



HiISP

开发参考

文档版本 00B06
发布日期 2013-02-05

版权所有 © 深圳市海思半导体有限公司 2012-2013。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 00B06(2013-02-05)

第 5 章 CCM

5.4 API 参考中，修改 HI_MPI_ISP_SetSaturationAttr 和 HI_MPI_ISP_GetSaturationAttr 的【语法】和【参数】；HI_MPI_ISP_SetSaturationAttr 的【注意】中增加描述。

5.5 数据类型中，增加 ISP_SATURATION_ATTR_S。

第 6 章 IMP

6.4 镜头阴影校正中

6.4.3 API 参考下，HI_MPI_ISP_SetShadingTable 增加错误码 HI_ERR_ISP_ILLEGAL_PARAM；【注意】中增加 Hi3518 的设置时的注意事项。

6.4.4 数据类型下，Hi3518 的 ISP_SHADINGTAB_S 增加成员 u16ShadingOffCenter_R、u16ShadingOffCenter_G、u16ShadingOffCenter_B 和 u16ShadingTableNodeNumber 及描述。

文档版本 00B05(2012-12-26)

第 3 章 AE

修改 ISP_OP_TYPE_E 的【成员】的描述和【注意事项】。

第 4 章 AWB

HI_MPI_ISP_SetAWBAttr 的【注意】中增加“配置 AWB 算法的色温上下限”的描述。

ISP_AWB_ATTR_S 的【定义】和【成员】中增加 u8HighColorTemp 和 HI_U8 u8LowColorTemp；【注意】中增加自动白平衡算法的色温上下限的描述。



第 6 章 IMP

6.1.3 数据类型, ISP_SHARPEN_ATTR_S 增加成员 HI_U8 u8SharpenAltD[8]和 HI_U8 u8SharpenAltUd[8]及描述; 修改成员 u32StrengthTarget 的描述及最小值; 修改 u8StrengthMin 的默认值; 【注意】中修改 sharpen 的强度值与系统增益的关系需要参见的内容。

6.7.2 功能描述, 更新去噪算法强度值的相关描述。

6.7.4 数据类型, ISP_DENOISE_ATTR_S 增加成员 HI_U8 u8SnrThresh[8]及描述。

第 7 章 Debug

新增。

文档版本 00B04(2012-11-25)

第 2 章 系统控制

2.2 API 参考中增加 HI_MPI_ISP_SetAntiFlickerAttr 和 HI_MPI_ISP_GetAntiFlickerAttr。

2.3 数据类型中增加 ISP_ANTIFLICKER_S。

第 3 章 AE

3.5 数据类型中 ISP_EXP_STA_INFO_S 的【成员】增加了取值范围的描述。

第 4 章 AWB

4.5.数据类型中修改 ISP_MWB_ATTR_S 的【成员】取值范围; ISP_WB_STA_INFO_S 的【成员】u16CbMin 和 u16CrMin 的取值范围。

第 5 章 CCM

5.4 API 中增加了 HI_MPI_ISP_SetCCM 的【注意】中关于三个色彩还原矩阵的描述。

5.5 数据类型中修改 ISP_COLORMATRIX_S 的【定义】和【成员】; 【注意事项】中增加高、中、低色温需要满足的条件。

第 6 章 IMP

ISP_GAMMA_TABLE_S 的【成员】中增加 Hi3518 的取值范围; 修改【差异说明】中 Hi3518 的相关值。

ISP_SHARPEN_ATTR_S 的【注意事项】中增加“sharpen 的强度值与系统增益的关系”。

ISP_GAMMA_CURVE_E 修改【定义】和【成员】。

6.7.2 功能描述中, 增加去噪算法的强度值和系统增益的相关描述。

ISP_DENOISE_ATTR_S 的【注意事项】中修改“u8ThreshTarget 值越大, 对噪点抑制”的相关描述。

文档版本 00B03(2012-10-30)

第 2 章 系统控制



2.2 API 参考中增加 HI_MPI_ISP_SetSlowFrameRate 和 HI_MPI_ISP_GetSlowFrameRate。

第 3 章 AE

3.4.1 AE 控制模块中新增 HI_MPI_ISP_SetExpStaInfo 和 HI_MPI_ISP_GetExpStaInfo。

3.5.1 AE 中新增 ISP_EXP_STA_INFO_S

ISP_AE_ATTR_S 的【参数】u16ExpTimeMax、u16ExpTimeMin、u16DgainMax、u16DgainMin、u16AgainMax 和 u16AgainMin 的取值范围有修改。

3.5.2 AI 中新增 ISP_OP_TYPE_E

第 4 章 AWB

4.5. ISP_AWB_ATTR_S 的【参数】

u8Weight[WEIGHT_ZONE_ROW][WEIGHT_ZONE_COLUMN]增加取值范围。

ISP_WB_ZONE_STA_INFO_S 的【参数】增加取值范围说明。

第 5 章 CCM

HI_MPI_ISP_GetSaturationAttr 的【参数】pu8Value 修改描述。

ISP_COLORMATRIX_S 的【成员】u16CorrMatrix[9]增加取值范围。

第 6 章 IMP

6.1.3 数据类型中 ISP_SHARPEN_ATTR_S 的【注意事项】中新增一条事项。

6.5.3 API 参考中 HI_MPI_ISP_SetDefectPixelAttr 和 HI_MPI_ISP_GetDefectPixelAttr 的【举例】更新。

第 7 章 错误码

新增 4 个错误码。

文档版本 00B02 (2012-09-20)

在 ISP_INPUT_TIMING_S、ISP_IMAGE_ATTR_S、ISP_AE_ATTR_S、ISP_ME_ATTR_S、ISP_AI_ATTR_S、ISP_MWB_ATTR_S、ISP_WB_ZONE_STA_INFO_S、ISP_WB_STA_INFO_S、ISP_COLORMATRIX_S、ISP_SHADINGTAB_S、ISP_DP_ATTR_S、ISP_CR_ATTR_S、ISP_DIS_INFO_S 中增加参数的取值范围。

ISP_AWB_ATTR_S 的【差异说明】中增加“仅排序后的前 u8ZoneSel 个区间统计结果参与 AWB 校正”的描述。

ISP_AWB_ATTR_S 的【差异说明】中增加参与 AWB 校正的相关描述。

修改 HI_MPI_ISP_GetSaturationAttr 的【参数】pu8Value 的描述。

修改 Sharpen、Gamma、DRC 的描述。

修改 ISP_SHARPEN_ATTR_S、ISP_GAMMA_ATTR_S、ISP_DRC_ATTR_S 的【成员】的描述。



修改 ISP_GAMMA_TABLE_S 的【定义】和【成员】。

将 6.4.4 数据类型中的“补偿”改为“校正”。

修改 ISP_SHADING_ATTR_S、ISP_SHADING_ATTR_S 的【定义】。

文档版本 00B01 (2012-08-15)

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	3
1.2.6 软件流程	4
2 系统控制.....	6
2.1 功能概述.....	6
2.2 API 参考	6
2.3 数据类型.....	19
3 AE.....	24
3.1 概述.....	24
3.2 重要概念.....	24
3.3 功能描述.....	25
3.4 API 参考	27
3.4.1 AE 控制模块	27
3.4.2 AI 控制模块	36
3.5 数据类型.....	39
3.5.1 AE	39
3.5.2 AI	44
4 AWB.....	48
4.1 概述.....	48
4.2 重要概念.....	48
4.3 功能描述.....	48
4.3.1 AWB 模块工作原理.....	48
4.4 API 参考	49



4.4.1 AWB 控制模块.....	49
4.4.2 WB 统计信息.....	55
4.5 数据类型.....	60
4.5.1 WB.....	60
5 CCM.....	66
5.1 概述.....	66
5.2 重要概念.....	66
5.3 功能描述.....	66
5.4 API 参考.....	67
5.5 数据类型.....	71
6 IMP.....	73
6.1 Sharpen.....	73
6.1.1 功能描述.....	73
6.1.2 API 参考.....	73
6.1.3 数据类型.....	75
6.2 Gamma.....	77
6.2.1 功能描述.....	77
6.2.2 API 参考.....	77
6.2.3 数据类型.....	81
6.3 DRC.....	84
6.3.1 功能描述.....	84
6.3.2 API 参考.....	84
6.3.3 数据类型.....	86
6.4 镜头阴影校正.....	88
6.4.1 概述.....	88
6.4.2 功能描述.....	88
6.4.3 API 参考.....	89
6.4.4 数据类型.....	92
6.5 Defect Pixel.....	98
6.5.1 概述.....	98
6.5.2 功能描述.....	98
6.5.3 API 参考.....	98
6.5.4 数据类型.....	101
6.6 CrossTalk Removal.....	103
6.6.1 概述.....	103
6.6.2 功能描述.....	103
6.6.3 API 参考.....	103
6.6.4 数据类型.....	105
6.7 Denoise.....	106



6.7.1 概述	106
6.7.2 功能描述	106
6.7.3 API 参考	107
6.7.4 数据类型	109
6.8 DIS	110
6.8.1 概述	110
6.8.2 功能描述	110
6.8.3 API 参考	111
6.8.4 数据类型	116
6.9 去雾.....	118
6.9.1 功能描述	118
6.9.2 API 参考	118
6.9.3 数据类型	120
6.10 去伪彩.....	121
6.10.1 概述	121
6.10.2 功能描述	121
6.10.3 API 参考	121
6.10.4 数据类型	123
7 Debug.....	125
7.1 概述.....	125
7.2 功能描述.....	125
7.3 API 参考	125
7.4 数据类型.....	128
8 错误码.....	131



1 概述

1.1 概述

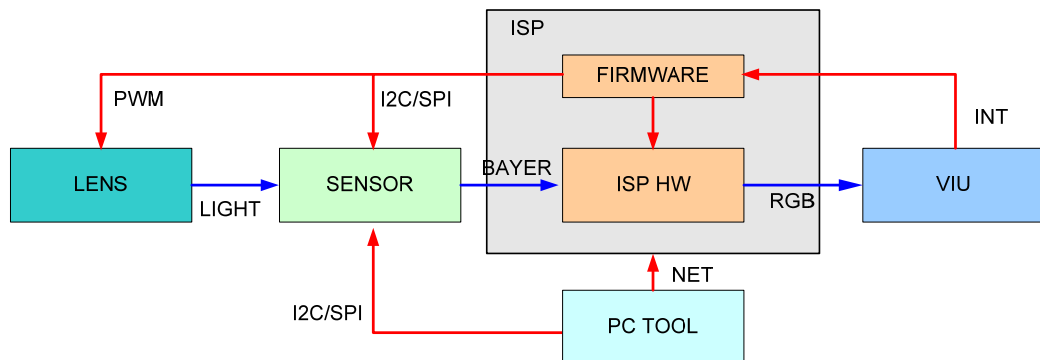
ISP 通过一系列数字图像处理算法完成对数字图像的效果处理。主要包括 3A、坏点校正、去噪、强光抑制、背光补偿、色彩增强、镜头阴影校正等处理。ISP 包括逻辑部分以及运行在其上的 firmware。这里主要介绍 ISP 的用户接口。

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

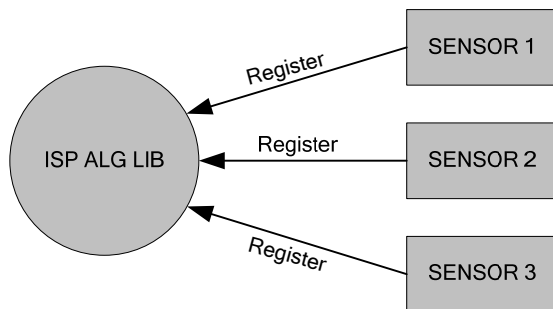


ISP 逻辑主要流程、具体概念和功能点请参见芯片手册。

1.2.2 设计思路

ISP 的 Firmware 包含两部分，一部分是算法控制部分，另一部分是 sensor 控制部分。Firmware 设计的基本思想是将算法控制部分和 sensor 控制部分分离开来，当有新的 sensor 需要适配时，只需改动 sensor 控制单元。ISP firmware 设计思路如图 1-2 所示。

图1-2 ISP firmware 设计思路



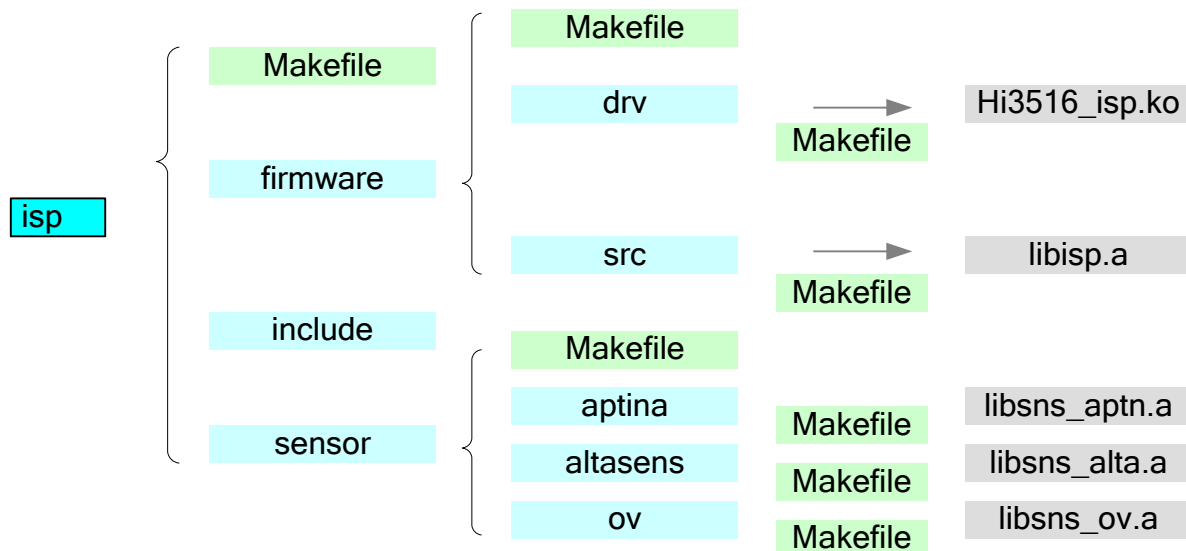
不同的 sensor 都向 ISP 的算法库注册控制函数，这些函数都以回调函数的形式存在。Firmware 通过这些回调函数控制 sensor，如调节曝光时间、模拟增益、数字增益，控制 lens 步进聚焦或旋转光圈等。

1.2.3 文件组织

ISP Firmware 的文件组织结构如图 1-3 所示，firmware 和 sensor 分开。Firmware 中的 drv 生成的驱动程序主要用来上报 ISP 中断，该中断是 firmware 运转的驱动力。Src 文件夹中包含 firmware 的核心算法，编译后生成 libisp.a，即基本算法库。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 中给出的形式符合一般客户的开发模式，算法库不变，用户来适配不同的 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。



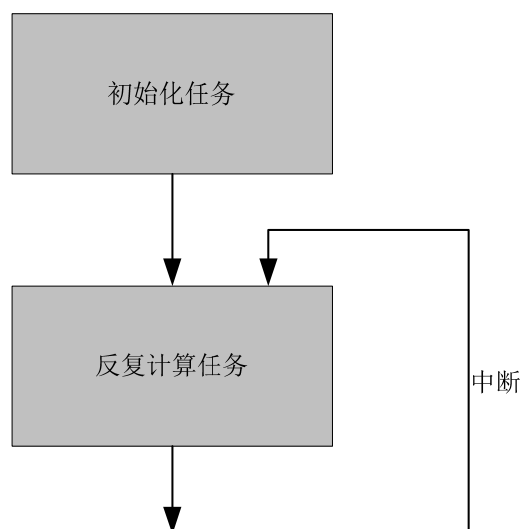
说明

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，主要完成整个 ISP 的初始化，包括 sensor 的部分初始化（回调形式实现）；另一部分是动态调节过程，在这个过程中，firmware 实时计算并进行相应控制。

图1-4 ISP firmware 内部流程

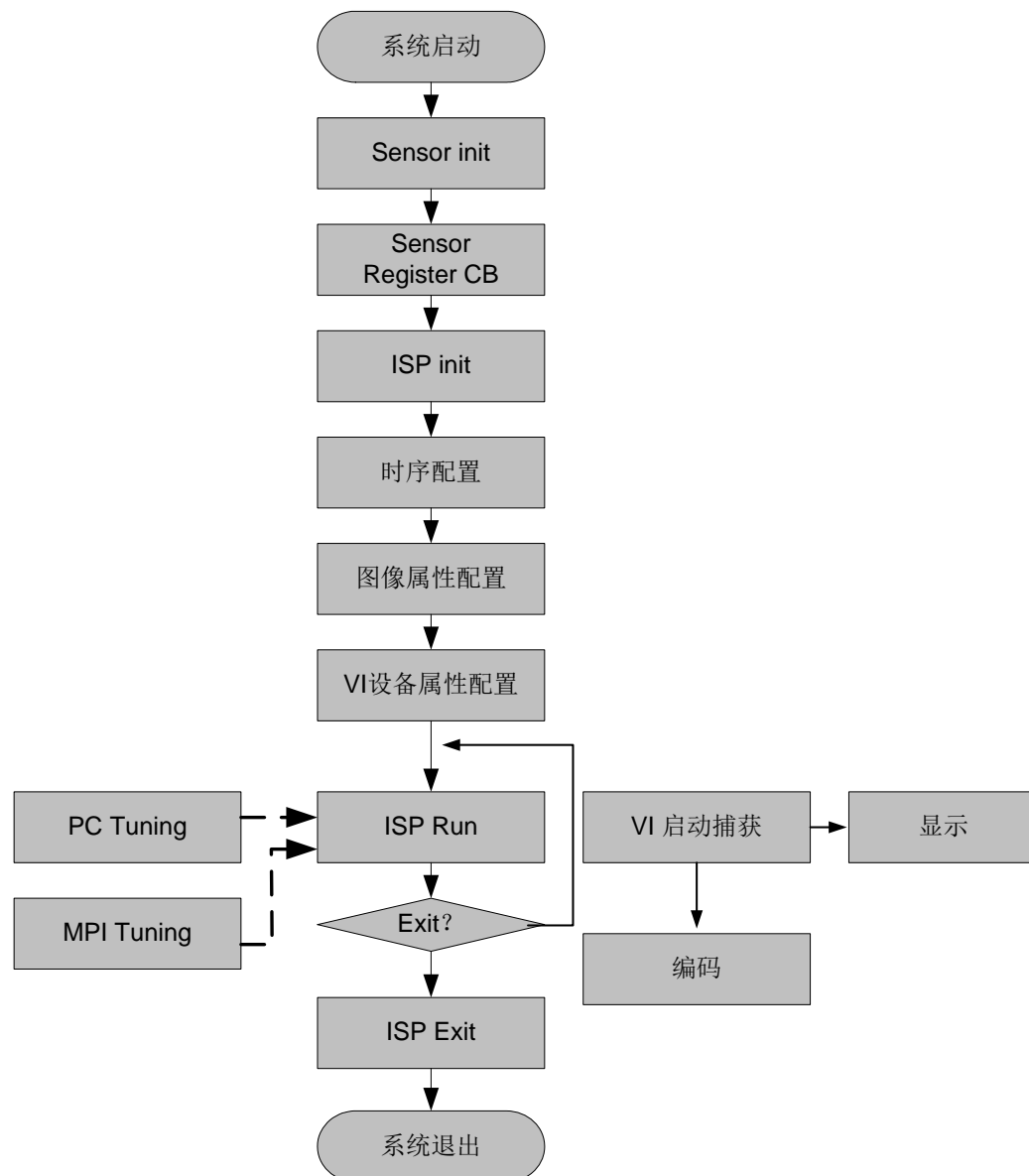


1.2.6 软件流程

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 1-5 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1-5 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_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。



【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](#)

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_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_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_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。

【注意事项】

无。

【相关数据类型及接口】

无。

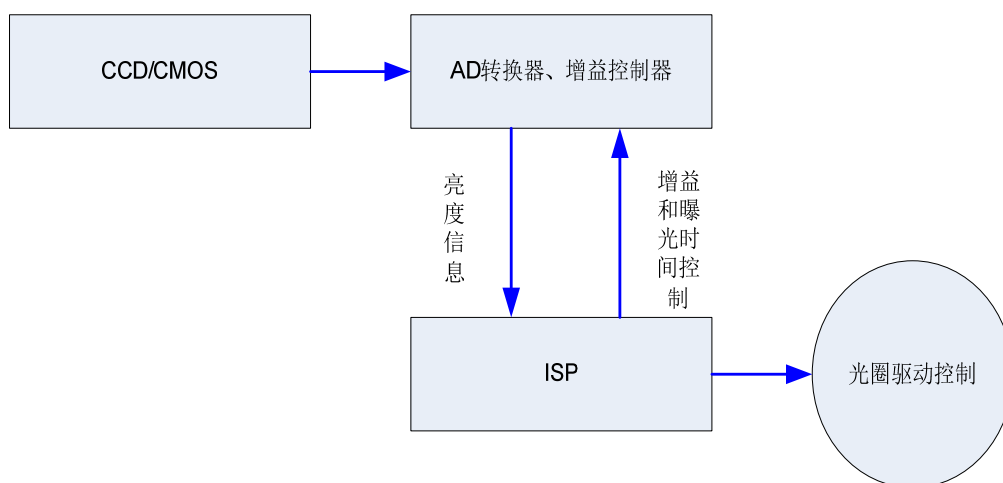


3 AE

3.1 概述

HiISP AE 模块实现的功能是:根据自动测光系统获得当前图像的曝光量,再自动配置镜头光圈、sensor 快门及增益来获得最佳的图像质量。自动曝光的算法主要分光圈优先、快门优先、增益优先。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

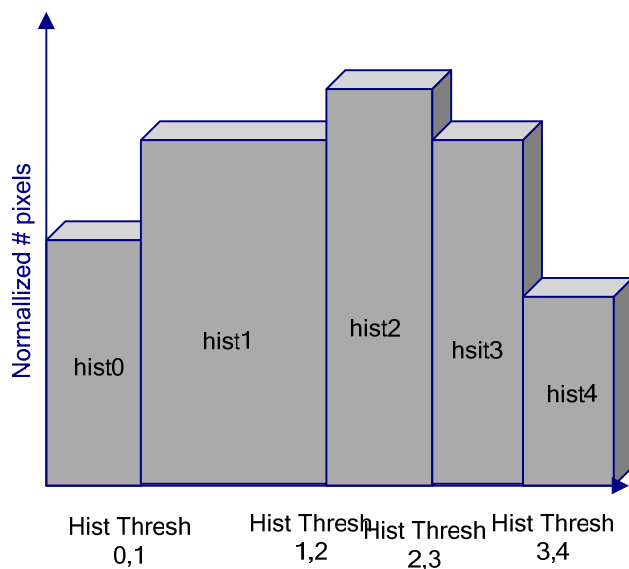
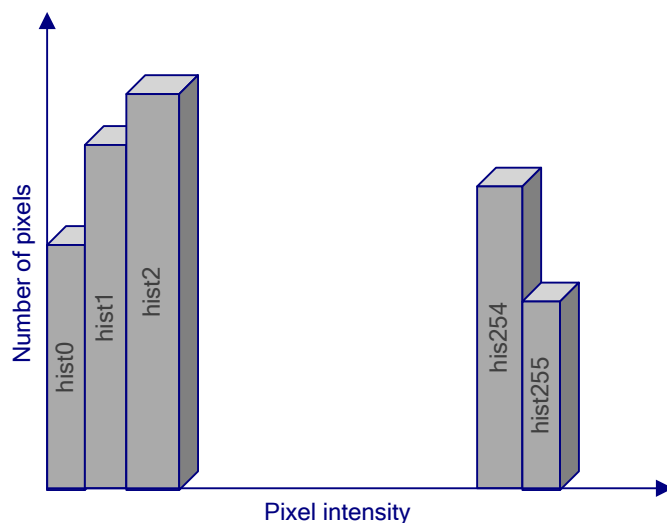
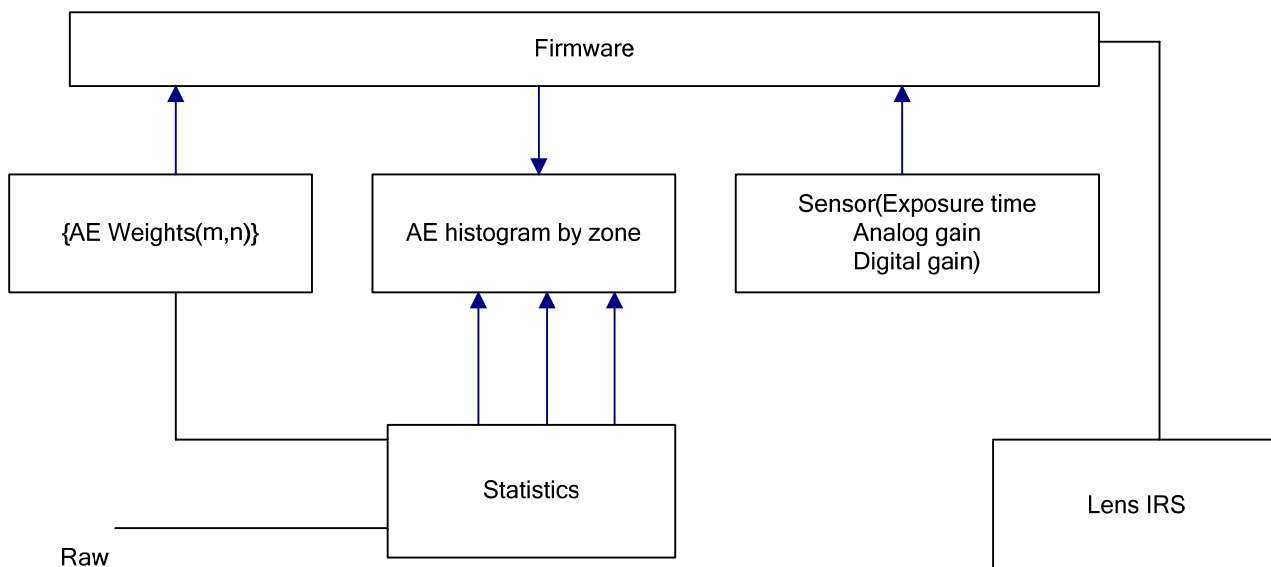


图3-3 AE 256 段统计信息直方图



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 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_SetMEAttr](#)：设置 ME 属性。
- [HI_MPI_ISP_GetMEAttr](#)：获取 ME 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_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.u16ExpTimeMax = 4000;  
    stAEAttr.u16ExpTimeMin = 2;  
    stAEAttr.u16AGainMax   = 28;  
    stAEAttr.u16AGainMin   = 0;  
    stAEAttr.u16DGainMax   = 38;  
    stAEAttr.u16DGainMin   = 0;  
    stAEAttr.u8ExpStep     = 8;  
}
```




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

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

【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_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_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



3.4.2 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 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_ME_ATTR_S](#)：定义 ISP 手动曝光属性。



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。
- 帧率优先模式是指自动曝光调节时优先保证帧率。

【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_DP_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。
- 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_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。



成员名称	描述
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_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;
} 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



成员名称	描述
	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]

【注意事项】

无。

【相关数据类型及接口】

无。

3.5.2 AI

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】

无。



【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]。



成员名称	描述
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。

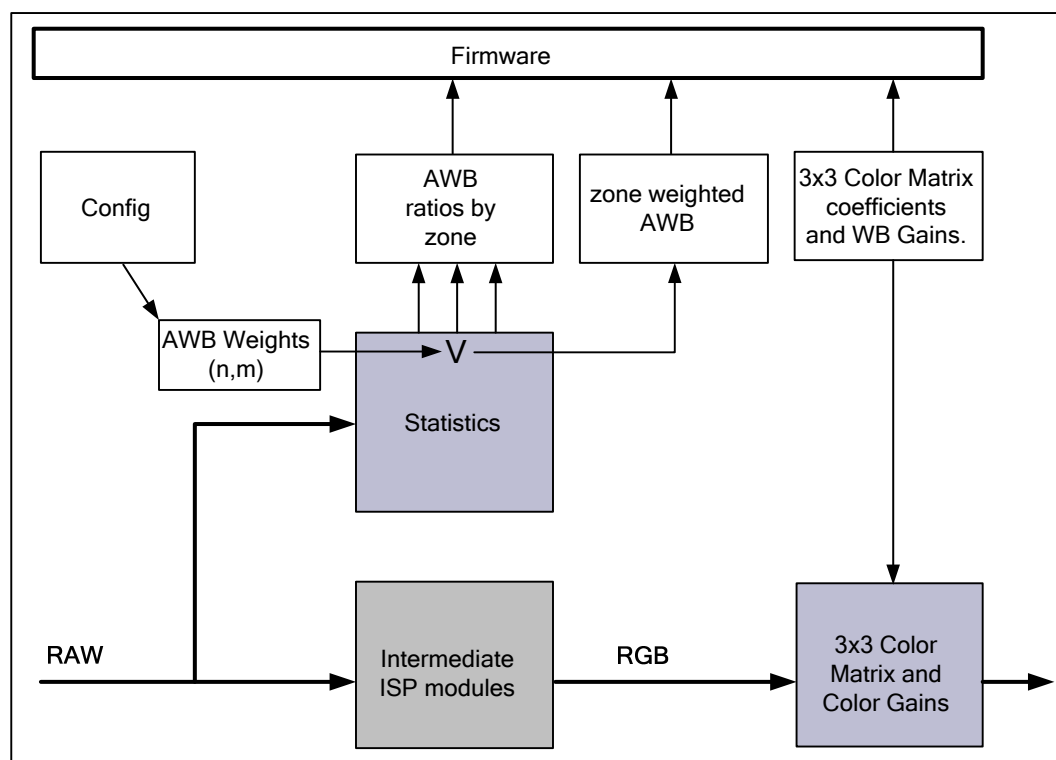
ISP 的 AWB 模块提供的统计信息主要为 R/G，B/G 的值。这两个值是一个平均值，是由每个 zone 中白点的 R/G 与 B/G 的总和与 AWB 的权重系数通过如下公式计算得到：

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

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 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_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。
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。
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](#)

4.4.2 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 *pu16ColorTemp);
```

【参数】

参数名称	描述	输入/输出
pu16ColorTemp	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。



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】

用户自定义 AWB 算法示例。

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

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

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

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



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

    stMWBAAttr.u16Rgain = stWBSTAInfo.u16GRgain;
    stMWBAAttr.u16Bgain = stWBSTAInfo.u16GBgain;
    stMWBAAttr.u16Ggain = 0x100;

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

【相关主题】

[HI_MPI_ISP_SetWBStaInfo](#)

4.5 数据类型

4.5.1 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_ATTR_S

【说明】

定义 ISP 自动白平衡属性。

【定义】

```
typedef struct hiISP_AWB_ATTR_S
{
    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
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，仅排序后的前 u8ZoneSel 个区间统计结果参与 AWB 校正，仅排序后的前 u8ZoneSel 个区间统计结果参与 AWB 校正</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，仅排序后的前 u8ZoneSel 个区间统计结果参与 AWB 校正，仅排序后的前 u8ZoneSel 个区间统计结果参与 AWB 校正。</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_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
u16Rgain	手动白平衡红色通道增益。 取值范围：[0x0,0xFFF]。
u16Ggain	手动白平衡绿色通道增益。 取值范围：[0x0,0xFFF]。
u16Bgain	手动白平衡蓝色通道增益。 取值范围：[0x0,0xFFF]。



【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, 0xFFF]。
u16Bg	区间白点 G/B 均值，数据格式为定点 4.8。 取值范围：[0x0, 0xFFF]。
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;

    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]。



成员名称	描述
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](#)



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_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】

- 饱和度期望值越大，颜色越鲜艳；期望值越小，颜色越平淡。AE 根据环境光线，实时调整 sensor 增益，当 sensor 增益小于 8x，实际饱和度等于期望饱和度；当 sensor 增益大于 8x 时，实际饱和度小于期望饱和度，且随着增益的增加，饱和度逐渐降低。
- 若使能手动饱和度，则实际饱和度值等于饱和度期望值。



【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_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;
}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 0x90。

【注意事项】

- 饱和度期望值越大, 颜色越鲜艳; 期望值越小, 颜色越平淡。AE 根据环境光线, 实时调整 sensor 增益, 当 sensor 增益小于 8x, 实际饱和度等于期望饱和度; 当 sensor 增益大于 8x 时, 实际饱和度小于期望饱和度, 且随着增益的增加, 饱和度逐渐降低。
- 若使能手动饱和度, 则实际饱和度值等于饱和度期望值。

【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：[0x0,0xFFFF]。
au16HighCCM[9]	高色温下的颜色校正矩阵。 取值范围：[0x0,0xFFFF]。
u16MidColorTemp	中等色温。
au16MidCCM[9]	中等色温下的颜色校正矩阵。 取值范围：[0x0,0xFFFF]。
u16LowColorTemp	低色温。
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, 通常情况建议在相同增益情形下设置的 u8SharpenAltUd 的值小于 u8SharpenAltD 的值, 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(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(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】

此功能目前暂不支持。

【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_SetDRCAttr(const ISP_DRC_ATTR_S *pstDRCAttr);
```

【参数】

参数名称	描述	输入/输出
pstDRCAttr	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_GetDRCAttr](#)

HI_MPI_ISP_GetDRCAttr

【描述】

获取 DRC 属性。

【语法】

```
HI_MPI_ISP_GetDRCAttr(ISP_DRC_ATTR_S *pstDRCAttr);
```

【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 bDRCManualEnable;
HI_U32  u32StrengthTarget;
HI_U32  u32SlopeMax;
HI_U32  u32SlopeMin;
HI_U32  u32WhiteLevel;
HI_U32  u32BlackLevel;
} ISP_DENOISE_ATTR_S;

```

【成员】

成员名称	描述
bDRCEnable	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。
bDRCManualEnable	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。 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。 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。 取值范围: [0, 0xFFFF]。 默认值由文件 cmos.c 内结构体 cmos_isp_default_t 的成员 iridix_wl 设定。
u32BlackLevel	DRC 增强的最小像素值。 取值范围: [0, 0xFFFF]。 默认值为 0。



【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。
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。 取值范围：[0x0, 0xFFFF]
u16ShadingCenterR_Y	R 分量中心点的 Y 轴坐标。 取值范围：[0x0, 0xFFFF]
u16ShadingCenterG_X	G 分量中心点的 X 轴坐标。 取值范围：[0x0, 0xFFFF]
u16ShadingCenterG_Y	G 分量中心点的 Y 轴坐标。 取值范围：[0x0, 0xFFFF]
u16ShadingCenterB_X	B 分量中心点的 X 轴坐标。 取值范围：[0x0, 0xFFFF]
u16ShadingCenterB_Y	B 分量中心点的 Y 轴坐标。 取值范围：[0x0, 0xFFFF]
u16ShadingTable_R	R 分量的校正表。 取值范围：[0x0, 0xFFFF]
u16ShadingTable_G	G 分量的校正表。 取值范围：[0x0, 0xFFFF]



成员名称	描述
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： [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： [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： [0x0, 0xFFFF]
u16ShadingTableNodeNumber	每个分量校正表中使用到的节点个数。 取值范围： [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动态坏点校正功能。

6.5.3 API 参考

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。
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.u16BadPixelCountMax = 0x200;
stDPAttr.u16BadPixelCountMin = 0x40;

for(i=0; i< 1024;i++)
{
    stDPAttr.u32BadPixelTable[i] = 0;
}
HI_MPI_ISP_SetDefectPixelAttr(&stDPAttr);
PAUSE;
HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
PAUSE;
```

【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 *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。
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  stDPAttr;
    HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
    stDPAttr.bEnableStatic = HI_TRUE;
    stDPAttr.bEnableDynamic= HI_FALSE;
    stDPAttr.bEnableDetect = HI_TRUE;
    stDPAttr.u16BadPixelCountMax = 0x200;
    stDPAttr.u16BadPixelCountMin = 0x40;
    for(i=0; i< 1024;i++)
    {
        stDPAttr.u32BadPixelTable[i] = 0;
    }

    HI_MPI_ISP_SetDefectPixelAttr(&stDPAttr);
    PAUSE;
    HI_MPI_ISP_GetDefectPixelAttr(&stDPAttr);
    PAUSE;

}
```

【相关主题】

无。

6.5.4 数据类型

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 bEnableStatic;
    HI_BOOL bEnableDynamic;
    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
bEnableStatic	使能静态坏点校正功能。
bEnableDynamic	使能动态坏点校正功能。
bEnableDetect	使能静态坏点检测功能。
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。

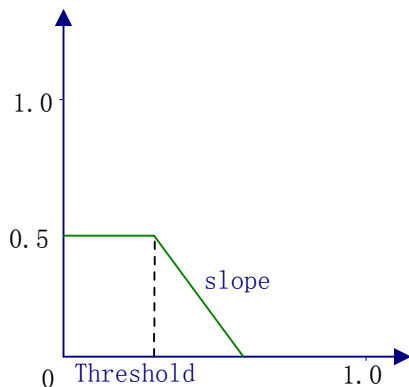
6.6.2 功能描述

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    u8Strength;
    HI_U8    u8Sensitivity;
    HI_U16   u16Threshold;
    HI_U16   u16Slope;
} ISP_CR_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能 CrossTalk remove 功能。
u8Strength	设置 Noise Profile 校正强度值。
u8Sensitivity	设置 CrossRemove 敏感度值。
u16Threshold	设置 CrossRemove 门限值。 取值范围：[0, 0xFFFF]



成员名称	描述
u16Slope	设置 CrossRemove 斜率值。 取值范围：[0, 0xFF]

【注意事项】

- u8Strength 值指的是与 Noise Profile 校正值保持一致，一般使用默认值。
- 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 6-4 所示。

表6-4 u8SnrThresh[8]在不同增益情况下对应的设置值

Snr_thresh	Again*Dgian(times)
u8SnrThresh [0]	1
u8SnrThresh [1]	2
u8SnrThresh [2]	4
u8SnrThresh [3]	8
u8SnrThresh [4]	16
u8SnrThresh [5]	32
u8SnrThresh [6]	64
u8SnrThresh [7]	128

【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

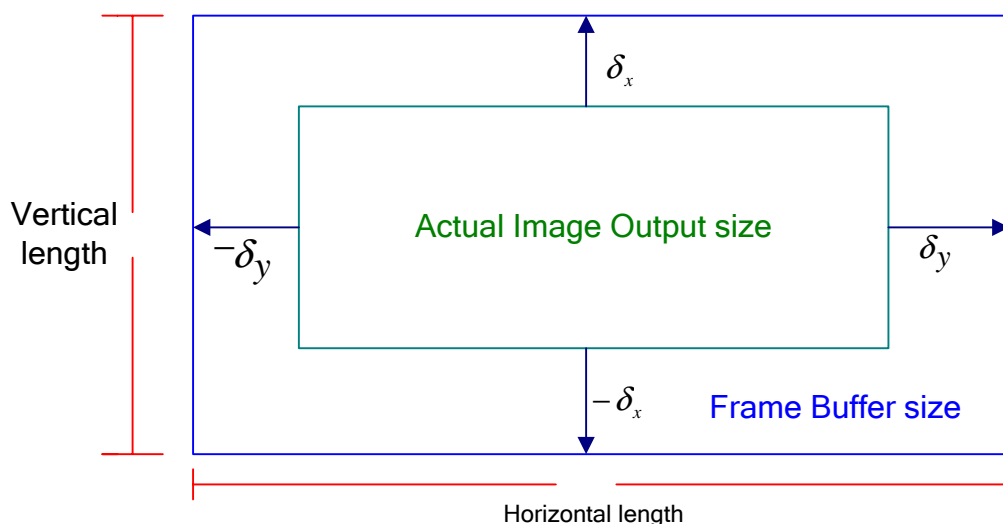
6.8.1 概述

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。
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

【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 s8Xoffset;
    HI_S8 s8Yoffset;
} ISP_DIS_INFO_S;
```

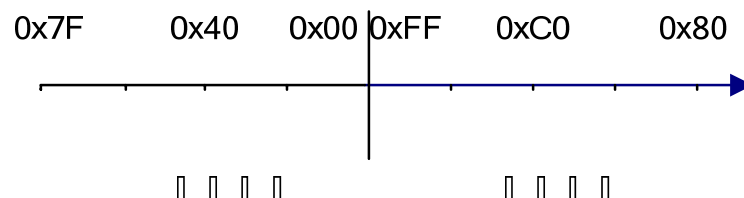
【成员】

成员名称	描述
s8Xoffset	DIS 模块输出的水平方向 Xoffset 取值范围为 [0x00, 0x80]
s8Yoffset	DIS 模块输出的水平方向 Yoffset 取值范围为[0x80, 0xFF]

【注意事项】

s8Xoffset 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

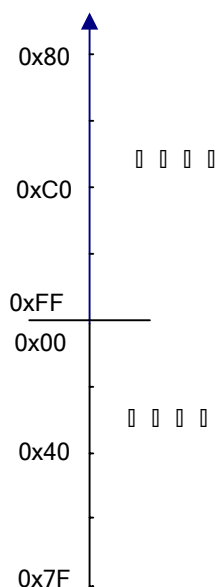


s8Yoffset 为有符号型，当图像水平向上移动时输出范围为[0x80, 0xFF]，输出以反码形式给出，即当输出为 0xFF，表示图像水平向上移动了 1 个像素，当输出为 0xFE，表示



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来实现的，该模块计算每帧图像灰度的均值信息，将该均值信息作为参数动态的改变四个数据通道的 R、Gr、Gb 和 B 的黑电平值。

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;
} ISP_ANTIFOG_S;
```



【成员】

成员名称	描述
bEnable	使能 antifog 功能

【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。



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 错误码

ISP API 错误码如表 8-1 所示。

表8-1 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