



3A HiISP

开发参考

文档版本 03

发布日期 2014-02-26

版权所有 © 深圳市海思半导体有限公司 2013-2014。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



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。

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
Hi3516	V100
Hi3518	V100


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。



符号	说明
 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 03(2014-02-26)

第 3 章 AE

新增 HI_MPI_ISP_SetAEDelayAttr 和 HI_MPI_ISP_GetAEDelayAttr;

3.5.1 AE ISP_AE_MODE_E 的【注意事项】新增“关于抗闪打开”的相关描述。

新增 AE_SENSOR_GAININFO_S。

3.5.2 AE, ISP_AE_ATTR_S 的【注意事项】中增加关于“最小曝光时间”的描述。

新增 AE_SENSOR_GAININFO_S; AE_SENSOR_EXP_FUNC_S 的【定义】和【成员】中新增 pfn_cmos_again_calc_table 和 pfn_cmos_dgain_calc_table;
AE_ACCURACY_E 的【定义】和【成员】中新增 AE_ACCURACY_TABLE;
AE_ACCURACY_S 的【成员】enAccuType 新增 TABLE 类型,【注意事项】中新增关于 TABLE 的相关说明; ISP_EXP_STA_INFO_S 新增【参数】s16HistError 及描述; 新增 ISP_AE_ROUTE_NODE_S、ISP_AE_ROUTE_S 和 ISP_AE_DELAY_S。

第 4 章 AWB

新增 HI_MPI_ISP_SetAWBAlgType、HI_MPI_ISP_GetAWBAlgType、
HI_MPI_ISP_SetAdvAWBAAttr、HI_MPI_ISP_GetAdvAWBAAttr、
HI_MPI_ISP_SetLightSource 和 HI_MPI_ISP_GetLightSource。

新增 ISP_AWB_ALG_TYPE_E、ISP_ADV_AWB_ATTR_S、
ISP_AWB_LIGHTSOURCE_INFO_S 和 ISP_AWB_ADD_LIGHTSOURCE_S。

第 6 章 IMP

6.3.3 数据类型, ISP_DRC_ATTR_S 的【成员】的描述与更新。

6.8.4 数据类型, ISP_DIS_INFO_S 的【成员】的描述有修改。



HI_MPI_ISP_GetGammaTable 的【注意】中增加事项。

新增 6.13 获取 ISP 模块虚拟地址。

ISP_DRC_ATTR_S 的【定义】和【描述】中增加 u32VarianceSpace 和 u32VarianceIntensity。

ISP_CR_ATTR_S 的【定义】和【描述】中 u8Strength[8];【注意事项】有更新。

ISP_DEMOSAIC_ATTR_S 的【定义】和【描述】中增加 u8LumThresh 和 u8NpOffset。

新增表 6-5、表 6-6。

文档版本 02(2013-09-25)

第 3 章 AE

HI_MPI_ISP_SetMEAttrEx、HI_MPI_ISP_GetMEAttrEx、
HI_MPI_ISP_QueryInnerStateInfoEx 的【语法】有更新。

新增 ISP_ME_ATTR_EX_S 和 ISP_INNER_STATE_INFO_EX_S。

ISP_AE_ATTR_EX_S 的【参数】u32SystemGainMax 的描述更新。

ISP_EXP_STA_INFO_S 新增【参数】u8ExpHistTarget[5]及描述；新增【注意事项】。

ISP_INNER_STATE_INFO_S 的【注意事项】中删除关于高精度 Sensor 模拟增益、数字增益和高精度 ISP 数字增益的相关描述。

第 10 章 Proc 调试信息

新增。

文档版本 01(2013-07-16)

第 1 次发布。



目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 功能描述.....	1
1.2.2 设计思路	2
1.2.3 文件组织	2
1.2.4 开发模式	3
1.2.5 内部流程	4
1.2.6 软件流程	5
2 系统控制.....	7
2.1 功能概述.....	7
2.2 API 参考	7
2.3 数据类型.....	34
3 AE.....	62
3.1 概述.....	62
3.2 重要概念.....	62
3.3 功能描述.....	63
3.4 API 参考	65
3.4.1 AE 库接口	65
3.4.2 AE 控制模块	67
3.4.3 AI 控制模块	87
3.5 数据类型.....	91
3.5.1 Register	91
3.5.2 AE	97
3.5.3 AI	110
4 AWB.....	114
4.1 概述.....	114
4.2 重要概念.....	114
4.3 功能描述.....	114



4.3.1 AWB 模块工作原理.....	114
4.4 API 参考	115
4.4.1 AWB 库接口.....	115
4.4.2 AWB 控制模块.....	118
4.4.3 WB 统计信息	131
4.5 数据类型.....	136
4.5.1 Register	136
4.5.2 WB.....	138
5 CCM.....	148
5.1 概述.....	148
5.2 重要概念.....	148
5.3 功能描述.....	148
5.4 API 参考	149
5.5 数据类型.....	154
6 IMP.....	158
6.1 Sharpen.....	158
6.1.1 功能描述	158
6.1.2 API 参考	158
6.1.3 数据类型	160
6.2 Gamma	162
6.2.1 功能描述	162
6.2.2 API 参考	162
6.2.3 数据类型	166
6.3 DRC	169
6.3.1 功能描述	169
6.3.2 API 参考	169
6.3.3 数据类型	171
6.4 镜头阴影校正.....	173
6.4.1 概述	173
6.4.2 功能描述	173
6.4.3 API 参考	174
6.4.4 数据类型	178
6.5 Defect Pixel.....	184
6.5.1 概述	184
6.5.2 功能描述	184
6.5.3 API 参考	184
6.5.4 数据类型	187
6.6 CrossTalk Removal	189
6.6.1 概述	189



6.6.2 功能描述	189
6.6.3 API 参考	190
6.6.4 数据类型	192
6.7 Denoise.....	193
6.7.1 概述	193
6.7.2 功能描述	193
6.7.3 API 参考	194
6.7.4 数据类型	196
6.8 DIS	197
6.8.1 概述	197
6.8.2 功能描述	197
6.8.3 API 参考	198
6.8.4 数据类型	203
6.9 去雾.....	205
6.9.1 功能描述	205
6.9.2 API 参考	205
6.9.3 数据类型	207
6.10 去伪彩.....	208
6.10.1 概述	208
6.10.2 功能描述	208
6.10.3 API 参考	208
6.10.4 数据类型	210
6.11 去马赛克.....	211
6.11.1 功能描述.....	211
6.11.2 API 参考	211
6.11.3 数据类型.....	213
6.12 黑电平.....	216
6.12.1 功能描述	216
6.12.2 API 参考	216
6.12.3 数据类型	218
6.13 获取 ISP 模块虚拟地址	219
6.13.1 功能描述	219
6.13.2 API 参考	219
6.13.3 数据类型	220
7 Debug.....	222
7.1 概述.....	222
7.2 功能描述.....	222
7.3 API 参考	222
7.4 数据类型.....	225



8 AF	228
8.1 概述.....	228
8.2 功能描述.....	228
8.3 API 参考	228
8.4 数据类型.....	230
9 错误码.....	232
10 Proc 调试信息说明.....	233
10.1 概述.....	233
10.2 ISP.....	233



插图目录

图 1-1 ISP 控制结构示意图.....	1
图 1-2 ISP firmware 设计思路	2
图 1-3 ISP firmware 文件组织	3
图 1-4 ISP firmware 内部流程	4
图 1-5 ISP firmware 软件结构	4
图 1-6 ISP firmware 使用流程	5
图 2-1 ISP 库 与 sensor 库间的接口.....	24
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	27
图 2-4 ISP 库 与 AF 库间的接口.....	28
图 3-1 AE 模块工作流程图	60
图 3-2 AE 五段统计信息直方图.....	61
图 3-3 AE 256 段统计信息直方图.....	62
图 3-4 AE 工作原理图	62
图 3-5 AE 库 与 sensor 库间的接口	65
图 3-6 AE 分配路线示意图	72
图 4-1 AWB 工作原理图.....	113
图 4-2 AWB 库 与 sensor 库间的接口.....	115
图 5-1 CCM 矩阵.....	146
图 6-1 CrossTalk Remove 门限	188
图 6-2 DIS 偏移示意图	196
图 6-3 DIS 水平偏移	202
图 6-4 DIS 垂直偏移	203



1 概述

1.1 概述

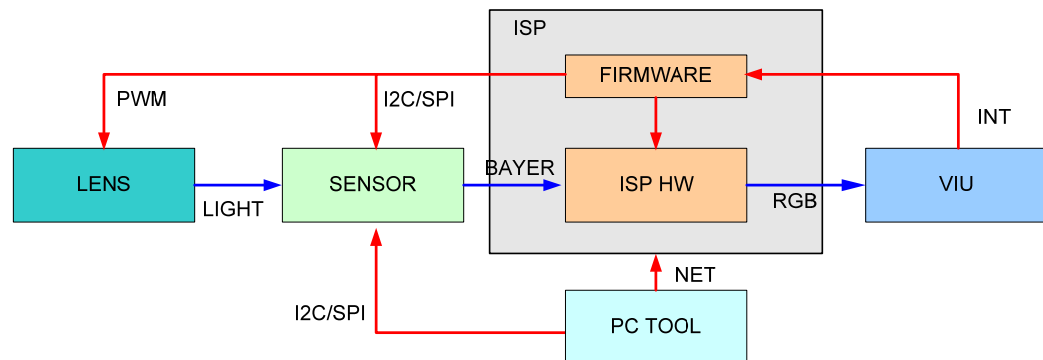
ISP 通过一系列数字图像处理算法完成对数字图像的效果处理。主要包括 3A、坏点校正、去噪、强光抑制、背光补偿、色彩增强、镜头阴影校正等处理。ISP 包括逻辑部分以及运行在其上的 firmware。这里主要介绍 ISP 的用户接口。

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 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。PC TOOL 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

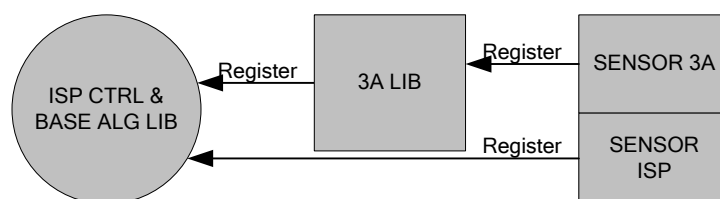


ISP 逻辑主要流程、具体概念和功能点请参见芯片手册。

1.2.2 设计思路

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 设计思路如图 1-2 所示。

图1-2 ISP firmware 设计思路

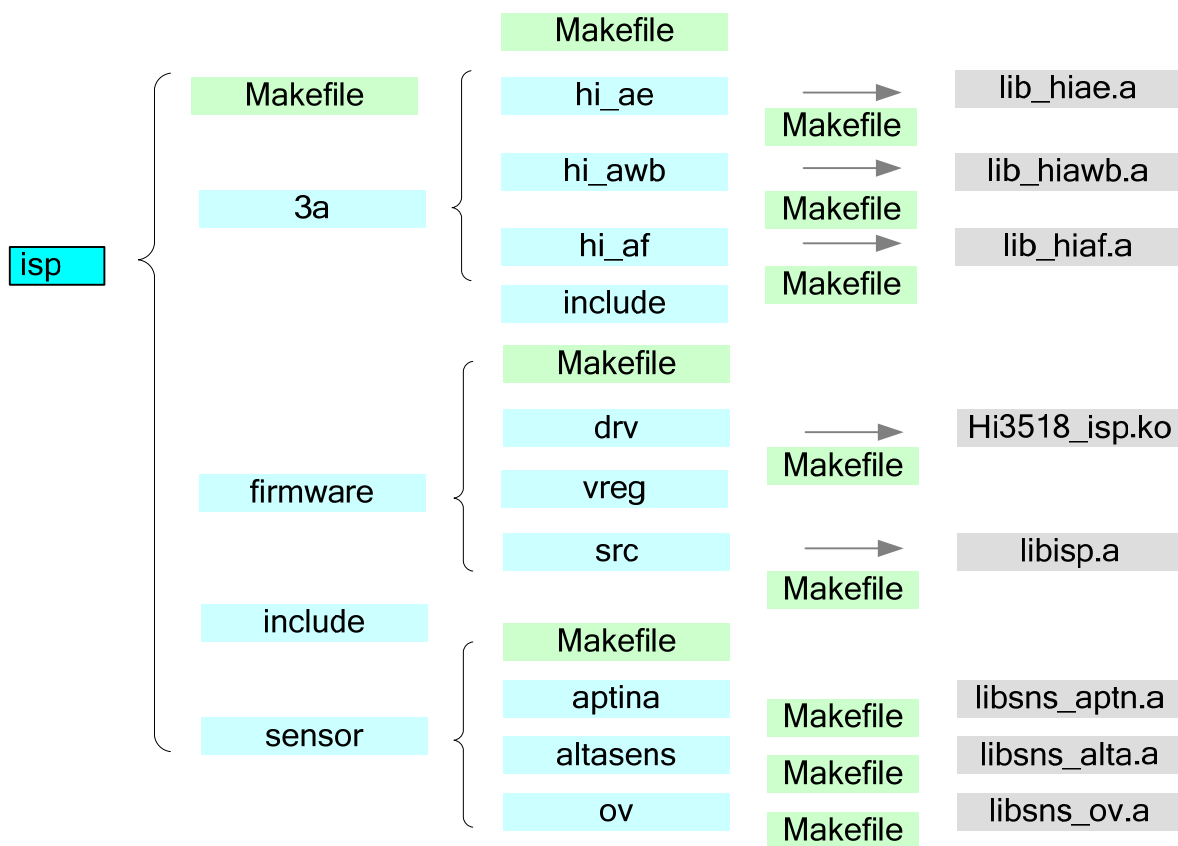


不同的 sensor 都向 ISP 的基础算法库和 3A 算法库注册控制函数，这些函数都以回调函数的形式存在。ISP 控制单元调度基础算法库和 3A 算法库时，将通过这些回调函数获取初始化参数，并控制 sensor，如调节曝光时间、模拟增益、数字增益，控制 lens 步进聚焦或旋转光圈等。

1.2.3 文件组织

ISP Firmware 的文件组织结构如图 1-3 所示，ISP 库和 3A 库、sensor 库分开。Firmware 中的 drv 生成的驱动程序上报 ISP 中断，并以该中断驱动 Firmware 的 ISP 控制单元运转，ISP 控制单元将从驱动程序中获取统计信息，并调度基础算法库和 3A 算法库，最后将需要配置的寄存器信息告知驱动程序。Src 文件夹中包含 ISP 控制单元和基础算法库，编译后生成 libisp.a，即 ISP 库。3a 文件夹中包含 AE/AWB/AF 算法库，用户也可以基于统一的接口界面开发自己的 3a 算法。Sensor 文件夹中包含了各个 sensor 的驱动程序，该部分代码开源，这里将其编译成库的形式，方便应用程序编译和连接，当然，用户可以根据自己的需要多样化处理。

图1-3 ISP firmware 文件组织



1.2.4 开发模式

SDK 中给出的形式支持用户的多种开发模式，用户可以使用海思的 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。Sensor 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。

用户也可以根据 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 se 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。

用户还可以部分使用海思 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。



说明

高级用户可以基于 ISP 寄存器进行自己的算法库开发，当然这需要对 ISP 逻辑比较熟悉，同时具有算法开发能力。



1.2.5 内部流程

Firmware 内部流程分两部分，如图 1-4 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 的软件结构如如图 1-5 所示。

图1-4 ISP firmware 内部流程

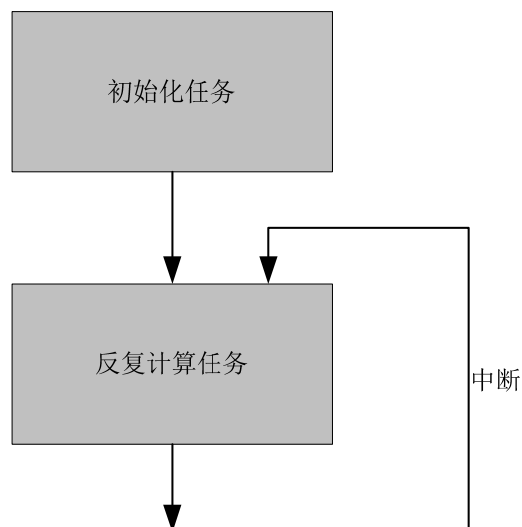
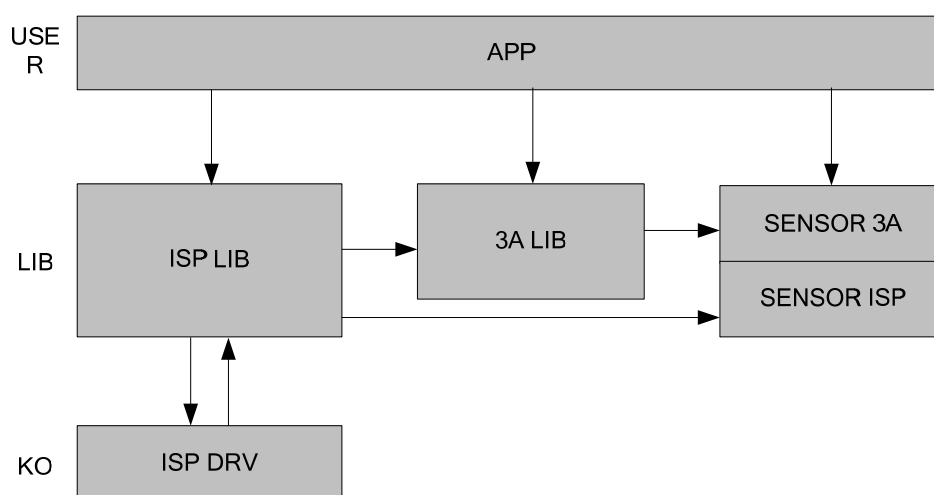


图1-5 ISP firmware 软件结构



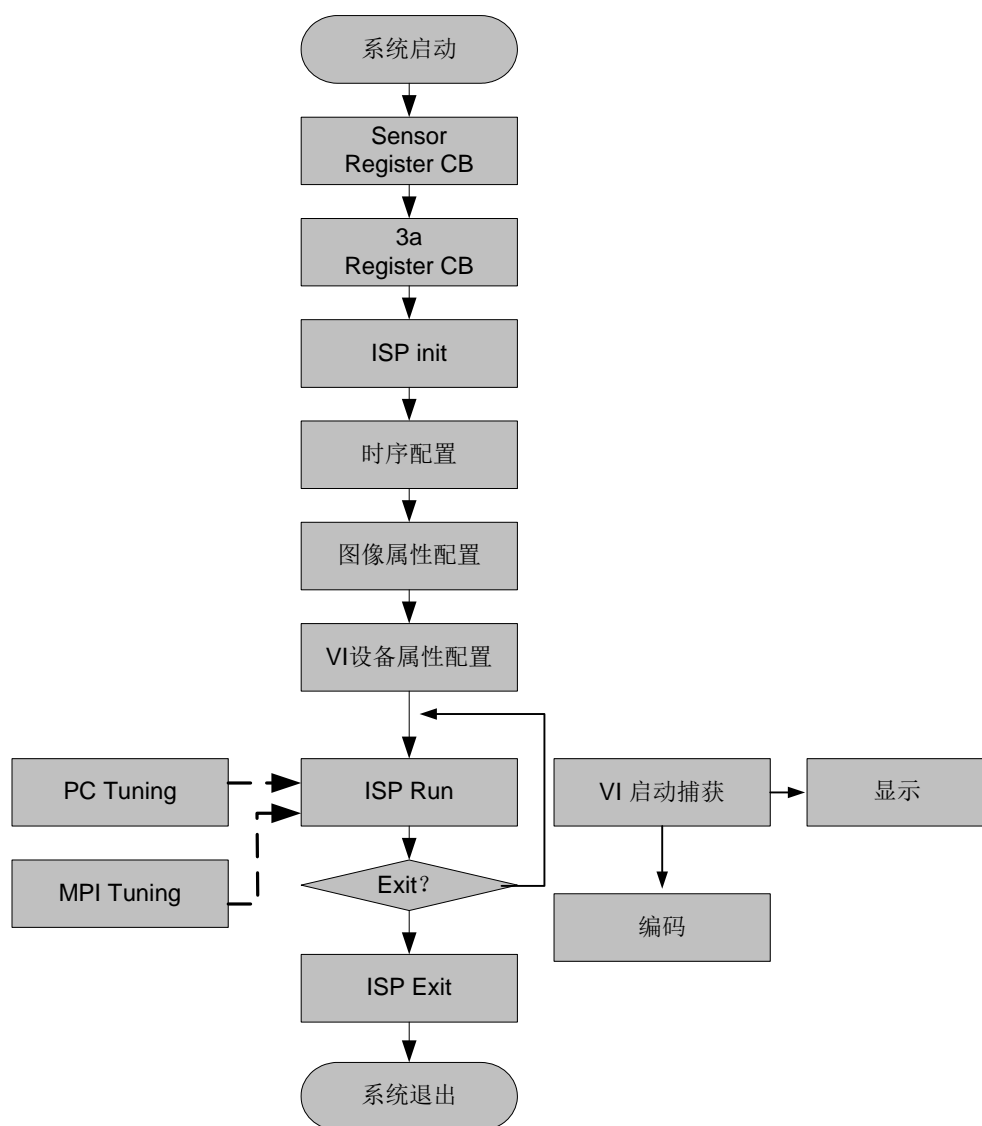


1.2.6 软件流程

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 采集到 DDR，进而送去显示或编码。软件使用流程如图 1-6 所示。

PC Tuning 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。如果在产品发布阶段没有 PC Tuning 工具，可以使用 MPI 中提供的图像质量调节接口进行简单的图像效果调试。

图1-6 ISP firmware 使用流程



如果用户调试好图像效果后，可以使用 PC Tuning 工具提供的保存配置文件进行配置参数保存，在下次启动时可以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 初始化时序配置, ISP 图像属性, 初始化 ISP Firmware, 运行 ISP firmware, 设置 ISP 各模块等功能。

2.2 API 参考

- [HI_MPI_ISP_SetInputTiming](#): 设置 ISP 输入时序。
- [HI_MPI_ISP_GetInputTiming](#): 获取 ISP 输入时序。
- [HI_MPI_ISP_SetImageAttr](#): 设置输入图像属性。
- [HI_MPI_ISP_GetImageAttr](#): 获取输入图像属性。
- [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_FreezeFmw](#): ISP firmware 冻结控制。
- [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_SetSlowFrameRate](#): 设置 ISP 降低帧率的倍数。
- [HI_MPI_ISP_GetSlowFrameRate](#): 获取 ISP 降低帧率的倍数值。
- [HI_MPI_ISP_SetAntiFlickerAttr](#): 设置 ISP 抗闪频率属性。
- [HI_MPI_ISP_GetAntiFlickerAttr](#): 获取 ISP 抗闪频率属性。
- [HI_MPI_ISP_GetVDTimeOut](#): 获取 ISP 中断信息。
- [HI_MPI_ISP_SetWdrAttr](#): 设置 ISP 宽动态属性。
- [HI_MPI_ISP_GetWdrAttr](#): 获取 ISP 宽动态属性。
- [HI_MPI_ISP_SensorRegCallBack](#): ISP 提供的 sensor 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_AwbLibRegCallBack](#): ISP 提供的 AWB 库注册的回调接口。



- [HI_MPI_ISP_AfLibRegCallBack](#): ISP 提供的 AF 库注册的回调接口。
- [HI_MPI_ISP_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_VRegInit](#): 初始化一段外部寄存器。
- [HI_MPI_ISP_VRegExit](#): 销毁一段外部寄存器。
- [HI_MPI_ISP_GetVRegAddr](#): 获取一段外部寄存器的物理地址。

HI_MPI_ISP_SetInputTiming

【描述】

设置 ISP 输入时序。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstInputTiming	输入时序属性。 静态属性。	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	参数无效。

【需求】

- 头文件: [hi_comm_isp.h](#)、[mpi_isp.h](#)
- 库文件: [libisp.a](#)

【注意】



- 设置前需要先初始化 ISP。
- 该接口从 sensor 输入的图像中获取 ISP 需要的有效图像内容。尤其是某些 sensor 有 OB 区输出，即图像输出有黑边，此时需要用该接口剪裁掉黑边。如果 sensor 输入的图像就是 ISP 需要的有效内容，接口模式中选择 ISP_WIND_NONE，此时剪裁区的配置无效。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetInputTiming](#)

HI_MPI_ISP_GetInputTiming

【描述】

获取 ISP 输入时序。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstInputTiming	输入时序属性。 静态属性。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a



【注意】

获取前需要先设置。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetInputTiming](#)

HI_MPI_ISP_SetImageAttr

【描述】

设置输入图像属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstImageAttr	输入图像属性。 静态属性。	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	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。
- 设置前需要先初始化 ISP。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetImageAttr](#)

HI_MPI_ISP_GetImageAttr

【描述】

获取输入图像属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetImageAttr( ISP_IMAGE_ATTR_S *pstImageAttr );
```

【参数】

参数名称	描述	输入/输出
pstImageAttr	输入图像属性。 静态属性。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

获取前需要先设置输入图像属性。



【举例】

无。

【相关主题】

[HI_MPI_ISP_SetImageAttr](#)

HI_MPI_ISP_Init

【描述】

初始化 ISP firmware。

【语法】

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

【参数】

无

【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_SNS_UNREGISTER	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 sensor 已经初始化，并且注册了回调函数。

【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(HI_VOID);
```

【参数】

无。

【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_SNS_UNREGISTER	Sensor 未注册。

【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，并且注册了回调函数。
- 运行前需要确保时序和图像属性已配置。
- 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(HI_VOID);
```

【参数】

无。

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_Init](#)

HI_MPI_ISP_FreezeFmw

【描述】

ISP firmware 冻结控制。

【语法】

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

【参数】

参数名称	描述	输入/输出
bFreeze	ISP firmware 冻结使能。	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】

无。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

当 bFreeze 值为 True 时 ISP Firmware 的 3A 算法，DRC 算法，NR 算法等都停止 ISP，Sensor 的寄存器将一直保持冻结前的值，直到再次调用此接口使 bFreeze 值为 False。

【举例】

无。

【相关主题】

无。

HI_MPI_ISP_SetModuleControl

【描述】

设定 ISP 功能模块的控制。

【语法】

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

【参数】

参数名称	描述	输入/输出
u32ModFlag	模块控制值。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。



【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。
- u32ModFlag 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。

Bit	描述
[31:27]	保留。
[26]	输出直接与输入相连。
[25:24]	00：全部处理； 01：旁路所有 ISP 处理(视频输入端口仍与输出端口连接)，输出感光器原始数据； 10：旁路所有 ISP 处理(视频输入端口仍与输出端口连接)，输出通道 1 和通道 2 最高位的感光器原始数据； 11：将输出端与地相连。
[23:15]	保留。
[14]	旁路 gamma 表。
[13]	旁路颜色矩阵。
[12]	旁路去马赛克模块(输出原始数据)。
[11]	旁路 DRC。
[10]	保留。
[9]	旁路阴影校正。
[8:7]	保留。
[6]	旁路黑电平和增益。
[5]	旁路去噪。
[4]	旁路坏点校正。
[3]	旁路绿平衡。
[2]	旁路 WDR 压缩前端查找。
[1]	旁路前端黑电平调节。



Bit	描述
[0]	旁路视频测试生成器。

【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_MPI_ISP_GetModuleControl(HI_U32 *pu32ModFlag);
```

【参数】

参数名称	描述	输入/输出
pu32ModFlag	模块控制值。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_SetModuleControl](#)

HI_MPI_ISP_SetSlowFrameRate

【描述】

设置 ISP 降低帧率的倍数。

【语法】

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

【参数】

参数名称	描述	输入/输出
u8Value	降低帧率的参数值	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_ILLEGAL_PARAM	参数错误。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

当前帧率=原帧率/(u8Value >> 4)。u8Value 允许的最小值为 0x10，表示当前的帧率和原帧率相等；当 u8Value 设置为 0x20，表示当前帧率为原帧率的 1/2；当 u8Value 设置为 0x30，表示当前帧率为原帧率的 1/3，以此类推。



【举例】

无。

【相关主题】

[HI_MPI_ISP_GetSlowFrameRate](#)

HI_MPI_ISP_GetSlowFrameRate

【描述】

获取 ISP 降低帧率的倍数值。

【语法】

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

【参数】

参数名称	描述	输入/输出
pu8Value	降低帧率参数值	输出

【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。

【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_SetSlowFrameRate](#)

HI_MPI_ISP_SetAntiFlickerAttr

【描述】

设置 ISP 抗闪频率属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAntiflicker	设置抗闪频率属性	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_ILLEGAL_PARAM	参数错误。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

当电源频率为 50Hz 时，抗闪频率值设置为 50；当电源频率为 60Hz 时，抗闪频率值设置为 60。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetAntiFlickerAttr](#)



HI_MPI_ISP_GetAntiFlickerAttr

【描述】

获取 ISP 抗闪频率属性。

【语法】

```
HI_MPI_ISP_GetAntiFlickerAttr(ISP_ANTIFLICKER_S *pstAntiflicker);
```

【参数】

参数名称	描述	输入/输出
pstAntiflicker	频率抗闪属性	输出

【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。

【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_SetAntiFlickerAttr](#)

HI_MPI_ISP_GetVDTimeOut

【描述】



获取 ISP 中断信息。

【语法】

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

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

【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】

- 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_SetWdrAttr

【描述】

设置 ISP 宽动态属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstWdrAttr	设置宽动态属性	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】

无。

【举例】

无。

【相关主题】

HI_MPI_ISP_GetWdrAttr

HI_MPI_ISP_GetWdrAttr

【描述】

获取 ISP 宽动态属性。

【语法】

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

【参数】



参数名称	描述	输入/输出
pstWdrAttr	获取宽动态属性	输出

【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】

无。

【举例】

无。

【相关主题】

HI_MPI_ISP_SetWdrAttr

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

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

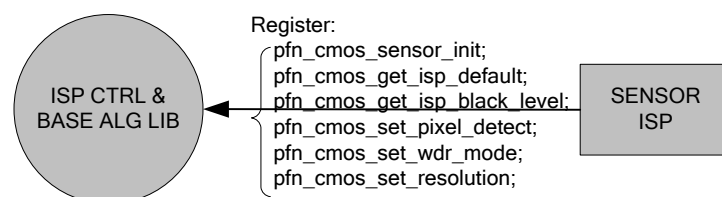
【需求】

- 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_SENSOR_REGISTER_S stIsRegister;
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIsRegister.stSnsExp;

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

【相关主题】



无。HI_MPI_ISP_GetExposureType

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

【参数】

参数名称	描述	输入/输出
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。

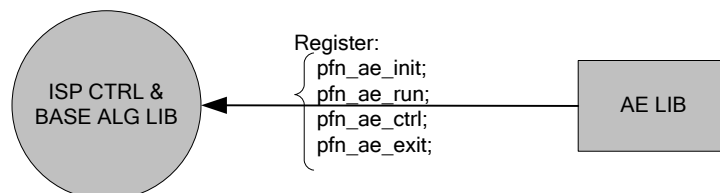
【需求】

- 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。

图2-2 ISP 库 与 AE 库间的接口



【举例】

```
ISP_AE_REGISTER_S stRegister;
```



```
HI_S32 s32Ret = HI_SUCCESS;

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

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

【相关主题】

无。 [HI_MPI_ISP_GetExposureType](#)

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

【参数】

参数名称	描述	输入/输出
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。



返回值	描述
非 0	失败，其值为错误码。

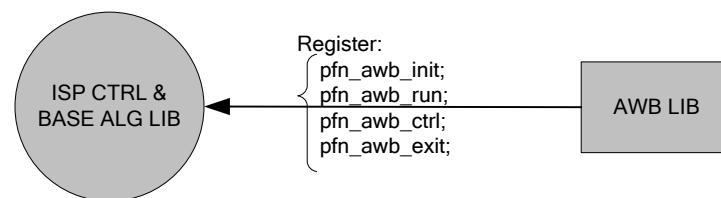
【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- ISP 提供统一的 AWB 算法库接口，初始化、运行、控制、销毁 AWB 算法库。

图2-3 ISP 库 与 AWB 库间的接口



【举例】

无。

【相关主题】

无。

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

【参数】

参数名称	描述	输入/输出
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。

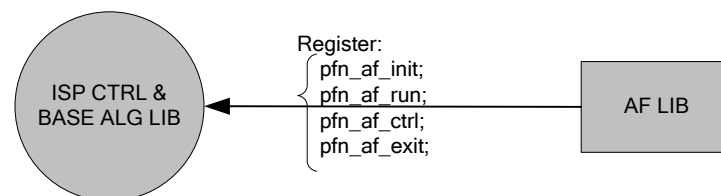
【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- ISP 提供统一的 AF 算法库接口，初始化、运行、控制、销毁 AF 算法库。

图2-4 ISP 库 与 AF 库间的接口



【举例】

无。

【相关主题】

无。

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(const ISP_BIND_ATTR_S *pstBindAttr);
```

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

【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】

无。

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_BIND_ATTR_S *pstBindAttr);
```

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

【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_VRegInit

【描述】

初始化一段外部寄存器。

【语法】

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

【参数】

参数名称	描述	输入/输出
u32BaseAddr	外部寄存器的基地址。	输入
u32Size	外部寄存器的大小。	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】

- 在 ISP 库、算法库中均用到外部寄存器，主要作用是允许多进程环境下控制 ISP 库和算法库的内部状态，实际上相当于共享内存的机制。
- 外部寄存器的机制是定义一段虚拟的寄存器地址（u32BaseAddr ~ u32BaseAddr+ u32Size），并分配一段相同大小的物理地址。无论是哪个进程，只



要虚拟的寄存器地址相同，都能实际地读写相应的物理地址中的值。在 ISP Firmware 文件组织的 isp/firmware/vreg 目录下，调用了 HI_MPI_VRegInit、HI_MPI_VRegExit、HI_MPI_GetVRegAddr 等接口，对外部寄存器进行了封装，规定：

0~0x10000 段地址空间对应 ISP 实际的硬件寄存器；

0x10000~0x20000 段地址空间对应 ISP 的外部寄存器；

0x20000~0x30000 段地址空间对应 AE 的外部寄存器，最多 16 个 AE 库；

0x30000~0x40000 段地址空间对应 AWB 的外部寄存器，最多 16 个 AWB 库；

0x40000~0x50000 段地址空间对应 AF 的外部寄存器，最多 16 个 AF 库。

每段外部寄存器均需要由 ISP 库或算法库调用封装后的接口单独创建和初始化。用户也可以自己实现多进程控制的机制，或者自定义外部寄存器的形式。

- 此函数将分配一段物理地址，并建立和记录外部寄存器地址与实际物理地址的对应关系，0~0x10000 段地址空间默认对应实际硬件寄存器的物理地址。

【举例】

无。

【相关主题】

无。

HI_MPI_ISP_VRegExit

【描述】

销毁一段外部寄存器。

【语法】

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

【参数】

参数名称	描述	输入/输出
u32BaseAddr	外部寄存器的基地址。	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】

此函数将销毁一段物理地址，并删除外部寄存器地址与实际物理地址的对应关系的记录。

【举例】

无。

【相关主题】

无。

HI_MPI_ISP_GetVRegAddr

【描述】

获取一段外部寄存器的物理地址。

【语法】

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

【参数】

参数名称	描述	输入/输出
u32BaseAddr	外部寄存器的基地址。	输入
pu32PhyAddr	外部寄存器的物理地址指针。	输出

【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】

- 此函数将根据外部寄存器的地址获取实际物理地址。
- 地址均是不带偏移的基地址。

【举例】



无。

【相关主题】

无。

2.3 数据类型

- [ISP_WIND_MODE_E](#): 定义 ISP 输入时序窗类型。
- [ISP_INPUT_TIMING_S](#): 定义 ISP 输入时序窗口属性。
- [ISP_BAYER_FORMAT_E](#): 定义输入 Bayer 图像数据格式。
- [ISP_IMAGE_ATTR_S](#): 定义 ISP 输入图像属性。
- [ISP_ANTIFLICKER_S](#): 定义 ISP 图像抗闪属性。
- [ISP_VD_INFO_S](#): 定义 ISP 帧同步信息结构体。
- [ISP_WDR_ATTR_S](#): 定义 ISP 宽动态属性。
- [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_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。
- [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_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_WIND_MODE_E

【说明】



定义 ISP 输入时序窗类型。

【定义】

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

} ISP_WIND_MODE_E;
```

【成员】

成员名称	描述
ISP_WIND_NONE	无剪裁窗。
ISP_WIND_HOR	水平剪裁。
ISP_WIND_VER	垂直剪裁。
ISP_WIND_ALL	水平、垂直都剪裁。

【注意事项】

无。

【相关数据类型及接口】

[ISP_INPUT_TIMING_S](#)

ISP_INPUT_TIMING_S

【说明】

定义 ISP 输入时序窗口。

【定义】

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

} ISP_INPUT_TIMING_S;
```



【成员】

成员名称	描述
enWndMode	剪裁窗口模式。
u16HorWndStart	水平起始位置，取值范围为[0x0, 0x780]。
u16HorWndLength	水平窗口长度，取值范围为[0x0, 0x780]。
u16VerWndStart	垂直起始位置，取值范围为[0x0, 0x4B0]。
u16VerWndLength	垂直窗口高度，取值范围为[0x0, 0x4B0]。

【注意事项】

对有 OB（光学黑区）输出的 sensor 图像，使用该接口进行有效区图像剪裁。

【相关数据类型及接口】

[ISP_WIND_MODE_E](#)

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。

【相关数据类型及接口】

[ISP_IMAGE_ATTR_S](#)

ISP_IMAGE_ATTR_S

【说明】

定义 ISP 输入图像属性。

【定义】

```
typedef struct hiISP_IMAGE_ATTR_S
{
    HI_U16  u16Width;
    HI_U16  u16Height;
    HI_U16  u16FrameRate;
    ISP_BAYER_FORMAT_E  enBayer;
} ISP_IMAGE_ATTR_S;
```

【成员】

成员名称	描述
u16Width	输入图像宽度，取值范围为[0x0, 0x780]。
u16Height	输入图像高度，取值范围为[0x0, 0x4B0]。
u16FrameRate	输入图像帧率，取值范围为[0x0,0xFF]。
enBayer	Bayer 数据格式。

【注意事项】

无。

【相关数据类型及接口】

[ISP_BAYER_FORMAT_E](#)

ISP_ANTIFLICKER_MODE_E

【说明】

定义 ISP 抗闪模式。

【定义】

```
typedef enum hiISP_ANTIFLICKER_MODE_E
```



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

【成员】

成员名称	描述
ISP_ANTIFLICKER_MODE_0	抗闪模式 0。
ISP_ANTIFLICKER_MODE_1	抗闪模式 1。

【注意事项】

- ISP_ANTIFLICKER_MODE_0 为抗闪模式 0，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)。
 - 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。而抗闪模式 0 的最小曝光时间不能匹配光源频率，产生过曝。
- ISP_ANTIFLICKER_MODE_1 为抗闪模式 1，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。与抗闪模式 0 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 ISP 图像抗闪属性。

【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。
enMode	抗闪模式

【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_WDR_ATTR_S

【说明】

定义 ISP 宽动态属性。

【定义】



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

【成员】

成员名称		描述
enWdrMode	ISP_SENSOR_LINEAR_MODE	Sensor 为线性模式。
	ISP_SENSOR_WDR_MODE	Sensor 为宽动态模式。

【注意事项】

无。

【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_U32(*pfn_cmos_get_isp_default)(ISP_CMOS_DEFAULT_S *pstDef);

    HI_U32(*pfn_cmos_get_isp_black_level)(ISP_CMOS_BLACK_LEVEL_S
    *pstBlackLevel);

    HI_VOID(*pfn_cmos_set_pixel_detect)(HI_BOOL bEnable);

    HI_VOID(*pfn_cmos_set_wdr_mode)(HI_U8 u8Mode);

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

【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_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。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_set_wdr_mode	设置 wdr 模式和线性模式切换的回调函数指针。
pfn_cmos_set_resolution	设置分辨率切换的回调函数指针。

【注意事项】

- 提供 pfn_cmos_get_isp_black_level 的回调函数的原因是部分 sensor 的黑电平并不是一个固定值，而是动态地根据 sensor 的增益变化。
- 如果回调函数指针不需要赋值，需要置为 NULL。例如有的 sensor 不支持切换分辨率，那么 pfn_cmos_set_resolution 需要置为 NULL。

【相关数据类型及接口】

ISP_SENSOR_REGISTER_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_COMM_S      stComm;

    ISP_CMOS_DENOISE_S    stDenoise;

    ISP_CMOS_DRC_S        stDrc;

    ISP_CMOS_AGC_TABLE_S  stAgcTbl;

    ISP_CMOS_NOISE_TABLE_S stNoiseTbl;

    ISP_CMOS_DEMOSAIC_S   stDemosaic;

    ISP_CMOS_GAMMAFE_S    stGammafe;

    ISP_CMOS_SHADING_S    stShading;
} ISP_CMOS_DEFAULT_S;
```

【成员】

成员名称	成员名称	描述
stComm	u8Rggb	sensor 输出 RGrGbB 的顺序，取值范围为 [0,3]。
	u8BalanceFe	不建议修改，推荐使用默认值 0x1。
stDenoise	u8SinterThresh	不建议修改，推荐使用默认值 0x15。
	u8NoiseProfile	噪声型式，设置为 0 表示 ISP 默认。建议使用默认值 0。
	u16Nr0	不建议修改，推荐使用默认值 0x0。
	u16Nr1	不建议修改，推荐使用默认值 0x0。
stDrc	u8DrcBlack	不建议修改，推荐使用默认值 0x0。
	u8DrcVs	不建议修改，推荐使用默认值。线性 0x04，WDR 0x08。
	u8DrcVi	不建议修改，推荐使用默认值。线性 0x08，WDR 0x01。
	u8DrcSm	不建议修改，推荐使用默认值。线性 0xa0，WDR 0x3c。
	u16DrcWl	不建议修改，推荐使用默认值。线性 0x4ff，WDR 0xffff。
stAgcTbl	bValid	该结构体的数据是否有效，取值范围为 [0,1]。



成员名称	成员名称	描述
	au8SharpenAltD	根据增益动态调节图像大边缘锐度的插值数组，取值范围为[0,255]。
	au8SharpenAltUd	根据增益动态调节图像小纹理锐度的插值数组，取值范围为[0,255]。
	au8SnrThresh	根据增益动态设置图像去噪强度的插值数组，取值范围为[0,255]。
	au8DemosaicLumThresh	该数组用来设置图像的大边缘锐度的亮度门限值，建议用户使用默认参数值，取值范围为[0,255]。
	au8DemosaicNpOffset	该数组用来设置图像的噪声参数，建议用户使用默认参数值，取值范围为[0,255]。
	au8GeStrength	该数组用来设置绿色平衡模块的参数，建议用户使用默认参数值，取值范围为[0,255]。
stNoiseTbl	bValid	该结构体的数据是否有效，取值范围为[0,1]。
	au8NoiseProfileWeightLut	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，建议用户使用默认参数值，取值范围为[0,255]。
	au8DemosaicWeightLut	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，建议用户使用默认参数值，取值范围为[0,255]。
stDemosaic	bValid	该结构体的数据是否有效，取值范围为[0,1]。
	u8VhSlope	垂直/水平混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8AaSlope	角度混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8VaSlope	VH-AA 混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8UuSlope	未定义的混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8SatSlope	饱和度混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8AcSlope	高频分量滤波斜率门限，推荐使用默认值，取值范围为[0,255]。



成员名称	成员名称	描述
	u16VhThresh	垂直/水平混合的门限，推荐使用默认值，取值范围为[0,0xFFFF]。
	u16AaThresh	角度混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16VaThresh	VA 混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16UuThresh	未定义的混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16SatThresh	饱和度混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16AcThresh	高频分量滤波门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
stGammafe	bValid	该结构体的数据是否有效，取值范围为[0,1]。通常在 sensor 支持 wdr 模式时需要配置。
	au16Gammafe	GammaFe 表，取值范围为[0,0xFFFF]。
stShading	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果不需要 shading 校正，可以配置为无效。
	u16RCenterX	R 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16RCenterY	R 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterX	G 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterY	G 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16BCenterX	B 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16BCenterY	B 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	au16RShadingTbl	R 分量的校正表。 取值范围为[0x0, 0xFFFF]
	au16GShadingTbl	G 分量的校正表。



成员名称	成员名称	描述
		取值范围为[0x0, 0xFFFF]
	au16BShadingTbl	B 分量的校正表。 取值范围为[0x0, 0xFFFF]
	u16ROffCenter	R 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16GOffCenter	G 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16BOffCenter	B 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16TblNodeNum	每个分量校正表中使用到的节点个数。 取值范围为[0x0, 0x81]，默认值为 0x81。

【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_U8   au8BlackLevel[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]。
au8BlackLevel	Sensor 的黑电平数组，取值范围[0,255]。

【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

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_AE_REGISTER_S

【说明】

定义 AE 注册结构体。

【定义】

```
typedef struct hiISP_AE_REGISTER_S
{
    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】

AE_EXP_FUNC_S

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。
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_U32 u32MaxIspDgain;

    HI_U32 u32MinIspDgain;

    HI_U32 u32IspDgainShift;
} 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。
u32MaxIspDgain	ISP 能提供的最大的数字增益，精度为 u32IspDgainShift。
u32MinIspDgain	ISP 能提供的最小的数字增益，精度为 u32IspDgainShift。
u32IspDgainShift	ISP 数字增益的精度。

【注意事项】

- 精度的意义为 $1 \ll u32IspDgainShift$ 。例如，当 $u32MaxIspDgain=512$ ， $u32IspDgainShift=4$ ，精度为 $1 \ll 4=16$ ，那么 ISP 能提供的最大数字增益为 $512/16$ 倍，即最大支持 32 倍数字增益。

【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_INFO_S;
```

【成员】

成员名称	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]。
	au16MeteringHist	五段直方图的统计信息数组，取值范围为[0, 0xFFFF]。
pstAeStat2	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au16MeteringMemArray	五段直方图的分区间的统计信息数组，取值范围为[0, 0xFFFF]。
pstAeStat3	au16HistogramMemArray	256 段直方图的统计信息数组，取值范围为[0, 0xFFFF]。

【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。
- 五段直方图和 256 段直方图的意义请参考 AE 章节的描述。
- 统计信息分三种形式提供，第一种是全局的五段直方图；第二种是分 15x17 个区间的五段直方图，每个区间单独统计；第三种是 256 段直方图。结构体指针不为空时表示该统计信息有效，后期将提供统计信息形式可配置的接口，敬请关注。
- 统计信息都是归一化为 0xFFFF 的。

【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  u32IspDgain;

    HI_U32  u32IspDgainShift;

    HI_U32  u32Iso;

    ISP_AE_STAT_ATTR_S stStatAttr;
} ISP_AE_RESULT_S;
```

【成员】

成员名称	成员名称	描述
u32IspDgain	-	ISP 的数字增益。
u32IspDgainShift	-	ISP 的数字增益的精度。
u32Iso	-	AE 计算得出的总增益值。
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]。
	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。

【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
```



```
{  
    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】

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。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。

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_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_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。
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_INFO_S;
```

【成员】

成员名称	成员名称	描述
u32FrameCnt		帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat1	u16MeteringAwbRg	统计信息中白点的 R 和 G 的平均值的比值，取值范围为[0, 0xFFFF]。
	u16MeteringAwbBg	统计信息中白点的 B 和 G 的平均值的比值，取值范围为[0, 0xFFFF]。
	u32MeteringAwbSum	统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat2	au16MeteringMemArrayRg	分区间的统计信息中白点的 R 和 G 的平均值的比值，取值范围为[0, 0xFFFF]。
	au16MeteringMemArrayBg	分区间的统计信息中白点的 B 和 G 的平均值的比值，取值范围为[0, 0xFFFF]。
	au16MeteringMemArraySum	分区间的统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。

【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。
- 统计信息分两种形式提供，第一种是全局的统计信息；第二种是分 15x17 个区间的统计信息，每个区间单独统计。

【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_RESULT_S;
```

【成员】

成员名称	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。
	u16MeteringWhiteLevelAwb	统计白点信息时，找白点的上限。默认值 0x3ac。
	u16MeteringBlackLevelAwb	统计白点信息时，找白点的下限。默认值 0x40。
	u16MeteringCrRefMaxAwb	统计白点信息时，白点区域的最大 R/G 值。默认值 512。
	u16MeteringCbRefMaxAwb	统计白点信息时，白点区域的最大 B/G 值。默认值 512。



	ul6MeteringCrRefMinAwb	统计白点信息时，白点区域的最小的 R/G 值。默认值 128。
	ul6MeteringCbRefMinAwb	统计白点信息时，白点区域的最小的 B/G 值。默认值 128。

【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。
- 对于 WDR 模式，因为有 GammaFe 作用，图像被拉至非线性空间，所以配置寄存器时会对 AWB 算法的返回结果做一个开方运算，然后配置到寄存器中。但用户开发新的 AWB 算法时不必担心，开方运算会在配置寄存器时由 ISP 控制模块完成，AWB 算法只需要返回正确的四个颜色通道的 16bit 精度的增益即可。
- ISP 数字增益会叠加在四个颜色通道的增益上，ISP 控制模块配置寄存器时会完成叠加运算，AWB 算法只需要返回正确的四个颜色通道的 16bit 精度的增益即可。
- 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。

【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
{
    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】

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	描述
u32FrameCnt	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAfStat	AF 的统计信息结构体指针。



【注意事项】

- AF 库暂未实现。

【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】

- AF 库暂未实现。

【相关数据类型及接口】

ISP_AF_EXP_FUNC_S

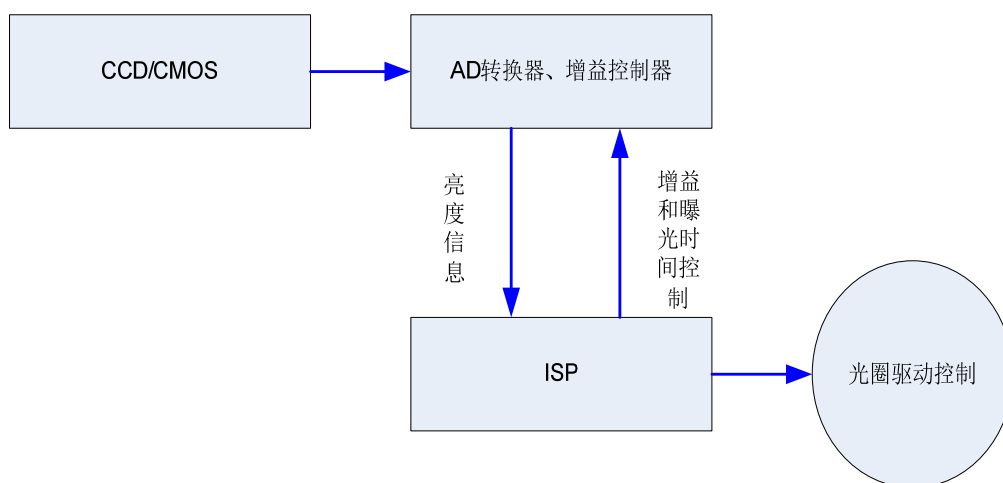


3 AE

3.1 概述

HiSP 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。快门优先时算法会优先保证曝光时间的限定,适合拍摄运动物体的场景。增益优先则是为了保证对 sensor 增益的限定,这样拍摄的图像噪声会比较小。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。

- 光圈和机械快门：光圈是镜头中可以改变中间孔大小的机械装置，机械快门是控制曝光时间长短的装置，两者结合可控制进光量。
- 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 输入数据数据的亮度信息统计。其提供的统计信息为直方图统计信息，可同时提供整幅图像的 5 段直方图统计信息和 256 段的直方图统计信息，还可提供将整幅图像分成 MxN 区块的每个区块的直方图统计信息，具体如图 3-2 和图 3-3 所示。

图3-2 AE 五段统计信息直方图

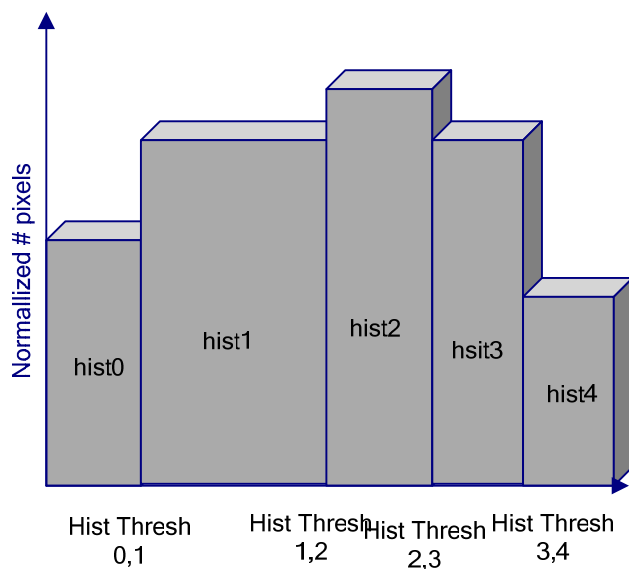
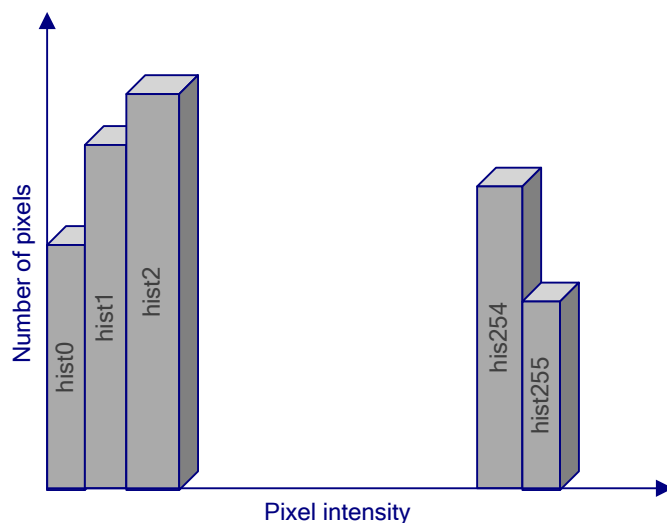
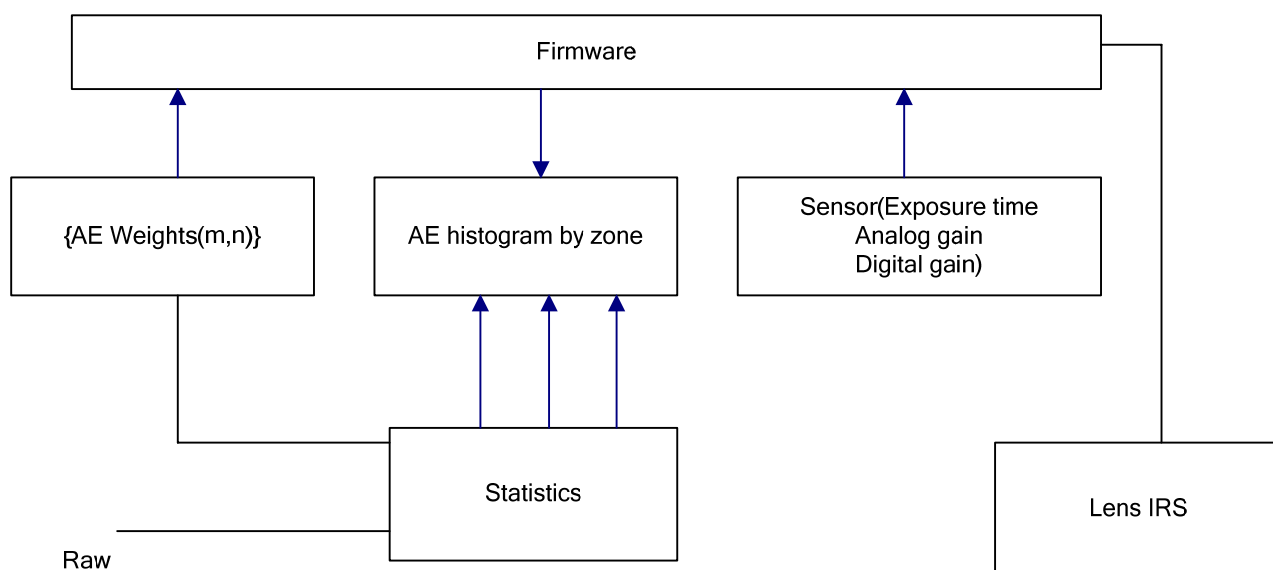


图3-3 AE 256 段统计信息直方图



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 3-4 所示。

图3-4 AE 工作原理图





3.4 API 参考

3.4.1 AE 库接口

- [HI_MPI_AE_Register](#): 向 ISP 注册 AE 库。
- [HI_MPI_AE_SensorRegCallBack](#): AE 库提供的 sensor 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(ALG_LIB_S *pstAeLib);
```

【参数】

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，其值为错误码。

【需求】

- 头文件: hi_comm_isp.h、mpi_ae.h
- 库文件: libisp.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，以实现向 ISP 库注册的功能。
- 用户调用此接口完成向 ISP 库注册。
- AE 库可以注册多个实例。

【举例】

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



```
stAeLib.s32Id = 1;  
  
HI_MPI_AE_Register(&stAeLib);
```

【相关主题】

HI_MPI_ISP_AeLibRegCallBack

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

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

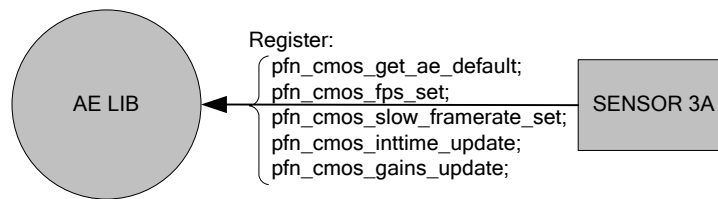
【需求】

- 头文件：hi_comm_isp.h、mpi_ae.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。
- 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3-5 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;

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

【相关主题】

无。

3.4.2 AE 控制模块

曝光控制接口：

- [HI_MPI_ISP_SetExposureType](#)：设置曝光类型。
- [HI_MPI_ISP_GetExposureType](#)：获取曝光类型。
- [HI_MPI_ISP_SetAEAttr](#)：设置 AE 属性。
- [HI_MPI_ISP_GetAEAttr](#)：获取 AE 属性。
- [HI_MPI_ISP_SetAEAttrEx](#)：设置 AE 扩展属性。
- [HI_MPI_ISP_GetAEAttrEx](#)：获取 AE 扩展属性。
- [HI_MPI_ISP_SetAERouteAttr](#)：设置 AE 曝光分配策略。



- [HI_MPI_ISP_GetAERouteAttr](#): 获取 AE 曝光分配策略。
- [HI_MPI_ISP_SetAEDelayAttr](#): 设置 AE 延时属性。
- [HI_MPI_ISP_GetAEDelayAttr](#): 获取 AE 延时属性。
- [HI_MPI_ISP_SetMEAttr](#): 设置 ME 属性。
- [HI_MPI_ISP_GetMEAttr](#): 获取 ME 属性。
- [HI_MPI_ISP_SetMEAttrEx](#): 设置 ME 扩展属性。
- [HI_MPI_ISP_GetMEAttrEx](#): 获取 ME 扩展属性。
- [HI_MPI_ISP_SetExpStaInfo](#): 设置 AE 曝光统计信息。
- [HI_MPI_ISP_GetExpStaInfo](#): 获取 AE 曝光统计信息。
- [HI_MPI_ISP_QueryInnerStateInfo](#): 获取内部状态信息。
- [HI_MPI_ISP_QueryInnerStateInfoEx](#): 获取内部状态信息。

HI_MPI_ISP_SetExposureType

【描述】

设定 AE 曝光控制类型。

【语法】

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

【参数】

参数名称	描述	输入/输出
enExpType	AE 曝光控制类型	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_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】

- 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。
- AE 曝光控制类型为手动时, 需要调用 API 接口 HI_MPI_SetMEAttr 自行设定手动曝光属性, 包括使能类型 (曝光时间使能、曝光模拟增益使能、曝光数字增益使能) 及相应的曝光参数 (曝光时间、曝光模拟增益、曝光数字增益)。

【举例】

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

【相关主题】

[HI_MPI_ISP_GetExposureType](#)

HI_MPI_ISP_GetExposureType

【描述】

获取 AE 曝光控制类型。

【语法】

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

【参数】



参数名称	描述	输入/输出
penExpType	AE 曝光控制类型。	输出

【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。

【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_SetExposureType](#)

HI_MPI_ISP_SetAEAttr

【描述】

设置 AE 曝光属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEAttr	AE 曝光属性。	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	参数无效。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 此接口用于设定曝光时间，数字增益，模拟增益的最大最小值。
- 曝光调整的步长属性用于调节曝光收敛的速度，步长值越大曝光的收敛速度会越快但也会导致收敛过程中出现反复震荡。
- 曝光容忍偏差属性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会越不明感且会导致同一目标亮度值多次调节得到的亮度偏差会越大，所以该属性推荐不能设定过大。
- 曝光的亮度补偿属性用于调节曝光的亮度。曝光亮度补偿值越大则图像亮度越高。
- 曝光权重表属性用于对调节感兴趣区域的曝光权重。

【举例】

```
{  
    ISP\_AE\_ATTR\_S stAEAttr;  
    HI_MPI_ISP_GetAEAttr(&stAEAttr);  
  
    stAEAttr.ul6ExpTimeMax = 4000;  
    stAEAttr.ul6ExpTimeMin = 2;  
    stAEAttr.ul6AGainMax   = 28;  
    stAEAttr.ul6AGainMin   = 0;  
    stAEAttr.ul6DGainMax   = 38;  
    stAEAttr.ul6DGainMin   = 0;  
    stAEAttr.u8ExpStep     = 8;  
}
```



```
stAEAttr.s16ExpTolerance = 4;  
stAEAttr.u8ExpCompensation = 0x20;
```

```
HI_MPI_ISP_SetAEAttr(&stAEAttr);  
}
```

【相关主题】

[HI_MPI_ISP_GetExposureType](#)

HI_MPI_ISP_GetAEAttr

【描述】

获取 AE 曝光属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEAttr	AE 曝光属性。	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。

【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_GetExposureType](#)

HI_MPI_ISP_SetAERouteAttr

【描述】

设置 AE 曝光分配策略。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAERouteAttr	AE 曝光分配策略。	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	参数无效。

【需求】

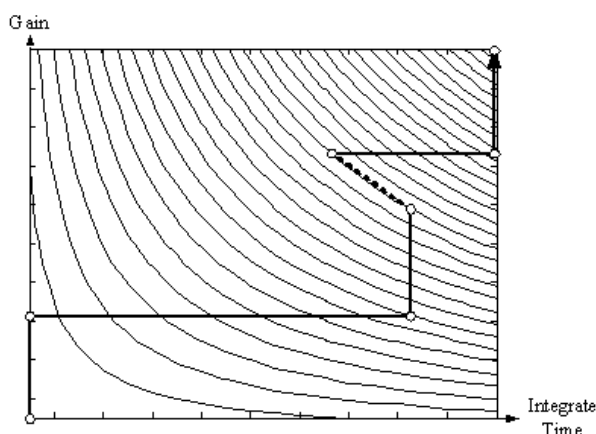
- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【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 3-6](#)所示。AE 分配路线遵循以下限定：
 - 最大支持八个节点，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。
- 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。
- 默认的 AE 分配策略是首先分配曝光时间，其次分配增益。如果当前的曝光量不在用户设定的路线范围当中，按默认策略分配。

图3-6 AE 分配路线示意图



【举例】

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

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

【相关主题】

[HI_MPI_ISP_GetAERouteAttr](#)



HI_MPI_ISP_GetAERouteAttr

【描述】

获取 AE 曝光分配策略。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAERouteAttr	AE 曝光分配策略。	输出

【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	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_SetAERouteAttr](#)



HI_MPI_ISP_SetAEAttrEx

【描述】

设置 AE 曝光扩展属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEAttrEx	AE 曝光属性。	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	参数无效。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 此接口是 AE 曝光属性扩展接口，与 HI_MPI_ISP_SetAEAttr 的区别是该接口支持 10bit 精度的增益(包括模拟增益，数字增益)，并增加支持设置 ISP 数字增益最大值，系统增益最大值。
- 此接口用于设定曝光时间，sensor 数字增益，模拟增益的最大最小值，ISP 数字增益，系统增益最大值。
- 曝光调整的步长属性用于调节曝光收敛的速度，步长值越大曝光的收敛速度会越快但也会导致收敛过程中出现反复震荡。



- 曝光容忍偏差属性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会越不明感且会导致同一目标亮度值多次调节得到的亮度偏差会越大，所以该属性推荐不能设定过大。
- 曝光的亮度补偿属性用于调节曝光的亮度。曝光亮度补偿值越大则图像亮度越高。
- 曝光权重表属性用于对调节感兴趣区域的曝光权重。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetExposureType](#)

HI_MPI_ISP_GetAEAttrEx

【描述】

获取 AE 曝光扩展属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEAttrEx	AE 曝光属性。	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。

【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_GetExposureType](#)

HI_MPI_ISP_SetAEDelayAttr

【描述】

设置 AE 延时属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEDelayAttr	AE 延时属性。	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_ae.h
- 库文件：lib_hiae.a

【注意】

- 此接口是 AE 延时属性接口，图像亮度小于/大于目标亮度时间超过设定值时，AE 才开始调节。



- 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。所以在设置延时属性后，若要达到与设置延时前相同的 AE 收敛速度，需要相应地修改 AE 调节步长。
- 人眼对于过曝比较敏感，建议 u16WhiteDelayFrame 要设置得比 u16BlackDelayFrame 小一些。

【举例】

无。

【相关主题】

HI_MPI_ISP_SetAETtrEx

HI_MPI_ISP_GetAEDelayAttr

【描述】

获取 AE 延时属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAEDelayAttr	AE 延时属性。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_ae.h
- 库文件：lib_hiae.a

【注意】



无。

【举例】

无。

【相关主题】

无。

HI_MPI_ISP_SetMEAttr

【描述】

设置手动曝光属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstMEAttr	手动曝光参数及手动曝光使能参数属性。	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 手动曝光使能参数包括曝光时间使能、模拟增益使能、数字增益使能。
- 手动曝光参数包括曝光时间、模拟增益、数字增益。
- 手动曝光使能参数有效时，必须设置相应的曝光参数。



- 手动曝光时间单位为 sensor 扫描行数。
- 手动模拟增益和数字增益单位为倍数。
- 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。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetMEAttr](#)

HI_MPI_ISP_GetMEAttr

【描述】

获取手动曝光属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetMEAttr( ISP_ME_ATTR_S *pstMEAttr );
```

【参数】

参数名称	描述	输入/输出
pstMEAttr	手动曝光参数及手动曝光使能参数属性。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

若手动曝光参数设置值不合理，获取值会与设置值有差异。



【举例】

无。

【相关主题】

- [HI_MPI_ISP_SetMEAttr](#)
- [HI_MPI_ISP_SetExposureType](#)

HI_MPI_ISP_SetMEAttrEx

【描述】

设置手动曝光属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstMEAttrEx	手动曝光参数及手动曝光使能参数属性。	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 该接口与 HI_MPI_ISP_SetMEAttr 的区别是该接口支持 10bit 精度的增益设置。
- 手动曝光使能参数包括曝光时间使能、模拟增益使能、数字增益使能。
- 由于不支持设置 ISP 数字增益，需要将相应 XX_CMOS.C 驱动程序中



- 手动曝光参数包括曝光时间、模拟增益、数字增益。
- 手动曝光使能参数有效时，必须设置相应的曝光参数。
- 手动曝光时间单位为 sensor 扫描行数。
- 手动模拟增益和数字增益单位为倍数。
- 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。

【举例】

无。

【相关主题】

HI_MPI_ISP_ISP_GetMEAttrEx

HI_MPI_ISP_GetMEAttrEx

【描述】

获取手动曝光属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstMEAttrEx	手动曝光参数及手动曝光使能参数属性。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a



【注意】

若手动曝光参数设置值不合理，获取值会与设置值有差异。

【举例】

无。

【相关主题】

- [HI_MPI_ISP_SetExposureType](#)
- [HI_MPI_ISP_SetMEAttrEx](#)

HI_MPI_ISP_SetExpStaInfo

【描述】

设置 AE 曝光统计信息。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstExpStatistic	曝光统计信息	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

使用该 MPI 设置 ISP 五段直方图的分段位置。



【举例】

无。

【相关主题】

[HI_MPI_ISP_GetExpStaInfo](#)

HI_MPI_ISP_GetExpStaInfo

【描述】

获取 AE 曝光统计信息（如获取 ISP 的 256 段直方图、5 段直方图、分块直方图和平均亮度信息）。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstExpStatistic	曝光统计信息	输出

【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。

【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_SetExpStaInfo](#)

HI_MPI_ISP_QueryInnerStateInfo

【描述】

获取内部状态信息。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstInnerStateInfo	内部状态信息。	输出

【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。

【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_QueryInnerStateInfoEx

【描述】

获取内部状态信息。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstInnerStateInfoEx	内部状态信息。	输出

【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。

【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。

3.4.3 AI 控制模块

光圈控制接口：



- [HI_MPI_ISP_SetIrisType](#): 设置光圈控制类型。
- [HI_MPI_ISP_GetIrisType](#): 获取光圈控制类型。
- [HI_MPI_ISP_SetAIAttr](#): 设置自动光圈的属性。
- [HI_MPI_ISP_GetAIAttr](#): 获取自动光圈的属性。

HI_MPI_ISP_SetIrisType

【描述】

设定光圈的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
enIrisType	光圈控制类型	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】

无

【需求】

- 头文件: `hi_comm_isp.h`、`mpi_isp.h`
- 库文件: `libisp.a`

【注意】

目前暂不支持此功能。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetIrisType](#)



HI_MPI_ISP_GetIrisType

【描述】

获取光圈的控制类型。

【语法】

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

【参数】

参数名称	描述	输入/输出
penIrisType	光圈控制类型	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_SetIrisType](#)

HI_MPI_ISP_SetAIAttr

【描述】

设定自动光圈的控制属性,该函数可实现自动光圈功能的使能,和光圈的校正功能。

【语法】

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



【参数】

参数名称	描述	输入/输出
pstAIAttr	自动光圈属性。	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

该接口分为自动光圈使能和光圈校正。光圈校正是为了获取单板上使光圈停止的电压值，光圈的校正环境要求校正程序启动时外界的光亮环境必须稳定。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetAIAttr](#)

HI_MPI_ISP_GetAIAttr

【描述】

获取自动光圈的控制属性。

【语法】

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

【参数】



参数名称	描述	输入/输出
pstAIAttr	自动光圈属性。	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。

【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_SetAIAttr](#)

3.5 数据类型

3.5.1 Register

- [AE_SENSOR_REGISTER_S](#)：定义 sensor 注册结构体。
- [AE_SENSOR_GAININFO_S](#)：定义 sensor 增益表格的结构体。
- [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_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_GAININFO_S

【说明】

定义 sensor 增益表格的结构体。

【定义】

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

【成员】

成员名称	描述
u32SnsTimes	增益的倍数
u32GainDb	在表格中的索引值

【注意事项】

- u32SnsTimes 表示 sensor 的倍数，精度为 10bit，即 1024 表示为 1 倍增益。



- u32GainDb 表示在 table 中查询得到的表格的 table 的索引。

【相关数据类型及接口】

AE_SENSOR_GAININFO_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_U8 u8Fps, AE_SENSOR_DEFAULT_S
*pstAeSnsDft);
    HI_VOID(*pfn_cmos_slow_framerate_set)(HI_U8 u8SlowFrameRate,
AE_SENSOR_DEFAULT_S *pstAeSnsDft);
    HI_VOID(*pfn_cmos_inttime_update)(HI_U32 u32IntTime);
    HI_VOID(*pfn_cmos_gains_update)(HI_U32 u32Again, HI_U32 u32Dgain);
    HI_VOID (*pfn_cmos_again_calc_table)(HI_U32 u32InTimes,
AE_SENSOR_GAININFO_S *pstAeSnsGainInfo);
    HI_VOID (*pfn_cmos_dgain_calc_table)(HI_U32 u32InTimes,
AE_SENSOR_GAININFO_S *pstAeSnsGainInfo);} AE_SENSOR_EXP_FUNC_S;
```

【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	设置 sensor 的模拟增益表。
pfn_cmos_dgain_calc_table	设置 sensor 的数字增益表。

【注意事项】

- 如果回调函数指针不需要赋值，需要置为 NULL。
- 降帧的意义参见 HI_MPI_ISP_SetSlowFrameRate。



- 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。

【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_U32    u32MaxIntTime;    /* unit is line */
    HI_U32    u32MinIntTime;
    HI_U32    u32MaxIntTimeTarget;
    HI_U32    u32MinIntTimeTarget;
    AE_ACCURACY_S stIntTimeAccu;

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

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

【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}，WDR 模式为{0x20,0x40,0x60,0x80}。
u8AeCompensation	AE 亮度目标值，取值范围为[0,255]，建议用 0x80。
u32LinesPer500ms	每 500ms 的总行数。
u32FlickerFreq	抗闪频率，数值为当前电源频率的 256 倍。
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	最大模拟增益，以倍为单位。
u32MinAgain	最小模拟增益，以倍为单位。
u32MaxAgainTarget	最大模拟增益目标值，以倍为单位。
u32MinAgainTarget	最小模拟增益目标值，以倍为单位。
stAgainAccu	模拟增益精度。
u32MaxDgain	最大数字增益，以倍为单位。
u32MinDgain	最小数字增益，以倍为单位。
u32MaxDgainTarget	最大数字增益目标值，以倍为单位。
u32MinDgainTarget	最小数字增益目标值，以倍为单位。
stDgainAccu	数字增益精度。

【注意事项】

无。

【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;
} 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。



【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。
- 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，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中 `AE_SENSOR_EXP_FUCN_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。
- 规定这两种精度类型，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`

3.5.2 AE

- `ISP_AE_MODE_E`：定义自动曝光的模式。
- `ISP_AE_ATTR_S`：定义 ISP 自动曝光属性。
- `ISP_AE_ATTR_EX_S`：定义 ISP 自动曝光属性（Sensor 增益为 10bit 精度）。
- `ISP_ME_ATTR_S`：定义 ISP 手动曝光属性。
- `ISP_ME_ATTR_EX_S`：定义 ISP 手动曝光属性（Sensor 增益为 10bit 精度）。
- `ISP_EXP_STA_INFO_S`：定义 ISP 曝光统计信息。
- `ISP_INNER_STATE_INFO_S`：定义 ISP 内部状态信息。
- `ISP_INNER_STATE_INFO_EX_S`：定义 ISP 内部状态信息（Sensor 增益为 10bit 精度）。

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_LOW_NOISE        = 0,
    AE_MODE_FRAME_RATE       = 1,
    AE_MODE_BUTT
} ISP_AE_MODE_E;
```

【成员】

成员名称	描述
AE_MODE_LOW_NOISE	噪声优先模式。
AE_MODE_FRAME_RATE	帧率优先模式。

【注意事项】

- 噪声优先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。该模式下，当抗闪打开时，降帧后的曝光时间增加方式是连续的，而不是按 1/100 或 1/120 ms 的倍数增加。
- 帧率优先模式是指自动曝光调节时优先保证帧率。低照度环境下噪声会较大。

【相关数据类型及接口】

无。

ISP_AE_ATTR_S

【说明】

定义 ISP 自动曝光属性。

【定义】

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



```
} ISP_AE_ATTR_S;
```

【成员】

成员名称	描述
enAEMode	自动曝光的优先模式，如帧率优先、噪声优先。
u16ExpTimeMax	自动曝光的最大曝光时间，可用于限定自动曝光的最大曝光时间，如对运动场景则可以将该成员变量设置较小值。 取值范围：[0x2, 0xFFFF]，具体范围与 sensor 相关。
u16ExpTimeMin	自动曝光的最小曝光时间。 取值范围：[0x2, 0xFFFF]，具体范围与 sensor 相关。
u16DGainMax	自动曝光的最大数字增益值。 取值范围：[0x1, 0xFF]，具体范围与 sensor 相关。
u16DGainMin	自动曝光的最小数字增益值。 取值范围：[0x1, 0xFF]，具体范围与 sensor 相关。
u16AGainMax	自动曝光的最大模拟增益值。 取值范围：[0x1, 0xFF]，具体范围与 sensor 相关。
u16AGainMin	自动曝光的最小模拟增益值。 取值范围：[0x1, 0xFF]，具体范围与 sensor 相关。
u8ExpStep	自动曝光调整时的初始步长。
s16ExpTolerance	自动曝光调整时对曝光量的容忍偏差。 取值范围：[0x0, 0xFFFF]。
u8ExpCompensation	自动曝光调整时对曝光补偿量。
enFrameEndUpdateMode	曝光变量更新模式。
bByPassAE	自动曝光算法是否使用。
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	自动曝光的权重表。

【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。最小数字增益值的设定目前暂不支持。
 - 当 enAEMode 为 LowNoise 模式时，最小曝光时间可以设置为大于正常帧率时一帧最大曝光时间。



- 当 enAEMode 为 FrameRate 模式时，最小曝光时间不能超过正常帧率时一帧的最大曝光时间，即使设置了超过一帧最大的曝光时间，也会被系统限制在一帧时间之内。
- 自动曝光调整时的初始步长
值越大则自动曝光的收敛速度越快，但自动曝光出现明暗震荡的几率越大。
- 自动曝光调整时对曝光量的容忍偏差
值越大则对外界环境亮度变化的敏感度越小。
- 自动曝光调整时对曝光补偿量
值越大则自动曝光的目标亮度值越大，图像越亮。
- 曝光变量更新模式
目前仅对 Pana34041，需要设置为 ISP_AE_FRAME_END_UPDATE_1，以避免打开抗闪功能后 AE 调整时图像出现闪烁；其它 sensor 不需要设置此变量或设置为 ISP_AE_FRAME_END_UPDATE_0。
- 自动曝光的权重表
自动曝光的静态统计信息分为 7x9 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的变化会使图像的统计信息产生更多的变化。

【相关数据类型及接口】

无。

ISP_AE_ATTR_EX_S

【说明】

定义 ISP 自动曝光属性（Sensor 增益为 10bit 精度）。

【定义】

```
typedef struct hiISP_AE_ATTR_EX_S
{
    ISP_AE_MODE_E enAEMode;
    HI_U32 u32ExpTimeMax;
    HI_U32 u32ExpTimeMin;
    HI_U32 u32DGainMax;
    HI_U32 u32DGainMin;
    HI_U32 u32AGainMax;
    HI_U32 u32AGainMin;
    HI_U32 u32ISPDGainMax;
    HI_U32 u32SystemGainMax;
    HI_U8 u8ExpStep;
    HI_S16 s16ExpTolerance;
    HI_U8 u8ExpCompensation;
    ISP_AE_FRAME_END_UPDATE_E enFrameEndUpdateMode;
    HI_BOOL bByPassAE;
```



```
HI_U8 u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_AE_ATTR_EX_S;
```

【成员】

成员名称	描述
enAEMode	自动曝光的优先模式，如帧率优先,噪声优先。
u32ExpTimeMax	自动曝光的最大曝光时间，可用于限定自动曝光的最大曝光时间，如对运动场景则可以将该成员变量设置较小值。 取值范围：[0x2, 0xFFFF]，具体范围与 sensor 相关。
u32ExpTimeMin	自动曝光的最小曝光时间。 取值范围：[0x2, 0xFFFF]，具体范围与 sensor 相关。
u32DGainMax	自动曝光的最大数字增益值,10bit 小数精度 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32DGainMin	自动曝光的最小数字增益值,10bit 小数精度 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32AGainMax	自动曝光的最大模拟增益值,10bit 小数精度 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32AGainMin	自动曝光的最小模拟增益值,10bit 小数精度 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32SystemGainMax	最大系统增益，即包括 sensor 的最大数字增益，最大模拟增益和 ISP 最大数字增益之积，10bit 小数精度。 取值范围：[0x400,0xFFFFFFFF]。
u8ExpStep	自动曝光调整时的初始步长。
s16ExpTolerance	自动曝光调整时对曝光量的容忍偏差。 取值范围：[0x0, 0xFFFF]。
u8ExpCompensation	自动曝光调整时对曝光补偿量。
enFrameEndUpdateMode	曝光变量更新模式。
bByPassAE	自动曝光算法是否使用。
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	自动曝光的权重表。



【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。最小数字增益值的设定目前暂不支持。
- 自动曝光调整时的初始步长
值越大则自动曝光的收敛速度越快，但自动曝光出现明暗震荡的几率越大。
- 自动曝光调整时对曝光量的容忍偏差
值越大则对外界环境亮度变化的敏感度越小。
- 自动曝光调整时对曝光补偿量
值越大则自动曝光的目标亮度值越大，图像越亮。
- 曝光变量更新模式
目前针对 Pana34041，imx104,需要设置为 ISP_AE_FRAME_END_UPDATE_1,以避免打开抗闪功能后，AE 调整图像出现闪烁，针对 OV9712，MT9P006 设置为 ISP_AE_FRAME_END_UPDATE_2，其他 sensor 采用默认值 ISP_AE_FRAME_END_UPDATE_0。
- 自动曝光的权重表
自动曝光的静态统计信息分为 7x9 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的变化会使图像的统计信息产生更多的变化。

【相关数据类型及接口】

无。

ISP_ME_ATTR_S

【说明】

定义 ISP 手动曝光属性。

【定义】

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

【成员】



成员名称	描述
s32AGain	手动模拟增益。 取值范围：[0x0, 0xFF]，具体范围与 sensor 相关。
s32DGain	手动数字增益。 取值范围：[0x0, 0xFF]，具体范围与 sensor 相关。
u32ExpLine	手动曝光时间。 取值范围：[0x0, 0xFFFF]，具体范围与 sensor 相关。
bManualExpLineEnable	手动曝光时间使能。
bManualAGainEnable	手动模拟增益使能。
bManualDGainEnable	手动数字增益使能。

【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。
- 手动曝光时间单位为 sensor 扫描行数。
- 手动模拟和数字增益单位为倍数。
- 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。

【相关数据类型及接口】

无。

ISP_ME_ATTR_EX_S

【说明】

定义 ISP 手动曝光属性（Sensor 增益为 10bit 精度）。

【定义】

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

【成员】



成员名称	描述
s32AGain	手动模拟增益。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
s32DGain	手动数字增益。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32ExpLine	手动曝光时间。 取值范围：[0x0, 0xFFFF]，具体范围与 sensor 相关。
bManualExpLineEnable	手动曝光时间使能。
bManualAGainEnable	手动模拟增益使能。
bManualDGainEnable	手动数字增益使能。

【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。
- 手动曝光时间单位为 sensor 扫描行数。
- 手动模拟和数字增益单位为倍数。
- 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。

【相关数据类型及接口】

无。

ISP_EXP_STA_INFO_S

【说明】

定义 ISP 曝光统计信息。

【定义】

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




【成员】

成员名称	描述
u8ExpHistThresh[4]	五段直方图分段点 取值范围：[0x0, 0xFF]
u16ExpStatistic[][5]	分区域统计信息 取值范围：[0x0, 0xFFFF]
u16Exp_Hist256Value[256]	256 段直方图统计信息 取值范围：[0x0, 0xFFFF]
u16Exp_Hist5Value[5]	5 段直方图统计信息 取值范围：[0x0, 0xFFFF]
u8AveLum	平均亮度信息 取值范围：[0x0, 0xFF]
u8ExpHistTarget[5]	五段直方图分段目标值 取值范围：[0x0, 0xFF]
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：[-0xFFFF, 0xFFFF]，该值为只读。

【注意事项】

AE 算法根据 u8ExpHistTarget 的值和实际图像落在五段直方图的像素值的误差的加权，调整曝光和增益，使得实际图像落在五段直方图的像素值接近 u8ExpHistTarget 的值。举例说明：如果 u8ExpHistTarget[0]、u8ExpHistTarget[1] 的值较大，那么说明我们期望最终落在较暗区间的像素数偏多，这样最终亮度会比较暗。

AE 算法的由亮到暗的收敛速度和由暗到亮的收敛速度，受五段直方图的分段方法的影响，如果中间第三段的中心值远远偏离 0x80，那么两个方向的收敛速度将不一致。在通过 u8ExpHistThresh 参数调整五段直方图时，同时需要调整 u8ExpHistTarget 的值，以确定各段直方图期望的像素目标值，以避免过亮或过暗。

【相关数据类型及接口】

无。

ISP_INNER_STATE_INFO_S

【说明】

定义 ISP 内部状态信息。

【定义】



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

【成员】

成员名称	描述
u32ExposureTime	Sensor 曝光时间，取值范围[0x0,0xFFFFFFFF]。
u32AnalogGain	Sensor 模拟增益，取值范围[0x0,0xFFFFFFFF]。
u32DigitalGain	Sensor 数字增益，取值范围[0x0,0xFFFFFFFF]。
u32Exposure	Sensor 曝光量，取值范围[0x0,0xFFFFFFFF]。
u16AE_Hist256Value[256]	256 段直方图，取值范围[0x0,0xFFFFFFFF]。
u16AE_Hist5Value[5]	区域加权后的 5 段直方图。
u8AveLum	图像平均亮度， 取值范围： [0x0,0xFF]
bExposureIsMAX	0: ISP 未达到最大曝光水平； 1: ISP 达到最大曝光水平。

【注意事项】

- 曝光量为曝光时间、模拟增益、数字增益、ISP 数字增益四者乘积。
- Sensor 数字增益和模拟增益使用倍数表示。
- 256 段和 5 段直方图均经过归一化处理，取值范围为 0~65535。
- 图像平均亮度经过归一化处理，取值范围为 0~255。
- ISP 是否达到最大曝光水平的计算未考虑光圈状态。

【相关数据类型及接口】

无。

ISP_INNER_STATE_INFO_EX_S

【说明】

定义 ISP 内部状态信息（Sensor 增益为 10bit 精度）。



【定义】

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

【成员】

成员名称	描述
u32ExposureTime	Sensor 曝光时间。 取值范围：[0x0, 0xFFFF]。
u32AnalogGain	Sensor 模拟增益,10bit 精度。 取值范围：[0x400, 0xFFFFFFFF]。
u32DigitalGain	Sensor 数字增益,10bit 精度。 取值范围：[0x400, 0xFFFFFFFF]。
u32ISPDigitalGain	ISP 数字增益：10bit 精度。 取值范围：[0x400, 0xFFFFFFFF]。
u32Exposure	Sensor 曝光量。 取值范围：[0x0, 0xFFFFFFFF]。
u16AE_Hist256Value[256]	256 段直方图。 取值范围：[0x0, 0xFFFF]
u16AE_Hist5Value[5]	区域加权后的 5 段直方图。 取值范围：[0x0, 0xFFFF]
u8AveLum	图像平均亮度。 取值范围：[0x0, 0xFF]。
bExposureIsMAX	0：ISP 未达到最大曝光水平； 1：ISP 达到最大曝光水平。

【注意事项】



- 曝光时间单位为扫描行数，曝光量为曝光时间、模拟增益、数字增益、ISP 数字增益四者乘积。
- Sensor 数字增益和模拟增益使用倍数表示。
- 256 段和 5 段直方图均经过归一化处理，取值范围为 0~65535。
- 图像平均亮度经过归一化处理，取值范围为 0~255。
- ISP 是否达到最大曝光水平的计算未考虑光圈状态。

【相关数据类型及接口】

无。

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;
    HI_U32  u32ApePercent;
} ISP_AE_ROUTE_NODE_S;
```

【成员】

成员名称	描述
u32IntTime	节点曝光时间。 取值范围：[0x0, 0xFFFF]。
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]。
u32ApePercent	节点光圈大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[0x0, 0x64]。

【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。
- 相邻节点若存在 2 个或以上的值变化，则必须为等曝光量节点，即曝光时间、增益和光圈的乘积必须相等。



- 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。

【相关数据类型及接口】

[HI_MPI_ISP_SetAERouteAttr](#)

ISP_AE_ROUTE_S

【说明】

定义 ISP 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 8。
astRouteNode [ISP_AE_ROUTE_MAX_NODES]	曝光分配路线节点属性。

【注意事项】

- 最大支持八个节点，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。
- 默认的 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。

【相关数据类型及接口】

- [ISP_AE_ROUTE_NODE_S](#)
- [HI_MPI_ISP_SetAERouteAttr](#)

ISP_AE_DELAY_S

【说明】

定义 ISP 曝光延时属性。



【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。

【相关数据类型及接口】

[HI_MPI_ISP_SetAEDelayAttr](#)

3.5.3 AI

- [ISP_OP_TYPE_E](#): 定义 ISP AI 模式。
- [ISP_IRIS_STATUS_E](#): 定义 ISP 光圈状态。
- [ISP_TRIGGER_STATUS_E](#): 定义 ISP 校正（检测）状态。
- [ISP_AI_ATTR_S](#): 定义 ISP AI 属性。

ISP_OP_TYPE_E

【说明】

定义 ISP AI 模式。

【定义】

```
typedef struct 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	曝光使用自动模式
OP_TYPE_MANUAL	曝光使用手动模式

【注意事项】

当前 AI 不支持手动模式当 AE 使用手动模式时，AI 的自动模式也不生效。

【相关数据类型及接口】

无。

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】

无。

【相关数据类型及接口】

无。



ISP_TRIGGER_STATUS_E

【说明】

定义 ISP 校正（检测）状态。

【定义】

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

【成员】

成员名称	描述
ISP_TRIGGER_INIT	初始状态，未校正
ISP_TRIGGER_SUCCESS	校正成功结束
ISP_TRIGGER_TIMEOUT	校正超时结束

【注意事项】

无。

【相关数据类型及接口】

无。

ISP_AI_ATTR_S

【说明】

定义 ISP AI 属性。

【定义】

```
typedef struct hiISP_AI_ATTR_S
{
    HI_BOOL bIrisEnable;
    HI_BOOL bIrisCalEnable;
    HI_U32  u32IrisHoldValue;
    ISP_IRIS_STATUS_E enIrisStatus;
    ISP_TRIGGER_STATUS_E enTriggerStatus;
    HI_U16  u16IrisStopValue;
    HI_U16  u16IrisCloseDrive;
```




```
HI_U16 u16IrisTriggerTime;  
HI_U8 u8IrisInertiaValue;  
} ISP_AI_ATTR_S;
```

【成员】

成员名称	描述
bIrisEnable	使能 AI 功能。
bIrisCalEnable	使能 AI 校正功能。
u32IrisHoldValue	AI 校正值。 取值范围为[0x0, 0x3E8]。
enIrisStatus	光圈状态。
enTriggerStatus	光圈校正结果状态信息。
u16IrisStopValue	AI 校正值的初始值。 取值范围为[0x0, 0x3E8]。
u16IrisCloseDrive	关闭光圈的驱动值。 取值范围为[0x0, 0x3E8]。
u16IrisTriggerTime	AI 校正超时时间，以帧数为单位。 取值范围为[0x0, 0xFFFF]。
u8IrisInertiaValue	自动光圈反向运动的惯性时间，以帧数为单位。

【注意事项】

- AI 校正功能主要是为了获取与单板相匹配的控制光圈停止的电压值，再做 AI 校正时必须确保外界的环境亮度稳定。
- 合理设定 u16IrisStopValue，能使 AI 校正工作较快完成。
- u16IrisCloseDrive 推荐设置为[700,900]，较大的值表示光圈关闭的更快。
- 若 AI 校正花费的时间大于等于 u16IrisTriggerTime 表示的帧数，则校正超时结束。
- 自动光圈的状态由关转向开时，不会立即进入开的状态，由于运动惯性，保持关的运动状态一段时间然后再进入开的状态。u8IrisInertiaValue 当前默认的值是 5，推荐设置为[5, 10]，设置值越大，AI 校准所需时间越长。

【相关数据类型及接口】

- [ISP_IRIS_STATUS_E](#)
- [ISP_TRIGGER_STATUS_E](#)



4 AWB

4.1 概述

可见光的光谱成分随色温变化而变化，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

4.3.1 AWB 模块工作原理

AWB 模块有硬件的 WB 统计信息模块及 AWB 控制策略算法 firmware 两部分组成。ISP 的 WB 统计信息模块统计 sensor 输出的 R, G, B 三个颜色通道的平均比值。可提供整幅图像加权后的比值，还可提供将整幅图像分成 M*N 区块的每个区块的比值。

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

$$awb_rg_{sum} = \sum_{p=0}^{P-1} \frac{G_p}{R_p} \cdot \theta_p$$

$$awb_{sum} = \sum_{p=0}^{P-1} \theta_p$$

其中 θ 指示当前点是否白点，R 表示白点的红色分量值，G 表示白点的绿色分量值，awb 是白点个数，awb_rg 是 G/R 的均值。

支持区间权重设置（默认各窗权重相同），输出加权后的全局统计信息。

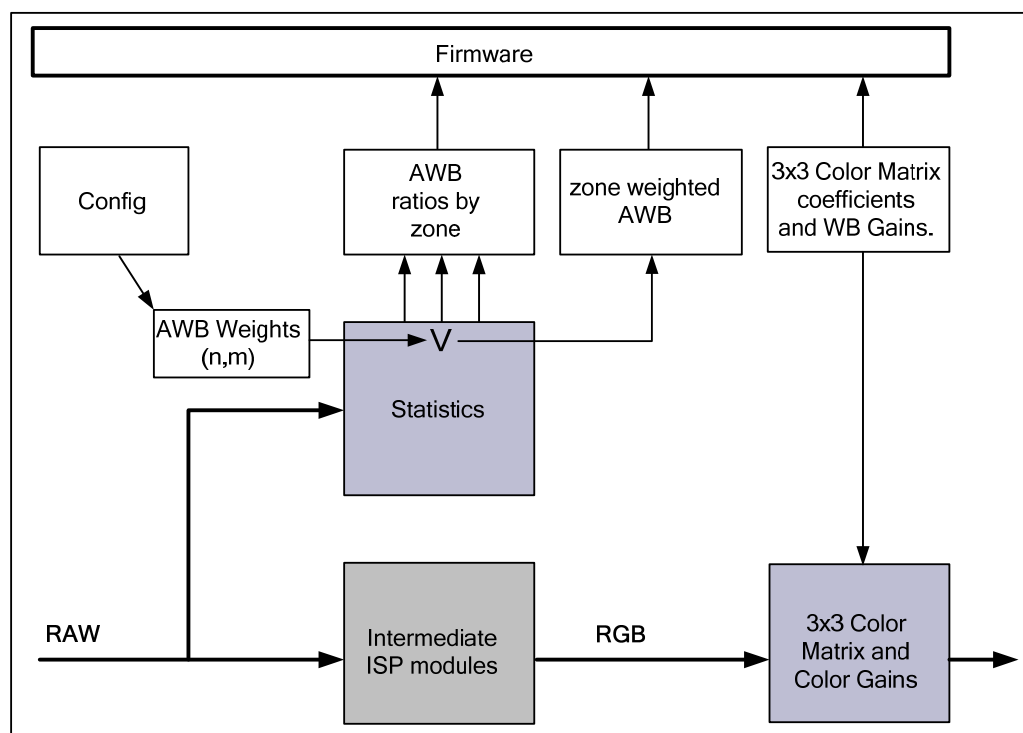
ISP 的 AWB 模块提供的统计信息主要为 R/G，B/G 的值。这两个值是一个平均值，是由每个 zone 中白点的 R/G 与 B/G 的总和与 AWB 的权重系数通过如下公式计算得到：

$$AWB_RG = \frac{\sum_{m,n} w_{m,n} * awb_rg_{m,n}}{\sum_{m,n} w_{m,n} * awb_{m,n}}$$

其中 awb_rg 为[m][n]zone 的白点的 R/G 均值，awb 为[m][n] zone 中白点的总数。

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

- [HI_MPI_AWB_Register](#): 向 ISP 注册 AWB 库。
- [HI_MPI_AWB_SensorRegCallBack](#): AWB 库提供的 sensor 注册的回调接口。



HI_MPI_AWB_Register

【描述】

向 ISP 注册 AWB 库。

【语法】

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

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

【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_SensorRegCallBack

【描述】

AWB 库提供的 sensor 注册的回调接口。

【语法】

```
HI_S32 HI_MPI_AWB_SensorRegCallBack(ALG_LIB_S *pstAwbLib, SENSOR_ID
```



```
SensorId, AWB_SENSOR_REGISTER_S *pstRegister);
```

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

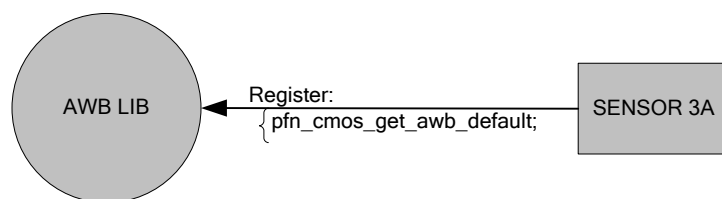
【需求】

- 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;

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

【相关主题】

无。

4.4.2 AWB 控制模块

- [HI_MPI_ISP_SetWBType](#): 设置白平衡类型
- [HI_MPI_ISP_GetWBType](#): 获取白平衡类型
- [HI_MPI_ISP_GetAWBAttr](#): 设置自动白平衡属性
- [HI_MPI_ISP_GetAWBAttr](#): 获取自动白平衡属性
- [HI_MPI_ISP_SetMWBAAttr](#): 设置手动白平衡属性
- [HI_MPI_ISP_GetMWBAAttr](#): 获取手动白平衡属性
- [HI_MPI_ISP_SetAWBAlgType](#): 设置白平衡算法类型
- [HI_MPI_ISP_GetAWBAlgType](#): 获取白平衡算法类型
- [HI_MPI_ISP_SetAdvAWBAttr](#): 设置 ADV 白平衡算法属性
- [HI_MPI_ISP_GetAdvAWBAttr](#): 获取 ADV 白平衡的属性
- [HI_MPI_ISP_SetLightSource](#): 设置独立光源点属性
- [HI_MPI_ISP_GetLightSource](#): 获取独立光源点属性

HI_MPI_ISP_SetWBType

【描述】

设置白平衡的控制类型。

【语法】

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

【参数】

参数名称	描述	输入/输出
enWBType	白平衡控制类型	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_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】

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

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetWBType](#)

HI_MPI_ISP_GetWBType

【描述】

获取白平衡的控制类型。

【语法】

```
HI_S32 HI_MPI_ISP_GetWBType( ISP\_OP\_TYPE\_E *penWBType );
```

【参数】

参数名称	描述	输入/输出
penWBType	白平衡控制类型	输出

【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。

【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_SetWBType](#)

HI_MPI_ISP_SetAWBAttr

【描述】

设置自动白平衡的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAWBAttr	自动白平衡属性。	输入

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【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】

此接口可配置 AWB 的权重表，通过调整 u8RGStrength 和 u8BGStrength 的值改变 AWB 调整强度，配置 AWB 算法的色温上下限以及选择使用全局 AWB 或区域 AWB 算法。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetWBType](#)

HI_MPI_ISP_GetAWBAttr

【描述】

获取自动白平衡的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAWBAttr	自动白平衡属性。	输出

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【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_SetWBType](#)

HI_MPI_ISP_SetMWBAAttr

【描述】

设置手动白平衡的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstMWBAAttr	手动白平衡属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。



【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】

当白平衡设置为手动时，通过此接口手动调整白平衡。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetWBType](#)

HI_MPI_ISP_GetMWBAAttr

【描述】

获取手动白平衡的属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstMWBAAttr	手动白平衡属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_SetWBType](#)

HI_MPI_ISP_SetAWBAlgType

【描述】

设置白平衡算法类型。

【语法】

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

【参数】

参数名称	描述	输入/输出
enALGType	白平衡算法类型	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_ILLEGAL_PARAM	参数无效。



【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 算法类型为 AWB_ALG_DEFAULT 时，与 SPC070 版本效果一致；算法类型为 AWB_ALG_ADVANCE 时，为改进后的 AWB 算法。
- 设置算法类型为 AWB_ALG_ADVANCE 后，建议通过 HI_MPI_ISP_SetAdvAWBAttr 接口调整算法参数，以达到最优效果。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetAWBAlgType](#)[HI_MPI_ISP_GetWBType](#)

HI_MPI_ISP_GetAWBAlgType

【描述】

获取白平衡算法类型。

【语法】

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

【参数】

参数名称	描述	输入/输出
penALGType	白平衡算法类型	输出

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【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_SetAWBAlgType](#)

HI_MPI_ISP_SetAdvAWBAttr

【描述】

设置 ADV 白平衡算法属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstAdvAWBAttr	ADV 白平衡属性。	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

- 仅在设置白平衡算法为 AWB_ALG_ADVANCE 时生效。
- 支持设置白平衡的容忍偏差，色温曲线的限制条件等参数。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetWBTypeHI_MPI_ISP_GetAdvAWBAttr](#)

HI_MPI_ISP_GetAdvAWBAttr

【描述】

获取 ADV 白平衡的属性。

【语法】

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



【参数】

参数名称	描述	输入/输出
pstAdvAWBAttr	ADV 白平衡属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_ERR_ISP_NULL_PTR	空指针错误。

【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【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_SetAdvAWBAttr](#)

HI_MPI_ISP_SetLightSource

【描述】



设置独立光源点属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstLightSource	独立光源点属性。	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。

【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

仅在设置白平衡算法为 AWB_ALG_ADVANCE 时生效。支持设置最多 4 个独立光源点。

【举例】

无。



【相关主题】

[HI_MPI_ISP_GetLightSource](#)

HI_MPI_ISP_GetLightSource

【描述】

获取独立光源点属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstLightSource	独立光源点属性。	输出

【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。

【差异说明】

芯片类型	描述
Hi3516	不支持该接口
Hi3518	支持

【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_SetLightSource](#)

4.4.3 WB 统计信息

- [HI_MPI_ISP_SetColorTemp](#): 设置目标色温
- [HI_MPI_ISP_GetColorTemp](#): 获取环境色温
- [HI_MPI_ISP_SetWBStaInfo](#): 设置白平衡统计相关参数
- [HI_MPI_ISP_GetWBStaInfo](#): 获取白平衡统计信息

HI_MPI_ISP_SetColorTemp

【描述】

设置 AWB 色温。

【语法】

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

【参数】

参数名称	描述	输入/输出
u16ColorTemp	色温，单位为开尔文。	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。



【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

目前暂不支持此功能。

【举例】

无。

【相关主题】

无。

HI_MPI_ISP_GetColorTemp

【描述】

获取 AWB 色温。

【语法】

```
HI_S32 HI_MPI_ISP_GetColorTemp(HI_U16 *pul6ColorTemp);
```

【参数】

参数名称	描述	输入/输出
pul6ColorTemp	当前 AWB 色温，单位为开尔文。	输出

【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。

【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_SetWBStaInfo

【描述】

设置白平衡统计信息参数。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstWBStatistic	白平衡统计信息。	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。

【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_GetWBStaInfo](#)

HI_MPI_ISP_GetWBStaInfo

【描述】

获取白平衡统计信息参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetWBStaInfo(ISP\_WB\_STA\_INFO\_S *pstWBStatistic);
```

【参数】

参数名称	描述	输入/输出
pstWBStatistic	白平衡统计信息。	输出

【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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

无。

【举例】

用户自定义 AWB 算法示例。

```
ISP_AWB_ATTR_S stAWBAttr;  
ISP_MWB_ATTR_S stMWBAAttr;
```



```
ISP_WB_STA_INFO_S stWBSTAInfo;
int i, j;

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

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

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

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

    stMWBAttr.ul6Rgain = stWBSTAInfo.ul6GRgain;
    stMWBAttr.ul6Bgain = stWBSTAInfo.ul6GBgain;
    stMWBAttr.ul6Ggain = 0x100;

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

【相关主题】

[HI_MPI_ISP_SetWBStaInfo](#)



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。
- [ISP_ME_ATTR_S](#): 定义 ISP 手动曝光属性。

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  ul6WbRefTemp;          /* reference color temperature for WB */
    HI_U16  au16GainOffset[4];     /* gain offset for white balance */
    HI_S32  as32WbPara[6];        /* parameter for wb curve */
    AWB_AGC_TABLE_S  stAgcTbl;
    AWB_CCM_S  stCcm;
} AWB_SENSOR_DEFAULT_S;
```

【成员】

成员名称	成员名称	描述
ul6WbRefTemp	-	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[0,0xFFFF]。
as32WbPara	-	校正工具给出的白平衡参数，取值范围为[0,0xFFFFFFFF]。
stAgcTbl	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]。
stCcm	u16HighColorTemp	高色温，取值范围为[0,0xFFFF]。



成员名称	成员名称	描述
	au16HighCCM	高色温颜色还原矩阵，取值范围为[0,0xFFFF]。
	u16MidColorTemp	中色温，取值范围为[0,0xFFFF]。
	au16MidCCM	中色温颜色还原矩阵，取值范围为[0,0xFFFF]。
	u16LowColorTemp	低色温，取值范围为[0,0xFFFF]。
	au16LowCCM	低色温颜色还原矩阵，取值范围为[0,0xFFFF]。

【注意事项】

无。

【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_MWB_ATTR_S](#): 定义 ISP 手动白平衡属性。
- [ISP_WB_ZONE_STA_INFO_S](#): 定义区间白平衡统计信息。
- [ISP_WB_STA_INFO_S](#): 定义白平衡统计信息。
- [ISP_AWB_CALIBRATION_S](#): 保存 ISP 自动白平衡校正参数。
- [ISP_AWB_ALG_TYPE_E](#): 定义白平衡算法类型。
- [ISP_ADV_AWB_ATTR_S](#): 定义 ADV 白平衡精调参数。
- [ISP_AWB_LIGHTSOURCE_INFO_S](#): 定义单个光源点的属性。
- [ISP_AWB_ADD_LIGHTSOURCE_S](#): 定义独立光源点属性。

ISP_AWB_ATTR_S

【说明】

定义 ISP 自动白平衡属性。

【定义】

```
typedef struct hiISP_AWB_ATTR_S
{
    ISP_AWB_CALIBRATION_S stAWBCalibration;
    HI_U16 u16Speed;
    HI_U8 u8RGStrength;
    HI_U8 u8BGStrength;
```



```
HI_U8 u8ZoneSel;  
HI_U8 u8HighColorTemp;  
HI_U8 u8LowColorTemp;  
HI_U8 u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];  
} ISP_AWB_ATTR_S;
```

【成员】

成员名称	描述
stAWBCalibration	自动白平衡校正参数
u16Speed	自动白平衡收敛速度 默认值：0x40
u8RGStrength	自动白平衡 R 通道校准强度。
u8BGStrength	自动白平衡 B 通道校准强度。
u8ZoneSel	自动白平衡算法选择。
u8HighColorTemp	自动白平衡算法的色温上限。
u8LowColorTemp	自动白平衡算法的色温下限。
u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]	自动白平衡分区间权重表。 取值范围：[0, 15]。

【差异说明】

芯片	描述
Hi3516	<pre>#define WEIGHT_ZONE_ROW 7 #define WEIGHT_ZONE_COLUMN 9</pre> <p>AWB 统计模块支持将图像平均分为 7x9 个区间，输出每个区间的 AWB 统计信息。</p> <p>u8ZoneSel 取值为 0 大于等于 63 时，选择全局 AWB 算法； u8ZoneSel 取值在[1, 62]之间时，选择区间 AWB 算法，具体取值没有影响。</p>
Hi3518	<pre>#define WEIGHT_ZONE_ROW 15 #define WEIGHT_ZONE_COLUMN 17</pre> <p>AWB 统计模块支持将图像平均分为 15x17 个区间，输出每个区间的 AWB 统计信息。</p> <p>u8ZoneSel 取值为 0 或 255 时，选择全局 AWB 算法； u8ZoneSel 取值在[1, 254]之间时，选择区间 AWB 算法，具体取值没有影响。</p>



【注意事项】

- 自动白平衡 R/B 通道校准强度
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。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。
- 自动白平衡算法选择
 - 全局 AWB 算法，整幅图像参与自动白平衡算法统计；
 - 区域 AWB 算法，AWB 算法自动对所有窗的统计信息排序，选择前 u8ZoneSel 个窗参与 AWB 运算。区域 AWB 算法对大面积纯色场景效果好，但 CPU 消耗较大。
- 自动白平衡算法的色温上限/下限
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，则图像会严重偏色。
- 自动白平衡权重表
AWB 全局统计信息为 17x15 区域进行加权平均得到，通过设定权重表可改变每个区域对统计信息值的影响度。如比较关心中心区域的白平衡效果可增大中心区域的权重值。仅选择全局 AWB 算法时区间权重设置生效。

【相关数据类型及接口】

无。

ISP_AWB_CALIBRATION_S

【说明】

保存 ISP 自动白平衡校正参数。

【定义】

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

【成员】

成员名称	描述
as32CurvePara[6]	自动白平衡曲线校正参数 取值范围：[0x0, 0xFFFFFFFF] 默认值由文件 cms.c 内结构体 cms_isp_default_t 的成员 wb_para[6]设定
au16StaticWB[4]	静态白平衡校正参数



成员名称	描述
	取值范围：[0x0, 0xFFFF] 默认值由文件 <code>cmos.c</code> 内结构体 <code>cmos_isp_default_t</code> 的成员 <code>gain_offset [4]</code> 设定
<code>u16RefTemp</code>	自动白平衡参考色温 取值范围：[0x0, 0xFFFF] 默认值由文件 <code>cmos.c</code> 内结构体 <code>cmos_isp_default_t</code> 的成员 <code>wb_ref_temp</code> 设定

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 u16Ggain;
    HI_U16 u16Bgain;
} ISP_MWB_ATTR_S;
```

【成员】

成员名称	描述
<code>u16Rgain</code>	手动白平衡红色通道增益。 取值范围：[0x0,0xFFFF]。
<code>u16Ggain</code>	手动白平衡绿色通道增益。 取值范围：[0x0,0xFFFF]。
<code>u16Bgain</code>	手动白平衡蓝色通道增益。 取值范围：[0x0,0xFFFF]。

【注意事项】

无。

【相关数据类型及接口】

无。



ISP_WB_ZONE_STA_INFO_S

【说明】

定义区间白平衡统计信息。

【定义】

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

【成员】

成员名称	描述
u16Rg	区间白点 G/R 均值，数据格式为定点 4.8。 取值范围：[0x0, 0xFFFF]。
u16Bg	区间白点 G/B 均值，数据格式为定点 4.8。 取值范围：[0x0, 0xFFFF]。
u32Sum	区间白点个数。 取值范围：[0x0, 0xFFFFFFFF]

【注意事项】

图像分为 M x N 个区间，输出各区间白点统计信息。

【相关数据类型及接口】

无。

ISP_WB_STA_INFO_S

【说明】

定义白平衡统计信息。

【定义】

```
typedef struct hiISP_WB_STA_INFO_S
{
    HI_U16 u16WhiteLevel;
    HI_U16 u16BlackLevel;
    HI_U16 u16CbMax;
    HI_U16 u16CbMin;
    HI_U16 u16CrMax;
```



```

HI_U16 u16CrMin;
HI_U16 u16GRgain;
HI_U16 u16GBgain;
HI_U32 u32GSum;
HI_U32 u32Rgain;
HI_U32 u32Ggain;
HI_U32 u32Bgain;
ISP_WB_ZONE_STA_INFO_S stZoneSta[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_WB_STA_INFO_S;

```

【成员】

成员名称	描述
u16WhiteLevel	白点的亮度上限值。 取值范围: (u16BlackLevel, 0x3FF]。
u16BlackLevel	白点的亮度下限值。 取值范围: [0, u16WhiteLevel)。
u16CbMax	白点的 B/G 上限, 数据格式为定点 4.8。 取值范围为[0x0, 0xFFF]。
u16CbMin	白点的 B/G 下限, 数据格式为定点 4.8。 [u16CbMin, u16CbMax]范围内的点参与白平衡统计。 取值范围为[0x0, u16CbMax)。
u16CrMax	白点的 R/G 上限, 数据格式为定点 4.8。 取值范围为[0x0, 0xFFF]。
u16CrMin	白点的 R/G 下限, 数据格式为定点 4.8。 [u16CrMin, u16CrMax]范围内的点参与白平衡统计。 取值范围为[0x0, u16CrMax)。
u16GRgain	加权后的全局白平衡统计信息, G/R。 取值范围为[0x0, 0xFFFF]。
u16GBgain	加权后的全局白平衡统计信息, G/B。 取值范围为[0x0, 0xFFFF]。
u32GSum	参与白平衡统计的白点个数。 取值范围为[0x0, 0xFFFF]。
u32Rgain	R 通道的实时增益值, 该值只能获取, 无法设置
u32Ggain	G 通道的实时增益值, 该值只能获取, 无法设置



成员名称	描述
u32Bgain	B 通道的实时增益值，该值只能获取，无法设置
stZoneSta[WEIGHT_ZONE_ROW] [WEIGHT_ZONE_COLUMN]	分区间白平衡统计信息。

【注意事项】

无。

【相关数据类型及接口】

[ISP_WB_ZONE_STA_INFO_S](#)

ISP_AWB_ALG_TYPE_E

【说明】

定义白平衡算法类型。

【定义】

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

【成员】

成员名称	描述
AWB_ALG_DEFAULT	SPC070 版本白平衡算法。
AWB_ALG_ADVANCE	优化后的白平衡算法。

【注意事项】

AWB_ALG_ADVANCE 在白平衡稳定性，混合光源和高低色温的精度等方面有提升。

【相关数据类型及接口】

无。

ISP_ADV_AWB_ATTR_S

【说明】

定义 ADV 白平衡精调参数。



【定义】

```
typedef struct hiISP_ADV_AWB_ATTR_S
{
    HI_BOOL bAccuPrior;
    HI_U8  u8Tolerance;
    HI_U16 u16CurveLLimit;
    HI_U16 u16CurveRLimit;
} ISP_ADV_AWB_ATTR_S;
```

【成员】

成员名称	描述
bAccuPrior	使能后，可提升室内普通场景白平衡精度。 混合光源，大面积纯色，室外等场景建议关闭。
u8Tolerance	白平衡的容忍偏差。 室外自然光源色温是渐变的，建议 ≤ 4 。 室内人工光源可适当增大容忍偏差，可排除运动纯色的干扰。
u16CurveLLimit	对色温曲线的左侧宽度做限制。 取值范围为[0x0, 0xFF]，取值越小，可支持的光源范围越宽，白平衡精度稍有下降。
u16CurveRLimit	对色温曲线的右侧宽度做限制。 取值范围为[0x100, 0xFFF]，取值越大，可支持的光源范围越宽，白平衡精度稍有下降。

【注意事项】

无。

【相关数据类型及接口】

无。

ISP_AWB_LIGHTSOURCE_INFO_S

【说明】

定义单个光源点的属性。

【定义】

```
typedef struct hiISP_AWB_LIGHTSOURCE_INFO_S
{
    HI_U16 u16WhiteRgain;
    HI_U16 u16WhiteBgain;
```



```
HI_U16 u16ExpQuant;  
HI_BOOL bLightStatus;  
} ISP_AWB_LIGHTSOURCE_INFO_S;
```

【成员】

成员名称	描述
u16WhiteRgain	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。
u16WhiteBgain	在该光源下捕获 ColorChecker Raw 数据，通过 Calibration Tool Staitic WB 选项校准得到的 Bgain。
u16ExpQuant	光源亮度信息，暂不支持。
bLightStatus	光源点状态。 0：未使能； 1：使能。

【注意事项】

无。

【相关数据类型及接口】

无。

ISP_AWB_ADD_LIGHTSOURCE_S

【说明】

定义独立光源点属性。

【定义】

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

【成员】

成员名称	描述
bLightEnable	使能独立光源点校正开关。 某些特殊光源，比如 CWF 等，白点落在预校准的色温曲线外，增加独立光源点可优化白平衡在该光源下的表现。



成员名称	描述
stLightInfo[LIGHTSOURCE_NUM]	独立光源点信息。

【注意事项】

#define LIGHTSOURCE_NUM 4

【相关数据类型及接口】

[ISP_AWB_LIGHTSOURCE_INFO_S](#)



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，使头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

- [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_SetSaturation](#): 设置颜色饱和度属性。
- [HI_MPI_ISP_GetSaturation](#): 获取颜色饱和度属性。
- [HI_MPI_ISP_SetCCM](#): 设置颜色校正基础矩阵。
- [HI_MPI_ISP_GetCCM](#): 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(const ISP_SATURATION_ATTR_S  
*pstSatAttr);
```

【参数】

参数名称	描述	输入/输出
ISP_SATURATION_ATTR_S	颜色饱和度属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_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\_SATURATION\_ATTR\_S *pstSatAttr);
```

【参数】

参数名称	描述	输入/输出
ISP_SATURATION_ATTR_S	颜色饱和度属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_SetSaturationAttr](#)



HI_MPI_ISP_SetSaturation

【描述】

设置颜色饱和度期望值。

【语法】

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

【参数】

参数名称	描述	输入/输出
u8Value	颜色饱和度期望值。	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】

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_GetSaturation

【描述】

获取颜色饱和度期望值。

【语法】

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

【参数】



参数名称	描述	输入/输出
pu32Value	颜色饱和度期望值。	输出

【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。

【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_SetCCM

【描述】

设置颜色矩阵。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstColorMatrix	颜色矩阵。	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。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.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。

【举例】

无。

【相关主题】

- [HI_MPI_ISP_GetCCM](#)

HI_MPI_ISP_GetCCM

【描述】

获取颜色矩阵。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstColorMatrix	颜色矩阵。	输出



【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。

【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_SetCCM](#)

5.5 数据类型

- [ISP_SATURATION_ATTR_S](#)：定义 ISP 颜色饱和度属性。
- [ISP_COLORMATRIX_S](#)：定义 ISP 颜色矩阵属性。

ISP_SATURATION_ATTR_S

【说明】

定义 ISP 颜色饱和度属性。

【定义】

```
typedef struct hiISP_SATURATION_ATTR_S
{
    HI_BOOL bSatManualEnable;
    HI_U8   u8SatTarget;
```



```
HI_U8    au8Sat[8];  
}ISP_SATURATION_ATTR_S;
```

【成员】

成员名称	描述
bSatManualEnable	手动颜色饱和度使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u8SatTarget	饱和度强度期望值。 取值范围: [0x00, 0xFF]。 默认值为 0x80。
au8Sat[8]	设置图像饱和度, 该数组的八个值分别对应 sensor 在不同增益情况下的饱和度值, 一般情况下增益越大, 设置的饱和度值越小。对应关系如表 5-1 所示。

表5-1 au8Sat[8]在不同的增益情况下的设置值

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

【注意事项】

- 设置饱和度功能分为自动和手动:
 - bSatManualEnable 为 HI_FALSE, 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 [8]的描述。变量 u8SatTarget 表示饱和度期望值, 实际饱和度值 = 根据增益自动调节后饱和度 * u8SatTarget / 0x80。
 - bSatManualEnable 为 HI_TRUE, 使用手动饱和度调节功能。
此时实际饱和度值等于期望值(u8SatTarget), 变量 u8Sat[8]无效。



【相关数据类型及接口】

无

ISP_COLORMATRIX_S

【说明】

定义 ISP 颜色矩阵属性。

【定义】

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

【成员】

成员名称	描述
u16HighColorTemp	高色温。 取值范围：[2000,10000]。
au16HighCCM[9]	高色温下的颜色校正矩阵。 取值范围：[0x0,0xFFFF]。
u16MidColorTemp	中等色温。取值范围：[2000,u16HighColorTemp-400]
au16MidCCM[9]	中等色温下的颜色校正矩阵。 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。 取值范围：[0x0,0xFFFF]。

【注意事项】

- 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：
 - $u16HighColorTemp - u16MidColorTemp \geq 400$
 - $u16MidColorTemp - u16LowColorTemp \geq 400$

【相关数据类型及接口】



无。



6 IMP

6.1 Sharpen

6.1.1 功能描述

Sharpen 模块用于调节图像边缘锐化属性，Sharpen 强度同时控制图像水平和垂直方向边缘增加的强度。

6.1.2 API 参考

- [HI_MPI_ISP_SetSharpenAttr](#): 设置边缘锐化属性。
- [HI_MPI_ISP_GetSharpenAttr](#): 获取边缘锐化属性。

HI_MPI_ISP_SetSharpenAttr

【描述】

设定边缘锐化属性。

【语法】

```
HI_MPI_ISP_SetSharpenAttr(const ISP_SHARPEN_ATTR_S *pstSharpenAttr);
```

【参数】

参数名称	描述	输入/输出
pstSharpenAttr	边缘锐化属性	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】

[HI_MPI_ISP_GetSharpenAttr](#)

HI_MPI_ISP_GetSharpenAttr

【描述】

获取边缘锐化属性。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstSharpenAttr	边缘锐化属性	输出

【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】

[HI_MPI_ISP_SetSharpenAttr](#)

6.1.3 数据类型

[ISP_SHARPEN_ATTR_S](#)：定义 ISP Sharpen 属性。

ISP_SHARPEN_ATTR_S

【说明】

定义 ISP Sharpen 属性。

【定义】

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

【成员】

成员名称	描述
bEnable	Sharpen 增强功能使能。



成员名称	描述
	HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_TRUE。
bManualEnable	手动 Sharpen 增强功能使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u32StrengthTarget	手动使能时, Sharpen 增强的强度期望值。 取值范围: [0x00, 0xFF]。 默认值为 0x80。
u8StrengthMin	Sharpen 增强的强度最小值。 取值范围: [0, 0xFF]。 默认值为 0x28。
u8SharpenAltD[8]	设置图像大边缘的锐度, 该数组的八个值分别对应 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。
u8SharpenAltUd[8]	设置图像小纹理的锐度, 该数组的八个值分别对应的 sensor 在不同的增益情况下不同的设置值, 具体对应关系如表 6-2 所示。

表6-1 u8SharpenAltD[8]在不同的增益情况下的设置值

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



表6-2 u8SharpenAltUd 在不同的增益情况下的设置值

u8SharpenAltUd	Again*Dgian*IspDgain(times)
u8SharpenAltUd [0]	1
u8SharpenAltUd [1]	2
u8SharpenAltUd [2]	4
u8SharpenAltUd [3]	8
u8SharpenAltUd [4]	16
u8SharpenAltUd [5]	32
u8SharpenAltUd [6]	64
u8SharpenAltUd [7]	128

【注意事项】

- Sharpen 功能开启后，u32StrengthTarget 的值越大，手动使能时 Sharpen 增强的强度越大。
- Sharpen 功能分为自动和手动：
 - bEnable 为 HI_TRUE，bManualEnable 为 HI_FALSE，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 u8SharpenAltD[8]和 u8SharpenAltUd[8]的描述。
 - bEnable 和 bManualEnable 均设置为 HI_TRUE，使用手动 Sharpen 功能。
- 根据 sensor 增益，Sharpen 强度会在期望值和最小值之间自动调节。
- 使能手动 Sharpen 增强功能时，实际 Sharpen 强度等于期望值(u32StrengthTarget)。

【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，对 Gamma 表之间的图像像素使用线性插值生成。

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_SetGammaTable](#): 设置 Gamma 表属性。
- [HI_MPI_ISP_GetGammaTable](#): 获取 Gamma 表属性。



HI_MPI_ISP_SetGammaAttr

【描述】

设定 Gamma 属性。

【语法】

```
HI_MPI_ISP_SetGammaAttr(const ISP_GAMMA_ATTR_S * pstGammaAttr);
```

【参数】

参数名称	描述	输入/输出
pstGammaAttr	Gamma 属性	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。

【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_GetGammaAttr](#)

HI_MPI_ISP_GetGammaAttr

【描述】



获取 Gamma 属性。

【语法】

```
HI_MPI_ISP_GetGammaAttr( ISP\_GAMMA\_ATTR\_S * pstGammaAttr );
```

【参数】

参数名称	描述	输入/输出
pstGammaAttr	Gamma 属性	输出

【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。

【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](#)

HI_MPI_ISP_SetGammaTable

【描述】

设定 Gamma 表属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetGammaTable(const ISP\_GAMMA\_TABLE\_S * pstGammaTable);
```



【参数】

参数名称	描述	输入/输出
pstGammaTable	Gamma 表属性	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】

- 设置 Gamma 表前，必须先设定 Gamma 属性。
- 用户选择 ISP 内部支持的 1.6、1.8、2.0、2.2 sRGB 标准曲线及默认 Gamma 曲线时，不需要设置 u16Gamma。
- 若用户自定义 Gamma 曲线，必须设置 u16Gamma。

【举例】

无。

【相关主题】

- [HI_MPI_ISP_SetGammaAttr](#)
- [HI_MPI_ISP_GetGammaAttr](#)

HI_MPI_ISP_GetGammaTable

【描述】

获取 Gamma 表属性。



【语法】

```
HI_S32 HI_MPI_ISP_GetGammaTable(ISP_GAMMA_TABLE_S * pstGammaTable);
```

【参数】

参数名称	描述	输入/输出
pstGammaTable	Gamma 表属性	输出

【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_SUCCESS	成功。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

仅可以读取 Gamma 表，ISP_GAMMA_CURVE_E 无法读取。

【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_E](#)：定义 ISP Gamma 曲线类型。
- [ISP_GAMMA_TABLE_S](#)：定义 ISP Gamma 表属性。



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_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。

【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】

无。

ISP_GAMMA_CURVE_E

【说明】

定义 ISP Gamma 曲线类型。

【定义】

```
typedef enum hiISP_GAMMA_CURVE_E
{
    ISP_GAMMA_CURVE_1_6 = 0x0,
    ISP_GAMMA_CURVE_1_8 = 0x1,
    ISP_GAMMA_CURVE_2_0 = 0x2,
    ISP_GAMMA_CURVE_2_2 = 0x3,
    ISP_GAMMA_CURVE_DEFAULT = 0x4,
    ISP_GAMMA_CURVE_SRGB = 0x5,
    ISP_GAMMA_CURVE_USER_DEFINE = 0x6,
    ISP_GAMMA_CURVE_BUTT
} ISP_GAMMA_CURVE_E;
```



【成员】

成员名称	描述
ISP_GAMMA_CURVE_1_6	1.6 sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_1_8	1.8 sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_2_0	2.0 sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_2_2	2.2 sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_SRGB	sRGB 标准 Gamma 曲线。
ISP_GAMMA_CURVE_DEFAULT	默认 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。

【相关数据类型及接口】

无。

ISP_GAMMA_TABLE_S

【说明】

定义 ISP Gamma 表属性。

【定义】

```
typedef struct hiISP_GAMMA_TABLE_S
{
    ISP_GAMMA_CURVE_E enGammaCurve;
    HI_U16 u16Gamma[GAMMA_LUT_ SIZE];
    HI_U16 u16Gamma_FE[GAMMA_FE_LUT_ SIZE];
} ISP_GAMMA_TABLE_S;
```

【成员】

成员名称	描述
enGammaCurve	Gamma 曲线选择。 默认值为 ISP_GAMMA_CURVE_DEFAULT。
u16Gamma[GAMMA_LUT_ SIZE]	Gamma 表。 Hi3516 取值范围：[0, 0xFFFF] Hi3518 取值范围：[0, 0xFFF]



成员名称	描述
u16Gamma_FE[GAMMA_FE_LUT_SIZE]	WDR sensor 的 Gamma_fe 表。 Hi3516 取值范围: [0, 0xFFFF] Hi3518 取值范围: [0, 0xFFF]

【差异说明】

芯片类型	GAMMA_LUT_SIZE 值	GAMMA_FE_LUT_SIZE 值
Hi3516	65	129
Hi3518	257	257

【注意事项】

- 用户自定义 Gamma 曲线时，必须确保 Gamma 表配置正确。
- 设置 Gamma 表，使用 [HI_MPI_ISP_SetGammaTable](#) 接口，不需要关注变量 u16Gamma_FE；WDR sensor 设置 Gamma_fe 表，使用 [HI_MPI_ISP_SetGammaFETable](#) 接口，不需要关注变量 enGammaCurve 和 u16Gamma。

【相关数据类型及接口】

[ISP_GAMMA_CURVE_E](#)

6.3 DRC

6.3.1 功能描述

Dynamic Range Compression，即动态范围压缩，目的是调整图像的动态范围，使得图像显示出更多的信息。DRC 模块是一个基于人眼视觉系统特性的高级局部色阶映射（多空间动态范围压缩）引擎。

6.3.2 API 参考

- [HI_MPI_ISP_SetDRCAAttr](#): 设定 DRC 属性。
- [HI_MPI_ISP_GetDRCAAttr](#): 获取 DRC 属性。

HI_MPI_ISP_SetDRCAAttr

【描述】

设定 DRC 属性。

【语法】



```
HI_S32 HI_MPI_ISP_SetDRCAAttr(const ISP_DRC_ATTR_S *pstDRCAAttr);
```

【参数】

参数名称	描述	输入/输出
pstDRCAAttr	DRC 属性	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。

【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_GetDRCAAttr](#)

HI_MPI_ISP_GetDRCAAttr

【描述】

获取 DRC 属性。

【语法】

```
HI_MPI_ISP_GetDRCAAttr(ISP_DRC_ATTR_S *pstDRCAAttr);
```

【参数】



参数名称	描述	输入/输出
pstDRCAAttr	DRC 属性	输出

【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。

【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_SetDRCAAttr](#)

6.3.3 数据类型

[ISP_DRC_ATTR_S](#)：定义 DRC 属性。

ISP_DRC_ATTR_S

【说明】

定义 DRC 属性。

【定义】

```
typedef struct hiISP_DENOISE_ATTR_S
{
    HI_BOOL bDRCEnable;
```



```
HI_BOOL bDRCTManualEnable;  
HI_U32 u32StrengthTarget;  
HI_U32 u32SlopeMax;  
HI_U32 u32SlopeMin;  
HI_U32 u32WhiteLevel;  
HI_U32 u32BlackLevel;  
HI_U32 u32VarianceSpace;  
HI_U32 u32VarianceIntensity;  
} ISP_DENOISE_ATTR_S;
```

【成员】

成员名称	描述
bDRCTEnable	HI_FALSE: 关闭 DRC 功能; HI_TRUE: 使能 DRC 功能。 普通 sensor 默认值为 HI_FALSE; WDR sensor 默认值为 HI_TRUE。
bDRCTManualEnable	FALSE: 关闭手动 DRC 功能; HI_TRUE: 使能手动 DRC 功能。 默认值为 HI_FALSE。
u32StrengthTarget	DRC 强度。 取值范围: [0, 0xFF]。 默认值为 0x80。
u32SlopeMax	DRC 增强控制参数, 用于控制增强当前像素点的曲线的斜率的最大值。 取值范围: [0, 0xFF]。 默认值由文件 cmos.c 内结构体 cmos_isp_default_t 的成员 iridix_sm 设定。 推荐值: [0x20, 0x60]。
u32SlopeMin	DRC 增强控制参数, 用于控制增强当前像素点的曲线的斜率的最小值。 取值范围: [0, 0xFF]。 普通 sensor 默认值为 0x40; WDR sensor 默认值为 0x10。 推荐值: [0x00, 0x30]。
u32WhiteLevel	DRC 增强的最大像素值(pixel value), 即 bayer 数据域的像素值, 高于 u32Whitelevel 的像素将不能参与 DRC 算法。 取值范围: [0, 0xFFF]。 默认值由文件 cmos.c 内结构体 cmos_isp_default_t 的成员



成员名称	描述
	iridix_wl 设定。
u32BlackLevel	DRC 增强的最小像素值(pixel value)，即 bayer 数据域的像素值，低于 u32Blacklevel 的像素将不能参与 DRC 算法。 取值范围：[0, 0xFF]。 默认值为 0。
u32VarianceSpace	该值越大，表示生成 tone curves 的曲线时包含的像素的窗口越大。 取值范围：[0x0, 0xF]。 默认值为 2。
u32VarianceIntensity	该值越大，表示每个像素点与周围像素点的 DRC 强度值变化越平滑。 取值范围：[0x0, 0xF]。 默认值为 1。

【注意事项】

- DRC 开启后，u32StrengthTarget 值越大，暗区越亮，噪声也会随之增大。
- 使能手动 DRC 功能时，实际 DRC 强度等于期望值（u32StrengthTarget）。
- 其他 DRC 参数属于高级参数，不建议修改。

【相关数据类型及接口】

无。

6.4 镜头阴影校正

6.4.1 概述

镜头的物理结构决定了 sensor 的中心比外围能接收到更多的光，相对中心来说外围就是阴影，这个现象叫做渐晕（vignetting）。镜头阴影校正就是用来对图像出现的暗角进行补偿校正。进行亮度校正时，R、G、B 三分量可以使用同样的参数；进行颜色校正时则使用各自单独的校正参数。

6.4.2 功能描述

6.4.2.1 Hi3516

使用 Mesh correction，将整幅图像分成多个区间（默认为 64*64），每个区间有一个 8bit 的校正系数（即增益）。对图像中的每一个像素，使用双线性插值生成实际的增



益。校正系数的类型通过全局参数 Mesh_Scale 定义。校正系数由 calibration 离线校正工具生成。Mesh_Scale 定义如表 6-3 所示。

表6-3 Mesh_Scale 定义

Mesh Scale	校正系数格式	最大增益
0	无符号小数，定点 1.7	X2
1	无符号小数，定点 2.6	X4
2	无符号小数，定点 3.5	X8
3	无符号小数，定点 4.4	X16

6.4.2.2 Hi3518

使用 Radial correction，对 R、G、B 三分量，分别设置中心点和校正系数。校正系数描述了以同心环状从中心点到最远角落处的增益，由大小为 129 的查找表表示，数据格式为无符号小数，定点 4.12，理论上能表示最大 X16 的增益。校正系数由 calibration 离线校正工具生成。

6.4.3 API 参考

- [HI_MPI_ISP_SetShadingAttr](#): 设置暗角校正属性。
- [HI_MPI_ISP_GetShadingAttr](#): 获取暗角校正属性。
- [HI_MPI_ISP_SetShadingTable](#): 设置镜头暗角补偿查找表。
- [HI_MPI_ISP_GetShadingTable](#): 获取镜头暗角补偿查找表。

HI_MPI_ISP_SetShadingAttr

【描述】

设定暗角补偿属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetShadingAttr(const ISP_SHADING_ATTR_S pstShadingAttr);
```

【参数】

参数名称	描述	输入/输出
pstShadingAttr	暗角补偿属性	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_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】

用于使能 lens shading 校正功能。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetShadingAttr](#)

HI_MPI_ISP_GetShadingAttr

【描述】

获取暗角补偿属性。

【语法】

```
HI_MPI_ISP_GetShadingAttr(ISP\_SHADING\_ATTR\_S *pstShadingAttr);
```

【参数】

参数名称	描述	输入/输出
pstShadingAttr	暗角补偿属性	输出

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【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_SetShadingAttr](#)

HI_MPI_ISP_SetShadingTable

【描述】

设定镜头暗角补偿查找表属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetShadingTable(const ISP\_SHADINGTAB\_S *pstShadingTab);
```

【参数】

参数名称	描述	输入/输出
pstShadingTab	暗角补偿查找表属性	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】

设置镜头暗角补偿查找表。Hi3518 仅设置查找表中前 u16ShadingTableNodeNumber 个值。

【举例】

无。

【相关主题】

- [HI_MPI_ISP_GetShadingTable](#)
- [HI_MPI_ISP_GetShadingAttr](#)

HI_MPI_ISP_GetShadingTable

【描述】

获取镜头暗角补偿查找表属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetShadingTable(ISP\_SHADINGTAB\_S *pstShadingTab);
```

【参数】

参数名称	描述	输入/输出
pstShadingTab	暗角补偿查找表属性	输出

【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。

【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_SetShadingTable](#)
- [HI_MPI_ISP_SetShadingAttr](#)

6.4.4 数据类型

6.4.4.1 Hi3516

- [ISP_SHADING_SCALE_E](#)：定义 ISP lens shading 校正值格式。
- [ISP_SHADING_TAB_E](#)：定义 ISP lens shading 表类型。
- [ISP_SHADING_ATTR_S](#)：定义 ISP lens shading 校正属性。
- [ISP_SHADINGTAB_S](#)：定义 ISP lens shading 表属性。

ISP_SHADING_SCALE_E

【说明】

定义 ISP lens shading 校正值格式。

【定义】

```
typedef enum hiISP_SHADING_SCALE_E
{
    ISP_SHADING_SCALE_2 = 0x0,
    ISP_SHADING_SCALE_4 = 0x1,
    ISP_SHADING_SCALE_8 = 0x2,
    ISP_SHADING_SCALE_16 = 0x3,
    ISP_SHADING_SCALE_BUTT
}ISP_SHADING_SCALE_E;
```



【成员】

成员名称	描述
ISP_SHADING_SCALE_2	lens shading 校正值的格式为定点 1.7。
ISP_SHADING_SCALE_4	lens shading 校正值的格式为定点 2.6。
ISP_SHADING_SCALE_8	lens shading 校正值的格式为定点 3.5。
ISP_SHADING_SCALE_16	lens shading 校正值的格式为定点 4.4。

【注意事项】

无。

【相关数据类型及接口】

无。

ISP_SHADING_TAB_E

【说明】

定义 ISP lens shading 表类型。

【定义】

```
typedef enum hiISP_SHADING_TAB_E
{
    SHADING_TAB_R        = 0,
    SHADING_TAB_G        = 1,
    SHADING_TAB_B        = 2,
    SHADING_TAB_BUTT
} ISP_SHADING_TAB_E;
```

【成员】

成员名称	描述
SHADING_TAB_R	lens shading 表 R。
SHADING_TAB_G	lens shading 表 G。
SHADING_TAB_B	lens shading 表 B。

【注意事项】

无。

【相关数据类型及接口】



无。

ISP_SHADING_ATTR_S

【说明】

定义 ISP lens shading 校正属性。

【定义】

```
typedef struct hiISP_SHADING_ATTR_S
{
    HI_BOOL Enable;
} ISP_SHADING_ATTR_S;
```

【成员】

成员名称	描述
bEnable	HI_FALSE: 关闭 lens shading 校正功能; HI_TRUE: 使能 lens shading 校正功能。 默认值为 HI_FALSE。

【注意事项】

Lens shading 校正可分别调用各自的 shading table。

【相关数据类型及接口】

无。

ISP_SHADINGTAB_S

【说明】

定义 ISP lens shading 表属性。

【定义】

```
typedef struct hiISP_SHADINGTAB_S
{
    ISP_SHADING_SCALE_E enScale;
    ISP_SHADING_TAB_E enMesh_R;
    ISP_SHADING_TAB_E enMesh_G;
    ISP_SHADING_TAB_E enMesh_B;
    HI_U8 u8ShadingTable_R[64*64];
    HI_U8 u8ShadingTable_G[64*64];
    HI_U8 u8ShadingTable_B[64*64];
} ISP_SHADINGTAB_S;
```



【成员】

成员名称	描述
enScale	lens shading 校正值格式。
enMesh_R	图像红色分量选择的 lens shading 表类型。
enMesh_G	图像绿色分量选择的 lens shading 表类型。
enMesh_B	图像蓝色分量选择的 lens shading 表类型。
u8ShadingTable_R[64*64]	lens shading 表 R。
u8ShadingTable_G[64*64]	lens shading 表 G。
u8ShadingTable_B[64*64]	lens shading 表 B。

【注意事项】

- enScale 控制 lens shading 表 R、G 和 B 三张表的数据值格式。
- 通常情况下，只需要配置一套 lens shading 表（例如 u8ShadingTable_R[64*64]），并设置 enMesh_R、enMesh_G、enMesh_B（例如取值 SHADING_TAB_R）使三个颜色分量均选择这套表，即可实现 lens shading correction 功能。
- 某些情况下（例如选择较差的镜头）需要使用 color shading correction 功能，则需要配置三套 lens shading 表，并让 enMesh_R、enMesh_G、enMesh_B 选择对应的 lens shading 表。使用 color shading correction 功能可改善图像颜色的不一致性。

【相关数据类型及接口】

- [ISP_SHADING_SCALE_E](#)
- [ISP_SHADING_TAB_E](#)

6.4.4.2 Hi3518

- [ISP_SHADING_ATTR_S](#)：定义 ISP lens shading 校正属性。
- [ISP_SHADINGTAB_S](#)：定义 ISP lens shading 表属性。

ISP_SHADING_ATTR_S

【说明】

定义 ISP lens shading 校正属性。

【定义】

```
typedef struct hiISP_SHADING_ATTR_S
{
    HI_BOOL Enable;
} ISP_SHADING_ATTR_S;
```

【成员】



成员名称	描述
bEnable	HI_FALSE: 关闭 lens shading 校正功能; HI_TRUE: 使能 lens shading 校正功能。 默认值为 HI_FALSE。

【注意事项】

Lens shading 校正可分别调用各自的 shading table。

【相关数据类型及接口】

无。

ISP_SHADINGTAB_S

【说明】

定义 ISP lens shading 校正属性。

【定义】

```
typedef struct hiISP_SHADINGTAB_S
{
    HI_U16 u16ShadingCenterR_X;
    HI_U16 u16ShadingCenterR_Y;
    HI_U16 u16ShadingCenterG_X;
    HI_U16 u16ShadingCenterG_Y;
    HI_U16 u16ShadingCenterB_X;
    HI_U16 u16ShadingCenterB_Y;

    HI_U16 u16ShadingTable_R[129];
    HI_U16 u16ShadingTable_G[129];
    HI_U16 u16ShadingTable_B[129];

    HI_U16 u16ShadingOffCenter_R;
    HI_U16 u16ShadingOffCenter_G;
    HI_U16 u16ShadingOffCenter_B;

    HI_U16 u16ShadingTableNodeNumber;
} ISP_SHADINGTAB_S;
```

【成员】

成员名称	描述
u16ShadingCenterR_X	R 分量中心点的 X 轴坐标。



成员名称	描述
	取值范围为[0x0, 0xFFFF]
u16ShadingCenterR_Y	R 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
u16ShadingCenterG_X	G 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
u16ShadingCenterG_Y	G 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
u16ShadingCenterB_X	B 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
u16ShadingCenterB_Y	B 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
u16ShadingTable_R	R 分量的校正表。 取值范围为[0x0, 0xFFFF]
u16ShadingTable_G	G 分量的校正表。 取值范围为[0x0, 0xFFFF]
u16ShadingTable_B	B 分量的校正表。 取值范围为[0x0, 0xFFFF]
u16ShadingOffCenter_R	R 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
u16ShadingOffCenter_G	G 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
u16ShadingOffCenter_B	B 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
u16ShadingTableNodeNumber	每个分量校正表中使用到的节点个数。 取值范围为[0x0, 0x81]，默认值为 0x81。

【注意事项】

图像左上角为原点，水平方向为 X 轴，垂直方向为 Y 轴，单位为像素。

【相关数据类型及接口】

无。



6.5 Defect Pixel

6.5.1 概述

坏点校正模块通过使用内部坏点校正逻辑完成坏点检测，校正过程完成之后，获取到了坏点坐标，通常的做法是将坏点坐标存在 Flash 中，以后断电重新启动之后加载坏点坐标，无须重新校正。

6.5.2 功能描述

Hi3518 坏点校正算法实现主要是通过一个窗口为 5x5 的模块找出该窗口内像素明显异于临近像素的坏点。该模块的实现了以下两种模式：

- 静态坏点检测/校正

在这个模式，光圈处于关闭状态，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。

- 动态坏点检测/校正

在这种模式中，校正模块在动态的进行坏点检测和校正，能够校正的坏点个数没有限制。一个中值滤波器被用于坏点检测模块，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时使用动态坏点校正功能。

6.5.3 API 参考

- [HI_MPI_ISP_SetDefectPixelAttr](#)：设置坏点校正属性。
- [HI_MPI_ISP_GetDefectPixelAttr](#)：获取坏点校正属性。

HI_MPI_ISP_SetDefectPixelAttr

【描述】

设置坏点校正属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetDefectPixelAttr( ISP_DP_ATTR_S *pstDPAttr );
```

【参数】

参数名称	描述	输入/输出
pstDPAttr	坏点校正属性	输入

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

该接口分为坏点校正和坏点检测两大类功能，坏点校正使能之后 ISP 会自动每帧对点进行处理，坏点检测只需启动一次，再获取坏点坐标后便可以关闭此功能。坏点检测的硬件环境为：先遮黑镜头或关闭快门，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 左右。

【举例】

```
ISP_DP_ATTR_S stDPAttr;  
HI_U16 i;  
HI_U32 u32Table[1024] = {0};  
  
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);  
stDPAttr.bEnableStatic = HI_TRUE;  
stDPAttr.bEnableDynamic = HI_TRUE;  
stDPAttr.bEnableDetect = HI_TRUE;  
stDPAttr.ul6BadPixelCountMax = 0x200;  
stDPAttr.ul6BadPixelCountMin = 0x40;  
  
for(i=0; i< 1024;i++)  
{  
    stDPAttr.u32BadPixelTable[i] = 0;  
}  
HI_MPI_ISP_SetDefectPixelAttr(&stDPAttr);  
PAUSE;  
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);  
PAUSE;
```

【相关主题】

[HI_MPI_ISP_GetDefectPixelAttr](#)

HI_MPI_ISP_GetDefectPixelAttr

【描述】

获取坏点校正属性。

【语法】



```
HI_S32 HI_MPI_ISP_GetDefectPixelAttr( ISP_DP_ATTR_S *pstDPAAttr);
```

【参数】

参数名称	描述	输入/输出
pstDPAAttr	坏点校正属性	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】

无。

【举例】

```
{
ISP_DP_ATTR_S  stDPAAttr;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAAttr);
stDPAAttr.bEnableStatic = HI_TRUE;
stDPAAttr.bEnableDynamic= HI_FALSE;
stDPAAttr.bEnableDetect = HI_TRUE;
stDPAAttr.ul6BadPixelCountMax = 0x200;
stDPAAttr.ul6BadPixelCountMin = 0x40;
for(i=0; i< 1024;i++)
{
stDPAAttr.u32BadPixelTable[i] = 0;
}

HI_MPI_ISP_SetDefectPixelAttr(&stDPAAttr);
PAUSE;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAAttr);
PAUSE;

}
```



【相关主题】

无。

6.5.4 数据类型

- [ISP_DP_ATTR_S](#): 定义 ISP 坏点校正属性。
- [ISP_TRIGGER_STATUS_E](#): 定义 ISP 校正（检测）状态。

ISP_DP_ATTR_S

【说明】

定义 ISP 坏点校正属性。

【定义】

```
typedef struct hiISP_DP_ATTR_S
{
    HI_BOOL bEnableDynamic;
    HI_U16 u16DynamicBadPixelSlope;
    HI_U16 u16DynamicBadPixelThresh;
    HI_BOOL bEnableStatic;

    HI_BOOL bEnableDetect;
    ISP_TRIGGER_STATUS_E enTriggerStatus;
    HI_U8 u8BadPixelStartThresh;
    HI_U8 u8BadPixelFinishThresh;
    HI_U16 u16BadPixelCountMax;
    HI_U16 u16BadPixelCountMin;
    HI_U16 u16BadPixelCount;
    HI_U16 u16BadPixelTriggerTime;
    HI_U32 u32BadPixelTable[1024];
} ISP_DP_ATTR_S;
```

【成员】

成员名称	描述
bEnableDynamic	使能动态坏点校正功能。
u16DynamicBadPixelSlope	动态坏点校正的强度 取值范围：[0, 0xFFFF]
u16DynamicBadPixelThresh	检测动态坏点的门限值 取值范围：[0, 0xFFFF]
bEnableStatic	使能静态坏点校正功能。
bEnableDetect	使能静态坏点检测功能。



成员名称	描述
enTriggerStatus	静态坏点检测结果状态信息。
u8BadPixelStartThresh	静态坏点检测开始时的门限值。
u8BadPixelFinishThresh	静态坏点检测结束时的门限值。
u16BadPixelCountMax	允许静态坏点的最大个数。 取值范围：[0, 0x3FF]
u16BadPixelCountMin	允许静态坏点的最小个数。 取值范围：[0, 0x3FF]
u16BadPixelCount	静态坏点个数。 取值范围：[0, 0x3FF]
u16BadPixelTriggerTime	静态坏点检测（校正）的超时时间，以帧数为单位。 取值范围：[0, 0x640]
u32BadPixelTable[1024]	坏点坐标值，前 22bit 有效，低 11bit 为 X 坐标值， 12~22bit 为 Y 坐标值。

【注意事项】

- ISP 坏点检测算法检测成功的标准：检测出的坏点数量是否 u16BadPixelCountMax 与 u16BadPixelCountMin 之间。所以不同类型的 sensor 在做坏点检测时需微调这两个值。
- u8BadPixelFinishThresh 只作为输出。针对同类型的 sensor，参考 u8BadPixelFinishThresh 值，设置合理的 u8BadPixelStartThresh，能加快静态坏点校正的速度。

【相关数据类型及接口】

[ISP_TRIGGER_STATUS_E](#)

ISP_TRIGGER_STATUS_E

【说明】

定义 ISP 校正（检测）状态。

【定义】

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



【成员】

成员名称	描述
ISP_TRIGGER_INIT	初始状态，未校正
ISP_TRIGGER_SUCCESS	校正成功结束
ISP_TRIGGER_TIMEOUT	校正超时结束

【注意事项】

无。

【相关数据类型及接口】

无。

6.6 CrossTalk Removal

6.6.1 概述

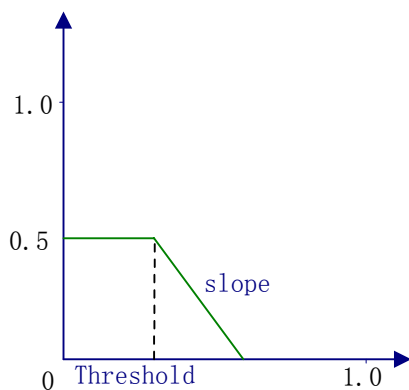
CrossTalk Removal 模块的主要功能是为了平衡 rawdata 之间临近像素 Gr 和 Gb 之间的差异，能够有效防止 demosaic 插值算法产生的方格或其他类似 pattern。由于 sensor 可能会因为特殊角度的光线入射而产生 CrossTalk，形成一些 pattern,根本原因就是因为在临近像素值之间 Gr 和 Gb 值域不一致。

6.6.2 功能描述

如图 6-1 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6-1 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(const ISP\_CR\_ATTR\_S *pstCRAAttr)
```

【参数】

参数名称	描述	输入/输出
pstCRAAttr	Crosstalk 属性	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。

【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(const ISP\_CR\_ATTR\_S *pstCRAAttr)
```

【参数】

参数名称	描述	输入/输出
pstCRAAttr	Crosstalk 属性	输出

【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。



【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_U8    u8Sensitivity;

    HI_U16   u16Threshold;

    HI_U16   u16Slope;

    HI_U8    u8Strength[8];
}ISP_CR_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能 CrossTalk remove 功能。
u8Sensitivity	设置 CrossRemove 敏感度值。
u16Threshold	设置 CrossRemove 门限值。 取值范围：[0, 0xFFF]
u16Slope	设置 CrossRemove 斜率值。 取值范围：[0, 0xFFF]



成员名称	描述
u8Strength	设置 CrossTalk remove 的强度值，该数组的八个值分别对应不同的 iso 值的强度。具体对应关系如表 6-4 所示。

表6-4 u8Strength 在不同的增益情况下的设置值

u8Strength	Again*Dgian*IspDgain (times)
u8Strength [0]	1
u8Strength [1]	2
u8Strength [2]	4
u8Strength [3]	8
u8Strength [4]	16
u8Strength [5]	32
u8Strength [6]	64
u8Strength [7]	128

【注意事项】

- u8Strength 值一般随着 iso 值的增大而减少。
- u8Sensitivity 值越大，表示绿色分量校正过程对边缘越不敏感。
- u16Threshold 值越大，表示整体处理的强度越大。
- u16Slope 值越大，表示处理强度随 Gr 与 Gb 之间的差值变化与越剧烈。

【相关数据类型及接口】

无。

6.7 Denoise

6.7.1 概述

去噪算法在空域内进行运算，处理 rawdata 数据，在去噪的同时保存边缘和纹理。

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，系统的模拟增益和数字增益同时增大，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_DENOISE_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_SetDenoiseAttr](#): 设定噪点抑制属性
- [HI_MPI_ISP_GetDenoiseAttr](#): 获取噪点抑制属性

HI_MPI_ISP_SetDenoiseAttr

【描述】

设定噪点抑制属性。

【语法】

```
HI_MPI_ISP_SetDenoiseAttr(const ISP\_DENOISE\_ATTR\_S *pstDenoiseAttr);
```

【参数】

参数名称	描述	输入/输出
pstDenoiseAttr	噪点抑制属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_GetDenoiseAttr](#)

HI_MPI_ISP_GetDenoiseAttr

【描述】

获取噪点抑制属性。

【语法】

```
HI_MPI_ISP_GetDenoiseAttr(const ISP\_DENOISE\_ATTR\_S *pstDenoiseAttr);
```

【参数】

参数名称	描述	输入/输出
pstDenoiseAttr	噪点抑制属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_SetDenoiseAttr](#)

6.7.4 数据类型

ISP_DENOISE_ATTR_S

【说明】

定义 ISP 噪点抑制属性。

【定义】

```
typedef struct hiISP_DENOISE_ATTR_S
{
    HI_BOOL bEnable;
    HI_BOOL bManualEnable;
    HI_U8 u8ThreshTarget;
    HI_U8 u8ThreshMax;
    HI_U8 u8SnrThresh[8]
} ISP_DENOISE_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能噪点抑制功能。
bManualEnable	使能手动噪点抑制功能。
u8ThreshTarget	手动使能时，噪点处理的目标门限值。
u8ThreshMax	噪点处理最大门限值。
u8SnrThresh[8]	设置图像去噪强度，该数组的八个值分别对应 sensor 在不同增益情况下的去噪强度值，一般情况下增益越大，设置的去噪强度值也越大。对应关系如表 6-5 所示。

表6-5 u8SnrThresh[8]在不同增益情况下对应的设置值

Snr_thresh	Again*Dgian*IspDgain(times)
u8SnrThresh [0]	1
u8SnrThresh [1]	2
u8SnrThresh [2]	4
u8SnrThresh [3]	8
u8SnrThresh [4]	16



Snr_thresh	Again*Dgian*IspDgain(times)
u8SnrThresh [5]	32
u8SnrThresh [6]	64
u8SnrThresh [7]	128

【注意事项】

- u8ThreshTarget 值越大，使能手动噪点抑制功能后对噪点的抑制越大。
- 使能手动噪点抑制功能时，实际噪点抑制强度等于期望值（u8ThreshTarget），噪点抑制。

【相关数据类型及接口】

无。

6.8 DIS

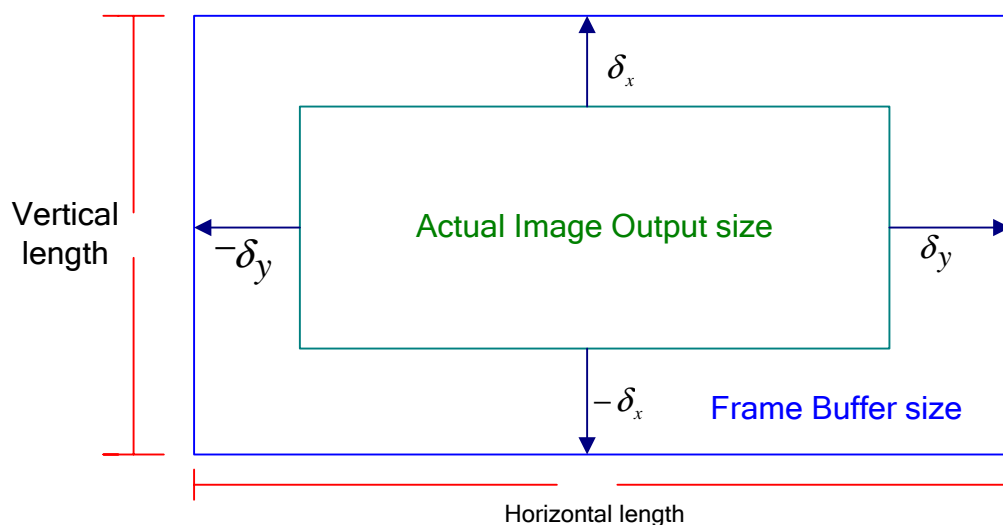
6.8.1 概述

DIS 模块比较当前的图像与前一帧图像来决定图像抖动或图像摇动的大小，这图像运动位移的值可以用来调整从 frame buffer 输出的图像，以提供稳定的图像输出。

6.8.2 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 δ_x 和垂直偏移量 δ_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：[-128, +128]，DIS 偏移示意如图 6-2 所示。

图6-2 DIS 偏移示意图



如图 6-2 所示，ISP 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 δ_x ， δ_y ，用户可以根据这两个偏移量的进行相应的开发以保证图像的稳定。

6.8.3 API 参考

- [HI_MPI_ISP_SetDISAttr](#): 设置 DIS 属性
- [HI_MPI_ISP_GetDISAttr](#): 获取 DIS 属性
- [HI_MPI_ISP_GetDISInfo](#): 获取 DIS 统计信息

HI_MPI_ISP_SetDISAttr

【描述】

设置 DIS 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetDISAttr(const ISP_DIS_ATTR_S *pstDISAttr);
```

【参数】

参数名称	描述	输入/输出
pstDISAttr	DIS 属性	输出

【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

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

无。

【举例】

请参见 [HI_MPI_ISP_GetDISInfo](#)。

【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\_DIS\_ATTR\_S *pstDISAttr);
```

【参数】

参数名称	描述	输入/输出
pstDISAttr	DIS 属性	输出

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【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_GetDISInfo

【描述】

获取 DIS 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetDISInfo(ISP_DIS_INFO_S*pstDISInfo);
```

【参数】

参数名称	描述	输入/输出
pstDISInfo	DIS 属性	输出

【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

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【注意】

无。

【举例】

```
HI_U32 ChnId;

    VI_DRV_CHN_STORE_INFO* pstStoreCfg;
    static RECT_S      stCapRect[VICAP_CHANNEL_NUM] = {{0, 0, 1280, 720}};
    static HI_U32      OFFSET_X,OFFSET_Y,Multiplier;
    HI_U32              OFFSET_Tmp_X,OFFSET_Tmp_Y;
    HI_U32              compare_value_x,compare_value_y;
    static HI_U32      Prepious_Value_X = 0;
    static HI_U32      Prepious_Value_Y = 0;
    ISP_DIS_INFO      pstDISInfo;
    ISP\_DIS\_ATTR\_S    pstDISAttr;
    pstDISAttr.

HI_MPI_ISP_SetDISAttr(const ISP\_DIS\_ATTR\_S *pstDISAttr);

HI_MPI_ISP_GetDISInfo(&pstDISInfo);
OFFSET_X = pstDISInfo.s8Xoffset;
OFFSET_Y = pstDISInfo.s8Yoffset;
if(1 == OFFSET_X%2)
    OFFSET_X = OFFSET_X -1;
if(1 == OFFSET_Y%2)
    OFFSET_Y = OFFSET_Y -1;
Multiplier = 1;
    OFFSET_Tmp_X = OFFSET_X;
OFFSET_Tmp_Y = OFFSET_Y;
    compare_value_x = (OFFSET_X > Prepious_Value_X)? (OFFSET_X -
Prepious_Value_X):(Prepious_Value_X - OFFSET_X);
    compare_value_y = (OFFSET_Y > Prepious_Value_Y)? (OFFSET_Y -
Prepious_Value_Y):(Prepious_Value_Y - OFFSET_Y);
```



```
        if(0x80 < OFFSET_X){

            if((compare_value_x<2)&&((0x100 - OFFSET_X) < 0xf))    /*****将差值小于
            2的并且幅度小于0xf的值滤除*****/
                OFFSET_X = 0x100;

            stCapRect[ChnId].s32X = 128 + (0x100 - OFFSET_X) * Multiplier;
            //输入给vi_crop的信息,    stCapRect[ChnId].s32X表示vi Crop帧的起始位置。

            stCapRect[ChnId].u32Width = 1280 + 128 + (0x100 - OFFSET_X) *
            Multiplier; //输入给vi_crop的信息,    stCapRect[ChnId].s32X表示vi Crop帧的起始位
            置。
        }
        else if(0x80 > OFFSET_X)
        {
            if((compare_value_x<2)&&(OFFSET_X < 0xf))
                OFFSET_X = 0;
            stCapRect[ChnId].s32X = 128 - OFFSET_X * Multiplier;
            stCapRect[ChnId].u32Width = 1280 + 128 - OFFSET_X * Multiplier;
        }
        else if(0x80 == OFFSET_X)
        {
            stCapRect[ChnId].s32X = 128;
            stCapRect[ChnId].u32Width = 1280 + 128;
        }

        if(0x80 < OFFSET_Y)
        {
            if((compare_value_y<2)&&((0x100 - OFFSET_Y) < 0x10))
                OFFSET_Y = 0x100;
            stCapRect[ChnId].s32Y = 128 + (0x100 - OFFSET_Y) * Multiplier;
            stCapRect[ChnId].u32Height = 720 + 128 + (0x100 - OFFSET_Y) *
            Multiplier;
        }
        if(0x80 > OFFSET_Y)
        {
            if((compare_value_y<2)&&(OFFSET_Y < 0x10))
                OFFSET_Y = 0;
            stCapRect[ChnId].s32Y = 128 - OFFSET_Y*Multiplier;
            stCapRect[ChnId].u32Height = 720 + 128 - OFFSET_Y * Multiplier;
        }
        else if(0x80 == OFFSET_Y)
        {

```



```
        stCapRect[ChnId].s32Y = 128;
        stCapRect[ChnId].u32Height = 720 + 128;
    }

    Prepious_Value_X = OFFSET_Tmp_X;
    Prepious_Value_Y = OFFSET_Tmp_Y;
    VI_DRV_SetChCrop(0, &stCapRect[ChnId]);
    pstStoreCfg->u32Width = stCapRect[ChnId].u32Width;
    pstStoreCfg->u32Height = stCapRect[ChnId].u32Height;
    VI_DRV_SetChDes(ChnId, pstStoreCfg);
    md_set_vdp_rect(HAL_DISP_LAYER_VSD1, 0, 0,
        stCapRect[ChnId].u32Width, stCapRect[ChnId].u32Height);
```

【相关主题】

无。

6.8.4 数据类型

- [ISP_DIS_ATTR_S](#): 定义 ISP DIS 属性。
- [ISP_DIS_INFO_S](#): 定义 ISP DIS 输出信息。

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。



ISP_DIS_INFO_S

【说明】

定义 ISP DIS 输出信息。

【定义】

```
typedef struct hiISP_DIS_INFO_S
{
    HI_S8 sXoffset;
    HI_S8 sYoffset;
} ISP_DIS_INFO_S;
```

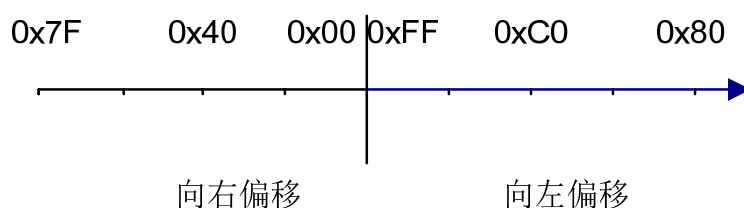
【成员】

成员名称	描述
sXoffset	DIS 模块输出的水平方向 Xoffset 取值范围：[0x00, 0xFF]
sYoffset	DIS 模块输出的水平方向 Yoffset 取值范围：[0x00, 0xFF]

【注意事项】

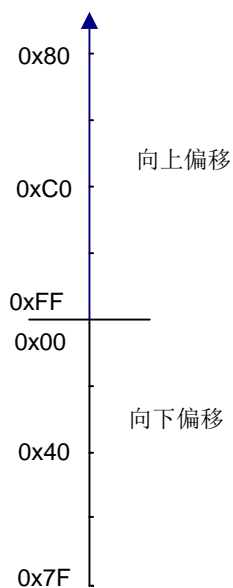
sXoffset 为有符号型，当图像水平向左移动时输出范围为[0x80, 0xFF]，输出以反码形式给出，即输出为 0xFF，表示图像水平向左移动了 1 个像素，输出为 0xFE，表示图像水平向左移动了 2 个像素；当图像水平向右移动时输出范围为[0x00, 0x80]，输出为 0x1 时，表示图像水平向右移动了 1 个像素，以此类推，具体如图 6-3 所示。

图6-3 DIS 水平偏移



sYoffset 为有符号型，当图像水平向上移动时输出范围为[0x80, 0xFF]，输出以反码形式给出，即当输出为 0xFF，表示图像水平向上移动了 1 个像素，当输出为 0xFE，表示图像水平向上移动了 2 个像素；当图像水平向下移动时输出范围为[0x00, 0x80]，输出为 0x1 时，表示图像水平向下移动了 1 个像素，以此类推，具体如图 6-4 所示。

图6-4 DIS 垂直偏移

**【相关数据类型及接口】**

无。

6.9 去雾

6.9.1 功能描述

Antifog 功能是通过动态的改变图象的对比度和亮度来实现的，该模块计算每帧图像的统计信息，估算雾的浓度，只有当图像中雾的浓度比较大且为整帧图像都有雾时，该算法才会起作用，否则算法模块认为当前不存在雾。

6.9.2 API 参考

- [HI_MPI_ISP_SetAntiFogAttr](#): 设置去雾属性
- [HI_MPI_ISP_GetAntiFogAttr](#): 获取去雾属性

HI_MPI_ISP_SetAntiFogAttr

【描述】

设置 Antifog 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAntiFogAttr(const ISP\_ANTIFOG\_S*pstAntiFog)
```

【参数】



参数名称	描述	输入/输出
pstAntiFog	去雾属性	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

【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_SetDenoiseAttr](#)

HI_MPI_ISP_GetAntiFogAttr

【描述】

获取 Antifog 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAntiFogAttr(ISP_ANTIFOG_S *pstAntiFog)
```

【参数】

参数名称	描述	输入/输出
pstAntiFog	去雾属性	输出



【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

【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_SetDenoiseAttr](#)

6.9.3 数据类型

ISP_ANTIFOG_S

【说明】

定义 ISP 去雾属性。

【定义】

```
typedef struct hiISP_ANTIFOG_S
{
    HI_BOOL bEnable;
    HI_U8  u8Strength;
} ISP_ANTIFOG_S;
```

【成员】



成员名称	描述
bEnable	使能 antifog 功能
u8Strength	去雾强度，取值范围[0,255]

【注意事项】

无。

【相关数据类型及接口】

无。

6.10 去伪彩

6.10.1 概述

去伪彩主要是指去除高频部分的摩尔纹效应。

6.10.2 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。

6.10.3 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】

设置去除伪彩属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAntiFalseColorAttr(const ISP_ANTI_FALSECOLOR_S
*pstAntiFC)
```

【参数】

参数名称	描述	输入/输出
pstAntiFC	去除伪彩	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。

【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】

获取去除伪彩属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAntiFalseColorAttr(const ISP\_ANTI\_FALSECOLOR\_S  
*pstAntiFC)
```

【参数】

参数名称	描述	输入/输出
pstAntiFC	去除伪彩	输出

【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。

【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。

6.10.4 数据类型

ISP_ANTI_FALSECOLOR_S

【说明】

定义 ISP 去伪彩属性。

【定义】

```
typedef struct hiISP_ANTI_FALSECOLOR_S
{
    HI_U8  u8Strength;
} ISP_ANTI_FALSECOLOR_S;
```

【成员】

成员名称	描述
u8Strength	去除伪彩强度值 取值范围：[0x0, 0x95]



【注意事项】

u8Strength 值为 0 时表示无去伪彩功能。

【相关数据类型及接口】

无。

6.11 去马赛克

6.11.1 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。

6.11.2 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_DEMOSAIC_ATTR_S *pstDemosaicAttr)
```

【参数】

参数名称	描述	输入/输出
pstDemosaicAttr	去马赛克属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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_GetDemosaicAttr

【描述】

获取去马赛克属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDemosaicAttr(ISP_DEMOSAIC_ATTR_S *pstDemosaicAttr)
```

【参数】

参数名称	描述	输入/输出
pstDemosaicAttr	去马赛克属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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。

6.11.3 数据类型

ISP_DEMOSAIC_ATTR_S

【说明】

定义 ISP 去马赛克属性。

【定义】

```
typedef struct hiISP_DEMOSAIC_ATTR_S
{
    HI_U8    u8VhSlope;    /*RW,Range: [0x0, 0xFF] */
    HI_U8    u8AaSlope;    /*RW,Range: [0x0, 0xFF] */
    HI_U8    u8VaSlope;    /*RW,Range: [0x0, 0xFF] */
    HI_U8    u8UuSlope;    /*RW,Range: [0x0, 0xFF] */
    HI_U16   u16VhThresh;  /*RW,Range: [0x0, 0xFFFF] */
    HI_U16   u16AaThresh;  /*RW,Range: [0x0, 0xFFFF] */
    HI_U16   u16VaThresh;  /*RW,Range: [0x0, 0xFFFF] */
    HI_U16   u16UuThresh;  /*RW,Range: [0x0, 0xFFFF] */
    HI_U8    u8DemosaicConfig; /*RW,Range: [0x0, 0xFF] */
    HI_U8    u8LumThresh[8]; /*RW, Range: [0x0, 0xFF] */
    HI_U8    u8NpOffset[8]; /*RW, Range: [0x0, 0xFF] */
}ISP_DEMOSAIC_ATTR_S;
```

【成员】

成员名称	描述
u8VhSlope	垂直、水平边缘混合阈值的斜率，调高此参数会提高水平、垂直解析度，加大噪声。 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员



成员名称	描述
	vh_slope 设定。
u8AaSlope	45°与 135°边缘混合阈值的斜率，调高此参数会提高斜解析度，加大噪声。 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 aa_slope 设定。
u8VaSlope	垂直、水平、45°与 135°边缘混合阈值的斜率，调高此参数会提高水平、垂直、斜解析度，加大噪声。 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 va_slope 设定。
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，且会引入伪彩。 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 uu_slope 设定。
u16VhThresh	垂直、水平边缘混合范围的阈值，调高此参数会降低伪彩、噪声与垂直、水平解析度。 取值范围：[0x0, 0xFFFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 vh_thresh 设定。
u16AaThresh	45°与 135°边缘混合范围的阈值，调高此参数会降低伪彩、噪声与斜解析度 取值范围：[0x0, 0xFFFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 aa_thresh 设定。
u16VaThresh	垂直、水平、45°与 135°边缘混合范围的阈值，调高此参数会降低伪彩、噪声与垂直、水平、斜解析度 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 va_thresh 设定。
u16UuThresh	全部边缘混合范围的阈值，调高此参数会降低伪彩、噪声与解析度、锐度。 取值范围：[0x0, 0xFF] 默认值由文件 cmos.c 内结构体 cmos_isp_demosaic_t 的成员 uu_thresh 设定。
u8DemosaicConfig	去马赛克模块调试模式



成员名称	描述
	取值范围：[0x0, 0xFF] 0 表示正常输出，4 表示 UU 调试模式，17 表示 AA 调试模式，18 表示 VA 调试模式，19 表示 VH 调试模式。其他值保留。
u8LumThresh	该值越大，图像的大边缘越明显，该数组的八个值分别对应不同的 iso 值下的强度。对应关系如表 6-6 所示。
u8NpOffset	该值表示 demosaic noise profile 的偏移值，该数组的八个值分别对应不同的 iso 值下的强度。对应关系如表 6-7 所示。

表6-6 u8LumThresh 在不同的增益情况下的设置值

u8LumThresh	Again*Dgian*IspDgain (times)
u8LumThresh [0]	1
u8LumThresh [1]	2
u8LumThresh [2]	4
u8LumThresh [3]	8
u8LumThresh [4]	16
u8LumThresh [5]	32
u8LumThresh [6]	64
u8LumThresh [7]	128

表6-7 u8NpOffset 在不同的增益情况下的设置值

u8NpOffset	Again*Dgian*IspDgain (times)
u8NpOffset [0]	1
u8NpOffset [1]	2
u8NpOffset [2]	4
u8NpOffset [3]	8
u8NpOffset [4]	16
u8NpOffset [5]	32
u8NpOffset [6]	64
u8NpOffset [7]	128



【注意事项】

- Slope 越低，越少的相应角度的边缘会参与混合。
- Thresh 越高，角度的判定范围越窄。当 Thresh 很高时，只有最接近垂直、水平、45°、135°的边缘才会被考虑到。
- 本结构体参数为高级参数，不建议修改。

【相关数据类型及接口】

无。

6.12 黑电平

6.12.1 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。

6.12.2 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(const ISP_BLACK_LEVEL_S
*pstBlackLevel)
```

【参数】

参数名称	描述	输入/输出
pstBlackLevel	黑电平属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。



【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_GetBlackLevelAttr

【描述】

获取黑电平属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetBlackLevelAttr(ISP_BLACK_LEVEL_S *pstBlackLevel)
```

【参数】

参数名称	描述	输入/输出
pstBlackLevel	黑电平属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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。

6.12.3 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]; /*RW, Range: [0x0, 0xFFFF]*/
} 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, 0xFFFF] 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。

6.13 获取 ISP 模块虚拟地址

6.13.1 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。

6.13.2 API 参考

- [HI_MPI_GetISPRegAttr](#)：获取 ISP 模块虚拟地址属性。

HI_MPI_GetISPRegAttr

【描述】

获取 ISP 基地址属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetISPRegAttr(ISP_REG_ATTR_S *pstIspRegAttr)
```

【参数】

参数名称	描述	输入/输出
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，其值为错误码。

【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】

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。

6.13.3 数据类型

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



成员名称	描述
u32AwbExtRegSize	ISP AWB 库对应的大小

【注意事项】

无。

【相关数据类型及接口】

无。



7 Debug

7.1 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。

7.2 功能描述

Debug 调试信息涉及 AE、AWB 和 Sys 模块，在 ISP 运行过程中，通过使能相应的 Debug 模块，可以获取相应模块的状态信息。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。

7.3 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\_DEBUG\_INFO\_S *pstIspDebug )
```

【参数】

参数名称	描述	输入/输出
pstIspDebug	调试信息	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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】

- 调试信息的成员变量分为 AE、AWB 和 SYS，可以单独设置相应部分的使能开关。
- 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;
HI_U32 u32PhyAddr;
HI_VOID *pVitAddr;
FILE *fp;
ISP_DEBUG_INFO_S stIspDebug;
PAUSE;

CHECK_RET(HI_MPI_ISP_GetDebug(&stIspDebug), "ISP debug get");

stIspDebug.u32DebugDepth = 30; /* */

HI_U32 u32SizeHi = (stIspDebug.u32AWBSize & 0xFFFF0000) >> 16;
/*status*/
HI_U32 u32SizeLo = stIspDebug.u32AWBSize & 0xFFFF; /*cfg*/
HI_U32 u32MemSize = u32SizeLo + u32SizeHi * stIspDebug.u32DebugDepth;
s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&u32PhyAddr, &pVitAddr, NULL, NULL,
u32MemSize);

stIspDebug.u32AWBAddr = u32PhyAddr;
```



```
stIspDebug.bAWBDebugEnable = 1;

CHECK_RET(HI_MPI_ISP_SetDebug(&stIspDebug), "ISP debug set");

PAUSE;

HI_U32 *pu32VirAddr = (HI_U32 *)HI_MPI_SYS_Mmap(stIspDebug.u32AWBAddr,
u32MemSize);

fp=fopen("mst_30000.dat","wb");
if(fp==NULL)
{
    printf("open file mst_30000.dat error \n");
    return -1;
}

fwrite(pu32VirAddr,1,u32MemSize,fp);
printf("write file\n");
PAUSE;
fclose(fp);

stIspDebug.bAWBDebugEnable = 0;
CHECK_RET(HI_MPI_ISP_SetDebug(&stIspDebug), "ISP debug set");

HI_MPI_SYS_Munmap(pu32VirAddr, u32MemSize);
HI_MPI_SYS_MmzFree(u32PhyAddr, pVitAddr);
```

【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\_DEBUG\_INFO\_S *pstIspDebug )
```

【参数】

参数名称	描述	输入/输出
pstIspDebug	调试信息	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【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。

7.4 数据类型

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  bAEDebugEnable ;
    HI_U32   u32AEAddr ;
    HI_U32   u32AESize ;
    HI_BOOL  bAWBDebugEnable ;
    HI_U32   u32AWBAddr ;
    HI_U32   u32AWBSize ;
    HI_U32   u32SysDebugEnable ;
    HI_U32   u32SysAddr ;
}
```



```
HI_U32 u32SysSize ;  
HI_U32 u32DebugDepth ;  
}
```

【成员】

成员名称	描述
bAEDebugEnable	使能 AE 调试。使能该位后，可以获得 AE 状态信息。
u32AEAddr	AE 调试地址。用户分配一个地址，AE 调试信息将被写入到该地址模块的内存。
u32AESize	AE 调试模块所需内存的大小，需要用户分配内存，低 16 位表示存储固定信息的所需内存的大小，高 16 位表示存储一帧状态信息的大小。 最后需要分配的内存大小=总帧数 x 每帧状态信息的大小+固定信息的大小。 u32AESize 在 MPI 中为只读，不能使用 HI_MPI_ISP_SetDebug 进行写入。
bAWBDebugEnable	使能 AWB 调试。使能该位后，可以获得 AWB 状态信息。
u32AWBAddr	AWB 调试地址。用户分配一个地址，AWB 调试信息将被写入该地址模块的内存。
u32AWBSize	AWB 调试模块的所需内存的大小，需要用户分配内存，低 16 位表示存储固定信息的所需内存的大小，高 16 位表示存储一帧状态信息的大小。 最后需要分配的内存大小=总帧数 x 每帧状态信息的大小+固定信息的大小。 u32AWBSize 在 MPI 中为只读，不能使用 HI_MPI_ISP_SetDebug 进行写入。
u32SysDebugEnable	使能系统调试。使能该位后，可以获得系统调试信息。
u32SysAddr	系统调试地址，用户分配一个地址，系统调试信息将被写入该地址模块的内存。
u32SysSize	系统调试模块的所需内存的大小，需要用户分配内存，低 16 位表示存储固定信息的所需内存的大小，高 16 位表示存储一帧状态信息的大小。 最后需要分配的内存大小=总帧数 x 每帧状态信息的大小+固定信息的大小。 u32SysSize 在 MPI 中为只读，不能使用 HI_MPI_ISP_SetDebug 进行写入。
u32DebugDepth	调试深度，即需要获取调试信息的帧数。



【注意事项】

调试信息是指 ISP 在运行过程中的状态信息，使用该 MPI 时，首先需要确定调试信息的模块，确定需要获取多少帧图像的调试信息，然后用户分配相应大小的内存以存储信息并传入分配的地址。以 AE 模块为例，假设用户需要获取 30 帧的 AE 调试状态信息，则 `u32DebugDepth = 30`，所需分配的内存总大小为 $((u32AESize \& 0xFFFF0000) \gg 16) * u32DebugDepth + u32AESize \& 0xFFFF$ 。

【相关数据类型及接口】

无。



8 AF

8.1 概述

自动对焦的原理为采用图像处理技术，对图像进行成像质量分析，得到系统当前的对焦状态，然后通过驱动电机调整系统镜头的焦距，从而实现自动聚焦。图像处理技术主要通过分析图像的灰度差值和高频分量的多少来判断图像是否为聚焦后的清晰图像。

8.2 功能描述

AF 模块提供自动聚焦的统计信息，包括整帧图像的归一化对比度值，分区间图像的归一化对比度值。

8.3 API 参考

HI_MPI_ISP_SetFocusStaInfo

【描述】

设置 AF 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_SetFocusStaInfo (const ISP_FOCUS_STA_INFO_S  
*pstFocusStatistics)
```

【参数】

参数名称	描述	输入/输出
pstFocusStatistic	对焦统计信息	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

【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_GetFocusStaInfo](#)

HI_MPI_ISP_GetFocusStaInfo

【描述】

获取 AF 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetFocusStaInfo(ISP\_FOCUS\_STA\_INFO\_S *pstFocusStatistic)
```

【参数】

参数名称	描述	输入/输出
pstFocusStatistic	对焦统计信息	输入

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，其值为错误码。

【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_SetFocusStaInfo](#)

8.4 数据类型

ISP_FOCUS_STA_INFO_S

【说明】

定义 ISP 对焦统计信息。

【定义】

```
typedef struct hiISP_FOUCS_STA_INFO_S
{
    HI_U16  u16FocusMetrics;
    HI_U16  u16ThresholdRead;
    HI_U16  u16ThresholdWrite;
    HI_U16  u16FocusIntensity;
    HI_U8   u8MetricsShift;
    HI_U8   u8NpOffset;
    HI_U16  u16ZoneMetrics[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN];
} ISP_FOCUS_STA_INFO_S;
```



【成员】

成员名称	描述
u16FocusMetrics	整帧图像的归一化对比度值，取值范围为 [0x0, 0xFFFF]
u16ThresholdRead	调试用
u16ThresholdWrite	调试用
u16FocusIntensity	调试用
u8MetricsShift	统计信息值的比例因子，默认值为 3。
u8NpOffset	统计信息值与 Noise Profile。
u16ZoneMetrics[WEIGHT_ZONE_ROW] [WEIGHT_ZONE_COLUMN]	分区间的归一化对比度值，取值范围为 [0x0, 0xFFFF]

【注意事项】

无。

【相关数据类型及接口】

无。



9 错误码

ISP API 错误码如表 9-1 所示。

表9-1 ISP API 错误码

错误代码	宏定义	描述
0xA00A8006	HI_ERR_ISP_NULL_PTR	空指针错误
0xA00A8003	HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效
0x A00A8043	HI_ERR_ISP_SNS_UNREGISTER	Sensor 未注册
0xA00A0040	HI_ERR_ISP_NOT_INIT	没有初始化
0xA00A0041	HI_ERR_ISP_TM_NOT_CFG	时序没有配置
0xA00A0044	HI_ERR_ISP_INVALID_ADDR	无效地址
0xA00A0042	HI_ERR_ISP_ATTR_NOT_CFG	属性未配置



10 Proc 调试信息说明

10.1 概述

调试信息采用了 Linux 下的 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】

- 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 hi3518_isp.ko proc_param=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。



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。

10.2 ISP

【调试信息】

```
# cat /proc/umap/isp
```



[ISP] Version: [Hi3518_ISP_V1.0.0.0 Release], Build Time[Aug 12 2013, 14:17:34]

-----AE INFO-----

Again	AShift	Dgain	DShift	IspDg	IShift	SysGain	SShift	Iso	Line
162	4	16	4	16	4	648	6	1000	1122

Comp	Step	Tole	Error	Fps	SlowFrt	MaxLine	MaxAg	MaxDg	MaxIDg
128	2	10	61	30	16	65535	256	128	64

Target0	Target1	Target2	Target3	Target4	Thresh0	Thresh1	Thresh2	Thresh3
0x 4000	0x 4000	0x 3000	0x 4000	0x 2000	13	40	96	128

ManuEn	MaLine	MaAg	MaDg	WdrMode	AuFlick	SlowMod	UpMode
0	0	0	0	LINE	0	1	0

-----AWB INFO-----

Gain0	Gain1	Gain2	Gain3	CoTemp
0x1c3	0x116	0x116	0x2a4	4098

Color00	Color01	Color02	Color10	Color11	Color12	Color20	Color21	Color22
0x 217	0x80fb	0x801e	0x8063	0x 1c4	0x8063	0x 14	0x80d3	0x 1bd

ManuEn	MaSatEn	Sat	SatTg	Zones	Speed
0	0	122	128	32	25

-----DRC INFO-----

En	ManuEn	DrcSt	DrcStTg
0	0	66	128

-----DEMOSAIC INFO-----

VhSlope	VhTh	AaSlope	AaTh	VaSlope	VaTh	UuSlope	UuTh
0xf5	0x 32	0x98	0x 5b	0xe6	0x 52	0x90	0x 40

SatSlope	SatTh	AcSlope	AcTh
0x5d	0x171	0xcf	0x1b3

-----NR INFO-----

En	ManuEn	Thresh	Offset
1	0	25	26

-----SHARPEN INFO-----

En	ManuEn	SpD	SpUd	LumTh	GeSt
----	--------	-----	------	-------	------



1 0 122 174 128 85

-----SHADING INFO-----

En

0

#

【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
AE INFO	Again	Sensor 模拟增益。
	AShift	Sensor 模拟增益精度。
	Dgain	Sensor 数字增益。
	DShift	Sensor 数字增益精度。
	IspDg	ISP 数字增益。
	IShift	ISP 数字增益精度。
	SysGain	系统增益。
	SShift	系统增益精度。
	Iso	ISO 值。
	Line	Sensor 曝光时间，以行为单位。
	Comp	ExpCompensation，自动曝光调整时对曝光补偿量。
	Step	ExpStep，自动曝光调整时的初始步长。
	Tole	ExpTolerance，自动曝光调整时对曝光量的容忍偏差。
	Error	自动曝光调整时曝光量的偏差。
	Fps	帧率。
	SlowFrt	降帧倍数。
	MaxLine	最大曝光时间。
	MaxAg	最大 Sensor 模拟增益。
	MaxDg	最大 Sensor 数字增益。
	MaxIDg	最大 Isp 数字增益。



参数		描述
	Target0	自动曝光调整时五段直方图第一段的目标值。
	Target1	自动曝光调整时五段直方图第二段的目标值。
	Target2	自动曝光调整时五段直方图第三段的目标值。
	Target3	自动曝光调整时五段直方图第四段的目标值。
	Target4	自动曝光调整时五段直方图第五段的目标值。
	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。
	WdrMode	WDR 模式。
	AuFlick	自动抗闪开关。
	SlowMod	自动降帧模式。
	UpMode	曝光变量更新模式。
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。
	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。



参数		描述
	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。
	MaSatEn	手动饱和度开关。
	Sat	饱和度值。
	SatTg	饱和度目标值。
	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。
	DrcStTg	DRC 强度目标值。
DEMOSA IC INFO	VhSlope	垂直、水平边缘混合阈值的斜率。
	VhTh	垂直、水平边缘混合范围的阈值。
	AaSlope	45°与 135°边缘混合阈值的斜率。
	AaTh	45°与 135°边缘混合范围的阈值。
	VaSlope	垂直、水平、45°与 135°边缘混合阈值的斜率。
	VaTh	垂直、水平、45°与 135°边缘混合范围的阈值。
	UuSlope	全部边缘混合阈值的斜率。
	UuTh	全部边缘混合范围的阈值。
	SatSlope	饱和度混合阈值的斜率。
	SatTh	饱和度混合范围的阈值。
	AcSlope	直流分量混合阈值的斜率。
	AcTh	直流分量混合范围的阈值。
NR INFO	En	空域去噪的开关。
	ManuEn	手动空域去噪的开关。
	Thresh	噪点处理的目标门限值。
	Offset	噪点处理的目标偏移值。
SHARPE	En	Sharpen 的开关。



参数		描述
N INFO	ManuEn	手动 Sharpen 的开关。
	SpD	图像大边缘的锐度。
	SpUd	图像小纹理的锐度
	LumTh	Luma Thresh。
	GeSt	Ge Strength。
SHADIN G INFO	En	镜头阴影校正的开关