u32Again	sensor 模拟增益，10bit 精度。 取值范围：[0x400, 0x3FFFFFF]。
u32Dgain	sensor 数字增益，10bit 精度。 取值范围：[0x400, 0x3FFFFFF]。
u32IspDgain	ISP 数字增益，10bit 精度。 取值范围：[0x400, 0x40000]。
enIrisFNO	节点光圈 F 值大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。
u32IrisFNOLin	节点光圈 F 值等效增益大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[1, 1024]。

### 【注意事项】

- 节点的曝光量是曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈的乘积，节点曝光量必须单调递增，即后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。在计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益 FNO =  $1 \ll ISP\_IRIS\_F\_NO\_XX\_XX$ 。由此可知 F32.0 对应增益 1，F22.0 对应增益 2，F16.0 对应增益 4，以此类推，F1.0 对应增益 1024。





- 可以通过设置结构体 [ISP\\_PIRIS\\_ATTR\\_S](#) 中的 `bFNOExValid` 来决定实际生效的扩展 AE route 节点光圈值采用 `enIrisFNO` 或 `u32IrisFNOLin`。`bFNOExValid` 为 `HI_TRUE` 时采用高精度的 `u32IrisFNOLin`，默认采用 `enIrisFNO`。`enIrisFNO` 会受到 [ISP\\_PIRIS\\_ATTR\\_S](#) 中 `enMaxIrisFNOTarget/ enMinIrisFNOTarget` 的限制，`u32IrisFNOLin` 会受到 [ISP\\_PIRIS\\_ATTR\\_S](#) 中 `u32MaxIrisFNOTarget/ u32MinIrisFNOTarget` 的限制。另外，`enIrisFNO` 和 `u32IrisFNOLin` 还会受到 [AE\\_SENSOR\\_DEFAULT\\_S](#) 中 `enMaxIrisFNO/ enMinIrisFNO` 的限制，所以对接 P-Iris 时要在 `cmos.c` 中给 `enMaxIrisFNO/ enMinIrisFNO` 赋合适的值。
- 不支持设置等曝光量节点。
- 为了保证曝光节点的曝光量，曝光节点的一个分量发生限制时，会对其他未达到最大值的分量进行调整，实际生效路径可能与设定路径不一致。
- 如果相邻节点的曝光量增加，则应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如 `sensor` 模拟增益分量增加，那么该段路线的分配策略是 `sensor` 模拟增益优先。
- 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制。
- 针对 P-Iris，建议将第一个节点的曝光时间、增益和光圈 F 值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈 F 值都设置为相应的最大目标值。
- 节点的曝光时间不能设置为 0，也不能设置过小，导致以 `us` 为单位的曝光时间对应的曝光行数为 0，否则可能产生异常。
- 节点增益的范围相比非扩展 AE route 有所缩小。此外，为防止曝光量溢出，`sensor` 模拟增益、`sensor` 数字增益和 ISP 数字增益的乘积等效为的 10bit 精度系统增益不能大于 `0xFFFFFFFF`。如果光圈分量使能，则同一节点的曝光时间与 10bit 精度系统增益的乘积最大值不能超过 `0x1FFFFFFFFFFFFFFF`，如果光圈分量不使能，则同一节点的曝光时间与 10bit 精度系统增益的乘积最大值不能超过 `0x7FFFFFFFFFFFFFFF`。
- 若某段路线需要设置 ISP 数字增益优先，请注意不要把 ISP 数字增益用满，因为 ISP 数字增益还有弥补分配精度的作用，建议至少留下 2 倍 ISP 数字增益的余量，供弥补精度使用，即分配路线中节点的 `ISPDgain` 不要大于 `MaxISPDgain/2`。

#### 【相关数据类型及接口】

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

## ISP\_AE\_ROUTE\_EX\_S

#### 【说明】

定义 AE 曝光分配策略扩展属性。

#### 【定义】

```
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;
```

#### 【成员】





成员名称	描述
u32TotalNum	曝光扩展分配路线节点数目，目前最大为16。
astRouteExNode [ISP_AE_ROUTE_EX_MAX_NODES]	曝光扩展分配路线节点属性。

#### 【注意事项】

- 最大支持 16 个节点，每个节点有曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈五个分量。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 扩展分配策略是曝光时间优先，再分配 sensor 模拟增益、sensor 数字增益、最后分配 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 针对 P-Iris，默认 AE 扩展分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配 sensor 模拟增益、sensor 数字增益和 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，扩展 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展 AE route。

#### 【相关数据类型及接口】

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

## ISP\_EXP\_INFO\_S

#### 【说明】

定义 ISP 曝光内部状态信息。

#### 【定义】

```
typedef struct hiISP_EXP_INFO_S
{
    HI_U32  u32ExpTime;
    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 u32FirstStableTime;
ISP_AE_ROUTE_S stAERoute;
ISP_AE_ROUTE_EX_S stAERouteEx;
}ISP_EXP_INFO_S;

```

**【成员】**

成员名称	描述
u32ExpTime	当前曝光时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]。
u32AGain	当前 sensor 模拟增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]。
u32DGain	当前 sensor 数字增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]。
u32ISPDGain	当前 ISP 数字增益，10bit 小数精度。 取值范围：[0x400, 0x40000]。
u32Exposure	当前曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为毫秒(ms)，曝光增益为 1bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]。
bExposureIsMAX	0：ISP 未达到最大曝光水平； 1：ISP 达到最大曝光水平。
s16HistError	统计信息，AE 的目标亮度值与实际值的差，该值为正表示当前期望的亮度信息大于实际的亮度信息，该值为负表示期望的亮度信息小于实际的亮度信息。 取值范围：[-0x8000, 0x7FFF]，该值为只读。
u32AE_Hist256Value[256]	全局 256 段直方图统计信息 取值范围：[0x0, 0xFFFFFFFF]。
u16AE_Hist5Value[5]	全局五段直方图统计信息 取值范围：[0x0, 0xFFFF]。 Hi3518EV200 不支持。
u8AveLum	平均亮度信息 取值范围：[0x0, 0xFF]。
u32LinesPer500ms	当前每 500ms 对应的曝光行数，可用于将曝光时间的单位由 us 转换成行数。 取值范围：[0x0, 0xFFFFFFFF]。



成员名称	描述
u32PirisFNO	当前 P-Iris 光圈 F 值对应的等效增益。 取值范围：[0x0, 0x400]。
u32Fps	实际图像帧率 * 100。
u32ISO	AE 计算得出的总增益值，ISO 表示系统增益，以常数 100 乘以倍数为单位,例如系统中 sensor 的增益为 2 倍，ISP 的增益为 1 倍，那么整个系统的 ISO 值计算方式为：2*1*100=200，即系统 ISO 为 200，本文档中涉及到的 ISO 都是采用这种计算方法。
u32FirstStableTime	ISP 启动 AE 第一次稳定的时间，单位为 us。
stAERoute	实际生效的 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。
stAERouteEx	实际生效的扩展 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。

#### 【注意事项】

- 曝光量的计算未考虑光圈状态，为曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益 4 者乘积，其中曝光时间的单位为曝光行数，曝光增益为 6bit 小数精度。若该值的精度不足以满足需求，可以根据高精度的曝光时间(us)和增益(10bit 小数精度)重新计算一个曝光量。
- ISP 是否达到最大曝光水平的计算未考虑光圈状态。如果当前曝光时间大于等于最大目标曝光时间，当前增益大于等于最大目标增益，那么就认为 ISP 达到最大曝光水平，否则就未达到最大曝光水平。
- 图像平均亮度经过归一化处理，取值范围为 0~255。
- u32Fps 可用于查询自动降帧状态下的实际帧率。

#### 【相关数据类型及接口】

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

### 3.5.3 AI

- [ISP\\_IRIS\\_STATUS\\_E](#)：定义 ISP 光圈状态。
- [ISP\\_IRIS\\_TYPE\\_E](#)：定义 ISP 光圈类型。
- [ISP\\_IRIS\\_F\\_NO\\_E](#)：定义 ISP 光圈 F 值。
- [ISP\\_MI\\_ATTR\\_S](#)：定义手动光圈属性。
- [ISP\\_DCIRIS\\_ATTR\\_S](#)：定义 DC-Iris AI 算法属性。



- [ISP\\_PIRIS\\_ATTR\\_S](#): 定义 P-Iris 属性。
- [ISP\\_IRIS\\_ATTR\\_S](#): 定义 ISP 光圈属性。

## ISP\_IRIS\_STATUS\_E

### 【说明】

定义 ISP 光圈状态。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
ISP_IRIS_KEEP	光圈保持当前状态。
ISP_IRIS_OPEN	光圈全开。
ISP_IRIS_CLOSE	光圈全关。

### 【注意事项】

该值设置为 ISP\_IRIS\_OPEN 或 ISP\_IRIS\_CLOSE 时，光圈处于全开或全关的状态，可用于测试 AI 电路和驱动是否正确。OPEN 和 CLOSE 的优先级高于 AI 使能和手动/自动模式，AI 算法运行时，为保证其能够正常工作，需将该值设置为 ISP\_IRIS\_KEEP。

### 【相关数据类型及接口】

无

## ISP\_IRIS\_TYPE\_E

### 【说明】

定义 ISP 光圈类型。

### 【定义】

```
typedef enum hiISP_IRIS_TYPE_E
{
    ISP_IRIS_DC_TYPE = 0,
    ISP_IRIS_P_TYPE,
    ISP_IRIS_TYPE_BUTT,
```



```
} ISP_IRIS_TYPE_E;
```

#### 【成员】

成员名称	描述
ISP_IRIS_DC_TYPE	DC-Iris 光圈。
ISP_IRIS_P_TYPE	P-Iris 光圈。

#### 【注意事项】

- 必须设置正确的光圈类型，AI 算法才能正常工作。
- 若对接的是手动光圈镜头，可将该值设置为 ISP\_IRIS\_DC\_TYPE，建议此时关闭 AI 使能。
- 由 DC-Iris 切换至 P-Iris，需提前设置好 P-Iris 镜头相关参数，并且将光圈状态设置为 ISP\_IRIS\_KEEP，再进行切换。

#### 【相关数据类型及接口】

无

## ISP\_IRIS\_F\_NO\_E

#### 【说明】

定义 ISP 光圈 F 值。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
ISP_IRIS_F_NO_32_0	光圈 F32.0。
ISP_IRIS_F_NO_22_0	光圈 F22.0。
ISP_IRIS_F_NO_16_0	光圈 F16.0。
ISP_IRIS_F_NO_11_0	光圈 F11.0。
ISP_IRIS_F_NO_8_0	光圈 F8.0。
ISP_IRIS_F_NO_5_6	光圈 F5.6。
ISP_IRIS_F_NO_4_0	光圈 F4.0。
ISP_IRIS_F_NO_2_8	光圈 F2.8。
ISP_IRIS_F_NO_2_0	光圈 F2.0。
ISP_IRIS_F_NO_1_4	光圈 F1.4。
ISP_IRIS_F_NO_1_0	光圈 F1.0。

#### 【注意事项】

针对 P-Iris，AE 算法根据分配路线计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益  $FNO = 1 \ll ISP\_IRIS\_F\_NO\_XX\_XX$ 。由此可知 F32.0 对应增益 1，F22.0 对应增益 2，F16.0 对应增益 4，以此类推，F1.0 对应增益 1024。

#### 【相关数据类型及接口】

无

## ISP\_MI\_ATTR\_S

#### 【说明】

定义手动光圈属性。

#### 【定义】

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

#### 【成员】

成员名称	描述
u32HoldValue	AI 校正值，用于 DC-Iris 的调试。



成员名称	描述
	取值范围为[0x0, 0x3E8]。
enIrisFNO	手动光圈大小，根据光圈 F 值进行区分，仅支持 P-Iris，不支持 DC-Iris。

#### 【注意事项】

- 对接 DC-Iris 镜头时，若 [ISP\\_IRIS\\_STATUS\\_E](#) 设置为 ISP\_IRIS\_KEEP，手动光圈使能，u32HoldValue 可用于 DC-Iris 的调试，此时 PWM 的占空比即为 u32HoldValue。
- 对接 P-Iris 镜头时，若 [ISP\\_IRIS\\_STATUS\\_E](#) 设置为 ISP\_IRIS\_KEEP，手动曝光模式且手动光圈使能时，enIrisFNO 可用于 P-Iris 的调试，此时会控制 P-Iris 步进电机走到光圈 F 值与 enIrisFNO 最接近的位置。自动曝光模式下，P-Iris 手动光圈不生效，此时若要固定光圈为某个 F 值，可以将 enMaxIrisFNOTarget/enMinIrisFNOTarget 设置为相同值来实现。

#### 【相关数据类型及接口】

无

## ISP\_DCIRIS\_ATTR\_S

#### 【说明】

定义 DC-Iris AI 算法属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
s32Kp	比例增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值过小光线剧烈变化收敛过程中容易出现振荡，该值过大容易出现超调，也会导致振荡。该值的合理设置与电路特性和镜头相关。建议值为 7000。



成员名称	描述
	取值范围为[0, 100000]。
s32Ki	积分增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值较小时，碰到强光收敛后画面会稳定在一个比较低的亮度上；该值较大时可能会导致强光场景光圈无法关闭。该值的合理设置与电路特性和镜头相关。建议值为 100，该值一般不需要修改。 取值范围为[0, 1000]。
s32Kd	微分增益，用于限制光线剧烈变化时光圈的开关速度，该值越大光线剧烈变化时光圈打开和关闭的速度越慢。该值过大对于瞬间变化的亮度过于敏感，会导致场景亮度快速变化时画面出现振荡。该值的合理设置与电路特性和镜头相关。建议值为 3000。 取值范围为[0, 100000]。
u32MinPwmDuty	最小 PWM 占空比。该值越小过曝时光圈关闭速度越快，但容易导致光圈来回震荡。该值的合理设置与电路特性和镜头相关。建议值为 250。 取值范围为[0, 1000]。
u32MaxPwmDuty	最大 PWM 占空比。该值越大画面全黑时光圈打开速度越快，该值过小则可能导致退出光圈控制区域时光圈仍未达到最大，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为 950。 取值范围为[0, 1000]。
u32OpenPwmDuty	光圈打开时的 PWM 占空比。当画面亮度稳定并且 PWM 占空比大于该值一段时间后，退出光圈控制区域。所以该值不能太小，否则容易导致光圈未达到最大就退出了光圈控制区域，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为 800。 取值范围为[0, 1000]。

#### 【注意事项】

- 当光圈关闭出现震荡时，一般意味着光圈关闭速度太快了，可以通过适当减小 s32Kp 和增大 u32MinPwmDuty 来解决。
- u32OpenPwmDuty 的取值要求在 u32MinPwmDuty 和 u32MaxPwmDuty 之间，必须确保该值能将光圈较快的打开。

#### 【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetDcirisAttr](#)
- [HI\\_MPI\\_ISP\\_GetDcirisAttr](#)





## ISP\_PIRIS\_ATTR\_S

### 【说明】

定义 P-Iris 属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
bStepFNOTableChange	P-Iris 步进电机位置与光圈 F 值映射表是否更新标志。该值为 HI_TRUE 时会更新 P-Iris 步进电机位置与 F 值映射表，为 HI_FALSE 时不更新。
bZeroIsMax	P-Iris 步进电机 Step 0 是否对应最大光圈位置标志，取值与 P-Iris 镜头相关。该值为 HI_TRUE 时表示步进电机处于位置 0 时，光圈打开至最大，为 HI_FALSE 时表示步进电机处于位置 0 时光圈关闭。
u16TotalStep	P-Iris 步进电机的总步数，具体大小与 P-Iris 镜头相关。 取值范围为[1, 1024]。
u16StepCount	P-Iris 步进电机的可用步数，具体大小与 P-Iris 镜头相关。 取值范围为[1, 1024]。
au16StepFNOTable	P-Iris 步进电机位置与 F 值映射表，具体数据与 P-Iris 镜头相关。 取值范围为[0, 1024]。
enMaxIrisFNOTarget	最大光圈目标值，可用于控制 AE 分配路线，实际生效光圈大小与 P-Iris 镜头相关。 取值范围为[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。



成员名称	描述
enMinIrisFNOTarget	最小光圈目标值，可用于控制 AE 分配路线，实际生效光圈大小与 P-Iris 镜头相关。 取值范围为[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。
bFNOExValid	对接 P-Iris 时，AE 分配路线是否采用更高精度的光圈 F 值等效增益标志。该值为 HI_TRUE 时表示 AE 分配路线采用高精度的光圈 F 值等效增益。默认为 HI_FALSE。
u32MaxIrisFNOTarget	最大光圈 F 值等效增益目标值，可用于控制 AE 分配路线，实际生效光圈大小与 P-Iris 镜头相关。 取值范围为[1, 1024]。
u32MinIrisFNOTarget	最小光圈 F 值等效增益目标值，可用于控制 AE 分配路线，实际生效光圈大小与 P-Iris 镜头相关。 取值范围为[1, 1024]。

**【注意事项】**

- 在线更换 P-Iris 镜头时，可以通过 MPI 设置该结构体，将新镜头的参数传递给 AE 算法和 P-Iris 驱动。
- bStepFNOTableChange 为只写寄存器，通过 MPI 获取该参数的值始终为 HI\_FALSE。
- u16TotalStep 表示 P-Iris 步进电机的总步数，可用于标定光圈的起始位置。  
u16StepCount 表示 P-Iris 步进电机的可用步数，一般小于 u16TotalStep，因为靠近光圈关闭端的位置在将光圈孔径转换为 F 值时误差较大，光圈调节过程中容易出现振荡，所以通常不会利用光圈关闭端附近的那些位置。当前最多支持 1024 步的 P-Iris 镜头，若 P-Iris 镜头的精度很高，可调节步数超过 1024，建议舍去光圈关闭端附近的那些位置。
- P-Iris 步进电机位置与光圈 F 值映射表 au16StepFNOTable 一般根据镜头原厂提供的步进电机位置与光圈孔径对应关系制作。由于 P-Iris 的控制通过 AE 分配路线进行，这就要求光圈 F 值与曝光时间和增益有较好的线性关系，因此映射表需要较高的精度。这里用 1024 表示 F1.0，512 表示 F1.4，以此类推，1 表示 F32.0。制作映射表时，可以根据最大光圈孔径对应的实际 F 值来规定映射表的最大值，也可以不用管光圈达到最大时的实际 F 值是多少，统一将最大光圈孔径映射为 1024。同一款镜头处于不同焦距时，au16StepFNOTable 可能需要修改。
- 以福光 NV03105P 这款 P-Iris 镜头为例说明与镜头相关的参数该如何设置。表 3-1 中步进电机位置与孔径面积对应关系是镜头原厂提供的。该款镜头步进电机总步数为 93，福光 P-Iris 镜头掉电后，按照其手册说明只停在偶数相位掉电也会丢步，因此设定最大步数为 94，确保关闭镜头会走到马达停止位置。步进电机位置为 0 时，光圈孔径面积最大，标称最大相对孔径为 F1.4，据此规定映射表 F 值最大值为 512。规定好最大值后，就可以计算出其他孔径面积对应的 F 值，因为孔径面积与 F 值成线性关系。比如说步进电机位置 1 对应的孔径面积为 48.835，得到 F 值为  $(48.835/49.366)*512=506$ ，以此类推，可以得到所有有效步进电机位置对应的 F 值。由下表可以看到，当镜头靠近关闭端时，孔径面积很小，与最大孔径面积



相差上万倍，映射表 F 值精度已不足以表现出孔径面积的变化，得到的 F 值都为 0。实际上，哪怕提高映射表精度，由于镜头关闭端附近标称孔径面积与实际孔径面积一般相差较大，得到的 F 值也是不正确的，建议不要使用，否则在光圈调节过程中容易出现振荡。对于该款镜头，建议制作映射表时只取前 62 步，因此得到镜头相关的参数如表 3-2 所示。

- 步进电机位置与光圈 F 值映射表 au16StepFNOTable 的前 u16StepCount 个元素才是有效的，会被 AE 算法计算时用到，需要保证数组座标小于 u16StepCount 时，值单调递增，当数组座标为(u16StepCount-1)时，达到光圈最大 F 值。通过 MPI 设置 au16StepFNOTable 时，只有前 u16StepCount 个元素才会被写入寄存器保存；通过 MPI 获取 au16StepFNOTable 时，只有前 u16StepCount 个元素才会读出有效数值，其他值为 0。
- 光圈类型为 P-Iris 时，海思 AE 算法计算最大/最小曝光量会参考 enMaxIrisFNOTarget/enMinIrisFNOTarget(若 bFNOExValid 为 true，则参考 u32MaxIrisFNOTarget/u32MinIrisFNOTarget)的值，因此这几个值要与 au16StepFNOTable 的值相匹配，即最大/最小值都在 au16StepFNOTable 有效数据范围内。

表3-1 P-Iris 步进电机位置与 F 值映射表，以福光 NV03105P 为例

步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
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
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



步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
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	-	-	-

表3-2 P-Iris 镜头相关参数，以福光 NV03105P 为例

参数名	参数值	备注
bZeroIsMax	HI_TRUE	步进电机位置 0 对应光圈最大孔径，因此该值为 HI_TRUE
u16TotalStep	94	步进电机总步数为 94
u16StepCount	62	步进电机可用步数为 62
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,	根据表 3-1，取步进电机前 62 步对应的 F 值，按照从小到大的排列顺序，制作映射表。AE 算法计算时，只会用到 au16StepFNOTable 数组中的前 u16StepCount 个元素。当数组座标小于



参数名	参数值	备注
	486,493,500,506,512}	u16StepCount 时，值必须保证是单调递增的。当数组座标为(u16StepCount-1)时，达到光圈最大 F 值 512
enMaxIrisFNOTarget	ISP_IRIS_F_NO_1_4	映射表最大值 512 对应 F1.4
enMinIrisFNOTarget	ISP_IRIS_F_NO_5_6	映射表最小值 30 接近对应 F5.6
bFNOExValid	HI_FALSE	默认不采用高精度的光圈 F 值等效增益
u32MaxIrisFNOTarget	512	取 au16StepFNOTable 有效数据范围内的最大值 512
u32MinIrisFNOTarget	30	取 au16StepFNOTable 有效数据范围内的最小值 30

【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetPirisAttr](#)
- [HI\\_MPI\\_ISP\\_GetPirisAttr](#)
- [HI\\_MPI\\_ISP\\_SetIrisAttr](#)
- [HI\\_MPI\\_ISP\\_SetAERouteAttrEx](#)

## ISP\_IRIS\_ATTR\_S

【说明】

定义 ISP 光圈属性。

【定义】

```
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;
```

【成员】

成员名称	描述
bEnable	自动光圈使能。



成员名称	描述
enOpType	自动光圈或手动光圈模式选择。
enIrisType	光圈类型，DC-Iris 或 P-Iris。
enIrisStatus	光圈状态。
stMIAttr	手动光圈属性设置结构体。

#### 【注意事项】

- 进行 AI 算法测试前，建议确认 AI 电路特性是否符合海思 IPC 要求。
- 针对 DC-Iris 镜头，AI 算法会根据画面亮度，调节 PWM 占空比对光圈进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE 直接返回，保持曝光时间和增益不变。当画面亮度稳定且 PWM 占空比维持在打开值一段时间后，AI 算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给 AE。处于光圈控制区时，更改 AE 算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE 会即时响应，根据新设定的参数和环境亮度，AI 算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭 AI 功能，否则 AE 的调节速度会受到一点影响。针对 DC-Iris 镜头建议一直打开 AI 功能，随意开关 AI 容易导致光圈控制出现异常。针对某些突然切换高亮场景的应用模式，由于硬件惯性，光圈可能会反复调整导致图像亮度出现震荡，收敛速度较慢。针对某些长焦的 DC-Iris 镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见 [ISP\\_DCIRIS\\_ATTR\\_S](#) 部分描述。
- 针对 P-Iris 镜头，光圈控制通过 AE 分配路线进行。P-Iris 对接重点在于正确设置镜头相关参数和合理设置 AE 分配路线，详见 [ISP\\_PIRIS\\_ATTR\\_S](#) 和 [HI\\_MPI\\_ISP\\_SetAERouteAttr](#) 部分描述，才能保证 P-Iris 正常工作。由于不同 P-Iris 的驱动方式可能会有差别，用户可以自行修改 P-Iris 驱动以适配不同镜头。
- 关闭 AI 功能，对于 DC-Iris 镜头，光圈会打开到最大；对于 P-Iris 镜头，光圈会打开到最大光圈目标值对应步进电机位置，但此时曝光分配仍会参考 AE 路线，可能导致画面亮度异常，因此对接 P-Iris 镜头时，若不想使能 AI，为保证曝光正常，需要把光圈类型切换至 `ISP_IRIS_DC_TYPE`。
- 由 DC-Iris 切换至 P-Iris，需提前设置好 P-Iris 镜头相关参数，并且将光圈状态设置为 `ISP_IRIS_KEEP`，再进行切换。
- 利用海思 Demo 板进行 DC-Iris 测试时，load ko 时要打开以下 2 个寄存器：`himm 0x200F00EC 0x0`；`himm 0x20030038 0x6`。
- 利用海思 Demo 板进行 DC-Iris 测试时，可以在插入 ISP 的 ko 时指定所用的 PWM Number，demo 板 `pwm_num=1`。代码里面默认是 `pwm_num=1`。命令如下：`insmod hi3518e_isp.ko pwm_num=1`。
- 利用海思 Demo 板进行 P-Iris 测试时，load ko 时要设置以下 4 个管脚复用寄存器：`himm 0x200F0050 0x0`；`himm 0x200F0054 0x0`；`himm 0x200F0058 0x0`；`himm 0x200F005C 0x0`。



- Huawei LiteOS 没内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk\_init.c 中执行的相关过程。设置参数需要在 release/ko 下 sdk\_init.c 中 ISP\_init 函数中，添加 stIsp\_Param.u32PwmNum=N 传入 ISP\_ModInit 中。

【相关数据类型及接口】

- [HI\\_MPI\\_ISP\\_SetIrisAttr](#)
- [HI\\_MPI\\_ISP\\_GetIrisAttr](#)



# 4 AWB

## 4.1 概述

色温随可见光的光谱成分变化而变化，在低色温光源下，白色物体偏红，在高色温光源下，白色物体偏蓝。人眼可根据大脑的记忆判断，识别物体的真实颜色，AWB 算法的功能是降低外界光源对物体真实颜色的影响，使得我们采集的颜色信息转变为在理想日光光源下的无偏色信息。

## 4.2 重要概念

- 色温：色温是按绝对黑体来定义的，光源的辐射在可见区和绝对黑体的辐射完全相同时，此时黑体的温度就称此光源的色温。
- 白平衡：在不同色温的光源下，白色在传感器中的响应会偏蓝或偏红。白平衡算法通过调整 R、G、B 三个颜色通道的强度，使白色真实呈现。

## 4.3 功能描述

AWB 模块有硬件的 WB 信息统计模块及 AWB 策略控制算法 firmware 两部分组成。ISP 的 WB 信息统计模块判断 sensor 输出的每个像素是否满足用户设定的白点条件，计算所有满足条件的像素的 R、G、B 三个颜色通道平均值。

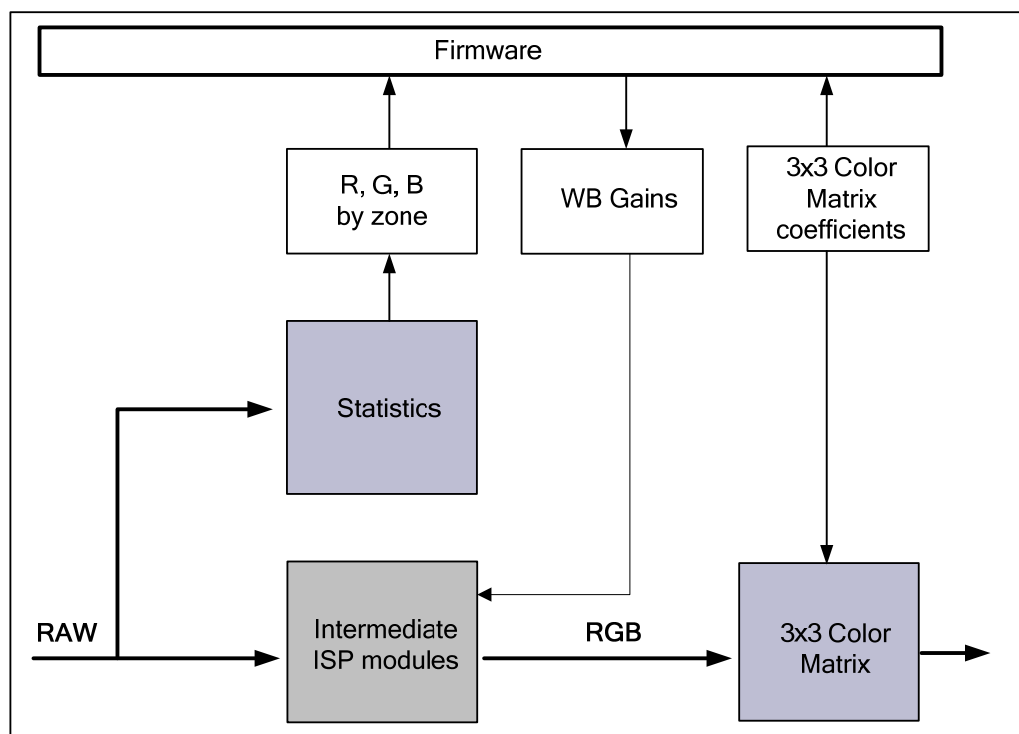
支持将图像分成 M\*N（M 行 N 列）区域，统计每个区域的 R、G、B 均值以及参与统计的白点个数。

支持输出整幅图像的 R、G、B 均值以及参与统计的白点个数。

AWB 工作原理如图 4-1 所示。



图4-1 AWB 工作原理图



## 4.4 API 参考

### 4.4.1 AWB 库接口

所有 AWB 库接口都只是针对海思 AWB 库，如果客户自己实现 AWB 库，不需要关注这些接口，且无法使用这些接口。

- [HI\\_MPI\\_AWB\\_Register](#): 向 ISP 注册 AWB 库。
- [HI\\_MPI\\_AWB\\_UnRegister](#): 向 ISP 注销 AWB 库。
- [HI\\_MPI\\_AWB\\_SensorRegCallBack](#): AWB 库提供的 sensor 注册的回调接口。
- [HI\\_MPI\\_AWB\\_SensorUnRegCallBack](#): AWB 库提供的 sensor 注销的回调接口。

#### HI\_MPI\_AWB\_Register

##### 【描述】

向 ISP 注册 AWB 库。

##### 【语法】

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

##### 【参数】



参数名称	描述	输入/输出
IspDev	设备编号	输入
pstAwbLib	AWB 算法库结构体指针	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_awb.h
- 库文件：libisp.a

#### 【注意】

- 该接口调用了 ISP 库提供的 AWB 注册回调接口 [HI\\_MPI\\_ISP\\_AWBLibRegCallBack](#)，以实现向 ISP 库注册的功能。
- 用户调用此接口完成海思 AWB 库向 ISP 库注册。
- AWB 库可以注册多个实例。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_AWB\_UnRegister

#### 【描述】

向 ISP 注销 AWB 库。

#### 【语法】

```
HI_S32 HI_MPI_AWB_UnRegister(ISP_DEV IspDev, ALG\_LIB\_S *pstAwbLib);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstAwbLib	AWB 算法库结构体指针	输入



#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_awb.h
- 库文件：libisp.a

#### 【注意】

- 该接口调用了 ISP 库提供的 AWB 反注册回调接口 [HI\\_MPI\\_ISP\\_AWBLibRegCallBack](#)，以实现 AWB 向 ISP 库反注册的功能。
- 用户调用此接口完成海思 AWB 库向 ISP 库反注册。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_AWB\_SensorRegCallBack

#### 【描述】

AWB 库提供的 sensor 注册的回调接口。

#### 【语法】

```
HI_S32 HI_MPI_AWB_SensorRegCallBack (ISP_DEV IspDev, ALG\_LIB\_S *pstAwbLib,  
SENSOR_ID SensorId, AWB\_SENSOR\_REGISTER\_S *pstRegister);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstAwbLib	AWB 算法库结构体指针	输入
SensorId	向 AWB 注册的 Sensor 的 Id。	输入
pstRegister	Sensor 注册结构体指针。	输入

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

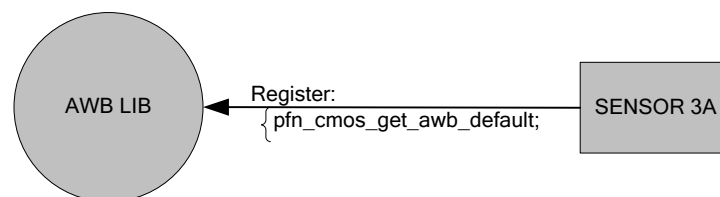
#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_awb.h
- 库文件：libisp.a

#### 【需求】

- SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 注册的 sensor 和向 3A 注册的 sensor 是否为同一个 sensor。
- AWB 通过 sensor 注册的一系列回调接口，获取差异化的初始化参数，并控制 sensor。

图4-2 AWB 库 与 sensor 库间的接口



#### 【举例】

```
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;
}
```

#### 【相关主题】



无

## HI\_MPI\_AWB\_SensorUnRegCallBack

### 【描述】

AWB 库提供的 sensor 注销的回调接口。

### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstAwbLib	AWB 算法库结构体指针	输入
SensorId	向 AWB 注册的 Sensor 的 Id。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_awb.h
- 库文件：libisp.a

### 【需求】

无

### 【相关主题】

无

## 4.4.2 AWB 控制模块

- [HI\\_MPI\\_ISP\\_SetWBAttr](#): 设置白平衡属性
- [HI\\_MPI\\_ISP\\_GetWBAttr](#): 获取白平衡属性
- [HI\\_MPI\\_ISP\\_SetAWBAttrEx](#): 设置自动白平衡扩展属性
- [HI\\_MPI\\_ISP\\_GetAWBAttrEx](#): 获取自动白平衡扩展属性



## HI\_MPI\_ISP\_SetWBAttr

### 【描述】

设置白平衡属性。

### 【语法】

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

### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstWBAttr	白平衡的参数属性	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

### 【注意】

- 白平衡控制类型为自动时，AWB 算法自动调节白平衡系数。
- 白平衡控制类型为手动时，AWB 算法失效，需自行设定 Rgain、Ggain、Bgain。

### 【举例】

无

### 【相关主题】

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



## HI\_MPI\_ISP\_GetWBAttr

### 【描述】

获取白平衡属性。

### 【语法】

```
HI_S32 HI_MPI_ISP_GetWBAttr(ISP_DEV IspDev, ISP\_WB\_ATTR\_S *pstWBAttr);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstWBAttr	白平衡的参数属性	输出

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

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



## HI\_MPI\_ISP\_SetAWBAttrEx

### 【描述】

设置自动白平衡扩展属性。

### 【语法】

```
HI_S32 HI_MPI_ISP_SetAWBAttrEx(ISP_DEV IspDev, ISP\_AWB\_ATTR\_EX\_S  
*pstAWBAttrEx);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号。	输入
pstAWBAttrEx	高级白平衡属性。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

### 【注意】

当 [HI\\_MPI\\_ISP\\_SetWBAttr](#) 接口成员 pstWBAttr->enAlgType 为 AWB\_ALG\_ADVANCE 时，此接口才有效。

### 【举例】

无

### 【相关主题】

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





## HI\_MPI\_ISP\_GetAWBAttrEx

### 【描述】

获取自动白平衡扩展属性。

### 【语法】

```
HI_S32 HI_MPI_ISP_GetAWBAttrEx(ISP_DEV IspDev, ISP\_AWB\_ATTR\_EX\_S  
*pstAWBAttrEx);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号。	输入
pstAWBAttrEx	扩展白平衡属性。	输出

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

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



### 4.4.3 WB 统计信息

- [HI\\_MPI\\_ISP\\_QueryWBInfo](#): 获取当前白平衡增益系数, 检测色温, 饱和度值, 颜色校正矩阵系数。
- [HI\\_MPI\\_ISP\\_CalGainByTemp](#): 计算特定色温下的白平衡增益系数。

#### HI\_MPI\_ISP\_QueryWBInfo

##### 【描述】

获取当前白平衡增益系数, 检测色温, 饱和度值, 颜色校正矩阵系数。

##### 【语法】

```
HI_S32 HI_MPI_ISP_QueryWBInfo(ISP_DEV IspDev, ISP\_WB\_INFO\_S *pstWBInfo);
```

##### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号。	输入
pstWBInfo	颜色相关状态参数。	输出

##### 【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 <a href="#">错误码</a> 。

##### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

##### 【需求】

- 头文件: [hi\\_awb\\_comm.h](#)、[mpi\\_awb.h](#)
- 库文件: [libisp.a](#)

##### 【注意】

无

##### 【举例】

无



#### 【相关主题】

无

## HI\_MPI\_ISP\_CalGainByTemp

#### 【描述】

计算特定色温下的白平衡增益系数。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
pstWBAttr	白平衡的参数属性。	输入
u16ColorTemp	色温值，单位为 Kelvin。 取值范围[1500, 15000]。	输入
s16Shift	白点与 Planckian 曲线的位置和距离。 取值范围[-64, 64]。	输入
pu16AWBGain	预设色温下的 R, Gr, Gb, B 四个通道增益。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	参数无效。

#### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a, lib\_hiawb.a



#### 【注意】

- 白平衡标定参数决定预设色温增益计算的准确性，因此，在调用该接口前，必须完成标定，且标定参数配置到 ISP。用户先通过 [HI\\_MPI\\_ISP\\_GetWBAttr](#) 接口获取标定结果，再调用 [HI\\_MPI\\_ISP\\_CalGainByTemp](#) 接口计算预设色温，过程中不能再修改标定结果。
- [ISP\\_AWB\\_ATTR\\_S](#) 结构体的 bGainNorm 参数会影响白平衡增益值。bGainNorm 为 0 时，最小增益为 0x100；bGainNorm 为 1 时，最小增益和 sensor 黑电平相关。
- s16Shift 参数决定光源点与 Planckian 曲线的位置关系，s16Shift 为负数时，光源点位于 Planckian 曲线左侧，计算的白平衡增益在预设色温下会稍微偏红；s16Shift 为正数时，光源点位于 Planckian 曲线右侧，计算的白平衡增益在预设色温下会稍微偏绿。

#### 【举例】

无

#### 【相关主题】

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

## 4.5 数据类型

### 4.5.1 Register

- [AWB\\_SENSOR\\_REGISTER\\_S](#)：定义 sensor 注册结构体。
- [AWB\\_SENSOR\\_EXP\\_FUNC\\_S](#)：定义 sensor 回调函数结构体。
- [AWB\\_SENSOR\\_DEFAULT\\_S](#)：定义 AWB 算法库的初始化参数结构体。
- [AWB\\_AGC\\_TABLE\\_S](#)：定义饱和度初始化参数结构体。
- [AWB\\_CCM\\_S](#)：定义 CCM 初始化参数结构体。

#### AWB\_SENSOR\_REGISTER\_S

##### 【说明】

定义 sensor 注册结构体。

##### 【定义】

```
typedef struct hiAWB_SENSOR_REGISTER_S
{
    AWB\_SENSOR\_EXP\_FUNC\_S stSnsExp;
} AWB_SENSOR_REGISTER_S;
```

##### 【成员】



成员名称	描述
stSnsExp	Sensor 注册的回调函数结构体。

【注意事项】

封装的目的是为了扩展。

【相关数据类型及接口】

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

## AWB\_SENSOR\_EXP\_FUNC\_S

【说明】

定义 sensor 回调函数结构体。

【定义】

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

【成员】

成员名称	描述
pfn_cmos_get_awb_default	获取 AWB 算法库的初始值的回调函数指针。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_SENSOR\\_REGISTER\\_S](#)

## AWB\_SENSOR\_DEFAULT\_S

【说明】

定义 AWB 算法库的初始化参数结构体。

【定义】

```
typedef struct hiAWB_SENSOR_DEFAULT_S
{
    HI_U16  u16WbRefTemp;
    HI_U16  au16GainOffset[4];
    HI_S32  as32WbPara[6];
}
```



```
AWB_AGC_TABLE_S stAgcTbl;  
AWB_CCM_S stCcm;  
} AWB_SENSOR_DEFAULT_S;
```

#### 【成员】

成员名称	描述
u16WbRefTemp	静态白平衡校正色温，取值范围为[0,0xFFFF]。
au16GainOffset	静态白平衡的 R、Gr、Gb、B 颜色通道的增益，取值范围为[0,0xFFF]，8bit 精度，最小取值为 0x100。
as32WbPara	校正工具给出的白平衡参数，取值范围为[0,0xFFFFFFFF]。
stAgcTbl	调节饱和初始化结构体。
stCcm	CCM 初始化结构体。

#### 【注意事项】

参考色温即静态白平衡校正的环境色温，需要提供色度计测量的实际值。

#### 【相关数据类型及接口】

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

## AWB\_AGC\_TABLE\_S

#### 【说明】

定义饱和度初始化参数结构体。

#### 【定义】

```
typedef struct hiAWB_AGC_TABLE_S  
{  
    HI_BOOL bValid;  
    HI_U8 au8Saturation[16];  
} AWB_AGC_TABLE_S;
```

#### 【成员】

成员名称	描述
bValid	该结构体的数据是否有效，取值范围为 0、1。
au8Saturation	根据增益动态调节饱和度的插值数组，取值范围为 [0,255]。

#### 【注意事项】



au8Saturation 为非单调递减序列。

#### 【相关数据类型及接口】

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

## AWB\_CCM\_S

#### 【说明】

定义 CCM 初始化参数结构体。

#### 【定义】

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

#### 【成员】

成员名称	描述
u16HighColorTemp	高色温，取值范围：[2800, 10000]。
au16HighCCM	高色温颜色还原矩阵，取值范围为[0, 0xFFFF]。
u16MidColorTemp	中色温，取值范围：[2400, u16HighColorTemp-400]。
au16MidCCM	中色温颜色还原矩阵，取值范围为[0, 0xFFFF]。
u16LowColorTemp	低色温，取值范围：[2000, u16MidColorTemp -400]。
au16LowCCM	低色温颜色还原矩阵，取值范围为[0,0xFFFF]。

#### 【注意事项】

推荐高色温 CCM 在 D50 光源标定，中色温 CCM 在 TL84 光源标定，低色温 CCM 在 A 光源标定。

#### 【相关数据类型及接口】

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

## 4.5.2 WB

- [ISP\\_AWB\\_ATTR\\_S](#)：定义 ISP 自动白平衡属性。
- [ISP\\_AWB\\_CBCR\\_TRACK\\_ATTR\\_S](#)：定义 Bayer 域统计信息的联动参数。



- [ISP\\_AWB\\_LUM\\_HISTGRAM\\_ATTR\\_S](#): 定义白平衡的亮度直方图统计参数。
- [ISP\\_AWB\\_ALG\\_TYPE\\_E](#): 定义白平衡的计算方式属性。
- [ISP\\_AWB\\_CT\\_LIMIT\\_ATTR\\_S](#): 定义白平衡的增益范围限制属性。
- [ISP\\_MWB\\_ATTR\\_S](#): 定义 ISP 手动白平衡属性。
- [ISP\\_WB\\_ATTR\\_S](#): 定义白平衡属性。
- [ISP\\_AWB\\_MULTI\\_LS\\_TYPE\\_E](#): 定义混合光源下的 AWB 策略。
- [ISP\\_AWB\\_ATTR\\_EX\\_S](#): 定义自动白平衡扩展属性。
- [ISP\\_AWB\\_EXTRA\\_LIGHTSOURCE\\_INFO\\_S](#): 定义独立光源点的信息。
- [ISP\\_AWB\\_IN\\_OUT\\_ATTR\\_S](#): 定义对场景做室内外判断的参数。
- [ISP\\_WB\\_INFO\\_S](#): 定义白平衡、饱和度、颜色校正信息。

## ISP\_AWB\_ATTR\_S

### 【说明】

定义 ISP 自动白平衡属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
bEnable	自动白平衡使能，默认 HI_TRUE。





成员名称	描述
u16RefColorTemp	校正静态白平衡系数的光源色温值。 推荐选用 5000K 附近光源进行静态白平衡校正。
au16StaticWB[4]	静态白平衡系数。 取值范围：[0x0, 0xFFF]。
as32CurvePara[6]	校准曲线的系数。 取值范围：as32CurvePara[3]!=0; as32CurvePara[4]=128。
enAlgType	自动白平衡算法类型选择。 取值范围：AWB_ALG_LOWCOST、 AWB_ALG_ADVANCE。
u8RGStrength	自动白平衡 R 通道校准强度。 取值范围：[0x0, 0xFF]。
u8BGStrength	自动白平衡 B 通道校准强度。 取值范围：[0x0, 0xFF]。
u16Speed	自动白平衡算法收敛速度。 取值范围：[0x0, 0xFFF]。
u16ZoneSel	自动白平衡算法全局或分区域计算的选择，取值范围：[0, 255]。
u16HighColorTemp	自动白平衡算法的色温上限。 推荐范围：[8000, 12000]。
u16LowColorTemp	自动白平衡算法的色温下限。 取值范围：[0x0, u8HighColorTemp)。
stCTLimit	白平衡算法环境色温超过预设色温范围后，增益计算模式及手动增益值设定。
bShiftLimitEn	自动白平衡算法的将超出色温曲线范围的点映射回色温曲线范围内的开关。
u8ShiftLimit	在指定范围外，将超出色温曲线范围的点映射回色温曲线范围。 取值范围：[0x0, 0xFF]。
bGainNormEn	自动白平衡算法的增益归一化的开关。
stCbCrTrack	Bayer 域统计信息与 ISO 的联动参数
stLumaHist	白平衡的亮度直方图统计参数



#### 【差异说明】

无

#### 【注意事项】

- 红外灯光源的光谱曲线特殊，导致红外光源下灰点分布不满足 AWB 标定的灰点范围。红外灯应用时如果需要使能 AWB 算法，建议设置 u16ZoneSel=0，采用近似灰度世界算法，不建议走常规 AWB 流程。
- bEanble 为 HI\_TRUE 时，AWB 正常工作，RGB 通道增益系数由 AWB 根据环境温度计算；bEnable 为 HI\_FALSE 时，AWB 不工作，RGB 通道增益系数固定，为校正的静态白平衡系数。
- u16RefColorTemp、au16StaticWB[4]、as32CurvePara[6]是用户通过校正工具得到的 AWB 参数，是 AWB 准确的前提。更换光学器件后，建议重新校正这三组参数。u16RefColorTemp 是进行静态白平衡校正光源的真实色温，推荐在 D50 光源下做静态白平衡校正。as32CurvePara[0]，as32CurvePara[1]，as32CurvePara[2]三个参数确定 Planckian 曲线，as32CurvePara[3]，as32CurvePara[4]，as32CurvePara[5]三个参数确定色温曲线，as32CurvePara[4]取值固定为 128。
- 自动白平衡 R/B 通道校准强度，一般情况不建议调整。  
可通过调整 R/B 通道校准强度使 AWB 校准偏强或者偏弱，0x80 表示标准强度。建议 u8RGStrength、u8BGStrength 两个参数取相同值，且小于等于 0x80。在低色温场景，校准强度小于 0x80 时图像偏红，大于 0x80 时图像偏蓝；在高色温场景，校准强度小于 0x80 时图像偏蓝，大于 0x80 时图像偏红。
- u16Speed 控制 AWB 增益的收敛速度。u16Speed 越大，切换光源时，AWB 收敛的速度越快，但帧间的波动幅度也变大；u16Speed 越小，切换光源时，AWB 收敛的速度越慢，但帧间的波动幅度变小，稳定性提高。
- u16ZoneSel 参数为 0 或 255 时，采用了近似灰度世界的算法；u16ZoneSel 为其他值时，AWB 算法会对所有统计块进行分类筛选，提高 AWB 精度。
- 自动白平衡算法的色温上限/下限，以 Kelvin 为单位。  
AWB 算法支持的最高/最低色温。若实际场景中的色温大于色温上限或小于色温下限，则 AWB 不能完全恢复，图像会偏向光源色，低色温下偏黄，高色温下偏蓝。
- bShiftLimitEn 决定 AWB 增益是否向 Planckian 曲线做映射。u8ShiftLimit 控制白色块的范围，取值较小时，对光源的支持范围较窄，精度较高；取值较大时，支持更多的光源，精度降低。
- 校正静态白平衡系数的光源色温值，要求在 AWB 算法设置的色温上下限范围内。
- bGainNormEn 使能，对 RGB 通道增益进行限制，可以改善低色温、低照度场景的信噪比。

#### 【相关数据类型及接口】

- [ISP\\_WB\\_ATTR\\_S](#)
- [ISP\\_AWB\\_CBCR\\_TRACK\\_ATTR\\_S](#)
- [ISP\\_AWB\\_LUM\\_HISTGRAM\\_ATTR\\_S](#)

## ISP\_AWB\_CBCR\_TRACK\_ATTR\_S

#### 【说明】



定义 Bayer 域统计信息的联动参数。

#### 【定义】

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

#### 【成员】

成员名称	描述
bEnable	Bayer 域统计信息参数与环境照度、色温的联动使能开关。
au16CrMax[-]	不同照度下的 CrMax 取值，取值范围：[0x0, 0xFFFF]。
au16CrMin[-]	不同照度下的 CrMin 取值，取值范围：[0x0, au16CrMax]。
au16CbMax[-]	不同照度下的 CbMax 取值，取值范围：[0x0, 0xFFFF]。
au16CbMin[-]	不同照度下的 CbMin 取值，取值范围：[0x0, au16CbMax]。

#### 【注意事项】

- 统计参数联动使能后，AWB 算法会根据环境照度、色温、用户设置的 au16CrMax 数组等实时计算 CrMax、CrMin、CbMax、CbMin 四个参数，并配置相应的逻辑寄存器，此时用户通过 PQ Tools 设置以上四个统计参数不生效。
- 统计参数联动使能，AWB 算法计算的 CrMax 等统计参数和环境色温相关。低色温时，白点范围较宽；中高色温时，白点范围较窄。
- 用户手动配置 CrMax、CrMin、CbMax、CbMin 四个参数时，需要先关闭联动功能。
- au16CrMax[0]、au16CrMin[0]、au16CbMax[0]、au16CbMin[0]的取值可在 AWB 参数标定时确定。用户确定了支持的色温范围后，捕获高低色温的 RAW 图片，计算白色区域的 R/G、B/G 值。au16CrMax[0]和 au16CbMin[0]分别对应低色温的 R/G、B/G；au16CrMin[0]和 au16CbMax[0]分别对应高色温的 R/G、B/G。推荐用户设置的 Cr、Cb 范围稍大于 RAW 图片统计的 R/G、B/G 取值范围。
- 建议在低色温(钠灯)环境标定 au16CrMax、au16CbMin 数组。用户统计不同照度下白色区域的 R/G、B/G 值，设置 au16CrMax、au16CbMin 数组。推荐用户设置的 Cr、Cb 范围稍大于 RAW 图片统计的 R/G、B/G 取值范围。



- 因低照度下环境色温多在 5000K 以下，au16CrMin、au16CbMax 两个数组的取值可设为常数。

表4-1 au16CrMax [16]在不同的增益情况下的设置值（仅供参考）

au16CrMax	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMax [0]	1	0x150
au16CrMax [1]	2	0x150
au16CrMax [2]	4	0x150
au16CrMax [3]	8	0x150
au16CrMax [4]	16	0x150
au16CrMax [5]	32	0x150
au16CrMax [6]	64	0x170
au16CrMax [7]	128	0x1B0
au16CrMax [8]	256	0x230
au16CrMax [9]	512	0x330
au16CrMax [10]	1024	0x400
au16CrMax [11]	2048	0x400
au16CrMax [12]	4096	0x400
au16CrMax [13]	8192	0x400
au16CrMax [14]	16384	0x400
au16CrMax [15]	32768	0x400

表4-2 au16CrMin [16]在不同的增益情况下的设置值（仅供参考）

au16CrMin	Again*Dgain*ISPDgain(倍数)	设置值
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	0x3D
au16CrMin [7]	128	0x37



au16CrMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMin [8]	256	0x2B
au16CrMin [9]	512	0x18
au16CrMin [10]	1024	0x18
au16CrMin [11]	2048	0x18
au16CrMin [12]	4096	0x18
au16CrMin [13]	8192	0x18
au16CrMin [14]	16384	0x18
au16CrMin [15]	32768	0x18

表4-3 au16CbMax [16]在不同的增益情况下的设置值（仅供参考）

au16CbMax	Again*Dgain*ISPDgain(倍数)	设置值
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	0x130
au16CbMax [7]	128	0x150
au16CbMax [8]	256	0x190
au16CbMax [9]	512	0x210
au16CbMax [10]	1024	0x310
au16CbMax [11]	2048	0x400
au16CbMax [12]	4096	0x400
au16CbMax [13]	8192	0x400
au16CbMax [14]	16384	0x400
au16CbMax [15]	32768	0x400



表4-4 au16CbMin [16]在不同的增益情况下的设置值（仅供参考）

au16CbMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CbMin [0]	1	0x30
au16CbMin [1]	2	0x30
au16CbMin [2]	4	0x30
au16CbMin [3]	8	0x30
au16CbMin [4]	16	0x30
au16CbMin [5]	32	0x30
au16CbMin [6]	64	0x2D
au16CbMin [7]	128	0x27
au16CbMin [8]	256	0x1B
au16CbMin [9]	512	0x10
au16CbMin [10]	1024	0x10
au16CbMin [11]	2048	0x10
au16CbMin [12]	4096	0x10
au16CbMin [13]	8192	0x10
au16CbMin [14]	16384	0x10
au16CbMin [15]	32768	0x10

#### 【相关数据类型及接口】

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

### ISP\_AWB\_LUM\_HISTGRAM\_ATTR\_S

#### 【说明】

定义白平衡的亮度直方图统计参数。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
bEnable	亮度是否影响分块的权重。
enOpType	自动模式下，AWB 算法对分块统计结果做亮度直方图统计，自动分配亮度权重。手动模式下，用户设置亮度直方图的门限和权重。
au8HistThresh[6]	用户设置亮度直方图的门限，仅手动模式有效。
au16HistWt[6]	用户设置亮度直方图的权重，自动模式和手动模式下均有效。8bit 小数精度。

#### 【注意事项】

- RGB 域 AWB 统计输出不包括亮度信息，该功能无效。
- au8HistThresh[0]固定为 0，au8HistThresh[5]固定为 0xFF。au8HistThresh[i+1]取值应大于 au8HistThresh[i]。
- au16HistWt 参数用来设置 au8HistThresh 的权重。对应用户可通过 au16HistWt 设置实现亮区优先、暗区优先。

#### 【相关数据类型及接口】

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

## ISP\_AWB\_ALG\_TYPE\_E

#### 【说明】

定义白平衡的算法类型属性。

#### 【定义】

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

#### 【成员】

成员名称	描述
AWB_ALG_LOW COST	改进的灰度世界算法，根据统计信息自动计算区间权重的 AWB 算法。
AWB_ALG_ADVANCE	对统计信息分类，再次筛选白色块的 AWB 算法。
AWB_ALG_BUTT	无效。



#### 【注意事项】

AWB\_ALG\_LOWCOST CPU 占用低：AWB 不同照度的稳定性较好，对光源的适应性稍好。

AWB\_ALG\_ADVANCE CPU 占用高：高低色温的极低照度情况下，保留轻微光源色；提高了大面积纯色场景（室外大面积绿色、室内大面积肤色等）AWB 算法准确度。

#### 【相关数据类型及接口】

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

### ISP\_AWB\_CT\_LIMIT\_ATTR\_S

#### 【说明】

定义白平衡的增益范围限制属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
bEnable	白平衡的增益范围限制开关。
enOpType	自动或手动设定白平衡的增益限制。
u16HighRgLimit	手动模式下用户设定高色温下的最大 R 增益，8bit 小数精度，取值范围：[0x0, 0xFFFF]。
u16HighBgLimit	手动模式下用户设定高色温下的最小 B 增益，8bit 小数精度，取值范围：[0x0, 0xFFFF]。
u16LowRgLimit	手动模式下用户设定低色温下的最小 R 增益，8bit 小数精度，- 取值范围：[0x0, u16HighRgLimit)。
u16LowBgLimit	手动模式下用户设定低色温下的最大 B 增益，8bit 小数精度，取值范围：(u16HighBgLimit, 0xFFFF]。

#### 【注意事项】





- ISP\_AWB\_CT\_LIMIT\_ATTR\_S 结构体定义环境色温超过用户设定的色温范围时，AWB 应采取的动作。AWB 检测到环境色温超过用户设置的上下限时，相关参数才会生效。
- 支持自动模式和手动模式选择。自动模式下，AWB 根据用户校正的色温曲线计算色温上下限色温的 AWB 增益，以限制 R, B 通道增益；手动模式下，高色温时，AWB 调用 u16HighRgLimi、u16HighBgLimit 参数限制 R, B 通道增益，低色温时，AWB 调用 u16LowRgLimit、u16LowBgLimit 参数限制 R, B 通道增益。
- 建议用户在进行 AWB 色温曲线校正时，确定了 AWB 工作的色温范围后，再确定 u16HighRgLimit、u16HighBgLimit、u16LowRgLimit、u16LowBgLimit 四个参数取值。Rg 对应 Planckian 曲线的横坐标值，Bg 对应 Planckian 曲线的纵坐标值。

【相关数据类型及接口】

ISP\_AWB\_ATTR\_S

## ISP\_MWB\_ATTR\_S

【说明】

定义 ISP 手动白平衡属性。

【定义】

```
typedef struct hiISP_MWB_ATTR_S
{
    HI_U16 u16Rgain;
    HI_U16 u16Grgain;
    HI_U16 u16Gbgain;
    HI_U16 u16Bgain;
} ISP_MWB_ATTR_S;
```

【成员】

成员名称	描述
u16Rgain	手动白平衡红色通道增益，8bit 小数精度。 取值范围：[0x0,0xFFF]。
u16Grgain	手动白平衡绿色通道增益，8bit 小数精度。 取值范围：[0x0,0xFFF]。
u16Gbgain	手动白平衡绿色通道增益，8bit 小数精度。 取值范围：[0x0,0xFFF]。
u16Bgain	手动白平衡蓝色通道增益，8bit 小数精度。 取值范围：[0x0,0xFFF]。

【注意事项】



无

【相关数据类型及接口】

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

## ISP\_WB\_ATTR\_S

【说明】

定义白平衡属性。

【定义】

```
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;
```

【成员】

成员名称	描述
bByPass	白平衡模块 Bypass 使能，默认值 HI_FALSE。
enOpType	自动白平衡和手动白平衡切换。
stManual	手动白平衡参数。
stAuto	自动白平衡参数。

【注意事项】

bByPass 为 HI\_TRUE 时，WB 的其它参数设置不生效，RGB 通道增益系数固定为 0x100。

【相关数据类型及接口】

- [ISP\\_MWB\\_ATTR\\_S](#)
- [ISP\\_AWB\\_ATTR\\_S](#)

## ISP\_AWB\_MULTI\_LS\_TYPE\_E

【说明】

定义混合光源下的 AWB 策略。

【定义】

```
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;
```

#### 【成员】

成员名称	描述
AWB_MULTI_LS_SAT	混合光源下自动调整饱和度。
AWB_MULTI_LS_CCM	混合光源下自动调整 CCM。
AWB_MULTI_LS_BUTT	无效。

#### 【注意事项】

- AWB\_MULTI\_LS\_CCM 对图像饱和度损失较少，但会改变颜色的色调。
- AWB\_MULTI\_LS\_SAT 方式降低图像整体饱和度，改善混合光源下的偏色程度。

#### 【相关数据类型及接口】

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

### ISP\_AWB\_ATTR\_EX\_S

#### 【说明】

定义自动白平衡扩展属性。

#### 【定义】

```
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 aul6MultiCTBin[8];  
    HI_U16 aul6MultiCTWt[8];  
    HI_BOOL bFineTunEn;  
    HI_U8 u8FineTunStrength;
```



```
} ISP_AWB_ATTR_EX_S;
```

### 【成员】

成员名称	描述
u8Tolerance	自动白平衡调整的偏差范围，检测误差在门限范围内时，AWB 不动作。
u8ZoneRadius	自动白平衡统计中对像素分类时用的距离范围。该值越小，AWB 精度越高，但会降低 AWB 算法稳定性。
u16CurveLLimit	自动白平衡色温曲线的左边界限。取值范围[0x0, 0x100]。
u16CurveRLimit	自动白平衡色温曲线的右边界限。取值范围[0x100, 0xFFFF]。
bExtraLightEn	自动白平衡计算时是否考虑色温曲线外的独立光源点。
stLightInfo[4]	色温曲线外的独立光源点的信息，最多可以添加 4 个。
stInOrOut	自动白平衡对场景做室内外判断的参数。
bMultiLightSourceEn	自动白平衡检测当前场景是否为混合光源，根据混合光源程度调整饱和度或 CCM。
enMultiLSType	混合光源调整策略选择，支持调整饱和度或 CCM。
u16MultiLSScaler	混合光源下，饱和度或 CCM 最大调整幅度。实际调整幅度还和场景混合光源程度有关。 取值范围为[0x0, 0x100]。
au16MultiCTBin[8]	混合光源下的色温分段参数。 取值范围：要求为单调递增序列。
au16MultiCTWt[8]	混合光源下的色温权重参数。 取值范围： [0x0, 0x400]。
bFineTunEn	自动白平衡特殊色检测开关，包括肤色检测等。
u8FineTunStrength	肤色、蓝色等单色检测的强度。仅在 bFineTunEn 使能时有效。 取值范围： [0x0, 0xFF]。

图4-3 色温曲线的参数示意

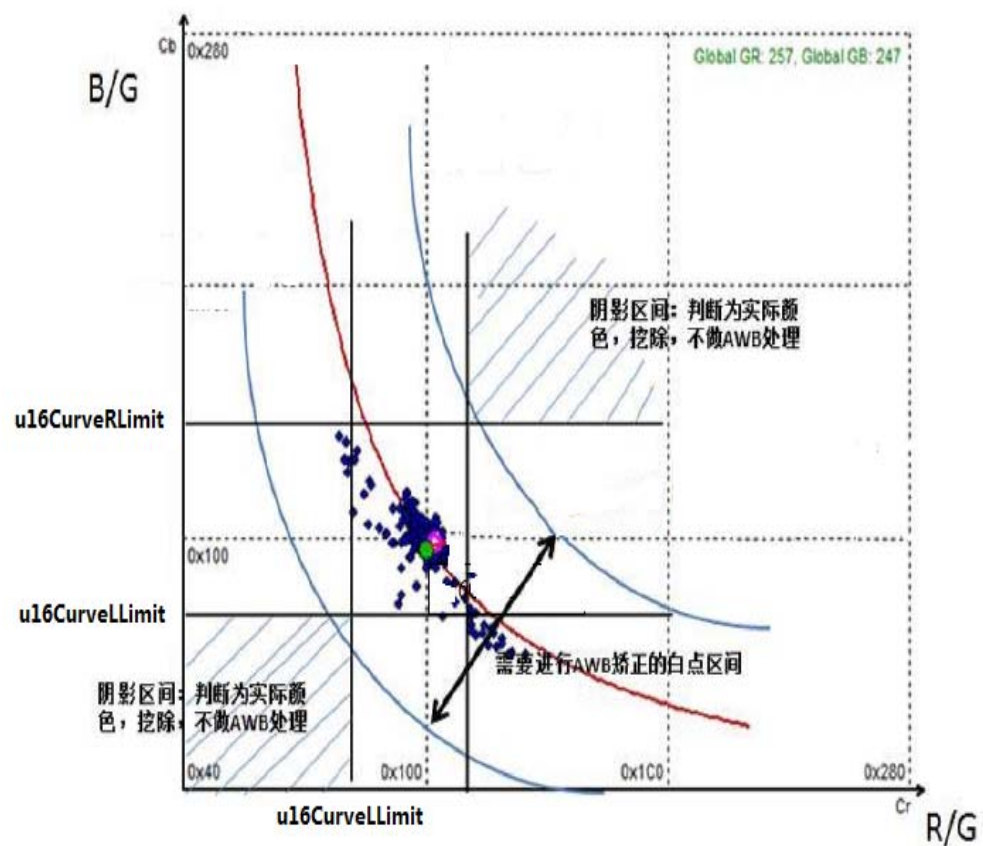
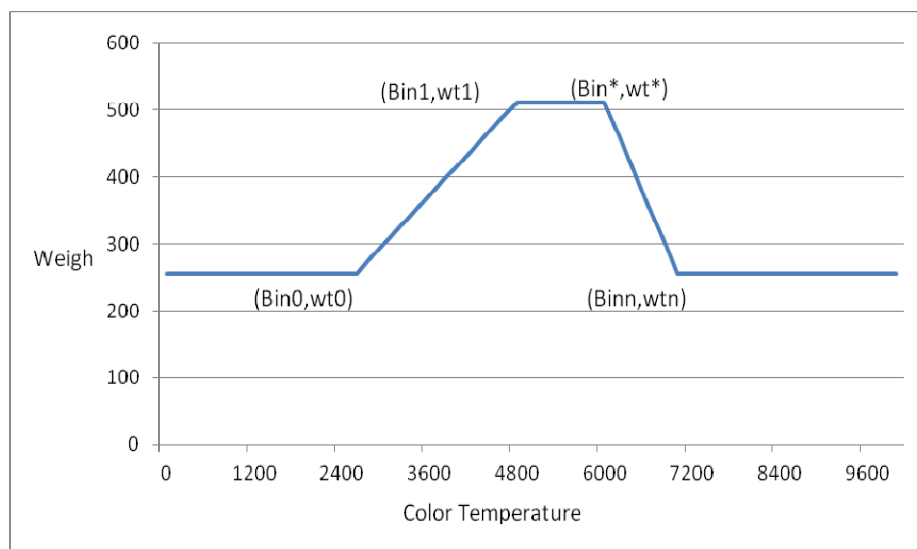


图4-4 混合光源场景色温权重设置示例说明 (n 为色温分段点个数)



【注意事项】



- **u8Tolerance** 是 AWB 的灵敏度参数。该值为 0 时，AWB 系数实时更新，CPU 占用较高；该值较大时，AWB 在检测到环境色温变化时再更新 AWB 系数，CPU 占用较低，但在环境色温微调时，AWB 可能出现轻微的偏色。推荐室外应用时 **u8Tolerance** 置为 0，室内应用时置为 2。
- **u8ZoneRadius** 是选择光源环境白色块的色差半径。**u8ZoneRadius** 越大，白色块的限制条件越宽，找到的白色块较多，稳定性稍好，精度稍低；**u8ZoneRadius** 越小，白色块的限制条件越严格，找到的白色块较少，AWB 精度更高。在混合光源的应用场景，推荐 **u8ZoneRadius** 比普通场景稍大。
- **u16CurveLLimit** 取值  $\leq 0x100$ ，**u16CurveRLimit** 取值  $\geq 0x100$ 。**u16CurveLLimit** 和 **u16CurveRLimit** 两个参数用来排除颜色块。如图 4-3 所示，**u16CurveLLimit** 排除了左侧的阴影区域，**u16CurveRLimit** 排除了右侧的阴影区域。
- AWB 算法支持在色温曲线外，再增加独立光源点信息，提升特殊光源下的 AWB 表现。
- **bMultiLightSourceEn** 使能后，AWB 算法判断场景是否混合光源，在混合光源场景，自动降低饱和度或调整 CCM，以减弱偏色程度。最大调整幅度由 **u16MultiLSScaler** 决定。调整 CCM 方式会改变色调。
- **enMultiLSType=AWB\_MULTI\_LS\_SAT** 时，调整饱和度来改善混合光源场景的偏色。**u16MultiLSScaler/0x100** 表示最终饱和度的增益系数，取值越大，图像饱和度越高。**u16MultiLSScaler=0xC0**，表示饱和度的增益系数是  $0xC0/0x100 = 0.75$ ；**u16MultiLSScaler=0x0**，表示饱和度的增益系数是  $0x0/0x100 = 0$ ，在混合光源下图像可能变黑白。
- **enMultiLSType=AWB\_MULTI\_LS\_CCM** 时，调整 CCM 来改善混合光源场景的偏色。**u16MultiLSScaler/0x100** 表示 CCM 的调整幅度，取值越大，图像饱和度越低。**u16MultiLSScaler=0xC0**，表示 CCM 的最大调整幅度为  $0xC0$ ；**u16MultiLSScaler=0x0**，表示 CCM 的最大调整幅度为 0，等效于 **bMultiLightSourceEn=0**。
- 海思 AWB 算法在 WDR 模式自动关闭混合光源判断功能。
- **au16MultiCTBin**、**au16MultiCTWt** 两组参数的合理配置，可以改善混合光源高色温或低色温区域颜色表现。在非混合光源场景，参数配置不生效。图 4-4 说明色温和权重配置的映射关系。
- 如果混合光源场景希望使能色温权重来改善 AWB 表现，但不做饱和度或 CCM 调解，可配置如下：**bMultiLightSourceEn=HI\_TRUE**，**enMultiLSType=AWB\_MULTI\_LS\_SAT**，**u16MultiLSScaler=0x100**。
- **bFineTunEn** 打开，AWB 会自动检测肤色等特殊色，改善肤色场景 AWB 表现，提高 AWB 精度。但可能在蓝色背景+3500K 光源时发生误判，导致图像轻微偏黄。**u8FineTunStrength** 调整肤色检测的强度，取值越大，肤色场景 AWB 表现越好，但肤色误判时的副作用越明显。推荐采用默认值  $0x80$ 。

#### 【相关数据类型及接口】

- [ISP\\_AWB\\_EXTRA\\_LIGHTSOURCE\\_INFO\\_S](#)
- [ISP\\_AWB\\_IN\\_OUT\\_ATTR\\_S](#)

## ISP\_AWB\_EXTRA\_LIGHTSOURCE\_INFO\_S

#### 【说明】



定义独立光源点的信息。

#### 【定义】

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

#### 【成员】

成员名称	描述
u16WhiteRgain	增加或删除区域的中心 Cr 坐标，8bit 小数精度。 取值范围[0x0, 0xFFFF]。
u16WhiteBgain	增加或删除区域的中心 Cb 坐标，8bit 小数精度。 取值范围[0x0, 0xFFFF]。
u16ExpQuant	对外界环境亮度做判断，暂不支持，不需要配置。
u8LightStatus	光源点状态。 0：不使能； 1：增加光源点； 2：删除干扰色；
u8 Radius	增加或删除区域的半径。 取值范围[0x0, 0xFF]。

#### 【注意事项】

- 独立光源点实现以下功能：用户设定 CbCr 坐标的某个区域，指定该区域为特殊光源下的白色表现，增加为独立光源点；或指定该区域为特定光源下的干扰色，进行删除。
- 通过 u8LightStatus 参数的设置实现功能选择。u8LightStatus 设置为 1 时，在用户校正的 Plankcian 曲线外，增加特殊光源点，以优化该光源下颜色表现，不影响设备在其它光源下的表现；u8LightStatus 设置为 2 时，支持用户删除特定光源的某个干扰色(肤色、绿色、蓝天等)。
- 建议删除干扰色仅用来优化特定环境的颜色表现，否则会影响其它光源的表现。在 CbCr 坐标，5500K 光源下的肤色与 3500K 光源的白色重合度很高，在 5500K 环境删除肤色的干扰，会导致 3500K 光源下偏色。
- u16WhiteRgain、u16WhiteBgain 的取值。在该光源下捕获 ColorChecker Raw 数据，通过 Calibration Tool Staitic WB 选项校准得到的 Rgain、Bgain 即



u16WhiteRgain、u16WhiteBgain 坐标值。删除干扰色，可选择干扰色区域进行静态白平衡校正，来确定 u16WhiteRgain、u16WhiteBgain 坐标值。

- u8 Radius 取值由用户设置，增加光源点，建议取值在[0x8, 0x10]范围内，删除干扰色，尽量避免[u16WhiteRgain、u16WhiteBgain]为中心，u8 Radius 为半径的圆形区域与 Plankcian 曲线重合。
- 当前支持定义最多 4 个光源色，4 个相互独立。4 个光源点需要避免交集，特别需要避免以下情况：CbCr 坐标的单个点，既指定为增加的独立光源点，又指定为删除的干扰色。

#### 【相关数据类型及接口】

#### ISP\_AWB\_ATTR\_EX\_S

### ISP\_AWB\_IN\_OUT\_ATTR\_S

#### 【说明】

定义对场景做室内外判断的参数。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
bEnable	对场景做室内外判断开关。
enOpType	对场景做室内外判断类型(自动或手动)。
bOutdoorStatus	当前的室内外判断状态，1 表示室外，0 表示室内。
u32OutThresh	室内外场景判定设置的阈值，以单位为 us 的曝光时间，小于该阈值则认为是室外。
u16LowStart	将位于低色温范围内的像素权重调低，低色温区起始值，建议 5000K。





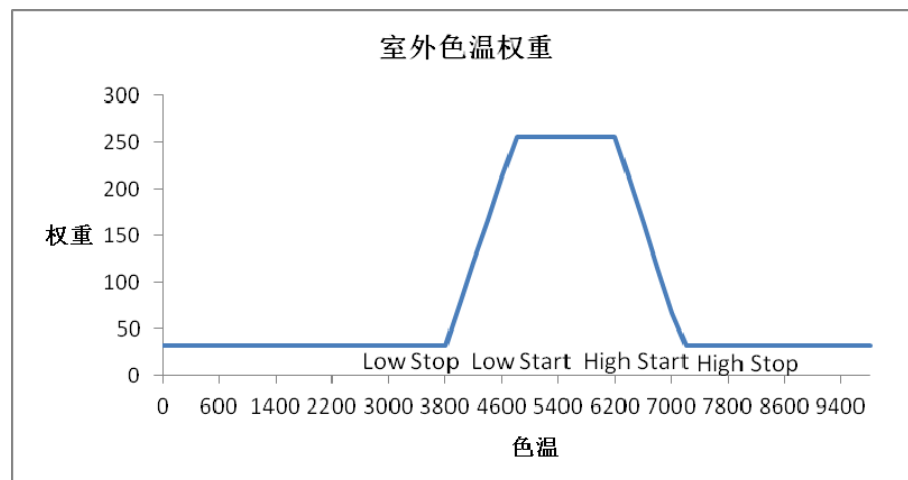
成员名称	描述
u16LowStop	将位于低色温范围内的像素权重调低，低色温区终止值，建议 4500K。 取值范围：(0, u16LowStart)。
u16HighStart	将位于高色温范围内的像素权重调低，高色温区起始值，建议 6500K。 取值范围：(u16LowStart, 0xFFFF]。
u16HighStop	将位于高色温范围内的像素权重调低，高色温区终止值，建议 8000K。 取值范围：(u16HighStart, 0xFFFF]。
bGreenEnhanceEn	在绿色植物场景下是否对绿色通道增强。
u8OutShiftLimit	判断为室外场景时，AWB 算法的白点范围限制 Shift= u8OutShiftLimit。 取值范围：[0x0, 0xFF]。

#### 【注意事项】

- 室内外判断类型为自动时，bOutdoorStatus 参数只读；室内外判断类型为手动时，bOutdoorStatus 参数只写，由用户指定当前场景是室内或室外。
- u32OutThresh 需要根据 sensor 的感光特性进行调整。
- 用户调用自己的 AE 库，且室内外判断类型为自动时，请务必将当前曝光时间和 ISO 传递给 AWB 库。
- 设置色温范围的 u16LowStart、u16LowStop、u16HighStart、u16HighStop 等 4 个参数，和 sensor 校正相关，建议用户针对 sensor 做微调。四个参数的取值范围要求：u16LowStop < u16LowStart < u16HighStart < u16HighStop。如果设置的室外色温范围[u16LowStart, u16HighStart]较宽，室外的大面积纯色场景颜色表现可能会稍差；如果设置的室外色温范围[u16LowStart, u16HighStart]较窄，在日落前等高低色温场景，AWB 不能完全恢复，低色温下稍微偏黄，高色温下稍微偏蓝。
- AWB 校正参数调整后，u16LowStart、u16LowStop、u16HighStart、u16HighStop 等 4 个参数要再次进行调整，以达到最佳效果。
- Shift 取值影响 AWB 算法表现。普通场景 u8ShiftLimit、u8OutShiftLimit 两个参数的取值大小对 AWB 结果影响不大。特殊光源、大面积单色场景，u8ShiftLimit、u8OutShiftLimit 两个参数的取值大小对 AWB 结果影响较大。两个 Shift 参数取值较小时，单色表现好，但特殊光源下容易偏色；两个 Shift 参数取值较大时，AWB 表现稳定性好，但大面积绿色、黄色等场景可能偏色。



图4-5 室外色温范围参数的意义



#### 【相关数据类型及接口】

[ISP\\_AWB\\_ATTR\\_EX\\_S](#)

### ISP\_WB\_INFO\_S

#### 【说明】

定义白平衡，饱和度，颜色校正信息。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
u16Rgain	当前 R 通道增益值，8bit 小数精度。



成员名称	描述
u16GrGain	当前 Gr 通道增益值，8bit 小数精度。
u16GbGain	当前 Gb 通道增益值，8bit 小数精度。
u16BGain	当前 B 通道增益值，8bit 小数精度。
u16Saturation	当前饱和度值，有效范围为[0, 255]。
u16ColorTemp	当前色温值。
au16CCM[9]	当前颜色校正矩阵值，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。
u16LS0CT	混合光源场景，主光源色温。
u16LS1CT	混合光源场景，次要光源色温。
u16LS0Area	混合光源场景，主光源面积。取值范围为[0x0, 0xFF]。
u16LS1Area	混合光源场景，次要光源面积。取值范围为[0x0, 0xFF]。
u8MultiDegree	当前场景是混合光源的概率。

#### 【注意事项】

- u8MultiDegree 非 0 时，表示场景是混合光源。该值越大，场景为混合光源的概率越大。
- u16LS0CT、u16LS1CT、u16LS0Area、u16LS1Area 等查询结果仅在 u8MultiDegree 非 0 时有效。
- 混合光源检测仅在室内场景打开，AWB 判断为室外场景后，自动关闭混合光源检测，以避免饱和度降低。
- 混合光源概率说明：主光源和次要光源的亮度、色温差等条件影响混合光源的概率计算。检测到场景内存在主光源和次要光源两种光源后，主光源和次要光源的色温差异越大、亮度差异越小，混合光源概率越大。
- u16LS0Area 和 u16LS1Area 之和有可能大于白平衡分区个数，原因是部分区域可能受到两种光源的影响，算法不将重叠区域严格识别，会叠加到主光源、次要光源。

#### 【相关数据类型及接口】

无



# 5 CCM

## 5.1 概述

sensor 对光谱的响应，在 RGB 各分量上与人眼对光谱的响应通常是有偏差的，通常通过一个色彩校正矩阵校正光谱响应的交叉效应和响应强度，使前端捕获的图片与人眼视觉在色彩上保持一致。

## 5.2 重要概念

- 色彩还原：通常通过一个色彩校正矩阵校正光谱响应的交叉效应和响应强度，使 ISP 处理后的图片与人眼视觉在色彩上保持一致。
- 饱和度：也称色彩的纯度。取决于该色中含色成分和消色成分(灰色)的比例。含色成分越大，饱和度越大；消色成分越大，饱和度越小。

## 5.3 功能描述

离线校准工具 Calibration Tool 支持 3x3 Color Correction Matrix 的预校正。在 ISP 运行时，FW 根据当前的光照强度，调整饱和度，实现 CCM（Color Correction Matrix）矩阵系数的动态调整。CCM 矩阵如图 5-1 所示。

图5-1 CCM 矩阵

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



## 5.4 API 参考

饱和度、色调和 CCM 接口只是针对海思 AWB 库，如果客户自己实现 AWB 库，不需要关注这些接口，且无法使用这些接口。

- [HI\\_MPI\\_ISP\\_SetSaturationAttr](#): 设置颜色饱和度属性。
- [HI\\_MPI\\_ISP\\_GetSaturationAttr](#): 获取颜色饱和度属性。
- [HI\\_MPI\\_ISP\\_SetCCMArr](#): 设置颜色校正基础矩阵。
- [HI\\_MPI\\_ISP\\_GetCCMArr](#): 获取颜色校正基础矩阵。

### HI\_MPI\_ISP\_SetSaturationAttr

#### 【描述】

设置颜色饱和度属性。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstSatAttr	颜色饱和度属性。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

无

#### 【需求】

- 头文件: [hi\\_awb\\_comm.h](#)、[mpi\\_awb.h](#)
- 库文件: [libisp.a](#)

#### 【注意】

无

#### 【举例】



无

【相关主题】

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

## HI\_MPI\_ISP\_GetSaturationAttr

【描述】

获取颜色饱和度属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetSaturationAttr(ISP_DEV IspDev, ISP_SATURATION_ATTR_S  
*pstSatAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstSatAttr	颜色饱和度属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无



【相关主题】

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

## HI\_MPI\_ISP\_SetCCMAAttr

【描述】

设置颜色矩阵。

【语法】

```
HI_S32 HI_MPI_ISP_SetCCMAAttr(ISP_DEV IspDev, const ISP_COLORMATRIX_ATTR_S  
*pstCCMAAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstCCMAAttr	颜色矩阵。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

【注意】

- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。
- 可根据当前色温，设置不同的 CCM，从而在高低色温下都达到较好的颜色还原。
- 该 MPI 支持高中低三个不同的色彩还原矩阵。只需要在高色温，中色温，低色温下分别校正一组 CCM 矩阵。



- 手动模式下，ISP 系统生效的 CCM 即用户手动配置的 CCM，有利于 CCM 精调，因此建议在手动模式下校正高中低三个光源下的 CCM 矩阵，得到三组 CCM 矩阵后再通过 MPI 写入，验证多个光源下的 CCM 组和效果。
- 手动 CCM 模式下，饱和度调整有效。在特定光源下做 CCM 精调时，需要先确认饱和度是否期望值。

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetCCMAAttr

【描述】

获取颜色矩阵。

【语法】

```
HI_S32 HI_MPI_ISP_GetCCMAAttr(ISP_DEV IspDev, ISP\_COLORMATRIX\_ATTR\_S  
*pstCCMAAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstCCMAAttr	颜色矩阵。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h





- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 5.5 数据类型

- [ISP\\_SATURATION\\_ATTR\\_S](#)：定义 ISP 颜色饱和度属性。
- [ISP\\_SATURATION\\_MANUAL\\_S](#)：定义手动饱和度属性。
- [ISP\\_SATURATION\\_AUTO\\_S](#)：定义自动饱和度属性。
- [ISP\\_COLORMATRIX\\_ATTR\\_S](#)：定义 ISP 颜色矩阵属性。
- [ISP\\_COLORMATRIX\\_AUTO\\_S](#)：定义自动颜色校正矩阵属性。
- [ISP\\_COLORMATRIX\\_MANUAL\\_S](#)：定义手动颜色校正矩阵属性。

### ISP\_SATURATION\_ATTR\_S

【说明】

定义 ISP 颜色饱和度属性。

【定义】

```
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;
```

【成员】

成员名称	描述
enOpType	设置饱和度类型(手动或自动)。
stManual	手动饱和度参数。
stAuto	自动饱和度参数。

【注意事项】



- 设置饱和度功能分为自动和手动：
  - enOpType 为 OP\_TYPE\_AUTO，使用自动饱和度调节功能。  
此时饱和度值会根据系统增益自动调节，饱和度与系统增益的关系请参见成员变量 au8Sat [16]的描述。
  - enOpType 为 OP\_TYPE\_MANUAL，使用手动饱和度调节功能。  
此时变量 u8Sat[16]无效。

【相关数据类型及接口】

- [ISP\\_SATURATION\\_AUTO\\_S](#)
- [ISP\\_COLORMATRIX\\_MANUAL\\_S](#)

## ISP\_SATURATION\_MANUAL\_S

【说明】

定义手动饱和度属性。

【定义】

```
typedef struct hiISP_SATURATION_MANUAL_S
{
    HI_U8 u8Saturation;
} ISP_SATURATION_MANUAL_S;
```

【成员】

成员名称	描述
u8Saturation	手动饱和度值。 取值范围：[0x0, 0xFF]。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_SATURATION\\_AUTO\\_S](#)

## ISP\_SATURATION\_AUTO\_S

【说明】

定义自动饱和度属性。

【定义】

```
typedef struct hiISP_SATURATION_AUTO_S
{
    HI_U8 au8Sat[ISP_AUTO_ISO_STENGTH_NUM];
```



```
} ISP_SATURATION_AUTO_S;
```

#### 【成员】

成员名称	描述
au8Sat[]	自动饱和度值。 推荐配置为递减序列。

表5-1 au8Sat[16]在不同的增益情况下的设置值，以 mn34220 为例

au8Sat	Again*Dgain*ISPDgain(倍数)	推荐设置值
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

#### 【注意事项】

无

#### 【相关数据类型及接口】

[ISP\\_SATURATION\\_MANUAL\\_S](#)



## ISP\_COLORMATRIX\_ATTR\_S

### 【说明】

定义 ISP 颜色矩阵属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
enOpType	颜色校正矩阵类型(手动或自动)。
stManual	手动颜色校正矩阵。
stAuto	自动颜色校正矩阵。

### 【注意事项】

无

### 【相关数据类型及接口】

- [ISP\\_COLORMATRIX\\_AUTO\\_S](#)
- [ISP\\_COLORMATRIX\\_MANUAL\\_S](#)

## ISP\_COLORMATRIX\_AUTO\_S

### 【说明】

定义自动颜色校正矩阵属性。

### 【定义】

```
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;

```

**【成员】**

成员名称	描述
bISOActEn	是否使能低照度下 CCM Bypass 功能。
bTempActEn	是否使能高、低色温下 CCM Bypass 功能。
u16HighColorTemp	高色温。 取值范围：[2800,10000]。
au16HighCCM[9]	高色温下的颜色校正矩阵，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。 取值范围：[0x0,0xFFFF]。
u16MidColorTemp	中等色温。取值范围：[2400, u16HighColorTemp-400]。
au16MidCCM[9]	中等色温下的颜色校正矩阵，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。 取值范围：[0x0,0xFFFF]。
u16LowColorTemp	低色温。取值范围：[2000, u16MidColorTemp-400]。
au16LowCCM[9]	低色温下的颜色校正矩阵，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。 取值范围：[0x0, 0xFFFF]。

**【注意事项】**

- 极低低照度下 Bypass CCM，物体的颜色更真实。用户关闭该功能后，极低照度下，饱和度数组仍生效。
- 在高色温(大于 10000K)和低色温场景(小于 2300K)，AWB 可能未完全恢复，此时采用高饱和度 CCM 矩阵，会加剧 AWB 的偏色程度。比如钠灯场景，AWB 校正后白色区域蓝色稍有不足，采用高饱和度 CCM 矩阵，白色区域的蓝色分量可能为 0，图像整体偏黄。使能 bTempActEn，在高低色温下自动 BypassCCM，优化类似场景颜色表现。
- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。
- u16HighColorTemp、u16MidColorTemp 和 u16LowColorTemp 必须满足如下条件：
  - u16HighColorTemp - u16MidColorTemp ≥ 400
  - u16MidColorTemp – u16LowColorTemp ≥ 400

**【相关数据类型及接口】**

[ISP\\_COLORMATRIX\\_MANUAL\\_S](#)



## ISP\_COLORMATRIX\_MANUAL\_S

### 【说明】

定义手动颜色校正矩阵属性。

### 【定义】

```
typedef struct hiISP_COLORMATRIX_MANUAL_S
{
    HI_BOOL bSatEn;
    HI_U16  au16CCM[9];
} ISP_COLORMATRIX_MANUAL_S;
```

### 【成员】

成员名称	描述
bSatEn	手动 CCM 模式下，饱和度是否生效。
au16CCM[9]	手动颜色校正矩阵，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。 取值范围：[0x0,0xFFFF]。

### 【注意事项】

手动 CCM 模式下，用户可通过 bSatEn 参数选择饱和度是否生效。bSatEn = 1，生效的 CCM = Manual CCM \* Saturation CCM，用户可用该功能确定不同照度下的饱和度数组；bSatEn = 0，生效的 CCM = Manual CCM。

### 【相关数据类型及接口】

[ISP\\_COLORMATRIX\\_AUTO\\_S](#)



# 6 IMP

IMP 是指影响图像效果的模块，对应 API 接口必须在调用 [HI\\_MPI\\_ISP\\_Init](#) 接口之后才能调用。

## 6.1 Sharpen

### 6.1.1 功能描述

Sharpen 模块用于增强图像的清晰度，包括调节图像边缘的锐化属性和增强图像的细节和纹理，同时还能控制图像水平和垂直方向的边缘锐化的强度。此外，还能控制锐化后的图像的 overshoot 和 undershoot，以及抑制噪声的增强。

### 6.1.2 API 参考

- [HI\\_MPI\\_ISP\\_SetSharpenAttr](#): 设置图像锐化属性。
- [HI\\_MPI\\_ISP\\_GetSharpenAttr](#): 获取图像锐化属性。

#### HI\_MPI\_ISP\_SetSharpenAttr

##### 【描述】

设定图像锐化属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetSharpenAttr(ISP_DEV IspDev, const ISP\_SHARPEN\_ATTR\_S *pstSharpenAttr);
```

##### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstSharpenAttr	图像锐化属性	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	输入参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetSharpenAttr

【描述】

获取图像锐化属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetSharpenAttr(ISP_DEV IspDev, ISP\_SHARPEN\_ATTR\_S  
*pstSharpenAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstSharpenAttr	图像锐化属性	输出





【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	输入参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.1.3 数据类型

- [ISP\\_SHARPEN\\_MANUAL\\_ATTR\\_S](#)：定义 ISP Sharpen 手动属性。
- [ISP\\_SHARPEN\\_AUTO\\_ATTR\\_S](#)：定义 ISP Sharpen 自动属性。
- [ISP\\_SHARPEN\\_ATTR\\_S](#)：定义 ISP Sharpen 属性。

### ISP\_SHARPEN\_MANUAL\_ATTR\_S

【说明】

定义 ISP Sharpen 手动属性。

【定义】

```
typedef struct hiISP_SHARPEN_MANUAL_ATTR_S
{
    HI_BOOL bEnLowLumaShoot;
    HI_U8 u8SharpenD;
```



```
HI_U8 u8SharpenUd;  
HI_U8 u8OverShoot;  
HI_U8 u8UnderShoot;  
HI_U8 u8TextureNoiseThd;  
HI_U8 u8EdgeNoiseThd;  
} ISP_SHARPEN_MANUAL_ATTR_S;
```

#### 【成员】

成员名称	描述
bEnLowLumaShoot	低照度环境下，图像锐化后的 shoot 的严格控制。 取值范围：[0, 1]。正常照度噪声小时，建议设为 0，图像清晰度效果更好，但是图像会有微弱 shoot；低照度噪声大时，建议设为 1，图像清晰度稍差，图像 shoot 会变少。
u8SharpenD	有方向的锐化，设置图像边缘锐度。 取值范围：[0, 0xFF]。
u8SharpenUd	无方向的锐化，设置图像纹理的锐度。 取值范围：[0, 0xFF]。
u8OverShoot	设置图像的 overshoot 的强度 取值范围：[0, 0xFF]。
u8UnderShoot	设置图像的 undershoot 的强度 取值范围：[0, 0xFF]。
u8TextureNoiseThd	图像纹理上噪声的控制阈值。 取值范围：[0, 0xFF]。该值越大，锐化后的图像噪声越小，纹理清晰度越差。
u8EdgeNoiseThd	图像边缘上噪声的控制阈值。 取值范围：[0, 0xFF]。该值越大，锐化后的图像噪声越小，边缘锐度越差。

## ISP\_SHARPEN\_AUTO\_ATTR\_S

#### 【说明】

定义 ISP Sharpen 自动属性。

#### 【定义】

```
typedef struct hiISP_SHARPEN_AUTO_ATTR_S  
{  
    HI_BOOL abEnLowLumaShoot[ISP_AUTO_STENGTH_NUM];  
    HI_U8 au8SharpenD[ISP_AUTO_STENGTH_NUM];  
}
```



```

HI_U8 au8SharpenUd[ISP_AUTO_STENGTH_NUM];
HI_U8 au8OverShoot[ISP_AUTO_STENGTH_NUM];
HI_U8 au8UnderShoot[ISP_AUTO_STENGTH_NUM];
HI_U8 au8TextureNoiseThd[ISP_AUTO_STENGTH_NUM];
HI_U8 au8EdgeNoiseThd[ISP_AUTO_STENGTH_NUM];
} ISP_SHARPEN_AUTO_ATTR_S;

```

## 【成员】

成员名称	描述
abEnLowLumaShoot [ISP_AUTO_STENGTH_NUM]	低照度环境下，图像锐化后的 shoot 的严格控制。正常照度噪声小时，建议设为 0；低照度噪声大时，建议设为 1。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-1 所示。
au8SharpenD [ISP_AUTO_STENGTH_NUM]	有方向的锐化，设置图像边缘和小细节的锐度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-2 所示。
au8SharpenUd [ISP_AUTO_STENGTH_NUM]	无方向的锐化，设置图像纹理的锐度，通常情况建议在相同增益情形下设置的 u8SharpenD 的值小于 u8SharpenUd 的值，该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-3 所示。
au8OverShoot [ISP_AUTO_STENGTH_NUM]	设置图像的 overshoot 的强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-4 所示。
au8UnderShoot [ISP_AUTO_STENGTH_NUM]	设置图像的 undershoot 强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-5 所示。
au8TextureNoiseThd [ISP_AUTO_STENGTH_NUM]	图像纹理上噪声的控制阈值。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-6 所示。
au8EdgeNoiseThd [ISP_AUTO_STENGTH_NUM]	图像边缘上噪声的控制阈值。该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-7 所示。

表6-1 abEnLowLumaShoot [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

abEnLowLumaShoot	Again*Dgain*ISPDgain(times)
abEnLowLumaShoot [0]	1
abEnLowLumaShoot [1]	2



abEnLowLumaShoot	Again*Dgain*ISPDgain(times)
abEnLowLumaShoot [2]	4
abEnLowLumaShoot [3]	8
abEnLowLumaShoot [4]	16
abEnLowLumaShoot [5]	32
abEnLowLumaShoot [6]	64
abEnLowLumaShoot [7]	128
abEnLowLumaShoot [8]	256
abEnLowLumaShoot [9]	512
abEnLowLumaShoot [10]	1024
abEnLowLumaShoot [11]	2048
abEnLowLumaShoot [12]	4096
abEnLowLumaShoot [13]	8192
abEnLowLumaShoot [14]	16384
abEnLowLumaShoot [15]	32768

表6-2 au8SharpenD [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8SharpenD	Again*Dgain*ISPDgain(times)
au8SharpenD [0]	1
au8SharpenD [1]	2
au8SharpenD [2]	4
au8SharpenD [3]	8
au8SharpenD [4]	16
au8SharpenD [5]	32
au8SharpenD [6]	64
au8SharpenD [7]	128
au8SharpenD [8]	256
au8SharpenD [9]	512
au8SharpenD [10]	1024
au8SharpenD [11]	2048
au8SharpenD [12]	4096



au8SharpenD	Again*Dgain*ISPDgain(times)
au8SharpenD [13]	8192
au8SharpenD [14]	16384
au8SharpenD [15]	32768

表6-3 au8SharpenUd[ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8SharpenUd	Again*Dgain*ISPDgain (times)
au8SharpenUd [0]	1
au8SharpenUd [1]	2
au8SharpenUd [2]	4
au8SharpenUd [3]	8
au8SharpenUd [4]	16
au8SharpenUd [5]	32
au8SharpenUd [6]	64
au8SharpenUd [7]	128
au8SharpenUd [8]	256
au8SharpenUd [9]	512
au8SharpenUd [10]	1024
au8SharpenUd [11]	2048
au8SharpenUd [12]	4096
au8SharpenUd [13]	8192
au8SharpenUd [14]	16384
au8SharpenUd [15]	32768

表6-4 au8OverShoot [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8OverShoot	Again*Dgain*ISPDgain(times)
au8OverShoot [0]	1
au8OverShoot [1]	2
au8OverShoot [2]	4
au8OverShoot [3]	8



au8OverShoot	Again*Dgain*ISPDgain(times)
au8OverShoot [4]	16
au8OverShoot [5]	32
au8OverShoot [6]	64
au8OverShoot [7]	128
au8OverShoot [8]	256
au8OverShoot [9]	512
au8OverShoot [10]	1024
au8OverShoot [11]	2048
au8OverShoot [12]	4096
au8OverShoot [13]	8192
au8OverShoot [14]	16384
au8OverShoot [15]	32768

表6-5 au8UnderShoot [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8UnderShoot	Again*Dgain*ISPDgain(times)
au8UnderShoot [0]	1
au8UnderShoot [1]	2
au8UnderShoot [2]	4
au8UnderShoot [3]	8
au8UnderShoot [4]	16
au8UnderShoot [5]	32
au8UnderShoot [6]	64
au8UnderShoot [7]	128
au8UnderShoot [8]	256
au8UnderShoot [9]	512
au8UnderShoot [10]	1024
au8UnderShoot [11]	2048
au8UnderShoot [12]	4096
au8UnderShoot [13]	8192
au8UnderShoot [14]	16384



au8UnderShoot	Again*Dgain*ISPDgain(times)
au8UnderShoot [15]	32768

表6-6 au8TextureNoiseThd [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8TextureNoiseThd	Again*Dgain*ISPDgain(times)
au8TextureNoiseThd [0]	1
au8TextureNoiseThd [1]	2
au8TextureNoiseThd [2]	4
au8TextureNoiseThd [3]	8
au8TextureNoiseThd [4]	16
au8TextureNoiseThd [5]	32
au8TextureNoiseThd [6]	64
au8TextureNoiseThd [7]	128
au8TextureNoiseThd [8]	256
au8TextureNoiseThd [9]	512
au8TextureNoiseThd [10]	1024
au8TextureNoiseThd [11]	2048
au8TextureNoiseThd [12]	4096
au8TextureNoiseThd [13]	8192
au8TextureNoiseThd [14]	16384
au8TextureNoiseThd [15]	32768

表6-7 au8EdgeNoiseThd [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8EdgeNoiseThd	Again*Dgain*ISPDgain(times)
au8EdgeNoiseThd [0]	1
au8EdgeNoiseThd [1]	2
au8EdgeNoiseThd [2]	4
au8EdgeNoiseThd [3]	8
au8EdgeNoiseThd [4]	16
au8EdgeNoiseThd [5]	32



au8EdgeNoiseThd	Again*Dgain*ISPDgain(times)
au8EdgeNoiseThd [6]	64
au8EdgeNoiseThd [7]	128
au8EdgeNoiseThd [8]	256
au8EdgeNoiseThd [9]	512
au8EdgeNoiseThd [10]	1024
au8EdgeNoiseThd [11]	2048
au8EdgeNoiseThd [12]	4096
au8EdgeNoiseThd [13]	8192
au8EdgeNoiseThd [14]	16384
au8EdgeNoiseThd [15]	32768

## ISP\_SHARPEN\_ATTR\_S

### 【说明】

定义 ISP Sharpen 属性。

### 【定义】

```
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_SHARPEN_ATTR_S;
```

### 【成员】

成员名称	描述
bEnable	Sharpen 增强功能使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_TRUE。
enOpType	Sharpen 工作类型。 OP_TYPE_AUTO: 自动; OP_TYPE_MANUAL: 手动。 默认值为 OP_TYPE_AUTO。





成员名称	描述
stManual	Sharpen 手动参数，具体参见 <a href="#">ISP_SHARPEN_MANUAL_ATTR_S</a>
stAuto	Sharpen 自动参数，具体参见 <a href="#">ISP_SHARPEN_AUTO_ATTR_S</a>

#### 【注意事项】

Sharpen 功能分为自动和手动：

- bEnable 为 HI\_TRUE，enOpType 为 OP\_TYPE\_AUTO，使用自动 Sharpen 功能。此时 sharpen 的强度值与系统增益的关系请参见成员变量 abEnLowLumaShoot[ISP\_AUTO\_STENGTH\_NUM]，au8SharpenD[ISP\_AUTO\_STENGTH\_NUM]、au8SharpenUd[ISP\_AUTO\_STENGTH\_NUM]、au8OverShoot[ISP\_AUTO\_STENGTH\_NUM]，au8UnderShoot[ISP\_AUTO\_STENGTH\_NUM]，au8TextureNoiseThd[ISP\_AUTO\_STENGTH\_NUM]和 au8EdgeNoiseThd[ISP\_AUTO\_STENGTH\_NUM]的描述。
- bEnable 设置为 HI\_TRUE，enOpType 设置为 OP\_TYPE\_MANUAL 使用手动 Sharpen 功能。

#### 【相关数据类型及接口】

无

## 6.2 Gamma

### 6.2.1 功能描述

Gamma 模块对图像进行亮度空间非线性转换以适配输出设备。Gamma 模块校正 R、G、B 时调用同一组 Gamma 表，Gamma 表各节点之间的间距相同，节点之间的图像像素值使用线性插值生成。

### 6.2.2 API 参考

- [HI\\_MPI\\_ISP\\_SetGammaAttr](#)：设置 Gamma 属性。
- [HI\\_MPI\\_ISP\\_GetGammaAttr](#)：获取 Gamma 属性。

#### HI\_MPI\_ISP\_SetGammaAttr

##### 【描述】

设定 Gamma 属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetGammaAttr(ISP_DEV IspDev, const ISP_GAMMA_ATTR_S *pstGammaAttr);
```



#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstGammaAttr	Gamma 属性。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

通过该接口设定 gamma 属性时会直接配置逻辑寄存器，然后在下一帧直接生效，如果用户在特殊应用场景需要 gamma 与其它相关算法配合使用时，需要考虑相关算法的属性设置的生效时间，防止二者 gamma 与其它算法生效时间不一致导致闪烁现象。

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetGammaAttr

#### 【描述】

获取 Gamma 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetGammaAttr(ISP_DEV IspDev, ISP\_GAMMA\_ATTR\_S
*pstGammaAttr);
```



【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstGammaAttr	Gamma 属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.2.3 数据类型

- [ISP\\_GAMMA\\_ATTR\\_S](#)：定义 ISP Gamma 校正属性。
- [ISP\\_GAMMA\\_CURVE\\_TYPE\\_E](#)：定义 ISP Gamma 曲线类型。

### ISP\_GAMMA\_ATTR\_S

【说明】

定义 ISP Gamma 校正属性。



### 【定义】

```
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;
```

### 【成员】

成员名称	描述
bEnable	Gamma 校正功能使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_TRUE。
enCurveType	Gamma 曲线选择。 默认值为 ISP_GAMMA_CURVE_DEFAULT。
u16Table[GAMMA_NODE_NUM]	Gamma 表。 取值范围: [0, 0xFFF]

### 【注意事项】

Gamma 校正 R、G、B 调用同一组 Gamma Table。

### 【相关数据类型及接口】

[ISP\\_GAMMA\\_CURVE\\_TYPE\\_E](#)

## ISP\_GAMMA\_CURVE\_TYPE\_E

### 【说明】

定义 ISP Gamma 曲线类型。

### 【定义】

```
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;
```

### 【成员】



成员名称	描述
ISP_GAMMA_CURVE_DEFAULT	默认 Gamma 曲线。
ISP_GAMMA_CURVE_SRGB	sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_USER_DEFINE	用户自定义 Gamma 曲线。

#### 【注意事项】

用户自定义 Gamma 曲线时，必须确保 Gamma 表配置正确。WDR 模式下的 Gamma 曲线与线性模式不一样，可以通过 PQ Tools 工具生成。

#### 【相关数据类型及接口】

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

## 6.3 DRC

### 6.3.1 功能描述

DRC 算法用于对 WDR 合成后的数据进行动态范围压缩（Dynamic Range Compression）。

图像一般需要在显示设备上显示，CRT 显示器的动态范围一般只有  $10^2$ dB，而 WDR 合成后的数据的动态范围可以达到  $10^9$ dB，如果直接在 CRT 显示器上显示，就会由于动态范围不匹配的问题，造成丢失亮度较高或者较低处的细节。

DRC 算法的目的就是要使真实场景的观察者和显示设备的观察者都能获得相同的视觉感受。DRC 会自适应的压缩亮度信号的动态范围，并且会使完全亮或者完全暗的区域的亮度和色差信号非线性且局部的细节化。

- 本模块可以控制图像的整体亮度。通过参数 `u8Asymmetry`，`u8SecondPole` 和 `u8Stretch` 来调整。
- 本模块还能控制暗区的色度以及亮度。通过 `u16DarkGainLmtY` 用来控制暗区亮度信号的增益的限制，用 `u16DarkGainLmtC` 用来控制暗区色度信号增益的限制，可以通过该参数来控制暗区的颜色不出现过饱和。一般情况下不用限制，采用默认值。
- 本模块可以增强亮区的对比度，而对暗区的对比度没有影响。通过参数 `u8LocalMixingBright` 可以加大亮区亮度值大于 `u8LocalMixingThres` 的物体的增益，而通过参数 `u8LocalMixingDark` 可以加大亮区亮度值小于 `u8LocalMixingThres` 的物体的增益，以此来增强对比度。
- 另外，本模块还可以根据亮度调整颜色。通过参数 `au16ColorCorrectionLut[33]` 可以调整不同亮度颜色的饱和度。

### 6.3.2 API 参考

- [HI\\_MPI\\_ISP\\_SetDRCAttr](#)：设置动态范围压缩参数。



- [HI\\_MPI\\_ISP\\_GetDRCAttr](#): 获取动态范围压缩参数。

## HI\_MPI\_ISP\_SetDRCAttr

### 【描述】

设置动态范围压缩参数。

### 【语法】

```
HI_S32 HI_MPI_ISP_SetDRCAttr(ISP_DEV IspDev, const ISP_DRC_ATTR_S  
*pstDRC);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDRC	动态范围压缩参数	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

### 【需求】

- 头文件: hi\_comm\_isp.h、mpi\_isp.h
- 库文件: libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】



## HI\_MPI\_ISP\_GetDRCAttr

### HI\_MPI\_ISP\_GetDRCAttr

#### 【描述】

获取动态范围压缩参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDRCAttr(ISP_DEV IspDev, ISP_DRC_ATTR_S *pstDRC);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDRC	动态范围压缩参数	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】



## HI\_MPI\_ISP\_SetDRCAttr

### 6.3.3 数据类型

- [ISP\\_DRC\\_MANUAL\\_ATTR\\_S](#): 定义 ISP DRC 手动属性
- [ISP\\_DRC\\_AUTO\\_ATTR\\_S](#): 定义 ISP DRC 自动属性
- [ISP\\_DRC\\_ATTR\\_S](#): 定义 ISP DRC 属性

#### ISP\_DRC\_MANUAL\_ATTR\_S

##### 【说明】

定义 ISP DRC 手动属性。

##### 【定义】

```
typedef struct hiISP_DRC_MANUAL_ATTR_S
{
    HI_U8  u8Strength;
    HI_U8  u8LocalMixingBright;
    HI_U8  u8LocalMixingDark;
} ISP_DRC_MANUAL_ATTR_S;
```

##### 【成员】

成员名称	描述
u8Strength	手动模式下 DRC 的强度，值越大，暗处细节看的越多。
u8LocalMixingBright	手动模式下用来控制大于某个阈值的亮区细节的增益。值越大，增益越大，取值范围为[0x0,0x80]。
u8LocalMixingDark	手动模式下用来控制小于某个阈值的暗区细节的增益。值越大，增益越大，取值范围为[0x0,0x60]。

##### 【注意事项】

u8Strength 在线性和 WDR 模式下均可以达到最大值 0xFF。

##### 【相关数据类型及接口】

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

#### ISP\_DRC\_AUTO\_ATTR\_S

##### 【说明】

定义 ISP DRC 自动属性。

##### 【定义】

```
typedef struct hiISP_DRC_AUTO_ATTR_S
```





```
{  
    HI_U8  u8Strength;  
    HI_U8  u8LocalMixingBright;  
    HI_U8  u8LocalMixingDark;  
} ISP_DRC_AUTO_ATTR_S;
```

#### 【成员】

成员名称	描述
u8Strength	自动模式下 DRC 的强度。线性模式及 sensor build in WDR 模式下，DRC 的实际强度与此处设置值及直方图分布相关，该值越大，实际强度越大，取值范围为[0x0, 0xFF]。
u8LocalMixingBright	自动模式下用来控制大于某个阈值的亮区细节的增益，为只读。值越大，增益越大，取值范围为[0x0, 0x80]。
u8LocalMixingDark	自动模式下用来控制小于某个阈值的暗区细节的增益，为只读。值越大，增益越大，取值范围为[0x0, 0x60]。

#### 【注意事项】

u8Strength 在线性模式下最大值是 0x80，在 WDR 模式下最大值是 0xFF。

#### 【相关数据类型及接口】

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

## ISP\_DRC\_ATTR\_S

#### 【说明】

定义 ISP DRC 属性。

#### 【定义】

```
typedef struct hiISP_DRC_ATTR_S  
{  
    HI_BOOL bEnable;  
    HI_U8  u8SpatialVar;  
    HI_U8  u8RangeVar;  
    HI_U8  u8Asymmetry;  
    HI_U8  u8SecondPole;  
    HI_U8  u8Stretch;  
    HI_U8  u8LocalMixingThres;  
    HI_U16 u16DarkGainLmtY;  
    HI_U16 u16DarkGainLmtC;  
    HI_U16 u16BrightGainLmt;  
    HI_U16 aul6ColorCorrectionLut[33];  
    ISP\_OP\_TYPE\_E enOpType;
```



```

    ISP_DRC_MANUAL_ATTR_S stManual;
    ISP_DRC_AUTO_ATTR_S   stAuto;
} ISP_DRC_ATTR_S;

```

## 【成员】

成员名称	描述	取值范围	默认值
bEnable	DRC 动态范围压缩使能	[0, 1] 0: 禁止; 1: 使能。	在线性模式下为 0, WDR 模式下为 1
u8SpatialVar	控制空域滤波次数, 值越大, 图像越平滑。	[0, 0xF]	0xA
u8RangeVar	控制时域滤波次数, 值越小, 图像局部对比度越大, 默认值为 0, 不建议调整。	[0, 0xF]	0x0
u8Asymmetry	用来生成局部 tone mapping 曲线, 值越小, 暗区拉伸越大, 见图 6-1。	[0x1, 0x1E]	0x2
u8SecondPole	用来生成局部 tone mapping 曲线, 值越大, 整体亮度拉伸越大, 见图 6-2。	[0x96, 0xD2]	0xB4
u8Stretch	用来生成局部 tone mapping 曲线, 值越小, 曲线峰值越往左偏移, 暗区拉伸越小, 见图 6-3。	[0x1E, 0x3C]	0x36
u8LocalMixingThres	用来区分亮区和暗区的阈值	[0x2, 0xA]	0x2
u16DarkGainLmtY	用来限制暗区亮度的增益, 值越大, 暗区的亮度越小	[0x0, 0x85]	0x0
u16DarkGainLmtC	用来限制暗区色度的增益, 值越大, 暗区的色度越小, 趋于灰色	[0x0, 0x85]	0x0
u16BrightGainLmt	用来限制亮区的亮度, 值越大, 亮区越亮, 该值只有在比较高的动态范围的场景才有效。	[0x0, 0xA0]	0x0
au16ColorCorrectionLut[COLOR_CORRECTIONLUT_NODE_NUMBER]	用来调整不同亮度的饱和度。该 LUT 的 Index 是指亮度值从低到高, 且 LUT 里的值越大, 表示饱和度的增益越大, 其中 1024 表示 1 倍, 即不改变饱和度, 当饱和度降低过小, 可能会带来高频细节的损失。	[0x0, 0xFFF]	0x400
enOpType	DRC 工作类型。	-	OP_TYPE_A

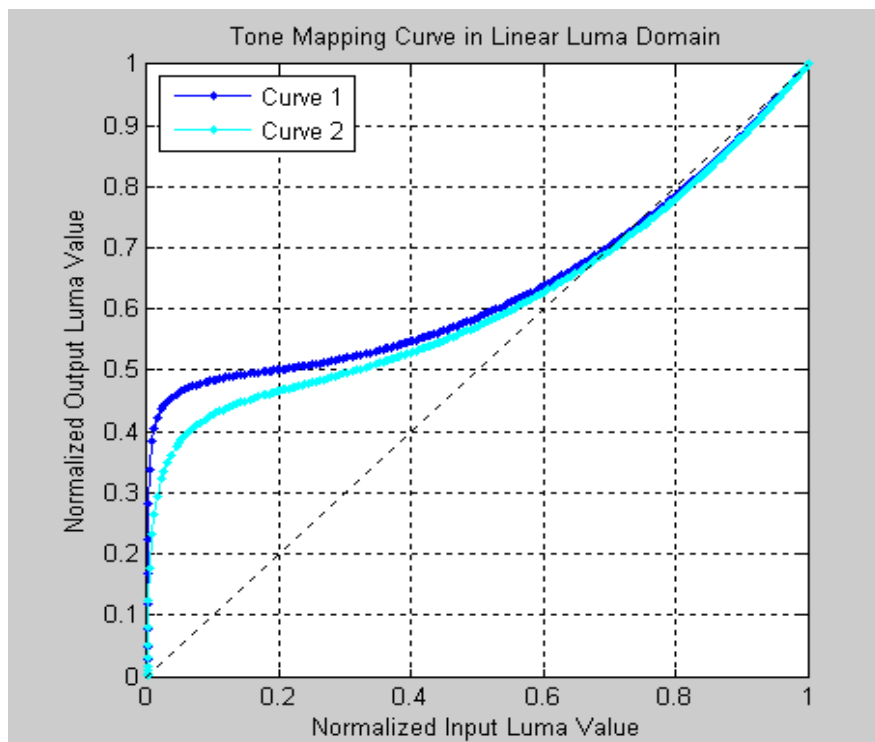


成员名称	描述	取值范围	默认值
	OP_TYPE_AUTO: 自动; OP_TYPE_MANUAL: 手动。		UTO
stManual	DRC 手动参数, 具体参见 <a href="#">ISP_DRC_MANUAL_ATTR_S</a>	-	-
stAuto	DRC 自动参数, 具体参见 <a href="#">ISP_DRC_AUTO_ATTR_S</a>	-	-

#### 【注意事项】

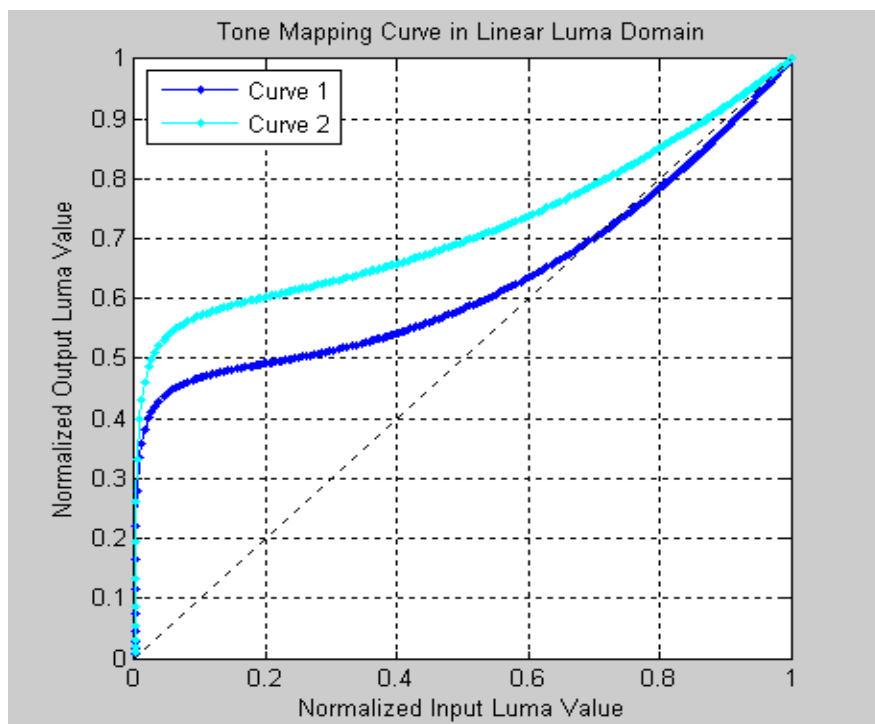
- DRC 属性中除 au16ColorCorrectionLut 为直接配置逻辑寄存器外, 其它成员均为配置外部寄存器来实现的, 用户如果在特殊应用场景同时使用 gamma 与 DRC 时, 建议先调用 [HI\\_MPI\\_ISP\\_GetVDTimeOut](#) 接口, 然后同时设置 gamma 与 DRC 属性来保证 DRC 与 gamma 同时生效防止闪烁, au16ColorCorrectionLut 不建议在使用过程中进行调整。
- DRC 功能分为自动和手动:
  - bEnable 为 HI\_TRUE, enOpType 为 OP\_TYPE\_AUTO, 使用自动 DRC 功能。
  - bEnable 设置为 HI\_TRUE, enOpType 设置为 OP\_TYPE\_MANUAL 使用手动 DRC 功能。
- u8Asymmetry, u8SecondPole 和 u8Stretch 的变化趋势如[图 6-1](#)、[图 6-2](#) 和 [图 6-3](#) 所示:  
设定 Curve1(u8Asymmetry1, u8SecondPole1, u8Stretch1)  
Curve2(u8Asymmetry2, u8SecondPole2, u8Stretch2)  
则 **CASE1**: Curve1 (1,180,53) Curve2 (5,180,53)

图6-1 u8Asymmetry 的变化对曲线的影响趋势



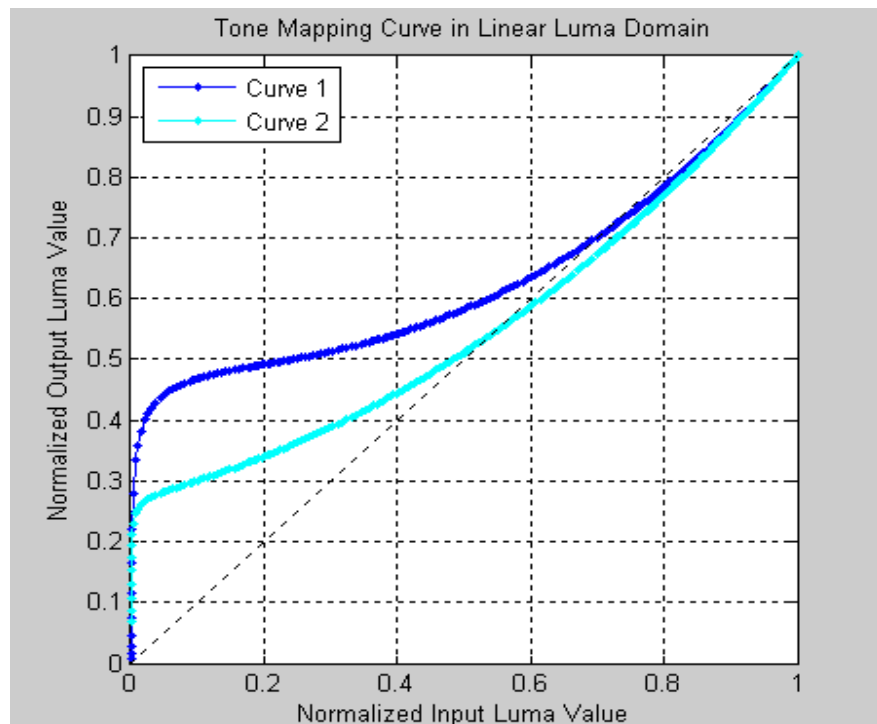
**CASE2:** Curve1 (2,180,53) Curve2 (2,200,53)

图6-2 u8SecondPole 的变化对曲线的影响趋势



**CASE3:** Curve1 (2,180,53) Curve2 (2,180,30)

图6-3 u8Stretch 的变化对曲线的影响趋势



【相关数据类型及接口】

无

## 6.4 镜头阴影校正

### 6.4.1 功能描述

LSC 模块主要用来处理由于镜头光学折射不均匀导致的镜头周围出现阴影的情况。目前流行的处理方式有 Radial（同轴圆）方式以及 Mesh（网格）方式。

在 Hi3518EV200 芯片中 LSC 算法使用 Mesh（网格）方式对画面进行标定/矫正，算法会将整个 Bayer 域画面分割成 16\*16 个子区域，这 16\*16 个区域大小不完全相同，靠近中心的子区域面积比较大，靠近画面边缘的子区域会比较密集。在数据处理过程中，算法将分别对 RAW 域中的每个通道进行处理。

### 6.4.2 API 参考

- [HI\\_MPI\\_ISP\\_SetMeshShadingAttr](#): 设定 LSC 算法参数
- [HI\\_MPI\\_ISP\\_GetMeshShadingAttr](#): 获取 LSC 算法参数。



## HI\_MPI\_ISP\_SetMeshShadingAttr

### 【描述】

设置 LSC 算法参数。

### 【语法】

```
HI_S32 HI_MPI_ISP_SetMeshShadingAttr(ISP_DEV IspDev, const  
ISP_SHADING_ATTR_S *pstShadingAttr);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstShadingAttr	LSC 算法参数	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

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



## HI\_MPI\_ISP\_GetMeshShadingAttr

### 【描述】

获取 LSC 算法参数。

### 【语法】

```
HI_S32 HI_MPI_ISP_GetMeshShadingAttr(ISP_DEV IspDev, ISP\_SHADING\_ATTR\_S
*pstShadingAttr);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstShadingAttr	LSC 算法参数	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

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



## 6.4.3 数据类型

**ISP\_SHADING\_ATTR\_S**: 定义 LSC 算法参数。

### ISP\_SHADING\_ATTR\_S

#### 【说明】

定义 LSC 算法参数。

#### 【定义】

```
typedef struct hiISP_SHADING_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    HI_U32 au32XGridWidth[8];
    HI_U32 au32YGridWidth[8];
    HI_U32 au32NoiseControlGain[SHADING_MESH_NUM_MAX];
    HI_U32 au32RGain[SHADING_MASH_NUM_MAX];
    HI_U32 au32GrGain[SHADING_MASH_NUM_MAX];
    HI_U32 au32GbGain[SHADING_MASH_NUM_MAX];
    HI_U32 au32BGain[SHADING_MASH_NUM_MAX];
} ISP_SHADING_ATTR_S;
```

#### 【成员】

成员名称	描述
bEnable	表示 LSC 使能。
enOpType	OP_TYPE_AUTO 和 OP_TYPE_MANUAL 两种有效选择，用来选定 LSC 模块工作在自动模式或者手动模式。
au32XGridWidth	用来储存各 GRID 分区宽度大小信息。该接口各分量最小值为 1，总和应为原画面宽度的四分之一。（例如原画面大小为 1080p，则该接口各参数总和应为 480，取值范围：[0x0, u32Width/4]，u32Width 为原画面的宽度）
au32YGridWidth	用来储存各 GRID 分区高度大小信息。该接口各分量最小值为 1，总和应为原画面高度的四分之一。（例如原画面大小为 1080p，则该接口各参数总和应为 270，取值范围：[0x0, u32Height/4]，u32Height 为原画面的高度）
au32NoiseControlGain	用来储存 LSC 所用整体增益控制强度数据。总共为 289 个，分别对应画面从左至右、从上至下每一个网格交点处的增益控制强度。该参数仅在 enOpType 设置为 OP_TYPE_AUTO 时生效。取值范围：[0x0, 0x1fff]。
au32RGain	用来储存 LSC 所用 R 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交





成员名称	描述
	点处的 R 分量阴影矫正增益数据值。取值范围：[0x0, 0x1fff]
au32GrGain	用来储存 LSC 所用 Gr 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 Gr 分量阴影矫正增益数据值。取值范围：[0x0, 0x1fff]
au32GbGain	用来储存 LSC 所用 Gb 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 Gb 分量阴影矫正增益数据值。取值范围：[0x0, 0x1fff]
au32BGain	用来储存 LSC 所用 B 通道标定数据。该通道增益共标定 289 个，从 0 到 288 分别表示画面从左至右、从上至下的网格交点处的 B 分量阴影矫正增益数据值。取值范围：[0x0, 0x1fff]

#### 【注意事项】

- 如果需要调整分区宽度或者高度，需要将 enOpType 设置为 OP\_TYPE\_MANUAL 模式，并调整 au32XGridWidth 或 au32YGridWidth 里的相对应的值。注意由于整体画面为上下、左右对称，因此每当设置一个分区的宽度或者高度时，其对称分区的宽度或高度也会相应发生变化。
- 改变后的分区宽度与高度请满足总和为原画面大小宽度或高度的四分之一。
- 如果需求改变某一区域整体增益强度，需要将 enOPType 设置为 OP\_TYPE\_AUTO 模式，并调整 au32NoiseControlGain 中相对应位置的值。该值亦为一个 10bit 精度的整形。默认值 1024 表示增提增益控制强度为 1 倍。
- 如果需要改变某一区域分量增益强度，需要将 enOPType 设置为 OP\_TYPE\_MANUAL 模式，并调整 R、GR、GB、B 四分量的相应区域的值。由于调节比较复杂并容易造成 Color Shading 现象，一般不建议手动调节。

#### 【相关数据类型及接口】

无

## 6.5 Defect Pixel

### 6.5.1 功能描述

DPC 算法通过在 5x5 的窗口中通过坏点检测算法找到该窗口中明显异于临近像素的坏点。该模块主要包含以下两种模式：

- 静态坏点标定/校正有两种流程：亮点和暗点流程
  - 在亮点流程中，光圈处于关闭状态，启动坏点标定程序，得到坏点坐标信息。坏点个数的总数由坏点校正模块的 memory 决定的。得到的坏点通过临近像素的中值滤波进行校正。



- 在暗点流程中，光圈处于正常打开状态，要求平坦背景，最好使用稳定均一的光源，图像整体平均亮度大约为最大亮度 50%，或 Bayer 格式中 B 通道亮度为最大亮度 20% 左右，启动坏点检测程序，得到坏点坐标。坏点个数的总数是由坏点校正模块的 memory 决定的。得到的坏点通过临近像素的中值滤波进行校正。
- 注意，现在的 DPC 模块所能支持的最大静态坏点个数为 2048。
- 动态坏点检测/校正

在这种模式中，校正模块使用动态检测方法的进行坏点检测，并采用中值滤波的方法进行坏点的校正，能够校正的坏点个数是没有限制的。这种模式总体上相对于静态模式更不可靠，但是在低照度情况下强烈推荐使用动态坏点校正功能，可以校正更多的坏点以及明显改善图像偏色的问题。

## 6.5.2 API 参考

- [HI\\_MPI\\_ISP\\_SetDPCalibrate](#): 设置静态坏点标定参数
- [HI\\_MPI\\_ISP\\_GetDPCalibrate](#): 获取静态坏点标定结果
- [HI\\_MPI\\_ISP\\_SetDPStaticAttr](#): 设置静态坏点校正属性
- [HI\\_MPI\\_ISP\\_GetDPStaticAttr](#): 获取静态坏点校正属性
- [HI\\_MPI\\_ISP\\_SetDPDynamicAttr](#): 设置动态坏点校正属性
- [HI\\_MPI\\_ISP\\_GetDPDynamicAttr](#): 获取动态坏点校正属性

### HI\_MPI\_ISP\_SetDPCalibrate

#### 【描述】

设置静态坏点标定参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDPCalibrate(ISP_DEV IspDev, const
ISP_DP_STATIC_CALIBRATE_S *pstDPCalibrate);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPCalibrate	静态坏点标定参数	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。



#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

该接口为静态坏点标定接口，坏点标定只需启动一次，标定完成之后自动关闭标定使能，恢复为正常的坏点校正模式。亮坏点检测的环境为：使用最小的模拟增益和数字增益，降帧率到 5~6fps，使曝光时间为 200ms（系统自动完成），遮黑镜头或关闭光圈（需手动操作）；暗坏点检测的环境为：光圈处于正常打开状态，要求平坦背景，最好使用辉度箱固定光源，图像整体平均亮度大约为最大亮度 50%，或 Bayer 格式中 B 通道亮度为最大亮度 20%左右。

#### 【举例】

以标定亮点为例：

```
ISP_DP_STATIC_CALIBRATE_S stDpcCalib;
stDpcCalib.bEnableDetect= 1; //使能静态坏点标定
stDpcCalib.enStaticDPTYPE = 0;
stDpcCalib.ul6CountMax = 1024;
stDpcCalib.ul6CountMin = 1;
stDpcCalib.u8StartThresh = 1;
stDpcCalib.ul6TimeLimit = 0x640;
HI_MPI_ISP_SetDPCalibrate(IspDev, &stDpcCalib);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetDPCalibrate

#### 【描述】

获取静态坏点标定结果。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDPCalibrate(ISP_DEV IspDev,
ISP_DP_STATIC_CALIBRATE_S *pstDPCalibrate);
```

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPCalibrate	静态坏点标定参数	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

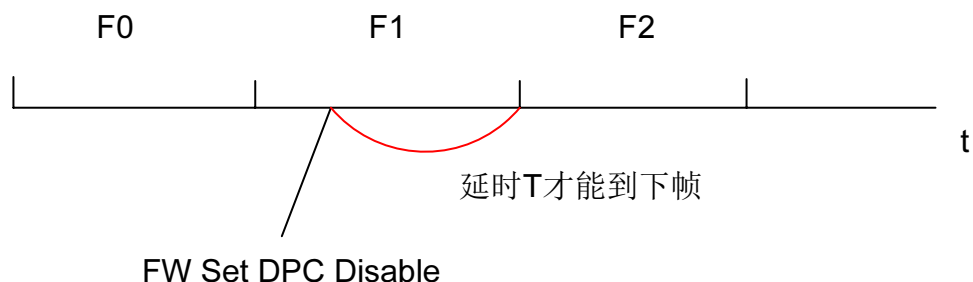
接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 要读取 DPC 的静态标定坐标，需要关闭 DPC 模块。假如 DPC 没有关闭就读标定结果，这样会出现读出来的坐标是相同的。HI\_MPI\_ISP\_GetDPCalibrate 里面有 200ms 的延时，这个是对应 DPC 静态标定时默认 5fps 的配置，刚好一帧是 200ms/1f。也就是 FW 当前帧配置 DPC 关闭，延时 200ms 后,就是下一帧了，再读取的时候就不会有问题（因为 DPC 已经关闭了）。
- 如果帧率改成 3 帧，或者也可以改成更低的帧率，那么一帧需要的时间要比 200ms 大（i.e. 3fps = 333ms/f），那延时 200ms 不能保证已经到下一帧了，至少需要延时 333ms。那帧率更低的话，需要延时的时间更长。示意图如下：





假如 FW 在 F1 帧某个时刻配置 DPC 关闭，然后延时 T(ms)读取坐标，如果 T 够大，那就一定能保证在下帧 F2 读到，T 不够大，那么还是处于 F1，读出来的坐标就会有问題。

因此，客户如果修改了 DPC 标定帧率（如 n fps，n<5），因为 HI\_MPI\_ISP\_GetDPCalibrate 已经有 200ms 的延时，对应的是 5fps 的默认帧率，那么还需要客户自己额外实现延时 T1，满足  $T1+200 \geq 1000/n$ 。

#### 【举例】

以获取亮点标定结果为例：

```
HI_U16 i;
HI_U16 ul6BrightBpNumber= 0;
HI_U32 au32BrightTable[2048]={0};
HI_MPI_ISP_GetDPCalibrate(IspDev,&stDpcCalib);
while(stDpcCalibrate.enStatus == ISP_STATE_INIT)
{
    sleep(1);
    HI_MPI_ISP_GetDPCalibrate(IspDev,&stDpcCalibrate);
}
ul6BrightBpNumber = stDpcCalib.ul6Count;
for (i=0;i< ul6BrightBpNumber;i++)
{
    au32BrightTable[i] = stDpcCalib.au32Table[i];
}
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_SetDPStaticAttr

#### 【描述】

设置静态坏点属性，合并坏点表，将坏点坐标信息与坏点数目写入系统内存。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDPStaticAttr(ISP_DEV IspDev, const
ISP_DP_STATIC_ATTR_S *pstDPStaticAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPStaticAttr	静态坏点属性	输入

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

亮点数目  $u16BrightCount \leq 2048$ ，暗点数目  $u16DarkCount \leq 2048$ ，假定亮点表与暗点表中有  $u16Same$  个重复的点， $u16BrightCount + u16DarkCount - u16Same \leq 2048$

#### 【举例】

```
ISP_DP_STATIC_ATTR_S stDpcStaticAttr;
stDpcStaticAttr.u16BrightCount = u16BrightBpNumber;
stDpcStaticAttr.u16DarkCount = u16DarkBpNumber;
for (i=0;i< u16BrightBpNumber;i++)
{
    stDpcStaticAttr.au32BrightTable[i]= au32BrightTable [i];
}
for (i=0;i< u16DarkBpNumber;i++)
{
    stDpcStaticAttr.au32DarkTable[i]= au32DarkTable [i];
}
HI_MPI_ISP_SetDPAttr(IspDev,&stDpcStaticAttr);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetDPStaticAttr

#### 【描述】

获取静态坏点属性。



#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDPStaticAttr(ISP_DEV IspDev, ISP_DP_STATIC_ATTR_S  
*pstDPStaticAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPStaticAttr	静态坏点属性	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

在设置静态坏点属性中，亮点坏点表中表示亮坏点坐标信息，暗坏点表中则保存的是暗坏点坐标信息；而获取静态坏点属性时，亮点坏点表中保存的是所有坏点的坐标信息。

#### 【举例】

```
HI_U16 u16BpNumber= 0;  
HI_U32 au32BpTable[2048]={0};  
HI_MPI_ISP_GetDPStaticAttr(IspDev, stDpcStaticAttr);  
u16BpNumber = stDpcStaticAttr.u16BrightCount;  
for (i=0;i< u16BpNumber;i++)  
{  
    au32BpTable[i]= stDpcStaticAttr.au32BrightTable[i];  
}
```



}

#### 【相关主题】

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

## HI\_MPI\_ISP\_SetDPDynamicAttr

#### 【描述】

设置动态坏点调试属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDPDynamicAttr(ISP_DEV IspDev, const  
ISP\_DP\_DYNAMIC\_ATTR\_S *pstDPDynamicAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPDynamicAttr	动态坏点属性	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】





无

【相关主题】

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

## HI\_MPI\_ISP\_GetDPDynamicAttr

【描述】

获取动态坏点调试属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDPDynamicAttr(ISP_DEV IspDev, ISP_DP_DYNAMIC_ATTR_S
*pstDPDynamicAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDPDynamicAttr	动态坏点属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无。

【举例】



无。

【相关主题】

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

## 6.5.3 数据类型

- [ISP\\_STATIC\\_DP\\_TYPE\\_E](#): 静态坏点标定类型
- [ISP\\_STATUS\\_E](#): ISP 校正（检测）状态
- [ISP\\_DP\\_STATIC\\_CALIBRATE\\_S](#): 静态坏点标定属性。
- [ISP\\_DP\\_STATIC\\_ATTR\\_S](#): 静态坏点校正属性。
- [ISP\\_DP\\_DYNAMIC\\_ATTR\\_S](#): 定义动态坏点校正属性
- [ISP\\_DP\\_DYNAMIC\\_MANUAL\\_ATTR\\_S](#): 定义动态坏点校正的手动属性
- [ISP\\_DP\\_DYNAMIC\\_AUTO\\_ATTR\\_S](#): 定义动态坏点校正的自动属性。

### ISP\_STATIC\_DP\_TYPE\_E

【说明】

定义静态坏点标定类型。

【定义】

```
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;
```

【成员】

成员名称	描述
ISP_STATIC_DP_BRIGHT	0x0: 选择亮点标定
ISP_STATIC_DP_DARK	0x1: 选择暗点标定

【注意事项】

无

【相关数据类型及接口】

[ISP\\_DP\\_STATIC\\_CALIBRATE\\_S](#)

### ISP\_STATUS\_E

【说明】

定义 ISP 校正（检测）状态。



### 【定义】

```
typedef enum hiISP_STATE_E
{
    ISP_STATE_INIT      = 0,
    ISP_STATE_SUCCESS   = 1,
    ISP_STATE_TIMEOUT   = 2,
    ISP_STATE_BUTT
}ISP_STATUS_E;
```

### 【成员】

成员名称	描述
ISP_STATE_INIT	0: 初始状态，未标定
ISP_STATE_SUCCESS	1: 静态坏点标定成功结束
ISP_STATE_TIMEOUT	2: 静态坏点标定超时结束

### 【注意事项】

无

### 【相关数据类型及接口】

[ISP\\_DP\\_STATIC\\_CALIBRATE\\_S](#)

## ISP\_DP\_STATIC\_CALIBRATE\_S

### 【说明】

定义静态坏点标定的属性。

### 【定义】

```
typedef struct hiISP_DP_STATIC_CALIBRATE_S
{
    HI_BOOL bEnableDetect;
    ISP\_STATIC\_DP\_TYPE\_E enStaticDPType;
    HI_U8   u8StartThresh;
    HI_U16  u16CountMax;
    HI_U16  u16CountMin;
    HI_U16  u16TimeLimit;
    HI_U32  au32Table[STATIC_DP_COUNT_MAX];
    HI_U8   u8FinishThresh;
    HI_U16  u16Count;
    ISP\_STATUS\_E enStatus;
} ISP_DP_STATIC_CALIBRATE_S;
```



## 【成员】

成员名称	描述	取值范围	默认值
bEnableDetect	静态坏点标定使能	[0,1] 0: 禁止; 1: 使能	0
enStaticDPType	静态坏点标定类型	[0,1] 0: 亮点; 1: 暗点	0
u8StartThresh	静态坏点标定开始时的检测门限值 该值的设置与 sensor 相关	[1, 0xFF]	0x3
u16CountMax	允许静态坏点的最大个数	[0,0x800]	0x400
u16CountMin	允许静态坏点的最小个数	[0,u16CountMax]	0x1
u16TimeLimit	允许标定超时门限值	[0, 0x640]	0x640
au32Table[2048]	只读, 亮暗坏点坐标值查找表, 低 25bit 有效, [12:0]bit 为坏点的 水平坐标, [24:13]bit 为坏点的垂 直坐标	每个坐标取值范 围: [0, 0x1FFFFFFF]	*
u8FinishThresh	只读, 静态坏点标定结束时的检 测门限值	[0, 0xFF]	0
u16Count	只读, 标定出的静态坏点的个数	[0,0x800]	0
enStatus	只读, 静态坏点标定结果状态信 息	[0,2]	0

## 【注意事项】

- 坏点检测算法检测成功的标准: 检测出的坏点数量是否在 u16CountMax 与 u16CountMin 之间。所以不同类型的 sensor 在做坏点检测时需微调这两个值。
- u8FinishThresh 只作为输出。针对同类型的 sensor, 参考 u8FinishThresh 值, 设置合理的 u8StartThresh, 能加快静态坏点校正过程。
- 对于 Hi3518EV200 来说, 25 比特的坏点表位宽是有冗余的, 所支持的图像最大为 2048\*1536
- 每次调用标定接口 [HI\\_MPI\\_ISP\\_SetDPCalibrate](#) 时, 算法内部会自动将 enStatus 恢复为 ISP\_STATE\_INIT 状态。

## 【相关数据类型及接口】

- [ISP\\_STATIC\\_DP\\_TYPE\\_E](#)
- [ISP\\_STATUS\\_E](#)



## ISP\_DP\_STATIC\_ATTR\_S

### 【说明】

定义静态坏点校正属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述	取值范围	默认值
bEnable	静态坏点校正使能	[0, 1] 0: 禁止; 1: 使能	1
u16BrightCount	亮坏点个数 <b>注意:</b> 当该变量作为输入时, 表示亮坏点的个数; 作为输出时, 表示总的坏点个数	[0, 0x800]	0
u16DarkCount	暗坏点个数 <b>注意:</b> 当该变量作为输入时, 表示暗坏点的个数; 作为输出时, 无效值 0。	[0, 0x800]	0
au32BrightTable[2048]	亮坏点坐标信息, 低 25bit 有效, [12:0]bit 为坏点水平坐标, [24:13]bit 为坏点垂直坐标。 <b>注意:</b> 当该变量作为输入时, 表示亮坏点的坐标值查找表; 作为输出时, 表示所有坏点的坐标值查找表。	[0, 0x1FFFFFFF]	



成员名称	描述	取值范围	默认值
au32DarkTable[2048]	暗的坏点坐标值，低 25bit 有效，[12:0]bit 为坏点水平坐标，[24:13]bit 为坏点垂直坐标。  <b>注意：当该变量作为输入时，表示暗坏点的坐标值查找表；作为输出时，无效值。</b>	[0, 0x1FFFFFFF]	
bShow	静态坏点显示使能	[0,1] 0: 禁止; 1: 使能	0

**【注意事项】**

bEnableDetect、bShow 与 bEnable 是 3 个互斥的设置，并且对应的优先级是逐次降低的。

**【相关数据类型及接口】**

无

## ISP\_DP\_DYNAMIC\_MANUAL\_ATTR\_S

**【说明】**

定义动态坏点校正的手动属性。

**【定义】**

```
typedef struct hiISP_DP_DYNAMIC_MANUAL_ATTR_S
{
    HI_U16  u16Slope;
    HI_U16  u16BlendRatio;
}ISP_DP_DYNAMIC_MANUAL_ATTR_S;
```

**【成员】**

成员名称	描述	取值范围	默认值
u16Slope	DPC 动态坏点处理强度，取值越大，DPC 处理强度越强，可以校正的动态坏点越多。	[0x0, 0xFF]	0
u16BlendRatio	坏点校正过程所需的融合比率。	[0x0, 0x100]	0

**【注意事项】**



u16Slope 的选择与 iso 值有很大的关系，iso 越高，对应的 u16Slope 应该越大，在图像中坏点校正较好时，图像中偏红严重，此时可以再调节 u16BlendRatio 改善图像质量。

#### 【相关数据类型及接口】

ISP\_DP\_DYNAMIC\_ATTR\_S

### ISP\_DP\_DYNAMIC\_AUTO\_ATTR\_S

#### 【说明】

定义动态坏点校正的自动属性

#### 【定义】

```
typedef struct hiISP_DP_DYNAMIC_AUTO_ATTR_S
{
    HI_U16  au16Slope[16];
    HI_U16  au16BlendRatio[16];
}ISP_DP_DYNAMIC_AUTO_ATTR_S;
```

#### 【成员】

成员名称	描述	取值范围
au16Slope[16]	自动模式下的 DPC 的处理强度，该数组的 16 个值分别对应 sensor 在不同增益下的 DPC 的处理强度值，一般情况下，增益越大，配置的 DPC 处理强度值越大，对应关系如表 6-8 所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个增益值对应的处理强度。	[0, 0xFF]
au16BlendRatio[16]	自动模式下的融合比率。其值越大，图像越偏绿。若连续两档的融合比率不相等，则通过线性插值的方法确定两档中间的某个增益值对应的融合比率。	[0, 0x100]

表6-8 au16Slope[16]在不同增益情况下对应的设置值

au16Slope	Again*Dgain*ISPDgain(times)
au16Slope [0]	1
au16Slope [1]	2
au16Slope [2]	4
au16Slope [3]	8
au16Slope [4]	16
au16Slope [5]	32



au16Slope	Again*Dgain*ISPDgain(times)
au16Slope [6]	64
au16Slope [7]	128
au16Slope [8]	256
au16Slope [9]	512
au16Slope [10]	1024
au16Slope [11]	2048
au16Slope [12]	4096
au16Slope [13]	8192
au16Slope [14]	16384
au16Slope [15]	32768

**【注意事项】**

- 该结构体中的默认值均在 `sensor_cmos.c` 中，如果用户需要修改默认值，请修改相应参数。如果用户需要对接新的 `sensor`，请参考已经提供的其它 `sensor` 的默认值。
- 在 WDR 模式与线性模式之下，`Again*Dgain*ISPDgain` 没有差异。

**【相关数据类型及接口】**[ISP\\_DP\\_DYNAMIC\\_ATTR\\_S](#)

## ISP\_DP\_DYNAMIC\_ATTR\_S

**【说明】**

定义动态坏点校正属性。

**【定义】**

```
typedef struct hiISP_DP_DYNAMIC_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_DP_DYNAMIC_MANUAL_ATTR_S stManual;
    ISP_DP_DYNAMIC_AUTO_ATTR_S stAuto;
}ISP_DP_DYNAMIC_ATTR_S;
```

**【成员】**

成员名称	描述	取值范围	默认值
bEnable	动态坏点校正功能使能	[0, 1]	1





成员名称	描述	取值范围	默认值
enOpType	DPC 动态坏点校正工作模式	[0,1] 0: 自动; 1: 手动	0
stManual	手动模式下配置动态坏点校正参数, 详参 <a href="#">ISP_DP_DYNAMIC_MANUAL_ATTR_S</a>	-	-
stAuto	自动模式下配置动态坏点校正参数, 详参 <a href="#">ISP_DP_DYNAMIC_AUTO_ATTR_S</a>	-	-

【注意事项】

无。

【相关数据类型及接口】

- [ISP\\_OP\\_TYPE\\_E](#)
- [ISP\\_DP\\_DYNAMIC\\_MANUAL\\_ATTR\\_S](#)
- [ISP\\_DP\\_DYNAMIC\\_AUTO\\_ATTR\\_S](#)

## 6.6 Crosstalk Removal

### 6.6.1 概述

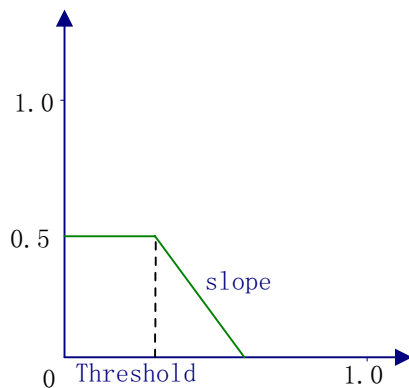
Crosstalk Removal 模块的主要功能是为了平衡 rawdata 之间临近像素 Gr 和 Gb 之间的差异, 能够有效防止 demosaic 插值算法产生的方格或其他类似 pattern。由于 sensor 可能会因为特殊角度的光线入射而产生 Crosstalk, 形成一些 pattern, 根本原因就是因为在临近像素值之间 Gr 和 Gb 值域不一致。

### 6.6.2 功能描述

如图 6-4 所示, 横坐标表示 Gr 与 Gb 之间的差值, 即 $|Gr-Gb|$ , 纵坐标表示处理的强度值, 当 Gr 与 Gb 之间的差值小于 Threshold 值时, 都按照最大的强度值 0.5 进行处理, Gr 与 Gb 之间的差值大于 Threshold 值时, 处理的强度逐渐减弱。Threshold 值越大, 图像整体被处理的强度越大, 当 slope 值越大, 从处理强度为最小值到处理强度为最大值之间的过渡越剧烈。



图6-4 CrossTalk Remove 门限



### 6.6.3 API 参考

- [HI\\_MPI\\_ISP\\_SetCrosstalkAttr](#): 设定 Crosstalk remove 属性
- [HI\\_MPI\\_ISP\\_GetCrosstalkAttr](#): 获取 Crosstalk remove 属性

#### HI\_MPI\_ISP\_SetCrosstalkAttr

##### 【描述】

设定 Crosstalk remove 属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetCrosstalkAttr(ISP_DEV IspDev, const ISP\_CR\_ATTR\_S
*pstCRAAttr);
```

##### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstCRAAttr	Crosstalk 属性	输入

##### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

##### 【错误码】



接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetCrosstalkAttr

【描述】

获取 Crosstalk 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetCrosstalkAttr(ISP_DEV IspDev, ISP\_CR\_ATTR\_S
*pstCRAAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstCRAAttr	Crosstalk 属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】



接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.6.4 数据类型

### ISP\_CR\_ATTR\_S

【说明】

定义 ISP CrossTalk 属性。

【定义】

```
typedef struct hiISP_CR_ATTR_S
{
    HI_BOOL  bEnable;
    HI_U16   au16Strength[ISP_AUTO_STENGTH_NUM];
    HI_U16   u16Threshold;
    HI_U8    u8Slope;
    HI_U8    u8Sensitivity;
    HI_U16   u16SensiThreshold;
}ISP_CR_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能 Crosstalk 功能。
u16Strength[ISP_AUTO_STENGTH_NUM]	设置 Noise Profile 校正强度值。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。取值范围：[0, 0x3FFF]。默认值 4096。



成员名称	描述
u16Threshold	设置 Crosstalk 门限值。值越大，表示整体处理的强度越大。 取值范围：[0, 0x3FFF]。默认值 4096。
u8Slope	设置 Crosstalk 斜率值。值越大，表示整体处理强度随 Gr 与 Gb 之间的差值变化越剧烈。 取值范围：[0, 0xE]。默认值 7。
u8Sensitivity	设置 Crosstalk 敏感度值。取值范围：[0, 0xE]。默认值 7。值越大，表示在边沿上的处理强度随 Gr 与 Gb 之间的差值变化越剧烈。
u16SensiThreshold	设置 Crosstalk 敏感度门限值。值越大，表示在更强的边沿上也会有绿平衡处理。取值范围：[0, 0x3FFF]。默认值 4096。

表6-9 u16Strength [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

u16Strength	Again*Dgain*ISPDgain (times)
u16Strength [0]	100
u16Strength [1]	200
u16Strength [2]	400
u16Strength [3]	800
u16Strength [4]	1600
u16Strength [5]	3200
u16Strength [6]	6400
u16Strength [7]	12800
u16Strength [8]	25600
u16Strength [9]	51200
u16Strength [10]	102400
u16Strength [11]	204800
u16Strength [12]	409600
u16Strength [13]	819200
u16Strength [14]	1638400
u16Strength [15]	3276800



【注意事项】

无

【相关数据类型及接口】

无

## 6.7 去噪算法

### 6.7.1 概述

去噪算法在空域内进行运算，处理 Raw 数据，在去噪的同时保存边缘和纹理。去噪算法针对每款 sensor 的噪点特性进行处理，sensor 的噪点特性通过离线校正工具得到。

### 6.7.2 功能描述

去噪算法有两种模式：自动模式和手动模式。

- 在自动模式中，去噪算法的强度值与系统的增益值成一个非线性的比例关系，去噪算法的强度值会随着环境的变化而自动改变，当系统所处环境比较暗，光线不足时，系统的增益增大，这时去噪算法的强度值增大；反之，当系统所处环境比较亮时，系统的模拟增益和数字增益比较小，去噪算法的强度值减小。去噪算法的强度值与系统的增益值的对应关系请参见数据类型 [ISP\\_NR\\_ATTR\\_S](#) 中的成员变量说明。
- 在手动模式中，去噪算法的真实强度值与目标值一致。

### 6.7.3 API 参考

- [HI\\_MPI\\_ISP\\_SetNRAttr](#)：设定噪点抑制属性
- [HI\\_MPI\\_ISP\\_GetNRAttr](#)：获取噪点抑制属性

#### HI\_MPI\_ISP\_SetNRAttr

【描述】

设定噪点抑制属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetNRAttr(ISP_DEV IspDev, const ISP\_NR\_ATTR\_S
*pstNRAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstNRAttr	噪点抑制属性	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetNRAttr

【描述】

获取噪点抑制属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetNRAttr(ISP_DEV IspDev, ISP\_NR\_ATTR\_S *pstNRAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstNRAttr	噪点抑制属性	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

头文件：hi\_comm\_isp.h、mpi\_isp.h

库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.7.4 数据类型

- [ISP\\_NR\\_MANUAL\\_ATTR\\_S](#)：定义 ISP NR 手动属性。
- [ISP\\_NR\\_AUTO\\_ATTR\\_S](#)：定义 ISP NR 自动属性。
- [ISP\\_NR\\_ATTR\\_S](#)：定义 ISP NR 属性。

### ISP\_NR\_MANUAL\_ATTR\_S

【说明】

定义 ISP 噪点抑制手动属性。

【定义】

```
typedef struct hiISP_NR_MANUAL_ATTR_S
{
    HI_U8 u8Strength;
    HI_U8 u8varStrength;
    HI_U8 u8fixStrength;
```





```
    HI_U8 u8LowFreqSlope;  
    HI_U16 u16Threshold;  
} ISP_NR_MANUAL_ATTR_S;
```

#### 【成员】

成员名称	描述
u8Strength	当前版本未使用，继承自 Hi3516A 版本。
u8varStrength	手动模式下，噪点抑制可变强度。值越大，自适应去噪强度可达到的值越大。 取值范围：[0, 0xFF]。
u8fixStrength	手动模式下，噪点抑制固定强度。值越大，去噪强度的基值越大。 取值范围：[0, 0xFF]。
u8LowFreqSlope	手动模式下，低频噪点抑制强度。值越大，对低频噪点的去噪强度越大。 取值范围：[0, 0x10]。
u16Threshold	手动模式下，边缘去噪门限。值越大，对边缘细节的去噪强度越大。 取值范围：[0, 0xFFFF]。

#### 【注意事项】

设置的 u8varStrength 要大于 u8fixStrength。在 u8fixStrength=0 时，若 u8varStrength 设置为较大值，则在 u16Threshold 较大时，边缘细节处都会有较强的去噪强度。

#### 【相关数据类型及接口】

无。

## ISP\_NR\_AUTO\_ATTR\_S

#### 【说明】

定义 ISP 噪点抑制自动属性。

#### 【定义】

```
typedef struct hiISP_NR_AUTO_ATTR_S  
{  
    HI_U8 au8varStrength[ISP_AUTO_STRENGTH_NUM];  
    HI_U8 au8fixStrength[ISP_AUTO_STRENGTH_NUM];  
    HI_U8 au8LowFreqSlope[ISP_AUTO_STRENGTH_NUM];  
    HI_U16 au16Threshold[ISP_AUTO_STRENGTH_NUM];  
} ISP_NR_AUTO_ATTR_S;
```



## 【成员】

成员名称	描述
au8Strength[ISP_AUTO_STRENGTH_NUM]	当前版本未使用，继承自 Hi3516A 版本。
au8varStrength[ISP_AUTO_STRENGTH_NUM]	自动模式下设置图像可变去噪强度，该数组的 16 个值分别对应 sensor 在不同增益情况下的可变去噪强度值，一般情况下随增益增加，设置的可变去噪强度值先增大后变小。取值范围：[0, 0xFF]。
au8fixStrength[ISP_AUTO_STRENGTH_NUM]	自动模式下设置图像固定去噪强度，该数组的 16 个值分别对应 sensor 在不同增益情况下的固定去噪强度值，一般情况下增益越大，设置的固定去噪强度值也越大。取值范围：[0, 0xFF]。
au8LowFreqSlope[ISP_AUTO_LOWFREQSLOPE_NUM]	自动模式下设置图像低频去噪强度，该数组的 16 个值分别对应 sensor 在不同增益情况下的低频去噪强度值，一般情况下随增益增加，设置的去噪强度值先增大后变小。取值范围：[0, 0x10]。
au16Threshold[ISP_AUTO_THRESHOLD_NUM]	自动模式下设置图像边缘去噪门限，该数组的 16 个值分别对应 sensor 在不同增益情况下的边缘去噪门限值，一般情况下增益越大，设置的边缘去噪门限值越小。取值范围：[0, 0xFFFF]。

表6-10 在不同增益情况下对应的设置

au8varStrength	au8fixStrength	au8LowFreqSlope	au16Threshold	Again*Dgian*ISP Dgain(times)
au8varStrength [0]	au8fixStrength [0]	au8LowFreqSlope [0]	au16Threshold [0]	1
au8varStrength [1]	au8fixStrength [1]	au8LowFreqSlope [1]	au16Threshold [1]	2
au8varStrength [2]	au8fixStrength [2]	au8LowFreqSlope [2]	au16Threshold [2]	4
au8varStrength [3]	au8fixStrength [3]	au8LowFreqSlope [3]	au16Threshold [3]	8
au8varStrength [4]	au8fixStrength [4]	au8LowFreqSlope [4]	au16Threshold [4]	16
au8varStrength [5]	au8fixStrength [5]	au8LowFreqSlope [5]	au16Threshold [5]	32
au8varStrength [6]	au8fixStrength [6]	au8LowFreqSlope [6]	au16Threshold [6]	64
au8varStrength [7]	au8fixStrength [7]	au8LowFreqSlope [7]	au16Threshold	128



au8varStrength	au8fixStrength	au8LowFreqSlope	au16Threshold	Again*Dgian*ISP Dgain(times)
			[7]	
au8varStrength [8]	au8fixStrength [8]	au8LowFreqSlope [8]	au16Threshold [8]	256
au8varStrength [9]	au8fixStrength [9]	au8LowFreqSlope [9]	au16Threshold [9]	512
au8varStrength [10]	au8fixStrength [10]	au8LowFreqSlope [10]	au16Threshold [10]	1024
au8varStrength [11]	au8fixStrength [11]	au8LowFreqSlope [11]	au16Threshold [11]	2048
au8varStrength [12]	au8fixStrength [12]	au8LowFreqSlope [12]	au16Threshold [12]	4096
au8varStrength [13]	au8fixStrength [13]	au8LowFreqSlope [13]	au16Threshold [13]	8192
au8varStrength [14]	au8fixStrength [14]	au8LowFreqSlope [14]	au16Threshold [14]	16384
au8varStrength [15]	au8fixStrength [15]	au8LowFreqSlope [15]	au16Threshold [15]	32768

#### 【注意事项】

同档 ISO 下 au8varStrength [x]> au8fixStrength [x]，其中 x 为 0~15。

#### 【相关数据类型及接口】

无

## ISP\_NR\_ATTR\_S

#### 【说明】

定义 ISP 噪点抑制属性。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
bEnable	使能噪点抑制功能。
enOpType	噪点抑制工作模式。
stManual	手动模式设置图像去噪强度。
stAuto	自动模式下设置图像去噪强度。

【注意事项】

无

【相关数据类型及接口】

无

## 6.8 UVNR

### 6.8.1 功能描述

UVNR 模块用于图像的色噪的去除，包括图像色噪去除强度的粗调和细调，以及极低照度下的图像整体偏色的微调。

### 6.8.2 API 参考

- [HI\\_MPI\\_ISP\\_SetUvnrAttr](#): 设置图像色度降噪属性。
- [HI\\_MPI\\_ISP\\_GetUvnrAttr](#): 获取图像色度降噪属性。

#### HI\_MPI\_ISP\_SetUvnrAttr

【描述】

设定图像色度降噪属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetUvnrAttr(ISP_DEV IspDev, const ISP_UVNR_ATTR_S  
*pstUvnrAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstUvnrAttr	图像色度降噪属性	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	输入参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetUvnrAttr

【描述】

获取图像色度降噪属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetUvnrAttr(ISP_DEV IspDev, ISP\_UVNR\_ATTR\_S  
*pstUvnrAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstUvnrAttr	图像色度降噪属性	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。
<a href="#">HI_ERR_ISP_ILLEGAL_PARAM</a>	输入参数无效。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.8.3 数据类型

- [ISP\\_UVNR\\_MANUAL\\_ATTR\\_S](#)：定义 ISP UVNR 手动属性。
- [ISP\\_UVNR\\_AUTO\\_ATTR\\_S](#)：定义 ISP UVNR 自动属性。
- [ISP\\_UVNR\\_ATTR\\_S](#)：定义 ISP UVNR 属性。

### ISP\_UVNR\_MANUAL\_ATTR\_S

【说明】

定义 ISP UVNR 手动属性。

【定义】

```
typedef struct hiISP_UVNR_MANUAL_ATTR_S
{
    HI_U8 u8ColorCast;
    HI_U8 u8UvnrThreshold;
```



```

        HI_U8 u8UvnrStrength;
    } ISP_UVNR_MANUAL_ATTR_S;

```

**【成员】**

成员名称	描述
<a href="#">u8ColorCast</a>	设置图像在整体偏色情况下的偏色微调强度。正常照度都建议设为 0，极低照度才建议设为非 0。值越大，则对偏色的纠正能力越强，但是，值越大，图像的色彩饱和度下降的越厉害。 取值范围：[0, 3]。
<a href="#">u8UvnrThreshold</a>	设置图像的色度降噪的细调参数。该值越大，对低频色噪的抑制强度越强。但是，该值过大，则可能会造成图像的颜色细节丢失或者颜色浸染。 取值范围：[0, 64]。
<a href="#">u8UvnrStrength</a>	设置图像的色度降噪的粗调参数。正常照度建议设为 0 或 1，低照度建议设的比 1 大。 取值范围：[0, 34]。

## ISP\_UVNR\_AUTO\_ATTR\_S

**【说明】**

定义 ISP UVNR 自动属性。

**【定义】**

```

typedef struct hiISP_UVNR_AUTO_ATTR_S
{
    HI_U8 au8ColorCast[ISP_AUTO_STENGTH_NUM];
    HI_U8 au8UvnrThreshold[ISP_AUTO_STENGTH_NUM];
    HI_U8 au8UvnrStrength[ISP_AUTO_STENGTH_NUM];
} ISP_UVNR_AUTO_ATTR_S;

```

**【成员】**

成员名称	描述
<a href="#">au8ColorCast</a> [ISP_AUTO_STENGTH_NUM]	设置图像在整体偏色情况下的偏色微调强度，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-11 所示。
<a href="#">au8UvnrThreshold</a> [ISP_A UTO_STENGTH_NUM]	设置图像的色度降噪的细调参数，该数组的 16 个值分别对应 sensor 在不同的增益情况下不同的设置值，具体对应关系如表 6-12 所示。
<a href="#">au8UvnrStrength</a> [ISP_AUTO_STENGTH_	设置图像的色度降噪的粗调参数，该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值，具体



成员名称	描述
NUM]	对应关系如表 6-13 所示。

表6-11 au8ColorCast [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8ColorCast	Again*Dgain*ISPDgain(times)
au8ColorCast [0]	1
au8ColorCast [1]	2
au8ColorCast [2]	4
au8ColorCast [3]	8
au8ColorCast [4]	16
au8ColorCast [5]	32
au8ColorCast [6]	64
au8ColorCast [7]	128
au8ColorCast [8]	256
au8ColorCast [9]	512
au8ColorCast [10]	1024
au8ColorCast [11]	2048
au8ColorCast [12]	4096
au8ColorCast [13]	8192
au8ColorCast [14]	16384
au8ColorCast [15]	32768

表6-12 au8UvnrThreshold [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8UvnrThreshold	Again*Dgain*ISPDgain(times)
au8UvnrThreshold [0]	1
au8UvnrThreshold [1]	2
au8UvnrThreshold [2]	4
au8UvnrThreshold [3]	8
au8UvnrThreshold [4]	16
au8UvnrThreshold [5]	32





au8UvnrThreshold	Again*Dgain*ISPDgain(times)
au8UvnrThreshold [6]	64
au8UvnrThreshold [7]	128
au8UvnrThreshold [8]	256
au8UvnrThreshold [9]	512
au8UvnrThreshold [10]	1024
au8UvnrThreshold [11]	2048
au8UvnrThreshold [12]	4096
au8UvnrThreshold [13]	8192
au8UvnrThreshold [14]	16384
au8UvnrThreshold [15]	32768

表6-13 au8UvnrStrength [ISP\_AUTO\_STENGTH\_NUM]在不同的增益情况下的设置值

au8UvnrStrength	Again*Dgain*ISPDgain (times)
au8UvnrStrength [0]	1
au8UvnrStrength [1]	2
au8UvnrStrength [2]	4
au8UvnrStrength [3]	8
au8UvnrStrength [4]	16
au8UvnrStrength [5]	32
au8UvnrStrength [6]	64
au8UvnrStrength [7]	128
au8UvnrStrength [8]	256
au8UvnrStrength [9]	512
au8UvnrStrength [10]	1024
au8UvnrStrength [11]	2048
au8UvnrStrength [12]	4096
au8UvnrStrength [13]	8192
au8UvnrStrength [14]	16384
au8UvnrStrength [15]	32768



## ISP\_UVNR\_ATTR\_S

### 【说明】

定义 ISP UVNR 属性。

### 【定义】

```
typedef struct hiISP_UVNR_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_UVNR_MANUAL_ATTR_S stManual;
    ISP_UVNR_AUTO_ATTR_S stAuto;
} ISP_UVNR_ATTR_S;
```

### 【成员】

成员名称	描述
bEnable	UVNR 色度降噪功能使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_TRUE。
enOpType	UVNR 工作类型。 OP_TYPE_AUTO: 自动; OP_TYPE_MANUAL: 手动。 默认值为 OP_TYPE_AUTO。
stManual	UVNR 手动参数, 具体参见 <a href="#">ISP_UVNR_MANUAL_ATTR_S</a>
stAuto	UVNR 自动参数, 具体参见 <a href="#">ISP_UVNR_AUTO_ATTR_S</a>

### 【注意事项】

UVNR 功能分为自动和手动:

- bEnable 为 HI\_TRUE, enOpType 为 OP\_TYPE\_AUTO, 使用自动 UVNR 功能。此时 UVNR 的色度降噪的强度值与系统增益的关系请参见成员变量 au8ColorCast [ISP\_AUTO\_STENGTH\_NUM]、au8UvnrThreshold [ISP\_AUTO\_STENGTH\_NUM] 和 au8UvnrStrength [ISP\_AUTO\_STENGTH\_NUM] 的描述。
- bEnable 设置为 HI\_TRUE, enOpType 设置为 OP\_TYPE\_MANUAL 使用手动 UVNR 功能。

### 【相关数据类型及接口】

无

## 6.9 DIS

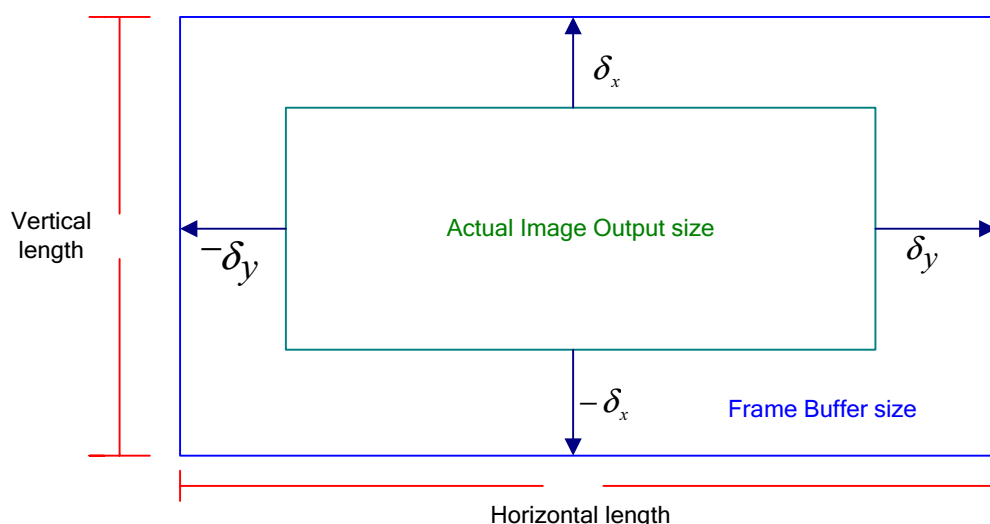
### 6.9.1 概述

DIS 模块通过比较当前图像与前一帧图像来计算出当前图像抖动位移大小，并传递到后级模块，进行输出图像范围的调整，从而起到防抖的效果。每帧图像经过 DIS 后得到一组 CROP 的起点 (X, Y)，再经过后端 VPSS CROP 功能处理，达到防抖效果。

### 6.9.2 功能描述

对于每一帧图像来说，DIS 模块确定了能够使图像保持稳定的水平偏移量  $\delta_x$  和垂直偏移量  $\delta_y$ 。默认情形下，硬件在水平方向和垂直方向输出的像素偏移幅度范围是：[-64, +64]，DIS 偏移示意如图 6-5 所示。

图6-5 DIS 偏移示意图



如图 6-5 所示，VPSS 裁剪后输出的图像大小为内部的长方形，ISP 的 frame buffer 存储的图像大小为外部长方形，当图像抖动后，DIS 模块输出水平方向和垂直方向的偏移量  $\delta_x$ 、 $\delta_y$ ，VPSS 根据这两个偏移量的进行裁剪后输出稳定的图像。

DIS 功能只能在 VI-VPSS 离线模式下使用。

### 6.9.3 API 参考

- [HI\\_MPI\\_ISP\\_SetDISAttr](#): 设置 DIS 属性
- [HI\\_MPI\\_ISP\\_GetDISAttr](#): 获取 DIS 属性

#### HI\_MPI\_ISP\_SetDISAttr

【描述】



设置 DIS 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDISAttr(ISP_DEV IspDev, const ISP_DIS_ATTR_S  
*pstDISAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备。	输入
pstDISAttr	DIS 属性。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

.....

```
/* 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;
stDisAttr.u32ConvergeRate = 16;
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;
}
.....
```

#### 【相关主题】

无

## HI\_MPI\_ISP\_GetDISAttr

#### 【描述】

获取 DIS 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDISAttr(ISP_DEV IspDev, ISP_DIS_ATTR_S *pstDISAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstDISAttr	DIS 属性。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。



【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

无

## 6.9.4 数据类型

[ISP\\_DIS\\_ATTR\\_S](#)：定义 ISP DIS 属性。

### ISP\_DIS\_ATTR\_S

【说明】

定义 ISP DIS 属性。

【定义】

```
typedef struct hiISP_DIS_ATTR_S
{
    HI_BOOL bEnable;
} ISP_DIS_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能 DIS 功能。

【注意事项】

无

【相关数据类型及接口】



无

## 6.10 Defog

### 6.10.1 功能描述

Defog 是通过动态的改变图象的对比度和亮度来实现的，将图像分块，统计每块内的像素值，估算出雾的浓度，根据局部自适应曲线调整去雾强度。Defog 分为手动和自动模式，两种模式下均可调节去雾强度。

### 6.10.2 API 参考

- [HI\\_MPI\\_ISP\\_SetDeFogAttr](#): 设置去雾属性
- [HI\\_MPI\\_ISP\\_GetDeFogAttr](#): 获取去雾属性

#### HI\_MPI\_ISP\_SetDeFogAttr

##### 【描述】

设置 Defog 属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetDeFogAttr(ISP_DEV IspDev, const ISP_DEFOG_ATTR_S  
*pstDefogAttr)
```

##### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstDefogAttr	去雾属性。	输入

##### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

##### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。





【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetDeFogAttr

【描述】

获取 Defog 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDeFogAttr(ISP_DEV IspDev, ISP\_DEFOG\_ATTR\_S  
*pstDefogAttr);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输出
pstDefogAttr	去雾属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误



【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.10.3 数据类型

- [ISP\\_DEFOG\\_MANUAL\\_ATTR\\_S](#)：定义去雾手动模式属性。
- [ISP\\_DEFOG\\_AUTO\\_ATTR\\_S](#)：定义去雾自动模式属性。
- [ISP\\_DEFOG\\_ATTR\\_S](#)：定义 ISP 去雾属性。

### ISP\_DEFOG\_MANUAL\_ATTR\_S

【说明】

定义去雾手动模式属性。

【定义】

```
typedef struct hiISP_DEFOG_MANUAL_ATTR_S
{
    HI_U8      u8strength;
}ISP_DEFOG_MANUAL_ATTR_S;
```

【成员】

成员名称	描述
u8strength	手动模式下去雾强度，取值范围[0,255]

### ISP\_DEFOG\_AUTO\_ATTR\_S

【说明】

定义去雾自动模式属性。

【定义】

```
typedef struct hiISP_DEFOG_AUTO_ATTR_S
{
```



```
        HI_U8          u8strength;  
    }ISP_DEFOG_AUTO_ATTR_S;
```

#### 【成员】

成员名称	描述
u8strength	自动模式下去雾强度的权重系数，每个块自动计算出的去雾强度会乘以权重系数，得到的结果作为最终的去雾强度，取值范围[0,255]

## ISP\_DEFOG\_ATTR\_S

#### 【说明】

定义 ISP 去雾属性。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
bEnable	使能 Defog 功能
u8HorizontalBlock	水平分块数，取值范围[6, 16]，分块越小，越趋向全局去雾，但 CPU 占用率也越少。
u8VerticalBlock	垂直分块数，取值范围[6, 15]，分块越小，越趋向全局去雾，但 CPU 占用率也越少。
enOpType	操作模式
stManual	手动模式
stAuto	自动模式

#### 【注意事项】

无



【相关数据类型及接口】

无

## 6.11 去伪彩

### 6.11.1 功能描述

高频分量在图像插值时易引起高频混叠。用镜头对准一个分辨率测试卡，当 sensor 表面没有 OLPF 时，在分辨率的高频部分容易出现伪彩。去伪彩主要是指去除高频部分因插值错误所导致的伪彩现象。增大去伪彩强度值会减弱伪彩现象，但可能导致正常的颜色灰度化。

### 6.11.2 API 参考

- [HI\\_MPI\\_ISP\\_SetAntiFalseColorAttr](#): 设置去伪彩属性
- [HI\\_MPI\\_ISP\\_GetAntiFalseColorAttr](#): 获取去伪彩属性

#### HI\_MPI\_ISP\_SetAntiFalseColorAttr

【描述】

设置去伪彩属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAntiFalseColorAttr(ISP_DEV IspDev, const  
ISP\_ANTI\_FALSECOLOR\_MANUAL\_S *pstAntiFC);
```

【参数】

参数名称	描述	输入/输出
IspDev	Isp 设备号。	输入
pstAntiFC	去伪彩。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】



接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

无

## HI\_MPI\_ISP\_GetAntiFalseColorAttr

【描述】

获取去伪彩属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAntiFalseColorAttr(ISP_DEV IspDev,  
ISP_ANTI_FALSECOLOR_MANUAL_S *pstAntiFC);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAntiFC	去伪彩。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】



接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

无

### 6.11.3 数据类型

- [ISP\\_ANTI\\_FALSECOLOR\\_MANUAL\\_S](#)：定义 ISP 去伪彩手动属性。
- [ISP\\_ANTI\\_FALSECOLOR\\_AUTO\\_S](#)：定义 ISP NR 去伪彩自动属性。
- [ISP\\_ANTI\\_FALSECOLOR\\_S](#)：定义 ISP 去伪彩属性。

#### ISP\_ANTI\_FALSECOLOR\_MANUAL\_S

【说明】

定义 ISP 去伪彩手动属性。

【定义】

```
typedef struct hiISP_ANTI_FALSECOLOR_MANUAL_S
{
    HI_U8  u8Strength;
    HI_U8  u8Threshold;
} ISP_ANTI_FALSECOLOR_MANUAL_S
```

【成员】

成员名称	描述
<a href="#">u8Strength</a>	去伪彩强度值。值越大，去伪彩强度越强。 取值范围：[0, 0xFF]。
<a href="#">u8Threshold</a>	去伪彩门限值。值越大，去伪彩区域越大。 取值范围：[0, 0xFF]。



【注意事项】

无

【相关数据类型及接口】

无

## ISP\_ANTI\_FALSECOLOR\_AUTO\_S

【说明】

定义 ISP 去伪彩自动属性。

【定义】

```
typedef struct hiISP_ANTI_FALSECOLOR_AUTO_ATTR_S
{
    HI_U8 au8Strength[ISP_AUTO_STENGTH_NUM];
    HI_U8 au8Threshold[ISP_AUTO_STENGTH_NUM];
} ISP_ANTI_FALSECOLOR_AUTO_ATTR_S;
```

【成员】

成员名称	描述
<a href="#">au8Strength[ISP_AUTO_STENGTH_NUM]</a>	自动模式下设置图像去伪彩强度值，该数组的 16 个值分别对应 sensor 在不同增益情况下的去伪彩强度值。一般情况下增益越大，值越小。
<a href="#">au8Threshold[ISP_AUTO_STENGTH_NUM]</a>	自动模式下设置图像去伪彩门限值，该数组的 16 个值分别对应 sensor 在不同增益情况下的去伪彩门限值。由于低照度下 FCR 打开，会导致平坦区域颜色会有变化，一般情况下增益越大，值越小。

au8Strength	au8Threshold	Again*Dgain*ISPDgain(times)
au8Strength [0]	au8Threshold [0]	1
au8Strength [1]	au8Threshold [1]	2
au8Strength [2]	au8Threshold [2]	4
au8Strength [3]	au8Threshold [3]	8
au8Strength [4]	au8Threshold [4]	16
au8Strength [5]	au8Threshold [5]	32



au8Strength	au8Threshold	Again*Dgain*ISPDgain(times)
au8Strength [6]	au8Threshold [6]	64
au8Strength [7]	au8Threshold [7]	128
au8Strength [8]	au8Threshold [8]	256
au8Strength [9]	au8Threshold [9]	512
au8Strength [10]	au8Threshold [10]	1024
au8Strength [11]	au8Threshold [11]	2048
au8Strength [12]	au8Threshold [12]	4096
au8Strength [13]	au8Threshold [13]	8192
au8Strength [14]	au8Threshold [14]	16384
au8Strength [15]	au8Threshold [15]	32768

**【注意事项】**

无

**【相关数据类型及接口】**

无

## ISP\_ANTI\_FALSECOLOR\_S

**【说明】**

定义 ISP 去伪彩属性。

**【定义】**

```
typedef struct hiISP_ANTI_FALSECOLOR_S
{
    HI_BOOL bEnable;
    HI_U8  ISP_ANTI_FALSECOLOR_MANNUAL_S stManual;
    HI_U8  ISP_ANTI_FALSECOLOR_AUTO_S stAuto;
} ISP_ANTI_FALSECOLOR_S;
```

**【成员】**

成员名称: HiISP	描述: HiISP
bEnable	使能去伪彩功能
StManual.u8Strength	手动模式去伪彩强度. 取值范围: [0x0, 0xFF]。
StManual.u8Threshold	手动模式去伪彩门限.





成员名称: HiISP	描述: HiISP
	取值范围: [0x0, 0xFF]。
StAuto.au8Strength[]	自动模式去伪彩强度。 取值范围: [0x0, 0xFF]。
StAuto.au8Threshold[]	自动模式去伪彩门限。 取值范围: [0x0, 0xFF]。

【注意事项】

无

【相关数据类型及接口】

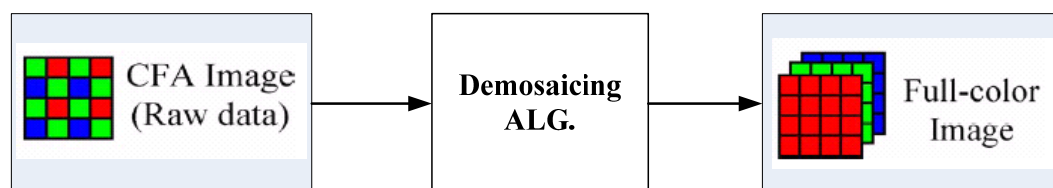
无

## 6.12 去马赛克

### 6.12.1 功能描述

去马赛克主要指将输入的 Bayer 数据转化成 RGB 域数据。在 Bayer 图像阵列的每一像素点只能获得 R, G, B 三基色中的一种彩色分量值。该像素点上其余两种彩色分量值需利用其与周围像素点彩色分量值之间的相关性估计得到。通过对图像阵列中的每一个像素点进行相类似的操作就可以获得采集图像的 R,G,B 三色阵列, 从而再现所采集图像, 这种处理过程被称为 demosaicing。

图6-6 去马赛克处理图



Demosaic 算法可以调节输出图像的分辨率线数和影响边缘锐利度。

### 6.12.2 API 参考

- [HI\\_MPI\\_ISP\\_SetDemosaicAttr](#): 设置去马赛克属性
- [HI\\_MPI\\_ISP\\_GetDemosaicAttr](#): 获取去马赛克属性

#### HI\_MPI\_ISP\_SetDemosaicAttr

【描述】



设置去马赛克属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDemosaicAttr(ISP_DEV IspDev, const  
ISP_DEMOSAIC_ATTR_S *pstDemosaicAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDemosaicAttr	去马赛克属性	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## HI\_MPI\_ISP\_GetDemosaicAttr

#### 【描述】

获取去马赛克属性。



#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDemosaicAttr(ISP_DEV IspDev, ISP_DEMOSAIC_ATTR_S  
*pstDemosaicAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstDemosaicAttr	去马赛克属性	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## 6.12.3 数据类型

- [ISP\\_DEMOSAIC\\_ATTR\\_S](#)：定义 ISP DEMOSAIC 属性。
- [ISP\\_DEMOSAIC\\_CFG\\_E](#)：定义 ISP 去马赛克工作模式。



## ISP\_DEMOSAIC\_ATTR\_S

### 【说明】

定义 ISP 去马赛克属性。

### 【定义】

```
typedef struct hiISP_DEMOSAIC_ATTR_S
{
    HI_U16  u16VhSlope;
    HI_U16  u16AaSlope;
    HI_U16  u16VaSlope;
    HI_U16  u16UuSlope;
    HI_U8   u8VhLimit;
    HI_U8   u8VhOffset;
    HI_U16  u16VhThresh;
    HI_U16  u16AaThresh;
    HI_U16  u16VaThresh;
    HI_U16  u16UuThresh;
    ISP_DEMOSAIC_CFG_E  enCfgType;
    HI_U16  au16LumThresh[ISP_AUTO_ISO_STENGTH_NUM];
    HI_U16  au16NpOffset[ISP_AUTO_ISO_STENGTH_NUM];
}ISP_DEMOSAIC_ATTR_S;
```

### 【成员】

成员名称	描述
u8VhSlope	垂直、水平边缘混合阈值的斜率，调低此参数会提高解析度，加大噪声。 取值范围：[0x0, 0xFF]。一般小于 64。一般默认值 24，实际默认值参考 <code>cmos.c</code> 。
u16AaSlope	不支持
u16VaSlope	不支持
u16UuSlope	全部边缘混合阈值的斜率，调高此参数会提高解析度、锐度，加大噪声。 取值范围：[0x0, 0x3FF]。一般小于 600。一般默认值 400，实际默认值参考 <code>cmos.c</code> 。
u8VhLimit	垂直、水平弱边缘基限值，调高此参数会降低弱边缘锐度，减少噪声。 取值范围：[0x0, 0xFF]。一般大于 16 且小于 64。一般默认值 24，实际默认值参考 <code>cmos.c</code> 。
u8VhOffset;	垂直、水平弱边缘偏差值，调高此参数会降低弱边缘锐度，减少噪声。



成员名称	描述
	取值范围：[0x0, 0xFF]。一般小于 64。一般默认值 16，实际默认值参考 <code>cmos.c</code> 。
<code>u16VhThresh</code>	不支持
<code>u16AaThresh</code>	不支持
<code>u16UuThresh</code>	不支持
<code>u16VaThresh</code>	不支持
<code>enCfgType</code>	不支持
<code>au16LumThresh</code>	不支持
<code>au16NpOffset[ISP_AUTO_STRENGTH_NUM]</code>	<p>该数组用来设置图像的噪声联动参数，取值范围为 [0,0xFF]。</p> <p>随增益的增加，该数组的 16 个值分别对应的 <code>sensor</code> 在不同的增益情况下不同的设置值。一般情况下增益越大，值越小。</p> <p>一般小于 512，正常照度时建议 512，其它照度时一般默认值 0，实际默认值参考 <code>cmos.c</code>。</p>

表6-14 `au16NpOffset[ISP_AUTO_STRENGTH_NUM]`在不同增益情况下对应的设置值

<code>au16NpOffset</code>	<code>Again*Dgain*ISPDgain(times)</code>
<code>au16NpOffset [0]</code>	1
<code>au16NpOffset [1]</code>	2
<code>au16NpOffset [2]</code>	4
<code>au16NpOffset [3]</code>	8
<code>au16NpOffset [4]</code>	16
<code>au16NpOffset [5]</code>	32
<code>au16NpOffset [6]</code>	64
<code>au16NpOffset [7]</code>	128
<code>au16NpOffset [8]</code>	256
<code>au16NpOffset [9]</code>	512
<code>au16NpOffset [10]</code>	1024
<code>au16NpOffset [11]</code>	2048
<code>au16NpOffset [12]</code>	4096
<code>au16NpOffset [13]</code>	8192



au16NpOffset	Again*Dgain*ISPDgain(times)
au16NpOffset [14]	16384
au16NpOffset [15]	32768

#### 【注意事项】

- u16UuSlope 与 au16NpOffset[]的关联性较强，u16UuSlope 与 ISO 无关，u16UuSlope 可以看作是一个基值，而 au16NpOffset[]在各 ISO 档位下的值，可以看作是对 u16UuSlope 基值的修正，修正后的值参见公式“u16UuSlope+au16NpOffset[当前 ISO 档]-512”。
- 随“u16UuSlope+ au16NpOffset[当前 ISO 档]-512”值逐渐减少，将会在颜色边缘(如色卡中颜色块边缘)出现类锯齿现象，反之将改善类锯齿现象。

#### 【相关数据类型及接口】

无

## ISP\_DEMOSAIC\_CFG\_E

#### 【说明】

定义 ISP 去马赛克工作模式。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
ISP_DEMOSAIC_CFG_DEFAULT	正常工作模式
ISP_DEMOSAIC_CFG_VH	VH 调试模式
ISP_DEMOSAIC_CFG_AA	AA 工作模式
ISP_DEMOSAIC_CFG_VA	VA 工作模式
ISP_DEMOSAIC_CFG_UU	UU 工作模式



【注意事项】

无

【相关数据类型及接口】

无

## 6.13 黑电平

### 6.13.1 功能描述

黑电平通常指没有外界光线输入时，sensor 仍会输出的亮度值。ISP 需要减去这个亮度值，以进行颜色的处理。

### 6.13.2 API 参考

- [HI\\_MPI\\_ISP\\_SetBlackLevelAttr](#): 设置黑电平属性
- [HI\\_MPI\\_ISP\\_GetBlackLevelAttr](#): 获取黑电平属性

#### HI\_MPI\_ISP\_SetBlackLevelAttr

【描述】

设置黑电平属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetBlackLevelAttr(ISP_DEV IspDev, const  
ISP\_BLACK\_LEVEL\_S *pstBlackLevel);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstBlackLevel	黑电平属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】



接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

若 cmos.c 里面的 [ISP\\_CMOS\\_BLACK\\_LEVEL\\_S](#) 结构体成员 bUpdate 设置为 HI\_TRUE，则表示始终使用 cmos.c 内的黑电平配置，此 MPI 无效。

【举例】

无

【相关主题】

无

## HI\_MPI\_ISP\_GetBlackLevelAttr

【描述】

获取黑电平属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetBlackLevelAttr(ISP_DEV IspDev, ISP\_BLACK\_LEVEL\_S
*pstBlackLevel);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstBlackLevel	黑电平属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】





接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

无

## 6.13.3 数据类型

### ISP\_BLACK\_LEVEL\_S

【说明】

定义 ISP 黑电平属性。

【定义】

```
typedef struct hiISP_BLACK_LEVEL_S
{
    HI_U16 au16BlackLevel[4];
} ISP_BLACK_LEVEL_S;
```

【成员】

成员名称	描述
au16BlackLevel[4]	黑电平的值，分别表示 R、Gr、Gb、B 分量的黑电平 取值范围：[0x0, 0x7FF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 black_level[4]设定。

【注意事项】

无

【相关数据类型及接口】

无

## 6.14 去 FPN

### 6.14.1 功能描述

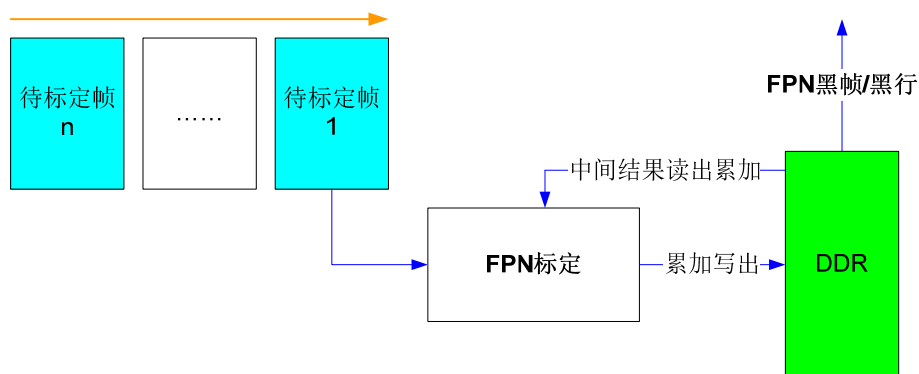
Sensor 将光信号转换成电信号，再通过数百万个 ADC 器件后输出图像。每个像素结构中的光电二极管的尺寸、掺杂浓度、生产过程中的沾污以及 MOS 场效应管的参数的偏差等都会造成像素输出信号的变化，由于这些偏差造成的噪声对于给定的单个像素它是固定的，这种噪声就是固定模式噪声 FPN（Fixed Pattern Noise）。

去 FPN 模块就是要把这些固定模式噪声消除。

FPN 的标定过程：

- 首先关闭镜头的光圈，让 sensor 采集黑帧。
- 启动标定，当第一帧过来时，把满足标定条件的 RAW 数据写到内存，当第二帧过来时，通过读出端口把内存中的黑帧读出来与第二帧进行累加后再通过写出端口再写回内存，后面的帧也是这样，但是最后一帧写出时会对累加的黑帧平均后再写回内存。
- 当求平均写出后，标定结束。此时，内存中就把低照度下的固定列噪声存储下来了，然后，把该黑帧存储到外存上即可。

图6-7 FPN 标定示意图

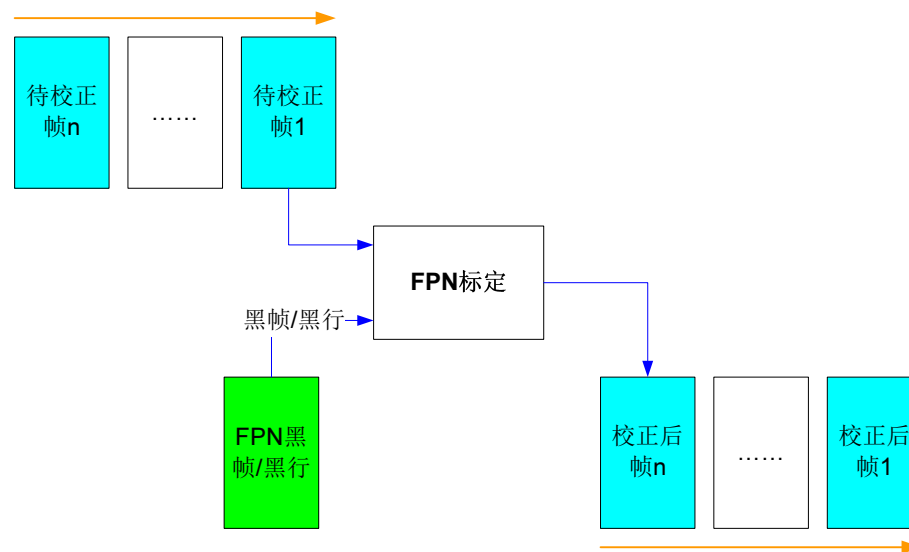


FPN 的校正过程：

- 把标定好的黑帧从外装载入到内存中。
- 当从 sensor 过采集过来的带 FPN 的正常图像过来后，用每帧都减去在内存中的黑帧，从而就得到了消除 FPN 后的校正帧。



图6-8 FPN 校正示意图



## 6.14.2 API 参考

- [HI\\_MPI\\_ISP\\_FPNCalibrate](#): 设置去 FPN 标定属性。
- [HI\\_MPI\\_ISP\\_SetFPNAttr](#): 设置去 FPN 属性。
- [HI\\_MPI\\_ISP\\_GetFPNAttr](#): 获取去 FPN 属性。

### HI\_MPI\_ISP\_FPNCalibrate

#### 【描述】

设置去 FPN 标定属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_FPNCalibrate(ISP_DEV IspDev, ISP\_FPN\_CALIBRATE\_ATTR\_S
*pstCalibrateAttr)
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstCalibrateAttr	去 FPN 标定属性指针。	输入

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 标定时用的内存注意要按 16bit 进行分配，标定完成之后要把黑帧/行，ISO，黑帧的长度、OFFSET、是否压缩标志等保存到外部存储介质，在校正的时候要用到这些信息。
- Hi3518EV200 支持的压缩模式为 COMPRESS\_MODE\_LINE。
- 在行模式时，只支持 16bit 非压缩。
- 在帧模式时，如果是 16bit，只能是非压缩；8bit、10bit、12bit 则可以是压缩，也可以是非压缩。
- 当前端帧率低于 12 帧时，可能会标定失败，请勿在较低帧率下进行 FPN 标定操作。
- 在 BayerRead/BayerDump 场景下，不支持开启 FPN 标定。

#### 【举例】

标定过程请参考 sample。

#### 【相关主题】

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

## HI\_MPI\_ISP\_SetFPNAttr

#### 【描述】

设置去 FPN 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetFPNAttr(ISP_DEV IspDev, const ISP\_FPN\_ATTR\_S
*pstFPNAttr)
```

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstFPNAttr	去 FPN 属性指针。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

- 在 FPN 校正场景下，不支持灌 raw 的操作。
- 根据标定时候保存的黑帧信息，读出 OFFSET、ISO 等信息，同时要把黑帧读入内存，这些是在校正时要输入信息。如果用户选择为自动模式时，会根据校正时的 ISO 来自动调节校正的强度；而手动模式会根据用户输入的校正强度进行校正。

#### 【举例】

```
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);  
.....
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetFPNAttr

#### 【描述】

获取去 FPN 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetFPNAttr(ISP_DEV IspDev, ISP_FPN_ATTR_S *pstFPNAttr)
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstFPNAttr	去 FPN 属性指针。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。



【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.14.3 数据类型

- [ISP\\_FPN\\_FRAME\\_INFO\\_S](#)：定义去 FPN 的标定黑帧信息。
- [ISP\\_FPN\\_CALIBRATE\\_ATTR\\_S](#)：定义去 FPN 的标定属性。
- [ISP\\_FPN\\_ATTR\\_S](#)：定义去 FPN 的校正属性。
- [ISP\\_FPN\\_MANUAL\\_ATTR\\_S](#)：定义去 FPN 在手动模式的校正属性。
- [ISP\\_FPN\\_AUTO\\_ATTR\\_S](#)：定义去 FPN 在手动模式的校正属性。
- [ISP\\_FPN\\_TYPE\\_E](#)：定义去 FPN 类型。

### ISP\_FPN\_FRAME\_INFO\_S

【说明】

定义去 FPN 的标定黑帧信息。

【定义】

```
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;
```

【成员】



成员名称	描述
u32Iso	标定黑帧时的 ISO，来自于标定时保存的 ISO。 ISO 值必须大于 0。
u32Offset	黑帧所有像素的平均值，来自于标定时保存的 OFFSET。 取值范围：[0, 0xFFF]。
u32FrmSize	帧的大小，输出的黑帧有二种：压缩、非压缩，来自于标定时保存的黑帧大小。
stFpnFrame	黑帧帧信息，首先分配内存，然后把标定时的黑帧从外存读入此内存，同时对此结构体成员进行赋值。

【注意事项】

无

【相关数据类型及接口】

[ISP\\_FPN\\_CALIBRATE\\_ATTR\\_S](#)

## ISP\_FPN\_CALIBRATE\_ATTR\_S

【说明】

定义去 FPN 的标定属性。

【定义】

```
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;
```

【成员】

成员名称	描述
u32Threshold	标定时的阈值，在标定时如果像素的值大于该值，则认为是坏点，该像素点不参与标定。取值范围[0x1, 0xFFF]
u32FrameNum	标定的帧数，取值范围为{1, 2, 4, 8, 16}，即为 2 的整数次幂。
enFpnType	标定的类型，有二种：帧模式与行模式，帧模式校正效果要优于行模式，而行模式要比帧模式省内存。
stFpnCaliFrame	标定出来的黑帧信息。





### 【注意事项】

阈值与标定时保存黑帧的比特数相关，假设标定时保存的比特数是 N，则阈值一般取  $(2^N) \gg 2$ 。

### 【相关数据类型及接口】

- [ISP\\_FPN\\_FRAME\\_INFO\\_S](#)
- [ISP\\_FPN\\_ATTR\\_S](#)

## ISP\_FPN\_ATTR\_S

### 【说明】

定义去 FPN 的校正属性。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
bEnable	校正使能。
enOpType	校正模式，分手动与自动模式。
enFpnType	校正类型，分帧模式与行模式。
stFpnFrmInfo	黑帧帧信息。
stManual	手动校正属性。
stAuto	自动校正属性。

### 【注意事项】

校正模式分为手动与自动模式，手动模式需要用户设置校正强度，而自动模式不需要设置校正强度，系统会根据当前图像的 ISO 计算出校正强度，进行动态校正。

### 【相关数据类型及接口】



- [HI\\_MPI\\_ISP\\_SetFPNAttr](#)
- [HI\\_MPI\\_ISP\\_GetFPNAttr](#)
- [ISP\\_FPN\\_MANUAL\\_ATTR\\_S](#)
- [ISP\\_FPN\\_AUTO\\_ATTR\\_S](#)

## ISP\_FPN\_MANUAL\_ATTR\_S

### 【说明】

定义去 FPN 在手动模式的校正属性。

### 【定义】

```
typedef struct hiISP_FPN_MANUAL_ATTR_S
{
    HI_U32          u32Strength;
}ISP_FPN_MANUAL_ATTR_S;
```

### 【成员】

成员名称	描述
u32Strength	手动模式时校正强度。

### 【注意事项】

无

### 【相关数据类型及接口】

无

## ISP\_FPN\_AUTO\_ATTR\_S

### 【说明】

定义去 FPN 在手动模式的校正属性。

### 【定义】

```
typedef struct hiISP_FPN_AUTO_ATTR_S
{
    HI_U32          u32Strength;
}ISP_FPN_AUTO_ATTR_S;
```

### 【成员】

成员名称	描述
u32Strength	自动模式实时校正强度，只读属性（ISP FWM 根据 ISO 自动调整）。



【注意事项】

无

【相关数据类型及接口】

无

## ISP\_FPN\_TYPE\_E

【说明】

定义去 FPN 类型。

【定义】

```
typedef enum hiISP_FPN_TYPE_E
{
    ISP_FPN_TYPE_FRAME = 0,
    ISP_FPN_TYPE_LINE = 1,
    ISP_FPN_TYPE_BUTT
}ISP_FPN_TYPE_E;
```

【成员】

成员名称	描述
ISP_FPN_TYPE_FRAME	帧模式校正。
ISP_FPN_TYPE_LINE	行模式校正。

【注意事项】

无

【相关数据类型及接口】

无

## 6.15 ACM

### 6.15.1 功能描述

ACM(Auto Color Managment) 提供基本的喜好色调节功能，通过对一定区间内的亮度、色调、饱和度的调节，达到对喜好色的调节，如绿色、蓝色、肤色的细化调节。ACM 模块提供了默认的 5 种系数模式。

### 6.15.2 API 参考

- [HI\\_MPI\\_ISP\\_SetAcmAttr](#): 设置 ACM 属性。



- [HI\\_MPI\\_ISP\\_GetAcmAttr](#): 获取 ACM 属性。
- [HI\\_MPI\\_ISP\\_SetAcmCoeff](#): 设置指定模式的 ACM 系数，并让当前使用的 ACM 系数模式立即生效。
- [HI\\_MPI\\_ISP\\_GetAcmCoeff](#): 获取指定模式的 ACM 系数。

## HI\_MPI\_ISP\_SetAcmAttr

### 【描述】

设置 ACM 属性。

### 【语法】

```
HI_S32 HI_MPI_ISP_SetAcmAttr(ISP_DEV IspDev, ISP\_ACM\_ATTR\_S *pstAcmAttr)
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstAcmAttr	ACM 属性	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件: `hi_comm_isp.h`、`mpi_isp.h`
- 库文件: `libisp.a`

### 【注意】

无

### 【举例】

无



【相关主题】

无

## HI\_MPI\_ISP\_GetAcmAttr

【描述】

获取 ACM 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAcmAttr(ISP_DEV IspDev, ISP_ACM_ATTR_S *pstAcmAttr)
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstAcmAttr	ACM 属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】



无

## HI\_MPI\_ISP\_SetAcmCoeff

### 【描述】

设置指定模式的 ACM 系数，并让当前使用的 ACM 系数模式立即生效。

### 【语法】

```
HI_S32 HI_MPI_ISP_SetAcmCoeff(ISP_DEV IspDev, ISP_ACM_LUT_S *pstAcmCoeff,  
ISP_ACM_MODE_E enMode)
```

### 【参数】

参数名称	描述	输入/输出
pstAcmCoeff	ACM 系数。	输入
enMode	ACM 的系数模式。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



## HI\_MPI\_ISP\_GetAcmCoeff

### 【描述】

获取指定模式的 ACM 系数。

### 【语法】

```
HI_S32 HI_MPI_ISP_GetAcmCoeff(ISP_DEV IspDev, ISP_ACM_LUT_S *pstAcmCoef,  
ISP_ACM_MODE_E enMode)
```

### 【参数】

参数名称	描述	输入/输出
pstAcmCoef	ACM 系数。	输出
enMode	ACM 的系数模式。	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



### 6.15.3 数据类型

- [ISP\\_ACM\\_MODE\\_E](#): 定义 ACM 系数模式属性。
- [ISP\\_ACM\\_ATTR\\_S](#): 定义 ACM 属性。
- [ISP\\_ACM\\_LUT\\_S](#): 定义 ACM 系数 LUT。

#### ISP\_ACM\_MODE\_E

##### 【说明】

定义 ACM 系数模式属性。

##### 【定义】

```
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;
```

##### 【成员】

成员名称	描述
ISP_ACM_MODE_BLUE	蓝色模式
ISP_ACM_MODE_GREEN	绿色模式
ISP_ACM_MODE_BG	蓝色和绿色模式
ISP_ACM_MODE_SKIN	肤色模式
ISP_ACM_MODE_VIVID	综合模式

##### 【注意事项】

无

##### 【相关数据类型及接口】

无

#### ISP\_ACM\_ATTR\_S

##### 【说明】

定义 ACM 属性。





### 【定义】

```
typedef struct hiISP_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;
```

### 【成员】

成员名称	描述
bEnable	使能，HI_TRUE:打开 ACM; HI_FALSE:关闭 ACM 功能
bDemoEnable	演示模式使能，左半部分不做，右半部做 ACM
enMode	ACM 系数模式选择
u32Stretch	输入数据范围，0: limit range; 1: full range
u32ClipRange	输出数据范围，0: limit range; 1: full range
u32AcmClipOrWrap	小数部分处理方式，0: 四舍五入; 1: 截位
u32CbcrThr	色度调整的阈值，取值范围[0,255]
u32GainLuma	亮度增益，取值范围[0,512]，精度 1/64
u32GainHue	色调增益，取值范围[0,512]，精度 1/64
u32GainSat	饱和度增益，取值范围[0,512]，精度 1/64

### 【注意事项】

无

### 【相关数据类型及接口】

无

## ISP\_ACM\_LUT\_S

### 【说明】



定义 ACM 系数 LUT。

#### 【定义】

```
typedef struct hiISP_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;
```

#### 【成员】

成员名称	描述
as16Y	亮度查找表，取值范围[-256, 255]
as16H	色调查找表，取值范围[-64, 63]
as16S	饱和度查找表，取值范围[-256, 255]

#### 【注意事项】

无

#### 【相关数据类型及接口】

无

## 6.16 ColorTone

### 6.16.1 功能描述

ColorTone 提供色调调节接口，用户可通过该接口实现图像颜色偏红、偏绿、偏蓝等喜好调节。Hi3518E 暂不支持 ColorTone 功能。

### 6.16.2 API 参考

- [HI\\_MPI\\_ISP\\_SetColorToneAttr](#): 设置色调的参数。
- [HI\\_MPI\\_ISP\\_GetColorToneAttr](#): 获取色调的参数。

#### HI\_MPI\_ISP\_SetColorToneAttr

##### 【描述】

设置色调的参数。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetColorToneAttr(ISP_DEV IspDev, const
```



```
ISP_COLOR_TONE_ATTR_S *pstCTAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	设备编号	输入
pstCTAttr	色调的参数。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

无

#### 【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetColorToneAttr

#### 【描述】

获取色调的参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetColorToneAttr(ISP_DEV IspDev, ISP_COLOR_TONE_ATTR_S  
*pstCTAttr);
```

#### 【参数】



参数名称	描述	输入/输出
IspDev	设备编号	输入
pstCTAttr	色调的参数。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_awb\_comm.h、mpi\_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## 6.16.3 数据类型

[ISP\\_COLOR\\_TONE\\_ATTR\\_S](#)：定义色调调整属性。

### ISP\_COLOR\_TONE\_ATTR\_S

【说明】

定义色调调整属性。

【定义】

```
typedef struct hiISP_COLOR_TONE_ATTR_S
{
```



```
HI_U16 u16RedCastGain;  
HI_U16 u16GreenCastGain;  
HI_U16 u16BlueCastGain;  
} ISP_COLOR_TONE_ATTR_S;
```

#### 【成员】

成员名称	描述
u16RedCastGain	R 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。
u16GreenCastGain	G 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。
u16BlueCastGain	B 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。

#### 【注意事项】

ColorTone 支持客户调节固定的颜色风格，与色温无关。

#### 【相关数据类型及接口】

无

## 6.17 GAMMAFE

### 6.17.1 功能描述

GammaFE 模块将 20bit 的数据压缩到 16bit。Sensor built-in WDR 模式输出的数据经过 ISP 模块时，ISP 相关的模块将数据解压成 20bit 数据，经过相关模块的处理之后再经过 GammaFE 模块。

### 6.17.2 API 参考

- [HI\\_MPI\\_ISP\\_SetGammaFEAttr](#): 设置 GammaFE 属性。
- [HI\\_MPI\\_ISP\\_GetGammaFEAttr](#): 获取 GammaFE 属性。

#### HI\_MPI\_ISP\_SetGammaFEAttr

##### 【描述】

设置 GammaFE 属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_SetGammaFEAttr(ISP_DEV IspDev, const ISP_GAMMAFE_ATTR_S
```



```
*pstGammaFEAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstGammaFEAttr	GammaFE 属性。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## HI\_MPI\_ISP\_GetGammaFEAttr

#### 【描述】

获取 GammaFE 属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetGammaFEAttr(ISP_DEV IspDev, ISP\_GAMMAFE\_ATTR\_S  
*pstGammaFEAttr);
```



#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstGammaFEAttr	GammaFE 属性。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## 6.17.3 数据类型

[ISP\\_GAMMAFE\\_ATTR\\_S](#)：定义 ISP GammaFE 属性。

### ISP\_GAMMAFE\_ATTR\_S

#### 【说明】

定义 ISP GammaFE 属性。

#### 【定义】



```
typedef struct hiISP_GAMMAFE_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16 u16Table[GAMMA_FE0_NODE_NUM + GAMMA_FE1_NODE_NUM];
} ISP_GAMMAFE_ATTR_S;
```

#### 【成员】

成员名称	描述
bEnable	HI_TRUE: 使能 GammaFE 表; HI_FALSE: 关闭 GammaFE 表。
u16Table	GammaFE 查找表。

#### 【注意事项】

无

#### 【相关数据类型及接口】

无

## 6.18 获取 ISP 模块虚拟地址

### 6.18.1 功能描述

从地址空间来分，当前 ISP 模块可分为五个部分：AE 库、AWB 库、AF 库、ISP 物理寄存器模块、ISP 其他模块。

### 6.18.2 API 参考

[HI\\_MPI\\_ISP\\_GetISPRegAttr](#): 获取 ISP 基地址属性。

#### HI\_MPI\_ISP\_GetISPRegAttr

##### 【描述】

获取 ISP 基地址属性。

##### 【语法】

```
HI_S32 HI_MPI_ISP_GetISPRegAttr(ISP_DEV IspDev, ISP\_REG\_ATTR\_S *  
pstIspRegAttr);
```

##### 【参数】





参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspRegAttr	ISP 模块虚拟地址属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

当前 ISP 模块虚拟地址不提供 AF 模块的虚拟地址。

【举例】

无

【相关主题】

无

## 6.18.3 数据类型

[ISP\\_REG\\_ATTR\\_S](#)：定义 ISP 各子模块寄存器的虚拟地址属性。

### ISP\_REG\_ATTR\_S

【说明】

定义 ISP 各子模块寄存器的虚拟地址属性。

【定义】

```
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;
```

#### 【成员】

成员名称	描述
u32IspRegAddr	ISP 内部（物理）寄存器对应的起始虚拟地址
u32IspRegSize	ISP 内部（物理）寄存器对应的大小
u32IspExtRegAddr	ISP 外部（虚拟）寄存器模块对应的起始虚拟地址
u32IspExtRegSize	ISP 外部（虚拟）寄存器模块对应的大小
u32AeExtRegAddr	ISP AE 库对应的起始虚拟地址
u32AeExtRegSize	ISP AE 库对应的大小
u32AwbExtRegAddr	ISP AWB 库对应的起始虚拟地址
u32AwbExtRegSize	ISP AWB 库对应的大小

#### 【注意事项】

无

#### 【相关数据类型及接口】

无

## 6.19 查询内部状态信息

### 6.19.1 功能描述

Inner State Information 接口的作用是输出数个与 ISO 相关的参数其逻辑寄存器当前配置的真实值。

用户在调试过程中可以通过 MPI 接口读取这些真实值参数，观察参数强度是否正确配置。通过该接口用户只可以获取参数，而非操作改变该寄存器里的值。



## 6.19.2 API 参考

[HI\\_MPI\\_ISP\\_QueryInnerStateInfo](#): 获取内部寄存器实际强度配置信息。

### HI\_MPI\_ISP\_QueryInnerStateInfo

#### 【描述】

获取内部寄存器实际强度配置。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstInnerStateInfo	内部寄存器实际配置信息	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】



无

【相关主题】

无

## 6.19.3 数据类型

**ISP\_INNER\_STATE\_INFO\_S**: 定义内部寄存器实际配置信息参数。

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

【说明】

定义内部寄存器实际配置信息参数。

【定义】

```
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 u32RgbirGain;
    HI_BOOL bWDRSwitchFinish;
    HI_BOOL bResSwitchFinish;
} ISP_INNER_STATE_INFO_S;
```

【成员】

成员名称	描述
u32DRCStrengthActual	表示 DRC 模块自动模式时的配置强度
u32NRStrengthActual	表示 NR 模块实际配置强度（Hi3518EV200 不支持）
u32SharpenStrengthDActual	表示 Sharpen 模块 SharpenD 参数实际配置强度
u32SharpenStrengthUdActual	表示 Sharpen 模块 SharpenUD 参数实际配置强度
u32SharpenStrengthRGBActual	表示 Sharpen 模块 SharpenRGB 参数实际配置强度（Hi3518EV200 不支持）
u32DefogStrengthActual	表示 Defog 模块实际配置强度
u32RgbirGain	表示 RGBIR 模块 Gain 值实际配置强度
bWDRSwitchFinish	表示 WDR 切换标志位（Hi3518EV200 不支持）



bResSwitchFinishes	表示 Res 切换标志位（Hi3518EV200 不支持）
--------------------	-------------------------------

**【注意事项】**

u32DRCStrengthActual 仅表示自动模式下 DRC 的强度值，手动模式下该值无意义。

**【相关数据类型及接口】**

无。



# 7 RGBIR

## 7.1 功能描述

针对 RGBIR sensor，该模块主要完成两个操作：

- 去除红外，恢复颜色

RGBIR sensor 由于有一个红外分量，通常使用双通滤光片，此时仍然有红外光照射在 sensor 表面，因此要先通过算法将 R、G、B 通道上的红外光去除掉，恢复出各通道正确的颜色，起到传统红外截止滤光片的作用。

- G 通道插值

同样由于有一个通道被 IR 分量占据，而传统的 RGB sensor 的这个被占据的通道是 G 分量，因此该模块在去除掉红外分量后，又将被 IR 分量占据的 G 分量通过邻域里像素值的大小插值出来，使得 RGBIR 之后的模块可以像处理传统 RGB sensor 一样处理 RGBIR sensor。

## 7.2 MPI 参考

- [HI\\_MPI\\_ISP\\_SetRgbirAttr](#): 设置 RGBIR 属性参数
- [HI\\_MPI\\_ISP\\_GetRgbirAttr](#): 获取 RGBIR 属性参数
- [HI\\_MPI\\_ISP\\_SetRgbirCtrl](#): 设置 RGBIR 控制参数
- [HI\\_MPI\\_ISP\\_GetRgbirCtrl](#): 获取 RGBIR 控制参数

### HI\_MPI\_ISP\_SetRgbirAttr

#### 【描述】

设置 RGBIR 属性参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetRgbirAttr(ISP_DEV IspDev, ISP\_RGBIR\_ATTR\_S  
*pstRgbirAttr);
```

#### 【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstRgbirAttr	RGBIR 属性参数	输入

**【返回值】**

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

**【错误码】**

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

**【需求】**

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

**【注意】**

无

**【举例】**

```
ISP_DEV IspDev = 0;
ISP_RGBIR_ATTR_S stRgbirAttr;
stRgbirAttr.bEnable = HI_TRUE;
stRgbirAttr.enIrPosType = ISP_IRPOS_TYPE_GR;
stRgbirAttr.ul6OverExpThresh = 0xffff;
HI_MPI_ISP_SetRgbirAttr(IspDev, &stRgbirAttr);
```

**【相关主题】**

无

**HI\_MPI\_ISP\_GetRgbirAttr****【描述】**

获取 RGBIR 属性参数。



#### 【语法】

```
HI_S32 HI_MPI_ISP_GetRgbirAttr(ISP_DEV IspDev, ISP_RGBIR_ATTR_S  
*pstRgbirAttr);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstRgbirAttr	RGBIR 属性参数	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## HI\_MPI\_ISP\_SetRgbirCtrl

#### 【描述】

设置 RGBIR 控制参数。





### 【语法】

```
HI_S32 HI_MPI_ISP_SetRgbirCtrl(ISP_DEV IspDev, ISP_RGBIR_CTRL_S  
*pstRgbirCtrl);
```

### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstRgbirCtrl	RGBIR 控制参数	输入

### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

### 【注意】

无

### 【举例】

```
ISP_DEV IspDev = 0;  
ISP_RGBIR_CTRL_S stRgbirCtrl;  
HI_U8 i;  
stRgbirCtrl.bIrFilterEn = HI_TRUE;  
stRgbirCtrl.bIrOutEn = HI_FALSE;  
stRgbirCtrl.bRemovelEn = HI_TRUE;  
stRgbirCtrl.enCompType = OP_TYPE_AUTO;  
stRgbirCtrl.ul6ManuGain = 0x100;  
for(i=0;i<15;i++)
```



```
{  
    stRgbirCtrl.as16ScaleCoef[i] = 0x100;  
}  
HI_MPI_ISP_SetRgbirCtrl(IspDev, &stRgbirCtrl);
```

#### 【相关主题】

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

## HI\_MPI\_ISP\_GetRgbirCtrl

#### 【描述】

获取 RGBIR 控制参数。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetRgbirCtrl(ISP_DEV IspDev, ISP\_RGBIR\_CTRL\_S  
*pstRgbirCtrl);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号	输入
pstRgbirCtrl	RGBIR 控制参数	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。
HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a



【注意】

无

【举例】

无

【相关主题】

无

## 7.3 数据类型

- [ISP\\_IRPOS\\_TYPE\\_E](#): IR 分量位置类型
- [ISP\\_RGBIR\\_ATTR\\_S](#): RGBIR 属性参数
- [ISP\\_RGBIR\\_CTRL\\_S](#): RGBIR 控制参数

### ISP\_IRPOS\_TYPE\_E

【说明】

定义 sensor 中 IR 分量的类型。

【定义】

```
typedef enum hiISP_IRPOS_TYPE_E
{
    ISP_IRPOS_TYPE_GR = 0x0,
    ISP_IRPOS_TYPE_GB,
    ISP_IRPOS_TYPE_BUTT
}ISP_IRPOS_TYPE_E;
```

【成员】

成员名称	描述
ISP_IRPOS_TYPE_GR	0x0: IR 分量在 RGB sensor 的 Gr 位置
ISP_IRPOS_TYPE_GB	0x1: IR 分量在 RGB sensor 的 Gb 位置
ISP_IRPOS_TYPE_BUTT	无

【注意事项】

如果此参数设置不正确，图像的颜色将不正确。

【相关数据类型及接口】

无



## ISP\_RGBIR\_ATTR\_S

### 【说明】

定义 RGBIR 模块的属性参数。

### 【定义】

```
typedef struct hiISP_RGBIR_ATTR_S
{
    HI_BOOL bEnable;
    ISP_IRPOS_TYPE_E enIrPosType;
    HI_U16 u16OverExpThresh;
}ISP_RGBIR_ATTR_S;
```

### 【成员】

成员名称	描述	取值范围	默认值
bEnable	RGBIR 模块使能	[0, 1] 0: 禁止; 1: 使能	0
enIrPosType	Sensor IR 分量位置	[ISP_IRPOS_TYPE_GR, ISP_IRPOS_TYPE_GB]	ISP_IRPOS_TYPE_GR
u16OverExpThresh	过曝值门限	[0, 0xFFFF]	4028

### 【注意事项】

- u16OverExpThresh 为判断像素点是否过曝的门限值，算法对于大于等于该值的像素点不做去除红外处理。如果设置过大，将会出现过曝区域偏紫或偏绿现象，如果设置的值过小，就会很容易出现过曝区。
- u16OverExpThresh 建议设置为 sensor 最大输出值减黑电平，比如 12bit sensor，最大值为 4095，那么此时 u16OverExpThresh 设置为 4095 减黑电平，10bit sensor，左移 2bit 之后最大值为 4092，那么此时 u16OverExpThresh 设置为 4092 减黑电平（注意此时的黑电平是 10bit 时黑电平左移 2 bit 后的值）。
- 另外，如果 u16OverExpThresh 设置的值太小，由于大于此值的像素不进行去 ir 操作，因此也会看到过曝区域偏色现象。

### 【相关数据类型及接口】

无

## ISP\_RGBIR\_CTRL\_S

### 【说明】

定义 RGBIR 模块控制参数。

### 【定义】

```
typedef struct hiISP_RGBIR_CTRL_S
```



```

{
    HI_BOOL bIrOutEn;
    HI_BOOL bIrFilterEn;
    HI_BOOL bRemovelEn;
    ISP_OP_TYPE_E enCompType;
    HI_U16 u16ManuGain;
    HI_S16 as16ScaleCoef[15];
}ISP_RGBIR_CTRL_S;

```

**【成员】**

成员名称	描述	取值范围	默认值
bIrOutEn	IR 图像输出使能	[0,1] 0: 禁止; 1: 使能	0
bIrFilterEn	IR 通道 G 插值滤波使能	[0,1] 0: 禁止; 1: 使能	1
bRemovelEn	去除红外使能	[0,1] 0: 禁止; 1: 使能	1
enCompType	增益补偿类型	[OP_TYPE_AUTO, OP_TYPE_MANUAL]	OP_TYPE_AUTO
u16ManuGain	手动增益补偿增益大小	[0x100, 0x3ff] 其中 8bit 精度	0x100
as16ScaleCoef[15]	颜色矫正系数 (矫正获取)	[-512, 511] 有符号数, 8bit 精度	

**【注意事项】**

- bRemovelEn 在非红外补光时应设置为 1，RGBIR 模块去除红外，恢复正确颜色，当黑暗环境，需要红外补光，而且图像转为黑白时，bRemovelEn 设置为 0，此时 RGBIR 保留红外，增加图像亮度。
- 如果 enCompType 设置为手动，则使用 u16ManuGain 进行补偿，u16ManuGain 越大，噪声越大，但是越不容易出现局部过曝现象。如果 enCompType 设置为自动，则算法根据红外强度自动计算补偿增益大小。使用自动补偿能够平衡噪声和局部过曝现象，建议设置为自动。
- as16ScaleCoef 为有符号数，数据存放方式是以补码方式存放，不同于 CCM。请使用矫正工具矫正出的参数。

**【相关数据类型及接口】**

无



# 8 统计信息

## 8.1 概述

ISP 提供的 3A 统计信息及相关配置。

## 8.2 API 参考

统计信息的 API 接口必须在调用 [HI\\_MPI\\_ISP\\_Init](#) 接口之后才能调用。

- [HI\\_MPI\\_ISP\\_SetStatisticsConfig](#): 设置 ISP 统计信息配置。
- [HI\\_MPI\\_ISP\\_GetStatisticsConfig](#): 获取 ISP 统计信息配置。
- [HI\\_MPI\\_ISP\\_GetStatistics](#): 获取 ISP 统计信息。

### HI\_MPI\_ISP\_SetStatisticsConfig

#### 【描述】

设置 ISP 统计信息配置。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetStatisticsConfig(ISP_DEV IspDev, const  
ISP_STATISTICS_CFG_S *pstStatCfg);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstStatCfg	ISP 统计信息配置。	输入

#### 【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

用户可以关闭不需要的统计信息，以提高系统性能。

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetStatisticsConfig

【描述】

获取 ISP 统计信息配置。

【语法】

```
HI_S32 HI_MPI_ISP_GetStatisticsConfig(ISP_DEV IspDev,  
ISP\_STATISTICS\_CFG\_S *pstStatCfg);
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstStatCfg	ISP 统计信息配置。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

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

## HI\_MPI\_ISP\_GetStatistics

【描述】

获取 ISP 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetStatistics(ISP_DEV IspDev, ISP\_STATISTICS\_S *pstStat)
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstStat	ISP 统计信息。	输出

【返回值】





返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

无

## 8.3 数据类型

- [ISP\\_STATISTICS\\_CFG\\_S](#)：定义 ISP 统计信息配置。
- [ISP\\_STATISTICS\\_CTRL\\_U](#)：定义 ISP 统计信息使能。
- [ISP\\_AE\\_STATISTICS\\_CFG\\_S](#)：定义 AE 统计信息配置。
- [ISP\\_WB\\_STATISTICS\\_CFG\\_S](#)：定义 AWB 统计信息配置。
- [ISP\\_WB\\_STATISTICS\\_CFG\\_PARAM\\_S](#)：定义 AWB 统计信息配置参数。
- [ISP\\_FOCUS\\_STATISTICS\\_CFG\\_S](#)：定义 AF 统计信息配置。
- [ISP\\_AF\\_CFG\\_S](#)：定义 AF 统计信息配置参数。
- [ISP\\_AF\\_PEAK\\_MODE\\_E](#)：统计模式。
- [ISP\\_AF\\_SQU\\_MODE\\_E](#)：平方模式。
- [ISP\\_AF\\_STATISTICS\\_POS\\_E](#)：AF 统计信息输出模式。
- [ISP\\_AF\\_H\\_PARAM\\_S](#)：AF 统计水平滤波器 IIR 参数设置。
- [ISP\\_AF\\_V\\_PARAM\\_S](#)：AF 统计垂直滤波器 IIR 参数设置。
- [ISP\\_AF\\_FV\\_PARAM\\_S](#)：AF 统计结果输出格式参数。



- [ISP\\_STATISTICS\\_S](#): 定义统计信息。
- [ISP\\_AE\\_STATISTICS\\_S](#): 定义 AE 统计信息。
- [ISP\\_WB\\_STATISTICS\\_S](#): 定义 AWB 统计信息。
- [ISP\\_FOCUS\\_STATISTICS\\_S](#): 定义 AF 统计信息。
- [ISP\\_WB\\_BAYER\\_STATISTICS\\_INFO\\_S](#): 定义 Bayer 域 AWB 统计信息。
- [ISP\\_WB\\_RGB\\_STATISTICS\\_INFO\\_S](#): 定义 RGB 域 AWB 统计信息。
- [ISP\\_FOCUS\\_ZONE\\_S](#): 定义 AF 统计信息参数。

## ISP\_STATISTICS\_CFG\_S

### 【说明】

定义 ISP 统计信息配置。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
unKey	统计信息使能。
stAECfg	AE 统计信息配置。
stWBCfg	AWB 统计信息配置。
stFocusCfg	AF 统计信息配置。

### 【注意事项】

无

### 【相关数据类型及接口】

无

## ISP\_STATISTICS\_CTRL\_U

### 【说明】

定义 ISP 统计信息使能。



### 【定义】

```
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  bit22Rsv         : 22;
    };
}ISP_STATISTICS_CTRL_U;
```

### 【成员】

成员名称	描述
bit1AeStat1	全局五段直方图使能。Hi3518EV200 不支持。
bit1AeStat2	分区间五段直方图使能。Hi3518EV200 不支持。
bit1AeStat3	全局 256 段直方图使能。
bit1AeStat4	全局统计平均值使能。
bit1AeStat5	分区间统计平均值使能。
bit1AwbStat1	RGB 域 AWB 全局统计信息使能。
bit1AwbStat2	RGB 域 AWB 分区间统计信息使能。
bit1AwbStat3	Bayer 域 AWB 全局统计信息使能。
bit1AwbStat4	Bayer 域 AWB 分区间统计信息使能。
bit1AfStat	AF 统计信息使能。
bit22Rsv	保留位。

### 【注意事项】



用户可以关闭不使用的统计信息，如此就不会在中断中读出相应统计信息，从而提高系统性能。

#### 【相关数据类型及接口】

无

## ISP\_AE\_STATISTICS\_CFG\_S

#### 【说明】

定义 AE 统计信息配置。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
au8HistThresh	五段直方图统计的门限值。Hi3518EV200 不支持。
enAESwitch	AE 统计模块（包括直方图和平均值统计）在 ISP pipeline 的位置，默认值为 0 0: After static WB; 1: Immediately from sensor, channel 1 (for WDR modes); (Hi3518EV200 不支持) 2: After shading; (Hi3518EV200 不支持) 3: After GammaFE; (Hi3518EV200 不支持) 4: After DRC; 5: Immediately from sensor, channel 2 (for WDR modes); (Hi3518EV200 不支持) 6: After WDR stitch; (Hi3518EV200 不支持) 7: After BLC, channel 2 (for WDR modes), only when GammaFE is after BLC of channel 1. (Hi3518EV200 不支持) 8: After ISP digital gain;
enHistSwitch	全局 256 段直方图统计模块在 ISP pipeline 的位置，默认值为 0。 0: Same as AE; 1: Immediately from sensor, channel 1 (for WDR modes); 2: After shading; 3: After GammaFE



成员名称	描述
	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. Hi3518EV200 不支持。
enAESumSwitch	全局和分区间统计平均值统计模块在 ISP pipeline 的位置，默认值为 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. Hi3518EV200 不支持。

#### 【注意事项】

15\*17 分区间权重表配置会影响全局 256 段直方图和全局 4 分量平均值的计算，与 AE 统计模块在 ISP pipeline 的位置无关。

#### 【相关数据类型及接口】

无

## ISP\_WB\_STATISTICS\_CFG\_S

#### 【说明】

定义 AWB 统计信息配置。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
stBayerCfg	Bayer 域 AWB 统计信息配置。
stRGBCfg	RGB 域 AWB 统计信息配置。Hi3518EV200 不支持。

【注意事项】

Hi3518EV200 仅支持 Bayer 域统计。

【相关数据类型及接口】

无

## ISP\_WB\_STATISTICS\_CFG\_PARA\_S

【说明】

定义 AWB 统计信息配置参数。

【定义】

```
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;
```

【成员】

成员名称	描述
u16WhiteLevel	统计白点信息时，找白点的亮度上限。取值范围[0x0, 0xFFFF]，默认值 0xFFFF。
u16BlackLevel	统计白点信息时，找白点的亮度下限。取值范围[0x0, 0xFFFF]，默认值 0x0。
u16CbMax	统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。
u16CbMin	统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。
u16CrMax	统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。



成员名称	描述
u16CrMin	统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。
u16CbHigh	统计白点信息时，六边形白点区域限制 CrMax 对应的 Cb 值，8bit 精度，默认值 512。Hi3518EV200 不支持。
u16CbLow	统计白点信息时，六边形白点区域限制 CrMin 对应的 Cb 值，8bit 精度，默认值 128。Hi3518EV200 不支持。
u16CrHigh	统计白点信息时，六边形白点区域限制 CbMax 对应的 Cr 值，8bit 精度，默认值 512。Hi3518EV200 不支持。
u16CrLow	统计白点信息时，六边形白点区域限制 CbMin 对应的 Cr 值，8bit 精度，默认值 128。Hi3518EV200 不支持。

具体含义参见图 2-6。

#### 【注意事项】

- 用户调用海思 AWB 库，AWB 算法会根据当前照度、色温等信息实时刷新统计信息相关参数，此时用户通过 [HI\\_MPI\\_ISP\\_SetStatisticsConfig](#) 配置 AWB 统计参数不生效。
- 后续版本会提供接口，支持用户使能、关闭 AWB 算法刷新统计参数的功能。
- Hi3518EV200 暂不支持 u16CbHigh、u16CbLow、u16CrHigh、u16CrLow 四个参数配置。

#### 【相关数据类型及接口】

无

## ISP\_FOCUS\_STATISTICS\_CFG\_S

#### 【说明】

定义 AF 统计信息配置。

#### 【定义】

```
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;
```

#### 【成员】



成员名称	描述
stConfig	AF 全局配置参数。
stHParam_IIR0	奇数行水平滤波器 IIR 参数设置。
stHParam_IIR1	偶数行水平滤波器 IIR 参数设置。
stVParam_FIR0	奇数行垂直滤波器 FIR 参数设置。
stVParam_FIR1	偶数行垂直滤波器 FIR 参数设置。
stFVParam	统计结果输出格式参数配置。

【注意事项】

无

【相关数据类型及接口】

无

## ISP\_AF\_CFG\_S

【说明】

定义 AF 统计信息配置参数。

【定义】

```
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;
    ISP_AF_STATISTICS_POS_E enStatisticsPos ;
}
```

【成员】

成员名称	描述
bEnable	AF 使能。
u16Hwnd	AF 统计水平方向窗口个数，取值范围[1, 17]。
u16Vwnd	AF 统计垂直方向窗口个数，取值范围[1, 15]。





成员名称	描述
u16Hsize	AF 统计输入图像宽度，取值范围[1, 0xFFF]。
u16Vsize	AF 统计输入图像高度，取值范围[1, 0xFFF]。
enPeakMode	PEAK 模式，决定分区间 IIR 统计值（u16h1 和 u16h2）是否进行求峰值处理。与 enSquMode 共同影响分区间统计值。
enSquMode	平方模式，决定分区间水平和垂直方向统计值是否进行平方处理。与 enPeakMode 共同影响分区间统计值。
enStatisticsPos	统计信息输出位置，可以选择从 YUV 域输出统计信息或从 RAW 域输出统计信息。

#### 【注意事项】

- 分区间 IIR 统计值（u16h1 和 u16h2）的计算方式为：  
enPeakMode 为 ISP\_AF\_STA\_NORM 且 enSquMode 为 ISP\_AF\_STA\_SUM\_NORM 模式时，对分区间内每个像素的 IIR 滤波器输出值求和即为分区间 IIR 统计值（u16h1 和 u16h2）；  
enPeakMode 为 ISP\_AF\_STA\_NORM 且 enSquMode 为 ISP\_AF\_STA\_SUM\_SQU 模式时，对分区间内每个像素的 IIR 滤波器输出值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）；  
enPeakMode 为 ISP\_AF\_STA\_PEAK 且 enSquMode 为 ISP\_AF\_STA\_SUM\_NORM 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求和即为分区间 IIR 统计值（u16h1 和 u16h2）；  
enPeakMode 为 ISP\_AF\_STA\_PEAK 且 enSquMode 为 ISP\_AF\_STA\_SUM\_SQU 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）。
- 分区间 FIR 统计值（u16v1 和 u16v2）的计算方式为：  
enSquMode 为 ISP\_AF\_STA\_SUM\_NORM 模式时，对分区间内每个像素的 FIR 滤波器输出值求和即为分区间 FIR 统计值（u16v1 和 u16v2）；  
enSquMode 为 ISP\_AF\_STA\_SUM\_SQU 模式时，对分区间每个像素的 FIR 滤波器输出值平方后求和即为分区间 FIR 统计值（u16v1 和 u16v2）。
- 分区间亮度统计值（u16y）的计算方式为：  
对分区间内每个像素的亮度求和即为分区间亮度统计值（u16y）。

#### 【相关数据类型及接口】

无

## ISP\_AF\_PEAK\_MODE\_E

#### 【说明】

PEAK 模式，决定分区间 IIR 统计值（u16h1 和 u16h2）是否进行求峰值处理。



【定义】

```
typedef enum hiISP_AF_PEAK_MODE_E
{
    ISP_AF_STA_NORM          = 0,
    ISP_AF_STA_PEAK          ,
    ISP_AF_STA_BUTT
}
```

【成员】

成员名称	描述
ISP_AF_STA_NORM	当 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，对分区间内每个像素的 IIR 滤波器输出值求和即为分区间 IIR 统计值（u16h1 和 u16h2）； 当 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，对分区间内每个像素的 IIR 滤波器输出值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）。
ISP_AF_STA_PEAK	当 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求和即为分区间 IIR 统计值（u16h1 和 u16h2）； 当 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）。

【注意事项】

垂直方向 FIR 滤波器统计值不受该字段影响。

【相关数据类型及接口】

无

## ISP\_AF\_SQU\_MODE\_E

【说明】

平方模式，决定分区间统计值是否进行平方处理。

【定义】

```
typedef enum hiISP_AF_SQU_MODE_E
{
    ISP_AF_STA_SUM_NORM      = 0,
    ISP_AF_STA_SUM_SQU      ,
    ISP_AF_STA_SUM_BUTT
}
```



}

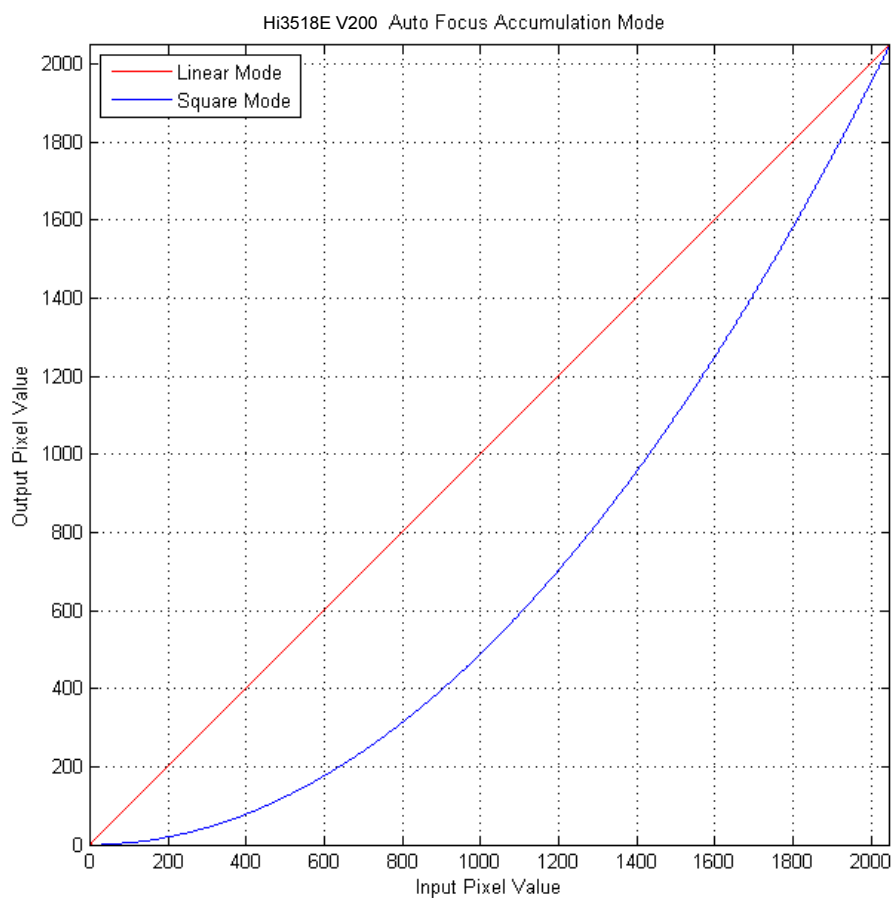
#### 【成员】

成员名称	描述
ISP_AF_STA_SUM_NORM	<p>水平方向统计值：</p> <p>当 enPeakMode 为 ISP_AF_STA_NORM 模式时，对分区间内每个像素的 IIR 滤波器输出值求和即为分区间 IIR 统计值（u16h1 和 u16h2）；</p> <p>当 enPeakMode 为 ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求和即为分区间 IIR 统计值（u16h1 和 u16h2）。</p> <p>垂直方向统计值：</p> <p>对分区间内每个像素的 FIR 滤波器输出直接累加即为分区间 FIR 统计值（u16v1 和 u16v2）。</p>
ISP_AF_STA_SUM_SQU	<p>水平方向统计值：</p> <p>当 enPeakMode 为 ISP_AF_STA_NORM 模式时，对分区间内每个像素的 IIR 滤波器输出值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）；</p> <p>当 enPeakMode 为 ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的 IIR 滤波器输出值取最大值，再对多行的最大值求平方和即为分区间 IIR 统计值（u16h1 和 u16h2）。</p> <p>垂直方向统计值：</p> <p>对分区间内每个像素的 FIR 滤波器输出平方后累加即为分区间 FIR 统计值（u16v1 和 u16v2）。</p>

#### 【注意事项】

采用 Square 模式，会对滤波器输出归一化后平方再做统计，如 7-1 所示，input 为滤波后输出像素值，output 为处理后输出值，相比 normal 模式，在临近焦点附近可以得到比较陡峭的 FV 曲线，并且对幅值较小的噪声信号有一定的抑制作用，但也会使 FV 曲线平坦区域更加平坦，用户应该根据实际的场景需求来设定这个参数。

图8-1 Square 模式



【相关数据类型及接口】

无

## ISP\_AF\_STATISTICS\_POS\_E

【说明】

AF 统计信息位置，统计信息可以从 YUV 域输出，也可以从 Bayer 域输出

【定义】

```
typedef enum hiISP_AF_STATISTICS_POS_E
{
    ISP_AF_STATISTICS_YUV          = 0,
    ISP_AF_STATISTICS_RAW          ,
    ISP_AF_STATISTICS_BUTT
}
```

【成员】



成员名称	描述
ISP_AF_STATISTICS_YUV	当 enStatisticsPos 为 ISP_AF_STATISTICS_YUV 模式时，AF 统计信息从 yuv 域输出；
ISP_AF_STATISTICS_RAW	当 enStatisticsPos 为 ISP_AF_STATISTICS_RAW 模式时，AF 统计信息从 Bayer 域输出；

【注意事项】

无

【相关数据类型及接口】

无

## ISP\_AF\_H\_PARAM\_S

【说明】

AF 统计水平滤波器 IIR 参数设置。

【定义】

```
typedef enum hiISP_AF_H_PARAM_S
{
    HI_BOOL    abIIREn[3];
    HI_S16     as16IIRGain[7];
    HI_U16     au16IIRShift[4];
    HI_U16     u16IIRThd;
}
```

【成员】

成员名称	描述
abIIREn[3]	IIR 级联中 IIR 的使能。
as16IIRGain[7]	IIR 滤波器系数，用于控制 IIR 滤波器的频率响应。
au16IIRShift[4]	IIR 滤波器移位调整，取值范围[0,0xF]。
u16IIRThd	IIR 滤波器统计阈值，取值范围[0,0x7FF]。当每个像素点的 FV 值大于阈值才参与对输出统计信息区块的 FV 值累积。

【注意事项】

as16IIRGain[0]为无符号 8bit 数，其他 as16IIRGain[1]-[6]为有符号 10bit 数，且不能等于-512。IIR 使能和滤波器系数以及 IIRShift 通过 PQ Tools 可以生成，用户可以根据实际需要调节 u16IIRThd 以获取最优 FV 统计值输出。



【相关数据类型及接口】

无

## ISP\_AF\_V\_PARAM\_S

【说明】

AF 统计垂直滤波器 FIR 参数设置。

【定义】

```
typedef enum hiISP_AF_V_PARAM_S
{
    HI_S16 as16FIRH[5];
    HI_U16 u16FIRThd;
}
```

【成员】

成员名称	描述
as16FIRH[5]	FIR 滤波器系数，取值范围[-31,31]。用于控制 FIR 滤波器的频率响应。
u16FIRThd	FIR 滤波器统计阈值，取值范围[0,0x7FF]。当每个像素点的 FV 值大于阈值才参与对输出统计信息区块的 FV 值累积。

【注意事项】

FIR 滤波器系数通过 PQ Tools 可以生成，用户可以根据实际需要调节 u16FIRThd 以获取最优垂直 FV 统计值输出。

【相关数据类型及接口】

无

## ISP\_AF\_FV\_PARAM\_S

【说明】

AF 统计结果输出格式参数。

【定义】

```
typedef enum hiISP_AF_H_PARAM_S
{
    HI_U16 u16AccShiftY;
    HI_U16 au16AccShiftH[2];
    HI_U16 au16AccShiftV[2];
}
```



#### 【成员】

成员名称	描述
u16AccShiftY	亮度 Y 统计值移位寄存器值，取值范围[0,0xF]。
au16AccShiftH[2]	水平 IIR 滤波值统计值移位寄存器值，取值范围[0,0xF]。
au16AccShiftV[2]	垂直 FIR 滤波值统计值移位寄存器值，取值范围[0,0xF]。

#### 【注意事项】

FV 值用 `unsinged 16bit` 表示，当场景细节较多或噪声较多时，FV 值可能会溢出，这就需要对输出进行向右移位以保证输出在合理范围。用户根据自己的需求对相应的 `shift` 值进行调节。

#### 【相关数据类型及接口】

无

## ISP\_STATISTICS\_S

#### 【说明】

定义统计信息。

#### 【定义】

```
typedef struct hiISP_STATISTICS_S
{
    ISP_AE_STATISTICS_S    stAESTat;
    ISP_WB_STATISTICS_S    stWBStat;
    ISP_FOCUS_STATISTICS_S stFocusStat;
} ISP_STATISTICS_S;
```

#### 【成员】

成员名称	描述
stAESTat	AE 统计信息。
stAWBStat	AWB 统计信息。
stAFStat	AF 统计信息。

#### 【注意事项】

无

#### 【相关数据类型及接口】

无



## ISP\_AE\_STATISTICS\_S

### 【说明】

定义 AE 统计信息。

### 【定义】

```
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;
```

### 【成员】

成员名称	描述
au16Hist5Value	全局五段直方图信息。
au16ZoneHist5Value	分区间五段直方图信息。
au32Hist256Value	全局 256 段直方图信息。
au16GlobalAvg	全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
au16ZoneAvg	分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。

### 【注意事项】

Hi3518EV200 不支持全局五段直方图统计信息及分区间五段直方图统计信息。

### 【相关数据类型及接口】

无

## ISP\_WB\_STATISTICS\_S

### 【说明】

定义 AWB 统计信息。

### 【定义】

```
typedef struct hiISP_WB_STATISTICS_S
{
    ISP_WB_BAYER_STATISTICS_INFO_S stBayerStatistics;
    ISP_WB_RGB_STATISTICS_INFO_S stRGBStatistics;
```





```
} ISP_WB_STATISTICS_S;
```

#### 【成员】

成员名称	描述
stBayerStatistics	Bayer 域 AWB 统计信息。
stRGBStatistics	RGB 域 AWB 统计信息。Hi3518EV200 不支持。

#### 【注意事项】

Hi3518EV200 只支持 Bayer 域统计信息，不支持 RGB 域统计信息。

#### 【相关数据类型及接口】

无

## ISP\_FOCUS\_STATISTICS\_S

#### 【说明】

定义 AF 统计信息。

#### 【定义】

```
typedef struct hiISP_FOCUS_STATISTICS_S
{
    ISP_FOCUS_ZONE_S stZoneMetrics[AF_ZONE_ROW][AF_ZONE_COLUMN];
} ISP_FOCUS_STATISTICS_S;
```

#### 【成员】

成员名称	描述
stZoneMetrics	AF 统计信息。

#### 【注意事项】

Hi3518EV200 支持 Bayer 域统计信息，也支持 RGB 域统计信息。

#### 【相关数据类型及接口】

[ISP\\_FOCUS\\_ZONE\\_S](#)

## ISP\_WB\_BAYER\_STATISTICS\_INFO\_S

#### 【说明】

定义 Bayer 域 AWB 统计信息。

#### 【定义】



```
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;
```

**【成员】**

成员名称	描述
u16GlobalR	Bayer 域全局统计的 R 分量平均值，取值范围[0x0, 0xFFfF]。
u16GlobalG	Bayer 域全局统计的 G 分量平均值，取值范围[0x0, 0xFFfF]。
u16GlobalB	Bayer 域全局统计的 B 分量平均值，取值范围[0x0, 0xFFfF]。
u16CountAll	全局统计的六边形灰色区域的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。
u16CountMin	全局统计的小于 BlackLevel 的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。
u16CountMax	全局统计的大于 Whitelevel 的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。
au16ZoneAvgR	Bayer 域分区间统计的 R 分量平均值，取值范围[0x0, 0xFFfF]。
au16ZoneAvgG	Bayer 域分区间统计的 G 分量平均值，取值范围[0x0, 0xFFfF]。
au16ZoneAvgB	Bayer 域分区间统计的 B 分量平均值，取值范围[0x0, 0xFFfF]。
au16ZoneCountAll	分区间统计的六边形灰色区域的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。
au16ZoneCountMin	分区间统计的小于 BlackLevel 的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。
au16ZoneCountMax	分区间统计的大于 Whitelevel 的像素个数，已做归一化，取值范围[0x0, 0xFFFF]。



【注意事项】

无

【相关数据类型及接口】

无

## ISP\_WB\_RGB\_STATISTICS\_INFO\_S

【说明】

定义 RGB 域 AWB 统计信息。

【定义】

```
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;
```

【成员】

成员名称	描述
u16GlobalGR	RGB 域全局统计的 G/R 值。
u16GlobalGB	RGB 域全局统计的 G/B 值。
u32GlobalSum	RGB 域全局统计的灰点个数。
au16ZoneGR	RGB 域分区间统计的 G/R 值。
au16ZoneGB	RGB 域分区间统计的 G/B 值。
au32ZoneSum	RGB 域分区间统计的灰点个数。

【注意事项】

Hi3518EV200 不支持 RGB 域内的相关统计信息。

【相关数据类型及接口】

无

## ISP\_FOCUS\_ZONE\_S

【说明】



定义 AF 统计信息参数。

#### 【定义】

```
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;
```

#### 【成员】

成员名称	描述
u16v1	分区间统计的 AF 垂直方向奇数列 FIR 滤波器的统计值。
u16h1	分区间统计的 AF 水平方向奇数行 IIR 滤波器的统计值。
u16v2	分区间统计的 AF 垂直方向偶数列 FIR 滤波器的统计值。
u16h2	分区间统计的 AF 水平方向偶数行 IIR 滤波器的统计值。
u16y	分区间统计的 AF 亮度统计值，区块中每个像素点的亮度累加值。

#### 【注意事项】

u16y 为每个块的 Y 分量累加值，用户可以用此值辅助 AF 算法，比如某些块中含有高亮光源，容易对 FV 值产生干扰，可根据亮度情况降低块 FV 对全局 FV 的影响，亦可用这些块实现一个移动侦测算法，用于检测聚焦过程中的一些运动干扰，用户可以根据自己 AF 算法的需求使用这个字段的统计信息。

#### 【相关数据类型及接口】

无



# 9 Cmos 默认参数配置

## 9.1 概述

每一个 sensor 对应一个 xxxx\_cmos.c 文件，包括系统控制（参考第 2 章）和默认参数配置部分。针对默认参数配置部分，支持两种配置方式：xxxx\_cfg.ini 文件解析方式和函数直接初始化参数方式。这两种配置方式可通过修改 Makefile 来指定（如 SDK 软件包中 mpp 文件目录下的 Makefile.param 中 ISP\_INI\_CONFIG 变量）。当使用 xxxx\_cfg.ini 文件解析方式配置时，支持调用接口 **sensor\_set\_inifile\_path(const char \*pcPath)** 配置 ini 文件路径。

启用 ini 文件解析方式，参考流程如下：

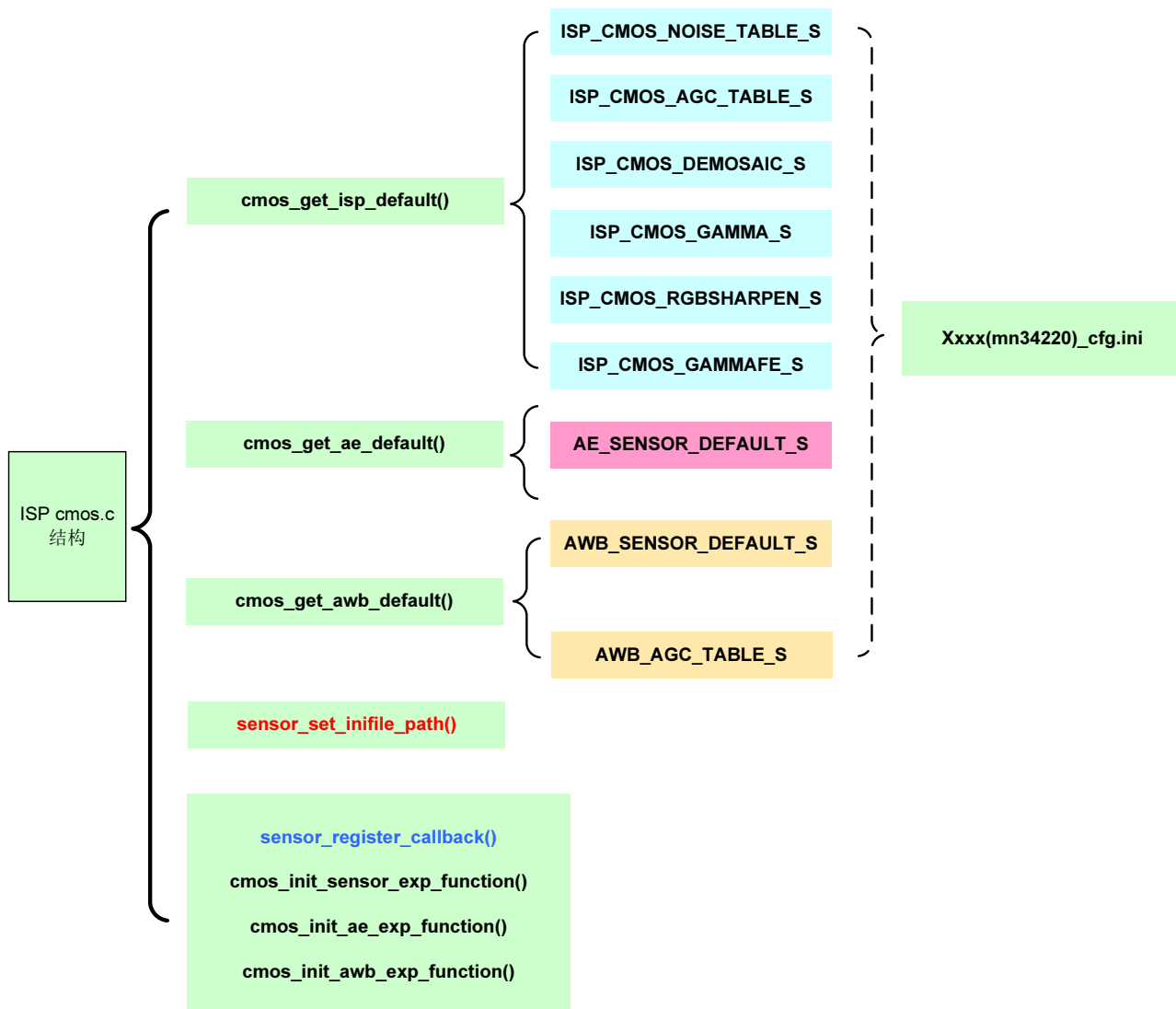
- 步骤 1. 修改 mpp 文件目录下的 Makefile.param 中 ISP\_INI\_CONFIG = y。
- 步骤 2. 在 ISP 目录下编译 sensor 库。
- 步骤 3. 拷贝 ISP 目录下编译出的 sensor 库、ini 相关库文件到用户的指定路径目录下。
- 步骤 4. 在业务程序中调用接口 **sensor\_set\_inifile\_path(const char \*pcPath)** 配置 ini 文件路径（参考 Sample）。
- 步骤 5. 在业务程序的 makefile 文件中添加 sensor 和 ini 相关库文件（lib\_iniparser 与 lib\_cmoscfg），并编译（参考 Sample）。

----结束



## 9.2 Cmos 结构图示意

图9-1 Cmos 结构示意图



## 9.3 INI 文件使用说明

每一个 Sensor 对应一个 `xxxx_cfg.ini` 文件，如 `panasonic_mn34220/mn34220_cfg.ini`。详情参考 `sensor` 文件夹下的 `ini` 文件。

在 `mn34220_cfg.ini` 文件中，分别对应【AE】、【AWB】和【ISP】三个字段，每一个字段下包含参数组数和每一组对应的参数列表。

### 9.3.1 AE

在 `ini` 文件中的【AE】字段下参数如下：



参数	描述
AEModeNumber	AE 参数模式组数；
AeCompensation_0	AE 亮度目标值；
MaxIntTimeTarget_0	最大曝光时间目标值；
MinIntTimeTarget_0	最小曝光时间目标值；
MaxAgainTarget_0	最大 sensor 模拟增益目标值；
MinAgainTarget_0	最小 sensor 模拟增益目标值；
MaxDgainTarget_0	最大 sensor 数字增益目标值；
MinDgainTarget_0	最小 sensor 数字增益目标值；
ISPDgainShift_0	ISP 数字增益精度；
MinISPDgainTarget_0	最小 ISP 数字增益目标值；
MaxISPDgainTarget_0	最大 ISP 数字增益目标值；



说明

后缀\_0 表示第一组模式参数，以下【AWB】和【ISP】中后缀含义相同。

## 9.3.2 AWB

在 ini 文件中的【AWB】字段下参数如下：

参数	描述
AWBModeNumber	AWB 参数模式组数；
HighColorTemp_0	CCM 高色温；
HighCCM_0	高色温颜色还原矩阵；
MidColorTemp_0	CCM 中色温；
MidCCM_0	中色温颜色还原矩阵；
LowColorTemp_0	CCM 低色温；
LowCCM_0	低色温颜色还原矩阵；
WbRefTemp_0	静态白平衡校正色温；
GainOffset_0	静态白平衡的 R、Gr、Gb、B 颜色通道的增益；
WbPara_0	校正工具给出的白平衡参数；
SatValid_0	Sat 数据有效使能；
Saturation_0	根据增益动态调节饱和度的插值数组；



### 9.3.3 ISP

在 ini 文件中的【ISP】字段下参数如下：

参数	描述
ISPMoDeNumber	ISP 参数模式组数；
AgcValid_0	Agc 数据有效使能；
SharpenAltD_0	根据增益动态调节图像大边缘锐度的插值数组；
SharpenAltUd_0	根据增益动态调节图像小纹理锐度的插值数组；
SnrThresh_0	根据增益动态设置图像去噪强度的插值数组；
DemosaicLumThresh_0	该数组用来设置图像的大边缘锐度的亮度门限值；
DemosaicNpOffset_0	该数组用来设置图像的噪声参数；
GeStrength_0	该数组用来设置绿色平衡模块的参数；
SharpenRGB_0	根据增益动态调节图像整体锐度的插值数组；
WeightValid_0	Weight 数据有效使能；
NoiseProfileWeight_0	该数组用来设置与 sensor 特性相关的噪声型式，作为去噪模块的输入；
DemosaicWeight_0	该数组用来设置与 sensor 特性相关的噪声型式模块，作为 demosaic 模块的输入；
demosaicValid_0	demosaic 数据有效使能；
VhSlope_0	垂直/水平混合的斜率门限；
AaSlope_0	角度混合的斜率门限；
VaSlope_0	VH-AA 混合的斜率门限；
UuSlope_0	不限方向的混合的斜率门限；
SatSlope_0	饱和度混合的斜率门限；
AcSlope_0	高频分量滤波斜率门限；
FcSlope_0	去伪彩色斜率门限；
VhThresh_0	垂直/水平混合的门限；
AaThresh_0	角度混合门限；
VaThresh_0	VH-AA 混合门限；
UuThresh_0	不限方向的混合门限；
SatThresh_0	饱和度混合门限；





参数	描述
AcThresh_0	高频分量滤波门限;
gammaRGBValid_0	Gamma 数据有效使能;
gammaRGB0_0	RGBGamma 表第 1 段;
gammaRGB1_0	RGBGamma 表第 2 段;
gammaRGB2_0	RGBGamma 表第 3 段;
gammafevalid_0	Gammafe 数据有效使能;
gammafe0_0	Gammafe0 表;
gammafe1.0_0	Gammafe1 表第 1 段;
gammafe1.1_0	Gammafe1 表第 2 段;
gammafe1.2_0	Gammafe1 表第 3 段;

## 9.4 注意事项

以上 mn34220\_cfg.ini 内的参数, 包含了 ISP cmos 一组参数, 里面的每一种参数模式组内每一小组 (空行隔开) 可根据用户需要进行裁剪, 可以根据需求删除以下两项, 如:

```
SatValid_0 = 1
Saturation_0 = xxxxxxx
```

注意, 只有带有 Valid 字符的参数可根据需求删除, 没有带 Valid 字符的参数是必配项, 而参数组数只能根据每项字段下的参数组数进行设置, 如设置为 3, 表示可以支持 3 组参数配置, 格式参考 mn34220\_cfg.ini, 对于【AE】、【AWB】与【ISP】三个字段最大支持 6 组参数, 可在相应字段下增加与减少参数组数, 参数格式需注意后缀, 如 xxxx\_0 ~ xxxx\_5。

对于各个参数模式组内的每一个参数名称应该严格对应, 如 mn34220\_cfg.ini 文件中的参数名称, 不能随意更改其他名称, 否则会导致参数配置失败, 如果有更改参数名称需求, 需同步更改 hi\_cmoscfg/hi\_cmos\_cfg.c 文件中对应名称。

对于每一个参数组中, 如:

```
Saturation_0
=0x80|0x80|0x80|0x78|0x70|0x68|0x58|0x48|0x40|0x38|0x38|0x38|0x38|0x38|0x38|
```

两个参数值分隔符可使用 “|” 或者 “,”, 其他字符不支持。对于参数值支持 16 进制与 10 进制两种形式。

在 INI 配置文件中, 特别注意 Gamma 表的格式, 需将 Gamma 尽量均分为 3 段进行配置, 以便于正确解析, 具体参考 mn34220\_cfg.ini 文件格式。

在使用 INI 配置文件需注意:

- 请注意不要出现相同的参数配置项，否则会出现某些解析错误。
- 字段【AE/AWB/ISP】书写时，请确保‘[]’在同一行出现。
- 字段【AE/AWB/ISP】名前除了空格符合法外，其它的一切字符都将会导致该字段名及其子项不可识别。
- 确保参数名称与它的值在同一行，否则会出现该参数名称不被识别的异常。
- 请不要使用诸如\r,\n,[,],';','"',等特殊字符，以免这些特殊的字符组合出现不可预见的异常。
- 在 INI 文件中随意使用';'，因为';'是以注释符存在的，';'后的所有字符都将被认为是注释，如果一定要使用，请以中文';'代替。



# 10 Debug

## 10.1 概述

ISP 提供 Debug MPI 供客户调用，以方便客户定位相关图像质量问题。

## 10.2 功能描述

Debug 提供记录 ISP 在运行过程中的状态信息，以方便记录在 ISP 运行过程中出现的异常状态，需要记录的帧数可以有用户自己指定。

## 10.3 API 参考

- [HI\\_MPI\\_ISP\\_SetDebug](#): 设置 ISP 调试接口。
- [HI\\_MPI\\_ISP\\_GetDebug](#): 获取 ISP 调试接口。

### HI\_MPI\_ISP\_SetDebug

#### 【描述】

设置 ISP 调试信息属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_SetDebug(ISP_DEV IspDev, const ISP\_DEBUG\_INFO\_S *  
pstIspDebug);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspDebug	调试信息。	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

【注意】

- 调试信息的成员变量为 SYS，可以设置使能开关。
- 需要用户来分配内存以存储相应的调试信息，且需要将分配的内存地址作为输入传入。

【举例】

```
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;
}
```

#### 【相关主题】

无

## HI\_MPI\_ISP\_GetDebug

#### 【描述】

获取 ISP 调试信息属性。

#### 【语法】

```
HI_S32 HI_MPI_ISP_GetDebug(ISP_DEV IspDev, ISP_DEBUG_INFO_S *
pstIspDebug);
```

#### 【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIspDebug	调试信息。	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 <a href="#">错误码</a> 。

#### 【错误码】

接口返回值	含义
<a href="#">HI_ERR_ISP_NULL_PTR</a>	空指针错误。

#### 【需求】

- 头文件：hi\_comm\_isp.h、mpi\_isp.h
- 库文件：libisp.a

#### 【注意】



无

【举例】

无

## 10.4 数据类型

**ISP\_DEBUG\_INFO\_S**: 定义 ISP 调试信息属性。

### ISP\_DEBUG\_INFO\_S

【说明】

定义 ISP 调试信息属性。

【定义】

```
typedef struct hiISP_DEBUG_INFO_S
{
    HI_BOOL bDebugEn;
    HI_U32  u32PhyAddr;
    HI_U32  u32Depth;
} ISP_DEBUG_INFO_S;
```

【成员】

成员名称	描述
bDebugEn	使能调试。
u32PhyAddr	调试信息的物理地址。
u32Depth	调试深度，即需要获取调试信息的帧数。

【注意事项】

无

【相关数据类型及接口】

无



# 11 错误码

ISP API 错误码如表 11-1 所示。

表11-1 ISP API 错误码

错误代码	宏定义	描述
0xA01C8006	HI_ERR_ISP_NULL_PTR	空指针错误
0xA01C8003	HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效
0xA01C8008	HI_ERR_ISP_NOT_SUPPORT	当前 ISP 不支持
0xA01C8043	HI_ERR_ISP_SNS_UNREGISTER	Sensor 未注册
0xA01C8041	HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
0xA01C8040	HI_ERR_ISP_NOT_INIT	ISP 没有初始化
0xA01C8044	HI_ERR_ISP_INVALID_ADDR	无效地址
0xA01C8042	HI_ERR_ISP_ATTR_NOT_CFG	属性未配置
0xA01C8045	HI_ERR_ISP_NOMEM	内存不足
0xA01C8046	HI_ERR_ISP_NO_INT	ISP 无中断



# 12 Proc 调试信息说明

## 12.1 概述

调试信息采用了 `proc` 文件系统，可实时反映当前系统的运行状态，所记录的信息可供问题定位及分析时使用。

### 【文件目录】

`/proc/umap/isp`

### 【开启方法】

- Linux  
加载 ISP 的 KO 时，添加模块参数 `proc_param=n`。其中 `n=0` 关闭 `proc`，非 0 表示每隔 `n` 帧更新一次 `proc`。如：`insmod hi3518e_isp.ko proc_param=30` 表示每隔 30 帧更新一次 `proc` 信息。
- Huawei LiteOS  
在 `release/ko` 下 `sdk_init.c` 中 `ISP_init` 函数中，添加参数 `stIsp_Param.u32ProcParam = n` 传入 `ISP_ModInit` 中，其中 `n=0` 关闭 `proc`，并且不分配存储 `Proc` 信息的内存；非 0 表示每隔 `n` 帧更新一次 `proc`，并且会分配存储 `Proc` 信息的内存。如：`u32ProcParam = 30` 表示每隔 30 帧更新一次 `proc` 信息。

**注意：**开启 `Proc` 信息会消耗 CPU 资源。推荐设为 30 帧更新一次，或仅 `Debug` 时开启。

### 【信息查看方法】

- 在控制台上可以使用 `cat` 命令查看信息，例如 `cat /proc/umap/isp`；也可以使用其他常用的文件操作命令，例如 `cp /proc/umap/isp ./-rf`，将 ISP 的 `proc` 文件拷贝到当前目录。
- 在应用程序中可以将上述文件当作普通只读文件进行读操作，例如 `fopen`、`fread` 等。Huawei LiteOS 只支持 `cat` 查看，不支持 `cp` 方式。



说明

参数在描述时有以下 2 种情况需要注意：





- 取值为{0, 1}的参数，如未列出具体的取值和含义的对应关系，则参数为 1 时表示肯定，为 0 时表示否定。
- 取值为{aaa, bbb, ccc}的参数，未列出具体的取值和含义的对应关系，但可直接根据取值 aaa、bbb 或 ccc 判断参数含义。

## 12.2 ISP

### 【调试信息】

```
# cat /proc/umap/isp
[ISP] Version: [Hi3518EV200_ISP_V1.0.0.0 Debug], Build Time[Dec 6 2015,
00:14:34]
-----MODULE PARAM-----
proc_param      int_bottomhalf  update_pos      pwm_num         lsc_update_mode
30              0              0              1              0

-----DRV INFO-----
IspDev          IntCnt          IntT            MaxIntT          IntGapT          MaxGapT          IntRat
IspResetCnt
0              1783          995            1020            40045          54998614          25
2

PtIntCnt         PtIntT          PtMaxIntT        PtIntGapT        PtMaxGapT        PtIntRat
SensorCfgT
1783           5              6             40042          54998610          25              8

-----PubAttr INFO-----
WndX            WndY            WndW            WndH            Bayer
0              0              1920           1080           GRBG

[AE] Version: [Hi3518EV200_ISP_V1.0.0.0 B010 Debug], Build Time[Apr 13
2016, 23:11:41]
-----AE INFO-----
Again          Dgain          IspDg          SysGain          Iso          Line          AEInter          Incrmnt          Exp          1stTime
3871          3072          2132          24179          2361          1271          1          256          30721062
800000
Comp          EVbias          OriAve          Offset          Speed          Tole          Error          Fps          RealFps          BDelay
WDelay
56          1024          44          12          64          2          0          25.00          2500          8          0

MaxLineMaxLineT  MaxAgT  MaxDgT  MaxIDgT  MaxSgT  Thresh0  Thresh1  Thresh2
Thresh3
1271  65535  4096  3072  8192  196608  12  24  96  128
```



```
ManuEn  MaLine  MaAg  MaDg  MaIspDg  WdrModeExpRatio  AnFlick  SlowMod
GainTh
      0      0      0      0      0  SensorWDR      64      0      1
98304
```

```
NodeId  IntTime  SysGain  IrisApe  UpStgy  DwStgy      Mltply
      0      32    1719      1      0      4      55008
      1    1271    1719      1      1      0    2184849
      2    1271   98304      1      4      1   124944384
```

```
AuIrEn  IrType  MaIrEn  DbgIrSt
      0  DCIris      0      0
```

[AWB] Version: [Hi3518EV200\_ISP\_V1.0.0.0 B010 Debug], Build Time[Apr 8 2016, 10:33:17]

```
-----AWB INFO-----
      Gain0  Gain1  Gain2  Gain3  CoTemp
      0x182  0x100  0x100  0x1fd  4291
Color00 Color01 Color02 Color10 Color11 Color12 Color20 Color21 Color22
      0x e9  0x  0  0x 17  0x8021  0x e4  0x 3d  0x800a  0x8048  0x 152
ManuEn      Sat  Zones  Speed
      0      92     32    128
```

```
-----DRC INFO-----
      En  ManuEn  DrcSt
      1      0    194
```

```
-----DEMOSAIC INFO-----
      VhSlope  UuSlope  VhLimit  VhOffset
      48      400     12      8
```

```
-----NR INFO-----
      En
      1
```

```
-----SHARPEN INFO-----
      En  LowLumaShoot  SharpenD  SharpenUD
      1      0      45      47
```

```
-----FPN INFO-----
      En  OpType  Strength  Offset  ModeCompressMode
      0  --      --      --
```

```
-----ACM INFO-----
      En  DemoEn  Mode  Stretch  Clip  Wrap  CbcrThrGainLuma
```



```

GainHueGainSta
    0    0    4    1    1    0    0    64    64    64
-----UVNR INFO-----
colorcast    thd    Sth
    0    10    0

-----RGBIR INFO-----
    En        AutoEn        AutoGain
    0          1          256

```

**【调试信息分析】**

记录当前 ISP 模块的使用情况。

**【参数说明】**

参数	描述
MODULE PARAM	proc_param 值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf 0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos 表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num 表示 pwm 使用情况。
	lsc_update_mode 表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
DRV INFO	IspDev ISP 设备号。
	IntCnt ISP 通道中断数。
	IntT ISP 中断占用时间。
	MaxIntT ISP 中断的最大占用时间。
	IntGapT ISP 相邻两中断之间的时间间隔。
	MaxGapT ISP 相邻两中断之间的最大时间间隔。
	IntRat 每秒 ISP 中断个数。
	IspResetCnt ISP 复位的次数。



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
	PtIntCnt	Port 通道的中断数。
	PtIntT	Port 中断占用时间。
	PtMaxIntT	Port 中断的最大占用时间。
	PtIntGapT	Port 相邻两中断之间的时间间隔。
	PtMaxGapT	Port 相邻两中断之间的最大时间间隔。
	PtIntRat	每秒 Port 中断个数。
	SensorCfgT	Sensor 配置时间。
PubAttr INFO	WndX	水平方向起始位置。
	WndY	垂直方向起始位置。
	WndW	图像宽度。
	WndH	图像高度。
	Bayer	输入 Bayer 图像数据格式。
AE INFO	Again	Sensor 模拟增益，10bit 精度。
	Dgain	Sensor 数字增益，10bit 精度。
	IspDg	ISP 数字增益，10bit 精度。
	SysGain	系统增益，10bit 精度。
	ISO	ISO 值,表示系统增益，以常数 100 乘以倍数为单位,例如系统中 sensor 的增益为 2 倍，ISP 的增益为 1 倍，那么整个系统的 ISO 值计算方式为：2*1*100=200，即系统



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
		ISO 为 200，本文档中涉及到的 ISO 都是采用这种计算方法。
	Line	Sensor 曝光时间，以行为单位。
	AEInter	AE 算法的运行间隔，2 表示每 2 帧运行一次 AE。
	Incrmnt	当前曝光量相比上帧曝光量增加的比例(8bit 精度)。
	Exp	当前曝光量，为快门、增益、光圈值的乘积。
	1stTime	AE 第一次稳定的时间，以 us 为单位。
	Comp	自动曝光调整时对曝光补偿量。
	EVbias	自动曝光调整时的曝光偏差值。
	OriAve	根据 256 段直方图计算出来的画面平均亮度
	Offset	AE 根据统计信息对平均亮度的偏移
	Speed	自动曝光调整时的初始步长。
	Tole	自动曝光调整时对曝光量的容忍偏差。
	Error	自动曝光调整时平均亮度的偏差。
	Fps	基准帧率。
	RealFps	实际帧率。
	BDelay	图像亮度低于目标亮度时，AE 调节的等待时间（单位：帧）
	WDelay	图像亮度高于目标亮度时，AE 调节的等待时间（单位：



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
		帧)
	MaxLine	Sensor 支持的最大曝光时间。
	MaxLineT	用户配置的最大曝光时间。
	MaxAgT	用户配置的最大 Sensor 模拟增益。
	MaxDgT	用户配置的最大 Sensor 数字增益。
	MaxIDgT	用户配置的最大 Isp 数字增益。
	MaxSgT	用户配置的最大系统增益。
	Thresh0	自动曝光调整时五段直方图第一段与第二段的分段点。
	Thresh1	自动曝光调整时五段直方图第二段与第三段的分段点。
	Thresh2	自动曝光调整时五段直方图第三段与第四段的分段点。
	Thresh3	自动曝光调整时五段直方图第四段与第五段的分段点。
	ManuEn	手动曝光开关。
	MaLine	手动曝光时间。
	MaAg	手动曝光的 Sensor 模拟增益。
	MaDg	手动曝光的 Sensor 数字增益。
	MaIspDg	手动曝光的 Isp 数字增益
	WdrMode	WDR 模式。
	ExpRatio	FSWDR 模式相邻 2 帧曝光比。



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
	AnFlick	自动抗闪开关。
	SlowMod	自动降帧模式。
	GainTh	进入自动降帧时(SLOW_SHUTTER mode)的系统增益。
	NodeId	节点 Id 号。
	IntTime	节点曝光时间，单位为微秒（us）。
	SysGain	节点增益，包括 Sensor 模拟、数字与 ISP 增益。
	IrisApe	节点光圈大小，仅支持 P-Iris 模式。
	UpStgy	AE 上行路线的分配策略。
	DwStgy	AE 下行路线的分配策略。
	Mltply	分配节点快门、增益、光圈值的乘积。
	AuIrEn	自动光圈使能开关。
	IrType	光圈类型。
	MaIrEn	手动光圈使能开关。
	DbgIrSt	光圈调试状态，0 为 Keep，1 为 Open，2 为 Close。
AWB INFO	Gain0	白平衡 R 通道增益。
	Gain1	白平衡 Gr 通道增益。
	Gain2	白平衡 Gb 通道增益。
	Gain3	白平衡 B 通道增益。



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
	CoTemp	当前检测的色温。
	Color00	色彩还原矩阵 RR 值。
	Color01	色彩还原矩阵 RG 值。
	Color02	色彩还原矩阵 RB 值。
	Color10	色彩还原矩阵 GR 值。
	Color11	色彩还原矩阵 GG 值。
	Color12	色彩还原矩阵 GB 值。
	Color20	色彩还原矩阵 BR 值。
	Color21	色彩还原矩阵 BG 值。
	Color22	色彩还原矩阵 BB 值。
	ManuEn	手动白平衡开关。
	Sat	饱和度值。
	Zones	自动白平衡算法选择。
	Speed	自动白平衡收敛速度。
DRC INFO	En	DRC 开关。
	ManuEn	DRC 手动开关。
	DrcSt	DRC 强度值。
DEMOSAI	VhSlope	垂直、水平边缘混合阈值的斜率。





参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
	UuSlope	全部边缘混合阈值的斜率。
	VhLimit	垂直、水平弱边缘基限值。
	VhOffset	垂直、水平弱边缘偏差值。
NR INFO	En	NR 使能的开关。
SHARPEN INFO	En	sharpen 使能开关。
	LowLumaShoot	低照度环境下，图像锐化后的 shoot 的严格控制。
	SharpenD	有方向锐化增益。
	SharpenUD	无方向锐化增益。
FPN INFO	En	FPN 校正的开关。
	OpType	类型选择(手动或自动)。
	Strength	手动时的 FPN 校正强度。
	Offset	黑帧平均像素值。
	ModeCompressMode	FPN 的压缩模式。
ACM INFO	En	ACM 开关。
	DemoEn	演示模式使能，左半部分不做，右半部做 ACM。
	Mode	ACMA 系数模式选择。
	Stretch	输入数据范围。



参数		描述
MODULE PARAM	proc_param	值为 0 关闭 proc，并且不分配存储 Proc 信息的内存；非 0 表示每隔 n 帧更新一次 proc，并且会分配存储 Proc 信息的内存。
	int_bottomhalf	0：获取 3A 统计信息操作等操作放在硬中断 ISR 中。 1：获取 3A 统计信息操作等操作放在中断底半部中，此时硬中断处理时间较短，但获取 3A 统计信息可能会不及时。Huawei LiteOS 不支持中断下半部功能，若在 Huawei LiteOS 下使能此参数，将打印不支持提示。
	update_pos	表示 sensor 寄存器配置中断更新位置是用帧起始中断还是在 AF 统计信息完成中断，0 表示在帧起始中断配置，1 表示在 AF 统计信息完成中断配置。
	pwm_num	表示 pwm 使用情况。
	lsc_update_mode	表示 lsc 表项配置位置，0 表示在用户态配置，1 表示在内核态 AF 统计信息完成中断配置。
	Clip	输出数据范围。
	Wrap	小数部分处理方式。
	CbcrThr	色度调整的阈值。
	GainLuma	亮度增益。
	GainHue	色调增益。
	GainSta	饱和度增益。
UVNR INFO	colorcast	偏色微调强度。
	thd	色度降噪的细调参数。
	Sth	色度降噪的粗调参数。
RGBIR INFO	En	RGBIR 开关。
	AutoEn	手动模式开关。
	AutoGain	自动模式下增益。

**【注意事项】**

用户通过 PQTools 手动配置 PubAttr 修改画面帧率时，sensor 的最大曝光时间会发生修改。绝大部分修改帧率时，帧率越高则最大曝光时间越低。但对于 ov9712，由于其修改帧率时的配置机制不同，帧率设置越高其最大曝光时间也会相应提高。