



Hi3516A/Hi3516D_ISP_3A

开发指南

文档版本 05

发布日期 2016-10-28

版权所有 © 深圳市海思半导体有限公司 2014-2016。保留一切权利。

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

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前言

概述

本文档描述 Hi3516A/Hi3516D_ISP_3A 的功能、如何使用与开发。3A 功能包括 AE、AWB、AF。



说明

本文未做特殊说明，Hi3516D 与 Hi3516A 完全一致。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3516A_ISP_3A	-
Hi3516D_ISP_3A	-


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
- 单板硬件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。



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

修订记录

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

文档版本 05 (2016-10-28)

2.2.2.1、3.2 和 3.4 小节涉及修改

文档版本 04 (2016-01-29)

2.2.2、3.2 和 3.3 小节涉及修改

文档版本 03 (2015-06-15)

2.2 节，表 2-5 有更新

3.2 节，表 3-4 有修改；3.3 节有更新

文档版本 02 (2015-02-10)

2.2.1 和 2.2.2 有修改，新增表 2-3

表 3-3、表 3-4、表 3-7、表 3-8 均有修改；3.3 节中内容有新增

文档版本 01 (2014-12-10)

第 1 次正式发布，并添加 Hi3516D 的相关内容



目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 功能描述.....	1
1.2.1 设计思路	1
1.2.2 文件组织	2
1.2.3 开发模式	2
1.2.4 内部流程	3
2 使用者指南.....	5
2.1 软件流程.....	5
2.2 Sensor 对接.....	9
2.2.1 Sensor 注册 ISP 库.....	9
2.2.2 Sensor 注册 3A 算法库.....	16
3 开发者指南.....	23
3.1 概述.....	23
3.2 AE 算法注册 ISP 库.....	23
3.3 AWB 算法注册 ISP 库	28
3.4 开发用户 AF 算法.....	37
3.4.1 AF 算法结构.....	37
3.4.2 AF 同步回调.....	38
4 附录.....	43
4.1 注册函数的关系.....	43
4.2 扩展性的设计考虑.....	43
4.3 3A 架构的设计思路.....	44
4.4 外部寄存器的说明	44



插图目录

图 1-1 ISP firmware 设计思路	1
图 1-2 ISP firmware 文件组织	2
图 1-3 ISP firmware 内部流程	4
图 1-4 ISP firmware 软件结构	4
图 2-1 ISP firmware 使用流程	6
图 2-2 Sensor 适配示意图	9
图 2-3 Sensor 向 ISP 库注册的回调函数	9
图 2-4 Sensor 向 AE 库注册的回调函数	17
图 2-5 Sensor 向 AWB 库注册的回调函数	21
图 3-1 AE 算法向 ISP 库注册的回调函数	23
图 3-2 AWB 算法向 ISP 库注册的回调函数	29
图 3-3 统计信息参数示意图	34
图 3-4 AF 算法结构图	37
图 3-5 AF 同步回调	38



表格目录

表 2-1 Sensor 向 ISP 库注册的回调函数	10
表 2-2 ISP_CMOS_DEFAULT_S 的成员变量	11
表 2-3 ISP_SNS_REGS_INFO_S 的成员变量	15
表 2-4 Sensor 向 AE 库注册的回调函数	17
表 2-5 AE_SENSOR_DEFAULT_S 的成员变量	18
表 2-6 Sensor 向 AWB 库注册的回调函数	21
表 2-7 AWB_SENSOR_DEFAULT_S 的成员变量	21
表 3-1 AE 算法向 ISP 库注册的回调函数	24
表 3-2 初始化参数 ISP_AE_PARAM_S 的成员变量	25
表 3-3 统计信息 ISP_AE_INFO_S 的成员变量	25
表 3-4 运算结果 ISP_AE_RESULT_S 的成员变量	27
表 3-5 AWB 算法向 ISP 库注册的回调函数	29
表 3-6 初始化参数 ISP_AWB_PARAM_S 的成员变量	30
表 3-7 统计信息 ISP_AWB_INFO_S 的成员变量	30
表 3-8 运算结果 ISP_AWB_RESULT_S 的成员变量	32



1 概述

1.1 概述

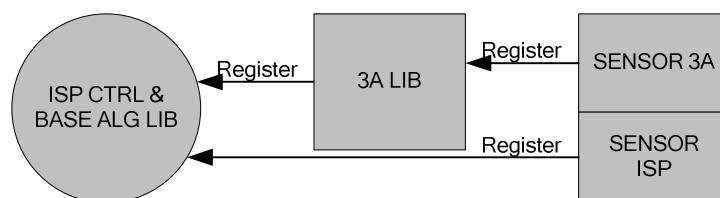
Hi3516A_ISP_3A 版本依赖于相应的 SDK 大版本，通过一系列数字图像处理算法完成对数字图像的效果处理。主要包含 Firmware 框架及海思 3A 库，Firmware 提供算法的基本框架，处理统计信息，驱动数字图像处理算法，并包含坏点校正、去噪、色彩增强、镜头阴影校正等处理。3A 库以注册的方式，添加到 Firmware 中，完成曝光、白平衡、色彩还原等处理。

1.2 功能描述

1.2.1 设计思路

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

图1-1 ISP firmware 设计思路



不同的 sensor 都向 ISP 库和 3A 算法库注册控制函数，这些函数都以回调函数的形式存在。ISP 控制单元调度基础算法单元和 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。

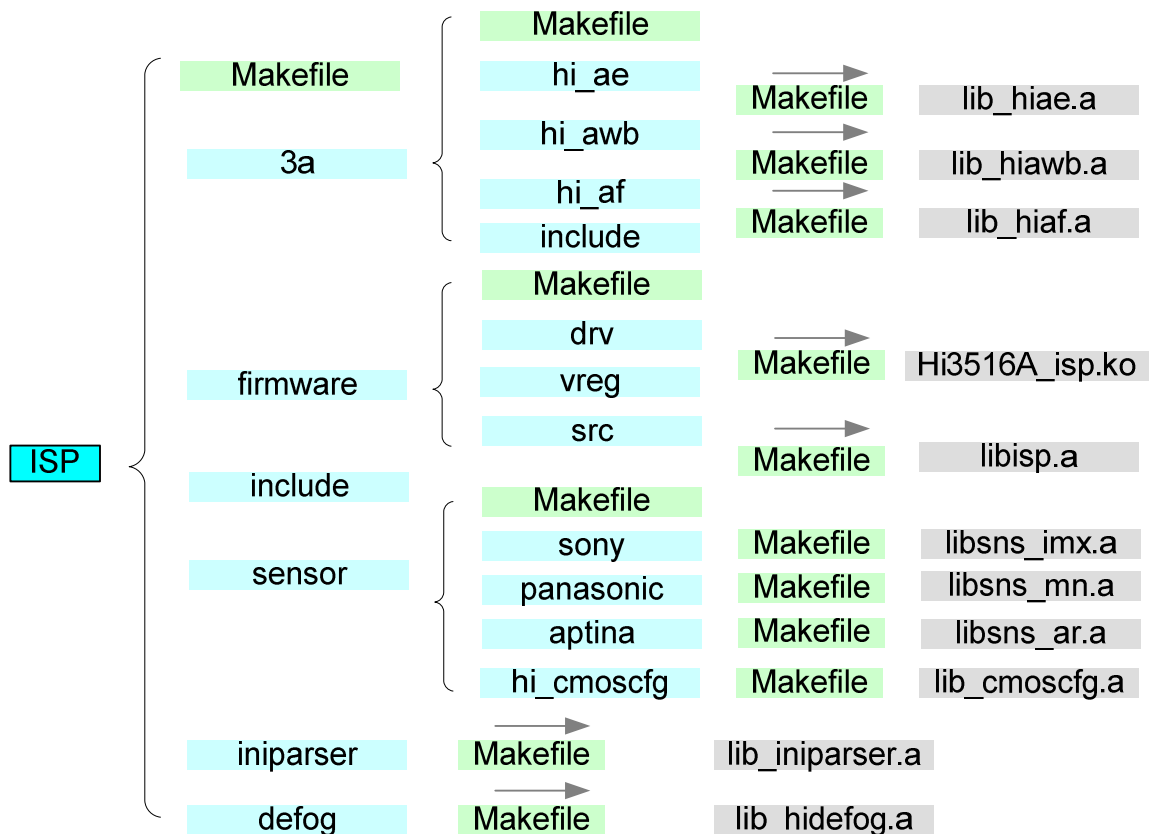


1.2.2 文件组织

ISP Firmware 的文件组织结构如图 1-2 所示，ISP 库和 3A 库、sensor 库、iniparser 库、defog 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/AF 算法库，用户也可以基于统一的接口界面开发自己的 3a 算法。Sensor 文件夹中包含了各个 sensor 的驱动程序，该部分代码开源。hi_cmoscfg 文件夹中包含解析 ini 文件所需的公共程序，该部分代码开源。iniparser 文件夹包含 ini 解析函数库，也可用于其它应用开发。defog 文件对应去雾算法程序，该部分代码不开源。

图1-2 ISP firmware 文件组织



1.2.3 开发模式

SDK 中给出的形式支持用户的多种开发模式，用户可以使用海思的 3A 算法库，也可以根据 ISP 库提供的 3A 算法注册接口，实现自己的 3A 算法库开发，或者可以部分使用海思 3A 算法库，部分实现自己的 3A 算法库，例如 AE 使用 lib_hiae.a，AWB 使用自己的 3A 算法库。SDK 提供了灵活多变的支持方式。



1.2.3.1 使用海思 3A 算法库

用户需要根据 ISP 基础算法单元和海思 3A 算法库给出的 sensor 适配接口去适配不同的 sensor。Sensor 文件夹中包含两个主要文件：

- sensor_cmos.c
该文件中主要实现 ISP 需要的回调函数，这些回调函数中包含了 sensor 的适配算法，不同的 sensor 可能有所不同。
- sensor_ctrl.c
sensor 的底层控制驱动，主要实现 sensor 的读写和初始化动作。用户可以根据 sensor 的 datasheet 进行这两个文件的开发，必要的时候可以向 sensor 厂家寻求支持。

1.2.3.2 开发 3A 算法库

用户需要根据 ISP 基础算法单元给出的 sensor 适配接口去适配不同的 sensor，用户开发的 3A 算法库需要自定义数据接口和回调函数去适配和控制不同的 sensor。用户自行开发 3A 算法后，可以调用 mpi_isp.h 中的接口，不能调用 mpi_ae.h 和 mpi_awb.h 中的接口。



说明

高级用户可以基于 Firmware 提供的统计信息进行自己的算法库开发，当然这需要对统计信息比较熟悉，同时具有算法开发能力。

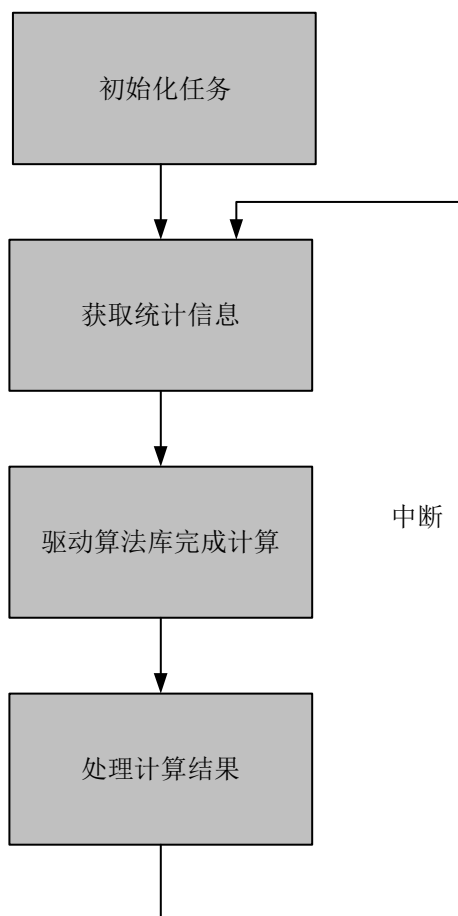
1.2.4 内部流程

Firmware 内部流程，如图 1-3 所示。首先完成 Firmware 控制单元的初始化、基础算法单元的初始化、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。当初始化完成之后，Firmware 由中断驱动，每帧从内核态获取统计信息，并驱动基础算法单元和 3A 算法库完成计算，并反馈计算结果，配置 ISP 寄存器和 sensor 寄存器。

同时用户可以通过 ISP 的 MPI，控制和改变 Firmware 中包含的基础算法单元的内部数据和状态，定制自己的图像质量效果。如果用户使用海思提供的 3A 算法库，可以通过 3A 算法库的 MPI，改变 3A 算法库的内部数据和状态，调节曝光、白平衡和色彩还原。

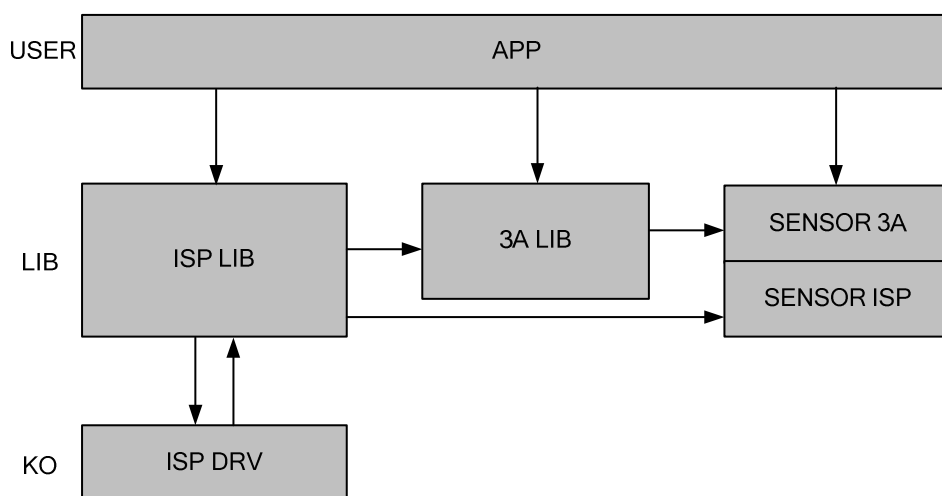


图1-3 ISP firmware 内部流程



Firmware 的软件结构如如图 1-4 所示。

图1-4 ISP firmware 软件结构





2 使用者指南

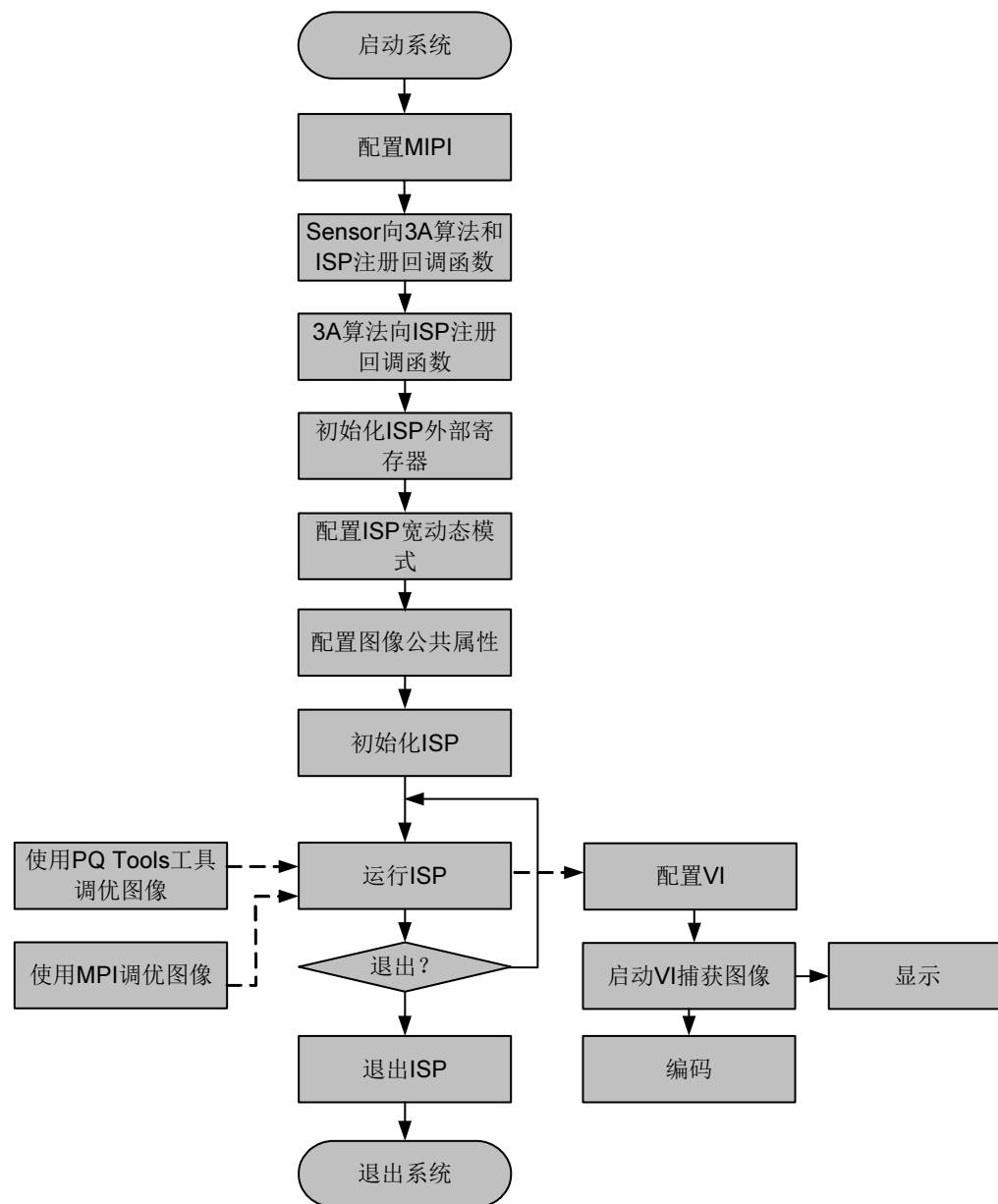
2.1 软件流程

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 2-1 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2-1 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 stLib;
ISP_PUB_ATTR_S stPubAttr;
pthread_t isp_pid;
/* 注册sensor库 */
s32Ret = sensor_register_callback();
```



```
if (HI_SUCCESS != s32Ret)
{
    printf("register sensor failed!\n");
    return s32Ret;
}

/* 注册海思AE算法库 */
stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AE_LIB_NAME);
s32Ret = HI_MPI_AE_Register(&stLib);
if (HI_SUCCESS != s32Ret)
{
    printf("register ae lib failed!\n");
    return s32Ret;
}

/* 注册海思AWB算法库 */
stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AWB_LIB_NAME);
s32Ret = HI_MPI_AWB_Register(&stLib);
if (HI_SUCCESS != s32Ret)
{
    printf("register awb lib failed!\n");
    return s32Ret;
}

/* 注册海思AF算法库 */
stLib.s32Id = 0;
strcpy(stLib.acLibName, HI_AF_LIB_NAME);
s32Ret = HI_MPI_AF_Register(&stLib);
if (HI_SUCCESS != s32Ret)
{
    printf("register af lib failed!\n");
    return s32Ret;
}

/* 初始化ISP外部寄存器 */
s32Ret = HI_MPI_ISP_MemInit(IspDev);
if (s32Ret != HI_SUCCESS)
{
    printf("%s: HI_MPI_ISP_Init failed!\n", __FUNCTION__);
    return s32Ret;
}
```



```
/* 配置ISP宽动态模式 */
ISP_WDR_MODE_S stWdrMode;
stWdrMode.enWDRMode = enWDRMode;
s32Ret = HI_MPI_ISP_SetWDRMode(0, &stWdrMode);
if (HI_SUCCESS != s32Ret)
{
    printf("start ISP WDR failed!\n");
    return s32Ret;
}

/* 配置图像公共属性 */
s32Ret = HI_MPI_ISP_SetPubAttr(IspDev, &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();
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();
if (HI_SUCCESS != s32Ret)
{
    printf("isp exit failed!\n");
}
```



```
        return s32Ret;
    }

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



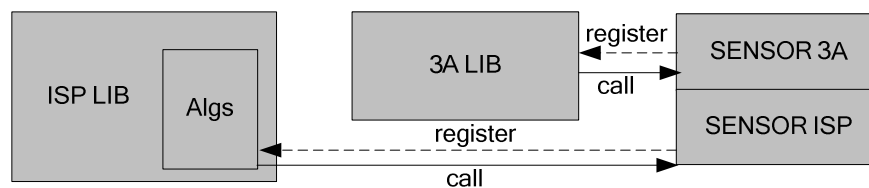
说明

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。

2.2 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 2-2](#) 表示了 Sensor 与 ISP 库和 3A 算法库之间的关系。

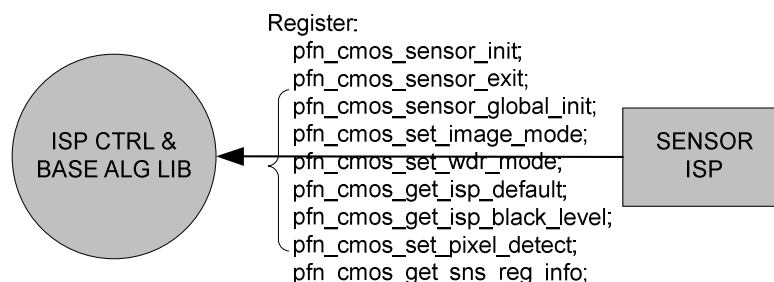
图2-2 Sensor 适配示意图



2.2.1 Sensor 注册 ISP 库

Sensor 注册 ISP 库调用 `HI_MPI_ISP_SensorRegCallBack`，如[图 2-3](#)所示，详细说明参见《HiISP 开发参考》。

图2-3 Sensor 向 ISP 库注册的回调函数



【举例】

```
ISP_SENSOR_REGISTER_S stIspRegister;
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIspRegister.stSnsExp;
memset(pstSensorExpFunc, 0, sizeof(ISP_SENSOR_EXP_FUNC_S));
```




```
pstSensorExpFunc->pfn_cmos_sensor_init = sensor_init;
pstSensorExpFunc->pfn_cmos_sensor_exit = sensor_exit;
pstSensorExpFunc->pfn_cmos_sensor_global_init = sensor_global_init;
pstSensorExpFunc->pfn_cmos_set_image_mode = cmos_set_image_mode;
pstSensorExpFunc->pfn_cmos_set_wdr_mode = cmos_set_wdr_mode;
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_get_sns_reg_info = cmos_get_sns_regs_info;

ISP_DEV IspDev = 0;
HI_S32 s32Ret;
s32Ret = HI_MPI_ISP_SensorRegCallBack(IspDev, IMX178_ID,
&stIspRegister);
if (s32Ret)
{
    printf("sensor register callback function failed!\n");
    return s32Ret;
}
```

需要在 xxx_cmos.c 中实现以下回调函数：

表2-1 Sensor 向 ISP 库注册的回调函数

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	退出 sensor 的回调函数指针。
pfn_cmos_sensor_global_init	初始化全局变量的回调函数指针。
pfn_cmos_set_image_mode	设置图像模式的回调函数指针。
pfn_cmos_set_wdr_mode	设置 wdr 模式和线性模式切换的回调函数指针。
pfn_cmos_get_isp_default	获取 ISP 基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取 sensor 的黑电平值的回调函数指针。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_get_sns_reg_info	设置 sensor 曝光与增益延时的回调函数指针。



说明

如果回调函数暂不实现，可以实现空函数，或者将回调函数指针置为 NULL 即可。



一般情况下，cmos sensor 曝光时间相关寄存器从配置到生效的时间延迟最大。若能够在消隐区完成对 sensor 的配置，sensor 曝光时间通常会在配置后的第 2 帧生效，增益会在配置后的第 1 帧生效。pfn_cmos_get_sns_reg_info 用于保证 AE 计算出来的每帧 sensor 曝光时间和增益配置同步以避免闪烁现象。pfn_cmos_get_sns_reg_info 调用参数 ISP_SNS_REGS_INFO_S 支持 I2C 和 SPI 接口，成员变量具体含义请参考表 2-3，其中 u8Cfg2ValidDelayMax 可用于保证 ISP 寄存器与 sensor 寄存器同步生效，如 ISPDgain 和曝光比。举例来说，若 sensor 曝光时间在配置后的第 2 帧生效，增益在配置后的第 1 帧生效，那么 u8Cfg2ValidDelayMax 需设置为 2，表示所有 sensor 寄存器从配置到生效延迟的帧数最大值为 2，曝光时间的 u8DelayFrmNum 需设置为 0 表示曝光时间不需延迟配置，增益的 u8DelayFrmNum 需设置为 1 表示增益需延迟 1 帧配置，如此正好保证配置下去后 AE 计算出来的结果是同时生效的；若 sensor 曝光时间和增益都在配置后的第 2 帧生效，则 u8Cfg2ValidDelayMax 需设置为 2，曝光时间和增益的 u8DelayFrmNum 都需设置为 0。除了曝光时间和增益外，可以通过增加变量个数，同步配置 sensor 每帧最大曝光时间 VMAX（以曝光行数为单位）或每行最大曝光时间 HMAX（以每行曝光对应的时钟个数为单位），如此在切换帧率时就不会出现闪烁现象，VMAX 或 HMAX 的 u8DelayFrmNum 一般与曝光时间的 u8DelayFrmNum 相同。对于不同 sensor，具体的参数配置需参考 sensor datasheet。在检验曝光时间与增益是否同步时，可以打开抗闪，如果不同步在抗闪时间改变时容易出现画面闪烁。

表2-2 ISP_CMOS_DEFAULT_S 的成员变量

成员名称	子成员名称	描述
stDrc	bEnable	HI_FALSE：关闭 DRC 功能； HI_TRUE：使能 DRC 功能。 线性模式默认值为 HI_FALSE，WDR 模式默认值为 HI_TRUE，且必须设置为 HI_TRUE。
	u32BlackLevel	DRC 算法的下限值，低于 u32Blacklevel 的像素点不经过 DRC 算法处理。 取值范围：[0, 0xFFFF]。 默认值为 0。
	u32WhiteLevel	DRC 算法的上限值，高于 u32Whitelevel 的像素点不经过 DRC 算法处理。 取值范围：[0, 0xFFFF]。 线性模式默认值为 0x4FF，WDR 模式默认值为 0xFFFF。
	u32SlopeMax	DRC tone curve 控制参数，用于限制 DRC 曲线暗区斜率（增益）的最大值。 取值范围：[0, 0xFF]。 线性模式默认值为 0x30，WDR 模式默认值为 0x38。
	u32SlopeMin	DRCtone curves 控制参数，用于限制 DRC 曲线亮区斜率（增益）的最小值。



成员名称	子成员名称	描述
		取值范围：[0, 0xFF]。 线性模式默认值为 0x0，WDR 模式默认值为 0xC0。
	u32VarianceSpace	DRC 算法的空间敏感度。该值越大，表示每个像素生成自己的 tone_curve 时会参考越多周围的像素。 取值范围：[0x0, 0xF]。线性模式默认值为 0x4，WDR 模式默认值为 0xA。
	u32VarianceIntensity	DRC 算法的亮度敏感度。该值越大，表示每个像素的 tone_curve 与周围像素的差异越小。 取值范围：[0x0, 0xF]。 线性模式默认值为 0x1，WDR 模式默认值为 0x4。
stAgcTbl	bValid	该结构体的数据是否有效，取值范围为 [0,1]。
	au8SharpenAltD	根据增益动态调节图像大边缘锐度的插值数组，取值范围为[0,255]。
	au8SharpenAltUd	根据增益动态调节图像小纹理锐度的插值数组，取值范围为[0,255]。
	au8SnrThresh	根据增益动态设置图像去噪强度的插值数组，取值范围为[0,255]。
	au8DemosaicLumThresh	该数组用来设置图像的大边缘锐度的亮度门限值，建议用户使用默认参数值，取值范围为[0,255]。
	au8DemosaicNpOffset	该数组用来设置图像的噪声参数，建议用户使用默认参数值，取值范围为 [0,255]。
	au8GeStrength	该数组用来设置绿色平衡模块的参数，建议用户使用默认参数值，取值范围为 [0,255]。
stNoiseTbl	bValid	该结构体的数据是否有效，取值范围为 [0,1]。
	au8NoiseProfileWeightLut	该数组用来设置与 sensor 特性相关的噪声型式，作为去噪模块的输入，建议用户使用默认参数值，取值范围为 [0,255]。
	au8DemosaicWeightLut	该数组用来设置与 sensor 特性相关的噪



成员名称	子成员名称	描述
		声型式，作为 demosaic 模块的输入，建议用户使用默认参数值，取值范围为[0,255]。
stDemosaic	bValid	该结构体的数据是否有效，取值范围为[0,1]。
	u8VhSlope	垂直/水平混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8AaSlope	角度混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8VaSlope	VH-AA 混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8UuSlope	未定义的混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8SatSlope	饱和度混合的斜率门限，推荐使用默认值，取值范围为[0,255]。
	u8AcSlope	高频分量滤波斜率门限，推荐使用默认值，取值范围为[0,255]。
	u16VhThresh	垂直/水平混合的门限，推荐使用默认值，取值范围为[0,0xFFFF]。
	u16AaThresh	角度混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16VaThresh	VA 混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16UuThresh	未定义的混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16SatThresh	饱和度混合门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
	u16AcThresh	高频分量滤波门限，推荐使用默认值，取值范围为[0, 0xFFFF]。
stGammafe	bValid	该结构体的数据是否有效，取值范围为[0,1]。WDR 模式必须配置。
	au16Gammafe0	GammaFe 表 0，用于减小查找表插值时出现的误差，与 au16Gammafe1 共同完成图像压缩的功能，由校正工具生成取值范围为[0,0xFFFF]。
	au16Gammafe1	GammaFe 表 1，与 au16Gammafe0 共同完成图像压缩的功能，由校正工具生



成员名称	子成员名称	描述
		成，取值范围为[0,0xFFFF]。
stGamma	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果使用默认 gamma 曲线，可以配置为无效。
	au16Gamma	Gamma 表，取值范围为[0,0xFFFF]。
stShading	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果不需要 shading 校正，可以配置为无效。
	u16RCenterX	R 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16RCenterY	R 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterX	G 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterY	G 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16BCenterX	B 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16BCenterY	B 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	au16RShadingTbl	R 分量的校正表。 取值范围为[0x0, 0xFFFF]
	au16GShadingTbl	G 分量的校正表。 取值范围为[0x0, 0xFFFF]
	au16BShadingTbl	B 分量的校正表。 取值范围为[0x0, 0xFFFF]
	u16ROffCenter	R 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16GOffCenter	G 分量中心点与最远的角的距离。距离越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16BOffCenter	B 分量中心点与最远的角的距离。距离越大，数值越小。



成员名称	子成员名称	描述
		取值范围为[0x0, 0xFFFF]
	u16TblNodeNum	每个分量校正表中使用到的节点个数。 取值范围为[0x0, 0x81]，默认值为0x81。
stRgbSharpen	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果使用默认 SharpenRGB 曲线，可以配置为无效。
	u8LutCore	主要影响 SharpenRGB 曲线最小边缘幅值到曲线峰值对应边缘幅值的曲线斜率；同时影响峰值对应边缘幅值到最大边缘幅值的曲线斜率。取值范围为[0, 255]。
	u8LutStrength	Sharpen 曲线的强度调节参数，取值范围为[0, 127]。
	u8LutMagnitude	主要影响 SharpenRGB 曲线峰值对应边缘幅值到最大边缘幅值的曲线斜率；同时影响最小边缘幅值到曲线峰值对应边缘幅值的曲线斜率。取值范围为[0, 31]。
stSensorMaxResolution	u32MaxWidth	Sensor 支持的最大宽度，防止分辨率切换时配置超过 sensor 支持的宽度。
	u32MaxHeight	Sensor 支持的最大高度，防止分辨率切换时配置超过 sensor 支持的高度。

表2-3 ISP_SNS_REGS_INFO_S 的成员变量

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。



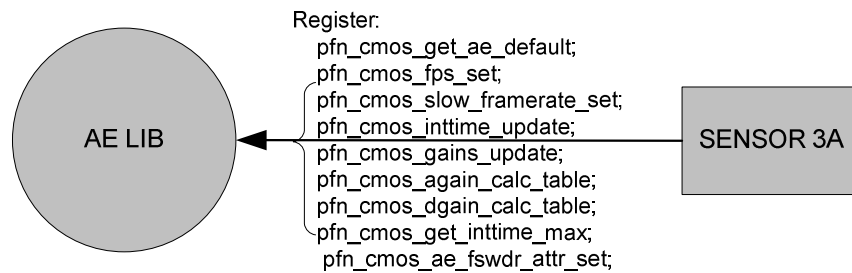
成员名称	子成员名称	描述
astI2cData	bUpdate	HI_TRUE: 数据会配置 sensor 寄存器; HI_FALSE: 数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u8DevAddr	Sensor 设备地址。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。
astSspData	bUpdate	HI_TRUE: 数据会配置 sensor 寄存器; HI_FALSE: 数据不会配置 sensor 寄存器。
	u8DelayFrmNum	sensor 寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
	u32DevAddr	Sensor 设备地址。
	u32DevAddrByteNum	Sensor 设备地址位宽。
	u32RegAddr	Sensor 寄存器地址。
	u32AddrByteNum	Sensor 寄存器地址位宽。
	u32Data	Sensor 寄存器数据。
	u32DataByteNum	Sensor 寄存器数据位宽。

2.2.2 Sensor 注册 3A 算法库

2.2.2.1 Sensor 注册 AE 算法库

Sensor 注册 AE 算法库调用 HI_MPI_AE_SensorRegCallBack, 如图 2-4 所示, 详细说明参见《HiISP 开发参考》。

图2-4 Sensor 向 AE 库注册的回调函数



【举例】

```
ALG_LIB_S stLib;
AE_SENSOR_REGISTER_S stAeRegister;
AE_SENSOR_EXP_FUNC_S *pstExpFuncs = &stAeRegister.stSnsExp;
pstExpFuncs->pfn_cmos_get_ae_default = cms_get_ae_default;
pstExpFuncs->pfn_cmos_fps_set = cms_fps_set;
pstExpFuncs->pfn_cmos_slow_framerate_set= cms_slow_framerate_set;
pstExpFuncs->pfn_cmos_inttime_update = cms_inttime_update;
pstExpFuncs->pfn_cmos_gains_update = cms_gains_update;
pstExpFuncs->pfn_cmos_again_calc_table = cms_again_calc_table;
pstExpFuncs->pfn_cmos_dgain_calc_table = cms_dgain_calc_table;
pstExpFuncs->pfn_cmos_get_inttime_max = cms_get_inttime_max;
pstExpFuncs->pfn_cmos_ae_fswdr_attr_set = cms_ae_fswdr_attr_set;
ISP_DEV IspDev = 0;
stLib.s32Id = 0;
strncpy(stLib.acLibName, HI_AE_LIB_NAME, sizeof(HI_AE_LIB_NAME));
s32Ret = HI_MPI_AE_SensorRegCallBack(IspDev, &stLib, IMX178_ID,
&stAeRegister);
if (s32Ret)
{
    printf("sensor register callback function to ae lib failed!\n");
    return s32Ret;
}
```

需要在 xxx_cmos.c 中实现以下回调函数：

表2-4 Sensor 向 AE 库注册的回调函数

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 的帧率的回调函数指针。



成员名称	描述
pfn_cmos_slow_framerate_set	降低 sensor 的帧率的回调函数指针。
pfn_cmos_inttime_update	设置 sensor 的曝光时间的回调函数指针。
pfn_cmos_gains_update	设置 sensor 的模拟增益和数字增益的回调函数指针。
pfn_cmos_again_calc_table	查表方式计算 AE 模拟增益的回调函数指针。
pfn_cmos_dgain_calc_table	查表方式计算 AE 数字增益的回调函数指针。
pfn_cmos_get_inttime_max	2To1_LINE 或 2To1_FRAME WDR 模式计算短曝光帧的最大曝光时间的回调函数指针。
pfn_cmos_ae_fswdr_attr_set	设置 FSWDR 运行模式。



说明

如果回调函数暂不实现，可以实现空函数，或者将回调函数指针置为 NULL 即可。

多帧合成 WDR 模式，需要调用 pfn_cmos_get_inttime_max 计算短曝光帧的最大曝光时间，AE 算法根据短曝光帧的限制和曝光比计算得到长、短帧的曝光时间和增益，其余流程与线性模式一致。为了保证多帧合成 WDR 模式在正常场景的效果，海思 AE 算法可以根据直方图统计信息计算实现自动曝光比，在正常场景时设置长、短帧曝光比为 1:1，如此 WDR 融合时只采用一帧数据，可以避免正常场景出现运动拖尾，同时也可减弱正常室内场景的工频闪现象。

表2-5 AE_SENSOR_DEFAULT_S 的成员变量

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}。
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。使用海思 AE 算法时，回调 cmsos_fps_set 函数时必须给该值赋值，用于返回实际生效的帧率。
u32InitExposure	默认初始曝光量，等于曝光时间(行数)*系统增益(6bit 小数精度)。海思 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。若不设置，该值默认为 0。
u32InitAESpeed	默认初始 AE 调节速度，海思 AE 算法采用该值作为初始



成员名称	描述
	100 帧的调节速度，可用于运动 DV 加速启动。建议关注快速启动的产品形态可将该值配置为 128。若不设置，该值默认为 0。AE 算法内部会将该值钳位至[64, 255]。
u32InitAETolerance	默认初始 AE 曝光容忍偏差，海思 AE 算法采用该值作为初始 100 帧的曝光容忍偏差值，可用于设置得到首次 AE 收敛稳定标志的亮度范围。AE 统计亮度第一次落在[目标亮度-u32InitAETolerance, 目标亮度+u32InitAETolerance]范围内时，得到首次 AE 收敛稳定标志。若不设置，该值默认为 0。若在 cmos.c 不对该值赋值或赋值为 0，则 AE 算法内部将其置为 u8AeCompensation/10。AE 算法内部会将该值钳位至(0, 255]。
u32FullLinesStd	基准帧率时 1 帧的有效行数。
u32FullLinesMax	sensor 降帧后 1 帧所能达到的最大行数，一般设为 sensor 所支持的最大行数。
u32FullLines	sensor 每帧实际生效的总行数。使用海思 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。
u32MaxIntTime	最大曝光时间，以行为单位。
u32MinIntTime	最小曝光时间，以行为单位。
u32MaxIntTimeTarget	最大曝光时间目标值，以行为单位。
u32MinIntTimeTarget	最小曝光时间目标值，以行为单位。
stIntTimeAccu	曝光时间精度。
u32MaxAgain	最大模拟增益，以倍为单位。
u32MinAgain	最小模拟增益，以倍为单位。
u32MaxAgainTarget	最大模拟增益目标值，以倍为单位。
u32MinAgainTarget	最小模拟增益目标值，以倍为单位。
stAgainAccu	模拟增益精度。
u32MaxDgain	最大数字增益，以倍为单位。
u32MinDgain	最小数字增益，以倍为单位。
u32MaxDgainTarget	最大数字增益目标值，以倍为单位。
u32MinDgainTarget	最小数字增益目标值，以倍为单位。
stDgainAccu	数字增益精度。
u32MaxISPDgainTarget	最大 ISP 数字增益目标值，以倍为单位。



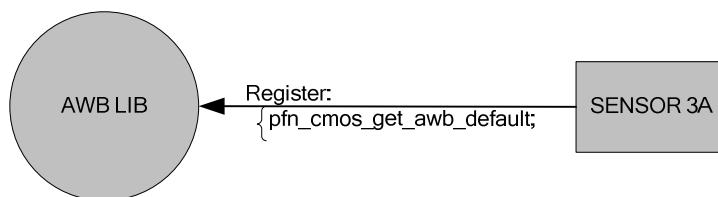
成员名称	描述
u32MinISPDgainTarget	最小 ISP 数字增益目标值，以倍为单位。
u32ISPDgainShift	ISP 数字增益精度。
stAERouteAttr	AE 默认分配路线。
bAERouteExValid	AE 扩展分配路线是否生效开关，该值为 HI_TRUE 时使用扩展分配路线，否则使用普通 AE 分配路线。
stAERouteAttrEx	AE 默认扩展分配路线，合理配置可用于优化 WDR 模式图像效果。
u16ManRatioEnable	两帧合成 WDR 手动曝光比使能，该值为 HI_TRUE 时，曝光比固定，不会根据场景自动更新。
u32Ratio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

2.2.2.2 Sensor 注册 AWB 算法库

Sensor 注册 AWB 算法库调用 HI_MPI_AWB_SensorRegCallBack，如图 2-5 所示，详细说明参见《HiISP 开发参考》。



图2-5 Sensor 向 AWB 库注册的回调函数



【举例】

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

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

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

需要在 xxx_cmos.c 中实现以下回调函数：

表2-6 Sensor 向 AWB 库注册的回调函数

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



说明

如果回调函数暂不实现，可以实现空函数，或者将回调函数指针置为 NULL 即可。

表2-7 AWB_SENSOR_DEFAULT_S 的成员变量

成员名称	子成员名称	描述
u16WbRefTemp	无	静态白平衡校正色温，取值范围为 [0,0xFFFF]。



成员名称	子成员名称	描述
au16GainOffset	无	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	无	校正工具给出的白平衡参数，取值范围为[0,0xFFFFFFFF]。
stAgcTbl	bValid	该结构体的数据是否有效，取值范围为[0,1]。
	au8Saturation	根据增益动态调节饱和度的插值数组，取值范围为[0,255]。
stCcm	u16HighColorTemp	高色温，取值范围为[0,0xFFFF]。
	au16HighCCM	高色温颜色还原矩阵，取值范围为[0,0xFFFF]。
	u16MidColorTemp	中色温，取值范围为[0,0xFFFF]。
	au16MidCCM	中色温颜色还原矩阵，取值范围为[0,0xFFFF]。
	u16LowColorTemp	低色温，取值范围为[0,0xFFFF]。
	au16LowCCM	低色温颜色还原矩阵，取值范围为[0,0xFFFF]。

2.2.2.3 Sensor 注册 AF 算法库

Sensor 注册 AF 算法库调用 HI_MPI_AF_SensorRegCallBack，AF 库暂未实现。



3 开发者指南

3.1 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。

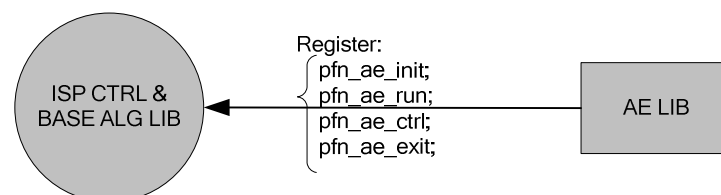
Sensor 注册到 ISP 库，以实现差异化适配 Firmware 中的基础算法单元的部分，仍需要参考使用者指南部分的内容实现，以保证 Firmware 中的坏点校正、去噪、色彩增强、镜头阴影校正等处理算法正常运行。Sensor 注册到 3A 算法库的部分，请用户根据自己开发定制的 3A 库，自行定义数据结构，实现 Sensor 曝光控制等。

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。

3.2 AE 算法注册 ISP 库

AE 算法注册 ISP 库调用 HI_MPI_ISP_AeLibRegCallBack，如图 3-1 所示，详细说明参见《HiISP 开发参考》。

图3-1 AE 算法向 ISP 库注册的回调函数



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3-1 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 命令有：

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

调用 HI_MPI_ISP_Exit 时将调用 pfn_ae_exit 回调函数，以销毁 AE 算法库。



表3-2 初始化参数 ISP_AE_PARAM_S 的成员变量

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

表3-3 统计信息 ISP_AE_INFO_S 的成员变量

成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAeStat1	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au16MeteringHist	五段直方图的统计信息数组，取值范围为 [0, 0xFFFF], au16MeteringHist[0]表示第一段，au16MeteringHist[1]表示第二段，au16MeteringHist[2]表示第四段，au16MeteringHist[3]表示第五段。其中第三段统计信息值为 0xFFFF-(au16MeteringHist[0]+au16MeteringHist[1]+au16MeteringHist[2]+au16MeteringHist[3])
pstAeStat2	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au16MeteringMemArray	五段直方图的分区间的统计信息数组，取值范围为[0, 0xFFFF]。
pstAeStat3	au32HistogramMemArray	256 段直方图的统计信息数组，取值范围为[0, 0xFFFFFFFF]。
pstAeStat4	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]。
pstAeStat5	au16ZoneAvg	分区间 R、Gr、Gb、B 分量平均值，取值范围为[0, 0xFFFF]。



说明

pstAeStat1 和 pstAeStat2 分别表示根据直方图分割门限得到的归一化全局和分区间 5 段直方图统计信息，取值范围[0, 0xFFFF]。以全局 5 段直方图为例，如图像中的所有像素都大于最大门限值，那么第 5 段直方图数据即为 0xFFFF，其他 4 段直方图数据为 0。全局 5 段直方图会受分区间权重影响，与该模块在 ISP pipeline 的位置无关。

pstAeStat3 表示全局 256 段直方图统计信息。该统计信息是取输入数据流中的高 8bit 数据统计得到的，每个 bin 中数据表示该灰度值对应的像素个数。256 个 bin 的数据之和即为参与统计的像素点个数，由寄存器 0x205a06c0 决定，如下图所示。一旦寄存器 0x205a06c0 的值是确定的，那么 256 个 bin 的数据之和也是确定的。目前，海思 AE 算法默认只用了 Gr 通道的统计信息，在大面积红色时，会采用 R 和 Gb 通道的统计信息，在大面积蓝色时，会采用 B 和 Gr 通道的统计信息。将该模块在 ISP pipeline 的位置与 5 段直方图统计模块的位置配成一样，则 256 段直方图会受到分区间权重的影响。

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

skip x [2:0]

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

offset x

0= start from the first column; 1=start from second column

skip y [2:0]

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

offset y

0= start from the first row; 1= start from second row

pstAeStat4 表示全局 R/Gr/Gb/B 4 分量的归一化均值，取值范围[0, 0xFFFF]。若统计数据是 12bit，0xFFFF 即表示某分量的均值为 12bit 的最大值 4095。全局 4 分量平均值会受分区间权重影响，与该模块在 ISP pipeline 的位置无关。

pstAeStat5 表示 15*17 区间每个区间 R/Gr/Gb/B 4 分量的归一化均值，取值范围[0, 0xFFFF]，意义与全局 4 分量平均值相同。

在 WDR 模式下，若把上述统计模块在 ISP pipeline 的位置配置在 GammaFE 之后，那么统计的是开过方的数据。所谓数据开方，指的是对输入数据归一化至 1 后做开方处理。以 256 段直方图统计为例，若输入数据为 12bit，某个像素值为 2048，线性模式下取高 8bit 数据进行统计，为灰度值 128 对应的 bin 像素个数加 1；WDR 模式下，像素值 2048 归一化至 1 后为 0.5，0.5 开方为 0.707，用 8bit 表示 0.707 为 181，此时在直方图上表现为灰度值 181 对应的 bin 像素个数加 1。由此可见，像素值较小的数据开方处理后像素值明显变大，相当于 WDR 模式下统计信息通过压缩亮区的统计精度来提升了暗区的统计精度。如何改变上述模块在 ISP pipeline 的位置，可参考《HiISP 开发参考》。在 WDR 模式下，假设曝光比为 N，若 AE 统计模块输入数据未开方，则全局 256 段直方图前面 $256/N$ 个 bin 为长帧的统计信息，剩下的为短帧统计信息。若 AE 统计模块输入数据经过了开方处理，则全局 256 段直方图前面 $256/\sqrt{N}$ 个 bin 为长帧的统计信息，剩下的为短帧统计信息。可见在曝光比相同的情况下，开方处理后直方图用了更多的 bin 来表示长帧。由于暗区一般采用的是长帧数据，这与前一段分析开方处理后 WDR 模式下统计信息通过压缩亮区的统计精度来提升了暗区的统计精度结论是一致的。



表3-4 运算结果 ISP_AE_RESULT_S 的成员变量

成员名称	子成员名称	描述
au32IntTime	-	AE 计算得出的曝光时间，以 us 为单位，将曝光时间由行数转换成 us 时需要考虑 cmos.c 中 f32Offset 的影响。线性模式和 sensor built-in WDR 模式下，只有 au32IntTime[0]有效，au32IntTime[1:3]建议配置等于 au32IntTime[0]；N 帧合成 WDR 模式下，au32IntTime[0:(N-1)]有效，配置值由小到大，依次表示最短到最长的曝光时间，用于计算长短帧曝光比，au32IntTime[(N-1):3]建议配置等于 au32IntTime[(N-1)]。au32IntTime[0]还需传递至其他模块用于与曝光时间相关的联动控制，会影响海思 AWB 效果。使用海思 AWB 算法和多帧 WDR 模式时必须配置该结构体。
u32IspDgain	-	ISP 的数字增益，使用 ISP 数字增益时必须配置；不使用时配置为 0x100。
u32Iso	-	AE 计算得出的总增益值，ISO 表示系统增益，以常数 100 乘以倍数为单位,例如系统中 sensor 的增益为 2 倍，ISP 的增益为 1 倍，那么整个系统的 ISO 值计算方式为：2*1*100=200，即系统 ISO 为 200，本文档中涉及到的 ISO 都是采用这种计算方法。此变量影响 ISP 的去噪，sharpen 等自适应效果，必须配置。
u8AERunInterval	-	AE 算法运行的间隔，取值范围为[1,255]，取值为 1 时表示每帧都运行 AE 算法，取值为 2 时表示每 2 帧运行 1 次 AE 算法，依此类推。建议该值设置不要大于 2，否则 AE 调节速度会受到影响。WDR 模式时，该值建议设置为 1，这样 AE 收敛会更加平滑，此变量决定 AE 计算结果配置到 sensor 和 ISP 寄存器的间隔帧数，必须配置。
bPirisValid	-	Piris 是否有效的标志，取值为 HI_TRUE 时在内核态回调 Piris 驱动配置步进电机的位置，取值为 HI_FALSE 时不回调。使用海思 AE 算法和海思 Piris 驱动对接 Piris 镜头时，该值须置为 HI_TRUE，对接非 Piris 镜头时该值须置为 HI_FALSE。
s32PirisPos	-	Piris 步进电机的位置，取值范围与具体 Piris 镜头相关。使用海思 Piris 驱动对接 Piris 镜头时，该值必须配置。
u32PirisGain	-	Piris 光圈等效增益，取值范围与具体 Piris 镜头相关。可用于计算 Piris 工作时的等效曝光量，供其他模块参考。取值范围为[0, 1024]。客户自行开发 AE 算法对接非 Piris 镜头时可将该值配置为 0。



成员名称	子成员名称	描述
enFSWDRMode	-	FSWDR 运行模式，目前包括正常 WDR 模式和长帧模式，供其他模块参考，长帧模式使能时可另行配置优化参数。客户自行开发 AE 算法，若不支持长帧模式，可将该值配置为 ISP_FSWDR_NORMAL_MODE，若需要 FSWDR 模块只输出长帧数据，可将该值配置为 ISP_FSWDR_LONG_FRAME_MODE。注意，长帧模式仅在行模式 WDR 下才有效。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au8WeightTable	15x17 个区间的 AE 权重表，取值范围为[0, 255]。



说明

将曝光时间 au32IntTime 由行转换成 us 时，可以通过 cmos.c 中的 u32LinesPer500ms 进行，转换关系如下：
$$\text{au32IntTime}[0] = (((\text{HI_U64})\text{au32IntTimeRst}[0] * 1024 - \text{u32Offset}) * 500000 / \text{pstAeSnsDft} \rightarrow \text{u32LinesPer500ms}) \gg 10$$

上式中 au32IntTimeRst[0]是以行数为单位的曝光时间，u32Offset=f32Offset * 1024，f32Offset 即为曝光时间的偏移量，详见《HiISP 开发参考》中 AE_ACCURACY_S 部分描述。

2 合 1 WDR 模式时，若 au32IntTime[0]等于 au32IntTime[1]，即长/短帧曝光比为 1:1 时，不做合成，直接采用长帧数据作为输出。

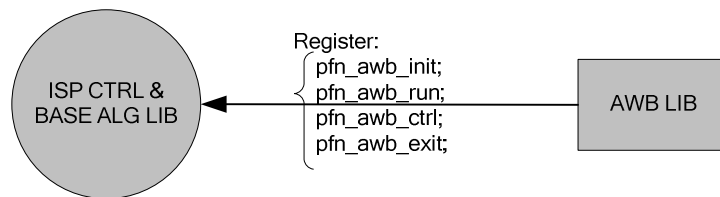
FSWDR 模式下，enFSWDRMode 为 ISP_FSWDR_LONG_FRAME_MODE 时，AE 将短帧曝光时间设置为最小值，长帧曝光时间接近 1 帧所允许的最大值，合成模块只输出长帧数据，可用于优化弱宽动态场景或低照度时的图像质量。注意：长帧模式只在行模式 WDR 有效，因为帧模式 WDR 长帧曝光时间本来就可以达到 1 帧所允许的最大值。工作于长帧模式时，只需要在 cmos.c 回调函数 cmos_get_inttime_max 中给 au32IntTimeMax[0]和 au32IntTimeMin[0]赋值，短帧采用 au32IntTimeMin[0]作为曝光时间。默认为 ISP_FSWDR_NORMAL_MODE。

3.3 AWB 算法注册 ISP 库

AWB 算法注册 ISP 库调用 HI_MPI_ISP_AwbLibRegCallBack，如图 3-2 所示，详细说明参见《HiISP 开发参考》。



图3-2 AWB 算法向 ISP 库注册的回调函数



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3-5 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。设置 ISO 的目的是为了实现 ISO 与饱和度的联动，增益大时色度噪声也会比较大，所以需要调节饱和度。

当前 Firmware 定义的 ctrl 命令有：

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

调用 HI_MPI_ISP_Exit 时将调用 pfn_awb_exit 回调函数，以销毁 AWB 算法库。



表3-6 初始化参数 ISP_AWB_PARAM_S 的成员变量

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

表3-7 统计信息 ISP_AWB_INFO_S 的成员变量

成员名称	子成员名称	描述
u32FrameCnt	-	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat1	u16MeteringAwbRg	RGB 域统计信息中白点 G/R 比值的平均值，取值范围为[0, 0xFFFF]。
	u16MeteringAwbBg	RGB 域统计信息中白点 G/B 比值的平均值，取值范围为[0, 0xFFFF]。
	u32MeteringAwbSum	RGB 域统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat2	au16MeteringMemArrayRg	RGB 域分区间的统计信息中白点 G/R 比值的平均值，取值范围为 [0, 0xFFFF]。
	au16MeteringMemArrayBg	RGB 域分区间的统计信息中白点 G/B 比值的平均值，取值范围为 [0, 0xFFFF]。
	au16MeteringMemArraySum	RGB 域分区间的统计信息中白点个数，取值范围为[0, 0xFFFFFFFF]。
pstAwbStat3	u16MeteringAwbAvgR	Bayer 域全局统计信息中白点的 R 分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	u16MeteringAwbAvgG	Bayer 域全局统计信息中白点的 G 分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	u16MeteringAwbAvgB	Bayer 域全局统计信息中白点的 B



成员名称	子成员名称	描述
		分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	u16MeteringAwbCountAll	Bayer 域全局统计信息中白点的个数。已做归一化，取值范围为[0, 0xFFFF]。
	u16MeteringAwbCountMin	Bayer 域全局统计信息中小于 u16MeteringBlackLevelAwb 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。
	u16MeteringAwbCountMax	Bayer 域全局统计信息中大于 u16MeteringWhiteLevelAwb 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。
pstAwbStat4	au16MeteringMemArrayAvgR	Bayer 域分区间统计信息中白点的 R 分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	au16MeteringMemArrayAvgG	Bayer 域分区间统计信息中白点的 G 分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	au16MeteringMemArrayAvgB	Bayer 域分区间统计信息中白点的 B 分量平均值，已减掉黑电平。取值范围为[0, 0xFFFF]，4bit 小数精度。
	au16MeteringMemArrayCountAll	Bayer 域分区间统计信息中白点的个数。已做归一化，取值范围为[0, 0xFFFF]。
	au16MeteringMemArrayCountMin	Bayer 域分区间统计信息中小于 u16MeteringBlackLevelAwb 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。
	au16MeteringMemArrayCountMax	Bayer 域分区间统计信息中大于 u16MeteringWhiteLevelAwb 的像素个数。已做归一化，取值范围为[0, 0xFFFF]。



表3-8 运算结果 ISP_AWB_RESULT_S 的成员变量

成员名称	子成员名称	描述
au32WhiteBalanceGain	-	白平衡算法得出的 R、Gr、Gb、B 颜色通道的增益，16bit 精度表示。
au16ColorMatrix	-	色彩还原矩阵，8bit 精度表示。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	u16MeteringWhiteLevelAwb	RGB 域统计白点信息时，找白点的亮度上限。取值范围 [0x0, 0x3FF]，默认值 0x3ac。
	u16MeteringBlackLevelAwb	RGB 域统计白点信息时，找白点的亮度下限。取值范围 [0x0, 0x3FF]，默认值 0x40。
	u16MeteringCrRefMaxAwb	RGB 域统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 0x200。
	u16MeteringCbRefMaxAwb	RGB 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 0x200。
	u16MeteringCrRefMinAwb	RGB 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 0x80。
	u16MeteringCbRefMinAwb	RGB 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 0x80。
	u16MeteringCrRefHighAwb	RGB 域统计白点信息时，六边形白点区域限制 CbMax 对应的 Cr 值，8bit 精度，默认值 0x200。
	u16MeteringCrRefLowAwb	RGB 域统计白点信息时，六边形白点区域限制 CbMin 对应的 Cr 值，8bit 精度，默认值 0x80。
	u16MeteringCbRefHighAwb	RGB 域统计白点信息时，六边形白点区域限制 CrMax 对应的 Cb 值，8bit 精度，默认值 0x200。



成员名称	子成员名称	描述
	u16MeteringCbRefLowAwb	RGB 域统计白点信息时，六边形白点区域限制 CrMin 对应的 Cb 值，8bit 精度，默认值 0x80。
stRawStatAttr	bChange	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 0x180。
	u16MeteringCbRefMaxAwb	Bayer 域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 0x180。
	u16MeteringCrRefMinAwb	Bayer 域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 0x40。
	u16MeteringCbRefMinAwb	Bayer 域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 0x40。
	u16MeteringCrRefHighAwb	Bayer 域统计白点信息时，六边形白点区域限制 CbMax 对应的 Cr 值，8bit 精度，默认值 0x180。
	u16MeteringCrRefLowAwb	Bayer 域统计白点信息时，六边形白点区域限制 CbMin 对应的 Cr 值，8bit 精度，默认值 0x40。
	u16MeteringCbRefHighAwb	Bayer 域统计白点信息时，六边形白点区域限制 CrMax 对应的 Cb 值，8bit 精度，默认值 0x180。
	u16MeteringCbRefLowAwb	Bayer 域统计白点信息时，六边形白点区域限制 CrMin 对应的 Cb 值，8bit 精度，默认

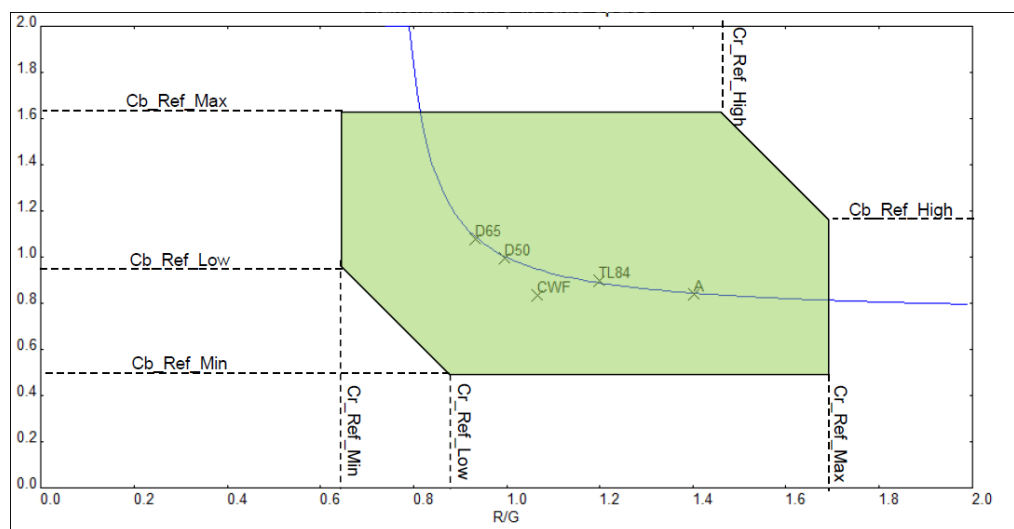


成员名称	子成员名称	描述
		值 0x40。

说明

- Bayer 域 WB 统计信息在 WB 增益前。RGB 域 WB 统计信息在 WB 增益后，利用 RGB 域统计信息实现 AWB 算法，AWB 是个反馈系统，RGB 域统计结果受 WB 增益影响。
- RGB 域统计结果 $u16MeteringAwbRg$ 是 G/R，统计参数 $u16MeteringCrRefMaxAwb$ 是 R/G。
- 统计信息异常：为了避免统计信息异常对算法的影响，建议白点个数满足一定条件的统计信息参与计算。RGB 域白点个数限制为 0x40，Bayer 域白点个数可不作限制。
- AWB 参数生效时间点：下一帧生效。
- 建议 AWB 统计参数根据 sensor 的光谱特性和产品支持的色温范围做调整，以达到更好的效果。
- Sensor WDR 模式下，PipeLine 的数据做了开方处理。用户通过 MPI 配置 AWB 统计参数 (CrMax 等)，需要先做开方处理；通过 MPI 获取 AWB 统计结果，需要做平方处理。
- 用户通过 ISP_AWB_RESULT_S 结构体配置 AWB 统计参数 ($u16MeteringCrRefMaxAwb$)，ISP 自动对参数做开方处理，用户不需要额外处理。用户通过 ISP_AWB_INFO_S 获取 AWB 统计结果，需要做平方处理。处理过程请参考示例代码。
- HI_MPI_ISP_GetStatistics 接口提供的 Bayer 域 AWB 统计结果，已减掉黑电平，但 WDR 模式下，未做解压缩处理。
- AWB 通常设置 G 通道增益为 1，R、B 通道大于 1。这样，G 通道数据减黑电平后，G 通道不饱和，而 R、B 通道饱和，出现亮区粉红色。因此，需要用户将 R、G、B 三通道的最小增益归一化为 $4095/(4095\text{-黑电平})$ 。

图3-3 统计信息参数示意图





【举例】

WDR 模式下, AWB 统计参数和结果的处理示例。以 MPI 接口为例, ISP_AWB_INFO_S 结构体的 AWB 统计结果处理方式与 MPI 统计结果处理方式相同。

```
ISP_DEV IspDev;
ISP_STATISTICS_CFG_S stStatCfg;
ISP_STATISTICS_S stStat;
HI_MPI_ISP_GetStatisticsConfig(IspDev, &stStatCfg);

/*Hi3516A platform, in WDR mode, user should configure awb parameter's
square root, because the pipeline data is non-linear. */

/*WhiteLevel is 10 bits valid. For example, in linear mode, user
configure WhiteLevel = 0x3AC, in wdr mode, user should configure square
root of 0x3AC*/
stStatCfg.stWBCfg.stRGCfg.ul6WhiteLevel = (HI_U16)(sqrt(0x3AC << 10));
/*BlackLevel is 10 bits valid. For example, in linear mode, user
configure BlackLevel = 0x40, in wdr mode, user should configure square
root of 0x40*/
stStatCfg.stWBCfg.stRGCfg.ul6BlackLevel = (HI_U16)(sqrt(0x40 << 10));
/*Cr Cb are 4.8-bit fixed-point*/
stStatCfg.stWBCfg.stRGCfg.ul6CrMax = (HI_U16)(sqrt(0x200 << 8));
stStatCfg.stWBCfg.stRGCfg.ul6CrMin = (HI_U16)(sqrt(0x80 << 8));
stStatCfg.stWBCfg.stRGCfg.ul6CbMax = (HI_U16)(sqrt(0x200 << 8));
stStatCfg.stWBCfg.stRGCfg.ul6CbMin = (HI_U16)(sqrt(0x80 << 8));
/*WhiteLevel is 12 bits valid. For example, in linear mode, user
configure WhiteLevel = 0xF00, in wdr mode, user should configure square
root of 0xF00 */
stStatCfg.stWBCfg.stBayerCfg.ul6WhiteLevel = (HI_U16)(sqrt(0xF00 <<
12));

/*BlackLevel is 12 bits valid, Warning: The min value of ul6BlackLevel
is sensor offset.
For example, in linear mode, user configure BlackLevel = 0x100, in wdr
mode, user should configure square root of 0x100*/
stStatCfg.stWBCfg.stBayerCfg.ul6BlackLevel = (HI_U16)(sqrt(0x100 <<
12));

/*Cr Cb are 4.8-bit fixed-point*/
stStatCfg.stWBCfg.stBayerCfg.ul6CrMax = (HI_U16)(sqrt(0x130 << 8));
stStatCfg.stWBCfg.stBayerCfg.ul6CrMin = (HI_U16)(sqrt(0x40 << 8));
stStatCfg.stWBCfg.stBayerCfg.ul6CbMax = (HI_U16)(sqrt(0x120 << 8));
stStatCfg.stWBCfg.stBayerCfg.ul6CbMin = (HI_U16)(sqrt(0x40 << 8));
```



```
HI_MPI_ISP_SetStatisticsConfig(IspDev, &stStatCfg);

/*Get AWB statistics*/
HI_MPI_ISP_GetStatistics(IspDev, &stStat);
/*RGB domain statistics, pow 2 to get linear statistics*/
stStat.stWBStat.stRGBStatistics.ul6GlobalGR =
POW2((HI_U32)stStat.stWBStat.stRGBStatistics.ul6GlobalGR) >> 8;
stStat.stWBStat.stRGBStatistics.ul6GlobalGB =
POW2((HI_U32)stStat.stWBStat.stRGBStatistics.ul6GlobalGB) >> 8;
for(i = 0; i < AWB_ZONE_ROW; i++)
{
    for(j = 0; j < AWB_ZONE_COLUMN; j++)
    {
        stStat.stWBStat.stRGBStatistics.au16ZoneGR[i][j] =
POW2((HI_U32)stStat.stWBStat.stRGBStatistics.au16ZoneGR[i][j]) >> 8;
        stStat.stWBStat.stRGBStatistics.au16ZoneGB[i][j] =
POW2((HI_U32)stStat.stWBStat.stRGBStatistics.au16ZoneGB[i][j]) >> 8;
    }
}
/*Bayer domain statistics, pow2 to get linear data. RGB average value
are 16 bit valid*/
stStat.stWBStat.stBayerStatistics.ul6GlobalR =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.ul6GlobalR) >> 16;
stStat.stWBStat.stBayerStatistics.ul6GlobalG =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.ul6GlobalG) >> 16;
stStat.stWBStat.stBayerStatistics.ul6GlobalB =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.ul6GlobalB) >> 16;
for(i = 0; i < AWB_ZONE_ROW; i++)
{
    for(j = 0; j < AWB_ZONE_COLUMN; j++)
    {
        stStat.stWBStat.stBayerStatistics.au16ZoneAvgR[i][j] =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.au16ZoneAvgR[i][j]) >> 16;
        stStat.stWBStat.stBayerStatistics.au16ZoneAvgG[i][j] =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.au16ZoneAvgG[i][j]) >> 16;
        stStat.stWBStat.stBayerStatistics.au16ZoneAvgB[i][j] =
POW2((HI_U32)stStat.stWBStat.stBayerStatistics.au16ZoneAvgB[i][j]) >> 16;
    }
}
```

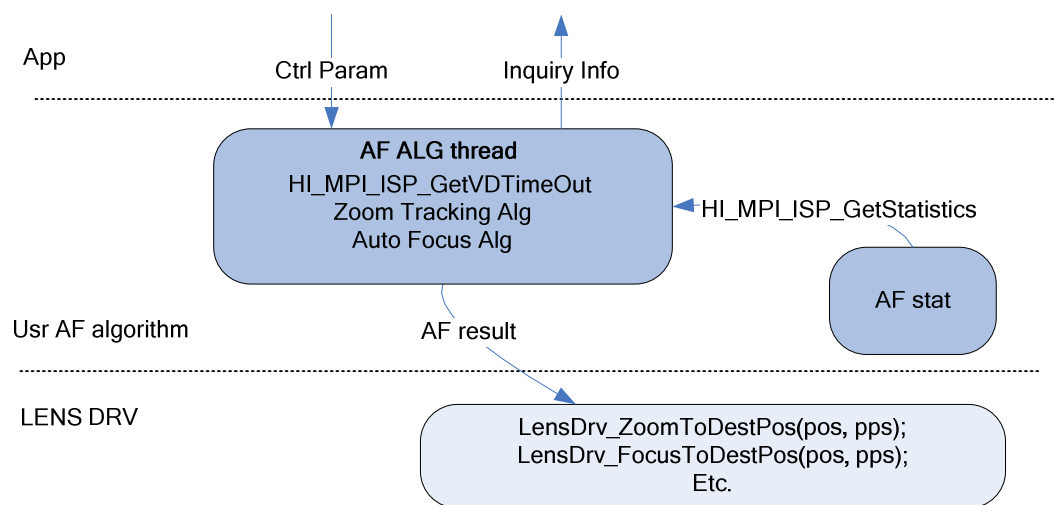


3.4 开发用户 AF 算法

3.4.1 AF 算法结构

用户开发自己的 AF 算法可以参考图 3-4 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，方面用户完成一键聚焦，变焦，手动聚焦，查询算法状态等操作。

图3-4 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 3.4.2 章节。

```
while (1)
{
    s32Ret = HI_MPI_ISP_GetVDTimeOut(IspDev, &stVdInfo, 5000);
    s32Ret |= HI_MPI_ISP_GetStatistics(IspDev, &stIspStatics);

    if (HI_SUCCESS != s32Ret)
    {
        printf("HI_MPI_ISP_GetStatistics error!(s32Ret = 0x%x)\n", s32Ret);
        return HI_FAILURE;
    }

    // User Auto Focus Alg Source
    UserAfAlgRun();
}
```



```
// Update Lens Position
LensDrv_ZoomToDestPos(pos, pps);
LensDrv_FocusToDestPos(pos, pps);
}
```

3.4.2 AF 同步回调

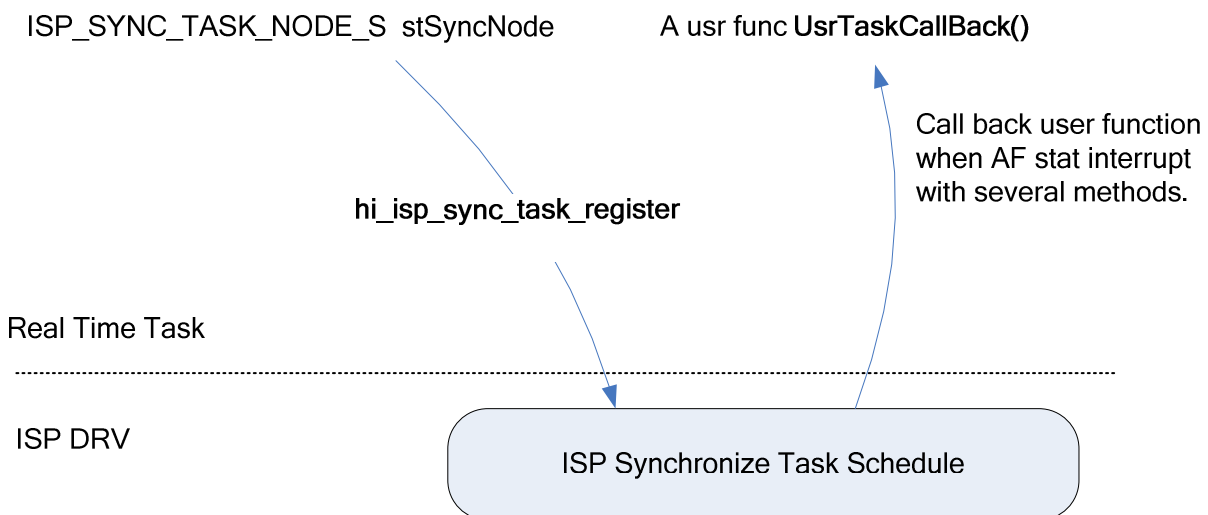
当一帧图像最后一个 pixel 通过 AF 模块后，统计值即更新，推荐用户通过 HI_MPI_ISP_GetVDTimeOut 同步获取统计值，AutoFocus 和 ZoomTracking 算法完成目标 focus 和 zoom position 的计算，如果用户使用的是 LINUX 系统，因为 user space 任务调度不能保证一致的实时性，建议将需要保证实时性的驱动配置放在 kernel space 完成。ISP 提供同步回调接口的注册，可以实现与 VD 同步。在 extdrv/sample_ist 有相应的 sample，用户可以将实时性要求较高的任务放在同步回调里面，底层提供 HwIRQ，Tasklet，Workqueue 三种方式实现，可以选择相应的实现方式以确定实时级别。



说明

HI_MPI_ISP_GetVDTimeOut 请参考《HiISP 开发参考》。

图3-5 AF 同步回调



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(ISP_DEV 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(ISP_DEV 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](#)

数据类型

- [ISP_SYNC_TSK_METHOD_E](#): 定义同步回调方法，决定实时性。
- [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_TSKLET,
    ISP_SYNC_TSK_METHOD_WORKQUE,
    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_TSKLET	使用 TaskLet 方式回调。
ISP_SYNC_TSK_METHOD_WORKQUE	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_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_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。

【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](#)



4 附录

4.1 注册函数的关系

HI_MPI_ISP_AeLibRegCallBack、HI_MPI_ISP_AwbLibRegCallBack、HI_MPI_ISP_AfLibRegCallBack 这三个接口是 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。例如海思提供的 3A 算法库的 HI_MPI_AE_Register、HI_MPI_AWB_Register、HI_MPI_AF_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，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、HI_MPI_AWB_SensorRegCallBack、HI_MPI_AF_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、HI_MPI_AF_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_XXX_Register 接口。同时也请自行实现 HI_MPI_XXX_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。

4.2 扩展性的设计考虑

在代码中有 ISP_DEV、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。

ISP_DEV 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。目前 ISP_DEV 只需设置为 0 即可。



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。

4.3 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/AF 的 MPI 不可复用，cmos.c 中的 AE/AWB/AF 相关的内容不可复用，对于 AE 的权重配置、五段直方图 Thresh 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、ISP_AWB_RESULT_S、ISP_AF_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、ISP_AWB_INFO_S、ISP_AF_INFO_S 结构体中读取即可。

4.4 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。

外部寄存器封装的接口为 VReg_Init、VReg_Exit、IORD_32DIRECT、IORD_16DIRECT、IORD_8DIRECT、IOWR_32DIRECT、IOWR_16DIRECT、IOWR_8DIRECT，地址设定如下：

0x0~0xFFFF 对应 ISP 的硬件寄存器，例如 IORD_32DIRECT(0x0008)读取 0x205A0008 硬件寄存器的值。

0x10000~0x1FFFF 对应 ISP firmware 的外部寄存器，例如 IOWR_16DIRECT(0x10020)。



0x20000~0x2FFFF 对应 AE 的外部寄存器，其中分成了 16 份，海思 AE 用了 0x20000~0x21FFF。0x30000~0x3FFFF 对应 AWB 的外部寄存器，0x40000~0x4FFFF 对应 AF 的外部寄存器，同样的也分成了 16 份。

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 hi_vreg.h 文件中。