Rockchip Developement Guide 3A ISP32

文件标识：RK-KF-GX-612

发布版本：V0.1.0

日期：2022-4-27

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文旨在描述RkAiq（Rk Auto Image Quality）模块的作用，整体工作流程，及相关的API接口。主要给使用RkAiq模块进行ISP功能开发的工程师提供帮助。

产品版本``

芯片名称	内核版本
RK1106	Linux 5.10

读者对象

本文档（本指南）主要适用于以下工程师：

ISP模块软件开发工程师

系统集成软件开发工程师

各芯片系统支持状态

芯片名称	**BuildRoot**	**Debian**	**Yocto**	**Android**
RK1106	Y	N	N	Y

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	朱林靖 池晓芳 胡克俊	2022-4-27	ISP3A 开发指南初版

目录

4. 2.4 参考代码样例

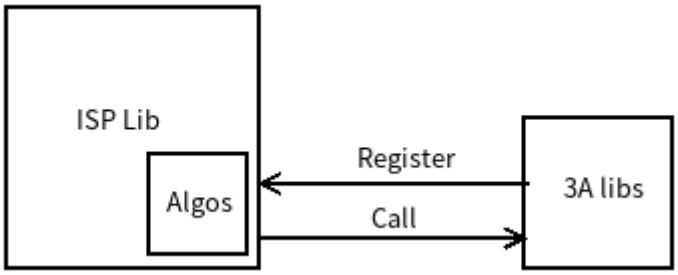
1 概述

该文档主要介绍3A库的实现方式，旨在指导用户如何实现定制化的3A算法库。

3A算法库依赖于AIQ，AIQ内已包含有RK的3A算法库，并且已经默认使能。用户可根据需要按该文档方式实现定制化的3A库。

1. 1.1 设计思路

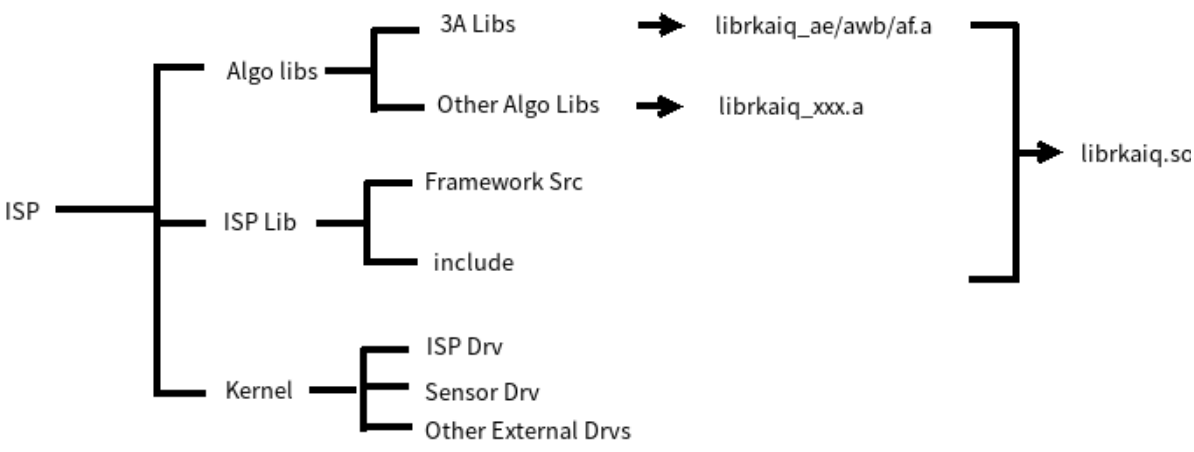
基本设计思路如下图所示：



- 3A库通过注册方式注册给ISP库，注意RK的3A库已隐式的注册，不需要用户显示注册
- 3A库注册给ISP库后，ISP库从驱动拿到3A统计后，回调3A库接口得到新的3A参数，ISP库将新的3A参数设置给驱动

2. 1.2 文件组织

文件组织如下图所示：



- ISP Firmware 分成应用层的librkaiq.so 和 驱动层 的 ISP 驱动以及外设驱动，包括 Sensor、VCM及Flashlight等
- librkaiq.so 中包含了众多的算法库，如 3A 算法库、HDR算法库等等，算法库都以静态库形式存在，最后链接形成librkaiq.so。除了 librkaiq_ae/awb/af.a 3A 库是不提供源码的，其他基础库源码都是开放的。
- 框架支持所有模块的算法库都使用客户算法，但一般来说，除3A库希望定制化外，其他基础库可使用RK提供的默认库。

3. 1.3 开发模式

支持以下三种开发模式：

- 3A库使用RK库。使用该方式时，RK的3A库API都可使用，具体包括：rk_aiq_user_api2_ae.h，rk_aiq_user_api2_af.h，rk_aiq_user_api2_awb.h。
- 3A库部分使用RK库，部分使用用户自定义库。如 AE 库使用自定义库，AWB 库使用RK库。
- 3A库自定义库和RK库同时使用。如AE库，自定义库和RK库同时跑时，会先跑RK AE 库，然后跑自定义AE库，自定义库结果覆盖RK AE库结果。此种模式用于简化自定义库开发，自定义库可不需要输出所有 AIQ 框架需要的结果，部分结果可由 RK AE 库输出。

4. 1.4 软件流程

4.1 1.4.1 基础流程

RK 3A 算法不需要用户显示注册，AIQ 框架内部已隐式注册。自定义 3A 算法注册，以自定义 AE 算法注册为例，示例伪代码如下：

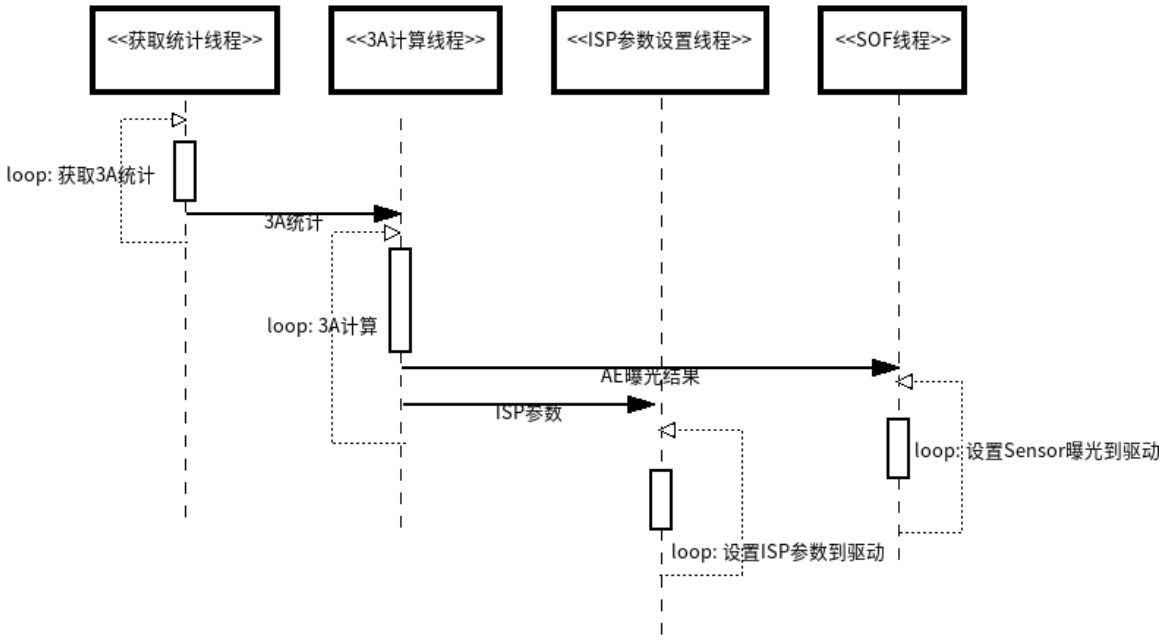
```
//  初始化使用场景，不是必须，默认为 normal，day，用于选择 json iq 文件中的场景参数
if (work_mode == RK_AIQ_WORKING_MODE_NORMAL)
        ret = rk_aiq_uapi2_sysctl_preInit_scene(sns_entity_name, "normal",
"day");
  else
        ret = rk_aiq_uapi2_sysctl_preInit_scene(sns_entity_name, "hdr", "day");

//  根据使用模式是环视还是单Camera，初始化  Group Ctx 或者 AIQ ctx
   if (!group_mode)
        ctx->aiq_ctx = rk_aiq_uapi2_sysctl_init(sns_entity_name, ctx->iqpath,
NULL, NULL);
  else {
       rk_aiq_camgroup_instance_cfg_t camgroup_cfg;
       memset(&camgroup_cfg, 0, sizeof(camgroup_cfg));
```

```
        camgroup_cfg.sns_num = 1;
        camgroup_cfg.sns_num++;
        camgroup_cfg.sns_ent_nm_array[0] = sns_entity_name;
        camgroup_cfg.sns_ent_nm_array[1] = sns_entity_name2;
        camgroup_cfg.config_file_dir = ctx->iqpath;
        camgroup_cfg.overlap_map_file = "srcOverlapMap.bin";
        ctx->camgroup_ctx = rk_aiq_uapi2_camgroup_create(&camgroup_cfg);
  }

// 如果需要注册自定义 AE 算法，则注册自定义 AE 回调
  rk_aiq_customeAe_cbs_t cbs = {
                            .pfn_ae_init = custom_ae_init,
                            .pfn_ae_run = custom_ae_run,
                            .pfn_ae_ctrl = custom_ae_ctrl,
                            .pfn_ae_exit = custom_ae_exit,
                        };
  rk_aiq_uapi2_customAE_register((const rk_aiq_sys_ctx_t*)(ctx->camgroup_ctx),
&cbs);

// 如果需要运行第三方 AE 算法，则使能自定义 AE 算法。第三方AE库注册接口，单摄和环视共用。
//当前 AIQ 版本，默认 RK 算法和第三方算法都会运行， 如果需要，可以调用
rk_aiq_uapi2_sysctl_enableAxlib 强制关掉 RK AE。

  rk_aiq_uapi2_customAE_enable((const rk_aiq_sys_ctx_t*)(ctx->camgroup_ctx),
true);

// 准备ISP pipeline 及配置 ISP、Sensor 等初始化参数
// 如果需要，可在prepare前调用模块 API，修改模块初始化参数，否则初始化参数由 IQ 文件指定，或者
是AIQ中硬代码指定，或者是芯片复位值
 if (!group_mode) {
      rk_aiq_uapi2_sysctl_prepare(ctx->aiq_ctx, ctx->width, ctx->height,
work_mode);
      rk_aiq_uapi2_sysctl_start(ctx->aiq_ctx );
 } else {
      rk_aiq_uapi2_camgroup_prepare(ctx->camgroup_ctx, work_mode);
      ret = rk_aiq_uapi2_camgroup_start(ctx->camgroup_ctx);
 }

//  开启 VI 数据流，注意该部分未调用任何 AIQ 库接口。
start_capturing(ctx);
......
//  AIQ 内部线程循环工作：从驱动获取 3A 统计信息，调用各算法库计算新的ISP参数、Sensor参数等，
下发新的参数给ISP驱动、Sensor驱动等。
// 此过程可调用 API 设置各算法模块参数
......

//  退出时先停止数据流
stop_capturing(ctx);

// 停止掉 AIQ ctx 或者 Group ctx
 if (!group_mode)
       rk_aiq_uapi2_sysctl_stop(ctx->aiq_ctx, false);
else
       rk_aiq_uapi2_camgroup_stop(ctx->camgroup_ctx);
```

```
// 反注册第三方 AE
 rk_aiq_uapi2_customAE_unRegister(ctx->aiq_ctx);

// 反初始化 AIQ ctx 或者 Group ctx
 if (!group_mode)
        rk_aiq_uapi2_sysctl_deinit(ctx->aiq_ctx);
else
       rk_aiq_uapi2_camgroup_destroy(ctx->camgroup_ctx);
```

4.2 1.4.2 内部运行流程

AIQ 内部运行如下图所示：



- 获取统计线程：该线程不断从ISP驱动获取 3A 统计，然后发送给 3A 计算线程。
- 3A计算线程：该线程收到统计后，开始调用各模块算法（包括第三方算法回调），计算新的参数，然后将新参数发给 ISP参数设置线程和 SOF线程。
- ISP参数设置线程：该线程收到新的ISP参数设置请求后，在合适时机下发给ISP驱动。
- SOF线程：该线程为 Sensor 帧头事件的响应函数，该线程收到新的曝光设置请求后，从队列中取出新曝光参数设置给Sensor驱动。

2 开发者指南

1. 2.1 AE 算法注册

AE算法注册流程涉及算法注册、算法使能、算法注销，注册调用rk_aiq_uapi_customAE_register接口，使能调用rk_aiq_uapi_customAE_enable接口， 注销调用rk_aiq_uapi_customAE_unRegister接口

1.1 2.1.1 算法注册API

1.1.1 rk_aiq_uapi2_customAE_register

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_customAE_register(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_customeAe_cbs_t* cbs)
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入
cbs	回调函数指针	

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.1.2 rk_aiq_uapi2_customAE_enable

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_customAE_enable(const rk_aiq_sys_ctx_t* ctx, bool enable);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入
enable	AE算法使能位	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.1.3 rk_aiq_uapi2_customAE_unRegister

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_customAE_unRegister(const rk_aiq_sys_ctx_t* ctx);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.2 2.1.2 回调函数以及数据类型

用户需要在自开发定制的AE库中实现以下回调函数：

```
rk_aiq_customeAe_cbs_t cbs = {
.pfn_ae_init = custom_ae_init,
.pfn_ae_run = custom_ae_run,
.pfn_ae_ctrl = custom_ae_ctrl,
.pfn_ae_exit = custom_ae_exit,
};
```

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【该参数暂时无效】
pfn_ae_exit	销毁AE的回调函数指针

1.2.1 custom_ae_init

【描述】
初始化AE算法库

【语法】

```
int32_t custom_ae_init(void* ctx);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.2.2 custom_ae_run

【描述】
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，及更新硬件配置参数

【语法】

```
int32_t custom_ae_run(void* ctx, const rk_aiq_customAe_stats_t* pstAeInfo,
                      rk_aiq_customeAe_results_t* pstAeResult);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入
pstAeInfo	输入数据参数指针，包含AE硬件统计信息及其同步的曝光信息	输入
pstAeResult	输出算法结果指针，包含sensor的曝光结果参数，及更新硬件配置参数	输出

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.2.3 custom_ae_ctrl

【描述】
改变算法库内部状态，暂无法使用

【语法】

```
int32_t custom_ae_ctrl(void* ctx, uint32_t u32Cmd, void *pValue);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.2.4 custom_ae_exit

【描述】
注销AE算法库

【语法】

```
int32_t custom_ae_exit(void* ctx);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针，可兼容单摄及环视应用	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

1.2.5 输入参数

【说明】

第三方AE输入数据参数包括图像亮度统计值及对应的曝光参数值，可兼容单摄和环视应用

【定义】

```
#define RK_AIQ_MAX_HDR_FRAME (3)
typedef struct rk_aiq_customAe_stats_s
{
    //hw stats
    Aec_Stat_Res_t rawae_stat[RK_AIQ_MAX_HDR_FRAME]; // with awb gain
    Aec_Stat_Res_t extra;      // with awb gain, lsc, TMO

    //exposure
    RkAiqExpParamComb_t linear_exp;
    RkAiqExpParamComb_t hdr_exp[RK_AIQ_MAX_HDR_FRAME];

    struct rk_aiq_customAe_stats_s* next; // for surround view(multiple cams)
} rk_aiq_customAe_stats_t;
```

【成员】

成员名称	描述
rawae_stat[RK_AIQ_MAX_HDR_FRAME]	基于raw图的前置硬件统计信息，最多支持3帧raw图统计。线性曝光模式下，仅rawae_stat[0]有效；Hdr曝光模式下，rawae_stat[0-2]依次表示短、中、长帧的硬件统计信息。1106平台至多支持HDR 2TO1模式，故rawae_stat仅0-1元素有效
extra	基于raw图的后置硬件统计信息
linear_exp	线性模式下的曝光参数，与硬件统计信息同步
hdr_exp[RK_AIQ_MAX_HDR_FRAME]	Hdr模式下的曝光参数，与硬件统计信息同步。1106平台仅0~1有效，分别表示短、长帧的曝光参数
next	仅环视多摄应用下有效，该指针指向下一个camera的输入数据参数，各camera对应的输入参数成员内容相同；非环视多摄应用，该指针为空。1106平台不支持

【说明】

- 输入数据参数分为两类参数，分别是图像的硬件统计信息与图像所对应的曝光参数
- 输入数据参数可兼容单摄及环视多摄应用，通过next指针获取多个camera的输入数据参数
- 图像的硬件统计信息数据类型为Aec_Stat_Res_t，曝光信息数据类型为RkAiqExpParamComb_t，数据类型说明详见《Rockchip_Development_Guide_ISP32》文档
- RK_AIQ_MAX_HDR_FRAME表示RK平台至多支持3帧HDR，针对1106平台仅支持2帧HDR

1.2.6 运算结果

【说明】

第三方AE输出结果参数包括曝光参数、硬件参数等，兼容单摄和环视应用

【定义】

```
#define RK_AIQ_MAX_HDR_FRAME (3)
typedef struct rk_aiq_i2c_data_s {
    bool            bValid;
    unsigned int    nNumRegs;
    unsigned int*   pRegAddr;
    unsigned int*   pAddrByteNum;
    unsigned int*   pRegValue;
    unsigned int*   pValueByteNum;
    unsigned int*   pDelayFrames;
} rk_aiq_i2c_data_t;

typedef struct rk_aiq_customeAe_results_singel_s
{
    //exposure result (including:reg value & real value)
    RkAiqExpParamComb_t linear_exp;
    RkAiqExpParamComb_t hdr_exp[RK_AIQ_MAX_HDR_FRAME];
    rk_aiq_i2c_data_t   exp_i2c_params;

    //hw result
```

```
    struct window meas_win;
    unsigned char meas_weight[15 * 15];

    struct rk_aiq_customeAe_results_singel_s* next; // for surround view(multiple
cams)
} rk_aiq_customeAe_results_single_t;

typedef struct rk_aiq_customeAe_results_s
{
    //exposure result (including:reg value & real value)
    RkAiqExpParamComb_t linear_exp;
    RkAiqExpParamComb_t hdr_exp[RK_AIQ_MAX_HDR_FRAME];
    rk_aiq_i2c_data_t   exp_i2c_params;

    //hw result
    struct window meas_win;
    unsigned char meas_weight[15 * 15];

    RkAiqIrisParamComb_t Iris;
    uint32_t frame_length_lines;
    bool     is_longfrm_mode;

    struct rk_aiq_customeAe_results_singel_s* next; // for surround view(multiple
cams)
} rk_aiq_customeAe_results_t;
```

【成员】

成员名称	子成员	描述
linear_exp	exp_real_params exp_sensor_params	线性模式下的曝光参数，包含曝光实际值（exp_real_params）和RK格式的寄存值（exp_sensor_params）
hdr_exp[RK_AIQ_MAX_HDR_FRAME]	exp_real_params exp_sensor_params	Hdr模式下的曝光参数，包含曝光实际值（exp_real_params）和RK格式的寄存值（exp_sensor_params） 其中0~2分别表示短、中、长帧的曝光参数，对于HDR2帧合成，0与1元素有效，对于HDR3帧合成，0-2皆有效
exp_i2c_params	bValid nNumRegs pRegAddr pAddrByteNum pRegValue pValueByteNum pDelayFrames	i2c寄存器值参数 当bValid为true时，使用exp_i2c_params参数进行寄存器值设置；当bValid为false时，使用上述linear_exp或hdr_exp中的RK格式寄存器值参数进行寄存器设置
frame_length_lines		sensor的vts值，与帧率设置有关
is_longfrm_mode		长帧模式使能位 true：开启长帧模式；false: 关闭长帧模式 该参数仅在Hdr模式下有效
Iris	PIris DCIris	光圈设置参数，包含P光圈和DC光圈设置参数
meas_win	h_offs v_offs h_size v_size	硬件统计窗口区域参数 h/v_offs分别表示窗口左上角顶点相对于感光区域左上角顶点沿水平、竖直方向的偏移； h/v_size分别表示窗口沿水平、竖直方向的尺寸大小
meas_weight		硬件统计权重参数，包含15X15个权重参数，取值范围0~32
next		非环视应用，该指针需为空； 环视应用，若多个camera需要设置相同的算法结果，该指针需为空，仅需设置rk_aiq_customeAe_results_t中的成员，而后所有camera皆使用rk_aiq_customeAe_results_t中的算法结果作为各自的最终结果； 若多个camera需要设置不同的算法结果，需由用户自行申请next指针内存，添加下一个camera的算法结果

【注意事项】

- 输出算法结果，可兼容单摄应用及环视多摄应用。环视多摄应用下，可兼容单一算法结果和多个算法结果。
- meas_win为AE硬件统计的窗口区域参数，硬件统计（分块亮度、直方图）将基于窗口区域展开。对于HDR多帧曝光应用，默认所有帧的硬件统计窗口一致。
- meas_weight为加权直方图统计所需的权重参数，一般该参数需与分块加权亮度均值所使用的的权重（软件权重）一致
- 设置曝光时，需要配置曝光实际值及对应寄存器值。曝光实际值包括：曝光时间（单位：秒）、曝光增益（单位：倍数）、DCG状态（0：LCG，1：HCG），供其他算法模块使用；曝光寄存器值为与sensor对接的寄存器值，支持RK格式和第三方格式。
- 设置曝光寄存器值时，支持使用RK格式和第三方格式。RK格式的寄存器值无需客户设置，内部自行根据曝光实际值转换，要求exp_i2c_params中的bValid值为false；第三方格式需要用户设置所需寄存器值及对应地址，要求exp_i2c_params中的bValid值为true.
- 对于环视应用，如环视中所有camera需设置相同的曝光和硬件值，仅需设置rk_aiq_customeAe_results_t内的参数，next指针为空；如环视中各camera需要设置不同的曝光和硬件中，则需要按照顺序依次设置结果值，为next指针分配内存指向下一个camera算法结果。需要注意的是，rk_aiq_customeAe_results_t与rk_aiq_customeAe_results_single_t中的参数存在不同之处，前者相较于二者多了个别结果参数，其作为公共参数，默认所有camera都设置相同值。
- 图像的硬件统计信息数据类型为Aec_Stat_Res_t，RK曝光信息数据类型为RkAiqExpParamComb_t，RK光圈参数数据类型为RkAiqIrisParamComb_t，上述数据类型说明详见《Rockchip_Development_Guide_ISP32》文档中统计信息模块

2. 2.2 AWB 算法注册

RK AWB 算法实现了一个 rk_aiq_uapi_customAWB_register 的注册函数，用户调用注册函数以实现向ISP注册 Custom AWB 算法，示例和 AE 算法库注册类似，并通过 rk_aiq_uapi_customAWB_enable 去使能 Custom AWB 算法。

注：为顺利开展移植工作，在移植前建议查看：

(1)《Rockchip_Color_Optimization_Guide》文档的以下内容，

(a)"2 AWB/2.1功能描述" 章节内容及AWB流程图内容

```
(b)"2 AWB/2.2关键参数/硬件的白点检测流程" 章节中图 AWB 白点检测流程图
```

(2)《Rockchip_Development_Guide_ISP32》"统计信息 / 数据类型 / rk_aiq_isp_awb_stats2_v3x_t " 章节

2.1 2.2.1 算法注册API

2.1.1 rk_aiq_uapi_customAWB_register

【描述】

Custom AWB 算法注册。

【语法】

```
XCamReturn
rk_aiq_uapi_customAWB_register(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_customeAwb_cbs_t* cbs);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
cbs	Custom AWB 算法向 ISP 库注册的回调函数，参考后面的 rk_aiq_customeAwb_cbs_t 结构体说明	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

【注意】

- 须先调用 rk_aiq_uapi_sysctl_init 初始化AIQ上下文指针 ctx。

【需求】

- 头文件：rk_aiq_user_api_custom_awb.h
- 库文件：librkaiq.so

2.1.2 rk_aiq_uapi_customAWB_enable

【描述】

Custom AWB 算法使能。

【语法】

```
XCamReturn
rk_aiq_uapi_customAWB_enable(const rk_aiq_sys_ctx_t* ctx, bool enable);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
enable	Custom AWB 使能开关 取值：true / false 默认值：false	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

【注意】

- 须在 rk_aiq_uapi_customAWB_register 完成 Custom AWB 算法注册之后调用。

【需求】

- 头文件：rk_aiq_user_api_custom_awb.h
- 库文件：librkaiq.so

2.1.3 rk_aiq_uapi_customAWB_unRegister

【描述】
Custom AWB 算法注销。

【语法】

```
XCamReturn
rk_aiq_uapi_customAWB_unRegister(const rk_aiq_sys_ctx_t* ctx);
```

【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入

【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

【注意】

- 须在 rk_aiq_uapi_customAWB_register 完成 Custom AWB 算法注册之后调用。

【需求】

- 头文件：rk_aiq_user_api_custom_awb.h
- 库文件：librkaiq.so

2.2 2.2.2 回调函数以及数据类型

2.2.1 向 ISP 库注册的回调函数

2.2.1.1 rk_aiq_customeAwb_cbs_t

【说明】
定义Custom AWB 算法向 ISP 库注册的回调函数。

【定义】

```
typedef struct rk_aiq_customeAwb_cbs_s
{
    int32_t (*pfn_awb_init)(void* ctx);
    int32_t (*pfn_awb_run)(void* ctx, const void* pstAwbInfo, void*
pstAwbResult);
    int32_t (*pfn_awb_run)(void* ctx, uint32_t u32Cmd, void *pValue);
    int32_t (*pfn_awb_exit)(void* ctx);
} rk_aiq_customeAwb_cbs_t;
```

【成员】

成员名称	描述
pfn_awb_init	初始化 第一次初始化后将被 AwbDemoPrepare 函数调用
pfn_awb_ctrl	控制 Custom AWB 内部状态的回调函数指针，暂不支持。
pfn_awb_run	运行 Custom AWB 的回调函数指针 pstAwbInfo实际类型为rk_aiq_customAwb_stats_t , pstAwbResult实际类型为rk_aiq_customeAwb_results_t ，均参考后面的说明 被 AwbDemoProcessing 调用 若 pstAwbResult==nullptr 表示为初始化那一次，用于配置初始化时的pstAwbResult，否则需实现基于统计信息 pstAwbInfo 计算 pstAwbResult 的功能
pfn_awb_exit	释放申请的内存等 被AwbDemoDestroyCtx调用

【注意】

- 用户需要在自开发定制的 AWB 库中实现以上回调函数。
- pfn_awb_run实现可参考third_party_awb_algo_v32.cpp 的custom_awb_run函数中的伪代码

2.2.2 统计信息

2.2.2.1 rk_aiq_customAwb_stats_t

【说明】
定义Custom AWB 算法获取的白平衡硬件统计信息。

【定义】

```
typedef struct rk_aiq_customAwb_stats_s
{
    rk_aiq_awb_stat_wp_res_light_v201_t
light[RK_AIQ_AWB_MAX_WHITEREGIONS_NUM_V32];
    int WpNo2[RK_AIQ_AWB_MAX_WHITEREGIONS_NUM_V32];
    rk_aiq_awb_stat_blk_res_v201_t   blockResult[RK_AIQ_AWB_GRID_NUM_TOTAL];
    rk_aiq_awb_stat_wp_res_v201_t
excWpRangeResult[RK_AIQ_AWB_STAT_WP_RANGE_NUM_V201];
    unsigned int WpNoHist[RK_AIQ_AWB_WP_HIST_BIN_NUM];
    struct rk_aiq_customAwb_stats_s* next;
} rk_aiq_customAwb_stats_t;
```

【成员】

成员名称	描述
light	主窗口下不同光源下的白点统计结果，最多RK_AIQ_AWB_MAX_WHITEREGIONS_NUM_V32个光源。
WpNo2	主窗口下不同光源下的xy域和uv域交集的白点个数，没有小数位。
blockResult	每个块的 RGB 累加 图像采用均匀分块方式，共15x15（RK_AIQ_AWB_GRID_NUM_TOTAL）块。
excWpRangeResult	落在 excludeWpRange 区域里的点的统计结果（只会记录 excludeWpRange 前四个区域），最多4个区域。
WpNoHist	白点直方图每个 bin 的白点个数，没有小数位； 统计的是 XY 大框还是 XY 中框的白点由寄存器 xyRangeTypeForWpHist 确定。
next	无用

【注意】

- 各成员详见《Rockchip_Development_Guide_ISP32》"统计信息/数据类型" 章节 rk_aiq_isp_awb_stats2_v32_t 结构体成员的定义。

2.2.3 运算结果

2.2.3.1 rk_aiq_customeAwb_results_t

【说明】
定义Custom AWB 算法的配置参数及运算结果。

【定义】

```
typedef struct rk_aiq_customeAwb_results_s
{
    bool  IsConverged; //true: converged; false: not converged
    rk_aiq_wb_gain_t awb_gain_algo;
    float awb_smooth_factor;
    rk_aiq_customAwb_hw_cfg_t  awbHwConfig;
    rk_aiq_customeAwb_single_results_t *next;//defalut vaue is nullptr,which
means all cameras with the same cfg;
} rk_aiq_customeAwb_results_t;
```

【成员】

成员名称	描述
IsConverged	表征当前AWBgain是否收敛； true 已收敛，false 未收敛； 默认值：false； 必须配置。
awb_gain_algo	Custom AWB 算法得出的R、Gr、Gb、B 颜色通道的增益； 默认值：{1.0, 1.0, 1.0, 1.0}，不做白平衡校正； 必须配置。
awb_smooth_factor	提供给 CCM 和 LSC 的帧间平滑因子，值越大当前帧的权重越小； 取值范围：[0,1]； 默认值：0.5； 可以不配置。
awbHwConfig	Custom AWB 算法的硬件配置参数； 大部分参数和模组相关需配置正确，其他参数均已配置了默认值，可以不更新； 详情看后面 rk_aiq_customAwb_hw_cfg_t 结构体说明。
next	无用

2.2.3.2 rk_aiq_customeAwb_single_results_t（无用）

【说明】
定义Custom AWB 算法的环视模式下各个camera的配置参数及运算结果，非环视无需关心

【定义】

```
typedef struct rk_aiq_customeAwb_single_results_s
{
    rk_aiq_wb_gain_t awb_gain_algo;//for each camera
    rk_aiq_customAwb_single_hw_cfg_t  awbHwConfig;//for each camera
    struct rk_aiq_customeAwb_single_results_s *next;
} rk_aiq_customeAwb_single_results_t;
```

【成员】

成员名称	描述
awb_gain_algo	同rk_aiq_customeAwb_results_s中awb_gain_algo成员的含义
awbHwConfig	该结构体成员与rk_aiq_customeAwb_results_s中awbHwConfig结构体中相同名字的成员含义相同
next	无用，同rk_aiq_customeAwb_results_s中anext成员的含义

2.2.3.3 rk_aiq_wb_gain_t

- 详见《Rockchip_Development_Guide_ISP32》"AWB/功能级API/数据类型 "章节 rk_aiq_wb_gain_t 结构体定义。

2.2.3.4 rk_aiq_customAwb_hw_cfg_t

【说明】
定义Custom AWB 算法的硬件配置参数，主窗口多窗口配置，统计帧选择等。

【定义】

```
typedef struct rk_aiq_customAwb_hw_cfg_s {
    bool awbEnable;
    rk_aiq_customAwb_Raw_Select_Mode_e frameChoose;
    unsigned short windowSet[4];
    unsigned char lightNum;
    unsigned short maxR;
    unsigned short minR;
    unsigned short maxG;
    unsigned short minG;
    unsigned short maxB;
    unsigned short minB;
    unsigned short maxY;
    unsigned short minY;
    bool multiwindow_en;
    unsigned short multiwindow[RK_AIQ_AWB_MULTIWINDOW_NUM_V201][4];
} rk_aiq_customAwb_hw_cfg_t;
```

【成员】

成员名称	描述
awbEnable	AWB 统计使能开关； true 使能，false 未使能； 默认值：true。
frameChoose	AWB 硬件统计的输入帧选择； 取值CUSTOM_AWB_INPUT_RAW_SHORT、 CUSTOM_AWB_INPUT_RAW_LONG、CUSTOM_AWB_INPUT_BAYERNR、 CUSTOM_AWB_INPUT_DRC。 CUSTOM_AWB_INPUT_RAW_SHORT 选短帧raw； CUSTOM_AWB_INPUT_RAW_LONG 选长帧raw（hdr模式才有效）； CUSTOM_AWB_INPUT_BAYERNR 选bayer2dnr模块的输出； CUSTOM_AWB_INPUT_DRC 选DRC模块的输出；； 默认值：CUSTOM_AWB_INPUT_BAYERNR。
windowSet	AWB 统计主窗口配置； windowSet=[h_offset,v_offset,h_size,v_size]，h：水平方向，v：垂直方向； 取值范围：[0x0, 0xfff]； h_size* v_size 需小于 5120*2880； 默认值：{0, 0, RawWidth, RawHeight}，全窗口,若不改变窗口，无需配置。
lightNum	无用 参与统计的光源数量； 取值范围：[0, 7]； 默认值：7。 需依据标定时采用的光源数配置，标定工具会输出。
maxR	RGB 域统计白点信息时，白点检测的R通道上限； 取值范围：[0x0, 0xff]； 默认值：230。
minR	RGB 域统计白点信息时，白点检测的R通道下限； 取值范围：[0x0, 0xff]； 默认值：3。
maxG	RGB 域统计白点信息时，白点检测的G通道上限； 取值范围：[0x0, 0xff]； 默认值：230。
minG	RGB 域统计白点信息时，白点检测的G通道下限； 取值范围：[0x0, 0xff]； 默认值：3。
maxB	RGB 域统计白点信息时，白点检测的B通道上限； 取值范围：[0x0, 0xff]； 默认值：230。
minB	RGB 域统计白点信息时，白点检测的B通道下限； 取值范围：[0x0, 0xff]； 默认值：3。

成员名称	描述
maxY	RGB 域统计白点信息时，白点检测的Y通道上限； 取值范围：[0x0, 0xff]； 默认值：230。
minY	RGB 域统计白点信息时，白点检测的Y通道下限； 取值范围：[0x0, 0xff]； 默认值：3。
multiwindow_en	AWB 多窗口统计使能开关； true 使能，false 未使能； 默认值：false。
multiwindow	AWB 多窗口配置，最多支持4个窗口，multiwindow[i]= [h_offset,v_offset,h_size,v_size]，h：水平方向，v：垂直方向； 取值范围：[0x0, 0xfff]。

【注意】

- 更深入了解这些参数可参考《Rockchip_Color_Optimization_Guide》文档的以下内容，

  (a)"2 AWB/2.1功能描述" 章节内容及AWB流程图内容

  ```
  (b)"2 AWB/2.2关键参数/硬件的白点检测流程" 章节中图 AWB 白点检测流程图
  ```

2.2.3.5 rk_aiq_customAwb_single_hw_cfg_t （无用）

【说明】
定义环视模式下各个camea差异化的硬件配置，非环视无需关心

【定义】

```
typedef struct rk_aiq_customAwb_single_hw_cfg_t {
    unsigned short windowSet[4];
    bool multiwindow_en;
    unsigned short multiwindow[RK_AIQ_AWB_MULTIWINDOW_NUM_V201][4];
} rk_aiq_customAwb_single_hw_cfg_t;
```

【成员】

成员名称	描述
windowSet	同rk_aiq_customAwb_hw_cfg_t中windowSet含义
multiwindow_en	同rk_aiq_customAwb_hw_cfg_t中multiwindow_en含义
multiwindow	同rk_aiq_customAwb_hw_cfg_t中multiwindow含义

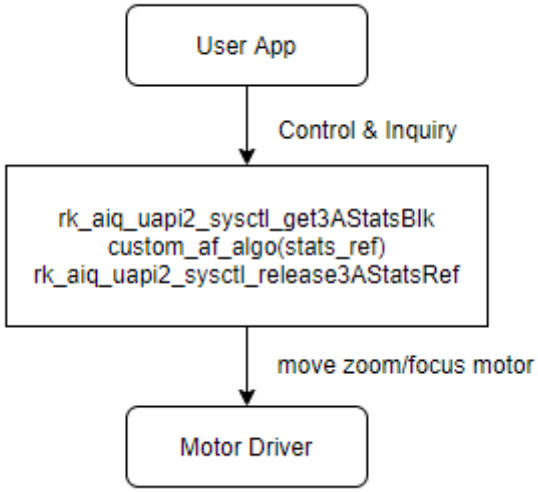
3. 2.3 开发用户AF算法

用户不使用RK AF算法库时，可以根据3A统计值开发AF算法，实现变倍对焦等功能。

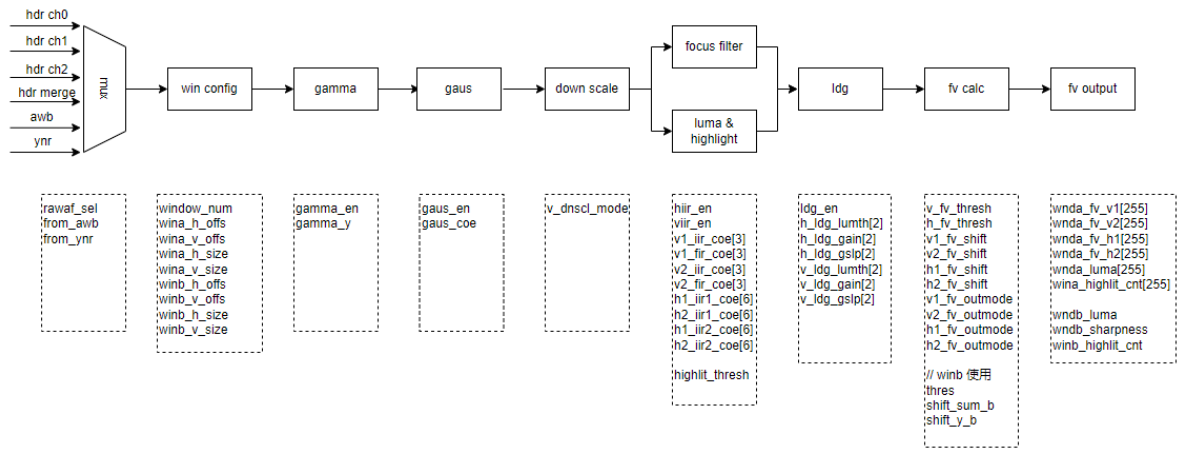
用户实现AF算法时，

1. 首先调用rk_aiq_user_api2_af_SetAttrib进行AF统计相关的配置；
2. 其次使用rk_aiq_uapi2_sysctl_get3AStatsBlk获取3A统计值，该API为阻塞式API，当有新3A统计值生成时，会立即返回；
3. 然后用户AF算法可以根据3A统计值进行相关运算，驱动变倍马达、对焦马达进行移动；
4. 最后需要调用rk_aiq_uapi2_sysctl_release3AStatsRef释放获取的3A统计值；

算法整体流程如下图所示。



3.1 2.3.1 AF统计模块



如果sensor输入HDR图像，AF模块可选择 HDR短/中/长曝的输出图像或HDR合成后的图像的一路数据，作为AF统计的输入数据。

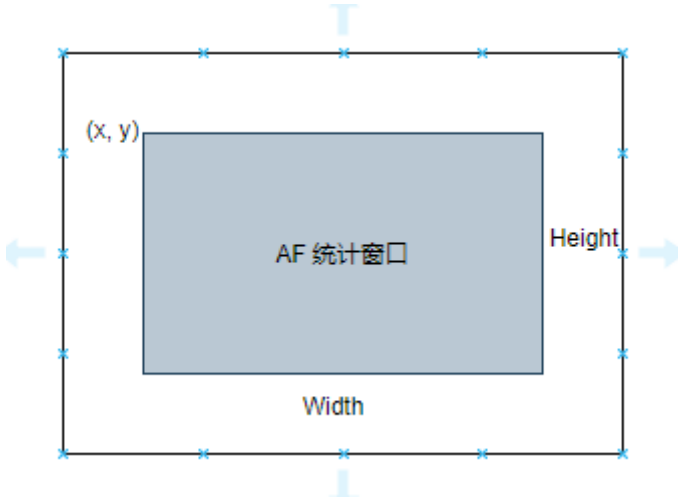
如果sensor输入Normal图像，AF模块可选择sensor的输入图像或debayer后图像作为AF统计的输入数据。

AF3.1支持主窗口A，它包含15*15子窗口，可以进行V1/V2/H1/H2四个滤波器的配置，输出V1/V2/H1/H2四个FV值、亮度值和高亮统计

值。独立窗口B处于废弃状态，建议不要使用。

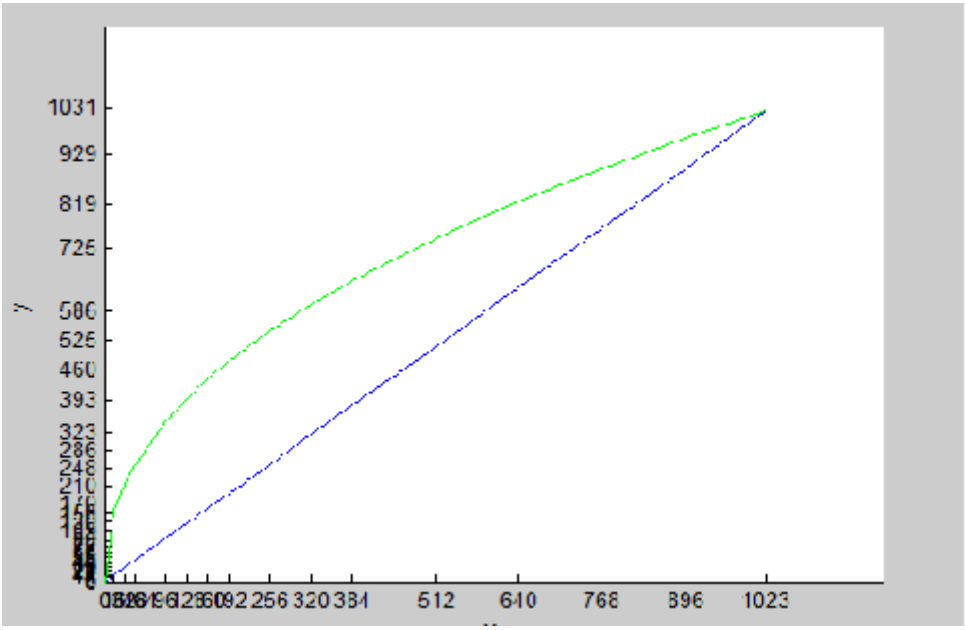
3.2 2.3.2 AF统计窗口配置

主窗口A支持矩形窗口配置。可配置矩形窗口左上角坐标和窗口宽高。



3.3 2.3.3 Gamma

Gamma将sensor输入raw图转换为人眼对自然亮度感知的程度，用于改善暗区对比度。
x坐标分段为0 to 1023:
16 16 16 16 32 32 32 32 64 64 64 128 128 128 128 128
y坐标取值范围为0 to 1023。



3.4 2.3.4 Gaus

可进行前置去噪处理，一般无需处理，按如下配置即可。

0 64 0

0 64 0

0 0 0

3.5 2.3.5 DownScale

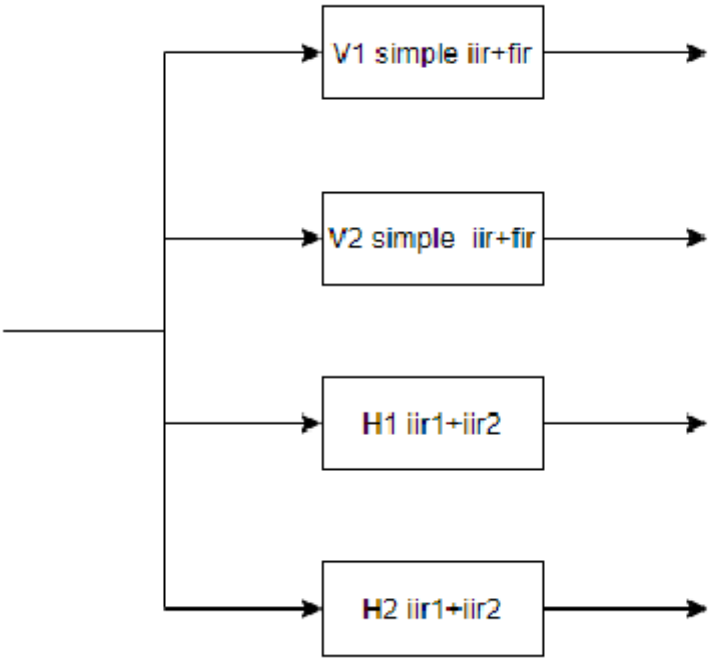
DownScale对输入的AF统计信号进行下采样处理，有助于支持更低频的滤波器频带。

3.6 2.3.6 Focus Filter

主窗口A提供V1/V2/H1/H2四个滤波器进行设置。

V1/V2/H1/H2四个滤波器的频带可以调整，使用滤波器设计工具，生成滤波器寄存器值。

常见的典型频带配置可采用[0.04n ~ 0.1n]，n为缩放比例，例如[0.01 ~ 0.025]、[0.02 ~ 0.05]、[0.04 ~ 0.1]、[0.08 ~ 0.2]等。



3.7 2.3.7 Luma/Highlight

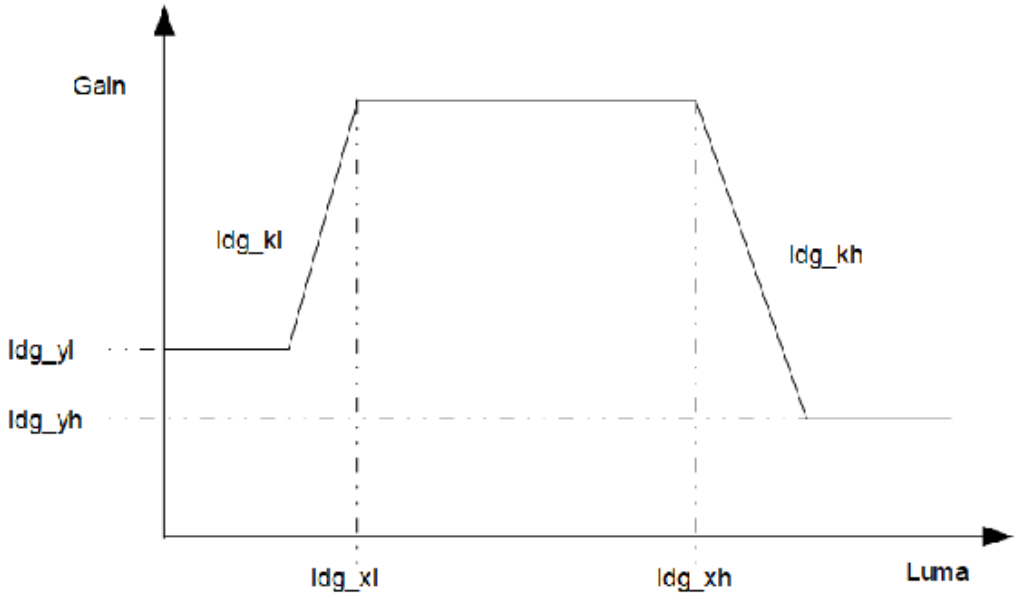
主窗口A提供亮度统计和高亮计数统计。

FV值容易受到光源的影响，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。

一般的解决方法是使用高亮计数器，聚焦模糊的时候因为光晕扩大，高亮点的个数会增加，清晰的时候，高亮点的个数会最小。

3.8 2.3.8 Luma Depend Gain

光源的影响也可以通过 LDG功能进行去除，根据像素亮度对FV值进行衰减，降低过亮点和过暗点处FV值。



亮度值在[ldg_xl, ldg_xh]之间时，Gain值输出为1，FV值不进行衰减；

亮度值在[0, ldg_xl]之间时，Gain值按照斜率ldg_kl进行衰减，Gain值最小为ldg_yl；

gain = 256 - ldg_kl*(ldg_xl - x)/256;
gain = max(gain, ldg_yl);

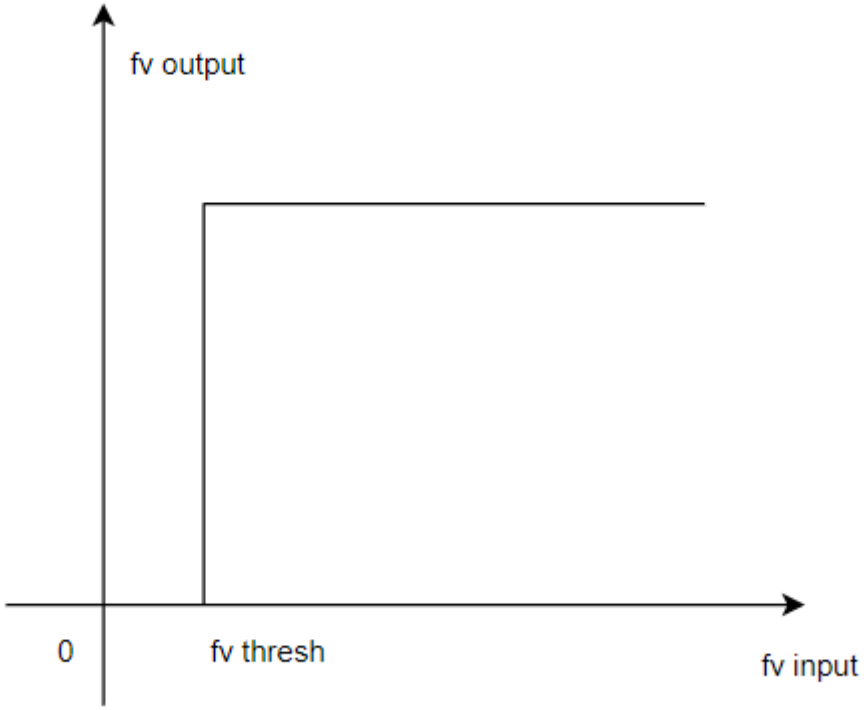
亮度值在[ldg_xh, 255]之间时，Gain值按照斜率ldg_kh进行衰减，Gain值最小为ldg_yh；

gain = 256 - ldg_kh*(x-ldg_xh)/256;
gain = max(gain, ldg_yh);

水平方向H1/H2共享一条LDG曲线，垂直方向V1/V2共享一条LDG曲线。

3.9 2.3.9 Fv threshold

在Fv值计算时，当Fv值小于Fv threshold阈值信息时，不计入最后的输出，可减少噪声的影响。



Fv threshold阈值是对滤波结果之后的LDG输出值做的阈值。

3.10 2.3.10 Fv Calc

Fv值支持绝对值模式和平方模式，平方模式将Fv值做平方运算，可增大清晰位置的FV值的比重。

硬件滤波器单像素的输出位宽为10bit，累积寄存器位宽31bit。

为了避免窗口统计累加时溢出，需要根据统计模式和窗口尺寸配置合适的shift移位寄存器，将像素FV值右移后再进行窗口累加。

目前用于shift的寄存器仅 3位，最多支持sum_shift=7，绝对值模式下支持的最大窗口为2^(31+7-10)=2^28，平方模式下FV值不超过

20bit，支持的最大窗口为2^(31+7-20)=2^18 (实际上，典型的带通配置下得到的FV值多数达不到上述门限，可以支持更大的窗口)。

3.11 2.3.11 Fv Output

主窗口A的输出包含15 * 15的v1/v2/h1/h2 Fv信息和15 * 15的luma/ highlight信息。

主窗口A的输出在图像上的分布如下图

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
2	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
3	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
4	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
5	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
6	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
7	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
8	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
9	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
10	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
11	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
12	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
13	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
14	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv
15	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv	Fv

3.12 2.3.12 最终FV值的计算

可将H1和V1配置为低通频带，用于粗搜索，将H2和V2 配置为高通频带，用于精搜索。

水平滤波输出H和垂直滤波输出V，可以用一定的权重进行加权。

FV = FvH * weight + FvV * (1-weight)

从各个block得到的FV值也可以根据需要按照一定的权重进行加权。

3.13 2.3.13 AF统计的配置

使用rk_aiq_user_api2_af_SetAttrib进行配置

```
XCamReturn
rk_aiq_user_api2_af_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_af_attrib_t
attr);
```

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	对焦的参数属性	输入

参数rk_aiq_af_attrib_t中的rk_aiq_af_algo_meas_v31_t说明如下：

```
typedef struct {
    unsigned char af_en;
    unsigned char rawaf_sel;
    unsigned char gamma_en;
    unsigned char gaus_en;
    unsigned char v1_fir_sel;
    unsigned char hiir_en;
    unsigned char viir_en;
    unsigned char v1_fv_outmode;    // 0 square, 1 absolute
```

```
    unsigned char v2_fv_outmode;    // 0 square, 1 absolute
    unsigned char h1_fv_outmode;    // 0 square, 1 absolute
    unsigned char h2_fv_outmode;    // 0 square, 1 absolute
    unsigned char ldg_en;
    unsigned char accu_8bit_mode;
    unsigned char ae_mode;
    unsigned char y_mode;
    unsigned char vldg_sel;
    unsigned char sobel_sel;
    unsigned char v_dnscl_mode;
    unsigned char from_awb;
    unsigned char from_ynr;
    unsigned char ae_config_use;
    unsigned char line_en[RKAIQ_RAWAF_LINE_NUM];
    unsigned char line_num[RKAIQ_RAWAF_LINE_NUM];

    unsigned char window_num;
    unsigned short wina_h_offs;
    unsigned short wina_v_offs;
    unsigned short wina_h_size;
    unsigned short wina_v_size;
    unsigned short winb_h_offs;
    unsigned short winb_v_offs;
    unsigned short winb_h_size;
    unsigned short winb_v_size;

    unsigned short gamma_y[RKAIQ_RAWAF_GAMMA_NUM];

    // [old version param]
    unsigned short thres;
    unsigned char shift_sum_a;
    unsigned char shift_sum_b;
    unsigned char shift_y_a;
    unsigned char shift_y_b;

    char gaus_coe[9];

    /**********[Vertical IIR (v1 & v2)]************/
    short v1_iir_coe[3];
    short v1_fir_coe[3];
    short v2_iir_coe[3];
    short v2_fir_coe[3];

    /**********[Horizontal IIR (h1 & h2)]************/
    short h1_iir1_coe[6];
    short h2_iir1_coe[6];
    short h1_iir2_coe[6];
    short h2_iir2_coe[6];

    /**********[Focus value statistic param]**********/
    // level depended gain
    // input8 lumi, output8bit gain
    unsigned char h_ldg_lumth[2];    //luminance thresh
    unsigned char h_ldg_gain[2];     //gain for [minLum,maxLum]
    unsigned short h_ldg_gslp[2];    //[slope_low,-slope_high]
```

```
    unsigned char v_ldg_lumth[2];
    unsigned char v_ldg_gain[2];
    unsigned short v_ldg_gslp[2];

    // coring
    unsigned short v_fv_thresh;
    unsigned short h_fv_thresh;

    // left shift, more needed if outmode=square
    unsigned char v1_fv_shift; //only for sel1
    unsigned char v2_fv_shift;
    unsigned char h1_fv_shift;
    unsigned char h2_fv_shift;

    /**********[High light]**********/
    unsigned short highlit_thresh;
} rk_aiq_af_algo_meas_v31_t;
```

成员名称	描述
af_en	是否使能AF 信息统计，0为关闭，1为打开
rawaf_sel	选择AF信息统计的通道，取值范围0-3，对应hdr模式的长/中/短/合成帧通道选择，一般AF选择中帧通道，非hdr模式设置为0，hdr模式设置为1
gamma_en	gamma模块使能开关，0为关闭，1为打开
gaus_en	需固定设置为1
v1_fir_sel	需固定设置为1
hiir_en	H1/H2通道使能开关，0为关闭，1为打开
viir_en	V1/V2通道使能开关，0为关闭，1为打开。需要注意gamma_en打开时，viir_en必须设置为1
v1_fv_outmode	V1通道FV输出模式选择，0为平方模式，1为绝对值模式
v2_fv_outmode	V2通道FV输出模式选择，0为平方模式，1为绝对值模式
h1_fv_outmode	H1通道FV输出模式选择，0为平方模式，1为绝对值模式
h2_fv_outmode	H2通道FV输出模式选择，0为平方模式，1为绝对值模式
ldg_en	LDG功能使能开关，0为关闭，1为打开
accu_8bit_mode	需固定设置为1
ae_mode	当ae_mode设置为1，RAWAF使能15x15亮度均值统计，复用了RAWAE_BIG模块的逻辑
y_mode	需固定设置为0
vldg_sel	需固定设置为0
sobel_sel	需固定设置为0
v_dnscl_mode	宽窄带模式选择，按照AF滤波器系数生成工具的输出进行设置
from_awb	AF统计输入从AWB获取
from_ynr	AF统计输入从YNR获取
ae_config_use	固定配置为0
line_en	目前暂未生效
line_num	目前暂未生效
window_num	生效的窗口数，window_num为1时，wina(主窗口)生效；window_num为2时，wina(主窗口)和winb(独立窗口)生效
wina_h_offs	wina(主窗口)左上角第一个像素的水平坐标，该值必须大于等于2
wina_v_offs	wina(主窗口)左上角第一个像素的垂直坐标，该值必须大于等于1

成员名称	描述
wina_h_size	wina(主窗口)的窗口宽度，该值必须小于图像宽度-2-wina_h_offs；同时该值必须为15的倍数；
wina_v_size	wina(主窗口)的窗口高度，该值必须小于图像高度-2-wina_v_offs；同时该值必须为15的倍数；
winb_h_offs	winb(独立窗口)左上角第一个像素的水平坐标，该值必须大于等于2
winb_v_offs	winb(独立窗口)左上角第一个像素的垂直坐标，该值必须大于等于1
winb_h_size	winb(独立窗口)的窗口宽度，该值必须小于图像宽度-2-wina_h_offs
winb_v_size	winb(独立窗口)的窗口高度，该值必须小于图像高度-2-wina_v_offs
gamma_y	gamma table的y值，取值范围0-1023；x坐标分段为0 to 1023: 16 16 16 16 32 32 32 32 64 64 64 128 128 128 128 128
thres	win b(独立窗口)的AF统计阈值，计算出的fv值小于该值时，fv值改为0，可减少噪声的影响，取值范围为0-0xFFFF
shift_sum_a	目前无法使用，固定设置为0即可
shift_sum_b	win b(独立窗口)的fv值的shit bit值，会按照该值将fv值向右移位，避免得到的fv值溢出，取值范围为0-7
shift_y_a	目前无法使用，固定设置为0即可
shift_y_b	win b(独立窗口)的luma值的shit bit值，会按照该值将luma值向右移位，避免得到的luma值溢出，取值范围为0-7
gaus_coe	3*3的预滤波
v1_iir_coe[3]	用于V1通道的1X3 IIR系数，按照AF滤波器系数生成工具的输出进行设置
v1_fir_coe[3]	用于V1通道的1x3 FIR系数，按照AF滤波器系数生成工具的输出进行设置
v2_iir_coe[3]	用于V2通道的1x3 IIR系数，按照AF滤波器系数生成工具的输出进行设置
v2_fir_coe[3]	用于V2通道的1x3 FIR系数，按照AF滤波器系数生成工具的输出进行设置
h1_iir1_coe[6]	用于H1通道的1X6 IIR1系数，按照AF滤波器系数生成工具的输出进行设置
h2_iir1_coe[6]	用于H2通道的1X6 IIR1系数，按照AF滤波器系数生成工具的输出进行设置
h1_iir2_coe[6]	用于H1通道的1X6 IIR2系数，按照AF滤波器系数生成工具的输出进行设置
h2_iir2_coe[6]	用于H2通道的1X6 IIR2系数，按照AF滤波器系数生成工具的输出进行设置
h_ldg_lumth[2]	用于H1/H2通道的ldg模块的亮度阈值系数，0为左边暗区设置，1为右边高亮区设置，取值范围为0~255
h_ldg_gain[2]	用于H1/H2通道的ldg模块的最小gain值，0为左边暗区设置，1为右边高亮区设置，取值范围为0~255
h_ldg_gslp[2]	用于H1/H2通道的ldg模块的斜率系数，0为左边暗区设置，1为右边高亮区设置，取值范围为0~65535

成员名称	描述
v_ldg_lumth[2]	用于V1/V2通道的ldg模块的亮度阈值系数，0为左边暗区设置，1为右边高亮区设置，取值范围为0~255
v_ldg_gain[2]	用于V1/V2通道的ldg模块的最小gain值，0为左边暗区设置，1为右边高亮区设置，取值范围为0~255
v_ldg_gslp[2]	用于V1/V2通道的ldg模块的最小gain值，0为左边暗区设置，1为右边高亮区设置，取值范围为0~255
v_fv_thresh	用于V1/V2通道的AF统计阈值，计算出的fv值小于该值时，fv值改为0，可减少噪声的影响，取值范围为0-0x0FFF
h_fv_thresh	用于H1/H2通道的AF统计阈值，计算出的fv值小于该值时，fv值改为0，可减少噪声的影响，取值范围为0-0x0FFF
v1_fv_shift	用于V1通道的fv值的shit bit值，会按照该值将fv值向右移位，避免得到的fv值溢出，取值范围为0-7
v2_fv_shift	用于V2通道的fv值的shit bit值，会按照该值将fv值向右移位，避免得到的fv值溢出，取值范围为0-7
h1_fv_shift	用于H1通道的fv值的shit bit值，会按照该值将fv值向右移位，避免得到的fv值溢出，取值范围为0-7
h2_fv_shift	用于H2通道的fv值的shit bit值，会按照该值将fv值向右移位，避免得到的fv值溢出，取值范围为0-7
highlit_thresh	表示高亮统计的阈值，当高于该值则认为是高亮点，纳入统计，只累加每个区域的高亮点的个数，取值范围为0-0x0FFF

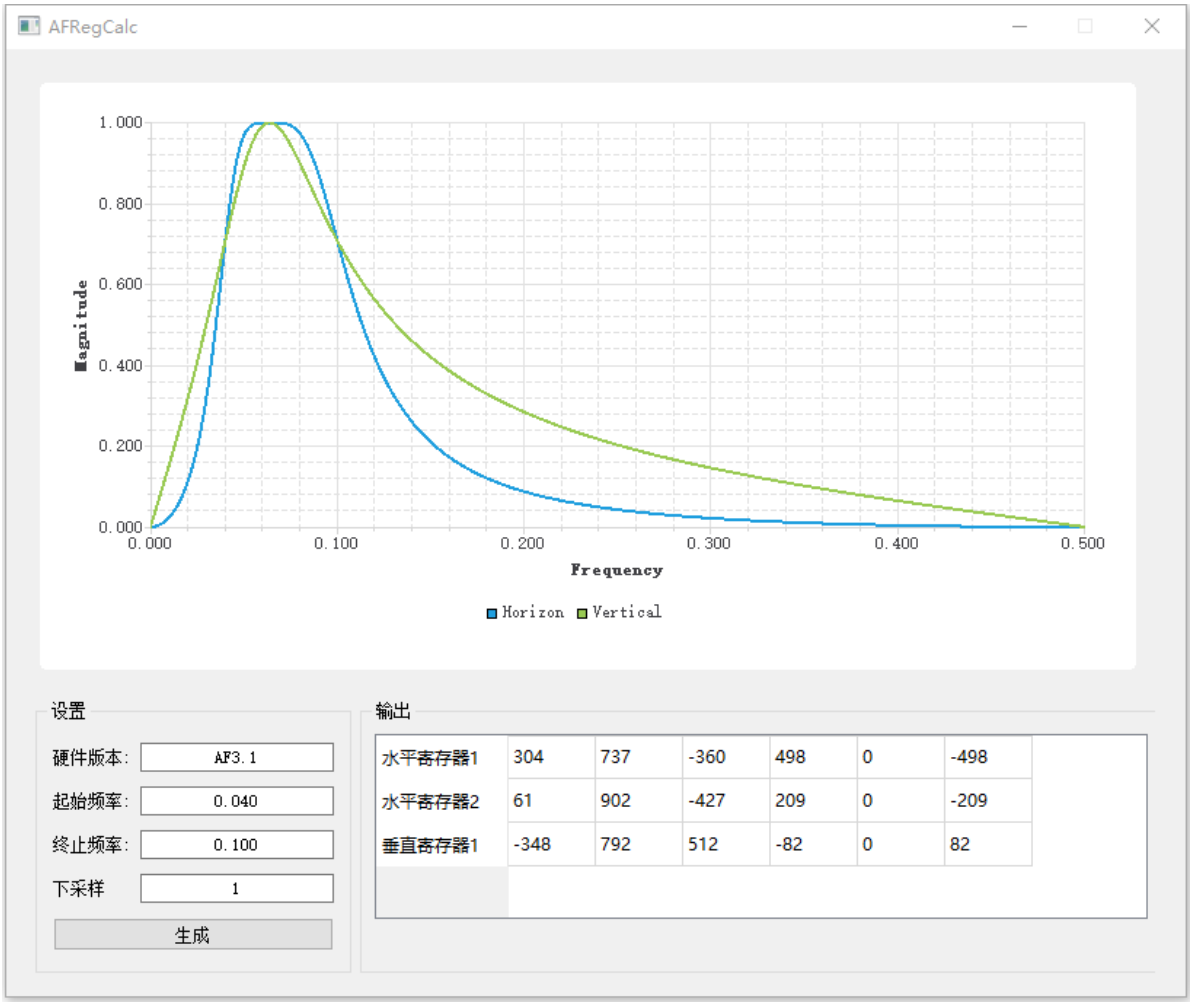
3.14 2.3.14 AF统计值的获取

详见《Rockchip_Development_Guide_ISP32》“统计信息”章节

rk_aiq_uapi2_sysctl_get3AStatsBlk () / rk_aiq_uapi2_sysctl_release3AStatsRef()

AF统计结果的相关结构体为rk_aiq_isp_af_stats_v3x_t

3.15 2.3.15 滤波器设计工具的使用



ISP32平台上硬件版本应输入AF3.1。

起始频率和终止频率的输入范围为0.005～0.490，但由于实际硬件限制，输入范围比理论输入范围要小一些，具体参考工具的输出。

下采样处理有助于支持更低频的滤波器频带，可输入1或2，工具根据起始频率和终止频率的输入可能会对该值进行修改。

点击生成按钮后，输出框会显示滤波器寄存器值，同时上方会显示相应的滤波器响应曲线。

4. 2.4 参考代码样例

客户3A算法实现参考代码样例，可以参考：

目录： AIQ根目录/rkisp_demo/demo/

```
|-------ae_algo_demo            // AE算法参考代码
|-------awb_algo_demo           // AWB算法参考代码
|-------af_algo_demo            // AF算法参考代码
```