



Hi3518_ISP_3A 版本

使用指南

文档版本 02

发布日期 2013-09-25

版权所有 © 深圳市海思半导体有限公司 2013。保留一切权利。

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

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前 言

概述

本文档描述 Hi3518_ISP_3A 的功能、如何使用与开发。3A 功能包括 AE、AWB、AF。

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
Hi3518_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。
 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。



符号	说明
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

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

文档版本 02 (2013-09-25)

第 2 章 使用者指南

2.1 软件流程中增加说明。

第 3 章 开发者指南

3.2 AE 算法注册 ISP 库的【举例】中增加注释。

第 4 章 附录

新增。

文档版本 01 (2013-07-16)

第 1 次正式发布。



目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 功能描述.....	1
1.2.1 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.....	6
2.1 软件流程.....	6
2.2 Sensor 对接.....	10
2.2.1 Sensor 注册 ISP 库.....	10
2.2.2 Sensor 注册 3A 算法库.....	14
3 开发者指南.....	19
3.1 概述.....	19
3.2 AE 算法注册 ISP 库.....	19
3.3 AWB 算法注册 ISP 库	22
3.4 AF 算法注册 ISP 库	24



1 概述

1.1 概述

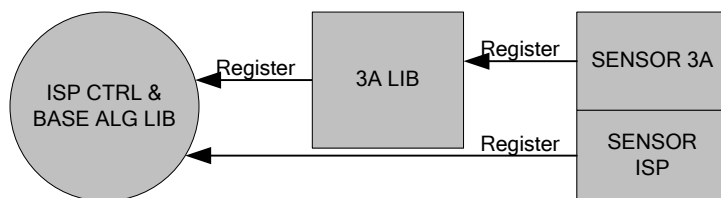
Hi3518_ISP_3A 版本依赖于相应的 SDK 大版本，通过一系列数字图像处理算法完成对数字图像的效果处理。主要包含 Firmware 框架及海思 3A 库，Firmware 提供算法的基本框架，处理统计信息，驱动数字图像处理算法，并包含坏点校正、去噪、色彩增强、镜头阴影校正等处理。3A 库以注册的方式，添加到 Firmware 中，完成曝光、白平衡、色彩还原等处理。

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



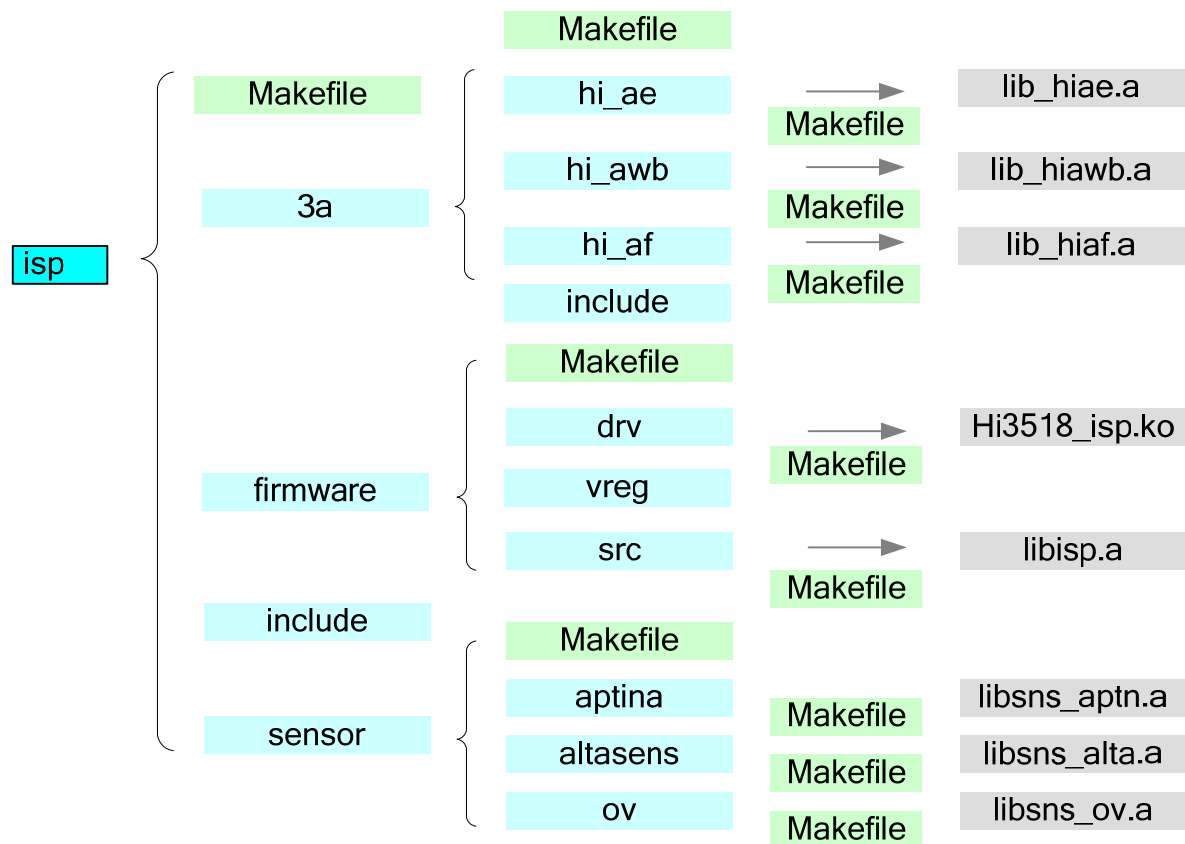
不同的 sensor 都向 ISP 库和 3A 算法库注册控制函数，这些函数都以回调函数的形式存在。ISP 控制单元调度基础算法单元和 3A 算法库时，将通过这些回调函数获取初始化参数，并控制 sensor，如调节曝光时间、模拟增益、数字增益，控制 lens 步进聚焦或旋转光圈等。



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 库分开。Firmware 中的 drv 生成的驱动程序上报 ISP 中断，并以该中断驱动 Firmware 的 ISP 控制单元运转，ISP 控制单元将从驱动程序中获取统计信息，并调度基础算法单元和 3A 算法库，最后将需要配置的寄存器信息告知驱动程序。Src 文件夹中包含 ISP 控制单元和基础算法单元，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，这里将其编译成库的形式，方便应用程序编译和连接，当然，用户可以根据自己的需要多样化处理。

图1-2 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。



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。

1.2.3.3 使用差异说明

Hi3518 的 SDK 之前提供的是 3A 没有剥离的 ISP 版本，当用户切换到 3A 剥离的 ISP 版本时，可能有以下差异项需要注意：

- Makefile 可能需要修改，旧的 ISP 版本仅需要链接 libisp.a 的库，新的 ISP 版本需要链接 libisp.a、lib_hiae.a、lib_hiawb.a、lib_hiaf.a 库。
- 包含的头文件增加，旧的 ISP 版本仅需包含 mpi_isp.h、hi_comm_isp.h、hi_comm_sns.h、hi_sns_ctrl.h 的头文件，新的 ISP 版本需要额外包含 hi_comm_3a.h、hi_ae_comm.h、hi_af_comm.h、hi_awb_comm.h、mpi_ae.h、mpi_af.h、mpi_awb.h 的头文件。
- 初始化流程存在差异，旧的流程需要调用 sensor_init 和 sensor_register_callback，新的流程仅需要调用 sensor_register_callback；另外新的流程需要调用 HI_MPI_AE_Register、HI_MPI_AWB_Register、HI_MPI_AF_Register 接口以注册 AE/AWB/AF 算法库，示例请参考“2.1 软件流程章节”中的示例。
- WDR 和线性模式的切换流程存在差异，旧的流程需要调用 sensor_mode_set、sensor_register_callback、HI_MPI_ISP_Init；新的流程仅需要调用 HI_MPI_ISP_SetWdrAttr。
- 其他 MPI 均可参考旧的 MPI。
- 用户自己调试优化过的 xxx_cmos.c 中的参数，需要合入到新版本的 xxx_cmos.c 中去，详细的数据结构请参考《HiISP 开发参考》或“2.2 Sensor 对接”章节。

1.2.4 内部流程

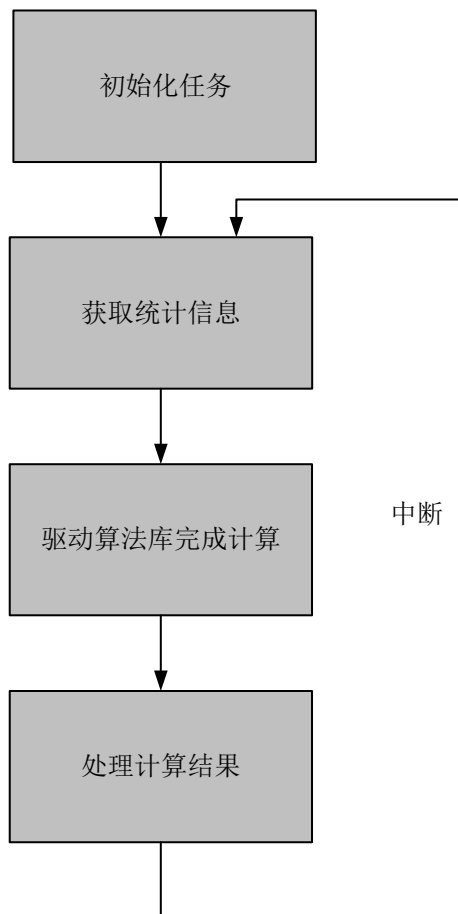
Firmware 内部流程，如图 1-3 所示。首先完成 Firmware 控制单元的初始化、基础算法单元的初始化、3A 算法库的初始化，包括调用 sensor 的回调获取 sensor 差异化的初始化参数。当初始化完成之后，Firmware 由中断驱动，每帧从内核态获取统计信息，并驱动基础算法单元和 3A 算法库完成计算，并反馈计算结果，配置 ISP 寄存器和 sensor 寄存器。

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。

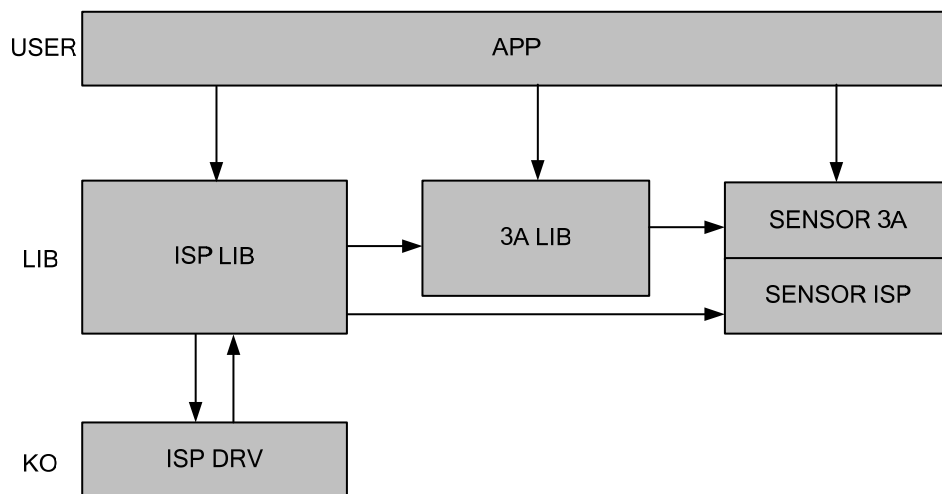
图1-3 ISP firmware 内部流程



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

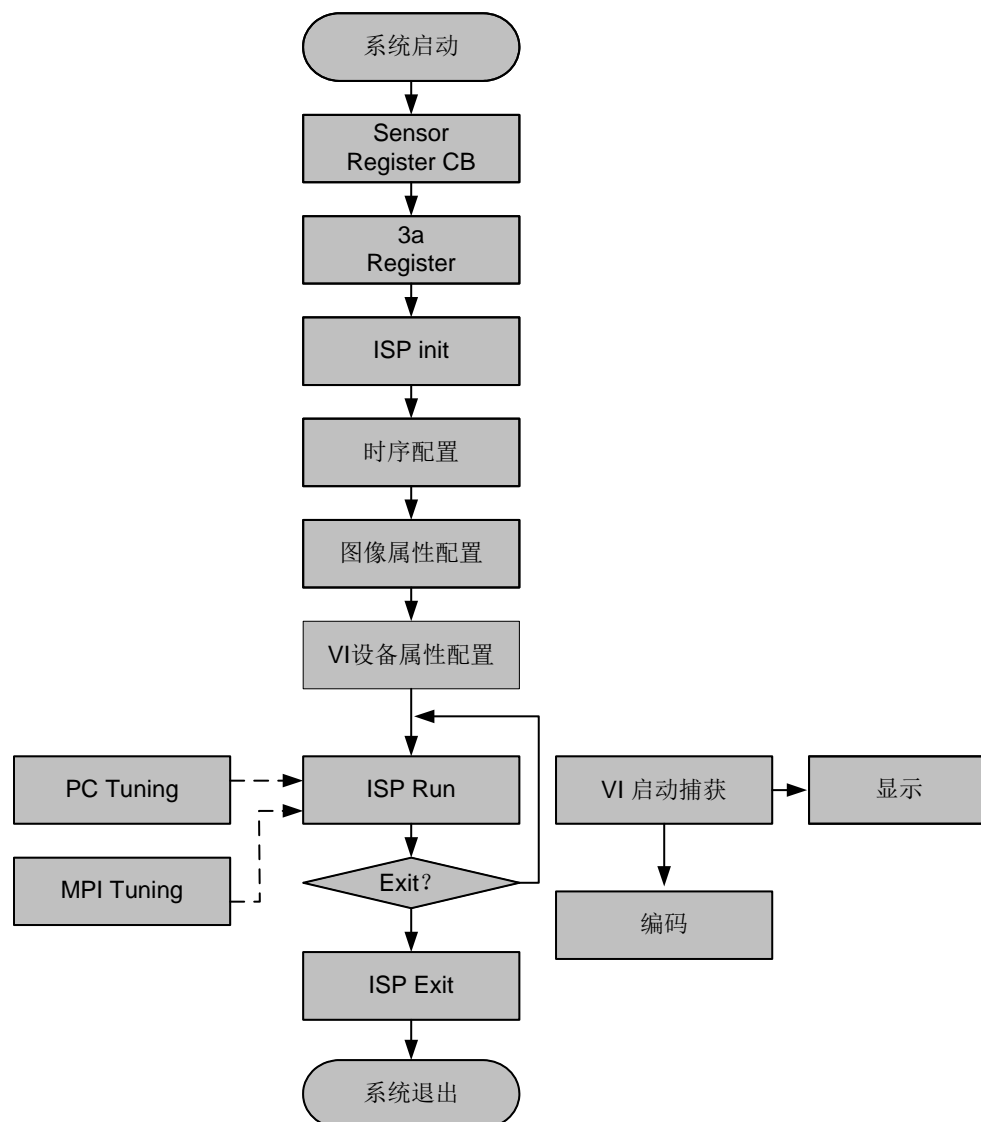
2.1 软件流程

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 采集到 DDR，进而送去显示或编码。软件使用流程如图 2-1 所示。

PC Tuning 主要完成在 PC 端进行动态图像质量调节，可以调节多个影响图像质量的因子，如去噪强度、色彩转换矩阵、饱和度等。如果在产品发布阶段没有 PC Tuning 工具，可以使用 MPI 中提供的图像质量调节接口进行简单的图像效果调试。



图2-1 ISP firmware 使用流程



如果用户调试好图像效果后，可以使用 PC Tuning 工具提供的保存配置文件进行配置参数保存，在下次启动时可以加载已经调节好的图像参数。

代码示例：

```
HI_S32 s32Ret;
ALG_LIB_S stLib;
ISP_IMAGE_ATTR_S stImageAttr;
ISP_INPUT_TIMING_S stInputTiming;
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 Firmware */
    s32Ret = HI_MPI_ISP_Init();
    if (HI_SUCCESS != s32Ret)
    {
        printf("isp init failed!\n");
        return s32Ret;
    }

    /* 设置ImageAttr和InputTiming */
```



```
stImageAttr.enBayer      = BAYER_GRBG;
stImageAttr.ul6FrameRate = 30;
stImageAttr.ul6Height    = 720;
stImageAttr.ul6Width     = 1280;
s32Ret = HI_MPI_ISP_SetImageAttr(&stImageAttr);
if (HI_SUCCESS != s32Ret)
{
    printf("set image attr failed!\n");
    return s32Ret;
}

stInputTiming.enWndMode = ISP_WIND_NONE;
s32Ret = HI_MPI_ISP_SetInputTiming(&stInputTiming);
if (HI_SUCCESS != s32Ret)
{
    printf("set input timing 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

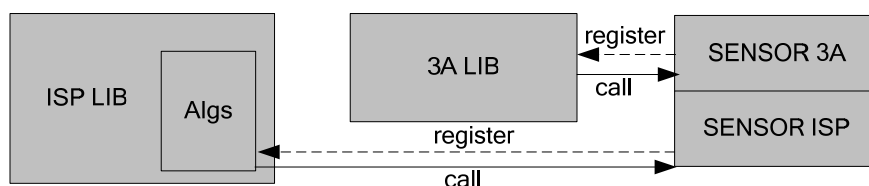
AE 库有用到标准 C 库的数学库，请使用者在 Makefile 中增加 `-lm` 编译条件。



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。

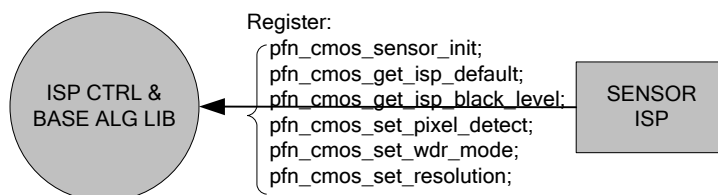
图2-2 Sensor 适配示意图



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 stIsprRegister;  
ISP_SENSOR_EXP_FUNC_S *pstSensorExpFunc = &stIsprRegister.stSnsExp;  
  
memset(pstSensorExpFunc, 0, sizeof(ISP_SENSOR_EXP_FUNC_S));  
pstSensorExpFunc->pfn_cmos_sensor_init = sensor_init;  
pstSensorExpFunc->pfn_cmos_get_isp_default = cmos_get_isp_default;  
pstSensorExpFunc->pfn_cmos_get_isp_black_level = cmos_get_isp_black_level;  
pstSensorExpFunc->pfn_cmos_set_pixel_detect = cmos_set_pixel_detect;  
pstSensorExpFunc->pfn_cmos_set_wdr_mode = cmos_set_wdr_mode;  
s32Ret = HI_MPI_ISP_SensorRegCallBack(IMX104_ID, &stIsprRegister);  
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_get_isp_default	获取 ISP 基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取 sensor 的黑电平值的回调函数指针。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_set_wdr_mode	设置 wdr 模式和线性模式切换的回调函数指针。
pfn_cmos_set_resolution	设置分辨率切换的回调函数指针。



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2-2 ISP_CMOS_DEFAULT_S 的成员变量

成员名称	子成员名称	描述
stComm	u8Rggb	sensor 输出 RGrGbB 的顺序，取值范围为[0,3]。
	u8BalanceFe	不建议修改，推荐使用默认值 0x1。
stDenoise	u8SinterThresh	不建议修改，推荐使用默认值 0x15。
	u8NoiseProfile	噪声型式，设置为 0 表示 ISP 默认。建议使用默认值 0。
	u16Nr0	不建议修改，推荐使用默认值 0x0。
	u16Nr1	不建议修改，推荐使用默认值 0x0。
stDrc	u8DrcBlack	不建议修改，推荐使用默认值 0x0。
	u8DrcVs	不建议修改，推荐使用默认值。线性 0x04，WDR 0x08。
	u8DrcVi	不建议修改，推荐使用默认值。线性 0x08，WDR 0x01。
	u8DrcSm	不建议修改，推荐使用默认值。线性



成员名称	子成员名称	描述
		0xa0, WDR 0x3c。
	u16DrcWl	不建议修改, 推荐使用默认值。线性 0x4ff, WDR 0xffff。
stAgcTbl	bValid	该结构体的数据是否有效, 取值范围为 [0,1]。
	au8SharpenAltD	根据增益动态调节图像大边缘锐度的插值数组, 取值范围为[0,255]。
	au8SharpenAltUd	根据增益动态调节图像小纹理锐度的插值数组, 取值范围为[0,255]。
	au8SnrThresh	根据增益动态设置图像去噪强度的插值数组, 取值范围为[0,255]。
	au8DemosaicLumThresh	该数组用来设置图像的大边缘锐度的亮度门限值, 建议用户使用默认参数值, 取值范围为[0,255]。
	au8DemosaicNpOffset	该数组用来设置图像的噪声参数, 建议用户使用默认参数值, 取值范围为 [0,255]。
	au8GeStrength	该数组用来设置绿色平衡模块的参数, 建议用户使用默认参数值, 取值范围为 [0,255]。
stNoiseTbl	bValid	该结构体的数据是否有效, 取值范围为 [0,1]。
	au8NoiseProfileWeightLut	该数组用来设置与 sensor 特性相关的噪声型式, 作为去噪模块的输入, 建议用户使用默认参数值, 取值范围为 [0,255]。
	au8DemosaicWeightLut	该数组用来设置与 sensor 特性相关的噪声型式, 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



成员名称	子成员名称	描述
		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]。通常在 sensor 支持 wdr 模式时需要配置。
	au16Gammafe	GammaFe 表，取值范围为[0,0xFFFF]。
stShading	bValid	该结构体的数据是否有效，取值范围为[0,1]。如果不需要 shading 校正，可以配置为无效。
	u16RCenterX	R 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16RCenterY	R 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterX	G 分量中心点的 X 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16GCenterY	G 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	u16BCenterX	B 分量中心点的 X 轴坐标。



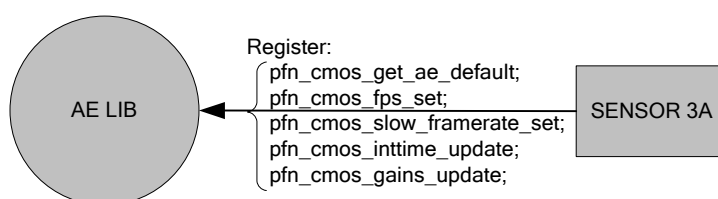
成员名称	子成员名称	描述
		取值范围为[0x0, 0xFFFF]
	u16BCenterY	B 分量中心点的 Y 轴坐标。 取值范围为[0x0, 0xFFFF]
	au16RShadingTbl	R 分量的校正表。 取值范围为[0x0, 0xFFFF]
	au16GShadingTbl	G 分量的校正表。 取值范围为[0x0, 0xFFFF]
	au16BShadingTbl	B 分量的校正表。 取值范围为[0x0, 0xFFFF]
	u16ROffCenter	R 分量中心点与最远的角的距离。距离 越大，数值越小。 取值范围为[0x0, 0xFFFF]
	u16GOffCenter	G 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。 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。

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;

memset(pstExpFuncs, 0, sizeof(AE_SENSOR_EXP_FUNC_S));
pstExpFuncs->pfn_cmos_get_ae_default = cmos_get_ae_default;
pstExpFuncs->pfn_cmos_fps_set = cmos_fps_set;
pstExpFuncs->pfn_cmos_slow_framerate_set = cmos_slow_framerate_set;
pstExpFuncs->pfn_cmos_inttime_update = cmos_inttime_update;
pstExpFuncs->pfn_cmos_gains_update = cmos_gains_update;

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

需要在 xxx_cmos.c 中实现以下回调函数：

表2-3 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。
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。



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2-4 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}，WDR 模式为

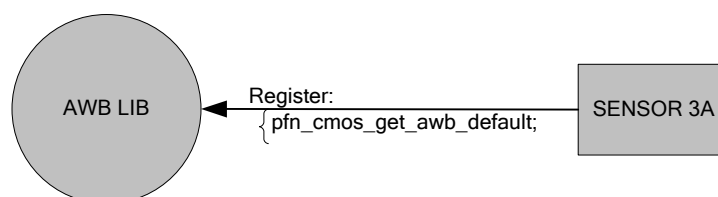


成员名称	描述
	{0x20,0x40,0x60,0x80}。
u8AeCompensation	AE 亮度目标值，取值范围为[0,255]，建议用 0x80。
u32LinesPer500ms	每 500ms 的总行数。
u32FlickerFreq	抗闪频率，数值为当前电源频率的 256 倍。
u32MaxIntTime	最大曝光时间，以行为单位。
u32MinIntTime	最小曝光时间，以行为单位。
u32MaxIntTimeTarget	最大曝光时间目标值，以行为单位。
u32MinIntTimeTarget	最小曝光时间目标值，以行为单位。
stIntTimeAccu	曝光时间精度。
u32MaxAgain	最大模拟增益，以倍为单位。
u32MinAgain	最小模拟增益，以倍为单位。
u32MaxAgainTarget	最大模拟增益目标值，以倍为单位。
u32MinAgainTarget	最小模拟增益目标值，以倍为单位。
stAgainAccu	模拟增益精度。
u32MaxDgain	最大数字增益，以倍为单位。
u32MinDgain	最小数字增益，以倍为单位。
u32MaxDgainTarget	最大数字增益目标值，以倍为单位。
u32MinDgainTarget	最小数字增益目标值，以倍为单位。
stDgainAccu	数字增益精度。

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;

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

需要在 xxx_cmos.c 中实现以下回调函数：

表2-5 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2-6 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]。
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]。



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

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。

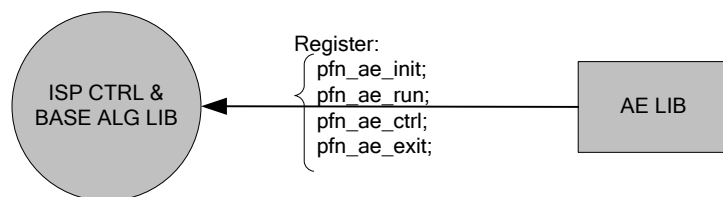
Sensor 注册到 ISP 库，以实现差异化适配 Firmware 中的基础算法单元的部分，仍需要参考使用者指南部分的内容实现，以保证 Firmware 中的坏点校正、去噪、色彩增强、镜头阴影校正等处理算法正常运行。Sensor 注册到 3A 算法库的部分，请用户根据自己开发定制的 3A 库，自行定义数据结构，实现 Sensor 曝光控制等，Firmware 并不对 sensor 的曝光进行控制。

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

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	无	帧的累加计数，取值范围为[0, 0xFFFFFFFF]。
pstAeStat1	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au16MeteringHist	五段直方图的统计信息数组，取值范围为[0, 0xFFFF]。
pstAeStat2	au8MeteringHistThresh	五段直方图的分割门限值数组，取值范围为[0, 255]。
	au16MeteringMemArray	五段直方图的分区间的统计信息数组，取值范围为[0, 0xFFFF]。
pstAeStat3	au16HistogramMemArray	256 段直方图的统计信息数组，取值范围为[0, 0xFFFF]。

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
u32IspDgain	无	ISP 的数字增益。
u32IspDgainShift	无	ISP 的数字增益的精度。
u32Iso	无	AE 计算得出的总增益值。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	au8MeteringHistThresh	五段直方图的分割门限值数组，取值

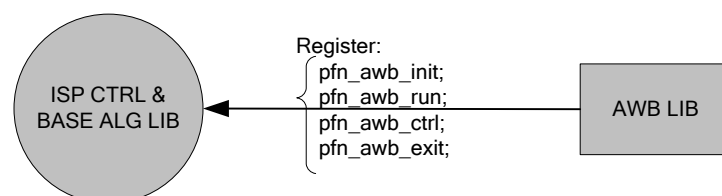


成员名称	子成员名称	描述
		范围为[0, 255]。
	au8WeightTable	15x17 个区间的 AE 权重表，取值范围为[0, 255]。

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



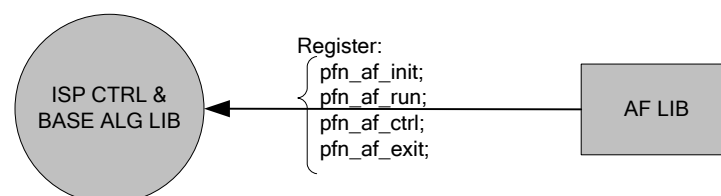
表3-8 运算结果 ISP_AWB_RESULT_S 的成员变量

成员名称	子成员名称	描述
au32WhiteBalanceGain	无	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	无	色彩还原矩阵，8bit 精度表示。
stStatAttr	bChange	该结构体中的值是否需要配置寄存器。
	u16MeteringWhiteLevelAwb	统计白点信息时，找白点的上限。默认值 0x3ac。
	u16MeteringBlackLevelAwb	统计白点信息时，找白点的下限。默认值 0x40。
	u16MeteringCrRefMaxAwb	统计白点信息时，白点区域的最大 R/G 值。默认值 512。
	u16MeteringCbRefMaxAwb	统计白点信息时，白点区域的最大 B/G 值。默认值 512。
	u16MeteringCrRefMinAwb	统计白点信息时，白点区域的最小 R/G 值。默认值 128。
	u16MeteringCbRefMinAwb	统计白点信息时，白点区域的最小 B/G 值。默认值 128。

3.4 AF 算法注册 ISP 库

AF 算法注册 ISP 库调用 HI_MPI_ISP_AfLibRegCallBack，如[图 3-3](#)所示，详细说明参见《HiISP 开发参考》。

图3-3 AF 算法向 ISP 库注册的回调函数



海思 AF 算法没有实现具体算法，只实现了一个注册框架，示例和 AE 算法库注册类似。



用户需要在自开发定制的 AF 库中实现以下回调函数：

表3-9 AF 算法向 ISP 库注册的回调函数

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



说明

调用 HI_MPI_ISP_Init 时将调用 pfn_af_init 回调函数，以初始化 AF 算法库。

调用 HI_MPI_ISP_Run 时将调用 pfn_af_run 回调函数，以运行 AF 算法库，计算得到白平衡增益、色彩校正矩阵。

pfn_af_ctrl 回调函数的目的是改变算法库内部状态。

调用 HI_MPI_ISP_Exit 时将调用 pfn_af_exit 回调函数，以销毁 AF 算法库。

表3-10 初始化参数 ISP_AF_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 AF 注册的 sensor 是否一致。
s32Rsv	保留参数。

表3-11 统计信息 ISP_AF_INFO_S 的成员变量

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

表3-12 运算结果 ISP_AF_RESULT_S 的成员变量

成员名称	描述
s32Rsv	保留参数。



说明

AF 最终需要调用类似 PWM 的外设实现自动对焦，不需要反馈结果给 ISP。



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 cmos.c 文件的修改

3A 版本对非 3A 版本的 cmos.c 文件进行了一些整理，主要结构体的说明请参考《使用者指南》章节，这里做一些补充说明。

与 ISP firmware 中自带的一些算法的参数整理到 ISP_CMOS_DEFAULT_S 结构体中，例如 RGGB 顺序、去噪、DRC、去马赛克、Gamma 等。

ISP_CMOS_BLACK_LEVEL_S 中是黑电平参数，因为有些 sensor 的黑电平会动态改变。

与海思 AE 相关的参数整理到 AE_SENSOR_DEFAULT_S 结构体中，例如曝光时间、模拟增益、数字增益、抗闪等，并修改为浮点精度表示的方式。统一用浮点精度的方式表示后，所有的曝光量分配的代码不再出现在 cmos.c 中。精度总体上分为倍数精度和分贝精度两类，倍数精度是线性增加的，分贝精度是幂级增加的。例如 0.125 倍数精度表示 sensor 最小可调的精度是 1/8 倍，例如 0.3 分贝精度表示 sensor 最小可调的精度是 0.3db。Sensor 的精度总是可以拆成倍数精度和分贝精度表示，例如对于 OV9712，增益是 $a(1+b/16)$ ，a 可以取 1、2、4、8、16，b 可取 1~15，那么可以认为模拟增益是分贝精度，最小可调精度是 6db(2 倍)，数字增益是线性精度，最小可调精度是 0.0625(1/16)倍，在 cmos_gains_update 中的 u32Again 和 u32Dgain 分别是以 6db 和 0.0625 倍为单位的。

与海思 AWB 相关的参数整理到 AWB_SENSOR_DEFAULT_S 结构体中，例如白平衡校正曲线参数、静态白平衡增益、饱和度等。

其他内容是完成一些操作的，例如 cmos_set_pixel_detect 设置坏点校正，cmos_set_wdr_mode 切换 WDR 模式，cmos_fps_set 设置基准帧率，cmos_slow_framerate_set 设置降帧。

用户如果开发 3A，那么 cmos.c 中和 AE、AWB 的部分也需要开发，例如 AE 的曝光时间、增益等的精度设置、AWB 的参数等，cmos_get_isp_default、cmos_get_isp_black_level、cmos_set_pixel_detect、cmos_set_wdr_mode 可以复用，。



4.4 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。

4.5 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，0x30000~0x3FFFF 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。