Rockchip Developement Guide 3A ISP39

文件标识：RK-KF-GX-612

发布版本：V1.0.0

日期：2024-7-29

文件密级：□绝密 □秘密 □内部资料 ■公开

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
RK3576	Linux 6.10

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
RK3576	Y	N	N	Y

修订记录

版本号	作者	修改日期	修改说明
v1.0.0	ISP	2024-7-29	ISP3A 开发指南初版

目录

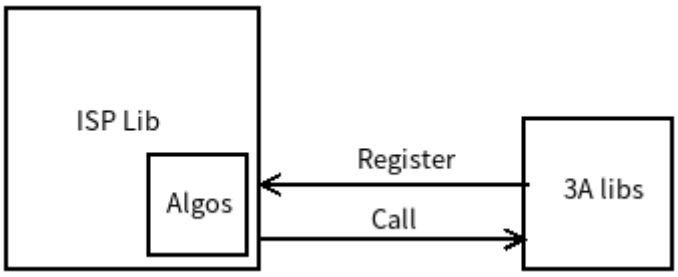
1 概述

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 设计思路

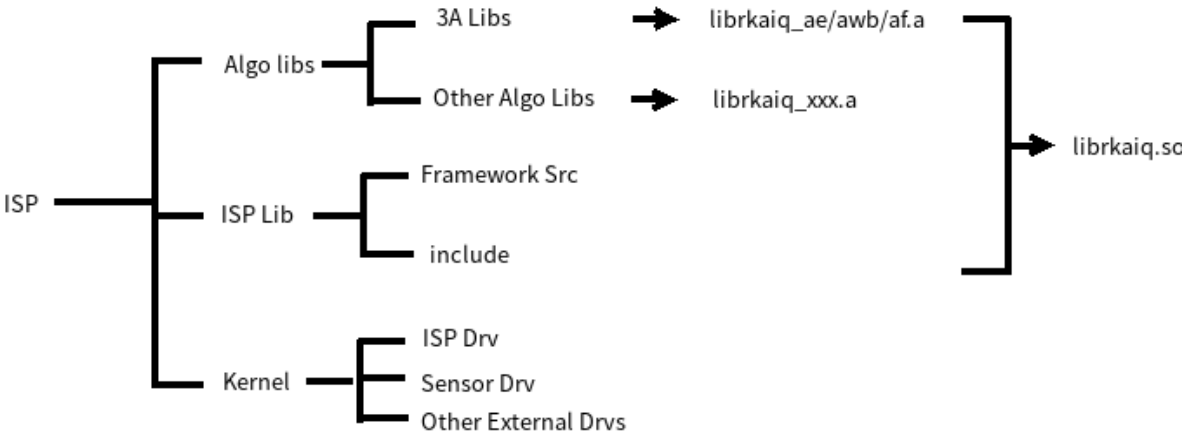
基本设计思路如下图所示：



- 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

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。

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。

1.4 软件流程

1.4.1 基础流程

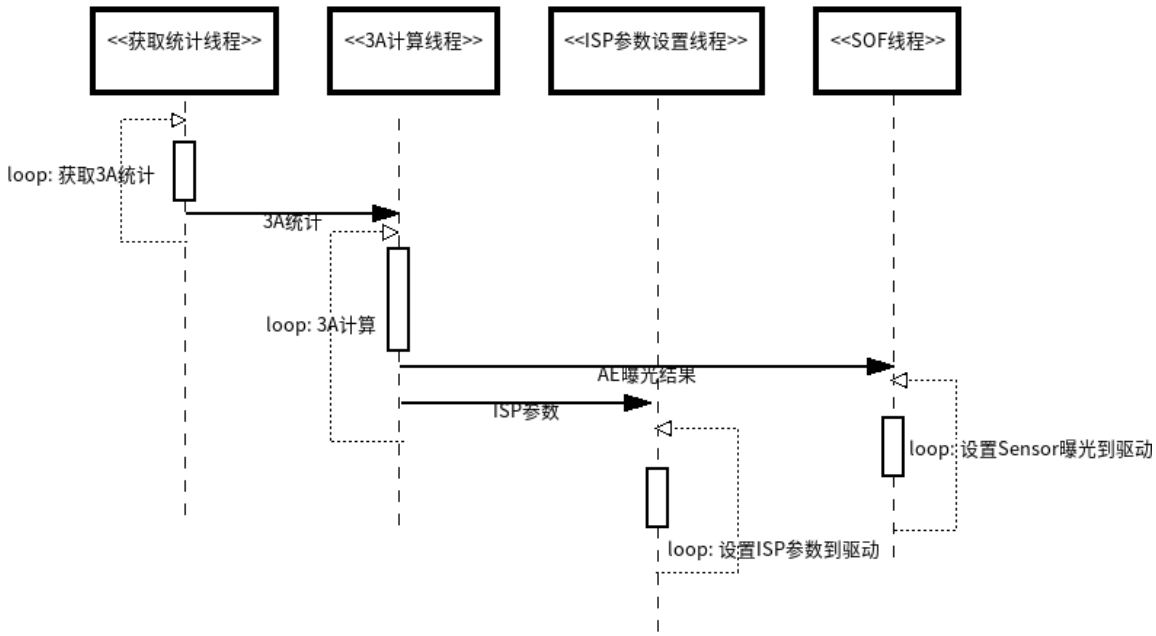
RK 3A 算法不需要用户显示注册，AIQ 框架内部已隐式注册。自定义 3A 算法注册，以自定义 AE 算法注册为例，示例伪代码如下：

```
// 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_pfnAe_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_ae_register((const rk_aiq_sys_ctx_t*)(ctx->camgroup_ctx), &cbs);

// 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_ae_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);
```

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

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_uapi2_ae_register接口，使能调用rk_aiq_uapi2_ae_enable接口， 注销调用rk_aiq_uapi2_ae_unRegister接口

rk_aiq_uapi2_ae_register

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_ae_register(const rk_aiq_sys_ctx_t* ctx, rk_aiq_pfnAe_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

rk_aiq_uapi2_ae_enable

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_ae_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

rk_aiq_uapi2_ae_unRegister

【描述】
注册AE算法库

【语法】

```
XCamReturn
rk_aiq_uapi2_ae_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

回调函数

用户需要在自开发定制的AE库中实现以下回调函数：

```
rk_aiq_pfnAe_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

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

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 ae_pfnAe_info_t* pstAeInfo,
                      ae_pfnAe_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

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

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

输入数据参数

【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 ae_rkExp_s {
  RkAiqExpParamComb_t linear_exp;
  RkAiqExpParamComb_t hdr_exp[RK_AIQ_MAX_HDR_FRAME];
} ae_rkExp_t;
typedef struct ae_pfnAe_info_s
{
  // 0) common
  uint16_t entValidBit;

  // 1) single hw stats
  aeStats_entityStats_t *pHwEnt0;
  aeStats_entityStats_t *pHwEnt1;
  aeStats_entityStats_t *pHwEnt2;
  aeStats_entityStats_t *pHwEnt3;
  aeStats_entityStats_t *pSwCoWkEnt03;

  // 2) single cis exposure
  ae_rkExp_t cisRkExp;
```

```
    struct ae_pfnAe_info_s* next; // for surround view(multiple cams)
} ae_pfnAe_info_t;
```

【成员】

成员名称	描述
entValidBit	多个ae硬件统计模块的有效bit：bit位0为1，则表示pHwEnt0存在有效统计；bit位1为1，则表示pHwEnt1存在有效统计；bit位2为1，则表示pHwEnt2存在有效统计；bit位3为1，则表示pHwEnt3存在有效统计；bit位8为1，则表示pSwCoWkEnt03存在有效统计
pHwEnt0~3	ae硬件统计模块，指针指向每个硬件统计模块对应的统计值。其有效性由entValidBit决定。
pSwCoWkEnt03	ae软件统计模块，该模块一般仅在3576平台使用built-in-Hdr时有效，实际有效性由entValidBit决定。
cisRkExp	输入曝光，与硬件统计信息同步。包含线性模式下的曝光参数（linear_exp）；Hdr模式下的曝光参数（hdr_exp），3576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。

【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aeStats_entityStats_t，曝光信息数据类型为RkAiqExpParamComb_t，数据类型说明详见《Rockchip_Development_Guide_ISP39》文档
- RK_AIQ_MAX_HDR_FRAME表示RK平台至多支持3帧HDR，针对3576平台仅支持2帧HDR

输出算法结果参数

【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

ae_pfnAe_results_t

【定义】

```
#define RK_AIQ_MAX_HDR_FRAME (3)

typedef struct ae_pfnAe_single_results_s
{
    ae_rkExp_t cisRkExp;
    ae_i2cExp_t cisI2cExp;
    ae_statsCfg_t statsCfg;

    struct ae_pfnAe_single_results_s* next; // for surround view(multiple cams)
} ae_pfnAe_single_results_t;

typedef struct ae_pfnAe_results_s
{
```

```
    // 0) common
    bool isConverged;
    bool isLongFrmMode;
    bool isRkExpValid;
    uint32_t frmLengthLines; //vts

    // 1) single result
    ae_rkExp_t cisRkExp; // if isRkExpValid = false ,also need float real exposure value
    ae_i2cExp_t cisI2cExp;

    RkAiqIrisParamComb_t iris;
    ae_statsCfg_t statsCfg;

    struct ae_pfnAe_single_results_s* next; // for surround view(multiple cams)
} ae_pfnAe_results_t;
```

【成员】

成员名称	子成员	描述
isConverged		收敛状态位，true代表收敛，false代表未收敛。该参数为后级模块所用
isLongFrmMode		长帧模式使能位，true代表开启长帧模式，false代表关闭长帧模式。仅在HDR曝光时有效，后级drc模块需要该参数。
isRkExpValid		是否使用RK格式曝光，true代表使用RK格式曝光，cisRkExp有效，仅需填写曝光实际浮点值，寄存器值无需填写，cisI2cExp无需填写；false代表使用常规i2c格式，cisI2cExp有效，需要完整填写，cisRkExp仅需填写曝光实际浮点值。
frmLengthLines		sensor的vts值，与帧率设置有关
cisRkExp	linear_exp hdr_exp	RK格式曝光参数。非HDR模式时，linear_exp有效，不论isRkExpValid为何值，第三方算法均需要填写linear_exp中的曝光实际值（exp_real_params），该实际值为后续模块所用；Hdr模式时，hdr_exp有效，不论isRkExpValid为何值，第三方算法均需要填写linear_exp中的曝光实际值（exp_real_params），该实际值为后续模块所用，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。
cisI2cExp	bValid nNumRegs pRegAddr pAddrByteNum pRegValue pValueByteNum pDelayFrames	i2c寄存器值参数 当isRkExpValid为false时，使用cisI2cExp参数进行寄存器值设置，详见下文
Iris	PIris DCIris HDCIris	光圈设置参数，包含P光圈、DC光圈、霍尔DC光圈设置参数
statsCfg	entValidBit hwEnt0~3 pSwCoWkEnt03	硬件统计配置参数，详见下文
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ae_pfnAe_results_t中的成员，而后所有camera皆使用ae_pfnAe_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。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

ae_pfnAe_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，ae_pfnAe_results_t与ae_pfnAe_single_results_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。

- 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isRkExpValid值为true；第三方格式需要用户设置所需寄存器值及对应地址，要求isRkExpValid值为false。

ae_statsCfg_t

【定义】

```
typedef struct ae_win_s {
  uint16_t hw_aeCfg_win_x;
  uint16_t hw_aeCfg_win_y;
  uint16_t hw_aeCfg_win_width;
  uint16_t hw_aeCfg_win_height;
} ae_win_t;

typedef struct ae_hist_s {
  uint8_t hw_aeCfg_zone_wgt[AESTATS_ZONE_15x15_NUM];
} ae_hist_t;

typedef struct ae_entity_s {
  ae_win_t mainWin;
  ae_hist_t hist;
} ae_entity_t;

typedef struct ae_statsCfg_s {
  uint16_t entValidBit;
  ae_entity_t hwEnt0;
  ae_entity_t hwEnt1;
  ae_entity_t hwEnt2;
  ae_entity_t hwEnt3;

  ae_entity_t *pSwCoWkEnt03;
} ae_statsCfg_t;
```

【成员】

成员名称	子成员名称	描述
entValidBit		多个ae硬件统计模块的有效bit：bit位0为1，则表示pHwEnt0存在有效统计；bit位1为1，则表示pHwEnt1存在有效统计；bit位2为1，则表示pHwEnt2存在有效统计；bit位3为1，则表示pHwEnt3存在有效统计；bit位8为1，则表示pSwCoWkEnt03存在有效统计。在配置硬件统计参数时，需要配置entValidBit，用于表示使用哪一个硬件统计模块。对于3576平台，建议配置AE_BIT(AE_HwEnt0_Bit) | AE_BIT(AE_HwEnt3_Bit);
pHwEnt0~3		ae硬件统计模块，指针指向每个硬件统计模块对应的配置参数。其有效性由entValidBit决定。
pSwCoWkEnt03		ae软件统计模块，该模块一般仅在3576平台使用built-in-Hdr时有效，实际有效性由entValidBit决定。
mainWin	hw_aeCfg_win_x	统计窗口左上角的横向相对偏移值。要求hw_aeCfg_win_x需要小于等于横向分辨率。
	hw_aeCfg_win_y	统计窗口左上角的横向相对偏移值。要求hw_aeCfg_win_y需要小于等于纵向分辨率。
	hw_aeCfg_win_width	统计窗口的宽度。要求宽度不可小于240.
	hw_aeCfg_win_height	统计窗口的高度。要求高度不可小于60.
hist	hw_aeCfg_zone_wgt	统计区域权重，统计区域分块为15X15，支持设置15X15个权重值。

ae_i2cExp_t

【定义】

```
typedef struct ae_i2cExp_s {
  unsigned int   nNumRegs;
  unsigned int*  pRegAddr;
  unsigned int*  pAddrByteNum;
  unsigned int*  pRegValue;
  unsigned int*  pValueByteNum;
  unsigned int*  pDelayFrames;
} ae_i2cExp_t;
```

【成员】

成员名称	描述
nNumRegs	需要设置的i2c寄存器个数
pRegAddr	需要设置的i2c寄存器值地址
pAddrByteNum	需要设置的i2c寄存器值地址所占bit数
pRegValue	需要设置的i2c寄存器值
pValueByteNum	需要设置的i2c寄存器值所占bit数
pDelayFrames	需要设置的i2c寄存器值的延迟帧数

ae_rkExp_t

【定义】

```
typedef struct ae_rkExp_s {
    RkAiqExpParamComb_t linear_exp;
    RkAiqExpParamComb_t hdr_exp[RK_AIQ_MAX_HDR_FRAME];
} ae_rkExp_t;
```

```
typedef struct {
    RkAiqExpRealParam_t exp_real_params; //real value
    RkAiqExpSensorParam_t exp_sensor_params;//reg value
} RkAiqExpParamComb_t;
typedef struct RkAiqExpRealParam_s {
    float integration_time;
    float analog_gain;
    float digital_gain;
    float isp_dgain;
    int   iso;
    int   longfrm_mode;
} RkAiqExpRealParam_t;
```

【成员】

成员名称	描述
integration_time	CIS曝光时间，单位为秒
analog_gain	CIS模拟增益，单位为倍数。3576平台，此处为CIS的totalgain，即analog_gain*digital_gain总值填在此处。
digital_gain	CIS数字增益，单位为倍数。3576平台，该值暂无效，固定填1.
isp_dgain	ISP数字增益，单位为倍数。
iso	该值暂无效，无需填写
longfrm_mode	该值暂无效，无需填写

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_ISP30》"统计信息 / 数据类型 / rk_aiq_isp_awb_stats2_v3x_t " 章节

2.2.1 API

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

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

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 数据类型

向 ISP 库注册的回调函数

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

统计信息

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。

运算结果

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

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

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。

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

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

2.3 开发用户AF算法

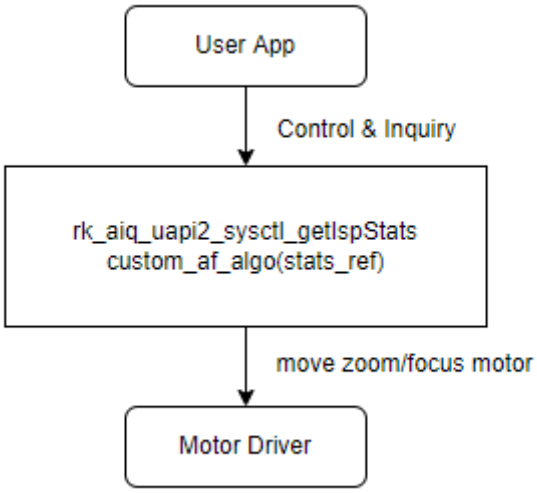
用户不使用RK AF算法库时，可以根据3A统计值开发AF算法，实现变倍对焦等功能。

用户实现AF算法时，

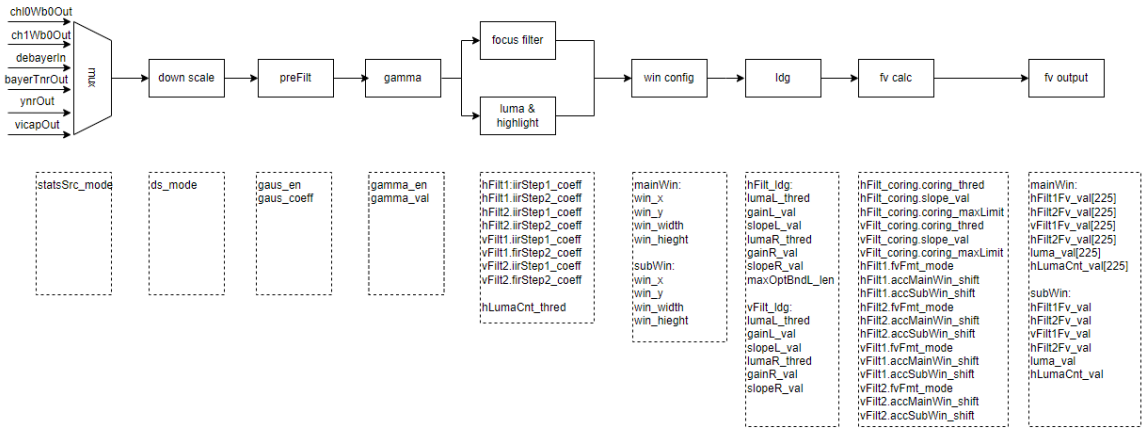
1. 首先调用rk_aiq_user_api2_af_SetAttrib进行AF统计相关的配置；
2. 其次使用rk_aiq_uapi2_sysctl_getIspStats获取3A统计值；

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；

算法整体流程如下图所示。



2.3.1 AF统计模块



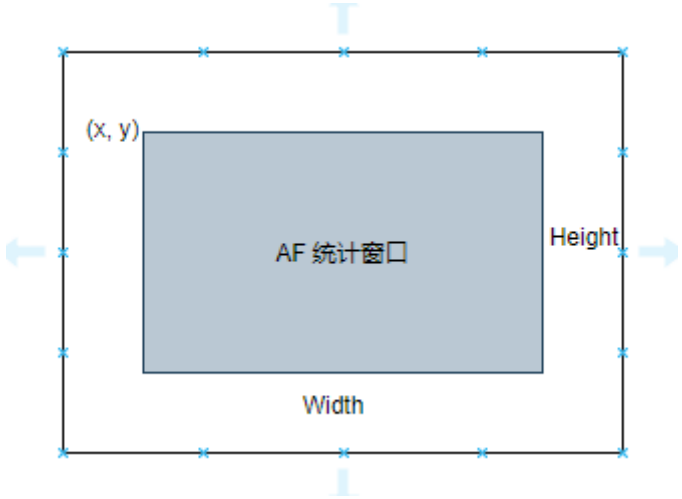
AF模块可选择 ch0Wb0Out/ch1Wb0Out/debayerIn/bayTnrOut/ynrOut/vicapOut的图像数据，作为AF统计的输入数据。

AF3.3支持主窗口A，它包含15*15子窗口，可以进行V1/H1/V2/H2四个滤波器的配置，输出V1/H1/V2/H2四个FV值、亮度值和高亮统计。

值。独立窗口B共享主窗口A的V1/H1/V2/H2滤波器的配置，输出V1/H1/V2/H2四个FV值、亮度值和高亮统计。

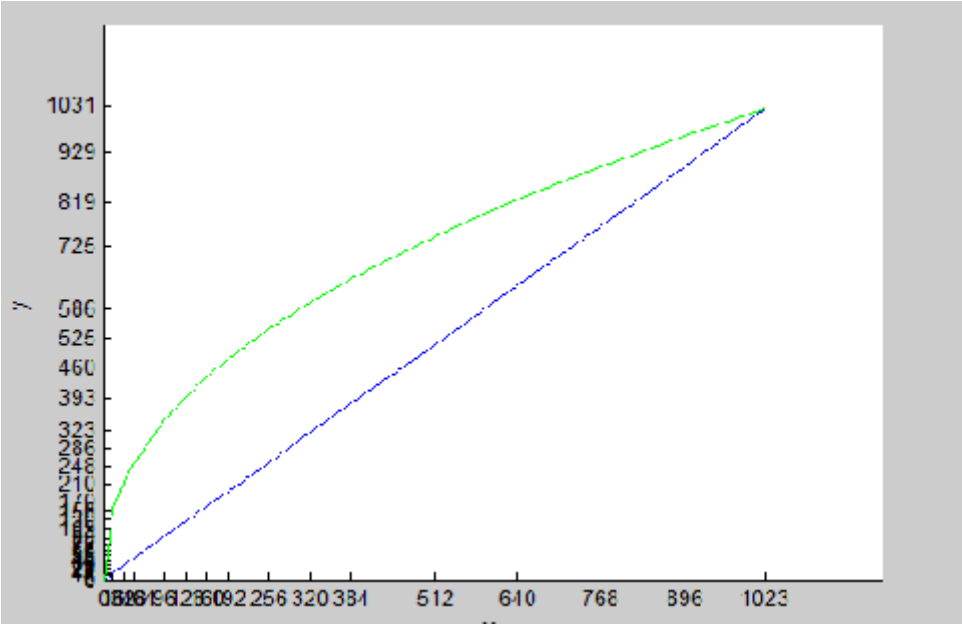
2.3.2 AF统计窗口配置

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。



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。



2.3.4 Gaus

可进行前置去噪处理，一般按如下配置即可。

0 64 0

0 64 0

0 0 0

2.3.5 DownScale

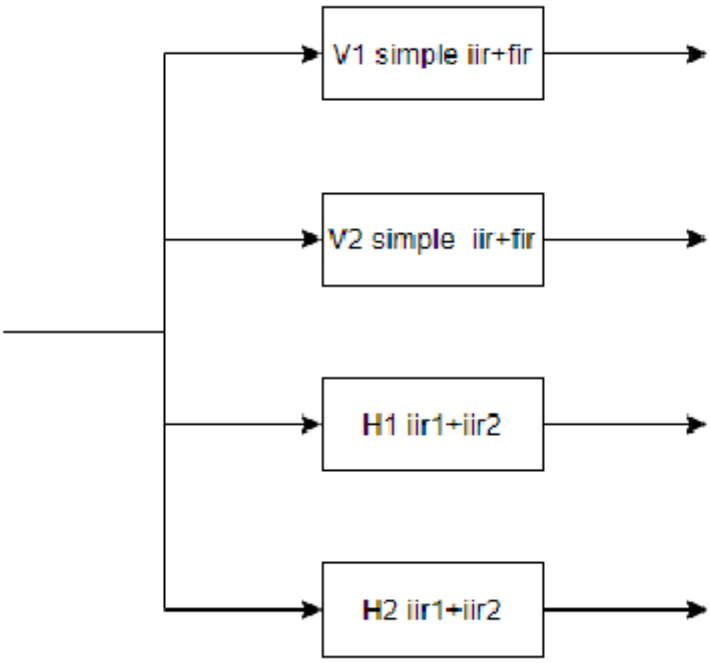
DownScale对输入的AF统计信号进行下采样处理，有助于支持更低频的滤波器频带。

2.3.6 Focus Filter

主窗口A提供V1/H1/V2/H2四个滤波器进行设置。

V1/H1/V2/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。



2.3.7 Luma/Highlight

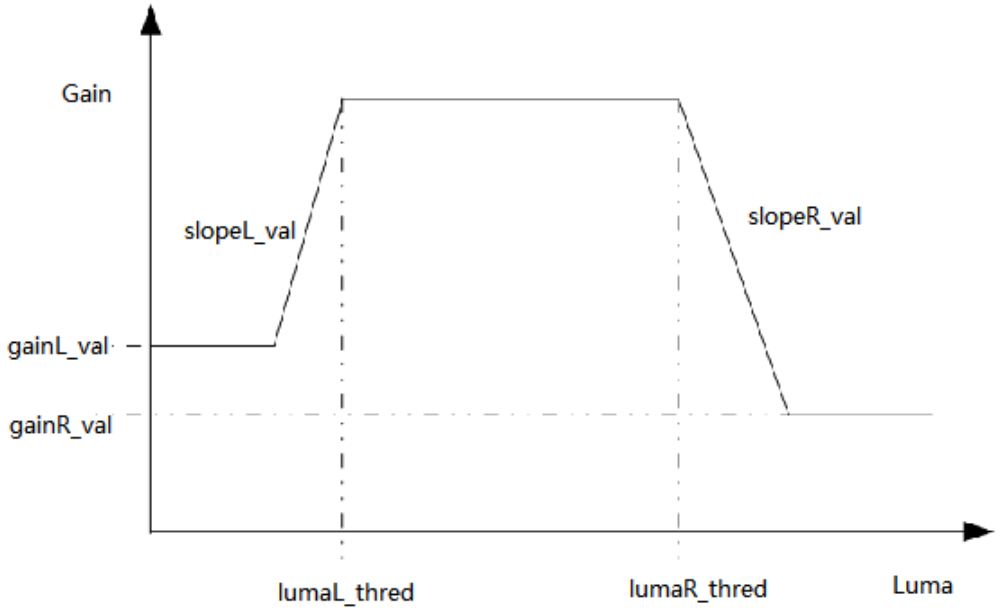
主窗口A提供亮度统计和高亮计数统计。

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。

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[lumaL_thred, lumaR_thred]之间时，gain值输出为1，FV值不进行衰减；

亮度值在[0, lumaL_thred]之间时，gain值按照斜率slopeL_val进行衰减，gain值最小为gainL_val；

gain = 256 - slopeL_val*(lumaL_thred - x)/256;
gain = max(gain, gainL_val);

亮度值在[lumaR_thred, 255]之间时，gain值按照斜率slopeR_val进行衰减，gain值最小为gainR_val；

gain = 256 - slopeR_val*(x-lumaR_thred)/256;
gain = max(gain, gainR_val);

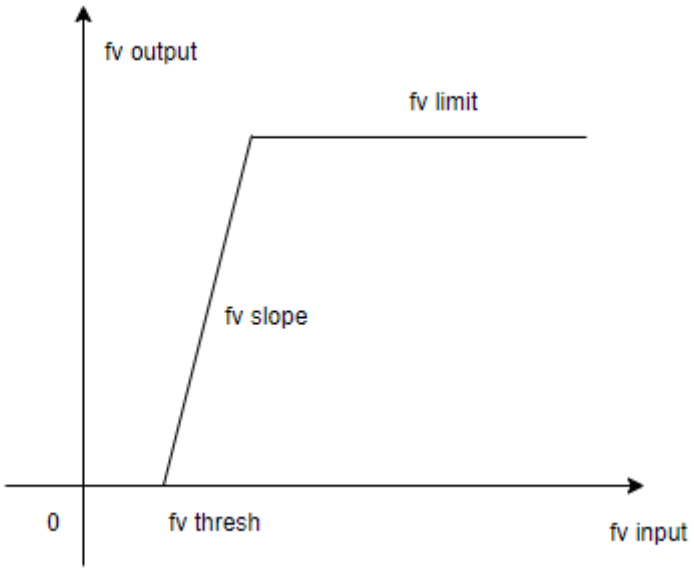
水平方向H1/H2共享一条LDG曲线，垂直方向V1/V2共享一条LDG曲线。

2.3.9 Fv Coring

当Fv值小于Fv thresh阈值信息时，输出Fv clip为0，不计入最后的输出。

当Fv值大于Fv thresh阈值信息时，输出Fv会乘上fv slope进行输出。

当Fv值大于Fv thresh阈值信息时，且输出Fv乘上fv slope之后大于fv limit时，会将输出fv限制为fv limit。



Fv Coring是对滤波结果之后的LDG输出值做的阈值。

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。

Fv值累计计算时可选择统计块内每行Fv的最大值模式和每行Fv的累加模式。

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。

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

20bit，支持的最大子窗口为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，可以支持更大的子窗口)。

独立窗口用于shift的寄存器为4位，最多支持sum_shift=15，绝对值模式下支持的最大子窗口为2^(31+15-10)=2^36，平方模式下FV值不超过

20bit，支持的最大子窗口为2^(31+15-20)=2^26 (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，可以支持更大的子窗口)。

2.3.11 Fv Output

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

独立窗口B的输出包含1 * 1的v1/h1/v2/h2 Fv信息和1 * 1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

2.3.12 最终FV值的计算

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。

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);
```

详见《Rockchip_Development_Guide_ISP39》AF章节

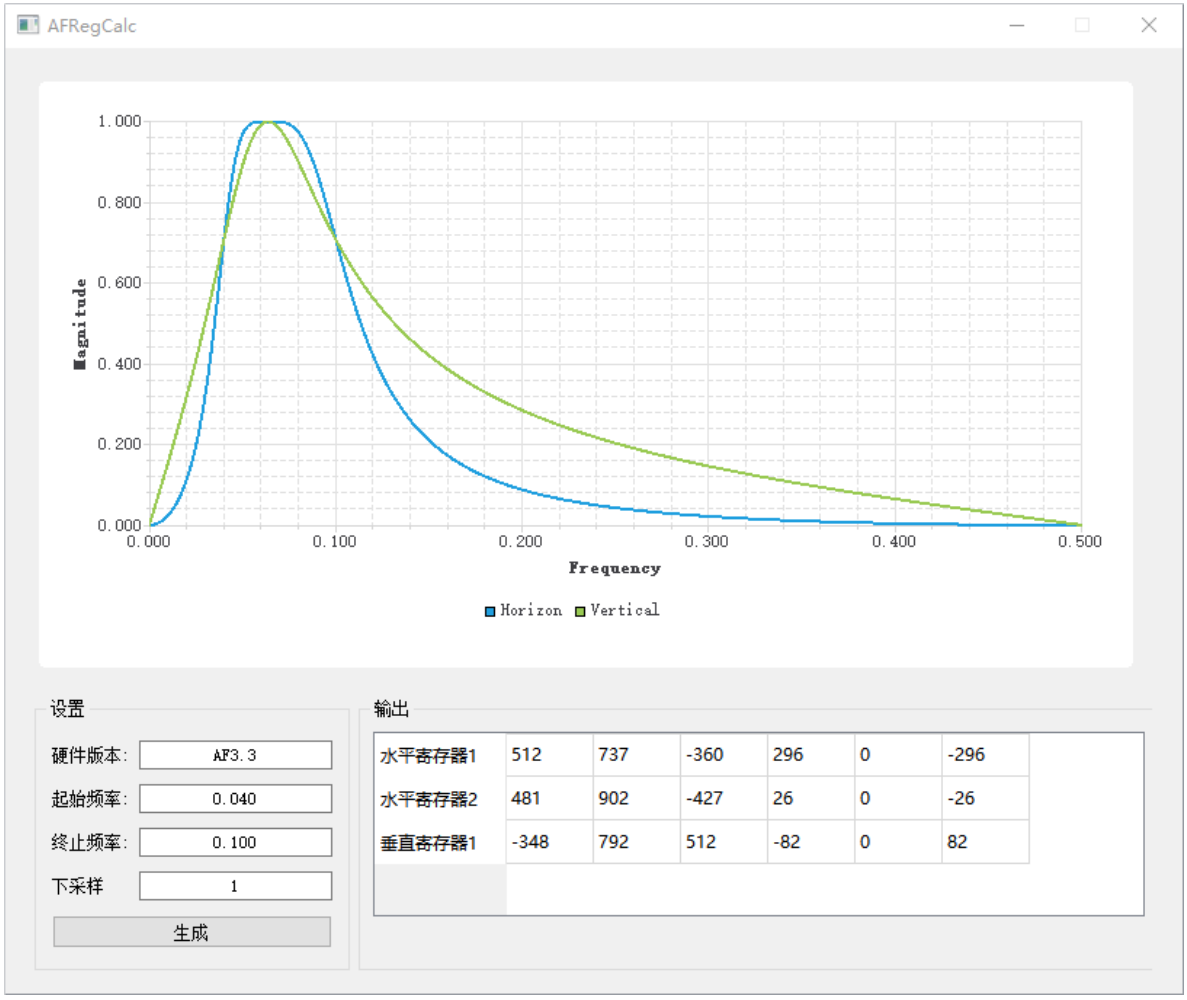
2.3.14 AF统计值的获取

详见《Rockchip_Development_Guide_ISP39》“统计信息”章节

rk_aiq_uapi2_sysctl_getIspStats ()

AF统计结果的相关结构体为afStats_stats_t

2.3.15 滤波器设计工具的使用



ISP33平台上硬件版本应输入AF3.3。

起始频率和终止频率的输入范围为0.001 ~ 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1/2/4/8，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。

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
```