



HiISP

开发参考

文档版本 12

发布日期 2020-12-08

版权所有 © 上海海思技术有限公司 2020。保留一切权利。

未经本公司书面许可，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海思技术有限公司的商标。

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。

上海海思技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<http://www.hisilicon.com/cn/>

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。

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3559A	V100
Hi3559C	V100
Hi3569	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300



产品名称	产品版本
Hi3559	V200
Hi3556	V200
Hi3516E	V300
Hi3516E	V200
Hi3518E	V300
Hi3516D	V200
Hi3562	V100
Hi3566	V100
Hi3568	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	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。



符号	说明
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

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 12 (2020-12-08)

第 12 次正式版本发布

2.2 小节，删除 HI_MPI_ISP_SetTunnelInfo 和 HI_MPI_ISP_GetTunnelInfo；
HI_MPI_ISP_SetSmartInfo 【注意】和【举例】涉及修改；HI_MPI_ISP_SetPubAttr、
HI_MPI_ISP_GetVDTimeOut 【注意】涉及修改

2.3 小节，修改 ISP_CMOS_SPLIT_POINT_S、ISP_CMOS_SPLIT_S、
ISP_CMOS_EXPANDER_POINT_S、ISP_CMOS_EXPANDER_S、ISP_SMART_INFO_S；
删除 ISP_TUNNEL_INFO_S

3.4.2 小节，HI_MPI_ISP_SetSmartExposureAttr 【举例】涉及修改

3.5.2 小节，ISP_EXPOSURE_ATTR_S 涉及修改

6.19.1 小节中添加注意事项

6.23.3 小节，ISP_CSC_ATTR_S 【成员】涉及修改

7.4 小节，ISP_AE_STATISTICS_S 【成员】涉及修改

13.6.3 涉及修改

文档版本 11 (2020-07-15)

第 11 次正式版本发布

2.3 小节，ISP_CMOS_ACS_S、ISP_CMOS_EXPANDER_S、ISP_INIT_ATTR_S 的【注意
事项】涉及修改；ISP_QUICK_START_PARAM_S 【成员】涉及修改



3.5.1 小节, AE_SENSOR_DEFAULT_S【成员】涉及修改

3.5.2 小节, ISP_AE_ATTR_S、ISP_WDR_EXPOSURE_ATTR_S、ISP_EXP_INFO_S【成员】涉及修改

4.3、6.37.2、6.37.3 和 10.2 小节涉及修改

文档版本 10 (2020-05-30)

第 10 次正式版本发布

刷新 Hi3559V200/Hi3556V200 支持规格情况。

2.2 小节, 新增 HI_MPI_ISP_SetTunnelInfo 和 HI_MPI_ISP_GetTunnelInfo。
HI_MPI_ISP_IrAutoRunOnce 涉及更新。

2.3 小节, 新增 ISP_TUNNEL_INFO_S。ISP_QUICK_START_PARAM_S、
ISP_AE_INFO_S、ISP_AE_RESULT_S、ISP_INIT_ATTR_S 涉及更新。

3.5.1 小节, AE_SENSOR_DEFAULT_S、AE_ACCURACY_E 涉及更新。

3.5.2 小节, ISP_SMART_EXPOSURE_ATTR_S、ISP_AE_ROUTE_EX_NODE_S、
ISP_EXP_INFO_S、ISP_COLORMATRIX_PARAM_S 涉及更新。

7.4 小节, ISP_AE_STATISTICS_S 涉及更新。

6.22.1、10.2、13.5.4、13.5.6、13.6 小节, 涉及更新。

14.5 小节, 图 14-1 涉及更新。

文档版本 09 (2020-03-31)

第 9 次正式版本发布

2.3 小节, 新增 ISP_ACS_Y_SHADING_LUT_S、ISP_ACS_COLOR_SHADING_LUT_S、
ISP_ACS_CALIB_PARAM_S 和 ISP_CMOS_ACS_S

ISP_CMOS_LOGLUT_S、ISP_CMOS_ALG_KEY_U、ISP_CMOS_DEFAULT_S 涉及修改

3.4.1 小节, HI_MPI_AE_SensorRegCallBack 涉及修改

3.5.1 小节, AE_SENSOR_EXP_FUNC_S 涉及修改

3.5.3 小节, 添加 ISP_EXP_CONVERT_PARAM 和 ISP_EXP_PARAM_REG

新增 6.33、6.37 和 13.6 小节



文档版本 08 (2020-01-20)

第 8 次正式版本发布

2.3 小节, ISP_AE_RESULT_S【注意事项】涉及修改,ISP_CTRL_PARAM_S【定义】、【成员】和【注意事项】涉及修改

3.5.1 小节, AE_SENSOR_DEFAULT_S【成员】涉及修改

3.5.2 小节, ISP_HDR_EXPOSURE_ATTR_S【成员】和【注意事项】涉及修改

6.13.3 小节, ISP_DEMOSAIC_MANUAL_ATTR_S【注意事项】涉及修改

6.17.3 小节, ISP_LOCAL_CAC_ATTR_S 和 ISP_GLOBAL_CAC_ATTR_S 涉及修改

6.23.3 小节, ISP_CSC_ATTR_S【注意事项】涉及修改

6.35.2 小节, HI_MPI_ISP_CalcFlickerType【注意】涉及修改

7.4 小节, ISP_FOCUS_STATISTICS_CFG_S【注意】涉及修改

新增 14.5 小节

文档版本 07 (2019-11-30)

第 7 次正式版本发布

添加 Hi3562V100 和 Hi3566V100 的相关内容

2.3 小节, ISP_CMOS_GE_S、ISP_CMOS_NOISE_CALIBRATION_S 涉及修改

3.5.2 小节, ISP_EXPOSURE_ATTR_S 涉及修改

6.8.2、6.15.2、6.24.3、6.30.3 及 10.2 小节涉及修改

文档版本 06 (2019-10-12)

第 6 次正式版本发布

2.2 小节, HI_MPI_ISP_Init【注意】涉及修改。

2.3 小节, ISP_SMART_INFO_S、ISP_AWB_PARAM_S、ISP_MODULE_CTRL_U、ISP_SNS_REGS_INFO_S、ISP_MOD_PARAM_S、ISP_INIT_ATTR_S 涉及修改

3.4.2 小节, HI_MPI_ISP_SetSmartExposureAttr 的【注意】涉及修改

3.5.2 小节, ISP_SMART_EXPOSURE_ATTR_S 的【注意】涉及修改



4.4.2 小节, HI_MPI_ISP_CalGainByTemp 涉及修改。

6.1.3 小节, ISP_SHARPEN_AUTO_ATTR_S 涉及修改

6.13.3 小节, ISP_DEMOSAIC_MANUAL_ATTR_S 和
ISP_DEMOSAIC_MANUAL_ATTR_S、ISP_CP_LUT_S、ISP_CA_ATTR_S【注意事项】涉
及修改。

6.17.3 小节, ISP_DEPURPLESTR_AUTO_ATTR_S【注意事项】涉及修改。

7.4 小节, ISP_AE_GRID_INFO_S、ISP_MG_GRID_INFO_S、ISP_AWB_GRID_INFO_S
与 ISP_FOCUS_GRID_INFO_S 涉及修改

6.35.1 小节涉及修改

文档版本 05 (2019-08-12)

第 5 次正式版本发布

2.3 小节, ISP_SNS_OBJ_S、ISP_CMOS_DRC_S、ISP_CMOS_DPC_S 和
ISP_AE_INFO_S 涉及更新。ISP_AE_RESULT_S【注意事项】涉及更新

2.3 小节, 新增数据类型 ISP_SNS_MIRRORFLIP_TYPE_E 和 ISP_SNS_BUS_EX_S。

3.5.1 小节, AE_SENSOR_DEFAULT_S【定义】和【成员】涉及修改,
AE_ACCURACY_S 的【成员】涉及修改

3.5.2 小节, ISP_SMART_EXPOSURE_ATTR_S【定义】、【成员】和【注意事项】涉及修
改

7.4 小节, ISP_AE_STATISTICS_S 涉及修改

文档版本 04 (2019-07-10)

第 4 次正式版本发布

2.2 小节, 新增 HI_MPI_ISP_SetRawPos 和 HI_MPI_ISP_GetRawPos;
HI_MPI_ISP_RunOnce【注意】和【举例】涉及修改

2.3 小节, 新增 ISP_CMOS_AWB_ATTR_S、ISP_RAW_POS_E、ISP_RAW_POS 和
ISP_SPECAWB_CONTROL_ATTR_S;

ISP_SENSOR_EXP_FUNC_S 和 ISP_PRIOR_FRAME_E 的【注意事项】涉及修改;

ISP_CMOS_ALG_KEY_U、ISP_CMOS_DEFAULT_S、ISP_INIT_ATTR_S【定义】和【成
员】涉及修改



- 3.5.2 小节, ISP_AE_STRATEGY_E 【注意事项】涉及修改
- 4.4.2 小节, 新增 HI_MPI_ISP_SetSpecAwbControlAttr 和 HI_MPI_ISP_SetSpecAwbControlAttr
- 4.5.2 小节, ISP_AWB_CBCR_TRACK_ATTR_S 【注意事项】涉及修改
- 6.1.1 小节涉及修改
- 6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S 【成员】涉及修改
- 6.11.3 小节, ISP_DEHAZE_ATTR_S 【注意事项】涉及修改
- 6.13.3 小节, ISP_DEMOSAIC_MANUAL_ATTR_S 和 ISP_DEMOSAIC_AUTO_ATTR_S 【定义】和 【成员】涉及修改
- 6.15.2 小节, HI_MPI_ISP_FPNCalibrate 【注意】涉及修改
- 7.4 小节, 新增 ISP_AWB_GAIN_SWITCH_E
- 10.2 小节涉及修改

文档版本 03 (2019-06-20)

- 2.3 小节, ISP_SENSOR_EXP_FUNC_S、ISP_AWB_INFO_S 和 ISP_SNS_STATE_S 涉及修改, 新增 ISP_CMOS_LCAC_MANUAL_S, ISP_CMOS_LCAC_AUTO_S, ISP_CMOS_LCAC_S。
- 3.4.2 小节, HI_MPI_ISP_SetAERouteAttr 【注意】涉及修改
- 3.5.1 小节, AE_SENSOR_DEFAULT_S 涉及修改
- 3.5.2 小节, ISP_AE_ATTR_S 【注意】涉及修改
- 7.4 小节, ISP_AE_HIST_CONFIG_S 【注意事项】涉及修改
- 6.11.3 小节, ISP_DEHAZE_ATTR_S 【注意事项】有添加
- 6.15.2 小节, HI_MPI_ISP_FPNCalibrate 【注意】涉及修改

文档版本 02 (2019-05-30)

- 2.3 小节, ISP_AE_RESULT_S 【成员】涉及更新。ISP_INIT_ATTR_S 【定义】【注意】涉及更新。



4.4.2 小节, HI_MPI_ISP_SetSpecAwbAttr、HI_MPI_ISP_GetSpecAwbAttr、HI_MPI_ISP_SetCaaControlAttr、HI_MPI_ISP_GetCaaControlAttr 【注意】涉及更新。

4.5.1 小节, AWB_SENSOR_DEFAULT_S 【定义】【成员】涉及更新。

新增 6.35 小节, 涉及更新。

文档版本 01 (2019-05-15)

2.2 小节, HI_MPI_ISP_SetPubAttr 涉及修改

2.3 小节, ISP_CMOS_SHARPEN_MANUAL_S、ISP_CMOS_SHARPEN_AUTO_S、ISP_CTRL_CMD_E、ISP_CTRL_PARAM_S 涉及修改

3.4.2 小节, HI_MPI_ISP_SetAERouteAttr、HI_MPI_ISP_SetAERouteAttrEx、HI_MPI_ISP_SetAERouteSFAttr、HI_MPI_ISP_SetAERouteSFAttrEx 【注意】涉及修改

3.5.2 小节, ISP_FSWDR_MODE_E、ISP_AE_ATTR_S、ISP_EXPOSURE_ATTR_S、ISP_EXP_INFO_S 【注意】涉及修改

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S 和 ISP_SHARPEN_AUTO_ATTR_S 涉及修改

6.5.3、6.10.2、6.10.3 和 6.30.2 小节涉及修改

6.18.3 小节, ISP_WDR_WBGAIN_POSITION_E 【注意事项】涉及修改

6.21.3 小节, ISP_LDCI_ATTR_S 涉及修改

10.2 小节涉及修改

文档版本 00B24 (2019-04-15)

第 24 次临时版本发布

2.2 小节, 新增 HI_MPI_ISP_SetSmartInfo 和 HI_MPI_ISP_GetSmartInfo。
HI_MPI_ISP_SetRegister 【注意】涉及更新。

2.3 小节, ISP_AWB_INFO_S、ISP_VD_TYPE_E、ISP_AE_RESULT_S 涉及更新。

3.4.2 小节, HI_MPI_ISP_SetExposureAttr 、HI_MPI_ISP_SetAERouteAttr、HI_MPI_ISP_SetAERouteAttrEx、HI_MPI_ISP_QueryExposureInfo 、
HI_MPI_ISP_SetSmartExposureAttr、HI_MPI_ISP_GetSmartExposureAttr 涉及更新。



新增 HI_MPI_ISP_SetAERouteSFAttr、HI_MPI_ISP_GetAERouteAttr、
HI_MPI_ISP_SetAERouteSFAttrEx、HI_MPI_ISP_GetAERouteSFAttrEx。

3.5.1 小节，AE_SENSOR_DEFAULT_S 涉及更新。

3.5.2 小节，ISP_AE_MODE_E、ISP_SMART_EXPOSURE_ATTR_S 涉及更新。新增
ISP_PRIOR_FRAME_E。

6.7.2 小节，HI_MPI_ISP_SetDPCalibrate 涉及更新。

7.4 小节，ISP_AE_STATISTICS_S 涉及更新。

文档版本 00B23 (2019-03-30)

第 23 次临时版本发布

新增 Hi3516DV200 内容。

文档版本 00B22 (2019-02-28)

第 22 次临时版本发布

新增 Hi3516AV300 内容。

2.3 小节，RECT_S【注意事项】涉及修改；ISP_MODULE_CTRL_U、
ISP_CMOS_SHARPEN_S【定义】涉及修改

6.1.3 小节，ISP_SHARPEN_MANUAL_ATTR_S、ISP_SHARPEN_AUTO_ATTR_S【定
义】涉及修改

6.23.3 小节，ISP_CSC_ATTR_S【注意事项】涉及修改

文档版本 00B21(2019-01-30)

第 21 次临时版本发布

2.2 小节，HI_MPI_ISP_SetRegister【注意】涉及更新。

2.3 小节，ISP_CMOS_RGBIR_S、ISP_AE_INFO_S、ISP_AE_RESULT_S、
ISP_SPECAWB_CAA_TBL_S、ISP_DCF_CONST_INFO_S 涉及更新。新增
ISP_SMART_INFO_S。

3.4.2 小节，新增 HI_MPI_ISP_SetSmartExposureAttr 和
HI_MPI_ISP_GetSmartExposureAttr。

3.5.2 小节，新增 ISP_SMART_EXPOSURE_ATTR_S。



3.5.3 小节, ISP_IRIS_ATTR_S 【注意事项】涉及更新。

4.5.2 小节, ISP_AWB_ATTR_S 【成员】和 ISP_AWB_ATTR_EX_S 【注意】涉及更新。

6.8.1 小节和 6.24.1 小节, 涉及更新。

6.17.2 小节, HI_MPI_ISP_SetGlobalCacAttr 和 HI_MPI_ISP_GetGlobalCacAttr 【注意】涉及更新。

6.17.3 小节, ISP_DEPURPLESTR_AUTO_ATTR_S 和 ISP_GLOBAL_CAC_ATTR_S 【注意】涉及更新。

6.30.2 小节, HI_MPI_ISP_QueryInnerStateInfo 【参数】涉及更新。

6.32.3 小节, ISP_MLSC_CALIBRATION_CFG_S 涉及更新。

6.34.3 小节, ISP_RGBIR_ATTR_S 【注意】涉及更新。

7.4 小节, ISP_AE_CROP_S 【注意】涉及更新。

文档版本 00B20(2019-01-14)

第 20 次临时版本发布

2.3 小节, ISP_CMOS_SHARPEN_MANUAL_S 【成员】涉及更新。
ISP_MOD_PARAM_S 【注意事项】涉及更新。

3.5.3 小节, ISP_IRIS_ATTR_S 【注意事项】涉及更新。

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S 涉及更新。

6.6.3 小节, ISP_SHADING_LUT_ATTR_S 涉及更新。

补充 Hi3518EV300 内容。

文档版本 00B19(2018-12-26)

第 19 次临时版本发布

2.2 小节, HI_MPI_ISP_SetPipeDifferAttr 涉及更新。

2.3 小节, ISP_MODULE_CTRL_U、ISP_CMOS_SHARPEN_MANUAL_S、
ISP_CMOS_SHARPEN_AUTO_S、ISP_MOD_PARAM_S 涉及更新。

3.5.1 小节, AE_SENSOR_DEFAULT_S 涉及更新。

4.4.2 小节, HI_MPI_ISP_CalGainByTemp 【参数】涉及更新。



5.5.2 小节, ISP_AWB_ATTR_S 【定义】涉及更新。

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S 和 ISP_SHARPEN_AUTO_ATTR_S 涉及更新。

6.6.3 小节, ISP_SHADING_LUT_ATTR_S 【注意事项】涉及更新。

6.7.1 小节, 涉及更新。

6.7.2 小节, HI_MPI_ISP_SetDPStaticAttr 【注意】涉及更新。

6.7.3 小节, ISP_DP_STATIC_CALIBRATE_S 【成员】涉及更新。

6.8.4 小节, ISP_CR_ATTR_S 【成员】涉及更新。

6.15.3 小节, ISP_FPN_FRAME_INFO_S 和 ISP_FPN_AUTO_ATTR_S 【注意事项】涉及更新。ISP_FPN_MANUAL_ATTR_S 【成员】涉及更新。

6.18.3 小节, ISP_FSWDR_MDT_ATTR_S、ISP_FUSION_ATTR_S 、ISP_WDR_WBGAIN_POSITION_E 涉及更新。

6.26.2 小节, HI_MPI_ISP_SetLogLUTAttr 【注意】涉及更新。

6.26.3 小节, ISP_LOGLUT_ATTR_S 【注意】涉及更新。

6.28.3 小节, ISP_RADIAL_SHADING_LUT_ATTR_S 涉及更新。

7.4 小节, ISP_AE_CROP_S、ISP_AWB_CROP_S、ISP_WB_STATISTICS_CFG_S、ISP_AE_GRID_INFO_S、ISP_AE_STATISTICS_S、ISP_MG_GRID_INFO_S、ISP_MG_STATISTICS_S、ISP_AWB_GRID_INFO_S、ISP_WB_STATISTICS_S、ISP_AF_CFG_S、ISP_AF_CROP_S、ISP_AF_H_PARAM_S、ISP_FOCUS_GRID_INFO_S、ISP_AF_STATISTICS_S 涉及更新。

6.15.1 小节、6.27.1 小节、10.2 小节, 涉及更新。

文档版本 00B18(2018-11-20)

新增 Hi3518EV300 内容。

第 18 次临时版本发布

2.2 小节, HI_MPI_ISP_GetVDTimeOut 涉及更新。

2.3 小节, ISP_CMOS_SHARPEN_MANUAL_S、ISP_CMOS_SHARPEN_AUTO_S、ISP_CMOS_BAYERNR_S、ISP_CMOS_ALG_KEY_U 、ISP_CMOS_DEFAULT_S、



ISP_SPECKCWB_S、ISP_SPECAWB_CAA_CONVERSION_S 涉及更新。新增
ISP_CMOS_RGBIR_S。

3.5.1 小节, AE_ACCURACY_S 涉及更新。

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S、ISP_SHARPEN_AUTO_ATTR_S 涉及更新。

6.5.3 小节, ISP_DRC_ATTR_S 涉及更新。

6.6.1 小节, 涉及更新。

6.6.3 小节, ISP_SHADING_ATTR_S、ISP_SHADING_GAIN_LUT_S、
ISP_SHADING_LUT_ATTR_S 涉及更新。

6.9.3 小节、ISP_NR_MANUAL_ATTR_S、ISP_NR_AUTO_ATTR_S 涉及更新。

6.13.1 小节, 涉及更新。

6.18.1 小节, 涉及更新。

6.30.3 小节, ISP_INNER_STATE_INFO_S 涉及更新。

7.4 小节, ISP_AE_STATISTICS_CFG_S、ISP_WB_STATISTICS_CFG_S、
ISP_AE_STATISTICS_S、ISP_MG_STATISTICS_S、ISP_WB_STATISTICS_S、
ISP_AF_CROP_S、ISP_AF_STATISTICS_S 涉及更新。新增 ISP_AE_CROP_S、
ISP_AWB_CROP_S、ISP_AE_GRID_INFO_S、ISP_AWB_GRID_INFO_S、
ISP_MG_GRID_INFO_S、ISP_AWB_GRID_INFO_S、ISP_FOCUS_GRID_INFO_S。

10.2 小节, 【调试信息】更新。

文档版本 00B17 (2018-11-05)

第 17 次临时版本发布

新增 Hi3516EV200 和 Hi3516EV300 内容。

2.2 小节, HI_MPI_ISP_MemInit、HI_MPI_ISP_Run 【注意】涉及修改

6.9.3 小节, ISP_NR_ATTR_S 【注意事项】涉及修改

6.10.3 小节, HI_MPI_ISP_SetDISAttr 和 HI_MPI_ISP_GetDISAttr 【注意】涉及修改

6.32.3 小节, ISP_MLSC_CALIBRATION_CFG_S 和 ISP_MESH_SHADING_TABLE_S
【成员】涉及修改

7.4 小节, ISP_WB_STATISTICS_CFG_S、ISP_AF_CFG_S 【注意事项】涉及修改



文档版本 00B16 (2018-10-26)

第 16 次临时版本发布

2.3 小节, SIZE_S、ISP_PUB_ATTR_S、ISP_CMOS_WDR_S、ISP_CTRL_PARAM_S【成员】涉及修改

6.32.2 小节, HI_MPI_ISP_MeshShadingCalibration【注意】涉及修改

6.32.3 小节, ISP_DE_ATTR_S【注意】涉及修改

文档版本 00B15 (2018-10-15)

第 15 次临时版本发布

2.2 小节, HI_MPI_ISP_MemInit、HI_MPI_ISP_Exit、HI_MPI_ISP_SetPubAttr 的【注意】涉及修改

2.3 小节, ISP_CMOS_SHARPEN_MANUAL_S、ISP_CMOS_SHARPEN_AUTO_S、ISP_CMOS_SHARPEN_S、ISP_CMOS_NOISE_CALIBRATION_S、ISP_PIPE_DIFF_ATTR_S 的【成员】涉及修改; Hi3559AV100/Hi3519AV100 的 ISP_CMOS_LSC_S【定义】涉及修改; ISP_AWB_ALG_E 名称涉及修改

3.5.3 小节, ISP_IRIS_ATTR_S 的【注意】涉及修改

6.1.2 小节, HI_MPI_ISP_GetIspSharpenAttr 的【注意】涉及修改

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S、ISP_SHARPEN_AUTO_ATTR_S 的【成员】涉及修改

6.23.3 小节, CSC_MATRX_S 的【成员】涉及修改

10.2 小节涉及修改

文档版本 00B14 (2018-09-29)

第 14 次临时版本发布

2.3 小节, 新增 ISP_CMOS_SHARPEN_MANUAL_S 和 ISP_CMOS_SHARPEN_AUTO_S; ISP_CMOS_SHARPEN_S【定义】和【成员】涉及修改

6.1.3 小节, ISP_SHARPEN_MANUAL_ATTR_S 和 ISP_SHARPEN_AUTO_ATTR_S【定义】和【成员】涉及修改

6.5.3 小节, ISP_DRC_ATTR_S【定义】和【成员】涉及修改



6.15.2 小节, HI_MPI_ISP_FPNCalibrate 和 HI_MPI_ISP_SetFPNAttr 【注意】涉及修改

7.2.3.1 小节涉及修改

添加 Hi3559V200 和 Hi3556V200 的相关内容

文档版本 00B13 (2018-09-03)

第 13 次临时版本发布

添加 Hi3516CV500 和 Hi3516DV300 的相关内容。

文档版本 00B12 (2018-08-08)

第 12 次临时版本发布

2.3 小节, ISP_AE_STAT_ATTR_S 【注意事项】及 ISP_SNS_STATE_S 【成员】涉及修改;

3.4.2 小节, HI_MPI_ISP_SetHDRExposureAttr、HI_MPI_ISP_GetHDRExposureAttr 和 ISP_HDR_EXPOSURE_ATTR_S 【描述】涉及修改

3.5.2 小节, ISP_EXP_INFO_S 【注意事项】涉及修改

6.8.2、6.10.3 和 6.15.2 小节涉及修改

文档版本 00B11 (2018-07-30)

第 11 次临时版本发布

2.2 小节, HI_MPI_ISP_SetPubAttr、HI_MPI_ISP_SetDCFInfo 【注意】涉及修改

2.3 小节, ISP_MODULE_CTRL_U、ISP_CMOS_WDR_S、ISP_CTRL_PARAM_S 的【定义】和【成员】涉及修改

4.5.2 小节, ISP_AWB_CBCR_TRACK_ATTR_S 和 ISP_WB_INFO_S 【注意事项】涉及修改

6.1.3 小节, 表 6-5 涉及修改

6.8.4 小节, ISP_CR_ATTR_S 【定义】和【注意事项】涉及修改

6.11.3 小节, ISP_DEHAZE_ATTR_S 【定义】和【成员】涉及修改

6.18.3 小节, ISP_FSWDR_MDT_ATTR_S 和 ISP_WDR_BNR_ATTR_S 【定义】和【成员】涉及修改



6.31.3 小节, ISP_MESH_SHADING_TABLE_S 【定义】和【成员】涉及修改

7.4 小节, ISP_AF_CROP_S、ISP_AF_STATISTICS_POS_E 涉及修改

6.10.2、10.2 小节涉及修改

文档版本 00B10 (2018-07-06)

第 10 次临时版本发布

1.2.5 涉及修改

2.2 小节, HI_MPI_ISP_SetPubAttr 和 HI_MPI_ISP_GetLightboxGain 【注意】涉及修改; HI_MPI_ISP_SensorRegCallBack 涉及修改

2.3 小节, 新增 ISP_CMOS_ALG_KEY_U 和 ISP_PIPE_DIFF_ATTR_S;

RECT_S、ISP_CMOS_DEMOSAIC_S、ISP_CMOS_SHARPEN_S、ISP_CTRL_CMD_E、ISP_CMOS_BLACK_LEVEL_S 和 ISP_CTRL_PROC_WRITE_S 【定义】和【成员】涉及修改

ISP_CMOS_DEFAULT_S 【注意】涉及修改;

6.1.3 小节, 新增表 6-11~表 6-13; ISP_SHARPEN_ATTR_S 【定义】和【成员】涉及修改

6.5.3 小节, ISP_DRC_ATTR_S 【定义】和【成员】涉及修改

6.6.3 小节, 新增 ISP_BNR_LSC_GAIN_LUT_S; ISP_SHADING_LUT_ATTR_S 【定义】和【成员】涉及修改

6.7.2 小节, HI_MPI_ISP_SetDPStaticAttr 【注意】涉及修改

6.7.3 小节, ISP_DP_STATIC_CALIBRATE_S 涉及修改

6.13.3 小节, ISP_DEMOSAIC_MANUAL_ATTR_S、ISP_DEMOSAIC_AUTO_ATTR_S 【定义】和【成员】涉及修改

7.4 小节, ISP_AE_STITCH_STATISTICS_S、ISP_MG_STATISTICS_S、ISP_WB_STATISTICS_S、ISP_WB_STITCH_STATISTICS_S 和 ISP_AF_STATISTICS_S 【注意】涉及修改

文档版本 00B09 (2018-06-15)

第 9 次临时版本发布



2.2 小节, HI_MPI_ISP_SetSnsSlaveAttr 的【注意】涉及修改

2.3 小节, ISP_SLAVE_SNS_SYNC_S、ISP_MODULE_CTRL_U、ISP_AE_RESULT_S 新增成员描述, ISP_SPECAWB_ATTR_S 的【注意】涉及修改

3.5.1 小节, AE_SENSOR_DEFAULT_S 新增成员描述

6.7.2 小节, HI_MPI_ISP_SetDPCalibrate 的【注意】涉及修改

6.14.2 小节, HI_MPI_ISP_SetBlackLevelAttr 的【注意】涉及修改

6.14.3 小节, ISP_BLACK_LEVEL_S 新增成员描述

6.23.3 小节, ISP_CLUT_LUT_S 的【注意】涉及修改

13.5.11.1 小节, 图 13-15 涉及修改

13.5.12.2 小节, 新增 ISP_DRV_FOCUS_ZONE_S、ISP_DRV_FOCUS_STATISTICS_S 和 ISP_DRV_AF_STATISTICS_S; ISP_SYNC_TASK_NODE_S 涉及修改

新增 6.32 小节

文档版本 00B08 (2018-05-18)

第 8 次临时版本发布

2.3 小节, ISP_MODULE_CTRL_U、ISP_CMOS_DRC_S、ISP_CMOS_DEMOSAIC_S、ISP_CMOS_PREGAMMA_S、ISP_CMOS_SHARPEN_S、ISP_CMOS_LDCI_S、ISP_CMOS_DPC_S 和 ISP_CMOS_DEFAULT_S 【定义】和【成员】涉及修改

删除 ISP_CMOS_DEGAMMA_S; 新增 ISP_CMOS_ALG_KEY_U

3.5.2 小节, ISP_FSWDR_MODE_E 【注意事项】涉及修改

4.5.1 小节, 新增 AWB_SPEC_SENSOR_DEFAULT_S

5.5 小节, ISP_SATURATION_ATTR_S、ISP_COLORMATRIX_AUTO_S 【注意事项】涉及修改

6.6.3 小节, ISP_SHADING_LUT_ATTR_S 【注意事项】涉及修改

6.13.3 小节, ISP_DEMOSAIC_MANUAL_ATTR_S、ISP_DEMOSAIC_AUTO_ATTR_S 【定义】和【成员】涉及修改

6.15.2 小节, HI_MPI_ISP_FPNCalibrate、HI_MPI_ISP_SetFPNAttr 【注意】涉及修改

6.31.1 小节涉及修改



7.4 小节, ISP_AF_CFG_S、ISP_AF_PEAK_MODE_E 和 ISP_AF_STATISTICS_S 【注意事项】涉及修改

10.2 小节涉及修改

文档版本 00B07 (2018-04-11)

第 7 次临时版本发布

添加 Hi3519AV100 的相关内容

文档版本 00B06 (2018-04-04)

第 6 次临时版本发布

2.3 小节, 新增 ISP_AE_STAT_ATTR_S、
ISP_AWB_ALG_TYPE_E~ISP_SPECAWB_CAA_CONVERSION_S

ISP_CMOS_DRC_S 【定义】和【成员】涉及修改

3.5.2 小节, ISP_WDR_EXPOSURE_ATTR_S 【注意事项】涉及修改

4.4.2 小节, 新增 HI_MPI_ISP_SetSpecAwbAttr~HI_MPI_ISP_GetCaaControlAttr

4.5.1 小节, AWB_SENSOR_DEFAULT_S 【定义】和【成员】涉及修改

4.5.2 小节, ISP_WB_ATTR_S 和 ISP_WB_INFO_S 涉及修改

6.4.3 节, ISP_DRC_AUTO_ATTR_S 涉及修改

6.5.3 小节, ISP_SHADING_LUT_ATTR_S 的【成员】和【注意事项】涉及修改

6.22.3 小节, ISP_CLUT_LUT_S 的【注意事项】涉及修改

文档版本 00B05 (2018-03-15)

第 5 次临时版本发布

2.2 小节, HI_MPI_ISP_SensorRegCallBack、HI_MPI_ISP_SensorUnRegCallBack、
HI_MPI_ISP_AELibRegCallBack、HI_MPI_ISP_AELibUnRegCallBack、
HI_MPI_ISP_AWBLibRegCallBack、HI_MPI_ISP_AWBLibUnRegCallBack 的【注意】
涉及修改

2.3 小节, RECT_S、ISP_MODULE_CTRL_U、ISP_VD_TYPE_E、ISP_CMOS_DRC_S、
ISP_CMOS_ISP_SHARPEN_S、ISP_CMOS_NOISE_CALIBRATION_S、



ISP_CMOS_WDR_S、ISP_AWB_INFO_S、ISP_AWB_RESULT_S、
HI_MPI_ISP_SetAERouteAttr、HI_MPI_ISP_SetAERouteAttrEx 涉及修改

3.5.2 小节，ISP_AE_STRATEGY_E 的【注意】涉及修改

4.4.2 小节，HI_MPI_ISP_SetWBAttr 的【注意】涉及修改

6.1.3 小节，ISP_SHARPEN_MANUAL_ATTR_S、ISP_SHARPEN_AUTO_ATTR_S 的【成员】涉及修改

6.2.1 小节涉及修改

6.5.3 小节，ISP_SHADING_LUT_ATTR_S 的【成员】涉及修改

6.6.3 小节，ISP_DP_STATIC_CALIBRATE_S、ISP_DP_STATIC_ATTR_S 的【成员】涉及修改

6.30.3 小节，ISP_MLSC_CALIBRATION_CFG_S 涉及修改

7.4 小节，ISP_STATISTICS_CTRL_U 的【注意】涉及修改

10.1、10.2 小节涉及修改

13.5.10、13.5.12 小节涉及修改

文档版本 00B04 (2018-02-10)

2.2 小节，新增 HI_MPI_ISP_SetModParam，HI_MPI_ISP_GetModParam 和
HI_MPI_ISP_GetLightboxGain；HI_MPI_ISP_SetBindAttr、HI_MPI_ISP_GetBindAttr
和【注意】涉及修改

2.3 小节，新增 ISP_LSC_CABLI_TABLE_S、ISP_AWB_Calibration_Gain_S 和
ISP_MOD_PARAM_S；ISP_CMOS_RLSC_S、ISP_CMOS_BAYERNR_S、
ISP_AE_INFO_S 和 ISP_AE_RESULT_S【成员】涉及修改

6.4.3、6.8.3、6.26.1 和 7.4 涉及修改

6.26.3 小节，ISP_RADIAL_SHADING_LUT_ATTR_S 涉及修改

新增 7.2 小节

文档版本 00B03 (2018-01-30)

第三次临时版本发布。

2.2 小节，HI_MPI_ISP_SetCtrlParam 和 HI_MPI_ISP_GetCtrlParam 涉及修改



2.3 小节, ISP_CTRL_PARAM_S 涉及修改

新增 6.30 小节

文档版本 00B02 (2018-01-15)

第二次临时版本发布。

2.3、3.5、5.4、6.1 及 7.3 小节均涉及修改

文档版本 00B01 (2017-11-15)

第一次临时版本发布。



目 录

前 言	i
1 概述	1
1.1 概述	1
1.2 功能描述	1
1.2.1 架构	2
1.2.2 开发模式	2
1.2.3 内部流程	3
1.2.4 软件流程	4
1.2.5 文件组织	8
2 系统控制	9
2.1 功能概述	9
2.2 API 参考	9
2.3 数据类型	61
3 AE	238
3.1 概述	238
3.2 重要概念	239
3.3 功能描述	239
3.4 API 参考	241
3.4.1 AE 库接口	241
3.4.2 AE 控制模块	247
3.4.3 AI 控制模块	278
3.5 数据类型	286
3.5.1 Register	286



3.5.2 AE	299
3.5.3 AI	337
4 AWB	354
4.1 概述	354
4.2 重要概念	354
4.3 功能描述	355
4.4 API 参考	356
4.4.1 AWB 库接口	356
4.4.2 AWB 控制模块	361
4.5 数据类型	375
4.5.1 Register	375
4.5.2 WB	381
5 CCM	409
5.1 概述	409
5.2 重要概念	409
5.3 功能描述	410
5.4 API 参考	410
5.5 数据类型	414
6 IMP	423
6.1 Sharpen	423
6.1.1 功能描述	423
6.1.2 API 参考	424
6.1.3 数据类型	426
6.2 EdgeMark	440
6.2.1 功能描述	440
6.2.2 API 参考	440
6.2.3 数据类型	442
6.3 HLC	443
6.3.1 功能描述	443
6.3.2 API 参考	444
6.3.3 数据类型	446
6.4 Gamma	447



6.4.1 功能描述.....	447
6.4.2 API 参考	447
6.4.3 数据类型.....	449
6.5 DRC.....	451
6.5.1 功能描述.....	451
6.5.2 API 参考	452
6.5.3 数据类型.....	454
6.6 Mesh Shading.....	469
6.6.1 功能描述.....	469
6.6.2 API 参考	469
6.6.3 数据类型.....	473
6.7 Defect Pixel.....	480
6.7.1 功能描述.....	480
6.7.2 API 参考	481
6.7.3 数据类型.....	489
6.8 Crosstalk Removal.....	499
6.8.1 概述.....	499
6.8.2 功能描述.....	499
6.8.3 API 参考	500
6.8.4 数据类型.....	502
6.9 去噪算法	504
6.9.1 功能描述.....	504
6.9.2 API 参考	504
6.9.3 数据类型.....	506
6.10 2 轴 DIS.....	512
6.10.1 概述.....	512
6.10.2 功能描述	512
6.10.3 API 参考	513
6.10.4 数据类型	517
6.11 Dehaze	518
6.11.1 功能描述	518
6.11.2 API 参考	518



6.11.3 数据类型	520
6.12 去伪彩	523
6.12.1 功能描述	523
6.12.2 API 参考	523
6.12.3 数据类型	525
6.13 去马赛克	528
6.13.1 功能描述	528
6.13.2 API 参考	529
6.13.3 数据类型	531
6.14 黑电平	537
6.14.1 功能描述	537
6.14.2 API 参考	537
6.14.3 数据类型	539
6.15 去 FPN	540
6.15.1 功能描述	540
6.15.2 API 参考	542
6.15.3 数据类型	547
6.16 CA	552
6.16.1 功能描述	552
6.16.2 API 参考	553
6.16.3 数据类型	555
6.17 CAC	558
6.17.1 功能描述	558
6.17.2 API 参考	559
6.17.3 数据类型	563
6.18 WDR	570
6.18.1 功能描述	570
6.18.2 API 参考	572
6.18.3 数据类型	574
6.19 AWB 量产标定工具	586
6.19.1 功能描述	586
6.19.2 API 参考	587



6.19.3 数据类型	587
6.20 ColorTone	588
6.20.1 功能描述	588
6.20.2 API 参考	588
6.20.3 数据类型	590
6.21 LDCI	591
6.21.1 功能描述	591
6.21.2 API 参考	591
6.21.3 数据类型	593
6.22 Radial Crop	598
6.22.1 功能描述	598
6.22.2 API 参考	598
6.22.3 数据类型	600
6.23 CSC	601
6.23.1 功能描述	601
6.23.2 API 参考	601
6.23.3 数据类型	603
6.24 CLUT	606
6.24.1 功能描述	606
6.24.2 API 参考	607
6.24.3 数据类型	611
6.25 PreLogLUT	621
6.25.1 功能描述	621
6.25.2 API 参考	622
6.25.3 数据类型	624
6.26 LogLUT	625
6.26.1 功能描述	625
6.26.2 API 参考	625
6.26.3 数据类型	627
6.27 PreGamma	628
6.27.1 功能描述	628
6.27.2 API 参考	628
6.27.3 数据类型	630



6.28 Radial Shading	631
6.28.1 功能描述	631
6.28.2 API 参考	631
6.28.3 数据类型	636
6.29 获取 ISP 模块虚拟地址	642
6.29.1 功能描述	642
6.29.2 API 参考	642
6.29.3 数据类型	643
6.30 查询内部状态信息	644
6.30.1 功能描述	644
6.30.2 API 参考	644
6.30.3 数据类型	646
6.31 DNG	649
6.31.1 功能描述	649
6.31.2 API 参考	649
6.31.3 数据类型	653
6.32 MeshShading 量产标定工具	661
6.32.1 功能描述	661
6.32.2 API 参考	662
6.32.3 数据类型	663
6.33 RadialShading 量产标定工具	667
6.33.1 功能描述	667
6.33.2 API 参考	667
6.33.3 数据类型	668
6.34 DE	672
6.34.1 功能描述	672
6.34.2 API 参考	672
6.34.3 数据类型	674
6.35 RGB-IR	677
6.35.1 功能描述	677
6.35.2 API 参考	678
6.35.3 数据类型	680



6.36 Auto Flicker Type Detection	684
6.36.1 功能描述	684
6.36.2 API 参考	684
6.36.3 数据类型	686
6.37 Auto Color Shading.....	688
6.37.1 功能描述	688
6.37.2 API 参考	689
6.37.3 数据类型	691
7 统计信息.....	693
7.1 概述.....	693
7.2 功能描述	693
7.2.1 AE 统计信息	693
7.2.2 MG 统计信息.....	696
7.2.3 AWB 统计信息.....	697
7.3 API 参考	699
7.4 数据类型	707
8 Debug	756
8.1 概述.....	756
8.2 功能描述	756
8.3 API 参考	756
8.4 数据类型	759
9 错误码.....	761
10 Proc 调试信息说明.....	762
10.1 概述	762
10.2 ISP.....	763
11 3A 开发指南概述.....	781
11.1 概述	781
11.2 功能描述.....	781
12 3A 开发使用者指南	782
12.1 软件流程	782



12.2 Sensor 对接.....	782
13 开发者指南.....	783
13.1 概述.....	783
13.2 AE 算法注册 ISP 库.....	783
13.3 AWB 算法注册 ISP 库.....	785
13.4 开发用户 AF 算法.....	786
13.5 AF 统计信息使用说明.....	788
13.5.1 概述.....	788
13.5.2 输入图像的裁剪.....	790
13.5.3 Bayer 域配置.....	790
13.5.4 抑制光源对 FV 值的影响.....	791
13.5.5 Coring.....	792
13.5.6 统计模式配置.....	793
13.5.7 统计信息配置注意事项.....	795
13.5.8 统计值的获取.....	796
13.5.9 FV 值计算.....	796
13.5.10 FV 计算参考代码.....	796
13.5.11 PQTools 滤波器设计工具使用说明.....	802
13.5.12 AF 同步回调.....	807
13.6 不带光敏 AE 快速收敛模式（无光敏快启）使用说明.....	815
13.6.1 概述.....	815
13.6.2 适配不带光敏 AE 快速收敛模式（无光敏快启）.....	815
13.6.3 注意事项.....	816
13.6.4 带 IR CUT 无光敏快启基本流程.....	817
14 附录.....	818
14.1 注册函数的关系.....	818
14.2 扩展性的设计考虑.....	819
14.3 3A 架构的设计思路.....	819
14.4 外部寄存器的说明.....	820
14.5 工频闪类型自适应 Sample 用例处理流程.....	821



图目录

图 1-1 ISP 控制结构示意图	2
图 1-2 ISP firmware 架构	2
图 1-3 ISP firmware 内部流程	3
图 1-4 ISP firmware 软件结构	4
图 1-5 ISP firmware 使用流程	5
图 1-6 ISP firmware 文件组织	8
图 2-1 ISP 库与 sensor 库间的接口	35
图 2-2 ISP 库与 AE 库间的接口	38
图 2-3 ISP 库与 AWB 库间的接口	41
图 2-4 同步信号配置时序图	76
图 2-5 同步信号极性翻转	76
图 2-6 同步信号使能	77
图 2-7 白色区域选择相关参数	200
图 3-1 AE 模块工作流程图	239
图 3-2 AE 1024 段统计信息直方图	240
图 3-3 AE 工作原理图	240
图 3-4 AE 库与 sensor 库间的接口	244
图 3-5 AE 分配路线示意图	261
图 4-1 AWB 工作原理图	355
图 4-2 AWB 库 与 sensor 库间的接口	359



图 4-3 色温曲线的参数示意.....	400
图 4-4 混合光源场景色温权重设置示例说明 (n 为色温分段点个数)	400
图 4-5 室外色温范围参数的意义	406
图 5-1 CCM 矩阵.....	410
图 6-1 Sharpen 的像素亮度 luma 和锐化强度 lumaWgt 的关系曲线.....	431
图 6-2 TextureStr[ISP_SHARPEN_GAIN_NUM]强度曲线示意图.....	432
图 6-3 EdgeStr[ISP_SHARPEN_GAIN_NUM]强度曲线示意图.....	432
图 6-4 动态范围压缩	452
图 6-5 u8Asymmetry 的变化对曲线的影响趋势.....	467
图 6-6 u8SecondPole 的变化对曲线的影响趋势.....	467
图 6-7 u8Stretch 的变化对曲线的影响趋势	468
图 6-8 u8Compress 的变化对曲线的影响趋势	468
图 6-9 Crosstalk 门限	499
图 6-10 DIS 偏移示意图	513
图 6-11 去马赛克处理图.....	529
图 6-12 FPN 标定示意图.....	541
图 6-13 FPN 校正示意图.....	542
图 6-14 色差图解.....	559
图 6-15 WDR 算法合成模型.....	571
图 6-16 AWB 小规模试产过程中确定 GoldenSample 流程图.....	587
图 6-17 COLOR_3D_LUT 示意图.....	607
图 6-18 传统 RGB-IR 2x2 pattern 排列.....	677
图 6-19 RGB-IR 4x4 pattern 排列.....	677
图 7-1 AWB 有效像素颜色空间	697
图 7-2 Square 模式.....	742
图 12-1 Sensor 适配示意图	782



图 13-1 AE 算法向 ISP 库注册的回调函数.....	784
图 13-2 AWB 算法向 ISP 库注册的回调函数.....	785
图 13-3 AF 算法结构图.....	787
图 13-4 FV 曲线.....	788
图 13-5 AF 统计模块框图.....	789
图 13-6 AF 4 组滤波器	789
图 13-7 AF 输入图像裁剪.....	790
图 13-8 Bayer 域统计预处理.....	790
图 13-9 关于 Gamma 配置的说明.....	791
图 13-10 高亮统计值.....	792
图 13-11 Level Depend Gain 曲线.....	792
图 13-12 Coring 曲线.....	793
图 13-13 平方配置模式映射曲线图.....	794
图 13-14 滤波器后的图像	795
图 13-15 自动对焦参数仿真工具.....	803
图 13-16 通过截止频率获取滤波系数（窄带）	804
图 13-17 通过截止频率获取滤波系数（宽带）	805
图 13-18 参考组列表.....	806
图 13-19 参考组的曲线显示.....	806
图 13-20 选中参考组曲线高亮显示.....	807
图 13-21 AF 同步回调.....	808
图 14-1 工频闪类型自适应 Sample 用例处理流程.....	821



表目录

表 2-1 统计信息 ISP_AE_INFO_S 的成员变量默认配置说明	181
表 3-1 P-Iris 步进电机位置与 F 值映射表, 以福光 NV03105P 为例	347
表 3-2 P-Iris 镜头相关参数, 以福光 NV03105P 为例	348
表 4-1 au16CrMax [16]在不同的增益情况下的设置值 (仅供参考)	387
表 4-2 au16CrMin [16]在不同的增益情况下的设置值 (仅供参考)	388
表 4-3 au16CbMax [16]在不同的增益情况下的设置值 (仅供参考)	389
表 4-4 au16CbMin [16]在不同的增益情况下的设置值 (仅供参考)	389
表 5-1 au8Sat[16]在不同的增益情况下的设置值, 以 mn34220 为例	417
表 6-1 u8MeshScale 与增益精度对应关系	480
表 6-2 au16Strength[16]在不同增益情况下对应的设置值	496
表 6-3 u16Strength [ISP_AUTO_STRENGTH_NUM]在不同的增益情况下的设置值	503
表 6-4 au8DePurpleCrStr[16]在不同曝光比情况下对应的设置值	565
表 6-5 u8RadialScale 与增益精度对应关系	640
表 7-1 AE 区域统计信息	694
表 7-2 AE 直方图统计信息	694
表 7-3 AE 直方图统计信息	695
表 7-4 MG 区域统计信息	696
表 7-5 AWB 区域统计信息	698
表 7-6 AWB 区域统计信息 (以 au16ZoneAvgR 为例, u16ZoneRow=32, u16ZoneCol=32, u16ZoneBin=4)	733



表 7-7 AWB 区域统计信息 (以 au16ZoneAvgR 为例, u16ZoneRow=32, u16ZoneCol=32, Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持 1 个 bin)	734
表 9-1 ISP API 错误码	761
表 13-1 AE 算法向 ISP 库注册的回调函数	784
表 13-2 AWB 算法向 ISP 库注册的回调函数	786
表 13-3 统计信息配置注意事项	795



1 概述

1.1 概述

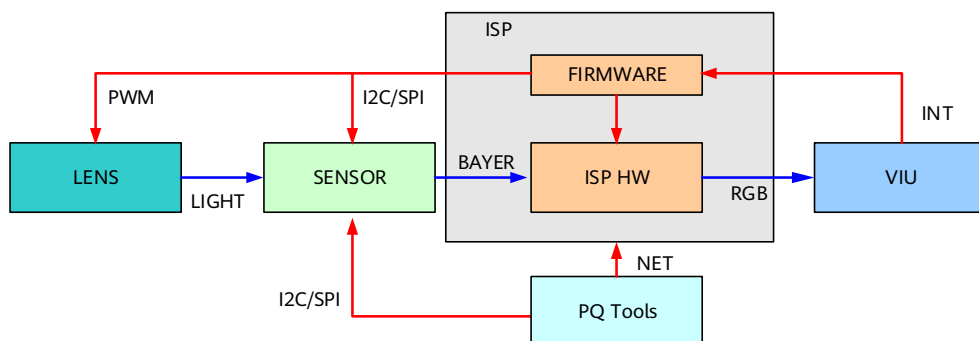
ISP 通过一系列数字图像处理算法完成对数字图像的效果处理。主要包括 3A、坏点校正、去噪、强光抑制、背光补偿、色彩增强、镜头阴影校正等处理。ISP 包括逻辑部分以及运行在其上的 firmware。这里主要介绍 ISP 的用户接口。

1.2 功能描述

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

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

图1-1 ISP 控制结构示意图

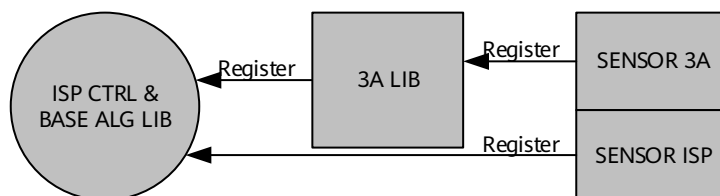


ISP 逻辑主要流程、具体概念和功能点请参见芯片手册。

1.2.1 架构

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

图1-2 ISP firmware 架构



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

1.2.2 开发模式

SDK 支持用户使用多种开发模式：



用户使用上海海思的 3A 算法库。这时用户需要根据 ISP 基础算法库和 3A 算法库给出的 sensor 适配接口去适配不同的 sensor。每款 sensor 对应一个文件夹，文件夹中包含两个主要文件：

- sensor_cmos.c

该文件中主要实现 ISP 需要的回调函数，这些回调函数中包含了 sensor 的适配算法，不同的 sensor 可能有所不同。

- sensor_ctrl.c

sensor 的底层控制驱动，主要实现 sensor 的读写和初始化动作。用户可以根据 sensor 的 datasheet 进行这两个文件的开发，必要的时候可以向 sensor 厂家寻求支持。

用户根据 ISP 库提供的 3A 算法注册接口，实现自己的 3A 算法库开发。这时用户需要根据 ISP 基础算法库和用户的 3A 算法库给出的 sensor 适配接口去适配不同的 sensor。

用户部分使用上海海思 3A 算法库，部分实现自己的 3A 算法库。例如 AE 使用 lib_hiaa.a，AWB 使用自己的 3A 算法库。SDK 提供了灵活多变的支持方式。

1.2.3 内部流程

Firmware 内部流程分两部分，如图 1-3 所示。一部分是初始化任务，主要完成 ISP 控制单元的初始化、ISP 基础算法库的初始化、3A 算法库的初始化，包括调用 sensor 的回调获取 sensor 差异化的初始化参数；另一部分是动态调节过程，在这个过程中，firmware 中的 ISP 控制单元调度 ISP 基础算法库和 3A 算法库，实时计算并进行相应控制。Firmware 的软件结构如图 1-4 所示。

图1-3 ISP firmware 内部流程

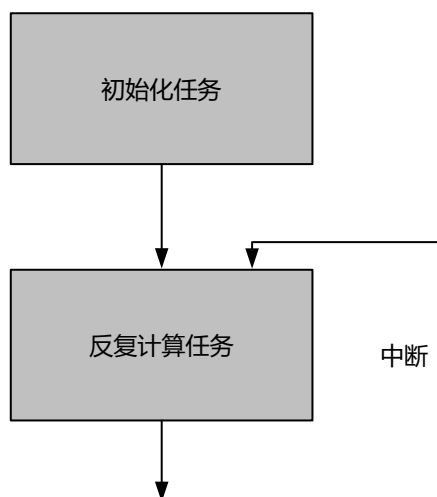
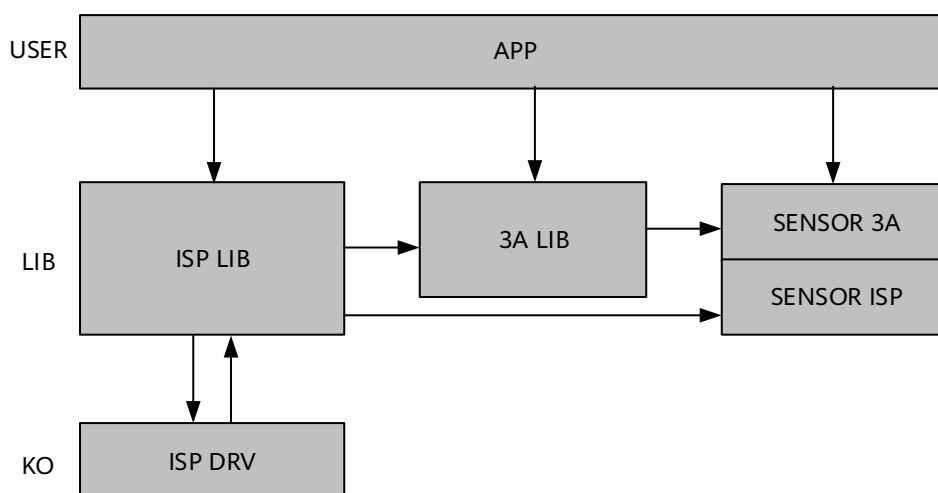


图1-4 ISP firmware 软件结构

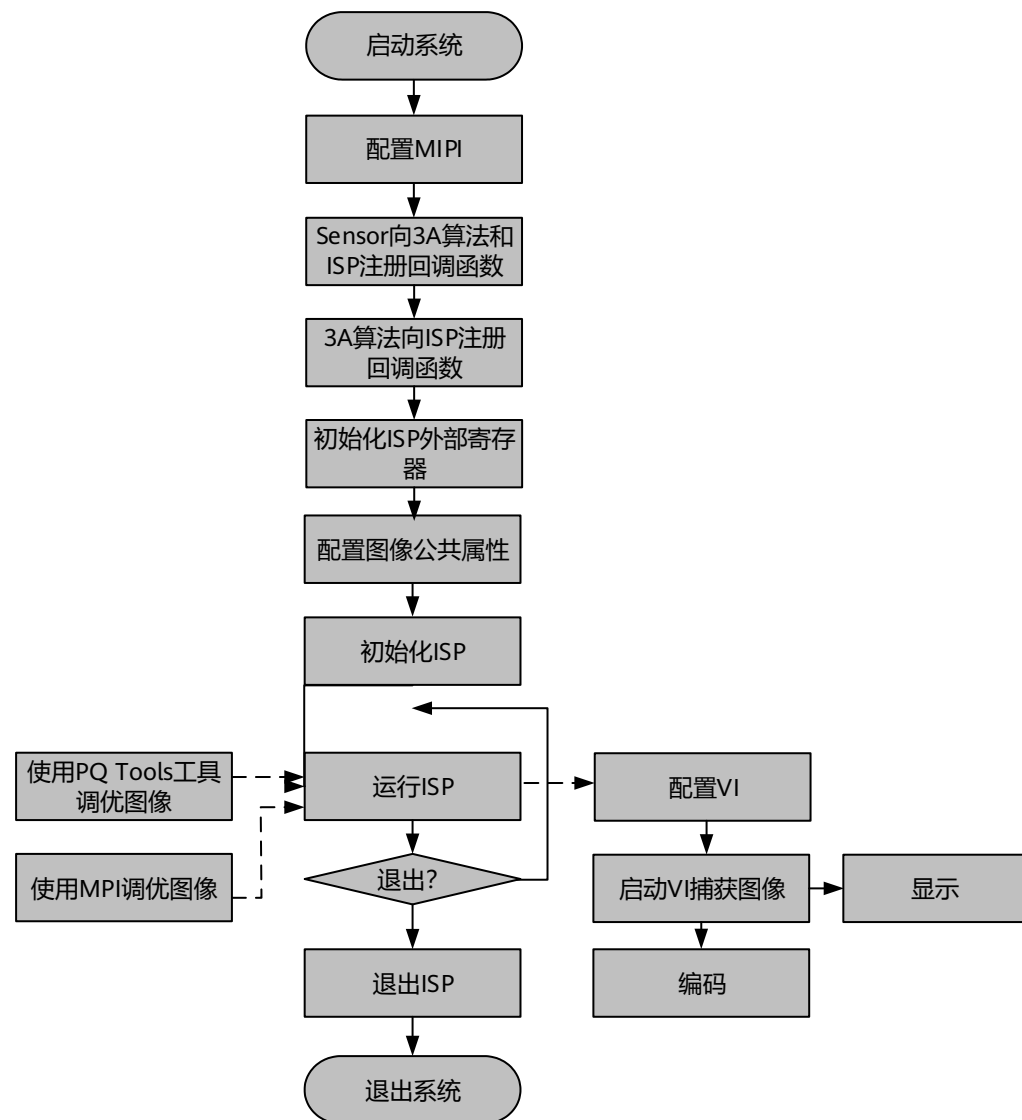


1.2.4 软件流程

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

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

图1-5 ISP firmware 使用流程



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

代码示例：

```
HI_S32 s32Ret;  
ALG_LIB_S stAeLib;  
ALG_LIB_S stAwbLib;  
ISP_PUB_ATTR_S stPubAttr;  
pthread_t isp_pid;  
VI_PIPE ViPipe = 0;
```



```
/* 注册sensor库 */  
  
s32Ret = sensor_register_callback(ViPipe, &stAeLib, &stAwbLib);  
if (HI_SUCCESS != s32Ret)  
{  
    printf("register sensor failed!\n");  
    return s32Ret;  
}  
  
/* 注册上海海思AE算法库 */  
  
stAeLib.s32Id = 0;  
strcpy(stAeLib.acLibName, HI_AE_LIB_NAME);  
s32Ret = HI_MPI_AE_Register(ViPipe, &stAeLib);  
if (HI_SUCCESS != s32Ret)  
{  
    printf("register ae lib failed!\n");  
    return s32Ret;  
}  
  
/* 注册上海海思AWB算法库 */  
  
stAwbLib.s32Id = 0;  
strcpy(stAwbLib.acLibName, HI_AWB_LIB_NAME);  
s32Ret = HI_MPI_AWB_Register(ViPipe, &stAwbLib);  
if (HI_SUCCESS != s32Ret)  
{  
    printf("register awb lib failed!\n");  
    return s32Ret;  
}  
  
/* 初始化ISP外部寄存器 */  
  
s32Ret = HI_MPI_ISP_MemInit(ViPipe);  
if (s32Ret != HI_SUCCESS)  
{  
    printf("%s: HI_MPI_ISP_Init failed!\n", __FUNCTION__);  
    return s32Ret;  
}
```



```
/* 配置图像公共属性 */

s32Ret = HI_MPI_ISP_SetPubAttr(ViPipe, &stPubAttr);
if (s32Ret != HI_SUCCESS)
{
printf("%s: HI_MPI_ISP_SetPubAttr failed with %#x!\n", __FUNCTION__, s32Ret);
return s32Ret;
}

/* 初始化ISP Firmware */

s32Ret = HI_MPI_ISP_Init(ViPipe);
if (HI_SUCCESS != s32Ret)
{
printf("isp init failed!\n");
return s32Ret;
}

/* HI_MPI_ISP_Run单独启动线程运行 */

if (0 != pthread_create(&isp_pid, 0, ISP_Run, NULL))
{
printf("create isp running thread failed!\n");
return HI_FAILURE;
}

/* 启动VI/VO等业务 */

.....

/* 停止VI/VO等业务 */

s32Ret = HI_MPI_ISP_Exit(ViPipe);
if (HI_SUCCESS != s32Ret)
{
printf("isp exit failed!\n");
return s32Ret;
}

pthread_join(isp_pid, 0);
return HI_SUCCESS;
```



说明

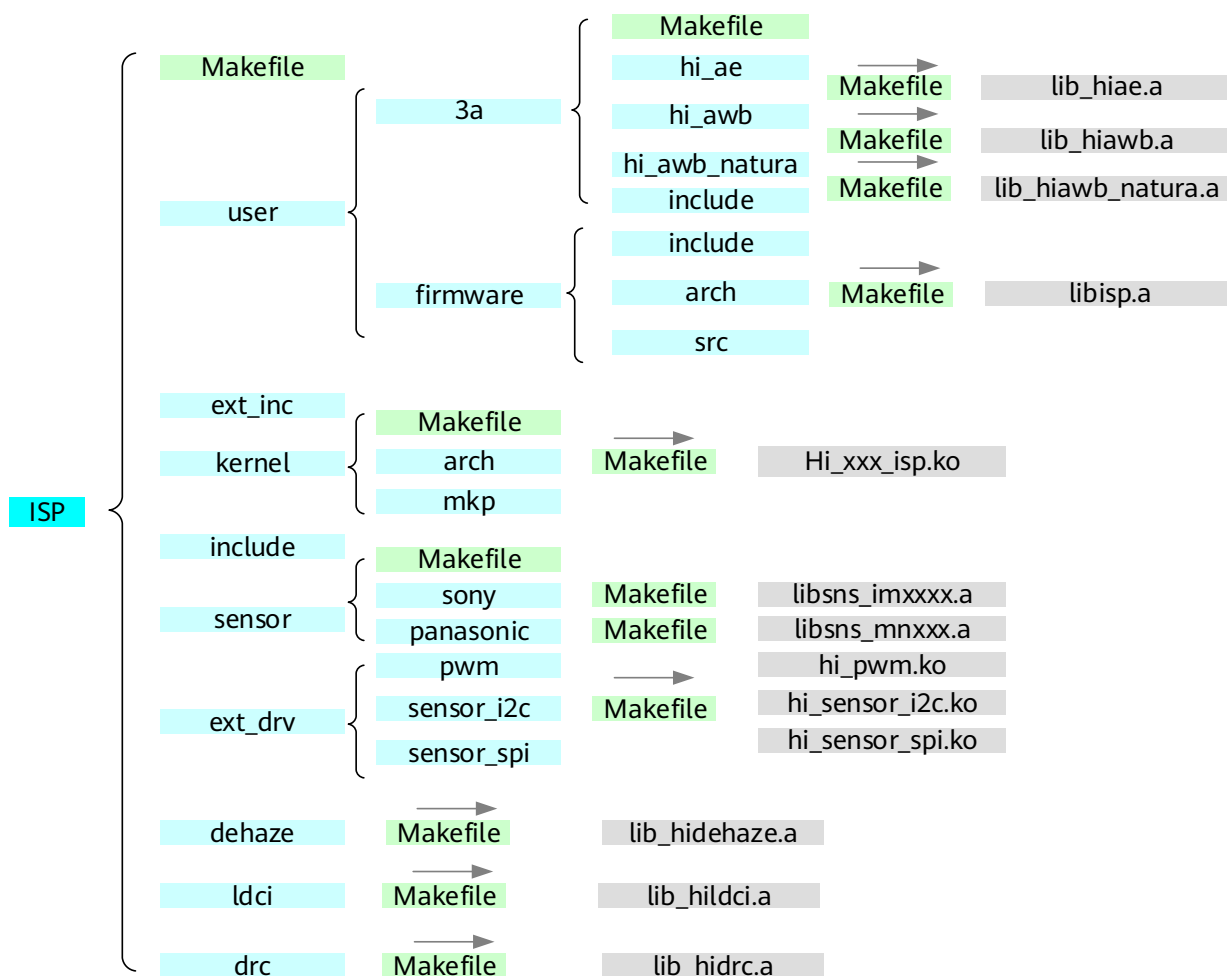
AE 库有用到标准 C 库的数学库，请使用者在 Makefile 中增加 -lm 编译条件。

1.2.5 文件组织

ISP Firmware 的文件组织结构如图 1-6 所示，ISP 库和 3A 库、sensor 库、dehaze 库、ldci 库、drc 库分别独立。Firmware 中的 drv 生成的驱动程序向用户态上报 ISP 中断，并以该中断驱动 Firmware 的 ISP 控制单元运转。ISP 控制单元从驱动程序中获取统计信息，并调度基础算法单元和 3A 算法库，最后通过驱动程序配置寄存器。

Src 文件夹中包含 ISP 控制单元和基础算法单元，编译后生成 libisp.a，即 ISP 库。3a 文件夹中包含 AE/AWB 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。dehaze 文件对应去雾算法程序，ldci 文件对应局域自动对比度增强算法程序，drc 文件夹对应动态范围压缩算法程序，该部分代码不开源。

图1-6 ISP firmware 文件组织





2 系统控制

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

2.1 功能概述

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

2.2 API 参考

本文档中接口，如无特殊说明，支持多进程。

- [HI_MPI_ISP_MemInit](#): 初始化 ISP 外部寄存器。
- [HI_MPI_ISP_Init](#): 初始化 ISP firmware。
- [HI_MPI_ISP_Run](#): 运行 ISP firmware。
- [HI_MPI_ISP_RunOnce](#): 运行 ISP firmware 一次。
- [HI_MPI_ISP_Exit](#): 退出 ISP firmware。
- [HI_MPI_ISP_SetPubAttr](#): 设置 ISP 公共属性。



- [HI_MPI_ISP_GetPubAttr](#): 获取 ISP 公共属性。
- [HI_MPI_ISP_SetFMWState](#): 设置 ISP firmware 状态。
- [HI_MPI_ISP_GetFMWState](#): 获取 ISP firmware 状态。
- [HI_MPI_ISP_SetSnsSlaveAttr](#): 设置从模式 sensor 行场同步信号。
- [HI_MPI_ISP_GetSnsSlaveAttr](#): 获取从模式 sensor 行场同步信号。
- [HI_MPI_ISP_SetModuleControl](#): 设定 ISP 功能模块的控制。
- [HI_MPI_ISP_GetModuleControl](#): 获取 ISP 功能模块的控制。
- [HI_MPI_ISP_SetRegister](#): 设置寄存器值。
- [HI_MPI_ISP_GetRegister](#): 获取寄存器值。
- [HI_MPI_ISP_GetVDTimeOut](#): 获取 ISP 中断信息。
- [HI_MPI_ISP_SensorRegCallBack](#): ISP 提供的 sensor 注册的回调接口。
- [HI_MPI_ISP_SensorUnRegCallBack](#): ISP 提供的 sensor 反注册的回调接口。
- [HI_MPI_ISP_AELibRegCallBack](#): ISP 提供的 AE 库注册的回调接口。
- [HI_MPI_ISP_AELibUnRegCallBack](#): ISP 提供的 AE 库反注册的回调接口。
- [HI_MPI_ISP_AWBLibRegCallBack](#): ISP 提供的 AWB 库注册的回调接口。
- [HI_MPI_ISP_AWBLibUnRegCallBack](#): ISP 提供的 AWB 库反注册的回调接口。
- [HI_MPI_ISP_SetBindAttr](#): 设置 ISP 库与 3A 库、sensor 的绑定关系。
- [HI_MPI_ISP_GetBindAttr](#): 获取 ISP 库与 3A 库、sensor 的绑定关系。
- [HI_MPI_ISP_SetDCFInfo](#): 设置 DCF 参数。
- [HI_MPI_ISP_GetDCFInfo](#): 获取 DCF 参数。
- [HI_MPI_ISP_SetPipeDifferAttr](#): 设置多路 ISP Pipe 差异属性。
- [HI_MPI_ISP_GetPipeDifferAttr](#): 获取多路 ISP Pipe 差异属性。
- [HI_MPI_ISP_SetCtrlParam](#): 设置 ISP 的控制参数。
- [HI_MPI_ISP_GetCtrlParam](#): 获取 ISP 的控制参数。
- [HI_MPI_ISP_SetModParam](#): 设置 ISP 模块参数。
- [HI_MPI_ISP_GetModParam](#): 获取 ISP 模块参数。
- [HI_MPI_ISP_SetSmartInfo](#): 向 ISP 模块传递智能信息。
- [HI_MPI_ISP_GetSmartInfo](#): 获取 ISP 模块参数。
- [HI_MPI_ISP_GetLightboxGain](#): 获取 AWB 在线标定得到的增益结构体。
- [HI_MPI_ISP_SetRawPos](#): 设置采集 Raw 的位置。



- [HI_MPI_ISP_GetRawPos](#): 获取采集 Raw 的位置。
- [HI_MPI_ISP_IrAutoRunOnce](#): 运行红外自动切换功能。

HI_MPI_ISP_MemInit

【描述】

初始化 ISP 外部寄存器。

【语法】

```
HI_S32 HI_MPI_ISP_MemInit(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	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】

- 外部寄存器初始化前需要确保 ko 已加载，sensor 向 ISP 注册了回调函数。
- 调用本接口后，才能调用 [HI_MPI_ISP_SetPubAttr](#) 图像公共属性。
- 不支持多进程，必须要与 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Run](#)、[HI_MPI_ISP_Exit](#) 接口在同一个进程调用。
- 当前业务正在运行 [HI_MPI_ISP_Run](#) 时，不能调用本接口。



- 推荐调用 [HI_MPI_ISP_Exit](#) 后，再调用本接口重新初始化。
- Huawei LiteOS 没有内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk_init.c 中执行的相关过程。
- 不支持相同 ViPipe 时，多线程执行 ISP 创建和销毁（多线程同时调用 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Exit](#)）
- Hi3556V200/Hi3516CV500/Hi3516EV200/Hi3516EV300 ISP 初始化后，需要一帧时间给硬件读取算法系数表。所以 [HI_MPI_ISP_Init](#) 后一帧时间内，不能调用 [HI_MPI_VI_StopPipe](#) 接口停止 pipe。

【举例】

无

【相关主题】

[HI_MPI_ISP_Exit](#)

HI_MPI_ISP_Init

【描述】

初始化 ISP firmware。

【语法】

```
HI_S32 HI_MPI_ISP_Init(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	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】

- 初始化前需要确保 ko 已加载，sensor 向 ISP 注册了回调函数。
- 初始化前需要确保已调用 [HI_MPI_ISP_MemInit](#) 初始化 ISP 外部寄存器。
- 初始化前需要确保已调用 [HI_MPI_ISP_SetPubAttr](#) 图像公共属性。
- 不支持多进程，必须要与 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Run](#)、[HI_MPI_ISP_Exit](#) 接口在同一个进程调用。
- 不支持重复调用本接口。
- 推荐调用 [HI_MPI_ISP_Exit](#) 后，再调用本接口重新初始化。
- Huawei LiteOS 没内核模块加载概念，Linux load ko 过程对应 Huawei LiteOS release/ko 下 sdk_init.c 中执行的相关过程。
- 不支持相同 ViPipe 时，多线程执行 ISP 创建和销毁（多线程同时调用 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Exit](#)）
- Hi3556V200/Hi3516CV500/Hi3516EV200/Hi3516EV300 ISP 初始化后，需要一帧时间给硬件读取算法系数表。所以 [HI_MPI_ISP_Init](#) 后一帧时间内，不能调用 [HI_MPI_VI_StopPipe](#) 接口（请参考《HiMPP 媒体处理软件 V4.0 开发参考》的视频输入章节）停止 pipe。

【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(VI_PIPE ViPipe);
```



【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	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】

- 运行前需要确保 sensor 已经初始化，并且向 ISP 注册了回调函数。
- 运行前需要确保已调用 [HI_MPI_ISP_Init](#) 初始化 ISP。
- 不支持多进程，必须要与 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Exit](#) 接口在同一个进程调用。
- 该接口是阻塞接口，建议用户采用实时线程处理。

【举例】

无

【相关主题】

[HI_MPI_ISP_Init](#)

HI_MPI_ISP_RunOnce

【描述】

运行 ISP firmware 一次。



【语法】

```
HI_S32 HI_MPI_ISP_RunOnce(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	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】

- 运行前需要确保 sensor 已经初始化，并且向 ISP 注册了回调函数。
- 运行前需要确保已调用 [HI_MPI_ISP_Init](#) 初始化 ISP。
- 不支持多进程，必须要与 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Exit](#) 接口在同一个进程调用。
- 该接口是阻塞接口，建议用户采用实时线程处理。
- 该接口主要用于离线模式，用户给 BE 灌 RAW。在使用时，要等待上一次发送的 RAW 数据处理完成之后才能进行下一次的 [HI_MPI_ISP_RunOnce](#) 接口调用+发送 RAW 数据（可通过在 [HI_MPI_VI_SendPipeRaw](#) 后调用接口 [HI_MPI_VI_GetChnFrame](#) 实现，接口具体信息请参见《HiMPP 媒体处理软件 V4.0 开发参考》的 VI 章节），具体参考【举例】中的伪代码。
- 使用 [HI_MPI_ISP_RunOnce](#) 模式时处理视频流时，支持模式切换及分辨率切换，切换流程与使用 [HI_MPI_ISP_Run](#) 处理视频流类似：即切换过程中 ISP 模块无需退



出，VI 模块需要销毁重建。区别在于：使用 HI_MPI_ISP_RunOnce 处理视频流时需要用户创建一个线程，参考示例中伪代码。

- [HI_MPI_ISP_Run](#) 和 [HI_MPI_ISP_RunOnce](#) 对同一个 ViPipe 不能同时使用。

【举例】

1. 上一次发送的raw数据处理完，才能继续调用HI_MPI_ISP_RunOnce:

```
.....  
s32Ret = HI_MPI_ISP_RunOnce(ViPipe);  
    if (HI_SUCCESS != s32Ret) {  
        SAMPLE_PRT("HI_MPI_ISP_RunOnce failed with %#x\n", s32Ret);  
        return s32Ret;  
    }  
  
s32Ret = HI_MPI_VI_SendPipeRaw(PipeNum, aPipeId, pastRawInfo, s32MilliSec);  
if (HI_SUCCESS != s32Ret) {  
    SAMPLE_PRT("HI_MPI_VI_SendPipeRaw failed with %#x\n", s32Ret);  
    return s32Ret;  
}  
  
s32Ret = HI_MPI_VI_GetChnFrame(ViPipe, ViChn, &stYUVFrameInfo, s32MilliSec);  
if (HI_SUCCESS != s32Ret) {  
    SAMPLE_PRT("HI_MPI_VI_GetChnFrame failed with %#x\n", s32Ret);  
    return s32Ret;  
}  
  
s32Ret = HI_MPI_VI_ReleaseChnFrame(ViPipe, ViChn, &stYUVFrameInfo);  
if (HI_SUCCESS != s32Ret) {  
    SAMPLE_PRT("HI_MPI_VI_ReleaseChnFrame failed with %#x\n", s32Ret);  
    return s32Ret;  
}
```

2. 使用HI_MPI_ISP_RunOnce处理视频流时需要用户创建一个线程:

```
...  
stViConfig.astViInfo[s32SnsId].stSnsInfo.enSnsType =  
SONY_IMX290_MIPI_2M_30FPS_12BIT_WDR3TO1,  
    stViConfig.astViInfo[s32SnsId].stDevInfo.enWDRMode = WDR_MODE_3To1_LINE;  
...
```



```
pthread_t thread;
s32Ret = pthread_create(&thread, NULL, Hi_Vi_SendWDRFrameProc,
(HI_VOID*)&stSendRawThreadInfo);

if (0 == s32Ret)
{
    pthread_detach(thread);
}

SAMPLE_COMM_VI_SwitchMode_StopVI(&stViConfig);
g_u32RunOnceSwitch = 1;
g_enWDRMode = WDR_MODE_NONE;

stViConfig.astViInfo[s32SnsId].stSnsInfo.enSnsType = SONY_IMX290_MIPI_2M_30FPS_12BIT;
stViConfig.astViInfo[s32SnsId].stDevInfo.enWDRMode = WDR_MODE_NONE;

stViConfig.astViInfo[0].stPipeInfo.aPipe[0]      = ViRawOutPipe;
stViConfig.astViInfo[0].stPipeInfo.aPipe[1]      = -1;
stViConfig.astViInfo[0].stPipeInfo.aPipe[2]      = -1;
stViConfig.astViInfo[0].stPipeInfo.aPipe[3]      = -1;
SAMPLE_RunonceSwitch_StartVi(&stViConfig);
SAMPLE_COMM_VI_SwitchISPMode(&stViConfig);

g_u32RunOnceSwitch = 0;

static void *Hi_Vi_SendWDRFrameProc(void *pArgs)
{
    .....
    while(1)
    {
        HI_S32 s32MilliSec = 100;
        i++;
        if(g_u32RunOnceSwitch == 1)
        {
            HI_MPI_ISP_RunOnce(ViRawOutPipe);
        }
        if ( g_enWDRMode == WDR_MODE_3To1_LINE ) {
            s32Ret = HI_MPI_VI_GetPipeFrame(ViRawOutPipe, &stRawInfo[0], s32MilliSec);
```



```
        if (HI_SUCCESS != s32Ret) {
            SAMPLE_PRT("HI_MPI_VI_GetPipeFrame failed with %#x\n", s32Ret);
            continue;
        }
        s32Ret = SAMPLE_Capture_VideoWDRFrameProc(ViRawOutPipe, &stRawInfo[0],
&stRawInfo[1], &stRawInfo[2]);
        if (HI_SUCCESS != s32Ret) {
            break;
        }
        s32Ret = HI_MPI_VI_ReleasePipeFrame(ViRawOutPipe, &stRawInfo[0]);
        if (HI_SUCCESS != s32Ret) {
            SAMPLE_PRT("HI_MPI_VI_ReleasePipeFrame failed with %#x\n", s32Ret);
            goto EXIT5;
        }
    }

    if ( g_enWDRMode == WDR_MODE_NONE )
    {

        s32Ret = HI_MPI_VI_GetPipeFrame(ViRawOutPipe, &stRawInfo[0], s32MilliSec);
        if (HI_SUCCESS != s32Ret) {
            SAMPLE_PRT("HI_MPI_VI_GetPipeFrame failed with %#x\n", s32Ret);
            continue;
        }

        s32Ret = SAMPLE_Capture_VideoFrameProc(ViRawOutPipe, &stRawInfo[0]);
        if (HI_SUCCESS != s32Ret) {
            break;
        }
        s32Ret = HI_MPI_VI_ReleasePipeFrame(ViRawOutPipe, &stRawInfo[0]);
        if (HI_SUCCESS != s32Ret) {
            SAMPLE_PRT("HI_MPI_VI_ReleasePipeFrame failed with %#x\n", s32Ret);
            goto EXIT5;
        }
    }
}

EXIT5:
    stDumpAttr.bEnable = HI_FALSE;
    stDumpAttr.u32Depth = 0;
```




```
HI_MPI_VI_SetPipeDumpAttr(ViRawOutPipe, &stDumpAttr);  
    return NULL;  
}
```

【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(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	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】

- 调用 [HI_MPI_ISP_Init](#) 和 [HI_MPI_ISP_Run](#) 之后，再调用本接口退出 ISP firmware。



- 不支持多进程，必须要与 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Run](#) 接口在同一个进程调用。
- 支持重复调用本接口。
- 在拼接模式时，必须先退出主 pipe，后退出其他 pipe。
- 不支持相同 ViPipe 时，多线程执行 ISP 创建和销毁（多线程同时调用 sensor_register_callback、[HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#)、[HI_MPI_ISP_MemInit](#)、[HI_MPI_ISP_Init](#)、[HI_MPI_ISP_Exit](#)）
- 推荐调用 [HI_MPI_ISP_Init](#) 之后，在调用本接口。

【举例】

无

【相关主题】

[HI_MPI_ISP_Init](#)

HI_MPI_ISP_SetPubAttr

【描述】

设置 ISP 公共属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetPubAttr(VI_PIPE ViPipe, const ISP\_PUB\_ATTR\_S *pstPubAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstPubAttr	ISP 公共属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。



【需求】

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

【注意】

- 图像属性即对应的 sensor 的采集属性。
- ISP 启动时，需要确保已调用 [HI_MPI_ISP_MemInit](#) 初始化 ISP 外部寄存器。
- ISP 支持运行过程中动态裁剪图像的起始位置。
- 调用本接口后 ISP 内的处理流程：
 - ISP firmware 判断图像 WDR 模式、分辨率、帧率是否变化，若都不变则直接返回；否则，ISP firmware 会调用 sensor cmos.c 里面的 `cmos_set_wdr_mode`、`cmos_set_image_mode` 函数改变 sensor 模式；
 - 若 sensor 模式不改变（返回值-2），判断 ISP 的裁剪宽高是否变化，若有变化，ISP firmware 切换分辨率，并调用 `sensor_init` 函数重新配置 sensor；
 - 若 sensor 模式改变（返回值为 0），则 ISP firmware 会调用 `sensor_init` 函数重新配置 sensor；
 - ISP firmware 将帧率信息传给上海海思 AE 库，并决定是否更改帧率。
- 若调用本接口实现动态分辨率和帧率切换时 sensor 模式发生了改变，请参照 sample 提供的切换流程操作（先停掉 Vi 设备，再创建 Vi 设备，然后设置 `ISPPubAttr` 进行切换）。当前系统不支持在 Vi 并行模式下切换帧率。另外，动态分辨率和帧率切换时，切换的分辨率和帧率必须有一项要不同（即不能切换到自己本身），否则，sensor 可能不会重新初始化而导致异常，模式切换时也不能切换到自己本身。对于 ISP 输入同样分辨率和帧率但需要采用不同初始化序列的情况，可以利用不同的 `u8SnsMode` 来实现模式切换。
- 使用 ISP 提供的裁剪功能时，需要注意：

动态裁剪图像的宽高时会重新初始化 sensor，切换流程参照 sample 提供的切换流程（先停掉 Vi 设备，再创建 Vi 设备，然后设置 `ISPPubAttr` 进行切换）。在线 WDR 模式下不支持 ISP 的裁剪功能。
- 用户可以更改 sensor cmos.c 里面的 `cmos_set_image_mode` 函数调整 sensor 模式切换的顺序。如只提供了 5M30fps 和 1080P60fps 初始化序列的 sensor，若要运行 1080P30fps，可以从 5M30fps 裁剪得到，也可以从 1080P60fps 降帧得到，修改 `cmos_set_image_mode` 函数实现即可。



- 通过 [HI_MPI_ISP_SetPubAttr](#) 接口配置超过 sensor 帧率范围的帧率时，该帧率值能配置到 ISP 中，但是 sensor_cmos.c 检测该帧率值超出范围而不做改变帧率的动作。此时应用层如果做模式切换（如：线性模式切 WDR 模式），sensor 重新初始化，并从 ISP 中读取帧率，由于 ISP 中存的是前一个模式配置的超出范围的帧率，sensor 重新配置帧率失败，会造成切换后的模式出现帧率异常，画面异常的现象。所以使用该接口配置帧率时不要配置超过 sensor 帧率范围的帧率值。
- 该接口不支持的情况：在不同的工作模式下从 WDR 切换到线性或不同的工作模式下的分辨率或帧率的切换(例如不支持从 VI_ONLINE_VPSS_OFFLINE 的 WDR 模式切换到 VI_PARALLEL_VPSS_OFFLINE 线性模式)。
- 切换线性模式和帧 WDR 模式时，同样会判断 cmos_set_image_mode 的返回值，因此线性模式和帧 WDR 模式应该采用不同的 u8ImgMode，才能保证切换成功。
- 若支持 Hi3516EV200 MCF(Mono Color Fusion)模式，因 ISP0 和 ISP1 控制的都是同一个 Sensor，所以当使用本接口配置帧率控制时，只能使用 ISP0 进行配置。如果同时使用 ISP0 和 ISP1 调用本接口对 sensor 进行帧率控制，MCF 的帧率输出会异常。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetPubAttr](#)

HI_MPI_ISP_GetPubAttr

【描述】

获取 ISP 公共属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetPubAttr(VI_PIPE ViPipe, ISP\_PUB\_ATTR\_S *pstPubAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstPubAttr	ISP 公共属性。	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetPubAttr](#)

HI_MPI_ISP_SetFMWState

【描述】

设置 ISP firmware 状态。

【语法】

```
HI_S32 HI_MPI_ISP_SetFMWState(VI_PIPE ViPipe, const ISP\_FMW\_STATE\_E enState);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
enState	ISP firmware 状态。	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

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

【举例】

无

【相关主题】

[HI_MPI_ISP_GetFMWState](#)

HI_MPI_ISP_GetFMWState

【描述】

获取 ISP firmware 状态。

【语法】

```
HI_S32 HI_MPI_ISP_GetFMWState(VI_PIPE ViPipe, ISP\_FMW\_STATE\_E *penState);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
penState	ISP firmware 状态。	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetFMWState](#)

HI_MPI_ISP_SetSnsSlaveAttr

【描述】

设置从模式 sensor 行场同步信号。

【语法】

```
HI_S32 HI_MPI_ISP_SetSnsSlaveAttr (SLAVE_DEV SlaveDev, const ISP\_SLAVE\_SNS\_SYNC\_S
*pstSnsSync);
```

【参数】

参数名称	描述	输入/输出
SlaveDev	Slave 设备号。	输入
pstSnsSync	同步信号配置。	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】

- 从模式 sensor 需要 Hi3559AV100/Hi3519AV100/Hi3516CV500 /Hi3559V200Hi3516EV300/Hi3516EV200 其提供行同步 XHS 和场同步信号 XVS，在这两个信号的控制下进行曝光与数据读出，这个接口主要配置同步信号发生模块，使其输出 sensor 要求的行场时序。此接口一般在 sensor 库里调用。
- 支持从模式 sensor 设置不同的时序配置，当前在 xxx_cmos.c/xxx_sensor_ctl.c 文件中设定当前 sensor 的时序配置。
- 从信号有两组绑定关系，首先 pipe 和 vsync 之间有绑定关系，其次 vsync 内部还可选择不同的信号源 slave，即不同 pipe 可以选择不同的从信号设备 vsync，不同的从信号设备 vsync 又可以选择不同的信号源 slave。
- 通常拼接模式，如果是两路拼接，最好选择同一个从信号设备，如果大于 2 路拼接，那么不同 sensor 必然会连接到不同的从信号设备上，这时不同的从信号设备要选择相同的信号源，这样才能保证不同 sensor 之间的同步。
- 通常如果多路从 sensor，不使用拼接模式而独立操作，那么这时不同的 sensor 要连接到不同的从信号设备上，并且从信号设备还要选择不同的信号源。

【举例】

4 路 imx477 slave sensor 拼接模式时，使用 pipe 的 id 分别是 0/2/4/6，他们分别绑定在从信号 vsync0、vsync0、vsync1、vsync1，因为是拼接模式，所以他们要选择相同的从信号源 slave，假设是 slave0，那么驱动中设置可以用如下方式进行赋值：

```
HI_S32 g_Imx477SlaveBindDev[ISP_MAX_PIPE_NUM] = {0, x, 0, x, 1, x, 1, x};  
HI_U32 g_Imx477SlaveSensorModeTime[ISP_MAX_PIPE_NUM] = {0, x, 0, x, 0, x, 0, x};
```




4 路 imx477 slave sensor 非拼接模式时，使用 pipe 的 id 分别是 0/2/4/6，他们需要分别绑定不同的从信号 vsync0、vsync1、vsync2、vsync3，因为是非拼接模式，所以他们要选择不同的从信号源 slave0、slave1、slave2、slave3，那么驱动中设置可以用如下方式进行赋值：

```
HI_S32 g_Imx477SlaveBindDev[ISP_MAX_PIPE_NUM] = {0, x, 1, x, 2, x, 3, x};  
HI_U32 g_Imx477SlaveSensorModeTime[ISP_MAX_PIPE_NUM] = {0, x, 1, x, 2, x, 3, x};
```

这里 x 代表满足接口的任意值，无需关注。

【相关主题】

[HI_MPI_ISP_GetSnsSlaveAttr](#)

HI_MPI_ISP_GetSnsSlaveAttr

【描述】

获取从模式 sensor 行场同步信号。

【语法】

```
HI_S32 HI_MPI_ISP_GetSnsSlaveAttr(SLAVE_DEV SlaveDev, ISP_SLAVE_SNS_SYNC_S *pstSnsSync);
```

【参数】

参数名称	描述	输入/输出
SlaveDev	Slave 设备号。	输入
pstSnsSync	同步信号配置。	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_SetSnsSlaveAttr](#)

HI_MPI_ISP_SetModuleControl

【描述】

设定 ISP 功能模块的控制。

【语法】

```
HI_S32 HI_MPI_ISP_SetModuleControl(VI_PIPE ViPipe, const ISP\_MODULE\_CTRL\_U *punModCtrl);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
punModCtrl	模块控制值。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 错误码 。

【需求】

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



【注意】

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

【举例】

无

【相关主题】

[HI_MPI_ISP_GetModuleControl](#)

HI_MPI_ISP_GetModuleControl

【描述】

获取 ISP 功能模块的控制。

【语法】

```
HI_S32 HI_MPI_ISP_GetModuleControl(VI_PIPE ViPipe, ISP_MODULE_CTRL_U *punModCtrl);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
punModCtrl	模块控制值。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetModuleControl](#)

HI_MPI_ISP_SetRegister

【描述】

设置寄存器值。

【语法】

```
HI_S32 HI_MPI_ISP_SetRegister(VI_PIPE ViPipe, HI_U32 u32Addr, HI_U32 u32Value);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
u32Addr	寄存器地址。	输入
u32Value	寄存器值。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 错误码 。

【需求】

- 头文件: hi_comm_isp.h、mpi_isp.h



- 库文件: libisp.a

【注意】

- 上海海思 AE 外部寄存器为(0x700000 +s32Id*0x2000)。
- 上海海思 AWB 外部寄存器为(0x600000 +s32Id*0x1000)。
- 上海海思 ISP 外部寄存器为(0x100000+ViPipeId*ISP_VREG_SIZE。
(Hi3559AV100/ Hi3519AV100/ Hi3516CV500/Hi3559V200 ISP_VREG_SIZE 是 0x20000, Hi3516EV200/Hi3516EV300 ISP_VREG_SIZE 是 0x10000)
- 本接口不支持设置逻辑寄存器。
- 调用本接口时, 需要确保已调用 [HI_MPI_ISP_MemInit](#) 初始化 ISP 外部寄存器。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetRegister](#)

HI_MPI_ISP_GetRegister

【描述】

获取寄存器值。

【语法】

```
HI_S32 HI_MPI_ISP_GetRegister(VI_PIPE ViPipe, HI_U32 u32Addr, HI_U32 *pu32Value);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
u32Addr	寄存器地址。	输入
pu32Value	寄存器值。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

调用本接口时，需要确保已调用 [HI_MPI_ISP_MemInit](#) 初始化 ISP 外部寄存器。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetRegister](#)

HI_MPI_ISP_GetVDDTimeOut

【描述】

获取 ISP 中断信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetVDDTimeOut(VI_PIPE ViPipe, ISP\_VD\_TYPE\_E enIsPVDType, HI_U32  
u32MilliSec);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
enIsPVDType	场同步信号。	输入
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 。

【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，中断是采用的帧开始中断。
- 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。
- 使用 [ISP_VD_FE_END](#) 方式获取 ISP 中断信息并读取统计信息。在极端情况下（cpu 占用率高等），会出现读取统计信息不及时情况。建议使用 [ISP_VD_FE_START](#) 方式获取 ISP 中断来读取统计信息。
- N 个进程同时调用该接口获取同样的 enIspVDType 时，每个进程获取到的中断信息为实际中断信息的(1/N)，比如在灌 raw 场景中会使用 [ISP_VD_FE_START](#) 的方式来发送 raw 数据，如果此时其他进程再使用 [ISP_VD_FE_START](#) 运行其他业务，会导致灌 raw 业务的帧率减半。

【举例】

无

【相关主题】

无

HI_MPI_ISP_SensorRegCallBack

【描述】



ISP 提供的 sensor 注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_SensorRegCallBack(VI_PIPE ViPipe, ISP_SNS_ATTR_INFO_S *pstSnsAttrInfo ,  
ISP_SENSOR_REGISTER_S *pstRegister);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstSnsAttrInfo	向 ISP 注册的 Sensor 的属性。	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。

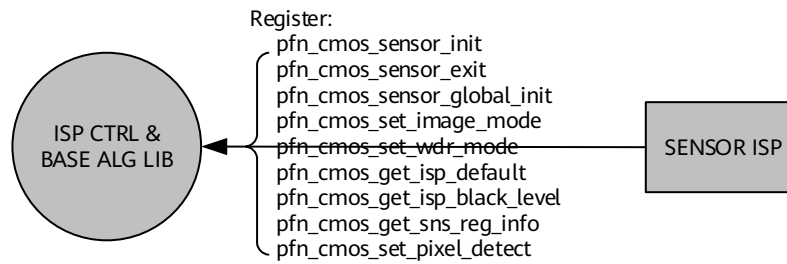
【需求】

- 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。
- 此接口不支持多进程操作。

图2-1 ISP 库与 sensor 库间的接口



【举例】

```

VI_PIPE ViPipe = 0;
HI_S32 s32Ret;
ISP_SENSOR_REGISTER_S stIspRegister;
ISP_SNS_ATTR_INFO_S stSnsAttrInfo;
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIspRegister.stSnsExp;
memset(pstSensorExpFunc, 0, sizeof(ISP_SENSOR_EXP_FUNC_S));
pstSensorExpFunc->pfn_cmos_sensor_init = sensor_init;
pstSensorExpFunc->pfn_cmos_sensor_exit = sensor_exit;
pstSensorExpFunc->pfn_cmos_sensor_global_init = sensor_global_init;
pstSensorExpFunc->pfn_cmos_get_isp_default = cmos_get_isp_default;
pstSensorExpFunc->pfn_cmos_get_isp_black_level = cmos_get_isp_black_level;
pstSensorExpFunc->pfn_cmos_set_pixel_detect = cmos_set_pixel_detect;
pstSensorExpFunc->pfn_cmos_set_wdr_mode = cmos_set_wdr_mode;
pstSensorExpFunc->pfn_cmos_get_sns_reg_info = cmos_get_sns_regs_info;
pstSensorExpFunc->pfn_cmos_set_image_mode = cmos_set_image_mode;
stSnsAttrInfo.eSensorId = IMX178_ID;
s32Ret = HI_MPI_ISP_SensorRegCallBack(ViPipe, &stSnsAttrInfo, &stIspRegister);
if (s32Ret)
{
    printf("sensor register callback function failed!\n");
    return s32Ret;
}
    
```

【相关主题】

[HI_MPI_ISP_SensorUnRegCallBack](#)



HI_MPI_ISP_SensorUnRegCallBack

【描述】

ISP 提供的 sensor 反注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_SensorUnRegCallBack(VI_PIPE ViPipe, SENSOR_ID SensorId);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
SensorId	向 ISP 注册的 Sensor 的 Id。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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 反注册的 sensor 和向 3A 反注册的 sensor 是否为同一个 sensor。
- 此接口不支持多进程操作。

【举例】

```
VI_PIPE ViPipe = 0;  
s32Ret = HI_MPI_ISP_SensorUnRegCallBack(ViPipe, IMX178_ID);  
if (s32Ret)  
{
```



```
printf("sensor unregister callback function failed!\n");  
return s32Ret;  
}
```

【相关主题】

[HI_MPI_ISP_SensorRegCallBack](#)

HI_MPI_ISP_AELibRegCallBack

【描述】

ISP 提供的 AE 库注册的回调接口。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAeLib	AE 库结构体指针。	输入
pstRegister	AE 库注册结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

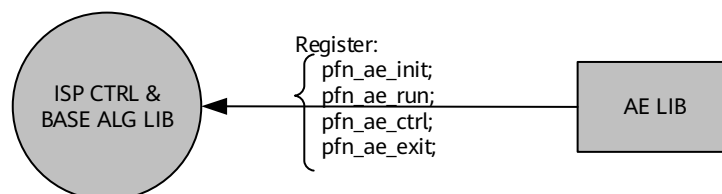
【需求】

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

【注意】

- ISP 提供统一的 AE 算法库接口，初始化、运行、控制、销毁 AE 算法库。使用上海海思 AE 算法库时，不需要关注此接口；使用用户自己的 AE 算法库时，需要调用此接口向 ISP 注册回调函数。
- 此接口不支持多进程操作。
- 最大支持 2 个 AE 库注册。

图2-2 ISP 库与 AE 库间的接口



【举例】

```

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

【相关主题】

[HI_MPI_ISP_AELibUnRegCallBack](#)

HI_MPI_ISP_AELibUnRegCallBack

【描述】

ISP 提供的 AE 库反注册的回调接口。

【语法】

```

HI_S32 HI_MPI_ISP_AELibUnRegCallBack(VI_PIPE ViPipe, ALG_LIB_S *pstAeLib);
  
```



【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAeLib	AE 库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 使用上海海思 AE 算法库时，不需要关注此接口；使用用户自己的 AE 算法库时，需要调用此接口向 ISP 反注册回调函数。
- 此接口不支持多进程操作。

【举例】

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

【相关主题】

[HI_MPI_ISP_AELibRegCallBack](#)



HI_MPI_ISP_AWBLibRegCallBack

【描述】

ISP 提供的 AWB 库注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_AWBLibRegCallBack(VI_PIPE ViPipe, ALG_LIB_S *pstAwbLib,  
ISP_AWB_REGISTER_S *pstRegister);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAwbLib	AWB 库结构体指针。	输入
pstRegister	AWB 库注册结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

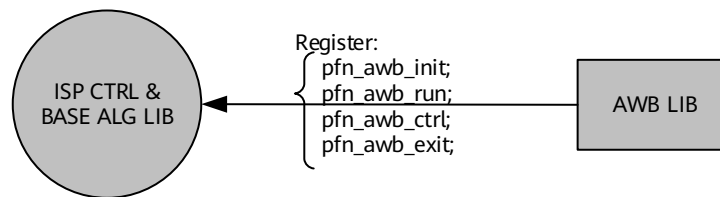
- 头文件：hi_comm_isp.h、mpi_isp.h
- 库文件：libisp.a

【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。使用上海海思 AWB 算法库时，不需要关注此接口；使用用户自己的 AWB 算法库时，需要调用此接口向 ISP 注册回调函数。
- 此接口不支持多进程操作。
- 最大支持 2 个 AWB 库注册。



图2-3 ISP 库与 AWB 库间的接口



【举例】

无

【相关主题】

[HI_MPI_ISP_AWBLibUnRegCallBack](#)

HI_MPI_ISP_AWBLibUnRegCallBack

【描述】

ISP 提供的 AWB 库反注册的回调接口。

【语法】

```
HI_S32 HI_MPI_ISP_AWBLibUnRegCallBack(VI_PIPE ViPipe, ALG\_LIB\_S *pstAwbLib);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAwbLib	AWB 库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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】

- 使用上海海思 AWB 算法库时，不需要关注此接口；使用用户自己的 AWB 算法库时，需要调用此接口向 ISP 反注册回调函数。
- 此接口不支持多进程操作。

【相关主题】

[HI_MPI_ISP_AWBLibRegCallBack](#)

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(VI_PIPE ViPipe, const ISP\_BIND\_ATTR\_S *pstBindAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstBindAttr	绑定结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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 库，并希望切换算法库时才需要调用。当注册多个 AE/AWB 库时，默认绑定的为最后一个注册的 AE 库和 AWB 库。
- 此接口不支持多进程操作。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetBindAttr](#)

HI_MPI_ISP_GetBindAttr

【描述】

获取 ISP 库与 3A 库、sensor 的绑定关系。

【语法】

```
HI_S32 HI_MPI_ISP_GetBindAttr(VI_PIPE ViPipe, ISP\_BIND\_ATTR\_S *pstBindAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstBindAttr	绑定结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】



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

【注意】

此接口不支持多进程操作。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetBindAttr](#)

HI_MPI_ISP_SetDCFInfo

【描述】

设置 DCF 参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetDCFInfo(VI_PIPE ViPipe, const ISP\_DCF\_INFO\_S *pstIspDCF);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspDCF	DCF 参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h



- 库文件: libisp.a

【注意】

调用该接口前须调用 HI_MPI_VB_SetSupplementConf(请参考《HiMPP 媒体处理软件 V4.0 开发参考》的系统控制部分 2.3 小节), 将 stSupplementConf 配置为 VB_SUPPLEMENT_JPEG_MASK。

【举例】

```
VB_SUPPLEMENT_CONF_S stSupplementConf;

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

.....

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

.....

VI_PIPE ViPipe;
s32Ret=HI_MPI_ISP_Init(ViPipe);

.....

ISP_DCF_INFO_S    stIspDCF;
//will:119 105 108 108
stIspDCF.stIspDCFConstInfo.au8ImageDescription[0]=119;
stIspDCF.stIspDCFConstInfo.au8ImageDescription[1]=105;
stIspDCF.stIspDCFConstInfo.au8ImageDescription[2]=108;
stIspDCF.stIspDCFConstInfo.au8ImageDescription[3]=108;
```



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

stlspDCF. stlspDCFUpdateInfo.u32ISOSpeedRatings = 500;
stlspDCF. stlspDCFUpdateInfo.u32ExposureBiasValue = 5;
stlspDCF. stlspDCFUpdateInfo.u32ExposureTime      = 0x00010004;
stlspDCF. stlspDCFUpdateInfo.u32FNumber           = 0x0001000f;
stlspDCF. stlspDCFConstInfo.u32FocalLength        = 0x00640001;
stlspDCF. stlspDCFUpdateInfo.u32MaxApertureValue = 0x00010001;
stlspDCF. stlspDCFConstInfo.u8Contrast             =5;
stlspDCF. stlspDCFConstInfo.u8CustomRendered      = 0;
stlspDCF. stlspDCFUpdateInfo.u8ExposureMode       = 0;
stlspDCF. stlspDCFConstInfo.u8ExposureProgram    = 1;
stlspDCF. stlspDCFConstInfo.u8FocalLengthIn35mmFilm = 0;
stlspDCF. stlspDCFConstInfo.u8GainControl = 1;
stlspDCF. stlspDCFConstInfo.u8LightSource = 1;
stlspDCF. stlspDCFConstInfo.u8MeteringMode = 1;
```



```
stlspDCF.stlspDCFConstInfo.u8Saturation    = 1;
stlspDCF.stlspDCFConstInfo.u8SceneCaptureType = 1;
stlspDCF.stlspDCFConstInfo.u8SceneType = 0;
stlspDCF.stlspDCFConstInfo.u8Sharpness =5;
stlspDCF.stlspDCFUpdateInfo.u8WhiteBalance = 0;
HI_MPI_ISP_SetDCFInfo(ViPipe,&stlspDCF);
```

【相关主题】

[HI_MPI_ISP_GetDCFInfo](#)

HI_MPI_ISP_GetDCFInfo

【描述】

获取 DCF 参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetDCFInfo(VI_PIPE ViPipe, ISP\_DCF\_INFO\_S *pstlspDCF)
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstlspDCF	DCF 参数结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetDCInfo](#)

HI_MPI_ISP_SetPipeDifferAttr

【描述】

设置多路 ISP Pipe 差异属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetPipeDifferAttr(VI_PIPE ViPipe, const ISP\_PIPE\_DIFF\_ATTR\_S *pstPipeDiffer);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstPipeDiffer	多路 ISP 差异属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 该接口不是必调接口，用于多路 ISP 拼接模式，对拼接效果要求更高时，可以通过标定工具（HiPQ_Stitching_Tool）标定出多路 ISP 的亮度、颜色等差异，通过



此接口配置减少多路 ISP 输出图像的差异。该接口必须在 [HI_MPI_ISP_Init](#) 之后调用。

- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持该接口。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetPipeDifferAttr](#)

HI_MPI_ISP_GetPipeDifferAttr

【描述】

获取多路 ISP Pipe 差异属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetPipeDifferAttr(VI_PIPE ViPipe, ISP\_PIPE\_DIFF\_ATTR\_S * pstPipeDiffer);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstPipeDiffer	多路 ISP 差异属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】



- 该接口不是必调接口，与 [HI_MPI_ISP_SetPipeDifferAttr](#) 配套使用，可以获取对应当前配置的多路 ISP 差异参数。
- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300/Hi3516DV200 不支持该接口。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetPipeDifferAttr](#)

HI_MPI_ISP_SetCtrlParam

【描述】

设置 ISP 控制参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetCtrlParam(VI_PIPE ViPipe, const ISP\_CTRL\_PARAM\_S *pstIspCtrlParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstIspCtrlParam	ISP 控制参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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



【注意】

接口使用限制详见 [ISP_CTRL_PARAM_S](#) 中的注意事项。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetCtrlParam](#)

HI_MPI_ISP_GetCtrlParam

【描述】

获取 ISP 控制参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetCtrlParam(VI_PIPE ViPipe, ISP\_CTRL\_PARAM\_S *pstIspCtrlParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstIspCtrlParam	ISP 控制参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetCtrlParam](#)

HI_MPI_ISP_SetModParam

【描述】

设置 ISP 模块参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetModParam(const ISP\_MOD\_PARAM\_S *pstModParam);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

接口使用限制详见 [ISP_MOD_PARAM_S](#) 中的注意事项。

【举例】



无

【相关主题】

[HI_MPI_ISP_GetModParam](#)

HI_MPI_ISP_GetModParam

【描述】

获取 ISP 模块参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetModParam(ISP\_MOD\_PARAM\_S *pstModParam);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetModParam

HI_MPI_ISP_SetSmartInfo

【描述】

向 ISP 模块传递智能信息。

【语法】

```
HI_S32 HI_MPI_ISP_SetSmartInfo(VI_PIPE ViPipe, const ISP_SMART_INFO_S *pstSmartInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstSmartInfo	智能信息，包括人脸、人形信息、隧道检测信息。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

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

【注意】

- 客户在使用该接口传递智能检测信息时建议每秒调用不少于 10 次，并且无论是否检测到目标物体，均需要连续不间断调用接口。
- 接口使用详见 [ISP_SMART_INFO_S](#) 说明。

【举例】

```
VI_PIPE ViPipe = 0;  
ISP_SMART_INFO_S stSmartInfo;
```



```
HI_MPI_ISP_GetSmartInfo(j, & stSmartInfo);

//此例子仅人脸检测
stSmartInfo.stPeopleROI[ISP_FACE_INDEX].bEnable= HI_TRUE;
stSmartInfo.stPeopleROI[ISP_FACE_INDEX].bAvailable = HI_TRUE;
stSmartInfo.stPeopleROI[ISP_FACE_INDEX].u8Luma = u8DetectFaceLuma;
stSmartInfo.stPeopleROI[ISP_PEOPLE_INDEX].bEnable= HI_FALSE;
stSmartInfo.stPeopleROI[ISP_PEOPLE_INDEX].bAvailable = HI_FALSE;
stSmartInfo.stPeopleROI[ISP_PEOPLE_INDEX].u8Luma = 0;

HI_MPI_ISP_SetSmartInfo(j, & stSmartInfo);
```

【相关主题】

[HI_MPI_ISP_GetSmartInfo](#)

HI_MPI_ISP_GetSmartInfo

【描述】

获取 ISP 模块参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetSmartInfo(VI_PIPE ViPipe, ISP\_SMART\_INFO\_S *pstSmartInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstSmartInfo	智能信息，包括人脸、人形信息、隧道检测信息。	输出

【返回值】

返回值	描述
0	成功。



非 0	失败，其值为 错误码 。

【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_SetSmartInfo](#)

HI_MPI_ISP_GetLightboxGain

【描述】

获取 AWB 在线标定得到的增益结构体。

【语法】

```
HI_S32 HI_MPI_ISP_GetLightboxGain(VI_PIPE ViPipe, ISP\_AWB\_Calibration\_Gain\_S  
*pstAWBCalibrationGain);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAWBCalibrationGain	AWB 在线标定输出的增益结构体。	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】

AWB 参数在线标定的接口主要为了支撑消费类客户对运动 DV 量产时产线上的 AWB 参数的校正。该接口的调用环境限于均匀光源下的均匀背景，且色温范围限于 4500K 到 6500K。

【举例】

无

【相关主题】

无

HI_MPI_ISP_SetRawPos

【描述】

设置采集 Raw 的位置。

【语法】

```
HI_S32 HI_MPI_ISP_SetRawPos(VI_PIPE ViPipe, const ISP\_RAW\_POS *rawPos);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
rawPos	Raw 位置属性。	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】

- 仅 Hi3559AV100 支持。
- 调用该接口获取 raw 数据时，raw 数据经过了 demosaic 之前的模块的处理，从 demosaic 模块之后，后续的 ISP 的模块都被 bypass；需要从 VI 模块的通道数据写出来得到 raw，即主要通过接口 HI_MPI_VI_GetChnAttr/HI_MPI_VI_GetChnFrame/HI_MPI_VI_ReleaseChnFrame 等配置来实现。
HI_MPI_VI_GetChnAttr/HI_MPI_VI_GetChnFrame/HI_MPI_VI_ReleaseChnFrame 具体请参考《HiMPP 媒体处理软件 V4.0 开发参考》“03 视频输入” 章节

【举例】

无

【相关主题】

[HI_MPI_ISP_GetRawPos](#)

HI_MPI_ISP_GetRawPos

【描述】

获取采集 Raw 的位置。

【语法】

```
HI_S32 HI_MPI_ISP_GetRawPos(VI_PIPE ViPipe, ISP\_RAW\_POS *rawPos);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
rawPos	Raw 位置属性。	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】

仅 Hi3559AV100 支持。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetRawPos](#)

HI_MPI_ISP_IrAutoRunOnce

【描述】

运行红外自动切换功能。

【语法】

```
HI_S32 HI_MPI_ISP_IrAutoRunOnce(ISP_DEV IspDev, ISP\_IR\_AUTO\_ATTR\_S *pstIrAttr);
```

【参数】



参数名称	描述	输入/输出
IspDev	ISP 设备号。	输入
pstIrrAttr	红外自动切换属性。	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, lib_hiir_auto.a

【注意】

使用此接口时需要包含 lib_hiir_auto.a。

【举例】

见 mpp/sample/ir_auto/sample_ir_auto.c

【相关主题】

无



2.3 数据类型

本文中变量，如未明确指定取值范围，则默认是数据类型对应的取值范围。例如 HI_U8 数据类型的变量取值范围为[0, 255]。本文中变量，如未明确指定数据精度，则默认是 1。

- [RECT_S](#): 定义裁剪窗口起始位置和图像宽高。
- [ISP_BAYER_FORMAT_E](#): 定义输入 Bayer 图像数据格式。
- [ISP_BAYER_RAWBIT_E](#): 定义输入 Bayer 图像数据位宽。
- [SIZE_S](#): 定义 sensor 输出的宽高
- [COLOR_GAMUT_E](#): 定义通道色域属性。
- [ISP_PUB_ATTR_S](#): 定义 ISP 公共属性。
- [ISP_FMW_STATE_E](#): 定义 ISPfirmware 状态。
- [ISP_SLAVE_SNS_SYNC_S](#): 定义从模式 sensor 同步信号配置。
- [ISP_WDR_MODE_S](#): 定义 ISP 宽动态模式。
- [WDR_MODE_E](#): 定义宽动态模式。
- [ISP_MODULE_CTRL_U](#): 定义 ISP 功能模块的控制。
- [ISP_VD_TYPE_E](#): 定义场同步信号。
- [ISP_SNS_ATTR_INFO_S](#): 定义 sensor 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_SENSOR_IMAGE_MODE_S](#): 定义 sensor 输出的宽高和帧率属性。
- [ISP_CMOS_DRC_S](#): 定义 DRC 模块参数。
- [ISP_CMOS_DEMOSAIC_S](#): 定义 demosaic 参数。
- [ISP_CMOS_PREGAMMA_S](#): 定义 PreGamma 参数。
- [ISP_CMOS_GAMMA_S](#): 定义 Gamma 结构体。
- [ISP_CMOS_SHARPEN_MANUAL_S](#): 定义 sharpen manual 档 cmos 参数。
- [ISP_CMOS_SHARPEN_AUTO_S](#): 定义 sharpen auto 档 cmos 参数。
- [ISP_CMOS_SHARPEN_S](#): 定义 sharpen 参数。
- [ISP_CMOS_EDGEMARK_S](#): 定义 edgemark 的参数。
- [ISP_CMOS_HLC_S](#): 定义 HLC 的参数。
- [ISP_CMOS_LDCI_S](#): 定义 LDCI 参数。



- [ISP_CMOS_DPC_S](#): 定义 DPC 参数。
- [ISP_LSC_CABLI_TABLE_S](#): 定义 LSC 的校正参数。
- [ISP_BNR_LSC_CALIB_TABLE_S](#): 定义 Bayer NR LSC 的标定 LUT。
- [ISP_CMOS_LSC_S](#): 定义 LSC 参数。
- [ISP_RLSC_CABLI_TABLE_S](#): 定义 RLSC 的校正参数。
- [ISP_ACS_Y_SHADING_LUT_S](#): 定义 Auto Color Shading 亮度分量上的校正强度表。
- [ISP_ACS_COLOR_SHADING_LUT_S](#): 定义 Auto Color Shading 颜色分量上的 Lut 表。
- [ISP_ACS_CALIB_PARAM_S](#): 定义 Auto Color Shading 的标定参数，用标定工具生成。
- [ISP_CMOS_ACS_S](#): 定义 Auto Color Shading 的 CMOS 参数。
- [ISP_CMOS_RLSC_S](#): 定义 RLSC 的校正参数。
- [ISP_CMOS_GE_S](#): 定义 GE 参数。
- [ISP_CMOS_ANTIFALSECOLOR_S](#): 定义 AntifalseColor 参数。
- [ISP_CMOS_BAYERNR_S](#): 定义 BayerNR 结构体。
- [ISP_CMOS_DETAIL_S](#): 定义 Detail Enhance 结构体。
- [ISP_CMOS_SPLIT_POINT_S](#): 定义 split 参数。
- [ISP_CMOS_SPLIT_S](#): 定义 SPLIT 参数。
- [ISP_CMOS_CA_S](#): 定义 CA 参数。
- [ISP_CMOS_EXPANDER_POINT_S](#): 定义 Expander 拐点坐标参数。
- [ISP_CMOS_EXPANDER_S](#): 定义 Expander 结构体。
- [ISP_CMOS_NOISE_CALIBRATION_S](#): 定义 NOISE 校正参数。
- [ISP_CMOS_SENSOR_MAX_RESOLUTION_S](#): 定义 sensor 最大分辨率结构体。
- [ISP_CMOS_CLUT_S](#): 定义 CLUT 结构体。
- [ISP_CMOS_LOGLUT_S](#): 定义 LOGLUT 结构体。
- [ISP_CMOS_PRELOGLUT_S](#): 定义 PRELOGLUT 的参数。
- [ISP_CMOS_SENSOR_MODE_S](#): 定义 sensor 模式寄存器。
- [ISP_CMOS_WDR_S](#): 定义 WDR 结构体。
- [ISP_CMOS_DEHAZE_S](#): 定义 Dehaze 结构体。
- [ISP_CMOS_RGBIR_S](#): 定义 RGBIR 模块参数。



- [ISP_CMOS_DNG_COLORPARAM_S](#): 定义 DNG 白平衡校正系数。
- [ISP_CMOS_WDR_SWITCH_ATTR_S](#): 定义 WDR 切换属性。
- [ISP_CMOS_AWB_ATTR_S](#): 定义 AWB 模块参数。
- [ISP_CMOS_ALG_KEY_U](#): 定义 ISP 的各算法是否采用 cmos 中的默认配置的标志位。
- [ISP_CMOS_DEFAULT_S](#): 定义 ISP 基础算法库的初始化参数结构体。
- [ISP_CMOS_BLACK_LEVEL_S](#): 定义 sensor 的黑电平结构体。
- [ISP_CMOS_LCAC_MANUAL_S](#): 定义 sensor 的 Local CAC 算法的手动算法结构体。
- [ISP_CMOS_LCAC_AUTO_S](#): 定义 sensor 的 Local CAC 算法的自动算法结构体。
- [ISP_CMOS_LCAC_S](#): 定义 sensor 的 Local CAC 算法结构体。
- [ISP_SNS_MODE_E](#): 定义 sensor 的曝光配置方式。
- [ISP_MCF_DATA_S](#): 定义多光谱融合 sensor 控制寄存器。
- [ISP_SNS_TYPE_E](#): 定义 sensor 通信接口方式。
- [ISP_SNS_COMMBUS_U](#): 定义 sensor 绑定信息。
- [ISP_SNS_REGS_INFO_S](#): 定义 sensor 的寄存器信息。
- [ALG_LIB_S](#): 定义 AE/AWB 算法库结构体。
- [ISP_BIND_ATTR_S](#): 定义 ISP 库与 Sensor、3A 库之间绑定关系的结构体。
- [ISP_CTRL_PROC_WRITE_S](#): 定义 ISP 的 PROC 信息。
- [ISP_CTRL_CMD_E](#): 定义 ISP 对 3A 的控制命令。
- [ISP_STITCH_ATTR_S](#): 定义 ISP 拼接结构体。
- [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_SMART_INFO_S](#): 定义 ISP 提供给 AE 的智能检测信息结构体。
- [ISP_AE_INFO_S](#): 定义 ISP 提供给 AE 的统计信息结构体。
- [ISP_AE_STAT_ATTR_S](#): 定义 AE 库返回给 ISP 的配置寄存器结构体。
- [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_STAT_1_S](#): 定义 AWB 统计信息结构体。
- [ISP_AWB_STAT_RESULT_S](#): 定义 AWB 统计信息结构体。
- [ISP_AWB_INFO_S](#): 定义 ISP 提供给 AWB 的统计信息结构体。
- [ISP_AWB_RAW_STAT_ATTR_S](#): 定义 AWB 库返回给 ISP 的配置寄存器结构体。
- [ISP_AWB_RESULT_S](#): 定义 AWB 库返回给 ISP 的配置寄存器结构体。
- [ISP_AWB_Calibration_Gain_S](#): 定义 AWB 在线标定输出的增益结构体。
- [ISP_DCF_CONST_INFO_S](#): 定义 DCF 信息中用户配置参数。
- [ISP_DCF_UPDATE_INFO_S](#): 定义 DCF 信息中 ISP 实时更新参数
- [ISP_DCF_INFO_S](#): 定义 DCF 信息参数结构体。
- [ISP_PIPE_DIFF_ATTR_S](#): 定义两路 ISP 差异参数结构体。
- [ISP_OP_TYPE_E](#): 定义模块运行状态。
- [ISP_CTRL_PARAM_S](#): 定义 ISP 控制参数结构体。
- [ISP_MOD_PARAM_S](#): 定义 ISP 模块参数结构体。
- [ISP_INIT_ATTR_S](#): 定义 ISP 第一次启动时 AE/AWB 的初始化参数结构体。
- [ISP_SNS_MIRRORFLIP_TYPE_E](#): 定义 sensor mirror-flip 枚举。
- [ISP_SNS_BUS_EX_S](#): 定义 sensor 通讯协议的扩展结构体。
- [ISP_SNS_OBJ_S](#): 定义指向 sensor 的对象。
- [ISP_SNS_STATE_S](#): 定义 sensor 相关全局变量参数结构体。
- [ISP_AWB_ALG_E](#): 定义 AWB 算法类型。

以下 SpecAWB 相关的数据类型，Hi3516EV200/Hi3516EV300 默认不支持，Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200 默认支持：

- [ISP_SPECAWB_ATTR_S](#): 定义机器学习 AWB 算法标定结果结构体。
- [ISP_SPECAWB_CONTROL_ATTR_S](#): 定义机器学习 AWB 算法控制参数结构体。
- [ISP_SPECKCWB_S](#): 定义 WB gain 值结构体。
- [ISP_SPECKCWB_S16](#): 定义 WB gain 值结构体，16bit 位宽。
- [ISP_SPECAWB_FACTTBL_ELEMENT_S](#): 定义机器学习 AWB 算法光源分布结构体。
- [ISP_SPECAWB_KELVIN_DBB_MAP_S](#): 定义色温/DeltaBlackBody 分布结构体。
- [ISP_SPECAWB_BBL_TBL_S](#): 定义普朗克曲线/Gain 值对应结构体。
- [ISP_SPECAWB_CAA_CONTROL_S](#): 定义色温转换表控制结构体。



- [ISP_SPECAWB_CAA_TBL_S](#): 定义单组色温转换表详细控制信息结构体。
- [ISP_SPECAWB_CAA_CONVERSION_S](#): 定义原始色温, 目的色温信息。
- [ISP_IR_STATUS_E](#): 定义设备当前的红外状态。
- [ISP_IR_SWITCH_STATUS_E](#): 定义设备的红外切换状态。
- [ISP_RAW_POS_E](#): 定义获取 raw 位置。
- [ISP_RAW_POS](#): 定义获取 RAW 的相关配置信息。
- [ISP_IR_AUTO_ATTR_S](#): 定义红外自动切换属性。

RECT_S

【说明】

定义裁剪窗口起始位置和图像宽高。

【定义】

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

【成员】

成员名称	描述
s32X	水平方向起始位置 • Hi3559AV100/Hi3519AV100 取值范围: [0, 8072] • Hi3516CV500 取值范围: [0, 2184] • Hi3516DV300 不同 ViPipe 的取值范围不同, 其中: ViPipe0/ViPipe1: [0, 2568] ViPipe2/ViPipe3: [0, 1928] • Hi3516AV300 不同 ViPipe 的取值范围不同, 其中: ViPipe0/ViPipe1: [0, 3720] ViPipe2/ViPipe3: [0, 1928]



成员名称	描述
	- Hi3559V200 不同 ViPipe 的取值范围不同，其中： ViPipe0: [0, 4488] ViPipe1: [0, 3720] ViPipe2/ViPipe3: [0, 1928] - Hi3516EV200 ViPipe0/ViPipe1 取值范围: [0, 2184] - Hi3516EV300 ViPipe0/ViPipe1 取值范围: [0, 2952] - Hi3562V100 取值范围: [0, 1928] - Hi3568V100 取值范围: [0, 3976]
s32Y	垂直方向起始位置 - Hi3559AV100/Hi3519AV100 取值范围: [0,8072] - Hi3516CV500 取值范围: [0, 2184] - Hi3516DV300 不同 ViPipe 的取值范围不同，其中： ViPipe0/ViPipe1: [0, 2568] ViPipe2/ViPipe3: [0, 1928] - Hi3516AV300 不同 ViPipe 的取值范围不同，其中： ViPipe0/ViPipe1: [0, 3720] ViPipe2/ViPipe3: [0, 1928] - Hi3559V200 不同 ViPipe 的取值范围不同，其中： ViPipe0: [0, 4488] ViPipe1: [0, 3720] ViPipe2/ViPipe3: [0, 1928] - Hi3516EV200 ViPipe0/ViPipe1 取值范围: [0, 2184] - Hi3516EV300 ViPipe0/ViPipe1 取值范围: [0, 2952] - Hi3562V100 取值范围: [0, 1416]
u32Width	图像宽度，4 对齐。使用 shading 功能时 4 对齐，否则 shading 功能会不正常。 - Hi3559AV100/Hi3519AV100 取值范围: [120, 8192] - Hi3516CV500 取值范围: [120, 2304]



成员名称	描述
	• Hi3516DV300 不同 ViPipe 取值范围不同, 其中: ViPipe0/ViPipe1: [120, 2688] ViPipe2/ViPipe3: [120, 2048] • Hi3516AV300 不同 ViPipe 的取值范围不同, 其中: ViPipe0/ViPipe1: [120, 3840] ViPipe2/ViPipe3: [120, 2048] • Hi3559V200 不同 ViPipe 的取值范围不同, 其中: ViPipe0: [120, 4608] ViPipe1: [120, 3840] ViPipe2/ViPipe3: [120, 2048] • Hi3516EV200 ViPipe0/ViPipe1 取值范围: [120, 2304] • Hi3516EV300 ViPipe0/ViPipe1 取值范围: [120, 3072] • Hi3562V100 取值范围: [120, 2048] • Hi3568V100 取值范围: [120,4096]
u32Height	图像高度, 4 对齐。使用 shading 功能时 4 对齐, 否则 shading 功能会不正常。 • Hi3559AV100/Hi3519AV100 取值范围: [120, 8192] • Hi3516CV500 取值范围: [120, 2304] • Hi3516DV300 不同 ViPipe 取值范围不同, 其中: ViPipe0/ViPipe1: [120, 2688] ViPipe2/ViPipe3: [120, 2048] • Hi3516AV300 不同 ViPipe 的取值范围不同, 其中: ViPipe0/ViPipe1: [120, 3840] ViPipe2/ViPipe3: [120, 2048] • Hi3559V200 不同 ViPipe 的取值范围不同, 其中: ViPipe0: [120, 4608] ViPipe1: [120, 3840] ViPipe2/ViPipe3: [120, 2048] • Hi3516EV200 ViPipe0/ViPipe1 取值范围: [120, 2304]



成员名称	描述
	- Hi3516EV300 ViPipe0/ViPipe1 取值范围: [120, 3072] - Hi3562V100 取值范围: [120, 1536]

【注意事项】

- 水平方向起始位置与图像宽度之和应小于 sensor 输出的图像宽度。
- 垂直方向起始位置与图像高度之和应小于 sensor 输出的图像高度，由于无法检测 sensor 实际输出的宽高，当不满足该条件时 MPI 不报错。
- 使能 AF 模块时，要求图像的最小宽度为 256。
- Hi3516CV500 支持的最大分辨率为 2304*1296。仅支持 ViPipe0 与 ViPipe1，其中 ViPipe1 只支持 WDR 模式。
- Hi3516DV300 不同 ViPipe 支持的分辨率不同，ViPipe0/ViPipe1 支持的最大分辨率为 2688*1944，ViPipe2/ViPipe3 支持的最大分辨率为 2048*1536。其中 ViPipe1、ViPipe3 只支持 WDR 模式。
- Hi3516AV300 不同 ViPipe 支持的分辨率不同，ViPipe0/ViPipe1 支持的最大分辨率为 3840*2160，ViPipe2/ViPipe3 支持的最大分辨率为 2048*1536。其中 ViPipe1、ViPipe3 只支持 WDR 模式。
- Hi3559V200 不同 ViPipe 支持的分辨率不同，ViPipe0 支持的最大分辨率为 4608*3456，ViPipe1 支持的最大分辨率为 3840*2160，ViPipe2/ViPipe3 支持的最大分辨率为 2048*2048。其中 ViPipe1、ViPipe3 只支持 WDR 模式。

【相关数据类型及接口】

无

ISP_BAYER_FORMAT_E

【说明】

定义输入 Bayer 图像数据格式。

【定义】

```
typedef enum hiISP_BAYER_FORMAT_E
{
    BAYER_RGGB    = 0,
    BAYER_GRGB    = 1,
```



```
BAYER_GBRG    = 2,  
BAYER_BGGR    = 3,  
BAYER_BUTT  
} ISP_BAYER_FORMAT_E;
```

【成员】

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

【注意事项】

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

【相关数据类型及接口】

无

ISP_BAYER_RAWBIT_E

【说明】

定义输入 Bayer 图像数据位宽。

【定义】

```
typedef enum hiISP_BAYER_FORMAT_E  
{  
    BAYER_RAWBIT_8BIT    = 8,  
    BAYER_RAWBIT_10BIT   = 10,  
    BAYER_RAWBIT_12BIT   = 12,  
    BAYER_RAWBIT_14BIT   = 14,  
    BAYER_RAWBIT_16BIT   = 16,  
    BAYER_RAWBIT_BUTT  
} ISP_BAYER_FORMAT_E;
```

【成员】



成员名称	描述
BAYER_RAWBIT_8BIT	Bayer 数据位宽：8bit。
BAYER_RAWBIT_10BIT	Bayer 数据位宽：10bit。
BAYER_RAWBIT_12BIT	Bayer 数据位宽：12bit。
BAYER_RAWBIT_14BIT	Bayer 数据位宽：14bit。
BAYER_RAWBIT_16BIT	Bayer 数据位宽：16bit。

【注意事项】

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

【相关数据类型及接口】

无

SIZE_S

【说明】

定义 sensor 输出的宽高。

【定义】

```
typedef struct hiSIZE_S
{
    HI_U32 u32Width;
    HI_U32 u32Height;
} SIZE_S;
```

【成员】

成员名称	描述
u32Width	Sensor 输出宽度 - Hi3559AV100/Hi3519AV100 取值范围：[120, 8192] - Hi3516CV500/Hi3559V200 取值范围：[120, 4608] - Hi3516EV200 ViPipe0/ViPipe1：[120, 2304] - Hi3516EV300 ViPipe0/ViPipe1：[120, 3072]



成员名称	描述
u32Height	Sensor 输出高度 - Hi3559AV100/Hi3519AV100 取值范围：[120, 8192] - Hi3516CV500/Hi3559V200 取值范围[120, 4608] - Hi3516EV200 ViPipe0/ViPipe1：[120, 2304] - Hi3516EV300 ViPipe0/ViPipe1：[120, 3072]

【注意事项】

图像宽度应小于 sensor 输出的图像宽度；图像高度应小于 sensor 输出的图像高度。

【相关数据类型及接口】

无

COLOR_GAMUT_E

【说明】

定义通道色域属性。

【定义】

```
typedef enum hiCOLOR_GAMUT_E
{
    COLOR_GAMUT_BT601 = 0,
    COLOR_GAMUT_BT709,
    COLOR_GAMUT_BT2020,
    COLOR_GAMUT_USER,
    COLOR_GAMUT_BUTT
}COLOR_GAMUT_E;
```

【成员】

成员名称	描述
COLOR_GAMUT_BT601	色域范围为 BT.601
COLOR_GAMUT_BT709	色域范围为 BT.709
COLOR_GAMUT_BT2020	色域范围为 BT.2020



成员名称	描述
COLOR_GAMUT_USER	用户自定义色域范围

【注意事项】

无。

【相关数据类型及接口】

[ISP_PUB_ATTR_S](#)

ISP_PUB_ATTR_S

【说明】

定义 ISP 公共属性。

【定义】

```
typedef struct hiISP_PUB_ATTR_S
{
    RECT_S          stWndRect;
    SIZE_S          stSnsSize;
    HI_FLOAT         f32FrameRate;
    ISP_BAYER_FORMAT_E enBayer;
    WDR_MODE_E      enWDRMode;
    HI_U8           u8SnsMode;
} ISP_PUB_ATTR_S;
```

【成员】

成员名称	描述
stWndRect	裁剪窗口起始位置和图像宽高。
stSnsSize	Sensor 输出的图像宽高。
f32FrameRate	输入图像帧率，取值范围为(0.00, 65535.00]。
enBayer	Bayer 数据格式。
enWDRMode	WDR 模式选择。



成员名称	描述
u8SnsMode	用于进行 Sensor 初始化序列的选择，在分辨率和帧率相同时，配置不同的 u8SnsMode 对应不同的初始化序列；其他情况，u8SnsMode 默认配置为 0，可通过 stSnsSize 和 f32FrameRate 进行初始化序列的选择。

【注意事项】

无

【相关数据类型及接口】

无

ISP_FMW_STATE_E

【说明】

定义 ISPfirmware 状态。

【定义】

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

【成员】

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

【注意事项】

无



【相关数据类型及接口】

无

ISP_SLAVE_SNS_SYNC_S

【说明】

定义从模式 sensor 同步信号配置。

【定义】

```
typedef struct hiISP_SLAVE_SNS_SYNC_S
{
    union
    {
        struct
        {
            HI_U32 bit16Rsv      : 16;
            HI_U32 bitHInv       : 1;
            HI_U32 bitVInv       : 1;
            HI_U32 bit12Rsv      : 12;
            HI_U32 bitHEnable    : 1;
            HI_U32 bitVEnable    : 1;
        } stBits;
        HI_U32 u32Bytes;
    } unCfg;
    HI_U32 u32VsTime;
    HI_U32 u32HsTime;
    HI_U32 u32VsCyc;
    HI_U32 u32HsCyc;
    HI_U32 u32HsDlyCyc;
    HI_U32 u32SlaveModeTime;
}ISP_SLAVE_SNS_SYNC_S;
```

【成员】

成员名称	描述
bit16Rsv	保留字段。
bitHInv	XHS 极性配置。



成员名称	描述
	• 0 表示正极; • 1 表示负极。
bitVInv	XVS 极性配置。
bit12Rsv	保留字段。
bitHEnable	XHS 输出使能。
bitVEnable	XVS 输出使能。
u32VsTime	XVS 信号周期, 单位: sensor 输入时钟周期。
u32HsTime	XHS 信号周期, 单位: sensor 输入时钟周期。
u32VsCyc	XVS 有效电平宽度, 单位: sensor 输入时钟周期。
u32HsCyc	XHS 有效电平宽度, 单位: sensor 输入时钟周期。
u32HsDlyCyc	XHS 脉冲输出相对 XVS 脉冲的延迟周期配置, 单位: sensor 输入时钟周期。
u32SlaveModeTime	Sensor 从模式时序配置选择寄存器: 0: 选择 SENSOR0 timing 配置; 1: 选择 SENSOR1 timing 配置; 2: 选择 SENSOR2 timing 配置; 3: 选择 SENSOR3 timing 配置。

【注意事项】

如图 2-4~图 2-6 所示, 说明了同步信号发生模块配置参数的含义。

图2-4 同步信号配置时序图

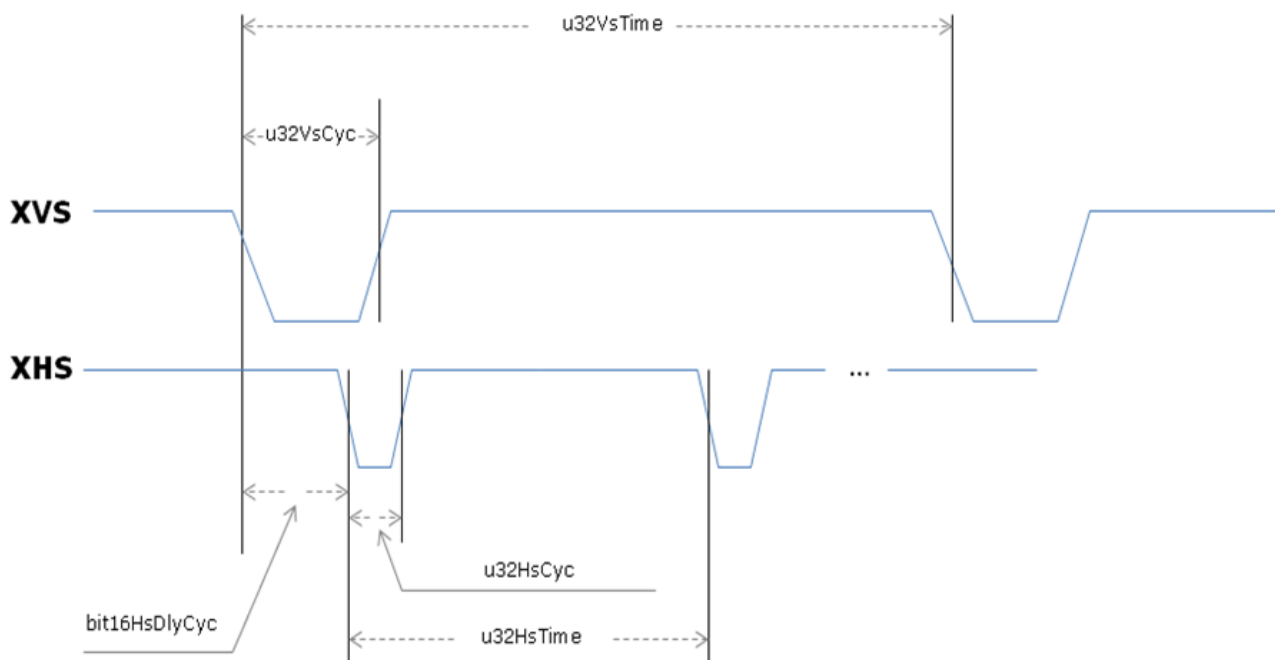


图2-5 同步信号极性翻转

`unCfg.stBits.bitVInv = 0`

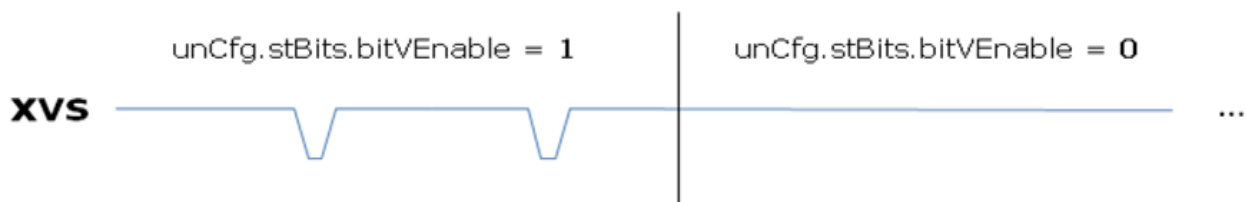


`unCfg.stBits.bitVInv = 1`





图2-6 同步信号使能



【相关数据类型及接口】

无

ISP_WDR_MODE_S

【说明】

定义 ISP 宽动态模式。

【定义】

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

【成员】

成员名称	描述
enWDRMode	宽动态模式。

【注意事项】

无

【相关数据类型及接口】

无

WDR_MODE_E

【说明】

定义宽动态模式。



【定义】

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

【成员】

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



成员名称	描述
WDR_MODE_4To1_FRAME_FULL_RATE	4 帧合成帧 WDR 全帧率模式。

【注意事项】

WDR_MODE_BUILT_IN 需要 sensor 支持。

【相关数据类型及接口】

无

ISP_MODULE_CTRL_U

【说明】

定义 ISP 功能模块的控制。

【定义】

```
typedef union hiISP_MODULE_CTRL_U
{
    HI_U64  u64Key;
    struct
    {
        HI_U64  bitBypassSPDGain      : 1 ; /* RW:[0] */
        HI_U64  bitBypassAntiFC       : 1 ; /* RW:[1] */
        HI_U64  bitBypassCrosstalkR   : 1 ; /* RW:[2] */
        HI_U64  bitBypassDPC          : 1 ; /* RW:[3] */
        HI_U64  bitBypassNR            : 1 ; /* RW:[4] */
        HI_U64  bitBypassDehaze        : 1 ; /* RW:[5] */
        HI_U64  bitBypassWBGain        : 1 ; /* RW:[6] */
        HI_U64  bitBypassMeshShading   : 1 ; /* RW:[7] */
        HI_U64  bitBypassDRC           : 1 ; /* RW:[8] */
        HI_U64  bitBypassDemosaic      : 1 ; /* RW:[9] */
        HI_U64  bitBypassColorMatrix   : 1 ; /* RW:[10] */
        HI_U64  bitBypassGamma         : 1 ; /* RW:[11] */
        HI_U64  bitBypassFSWDR         : 1 ; /* RW:[12] */
        HI_U64  bitBypassCA            : 1 ; /* RW:[13] */
        HI_U64  bitBypassCsConv        : 1 ; /* RW:[14] */
        HI_U64  bitBypassRadialCrop    : 1 ; /* RW:[15] */
    };
};
```



```

HI_U64 bitBypassSharpen      : 1 ; /* RW;[16] */
HI_U64 bitBypassLCAC         : 1 ; /* RW;[17] */
HI_U64 bitBypassGCAC         : 1 ; /* RW;[18] */
HI_U64 bit2ChnSelect         : 2 ; /* RW;[19:20] */
HI_U64 bitBypassLdci         : 1 ; /* RW;[21] */
HI_U64 bitBypassPreGamma     : 1 ; /* RW;[22] */
HI_U64 bitBypassRadialShading : 1 ; /* RW;[23] */
HI_U64 bitBypassAStatFE      : 1 ; /* RW;[24] */
HI_U64 bitBypassAStatBE      : 1 ; /* RW;[25] */
HI_U64 bitBypassMGStat       : 1 ; /* RW;[26] */
HI_U64 bitBypassDE           : 1 ; /* RW;[27] */
HI_U64 bitBypassAFStatFE     : 1 ; /* RW;[28] */
HI_U64 bitBypassAFStatBE     : 1 ; /* RW;[29] */
HI_U64 bitBypassAWBStat      : 1 ; /* RW;[30] */
HI_U64 bitBypassCLUT         : 1 ; /* RW;[31] */
HI_U64 bitBypassHLC          : 1 ; /* RW;[32] */
HI_U64 bitBypassEdgeMark     : 1 ; /* RW;[33] */
HI_U64 bitBypassRGBIR        : 1 ; /* RW;[34] ; Not support*/
HI_U64 bitRsv29              : 29 ; /* H  ; [35:63] */

};
} ISP_MODULE_CTRL_U;

```

【成员】

成员名称	描述
u64Key	结构体枚举的整型值。
bitBypassISPDGain	旁路数字增益。
bitBypassAntiFC	旁路去伪彩功能。
bitBypassCrosstalkR	旁路 Crosstalk Removal。
bitBypassDPC	旁路坏点校正。
bitBypassNR	旁路去噪。
bitBypassDehaze	旁路去雾。
bitBypassWBGain	旁路白平衡增益和偏移量。



成员名称	描述
bitBypassMeshShading	旁路镜头阴影校正。
bitBypassDRC	旁路 DRC。
bitBypassDemosaic	旁路去马赛克模块。
bitBypassColorMatrix	旁路颜色矩阵。
bitBypassGamma	旁路 Gamma 表。
bitBypassFSWDR	旁路多帧合成 WDR。
bitBypassCA	旁路 CA。
bitBypassCsConv	旁路 CSC 转换。
bitBypassRadialCrop	旁路 RadialCrop, Hi3516CV500/Hi3559V200 不支持
bitBypassSharpen	旁路 Sharpen。
bitBypassLCAC	旁路 Local CAC。
bitBypassGCAC	旁路 Global CAC。Hi3516EV200/Hi3516EV300 不支持
bit2ChnSelect	WDR 模式主路数据来源, 一般在旁路多帧合成 WDR 模块后用于 debug。 0: 主路数据来源于超短帧; 1: 主路数据来源于短帧; 2: 主路数据来源于中帧; 3: 主路数据来源于长帧。
bitBypassLdci	旁路 Local DCI。
bitBypassPreGamma	旁路 PreGamma。
bitBypassRadialShading	旁路 Radial Shading, 仅 Hi3559AV100/Hi3519AV100 支持。
bitBypassAESTatFE	旁路位于 FE 的 AE 统计信息。
bitBypassAESTatBE	旁路位于 BE 的 AE 统计信息。
bitBypassMGSta	旁路 MG 统计信息。



成员名称	描述
bitBypassDE	旁路 Detail Enhance。仅 Hi3516CV500/Hi3559V200 支持。
bitBypassAFStatFE	旁路位于 FE 的 AF 统计信息。仅 Hi3559AV100/Hi3519AV100 支持。
bitBypassAFStatBE	旁路位于 BE 的 AF 统计信息。
bitBypassAWBStat	旁路 AWB 统计信息。
bitBypassCLUT	旁路 CLUT。Hi3516EV200/Hi3516EV300 不支持。
bitBypassHLC	旁路 HLC。Hi3559AV100/Hi3516EV200/Hi3516EV300 不支持
bitBypassEdgeMark	旁路 EdgeMark。Hi3516EV200/Hi3516EV300 不支持。
bitBypassRGBIR	仅 Hi3516EV200/Hi3516EV300 支持。
bitRsv29	保留位。

【注意事项】

WDR 模式下，开关 WDR 模块的使能，图像会有几帧颜色表现异常。

【相关数据类型及接口】

无

ISP_VD_TYPE_E

【说明】

定义 ISP 场同步信号。

【定义】

```
typedef struct hiISP_VD_TYPE_E
{
    ISP_VD_FE_START    = 0,
    ISP_VD_FE_END,
    ISP_VD_BE_END,
    ISP_VD_BUTT
```




```
} hiISP_VD_TYPE_E;
```

【成员】

成员名称	描述
ISP_VD_FE_START	FE 帧起始
ISP_VD_FE_END	FE 帧结束
ISP_VD_BE_END	BE 帧结束

【注意事项】

在线和并行模式下不支持 ISP_VD_BE_END 方式。

【相关数据类型及接口】

无

ISP_SNS_ATTR_INFO_S

【说明】

定义 ISP sensor 属性。

【定义】

```
typedef struct hiISP_SNS_ATTR_INFO_S  
{  
    SENSOR_ID          eSensorId;  
} ISP_SNS_ATTR_INFO_S;
```

【成员】

成员名称	描述
eSensorId	Sensor ID 号。

【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)(VI_PIPE ViPipe) ;
    HI_VOID(*pfn_cmos_sensor_global_init)(VI_PIPE ViPipe);
    HI_VOID(*pfn_cmos_sensor_exit)(VI_PIPE ViPipe);
    HI_S32(*pfn_cmos_set_image_mode)(VI_PIPE ViPipe, ISP_CMOS_SENSOR_IMAGE_MODE_S
    *pstSensorImageMode);
```



```

HI_S32(*pfn_cmos_set_wdr_mode)(VI_PIPE ViPipe,HI_U8 u8Mode);
HI_U32(*pfn_cmos_get_isp_default)(VI_PIPE ViPipe,ISP_CMOS_DEFAULT_S *pstDef);
HI_U32(*pfn_cmos_get_isp_black_level)(VI_PIPE ViPipe, ISP_CMOS_BLACK_LEVEL_S
*pstBlackLevel);
HI_S32(*pfn_cmos_get_sns_reg_info)(VI_PIPE ViPipe,ISP_SNS_REGS_INFO_S *pstSnsRegsInfo);
HI_VOID(*pfn_cmos_set_pixel_detect)(VI_PIPE ViPipe,HI_BOOL bEnable);
HI_S32 (*pfn_cmos_get_awb_gains)(VI_PIPE ViPipe, HI_U32 *pu32SensorAwbGain);
} ISP_SENSOR_EXP_FUNC_S;

```

【成员】

成员名称	描述
pfn_cmos_sensor_init	初始化 sensor 的回调函数指针。
pfn_cmos_sensor_exit	sensor 的回调退出函数指针。
pfn_cmos_sensor_global_init	初始化全局变量的回调函数指针。
pfn_cmos_set_image_mode	设置分辨率和帧率切换的回调函数指针。返回值 0 表示 sensor 模式发生改变，ISP 会调用 pfn_cmos_sensor_init 重新配置 sensor；返回值为 -2 sensor 模式没有变化，ISP 不会重新配置 sensor。
pfn_cmos_set_wdr_mode	设置 wdr 模式的回调函数指针。
pfn_cmos_get_isp_default	获取 ISP 基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取 sensor 的黑电平值的回调函数指针，支持根据 sensor 增益动态调整黑电平值。若此处动态调整黑电平值，则外部只能通过接口 HI_MPI_ISP_SetBlackLevelAttr 的手动模式设置黑电平。
pfn_cmos_get_sns_reg_info	获取 sensor 寄存器信息的回调函数指针，用于实现内核态配置 AE 信息。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_get_awb_gains	获取 AWB 增益的回调函数指针。



【注意事项】

- pfn_cmos_sensor_init, pfn_cmos_get_isp_default, pfn_cmos_get_isp_black_level, pfn_cmos_set_pixel_detect 和 pfn_cmos_get_sns_reg_info 必须赋值，其他回调函数指针如果不需要赋值，应置为 NULL。例如有的 Sensor 不支持切换分辨率，那么 pfn_cmos_set_image_mode 需要置为 NULL。
- Hi3559AV100、Hi3519AV100 支持 AWB 增益在 Sensor 配置；拼接模式下，不支持 AWB 增益在 Sensor 配置。
- pfn_cmos_get_awb_gains 实现将当前场景的 AWB 增益写入 [ISP_SNS_STATE_S](#) 结构体的 au32SensorWBGain 参数。pfn_cmos_get_awb_gains 赋值，设置 ISP_CMOS_DEFAULT_S 结构体 bit1Awb 参数为 1，设置 enAwbGainSwitch = ISP_AWB_GAIN_IN_SENSOR，则 AWB 增益在 Sensor 配置。用户需要将 au32SensorWBGain 的 AWB 增益写入 sensor 对应寄存器。pfn_cmos_get_awb_gains 置为 NULL，AWB 增益在 ISP 配置。
- [ISP_SNS_STATE_S](#) 结构体的 au32SensorWBGain 参数的 AWB 增益范围是[0x0, 0xFFF]。如果 Sensor 支持的 AWB 增益范围小于 16x，可能导致高低色温下 AWB 不能完全还原。如果通过 Sensor 的 RGGB 4 个通道的 Digital Gain 寄存器实现 AWB 功能，需要限制 AE 的 Digital Gain 上限，保证低照下 AE Digital Gain * AWB Gain 在 Sensor Digital Gain 寄存器取值范围内。
- 不支持切换 AWB 增益配置位置。

【相关数据类型及接口】

- [ISP_SENSOR_REGISTER_S](#)
- [ISP_SNS_STATE_S](#)
- [ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_SENSOR_IMAGE_MODE_S

【说明】

定义 sensor 输出的宽高和帧率属性。

【定义】

```
typedef struct hiISP_CMOS_SENSOR_IMAGE_MODE_S
{
    HI_U16  u16Width;
    HI_U16  u16Height;
    HI_FLOAT f32Fps;
    HI_U8  u8SnsMode;
```



```
ISP_CMOS_SENSOR_IMAGE_MODE_S;
```

【成员】

成员名称	描述
u16Width	Sensor 输出的宽度。
u16Height	Sensor 输出的高度。
f32Fps	Sensor 输出的帧率。
u8SnsMode	用于进行 Sensor 初始化序列的选择，在分辨率和帧率相同时，配置不同的 u8SnsMode 对应不同的初始化序列；其他情况，u8SnsMode 默认为 0

【注意事项】

无

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

ISP_CMOS_DRC_S

【说明】

DRC 模块参数定义。

【定义】

```
typedef struct hiISP_CMOS_DRC_S
{
    HI_BOOL bEnable;
    ISP\_OP\_TYPE\_E enOpType;
    HI_U16 u16ManualStrength;
    HI_U16 u16AutoStrength;
    HI_U8 u8SpatialFltCoef;
    HI_U8 u8RangeFltCoef;
    HI_U8 u8ContrastControl;
    HI_S8 s8DetailAdjustFactor;
    HI_U8 u8RangeAdaMax;
    HI_U8 u8FltScaleFine;
```



```
    HI_U8    u8FltScaleCoarse;
    HI_U8    u8GradRevMax;
HI_U8    u8GradRevThr;
HI_U8    u8BrightGainLmt;
    HI_U8    u8BrightGainLmtStep;
    HI_U8    u8DarkGainLmtY;
    HI_U8    u8DarkGainLmtC;
    HI_U8    u8PDStrength;
    HI_U8    u8LocalMixingBrightMax;
HI_U8    u8LocalMixingBrightMin;
HI_U8    u8LocalMixingBrightThr;
    HI_S8    s8LocalMixingBrightSlo;
    HI_U8    u8LocalMixingDarkMax;
HI_U8    u8LocalMixingDarkMin;
HI_U8    u8LocalMixingDarkThr;
    HI_S8    s8LocalMixingDarkSlo;
HI_U16    u16ColorCorrectionLut[HI_ISP_DRC_CC_NODE_NUM];
HI_U16    u16ToneMappingValue[HI_ISP_DRC_TM_NODE_NUM];
    HI_U8    u8Asymmetry;
    HI_U8    u8SecondPole;
    HI_U8    u8Stretch;
    HI_U8    u8Compress;
    HI_U8    u8CurveSel;
    HI_U16    au16Xpoint[HI_ISP_DRC_CUBIC_POINT_NUM];
    HI_U16    au16Ypoint[HI_ISP_DRC_CUBIC_POINT_NUM];
    HI_U16    au16Slope[HI_ISP_DRC_CUBIC_POINT_NUM];
} ISP_CMOS_DRC_S;
```

【成员】

成员名称	描述
bEnable	DRC 动态范围压缩使能，取值范围：[0, 1]
enOpType	DRC 工作模式选择。 - OP_TYPE_AUTO：自动， - OP_TYPE_MANUAL：手动。 仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300



成员名称	描述
	支持。
u16ManualStrength	DRC 手动模式默认强度。 - Hi3559AV100 取值范围: [0x0, 0xFF] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围: [0x0, 0x3FF]
u16AutoStrength	DRC 自动模式默认强度。 - Hi3559AV100 取值范围: [0x80, 0xFF] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围: [0x0, 0x3FF]。
u8SpatialFltCoef	控制空域滤波系数。值越大, 运动 halo 越不明显, 细节越少, 值越小, 细节表现越好, 运动 halo 越明显。 - Hi3559AV100 取值范围: [0x0, 0xA]; - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围: [0x0, 0x5]。
u8RangeFltCoef	控制值域滤波系数。值越大, halo 越明显; 值越小, halo 表现越好, 但是在强边缘处可能会出现边线。取值范围: [0x0, 0xA]。
u8ContrastControl	全局对比度控制参数, 取值范围: [0x0, 0xF]。参数效果与图像的亮度分布相关, 一般建议值为 6 到 10 之间。
s8DetailAdjustFactor	细节微调系数。值越大, 整体细节越强; 值越小, 整体细节越弱。取值范围: [-15, 15]。
u8RangeAdaMax	控制值域滤波系数的自适应范围。取值范围[0x0, 0x8]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8FltScaleFine	控制滤波窗口大小。值越大, 运动 halo 越不明显, 局部细节越弱。取值范围: [0x0, 0xF]。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
u8FltScaleCoarse	控制滤波窗口大小。值越大, 运动 halo 越不明显, 局部细节越弱。相比 u8FltScaleFine, 其调节步长更大。取值



成员名称	描述
	范围：[0x0, 0xF]。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
u8GradRevMax	去边线强度。值越大，边线减弱越明显，但是可能会带来细节损失。取值范围：[0x0, 0x40]。
u8GradRevThr	边线检测阈值。值越大，边线减弱越明显，但是可能会带来细节损失。取值范围：[0x0, 0x80]。
u8BrightGainLmt	亮区亮度增益限制目标值；值越大，限制越大。取值范围：[0x0, 0xF]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8BrightGainLmtStep	亮区亮度增益限制自适应步长；值越小，限制越大。取值范围：[0x0, 0xF]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8DarkGainLmtY	暗区亮度增益限制；值越大，限制越大。取值范围：[0x0, 0x85]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8DarkGainLmtC	暗区色度增益限制；值越大，限制越大。取值范围：[0x0, 0x85]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8LocalMixingBrightMax	用来控制正向细节的增益最大值。取值范围：[0x0, 0x80]。
u8LocalMixingBrightMin	用来控制正向细节的增益最小值。取值范围：[0x0, 0x40]。
u8LocalMixingBrightThr	线性细节增强：用来控制正向细节增益随亮度自适应的阈值。取值范围：[0x0, 0xFF]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
s8LocalMixingBrightSl	线性细节增强：用来控制正向细节增益随亮度自适应的斜



成员名称	描述
o	率。取值范围：[-7, 7]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8LocalMixingDarkMax	用来控制负向细节的增益最大值。取值范围：[0x0, 0x80]。
u8LocalMixingDarkMin	用来控制负向细节的增益最小值。取值范围：[0x0, 0x40]。
u8LocalMixingDarkThr	线性细节增强：用来控制负向细节增益随亮度自适应的阈值。取值范围：[0x0, 0xFF]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
s8LocalMixingDarkSlo	线性细节增强：用来控制负向细节增益随亮度自适应的斜率。取值范围：[-7, 7]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u16ColorCorrectionLut [HI_ISP_DRC_CC_NOD E_NUM]	颜色校正表，用来校正饱和度的表。取值范围：[0x0, 0x400]。
u16ToneMappingValue [HI_ISP_DRC_TM_NO DE_NUM]	用户自定义曲线 LUT，取值范围：[0x0, 0xFFFF]。仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u8Asymmetry	用来生成 Asymmetry Curve，值越小，暗区拉伸越大。取值范围：[0x1, 0x1E]。
u8SecondPole	用来生成 Asymmetry Curve，值越大，整体亮度拉伸越大。取值范围：[0x96, 0xD2]。
u8Stretch	用来生成 Asymmetry Curve，值越小，整体图像越亮。取值范围：[0x1E, 0x3C]。
u8Compress	用来生成 Asymmetry Curve，值越大，在垂直方向压缩曲线程度越大，整体亮度变低。取值范围：[0x64, 0xC8]。
u8CurveSel	选择 tone mapping 曲线。值为 0，选择 Asymmetry curve；值为 1，选择 Cubic curve (仅 Hi3559AV100 支



成员名称	描述
	持), 值为 2, 选择用户自定义曲线。
au16Xpoint[HI_ISP_DR C_CUBIC_POINT_NUM]	Cubic curve 点的横坐标, 取值范围: [0x0, 0x3E8]。 仅 Hi3559AV100 支持。
au16Ypoint[HI_ISP_DR C_CUBIC_POINT_NUM]	Cubic curve 点的纵坐标, 取值范围: [0x0, 0x3E8]。 仅 Hi3559AV100 支持。
au16Slope[HI_ISP_DR C_CUBIC_POINT_NUM]	Cubic curve 点的斜率, 取值范围: [0x0, 0x2710]。 仅 Hi3559AV100 支持。

【注意事项】

无

【相关数据类型及接口】

ISP_CMOS_DEFAULT_S

ISP_CMOS_DEMOSAIC_S

【说明】

定义 demosaic 参数。

【定义】

Hi3559AV100 定义:

```
typedef struct hiISP_CMOS_DEMOSAIC_S
{
    HI_BOOL bEnable;
    HI_U8   au8NonDirMFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8NonDirHFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8NonDirStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8DetailSmoothRange[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16  au16DetailSmoothStr[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_DEMOSAIC_S;
```

Hi3519AV100/Hi3516CV500/Hi3559V200 定义:



```
typedef struct hiISP_CMOS_DEMOSAIC_S
{
    HI_BOOL bEnable;
    HI_U8   au8NonDirStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8NonDirMFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8NonDirHFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8   au8DetailSmoothRange[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_DEMOSAIC_S;
```

【成员】

成员名称	描述
bEnable	使能去马赛克模块
au8NonDirStr	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0, 0xff]。
au8NonDirMFDetailEhcStr	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 - Hi3559AV100 取值范围：[0x0, 0x10] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0x7F]
au8NonDirHFDetailEhcStr	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0, 0x10]。
au8DetailSmoothRange	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 - Hi3559AV100 取值范围：[0x1, 0x8] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x1, 0x7]
au16DetailSmoothStr	细节平滑强度，值越大，平滑处理强度越大，对伪细节的抑制强度越大。取值范围：[0, 0x100]。 仅 Hi3559AV100 支持



【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_PREGAMMA_S

【说明】

定义 PreGamma 参数。

【定义】

```
typedef struct hiISP_CMOS_PREGAMMA_S
{
    HI_BOOL bEnable;
    HI_U32  au32PreGamma[PREGAMMA_NODE_NUM];
} ISP_CMOS_PREGAMMA_S;
```



说明

#define PREGAMMA_NODE_NUM 257

【成员】

成员名称	描述
bEnable	模块使能
au32PreGamma	PreGamma 表 - Hi3559AV100/Hi3519AV100 取值范围为[0, 0x100000] - Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围为[0, 0xFFFFF]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)



ISP_CMOS_GAMMA_S

【说明】

定义 Gamma 结构体。

【定义】

Hi3559AV100/Hi3519AV100 定义:

```
#define GAMMA_NODE_NUMBER 1025
typedef struct hiISP_CMOS_GAMMA_S
{
    HI_U16 au16Gamma[GAMMA_NODE_NUMBER];
} ISP_CMOS_GAMMA_S;
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 定义:

```
#define GAMMA_EXT_NODE_NUMBER 1025
typedef struct hiISP_CMOS_GAMMA_S
{
    HI_U16 au16Gamma[GAMMA_EXT_NODE_NUMBER];
} ISP_CMOS_GAMMA_S;
```

【成员】

成员名称	描述
au16Gamma	Gamma 表, 取值范围为[0, 0xFFFF]。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_SHARPEN_MANUAL_S

【说明】

定义 sharpen manual 档 cmos 参数。

【定义】



Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 定义:

typedef struct HiISP_CMOS_SHARPEN_MANUAL_S

```
{
    HI_U8  au8LumaWgt[ISP_SHARPEN_LUMA_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM];
    HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM];
    HI_U16 u16TextureFreq;
    HI_U16 u16EdgeFreq;
    HI_U8  u8OverShoot;
    HI_U8  u8UnderShoot;
    HI_U8  u8ShootSupStr;
    HI_U8  u8ShootSupAdj;
    HI_U8  u8DetailCtrl;
    HI_U8  u8DetailCtrlThr;
    HI_U8  u8EdgeFiltStr;
    HI_U8  u8EdgeFiltMaxCap;
    HI_U8  u8RGain;
    HI_U8  u8GGain;
    HI_U8  u8BGain;
    HI_U8  u8SkinGain;
    HI_U16 u16MaxSharpGain;
} ISP_CMOS_SHARPEN_MANUAL_S;
```

Hi3516EV200/ Hi3516EV300 定义:

typedef struct HiISP_CMOS_SHARPEN_MANUAL_S

```
{
    HI_U8  au8LumaWgt[ISP_SHARPEN_LUMA_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM];
    HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM];
    HI_U16 u16TextureFreq;
    HI_U16 u16EdgeFreq;
    HI_U8  u8OverShoot;
    HI_U8  u8UnderShoot;
    HI_U8  u8ShootSupStr;
    HI_U8  u8ShootSupAdj;
    HI_U8  u8DetailCtrl;
    HI_U8  u8DetailCtrlThr;
```



```
HI_U8 u8EdgeFiltStr;  
HI_U8 u8EdgeFiltMaxCap;  
HI_U8 u8RGain;  
HI_U8 u8GGain;  
    HI_U8 u8BGain;  
    HI_U8 u8SkinGain;  
HI_U16 u16MaxSharpGain;  
HI_U8 u8WeakDetailGain;  
} ISP_CMOS_SHARPEN_MANUAL_S;
```

【成员】

成员名称	描述
au8LumaWgt[ISP_SHARPE N_LUMA_NUM]	亮度锐化权重。满量程 0-255 的亮度被 32 个等分点平均分为 32 段亮度区间，每一段亮度区间对应一个亮度权重。比如 0-7 的亮度区间的权重是 au8LumaWgt[0]，8-15 的亮度区间的权重是 au8LumaWgt[1]，依次类推。如图 6-1 所示。 该值越大，图像锐化程度越强，反之，越弱。 Hi3516CV500/Hi3559V200 取值范围：[0, 127]。建议值 127。 Hi3516EV200/Hi3516EV300 取值范围：[0, 31]。建议值 31。
au16TextureStr [ISP_SHARPEN_GAIN_NUM]	无方向的细节纹理的锐化强度，设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。该参数是一个 ISP_SHARPEN_GAIN_NUM=32 的数组，是一个 32 段的连续的强度曲线，如图 6-2 所示。 取值范围：[0, 4095]。建议值 300。
au16EdgeStr [ISP_SHARPEN_GAIN_NUM]	带方向的边缘的锐化强度，设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。该参数是一个 ISP_SHARPEN_GAIN_NUM=32 的数组，是一个 32 段的连续的强度曲线，如图 6-3 所示。



成员名称	描述
	取值范围：[0, 4095]。建议值 400。
u16TextureFreq	图像的无方向细节纹理的增强频段控制。设置图像的细节纹理增强的频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。u16TextureFreq 对应于强度参数 u16TextureStr。u16TextureFreq 越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]。建议值 128。
u16EdgeFreq	图像的带方向的边缘的增强频段控制。设置图像边缘增强的频率。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。u16EdgeFreq 对应于强度参数 u16EdgeStr。u16EdgeFreq 越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]。建议值 96。
u8OverShoot	设置图像的 overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 100。
u8UnderShoot	设置图像的 undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 127。
u8ShootSupStr	图像锐化后的 overshoot 和 undershoot 的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的 overshoot 和 undershoot 的宽度和幅



成员名称	描述
	度。 该值越大，锐化后的图像的 overshoot 和 undershoot 的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。该参数需要和 ShootSupAdj 配合使用。 取值范围：[0, 255]。建议值 8。
u8ShootSupAdj	图像锐化后的 overshoot 和 undershoot 的抑制强度的调节。该参数配合 ShootSupStr 使用，用于调节 ShootSupStr 作用的区域范围。该值越小，则越多的纹理区域的 shoot 会被 ShootSupStr 抑制；该值越大，则只有很强的边缘的 shoot 会被 ShootSupStr 抑制，纹理区域的 shoot 不会被抑制。 取值范围：[0,15]。建议值 9。
u8DetailCtrl	图像的细节纹理区的 shoot 强度的控制。用于控制图像的细节纹理区域的 shoot 强度，shoot 越大，细节纹理区的清晰度越高。 取值范围：[0, 0xFF]。该值等于 128，则图像的细节纹理区域的 shoot 强度和大边缘的 shoot 强度一致，都分别等于 u8OverShoot 和 u8UnderShoot。该值大于 128，则图像的细节纹理的 shoot 强度大于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。该值小于 128，则图像的细节纹理的 shoot 强度小于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。
u8DetailCtrlThr	图像的细节纹理区的 shoot 强度的控制阈值。该值配合 DetailCtrl 使用，用于区分 DetailCtrl 所控制 shoot 的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的 shoot 会被 DetailCtrl 单独控制，而大于该阈值的边缘区域的 shoot 依然等于 u8OverShoot 和 u8UnderShoot。



成员名称	描述
	取值范围：[0, 0xFF]，建议值 160。
u8EdgeFiltStr	边缘滤波强度调试参数：实现图像锐化边缘的范围和边缘平滑强度的控制。该值越大，判为边缘的区域越多、也越宽，EdgeStr 起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，EdgeStr 起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]；建议值 53。
u8EdgeFiltMaxCap	边缘滤波强度范围的调试参数：该值越大，边缘滤波的最大强度也最大，u8EdgeFiltStr 的可调试范围也越大；一般建议该值大小控制 30 以内。 取值范围：[0, 47]；建议值 18。
u8RGain	深红色区域的锐化增益控制。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。
u8GGain	绿色区域的锐化增益控制。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]；建议值 32。
u8BGain	深蓝色区域的锐化增益控制。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。
u8SkinGain	肤色区域的锐化增益控制。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]；建议值 23。
u16MaxSharpGain	图像锐化的最大增益限制值。该值越大，则图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，可以减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值 160。



成员名称	描述
u8WeakDetailGain	对弱纹理单独进行锐化的增益控制。该值等于 0，图像弱纹理和图像其他区域的锐化程度相当，该值越大，图像弱纹理的锐化程度越大，当该参数过大，可能带来白边加重甚至虚边的副作用。 取值范围：[0, 127]，建议值 0。 仅 Hi3516EV200/Hi3516EV300 支持该参数。

ISP_CMOS_SHARPEN_AUTO_S

【说明】

定义 sharpen auto 档 cmos 参数。

【定义】

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 定义：

```
typedef struct HiISP_CMOS_SHARPEN_AUTO_S
{
    HI_U8  au8LumaWgt[ISP_SHARPEN_LUMA_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8OverShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8UnderShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8ShootSupStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8ShootSupAdj[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8DetailCtrl[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8DetailCtrlThr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8EdgeFiltStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8EdgeFiltMaxCap[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8RGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8GGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8BGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8SkinGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16MaxSharpGain[ISP_AUTO_ISO_STRENGTH_NUM];
}
```



```
} ISP_CMOS_SHARPEN_AUTO_S;
```

Hi3516EV200/Hi3516EV300 定义:

```
typedef struct HiISP_CMOS_SHARPEN_AUTO_S
{
    HI_U8  au8LumaWgt[ISP_SHARPEN_LUMA_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8OverShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8UnderShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8ShootSupStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8ShootSupAdj[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8DetailCtrl[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8DetailCtrlThr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8EdgeFiltStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8EdgeFiltMaxCap[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8RGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8GGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8BGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8SkinGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16MaxSharpGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8  au8WeakDetailGain [ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_SHARPEN_AUTO_S;
```

【注意事项】

Auto 档参数分别对应 sensor 在 16 种不同增益 (Again*Dgain* ISPDgain(times)) 情况下的设置值, 成员参数的含义与 Manual 档 cmos 成员参数一致。16 种增益为: 1、2、4、8、16、32、64、128、256、512、1024、2048、4096、8192、16384、32768。

ISP_CMOS_SHARPEN_S

【说明】

定义 sharpen 参数。

【定义】



```
typedef struct hiISP_CMOS_SHARPEN_S
{
    HI_U8 u8SkinUmin;
    HI_U8 u8SkinVmin;
    HI_U8 u8SkinUmax;
    HI_U8 u8SkinVmax;
    ISP_CMOS_SHARPEN_MANUAL_S stManual;
    ISP_CMOS_SHARPEN_AUTO_S stAuto;
} ISP_CMOS_SHARPEN_S;
```

【成员】

成员名称	描述
u8SkinUmin	肤色区域范围矩形窗的左下角的最小坐标的 U 值
u8SkinVmin	肤色区域范围矩形窗的左下角的最小坐标的 V 值
u8SkinUmax	肤色区域范围矩形窗的右上角的最大坐标的 U 值
u8SkinVmax	肤色区域范围矩形窗的右上角的最大坐标的 V 值
stManual	Sharpen 手动参数, 具体参见 ISP_CMOS_SHARPEN_MANUAL_S
stAuto	Sharpen 自动参数, 具体参见 ISP_CMOS_SHARPEN_AUTO_S

【注意事项】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 无 3DNR 后 Sharpen。

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_EDGEMARK_S

【说明】

定义 edgemark 的参数。

【定义】

```
typedef struct hiISP_CMOS_EDGEMARK_S
{
    HI_BOOL bEnable;
```



```
    HI_U8    u8Threshold;  
    HI_U32   u32Color;  
} ISP_CMOS_EDGEMARK_S;
```

【成员】

成员名称	描述
bEnable	EdgeMark 边缘标记功能的使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u8Threshold	边缘标记的阈值。计算得到的图像的局部统计量大于该阈值, 则认为是强边缘或者强的纹理特征, 将大于该阈值的强边缘和强的纹理特征标记上色。 取值范围: [0, 255]。默认值 100。 该值越小, 则被标记的边缘和强的纹理特征越多。
u32Color	边缘标记的颜色设置值。该值设置的是 RGB 三分量值的大小。该值的 0-7bit 的低 8 位是表示 B 分量的大小, 8-15bit 的中间 8 位是 G 分量的大小, 16-23bit 的高 8 位是 R 分量的大小, 24-31bit 的最高 8 位保留不用。通过设置 u32Color 可以为边缘标记的边缘标记上色选择合适的颜色, 一般所选的颜色最好和图像的背景颜色形成明显的反差对比。 取值范围: [0, 0xFFFFFF]。默认值 0xFF0000。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_HLC_S

【说明】

定义 HLC 的参数。



【定义】

```
typedef struct hiISP_CMOS_HLC_S
{
    HI_BOOL bEnable;
    HI_U8 u8LumaThr;
    HI_U8 u8LumaTarget;
} ISP_CMOS_HLC_S;
```

【成员】

成员名称	描述
bEnable	HLC 高光遮蔽功能的使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u8LumaThr	高光遮蔽的阈值。亮度值大于该值的区域将会被拉灰成目标值为 u8LumaTarget 的低亮度值。 取值范围: [0, 255]。默认值 248。 该值一般都设置的比较大, 否则, 图像的很多区域都会被拉灰。
u8LumaTarget	高光遮蔽拉灰的目标值。亮度值大于 u8LumaThr 的区域将被拉灰成目标值为 u8LumaTarget 的低亮度值, 也即该区域的亮度将会被明显拉低拉暗。 取值范围: [0, 255]。默认值 10。 该值一般都设置的比较小, 以减少高亮光源对人眼的刺激。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_LDCI_S

【说明】



定义 LDCI 参数。

【定义】

```
typedef struct hiISP_CMOS_LDCI_S
{
    HI_BOOL  bEnable;
    HI_U8    u8GaussLPFSigma;
    HI_U8    au8HePosWgt[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8HePosSigma[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8HePosMean[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8HeNegWgt[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8HeNegSigma[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8HeNegMean[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16BlcCtrl[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_LDCI_S;
```

【成员】

成员名称	描述
bEnable	使能模块。
au8HePosWgt	Gauss 分布曲线权重，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。 - Hi3559AV100/Hi3519AV100 取值范围：[0x0, 0x80] - Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0xFF]
au8HePosSigma	Gauss 分布曲线标准差，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。取值范围：[0x1, 0xFF]
au8HePosMean	Gauss 分布曲线数学期望，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。取值范围：[0x0, 0xFF]
au8HeNegWgt	Gauss 分布曲线权重，可根据原图亮度控制减暗程度；曲线呈高斯分布，横坐标为亮度。取值范围：[0x0, 0x80]
au8HeNegSigma	Gauss 分布曲线标准差，可根据原图亮度控制减暗程度；曲线呈高斯分布，横坐标为亮度。取值范围：[0x1, 0xFF]
au8HeNegMean	Gauss 分布曲线数学期望，可根据原图亮度控制减暗程度；曲线



成员名称	描述
	呈高斯分布，横坐标为亮度。取值范围：[0x0, 0xFF]
au16BlcCtrl	自动模式下暗区增益控制强度，用于抑制暗区噪声，取值范围：[0x0, 0xFF]。

【注意事项】

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_DPC_S

【说明】

定义 DPC 参数。

【定义】

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 定义：

```
typedef struct hiISP_CMOS_DPC_S
{
    HI_U16    au16Strength[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16    au16BlendRatio[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_DPC_S;
```

Hi3516EV200/Hi3516EV300 定义：

```
typedef struct hiISP_CMOS_DPC_S
{
    HI_U16    au16Strength[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16    au16BlendRatio[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8      u8DpcPos;
} ISP_CMOS_DPC_S;
```

【成员】

成员名称	描述
au16Strength	DPC 动态坏点处理强度，取值越大，DPC 处理强度越强，可以校正的动态坏点越多。



成员名称	描述
au16BlendRatio	坏点校正过程所需的融合比率。
u8DpcPos	DPC 模块在 ISP pipeline 的位置。 0: 在 DRC 模块后; 1: 在 WDR 模块前。默认值 0。 线性模式取值范围: [0, 1], 非线性模式只支持配置 0。 仅 Hi3516EV200/Hi3516EV300 支持。

【注意事项】

- 仅 Hi3516EV200/Hi3516EV300 支持 DPC 模块位置可调。非线性模式下, DPC 模块位置不可调, 只能配置在 DRC 后。
- DPC 模块配置在 DRC 模块后, BNR 去噪强度会影响坏点检测。BNR 去噪参数较强时, 坏点已被抑制, 可能导致部分坏点不能被 DPC 模块检出, 建议调整 DPC 模块位置。BNR 去噪强度较小时, DPC 模块在 DRC 后去坏点能力更强。
- DPC 强度大于 242 时, 坏点检测的阈值很小, 低照度下噪声误检为坏点的概率较高, 会导致图像出现黑点。DPC 模块位置在 WDR 前, 黑点现象会更明显。不建议设置 DPC 强度大于 242。

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_LSC_CABLI_TABLE_S

【说明】

定义 LSC 的校正参数。

【定义】

```
typedef struct hiISP_LSC_CABLI_TABLE_S
{
    HI_U16 au16R_Gain[HI_ISP_LSC_GRID_POINTS];
    HI_U16 au16Gr_Gain[HI_ISP_LSC_GRID_POINTS];
    HI_U16 au16Gb_Gain[HI_ISP_LSC_GRID_POINTS];
    HI_U16 au16B_Gain[HI_ISP_LSC_GRID_POINTS];
} ISP_LSC_CABLI_TABLE_S;
```



【成员】

成员名称	描述
au16R_Gain	R 通道校正增益表。取值范围：[0, 1023]
au16Gr_Gain	Gr 通道校正增益表。取值范围：[0, 1023]
au16Gb_Gain	Gb 通道校正增益表。取值范围：[0, 1023]
au16B_Gain	B 通道校正增益表。取值范围：[0, 1023]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_LSC_S](#)

ISP_BNR_LSC_CALIB_TABLE_S

【说明】

定义 Bayer NR LSC 的标定 LUT。

【定义】

```
typedef struct hiISP_BNR_LSC_CALIB_TABLE_S
{
    HI_U16 au16R_Gain[HI_ISP_RLSC_POINTS];
    HI_U16 au16Gr_Gain[HI_ISP_RLSC_POINTS];
    HI_U16 au16Gb_Gain[HI_ISP_RLSC_POINTS];
    HI_U16 au16B_Gain[HI_ISP_RLSC_POINTS];
}ISP_BNR_LSC_CALIB_TABLE_S;
```

【成员】

成员名称	描述
au16R_Gain	R 通道增益表。
au16Gr_Gain	Gr 通道增益表。
au16Gb_Gain	Gb 通道增益表。



成员名称	描述
au16B_Gain	B 通道增益表。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_LSC_S](#)

ISP_CMOS_LSC_S

【说明】

定义 LSC 参数。

【定义】

Hi3559AV100/Hi3519AV100 定义：

```
typedef struct hiISP_CMOS_LSC_S
{
    HI_U8 u8MeshScale;
    ISP\_LSC\_CABLI\_TABLE\_S astLscCalibTable[HI\_ISP\_MLSC\_GAIN\_LUT\_NUM];
} ISP_CMOS_LSC_S;
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 定义：

```
typedef struct hiISP_CMOS_LSC_S
{
    HI_U16 u16MeshStrength;
    HI_U8 u8MeshScale;
    ISP\_LSC\_CABLI\_TABLE\_S astLscCalibTable[HI\_ISP\_MLSC\_GAIN\_LUT\_NUM];
    ISP\_BNR\_LSC\_CALIB\_TABLE\_S stBNRLscCalibTable;
} ISP_CMOS_LSC_S;
```

【成员】



成员名称	描述
u16MeshStrength	用来全局纠正 Mesh Shading 标定后的强度。 取值范围[0, 65535]。 仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持
u8MeshScale	增益表精度控制参数。取值范围：[0, 7]
astLscCalibTable	校正表格，参见 ISP_LSC_CABLI_TABLE_S 。
stBNRLscCalibTable	Bayer NR 模块中所用的 LSC 表格，参见 ISP_BNR_LSC_CALIB_TABLE_S 。 仅 Hi3516CV500/Hi3559V200 支持

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_RLSC_CABLI_TABLE_S

【说明】

定义 RLSC 的校正参数。

【定义】

```
typedef struct hiISP_RLSC_CABLI_TABLE_S
{
    HI_U16 u16WBRGain;
    HI_U16 u16WBBGain;
    HI_U16 au16R_Gain[HI\_ISP\_RLSC\_POINTS];
    HI_U16 au16Gr_Gain[HI\_ISP\_RLSC\_POINTS];
    HI_U16 au16Gb_Gain[HI\_ISP\_RLSC\_POINTS];
    HI_U16 au16B_Gain[HI\_ISP\_RLSC\_POINTS];
} ISP_RLSC_CABLI_TABLE_S;
```

【成员】



成员名称	描述
u16WBRGain	白平衡的 R 通道增益，作为校正前的输入。
u16WBBGain	白平衡的 B 通道增益，作为校正前的输入。
au16R_Gain	R 通道校正增益表，校正输出。
au16Gr_Gain	Gr 通道校正增益表，校正输出。
au16Gb_Gain	Gb 通道校正增益表，校正输出。
au16B_Gain	B 通道校正增益表，校正输出。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_ACS_Y_SHADING_LUT_S

【说明】

定义 Auto Color Shading 亮度分量上的校正强度表，也就是 Gr/Gb 分量的校正强度，用标定工具生成。

【定义】

```
typedef struct hiISP_ACS_Y_SHADING_LUT_S
{
    HI_U16 u16GParamHighCT[HI_ISP_LSC_GRID_POINTS];
    HI_U16 u16GParamLowCt[HI_ISP_LSC_GRID_POINTS];
} ISP_ACS_Y_SHADING_LUT_S;
```

【成员】

成员名称	描述
u16GParamHighCT	Gr/Gb 分量上的校正强度表，校正强度较大
u16GParamLowCt	Gr/Gb 分量上的校正强度表，校正强度较小



【注意事项】

算法根据场景从 u16GparamHighCT 和 u16GParamLowCT 两张表中进行插值。

【相关数据类型及接口】

[ISP_CMOS_ACS_S](#)

ISP_ACS_COLOR_SHADING_LUT_S

【说明】

定义 Auto Color Shading 颜色分量上的 Lut 表，用标定工具生成，算法会根据 R/B 分量上的 Lut 表，动态生成适合当前场景的 Lut 表。

【定义】

```
typedef struct hiISP_ACS_COLOR_SHADING_LUT_S
{
    HI_FLOAT f32AvgRgMap[HI_ISP_LSC_GRID_POINTS];
    HI_FLOAT f32AvgBgMap[HI_ISP_LSC_GRID_POINTS];
    HI_FLOAT f32ProfRgMap[HI_ISP_LSC_GRID_POINTS];
    HI_FLOAT f32ProfBgMap[HI_ISP_LSC_GRID_POINTS];
} ISP_ACS_COLOR_SHADING_LUT_S;
```

【成员】

成员名称	描述
f32AvgRgMap	R 分量上的 Color Shading 表
f32AvgBgMap	B 分量上的 Color Shading 表
f32ProfRgMap	R 分量上的 Color Shading 表
f32ProfBgMap	B 分量上的 Color Shading 表

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_ACS_S](#)



ISP_ACS_CALIB_PARAM_S

【说明】

定义 Auto Color Shading 的标定参数，用标定工具生成。

【定义】

```
typedef struct hiISP_ACS_CALIB_PARAM_S
{
    HI_S16 u16LightIndex[HI_ISP_ACS_LIGHT_NUM * HI_ISP_ACS_CHN_NUM];
    HI_FLOAT f32ModelARMin;
    HI_FLOAT f32ModelARStep;
    HI_FLOAT f32ModelABMin;
    HI_FLOAT f32ModelABStep;
    HI_S16 u16LightTypeGHigh;
    HI_S16 u16LightTypeGLow;
} ISP_ACS_CALIB_PARAM_S;
```



说明

宏定义如下：

```
#define HI_ISP_ACS_LIGHT_NUM    32
#define HI_ISP_ACS_CHN_NUM     2
```

【成员】

成员名称	描述
u16LightIndex	标定的光源在算法模型当中的光源坐标
f32ModelARMin	标定得到的算法模型参数
f32ModelARStep	标定得到的算法模型参数
f32ModelABMin	标定得到的算法模型参数
f32ModelABStep	标定得到的算法模型参数
u16LightTypeGHigh	对于 u16GParamHighCT 表对应的光源坐标
u16LightTypeGLow	对于 u16GParamLowCT 表对应的光源坐标

【注意事项】

无



【相关数据类型及接口】

ISP_CMOS_ACS_S

ISP_CMOS_ACS_S

【说明】

定义 Auto Color Shading 的 CMOS 参数。

【定义】

```
typedef struct hiISP_CMOS_ACS_S
{
    HI_BOOL                bAcsEn;
    HI_U16                 u16AcsShadingStr;
    ISP_ACS_CALIB_PARAM_S  stAcsCalibParam;
    ISP_ACS_Y_SHADING_LUT_S stAcsYShadingLut;
    ISP_ACS_COLOR_SHADING_LUT_S stAcsColorShadingLut;
} ISP_CMOS_ACS_S;
```

【成员】

成员名称	描述
bAcsEn	ACS 模块使能开关
u16AcsShadingStr	Y Shading 上的校正强度
stAcsCalibParam	参考 ISP_ACS_CALIB_PARAM_S
stAcsYShadingLut	参考 ISP_ACS_Y_SHADING_LUT_S
stAcsColorShadingLut	参考 ISP_ACS_COLOR_SHADING_LUT_S

【注意事项】

- 增益表的默认配置与 [ISP_CMOS_ALG_KEY_U](#) 中的 bit1Acs 标志位有关，如果 bit1Acs=1，则采用 cmos_ex.h 中的配置值作为默认值；否则默认配置都为 0。
- ACS 模块的 otp 通过 [ISP_CMOS_LSC_S](#) 中的 astLscCalibTable 接口实现，其中 astLscCalibTable[0]配置为 golden sample 在 D50 下标定的表，astLscCalibTable[1]配置为当前镜头模组在 D50 下标定的表，可以解决镜头的一致性问题的，模组与 golden 之间的差异越小，校正的效果越好。也可以在 ISP 启动



后配置接口 [HI_MPI_ISP_SetMeshShadingGainLutAttr](#) 中的 `astLscGainLut[0]`和 `astLscGainLut[1]`，用法与上面描述一致。

【相关数据类型及接口】

- [ISP_CMOS_DEFAULT_S](#)
- [ISP_ACS_Y_SHADING_LUT_S](#)
- [ISP_ACS_COLOR_SHADING_LUT_S](#)
- [ISP_ACS_CALIB_PARAM_S](#)

ISP_CMOS_RLSC_S

【说明】

定义 RLSC 的校正参数。

【定义】

```
typedef struct hiISP_CMOS_RLSC_S
{
    HI_U8    u8Scale;
    HI_U16   u16CenterRX;
    HI_U16   u16CenterRY;
    HI_U16   u16CenterGrX;
    HI_U16   u16CenterGrY;
    HI_U16   u16CenterGbX;
    HI_U16   u16CenterGbY;
    HI_U16   u16CenterBX;
    HI_U16   u16CenterBY;
    HI_U16   u16OffCenterR;
    HI_U16   u16OffCenterGr;
    HI_U16   u16OffCenterGb;
    HI_U16   u16OffCenterB;
    ISP\_RLSC\_CABLI\_TABLE\_S stLscCalibTable[HI\_ISP\_RLSC\_GAIN\_LUT\_NUM];
} ISP_CMOS_RLSC_S;
```

【成员】

成员名称	描述
u8Scale	增益表精度控制参数。取值范围：[0, 13]



成员名称	描述
u16CenterRX	R 分量光学中心位置指示：画面宽方向
u16CenterRY	R 分量光学中心位置指示：画面高方向
u16CenterGrX	Gr 分量光学中心位置指示：画面宽方向
u16CenterGrY	Gr 分量光学中心位置指示：画面高方向
u16CenterGbX	Gb 分量光学中心位置指示：画面宽方向
u16CenterGbY	Gb 分量光学中心位置指示：画面高方向
u16CenterBX	B 分量光学中心位置指示：画面宽方向
u16CenterBY	B 分量光学中心位置指示：画面高方向
u16OffCenterR	R 分量与成像的校正范围半径相关的量
u16OffCenterGr	Gr 分量与成像的校正范围半径相关的量
u16OffCenterGb	Gb 分量与成像的校正范围半径相关的量
u16OffCenterB	B 分量与成像的校正范围半径相关的量
stLscCalibTable	镜头畸变校正表

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_GE_S

【说明】

定义 GE 参数。

【定义】

```
typedef struct hiISP_CMOS_GE_S
{
    HI_BOOL bEnable;
```



```
HI_U8  u8Slope;
    HI_U8  u8SensiSlope;
    HI_U16 u16SensiThr;
    HI_U16 au16Threshold[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16Strength[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16NpOffset[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_GE_S;
```

【成员】

成员名称	描述
bEnable	使能 Crosstalk 功能。
u8Slope	设置 Crosstalk 斜率值。值越大，表示在门限值 u16Threshold 之上整体处理强度衰减得越慢。 默认值 9。 Hi3516EV200/Hi3516EV300 取值范围：[0,16]，其它芯片取值范围：[0, 14]。
u8SensiSlope	设置 Crosstalk 敏感度值。默认值 9。值越大，表示在门限值 u16SensiThr 之上，边沿上的处理强度衰减得越慢。 Hi3516EV200/Hi3516EV300 取值范围：[0,16]，其它芯片取值范围：[0, 14]。
u16SensiThr	设置 Crosstalk 敏感度门限值。值越大，表示在更强的边沿上也会有绿平衡处理。默认值 300。 Hi3516EV200/Hi3516EV300 取值范围：[0, 0xFFFF]，其它芯片取值范围：[0, 0x3FFF]。
au16Threshold	设置 Crosstalk 门限值。值越大，表示整体处理的强度越大。 默认值 300。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。 Hi3516EV200/Hi3516EV300 取值范围：[0, 0xFFFF]，其它芯片取值范围：[0, 0x3FFF]。
au16Strength	设置 Crosstal 强度值。值越大，处理强度越大。取值范围：[0, 0x100]。默认值 128。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。



成员名称	描述
au16NpOffset	设置 Noise Profile 值。值越大，表示抗噪能力越大。默认值 1024。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。 Hi3516EV200/Hi3516EV300 取值范围：[0x2000, 0xFFFF]，其它芯片取值范围：[0x200, 0x3FFF]。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_ANTIFALSECOLOR_S

【说明】

定义 AntifalseColor 参数。

【定义】

```
typedef struct hiISP_CMOS_ANTIFALSECOLOR_S
{
    HI_BOOL bEnable;
    HI_U8    au8AntiFalseColorThreshold[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8AntiFalseColorStrength[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_ANTIFALSECOLOR_S;
```

【成员】

成员名称	描述
bEnable	使能该模块功能。
au8AntiFalseColorThreshold	设置去伪彩门限。取值范围：[0, 0x20]
au8AntiFalseColorStrength	设置去伪彩强度值。取值范围：[0, 0x1F]

【注意事项】



无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_BAYERNR_S

【说明】

定义 BayerNR 结构体。

【定义】

```
#define HI_ISP_BAYERNR_LUT_LENGTH    (33)
```

Hi3559AV100/Hi3519AV100 定义:

```
typedef struct hiISP_CMOS_BAYERNR_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bLowPowerEnable;
    HI_BOOL  bBnrMonoSensorEn;
    HI_BOOL  bNrLscEnable;
    HI_U8     u8NrLscRatio;
    HI_U8     au8LutFineStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8     au8ChromaStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8     au8WDRFrameStr[WDR_MAX_FRAME_NUM];
    HI_U16    au16CoarseStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16    au16LutCoringWgt[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16    au16LutCoringRatio[HI_ISP_BAYERNR_LUT_LENGTH];
} ISP_CMOS_BAYERNR_S;
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 定义:

```
typedef struct hiISP_CMOS_BAYERNR_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bBnrMonoSensorEn;
    HI_BOOL  bNrLscEnable;
    HI_U8     u8BnrLscMaxGain;
    HI_U16    u16BnrLscCmpStrength;
    HI_U8     au8LutFineStr[ISP_AUTO_ISO_STRENGTH_NUM];
```



```

HI_U8    au8ChromaStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
HI_U8    au8WDRFrameStr[WDR_MAX_FRAME_NUM];
HI_U8    au8FusionFrameStr[WDR_MAX_FRAME_NUM];
HI_U16   au16CoarseStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
HI_U16   au16LutCoringWgt[ISP_AUTO_ISO_STRENGTH_NUM];
HI_U16   au16LutCoringRatio[HI_ISP_BAYERNR_LUT_LENGTH];
} ISP_CMOS_BAYERNR_S;

```

【成员】

成员名称	描述
bEnable	使能去噪模块。
bLowPowerEnable	低功耗模式开关。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
bBnrMonoSensorEn	黑白 sensor 模式使能。 Hi3559AV100 和 Hi3519AV100 不支持。
bNrLscEnable	NR 参考 LSC 增益功能使能。
u8NrLscRatio	NR 参考 LSC 增益比例。取值范围[0x0, 0xFF]。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
u8BnrLscMaxGain	参考 LSC 增益最大倍数。 值越大，参考 LSC 去噪效果越明显。 取值范围[0x0, 0xBF]。 仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
u16BnrLscCmpStrength	参考 LSC 增益强度。取值范围[0x0, 0x100]。 值越大，参考 LSC 去噪效果越明显。 仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
au8LutFineStr	亮噪去除强度，值越大，亮噪去除强度越大。 范围：[0x0,0x80]



成员名称	描述
au8ChromaStr	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。取值范围：[0x0, 0x3] Hi3516EV200/Hi3516EV300 不支持。
au8WDRFrameStr	Hi3559AV100： auWDRFrameStr[0]：WDR 模式下合成后超短帧区域去噪强度。 auWDRFrameStr[1]：WDR 模式下合成后短帧区域去噪强度。 auWDRFrameStr[2]：WDR 模式下合成后中帧区域去噪强度。 auWDRFrameStr[3]：WDR 模式下合成后长帧区域去噪强度。 Hi3519AV100： auWDRFrameStr[0]：WDR 模式下合成后短帧区域去噪强度。 auWDRFrameStr[1]：WDR 模式下合成后长帧区域去噪强度。 auWDRFrameStr[2]：Hi3519AV100 下不支持。 auWDRFrameStr[3]：Hi3519AV100 下不支持。 Hi3516CV500/Hi3559V200/Hi3516EV200： auWDRFrameStr[0]：WDR 模式下合成后短帧区域去噪强度。 auWDRFrameStr[1]：WDR 模式下合成后长帧区域去噪强度。 auWDRFrameStr[2]：WDR 模式下合成后运动区域去噪强度。 auWDRFrameStr[3]：不支持。值越大，去噪强度越大，值越小，去噪强度越弱。取值范围：[0x0, 0x50]。模式下长帧和短帧分别对应的全局亮度去噪强度。值越大，去噪强度越大，值越小，去噪强度越弱。取值范围：[0x0, 0x50]
au8FusionFrameS	Au8FusionFrameStr[0]：Fusion 模式下融合后短帧区域去噪



成员名称	描述
tr	强度。 Au8FusionFrameStr[1]: Fusion 模式下融合后长帧区域去噪强度。 取值范围: [0x0, 0x50]。 仅 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持。
au16CoarseStr	控制亮度噪声整体去噪强度, 四通道强度独立可调, 值越大, 亮噪去噪强度越大。取值范围: [0x0, 0x360]
au16LutCoringWgt	随机噪声保留强度, 值越大, 保留的随机噪声越多。 取值范围: [0x0,0xC80]
au16LutCoringRatio	噪声保留强度, 根据亮度在 au16CoringRatio[33]数组里查找对应的噪声保留强度, 值越大, 亮度噪声保留越多, 值越小, 亮度噪声保留越少。取值范围: [0x0, 0x3FF]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_DETAIL_S

【说明】

定义 Detail Enhance 结构体。

【定义】

```
typedef struct hiISP_CMOS_DETAIL_S
{
    ISP\_DE\_ATTR\_S stAttr;
} ISP_CMOS_DETAIL_S;
```

【成员】



成员名称	描述
stAttr	Detail Enhance 属性。具体参见 ISP_DE_ATTR_S 。仅 Hi3516CV500/Hi3559V200 支持

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_SPLIT_POINT_S

【说明】

定义 split 参数。

【定义】

```
typedef struct hiISP_CMOS_SPLIT_POINT_S
{
    HI_U8   u8X;
    HI_U16  u16Y;
}ISP_CMOS_SPLIT_POINT_S;
```

【成员】

成员名称	描述
u8X	X 坐标。取值范围：[0x0, 0x80]
u16Y	Y 坐标。取值范围：[0x0, 0x8000]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)



ISP_CMOS_SPLIT_S

【说明】

定义 SPLIT 参数。

【定义】

```
typedef struct hiISP_CMOS_SPLIT_S
{
    HI_BOOL bEnable;
    HI_U8   u8InputWidthSel;
    HI_U8   u8ModeIn;
    HI_U8   u8ModeOut;
    HI_U32  u32BitDepthOut;
    HI_U8   u8SplitPointNum;
    ISP\_CMOS\_SPLIT\_POINT\_S astSplitPoint[ISP_SPLIT_POINT_NUM_MAX];
}ISP_CMOS_SPLIT_S;
```



说明

宏定义如下：

```
#define SPLIT_NODE_NUM 129
#define ISP_SPLIT_POINT_NUM_MAX (SPLIT_NODE_NUM - 1)
```

【成员】

成员名称	描述
bEnable	使能 Split 模块。
u8InputWidthSel	输入数据位宽的选择，其中 0=12bit；1=14bit；2=16bit； 取值范围：[0, 3]
u8ModeIn	输入模式的选择，其中 0=linear； 1=2~3mux（不支持）； 2=16log； 3=sensor-built-in 模式 默认值为 0。取值范围：[0, 3]
u8ModeOut	输出模式的选择： 0：输出 1 路高比特数据（最高到 16bit）并跳过 WDR 模块进入



成员名称	描述
	isp; 1: 拆分 2 路 12 比特数据并进入 WDR 进行合成; 2: 拆分 3 路 12 比特数据并进入 WDR 进行合成; 默认为 1。取值范围: [0, 3]
u32BitDepthOut	进行拆分前的数据位宽。取值范围: [0xC, 0x14]
u8SplitPointNum	拐点坐标的数目。取值范围: [0x0, 0x80]
astSplitPoint	在 sensor-built-in 模式下解压的拐点 (包括横纵坐标)。

【注意事项】

在 sensor 手册中会给出 sensor 在 sensor-built-in 模式下内部压缩时使用的拐点配置, 当 u8Modeln 取值为 3 时, 需要将这几个拐点配置按照对应的原则进行转换, 然后配置到 astSplitPoint 中即可。

转换原则如下:

- astSplitPoint 的横坐标 u8X 需要根据 sensor 压缩曲线转换到[0,128]之间 (7bit), 例如 sensor 压缩输出的后有效数据位宽是 12bit, 则需要将 sensor 压缩曲线拐点的纵坐标右移 5bit 得到 astSplitPoint 的横坐标 u8X;
- astSplitPoint 的纵坐标 u16Y 需要根据 sensor 压缩曲线转换到[0,32768]之间 (15bit), 例如 sensor 合成有效数据未压缩之前有效位宽是 20bit, 则需要将 sensor 压缩曲线的拐点的横坐标右移 5bit, 得到 astSplitPoint 的纵坐标 u16Y; 如 sensor 合成后有效数据位宽是 16bit, 则需要将拐点的横坐标右移 1bit。
- 不必配置(0,0)的拐点坐标, 代码内部在进行插值时, 会自动将(0,0)作为第一个拐点坐标。

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_CA_S

【说明】

定义 CA 参数。

【定义】



Hi3559AV100/Hi3519AV100 定义:

```
typedef struct hiISP_CMOS_CA_S
{
    HI_BOOL    bEnable;
    HI_U32     au32YRatioLut[HI_ISP_CA_YRATIO_LUT_LENGTH];
    HI_S16     as16ISORatio[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_BOOL    bCpEnable;
    HI_U32     au32YRatioLUTY[HI_ISP_CA_YRATIO_LUT_LENGTH];
    HI_U32     au32YRatioLUTU[HI_ISP_CA_YRATIO_LUT_LENGTH];
    HI_U32     au32YRatioLUTV[HI_ISP_CA_YRATIO_LUT_LENGTH];
} ISP_CMOS_CA_S;
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 定义:

```
typedef struct hiISP_CMOS_CA_S
{
    HI_BOOL    bEnable;
    HI_U16     au16YRatioLut[HI_ISP_CA_YRATIO_LUT_LENGTH];
    HI_S16     as16ISORatio[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CMOS_CA_S;
```

【成员】

成员名称	描述
bEnable	使能 CA 模块
bCpEnable	CP 模式使能, Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
au32YRatioLut/au16YRatioLut	CA 模式, 根据亮度 Y 查找 UV 增益的查找表。 取值范围: [0, 2047]
as16ISORatio	CA 模式, 根据 ISO 值查找 UV 增益的查找表。 取值范围: [0, 2047]
au32YRatioLUTY	CP 模式, 根据亮度 Y 查找色板中的 Y 值。取值范围: [0, 255] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
au32YRatioLUTU	CP 模式, 根据亮度 Y 查找色板中的 U 值。取值范围: [0, 255]



成员名称	描述
	Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
au32YRatioLUTV	CP 模式，根据亮度 Y 查找色板中的 V 值。取值范围：[0, 255] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_EXPANDER_POINT_S

【说明】

定义 Expander 拐点坐标参数。

【定义】

```
typedef struct hiISP_CMOS_EXPANDER_POINT_S
{
    HI_U16 u16X;
    HI_U32 u32Y;
} ISP_CMOS_EXPANDER_POINT_S;
```

【成员】

成员名称	描述
U16X	X 坐标。 Hi3516CV500/Hi3559V200 取值范围：[0x0,0x100] Hi3516EV200/Hi3516EV300 取值范围：[0x0,0x80]
U32Y	Y 坐标。 Hi3516CV500/Hi3559V200 取值范围：[0x0, 0x100000] Hi3516EV200/Hi3516EV300 取值范围：[0x0,0x8000]



【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_EXPANDER_S

【说明】

定义 Expander 结构体。

【定义】

```
typedef struct hiISP_CMOS_EXPANDER_S
{
    HI_BOOL bEnable;
    HI_U8   u8BitDepthIn;
    HI_U8   u8BitDepthOut;
    HI_U16  u16ExpanderPointNum;
    ISP\_CMOS\_EXPANDER\_POINT\_S astExpanderPoint[ISP_EXPANDER_POINT_NUM_MAX];
} ISP_CMOS_EXPANDER_S;
```



说明

宏定义如下：

Hi3516CV500/Hi3559V200 宏定义如下：

```
#define EXPANDER_NODE_NUM 257
```

Hi3516EV200/Hi3516EV300 宏定义如下：

```
#define EXPANDER_NODE_NUM 129
```

以上宏值代入代码中：

```
#define ISP_EXPANDER_POINT_NUM_MAX (EXPANDER_NODE_NUM - 1)
```

【成员】

成员名称	描述
bEnable	使能 Expander 模块
u8BitDepthIn	输入数据位宽。取值范围：[0xC,0x14]
u8BitDepthOut	输出数据位宽。取值范围：[0xC,0x14]
u16ExpanderPointNum	拐点坐标的数目。



成员名称	描述
	Hi3516CV500/Hi3559V200 取值范围: [0x0, 0x100] Hi3516EV200/Hi3516EV300 取值范围: [0x0,0x80]
astExpanderPoint	解压的拐点 (包括横纵坐标)

【注意事项】

在 sensor 手册中会给出 sensor 在 sensor-built-in 模式下内部压缩时使用的拐点配置, 需要将这几个拐点配置按照对应的原则进行转换, 然后配置到 astExpanderPoint 中即可。

Hi3516CV500 转换原则如下:

- astExpanderPoint 的横坐标 u16X 需要根据 sensor 压缩曲线转换到[0,256]之间 (8bit), 例如 sensor 压缩输出的后有效数据位宽是 12bit, 则需要将 sensor 压缩曲线拐点的纵坐标右移 4bit 得到 astExpanderPoint 的横坐标 u16X;
- astExpanderPoint 的纵坐标 u32Y 需要根据 sensor 压缩曲线转换到[0,1048576]之间 (20bit), 例如 sensor 合成有效数据未压缩之前有效位宽是 16bit, 则需要将 sensor 压缩曲线的拐点的横坐标左移 4bit, 得到 astExpanderPoint 的纵坐标 u32Y。
- 不必配置(0,0)的拐点坐标, 代码内部在进行插值时, 会自动将(0,0)作为第一个拐点坐标。

Hi3516EV200 转换原则如下:

- astExpanderPoint 的横坐标 u16X 需要根据 sensor 压缩曲线转换到[0,128]之间 (7bit), 例如 sensor 压缩输出的后有效数据位宽是 12bit, 则需要将 sensor 压缩曲线拐点的纵坐标右移 5bit 得到 astExpanderPoint 的横坐标 u16X;
- astExpanderPoint 的纵坐标 u32Y 需要根据 sensor 压缩曲线转换到[0,32768]之间 (15bit), 例如 sensor 合成有效数据未压缩之前有效位宽是 20bit, 则需要将 sensor 压缩曲线的拐点的横坐标右移 5bit, 得到 astExpanderPoint 的纵坐标 u32Y; 如 sensor 合成后有效数据位宽是 16bit, 则需要将拐点的横坐标右移 1bit。
- 不必配置(0,0)的拐点坐标, 代码内部在进行插值时, 会自动将(0,0)作为第一个拐点坐标。

【相关数据类型及接口】



ISP_CMOS_DEFAULT_S

ISP_CMOS_NOISE_CALIBRATION_S

【说明】

定义 NOISE 校正参数。

【定义】

```
typedef struct hiISP_CMOS_NOISE_CALIBRATION_S
{
    HI_U16    u16CalibrationLutNum;
    HI_FLOAT
afCalibrationCoef[BAYER_CALIBTAION_MAX_NUM][BAYER_CALIBRATION_PARA_NUM];
} ISP_CMOS_NOISE_CALIBRATION_S;
```



说明

宏定义如下：

```
#define BAYER_CALIBTAION_MAX_NUM    50
#define BAYER_CALIBRATION_PARA_NUM  3
```

【成员】

成员名称	描述
u16CalibrationLutNum	标定序列数目，取值范围为[0, BAYER_CALIBTAION_MAX_NUM-1]。
afCalibrationCoef[BAYER_CALIBTAION_MAX_NUM][BAYER_CALIBRATION_PARA_NUM]	噪声标定表

【注意事项】

无

【相关数据类型及接口】

ISP_CMOS_DEFAULT_S

ISP_CMOS_SENSOR_MAX_RESOLUTION_S

【说明】



定义 sensor 最大分辨率结构体。

【定义】

```
typedef struct hiISP_CMOS_SENSOR_MAX_RESOLUTION_S
{
    HI_U32  u32MaxWidth;
    HI_U32  u32MaxHeight;
} ISP_CMOS_SENSOR_MAX_RESOLUTION_S;
```

【成员】

成员名称	描述
u32MaxWidth	最大宽度。
u32MaxHeight	最大高度。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_CLUT_S

【说明】

定义 CLUT 结构体。

【定义】

```
typedef struct hiISP_CMOS_CLUT_S
{
    HI_BOOL bEnable      ;
    HI_U32  u32GainR      ;
    HI_U32  u32GainG      ;
    HI_U32  u32GainB      ;
    ISP\_CLUT\_LUT\_S stClutLut;
} ISP_CMOS_CLUT_S;
```

【成员】



成员名称	描述
bEnable	CLut 使能。
u32GainR	R 增益。
u32GainG	G 增益。
u32GainB	B 增益。
stClutLut	CLut 查找表。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_LOGLUT_S

【说明】

定义 LOGLUT 结构体。

【定义】

```
typedef struct hiISP_CMOS_LOGLUT_S
{
    HI_BOOL  bEnable;
    HI_U8    u8BitDepthOut;
    HI_U32    au32LogLUT[LOG_LUT_SIZE];
} ISP_CMOS_LOGLUT_S;
```



说明

宏定义如下：

```
#define LOG_LUT_SIZE    (1025)
```

【成员】

成员名称	描述
bEnable	LogLUT 使能



u8BitDepthOut	解压后 bit 位宽。取值范围：[10, 20]
au32LogLUT	logLUT 的增益表。 - Hi3559AV100 取值范围：[0, 0xFFFF] - Hi3519AV100 取值范围：[0, 0x3FF]

【注意事项】

当前 bEnable 和 u8BitDepthOut 仅 Hi3559AV100 支持。

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_PRELOGLUT_S

【说明】

定义 PRELOGLUT 的参数。

【定义】

```
#define PRE_LOG_LUT_SIZE    (1025)
typedef struct hiISP_CMOS_PRELOGLUT_S
{
    HI_U32    au32PreLogLUT[PRE_LOG_LUT_SIZE];
} ISP_CMOS_PRELOGLUT_S;
```

【成员】

成员名称	描述
au32PreLogLUT	PreLogLUT 的增益表。取值范围：[0, 0x3FF]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)



ISP_CMOS_SENSOR_MODE_S

【说明】

定义 sensor 模式寄存器。

【定义】

```
typedef struct hiISP_CMOS_SENSOR_MODE_S
{
    HI_U32  u32SensorID;
    HI_U8   u8SensorMode;
    HI_BOOL bValidDngRawFormat;
    DNG\_RAW\_FORMAT\_S stDngRawFormat;
} ISP_CMOS_SENSOR_MODE_S;
```

【成员】

成员名称	描述
u32SensorID	Sensor id 号。
u8SensorMode	Sensor 自定义工作模式，不同分辨率与帧率对应不同的工作模式。
bValidDngRawFormat	有效的 DNG raw 格式。
stDngRawFormat	DNG Raw 格式。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_WDR_S

【说明】

定义 WDR 结构体。

【定义】

```
#define WDR_SIGMA_GAIN_NUM 3
```



Hi3559AV100:

```
#define WDR_MAX_FRAME    4
typedef struct hiISP_CMOS_WDR_S
{
    HI_BOOL    bFusionMode;
    HI_BOOL    bMotionComp;
    HI_U16     u16ShortThr;
    HI_U16     u16LongThr;
    HI_BOOL    bShortExpoChk;
    HI_U16     u16ShortCheckThd;
    HI_BOOL    bMDRefFlicker;
    HI_U8      au8MdThrLowGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8      au8MdThrHigGain[ISP_AUTO_ISO_STRENGTH_NUM];
    ISP_BNR_MODE_E enBnrMode;
    HI_U16     au16FusionThr[WDR_MAX_FRAME];
    HI_U8      au8GsigmaGain[WDR_SIGMA_GAIN_NUM];
    HI_U8      au8RBsigmaGain[WDR_SIGMA_GAIN_NUM];
    HI_U8      u8MdtStillThd;
    HI_U8      u8MdtFullThd;
    HI_U8      u8MdtLongBlend;
    HI_U8      u8FullMdtSigWgt;
    HI_U8      au8NoiseFloor[NOISESET_ELENUM];
} ISP_CMOS_WDR_S;
```

Hi3519AV100:

```
#define WDR_MAX_FRAME    (2)
typedef struct hiISP_CMOS_WDR_S
{
    HI_BOOL    bFusionMode;
    HI_BOOL    bMotionComp;
    HI_U16     u16ShortThr;
    HI_U16     u16LongThr;
    HI_BOOL    bForceLong;
    HI_U16     u16ForceLongLowThr;
    HI_U16     u16ForceLongHigThr;
    HI_BOOL    bShortExpoChk;
    HI_U16     u16ShortCheckThd;
```



```
    HI_BOOL    bMDRefFlicker;
    HI_U8      au8MdThrLowGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8      au8MdThrHigGain[ISP_AUTO_ISO_STRENGTH_NUM];
    ISP_BNR_MODE_E  enBnrMode;
    HI_U16     au16FusionThr[WDR_MAX_FRAME];
    HI_U8      au8GsigmaGain[WDR_SIGMA_GAIN_NUM];
    HI_U8      au8RBsigmaGain[WDR_SIGMA_GAIN_NUM];
    HI_U8      u8MdtStillThd;
    HI_U8      u8MdtLongBlend;
    HI_U8      u8FullMdtSigGWgt;
    HI_U8      u8FullMdtSigRBWgt;
    HI_U8      au8NoiseFloor[NOISESET_ELENUM];
} ISP_CMOS_WDR_S;
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300:

```
#define WDR_MAX_FRAME    (2)
typedef struct hiISP_CMOS_WDR_S
{
    HI_BOOL    bFusionMode;
    HI_BOOL    bMotionComp;
    HI_U16     u16ShortThr;
    HI_U16     u16LongThr;
    HI_BOOL    bForceLong;
    HI_U16     u16ForceLongLowThr;
    HI_U16     u16ForceLongHigThr;
    HI_BOOL    bShortExpoChk;
    HI_U16     u16ShortCheckThd;
    HI_BOOL    bMDRefFlicker;
    HI_U8      au8MdThrLowGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8      au8MdThrHigGain[ISP_AUTO_ISO_STRENGTH_NUM];
    ISP_BNR_MODE_E  enBnrMode;
    HI_U16     au16FusionThr[WDR_MAX_FRAME];
    HI_U8      u8MdtStillThd;
    HI_U8      u8MdtLongBlend;
} ISP_CMOS_WDR_S;
```

【成员】



成员名称	描述
bFusionMode	Fusion 模式是否使能
bMotionComp	运动补偿是否打开
u16ShortThr	短曝光门限值，超过门限值的图像数据只选择短曝光数据。 取值范围：[0x0, 0xFFFF]
u16LongThr	长曝光门限值，低于门限值的图像数据只选择长曝光数据。 取值范围：[0x0, u16ShortThr]
bForceLong	强制输出长帧使能 Hi3559AV100 不支持。
u16ForceLongLowThr	强制输出长帧的低阈值，亮度低于该阈值的运动区域强制输出长帧，取值范围：[0, 0xFFFF] Hi3559AV100 不支持。
u16ForceLongHigThr	强制输出长帧的高阈值，亮度大于该阈值的运动区域不强制输出长帧，取值范围：[0, 0xFFFF] Hi3559AV100 不支持。
bShortExpoChk	控制运动检测是否检查短帧数据大小，当短帧数据接近 0 时，WDR 融合强制选长帧。 0：WDR 融合时不检查短帧数据大小； 1：WDR 融合时检查短帧数据大小。
u16ShortCheckThd	短帧检查阈值。当短帧亮度小于该阈值时，WDR 强制输出长帧。 取值范围：[0x0,0xFFFF]
bMDRefFlicker	控制运动判断是否参考 Flicker 检测结果。 0：做运动判断时不参考 Flicker 检测结果 1：做运动判断时参考 Flicker 检测结果
au8MdThrLowGain	手动模式下判断是否运动的低阈值系数。值越大，运动程度越小。取值范围：[0, au8MdThrHigGain]



成员名称	描述
au8MdThrHigGain	手动模式下判断是否运动的高阈值系数。值越大，运动程度越小。取值范围：[0, 0xFF]
enBnrMode	去噪功能的开关 0：BNR 关闭； 1：BNR 打开。
au16FusionThr	在做 Fusion 功能时，不同帧的阈值。取值范围：[0x0, 0x3FFF]
au8GsigmaGain[WDR_SIGMA_GAIN_NUM]	G 通道的三级降噪强度，值越大，强度越大，取值范围：[0, 0xFF] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
au8RbsigmaGain[WD R_SIGMA_GAIN_NUM]	RB 通道的三级降噪强度，值越大，强度越大，取值范围：[0, 0xFF] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
u8MdtStillThd	运动检测中判断为静止的阈值，当运动程度小于该阈值时，判断该像素点为静止。取值范围：[0, 0xFE]
u8MdtFullThd	运动检测中判断为全运动的阈值，当运动程度大于该阈值时，判断该像素点为全运动。取值范围：[0, 0xFE] Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
u8MdtLongBlend	运动区域叠加长帧的权重，当该值为 0xFE 时，运动区域全部选择长帧。取值范围：[0, 0xFF]
u8FullMdtSigGWgt	全运动区域 G 通道的降噪强度，值越大，降噪强度越大，取值范围：[0, 0x1F] Hi3559AV100/Hi3516CV500//Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
u8FullMdtSigRBWgt	全运动区域 RB 通道的降噪强度，值越大，降噪强度越大，取值范围：[0, 0x1F] Hi3559AV100/Hi3516CV500/Hi3559V200/Hi3516EV200



成员名称	描述
	/Hi3516EV300 不支持
u8FullMdtSigWgt	全运动区域的降噪强度，不分通道，值越大，降噪强度越大，取值范围：[0, 0x1F] Hi3519AV100/Hi3516CV500/Hi3559V200 不支持
au8NoiseFloor[NOISE SET_ELENUM]	噪声基底，值越大，降噪越强，取值范围：[0, 0xFF] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_DEHAZE_S

【说明】

定义 Dehaze 结构体。

【定义】

```
typedef struct HiSP_CMOS_DEHAZE_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bUserLutEnable;
    ISP\_OP\_TYPE\_E    enOpType;
    HI_U8    u8AutoStrength;
    HI_U8    u8ManualStrength;
} ISP_CMOS_DEHAZE_S;
```

【成员】

成员名称	描述
bEnable	Dehaze 模块使能



成员名称	描述
bUserLutEnable	用户自定义去雾曲线使能
enOpType	操作模式
u8AutoStrength	自动模式下去雾强度。取值范围：[0, 255]
u8ManualStrength	手动模式下去雾强度。取值范围：[0, 255]

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_RGBIR_S

【说明】

定义 RGBIR 模块参数

【定义】

```
typedef struct hiISP_CMOS_RGBIR_S
{
    HI_BOOL  bEnable;
    HI_S16  as16CvtMatrix[RGBIR_CVTMATRIX_NUMBER];
} ISP_CMOS_RGBIR_S;
```



说明

宏定义如下：

```
#define RGBIR_CVTMATRIX_NUMBER 12
```

【成员】

成员名称	描述
bEnable	RGBIR 使能开关。
as16CvtMatrix	RGBIR 转换矩阵参数配置。



【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_DNG_COLORPARAM_S

【说明】

定义 DNG 白平衡校正系数。

【定义】

```
typedef struct hiISP_CMOS_DNG_COLORPARAM_S
{
    ISP\_DNG\_WBGAIN\_S stWbGain1;
    ISP\_DNG\_WBGAIN\_S stWbGain2;
}ISP_CMOS_DNG_COLORPARAM_S;
```

【成员】

成员名称	描述
stWbGain1	DNG 白平衡校正系数 1。
stWbGain2	DNG 白平衡校正系数 2。

【注意事项】

无

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_WDR_SWITCH_ATTR_S

【说明】

定义 WDR 切换属性。

【定义】

```
typedef struct hiISP_CMOS_WDR_SWITCH_ATTR_S
```



```
{  
    HI_U32    au32ExpRatio[EXP_RATIO_NUM];  
}ISP_CMOS_WDR_SWITCH_ATTR_S;
```

【成员】

成员名称	描述
au32ExpRatio	曝光比期望值

【注意事项】

在 `cmos_get_esp_default` 函数里根据 WDR 模式给 `au32ExpRatio` 赋默认值，所赋值要与 `cmos_get_ae_default` 中 AE 初始化的曝光比保持一致性：

- 若 `pstAeSnsDft->u16ManRatioEnable` 为 TRUE，`au32ExpRatio` 与 `pstAeSnsDft->au32Ratio` 的值相同；
- 若 `pstAeSnsDft->u16ManRatioEnable` 为 FALSE，`au32ExpRatio` 为 0x40。

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_S](#)

ISP_CMOS_AWB_ATTR_S

【说明】

定义 AWB 模块参数。

【定义】

```
typedef struct hiISP_CMOS_AWB_ATTR_S {  
    ISP\_AWB\_GAIN\_SWITCH\_E enAwbGainSwitch;  
} ISP_CMOS_AWB_ATTR_S;
```

【成员】

成员名称	描述
enAwbGainSwitch	AWB 增益生效位置。取值范围：[0, 1]

【注意事项】



- 默认 AWB 增益在 ISP 生效。
- 实现 AWB 增益在 Sensor 生效，需要配置 enAwbGainSwitch 为 [ISP_AWB_GAIN_IN_SENSOR](#)，对回调函数 pfn_cmos_get_awb_gains 赋值，且将 au32SensorWBGain 参数的 AWB 增益值配置到对应的 Sensor 寄存器。
- 如果回调函数 pfn_cmos_get_awb_gains 未赋值，则 enAwbGainSwitch 参数设置无效，AWB 增益在 ISP 生效。

【相关数据类型及接口】

- [ISP_CMOS_DEFAULT_S](#)
- [ISP_SENSOR_EXP_FUNC_S](#)
- [ISP_SNS_STATE_S](#)

ISP_CMOS_ALG_KEY_U

【说明】

定义 ISP 的各算法是否采用 cmos 中的默认配置的标志位。

【定义】

Hi3559AV100 中定义：

```
typedef union hiISP_CMOS_ALG_KEY_U
{
    HI_U64 u64Key;
    struct
    {
        HI_U64 bit1Drc          : 1; /* [0] */
        HI_U64 bit1Demosaic     : 1; /* [1] */
        HI_U64 bit1PreGamma     : 1; /* [2] */
        HI_U64 bit1Gamma        : 1; /* [3] */
        HI_U64 bit1Sharpen      : 1; /* [4] */
        HI_U64 bit1EdgeMark     : 1; /* [5] */
        HI_U64 bit1Ldci         : 1; /* [6] */
        HI_U64 bit1Dpc          : 1; /* [7] */
        HI_U64 bit1Lsc          : 1; /* [8] */
        HI_U64 bit1RLsc         : 1; /* [9] */
        HI_U64 bit1Ge           : 1; /* [10] */
        HI_U64 bit1AntiFalseColor : 1; /* [11] */
        HI_U64 bit1BayerNr      : 1; /* [12] */
    };
};
```



```
HI_U64 bit1Split      : 1 ; /* [13] */
HI_U64 bit1Ca         : 1 ; /* [14] */
HI_U64 bit1Clut       : 1 ; /* [15] */
HI_U64 bit1LogLUT     : 1 ; /* [16] */
HI_U64 bit1PreLogLUT  : 1 ; /* [17] */
HI_U64 bit1Wdr        : 1 ; /* [18] */
HI_U64 bit1Awb        : 1 ; /* [19] */
HI_U64 bit44Rsv       : 44; /* [20:63] */
};
} ISP_CMOS_ALG_KEY_U;
```

Hi3519AV100 中定义:

```
typedef union hiISP_CMOS_ALG_KEY_U
{
    HI_U64 u64Key;
    struct
    {
        HI_U64 bit1Drc      : 1 ; /* [0] */
        HI_U64 bit1Demosaic : 1 ; /* [1] */
        HI_U64 bit1PreGamma  : 1 ; /* [2] */
        HI_U64 bit1Gamma     : 1 ; /* [3] */
        HI_U64 bit1Sharpen   : 1 ; /* [4] */
        HI_U64 bit1EdgeMark  : 1 ; /* [5] */
        HI_U64 bit1Hlc       : 1 ; /* [6] */
        HI_U64 bit1Ldci      : 1 ; /* [7] */
        HI_U64 bit1Dpc       : 1 ; /* [8] */
        HI_U64 bit1Lsc       : 1 ; /* [9] */
        HI_U64 bit1RLsc      : 1 ; /* [10] */
        HI_U64 bit1Ge        : 1 ; /* [11] */
        HI_U64 bit1AntiFalseColor : 1 ; /* [12] */
        HI_U64 bit1BayerNr   : 1 ; /* [13] */
        HI_U64 bit1Split     : 1 ; /* [14] */
        HI_U64 bit1Ca        : 1 ; /* [15] */
        HI_U64 bit1Clut      : 1 ; /* [16] */
        HI_U64 bit1LogLUT    : 1 ; /* [17] */
        HI_U64 bit1PreLogLUT : 1 ; /* [18] */
        HI_U64 bit1Wdr       : 1 ; /* [19] */
        HI_U64 bit1Awb       : 1 ; /* [20] */
    };
};
```



```
        HI_U64 bit43Rsv      : 43; /* [21:63] */
    };
} ISP_CMOS_ALG_KEY_U;
```

Hi3516CV500/Hi3559V200 中定义:

```
typedef union hiISP_CMOS_ALG_KEY_U
{
    HI_U64  u64Key;
    struct
    {
        HI_U64  bit1Drc      : 1; /* [0] */
        HI_U64  bit1Demosaic : 1; /* [1] */
        HI_U64  bit1PreGamma : 1; /* [2] */
        HI_U64  bit1Gamma    : 1; /* [3] */
        HI_U64  bit1Sharpen  : 1; /* [4] */
        HI_U64  bit1EdgeMark : 1; /* [5] */
        HI_U64  bit1Hlc      : 1; /* [6] */
        HI_U64  bit1Ldci     : 1; /* [7] */
        HI_U64  bit1Dpc      : 1; /* [8] */
        HI_U64  bit1Lsc      : 1; /* [9] */
        HI_U64  bit1Ge       : 1; /* [10] */
        HI_U64  bit1AntiFalseColor : 1; /* [11] */
        HI_U64  bit1BayerNr  : 1; /* [12] */
        HI_U64  bit1Detail   : 1; /* [13] */
        HI_U64  bit1Ca       : 1; /* [14] */
        HI_U64  bit1Expander : 1; /* [15] */
        HI_U64  bit1Clut     : 1; /* [16] */
        HI_U64  bit1Wdr      : 1; /* [17] */
        HI_U64  bit1Dehaze   : 1; /* [18] */
        HI_U64  bit1Lcac     : 1; /* [19] */
        HI_U64  bit1Acs      : 1; /* [20] */
        HI_U64  bit43Rsv     : 43; /* [21:63] */
    };
} ISP_CMOS_ALG_KEY_U;
```

Hi3516EV200/Hi3516EV300 中定义:

```
typedef union hiISP_CMOS_ALG_KEY_U
{
```




```

HI_U64 u64Key;
struct
{
    HI_U64 bit1Drc          : 1; /* [0]*/
    HI_U64 bit1Demosaic    : 1; /* [1]*/
    HI_U64 bit1PreGamma    : 1; /* [2]*/
    HI_U64 bit1Gamma       : 1; /* [3]*/
    HI_U64 bit1Sharpen     : 1; /* [4]*/
    HI_U64 bit1Ldci        : 1; /* [5]*/
    HI_U64 bit1Dpc         : 1; /* [6]*/
    HI_U64 bit1Lsc         : 1; /* [7]*/
    HI_U64 bit1Ge          : 1; /* [8] */
    HI_U64 bit1AntiFalseColor : 1; /* [9] */
    HI_U64 bit1BayerNr     : 1; /* [10] */
    HI_U64 bit1Ca          : 1; /* [11] */
    HI_U64 bit1Expander    : 1; /* [12] */
    HI_U64 bit1Wdr         : 1; /* [13] */
    HI_U64 bit1Dehaze      : 1; /* [14] */
    HI_U64 bit1Rgbir       : 1; /* [15] */
    HI_U64 bit1Lcac        : 1; /* [16] */
    HI_U64 bit47Rsv        : 47; /* [17:63] */
};
} ISP_CMOS_ALG_KEY_U;

```

【成员】

成员名称	描述
bit1Drc	drc 模块是否采用 cmos 默认配置的标志位
bit1Demosaic	demosaic 模块是否采用 cmos 默认配置的标志位
bit1PreGamma	pregamma 模块是否采用 cmos 默认配置的标志位
bit1Gamma	gamma 模块是否采用 cmos 默认配置的标志位
bit1Sharpen	sharpen 模块是否采用 cmos 默认配置的标志位
bit1EdgeMark	edge mark 模块是否采用 cmos 默认配置的标志位 Hi3516EV200/Hi3516EV300 不支持



成员名称	描述
bit1Hlc	hlc 模块是否采用 cmos 默认配置的标志位。Hi3559AV100/Hi3516EV200/Hi3516EV300 不支持
bit1Ldci	ldci 模块是否采用 cmos 默认配置的标志位
bit1Dpc	dpc 模块是否采用 cmos 默认配置的标志位
bit1Lsc	lsc 模块是否采用 cmos 默认配置的标志位
bit1RLsc	rlsc 模块是否采用 cmos 默认配置的标志位 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
bit1Acs	acs 模块是否采用 cmos 默认配置的标志位 Hi3559AV100/Hi3519AV100/Hi3516EV200/Hi3516EV300 不支持
bit1Ge	ge 模块是否采用 cmos 默认配置的标志位
bit1AntiFalseColor	anti false color 模块是否采用 cmos 默认配置的标志位
bit1BayerNr	bayer nr 模块是否采用 cmos 默认配置的标志位
bit1Split	split 模块是否采用 cmos 默认配置的标志位 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
bit1Detail	detail enhance 模块是否采用 cmos 默认配置的标志位 仅 Hi3516CV500/Hi3559V200 支持
bit1Ca	ca 模块是否采用 cmos 默认配置的标志位
bit1Expander	expander 模块是否采用 cmos 默认配置的标志位 Hi3559AV100/Hi3519AV100 不支持
bit1Clut	clut 模块是否采用 cmos 默认配置的标志位, Hi3516EV200/Hi3516EV300 不支持。
bit1LogLUT	loglut 模块是否采用 cmos 默认配置的标志位 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支



成员名称	描述
	持
bit1PreLogLUT	pre loglut 模块是否采用 cmos 默认配置的标志位 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
bit1Wdr	wdr 模块是否采用 cmos 默认配置的标志位
bit1Dehaze	dehaze 模块是否采用 cmos 默认配置的标志位 Hi3559AV100/Hi3519AV100 不支持
bit1Rgbir	RGBIR 模块是否采用 cmos 默认配置标志位。仅 Hi3516EV200/Hi3516EV300 支持。
bit1Lcac	LCAC 模块是否采用 cmos 默认配置标志位。仅 Hi3516EV200/Hi3516EV300 支持
bit1Awb	AWB 模块是否采用 cmos 默认配置标志位。仅 Hi3559AV100/Hi3519AV10 支持。

【注意事项】

如果 ISP 的某个算法模块要使用 cmos 中的配置，要将对应的标志位置为 1，否则采用的是算法内部的默认配置

【相关数据类型及接口】

[ISP_CMOS_DEFAULT_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】

Hi3559A V100 定义：

```
typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP\_CMOS\_ALG\_KEY\_U          unKey;
```



```

const ISP_CMOS_DRC_S          *pstDrc;
const ISP_CMOS_DEMOSAIC_S     *pstDmosaic;
const ISP_CMOS_PREGAMMA_S     *pstPreGamma;
const ISP_CMOS_GAMMA_S        *pstGamma;
const ISP_CMOS_SHARPEN_S      *pstSharpen;
const ISP_CMOS_EDGEMARK_S     *pstEdgeMark;
const ISP_CMOS_LDCI_S         *pstLdci;
const ISP_CMOS_DPC_S          *pstDpc;
const ISP_CMOS_LSC_S          *pstLsc;
const ISP_CMOS_RLSC_S         *pstRLsc;
const ISP_CMOS_GE_S           *pstGe;
const ISP_CMOS_ANTIFALSECOLOR_S *pstAntiFalseColor;
const ISP_CMOS_BAYERNR_S      *pstBayerNr;
const ISP_CMOS_SPLIT_S        *pstSplit;
const ISP_CMOS_CA_S           *pstCa;
const ISP_CMOS_CLUT_S         *pstClut;
const ISP_CMOS_LOGLUT_S       *pstLogLUT;
const ISP_CMOS_PRELOGLUT_S    *pstPreLogLUT;
const ISP_CMOS_WDR_S          *pstWdr;
ISP_CMOS_NOISE_CALIBRATION_S  stNoiseCalibration;
ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
ISP_CMOS_SENSOR_MODE_S        stSensorMode;
ISP_CMOS_DNG_COLORPARAM_S     stDngColorParam;
ISP_CMOS_WDR_SWITCH_ATTR_S    stWdrSwitchAttr;
ISP_CMOS_AWB_ATTR_S           stAwbAttr;
} ISP_CMOS_DEFAULT_S;

```

Hi3519A V100 定义:

```

typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_ALG_KEY_U      unKey;
    const ISP_CMOS_DRC_S    *pstDrc;
    const ISP_CMOS_DEMOSAIC_S *pstDmosaic;
    const ISP_CMOS_PREGAMMA_S *pstPreGamma;
    const ISP_CMOS_GAMMA_S   *pstGamma;
    const ISP_CMOS_SHARPEN_S *pstSharpen;
    const ISP_CMOS_EDGEMARK_S *pstEdgeMark;
    const ISP_CMOS_HLC_S     *pstHlc;

```



```

const ISP_CMOS_LDCI_S          *pstLdci;
const ISP_CMOS_DPC_S           *pstDpc;
const ISP_CMOS_LSC_S           *pstLsc;
const ISP_CMOS_RLSC_S          *pstRLsc;
const ISP_CMOS_GE_S            *pstGe;
const ISP_CMOS_ANTIFALSECOLOR_S *pstAntiFalseColor;
const ISP_CMOS_BAYERNR_S       *pstBayerNr;
const ISP_CMOS_SPLIT_S         *pstSplit;
const ISP_CMOS_CA_S            *pstCa;
const ISP_CMOS_CLUT_S          *pstClut;
const ISP_CMOS_LOGLUT_S        *pstLogLUT;
const ISP_CMOS_PRELOGLUT_S     *pstPreLogLUT;
const ISP_CMOS_WDR_S           *pstWdr;
ISP_CMOS_NOISE_CALIBRATION_S   stNoiseCalibration;
ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
ISP_CMOS_SENSOR_MODE_S        stSensorMode;
ISP_CMOS_DNG_COLORPARAM_S     stDngColorParam;
ISP_CMOS_WDR_SWITCH_ATTR_S    stWdrSwitchAttr;
ISP_CMOS_AWB_ATTR_S           stAwbAttr;
} ISP_CMOS_DEFAULT_S;

```

Hi3516CV500/Hi3559V200 定义

```

typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_ALG_KEY_U          unKey;
    const ISP_CMOS_DRC_S        *pstDrc;
    const ISP_CMOS_DEMOSAIC_S   *pstDmosaic;
    const ISP_CMOS_PREGAMMA_S   *pstPreGamma;
    const ISP_CMOS_GAMMA_S      *pstGamma;
    const ISP_CMOS_SHARPEN_S    *pstSharpen;
    const ISP_CMOS_EDGEMARK_S   *pstEdgeMark;
    const ISP_CMOS_HLC_S        *pstHlc;
    const ISP_CMOS_LDCI_S       *pstLdci;
    const ISP_CMOS_DPC_S        *pstDpc;
    const ISP_CMOS_LSC_S        *pstLsc;
    const ISP_CMOS_GE_S         *pstGe;
    const ISP_CMOS_ANTIFALSECOLOR_S *pstAntiFalseColor;
    const ISP_CMOS_BAYERNR_S    *pstBayerNr;

```



```
const ISP_CMOS_DETAIL_S      *pstDetail;
const ISP_CMOS_CA_S          *pstCa;
const ISP_CMOS_EXPANDER_S    *pstExpander;
const ISP_CMOS_CLUT_S        *pstClut;
const ISP_CMOS_WDR_S          *pstWdr;
const ISP_CMOS_DEHAZE_S      *pstDehaze;
const ISP_CMOS_LCAC_S         *pstLcac;
const ISP_CMOS_ACS_S          *pstAcs;
    ISP_CMOS_NOISE_CALIBRATION_S    stNoiseCalibration;
    ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
    ISP_CMOS_SENSOR_MODE_S          stSensorMode;
ISP_CMOS_DNG_COLORPARAM_S      stDngColorParam;
ISP_CMOS_WDR_SWITCH_ATTR_S     stWdrSwitchAttr;
} ISP_CMOS_DEFAULT_S;
```

Hi3516EV200/Hi3516EV300 定义:

```
typedef struct hiISP_CMOS_DEFAULT_S
{
    ISP_CMOS_ALG_KEY_U      unKey;
    const ISP_CMOS_DRC_S     *pstDrc;
    const ISP_CMOS_DEMOSAIC_S *pstDmosaic;
    const ISP_CMOS_PREGAMMA_S *pstPreGamma;
    const ISP_CMOS_GAMMA_S    *pstGamma;
    const ISP_CMOS_SHARPEN_S  *pstSharpen;
    const ISP_CMOS_LDCI_S     *pstLdci;
    const ISP_CMOS_DPC_S      *pstDpc;
    const ISP_CMOS_LSC_S      *pstLsc;
    const ISP_CMOS_GE_S       *pstGe;
    const ISP_CMOS_ANTIFALSECOLOR_S *pstAntiFalseColor;
    const ISP_CMOS_BAYERNR_S  *pstBayerNr;
    const ISP_CMOS_CA_S       *pstCa;
    const ISP_CMOS_EXPANDER_S *pstExpander;
    const ISP_CMOS_WDR_S      *pstWdr;
    const ISP_CMOS_DEHAZE_S   *pstDehaze;
const ISP_CMOS_RGBIR_S      *pstRgbir;
const ISP_CMOS_LCAC_S       *pstLcac;
    ISP_CMOS_NOISE_CALIBRATION_S    stNoiseCalibration;
    ISP_CMOS_SENSOR_MAX_RESOLUTION_S stSensorMaxResolution;
```



```

    ISP_CMOS_SENSOR_MODE_S      stSensorMode;
    ISP_CMOS_DNG_COLORPARAM_S    stDngColorParam;
    ISP_CMOS_WDR_SWITCH_ATTR_S   stWdrSwitchAttr;
} ISP_CMOS_DEFAULT_S;

```

【成员】

成员名称	描述
unKey	标识各算法是否采用 cmos 中默认配置的 key
*pstDrc	DRC 结构体指针
*pstDemosaic	Demosaic 结构体指针
*pstPreGamma	PreGamma 结构体指针
*pstGamma	Gamma 结构体指针
*pstSharpen	Sharpen 结构体指针
*pstEdgeMark	EdgeMark 结构体指针, Hi3516EV200/Hi3516EV300 不支持
*pstHlc	HLC 结构体指针。 Hi3559AV100/Hi3516EV200/Hi3516EV300 不支持
*pstLdci	LDCI 结构体指针
*pstDpc	DPC 结构体指针
*pstLsc	LSC 结构体指针
*pstRLsc	Radial shading 的结构体指针, Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
*pstGe	GE 模块结构体指针
*pstAntiFalseColor	AntiFalse 结构体指针
*pstBayerNr	BayerNR 结构体指针
*pstDetail	Detail Enhance 结构体指针。仅 Hi3516CV500/Hi3559V200 支持



成员名称	描述
*pstSplit	Split 结构体指针, Hi3516EV200/Hi3516EV300 不支持
*pstCa	CA 模块结构体指针
*pstExpander	Expander 结构体指针。Hi3559AV100/Hi3519AV100 不支持
*pstClut	Clut 结构体指针, Hi3516EV200/Hi3516EV300 不支持
*pstLogLUT	logLUT 结构体指针。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
*pstPreLogLUT	PreLogLUT 结构体指针。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
*pstWdr	WDR 模式结构体指针
*pstDehaze	Dehaze 结构体指针。Hi3559AV100/Hi3519AV100 不支持
*pstRgbir	RGBIR 模式结构体指针, 仅 Hi3516EV200/Hi3516EV300 支持。
*pstLcac	LCAC 结构体指针, 仅 Hi3516EV200/Hi3516EV300 支持。
*pstAcs	ACS 结构体指针, 仅 Hi3516CV500 支持。
stNoiseCalibration	Noise 校正结构体。
stSensorMaxResolution	Sensor 最大宽高结构体。
stSensorMode	Sensor 模式结构体。
stDngColorParam	DNG 结构体。
stWdrSwitchAttr	WDR 切换属性。

【注意事项】



- stSensorMaxResolution、stSensorMode、stDngColorParam 的默认值在 cmos.c 中，其他 ISP 基础算法的默认值均在 cmos_ex.h 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。
- ISP 的各算法模块如果要采用 cmos 的默认配置，要在 cmos_get_isp_default 函数中将对应的标志位置 1，并给该算法模块的 cmos 结构体指针赋值。

【相关数据类型及接口】

ISP_SENSOR_EXP_FUNC_S

ISP_CMOS_BLACK_LEVEL_S

【说明】

定义 sensor 的黑电平结构体。

【定义】

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

【成员】

成员名称	描述
bUpdate	Sensor 的黑电平是否会动态根据增益改变，取值范围：[0,1]。若设置为 HI_TRUE，ISP 内部始终使用 cmos.c 内联动的黑电平配置；此时若需要手动改变 ISP 的黑电平，则设置 HI_MPI_ISP_SetBlackLevelAttr 为手动模式。
au16BlackLevel	Sensor 的黑电平数组，取值范围：[0, 4095]

【注意事项】

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

【相关数据类型及接口】

ISP_SENSOR_EXP_FUNC_S



ISP_CMOS_LCAC_MANUAL_S

【说明】

定义 sensor 的 Local CAC 算法的手动算法结构体。

【定义】

```
typedef struct hiISP_CMOS_LCAC_MANUAL_S
{
    HI_U8 u8DePurpleCrStr;
    HI_U8 u8DePurpleCbStr;
} ISP_CMOS_LCAC_MANUAL_S;
```

【成员】

成员名称	描述
u8DePurpleCrStr	表示手动模式下，对 Cr 分量的去紫边强度大小，取值范围[0, 8]。
u8DePurpleCbStr	表示手动模式下，对 Cb 分量的去紫边强度大小，取值范围[0, 8]。

【注意事项】

无

【相关数据类型及接口】

- [ISP_CMOS_LCAC_S](#)
- [ISP_CMOS_LCAC_AUTO_S](#)

ISP_CMOS_LCAC_AUTO_S

【说明】

定义 sensor 的 Local CAC 算法的自动算法结构体。

【定义】

```
typedef struct hiISP_CMOS_LCAC_AUTO_S
{
    HI_U8 au8DePurpleCrStr[ISP_EXP_RATIO_STRENGTH_NUM];
    HI_U8 au8DePurpleCbStr[ISP_EXP_RATIO_STRENGTH_NUM];
}
```



```
} ISP_CMOS_LCAC_AUTO_S;
```

【成员】

成员名称	描述
au8DePurpleCrStr	表示自动模式下，在各个档位上，对 Cr 分量的去紫边强度大小，取值范围[0, 8]。
au8DePurpleCbStr	表示自动模式下，在各个档位上，对 Cb 分量的去紫边强度大小，取值范围[0, 8]。

【注意事项】

这里的分档不是和 ISO 联动，而是和当前模式下 sensor 的曝光比联动。线性模式下固定为第一档，sensor built-in 模式下固定为最后一档。

【相关数据类型及接口】

- [ISP_CMOS_LCAC_MANUAL_S](#)
- [ISP_CMOS_LCAC_S](#)

ISP_CMOS_LCAC_S

【说明】

定义 sensor 的 Local CAC 算法结构体。

【定义】

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 定义：

```
typedef struct hiISP_CMOS_LCAC_S
{
    HI_BOOL          bEnable;
    ISP\_OP\_TYPE\_E    enOpMode;
    HI_U16            u16VarThr;
    HI_U16            au16RLumaThd[ISP_LCAC_DET_NUM];
    HI_U16            au16GLumaThd[ISP_LCAC_DET_NUM];
    HI_U16            au16BLumaThd[ISP_LCAC_DET_NUM];
    HI_U16            au16YLumaThd[ISP_LCAC_DET_NUM];
    HI_S16            as16CbCrRatio[ISP_LCAC_DET_NUM];
    HI_U16            u16PurpleDetRange;
```



```
ISP_CMOS_LCAC_MANUAL_S stManual;  
ISP_CMOS_LCAC_AUTO_S   stAuto;  
} ISP_CMOS_LCAC_S;
```

Hi3516EV200 定义:

```
typedef struct hiISP_CMOS_LCAC_S  
{  
    HI_BOOL          bEnable;  
    HI_U16            u16PurpleDetRange;  
    HI_U16            u16VarThr;  
    HI_U16            au16RLumaThd[ISP_LCAC_DET_NUM];  
    HI_U16            au16GLumaThd[ISP_LCAC_DET_NUM];  
    HI_U16            au16BLumaThd[ISP_LCAC_DET_NUM];  
    HI_U16            au16YLumaThd[ISP_LCAC_DET_NUM];  
    HI_S16            as16CbCrRatio[ISP_LCAC_DET_NUM];  
    ISP_OP_TYPE_E     enOpMode;  
    ISP_CMOS_LCAC_MANUAL_S stManual;  
    ISP_CMOS_LCAC_AUTO_S   stAuto;  
} ISP_CMOS_LCAC_S;
```



说明

宏定义如下:

```
#define ISP_LCAC_DET_NUM    (3)
```

【成员】

成员名称	描述
bEnable	表示 Local CAC 模块是否使能。
u16PurpleDetRange	表示本模块的紫色区域检测范围设定, 取值范围: [0, 410]
VarThr	用来表示本模块的边缘检测强度, 改值设定越大, 则越不容易被判断成边缘, 取值范围: [0, 4095]
au16RLumaThd	高亮区域检测 R 分量阈值, 取值范围: [0, 4095]
au16GLumaThd	高亮区域检测 G 分量阈值, 取值范围: [0, 4095]
au16BLumaThd	高亮区域检测 B 分量阈值, 取值范围[0, 4095]
au16YLumaThd	高亮区域检测亮度阈值, 取值范围: [0, 4095]



成员名称	描述
as16CbCrRatio	紫边检测, 区域颜色下限, 取值范围: [-2048, 2047]
enOpMode	本模块参数执行模式: 手动或自动模式
stManual	Local CAC 手动参数, 详情请参考 ISP_CMOS_LCAC_MANUAL_S
stAuto	Local CAC 自动参数, 详情请参考 ISP_CMOS_LCAC_AUTO_S

【注意事项】

无

【相关数据类型及接口】

- [ISP_CMOS_LCAC_MANUAL_S](#)
- [ISP_CMOS_LCAC_AUTO_S](#)

ISP_SNS_MODE_E

【说明】

定义 sensor 的曝光配置方式。

【定义】

```
typedef enum hiISP_SNS_MODE_E {  
    SNS_EXP_MODE = 0,  
    SNS_SHR_MODE = 1,  
    SNS_MODE_BUTT  
}ISP_SNS_MODE_E;
```

【成员】

成员名称	描述
SNS_EXP_MODE	直接配置曝光时间的方式。例如: 当前曝光时间为 Exp, 即直接将 Exp 配置给 sensor 寄存器。
SNS_SHR_MODE	间接配置曝光时间的方式。例如: 当前曝光时间为 Exp, 即将 (曝光最大行数-Exp) 配置给 sensor 寄存器。



【注意事项】

无

【相关数据类型及接口】

无

ISP_MCF_DATA_S

【说明】

定义多光谱融合 sensor 控制寄存器。

【定义】

```
typedef struct hiISP_MCF_DATA_S {  
    HI_U32  u32SnsVmaxAddr[ISP_MAX_SNS_EXP_ADDR_NUM];  
    HI_U32  u32SnsExpAddr[ISP_MAX_SNS_EXP_ADDR_NUM];  
    HI_U32  u32SnsReadoutTime;  
    ISP_SNS_MODE_E enSnsExpMode;  
} ISP_MCF_DATA_S;
```

【成员】

成员名称	描述
u32SnsVmaxAddr	Sensor Vmax 寄存器地址，即曝光时间最大值寄存器地址。
u32SnsExpAddr	Sensor 曝光寄存器地址。
u32SnsReadoutTime	Sensor 读出数据的时间，根据 sensor 不同而不同，一般取 sensor 输出图像高度即可。 注：有些 sensor 内部做了时钟倍频，此时需要将图像高度乘以倍频系数。
enSnsExpMode	Sensor 曝光配置方式。



说明

宏定义如下：

```
#define ISP_MAX_SNS_EXP_ADDR_NUM    (3)
```

【注意事项】



- 以上结构体成员相关配置需要参考对应 sensor datasheet 手册。
- 以上结构体成员仅在 Hi3516DV200/Hi3516EV300 芯片上支持。

【相关数据类型及接口】

无

ISP_SNS_TYPE_E

【说明】

定义 sensor 通信接口方式。

【定义】

```
typedef enum hiISP_SNS_TYPE_E {  
    ISP_SNS_I2C_TYPE = 0,  
    ISP_SNS_SSP_TYPE,  
    ISP_SNS_TYPE_BUTT,  
} ISP_SNS_TYPE_E;
```

【成员】

成员名称	描述
ISP_SNS_I2C_TYPE	使用 i2c 方式通信。
ISP_SNS_SSP_TYPE	使用 spi 方式通信。

【注意事项】

无

【相关数据类型及接口】

无

ISP_SNS_COMMBUS_U

【说明】

定义 sensor 绑定信息。

【定义】

```
typedef union hiISP_SNS_COMMBUS_U {
```



```
HI_S8 s8I2cDev;
struct {
    HI_S8 bit4SspDev : 4;
    HI_S8 bit4SspCs : 4;
} s8SspDev;
} ISP_SNS_COMMBUS_U;
```

【成员】

成员名称	描述
s8I2cDev	Sensor 绑定的 I2C 设备号。
s8SspDev	Sensor 绑定 SPI 设备号结构体。

【注意事项】

无

【相关数据类型及接口】

无

ISP_SNS_REGS_INFO_S

【说明】

定义 sensor 的寄存器信息。

【定义】

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




```
struct
{
    HI_BOOL bUpdate;
    HI_U8  u8DelayFrmNum;
    HI_U32 u32SlaveVsTime; /* time of vsync. Unit: inck clock cycle */
    HI_U32  u32SlaveBindDev;
} stSlvSync;
ISP_MCF_DATA_S stMcfData;
HI_BOOL bConfig;
} ISP_SNS_REGS_INFO_S;
```



说明

宏定义如下：

```
#define ISP_MAX_SNS_REGS    (32)
```

【成员】

成员名称	子成员名称	描述	
enSnsType	ISP_SNS_I2C_TYPE	Sensor 与 ISP 使用 I2C 接口通信。	
	ISP_SNS_SSP_TYPE	Sensor 与 ISP 使用 SSP 接口通信。	
u32RegNum	-	曝光结果写到 sensor 时需要配置的寄存器个数，不支持动态修改。	
u8Cfg2ValidDelayMax	-	所有 Sensor 寄存器从配置到生效延迟的帧数的最大值，单位为帧，用于保证 sensor 寄存器和 ISP 寄存器的同步。一般情况下，cmos sensor 的曝光时间寄存器的延迟最大，为 1~2 帧，因此配置一般为 1 或 2。	
unComBus	s8I2cDev	Sensor 绑定的 I2C 设备号	
	s8SspDev	Sensor 绑定 SPI 设备号结构体	
		bit4SspDev	Sensor 绑定的 SPI 设备号
		bit4SspCs	Sensor 绑定的 SPI 片选信号



成员名称	子成员名称	描述
astI2cData	bUpdate	HI_TRUE: 数据会配置 sensor 寄存器; HI_FALSE: 数据不会配置 sensor 寄存器。
	u8DelayFrmNum	Sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u8IntPos	Sensor 寄存器的配置生效的位置。 • 设置为 0x0 时表示寄存器在超短帧起始中断配置生效, 设置为 0x1 时表示寄存器在超短帧结束中断配置生效。 • 设置为 0x10 时表示寄存器在短帧起始中断配置生效, 设置为 0x11 时表示寄存器在短帧结束中断配置生效。 • 设置为 0x20 时表示寄存器在中帧起始中断配置生效, 设置为 0x21 时表示寄存器在中帧结束中断配置生效。 • 设置为 0x30 时表示寄存器在长帧起始中断配置生效, 设置为 0x31 时表示寄存器在长帧结束中断配置生效。
	u8DevAddr	Sensor 设备地址。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。
astSspData	bUpdate	HI_TRUE: 数据会配置 sensor 寄存



成员名称	子成员名称	描述
		器； HI_FALSE：数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u8IntPos	Sensor 寄存器的配置生效的位置。 • 设置为 0x0 时表示寄存器在帧起始中断配置生效，设置为 1 时表示寄存器在 AF 中断配置生效。 • 设置为 0x10 时表示寄存器在短帧起始中断配置生效，设置为 0x11 时表示寄存器在短帧结束中断配置生效。 • 设置为 0x20 时表示寄存器在中帧起始中断配置生效，设置为 0x21 时表示寄存器在中帧结束中断配置生效。 • 设置为 0x30 时表示寄存器在长帧起始中断配置生效，设置为 0x31 时表示寄存器在长帧结束中断配置生效。
	u32DevAddr	Sensor 设备地址。
	u32DevAddrByteNum	Sensor 设备地址位宽。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。
stSlvSync	bUpdate	HI_TRUE：数据会配置 sensor 寄存器；



成员名称	子成员名称	描述
		HI_FALSE: 数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u32SlaveVsTime	XVS 信号周期, 单位: sensor 输入时钟周期。
	u32SlaveBindDev	Slave 设备号与 ViPipe 绑定关系。
stMcfData	详情请参考 ISP_MCF_DATA_S	Sensor 与 MCF 相关的结构体
bConfig	--	Sensor 寄存器数据配置完成标志。 • HI_TRUE: 完成配置。 • HI_FALSE: 还未配置。

【注意事项】

无

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

ALG_LIB_S

【说明】

定义 AE/AWB 算法库结构体。

【定义】

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

【成员】



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

【注意事项】

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

【相关数据类型及接口】

无

ISP_BIND_ATTR_S

【说明】

定义 ISP 库与 Sensor、3A 库之间绑定关系的结构体。

【定义】

```
typedef struct hiISP_BIND_ATTR_S
{
    SENSOR_ID    SensorId;
    ALG_LIB_S    stAeLib;
    ALG_LIB_S    stAfLib;
    ALG_LIB_S    stAwbLib;
} ISP_BIND_ATTR_S;
```

【成员】

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



【注意事项】

无

【相关数据类型及接口】

无

ISP_CTRL_PROC_WRITE_S

【说明】

定义 ISP 的 PROC 信息。

【定义】

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

【成员】

成员名称	描述
pcProcBuff	ISP 传给当前算法的 Proc 信息 Buffer 指针。
u32BuffLen	ISP 传给当前算法的 Proc 信息 Buffer 当前剩余字节数。Buffer 总大小为 8K 字节。
u32WriteLen	当前算法传给 ISP 的 Proc 信息字节数。

【注意事项】

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

【相关数据类型及接口】

无



ISP_CTRL_CMD_E

【说明】

定义 ISP 对 3A 的控制命令。

【定义】

```
typedef enum hiISP_CTRL_CMD_E
{
    ISP_WDR_MODE_SET = 8000,
    ISP_PROC_WRITE,
    ISP_AE_FPS_BASE_SET,
    ISP_AE_BLC_SET,
    ISP_AE_RC_SET,
    ISP_AE_BAYER_FORMAT_SET,
    ISP_AWB_ISO_SET,
    ISP_CHANGE_IMAGE_MODE_SET,
    ISP_UPDATE_INFO_GET,
    ISP_FRAMEINFO_GET,
    ISP_ATTACHINFO_GET,
    ISP_COLORGAMUTINFO_GET,
    ISP_AWB_INTTIME_SET,
    ISP_BAS_MODE_SET,
    ISP_PROTRIGGER_SET,
    ISP_AWB_PIRIS_SET,
    ISP_AWB_SNAP_MODE_SET,
    ISP_AWB_ZONE_ROW_SET,
    ISP_AWB_ZONE_COL_SET,
    ISP_AWB_ZONE_BIN_SET,
    ISP_CTRL_CMD_BUTT,
} ISP_CTRL_CMD_E;
```

【成员】

成员名称	描述
ISP_WDR_MODE_SET	设置 WDR 模式，将 ISP 控制单元的 WDR 模式配置到算法模块，此命令对应的参数数据类型是 WDR_MODE_E 。



成员名称	描述
ISP_PROC_WRITE	设置写 PROC 信息，将算法模块的 PROC 信息配置到 ISP 控制单元，此命令对应的参数数据类型是 ISP_CTRL_PROC_WRITE_S 。
ISP_AE_FPS_BASE_SET	设置帧率，将 ISP 控制单元的帧率信息配置到 AE 算法模块，此命令对应的参数与 ISP_PUB_ATTR_S 里面的 f32FrameRate 一样。
ISP_AE_BLC_SET	设置黑电平，将黑电平信息配置到 AE 算法模块。
ISP_AE_RC_SET	设置 RC 模块使能，将 Radial Crop 模块是否使能信息传入 AE 算法模块。
ISP_AE_BAYER_FORMAT_SET	设置 Bayer 图像数据格式，将 Bayer 图像数据格式信息传入 AE 算法模块。
ISP_AWB_ISO_SET	设置 ISO 值，将 AE 当前的 ISO 值配置到 AWB 模块，用于自动调整饱和度，此命令对应的参数与 ISP_AE_RESULT_S 里面的 u32Iso 一样。
ISP_CHANGE_IMAGE_MODE_SET	设置图像分辨率切换标识，将 ISP 控制单元的图像分辨率标识配置到算法模块，此命令对应的参数数据类型为 HI_U8，参数值为 0 表示图像分辨率未切换，其他值表示图像分辨率已切换。
ISP_UPDATE_INFO_GET	AE，AWB 状态信息更新，将算法获取运行过程中的状态信息，包括 AE，AWB 的状态信息。
ISP_FRAMEINFO_GET	获取 ISP 帧信息，包括 ISO、去噪强度，与编码模块配合使用。
ISP_ATTACHINFO_GET	获取 ISP 帧额外信息，包括 ISO、各个模块的算法参数等。
ISP_COLORGAMUTINFO_GET	获取通道色域属性。
ISP_AWB_INTTIME_SET	设置曝光量值，将 AE 当前的曝光量值配置到



成员名称	描述
	AWB 模块，用于室内外检测，此命令对应的参数与 ISP_AE_RESULT_S 里面的 u32IntTimeUs 一样。
ISP_BAS_MODE_SET	设置 BAS 模式。
ISP_PROTRIGGER_SET	专业拍照触发信号设置，当用户触发一次专业拍照时，ISP 将此信号设置给 AE，AE 启动专业曝光控制。注：该版本不支持。
ISP_AWB_PIRIS_SET	设置 piris 运行的增益信息，用于获取当前 piris 实际的状态信息。
ISP_AWB_SNAP_MODE_SET	设置是否为抓拍模式，将 ISP 当前是否为抓拍模式的信息配置到 AWB 模块。
ISP_AWB_ZONE_ROW_SET	设置 AWB 分块统计信息的行数。拼接、Crop 等处理导致 AWB 统计信息的行数不固定，需要将行数配置到 AWB 模块。
ISP_AWB_ZONE_COL_SET	设置 AWB 分块统计信息的列数。拼接、Crop 等处理导致 AWB 统计信息的列数不固定，需要将列数配置到 AWB 模块。
ISP_AWB_ZONE_BIN_SET	设置 AWB 分块统计信息的亮度分组数。 Hi3559AV100, Hi3519AV100 支持亮度分组的 AWB 统计，需要将亮度分组数配置到 AWB 模块。

【注意事项】

无

【相关数据类型及接口】

无

ISP_STITCH_ATTR_S

【说明】



定义 ISP 拼接结构体。

【定义】

```
typedef struct hiISP_STITCH_ATTR_S
{
    HI_BOOL bStitchEnable;
    HI_BOOL bMainPipe;
    HI_U8   u8StitchPipeNum;
    HI_S8   as8StitchBindId[VI_MAX_PIPE_NUM];
} ISP_STITCH_ATTR_S;
```



说明

宏定义如下：

Hi3559AV100 中 VI_MAX_PIPE_NUM 宏定义为：

```
#define VI_MAX_PIPE_NUM (8)
```

Hi3519AV100 中 VI_MAX_PIPE_NUM 宏定义为：

```
#define VI_MAX_PIPE_NUM (6)
```

Hi3516CV500/Hi3559V200 中 VI_MAX_PIPE_NUM 宏定义为：

```
#define VI_MAX_PIPE_NUM (4)
```

Hi3516EV200/Hi3516EV300 中 VI_MAX_PIPE_NUM 宏定义为：

```
#define VI_MAX_PIPE_NUM (4)
```

【成员】

成员名称	描述
bStitchEnable	拼接使能
bMainPipe	是否为主 Pipe
u8StitchPipeNum	拼接总路数
as8StitchBindId	拼接绑定的 pipe 的号

【注意事项】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【相关数据类型及接口】

无。



ISP_AE_REGISTER_S

【说明】

定义 AE 注册结构体。

【定义】

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

【成员】

成员名称	描述
stAeExpFunc	AE 注册的回调函数结构体。

【注意事项】

封装的目的是为了扩展。

【相关数据类型及接口】

无

ISP_AE_EXP_FUNC_S

【说明】

定义 AE 回调函数结构体。

【定义】

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



```
} ISP_AE_EXP_FUNC_S;
```

【成员】

成员名称	描述
pfn_ae_init	初始化 AE 的回调函数指针。
pfn_ae_run	运行 AE 的回调函数指针。
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 命令详参 [ISP_CTRL_CMD_E](#)。

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

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

【相关数据类型及接口】

[ISP_AE_REGISTER_S](#)

ISP_AE_PARAM_S

【说明】

定义 ISP 提供给 AE 的初始化参数结构体。

【定义】

```
typedef struct hiISP_AE_PARAM_S  
{  
    SENSOR_ID SensorId;  
    HI_U8 u8WDRMode;  
    HI_U8 u8HDRMode;  
    HI_U16 u16BlackLevel;  
    HI_FLOAT f32Fps;  
    ISP\_BAYER\_FORMAT\_E enBayer;  
    ISP\_STITCH\_ATTR\_S stStitchAttr;  
    HI_S32 s32Rsv;  
} ISP_AE_PARAM_S;
```

【成员】



成员名称	描述
SensorId	向 ISP 注册的 sensor 的 id，用以检查向 ISP 注册的 sensor 和向 AE 注册的 sensor 是否一致。
u8WDRMode	宽动态模式，ISP 向 AE 提供宽动态模式信息。
u8HDRMode	HDR 模式，ISP 向 AE 提供 HDR 模式信息。 注：仅 Hi3559AV100 支持。
u16BlackLevel	黑电平值，12bit 精度，ISP 向 AE 提供黑电平信息。
f32Fps	帧率，ISP 向 AE 提供帧率信息。
enBayer	Sensor Bayer Pattern，包括 RGGB、GRBG、GBRG、BGGR 四种格式。
stStitchAttr	拼接模式，ISP 向 AE 提供拼接模式信息。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持

【注意事项】

无

【相关数据类型及接口】

[ISP_AE_EXP_FUNC_S](#)

ISP_SMART_INFO_S

【说明】

定义 ISP 提供给 AE 的智能检测信息结构体。

【定义】

```
typedef struct hiISP_SMART_INFO_S
{
    ISP_PEOPLE_ROI_S  stPeopleROI[PEOPLE_CLASS_MAX];
    ISP_TUNNEL_ROI_S  stTunnelROI[TUNNEL_CLASS_MAX];
} ISP_SMART_INFO_S;
```



说明

宏定义如下：



```
#define PEOPLE_CLASS_MAX    (2)
#define TUNNEL_CLASS_MAX    (2)
```

【成员】

成员名称	子成员名称	描述
stPeopleROI	bEnable	人形、人脸检测模型是否使能。
	bAvailable	人形、人脸检测模型是否有可用的检测结果。
	u8Luma	人形、人脸检测模型检测结果的整体亮度。取值范围：[0x0, 0xFF]。
stTunnelROI	bEnable	隧道检测模型是否使能。
	bAvailable	隧道检测模型是否有可用的检测结果。
	u32TunnelAreaRatio	隧道检测模型检测结果占画面的面积比例。取值范围：[0, 10000]。
	u32TunnelExpPerf	隧道检测模型检测结果的曝光表现。取值范围：[0, 10000]。

【注意事项】

- stPeopleROI 支持人脸模型与人形模型检测结果，stPeopleROI 下标为 0 时为人脸检测结果，下标为 1 时为人形检测结果。
- 当仅有人形模型可用时，会根据人形检测结果估算出一个人脸亮度，此亮度可靠性会低于人脸检测结果。
- u8Luma 采用的是 YUV 图像中检测到的人脸或人形框内 Y 分量的亮度平均值。客户在使用自己的智能信息时，当存在多个人脸或人形检测结果，建议取多个检测结果的整体亮度平均值或感兴趣检测框的亮度值。框内人脸的亮度值可自行计算，也可通过调用 HI_MPI_VGS_AddLumaTaskArray 接口计算。
- stTunnelROI 支持隧道入口、出口模型检测结果，stTunnelROI 下标为 0 时为隧道入口检测结果，下标为 1 时为隧道出口检测结果。
- u32TunnelExpPerf 当前仅隧道出口使用，采用的是 YUV 图像中检测到的隧道出口框内过曝区域占整个检测框的比例。



- 向 AE 传递隧道检测结果信息，可实现行车使用场景中出入隧道时的极限收光。此功能仅支持在 WDR 模式下使用，同时需要设置曝光比为自动模式。在线性模式下不建议使用，否则效果不保证。

【相关数据类型及接口】

ISP_AE_INFO_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_SMART_INFO_S stSmartInfo;
    ISP_FE_AE_STAT_1_S* pstFEAeStat1;
    ISP_FE_AE_STAT_2_S* pstFEAeStat2;
    ISP_FE_AE_STAT_3_S* pstFEAeStat3;
    ISP_FE_AE_STITCH_STAT_3_S *pstFEAeStiStat;
    ISP_BE_AE_STAT_1_S* pstBEAeStat1;
    ISP_BE_AE_STAT_2_S* pstBEAeStat2;
    ISP_BE_AE_STAT_3_S* pstBEAeStat3;
    ISP_BE_AE_STITCH_STAT_3_S *pstBEAeStiStat;
} ISP_AE_INFO_S;
```

【成员】

成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
stSmartInfo	-	请参考 ISP_SMART_INFO_S 接口说明。
pstFEAeStat1	u32PixelCount	统计的像素点总个数。
	u32PixelWeight	带权重统计的像素点总个数。
	au32HistogramMem	1024 段直方图的统计信息数组，取值范



成员名称	子成员名称	描述
	Array	围为[0, 0xFFFFFFFF]。
pstFEAeStat2	u16GlobalAvgR	全局 R 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgGr	全局 Gr 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgGb	全局 Gb 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgB	全局 B 分量平均值, 取值范围为[0, 0xFFFF]。
pstFEAeStat3	au16ZoneAvg	分区间 R、Gr、Gb、B 分量平均值, 取值范围为[0, 0xFFFF]。
pstFEAeStiStat	au16ZoneAvg	拼接模式下拼接后分区间 R、Gr、Gb、B 分量平均值, 取值范围为[0, 0xFFFF]。 只有参与拼接的 Pipe 分区间均值有效, 其余 Pipe 的均值无效。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
pstBEAeStat1	u32PixelCount	统计的像素点总个数。
	u32PixelWeight	带权重统计的像素点总个数。
	au32HistogramMemArray	1024 段直方图的统计信息数组, 取值范围为[0, 0xFFFFFFFF]。
pstBEAeStat2	u16GlobalAvgR	全局 R 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgGr	全局 Gr 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgGb	全局 Gb 分量平均值, 取值范围为[0, 0xFFFF]。
	u16GlobalAvgB	全局 B 分量平均值, 取值范围为[0, 0xFFFF]。



成员名称	子成员名称	描述
pstBEAeStat3	au16ZoneAvg	分区间 R、Gr、Gb、B 分量平均值，取值范围为[0, 0xFFFF]。
pstBEAeStiStat	au16ZoneAvg	拼接模式下拼接后分区间 R、Gr、Gb、B 分量平均值，取值范围为[0, 0xFFFF]。 只有进行拼接对应 Pipe 的分区间均值有效，其余 Pipe 的均值无效。 Hi3516CV500 /Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【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。
- pstFEAeStat1、pstBEAeStat1 分别表示位于 FE 及 BE 的全局 1024 段直方图统计信息。该统计信息是取输入数据流中的高 10bit 数据统计得到的，每个 bin 中数据表示该灰度值对应的像素个数。全局 1024 段直方图会受到分区间权重的影响，1024 个 bin 的数据之和即为带权重参与统计的像素点个数。目前，上海海思 AE 算法默认只用了 Gr 通道的统计信息，在大面积红色时，会采用 R 和 Gb 通道的统计信息，在大面积蓝色时，会采用 B 和 Gr 通道的统计信息。
- pstFEAeStat2、pstBEAeStat2 分别表示位于 FE 及 BE 的全局 R/Gr/Gb/B 4 分量取高 16bit 统计得到的均值，取值范围：[0, 0xFFFF]。全局 4 分量平均值会受分区间权重影响。
- pstFEAeStat3、pstBEAeStat3 分别表示位于 FE 及 BE 的 15*17 区间每个区间 R/Gr/Gb/B 4 分量取高 16bit 统计得到的均值，取值范围：[0, 0xFFFF]。
- AE 统计模块可将输入数据开方后再进行统计。所谓数据开方，指的是对输入数据归一化至 1 后做开方处理。以 1024 段直方图统计为例，若输入数据为 12bit，某个像素值为 2048，若数据开方关闭，直接取高 10bit 数据进行统计，为灰度值 512 对应的 bin 像素个数加 1；若数据开方使能，像素值 2048 归一化至 1 后为 0.5，0.5 开方为 0.707，用 10bit 表示 0.707 为 724，此时在直方图上表现为灰度值 724 对应的 bin 像素个数加 1。由此可见，像素值较小的数据开方处理后像素值明显变大，相当于统计信息通过压缩亮区的统计精度来提升暗区的统计精度。建议在 WDR 模式下数据开方使能，线性模式下数据开方关闭。在开方模式下，统计精度为 11bit，低 5bit 值为 0，因此 16bit 最大值为 0xFFE0。此外，AE



统计模块在 ISP pipeline 的位置可以变化，具体可参考第七章“统计信息”章节部分相关描述。

表2-1 统计信息 ISP_AE_INFO_S 的成员变量默认配置说明

成员名称	主要统计信息	默认位置	黑电平	权重表影响
pstFEAeStat 1	WDR 合成前 1024 段直方图	FE-WB 之后	未减	是
pstFEAeStat 2	WDR 合成前全局均值	FE-WB 之后	未减	是
pstFEAeStat 3	WDR 合成前分块均值	FE-WB 之后	未减	否
pstBEAeStat 1	WDR 合成后 1024 段直方图	BE-WB 之后	已减	是
pstBEAeStat 2	WDR 合成后全局均值	BE-WB 之后	已减	是
pstBEAeStat 3	WDR 合成后分块均值	BE-WB 之后	已减	否

说明

- 表中说明仅在 ISP 默认配置下生效，实际会受到黑电平配置及 AE 统计信息位置的影响。
- WDR 合成前 (FE) 统计信息为固定在 WB 模块之后，不可配置。使用 FE 统计信息时需要减去黑电平，直方图应减去 10bit 黑电平，均值应减去 16bit 黑电平。WDR 合成前(FE)统计信息受到位于 FE AE 之前处理模块（在 Hi3559AV100/Hi19AV100 中，FE AE 之前的处理模块有 LSC/DG/WB，在 Hi3516CV500/Hi3559V200/ Hi3516EV200/Hi3516EV300 中，FE AE 之前的模块有 DG/WB）增益影响，这些模块的 FE 增益值由算法内部保证与 BE 保持一致，无需单独配置。
- 在 Hi3559AV100/Hi3519AV100 中第 0 路 FE 统计信息会经过水平下采样，因此点数减半。
- 线性模式推荐使用 WDR 合成前(FE)1024 段直方图。WDR 模式推荐使用 WDR 合成前 (FE)1024 段直方图，或者使用 WDR 合成后(BE)开方模式的 1024 段直方图。离线模式下业务压力大时，WDR 合成前(FE)实时性更好，因此离线模式下推荐使用 FE 统计信息。上海海思的 AE 算法在如果采用的是 BE 统计信息，默认在线性模式下使用 (BE) 不开方模式的统计信息，在 WDR 模式下使用 (BE) 开方模式的统计信息。如果线性模式下使用 (BE) 开方模式的统计信息，或者 WDR 模式下使用 (BE) 不开方模式的统计信息，会导致上海海思 AE 算法的出现异常。

【相关数据类型及接口】

[ISP_AE_EXP_FUNC_S](#)



ISP_AE_STAT_ATTR_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_STAT_ATTR_S
{
    HI_U32  u32IntTime[AE_INT_TIME_NUM];
    HI_BOOL bChange;
    HI_BOOL bHistAdjust;
    HI_U8  u8AEBESel;
    HI_U8  u8FourPlaneMode;
    HI_U8  u8HistOffsetX;
    HI_U8  u8HistOffsetY;
    HI_U8  u8HistSkipX;
    HI_U8  u8HistSkipY;
    HI_BOOL bModeUpdate;
    HI_U8  u8HistMode;
    HI_U8  u8AverMode;
    HI_U8  u8MaxGainMode;
    HI_BOOL bWightTableUpdate;
    HI_U8  au8WeightTable[VI_MAX_PIPE_NUM][AE_ZONE_ROW][AE_ZONE_COLUMN];
} ISP_AE_STAT_ATTR_S;
```



说明

宏定义如下：

```
#define AE_ZONE_ROW          (15)
#define AE_ZONE_COLUMN      (17)
```

【成员】

子成员名称	描述
bChange	该结构体中的值是否需要配置寄存器。
bHistAdjust	AE 算法直方图调整使能，影响 u8AEBESel/u8FourPlaneMode /u8HistOffsetX/ u8HistOffsetY/u8HistSkipX/u8HistSkipY 六个参数的配置。 bHistAdjust 使能时，以上六个参数以 ISP_AE_RESULT_S 中的值为准



子成员名称	描述
	配置逻辑寄存器； bHistAdjust 不使能时，以上六个参数以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
u8AEBESel	BE 中 AE 统计模块在 ISP pipeline 的位置，默认值为 1。FE 中 AE 统计模块固定在 WB 之后，位置不可移动。 0: After ISP digital gain; 1: After static WB; 2: After DRC。
u8FourPlaneMode	FourPlaneMode 使能，默认值为 0，使能时 1024 段直方图为 BGGR 四通道 256 段直方图。 0: 不使能； 1: 使能。
u8HistSkipX	直方图统计时水平方向采样点设置，0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。 即取值为 0 时表示每 1 个像素采样一个进行统计； 取值为 1 时表示每 2 个像素采样一个进行统计，以此类推。 0 仅支持 FourPlaneMode 使能时配置。
u8HistSkipY	直方图统计时垂直方向采样点设置，0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。
u8HistOffsetX	直方图统计时水平方向起始点设置。 0 表示从第一列开始统计； 1 表示从第二列开始统计。
u8HistOffsetY	直方图统计时垂直方向起始点设置。 0 表示从第一行开始统计； 1 表示从第二行开始统计。
bModeUpdate	AE 算法开方模式配置使能，影响 u8HistMode/ u8AverMode/ u8MaxGainMode 三个参数的配置。bModeUpdate 不为 0 时，以上



子成员名称	描述
	三个参数以 ISP_AE_RESULT_S 中的值为准配置逻辑寄存器； bModeUpdate 为 0 时，以上三个参数以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
u8HistMode	全局 1024 段直方图开方模式。 0 为不开方； 1 为开方。 仅影响 BE 直方图统计信息。
u8AverMode	均值开方模式。 0 为不开方； 1 为开方。 仅影响 BE 均值统计信息。
u8MaxGain Mode	MG 模块开方模式。 0 为不开方； 1 为开方。 仅影响 MG 模块统计信息。为了与 AE 分块统计信息进行比较，建议与 u8AverMode 配置为同一模式。
bWightTableUpdate	AE 算法权重表配置使能，影响 au8WeightTable 的配置。 u8WightTableUpdate 不为 0 时，权重表以 ISP_AE_RESULT_S 中的值为准配置逻辑寄存器； u8WightTableUpdate 为 0 时，权重表以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
au8Weight Table	15x17 个区间的 AE 权重表，取值范围为[0, 15]。

【注意事项】

- au8WeightTable 支持拼接模式下在主路配置多路权重，在主路对所有支路对应 Pipe 下标的权重表进行赋值即可。在非拼接模式下，只有对应 Pipe 下标的权重表生效。



- 上海海思 AE 算法只支持 1024 段直方图，在使用上海海思 AE 算法情况下使能 u8FourPlaneMode 会导致 AE 算法异常。

【相关数据类型及接口】

ISP_AE_RESULT_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  u32IntTime[AE_INT_TIME_NUM];
    HI_U32  u32IspDgain;
    HI_U32  u32Again;
    HI_U32  u32Dgain;
    HI_U32  u32Iso;
    HI_U32  u32IspDgainSF;
    HI_U32  u32AgainSF;
    HI_U32  u32DgainSF;
    HI_U32  u32IsoSF;
    HI_U8   u8AERunInterval;
    HI_BOOL bPirisValid;
    HI_S32  s32PirisPos;
    HI_U32  u32PirisGain;
    HI_U32  u32SnsLhcgExpRatio;
    ISP_FSWDR_MODE_E enFSWDRMode;
    HI_U32  au32WDRGain[WDR_MAX_FRAME_NUM];
    HI_U32  u32HmaxTimes;
    HI_U32  u32Vmax;
    ISP_AE_STAT_ATTR_S stStatAttr;
    ISP_DCF_UPDATE_INFO_S stUpdateInfo;
} ISP_AE_RESULT_S;
```



说明

宏定义如下：

```
#define AE_INT_TIME_NUM    (4)
```



【成员】

成员名称	子成员名称	描述
u32IntTime	-	AE 计算得出的曝光时间，以 1/16 us 为单位，将曝光时间由行数转换成 us 时需要考虑 cmos.c 中 f32Offset 的影响。线性模式和 sensor built-in WDR 模式下，只有 u32IntTime[0]有效，u32IntTime[1:3]建议配置等于 u32IntTime[0]；N 帧合成 WDR 模式下，u32IntTime[0:(N-1)]有效，配置值由小到大，依次表示最短到最长的曝光时间，用于计算长短帧曝光比，u32IntTime[(N-1):3]建议配置等于 u32IntTime[(N-1)]。u32IntTime[0]还需传递至其他模块用于与曝光时间相关的联动控制，会影响上海海思 AWB 效果。使用上海海思 AWB 算法和多帧 WDR 模式时必须配置该结构体。
u32IspDgain	-	ISP 的数字增益，8bit 精度。使用 ISP 数字增益时必须配置；不使用时配置为 0x100。
u32Again		Sensor 的模拟增益，10bit 精度。使用上海海思多帧 WDR 算法时必须配置；不使用时配置为 0x400。
u32Dgain		Sensor 的数字增益，10bit 精度。使用上海海思多帧 WDR 算法时必须配置；不使用时配置为 0x400。
u32Iso	-	AE 计算得出的总增益值，ISO 表示系统增益，以常数 100 乘以倍数为单位，例如系统中 sensor 的增益为 2 倍，ISP 的增益为 1 倍，那么整个系统的 ISO 值计算方式为： $2*1*100=200$ ，即系统 ISO 为 200，本文档中涉及到的 ISO 都是采用这种计算方法。此变量影响 ISP 的去噪，sharpen 等自适应效果，必须配置。在 2 合 1WDR 模式下，ISO 计算包含 WDRGain。
u32IspDgainSF	-	短帧的 ISP 的数字增益，8bit 精度。使用上海海思 WDR 算法时必须配置；不使用时配置为 0x400。



成员名称	子成员名称	描述
u32AgainSF		短帧的 Sensor 的模拟增益, 10bit 精度。使用上海海思 WDR 算法时必须配置; 不使用时配置为 0x400。
u32DgainSF		短帧的 Sensor 的数字增益, 10bit 精度。使用上海海思 WDR 算法时必须配置; 不使用时配置为 0x400。
u32IsoSF	-	AE 计算得出的短帧的总增益值。在 2 合 1WDR 模式下, ISO 计算包含 WDRGain。
u8AERunInterval	-	AE 算法运行的间隔, 取值范围为[1, 255]。 取值为 1 时表示每帧都运行 AE 算法; 取值为 2 时表示每 2 帧运行 1 次 AE 算法, 依此类推。 建议该值设置不要大于 2, 否则 AE 调节速度会受到影响。WDR 模式时, 该值建议设置为 1, 这样 AE 收敛会更加平滑。此变量决定 AE 计算结果配置到 sensor 和 ISP 寄存器的间隔帧数, 必须配置。
bPirisValid	-	Piris 是否有效的标志。 取值为 HI_TRUE 时在内核态回调 Piris 驱动配置步进电机的位置。 取值为 HI_FALSE 时不回调。 使用上海海思 AE 算法和上海海思 Piris 驱动对接 Piris 镜头时, 该值须置为 HI_TRUE, 对接非 Piris 镜头时该值须置为 HI_FALSE。
s32PirisPos	-	Piris 步进电机的位置, 取值范围与具体 Piris 镜头相关。使用上海海思 Piris 驱动对接 Piris 镜头时, 该值必须配置。
u32PirisGain	-	Piris 光圈等效增益, 取值范围与具体 Piris 镜头相关。可用于计算 Piris 工作时的等效曝光量, 供其他模块参考。取值范围为[0, 1024]。对接非 Piris 镜头时建议将该值配置为 512。



成员名称	子成员名称	描述
u32SnsLhcgExpRatio		LCG+HCG 模式的基础曝光比，仅 Sensor 支持 LCG+HCG 模式时生效，不使用时需将该值配置为 64。
enFSWDRMode	-	WDR 合成模式。 0 表示普通多帧合成 WDR 模式； 1 表示长帧模式； 2 表示自动长帧模式。
au32WDRGain	-	WDR 合成前多路的数字增益，8bit 精度。使用 WDR 合成前多路数字 ISP 数字增益时必须配置；不使用时配置为 0x100。
u32HmaxTimes	-	Sensor 对应读出一行的时间，单位：ns。
u32Vmax	-	Sensor 每帧实际生效的总行数，单位：行。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	bHistAdjust	AE 算法直方图调整使能，影响 u8AEBESel/u8FourPlaneMode /u8HistOffsetX/u8HistOffsetY/u8HistSkipX/u8HistSkipY 六个参数的配置。bHistAdjust 使能时，以上六个参数以 ISP_AE_RESULT_S 中的值为准配置逻辑寄存器；bHistAdjust 不使能时，以上六个参数以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
	u8AEBESel	BE 中 AE 统计模块在 ISP pipeline 的位置，默认值为 1。FE 中 AE 统计模块固定在 WB 之后，位置不可移动。 0: After ISP digital gain; 1: After static WB; 2: After DRC。
	u8FourPlaneMode	FourPlaneMode 使能，默认值为 0，使能时 1024 段直方图为 BGGR 四通道 256 段直方图。 0: 不使能；



成员名称	子成员名称	描述
		1: 使能。
	u8HistSkipX	直方图统计时水平方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。即取值为 0 时表示每 1 个像素采样一个进行统计, 取值为 1 时表示每 2 个像素采样一个进行统计, 以此类推。0 仅支持 FourPlaneMode 使能时配置。
	u8HistSkipY	直方图统计时垂直方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。
	u8HistOffsetX	直方图统计时水平方向起始点设置。 0 表示从第一列开始统计; 1 表示从第二列开始统计。
	u8HistOffsetY	直方图统计时垂直方向起始点设置。 0 表示从第一行开始统计; 1 表示从第二行开始统计。
	bModeUpdate	AE 算法开方模式配置使能, 影响 u8HistMode/ u8AverMode/ u8MaxGainMode 三个参数的配置。bModeUpdate 不为 0 时, 以上三个参数以 ISP_AE_RESULT_S 中的值为准配置逻辑寄存器; bModeUpdate 为 0 时, 以上三个参数以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
	u8HistMode	全局 1024 段直方图开方模式。0 为不开方, 1 为开方。仅影响 BE 直方图统计信息。
	u8AverMode	均值开方模式。0 为不开方, 1 为开方。仅影响 BE 均值统计信息。
	u8MaxGainMode	MG 模块开方模式。0 为不开方, 1 为开方。仅影响 MG 模块统计信息。为了与 AE 分块统计信息进行比较, 建议与 u8AverMode 配置为同一模式。



成员名称	子成员名称	描述
	bWightTableUpdate	AE 算法权重表配置使能，影响 bWeightTable 的配置。bWightTableUpdate 不为 0 时，权重表以 ISP_AE_RESULT_S 中的值为准配置逻辑寄存器；bWightTableUpdate 为 0 时，权重表以外部寄存器的值(MPI 配置)为准配置逻辑寄存器。
	au8WeightTable	15x17 个区间的 AE 权重表，取值范围为[0, 15]。
stUpdateInfo	-	用于传递 AE 相关 DCF 信息，仅曝光相关参数生效。详细描述请参考 ISP_DCF_UPDATE_INFO_S 说明。

【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。
- au8WeightTable 支持拼接模式下在主路配置多路权重，在主路对所有支路对应 Pipe 下标的权重表进行赋值即可。在非拼接模式下，只有对应 Pipe 下标的权重表生效。
- 将曝光时间 u32IntTime 由行转换成 us 时，可以通过 cmos.c 中的 u32LinesPer500ms 进行，转换关系如下：
$$u32IntTime[0] = (((HI\_U64)au32IntTimeRst[0] * 1024 - s32Offset) * 500000 / pstAeSnsDft \rightarrow u32LinesPer500ms) \gg 10$$

上式中 au32IntTimeRst[0]是以行数为单位的曝光时间，s32Offset=f32Offset * 1024，f32Offset 即为曝光时间的偏移量，详见 [AE_ACCURACY_S](#) 的描述。
- 为了保证计算曝光比的精度，u32IntTime 中曝光时间精度为 1/16us，计算曝光时间时需要保证在 us 的基础上左移四位，否则曝光比计算可能会有误差，其他模块获取的曝光时间也会偏小。
- 客户在使用非上海海思 AE 算法时，参数 u32IntTime、u32IspDgain、u32Again、u32Dgain、u32Iso、u32HmaxTimes、u32Vmax 必须配置，否则会影响其他模块的联动控制，其余参数则可根据需要选择性配置。

【相关数据类型及接口】

[ISP_AE_EXP_FUNC_S](#)



ISP_AWB_REGISTER_S

【说明】

定义 AWB 注册结构体。

【定义】

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

【成员】

成员名称	描述
stAwbExpFunc	AWB 注册的回调函数结构体。

【注意事项】

封装的目的是为了扩展。

【相关数据类型及接口】

[ISP_AWB_EXP_FUNC_S](#)

ISP_AWB_EXP_FUNC_S

【说明】

定义 AWB 回调函数结构体。

【定义】

```
typedef struct hiISP_AWB_EXP_FUNC_S
{
    HI_S32 (*pfn_awb_init)(HI_S32 s32Handle, const ISP_AWB_PARAM_S *pstAwbParam,
    ISP_AWB_RESULT_S *pstAwbResult);
    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。
- pfn_awb_init 回调函数的 [ISP_AWB_RESULT_S](#) 参数返回 ISP 启动时的初始 AWB 增益和初始色彩校正矩阵。
- 调用 [HI_MPI_ISP_Run](#) 时将调用 pfn_awb_run 回调函数，以运行 AWB 算法库，计算得到白平衡增益、色彩校正矩阵。
- 运行时 ISP 控制单元会隐式调用 pfn_awb_ctrl 回调函数，通知 AWB 算法库切换 WDR 和线性模式、设置 ISO 和曝光时间（曝光时间单位是 us）。设置 ISO 的目的是为了实现 ISO 与饱和度的联动，增益大时色度噪声也会比较大，所以需要调节饱和度。设置曝光时间是为了辅助室内外判断。

当前 Firmware 定义的 ctrl 命令详细描述参见 [ISP_CTRL_CMD_E](#)。

- 调用 [HI_MPI_ISP_Exit](#) 时将调用 pfn_awb_exit 回调函数，以销毁 AWB 算法库。

【相关数据类型及接口】

[ISP_AWB_REGISTER_S](#)

ISP_AWB_PARAM_S

【说明】

定义 ISP 提供给 AWB 的初始化参数结构体。

【定义】

```
typedef struct hiISP_AWB_PARAM_S
```



```
{  
    SENSOR_ID SensorId;  
    HI_U8 u8WDRMode;  
    HI_U8 u8AWBZoneRow;  
    HI_U8 u8AWBZoneCol;  
    HI_U8 u8AWBZoneBin;  
    ISP_STITCH_ATTR_S stStitchAttr;  
    HI_U16 u16AWBWidth;  
    HI_U16 u16AWBHeight;  
    HI_U32 u32InitIso;  
    HI_S8 s8Rsv;  
} ISP_AWB_PARAM_S;
```

【成员】

成员名称	描述
SensorId	向 ISP 注册的 sensor 的 id，用以检查向 ISP 注册的 sensor 和向 AWB 注册的 sensor 是否一致。
u8WDRMode	宽动态模式，ISP 向 AWB 提供宽动态模式信息。
u8AWBZoneRow	AWB 统计结果行数。
u8AWBZoneCol	AWB 统计结果列数。
u8AWBZoneBin	AWB 统计结果亮度分组个数。
stStitchAttr	拼接信息结构体， Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持
u16AWBWidth	Firmware 传递给 AWB 算法库的图像宽度。
u16AWBHeight	Firmware 传递给 AWB 算法库的图像高度。
u32InitIso	Firmware 传递给 AWB 算法库的 AE 曝光 ISO 值。
s8Rsv	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_STAT_1_S

【说明】

定义 AWB 统计信息结构体。

【定义】

```
typedef struct hiISP_AWB_STAT_1_S
{
    HI_U16  u16MeteringAwbAvgR;
    HI_U16  u16MeteringAwbAvgG;
    HI_U16  u16MeteringAwbAvgB;
    HI_U16  u16MeteringAwbCountAll;
} ISP_AWB_STAT_1_S;
```

【成员】

成员名称	描述
u16MeteringAwbAvgR	Bayer 域全局统计信息中白点的 R 分量平均值。 R 分量均值取值范围为[0, 0xFFFF]。
u16MeteringAwbAvgG	Bayer 域全局统计信息中白点的 G 分量平均值。 G 分量均值取值范围为[0, 0xFFFF]。
u16MeteringAwbAvgB	Bayer 域全局统计信息中白点的 B 分量平均值。 B 分量均值取值范围为[0, 0xFFFF]。
u16MeteringAwbCountAll	Bayer 域全局统计信息中白点的个数。已做归一化。 白点个数取值范围为[0, 0xFFFF]。

【注意事项】

无

【相关数据类型及接口】

[ISP_AWB_INFO_S](#)



ISP_AWB_STAT_RESULT_S

【说明】

定义 AWB 统计信息结构体。

【定义】

```
typedef struct hiISP_AWB_STAT_RESULT_S
{
    HI_U16* pau16ZoneAvgR;
    HI_U16* pau16ZoneAvgG;
    HI_U16* pau16ZoneAvgB;
    HI_U16* pau16ZoneCount;
} ISP_AWB_STAT_RESULT_S;
```

【成员】

成员名称	描述
* pau16ZoneAvgR	Bayer 域分区间统计信息中白点的 R 分量平均值数组起始地址。 R 分量均值取值范围为[0, 0xFFFF]。
* pau16ZoneAvgG	Bayer 域分区间统计信息中白点的 G 分量平均值数组起始地址。 G 分量均值取值范围为[0, 0xFFFF]。
* pau16ZoneAvgB	Bayer 域分区间统计信息中白点的 B 分量平均值数组起始地址。 B 分量均值取值范围为[0, 0xFFFF]。
* pau16ZoneCount	Bayer 域分区间统计信息中白点的个数数组起始地址。已做归一化。 白点个数取值范围为[0, 0xFFFF]。

【注意事项】

无

【相关数据类型及接口】



ISP_AWB_INFO_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_RESULT_S stAwbStat2;
    HI_U8  u8AwbGainSwitch;
    HI_U8  u8AwbStatSwitch;
    HI_BOOL bWbGainInSensor;
    HI_U32 au32WDRWBGain[ISP_BAYER_CHN_NUM];
} ISP_AWB_INFO_S;
```

【成员】

成员名称	描述
u32FrameCnt	帧的累加计数。 取值范围为[0, 0xFFFFFFFF]。
pstAwbStat1	Awb 统计信息 1
stAwbStat2	Awb 统计信息 2
u8AwbGainSwitch	白平衡增益在 ISP 的位置。取值范围为[0,1]。 0：白平衡增益配置在 WDR 合成前的 DG1 处。 1：白平衡增益配置在 WB 处。
u8AwbStatSwitch	白平衡统计模块在 ISP 的位置。取值范围为[0, 1, 2]。 0：白平衡统计模块在 DG 后。 1：白平衡统计模块在 EXPANDER 后。 2：白平衡统计模块在 DRC 后。



成员名称	描述
bWbGainInSensor	白平衡增益是否在 Sensor 配置。取值范围为[0, 1]。 0：白平衡增益在 ISP 配置。 1：白平衡增益在 Sensor 配置。
au32WDRWBGain [ISP_BAYER_CHN_NUM]	在 WDR 合成前的 DG1 处配置的白平衡增益值。

【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。
- ISP_AWB_INFO_S 提供了全局统计信息和分块统计信息，不同模式下水平、垂直分块个数不固定，可通过 ISP_AWB_PARAM_S 的 u8AWBZoneRow、u8AWBZoneCol、u8AWBZoneBin 三个参数获取。AWB 分块个数 = u8AWBZoneRow * u8AWBZoneCol * u8AWBZoneBin。
- 切换白平衡统计模块位置，会导致 2 帧统计信息错误，建议在启动时配合合理值，避免切换。如果必须进行位置切换，建议切换后，AWB 算法冻结 2 帧以上，待统计信息正确后，再重新计算。
- 通过 [HI_MPI_ISP_SetStatisticsConfig](#) 接口配置白平衡统计模块位置在 DRC 后，通过 [HI_MPI_ISP_SetWBAttr](#) 接口关闭亮度对白平衡权重的影响，可以优化 WDR 模式下暗区偏红问题。

【相关数据类型及接口】

[ISP_AWB_EXP_FUNC_S](#)

ISP_AWB_RAW_STAT_ATTR_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_RAW_STAT_ATTR_S
{
    HI_BOOL bStatCfgUpdate;
    HI_U16  u16MeteringWhiteLevelAwb;
    HI_U16  u16MeteringBlackLevelAwb;
```



```
    HI_U16  u16MeteringCrRefMaxAwb;  
    HI_U16  u16MeteringCbRefMaxAwb;  
    HI_U16  u16MeteringCrRefMinAwb;  
    HI_U16  u16MeteringCbRefMinAwb;  
} ISP_AWB_RAW_STAT_ATTR_S;
```

【成员】

成员名称	描述
bStatCfgUpdate	该结构体中的值是否需要配置寄存器
u16MeteringWhiteLevelAwb	Bayer 域统计白点信息时，找白点的亮度上限。 取值范围：[0x0, 0xFFFF]，默认值 0xFFFF。
u16MeteringBlackLevelAwb	Bayer 域统计白点信息时，找白点的亮度下限。 取值范围：[0x0, 0xFFFF]，默认值 0x0。
u16MeteringCrRefMaxAwb	Bayer 域统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。
u16MeteringCbRefMaxAwb	Bayer 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。
u16MeteringCrRefMinAwb	Bayer 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。
u16MeteringCbRefMinAwb	Bayer 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。

【注意事项】

无

【相关数据类型及接口】

无

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];
    HI_U32  u32ColorTemp;
    HI_U8   u8Saturation;
    ISP_AWB_RAW_STAT_ATTR_S stRawStatAttr;
} ISP_AWB_RESULT_S;
```

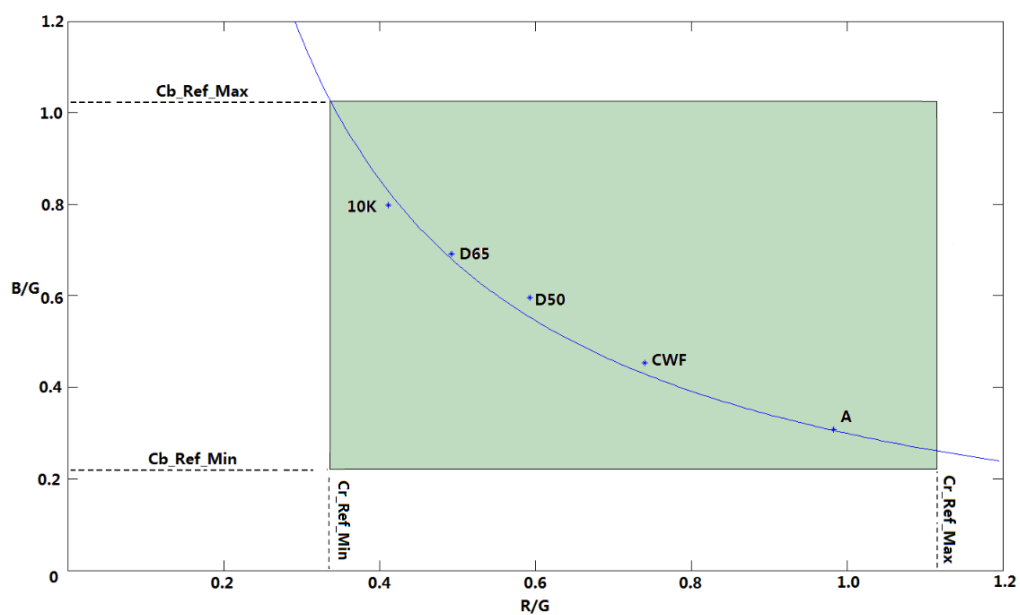
【成员】

成员名称	子成员名称	描述
au32WhiteBalanceGain	-	白平衡算法得出的 R、Gr、Gb、B 颜色通道的增益，16bit 精度表示。
au16ColorMatrix	-	色彩还原矩阵，8bit 精度表示。
u32ColorTemp	-	AWB 当前的色温。
u8Saturation	-	当前的饱和度。
stRawStatAttr	bStatCfgUpdate	该结构体中的值是否需要配置寄存器。
	u16MeteringWhiteLevelAwb	Bayer 域统计白点信息时，找白点的亮度上限。 取值范围：[0x0, 0xFFFF]，默认值 0xFFFF。
	u16MeteringBlackLevelAwb	Bayer 域统计白点信息时，找白点的亮度下限。 取值范围：[0x0, 0xFFFF]，默认值 0x0。
	u16MeteringCrRefMaxAwb	Bayer 域统计白点信息时，色差 R/G 的最大值，



成员名称	子成员名称	描述
		8bit 精度，默认值 512。
	u16MeteringCbRefMaxAwb	Bayer 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。
	u16MeteringCrRefMinAwb	Bayer 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。
	u16MeteringCbRefMinAwb	Bayer 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。

图2-7 白色区域选择相关参数



【注意事项】

- AWB 算法首先将会计算出 R、Gr、Gb、B 颜色通道的增益，以校正出白色，16bit 精度表明最后 16 位为小数。



- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200/Hi3516EV300 WDR 模式，AWB 统计结果和增益不需要做特别处理，和 Linear 模式一致。
- AWB 算法其次会计算一个 3x3 的颜色校正矩阵，以还原出真实的色彩，8bit 精度表明最后 8 位为小数。
- stStatAttr 结构体中的信息决定什么样的像素点被认为是白点，从而参与统计。用户开发新的 AWB 算法时可以使用默认值，也可以自定义配置，bStatCfgUpdate 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。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300AWB 统计模块在 ISP Pipeline 位置支持可配置，可配置为 Digital Gain 后，Expander 后、或 DRC 后。默认在 Digital Gain 后。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 只支持 Bayer 域统计信息。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持 u16MeteringCrRefHighAwb、u16MeteringCrRefLowAwb、u16MeteringCbRefHighAwb、u16MeteringCbRefLowAwb 四个参数。

【相关数据类型及接口】

[ISP_AWB_EXP_FUNC_S](#)

ISP_AWB_Calibration_Gain_S

【说明】

定义 AWB 在线标定输出的增益结构体。

【定义】

```
typedef struct hiISP_AWB_Calibration_Gain_S
{
    HI_U16 u16AvgRgain;
    HI_U16 u16AvgBgain;
}ISP_AWB_Calibration_Gain_S;
```

【成员】

成员名称	描述
u16AvgRgain	AWB 在线标定输出的 Rgain 的值。
u16AvgBgain	AWB 在线标定输出的 Bgain 的值。



【注意事项】

Rgain 和 Bgain 值为图像中心位置若干个块的 Rgain 和 Bgain 的均值。

【相关数据类型及接口】

无

ISP_DCF_CONST_INFO_S

【说明】

定义 DCF 信息中用户配置参数。

【定义】

```
typedef struct hiISP_DCF_CONST_INFO_S
{
    HI_U8      au8ImageDescription[DCF_DRSCRIPTION_LENGTH];
    HI_U8      au8Make[DCF_DRSCRIPTION_LENGTH];
    HI_U8      au8Model[DCF_DRSCRIPTION_LENGTH];
    HI_U8      au8Software[DCF_DRSCRIPTION_LENGTH];
    HI_U8      u8LightSource;
    HI_U32     u32FocalLength;
    HI_U8      u8SceneType;
    HI_U8      u8CustomRendered;
    HI_U8      u8FocalLengthIn35mmFilm;
    HI_U8      u8SceneCaptureType;
    HI_U8      u8GainControl;
    HI_U8      u8Contrast;
    HI_U8      u8Saturation;
    HI_U8      u8Sharpness;
    HI_U8      u8MeteringMode;
} ISP_DCF_CONST_INFO_S;
```



说明

#define DCF_DRSCRIPTION_LENGTH 32

【成员】



成员名称	描述
au8ImageDescription	图像描述、来源，指生成图像的工具。 数据格式：ascii strings，长度最大 32。
au8Make	生产者，指产品生产厂家。 数据格式：ascii strings，长度最大 32。
au8Model	型号，指设备型号。 数据格式：ascii strings，长度最大 32。
au8Software	软件，显示固件 Firmware 版本。 数据格式：ascii strings，长度最大 32。
u8MeteringMode	曝光的测光模式。 0：未知； 1：平均测光； 2：中央重点测光； 3：点测光； 4：多点测光； 5：多区域测光； 6：部分测光； 255：其他。 数据格式：unsigned short。 只读。
u8LightSource	光源，表示白平衡设置。 0：未知。 1：日光； 2：荧光灯； 3：白炽灯(钨丝)； 4：闪光灯； 10：阴天； 17：标准光 A；



成员名称	描述
	18: 标准光 B; 19: 标准光 C; 20: D55; 21: D65; 22: D75; 255: 其他。 数据格式: unsigned short。
u32FocalLength	拍摄照片时的镜头的焦距长度, 单位是毫米。数据格式: unsigned rational, 高 16 位为分子, 低 16 位为分母。
u8SceneType	表示拍摄场景的类型, 值 '0x01' 表示图像是指通过相机直接拍摄。 数据格式: unsigned short。 暂不支持。
u8CustomRendered	自定义图像处理。0: 标准; 1: 自定义。
u8FocalLengthIn35mmFilm	35 毫米胶片焦距, 0: 表示这个焦距不存在。 数据格式: unsigned short。
u8SceneCaptureType	场景拍摄类型。 0: 标准; 1: 风景模式; 2: 肖像模式; 3: 夜间模式。 数据格式: unsigned short。
u8GainControl	增益控制。 0: None; 1: Low gain up; 2 : High gain up; 3: Low gain down;



成员名称	描述
	4: High gain down。 数据格式: unsigned short。
u8Contrast	对比度。 数据格式: unsigned short。
u8Saturation	饱和度。 0: 无; 1: 低; 2: 高。 数据格式: unsigned short。
u8Sharpness	锐度。 数据格式: unsigned short。

【注意事项】

无

【相关数据类型及接口】

- [HI_MPI_ISP_SetDCFInfo](#)
- [HI_MPI_ISP_GetDCFInfo](#)

ISP_DCF_UPDATE_INFO_S

【说明】

定义 DCF 信息中 ISP 实时更新配置参数。

【定义】

```
typedef struct hiISP_DCF_UPDATE_INFO_S
{
    HI_U32    u32ISOSpeedRatings;
    HI_U32    u32ExposureTime;
    HI_U32    u32ExposureBiasValue;
    HI_U8     u8ExposureProgram;
    HI_U32    u32FNumber;
```



```
    HI_U32    u32MaxApertureValue;  
    HI_U8     u8ExposureMode;  
    HI_U8     u8WhiteBalance;  
} ISP_DCF_UPDATE_INFO_S;
```

【成员】

成员名称	描述
u32ISOSpeedRatings	感光度。 数据格式：unsigned long。 只读。
u32FNumber	光圈系数。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32MaxApertureValue	镜头的最大光圈值。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32ExposureTime	曝光时间（快门速度的倒数），单位是秒。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u32ExposureBiasValue	照片拍摄时的曝光补偿，单位是 APEX(EV)。 数据格式：unsigned rational，高 16 位为分子，低 16 位为分母。 只读。
u8ExposureProgram	拍照时相机使用的曝光程序。 1：手动曝光； 2：正常程序曝光； 3：光圈优先曝光；



成员名称	描述
	4: 快门优先曝光; 5: 创意程序(慢速程序); 6: 动作程序(高速程序); 7: 肖像模式; 8: 风景模式。 数据格式: unsigned short。 只读。
u8ExposureMode	曝光模式。 0: 自动曝光; 1: 手动曝光; 2: 自动包围曝光。 数据格式: unsigned short。 只读。
u8WhiteBalance	白平衡。 0: 自动白平衡; 1: 手动白平衡。 数据格式: unsigned short。 只读。

【注意事项】

无

【相关数据类型及接口】

- [HI_MPI_ISP_SetDCFInfo](#)
- [HI_MPI_ISP_GetDCFInfo](#)

ISP_DCF_INFO_S

【说明】

定义 DCF 信息参数结构体。



【定义】

```
typedef struct hiISP_DCF_INFO_S
{
    ISP_DCF_CONST_INFO_S    stIsPDCFConstInfo;
    ISP_DCF_UPDATE_INFO_S   stIsPDCFUpdateInfo;
}ISP_DCF_INFO_S;
```

【成员】

成员名称	描述
stIsPDCFConstInfo	DCF 信息中用户配置参数。
stIsPDCFUpdateInfo	DCF 信息中 ISP 实时更新参数。

【注意事项】

无

【相关数据类型及接口】

- [HI_MPI_ISP_SetDCFInfo](#)
- [HI_MPI_ISP_GetDCFInfo](#)

ISP_PIPE_DIFF_ATTR_S

【说明】

定义两路 ISP 差异参数结构体。

【定义】

```
typedef struct hiISP_PIPE_DIFF_ATTR_S
{
    HI_S32 as32Offset[ISP_BAYER_CHN_NUM];
    HI_U32 au32Gain[ISP_BAYER_CHN_NUM];
    HI_U16 au16ColorMatrix[CCM_MATRIX_SIZE];
} ISP_PIPE_DIFF_ATTR_S;
```

【成员】



成员名称	描述
as32Offset[ISP_BAYER_CHN_NUM]	多路黑电平差异偏移值。 取值范围：[-0xFFF, 0xFFF] 该数组 4 个值分别对应 R、Gr、Gb、B 四个通道。
au32Gain[ISP_BAYER_CHN_NUM]	多路增益差异比值，8bit 小数精度。 取值范围：[0x80, 0x400] 该数组 4 个值分别对应 R、Gr、Gb、B 四个通道。
au16ColorMatrix[CCM_MATRIX_SIZE]	多路颜色校正矩阵差异比值，8bit 小数精度。 bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。 取值范围：[0x0, 0xFFFF]

【注意事项】

ISP_PIPE_DIFF_ATTR_S 主要用于拼接模式下，配置多路图像的亮度和颜色的差异，用于校正多路的图像的差异，从而使得拼接画面融合区域过渡平滑。

【相关数据类型及接口】

无

ISP_OP_TYPE_E

【说明】

定义模块运行状态。

【定义】

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



```
} ISP_OP_TYPE_E;
```

【成员】

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

【注意事项】

无

【相关数据类型及接口】

- [ISP_ME_ATTR_S](#)
- [ISP_EXPOSURE_ATTR_S](#)
- [ISP_WDR_EXPOSURE_ATTR_S](#)
- [ISP_HDR_EXPOSURE_ATTR_S](#)
- [ISP_IRIS_ATTR_S](#)
- [ISP_AWB_LUM_HISTGRAM_ATTR_S](#)
- [ISP_AWB_CT_LIMIT_ATTR_S](#)
- [ISP_WB_ATTR_S](#)
- [ISP_AWB_IN_OUT_ATTR_S](#)
- [ISP_SATURATION_ATTR_S](#)
- [ISP_COLOMATRIX_ATTR_S](#)
- [ISP_SHARPEN_ATTR_S](#)
- [ISP_DRC_ATTR_S](#)
- [ISP_DP_DYNAMIC_ATTR_S](#)
- [ISP_NR_ATTR_S](#)
- [ISP_DEHAZE_ATTR_S](#)
- [ISP_ANTIFALSECOLOR_ATTR_S](#)
- [ISP_DEMOSAIC_ATTR_S](#)
- [ISP_BLACK_LEVEL_S](#)
- [ISP_FPN_ATTR_S](#)
- [ISP_LOCAL_CAC_ATTR_S](#)
- [ISP_FSWDR_MDT_ATTR_S](#)
- [ISP_LDCI_ATTR_S](#)



- ISP_DE_ATTR_S

ISP_CTRL_PARAM_S

【说明】

定义 ISP 控制参数结构体。

【定义】

```
typedef struct hiISP_CTRL_PARAM_S
{
    HI_U8      u8BeBufNum;
    HI_U32     u32ProcParam;
    HI_U32     u32StatIntvl;
    HI_U32     u32UpdatePos;
    HI_U32     u32IntTimeOut;
    HI_U32     u32PwmNumber;
    HI_U32     u32PortIntDelay;
    HI_BOOL    bLdciTprFltEn;
}ISP_CTRL_PARAM_S;
```

【成员】

成员名称	描述
u8BeBufNum	离线模式下，ISP BE config buffer 的数目。仅离线模式有效取值范围：[2, 9]，默认值为 4
u32ProcParam	ISP 的 PROC 信息更新频率。默认值 30 最小值为 1，无上限。u32ProcParam 为 n 表示每隔 n 帧更新一次 ISP 的 PROC 信息。
u32StatIntvl	ISP 统计信息更新频率。注意：在高帧率业务时，如 120fps 及其以上帧率，可通过参数 u32StatIntvl 设置其 ISP 统计信息更新频率，从而可降低 ISP 的 CPU 占用率，同时降低性能消耗。 取值范围：(0,0xffffffff]
u32UpdatePos	默认值为 0。 0：根据 ISP_SNS_REGS_INFO_S 结构体中 u8IntPos 变量的配置值，sensor 寄存器在帧起始中断或帧结束中断配置；



成员名称	描述
	其他值：sensor 寄存器在帧结束中断配置。 取值范围：[0,1]
u32IntTimeOut	表示中断超时的时间（ms）。
u32PwmNumber	表示 PWM 号。
u32PortIntDelay	表示 Port 中断延时时间，默认值为 0。 解决部分 sensor（如：imx323）Half WDR 模式下，在前面几行配置 sensor 寄存器会出现闪烁，需要延时。 port_init_delay 是按照 VI 的工作时钟频率计算，以时钟为单位。如：VI 时钟是 300MHz。延时 1ms，port_init_delay 的计算如下： $\text{port\_init\_delay (1ms)} = 300\text{M}/1000\text{ms} = 300000$ 注意：port_init_delay 仅在 Half WDR 模式下生效，因为其他模式配置 sensor 不用 Port 中断。
bLdciTprFltEn	表示 LDCI 是否使能时域滤波，默认值为 0。

【注意事项】

- u32ProcParam 的默认值为 30，即每 30 帧更新一次 ISP Proc 信息，如果要关闭 ISP Proc 信息，则在 [HI_MPI_ISP_MemInit](#) 之前通过接口 [HI_MPI_ISP_SetCtrlParam](#) 设置 ISP 控制参数 u32ProcParam 为 0，不分配存储 ISP Proc 信息的内存，而且后续不能将 u32ProcParam 设置为非 0 值。
- 通过接口 [HI_MPI_ISP_SetCtrlParam](#) 第一次设置 u32ProcParam 为非 0 时，必需在 [HI_MPI_ISP_MemInit](#) 接口调用之前，因为需要分配 Proc 信息存储的内存，后续设置只能在非 0 值之间动态切换。
- ISP Proc 信息频繁更新会消耗 CPU 资源。推荐设为 30 帧更新一次，或仅 Debug 时开启。
- 只能在 [HI_MPI_ISP_MemInit](#) 之前通过接口 [HI_MPI_ISP_SetCtrlParam](#) 设置 u32UpdatePos、u32PwmNumber、u32PortIntDelay、bLdciTprFltEn 和 u8BeBufNum，在 [HI_MPI_ISP_MemInit](#) 之后不可再更改上述 5 个参数的配置值；而 [HI_MPI_ISP_GetCtrlParam](#) 接口调用顺序没有限制。



- 可通过接口 [HI_MPI_ISP_SetCtrlParam](#) 动态更改 u32ProcParam、u32StatIntvl、u32IntTimeOut 的配置值。
- 不支持加载 ko 时设置 ISP 控制参数。
- 离线模式下，VI 缓存 RAW 数据的队列深度为 8，若业务量较大时，采用默认的 u8BeBufNum，有可能会出现不丢帧但是 ISP 中有 “get FreeBeBuf is fail” 错误打印的问题，此时可以增大 u8BeBufNum 的配置值，缓解此问题；另外对于业务量不繁重、并且对于内存占用要求较高的应用场景，可以适当的减小 u8BeBufNum 的配置值。

【相关数据类型及接口】

- [HI_MPI_ISP_SetCtrlParam](#)
- [HI_MPI_ISP_GetCtrlParam](#)

ISP_MOD_PARAM_S

【说明】

定义 ISP 模块参数结构体。

【定义】

```
typedef struct hiISP_MOD_PARAM_S
{
    HI_U32      u32IntBotHalf;
    HI_U32      u32QuickStart;
    HI_BOOL     bLongFrmlIntEn;
}ISP_MOD_PARAM_S;
```

【成员】

成员名称	描述
u32IntBotHalf	表示 ISP 中断处理是否采用底半部机制，默认值为 0。 u32IntBotHalf =0：ISP 内核态处理（读统计信息和配置 sensor 和 ISP 同步寄存器）在中断服务程序中完成； u32IntBotHalf =1：ISP 内核态处理（读统计信息和配置 sensor 和 ISP 同步寄存器）在中断下半部完成。
u32QuickStart	表示 ISP 是否采用快速启动，默认值为 0。 u32QuickStart=0：ISP 初始化时配置 sensor 序列



成员名称	描述
	u32QuickStart=1：ISP 初始化时不配置 sensor 序列。仅在 Hi3516EV200/Hi3516EV300 中支持。
bLongFrmIntEn	表示 WDR 时，ISP 是否响应长帧中断，默认值为 0。 0：不使能； 1：使能。 仅在 Hi3516EV200/Hi3516EV300 中支持。

【注意事项】

- 通过接口 [HI_MPI_ISP_SetModParam](#) 设置 u32IntBotHalf 中断底半部时，接口调用需要在 ISP 起主业务之前（如：起多路 PIPE 场景业务时，需要在将接口调用顺序放在起多路主业务之前），且 ISP KO 已经加载；而 [HI_MPI_ISP_GetModParam](#) 接口调用顺序没有限制，可查询当前状态。
- 通过接口 [HI_MPI_ISP_SetModParam](#) 设置 u32QuickStart 快速启动时，接口调用需要在 ISP 起主业务之前（如：起多路 PIPE 场景业务时，需要在将接口调用顺序放在起多路主业务之前），且 ISP KO 已经加载；而 [HI_MPI_ISP_GetModParam](#) 接口调用顺序没有限制，可查询当前状态
- 不支持加载 ko 时设置 ISP 模块参数。
- bLongFrmIntEn 设置为 1 时，WDR 模式时，ISP 响应中断次数会增加，会影响 ISP 中断响应时间。增加 cpu 负载。

【相关数据类型及接口】

- [HI_MPI_ISP_SetModParam](#)
- [HI_MPI_ISP_GetModParam](#)

ISP_QUICK_START_PARAM_S

【说明】

设定不带光敏 AE 快速启动参数结构体。

【定义】

```
typedef struct hiISP_QUICK_START_PARAM_S
{
    HI_BOOL bQuickStartEn;
```



```
HI_U8 u8BlackFrameNum;  
HI_BOOL bIrModeEn;  
HI_U32 u32InitExposureIr;  
HI_U32 u32ISOThrIr;  
HI_U16 u16IrCutDelayTime;  
} ISP_QUICK_START_PARAM_S;
```

【成员】

成员名称	描述
bQuickStartEn	设置是否打开 AE 启动快速收敛模式。该值为 HI_TRUE 时，打开 AE 开机快速收敛模式，AE 启动时会以最快速度进行收敛（对于大部分场景可在 10 帧内收敛完成）。此模式可以在没有光敏的时候达到启动 AE 快速收敛的需求。
u8BlackFrameNum	Sensor 初始输出的坏帧数。此参数在 bQuickStartEn 为 HI_TRUE 的情况下生效。
bIrModeEn	AE 快速收敛模式下，支持红外模式开关。该值为 HI_TRUE 时，AE 快速收敛支持红外模式。此参数在 bQuickStartEn 为 HI_TRUE 的情况下生效。
u32InitExposureIr	红外模式下的初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。此参数在 bQuickStartEn 和 bIrModeEn 都为 HI_TRUE 的情况下生效。
u32ISOThrIr	在 AE 快速收敛模式下，设定切换到红外模式（打开 IR CUT、IR LED）的 ISO 阈值，此参数在 bQuickStartEn 为 HI_TRUE 的情况下生效。
u16IrCutDelayTime	设置打开 IR CUT 物理上需要的时间。单位为 ms。此参数在 bQuickStartEn 为 HI_TRUE 的情况下生效。

【注意事项】

- u8BlackFrameNum 填入 Sensor 初始输出的坏帧数。因部分 Sensor 在启动时，会输出若干坏帧。启动时没有坏帧的 Sensor 则填入值为 0。



- 打开 AE 快速收敛模式状态下，如果外部硬件上没有 IR CUT 和 IR LED，建议将 blrModeEn 设定为 HI_FALSE。此时 AE 启动快速收敛模式，不会进行相应的 IR 快速收敛模式。
- 在打开 IR CUT 过程中，对 Sensor 的产生较大的亮度变化，此时 AE 算法内部会等待 IR CUT 切换完成。为了保证 AE 收敛的速度，加快快速收敛的速度，u16IrCutDelayTime 的值要设定合适，不能远大于实际 IR CUT 切换的时间。

【相关数据类型及接口】

无。

ISP_INIT_ATTR_S

【说明】

定义 ISP 第一次启动时 AE/AWB 的初始化参数结构体。

【定义】

```
typedef struct hiISP_INIT_ATTR_S
{
    HI_BOOL blsrMode;
    HI_U32 u32ExpTime;
    HI_U32 u32AGain;
    HI_U32 u32DGain;
    HI_U32 u32ISPDGain;
    HI_U32 u32Exposure;
    HI_U32 u32InitIso;
    HI_U32 u32LinesPer500ms;
    HI_U32 u32PirisFNO;
    HI_U16 u16WBRgain;
    HI_U16 u16WBGgain;
    HI_U16 u16WBBgain;
    HI_U16 u16SampleRgain;
    HI_U16 u16SampleBgain;
    HI_U16 au16CCM[CCM_MATRIX_SIZE];
    HI_BOOL bAERouteExValid;
    ISP_QUICK_START_PARAM_S stQuickStart;
    ISP_AE_ROUTE_S stAERoute;
    ISP_AE_ROUTE_EX_S stAERouteEx;
    ISP_AE_ROUTE_S stAERouteSF;
```



```
ISP_AE_ROUTE_EX_S stAERouteSFEx;  
} ISP_INIT_ATTR_S;
```

【成员】

成员名称	描述
bIsIrMode	设置 ISP 启动状态是否红外模式。
u32ExpTime	设置 ISP 第一次启动时 AE 初始曝光时间，单位为 us，FSWDR 模式下，表示当前最短帧(VS)曝光时间。不支持。
u32AGain	设置 ISP 第一次启动时 AE 初始 sensor 模拟增益，精度为 10bit，不支持。
u32DGain	设置 ISP 第一次启动时 AE 初始 sensor 数字增益，精度为 10bit，不支持。
u32ISPDGain	设置 ISP 第一次启动时 AE 初始 ISP 数字增益，精度为 10bit，不支持。
u32Exposure	设置 ISP 第一次启动时 AE 初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。
u32InitIso	设置 ISP 第一次启动时 AE 初始 ISO 值。
u32LinesPer500ms	设置每 500ms 对应的曝光行数，用于计算 AE 初始曝光量。
u32PirisFNO	P-Iris 光圈 F 值对应的等效增益，不支持。
u16WBRgain	设置 ISP 第一次启动时 AWB 的 R 通道增益。
u16WBGgain	设置 ISP 第一次启动时 AWB 的 G 通道增益。
u16WBBgain	设置 ISP 第一次启动时 AWB 的 B 通道增益。
u16SampleRgain	设置当前样机 AWB 在线标定的 G/R 的值。
u16SampleBgain	设置当前样机 AWB 在线标定的 G/B 的值。
au16CCM	设置 ISP 第一次启动时 CCM 值。
bAERouteExValid	设置 ISP 第一次启动时 AE 扩展分配路线生效开关，该值为 HI_TRUE 时使用扩展分配路线，否则使用普通分配路线。
stQuickStart	设置不带光敏快启相关参数。



成员名称	描述
stAERoute	设置 ISP 第一次启动时 AE 的曝光分配路线。
stAERouteEx	设置 ISP 第一次启动时 AE 的扩展曝光分配路线。
stAERouteSF	设置 ISP 第一次启动时 AE 短帧的曝光分配路线，仅在 WDR 模式下使用。
stAERouteSFEx	设置 ISP 第一次启动时 AE 短帧的扩展曝光分配路线，仅在 WDR 模式下使用。

【注意事项】

- 在 ISP 启动前，设置初始 AWB 增益和 CCM 系数，可改善前后帧颜色的一致性。
- 在 ISP 启动前，设置初始的 AE 曝光路线，可以实现帧率改变后，AE 算法能够将初始设置的 AE route 保留并自动生效。

【相关数据类型及接口】

[ISP_SNS_OBJ_S](#)

[ISP_QUICK_START_PARAM_S](#)

ISP_SNS_MIRRORFLIP_TYPE_E

【说明】

定义 sensor mirror-flip 枚举。

【定义】

```
typedef enum hiISP_SNS_MIRRORFLIP_TYPE_E {  
    ISP_SNS_NORMAL        = 0,  
    ISP_SNS_MIRROR        = 1,  
    ISP_SNS_FLIP          = 2,  
    ISP_SNS_MIRROR_FLIP = 3,  
    ISP_SNS_BUTT  
} ISP_SNS_MIRRORFLIP_TYPE_E;
```

【成员】



成员名称	描述
ISP_SNS_NORMAL	sensor 正常输出
ISP_SNS_MIRROR	sensor mirror 输出
ISP_SNS_FLIP	sensor flip 输出
ISP_SNS_MIRROR_FLIP	Sensor mirror-flip 输出

【注意事项】

无。

【相关数据类型及接口】

[ISP_SNS_OBJ_S](#)

ISP_SNS_BUS_EX_S

【说明】

定义 sensor 通讯协议的扩展结构体。

【定义】

```
typedef struct hiISP_SNS_BUS_EX_S {  
    char bus_addr;  
} ISP_SNS_BUS_EX_S;
```

【成员】

成员名称	描述
bus_addr	指定该 pipe 绑定的 sensor 对应到 serdes 的设备地址。

【注意事项】

无。

【相关数据类型及接口】

[ISP_SNS_OBJ_S](#)



ISP_SNS_OBJ_S

【说明】

定义指向 sensor 的对象。

【定义】

```
typedef struct hiISP_SNS_OBJ_S
{
    HI_S32 (*pfnRegisterCallback)(VI_PIPE ViPipe, ALG_LIB_S* pstAeLib, ALG_LIB_S*
pstAwbLib);
    HI_S32 (*pfnUnRegisterCallback)(VI_PIPE ViPipe, ALG_LIB_S* pstAeLib, ALG_LIB_S*
pstAwbLib);
    HI_S32 (*pfnSetBusInfo)(VI_PIPE ViPipe, ISP_SNS_COMMBUS_U unSNSBusInfo);
    HI_S32 (*pfnSetBusExInfo)(VI_PIPE ViPipe, ISP_SNS_BUS_EX_S *pstSerdesInfo);
    HI_VOID (*pfnStandby)(VI_PIPE ViPipe);
    HI_VOID (*pfnRestart)(VI_PIPE ViPipe);
    HI_VOID (*pfnMirrorFlip)(VI_PIPE ViPipe, ISP_SNS_MIRRORFLIP_TYPE_E eSnsMirrorFlip);
    HI_S32 (*pfnWriteReg)(VI_PIPE ViPipe, HI_S32 s32Addr, HI_S32 s32Data);
    HI_S32 (*pfnReadReg)(VI_PIPE ViPipe, HI_S32 s32Addr);
    HI_S32 (*pfnSetInit)(ISP_DEV IspDev, ISP_INIT_ATTR_S *pstInitAttr);
} ISP_SNS_OBJ_S;
```

【成员】

成员名称	描述
pfnRegisterCallback	指向 sensor 注册函数的指针
pfnUnRegisterCallback	指向 sensor 反注册函数的指针
pfnSetBusInfo	指向 sensor 与 I2C/SPI 的绑定关系函数的指针
pfnSetBusExInfo	指向 sensor 扩展通讯函数的指针
pfnStandby	指向 sensor standby 的函数指针
pfnRestart	指向 sensor Restart 的函数指针
pfnMirrorFlip	指向 sensor Mirror Flip
pfnWriteReg	指向 sensor 写寄存器函数的指针



成员名称	描述
pfnReadReg	指向 sensor 读寄存器函数的指针
pfnSetInit	指向 sensor 设置 AW/AWB 初始化参数函数的指针

【注意事项】

ISP_SNS_OBJ_S 是

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 为了区分不同 sensor 库新增的结构体，用法如下：

```
ISP_SNS_OBJ_S stSnsXXXObj =
{
    pfnRegisterCallback    = sensor_register_callback,
    pfnUnRegisterCallback  = sensor_unregister_callback,
    pfnStandby              = xxx_standby,
    pfnRestart              = xxx_restart,
    pfnMirrorFlip           = xxx_Mirror_Flip,
    pfnWriteReg             = xxx_write_register,
    pfnReadReg              = xxx_read_register,
    pfnSetInit              = sensor_set_init,
};
```

【相关数据类型及接口】

无

ISP_SNS_STATE_S

【说明】

定义 sensor 相关全局变量参数结构体。

【定义】

```
typedef struct hiISP_SNS_STATE_S
{
    HI_BOOL    bInit;                /* HI_TRUE: Sensor init */
    HI_BOOL    bSynclnit;            /* HI_TRUE: Sync Reg init */
    HI_U8      u8ImgMode;
    HI_U8      u8Hdr;
```



```

WDR_MODE_E enWDRMode;
ISP_SNS_REGS_INFO_S astRegsInfo[ISP_SNS_SAVE_INFO_MAX];
HI_U32      au32FL[ISP_SNS_SAVE_INFO_MAX];
HI_U32      u32FLStd; /* FullLines std */
HI_U32      au32WDRIntTime[WDR_MAX_FRAME_NUM];
HI_U32      au32SensorWBGain[ISP_BAYER_CHN_NUM];
} ISP_SNS_STATE_S;

```



说明

宏定义如下：

```
#define ISP_SNS_SAVE_INFO_MAX 2
```

【成员】

成员名称	描述
bInit	Sensor 初始化状态标记。
bSyncInit	Sensor 寄存器同步初始化状态标记。
u8ImgMode	Sensor 分辨率模式设置。
u8Hdr	记录是否为 HDR 模式，仅在 Hi3559AV100 中支持。
enWDRMode	Sensor WDR 模式设置。
astRegsInfo[ISP_SNS_SAVE_INFO_MAX]	Sensor 寄存器状态，astRegsInfo[0]表示当前帧 sensor 寄存器状态，astRegsInfo[1]表示上一帧 sensor 寄存器状态。
au32FL[ISP_SNS_SAVE_INFO_MAX]	记录实际生效的一帧的总行数，au32FL[0]表示当前帧的行数，au32FL[1]表示上一帧的行数。
u32FLStd	基准帧率下的一帧的总行数。
au32WDRIntTime[WDR_MAX_FRAME_NUM]	WDR 模式下曝光时间，au32WDRIntTime[0]表示 VS 帧曝光时间，au32WDRIntTime[1]表示 S 帧曝光时间，au32WDRIntTime[2]表示 M 帧曝光时间，au32WDRIntTime[3]表示 L 帧曝光时间。
au32SensorWBGain	Sensor 应配置的 AWB 增益。8bit 小数精度，取值范围：[0x0,0xFFF]。增益按照 RGGB 顺序排



成员名称	描述
	列。

【注意事项】

- [ISP_SNS_STATE_S](#) 是 Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 保存 cmos.c 中全局变量而新增的结构体，多路 ISP 同时加载 sensor 库，sensor 状态相关的全局变量以 ViPipe 进行区别。
- 默认 AWB 增益在 ISP 配置，用户不需要关注 au32SensorWBGain 参数。
- 实现 AWB 增益在 Sensor 配置，需要对回调函数 pfn_cmos_get_awb_gains 赋值，且将 au32SensorWBGain 参数的 AWB 增益值配置到对应的 Sensor 寄存器。

【相关数据类型及接口】

无

ISP_AWB_ALG_E

【说明】

定义 AWB 算法类型。

【定义】

```
typedef enum hiISP_AWB_ALG_E
{
    ALG_AWB          = 0,
    ALG_AWB_SPEC     = 1,
    ALG_BUTT
} ISP_AWB_ALG_E;
```

【成员】

成员名称	描述
ALG_AWB	AWB 算法
ALG_AWB_SPEC	机器学习 AWB 算法



【注意事项】

ISP_AWB_ALG_E 是为了区分原有的 AWB 算法和机器学习 AWB 算法而新增的 enum 体，原有的 AWB 算法与机器学习 AWB 算法在同一个算法库中，可以使用该 Enum 体实时切换算法。

【相关数据类型及接口】

ISP_WB_ATTR_S

ISP_SPECAWB_ATTR_S

【说明】

定义机器学习 AWB 算法标定结果结构体。

【定义】

```
typedef struct hiISP_SPECAWB_ATTR_S
{
    ISP_SPECKCWB_S    stWBCenter;
    ISP_SPECKCWB_S    stWBMin;
    ISP_SPECKCWB_S    stWBMax;
    ISP_SPECKCWB_S    stLogFact;
    HI_S32             s32LimitFactMin;
    HI_S32             s32LimitFactMax;
    HI_S32             s32LimitFactLimit;
    HI_S32             s32FactOffset;
    HI_S32             s32BFact;
    HI_S32             s32ACntMin;
    ISP_SPECKCWB_S    stWBNeutral;
    HI_U8              bLastIsFlash;
    HI_S16             s16BvFlashELimit;
    HI_S32             s32APercent;
    ISP_SPECKCWBS16_S stWBCnvTbl[64][64];
    HI_U8              u8ElementNum;
    ISP_SPECAWB_FACTTBL_ELEMENT_S stFactElement[7];
    ISP_SPECAWB_KELVIN_DBB_MAP_S    stKelvinDBBTbl[64][64];
    ISP_SPECAWB_BBL_TBL_S           stBlackBodyTbl[40];
    HI_U16                          u16Fno;
}ISP_SPECAWB_ATTR_S;
```



【成员】

成员名称	描述
stWBCenter	Wb gain 二维坐标中心值
stWBMin	Wb gain 二维坐标最小值
stWBMax	Wb gain 二维坐标最大值
stLogFact	用于搜索 WB gain 值表
s32LimitFactMin	AWB 统计信息阈值最小值, 取值范围: [0, s32LimitFactMax]
s32LimitFactMax	AWB 统计信息阈值最大值, 取值范围: [s32LimitFactMin, 0xFFFF]
s32LimitFactLimit	AWB 统计信息限定值, 取值范围: [0x600, 0x4000]
s32FactOffset	AWB 统计信息权重 offset, 取值范围: [0, 0x800]
s32BFact	AWB 统计信息 B value 权重, 取值范围: [0x399, 0x500]
s32ACntMin	AWB 分块内白点有效 count 阈值, 取值范围: [0, 0x20]
stWBNeutral	当 AWB 无法计算出色温时默认使用的 Gain 值
bLastIsFlash	光源分布表中最后一张表是否作为闪光灯情况使用, 取值为: 1
s16BvFlashELimit	当 Bv 值低于该阈值时并且闪光灯使能则最后一张表作为闪光灯使用, 取值范围: [-32768, 32767]
s32APercent	统计信息中的 A value 权重, 取值范围: [0, 100]
stWBConvTbl	默认色温转换表对应的 wb gain 值表
u8ElementNum	光源分布表数量, 取值为: 7
stFactElement	光源分布表
stKelvinDBBTbl	Kelvin, DeltaBlackBody 坐标表
stBlackBodyTbl	普朗克曲线坐标
u16Fno	光圈大小(F1.2=12,F2.8=28,F4.6=46.....), 取值范围: [10, 100]



【注意事项】

- 该结构体处 u16Fno 以外的变量都由标定生成，客户需要根据自己所使用的镜头光圈修改 u16Fno 的值使算法得到正确的环境亮度。
- 该结构体成员除 u16Fno 外，其他均为标定生成，不推荐客户手动调整参数。

【相关数据类型及接口】

- [HI_MPI_ISP_SetSpecAwbAttr](#)
- [HI_MPI_ISP_GetSpecAwbAttr](#)

ISP_SPECAWB_CONTROL_ATTR_S

【说明】

定义机器学习 AWB 控制参数结构体。

【定义】

```
typedef struct hiISP_SPECAWB_CONTROL_ATTR_S
{
    HI_S16 s16BlendHighBvThresh;
    HI_U16 u16BlendHighBvWt;
    HI_S16 s16BlendLowBvThresh;
    HI_U16 u16BlendLowBvWt;
} ISP_SPECAWB_CONTROL_ATTR_S;
```

【成员】

成员名称	描述
s16BlendHighBvThresh	高亮环境的亮度值，取值范围：[-32768, 32767]
u16BlendHighBvWt	高亮环境，弱标定 AWB 结果的权重系数。 取值范围：[0, 2048]
s16BlendLowBvThresh	低照环境的亮度值。 取值范围：[-32768, s16BlendHighBvThresh)
u16BlendLowBvWt	低照环境，弱标定 AWB 结果的权重系数。 取值范围：[0, 2048]



【注意事项】

弱标定 AWB 结果，指放宽标定系数对 AWB 的影响，计算得到的 AWB 结果。在 ColorShading 较大时，推荐适当增大弱标定的权重。

【相关数据类型及接口】

- [HI_MPI_ISP_SetSpecAwbControlAttr](#)
- [HI_MPI_ISP_GetSpecAwbControlAttr](#)

ISP_SPECKCWB_S

【说明】

定义 WB gain 值结构体。

【定义】

```
typedef struct hiISP_SPECKCWB_S
{
    HI_S32    s32RGain;
    HI_S32    s32BGain;
} ISP_SPECKCWB_S;
```

【成员】

成员名称	描述
s32RGain	R Gain, 取值范围: [-2147483648, 2147483647]
s32BGain	B Gain, 取值范围: [-2147483648, 2147483647]

【注意事项】

无。

【相关数据类型及接口】

无

ISP_SPECKCWBS16_S

【说明】

定义 WB gain 值结构体(16bit 位宽)。



【定义】

```
typedef struct hiISP_SPECKCWBS16_S
{
    HI_S16    s16RGain;
    HI_S16    s16BGain;
} ISP_SPECKCWBS16_S;
```

【成员】

成员名称	描述
s16RGain	R Gain(16bit), 取值范围: [-32768, 32767]
s16BGain	B Gain(16bit), 取值范围: [-32768, 32767]

【注意事项】

无。

【相关数据类型及接口】

无

ISP_SPECAWB_FACTTBL_ELEMENT_S

【说明】

定义光源分布结构体。

【定义】

```
typedef struct hiISP_SPECAWB_FACTTBL_ELEMENT_S {
    HI_S16    s16Bv;
    HI_U8     u8FactTbl[64][64];
} ISP_SPECAWB_FACTTBL_ELEMENT_S;
```

【成员】

成员名称	描述
s16Bv	光源分布表对应的 bv 值, 取值范围: [-32768, 32767]
u8FactTbl[64][64]	光源分布表权重, 取值范围: [0, 255]



【注意事项】

s16Bv 为标定生成的 Bv 值，不是环境 Bv。

【相关数据类型及接口】

[ISP_SPECAWB_ATTR_S](#)

ISP_SPECAWB_KELVIN_DBB_MAP_S

【说明】

定义色温/DeltaBlackBody 分布结构体。

【定义】

```
typedef struct hiISP_SPECAWB_KELVIN_DBB_MAP_S{  
    HI_S16 s16Kelvin;  
    HI_S16 s16DBB;  
} ISP_SPECAWB_KELVIN_DBB_MAP_S;
```

【成员】

成员名称	描述
s16Kelvin	色温，取值范围：[-32768, 32767]
s16DBB	DeltaBlackBody 值，取值范围：[-32768, 32767]

【注意事项】

无。

【相关数据类型及接口】

[ISP_SPECAWB_ATTR_S](#)

ISP_SPECAWB_BBL_TBL_S

【说明】

定义普朗克曲线/Gain 值对应表结构体。

【定义】

```
typedef struct hiISP_SPECAWB_BBL_TBL_S{
```



```
HI_S16 s16Kelvin;  
HI_S16 s16Wbr;  
HI_S16 s16Wbb;  
} ISP_SPECAWB_BBL_TBL_S;
```

【成员】

成员名称	描述
s16Kelvin	色温, 取值范围: [-32768, 32767]
s16Wbr	R Gain 值, 取值范围: [-32768, 32767]
s16Wbb	B Gain 值, 取值范围: [-32768, 32767]

【注意事项】

无。

【相关数据类型及接口】

[ISP_SPECAWB_ATTR_S](#)

ISP_SPECAWB_CAA_CONTROL_S

【说明】

定义色温转换表控制结构体。

【定义】

```
typedef struct hiISP_SPECAWB_CAA_CONTROL_S {  
    ISP\_SPECAWB\_CAA\_TBL\_S stControl[3];  
} ISP_SPECAWB_CAA_CONTROL_S;
```

【成员】

成员名称	描述
stControl	色温转换 lut,共有 3 组

【注意事项】

无。



【相关数据类型及接口】

[HI_MPI_ISP_SetCaaControlAttr](#)

ISP_SPECAWB_CAA_TBL_S

【说明】

定义单组色温转换表详细控制信息结构体。

【定义】

```
typedef struct hiISP_SPECAWB_CAA_TBL_S
{
    HI_U8    u8Enable;
    HI_S16   s16Bv;
    ISP\_SPECAWB\_CAA\_CONVERSION\_S stKelvinCon[SPECAWB_KEVIN_CONVER_MAX_NUM];
    ISP\_SPECKCWB516\_S stWBCnvTbl[64][64];
} ISP_SPECAWB_CAA_TBL_S;
```

【成员】

成员名称	描述
u8Enable	该组色温表是否使能, 取值范围: [0,1]
s16Bv	该组色温表对应的 Bv 值, 取值范围: [-32768 , 32767]
stKelvinCon	原色温, 目的色温, 支持 8 组
stWBCnvTbl	实际使用的色温转换 lut

【注意事项】

- s16Bv 为用户希望色温转换表覆盖的亮度范围, 推荐用户由亮到按设定 3 组 bv 值, 若当前环境 bv 值(通过 [ISP_WB_INFO_S](#) 中的成员 s16Bv 得到)落在用户设定的两组 bv 值中间, 则使用 bv 对应的两组色温转换表的插值结果作为最终结果。
- stKelvinCon 的目的是与 PQ 工具交互使用, 并不直接作用于 Awb 算法, stWBCnvTbl 为 AWB 库真正使用的色温转换 lut。

【相关数据类型及接口】

[ISP_SPECAWB_CAA_CONTROL_S](#)



ISP_SPECAWB_CAA_CONVERSION_S

【说明】

定义原始色温，目的色温信息结构体。

【定义】

```
typedef struct hiISP_SPECAWB_CAA_CONVERSION_S {  
    int s32SrcKelvin;  
    int s32DstKelvin;  
} ISP_SPECAWB_CAA_CONVERSION_S;
```

【成员】

成员名称	描述
s32SrcKelvin	原始色温, 取值范围: [-2147483648, 2147483647]
s32DstKelvin	目的色温, 取值范围: [-2147483648, 2147483647]

【注意事项】

- s32SrcKelvin, s32DstKelvin 由用户通过 PQ tools 生成。
- 暂不支持此功能。

【相关数据类型及接口】

[ISP_SPECAWB_CAA_CONTROL_S](#)

ISP_IR_STATUS_E

【说明】

定义设备当前的红外状态。

【定义】

```
typedef enum hiISP_IR_STATUS_E  
{  
    ISP_IR_STATUS_NORMAL = 0,  
    ISP_IR_STATUS_IR      = 1,  
    ISP_IR_BUTT  
} ISP_IR_STATUS_E;
```



【成员】

成员名称	描述
ISP_IR_STATUS_NORMAL	设备当前为普通状态（非红外状态）。
ISP_IR_STATUS_IR	设备当前为红外状态。

【注意事项】

无

【相关数据类型及接口】

无

ISP_IR_SWITCH_STATUS_E

【说明】

定义设备的红外切换状态。

【定义】

```
typedef enum hiISP_IR_SWITCH_STATUS_E
{
    ISP_IR_SWITCH_NONE      = 0,
    ISP_IR_SWITCH_TO_NORMAL = 1,
    ISP_IR_SWITCH_TO_IR     = 2,
    ISP_IR_SWITCH_BUTT
} ISP_IR_SWITCH_STATUS_E;
```

【成员】

成员名称	描述
ISP_IR_SWITCH_NONE	设备不切换红外状态。
ISP_IR_SWITCH_TO_NORMAL	设备切换为普通状态（非红外状态）。
ISP_IR_SWITCH_TO_IR	设备切换为红外状态。

【注意事项】



无

【相关数据类型及接口】

无

ISP_RAW_POS_E

【说明】

定义获取 raw 位置。

【定义】

```
typedef enum hiISP_RAW_POS_E {  
    RAW_POS_DEMOSAIC    = 0,  
    RAW_POS_BUTT  
} ISP_RAW_POS_E;
```

【成员】

成员名称	描述
RAW_POS_DEMOSAIC	支持在 Demosaic 模块之前获取 raw。

【注意事项】

仅 Hi3559AV100 支持。

【相关数据类型及接口】

无

ISP_RAW_POS

【说明】

定义获取 RAW 的相关配置信息。

【定义】

```
typedef struct hiISP_RAW_POS {  
    ISP_RAW_POS_E enRawPos;  
    HI_BOOL bEnable;  
} ISP_RAW_POS;
```




【成员】

成员名称	描述
enRawPos	raw 的位置
bEnable	使能获取 raw。 HI_FALSE: 关闭获取 raw; HI_TRUE: 使能获取 raw。

【注意事项】

- 当不用获取 Raw 数据时，需要设置 bEable 为 HI_FALSE。
- 仅 Hi3559AV100 支持。

【相关数据类型及接口】

无

ISP_IR_AUTO_ATTR_S

【说明】

定义红外自动切换属性。

【定义】

```
typedef struct hiISP_IR_AUTO_ATTR_S
{
    HI_BOOL bEnable;
    HI_U32  u32Normal2IrIsoThr;
    HI_U32  u32Ir2NormalIsoThr;
    HI_U32  u32RGMax;
    HI_U32  u32RGMin;
    HI_U32  u32BGMax;
    HI_U32  u32BGMin;
    ISP_IR_STATUS_E enIrStatus;
    ISP_IR_SWITCH_STATUS_E enIrSwitch;
} ISP_IR_AUTO_ATTR_S;
```

【成员】



成员名称	描述
bEnable	红外自动切换使能。 HI_FALSE: 关闭; HI_TRUE: 使能。
u32Normal2IrIsoThr	从普通状态切换到红外状态的 ISO 阈值。当实际生效的 ISO 大于此阈值时, 系统需要切换到红外状态。 取值范围: [0, 0xFFFFFFFF]。
u32Ir2NormalIsoThr	从红外状态切换到普通状态的 ISO 阈值。当实际生效的 ISO 小于此阈值时, 系统需要切换到普通状态。 取值范围: [0, 0xFFFFFFFF]。
u32RGMax	红外状态下的 R/G 最大值。实际图像的 R/G 大于此参数时, 系统需要切换到普通状态。4.8 格式。 取值范围: [0, 0xFFF]。
u32RGMin	红外状态下的 R/G 最小值。实际图像的 R/G 小于此参数时, 系统需要切换到普通状态。4.8 格式。 取值范围: [0, u32RGMax]。
u32BGMax	红外状态下的 B/G 最大值。实际图像的 B/G 大于此参数时, 系统需要切换到普通状态。4.8 格式。取值范围: [0, 0xFFF]。
u32BGMin	红外状态下的 B/G 最小值。实际图像的 B/G 小于此参数时, 系统需要切换到普通状态。4.8 格式。 取值范围: [0, u32BGMax]。
enIrStatus	设备当前的红外状态。应配置为设备实际的红外状态, 需要用户保证状态的正确性。
enIrSwitch	设备的红外切换状态, 为只读。

【注意事项】

- u32Normal2IrIsoThr/u32Ir2NormalIsoThr/ u32RGMax/u32RGMin/ u32BGMax/u32BGMin 的配置值和 sensor/镜头/滤光片/红外灯相关。



- u32RGMax/u32RGMin/ u32BGMax/u32BGMin 的配置值可参考 mpp/sample/ir_auto/sample_ir_auto.c 中标定流程生成。

【相关数据类型及接口】

无



3 AE

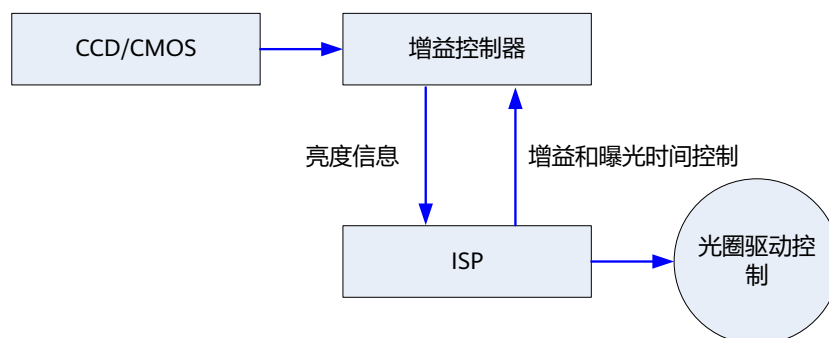
说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

3.1 概述

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

图3-1 AE 模块工作流程图



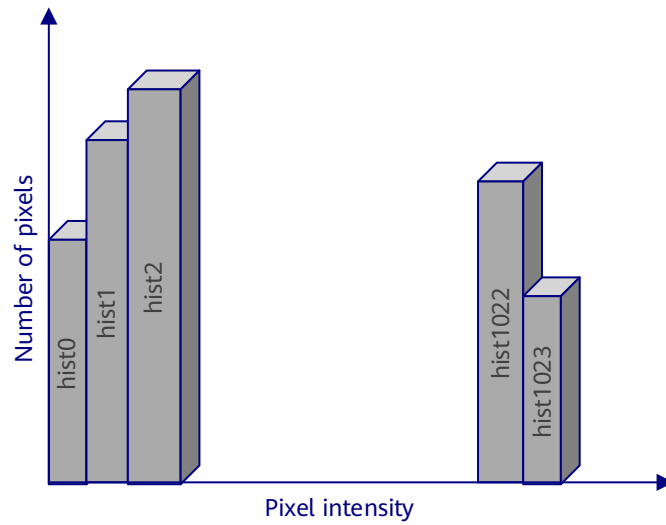
3.2 重要概念

- 曝光时间：sensor 积累电荷的时间，是 sensor pixel 从开始曝光到电量被读出的这段时间。
- 曝光增益：对 sensor 的输出电荷的总的放大系数，一般有数字增益和模拟增益，模拟增益引入的噪声会稍小，所以一般优先用模拟增益。
- 光圈：光圈是镜头中可以改变中间孔大小的机械装置。
- 抗闪烁：由于电灯电源工频与 sensor 的帧率不匹配而导致的画面闪烁，一般通过限定曝光时间和修改 sensor 的帧率来达到抗闪烁的效果。

3.3 功能描述

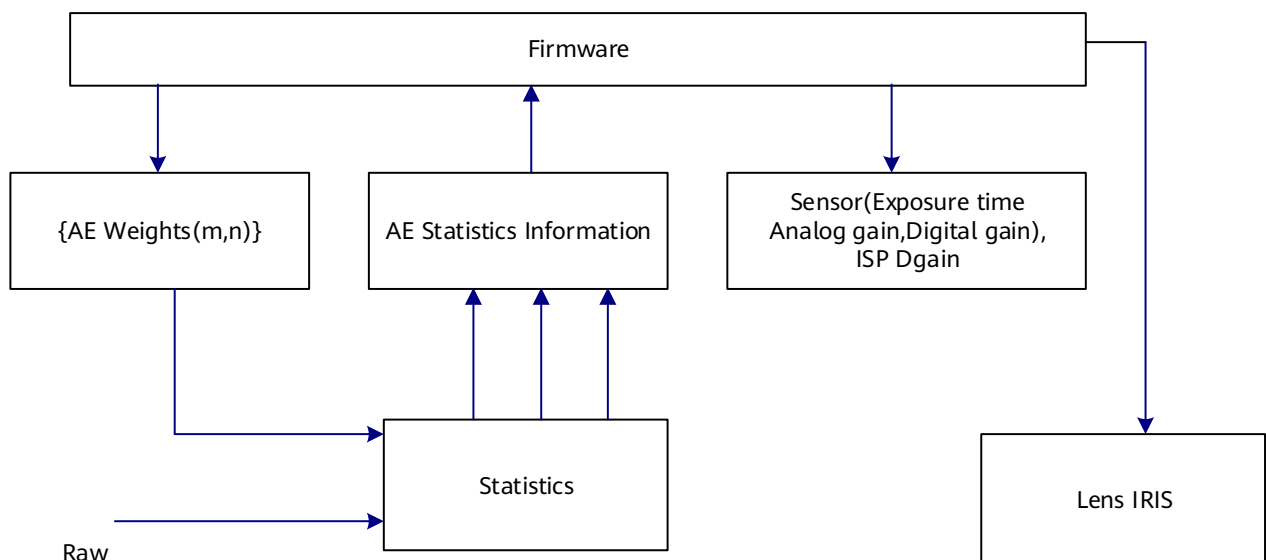
AE 模块主要有 ISP 的 AE 统计信息模块及 AE 控制策略的 AE 算法 Firmware 两部分组成。ISP 的 AE 统计信息模块主要是提供 sensor 输入数据的亮度信息统计。其提供的统计信息包括直方图和平均值，可同时提供整幅图像的 1024 段的直方图和 R/Gr/Gb/B 四分量平均值统计信息，还可提供将整幅图像分成 MxN 区块的每个区块的 R/Gr/Gb/B 四分量平均值统计信息，具体如图 3-2 所示。

图3-2 AE 1024 段统计信息直方图



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

图3-3 AE 工作原理图





3.4 API 参考

3.4.1 AE 库接口

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

- [HI_MPI_AE_Register](#): 向 ISP 注册 AE 库。
- [HI_MPI_AE_UnRegister](#): 向 ISP 反注册 AE 库。
- [HI_MPI_AE_SensorRegCallBack](#): AE 库提供的 sensor 注册的回调接口。
- [HI_MPI_AE_SensorUnRegCallBack](#): AE 库提供的 sensor 反注册的回调接口。

HI_MPI_AE_Register

【描述】

向 ISP 注册 AE 库。

【语法】

```
HI_S32 HI_MPI_AE_Register(VI_PIPE ViPipe, ALG\_LIB\_S *pstAeLib);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件: hi_comm_isp.h、 mpi_ae.h



- 库文件：libisp.a、lib_hiaa.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](#)，以实现上海海思 AE 库向 ISP 库注册的功能。
- AE 库可以注册多个实例。
- 此接口不支持多进程操作。

【举例】

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

【相关主题】

无

HI_MPI_AE_UnRegister

【描述】

向 ISP 反注册 AE 库。

【语法】

```
HI_S32 HI_MPI_AE_UnRegister(VI_PIPE ViPipe, ALG\_LIB\_S *pstAeLib);
```

【参数】

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

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_comm_isp.h、mpi_ae.h
- 库文件：libisp.a、lib_hiaa.a

【注意】

- 该接口调用了 ISP 库提供的 AE 反注册回调接口 [HI_MPI_ISP_AELibUnRegCallBack](#)，以实现 AE 向 ISP 库反注册的功能。
- 此接口不支持多进程操作。

【举例】

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

【相关主题】

无

HI_MPI_AE_SensorRegCallBack

【描述】

AE 库提供的 sensor 注册的回调接口。

【语法】

```
HI_S32 HI_MPI_AE_SensorRegCallBack(VI_PIPE ViPipe, ALG\_LIB\_S *pstAeLib,  
ISP\_SNS\_ATTR\_INFO\_S *pstSnsAttrInfo, AE\_SENSOR\_REGISTER\_S *pstRegister);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAeLib	AE 算法库结构体指针。	输入
pstSnsAttrInfo	向 AE 注册的 Sensor 的属性。	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 。

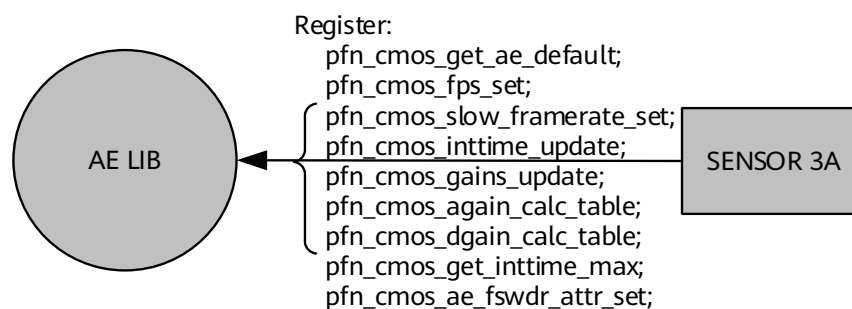
【需求】

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

【注意】

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

图3-4 AE 库与 sensor 库间的接口





【举例】

```
ALG_LIB_S stLib;
AE_SENSOR_REGISTER_S stAeRegister;
ISP_SNS_ATTR_INFO_S stSnsAttrInfo;
AE_SENSOR_EXP_FUNC_S *pstExpFuncs = &stAeRegister.stSnsExp;
memset(pstExpFuncs, 0, sizeof(AE_SENSOR_EXP_FUNC_S));
pstExpFuncs->pfn_cmos_get_ae_default = cmos_get_ae_default;
pstExpFuncs->pfn_cmos_fps_set = cmos_fps_set;
pstExpFuncs->pfn_cmos_slow_framerate_set = cmos_slow_framerate_set;
pstExpFuncs->pfn_cmos_inttime_update = cmos_inttime_update;
pstExpFuncs->pfn_cmos_gains_update = cmos_gains_update;
pstExpFuncs->pfn_cmos_again_calc_table = cmos_again_calc_table;
pstExpFuncs->pfn_cmos_dgain_calc_table = cmos_dgain_calc_table;
pstExpFuncs->pfn_cmos_get_inttime_max = cmos_get_inttime_max;
pstExpFuncs->pfn_cmos_ae_fswdr_attr_set = cmos_ae_fswdr_attr_set;
pstExpFuncs->pfn_cmos_ae_quick_start_status_set = cmos_ae_quick_start_status_set;

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

【相关主题】

无

HI_MPI_AE_SensorUnRegCallBack

【描述】

AE 库提供的 sensor 反注册的回调接口。



【语法】

```
HI_S32 HI_MPI_AE_SensorUnRegCallBack(VI_PIPE ViPipe, ALG_LIB_S *pstAeLib, SENSOR_ID  
SensorId);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
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、lib_hiaa.a

【注意】

- SensorId 是 sensor 库中自定义的值，主要用于校对向 ISP 反注册的 sensor 和向 3A 反注册的 sensor 是否为同一个 sensor。
- 此接口不支持多进程操作。

【举例】

```
ALG_LIB_S stLib;  
VI_PIPE ViPipe = 0;  
stLib.s32Id = 0;  
strcpy(stLib.acLibName, HI_AE_LIB_NAME);  
s32Ret = HI_MPI_AE_SensorUnRegCallBack(ViPipe, &stLib, IMX104_ID);  
if (s32Ret)
```



```
{  
    printf("sensor register callback function to ae lib failed!\n");  
    return s32Ret;  
}
```

【相关主题】

无

3.4.2 AE 控制模块

曝光控制接口：

- [HI_MPI_ISP_SetExposureAttr](#)：设置 AE 曝光属性。
- [HI_MPI_ISP_GetExposureAttr](#)：获取 AE 曝光属性。
- [HI_MPI_ISP_SetWDRExposureAttr](#)：设置 WDR 模式下的 AE 曝光属性。
- [HI_MPI_ISP_GetWDRExposureAttr](#)：获取 WDR 模式下的 AE 曝光属性。
- [HI_MPI_ISP_SetHDXposureAttr](#)：设置 HDR 模式下的 AE 曝光属性。
- [HI_MPI_ISP_GetHDXposureAttr](#)：获取 HDR 模式下的 AE 曝光属性。
- [HI_MPI_ISP_SetSmartExposureAttr](#)：设置智能模式下的 AE 曝光属性。
- [HI_MPI_ISP_GetSmartExposureAttr](#)：获取智能模式下的 AE 曝光属性。
- [HI_MPI_ISP_SetAERouteAttr](#)：设置 AE 曝光分配策略属性。
- [HI_MPI_ISP_GetAERouteAttr](#)：获取 AE 曝光分配策略属性。
- [HI_MPI_ISP_SetAERouteAttrEx](#)：设置 AE 曝光分配扩展属性，支持分别设置 AE 分配策略中的 sensor 模拟增益，sensor 数字增益和 ISP 数字增益。
- [HI_MPI_ISP_GetAERouteAttrEx](#)：获取 AE 曝光分配策略扩展属性。
- [HI_MPI_ISP_SetAERouteSFAttr](#)：WDR 模式下，设置 AE 短帧的曝光分配策略属性。
- [HI_MPI_ISP_GetAERouteSFAttr](#)：获取 AE 短帧曝光分配策略属性。
- [HI_MPI_ISP_SetAERouteSFAttrEx](#)：WDR 模式下，设置 AE 短帧的曝光分配策略扩展属性。
- [HI_MPI_ISP_GetAERouteSFAttrEx](#)：获取 AE 短帧曝光分配策略扩展属性。
- [HI_MPI_ISP_QueryExposureInfo](#)：获取 AE 内部状态信息。
- [HI_MPI_ISP_SetExpConvert](#)：设置不同帧率下等曝光量转换的相关属性。



- [HI_MPI_ISP_GetExpConvert](#): 获取不同帧率下等曝光量转换结果相关的曝光参数属性。

HI_MPI_ISP_SetExposureAttr

【描述】

设定 AE 曝光属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetExposureAttr(VI_PIPE ViPipe, const ISP\_EXPOSURE\_ATTR\_S *pstExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstExpAttr	AE 曝光属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- AE 曝光控制类型为自动时，曝光时间，曝光增益都由 AE 算法自动控制，可以通过配置自动曝光属性结构体 stAuto 里面的参数得到不同的曝光效果。
- AE 曝光控制类型为手动时，可以通过配置手动曝光属性结构体 stManual 控制使能类型（曝光时间使能、sensor 模拟增益使能、sensor 数字增益使能、ISP 数字增益使能）及相应的曝光参数（曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益）。



- AE 曝光控制类型为自动时，配置手动曝光属性的参数无效。同理，AE 曝光控制类型为手动时，配置自动曝光属性的参数无效。
- AE 曝光控制类型为手动时，若曝光参数设置超出最大（小）值，将使用 sensor 支持的最大（小）值代替。
- 无论是自动曝光还是手动曝光，曝光时间的单位为微秒(us)，曝光增益的单位为 10bit 精度的倍数，即 1024 代表 1 倍，2048 代表 2 倍等。
- WDR 模式下，优先帧设置为长帧时，优先根据长帧的曝光路线进行曝光，其中 2 合 1WDR 模式下当增益分开配置时，短帧曝光路线根据长帧的曝光参数进行调整；优先帧设置为短帧时，优先根据短帧的曝光路线进行曝光，其中 2 合 1WDR 模式下当增益分开配置时，长帧曝光路线根据短帧的曝光参数进行调整。
- 2 合 1WDR 模式下，设置增益分开配置，若 sensor 支持长短帧不同增益，可实现长短帧不同的 sensor 模拟增益、sensor 数字增益、WDR 增益，若 sensor 不支持长短帧不同增益，可实现长短帧不同的 WDR 增益。

【举例】

自动曝光属性设置：

```
VI_PIPE ViPipe = 0;
ISP_EXPOSURE_ATTR_S stExpAttr;

HI_MPI_ISP_GetExposureAttr(ViPipe, &stExpAttr);

stExpAttr.bByPass = HI_FALSE;
stExpAttr.enPriorFrame = LONG_FRAME;
stExpAttr.bAEGainSepCfg = HI_FALSE;
stExpAttr.enOpType = OP_TYPE_AUTO;
stExpAttr.stAuto.stExpTimeRange.u32Max = 40000;
stExpAttr.stAuto.stExpTimeRange.u32Min = 10;
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);

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

stExpAttr.stAuto.enAESTrategyMode = AE_EXP_HIGHLIGHT_PRIOR;
stExpAttr.stAuto.u16HistRatioSlope = 0x100;
stExpAttr.stAuto.u8MaxHistOffset = 0x40;
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);
```



```
stExpAttr.stAuto.stAntiflicker.bEnable = HI_TRUE;
stExpAttr.stAuto.stAntiflicker.u8Frequency = 50;
stExpAttr.stAuto.stAntiflicker.enMode = ISP_ANTIFLICKER_NORMAL_MODE;
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);

stExpAttr.stAuto.stAEDelayAttr.u16BlackDelayFrame = 10;
stExpAttr.stAuto.stAEDelayAttr.u16WhiteDelayFrame = 0;
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);
HI_U8 ij;
HI_U8 u8Weighttable[15][17]={1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                                };
for (i = 0; i < AE_ZONE_ROW; i++)
{
    for (j = 0; j < AE_ZONE_COLUMN; j++)
    {
        stExpAttr.stAuto.au8Weight[i][j] = u8Weighttable[i][j];
    }
}
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);
```

手动曝光属性设置：

```
VI_PIPE ViPipe = 0;
```




```
ISP_EXPOSURE_ATTR_S stExpAttr;  
  
HI_MPI_ISP_GetExposureAttr(ViPipe, &stExpAttr);  
stExpAttr.bByPass = HI_FALSE;  
stExpAttr.enOpType = OP_TYPE_MANUAL;  
stExpAttr.stManual.enAGainOpType = OP_TYPE_MANUAL;  
stExpAttr.stManual.enDGainOpType = OP_TYPE_MANUAL;  
stExpAttr.stManual.enISPDGainOpType = OP_TYPE_MANUAL;  
stExpAttr.stManual.enExpTimeOpType = OP_TYPE_MANUAL;  
stExpAttr.stManual.u32AGain = 0x400;  
stExpAttr.stManual.u32DGain = 0x400;  
stExpAttr.stManual.u32ISPDGain = 0x400;  
stExpAttr.stManual.u32ExpTime= 0x40000;  
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr)
```

【相关主题】

[HI_MPI_ISP_GetExposureAttr](#)

HI_MPI_ISP_GetExposureAttr

【描述】

获取 AE 曝光属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetExposureAttr  
(VI_PIPE ViPipe, ISP\_EXPOSURE\_ATTR\_S *pstExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstExpAttr	AE 曝光属性结构体指针。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetExposureAttr](#)

HI_MPI_ISP_SetWDRExposureAttr

【描述】

设置 WDR 模式下的 AE 曝光属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetWDRExposureAttr(VI_PIPE ViPipe, const ISP\_WDR\_EXPOSURE\_ATTR\_S
*pstWDRExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstWDRExpAttr	WDR 模式下的 AE 曝光属性结构体指针。	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_GetWDRExposureAttr](#)

HI_MPI_ISP_GetWDRExposureAttr

【描述】

获取 WDR 模式下的 AE 曝光属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetWDRExposureAttr(VI_PIPE ViPipe, ISP\_WDR\_EXPOSURE\_ATTR\_S  
*pstWDRExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstWDRExpAttr	WDR 模式下的 AE 曝光属性结构体指针。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetWDRExposureAttr](#)

HI_MPI_ISP_SetHDXposureAttr

【描述】

设置 HDR 模式下的 AE 曝光属性。仅在 Hi3559AV100 中支持。

【语法】

```
HI_S32 HI_MPI_ISP_SetHDXposureAttr(VI_PIPE ViPipe, const ISP\_HDX\_EXPOSURE\_ATTR\_S
*pstHDXExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstHDXExpAttr	HDR 模式下的 AE 曝光属性结构体指针。	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_GetHdRExposureAttr](#)

HI_MPI_ISP_GetHdRExposureAttr

【描述】

获取 HDR 模式下的 AE 曝光属性。仅在 Hi3559AV100 中支持。

【语法】

```
HI_S32 HI_MPI_ISP_GetHdRExposureAttr(VI_PIPE ViPipe, ISP\_HDR\_EXPOSURE\_ATTR\_S  
*pstHdRExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstHdRExpAttr	HDR 模式下的 AE 曝光属性结构体指针。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetHdRExposureAttr](#)

HI_MPI_ISP_SetSmartExposureAttr

【描述】

设置智能模式下的 AE 曝光属性。仅在有人形、人脸智能信息时生效。

【语法】

```
HI_S32 HI_MPI_ISP_SetSmartExposureAttr(VI_PIPE ViPipe, const ISP\_SMART\_EXPOSURE\_ATTR\_S
*pstSmartExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstSmartExpAttr	智能模式下的 AE 曝光属性结构体指针。	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 此功能生效需要使能 IVP 智能相关模块，详情参考《HiIVP API 参考》。目前仅在 Hi3516EV200 系列芯片上具有智能模块，客户如果希望在其他芯片上使用此功能可通过自己的智能模块得到相应智能信息并传递给 ISP，传递方式可参考 [HI_MPI_ISP_SetSmartInfo](#) 接口，ISP 得到人脸或人形的亮度信息后，会针对性地调节曝光，以使人脸或人形亮度达到设定的目标值。
- 接口使用详见 [ISP_SMART_EXPOSURE_ATTR_S](#) 说明。

【举例】

```
VI_PIPE ViPipe = 0;
ISP_SMART_EXPOSURE_ATTR_S stSmartExpAttr;

HI_MPI_ISP_GetWDRExposureAttr(ViPipe, &stWdrExposureAttr);

stWdrExposureAttr.bEnable = HI_TRUE;
stWdrExposureAttr.bIRMode= HI_FALSE;
stWdrExposureAttr.enSmartExpType= OP_TYPE_AUTO;
stWdrExposureAttr.u8LumaTarget= 110;
stWdrExposureAttr.u16ExpCoefMax= 4096;
stWdrExposureAttr.u16ExpCoefMin= 256;
stWdrExposureAttr.u8SmartInterval= 3;
stWdrExposureAttr.u8SmartSpeed= 64;
stWdrExposureAttr.u16SmartDelayNum= 180;
HI_MPI_ISP_SetWDRExposureAttr(ViPipe, &stWdrExposureAttr);
```

【相关主题】



HI_MPI_ISP_GetSmartExposureAttr

HI_MPI_ISP_GetSmartExposureAttr

【描述】

获取智能模式下的 AE 曝光属性。仅在有人形、人脸智能信息时生效。

【语法】

```
HI_S32 HI_MPI_ISP_GetSmartExposureAttr(VI_PIPE ViPipe, ISP\_SMART\_EXPOSURE\_ATTR\_S  
*pstSmartExpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstSmartExpAttr	智能模式下的 AE 曝光属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】



HI_MPI_ISP_SetSmartExposureAttr

HI_MPI_ISP_SetAERouteAttr

【描述】

设置 AE 曝光分配策略属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAERouteAttr(VI_PIPE ViPipe, const ISP_AE_ROUTE_S *pstAERouteAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteAttr	AE 曝光分配策略结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

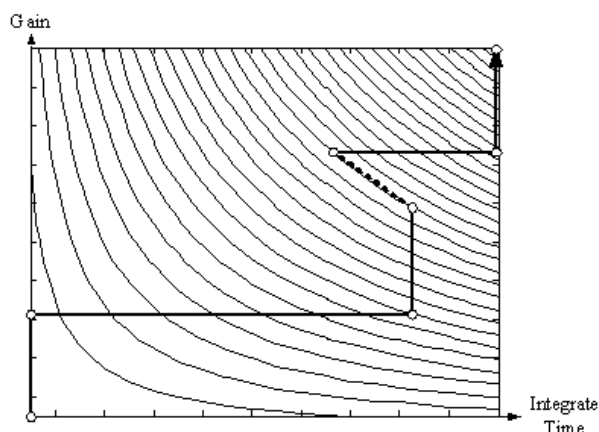
【注意】

- 此接口用于设定 AE 曝光分配路线，AE 计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光优先、增益优先、光圈优先。
- AE 分配路线示意图如[图 3-5](#) 所示。AE 分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含模拟增益、数字增益、ISP 数字增益。
 - 节点中曝光时间的单位为 us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。



- 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。
- 节点的曝光量是曝光时间、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。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 分配策略是首先分配曝光时间，其次分配增益。针对 P-Iris 镜头，默认 AE 分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置 AE route。
- 2 合 1 WDR 模式下，优先帧为短帧且增益分开配置不使能时，AE route 不生效。
- 自动降帧时，若 cmos.c 里面设置了 AE route，则在切换后会采用 cmos.c 中的 AE route，否则采用已生效的 AE route。同时，最大曝光时间的改变会更新到 AE route 中。
- 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了 AE route，则在切换后会采用 cmos.c 中的 route，否则采用默认的 AE route。
- 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与 WDR 模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的 AE route 可能与 MPI 设置的不一致，此时可以通过 HI_MPI_ISP_QueryExposureInfo 获取实际生效的 AE route。

图3-5 AE 分配路线示意图



【举例】

```

VI_PIPE ViPipe = 0;
ISP_AE_ROUTE_S stRoute;
HI_U32 au32RouteNode[3][3]
    = {{100,1024,1},{40000,1024,1},{40000,16384,1}};

HI_MPI_ISP_GetAERouteAttr(ViPipe, &stRoute);
stRoute.u32TotalNum = 3;
memcpy(stRoute.astRouteNode, au32RouteNode, sizeof(au32RouteNode));
HI_MPI_ISP_SetAERouteAttr(ViPipe, &stRoute);

```

【相关主题】

- [HI_MPI_ISP_GetAERouteAttr](#)
- [ISP_AE_ROUTE_S](#)

HI_MPI_ISP_GetAERouteAttr

【描述】

获取 AE 曝光分配策略属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAERouteAttr(VI_PIPE ViPipe, ISP\_AE\_ROUTE\_S *pstAERouteAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteAttr	AE 曝光分配策略结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetAERouteAttr](#)

HI_MPI_ISP_SetAERouteAttrEx

【描述】

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

【语法】

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

【参数】



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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 此接口用于设定 AE 曝光分配扩展属性，AE 计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor 模拟增益优先、sensor 数字增益优先、ISP 数字增益优先和光圈优先。该接口可用于设置 WDR 模式下的曝光分配路线，减轻正常室内照度多帧合成 WDR 产生的工频闪现象，优化 WDR 模式图像效果。
- AE 曝光分配扩展属性是否生效可通过配置 [HI_MPI_ISP_SetExposureAttr](#) 接口中的 bAERouteExValid 来实现。bAERouteExValid 为 HI_TRUE 时使用扩展 AE route，否则使用正常 AE route。
- AE 扩展分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈五个分量。
 - 节点中曝光时间的单位为 us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
 - 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。



- 节点的曝光量是曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
- 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如 sensor 模拟增益分量增加，那么该段路线的分配策略是 sensor 模拟增益优先。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 扩展分配策略是先分配曝光时间，再分配 sensor 模拟增益、sensor 数字增益，最后分配 ISP 数字增益。针对 P-Iris 镜头，默认 AE 扩展分配策略是先调节光圈，将光圈调至最大后调节曝光时间，最后再分配 sensor 模拟增益、sensor 数字增益和 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，扩展 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展 AE route。
- 2 合 1WDR 模式下，优先帧为短帧且增益分开配置不使能时，扩展 AE route 不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了扩展 AE route，且 cmos.c 中的扩展 AE route 标识符 bAERouteExValid 为 HI_TRUE，则在切换后会采用 cmos.c 中的扩展 AE route，否则采用默认的 AE route。
- 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与 WDR 模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的扩展 AE route 可能与 MPI 设置的不一致，此时可以通过 [HI_MPI_ISP_QueryExposureInfo](#) 获取实际生效的扩展 AE route。

【举例】

```
VI_PIPE ViPipe = 0;  
ISP_EXPOSURE_ATTR_S stExpAttr;  
ISP_AE_ROUTE_EX_S stRouteEx;  
HI_U32 au32RouteExNode1[6][5]  
    = {{ 30, 1024, 1024, 1024, 0},
```



```
{ 30, 1024, 1024, 1024, 10},
{ 30, 16384, 1024, 1024, 10},
{1000000, 16384, 1024, 1024, 10},
{1000000, 16384, 16384, 1024, 10},
{1000000, 16384, 16384, 4096, 10}};
HI_MPI_ISP_GetAERouteAttrEx(ViPipe, &stRouteEx);
HI_MPI_ISP_GetExposureAttr(ViPipe, &stExpAttr);
stExpAttr.bAERouteExValid = HI_TRUE;
stRouteEx.u32TotalNum = 6;
memcpy(stRouteEx.astRouteExNode, au32RouteExNode1, sizeof(au32RouteExNode1));
HI_MPI_ISP_SetAERouteAttrEx(ViPipe, &stRouteEx);
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);
```

【相关主题】

- [HI_MPI_ISP_GetAERouteAttrEx](#)
- [HI_MPI_ISP_SetExposureAttr](#)
- [ISP_AE_ROUTE_S](#)

HI_MPI_ISP_GetAERouteAttrEx

【描述】

获取 AE 曝光分配策略扩展属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAERouteAttrEx(VI_PIPE ViPipe, ISP\_AE\_ROUTE\_EX\_S *pstAERouteAttrEx);
```

【参数】

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

【返回值】

返回值	描述
0	成功。



非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetAERouteAttrEx](#)

HI_MPI_ISP_SetAERouteSFAttr

【描述】

WDR 模式下，设置 AE 短帧的曝光分配策略属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAERouteSFAttr(VI_PIPE ViPipe, const ISP\_AE\_ROUTE\_S  
*pstAERouteSFAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteSFAttr	AE 短帧的曝光分配策略结构体指针。	输入

【返回值】

返回值	描述
0	成功。



非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 此接口用于 WDR 模式下设定 AE 短帧的曝光分配路线，AE 计算得到的短帧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 分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含模拟增益、数字增益、ISP 数字增益。
 - 节点中曝光时间的单位为 us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
 - 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。光圈分量仅在优先帧为短帧时可控。
 - 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。
 - 不支持设置等曝光量节点。
 - 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
 - 针对 DC-Iris 和手动光圈镜头，默认 AE 分配策略是首先分配曝光时间，其次分配增益。针对 P-Iris 镜头，默认 AE 分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
 - 在线进行 DC-Iris 和 P-Iris 切换，短帧 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置短帧 AE route。



- 优先帧为长帧且增益分开配置不使能、以及线性模式下，短帧 AE route 不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了短帧 AE route，则在切换后会采用 cmos.c 中的 route，否则采用默认的短帧 AE route。
- 2 合 1WDR 模式下，帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间的一半，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间的一半。
- 发生自动降帧、线性与 2 合 1WDR 模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的 AE route 可能与 MPI 设置的不一致，此时可以通过 [HI_MPI_ISP_QueryExposureInfo](#) 获取实际生效的短帧 AE route。

【举例】

```
VI_PIPE ViPipe = 0;
ISP_AE_ROUTE_S stRoute;
HI_U32 au32RouteNode[3][3]
    = {{100,1024,1},{20000,1024,1},{20000,16384,1}};

HI_MPI_ISP_GetAERouteSFAttr(ViPipe, &stRoute);
stRoute.u32TotalNum = 3;
memcpy(stRoute.astRouteNode, au32RouteNode, sizeof(au32RouteNode));
HI_MPI_ISP_SetAERouteSFAttr(ViPipe, &stRoute);
```

【相关主题】

- [HI_MPI_ISP_GetAERouteSFAttr](#)
- [ISP_AE_ROUTE_S](#)

HI_MPI_ISP_GetAERouteSFAttr

【描述】

获取 AE 短帧曝光分配策略属性

【语法】

```
HI_S32 HI_MPI_ISP_GetAERouteSFAttr(VI_PIPE ViPipe, ISP\_AE\_ROUTE\_S *pstAERouteSFAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteSFAttr	AE 短帧的曝光分配策略结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetAERouteSFAttr](#)

HI_MPI_ISP_SetAERouteSFAttrEx

【描述】

WDR 模式下，设置 AE 短帧的曝光分配策略扩展属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAERouteSFAttrEx(VI_PIPE ViPipe, const ISP\_AE\_ROUTE\_EX\_S  
*pstAERouteSFAttrEx);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteSFAttrEx	AE 短帧的曝光分配策略扩展属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 此接口用于 WDR 模式下设定 AE 短帧的曝光分配扩展属性，AE 计算得到的短帧的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor 模拟增益优先、sensor 数字增益优先、ISP 数字增益优先和光圈优先。
- AE 曝光分配扩展属性是否生效可通过配置 [HI_MPI_ISP_SetExposureAttr](#) 接口中的 bAERouteExValid 来实现。bAERouteExValid 为 HI_TRUE 时使用扩展 AE route，否则使用正常 AE route。
- AE 扩展分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈五个分量。
 - 节点中曝光时间的单位为 us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。节点曝光时间应设置不超过 sensor 最大曝光时间的一半。
 - 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制，所以 DC-Iris 和手动光圈镜头光圈分量是无效的。即光圈类型为 DC-Iris 时，节点光圈分量不会对曝光量分配产生任何影响。光圈分量仅在优先帧为短帧时可控。



- 节点的曝光量是曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
- 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如 sensor 模拟增益分量增加，那么该段路线的分配策略是 sensor 模拟增益优先。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 扩展分配策略是先分配曝光时间，再分配 sensor 模拟增益、sensor 数字增益，最后分配 ISP 数字增益。针对 P-Iris 镜头，默认 AE 扩展分配策略是先调节光圈，将光圈调至最大后调节曝光时间，最后再分配 sensor 模拟增益、sensor 数字增益和 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，扩展 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置短帧扩展 AE route。
- 优先帧为长帧且增益分开配置不使能、以及线性模式下，短帧扩展 AE route 不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与 WDR 模式切换时，若 cmos.c 里面设置了短帧扩展 AE route，且 cmos.c 中的扩展 AE route 标识符 bAERouteExValid 为 HI_TRUE，则在切换后会采用 cmos.c 中的短帧扩展 AE route，否则采用默认的短帧 AE route。
- 2 合 1WDR 模式下，帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与 WDR 模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的扩展 AE route 可能与 MPI 设置的不一致，此时可以通过 [HI_MPI_ISP_QueryExposureInfo](#) 获取实际生效的短帧扩展 AE route。

【举例】

```
VI_PIPE ViPipe = 0;  
ISP_EXPOSURE_ATTR_S stExpAttr;  
ISP_AE_ROUTE_EX_S stRouteEx;  
HI_U32 au32RouteExNode1[6][5]
```



```
= {{ 30, 1024, 1024, 1024, 0},
    { 30, 1024, 1024, 1024, 10},
    { 30, 16384, 1024, 1024, 10},
    {20000, 16384, 1024, 1024, 10},
    {20000, 16384, 16384, 1024, 10},
    {20000, 16384, 16384, 4096, 10}};

HI_MPI_ISP_GetAERouteSFAAttrEx(ViPipe, &stRouteEx);
HI_MPI_ISP_GetExposureAttr(ViPipe, &stExpAttr);
stExpAttr.bAERouteExValid = HI_TRUE;
stRouteEx.u32TotalNum = 6;
memcpy(stRouteEx.astRouteExNode, au32RouteExNode1, sizeof(au32RouteExNode1));
HI_MPI_ISP_SetAERouteSFAAttrEx(ViPipe, &stRouteEx);
HI_MPI_ISP_SetExposureAttr(ViPipe, &stExpAttr);
```

【相关主题】

- [HI_MPI_ISP_GetAERouteSFAAttrEx](#)
- [HI_MPI_ISP_SetExposureAttr](#)
- [ISP_AE_ROUTE_EX_S](#)

HI_MPI_ISP_GetAERouteSFAAttrEx

【描述】

获取 AE 短帧曝光分配策略扩展属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAERouteSFAAttrEx(VI_PIPE ViPipe, ISP\_AE\_ROUTE\_EX\_S
*pstAERouteSFAAttrEx);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstAERouteSFAAttrEx	AE 曝光分配策略扩展属性结构体指针。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetAERouteSFAAttrEx](#)

HI_MPI_ISP_QueryExposureInfo

【描述】

获取 AE 内部状态信息，包括全局 5 段直方图、1024 段直方图和平均亮度等统计信息，同时还可获取 AE 运行状态中的曝光时间、增益、曝光量和实际生效的 AE route 等信息。

【语法】

```
HI_S32 HI_MPI_ISP_QueryExposureInfo(VI_PIPE ViPipe, ISP\_EXP\_INFO\_S *pstExpInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstExpInfo	曝光内部状态信息结构体指针。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 获取的曝光时间以微秒(us)为单位，获取的 sensor 模拟增益、sensor 数字增益和 ISP 数字增益以倍数为单位，精度是 10bit。
- 获取的曝光量=(曝光时间 * 曝光增益)，未考虑光圈状态，其中曝光时间以行数为单位，曝光增益包括 sensor 模拟增益、sensor 数字增益和 ISP 数字增益，以倍数为单位，6bit 小数精度。若该值的精度不足以满足需求，可以根据上面高精度的曝光时间(us)和增益(10bit 小数精度)重新计算一个曝光量。
- 可通过查询 s16HistError 来获取 AE 是否稳定的信息，若 s16HistError 的绝对值小于曝光容忍偏差值，意味着当前 AE 不会动作。
- 通过该接口获取的 AE route 与 Proc 信息中的 AE route 都是实际生效的值。只不过该接口的节点曝光时间以 us 为单位，而 Proc 信息中的曝光时间以行数为单位；该接口的光圈分量为 F 值大小，取值范围为[0,10]，而 Proc 信息中的光圈分量为 F 值等效增益，大小为(1<F 值)。扩展 AE route 以及短帧 AE route 和短帧扩展 AE route 也是同样的对应关系。
- 若用户使用非上海海思 AE 算法，则该接口需要自己实现并且需要同步修改 PQTOOLS 的 xml 文件，具体修改方式请参考《图像质量调试工具使用指南》第 2.1.3.7 小节。
- 调用该接口需要确保系统已经运行起来并且统计信息已经生成；当系统刚运行起来第一帧，AE 内部还没有统计信息时，调用该接口统计信息会返回全 0，这时候系统内存是足够的。

【举例】

```
VI_PIPE ViPipe = 0;  
ISP_EXP_INFO_S stExpInfo;
```




```
HI_MPI_ISP_QueryExposureInfo(ViPipe, &stExpInfo);

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

【相关主题】

无

HI_MPI_ISP_SetExpConvert

【描述】

设定等曝光量转换的曝光参数相关属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetExpConvert(VI_PIPE ViPipe, ISP\_EXP\_CONVERT\_PARAM
*pstExpConvParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstExpConvParam	等曝光量转换相关曝光量属性指针。	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、lib_hiaa.a

【注意】

作为输入的值，除了值要设定输入的 ViPipe 号，还需对 pstExpConvParam 中的成员变量 u32TarFps 进行设置目标帧率即可。

【举例】

```
VI_PIPE ViPipe = 0;
HI_S32 i;
ISP_EXP_CONVERT_PARAM stExpConvParam;

stExpConvParam.u32TarFps = 3000;

HI_MPI_ISP_SetExpConvert(ViPipe, &stExpConvParam);
HI_MPI_ISP_GetVDTimeOut(ViPipe, ISP_VD_FE_START, 50);
HI_MPI_ISP_GetExpConvert(ViPipe, &stExpConvParam);
for (i = 0; i < 4; i++) {
    printf("TimeReg.Addr[%d]: 0x%x, TimeReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stTimeReg[i].u32RegAddr, i, stExpConvParam.stTimeReg[i].u32RegValue);

    printf("AgainReg.Addr[%d]: 0x%x, AgainReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stAgainReg[i].u32RegAddr, i, stExpConvParam.stAgainReg[i].u32RegValue);
    printf("DgainReg.Addr[%d]: 0x%x, DgainReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stDgainReg[i].u32RegAddr, i, stExpConvParam.stDgainReg[i].u32RegValue);
}
```

【相关主题】

无



HI_MPI_ISP_GetExpConvert

【描述】

设定等曝光量转换的曝光参数相关属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetExpConvert(VI_PIPE ViPipe, ISP_EXP_CONVERT_PARAM
*pstExpConvParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstExpConvParam	等曝光量转换相关曝光量属性指针。	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、lib_hiaae.a

【注意】

- 转换出来的 Sensor 曝光时间、模拟增益、数字增益均为 Sensor 寄存器的值并有着对应的寄存器地址，可以直接配给 Sensor 寄存器使用。转换出来的 ISP 数字增益以倍数为单位，精度是 10bit。
- 转换出来的 Sensor 曝光时间、模拟增益、数字增益分别最多为 10 个寄存器值和 10 个寄存器地址。

【举例】

```
VI_PIPE ViPipe = 0;
```



```
HI_S32 i;
ISP_EXP_CONVERT_PARAM stExpConvParam;

stExpConvParam.u32TarFps = 3000;

HI_MPI_ISP_SetExpConvert(ViPipe, &stExpConvParam);
HI_MPI_ISP_GetVDTimeOut(ViPipe, ISP_VD_FE_START, 50);
HI_MPI_ISP_GetExpConvert(ViPipe, &stExpConvParam);
for (i = 0; i < 4; i++) {
    printf("TimeReg.Addr[%d]: 0x%x, TimeReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stTimeReg[i].u32RegAddr, i, stExpConvParam.stTimeReg[i].u32RegValue);

    printf("AgainReg.Addr[%d]: 0x%x, AgainReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stAgainReg[i].u32RegAddr, i, stExpConvParam.stAgainReg[i].u32RegValue);
    printf("DgainReg.Addr[%d]: 0x%x, DgainReg.Value[%d]: 0x%x\n", i,
stExpConvParam.stDgainReg[i].u32RegAddr, i, stExpConvParam.stDgainReg[i].u32RegValue);
}
```

【相关主题】

无

3.4.3 AI 控制模块

光圈控制接口：

- [HI_MPI_ISP_SetIrisAttr](#)：设置光圈的控制属性。
- [HI_MPI_ISP_GetIrisAttr](#)：获取光圈的控制属性。
- [HI_MPI_ISP_SetDcirisAttr](#)：设置 DC-Iris 自动光圈控制属性。
- [HI_MPI_ISP_GetDcirisAttr](#)：获取 DC-Iris 自动光圈控制属性。
- [HI_MPI_ISP_SetPirisAttr](#)：设置 P-Iris 自动光圈控制属性。
- [HI_MPI_ISP_GetPirisAttr](#)：获取 P-Iris 自动光圈控制属性。

HI_MPI_ISP_SetIrisAttr

【描述】

设定光圈的控制属性，该函数可实现手动光圈属性和光圈类型等参数的设置。

【语法】



```
HI_S32 HI_MPI_ISP_SetIrisAttr(VI_PIPE ViPipe, const ISP_IRIS_ATTR_S *pstIrisAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIrisAttr	光圈控制属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

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

【举例】

无

【相关主题】

- [ISP_IRIS_ATTR_S](#)
- [HI_MPI_ISP_SetDcirisAttr](#)
- [HI_MPI_ISP_SetPirisAttr](#)



HI_MPI_ISP_GetIrisAttr

【描述】

获取光圈的控制属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetIrisAttr(VI_PIPE ViPipe, ISP_IRIS_ATTR_S *pstIrisAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIrisAttr	光圈控制属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

无

【举例】

无

【相关主题】

无



HI_MPI_ISP_SetDcirisAttr

【描述】

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

【语法】

```
HI_S32 HI_MPI_ISP_SetDcirisAttr(VI_PIPE ViPipe, const ISP_DCIRIS_ATTR_S *pstDcirisAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- DC-Iris 光圈控制采用 PID 算法，算法根据画面亮度，调节 PWM 占空比对光圈大小进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE 直接返回，保持曝光时间和增益不变。当画面亮度稳定且 PWM 占空比维持在打开值一段时间后，AI 算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给 AE。处于光圈控制区时，更改 AE 算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE 会即时响应，根据新设定的参数和环境亮度，AI 算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭 AI 功能，否则 AE 的调节速度会受到一点影响。针对 DC-Iris



镜头建议一直打开 AI 功能，随意开关 AI 容易导致光圈控制出现异常。针对某些长焦的 DC-Iris 镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见 [ISP_DCIRIS_ATTR_S](#) 部分描述。

- 关闭 AI 功能，对于 DC-Iris 镜头，光圈会打开到最大。

【举例】

无

【相关主题】

- [ISP_IRIS_ATTR_S](#)
- [ISP_DCIRIS_ATTR_S](#)

HI_MPI_ISP_GetDcirisAttr

【描述】

获取 DC-Iris 自动光圈的属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDcirisAttr(VI_PIPE ViPipe, ISP\_DCIRIS\_ATTR\_S *pstDcirisAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h



- 库文件: libisp.a、lib_hiaa.a

【注意】

无

【举例】

无

【相关主题】

无

HI_MPI_ISP_SetPirisAttr

【描述】

设定 P-Iris 自动光圈的属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetPirisAttr(VI_PIPE ViPipe, const ISP_PIRIS_ATTR_S *pstPirisAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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



【注意】

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

【举例】

```
VI_PIPE ViPipe = 0;
ISP_PIRIS_ATTR_S stPirisAttr, stPirisAttrDef;
HI_U16 u16TotalStepDef = 93;
HI_U16 u16StepCountDef = 62;
HI_U16 au16StepFNOTableDef[1024] =
{30,35,40,45,50,56,61,67,73,79,85,92,98,105,112,120,127,135,143,150,158,166,174,183,191,200,20
8,217,225,234,243,252,261,270,279,289,298,307,316,325,335,344,353,362,372,381,390,399,408,41
7,426,435,444,453,462,470,478,486,493,500,506,512};
ISP_IRIS_F_NO_E enMaxIrisFNOTargetDef = 9;
ISP_IRIS_F_NO_E enMinIrisFNOTargetDef = 5;
stPirisAttrDef.bStepFNOTableChange = HI_TRUE;
stPirisAttrDef.bZeroIsMax = HI_TRUE;
stPirisAttrDef.u16StepCount = u16StepCountDef;
stPirisAttrDef.u16TotalStep = u16TotalStepDef;
stPirisAttrDef.enMaxIrisFNOTarget = enMaxIrisFNOTargetDef;
stPirisAttrDef.enMinIrisFNOTarget = enMinIrisFNOTargetDef;
memcpy(stPirisAttrDef.au16StepFNOTable, au16StepFNOTableDef,
sizeof(stPirisAttrDef.au16StepFNOTable));
HI_MPI_ISP_SetPirisAttr(ViPipe, &stPirisAttrDef);
HI_MPI_ISP_GetPirisAttr(ViPipe, &stPirisAttr);
```

【相关主题】

- [ISP_IRIS_ATTR_S](#)
- [ISP_PIRIS_ATTR_S](#)



HI_MPI_ISP_GetPirisAttr

【描述】

获取 P-Iris 自动光圈的控制属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetPirisAttr(VI_PIPE ViPipe, ISP\_PIRIS\_ATTR\_S *pstPirisAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
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、lib_hiaa.a

【注意】

无

【举例】

无

【相关主题】

无



3.5 数据类型

3.5.1 Register

- [AE_SENSOR_REGISTER_S](#): 定义 sensor 注册结构体。
- [AE_SENSOR_EXP_FUNC_S](#): 定义 sensor 回调函数结构体。
- [AE_SENSOR_DEFAULT_S](#): 定义 AE 算法库的初始化参数结构体。
- [AE_ACCURACY_E](#): 定义曝光时间、增益的精度类型的枚举。
- [AE_ACCURACY_S](#): 定义曝光时间、增益的精度结构体。

AE_SENSOR_REGISTER_S

【说明】

定义 sensor 注册结构体。

【定义】

```
typedef struct hiAE_SENSOR_REGISTER_S
{
    AE\_SENSOR\_EXP\_FUNC\_S stSnsExp;
} AE_SENSOR_REGISTER_S;
```

【成员】

成员名称	描述
stSnsExp	Sensor 注册的回调函数结构体。

【注意事项】

封装的目的是为了扩展。

【相关数据类型及接口】

[AE_SENSOR_EXP_FUNC_S](#)

AE_SENSOR_EXP_FUNC_S

【说明】

定义 sensor 回调函数结构体。



【定义】

```
typedef struct hiAE_SENSOR_EXP_FUNC_S
{
    HI_S32(*pfn_cmos_get_ae_default)(VI_PIPE ViPipe, AE_SENSOR_DEFAULT_S *pstAeSnsDft);
    HI_VOID(*pfn_cmos_fps_set)(VI_PIPE ViPipe, HI_FLOAT f32Fps, AE_SENSOR_DEFAULT_S
    *pstAeSnsDft);
    HI_VOID(*pfn_cmos_slow_framerate_set)(VI_PIPE ViPipe, HI_U32 u32FullLines,
    AE_SENSOR_DEFAULT_S *pstAeSnsDft);
    HI_VOID(*pfn_cmos_inttime_update)(VI_PIPE ViPipe, HI_U32 u32IntTime);
    HI_VOID(*pfn_cmos_gains_update)(VI_PIPE ViPipe, HI_U32 u32Again, HI_U32 u32Dgain);
    HI_VOID (*pfn_cmos_again_calc_table)(VI_PIPE ViPipe, HI_U32 *pu32AgainLin, HI_U32
    *pu32AgainDb);
    HI_VOID (*pfn_cmos_dgain_calc_table)(VI_PIPE ViPipe, HI_U32 *pu32DgainLin, HI_U32
    *pu32DgainDb);
    HI_VOID (*pfn_cmos_get_inttime_max)(VI_PIPE ViPipe, HI_U16 u16ManRatioEnable, HI_U32
    *au32Ratio, HI_U32 *au32IntTimeMax, HI_U32 *au32IntTimeMin, HI_U32 *pu32LFMaxIntTime);
    HI_VOID(*pfn_cmos_ae_fswdr_attr_set)(VI_PIPE ViPipe, AE_FSWDR_ATTR_S *pstAeFSWDRAttr);
    HI_VOID (*pfn_cmos_ae_quick_start_status_set)(VI_PIPE ViPipe, HI_BOOL bQuickStartStatus);
} AE_SENSOR_EXP_FUNC_S;
```

【成员】

成员名称	描述
pfn_cmos_get_ae_default	获取 AE 算法库的初始值的回调函数指针。
pfn_cmos_fps_set	设置 sensor 的帧率。
pfn_cmos_slow_framerate_set	设置 sensor 的降帧。
pfn_cmos_inttime_update	设置 sensor 的曝光时间。
pfn_cmos_gains_update	设置 sensor 的模拟增益和数字增益。
pfn_cmos_again_calc_table	计算 TABLE 类型 sensor 模拟增益。
pfn_cmos_dgain_calc_table	计算 TABLE 类型 sensor 数字增益。
pfn_cmos_get_inttime_max	WDR 模式下，计算短帧最大曝光时间的回调函数指针，与 sensor 强相关。
pfn_cmos_ae_fswdr_attr_set	2to1LineWDR 模式下，设置长帧模式。



成员名称	描述
pfn_cmos_ae_quick_start_status_set	设置 AE 无光敏快启收敛状态。

【注意事项】

- 如果回调函数指针不需要赋值，需要置为 NULL。
- 在 [AE_SENSOR_DEFAULT_S](#) 中定义了曝光时间和增益的精度，pfn_cmos_inttime_update 和 pfn_cmos_gains_update 中设置的曝光时间和增益都是带精度的值，如何转换成 sensor 的配置值与 sensor 强相关，请参阅 sensor 手册。
- 不使用无光敏快启功能时，pfn_cmos_ae_quick_start_status_set 需要设置为 NULL。
- bQuickStartStatus 是 AE 无光敏快启收敛状态的标志位。该值为 HI_TRUE 时，无光敏快启收敛完成。

【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[HIST_THRESH_NUM];
    HI_U8    u8AeCompensation;
    HI_U32   u32LinesPer500ms;
    HI_U32   u32FlickerFreq;
    HI_FLOAT f32Fps;
    HI_U32   u32HmaxTimes;
    HI_U32   u32InitExposure;
    HI_U32   u32InitAESpeed;
    HI_U32   u32InitAETolerance;
    HI_U32   u32FullLinesStd;
```



```
HI_U32  u32FullLinesMax;
HI_U32  u32FullLines;
HI_U32  u32BinningFullLines;
HI_U32  u32MaxIntTime;
HI_U32  u32MinIntTime;
    HI_U32  u32MaxIntTimeTarget;
    HI_U32  u32MinIntTimeTarget;
    AE_ACCURACY_S stIntTimeAccu;
    HI_U32  u32MaxAgain;
    HI_U32  u32MinAgain;
    HI_U32  u32MaxAgainTarget;
    HI_U32  u32MinAgainTarget;
    AE_ACCURACY_S stAgainAccu;
    HI_U32  u32MaxDgain;
    HI_U32  u32MinDgain;
    HI_U32  u32MaxDgainTarget;
    HI_U32  u32MinDgainTarget;
    AE_ACCURACY_S stDgainAccu;
    HI_U32  u32MaxISPDgainTarget;
    HI_U32  u32MinISPDgainTarget;
HI_U32  u32ISPDgainShift;
HI_U32  u32MaxIntTimeStep;
HI_BOOL bMaxTimeStepEnable;
HI_U32  u32MaxIncTimeStep[WDR_MAX_FRAME_NUM];
HI_U32  u32MaxDecTimeStep[WDR_MAX_FRAME_NUM];
HI_U32  u32LFMaxShortTime;
HI_U32  u32LFMinExposure;
ISP_AE_ROUTE_S stAERouteAttr;
HI_BOOL bAERouteExValid;
ISP_AE_ROUTE_EX_S stAERouteAttrEx;
ISP_AE_ROUTE_S stAERouteSFAttr;
ISP_AE_ROUTE_EX_S stAERouteSFAttrEx;
HI_U16  u16ManRatioEnable;
HI_U32  au32Ratio[EXP_RATIO_NUM];
ISP_IRIS_TYPE_E enIrisType;
ISP_PIRIS_ATTR_S stPirisAttr;
ISP_IRIS_F_NO_E enMaxIrisFNO;
ISP_IRIS_F_NO_E enMinIrisFNO;
```



```
ISP_AE_STRATEGY_E enAeExpMode;  
HI_U16 u16ISOCalCoef;  
HI_U8 u8AERunInterval;  
HI_U32 u32ExpRatioMax;  
HI_U32 u32ExpRatioMin;  
HI_BOOL bDiffGainSupport;  
ISP_QUICK_START_PARAM_S stQuickStart;  
ISP_PRIOR_FRAME_E enPriorFrame;  
HI_BOOL bAEGainSepCfg;  
HI_BOOL bLhcgSupport;  
HI_U32 u32SnsLhcgExpRatio;  
} AE_SENSOR_DEFAULT_S;
```



说明

宏定义如下：

```
#define HIST_THRESH_NUM (4)
```

【成员】

成员名称	描述
au8HistThresh	五段直方图的分割门限值数组，取值范围为[0,255]。推荐使用默认值，线性模式为{0xd,0x28,0x60,0x80}，BUILT_IN WDR 模式为{0x20,0x40,0x60,0x80}，帧合成 WDR 模式为{0xc,0x18,0x60,0x80}。 Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
u8AeCompensation	AE 亮度目标值，取值范围为[0,255]，建议用 0x38~0x40。
u32LinesPer500ms	每 500ms 的总行数。
u32FlickerFreq	抗闪频率，数值为当前电源频率的 256 倍。
f32Fps	基准帧率。
u32HmaxTimes	Sensor 读出一行的时间，单位：ns。
u32InitExposure	默认初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。AE 算法采用该值作为初始 5 帧的曝光量，可用于运动 DV 加速启动。 建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到 AE 快速收敛。线性/WDR 模式切换



成员名称	描述
	时，也会采用该值作为切换后第一帧的曝光量。合理修改 sensor 初始化序列的曝光时间和增益，将曝光时间(行数)*增益(6bit 小数精度)计算得到曝光量，并赋值给该变量，可使得切换更加平滑。若不设置，该值默认为 0。
u32InitAESpeed	默认初始 AE 调节速度，上海海思 AE 算法采用该值作为初始 100 帧的调节速度，可用于运动 DV 加速启动。 建议关注快速启动的产品形态可将该值配置为 128。若不设置，该值默认为 0。AE 算法内部会将该值钳位至[64, 255]。
u32InitAETolerance	默认初始 AE 曝光容忍偏差，上海海思 AE 算法采用该值作为初始 100 帧的曝光容忍偏差值，可用于设置得到首次 AE 收敛稳定标志的亮度范围。AE 统计亮度第一次落在[目标亮度- u32InitAETolerance, 目标亮度+u32InitAETolerance]范围内时，得到首次 AE 收敛稳定标志。若不设置，该值默认为 0。若在 cmos.c 不对该值赋值或赋值为 0，则 AE 算法内部将其置为 u8AeCompensation/10。AE 算法内部会将该值钳位至(0, 255]。
u32FullLinesStd	基准帧率时 1 帧的有效行数。
u32FullLinesMax	sensor 降帧后 1 帧所能达到的最大行数，一般设为 sensor 所支持的最大行数。
u32FullLines	sensor 每帧实际生效的总行数。使用上海海思 AE 算法时，回调 cmos_fps_set 和 cmos_slow_framerate_set 时必须给该值赋值，用于返回每帧实际生效的总行数。
u32BinningFullLines	sensor 在 binning 模式下的等效曝光总行数，以行为单位。
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。



成员名称	描述
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。
u32MaxISPDgainTarget	最大 ISP 数字增益目标值，以倍为单位。
u32MinISPDgainTarget	最小 ISP 数字增益目标值，以倍为单位。
u32ISPDgainShift	ISP 数字增益精度。
u32MaxIntTimeStep	自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。
bMaxTimeStepEnable	短帧曝光时间最大步长限定开关，该值为 HI_TRUE 时，短帧曝光时间最大步长限制功能生效。仅在 2 帧行合成 WDR 模式下生效。
u32MaxIncTimeStep[4]	目前只开放 u32MaxIncTimeStep[0]，u32MaxIncTimeStep[0]为短帧曝光时间增加的最大步长，以行为单位。即下一帧的曝光行数相比当前帧增加的最大曝光行数不超过 u32MaxIncTimeStep[0]。仅在 2 帧行合成 WDR 模式下 bMaxTimeStepEnable 开关使能后生效。
u32MaxDecTimeStep	目前只开放 u32MaxDecTimeStep [0]，



成员名称	描述
[4]	u32MaxDecTimeStep [0]为短帧曝光时间减少的最大步长，以行为单位。即下一帧的曝光行数相比当前帧减少的最大曝光行数不超过 u32MaxDecTimeStep [0]。仅在 2 帧合成 WDR 模式下 bMaxTimeStepEnable 开关使能后生效。
u32LFMaxShortTime	自动长帧模式下短帧最大曝光时间。
u32LFMinExposure	自动长帧模式下强制输出长帧的最小曝光量。
stAERouteAttr	AE 默认分配路线。
bAERouteExValid	AE 扩展分配路线是否生效开关，该值为 HI_TRUE 时使用扩展分配路线，否则使用普通 AE 分配路线。
stAERouteAttrEx	AE 默认扩展分配路线，合理配置可用于优化 WDR 模式图像效果。
stAERouteSFAttr	AE 短帧默认分配路线。
stAERouteSFAttrEx	AE 短帧默认扩展分配路线，合理配置可用于优化 WDR 模式图像效果。
u16ManRatioEnable	多帧合成 WDR 手动曝光比使能，该值为 HI_TRUE 时，曝光比固定，不会根据场景自动更新。
au32Ratio[3]	多帧合成 WDR 相邻 2 帧默认曝光比。当 u16ManRatioEnable 为 HI_TRUE 时，采用 au32Ratio[0] 作为 S/VS 默认曝光比，au32Ratio[1]作为 M/S 默认曝光比，au32Ratio[2]作为 L/M 默认曝光比。上述 VS, S, M, L 分别表示极短帧，短帧，中帧和长帧的曝光时间，2 帧合成时只有 VS 和 S；3 帧合成时采用 VS, S 和 M；4 帧合成时采用 VS, S, M 和 L。6bit 小数精度。 取值范围：[0x40, 0xFFFF]
enIrisType	默认镜头类型，DC-Iris 或 P-Iris。默认镜头类型与实际对接镜头类型不一致时，AI 算法无法正常工作。
stPirisAttr	P-Iris 参数，与具体镜头相关。只有默认镜头类型是 P-Iris 时，才需要用该结构体参数。



成员名称	描述
enMaxIrisFNO	P-Iris 可用的最大 F 值，与具体使用镜头相关，使用 P-Iris 时必须设置，用于限制实际生效的 enMaxIrisFNOTarget 和 enMinIrisFNOTarget，避免光圈目标值落到不合理的范围导致 AE 分配异常。 取值范围为[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。
enMinIrisFNO	P-Iris 可用的最小 F 值，与具体使用镜头相关，使用 P-Iris 时必须设置，用于限制实际生效的 enMaxIrisFNOTarget 和 enMinIrisFNOTarget，避免光圈目标值落到不合理的范围导致 AE 分配异常。 取值范围为[ISP_IRIS_F_NO_32_0, enMaxIrisFNO]。
enAeExpMode	默认曝光策略，高光优先或低光优先。建议 FSWDR 模式设置为低光优先，线性模式及 BuiltIn WDR 设置为高光优先。若不设置该值，默认为高光优先。
u16ISOCalCoef	ISO 标定系数，用于保证拍照所需 DCF 信息中显示的 ISO 是标准的，8bit 精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x100。
u8AERunInterval	默认 AE 算法运行间隔，以帧为单位。若不设置，则默认 AE 每帧执行一次。 取值范围：(0x0, 0xFF]
u32ExpRatioMax	默认最大曝光比，仅在多帧合成 WDR 模式下有效。 当 u16ManRatioEnable 为 HI_FALSE 时，u32ExpRatioMax 表示最长帧与最短帧曝光时间比值的最大值。即 2 帧合成时表示 S/VS 曝光比的最大值，3 帧合成时表示 M/VS 曝光比的最大值，4 帧合成时表示 L/VS 曝光比的最大值。 当 u16ManRatioEnable 为 HI_TRUE 时，u32ExpRatioMax 无效。 6bit 小数精度，0x40 表示曝光比为 1 倍。 取值范围：[0x40, 0x4000]
u32ExpRatioMin	默认最小曝光比，仅在多帧合成 WDR 模式下有效。



成员名称	描述
	当 u16ManRatioEnable 为 HI_FALSE 时, u32ExpRatioMin 表示长帧曝光时间与短帧曝光时间比值的最小值。 当 u16ManRatioEnable 为 HI_TRUE 时, u32ExpRatioMin 无效。 6bit 小数精度, 0x40 表示曝光比为 1 倍。 取值范围: [0x40, u32ExpRatioMax]
bDiffGainSupport	sensor 是否支持长短帧不同增益配置, 该值为 HI_TRUE 时, sensor 支持长短帧不同 sensor 模拟增益、sensor 数字增益配置, 该值为 HI_FALSE 时, sensor 不支持长短帧不同 sensor 模拟增益、sensor 数字增益配置。
stQuickStart	设置不带光敏快启相关参数。
enPriorFrame	WDR 模式下曝光路线生效的优先帧, 线性模式下此参数需配置为 LONG_FRAME。
bAEGainSepCfg	长短帧增益是否分开分配。
bLhcgSupport	Sensor 是否使用 LCG+HCG 模式, 该值为 HI_TRUE 时, Sensor 使用 LCG+HCG 模式, 该值为 HI_FALSE 时, Sensor 使用普通模式。
u32SnsLhcgExpRatio	LCG+HCG 模式的基础曝光比。

【注意事项】

- 线性/WDR 模式切换时, 会回调 pfn_cmos_get_ae_default 函数更新 AE 相关默认参数。若 WDR 模式要使用 AE 扩展分配路线而线性模式不需要, 建议在 cmos_get_ae_default 函数里面先对 AE 路线清零: bAERouteExValid = HI_FALSE, stAERouteAttr.u32TotalNum = 0, stAERouteAttrEx.u32TotalNum = 0, 然后视需要在 WDR 分支赋值。
- 上海海思 AE 算法采用 u32InitExposure 作为初始 5 帧的曝光量, 可用于运动 DV 加速启动。建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量, 以达到 AE 快速收敛。FSWDR 模式该值对应的是长帧曝光量, FSWDR 模式若要快速启动, 最好在 cmos.c 设置为固定曝光比, 以减少曝光比调整的时间, 待 AE 稳定后再根据需要设置为自动曝光比。若在 cmos.c 未给 u32InitExposure 赋



值或将 u32InitExposure 赋值为 0，则 AE 算法内部按起始曝光量为 1024 开始计算。WDR 模式切换时，上海海思 AE 算法内部会计算切换曝光量以保证切换平滑，如果希望在 WDR 模式切换时 u32InitExposure 生效，可以通过配置 u32InitAESpeed 为 0xFFFFFFFF 实现。

- 对于特定 Sensor，如 OV2718 DCG 模式，内部仅支持固定曝光比，需要配置 u16ManRatioEnable = HI_TRUE，并把 u32Ratio 配置为 Sensor 支持的固定曝光比，如果使用自动曝光比或手动配置为不支持的曝光比，会使图像效果不正常。
- u32MaxIntTimeStep 自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。此参数针对短帧曝光时间减小过快会出现坏帧的 sensor 开放，一般 sensor 可以设定为较大值，则可以受此参数限制。
- bMaxTimeStepEnable, u32MaxIncTimeStep[0], u32MaxDecTimeStep[0]，这部分参数针对 2 合 1 行合成 WDR 模式下，短帧曝光时间前后 2 帧之间增加或者减少有限制的 sensor 开放。
- u32LFMaxShortTime 为自动长帧模式下短帧曝光时间的最大值，如果此参数设置过小会导致自动长帧模式下亮区噪声表现变差。
- 对于支持 LCG+HCG 模式的 Sensor，如 OV2775，如果使用该模式，需正确配置基础曝光比 u32SnsLhcgExpRatio，并且必须配置 enPriorFrame = LONG_FRAME，bAEGainSepCfg = HI_TRUE，否则会使图像效果不正常。

【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](#)

[ISP_QUICK_START_PARAM_S](#)

AE_ACCURACY_S

【说明】

定义曝光时间、增益的精度结构体。

【定义】

```
typedef struct hiAE_ACCURACY_S
{
    AE\_ACCURACY\_E enAccuType;
    float    f32Accuracy;
    float    f32Offset;
} AE_ACCURACY_S;
```

【成员】

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



【注意事项】

- 大部分增益的精度均可归结为线性精度、DB 精度和 TABLE 精度这几类。
- 线性精度指的是增益的倍数均匀增加。例如 sensor 能支持的 again 为 1 倍, $1(1+1/16)$ 倍, $1(1+2/16)$ 倍.....这种情况下, 精度类型可以设定为 AE_ACCURACY_LINEAR, 精度值设置为 0.0625, 在这种精度下, again 为 32 则表示 $32 \times 0.0625 = 2$ 倍增益。
- DB 精度指的是增益的倍数以倍增的形式增加。例如 sensor 的芯片手册说明能支持的 again 为 0db, 0.3db, 0.6db.....这种情况下, 精度类型可以设定为 AE_ACCURACY_DB, 精度值设置为 0.3, 在这种精度下, again 为 80 则表示 $80 \times 0.3\text{db} = 24\text{db}$, 24db 是 16 倍增益。再例如 sensor 能支持的 again 为 1 倍, 2 倍, 4 倍, 8 倍.....这种情况下, 对应为 db 则为 0db, 6db, 12db, 18db, 所以精度类型设定为 AE_ACCURACY_DB, 精度值设置为 6。由于 AE 算法内部实现线性转 DB 时计算精度存在误差, 当 DB 精度较高(小于 1)时, 建议采用 TABLE 精度来实现。
- TABLE 精度指增益的倍数是通过查表的形式获得, 表格统一使用 10bit 精度, 即 1024 表示 1 倍增益。当某些 sensor 的增益值的增加规律非线性或者 DB 精度较高时, 可以使用 TABLE 方式, AE 算法计算出需要的模拟增益/数字增益的数值, 用查询 sensor 增益表格中最接近的值作为数字增益/模拟增益的值。
- 使用 TABLE 模式时, 需要相应的初始化回调结构体 AE_SENSOR_EXP_FUCN_S 中的回调函数。
- 曝光时间的精度通常都是线性的, 都是以行为单位。若曝光时间为线性精度, 精度 f32Accuracy 为大于 1 的整数, 则意味着 AE 计算出来的曝光行数只能为最小曝光时间加上 f32Accuracy 的整数倍。该功能可用于计算某些 sensor WDR 模式下的长/短帧曝光时间, 如某些 sensor 曝光行数必须为 $2n+1$, 则可以配置最小曝光时间为 1 或 3, 配置 f32Accuracy 为 2。
- 曝光时间与 sensor 增益保持线性关系是 AE 曝光量分配的前提, 即认为曝光量一定的情况下, 曝光时间和 sensor 增益可以相互转换而保持图像亮度基本不变。比如说曝光量为 4096, 那么分配为曝光时间(2)*增益(2048), 也可以分配为曝光时间(4)*增益(1024), 这 2 种情况下图像亮度应该基本不变。若不满足这种线性关系, 在高亮环境下, 由于曝光时间很短, 1 行曝光时间的变化也容易使画面发生闪烁。某些 sensor, 如 Panasonic MN34220 采用 1080p@30fps 的初始化序列配置时, 曝光时间必须偏移 0.8018 行之后才与 sensor 增益有线性关系, 因此增加了 f32Offset 这个变量, 它为 float 型变量, 以行为单位, 将 f32Offset 设置成



0.8018, AE 算法内部即可完成偏移处理。一般 sensor 的曝光时间和增益不存在这种偏移关系, 需在 `cmos.c` 中将该值配为 0。

- 规定这 3 种精度类型, 可以以统一的接口形式对接绝大部分 sensor。

【相关数据类型及接口】

[ISP_SENSOR_REGISTER_S](#)

3.5.2 AE

- [ISP_AE_MODE_E](#): 定义自动曝光的模式。
- [ISP_AE_STRATEGY_E](#): 定义 AE 曝光策略模式。
- [ISP_AE_DELAY_S](#): 定义 AE 延时属性。
- [ISP_AE_RANGE_S](#): 定义曝光时间或增益的最大值和最小值。
- [ISP_ANTIFLICKER_MODE_E](#): 定义抗闪模式。
- [ISP_ANTIFLICKER_S](#): 定义抗闪属性。
- [ISP_SUBFLICKER_S](#): 定义 ISP 图像亚抗闪属性。
- [ISP_FSWDR_MODE_E](#): 定义 ISP FSWDR 运行模式。
- [ISP_AE_ATTR_S](#): 定义自动曝光属性。
- [ISP_ME_ATTR_S](#): 定义手动曝光属性。
- [ISP_EXPOSURE_ATTR_S](#): 定义 ISP 曝光属性。
- [ISP_WDR_EXPOSURE_ATTR_S](#): 定义 WDR 模式下的曝光属性。
- [ISP_HDR_EXPOSURE_ATTR_S](#): 定义 HDR 模式下的曝光属性。
- [ISP_SMART_EXPOSURE_ATTR_S](#): 定义智能模式下的曝光属性。
- [ISP_AE_ROUTE_NODE_S](#): 定义 AE 分配路线节点属性。
- [ISP_AE_ROUTE_S](#): 定义 AE 曝光分配策略属性。
- [ISP_AE_ROUTE_EX_NODE_S](#): 定义 AE 扩展分配路线节点属性。
- [ISP_AE_ROUTE_EX_S](#): 定义 AE 曝光分配策略扩展属性。
- [ISP_EXP_INFO_S](#): 定义 ISP 曝光内部状态信息。
- [ISP_PRIOR_FRAME_E](#): 定义 WDR 模式下的优先帧。

ISP_AE_MODE_E

【说明】

定义自动曝光的模式。



【定义】

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

【成员】

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

【注意事项】

- 自动降帧模式是指自动曝光调节时会优先增大曝光时间来尽量减小增益。当 sensor 增益达到用户设置的最大值时，自动曝光调节会逐渐降帧率并延长曝光时间，持续到曝光时间等于自动曝光的最大时间为止。低照度环境下噪声较小但帧率会降低。
- 固定帧率模式是指自动曝光调节时保持帧率不变，低照度环境下噪声会较大。
- Built-In WDR 模式和多帧合成 WDR 模式支持降帧。若是客户自己对接的 Line-WDR sensor，支持自动降帧处理需要修改 cmos.c，修改方法可以参考 imx290 的 cmos.c 文件。另外，在 cmos.c 中 Built-In WDR 模式降帧设置应和线性一样。

【相关数据类型及接口】

无

ISP_AE_STRATEGY_E

【说明】

定义 AE 曝光策略模式。

【定义】

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



```
    AE_EXP_LOWLIGHT_PRIOR = 1,  
    AE_STRATEGY_MODE_BUTT  
} ISP_AE_STRATEGY_E;
```

【成员】

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

【注意事项】

- 线性模式下，上海海思 AE 算法默认曝光策略为高光优先，尽量避免画面过曝，但在宽动态场景时，暗处亮度较低。此时若主要关注暗处区域，可采用低光优先模式，但亮处容易过曝。
- WDR 模式下，上海海思 AE 算法默认使用低光优先模式。在低光优先模式和高光优先模式都是控制长帧曝光，手动模式曝光时间、最大最小曝光时间、AE_Route、AE_Route_Ex 中曝光时间参数影响长帧曝光时间。

【相关数据类型及接口】

无

ISP_AE_DELAY_S

【说明】

定义 AE 延时属性。

【定义】

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

【成员】



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

【注意事项】

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

【相关数据类型及接口】

无

ISP_AE_RANGE_S

【说明】

定义曝光时间或增益的最大值和最小值。

【定义】

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

【成员】

成员名称	描述
u32Max	最大值。
u32Min	最小值。



【注意事项】

最小值不能大于最大值。

【相关数据类型及接口】

无

ISP_ANTIFLICKER_MODE_E

【说明】

定义抗闪模式。

【定义】

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

【成员】

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

【注意事项】

- ISP_ANTIFLICKER_NORMAL_MODE 为普通抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间固定为 1/120 sec（60Hz）或 1/100 sec(50Hz)，不受曝光时间最小值的限制。
 - 有灯光的环境：曝光时间可与光源频率相匹配，能够防止图像闪烁。
 - 高亮度的环境：亮度越高，要求曝光时间就最短。而普通抗闪模式的最小曝光时间不能匹配光源频率，产生过曝。



- ISP_ANTIFLICKER_AUTO_MODE 为自动抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间可达到 sensor 的最小曝光时间。与普通抗闪模式的差别主要在高亮度的环境体现。
 - 高亮度的环境：最小曝光时间可以达到 sensor 的最小曝光时间，能够有效抑制过曝，但此时抗闪失效。

【相关数据类型及接口】

无

ISP_ANTIFLICKER_S

【说明】

定义抗闪属性。

【定义】

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

【成员】

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

【注意事项】

- 抗闪开启后曝光时间会受到最大/最小曝光时间的限制，若最小抗闪时间大于最大曝光时间，那么开启抗闪后曝光时间将被限制为最大曝光时间。



- 开启抗闪时，AERoute 中各节点的 Again、DGain、ISPDGain 参数至少有一个需小于对应参数范围的上限，避免特定场景下无法进行补偿导致图像闪烁。

【相关数据类型及接口】

[ISP_ANTIFLICKER_MODE_E](#)

ISP_SUBFLICKER_S

【说明】

定义 ISP 图像亚抗闪属性。

【定义】

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

【成员】

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

【注意事项】

- 强制抗闪模式时，最小曝光时间固定为 1/120 sec（60Hz）或 1/100 sec(50Hz)，在某些场景（如室内对准室外窗户的背光场景）画面可能会过曝严重，但不抗闪画面工频闪又比较厉害。在这种情况下引入亚抗闪模式是为了取得过曝与闪烁之间的平衡。在强制抗闪模式下，当亚抗闪功能生效时，若画面亮度小于 (AeCompensation + u8LumaDiff)，那么曝光时间仍会固定为最小抗闪时间 1/120 sec（60Hz）或 1/100 sec(50Hz)以防止图像闪烁。若画面亮度大于 (AeCompensation + u8LumaDiff)，则取消抗闪，调整画面目标亮度为 (AeCompensation + u8LumaDiff)，通过引入一定程度的画面闪烁来避免画面过曝严重。注意：非背光场景 u8LumaDiff 较小时（如小于 10）可能会导致曝光时



间在最小抗闪点左右来回变化，导致画面轻微闪烁，建议设置该值不小于 20。实际上，若 u8LumaDiff 设置较小，背光场景下相当于取消了抗闪功能，与自动抗闪模式下的效果相近，达不到亚抗闪的目的，因此该值只有设置一个稍大些的值才有意义。

- 只有在满足打开抗闪、强制抗闪模式且抗闪频率值不等于 0 的前提下，亚抗闪功能才能生效。

【相关数据类型及接口】

无

ISP_FSWDR_MODE_E

【说明】

定义 ISP FSWDR 运行模式。

【定义】

```
typedef enum hiISP_FSWDR_MODE_E
{
    ISP_FSWDR_NORMAL_MODE = 0x0,
    ISP_FSWDR_LONG_FRAME_MODE = 0x1,
    ISP_FSWDR_AUTO_LONG_FRAME_MODE = 0x2,
    ISP_FSWDR_MODE_BUTT
}ISP_FSWDR_MODE_E;
```

【成员】

成员名称	描述
ISP_FSWDR_NORMAL_MODE	正常 FSWDR 模式，此时 AE 和合成模块按照自动/手动曝光比进行工作。
ISP_FSWDR_LONG_FRAME_MODE	长帧模式，此时 AE 将短帧曝光时间设置为最小值，长帧曝光时间接近 1 帧所允许的最大值，合成模块只输出长帧数据，可用于优化弱宽动态场景或低照度时的图像质量。
ISP_FSWDR_AUTO_LONG_FRAME_MODE	自动长帧模式，此时 AE 短帧曝光时间不会超过 sensor 限制的长帧模式下短帧最大曝光时间，在曝光量超过



成员名称	描述
	sensor 设置的阈值时，自动切换为长帧模式，合成模块只输出长帧数据。

【注意事项】

- 长帧模式仅在行模式 WDR 长帧优先模式下有效，自动长帧模式仅在 2to1 行模式 WDR 长帧优先模式下有效，并且为了保证图像质量，WDR 模块的运动检测及长短帧融合阈值均为自动配置，不支持手动配置。在线将模式切换到 WDR 模式，默认为正常 FSWDR 模式。
- 自动长帧模式下，手动曝光比生效，如果手动曝光比大于 1，即使曝光量大于设置的阈值也不会进入长帧模式；如果手动曝光比为 1，则会自动进入长帧模式。
- 长帧模式下，如果 WDR 模块开启 FUSION 模式，可能会导致亮区颜色偏粉或变灰，因此不推荐长帧模式与 WDR 模块 FUSION 模式同时开启。
- 长帧模式和自动长帧模式仅支持在优先帧为长帧且增益分开配置不使能的情况下使用。

【相关数据类型及接口】

无

ISP_AE_ATTR_S

【说明】

定义自动曝光属性。

【定义】

```
typedef struct hiISP_AE_ATTR_S
{
    ISP_AE_RANGE_S stExpTimeRange;
    ISP_AE_RANGE_S stAGainRange;
    ISP_AE_RANGE_S stDGainRange;
    ISP_AE_RANGE_S stISPDGainRange;
    ISP_AE_RANGE_S stSysGainRange;
    HI_U32 u32GainThreshold;
    HI_U8  u8Speed;
    HI_U16 u16BlackSpeedBias;
```



```

HI_U8  u8Tolerance;
HI_U8  u8Compensation;
HI_U16 u16EVBias;
ISP_AE_STRATEGY_E enAESTrategyMode;
HI_U16 u16HistRatioSlope;
HI_U8  u8MaxHistOffset;
ISP_AE_MODE_E enAEMode;
ISP_ANTIFLICKER_S stAntiflicker;
ISP_SUBFLICKER_S stSubflicker;
ISP_AE_DELAY_S stAEDelayAttr;
HI_BOOL bManualExpValue;
HI_U32 u32ExpValue;
ISP_FSWDR_MODE_E enFSWDRMode;
HI_BOOL bWDRQuick;
HI_U16 u16ISOCalCoef;
} ISP_AE_ATTR_S;

```

【成员】

成员名称	描述
stExpTimeRange	曝光时间范围，设置最大值和最小值，以微秒(us)为单位。 取值范围：[0x0, 0xFFFFFFFF]，具体范围与 sensor 相关。
stAGainRange	Sensor 模拟增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
stDGainRange	Sensor 数字增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
stISPDGainRange	ISP 数字增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0x40000]。
stSysGainRange	系统增益范围，设置最大值和最小值，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32GainThreshold	自动降帧时的系统增益门限值，10bit 小数精度。



成员名称	描述
	取值范围：[0x400, 0xFFFFFFFF]。
u8Speed	自动曝光调整时的速度。 取值范围：[0x0, 0xFF]，默认值为 0x40。
u16BlackSpeedBias	画面由暗到亮 AE 调节速度的偏差值，该值越大，画面从暗到亮的速度越快。 取值范围：[0x0, 0xFFFF]，默认值为 0x90。
u8Tolerance	自动曝光调整时对画面亮度的容忍偏差。 取值范围：[0x0, 0xFF]，默认值为 0x2。
u8Compensation	自动曝光调整时的目标亮度。 取值范围：[0x0, 0xFF]，默认值为 0x38。
u16EVBias	自动曝光调整时的目标亮度偏差值，10bit 小数精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x400。
enAESTrategyMode	自动曝光策略，高光优先或低光优先。
u16HistRatioSlope	感兴趣区域的权重。 取值范围：[0x0, 0xFFFF]，默认值为 0x80。
u8MaxHistOffset	感兴趣区域对统计平均值影响的最大程度。 取值范围：[0x0, 0xFF]，默认值为 0x10。
enAEMode	自动曝光模式，自动降帧模式或固定帧率模式。
stAntiflicker	抗闪属性设置。默认抗闪不使能。
stSubflicker	亚抗闪属性设置。默认亚抗闪不使能。
stAEDelayAttr	延时属性设置。默认 u16BlackDelayFrame=8, u16WhiteDelayFrame=0。
bManualExpValue	手动曝光量使能，该值为 HI_TRUE 时，AE 算法采用 u32ExpValue 作为当前曝光量进行曝光时间和增益等的分配，为 HI_FALSE 时采用自动计算的曝光量进行分配。默认为 HI_FALSE。



成员名称	描述
u32ExpValue	手动曝光量值，等于曝光时间*系统增益，其中曝光时间的单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]。
enFSWDRMode	FSWDR 运行模式。 默认为 ISP_FSWDR_NORMAL_MODE。
bWDRQuick	WDR 模式下，AE 算法从稳定状态（亮度误差小于等于容忍偏差值 u8Tolerance）重新调整时，默认前 50 帧调整会进行时域滤波，以调整更加平滑。该值为 HI_TRUE 时，取消 50 帧时域滤波，使 AE 收敛速度更快。默认为 HI_FALSE。
u16ISOCalCoef	ISO 标定系数，用于保证拍照所需 DCF 信息中显示的 ISO 是标准的，8bit 精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x100。

【注意事项】

- 自动模式下，更改手动曝光属性的值不会生效。
- 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。

注意：

为了防止图像过曝区域颜色不饱和，驱动内部会根据 sensor 黑电平自动修正最小的 ISPDGain 值，因此 AE 在自动模式下，即使限制 ISPDGain 最大值为 1024，ISPDGain 实际生效值仍会大于 1024。如果需要强制 ISPDGain 为 1024,可以通过手动模式实现，但是此时图像可能会出现亮区偏粉的问题。

- 自动曝光的系统增益

非扩展路径下，若(sensor 模拟增益最小值*sensor 数字增益最小值*ISP 数字增益最小值)小于系统增益最小值，则 AE 算法内部计算时最小增益会被限制到系统增益的最小值。若(sensor 模拟增益最大值*sensor 数字增益最大值*ISP 数字增益最大值)大于系统增益最大值，则 AE 算法内部计算时最大增益会被限制到系统增益的最大值。推荐通过设置系统增益的最大、最小值进行增益限制，分别限制

sensor 模拟增益、sensor 数字增益和 ISP 数字增益时，若把较高精度的 ISP 数字增益限制到 1 倍，容易导致闪烁。

扩展路径下，采用系统增益的最大/最小值结合曝光时间的范围来限制曝光量的范围，sensor 模拟增益、sensor 数字增益及 ISP 数字增益不直接受系统增益的影响。实际上，AE 算法内部利用系统增益来计算最大/最小曝光量，而并不是直接限制某项增益的值。比如说系统增益最大值限制为 1 倍，但 sensor 模拟增益的最小值限制为 2 倍，则实际生效的结果以 sensor 模拟增益的限制为准，sensor 数字增益及 ISP 数字增益的限制也同理。手动模式下，系统增益不起作用，不对某项增益的值进行限制。

- 自动降帧时的系统增益门限值

在 SLOW_SHUTTER 模式下，当系统增益达到所设置的门限值时，系统将自动进入 SLOW_SHUTTER 模式。

- u8Speed 用于设定自动曝光时的收敛速度，该值越大曝光的收敛速度越快，但也会导致收敛过程中出现反复震荡。对于 Panasonic mn34220 这款 sensor，受 sensor 性能限制，在 2 帧合成 WDR 模式下，建议 u8Speed 设置不能超过 64，否则在光线剧烈变化场景时可能出现闪烁。
- u16BlackSpeedBias 用于设定画面由暗到亮 AE 调节速度的偏差值。默认 u8Speed 下，画面从亮到暗的速度会快于画面从暗到亮的速度，若想双方向的 AE 调节速度差不多，可以调高 u16BlackSpeedBias。
- 亮度补偿属性 u8Compensation 用于调节曝光的目标亮度。曝光亮度补偿值越大则图像亮度越高。
- 曝光偏差属性 u16EVBias 用于在特殊场景下对画面的目标亮度进行微调，也可认为是更高精度的亮度补偿值，通过调节该值来改变画面目标亮度，真实生效的 AE 目标亮度为 $u8Compensation * u16EVBias / 1024$ 。u8Compensation 不变时，该值越大则图像亮度越高。
- 曝光容忍偏差属性 u8Tolerance 用于调节曝光对环境的灵敏度，曝光容忍偏差值越大则曝光越不敏感，且可能导致同一目标亮度值多次调节得到的亮度差异越大，所以该属性推荐不能设定过大。尤其当目标亮度较低时，Tolerance 较大，最终画面收敛亮度可能会有明显差异。通过改变 EVBias 对画面目标亮度进行微调时，若 Tolerance 较大，过小的 EVBias 变化量可能会看不出调整效果。
- 曝光策略属性 enAESTrategyMode 用于选择对高光优先或低光优先的曝光策略。高光优先意味着对高光敏感，尽量避免画面过曝。低光优先意味着对低光敏感，尽量看清楚暗处区域，不管画面是否过曝。默认的曝光策略是高光优先，用户可根据场景需要进行调整。



- u16HistRatioSlope 用于设定感兴趣区域的权重。若是高光优先曝光模式，则该值设置的是高光区域的权重，该值越大，则意味着对高光区域越敏感。反之，若是低光优先模式，则该值设置的是低光区域的权重，该值越大，则意味着对暗处区域越敏感，建议 u16HistRatioSlope 的值设置不要超过 0x100。高光优先曝光模式下，如果该值过大，只要画面中有一小部分过曝区域，AE 算法就会认为画面亮度远大于目标亮度，这样曝光量就会不断往下调，导致画面变得很黑，同时画面中的过曝区域也在快速变小，达到某个程度的话 AE 算法又会认为画面亮度比目标亮度小了，曝光量又会往上调，如此反复出现来回调的现象，看起来像是画面闪烁。类似的，在低光优先模式下，如果 AE 目标亮度设置得比较低，若画面中有黑色物体来回移动，对画面亮度也会有较大影响，此时调低曝光调整步长 u8ExpStep 和提高容忍偏差 s16ExpTolerance 能够缓解该问题。
- 自动曝光时，可设定感兴趣区域对统计平均值影响的最大程度 u8MaxHistOffset。该值相当于对提高 u16HistRatioSlope 时增加的权重做限制，若该值为 0，无论 u16HistRatioSlope 多大，也不会对高光或低光区域做特殊处理，此时的统计平均值就是原始值。通过合理设置该值，可以保证任何场景 AE 稳定后画面平均亮度都在一定范围内，在高光优先曝光模式下，如果该值设置较大，在对比度稍高一些的场景，如晴天室外场景，有天空有树木，则可能导致整体画面亮度偏低，因为此时优先保证了亮区天空的效果，通过限制该值从而限制对亮区的倾斜程度可以解决该问题。
- 曝光策略切换时，最好同时更新 u16HistRatioSlope 和 u8MaxHistOffset 的值，否则这 2 个值会采用默认配置，效果可能与预期不符。
- 在做强光抑制方案时，例如在 RGBIR 模式下，建议设置成高光优先曝光，通过降低 AE 目标亮度，同时适当设置 u16HistRatioSlope 和 u8MaxHistOffset 为较大值来抑制强光，暗区则可以通过使能 DRC 来看清楚。低光优先则可以用于实现非指定区域的背光补偿。
- AE 曝光控制类型为自动时，可设定曝光模式 enAEMode。该值可设定为慢快门模式(SLOW_SHUTTER)或固定帧率模式。慢快门模式通常用于低照度场景下进行自动降帧，以减少画面噪声。
- 抗闪属性结构体 stAntiflicker 可用于设定抗闪使能，抗闪频率和抗闪模式等属性。FSWDR 模式下，长帧抗闪有效，在优先帧为短帧且增益分开配置时，长短帧抗闪都有效。由于随着曝光比的变化，短帧曝光时间的最大值会受到限制，建议 FSWDR 模式使用自动抗闪模式。

注意：OV2718 WDR 模式下只曝光一次，曝光时间只有短帧有效，因此抗闪及亚抗闪效果并不能达到预期。



- 亚抗闪属性结构体 `stSubflicker` 可用于设定亚抗闪使能及抗闪程度属性。若有自动光圈，建议关闭亚抗闪功能。
- AE 曝光控制类型为自动时，可设定 AE 延时生效属性结构体 `stAEDelayAttr`。该值的合理设置可提高画面亮度的稳定性，防止快速运动物体经过导致画面亮度发生变化。低码率设置时可适当增加该值，以避免 AE 调节时出现块效应。
- 用户可以通过将 `bManualExpValue` 置为 `HI_TRUE`，手动设置曝光量 `u32ExpValue` 来屏蔽上海海思 AE 算法的曝光量调整部分，只用到曝光量分配部分。`u32ExpValue` 的值会受到最大、最小曝光量的限制。最大、最小曝光量分别对应最大、最小曝光时间与增益的乘积。
- WDR 模式下，优先帧为长帧时，`stExpTimeRange` 设置的是长帧的曝光时间范围，优先帧为短帧时，`stExpTimeRange` 设置的是短帧的曝光时间范围。
- FSWDR 运行模式改变时，需要相应修改 `cmos.c` 中的以下回调函数：
`cmos_long_frame_mode_set`, `cmos_get_ae_default`, `cmos_fps_set`, `cmos_get_inttime_max`，以保证长、短帧曝光时间在合理范围内。正常 WDR 和长帧模式切换时，会回调 `cmos_get_ae_default`，更新曝光时间最大/最小值，增益最大/最小值，光圈最大/最小值和 AE compensation 等参数，同时，AE 内部会将 `u16HistRatioSlope` 和 `u8MaxHistOffset` 更新为算法默认值。需要注意的是，此时曝光比相关设置不会更新，仍会保持上一次的值。另外，长帧模式打开时，DRC 强度会逐渐收敛至 Auto 目标值。客户可以根据需要，在切换完成后调用 MPI 接口修改参数。

注意：平滑切换的前提是切换前/后短帧曝光量与长帧曝光量相等。当曝光比较大时，由于正常 WDR 模式会限制增益，而长帧模式为了保证低照度效果会放开增益限制，可能会导致切换前/后短帧曝光量与长帧曝光量不等，这样切换就会有几帧突然之间变亮或者变暗，属于正常现象。若要避免这种现象，可以在切换完成之后再放开增益限制。

- 长帧模式仅在行模式 WDR 下才有效，帧模式 WDR 把曝光比设置为 1:1，关闭运动检测，即可达到与行模式 WDR 长帧模式一样的效果。建议非行模式 WDR 统一将 `enFSWDRMode` 设置为 `ISP_FSWDR_NORMAL_MODE`。
- 行模式 WDR 只曝光一次的 sensor 不支持长帧模式，如 OV2718。

【相关数据类型及接口】

[HI_MPI_ISP_SetExposureAttr](#)

ISP_ME_ATTR_S

【说明】



定义手动曝光属性。

【定义】

```
typedef struct hiISP_ME_ATTR_S
{
    ISP_OP_TYPE_E enExpTimeOpType;
    ISP_OP_TYPE_E enAGainOpType;
    ISP_OP_TYPE_E enDGainOpType;
    ISP_OP_TYPE_E enISPDGainOpType;
    HI_U32 u32ExpTime;
    HI_U32 u32AGain;
    HI_U32 u32DGain;
    HI_U32 u32ISPDGain;
} ISP_ME_ATTR_S;
```

【成员】

成员名称	描述
enExpTimeOpType	手动曝光时间使能，默认值为 OP_TYPE_AUTO
enAGainOpType	手动 sensor 模拟增益使能，默认值为 OP_TYPE_AUTO
enDGainOpType	手动 sensor 数字增益使能，默认值为 OP_TYPE_AUTO
enISPDGainOpType	手动 ISP 数字增益使能，默认值为 OP_TYPE_AUTO
u32ExpTime	手动曝光时间，以微秒(us)为单位，默认值为 0x4000。 取值范围：[0x0, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32AGain	手动 sensor 模拟增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32DGain	手动 sensor 数字增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0xFFFFFFFF]，具体范围与 sensor 相关。
u32ISPDGain	手动 ISP 数字增益，10bit 小数精度，默认值为 0x400。 取值范围：[0x400, 0x40000]，具体范围与 sensor 相关。

【注意事项】



- 手动模式下，手动曝光时间和增益的值会受到自动模式下最大/最小曝光时间和增益的限制。
- 手动曝光使能参数有效时，必须设置相应的手动曝光参数，若不设置，则采用系统默认值。
- 增益单位为 10bit 小数精度的倍数，即 1024 代表 1 倍。
- 若曝光参数设置超出最大（小）值，将使用 sensor 支持的最大（小）值代替。

【相关数据类型及接口】

[HI_MPI_ISP_SetExposureAttr](#)

ISP_EXPOSURE_ATTR_S

【说明】

定义 ISP 曝光属性。

【定义】

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

【成员】

成员名称	描述
bByPass	AE 模块 bypass 功能使能，默认为 HI_FALSE。
enOpType	自动曝光或手动曝光开关，默认为 OP_TYPE_AUTO。
u8AERunInterval	AE 算法运行的间隔，取值范围为[1, 255]。



成员名称	描述
	取值为 1 时表示每帧都运行 AE 算法； 取值为 2 时表示每 2 帧运行 1 次 AE 算法，依此类推。 建议该值设置不要大于 2，否则 AE 调节速度会受到影响。WDR 模式时，该值建议设置为 1，这样 AE 收敛会更加平滑。该值默认为 1。
bHistStatAdjust	1024 段直方图统计方式是否根据场景进行调整开关，默认为 HI_TRUE。
bAERouteExValid	AE 扩展分配路线是否生效开关，HI_TRUE 时使用 AE 扩展分配路线，否则使用普通 AE 分配路线。默认为 HI_FALSE。
stManual	手动曝光属性结构体
stAuto	自动曝光属性结构体
enPriorFrame	WDR 模式下曝光路线生效的优先帧，线性模式下此参数需配置为 LONG_FRAME
bAEGainSepCfg	长短帧增益是否分开分配
bAdvanceAE	是否使能 Advance AE

【注意事项】

- AE ByPass 为 HI_TRUE 时，AE 模块被 bypass，任何 AE 配置都不会对图像亮度产生影响。ISP_AE_RESULT_S 保持为 AE bypass 前一帧的值。
- 上海海思 AE 主要基于全局 1024 段直方图信息进行亮度调整，正常场景 1024 段直方图只统计 Gb 或 Gr 分量。bHistStatAdjust 为 HI_TRUE 时，若进入了大面积单色场景（如大面积红色或蓝色），AE 算法会根据全局平均值信息对 1024 段直方图统计方式进行调整，把 R 分量或 B 分量也考虑进来，如此可以防止大面积红色或蓝色场景画面亮度偏高。bHistStatAdjust 为 HI_FALSE 时，AE 算法不会调整 1024 段直方图统计方式。红外场景建议将 bHistStatAdjust 设置为 HI_FALSE。
- WDR/线性模式切换时，u8AERunInterval 会重置为 1，用户可以在切换完成后，根据需要修改该值。
- bAEGainSepCfg 仅在 2to1WDR 模式下配置生效。
- 手动模式不支持长短帧增益分开分配。



- 曝光比变化过程中，切换 bAEGainSepCfg 会交叉影响长短帧的曝光时间限制，可能会导致图像亮度突变，建议在曝光比收敛状态下切换 bAEGainSepCfg。
- Advance AE 目前仅在 WDR 模式下使用低光优先时生效，使能 Advance AE，可使图像暗区和亮区的曝光更加合理、准确，当场景发生亮暗比例变化时，图像亮暗区曝光更加稳定，在高光优先模式下使能 Advance AE 效果不保证。

【相关数据类型及接口】

- [HI_MPI_ISP_SetExposureAttr](#)
- [HI_MPI_ISP_GetExposureAttr](#)

ISP_WDR_EXPOSURE_ATTR_S

【说明】

定义 WDR 模式下的曝光属性。

【定义】

```
typedef struct hiISP_WDR_EXPOSURE_ATTR_S
{
    ISP_OP_TYPE_E enExpRatioType;
    HI_U32 au32ExpRatio[EXP_RATIO_NUM];
    HI_U32 u32ExpRatioMax;
    HI_U32 u32ExpRatioMin;
    HI_U16 u16Tolerance;
    HI_U16 u16Speed;
    HI_U16 u16RatioBias;
    HI_U16 u16HighLightTarget;
    HI_U16 u16ExpCoefMin;
} ISP_WDR_EXPOSURE_ATTR_S;
```



说明

宏定义如下：

```
#define EXP_RATIO_NUM      (3)
```

【成员】

成员名称	描述
enExpRatioType	仅在多帧合成 WDR 模式下有效。 OP_TYPE_AUTO：根据场景自动计算长短帧曝光比； OP_TYPE_MANUAL：手动配置长短帧曝光比。



成员名称	描述
au32ExpRatio	仅在多帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时, au32ExpRatio 无效。 当 enExpRatioType 为 OP_TYPE_MANUAL 时, au32ExpRatio 为可读写, 表示多帧合成 WDR 相邻 2 帧曝光比期望值。其中 au32ExpRatio[0]作为 S/VS 曝光比, au32ExpRatio[1]作为 M/S 曝光比, au32ExpRatio[2]作为 L/M 曝光比。上述 VS, S, M, L 分别表示极短帧, 短帧, 中帧和长帧的曝光时间, 2 帧合成时只有 VS 和 S; 3 帧合成时采用 VS, S 和 M; 4 帧合成时采用 VS, S, M 和 L。6bit 小数精度, 0x40 表示曝光比为 1 倍。 取值范围: [0x40, 0xFFFF]
u32ExpRatioMax	仅在多帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时, u32ExpRatioMax 表示最长帧与最短帧曝光时间比值的最大值。即 2 帧合成时表示 S/VS 曝光比的最大值, 3 帧合成时表示 M/VS 曝光比的最大值, 4 帧合成时表示 L/VS 曝光比的最大值。 当 enExpRatioType 为 OP_TYPE_MANUAL 时, u32ExpRatioMax 无效。 6bit 小数精度, 0x40 表示曝光比为 1 倍。 取值范围: [0x40, 0x4000]
u32ExpRatioMin	仅在多帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时, u32ExpRatioMin 表示长帧曝光时间与短帧曝光时间比值的最小值。 当 enExpRatioType 为 OP_TYPE_MANUAL 时, u32ExpRatioMin 无效。 格式为无符号 6.6bit 定点, 0x40 表示长帧曝光时间与短帧曝光时间的比值为 1 倍。默认值为 0x40。 取值范围: [0x40, u32ExpRatioMax]



成员名称	描述
u16Tolerance	曝光比容忍值，仅在两帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时，该值越大，表示场景动态范围变化在一定范围内时，曝光比保持不变。默认值为 0xC。 取值范围：[0x0, 0xFF]
u16Speed	自动曝光比调节速度，仅在两帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时，该值越大，自动曝光比调节速度越快。默认值为 0x20。 取值范围：[0x0, 0xFF]
u16RatioBias	曝光比偏差值，仅在多帧合成 WDR 模式下有效。 当 enExpRatioType 为 OP_TYPE_AUTO 时，该值越大，自动曝光比越大。默认值为 0x400，表示不对自动曝光比算法的计算结果进行调整。经过该值调整的曝光比会受到曝光比最大/最小值的限制。 取值范围：[0x0, 0xFFFF]
u16HighLightTarget	图像过曝程度的目标值，仅在使能 AdvanceAE 时有效。该值越大，图像过曝程度越大。默认值为 0x30。 取值范围：[0x0, 0x400]
u16ExpCoefMin	曝光系数最小值，仅在使能 AdvanceAE 时有效。 10bit 小数精度，0x400 表示曝光系数最小值为 1 倍，默认值为 0x400。 取值范围：[0x0, 0x400]

【注意事项】

- 建议 u16Tolerance 不要设置为太大的值，避免曝光比不能正确的反映场景动态范围变化。
- 针对某些对短帧最大曝光时间有限制的 sensor，曝光比较小时，长帧最大曝光时间较短，图像动态范围及噪声表现较差，导致自动曝光比计算不准确，此时建议限制最小曝光比，保证长帧最大曝光时间至少达到 3ms。如 Sony



IMX274@30fps, 短帧最大曝光时间为 680us, 此时建议限制最小曝光比为 1:5, 即设置 u32ExpRatioMin 为 0x140, 以保证图像质量。

- 建议 u16Speed 不要设置小于 0x8, 避免某些场景由于计算精度不足导致曝光比调节太慢甚至不调。u16Speed 过大可能导致曝光比变化太快, 造成画面亮度出现振荡。
- 建议手动模式下 u32ExpRatio 不要设置大于 0x400。如果曝光比大于 0x400, 在较亮的超宽动态场景, 适当调高曝光比会优化长帧图像噪声表现。但是在较暗或低宽动态场景, 如果曝光比过大, 由于短帧最大曝光时间会被压缩, 导致图像噪声表现会变差, 而且会出现明显的噪声不连续, 运动表现也会变差。
- 使能 AdvanceAE, 当曝光比达到设置的最大曝光比, 图像过曝程度仍未达到目标值时, 会适当降低 AE 亮度目标值, 以换取图像整体过曝程度的降低。通过合理设置 u16ExpCoefMin, 可使 AE 亮度目标值最多可以下降到 $u8Compensation * u16ExpCoefMin / 1024$ 。

【相关数据类型及接口】

- [HI_MPI_ISP_SetWDRExposureAttr](#)
- [HI_MPI_ISP_GetWDRExposureAttr](#)

ISP_HDR_EXPOSURE_ATTR_S

【说明】

定义 HDR 模式下的曝光属性。仅在 Hi3559AV100 中支持。

【定义】

```
typedef struct hiISP_HDR_EXPOSURE_ATTR_S
{
    ISP_OP_TYPE_E enExpHDLvType;
    HI_U32 u32ExpHDLv;
    HI_U32 u32ExpHDLvMax;
    HI_U32 u32ExpHDLvMin;
    HI_U32 u32ExpHDLvWeight;
} ISP_HDR_EXPOSURE_ATTR_S;
```

【成员】

成员名称	描述
enExpHDLvType	仅在 WDR-HDR 模式下有效, 在 linear-HDR 模式下不生



成员名称	描述
	效。 OP_TYPE_AUTO: 根据场景自动计算曝光等级; OP_TYPE_MANUAL: 手动配置曝光等级。
u32ExpHDRLv	仅在 HDR 模式下有效。 - 当 enExpHDRLvType 为 OP_TYPE_AUTO 时, u32ExpHDRLv 为 AE 算法根据场景自动计算的曝光等级。 - 当 enExpHDRLvType 为 OP_TYPE_MANUAL 时, u32ExpHDRLv 为手动配置的曝光等级, 值越大, 同样的场景, AE 曝光量越大。6bit 小数精度, 0x40 表示曝光等级为 1 倍。 取值范围: [0x40, 0x400]
u32ExpHDRLvMax	仅在 WDR-HDR 模式下有效,在 linear-HDR 模式下不生效。 u32ExpHDRLvMax 表示曝光等级最大值。在 enExpHDRLvType 为 OP_TYPE_AUTO 或 OP_TYPE_MANUAL 时均生效。6bit 小数精度, 0x40 表示曝光比为 1 倍。 取值范围: [0x40, 0x400]
u32ExpHDRLvMin	仅在 WDR-HDR 模式下有效,在 linear-HDR 模式下不生效。 u32ExpHDRLvMin 表示曝光等级最小值。在 enExpHDRLvType 为 OP_TYPE_AUTO 或 OP_TYPE_MANUAL 时均生效。6bit 小数精度, 0x40 表示曝光比为 1 倍。 取值范围: [0x40, u32ExpHDRLvMax]
u32ExpHDRLvWeight	曝光等级计算权重, 仅在 WDR-HDR 模式下有效,在 linear-HDR 模式下不生效。 当 enExpRatioType 为 OP_TYPE_AUTO 时, 该值越大, 曝光等级越大。默认值为 0xA0。 取值范围: [0x0, 0x400]



【相关数据类型及接口】

- [HI_MPI_ISP_SetHdrexposureAttr](#)
- [HI_MPI_ISP_GetHdrexposureAttr](#)

【注意事项】

WDR-HDR 模式是指前端 ISP 配置使用 WDR 多帧融合模式进行数据生成，linear-HDR 模式是指前端 ISP 配置使用线性模式进行数据生成。

ISP_SMART_EXPOSURE_ATTR_S

【说明】

定义智能模式下的曝光属性。

【定义】

```
typedef struct hiISP_SMART_EXPOSURE_ATTR_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bIRMode;
    ISP_OP_TYPE_E enSmartExpType;
    HI_U8    u8LumaTarget;
    HI_U16   u16ExpCoef;
    HI_U16   u16ExpCoefMax;
    HI_U16   u16ExpCoefMin;
    HI_U8    u8SmartInterval;
    HI_U8    u8SmartSpeed;
    HI_U16   u16SmartDelayNum;
} ISP_SMART_EXPOSURE_ATTR_S;
```

【成员】

成员名称	描述
bEnable	智能 AE 是否使能，默认为 HI_FALSE。
bIRMode	是否为红外模式，默认为 HI_FALSE。 HI_TRUE 为红外模式； HI_FALSE 为普通模式。



成员名称	描述
enSmartExpType	OP_TYPE_AUTO: 根据目标检测结果自动调整曝光系数; OP_TYPE_MANUAL: 手动配置曝光系数。
u8LumaTarget	人脸目标亮度, 此亮度为 YUV 图像中人脸区域 Y 分量均值。 取值范围: [0x0, 0xFF]
u16ExpCoef	基于原始曝光调整的曝光系数, 值越大, 图像整体亮度越高; 值越小, 图像整体亮度越低。 当 enSmartExpType 为 OP_TYPE_AUTO 时, u16ExpCoef 为 AE 算法根据目标检测结果计算的曝光系数。 当 enExpRatioType 为 OP_TYPE_MANUAL 时, u16ExpCoef 为手动配置的曝光系数。 10bit 小数精度, 0x400 表示曝光系数为 1 倍。 取值范围: [0x0, 0xFFFF]
u16ExpCoefMax	智能曝光系数最大值。 - 当 enSmartExpType 为 OP_TYPE_AUTO 时, u16ExpCoefMax 表示智能曝光系数最大值。 - 当 enSmartExpType 为 OP_TYPE_MANUAL 时, u16ExpCoefMax 无效。 10bit 小数精度, 0x400 表示曝光系数最大值为 1 倍。 取值范围: [0x0, 0xFFFF]
u16ExpCoefMin	智能曝光系数最小值。 当 enSmartExpType 为 OP_TYPE_AUTO 时, u16ExpCoefMin 表示智能曝光系数最小值。 当 enSmartExpType 为 OP_TYPE_MANUAL 时, u16ExpCoefMin 无效。 10bit 小数精度, 0x400 表示曝光系数最小值为 1 倍。 取值范围: [0x0, 0xFFFF]
u8SmartInterval	智能 AE 运行间隔。 取值为 1 时表示每帧运行, 取值为 n 时, 表示每 n 帧运行一



成员名称	描述
	次。 取值范围：[0x1, 0xFF]
u8SmartSpeed	智能 AE 调整速度。 默认值为 0x40，值越小，调整速度越慢；值越大，调整速度越快，也越容易出现震荡。 取值范围：[0x0, 0xFF]
u16SmartDelayNum	智能 AE 延迟恢复帧数。 默认值为 0xB4，值越小，人离开场景后的曝光恢复时间越短，值越大，人离开场景后的曝光恢复时间越长。 取值范围：[0x0, 0x400]

【注意事项】

- 建议 u8LumaTarget 设置范围为[90,130]，如果设置的值过于不合理，会导致曝光调整结果与智能检测结果相互影响，出现图像亮度振荡。同样基于曝光调整结果与智能检测结果相互影响的考虑，收敛后人脸亮度可能在 u8LumaTarget 左右一定范围内，属正常现象。
- 建议 u16ExpCoefMax 值不超过 0x1000,否则在人脸很暗的场景亮区会出现大面积过曝，而损失过多亮区细节。建议 u16ExpCoefMin 值不超过 0x100,否则在人脸很亮的场景亮区会出现暗区过黑，而损失过多暗区细节。
- 智能 AE 运行间隔 u8SmartInterval 受到 AE 自身运行间隔 u8AERunInterval 影响，如 u8AERunInterval 为 2，u8SmartInterval 为 3，则智能 AE 间隔实际为两者乘积，每六帧运行一次。
- 智能 AE 运行间隔 u8SmartInterval 建议与智能检测结果保持一致，如当前 Sensor 帧率为 25 帧，智能检测结果帧率为 8 帧，则建议运行间隔为 3。
- [HI_MPI_ISP_SetSmartInfo](#) 接口调用需要和智能检测帧率保持一致，如智能检测结果帧率为 10 帧，则每秒调用 10 次 [HI_MPI_ISP_SetSmartInfo](#)。
- 调整智能运行间隔 u8SmartInterval 与智能调整速度 u8SmartSpeed 均可以调整收敛速度，当运行间隔较小，调整速度过大时，图像可能会出现亮度震荡，此时可适当降低运行速度。



- 智能 AE 延迟恢复帧数 u16SmartDelayNum 可以调整人离开场景后的曝光恢复时间，该值设置过小时在人形人脸信息出现漏检的情况可能会导致图像亮度震荡，另外，适当设置较大值可以避免人连续进出的场景下图像亮度反复调节。
- 当人脸模型不可用，只有人形模型可用时，普通模式下计算的曝光系数不会低于 1 倍，红外模式下计算的曝光系数不会高于 1 倍。

【相关数据类型及接口】

- [HI_MPI_ISP_SetSmartExposureAttr](#)
- [HI_MPI_ISP_GetSmartExposureAttr](#)

ISP_AE_ROUTE_NODE_S

【说明】

定义 AE 分配路线节点属性。

【定义】

```
typedef struct hiISP_AE_ROUTE_NODE_S
{
    HI_U32  u32IntTime;
    HI_U32  u32SysGain;
    ISP\_IRIS\_F\_NO\_E  enIrisFNO;
    HI_U32  u32IrisFNOLin;
} ISP_AE_ROUTE_NODE_S;
```

【成员】

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



成员名称	描述
	取值范围：[1, 1024]

【注意事项】

- 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量必须单调递增，即后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。在计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益 $FNO = 1 \ll ISP\_IRIS\_F\_NO\_XX\_XX$ 。由此可知 F32.0 对应增益 1，F22.0 对应增益 2，F16.0 对应增益 4，以此类推，F1.0 对应增益 1024。
- 可以通过设置结构体 [ISP_PIRIS_ATTR_S](#) 中的 bFNOExValid 来决定实际生效的 AE route 节点光圈值采用 enIrisFNO 或 u32IrisFNOLin。bFNOExValid 为 HI_TRUE 时采用高精度的 u32IrisFNOLin，默认采用 enIrisFNO。enIrisFNO 会受到 [ISP_PIRIS_ATTR_S](#) 中 enMaxIrisFNOTarget/ enMinIrisFNOTarget 的限制，u32IrisFNOLin 会受到 [ISP_PIRIS_ATTR_S](#) 中 u32MaxIrisFNOTarget/ u32MinIrisFNOTarget 的限制。另外，enIrisFNO 和 u32IrisFNOLin 还会受到 [AE_SENSOR_DEFAULT_S](#) 中 enMaxIrisFNO/ enMinIrisFNO 的限制，所以对接 P-Iris 时要在 cmos.c 中给 enMaxIrisFNO/ enMinIrisFNO 赋合适的值。
- 不支持设置等曝光量节点。
- 为了保证曝光节点的曝光量，曝光节点的一个分量发生限制时，会对其他未达到最大值的分量进行调整，实际生效路径可能与设定路径不一致。为了防止曝光量溢出，如果光圈分量使能，则同一节点的曝光时间与增益的乘积最大值不能超过 0x1FFFFFFFFFFFFFFF，如果光圈分量不使能，则同一节点的曝光时间与 10bit 精度系统增益的乘积最大值不能超过 0x7FFFFFFFFFFFFFFF。
- 如果相邻节点的曝光量增加，则应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
- 光圈分量仅支持 P-Iris，不支持 DC-Iris，因为 DC-Iris 无法精确控制。
- 针对 P-Iris，建议将第一个节点的曝光时间、增益和光圈 F 值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈 F 值都设置为相应的最大目标值。

【相关数据类型及接口】

[HI_MPI_ISP_SetAERouteAttr](#)



ISP_AE_ROUTE_S

【说明】

定义 AE 曝光分配策略属性。

【定义】

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



说明

宏定义如下：

```
#define ISP_AE_ROUTE_MAX_NODES (16)
```

【成员】

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

【注意事项】

- 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含 sensor 模拟增益、sensor 数字增益、ISP 数字增益。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 分配策略是曝光时间优先，其次分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。在 DC_Iris 调节时，曝光量调节并不会参考 AE_ROUTE，因此实际生效的 AE_ROUTE 不会更新。
- 针对 P-Iris，默认 AE 分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。



- 在线进行 DC-Iris 和 P-Iris 切换，AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置 AE route。默认分配策略的 AE route 参数与切换时的 AE 曝光时间最小/最大值，增益最大/最小值和光圈分量最大/最小值相对应。注意：由于 cmos.c 也可以设置 AE 分配路线，若要在线进行 DC-Iris 和 P-Iris 切换，请不要在 cmos.c 中给 enIrisType 赋值，这样 AE 算法设置的光圈类型就会传递到 cmos.c 中，此时回调 cmos_get_ae_default 或 cmos_fps_set 等函数，才能得到与光圈类型相匹配的 AE route。

【相关数据类型及接口】

[HI_MPI_ISP_SetAERouteAttr](#)

ISP_AE_ROUTE_EX_NODE_S

【说明】

定义 AE 扩展分配路线节点属性。

【定义】

```
typedef struct hiISP_AE_ROUTE_EX_NODE_S
{
    HI_U32  u32IntTime;
    HI_U32  u32Again;
    HI_U32  u32Dgain;
    HI_U32  u32IspDgain;
    ISP\_IRIS\_F\_NO\_E enIrisFNO;
    HI_U32  u32IrisFNOLin;
} ISP_AE_ROUTE_EX_NODE_S;
```

【成员】

成员名称	描述
u32IntTime	节点曝光时间，单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]
u32Again	sensor 模拟增益，10bit 精度。 取值范围：[0x400, 0x3FFFFFF]
u32Dgain	sensor 数字增益，10bit 精度。 取值范围：[0x400, 0x3FFFFFF]



成员名称	描述
u32IspDgain	ISP 数字增益，10bit 精度。 取值范围：[0x400, 0x40000]
enIrisFNO	节点光圈 F 值大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]
u32IrisFNOLin	节点光圈 F 值等效增益大小，仅支持 P-Iris，不支持 DC-Iris。 取值范围：[1, 1024]

【注意事项】

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



- 针对 P-Iris，建议将第一个节点的曝光时间、增益和光圈 F 值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈 F 值都设置为相应的最大目标值。
- 节点的曝光时间不能设置为 0，也不能设置过小，导致以 us 为单位的曝光时间对应的曝光行数为 0，否则可能产生异常。
- 节点增益的范围相比非扩展 AE route 有所缩小。此外，为防止曝光量溢出，sensor 模拟增益、sensor 数字增益和 ISP 数字增益的乘积等效为的 10bit 精度系统增益不能大于 0xFFFFFFFF。如果光圈分量使能，则同一节点的曝光时间与 10bit 精度系统增益的乘积最大值不能超过 0x1FFFFFFFFFFFFFFF，如果光圈分量不使能，则同一节点的曝光时间与 10bit 精度系统增益的乘积最大值不能超过 0x7FFFFFFFFFFFFFFF。
- 若某段路线需要设置 ISP 数字增益优先，请注意不要把 ISP 数字增益用满，因为 ISP 数字增益还有弥补分配精度的作用，建议至少留下 2 倍 ISP 数字增益的余量，供弥补精度使用，即分配路线中节点的 ISPDgain 不要大于 MaxISPDgain/2。
- 开启增益分开配置功能时，曝光节点中的 ISP 数字增益在实际分配时可能有部分被分配到 WDRGain。

【相关数据类型及接口】

[HI_MPI_ISP_SetAERouteAttrEx](#)

ISP_AE_ROUTE_EX_S

【说明】

定义 AE 曝光分配策略扩展属性。

【定义】

```
typedef struct hiISP_AE_ROUTE_EX_S
{
    HI_U32 u32TotalNum;
    ISP\_AE\_ROUTE\_EX\_NODE\_S astRouteExNode[ISP_AE_ROUTE_EX_MAX_NODES];
} ISP_AE_ROUTE_EX_S;
```



说明

宏定义如下：

```
#define ISP_AE_ROUTE_EX_MAX_NODES (16)
```

【成员】



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

【注意事项】

- 最大支持 16 个节点，每个节点有曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益和光圈五个分量。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对 DC-Iris 和手动光圈镜头，默认 AE 扩展分配策略是曝光时间优先，再分配 sensor 模拟增益、sensor 数字增益、最后分配 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。在 DC_Iris 调节时，曝光量调节并不会参考 AE_ROUTE_EX，因此实际生效的 AE_ROUTE_EX 不会更新。
- 针对 P-Iris，默认 AE 扩展分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配 sensor 模拟增益、sensor 数字增益和 ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行 DC-Iris 和 P-Iris 切换，扩展 AE route 会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展 AE route。默认分配策略的 AE route 参数与切换时的 AE 曝光时间最小/最大值，增益最大/最小值和光圈分量最大/最小值相对应。注意：由于 cmos.c 也可以设置 AE 分配路线，若要在线进行 DC-Iris 和 P-Iris 切换，请不要在 cmos.c 中给 enIrisType 赋值，这样 AE 算法设置的光圈类型就会传递到 cmos.c 中，此时回调 cmos_get_ae_default 或 cmos_fps_set 等函数，才能得到与光圈类型相匹配的扩展 AE route。

【相关数据类型及接口】

[HI_MPI_ISP_SetAERouteAttrEx](#)

ISP_EXP_INFO_S

【说明】

定义 ISP 曝光内部状态信息。

【定义】



```
typedef struct hiISP_EXP_INFO_S
{
    HI_U32 u32ExpTime;
    HI_U32 u32ShortExpTime;
    HI_U32 u32MedianExpTime;
    HI_U32 u32LongExpTime;
    HI_U32 u32AGain;
    HI_U32 u32DGain;
    HI_U32 u32AGainSF;
    HI_U32 u32DGainSF;
    HI_U32 u32ISPDGain;
    HI_U32 u32Exposure;
    HI_BOOL bExposureIsMAX;
    HI_S16 s16HistError;
    HI_U32 au32AE_Hist1024Value[HIST_NUM];

    HI_U8 u8AveLum;
    HI_U32 u32LinesPer500ms;
    HI_U32 u32PirisFNO;
    HI_U32 u32Fps;
    HI_U32 u32ISO;
    HI_U32 u32ISOSF;
    HI_U32 u32ISOCalibrate;
    HI_U32 u32RefExpRatio;
    HI_U16 u16WdrExpCoef;
    HI_U32 u32FirstStableTime;
    HI_U32 u32QuickStarISO;
    ISP_AE_ROUTE_S stAERoute;
    ISP_AE_ROUTE_EX_S stAERouteEx;
    ISP_AE_ROUTE_S stAERouteSF;
    ISP_AE_ROUTE_EX_S stAERouteSFEx;
}ISP_EXP_INFO_S;
```



说明

宏定义如下：

```
#define HIST_NUM      (1024)
```

【成员】



成员名称	描述
u32ExpTime	当前曝光时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
u32ShortExpTime	FSWDR 模式下，表示当前短帧(S)曝光时间，单位为微秒(us)。线性模式不用关注该值。 取值范围：[0x0, 0xFFFFFFFF]
u32MedianExpTime	FSWDR 模式下，表示当前中帧(M)曝光时间，单位为微秒(us)。线性模式不用关注该值。 取值范围：[0x0, 0xFFFFFFFF]
u32LongExpTime	FSWDR 模式下，表示当前长帧曝光时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
u32AGain	当前 sensor 模拟增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]
u32DGain	当前 sensor 数字增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]
u32AGainSF	当前 sensor 短帧模拟增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]
u32DGainSF	当前 sensor 短帧数字增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]
u32ISPDGain	当前 ISP 数字增益，10bit 小数精度。 取值范围：[0x400, 0xFFFFFFFF]
u32Exposure	当前曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
bExposureIsMAX	0：ISP 未达到最大曝光水平； 1：ISP 达到最大曝光水平。
s16HistError	统计信息，AE 的目标亮度值与实际值的差，该值为正表示



成员名称	描述
	当前期望的亮度信息大于实际的亮度信息，该值为负表示期望的亮度信息小于实际的亮度信息。 取值范围：[-0x8000, 0x7FFF]
au32AE_Hist1024Value[1024]	全局 1024 段直方图统计信息 取值范围：[0x0, 0xFFFFFFFF]
u8AveLum	平均亮度信息 取值范围：[0x0, 0xFF]
u32LinesPer500ms	当前每 500ms 对应的曝光行数，可用于将曝光时间的单位由 us 转换成行数。 取值范围：[0x0, 0xFFFFFFFF]
u32PirisFNO	当前 P-Iris 光圈 F 值对应的等效增益。 取值范围：[0x0, 0x400]
u32Fps	实际图像帧率 * 100。 取值范围：[0x0, 0xFFFFFFFF]
u32ISO	当前 sensor 模拟增益*sensor 数字增益*ISP 数字增益*100，其中增益的精度都为 10bit。 取值范围：[0x64, 0xFFFFFFFF]
u32ISOSF	当前 sensor 短帧模拟增益*sensor 短帧数字增益*ISP 短帧数字增益*100，其中增益的精度都为 10bit。 取值范围：[0x64, 0xFFFFFFFF]
u32ISOCalibrate	标准 ISO，用于拍照 DCF 信息显示。 $u32ISOCalibrate = u32ISO * 256 / u16ISOCalCoef。$ 取值范围：[0x0, 0xFFFFFFFF]
u32RefExpRatio	参考曝光比，用于估计当前场景的动态范围，会受到 ISP_WDR_EXPOSURE_ATTR_S 中 Tolerance 和 Speed 等值的影响。 取值范围：[0x40, 0x4000]
u16WdrExpCoef	基于原始亮度目标值调整的曝光系数，仅在使能



成员名称	描述
	AdvanceAE 时有效，值越小，图像整体亮度越低。AE 实际亮度目标值为 $u8Compensation * u16WdrExpCoef / 1024$ 。 取值范围：[0x0, 0x400]
u32QuickStarISO	不带光敏 AE 启动快速收敛模式下，AE 快速计算出当前亮度环境下，收敛完成后大致的 ISO 值，用户可根据此 ISO 值决定是否要打开 IR CUT 和 IR LED。
u32FirstStableTime	首次 AE 收敛稳定的时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
stAERoute	实际生效的 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。
stAERouteEx	实际生效的扩展 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。
stAERouteSF	实际生效的短帧 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。
stAERouteSFEx	实际生效的短帧扩展 AE route，各个节点中的曝光时间以 us 为单位，增益为 10bit 精度，光圈取值范围为 [ISP_IRIS_F_NO_32_0, ISP_IRIS_F_NO_1_0]。光圈类型为 DC-Iris 时，节点光圈值不会对曝光量分配产生影响。

【注意事项】

- 曝光量的计算未考虑光圈状态，为曝光时间、sensor 模拟增益、sensor 数字增益、ISP 数字增益 4 者乘积，其中曝光时间的单位为曝光行数，曝光增益为 6bit 小数精度。若该值的精度不足以满足需求，可以根据高精度的曝光时间(us)和增益(10bit 小数精度)重新计算一个曝光量。



- ISP 是否达到最大曝光水平的计算未考虑光圈状态。如果当前曝光时间大于等于最大目标曝光时间，当前增益大于等于最大目标增益，那么就认为 ISP 达到最大曝光水平，否则就未达到最大曝光水平。
- 图像平均亮度经过归一化处理，取值范围为 0 ~ 255。
- u32Fps 可用于查询自动降帧状态下的实际帧率。
- u32ISO 和 u32RefExpRatio 都可作为 FSWDR 模式下切换到长帧模式的参考量。不过需要注意的是，若以 u32ISO 作为参考量，需要考虑到曝光比对于切换阈值的影响，避免来回在长帧模式和正常 WDR 模式之间切换；若以 u32RefExpRatio 作为参考量，长帧模式下亮区可能是过曝的，u32RefExpRatio 已不能正确表示场景动态范围，这会影响到长帧模式切换回正常 WDR 模式的判别。线性模式下 u32RefExpRatio 值可作为场景动态范围参考，当前场景动态范围越小，u32RefExpRatio 越小；当前场景动态范围越大，u32RefExpRatio 越大。
- u32AgainSF、u32DgainSF、u32ISOSF 在线性模式下默认为 1024、1024、100，但从其他模式切换到线性模式时保留上一次的值不更新。
- 当长短帧增益分开配置时，u32ISO、u32ISOSF 的结果还包含对应帧的 WDRGain。
- WDR 模式下，优先帧为长帧时，bExposureIsMAX 表示长帧是否达到最大曝光量，优先帧为短帧时，bExposureIsMAX 表示短帧是否达到最大曝光量。

【相关数据类型及接口】

[HI_MPI_ISP_QueryExposureInfo](#)

ISP_PRIOR_FRAME_E

【说明】

WDR 模式下的优先帧。

【定义】

```
typedef enum hiISP_PRIOR_FRAME_E
{
    LONG_FRAME      = 0,
    SHORT_FRAME     = 1,
    PRIOR_FRAME_BUTT
} ISP_PRIOR_FRAME_E;
```

【成员】



成员名称	描述
LONG_FRAME	优先帧为长帧。
SHORT_FRAME	优先帧为短帧。

【注意事项】

- 优先帧为长帧时，优先根据长帧的曝光路线设置长帧的曝光参数，其中 2 合 1WDR 模式下当增益分开配置时，短帧的曝光路线会根据长帧的曝光参数进行调整，优先帧为短帧时同理。默认优先帧为长帧。
- 优先帧切换时，会回调 `cmos_get_ae_default`，更新曝光时间最大/最小值，增益最大/最小值，光圈最大/最小值和 AE compensation 等参数，同时，AE 内部会将 `u16HistRatioSlope` 和 `u8MaxHistOffset` 更新为算法默认值。客户可以根据需要，在切换完成后调用 MPI 接口修改参数。
- 线性模式下优先帧设置无效。

【相关数据类型及接口】

无

3.5.3 AI

- [ISP_IRIS_STATUS_E](#): 定义 ISP 光圈状态。
- [ISP_IRIS_TYPE_E](#): 定义 ISP 光圈类型。
- [ISP_IRIS_F_NO_E](#): 定义 ISP 光圈 F 值。
- [ISP_MI_ATTR_S](#): 定义手动光圈属性。
- [ISP_DCIRIS_ATTR_S](#): 定义 DC-Iris AI 算法属性。
- [ISP_PIRIS_ATTR_S](#): 定义 P-Iris 属性。
- [ISP_IRIS_ATTR_S](#): 定义 ISP 光圈属性。
- [ISP_EXP_CONVERT_PARAM](#): 定义不同帧率等曝光量转换相关曝光参数属性。
- [ISP_EXP_PARAM_REG](#): 定义寄存器状态。

ISP_IRIS_STATUS_E

【说明】

定义 ISP 光圈状态。



【定义】

```
typedef enum hiISP_IRIS_STATUS_E
{
    ISP_IRIS_KEEP    = 0,
    ISP_IRIS_OPEN    = 1,
    ISP_IRIS_CLOSE   = 2,
    ISP_IRIS_BUTT
} ISP_IRIS_STATUS_E;
```

【成员】

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

【注意事项】

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

【相关数据类型及接口】

无

ISP_IRIS_TYPE_E

【说明】

定义 ISP 光圈类型。

【定义】

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




```
} ISP_IRIS_TYPE_E;
```

【成员】

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

【注意事项】

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

【相关数据类型及接口】

无

ISP_IRIS_F_NO_E

【说明】

定义 ISP 光圈 F 值。

【定义】

```
typedef enum hiISP_IRIS_F_NO_E
{
    ISP_IRIS_F_NO_32_0 = 0,
    ISP_IRIS_F_NO_22_0,
    ISP_IRIS_F_NO_16_0,
    ISP_IRIS_F_NO_11_0,
    ISP_IRIS_F_NO_8_0,
    ISP_IRIS_F_NO_5_6,
    ISP_IRIS_F_NO_4_0,
    ISP_IRIS_F_NO_2_8,
    ISP_IRIS_F_NO_2_0,
    ISP_IRIS_F_NO_1_4,
```



```
ISP_IRIS_F_NO_1_0,  
ISP_IRIS_F_NO_BUTT,  
} ISP_IRIS_F_NO_E;
```

【成员】

成员名称	描述	等效增益
ISP_IRIS_F_NO_32_0	光圈 F32.0。	1
ISP_IRIS_F_NO_22_0	光圈 F22.0。	2
ISP_IRIS_F_NO_16_0	光圈 F16.0。	4
ISP_IRIS_F_NO_11_0	光圈 F11.0。	8
ISP_IRIS_F_NO_8_0	光圈 F8.0。	16
ISP_IRIS_F_NO_5_6	光圈 F5.6。	32
ISP_IRIS_F_NO_4_0	光圈 F4.0。	64
ISP_IRIS_F_NO_2_8	光圈 F2.8。	128
ISP_IRIS_F_NO_2_0	光圈 F2.0。	256
ISP_IRIS_F_NO_1_4	光圈 F1.4。	512
ISP_IRIS_F_NO_1_0	光圈 F1.0。	1024

【注意事项】

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

【相关数据类型及接口】

无

ISP_MI_ATTR_S

【说明】

定义手动光圈属性。



【定义】

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

【成员】

成员名称	描述
u32HoldValue	AI 校正值，用于 DC-Iris 的调试。 取值范围为[0x0, 0x3E8]。
enIrisFNO	手动光圈大小，根据光圈 F 值进行区分，仅支持 P-Iris，不支持 DC-Iris。

【注意事项】

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

【相关数据类型及接口】

无

ISP_DCIRIS_ATTR_S

【说明】

定义 DC-Iris AI 算法属性。

【定义】

```
typedef struct hiISP_DCIRIS_ATTR_S
```



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

【成员】

成员名称	描述
s32Kp	比例增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值过小光线剧烈变化收敛过程中容易出现振荡，该值过大容易出现超调，也会导致振荡。该值的合理设置与电路特性和镜头相关。建议值为 7000。 取值范围为[0, 100000]。
s32Ki	积分增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值较小时，碰到强光收敛后画面会稳定在一个比较低的亮度上；该值较大时可能会导致强光场景光圈无法关闭。该值的合理设置与电路特性和镜头相关。建议值为 100，该值一般不需要修改。 取值范围为[0, 1000]。
s32Kd	微分增益，用于限制光线剧烈变化时光圈的开关速度，该值越大光线剧烈变化时光圈打开和关闭的速度越慢。该值过大对于瞬间变化的亮度过于敏感，会导致场景亮度快速变化时画面出现振荡。该值的合理设置与电路特性和镜头相关。建议值为 3000。 取值范围为[0, 100000]。
u32MinPwmDuty	最小 PWM 占空比。该值越小过曝时光圈关闭速度越快，但容易导致光圈来回震荡。该值的合理设置与电路特性和镜头相关。建议值为 250。 取值范围为[0, 1000]。



成员名称	描述
u32MaxPwmDuty	最大 PWM 占空比。该值越大画面全黑时光圈打开速度越快，该值过小则可能导致退出光圈控制区域时光圈仍未达到最大，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为 950。 取值范围为[0, 1000]。
u32OpenPwmDuty	光圈打开时的 PWM 占空比。当画面亮度稳定并且 PWM 占空比大于该值一段时间后，退出光圈控制区域。所以该值不能太小，否则容易导致光圈未达到最大就退出了光圈控制区域，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为 800。 取值范围为[0, 1000]。

【注意事项】

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

【相关数据类型及接口】

- [HI_MPI_ISP_SetDcirisAttr](#)
- [HI_MPI_ISP_GetDcirisAttr](#)

ISP_PIRIS_ATTR_S

【说明】

定义 P-Iris 属性。

【定义】

```
typedef struct hiISP_PIRIS_ATTR_S
{
    HI_BOOL bStepFNOTableChange;
    HI_BOOL bZeroIsMax;
    HI_U16 u16TotalStep;
    HI_U16 u16StepCount;
```



```
HI_U16 au16StepFNOTable[AI_MAX_STEP_FNO_NUM];  
ISP_IRIS_F_NO_E enMaxIrisFNOTarget;  
ISP_IRIS_F_NO_E enMinIrisFNOTarget;  
HI_BOOL bFNOExValid;  
HI_U32 u32MaxIrisFNOTarget;  
HI_U32 u32MinIrisFNOTarget;  
} ISP_PIRIS_ATTR_S;
```



说明

宏定义如下：

```
#define AI_MAX_STEP_FNO_NUM    (1024)
```

【成员】

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



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

【注意事项】

- 在线更换 P-Iris 镜头时，可以通过 MPI 设置该结构体，将新镜头的参数传递给 AE 算法和 P-Iris 驱动。
- bStepFNOTableChange 为只写寄存器，通过 MPI 获取该参数的值始终为 HI_FALSE。
- u16TotalStep 表示 P-Iris 步进电机的总步数，可用于标定光圈的起始位置。
u16StepCount 表示 P-Iris 步进电机的可用步数，一般小于 u16TotalStep，因为靠近光圈关闭端的位置在将光圈孔径转换为 F 值时误差较大，光圈调节过程中容易出现振荡，所以通常不会利用光圈关闭端附近的那些位置。当前最多支持 1024 步的 P-Iris 镜头，若 P-Iris 镜头的精度很高，可调节步数超过 1024，建议舍去光圈关闭端附近的那些位置。
- P-Iris 步进电机位置与光圈 F 值映射表 au16StepFNOTable 一般根据镜头原厂提供的步进电机位置与光圈孔径对应关系制作。由于 P-Iris 的控制通过 AE 分配路线进行，这就要求光圈 F 值与曝光时间和增益有较好的线性关系，因此映射表需要较高的精度。这里用 1024 表示 F1.0，512 表示 F1.4，以此类推，1 表示 F32.0。制作映射表时，可以根据最大光圈孔径对应的实际 F 值来规定映射表的最大值，

也可以不用管光圈达到最大时的实际 F 值是多少，统一将最大光圈孔径映射为 1024。同一款镜头处于不同焦距时，au16StepFNOTable 可能需要修改。

- 以福光 NV03105P 这款 P-Iris 镜头为例说明与镜头相关的参数该如何设置。表 3-1 中步进电机位置与孔径面积对应关系是镜头原厂提供的。该款镜头步进电机总步数为 93，步进电机位置为 0 时，光圈孔径面积最大，标称最大相对孔径为 F1.4，据此规定映射表 F 值最大值为 512。规定好最大值后，就可以计算出其他孔径面积对应的 F 值，因为孔径面积与 F 值成线性关系。比如说步进电机位置 1 对应的孔径面积为 48.835，得到 F 值为 $(48.835/49.366)*512=506$ ，以此类推，可以得到所有有效步进电机位置对应的 F 值。由下表可以看到，当镜头靠近关闭端时，孔径面积很小，与最大孔径面积相差上万倍，映射表 F 值精度已不足以表现出孔径面积的变化，得到的 F 值都为 0。实际上，哪怕提高映射表精度，由于镜头关闭端附近标称孔径面积与实际孔径面积一般相差较大，得到的 F 值也是不正确的，建议不要使用，否则在光圈调节过程中容易出现振荡。对于该款镜头，建议制作映射表时只取前 62 步，因此得到镜头相关的参数如表 3-2 所示。
- 步进电机位置与光圈 F 值映射表 au16StepFNOTable 的前 u16StepCount 个元素才是有效的，会被 AE 算法计算时用到，需要保证数组座标小于 u16StepCount 时，值单调递增，当数组座标为(u16StepCount-1)时，达到光圈最大 F 值。通过 MPI 设置 au16StepFNOTable 时，只有前 u16StepCount 个元素才会被写入寄存器保存；通过 MPI 获取 au16StepFNOTable 时，只有前 u16StepCount 个元素才会读出有效数值，其他值为 0。
- 光圈类型为 P-Iris 时，上海海思 AE 算法计算最大/最小曝光量会参考 enMaxIrisFNOTarget/enMinIrisFNOTarget(若 bFNOExValid 为 true，则参考 u32MaxIrisFNOTarget/u32MinIrisFNOTarget)的值，因此这几个值要与 au16StepFNOTable 的值相匹配，即最大/最小值都在 au16StepFNOTable 有效数据范围内。
- 在手动配置 DCF 光圈相关信息时，需要关闭 AI。如果需要配置的最大及实际光圈值为 ISP_IRIS_F_NO_E 中定义的 10 个标准值，配置 enMaxIrisFNOTarget 为最大光圈值，配置 enMinIrisFNOTarget 为实际光圈值即可。如果光圈不为 10 个标准值，则需要配置 bFNOExValid 为 HI_TRUE，以 F32.0 等效增益为 1 倍为基数计算实际光圈及最大光圈等效增益，并配置 u32MaxIrisFNOTarget 为最大光圈等效增益，配置 u32MinIrisFNOTarget 为实际光圈等效增益。如果 AI 使能，则根据实际光圈生效的大小计算光圈值，并更新 DCF 信息。光圈最小可支持到 F32.0。



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

步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
0	49.366	512	35	20.07	208	70	0.136	1
1	48.835	506	36	19.241	200	71	0.095	1
2	48.234	500	37	18.42	191	72	0.067	1
3	47.571	493	38	17.609	183	73	0.045	0
4	46.856	486	39	16.808	174	74	0.028	0
5	46.11	478	40	16.017	166	75	0.016	0
6	45.324	470	41	15.237	158	76	0.008	0
7	44.511	462	42	14.469	150	77	0.004	0
8	43.674	453	43	13.788	143	78	0.003	0
9	42.822	444	44	12.972	135	79	0.003	0
10	41.963	435	45	12.254	127	80	0.002	0
11	41.099	426	46	11.541	120	81	0.002	0
12	40.231	417	47	10.843	112	82	0.001	0
13	39.357	408	48	10.162	105	83	0.001	0
14	38.478	399	49	9.497	98	84	0.001	0
15	37.608	390	50	8.851	92	85	close	0
16	36.721	381	51	8.222	85	86	close	0
17	35.832	372	52	7.611	79	87	close	0
18	34.94	362	53	7.018	73	88	close	0
19	34.047	353	54	6.443	67	89	close	0
20	33.153	344	55	5.893	61	90	close	0
21	32.259	335	56	5.354	56	91	close	0
22	31.365	325	57	4.832	50	92	close	0
23	30.473	316	58	4.329	45	93	M-stop	0
24	29.582	307	59	3.843	40			
25	28.706	298	60	3.376	35			



步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
26	27.82	289	61	2.926	30			
27	26.937	279	62	2.494	26			
28	26.059	270	63	2.08	22			
29	25.184	261	64	1.684	17			
30	24.315	252	65	1.305	14			
31	23.451	243	66	0.949	10			
32	22.593	234	67	0.607	6			
33	21.741	225	68	0.374	4			
34	20.896	217	69	0.225	2			

表3-2 P-Iris 镜头相关参数，以福光 NV03105P 为例

参数名	参数值	备注
bZeroIsMax	HI_TRUE	步进电机位置 0 对应光圈最大孔径，因此该值为 HI_TRUE
u16TotalStep	93	步进电机总步数为 93
u16StepCount	62	步进电机可用步数为 62
au16StepFNOTable	{30,35,40,45,50,56,61,67,73,79,85,92,98,105,112,120,127,135,143,150,158,166,174,183,191,200,208,217,225,234,243,252,261,270,279,289,298,307,316,325,335,344,353,362,372,381,390,399,408,417,426,435,444,453,462,470,478,486,493,500,506,512}	根据表 3-1，取步进电机前 62 步对应的 F 值，按照从小到大的排列顺序，制作映射表。AE 算法计算时，只会用到 au16StepFNOTable 数组中的前 u16StepCount 个元素。当数组座标小于 u16StepCount 时，值必须保证是单调递增的。当数组座标为(u16StepCount-1) 时，达到光圈最大 F 值 512。



参数名	参数值	备注
enMaxIrisFNOTarget	ISP_IRIS_F_NO_1_4	映射表最大值 512 对应 F1.4
enMinIrisFNOTarget	ISP_IRIS_F_NO_5_6	映射表最小值 30 接近对应 F5.6
bFNOExValid	HI_FALSE	默认不采用高精度的光圈 F 值等效增益
u32MaxIrisFNOTarget	512	取 au16StepFNOTable 有效数据范围内的最大值 512
u32MinIrisFNOTarget	30	取 au16StepFNOTable 有效数据范围内的最小值 30

【相关数据类型及接口】

- [HI_MPI_ISP_SetPirisAttr](#)
- [HI_MPI_ISP_GetPirisAttr](#)
- [HI_MPI_ISP_SetAERouteAttr](#)
- [HI_MPI_ISP_SetAERouteAttrEx](#)

ISP_IRIS_ATTR_S

【说明】

定义 ISP 光圈属性。

【定义】

```
typedef struct hiISP_IRIS_ATTR_S
{
    HI_BOOL bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_IRIS_TYPE_E enIrisType;
    ISP_IRIS_STATUS_E enIrisStatus;
    ISP_MI_ATTR_S stMIAttr;
} ISP_IRIS_ATTR_S;
```

【成员】



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

【注意事项】

- 进行 AI 算法测试前，建议确认 AI 电路特性是否符合上海海思 IPC 要求。
- 针对 DC-Iris 镜头，AI 算法会根据画面亮度，调节 PWM 占空比对光圈进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE 直接返回，保持曝光时间和增益不变。当画面亮度稳定且 PWM 占空比维持在打开值一段时间后，AI 算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给 AE。处于光圈控制区时，更改 AE 算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE 会即时响应，根据新设定的参数和环境亮度，AI 算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭 AI 功能，否则 AE 的调节速度会受到一点影响。针对 DC-Iris 镜头建议一直打开 AI 功能，随意开关 AI 容易导致光圈控制出现异常。针对某些长焦的 DC-Iris 镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见 [ISP_DCIRIS_ATTR_S](#) 部分描述。
- 针对 P-Iris 镜头，光圈控制通过 AE 分配路线进行。P-Iris 对接重点在于正确设置镜头相关参数和合理设置 AE 分配路线，详见 [ISP_PIRIS_ATTR_S](#) 和 [HI_MPL_ISP_SetAERouteAttr](#) 部分描述，才能保证 P-Iris 正常工作。由于不同 P-Iris 的驱动方式可能会有差别，用户可以自行修改 P-Iris 驱动以适配不同镜头。
- 关闭 AI 功能，对于 DC-Iris 镜头，光圈会打开到最大；对于 P-Iris 镜头，光圈会打开到最大光圈目标值对应步进电机位置，但此时曝光分配仍会参考 AE 路线，可能导致画面亮度异常，因此对接 P-Iris 镜头时，若不想使能 AI，为保证曝光正常，需要把光圈类型切换至 `ISP_IRIS_DC_TYPE`。
- 由 DC-Iris 切换至 P-Iris，需提前设置好 P-Iris 镜头相关参数，并且将光圈状态设置为 `ISP_IRIS_KEEP`，再进行切换。



- Hi3559AV100 芯片利用上海海思 Demo 板或 Ref 板进行 DC-Iris 测试时，load ko 时要配置以下 3 个寄存器，用于配置管脚复用和时钟：himm 0x1805006C 0x1; himm 0x40020000 0x1; himm 0x40020018 0x210。

Hi3519AV100 芯片利用上海海思 DemoPlus 板进行 DC-Iris 测试时，load ko 时要配置以下寄存器，用于配置管脚复用和时钟：himm 0x04058080 0x1601。

Hi3516DV300 芯片利用上海海思 Demo 板进行 DC-Iris 测试时，load ko 时要配置以下寄存器，用于配置管脚复用和时钟：himm 0x111F0028 0x601; himm 0x120101BC 0x1800982。

Hi3516EV300 芯片利用上海海思 Demo 板进行 DC-Iris 测试时，load ko 时要配置以下寄存器，用于配置管脚复用和时钟：himm 0x100C00A0 0x1001; himm 0x120101BC 0x1183。

- Hi3559AV100 利用上海海思 Demo 板或 Ref 板进行 DC-Iris 测试时，需要保证 SW1.4 拨码位置为 1，PWM Number 可以通过模块参数设置接口配置。

注意：采用上海海思 AE 库时，AE_init 会打开一次默认的 PWM3 信号，若客户需要采用 PWM3 做其他处理，需要指定 AI 对应的 PWM Number 为其他值。

【相关数据类型及接口】

- [HI_MPI_ISP_SetIrisAttr](#)
- [HI_MPI_ISP_GetIrisAttr](#)

ISP_EXP_CONVERT_PARAM

【说明】

定义不同帧率等曝光量转换相关曝光参数属性。

【定义】

```
typedef struct {  
    HI_U32 u32TarFps;  
    HI_U32 u32TarIspDgain;  
    ISP_EXP_PARAM_REG stTimeReg[10];  
    ISP_EXP_PARAM_REG stAgainReg[10];  
    ISP_EXP_PARAM_REG stDgainReg[10];  
} ISP_EXP_CONVERT_PARAM;
```

【成员】



成员名称	描述
u32TarFps	输入参数，等曝光量转换目标帧率 * 100。 取值范围：[0x0, 0xFFFFFFFF]
u32TarIspDgain	等曝光量转换后的 ISP 数字增益，精度 10bit。
stTimeReg[10]	等曝光量转换后的 Sensor 曝光时间的寄存器值和寄存器地址。其成员变量 u32RegAddr 为 Sensor 寄存器的地址，u32RegValue 为寄存器的值。最大可转换输出 10 个寄存器值和地址。
stAgainReg[10]	等曝光量转换后的 Sensor 模拟增益的寄存器值和寄存器地址。其成员变量 u32RegAddr 为 Sensor 寄存器的地址，u32RegValue 为寄存器的值。最大可转换输出 10 个寄存器值和地址。
stAgainReg[10]	等曝光量转换后的 Sensor 数字增益的寄存器值和寄存器地址。其成员变量 u32RegAddr 为 Sensor 寄存器的地址，u32RegValue 为寄存器的值。最大可转换输出 10 个寄存器值和地址。

【注意事项】

无。

【相关数据类型及接口】

- [HI_MPI_ISP_SetExpConvert](#)
- [HI_MPI_ISP_GetExpConvert](#)

ISP_EXP_PARAM_REG

【说明】

定义寄存器状态。

【定义】

```
typedef struct {  
    HI_U32 u32RegAddr;  
    HI_U32 u32RegValue;  
} ISP_EXP_PARAM_REG;
```



【成员】

成员名称	描述
u32RegAddr	寄存器的地址。
u32RegValue	寄存器的值。

【注意事项】

无。

【相关数据类型及接口】

[ISP_EXP_CONVERT_PARAM](#)



4 AWB



说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

4.1 概述

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

4.2 重要概念

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

4.3 功能描述

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

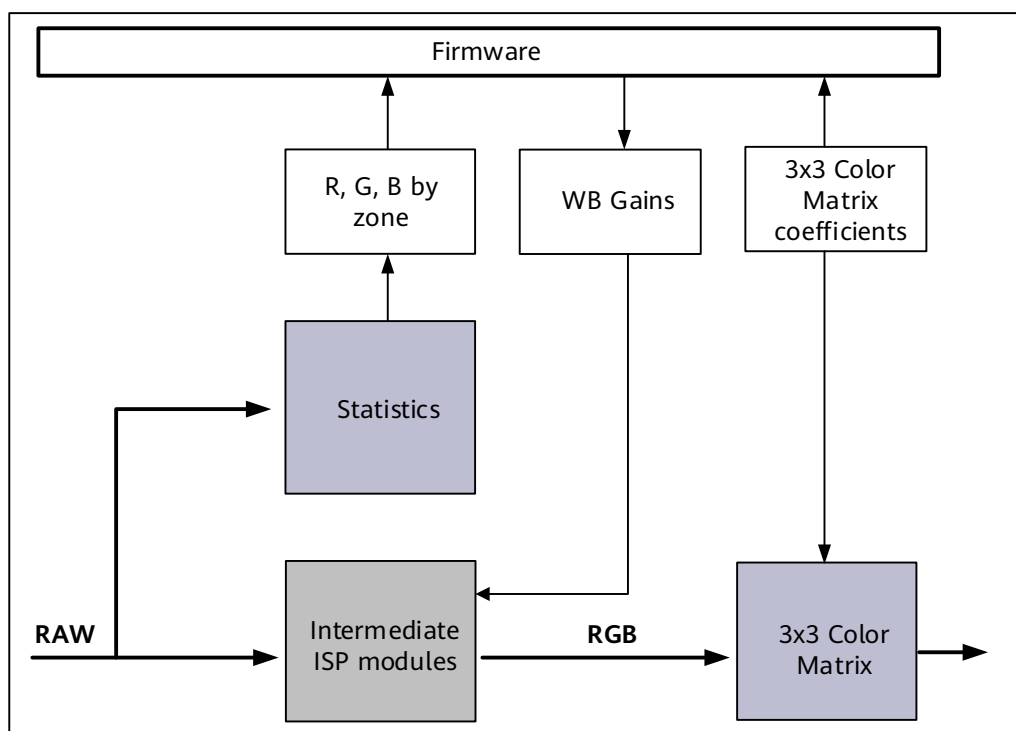
支持将图像分成 $M \times N$ (M 行 N 列) 区域, 统计每个区域的 R、G、B 均值以及参与统计的白点个数。

支持输出整幅图像的 R、G、B 均值以及参与统计的白点个数。

AWB 策略控制算法 firmware 在 lib_hiawb.a 和 lib_hiawb_natura.a 两个库中，这两个库的功能相同，针对不同应用场景做了相应的优化，使用的时候选择一个加载，不能同时加载。

AWB 工作原理如图 4-1 所示。

图4-1 AWB 工作原理图





4.4 API 参考

4.4.1 AWB 库接口

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

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

HI_MPI_AWB_Register

【描述】

向 ISP 注册 AWB 库。

【语法】

```
HI_S32 HI_MPI_AWB_Register(VI_PIPE ViPipe, ALG\_LIB\_S *pstAwbLib);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstAwbLib	AWB 算法库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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上海海思 AWB 库向 ISP 库注册。
- AWB 库可以注册多个实例，此接口不支持多进程操作。

【举例】

无

【相关主题】

[HI_MPI_ISP_AWBLibRegCallBack](#)

HI_MPI_AWB_UnRegister

【描述】

向 ISP 注销 AWB 库。

【语法】

```
HI_S32 HI_MPI_AWB_UnRegister(VI_PIPE ViPipe, ALG\_LIB\_S *pstAwbLib);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstAwbLib	AWB 算法库结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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 反注册回调接口 [HI_MPI_ISP_AWBLibRegCallBack](#)，以实现 AWB 向 ISP 库反注册的功能。
- 用户调用此接口完成上海海思 AWB 库向 ISP 库反注册，此接口不支持多进程操作。

【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 (VI_PIPE ViPipe, ALG\_LIB\_S *pstAwbLib,  
ISP_SNS_ATTR_INFO_S *pstSnsAttrInfo, AWB\_SENSOR\_REGISTER\_S *pstRegister);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstAwbLib	AWB 算法库结构体指针	输入
pstSnsAttrInfo	向 AWB 注册的 Sensor 的属性。	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 。

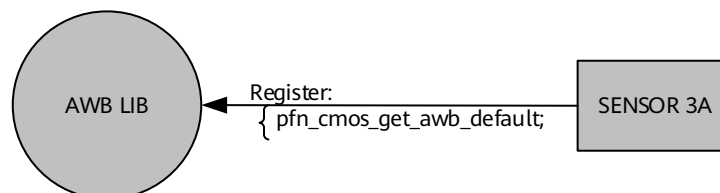
【需求】

- 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。
- 此接口不支持多进程操作。

图4-2 AWB 库 与 sensor 库间的接口



【举例】

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

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

VI_PIPE ViPipe = 0;
stLib.s32Id = 0;
```



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

【相关主题】

无

HI_MPI_AWB_SensorUnRegCallBack

【描述】

AWB 库提供的 sensor 注销的回调接口。

【语法】

```
HI_S32 HI_MPI_AWB_SensorUnRegCallBack (VI_PIPE ViPipe, ALG\_LIB\_S *pstAwbLib, SENSOR_ID
SensorId);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstAwbLib	AWB 算法库结构体指针	输入
SensorId	向 AWB 注册的 Sensor 的 Id。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。



【需求】

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

【注意】

此接口不支持多进程操作。

【相关主题】

无

4.4.2 AWB 控制模块

- [HI_MPI_ISP_SetWBAttr](#)：设置白平衡属性。
- [HI_MPI_ISP_GetWBAttr](#)：获取白平衡属性。
- [HI_MPI_ISP_SetAWBAttrEx](#)：设置自动白平衡扩展属性。
- [HI_MPI_ISP_GetAWBAttrEx](#)：获取自动白平衡扩展属性。
- [HI_MPI_ISP_QueryWBInfo](#)：获取当前白平衡增益系数，检测色温，饱和度值，颜色校正矩阵系数。
- [HI_MPI_ISP_CalGainByTemp](#)：计算特定色温下的白平衡增益系数。
- [HI_MPI_ISP_SetSpecAwbAttr](#)：设定机器学习白平衡的属性，该属性主要来源于标定结果。
- [HI_MPI_ISP_GetSpecAwbAttr](#)：获取机器学习白平衡的属性。
- [HI_MPI_ISP_SetCaaControlAttr](#)：设定色温转换表的属性。
- [HI_MPI_ISP_GetCaaControlAttr](#)：获取色温转换表属性。
- [HI_MPI_ISP_SetSpecAwbControlAttr](#)：设定机器学习白平衡的控制参数。
- [HI_MPI_ISP_GetSpecAwbControlAttr](#)：获取机器学习白平衡的控制参数。

HI_MPI_ISP_SetWBAttr

【描述】

设置白平衡属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetWBAttr(VI_PIPE ViPipe, const ISP\_WB\_ATTR\_S *pstWBAttr);
```



【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstWBAttr	白平衡的参数属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

- 白平衡控制类型为自动时，AWB 算法自动调节白平衡系数。
- 白平衡控制类型为手动时，AWB 算法失效，需自行设定 Rgain、Ggain、Bgain。
- 环境色温、照度会影响白点的分布范围，上海海思 AWB 算法在运行过程中，会根据环境参数自动刷新 AWB Byaer 域统计信息的白点参数。用户希望在上海海思 AWB 算法运行时修改 AWB 统计参数，需要调用 [HI_MPI_ISP_SetWBAttr](#) 接口，关闭统计参数自动刷新功能。用户关闭统计参数自动刷新功能，到 ISP 收到配置并响应，有 2 帧的延时。因此，用户设置后需要等待 2 帧时间再修改 AWB 统计参数。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetWBAttr](#)



HI_MPI_ISP_GetWBAttr

【描述】

获取白平衡属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetWBAttr(VI_PIPE ViPipe, ISP\_WB\_ATTR\_S *pstWBAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstWBAttr	白平衡的参数属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetWBAttr](#)



HI_MPI_ISP_SetAWBAttrEx

【描述】

设置自动白平衡扩展属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetAWBAttrEx(VI_PIPE ViPipe, ISP\_AWB\_ATTR\_EX\_S *pstAWBAttrEx);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstAWBAttrEx	高级白平衡属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

当 [HI_MPI_ISP_SetWBAttr](#) 接口成员 pstWBAttr->enAlgType 为 AWB_ALG_ADVANCE 时，此接口才有效。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetAWBAttrEx](#)



HI_MPI_ISP_GetAWBAttrEx

【描述】

获取自动白平衡扩展属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetAWBAttrEx(VI_PIPE ViPipe, ISP\_AWB\_ATTR\_EX\_S *pstAWBAttrEx);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstAWBAttrEx	扩展白平衡属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetAWBAttrEx](#)



HI_MPI_ISP_QueryWBInfo

【描述】

获取当前白平衡增益系数，检测色温，饱和度值，颜色校正矩阵系数。

【语法】

```
HI_S32 HI_MPI_ISP_QueryWBInfo(VI_PIPE ViPipe, ISP_WB_INFO_S *pstWBInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号。	输入
pstWBInfo	颜色相关状态参数。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

若用户使用非上海海思 AWB 算法，则该接口需要自己实现并且需要同步修改 PQTOOLS 的 xml 文件，具体修改方式请参考《图像质量调试工具使用指南》第 2.1.3.7 小节。

【举例】

无

【相关主题】

无



HI_MPI_ISP_CalGainByTemp

【描述】

计算特定色温下的白平衡增益系数。

【语法】

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

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstWBAttr	白平衡的参数属性。需要用到 ISP_AWB_ATTR_S 中的 au16StaticWB, as32CurvePara, bGainNormEn 三组参数。	输入
u16ColorTemp	色温值, 单位为 Kelvin。 取值范围: [1500, 15000]	输入
s16Shift	白点与 Planckian 曲线的位置和距离。 取值范围: [-64, 64]	输入
pu16AWBGain	预设色温下的 R, Gr, Gb, B 四个通道增益。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 错误码 。

【需求】

- 头文件: hi_awb_comm.h、 mpi_awb.h
- 库文件: libisp.a, lib_hiawb.a



【注意】

- 白平衡标定参数决定预设色温增益计算的准确性，因此，在调用该接口前，必须完成标定，且标定参数配置到 ISP。用户先通过 [HI_MPI_ISP_GetWBAttr](#) 接口获取标定结果，再调用 [HI_MPI_ISP_CalGainByTemp](#) 接口计算预设色温，过程中不能再修改标定结果。
- [HI_MPI_ISP_CalGainByTemp](#) 功能基于 AWB 标定参数实现，因此，该接口仅对 [ISP_WB_ATTR_S](#) 结构体的标定参数 `au16StaticWB`、`as32CurvePara` 做有效范围检查，不对其他参数做约束。
- [ISP_AWB_ATTR_S](#) 结构体的 `bGainNorm` 参数会影响白平衡增益值。`bGainNorm` 为 0 时，最小增益为 0x100；`bGainNorm` 为 1 时，最小增益和 sensor 黑电平相关。
- `s16Shift` 参数决定光源点与 Planckian 曲线的位置关系，`s16Shift` 为负数时，光源点位于 Planckian 曲线左侧，计算的白平衡增益在预设色温下会稍微偏红；`s16Shift` 为正数时，光源点位于 Planckian 曲线右侧，计算的白平衡增益在预设色温下会稍微偏绿。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetWBAttr](#)

HI_MPI_ISP_SetSpecAwbAttr

【描述】

设置机器学习白平衡属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetSpecAwbAttr(VI_PIPE ViPipe, const ISP\_SPECAWB\_ATTR\_S
*pstSpecAwbAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbAttr	机器学习 AWB 算法属性，该属性主要来源于标定结果	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_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

- 该接口设定的属性除了 u16Fno 外都来自于标定结果。
- Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口，需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECAWB_SUPPORT 1
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200 默认支持该接口

【举例】

无

【相关主题】

[HI_MPI_ISP_GetSpecAwbAttr](#)

HI_MPI_ISP_GetSpecAwbAttr

【描述】

获取机器学习 AWB 算法的属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetSpecAwbAttr(VI_PIPE ViPipe, ISP\_SPECAWB\_ATTR\_S *pstSpecAwbAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbAttr	机器学习 AWB 算法属性, 该属性主要来源于标定结果	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_awb_comm.h、mpi_awb.h
- 库文件: libisp.a

【注意】

Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口, 需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECAWB_SUPPORT 1。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetSpecAwbAttr](#)

HI_MPI_ISP_SetCaaControlAttr

【描述】

设定色温转换表属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetCaaControlAttr(VI_PIPE ViPipe, const ISP\_SPECAWB\_CAA\_CONTROL\_S
*pstSpecAwbCaaAttr);
```




【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbCaaAttr	色温转换表属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

- 色温转换表由 PQ tools 工具结合用户标定结果生成，用户可以通过该接口将 PQ tools 生成的实际色温转换 lut 写入 ISP 观察效果。
- Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口，需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECAWB_SUPPORT 1。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetCaaControlAttr](#)

HI_MPI_ISP_GetCaaControlAttr

【描述】

获取色温转换表属性。



【语法】

```
HI_S32 HI_MPI_ISP_GetCaaControlAttr(VI_PIPE ViPipe, ISP\_SPECAWB\_CAA\_CONTROL\_S  
*pstSpecAwbCaaAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbCaaAttr	色温转换表属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口，需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECAWB_SUPPORT 1。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetCaaControlAttr](#)

HI_MPI_ISP_SetSpecAwbControlAttr

【描述】



设置机器学习白平衡控制参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetSpecAwbControlAttr(VI_PIPE ViPipe, ISP\_SPECAWB\_CONTROL\_ATTR\_S  
*pstSpecAwbCtrlAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbCtrlAttr	机器学习 AWB 控制参数	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_awb_comm.h、mpi_awb.h
- 库文件：libisp.a, lib_hiawb_natura.a

【注意】

- Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口，需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECARB_SUPPORT 1。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200 默认支持该接口

【举例】

无

【相关主题】

- [HI_MPI_ISP_GetSpecAwbControlAttr](#)



- [ISP_SPECAXB_CONTROL_ATTR_S](#)

HI_MPI_ISP_GetSpecAwbControlAttr

【描述】

获取机器学习白平衡控制参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetSpecAwbControlAttr(VI_PIPE ViPipe, ISP\_SPECAXB\_CONTROL\_ATTR\_S  
*pstSpecAwbCtrlAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSpecAwbCtrlAttr	机器学习 AWB 控制参数	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_awb_comm.h、mpi_awb.h
- 库文件：libisp.a, lib_hiawb_natura.a

【注意】

- Hi3516EV200/Hi3516EV300 默认不支持该接口。Hi3516EV200/Hi3516EV300 要支持该接口，需要在 hi_comm_isp.h 中加上 #define CONFIG_HI_ISP_SPECAXB_SUPPORT 1。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200 默认支持该接口。

【举例】



无

【相关主题】

- [HI_MPI_ISP_SetSpecAwbControlAttr](#)
- [ISP_SPECAWB_CONTROL_ATTR_S](#)

4.5 数据类型

4.5.1 Register

- [AWB_SENSOR_REGISTER_S](#): 定义 sensor 注册结构体。
- [AWB_SENSOR_EXP_FUNC_S](#): 定义 sensor 回调函数结构体。
- [AWB_SENSOR_DEFAULT_S](#): 定义 AWB 算法库的初始化参数结构体。
- [AWB_SPEC_SENSOR_DEFAULT_S](#): 定义 SPECAWB 算法的初始化参数结构体。
- [AWB_AGC_TABLE_S](#): 定义饱和度初始化参数结构体。
- [AWB_CCM_TAB_S](#): 定义不同色温下自动颜色校正矩阵系数。
- [AWB_CCM_S](#): 定义 CCM 颜色校正矩阵属性。

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)(VI_PIPE ViPipe, AWB\_SENSOR\_DEFAULT\_S *pstAwbSnsDft);
    HI_S32(*pfn_cmos_get_awb_spec_default)(VI_PIPE ViPipe, AWB\_SPEC\_SENSOR\_DEFAULT\_S
    *pstAwbSpecSnsDft);
} AWB_SENSOR_EXP_FUNC_S;
```

【成员】

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

【注意事项】

无

【相关数据类型及接口】

[ISP_SENSOR_REGISTER_S](#)

AWB_SENSOR_DEFAULT_S

【说明】

定义 AWB 算法库的初始化参数结构体。

【定义】



```
typedef struct hiAWB_SENSOR_DEFAULT_S
{
    HI_U16  u16WbRefTemp;
    HI_U16  au16GainOffset[4];
    HI_S32  as32WbPara[6];
    HI_U16  u16GoldenRgain;
    HI_U16  u16GoldenBgain;
    HI_U16  u16SampleRgain;
    HI_U16  u16SampleBgain;
    AWB_AGC_TABLE_S stAgcTbl;
    AWB_CCM_S stCcm;
    HI_U16  u16InitRgain;
    HI_U16  u16InitGgain;
    HI_U16  u16InitBgain;
    HI_U8   u8AWBRunInterval;
    HI_U16  au16InitCCM[CCM_MATRIX_SIZE];
} AWB_SENSOR_DEFAULT_S;
```

【成员】

成员名称	子成员名称	描述
u16WbRefTemp	-	静态白平衡校正色温，取值范围为[0, 0xFFFF]。
au16GainOffset	-	静态白平衡的 R、Gr、Gb、B 颜色通道的增益，取值范围为[0, 0xFFFF]。
as32WbPara	-	校正工具给出的白平衡参数，取值范围为 [0, 0xFFFFFFFF]。
u16GoldenRgain	-	Golden 样机 AWB 在线标定得到的 G/R 值。
u16GoldenBgain	-	Golden 样机 AWB 在线标定得到的 G/B 值。
u16SampleRgain	-	当前样机 AWB 在线标定得到的 G/R 值。
u16SampleBgain	-	当前样机 AWB 在线标定得到的 G/B 值。
stAgcTbl	bValid	该结构体的数据是否有效，取值范围为[0,



成员名称	子成员名称	描述
		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	u16CCMTabNum	当前配置的 CCM 的组数。 取值范围：[3, 7]
	astCCMTab[CCM_MATRIX_NUM]	不同色温下的颜色校正矩阵和对应的色温值。
u16InitRgain	-	ISP 启动时 AWB 初始 R 通道增益。
u16InitGgain	-	ISP 启动时 AWB 初始 G 通道增益。
u16InitBgain	-	ISP 启动时 AWB 初始 B 通道增益。
u8AWBRunInterval	-	ISP 启动时 AWB 工作频率。 取值范围：[0x1, 0xFF]
au16InitCCM	-	ISP 启动时 CCM 初始值。

【注意事项】

参考色温即静态白平衡校正的环境色温，需要提供色度计测量的实际值。

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

AWB_SPEC_SENSOR_DEFAULT_S

【说明】

定义 SPECAWB 算法的初始化参数结构体。

【定义】

```
typedef struct hiAWB_SPEC_SENSOR_DEFAULT_S
{
    ISP\_SPECAWB\_ATTR\_S stSpecAwbAttrs;
    ISP\_SPECAWB\_CAA\_CONTROL\_S stCaaControl;
} AWB_SPEC_SENSOR_DEFAULT_S;
```




【成员】

成员名称	子成员名称	描述
stSpecAwbAttrs	参考 ISP_SPECAWB_ATTR_S	光谱分析 AWB 算法属性
stCaaControl	参考 ISP_SPECAWB_CAA_CONTROL_S	色温转换表属性

【注意事项】

无

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

AWB_AGC_TABLE_S

【说明】

定义饱和度初始化参数结构体。

【定义】

```
typedef struct hiAWB_AGC_TABLE_S
{
    HI_BOOL bValid;
    HI_U8 au8Saturation[ISP\_AUTO\_ISO\_STRENGTH\_NUM];
} AWB_AGC_TABLE_S;
```

【成员】

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



【注意事项】

au8Saturation 为非单调递减序列。

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

AWB_CCM_TAB_S

【说明】

定义不同色温下自动颜色校正矩阵系数

【定义】

```
typedef struct hiAWB_CCM_TAB_S
{
    HI_U16 u16ColorTemp;
    HI_U16 au16CCM[CCM_MATRIX_SIZE];
} AWB_CCM_TAB_S;
```

【成员】

成员名称	描述
u16ColorTemp	当前配置的 CCM 对应的色温。 取值范围：[500, 30000]
au16CCM[CCM_MATRIX_SIZE]	不同色温下的颜色校正矩阵，8bit 小数精度。Bit [15]是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。Bit[12],Bit[13],Bit[14]是无效的，即正数从 0x0000 到 0x0FFF，负数从 0x8000 到 0x8FFF，是 CCM 的合理参数。 取值范围：[0x0, 0xFFFF]

【注意事项】

无

【相关数据类型及接口】

无



AWB_CCM_S

【说明】

定义 CCM 颜色校正矩阵属性。

【定义】

```
typedef struct hiAWB_CCM_S
{
    HI_U16  u16CCMTabNum;
    AWB_CCM_TAB_S astCCMTab[CCM_MATRIX_NUM];
}AWB_CCM_S;
```



说明

宏定义如下：

```
#define CCM_MATRIX_NUM          (7)
```

【成员】

成员名称	描述
u16CCMTabNum	当前配置的 CCM 的组数。 取值范围：[3, 7]
astCCMTab[CCM_MATRIX_NUM]	不同色温下的颜色校正矩阵和对应的色温值。

【注意事项】

如果选择三组 CCM，推荐高色温 CCM 在 D50 光源标定，中色温 CCM 在 TL84 光源标定，低色温 CCM 在 A 光源标定。

【相关数据类型及接口】

[ISP_SENSOR_EXP_FUNC_S](#)

4.5.2 WB

- [ISP_AWB_ATTR_S](#)：定义 ISP 自动白平衡属性。
- [ISP_AWB_CBCR_TRACK_ATTR_S](#)：定义 Bayer 域统计信息的联动参数。
- [ISP_AWB_LUM_HISTGRAM_ATTR_S](#)：定义白平衡的亮度直方图统计参数。
- [ISP_AWB_ALG_TYPE_E](#)：定义白平衡的计算方式属性。
- [ISP_AWB_CT_LIMIT_ATTR_S](#)：定义白平衡的增益范围限制属性。



- [ISP_MWB_ATTR_S](#): 定义 ISP 手动白平衡属性。
- [ISP_WB_ATTR_S](#): 定义白平衡属性。
- [ISP_AWB_MULTI_LS_TYPE_E](#): 定义混合光源下的 AWB 策略。
- [ISP_AWB_INDOOR_OUTDOOR_STATUS_E](#): 定义室内外状态。
- [ISP_AWB_ATTR_EX_S](#): 定义自动白平衡扩展属性。
- [ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S](#): 定义独立光源点的信息。
- [ISP_AWB_IN_OUT_ATTR_S](#): 定义对场景做室内外判断的参数。
- [ISP_WB_INFO_S](#): 定义白平衡, 饱和度, 颜色校正信息。

ISP_AWB_ATTR_S

【说明】

定义 ISP 自动白平衡属性。

【定义】

```
typedef struct hiISP_AWB_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16 u16RefColorTemp;
    HI_U16 au16StaticWB[ISP\_BAYER\_CHN\_NUM];
    HI_S32 as32CurvePara[AWB\_CURVE\_PARA\_NUM];
    ISP\_AWB\_ALG\_TYPE\_E enAlgType;
    HI_U8 u8RGStrength;
    HI_U8 u8BGStrength;
    HI_U16 u16Speed;
    HI_U16 u16ZoneSel;
    HI_U16 u16HighColorTemp;
    HI_U16 u16LowColorTemp;
    ISP\_AWB\_CT\_LIMIT\_ATTR\_S stCTLimit;
    HI_BOOL bShiftLimitEn;
    HI_U8 u8ShiftLimit;
    HI_BOOL bGainNormEn;
    HI_BOOL bNaturalCastEn;
    ISP\_AWB\_CBCR\_TRACK\_ATTR\_S stCbCrTrack;
    ISP\_AWB\_LUM\_HISTGRAM\_ATTR\_S stLumaHist;
    HI_BOOL bAWBZoneWtEn;
```



```
HI_U8  au8ZoneWt[AWB_ZONE_ORIG_ROW * AWB_ZONE_ORIG_COLUMN];
} ISP_AWB_ATTR_S;
```



说明

宏定义如下：

- #define AWB_ZONE_ORIG_ROW (32)
- #define AWB_ZONE_ORIG_COLUMN (32)
- #define AWB_CURVE_PARA_NUM (6)

【成员】

成员名称	描述
bEnable	自动白平衡使能，默认 HI_TRUE。
u16RefColorTemp	校正静态白平衡系数的光源色温值。 推荐选用 5000K 附近光源进行静态白平衡校正。
au16StaticWB[4]	静态白平衡系数。 取值范围：[0x0, 0xFFF]
as32CurvePara[6]	校准曲线的系数。 取值范围：as32CurvePara[3]!=0; as32CurvePara[4]=128。
enAlgType	自动白平衡算法类型选择。 取值范围：AWB_ALG_LOWCOST、 AWB_ALG_ADVANCE。
u8RGStrength	自动白平衡 R 通道校准强度。 取值范围：[0x0, 0xFF]
u8BGStrength	自动白平衡 B 通道校准强度。 取值范围：[0x0, 0xFF]
u16Speed	自动白平衡算法收敛速度。 取值范围：[0x0, 0xFFF]
u16ZoneSel	自动白平衡算法全局或分区域计算的选择，取值范围：[0, 255]
u16HighColorTemp	自动白平衡算法的色温上限。



成员名称	描述
	推荐范围：[8500, 10000]
u16LowColorTemp	自动白平衡算法的色温下限。 取值范围：[0x0, u8HighColorTemp)
stCTLimit	白平衡算法环境色温超过预设色温范围后，增益计算模式及手动增益值设定。
bShiftLimitEn	自动白平衡算法的将超出白点范围的增益映射回白点范围内的开关。
u8ShiftLimit	室内场景或室内外检测关闭时，自动白平衡的白点范围参数。 取值范围：[0x0, 0xFF]
bGainNormEn	自动白平衡算法的增益归一化的开关。
bNaturalCastEn	低色温下 AWB 风格喜好开关。
stCbCrTrack	Bayer 域统计信息与 ISO 的联动参数
stLumaHist	白平衡的亮度直方图统计参数
bAWBZoneWtEn	白平衡的分块权重使能开关，默认 HI_FALSE。
au8ZoneWt	白平衡的 1024 分块权重表，取值范围：[0x0, 0xFF]。

【差异说明】

无

【注意事项】

- bEanble 为 HI_TRUE 时，AWB 正常工作，RGB 通道增益系数由 AWB 根据环境色温计算；bEnable 为 HI_FALSE 时，AWB 不工作，RGB 通道增益系数固定，为校正的静态白平衡系数。
- u16RefColorTemp、au16StaticWB[4]、as32CurvePara[6]是用户通过校正工具得到的 AWB 参数，是 AWB 准确的前提。更换光学器件后，建议重新校正这三组参数。u16RefColorTemp 是进行静态白平衡校正光源的真实色温，推荐在 D50 光

源下做静态白平衡校正。as32CurvePara[0], as32CurvePara[1], as32CurvePara[2]三个参数确定 Planckian 曲线, as32CurvePara[3], as32CurvePara[4], as32CurvePara[5]三个参数确定色温曲线, as32CurvePara[4]取值固定为 128。

- 自动白平衡 R/B 通道校准强度, 一般情况不建议调整。

可通过调整 R/B 通道校准强度使 AWB 校准偏强或者偏弱, 0x80 表示标准强度。建议 u8RGStrength、u8BGStrength 两个参数取相同值, 且小于等于 0x80。在低色温场景, 校准强度小于 0x80 时图像偏红, 大于 0x80 时图像偏蓝; 在高色温场景, 校准强度小于 0x80 时图像偏蓝, 大于 0x80 时图像偏红。

- u16Speed 控制 AWB 增益的收敛速度。u16Speed 越大, 切换光源时, AWB 收敛的速度越快, 但帧间的波动幅度也变大; u16Speed 越小, 切换光源时, AWB 收敛的速度越慢, 但帧间的波动幅度变小, 稳定性提高。
- u16ZoneSel 参数为 0 或 255 时, 采用了近似灰度世界的算法; u16ZoneSel 为其他值时, AWB 算法会对所有统计块进行分类筛选, 提高 AWB 精度。
- 自动白平衡算法的色温上限/下限, 以 Kelvin 为单位。

AWB 算法支持的最高/最低色温。若实际场景中的色温大于色温上限或小于色温下限, 则 AWB 不能完全恢复, 图像会偏向光源色, 低色温下偏黄, 高色温下偏蓝。

- bShiftLimitEn 决定 AWB 增益是否向 Planckian 曲线做映射。u8ShiftLimit 控制白色块的范围, 取值较小时, 对光源的支持范围较窄, 精度较高; 取值较大时, 支持更多的光源, 精度降低。bShiftLimitEn 仅控制 AWB 增益是否向 Planckian 曲线做映射, 无论 bShiftLimitEn 打开或关闭, u8ShiftLimit 取值的改变都会影响 AWB 结果。校正静态白平衡系数的光源色温值, 要求在 AWB 算法设置的色温上下限范围内。
- bGainNormEn 使能, 对 RGB 通道增益进行限制, 可以改善低色温、低照度场景的信噪比。
- bNaturalCastEn 使能, 低色温下 AWB 保留光源色, 图像颜色更自然。
- 权重表与 AWB 统计分块的对应顺序如下:
 - 非拼接模式下, AWB 统计信息分块个数是 32x32, au8ZoneWt[0]对应第一行第一列, au8ZoneWt[1]对应第一行第二列, u8ZoneWt[31]对应第一行第 32 列, au8ZoneWt[32]对应第二行第一列, 依次类推。
 - 不支持拼接模式下配置 AWB 统计信息权重。
- 特定应用产品, 客户使能 bAWBZoneWtEn 后, 可通过设置权重表改变每个区域的权重, 优化 AWB 表现。在 Shading 较严重时, 可加大中心区域的权重, 保证



中间区域 AWB 准确还原，降低 Shading 对 AWB 的影响。在行车记录仪应用，感兴趣区域一般在画面的中心偏下区域，可加大此区域的权重，降低天空、树木等区域对 AWB 的干扰。

【相关数据类型及接口】

- [ISP_WB_ATTR_S](#)
- [ISP_AWB_CBCR_TRACK_ATTR_S](#)
- [ISP_AWB_LUM_HISTGRAM_ATTR_S](#)

ISP_AWB_CBCR_TRACK_ATTR_S

【说明】

定义 Bayer 域统计信息的联动参数。

【定义】

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

【成员】

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

【注意事项】



- 统计参数联动使能后，AWB 算法会根据环境照度、色温、用户设置的 au16CrMax 数组等实时计算 CrMax、CrMin、CbMax、CbMin 四个参数，并配置相应的逻辑寄存器，此时用户通过 PQ Tools 设置以上四个统计参数不生效。
- 统计参数联动使能，AWB 算法计算的 CrMax 等统计参数在根据 ISO 插值的基础上还和环境色温相关。低色温时，白点范围较宽；中高色温时，白点范围较窄。
- 用户手动配置 CrMax、CrMin、CbMax、CbMin 四个参数时，需要先关闭联动功能。
- au16CrMax[0]、au16CrMin[0]、au16CbMax[0]、au16CbMin[0]的取值可在 AWB 参数标定时确定。用户确定了支持的色温范围后，捕获高低色温的 RAW 图片，计算白色区域的 R/G、B/G 值。au16CrMax[0]和 au16CbMin[0]分别对应低色温的 R/G、B/G；au16CrMin[0]和 au16CbMax[0]分别对应高色温的 R/G、B/G。推荐用户设置的 Cr、Cb 范围稍大于 RAW 图片统计的 R/G、B/G 取值范围。
- AWB 算法计算的 CrMax 等统计参数在根据 ISO 插值的时候，会基于 au16CrMax[0]、au16CrMin[0]、au16CbMax[0]、au16CbMin[0]的值。所以不同 ISO 下的取值设定应以 ISO100 的值为基础，单调递增或递减。
- 建议在低色温(钠灯)环境标定 au16CrMax、au16CbMin 数组。用户统计不同照度下白色区域的 R/G、B/G 值，设置 au16CrMax、au16CbMin 数组。推荐用户设置的 Cr、Cb 范围稍大于 RAW 图片统计的 R/G、B/G 取值范围。
- 因低照度下环境色温多在 5000K 以下，au16CrMin、au16CbMax 两个数组的取值可设为常数。
- 当 AWB gain 配置在 Sensor 处；AWB gain 配置在 WDR 合成前的 DG1 处；AWB 统计信息配置在 DRC 后的时候，在这三种情况下，[ISP_AWB_CBCR_TRACK_ATTR_S](#) 参数不生效，Min 固定为 128，Max 固定为 512。

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

au16CrMax	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMax [0]	1	0x130
au16CrMax [1]	2	0x130
au16CrMax [2]	4	0x130
au16CrMax [3]	8	0x130
au16CrMax [4]	16	0x130



au16CrMax	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMax [5]	32	0x130
au16CrMax [6]	64	0x130
au16CrMax [7]	128	0x14D
au16CrMax [8]	256	0x162
au16CrMax [9]	512	0x177
au16CrMax [10]	1024	0x189
au16CrMax [11]	2048	0x1A1
au16CrMax [12]	4096	0x1A6
au16CrMax [13]	8192	0x1A6
au16CrMax [14]	16384	0x1A6
au16CrMax [15]	32768	0x1A6

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

au16CrMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMin [0]	1	0x40
au16CrMin [1]	2	0x40
au16CrMin [2]	4	0x40
au16CrMin [3]	8	0x40
au16CrMin [4]	16	0x40
au16CrMin [5]	32	0x40
au16CrMin [6]	64	0x40
au16CrMin [7]	128	0x3E
au16CrMin [8]	256	0x3C
au16CrMin [9]	512	0x3A
au16CrMin [10]	1024	0x38
au16CrMin [11]	2048	0x36
au16CrMin [12]	4096	0x35



au16CrMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CrMin [13]	8192	0x34
au16CrMin [14]	16384	0x32
au16CrMin [15]	32768	0x32

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

au16CbMax	Again*Dgain*ISPDgain(倍数)	设置值
au16CbMax [0]	1	0x120
au16CbMax [1]	2	0x120
au16CbMax [2]	4	0x120
au16CbMax [3]	8	0x124
au16CbMax [4]	16	0x128
au16CbMax [5]	32	0x12C
au16CbMax [6]	64	0x130
au16CbMax [7]	128	0x13E
au16CbMax [8]	256	0x149
au16CbMax [9]	512	0x154
au16CbMax [10]	1024	0x15D
au16CbMax [11]	2048	0x169
au16CbMax [12]	4096	0x16C
au16CbMax [13]	8192	0x16C
au16CbMax [14]	16384	0x16C
au16CbMax [15]	32768	0x16C

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

au16CbMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CbMin [0]	1	0x40



au16CbMin	Again*Dgain*ISPDgain(倍数)	设置值
au16CbMin [1]	2	0x40
au16CbMin [2]	4	0x40
au16CbMin [3]	8	0x40
au16CbMin [4]	16	0x40
au16CbMin [5]	32	0x40
au16CbMin [6]	64	0x40
au16CbMin [7]	128	0x3C
au16CbMin [8]	256	0x38
au16CbMin [9]	512	0x34
au16CbMin [10]	1024	0x30
au16CbMin [11]	2048	0x2C
au16CbMin [12]	4096	0x2A
au16CbMin [13]	8192	0x28
au16CbMin [14]	16384	0x28
au16CbMin [15]	32768	0x28

【相关数据类型及接口】[ISP_AWB_ATTR_S](#)**ISP_AWB_LUM_HISTGRAM_ATTR_S****【说明】**

定义白平衡的亮度直方图统计参数。

【定义】

```
typedef struct hiISP_AWB_LUM_HISTGRAM_ATTR_S
{
    HI_BOOL bEnable;
    ISP\_OP\_TYPE\_E enOpType;
    HI_U8   au8HistThresh[AWB_LUM_HIST_NUM];
    HI_U16  au16HistWt[AWB_LUM_HIST_NUM];
}
```



```
} ISP_AWB_LUM_HISTGRAM_ATTR_S;
```



说明

宏定义如下：

```
#define AWB_LUM_HIST_NUM (6)
```

【成员】

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

【注意事项】

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

【相关数据类型及接口】

[ISP_AWB_ATTR_S](#)

ISP_AWB_ALG_TYPE_E

【说明】

定义白平衡的算法类型属性。

【定义】

```
typedef enum hiISP_AWB_ALG_TYPE_E
```



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

【成员】

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

【注意事项】

- AWB_ALG_LOWCOST CPU 占用低：AWB 不同照度的稳定性较好，对光源的适应性稍好。
- AWB_ALG_ADVANCE CPU 占用高：高低色温的极低照度情况下，保留轻微光源色；提高了大面积纯色场景（室外大面积绿色、室内大面积肤色等）AWB 算法准确度。

【相关数据类型及接口】

[ISP_AWB_ATTR_S](#)

ISP_AWB_CT_LIMIT_ATTR_S

【说明】

定义白平衡的增益范围限制属性。

【定义】

```
typedef struct hiISP_AWB_CT_LIMIT_ATTR_S  
{  
    HI_BOOL bEnable;  
    ISP\_OP\_TYPE\_E enOpType;  
    HI_U16 u16HighRgLimit;  
    HI_U16 u16HighBgLimit;
```



```
HI_U16 u16LowRgLimit;  
HI_U16 u16LowBgLimit;  
} ISP_AWB_CT_LIMIT_ATTR_S;
```

【成员】

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

【注意事项】

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

【相关数据类型及接口】



ISP_AWB_ATTR_S

ISP_MWB_ATTR_S

【说明】

定义 ISP 手动白平衡属性。

【定义】

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

【成员】

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

【注意事项】

无

【相关数据类型及接口】

ISP_WB_ATTR_S



ISP_WB_ATTR_S

【说明】

定义白平衡属性。

【定义】

```
typedef struct hiISP_WB_ATTR_S
{
    HI_BOOL bByPass;
    HI_U8    u8AWBRunInterval;
    ISP_OP_TYPE_E enOpType;
    ISP_MWB_ATTR_S stManual;
    ISP_AWB_ATTR_S stAuto;
    ISP_AWB_ALG_E enAlgType;
} ISP_WB_ATTR_S;
```

【成员】

成员名称	描述
bByPass	白平衡模块 Bypass 使能，默认值 HI_FALSE。
u8AWBRunInterval	白平衡模块工作频率。取值范围：[0x1, 0xFF]
enOpType	自动白平衡和手动白平衡切换。
stManual	手动白平衡参数。
stAuto	自动白平衡参数。
enAlgType	算法类型选择。

【注意事项】

- bByPass 为 HI_TRUE 时，WB 的其它参数设置不生效，RGB 通道增益系数固定为 0x100。
- u8AWBRunInterval 默认为 4，表示每 4 帧运行一次 AWB。客户可根据业务需求降低 AWB 的运行频率。
- enAlgType 默认为 ALG_AWB 代表使用原有的 AWB 算法，用户可以使用 ALG_AWB_SPEC 切换为机器学习 AWB 算法。



【相关数据类型及接口】

- [ISP_MWB_ATTR_S](#)
- [ISP_AWB_ATTR_S](#)

ISP_AWB_MULTI_LS_TYPE_E

【说明】

定义混合光源下的 AWB 策略。

【定义】

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

【成员】

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

【注意事项】

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

【相关数据类型及接口】

[ISP_AWB_ATTR_S](#)

ISP_AWB_INDOOR_OUTDOOR_STATUS_E

【说明】

定义 AWB 室内外状态。



【定义】

```
typedef enum hiISP_AWB_INDOOR_OUTDOOR_STATUS_E
{
    AWB_INDOOR_MODE = 0,
    AWB_OUTDOOR_MODE = 1,
    AWB_INDOOR_OUTDOOR_BUTT
}ISP_AWB_INDOOR_OUTDOOR_STATUS_E;
```

【成员】

成员名称	描述
AWB_INDOOR_MODE	室内模式。
AWB_OUTDOOR_MODE	室外模式。
AWB_INDOOR_OUTDOOR_BUTT	无效。

【注意事项】

无

【相关数据类型及接口】

[ISP_AWB_IN_OUT_ATTR_S](#)

ISP_AWB_ATTR_EX_S

【说明】

定义自动白平衡扩展属性。

【定义】

```
typedef struct hiISP_AWB_ATTR_EX_S
{
    HI_U8 u8Tolerance;
    HI_U8 u8ZoneRadius;
    HI_U16 u16CurveLLimit;
    HI_U16 u16CurveRLimit;
    HI_BOOL bExtraLightEn;
    ISP\_AWB\_EXTRA\_LIGHTSOURCE\_INFO\_S stLightInfo[AWB_LS_NUM];
    ISP\_AWB\_IN\_OUT\_ATTR\_S stInOrOut;
```



```
HI_BOOL bMultiLightSourceEn;  
ISP_AWB_MULTI_LS_TYPE_E enMultiLSType;  
HI_U16 u16MultiLSScaler;  
HI_U16 au16MultiCTBin[AWB_MULTI_CT_NUM];  
HI_U16 au16MultiCTWt[AWB_MULTI_CT_NUM];  
HI_BOOL bFineTunEn;  
HI_U8 u8FineTunStrength;  
} ISP_AWB_ATTR_EX_S;
```



说明

宏定义如下：

```
#define AWB_LS_NUM          (4)  
#define AWB_MULTI_CT_NUM   (8)
```

【成员】

成员名称	描述
u8Tolerance	自动白平衡调整的偏差范围，检测误差在门限范围内时，AWB 不动作。
u8ZoneRadius	自动白平衡统计中对像素分类时用的距离范围。该值越小，AWB 精度越高，但会降低 AWB 算法稳定性。
u16CurveLLimit	自动白平衡色温曲线的左边界限。取值范围：[0x0, 0x100]
u16CurveRLimit	自动白平衡色温曲线的右边界限。取值范围：[0x100, 0xFFFF]
bExtraLightEn	自动白平衡计算时是否考虑色温曲线外的独立光源点。
stLightInfo[AWB_LS_NUM]	色温曲线外的独立光源点的信息，最多可以添加 4 个。
stInOrOut	自动白平衡对场景做室内外判断的参数。
bMultiLightSourceEn	自动白平衡检测当前场景是否为混合光源，根据混合光源程度调整饱和度或 CCM。



成员名称	描述
enMultiLSType	混合光源调整策略选择，支持调整饱和度或 CCM。
u16MultiLSScaler	混合光源下，饱和度或 CCM 最大调整幅度。实际调整幅度还和场景混合光源程度有关。 取值范围：[0x0, 0x100]
au16MultiCTBin[AWB_MULTI_CT_NUM]	混合光源下的色温分段参数。 取值范围：要求为单调递增序列。
au16MultiCTWt[AWB_MULTI_CT_NUM]	混合光源下的色温权重参数。 取值范围：[0x0, 0x400]
bFineTunEn	自动白平衡特殊色检测开关，包括肤色检测等。
u8FineTunStrength	肤色、蓝色等单色检测的强度。仅在 bFineTunEn 使能时有效。 取值范围：[0x0, 0xFF]

图4-3 色温曲线的参数示意

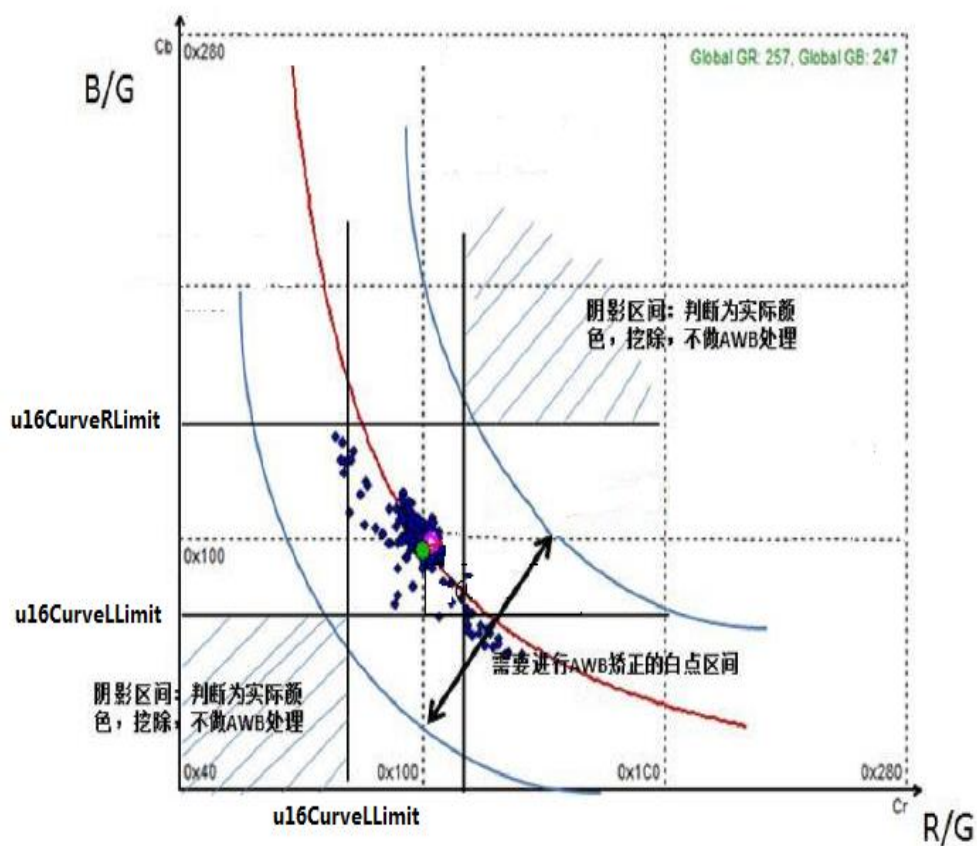
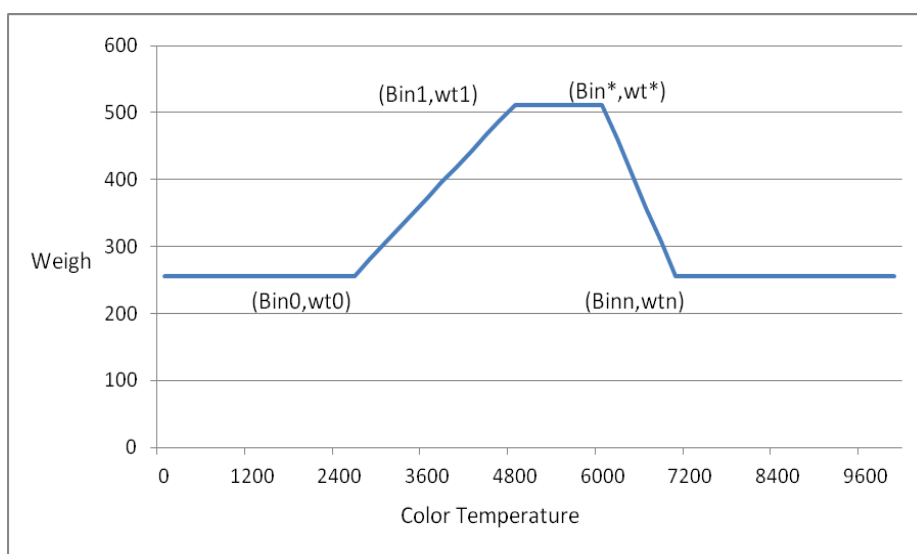


图4-4 混合光源场景色温权重设置示例说明 (n 为色温分段点个数)



【注意事项】

- u8Tolerance 是 AWB 的灵敏度参数。该值为 0 时，AWB 系数实时更新，CPU 占用较高；该值较大时，AWB 在检测到环境色温变化时再更新 AWB 系数，CPU 占用较低，但在环境色温微调时，AWB 可能出现轻微的偏色。推荐室外应用时 u8Tolerance 置为 0，室内应用时置为 2。
- u8ZoneRadius 是选择光源环境白色块的色差半径。u8ZoneRadius 越大，白色块的限制条件越宽，找到的白色块较多，稳定性稍好，精度稍低；u8ZoneRadius 越小，白色块的限制条件越严格，找到的白色块较少，AWB 精度更高。在混合光源的应用场景，推荐 u8ZoneRadius 比普通场景稍大。
- u16CurveLLimit 取值 $\leq 0x100$ ，u16CurveRLimit 取值 $\geq 0x100$ 。u16CurveLLimit 和 u16CurveRLimit 两个参数用来排除颜色块。如图 4-3 所示，u16CurveLLimit 排除了左侧的阴影区域，u16CurveRLimit 排除了右侧的阴影区域。
- AWB 算法支持在色温曲线外，再增加独立光源点信息，提升特殊光源下的 AWB 表现。
- bMultiLightSourceEn 使能后，AWB 算法判断场景是否混合光源，在混合光源场景，自动降低饱和度或调整 CCM，以减弱偏色程度。最大调整幅度由 u16MultiLSScaler 决定。调整 CCM 方式会改变色调。
- enMultiLSType= AWB_MULTI_LS_SAT 时，调整饱和度来改善混合光源场景的偏色。u16MultiLSScaler/0x100 表示饱和度增益系数的下限。
u16MultiLSScaler=0xC0，表示饱和度的增益系数下限是 $0xC0/0x100 = 0.75$ ，图像饱和度的波动范围是 $[0.75, 1.0]$ （1.0 表示关闭 bMultiLightSourceEn 功能的图像饱和度）；u16MultiLSScaler=0x0，表示饱和度的增益系数下限是 $0x0/0x100 = 0$ ，图像饱和度的波动范围是 $[0, 1]$ ，在混合光源下图像可能变黑白。
u16MultiLSScaler 取值越小，图像饱和度的波动范围越大，颜色稳定性下降。
- enMultiLSType= AWB_MULTI_LS_CCM 时，调整 CCM 来改善混合光源场景的偏色。u16MultiLSScaler/0x100 表示 CCM 的调整幅度，取值越大，图像饱和度越低。u16MultiLSScaler=0xC0，表示 CCM 的最大调整幅度为 0xC0；
u16MultiLSScaler=0x0，表示 CCM 的最大调整幅度为 0，等效于 bMultiLightSourceEn=0。
- 上海海思 AWB 算法在 WDR 模式自动关闭混合光源判断功能。
- au16MultiCTBin、au16MultiCTWt 两组参数的合理配置，可以改善混合光源高色温或低色温区域颜色表现。在非混合光源场景，参数配置不生效。图 4-4 说明色温和权重配置的映射关系。



- 如果混合光源场景希望使能色温权重来改善 AWB 表现，但不做饱和度或 CCM 调解，可配置如下：bMultiLightSourceEn=HI_TRUE，enMultiLSType=AWB_MULTI_LS_SAT，u16MultiLSScaler=0x100。bFineTunEn 打开，AWB 会自动检测肤色等特殊色，改善肤色场景 AWB 表现，提高 AWB 精度。但可能在蓝色背景+3500K 光源时发生误判，导致图像轻微偏黄。
- bFineTunEn 打开，AWB 会自动检测肤色等特殊色，改善肤色场景 AWB 表现，提高 AWB 精度。但可能在蓝色背景+3500K 光源时发生误判，导致图像轻微偏黄。
- u8FineTunStrength 调整肤色检测的强度，取值越大，肤色场景 AWB 表现越好，但肤色误判时的副作用越明显。推荐采用默认值 0x80。

【相关数据类型及接口】

- [ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S](#)
- [ISP_AWB_IN_OUT_ATTR_S](#)

ISP_AWB_EXTRA_LIGHTSOURCE_INFO_S

【说明】

定义独立光源点的信息。

【定义】

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

【成员】

成员名称	描述
u16WhiteRgain	增加或删除区域的中心 Cr 坐标，8bit 小数精度。 取值范围：[0x0, 0xFFFF]
u16WhiteBgain	增加或删除区域的中心 Cb 坐标，8bit 小数精度。



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

【注意事项】

- 独立光源点实现以下功能：用户设定 CbCr 坐标的某个区域，指定该区域为特殊光源下的白色表现，增加为独立光源点；或指定该区域为特定光源下的干扰色，进行删除。
- 通过 u8LightStatus 参数的设置实现功能选择。u8LightStatus 设置为 1 时，在用户校正的 Plankcian 曲线外，增加特殊光源点，以优化该光源下颜色表现，不影响设备在其它光源下的表现；u8LightStatus 设置为 2 时，支持用户删除特定光源的某个干扰色(肤色、绿色、蓝天等)。
- 建议删除干扰色仅用来优化特定环境的颜色表现，否则会影响其它光源的表现。在 CbCr 坐标，5500K 光源下的肤色与 3500K 光源的白色重合度很高，在 5500K 环境删除肤色的干扰，会导致 3500K 光源下偏色。
- u16WhiteRgain、u16WhiteBgain 的取值。在该光源下捕获 ColorChecker Raw 数据，通过 Calibration Tool Staitic WB 选项校准得到的 Rgain、Bgain 即 u16WhiteRgain、u16WhiteBgain 坐标值。删除干扰色，可选择干扰色区域进行静态白平衡校正，来确定 u16WhiteRgain、u16WhiteBgain 坐标值。
- u8 Radius 取值由用户设置，增加光源点，建议取值在[0x8, 0x10]范围内，删除干扰色，尽量避免[u16WhiteRgain、u16WhiteBgain]为中心，u8 Radius 为半径的圆形区域与 Plankcian 曲线重合。增加光源点或删除干扰色，设置 u8 Radius 半径太大时(覆盖场景内大多数统计块)，会导致功能失效，甚至起反作用。



- 当前支持定义最多 4 个光源色，4 个相互独立。4 个光源点需要避免交集，特别需要避免以下情况：CbCr 坐标的单个点，既指定为增加的独立光源点，又指定为删除的干扰色。

【相关数据类型及接口】

[ISP_AWB_ATTR_EX_S](#)

ISP_AWB_IN_OUT_ATTR_S

【说明】

定义对场景做室内外判断的参数。

【定义】

```
typedef struct hiISP_AWB_IN_OUT_ATTR_S
{
    HI_BOOL bEnable;
    ISP\_OP\_TYPE\_E enOpType;
    ISP\_AWB\_INDOOR\_OUTDOOR\_STATUS\_E enOutdoorStatus;
    HI_U32 u32OutThresh;
    HI_U16 u16LowStart;
    HI_U16 u16LowStop;
    HI_U16 u16HighStart;
    HI_U16 u16HighStop;
    HI_BOOL bGreenEnhanceEn;
    HI_U8 u8OutShiftLimit;
} ISP_AWB_IN_OUT_ATTR_S;
```

【成员】

成员名称	描述
bEnable	对场景做室内外判断开关。
enOpType	对场景做室内外判断类型(自动或手动)。
enOutdoorStatus	手动模式下，用户输入室内外状态信息。
u32OutThresh	室内外场景判定设置的阈值，以单位为 us 的曝光时间，小于该阈值则认为是室外。
u16LowStart	将位于低色温范围内的像素权重调低，低色温区起始值，建议



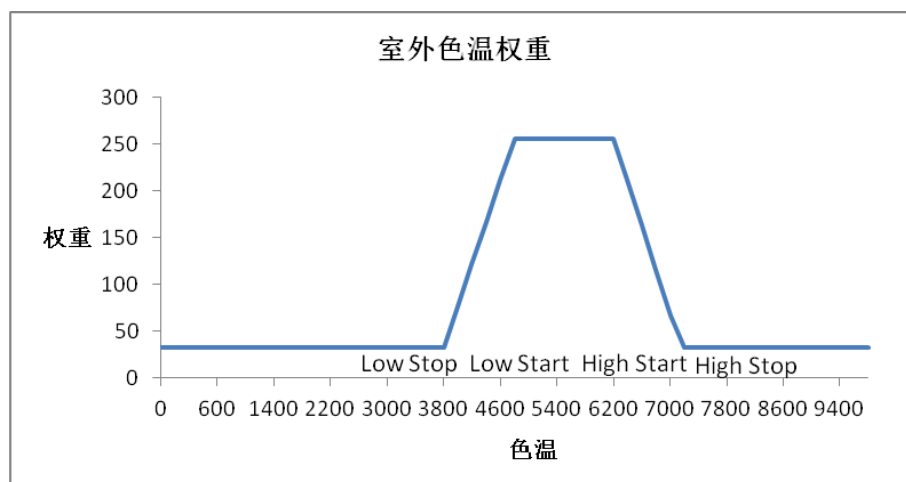
成员名称	描述
	5000K。
u16LowStop	将位于低色温范围内的像素权重调低，低色温区终止值，建议4500K。 取值范围：(0, u16LowStart)
u16HighStart	将位于高色温范围内的像素权重调低，高色温区起始值，建议6500K。 取值范围：(u16LowStart, 0xFFFF]
u16HighStop	将位于高色温范围内的像素权重调低，高色温区终止值，建议8000K。 取值范围：(u16HighStart, 0xFFFF]
bGreenEnhanceEn	在绿色植物场景下是否对绿色通道增强。
u8OutShiftLimit	判断为室外场景时，AWB 算法的白点范围限制 Shift=u8OutShiftLimit。 取值范围：[0x0, 0xFF]

【注意事项】

- u32OutThresh 需要根据 sensor 的感光特性进行调整。等效曝光时间=曝光时间*模拟增益*数字增益*ISP 数字增益。
- 用户调用自己的 AE 库，且室内外判断类型为自动时，请务必将当前曝光时间和 ISO 传递给 AWB 库。
- 设置色温范围的 u16LowStart、u16LowStop、u16HighStart、u16HighStop 等 4 个参数，和 sensor 校正相关，建议用户针对 sensor 做微调。四个参数的取值范围要求：u16LowStop < u16LowStart < u16HighStart < u16HighStop。如果设置的室外色温范围[u16LowStart, u16HighStart]较宽，室外的大面积纯色场景颜色表现可能会稍差；如果设置的室外色温范围[u16LowStart, u16HighStart]较窄，在日落前等高低色温场景，AWB 不能完全恢复，低色温下稍微偏黄，高色温下稍微偏蓝。
- AWB 校正参数调整后，u16LowStart、u16LowStop、u16HighStart、u16HighStop 等 4 个参数要再次进行调整，以达到最佳效果。

- Shift 取值影响 AWB 算法表现。普通场景 u8ShiftLimit、u8OutShiftLimit 两个参数的取值大小对 AWB 结果影响不大。特殊光源、大面积单色场景，u8ShiftLimit、u8OutShiftLimit 两个参数的取值大小对 AWB 结果影响较大。两个 Shift 参数取值较小时，单色表现好，但特殊光源下容易偏色；两个 Shift 参数取值较大时，AWB 表现稳定性好，但大面积绿色、黄色等场景可能偏色。

图4-5 室外色温范围参数的意义



【相关数据类型及接口】

[ISP_AWB_ATTR_EX_S](#)

ISP_WB_INFO_S

【说明】

定义白平衡，饱和度，颜色校正信息。

【定义】

```
typedef struct hiISP_WB_INFO_S
{
    HI_U16 u16Rgain;
    HI_U16 u16Grgain;
    HI_U16 u16Gbgain;
    HI_U16 u16Bgain;
    HI_U16 u16Saturation;
    HI_U16 u16ColorTemp;
    HI_U16 au16CCM[CCM_MATRIX_SIZE];
    HI_U16 u16LSOCT;
```



```

HI_U16 u16LS1CT;
HI_U16 u16LS0Area;
HI_U16 u16LS1Area;
HI_U8  u8MultiDegree;
HI_U32 u32FirstStableTime;
HI_U16 u16ActiveShift;
ISP_AWB_INDOOR_OUTDOOR_STATUS_E enInOutStatus;
HI_S16  s16Bv;
} ISP_WB_INFO_S;

```

【成员】

成员名称	描述
u16Rgain	当前 R 通道增益值，8bit 小数精度。
u16Grgain	当前 Gr 通道增益值，8bit 小数精度。
u16Gbgain	当前 Gb 通道增益值，8bit 小数精度。
u16Bgain	当前 B 通道增益值，8bit 小数精度。
u16Saturation	当前饱和度值，有效范围为[0, 255]。
u16ColorTemp	当前色温值。
au16CCM[CCM_MATRIX_SIZE]	当前颜色校正矩阵值，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。
u16LS0CT	混合光源场景，主光源色温。
u16LS1CT	混合光源场景，次要光源色温。
u16LS0Area	混合光源场景，主光源面积。取值范围为[0x0, 0xFF]。
u16LS1Area	混合光源场景，次要光源面积。取值范围为[0x0, 0xFF]。
u8MultiDegree	当前场景是混合光源的概率。
u32FirstStableTime	首次 AWB 收敛稳定的时间，以帧为计数单位。
u16ActiveShift	当前场景最终生效的白点范围限制 shift 值。取值



成员名称	描述
	范围为[0x0,0xFF]
enInOutStatus	室内外检测结果。
s16Bv	当前环境 bv 值

【注意事项】

- u8MultiDegree 非 0 时，表示场景是混合光源。该值越大，场景为混合光源的概率越大。
- u16LS0CT、u16LS1CT、u16LS0Area、u16LS1Area 等查询结果仅在 u8MultiDegree 非 0 时有效。
- 混合光源检测仅在室内场景打开，AWB 判断为室外场景后，逐渐减弱混合光源检测的强度，以避免饱和度降低。
- 混合光源概率说明：主光源和次要光源的亮度、色温差等条件影响混合光源的概率计算。检测到场景内存在主光源和次要光源两种光源后，主光源和次要光源的色温差异越大、亮度差异越小，混合光源概率越大。
- u16LS0Area 和 u16LS1Area 之和有可能大于白平衡分区个数，原因是部分区域可能受到两种光源的影响，算法不将重叠区域严格识别，会叠加到主光源、次要光源。
- s16Bv 为上海海思 AWB 库计算出的环境亮度值，该亮度值与用户在 [ISP_SPEC_AWB_ATTR_S](#) 中输入的 u16Fno 相关。用户可以参考实际场景输出的 s16Bv 调整色温转换表([ISP_SPEC_AWB_CAA_TBL_S](#))内 Bv 数值来调整色温转换表覆盖亮度的区间。
- u32FirstStableTime 是 AWB 首次收敛稳定后所对应的帧数，非 0 表示 AWB 已稳定。ISP 系统启动后，1-128 帧 AWB 固定每帧执行一次，以加速 AWB 收敛。

【相关数据类型及接口】

无



5 CCM

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

5.1 概述

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

5.2 重要概念

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



5.3 功能描述

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

图5-1 CCM 矩阵

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

5.4 API 参考

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

- [HI_MPI_ISP_SetSaturationAttr](#)：设置颜色饱和度属性。
- [HI_MPI_ISP_GetSaturationAttr](#)：获取颜色饱和度属性。
- [HI_MPI_ISP_SetCCMAAttr](#)：设置颜色校正基础矩阵。
- [HI_MPI_ISP_GetCCMAAttr](#)：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(VI_PIPE ViPipe, const ISP_SATURATION_ATTR_S  
*pstSatAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入



pstSatAttr	颜色饱和度属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_GetSaturationAttr](#)

HI_MPI_ISP_GetSaturationAttr

【描述】

获取颜色饱和度属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetSaturationAttr(VI_PIPE ViPipe, ISP\_SATURATION\_ATTR\_S *pstSatAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstSatAttr	颜色饱和度属性。	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetSaturationAttr](#)

HI_MPI_ISP_SetCCMAAttr

【描述】

设置颜色矩阵。

【语法】

```
HI_S32 HI_MPI_ISP_SetCCMAAttr(VI_PIPE ViPipe, const ISP\_COLORMATRIX\_ATTR\_S *pstCCMAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstCCMAAttr	颜色矩阵。	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。
- 可根据当前色温，设置不同的 CCM，从而在高低色温下都达到较好的颜色还原。
- 该 MPI 支持最大七个最小三个不同色温下的色彩还原矩阵，要按照色温递减的方式配置 CCM，即 0-6 组 CCM 矩阵由高色温到低色温。典型的三组 CCM 为 D50, TL84, A 三个光源下的 CCM。典型的五组 CCM 为 10K, D65, D50, TL84, A 五个光源下的 CCM。前一组的色温值与后一组的色温值需符合如下规则： $T_{pre} * (100 - 6.25) > T_{post} * (100 + 6.25)$ 。
- 手动模式下，ISP 系统生效的 CCM 即用户手动配置的 CCM，有利于 CCM 精调，因此建议在手动模式下校正不同色温光源下的 CCM 矩阵，得到多组 CCM 矩阵后再通过 MPI 写入，验证多个光源下的 CCM 组和效果。手动 CCM 模式下，饱和度调整有效。在特定光源下做 CCM 精调时，需要先确认饱和度是否期望值。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetCCMAttr](#)

HI_MPI_ISP_GetCCMAttr

【描述】

获取颜色矩阵。

【语法】



```
HI_S32 HI_MPI_ISP_GetCCMAAttr(VI_PIPE ViPipe, ISP\_COLORMATRIX\_ATTR\_S *pstCCMAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi PIPE 号	输入
pstCCMAAttr	颜色矩阵。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h
- 库文件：libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetCCMAAttr](#)

5.5 数据类型

- [ISP_SATURATION_ATTR_S](#)：定义 ISP 颜色饱和度属性。
- [ISP_SATURATION_MANUAL_S](#)：定义手动饱和度属性。
- [ISP_SATURATION_AUTO_S](#)：定义自动饱和度属性。
- [ISP_COLORMATRIX_ATTR_S](#)：定义 ISP 颜色矩阵属性。



- [ISP_COLORMATRIX_AUTO_S](#): 定义自动颜色校正矩阵属性。
- [ISP_COLORMATRIX_MANUAL_S](#): 定义手动颜色校正矩阵属性。
- [ISP_COLORMATRIX_PARAM_S](#): 定义多组颜色校正矩阵系数

ISP_SATURATION_ATTR_S

【说明】

定义 ISP 颜色饱和度属性。

【定义】

```
typedef struct hiISP_SATURATION_ATTR_S
{
    ISP\_OP\_TYPE\_E enOpType;
    ISP\_SATURATION\_MANUAL\_S stManual;
    ISP\_SATURATION\_AUTO\_S stAuto;
}ISP_SATURATION_ATTR_S;
```

【成员】

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

【注意事项】

- 设置饱和度功能分为自动和手动：
 - enOpType 为 OP_TYPE_AUTO，使用自动饱和度调节功能。
此时饱和度值会根据系统增益自动调节，饱和度与系统增益的关系请参见成员变量 au8Sat [16]的描述。
在自动饱和度模式下，系统最终的饱和度由系统增益对应的 au8Sat [16]变量，AWB 控制的混合光源下自动调节饱和度 AWB_MULTI_LS_SAT，极低照度下 CCM Bypass 功能 ISOAct，高、低色温下 CCM Bypass 功能 TempAct，几个因素共同决定。
 - enOpType 为 OP_TYPE_MANUAL，使用手动饱和度调节功能。



此时变量 u8Sat[16]无效。

在手动饱和度模式下，系统最终的饱和度由用户完全控制。

【相关数据类型及接口】

- [ISP_SATURATION_AUTO_S](#)
- [ISP_COLORMATRIX_MANUAL_S](#)

ISP_SATURATION_MANUAL_S

【说明】

定义手动饱和度属性。

【定义】

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

【成员】

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

【注意事项】

无

【相关数据类型及接口】

[ISP_SATURATION_AUTO_S](#)

ISP_SATURATION_AUTO_S

【说明】

定义自动饱和度属性。

【定义】

```
typedef struct hiISP_SATURATION_AUTO_S
```



```
{  
HI_U8 au8Sat[ISP_AUTO_ISO_STRENGTH_NUM];  
} ISP_SATURATION_AUTO_S;
```

【成员】

成员名称	描述
au8Sat[ISP_AUTO_ISO_STRENGTH_NUM]	自动饱和度值。 推荐配置为递减序列。

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

au8Sat	Again*Dgain*ISPDgain(倍数)	推荐设置值
au8Sat [0]	1	0x74
au8Sat [1]	2	0x74
au8Sat [2]	4	0x70
au8Sat [3]	8	0x6b
au8Sat [4]	16	0x64
au8Sat [5]	32	0x5b
au8Sat [6]	64	0x52
au8Sat [7]	128	0x44
au8Sat [8]	256	0x3b
au8Sat [9]	512	0x38
au8Sat [10]	1024	0x38
au8Sat [11]	2048	0x38
au8Sat [12]	4096	0x38
au8Sat [13]	8192	0x38
au8Sat [14]	16384	0x38
au8Sat [15]	32768	0x38

【注意事项】



无

【相关数据类型及接口】

[ISP_SATURATION_MANUAL_S](#)

ISP_COLORMATRIX_ATTR_S

【说明】

定义 ISP 颜色矩阵属性。

【定义】

```
typedef struct hiISP_COLORMATRIX_ATTR_S
{
    ISP\_OP\_TYPE\_E enOpType;
    ISP\_COLORMATRIX\_MANUAL\_S stManual;
    ISP\_COLORMATRIX\_AUTO\_S stAuto;
} ISP_COLORMATRIX_ATTR_S;
```

【成员】

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

【注意事项】

无

【相关数据类型及接口】

- [ISP_COLORMATRIX_AUTO_S](#)
- [ISP_COLORMATRIX_MANUAL_S](#)

ISP_COLORMATRIX_AUTO_S

【说明】

定义自动颜色校正矩阵属性。



【定义】

```
typedef struct hiISP_COLORMATRIX_AUTO_S
{
    HI_BOOL bISOActEn;
    HI_BOOL bTempActEn;
    HI_U16 u16CCMTabNum;
    ISP_COLORMATRIX_PARAM_S astCCMTab[CCM_MATRIX_NUM];
} ISP_COLORMATRIX_AUTO_S;
```

【成员】

成员名称	描述
bISOActEn	是否使能低照度下 CCM Bypass 功能。
bTempActEn	是否使能高、低色温下 CCM Bypass 功能。
u16CCMTabNum	当前配置的 CCM 的组数。 取值范围：[3,7]
astCCMTab[CCM_MATRIX_NUM]	不同色温下的颜色校正矩阵和对应的色温值。

【注意事项】

- 当 [ISP_SATURATION_ATTR_S](#) 中的 enOpType 为 OP_TYPE_AUTO，使用自动饱和度和调节功能时，ISOAct 和 TempAct 功能生效。当 enOpType 为 OP_TYPE_MANUAL，使用手动饱和度和调节功能时，ISOAct 和 TempAct 功能不生效。
- ISOAct 功能指的是极低照度下（ISO 大于 12800）逐渐 Bypass CCM，将由饱和度数组 au8Sat [16]控制得到的 CCM 矩阵逐渐调向接近单位阵，这时 CCM 的实际饱和度会和饱和度数组中设定的不一致。用户关闭该功能后，极低照度下，饱和度数组仍生效。
- TempAct 功能指的是在高色温(大于 8000K)和低色温场景(小于 2500K)，AWB 可能未完全恢复，此时采用高饱和度 CCM 矩阵，会加剧 AWB 的偏色程度。比如钠灯场景，AWB 校正后白色区域蓝色稍有不足，采用高饱和度 CCM 矩阵，白色区域的蓝色分量可能为 0，图像整体偏黄。使能 bTempActEn，在高低色温下逐渐 BypassCCM，将由饱和度数组 au8Sat [16]控制得到的 CCM 矩阵逐渐调向接近单位阵，优化类似场景颜色表现。



- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。

【相关数据类型及接口】

- [ISP_COLORMATRIX_MANUAL_S](#)
- [ISP_COLORMATRIX_PARAM_S](#)

ISP_COLORMATRIX_PARAM_S

【说明】

定义不同色温下自动颜色校正矩阵系数。

【定义】

```
typedef struct hiISP_COLORMATRIX_PARAM_S
{
    HI_U16 u16ColorTemp;
    HI_U16 au16CCM[CCM_MATRIX_SIZE];
} ISP_COLORMATRIX_PARAM_S;
```



说明

宏定义如下：

```
#define CCM_MATRIX_SIZE      (9)
```

【成员】

成员名称	描述
u16ColorTemp	当前配置的 CCM 对应的色温。 取值范围：[500, 30000]
au16CCM[CCM_MATRIX_SIZE]	不同色温下的颜色校正矩阵，8bit 小数精度。bit 15 是符号位，0 表示正数，1 表示负数，例如 0x8010 表示-16。bit12,13,14 是无效的，即正数从 0x0000 到 0x0FFF，负数从 0x8000 到 0x8FFF，是 CCM 的合理参数。 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。



- 当饱和度为自动的时候，实际生效的 CCM 参数和 AWB 算法反馈的当前色温值，自动饱和度参数有关。当饱和度为手动的时候，实际生效的 CCM 参数和 AWB 算法反馈的当前色温值有关。
- Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300/Hi3562V100/Hi3566V100 在相对 24 色卡标准值饱和度 170%以上的 CCM 参数下，有可能会发生图像中饱和度高的颜色变暗的现象，建议使用正常饱和度范围内的 CCM 参数。

【相关数据类型及接口】

[ISP_COLORMATRIX_AUTO_S](#)

ISP_COLORMATRIX_MANUAL_S

【说明】

定义手动颜色校正矩阵属性。

【定义】

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

【成员】

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

【注意事项】

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



- bSatEn = 0, 生效的 CCM = Manual CCM。
- bSatEn = 1, 生效的 CCM 会自动修正, 保持每行参数的和为 0x100。bSatEn = 0, 生效的 CCM 为手动设定参数值。

【相关数据类型及接口】

[ISP_COLORMATRIX_AUTO_S](#)



6 IMP

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

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

6.1 Sharpen

6.1.1 功能描述

整个 ISP 系统包含两个 sharpen，前 Sharpen（ISP Sharpen）和后 Sharpen（VPSS Sharpen），其中：

- 只有 Hi3559AV100 和 Hi3519AV100 有后 Sharpen。
- 前 Sharpen 和后 Sharpen 的 MPI 接口都是一致的，但前 Sharpen 多了 ShootSupAdj、RGain、BGain、SkinGain、EdgeFiltMaxCap 和 MaxSharpGain 这 6 个接口，其它的所有接口都一致。

Sharpen 模块用于增强图像的清晰度，包括调节图像边缘的锐化属性和增强图像的细节和纹理的清晰度，同时还能分别独立控制图像的带方向的边缘和无方向的细节纹理的锐化强度。此外，还能控制锐化后的图像的 overshoot（白边白点）和 undershoot（黑边黑点），以及抑制噪声的增强。



6.1.2 API 参考

- [HI_MPI_ISP_SetIspSharpenAttr](#): 设置图像锐化属性。
- [HI_MPI_ISP_GetIspSharpenAttr](#): 获取图像锐化属性。

HI_MPI_ISP_SetIspSharpenAttr

【描述】

设定图像锐化属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetIspSharpenAttr(VI_PIPE ViPipe, const ISP\_SHARPEN\_ATTR\_S
*pstIspShpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspShpAttr	图像锐化属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_GetIspSharpenAttr](#)

HI_MPI_ISP_GetIspSharpenAttr

【描述】

获取图像锐化属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetIspSharpenAttr(VI_PIPE ViPipe, ISP\_SHARPEN\_ATTR\_S *pstIspShpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspShpAttr	图像锐化属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 无 3DNR 后 Sharpen。

【举例】

无



【相关主题】

[HI_MPI_ISP_SetIspSharpenAttr](#)

6.1.3 数据类型

- [ISP_SHARPEN_MANUAL_ATTR_S](#): 定义 ISP Sharpen 手动属性。
- [ISP_SHARPEN_AUTO_ATTR_S](#): 定义 ISP Sharpen 自动属性。
- [ISP_SHARPEN_ATTR_S](#): 定义 ISP Sharpen 属性。

ISP_SHARPEN_MANUAL_ATTR_S

【说明】

定义 ISP Sharpen 手动属性。

【定义】

```
typedef struct hiISP_SHARPEN_MANUAL_ATTR_S
{
    HI_U8 au8LumaWgt[ISP_SHARPEN_LUMA_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM];
        HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM];
        HI_U16 u16TextureFreq;
        HI_U16 u16EdgeFreq;
        HI_U8  u8OverShoot;
        HI_U8  u8UnderShoot;
        HI_U8 u8ShootSupStr;
    HI_U8  u8ShootSupAdj;
        HI_U8  u8DetailCtrl;
        HI_U8  u8DetailCtrlThr;
    HI_U8  u8EdgeFiltStr;
    HI_U8  u8EdgeFiltMaxCap;
        HI_U8  u8RGain;
    HI_U8  u8GGain;
    HI_U8  u8BGain;
    HI_U8  u8SkinGain;
    HI_U16 u16MaxSharpGain;
    HI_U8  u8WeakDetailGain;
} ISP_SHARPEN_MANUAL_ATTR_S;
```




说明

宏定义如下：

```
#define ISP_SHARPEN_GAIN_NUM      (32)
```

```
#define ISP_SHARPEN_LUMA_NUM      (32)
```

【成员】

成员名称	描述
au8LumaWgt	亮度锐化权重。满量程 0-255 的亮度被 32 个等分点平均分为 32 段亮度区间，每一段亮度区间对应一个亮度权重。比如 0-7 的亮度区间的权重是 au8LumaWgt[0]，8-15 的亮度区间的权重是 au8LumaWgt[1]，依次类推。如图 6-1 所示。值越大，图像锐化程度越高，反之，越弱。 Hi3516CV500/Hi3559V200 取值范围：[0, 127]。建议值 127。 Hi3516EV200/Hi3516EV300 取值范围：[0, 31]。建议值 31。 3DNR 后 Sharpen 中该参数与 ISO 不联动。
au16TextureStr	无方向的细节纹理的锐化强度，设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。该参数是一个 <code>ISP_SHARPEN_GAIN_NUM=32</code> 的数组，是一个 32 段的连续的强度曲线，如图 6-2 所示。 取值范围：[0, 4095]。建议值 300。
au16EdgeStr	带方向的边缘的锐化强度，设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。该参数是一个 <code>ISP_SHARPEN_GAIN_NUM=32</code> 的数组，是一个 32 段的连续的强度曲线，如图 6-3 所示。 取值范围：[0, 4095]。建议值 400。
u16TextureFreq	图像的无方向细节纹理的增强频段控制。设置图像的细节纹理增强的频率。 该值越大，细节纹理的增强就越偏向于高频增强，



成员名称	描述
	细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。u16TextureFreq 对应于强度参数 u16TextureStr。u16TextureFreq 越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]。建议值 128。
u16EdgeFreq	图像的带方向的边缘的增强频段控制。设置图像边缘增强的频率。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。u16EdgeFreq 对应于强度参数 u16EdgeStr。u16EdgeFreq 越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]。建议值 96。
u8OverShoot	设置图像的 overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 100。
u8UnderShoot	设置图像的 undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 127。
u8ShootSupStr	图像锐化后的 overshoot 和 undershoot 的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的 overshoot 和 undershoot 的宽度和幅度。 该值越大，锐化后的图像的 overshoot 和 undershoot 的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以



成员名称	描述
	后，会减弱人眼的锐度感受。该参数需要和 ShootSupAdj 配合使用。 取值范围：[0, 255]。建议值 8。
u8ShootSupAdj	图像锐化后的 overshoot 和 undershoot 的抑制强度的调节。该参数配合 ShootSupStr 使用，用于调节 ShootSupStr 作用的区域范围。该值越小，则越多的纹理区域的 shoot 会被 ShootSupStr 抑制；该值越大，则只有很强的边缘的 shoot 会被 ShootSupStr 抑制，纹理区域的 shoot 不会被抑制。 取值范围：[0, 15]。建议值 9。 3DNR 后 Sharpen 没有该接口。
u8DetailCtrl	图像的细节纹理区的 shoot 强度的控制。用于控制图像的细节纹理区域的 shoot 强度，shoot 越大，细节纹理区的清晰度越高。 取值范围：[0, 0xFF]。该值等于 128，则图像的细节纹理区域的 shoot 强度和大边缘的 shoot 强度一致，都分别等于 u8OverShoot 和 u8UnderShoot。该值大于 128，则图像的细节纹理的 shoot 强度大于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。该值小于 128，则图像的细节纹理的 shoot 强度小于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。
u8DetailCtrlThr	图像的细节纹理区的 shoot 强度的控制阈值。该值配合 DetailCtrl 使用，用于区分 DetailCtrl 所控制 shoot 的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的 shoot 会被 DetailCtrl 单独控制，而大于该阈值的边缘区域的 shoot 依然等于 u8OverShoot 和 u8UnderShoot。 取值范围：[0, 0xFF]，建议值 160。



成员名称	描述
u8EdgeFiltStr	边缘滤波强度调试参数：实现图像锐化边缘的范围和边缘平滑强度的控制。该值越大，判为边缘的区域越多、也越宽，EdgeStr 起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，EdgeStr 起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]。建议值 53。
u8EdgeFiltMaxCap	边缘滤波强度范围的调试参数：该值越大，边缘滤波的最大强度也最大，u8EdgeFiltStr 的可调试范围也越大；一般建议该值大小控制 30 以内。 取值范围：[0, 47]；建议值 18。 3DNR 后 Sharpen 没有该接口。
u8RGain	深红色区域的锐化增益控制。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。 3DNR 后 Sharpen 没有该接口。
u8GGain	绿色区域的锐化增益控制。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]；建议值 32。 3DNR 后 Sharpen 没有该接口。
u8BGain	深蓝色区域的锐化增益控制。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。 3DNR 后 Sharpen 没有该接口。
u8SkinGain	肤色区域的锐化增益控制。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]；建议值 23。 3DNR 后 Sharpen 没有该接口。
u16MaxSharpGain	图像锐化的最大增益限制值。该值越大，图像的锐



成员名称	描述
	化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，可以减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值 160。 3DNR 后 Sharpen 没有该接口。
u8WeakDetailGain	对弱纹理单独进行锐化的增益控制。该值等于 0，图像弱纹理和图像其他区域的锐化程度相当，该值越大，图像弱纹理的锐化程度越大，当该参数过大，可能带来白边加重甚至虚边的副作用。 取值范围：[0, 127]，建议值 0。 仅 Hi3516EV200/Hi3516EV300 支持该功能。 3DNR 后 Sharpen 没有该接口。

图6-1 Sharpen 的像素亮度 luma 和锐化强度 lumaWgt 的关系曲线

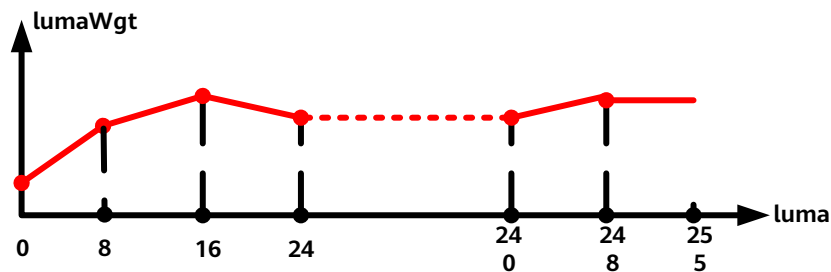
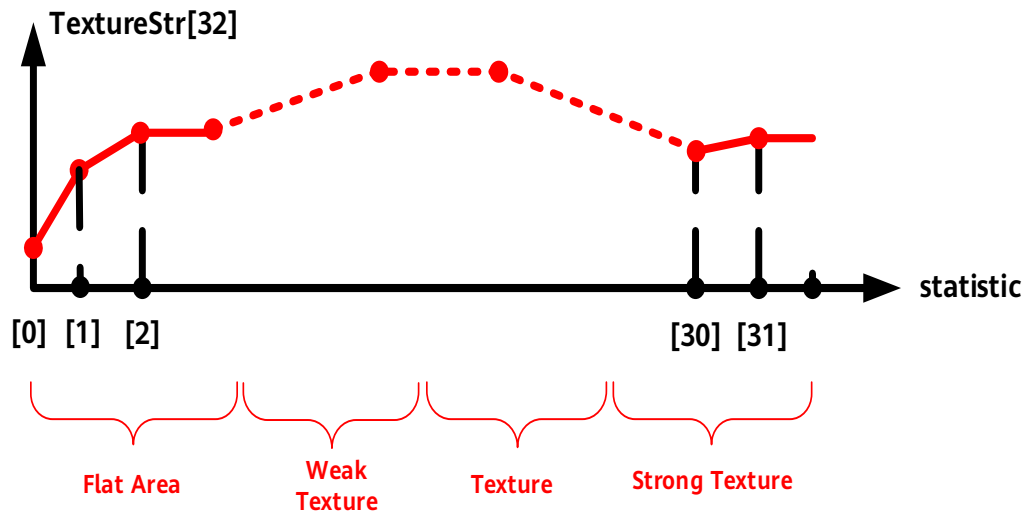
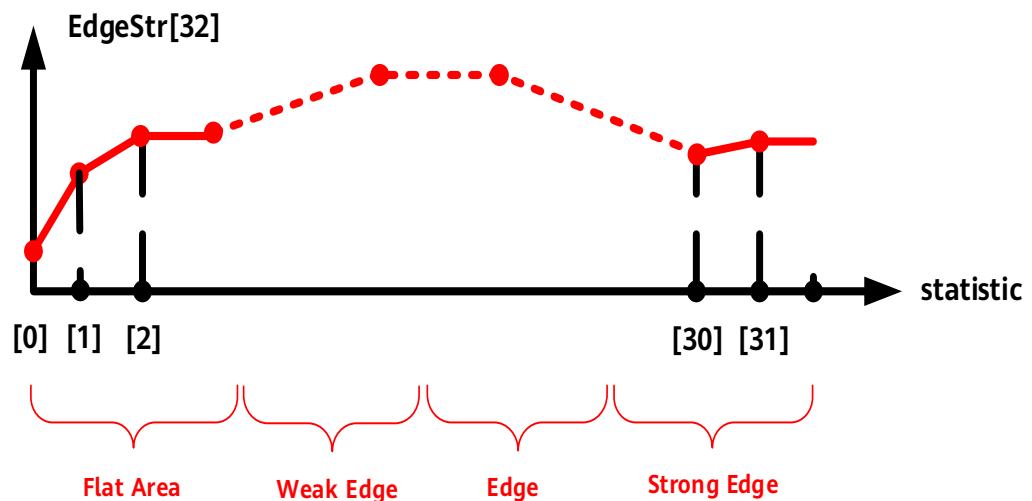


图6-2 TextureStr[ISP_SHARPEN_GAIN_NUM]强度曲线示意图



强度曲线的横坐标 var 是从图像中提取的方差统计特征，横坐标 var 被均分为 32 段，用于区分出图像的 Flat Area（平坦区域）、Weak Texture（弱纹理）、Texture（纹理）和 Strong Texture（强纹理）。纵坐标就是强度参数 TextureStr 的 32 个强度值，用户可以通过设置该曲线上的 32 个强度值来为平坦区域、弱纹理区域、纹理区域和强纹理区域设置不同的锐化强度。这 4 个区域并没有明显的区分界限，都是连续过渡，用户可以通过实际效果来调整纵坐标强度来为不同的区域设置不同的强度。

图6-3 EdgeStr[ISP_SHARPEN_GAIN_NUM]强度曲线示意图





强度曲线的横坐标 var 是从图像中提取的方差统计值，横坐标 var 被均分为 32 段，用于区分出图像的 Flat Area（平坦区域）、Weak Edge（弱边缘）、Edge（边缘）和 Strong Edge（强边缘）。纵坐标就是强度参数 EdgeStr 的 32 个强度值，用户可以通过设置该曲线上的 32 个强度值来为平坦区域、弱边缘、边缘和强边缘设置不同的锐化强度。这 4 个区域并没有明显的区分界限，都是连续过渡，用户可以通过实际效果来调整纵坐标强度来为不同的区域设置不同的强度。

ISP_SHARPEN_AUTO_ATTR_S

【说明】

定义 ISP Sharpen 自动属性。

【定义】

```
typedef struct hiISP_SHARPEN_AUTO_ATTR_S
{
    HI_U8
    au8LumaWgt[ISP_SHARPEN_LUMA_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeStr[ISP_SHARPEN_GAIN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16TextureFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16EdgeFreq[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8OverShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8UnderShoot[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8ShootSupStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8ShootSupAdj[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8DetailCtrl[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8DetailCtrlThr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8EdgeFiltStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8EdgeFiltMaxCap[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8RGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8GGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8BGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8SkinGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16 au16MaxSharpGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8WeakDetailGain[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_SHARPEN_AUTO_ATTR_S;
```



说明

宏定义如下：



```
#define ISP_SHARPEN_GAIN_NUM      (32)
#define ISP_SHARPEN_LUMA_NUM      (32)
#define ISP_AUTO_ISO_STRENGTH_NUM (16)
```

【成员】

成员名称	描述
au8LumaWgt	亮度锐化权重。满量程 0-255 的亮度被 32 个等分点平均分为 32 段亮度区间，每一段亮度区间对应一个亮度权重。比如 0-7 的亮度区间的权重是 au8LumaWgt[0]，8-15 的亮度区间的权重是 au8LumaWgt[1]，依次类推。如图 6-1 所示。值越大，图像锐化程度越高，反之，越弱。 Hi3516CV500/Hi3559V200 取值范围：[0, 127]。建议值 127。 Hi3516EV200/Hi3516EV300 取值范围：[0, 31]。建议值 31。 3DNR 后 Sharpen 中该参数与 ISO 不联动。
au16TextureStr	无方向的细节纹理的锐化强度，设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。该参数是一个 ISP_SHARPEN_GAIN_NUM=32 的数组，是一个 32 段的连续的强度曲线，如图 6-2 所示。 取值范围：[0, 4095]。建议值 300。
au16EdgeStr	带方向的边缘的锐化强度，设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。该参数是一个 ISP_SHARPEN_GAIN_NUM=32 的数组，是一个 32 段的连续的强度曲线，如图 6-3 所示。 取值范围：[0, 4095]。建议值 400。
au16TextureFreq	图像的无方向细节纹理的增强频段控制。设置图像的细节纹理增强的频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就



成员名称	描述
	越粗越圆润。u16TextureFreq 对应于强度参数 u16TextureStr。u16TextureFreq 越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]。建议值 128。
au16EdgeFreq	图像的带方向的边缘的增强频段控制。设置图像边缘增强的频率。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。u16EdgeFreq 对应于强度参数 u16EdgeStr。u16EdgeFreq 越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]。建议值 96。
au8OverShoot	设置图像的 overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 100。
au8UnderShoot	设置图像的 undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]。建议值 127。
au8ShootSupStr	图像锐化后的 overshoot 和 undershoot 的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的 overshoot 和 undershoot 的宽度和幅度。 该值越大，锐化后的图像的 overshoot 和 undershoot 的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。该参数需要和



成员名称	描述
	ShootSupAdj 配合使用。 取值范围：[0, 255]。建议值 8。
au8ShootSupAdj	图像锐化后的 overshoot 和 undershoot 的抑制强度的调节。该参数配合 ShootSupStr 使用，用于调节 ShootSupStr 作用的区域范围。该值越小，则越多的纹理区域的 shoot 会被 ShootSupStr 抑制；该值越大，则只有很强的边缘的 shoot 会被 ShootSupStr 抑制，纹理区域的 shoot 不会被抑制。 取值范围：[0, 15]。建议值 9。 3DNR 后 Sharpen 没有该接口。
au8DetailCtrl	图像的细节纹理区的 shoot 强度的控制。用于控制图像的细节纹理区域的 shoot 强度，shoot 越大，细节纹理区的清晰度越高。 取值范围：[0, 0xFF]。该值等于 128，则图像的细节纹理区域的 shoot 强度和大边缘的 shoot 强度一致，都分别等于 u8OverShoot 和 u8UnderShoot。该值大于 128，则图像的细节纹理的 shoot 强度大于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。该值小于 128，则图像的细节纹理的 shoot 强度小于大边缘，大边缘的 shoot 强度分别等于 u8OverShoot 和 u8UnderShoot。
au8DetailCtrlThr	图像的细节纹理区的 shoot 强度的控制阈值。该值配合 DetailCtrl 使用，用于区分 DetailCtrl 所控制 shoot 的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的 shoot 会被 DetailCtrl 单独控制，而大于该阈值的边缘区域的 shoot 依然等于 u8OverShoot 和 u8UnderShoot。 取值范围：[0, 0xFF]，建议值 160。
au8EdgeFiltStr	边缘滤波强度调试参数：实现图像锐化边缘的范围



成员名称	描述
	和边缘平滑强度的控制。该值越大，判为边缘的区域越多、也越宽，EdgeStr 起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，EdgeStr 起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]。建议值 53。
au8EdgeFiltMaxCap	边缘滤波强度范围的调试参数：该值越大，边缘滤波的最大强度也最大，u8EdgeFiltStr 的可调试范围也越大；一般建议该值大小控制 30 以内。 取值范围：[0, 47]；建议值 18。 3DNR 后 Sharpen 没有该接口。
au8RGain	深红色区域的锐化增益控制。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。 3DNR 后 Sharpen 没有该接口。
au8GGain	绿色区域的锐化增益控制。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]；建议值 32。 3DNR 后 Sharpen 没有该接口。
au8BGain	深蓝色区域的锐化增益控制。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。 3DNR 后 Sharpen 没有该接口。
au8SkinGain	肤色区域的锐化增益控制。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]；建议值 23。 3DNR 后 Sharpen 没有该接口。
au16MaxSharpGain	图像锐化的最大增益限制值。该值越大，图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该



成员名称	描述
	参数，可以减少图像的过锐化，可以减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值 160。 3DNR 后 Sharpen 没有该接口。
au8WeakDetailGain	对弱纹理单独进行锐化的增益控制。该值等于 0，图像弱纹理和图像其他区域的锐化程度相当，该值越大，图像弱纹理的锐化程度越大，当该参数过大，可能带来白边加重甚至虚边的副作用。 取值范围：[0, 127]，建议值 0。 仅 Hi3516EV200/Hi3516EV300 支持该功能。 3DNR 后 Sharpen 没有该接口。

【注意事项】

Auto 档参数分别对应 sensor 在 16 种不同增益 (Again*Dgain* ISPDgain(times)) 情况下的设置值，成员参数的含义与 Manual 档 cmos 成员参数一致。16 种增益为：1、2、4、8、16、32、64、128、256、512、1024、2048、4096、8192、16384、32768。

ISP_SHARPEN_ATTR_S

【说明】

定义 ISP Sharpen 属性。

【定义】

```
typedef struct hiISP_SHARPEN_ATTR_S
{
    HI_BOOL bEnable;
    HI_U8 u8SkinUmin;
    HI_U8 u8SkinVmin;
    HI_U8 u8SkinUmax;
    HI_U8 u8SkinVmax;
    ISP_OP_TYPE_E enOpType;
    ISP_SHARPEN_MANUAL_ATTR_S stManual;
```



```
ISP_SHARPEN_AUTO_ATTR_S stAuto;  
} ISP_SHARPEN_ATTR_S;
```

【成员】

成员名称	描述
bEnable	Sharpen 增强功能使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_TRUE。
u8SkinUmin	肤色区域范围矩形窗的左下角的最小坐标的 U 值, 取值范围: [0, 255]。
u8SkinVmin	肤色区域范围矩形窗的左下角的最小坐标的 V 值, 取值范围: [0, 255]。
u8SkinUmax	肤色区域范围矩形窗的右上角的最大坐标的 U 值, 取值范围: [0, 255]。
u8SkinVmax	肤色区域范围矩形窗的右上角的最大坐标的 V 值, 取值范围: [0, 255]。
enOpType	Sharpen 工作类型。 OP_TYPE_AUTO: 自动; OP_TYPE_MANUAL: 手动。 默认值为 OP_TYPE_AUTO。
stManual	Sharpen 手动参数。
stAuto	Sharpen 自动参数。

【注意事项】

Sharpen 功能分为自动和手动:

- bEnable 为 HI_TRUE, enOpType 为 OP_TYPE_AUTO, 使用自动 Sharpen 功能。
此时 sharpen 的强度值与系统增益的关系请参见成员变量。



- bEnable 设置为 HI_TRUE，enOpType 设置为 OP_TYPE_MANUAL 使用手动 Sharpen 功能。

【相关数据类型及接口】

无

6.2 EdgeMark

6.2.1 功能描述

EdgeMark 模块用于将图像中的强边缘和强的纹理特征标记上色。该功能用于手动对焦时，帮助用户直观的看清确定当前图像是否对焦清晰。建议用户只在手动对焦的过程中使用该功能，对强边缘和强的纹理特征标记的颜色尽量跟图像背景颜色形成明显的反差，这样，在对焦过程中，用户对焦越清晰，就会看到图像中被标记上色的强边缘和强的纹理特征越多，从而可以让用户直观的了解到对焦的清晰程度，辅助用户手动对焦。由于 EdgeMark 模块做在 Sharpen 模块一起，所以，当 Sharpen 模块被 bypass 关闭时，EdgeMark 的功能也将失效。但是，EdgeMark 被 bypass 关闭不会影响 Sharpen 的功能。Hi3516EV200/Hi3516EV300 不支持此模块。

6.2.2 API 参考

- [HI_MPI_ISP_SetIspEdgeMarkAttr](#)：设置图像的边缘标记属性。
- [HI_MPI_ISP_GetIspEdgeMarkAttr](#)：获取图像的边缘标记属性。

HI_MPI_ISP_SetIspEdgeMarkAttr

【描述】

设置图像边缘标记属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetIspEdgeMarkAttr(VI_PIPE ViPipe, const ISP\_EDGEMARK\_ATTR\_S
*pstIspEdgeMarkAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入



pstIspEdgeMarkAttr	图像的边缘标记属性。	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_GetIspEdgeMarkAttr](#)

HI_MPI_ISP_GetIspEdgeMarkAttr

【描述】

获取图像的边缘标记属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetIspEdgeMarkAttr(VI_PIPE ViPipe, ISP\_EDGEMARK\_ATTR\_S
*pstIspEdgeMarkAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入



pstIspEdgeMarkAttr	图像的边缘标记属性。	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_SetIspEdgeMarkAttr](#)

6.2.3 数据类型

[ISP_EDGEMARK_ATTR_S](#)：定义 ISP EdgeMark 属性。

ISP_EDGEMARK_ATTR_S

【说明】

定义 ISP EdgeMark 属性。

【定义】

```
typedef struct hiISP_EDGEMARK_ATTR_S
{
    HI_BOOL bEnable;
    HI_U8    u8Threshold;
```




```
HI_U32    u32Color;  
} ISP_EDGE MARK_ATTR_S;
```

【成员】

成员名称	描述
bEnable	EdgeMark 边缘标记功能的使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u8Threshold	边缘标记的阈值。计算得到的图像的局部统计量大于该阈值, 则认为是强边缘或者强的纹理特征, 将大于该阈值的强边缘和强的纹理特征标记上色。 取值范围: [0, 255]。默认值 100。 该值越小, 则被标记的边缘和强的纹理特征越多。
u32Color	边缘标记的颜色设置值。该值设置的是 RGB 三分量值的大小。该值的 0-7bit 的低 8 位是表示 B 分量的大小, 8-15bit 的中间 8 位是 G 分量的大小, 16-23bit 的高 8 位是 R 分量的大小, 24-31bit 的最高 8 位保留不用。通过设置 u32Color 可以为边缘标记的边缘标记上色选择合适的颜色, 一般所选的颜色最好和图像的背景颜色形成明显的反差对比。 取值范围: [0, 0xFFFFFFFF]。默认值 0xFF0000。

6.3 HLC

6.3.1 功能描述

HLC (High Light Cover) 高光遮蔽模块, 是用于将图像中亮度最高的区域 (一般都是夜晚场景的车灯或者路灯等光源) 拉灰, 以减少类似于车灯等高亮光源对人眼的刺激。Hi3559AV100/Hi3516EV200/Hi3516EV300 不支持此模块。



6.3.2 API 参考

- [HI_MPI_ISP_SetIspHlcAttr](#): 设置高光遮蔽的属性。
- [HI_MPI_ISP_GetIspHlcAttr](#): 获取高光遮蔽的属性。

HI_MPI_ISP_SetIspHlcAttr

【描述】

设置高光遮蔽的属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetIspHlcAttr(VI_PIPE ViPipe, const ISP\_HLC\_ATTR\_S *pstIspHlcAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspHlcAttr	图像的高光遮蔽属性。	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_SetIspHlcAttr](#)

HI_MPI_ISP_GetIspHlcAttr

【描述】

获取图像的高光遮蔽属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetIspHlcAttr(VI_PIPE ViPipe, ISP\_HLC\_ATTR\_S *pstIspHlcAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspHlcAttr	图像的高光遮蔽属性。	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_GetIspHlcAttr

6.3.3 数据类型

ISP_HLC_ATTR_S: 定义 ISP HLC 属性。

ISP_HLC_ATTR_S

【说明】

定义 ISP HLC 属性。

【定义】

```
typedef struct hiISP_HLC_ATTR_S
{
    HI_BOOL bEnable;
    HI_U8   u8LumaThr;
    HI_U8   u8LumaTarget;
} ISP_HLC_ATTR_S;
```

【成员】

成员名称	描述
bEnable	HLC 高光遮蔽功能的使能。 HI_FALSE: 关闭; HI_TRUE: 使能。 默认值为 HI_FALSE。
u8LumaThr	高光遮蔽的阈值。亮度值大于该值的区域将会被拉灰成目标值为 u8LumaTarget 的低亮度值。 取值范围: [0, 255]。默认值 248。 该值一般都设置的比较大, 否则, 图像的很多区域都会被拉灰。
u8LumaTarget	高光遮蔽拉灰的目标值。亮度值大于 u8LumaThr 的区域将被拉灰成目标值为 u8LumaTarget 的低亮度值, 也即该区域的亮度将会被明显拉低拉暗。 取值范围: [0, 255]。默认值 10。 该值一般都设置的比较小, 以减少高亮光源对人眼的刺激。



6.4 Gamma

6.4.1 功能描述

Gamma 模块对图像进行亮度空间非线性转换以适配输出设备。Gamma 模块校正 R、G、B 时调用同一组 Gamma 表，Gamma 表各节点之间的间距相同，节点之间的图像像素值使用线性插值生成。

6.4.2 API 参考

- [HI_MPI_ISP_SetGammaAttr](#)：设置 Gamma 属性。
- [HI_MPI_ISP_GetGammaAttr](#)：获取 Gamma 属性。

HI_MPI_ISP_SetGammaAttr

【描述】

设置 Gamma 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetGammaAttr(VI_PIPE ViPipe, const ISP\_GAMMA\_ATTR\_S *pstGammaAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstGammaAttr	Gamma 属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_GetGammaAttr](#)

HI_MPI_ISP_GetGammaAttr

【描述】

获取 Gamma 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetGammaAttr(VI_PIPE ViPipe, ISP_GAMMA_ATTR_S *pstGammaAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstGammaAttr	Gamma 属性	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetGammaAttr](#)

6.4.3 数据类型

- [ISP_GAMMA_ATTR_S](#)：定义 ISP Gamma 校正属性。
- [ISP_GAMMA_CURVE_TYPE_E](#)：定义 ISP Gamma 传输曲线类型。

ISP_GAMMA_ATTR_S

【说明】

定义 ISP Gamma 校正属性。

【定义】

```
typedef struct hiISP_GAMMA_ATTR_S
{
    HI_BOOL          bEnable;
    HI_U16           u16Table[GAMMA_NODE_NUM];
    ISP\_GAMMA\_CURVE\_TYPE\_E enCurveType;
}ISP_GAMMA_ATTR_S;
```



说明

宏定义如下：

```
#define GAMMA_NODE_NUM    (1025)
```

【成员】

成员名称	描述
bEnable	Gamma 校正功能使能。 HI_FALSE：关闭；



成员名称	描述
	HI_TRUE: 使能。 默认值为 HI_TRUE。
u16Table[GAMMA_NODE_NUM]	1024 段 LUT 表, 用来表示输入输出值大小。 取值范围: [0x0, 0xFFFF]
enCurveType	用来表示传输曲线类型 默认值为 ISP_GAMMA_CURVE_DEFAULT。

【注意事项】

- Gamma 校正 R、G、B 调用同一组 Gamma Table。
- enCurveType 选择为 ISP_GAMMA_CURVE_DEFAULT、ISP_GAMMA_CURVE_SRGB、ISP_GAMMA_CURVE_HDR 等三种模式时, 系统会自动配置预设的曲线, 此时通过工具拖拉曲线均不生效。只有选择 ISP_GAMMA_CURVE_USER_DEFINE 时, 才能通过工具进行曲线修改。
- 使用任意类型的预设曲线后, 可以切换成 ISP_GAMMA_CURVE_USER_DEFINE 并以预设曲线为基础进行 Gamma 调试。

【相关数据类型及接口】

无。

ISP_GAMMA_CURVE_TYPE_E

【说明】

定义 Gamma 曲线类型

【定义】

```
typedef enum hiISP_GAMMA_CURVE_TYPE_E
{
    ISP_GAMMA_CURVE_DEFAULT = 0x0,
    ISP_GAMMA_CURVE_SRGB,
    ISP_GAMMA_CURVE_HDR,
    ISP_GAMMA_CURVE_USER_DEFINE,
    ISP_GAMMA_CURVE_BUTT
} ISP_GAMMA_CURVE_TYPE_E;
```




【成员】

成员名称	描述
ISP_GAMMA_CURVE_DEFAULT	默认传输曲线
ISP_GAMMA_CURVE_SRGB	默认 SDR 传输曲线，由 BT.709 标准定义
ISP_GAMMA_CURVE_HDR	默认 HDR 传输曲线，由 SMPTE.2084 标准定义，仅在 Hi3559AV100 HDR 中支持。
ISP_GAMMA_CURVE_USER_DEFINE	用户自定义传输曲线
ISP_GAMMA_CURVE_BUTT	传输曲线类型枚举终止位指示

【注意事项】

用户自定义 Gamma 曲线时，必须确保 Gamma 表配置正确。WDR 模式下的 Gamma 曲线与线性模式不一样，可以通过 PQ Tools 工具生成。

【相关数据类型及接口】

无。

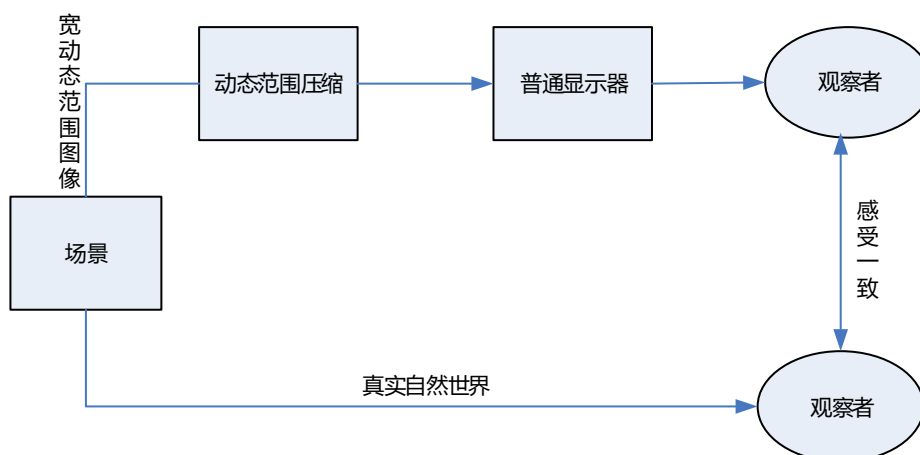
6.5 DRC

6.5.1 功能描述

DRC 算法用于对 WDR 合成后的数据进行动态范围压缩 (Dynamic Range Compression)。图像一般需要在显示设备上显示，CRT 显示器的动态范围一般只有 10^2 dB，而 WDR 合成后的数据的动态范围可以达到 10^9 dB，如果直接在 CRT 显示器上显示，就会由于动态范围不匹配的问题，造成丢失亮度较高或者较低处的细节。

DRC 算法的目的就是要使真实场景的观察者和显示设备的观察者都能获得相同的视觉感受。DRC 算法将高动态范围的图像压缩到显示器的动态范围，同时尽可能的保留原图像的细节和对比度。

图6-4 动态范围压缩



6.5.2 API 参考

- [HI_MPI_ISP_SetDRCAttr](#): 设置动态范围压缩参数。
- [HI_MPI_ISP_GetDRCAttr](#): 获取动态范围压缩参数。

HI_MPI_ISP_SetDRCAttr

【描述】

设置动态范围压缩参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetDRCAttr(VI_PIPE ViPipe, const ISP_DRC_ATTR_S *pstDRC);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDRC	动态范围压缩参数。	输入

【返回值】

返回值	描述
0	成功。



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】

无。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetDRCAttr](#)

HI_MPI_ISP_GetDRCAttr

【描述】

获取动态范围压缩参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetDRCAttr(VI_PIPE ViPipe, ISP\_DRC\_ATTR\_S *pstDRC);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDRC	动态范围压缩参数。	输入

【返回值】

返回值	描述
0	成功。



非 0	失败，其值为 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_SetDRCAttr](#)

6.5.3 数据类型

- [ISP_DRC_MANUAL_ATTR_S](#)：定义 ISP DRC 手动属性。
- [ISP_DRC_AUTO_ATTR_S](#)：定义 ISP DRC 自动属性。
- [ISP_DRC_CURVE_SELECT_E](#)：定义 ISP DRC Tone Mapping 曲线类型。
- [ISP_DRC_CUBIC_POINT_ATTR_S](#)：定义 ISP DRC Cubic Point 属性。
- [ISP_DRC_ASYMMETRY_CURVE_ATTR_S](#)：定义 ISP DRC Asymmetry curve 属性。
- [ISP_DRC_ATTR_S](#)：定义 ISP DRC 属性。

ISP_DRC_MANUAL_ATTR_S

【说明】

定义 ISP DRC 手动属性。

【定义】

```
typedef struct hiISP_DRC_MANUAL_ATTR_S
{
    HI_U16    u16Strength;
} ISP_DRC_MANUAL_ATTR_S;
```



【成员】

成员名称	描述
u16Strength	手动模式下 DRC 的强度，值越大，整体图像越亮。 - Hi3559AV100 取值范围：[0x0, 0xFF] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0x3FF]

【注意事项】

在线性和 WDR 模式下均可以达到最大值。

【相关数据类型及接口】

[ISP_DRC_ATTR_S](#)

ISP_DRC_AUTO_ATTR_S

【说明】

定义 ISP DRC 自动属性。

【定义】

```
typedef struct hiISP_DRC_AUTO_ATTR_S
{
    HI_U16  u16Strength;
    HI_U16  u16StrengthMax;
    HI_U16  u16StrengthMin;
} ISP_DRC_AUTO_ATTR_S;
```

【成员】

成员名称	描述
u16Strength	自动模式下 DRC 的强度，参数可以读写，值越大，整体强度越大。 - Hi3559AV100 取值范围：[0x0, 0xFF] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0,



成员名称	描述
	0x3FF]
u16StrengthMax	自动模式下 DRC 实际生效强度值的上限。取值范围同 u16Strength。
u16StrengthMin	自动模式下 DRC 实际生效强度值的下限。取值范围同 u16Strength。

【注意事项】

- 对于 Hi3559AV100，实际生效强度为：算法 AUTO 模式计算出的强度 * u16Strength / 128；对于 Hi3519AV100/Hi3516CV500 /Hi3559V200/Hi3516EV200/Hi3516EV300，实际生效强度为：算法 AUTO 模式计算出的强度 * u16Strength / 512。
- 注意 u16StrengthMax 必须大于等于 u16StrengthMin，否则两个参数均不生效。
- 线性和 WDR 自动模式下 DRC 强度的建议取值范围不同，u16StrengthMax/Min 的建议取值范围也不同：
 - 在 Hi3559AV100 线性自动模式下，u16StrengthMax/Min 的建议取值范围为 [0, 0x80]，其中 u16StrengthMax 默认值为 0x80，u16StrengthMin 默认值为 0x0；在 Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 线性自动模式下，u16StrengthMax/Min 的建议取值范围为 [0, 0x200]，其中 u16StrengthMax 默认值为 0x200，u16StrengthMin 默认值为 0x0；
 - 在 Hi3559AV100 WDR 自动模式下，u16StrengthMax/Min 的建议取值范围为 [0x80, 0xFF]，其中 u16StrengthMax 默认值为 0xFF，u16StrengthMin 默认值为 0x80；在 Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV300 WDR 自动模式下，u16StrengthMax/Min 的建议取值范围为 [0x200, 0x3FF]，其中 u16StrengthMax 默认值为 0x3FF，u16StrengthMin 默认值为 0x200。

【相关数据类型及接口】

[ISP_DRC_ATTR_S](#)

ISP_DRC_CUBIC_POINT_ATTR_S

【说明】

定义 ISP DRC Cubic Point 属性。



【定义】

```
typedef struct hiISP_DRC_CUBIC_POINT_ATTR_S
{
    HI_U16 u16X;
    HI_U16 u16Y;
    HI_U16 u16Slope;
} ISP_DRC_CUBIC_POINT_ATTR_S;
```

【成员】

成员名称	描述
u16X	点的横坐标。
u16Y	点的纵坐标。
u16Slope	点的斜率。

【注意事项】

仅 Hi3559AV100 支持。

【相关数据类型及接口】

无

ISP_DRC_ASYMMETRY_CURVE_ATTR_S

【说明】

定义 ISP DRC Asymmetry curve 属性。

【定义】

```
typedef struct hiISP_DRC_ASYMMETRY_CURVE_ATTR_S
{
    HI_U8    u8Asymmetry;
    HI_U8    u8SecondPole;
    HI_U8    u8Stretch;
    HI_U8    u8Compress;
} ISP_DRC_ASYMMETRY_CURVE_ATTR_S;
```

【成员】



成员名称	描述
u8Asymmetry	用于生成 Asymmetry Curve。
u8SecondPole	用于生成 Asymmetry Curve。
u8Stretch	用于生成 Asymmetry Curve。
u8Compress	用于生成 Asymmetry Curve。

【注意事项】

无

【相关数据类型及接口】

[ISP_DRC_ASYMMETRY_CURVE_ATTR_S](#)

ISP_DRC_CURVE_SELECT_E

【说明】

定义 ISP DRC Tone Mapping 曲线类型。

【定义】

```
typedef enum hiISP_DRC_CURVE_SELECT_E
{
    DRC_CURVE_ASYMMETRY = 0x0,
    DRC_CURVE_CUBIC,
    DRC_CURVE_USER,
    DRC_CURVE_BUTT
} ISP_DRC_CURVE_SELECT_E;
```

【成员】

成员名称	描述
DRC_CURVE_ASYMMETRY	选择 Asymmetry Curve。
DRC_CURVE_CUBIC	选择 Cubic Curve。
DRC_CURVE_USER	选择用户自定义曲线。
DRC_CURVE_BUTT	无效值。



【注意事项】

DRC_CURVE_CUBIC 类型仅 Hi3559AV100 支持。

【相关数据类型及接口】

[ISP_DRC_CURVE_SELECT_E](#)

ISP_DRC_ATTR_S

【说明】

定义 ISP DRC 属性。

【定义】

```
typedef struct hiISP_DRC_ATTR_S
{
    HI_BOOL bEnable;
    ISP\_DRC\_CURVE\_SELECT\_E enCurveSelect;
    HI_U8 u8PDStrength;
    HI_U8 u8LocalMixingBrightMax;
    HI_U8 u8LocalMixingBrightMin;
    HI_U8 u8LocalMixingBrightThr;
    HI_S8 s8LocalMixingBrightSlo;
    HI_U8 u8LocalMixingDarkMax;
    HI_U8 u8LocalMixingDarkMin;
    HI_U8 u8LocalMixingDarkThr;
    HI_S8 s8LocalMixingDarkSlo;
    HI_U8 u8DetailBrightStr;
    HI_U8 u8DetailDarkStr;
    HI_U8 u8DetailBrightStep;
    HI_U8 u8DetailDarkStep;
    HI_U8 u8BrightGainLmt;
    HI_U8 u8BrightGainLmtStep;
    HI_U8 u8DarkGainLmtY;
    HI_U8 u8DarkGainLmtC;
    HI_U16 au16ColorCorrectionLut[HI\_ISP\_DRC\_CC\_NODE\_NUM];
    HI_U16 au16ToneMappingValue[HI\_ISP\_DRC\_TM\_NODE\_NUM];
    HI_U8 u8FltScaleCoarse;
    HI_U8 u8FltScaleFine;
```



```

    HI_U8  u8ContrastControl;
    HI_S8  s8DetailAdjustFactor;
    HI_U8  u8SpatialFltCoef;
    HI_U8  u8RangeFltCoef;
HI_U8  u8RangeAdaMax;
    HI_U8  u8GradRevMax;
    HI_U8  u8GradRevThr;
    HI_U8  u8DpDetectRangeRatio;
    HI_U8  u8DpDetectThrSlo;
    HI_U16 u16DpDetectThrMin;
    ISP_OP_TYPE_E enOpType;
    ISP_DRC_MANUAL_ATTR_S stManual;
    ISP_DRC_AUTO_ATTR_S  stAuto;
    ISP_DRC_CUBIC_POINT_ATTR_S astCubicPoint[HI_ISP_DRC_CUBIC_POINT_NUM];
    ISP_DRC_ASYMMETRY_CURVE_ATTR_S stAsymmetryCurve;
} ISP_DRC_ATTR_S;

```



说明

宏定义如下：

```

#define HI_ISP_DRC_CC_NODE_NUM    (33)
#define HI_ISP_DRC_TM_NODE_NUM    (200)
#define HI_ISP_DRC_CUBIC_POINT_NUM    (5)

```

【成员】

成员名称	描述	取值范围	默认值
bEnable	DRC 动态范围压缩使能 0：关闭； 1：使能。	[0,1]	在线性模式下为0，WDR模式下为1
enCurveSelect	DRC Tone mapping 曲线选择 0：Asymmetry curve； 1：Cubic curve； 2：用户自定义曲线 Cubic curve 仅 Hi3559AV100 支持，因此，以下 astCubicPoint 中所有参数在 Hi3559AV100 以	[0x0,0x2]	-



成员名称	描述	取值范围	默认值
	外的芯片均不支持		
u8PDStrength	紫边校正强度，值越大，紫边校正强度越大。 Hi3516EV200/Hi3516EV300 不支持	[0x0,0x80]	-
u8LocalMixingBrightMax	线性细节增强：用来控制正向细节的增益最大值。	[0x0,0x80]	-
u8LocalMixingBrightMin	线性细节增强：用来控制正向细节的增益最小值。	[0x0,0x40]	-
u8LocalMixingBrightThr	线性细节增强：用来控制正向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最大值向最小值过渡（或反之，取决于斜率参数的符号）。	[0x0,0xFF]	-
s8LocalMixingBrightSlo	线性细节增强：用来控制正向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。	[-0x7, 0x7]	-
u8LocalMixingDarkMax	线性细节增强：用来控制负向细节的增益最大值。	[0x0, 0x80]	-
u8LocalMixingDarkMin	线性细节增强：用来控制负向细节的增益最小值。	[0x0, 0x40]	-
u8LocalMixingDarkThr	线性细节增强：用来控制负向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最小值向最大值过渡（或反之，取决于斜率参数的符号）。	[0x0, 0xFF]	-
s8LocalMixingDarkSlo	线性细节增强：用来控制负向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。	[-0x7, 0x7]	-



成员名称	描述	取值范围	默认值
u8DetailBrightStr	非线性细节增强：正向细节的增强 (仅 Hi3559AV100 支持)。	[0x0,0x80]	-
u8DetailDarkStr	非线性细节增强：负向细节的增强 (仅 Hi3559AV100 支持)。	[0x0,0x80]	-
u8DetailBrightStep	非线性细节增强：正向细节增强步长 (仅 Hi3559AV100 支持)。	[0x0,0x80]	-
u8DetailDarkStep	非线性细节增强：负向细节增强步长 (仅 Hi3559AV100 支持)。	[0x0,0x80]	-
u8BrightGainLmt	亮区亮度增益限制目标值；值越大，限制越大。	[0x0,0xF]	-
u8BrightGainLmtStep	亮区亮度增益限制自适应步长；值越小，限制越大。	[0x0,0xF]	-
u8DarkGainLmtY	暗区亮度增益限制；值越大，限制越大。	[0x0,0x85]	-
u8DarkGainLmtC	暗区色度增益限制；值越大，限制越大。	[0x0,0x85]	-
au16ColorCorrectionLut[HI_ISP_DRC_CC_NODE_NUM]	颜色校正表，值越小，饱和度越低。	[0x0,0x400]	-
au16ToneMappingValue[HI_ISP_DRC_TM_NODE_NUM]	用户自定义曲线 LUT。	[0x0, 0xffff]	-
u8FltScaleCoarse	控制滤波窗口大小。值越大，运动 halo 越不明显，局部细节越弱。相比 u8FltScaleFine，其调节步长更大。 Hi3516CV500/Hi3559V200/Hi3	[0x0, 0xF]	-



成员名称	描述	取值范围	默认值
	516EV200/Hi3516EV300 不支持。		
u8FltScaleFine	控制滤波窗口大小。值越大，运动 halo 越不明显，局部细节越弱。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。	[0x0, 0xF]	-
u8ContrastControl	全局对比度控制参数。参数效果与图像亮度分布相关，一般场景下建议值为 6 到 10 之间。	[0x0, 0xF]	-
s8DetailAdjustFactor	细节微调系数。值越大，整体细节越强；值越小，整体细节越弱。	[-0xF, 0xF]	
u8SpatialFltCoef	控制空域滤波系数。值越大，运动 halo 越不明显，细节越少，值越小，细节表现越好，运动 halo 越明显。	Hi3559AV100 取值范围：[0x0, 0xA]； Hi3519AV100/Hi3516CV500/Hi3559V200 取值范围：[0x0, 0x5]。	-
u8RangeFltCoef	控制值域滤波系数。值越大，halo 越明显；值越小，halo 表现越好，但是在强边缘处可能会出现边线。	[0x0, 0xA]	-
u8RangeAdaMax	控制值域滤波系数的自适应范围。实际生效滤波系数在 [u8RangeFltCoef, MIN(u8RangeFltCoef + u8RangeAdaMax, 0xA)] 范围内。调大该值有助于减弱边线，	[0x0, 0x8]	-



成员名称	描述	取值范围	默认值
	但同时可能造成背光物体细节损失。		
u8GradRevMax	去边线强度。值越大，边线减弱越明显，但是可能会带来细节损失。	[0x0, 0x40]	-
u8GradRevThr	边线检测阈值。值越大，边线减弱越明显，但是可能会带来细节损失。	[0x0, 0x80]	-
u8DpDetectRangeRatio	控制 DRC 内部坏点检测条件。值越小，检测条件越宽松，漏检概率降低，误检概率增加。仅 Hi3516EV200/Hi3516EV300 支持。	[0x0, 0x1F]	
u8DpDetectThrSlo	控制 DRC 内部坏点检测条件。值越小，检测条件越宽松，漏检概率降低，误检概率增加。仅 Hi3516EV200/Hi3516EV300 支持。	[0x0, 0x1F]	
u16DpDetectThrMin	控制 DRC 内部坏点检测条件。值越小，检测条件越宽松，漏检概率降低，误检概率增加。仅 Hi3516EV200/Hi3516EV300 支持。	[0x0, 0xFFFF]	
enOpType	DRC 工作类型。 • OP_TYPE_AUTO：自动； • OP_TYPE_MANUAL：手动。	-	OP_TYPE_AUTO
stManual	DRC 手动参数，具体参见 ISP_DRC_MANUAL_ATTR_S	-	-
stAuto	DRC 自动参数，具体参见 ISP_DRC_AUTO_ATTR_S	-	-
astCubicPoint	第 0 个点横坐标，用于生成	0x0	-



成员名称	描述	取值范围	默认值
[0].u16X	cubic curve		
astCubicPoint [1].u16X	第 1 个点横坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [2].u16X	第 2 个点横坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [3].u16X	第 3 个点横坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [4].u16X	第 4 个点横坐标, 用于生成 cubic curve,默认为 0x3E8	0x3E8	-
astCubicPoint [0].u16Y	第 0 个点纵坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [1].u16Y	第 1 个点纵坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [2].u16Y	第 2 个点纵坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [3].u16Y	第 3 个点纵坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [4].u16Y	第 4 个点纵坐标, 用于生成 cubic curve	[0x0,0x3E8]	-
astCubicPoint [0].u16Slope	第 0 个点斜率, 用于生成 cubic curve	[0x0,0x2710]	-
astCubicPoint [1].u16Slope	第 1 个点斜率, 用于生成 cubic curve	[0x0,0x2710]	-
astCubicPoint [2].u16Slope	第 2 个点斜率, 用于生成 cubic curve	[0x0,0x2710]	-
astCubicPoint [3].u16Slope	第 3 个点斜率, 用于生成 cubic curve	[0x0,0x2710]	-
astCubicPoint [4].u16Slope	第 4 个点斜率, 用于生成 cubic curve	[0x0,0x2710]	-
stAsymmetryCurve .u8Asymmetry	用来生成 Asymmetry curve, 值越小, 暗区拉伸越大, 见图 6-	[0x1,0x1E]	-



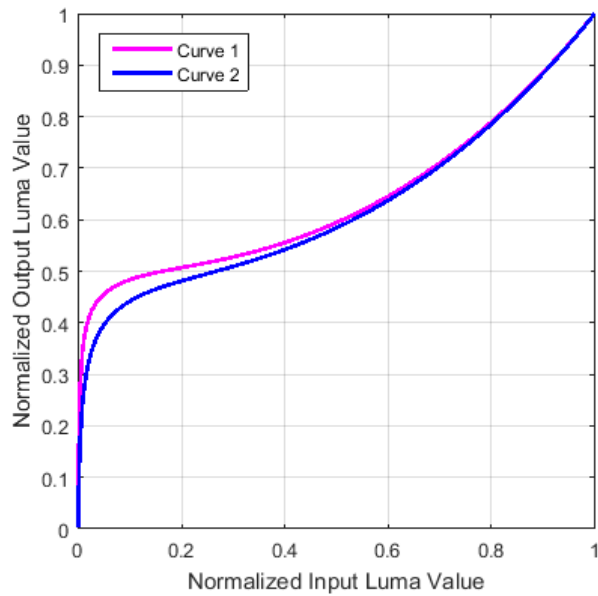
成员名称	描述	取值范围	默认值
	5。		
stAsymmetryCurve.u8SecondPole	用来生成 Asymmetry curve, 值越大, 整体亮度拉伸越大, 见图 6-6。	[0x96,0xD2]	-
stAsymmetryCurve.u8Stretch	用来生成 Asymmetry curve, 值越小, 整体图像越亮, 见图 6-7。	[0x1E,0x3C]	-
stAsymmetryCurve.u8Compress	用来生成 Asymmetry curve, compress 越大,在垂直方向压缩曲线程度越大, 整体亮度变低, 见图 6-8。	[0x64,0xC8]	-

【注意事项】

- DRC 功能分为自动和手动:
 - bEnable 为 HI_TRUE, enOpType 为 OP_TYPE_AUTO, 使用自动 DRC 功能。
 - bEnable 设置为 HI_TRUE, enOpType 设置为 OP_TYPE_MANUAL 使用手动 DRC 功能。
- u8Asymmetry, u8SecondPole 和 u8Stretch, u8Compress 的变化趋势如图 6-5 至图 6-8 所示设定:
 - Curve1(u8Asymmetry1, u8SecondPole1, u8Stretch1,u8Compress)
 - Curve2(u8Asymmetry2, u8SecondPole2, u8Stretch2, u8Compress)

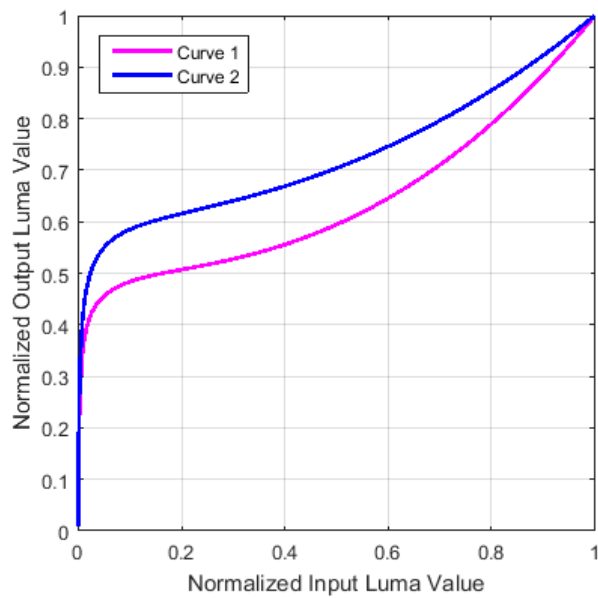
CASE1: Curve1 (2,180,54,180) Curve2 (5,180,54,180)

图6-5 u8Asymmetry 的变化对曲线的影响趋势



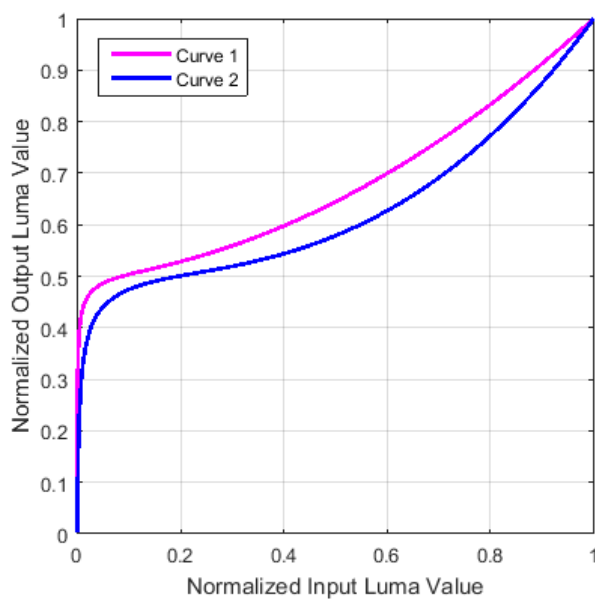
CASE2: Curve1 (2,180,54,180) Curve2 (2,200,54,180)

图6-6 u8SecondPole 的变化对曲线的影响趋势



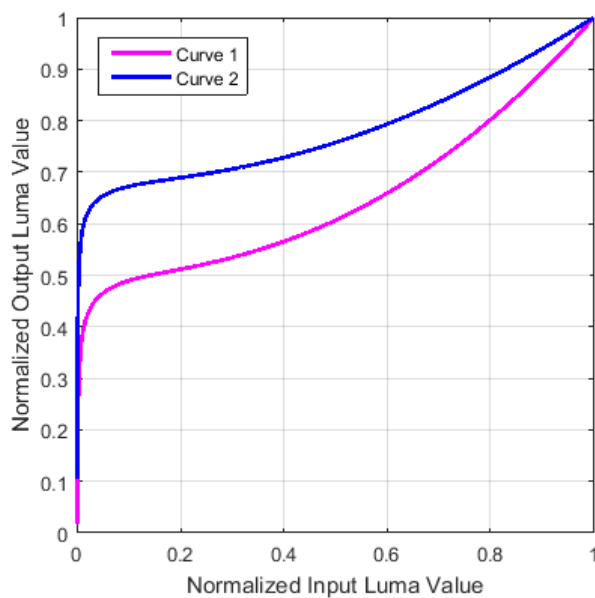
CASE3: Curve1 (2,180,40,180) Curve2 (2,180,60,180)

图6-7 u8Stretch 的变化对曲线的影响趋势



CASE4: Curve1 (2,180,50,180) Curve2 (2,180,50,100)

图6-8 u8Compress 的变化对曲线的影响趋势



- 建议 u16ColorCorrectionLut 值不要调试过小，饱和度降低过小，可能会带来高频细节的损失，用户可以配合 CA 模块来调试整体饱和度。



- u8DarkGainLmtY 和 u8DarkGainLmtC 两个接口的实际效果不完全平滑，即参数在两个相邻值之间切换时，效果可能存在跳变。

【相关数据类型及接口】

无。

6.6 Mesh Shading

6.6.1 功能描述

LSC 模块主要用来处理由于镜头光学折射不均匀导致的镜头周围出现阴影的情况。目前流行的处理方式有 Radial（同轴圆）方式以及 Mesh（网格）方式。

- 在 Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 芯片中 Mesh Shading 算法使用 Mesh（网格）方式对画面进行标定/矫正，算法会将整个 Bayer 域画面分割成 32*32 个子区域，这 32*32 个区域大小大致相等。
- 在 Hi3516EV200/Hi3516EV300 芯片中，算法会将整个 Bayer 域画面分割成 16*16 个子区域，这 16*16 个区域大小大致相等。

在数据处理过程中，算法将分别对 RAW 域中的每个通道进行处理。

6.6.2 API 参考

- [HI_MPI_ISP_SetMeshShadingAttr](#)：设定 Mesh Shading 算法参数。
- [HI_MPI_ISP_GetMeshShadingAttr](#)：获取 Mesh Shading 算法参数。
- [HI_MPI_ISP_SetMeshShadingGainLutAttr](#)：设定 Mesh Shading 增益表属性。
- [HI_MPI_ISP_GetMeshShadingGainLutAttr](#)：获取 Mesh Shading 增益表属性。

HI_MPI_ISP_SetMeshShadingAttr

【描述】

设置 Mesh Shading 算法参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetMeshShadingAttr(VI_PIPE ViPipe, const ISP\_SHADING\_ATTR\_S
*pstShadingAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstShadingAttr	Mesh Shading 算法参数。	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_GetMeshShadingAttr](#)

HI_MPI_ISP_GetMeshShadingAttr

【描述】

获取 Mesh Shading 算法参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetMeshShadingAttr(VI_PIPE ViPipe, ISP\_SHADING\_ATTR\_S *pstShadingAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstShadingAttr	Mesh Shading 算法参数。	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_SetMeshShadingAttr](#)

HI_MPI_ISP_SetMeshShadingGainLutAttr

【描述】

设定 Mesh Shading 增益表属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetMeshShadingGainLutAttr(VI_PIPE ViPipe, const  
ISP\_SHADING\_LUT\_ATTR\_S *pstShadingGainLutAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstShadingGainLutAttr	Mesh Shading 增益表属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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_GetMeshShadingGainLutAttr](#)

HI_MPI_ISP_GetMeshShadingGainLutAttr

【描述】

获取 Mesh Shading 增益表属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetMeshShadingGainLutAttr(VI_PIPE ViPipe, ISP\_SHADING\_LUT\_ATTR\_S  
*pstShadingGainLutAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstShadingGainLutAttr	Mesh Shading 增益表属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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_SetMeshShadingGainLutAttr](#)

6.6.3 数据类型

- [ISP_SHADING_ATTR_S](#)：定义 Mesh Shading 算法参数。
- [ISP_SHADING_GAIN_LUT_S](#)：定义 Mesh Shading 增益表参数。
- [ISP_SHADING_LUT_ATTR_S](#)：定义 Mesh Shading 增益表属性。

ISP_SHADING_ATTR_S

【说明】

定义 Mesh Shading 算法参数。



【定义】

```
typedef struct hiISP_SHADING_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16MeshStr;
    HI_U16  u16BlendRatio;
} ISP_SHADING_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	表示 Mesh Shading 使能	[0,1] 0: 禁止; 1: 使能	0
u16MeshStr	用来全局纠正 Mesh Shading 标定后的强度。当镜头 shading 很严重的时候, 画面四个角补偿的增益很大, 容易导致噪声很大, 而且画面有格子现象。这个调整 u16MeshStrength 小于一倍默认值, 可以减少 Mesh Shading 的补偿值, 让四个角的补偿增益较小一些, 达到优化噪声和画面格子问题。 - Hi3516EV200/Hi3516EV300 范围 [0,1023]; - Hi3519AV100/Hi3559AV100/Hi3516CV500/Hi3559V200 范围[0, 65535]。	[0, 65535]	- Hi3516EV200/Hi3516EV300 默认值为 256, 为一倍强度。 - 其它芯片默认值为 4096, 为一倍强度。
u16BlendRatio	两张增益表的融合比例, 表示为第 0 张表的权重值, 256 表示只选择第 0 张表, 0 表示只选择第 1 张表。	[0,256]	256

【注意事项】

无

【相关数据类型及接口】



- [HI_MPI_ISP_SetMeshShadingAttr](#)
- [HI_MPI_ISP_GetMeshShadingAttr](#)

ISP_SHADING_GAIN_LUT_S

【说明】

定义 Mesh Shading 增益表参数。

【定义】

```
typedef struct hiISP_SHADING_LUT_S
{
    HI_U16  au16RGain[HI_ISP_LSC_GRID_POINTS];
    HI_U16  au16GrGain[HI_ISP_LSC_GRID_POINTS];
    HI_U16  au16GbGain[HI_ISP_LSC_GRID_POINTS];
    HI_U16  au16BGain[HI_ISP_LSC_GRID_POINTS];
} ISP_SHADING_GAIN_LUT_S;
```



说明

Hi3516EV200/Hi3516EV300 宏定义如下：

```
#define HI_ISP_LSC_GRID_POINTS    (17*17)
```

其它芯片宏定义如下：

```
#define HI_ISP_LSC_GRID_POINTS    (33*33)
```

【成员】

成员名称	描述	取值范围	默认值
au16RGain[HI_ISP_LSC_GRID_POINTS]	用来储存 Mesh Shading 所用 R 通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的 R 分量阴影矫正增益数据值。	[0, 1023]	--
au16GrGain[HI_ISP_LSC_GRID_POINTS]	用来储存 Mesh Shading 所用 Gr 通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的 Gr 分量阴影矫正增益数据值。	[0, 1023]	--
au16GbGain[HI_ISP_LSC_GRID_POINTS]	用来储存 Mesh Shading 所用 Gb 通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的 Gb 分量阴影矫正增益数据值。	[0, 1023]	--



成员名称	描述	取值范围	默认值
au16BGain[HI_ISP_LSC_GRID_POINTS]	用来储存 Mesh Shading 所用 B 通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的 B 分量阴影矫正增益数据值。	[0, 1023]	--

【注意事项】

- 增益表的默认配置与 [ISP_CMOS_ALG_KEY_U](#) 中的 bit1(Lsc 标志位)有关，如果 bit1(Lsc=1)，则采用 cmos_ex.h 中的配置值作为默认值；否则默认配置为 1 倍增益。
- 增益表的精度与 u8MeshScale 的值有关系，详参[表 6-1](#)。在 u8MeshScale 配置为默认值 4 的情况下，增益表配置为 0 值表示增益为 1 倍。

【相关数据类型及接口】

- [ISP_SHADING_LUT_ATTR_S](#)
- [HI_MPI_ISP_SetMeshShadingGainLutAttr](#)
- [HI_MPI_ISP_GetMeshShadingGainLutAttr](#)

ISP_BNR_LSC_GAIN_LUT_S**【说明】**

定义 BayerNR 所参考的 LSC 增益表参数。

【定义】

```
typedef struct hiISP_BNR_LSC_GAIN_LUT_S
{
    HI_U16  au16RGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16GrGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16GbGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16BGain[HI_ISP_RLSC_POINTS];
} ISP_BNR_LSC_GAIN_LUT_S;
```

【成员】

成员名称	描述	取值范围	默认值
au16RGain[129]	用来储存 BNR LSC 所用 R 通道标定	[0, 65535]	--



成员名称	描述	取值范围	默认值
	数据。		
au16GrGain[129]	用来储存 BNR LSC 所用 Gr 通道标定数据。	[0, 65535]	--
au16GbGain[129]	用来储存 BNR LSC 所用 Gb 通道标定数据。	[0, 65535]	--
au16BGain[129]	用来储存 BNR LSC 所用 B 通道标定数据。	[0, 65535]	--

【注意事项】

Hi3559AV100/Hi3519AV100/Hi3516EV200/Hi3516EV300 不支持该数据类型。

【相关数据类型及接口】

- [ISP_SHADING_LUT_ATTR_S](#)
- [HI_MPI_ISP_SetMeshShadingGainLutAttr](#)
- [HI_MPI_ISP_GetMeshShadingGainLutAttr](#)

ISP_SHADING_LUT_ATTR_S

【说明】

定义 Mesh Shading 增益表属性。

【定义】

```
typedef struct hiISP_SHADING_LUT_ATTR_S
{
    HI_U8    u8MeshScale;
    HI_U16   au16XGridWidth[HI_ISP_MLSC_X_HALF_GRID_NUM];
    HI_U16   au16YGridWidth[HI_ISP_MLSC_Y_HALF_GRID_NUM];
    ISP\_SHADING\_GAIN\_LUT\_S astLscGainLut[HI_ISP_MLSC_GAIN_LUT_NUM];
    ISP\_BNR\_LSC\_GAIN\_LUT\_S stBNRLscGainLut;
} ISP_SHADING_LUT_ATTR_S;
```



说明

```
#define HI_ISP_MLSC_GAIN_LUT_NUM          2
#define HI_ISP_MLSC_X_HALF_GRID_NUM      ((HI_ISP_LSC_GRID_COL - 1) / 2)
#define HI_ISP_MLSC_Y_HALF_GRID_NUM      ((HI_ISP_LSC_GRID_ROW - 1) / 2)
```



Hi3516EV200/Hi3516EV300 宏定义如下:

```
#define HI_ISP_LSC_GRID_COL    (17)
#define HI_ISP_LSC_GRID_ROW    (17)
```

其他芯片宏定义如下:

```
#define HI_ISP_LSC_GRID_COL    (33)
#define HI_ISP_LSC_GRID_ROW    (33)
```

【成员】

成员名称	描述	取值范围	默认值
u8MeshScale	增益表精度控制参数。当选择的精度较低时（比如 MeshScale 为 3），画面衰减幅度大，可能会出现格子现象，可以提高精度（比如 2 或 7）重新标定	[0,7]	3
au16XGridWidth [(HI_ISP_LSC_GRID_COL - 1)/2]	用来储存各 GRID 分区宽度大小信息。该接口各分量最小值为 4，总和应为原画面宽度的四分之一。（例如原画面大小为 1080p，则该接口各参数总和应为 480）	[4,Width/4 - 60] Width 为原画面的宽度 Hi3516EV200/Hi3516EV300 为: [4, 255]	-
au16YGridWidth[(HI_ISP_LSC_GRID_ROW - 1)/2]	用来储存各 GRID 分区高度大小信息。该接口各分量最小值为 4，总和应为原画面高度的四分之一。（例如原画面大小为 1080p，则该接口各参数总和应为 270）	[4,Height/4 - 60] u32Height 为原画面的高度 Hi3516EV200/Hi3516EV300 为: [4, 255]	-
astLscGainLut[2]	两组色温下的增益表配置。硬件基于这两组表以及 u16BlendRatio 进行当前色温下校正增益表的计算	[0, 1023]	-
stBNRLscGainLut	用于 BNR LSC 参考所用的增益表。 Hi3559AV100/Hi3519AV100/Hi35	[0, 65535]	-



成员名称	描述	取值范围	默认值
	16EV200/Hi3516EV300 不支持该数据类型。		

【注意事项】

- 调整 au16XGridWidth 或 au16YGridWidth 里的值即调整分区宽度或者高度。注意由于整体画面为上下、左右对称，因此每当设置一个分区的宽度或者高度时，其对称分区的宽度或高度也会相应发生变化。改变后的分区宽度与高度请满足总和为原画面大小宽度或高度的四分之一。由于调节比较复杂并容易造成 Color Shading 现象，一般不建议手动调节。分辨率切换的时候，au16XgridWidth 与 au16YgridWidth 自动重新配置。
- 硬件基于配置的两组色温下的增益表以及 u16BlendRatio 进行插值得到当前色温下的校正增益表的计算，从而校正整幅图像的 shading 现象，其中 u16BlendRatio 为 astLscGainLut[0]对应的权重。
- 在单光源模式下，astLscGainLut 结构体需要配置成一致，此时需要配置成标定的第一个光源。
- 增益位宽为 10bit，即最大值为 1023。
 - 当 u8MeshScale 为 0 至 3 时，增益的表示与其他算法一致，增益精度从高到低，取值范围从小变大，1 倍增益不为 0；
 - 当 u8MeshScale 为 4 至 7 时，精度从高到低，取值范围从小变大，增益的表示方式略有差异，实际的增益计算方式为通用增益减去对应精度的 1 倍增益，比如 u8MeshScale 为 5 时，按照增益精度为 1.9，若通用增益为 886，1 倍对应增益 512，那么实际的增益为 886-512=374，这种表示方式下，最小的增益为 1 倍，但是表示的范围比上一种要大。
- 在能够完全校正的情况下，增益精度 u8MeshScale 应该下尽量选择精度高的档位，提高校正的准确度，例如当 u8MeshScale 为 0 或 4 时都能校正，应当选择 4。对于某些镜头阴影特别严重的极端情况下，存在无法完全校正的可能，当选择较大增益倍数（如 u8MeshScale 为 3）校正时，可能会在边缘处出现暗条纹，此时需要降低增益倍数来提高精度（设 u8MeshScale 为 2 或 7），重新标定，直到暗条纹消失为止。



表6-1 u8MeshScale 与增益精度对应关系

u8MeshScale	增益表的数据格式	增益取值范围	1 倍增益对应增益配置值
0	1.9	[0,2)	512
1	2.8	[0,4)	256
2	3.7	[0,8)	128
3	4.6	[0,16)	64
4	0.10	[1,2)	0
5	1.9	[1,3)	0
6	2.8	[1,5)	0
7	3.7	[1,9)	0

【相关数据类型及接口】

- [ISP_SHADING_GAIN_LUT_S](#)
- [ISP_BNR_LSC_GAIN_LUT_S](#)
- [HI_MPI_ISP_SetMeshShadingGainLutAttr](#)
- [HI_MPI_ISP_GetMeshShadingGainLutAttr](#)

6.7 Defect Pixel

6.7.1 功能描述

DPC 算法通过在 5x5 的窗口中通过某些坏点检测算法找到该窗口中明显异于临近像素的坏点。该模块主要包含以下两种模式：

- 静态坏点标定/校正有两种流程：亮点和暗点流程
 - 在亮点流程中，光圈处于关闭状态，启动坏点标定程序，得到坏点坐标信息。坏点个数的总数由坏点校正模块的 memory 决定的。得到的坏点通过临近像素的中值滤波进行校正。
 - 在暗点流程中，光圈处于正常打开状态，要求平坦背景，最好使用灰度箱固定光源，图像整体平均亮度大约为最大亮度 50%，或 Bayer 格式中 B 通道亮度为最大亮度 20%左右，启动坏点检测程序，得到坏点坐标。坏点个数的总数是由坏点校正模块的 memory 决定的。得到的坏点通过临近像素的中值滤波进行校正。



注意:

- 对于 Hi3559AV100/Hi3519AV100 DPC 模块所能支持的最大静态坏点个数为 8192, 对于 Hi3516CV500/Hi3559V200DPC 所能支持的最大静态坏点数为 6144。
- Hi3516EV200/Hi3516EV300 DPC 不支持静态坏点标定/校正。

- 动态坏点检测/校正

在这种模式中, 校正模块使用动态检测方法的进行坏点检测, 能够校正的坏点个数是有限制的。这种模式总体上相对于静态模式更不可靠, 但是在低照度情况下强烈推荐使用动态坏点校正功能, 可以校正更多的坏点以及明显改善图像偏色的问题。

6.7.2 API 参考

- [HI_MPI_ISP_SetDPCalibrate](#): 设置静态坏点标定参数。
- [HI_MPI_ISP_GetDPCalibrate](#): 获取静态坏点标定结果。
- [HI_MPI_ISP_SetDPStaticAttr](#): 设置静态坏点校正属性。
- [HI_MPI_ISP_GetDPStaticAttr](#): 获取静态坏点校正属性。
- [HI_MPI_ISP_SetDPDynamicAttr](#): 设置动态坏点校正属性。
- [HI_MPI_ISP_GetDPDynamicAttr](#): 获取动态坏点校正属性。

HI_MPI_ISP_SetDPCalibrate

【描述】

设置静态坏点标定参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetDPCalibrate(VI_PIPE ViPipe, const ISP_DP_STATIC_CALIBRATE_S  
*pstDPCalibrate);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPCalibrate	静态坏点标定参数	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 该接口为静态坏点标定接口，坏点标定只需启动一次，标定完成之后自动关闭标定使能，恢复为正常的坏点校正模式。亮坏点检测的环境为：使用最小的模拟增益和数字增益，降帧率到 5~6fps，使曝光时间为 200ms（系统自动完成），遮黑镜头或关闭光圈（需手动操作）；暗坏点检测的环境为：光圈处于正常打开状态，要求平坦背景，最好使用辉度箱固定光源，图像整体平均亮度大约为最大亮度 50%，或 Bayer 格式中 B 通道亮度为最大亮度 20%左右。
- Hi3559AV100 不支持离线模式下静态坏点标定。
- 支持 WDR 模式下静态坏点标定。WDR 模式下默认选择短帧进行静态坏点标定，推荐在标定前，通过 [HI_MPI_ISP_SetModuleControl](#) 接口的 bit2ChnSelect 变量选择长帧，再启动静态坏点标定。一般情况下，长帧的标定即可满足需求。若特殊情况下，长帧标定后仍然有坏点残留，可通过 bit2ChnSelect 变量选择中/短帧再次进行标定，并将多个坏点表合并成一个表。
- Hi3516EV200/Hi3516EV300 不支持静态坏点标定。

【举例】

以标定亮点为例：

```
ISP_DP_STATIC_CALIBRATE_S stDpcCalib;  
stDpcCalib.bEnableDetect= 1;//使能静态坏点标定  
  
stDpcCalib.enStaticDPTYPE = 0;  
stDpcCalib.u16CountMax = 1024;  
stDpcCalib.u16CountMin = 1;  
stDpcCalib.u8StartThresh = 1;  
stDpcCalib.u16TimeLimit = 0x640;
```




```
HI_MPI_ISP_SetDPCalibrate(ViPipe,&stDpcCalib);
```

【相关主题】

[HI_MPI_ISP_GetDPCalibrate](#)

HI_MPI_ISP_GetDPCalibrate

【描述】

获取静态坏点标定结果。

【语法】

```
HI_S32 HI_MPI_ISP_GetDPCalibrate(VI_PIPE ViPipe, ISP\_DP\_STATIC\_CALIBRATE\_S  
*pstDPCalibrate);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPCalibrate	静态坏点标定参数	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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】



以获取亮点标定结果为例：

```
HI_U16 i;
HI_U16 u16BrightBpNumber= 0;
HI_U32 au32BrightTable[STATIC_DP_COUNT_MAX];
HI_MPI_ISP_GetDPCalibrate(ViPipe,&stDpcCalib);
while(stDpcCalibrate.enStatus == ISP_STATE_INIT)
{
    sleep(1);
    HI_MPI_ISP_GetDPCalibrate(ViPipe,&stDpcCalibrate);
}
u16BrightBpNumber = stDpcCalib.u16Count;
for (i=0;i< u16BrightBpNumber;i++)
{
    au32BrightTable[i] = stDpcCalib.au32Table[i];
}
```

【相关主题】

[HI_MPI_ISP_SetDPCalibrate](#)

HI_MPI_ISP_SetDPStaticAttr

【描述】

设置静态坏点属性，合并坏点表，将坏点坐标信息与坏点数目写入系统内存。

【语法】

```
HI_S32 HI_MPI_ISP_SetDPStaticAttr(VI_PIPE ViPipe, const ISP\_DP\_STATIC\_ATTR\_S
*pstDPStaticAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPStaticAttr	静态坏点属性	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 每一路 ISP BE 所能支持的最大静态坏点个数为 STATIC_DP_COUNT_NORMAL(在 Hi3559AV100/Hi3519AV100 中，STATIC_DP_COUNT_NORMAL = 4096；在 Hi3516CV500/Hi3559V200 中，STATIC_DP_COUNT_NORMAL = 2048)，因此 DPC 模块所能支持的最大静态坏点个数为 $\text{Count} = \text{STATIC\_DP\_COUNT\_NORMAL} * (\text{当前业务场景下 ISP BE 的分块数目})$ 。则亮点数目 $\text{u16BrightCount} \leq \text{Count}$ 且 $\text{u16DarkCount} \leq \text{Count}$ ，若亮点表和暗点表中有 u16Same 个重复的坏点，则 $\text{u16BrightCount} + \text{u16DarkCount} - \text{u16Same} < = \text{Count}$ 。ISP BE 的分块数目参考 ISP Proc 信息中的“BlockNum”。
- Hi3516EV200/Hi3516EV300 不支持静态坏点校正。

【举例】

```
ISP_DP_STATIC_ATTR_S stDpcStaticAttr;
stDpcStaticAttr.u16BrightCount = u16BrightBpNumber;
stDpcStaticAttr.u16DarkCount = u16DarkBpNumber;
for (i=0;i< u16BrightBpNumber;i++)
{
    stDpcStaticAttr.au32BrightTable[i]= au32BrightTable [i];
}
for (i=0;i< u16DarkBpNumber;i++)
{
    stDpcStaticAttr.au32DarkTable[i]= au32DarkTable [i];
}
HI_MPI_ISP_SetDPAttr(ViPipe,&stDpcStaticAttr);
```

【相关主题】



HI_MPI_ISP_GetDPStaticAttr

HI_MPI_ISP_GetDPStaticAttr

【描述】

获取静态坏点属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDPStaticAttr(VI_PIPE ViPipe, ISP_DP_STATIC_ATTR_S *pstDPStaticAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPStaticAttr	静态坏点属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

在设置静态坏点属性中，亮点坏点表中表示亮坏点坐标信息，暗坏点表中则保存的是暗坏点坐标信息；而获取静态坏点属性时，亮点坏点表中保存的是所有坏点的坐标信息。

【举例】

```
HI_U16 u16BpNumber= 0;  
HI_U32 au32BpTable[STATIC_DP_COUNT_MAX];  
HI_MPI_ISP_GetDPStaticAttr(ViPipe, stDpcStaticAttr);
```



```
u16BpNumber = stDpcStaticAttr.u16BrightCount;
for (i=0;i< u16BpNumber;i++)
{
    au32BpTable[i]= stDpcStaticAttr.au32BrightTable[i];
}
```

【相关主题】

[HI_MPI_ISP_SetDPStaticAttr](#)

HI_MPI_ISP_SetDPDynamicAttr

【描述】

设置动态坏点调试属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetDPDynamicAttr(VI_PIPE ViPipe, const ISP\_DP\_DYNAMIC\_ATTR\_S
*pstDPDynamicAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPDynamicAttr	动态坏点属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_GetDPDynamicAttr](#)

HI_MPI_ISP_GetDPDynamicAttr

【描述】

获取动态坏点调试属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDPDynamicAttr(VI_PIPE ViPipe, ISP\_DP\_DYNAMIC\_ATTR\_S
*pstDPDynamicAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI pipe 号	输入
pstDPDynamicAttr	动态坏点属性	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetDPDynamicAttr](#)

6.7.3 数据类型

- [ISP_STATIC_DP_TYPE_E](#): 静态坏点标定类型。
- [ISP_STATUS_E](#): ISP 校正（检测）状态。
- [ISP_DP_STATIC_CALIBRATE_S](#): 静态坏点标定属性。
- [ISP_DP_STATIC_ATTR_S](#): 静态坏点校正属性。
- [ISP_DP_DYNAMIC_ATTR_S](#): 定义动态坏点校正属性。
- [ISP_DP_DYNAMIC_MANUAL_ATTR_S](#): 定义动态坏点校正的手动属性。
- [ISP_DP_DYNAMIC_AUTO_ATTR_S](#): 定义动态坏点校正的自动属性。

ISP_STATIC_DP_TYPE_E

【说明】

定义静态坏点标定类型。

【定义】

```
typedef enum hiISP_STATIC_DP_TYPE_E{  
    ISP_STATIC_DP_BRIGHT = 0x0,  
    ISP_STATIC_DP_DARK,  
    ISP_STATIC_DP_BUTT  
}ISP_STATIC_DP_TYPE_E;
```

【成员】

成员名称	描述
ISP_STATIC_DP_BRIGHT	0x0: 选择亮点标定
ISP_STATIC_DP_DARK	0x1: 选择暗点标定



【注意事项】

无。

【相关数据类型及接口】

[ISP_DP_STATIC_CALIBRATE_S](#)

ISP_STATUS_E

【说明】

定义 ISP 校正（检测）状态。

【定义】

```
typedef enum hiISP_STATE_E
{
    ISP_STATE_INIT      = 0,
    ISP_STATE_SUCCESS   = 1,
    ISP_STATE_TIMEOUT   = 2,
    ISP_STATE_BUTT
}ISP_STATUS_E;
```

【成员】

成员名称	描述
ISP_STATE_INIT	0：初始状态，未标定
ISP_STATE_SUCCESS	1：静态坏点标定成功结束
ISP_STATE_TIMEOUT	2：静态坏点标定超时结束

【注意事项】

只读。

【相关数据类型及接口】

[ISP_DP_STATIC_CALIBRATE_S](#)



ISP_DP_STATIC_CALIBRATE_S

【说明】

定义静态坏点标定的属性。

【定义】

```
typedef struct hiISP_DP_STATIC_CALIBRATE_S
{
    HI_BOOL bEnableDetect;
    ISP_STATIC_DP_TYPE_E enStaticDPTYPE;
    HI_U8   u8StartThresh;
    HI_U16  u16CountMax;
    HI_U16  u16CountMin;
    HI_U16  u16TimeLimit;
    HI_U32  au32Table[STATIC_DP_COUNT_MAX];
    HI_U8   u8FinishThresh;
    HI_U16  u16Count;
    ISP_STATUS_E enStatus;
} ISP_DP_STATIC_CALIBRATE_S;
```



说明

Hi3559AV100/Hi3519AV100 中宏定义如下：

```
#define STATIC_DP_COUNT_MAX    (8192)
```

Hi3516CV500/Hi3559V200 中宏定义如下：

```
#define STATIC_DP_COUNT_MAX    (6144)
```

【成员】

成员名称	描述	取值范围	默认值
bEnableDetect	静态坏点标定使能 Hi3516EV200/Hi3516EV300 不支持静态坏点标定，静态坏点标定相关参数无效。	[0,1] 0：禁止； 1：使能。	0
enStaticDPTYPE	静态坏点标定类型	[0,1] 0：亮点； 1：暗点	0
u8StartThresh	静态坏点标定开始时的检测门限值	[1, 0xFF]	0x3



成员名称	描述	取值范围	默认值
	该值的设置与 sensor 相关		
u16CountMax	允许静态坏点的最大个数	[0, STATIC_DP_COUNT_MAX]	0x400
u16CountMin	允许静态坏点的最小个数	[0, u16CountMax)	0x1
u16TimeLimit	允许标定超时门限值	[0, 0x640]	0x640
au32Table [STATIC_DP_COUNT_MAX]	只读，亮暗坏点坐标值查找表，低 29bit 有效，[12:0]bit 为坏点的水平坐标，[28:16]bit 为坏点的垂直坐标。	每个坐标取值范围：[0, 0x1FFF1FFF]	0
u8FinishThresh	只读，静态坏点标定结束时的检测门限值	[0, 0xFF]	0x3
u16Count	只读，标定出的静态坏点的个数	[0, STATIC_DP_COUNT_MAX]	0
enStatus	只读，静态坏点标定结果状态信息	[0,2]	0

【注意事项】

- u16CountMax 的配置值与当前的业务场景有关系。每一路 ISP BE 所能支持的静态坏点个数为 STATIC_DP_COUNT_NORMAL，则 u16CountMax 的最大值为 STATIC_DP_COUNT_NORMAL * (当前业务场景下 ISP BE 的分块数目)，ISP BE 的分块数目参考 ISP Proc 信息中的 "BlockNum"。
- 坏点检测算法检测成功标准：检测出的坏点数量是否在 [u16CountMin, u16CountMax)。所以不同类型的 sensor 在做坏点检测时需微调这两个值。
- 系统支持的坏点个数上限是 STATIC_DP_COUNT_MAX。当 sensor 坏点个数大于 STATIC_DP_COUNT_MAX，系统反馈坏点个数等于 STATIC_DP_COUNT_MAX。因此，坏点个数等于 STATIC_DP_COUNT_MAX 不能反映当前实际坏点情况。



- u8FinishThresh 只作为输出。针对同类型的 sensor，参考 u8FinishThresh 值，设置合理的 u8StartThresh，能加快静态坏点校正过程。
- 每次调用标定接口 HI_MPI_ISP_SetDPCalibrate 时，算法内部会自动将 enStatus 恢复为 ISP_STATE_INIT 状态。
- 标定亮坏点时，推荐降帧，亮坏点超时时间近似等于 u16TimeLimit/8 秒。标定暗坏点时，不做降帧动作，暗坏点超时时间近似等于 u16TimeLimit/48 秒。

【相关数据类型及接口】

- [ISP_STATIC_DP_TYPE_E](#)
- [ISP_STATUS_E](#)

ISP_DP_STATIC_ATTR_S

【说明】

定义静态坏点校正属性。

【定义】

```
typedef struct hiISP_DP_STATIC_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16BrightCount;
    HI_U16  u16DarkCount;
    HI_U32  au32BrightTable[STATIC_DP_COUNT_MAX];
    HI_U32  au32DarkTable[STATIC_DP_COUNT_MAX];
    HI_BOOL bShow;
}ISP_DP_STATIC_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	静态坏点校正使能 Hi3516EV200/Hi3516EV300 不支持静态坏点校正，静态坏点校正相关参数无效。	[0, 1] 0：禁止； 1：使能。	1
u16BrightCount	亮坏点个数 注意：当该变量作为输入时，表示亮坏点的个数；作为输出时，	[0, STATIC_DP_COUNT_MAX]	0



成员名称	描述	取值范围	默认值
	表示总的坏点个数		
u16DarkCount	暗坏点个数 注意：当该变量作为输入时，表示暗坏点的个数；作为输出时，无效值 0。	[0, STATIC_DP_COUNT_MAX]	0
au32BrightTable [STATIC_DP_COUNT_MAX]	亮坏点坐标信息，低 29bit 有效，[12:0]bit 为坏点水平坐标，[28:16]bit 为坏点垂直坐标。 注意：当该变量作为输入时，表示亮坏点的坐标值查找表；作为输出时，表示所有坏点的坐标值查找表。	[0, 0x1FFF1FFF]	0
au32DarkTable [STATIC_DP_COUNT_MAX]	暗的坏点坐标值，低 29bit 有效，[12:0]bit 为坏点水平坐标，[28:16]bit 为坏点垂直坐标。 注意：当该变量作为输入时，表示暗坏点的坐标值查找表；作为输出时，无效值。	[0, 0x1FFF1FFF]	0
bShow	静态坏点显示使能	[0, 1] 0：禁止； 1：使能。	0

【注意事项】

bEnableDetect、bShow 与 bEnable 是 3 个互斥的设置，并且对应的优先级是逐次降低的。

【相关数据类型及接口】

无。

ISP_DP_DYNAMIC_MANUAL_ATTR_S

【说明】



定义动态坏点校正的手动属性。

【定义】

```
typedef struct hiISP_DP_DYNAMIC_MANUAL_ATTR_S
{
    HI_U16    u16Strength;
    HI_U16    u16BlendRatio;
}ISP_DP_DYNAMIC_MANUAL_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
u16Strength	DPC 动态坏点处理强度，取值越大，DPC 处理强度越强，可以校正的动态坏点越多。	[0x0, 0xFF]	-
u16BlendRatio	坏点校正过程所需的融合比率	[0x0, 0x80]	-

【注意事项】

- u16Strength 的选择与 ISO 值有很大的关系，ISO 越高，对应的 u16Strength 应该越大，在图像中坏点校正较好时，图像中偏红严重，此时可以再调节 u16BlendRatio 改善图像质量。
- u16BlendRatio 值越大，图像越模糊。

【相关数据类型及接口】

[ISP_DP_DYNAMIC_ATTR_S](#)

ISP_DP_DYNAMIC_AUTO_ATTR_S

【说明】

定义动态坏点校正的自动属性。

【定义】

```
typedef struct hiISP_DP_DYNAMIC_AUTO_ATTR_S
{
    HI_U16    au16Strength[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16    au16BlendRatio[ISP_AUTO_ISO_STRENGTH_NUM];
}
```



```
}ISP_DP_DYNAMIC_AUTO_ATTR_S;
```

【成员】

成员名称	描述	取值范围
au16Strength[ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下的 DPC 的处理强度，该数组的 16 个值分别对应 sensor 在不同增益下的 DPC 的处理强度值，一般情况下，增益越大，配置的 DPC 处理强度值越大，对应关系如表 6-2 所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个增益值对应的处理强度。	[0, 0xFF]
au16BlendRatio[ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下的融合比率。其值越大，图像越偏绿。若连续两档的融合比率不相等，则通过线性插值的方法确定两档中间的某个增益值对应的融合比率。	[0, 0x80]

表6-2 au16Strength[16]在不同增益情况下对应的设置值

au16Strength	Again*Dgain*ISPDgain(times)
au16Strength[0]	1
au16Strength[1]	2
au16Strength[2]	4
au16Strength[3]	8
au16Strength[4]	16
au16Strength[5]	32
au16Strength[6]	64
au16Strength[7]	128
au16Strength[8]	256
au16Strength[9]	512
au16Strength[10]	1024
au16Strength[11]	2048



au16Strength	Again*Dgain*ISPDgain(times)
au16Strength[12]	4096
au16Strength[13]	8192
au16Strength[14]	16384
au16Strength[15]	32768

【注意事项】

- 该结构体中的默认值均在 sensor_cmos.c 中，如果用户需要修改默认值，请修改相应参数。如果用户需要对接新的 sensor，请参考已经提供的其它 sensor 的默认值
- 在 WDR 模式与线性模式之下，Again*Dgain*ISPDgain 没有差异

【相关数据类型及接口】

[ISP_DP_DYNAMIC_ATTR_S](#)

ISP_DP_DYNAMIC_ATTR_S

【说明】

定义动态坏点校正属性。

【定义】

```
typedef struct hiISP_DP_DYNAMIC_ATTR_S
{
    HI_BOOL  bEnable;
    HI_BOOL   bSupTwinkleEn;
    HI_S8     s8SoftThr;
    HI_U8     u8SoftSlope;
    ISP\_OP\_TYPE\_E enOpType;
    ISP\_DP\_DYNAMIC\_MANUAL\_ATTR\_S stManual;
    ISP\_DP\_DYNAMIC\_AUTO\_ATTR\_S stAuto;
} ISP_DP_DYNAMIC_ATTR_S;
```

【成员】



成员名称	描述	取值范围	默认值
bEnable	动态坏点校正功能使能	[0, 1]	1
enOpType	DPC 动态坏点校正工作模式	[0, 1] 0: 自动; 1: 手动	0
stManual	手动模式下配置动态坏点校正参数, 请参见 ISP_DP_DYNAMIC_MANUAL_ATTR_S	-	-
stAuto	自动模式下配置动态坏点校正参数, 请参见 ISP_DP_DYNAMIC_AUTO_ATTR_S	-	-
bSupTwinkleEn	闪烁抑制使能	[0,1] 0: 禁止; 1: 使能	0
s8SoftThr	闪烁抑制的阈值, 值越小, 所能处理的像素点越多, 图像会越模糊	[0,127]	6
u8SoftSlope	闪烁抑制模块中, 控制偏差小于 s8SoftThr 的像素点的校正情况。值越大, 参与校正处理的像素点越多, 图像会越模糊	[0,255]	21

【注意事项】

- 闪烁抑制功能与 u16Strength 存在联动关系, s8SoftThr 和 u8SoftSlope 取值不合理时, 调节 u16Strength, 有可能在 u16Strength 的某些档位之间存在校正效果的突变, 因此建议在保证 s8SoftThr 和 u8SoftSlope 的配置值不明显影响图像的清晰度的前提下, 调节 u16Strength, 看坏点的校正以及闪烁抑制的效果。
- u8SoftSlope 参数设置过大, 图像会模糊, 同时可能会出现伪彩等副作用, 因此 u8SoftSlope 应设置为不明显影响图像清晰度的参数。

【相关数据类型及接口】

- [ISP_OP_TYPE_E](#)
- [ISP_DP_DYNAMIC_MANUAL_ATTR_S](#)
- [ISP_DP_DYNAMIC_AUTO_ATTR_S](#)

6.8 Crosstalk Removal

6.8.1 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，根本原因就是临近像素值之间 Gr 和 Gb 值域不一致。

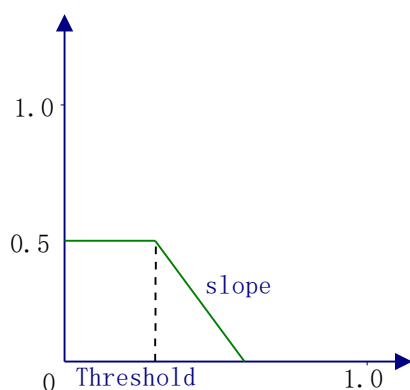
6.8.2 功能描述

Crosstalk 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，形成这些 pattern 的根本原因就是临近像素值之间 Gr 和 Gb 值不一致。

如图 6-9 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。图中的 Threshold, slope 与参数中的 au16Threshold, u8Slope 不完全对应，只是算法的大致示意。

- 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，Threshold 值越大，图像整体被处理的强度越大；
- 当 Gr 与 Gb 之间的差值大于 Threshold 值时，由 slope 控制处理的强度，当 slope 值越大，整体处理强度衰减得越慢，也就是处理的强度越大。

图6-9 Crosstalk 门限





6.8.3 API 参考

- [HI_MPI_ISP_SetCrosstalkAttr](#): 设定 Crosstalk 属性。
- [HI_MPI_ISP_GetCrosstalkAttr](#): 获取 Crosstalk 属性。

HI_MPI_ISP_SetCrosstalkAttr

【描述】

设定 Crosstalk 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetCrosstalkAttr(VI_PIPE ViPipe, const ISP\_CR\_ATTR\_S *pstCRAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstCRAAttr	Crosstalk 属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_GetCrosstalkAttr](#)

HI_MPI_ISP_GetCrosstalkAttr

【描述】

获取 Crosstalk 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetCrosstalkAttr(VI_PIPE ViPipe, ISP\_CR\_ATTR\_S *pstCRAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstCRAAttr	Crosstalk 属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetCrosstalkAttr

6.8.4 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    u8Slope;
    HI_U8    u8SensiSlope;
    HI_U16   u16SensiThr;
    HI_U16   au16Threshold[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16Strength[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16NpOffset[ISP_AUTO_ISO_STRENGTH_NUM];
}ISP_CR_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能 Crosstalk 功能。
u8Slope	设置 Crosstalk 门限之上的处理强度。值越大，表示在门限值 u16Threshold 之上整体处理强度衰减得越慢。 默认值 9。 Hi3516EV200/Hi3516EV300 取值范围[0,16]，其它芯片取值范围：[0, 14]。
u8SensiSlope	设置 Crosstalk 敏感度值。默认值 9。值越大，表示在门限值 u16SensiThr 之上，边沿上的处理强度衰减得越慢。 Hi3516EV200/Hi3516EV300 取值范围[0,16]，其它芯片取值范围：[0, 14]。
u16SensiThr	设置 Crosstalk 敏感度门限值。值越大，表示在更强的边



成员名称	描述
	沿上也会有绿平衡处理。默认值 300。 Hi3516EV200/Hi3516EV300 取值范围[0, 0xFFFF]，其它芯片取值范围：[0, 0x3FFF]。
au16Threshold[ISP_AUTO_ISO_STRENGTH_NUM]	设置 Crosstalk 门限值。值越大，表示整体处理的强度越大。 默认值 300。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。 Hi3516EV200/Hi3516EV300 取值范围[0, 0xFFFF]，其它芯片取值范围：[0, 0x3FFF]。
au16Strength[ISP_AUTO_ISO_STRENGTH_NUM]	设置 Crosstalk 强度值。值越大，处理强度越大。取值范围：[0, 0x100]。默认值 128。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。
au16NpOffset[ISP_AUTO_ISO_STRENGTH_NUM]	设置 Noise Profile 值。值越大，表示抗噪能力越大。默认值 1024。该数组的 16 个值分别对应的 sensor 在不同的增益情况下不同的设置值。 Hi3516EV200/Hi3516EV300 取值范围[0x2000, 0xFFFF]，其它芯片取值范围：[0x200, 0x3FFF]。

表6-3 u16Strength [ISP_AUTO_STRENGTH_NUM]在不同的增益情况下的设置值

u16Strength	Again*Dgain*ISPDgain (times)
u16Strength [0]	100
u16Strength [1]	200
u16Strength [2]	400
u16Strength [3]	800
u16Strength [4]	1600
u16Strength [5]	3200
u16Strength [6]	6400
u16Strength [7]	12800
u16Strength [8]	25600



u16Strength	Again*Dgain*ISPDgain (times)
u16Strength [9]	51200
u16Strength [10]	102400
u16Strength [11]	204800
u16Strength [12]	409600
u16Strength [13]	819200
u16Strength [14]	1638400
u16Strength [15]	3276800

【注意事项】

- u16SensiThr 参数变化会影响 au16Threshold 参数的表现，建议先调节 u16SensiThr 使之固定，再调节 au16Threshold。
- 各参数请在默认值附近微调，调试不当可能会在边缘或细节处出现伪彩，或使其变模糊。

【相关数据类型及接口】

无。

6.9 去噪算法

6.9.1 功能描述

去噪模块在 Bayer 域进行空域处理，分别对亮度噪声和色度噪声做去噪处理，且四个通道强度分别可调，具有保留边缘和纹理的优点，并对随机噪声有一定的保留，提高整体图像信噪比。

6.9.2 API 参考

- [HI_MPI_ISP_SetNRAttr](#)：设置 NR 参数。
- [HI_MPI_ISP_GetNRAttr](#)：获取 NR 参数。

HI_MPI_ISP_SetNRAttr

【描述】



设置 NR 参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetNRAttr(VI_PIPE ViPipe, const ISP\_NR\_ATTR\_S *pstNRAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstNRAttr	噪声抑制参数。	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_GetNRAttr](#)

HI_MPI_ISP_GetNRAttr

【描述】

获取噪声抑制参数。



【语法】

```
HI_S32 HI_MPI_ISP_GetNRAttr(VI_PIPE ViPipe, ISP\_NR\_ATTR\_S *pstNRAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstNRAttr	噪声抑制参数。	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_SetNRAttr](#)

6.9.3 数据类型

- [ISP_NR_MANUAL_ATTR_S](#)：定义 ISP NR 手动属性。
- [ISP_NR_AUTO_ATTR_S](#)：定义 ISP NR 自动属性。
- [ISP_NR_WDR_ATTR_S](#)：定义 ISP 在 WDR 模式下属性。



- [ISP_NR_ATTR_S](#): 定义 ISP NR 属性。

ISP_NR_MANUAL_ATTR_S

【说明】

定义 ISP NR 手动属性。

【定义】

```
typedef struct hiISP_NR_MANUAL_ATTR_S
{
    HI_U8    au8ChromaStr[ISP\_BAYER\_CHN\_NUM];
    HI_U8    u8FineStr;
    HI_U16   u16CoringWgt;
    HI_U16   au16CoarseStr[ISP\_BAYER\_CHN\_NUM];
} ISP_NR_MANUAL_ATTR_S;
```

【成员】

成员名称	描述
au8ChromaStr	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0, 0x3]。 Hi3516EV200/Hi3516EV300 不支持。
u8FineStr	亮噪去除强度，值越大，亮噪去除强度越大。 范围：[0x0,0x80]。
u16CoringWgt	随机噪声保留强度，值越大，保留的随机噪声越多。 范围：[0x0,0xC80]。
au16CoarseStr	控制亮度噪声整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0x0, 0x360]。

【注意事项】

- 注意各参数的取值范围。



- 低照度下，参数 au16CoarseStr 与 u8FineStr 同时调到最大值时，容易出现由于去噪强度过大导致的边缘虚化，建议根据效果调到合适取值。

【相关数据类型及接口】

ISP_NR_ATTR_S

ISP_NR_AUTO_ATTR_S

【说明】

定义 ISP NR 自动属性。

【定义】

```
typedef struct hiISP_NR_AUTO_ATTR_S
{
    HI_U8    au8ChromaStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8FineStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16CoringWgt[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16CoarseStr[ISP_BAYER_CHN_NUM][ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_NR_AUTO_ATTR_S;
```

【成员】

成员名称	描述
au8ChromaStr	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0x0, 0x3]。 Hi3516EV200/Hi3516EV300 不支持。
au8FineStr	亮噪去除强度，值越大，亮噪去除强度越大。 范围：[0x0,0x80]。
au16CoringWgt	随机噪声保留强度，值越大，保留的随机噪声越多。 范围：[0x0,0xC80]。
au16CoarseStr	控制亮度噪声整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0x0, 0x360]。



【注意事项】

- 注意各参数的取值范围。
- 低照度下，参数 au16CoarseStr 与 au8FineStr 同时调到最大值时，容易出现由于去噪强度过大导致的边缘虚化，建议根据效果调到合适取值。

【相关数据类型及接口】

[ISP_NR_ATTR_S](#)

ISP_NR_WDR_ATTR_S

【说明】

定义 ISP NR 在 WDR 模式下属性，与自动/手动模式并行。

【定义】

```
typedef struct hiISP_NR_WDR_ATTR_S
{
    HI_U8    au8WDRFrameStr[WDR_MAX_FRAME_NUM];
    HI_U8    au8FusionFrameStr[WDR_MAX_FRAME_NUM];
}ISP_NR_WDR_ATTR_S;
```



说明

宏定义如下：

```
#define WDR_MAX_FRAME_NUM      (4)
```

【成员】

成员名称	描述
au8WDRFrameStr	WDR 模式下每一合成帧分别对应的全局亮度去噪强度。值越大，去噪强度越大，值越小，去噪强度越弱。 取值范围：[0x0, 0x50]。
au8FusionFrameStr	Fusion 模式下每一融合帧分别对应的全局亮度去噪强度。 取值范围：[0x0, 0x50]。 Hi3559AV100/Hi3519AV100 不支持。

【注意事项】

注意各参数的取值范围。



【相关数据类型及接口】

ISP_NR_ATTR_S

ISP_NR_ATTR_S

【说明】

定义 ISP NR 属性。

【定义】

```
typedef struct hiISP_NR_ATTR_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bLowPowerEnable;
    HI_BOOL  bNrLscEnable;
    HI_U8     u8NrLscRatio;
    HI_U8     u8BnrLscMaxGain;
    HI_U16    u16BnrLscCmpStrength;
    HI_U16    au16CoringRatio[HI_ISP_BAYERNR_LUT_LENGTH];
    ISP_OP_TYPE_E  enOpType;
    ISP_NR_AUTO_ATTR_S  stAuto;
    ISP_NR_MANUAL_ATTR_S stManual;
    ISP_NR_WDR_ATTR_S  stWdr;
}ISP_NR_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	NR 模块使能	[0, 1] 0：禁止； 1：使能。	0x1
bLowPowerEnable	NR 低功耗模式开关。 Hi3516CV500/ Hi3559V200/Hi3516EV200/ Hi3516EV300 不支持	[0, 1] 0：禁止； 1：使能。	0x0
bNrLscEnable	NR 参考 LSC 增益功能使能。	[0, 1] 0：禁止；	0x0



成员名称	描述	取值范围	默认值
		1: 使能。	
u8NrLscRatio	NR 参考 LSC 增益比例。 Hi3516CV500/Hi3559V200/ Hi3516EV200/Hi3516EV300 不支持	[0x0, 0xFF]	0x0
u8BnrLscMaxGain	参考 LSC 增益最大倍数。 Hi3559AV100/Hi3519AV100 不支持	[0x0, 0xBF]	0x60
u16BnrLscCmpStrength	参考 LSC 增益强度。 Hi3559AV100/Hi3519AV100 不支持	[0x0, 0x100]	0x100
au16CoringRatio	噪声保留强度, 根据亮度在 au16CoringRatio[33]数组里 查找对应的噪声保留强度, 值 越大, 亮度噪声保留越多, 值 越小, 亮度噪声保留越少。	[0x0, 0x3FF]	-
enOpType	NR 工作类型。 OP_TYPE_AUTO: 自动; OP_TYPE_MANUAL: 手动。	-	OP_TYPE_AUTO
stAuto	NR 自动参数。	-	-
stManul	NR 手动参数。	-	-
stWdr	NR WDR 模式下参数。	-	-

【注意事项】

NR 功能分为自动和手动参数:

- bEnable 为 HI_TRUE, enOpType 为 OP_TYPE_AUTO, 使用自动 NR 功能。
- bEnable 设置为 HI_TRUE, enOpType 设置为 OP_TYPE_MANUAL 使用手动 NR 功能。



- 当工作在 WDR 模式下，NR 整体去噪强度参数 stWdr 生效。
- 低功耗模式下，去噪效果有所下降，下降程度与 NR 使用强度关联。
- BNR 参考 LSC 去噪功能需要在 LSC 模块打开下调试，当 LSC 模块本身关闭时，BNR 参考 LSC 不生效。

【相关数据类型及接口】

无。

6.10 2 轴 DIS

6.10.1 概述

DIS 模块通过比较当前图像与前一帧图像来计算出当前图像抖动位移大小，并传递到后级模块，进行输出图像范围的调整，从而起到防抖的效果。每帧图像经过 DIS 后得到一组 CROP 的起点 (X, Y)，再经过后端 VPSS CROP 功能处理，达到防抖效果。

Hi3516CV500/Hi3559V200 不支持此功能。

其他防抖模式实现请参见《DIS 调试指南》。

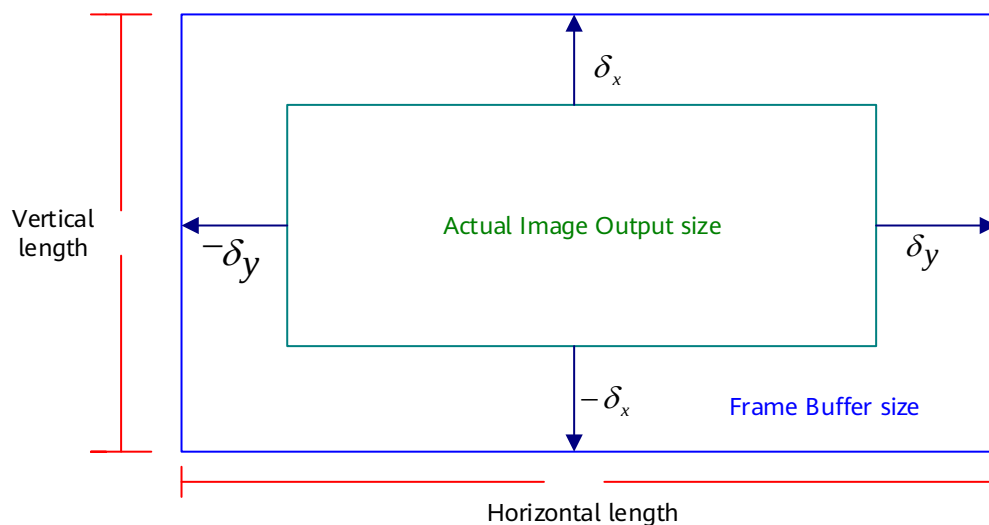
6.10.2 功能描述

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：

- Hi3559AV100/Hi3519AV100：[-128, +128]
- Hi3516EV200/Hi3516EV300：[-64,+64]

DIS 偏移示意图如图 6-10 所示。

图6-10 DIS 偏移示意图



如图 6-10 所示，VPSS 裁剪后输出的图像大小为内部的长方形，ISP 的 frame buffer 存储的图像大小为外部长方形，当图像抖动后，DIS 模块输出水平方向和垂直方向的偏移量 δ_x , δ_y ，VPSS 根据这两个偏移量的进行裁剪后输出稳定的图像。

Hi3559AV100/Hi3519AV100 2 轴 DIS 功能只能 VI 在线，VPSS 离线模式下使用。

Hi3516EV200/Hi3516EV300 2 轴 DIS 功能只能 VPSS 离线模式下使用。

6.10.3 API 参考

- [HI_MPI_ISP_SetDISAttr](#): 设置 2 轴 DIS 属性。
- [HI_MPI_ISP_GetDISAttr](#): 获取 2 轴 DIS 属性。

HI_MPI_ISP_SetDISAttr

【描述】

设置 2 轴 DIS 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetDISAttr(VI_PIPE ViPipe, const ISP\_DIS\_ATTR\_S *pstDISAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe。	输入
pstDISAttr	2 轴 DIS 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】

- Hi3559AV100/Hi3519AV100 2 轴 DIS 功能只能 VI 在线，VPSS 离线模式下使用。
- Hi3516EV200/Hi3516EV300 2 轴 DIS 功能只能 VPSS 离线模式下使用。

【举例】

```
.....  
  
/* SET & ENABLE  VI  DEV */  
stChnAttr.stCapRect.s32X = 0;  
stChnAttr.stCapRect.s32Y = 0;  
stChnAttr.stCapRect.u32Width = 1280+100;  
stChnAttr.stCapRect.u32Height = 720+100;  
stChnAttr.stDestSize.u32Width = stChnAttr.stCapRect.u32Width;  
stChnAttr.stDestSize.u32Height = stChnAttr.stCapRect.u32Height;  
s32Ret  = HI_MPI_VI_SetChnAttr(ViChn, &stChnAttr);  
if (s32Ret != HI_SUCCESS)  
{  
    SAMPLE_PRT("HI_MPI_VI_SetChnAttr failed with %#x\n", s32Ret);  
    return HI_FAILURE;  
}
```




```
}
s32Ret = HI_MPI_VI_EnableChn(ViChn);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VI_EnableChn failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VPSS_CreateGrp(VpssGrp, &stVpssGrpAttr);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_CreateGrp failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VPSS_StartGrp(VpssGrp);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_StartGrp failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
VPSS_CROP_INFO_S stVpssCropInfo;

stVpssCropInfo.bEnable = HI_TRUE;
stVpssCropInfo.enCropCoordinate = VPSS_CROP_ABS_COOR;
stVpssCropInfo.stCropRect.s32X = 50;
stVpssCropInfo.stCropRect.s32Y = 50;
stVpssCropInfo.stCropRect.u32Width = 1280;
stVpssCropInfo.stCropRect.u32Height = 720;
s32Ret = HI_MPI_VPSS_SetGrpCrop(VpssGrp, &stVpssCropInfo);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_SetGrpCrop failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
s32Ret = HI_MPI_VPSS_SetChnAttr(VpssGrp, VpssChn, &stVpssChnAttr);
if (s32Ret != HI_SUCCESS)
{
    SAMPLE_PRT("HI_MPI_VPSS_SetChnAttr failed with %#x\n", s32Ret);
    return HI_FAILURE;
}
```



```
}  
s32Ret = HI_MPI_VPSS_EnableChn(VpssGrp, VpssChn);  
if (s32Ret != HI_SUCCESS)  
{  
    SAMPLE_PRT("HI_MPI_VPSS_EnableChn failed with %#x\n", s32Ret);  
    return HI_FAILURE;  
}  
ISP_DIS_ATTR_S stDisAttr;  
stDisAttr.bEnable = HI_TRUE;  
stDisAttr.u32ConvergeRate = 16;  
int ViPipe = 0;  
s32Ret = HI_MPI_ISP_SetDISAttr(ViPipe, &stDisAttr);  
if (s32Ret != HI_SUCCESS)  
{  
    SAMPLE_PRT("HI_MPI_ISP_SetDISAttr failed with %#x\n", s32Ret);  
    return HI_FAILURE;  
}  
.....
```

【相关主题】

无

HI_MPI_ISP_GetDISAttr

【描述】

获取 2 轴 DIS 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDISAttr(VI_PIPE ViPipe, ISP\_DIS\_ATTR\_S *pstDISAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDISAttr	2 轴 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 。

【需求】

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

【注意】

- Hi3559AV100/Hi3519AV100 2 轴 DIS 功能只能 VI 在线，VPSS 离线模式下使用。
- Hi3516EV200/Hi3516EV300 2 轴 DIS 功能只能 VPSS 离线模式下使用。

【举例】

无

【相关主题】

无

6.10.4 数据类型

[ISP_DIS_ATTR_S](#)：定义 ISP 2 轴 DIS 属性。

ISP_DIS_ATTR_S

【说明】

定义 ISP 2 轴 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 2 轴 DIS 功能。

【注意事项】

无

【相关数据类型及接口】

无

6.11 Dehaze

6.11.1 功能描述

Dehaze 是通过动态的改变图象的对比度和亮度来实现的，将图像分块，统计每块内的像素值，估算出雾的浓度，根据局部自适应曲线调整去雾强度。Dehaze 分为手动和自动模式，两种模式下均可调节去雾强度。

6.11.2 API 参考

- [HI_MPI_ISP_SetDehazeAttr](#)：设置去雾属性。
- [HI_MPI_ISP_GetDehazeAttr](#)：获取去雾属性。

HI_MPI_ISP_SetDehazeAttr

【描述】

设置 Dehaze 属性

【语法】

```
HI_S32 HI_MPI_ISP_SetDehazeAttr(VI_PIPE ViPipe, const ISP_DEHAZE_ATTR_S *pstDehazeAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入



pstDehazeAttr	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 。

【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_GetDehazeAttr](#)

HI_MPI_ISP_GetDehazeAttr

【描述】

获取 Dehaze 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDehazeAttr(VI_PIPE ViPipe, ISP\_DEHAZE\_ATTR\_S *pstDehazeAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDehazeAttr	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 。

【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_SetDehazeAttr](#)

6.11.3 数据类型

- [ISP_DEHAZE_MANUAL_ATTR_S](#)：定义去雾手动模式属性。
- [ISP_DEHAZE_AUTO_ATTR_S](#)：定义去雾自动模式属性。
- [ISP_DEHAZE_ATTR_S](#)：定义 ISP 去雾属性。

ISP_DEHAZE_MANUAL_ATTR_S

【说明】

定义去雾手动模式属性。

【定义】

```
typedef struct hiISP_DEHAZE_MANUAL_ATTR_S
{
    HI_U8          u8strength;
}ISP_DEHAZE_MANUAL_ATTR_S;
```



【成员】

成员名称	描述
u8strength	手动模式下去雾强度，取值范围：[0,255]

【注意事项】

无

【相关数据类型及接口】

无

ISP_DEHAZE_AUTO_ATTR_S

【说明】

定义去雾自动模式属性。

【定义】

```
typedef struct hiISP_DEHAZE_AUTO_ATTR_S
{
    HI_U8          u8strength;
}ISP_DEHAZE_AUTO_ATTR_S;
```

【成员】

成员名称	描述
u8strength	自动模式下去雾强度的权重系数，每个块自动计算出的去雾强度会乘以权重系数，得到的结果作为最终的去雾强度，取值范围：[0,255]，默认值为 128，用户可以手动修改强度值。

【注意事项】

无

【相关数据类型及接口】

无



ISP_DEHAZE_ATTR_S

【说明】

定义 ISP 去雾属性。

【定义】

```
typedef struct hiISP_DEHAZE_ATTR_S
{
    HI_BOOL  bEnable;
    HI_BOOL  bUserLutEnable;
    HI_U8    au8DehazeLut[DEHAZE_LUT_SIZE];
    ISP_OP_TYPE_E    enOpType;
    ISP_DEHAZE_MANUAL_ATTR_S stManual;
    ISP_DEHAZE_AUTO_ATTR_S    stAuto;
    HI_U16 u16TmprfltIncrCoef;
    HI_U16 u16TmprfltDecrCoef;
}ISP_DEHAZE_ATTR_S;
```



说明

宏定义如下:

```
#define DEHAZE_LUT_SIZE    256
```

【成员】

成员名称	描述
bEnable	使能 Dehaze 功能。
bUserLutEnable	用户自定义去雾曲线使能
au8DehazeLut[DEHAZE_LUT_SIZE]	用户自定义去雾曲线
enOpType	操作模式。
stManual	手动模式。
stAuto	自动模式。
u16TmprfltIncrCoef	时域滤波递增系数，值越大，画面由暗到亮收敛速度越快，取值范围[0x0,0x80]
u16TmprfltDecrCoef	时域滤波递减系数，值越大，画面由亮到暗收敛速度越快，取值范围[0x0,0x80]



【注意事项】

- 去雾使能后，为了避免画面异常，延迟两帧生效。
- **注意 Hi3559AV100 和 Hi3519AV100 在使用 VI parallel mode 或者 striping 模式时，去雾强度过大可能导致在中心交界处出现分界线**，该现象与 au8DehazeLut[256]/ u8strength 参数配置有关，推荐该模式下使用默认参数，去雾强度不要超过 200。
- 当 u16TmprfltIncrCoef 或者 u16TmprfltDecrCoef 为 0 时，不做时域滤波。

【相关数据类型及接口】

无

6.12 去伪彩

6.12.1 功能描述

高频分量在图像插值时易引起高频混叠。用镜头对准一个分辨率测试卡，当 sensor 表面没有 OLPF 时，在分辨率的高频部分容易出现伪彩。

去除伪彩主要是指去除高频部分因插值错误所导致的伪彩现象。增大去伪彩强度值会减弱伪彩现象，但可能导致正常的颜色灰度化。

该模块原理是计算出伪彩出现区域及范围，通过可配去伪彩强度寄存器值调整去伪彩强度值，将 RGB 三个通道值灰度化，从而达到减弱伪彩现象的目的。

6.12.2 API 参考

- [HI_MPI_ISP_SetAntiFalseColorAttr](#)：设置 AntiFalseColor 参数。
- [HI_MPI_ISP_GetAntiFalseColorAttr](#)：获取 AntiFalseColor 参数。

HI_MPI_ISP_SetAntiFalseColorAttr

【描述】

设置 AntiFalseColor 参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetAntiFalseColorAttr (VI_PIPE ViPipe, const ISP\_ANTIFALSECOLOR\_ATTR\_S
```



```
*pstAntiFalseColor);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstAntiFalseColor	AntiFalseColor 参数	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_GetAntiFalseColorAttr](#)

HI_MPI_ISP_GetAntiFalseColorAttr

【描述】

获取 AntiFalseColor 参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetAntiFalseColorAttr (VI_PIPE ViPipe, ISP\_ANTIFALSECOLOR\_ATTR\_S  
*pstAntiFalseColor);
```



【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstAntiFalseColor	AntiFalseColor 参数	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_SetAntiFalseColorAttr](#)

6.12.3 数据类型

- [ISP_ANTIFALSECOLOR_MANUAL_ATTR_S](#)：定义 ISP ANTIFALSECOLOR 手动属性
- [ISP_ANTIFALSECOLOR_AUTO_ATTR_S](#)：定义 ISP ANTIFALSECOLOR 自动属性
- [ISP_ANTIFALSECOLOR_ATTR_S](#)：定义 ISP ANTIFALSECOLOR 属性



ISP_ANTIFALSECOLOR_MANUAL_ATTR_S

【说明】

定义 ISP ANTIFALSECOLOR 手动属性。

【定义】

```
typedef struct hiISP_ANTIFALSECOLOR_MANUAL_ATTR_S
{
    HI_U8    u8AntiFalseColorThreshold;
    HI_U8    u8AntiFalseColorStrength;
} ISP_ANTIFALSECOLOR_MANUAL_ATTR_S;
```

【成员】

成员名称	描述
u8AntiFalseColorThreshold	去伪彩阈值，值越大，伪彩去除的区域越多。 取值范围：[0, 0x20]。
u8AntiFalseColorStrength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变。 取值范围：[0, 0x1F]。

【注意事项】

注意各参数的取值范围。

【相关数据类型及接口】

[ISP_ANTIFALSECOLOR_ATTR_S](#)

ISP_ANTIFALSECOLOR_AUTO_ATTR_S

【说明】

定义 ISP ANTIFALSECOLOR 自动属性。

【定义】

```
typedef struct hiISP_ANTIFALSECOLOR_AUTO_ATTR_S
{
    HI_U8    au8AntiFalseColorThreshold[ISP_AUTO_ISO_STRENGTH_NUM];
}
```



```
HI_U8  au8AntiFalseColorStrength[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_ANTIFALSECOLOR_AUTO_ATTR_S;
```

【成员】

成员名称	描述
au8AntiFalseColorThreshold	去伪彩阈值，值越大，伪彩去除的区域越多。 取值范围：[0, 0x20]。
au8AntiFalseColorStrength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变。 取值范围：[0, 0x1F]。

【注意事项】

注意各参数的取值范围。

【相关数据类型及接口】

[ISP_ANTIFALSECOLOR_ATTR_S](#)

ISP_ANTIFALSECOLOR_ATTR_S

【说明】

定义 ISP ANTIFALSECOLOR 属性。

【定义】

```
typedef struct hiISP_ANTIFALSECOLOR_ATTR_S
{
    HI_BOOL  bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_ANTIFALSECOLOR_AUTO_ATTR_S stAuto;
    ISP_ANTIFALSECOLOR_MANUAL_ATTR_S stManual;
}ISP_ANTIFALSECOLOR_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	ANTIFALSECOLOR 模块使能。	[0,1]	0x1



成员名称	描述	取值范围	默认值
		0：禁止； 1：使能。	
enOpType	ANTIFALSECOLOR 工作类型。 OP_TYPE_AUTO：自动； OP_TYPE_MANUAL：手动。	-	OP_TYPE_AUTO
stAuto	ANTIFALSECOLOR 自动参数。	-	-
stManual	ANTIFALSECOLOR 手动参数。	-	-

【注意事项】

ANTIFALSECOLOR 功能分为自动和手动参数：

- bEnable 为 HI_TRUE，enOpType 为 OP_TYPE_AUTO，使用自动 ANTIFALSECOLOR 功能。
- bEnable 设置为 HI_TRUE，enOpType 设置为 OP_TYPE_MANUAL 使用手动 ANTIFALSECOLOR 功能。

【相关数据类型及接口】

无。

6.13 去马赛克

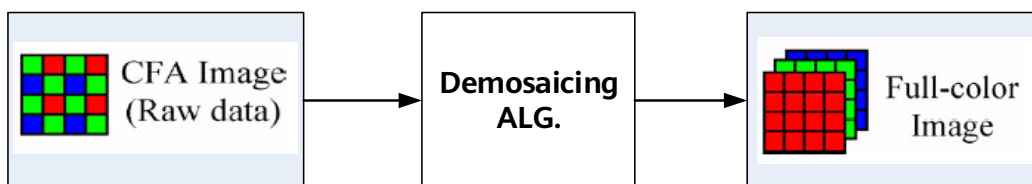
6.13.1 功能描述

Demosaic 模块实现的功能是：将输入的 Bayer 数据转化成 RGB 数据。CFA（Color Filter Array，彩色滤波阵列）得到的 Bayer 数据，每一像素点只能获得 R、G、B 三基色中的一种彩色分量值。为获得彩色图像，需要利用当前像素及周围像素的色彩分量值，估计出当前点缺失的其他两个分量值（如图 6-11 所示）。

该模块能输出较高的图像分辨率，对弱细节及清晰度有较大提升。



图6-11 去马赛克处理图



须知

在处理高频黑白图像时，由于去马赛克模块对输入 Bayer CFA 数据进行了插值，因此在高频区域容易发生混叠，导致出现打开去马赛克模块高频区域清晰度反而下降的情况，此为正常现象。

6.13.2 API 参考

- [HI_MPI_ISP_SetDemosaicAttr](#)：设置 Demosaic 参数。
- [HI_MPI_ISP_GetDemosaicAttr](#)：获取 Demosaic 参数。

HI_MPI_ISP_SetDemosaicAttr

【描述】

设置 Demosaic 参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetDemosaicAttr (VI_PIPE ViPipe, const ISP\_DEMOSAIC\_ATTR\_S  
*pstDemosaicAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDemosaicAttr	Demosaic 参数。	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_GetDemosaicAttr](#)

HI_MPI_ISP_GetDemosaicAttr

【描述】

获取 Demosaic 参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetDemosaicAttr (VI_PIPE ViPipe, ISP\_DEMOSAIC\_ATTR\_S *pstDemosaicAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDemosaicAttr	Demosaic 参数。	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_SetDemosaicAttr](#)

6.13.3 数据类型

- [ISP_DEMOSAIC_MANUAL_ATTR_S](#)：定义 ISP DEMOSAIC 手动属性。
- [ISP_DEMOSAIC_AUTO_ATTR_S](#)：定义 ISP DEMOSAIC 自动属性。
- [ISP_DEMOSAIC_ATTR_S](#)：定义 ISP DEMOSAIC 属性。

ISP_DEMOSAIC_MANUAL_ATTR_S

【说明】

定义 ISP DEMOSAIC 手动属性。

【定义】

```
typedef struct hiISP_DEMOSAIC_MANUAL_ATTR_S
{
    HI_U8    u8NonDirStr;
    HI_U8    u8NonDirMFDetailEhcStr;
    HI_U8    u8NonDirHFDdetailEhcStr;
    HI_U8    u8DetailSmoothRange;
```



```

HI_U16  u16DetailSmoothStr;
    HI_U8   u8ColorNoiseThdF;
    HI_U8   u8ColorNoiseStrF;
    HI_U8   u8ColorNoiseThdY;
HI_U8   u8ColorNoiseStrY;
} ISP_DEMOSAIC_MANUAL_ATTR_S;

```

【成员】

成员名称	描述
u8NonDirStr	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0,0xFF]
u8NonDirMFDetailEhcStr	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 - Hi3559AV100 取值范围：[0x0, 0x10] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0x7F]
u8NonDirHFDetailEhcStr	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0,0x10]
u8DetailSmoothRange	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 - Hi3559AV100 取值范围：[0x1, 0x8] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x1, 0x7]
u16DetailSmoothStr	细节平滑强度，值越大，平滑处理强度越大，对伪细节的抑制强度越大。 取值范围：[0x0, 0x100]。仅适用于 Hi3559AV100
u8ColorNoiseThdF	根据画面高频程度降低颜色噪声，值越大，则越容易在平坦区域进行去色噪功能。值越小，则影响的像素越少。取值范围为[0x0, 0xFF]，默认值为 0
u8ColorNoiseStrF	根据画面高频程度降低颜色噪声的强度，值越大，则降饱和度强度越高。取值范围为[0x0, 0x8]，默认值为 8。



成员名称	描述
	Hi3559AV100 不支持该接口
u8ColorNoiseThdY	根据亮度和饱和度降低颜色噪声，值越大，对暗处影响越大，对饱和度高的像素影响越大。值越小，影响的像素越少。 - Hi3559AV100/ Hi3519AV100/Hi3516CV500/ Hi3559V200 取值范围：[0x0, 0xFF] - Hi3516EV200/ Hi3516EV300 取值范围：[0x0, 0xF]
u8ColorNoiseStrY	根据亮度和饱和度降低颜色噪声的强度，值越大，对影响的像素降饱和度越多，值越小，降饱和度程度越少。 - Hi3559AV100/ Hi3519AV100/Hi3516CV500/ Hi3559V200 取值范围：[0x0, 0xFF] - Hi3516EV200/ Hi3516EV300 取值范围：[0x0, 0x3F]

【注意事项】

- 在红外模式下，或者在 CCM 饱和度设定成 0，输出灰度图的情况下，需要将 u8ColorNoiseThdY 和 u8ColorNoiseStrY 参数都设定成 0，关闭降低颜色噪声的功能。
- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200 中使用 u8ColorNoiseThdY 和 u8ColorNoiseStrY 参数，受降低颜色噪声的功能影响，会导致 CCM 饱和度设定成 0 时图像不会完全灰色，调节 CCM 饱和度时，图像饱和度不是线性变化。Hi3516EV200/ Hi3516EV300 不存在这类问题。
- u8ColorNoiseThdF 由于算法采取的是硬钳位方法进行判断是否需要降低饱和度，因此如果这个参数设置的过大，在有一定亮噪的情况下，会出现颜色过渡异常的情况。

【相关数据类型及接口】

[ISP_DEMOSAIC_ATTR_S](#)

ISP_DEMOSAIC_AUTO_ATTR_S

【说明】

定义 ISP DEMOSAIC 自动属性。



【定义】

```
typedef struct hiISP_DEMOSAIC_AUTO_ATTR_S
{
    HI_U8    au8NonDirStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8NonDirMFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8NonDirHFDetailEhcStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8DetailSmoothRange[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16   au16DetailSmoothStr[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8ColorNoiseThdF[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8ColorNoiseStrF[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8ColorNoiseThdY[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8    au8ColorNoiseStrY[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_DEMOSAIC_AUTO_ATTR_S;
```



说明

宏定义如下：

```
#define ISP_AUTO_ISO_STRENGTH_NUM    (16)
```

【成员】

成员名称	描述
au8NonDirStr	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0, 0xFF]
au8NonDirMFDetailEhcStr	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 - Hi3559AV100 取值范围：[0x0, 0x10] - Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0x7F]
au8NonDirHFDetailEhcStr	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0, 0x10]
au8DetailSmoothRange	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 - Hi3559AV100 取值范围：[0x1, 0x8] - Hi3519AV100/Hi3516CV500/



成员名称	描述
	Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x1, 0x7]
au16DetailSmoothStr	细节平滑强度，值越大，平滑处理强度越大，对伪细节的抑制强度越大。 取值范围：[0x0, 0x100]。仅适用于 Hi3559AV100
au8ColorNoiseThdF	根据画面高频程度降低颜色噪声，值越大，则越容易在平坦区域进行去噪功能。值越小，则影响的像素越少。取值范围为[0x0, 0xFF]，默认值为 0
au8ColorNoiseStrF	根据画面高频程度降低颜色噪声的强度，值越大，则降饱和度强度越高。取值范围为[0x0, 0x8]，默认值为 8。 Hi3559AV100 不支持该接口
au8ColorNoiseThdY	根据亮度和饱和度降低颜色噪声，值越大，对暗处影响越大，对饱和度高的像素影响越大。值越小，影响的像素越少。 - Hi3559AV100/ Hi3519AV100/Hi3516CV500/ Hi3559V200 取值范围：[0x0, 0xFF] - Hi3516EV200/ Hi3516EV300 取值范围：[0x0, 0xF]
au8ColorNoiseStrY	根据亮度和饱和度降低颜色噪声的强度，值越大，对影响的像素降饱和度越多，值越小，降饱和度程度越少。 - Hi3559AV100/ Hi3519AV100/Hi3516CV500/ Hi3559V200 取值范围：[0x0, 0xFF] - Hi3516EV200/ Hi3516EV300 取值范围：[0x0, 0x3F]

【注意事项】

注意各参数的取值范围。

【相关数据类型及接口】

[ISP_DEMOSAIC_ATTR_S](#)

ISP_DEMOSAIC_ATTR_S

【说明】



定义 ISP DEMOSAIC 属性。

【定义】

```
typedef struct hiISP_DEMOSAIC_ATTR_S
{
    HI_BOOL  bEnable;
    ISP_OP_TYPE_E enOpType;
    ISP_DEMOSAIC_AUTO_ATTR_S stAuto;
    ISP_DEMOSAIC_MANUAL_ATTR_S stManual;
}ISP_DEMOSAIC_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	Demosaic 模块使能	[0, 1] 0: 禁止; 1: 使能	0x1
enOpType	DEMOSAIC 工作类型。 - OP_TYPE_AUTO: 自动; - OP_TYPE_MANUAL: 手动。	-	OP_TYPE_AUTO
stAuto	DEMOSAIC 自动参数。	-	-
stManual	DEMOSAIC 手动参数。	-	-

【注意事项】

DEMOSAIC 功能分为自动和手动参数：

- bEnable 为 HI_TRUE, enOpType 为 OP_TYPE_AUTO, 使用自动 DEMOSAIC 功能。
- bEnable 设置为 HI_TRUE, enOpType 设置为 OP_TYPE_MANUAL 使用手动 DEMOSAIC 功能。

【相关数据类型及接口】

无。



6.14 黑电平

6.14.1 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。

6.14.2 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(VI_PIPE ViPipe, const ISP\_BLACK\_LEVEL\_S *pstBlackLevel);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstBlackLevel	黑电平属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_comm_isp.h、mpi_isp.h



- 库文件：libisp.a

【注意】

- 如果接口配置为自动模式：若 cmos.c 里面的 `ISP_CMOS_BLACK_LEVEL_S` 结构体成员 `bUpdate` 设置为 `HI_TRUE`，则表示始终使用 cmos.c 内联动的黑电平配置，而 `bUpdate` 设置为 `HI_FALSE`，则表示使用 cmos.c 非联动的黑电平配置；如果接口配置为手动模式：此时接口配置的黑电平配置会生效，ISP 内部采用接口配置值。
- 线性模式下，black level 模块集成在 isp dgain 模块中，所以 bypass isp dgain 会同时关闭 isp dgain 模块中的 black level 功能。对图像质量影响较大。
- WDR 模式下，black level 模块集成在 isp dgain 及 WDR 模块中，bypass isp dgain 也会对图像质量有部分影响。

【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(VI_PIPE ViPipe, ISP_BLACK_LEVEL_S *pstBlackLevel);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstBlackLevel	黑电平属性。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 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

6.14.3 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
{
    ISP\_OP\_TYPE\_E enOpType;
    HI_U16 au16BlackLevel[ISP\_BAYER\_CHN\_NUM];
} ISP_BLACK_LEVEL_S;
```

【成员】



成员名称	描述
enOpType	黑电平工作类型。 - OP_TYPE_AUTO：自动； - OP_TYPE_MANUAL：手动。 默认：OP_TYPE_AUTO 模式。
au16BlackLevel[ISP_BAYER_CHN_NUM]	黑电平的值，分别表示 R、Gr、Gb、B 分量的黑电平 取值范围：[0x0, 0xFF]F 默认值由文件 cmos.c 内结构体 ISP_CMOS_BLACK_LEVEL_S 的成员 au16BlackLevel[ISP_BAYER_CHN_NUM]设定。

【注意事项】

无

【相关数据类型及接口】

无

6.15 去 FPN

6.15.1 功能描述

Sensor 将光信号转换成电信号，再通过数百万个 ADC 器件后输出图像。每个像素结构中的光电二极管的尺寸、掺杂浓度、生产过程中的沾污以及 MOS 场效应管的参数的偏差等都会造成像素输出信号的变化，由于这些偏差造成的噪声对于给定的单个像素它是固定的，这种噪声就是固定模式噪声 FPN（Fixed Pattern Noise）。

去 FPN 模块就是要把这些固定模式噪声消除。

FPN 的标定过程：

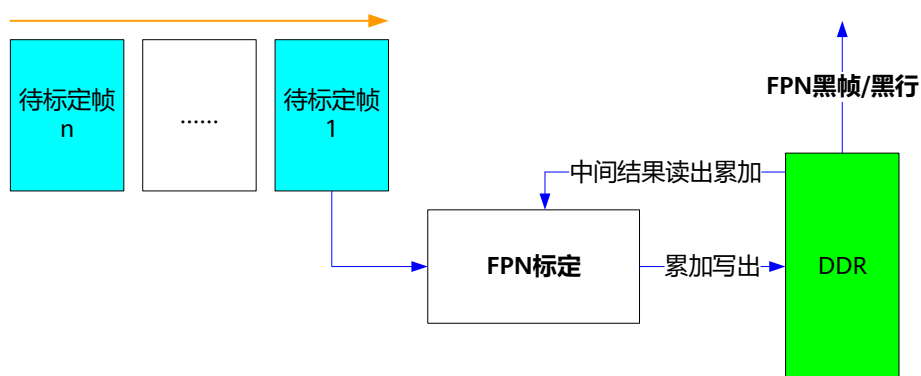
- 步骤 1 首先确定 FPN 在哪个 ISO 下最为严重，在此 ISO 下固定增益，关闭镜头的光圈，让 sensor 采集黑帧。

步骤 2 启动标定，当第一帧过来时，把满足标定条件的 RAW 数据写到内存，当第二帧过来时，通过读出端口把内存中的黑帧读出来与第二帧进行累加后再通过写出端口再写回内存，后面的帧也是这样，但是最后一帧写出时会对累加的黑帧平均后再写回内存。

步骤 3 当求平均写出后，标定结束。此时，内存中就把低照度下的固定列噪声存储下来了，然后，把该黑帧存储到外存上即可。

----结束

图6-12 FPN 标定示意图



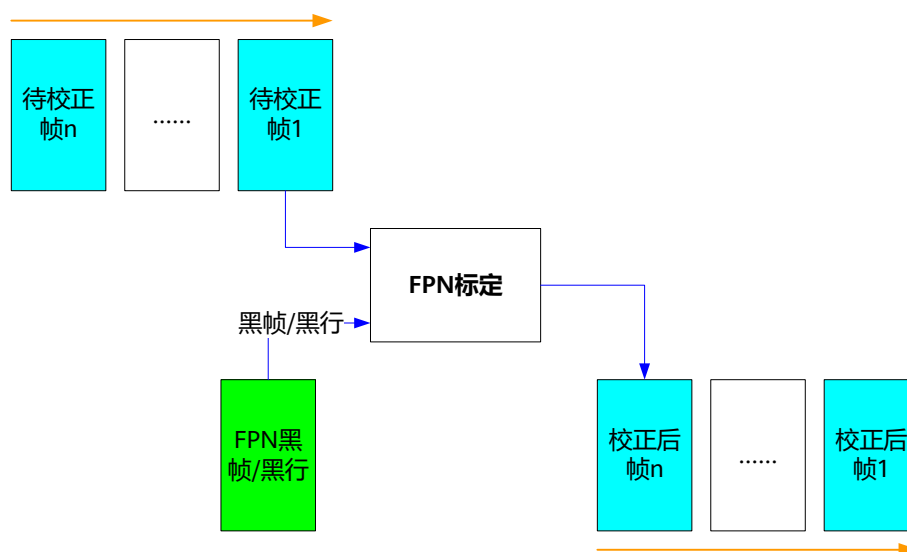
FPN 的校正过程：

步骤 1 把标定好的黑帧从外装载入到内存中。

步骤 2 当从 sensor 过采集过来的带 FPN 的正常图像过来后，用每帧都减去在内存中的黑帧，从而就得到了消除 FPN 后的校正帧。

----结束

图6-13 FPN 校正示意图



6.15.2 API 参考

- [HI_MPI_ISP_FPNCalibrate](#): 设置去 FPN 标定属性。
- [HI_MPI_ISP_SetFPNAttr](#): 设置去 FPN 属性。
- [HI_MPI_ISP_GetFPNAttr](#): 获取去 FPN 属性。

HI_MPI_ISP_FPNCalibrate

【描述】

设置去 FPN 标定属性。

【语法】

```
HI_S32 HI_MPI_ISP_FPNCalibrate(VI_PIPE ViPipe, ISP_FPN_CALIBRATE_ATTR_S *pstCalibrateAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstCalibrateAttr	去 FPN 标定属性指针。	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 标定时用的内存注意要按 16bit 进行分配，标定完成之后要把黑帧/行，ISO，黑帧的长度、OFFSET、是否压缩标志等保存到外部存储介质，在校正的时候要用到这些信息。
- 当 VI 的 PIPE 处于 VI 在线 VPSS 离线，或者 VI 并行 VPSS 离线下才支持 FPN 标定。
- VI 并行 VPSS 离线模式下只支持 8/16bit 非压缩的标定。
- 在行模式时，只支持 16bit 非压缩。
- 在帧模式时，如果是 16bit，只能是非压缩；8bit、10bit、12bit 则可以是压缩，也可以是非压缩，压缩只支持 line 模式。
- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持行模式标定。
- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持分块模式下标定。
- Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 FPN 标定不支持压缩，只能是非压缩模式。
- 在 WDR 场景下，不支持开启 FPN 标定。
- VI 物理通道使能时，不支持开启 FPN 标定。
- Hi3516EV200/Hi3516EV300 图像宽度大于 2304 时，不支持 FPN 标定和矫正。
- 当 VI 模块的 PIPE 有销毁操作时（例如：分辨率切换、WDR 模式切换等情况），保存在 VI 模块的 FPN 属性会丢失，FPN 功能需要在 VI 模块的 PIPE 创建后重新配置才能恢复。

【举例】



标定过程请参考 sample。

【相关主题】

[HI_MPI_ISP_SetFPNAttr](#)

HI_MPI_ISP_SetFPNAttr

【描述】

设置去 FPN 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetFPNAttr(VI_PIPE ViPipe, const ISP\_FPN\_ATTR\_S *pstFPNAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstFPNAttr	去 FPN 属性指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

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

【注意】

- 16bit 的 FPN 校正必须是非压缩的。
- Hi3559AV100 VI 离线且图像宽度大于 4096 或者 VI 并行模式下只支持 8/16bit 非压缩的校正。
- 在行模式时，只支持 16bit 非压缩。



- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持行模式校正。
- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持分块模式下校正。
- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 在 WDR 场景下不支持 FPN 校正。
- Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 FPN 校正不支持压缩，只能是非压缩模式。
- Hi3519AV100 图像宽度大于 4096 时，不支持 FPN 校正。
- 根据标定时候保存的黑帧信息，读出 OFFSET、ISO 等信息，同时要把黑帧读入内存，这些是在校正时要输入信息。如果用户选择为自动模式时，会根据校正时的 ISO 来自动调节校正的强度；而手动模式会根据用户输入的校正强度进行校正。
- Hi3516EV200/Hi3516EV300 图像宽度大于 2304 时，不支持 FPN 标定和矫正。

【举例】

```
VI_CHN ViChn = 0;
VI_CHN_ATTR_S stTempChnAttr;
ISP_FPN_ATTR_S stFPNAttr;
SAMPLE_VI_FRAME_INFO_S stVMstFrame;
VI_PIPE ViPipe = 0;
ISP_FPN_FRAME_INFO_S *pstFpnFrmlInfo;
HI_S32 s32Ret;
HI_U32 u32Iso;

.....

/* start&enable vi dev */
/* start&enable vi chn */

.....

.....

/* read dark frame file:dark frame, offset, iso */

.....

pstFpnFrmlInfo = &stFPNAttr.stFpnFrmlInfo;
pstFpnFrmlInfo->u32FrmSize = stVMstFrame.u32FrmSize;
memcpy(&pstFpnFrmlInfo->stFpnFrame,
       &stVMstFrame.stVideoFrame,
       sizeof(VIDEO_FRAME_INFO_S));
stFPNAttr.bEnable = HI_TRUE;
```



```
stFPNAttr.enOpType = OP_TYPE_AUTO;
stFPNAttr.enFpnType = ISP_FPN_TYPE_FRAME;
stFPNAttr.stFpnFrmlInfo.u32Iso = u32Iso;
memcpy(&stFPNAttr.stFpnFrmlInfo.stFpnFrame, &stVMstFrame.stVideoFrame,
sizeof(VIDEO_FRAME_INFO_S));
/* start correction */
HI_MPI_ISP_SetFPNAttr(ViPipe, &stFPNAttr);
.....
```

【相关主题】

[HI_MPI_ISP_GetFPNAttr](#)

HI_MPI_ISP_GetFPNAttr

【描述】

获取去 FPN 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetFPNAttr(VI_PIPE ViPipe, ISP\_FPN\_ATTR\_S *pstFPNAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstFPNAttr	去 FPN 属性指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetFPNAttr](#)

6.15.3 数据类型

- [ISP_FPN_FRAME_INFO_S](#): 定义去 FPN 的标定黑帧信息。
- [ISP_FPN_CALIBRATE_ATTR_S](#): 定义去 FPN 的标定属性。
- [ISP_FPN_ATTR_S](#): 定义去 FPN 的校正属性。
- [ISP_FPN_MANUAL_ATTR_S](#): 定义去 FPN 在手动模式的校正属性。
- [ISP_FPN_AUTO_ATTR_S](#): 定义去 FPN 在手动模式的校正属性。
- [ISP_FPN_TYPE_E](#): 定义去 FPN 类型。

ISP_FPN_FRAME_INFO_S

【说明】

定义去 FPN 的标定黑帧信息。

【定义】

```
typedef struct hiISP_FPN_FRAME_INFO_S
{
    HI_U32          u32Iso;
    HI_U32          u32Offset[VI_MAX_SPLIT_NODE_NUM];
    HI_U32          u32FrmSize;
    VIDEO_FRAME_INFO_S  stFpnFrame;
}ISP_FPN_FRAME_INFO_S;
```



说明

宏定义如下:

```
#define VI_MAX_SPLIT_NODE_NUM    (3)
```

【成员】



成员名称	描述
u32Iso	标定黑帧时的 ISO，来自于标定时的 ISO。 ISO 值必须大于 0。
u32Offset[3]	黑帧所有像素的平均值，来自于标定时的 OFFSET。 取值范围：[0, 0xFFF]。
u32FrmSize	帧的大小，输出的黑帧有二种：压缩、非压缩，来自于标定时保存的黑帧大小。
stFpnFrame	黑帧帧信息，首先分配内存，然后把标定时的黑帧从外存读入此内存，同时对此结构体成员进行赋值。具体描述请参见《HiMPP 媒体处理软件 V4.0 开发参考》的系统控制章节。

【注意事项】

通常情况下 OFFSET 比较接近 sensor 的黑电平，若差别较大，可能会造成图像偏色，需要排查原因，OFFSET 可以在/proc/umap/isp 中查询得到。

【相关数据类型及接口】

[ISP_FPN_CALIBRATE_ATTR_S](#)

ISP_FPN_CALIBRATE_ATTR_S

【说明】

定义去 FPN 的标定属性。

【定义】

```
typedef struct hiISP_FPN_CALIBRATE_ATTR_S
{
    HI_U32                u32Threshold;
    HI_U32                u32FrameNum;
    ISP\_FPN\_TYPE\_E        enFpnType;
    ISP\_FPN\_FRAME\_INFO\_S  stFpnCaliFrame;
}ISP_FPN_CALIBRATE_ATTR_S;
```

【成员】



成员名称	描述
u32Threshold	标定时的阈值，在标定时如果像素的值大于该值，则认为是坏点，该像素点不参与标定。取值范围：[0x1, 0xFFFF]
u32FrameNum	标定的帧数，取值范围为{1, 2, 4, 8, 16}，即为 2 的整数次幂。
enFpnType	标定的类型，有二种：帧模式与行模式，帧模式校正效果要优于行模式，而行模式要比帧模式省内存。
stFpnCaliFrame	标定出来的黑帧信息。

【注意事项】

- 阈值与标定时保存黑帧的比特数相关，假设标定时保存的比特数是 N，则阈值一般取 $(2^N) \gg 2$ 。
- FPN 不支持在离线模式下进行标定。

【相关数据类型及接口】

- [ISP_FPN_FRAME_INFO_S](#)
- [ISP_FPN_ATTR_S](#)

ISP_FPN_ATTR_S

【说明】

定义去 FPN 的校正属性。

【定义】

```
typedef struct hiISP_FPN_ATTR_S
{
    HI_BOOL          bEnable;
    ISP_OP_TYPE_E    enOpType;
    ISP_FPN_TYPE_E    enFpnType;
    ISP_FPN_FRAME_INFO_S stFpnFrmInfo;
    ISP_FPN_MANUAL_ATTR_S stManual;
    ISP_FPN_AUTO_ATTR_S stAuto;
}ISP_FPN_ATTR_S;
```



【成员】

成员名称	描述
bEnable	校正使能。
enOpType	校正模式，分手动与自动模式。
enFpnType	校正类型，分帧模式与行模式。
stFpnFrmlInfo	黑帧帧信息。
stManual	手动校正属性。
stAuto	自动校正属性。

【注意事项】

校正模式分为手动与自动模式，手动模式需要用户设置校正强度，而自动模式不需要设置校正强度，系统会根据当前图像的 ISO 计算出校正强度，进行动态校正。

【相关数据类型及接口】

- [HI_MPI_ISP_SetFPNAttr](#)
- [HI_MPI_ISP_GetFPNAttr](#)
- [ISP_FPN_MANUAL_ATTR_S](#)
- [ISP_FPN_AUTO_ATTR_S](#)

ISP_FPN_MANUAL_ATTR_S

【说明】

定义去 FPN 在手动模式的校正属性。

【定义】

```
typedef struct hiISP_FPN_MANUAL_ATTR_S
{
    HI_U32          u32Strength;
}ISP_FPN_MANUAL_ATTR_S;
```

【成员】



成员名称	描述
u32Strength	手动模式时校正强度，1 倍强度为 256。

【注意事项】

无

【相关数据类型及接口】

无

ISP_FPN_AUTO_ATTR_S

【说明】

定义去 FPN 在手动模式的校正属性。

【定义】

```
typedef struct hiISP_FPN_AUTO_ATTR_S
{
    HI_U32          u32Strength;
}ISP_FPN_AUTO_ATTR_S;
```

【成员】

成员名称	描述
u32Strength	自动模式实时校正强度，只读属性（ISP FWM 根据 ISO 自动调整）。

【注意事项】

自动模式下，u32Strength 根据标定时的 ISO 与当前 ISO 动态调整，

- 当前 ISO 大于标定时的 ISO，u32Strength 大于 256；
- 当前 ISO 小于标定时的 ISO，u32Strength 小于 256。

【相关数据类型及接口】

无



ISP_FPN_TYPE_E

【说明】

定义去 FPN 类型。

【定义】

```
typedef enum hiISP_FPN_TYPE_E
{
    ISP_FPN_TYPE_FRAME = 0,
    ISP_FPN_TYPE_LINE = 1,
    ISP_FPN_TYPE_BUTT
}ISP_FPN_TYPE_E;
```

【成员】

成员名称	描述
ISP_FPN_TYPE_FRAME	帧模式校正。
ISP_FPN_TYPE_LINE	行模式校正。

【注意事项】

无

【相关数据类型及接口】

无

6.16 CA

6.16.1 功能描述

颜色调整模块支持在 YUV 空间进行色域调整的操作，这个模块下有两个模式，一个是 CA 模式，另外一个 CP 模式（热成像上色），工作的时候，两者只能二选一。

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持 CP 模式。

在 CA 模式下，通过下面的公式可以将一个像素点 (Y, U, V) 映射到另一个像素点 (Y',U',V')。

$Y'=Y;$



$U'=aU;$

$V'=aV;$

其中 a 是转换系数，采用这组公式可以在一定程度上保持亮度和色调的恒定，对像素点的饱和度做一个调整。转换系数 a 和像素点亮度 Y 联系，就可以根据亮度的变化来调整饱和度，达到局部调整饱和度的目的，亮处的颜色更鲜艳，暗处的色噪不明显。同时，转换系数 a 和 ISP 的 ISO 值联系，达到降低低照度下的暗处色噪的目的。

在 CP 模式下，热成像的图像只有亮度信息，该模式下通过亮度信息 Y 查找上色的色板，查找对应的 YUV 的值作为输出的值。其中，色板是通过 YUV 格式存储的。

6.16.2 API 参考

- [HI_MPI_ISP_SetCAAttr](#): 设置 CA 属性。
- [HI_MPI_ISP_GetCAAttr](#): 获取 CA 属性。

HI_MPI_ISP_SetCAAttr

【描述】

设置 CA 属性

【语法】

```
HI_S32 HI_MPI_ISP_SetCAAttr(VI_PIPE ViPipe, const ISP\_CA\_ATTR\_S *pstCAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstCAAttr	CA 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_GetCAAttr](#)

HI_MPI_ISP_GetCAAttr

【描述】

获取 CA 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetCAAttr(VI_PIPE ViPipe, ISP\_CA\_ATTR\_S*pstCAAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstCAAttr	CA 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】

HDR 模式，默认关闭 CA，客户可以自行设置打开。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetCAAttr](#)

6.16.3 数据类型

- [ISP_CA_TYPE_E](#)：色彩调整模块的工作模式，0 为 CA 模式，1 为 CP 模式。
- [ISP_CA_LUT_S](#)：CA 模式的属性。
- [ISP_CP_LUT_S](#)：CP 模式的属性。
- [ISP_CA_ATTR_S](#)：定义颜色调整模块的属性，包含了 CA 模式属性和 CP 模式属性。

ISP_CA_TYPE_E

【说明】

定义颜色调整模块的工作模式。0 为 CA 模式，1 为 CP 模式。

【定义】

```
typedef enum hiISP_CA_TYPE_E
{
    ISP_CA_ENABLE = 0x0,
    ISP_CP_ENABLE,
    ISP_CA_BUTT
} ISP_CA_TYPE_E;
```

【成员】

成员名称	描述
ISP_CA_ENABLE	0x0：CA 模式。



成员名称	描述
ISP_CP_ENABLE	0x1: CP 模式。

【注意事项】

无。

【相关数据类型及接口】

无

ISP_CA_LUT_S

【说明】

定义 CA 模式的属性。

【定义】

```
typedef struct hiISP_CA_LUT_S
{
    HI_U32 au32YRatioLut[HI_ISP_CA_YRATIO_LUT_LENGTH];
    HI_S32 as32ISORatio[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_CA_LUT_S;
```

 说明

Hi3559AV100/Hi3519AV100 中宏定义如下:

```
#define HI_ISP_CA_YRATIO_LUT_LENGTH    (256)
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 中宏定义如下:

```
#define HI_ISP_CA_YRATIO_LUT_LENGTH    (128)
```

【成员】

成员名称	描述
au32YRatioLut	CA 模式, 根据亮度 Y 查找 UV 增益的查找表
as32ISORatio	CA 模式, 根据 ISO 值查找 UV 增益的查找表

【注意事项】

无。



【相关数据类型及接口】

无

ISP_CP_LUT_S

【说明】

定义 CP 模式的属性，也就是色板的 YUV 值。

【定义】

```
typedef struct hiISP_CP_LUT_S
{
    HI_U8    au8CPLutY[HI_ISP_CA_YRATIO_LUT_LENGTH]; //8bit
    HI_U8    au8CPLutU[HI_ISP_CA_YRATIO_LUT_LENGTH]; //8bit
    HI_U8    au8CPLutV[HI_ISP_CA_YRATIO_LUT_LENGTH]; //8bit
} ISP_CP_LUT_S;
```

【成员】

成员名称	描述
au8CPLutY	CP 模式，根据亮度 Y 查找色板中的 Y 值
au8CPLutU	CP 模式，根据亮度 Y 查找色板中的 U 值
au8CPLutV	CP 模式，根据亮度 Y 查找色板中的 V 值

【注意事项】

仅 Hi3559AV100/Hi3519AV100 支持 CP 模式。

【相关数据类型及接口】

无

ISP_CA_ATTR_S

【说明】

定义颜色调整模块的属性，包含了 CA 模式属性和 CP 模式属性。

【定义】

```
typedef struct hiISP_CA_ATTR_S
```



```
{  
    HI_BOOL bEnable;          /* RW, Range: [0, 1] */  
    ISP_CA_TYPE_E eCaCpEn;  
    ISP_CA_LUT_S stCA;  
    ISP_CP_LUT_S stCP;  
}ISP_CA_ATTR_S;
```

【成员】

成员名称	描述
bEnable	色彩调整模块的使能。
eCaCpEn	色彩调整模块的工作模式。 仅 Hi3559AV100/Hi3519AV100 支持
stCA	CA 模式的属性。
stCP	CP 模式的属性。 仅 Hi3559AV100/Hi3519AV10 支持

【注意事项】

无

【相关数据类型及接口】

无

6.17 CAC

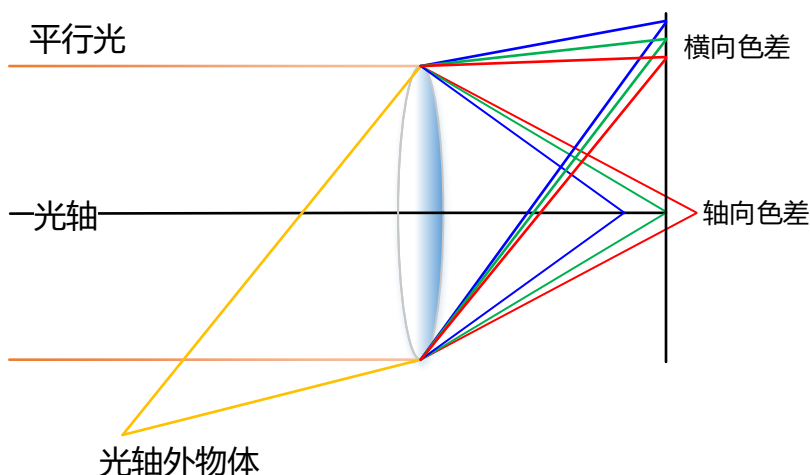
6.17.1 功能描述

色差(Chromatic Aberration)是指光学上透镜无法将各种波长的光聚焦在同一点上的现象，是一种与镜头有关的缺陷，它产生的主要原因是不同波长的光具有不同的折射率(色散现象)。色差可以分为如下的两类：

- 轴向色差(Axial Chromatic Aberration)
 - 透镜对于不同的光会有不同的折射率，因此不同颜色的光会聚焦在不同的焦平面上，大口径镜头容易产生这种色差，缩小光圈可以减弱轴向色差。

- 人眼对于 G 通道更敏感，一般 G 通道可以正确对焦，从而引起 R、B 的模糊，造成高光区与低光区交界处出现明显的紫边表现。
- 具有明显的局部特性，因此校正紫边时采用 Local CAC。
- 横向色差(Lateral Chromatic Aberration)
 - 透镜的放大倍数也与折射率有关，它使得不同波长光线的像高不同，即不同波长的光会聚焦在焦平面上不同的位置，会造成 R、G、B 3 通道具有不同的影像高度，在影像上产生色的错位。
 - 越偏离图像中心，横向色差越明显，一般横向色差表现为物体相对的两侧边缘出现不同的颜色，但具体表现为什么颜色与镜头组密切相关，不同的镜头组会表现出不同种类的颜色边缘。
 - 具有全局特性，在校正时采用 Global CAC，而不同镜头的校正参数由标定得到。

图6-14 色差图解



6.17.2 API 参考

- [HI_MPI_ISP_SetLocalCacAttr](#): 设置紫边检测校正参数。
- [HI_MPI_ISP_GetLocalCacAttr](#): 获取紫边检测校正参数。
- [HI_MPI_ISP_SetGlobalCacAttr](#): 设置横向色差校正参数。
- [HI_MPI_ISP_GetGlobalCacAttr](#): 获取横向色差校正参数。

HI_MPI_ISP_SetLocalCacAttr

【描述】



设置紫边检测校正参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetLocalCacAttr(VI_PIPE ViPipe, const ISP_LOCAL_CAC_ATTR_S  
*pstLocalCacAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstLocalCacAttr	紫边检测校正参数。	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_GetLocalCacAttr](#)

HI_MPI_ISP_GetLocalCacAttr

【描述】

获取紫边检测校正参数。



【语法】

```
HI_S32 HI_MPI_ISP_GetLocalCacAttr(VI_PIPE ViPipe, ISP\_LOCAL\_CAC\_ATTR\_S *pstLocalCacAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstLocalCacAttr	紫边检测校正参数。	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_SetLocalCacAttr](#)

HI_MPI_ISP_SetGlobalCacAttr

【描述】

设置横向色差校正参数。

【语法】



```
HI_S32 HI_MPI_ISP_SetGlobalCacAttr(VI_PIPE ViPipe, const ISP\_GLOBAL\_CAC\_ATTR\_S  
*pstGlobalCacAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstGlobalCacAttr	横向色差校正参数。	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】

- 校正参数由标定得到，不同的镜头的校正参数是不同的，因此需要对每个镜头进行标定。
- Hi3516EV200 系列芯片不支持 GlobalCAC 功能，调用 SetGlobalCacAttr 和 GetGlobalCacAttr 接口将无效。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetGlobalCacAttr](#)

HI_MPI_ISP_GetGlobalCacAttr

【描述】



获取横向色差校正参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetGlobalCacAttr(VI_PIPE ViPipe, ISP\_GLOBAL\_CAC\_ATTR\_S  
*pstGlobalCacAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstGlobalCacAttr	横向色差校正参数。	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】

Hi3516EV200 系列芯片不支持 GlobalCAC 功能，调用 SetGlobalCacAttr 和 GetGlobalCacAttr 接口将无效。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetBlackLevelAttr](#)

6.17.3 数据类型

- [ISP_DEPURPLESTR_MANUAL_ATTR_S](#)：紫边校正手动属性。



- [ISP_DEPURPLESTR_AUTO_ATTR_S](#): 紫边校正自动属性。
- [ISP_LOCAL_CAC_ATTR_S](#): 紫边检测校正属性。
- [ISP_GLOBAL_CAC_ATTR_S](#): 横向色差校正属性。

ISP_DEPURPLESTR_MANUAL_ATTR_S

【说明】

定义紫边校正手动属性。

【定义】

```
typedef struct hiISP_DEPURPLESTR_MANUAL_ATTR_S
{
    HI_U8 u8DePurpleCrStr;    /*Range: [0,8]*/
    HI_U8 u8DePurpleCbStr;    /*Range: [0,8]*/
}ISP_DEPURPLESTR_MANUAL_ATTR_S;
```

【成员】

成员名称	描述
u8DePurpleCrStr	R 通道的校正强度。
u8DePurpleCbStr	B 通道的校正强度。

【注意事项】

u8DePurpleCrStr 和 u8DePurpleCbStr 矫正强度过大，可能出现边界不连续现象或闪烁现象。

【相关数据类型及接口】

[ISP_LOCAL_CAC_ATTR_S](#)

ISP_DEPURPLESTR_AUTO_ATTR_S

【说明】

定义紫边校正自动属性

【定义】

```
typedef struct hiISP_DEPURPLESTR_AUTO_ATTR_S
```



```
{  
    HI_U8 au8DePurpleCrStr[ISP_EXP_RATIO_STRENGTH_NUM];    /*Range: [0,8]*/  
    HI_U8 au8DePurpleCbStr[ISP_EXP_RATIO_STRENGTH_NUM];    /*Range: [0,8]*/  
}ISP_DEPURPLESTR_AUTO_ATTR_S;
```



说明

宏定义如下：

```
#define ISP_EXP_RATIO_STRENGTH_NUM    (16)
```

【成员】

成员名称	描述	取值范围
au8DePurpleCrStr[ISP_EXP_RATIO_STRENGTH_NUM]	自动模式下，R 通道的校正强度。该组的 16 个值分别对应 16 个不同的曝光比，一般情况下，曝光比越大，配置的去紫边的强度值越大，对应关系如表 6-4 所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个曝光比值对应的处理强度。	[0, 8]
au8DePurpleCbStr[ISP_EXP_RATIO_STRENGTH_NUM]	自动模式下，B 通道的校正强度。	[0,8]

表6-4 au8DePurpleCrStr[16]在不同曝光比情况下对应的设置值

au8DePurpleCrStr	曝光比
au8DePurpleCrStr[0]	64
au8DePurpleCrStr[1]	128
au8DePurpleCrStr[2]	192
au8DePurpleCrStr[3]	256
au8DePurpleCrStr[4]	320
au8DePurpleCrStr [5]	384
au8DePurpleCrStr [6]	448
au8DePurpleCrStr [7]	512
au8DePurpleCrStr [8]	576



au8DePurpleCrStr	曝光比
au8DePurpleCrStr[9]	640
au8DePurpleCrStr[10]	704
au8DePurpleCrStr[11]	768
au8DePurpleCrStr[12]	832
au8DePurpleCrStr[13]	896
au8DePurpleCrStr[14]	960
au8DePurpleCrStr[15]	1023

【注意事项】

- 曝光比是 WDR 模式下才有的概念，线性模式下获取的曝光比值总为 1，因此在线性模式下虽然有联动，生效的总是第一组配置值。
- 在 built-in WDR 模式下，本参数并不和曝光比联动，为与线性模式区分，生效的总是最后一组即曝光比为 1023 的配置值。
- 在 WDR 自动长帧模式和 WDR 长帧模式下，曝光比生效的总是第一组配置，不会发生变化。

【相关数据类型及接口】

[ISP_LOCAL_CAC_ATTR_S](#)

ISP_LOCAL_CAC_ATTR_S

【说明】

紫边检测校正属性。

【定义】

```
typedef struct hiISP_LOCAL_CAC_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16      u16PurpleDetRange;
    HI_U16      u16VarThr;
    HI_U16      u16RDetThr[ISP_LCAC_DET_NUM];
    HI_U16      u16GDetThr[ISP_LCAC_DET_NUM];
    HI_U16      u16BDetThr[ISP_LCAC_DET_NUM];
}
```



```
HI_U16          u16LumaDetThr[ISP_LCAC_DET_NUM];
HI_S16          s16CbCrRatio[ISP_LCAC_DET_NUM];
ISP_OP_TYPE_E   enOpType;
ISP_DEPURPLESTR_MANUAL_ATTR_S  stManual;
ISP_DEPURPLESTR_AUTO_ATTR_S  stAuto;
} ISP_LOCAL_CAC_ATTR_S;
```

说明

宏定义如下：

```
#define ISP_LCAC_DET_NUM    3
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	紫边校正使能	[0,1] 0：禁止； 1：使能	1
u16PurpleDetRange	紫色检测的范围 值越大，越多非高亮区域的 紫色被界定为紫边区域	[0,410]	线性模式:60 WDR 模式:195
u16VarThr	边缘检测阈值	[0, 4095]	0
u16RDetThr	高亮区域检测 R 分量阈值	[0, 4095]	[1500, 1500, 0]
u16GDetThr	高亮区域检测 G 分量阈值	[0, 4095]	[1500, 1500, 0]
u16BDetThr	高亮区域检测 B 分量阈值	[0, 4095]	[4095, 1500, 0]
u16LumaDetThr	高亮区域检测亮度阈值	[0, 4095]	[3200, 1500, 0]
s16CbCrRatio	紫边检测，区域颜色下限	[-2048, 2047]	[-50, -50, -50]
enOpType	紫边校正工作模式	[0,1] 0：自动 1：手动	0
stManual	手动模式下配置紫边校正强度参数。	-	-
stAuto	自动模式下配置紫边校正强	-	-



成员名称	描述	取值范围	默认值
	度参数。		

【注意事项】

- 由于算法设计，LocalCAC 矫正强度会受到 Demosaic 结果影响。因此将 LocalCAC 强度设为最小时依然会保留一定去紫边能力，对高光处紫边表现有一定影响。如果想完全关闭 LocalCAC 效果需直接将 bEnable 置为 FALSE。
- s16CbCrRatio 接口仅在 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 中有效。

【相关数据类型及接口】

- [ISP_OP_TYPE_E](#)
- [ISP_DEPURPLESTR_AUTO_ATTR_S](#)
- [ISP_DEPURPLESTR_MANUAL_ATTR_S](#)

ISP_GLOBAL_CAC_ATTR_S

【说明】

定义横向色差校正属性。

【定义】

```
typedef struct hiISP_GLOBAL_CAC_ATTR_S
{
    HI_BOOL      bEnable;
    HI_U16      u16VerCoordinate;
    HI_U16      u16HorCoordinate;
    HI_S16      s16ParamRedA;
    HI_S16      s16ParamRedB;
    HI_S16      s16ParamRedC;
    HI_S16      s16ParamBlueA;
    HI_S16      s16ParamBlueB;
    HI_S16      s16ParamBlueC;
    HI_U8       u8VerNormShift;
    HI_U8       u8VerNormFactor;
    HI_U8       u8HorNormShift;
    HI_U8       u8HorNormFactor;
}
```



```
HI_U16      u16CorVarThr;
} ISP_GLOBAL_CAC_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	横向色差校正使能	[0,1] 0: 禁止; 1: 使能	0
u16VerCoordinate	光学中心垂直坐标	[0,8191]	-
u16HorCoordinate	光学中心水平坐标	[0,8191]	-
s16ParamRedA	R 通道对应的半径多项式的系数 a	[-256,255]	3
s16ParamRedB	R 通道对应的半径多项式的系数 b	[-256,255]	-10
s16ParamRedC	R 通道对应的半径多项式的系数 c	[-256,255]	2
s16ParamBlueA	B 通道对应的半径多项式的系数 a	[-256,255]	3
s16ParamBlueB	B 通道对应的半径多项式的系数 b	[-256,255]	-16
s16ParamBlueC	B 通道对应的半径多项式的系数 c	[-256,255]	7
u8VerNormShift	垂直方向的归一化移位参数	[0,7]	6
u8VerNormFactor	垂直方向的归一化系数	[0,31]	29
u8HorNormShift	水平方向的归一化移位参数	[0,7]	6
u8HorNormFactor	水平方向的归一化参数	[0,31]	29
u16CorVarThr	横向色差校正的方差阈值	[0,4095]	130

【注意事项】

- Hi3516EV200 系列不支持。
- 假定图像的宽高别为 u32Width、u32Height，则 u16VerCoordinate、u16HorCoordinate 的默认值分别为： $u16VerCoordinate = u32Height/2 - 1$ ， $u16HorCoordinate = u32Width/2 - 1$ ，1080P 下对应的默认值分别为 539、959。



- u8VerNormShift、u8VerNormFactor、u8HorNormShift、u8HorNormFactor 是与图像分辨率相关的参数，在图像分辨率确定之后，这四个参数是固定的，1080p 下，这 4 个参数对应的默认值为：6、29、6、29。分辨率切换后，这四个值需要重新标定得到。

【相关数据类型及接口】

无。

6.18 WDR

6.18.1 功能描述

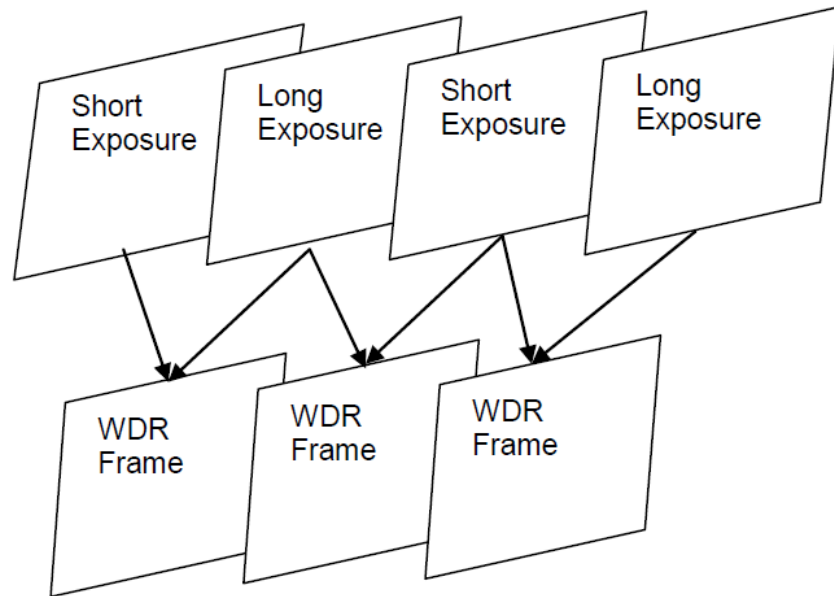
由于传统的图像传感器的动态范围比较低，因此，如果用传统的图像传感器拍摄高动态范围的场景，就会要么亮处过曝，丢失很多细节；要么暗处曝光不足，细节难以分辨。为了记录高动态范围场景的每一个细节，可以采用多次曝光图像合成的方法。

WDR 算法的目的就是将从 sensor 输出的长短帧数据合成为一帧 20bit 的数据。如图 6-15 所示，长短帧数据是先缓存在 DDR 中，然后同时给 WDR 模块进行合成。合成的原则是择优选取，即对于比较亮的区域，采用短帧数据，而对于比较暗的区域，采用长帧数据。

须知

Hi3516EV200 只支持 WDR 帧模式功能。

图6-15 WDR 算法合成模型



- 本模块可以通过 `enWDRMergeMode` 来决定合成的方式。若该值为 0，则用传统的 WDR 功能合成；若该值为 1，则打开 Fusion 功能，即用来改善 Flicker 现象；其中 `stWDRCombine` 结构体用来控制传统的 WDR 功能，而 `stFusion` 结构体用来控制 Fusion 功能。
- 传统的 WDR，包括以下功能：
 - 可以去除鬼影。通过 `enFSWDRComMode` 来选择融合区域是采用短帧还是长帧；
 - 可以通过 `u16ShortThr` 和 `u16LongThr` 来控制选择长短帧的范围。即对于大于 `u16ShortThr` 的像素选择短帧，小于 `u16LongThr` 的像素选择长帧，而对于在两者之间的像素采用长短帧融合的方式；
 - 可以通过 `u16ForceLongLowThr` 和 `u16ForceLongHigThr` 来控制运动区域选择长帧，对于亮度小于 `u16ForceLongLowThr` 的运动区域强制选择长帧，亮度大于 `u16ForceLongHigThr` 的运动区域不强制选择长帧，对于亮度在两者之间的运动区域采用长短帧融合的方式；
 - 可以通过 `u8MdThrLowGain`，`u8MdThrHigGain` 来控制运动检测；
 - 可以通过 `bMDRefFlicker` 来决定运动检测是否参考 Flicker 检测结果；
 - 可以通过 `bShortExpoChk` 来控制 WDR 融合是否检查短帧数据大小。
- Fusion 功能指适用于一般的室内场景（非宽动态）以及夜晚场景，用来改善工频闪问题。其中 `au16FusionThr[4]` 是用来控制 Fusion 选择时每帧的阈值。



- 本模块的降噪功能，可以通过 stBnr 结构体来控制。其中 enBnrMode 可以控制 BNR 功能是否开关，并通过 u8FusionBnrStr/au8GsigmaGain/au8RbsigmaGain 等参数来控制 BNR 的强度。
Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持降噪功能。

6.18.2 API 参考

- [HI_MPI_ISP_SetFSWDRAttr](#)：设置动态范围压缩参数。
- [HI_MPI_ISP_GetFSWDRAttr](#)：获取动态范围压缩参数。

HI_MPI_ISP_SetFSWDRAttr

【描述】

设置帧合成参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetFSWDRAttr(VI_PIPE ViPipe, const ISP\_WDR\_FS\_ATTR\_S *pstFSWDRAttr)
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstFSWDRAttr	帧合成参数	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】



当前 ISP 处理的图像宽度大于 4096 时，不支持 FS-WDR 模式。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetFSWDRAttr](#)

HI_MPI_ISP_GetFSWDRAttr

【描述】

获取帧合成参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetFSWDRAttr(VI_PIPE ViPipe, ISP\_WDR\_FS\_ATTR\_S *pstFSWDRAttr)
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstFSWDRAttr	帧合成参数	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】

当前 ISP 处理的图像宽度大于 4096 时，不支持 WDR 模式。



【举例】

无

【相关主题】

[HI_MPI_ISP_SetFSWDRAttr](#)

6.18.3 数据类型

- [ISP_WDR_MERGE_MODE_E](#): 定义 ISP 合成方式属性。
- [ISP_FSWDR_MDT_ATTR_S](#): 定义 ISP FSWDR 运动检测的属性。
- [ISP_WDR_COMBINE_ATTR_S](#): 定义 ISP FSWDR 属性。
- [ISP_FUSION_ATTR_S](#): 定义 ISP FUSION 属性。
- [ISP_WDR_BNR_ATTR_S](#): 定义 WDR BNR 属性。
- [ISP_BNR_MODE_E](#): 定义 BNR 开关属性。
- [ISP_WDR_WBGAIN_POSITION_E](#): 定义白平衡增益在 ISP 的位置属性。
- [ISP_FSWDR_MANUAL_ATTR_S](#): 定义 ISP FSWDR 手动属性。
- [ISP_FSWDR_AUTO_ATTR_S](#): 定义 ISP FSWDR 自动属性。
- [ISP_WDR_FS_ATTR_S](#): 定义 ISP FSWDR 属性。

ISP_WDR_MERGE_MODE_E

【说明】

定义 ISP 合成方式属性。

【定义】

```
typedef enum hiISP_WDR_MERGE_MODE_E
{
    MERGE_WDR_MODE        = 0,
    MERGE_FUSION_MODE      = 1,
    MERGE_BUTT
} ISP_WDR_MERGE_MODE_E;
```

【成员】

成员名称	描述
MERGE_WDR_MODE	0x0: 运行在传统 WDR 模式下



MERGE_FUSION_MODE	0x1：运行在 FUSION 模式下
MERGE_BUTT	无效值

【注意事项】

无。

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_FSWDR_MDT_ATTR_S

【说明】

定义 ISPFSWDR 运动检测属性。

【定义】

```
typedef struct hiISP_MDT_ATTR_S
{
    HI_BOOL bShortExpoChk;
    HI_U16  u16ShortCheckThd;
    HI_BOOL bMDRefFlicker;
    HI_U8   u8MdtStillThd;
    HI_U8   u8MdtFullThd;
    HI_U8   u8MdtLongBlend;
    ISP\_OP\_TYPE\_E enOpType;
    ISP\_FSWDR\_MANUAL\_ATTR\_S stManual;
    ISP\_FSWDR\_AUTO\_ATTR\_S   stAuto;
} ISP_FSWDR_MDT_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bShortExpoChk	控制运动检测是否检查短帧数据大小，当短帧数据接近 0 时，WDR 融合强制选长帧。 0：WDR 融合时不检查短帧数据大	[0,0x1] 1：使能； 0：禁止。	0x0



成员名称	描述	取值范围	默认值
	小; 1: WDR 融合时检查短帧数据大小。		
u16ShortCheckThd	短帧检查阈值。当短帧亮度小于该阈值时, WDR 强制输出长帧。	[0x,0xFFF]	-
bMDRefFlicker	控制运动判断是否参考 Flicker 检测结果。 0: 做运动判断时不参考 Flicker 检测结果; 1: 做运动判断时参考 Flicker 检测结果。	[0,0x1] 1: 使能; 0: 禁止。	0x0
u8MdtStillThd	运动检测中判断为静止的阈值, 当运动程度小于该阈值时, 判断该像素点为静止	[0,0xFE]	-
u8MdtFullThd	运动检测中判断为全运动的阈值, 当运动程度大于该阈值时, 判断该像素点为全运动, Hi3559AV100 WDR 两帧合成不支持该参数。 Hi3519AV100/Hi3516CV500//Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。	[0,0xFE]	-
u8MdtLongBlend	运动区域叠加长帧的权重, 当该值为 0xFE 时, 运动区域全部选择长帧	[0x0,0xFE]	-
enOpType	FSWDR 工作类型。 • OP_TYPE_AUTO: 自动; • OP_TYPE_MANUAL: 手动。	-	OP_TYPE_AUTO
stManual	FSWDR 手动参数, 具体参见 ISP_FSWDR_MANUAL_ATTR_S	-	-
stAuto	FSWDR 自动参数, 具体参见 ISP_FSWDR_AUTO_ATTR_S	-	-



【注意事项】

无。

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_WDR_COMBINE_ATTR_S

【说明】

定义 ISPFSDR 属性。

【定义】

```
typedef struct hiISP_WDR_COMBINE_ATTR_S
{
    HI_BOOL bMotionComp;
    HI_U16  u16ShortThr;
    HI_U16  u16LongThr;
    HI_BOOL bForceLong
    HI_U16  u16ForceLongLowThr
    HI_U16  u16ForceLongHigThr;
    ISP\_FSWDR\_MDT\_ATTR\_S stWDRMdt;
} ISP_WDR_COMBINE_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bMotionComp	WDR 运动补偿使能。 0：关闭； 1：使能。 在某些室内日光灯场景下，运动补偿会加重频闪带来的横条纹现象，此时可以关闭运动补偿。	[0, 0x1] 1：使能； 0：禁止；	0x1
u16ShortThr	短曝光门限值，超过门限值的图像数据只选择短曝光数据。	[0,0xFFF]	0xFC0



u16LongThr	长曝光门限值，低于门限值的图像数据只选择长曝光数据。	[0, u16ShortThr]	0xBC0
bForceLong	强制输出长帧使能 Hi3559AV100 不支持。	[0,0x1] 1: 使能 0: 禁止	-
u16ForceLongLowThr	强制输出长帧的低阈值，亮度低于该阈值的运动区域强制输出长帧，Hi3559AV100 不支持。	[0x0, 0xFFFF]	-
u16ForceLongHighThr	强制输出长帧的高阈值，亮度大于该阈值的运动区域不强制输出长帧，Hi3559AV100 不支持。	[0x0,0xFFFF]	-
stWDRMdt	FSWDR 运动检测参数，具体参见 ISP_FSWDR_MDT_ATTR_S 。	-	-

【注意事项】

注意 u16ForceLongLowThr 应小于 u16ForceLongHighThr。

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_FUSION_ATTR_S

【说明】

定义 ISP FUSION 属性。

【定义】

```
typedef struct hiISP_FUSION_ATTR_S
{
    HI_U16  au16FusionThr[WDR_MAX_FRAME_NUM];
} ISP_FUSION_ATTR_S;
```




【成员】

成员名称	描述	取值范围	默认值
au16FusionThr[WDR_MAX_FRAME_NUM]	对应 Fusion 模式下超短帧，短帧，中帧，长帧的阈值，将输入数据限制在阈值以下。	[0,0x3FFF]	[0x3E80,0x36B0,0x3E80,0x36B0]

【注意事项】

无。

【相关数据类型及接口】

ISP_WDR_FS_ATTR_S

ISP_WDR_BNR_ATTR_S

【说明】

定义 WDR BNR 属性。

【定义】

```
typedef struct hiISP_WDR_BNR_ATTR_S
{
    ISP_BNR_MODE_E enBnrMode;
    HI_BOOL        bShortFrameNR;
    HI_U8          u8ShortFrameNRStr;
    HI_U8          u8FusionBnrStr;
    HI_U8          au8NoiseFloor[NOISESET_ELENUM];
    HI_U8          u8FullMdtSigGWgt;
    HI_U8          u8FullMdtSigRBWgt;
    HI_U8          u8FullMdtSigWgt;
    HI_U8          au8GsigmaGain[3];
    HI_U8          au8RBSigmaGain[3];
} ISP_WDR_BNR_ATTR_S;
```



说明

宏定义如下：

```
#define NOISESET_ELENUM (7)
```



【成员】

成员名称	描述	取值范围	默认值
enBnrMode	去噪功能的开关 0: BNR 关闭; 1: BNR 打开。 Hi3516CV500/Hi3559V200 不支持	[0, 0x1]	0x1
bShortFrameNR	短帧降噪开关 1: 使能; 0: 禁止。 Hi3559AV100/Hi3516CV500/Hi3559V200 / Hi3516EV200/Hi3516EV300 不支持	[0, 0x1]	-
u8ShortFrameNRSt r	短帧降噪强度, 值越大, 降噪强度越大, Hi3559AV100/Hi3516CV500/Hi3559V200 /Hi3516EV200/ Hi3516EV300 不支持	[0,0x3F]	-
u8FusionBnrStr	Fusion 模式下降噪强度, 值越大, 降噪强度越大, Hi3559AV100/Hi3516CV500/Hi3559V200 /Hi3516EV200/Hi3516EV300 不支持	[0,0x3F]	-
au8NoiseFloor[NOI SESET_ELENUM]	噪声基底, 值越大, 降噪越强 Hi3516CV500/Hi3559V200/Hi3516EV200 /Hi3516EV300 不支持	[0,0xFF]	-
u8FullMdtSigGWgt	全运动区域 G 通道的降噪强度, 值越大, 降噪强度越大。 Hi3559AV100/Hi3516CV500/Hi3559V200 /Hi3516EV200/Hi3516EV300 不支持	[0,0x1F]	-
u8FullMdtSigRBWg t	全运动区域 RB 通道的降噪强度, 值越大, 降噪强度越大。 Hi3559AV100/Hi3516CV500/Hi3559V200 /Hi3516EV200/Hi3516EV300 不支持	[0,0x1F]	-
u8FullMdtSigWgt	全运动区域的降噪强度, 不分通道, 值越	[0,0x1F	-



成员名称	描述	取值范围	默认值
	大，降噪强度越大。 Hi3519AV100/Hi3516CV500/Hi3559V200 /Hi3516EV200/Hi3516EV300 不支持	]	
au8GsigmaGain[3]	G 通道的三级降噪强度，值越大，强度越大， Hi3516CV500/Hi3559V200/Hi3516EV200 /Hi3516EV300 不支持。	[0,0xFF]	-
au8RBsigmaGain[3]	RB 通道的三级降噪强度，值越大，强度越大， Hi3516CV500/Hi3559V200/Hi3516EV200 /Hi3516EV300 不支持。	[0,0xFF]	-

【注意事项】

- 当 au8GsigmaGain[3]参数调到较大时，强边缘可能会出现绿边。
- 当 au8RBsigmaGain [3]参数调到较大时，强边缘可能会出现紫边。

【相关数据类型及接口】[ISP_WDR_FS_ATTR_S](#)**ISP_BNR_MODE_E****【说明】**

定义 ISP BNR 开关属性。

【定义】

```
typedef enum hiISP_BNR_MODE_E
{
    BNR_OFF_MODE    = 0,
    BNR_ON_MODE     = 1,
    BNR_BUTT
} ISP_BNR_MODE_E;
```

【成员】



成员名称	描述
BNR_OFF_MODE	0: BNR 关闭;
BNR_ON_MODE	1: BNR 打开;
BNR_BUTT	无效值

【注意事项】

无。

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_WDR_WBGAIN_POSITION_E

【说明】

定义白平衡增益在 ISP 的位置属性。

【定义】

```
typedef enum hiISP_WDR_WBGAIN_POSITION_E
{
    WDR_WBGAIN_IN_DG1      = 0,
    WDR_WBGAIN_IN_WB       = 1,
    WDR_WBGAIN_POSITION_BUTT
} ISP_WDR_WBGAIN_POSITION_E;
```

【成员】

成员名称	描述
WDR_WBGAIN_IN_DG1	0x0: 白平衡增益配置在 WDR 合成前的 DG1 处
WDR_WBGAIN_IN_WB	0x1: 白平衡增益配置在 WB 处
WDR_WBGAIN_POSITION_BUTT	无效值

【注意事项】



- 白平衡增益配置在 WDR 合成前的 DG1 处，是为 Hi3559AV100 的 WDR 设计的工作模式，其他芯片请保持白平衡增益配置在 WB 处。
- 如果需要在 WDR 正常模式，WDR Fusion 模式和长帧模式间切换，在在线模式下，需要将白平衡增益配置在 WDR 合成前的 DG1 处，并保持不变。在线模式下，如果改变增益配置的位置，会导致切换时的图像异常。在离线模式下，可以改变增益配置的位置。
- 当白平衡增益配置在 WDR 合成前的 DG1 处时，BypassWBGain 功能失效，因为 BypassWBGain 功能是关闭 WB 处的白平衡增益，无法关闭 DG1 处的白平衡增益。

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_FSWDR_MANUAL_ATTR_S

【说明】

定义 ISP FSWDR 手动属性。

【定义】

```
typedef struct hiISP_FSWDR_MANUAL_ATTR_S
{
    HI_U8 u8MdThrLowGain;
    HI_U8 u8MdThrHigGain;
} ISP_FSWDR_MANUAL_ATTR_S;
```

【成员】

成员名称	描述	取值范围
u8MdThrLowGain	手动模式下判断是否运动的低阈值系数。值越大，运动程度越小。	[0, u8MdThrHigGain]
u8MdThrHigGain	手动模式下判断是否运动的高阈值系数。值越大，运动程度越小。	[0, 0xFF]

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)



ISP_FSWDR_AUTO_ATTR_S

【说明】

定义 ISP FSWDR 自动属性。

【定义】

```
typedef struct hiISP_FSWDR_AUTO_ATTR_S
{
    HI_U8 au8MdThrLowGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U8 au8MdThrHigGain[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_FSWDR_AUTO_ATTR_S;
```

【成员】

成员名称	描述	取值范围
au8MdThrLowGain [ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下判断是否运动的低 阈值系数。值越大，运动程度 越小。	[0,0xFF]
au8MdThrHigGain [ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下判断是否运动的高 阈值系数。值越大，运动程度 越小。	[0,0xFF]

【相关数据类型及接口】

[ISP_WDR_FS_ATTR_S](#)

ISP_WDR_FS_ATTR_S

【说明】

定义 ISP FSWDR 属性。

【定义】

```
typedef struct hiISP_WDR_FS_ATTR_S
{
    ISP\_WDR\_MERGE\_MODE\_E enWDRMergeMode;
    ISP\_WDR\_COMBINE\_ATTR\_S stWDRCombine;
    ISP\_WDR\_BNR\_ATTR\_S stBnr;
    ISP\_FUSION\_ATTR\_S stFusion;
```



```
ISP_WDR_WBGAIN_POSITION_E enWDRWbgainPosition;  
} ISP_WDR_FS_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
enWDRMergeMode	合成方式 0: 传统的 WDR 合成; 1: Fusion 合成。 在室内正常照度下存在工频闪时或者夜晚场景, 建议采用 Fusion 合成, 具体参见 ISP_WDR_MERGE_MODE_E	[0,0x1]	0x0
stWDRCombine	传统的 WDR 合成结构体, 具体参见 ISP_WDR_COMBINE_ATTR_S		
stFusion	Fusion 合成结构体, 具体参见 ISP_FUSION_ATTR_S		
stBnr	去噪功能结构体, 具体参见 ISP_WDR_BNR_ATTR_S , 仅 Hi3559AV100/Hi3519AV100 支持		
enWDRWbgainPosition	白平衡增益在 ISP 的位置。 0: 白平衡增益配置在 WDR 合成前的 DG1 处。 1: 白平衡增益配置在 WB 处。 Hi3559AV100 工作在 Fusion 合成模式下, 建议采用白平衡增益配置在 WDR 合成前的 DG1 处, 具体参见 ISP_WDR_WBGAIN_POSITION_E 。 Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持		

【注意事项】



WDR 功能分为自动和手动：

- enOpType 为 OP_TYPE_AUTO，使用自动 WDR 功能。
- enOpType 设置为 OP_TYPE_MANUAL，使用手动 WDR 功能。

6.19 AWB 量产标定工具

6.19.1 功能描述

AWB 量产标定工具实现在量产过程中，针对不同机器的差异进行对 AWB 差异参数校正的功能。用户可以调用 [HI_MPI_ISP_GetLightboxGain](#) 接口，在色温范围 4500K 到 6500K 的均匀光源下的均匀背景环境下，获取图像中心若干个块的 Rgain 和 Bgain 的均值，即 AWB 增益，用于量产过程中矫正不同机器的 AWB 差异参数。

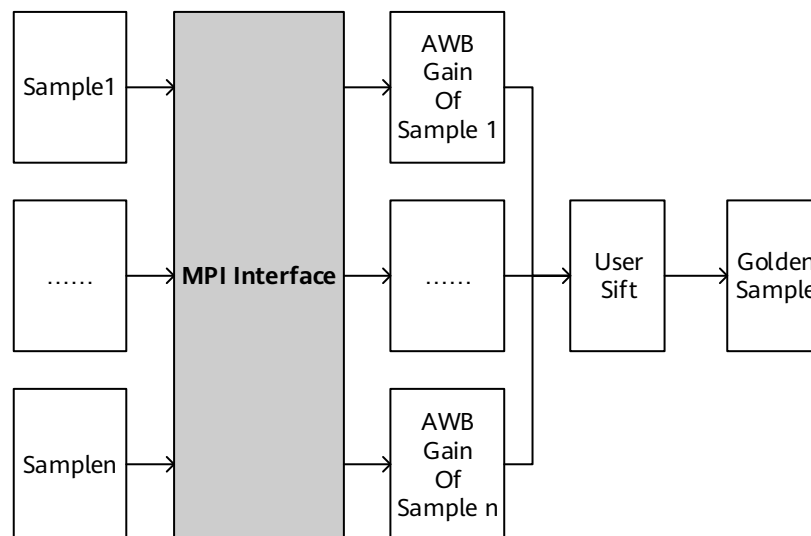
AWB 差异参数的校正过程：

- 步骤 1 小规模试产过程中进行：取若干台样机，分别放置到量产标定环境（色温范围 4500K 到 6500K，均匀光源，均匀背景），待 AE 稳定后（避免 RGrGbB 任一分量饱和），调用 [HI_MPI_ISP_GetLightboxGain](#) 接口，得到每台样机在量产标定环境的 AWB 增益值（请参考 [ISP_AWB_Calibration_Gain_S](#) 结构体定义）。
- 步骤 2 小规模试产过程中进行：根据步骤 1 中若干台样机在量产标定环境的 AWB 增益，由用户自己确定其中一台机器作为 GoldenSample。推荐基于 GoldenSample 机器完成 AWB, CCM 标定，同时 GoldenSample 作为 AWB 差异参数校准的基准。
- 步骤 3 取 GoldenSample，采集多个色温下 24 色卡 Raw 数据，完成 AWB、CCM 标定流程（标定得到静态白平衡系数、普朗克曲线参数、色温曲线参数、多色温 CCM 矩阵系数等）。
- 步骤 4 将 GoldenSample 标定得到的 AWB、CCM 参数通过 `cmos.c` 文件的 `cmos_get_awb_default` 接口配置。定义 `cmos.c` 文件的 `GOLDEN_RGAIN=GoldenSample 在量产标定环境得到的 u16AvgRgain`，定义 `cmos.c` 文件的 `GOLDEN_BGAIN=GoldenSample 在量产标定环境得到的 u16AvgBgain`。
- 步骤 5 若用户不进行步骤 1、2，则任意选取一台机器作为 GoldenSample，完成步骤 3、4 的操作。
- 步骤 6 量产过程中进行：取任意一台需要矫正的机器作为 Sample，重复步骤 1，得到该台机器在量产标定环境的 AWB 增益。通过 `cmos.c` 中的 `sensor_set_init` 对 Sample 的 AWB 增益进行配置。（请参考 [ISP_INIT_ATTR_S](#) 和 [ISP_AWB_Calibration_Gain_S](#) 结构体定

义。ISP_INIT_ATTR_S.u16SampleRgain=Sample 在量产标定环境得到的 u16AvgRgain, ISP_INIT_ATTR_S.u16SampleBgain=Sample 在量产标定环境得到的 u16AvgBgain)。

步骤 7 AWB 程序将会根据 GoldenSample 机器的 AWB 增益以及 Sample 机器的 AWB 增益实现对 Sample 机器的 AWB 差异参数的校正, 改善机器个体之间的颜色一致性。

图6-16 AWB 小规模试产过程中确定 GoldenSample 流程图



⚠ 注意

如果用户配置了 AWB 个体差异标定结果, ISP 启动时会自动修正 AWB 的静态白平衡系数。ISP 启动后, 请勿通过 MPI 接口修改静态白平衡系数, 否则会覆盖之前的自动修正结果, 导致量产标定失效。

----结束

6.19.2 API 参考

请参考 2.2 小节中的 [HI_MPI_ISP_GetLightboxGain](#)。

6.19.3 数据类型

请参考 2.3 小节中的 [ISP_AWB_Calibration_Gain_S](#)。



6.20 ColorTone

6.20.1 功能描述

ColorTone 提供色调调节接口，用户可通过该接口实现图像颜色偏红、偏绿、偏蓝等喜好调节。

6.20.2 API 参考

- [HI_MPI_ISP_SetColorToneAttr](#): 设置色调的参数。
- [HI_MPI_ISP_GetColorToneAttr](#): 获取色调的参数。

HI_MPI_ISP_SetColorToneAttr

【描述】

设置色调的参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetColorToneAttr(VI_PIPE ViPipe, const ISP\_COLOR\_TONE\_ATTR\_S *pstCTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi Pipe 号	输入
pstCTAttr	色调的参数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【需求】

- 头文件：hi_awb_comm.h、mpi_awb.h



- 库文件: libisp.a

【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_GetColorToneAttr](#)

HI_MPI_ISP_GetColorToneAttr

【描述】

获取色调的参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetColorToneAttr(VI_PIPE ViPipe, ISP\_COLOR\_TONE\_ATTR\_S *pstCTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Vi Pipe 号	输入
pstCTAttr	色调的参数。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为 错误码 。

【需求】

- 头文件: hi_awb_comm.h、mpi_awb.h
- 库文件: libisp.a



【注意】

无

【举例】

无

【相关主题】

[HI_MPI_ISP_SetColorToneAttr](#)

6.20.3 数据类型

[ISP_COLOR_TONE_ATTR_S](#)：定义色调调整属性。

ISP_COLOR_TONE_ATTR_S

【说明】

定义色调调整属性。

【定义】

```
typedef struct hiISP_COLOR_TONE_ATTR_S
{
    HI_U16 u16RedCastGain;
    HI_U16 u16GreenCastGain;
    HI_U16 u16BlueCastGain;
} ISP_COLOR_TONE_ATTR_S;
```

【成员】

成员名称	描述
u16RedCastGain	R 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。
u16GreenCastGain	G 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。
u16BlueCastGain	B 通道增益，8bit 小数精度。 取值范围：[0x100, 0x180]。



【注意事项】

ColorTone 支持客户调节固定的颜色风格，与色温无关。

【相关数据类型及接口】

无

6.21 LDCI

6.21.1 功能描述

LDCI 是局域自动对比度增强的简称 (Local Dynamic Contrast Improvement)。该算法采用局域直方图均衡来增强图像局部对比度，提升暗区细节。

6.21.2 API 参考

- [HI_MPI_ISP_SetLDCIAttr](#): 设置 LDCI 属性参数。
- [HI_MPI_ISP_GetLDCIAttr](#): 获取 LDCI 属性参数。

HI_MPI_ISP_SetLDCIAttr

【描述】

设置 LDCI 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetLDCIAttr(VI_PIPE ViPipe, const ISP\_LDCI\_ATTR\_S *pstLDCIAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstLDCIAttr	LDCI 属性参数。	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_GetLDCIAttr](#)

HI_MPI_ISP_GetLDCIAttr

【描述】

获取 LDCI 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetLDCIAttr(VI_PIPE ViPipe, ISP\_LDCI\_ATTR\_S *pstLDCIAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstLDCIAttr	LDCI 属性参数。	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_SetLDCIAttr](#)

6.21.3 数据类型

- [ISP_LDCI_GAUSS_COEF_ATTR_S](#)：定义 GAUSS 曲线属性参数。
- [ISP_LDCI_HE_WGT_ATTR_S](#)：定义直方图均衡权重属性参数。
- [ISP_LDCI_MANUAL_ATTR_S](#)：定义 LDCI 手动模式属性参数。
- [ISP_LDCI_AUTO_ATTR_S](#)：定义 LDCI 自动模式属性参数。
- [ISP_LDCI_ATTR_S](#)：定义 LDCI 属性参数。

ISP_LDCI_GAUSS_COEF_ATTR_S

【说明】

定义 GAUSS 分布属性参数。

【定义】

```
typedef struct hiISP_LDCI_GAUSS_COEF_ATTR_S
{
    HI_U8    u8Wgt;
    HI_U8    u8Sigma;
```



```
HI_U8    u8Mean;  
}ISP_LDCI_GAUSS_COEF_ATTR_S;
```

【成员】

成员名称	描述
u8Wgt	Gauss 分布曲线权重 Hi3559AV100/Hi3519AV100 取值范围: [0x0, 0x80] Hi3516CV500/Hi3559V200/Hi3516EV200/ Hi3516EV300 取值范围: [0x0, 0xFF]
u8Sigma	Gauss 分布曲线标准差, 取值范围: [0x1, 0xFF]。
u8Mean	Gauss 分布曲线数学期望, 取值范围: [0x0, 0xFF]。

【注意事项】

无

【相关数据类型及接口】

无

ISP_LDCI_HE_WGT_ATTR_S

【说明】

定义直方图均衡与原图融合权重属性参数。

【定义】

```
typedef struct hiISP_LDCI_HE_WGT_ATTR_S  
{  
    ISP_LDCI_GAUSS_COEF_ATTR_S  stHePosWgt;  
    ISP_LDCI_GAUSS_COEF_ATTR_S  stHeNegWgt;  
} ISP_LDCI_HE_WGT_ATTR_S;
```

【成员】



成员名称	描述
stHePosWgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。
stHeNegWgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制减暗程度；曲线呈高斯分布，横坐标为亮度。

【注意事项】

无

【相关数据类型及接口】

无

ISP_LDCI_MANUAL_ATTR_S

【说明】

定义 LDCI 手动模式属性参数。

【定义】

```
typedef struct hiISP_LDCI_MANUAL_ATTR_S
{
    ISP_LDCI_HE_WGT_ATTR_S    stHeWgt;
    HI_U16                    u16BlcCtrl;
}ISP_LDCI_MANUAL_ATTR_S;
```

【成员】

成员名称	描述
stHeWgt	手动模式下控制直方图均衡结果与原图融合权重曲线，可根据原图亮度分开控制提亮减暗程度；曲线呈高斯分布，横坐标为亮度。
u16BlcCtrl	手动模式下暗区增益控制强度，用于抑制暗区噪声，取值范围：[0x0, 0x1FF]。

【注意事项】



无

【相关数据类型及接口】

无

ISP_LDCI_AUTO_ATTR_S

【说明】

定义 LDCI 自动模式下属性参数。

【定义】

```
typedef struct hiISP_LDCI_AUTO_ATTR_S
{
    ISP_LDCI_HE_WGT_ATTR_S astHeWgt[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16                  au16BlcCtrl[ISP_AUTO_ISO_STRENGTH_NUM];
}ISP_LDCI_AUTO_ATTR_S;
```

【成员】

成员名称	描述
astHeWgt[ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下控制直方图均衡结果与原图融合权重曲线，可根据原图亮度分开控制提亮减暗程度；曲线呈高斯分布，横坐标为亮度。
au16BlcCtrl[ISP_AUTO_ISO_STRENGTH_NUM]	自动模式下暗区增益控制强度，用于抑制暗区噪声，取值范围：[0x0, 0x1FF]。

【注意事项】

无

【相关数据类型及接口】

无

ISP_LDCI_ATTR_S

【说明】



定义 LDCI 属性参数。

【定义】

```
typedef struct hiISP_LDCI_ATTR_S
{
    HI_BOOL          bEnable;
    HI_U8            u8GaussLPFSigma;
    ISP_OP_TYPE_E    enOpType;
    ISP_LDCI_MANUAL_ATTR_S stManual;
    ISP_LDCI_AUTO_ATTR_S stAuto;
    HI_U16           u16TprIncrCoef;
    HI_U16           u16TprDecrCoef;
} ISP_LDCI_ATTR_S;
```

【成员】

成员名称	描述
bEnable	LDCI 功能使能开关。
u8GaussLPFSigma	LPF 滤波强度，其值越小说明 LDCI 作用越局域，取值范围： [0x1, 0xFF]。
enOpType	LDCI 工作类型。 OP_TYPE_AUTO：自动； OP_TYPE_MANUAL：手动。 默认值为 OP_TYPE_AUTO。
stManual	LDCI 手动参数。
stAuto	LDCI 自动参数。
u16TprIncrCoef	画面由暗到亮变化时域滤波系数，值越大，当前帧像素比重越大，取值范围：[0x0, 0x100]。
u16TprDecrCoef	画面由亮到暗变化时域滤波系数，值越大，当前帧像素比重越大，取值范围：[0x0, 0x100]。

【注意事项】



无

【相关数据类型及接口】

无

6.22 Radial Crop

6.22.1 功能描述

WDR 模式下，对于鱼眼镜头，DRC 会将四周暗处拉亮，最后呈现出的图像的整体效果不是很好，因此新引入一个算法，将设定半径之外的地方直接拉黑掉。由于半径之外的数据为 0，带黑电平 FE 统计信息可以以此为根据对半径以外的点进行舍弃。

Hi3516CV500/Hi3559V200 不支持该功能，Hi3516EV300 仅在线性下支持该功能。在 Hi3516EV300 中，如果在线性模式下使能了此功能，并通过模式切换流程切换到 WDR 模式，会导致合成后图的图像异常。

6.22.2 API 参考

- [HI_MPI_ISP_SetRcAttr](#)：设定 Radial Crop 属性。
- [HI_MPI_ISP_GetRcAttr](#)：获取 Radial Crop 属性。

HI_MPI_ISP_SetRcAttr

【描述】

设定 Radial Crop 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetRcAttr(VI_PIPE ViPipe, const ISP\_RC\_ATTR\_S *pstRcAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstRcAttr	Radial Crop 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】

Hi3516CV500/Hi3559V200 不支持。

【举例】

无。

【相关主题】

[HI_MPI_ISP_GetRcAttr](#)

HI_MPI_ISP_GetRcAttr

【描述】

获取 Radial Crop 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetRcAttr(VI_PIPE ViPipe, ISP\_RC\_ATTR\_S *pstRcAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstRcAttr	Radial Crop 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】

Hi3516CV500/Hi3559V200 不支持。

【举例】

无。

【相关主题】

[HI_MPI_ISP_SetRcAttr](#)

6.22.3 数据类型

[ISP_RC_ATTR_S](#)：定义 ISP Radial Crop 属性。

ISP_RC_ATTR_S

【说明】

定义 ISP Radial Crop 属性。

【定义】

```
typedef struct hiISP_RC_ATTR_S
{
    HI_BOOL bEnable;
    POINT_S stCenterCoor;
    HI_U32 u32Radius;
}ISP_RC_ATTR_S;
```

【成员】



成员名称	描述	取值范围	默认值
bEnable	使能 Radial Crop 功能。	[0,1] 0:禁止; 1:使能	0
stCenterCoor	中心点的坐标配置信息	s32X 取值范围: [0,u32Width -1] s32Y 取值范围: [0,u32Height -1]	s32X 默认为: u32Width/2 s32Y 默认为: u32Height/2
u32Radius	拉黑的半径, 到中心点的距离大于半径的区域直接拉黑	[0, $\sqrt{(u32Width)^2 + (u32Height)^2}$]	$\sqrt{(stCenter.s32X)^2 + (stCenter.s32Y)^2}$

【注意事项】

中心点的坐标以及半径的配置值可以直接采用 fisheye 算法的标定结果, 分辨率切换之后, 对应的中心点左边与半径也要随之更新, 以避免图像的不正常, Hi3516CV500/Hi3559V200 不支持。

【相关数据类型及接口】

- [HI_MPI_ISP_SetRcAttr](#)
- [HI_MPI_ISP_GetRcAttr](#)

6.23 CSC

6.23.1 功能描述

色彩空间转换模块(Color Space Conversion), 用来将 RGB 信号转换成符合标准的 YUV 信号

6.23.2 API 参考

- [HI_MPI_ISP_SetCSCAttr](#): 设定 CSC (色彩空间转换) 模块属性。
- [HI_MPI_ISP_GetCSCAttr](#): 获取 CSC (色彩空间转换) 模块属性。

HI_MPI_ISP_SetCSCAttr

【描述】



设定 CSC（色彩空间转换）属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetCSCAttr(VI_PIPE ViPipe, const ISP\_CSC\_ATTR\_S *pstCSCAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstCSCAttr	CSC 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_GetCSCAttr](#)

HI_MPI_ISP_GetCSCAttr

【描述】

获取 CSC（色彩空间转换）属性。



【语法】

```
HI_S32 HI_MPI_ISP_GetCSCAttr(VI_PIPE ViPipe, ISP\_CSC\_ATTR\_S *pstCSCAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstCSCAttr	CSC 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_SetCSCAttr](#)

6.23.3 数据类型

[ISP_CSC_ATTR_S](#)：定义 ISP 色彩空间转换模块属性。

ISP_CSC_ATTR_S

【说明】



定义 ISP 色彩空间转换模块属性。

【定义】

```
typedef struct hiISP_CSC_ATTR_S
{
    HI_BOOL bEnable;
    COLOR_GAMUT_E enColorGamut;
    HI_U8 u8Hue;
    HI_U8 u8Luma;
    HI_U8 u8Contr;
    HI_U8 u8Satu;
    HI_BOOL bLimitedRangeEn;
    HI_BOOL bExtCscEn;
    HI_BOOL bCtModeEn;
    CSC_MATRX_S stCscMagtrx;
} ISP_CSC_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	使能 CSC 功能。	[0,1] 0: 禁止; 1: 使能。	1
enColorGamut	色彩空间类型枚举	[0,3]	COLOR_GAMUT_BT709
u8Hue	目标 YUV 空间色调	[0,100]	50
u8Luma	目标 YUV 空间亮度	[0,100]	50
u8Contr	目标 YUV 空间对比度	[0,100]	50
u8Satu	目标 YUV 空间饱和度	[0,100]	50
bLimitedRangeEn	目标 YUV 空间动态范围选择	[0,1] 0: 输出为 Full Range; 1: 输出为 Limited Range。	0



成员名称	描述	取值范围	默认值
bExtCscEn	选择亮度调整模式（与 u8Luma 配合）	[0, 1] 1:表示 u8Luma 可以调节更宽的亮度变化	1
bCtModeEn	选择输出的对比度调整模式	[0, 1] 1:改变 u8Contr 画面亮度不会变化	1
stCscMagtrx	色彩空间转换矩阵系数	--	--

【注意事项】

- enColorGamut 中可选择 COLOR_GAMUT_BT601, COLOR_GAMUT_BT709, COLOR_GAMUT_BT2020, COLOR_GAMUT_USER 四种模式。其中 COLOR_GAMUT_BT2020 模式仅在 Hi3559AV100 芯片中支持。在选择 COLOR_GAMUT_USER 模式下, 可根据实际情况自由设定 stCscMagtrx 的参数来进行色彩空间转换。
- 如果将 u8Contr 值调小 (20 左右) 时, 在平坦区域容易出现等高线, 出现的原因是因为 u8Contr 值调小后, 会影响画面的动态范围, 导致原本 0~255 (或者 10bit 0~1023) 的动态范围缩小到原来的 20%左右, 因此在平坦区域如果有亮度变化, 则非常容易产生量化误差出现等高线。

【相关数据类型及接口】

- [HI_MPI_ISP_SetCSCAttr](#)
- [HI_MPI_ISP_GetCSCAttr](#)

CSC_MATRX_S

【说明】

定义色彩空间转换矩阵系数。

【定义】

```
typedef struct hiCSC_MATRX_S
{
    HI_S16 as16CSCIdc[ISP_CSC_DC_NUM];
    HI_S16 as16CSCOdC[ISP_CSC_DC_NUM];
}
```



```
HI_S16 as16CSCCoef[ISP_CSC_COEF_NUM];  
}CSC_MATRX_S;
```



说明

宏定义:

```
#define ISP_CSC_DC_NUM      3  
#define ISP_CSC_COEF_NUM   9
```

【成员】

成员名称	描述
as16CSCIdc[ISP_CSC_DC_NUM]	取值范围: [-1024,1023] CSC 转换矩阵, 输入直流分量
as16CSCOdC[ISP_CSC_DC_NUM]	取值范围: [-1024,1023] CSC 转换矩阵, 输出直流分量
as16CSCCoef[ISP_CSC_COEF_NUM]	取值范围: [-4096, 4095] CSC 转换矩阵, 3x3 系数

【注意事项】

无

【相关数据类型及接口】

[ISP_CSC_ATTR_S](#)

6.24 CLUT

6.24.1 功能描述

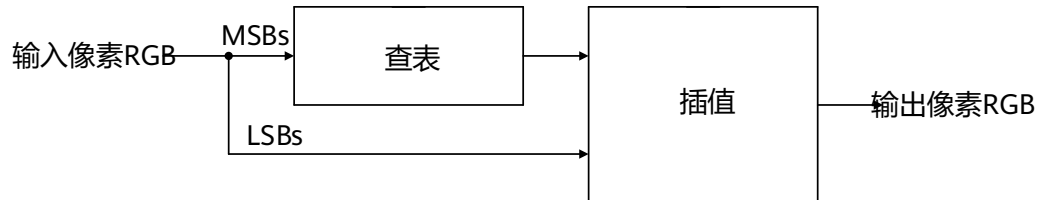
COLOR_3D_LUT 是利用线性 RGB 空间 27x27x27 或 17x17x17 分格的 3D LUT 实现复杂的颜色调整操作, 比如亮度的调整, 饱和度的调整, 色调的调整。阴影区域, 中间亮度区域, 高亮区域分别调整。

它将像素的 RGB 信号的 bit 高位用来查找像素点位于表中的位置, 利用 RGB 信号的 bit 低位用来在表的一个方格中插值出精确的偏移量。将偏移量应用于原像素值, 得到调整后的最终像素值, 从而使颜色得到调节。

须知

Hi3516EV200/Hi3516EV300 不支持该功能。

图6-17 COLOR_3D_LUT 示意图



该单元通过查表的方式，获取 R、G 以及 B 分别的调节量 ΔR 、 ΔG 和 ΔB 。

总共需要三类查找表，R、G 和 B 各一张表。

- Hi3559AV100：大小分别为 R：27x27x27x12bit，G：27x27x27x12bit，B：27x27x27x12bit，即 R 的调节量为[-2047,2047]，G 的调节量为[-2047,2047]，B 的调节量为[-2047,2047]。
- Hi3519AV100/Hi3516CV500/Hi3559V200：大小分别为 R：17x17x17x10bit，G：17x17x17x10bit，B：17x17x17x10bit，即 R 的调节量为[-511,511]，G 的调节量为[-511,511]，B 的调节量为[-511,511]。

CLUT 调节量处理的数据是[0, 4095]范围的 12bit 线性 RGB 数据。

6.24.2 API 参考

- [HI_MPI_ISP_SetClutAttr](#)：设置 CLUT 属性。
- [HI_MPI_ISP_GetClutAttr](#)：获取 CLUT 属性。
- [HI_MPI_ISP_SetClutCoeff](#)：设置 CLUT 系数。
- [HI_MPI_ISP_GetClutCoeff](#)：获取 CLUT 系数。

HI_MPI_ISP_SetClutAttr

【描述】

设置 CLUT 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetClutAttr(VI_PIPE ViPipe, const ISP\_CLUT\_ATTR\_S *pstClutAttr);
```



【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstClutAttr	CLUT 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_GetClutAttr](#)

HI_MPI_ISP_GetClutAttr

【描述】

获取 CLUT 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetClutAttr(VI_PIPE ViPipe, ISP\_CLUT\_ATTR\_S *pstClutAttr);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstClutAttr	CLUT 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_SetClutAttr](#)

HI_MPI_ISP_SetClutCoeff

【描述】

设置 CLUT 系数。

【语法】

```
HI_S32 HI_MPI_ISP_SetClutCoeff(VI_PIPE ViPipe, const ISP\_CLUT\_LUT\_S *pstClutLUT);
```

【参数】

参数名称	描述	输入/输出



ViPipe	ViPipe 号。	输入
pstClutLUT	CLUT 系数。	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_GetClutCoeff](#)

HI_MPI_ISP_GetClutCoeff

【描述】

获取 CLUT 系数。

【语法】

```
HI_S32 HI_MPI_ISP_GetClutCoeff(VI_PIPE ViPipe, ISP_CLUT_LUT_S *pstClutLUT);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入



参数名称	描述	输入/输出
pstClutLUT	CLUT 系数。	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_SetClutCoeff](#)

6.24.3 数据类型

- [ISP_CLUT_ATTR_S](#)：CLUT 的增益。
- [ISP_CLUT_LUT_S](#)：CLUT 的查找表。

ISP_CLUT_ATTR_S

【说明】

定义 CLUT 的增益。

【定义】

```
typedef struct hiISP_CLUT_ATTR_S
```



```
{  
    HI_BOOL      bEnable      ;  
    HI_U32       u32GainR     ;  
    HI_U32       u32GainG     ;  
    HI_U32       u32GainB     ;  
} ISP_CLUT_ATTR_S;
```

【成员】

成员名称	描述
bEnable	CLUT 使能, 取值范围: [0,1]
u32GainR	R 通道增益, 取值范围: [0,4095]。
u32GainG	G 通道增益, 取值范围: [0,4095]。
u32GainB	B 通道增益, 取值范围: [0,4095]。

【注意事项】

Hi3559AV100 增益设定为 128 时, 相当于增益是 1.0。

Hi3519AV100/Hi3516CV500/Hi3559V200 增益设定为 128 时, 相当于增益是 1.0。增益和 LUT 查表得出的调整量相乘, 得到实际调整量。实际调整量和输入 RGB 值相加得到输出 RGB 值。Hi3559AV100 建议默认增益为 128,
Hi3519AV100/Hi3516CV500/Hi3559V200 建议默认增益为 512。

【相关数据类型及接口】

无

ISP_CLUT_LUT_S

【说明】

定义 CLUT 的查找表。

【定义】

```
typedef struct hiISP_CLUT_LUT_S  
{  
    HI_U32 au32lut[HI_ISP_CLUT_LUT_LENGTH];  
} ISP_CLUT_LUT_S;
```



说明

宏定义如下：

Hi3559AV100:

```
#define HI_ISP_CLUT_LUT_LENGTH (22167)
```

Hi3519AV100:

```
#define HI_ISP_CLUT_LUT_LENGTH (4913)
```

Hi3516CV500/Hi3559V200:

```
#define HI_ISP_CLUT_LUT_LENGTH (5508)
```

【成员】

成员名称	描述
au32lut	查找表 - Hi3559AV100 取值范围：[0, 4294967295] - Hi3519AV100/Hi3516CV500/Hi3559V200 取值范围：[0, 1073741823] - Hi3516EV200/Hi3516EV300 不支持该接口。

【注意事项】

由于采用了压缩存储的格式，这个表的内容会由 PQTools 生成。一般用户在 PQTools 界面调试后可以生成这个格式的 LUT，不需要关心数据格式。

对于 Hi3559AV100 的用户：

如果用户不使用 PQTools，需要将自行生成的 3D LUT 表加载到芯片。需要做的工作是将 27x27x27x3 的 LUT 放入 8x3 的 LUT，然后将 8x3 的 LUT 紧凑排列填入长度为 22167 的 LUT。

27x27x27x3 的 LUT 每个元素的范围为[-2047, 2047]，使用 12bit 有符号数表示。

将 27x27x27x3 的 LUT 取成 8x3 的 LUT 的过程为：

三维表的下标为偶数时定义为 0，下标为奇数时定义为 1。R[i][j][k]，G[i][j][k]，B[i][j][k]中的 i 代表 r 的分格，j 代表 b 的分格，k 代表 g 的分格，即 ijk 代表三维表的地址坐标，对应的是 rgb 的排列顺序。R[i][j][k]，G[i][j][k]，B[i][j][k]的值是 deltaR，deltaG，deltaB 的值。用 C 代表 RGB，真值表如下所示：



k	j	i	C
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

上图对应的伪代码如下所示：

```
for(k = 0; k < 27; k++)
{
    for(j = 0; j < 27; j++)
    {
        for(i = 0; i < 27; i++)
        {
            switch((i&1) + ((j&1)<<1) + ((k&1)<<2))
            {
                case 0:R0[n0]=R[i][j][k];
                        G0[n0]=G[i][j][k];
                        B0[n0]=B[i][j][k];
                case 1:R1[n1]=R[i][j][k];
                        G1[n1]=G[i][j][k];
                        B1[n1]=B[i][j][k];
                case 2:R2[n2]=R[i][j][k];
                        G2[n2]=G[i][j][k];
                        B2[n2]=B[i][j][k];
                case 3:R3[n3]=R[i][j][k];
                        G3[n3]=G[i][j][k];
                        B3[n3]=B[i][j][k];
                case 4:R4[n4]=R[i][j][k];
                        G4[n4]=G[i][j][k];
                        B4[n4]=B[i][j][k];
                case 5:R5[n5]=R[i][j][k];
                        G5[n5]=G[i][j][k];
                        B5[n5]=B[i][j][k];
                case 6:R6[n6]=R[i][j][k];
```



```
        G6[n6]=G[i][j][k];  
        B6[n6]=B[i][j][k];  
    case 7:R7[n7]=R[i][j][k];  
        G7[n7]=G[i][j][k];  
        B7[n7]=B[i][j][k];  
    }  
}  
}  
}
```

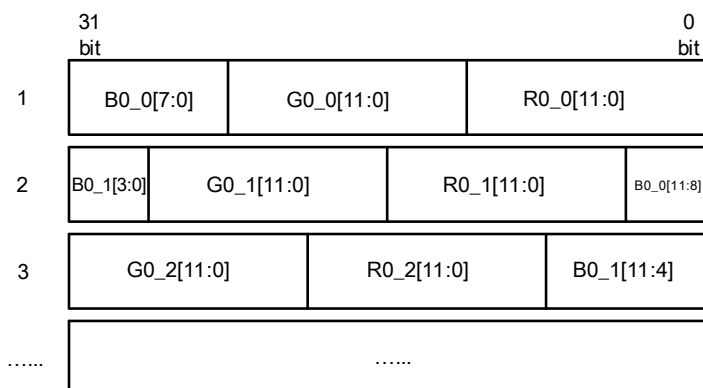
用 C 代表 RGB，取成的 8 个 LUT 分别为 c0[2744]、c1[2548]、c2[2548]、c3[2366]、c4[2548]、c5[2366]、c6[2366]、c7[2197]。

下面的描述用 _0 代表一维数组 c0[2744]第 0 个数据，_1 代表一维数组 c0[2744]第 1 个数据。

8x3 的 LUT 按照每 8x3 个数据紧凑排列到 9 个 32bit 数据内的格式定义为：

```
{B0_0[7:0],G0_0,R0_0},  
{B0_1[3:0],G0_1,R0_1,B0_0[11:8]},  
{G0_2,R0_2,B0_1[11:4]},  
{G0_3[7:0],R0_3,B0_2},  
{G0_4[3:0],R0_4,B0_3,G0_3[11:8]},  
{R0_5,B0_4,G0_4[11:4]},  
{R0_6[7:0],B0_5,G0_5},  
{R0_7[3:0],B0_6,G0_6,R0_6[11:8]},  
{B0_7,G0_7,R0_7[11:4]}
```

上述格式可视化如下图所示：



8x3 的 LUT 连续存储成一个长度为 22167 的一维数组。

c0[2744]在连续存储的时候占用空间 $2744/8*9=3087$ 。

c1[2548]占用空间 $\lceil 2548/8 \rceil *9=2871$ ，最后不足 9 个需要补足 9 个。

c2[2548]占用空间 2871，c3[2366]占用空间 2664，c4[2548]占用空间 2871，
c5[2366]占用空间 2664，c6[2366]占用空间 2664，c7[2197]占用空间 2475。

对于 Hi3519AV100 的用户：

如果用户不使用 PQTools，需要将自行生成的 3D LUT 表加载到芯片。需要做的工作是将 17x17x17x3 的 LUT 放入 8x3 的 LUT，然后将 8x3 的 LUT 紧凑排列填入长度为 4913 的 LUT。

17x17x17x3 的 LUT 每个元素的范围为[-511, 511]，使用 10bit 有符号数表示。

将 17x17x17x3 的 LUT 取成 8x3 的 LUT 的过程为：

三维表的下标为偶数时定义为 0，下标为奇数时定义为 1。R[i][j][k]，G[i][j][k]，B[i][j][k]中的 i 代表 r 的分格，j 代表 b 的分格，k 代表 g 的分格，即 ijk 代表三维表的地址坐标，对应的是 rgb 的排列顺序。R[i][j][k]，G[i][j][k]，B[i][j][k]的值是 deltaR，deltaG，deltaB 的值。用 C 代表 RGB，真值表如下所示：

k	j	i	C
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7



上图对应的伪代码如下所示：

```
for(k = 0; k < 17; k++)
{
    for(j = 0; j < 17; j++)
    {
        for(i = 0; i < 17; i++)
        {
            switch((i&1) + ((j&1)<<1) + ((k&1)<<2))
            {
                case 0:R0[n0]=R[i][j][k];
                    G0[n0]=G[i][j][k];
                    B0[n0]=B[i][j][k];
                case 1:R1[n1]=R[i][j][k];
                    G1[n1]=G[i][j][k];
                    B1[n1]=B[i][j][k];
                case 2:R2[n2]=R[i][j][k];
                    G2[n2]=G[i][j][k];
                    B2[n2]=B[i][j][k];
                case 3:R3[n3]=R[i][j][k];
                    G3[n3]=G[i][j][k];
                    B3[n3]=B[i][j][k];
                case 4:R4[n4]=R[i][j][k];
                    G4[n4]=G[i][j][k];
                    B4[n4]=B[i][j][k];
                case 5:R5[n5]=R[i][j][k];
                    G5[n5]=G[i][j][k];
                    B5[n5]=B[i][j][k];
                case 6:R6[n6]=R[i][j][k];
                    G6[n6]=G[i][j][k];
                    B6[n6]=B[i][j][k];
                case 7:R7[n7]=R[i][j][k];
                    G7[n7]=G[i][j][k];
                    B7[n7]=B[i][j][k];
            }
        }
    }
}
```



```
}

```

用 C 代表 RGB，取成的 8 个 LUT 分别为 c0[729]、c1[648]、c2[648]、c3[576]、c4[648]、c5[576]、c6[576]、c7[512]。

下面的描述用 _0 代表一维数组 c0[729]第 0 个数据，_1 代表一维数组 c0[729]第 1 个数据。

8x3 的 LUT 按照每 3 个数据紧凑排列到 1 个 32bit 数据内的格式定义为：

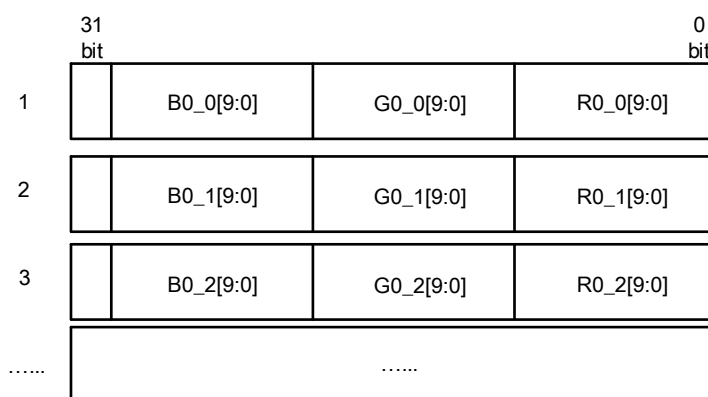
```
{2'b00,B0_0,G0_0,R0_0},
```

```
{2'b00,B0_1,G0_1,R0_1},
```

```
{2'b00,B0_2,G0_2,R0_2},
```

```
{.....},
```

上述格式可视化如下图所示：



8x3 的 LUT 连续存储成一个长度为 4913 的一维数组。

对于 Hi3516CV500/Hi3559V200 的用户：

如果用户不使用 PQTools，需要将自行生成的 3D LUT 表加载到芯片。需要做的工作是将 17x17x17x3 的 LUT 放入 8x3 的 LUT，然后将 8x3 的 LUT 紧凑排列填入长度为 5508 的 LUT。

17x17x17x3 的 LUT 每个元素的范围为[-511, 511]，使用 10bit 有符号数表示。

将 17x17x17x3 的 LUT 取成 8x3 的 LUT 的过程为：

三维表的下标为偶数时定义为 0，下标为奇数时定义为 1。R[i][j][k]，G[i][j][k]，B[i][j][k]中的 i 代表 r 的分格，j 代表 b 的分格，k 代表 g 的分格，即 ijk 代表三维表的



地址坐标，对应的是 rgb 的排列顺序。R[i][j][k]，G[i][j][k]，B[i][j][k]的值是 deltaR，deltaG，deltaB 的值。用 C 代表 RGB，真值表如下所示：

k	j	i	C
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

上图对应的伪代码如下所示：

```
for(k = 0; k < 17; k++)
{
    for(j = 0; j < 17; j++)
    {
        for(i = 0; i < 17; i++)
        {
            switch((i&1) + ((j&1)<<1) + ((k&1)<<2))
            {
                case 0:R0[n0]=R[i][j][k];
                        G0[n0]=G[i][j][k];
                        B0[n0]=B[i][j][k];
                case 1:R1[n1]=R[i][j][k];
                        G1[n1]=G[i][j][k];
                        B1[n1]=B[i][j][k];
                case 2:R2[n2]=R[i][j][k];
                        G2[n2]=G[i][j][k];
                        B2[n2]=B[i][j][k];
                case 3:R3[n3]=R[i][j][k];
                        G3[n3]=G[i][j][k];
                        B3[n3]=B[i][j][k];
                case 4:R4[n4]=R[i][j][k];
                        G4[n4]=G[i][j][k];
                        B4[n4]=B[i][j][k];
                case 5:R5[n5]=R[i][j][k];
                        G5[n5]=G[i][j][k];
```



```
        B5[n5]=B[i][j][k];  
        case 6:R6[n6]=R[i][j][k];  
        G6[n6]=G[i][j][k];  
        B6[n6]=B[i][j][k];  
        case 7:R7[n7]=R[i][j][k];  
        G7[n7]=G[i][j][k];  
        B7[n7]=B[i][j][k];  
    }  
    }  
    }  
    }
```

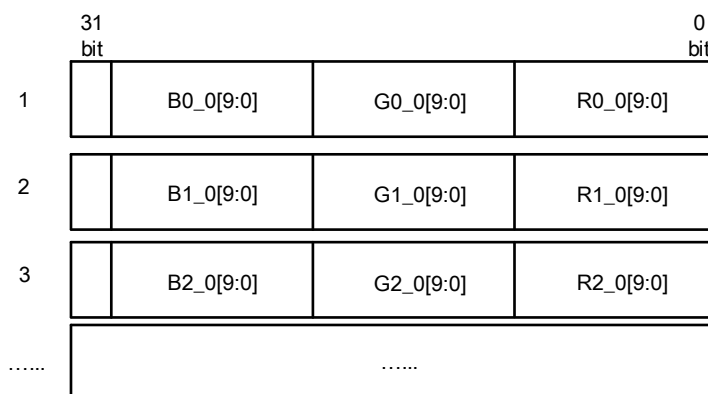
用 C 代表 RGB，取成的 8 个 LUT 分别为 c0[729]、c1[648]、c2[648]、c3[576]、c4[648]、c5[576]、c6[576]、c7[512]。

下面的描述用 _0 代表一维数组 c0[729]第 0 个数据，_1 代表一维数组 c0[729]第 1 个数据。

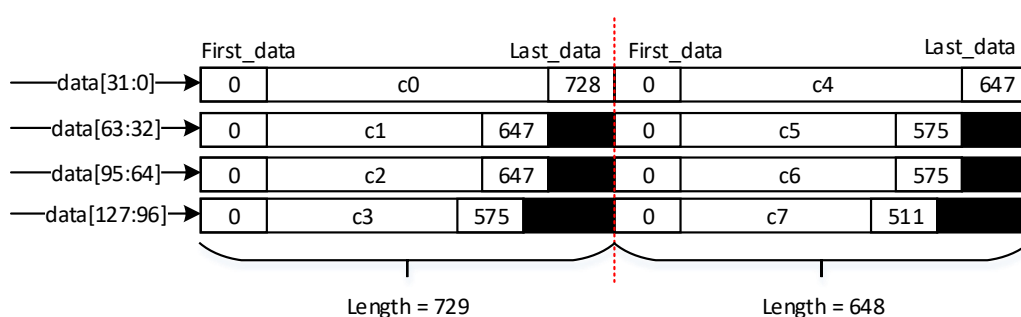
8x3 的 LUT 按照每 3 个数据紧凑排列到 1 个 32bit 数据内的格式定义为：

```
{2'b00,B0_0,G0_0,R0_0},  
{2'b00,B1_0,G1_0,R1_0},  
{2'b00,B2_0,G2_0,R2_0},  
{2'b00,B3_0,G3_0,R3_0},  
{2'b00,B0_1,G0_1,R0_1},  
{2'b00,B1_1,G1_1,R1_1},  
{2'b00,B2_1,G2_1,R2_1},  
{2'b00,B3_1,G3_1,R3_1},  
{.....},
```

上述格式可视化如下图所示：



整体看如下图所示：



8x3 的 LUT 按照 c0[729] 占用空间 $729 \times 4 = 2916$ 和 c4[648] 占用空间 $648 \times 4 = 2592$ 连续存储成一个长度为 5508 的一维数组。

【相关数据类型及接口】

无

6.25 PreLogLUT

6.25.1 功能描述

PreLogLUT 采用 Log 曲线对 RAW 数据进行压缩存储，在 LOG 模式下以记录更多的亮部和暗部细节，简单来说就是将画面中的大范围的高光以及暗部信息压缩到 RAW 中，形成一种对比度低且包含丰富动态范围的画面。采用 PreLogLUT 压缩的 RAW 数据画面动态范围太高，需要对应的 LUT 进行处理，保留较高动态范围的同时，还原出正确的对比度以及颜色信息。该模块将 16bit 数据压缩至 10bit，采用 1025 点 LUT 进行线



性插值，用户可以根据自己的需求在 [ISP_CMOS_PRELOGLUT_S](#) 中自定义 LOG 曲线。
Hi3516CV500/Hi3559V200 不支持该功能。

6.25.2 API 参考

- [HI_MPI_ISP_SetPreLogLUTAttr](#): 设置 PreLogLUT 属性参数。
- [HI_MPI_ISP_GetPreLogLUTAttr](#): 获取 PreLogLUT 属性参数。

HI_MPI_ISP_SetPreLogLUTAttr

【描述】

设置 PreLogLUT 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetPreLogLUTAttr(VI_PIPE ViPipe, const ISP\_PRELOGLUT\_ATTR\_S
*pstPreLogLUTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstPreLogLUTAttr	pstPreLogLUTAttr 属性参数。	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】



Hi3516CV500/Hi3559V200 不支持。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetPreLogLUTAttr](#)

HI_MPI_ISP_GetPreLogLUTAttr

【描述】

获取 PreLogLUT 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetPreLogLUTAttr(VI_PIPE ViPipe, ISP\_PRELOGLUT\_ATTR\_S  
*pstPreLogLUTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstPreLogLUTAttr	PreLogLUT 属性参数。	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】



Hi3516CV500/Hi3559V200 不支持。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetPreLogLUTAttr](#)

6.25.3 数据类型

[ISP_PRELOGLUT_ATTR_S](#): 定义 PreLogLUT 属性参数。

ISP_PRELOGLUT_ATTR_S

【说明】

定义 PreLogLUT 属性参数。

【定义】

```
typedef struct hiISP_PRELOGLUT_ATTR_S
{
    HI_BOOL    bEnable;
} ISP_PRELOGLUT_ATTR_S;
```

【成员】

成员名称	描述
bEnable	PreLogLUT 功能使能，取值范围：[0x0, 0x1]。

【注意事项】

Hi3516CV500/Hi3559V200 不支持。

【相关数据类型及接口】

无



6.26 LogLUT

6.26.1 功能描述

LogLUT 采用 Log 曲线对 RAW 数据进行压缩存储，在 LOG 模式下以记录更多的亮部和暗部细节，简单来说就是将画面中的大范围的高光以及暗部信息压缩到 RAW 中，形成一种对比度低且包含丰富动态范围的画面。采用 LogLUT 压缩的 RAW 数据画面动态范围太高，需要对应的 LUT 进行处理，保留较高动态范围的同时，还原出正确的对比度以及颜色信息。该模块将 16bit 数据压缩至 10bit，采用 1025 点 LUT 进行线性插值，用户可以根据自己的需求在 [ISP_CMOS_LOGLUT_S](#) 中自定义 LOG 曲线。

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持该功能。

6.26.2 API 参考

- [HI_MPI_ISP_SetLogLUTAttr](#): 设置 LogLUT 属性参数。
- [HI_MPI_ISP_GetLogLUTAttr](#): 获取 LogLUT 属性参数。

HI_MPI_ISP_SetLogLUTAttr

【描述】

设置 LogLUT 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetLogLUTAttr(VI_PIPE ViPipe, const ISP\_LOGLUT\_ATTR\_S *pstLogLUTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstLogLUTAttr	pstLogLUTAttr 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetLogLUTAttr](#)

HI_MPI_ISP_GetLogLUTAttr

【描述】

获取 LogLUT 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetLogLUTAttr(VI_PIPE ViPipe, ISP\_LOGLUT\_ATTR\_S *pstLogLUTAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstLogLUTAttr	LogLUT 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】

无

【相关主题】

[HI_MPI_ISP_SetLogLUTAttr](#)

6.26.3 数据类型

[ISP_LOGLUT_ATTR_S](#): 定义 LogLUT 属性参数。

ISP_LOGLUT_ATTR_S

【说明】

定义 LogLUT 属性参数。

【定义】

```
typedef struct hiISP_LOGLUT_ATTR_S
{
    HI_BOOL    bEnable;
} ISP_LOGLUT_ATTR_S;
```

【成员】

成员名称	描述
bEnable	LogLUT 功能使能，取值范围：[0x0, 0x1]。

【注意事项】

LogLUT 与 PreLogLUT 实现功能一样，唯一区别是在系统中所处位置不同。
Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。



【相关数据类型及接口】

无

6.27 PreGamma

6.27.1 功能描述

PreGamma 是类似 Gamma 对图像进行亮度空间非线性转换模块，在 DRC 之前对动态范围进行适当压缩。PreGamma 插值点数为 257，精度为 20bit，节点之间的图像像素值使用线性插值生成。

Hi3516EV200/Hi3516EV300 中，PreGamma 和 DRC 存在耦合，不是串行关系，在效果上表现为：

- PreGamma 必须在 DRC 模块使能时才有效果，并且其效果会受 DRC 强度的影响，DRC 强度越小，PreGamma 效果也越弱；
- PreGamma 调成斜率为 0 的直线时，由于 DRC 模块的影响，画面并不是全灰的。

6.27.2 API 参考

- [HI_MPI_ISP_SetPreGammaAttr](#)：设置 PreGamma 属性参数。
- [HI_MPI_ISP_GetPreGammaAttr](#)：获取 PreGamma 属性参数。

HI_MPI_ISP_SetPreGammaAttr

【描述】

设置 PreGamma 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetPreGammaAttr(VI_PIPE ViPipe, const ISP_PREGAMMA_ATTR_S  
*pstPreGammaAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入



pstPreGammaAttr	PreGamma 属性参数。	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_GetPreGammaAttr](#)

HI_MPI_ISP_GetPreGammaAttr

【描述】

获取 PreGamma 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetPreGammaAttr(VI_PIPE ViPipe,  
ISP\_PREGAMMA\_ATTR\_S*pstPreGammaAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出



pstPreGammaAttr	PreGamma 属性参数。	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_SetPreGammaAttr](#)

6.27.3 数据类型

[ISP_PREGAMMA_ATTR_S](#)：定义 PreGamma 属性参数。

ISP_PREGAMMA_ATTR_S

【说明】

定义 PreGamma 属性参数。

【定义】

```
typedef struct hiISP_PREGAMMA_ATTR_S
{
    HI_BOOL    bEnable;
    HI_U32     au32Table[PREGAMMA\_NODE\_NUM];
}
```



```
} ISP_PREGAMMA_ATTR_S;
```

【成员】

成员名称	描述
bEnable	PreGamma 功能使能，取值范围：[0x0, 0x1]。
au32Table[PREGAMMA_NODE_NUM]	257 段 LUT 表，用来表示 PreGamma 输入输出值大小。 Hi3559AV100/Hi3519AV100 取值范围：[0x0, 0x100000] Hi3516CV500/ Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0xFFFFF]

【注意事项】

PreGamma 不支持乒乓切换，切换过程可能导致图像异常，该属性接口仅用来支持调试；

【相关数据类型及接口】

无

6.28 Radial Shading

6.28.1 功能描述

Radial Shading 模块是采用同心圆方式对镜头阴影现象进行矫正的模块。作为 Mesh Shading 模块功能上的补充，主要用来对鱼眼等特殊镜头进行镜头阴影矫正的功能。为了适配鱼眼镜头，当选择需要校正的成像范围（通过四个 OffCenter 参数调节）小于图像显示的范围时，会对成像范围外的图像进行裁剪。

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持该功能。

6.28.2 API 参考

- [HI_MPI_ISP_SetRadialShadingAttr](#)：设置 Radial Shading 属性参数。



- [HI_MPI_ISP_GetRadialShadingAttr](#): 获取 Radial Shading 属性参数。
- [HI_MPI_ISP_SetRadialShadingLUT](#): 设置 Radial Shading LUT 属性参数。
- [HI_MPI_ISP_GetRadialShadingLUT](#): 获取 Radial Shading LUT 属性参数。

HI_MPI_ISP_SetRadialShadingAttr

【描述】

设置 Radial Shading 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetRadialShadingAttr(VI_PIPE ViPipe, const ISP\_RADIAL\_SHADING\_ATTR\_S *pstRaShadingAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstRaShadingAttr	Radial Shading 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】



无

【相关主题】

[HI_MPI_ISP_GetRadialShadingAttr](#)

HI_MPI_ISP_GetRadialShadingAttr

【描述】

获取 Radial Shading 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetRadialShadingAttr(VI_PIPE ViPipe, ISP\_RADIAL\_SHADING\_ATTR\_S  
*pstRaShadingAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstRaShadingAttr	Radial Shading 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】



无

【相关主题】

[HI_MPI_ISP_SetRadialShadingAttr](#)

HI_MPI_ISP_SetRadialShadingLUT

【描述】

设置 Radial Shading 对应 LUT 参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetRadialShadingLUT(VI_PIPE ViPipe, const ISP_RADIAL_SHADING_LUT_ATTR_S
*pstRaShadingLutAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstRaShadingLutAttr	Radial Shading LUT 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】

无



【相关主题】

[HI_MPI_ISP_GetRadialShadingLUT](#)

HI_MPI_ISP_GetRadialShadingLUT

【描述】

获取 Radial Shading LUT 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetRadialShadingLUT(VI_PIPE ViPipe, ISP\_RADIAL\_SHADING\_LUT\_ATTR\_S
*pstRaShadingLutAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstRaShadingLutAttr	Radial Shading LUT 属性参数。	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】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【举例】

无

【相关主题】



HI_MPI_ISP_SetRadialShadingLUT

6.28.3 数据类型

- [ISP_RADIAL_SHADING_ATTR_S](#): 定义 Radial Shading 属性参数。
- [ISP_RADIAL_SHADING_LUT_ATTR_S](#): 定义 Radial Shading LUT 属性参数。
- [ISP_RADIAL_SHADING_GAIN_LUT_S](#): 定义和传递 Radial Shading 不同光源及不同通道下的 Gain 值。

ISP_RADIAL_SHADING_ATTR_S

【说明】

定义 Radial Shading 属性参数。

【定义】

```
typedef struct hiISP_RADIAL_SHADING_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16RadialStr;
} ISP_RADIAL_SHADING_ATTR_S;
```

【成员】

成员名称	描述
bEnable	取值范围: [0x0, 0x1]。 RLSC 模块使能。
u16RadialStr	取值范围: [0x0, 0xffff]。 镜头阴影矫正强度, 默认 4096

【注意事项】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【相关数据类型及接口】

无



ISP_RADIAL_SHADING_LUT_ATTR_S

【说明】

定义 Radial Shading LUT 属性参数。

【定义】

```
typedef struct hiISP_RADIAL_SHADING_LUT_ATTR_S
{
    OPERATION_MODE_E enLightMode;
    HI_U16  u16BlendRatio;
    HI_U8   u8LightType1;
    HI_U8   u8LightType2;
    HI_U8   u8RadialScale;
    HI_U16  u16CenterRX;
    HI_U16  u16CenterRY;
    HI_U16  u16CenterGrX;
    HI_U16  u16CenterGrY;
    HI_U16  u16CenterGbX;
    HI_U16  u16CenterGbY;
    HI_U16  u16CenterBX;
    HI_U16  u16CenterBY;
    HI_U16  u16OffCenterR;
    HI_U16  u16OffCenterGr;
    HI_U16  u16OffCenterGb;
    HI_U16  u16OffCenterB;
    ISP_RADIAL_SHADING_GAIN_LUT_S astRLscGainLut[HI_ISP_RLSC_GAIN_LUT_NUM];
} ISP_RADIAL_SHADING_LUT_ATTR_S;
```



说明

枚举类型 OPERATION_MODE_E 定义如下：

```
#define HI_ISP_RLSC_GAIN_LUT_NUM    3
```

```
typedef enum hiOPERATION_MODE_E
{
    OPERATION_MODE_AUTO    = 0,
    OPERATION_MODE_MANUAL = 1,
    OPERATION_MODE_BUTT
} OPERATION_MODE_E;
```

【成员】



成员名称	描述
enLightMode	取值范围：[0x0, 0x1]。 光源选择为自动模式还是手动模式，默认 0x0（自动模式）。
u16BlendRatio	取值范围：[0, 256]。 光源选择为手动模式下，光源 1 在融合时的权重值，默认为 256。
u8LightType1	取值范围：[0, 2]。 光源 1 的选择为第 x 套 LUT 表，默认 0。
u8LightType2	取值范围：[0, 2]。 光源 2 的选择为第 x 套 LUT 表，默认 0。
u8RadialScale	取值范围：[0, 13]。 表示输入 LUT 的精度。具体请参考下表。默认值为 3。
u16CenterRX	取值范围：[0, u32Width-1]。 表示图像的光学中心 R 分量 x 轴坐标，默认取图像 1/2 的位置。
u16CenterRY	取值范围：[0, u32Height-1]。 表示图像的光学中心 R 分量 y 轴坐标，默认取图像 1/2 的位置。
u16CenterGrX	取值范围：[0, u32Width-1]。 表示图像的光学中心 Gr 分量 x 轴坐标，默认取图像 1/2 的位置。
u16CenterGrY	取值范围：[0, u32Height-1]。 表示图像的光学中心 Gr 分量 y 轴坐标，默认取图像 1/2 的位置。
u16CenterGbX	取值范围：[0, u32Width-1]。 表示图像的光学中心 Gb 分量 x 轴坐标，默认取图像 1/2 的位置。
u16CenterGbY	取值范围：[0, u32Height-1]。 表示图像的光学中心 Gb 分量 y 轴坐标，默认取图像 1/2 的位置。
u16CenterBX	取值范围：[0, u32Width-1]。 表示图像的光学中心 B 分量 x 轴坐标，默认取图像 1/2 的位置。



成员名称	描述
u16CenterBY	取值范围：[0, u32Height-1]。 表示图像的光学中心 B 分量 y 轴坐标，默认取图像 1/2 的位置。
u16OffCenterR	取值范围：[0, 65535]。 R 通道与校正范围半径相关的量，默认由中心点计算得到。
u16OffCenterGr	取值范围：[0, 65535]。 Gr 通道与校正范围半径相关的量，默认由中心点计算得到。
u16OffCenterGb	取值范围：[0, 65535]。 Gb 通道与校正范围半径相关的量，默认由中心点计算得到。
u16OffCenterB	取值范围：[0, 65535]。 B 通道与校正范围半径相关的量，默认由中心点计算得到。
astRLscGainLut	表示不同光源下的增益值（LUT 表），每种光源包含 4 个不同通道（R/Gr/Gb/B）的增益，每个通道用 129 x 16bit 的 LUT 表表示。

【注意事项】

- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
- 当选择 LightMode 为 OPERATION_MODE_AUTO 时（自动模式，Radial Shading 默认为自动模式），BlendRatio 光源混合比例、LightType1 与 LightType2 选择光源参数不生效，算法自动从系统中获取当前的白平衡增益，从已有的三张表选择最优的两张增益表进行加权平均，得到最终的增益表进行校正计算。
- 当选择 LightMode 为 OPERATION_MODE_MANUAL 时（手动模式），需要先设置好选择第一组光源的参数 LightType1，选择第二组光源参数 LightType2，第一组光源的混合比例 BlendRatio，算法将选择好的两组光源的增益表进行混合加权，得到最终的增益表进行校正计算。
- 与图像光学中心(CenterX, CenterY)相关的 8 个量，初始化的时候，当选择 Cmos 中的 bValid 为 0 时，Cmos 中配置的参数不生效，Radial Shading 使用默认的参数进行配置，取图像的几何中心，并将各颜色分量的坐标按照 Bayer Pattern 方式来排布。举例，当 Sensor 为 RGGB 排布，中心点为(960,540)时，R 分量为



(960,540)，Gr 分量为(961,540)，Gb 分量为(960,541)，B 分量为(961,541)。建议用户手动配置的情况下也参照这种方式来配置，若中心点都为相同的值，图像的边缘有可能会出现伪彩。

- 图像在半径的平方上对其进行 128 等分，也即是有 129 个增益点，图像在实际中来看就是离中心点越远，同心圆环越密集。假设当前通道的 Index 为 $N(N=0,1,2,3,\dots,128)$ ，(PQ 工具中以 1 开始，计算时请减去 1)，当前 Index 对应的半径为 r ，图像的半径为 R ，例如当前的分辨率为 3840×2160 ，则 $R = \sqrt{1920^2 + 1080^2}$ ，那么 $N = 128 \cdot r^2 / R^2$ ，由此可以计算出 N 对应的半径 r 或者半径 r 对应的 Index。
- 与图像成像的校正范围半径相关的量 OffCenter，计算公式 $\text{OffCenter} = (2^{31}) / R^2$ ，与半径的平方成反比关系，也就是半径越大，OffCenter 越小；当四个颜色分量的校正范围不一致时，在裁剪范围选择时会选择半径最小的那个值。
- astRLscGainLut 中的值默认由 Cmos 中的标定结果读取。可直接修改生效。
- u8RadialScale 与增益精度的对应关系如表 6-5 所示。u8RadialScale 为 0~6 时，小于 1 倍增益的增益配置值不起作用。

表6-5 u8RadialScale 与增益精度对应关系

u8RadialScale	增益表的数据格式	增益取值范围	1 倍增益对应增益配置值
0	1.15	[0,2)	32768
1	2.14	[0,4)	16384
2	3.13	[0,8)	8192
3	4.12	[0,16)	4096
4	5.11	[0,32)	2048
5	6.10	[0,64)	1024
6	7.9	[0,128)	512
7	0.16	[1,2)	0
8	1.15	[1,3)	0
9	2.14	[1,5)	0
10	3.13	[1,9)	0
11	4.12	[1,17)	0
12	5.11	[1,33)	0



u8RadialScale	增益表的数据格式	增益取值范围	1 倍增益对应增益配置值
13	6.10	[1,65)	0

【相关数据类型及接口】

ISP_RADIAL_SHADING_GAIN_LUT_S

ISP_RADIAL_SHADING_GAIN_LUT_S

【说明】

定义和传递 Radial Shading 不同光源及不同通道下的 Gain 值。

【定义】

```
typedef struct hiISP_RADIAL_SHADING_LUT_S
{
    HI_U16  au16RGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16GrGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16GbGain[HI_ISP_RLSC_POINTS];
    HI_U16  au16BGain[HI_ISP_RLSC_POINTS];
} ISP_RADIAL_SHADING_GAIN_LUT_S;
```



说明

宏定义如下：

```
#define HI_ISP_RLSC_POINTS    (129)
```

【成员】

成员名称	描述
au16RGain[HI_ISP_RLSC_POINTS]	取值范围：[0x0, 0xffff]。 表示该光源下的 R 通道 Gain 值。
au16GrGain[HI_ISP_RLSC_POINTS]	取值范围：[0x0, 0xffff]。 表示该光源下的 Gr 通道 Gain 值。
au16GbGain[HI_ISP_RLSC_POINTS]	取值范围：[0x0, 0xffff]。 表示该光源下的 Gb 通道 Gain 值。
au16BGain[HI_ISP_RLSC_POINTS]	取值范围：[0x0, 0xffff]。 表示该光源下的 B 通道 Gain 值。



【注意事项】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持

【相关数据类型及接口】

[ISP_RADIAL_SHADING_LUT_ATTR_S](#)

6.29 获取 ISP 模块虚拟地址

6.29.1 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。

6.29.2 API 参考

[HI_MPI_ISP_GetISPRegAttr](#)：获取 ISP 基地址属性。

HI_MPI_ISP_GetISPRegAttr

【描述】

获取 ISP 基地址属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetISPRegAttr(VI_PIPE ViPipe, ISP\_REG\_ATTR\_S * pstIsPRegAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIsPRegAttr	ISP 模块虚拟地址属性。	输出

【返回值】



返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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

6.29.3 数据类型

[ISP_REG_ATTR_SISP_REG_ATTR_S](#)：定义 ISP 各子模块寄存器的虚拟地址属性。

ISP_REG_ATTR_S

【说明】

定义 ISP 各子模块寄存器的虚拟地址属性。

【定义】

```
typedef struct hiISP_REG_ATTR_S
{
    HI_VOID* pIspExtRegAddr;
    HI_U32 u32IspExtRegSize;
    HI_VOID* pAeExtRegAddr;
    HI_U32 u32AeExtRegSize;
    HI_VOID* pAwbExtRegAddr;
    HI_U32 u32AwbExtRegSize;
} ISP_REG_ATTR_S;
```



【成员】

成员名称	描述
plspExtRegAddr	ISP 外部（虚拟）寄存器模块对应的起始虚拟地址。
u32lspExtRegSize	ISP 外部（虚拟）寄存器模块对应的大小。
pAeExtRegAddr	ISP AE 库对应的起始虚拟地址。
u32AeExtRegSize	ISP AE 库对应的大小。
pAwbExtRegAddr	ISP AWB 库对应的起始虚拟地址。
u32AwbExtRegSize	ISP AWB 库对应的大小。

【注意事项】

无

【相关数据类型及接口】

无

6.30 查询内部状态信息

6.30.1 功能描述

Inner State Information 接口的作用是输出数个与 ISO 相关的参数其逻辑寄存器当前配置的真实值。

用户在调试过程中可以通过 MPI 接口读取这些真实值参数，观察参数强度是否正确配置。通过该接口用户只可以获取参数，而非操作改变该寄存器里的值。

6.30.2 API 参考

[HI_MPI_ISP_QueryInnerStateInfo](#)：获取内部寄存器实际强度配置信息和内部状态。

HI_MPI_ISP_QueryInnerStateInfo

【描述】



获取内部寄存器实际强度配置和内部状态。

【语法】

```
HI_S32 HI_MPI_ISP_QueryInnerStateInfo(VI_PIPE ViPipe, ISP_INNER_STATE_INFO_S  
*pstInnerStateInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstInnerStateInfo	内部寄存器实际配置信息和状态值。	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】

- 当用户切换分辨率或者在线性模式和 WDR 模式之间切换时候，可以调用这个接口来查询切换是否已经完成。
- 分辨率/帧率切换时，只查询 bResSwitchFinish 标记。模式切换时，只查询 bWDRSwitchFinish 标记。
- 业务正常运转时，在不改变 WDR 模式的情况下，合法调用 [HI_MPI_ISP_SetPubAttr](#)，查询到的 bWDRSwitchFinish 始终为 1。

【举例】

无



【相关主题】

无

6.30.3 数据类型

[ISP_INNER_STATE_INFO_S](#): 定义内部寄存器实际配置信息参数。

ISP_INNER_STATE_INFO_S

【说明】

定义内部寄存器实际配置信息参数。

【定义】

```
typedef struct hiISP_INNER_STATE_INFO_S
{
    HI_U16 au16TextureStr[ISP\_SHARPEN\_GAIN\_NUM];
    HI_U16 au16EdgeStr[ISP\_SHARPEN\_GAIN\_NUM];
    HI_U16 u16TextureFreq;
    HI_U16 u16EdgeFreq;
    HI_U8  u8OverShoot;
    HI_U8  u8UnderShoot;
    HI_U8  u8ShootSupStr;
    HI_U8  u8NrLscRatio;
    HI_U16 au16CoarseStr[ISP\_BAYER\_CHN\_NUM];
    HI_U8  au8WDRFrameStr[WDR\_MAX\_FRAME\_NUM];
    HI_U8  au8ChromaStr[ISP\_BAYER\_CHN\_NUM];
    HI_U8  u8FineStr;
    HI_U16 u16CoringWgt;
    HI_U16 u16DeHazeStrengthActual;
    HI_U16 u16DrcStrengthActual;
    HI_U32 u32WDRExpRatioActual[WDR\_MAX\_FRAME\_NUM - 1];
    HI_BOOL bWDRSwitchFinish;
    HI_BOOL bResSwitchFinish;
    HI_U16 au16BLActual[ISP\_BAYER\_CHN\_NUM];
} ISP_INNER_STATE_INFO_S;
```



说明

宏定义如下:

```
#define ISP_BAYER_CHN_NUM      (4)
```



【成员】

成员名称	描述
au16TextureStr[ISP_SHARPEN_GAIN_NUM]	无方向的细节纹理的锐化强度，图像无方向的细节纹理的锐度。 取值范围：[0, 4095]。
au16EdgeStr[ISP_SHARPEN_GAIN_NUM]	带方向的边缘的锐化强度，图像带方向的边缘的锐度。 取值范围：[0, 4095]
u16TextureFreq	图像的无方向细节纹理的增强频段控制。图像的 细节纹理增强的频率。 取值范围：[0, 4095]
u16EdgeFreq	图像的带方向的边缘的增强频段控制。图像 边缘增强的频率。 取值范围：[0, 4095]
u8OverShoot	图像的 overshoot（锐化后的白边白点）的 强度。 取值范围：[0, 127]
u8UnderShoot	图像的 undershoot（锐化后的黑边黑点） 的强度。 取值范围：[0, 127]
u8ShootSupStr	图像锐化后的 overshoot 和 undershoot 的 抑制强度。用于在保证清晰度不明显下降的 前提下，抑制锐化后的图像的 overshoot 和 undershoot 的宽度和幅度。
u8NrLscRatio	NR 参考 LSC 增益比例。 取值范围：[0x0, 0xFF] Hi3516EV200/Hi3516EV300 不支持
au16CoarseStr[ISP_BAYER_CHN_NUM]	线性模式下控制亮度噪声整体去噪强度，四 通道强度独立可调，值越大，亮噪去噪强度 越大。



成员名称	描述
	取值范围：[0, 0x360]
au8WDRFrameStr[WDR_MAX_FRAME_NUM]	WDR 模式下每一融合帧分别对应的全局亮度去噪强度。 取值范围：[0, 0x50]。
au8ChromaStr[ISP_BAYER_CHN_NUM]	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0, 0x3]。 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
u8FineStr	亮噪去除强度，值越大，亮噪去除强度越大。 取值范围：[0x0,0x80]。
u16CoringWgt	随机噪声保留强度，值越大，保留的随机噪声越多。 取值范围：[0x0,0xC80]。
u16DeHazeStrengthActual	去雾强度值
u16DrcStrengthActual	DRC 强度生效值。 Hi3559AV100 取值范围：[0x0, 0xFF] Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 取值范围：[0x0, 0x3FF]
u32WDRExpRatioActual[WDR_MAX_FRAME_NUM - 1]	WDR 曝光比 2to1 时用 u32WDRExpRatioActual[0]表示曝光比 3to1 时用 u32WDRExpRatioActual[0]及 u32WDRExpRatioActual[1]表示曝光比 4to1 时用 u32WDRExpRatioActual[0], u32WDRExpRatioActual[1], u32WDRExpRatioActual[2]分别表示曝光比。



成员名称	描述
bWDRSwitchFinish	WDR 切换是否完成标志。
bResSwitchFinish	分辨率切换是否完成标志。
au16BLActual[ISP_BAYER_CHN_NUM]	实际生效的黑电平的值，分别表示 R、Gr、Gb、B 分量的黑电平。 取值范围：[0x0, 0xFF]

【注意事项】

- sensor-built-in 模式下，查询到的实际生效的黑电平值为 14bits；其他模式下，实际生效的黑电平值均为 12bits。
- Hi3559AV100 BE 分块场景下，由于内部算法限制，Dehaze Manual 模式下用户配置强度值大于实际生效强度值 u16DeHazeStrengthActual。

【相关数据类型及接口】

无。

6.31 DNG

6.31.1 功能描述

DNG (Digital Negative)是 Adobe 开发的一种开放的 **raw image file format**，里面使用的 tag 基本上都定义在 TIFF 或者 TIFF/EP 中，在 DNG Sepcification 中只是定义或者建议了数据的组织方式，颜色空间的转换等等。DNG spec 标准文档来定义和确定具体的 dng 数据格式，对标准文档中定义的标签和规定，根据实际应用场景和客户需求选择必要的最小集合实现。将 raw 数据封装成标准 dng 文件，可以由 dng 解析工具进行解析。

6.31.2 API 参考

- [HI_MPI_ISP_GetDngImageStaticInfo](#)：获取 DNG 静态信息。
- [HI_MPI_ISP_SetDngColorParam](#)：设置 DNG 颜色信息。
- [HI_MPI_ISP_GetDngColorParam](#)：获取 DNG 颜色信息。



HI_MPI_ISP_GetDngImageStaticInfo

【描述】

获取 DNG 静态信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetDngImageStaticInfo(VI_PIPE ViPipe, DNG\_IMAGE\_STATIC\_INFO\_S
*pstDngImageStaticInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号	输入
pstDngImageStaticInfo	DNG 静态信息	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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_SetDngColorParam

【描述】

设置 DNG 颜色信息。

【语法】

```
HI_S32 HI_MPI_ISP_SetDngColorParam(VI_PIPE ViPipe , const ISP_DNG_COLORPARAM_S  
*pstDngColorParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDngColorParam	DNG 颜色信息	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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_GetDngColorParam

【描述】

获取 DNG 颜色信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetDngColorParam(VI_PIPE ViPipe , ISP\_DNG\_COLORPARAM\_S  
*pstDngColorParam);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstDngColorParam	DNG 颜色信息	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。

【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



6.31.3 数据类型

- [DNG_IMAGE_STATIC_INFO_S](#): 定义 DNG 静态格式信息。
- [ISP_DNG_COLORPARAM_S](#): 定义 DNG 白平衡校正系数。
- [DNG_CFALAYOUT_TYPE_E](#): 定义滤色阵列类型。
- [DNG_SRATIONAL_S](#): 定义 DNG 除法结构体。
- [DNG_BLCREPEATDIM_S](#): 定义黑电平像素重复维度。
- [DNG_DEFAULTSCALE_S](#): 定义将图像转换成方形像素默认的缩放系数。
- [DNG_REPEATPATTERN_DIM_S](#): 定义 DNG 颜色分量重复次数。
- [DNG_RAW_FORMAT_S](#): 定义 DNG RAW 格式。
- [ISP_DNG_WBGAIN_S](#): 定义在不同光源下的白平衡增益校正。

DNG_IMAGE_STATIC_INFO_S

【说明】

定义 DNG 静态格式信息。

【定义】

```
typedef struct hiDNG_IMAGE_STATIC_INFO_S
{
    DNG_RAW_FORMAT_S stDngRawFormat;
    DNG_SRATIONAL_S astColorMatrix1[CCM_MATRIX_SIZE];
    DNG_SRATIONAL_S astColorMatrix2[CCM_MATRIX_SIZE];
    DNG_SRATIONAL_S astCameraCalibration1[CCM_MATRIX_SIZE];
    DNG_SRATIONAL_S astCameraCalibration2[CCM_MATRIX_SIZE];
    DNG_SRATIONAL_S astForwadMatrix1[CCM_MATRIX_SIZE];
    DNG_SRATIONAL_S astForwadMatrix2[CCM_MATRIX_SIZE];
    HI_U8 u8CalibrationIlluminant1;
    HI_U8 u8CalibrationIlluminant2;
}DNG_IMAGE_STATIC_INFO_S;
```

【成员】

成员名称	描述
stDngRawFormat	DNG raw 格式



成员名称	描述
astColorMatrix1	在第一组校正光源下, XYZ 空间到参考相机颜色空间的转换矩阵
astColorMatrix2	在第二组校正光源下, XYZ 空间到参考相机颜色空间的第一组转换矩阵
astCameraCalibration1	在第一组校正光源下, 参考相机颜色空间到个体相机颜色空间转换矩阵
astCameraCalibration2	在第二组校正光源下, 参考相机颜色空间到个体相机颜色空间转换矩阵
astForwadMatrix1	在第一组校正光源下, 相机颜色到 XYZ D50 颜色的转换矩阵
astForwadMatrix2	在第二组校正光源下, 相机颜色到 XYZ D50 颜色的转换矩阵
u8CalibrationIlluminant1	校正光源 1
u8CalibrationIlluminant2	校正光源 2

【注意事项】

无

【相关数据类型及接口】

无

ISP_DNG_COLORPARAM_S

【说明】

定义 DNG 白平衡校正系数。

【定义】

```
typedef struct hiISP_DNG_COLORPARAM_S
{
    ISP_DNG_WBGAIN_S stWbGain1;
    ISP_DNG_WBGAIN_S stWbGain2;
```



```
}ISP_DNG_COLORPARAM_S;
```

【成员】

成员名称	描述
stWbGain1	在低色温光源下校正的白平衡系数。
stWbGain2	在高色温光源下校正的白平衡系数。

【注意事项】

无

【相关数据类型及接口】

无

DNG_CFALAYOUT_TYPE_E

【说明】

定义 DNG 滤色阵列类型。

【定义】

```
typedef enum hiDNG_CFALAYOUT_TYPE_E
{
    CFALAYOUT_TYPE_RECTANGULAR = 1,
    CFALAYOUT_TYPE_A,
    CFALAYOUT_TYPE_B,
    CFALAYOUT_TYPE_C,
    CFALAYOUT_TYPE_D,
    CFALAYOUT_TYPE_E,
    CFALAYOUT_TYPE_F,
    CFALAYOUT_TYPE_G,
    CFALAYOUT_TYPE_H,
    CFALAYOUT_TYPE_BUTT
} DNG_CFALAYOUT_TYPE_E;
```

【成员】



成员名称	描述
CFALAYOUT_TYPE_RECTANGULAR	滤色阵列像素点为方形
CFALAYOUT_TYPE_A	滤色阵列 A 类型，暂未支持
CFALAYOUT_TYPE_B	滤色阵列 B 类型，暂未支持
CFALAYOUT_TYPE_C	滤色阵列 C 类型，暂未支持
CFALAYOUT_TYPE_D	滤色阵列 D 类型，暂未支持
CFALAYOUT_TYPE_E	滤色阵列 E 类型，暂未支持
CFALAYOUT_TYPE_F	滤色阵列 F 类型，暂未支持
CFALAYOUT_TYPE_G	G 滤色阵列类型，暂未支持
CFALAYOUT_TYPE_H	H 滤色阵列类型，暂未支持
CFALAYOUT_TYPE_BUTT	滤色阵列类型结束

【注意事项】

无

【相关数据类型及接口】

无

DNG_SRATIONAL_S

【说明】

定义 DNG 除法结构体。

【定义】

```
typedef struct hiDNG_SRATIONAL_S
{
    HI_S32 s32Numerator;
    HI_S32 s32Denominator;
} DNG_SRATIONAL_S;
```

【成员】



成员名称	描述
s32Numerator	分子
s32Denominator	分母

【注意事项】

无

【相关数据类型及接口】

无

DNG_BLCREPEATDIM_S

【说明】

定义黑电平像素重复维度。

【定义】

```
typedef struct hiDNG_BLCREPEATDIM_S
{
    HI_U16 u16BlcRepeatRows;
    HI_U16 u16BlcRepeatCols;
}DNG_BLCREPEATDIM_S;
```

【成员】

成员名称	描述
u16BlcRepeatRows	黑电平在行方向上像素重复次数。
u16BlcRepeatCols	黑电平在列方向上像素重复次数。

【注意事项】

无

【相关数据类型及接口】

无



DNG_DEFAULTSCALE_S

【说明】

定义将图像转换成方形像素默认的缩放系数。

【定义】

```
typedef struct hiDNG_DEFAULTSCALE_S
{
    DNG_RATIONAL_S stDefaultScaleH;
    DNG_RATIONAL_S stDefaultScaleV;
}DNG_DEFAULTSCALE_S;
```

【成员】

成员名称	描述
stDefaultScaleH	将图像转换成方形像素的默认水平缩放系数。
stDefaultScaleV	将图像转换成方形像素的默认垂直缩放系数。

【注意事项】

无

【相关数据类型及接口】

无

DNG_REPEATPATTERN_DIM_S

【说明】

定义 DNG 颜色分量重复次数。

【定义】

```
typedef struct hiDNG_REPEATPATTERN_DIM_S
{
    HI_U16 u16RepeatPatternDimRows;
    HI_U16 u16RepeatPatternDimCols;
}DNG_REPEATPATTERN_DIM_S;
```

【成员】



成员名称	描述
u16RepeatPatternDimRows	颜色分量在行方向上像素重复次数。
u16RepeatPatternDimCols	颜色分量在列方向上像素重复次数。

【注意事项】

无

【相关数据类型及接口】

无

DNG_RAW_FORMAT_S

【说明】

定义 DNG RAW 数据格式。

【定义】

```
typedef struct hiDNG_RAW_FORMAT_S
{
    HI_U8 u8BitsPerSample;
    HI_U8 au8CfaPlaneColor[CFACOLORPLANE];
    DNG_CFALAYOUT_TYPE_E enCfaLayout;
    DNG_BLCREPEATDIM_S stBlcRepeatDim;
    HI_U32 u32WhiteLevel;
    DNG_DEFAULTSCALE_S stDefaultScale;
    DNG_REPEATPATTERNDIM_S stCfaRepeatPatternDim;
    HI_U8 au8CfaPattern[ISP_BAYER_CHN];
}DNG_RAW_FORMAT_S;
```

说明

宏定义如下：

```
#define CFACOLORPLANE    (3)
#define ISP_BAYER_CHN    (4)
```

【成员】



成员名称	描述
u8BitsPerSample	Raw 数据位宽
au8CfaPlaneColor[CFACOLORPLANE]	滤色阵列的颜色分量
enCfaLayout	滤色阵列排列类型
stBlcRepeatDim	黑电平的重叠维度
u32WhiteLevel	白电平
stDefaultScale	默认缩放系数
stCfaRepeatPatternDim	在每个方向上的颜色分量个数
au8CfaPattern[ISP_BAYER_CHN]	滤色阵列的颜色分量排列顺序

【注意事项】

无

【相关数据类型及接口】

无

ISP_DNG_WBGAIN_S

【说明】

定义 DNG 白平衡校正系数。

【定义】

```
typedef struct hiISP_DNG_WBGAIN_S
{
    HI_U16 u16Rgain;
    HI_U16 u16Ggain;
    HI_U16 u16Bgain;
}ISP_DNG_WBGAIN_S;
```

【成员】



成员名称	描述
u16Rgain	R 通道增益
u16Ggain	G 通道增益
u16Bgain	B 通道增益

【注意事项】

无

【相关数据类型及接口】

无

6.32 MeshShading 量产标定工具

6.32.1 功能描述

本模块用于在 Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 芯片上实现对单一光源的在线 Mesh LSC 标定功能。具体调用实例可参考开发包中的 lsc_online_cali 用例。本模块可针对单一光源生成符合 Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3556V200/Hi3516EV200/ Hi3516EV300 芯片中 Mesh LSC 的标定数据。

在使用本模块进行标定过程中，标定环境与离线标定 MLSC 时需要的标定环境相同。

在进行 Mesh LSC 在线标定前，首先需要确定 MeshScale 的大小，具体做法为：在需要标定的一批次镜头中，挑选几个典型的镜头样本，将样本镜头在实验室环境运行量产 Shading 在线标定的用例并预设 MeshScale 的值，例如预设的 MeshScale =1，运行用例的时候，报出以下打印 Please set Mesh scale to 2，表示当前挑选的镜头的 Shading 的倍数大于 4 倍，且小于 8 倍，将打印推荐的 MeshScale=2 重新进行预设，并运行 Mesh LSC 在线标定程序得到 MeshShading Table，将 MeshShading Table 和 MeshScale 导入到 PQ 工具中，开启 Mesh Shading 校正模块，观察画面的四周的镜头阴影区是否得到合理校正且画面整体亮度无异常。如果镜头的 Shading 得到了合理的校正，接下来将实验室环境得到 MeshScale 值，再大规模应用到产线标定中。



以上做法是基于假设同一批次的镜头，Shading 的程度表现差异不大，如果不同批次的镜头，需要在实验室环境重新确定 MeshScale 的大小。

MeshShading 差异的校正过程：

步骤 1 产线上 Shading 校正时，对于消费类 sensor 不同的模式，可以在线抓取其中最大分辨率下对应光源的 RAW(如 5000K 色温)，其余模式下的分辨率，可以根据与最大分辨率的相对位置，调用 [HI_MPI_ISP_MeshShadingCalibration](#) 接口，标定得到对应光源 RAW 的 MeshShading Table 表。

步骤 2 在步骤 1 得到 MeshShading Table 表的基础上，用户调用 [HI_MPI_ISP_SetMeshShadingAttr](#) 接口将生成的 MeshShading Table 表写入到板端的 Flash，以便于下次机器上电能够获取到相关的 Mesh Shading Table 表。

---结束

6.32.2 API 参考

[HI_MPI_ISP_MeshShadingCalibration](#)：获取 Mesh LSC 在线标定结果。

HI_MPI_ISP_MeshShadingCalibration

【描述】

获取 Mesh LSC 在线标定结果。

【语法】

```
HI_S32 HI_MPI_ISP_MeshShadingCalibration(VI_PIPE ViPipe, HI_U16* pu16SrcRaw,  
ISP\_MLSC\_CALIBRATION\_CFG\_S\* pstMLSCCaliCfg, ISP\_MESH\_SHADING\_TABLE\_S\* pstMLSCTable)
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pu16SrcRaw	输入待标定的 Raw 域图像。	输入
pstMLSCCaliCfg	输入 MLSC 在线标定 Cfg 参数。	输入
pstMLSCTable	输出 MLSC 标定结果。	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】

请参考 sample 当中的在线标定 (lsc_online_cali) 对该接口的使用方法，该接口一般不会单独调用。如果需要直接调用，请注意 pu16SrcRaw 是一个指向图像 Raw 数据的指针，接口内部无法对传入数据的长度进行检验，必须在外部分配正确，否则会出现未知错误，分配的空间长度与 pstMLSCCaliCfg 结构体中定义的长和宽相关，请按照下面的示例来分配 Raw 的内存空间：

```
pu16SrcRaw = (HI_U16*)malloc(pstMLSCCaliCfg->u16ImageHeight * pstMLSCCaliCfg->u16ImageWidth * sizeof(HI_U16))
```

【举例】

无

【相关主题】

无

6.32.3 数据类型

- [ISP_MLSC_CALIBRATION_CFG_S](#)：定义 MLSC 在线标定 CFG 参数信息。
- [ISP_MESH_SHADING_TABLE_S](#)：定义 MLSC 标定结果格式。

ISP_MLSC_CALIBRATION_CFG_S

【说明】

定义 MLSC 在线标定 CFG 参数信息。

【定义】

```
typedef struct hiISP_MLSC_CALIBRATION_CFG_S  
{
```



```

    ISP_BAYER_FORMAT_E enBayer;
    ISP_BAYER_RAWBIT_E enRawBit;
    HI_U16  u16ImgHeight;
HI_U16  u16ImgWidth;
    HI_U16  u16DstImgHeight
HI_U16  u16DstImgWidth
HI_U16  u16OffsetX
HI_U16  u16OffsetY;
    HI_U32  u32MeshScale;
    HI_U16  u16BLCOffsetR;
    HI_U16  u16BLCOffsetGr;
    HI_U16  u16BLCOffsetGb;
    HI_U16  u16BLCOffsetB;
}ISP_MLSC_CALIBRATION_CFG_S;

```

【成员】

成员名称	描述	取值范围
enBayer	输入图像的 Bayer 域排布。	[0, 3]
enRawBit	输入图像的 Bit 位宽。	只能配置 8/10/12/14/16
u16ImgHeight	输入图像高度，要求 4 对齐。	[0, 65535]
u16ImgWidth	输入图像宽度，要求 4 对齐。	[0, 65535]
u16DstImgHeight	裁剪标定时目标图像的高度，要求 4 对齐；不需要裁剪时，与 u16ImgHeight 一致	[0, u16ImgHeight]
u16DstImgWidth	裁剪标定时目标图像的宽度，要求 4 对齐；不需要裁剪时，与 u16ImgWidth 一致	[0, u16ImgWidth]
u16OffsetX	裁剪标定时目标图像对于输入图像的偏移值的横坐标，要求 2 对齐；不需要裁剪时，设置为 0	[0, u16ImgWidth - u16DstImgWidth]
u16OffsetY	裁剪标定时目标图像对于输入图像的偏移值的纵坐标，要求 2 对齐；不需	[0, u16ImgHeight - u16DstImgHeight]



成员名称	描述	取值范围
	要裁剪时，设置为 0	
u32MeshScale	表示在 MLSC 矫正表的 Gain 值精度选择。	[0,7]
u16BLCOffsetR	输入图像的 BLC 值，R 分量。	[0, 4095]
u16BLCOffsetGr	输入图像的 BLC 值，GR 分量。	[0, 4095]
u16BLCOffsetGb	输入图像的 BLC 值，GB 分量。	[0, 4095]
u16BLCOffsetB	输入图像的 BLC 值，B 分量。	[0, 4095]

【注意事项】

u32MeshScale 的取值范围为[0, 7]，其分别对应的 Gain 值精度选择如下：

u32MeshScale	对应精度	标定结果范围
0	1.9	0~2
1	2.8	0~4
2	3.7	0~8
3	4.6	0~16
4	0.10	1~2
5	1.9	1~3
6	2.8	1~5
7	3.7	1~9

- Mesh Shading 在线标定时需要减去黑电平，设置各颜色通道的黑电平时要以 12bit 为基准，因此输入 Raw 的比特位宽也一定要相同。
- 当使用裁剪的接口 u16DstImgHeight, u16DstImgWidth, u16OffsetX, u16OffsetY 时，注意偏移值和裁剪图像的宽高不要超出输入图像的宽高。

【相关数据类型及接口】

无



ISP_MESH_SHADING_TABLE_S

【说明】

定义 MLSC 标定结果格式。

【定义】

```
typedef struct hiISP_MESH_SHADING_TABLE_S
{
    HI_U8    u8MeshScale;
    HI_U16   au16XGridWidth[HI_ISP_MLSC_X_HALF_GRID_NUM];
    HI_U16   au16YGridWidth[HI_ISP_MLSC_Y_HALF_GRID_NUM];
    ISP_SHADING_GAIN_LUT_S stLscGainLut;
    ISP_BNR_LSC_GAIN_LUT_S stBNRLscGainLut;
}ISP_MESH_SHADING_TABLE_S;
```

【成员】

成员名称	描述	取值范围
u8MeshScale	表示在 MLSC 矫正表的 Gain 值精度选择	[0, 7]
au16XGridWidth[HI_ISP_MLSC_X_HALF_GRID_NUM]	储存 MeshLSC 每个 Grid 的宽方向的值	[4,u16ImgWidth/4 - 60]
au16YGridWidth[HI_ISP_MLSC_Y_HALF_GRID_NUM]	储存 MeshLSC 每个 Grid 的高方向的值	[4,u16ImgHeight/4 - 60]
stLscGainLut	存储各颜色分量上的 Gain Table, 是 Mesh Shading 表	[0, 1023]
stBNRLscGainLut	存储各颜色分量上的 Gain Table, 是 Radial Shading 表, 给 BNR 的 LSC 提供参考, Hi3559AV100/Hi3519AV100/Hi3516EV200/Hi3516EV300 不支持该数据类型。	[0, 65535]

【注意事项】



无

【相关数据类型及接口】

无

6.33 RadialShading 量产标定工具

6.33.1 功能描述

本模块用于在 Hi3559AV100/Hi3519AV100 芯片上实现对单一光源的在线 Radial LSC 标定功能。具体调用实例可参考开发包中的 rlsc_online_cali 用例。本模块可针对单一光源生成符合 Hi3559AV100/Hi3519AV100 芯片中 Radial LSC 的标定数据。

在使用本模块进行标定过程中，标定环境与离线标定 RLSC 时需要的标定环境相同。

标定步骤与 MeshShading 量产标定工具类似。

6.33.2 API 参考

[HI_MPI_ISP_RadialShadingCalibration](#)：获取 Radial LSC 在线标定结果。

HI_MPI_ISP_RadialShadingCalibration

【描述】

获取 Radial LSC 在线标定结果。

【语法】

```
HI_S32 HI_MPI_ISP_RadialShadingCalibration(VI_PIPE ViPipe, HI_U16* pu16SrcRaw,  
ISP\_RLSC\_CALIBRATION\_CFG\_S\* pstRLSCCaliCfg, ISP\_RADIAL\_SHADING\_TABLE\_S\* pstRLSCTable)
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pu16SrcRaw	输入待标定的 Raw 域图像。	输入
pstRLSCCaliCfg	输入 RLSC 在线标定 Cfg 参数。	输入
pstRLSCTable	输出 RLSC 标定结果。	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】

请参考 sample 当中的在线标定 (rlsc_online_cali) 对该接口的使用方法，该接口一般不会单独调用。如果需要直接调用，请注意 pu16SrcRaw 是一个指向图像 Raw 数据的指针，接口内部无法对传入数据的长度进行检验，必须在外部分配正确，否则会出现未知错误，分配的空间长度与 pstRLSCCaliCfg 结构体中定义的长和宽相关，请按照下面的示例来分配 Raw 的内存空间：

```
pu16SrcRaw = (HI_U16*)malloc(pstRLSCCaliCfg->u16ImageHeight * pstMLSCCaliCfg->u16ImageWidth * sizeof(HI_U16))
```

【举例】

无

【相关主题】

无

6.33.3 数据类型

- [ISP_RLSC_CALIBRATION_CFG_S](#)：定义 RLSC 在线标定 CFG 参数信息。
- [ISP_RLSC_CABLI_TABLE_S](#)：定义 RLSC 标定结果格式。

ISP_RLSC_CALIBRATION_CFG_S

【说明】

定义 RLSC 在线标定 CFG 参数信息。



【定义】

```
typedef struct hiISP_RLSC_CALIBRATION_CFG_S
{
    ISP_BAYER_FORMAT_E enBayer;
    ISP_BAYER_RAWBIT_E enRawBit;
    HI_U16  u16ImgHeight;
    HI_U16  u16ImgWidth;
    HI_U16  u16CenterX;
    HI_U16  u16CenterY;
    HI_U16  u16Radius;
    HI_U32  u32RadialScale;
    HI_U16  u16BLCOffsetR;
    HI_U16  u16BLCOffsetGr;
    HI_U16  u16BLCOffsetGb;
    HI_U16  u16BLCOffsetB;
}ISP_RLSC_CALIBRATION_CFG_S;
```

【成员】

成员名称	描述	取值范围
enBayer	输入图像的 Bayer 域排布。	[0, 3]
enRawBit	输入图像的 Bit 位宽。	只能配置 8/10/12/14/16
u16ImgHeight	输入图像高度，要求 4 对齐。	[0, 65535]
u16ImgWidth	输入图像宽度，要求 4 对齐。	[0, 65535]
u16CenterX	光学中心点的横坐标	[0, 65535]
u16CenterY	光学中心点的纵坐标	[0, 65535]
u16Radius	标定的区域的半径	[0, 65535]
u32RadialScale	表示在 RLSC 矫正表的 Gain 值精度选择。	[0, 13]
u16BLCOffsetR	输入图像的 BLC 值，R 分量。	[0, 4095]
u16BLCOffsetGr	输入图像的 BLC 值，GR 分量。	[0, 4095]



成员名称	描述	取值范围
u16BLCOffsetGb	输入图像的 BLC 值, GB 分量。	[0, 4095]
u16BLCOffsetB	输入图像的 BLC 值, B 分量。	[0, 4095]

【注意事项】

u32RadialScale 的取值范围为[0, 13], 其分别对应的 Gain 值精度选择如下:

u8RadialScale	增益表的数据格式	增益取值范围	1 倍增益对应增益配置值
0	1.15	[0,2)	32768
1	2.14	[0,4)	16384
2	3.13	[0,8)	8192
3	4.12	[0,16)	4096
4	5.11	[0,32)	2048
5	6.10	[0,64)	1024
6	7.9	[0,128)	512
7	0.16	[1,2)	0
8	1.15	[1,3)	0
9	2.14	[1,5)	0
10	3.13	[1,9)	0
11	4.12	[1,17)	0
12	5.11	[1,33)	0
13	6.10	[1,65)	0

Radial Shading 在线标定时需要减去黑电平, 设置各颜色通道的黑电平时要以 12bit 为基准, 因此输入 Raw 的比特位宽也一定要相同。

【相关数据类型及接口】

[HI_MPI_ISP_RadialShadingCalibration](#)



ISP_RADIAL_SHADING_TABLE_S

【说明】

定义 RLSC 标定结果格式。

【定义】

```
typedef struct hiISP_RADIAL_SHADING_TABLE_S
{
    HI_U8    u16RadialScale;
    HI_U16   u16CenterX;
    HI_U16   u16CenterY;
    HI_U16   u16OffCenter;
    ISP\_RADIAL\_SHADING\_GAIN\_LUT\_S stRlscGainLut;
}ISP_RADIAL_SHADING_TABLE_S;
```

【成员】

成员名称	描述	取值范围
u16RadialScale	表示在 RLSC 矫正表的 Gain 值精度选择	[0, 13]
u16CenterX	光学中心点的横坐标	[0, 65535]
u16CenterY	光学中心点的纵坐标	[0, 65535]
u16OffCenter	与 $1/R^2$ 相关的参数	[0, 65535]
stRlscGainLut	存储各颜色分量上的 Gain Table	[0, 65535]

【注意事项】

无

【相关数据类型及接口】

[HI_MPI_ISP_RadialShadingCalibration](#)



6.34 DE

6.34.1 功能描述

改善噪声形态，使得噪声均匀细碎，同时改善边缘毛糙。仅 Hi3516CV500/Hi3559V200 支持该功能。

6.34.2 API 参考

- [HI_MPI_ISP_SetDEAttr](#)：设定 DE 属性。
- [HI_MPI_ISP_GetDEAttr](#)：获取 DE 属性。

HI_MPI_ISP_SetDEAttr

【描述】

设定 DE 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetDEAttr(VI_PIPE ViPipe, const ISP\_DE\_ATTR\_S *pstDEAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstDEAttr	DE 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_GetDEAttr](#)

HI_MPI_ISP_GetDEAttr

【描述】

获取 DE 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDEAttr(VI_PIPE ViPipe, ISP\_DE\_ATTR\_S *pstDEAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstDEAttr	DE 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_SetDEAttr](#)

6.34.3 数据类型

- [ISP_DE_AUTO_ATTR_S](#): 定义 ISP DE 自动参数属性。
- [ISP_DE_MANUAL_ATTR_S](#): 定义 ISP DE 手动参数属性
- [ISP_DE_ATTR_S](#): 定义 ISP DE 属性。

ISP_DE_AUTO_ATTR_S

【说明】

定义 ISP DE 自动参数属性。

【定义】

```
typedef struct hiISP_DE_AUTO_ATTR_S
{
    HI_U16      au16GlobalGain[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16      au16GainLF[ISP_AUTO_ISO_STRENGTH_NUM];
    HI_U16      au16GainHF[ISP_AUTO_ISO_STRENGTH_NUM];
} ISP_DE_AUTO_ATTR_S;
```

【成员】

成员名称	描述	取值范围
au16GlobalGain[16]	自动模式下 DE 全局叠加强度增益	[0,256]
au16GainLF[16]	自动模式下 DE 低频叠加强度增益	[0,32]
au16GainHF[16]	自动模式下 DE 高频叠加强度增益	[0,32]



【注意事项】

仅 Hi3516CV500/Hi3559V200 中支持该结构体参数的配置与获取。

【相关数据类型及接口】

[ISP_DE_ATTR_S](#)

ISP_DE_MANUAL_ATTR_S

【说明】

定义 ISP DE 手动参数属性。

【定义】

```
typedef struct hiISP_DE_MANUAL_ATTR_S
{
    HI_U16      u16GlobalGain;
    HI_U16      u16GainLF;
    HI_U16      u16GainHF;
} ISP_DE_MANUAL_ATTR_S;
```

【成员】

成员名称	描述	取值范围	默认值
u16GlobalGain	DE 全局叠加强度增益	[0,256]	150
u16GainLF	DE 低频叠加强度增益	[0,32]	9
u16GainHF	DE 高频叠加强度增益	[0,32]	9

【注意事项】

仅 Hi3516CV500/Hi3559V200 中支持该结构体参数的配置与获取。

【相关数据类型及接口】

[ISP_DE_ATTR_S](#)

ISP_DE_ATTR_S

【说明】

定义 ISP DE 属性。



【定义】

```
typedef struct hiISP_DE_ATTR_S
{
    HI_BOOL      bEnable;
    HI_U16      au16LumaGainLut[HI_ISP_DE_LUMA_GAIN_LUT_N];
    ISP_OP_TYPE_E enOpType;
    ISP_DE_AUTO_ATTR_S stAuto;
    ISP_DE_MANUAL_ATTR_S stManual;
} ISP_DE_ATTR_S;
```



说明

宏定义如下：

```
#define HI_ISP_DE_LUMA_GAIN_LUT_N    (17)
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	使能 DE 功能。	[0,1]	0
au16LumaGainLut[17]	DE 亮度区间回叠控制表	[0,256]	-
enOpType	DE 的工作模式。	[0,1]	0
stAuto	DE 自动属性，详参： ISP_DE_AUTO_ATTR_S	-	-
stManual	DE 手动属性，详参： ISP_DE_MANUAL_ATTR_S	-	-

【注意事项】

- 仅 Hi3516CV500/Hi3559V200 中支持该结构体参数的配置与获取。
- DE 模块和 3DNR 第 0 级（nry0）是互斥关系，当打开 nry0 后，DE 会被强制关闭，参数不生效。
- DE 必须在 BayerNR 模块打开的前提下进行调试，同时如果 BayerNR 强度太小，DE 效果也难以体现。需要先将 BayerNR 调到一定强度下，再调试 DE 才能发挥出相应的效果。

【相关数据类型及接口】

- [ISP_OP_TYPE_E](#)

- [ISP_DE_AUTO_ATTR_S](#)
- [ISP_DE_MANUAL_ATTR_S](#)

6.35 RGB-IR

6.35.1 功能描述

注意

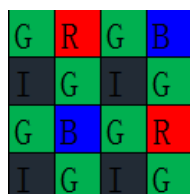
- 仅 Hi3516EV200 和 Hi3516EV300 支持 RGB-IR 功能。
- 目前市售大部分用于 IPC 产品形态的 RGB-IR 4x4 pattern 芯片波长在近红外区间 (NIR, 波长为 850nm)。如果需要使用 RGB-IR sensor 需要同时配合对应波长的双通滤光片使用。

芯片中所包含 RGB-IR 模块可用于处理排列满足 RGB-IR 4x4 pattern(以下简称为 RGB-IR4x4)的 RGB-IR sensor (RGB-IR 2x2 pattern 和 RGB-IR 4x4 pattern 的区别请参考图 6-18 和图 6-19)。通过本模块的开启, HiSP 系统可识别 RGB-IR4x4 信号并处理为正确的颜色信号, 同时可输出当前帧对应的 IR 分量图像供其他应用。

图6-18 传统 RGB-IR 2x2 pattern 排列



图6-19 RGB-IR 4x4 pattern 排列





6.35.2 API 参考

- [HI_MPI_ISP_SetRGBIRAttr](#): 设定 RGBIR 属性。
- [HI_MPI_ISP_GetRGBIRAttr](#): 获取 RGBIR 属性。

HI_MPI_ISP_SetRGBIRAttr

【描述】

设定 RGBIR 属性。

【语法】

```
HI_S32 HI_MPI_ISP_SetRGBIRAttr(VI_PIPE ViPipe, const ISP\_RGBIR\_ATTR\_S *pstRGBIRAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstRGBIRAttr	RGBIR 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_GetRGBIRAttr](#)

HI_MPI_ISP_GetRGBIRAttr

【描述】

获取 RGBIR 属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetRGBIRAttr(VI_PIPE ViPipe, ISP\_RGBIR\_ATTR\_S *pstRGBIRAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入
pstRGBIRAttr	RGBIR 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_SetRGBIRAttr

6.35.3 数据类型

- [ISP_RGBIR_ATTR_S](#): 定义 RGBIR 转换模块属性。
- [ISP_IRBAYER_FORMAT_E](#): 定义 IR pattern 输入 Pattern 类型。
- [ISP_IR_CVTMAT_MODE_E](#): 定义 IR 转换矩阵模式。

ISP_RGBIR_ATTR_S

【说明】

定义 RGBIR 转换模块属性。

【定义】

```
typedef struct hiISP_RGBIR_ATTR_S
{
    HI_BOOL      bEnable;
    ISP\_IRBAYER\_FORMAT\_E    enInPattern;
    ISP\_BAYER\_FORMAT\_E      enOutPattern;
    HI_U16        au16ExpCtrl[ISP_EXP_CTRL_NUM];
    HI_U16        au16Gain[ISP_EXP_CTRL_NUM];
    ISP\_IR\_CVTMAT\_MODE\_E    enIRCvtMatMode;
    HI_S16        as16CvtMatrix[ISP_CVTMAT_NUM];
} ISP_RGBIR_ATTR_S;
```



说明

宏定义如下：

```
#define ISP_CVTMAT_NUM 12
#define ISP_EXP_CTRL_NUM 2
```

【成员】

成员名称	描述	取值范围	默认值
bEnable	RGBIR 模块使能模块	[0x0,0x1]	0x0
enInPattern	RGBIR 输入格式选择	[0x0,0x7]	-
enOutPattern	RGBIR 模块输出格式选择，与普通 Bayer Pattern 一致	[0x0,0x3]	-
au16ExpCtrl	过曝处理模块相关，曝光亮度区间确	[0, 2047]	-



成员名称	描述	取值范围	默认值
	定参数		
au16Gain	过曝处理模块相关，在通过 au16ExpCtrl 确定了区间后，使用本参数进行过曝处理。	[0, 511]	256
enIRCvtMatMode	RGBIR 转换矩阵功能选择，分 Normal 模式、Mono 模式和 User 模式	[0, 2]	0
as16CvtMatrix	RGBIR 转换矩阵参数配置。Normal 模式和 Mono 模式下使用系统默认参数进行配置，User 模式下可配置	[-32768, 32767]	

【注意事项】

仅 Hi3516EV200/Hi3516EV300 中支持该结构体参数的配置与获取。

【相关数据类型及接口】

- [ISP_IRBAYER_FORMAT_E](#)
- [ISP_BAYER_FORMAT_E](#)
- [ISP_IR_CVTMAT_MODE_E](#)

ISP_IRBAYER_FORMAT_E

【说明】

定义 IR pattern 输入 Pattern 类型。

【定义】

```
typedef enum hiISP_IRBAYER_FORMAT_E
{
    IRBAYER_GRGBI = 0,
    IRBAYER_RGBGI = 1,
    IRBAYER_GBGR = 2,
    IRBAYER_BGRGI = 3,
    IRBAYER_IGRGB = 4,
    IRBAYER_IRGBG = 5,
    IRBAYER_IBGRG = 6,
```



```
IRBAYER_IGBGR = 7,  
IRBAYER_BUTT  
} ISP_IRBAYER_FORMAT_E;
```

【成员】

成员名称	描述
IRBAYER_GRGBI	输入 4x4 IR pattern，排列顺序如下所示： G R G B I G I G G B G R I G I G
IRBAYER_RGBGI	输入 4x4 IR pattern，排列顺序如下所示： R G B G G I G I B G R G G I G I
IRBAYER_GBGR I	输入 4x4 IR pattern，排列顺序如下所示： G B G R I G I G G R G B I G I G
IRBAYER_BGRGI	输入 4x4 IR pattern，排列顺序如下所示： B G R G G I G I R G B G G I G I
IRBAYER_IGRGB	输入 4x4 IR pattern，排列顺序如下所示： I G I G G R G B I G I G G B G R
IRBAYER_IRGBG	输入 4x4 IR pattern，排列顺序如下所示： G I G I R G B G



成员名称	描述
	G I G I B G R G
IRBAYER_IBGRG	输入 4x4 IR pattern, 排列顺序如下所示: G I G I B G R G G I G I R G B G
IRBAYER_IGBGR	输入 4x4 IR pattern, 排列顺序如下所示: I G I G G B G R I G I G G R G B

【注意事项】

仅 Hi3516EV200/Hi3516EV300 中支持该结构体参数的配置与获取。

【相关数据类型及接口】

[ISP_RGBIR_ATTR_S](#)

ISP_IR_CVTMAT_MODE_E

【说明】

定义 IR 转换矩阵模式。

【定义】

```
typedef enum hiISP_IR_CVTMAT_MODE_E
{
    ISP_IR_CVTMAT_MODE_NORMAL = 0,
    ISP_IR_CVTMAT_MODE_MONO = 1,
    ISP_IR_CVTMAT_MODE_USER = 2,
    ISP_IR_CVTMAT_MODE_BUTT
} ISP_IR_CVTMAT_MODE_E;
```

【成员】



成员名称	描述
ISP_IR_CVTMAT_MODE_NORMAL	枚举定义, RGBIR 模式选择, 彩色模式, 正常使用默认 CVTMAT 进行 IR 去除功能
ISP_IR_CVTMAT_MODE_MONO	枚举定义, RGBIR 模式选择, 夜晚红外模式下黑白图像
ISP_IR_CVTMAT_MODE_USER	枚举定义, RGBIR 模式选择, 用户自定义模式, 转换矩阵可自选

【注意事项】

仅 Hi3516EV200/Hi3516EV300 中支持该结构体参数的配置与获取。

【相关数据类型及接口】

[ISP_RGBIR_ATTR_S](#)

6.36 Auto Flicker Type Detection

6.36.1 功能描述

Auto Flicker Type Detection (自动工频闪类型检测) 是一个结合软件和硬件一起实现的算法。该算法能够计算出当前环境光源的频闪频率是 50Hz 还是 60Hz。

须知

- 在某些环境中, 当存在有 50Hz 光源和 60Hz 光源来回变动的情况时 (比如在室内 50Hz 的灯光下, 用户可能会关闭灯光, 打开 60Hz 光源的电视), Camera 无法应用合适的抗闪参数。所以此时需要检测出当前环境光源的工频闪频率, Camera 根据当前环境光源的工频闪频率适配合适的抗闪参数。
- 目前工频闪类型计算在 vpss 节点获取图像数据进行计算, 前端的 DIS、LDC、鱼眼校正等模块如果对图像的内容形态改变过大会影响工频闪类型计算的准确性。另外图像噪声过大, 也会影响工频闪类型计算的准确性。

6.36.2 API 参考

[HI_MPI_ISP_CalcFlickerType](#): 获取工频闪类型检测的结果。



HI_MPI_ISP_CalcFlickerType

【描述】

输出帧信息，获取工频闪类型检测的结果。

【语法】

```
HI_MPI_ISP_CalcFlickerType(VI_PIPE ViPipe, ISP\_CALCFLICKER\_INPUT\_S  
*pstInputParam, ISP\_CALCFLICKER\_OUTPUT\_S *pstOutputParam, VIDEO_FRAME_INFO_S  
stFrame[], HI_U32 u32ArraySize);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstInputParam	自动计算工频闪输入相关参数指针。	输入
pstOutputParam	自动计算工频闪输出结果参数指针。	输出
stFrame[]	输入需要计算的连续帧数据 VIDEO_FRAME_INFO_S 具体请参考《HiMPP 媒体处理软件 V4.0 开发参考》“02 系统控制”章节	输入
u32ArraySize	输入需要计算连续帧的个数	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：mpi_isp.h
- 库文件：lib_hicalcflicker.a

【注意】



- 必须使用连续的 3 帧数据进行计算，输入的 u32ArraySize 帧数需要等于 3。同时 stFrame[] 指向的数据必须保证有效。
- 计算的帧数据的分辨率不能超过 ISP 模块最大能够处理的图像分辨率。
- 为了加快计算速度，接口内部使用 VGS 硬件模块参与计算，所以用于计算工频闪的图像分辨率不能超过 VGS 的最大处理分辨率范围。
- 在不同频闪混合的光源下，比如光源环境同时存在 50Hz 和 60Hz 的光源，或者工频闪光源只占整个环境光源非常小的一部分，容易出现误判。
- Sample 用例中有使用 HI_MPI_VPSS_GetGrpFrame 接口（接口具体信息请参见《HiMPP 媒体处理软件 V4.0 开发参考》的 VPSS 章节）获取 3 帧数据，但此接口只在 Vpss 模块工作在离线模式时才有效。当 Vpss 处于在线模式时，无法获取有效帧数据进行工频闪类型计算。
- 可以参考附录 14.5 “[工频闪类型自适应 Sample 用例处理流程](#)”的接口相关用法。

【举例】

```
VI_PIPE ViPipe = 0;
HI_U16 u16FrameNum = 3;
VIDEO_FRAME_INFO_S stFrame[3];
ISP_CALCFLICKER_INPUT_S stInputParam;
ISP_CALCFLICKER_OUTPUT_S stOutputParam;
ISP_EXP_INFO_S pstExpInfo;
HI_MPI_ISP_QueryExposureInfo(ViPipe, &pstExpInfo);
stInputParam.u32LinesPerSecond = pstExpInfo.u32LinesPer500ms*2
... ..
获取3帧连续帧数据给 stFrame[3](此处略)
... ..
s32Ret = HI_MPI_ISP_CalcFlickerType(ViPipe, pstInputParam, pstOutputParam, stFrame,
u16FrameNum);
```

【相关主题】

[HI_MPI_ISP_CalcFlickerType](#)

6.36.3 数据类型

- [ISP_CALCFLICKER_INPUT_S](#)：定义计算工频闪需要的输入参数。
- [ISP_CALCFLICKER_OUTPUT_S](#)：定义自动工频闪类型检测输出结果。



ISP_CALCFLICKER_INPUT_S

【说明】

定义计算工频闪输入参数。

【定义】

```
typedef struct hiISP_CALCFLICKER_INPUT_S {  
    HI_U32 u32LinesPerSecond;  
} ISP_CALCFLICKER_INPUT_S;
```

【成员】

成员名称	描述	取值范围	默认值
u32LinesPerSecond	当前序列 1 秒的最大曝光总行数	[0x1F4,0xFFFF FFFFF]	0x0

【注意事项】

- u32LinesPerSecond 等于 1 秒的最大总曝光行数，比如一帧的最大曝光时间为 1125 行，帧率为 30fps，那么 u32LinesPerSecond = 1125*30。通常，可以使用 ISP_EXP_INFO_S 这个结构体成员变量 u32LinesPer500ms 乘以 2。
- 需要注意这里的 1 行时间需要和 1 行数据的 Readout Timing 的时间相等，一般 sensor 的曝光行数单位时间和 Readout 1 行的时间相同，需要注意少部分 sensor 的 Readout 1 行时间等于 2 行，那么计算 u32LinesPerSecond 时需要除以 2。其他等效倍率则除以相应的倍率。

【相关数据类型及接口】

[HI_MPI_ISP_CalcFlickerType](#)

ISP_CALCFLICKER_OUTPUT_S

【说明】

定义自动工频闪类型检测输出结果。

【定义】

```
typedef struct hiISP_CALCFLICKER_OUTPUT_S {  
    ISP_CALCFLICKER_TYPE_E enFlickerType; /*The calculate result of flicker type*/  
} ISP\_CALCFLICKER\_OUTPUT\_S;
```



```
typedef enum hiISP_CALCFLICKER_TYPE_E
{
    FLICKER_TYPE_NONE = 0,
    FLICKER_TYPE_UNKNOW,
    FLICKER_TYPE_50HZ,
    FLICKER_TYPE_60HZ,
    FLICKER_TYPE_BUTT,
} ISP_CALCFLICKER_TYPE_E;
```

【成员】

成员名称	描述
enFlickerType	ISP_CALCFLICKER_TYPE_E:工频闪计算结果。有以下 4 种状态: FLICKER_TYPE_NONE 无工频闪 FLICKER_TYPE_UNKNOW 有工频闪, 但不确定是哪种类型 FLICKER_TYPE_50HZ 50HZ 的工频闪 FLICKER_TYPE_60HZ 60HZ 的工频闪

【注意事项】

无

【相关数据类型及接口】

[HI_MPI_ISP_CalcFlickerType](#)

6.37 Auto Color Shading

6.37.1 功能描述

Auto Color Shading (ACS)算法是一个软件上实现的算法, 该算法能够根据场景自适应地生成 LSC 的校正表, 通过配置 Mesh Shading 的 Lut 校正表, 从而能够减轻或消除镜头和 sensor 造成的 color shading。该算法需要在使用前进行标定, 标定的数据预先放入 cmos.c 中。



6.37.2 API 参考

- [HI_MPI_ISP_SetAutoColorShadingAttr](#): 设置 Auto Color Shading 属性参数。
- [HI_MPI_ISP_GetAutoColorShadingAttr](#): 获取 Auto Color Shading 属性参数。

HI_MPI_ISP_SetAutoColorShadingAttr

【描述】

设置 Auto Color Shading 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_SetAutoColorShadingAttr(VI_PIPE ViPipe, const ISP\_ACS\_ATTR\_S
*pstACSAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstACSAttr	Auto Color Shading 属性参数。	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：lib_hiacs.a

【注意】

无

【举例】



无

【相关主题】

[HI_MPI_ISP_GetAutoColorShadingAttr](#)

HI_MPI_ISP_GetAutoColorShadingAttr

【描述】

获取 Auto Color Shading 属性参数。

【语法】

```
HI_S32 HI_MPI_ISP_GetAutoColorShadingAttr(VI_PIPE ViPipe, ISP\_ACS\_ATTR\_S *pstACSAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输出
pstACSAttr	Auto Color Shading 属性参数。	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：lib_hiacc.a

【注意】

无

【举例】

无



【相关主题】

[HI_MPI_ISP_SetAutoColorShadingAttr](#)

6.37.3 数据类型

[ISP_ACS_ATTR_S](#): 定义 Auto Color Shading 属性参数。

ISP_ACS_ATTR_S

【说明】

定义 Auto Color Shading 属性参数。

【定义】

```
typedef struct hiISP_ACS_ATTR_S
{
    HI_BOOL bEnable;
    HI_U16  u16YStrength;
    HI_U16  u16RunInterval;
    HI_BOOL bLockEnable;
} ISP_ACS_ATTR_S;
```

【成员】

成员名称	描述
bEnable	取值范围：[0x0, 0x1]。 ACS 模块使能。
u16YStrength	取值范围：[0x0, 0x100]。 控制 ACS 算法 Luma Shading 也就是 Y Shading 的校正强度，默认 256
u16RunInterval	取值范围：[0x1, 0xff]。 控制 ACS 算法多少帧运行一次，默认 2 帧运行一次
bLockEnable	取值范围：[0x0, 0x1]。 固定 ACS 产生的 Lut 表，保持当前的 LSC 校正表。

【注意事项】



- Hi3559AV100/Hi3519AV100 在拼接模式下不支持。
- ACS 模块运行需要正确参考 AWB 统计 [ISP_AWB_SWITCH_E](#)，ACS 未开启时，AWB 统计模块位置在 [ISP_AWB_AFTER_DG](#)；ACS 开启时，会自动设置到 [ISP_AWB_AFTER_Expander](#)，应保持该设置，若此后再手动设置为别的位置后，容易造成 ACS 震荡等问题。
- ACS 模块开关与 LSC 的模块开关互不影响。ACS 算法通过调用 LSC 的接口来配置 Lut 校正表，需要看到 ACS 模块效果时，Mesh Shading 模块开关必须使能。
- 关闭 ACS 模块使能开关时，LSC 的 Lut 表会恢复到未开 ACS 前的状态。
- ACS 会参考 u8MeshScale 的值来配置 LSC 相应的 Lut 表，cmos 中 bit1Lsc 使能时会参考 cmos 中 LSC 的 u8MeshScale 配置。
- 推荐 u8MeshScale 的值取 0~3，而不是 4~7，因为 4~7 无法表示小于 1 倍的值，当 u16YStrength 的值较小时，校正 Color Shading 可能需要小于 1 倍的值。
- ACS 模块需要在上海海思 awb 库下运行。

【相关数据类型及接口】

无



7 统计信息

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

7.1 概述

ISP 提供的 3A 统计信息及相关配置。

7.2 功能描述

7.2.1 AE 统计信息

实现自动曝光信息的统计，软件根据统计信息调节 sensor 可实现自动曝光的功能。AE 统计信息包含分区间的 R/Gr/Gb/B 均值统计、带权重的全局 R/Gr/Gb/B 均值、1024 段直方图信息。

7.2.1.1 AE 区域统计信息

AE 最大支持 17×15 分块，最小支持 1×1 分块，每个分块均可以输出 16bit R, Gr, Gb, B 均值(Average R/Gr/Gb/B)。以 R、Gr 为例，区块的统计信息读取地址与数据的对应关系如表 7-1 所示。m 表示水平分区个数，n 表示垂直分区个数。



表7-1 AE 区域统计信息

Memory address	Zone	Memory Read Data	
		[31:16]	[15:0]
0	0	Average R	Average Gr
1	1	Average R	Average Gr
2	2	Average R	Average Gr
3	3	Average R	Average Gr
.....			
m*n-3	m*n-3	Average R	Average Gr
m*n-2	m*n-2	Average R	Average Gr
m*n-1	m*n-1	Average R	Average Gr

7.2.1.2 带权重的 AE 全局统计信息

与 AE 的区域统计信息基本一致，对于整幅图像，AE 也提供了 4 个全局统计信息。分别为 ISP_AE_TOTAL_R_AVER, ISP_AE_TOTAL_GR_AVER, ISP_AE_TOTAL_GB_AVER, ISP_AE_TOTAL_B_AVER。其物理意义与分块统计一致。

7.2.1.3 带权重的 AE 直方图信息

AE 提供 1024 阶直方图统计 0-1023，直方图信息读取地址与数据的对应关系如表 7-2 所示。

表7-2 AE 直方图统计信息

Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
0	0	0	Pixel Value 为 0 的个数
1	1	0	Pixel Value 为 1 的个数
2	2	0	Pixel Value 为 2 的个数
3	3	0	Pixel Value 为 3 的个数
.....			



Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
1022	1022	0	Pixel Value 为 1022 的个数
1023	1023	0	Pixel Value 为 1023 的个数

AE 支持 FourPlaneMode 模式，在 FourPlaneMode 模式下，AE 提供四通道 256 段直方图统计 0~255，直方图信息读取地址与数据的对应关系如表 7-3 所示。

表7-3 AE 直方图统计信息

Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
0	0	0	B 通道 Pixel Value 为 0 的个数
1	1	0	B 通道 Pixel Value 为 1 的个数
.....			
255	255	0	B 通道 Pixel Value 为 255 的个数
256	256	0	Gb 通道 Pixel Value 为 0 的个数
.....			
511	511	0	Gb 通道 Pixel Value 为 255 的个数
512	512	0	Gr 通道 Pixel Value 为 0 的个数
.....			
767	767	0	Gr 通道 Pixel Value 为 255 的个数
768	768	0	R 通道 Pixel Value 为 0 的个数
.....			
1022	1022	0	R 通道 Pixel Value 为 254 的个数



Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
1023	1023	0	R 通道 Pixel Value 为 255 的个数

7.2.2 MG 统计信息

MG 模块统计 DRC 后分块均值，与 AE 统计分块均值相比，可以得出分块均值增益最大值。MG 统计信息包含全局分区间 R/Gr/Gb/B 均值统计。

7.2.2.1 MG 区域统计信息

MG 最大支持 17×15 分块，最小支持 1×1 分块，每个分块均可以输出 8bit R, Gr, Gb, B 均值(Average R/Gr/Gb/B)。区块的统计信息读取地址与数据的对应关系如表 7-4 所示。m 表示水平分区个数，n 表示垂直分区个数。

表7-4 MG 区域统计信息

Memory address	Zone	Memory Read Data			
		[31:24]	[23:16]	[15:8]	[7:0]
0	0	Average R	Average Gr	Average Gb	Average B
1	1	Average R	Average Gr	Average Gb	Average B
2	2	Average R	Average Gr	Average Gb	Average B
3	3	Average R	Average Gr	Average Gb	Average B
.....					
m*n-3	m*n-3	Average R	Average Gr	Average Gb	Average B
m*n-2	m*n-2	Average R	Average Gr	Average Gb	Average B
m*n-1	m*n-1	Average R	Average Gr	Average Gb	Average B

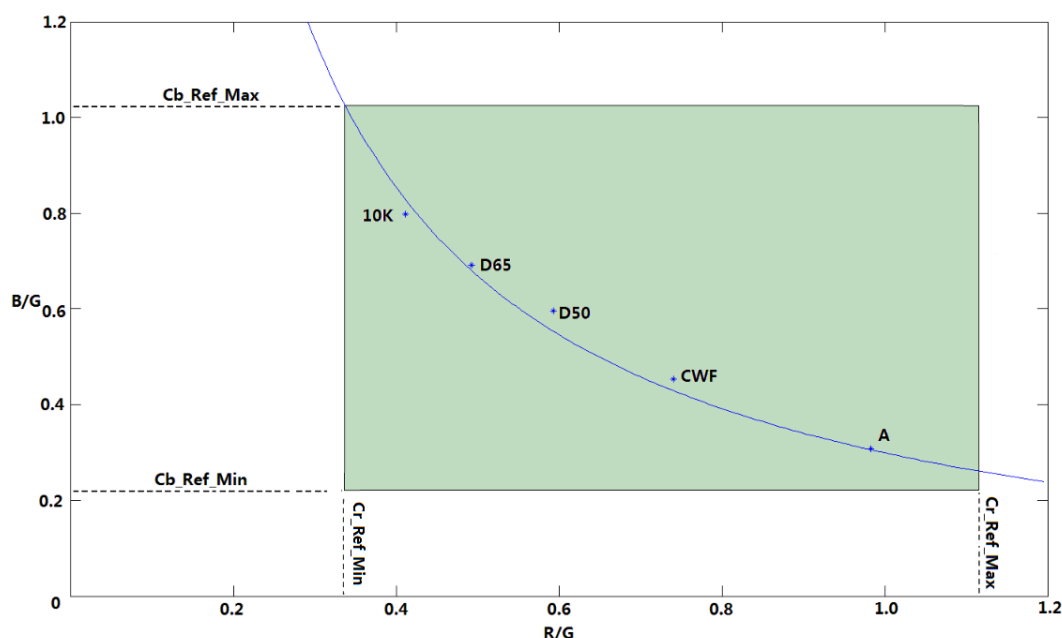
7.2.3 AWB 统计信息

AWB 统计信息包含全局统计信息和区域统计信息。

- 全局统计信息：整幅图像的 R, G, B 均值, 以及有效统计点的个数。
- 区域统计信息：最大支持图像的 32x32 分块, 每个分块均可以输出 R, G, B 均值, 以及有效统计点的个数。

AWB 中统计的有效点, 除了要满足配置的 RGB 上限与下限要求外 (ISP_AWB_THD_MIN, ISP_AWB_THD_MAX), 像素点颜色还需要满足配置的有效像素颜色空间, 具体如图 7-1 所示, 绿色区块内的为有效像素的颜色范围。

图7-1 AWB 有效像素颜色空间



AWB 统计的有效像素区域是一个四边形, 其中上下左右四个边界由 cr_ref_min/max, cb_ref_min/max 这 4 个参数决定, 它们代表了允许的 r/g, b/g 的最大与最小值。

7.2.3.1 AWB 区域统计信息

AWB 中最大支持图像的 32x32 分块, 每个分块均可以输出 R、G、B 均值(Average R/G/B)。其中统计的像素点均为 16bit, 分块内统计有效点个数 Count All, 点数统计值均已根据图像大小做了归一化处理, 归一化到 16bit。同时, 每一个分块支持分亮度区间进行统计, 最大支持分 4 个亮度区间。区块的统计信息读取地址与数据的对应关系如表 7-5 所示。



须知

- 当分辨率较小，分块数较多时，如果每块的分辨率小于 60x14，那么需要减少分块数，以免每块过小。建议在分辨率小的情况下，使用较少的分块数。
- 如果要输出低分辨率，需要在业务启动的时候设置一个比较小的 AWB 统计信息分块数。

表7-5 AWB 区域统计信息

Memory address	Zone	Bin	Memory Read Data	
			[31:16]	[15:0]
0	0	0	Average G	Average R
1	0	0	Count All	Average B
2	0	1	Average G	Average R
3	0	1	Count All	Average B
4	0	2	Average G	Average R
5	0	2	Count All	Average B
6	0	3	Average G	Average R
7	0	3	Count All	Average B
8	1	0	Average G	Average R
9	1	0	Count All	Average B
				
8191	1023	3	Count All	Average B

7.2.3.2 AWB 全局统计信息

与 AWB 的区域统计信息基本一致，对于整幅图像，AWB 也提供了四个全局统计信息。分别为 ISP_AWB_AVG_R/G/B，ISP_AWB_CNT_ALL。其物理意义与分块统计一致。



7.3 API 参考

统计信息的 API 接口必须在调用 [HI_MPI_ISP_Init](#) 接口之后才能调用。

- [HI_MPI_ISP_SetStatisticsConfig](#): 设置 ISP 3A 统计信息配置。
- [HI_MPI_ISP_GetStatisticsConfig](#): 获取 ISP 3A 统计信息配置。
- [HI_MPI_ISP_GetAESTatistics](#): 获取 AE 统计信息。
- [HI_MPI_ISP_GetAESTitchStatistics](#): 获取拼接模式 AE 统计信息。
- [HI_MPI_ISP_GetMGStatistics](#): 获取 MG 统计信息。
- [HI_MPI_ISP_GetWBStatistics](#): 获取 AWB 统计信息。
- [HI_MPI_ISP_GetWBStitchStatistics](#): 获取拼接模式 AWB 统计信息。
- [HI_MPI_ISP_GetFocusStatistics](#): 获取 AF 统计信息。

HI_MPI_ISP_SetStatisticsConfig

【描述】

设置 ISP 3A 统计信息配置。

【语法】

```
HI_S32 HI_MPI_ISP_SetStatisticsConfig(VI_PIPE ViPipe, const ISP\_STATISTICS\_CFG\_S *pstStatCfg);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstStatCfg	ISP 统计信息配置。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 错误码 。



【需求】

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

【注意】

用户可以关闭不需要的统计信息，以提高系统性能。

【举例】

无

【相关主题】

[HI_MPI_ISP_GetStatisticsConfig](#)

HI_MPI_ISP_GetStatisticsConfig

【描述】

获取 ISP 3A 统计信息配置。

【语法】

```
HI_S32 HI_MPI_ISP_GetStatisticsConfig(VI_PIPE ViPipe, ISP_STATISTICS_CFG_S *pstStatCfg);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstStatCfg	ISP 统计信息配置。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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_SetStatisticsConfig](#)

HI_MPI_ISP_GetAESTatistics

【描述】

获取 AE 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetAESTatistics(VI_PIPE ViPipe, ISP\_AE\_STATISTICS\_S *pstAeStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstAeStat	AE 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 。

【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_GetAESTitchStatistics

【描述】

获取拼接模式 AE 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetAESTitchStatistics(VI_PIPE ViPipe, ISP\_AE\_STITCH\_STATISTICS\_S  
*pstStitchStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstStitchStat	拼接模式 AE 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 。

【需求】

- 头文件: hi_comm_isp.h、 mpi_isp.h



- 库文件: libisp.a

【注意】

Hi3516CV500/Hi3559V200/Hi3516EV300/Hi3516EV200 不支持。

【举例】

无

【相关主题】

无

HI_MPI_ISP_GetMGStatistics

【描述】

获取 MG 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetMGStatistics(VI_PIPE ViPipe, ISP_MG_STATISTICS_S *pstMgStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstMgStat	MG 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 。

【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_GetWBStatistics

【描述】

获取 WB 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetWBStatistics(VI_PIPE ViPipe, ISP_WB_STATISTICS_S *pstWBStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstWBStat	WB 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 。

【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_GetWBStitchStatistics

【描述】

获取拼接模式下 WB 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetWBStitchStatistics (VI_PIPE ViPipe, ISP_WB_STITCH_STATISTICS_S*  
pstStitchWBStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstStitchWBStat	拼接模式 WB 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 。

【需求】

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

【注意】



Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持

【举例】

无

【相关主题】

无

HI_MPI_ISP_GetFocusStatistics

【描述】

获取 AF 统计信息。

【语法】

```
HI_S32 HI_MPI_ISP_GetFocusStatistics(VI_PIPE ViPipe, ISP\_AF\_STATISTICS\_S *pstAfStat);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。	输入
pstAfStat	AF 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 。

【需求】

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

【注意】

无



【举例】

无

【相关主题】

无

7.4 数据类型

- [ISP_STATISTICS_CFG_S](#): 定义 ISP 3A 统计信息配置。
- [ISP_STATISTICS_CTRL_U](#): 定义 ISP 3A 统计信息使能。
- [ISP_AE_STATISTICS_CFG_S](#): 定义 AE 统计信息配置。
- [ISP_AE_SWITCH_E](#): 定义 AE 统计模块在 ISP pipeline 的位置。
- [ISP_AE_FOUR_PLANE_MODE_E](#): FourPlaneMode 使能。
- [ISP_AE_HIST_SKIP_E](#): 直方图统计时采样点设置。
- [ISP_AE_HIST_OFFSET_X_E](#): 直方图统计时水平方向起始点设置。
- [ISP_AE_HIST_OFFSET_Y_E](#): 直方图统计时垂直方向起始点设置。
- [ISP_AE_HIST_CONFIG_S](#): 定义全局直方图统计时的采样点方式。
- [ISP_AE_STAT_MODE_E](#): 定义统计模块的开方模式。
- [ISP_AE_CROP_S](#): 对 AE 模块输入图像进行裁剪。
- [ISP_AWB_SWITCH_E](#): 定义 ISP WB 统计模块在 ISP pipeline 的位置。
- [ISP_AWB_GAIN_SWITCH_E](#): 定义 AWB 增益的位置。
- [ISP_AWB_CROP_S](#): 对 AWB 模块输入图像进行裁剪。
- [ISP_WB_STATISTICS_CFG_S](#): 定义 AWB 统计信息配置。
- [ISP_AE_GRID_INFO_S](#): 定义 AE 分区间统计信息的坐标信息。
- [ISP_AE_STATISTICS_S](#): 定义 AE 统计信息。
- [ISP_MG_GRID_INFO_S](#): 定义 MG 分区间统计信息的坐标信息。
- [ISP_AE_STITCH_STATISTICS_S](#): 定义拼接模式 AE 统计信息。
- [ISP_AWB_GRID_INFO_S](#): 定义 AWB 分区间统计信息的坐标信息。
- [ISP_MG_STATISTICS_S](#): 定义 MG 统计信息。
- [ISP_WB_STATISTICS_S](#): 定义 AWB 统计信息。
- [ISP_WB_STITCH_STATISTICS_S](#): 定义拼接模式 AWB 统计信息。



- [ISP_FOCUS_STATISTICS_CFG_S](#): 定义 AF 统计信息配置。
- [ISP_AF_CFG_S](#): 定义 AF 统计信息配置参数。
- [ISP_AF_PEAK_MODE_E](#): 统计模式。
- [ISP_AF_SQU_MODE_E](#): 平方模式。
- [ISP_AF_CROP_S](#): AF 输入图像的裁剪。
- [ISP_AF_STATISTICS_POS_E](#): AF 统计位置。
- [ISP_AF_RAW_CFG_S](#): AF RAW 域统计配置。
- [ISP_AF_PRE_FILTER_CFG_S](#): AF 预滤波器配置。
- [ISP_AF_LD_S](#): AF Level Depend Gain 配置。
- [ISP_AF_CORING_S](#): AF Coring 配置。
- [ISP_AF_H_PARAM_S](#): AF 统计水平滤波器 IIR 参数设置。
- [ISP_AF_V_PARAM_S](#): AF 统计水平滤波器 IIR 参数设置。
- [ISP_AF_FV_PARAM_S](#): AF 统计结果输出格式参数。
- [ISP_FOCUS_ZONE_S](#): 定义 AF 统计信息参数。
- [ISP_FE_FOCUS_STATISTICS_S](#): 定义 FE AF 统计信息。
- [ISP_BE_FOCUS_STATISTICS_S](#): 定义 BE AF 统计信息。
- [ISP_FOCUS_GRID_INFO_S](#): 定义 AF 分区间统计信息的坐标信息。
- [ISP_AF_STATISTICS_S](#): 定义 FE 和 BE 的 AF 统计信息。

ISP_STATISTICS_CFG_S

【说明】

定义 ISP 3A 统计信息配置。

【定义】

```
typedef struct hiISP_STATISTICS_CFG_S
{
    ISP\_STATISTICS\_CTRL\_U      unKey;
    ISP\_AE\_STATISTICS\_CFG\_S    stAECfg;
    ISP\_WB\_STATISTICS\_CFG\_S    stWBCfg;
    ISP\_FOCUS\_STATISTICS\_CFG\_S stFocusCfg;
} ISP_STATISTICS_CFG_S;
```

【成员】



成员名称	描述
unKey	统计信息使能。
stAECfg	AE 统计信息配置。
stWBCfg	AWB 统计信息配置。
stFocusCfg	AF 统计信息配置。

【注意事项】

无。

【相关数据类型及接口】

无

ISP_STATISTICS_CTRL_U

【说明】

定义 ISP 3A 统计信息使能。

【定义】

```
typedef union hiISP_STATISTICS_CTRL_U
{
    HI_U64 u64Key;
    struct
    {
        HI_U64 bit1FEAeGloStat : 1; /* [0] */
        HI_U64 bit1FEAeLocStat : 1; /* [1] */
        HI_U64 bit1FEAeStiGloStat : 1; /* [2] */
        HI_U64 bit1FEAeStiLocStat : 1; /* [3] */
        HI_U64 bit1BEAeGloStat : 1; /* [4] */
        HI_U64 bit1BEAeLocStat : 1; /* [5] */
        HI_U64 bit1BEAeStiGloStat : 1; /* [6] */
        HI_U64 bit1BEAeStiLocStat : 1; /* [7] */
        HI_U64 bit1AwbStat1 : 1; /* [8] */
        HI_U64 bit1AwbStat2 : 1; /* [9] */
        HI_U64 bit2Rsv0 : 2; /* [10:11] */
    }
}
```



```

HI_U64 bit1FEAfStat      : 1; /* [12] */
HI_U64 bit1BEAfStat      : 1; /* [13] */
HI_U64 bit2Rsv1          : 2; /* [14:15] */
HI_U64 bit1Dehaze        : 1; /* [16] */
HI_U64 bit1MgStat        : 1; /* [17] */
HI_U64 bit14Rsv          : 14; /* [18:31] */
HI_U64 bit32IsrAccess    : 32; /* [32:63] */

};
}ISP_STATISTICS_CTRL_U;

```

【成员】

成员名称	描述
bit1FEAeGloStat	位于 FE 的 AE 全局统计信息使能。包括直方图和全局均值。
bit1FEAeLocStat	位于 FE 的 AE 分块均值统计信息使能。
bit1FEAeStiGloStat	拼接后位于 FE 的 AE 全局统计信息使能。包括直方图和全局均值。仅在拼接模式下生效。仅 Hi3559AV100/Hi3519AV100 支持。
bit1FEAeStiLocStat	拼接后位于 FE 的 AE 分块均值统计信息使能。仅在拼接模式下生效。仅 Hi3559AV100/Hi3519AV100 支持
bit1BEAeGloStat	位于 BE 的 AE 全局统计信息使能。包括直方图和全局均值。
bit1BEAeLocStat	位于 BE 的 AE 分块均值统计信息使能。
bit1BEAeStiGloStat	拼接后位于 BE 的 AE 全局统计信息使能。包括直方图和全局均值。仅在拼接模式下生效。仅 Hi3559AV100/Hi3519AV100 支持
bit1BEAeStiLocStat	拼接后位于 BE 的 AE 分块均值统计信息使能。仅在拼接模式下生效。仅 Hi3559AV100/Hi3519AV100 支持
bit1AwbStat1	Bayer 域 AWB 全局统计信息使能。
bit1AwbStat2	Bayer 域 AWB 分区间统计信息使能。
bit2Rsv0	保留位。
bit1FEAfStat	位于 FE 的 AF 统计信息使能。仅 Hi3559AV100/Hi3519AV100 支持。



成员名称	描述
bit1BEAfStat	位于 BE 的 AF 统计信息使能。Hi3516EV200/Hi3516EV300 不支持。
bit2Rsv1	保留位。
bit1Dehaze	Dehaze 统计信息使能。
bit1MgStat	MG 统计信息使能。
bit14Rsv	保留位。
bit32IsrAccess	中断服务程序中更新统计信息开关。

【注意事项】

- 用户可以关闭不使用的统计信息，如此就不会读相应统计信息，从而提高系统性能。
- bit1FEAeGloStat~ bit14Rsv 32 个 bit 为使用 HI_MPI_ISP_GetxxStatistics 接口（xx 表示 AE/AWB/AF）获取统计信息时相应的开关。isp firmware 运转需要获取统计信息供上海海思相关算法使用。
- 如果用户使用 HI_MPI_ISP_GetxxStatistics（xx 表示 AE/AWB/AF）实现自己的 3A 算法，可以通过 disable bit32IsrAccess 中相应的 bit 来关掉中断服务程序中相关统计信息的更新以达到降低 CPU 占用率的目的。bit32IsrAccess 中 bit 顺序定义也是 bit1FEAeGloStat ~ bit14Rsv。

例如：可以置零 bit32IsrAccess 的 bit0 来停止更新 FEAeStat 统计信息。

【相关数据类型及接口】

无

ISP_AE_STATISTICS_CFG_S

【说明】

定义 AE 统计信息配置。

【定义】

```
typedef struct hiISP_AE_STATISTICS_CFG_S
{
    ISP_AE_SWITCH_E enAESwitch;
```



```
ISP_AE_HIST_CONFIG_S stHistConfig;  
ISP_AE_FOUR_PLANE_MODE_E enFourPlaneMode;  
ISP_AE_STAT_MODE_E enHistMode;  
ISP_AE_STAT_MODE_E enAverMode;  
ISP_AE_STAT_MODE_E enMaxGainMode;  
ISP_AE_CROP_S stCrop;  
HI_U8 au8Weight[AE_ZONE_ROW][AE_ZONE_COLUMN];  
} ISP_AE_STATISTICS_CFG_S;
```

【成员】

成员名称	描述
enAESwitch	WDR 合成后 AE 统计模块在 ISP pipeline 的位置，默认值为 1 0: After ISP digital gain; 1: After static WB; 2: After DRC。
stHistConfig	全局直方图统计时的采样点方式配置。
enFourPlaneMode	FourPlaneMode 使能，默认值为 0，使能时 1024 段直方图为 BGGR 四通道 256 段直方图。 0: 不使能; 1: 使能。
enHistMode	全局 1024 段直方图开方模式。0 为不开方，1 为开方。仅影响 BE 直方图统计信息。
enAverMode	均值开方模式。0 为不开方，1 为开方。仅影响 BE 均值统计信息。
enMaxGainMode	MG 模块开方模式。0 为不开方，1 为开方。仅影响 MG 模块统计信息。为了与 AE 分块统计信息进行比较，建议与 enAverMode 配置为同一模式。
stCrop	AE 输入图像裁剪配置。
au8Weight	15x17 个区间的 AE 权重表，取值范围为[0, 15]。

【注意事项】



- 线性模式数据流走的是 channel 1，默认使用全局 1024 段直方图进行 AE 控制，此时 channel 2/3/4 WDR 合成前全局 1024 段直方图全为 0。
- 全局 4 分量统计平均值会受 15*17 权重表的影响，。
- Channel 1/2/3/4 全局 1024 段直方图会受 15*17 权重表的影响，WDR 模式可以使用这几路直方图信息进行 AE 控制。如 2 合 1 可使用 channel 1/2，3 合 1 可使用 channel 1/2/3，4 合 1 可使用 channel 1/2/3/4 等。
- 配置 enAESwitch /stHistConfig /enFourPlaneMode 相关参数时，需要把上海海思 AE 的大面积单色判断功能关闭。
- 自动曝光的静态统计信息分为 15*17 个区域，可通过设定权重表改变每个区域的权重。如可使中心区域的权重加大，则中心区域的亮度变化会使图像的全局直方图统计信息产生更多的变化。该属性可调节感兴趣区域的曝光权重，可用于实现指定区域的背光补偿。若能确定具体产品形态感兴趣区域的位置，可将这些位置的权重加大。如对于运动 DV 和行车记录仪，感兴趣区域一般在画面的中下部分，那么可适当加大场景中下部分区域的权重，使得感兴趣区域得到合理曝光。

【相关数据类型及接口】

无

ISP_AE_SWITCH_E

【说明】

定义 AE 统计模块在 ISP pipeline 的位置。

【定义】

```
typedef enum hiISP_AE_SWITCH_E
{
    ISP_AE_AFTER_DG = 0,
    ISP_AE_AFTER_WB,
    ISP_AE_AFTER_DRC,
    ISP_AE_SWITCH_BUTT
} ISP_AE_SWITCH_E;
```

【成员】

成员名称	描述
ISP_AE_AFTER_DG	After ISP digital gain
ISP_AE_AFTER_WB	After static WB



成员名称	描述
ISP_AE_AFTER_DRC	After DRC

【注意事项】

配置此参数时，需要把上海海思 AE 的大面积单色判断功能关闭。

【相关数据类型及接口】

[ISP_AE_STATISTICS_CFG_S](#)

ISP_AE_FOUR_PLANE_MODE_E

【说明】

FourPlaneMode 使能。

【定义】

```
typedef enum hiISP_AE_FOUR_PLANE_MODE_E
{
    ISP_AE_FOUR_PLANE_MODE_DISABLE = 0,
    ISP_AE_FOUR_PLANE_MODE_ENABLE,
    ISP_AE_FOUR_PLANE_MODE_BUTT
} ISP_AE_FOUR_PLANE_MODE_E;
```

【成员】

成员名称	描述
ISP_AE_FOUR_PLANE_MODE_DISABLE	1024 段直方图
ISP_AE_FOUR_PLANE_MODE_ENABLE	BGGR 四通道 256 段直方图

【注意事项】

配置此参数时，需要把上海海思 AE 的大面积单色判断功能关闭。

【相关数据类型及接口】

[ISP_AE_STATISTICS_CFG_S](#)



ISP_AE_HIST_SKIP_E

【说明】

直方图统计时采样点设置。

【定义】

```
typedef enum hiISP_AE_HIST_SKIP_E
{
    ISP_AE_HIST_SKIP_EVERY_PIXEL = 0,
    ISP_AE_HIST_SKIP_EVERY_2ND_PIXEL,
    ISP_AE_HIST_SKIP_EVERY_3RD_PIXEL,
    ISP_AE_HIST_SKIP_EVERY_4TH_PIXEL,
    ISP_AE_HIST_SKIP_EVERY_5TH_PIXEL,
    ISP_AE_HIST_SKIP_EVERY_8TH_PIXEL,
    ISP_AE_HIST_SKIP_EVERY_9TH_PIXEL,
    ISP_AE_HIST_SKIP_BUTT
} ISP_AE_HIST_SKIP_E;
```

【成员】

成员名称	描述
ISP_AE_HIST_SKIP_EVERY_PIXEL	图像中每个点都统计。
ISP_AE_HIST_SKIP_EVERY_2ND_PIXEL	图像在水平或垂直方向上每两个点统计一次。
ISP_AE_HIST_SKIP_EVERY_3RD_PIXEL	图像在水平或垂直方向上每三个点统计一次。
ISP_AE_HIST_SKIP_EVERY_4TH_PIXEL	图像在水平或垂直方向上每四个点统计一次。
ISP_AE_HIST_SKIP_EVERY_5TH_PIXEL	图像在水平或垂直方向上每五个点统计一次。
ISP_AE_HIST_SKIP_EVERY_8TH_PIXEL	图像在水平或垂直方向上每八个点统计一次。
ISP_AE_HIST_SKIP_EVERY_9TH_PIXEL	图像在水平或垂直方向上每九个点统计一次。



【注意事项】

无

【相关数据类型及接口】

[ISP_AE_HIST_CONFIG_S](#)

ISP_AE_HIST_OFFSET_X_E

【说明】

直方图统计时水平方向起始点设置。

【定义】

```
typedef enum hiISP_AE_HIST_OFFSET_X_E
{
    ISP_AE_HIST_START_FORM_FIRST_COLUMN = 0,
    ISP_AE_HIST_START_FORM_SECOND_COLUMN,
    ISP_AE_HIST_OFFSET_X_BUTT
} ISP_AE_HIST_OFFSET_X_E;
```

【成员】

成员名称	描述
ISP_AE_HIST_START_FORM_FIRST_COLUMN	从图像的第一列开始统计。
ISP_AE_HIST_START_FORM_SECOND_COLUMN	从图像的第二列开始统计。

【注意事项】

无

【相关数据类型及接口】

[ISP_AE_HIST_CONFIG_S](#)

ISP_AE_HIST_OFFSET_Y_E

【说明】

直方图统计时垂直方向起始点设置。



【定义】

```
typedef enum hiISP_AE_HIST_OFFSET_Y_E
{
    ISP_AE_HIST_START_FORM_FIRST_ROW = 0,
    ISP_AE_HIST_START_FORM_SECOND_ROW,
    ISP_AE_HIST_OFFSET_Y_BUTT
} ISP_AE_HIST_OFFSET_Y_E;
```

【成员】

成员名称	描述
ISP_AE_HIST_START_FORM_FIRST_ROW	从图像的第一行开始统计。
ISP_AE_HIST_START_FORM_SECOND_ROW	从图像的第二行开始统计。

【注意事项】

无

【相关数据类型及接口】

[ISP_AE_HIST_CONFIG_S](#)

ISP_AE_HIST_CONFIG_S

【说明】

定义全局直方图统计时的采样点方式。

【定义】

```
typedef struct hiISP_AE_HIST_CONFIG_S
{
    ISP\_AE\_HIST\_SKIP\_E enHistSkipX;
    ISP\_AE\_HIST\_SKIP\_E enHistSkipY;
    ISP\_AE\_HIST\_OFFSET\_X\_E enHistOffsetX;
    ISP\_AE\_HIST\_OFFSET\_Y\_E enHistOffsetY;
} ISP_AE_HIST_CONFIG_S;
```

【成员】



成员名称	描述
enHistSkipX	直方图统计时水平方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6=every 9th pixel。即取值为 0 时表示每 1 个像素采样一个进行统计, 取值为 1 时表示每 2 个像素采样一个进行统计, 以此类推。0 仅支持 FourPlaneMode 使能时配置。
enHistSkipY	直方图统计时垂直方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6=every 9th pixel。
enHistOffsetX	直方图统计时水平方向起始点设置, 0 表示从第一列开始统计, 1 表示从第二列开始统计。
enHistOffsetY	直方图统计时垂直方向起始点设置, 0 表示从第一行开始统计, 1 表示从第二行开始统计。

【注意事项】

- 可根据 Sensor 输入的 Bayer 数据格式, 合理配置该结构体, 使得全局直方图统计时只计入某单个分量数据。
- RGBIR 模式推荐全局直方图统计时只计入 G 分量, 可根据 RGBIR Sensor 的 Bayer 数据格式合理配置该结构体实现。
- 配置 stHistConfig 相关参数时, 需要把上海海思 AE 的大面积单色判断功能关闭。

【相关数据类型及接口】

无

ISP_AE_STAT_MODE_E

【说明】

定义统计模块的开方模式。

【定义】

```
typedef enum hiISP_AE_STAT_MODE_E
{
    ISP_AE_NORMAL = 0,
    ISP_AE_ROOT,
```



```
ISP_AE_MODE_BUTT  
} ISP_AE_STAT_MODE_E;
```

【成员】

成员名称	描述
ISP_AE_NORMAL	普通模式，不做处理。
ISP_AE_ROOT	开方模式。

【注意事项】

无

【相关数据类型及接口】

[ISP_AE_STATISTICS_CFG_S](#)

ISP_AE_CROP_S

【说明】

对 AE 模块输入图像进行裁剪。

【定义】

```
typedef struct hiISP_AE_CROP_S  
{  
    HI_BOOL        bEnable;  
    HI_U16          u16X;  
    HI_U16          u16Y;  
    HI_U16          u16W;  
    HI_U16          u16H;  
} ISP_AE_CROP_S;
```

【成员】

成员名称	描述
bEnable	使能 Crop，0 为 Crop 不使能，1 为 Crop 使能。
u16X	Crop X 起始位置 取值范围：[0, 图像宽度 - 256]



成员名称	描述
u16Y	Crop Y 起始位置 取值范围: [0, 图像高度 - 120]
u16W	Crop 宽度 取值范围: [256, 图像宽度]
u16H	Crop 高度 取值范围: [120, 图像高度]

【注意事项】

- 当裁剪图像宽小于 600 或高小于 200 时 (其中 Hi3516E V200、Hi3516EV300、Hi3518EV200 当裁剪图像宽小于 760 或高小于 200 时), AE 统计信息仅全局直方图可用。
- 水平方向起始位置与 Crop 宽度之和应小于图像宽度, 垂直方向起始位置与 Crop 高度之和应小于图像高度。
- 对于 BE 的 AE 统计信息, ISP BE 分块场景下, 不支持 AE CROP (输入图像裁剪) 功能。
- AE CROP (输入图像裁剪) 参数会同步对 MG 模块生效。

【相关数据类型及接口】

[ISP_AE_STATISTICS_CFG_S](#)

ISP_AWB_SWITCH_E

【说明】

定义 AWB 统计模块在 ISP pipeline 的位置。

【定义】

```
typedef enum hiISP_AWB_SWITCH_E
{
    ISP_AWB_AFTER_DG = 0 ,
    ISP_AWB_AFTER_Expander,
    ISP_AWB_AFTER_DRC,
    ISP_AWB_SWITCH_BUTT
} ISP_AWB_SWITCH_E;
```



【成员】

成员名称	描述
ISP_AWB_AFTER_DG	AWB 统计模块位置在 Ditial Gain 后
ISP_AWB_AFTER_Expander	AWB 统计模块位置在 Expander 后
ISP_AWB_AFTER_DRC	AWB 统计模块位置在 DRC 后

【注意事项】

无

【相关数据类型及接口】

[ISP_WB_STATISTICS_CFG_S](#)

ISP_AWB_GAIN_SWITCH_E

【说明】

定义 AWB 增益的位置。

【定义】

```
typedef enum hiISP_AWB_GAIN_SWITCH_E {  
    ISP_AWB_GAIN_IN_ISP = 0,  
    ISP_AWB_GAIN_IN_SENSOR,  
    ISP_AWB_GAIN_IN_BUTT  
} ISP_AWB_GAIN_SWITCH_E;
```

【成员】

成员名称	描述
ISP_AWB_GAIN_IN_ISP	AWB 增益在 ISP 生效
ISP_AWB_GAIN_IN_SENSOR	AWB 增益在 Sensor 生效

【注意事项】

无



【相关数据类型及接口】

ISP_WB_STATISTICS_CFG_S

ISP_AWB_CROP_S

【说明】

对 AWB 模块输入图像进行裁剪。

【定义】

```
typedef struct hiISP_AWB_CROP_S
{
    HI_BOOL          bEnable;
    HI_U16           u16X;
    HI_U16           u16Y;
    HI_U16           u16W;
    HI_U16           u16H;
} ISP_AWB_CROP_S;
```

【成员】

成员名称	描述
bEnable	使能 Crop, 0 为 Crop 不使能, 1 为 Crop 使能。
u16X	Crop X 起始位置 取值范围: [0, 图像宽度 - 60 * u16ZoneCol]
u16Y	Crop Y 起始位置 取值范围: [0, 图像高度 - 14 * u16ZoneRow]
u16W	Crop 宽度 取值范围: [60 * u16ZoneCol, 图像宽度]
u16H	Crop 高度 取值范围: [14 * u16ZoneRow, 图像高度]

【注意事项】



- AWB 要求每个分块的最小分辨率为 60*14，当裁剪后的图像分辨率较小，分块数较多时，如果每块的分辨率小于 60x14，那么需要减少分块数，以免每块过小。
- 水平方向起始位置与 Crop 宽度之和应小于图像宽度，垂直方向起始位置与 Crop 高度之和应小于图像高度。
- ISP BE 分块场景下，不支持 AWB CROP（输入图像裁剪）功能。

【相关数据类型及接口】

ISP_WB_STATISTICS_CFG_S

ISP_WB_STATISTICS_CFG_S

【说明】

定义 AWB 统计信息配置。

【定义】

```
typedef struct hiISP_WB_STATISTICS_CFG_S
{
    ISP_AWB_SWITCH_E enAWBSwitch;
    HI_U16 u16ZoneRow;
    HI_U16 u16ZoneCol;
    HI_U16 u16ZoneBin;
    HI_U16 au16HistBinThresh[ AWB_ZONE_BIN_MAX];
    HI_U16 u16WhiteLevel;
    HI_U16 u16BlackLevel;
    HI_U16 u16CbMax;
    HI_U16 u16CbMin;
    HI_U16 u16CrMax;
    HI_U16 u16CrMin;
    ISP_AWB_CROP_S stCrop;
} ISP_WB_STATISTICS_CFG_S;
```



说明

宏定义如下：

```
#define AWB_ZONE_BIN_MAX 4
```

【成员】

成员名称	描述
enAWBSwitch	WB 统计模块位置，默认值为 0，在 Digital Gain 后。



成员名称	描述
u16ZoneRow	WB 统计垂直方向分块个数。取值范围：[0x1, min(0x20, ImageHeight / 14)]。 默认值与图像高度相关(保证每个分块的高度不小于 14)： 如果图像高度大于等于 448，默认值为 0x20 如果图像高度小于 448，默认值为 ImageHeight/14
u16ZoneCol	WB 统计水平方向分块个数，取值范围：[BlkNum, min(0x20, ImageWidth/60)], BlkNum 为 ISP BE 的分块数目。 默认值与图像的宽度相关(保证每个分块的宽度小于 60)： 如果图像宽度大于 1920，默认值为 0x20 如果图像宽度小于 1920，默认值为 ImageWidth/60
u16ZoneBin	WB 统计每个分块的亮度分组个数，默认值 0x1。 Hi3559AV100/Hi3519AV100 取值范围：[0x1, 0x4]， Hi3516CV500/Hi3559V200/Hi3556V200/Hi3516EV200/ Hi3516EV300 中 u16ZoneBin 不可调，只能为 1。
au16HistBinThresh[4]	WB 统计每个分块的亮度分组门限值，要求为单调递增序列，取值范围：[0x0, 0xFFFF]。默认值[0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF]。门限值为 RGB 的加权和，默认为 B 值，即从 au16HistBinThresh[0]到 au16HistBinThresh[3]，B 值逐渐增大。
u16WhiteLevel	统计白点信息时，找白点的亮度上限。取值范围：[0x0, 0xFFFF]，默认值 0xFFFF。
u16BlackLevel	统计白点信息时，找白点的亮度下限。取值范围：[0x0, u16WhiteLevel]，默认值 0x0。
u16CbMax	统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。取值范围：[0x0, 0xFFF]。
u16CbMin	统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。取值范围：[0x0, 0xFFF]。
u16CrMax	统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。取值范围：[0x0, 0xFFF]。



成员名称	描述
u16CrMin	统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。取值范围：[0x0, 0xFF]。
stCrop	AWB 输入图像裁剪配置。

【注意事项】

拼接模式下，每一路的参数是独立配置的。每一路的分块个数都需要配置，而且需要相同。即每一路的 enAWBSwitch, u16ZoneRow, u16ZoneCol, u16ZoneBin 需要相同。

【相关数据类型及接口】

无

ISP_AE_GRID_INFO_S

【说明】

定义 AE 分区间统计信息的坐标信息

【定义】

```
typedef struct hiISP_AE_GRID_INFO_S
{
    HI_U16 au16GridYPos[AE_ZONE_ROW + 1];
    HI_U16 au16GridXPos[AE_ZONE_COLUMN + 1];
    HI_U8  u8Status;
}ISP_AE_GRID_INFO_S;
```

【成员】

成员名称	描述
au16GridYPos	AE 每个分区 Y 方向坐标值。
au16GridXPos	AE 每个分区 X 方向坐标值。
u8Status	AE 统计信息坐标更新状态。



【注意事项】

- 非分块场景下，AE 分区间统计信息为 15*17 个均匀分区的统计信息。
- BE 分块场景下，AE 分区间统计信息的分区大小是不均匀的，若按照均匀分区间的方式显示 AE 统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

【相关数据类型及接口】

无

ISP_AE_STATISTICS_S

【说明】

定义 AE 统计信息。

【定义】

```
typedef struct hiISP_AE_STATISTICS_S
{
    HI_U32 au32FEHist1024Value[ISP_CHN_MAX_NUM][HIST_NUM];
    HI_U16 au16FEGlobalAvg[ISP_CHN_MAX_NUM][BAYER_PATTERN_NUM];
    HI_U16 au16FEZoneAvg[ISP_CHN_MAX_NUM][AE_ZONE_ROW]
[AE_ZONE_COLUMN][BAYER_PATTERN_NUM];
    HI_U32 au32BEHist1024Value[HIST_NUM];
    HI_U16 au16BEGlobalAvg[BAYER_PATTERN_NUM];
    HI_U16 au16BEZoneAvg[AE_ZONE_ROW][AE_ZONE_COLUMN][BAYER_PATTERN_NUM];
    ISP_AE_GRID_INFO_S stFEGridInfo;
    ISP_AE_GRID_INFO_S stBEGridInfo;
}ISP_AE_STATISTICS_S;
```



说明

宏定义如下：

```
#define ISP_CHN_MAX_NUM      (4)
#define BAYER_PATTERN_NUM    (4)
```

【成员】

成员名称	描述
au32FEHist1024Value	FE 全局 1024 段直方图信息。 Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV 200 硬件统计的直方图为 256 段，软件扩充为 1024



成员名称	描述
	段。
au16FEGlobalAvg	FE 全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV200 不支持
au16FEZoneAvg	FE 分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 Hi3516CV500/Hi3516DV300/Hi3516AV300/Hi3559V200/Hi3556V200 ViPipe1、ViPipe3 不支持，帧 WDR 模式数组下标 0 和 2 分别对应 ViPipe1、ViPipe3 的统计信息，行 WDR 模式数组下标 1 和 3 分别对应 ViPipe1、ViPipe3 的统计信息。 Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV200 不支持。
au32BEHist1024Value	BE 全局 1024 段直方图信息。
au16BEGlobalAvg	BE 全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
au16BEZoneAvg	BE 分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
stFEGridInfo	FE 分区间统计信息的坐标信息。
stBEGridInfo	BE 分区间统计信息的坐标信息。

【注意事项】

- WDR 模式下只在主路即最短帧所在 Pipe 才可以获取所有长短帧的统计信息，以 WDR2TO1 为例，ISP_CHN_MAX_NUM 对应下标为 0 时表示短帧统计信息，为 1 时表示长帧信息。
- WDR 合成前(FE)统计信息受到位于 FE AE 之前处理模块（在 Hi3559AV100/Hi3519AV100 中，FE AE 之前的处理模块有 LSC/DG/WB，在 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 中，FE AE 之前的模块有 DG/WB）增益影响，这些模块的 FE 增益值由算法内部保证与 BE 保持一致，无需单独配置。



- 当图像宽小于 600 或高小于 200 时，分区间统计信息不可用。

【相关数据类型及接口】

无

ISP_AE_STITCH_STATISTICS_S

【说明】

定义拼接模式 AE 统计信息。

【定义】

```
typedef struct hiISP_AE_STITCH_STATISTICS_S
{
    HI_U32 au32FEHist1024Value[ISP_CHN_MAX_NUM][HIST_NUM];
    HI_U16 au16FEGlobalAvg[ISP_CHN_MAX_NUM][BAYER_PATTERN_NUM];
    HI_U16
    au16FEZoneAvg[VI_MAX_PIPE_NUM][ISP_CHN_MAX_NUM][AE_ZONE_ROW][AE_ZONE_COLUMN][BAYER_PATTERN_NUM];
    HI_U32 au32BEHist1024Value[HIST_NUM];
    HI_U16 au16BEGlobalAvg[BAYER_PATTERN_NUM];
    HI_U16 au16BEZoneAvg[VI_MAX_PIPE_NUM][AE_ZONE_ROW][AE_ZONE_COLUMN][BAYER_PATTERN_NUM];
}ISP_AE_STITCH_STATISTICS_S;
```

【成员】

成员名称	描述
au32FEHist1024Value	FE 全局 1024 段直方图信息。
au16FEGlobalAvg	FE 全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
au16FEZoneAvg	FE 分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
au32BEHist1024Value	BE 全局 1024 段直方图信息。
au16BEGlobalAvg	BE 全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。



成员名称	描述
au16BEZoneAvg	BE 分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。

【注意事项】

- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持该结构体。
- 拼接后统计信息仅在拼接模式下生效，并且仅在拼接主路可以获取。
- 拼接后的分块均值只有参与拼接的 Pipe 分区间均值有效，其余 Pipe 的均值无效。如 Pipe0 和 Pipe1 进行拼接，则 VI_MAX_PIPE_NUM 对应下标为 0,1 时 au16FEZoneAvg/ au16BEZoneAvg 才有有效统计信息。
- WDR 模式下只在拼接主路即最短帧所在 Pipe 才可以获取所有长短帧的统计信息，以 WDR2TO1 为例，ISP_CHN_MAX_NUM 对应下标为 0 时表示短帧统计信息，为 1 时表示长帧信息。
- WDR 合成前(FE)统计信息受到位于 FE AE 之前处理模块（在 Hi3559AV100/Hi19AV100 中，FE AE 之前的处理模块有 LSC/DG/WB，在 Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 中，FE AE 之前的模块有 DG/WB）增益影响，这些模块的 FE 增益值由算法内部保证与 BE 保持一致，无需单独配置。

【相关数据类型及接口】

无

ISP_MG_GRID_INFO_S

【说明】

定义 MG 分区间统计信息的坐标信息

【定义】

```
typedef struct hiISP_MG_GRID_INFO_S
{
    HI_U16 au16GridYPos[MG_ZONE_ROW + 1];
    HI_U16 au16GridXPos[MG_ZONE_COLUMN + 1];
    HI_U8  u8Status;           //0:not update, 1: update,others:reserved
}ISP_MG_GRID_INFO_S;
```



【成员】

成员名称	描述
au16GridYPos	MG 每个分区 Y 方向坐标值。
au16GridXPos	MG 每个分区 X 方向坐标值。
u8Status	MG 统计信息坐标更新状态。

【注意事项】

- 非分块场景下，MG 分区间统计信息为 15*17 个均匀分区的统计信息。
- BE 分块场景下，MG 分区间统计信息的分区大小是不均匀的，若按照均匀分区间的方式显示 MG 统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

【相关数据类型及接口】

无

ISP_MG_STATISTICS_S

【说明】

定义 MG 统计信息。

【定义】

```
typedef struct hiISP_MG_STATISTICS_S
{
    HI_U16 au16ZoneAvg[MG_ZONE_ROW][MG_ZONE_COLUMN][BAYER_PATTERN_NUM];
    ISP_MG_GRID_INFO_S stGridInfo;
}ISP_MG_STATISTICS_S;
```



说明

宏定义如下：

```
#define MG_ZONE_ROW          (15)
#define MG_ZONE_COLUMN      (17)
```

【成员】



成员名称	描述
au16ZoneAvg	分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。
stGridInfo	MG 分区间统计信息的坐标值。

【注意事项】

无

【相关数据类型及接口】

无

ISP_AWB_GRID_INFO_S

【说明】

定义 AWB 分区间统计信息的坐标信息。

【定义】

```
typedef struct hiISP_AWB_GRID_INFO_S
{
    HI_U16 au16GridYPos[AWB_ZONE_ORIG_ROW + 1];
    HI_U16 au16GridXPos[AWB_ZONE_ORIG_COLUMN + 1];
    HI_U8 u8Status;
}ISP_AWB_GRID_INFO_S;
```

【成员】

成员名称	描述
au16GridYPos	AWB 每个分区 Y 方向坐标值。
au16GridXPos	AWB 每个分区 X 方向坐标值。
u8Status	AWB 统计信息坐标更新状态。

【注意事项】

- 非分块场景下，AWB 分区间统计信息为 $u16ZoneRow * u16ZoneCol$ 个均匀分区的统计信息。



- BE 分块场景下，AWB 分区间统计信息的分区大小是不均匀的，若按照均匀分区的方式显示 AWB 统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

【相关数据类型及接口】

无

ISP_WB_STATISTICS_S

【说明】

定义 AWB 统计信息。

【定义】

```
typedef struct hiISP_WB_STATISTICS_S
{
    HI_U16 u16GlobalR;
    HI_U16 u16GlobalG;
    HI_U16 u16GlobalB;
    HI_U16 u16CountAll;
    HI_U16 au16ZoneAvgR[AWB_ZONE_NUM];
    HI_U16 au16ZoneAvgG[AWB_ZONE_NUM];
    HI_U16 au16ZoneAvgB[AWB_ZONE_NUM];
    HI_U16 au16ZoneCountAll[AWB_ZONE_NUM];
    ISP\_AWB\_GRID\_INFO\_S stGridInfo;
} ISP_WB_STATISTICS_S;
```



说明

宏定义如下：

```
#define AWB_ZONE_ORIG_ROW          (32)
```

```
#define AWB_ZONE_ORIG_COLUMN      (32)
```

Hi3559AV100/Hi3519AV100 中 AWB_ZONE_BIN 的宏定义为：

```
#define AWB_ZONE_BIN              (4)
```

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 中 AWB_ZONE_BIN 的宏定义为：

```
#define AWB_ZONE_BIN              (1)
```

```
#define AWB_ZONE_NUM              (AWB_ZONE_ORIG_ROW *  
AWB_ZONE_ORIG_COLUMN * AWB_ZONE_BIN)
```

【成员】



成员名称	描述
u16GlobalR	Bayer 域全局统计的 R 分量平均值，取值范围：[0x0, 0xFFFF]。
u16GlobalG	Bayer 域全局统计的 G 分量平均值，取值范围：[0x0, 0xFFFF]。
u16GlobalB	Bayer 域全局统计的 B 分量平均值，取值范围：[0x0, 0xFFFF]。
u16CountAll	全局统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。
au16ZoneAvgR	Bayer 域分区间统计的 R 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneAvgG	Bayer 域分区间统计的 G 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneAvgB	Bayer 域分区间统计的 B 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneCountAll	分区间统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。
stGridInfo	Awb 分区间统计信息的坐标信息。

【注意事项】

Hi3559AV100/Hi3519AV100 AWB 区域统计信息排列顺序见[表 7-6](#)。

表7-6 AWB 区域统计信息 (以 au16ZoneAvgR 为例, u16ZoneRow=32, u16ZoneCol=32, u16ZoneBin=4)

Index	Zone	Bin	读取数据说明
0	0	0	第 0 个分块第 0 个 bin 的 Average R
1	0	1	第 0 个分块第 1 个 bin 的 Average R
2	0	2	第 0 个分块第 2 个 bin 的 Average R



Index	Zone	Bin	读取数据说明
3	0	3	第 0 个分块第 3 个 bin 的 Average R
4	1	0	第 1 个分块第 0 个 bin 的 Average R
5	1	1	第 1 个分块第 1 个 bin 的 Average R
6	1	2	第 1 个分块第 2 个 bin 的 Average R
7	2	3	第 1 个分块第 3 个 bin 的 Average R
8	2	0	第 2 个分块第 0 个 bin 的 Average R
9	2	1	第 2 个分块第 1 个 bin 的 Average R
.....			
4095	1023	3	第 1023 个分块第 3 个 bin 的 Average R

表 7-6 是对应 u16ZoneRow=32, u16ZoneCol=32, u16ZoneBin=4 的配置的 au16ZoneAvgR 的数据排布, 排布是紧密排布的, 如果 bin 配置为 1, 则最大 Index 为 1023。

- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 AWB 区域统计信息排列顺序见表 7-7。

表 7-7 AWB 区域统计信息 (以 au16ZoneAvgR 为例, u16ZoneRow=32, u16ZoneCol=32, Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持 1 个 bin)

Index	Zone	Bin	读取数据说明
0	0	0	第 0 个分块的 Average R
1	1	0	第 1 个分块的 Average R
2	2	0	第 2 个分块的 Average R
3	3	0	第 3 个分块的 Average R
4	4	0	第 4 个分块的 Average R
.....			



Index	Zone	Bin	读取数据说明
1023	1023	0	第 1023 个分块的 Average R

- au16ZoneCountAll 做归一化是为了消除分辨率差异对灰点个数的影响。归一化公式： $\text{CountAll} = (\text{Count of Gray Pixels} \ll 16) / (\text{Count of All Pixels})$ 。
- WB 统计每个分块的亮度分组门限值，要求为单调递增序列，取值范围：[0x0, 0xFFFF]。门限值为 RGB 的加权和，默认为 B 值，即从 bin[0]到 bin[3]，B 值逐渐增大。

【相关数据类型及接口】

无

ISP_WB_STITCH_STATISTICS_S

【说明】

定义拼接模式 WB 统计信息。

【定义】

```
typedef struct hiISP_WB_STITCH_STATISTICS_S
{
    HI_U16  u16ZoneRow;
    HI_U16  u16ZoneCol;
    HI_U16  au16ZoneAvgR[AWB_ZONE_STITCH_MAX];
    HI_U16  au16ZoneAvgG[AWB_ZONE_STITCH_MAX];
    HI_U16  au16ZoneAvgB[AWB_ZONE_STITCH_MAX];
    HI_U16  au16ZoneCountAll[AWB_ZONE_STITCH_MAX];
} ISP_WB_STITCH_STATISTICS_S;
```



说明

宏定义如下：

Hi3559AV100 中 ISP_MAX_STITCH_NUM 宏定义：

```
#define ISP_MAX_STITCH_NUM      (8)
```

Hi3519AV100 中 ISP_MAX_STITCH_NUM 宏定义：

```
#define ISP_MAX_STITCH_NUM      (4)
```

Hi3516CV500/Hi3559V200 中 ISP_MAX_STITCH_NUM 宏定义：

```
#define ISP_MAX_STITCH_NUM      (1)
```



```
#define AWB_ZONE_STITCH_MAX (AWB_ZONE_NUM *ISP_MAX_STITCH_NUM)
```

【成员】

成员名称	描述
u16ZoneRow	拼接统计分块行数，取值范围：[0x1,0x20]。
u16ZoneCol	拼接统计分块列数，取值范围：[0x1,0x100]。
au16ZoneAvgR	Bayer 域分区间统计的 R 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneAvgG	Bayer 域分区间统计的 G 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneAvgB	Bayer 域分区间统计的 B 分量平均值，取值范围：[0x0, 0xFFFF]。
au16ZoneCountAll	分区间统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。

【注意事项】

- Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持该结构体参数的获取
- AWB 统计信息的排列顺序请参考[表 7-6](#)。

【相关数据类型及接口】

无

ISP_FOCUS_STATISTICS_CFG_S

【说明】

定义 AF 统计信息配置。

【定义】

```
typedef struct hiISP_FOCUS_STATISTICS_CFG_S
{
    ISP_AF_CFG_S      stConfig;
    ISP_AF_H_PARAM_S  stHParam_IIR0;
```



```
ISP_AF_H_PARAM_S    stHParam_IIR1;  
ISP_AF_V_PARAM_S    stVParam_FIR0;  
ISP_AF_V_PARAM_S    stVParam_FIR1;  
ISP_AF_FV_PARAM_S   stFVParam;  
} ISP_FOCUS_STATISTICS_CFG_S;
```

【成员】

成员名称	描述
stConfig	AF 全局配置参数。
stHParam_IIR0	奇数行水平滤波器 IIR 参数设置。
stHParam_IIR1	偶数行水平滤波器 IIR 参数设置。
stVParam_FIR0	奇数行垂直滤波器 FIR 参数设置。
stVParam_FIR1	偶数行垂直滤波器 FIR 参数设置。
stFVParam	统计结果输出格式参数配置。

【注意事项】

- 当分辨率较小，分块数较多时，如果每块的分辨率小于 32x32，那么需要减少分块数，以免每块过小。建议在分辨率小的情况下，使用较少的分块数。
- Hi3562V100/Hi3566V100 不支持 AF 统计信息。

【相关数据类型及接口】

无

ISP_AF_CFG_S

【说明】

定义 AF 统计信息配置参数。

【定义】

```
typedef struct hiISP_AF_CFG_S  
{  
    HI_BOOL          bEnable;  
    HI_U16           u16Hwnd;
```



```

    HI_U16                u16Vwnd;
    HI_U16                u16Hsize;
    HI_U16                u16Vsize;
    ISP_AF_PEAK_MODE_E    enPeakMode;
    ISP_AF_SQU_MODE_E     enSquMode;
    ISP_AF_CROP_S         stCrop;
    ISP_AF_STATISTICS_POS_E enStatisticsPos;
    ISP_AF_RAW_CFG_S      stRawCfg;
    ISP_AF_PRE_FILTER_CFG_S stPreFltCfg;
    HI_U16                u16HighLumaTh;
} ISP_AF_CFG_S;

```

【成员】

成员名称	描述
bEnable	AF 使能。
u16Hwnd	AF 统计水平方向窗口个数，取值范围：[1, 17]。
u16Vwnd	AF 统计垂直方向窗口个数，取值范围：[1, 15]。
u16Hsize	AF 统计输入图像宽度，取值范围：[256, ImageWidth]。 该参数目前不生效。
u16Vsize	AF 统计输入图像高度，取值范围：[120, ImageHeight]。 该参数目前不生效。
enPeakMode	PEAK 模式，决定分区间统计值是否进行求峰值处理。与 enSquMode 共同影响分区间统计值。
enSquMode	平方模式，决定分区间水平和垂直方向统计值是否进行平方处理。与 enPeakMode 共同影响分区间统计值。
stCrop	AF 输入图像的裁剪配置。
enStatisticsPos	AF 统计信息位置配置，决定是在 Bayer 还是 YUV 域统计。
stRawCfg	AF Bayer 域配置，如果用户选择将 AF 模块放在 Bayer 域，那么需要配置这里。
stPreFltCfg	AF 模块预滤波配置，此模块可以帮助消除画面中的椒盐噪声。



成员名称	描述
u16HighLumaTh	AF 高亮点统计值 Threshold 设置，取值范围：[0,0xFF]。

【注意事项】

- Hi3562V100/Hi3566V100 不支持 AF 统计信息。分区间统计值的计算方式为：
 - enPeakMode 为 ISP_AF_STA_NORM 且 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，对分区间内每个像素的滤波器输出值求和即为分区间统计值；
 - enPeakMode 为 ISP_AF_STA_NORM 且 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值；
 - enPeakMode 为 ISP_AF_STA_PEAK 且 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间统计值；
 - enPeakMode 为 ISP_AF_STA_PEAK 且 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。
- 分区间亮度统计值 (u16y) 的计算方式为：
对分区间内每个像素的亮度求和即为分区间亮度统计值 (u16y)。
- 当 VI 处于分块方式时，AF 统计信息的水平方向窗口个数最小为 BlkNum 分块的个数，此时取值范围为[BlkNum, 17]。

【相关数据类型及接口】

无

ISP_AF_PEAK_MODE_E

【说明】

PEAK 模式，决定分区间统计值是否进行求峰值处理。

【定义】

```
typedef enum hiISP_AF_PEAK_MODE_E
{
    ISP_AF_STA_NORM        = 0,
```



```
ISP_AF_STA_PEAK      ,  
ISP_AF_STA_BUTT  
} ISP_AF_PEAK_MODE_E;
```

【成员】

成员名称	描述
ISP_AF_STA_NORM	- 当 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，对分区间内每个像素的输出值求和即为分区间统计值； - 当 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值。
ISP_AF_STA_PEAK	- 当 enSquMode 为 ISP_AF_STA_SUM_NORM 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间统计值； - 当 enSquMode 为 ISP_AF_STA_SUM_SQU 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。

【注意事项】

无

【相关数据类型及接口】

无

ISP_AF_SQU_MODE_E

【说明】

平方模式，决定分区间统计值是否进行平方处理。

【定义】

```
typedef enum hiISP_AF_SQU_MODE_E  
{  
    ISP_AF_STA_SUM_NORM      = 0,  
    ISP_AF_STA_SUM_SQU      ,  
    ISP_AF_STA_SUM_BUTT
```



```
}ISP_AF_SQU_MODE_E;
```

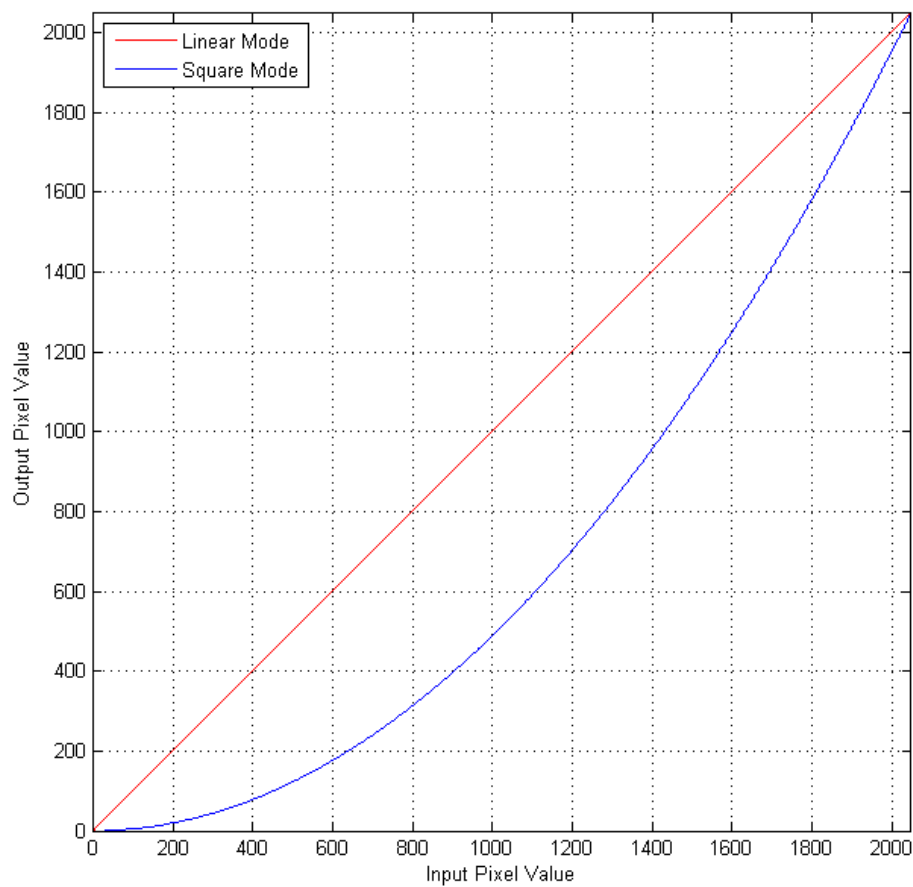
【成员】

成员名称	描述
ISP_AF_STA_SUM_NORM	- 当 enPeakMode 为 ISP_AF_STA_NORM 模式时，对分区间内每个像素的滤波器输出值求和即为分区间统计值； - 当 enPeakMode 为 ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间 IIR 统计值。
ISP_AF_STA_SUM_SQU	- 当 enPeakMode 为 ISP_AF_STA_NORM 模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值； - 当 enPeakMode 为 ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。

【注意事项】

采用 Square 模式，会对滤波器输出归一化后平方再做统计，如图 7-2 所示，input 为滤波后输出像素值，output 为处理后输出值，相比 normal 模式，在临近焦点附近可以得到比较陡峭的 FV 曲线，并且对幅值较小的噪声信号有一定的抑制作用，但也会使 FV 曲线平坦区域更加平坦，用户应该根据实际的场景需求来设定这个参数。

图7-2 Square 模式



【相关数据类型及接口】

无

ISP_AF_CROP_S

【说明】

对 AF 模块输入图像进行裁剪。

【定义】

```
typedef struct hiISP_AF_CROP_S
{
    HI_BOOL        bEnable;
    HI_U16         u16X;
    HI_U16         u16Y;
    HI_U16         u16W;
    HI_U16         u16H;
}
```




```
} ISP_AF_CROP_S;
```

【成员】

成员名称	描述
bEnable	使能 Crop，取值范围：[0, 1]
u16X	Crop X 起始位置，取值范围：[0, ImageWidth - 256]
u16Y	Crop Y 起始位置，取值范围：[0, ImageHeight - 120]
u16W	Crop 宽度，8 对齐，取值范围：[256, ImageWidth]
u16H	Crop 高度，2 对齐，取值范围：[120, ImageHeight]

【注意事项】

- 水平方向起始位置与 Crop 宽度之和应小于图像宽度，垂直方向起始位置与 Crop 高度之和应小于图像高度。
- 对于 BE 的 AF 统计信息，ISP BE 分块场景下，不支持 AF CROP（输入图像裁剪）功能。

【相关数据类型及接口】

无

ISP_AF_STATISTICS_POS_E

【说明】

配置 AF 模块统计位置。

【定义】

```
typedef enum hiISP_AF_STATISTICS_POS_E
{
    ISP_AF_STATISTICS_AFTER_DGAIN    = 0,
    ISP_AF_STATISTICS_AFTER_DRC      ,
    ISP_AF_STATISTICS_AFTER_CSC      ,
    ISP_AF_STATISTICS_BUTT
} ISP_AF_STATISTICS_POS_E;
```

【成员】



成员名称	描述
ISP_AF_STATISTICS_AFTER_DGAIN	将 AF 模块放在 DGain 后进行统计，属于 RAW 域。
ISP_AF_STATISTICS_AFTER_DRC	将 AF 模块放在 DRC 及 Dither 后进行统计，属于 RAW 域。
ISP_AF_STATISTICS_AFTER_CSC	将 AF 模块放在 CSC 后进行统计，属于 YUV 域。

【注意事项】

- WDR 模式下，非宽动态场景的 Bayer 域（即 RAW 域）统计信息无明显变化趋势，推荐使用 YUV 域统计信息。
- AF 的 FE 统计信息只支持 DGain 之后进行统计，而 BE 统计信息支持 DGain、DRC 和 CSC 之后进行统计，详见《HiSP 图像调优指南》文档的 ISP 功能框图章节。

【相关数据类型及接口】

无

ISP_AF_RAW_CFG_S

【说明】

AF 模块 Bayer 域配置。

【定义】

```
typedef struct hiISP_AF_RAW_CFG_S
{
    HI_U8          GammaGainLimit;
    HI_U8          GammaValue;
    ISP_BAYER_FORMAT_E enPattern;
} ISP_AF_RAW_CFG_S;
```

【成员】



成员名称	描述
GammaGainLimit	AF 模块 gamma 限制。 取值范围：[0x0, 0x5] Hi3516EV200/Hi3516EV300 不支持。
GammaValue	AF 模块 gamma 类型。 取值范围：[0x0, 0x6] Hi3516EV200/Hi3516EV300 取值范围[0, 1]。
enPattern	设置 Bayer pattern，支持 BAYER_RGGB，BAYER_GRGB，BAYER_GBRG，BAYER_BGGR 四种。

【注意事项】

无

【相关数据类型及接口】

无

ISP_AF_PRE_FILTER_CFG_S

【说明】

AF 模块预滤波器配置。

【定义】

```
typedef struct hiISP_AF_PRE_FILTER_CFG_S
{
    HI_BOOL          bEn;
    HI_U16           u16strength;
} ISP_AF_PRE_FILTER_CFG_S;
```

【成员】

成员名称	描述
bEn	使能 AF 预滤波模块。
u16strength	设置滤波强度，取值范围：[0x0, 0xFFFF]。



【注意事项】

无

【相关数据类型及接口】

无

ISP_AF_LD_S

【说明】

AF Level Depend Gain 模块配置。

【定义】

```
typedef struct hiISP_AF_LD_S
{
    HI_BOOL    bLdEn;
    HI_U16     u16ThLow;
    HI_U16     u16GainLow;
    HI_U16     u16SlpLow;
    HI_U16     u16ThHigh;
    HI_U16     u16GainHigh;
    HI_U16     u16SlpHigh;
} ISP_AF_LD_S;
```

【成员】

成员名称	描述
bEn	使能 Level Depend Gain 模块。
u16ThLow	设置 Dark Side Threshold，取值范围：[0x0, 0xFF]。
u16GainLow	设置 Dark Side Minimum Gain，取值范围：[0x0, 0xFF]。
u16SlpLow	设置 Dark Side Slope，取值范围：[0x0, 0xF]。
u16ThHigh	设置 Bright Side Threshold，取值范围：[0x0, 0xFF]。
u16GainHigh	设置 Brigh Side Minimum Gain，取值范围：[0x0, 0xFF]。
u16SlpHigh	设置 Brigh Side Slope，取值范围：[0x0, 0xF]。



【注意事项】

无

【相关数据类型及接口】

无

ISP_AF_CORING_S

【说明】

AF Coring 模块配置。

【定义】

```
typedef struct hiISP_AF_CORING_S
{
    HI_U16      u16Th;
    HI_U16      u16Slp;
    HI_U16      u16Lmt;
} ISP_AF_CORING_S;
```

【成员】

成员名称	描述
u16Th	设置 Coring Threshold, 取值范围: [0x0, 0x7FF]。
u16Slp	设置 Coring Slope, 取值范围: [0x0, 0xF]。
u16Lmt	设置 Coring Maximum Output, 取值范围: [0x0, 0x7FF]。

【注意事项】

无

【相关数据类型及接口】

无

ISP_AF_H_PARAM_S

【说明】



AF 统计水平滤波器 IIR 参数设置。

【定义】

```
typedef struct hiISP_AF_H_PARAM_S
{
    HI_BOOL          bNarrowBand;
    HI_BOOL          abIIREn[IIR_EN_NUM];
    HI_U8            u8IIRShift;
    HI_S16           as16IIRGain[IIR_GAIN_NUM];
    HI_U16           au16IIRShift[IIR_SHIFT_NUM];
    ISP_AF_LD_S      stLd;
    ISP_AF_CORING_S  stCoring;
} ISP_AF_H_PARAM_S;
```



说明

宏定义如下：

```
#define IIR_EN_NUM      (3)
#define IIR_GAIN_NUM    (7)
#define IIR_SHIFT_NUM   (4)
```

【成员】

成员名称	描述
bNarrowBand	IIR 滤波器窄带宽使能。
abIIREn[IIR_EN_NUM]	IIR 级联中 IIR 的使能。
u8IIRShift	IIR 滤波器整体位移量,取值范围：[0, 63] Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。
as16IIRGain[IIR_GAIN_NUM]	IIR 滤波器系数，用于控制 IIR 滤波器的频率响应。 取值范围：Gain0:[0, 255]；Others:[-511, 511]。
au16IIRShift[IIR_SHIFT_NUM]	IIR 滤波器移位调整，取值范围：[0, 0x7]。
stLd	IIR 滤波器 Level Depend Gain 模块配置。
stCoring	IIR 滤波器 Coring 模块配置。



【注意事项】

as16IIRGain[0]为无符号 8bit 数，其他 as16IIRGain[1]-[6]为有符号 10bit 数，且不能等于-512。IIR 使能、bNarrowBand 使能、滤波器系数以及 IIRShift 通过 PQ Tools 可以生成，用户可以根据实际需要调节 LDG 和 Coring 模块以获取最优 FV 统计值输出。

【相关数据类型及接口】

无

ISP_AF_V_PARAM_S

【说明】

AF 统计垂直滤波器 FIR 参数设置。

【定义】

```
typedef struct hiISP_AF_V_PARAM_S
{
    HI_S16          as16FIRH[FIR_GAIN_NUM];
    ISP_AF_LD_S     stLd;
    ISP_AF_CORING_S stCoring;
} ISP_AF_V_PARAM_S;
```



说明

宏定义如下：

```
#define FIR_GAIN_NUM    (5)
```

【成员】

成员名称	描述
as16FIRH[FIR_GAIN_NUM]	FIR 滤波器系数，取值范围：[-31, 31]。用于控制 FIR 滤波器的频率响应。
stLd	FIR 滤波器 Level Depend Gain 模块配置。
stCoring	FIR 滤波器 Coring 模块配置。

【注意事项】

FIR 滤波器系数通过 PQ Tools 可以生成，用户可以根据实际需要调节 LDG 和 Coring 模块以获取最优垂直 FV 统计值输出。



【相关数据类型及接口】

无

ISP_AF_FV_PARAM_S

【说明】

AF 统计结果输出格式参数。

【定义】

```
typedef struct hiISP_AF_FV_PARAM_S
{
    HI_U16 u16AccShiftY;
    HI_U16 au16AccShiftH[ACC_SHIFT_H_NUM];
    HI_U16 au16AccShiftV[ACC_SHIFT_V_NUM];
    HI_U16 u16HlCntShift;
} ISP_AF_FV_PARAM_S;
```



说明

宏定义如下：

- #define ACC_SHIFT_H_NUM (2)
- #define ACC_SHIFT_V_NUM (2)

【成员】

成员名称	描述
u16AccShiftY	亮度 Y 统计值移位寄存器值，取值范围：[0, 0xF]。
au16AccShiftH[ACC_SHIFT_H_NUM]	水平 IIR 滤波值统计值移位寄存器值，取值范围：[0, 0xF]。
au16AccShiftV[ACC_SHIFT_V_NUM]	垂直 FIR 滤波值统计值移位寄存器值，取值范围：[0, 0xF]。
u16HlCntShift	高亮计数器统计值移位寄存器值，取值范围：[0, 0xF]。

【注意事项】



FV 值用 unsigned 16bit 表示，当场景细节较多或噪声较多时，FV 值可能会溢出，这就需要对输出进行向右移位以保证输出在合理范围。用户根据自己的需求对相应的 shift 值进行调节。

【相关数据类型及接口】

无

ISP_FOCUS_ZONE_S

【说明】

定义 AF 统计信息参数。

【定义】

```
typedef struct hiISP_FOCUS_ZONE_S
{
    HI_U16  u16v1;
    HI_U16  u16h1;
    HI_U16  u16v2;
    HI_U16  u16h2;
    HI_U16  u16y;
    HI_U16  u16H1Cnt;
} ISP_FOCUS_ZONE_S;
```

【成员】

成员名称	描述
u16v1	分区间统计的 AF 垂直方向奇数列 FIR 滤波器的统计值。
u16h1	分区间统计的 AF 水平方向奇数行 IIR 滤波器的统计值。
u16v2	分区间统计的 AF 垂直方向偶数列 FIR 滤波器的统计值。
u16h2	分区间统计的 AF 水平方向偶数行 IIR 滤波器的统计值。
u16y	分区间统计的 AF 亮度统计值，区块中每个像素点的亮度累加值。
u16H1Cnt	分区间统计的 AF 亮度超阈值计数器值。

【注意事项】



u16y 为每个块的 Y 分量累加值，用户可以用此值辅助 AF 算法，比如某些块中含有高亮光源，容易对 FV 值产生干扰，可根据亮度情况降低块 FV 对全局 FV 的影响，亦可用这些块实现一个移动侦测算法，用于检测聚焦过程中的一些运动干扰，用户可以根据自己 AF 算法的需求使用这个字段的统计信息。

【相关数据类型及接口】

无

ISP_FE_FOCUS_STATISTICS_S

【说明】

定义 FE AF 统计信息。

【定义】

```
typedef struct hiISP_FE_FOCUS_STATISTICS_S
{
    ISP_FOCUS_ZONE_S stZoneMetrics[WDR_CHN_MAX][AF_ZONE_ROW][AF_ZONE_COLUMN];
} ISP_FE_FOCUS_STATISTICS_S;
```



说明

宏定义如下：

- #define WDR_CHN_MAX (4)
- #define AF_ZONE_ROW (15)
- #define AF_ZONE_COLUMN (17)

【成员】

成员名称	描述
stZoneMetrics	ISP AF FE 分块统计信息。

【注意事项】

Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 不支持。

【相关数据类型及接口】

无

ISP_BE_FOCUS_STATISTICS_S

【说明】



定义 BE AF 统计信息。

【定义】

```
typedef struct hiISP_BE_FOCUS_STATISTICS_S
{
    ISP_FOCUS_ZONE_S stZoneMetrics[AF_ZONE_ROW][AF_ZONE_COLUMN];
} ISP_BE_FOCUS_STATISTICS_S;
```

【成员】

成员名称	描述
stZoneMetrics	ISP AF BE 分块统计信息。

【注意事项】

无

【相关数据类型及接口】

无

ISP_FOCUS_GRID_INFO_S

【说明】

定义 AF 分区间统计信息的坐标信息。

【定义】

```
typedef struct hiISP_FOCUS_GRID_INFO_S
{
    HI_U16 au16GridYPos[AF_ZONE_ROW + 1];
    HI_U16 au16GridXPos[AF_ZONE_COLUMN + 1];
    HI_U8 u8Status;
}ISP_FOCUS_GRID_INFO_S;
```

【成员】

成员名称	描述
au16GridYPos	AF 每个分区 Y 方向坐标值。



成员名称	描述
au16GridXPos	AF 每个分区 X 方向坐标值。
u8Status	AF 统计信息坐标更新状态。

【注意事项】

- Hi3562V100/Hi3566V100 不支持。
- 非分块场景下，AF 分区间统计信息为 u16Hwnd * u16Vwnd 个均匀分区的统计信息。
- BE 分块场景下，AF 分区间统计信息的分区大小是不均匀的，若按照均匀分区间的方式显示 AF 统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

【相关数据类型及接口】

无

ISP_AF_STATISTICS_S

【说明】

定义 FE 和 BE 的 AF 统计信息。

【定义】

```
typedef struct hiISP_AF_STATISTICS_S
{
    ISP_FE_FOCUS_STATISTICS_S    stFEAFStat;
    ISP_BE_FOCUS_STATISTICS_S    stBEAFStat;
    ISP_FOCUS_GRID_INFO_S        stFEAFGridInfo;
    ISP_FOCUS_GRID_INFO_S        stBEAFGridInfo;
} ISP_AF_STATISTICS_S;
```

【成员】

成员名称	描述
stFEAFStat	ISP FE 的 AF 统计信息。仅 Hi3559AV100/Hi3519AV100 支持。
stBEAFStat	ISP BE 的 AF 统计信息。



成员名称	描述
stFEAFGridInfo	ISP FE 的 AF 分区间统计信息的坐标。仅 Hi3559AV100/Hi3519AV100 支持。
stBEAFGridInfo	ISP BE 的 AF 分区间统计的信息坐标。

【注意事项】

- Hi3559AV100/Hi3519AV100 性能上超过 4K (3840*2160) 60fps 或图像宽度大于 4096 的数据流只支持 FE 的统计信息。
- Hi3559AV100/Hi3519AV100 线性模式下离线模式推荐使用 FE 的统计信息(实时性更佳)。
- WDR 模式下推荐使用 BE DRC 后的统计信息(参考 [ISP_AF_STATISTICS_POS_E](#))。
- Hi3559AV100 中 AF 支持从 FE0、FE1、FE4、FE5 四个通道获取统计信息，不支持从 FE2、FE3、FE6、FE7 四个通道获取统计信息；Hi3519AV100 中 AF 支持从 FE0、FE1 两个通道获取统计信息，不支持从其他 FE 通道获取统计信息。请参考《HiISP 图像调优指南》第 5 页关于 ISP FE 的说明。
- Hi3562V100/Hi3566V100 不支持 AF 统计信息。

【相关数据类型及接口】

无



8 Debug

8.1 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。

8.2 功能描述

Debug 提供记录 ISP 在运行过程中的状态信息，以方便记录在 ISP 运行过程中出现的异常状态，需要记录的帧数可以有用户自己指定。

8.3 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(VI_PIPE ViPipe, const ISP\_DEBUG\_INFO\_S * pstIspDebug);
```

【参数】



参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstlspDebug	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 。

【需求】

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

【注意】

- 调试信息的成员变量为 SYS，可以设置使能开关。
- 需要用户来分配内存以存储相应的调试信息，且需要将分配的内存地址作为输入传入。

【举例】

```
HI_S32 s32Ret = 0;
HI_U32  au32Depth  = 10;
HI_U32  au32Size   = 0;
ISP_DEBUG_INFO_S  stDebug;
HI_VOID *apVitAddr  = HI_NULL;    /* virt addr malloc memory */
HI_VOID *apVirtAddr  = HI_NULL;    /* virt addr mmap */
au32Size = sizeof(ISP_DBG_ATTR_S) + sizeof(ISP_DBG_STATUS_S) * au32Depth;
s32Ret = HI_MPI_SYS_MmzAlloc_Cached(&au64PhyAddr, &apVitAddr, HI_NULL, HI_NULL,
au32Size);
if (HI_SUCCESS != s32Ret)
{
    printf("Buf not enough!\n");
    return HI_FAILURE;
}
```



```
    }

    stDebug.bDebugEn = HI_TRUE;
    stDebug.u32Depth = au32Depth;
    stDebug.u64PhyAddr = au64PhyAddr;
    s32Ret = HI_MPI_ISP_SetDebug(ViPipe, &stDebug);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_ISP_SetDebug failed 0x%x!\n", s32Ret);
        return HI_FAILURE;
    }
}
```

【相关主题】

无

HI_MPI_ISP_GetDebug

【描述】

获取 ISP 调试信息属性。

【语法】

```
HI_S32 HI_MPI_ISP_GetDebug(VI_PIPE ViPipe, ISP\_DEBUG\_INFO\_S * pstIspDebug);
```

【参数】

参数名称	描述	输入/输出
ViPipe	ViPipe 号。	输入
pstIspDebug	调试信息。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为 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

8.4 数据类型

[ISP_DEBUG_INFO_S](#)：定义 ISP 调试信息属性。

ISP_DEBUG_INFO_S

【说明】

定义 ISP 调试信息属性。

【定义】

```
typedef struct hiISP_DEBUG_INFO_S
{
    HI_BOOL bDebugEn;
    HI_U64  u64PhyAddr;
    HI_U32  u32Depth;
} ISP_DEBUG_INFO_S;
```

【成员】

成员名称	描述
bDebugEn	使能调试。
U64PhyAddr	调试信息的物理地址。
u32Depth	调试深度，即需要获取调试信息的帧数。

【注意事项】



无

【相关数据类型及接口】

无



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
0xA01C8006	HI_ERR_ISP_NULL_PTR	空指针错误
0xA01C8003	HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效
0xA01C8008	HI_ERR_ISP_NOT_SUPPORT	当前 ISP 不支持
0xA01C8043	HI_ERR_ISP_SNS_UNREGISTER	Sensor 未注册
0xA01C8041	HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
0xA01C8040	HI_ERR_ISP_NOT_INIT	ISP 没有初始化
0xA01C8044	HI_ERR_ISP_INVALID_ADDR	无效地址
0xA01C8042	HI_ERR_ISP_ATTR_NOT_CFG	属性未配置
0xA01C8045	HI_ERR_ISP_NOMEM	内存不足
0xA01C8046	HI_ERR_ISP_NO_INT	ISP 无中断



10 Proc 调试信息说明



说明

- 未有特殊说明, Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明, Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明, Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明, Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明, Hi3559V200、Hi3562V100、Hi3566V100 与 Hi3556V200 内容一致

10.1 概述

调试信息采用了 proc 文件系统, 可实时反映当前系统的运行状态, 所记录的信息可供问题定位及分析时使用。

【文件目录】

/proc/umap/isp

【开启方法】

- 调用 [HI_MPI_ISP_SetCtrlParam](#) 设置 ISP 控制参数 u32ProcParam = n, 其中 n=0 关闭 proc, 非 0 表示每隔 n 帧更新一次 proc。
- 调用 [HI_MPI_ISP_SetCtrlParam](#) 设置 u32UpdatePos = 0 或 u32UpdatePos = 1, 默认值为 0
u32UpdatePos = 0 表示 ISP 中断配置寄存器的位置是使用帧起始中断
u32UpdatePos = 1 表示 ISP 中断配置寄存器的位置是使用帧结束中断
- 调用 [HI_MPI_ISP_SetModParam](#) 设置 u32IntBotHalf 中断底半部, 默认值为 0



u32IntBotHalf =0 表示 ISP 内核态处理（读统计信息和配置 sensor 和 ISP 同步寄存器）在中断服务程序中完成

u32IntBotHalf =1 表示 ISP 内核态处理（读统计信息和配置 sensor 和 ISP 同步寄存器）在中断下半部完成

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】

```
[ISP] Version: [Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200_MPP_V1.0.0.0 B010
Release], Build Time[Mar  3 2018, 09:38:17]
```

```
----- ISP PROC PIPE[0] -----
```

```
-----MODULE/CONTROL PARAM-----
```

ProcParam	StatIntvl	UpdatePos	IntBothalf	IntTimeout	PwmNumber
30	1	0	0	200	3

PortIntDelay	QuickStart	LdciTprFltEn	LongFrmlntEn	BeBufNum
0	0	0	0	8

```
-----ISP Mode-----
```



	StitchMode		RunningMode		BlockNum					
	NORMAL		ONLINE		1					
-----SENSOR INFO-----										
	SensorType		Dev							
	I2C		0							
-----DRV INFO-----										
	ViPipe	IntCnt	IntT	MaxIntT	IntGapT	MaxGapT	IntRat	IspResetCnt		
IspBeStaLost										
	0	629	1898	2948	33372	33546	30			
2	0									
		PtIntCnt	PtIntT	PtMaxIntT	PtIntGapT	PtMaxGapT	PtIntRat	SensorCfgT		
SensorMaxT										
		0	0	0	0	0	0	3		
2913										
-----BeCfg Phy Addr-----										
	BeCfg[0]		BeCfg[1]		BeCfg[2]		BeCfg[3]			
	0x86b7c000		0x86c21900		0x86cc7200		0x86d6cb00			
-----PubAttr INFO-----										
	WndX	WndY	WndW	WndH		SnsW		SnsH		
Bayer										
	0	0	1920	1080	1920		1080		RGGB	
-----SNAPATTR INFO-----										
	SnapType	PipeMode	OPType	ProFrmNum						
	NORMAL	NONE	Auto	3						
[AE] Version: [Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200_MPP_V1.0.0.0 B010										
Release], Build Time[Mar 3 2018, 14:05:18]										
-----AE INFO-----										
	Again	Dgain	IspDg	SysGain	Iso	Line	AEInter	Incrmnt	Exp	1stTime
	8439	1024	1084	8604	872	1124	1	256	9661965	
466666										
	Comp	EVbias	OriAve	Offset	Speed	Tole	Error	Fps	RealFps	BDelay



```

WDelay
    56    1024    42    16    64    2    -2    30.00    3000    8
0

MaxLineMaxLineT  MaxAgT  MaxDgT  MaxIDgT  MaxSgT  ManuEn  MaLine  MaAg
MaDg  MalspDg
1124  65535  62564  20013  2048 2445495    0    0    0    0    0
    WdrMode  WdrGain[0] WdrGain[1] WdrGain[2] WdrGain[3]  ExpRatio0
    LINE      256      256      256      256      64
ExpRatio1 ExpRatio2 AnFlick SlowMod  GainTh
    64      64      0      1      2445495

NodeId IntTime SysGain IrisApe  UpStgy  DwStgy  Mltply
    0      1    1024      1      0      4      1024
    1    1124    1024      1      1      0    1150976
    2    1124 2445495      1      4      1 2748736380

AulrEn  IrType  MalrEn DbglrSt
    0  DClris      0      0

[AWB] Version: [Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200_MPP_V1.0.0.0 B010
Release], Build Time[Mar  3 2018, 14:05:26]

-----AWB INFO-----
    Gain0  Gain1  Gain2  Gain3  CoTemp
    0x1d2  0x100  0x100  0x1e6  4587

Color00 Color01 Color02 Color10 Color11 Color12 Color20 Color21 Color22
0x201  0x80f8  0x8009  0x803b  0x176  0x803b  0xd  0x8099  0x18c

    ManuEn  Sat  Zones  Speed
    0      123    32    256

-----DPC INFO-----
    Enable  Strength  BlendRatio
    1      0      0

-----CrossTalk INFO-----
    Enable  NpOffset  Threshold  Strength
    1      1024    300      128

```



```
-----FrameWDR INFO-----
                MdtEn        LongThr        ShortThr        MdThrLowGain        MdThrHighGain
                1            3600            4032            0                    0

-----Black Level Actual INFO-----
                BlcR          BlcGr          BlcGb          BlcB
                240          240          240          240

-----BAYERNR INFO-----
                Enable        NrLscEnable        NrLscRatio        CoarseStr0        CoarseStr1
CoarseStr2        CoarseStr3
                1            0            0            0            0
0                0

-----DRC INFO-----
                En          ManuEn        Strength
                0          0            255

-----LCAC INFO-----
                Enable        CrStr        CbStr
                1            0            3

-----DEMOSAIC INFO-----
                Enable        NonDirStr        NonDirMfStr        NonDirHfStr        DeSmthRange
                1            48            82            0            2
ColorNoiseThdF        ColorNoiseStrF        ColorNoiseThdY        ColorNoiseStrY
                0            8            30            0

-----Anti False Color INFO-----
                Enable        Threshold        Strength
                1            10            12

-----CA INFO-----
                Enable        IsoRatio
                1            1300

-----FPN CORRECT INFO-----
```




En	OpType	Strength	Offset				
0	--	--	--				
-----SHARPEN INFO-----							
bSharpenEn							
1							
LumaWgt 0--7:							
127	127	127	127	127	127	127	127
LumaWgt 8--15:							
127	127	127	127	127	127	127	127
LumaWgt 16--23:							
127	127	127	127	127	127	127	127
LumaWgt 24--31:							
127	127	127	127	127	127	127	127
TextureStr 0--7:							
244	244	255	245	250	250	250	255
TextureStr 8--15:							
265	269	279	279	284	284	294	294
TextureStr 16--23:							
299	299	299	299	299	284	284	269
TextureStr 24--31:							
260	255	240	230	220	200	200	180
EdgeStr 0--7:							
190	190	190	190	190	195	195	200
EdgeStr 8--15:							
210	220	230	240	250	260	270	280
EdgeStr 16--23:							
290	280	270	260	250	245	245	235



```

EdgeStr 24--31:
    225    210    200    190    185    185    175    175

TextureFreq    EdgeFreq    OverShoot    UnderShoot    ShootSupStr    DetailCtrl
200           128           97           117           29           128
EdgeFiltStr    EdgeFiltMaxCap    RGain    Ggain    BGain    SkinGain
    51           18    31           32    31           31

ShootSupAdj    DetailCtrlThr    MaxSharpGain    SkinUmax    SkinUmin    SkinVmax
SkinVmin    WeakDetailGain
9           160           40           128           100           150           135    6

-----EDGEMARK INFO-----
    bEnable    Threshold    Color
        0           100           0

-----LDCI INFO-----
    Enable    Manual    GaussLPFSigma    HePosWgt    HePosSigma    HePosMean
HeNegWgt    HeNegSigma    HeNegMean    BlcCtrl
    1           0           20           40           72           0
25           80           0           25

-----PreGamma INFO-----
    Enable
        0

-----PreLogLUT INFO-----
    Enable
        0

-----LogLUT INFO-----
    Enable
        0

-----HLC INFO-----
    Enable    LumaThr    LumaTarget
        0           248           10

-----DetailEnhance INFO-----

```



Enable	enOpType	GlobalGain	GainHF	GainLF
0	0	150	9	9
----- ISP PROC END[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
MODULE/CONTROL PARAM	ProcParam	表示 ISP Proc 信息更新的次数。
	StatIntvl	表示统计信息更新频率。
	Updatepos	表示中断更新位置是用帧起始中断还是在帧结束中断。
	Intbothalf	表示统计信息读取是否在中断执行。
	IntTimeout	表示获取 ISP 中断超时的最大时间
	PwmNum	表示 pwm 使用情况。
	PortIntDelay	表示 Port 中断延时时间。
	QuickStart	表示 ISP 是否采用快速启动
	LdciTprFltEn	表示 LDCI 是否使能时域滤波
	BeBufNum	表示离线模式下，BE config buffer 的数目。仅离线模式有效。
ISP Mode	StitchMode	ISP 拼接模式。
	RunningMode	ISP 运行模式。
	BlockNum	ISP BE 的分块数目。
Sensor Info	SensorType	Sensor 与 ISP 使用的接口通信类型
	Dev	Sensor 绑定的 I2C/SPI 设备号，其值为-1 时，表示当前 ViPipe 的 AE 不配置 sesnor
DRV INFO	ViPipe	ViPipe 号。



参数		描述
	IntCnt	ISP 通道中断数。
	IntT	ISP 中断占用时间。
	MaxIntT	ISP 中断的最大占用时间。
	IntGapT	ISP 相邻两中断之间的时间间隔。
	MaxGapT	ISP 相邻两中断之间的最大时间间隔。
	IntRat	每秒 ISP 中断个数。
	IspResetCnt	ISP 复位的次数。
	IspBeStaLost	BE 统计信息不及时次数。
	PtIntCnt	Port 通道的中断数。
	PtIntT	Port 中断占用时间。
	PtMaxIntT	Port 中断的最大占用时间。
	PtIntGapT	Port 相邻两中断之间的时间间隔。
	PtMaxGapT	Port 相邻两中断之间的最大时间间隔。
	PtIntRat	每秒 Port 中断个数。
	SensorCfgT	Sensor 配置时间。
	SensorMaxT	Sensor 配置的最大时间
BeCfg Phy Addr	BeCfg	离线模式下, ISP BE 寄存器配置 buffer 的物理地址, 共 MAX_ISP_BE_BUF_NUM 块配置 buffer。
PubAttr INFO	WndX	水平方向起始位置。
	WndY	垂直方向起始位置。
	WndW	图像宽度。
	WndH	图像高度。
	SnsW	Sensor 图像宽度。



参数		描述
	SnsH	Sensor 图像高度。
	Bayer	输入 Bayer 图像数据格式。
SNAPATTR INFO	SnapType	拍照类型。
	PipeMode	拍照时通道工作模式。
	OPType	专业模式类型：自动或手动
	ProFrmNum	专业拍照张数。
DPC INFO	Enable	DPC 动态坏点校正使能
	Strength	DPC 动态坏点处理强度
	BlendRatio	坏点校正过程所需的融合比率
CrossTalk INFO	Enable	使能 Crosstalk 功能
	NpOffset	设置 Noise Profile 值。值越大，表示抗噪能力越大
	Threshold	设置 Crosstalk 门限值。值越大，表示整体处理的强度越大
	Strength	设置 Crosstalk 强度值。值越大，处理强度越大
FrameWDR INFO	MdtEn	运动检测使能。
	LongThr	长曝光门限值，低于门限值的图像数据只选择长曝光数据。
	ShortThr	短曝光门限值，超过门限值的图像数据只选择短曝光数据。
	MdThrLowGain	手动模式下判断是否运动的低阈值系数。值越大，运动程度越小。
	MdThrHighGain	手动模式下判断是否运动的高阈值系数。值越大，运动程度越小。
Black Level Actual INFO	BlcR	实际生效的 R 通道的黑电平值
	BlcGr	实际生效的 Gr 通道的黑电平值



参数		描述
	BlcGb	实际生效的 Gb 通道的黑电平值
	BlcB	实际生效的 B 通道的黑电平值
AE INFO	Again	Sensor 模拟增益, 10bit 精度。
	Dgain	Sensor 数字增益, 10bit 精度。
	AgainSF	Sensord 短帧模拟增益, 10bit 精度。仅在 WDR 模式支持。
	DgainSF	Sensor 短帧数字增益, 10bit 精度。仅在 WDR 模式支持。
	IspDg	ISP 数字增益, 10bit 精度。
	SysGain	系统增益, 10bit 精度。
	Iso	ISO 值,表示系统增益, 以常数 100 乘以倍数为单位,例如系统中 sensor 的增益为 2 倍, ISP 的增益为 1 倍, 那么整个系统的 ISO 值计算方式为: $2*1*100=200$, 即系统 ISO 为 200, 本文档中涉及到的 ISO 都是采用这种计算方法。
	IsoSF	短帧 ISO 值, 表示短帧的系统增益。仅在 WDR 模式支持。
	Line	Sensor 曝光时间, 以行为单位。
	LineSS	Sensor 短帧曝光时间, 以行为单位, 仅在 WDR 模式支持。
	LineS	Sensor 次短帧曝光时间, 以行为单位, 仅在 WDR 模式支持。
	LineL	Sensor 次长帧曝光时间, 以行为单位, 仅在 WDR 模式支持。
	LineLL	Sensor 长帧曝光时间, 以行为单位, 仅在 WDR 模式支持。
	AEInter	AE 算法的运行间隔, 2 表示每 2 帧运行一次



参数		描述
		AE。
	Incrmnt	当前曝光量相比上帧曝光量增加的比例(8bit 精度)。
	Exp	当前曝光量，为快门、增益、光圈值的乘积。
	1stTime	AE 第一次稳定的时间，以 us 为单位。
	Comp	自动曝光调整时对曝光补偿量。
	EVbias	自动曝光调整时的曝光偏差值。
	OriAve	根据 256 段直方图计算出来的画面平均亮度。
	Offset	AE 根据统计信息对平均亮度的偏移。
	Speed	自动曝光调整时的初始步长。
	Tole	自动曝光调整时对曝光量的容忍偏差。
	Error	自动曝光调整时平均亮度的偏差。
	Fps	基准帧率。
	RealFps	实际帧率。
	BDelay	图像亮度低于目标亮度时，AE 调节的等待时间 (单位：帧)
	WDelay	图像亮度高于目标亮度时，AE 调节的等待时间 (单位：帧)
	MaxLine	Sensor 支持的最大曝光时间。
	MaxLineT	用户配置的最大曝光时间。
	MaxAgT	用户配置的最大 Sensor 模拟增益。
	MaxDgT	用户配置的最大 Sensor 数字增益。
	MaxIDgT	用户配置的最大 Isp 数字增益。
	MaxSgT	用户配置的最大系统增益。
	ManuEn	手动曝光开关。



参数		描述
	MaLine	手动曝光时间开关。
	MaAg	手动曝光的 Sensor 模拟增益开关。
	MaDg	手动曝光的 Sensor 数字增益开关。
	MalspDg	手动曝光的 Isp 数字增益开关。
	WdrMode	WDR 模式。
	ExpRatio	FSWDR 模式相邻 2 帧曝光比。
	AnFlick	自动抗闪开关。
	SlowMod	自动降帧模式。
	GainTh	进入自动降帧时(SLOW_SHUTTER mode)的系统增益。
	NodeId	节点 Id 号。
	IntTime	节点曝光时间，以行为单位。
	SysGain	节点增益，包括 Sensor 模拟、数字与 ISP 增益。
	IrisApe	节点光圈大小，仅支持 P-Iris 模式。
	UpStgy	AE 上行路线的分配策略。
	DwStgy	AE 下行路线的分配策略。
	Mltply	分配节点快门、增益、光圈值的乘积。
	AuIrEn	自动光圈使能开关。
	IrType	光圈类型。
	MaIrEn	手动光圈使能开关。
	DbgIrSt	光圈调试状态，0 为 Keep，1 为 Open，2 为 Close。
BAYERNR INFO	Enable	NR 使能的开关。
	NrLscEnable	NR 参考 LSC 使能开关。



参数		描述
	NrLscRatio	NR 参考 LSC 增益比例。
	CoarseStr0	控制亮度噪声整体去噪强度，R 通道。
	CoarseStr1	控制亮度噪声整体去噪强度，Gr 通道。
	CoarseStr2	控制亮度噪声整体去噪强度，Gb 通道。
	CoarseStr3	控制亮度噪声整体去噪强度，B 通道。
DRC INFO	En	DRC 使能开关
	ManuEn	DRC 手动使能开关
	Strength	DRC 强度值
LCAC INFO	Enable	Lcac 使能
	CrStr	R 通道的校正强度
	CbStr	B 通道的校正强度
DEMOSAIC INFO	Enable	Demosaic 使能的开关。
	NonDirStr	无方向插值强度，值越大，无方向插值所占比重越大
	NonDirMfStr	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强
	NonDirHfStr	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升
	DeSmthRange	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节
	ColorNoiseThdF	根据画面高频程度降低颜色噪声，值越大，则越容易在平坦区域进行去色噪功能。值越小，则影响的像素越少
	ColorNoiseStrF	根据画面高频程度降低颜色噪声的强度，值越大，则降饱和度强度越高
	ColorNoiseThdY	根据亮度和饱和度降低颜色噪声，值越大，对暗



参数		描述
		处影响越大，对饱和度高的像素影响越大。值越小，影响的像素越少
	ColorNoiseStrY	根据亮度和饱和度降低颜色噪声的强度，值越大，对影响的像素降饱和度越多，值越小，降饱和度程度越少
Anti False Color INFO	Enable	ANTIFALSECOLOR 模块使能。
	Threshold	去伪彩阈值，值越大，伪彩去除的区域越多
	Strength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变
CA INFO	Enable	CA 模块使能
	IsoRatio	当前 ISO 下实际生效的 UV 增益
FPN CORRECT INFO	En	FPN 使能开关
	OpType	FPN 模式类型
	Strength	FPN 强度值
	Offset	FPN 强度偏移
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。



参数		描述
	Color12	色彩还原矩阵 GB 值。
	Color20	色彩还原矩阵 BR 值。
	Color21	色彩还原矩阵 BG 值。
	Color22	色彩还原矩阵 BB 值。
	ManuEn	手动白平衡开关。
	Sat	饱和度值。
	Zones	自动白平衡算法选择。
	Speed	自动白平衡收敛速度。
SHARPEN INFO	bSharpenEn	Sharpen 使能
	LumaWgt	亮度锐化权重。
	TextureStr	无方向的纹理锐化强度。
	EdgeStr	边缘锐化强度。
	TextureFreq	纹理的细碎度调节强度。
	EdgeFreq	边缘锐化后的纤薄度调节。
	OverShoot	锐化后的白边白点强度。
	UnderShoot	锐化后的黑边黑点强度。
	ShootSupStr	黑白边宽窄的单独调节强度。
	DetailCtrl	图像的细节纹理区的 shoot 强度的控制。
	EdgeFiltStr	边缘滤波强度的控制参数。
	EdgeFiltMaxCap	边缘滤波强度范围的控制参数。
	RGain	深红色锐化增益控制。
	Ggain	绿色区域的锐化增益控制。
	BGain	深蓝色锐化增益控制。
	SkinGain	肤色锐化增益控制。



参数		描述
	ShootSupAdj	图像锐化后的 shoot 抑制调节。
	DetailCtrlThr	细节纹理区的 shoot 控制阈值。
	MaxSharpGain	图像锐化的最大增益的限制值。
	SkinUmax	肤色矩形窗的右上最大坐标的 U 值。
	SkinUmin	肤色矩形窗的左下最小坐标的 U 值。
	SkinVmax	肤色矩形窗的右上最大坐标的 V 值。
	SkinVmin	肤色矩形窗的左下最小坐标的 V 值。
Edge Mark INFO	Enable	EdgeMark 边缘标记功能的使能。
	Threshold	边缘标记功能的阈值。
	Color	边缘标记功能颜色的设置。
LDCI INFO	Enable	LDCI 功能使能。
	Manual	LDCI 工作类型。 0: 自动模式; 1: 手动模式。
	GaussLPFSigma	LDCI 滤波局域程度, 其值越小说明 LDCI 作用越局域。
	HePosWgt	局域直方图均衡结果与原图融合权重曲线, 根据亮度控制提亮程度, 曲线呈 GAUSS 分布, 该参数为 GAUSS 分布最大值。
	HePosSigma	局域直方图均衡结果与原图融合权重曲线, 根据亮度控制提亮程度, 曲线呈 GAUSS 分布, 该参数为 GAUSS 分布均方差。
	HePosMean	局域直方图均衡结果与原图融合权重曲线, 根据亮度控制提亮程度, 曲线呈 GAUSS 分布, 该参数为 GAUSS 分布数学期望。
	HeNegWgt	局域直方图均衡结果与原图融合权重曲线, 根据亮度控制变暗程度, 曲线呈 GAUSS 分布, 该参



参数		描述
		数为 GAUSS 分布最大值。
	HeNegSigma	局域直方图均衡结果与原图融合权重曲线，根据亮度控制变暗程度，曲线呈 GAUSS 分布，该参数为 GAUSS 分布均方差。
	HeNegMean	局域直方图均衡结果与原图融合权重曲线，根据亮度控制变暗程度，曲线呈 GAUSS 分布，该参数为 GAUSS 分布数学期望。
	BlcCtrl	暗区增益控制强度，用于抑制暗区噪声。
PreGamma INFO	Enable	PreGamma 使能开关
PreLogLUT INFO	Enable	PreLogLUT 使能开关
LogLUT INFO	Enable	LogLUT 使能开关
HLC	Enable	HLC 高光遮蔽功能的使能。
	LumaThr	高光遮蔽的阈值。
	LumaTarget	高光遮蔽拉灰的目标值。
DE INFO	Enable	DetailEnhance 使能开关
	enOpType	DetailEnhance 手动模式使能开关
	GlobalGain	DetailEnhance 全局叠加增益
	GainHF	DetailEnhance 高频信息叠加增益
	GainLF	DetailEnhance 低频信息叠加增益



说明

宏定义如下：

```
#define MAX_ISP_BE_BUF_NUM
```

4

【注意事项】



用户通过 PQTools 手动配置 PubAttr 修改画面帧率时，sensor 的最大曝光时间会发生修改。绝大部分修改帧率时，帧率越高则最大曝光时间越低。



11

3A 开发指南概述

11.1 概述

ISP 3A 版本依赖于相应的 SDK 大版本，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。主要包含 Firmware 框架及上海海思 3A 库，Firmware 提供算法的基本框架，处理统计信息，驱动数字图像处理算法，并包含坏点校正、去噪、色彩增强、镜头阴影校正等处理。3A 库以注册的方式，添加到 Firmware 中，完成曝光、白平衡、色彩还原等处理。

11.2 功能描述

ISP 的 Firmware 架构及 Firmware 工作流程及文件组织结构请参考本文档 [1.2 “功能描述”](#) 小节，本小节不再赘述。

12 3A 开发使用者指南

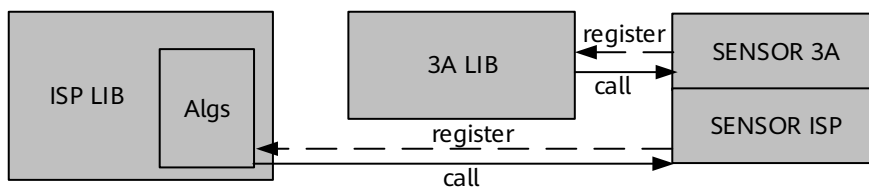
12.1 软件流程

ISP 软件流程请参考第一章 1.2.4 “软件流程” 小节，本小节不再赘述。

12.2 Sensor 对接

Sensor 库主要是为了提供差异化的 sensor 适配，里面的内容可以分为两部分：Sensor 向 ISP 库注册的差异化适配，这些差异化适配主要由 Firmware 中的基础算法单元决定；Sensor 向上海海思 3A 库注册的差异化适配。Sensor 的适配包括算法的初始化默认值，及 sensor 控制接口，sensor 的适配是通过接口回调的形式注册给 ISP 库和 3A 算法库。图 12-1 表示了 Sensor 与 ISP 库和 3A 算法库之间的关系。

图12-1 Sensor 适配示意图



关于 sensor 注册 ISP 库，sensor 注册 3A 算法库部分详细请参考《sensor 调试指南》



13 开发者指南

说明

- 未有特殊说明，Hi3519AV100、Hi3559CV100、Hi3568V100、Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3518EV300 与 Hi3516EV200 内容一致
- 未有特殊说明，Hi3516DV200 与 Hi3516EV300 内容一致
- 未有特殊说明，Hi3516CV500、Hi3516DV300、Hi3516AV300 内容一致
- 未有特殊说明，Hi3562V100、Hi3566V100、Hi3559V200 与 Hi3556V200 内容一致

13.1 概述

用户可以基于 Firmware 框架开发定制 3A 库，并注册到 Firmware 中，Firmware 将会在中断的驱动下，获取每帧的统计信息，运行算法库，配置 ISP 寄存器。

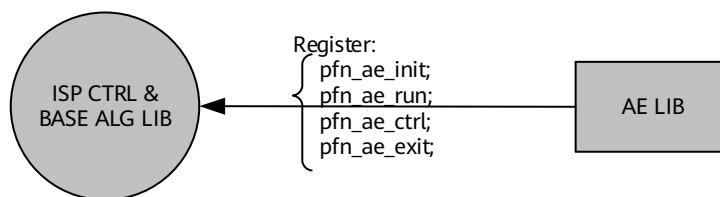
Sensor 注册到 ISP 库，以实现差异化适配 Firmware 中的基础算法单元的部分，仍需要参考使用者指南部分的内容实现，以保证 Firmware 中的坏点校正、去噪、色彩增强、镜头阴影校正等处理算法正常运行。Sensor 注册到 3A 算法库的部分，请用户根据自己开发定制的 3A 库，自行定义数据结构，实现 Sensor 曝光控制等。

如果用户只是使用上海海思 3A 算法库，并不自己开发 3A 算法库，可以忽略此章节内容。

13.2 AE 算法注册 ISP 库

AE 算法注册 ISP 库调用 HI_MPI_ISP_AeLibRegCallBack，如图 13-1 所示：

图13-1 AE 算法向 ISP 库注册的回调函数



上海海思 AE 算法实现了一个 HI_MPI_AE_Register 的注册函数，在这个函数中调用 ISP 提供的 HI_MPI_ISP_AeLibRegCallBack 回调接口，用户调用注册函数以实现向 ISP 注册 AE 算法，示例如下：

【举例】

```
/* 实现注册函数 */
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);

/* 调用钩子函数 */
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");
}
```

用户需要在自开发定制的 AE 库中实现以下回调函数：

表13-1 AE 算法向 ISP 库注册的回调函数

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。



说明

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。

pfn_ae_ctrl 回调函数的目的是改变算法库内部状态。运行时 Firmware 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。

当前 Firmware 定义的 ctrl 命令详参 [ISP_CTRL_CMD_E](#)。

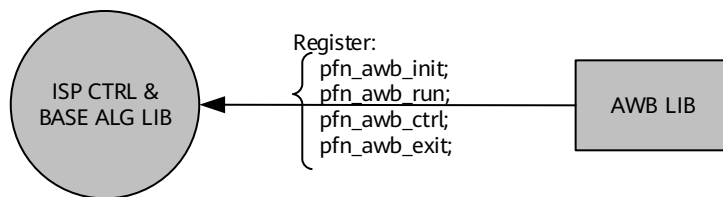
调用 HI_MPI_ISP_Exit 时将调用 pfn_ae_exit 回调函数，以销毁 AE 算法库。

结构体 [ISP_AE_PARAM_S](#)，[ISP_AE_INFO_S](#)，[ISP_AE_RESULT_S](#) 的成员变量请参看 [2.3](#) 小节相应的数据类型。

13.3 AWB 算法注册 ISP 库

AWB 算法注册 ISP 库调用 HI_MPI_ISP_AwbLibRegCallBack，如[图 13-2](#) 所示：

图13-2 AWB 算法向 ISP 库注册的回调函数



上海海思 AWB 算法实现了一个 HI_MPI_AWB_Register 的注册函数，在这个函数中调用 ISP 提供的 HI_MPI_ISP_AwbLibRegCallBack 回调接口，用户调用注册函数以实现向 ISP 注册 AWB 算法，示例和 AE 算法库注册类似。

用户需要在自开发定制的 AWB 库中实现以下回调函数：



表13-2 AWB 算法向 ISP 库注册的回调函数

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。



说明

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。

pfn_awb_ctrl 回调函数的目的是改变算法库内部状态。运行时 Firmware 会隐式调用 pfn_awb_ctrl 回调函数，通知 AWB 算法库切换 WDR 和线性模式、设置 ISO 和曝光时间。设置 ISO 的目的是为了实现 ISO 与饱和度的联动，增益大时色度噪声也会比较大，所以需要调节饱和度。设置曝光时间是为了 awb 算法判断环境照度，优化场景效果。当前 Firmware 定义的 ctrl 命令详参 [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。

结构体 [ISP_AWB_PARAM_S](#)，[ISP_AWB_INFO_S](#)，[ISP_AWB_RESULT_S](#) 的成员变量请参看 2.3 小节相应的数据类型。

13.4 开发用户 AF 算法

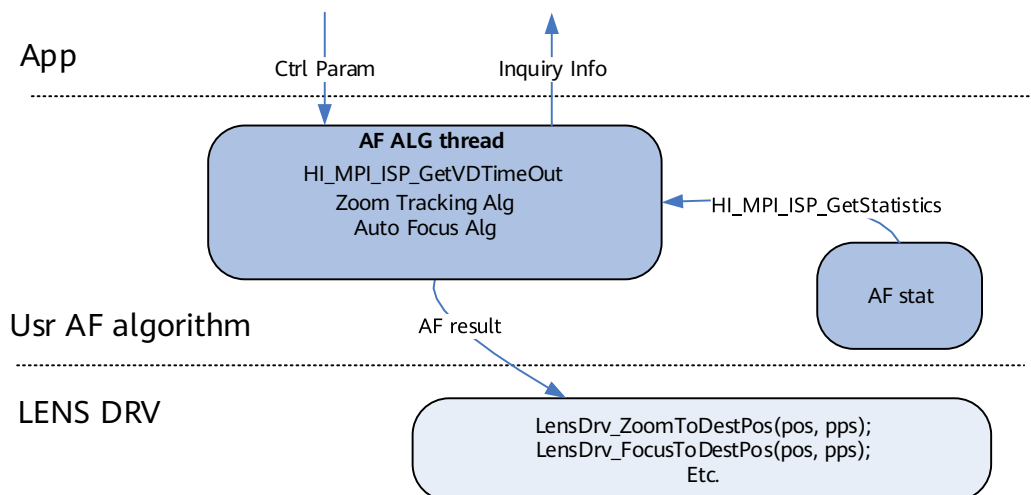


说明

[HI_MPI_ISP_GetVDTimeOut](#) 和 [HI_MPI_ISP_GetStatistics](#) 请参考前面章节。

用户开发的 AF 算法可以参考图 13-3 所示的结构，AF 算法在一个 thread 内运行，使用 [HI_MPI_ISP_GetVDTimeOut](#) 接口将算法同步在 VD 下运行。通过调用 [HI_MPI_ISP_GetStatistics](#) 来获取 AF 相关的统计信息，相关算法参考统计信息完成目标 zoom 和 focus 位置的计算，AF 算法输出 result，包括 lens 的位置信息和速度。调用 Lens Driver 将镜头镜片驱动到设定的位置即完成当前帧的操作。AF Alg 向上提供控制接口和查询接口，方便用户完成一键聚焦，变焦，手动聚焦，查询算法状态等操作。

图13-3 AF 算法结构图



【举例】

如果用户使用的是 Linux 系统，推荐将算法放在 user space 完成，lens driver 放在 kernel space，ISP 驱动提供同步回调功能，参考 13.5.12 “[AF 同步回调](#)” 章节。

```

while (1)
{
    s32Ret = HI_MPI_ISP_GetVDTimeOut(ViPipe, ISP_VD_FE_START, 5000);
    s32Ret |= HI_MPI_ISP_GetFocusStatistics(ViPipe, &stIspsStatics);

    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_ISP_GetFocusStatistics error!(s32Ret = 0x%x)\n", s32Ret);
        return HI_FAILURE;
    }

    // User Auto Focus Alg Source
    UsrAfAlgRun();

    // Update Lens Position
    LensDrv_ZoomToDestPos(pos, pps);
    LensDrv_FocusToDestPos(pos, pps);
}

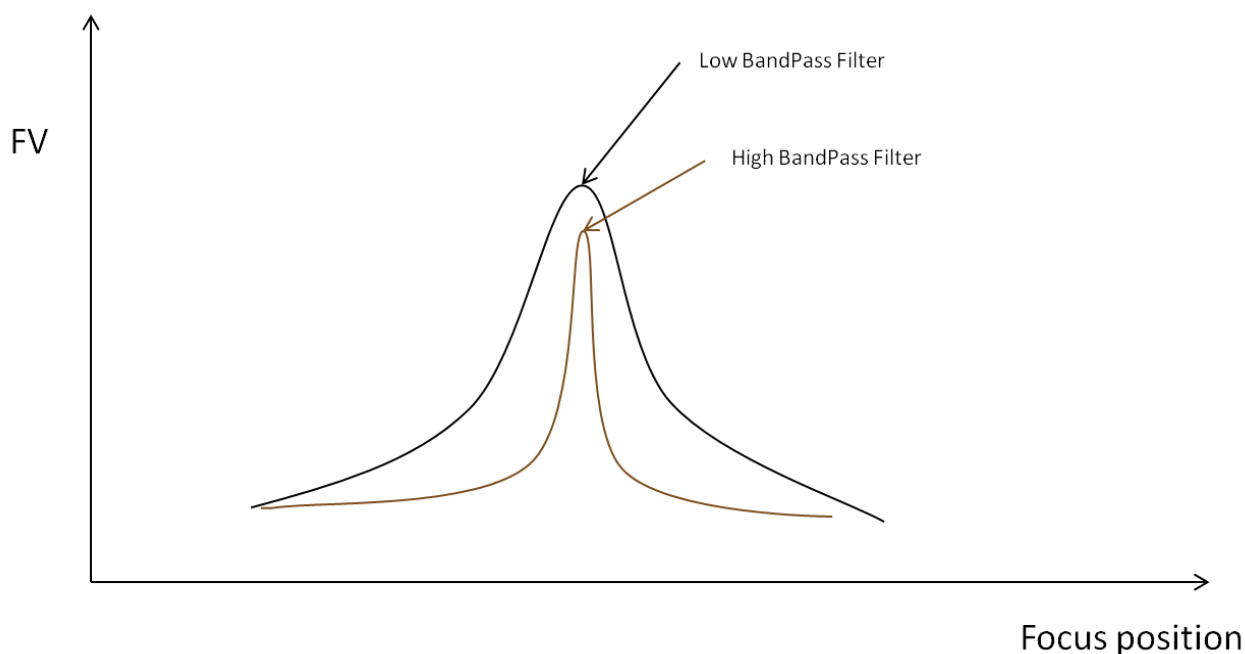
```

13.5 AF 统计信息使用说明

13.5.1 概述

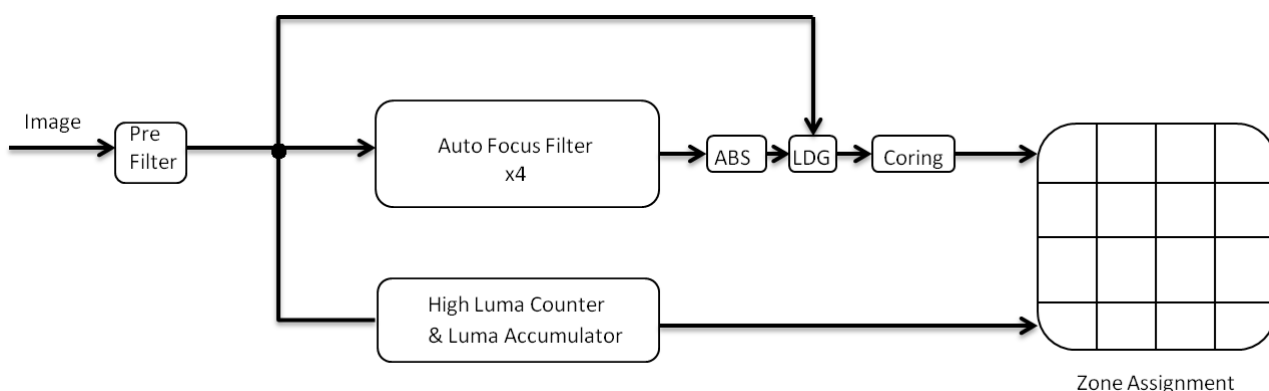
被动式自动对焦一般是通过分析图像特征得出图像清晰度值 FV(Focus Value)，通过驱动对焦马达调节焦点到最佳位置，如图 13-4 所示，图像越清晰 FV 值越大，Peak 点对应聚焦清晰点。获取图像清晰度算法有多种，如灰度梯度法，高频分量法等，Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 采用高频分量法来计算 FV，即图像越清晰的时候高频部分幅值越大，将图像通过高通滤波器便可以得到高频分量，Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 一共提供四个滤波器和亮度信息，分别为水平方向滤波 H1，H2，垂直方向滤波 V1，V2，以及 Y 和高亮计数器 HLCnt。Hi3559AV100/Hi3519AV100 提供 FE (Front End) 及 BE (Behind End) 两个独立的 AF 统计信息获取模块，Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 只提供 BE (Behind End) 的 AF 统计信息获取模块。Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 AF 模块支持的最小分辨率为 256*120，低于该分辨率 AF 逻辑功能无法支持。

图13-4 FV 曲线



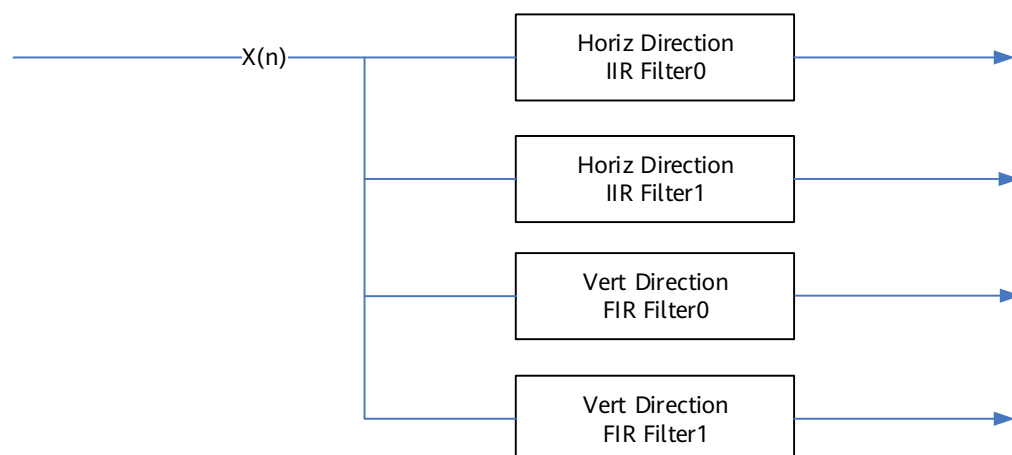
Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 提供 Bayer 域或者 YUV 域进行统计，配置 MPI 中的 enStatisticsPos 字段即可以进行统计位置的调整。WDR 模式下，非宽动态场景的 Bayer 域（即 RAW 域）统计信息无明显变化趋势，推荐使用 YUV 域统计信息。首先对图像进行预处理滤波，去除椒盐噪声对统计值的干扰，用户可以设置 stPreFltCfg.bEn 来使能 PreFilter。配置 stPreFltCfg.u16strength 来控制滤波强度。在经过水平垂直四组滤波器后会输出做 Level Depend Gain 和 Coring，关于这两个模块将在后面的章节详述。模块框图 13-5 所示。

图13-5 AF 统计模块框图



Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516EV200/Hi3516EV300 水平垂直滤波器分别采用 IIR 和 FIR 实现如图 13-6 所示。

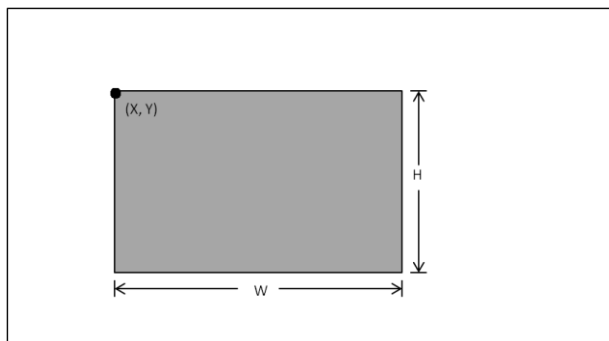
图13-6 AF 4 组滤波器



13.5.2 输入图像的裁剪

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 AF 模块支持对输入图像的裁剪，如下如图 13-7 用户可以根据实际需求配置 stCrop 中 X, Y, W, H 四个参数对指定区域进行统计。

图13-7 AF 输入图像裁剪



13.5.3 Bayer 域配置

如果用户将统计位置调整到 Bayer 域则需要配置以下几点。

1. 配置 enPattern 来指定 bayer 数据的 pattern。
Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 支持如图 13-8 所示的 4 种 pattern。
2. 配置 AF 模块 Gamma 使能，raw 上的线性数据需要做 Gamma 处理。

图13-8 Bayer 域统计预处理

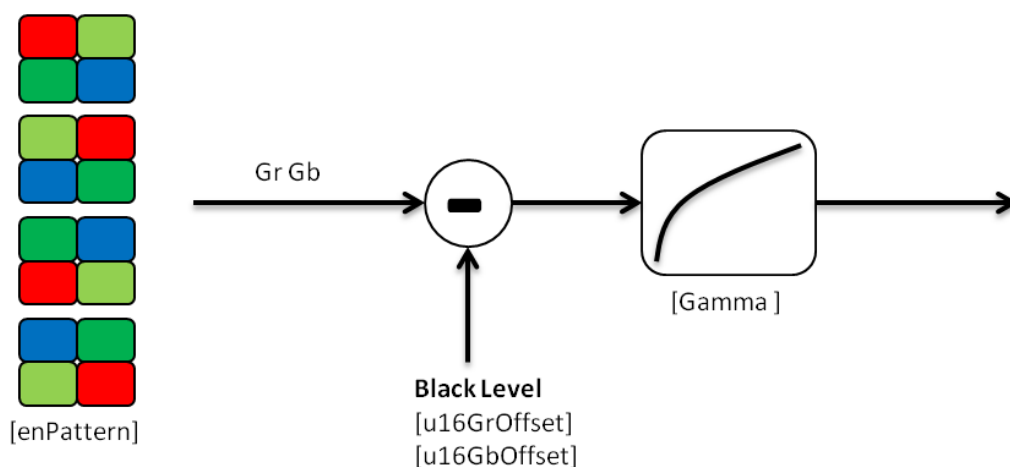
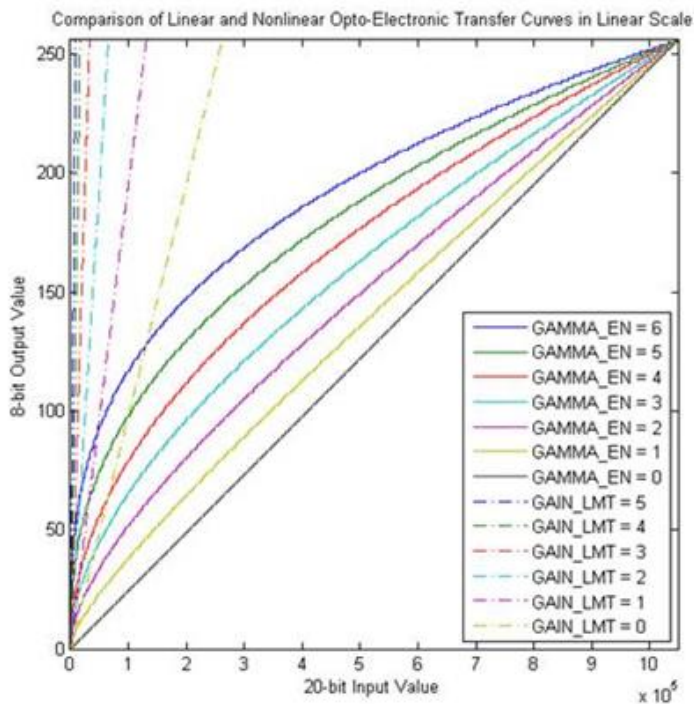


图13-9 关于 Gamma 配置の説明



须知

图中的 GAMMA_EN 与 ISP_AF_RAW_CFG_S 中的 GammaValue 等效，GAIN_LMT 与 ISP_AF_RAW_CFG_S 中的 GammaGainLimit 等效。

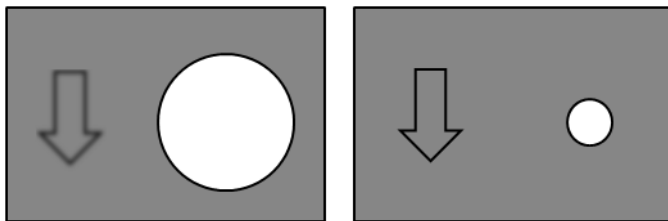
13.5.4 抑制光源对 FV 值的影响

因为 FV 值容易受到 SpotLight 的影响，在聚焦模糊的时候因为光晕扩散，图像中低频分量会增加，会出现图像模糊反而 FV 值变大的现象。为了抑制这一现象，Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 提供以下方法。

- 高亮计数器

如图 13-10，在聚焦模糊的时候因为光晕扩大画面中高亮点的个数增加，清晰的时候高亮点的个数最小，通过配置合适的高亮点判断门限 u16HighLumaTh 即可以辅助实现 SpotLight 场景聚焦。

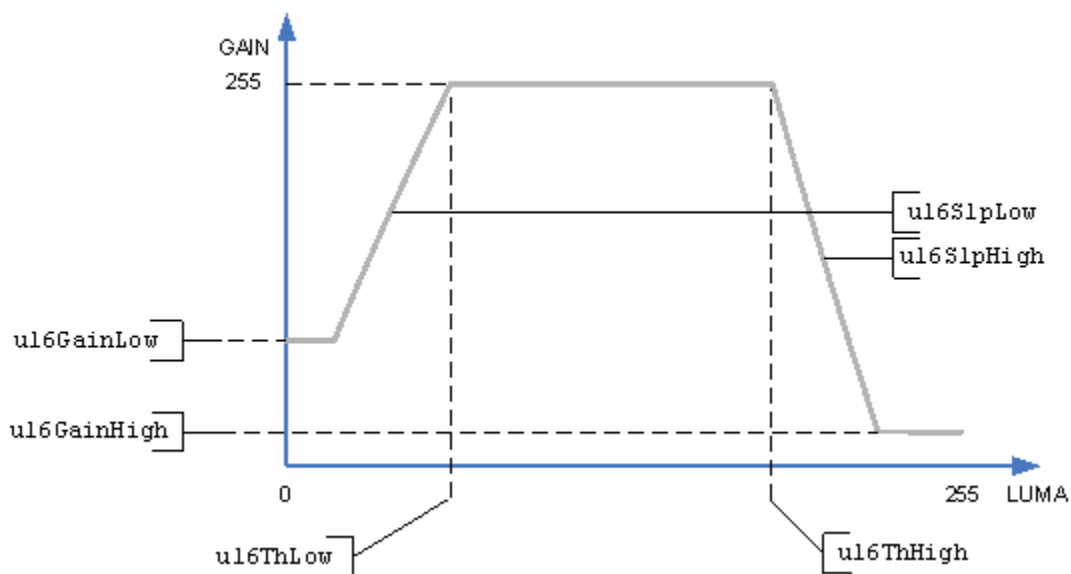
图13-10 高亮统计值



- Level Depend Gain

LDG 的主要原理为通过参考画面像素亮度来衰减滤波器的输出，从而抑制 SpotLight 对 FV 值的影响。如图 13-11 横坐标为像素亮度值，纵坐标为对滤波器输出的 Gain，此处的 Gain 为 8bit 定点小数 Q8。用户可以通过调整图中 6 个参数控制曲线的形状。

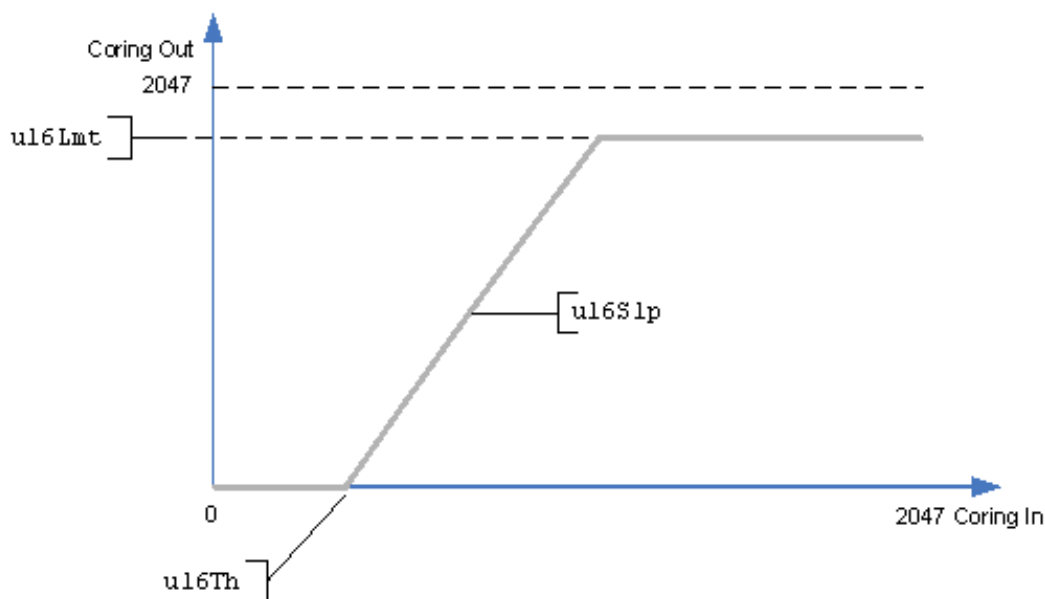
图13-11 Level Depend Gain 曲线



13.5.5 Coring

将滤波器输出通过 Coring 模块可以提高 FV 曲线的抗噪声能力，如图 13-12 所示，设置图中 3 个参数便可以确定 coring 曲线的形状，通过设置合理的参数便可以实现不统计噪声，以及增加有效信号对统计信息的贡献，从而提升统计信息在低照度场景中的性能。

图13-12 Coring 曲线



13.5.6 统计模式配置

统计模块支持以下两种模式：

- 峰值配置模式

设置为 peak 模式 block 的统计值为滤波后图像每行最大值之和，否则直接将每个点的值求和。求和模式适用于噪声较大场景。

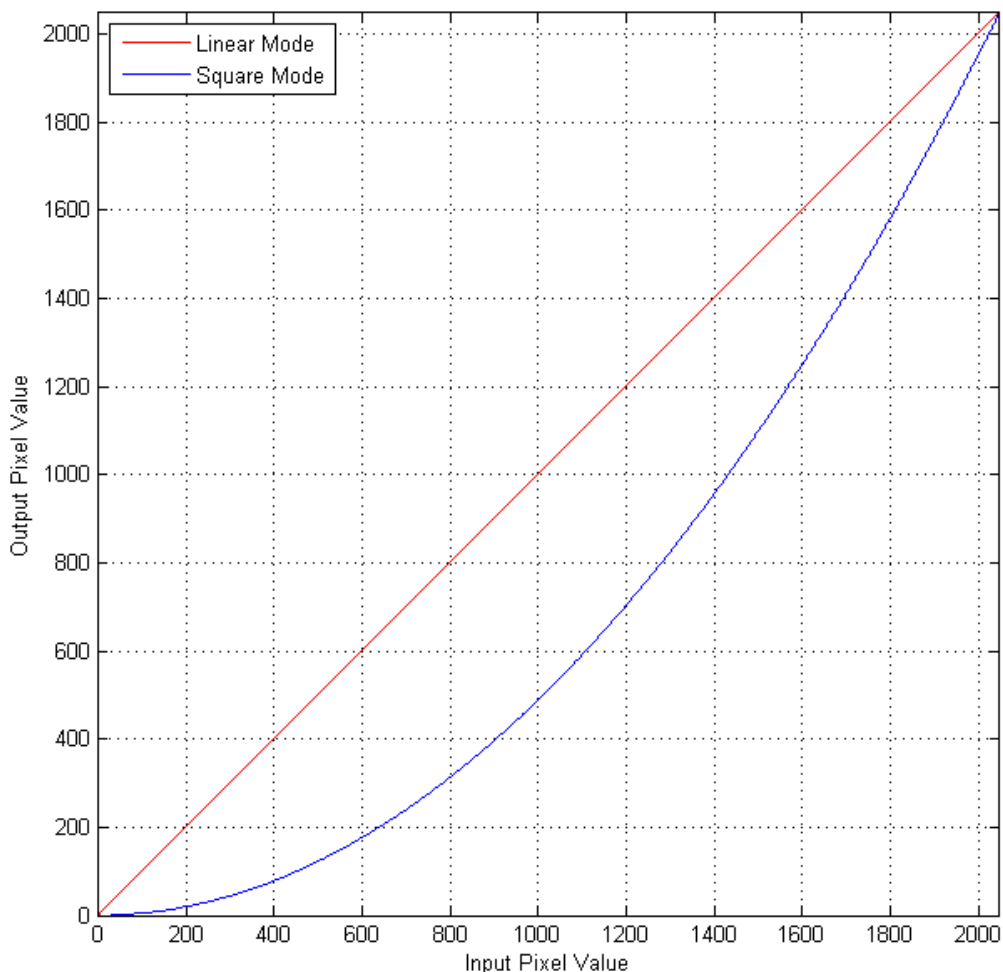
- 平方配置模式，有以下两种选择。

- 0: Linear mode
- 1: Square mode

设置为 Square 模式后，会先对滤波器输出进行归一化后进行平方再做统计。

Square 模式的 FV 曲线在焦点附近会更加陡峭，相应的映射曲线如图 13-13 所示。

图13-13 平方配置模式映射曲线图



同时也支持以下配置：

- 输出 shift

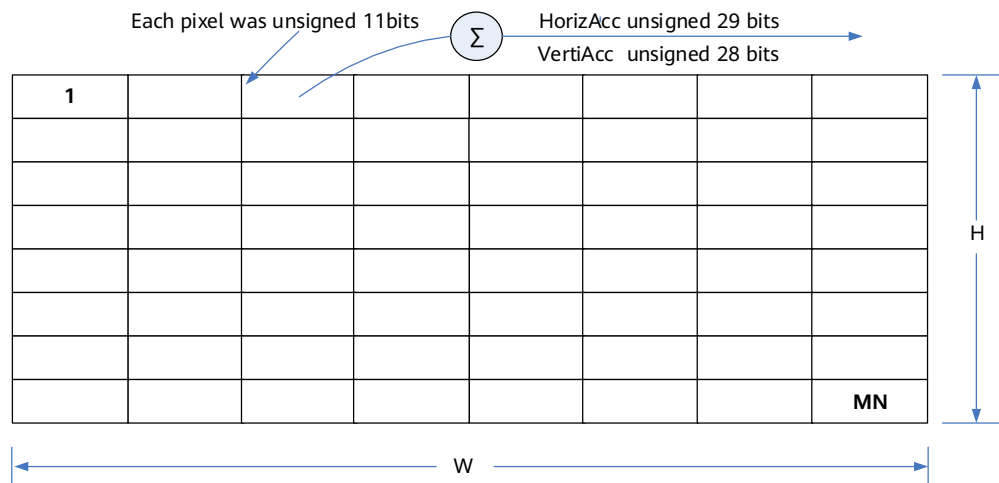
对输出值 right shift 缩小，防止值溢出。

- Block 大小

对通过滤波器后的图像可以进行分块统计，如图 13-14 所示，块的大小可以设置，

Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3559V200/Hi3516EV200/Hi3516EV300 最大支持 17*15 个 blocks，最终块的统计值是对每个 pixel 的累加或者对每行最大值累加。以水平方向为例，因为累加器的宽度为 unsigned 29bits，滤波器输出 pixel 宽度为 unsigned 11bits，所以当块内输出值都为 2047 的时候，最大支持累加的 pixel 个数为 2^{18} ，否则会发生溢出，用户设置大小时应该注意。

图13-14 滤波器后的图像



13.5.7 统计信息配置注意事项

表13-3 统计信息配置注意事项

类型	描述
分块大小	最大 17 乘以 15
统计模块工作域	RAW/YUV
Bayer 统计模块位置	Raw 在 DG 后、DRC 后
YUV 统计模块位置	CSC 后
Bayer 统计参数是否减去黑电平	已减去
统计输出 YUV 是否减去黑电平	已减去
是否支持 IIR 滤波器 filterShift 补偿	支持
是否支持 IIR 滤波器 narrowband	支持
WDR 模式下特殊配置	YUV 输入时 GammaValue (参考 ISP_AF_RAW_CFG_S) 设置为 0。 RAW 需要配置在 DRC 之后



13.5.8 统计值的获取

当一帧图像最后一个 pixel 通过 AF 模块后, 统计值即更新, 推荐用户通过 HI_MPI_ISP_GetVDTimeOut 同步获取统计值, 然后 AutoFocus 和 ZoomTracking 算法完成目标 focus 和 zoom position 的计算, 需要注意的是因为 LINUX user space 任务调度不能保证一致的实时性, 建议将需要保证实时性的驱动配置放在 kernel space 完成。

13.5.9 FV 值计算

一般的推荐最终生成如图 13-4 所示的两条曲线, 我们将 low BandPass filter 生成的 FV 命名为 FV1, High BandPass filter 生成的命名为 FV2, 13.5.10 参考代码中变量 stFocusCfg 的值为默认推荐使用的 AF 滤波器参数, FV1 使用在低照度场合, FV2 使用在正常照度场合。对于每一个 block, FV1, FV2 的计算方式为:

$$\begin{cases} FV1_n = \alpha * H1_n + (1 - \alpha) * V1_n \\ FV2_n = \beta * H2_n + (1 - \beta) * V2_n \end{cases}$$

H1, H2, V1, V2 分别为水平垂直四组滤波器的统计值, 可设置适当的权重对水平和垂直方向的滤波器输出统计值进行混合。

最终的 FV 值还需要对每个 block 进行加权, 即 FV1, FV2 采用如下公式计算。

$$FV = \frac{\sum_{n=1}^{BLOCKS} (FV_n * Weight_n)}{\sum Weight_n}$$

13.5.10 FV 计算参考代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "hi_type.h"
#include "mpi_isp.h"

#define BLEND_SHIFT 6
#define ALPHA      64      // 1
#define BELTA      54      // 0.85
#define FEINFO
```



```
//static int AFWeight[8][8] = {{1,1,1,1,1,1,1,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,2,2,2,2,2,2,1},
//                               {1,1,1,1,1,1,1,1},
//                               };

static int AFWeight[15][17] = {{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,2,2,2,2,2,2,2,2,2,2,2,2,2,1},
                               {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
                               };

int main(int argc, char* argv[])
{
    HI_S32 s32Ret = HI_SUCCESS;
    HI_U8 u8WdrChn;
    HI_U32 u32FrmCnt = 0;
    HI_U32 i, j, k;
    HI_U16 u16StatData;
    VI_PIPE ViPipe;
    ISP_AF_STATISTICS_S stIsPStatics;
```



```
ISP_STATISTICS_CFG_S stIsStaticCfg;
ISP_PUB_ATTR_S stPubattr;
//ISP_PUB_ATTR_S stPubattr;
ISP_FOCUS_ZONE_S stZoneMetrics[WDR_CHN_MAX][AF_ZONE_ROW][AF_ZONE_COLUMN]
= {0};

ISP_FOCUS_STATISTICS_CFG_S stFocusCfg =
{
    {1, 17, 15, 4096, 2160, 1, 0, {0, 0, 0, 0, 0}, 0, {0x2, 0x4, 0}, {1, 0x9bff, 0xf0},
    {1, {1, 1, 1}, 15, {188, 414, -330, 486, -461, 400, -328}, {7, 0, 3, 1}, {1, 0, 255, 0, 220, 8,
14}, {127, 12, 2047} },
    {0, {1, 1, 0}, 2, {200, 200, -110, 461, -415, 0, 0}, {6, 0, 1, 0}, {0, 0, 0, 0, 0, 0}, {15, 12,
2047} },
    {{20, 16, 0, -16, -20}, {1, 0, 255, 0, 220, 8, 14}, {38, 12, 1800} },
    {{-12, -24, 0, 24, 12}, {1, 0, 255, 0, 220, 8, 14}, {15, 12, 2047} },
    {4, {0, 0}, {1, 1}, 0}
};
char param1, param2;

if (argc < 3)
{
    printf("use like. ./sample_af isp_Pipe c    -> Fv curve\n");
    printf("..... ./sample_af isp_Pipe h 0  -> h0 blocks\n");
    printf("..... ./sample_af isp_Pipe h 1  -> h1 blocks\n");
    printf("..... ./sample_af isp_Pipe v 0  -> v0 blocks\n");
    printf("..... ./sample_af isp_Pipe v 1  -> v1 blocks\n");
    printf("..... ./sample_af isp_Pipe y 0  -> y  blocks\n");
    printf("..... ./sample_af isp_Pipe y 1  -> hlcnt blocks\n");
    return -1;
}

if ((param1 = *argv[2]) != 'c' && !argv[3])
{
    printf("args was too less!, should be %c + X\n", *argv[1]);
    return -1;
}
else if (argv[3])
```




```
{
    param2 = *argv[3];
}

if ('0' != *argv[1] && '1' != *argv[1])
{
    printf("ViPipe num Err!\n");
    return -1;
}
ViPipe = atoi(argv[1]);

s32Ret = HI_MPI_ISP_GetStatisticsConfig(ViPipe, &stIsPStaticsCfg);
s32Ret |= HI_MPI_ISP_GetPubAttr(ViPipe, &stPubattr);

stFocusCfg.stConfig.u16Vsize = stPubattr.stWndRect.u32Height;
stFocusCfg.stConfig.u16Hsize = stPubattr.stWndRect.u32Width;

if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_ISP_GetStatisticsConfig error!(s32Ret = 0x%x)\n", s32Ret);
    return HI_FAILURE;
}
memcpy(&stIsPStaticsCfg.stFocusCfg, &stFocusCfg, sizeof(ISP_FOCUS_STATISTICS_CFG_S));
s32Ret = HI_MPI_ISP_SetStatisticsConfig(ViPipe, &stIsPStaticsCfg);
if (HI_SUCCESS != s32Ret)
{
    printf("HI_MPI_ISP_SetStatisticsConfig error!(s32Ret = 0x%x)\n", s32Ret);
    return HI_FAILURE;
}
#endif
while (1)
{
    s32Ret = HI_MPI_ISP_GetVDTimeOut(ViPipe, ISP_VD_FE_START, 5000);
    s32Ret |= HI_MPI_ISP_GetFocusStatistics(ViPipe, &stIsPStatics);
    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_ISP_GetStatistics error!(s32Ret = 0x%x)\n", s32Ret);
        return HI_FAILURE;
    }
}
```



```
    }

#ifdef FEINFO
    memcpy(stZoneMetrics, &stlspStatics.stFEAFStat, sizeof(ISP_FOCUS_ZONE_S) *
WDR_CHN_MAX * AF_ZONE_ROW * AF_ZONE_COLUMN);
#else
    memcpy(stZoneMetrics[0], &stlspStatics.stBEAFStat, sizeof(ISP_FOCUS_ZONE_S) *
AF_ZONE_ROW * AF_ZONE_COLUMN);
#endif

    if (param1 != 'c')
    {
        if ((++u32FrmCnt % 30))
        {
            continue;
        }

        u8WdrChn = 1;
        for(k = 0; k < u8WdrChn; k++)
        {
            for (i = 0; i < 15; i++)
            {
                for (j = 0; j < 17; j++)
                {
                    if (param1 == 'h' && param2 == '0')
                    {
                        u16StatData = stZoneMetrics[k][i][j].u16h1;
                    }
                    else if (param1 == 'h' && param2 == '1')
                    {
                        u16StatData = stZoneMetrics[k][i][j].u16h2;
                    }
                    else if (param1 == 'v' && param2 == '0')
                    {
                        u16StatData = stZoneMetrics[k][i][j].u16v1;
                    }
                    else if (param1 == 'v' && param2 == '1')
```



```
        {
            u16StatData = stZoneMetrics[k][i][j].u16v2;
        }
        else if (param1 == 'y' && param2 == '0')
        {
            u16StatData = stZoneMetrics[k][i][j].u16y;
        }
        else
        {
            u16StatData = stZoneMetrics[k][i][j].u16HlCnt;
        }
        printf("%6d", u16StatData);
    }
    printf("\n");
}
}
printf("-----%c-%c-----\n\n\n\n", param1, param2);

    continue;
}

HI_U32 u32SumFv1 = 0;
HI_U32 u32SumFv2 = 0;
HI_U32 u32WgtSum = 0;
HI_U32 u32Fv1_n, u32Fv2_n, u32Fv1, u32Fv2;

if ((++u32FrmCnt % 2))
{
    continue;
}
u8WdrChn = 1;
for(k = 0; k < u8WdrChn; k++)
{
    for ( i = 0 ; i < stFocusCfg.stConfig.u16Vwnd; i++ )
    {
        for ( j = 0 ; j < stFocusCfg.stConfig.u16Hwnd; j++ )
        {
```



```
HI_U32 u32H1 = stZoneMetrics[k][i][j].u16h1;
HI_U32 u32H2 = stZoneMetrics[k][i][j].u16h2;
HI_U32 u32V1 = stZoneMetrics[k][i][j].u16v1;
HI_U32 u32V2 = stZoneMetrics[k][i][j].u16v2;

u32Fv1_n = (u32H1 * ALPHA + u32V1 * ((1<<BLEND_SHIFT) - ALPHA)) >>
BLEND_SHIFT;

u32Fv2_n = (u32H2 * BELTA + u32V2 * ((1<<BLEND_SHIFT) - BELTA)) >>
BLEND_SHIFT;

u32SumFv1 += AFWeight[i][j] * u32Fv1_n;
u32SumFv2 += AFWeight[i][j] * u32Fv2_n;
u32WgtSum += AFWeight[i][j];
    }
}
}

u32Fv1 = u32SumFv1 / u32WgtSum;
u32Fv2 = u32SumFv2 / u32WgtSum;

printf("%4d    %4d\n", u32Fv1, u32Fv2);
}
#endif
return HI_SUCCESS;
}
```

13.5.11 PQTools 滤波器设计工具使用说明

HiPQ 工具提供了自动对焦参数仿真插件，使图像质量调试人员可以：

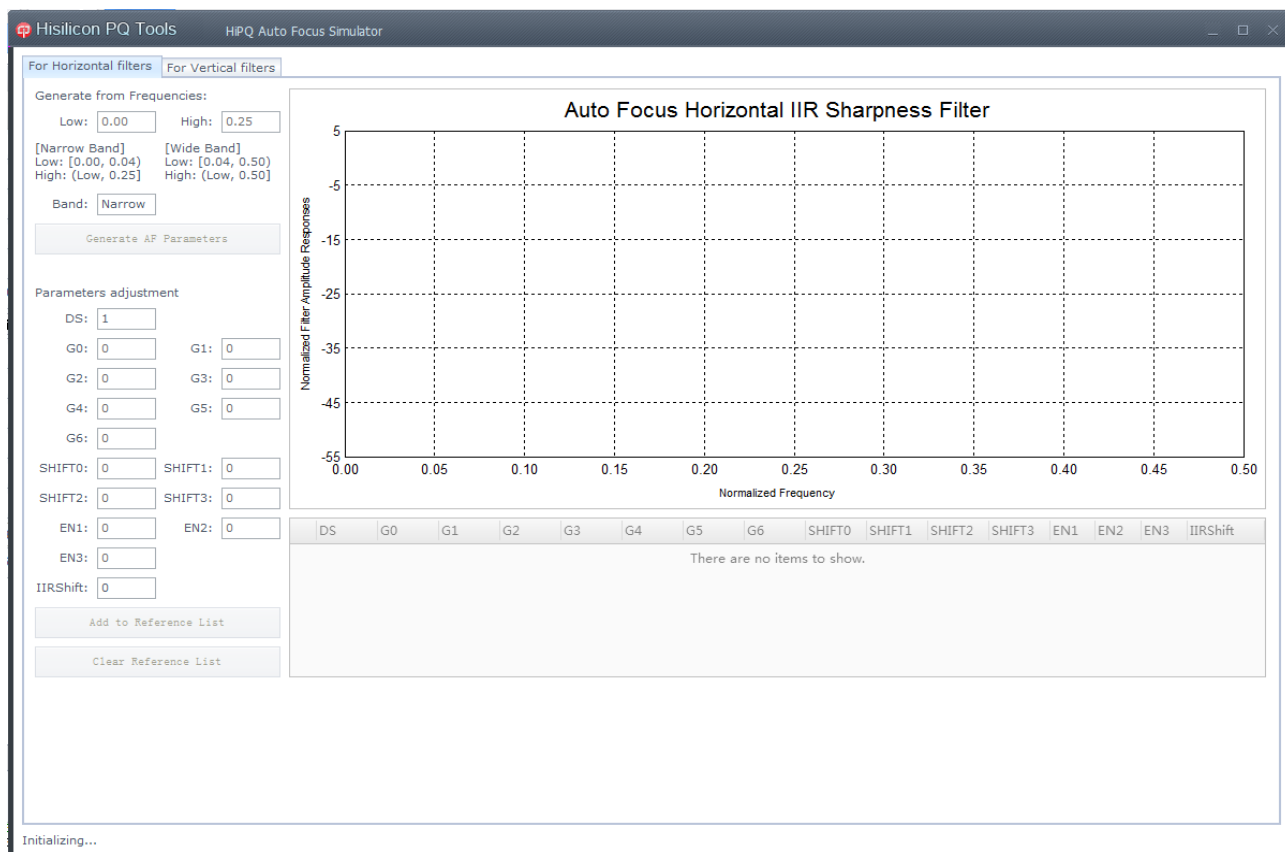
- 根据截止频率自动获取较适合的 AF 参数，并查看对应的频率响应曲线；
- 自行配置 AF 参数，并查看对应的频率响应曲线。

目前版本的插件可用于图像质量调测，使用该工具之前，用户需要安装 Matlab 运行时的 2012a (32 位) 版本。

13.5.11.1 工具界面

从 HiPQ 工具主界面工具栏的外挂插件下拉框中选择“HiPQ Auto Focus Simulator”，可以打开自动对焦参数仿真工具，如图 13-15 所示。

图13-15 自动对焦参数仿真工具



工具分为 For Horizontal filters 和 For Vertical filters 两个页签，相互独立，分别对应水平和垂直方向的滤波器参数仿真。下面叙述的内容对两个页签均有效。

AF 水平滤波器具备以下两种模式：

- 窄带模式 (Narrow Band)：此种模式下，AF 水平滤波器将在一个较小的频率范围内进行对焦。当用户设置的截止频率的低频小于 0.04 时，工具将推荐用户使用窄带模式。这个模式下，截止频率高频最大值为 0.25。
- 宽带模式 (Wide Band)：此种模式下，AF 水平滤波器将在一个较大的频率范围内进行对焦。当截止频率的低频大于或等于 0.04 时，工具将推荐用户使用宽带模式。这个模式下，截止频率高频最大值为 0.5。

用户可以在 PQ 工具主界面（已经加载调试表）上找到 StatisticsConfig -> AFConfig_HPParam_IIR0 的寄存器分组来对模式和滤波系数进行设置。

13.5.11.2 通过截止频率获取 AF 滤波系数

在每个页签中，用户均可以用截止频率来生成滤波系数。在 Generate from Frequencies 处输入截止频率的低频和高频后，点击 “Generate AF Parameters”，即可生成截止频率对应的滤波系数，显示在 Parameters adjustment 下方的各个文本框中，并在右侧的坐标图上显示对应该组滤波系数的频率响应曲线，曲线以红色表示，如图 13-16 和图 13-17 所示。

图13-16 通过截止频率获取滤波系数（窄带）

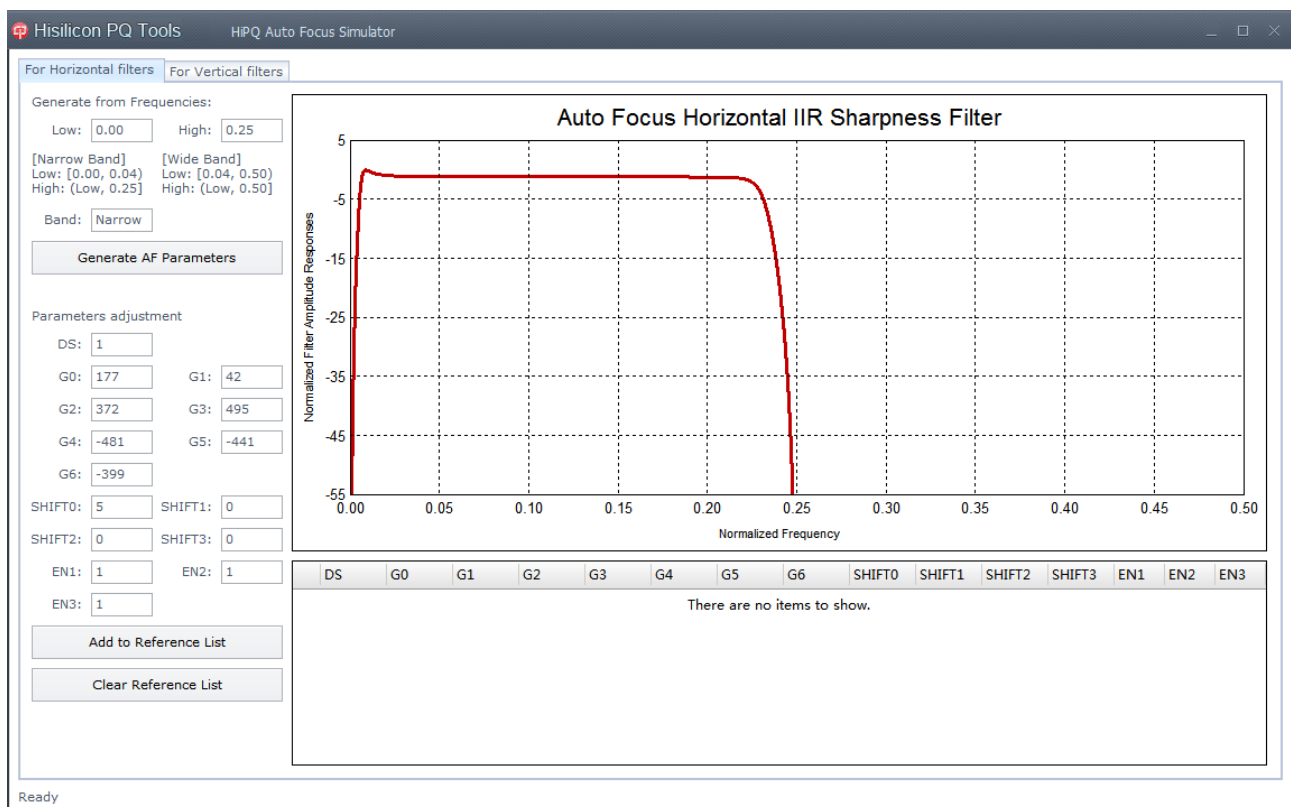
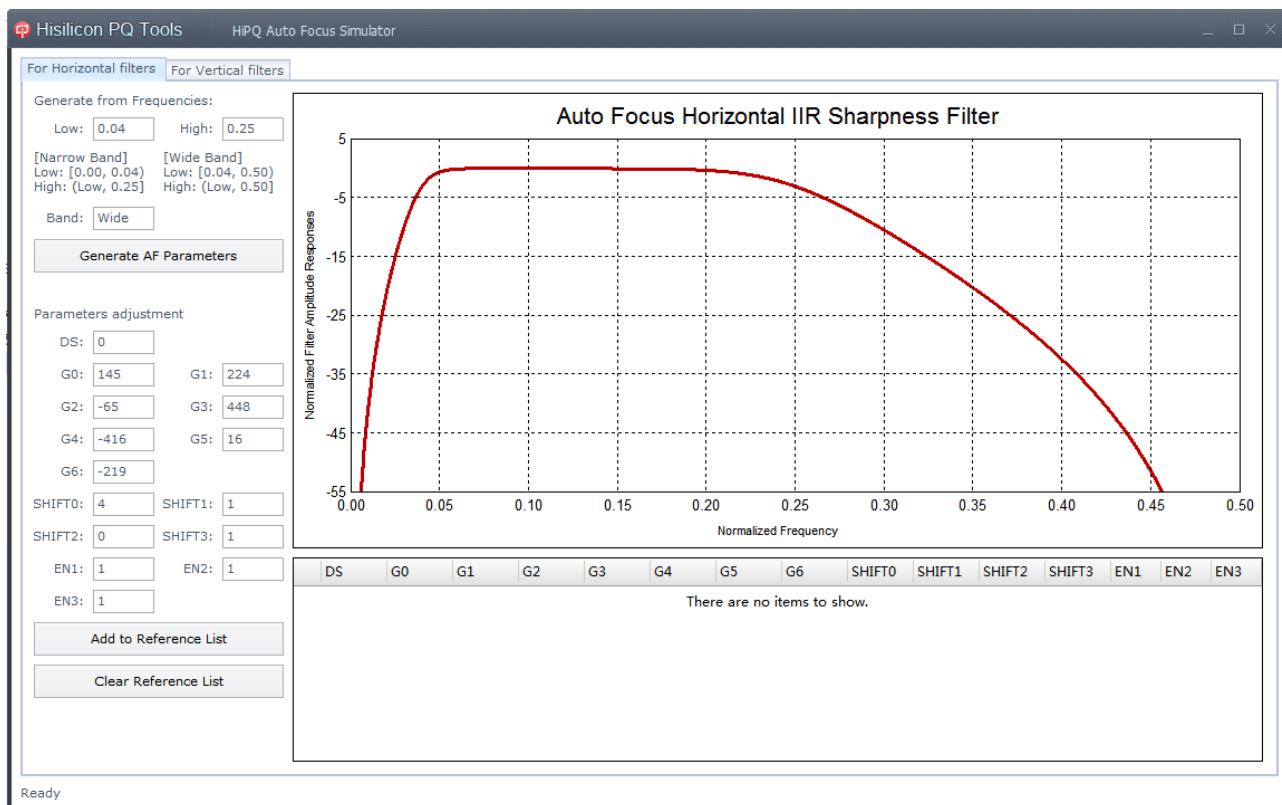


图13-17 通过截止频率获取滤波系数（宽带）



须知

截止频率在工具中会自动适配精度。水平滤波器的截止频率取小数点后两位，垂直滤波器的截止频率取小数点后一位。取精度时，多余的小数位数直接舍弃。

13.5.11.3 手动配置滤波系数

除了通过截止频率获取对应的滤波系数，用户还可以手动进行每个滤波系数项进行设置。在 Parameters adjustment 处，找到对应的参数并进行修改后，工具将自动计算这组参数的频率响应曲线并绘制在右侧。曲线以红色表示。

13.5.11.4 设置参考组

如果用户想要在不同组的滤波系数间对比，可以使用参考组功能。当用户需要将某一组滤波系数列为参考组时，点击“Add to Reference List”按钮，工具即会在右下角的列表中添加一行，列出该参考组的参数项，如图 13-18 所示。

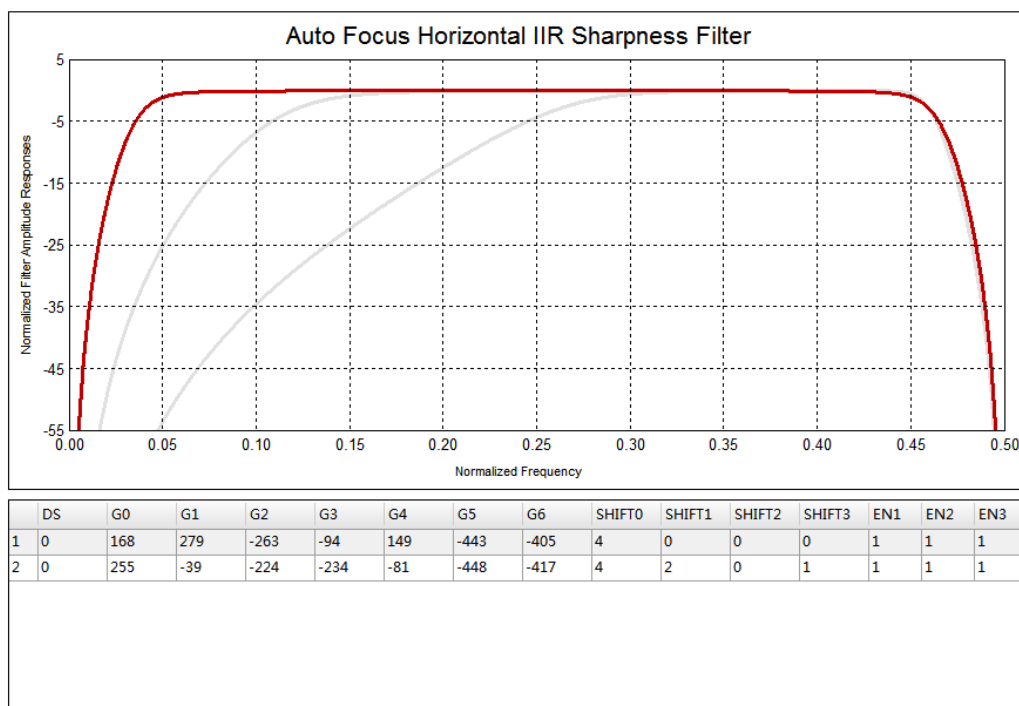


图13-18 参考组列表

	DS	G0	G1	G2	G3	G4	G5	G6	SHIFT0	SHIFT1	SHIFT2	SHIFT3	EN1	EN2	EN3
1	0	145	224	-65	448	-416	16	-219	4	1	0	1	1	1	1

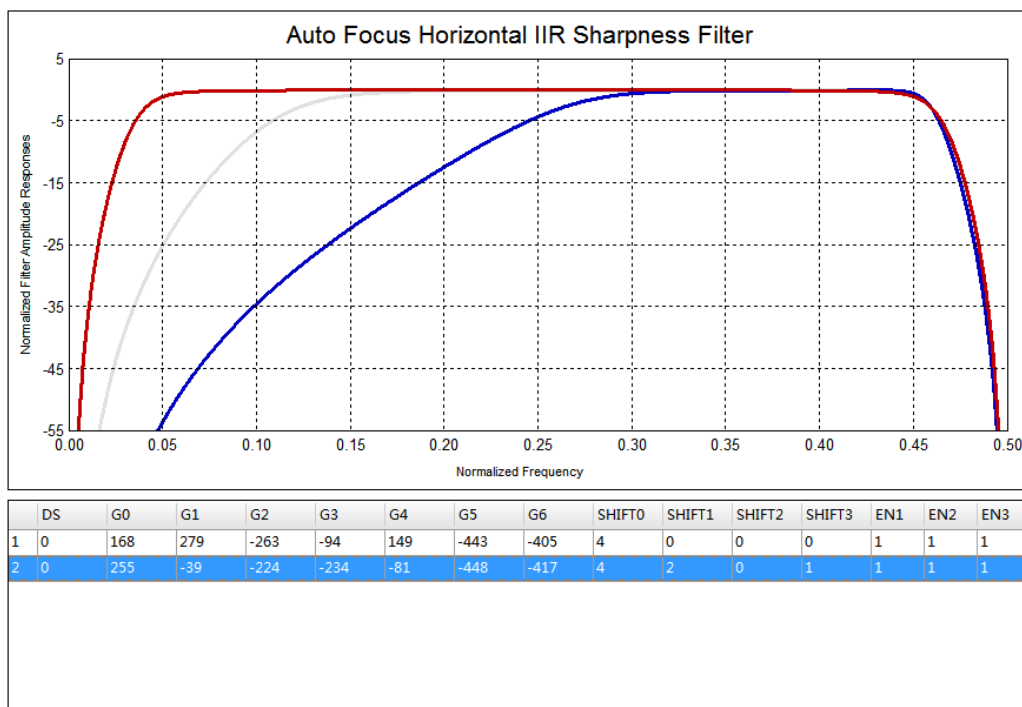
对于每一个参考组，工具还将在上方的坐标图上生成一条曲线，以灰色表示，如图 13-19 所示：

图13-19 参考组的曲线显示



用户还可以将某一参考组的曲线高亮显示出来。在列表中点击需要高亮显示的参考组，此时对应的曲线会在上方的坐标图中变成蓝色，如图 13-20 所示：

图13-20 选中参考组曲线高亮显示



通过这样的方式，用户可以在不同的参考组间进行频率响应曲线对比，从而发现更优的方案。

须知

- 在坐标图上，红色的当前参数曲线和蓝色的参考组高亮曲线相对其他的灰色参考组曲线处于更上方的位置。因此，会出现刚刚添加参考组时看不到灰色曲线的状况。此时只要修改滤波系数，使红色曲线重新绘制，就能看到灰色的参考曲线。
- 每个页签上最多只能添加 6 组参考组。如果需要添加更多参考组，请先点击 Clear Reference List 删除所有参考组。

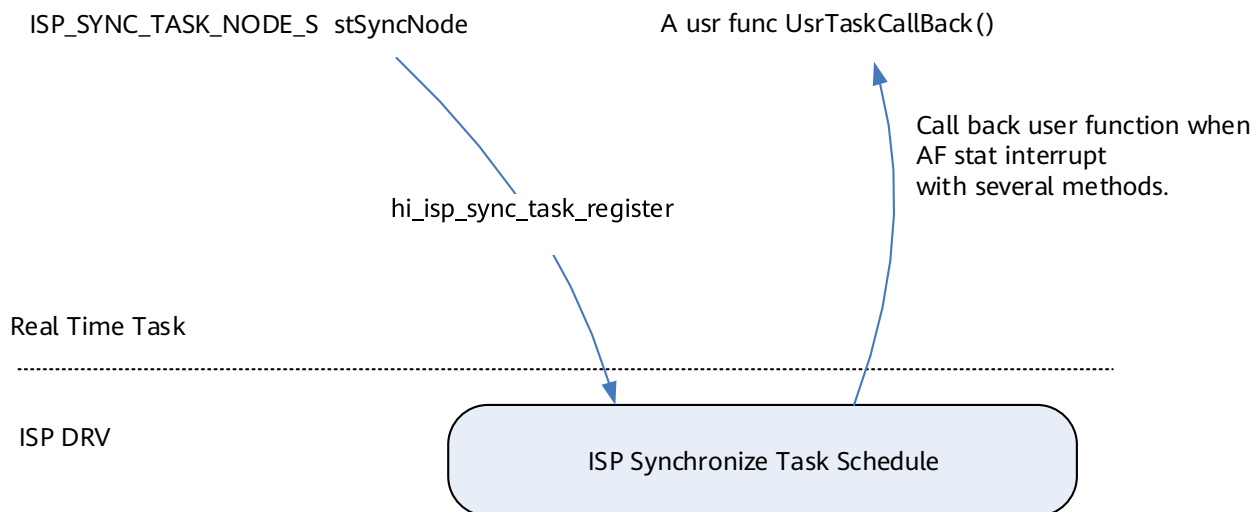
13.5.12 AF 同步回调

当一帧图像最后一个 pixel 通过 AF 模块后，统计值即更新，推荐用户通过 HI_MPI_ISP_GetVDTimeOut 同步获取统计值，AutoFocus 和 ZoomTracking 算法完成目标 focus 和 zoom position 的计算，如果用户使用的是 LINUX 系统，因为 user space 任务调度不能保证一致的实时性，建议将需要保证实时性的驱动配置放在 kernel space 完成。ISP 提供同步回调接口的注册，可以实现与 VD 同步。在本章有相



应的接口使用描述，用户可以将实时性要求较高的任务放在同步回调里面，底层提供 HwiIRQ，Workqueue 两种方式实现，可以选择相应的实现方式以确定实时级别。HiwiIRQ 是指任务放在中断服务中实现，实时性最高;Workqueue 的实时性取决于 linux 系统调用。

图13-21 AF 同步回调



13.5.12.1 API 参考

- [hi_isp_sync_task_register](#): 向 ISP 注册同步回调接口。
- [hi_isp_sync_task_unregister](#): 向 ISP 反注册同步回调接口。

hi_isp_sync_task_register

【描述】

向 ISP 注册同步回调接口。

【语法】

```
HI_S32 hi_isp_sync_task_register(VI_PIPE dev, ISP_SYNC_TASK_NODE_S * pstNewNode);
```

【参数】

参数名称	描述	输入/输出
dev	ISP 设备号。	输入



pstNewNode	新插入的同步回调节点。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败。

【需求】

头文件：isp_ext.h.

【注意】

- 使用前需要确保 ISP 驱动已加载。
- 因为 ISP 同步回调内部实现不保存用户传入的 pstNewNode 指向的实体，所以要求使用 [ISP_SYNC_TASK_NODE_S](#) 定义实体时不能为局部变量。

【举例】

无

【相关主题】

[hi_isp_sync_task_unregister](#)

hi_isp_sync_task_unregister

【描述】

向 ISP 反注册同步回调接口。

【语法】

```
HI_S32 hi_isp_sync_task_unregister(VI_PIPE dev, ISP\_SYNC\_TASK\_NODE\_S *pstDelNode)
```

【参数】

参数名称	描述	输入/输出
dev	ISP 设备号。	输入



pstDelNode	需要删除的同步回调节点	输入

【返回值】

返回值	描述
0	成功。
非 0	失败。

【需求】

头文件：isp_ext.h.

【注意】

使用前需要确保 ISP 驱动已加载。

【举例】

无

【相关主题】

[hi_isp_sync_task_register](#)

13.5.12.2 数据类型

- [ISP_SYNC_TSK_METHOD_E](#): 定义同步回调方法，决定实时性。
- [ISP_DRV_FOCUS_ZONE_S](#): AF 分区间统计信息。
- [ISP_DRV_FOCUS_STATISTICS_S](#): 分块统计信息。
- [ISP_DRV_AF_STATISTICS_S](#): AF 模块统计信息。
- [ISP_SYNC_TASK_NODE_S](#): 定义同步回调节点信息。

ISP_SYNC_TSK_METHOD_E

【说明】

定义同步回调方法，决定实时性。

【定义】

```
typedef enum hiISP_SYNC_TSK_METHOD_E
```



```
{  
    ISP_SYNC_TSK_METHOD_HW_IRQ = 0,  
    ISP_SYNC_TSK_METHOD_WORKQUEUE,  
    ISP_SYNC_TSK_METHOD_BUTT  
} ISP_SYNC_TSK_METHOD_E;
```

【成员】

成员名称	描述
ISP_SYNC_TSK_METHOD_HW_IRQ	使用硬件中断方式回调。
ISP_SYNC_TSK_METHOD_WORKQUEUE	使用工作队列方式回调。

【注意事项】

无

【相关数据类型及接口】

无

ISP_DRV_FOCUS_ZONE_S

【说明】

AF 分区间统计信息。

【定义】

```
typedef struct hiISP_DRV_FOCUS_ZONE_S  
{  
    HI_U16  u16v1;  
    HI_U16  u16h1;  
    HI_U16  u16v2;  
    HI_U16  u16h2;  
    HI_U16  u16y;  
    HI_U16  u16HlCnt;  
} ISP_DRV_FOCUS_ZONE_S;
```

【成员】



成员名称	描述
u16v1	分区间统计的 AF 垂直方向奇数列 FIR 滤波器的统计值。
u16h1	分区间统计的 AF 水平方向奇数行 IIR 滤波器的统计值。
u16v2	分区间统计的 AF 垂直方向偶数列 FIR 滤波器的统计值。
u16h2	分区间统计的 AF 水平方向偶数行 IIR 滤波器的统计值。
u16y	分区间统计的 AF 亮度统计值，区块中每个像素点的亮度累加值。
u16HlCnt	分区间统计的 AF 亮度超阈值计数器值。

【注意事项】

无

【相关数据类型及接口】

无

ISP_DRV_FOCUS_STATISTICS_S

【说明】

分块统计信息。

【定义】

```
typedef struct hiISP_DRV_FOCUS_STATISTICS_S
{
    ISP_DRV_FOCUS_ZONE_S
    stZoneMetrics[DRV_WDR_CHN_MAX][DRV_AF_ZONE_ROW][DRV_AF_ZONE_COLUMN];
} ISP_DRV_FOCUS_STATISTICS_S;
```

【成员】

成员名称	描述
stZoneMetrics	分区间 AF 统计信息结构体

【注意事项】



无

【相关数据类型及接口】

无

ISP_DRV_AF_STATISTICS_S

【说明】

AF 模块统计信息。

【定义】

```
typedef struct hiISP_DRV_AF_STATISTICS_S
{
    ISP_DRV_FOCUS_STATISTICS_S    stFEAFStat;
    ISP_DRV_FOCUS_STATISTICS_S    stBEAFStat;
} ISP_DRV_AF_STATISTICS_S;
```

【成员】

成员名称	描述
stFEAFStat	FE 模块 AF 统计信息结构体。
stBEAFStat	BE 模块 AF 统计信息结构体。

【注意事项】

无

【相关数据类型及接口】

无

ISP_SYNC_TASK_NODE_S

【说明】

定义同步回调节点信息。

【定义】

```
typedef struct hiISP_SYNC_TASK_NODE_S
{
```



```
ISP_SYNC_TSK_METHOD_E enMethod;  
HI_S32 ( *pfnIspSyncTskCallBack ) (HI_U64 u64Data);  
HI_U64 u64Data;  
const char *pszId;  
struct list_head list;  
ISP_DRV_AF_STATISTICS_S *pstFocusStat  
} ISP_SYNC_TASK_NODE_S;
```

【成员】

成员名称	描述
enMethod	回调方式。
pfnIspSyncTskCallBack	回调函数，用户注册时传入。
u64Data	回调函数参数，用户注册时传入。
pszId	节点 ID
list	list 节点，用于管理多个回调节点，无需关注。
pstFocusStat	AF 统计信息结构体。

【举例】

```
ISP_SYNC_TASK_NODE_S stSyncNode =  
{  
.enMethod = ISP_SYNC_TSK_METHOD_HW_IRQ,  
.pfnIspSyncTskCallBack = UsrTaskCallBack,  
.u64Data = 0,  
.pszId = "HardwareInterrupt "  
};
```

【注意事项】

无

【相关数据类型及接口】

[ISP_SYNC_TSK_METHOD_E](#)

13.6 不带光敏 AE 快速收敛模式（无光敏快启）使用说明

13.6.1 概述

无光敏快启是在没有光敏传感器反馈当前亮度的情况下，AE 算法在启动后以最快速收敛到目标亮度范围的功能。此功能目前在大部分场景下可以在 10 帧内收敛到目标亮度范围。其中目标亮度范围为[目标亮度-目标亮度*10%，目标亮度+目标亮度*10%]。例如，当目标亮度为 50，则目标亮度范围为[45,55]。在使用此功能时，Sensor 的驱动要进行相应的适配。

13.6.2 适配不带光敏 AE 快速收敛模式（无光敏快启）

步骤 1 适配 Sensor 的写寄存器方式。无光敏快启要求对配置给 Sensor 的曝光参数的延迟生效要求为 3 帧。即 AE 算法在第 1 帧对 Sensor 配置曝光参数，在第 4 帧时能够获取到相应曝光参数生效的亮度统计信息（即 Raw）。如果 Sensor 的延迟生效帧数不符合 3 帧，则需要对 Sensor 的驱动进行修改适配。一般来说，目前大部分的 Sensor 在上海海思平台上，总体的延迟生效帧数为 4 帧。可将 Sensor 驱动适配为调用 I2C 接口直接写 Sensor 寄存器的方式设定曝光参数（包括曝光行数、Sensor 模拟增益、Sensor 数字增益）。这样的方式可以使 Sensor 提前 1 帧生效曝光参数。具体可参考 F37 Sensor 的驱动代码进行适配。

需要注意将 Sensor 改为用 I2C 接口直接写寄存器的方式后，相应的曝光行数、Sensor 模拟增益、Sensor 数字增益和 ISP 数字增益的生效延迟帧数也要同保证同步，以确保曝光的同步。

步骤 2 配置 pfn_cmos_ae_quick_start_status_set 回调函数。无光敏快启收敛后，会调用此回调函数同时设定 bQuickStartStatus 为 HI_TRUE。此时通过此参数状态，将步骤 1 中的直接调用 I2C 接口方式配置寄存器的方式更改回普通方式。

步骤 3 设定无光敏快启相关参数：

1. pstAeSnsDft-> stQuickStart.bQuickStartEn 设置为 HI_TRUE 打开无光敏快启功能。
2. pstAeSnsDft-> stQuickStart .u8BlackFrameNum 设定初始坏帧数。因部分 Sensor 在启动时，会输出若干坏帧，为了确保 AE 算法能够及时获取到有效的统计数据。需要填入正确的坏帧数，没有坏帧则填入 0。
3. pstAeSnsDft-> stQuickStart .blrModeEn 设置无光敏快启下是否支持红外模式。如果是需要支持带有 IR CUT 和 IR LED 无光敏快启，则需要打开此模式。



4. `pstAeSnsDft-> stQuickStart . u32InitExposureIr` 设置无光敏快启红外模式下的初始曝光量。
5. `pstAeSnsDft-> stQuickStart . u32ISOThrIr` 设置无光敏快启普通模式切换到红外模式的 ISO 阈值。上层根据获取到的 ISO 值是否大于 `u32ISOThrIr` 判定是否要打开 IR CUT 和 IR LED 进入红外模式。同时 AE 快速收敛算法内部也会根据此参数判定是否要切换到快速收敛的红外模式进行继续收敛。
6. `pstAeSnsDft-> stQuickStart . u16IrCutDelayTime` 设置无光敏快启 IR CUT 完成整个切换需要的时间。单位为 ms。

以上变量值均可通过 Sensor 驱动中的 `sensor_set_init()` 函数由外部传入进行灵活设定。

步骤 4 标定初始曝光量。在相同的曝光参数下，由于不同 Sensor 具有不同的光电敏感度。同时不同大小的光圈和不同的滤光片的选择都会对 Sensor 的最终成像亮度有不同影响。当前的无光敏快启的初始曝光量需要进行标定。标定步骤和方法如下：

1. 将辉度箱设定为 EV8 亮度。
2. 将 `MaxHistOffset` 设置为 0，同时将 AE 算法设定为手动模式。
3. 将相机镜头对着辉度箱。调节曝光量，同时查看 Proc 调试信息的 `OriAve` 值（不带权重的画面平均亮度），当 `OriAve` 值为 80 时，此时对应的曝光量作为初始曝光量。

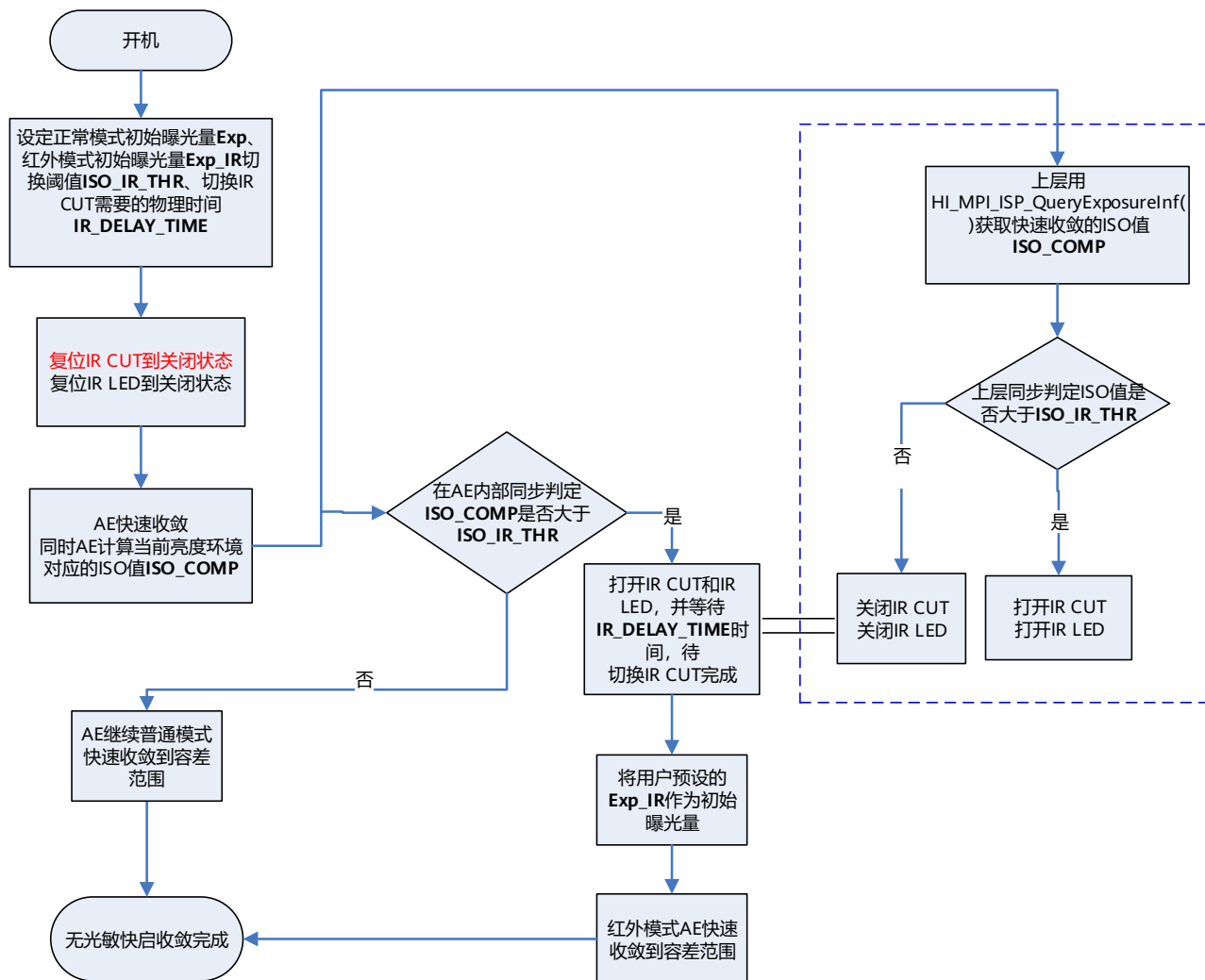
步骤 5 初始曝光量微调。对于不同的产品形态有不同的亮度使用范围，可以对步骤 4 的初始曝光量进行微调，加快无光敏快启 AE 算法收敛的速度，以适应不同产品需求。

----结束

13.6.3 注意事项

- 在以上步骤 2 中，将 Sensor 配置为用 I2C 接口直接写 Sensor 寄存器的方式设定曝光量。在无光敏快启收敛完成后，需要进行相应的配置将 Sensor 的配置方式设定回普通方式。
- 初始亮度 80 为推荐值，用户可以根据产品形态和常用应用场景亮度此基础上进行上下调节以达到最佳收敛速度。如果没有辉度箱标定环境。可以使用曝光时间 24ms，ISO100，光圈 F1.6 相应的等曝光量曝光参数组合作为初始曝光量的进行调试。
- 注意 AE 分块权重设置需要合理，推荐高权重区域和低权重区域的差值为 1 以内，比较极端的分块权重设定会影响无光敏快启收敛速度。

13.6.4 带 IR CUT 无光敏快启基本流程



- 在确定长时间处于低照度环境下，为了减少 IR CUT 的切换次数，可以不进行 IR CUT 复位操作，即省略如上红色字体部分操作。
- 蓝框内的流程为上层应用操作。



14 附录

14.1 注册函数的关系

[HI_MPI_ISP_AELibRegCallBack](#) 和 [HI_MPI_ISP_AWBLibRegCallBack](#) 这两个接口是 ISP firmware 库提供的钩子函数，用于开发 3A 算法库时实现注册动作。例如上海海思提供的 3A 算法库的 [HI_MPI_AE_Register](#) 和 [HI_MPI_AWB_Register](#) 接口，在实现时调用了相应的钩子函数，所以调用 [HI_MPI_AE_Register](#) 能实现 AE 算法库向 ISP firmware 库注册。

同样的，上海海思 3A 算法库同样也提供了钩子函数，用于 Sensor 库实现向 3A 算法库注册的动作。例如 [HI_MPI_AE_SensorRegCallBack](#) 和 [HI_MPI_AWB_SensorRegCallBack](#)，在 xxx_cmos.c 中可以看到调用了这些钩子函数的函数 `sensor_register_callback`。用户在开发 3A 算法库时，也可以通过提供钩子函数的方式，实现 Sensor 库向 3A 算法库的注册。

当然，ISP firmware 库也提供了钩子函数，用于 Sensor 库实现向 ISP firmware 库注册的动作。例如 [HI_MPI_ISP_SensorRegCallBack](#)，在 xxx_cmos.c 中可以看到调用了该钩子函数的函数 `sensor_register_callback`。

所以，当用户调用 [HI_MPI_AE_Register](#)、[HI_MPI_AWB_Register](#) 和 `sensor_register_callback` 就完成了 3A 算法库向 ISP firmware 库注册、Sensor 库向 3A 算法库和 ISP firmware 库注册。



说明

用户开发 3A 算法库时，请自行实现 [HI_MPI_AXX_Register](#) 接口。同时也请自行实现 [HI_MPI_AXX_SensorRegCallBack](#) 钩子函数，并在 `sensor_register_callback` 中增加调用该钩子函数的代码，相关代码可以参考 ISP firmware 库的开源代码。



14.2 扩展性的设计考虑

在代码中有 VI_PIPE、ALG_LIB_S、SENSOR_ID 这样一些概念，这些概念是出于架构扩展性的考虑。

VI_PIPE 主要考虑的是支持多个 ISP 单元的情形。无论是多个 ISP 硬件单元，或是一个 ISP 硬件单元分时复用，从软件意义上讲，需要预留出扩展性。

ALG_LIB_S 主要考虑的是支持多个算法库，并动态切换的情形。例如用户实现了一套 AE 算法代码，但注册两个库，分别用于正常场景和抓拍场景，那么这时候需要用结构体中的 s32Handle 来进行区分。例如用户实现了一套 AWB 算法代码，同时又想在某些场景下使用上海海思 AWB 算法库，那么这时候可以用结构体中的 acLibName 进行区分。当用户注册多个 AE 库，或 AWB 库时，ISP firmware 将会全部对它们进行初始化，但是在运行时，仅会调用有效的库，设置有效库的接口是 [HI_MPI_ISP_SetBindAttr](#)，通过此接口可以快速切换运算的库。

SENSOR_ID 仅起一个校验作用，确认注册给 ISP firmware 库和 3A 算法库的是同一款 sensor。

这些概念仅是设计时预留的冗余，如果完全不需要这些概念，可以在开发时去掉这些概念。

14.3 3A 架构的设计思路

设计思路基本是这样，ISP firmware 初始化并销毁各个算法单元；在运行时，提供前一帧的统计信息，并根据返回值配置寄存器，其他内容，均由用户开发。所以当用户替换自己的 3A 算法后，当前的 AE/AWB 的 MPI 不可复用，cmos.c 中的 AE/AWB 相关的内容不可复用，对于 AE 的权重配置和 AWB 的找白点配置的内容不可复用，这几个配置理论上是由 3A 算法配置，而不是从 ISP firmware 获取，ISP firmware 中仅有简单的初始化值。

3A 算法并不需要显式地去配置 ISP 寄存器，只需将需要配置的 ISP 寄存器值写到 [ISP_AE_RESULT_S](#) 和 [ISP_AWB_RESULT_S](#) 结构体中即可；也不需要显式地去读取 ISP 寄存器，只需从 [ISP_AE_INFO_S](#) 和 [ISP_AWB_INFO_S](#) 结构体中读取即可。



14.4 外部寄存器的说明

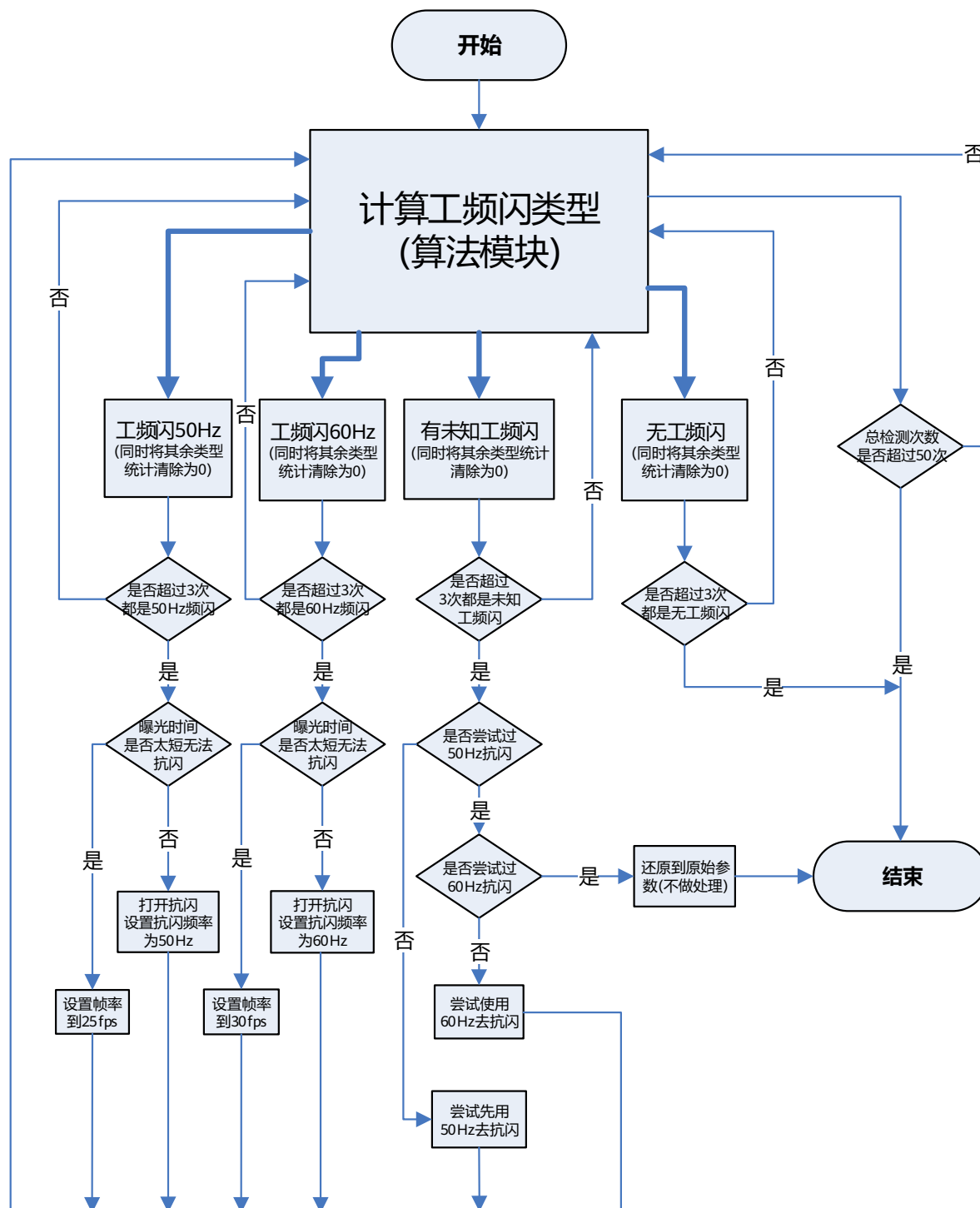
在 IPC 的应用中，通常除了业务主程序进程外，板端还会有另外的进程去支持 PC 端的工具来调节图像质量。ISP 的各个算法的许多状态、参数均驻留于全局变量中，不足以支持多进程的访问，所以引入外部寄存器的概念，用以支持多进程的业务场景。用户通过 PC 端工具与板端进程通信，调用上海海思提供的 MPI，实际是改变外部寄存器中的内容，从而改变业务主程序中的 ISP 的各个算法的状态和参数。

外部寄存器还能与实际的硬件寄存器通过统一的接口读写，形式上与实际的硬件寄存器无差别。

用户如果使用外部寄存器的话，有这些已封装好的接口可以使用，当然用户也可以有其他方案实现多进程的支持。外部寄存器的地址空间是自定义的，只要不冲突即可。详细请参考开源代码，接口定义在 `isp_vreg.h` 文件中。

14.5 工频闪类型自适应 Sample 用例处理流程

图14-1 工频闪类型自适应 Sample 用例处理流程





- 以上为工频闪类型自适应 Sample 用例的处理流程（calcflicker sample 用例 0）。此用例使用多次检测的结果进行确认后自动适配相应的抗闪参数。此用例作为使用 Auto Flicker Type Detection（自动工频闪类型检测）接口进行自适应抗闪的参考。
- 检测结果为 FLICKER_TYPE_UNKNOWN（有未知类型的工频闪）的原因有几种情况：
 1. Camera 摇晃导致检测不稳定。
 2. 图像中存在大面积的运动物体。
 3. 环境光源为同时存在 50Hz 和 60Hz 的混合光源。
 4. 环境光源为既不是 50Hz 也不是 60Hz 频率的非通用频率光源。以上这几种情况可根据实际情况灵活处理。



A 缩略语

缩写	全称
3A	AE, AWB and AF
ACM	Auto Color Managment
ACS	Auto Color Shading
AE	Auto Exposure
AEB	Automatic Exposure Bracketing
AF	Auto Focus
AG	Analog Gain
AGC	Auto Gain Control
API	Application Programming Interface
AWB	Auto White Balance Correction
CAC	Chromatic Aberration Correction
CA	Chroma Adjust
CSC	Color Space Conversion
CCM	Color Correction Matrix
CPU	Central Processing Unit
DG	Digital Gain
DIS	Digital Image Stabilization
DPC	Defect Pixel Correction
DRC	Dynamic Range Compression



缩写	全称
FPN	Fixed Pattern Noise
FW	Firmware
HDR	High Dynamic Range
IMP	Image Processing
ISO	ISO standard 12232:2006. A measure of photographic sensitivity to light
ISP	Image Signal Processor
LUT	Lookup Table
NP	Noise Profile
NR	Noise Reduction
WB	White Balance
WDR	Wide Dynamic Range
YCbCr	Method of encoding RGB colorspace according to the ITU-R BT.601 or ITU- R BT 709 standards
YUV	A method of encoding RGB colorspace in practice equivalent to YCbCr