

# Rockchip Development Guide ISP39

文件标识：RK-KF-GX-612

发布版本：V1.0.0

日期：2024-7-15

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。

## 版权所有 © 2024 瑞芯微电子股份有限公司

超越合理使用范畴，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。

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](http://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](mailto: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 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
RK3576	Y	N	N	N

## 修订记录

版本号	作者	修改日期	修改说明
v1.0.0	All	2024-07-29	RK3576 ISP3.9 alpha版

## 目录

### Rockchip Developement Guide ISP39

[总体概要](#)

[结构图](#)

[工作模式](#)

[性能及约束概述](#)

[概述](#)

[功能描述](#)

[RkAiq架构](#)

[软件架构](#)

[AIQ 工作模式](#)

[软件流程](#)

[API说明](#)

[系统控制](#)

[功能概述](#)

[API参考](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_preinit](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_preinit\\_scene](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_preinit\\_iq\\_addr](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_preinit\\_calibproj](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_preinit\\_devBufCnt](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_regHwEvtCb](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_switch\\_scene](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_init](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_deinit](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_prepare](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_start](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_stop](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_getStaticMetas](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_enumStaticMetas](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_enumStaticMetasByPhyId](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_getBindedSnsEntNmByVd](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_updateIq](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_getCrop](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_setCrop](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_getCamInfos](#)  
[rk\\_aiq\\_uapi2\\_get\\_aiqversion\\_info](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_getModuleCtl](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setModuleCtl](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_getModuleEn](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setModuleEn](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setSharpFbcRotation](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setMulCamConc](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_pause](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_resume](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setListenStrmStatus](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setUserOtpInfo](#)

#### 数据类型

[rk\\_aiq\\_working\\_mode\\_t](#)  
[rk\\_aiq\\_static\\_info\\_t](#)  
[rk\\_aiq\\_sensor\\_info\\_t](#)  
[rk\\_aiq\\_rect\\_t](#)  
[rk\\_aiq\\_module\\_ctl\\_t](#)  
[rk\\_aiq\\_module\\_list\\_t](#)  
[camAlgoResultType](#)

#### 离线帧处理

##### 内部离线帧

[概述](#)  
[功能框图](#)  
[功能描述](#)  
[RK-RAW格式说明](#)  
[支持的RAW格式](#)

##### API参考

[rk\\_aiq\\_uapi2\\_sysctl\\_prepareRkRaw](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_enqueueRkRawBuf](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_enqueueRkRawFile](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_registRkRawCb](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_setIspParamsDelayCnts](#)

##### 数据结构

[rk\\_aiq\\_format\\_t](#)  
[rk\\_aiq\\_rawbuf\\_type\\_t](#)  
[rk\\_aiq\\_raw\\_prop\\_t](#)

##### 注意事项

##### 参考示例

#### 外部离线帧

##### 概述

##### API参考

[rk\\_aiq\\_uapi2\\_sysctl\\_rawReproc\\_preInit](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_preInit\\_rkrawstream\\_info](#)  
[rk\\_aiq\\_uapi2\\_sysctl\\_rawReproc\\_genIspParams](#)

##### 数据结构

[rk\\_aiq\\_rkrawstream\\_mode\\_t](#)  
[rk\\_aiq\\_rkrawstream\\_info\\_t](#)  
[rk\\_aiq\\_frame\\_info\\_t](#)

#### Camera组

##### 概述

[rk\\_aiq\\_uapi2\\_camgroup\\_create](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_prepare](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_start](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_stop](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_destroy](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_getOverlapMap](#)  
[rk\\_aiq\\_uapi2\\_camgroup\\_getOverlapMap](#)

rk\_aiq\_uapi2\_camgroup\_getAiqCtxBySnsNm  
rk\_aiq\_uapi2\_camgroup\_getCamInfos  
rk\_aiq\_uapi2\_camgroup\_bind  
rk\_aiq\_uapi2\_camgroup\_unbind  
rk\_aiq\_uapi2\_sysctl\_setSnsSyncMode

#### 数据结构

rkmodule\_sync\_mode  
rk\_aiq\_camgroup\_instance\_cfg\_t  
rk\_aiq\_camgroup\_camInfos\_t  
struct RK\_PS\_SrcOverlapMap

### AE

#### 概述

#### 重要概念

#### 功能描述

#### 功能级API参考

#### 模块级API参考

rk\_aiq\_user\_api2\_ae\_setExpSwAttr  
    设置手动曝光属性  
    设置自动曝光属性  
    设置半手动曝光属性  
    设置固定帧率或自动降帧  
    设置曝光调节速度及延迟帧数  
    设置抗闪功能  
    设置AE权重  
rk\_aiq\_user\_api2\_ae\_getExpSwAttr  
rk\_aiq\_user\_api2\_ae\_setLinExpAttr  
    设置线性曝光亮度力度  
    设置背光补偿功能  
    设置强光抑制功能  
rk\_aiq\_user\_api2\_ae\_getLinExpAttr  
rk\_aiq\_user\_api2\_ae\_setHdrExpAttr  
    设置固定/动态曝光比  
    设置亮度力度  
    设置中帧参数  
rk\_aiq\_user\_api2\_ae\_getHdrExpAttr  
rk\_aiq\_user\_api2\_ae\_setIrisAttr  
rk\_aiq\_user\_api2\_ae\_getIrisAttr  
rk\_aiq\_user\_api2\_ae\_setExpWinAttr  
    设置ROI曝光功能  
rk\_aiq\_user\_api2\_ae\_getExpWinAttr  
rk\_aiq\_user\_api2\_ae\_queryExpResInfo

#### 模块级API数据类型

ae\_api\_expSwAttr\_t  
    ae\_commCtrl\_t  
        ae\_meCtrl\_t  
        ae\_speed\_t  
        ae\_delay\_t  
        ae\_frmRate\_t  
        ae\_antiFlicker\_t  
        ae\_envLvCalib\_t  
        ae\_winScale\_t  
        ae\_advanced\_t  
            ae\_expRange\_t  
ae\_api\_linExpAttr\_t  
    ae\_linInitExp\_t  
    ae\_linRoute\_t  
    ae\_dynSetpoint\_t  
    ae\_backLitCtrl\_t

ae\_overExpCtrl\_t  
ae\_api\_hdrExpAttr\_t  
    ae\_hdrInitExp\_t  
    ae\_hdrRoute\_t  
    ae\_hdrExpRatioCtrl\_t  
    ae\_hdrLfrmMode\_t  
    ae\_hdrLfrmCtrl\_t  
    ae\_hdrMfrmCtrl\_t  
    ae\_hdrSfrmCtrl\_t  
ae\_api\_irisAttr\_t  
    ae\_initIris\_t  
    ae\_manIris\_t  
    ae\_pIrisCtrl\_t  
    ae\_dclIrisCtrl\_t  
    ae\_hdclIrisCtrl\_t  
ae\_api\_queryInfo\_t  
    ae\_linExplnfo\_t  
    ae\_hdrExplnfo\_t  
Uapi\_ExpWin\_t  
Uapi\_AecStatsCfg\_t

常见问题定位及debug方法  
    曝光统计同步测试功能  
    曝光变化时出现闪烁

## AWB

概述  
重要概念  
功能描述  
功能级API参考  
模块级API参考  
rk\_aiq\_user\_api2\_awb\_SetAttrib  
rk\_aiq\_user\_api2\_awb\_GetAttrib  
rk\_aiq\_user\_api2\_awb\_QueryWBInfo  
rk\_aiq\_user\_api2\_awb\_Lock  
rk\_aiq\_user\_api2\_awb\_Unlock  
rk\_aiq\_user\_api2\_awb\_SetWbGainCtrlAttrib  
rk\_aiq\_user\_api2\_awb\_GetWbGainCtrlAttrib  
rk\_aiq\_user\_api2\_awb\_SetAwbStatsAttrib  
rk\_aiq\_user\_api2\_awb\_GetAwbStatsAttrib  
rk\_aiq\_user\_api2\_awb\_SetAwbGnCalcStepAttrib  
rk\_aiq\_user\_api2\_awb\_GetAwbGnCalcStepAttrib  
rk\_aiq\_user\_api2\_awb\_SetAwbGnCalcOthAttrib  
rk\_aiq\_user\_api2\_awb\_GetAwbGnCalcOthAttrib  
模块级API数据类型  
rk\_aiq\_wb\_querry\_info\_t  
awb\_gainCtrl\_t  
awb\_Stats\_t  
awb\_gainCalcStep\_t  
awb\_gainCalcOth\_t  
awb\_api\_attrib\_t

## AF

概述  
功能描述  
开发用户AF算法  
功能级API参考  
模块级API参考  
rk\_aiq\_user\_api2\_af\_SetAttrib  
rk\_aiq\_user\_api2\_af\_GetAttrib  
模块级API数据类型

RKAIQ\_AF\_MODE

RKAIQ\_AF\_HWVER

manual\_afStats\_cfg

rk\_aiq\_af\_attrib\_t

#### 其它说明

VCM马达模组驱动验证

电动马达模组驱动验证

#### IMGPROC

##### 概述

##### Merge

功能描述

重要概念

##### 模块级API参考

rk\_aiq\_user\_api2\_merge\_SetAttrib

rk\_aiq\_user\_api2\_amege\_v12\_GetAttrib

rk\_aiq\_user\_api2\_merge\_QueryStatus

##### 模块级API数据类型

#### DRC

功能描述

重要概念

##### 功能级API参考

##### 模块级API参考

rk\_aiq\_user\_api2\_drc\_SetAttrib

rk\_aiq\_user\_api2\_drc\_GetAttrib

rk\_aiq\_user\_api2\_drc\_QueryStatus

rk\_aiq\_user\_api2\_drc\_SetStrength

rk\_aiq\_user\_api2\_drc\_SetStrength

##### 模块级API数据类型

adrc\_strength\_t

#### Noise Removal

功能描述

##### 功能级API参考

##### 模块级API参考

rk\_aiq\_user\_api2\_btnr\_SetAttrib

rk\_aiq\_user\_api2\_btnr\_GetAttrib

rk\_aiq\_user\_api2\_btnr\_QueryStatus

rk\_aiq\_user\_api2\_btnr\_SetStrength

rk\_aiq\_user\_api2\_btnr\_GetStrength

rk\_aiq\_user\_api2\_ynr\_SetAttrib

rk\_aiq\_user\_api2\_ynr\_GetAttrib

rk\_aiq\_user\_api2\_ynr\_QueryStatus

rk\_aiq\_user\_api2\_ynr\_SetStrength

rk\_aiq\_user\_api2\_ynr\_GetStrength

rk\_aiq\_user\_api2\_cnr\_SetAttrib

rk\_aiq\_user\_api2\_cnr\_GetAttrib

rk\_aiq\_user\_api2\_cnr\_QueryStatus

rk\_aiq\_user\_api2\_cnr\_SetStrength

rk\_aiq\_user\_api2\_cnr\_GetStrength

##### 模块级API数据类型

abtnr\_strength\_t

aynr\_strength\_t

acnr\_strength\_t

#### BLC

功能描述

##### 模块级API参考

rk\_aiq\_user\_api2\_blc\_SetAttrib

rk\_aiq\_user\_api2\_blc\_GetAttrib

rk\_aiq\_user\_api2\_blc\_QueryStatus

## 模块级API数据类型

Dehaze&Enhance

功能描述

功能级API参考

rk\_aiq\_user\_api2\_dehaze\_SetAttrib

rk\_aiq\_user\_api2\_dehaze\_GetAttrib

rk\_aiq\_user\_api2\_dehaze\_QueryStatus

rk\_aiq\_user\_api2\_setDehazeEnhanceStrth

rk\_aiq\_user\_api2\_getDehazeEnhanceStrth

模块级API数据类型

HistEq

功能描述

功能级API参考

模块级API参考

rk\_aiq\_user\_api2\_histeq\_SetAttrib

rk\_aiq\_user\_api2\_histeq\_GetAttrib

rk\_aiq\_user\_api2\_histeq\_QueryStatus

模块级API数据类型

CPROC

功能描述

模块级API参考

rk\_aiq\_user\_api2\_cp\_SetAttrib

rk\_aiq\_user\_api2\_cp\_GetAttrib

rk\_aiq\_user\_api2\_cp\_QueryStatus

模块级API数据类型

cp\_params\_static\_t

cp\_param\_auto\_t

cp\_param\_t

cp\_api\_attrib\_t

cp\_status\_t

IE

功能描述

模块级API参考

rk\_aiq\_user\_api2\_ie\_SetAttrib

rk\_aiq\_user\_api2\_ie\_GetAttrib

rk\_aiq\_user\_api2\_ie\_QueryStatus

模块级API数据类型

ie\_params\_static\_t

ie\_param\_auto\_t

ie\_param\_t

ie\_api\_attrib\_t

ie\_status\_t

CSM

功能描述

模块级API参考

rk\_aiq\_user\_api2\_csm\_SetAttrib

rk\_aiq\_user\_api2\_csm\_GetAttrib

rk\_aiq\_user\_api2\_csm\_QueryStatus

模块级API数据类型

csm\_params\_static\_t

csm\_param\_t

csm\_api\_attrib\_t

csm\_status\_t

Sharp

功能描述

功能级API参考

模块级API参考

rk\_aiq\_user\_api2\_sharp\_SetAttrib

rk\_aiq\_user\_api2\_sharp\_SetAttrib  
rk\_aiq\_user\_api2\_sharp\_QueryStatus  
rk\_aiq\_user\_api2\_sharp\_SetStrength  
rk\_aiq\_user\_api2\_sharp\_GetStrength

模块级API数据类型  
asharp\_strength\_t

Gamma

功能描述  
功能级API参考  
模块级API参考

rk\_aiq\_user\_api2\_gamma\_SetAttrib  
rk\_aiq\_user\_api2\_gamma\_GetAttrib  
rk\_aiq\_user\_api2\_gamma\_QueryStatus

模块级API数据类型

CCM

功能描述  
模块级API参考

rk\_aiq\_user\_api2\_ccm\_SetAttrib  
rk\_aiq\_user\_api2\_ccm\_GetAttrib  
rk\_aiq\_user\_api2\_ccm\_QueryStatus  
rk\_aiq\_user\_api2\_ccm\_SetCalib  
rk\_aiq\_user\_api2\_ccm\_GetCalib

模块级API数据类型

ccm\_ccmAlpha\_yFac\_t  
ccm\_ccmAlpha\_satFac\_t  
ccm\_enhance\_t  
ccm\_matrix\_t  
ccm\_param\_static\_t  
ccm\_param\_dyn\_t  
ccm\_param\_t  
accm\_gain2SatCurve\_t  
accm\_param\_illuLink\_t  
accm\_param\_isoLink\_t  
accm\_param\_dyn\_t  
accm\_param\_static\_t  
ccm\_param\_auto\_t  
ccm\_api\_attrib\_t  
accm\_matrixAll\_t  
accm\_ccmCalib\_t  
accm\_status\_t  
ccm\_status\_t

LDC

功能描述  
功能级API参考

rk\_aiq\_uapi2\_setLdchEn  
rk\_aiq\_uapi2\_setLdchCorrectLevel  
rk\_aiq\_uapi2\_setLdchLdcvEn  
rk\_aiq\_uapi2\_setLdchLdcvCorrectLevel

模块级API参考

rk\_aiq\_user\_api2\_ldc\_SetAttrib  
rk\_aiq\_user\_api2\_ldc\_GetAttrib  
rk\_aiq\_user\_api2\_ldc\_QueryStatus  
rk\_aiq\_user\_api2\_ldc\_GetManualAttrib  
rk\_aiq\_user\_api2\_ldc\_SetManualAttrib

模块级API数据类型

ldc\_api\_attrib\_t  
ldc\_param\_auto\_t  
ldc\_autoGenMesh\_static\_t

ldc\_lensDistorCoeff\_static\_t  
ldc\_extMeshFile\_static\_t  
ldc\_param\_t  
ldc\_ldch\_params\_cfg\_t  
ldc\_lutMapCfg\_t

#### DeBayer

功能描述

模块级API参考

rk\_aiq\_user\_api2\_dm\_SetAttrib  
rk\_aiq\_user\_api2\_dm\_GetAttrib  
rk\_aiq\_user\_api2\_dm\_QueryStatus

模块级API数据类型

#### DPCC

功能描述

模块级API参考

rk\_aiq\_user\_api2\_dpc\_SetAttrib  
rk\_aiq\_user\_api2\_dpc\_GetAttrib  
rk\_aiq\_user\_api2\_dpc\_QueryStatus

模块级API数据类型

#### LSC

功能描述

模块级API参考

rk\_aiq\_user\_api2\_lsc\_SetAttrib  
rk\_aiq\_user\_api2\_lsc\_GetAttrib  
rk\_aiq\_user\_api2\_lsc\_QueryStatus  
rk\_aiq\_user\_api2\_lsc\_SetCalib  
rk\_aiq\_user\_api2\_lsc\_GetCalib

数据类型

lsc\_meshGrid\_t  
lsc\_meshGrid\_mode\_t  
lsc\_param\_static\_t  
lsc\_meshGain\_t  
lsc\_param\_dyn\_t  
lsc\_param\_t  
alsc\_gain2VigCurve\_t  
alsc\_param\_illuLink\_t  
alsc\_param\_dyn\_t  
alsc\_param\_static\_t  
lsc\_param\_auto\_t  
lsc\_api\_attrib\_t  
alsc\_tableAll\_t  
alsc\_lscCalib\_t  
alsc\_status\_t  
lsc\_status\_t

#### GIC

功能描述

模块级API参考

rk\_aiq\_user\_api2\_gic\_SetAttrib  
rk\_aiq\_user\_api2\_gic\_GetAttrib  
rk\_aiq\_user\_api2\_gic\_QueryStatus  
rk\_aiq\_user\_api2\_gic\_SetAttrib  
rk\_aiq\_user\_api2\_gic\_GetAttrib  
rk\_aiq\_user\_api2\_gic\_QueryStatus

模块级API数据类型

#### CGC

功能描述

模块级API参考

rk\_aiq\_user\_api2\_cgc\_SetAttrib

[rk\\_aiq\\_user\\_api2\\_cgc\\_GetAttrib](#)  
[rk\\_aiq\\_user\\_api2\\_cgc\\_QueryStatus](#)

模块级API数据类型

[cgc\\_params\\_static\\_t](#)  
[cgc\\_param\\_auto\\_t](#)  
[cgc\\_param\\_t](#)  
[cgc\\_api\\_attrib\\_t](#)  
[cgc\\_status\\_t](#)

CAC

功能描述

模块级API参考

[rk\\_aiq\\_user\\_api2\\_cac\\_SetAttrib](#)  
[rk\\_aiq\\_user\\_api2\\_cac\\_GetAttrib](#)  
[rk\\_aiq\\_user\\_api2\\_cac\\_QueryStatus](#)

模块级API数据类型

3DLut

功能描述

模块级API参考

[rk\\_aiq\\_user\\_api2\\_3dlut\\_SetAttrib](#)  
[rk\\_aiq\\_user\\_api2\\_3dlut\\_GetAttrib](#)  
[rk\\_aiq\\_user\\_api2\\_3dlut\\_QueryStatus](#)  
[rk\\_aiq\\_user\\_api2\\_3dlut\\_SetCalib](#)  
[rk\\_aiq\\_user\\_api2\\_3dlut\\_GetCalib](#)

模块级API数据类型

IMGPROC功能级API

AE相关

[rk\\_aiq\\_uapi2\\_setAeLock](#)  
[rk\\_aiq\\_uapi2\\_setExpMode](#)  
[rk\\_aiq\\_uapi2\\_getExpMode](#)  
[rk\\_aiq\\_uapi2\\_setExpTimeMode](#)  
[rk\\_aiq\\_uapi2\\_getExpTimeMode](#)  
[rk\\_aiq\\_uapi2\\_setExpGainMode](#)  
[rk\\_aiq\\_uapi2\\_getExpGainMode](#)  
[rk\\_aiq\\_uapi2\\_setExpManualGain](#)  
[rk\\_aiq\\_uapi2\\_setExpManualTime](#)  
[rk\\_aiq\\_uapi2\\_setExpGainRange](#)  
[rk\\_aiq\\_uapi2\\_getExpGainRange](#)  
[rk\\_aiq\\_uapi2\\_setExpTimeRange](#)  
[rk\\_aiq\\_uapi2\\_getExpTimeRange](#)  
[rk\\_aiq\\_uapi2\\_setBLCMode](#)  
[rk\\_aiq\\_uapi2\\_setBLCStrength](#)  
[rk\\_aiq\\_uapi2\\_setHLCMode](#)  
[rk\\_aiq\\_uapi2\\_setHLCStrength](#)  
[rk\\_aiq\\_uapi2\\_setAntiFlickerEn](#)  
[rk\\_aiq\\_uapi2\\_getAntiFlickerEn](#)  
[rk\\_aiq\\_uapi2\\_setAntiFlickerMode](#)  
[rk\\_aiq\\_uapi2\\_getAntiFlickerMode](#)  
[rk\\_aiq\\_uapi2\\_setExpPwrLineFreqMode](#)  
[rk\\_aiq\\_uapi2\\_getExpPwrLineFreqMode](#)

AWB相关

[rk\\_aiq\\_uapi2\\_setWBMode](#)  
[rk\\_aiq\\_uapi2\\_getWBMode](#)  
[rk\\_aiq\\_uapi2\\_lockAWB](#)  
[rk\\_aiq\\_uapi2\\_unlockAWB](#)  
[rk\\_aiq\\_uapi2\\_setMWBScene](#)  
[rk\\_aiq\\_uapi2\\_getMWBScene](#)  
[rk\\_aiq\\_uapi2\\_setMWBGain](#)  
[rk\\_aiq\\_uapi2\\_getMWBGain](#)

rk\_aiq\_uapi2\_setMWBCT  
rk\_aiq\_uapi2\_getWBCT  
rk\_aiq\_uapi2\_setAwbGainOffsetAttrib  
rk\_aiq\_uapi2\_getAwbGainOffsetAttrib

#### AF相关

rk\_aiq\_uapi2\_setFocusMode  
rk\_aiq\_uapi2\_getFocusMode  
rk\_aiq\_uapi2\_setFocusWin  
rk\_aiq\_uapi2\_getFocusWin  
rk\_aiq\_uapi2\_lockFocus  
rk\_aiq\_uapi2\_unlockFocus  
rk\_aiq\_uapi2\_oneshotFocus  
rk\_aiq\_uapi2\_manualTrigerFocus  
rk\_aiq\_uapi2\_trackingFocus  
rk\_aiq\_uapi2\_getSearchResult  
rk\_aiq\_uapi2\_getZoomRange  
rk\_aiq\_uapi2\_setOpZoomPosition  
rk\_aiq\_uapi2\_getOpZoomPosition  
rk\_aiq\_uapi2\_endOpZoomChange  
rk\_aiq\_uapi2\_getFocusRange  
rk\_aiq\_uapi2\_setFocusPosition  
rk\_aiq\_uapi2\_getFocusPosition  
rk\_aiq\_uapi2\_startZoomCalib  
rk\_aiq\_uapi2\_resetZoom

#### 去雾及对比度

rk\_aiq\_uapi2\_setMDehazeStrth  
rk\_aiq\_uapi2\_getMDehazeStrth  
rk\_aiq\_uapi2\_setMEnhanceStrth  
rk\_aiq\_uapi2\_getMEnhanceStrth  
rk\_aiq\_uapi2\_setMEnhanceChromeStrth  
rk\_aiq\_uapi2\_setMEnhanceChromeStrth

#### DRC相关

rk\_aiq\_uapi2\_setHDRStrth  
rk\_aiq\_uapi2\_getHDRStrth  
rk\_aiq\_uapi2\_setDarkAreaBoostStrth  
rk\_aiq\_uapi2\_getDarkAreaBoostStrth

#### 去噪

rk\_aiq\_uapi2\_setNRMode  
rk\_aiq\_uapi2\_getNRMode  
rk\_aiq\_uapi2\_setANRStrth  
rk\_aiq\_uapi2\_getANRStrth  
rk\_aiq\_uapi2\_setMSpaNRStrth  
rk\_aiq\_uapi2\_getMSpaNRStrth  
rk\_aiq\_uapi2\_setMTNRStrth  
rk\_aiq\_uapi2\_getMTNRStrth

#### CCM相关

rk\_aiq\_uapi2\_setCCMMode  
rk\_aiq\_uapi2\_getCCMMode  
rk\_aiq\_uapi2\_setMCcCoef  
rk\_aiq\_uapi2\_getMCcCoef  
rk\_aiq\_uapi2\_getACcmSat  
rk\_aiq\_uapi2\_getACcmMatrixName

#### 3DLUT相关

rk\_aiq\_uapi2\_setLut3dMode  
rk\_aiq\_uapi2\_getLut3dMode  
rk\_aiq\_uapi2\_setM3dLut  
rk\_aiq\_uapi2\_getM3dLut  
rk\_aiq\_uapi2\_getA3dLutStrth

rk\_aiq\_uapi2\_getA3dLutName  
对比度  
    rk\_aiq\_uapi2\_setContrast  
    rk\_aiq\_uapi2\_getContrast  
亮度  
    rk\_aiq\_uapi2\_setBrightness  
    rk\_aiq\_uapi2\_getBrightness  
饱和度  
    rk\_aiq\_uapi2\_setSaturation  
    rk\_aiq\_uapi2\_getSaturation  
色度  
    rk\_aiq\_uapi2\_setHue  
    rk\_aiq\_uapi2\_getHue  
色彩模式配置  
    rk\_aiq\_uapi2SetColorMode  
    rk\_aiq\_uapi2\_getColorMode  
灰度范围、色彩空间配置  
    rk\_aiq\_uapi2SetColorSpace  
    rk\_aiq\_uapi2\_getColorSpace  
灰度模式配置  
    rk\_aiq\_uapi2\_setGrayMode  
    rk\_aiq\_uapi2\_getGrayMode  
GAMMA相关  
    rk\_aiq\_uapi2\_setGammaCoef  
功能级API数据类型  
    opMode\_t  
    paRange\_t  
    aeMeasAreaType\_t  
    expPwrLineFreq\_t  
    antiFlickerMode\_t  
    rk\_aiq\_wb\_op\_mode\_t  
    rk\_aiq\_wb\_mwb\_mode\_t  
    rk\_aiq\_wb\_gain\_t  
    rk\_aiq\_wb\_scene\_t  
    rk\_aiq\_wb\_cct\_t  
    rk\_aiq\_wb\_mwb\_attrib\_t  
    rk\_aiq\_uapiV2\_wb\_awb\_wbGainOffset\_t  
    CalibDbV2\_Awb\_gain\_offset\_cfg\_t  
    rk\_aiq\_uapiV2\_wbV32\_awb\_gainAdjust\_t  
    rk\_aiq\_uapiV2\_wbV32\_awb\_mulWindow\_t  
    rk\_aiq\_wb\_awb\_alg\_method\_t  
    rk\_aiq\_uapiV2\_wb\_awb\_dampFactor\_t  
    CalibDbV2\_Awb\_DaylgtClip\_Cfg\_t  
    rk\_aiq\_uapiV2\_wb\_awb\_cctClipCfg\_t  
    rk\_aiq\_uapiV2\_wbV32\_awb\_attrib\_t  
    rk\_aiq\_uapiV2\_wbV32\_attrib\_t  
    rk\_aiq\_af\_zoomrange  
    rk\_aiq\_af\_focusrange  
    rk\_aiq\_af\_result\_t  
    rk\_aiq\_ccm\_matrix\_t  
    rk\_aiq\_lut3d\_table\_t

统计信息  
    概述  
    功能描述  
        AE统计信息  
            基于raw域的AE统计  
        AWB统计信息  
        AF统计信息

## API参考

[rk\\_aiq\\_uapi\\_sysctl\\_get3AStatsBlk](#)  
[rk\\_aiq\\_uapi\\_sysctl\\_release3AStatsRef](#)  
[rk\\_aiq\\_uapi2\\_stats\\_getLspStats](#)

## 数据类型

[rk\\_aiq\\_isp\\_stats\\_t](#)  
AE统计数据类型  
RKAIqAecStats\_t  
RKAIqAecExplInfo\_t  
RkAiqExpParamComb\_t  
RKAIqExpl2cParam\_t  
RkAiqIrisParamComb\_t  
RkAiqAecHwStatsRes\_t  
aeStats\_entityStats\_t  
aeStats\_mainWinStats\_t  
aeStats\_subWinStats\_t  
aeStats\_histStats\_t

## AWB统计数据类型

[awbStats\\_stats\\_t](#)  
[awbStats\\_wpStats\\_t](#)  
[awbStats\\_pixStats\\_t](#)  
[afStats\\_stats\\_t](#)

## ISP效果参数封装

### JSON-IQ与场景切换

### IQ文件格式说明

模块说明:

差分参数特性:

编程接口:

    接口描述:

### JSON-IQ转BINARY-IQ

### 概述

### 转换工具使用说明

    编译:

    转换说明:

        参数说明

## SmartIr

### 概述

### 功能描述

### API参考

[rk\\_smart\\_ir\\_init](#)  
[rk\\_smart\\_ir\\_deinit](#)  
[rk\\_smart\\_ir\\_setAttr](#)  
[rk\\_smart\\_ir\\_getAttr](#)  
[rk\\_smart\\_ir\\_queryInfo](#)  
[rk\\_smart\\_ir\\_runOnce](#)  
[rk\\_smart\\_ir\\_groupRunOnce](#)  
[rk\\_smart\\_ir\\_config](#)  
[rk\\_smart\\_ir\\_set\\_status](#)  
[rk\\_smart\\_ir\\_auto\\_irled](#)

## 数据类型

枚举型数据说明  
[rk\\_smart\\_ir\\_params\\_t](#)  
[rk\\_smart\\_ir\\_attr\\_t](#)  
[rk\\_smart\\_ir\\_result\\_t](#)  
[rk\\_smart\\_ir\\_query\\_info\\_t](#)  
[rk\\_smart\\_ir\\_autoled\\_t](#)

## 功能集成

## 参数调试

[AIISP](#)  
    [功能描述](#)  
    [约束](#)  
    [API参考](#)

[rk\\_aiq\\_uapi2\\_sysctl\\_initAiisp](#)  
[log解读](#)

[Debug & FAQ](#)  
    [如何获取版本号](#)  
        [获取简略版本信息](#)  
        [获取完整版本信息](#)  
        [版本号匹配规则说明](#)

[AIQ Log](#)  
    [概述](#)

**AE**  
    [配置指南](#)  
    [log解读](#)

**AWB**  
    [配置指南](#)  
    [log解读](#)

**AF**  
    [配置指南](#)  
    [log解读](#)

**MERGE**  
    [配置指南](#)  
    [log解读](#)

**DRC**  
    [配置指南](#)  
    [log解读](#)

**NR&Sharp**  
    [配置指南](#)  
    [log解读](#)

**Dhz&Ehz**  
    [配置指南](#)  
    [Log解读](#)

**CamHW**  
    [配置指南](#)  
    [log解读](#)

[如何采集Raw/YUV图像](#)  
    [Raw数据存储格式](#)  
        [非紧凑型存储格式](#)  
        [紧凑型存储格式](#)  
        [RK-Raw V1.0](#)  
        [RK-Raw V2.0](#)  
    [Raw/YUV数据采集方式](#)

[错误码](#)  
[缩略语](#)

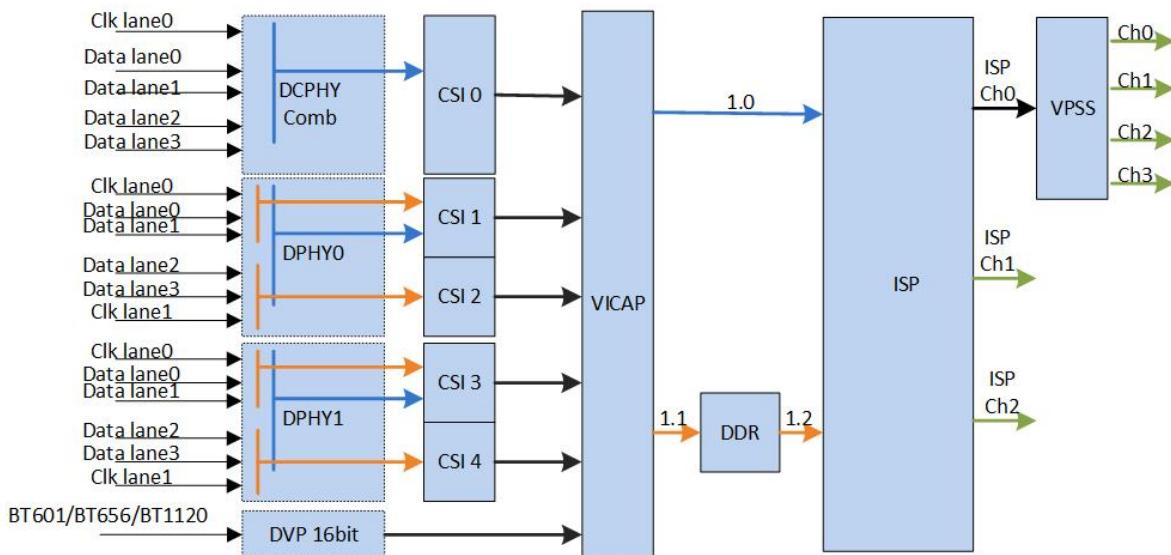
---

## 总体概要

---

## 结构图

## RK3576 VI



→ 蓝色与橙色数据流通路互斥，2选1  
→ 绿色数据流代表用户获取的最终数据

详细框图可参考《Rockchip\_Tuning\_Guide\_ISP39》文档 ISP39 Introduction章节。

RK3576 VI作为视频输入模块总称，其中包含：

- 接口以及数据采集部分：DPHY、DVP、CSI、VICAP
- 图像处理部分：ISP、VPSS

RK3576 ISP属于RK ISP v3.9版本，后续文档以ISP39标识。

## 工作模式

**单ISP单CIS(cmos image sensor)：**

- Raw直通/在线模式：结构图中蓝色箭头指示数据流1.0通路
- Raw回读/离线模式：结构图中橙色箭头指示数据流1.1，1.2通路

**多CIS：**

- 单ISP采用Raw回读/离线模式下，可以采用时分复用的方式支持多CIS的输入。

## 性能及约束概述

- 支持分辨率及吞吐率：

工作模式	吞吐率	支持最大分辨率	支持最小分辨率
单ISP单CIS	16M@30fps 或者 48M@10fps	8064x6048	264x264
多CIS时分复用	最多支持4 CIS 时分复用 总吞吐率为 16Mega@30fps	4 CIS： 3840x1664	
<b>注意：表格中最大分辨率 8064x6048 是将图像分割成左右两片处理的，ISP39单帧图像支持的最大分辨率为 16M。</b>			

- 支持Raw12数据输入。

ISP39支持处理Raw12数据。Raw14数据输入将低位丢弃成Raw12处理。位宽不足12bit图像数据输入将低位补0成Raw12处理。

- 支持Bayer color CIS、Mono CIS。
- 支持2合1 多帧HDR。支持HDR CIS 类型如下：

Multi exposure HDR	RK Platform support	CIS	应用
Sequential / Framebase HDR	ISP20: RV1109/RV1126 etc ISP21: RK356X etc ISP30: RK3588 ISP32: RV1106 ISP33:RV1103 ISP39:RK3576	OV4689	IPC/CVR
Staggered / DOL HDR	ISP20: RV1109/RV1126 etc ISP21: RK356X etc ISP30: RK3588 ISP32: RV1106 ISP33:RV1103 ISP39:RK3576	OS04A10 IMX464 AR0239	IPC/CVR
DCG HDR	ISP20: RV1109/RV1126 etc ISP21: RK356X etc ISP30: RK3588 ISP32: RV1106 ISP33:RV1103 ISP39:RK3576	OS04A10 IMX585	IPC/CVR
12bit(PWL) compressed combined raw	ISP39:RK3576	OX03C10	IPC/CVR
SME HDR	N	IMX378	消费类
BME HDR	N	IMX258	消费类
QBC HDR	N	OV50C40 IMX586	消费类
Lateral overflow HDR	N	AR0821	IPC
Split-diode pixel HDR	N	OV10640	CVR

- Raw直通/在线模式下，ISP39数据输出至DDR通知后级的最小时延：约40~50行。
- ISP时分复用方式下，单ISP最多支持4路数据源输入。

详细约束信息可参考《Rockchip\_Tuning\_Guide\_ISP39》文档 "ISP39约束" 章节。

## 概述

ISP39 包含了一系列的图像处理算法模块，主要包括：暗电流矫正、坏点矫正、3A、HDR、镜头阴影矫正、镜头畸变矫正、HSV、去噪（包括RAW域去噪，多帧降噪，颜色去噪等）、锐化等。

ISP39包括硬件算法实现及软件逻辑控制部分，RkAiq即为软件逻辑控制部分的实现。

RkAiq软件模块主要实现的功能为：从ISP驱动获取图像统计，结合IQ Tuning参数，使用一系列算法计算出新的ISP、Sensor等硬件参数，不断迭代该过程，最终达到最优的图像效果。

## 功能描述

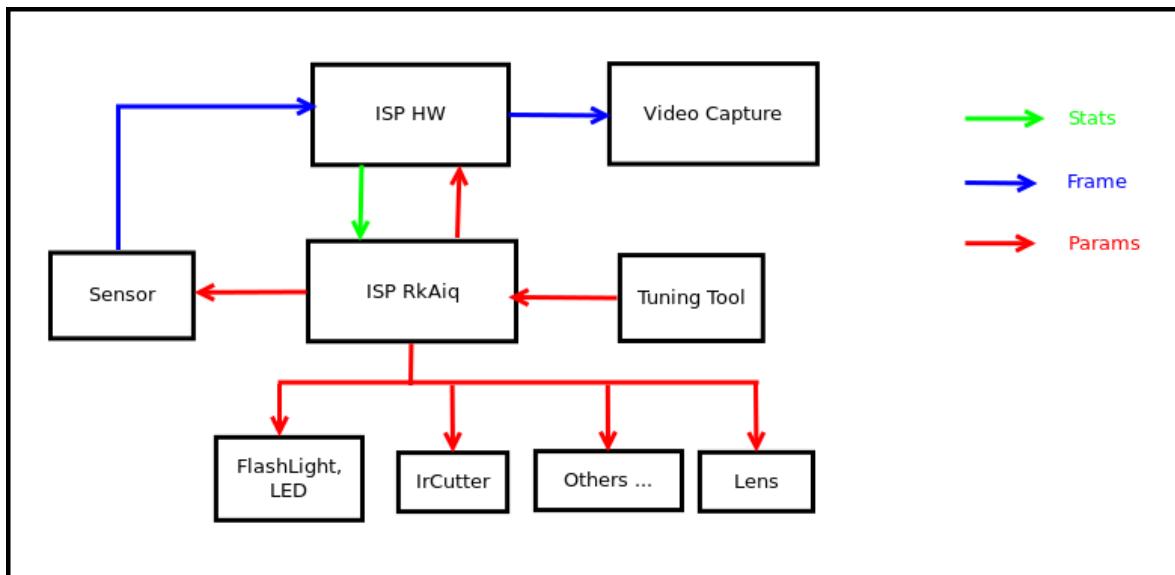


图1-1 ISP21 系统框图

ISP39总体软硬件框图如图1-1所示。Sensor输出数据流给ISP HW，ISP HW再输出经过一系列图像处理算法后的图像。RkAiq不断从ISP HW获取统计数据，并经过3A等算法生成新的参数反馈给各硬件模块。Tunning tool可在线实时调试参数，调试好后可保存生成新的iq参数文件。

## RkAiq架构

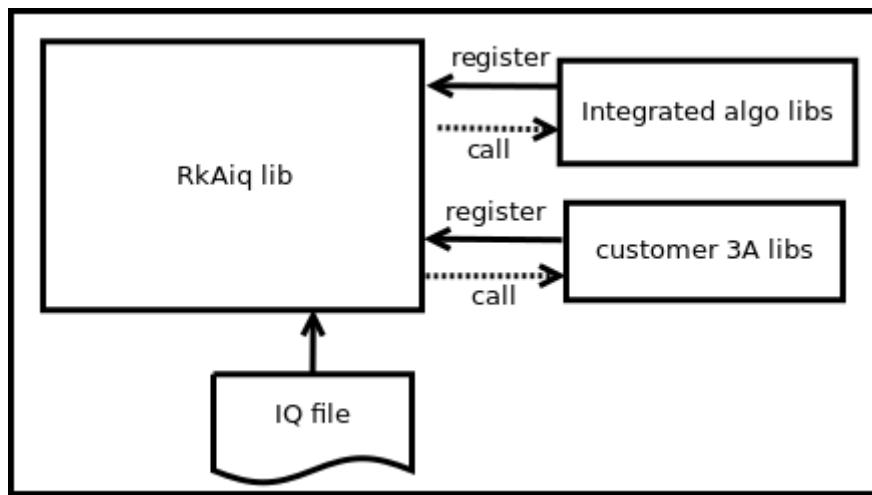


图1-2 RkAiq总体架构图

ISP39 RkAiq软件设计思路如图1-2所示。主要分成以下四个部分：

1. RkAiq lib 动态库。该库包含了主要的逻辑部分，负责从驱动获取统计，并传送给各个 算法库。
2. Integrated algo libs。Rk提供的静态算法库，已默认注册到RkAiq lib动态库。
3. customer 3A libs。客户可根据算法库接口定义实现自己的3A算法库，或者其他算法库。将自定义 算法库注册给RkAiq lib动态库后，可根据提供的接口选择跑自定义库还是跑Rk库。
4. IQ file。iq tuning结果文件，保存的是算法相关参数以及CIS等一些系统静态参数。

## 软件架构

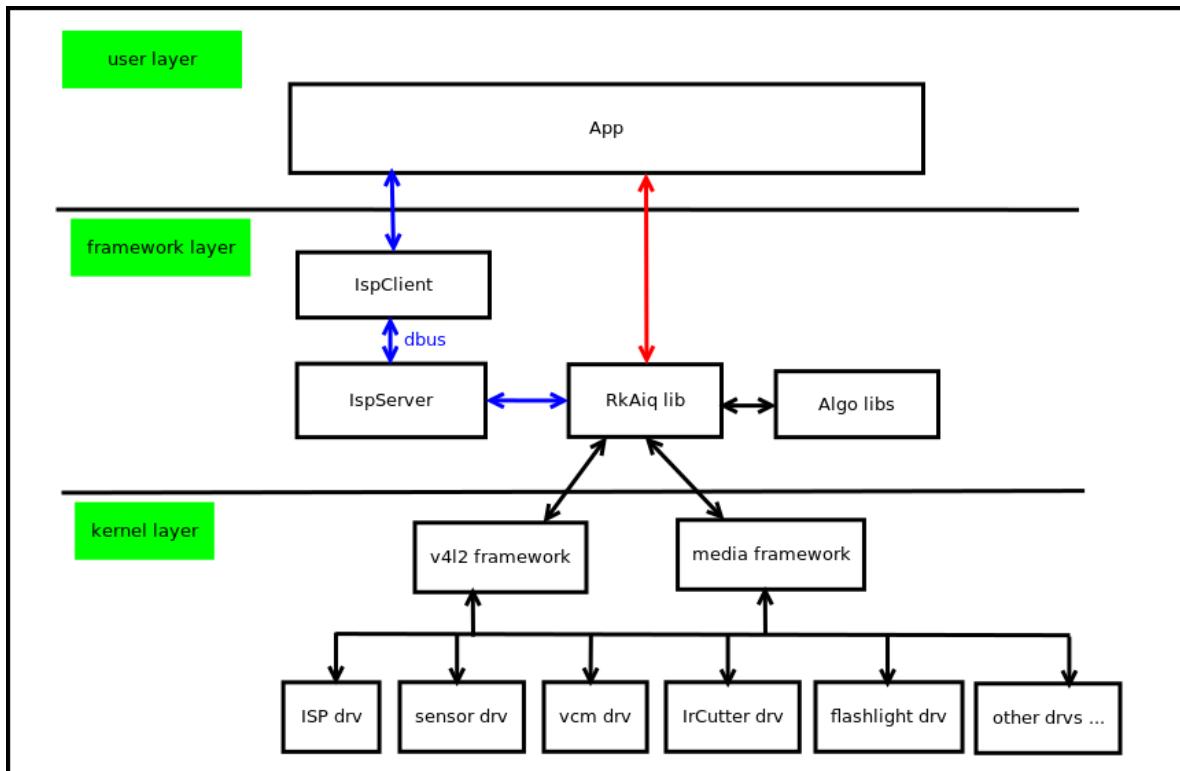


图1-3 软件架构框图

ISP39 软件框图如图1-3所示。主要分成以下三层：

1. **kernel layer**。该层包含所有Camera系统的硬件驱动，主要有ISP驱动、sensor驱动、vcm驱动、flashlight驱动、IrCutter驱动等等。驱动都基于V4L2及Media框架实现。
2. **framework layer**。该层为RkAiq lib的集成层，Rkaiq lib有两种集成方式：
  - **IspServer 方式**  
该方式Rkaiq lib跑在 IspServer独立进程，客户端通过dbus或者socket等方式与之通信。此外，该方式可为v4l-ctl等现有第三方应用，在不修改源码的情况下，提供具有ISP调试效果的图像。
  - **直接集成方式**  
RkAiq lib可直接集成进应用。
3. **user layer**。用户应用层。

## AIQ 工作模式

- **AIQ在线模式：**  
结合“总体概要”章节的框图，ISP工作在Raw直通/在线模式，即VICAP与ISP的数据流通路1.0。AIQ控制ISP设备的参数、CIS设备曝光、马达设备。不参与ISP的数据流操作。
- **AIQ半离线模式：**  
结合“总体概要”章节的框图，ISP工作在Raw回读/离线模式，即VICAP与ISP的数据流通路1.1与1.2。AIQ半离线模式下，将VICAP采集存储在DDR中的帧数据，以帧buffer轮转的方式控制ISP来处理这些数据。  
与AIQ在线模式一样，ISP参数的控制、CIS设备、马达等设备依旧由AIQ内部控制。
- **AIQ全离线模式：**  
ISP作为一个从用户指定内存中读取图像进行处理设备。APP通过AIQ提供接口输入希望ISP设备处理的图像Raw数据、以及图像Raw数据配套的曝光信息。AIQ根据输入的曝光信息生成ISP参数来配置ISP设备，同时将raw数据输入给ISP设备进行处理。IQ仅控制ISP单个设备，不操作其他设备：CIS、马达等。

这里区分的AIQ工作模式和硬件前述的硬件工作模式不太相同，是AIQ软件模块的工作模式。主要是用来区分是否需要搬运RAW数据，以及RAW数据是由AIQ还是外部（ISP驱动、应用、Rockit）负责搬运。与AIQ工作模式相关API如下：

1. rk\_aiq\_uapi2\_sysctl\_setMulCamConc: 设置为TRUE时，AIQ工作在半离线模式，且RAW由AIQ负责搬运
2. rk\_aiq\_uapi2\_sysctl\_rawReproc\_preInit: mode为0时 AIQ工作在半离线模式，且RAW由外部负责搬运  
mode为1时，AIQ工作在全离线模式，且RAW由外部负责搬运
3. rk\_aiq\_uapi2\_sysctl\_prepareRkRaw: AIQ工作在全离线模式，且RAW由AIQ负责搬运  
其中 2 和 3 互斥，优先级高于 1。  
如未调用以上API，那么AIQ工作在线模式，此时可能是由外部搬运RAW，ISP硬件实际上可能处于离线工作模式。

为了区分全/半离线模式以及RAW搬运方式，引入以下概念：

**\*\* 外部全/半离线方式 \*\***：RAW由外部搬运，可以是RawStream库、Rockit或者应用

**\*\* 内部全/半离线方式 \*\***：RAW由AIQ搬运

## 软件流程

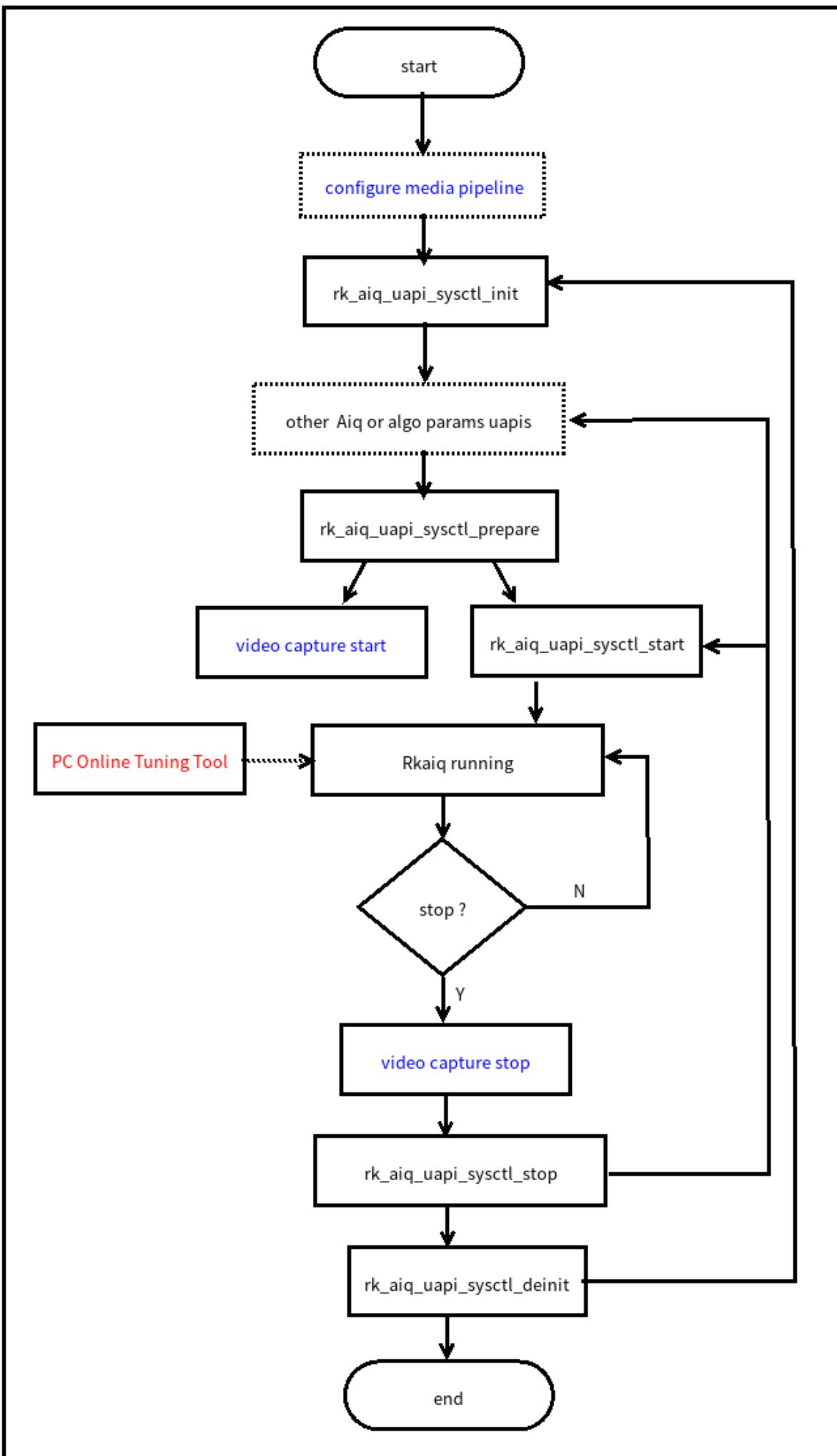


图1-4 流程图

RkAiq接口调用流程如图1-4所示。图中虚线框部分为可选部分，蓝色字体部分为应用需要配合RkAiq流程所作的配置。

- configure media pipeline。可选项，配置ISP30 pipeline，如sensor输出分辨率等等，驱动已有默认配置。
- rk\_aiq\_uapi2\_sysctl\_init。初始化sensor的AIQ上下文（Context），包括IQ tuning参数及各算法库初始化。
- other AiQ or algo params uapis。可选项，可通过各算法提供的API接口配置需要的参数，以及注册第三方算法库等等。
- rk\_aiq\_uapi2\_sysctl\_prepare。准备各算法库及各硬件模块的初始化参数，并设置到驱动。
- video capture start。该流程为应用端ISP数据流的开启，该流程需要在rk\_aiq\_uapi2\_sysctl\_prepare后调用。
- rk\_aiq\_uapi2\_sysctl\_start。启动RkAiq内部流程，该接口调用成功后，sensor开始输出数据，ISP开始处理数据，并输出处理后的图像。
- Rkaiq running。RkAiq不断从ISP驱动获取统计数据，调用3A等算法计算新参数，并应用新参数到驱动。
- PC Online Tunning Tool。PC端可通过Tunning Tool在线调整参数。
- video capture stop。停止RkAiq流程前需要先停止数据流部分。
- rk\_aiq\_uapi2\_sysctl\_stop。停止 RkAiq running 流程。可调整参数后再启动或者直接再启动。
- rk\_aiq\_uapi2\_sysctl\_deinit。反初始化RkAiq。

\*\* 注意 \*\*

- 同一sensor的AIQ运行上下文（Context）只支持单个实例，不支持多个实例，无论是在多进程中还是在单进程中
- 该流程为单摄在线模式流程，快起流程参考产品部快起文档，多摄流程参考[Camera组](#)章节，离线帧模式参考[离线帧处理](#)章节
- AIQ内部半离线时注意点：

ISP 和 Vicap 驱动有以下限制：

1. 当 ISP 工作在直通模式时，sensor出流由 ISP 的 MP/SP 等YUV节点控制，任一 YUV 数据流开启时，sensor开始出流
2. 当 ISP 工作在离线模式时，sensor出流由 Vicap 控制，Vicap节点的任一数据流开启时，sensor开始出流
3. ISP 驱动提供有 YUV 节点 STREAM ON/OFF 事件，当应用有注册监听 STREAM ON 事件时，ISP 不管工作在在线还是离线模式，ISP驱动在第一路 YUV 流开启时会等待AIQ下初始参数，目前超时等待时间是 1 秒。

基于上述信息，AIQ有两种工作流程：

1. rk\_aiq\_uapi2\_sysctl\_setListenStrmStatus 设置为false时，那么 AIQ 不监听 STREAM ON 事件。  
当 AIQ 工作在内部半离线时，Vicap 由AIQ控制，rk\_aiq\_uapi2\_sysctl\_start 调用时时启动 Vicap，Sensor开始出流。
2. 如果未调用rk\_aiq\_uapi2\_sysctl\_setListenStrmStatus，或者设置成true，那么AIQ会监听 STREAM ON 事件。  
当 AIQ 工作在内部半离线时，rk\_aiq\_uapi2\_sysctl\_start 调用时时并不会立即启动Vicap，只会启动监听线程，等到监听到STREAM ON事件后才启动Vicap，Sensor开始出流。

默认情况下，AIQ工作在方式2。只有在 ISP 3A Server 方式下，AIQ 工作在方式 1，由于 3A Server 中监听了 STREAM ON/OFF 事件，AIQ 内部不再监听。

由于 YUV 数据流开启和 AIQ start 在应用端是2个操作，且2个操作可能中间有时延，特别是在多 Sensor情况下，如果使用方式 1，如果 AIQ先start，可能造成丢帧，这要求 AIQ start 在数据流开启后执行；如果使用方式 2，

由于监听事件动作是在AIQ Start后，如果数据流先启动，则会造成AIQ监听不到STREAMON事件，导致未启动Vicap，从而造成不出流。为解决这个矛盾，做出以下规定：

1. 只有在 3A server 时，AIQ 才不监听 STREAM ON/OFF 事件
2. 非 3A server 时，且AIQ为内部办半离线模式时, rk\_aiq\_uapi2\_sysctl\_start 必须在YUV数据流启动前调用。  
在线模式时没有这个限制，为了统一流程，上述在线模式流程也可以先调用 rk\_aiq\_uapi2\_sysctl\_start，再启动数据流。

## API说明

- RKAiq提供的API分为两个级别：功能级别API 与 模块级别API。
  - **功能级别API**：基于模块级别API封装而成，主要是面对产品应用基于该模块的一些简单功能设计。头文件为 include/uAPI2/rk\_aiq\_user\_api2\_imgproc.h
  - **模块级别API**：该模块的详细参数设置以及查询，未对功能进行API区分。头文件在 include/uAPI2/ 文件夹内
- 提供 rk\_aiq\_user\_api2\_isp.h, 该头文件包含了所有 RK3576 平台所需的模块级 API 头文件。应用需要使用模块级API时，需要先包含该文件。虽然应用可只包含某个模块的头文件，但可能导致某些宏未定义造成编译错误。因此，推荐直接包含 rk\_aiq\_user\_api2\_isp39.h。  
注意：AIQ 库编译完成时，会同时将 rk\_aiq\_user\_api2\_isp39.h 安装成 rk\_aiq\_user\_api2\_isp.h。因此，应用可包含 rk\_aiq\_user\_api2\_isp.h 或者 rk\_aiq\_user\_api2\_isp39.h。
- 模块API接口一般提供 get、set 及 query 接口。get 接口可获取当前配置，set 接口可设置新参数，query 接口可查询当前硬件生效参数及一些算法状态。一般操作为先get得到当前配置，后加入所需修改的参数得到新的配置，再调用 set。此外，set 参数并非立即应用到ISP硬件，一般需要 1~2 帧生效时间。
- AIQ API参考代码按照模块进行划分，单独提供API示例，建议客户使用时可以直接参考 rkisp\_demo/demo/sample。所提供的sample代码为多平台共用，ISP39请参考 USE\_NEWSTRUCT 宏部分。

文件名	模块
sample_3dlut_module.c	HSV
sample_ccm_module.c	CCM
sample_csm_module.c	CSM
sample_ablc_module.c	BLC
sample_abayertnr_module.c	BTNR
sample_aynr_module.c	YNR
sample_acnr_module.c	CNR
sample_asharp_module.c	Sharp
sample_amege_module.c	Merge(HDR)
sample_adrc_module.c	DRC
sample_adehaze_module.c	Enhance & HISTEQ
sample_agamma_module.c	Gamma
sample_awb_module2.c	AWB
sample_ae_module.c	AE
sample_af_module.c	AF
sample_cac_module.c	CAC
sample_gic_module.c	GIC
sample_aie_module.c	IE
sample_ainr_module.c	AINR
sample_alsc_module.c	LSC
sample_rgbir_module.c	RGBIR
sample_smartir_module.c	smartir

## 系统控制

### 功能概述

系统控制部分包含了AIQ 公共属性配置，初始化 AIQ、运行 AIQ、退出AIQ，设置 AIQ各模块等功能，所需包含的头文件为 rk\_aiq\_user\_api2\_sysctl.h。

### API参考

#### **rk\_aiq\_uapi2\_sysctl\_preInit**

## 【描述】

初始化AIQ前，预设值使用的IQ文件。

【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_preInit (const char* sns_ent_name,  
                           rk_aiq_working_mode_t mode,  
                           const char* force_iq_file);
```

## 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
mode	保留参数，不使用	输入
force_iq_file	强制使用的 iq 文件	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
  - 库文件: librkaiq.so

### 【注意】

- 应先于rk\_aiq\_uapi2\_sysctl\_init 调用
  - 参数 mode 不使用。工作模式需要在 rk\_aiq\_uapi2\_sysctl\_prepare 中配置。
  - 参数 force\_iq\_file 为iq文件名称，iq路径在 rk\_aiq\_uapi2\_sysctl\_init 时指定
  - 如不调用该API，AIQ根据默认规则选择IQ文件。默认的IQ文件命名需与内核DTS中Camera信息定义一致。具体规则如下：  
<sensor\_name><rockchip,camera-module-name><rockchip,camera-module-lens-name>.json
  - 该函数不是必须，仅用于 rk\_aq\_uapi2\_sysctl\_init 更改使用的IO文件

## rk aq uapi2 sysctl prelnit scene

## 【描述】

初始化AIQ前，选择AIQ启动时的场景参数。如不调用该API，则AIQ默认选择 normal-day 场景参数。

【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_preInit_scene (const char* sns_ent_name,  
        const char *main_scene,  
        const char *sub_scene);
```

## 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
main_scene	主场景, 值在IQ文件中定义	输入
sub_scene	子场景, 值在IQ文件中定义	输入

## 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

## 【注意】

- 应先于rk\_aiq\_uapi2\_sysctl\_init 调用
- main\_scene 需要根据 IQ 文件中主场景数组来选择, 默认一般是 “normal”
- sub\_scene 需要根据 IQ 文件中主场景里的子场景数组来选择, 默认一般是 “day”
- 由于目前默认选择的场景是 normal-day, 如果要跑 hdr-day, 或者其他自定义模式, 则需要调用该函数进行指定, 否则IQ参数可能不正确

## **rk\_aiq\_uapi2\_sysctl\_preInit\_iq\_addr**

### 【描述】

设置 IQ bin 加载地址, 一般在快起项目中使用。

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_preInit_iq_addr(const char* sns_ent_name, void *addr, size_t len);
```

## 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
addr	iq bin 内存地址	输入
len	iq 参数长度	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 应先于rk\_aiq\_uapi2\_sysctl\_init 调用
- 调用该接口后，不再使用文件系统中的 IQ json 及 IQ bin 文件
- iq bin 需要由 j2s\_4b 工具生成，Linux 平台在编译 AIQ 库时会同时编译出 j2s\_4b 工具，以及生成 iq json 对应的 iq bin 文件；Android 平台不支持 iq bin 文件自动生成功能。
- 输入参数中 addr 对应的内存由应用管理，包含申请和释放。该地址通过该接口传递给 AIQ 后，AIQ 启动后其内部会频繁访问该地址，应用务必保证其在停止 AIQ 前不释放该内存。

### **rk\_aiq\_uapi2\_sysctl\_preInit\_calibproj**

#### 【描述】

ISP39 不支持

### **rk\_aiq\_uapi2\_sysctl\_preInit\_devBufCnt**

#### 【描述】

AIQ 半离线模式时，且 Raw 数据轮转由 AIQ 控制时，设置 vicap 轮转 buf 数量。

#### 【语法】

```
rk_aiq_uapi2_sysctl_preInit_devBufCnt(const char* sns_ent_name, const char* dev_ent,
                                         int buf_cnt);
```

#### 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
dev_ent	可固定设置成 "rkraw_tx"	输入
buf_cnt	buf 数量	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- 应先于rk\_aiq\_uapi2\_sysctl\_init 调用

## **rk\_aiq\_uapi2\_sysctl\_regHwEvtCb**

### 【描述】

注册硬件事件通知，用于同步应用和AIQ控制流程

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_regHwEvtCb (const char* sns_ent_name,  
                                rk_aiq_hwEvt_cb hwevt_cb,  
                                void* cb_ctx);
```

### 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
hwevt_cb	事件回调函数	输入
cb_ctx	事件回调上下文	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- 目前支持如下事件：

RK\_AIQ\_STATUS\_PREAMP\_DONE：快起模式时通知应用ISP首帧参数已经下发给ISP驱动，一般只在二次启动AIQ时使用，具体参考产品部快起相关文档

RK\_AIQ\_STATUS\_VICAP\_XXX：内部半离线模式时，用于通知应用vicap状态。应用可能需要在VICAP准备好后做一些特殊处理，如设置一些Sensor参数等，目前较少使用

## **rk\_aiq\_uapi2\_sysctl\_switch\_scene**

### 【描述】

初始化AIQ后，切换AIQ场景参数。

### 【语法】

```

int
rk_aiq_uapi2_sysctl_switch_scene (const rk_aiq_sys_ctx_t* sys_ctx,
    const char *main_scene,
    const char *sub_scene);

```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ 上下文指针	输入
main_scene	主场景，值在IQ文件中定义	输入
sub_scene	子场景，值在IQ文件中定义	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- 在 rk\_aiq\_uapi2\_sysctl\_init 后调用
- main\_scene 需要根据 IQ 文件中主场景数组来选择
- sub\_scene 需要根据 IQ 文件中主场景里的子场景数组来选择
- AIQ中为各场景参数分配了独立内存，模块API设置参数时，只作用于当前场景。因此，调用该API切换场景参数后，调用模块API设置的参数可能被覆盖导致失效。

## rk\_aiq\_uapi2\_sysctl\_init

### 【描述】

初始化AIQ上下文。

### 【语法】

```

rk_aiq_sys_ctx_t*
rk_aiq_uapi2_sysctl_init (const char* sns_ent_name,
    const char* iq_file_dir,
    rk_aiq_error_cb err_cb,
    rk_aiq_metas_cb metas_cb);

```

### 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
iq_file_dir	标定参数文件路径	输入
err_cb	出错回调函数，可为NULL	输入
metas_cb	meta数据回调函数，可为NULL	输入

#### 【返回值】

返回值	描述
rk_aiq_sys_ctx_t*	AIQ上下文指针

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 应在非preinit函数前调用。
- metas\_cb 目前仅用作通知已处理完某帧数据，应用可在该回调中调用模块API查询当前状态等。Android 中使用该回调来同步帧参数给应用。

## rk\_aiq\_uapi2\_sysctl\_deinit

#### 【描述】

反初始化AIQ上下文环境。

#### 【语法】

```
void
rk_aiq_uapi2_sysctl_deinit( 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
无	无

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 如rk\_aiq\_uapi2\_sysctl\_start已调用，则需要先调用 rk\_aiq\_uapi2\_sysctl\_stop

## **rk\_aiq\_uapi2\_sysctl\_prepare**

### **【描述】**

准备AIQ运行环境。

### **【语法】**

```
XCamReturn
rk_aiq_uapi2_sysctl_prepare(const rk_aiq_sys_ctx_t* ctx,
    uint32_t width,
    uint32_t height,
    rk_aiq_working_mode_t mode);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
width	sensor输出的分辨率宽度，保留参数	输入
height	sensor输出的分辨率高度，保留参数	输入
mode	ISP Pipeline工作模式(NORMAL/HDR)	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

### **【注意】**

- 应在rk\_aiq\_uapi2\_sysctl\_start函数之前调用。
- 如果需要在rk\_aiq\_uapi2\_sysctl\_start之后调用本函数，那么先调用rk\_aiq\_uapi2\_sysctl\_stop函数，再调用rk\_aiq\_uapi2\_sysctl\_prepare重新准备运行环境。
- 当ISP pipeline 信息有变化时，如输入的分辨率改变，HDR/Normal 切换，应用可以调用该API重新初始化AIQ信息，而不需要调用 rk\_aiq\_uapi2\_sysctl\_deinit/rk\_aiq\_uapi2\_sysctl\_init
- 需要在数据流启动之前调用，否则会导致ISP无初始参数，导致效果异常

## **rk\_aiq\_uapi2\_sysctl\_start**

### **【描述】**

启动AIQ控制系统。AIQ启动后，会不断的从ISP驱动获取3A统计信息，运行3A算法，并应用计算出的新参数。

### **【语法】**

```
XCamReturn  
rk_aiq_uapi2_sysctl_start(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

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

#### 【注意】

- 应在rk\_aiq\_uapi2\_sysctl\_prepare函数之后调用。
- AIQ 内部半离线模式时需要在 YUV  
数据流启动前调用，具体原因参见[软件流程](#)章节。

## rk\_aiq\_uapi2\_sysctl\_stop

#### 【描述】

停止AIQ控制系统。

#### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_stop(const rk_aiq_sys_ctx_t* ctx, bool keep_ext_hw_st);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
keep_ext_hw_st	无作用，可传入false	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_sysctl\_getStaticMetas**

### **【描述】**

查询sensor对应静态信息，如分辨率，数据格式等。

### **【语法】**

```
XCamReturn
rk_aiq_uapi2_sysctl_getStaticMetas(const char* sns_ent_name, rk_aiq_static_info_t* static_info);
```

### **【参数】**

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
static_info	静态信息结构体指针	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_sysctl\_enumStaticMetas**

### **【描述】**

枚举AIQ获取到的静态信息。推荐使用 API rk\_aiq\_uapi2\_sysctl\_enumStaticMetasByPhyId 替换。

### **【语法】**

```
XCamReturn
rk_aiq_uapi2_sysctl_enumStaticMetas(int index, rk_aiq_static_info_t* static_info);
```

### **【参数】**

参数名称	描述	输入/输出
index	索引号，从0开始	输入
static_info	静态信息结构体指针	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

#### 【注意】

- 该接口用于枚举系统中所有sensor信息，参数 index 从 0 开始，为逻辑编号，与内核DTS中 sensor的 ID 信息无关

### **rk\_aiq\_uapi2\_sysctl\_enumStaticMetasByPhyId**

#### 【描述】

获取与指定物理ID对应的Sensor信息。

#### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_enumStaticMetasByPhyId(int index, rk_aiq_static_info_t* static_info);
```

#### 【参数】

参数名称	描述	输入/输出
index	sensor物理ID	输入
static_info	静态信息结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

#### 【注意】

- 参数 index 为 sensor 物理 ID，值需要与内核DTS中 sensor 定义的 ID信息一致

### **rk\_aiq\_uapi2\_sysctl\_getBindedSnsEntNmByVd**

#### 【描述】

查询video结点所对应的sensor entity name。

#### 【语法】

```
const char* rk_aiq_uapi2_sysctl_getBindedSnsEntNmByVd(const char* vd);
```

#### 【参数】

参数名称	描述	输入/输出
vd	video路径, 如/dev/video20	输入

#### 【返回值】

返回值	描述
sensor entity name	字符串指针

#### 【注意】

- 参数必须为 MP (main path) 或者 SP (self path) 结点路径。

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_sysctl\_updateIq**

#### 【描述】

动态更新当前所使用的iq参数文件，不需要停止数据流。推荐使用 API rk\_aiq\_uapi2\_sysctl\_switch\_scene 替换。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_sysctl_updateIq(const rk_aiq_sys_ctx_t* sys_ctx, char* iqfile);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
iqfile	新的iq文件	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】

- iqfile 需要为全路径。
- 更新iq参数，并不意味着能切换运行模式，如需要切换hdr与normal，并不能通过更新 iq文件实现；但某些功能的切换却可以通过iq参数的不同配置来实现，如日、夜切换可完全通过iq配置来实现切换。

- 切换iq时，iq中的配置参数将会覆盖掉用户API的设置。如AWB模块，在iq中可配置手动、自动模式，那么执行该函数后，不管当前AWB处于何种模式，最终都会被新iq中的默认配置覆盖掉。

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_sysctl\_getCrop**

#### 【描述】

获取 VICAP crop参数，建议使用 Rockit 接口替换。

#### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_getCrop(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_rect_t *rect);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
rect	crop参数结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_sysctl\_setCrop**

#### 【描述】

设置 VICAP crop参数，建议使用 Rockit 接口替换。

#### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_setCrop(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_rect_t rect);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
rect	crop参数结构体指	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

### 【注意】

- 该接口会改变 sensor 输出RAW 分辨率，进而影响 ISP pipeline 分辨率
- 需要在 rk\_aiq\_uapi2\_sysctl\_init 后 rk\_aiq\_uapi2\_sysctl\_prepare 前使用

## **rk\_aiq\_uapi2\_sysctl\_getCamInfos**

### 【描述】

获取 AIQ 上下文所对应的 sensor 信息

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_getCamInfos(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_ctx_camInfo_t* camInfo);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
camInfo	AIQ上下文 所对应的 sensor 信息	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_get\_aiqversion\_info**

### 【描述】

获取 AIQ 版本信息

### 【语法】

```
void rk_aiq_uapi2_get_aiqversion_info(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_version_info_t* vers);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
vers	AIQ版本信息	输出

#### 【返回值】

- 无

### **rk\_aiq\_uapi2\_sysctl\_getModuleCtl**

#### 【描述】

获取ISP硬件模块使能信息。使用 API rk\_aiq\_uapi2\_sysctl\_getModuleEn 替代。

### **rk\_aiq\_uapi2\_sysctl\_setModuleCtl**

#### 【描述】

获取ISP硬件模块使能信息。使用 rk\_aiq\_uapi2\_sysctl\_setModuleEn 替代。

### **rk\_aiq\_uapi2\_sysctl\_getModuleEn**

#### 【描述】

获取ISP硬件模块使能信息。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_sysctl_getModuleEn(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_module_list_t* mod);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mod	模块使能列表	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_sysctl\_setModuleEn**

### **【描述】**

设置ISP硬件模块使能及Bypass

### **【语法】**

```
XCamReturn
rk_aiq_uapi2_sysctl_setModuleEn(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_module_list_t* mod);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mod	模块使能列表	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

### **【注意】**

- 模块API也有使能设置功能，该 API 只是提供一个所有模块使能配置的统一入口。
- 某些模块有同开同关的限制，此时必需调用该接口，而非分别调用模块接口。具体包括：
  - ynr, cnr, sharp 需要同开同关，如果需要单独关闭某个模块，可以用bypass替代
  - DRC 开启时，CAC 和 GIC 必须开启
 如果API参数（包括模块API）不满足这些限制时，AIQ会有log提示报错。更具体的约束参考《Rockchip\_Tuning\_Guide\_ISP39》文档，“ISP39 Introduction”“ISP39约束”章节

## **rk\_aiq\_uapi2\_sysctl\_setSharpFbcRotation**

### **【描述】**

ISP39不支持

## **rk\_aiq\_uapi2\_sysctl\_setMulCamConc**

### **【描述】**

设置 AIQ 工作在在线还是内部半离线模式，内部半离线模式时RAW数据由AIQ搬运。

### **【语法】**

```
void
rk_aiq_uapi2_sysctl_setMulCamConc(const rk_aiq_sys_ctx_t* ctx, bool cc);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
cc	true: 离线 false: 在线	输入

#### 【返回值】

无

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

#### 【注意】

- 需要在 rk\_aiq\_uapi2\_prepare 前调用。

### **rk\_aiq\_uapi2\_sysctl\_pause**

#### 【描述】

暂停vicap采集

#### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_pause(const rk_aiq_sys_ctx_t* ctx, bool isSingleMode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
isSingleMode	是否单帧模式	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

#### 【注意】

- 一般配合休眠唤醒流程使用。该接口可暂停vicap数据流，也包括停止sensor出流。
- 参数 isSingleMode 为true时，唤醒后vicap驱动会采集一帧数据，并自动停流，一般用于AOV产品形态；为false时，唤醒后恢复正常出流。

### **rk\_aiq\_uapi2\_sysctl\_resume**

### 【描述】

继续vicap采集

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_resume(const rk_aiq_sys_ctx_t* ctx);
```

### 【参数】

参数名称	描述	输入/输出
sys_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

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- 与pause接口相对，继续vicap采集。可调用该接口以便从单帧模式中恢复为正常模式。

## **rk\_aiq\_uapi2\_sysctl\_setListenStrmStatus**

### 【描述】

设置AIQ是否需要监听数据流启动停止消息

### 【语法】

```
void  
rk_aiq_uapi2_sysctl_setListenStrmStatus(rk_aiq_sys_ctx_t* sys_ctx, bool isListen);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
isListen	是否监听	输入

### 【返回值】

无

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- AIQ默认监听了数据流启动停止消息。如果为3A server等独立进程跑AIQ，且YUV数据流在另一进程中控制时，需要设置成false，监听动作需要在3A Server中自行实现。  
具体可参见[软件流程](#)章节。

## rk\_aiq\_uapi2\_sysctl\_setUserOtpInfo

### 【描述】

设置用户AWB等OTP参数

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_setUserOtpInfo(rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_user_otp_info_t otp_info);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
otp_info	OTP信息	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- OTP一般存在sensor模组的OTP中，AIQ会自动通过Sensor驱动接口读取并应用，不需要调用该接口进行设置。  
该接口适用于Sensor驱动未按标准实现OTP接口或者OTP信息并非存储在sensor模组OTP中。

## 数据类型

### rk\_aiq\_working\_mode\_t

#### 【说明】

AIQ pipeline工作模式

#### 【定义】

```
typedef enum {  
    RK_AIQ_WORKING_MODE_NORMAL,  
    RK_AIQ_WORKING_MODE_ISP_HDR2 = 0x10,  
    RK_AIQ_WORKING_MODE_ISP_HDR3 = 0x20,  
} rk_aiq_working_mode_t;
```

## 【成员】

成员名称	描述
RK_AIQ_WORKING_MODE_NORMAL	普通模式
RK_AIQ_WORKING_MODE_ISP_HDR2	两帧HDR模式
RK_AIQ_WORKING_MODE_ISP_HDR3	三帧HDR模式

## 【注意事项】

- 需要先查询sensor及AIQ所支持的模式，若设置的模式不支持则设置无效。

## rk\_aiq\_static\_info\_t

### 【说明】

AIQ 静态信息

### 【定义】

```
typedef struct {
    rk_aiq_sensor_info_t sensor_info;
    rk_aiq_lens_info_t lens_info;
    bool has_lens_vcm;
    bool has_fl;
    bool fl_strth_adj_sup;
    bool has_irc;
    bool fl_ir_strth_adj_sup;
} rk_aiq_static_info_t;
```

## 【成员】

成员名称	描述
sensor_info	sensor的名称、支持的分辨率等描述
lens_info	镜头信息
has_lens_vcm	是否带vcm
has_fl	是否带闪光灯
fl_strth_adj_sup	带闪光灯是否可调
bool has_irc	是否带IR-CUT
bool fl_ir_strth_adj_sup	

## rk\_aiq\_sensor\_info\_t

### 【说明】

sensor信息

### 【定义】

```

typedef struct {
    char sensor_name[32];
    rk_frame_fmt_t support_fmt[SUPPORT_FMT_MAX];
    int32_t num;
    /* binded pp stream media index */
    int8_t binded_strm_media_idx;
} rk_aiq_sensor_info_t;

```

### 【成员】

成员名称	描述
sensor_name	sensor的名称
support_fmt	支持的格式
num	支持的格式个数
has_fl	是否带闪光灯
binded_strm_media_idx	该sensor挂载的media节点号

## rk\_aiq\_rect\_t

### 【说明】

定义crop参数结构体

### 【定义】

```

typedef struct rk_aiq_rect_s {
    int left;
    int top;
    int width;
    int height;
} rk_aiq_rect_t;

```

### 【成员】

成员名称	描述
left	horizontal output offset
top	vertical output offset
width	horizontal output size
height	vertical output size

## rk\_aiq\_module\_ctl\_t

### 【说明】

模块使能结构体，注意并非所有模块都支持硬件使能和bypass控制

### 【定义】

```
typedef struct rk_aiq_module_ctl_s {
    camAlgoResultType type;
    bool en;
    bool bypass;
    rk_aiq_op_mode_t opMode;
} rk_aiq_module_ctl_t;
```

### 【成员】

成员名称	描述
type	模块类型
en	是否使能
bypass	是否bypass
opMode	模式，手动或者自动

## rk\_aiq\_module\_list\_t

### 【说明】

模块使能列表

### 【定义】

```
typedef struct rk_aiq_module_list_s {
    rk_aiq_module_ctl_t module_ctl[RESULT_TYPE_MAX_PARAM];
} rk_aiq_module_list_t;
```

### 【成员】

成员名称	描述
module_ctl	模块数组

## camAlgoResultType

### 【说明】

模块类型

### 【定义】

```
typedef enum _camAlgoResultType {
    RESULT_TYPE_INVALID = -1,
    ...
} camAlgoResultType;
```

### 【成员】

成员名称	描述
RESULT_TYPE_INVALID	非法值
RESULT_TYPE_EXPOSURE_PARAM	Sensor曝光，暂不使用
RESULT_TYPE_AEC_PARAM	AE算法模块
RESULT_TYPE_HIST_PARAM	Hist统计模块
RESULT_TYPE_AWB_PARAM	AWB算法模块
RESULT_TYPE_AF_PARAM	AF模块
RESULT_TYPE_DPCC_PARAM	DPCC模块
RESULT_TYPE_MERGE_PARAM	merge 模块
RESULT_TYPE_TMO_PARAM	不支持
RESULT_TYPE_CCM_PARAM	CCM 模块
RESULT_TYPE_LSC_PARAM	LSC模块
RESULT_TYPE_BLC_PARAM	BLC模块
RESULT_TYPE_RAWNR_PARAM	不支持
RESULT_TYPE_GIC_PARAM	GIC 模块
RESULT_TYPE_DEBAYER_PARAM	Debayer 模块
RESULT_TYPE_LDCH_PARAM	LDCH模块
RESULT_TYPE_LUT3D_PARAM	3DLut模块
RESULT_TYPE_DEHAZE_PARAM	不支持
RESULT_TYPE_AGAMMA_PARAM	Gamma 模块
RESULT_TYPE_ADEGAMMA_PARAM	不支持
RESULT_TYPE_WDR_PARAM	不支持
RESULT_TYPE_CSM_PARAM	CSM 模块
RESULT_TYPE_CGC_PARAM	CGC 模块
RESULT_TYPE_CONV422_PARAM	不支持
RESULT_TYPE_YUVCONV_PARAM	不支持
RESULT_TYPE_ADEGAMMA_PARAM	不支持
RESULT_TYPE_GAIN_PARAM	不支持
RESULT_TYPE_CP_PARAM	CP模块
RESULT_TYPE_IE_PARAM	IE模块
RESULT_TYPE_MOTION_PARAM	Motion模块

成员名称	描述
RESULT_TYPE_IRIS_PARAM	不支持
RESULT_TYPE_CPSL_PARAM	不支持
RESULT_TYPE_FLASH_PARAM	不支持
RESULT_TYPE_TNR_PARAM	BayerTnr 模块
RESULT_TYPE_YNR_PARAM	YNR模块
RESULT_TYPE_UVNR_PARAM	UVNR模块，等同于 CNR模块
RESULT_TYPE_CNR_PARAM	CNR模块
RESULT_TYPE_SHARPEN_PARAM	sharp模块
RESULT_TYPE_EDGEFLT_PARAM	不支持
RESULT_TYPE_ORB_PARAM	不支持
RESULT_TYPE_FOCUS_PARAM	不支持
RESULT_TYPE_DRC_PARAM	DRC模块
RESULT_TYPE_CAC_PARAM	CAC模块
RESULT_TYPE_AFD_PARAM	AFD模块
RESULT_TYPE_RGBIR_PARAM	不支持
RESULT_TYPE_TRANS_PARAM	数据位数压缩模块
RESULT_TYPE_LDC_PARAM	不支持
RESULT_TYPE_AESTATS_PARAM	不支持
RESULT_TYPE_HISTEQ_PARAM	HistEq 模块
RESULT_TYPE_ENH_PARAM	Enhance模块
RESULT_TYPE_TEXEST_PARAM	TexEst 模块
RESULT_TYPE_HSV_PARAM	不支持

## 离线帧处理

目前离线帧处理有以下几种方式：

### 1. AIQ API

AIQ 提供一组全离线模式 API，RAW数据和曝光信息以 RK RAW 格式传给 AIQ，AIQ 内部负责喂 RAW数据给 ISP 进行处理。

### 2. RawStream库

提供独立RawStream库来获取RAW，以及喂RAW给ISP，即将RAW数据流的操作从AIQ独立出来了。全离线模式时，AIQ提供API，应用需用调用相应API将离线帧曝光信息传递给AIQ；半离线时需要调用AIQ相应的API进行ISP参数和RAW数据流同步。RawStream库使用方式参考文档《Rockchip Development Guide RkRawStream.pdf》。此外注意，RawStream库目前只支持C++版本。

### 3. Rockit 方式

该方式基本等同于 RawStream库方式，Rockit 封装了RawStream库功能，提供了新的Raw数据流操作接口。具体参考Rockit文档。

**注意，以上方式，推荐优先使用 Rockit 方式，RawStream库次之，AIQ API 方式目前仅用作 debug 使用。**

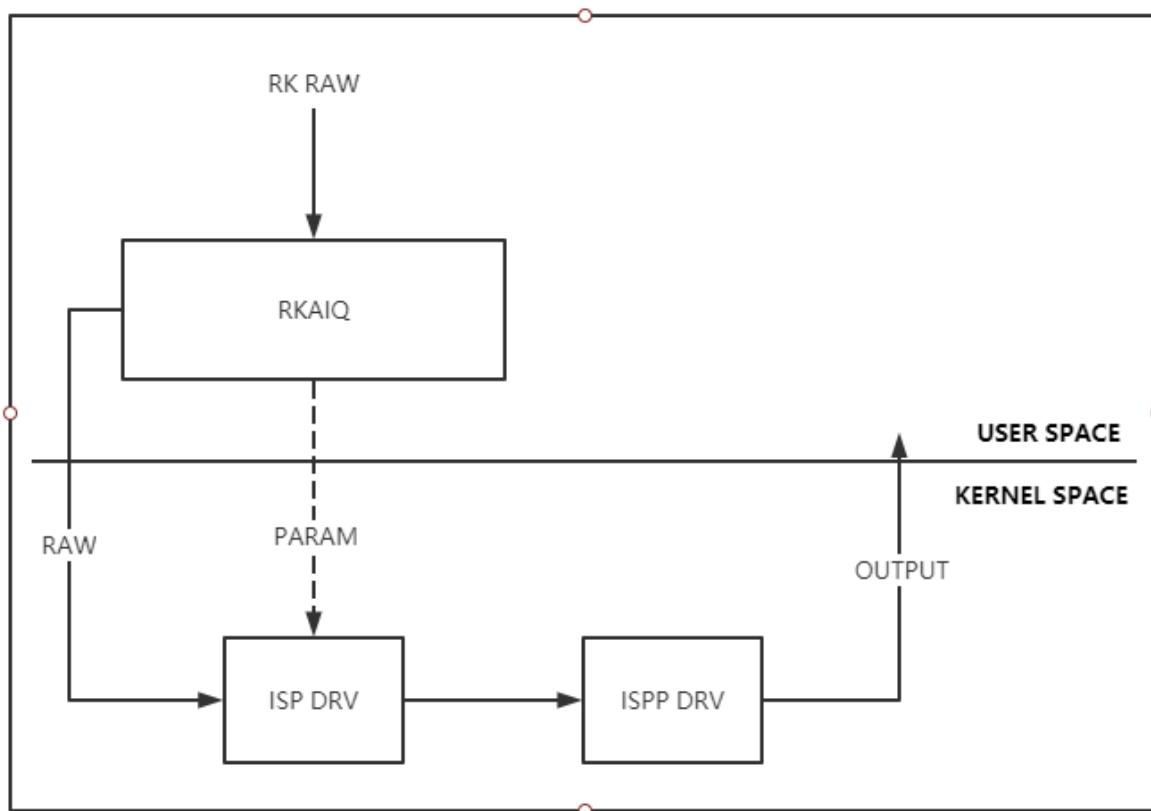
方式1可称作内部离线帧方式，2和3可称作外部离线帧方式。

## 内部离线帧

### 概述

RKAIQ提供离线RAW帧处理功能，即RK自定义的RAW格式文件经RKAIQ解析后送ISP处理，输出为可显示正常效果的图像的功能。

### 功能框图



离线帧处理框图

### 功能描述

- 支持RK-Raw文件输入。  
使用文件输入接口，调用进程将被阻塞，直到文件处理完成并成功输出。
- 支持RK-Raw buffer输入，异步处理模式。  
使用buffer输入接口，调用进程不会阻塞，buffer处理完成后将调用回调函数(如有注册回调函数)。

- 支持RK-Raw buffer输入，同步处理模式。  
使用buffer输入接口，调用进程将被阻塞，直到buffer处理完成并成功输出。

## RK-Raw格式说明

参见《RK RAW文件格式》说明文档

## 支持的RAW格式

支持raw8/raw10/raw12，支持BGGR/GBRG/GRBG/RGGB四种bayer格式。

## API参考

### rk\_aiq\_uapi2\_sysctl\_prepareRkRaw

#### 【描述】

配置 RkRaw 格式参数信息。

#### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_prepareRkRaw(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_raw_prop_t prop);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
prop	raw 格式信息	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 该种离线帧工作方式一般用于debug，可以使用RawStream或者Rockit方式替代
- AIQ 必须工作在 Fake Camera 模式。即 ISP 未连接真实sensor，或者有设置环境变量：  
linux平台：USE\_AS\_FAKE\_CAM  
Android平台：persist.vendor.rkisp.use\_as\_fake\_cam
- IQ文件需要命名成 FakeCameraX.json，其中 X 为 ISP 编号，具体值需要看FakeCamera 链接到哪个ISP上。也可调用 API rk\_aiq\_uapi\_sysctl\_preInit 配置指定的IQ文件。
- 与 API rk\_aiq\_uapi2\_sysctl\_prepareRkRaw，rk\_aiq\_uapi2\_sysctl\_enqueueRkRawFile，rk\_aiq\_uapi2\_sysctl\_enqueueRkRawBuf，rk\_aiq\_uapi2\_sysctl\_registRkRawCb 等一同使用。
- 注意与前述 API rk\_aiq\_uapi2\_sysctl\_rawReproc\_preInit 处理 RAW 方式相区别，具体参见[离线帧处理](#)

## **rk\_aiq\_uapi2\_sysctl\_enqueueRkRawBuf**

### **【描述】**

配置需要处理的RkRaw数据给AIQ。详细信息参考章节[离线帧处理](#)

### **【语法】**

```
XCamReturn  
rk_aiq_uapi2_sysctl_enqueueRkRawBuf(const rk_aiq_sys_ctx_t* sys_ctx, void *rawdata,  
bool syn);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
rawdata	RK RAW 格式数据	输入
syn	同步模式	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### **【注意】**

- 与 API rk\_aiq\_uapi2\_sysctl\_prepareRkRaw, rk\_aiq\_uapi2\_sysctl\_registRkRawCb 等一同使用。
- 参数 sync 为false时，为异步处理模式，该模式下需要调用 API rk\_aiq\_uapi2\_sysctl\_registRkRawCb注册回调，只有在回调被调用时，参数 rawdata 内容才可以变更。

## **rk\_aiq\_uapi2\_sysctl\_enqueueRkRawFile**

### **【描述】**

配置需要处理的RkRaw数据给AIQ。详细信息参考章节[离线帧处理](#)

### **【语法】**

```
XCamReturn  
rk_aiq_uapi2_sysctl_enqueueRkRawFile(const rk_aiq_sys_ctx_t* ctx, const char *path);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
path	RkRaw格式文件	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 与 API rk\_aiq\_uapi2\_sysctl\_prepareRkRaw一同使用。
- 参数 path 为存储RkRaw格式的Raw数据文件，目前只支持单帧。

## rk\_aiq\_uapi2\_sysctl\_registRkRawCb

#### 【描述】

注册RkRaw回调函数。详细信息参考章节[离线帧处理](#)

#### 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_registRkRawCb(const rk_aiq_sys_ctx_t* ctx, void (*callback)(void*));
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
callback	回调函数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

#### 【注意】

- 与 rk\_aiq\_uapi2\_sysctl\_enqueueRkRawBuf 一同使用。
- AIQ处理完一帧RAW后触发回调通知应用。

## **rk\_aiq\_uapi2\_sysctl\_setISPParamsDelayCnts**

### 【描述】

设置ISP参数延时生效帧

### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_setISPParamsDelayCnts(const rk_aiq_sys_ctx_t* ctx, , int8_t delay_cnts);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
delay_cnts	延时生效帧数	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

### 【注意】

- 默认生效帧为2，即由第n帧统计计算生成的ISP参数将在第n+2帧时生效。完全离线帧时，需要设置成1，即第n帧统计生成的ISP参数在第n+1帧时生效。
- delay\_cnts 设置成 INT8\_MAX 可恢复成默认生效帧。

## **数据结构**

### **rk\_aiq\_format\_t**

#### 【说明】

RAW 图存储格式

#### 【定义】

```
typedef enum {
...
RK_PIX_FMT_SBGGR10 = rk_fmt_fourcc('B', 'G', '1', '0'), /* 10 BGBG..GRGR.. */
RK_PIX_FMT_SGBRG10 = rk_fmt_fourcc('G', 'B', '1', '0'), /* 10 GBGB..RGGRG.. */
RK_PIX_FMT_SGRBG10 = rk_fmt_fourcc('B', 'A', '1', '0'), /* 10 GRGR..BGBG.. */
RK_PIX_FMT_SRGGGB10 = rk_fmt_fourcc('R', 'G', '1', '0'), /* 10RGRG.. GBGB.. */
...
} rk_aiq_format_t;
```

## 【成员】

成员名称	描述
RK_PIX_FMT_SBGGR10	RAW10 BGGR
...	RAW 存储格式

## rk\_aiq\_rawbuf\_type\_t

### 【说明】

与AIQ交互的RAW 图存储方式

### 【定义】

```
typedef enum rk_aiq_rawbuf_type_s {
    RK_AIQ_RAW_ADDR,
    RK_AIQ_RAW_FD,
    RK_AIQ_RAW_DATA,
    RK_AIQ_RAW_FILE
} rk_aiq_rawbuf_type_t;
```

## 【成员】

成员名称	描述
RK_AIQ_RAW_ADDR	指针地址
RK_AIQ_RAW_FD	文件描述符
RK_AIQ_RAW_FILE	文件

## rk\_aiq\_raw\_prop\_t

### 【说明】

RAW 基本信息

### 【定义】

```
typedef struct rk_aiq_raw_prop_s {
    uint32_t frame_width;
    uint32_t frame_height;
    rk_aiq_format_t format;
    rk_aiq_rawbuf_type_t rawbuf_type;
} rk_aiq_raw_prop_t;
```

## 【成员】

成员名称	描述
frame_width	RAW图分辨率宽
frame_height	RAW图分辨率高
format	RAW图格式
rawbuf_type	

## 注意事项

- 使用RK Raw数据处理功能，在创建AIQ Context时，rk\_aiq\_uapi2\_sysctl\_init接口的参数sns\_ent\_name须为“FakeCamera”。
- Raw数据的处理依赖IQ json效果文件，json文件生成时的分辨率应与传入的RK Raw帧数据的分辨率一致。效果文件须命名为FakeCamera.json，放置于json文件的加载路径下。
- 可通过 USE\_AS\_FAKE\_CAM (Linux) 或者 persist.vendor.rkisp.use\_as\_fake\_cam (Android) 环境变量强制进入FakeCam模式，该方式时则无前述限制。

## 参考示例

离线帧处理API的使用方法请参考rkisp\_demo，路径为 /external/camera\_engine\_rkaiq/rkisp\_demo  
以上离线帧处理方式为AIQ负责喂RAW数据给ISP硬件，为全离线模式，AIQ不控制Sensor曝光，RAW帧曝光信息需要在 RK RAW头信息中给出。该方式一般在调试时使用。

## 外部离线帧

### 概述

外部离线帧总体流程与内部离线帧类似，只是RAW数据由外部搬运。

### API参考

#### rk\_aiq\_uapi2\_sysctl\_rawReproc\_preInit

##### 【描述】

离线或半离线模式时，且Raw数据流由外部控制时，配置 ISP 链路信息，从而让 AIQ 知晓生成的参数对应的是哪个ISP的。

##### 【语法】

```
const char*
rk_aiq_uapi2_sysctl_rawReproc_preInit(const char* isp_driver,
                                         const char*
offline_sns_name,
                                         rk_aiq_frame_info_t two_frm_exp_info[2]);
```

##### 【参数】

参数名称	描述	输入/输出
isp_driver	isp 节点名称	输入
two_frm_exp_info	起始两帧曝光信息	输入

### 【返回值】

返回值	描述
非NULL	sensor entity name
NULL	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_sysctl.h
- 库文件: librkaiq.so

### 【注意】

- 应先于所有 AIQ API 调用
- 参数 isp\_driver 表示用哪一路 isp 处理 RAW，需要赋值成 media 拓扑信息中 model 字符串
- 返回值为 sensor entity name，需要在 rk\_aiq\_uapi2\_sysctl\_init 或者其他需要的preinit函数中传入
- 该接口配合 RawStream库或者应用控制 RAW 数据流方式时使用
- 参数 two\_frm\_exp\_info 为起始的2帧曝光信息，全离线时使用
- 与 rk\_aiq\_uapi2\_sysctl\_rawReproc\_genIspParams,  
rk\_aiq\_uapi2\_sysctl\_setIspParamsDelayCnts, rk\_aiq\_uapi2\_sysctl\_preInit\_rkrawstream\_info 等一同使用
- 用做全离线时，注意与 rk\_aiq\_uapi2\_sysctl\_prepareRkRaw 处理 RAW 方式相区别
- 用做半离线时，仅用作通知AIQ RAW由外部搬运，不需要AIQ参与

## rk\_aiq\_uapi2\_sysctl\_preInit\_rkrawstream\_info

### 【描述】

设置 ISP 处理的 RAW 数据帧格式等信息

### 【语法】

```
XCamReturn
rk_aiq_uapi2_sysctl_preInit_rkrawstream_info(const char* sns_ent_name,
                                                const rk_aiq_rkrawstream_info_t* info);
const rk_aiq_rkrawstream_info_t* info)
```

### 【参数】

参数名称	描述	输入/输出
sns_ent_name	sensor entity name	输入
info	RAW格式信息	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- 应先于rk\_aiq\_uapi2\_sysctl\_init 调用
- 该接口配合 RawStream库或者应用控制 RAW 数据流方式时使用
- 其中 info 参数内的 mode 用于区分半离线和全离线模式，半离线模式指的是AIQ需要控制sensor 曝光，全离线指的是AIQ不需要控制sensor曝光，帧曝光信息需要由 rk\_aiq\_uapi2\_sysctl\_rawReproc\_genISPParams 接口配置进来

## **rk\_aiq\_uapi2\_sysctl\_rawReproc\_genISPParams**

### 【描述】

配置曝光信息，并同步RAW数据流和ISP处理流程。

### 【语法】

```
void rk_aiq_uapi2_sysctl_rawReproc_genISPParams (rk_aiq_sys_t* sys_ctx,
                                                uint32_t sequence,
                                                rk_aiq_frame_info_t *next_frm_info,
                                                int mode);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ ctx	输入
sequence	帧号	输入
next_frm_info	下一帧曝光信息	输入
mode	0: 半离线 1: 离线	输入

### 【返回值】

无

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

### 【注意】

- next\_frm\_info 为下一帧曝光，全离线时需要配置

- AIQ 产生的ISP参数与RAW帧是帧匹配的，而应用RAW数据流和AIQ是异步执行，因此当使用外部离线方式时，需要流程同步。一般流程为：

取RAW

调用rk\_aiq\_uapi2\_sysctl\_rawReproc\_genISPParams，AIQ会等到产生完整帧参数并设置到ISP后返回

送RAW给ISP处理

- 外部半离线时，如果不考虑参数与RAW同步，可不调用该接口

## 数据结构

### `rk_aiq_rkrawstream_mode_t`

#### 【说明】

AIQ 工作模式

#### 【定义】

```
typedef enum {
    RK_ISP_RKRAWSTREAM_MODE_INVALID = 0,
    RK_ISP_RKRAWSTREAM_MODE_HALF_ONLINE,
    RK_ISP_RKRAWSTREAM_MODE_OFFLINE,
} rk_aiq_rkrawstream_mode_t;
```

#### 【成员】

成员名称	描述
RK_ISP_RKRAWSTREAM_MODE_INVALID	非法
RK_ISP_RKRAWSTREAM_MODE_HALF_ONLINE	AIQ外部半离线模式
RK_ISP_RKRAWSTREAM_MODE_OFFLINE	AIQ外部离线模式

### `rk_aiq_rkrawstream_info_t`

#### 【说明】

RAW 帧基本信息

#### 【定义】

```
typedef struct {
    int width;
    int height;
    rk_aiq_format_t format;
    rk_aiq_rkrawstream_mode_t mode;
} rk_aiq_rkrawstream_info_t;
```

#### 【成员】

成员名称	描述
width	RAW 图分辨率宽
height	RAW 图分辨率高
format	RAW 存储格式
mode	离线或者半离线

### **rk\_aiq\_frame\_info\_t**

#### 【说明】

RKRAW 离线帧信息，具体参考《RK RAW文件格式》说明文档

## Camera组

### 概述

如环视等应用场景，需要将多个Camera归为一组做统一管理，曝光等参数需要统一设置等。Camera组相关API，可通过创建组，将多个Camera归类为同一组。

除以下流程性相关的组API外，其他模块API及功能性API，如无特殊说明，适用于单个AIQ的API也同样适用于Camera组，AIQ内部实现会根据传递的Context类型将请求分发给Camera组或者单Camera。

需要注意的是，Camera组的流程控制API与单摄不同，需要替换的API对应关系如下，单摄时使用如下API控制AIQ流程：

单摄	组Camera
rk_aiq_uapi2_sysctl_init	rk_aiq_uapi2_camgroup_create
rk_aiq_uapi2_sysctl_prepare	rk_aiq_uapi2_camgroup_prepare
rk_aiq_uapi2_sysctl_start	rk_aiq_uapi2_camgroup_start
rk_aiq_uapi2_sysctl_stop	rk_aiq_uapi2_camgroup_stop
rk_aiq_uapi2_sysctl_deinit	rk_aiq_uapi2_camgroup_destroy

### **rk\_aiq\_uapi2\_camgroup\_create**

#### 【描述】

创建Camera组Ctx，会为每个Camera创建独立的 AIQ ctx，然后交给 Group Ctx 管理。

#### 【语法】

```
rk_aiq_camgroup_ctx_t*
rk_aiq_uapi2_camgroup_create (rk_aiq_camgroup_instance_cfg_t* cfg);
```

#### 【参数】

参数名称	描述	输入/输出
cfg	配置参数	输入

## 【返回值】

返回值	描述
rk_aiq_camgroup_ctx_t*	组运行环境
NULL	失败

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

## 【注意】

- 与 rk\_aiq\_uapi2\_sysctl\_init 相区分, rk\_aiq\_uapi2\_sysctl\_init 创建单camera的 AIQ 运行环境。使用 rk\_aiq\_uapi2\_sysctl\_init 创建运行环境的sensor entity, 不得再用 rk\_aiq\_uapi2\_camgroup\_create 创建组运行环境, 反之亦然。
- 由于默认选择的IQ场景是 normal-day, 如果需要为HDR或者其他模式, 需要使用 rk\_aiq\_uapi2\_sysctl\_prelinit\_scene 进行预先设置

## **rk\_aiq\_uapi2\_camgroup\_prepare**

### 【描述】

根据选择的pipeline模式, 生成初始化参数以及建立pipeline。会调用每个Camera对应AIQ ctx的 rk\_aiq\_uapi2\_sysctl\_prepare。

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_camgroup_prepare (rk_aiq_camgroup_ctx_t* camgroup_ctx,  
                                rk_aiq_working_mode_t mode);
```

### 【参数】

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入
mode	pipeline工作模式	输入

## 【返回值】

返回值	描述
0	成功
非0	参见错误码

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_start**

### 【描述】

运行Camera组。会调用每个Camera对应AIQ ctx的 rk\_aiq\_uapi2\_sysctl\_start。

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_camgroup_start (rk_aiq_camgroup_ctx_t* camgroup_ctx);
```

### 【参数】

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，参见错误码

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_camgroup.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_stop**

### 【描述】

停止Camera组。会调用每个Camera对应AIQ ctx的 rk\_aiq\_uapi2\_sysctl\_stop。

### 【语法】

```
XCamReturn  
rk_aiq_uapi2_camgroup_stop (rk_aiq_camgroup_ctx_t* camgroup_ctx);
```

### 【参数】

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，参见错误码

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_camgroup.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_destroy**

### **【描述】**

销毁Camera组。会调用每个Camera对应AIQ ctx的 rk\_aiq\_uapi2\_sysctl\_deinit。

### **【语法】**

```
XCamReturn  
rk_aiq_uapi2_camgroup_destroy (rk_aiq_camgroup_ctx_t* camgroup_ctx);
```

### **【参数】**

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，参见错误码

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_camgroup.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_getOverlapMap**

### **【描述】**

环视应用场景下，从CameraGroup上下文获取各camera的重叠区域信息。

### **【语法】**

```
struct RK_PS_SrcOverlapMap*  
rk_aiq_uapi2_camgroup_getOverlapMap (rk_aiq_camgroup_ctx_t* camgroup_ctx);
```

### **【参数】**

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入

### **【返回值】**

返回值	描述
struct RK_PS_SrcOverlapMap*	重叠区域信息结构体指针
NULL	失败

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_getOverlapMap**

### **【描述】**

环视应用场景下，从文件获取各camera的重叠区域信息。

### **【语法】**

```
XCamReturn
rk_aiq_uapi2_camgroup_getOverlapMap_from_file (const char* map_file,
                                                struct RK_PS_SrcOverlapMap** overlapMap);
```

### **【参数】**

参数名称	描述	输入/输出
map_file	存储重叠信息的文件	输入
overlapMap	存储获取到的重叠信息	输入, 输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，参见错误码

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_camgroup\_getAiqCtxBySnsNm**

### **【描述】**

通过组内 Sensor 实体名获取对应的单个Camera的AIQ上下文。

### **【语法】**

```
rk_aiq_sys_ctx_t*
rk_aiq_uapi2_camgroup_getAiqCtxBySnsNm (rk_aiq_camgroup_ctx_t* camgroup_ctx,
                                           const char* sns_entity_name);
```

### **【参数】**

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入
sns_entity_name	组内sensor实体名称	输入

### **【返回值】**

返回值	描述
rk_aiq_sys_ctx_t*	获取到的AIQ运行环境
NULL	失败

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_camgroup\_getCamInfos**

#### 【描述】

获取组内camera的信息，如Sensor实体名称等。

#### 【语法】

```
XCamReturn
rk_aiq_uapi2_camgroup_getCamInfos (rk_aiq_camgroup_ctx_t* camgroup_ctx,
rk_aiq_camgroup_camInfos_t* camInfos);
```

#### 【参数】

参数名称	描述	输入/输出
camgroup_ctx	组运行环境	输入
camInfos	获取到的组内camera信息	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，参见错误码

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_camgroup.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_camgroup\_bind**

#### 【描述】

预留接口，未实现。

### **rk\_aiq\_uapi2\_camgroup\_unbind**

#### 【描述】

预留接口，未实现。

### **rk\_aiq\_uapi2\_sysctl\_setSnsSyncMode**

#### 【描述】

环视模式时，设置多sensor间硬件帧同步模式

## 【语法】

```
XCamReturn  
rk_aiq_uapi2_sysctl_setSnsSyncMode(rk_aiq_sys_ctx_t* sys_ctx, enum rkmodule_sync_mode  
sync_mode);
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
sync_mode	sensor同步模式	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

## 【注意】

- 需要 sensor 支持硬同步方式
- AIQ内部默认为 INTERNAL\_MASTER 和 EXTERNAL\_MASTER 同步方式
- 如需修改，该接口需要在rk\_aiq\_uapi2\_camgroup\_prepare 后，rk\_aiq\_uapi2\_camgroup\_start 前调用
- 各sensor的AIQ 上下文可通过 rk\_aiq\_uapi2\_camgroup\_getCamInfos 获取。

## 数据结构

### **rkmodule\_sync\_mode**

#### 【说明】

sensor间硬件同步方式，硬件同步方式可参考驱动文档《Rockchip\_Driver\_Guide\_VI》。

### **rk\_aiq\_camgroup\_instance\_cfg\_t**

#### 【说明】

Camera Group 配置参数

#### 【定义】

```

typedef struct rk_aiq_camgroup_instance_cfg_s {
    const char* sns_ent_nm_array[RK_AIQ_CAM_GROUP_MAX_CAMS];
    int sns_num;
    const char* config_file_dir;
    /* followings are relative path to config_file_dir */
    const char* single_iq_file;
    const char* group_iq_file;
    const char* overlap_map_file;
} rk_aiq_camgroup_instance_cfg_t;

```

### 【成员】

成员名称	描述
sns_ent_nm_array	需要加入组管理的Sensor实体 数组
sns_num	需要加入组管理的Sensor实体个数
config_file_dir	指定IQ文件路径
single_iq_file	指定IQ文件名称，可为NULL， NULL时由AIQ自动选择
group_iq_file	指定Camera组IQ配置文件，目前暂不使用
overlap_map_file	指定Camera重叠信息文件， 可为NULL

### 【注意】

- 如未调用 rk\_aiq\_uapi2\_sysctl\_setSnsSyncMode，那么 sns\_ent\_nm\_array 中数组第一个Sensor 同步模式会被设置成 INTERNAL\_MASTER\_MODE，其他Sensor同步模式会被设置成 EXTERNAL\_MASTER

## **rk\_aiq\_camgroup\_camInfos\_t**

### 【说明】

Camera 组内各Camera信息

### 【定义】

```

typedef struct rk_aiq_camgroup_camInfos_s {
    int valid_sns_num;
    const char* sns_ent_nm[RK_AIQ_CAM_GROUP_MAX_CAMS];
    int sns_camPhyId[RK_AIQ_CAM_GROUP_MAX_CAMS];
} rk_aiq_camgroup_camInfos_t;

```

### 【成员】

成员名称	描述
valid_sns_num	组内包含的Sensor实体数量
sns_ent_nm	组内包含的Sensor实体名称数组
sns_camPhyId	组内包含的Sensor实体对应的物理ID数组

## **struct RK\_PS\_SrcOverlapMap**

## 【说明】

定义Camera重叠区域信息

## 【定义】

```
struct RK_PS_SrcOverlapMap
{
    char versionInfo[64];
    RK_PS_SrcOverlapPosition srcOverlapPositon[8];
    unsigned char overlapMap[15 * 15 * 8];
};
```

## 【成员】

成员名称	描述
versionInfo	版本信息
srcOverlapPositon	各Camera的安装时的旋转方向，可为0,90,180,270度
overlapMap	8个camera每个统计块的重叠信息，具体如下： 每个相机的统计数据是15*15列，总共8个相机(编号0~7)，实际不足8个相机的，相应位置用0补齐。按行存储，存完8个相机的第一行后，紧接着存第二行。以此类推。 取值范围为0~255，完全不重叠则值为0，完全重叠则值为255，部分重叠根据占比进行插值 camera0 row0; camera1 row0; camera2 row0; camera3 row0; camera4 row0; camera5 row0; camera6 row0; camera7 row0; camera0 row1; camera1 row1; camera2 row1; camera3 row1; camera4 row1; camera5 row1; camera6 row1; camera7 row1; ..... camera0 row14; camera1 row14; camera2 row14; camera3 row14; camera4 row14; camera5 row14; camera6 row14; camera7 row14;

# AE

## 概述

AE 模块实现的功能是：根据自动测光系统获得当前图像的曝光量，再自动配置镜头光圈、 sensor 快门及增益来获得最佳的图像质量。

## 重要概念

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

## 功能描述

AE 模块由AE 统计信息及 AE 控制策略的算法两部分组成。

## 功能级API参考

参考章节 [IMGPROC功能级API](#) AE 相关

## 模块级API参考

### rk\_aiq\_user\_api2\_ae\_setExpSwAttr

#### 【描述】

设定 AE曝光软件属性。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ae_setExpSwAttr(const rk_aiq_sys_ctx_t* ctx,  
const ae_api_expSwAttr_t expSwAttr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
expSwAttr	AE公共功能控制参数结构体	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

#### 【举例】

##### 设置手动曝光属性

曝光分量包括sensor曝光时间、sensor曝光增益、isp数字增益。设置手动曝光模式之后，还需要分别设置各曝光分量的手动状态（sw\_aeT\_manTime\_en、sw\_aeT\_manGain\_en、sw\_aeT\_manIspDGain\_en）及其对应手动值。设置的各曝光分量的值，会受到sensor的限制。如设置的曝光分量值超过sensor限制，算法内部会自动进行校正。

```
ae_api_expSwAttr_t expSwAttr;  
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);  
expSwAttr.commCtrl.sw_aeT_opt_mode = RK_AIQ_OP_MODE_MANUAL;  
//LinearAE  
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manGain_en = true;  
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manTime_en = true;  
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manGain_en = 1.0f; /*gain = 1x*/
```

```

expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manTime_val = 0.02f; /*time = 1/50s*/

//HdrAE (should set all frames)
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manGain_en = true;
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_en = true;
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manGain_val[0] = 1.0f; /*sframe gain = 1x*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[0] = 0.002f; /*sframe time = 1/500s*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manGain_val[1] = 2.0f; /*mframe gain = 2x*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[1] = 0.01f; /*mframe time = 1/100s*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manGain_val[2] = 4.0f; /*lframe gain = 4x*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[2] = 0.02f; /*lframe time = 1/50s*/

ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

```

## 设置自动曝光属性

自动曝光可以设置曝光分量的作用范围，若设置的曝光分量范围超过sensor的限制，算法内部会自动进行校正。

【注】设置曝光分量range的参数与获取曝光分量range的参数不同。

```

//(1)set time range in struct "advanced"
ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);

expSwAttr.advanced.sw_aeT_advAeRange_en = true; /*must enable*/
//LinAE
expSwAttr.advanced.linExpRange.sw_aeT_time_max = 0.04f; /*time_max = 0.04*/
expSwAttr.advanced.linExpRange.sw_aeT_time_min = 0.001f; /*time_min = 0.001*/
//HdrAE
expSwAttr.advanced.hdrExpRange[0].sw_aeT_time_max = 0.002f; /*sframe time_max = 0.02*/
expSwAttr.advanced.hdrExpRange[0].sw_aeT_time_min = 0.001f; /*sframe time_min = 0.01*/
expSwAttr.advanced.hdrExpRange[1].sw_aeT_time_max = 0.003f; /*mframe time_max = 0.03*/
expSwAttr.advanced.hdrExpRange[1].sw_aeT_time_min = 0.002f; /*mframe time_min = 0.02*/
expSwAttr.advanced.hdrExpRange[2].sw_aeT_time_max = 0.04f; /*lframe time_max = 0.03*/
expSwAttr.advanced.hdrExpRange[2].sw_aeT_time_min = 0.03f; /*lframe time_min = 0.02*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

//(2)set gain range in struct "stAdvanced"
expSwAttr.advanced.sw_aeT_advAeRange_en = true; /*must enable*/
//LinAE
expSwAttr.advanced.linExpRange.sw_aeT_gain_max = 32.0f; /*gain_max = 32x*/
expSwAttr.advanced.linExpRange.sw_aeT_gain_min = 1.0f; /*gain_min = 1x*/
//HdrAE
expSwAttr.advanced.hdrExpRange[0].sw_aeT_gain_max = 32.0f; /*sframe gain_max = 2x*/
expSwAttr.advanced.hdrExpRange[0].sw_aeT_gain_min = 1.0f; /*sframe gain_min = 1x*/
expSwAttr.advanced.hdrExpRange[1].sw_aeT_gain_max = 64.0f; /*mframe gain_max = 64x*/
expSwAttr.advanced.hdrExpRange[1].sw_aeT_gain_min = 1.0f; /*mframe gain_min = 1x*/
expSwAttr.advanced.hdrExpRange[2].sw_aeT_gain_max = 64.0f; /*lframe gain_max = 64x*/
expSwAttr.advanced.hdrExpRange[2].sw_aeT_gain_min = 1.0f; /*lframe gain_min = 1x*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

//(3)get time & gain range
ae_api_queryInfo_t queryInfo;
memset(&queryInfo, 0, sizeof(ae_api_queryInfo_t));
ret = rk_aiq_user_api2_ae_queryExpResInfo(ctx, &queryInfo);

```

```

printf("linear time range=[%f,%f]\n",
    queryInfo.linExpInfo.expRange.sw_aeT_time_min,
    queryInfo.linExpInfo.expRange.sw_aeT_time_max);
printf("linear gain range=[%f,%f]\n",
    queryInfo.linExpInfo.expRange.sw_aeT_gain_min,
    queryInfo.linExpInfo.expRange.sw_aeT_gain_max);

printf("hdr stime range=[%f,%f], mtime range=[%f,%f], ltime range=[%f,%f]\n",
    queryInfo.hdrExpInfo.expRange[0].sw_aeT_time_min,
    queryInfo.hdrExpInfo.expRange[0].sw_aeT_time_max,
    queryInfo.hdrExpInfo.expRange[1].sw_aeT_time_min,
    queryInfo.hdrExpInfo.expRange[1].sw_aeT_time_max,
    queryInfo.hdrExpInfo.expRange[2].sw_aeT_time_min,
    queryInfo.hdrExpInfo.expRange[2].sw_aeT_time_max);
printf("hdr sgain range=[%f,%f], mgain range=[%f,%f], lgain range=[%f,%f]\n",
    queryInfo.hdrExpInfo.expRange[0].sw_aeT_gain_min,
    queryInfo.hdrExpInfo.expRange[0].sw_aeT_gain_max,
    queryInfo.hdrExpInfo.expRange[1].sw_aeT_gain_min,
    queryInfo.hdrExpInfo.expRange[1].sw_aeT_gain_max,
    queryInfo.hdrExpInfo.expRange[2].sw_aeT_gain_min,
    queryInfo.hdrExpInfo.expRange[2].sw_aeT_time_max);

```

## 设置半手动曝光属性

半手动曝光是曝光分量（sensor曝光时间、sensor曝光增益、isp数字增益）中至少有一个曝光分量为手动状态，其他曝光分量为自动状态，否则将报错退出。例如，为了实现曝光增益手动，曝光时间、isp数字增益自动的半自动曝光功能，有以下两种实现方式：方式一，在手动曝光模式中，将曝光时间、isp数字增益的手动使能设为false，设置曝光增益的手动使能为true，并设置对应手动值；方式二，在自动曝光模式下，将曝光增益的最大最小值均设为需要固定的值。

**【注】** 方式一：HDR曝光模式下，所有帧的手动行为是同步的（即各帧的对应曝光分量手动行为一致），同时需保证长帧最大曝光大于短帧最大曝光，长帧最小曝光大于短帧最小曝光；方式二：HDR曝光模式下，各帧的手动行为允许不一致（如短帧增益手动，长帧增益可为自动），同时需保证长帧最大曝光大于短帧最大曝光，长帧最小曝光大于短帧最小曝光。

```

//Method One
ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.commCtrl.sw_aeT_opt_mode = RK_AIQ_OP_MODE_MANUAL;
//LinearAE
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manGain_en = false;
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manTime_en = true;
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manIspDGain_en = false;
expSwAttr.commCtrl.meCtrl.linMe.sw_aeT_manTime_val = 0.01f; /*time = 0.01 s*/
//HdrAE (NOTE: hdr total int time should < 1/fps)
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manGain_en = false;
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_en = true;
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manIspDGain_en = false;
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[0] = 0.008f; /*sframe time = 0.008s*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[1] = 0.01f; /*mframe time = 0.01s*/
expSwAttr.commCtrl.meCtrl.hdrMe.sw_aeT_manTime_val[2] = 0.01f; /*lframe time = 0.01s*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

//Method Two
ae_api_expSwAttr_t expSwAttr;

```

```

ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.commCtrl.sw_aeT_opt_mode = RK_AIQ_OP_MODE_MANUAL;
//set gain range
expSwAttr.advanced.sw_aeT_advAeRange_en = true; /*must enable*/
//LinAE
expSwAttr.advanced.linExpRange.sw_aeT_gain_max = 2.0f; /*gain_max = 2x*/
expSwAttr.advanced.linExpRange.sw_aeT_gain_min = 2.0f; /*gain_min = 2x*/
//HdrAE (allow to set only one frame)
expSwAttr.advanced.hdrExpRange[0].sw_aeT_gain_max = 2.0f; /*sframe gain_max = 2x*/
expSwAttr.advanced.hdrExpRange[0].sw_aeT_gain_min = 2.0f; /*sframe gain_min = 2x*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

```

## 设置固定帧率或自动降帧

固定帧率模式下，允许设置的帧率不得超过驱动序列表中定义的最大帧率，如超过会有log提醒，且帧率将设置为驱动所允许的最大帧率。

```

//set fixed framemode
ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.commCtrl.frmRate.sw_aeT_frmRate_mode = ae_frmRate_fix_mode
expSwAttr.commCtrl.frmRate.sw_aeT_frmRate_val = 20; /*fps = 20*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

//set auto framemode
ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.commCtrl.frmRate.sw_aeT_frmRate_mode = ae_frmRate_auto_mode
/*一般自动降帧模式由tunning人员事先配置好最低帧率和切换帧率对应的gain值*/
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

```

## 设置曝光调节速度及延迟帧数

```

ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
//set ae speed
expSwAttr.commCtrl.speed.sw_aeT_damp_over = 0.8f;
expSwAttr.commCtrl.speed.sw_aeT_damp_dark2Bright = 0.8f;
expSwAttr.commCtrl.speed.sw_aeT_damp_under = 0.8f;
expSwAttr.commCtrl.speed.sw_aeT_damp_bright2Dark = 0.8f;
//set ae delay
expSwAttr.commCtrl.delay.sw_aeT_delay_mode = ae_delay_frame_mode; //以帧为单位进行延迟
expSwAttr.commCtrl.delay.sw_aeT_whiteDelay_val = 10;
expSwAttr.commCtrl.delay.sw_aeT_blackDelay_val = 15;
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

```

## 设置抗闪功能

```

ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);
//set antiflicker mode
expSwAttr.commCtrl.antiFlicker.sw_aeT_antiFlicker_en = true;
expSwAttr.commCtrl.antiFlicker.sw_aeT_antiFlicker_freq = ae_antiFlicker_50hz_freq;
expSwAttr.commCtrl.antiFlicker.sw_aeT_antiFlicker_mode = ae_antiFlicker_auto_mode;
ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);

```

## 设置AE权重

设置15X15权重，算法内部根据硬件实际分块规格，进行权重的压缩。目前有两种设置权重的方式：方式一直接修改json中的权重；方式二：通过stadvanced修改权重，若消除使能，即可还原回json中的权重（推荐使用这种方式）。

针对人脸应用建议使用方式二，出现人脸时expSwAttr.stAdvanced.enable = true，使用expSwAttr.stAdvanced结构体中的权重，人脸消失时expSwAttr.stAdvanced.enable = false，使用原权重。

```
ae_api_expSwAttr_t expSwAttr;
ret = rk_aiq_user_api2_ae_getExpSwAttr(ctx, &expSwAttr);

uint8_t GridWeights[225]={
0, 0, 1, 2, 2, 3, 4, 5, 4, 3, 2, 2, 1, 0, 0,
0, 1, 2, 3, 3, 4, 5, 6, 5, 4, 3, 3, 2, 1, 0,
1, 2, 3, 5, 5, 6, 7, 8, 7, 6, 5, 5, 3, 2, 1,
2, 3, 5, 7, 7, 8, 9, 10, 9, 8, 7, 7, 5, 3, 2,
2, 3, 5, 7, 8, 9, 10, 11, 10, 9, 8, 7, 5, 3, 2,
2, 4, 6, 8, 9, 10, 11, 12, 11, 10, 9, 8, 6, 4, 2,
2, 4, 6, 9, 10, 11, 12, 13, 12, 11, 10, 9, 6, 4, 2,
3, 5, 7, 10, 11, 12, 13, 14, 13, 12, 11, 10, 7, 5, 3,
2, 4, 6, 9, 10, 11, 12, 13, 12, 11, 10, 9, 6, 4, 2,
2, 4, 6, 8, 9, 10, 11, 12, 11, 10, 9, 8, 6, 4, 2,
2, 3, 5, 7, 8, 9, 10, 11, 10, 9, 8, 7, 5, 3, 2,
2, 3, 5, 7, 7, 8, 9, 10, 9, 8, 7, 7, 5, 3, 2,
1, 2, 4, 6, 6, 7, 8, 9, 8, 7, 6, 6, 4, 2, 1,
0, 1, 3, 5, 5, 6, 7, 8, 7, 6, 5, 5, 3, 1, 0,
0, 1, 3, 5, 5, 6, 7, 8, 7, 6, 5, 5, 3, 1, 0
};

//method one:
memcpy(expSwAttr.commCtrl.sw_aeT_grid_wgt, GridWeights,
sizeof(expSwAttr.commCtrl.sw_aeT_grid_wgt));

//method two:
expSwAttr.advanced.sw_aeT_advGridWgt_en = true; //important! true means preferring to use these
parameters
memcpy(expSwAttr.advanced.sw_aeT_advGrid_wgt, GridWeights,
sizeof(expSwAttr.advanced.sw_aeT_advGrid_wgt));

ret = rk_aiq_user_api2_ae_setExpSwAttr(ctx, expSwAttr);
```

## rk\_aiq\_user\_api2\_ae\_getExpSwAttr

### 【描述】

获取 AE 曝光软件属性。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_ae_getExpSwAttr(const rk_aiq_sys_ctx_t* ctx,
ae_api_expSwAttr_t* pExpSwAttr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pExpSwAttr	AE曝光软件属性结构体指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

## rk\_aiq\_user\_api2\_ae\_setLinExpAttr

### 【描述】

设置AE线性模式曝光参数。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_ae_setLinExpAttr(const rk_aiq_sys_ctx_t* ctx, const ae_api_linExpAttr_t linExpAttr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
linExpAttr	AE曝光参数结构体	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

### 【举例】

设置AE线性模式曝光参数，包括但不限于：设置调整亮度力度sw\_aeT\_evBias\_strg，设置背光补偿功能BackLightCtrl，设置强光抑制功能OverExpCtrl等。值得注意的是，背光补偿和强光抑制不能同时使用。

## 设置线性曝光亮度力度

```

ae_api_linExpAttr_t LinExpAttr;
memset(&LinExpAttr,0,sizeof(ae_api_linExpAttr_t));
ret = rk_aiq_user_api2_ae_getLinExpAttr(ctx, &linExpAttr);
//set Evbias
printf("Evbias=%f\n", LinExpAttr.sw_aeT_evBias_strg);
LinExpAttr.sw_aeT_evBias_strg = 100.0f;

ret = rk_aiq_user_api2_ae_setLinExpAttr(ctx, linExpAttr);

```

## 设置背光补偿功能

```

ae_api_linExpAttr_t LinExpAttr;
memset(&LinExpAttr,0,sizeof(ae_api_linExpAttr_t));
ret = rk_aiq_user_api2_ae_getLinExpAttr(ctx, &linExpAttr);
//set BackLightCtrl
LinExpAttr.backLightCtrl.sw_aeT_backLit_en = true;
LinExpAttr.backLightCtrl.sw_aeT_backLitBias_strg = 200.0f;

ret = rk_aiq_user_api2_ae_setLinExpAttr(ctx, linExpAttr);

```

## 设置强光抑制功能

```

ae_api_linExpAttr_t LinExpAttr;
memset(&LinExpAttr,0,sizeof(ae_api_linExpAttr_t));
ret = rk_aiq_user_api2_ae_getLinExpAttr(ctx, &linExpAttr);
//set OverExpCtrl
LinExpAttr.overExpCtrl.sw_aeT_overExp_en = true;
LinExpAttr.overExpCtrl.sw_aeT_overExpBias_strg = 100.0f;

ret = rk_aiq_user_api2_ae_setLinExpAttr(ctx, linExpAttr);

```

## **rk\_aiq\_user\_api2\_ae\_getLinExpAttr**

### 【描述】

获取AE线性模式曝光参数。

### 【语法】

```

XCamReturn
rk_aiq_user_api2_ae_getLinExpAttr(const rk_aiq_sys_ctx_t* ctx, ae_api_linExpAttr_t* pLinExpAttr);

```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pLinExpAttr	AE曝光参数结构体指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_ae\_setHdrExpAttr**

### 【描述】

设置AE HDR模式曝光参数。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_ae_setHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, const ae_api_hdrExpAttr_t
hdrExpAttr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
hdrExpAttr	AE曝光参数结构体	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

### 【举例】

设置AE HDR模式曝光参数，包括但不限于：

### 设置固定/动态曝光比

```

ae_api_hdrExpAttr_t HdrExpAttr;
memset(&HdrExpAttr,0,sizeof(ae_api_hdrExpAttr_t));
ret = rk_aiq_user_api2_ae_getHdrExpAttr(ctx, &HdrExpAttr);
int len = 6;
float exp_level[6]={0,0.2,0.4,0.8,1.2,1.5};
float ratio_max[6]={32,30,28,26,24,22};
// set ExpRatioType (AUTO/FIX)
HdrExpAttr.expRatioCtrl.sw_aeT_expRatio_mode = ae_expRatio_auto_mode;
HdrExpAttr.expRatioCtrl.expRatio.sw_aeT_expRatio_len = len;
memcpy(HdrExpAttr.expRatioCtrl.expRatio.sw_aeT_expLevel_dot,exp_level,len*sizeof(float));
memcpy(HdrExpAttr.expRatioCtrl.expRatio.sw_aeT_m2sRatioFix_dot,ratio_max,len*sizeof(float));
ret = rk_aiq_user_api2_ae_setHdrExpAttr(ctx, HdrExpAttr);

```

## 设置亮度力度

```

ae_api_hdrExpAttr_t HdrExpAttr;
memset(&HdrExpAttr,0,sizeof(ae_api_hdrExpAttr_t));
ret = rk_aiq_user_api2_ae_getHdrExpAttr(ctx, &HdrExpAttr);
//set Evbias
HdrExpAttr.sw_aeT_evBias_strg = -100.0f;

ret = rk_aiq_user_api2_ae_setHdrExpAttr(ctx, HdrExpAttr);

```

## 设置中帧参数

```

ae_api_hdrExpAttr_t HdrExpAttr;
memset(&HdrExpAttr,0,sizeof(ae_api_hdrExpAttr_t));
ret = rk_aiq_user_api2_ae_getHdrExpAttr(ctx, &HdrExpAttr);
//set hdr mframe params
int len = 8;
float explevel[8] = {0, 0.096, 0.192, 0.384, 0.96, 1.344, 1.92, 3};
float setpoint[8] = {50, 45, 40, 35, 30, 25, 20, 15};
ret = rk_aiq_user_api2_ae_setHdrExpAttr(ctx, HdrExpAttr);

```

## **rk\_aiq\_user\_api2\_ae\_getHdrExpAttr**

### 【描述】

获取AE HDR模式曝光参数。

### 【语法】

```

XCamReturn
rk_aiq_user_api2_ae_getHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, ae_api_hdrExpAttr_t* pHdrExpAttr);

```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pHdrExpAttr	AE曝光参数结构体指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_ae\_setIrisAttr**

### 【描述】

设置AE光圈控制参数。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_ae_setIrisAttr(const rk_aiq_sys_ctx_t *sys_ctx, const ae_api_irisAttr_t irisAttr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
irisAttr	光圈控制参数结构体	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

### 【举例】

```
ae_api_irisAttr_t irisAttr;
ret = rk_aiq_user_api2_ae_getIrisAttr(ctx, &irisAttr);
irisAttr.sw_aeT_iris_en = true; /*run Alris*/
//set P-iris attributes
irisAttr.sw_aeT_iris_type = ae_iris_p_type;
irisAttr.pIrisCtrl.sw_aeT_totalStep_val = 81;
irisAttr.pIrisCtrl.sw_aeT_effcStep_val = 44;
irisAttr.pIrisCtrl.sw_aeT_zerolsMax_en = true;
uint16_t StepTable[1024] = {512, 511, 506, 499, 491, 483, 474, 465, 456,
                           446, 437, 427, 417, 408, 398, 388, 378, 368,
                           359, 349, 339, 329, 319, 309, 300, 290, 280,
```

```

271, 261, 252, 242, 233, 224, 214, 205, 196,
187, 178, 170, 161, 153, 144, 136, 128, 120,
112, 105, 98, 90, 83, 77, 70, 64, 58,
52, 46, 41, 36, 31, 27, 23, 19, 16,
13, 10, 8, 6, 4, 3, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0};

memcpy(irisAttr.pIrisCtrl.sw_aeT_step2Gain_table,StepTable,sizeof(irisAttr.pIrisCtrl.sw_aeT_step2Gain_
table));
ret = rk_aiq_user_api2_ae_setIrisAttr(ctx, irisAttr);

//set DC-iris attributes
irisAttr.sw_aeT_iris_type = ae_iris_dc_type;
irisAttr.dclIrisCtrl.sw_aeT_dcliris_Kp = 0.5f;
irisAttr.dclIrisCtrl.sw_aeT_dcliris_Ki = 0.2f;
irisAttr.dclIrisCtrl.sw_aeT_dcliris_Kd = 0.3f;
irisAttr.dclIrisCtrl.sw_aeT_pwmDuty_open = 40;
irisAttr.dclIrisCtrl.sw_aeT_pwmDuty_close = 22;
irisAttr.dclIrisCtrl.sw_aeT_pwmDuty_min = 0;
irisAttr.dclIrisCtrl.sw_aeT_pwmDuty_max = 100;
ret = rk_aiq_user_api2_ae_setIrisAttr(ctx, irisAttr);

//set HDC-iris attributes
irisAttr.hdclIrisCtrl.sw_aeT_damp_over = 0.35f;
irisAttr.hdclIrisCtrl.sw_aeT_damp_under = 0.2f;
irisAttr.hdclIrisCtrl.sw_aeT_zerolsMax_en = true;
irisAttr.hdclIrisCtrl.sw_aeT_target_min = 0;
irisAttr.hdclIrisCtrl.sw_aeT_target_max = 500;
uint16_t ZoomTargetDot[1024] = {0, 100, 200, 300};
int16_t ZoomtDot[1024] = {1000, 800, 0, 499};
uint16_t IrisTargetDot[1024] = {0, 100, 200, 300, 500};
uint16_t GainDot[1024] = {512, 511, 506, 499, 495};
irisAttr.hdclIrisCtrl.sw_aeC_zoom2Iris_len = 4;
irisAttr.hdclIrisCtrl.sw_aeC_iris2Gain_len = 5;
memcpy(irisAttr.hdclIrisCtrl.sw_aeC_zoom2Iris_val, ZoomTargetDot,
sizeof(irisAttr.hdclIrisCtrl.sw_aeC_zoom2Iris_val));
memcpy(irisAttr.hdclIrisCtrl.sw_aeC_zoom2Iris_idx, ZoomtDot,
sizeof(irisAttr.hdclIrisCtrl.sw_aeC_zoom2Iris_idx));
memcpy(irisAttr.hdclIrisCtrl.sw_aeC_iris2Gain_val, IrisTargetDot,
sizeof(irisAttr.hdclIrisCtrl.sw_aeC_iris2Gain_val));
memcpy(irisAttr.hdclIrisCtrl.sw_aeC_iris2Gain_idx, GainDot,
sizeof(irisAttr.hdclIrisCtrl.sw_aeC_iris2Gain_idx));
ret = rk_aiq_user_api2_ae_setIrisAttr(ctx, irisAttr);

//set manual iris with auto ae
irisAttr.manIris.sw_aeT_manIris_en = true;
if(irisAttr.sw_aeT_iris_type == ae_iris_p_type);
    irisAttr.manIris.sw_aeT_manPIrisGain_val = 512; /*p-iris F#=1.4*/
if(irisAttr.sw_aeT_iris_type == ae_iris_dc_type);
    irisAttr.manIris.sw_aeT_manDCIrisHold_val = 20; /*dc-iris PwmDuty=20*/
if(irisAttr.sw_aeT_iris_type == ae_iris_hdc_type);
    irisAttr.manIris.sw_aeT_manHDCIrisGain_val = 100; /*hdc-iris target=100*/
ret = rk_aiq_user_api2_ae_setIrisAttr(ctx, irisAttr);

```

## rk\_aiq\_user\_api2\_ae\_getIrisAttr

### 【描述】

获取AE 光圈控制参数。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_ae_getIrisAttr(const rk_aiq_sys_ctx_t *sys_ctx, const ae_api_irisAttr_t* irisAttr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
irisAttr	光圈控制参数结构体	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_ae\_setExpWinAttr**

### 【描述】

设置AE统计窗口属性。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_ae_setExpWinAttr(const rk_aiq_sys_ctx_t *sys_ctx, const Uapi_ExpWin_t ExpWin);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
ExpWin	窗口属性参数	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h

- 库文件：librkaiq.so

### 【举例】

#### 设置ROI曝光功能

根据ROI位置修改AE统计坐标，实现针对ROI区域进行亮度统计。

【注】设置统计坐标时，需保证ExpWin.h\_offs+ExpWin.h\_size<=pic\_width,  
ExpWin.v\_offs+ExpWin.v\_size<=pic\_height。

```
Uapi_ExpWin_t ExpWin;
//假设sensor分辨率为2688X1520 ROI相对于画面左上角的横向偏移为100个pixel，纵向偏移为100个pixel,
//ROI尺寸为边长100个pixel的正方形
ExpWin.Params.h_offs=100;
ExpWin.Params.v_offs=100;
ExpWin.Params.h_size=240;
ExpWin.Params.v_size=120;
rk_aiq_user_api2_ae_setExpWinAttr(ctx, ExpWin);
```

### rk\_aiq\_user\_api2\_ae\_getExpWinAttr

#### 【描述】

获取AE统计窗口属性。

#### 【语法】

```
XCamReturn
rk_aiq_user_api2_ae_getExpWinAttr(const rk_aiq_sys_ctx_t* sys_ctx,const Uapi_ExpWin_t* ExpWin);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
ExpWin	窗口属性参数	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

### rk\_aiq\_user\_api2\_ae\_queryExpResInfo

#### 【描述】

获取AE内部状态信息。

#### 【语法】

XCamReturn

```
rk_aiq_user_api2_ae_queryExpResInfo(const rk_aiq_sys_ctx_t* ctx, ae_api_queryInfo_t* pExpResInfo);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pExpResInfo	AE内部状态信息结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ae.h、rk\_aiq\_api\_types\_ae.h
- 库文件：librkaiq.so

## 模块级API数据类型

### ae\_api\_expSwAttr\_t

#### 【说明】

AE公共功能控制参数结构体。

#### 【定义】

```
typedef struct ae_api_expSwAttr_s {
    ae_commCtrl_t commCtrl;
    ae_advanced_t advanced;
} ae_api_expSwAttr_t;
```

#### 【相关数据类型】

- ae\_commCtrl\_t
- ae\_advanced\_t

### ae\_commCtrl\_t

#### 【说明】

AE公共功能控制参数结构体。

#### 【定义】

```
typedef struct ae_commCtrl_s {
    bool sw_aeT_algo_en;
    uint8_t sw_aeT_algo_interval;
    rk_aiq_op_mode_t sw_aeT_opt_mode;
    ae_histStats_mode_t sw_aeT_histStats_mode;
```

```

ae_rawStats_mode_t sw_aeT_rawStats_mode;
ae_yRange_mode_t sw_aeT_yRange_mode;
uint8_t sw_aeT_grid_wgt[AE_GRIDWEIGHT_MAX_NUM];
ae_meCtrl_t meCtrl;
ae_speed_t speed;
ae_delay_t delay;
ae_frmRate_t frmRate;
ae_antiFlicker_t antiFlicker;
ae_envLvCalib_t envLvCalib;
ae_winScale_t winScale;
} ae_commCtrl_t;

```

### 【成员】

成员名称	描述
sw_aeT_algo_en	Aec使能开关。1，开启Aec算法；0，关闭Aec算法，曝光保持在关闭前的值。
sw_aeT_rawStats_mode	Aec模块亮度统计模式。共4种模式分别为：Y/R/G/B，默认为Y模式。
sw_aeT_histStats_mode	Aec模块直方图统计模式。共4种模式分别为：Y/R/G/B，默认为Y模式。
sw_aeT_yRange_mode	Aec模块Y通道Range模式。共两种模式分别为FULL/LIMITED，默认为FULL模式。
sw_aeT_algo_interval	Ae算法运行间隔，取值范围[0,255]，默认值为0。取值为0时，每帧运行AE；取值为1时，每隔1帧运行AE；以此类推。建议该值不要超过3，以免影响曝光过渡的平滑
sw_aeT_opt_mode	曝光模式，分为自动曝光(RK_AIQ_OP_MODE_AUTO)模式/手动(RK_AIQ_OP_MODE_MANUAL)曝光模式。默认为AUTO模式。手动曝光模式需要与meCtrl一起配合，进行手动曝光值的设置。
sw_aeT_grid_wgt	窗口（直方图）权重，包含15*15个数组元素，取值范围[0,32]
speed	AE调节速度参数结构体
delay	AE响应延迟参数结构体
frmRate	AE帧率参数结构体
antiFlicker	AE抗工频闪烁参数结构体
envLvCalib	AE环境亮度标定参数结构体
winScale	AE测光窗口大小参数结构体
meCtrl	手动曝光参数结构体

### 【相关数据类型】

- ae\_speed\_t
- ae\_delay\_t
- ae\_frmRate\_t
- ae\_antiFlicker\_t

- ae\_envLvCalib\_t
- ae\_winScale\_t
- ae\_meCtrl\_t

### **ae\_meCtrl\_t**

#### **【说明】**

定义手动曝光参数结构体，根据曝光模式分为linMe和hdrMe两套参数。

#### **【定义】**

```

typedef struct ae_linMe_s {
    bool sw_aeT_manTime_en;
    bool sw_aeT_manGain_en;
    bool sw_aeT_manIspDGain_en;
    float sw_aeT_manTime_val;
    float sw_aeT_manGain_val;
    float sw_aeT_manIspDGain_val;
} ae_linMe_t;

typedef struct ae_hdrMe_s {
    bool sw_aeT_manTime_en;
    bool sw_aeT_manGain_en;
    bool sw_aeT_manIspDGain_en;
    float sw_aeT_manTime_val[AE_HDRFRAME_MAX_NUM];
    float sw_aeT_manGain_val[AE_HDRFRAME_MAX_NUM];
    float sw_aeT_manIspDGain_val[AE_HDRFRAME_MAX_NUM];
} ae_hdrMe_t;

typedef struct ae_meCtrl_s {
    ae_linMe_t linMe;
    ae_hdrMe_t hdrMe;
} ae_meCtrl_t;

```

#### **【成员】**

成员名称	描述
sw_aeT_manTime_en	手动曝光时间使能， 默认值为1。
sw_aeT_manGain_en	手动sensor增益使能， 默认值为1。
sw_aeT_manIspDGain_en	手动ISP数字增益使能， 默认值为1。
sw_aeT_manTime_val	手动曝光时间值， 以秒为单位， 参数值受sensor限制。
sw_aeT_manGain_val	手动sensor增益值， 以倍数为单位， 参数值受sensor限制。
sw_aeT_manIspDGain_val	手动ISP数字增益值， 以倍数为单位， 参数值受ISP限制。

#### **【注意事项】**

- 该模块参数仅在AeOptype = MANUAL时有效。sw\_aeT\_manTime\_en、sw\_aeT\_manGain\_en、sw\_aeT\_manIspDGain\_en皆为1时，为手动模式；以上三者中只要任意一项不使能，则为半自动模式；以上三者皆为0，则等同自动模式，系统会报错提醒。
- 手动/半手动模式下，手动曝光时间和增益会受系统最大/最小曝光时间和增益限制。超出系统曝光限制的范围之后，将使用系统曝光限制的最大/最小值替代。

- ae\_hdrMe\_t中，sw\_aeT\_manTime\_val、sw\_aeT\_manGain\_val、sw\_aeT\_manIspDGain\_val皆为成员个数为3的数组。Hdr 2帧模式下，数组[0-1]成员有效，分别表示短、长帧；Hdr 3帧模式下，数组[0-2]成员皆有效，分别对应短、中、长帧。

## ae\_speed\_t

### 【说明】

定义AE调节速度属性。

### 【定义】

```
typedef struct ae_dynDamp_s {
    bool     sw_aeT_dynDamp_en;
    ae_opt_mode_t sw_aeT_slowOpt_mode;
    float    sw_aeT_slowRange_val;
    float    sw_aeT_slowDamp_val;
} ae_dynDamp_t;

typedef struct ae_speed_s {
    bool     sw_aeT_smooth_en;
    float    sw_aeT_damp_under;
    float    sw_aeT_damp_over;
    float    sw_aeT_damp_dark2Bright;
    float    sw_aeT_damp_bright2Dark;
    ae_dynDamp_t  dynDamp;
} ae_speed_t;
```

### 【成员】

成员名称	描述
sw_aeT_smooth_en	平滑开关。开启：实现曝光平滑；关闭：关闭曝光平滑，可提升曝光调节速度。
sw_aeT_damp_under	环境亮度未剧烈变化，当前图像亮度低于目标值时需要抬高曝光，对应的曝光调节速度，取值范围[0, 1]。值越小，曝光调节速度越快。
sw_aeT_damp_over	环境亮度未剧烈变化，当前图像亮度高于目标值时需要降低曝光，对应的曝光调节速度，取值范围[0, 1]。值越小，曝光调节速度越快。
sw_aeT_damp_dark2Bright	环境亮度存在从暗到亮的突变时，对应的曝光调节速度，取值范围[0, 1]。值越小，曝光调节速度越快。
sw_aeT_damp_bright2Dark	环境亮度存在从亮到暗的突变时，对应的曝光调节速度，取值范围[0, 1]。值越小，曝光调节速度越快。
dynDamp	动态曝光调节速度模块。

- ae\_dynDamp\_t 结构体成员如下：

成员名称	描述
sw_aeT_dynDamp_en	动态调节速度开关。开启：曝光调节速度随场景亮度动态调节；关闭：曝光调节速度固定采用 sw_aeT_damp_over/under/dark2Bright/bright2Dark值。
sw_aeT_slowOpt_mode	减速收敛模式。包括手动模式 (ae_opt_manual_mode) 与自动模式 (ae_opt_auto_mode) 。
sw_aeT_slowRange_val	减速收敛对应的亮度范围，仅在 SlowOPType=ae_opt_manual_mode下有效，取值范围[0, 100]，单位百分比。代表当前亮度在目标亮度±SlowRange时，采用 SlowDamp作为 曝光调节速度。
sw_aeT_slowDamp_val	减速收敛对应的曝光调节速度，仅在 SlowOPType=ae_opt_manual_mode下有效，取值范围[0, 1]。

## ae\_delay\_t

### 【说明】

定义AE响应延时属性。

### 【定义】

```
typedef struct ae_delay_s {
    ae_delay_mode_t sw_aeT_delay_mode;
    uint8_t     sw_aeT_blackDelay_val;
    uint8_t     sw_aeT_whiteDelay_val;
} ae_delay_t;
```

### 【成员】

成员名称	描述
sw_aeT_delay_mode	延时模式，包括帧延迟模式ae_delay_frame_mode（延时单位为帧）和时间延迟模式（延时单位固定为1/30秒）。
sw_aeT_blackDelay_val	自动曝光触发延时属性，其单位随sw_aeT_delay_mode变化。图像亮度低于目标值超过sw_aeT_blackDelay_val时，Ae开始调节。
sw_aeT_whiteDelay_val	自动曝光触发延时属性，其单位随sw_aeT_delay_mode变化。图像亮度高于目标值超过sw_aeT_whiteDelay_val时，Ae开始调节。

## ae\_frmRate\_t

### 【说明】

定义AE 帧率属性。

### 【定义】

```
typedef struct ae_frmRate_s {
    ae_frmRate_mode_t sw_aeT_frmRate_mode;
    float      sw_aeT_frmRate_val;
} ae_frmRate_t;
```

### 【成员】

成员名称	描述
sw_aeT_frmRate_mode	自动曝光固定帧率模式选者，ae_frmRate_auto_mode时，采用自动降帧模式；ae_frmRate_fix_mode时，表示固定帧率模式。
sw_aeT_frmRate_val	mode=auto时，用于表示自动降帧模式的最高帧率：默认值为0时，使用驱动setting中的帧率作为最高帧率；默认值不为0时，使用设定的帧率作为最高帧率。 mode=manual时，用于表示固定帧模式的固定帧率值：默认值为0时，使用驱动setting中的帧率作为固定帧率；默认值不为0时，使用设定的帧率作为固定帧率。

## ae\_antiFlicker\_t

### 【说明】

定义抗闪属性。

### 【定义】

```
typedef struct ae_antiFlicker_s {
    bool      sw_aeT_antiFlicker_en;
    ae_antiFlicker_freq_t sw_aeT_antiFlicker_freq;
    ae_antiFlicker_mode_t sw_aeT_antiFlicker_mode;
} ae_antiFlicker_t;
```

### 【成员】

成员名称	描述
sw_aeT_antiFlicker_en	工频抗闪功能使能状态，0：关闭抗闪功能；1：开启抗闪功能。
sw_aeT_antiFlicker_freq	抗闪频率属性，共包含3种帧率，分别为ae_antiFlicker_50hz_freq、ae_antiFlicker_60hz_freq、ae_antiFlicker_auto_freq。
sw_aeT_antiFlicker_mode	抗闪模式，共包含两种抗闪模式： ae_antiFlicker_normal_mode（普通抗闪模式）、 ae_antiFlicker_auto_mode（自动抗闪模式）。

### 【注意事项】

- 关闭抗闪功能，需将抗闪使能位sw\_aeT\_antiFlicker\_en置0，算法内部自动配置sw\_aeT\_antiFlicker\_freq=ae\_antiFlicker\_off\_freq；如抗闪使能位enable置1时，抗闪频率配置为ae\_antiFlicker\_off\_freq，该频率值将配置无效，使用默认值ae\_antiFlicker\_50hz\_freq。
- Frequency选择ae\_antiFlicker\_auto\_freq，代表由内部算法自动判断当前交流电工频，目前仅支持判断50Hz与60Hz工频交流电。
- ae\_antiFlicker\_normal\_mode 为普通抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间固定为 1/120 sec (60Hz) 或 1/100 sec (50Hz)，不受曝光时间最小值的限制。  
有灯光的环境：曝光时间可与光源频率相匹配，能够防止图像闪烁。  
高亮度的环境：亮度越高，要求曝光时间就最短。而普通抗闪模式的最小曝光时间不能匹配光源频率，产生过曝。
- ae\_antiFlicker\_auto\_mode 为自动抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间可达到 sensor 的最小曝光时间。与普通抗闪模式的差别主要在高亮度的环境体现。

高亮度的环境：最小曝光时间可以达到 sensor 的最小曝光时间，能够有效抑制过曝，但此时抗闪失效。

### ae\_envLvCalib\_t

#### 【说明】

定义环境亮度标定参数。

#### 【定义】

```
typedef struct ae_envLvCalib_s {
    bool sw_aeT_envCalib_en;
    float sw_aeC_envCalib_Fn;
    float sw_aeC_envCalib_coeff[2];
} ae_envLvCalib_t;
```

#### 【成员】

成员名称	描述
sw_aeT_envCalib_en	环境亮度计算模块开关，0：关闭；1：开启。
sw_aeC_envCalib_Fn	环境亮度标定时的基准相对光圈大小，该值与镜头有关。
sw_aeC_envCalib_coeff	环境亮度标定曲线系数。

#### 【注意事项】

- sw\_aeT\_envCalib\_en、sw\_aeC\_envCalib\_Fn暂时无效，默认为0
- 有快启需求，使用MCU fastae时，sw\_aeC\_envCalib\_coeff有效，为环境亮度标定值；非快启应用，该参数无效

### ae\_winScale\_t

#### 【说明】

定义AE模块硬件统计窗口大小比例参数，包括有：基于raw图的AE硬件统计窗口、基于tmo模块后raw图的AE硬件统计窗口、基于yuv图的AE硬件统计窗口。窗口大小比例配置参数是以sensor分辨率为基准，设置对应的比例值。

#### 【定义】

```
typedef struct ae_winRatio_s {
    float hw_aeCfg_win_x;
    float hw_aeCfg_win_y;
    float hw_aeCfg_win_width;
    float hw_aeCfg_win_height;
} ae_winRatio_t;

typedef struct ae_winScale_s {
    ae_winRatio_t inRawWinScale;
    ae_winRatio_t tmoRawWinScale;
    ae_winRatio_t yuvWinScale;
} ae_winScale_t;
```

#### 【成员】

成员名称	描述
hw_aeCfg_win_x	硬件统计窗口左上角相对sensor感光区域的水平方向偏移值，参数的range=[0, 1]。
hw_aeCfg_win_y	硬件统计窗口水平方向的尺寸，参数的range=[0, 1]。
hw_aeCfg_win_width	硬件统计窗口左上角相对sensor感光区域的竖直方向偏移值，参数的range=[0, 1]。
hw_aeCfg_win_height	硬件统计窗口竖直方向的尺寸，参数的range=[0, 1]。

#### 【注意】

- 3576平台，仅inRawWinScale与tmoRawWinScale有效

### ae\_advanced\_t

#### 【说明】

定义优先使用参数结构体。

#### 【定义】

```
typedef struct ae_advanced_s {
    bool      sw_aeT_advGridWgt_en;
    uint8_t   sw_aeT_advGrid_wgt[AE_GRIDWEIGHT_MAX_NUM];
    bool      sw_aeT_advAeRange_en;
    ae_expRange_t linExpRange;
    ae_expRange_t hdrExpRange[AE_HDRFRAME_MAX_NUM];
} ae_advanced_t;
```

#### 【成员】

成员名称	描述
sw_aeT_advGridWgt_en	置1，优先使用该结构体中的权重参数；置0，使用结构体上一级中的权重参数。
sw_aeT_advGrid_wgt	AE权重，大小为15X15，取值范围[0, 32]。
sw_aeT_advAeRange_en	置1，设置自动曝光分量range；置0，不设置自动曝光分量range，以IQ文件为准。
linExpRange	线性曝光的自动曝光分量range参数结构体（并非最终生效range）。
hdrExpRange	HDR曝光的自动曝光分量range参数结构体（并非最终生效range）。

#### 【注意事项】

- ae\_commCtrl\_t 结构体中定义了一组AE权重，权重个数为15X15，取值范围[0, 32]。ae\_advanced\_t 中也定义的一组AE权重，权重个数为15X15，取值范围[0, 32]，需要打开使能sw\_aeT\_advGridWgt\_en，将其置1。如有人脸应用，需要分别设置有人脸的权重和无人脸的权重时，可以使用ae\_advanced\_t 中的权重作为人脸权重，ae\_commCtrl\_t 中的权重作为 无人脸是

的权重，通过ae\_advanced\_t 中的sw\_aeT\_advGridWgt\_en实现两个权重的切换，en=1时使用ae\_advanced\_t 中的权重，en=0时使用ae\_commCtrl\_t 中的权重。

- 通过api设置AE参数范围时，需要将sw\_aeT\_advAeRange\_en置1，否则默认使用调试文件中的自动曝光参数范围。曝光参数范围的设置，按照曝光模式的不同，分为linExpRange和hdrExpRange两套。其中hdrExpRange内支持各帧自动曝光参数范围的设置。另，最终生效的曝光时间及增益range可通过ae\_api\_queryInfo\_t查询，此处仅为设置的range，而非最终生效range。
- hdrExpRange 预定义为包含3个成员的数组。2帧HDR模式下，仅成员0、1有效、分别表示短帧和长帧对应的曝光分量范围；3帧HDR模式下，0、1、2皆有效，分别表示短、中、长帧对应的曝光分量范围。
- 各曝光分量最大值/最小值为默认值0时，设置的曝光分量范围不生效，此时各曝光分量实际最大值/最小值以算法校正过的曝光分解路线节点最小值和最大值决定。

### ae\_expRange\_t

#### 【说明】

定义AE参数范围。

#### 【定义】

```
typedef struct ae_expRange_s {  
    float sw_aeT_time_min;  
    float sw_aeT_time_max;  
    float sw_aeT_gain_min;  
    float sw_aeT_gain_max;  
    float sw_aeT_ispDGain_min;  
    float sw_aeT_ispDain_max;  
    int sw_aeT_pIrisGain_min;  
    int sw_aeT_pIrisGain_max;  
} ae_expRange_t;
```

#### 【成员】

成员名称	描述
sw_aeT_time_min	曝光时间设置最小值，以秒为单位
sw_aeT_time_max	曝光时间设置最大值，以秒为单位
sw_aeT_gain_min	sensor模拟增益设置最小值，以倍数为单位
sw_aeT_gain_max	sensor模拟增益设置最大值，以倍数为单位
sw_aeT_ispDGain_min	ISP数字增益设置最小值，以倍数为单位
sw_aeT_ispDain_max	ISP数字增益设置最大值，以倍数为单位
sw_aeT_pIrisGain_min	光圈等效增益设置最小值，仅支持P-Iris光圈大小控制，以倍数为单位
sw_aeT_pIrisGain_max	光圈等效增益设置最大值，仅支持P-Iris光圈大小控制，以倍数为单位

#### 【注意事项】

- 各曝光分量最大值/最小值为默认值0时，设置的曝光分量范围不生效，各曝光分量实际最大值/最小值以Aec Route曝光分解路线节点最小值和最大值决定。
- 各曝光分量最大值/最小值不为0时，设置的曝光分量范围生效，当设置的曝光分量范围不超过sensor或ISP的限制，则各曝光分量实际最大值/最小值以设置的曝光分量范围为准，并对Aec

Route曝光分解路线做校正，节点最大值/最小值改为设置的曝光分量最大值/最小值；若超过sensor或ISP的限制，则各曝光分量实际最大值/最小值以Aec Route曝光分解路线节点最大值和最小值为准。

## ae\_api\_linExpAttr\_t

### 【说明】

定义AE线性曝光调试参数。

### 【定义】

```
typedef struct ae_linAeCtrl_s {
    float      sw_aeT_tolerance_in;
    float      sw_aeT_tolerance_out;
    float      sw_aeT_evBias_strg;
    ae_strategy_mode_t sw_aeT_strategy_mode;
    ae_linInitExp_t  initExp;
    ae_linRoute_t   route;
    ae_dynSetpoint_t dynSetpoint;
    ae_backLitCtrl_t backLightCtrl;
    ae_overExpCtrl_t overExpCtrl;
} ae_linAeCtrl_t;

typedef ae_linAeCtrl_t ae_api_linExpAttr_t;
```

### 【成员】

成员名称	描述
sw_aeT_tolerance_in/out	自动曝光调节时，画面亮度的容忍度。单位为%，取值范围为[0, 100]。
sw_aeT_evBias_strg	自动曝光调节时，曝光量的偏差百分比，单位为%，参考取值范围为[-200, +200]。
sw_aeT_strategy_mode	自动曝光策略模式，高光优先或低光优先（暂时无效）。
initExp	线性曝光模式初始值参数。
route	自动曝光分解策略属性。
dynSetpoint	自动曝光调节的动态目标亮度值属性，随曝光量动态变化，取值范围为[0, 255]。
backLightCtrl	背光补偿功能参数。
OverExpCtrl	强光抑制功能参数。

### 【注意事项】

- 曝光量偏差sw\_aeT\_evBias\_strg，用于特殊场景下对（固定/动态）目标亮度值（setPoint）进行调整。真实生效目标亮度为  $\text{setPoint} * [1 + \text{abs}(\text{evBias}) / 100] ^ [\text{evBias} / \text{abs}(\text{evBias})]$ 。如设置evBias=100时，目标亮度为默认参数的2倍；evBias=-100时，目标亮度为默认参数的1/2。
- 自动曝光画面亮度的容忍度为tolerance，当自动曝光收敛时，画面亮度值应在 [真实生效目标亮度 \*  $(1 - \text{tolerance} / 100)$ ]，真实生效目标亮度 \*  $(1 + \text{tolerance} / 100)$ ] 范围内。tolerance in/out设置较大，

一方面会影响AE的响应速度，一方面会影响evBias值。当evBias调整的间隔值低于tolerance in/out，有可能导致亮度调整不生效。

- sw\_aeT\_strategy\_mode暂无效。

## ae\_linInitExp\_t

### 【说明】

定义线性曝光初始值参数结构体。

### 【定义】

```
typedef struct ae_linInitExp_s {
    float sw_aeT_initTime_val;
    float sw_aeT_initGain_val;
    float sw_aeT_initIspDGain_val;
} ae_linInitExp_t;
```

### 【成员】

成员名称	描述
sw_aeT_initTime_val	初始曝光时间值，单位为秒。
sw_aeT_initGain_val	初始sensor增益值，此处增益值为实际值，单位为1x。
sw_aeT_initIspDGain_val	初始ISP数字增益值，此处增益值为实际值，单位为1x。

## ae\_linRoute\_t

### 【说明】

定义AE线性曝光分解路径属性。

### 【定义】

```
typedef struct ae_linRoute_s {
    uint8_t sw_aeT_route_len;
    float sw_aeT_time_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_gain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_ispDGain_dot[AE_ROUTE_DOT_MAX_NUM];
    int sw_aeT_pIrisGain_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_linRoute_t;
```

### 【成员】

成员名称	描述
sw_aeT_route_len	route节点的长度，每个曝光分量的节点个数需要一致，最多支持12个分解节点。
sw_aeT_time_dot	曝光时间节点，单位为秒。
sw_aeT_gain_dot	sensor增益节点，此处增益值为实际值，单位为1x。
sw_aeT_ispDGain_dot	lsp数字增益节点，此处增益值为实际值，单位为1x。
sw_aeT_pIrisGain_dot	光圈等效增益节点，此处增益值为实际值，单位为1x。

## 【注意事项】

- 曝光分解曲线节点个数最多支持12个，建议不要少于6个。
- 节点的曝光量是曝光时间、sensor增益、ISP数字增益、光圈等效增益等各分量的乘积。节点曝光量必须单调递增，即后一个节点的曝光量必须大于前一个节点的曝光量。第一个节点的曝光量最小，第二个节点的曝光量最大。
- 节点中曝光时间分量的单位为秒，最小值允许为0，实际最小曝光时间代码内部会根据sensor限制进行校正。
- 光圈分量仅支持P-Iris, 不支持DC-Iris和HDC-iris。P-iris等效增益分量仅在Airis自动光圈功能使能时有效，否则默认光圈固定为初始值大小。P-iris等效增益的计算详见ae\_api\_irisAttr\_t结构体。
- 设置的曝光分解路线节点不是最终生效的曝光分解路线。最终生效的曝光分解路线各分量的值会受到帧率、帧率模式、api配置的曝光分量最大/小值、sensor的共同限制。先对曝光分解路线节点最大/小值做第一次校正，当节点最大/小值不超过sensor或isp的限制时，节点最大/小值不变；当节点最大/小值超过sensor或isp的限制时，节点最大/小值以sensor或isp的限制为准。当api配置的曝光分量最大/小值为0时，最终生效的曝光分解路线以第一次校正的分解路线为准；当api配置的曝光分量最大/小值不为0时，且设置的最大/小值不超过sensor或isp的限制时，对曝光分解路线做第二次校正，节点最大/小值以手动设置的范围为准；若设置曝光分量的最大/小值超过sensor或isp的限制时，曝光分解路线曝光分量的节点最大/小值以第一次校正结果为准。
- 如果相邻节点的曝光量增加，则应该只有一个曝光分量增加，其他曝光分量固定。增加的分量决定该段路线的分配策略。例如增益分量增加，其他分量固定，那么该段路线的分配策略是增益优先。

## ae\_dynSetpoint\_t

### 【说明】

定义自动曝光调节的动态目标亮度值结构体。

### 【定义】

```
typedef struct ae_dynSetpoint_s {
    uint8_t sw_aeT_dynSetpoint_len;
    float sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_dynSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_dynSetpoint_t;
```

### 【成员】

成员名称	描述
sw_aeT_dynSetpoint_len	dynSetpoint节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	动态曝光量节点属性，节点值为当前曝光量值，建议至少设置6个节点，以防曝光过渡不平滑。
sw_aeT_dynSetpoint_dot	动态目标亮度值节点属性，取值范围[0, 255]，节点值随曝光量动态变化，曝光量节点值越大，目标亮度节点值越小，并与曝光量节点一一对应。需要与sw_aeT_expLevel_dot节点个数一致，建议至少设置6个节点，以防曝光过渡不平滑。

## ae\_backLitCtrl\_t

### 【说明】

定义背光补偿功能参数结构体。

### 【定义】

```

typedef struct ae_backLitSetpoint_s {
    uint8_t sw_aeT_backLitSetpoint_len;
    float sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_nonOEPdfTh_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_loLitPdfTh_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_loLitSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_backLitSetpoint_t;

typedef struct ae_backLitCtrl_s {
    bool sw_aeT_backLit_en;
    float sw_aeT_backLitBias_strg;
    ae_measArea_mode_t sw_aeT_measArea_mode;
    float sw_aeT_oeROILow_thred;
    float sw_aeT_lumaDist_thred;
    float sw_aeT_loLv_thred;
    float sw_aeT_hiLv_thred;
    ae_backLitSetpoint_t backLitSetpoint;
} ae_backLitCtrl_t;

```

## 【成员】

成员名称	描述
sw_aeT_backLit_en	模块使能位，1：使能，0：关闭。
sw_aeT_backLitBias_strg	背光补偿时，调整亮度力度，单位为%，取值范围为[-200, +200]。
sw_aeT_measArea_mode	暗区检测区域，共包含6种模式： ae_measArea_auto_mode (up/bottom/left/right/center)
sw_aeT_lumaDist_thred	区域增长容忍度。
sw_aeT_oeROILow_thred	过曝区域亮度最低值，用于区分过曝区域与非过曝区域。
sw_aeT_hiLv_thred	环境亮度高阈值。
sw_aeT_loLv_thred	环境亮度低阈值。
backLitSetpoint	背光补偿目标值。

- ae\_backLitSetpoint\_t 结构体成员如下：

成员名称	描述
sw_aeT_backLitSetpoint_len	backLitSetpoint节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	动态曝光量节点属性，节点值为当前曝光量值，建议至少设置6个节点，以防曝光过渡不平滑。节点值即为曝光值 (gain * time, time以秒为单位)。
sw_aeT_nonOEPdfTh_dot	非过曝区域占比阈值，取值范围[0, 1]，建议至少设置6个节点，以防曝光过渡不平滑。节点个数需要与expLevel一致，节点值需要与expLevel一一对应。
sw_aeT_loLitPdfTh_dot	暗区占比阈值，取值范围[0, 1]，建议至少设置6个节点，以防曝光过渡不平滑。节点个数需要与expLevel一致，节点值需要与expLevel一一对应。
sw_aeT_loLitSetpoint_dot	动态暗区亮度目标值，取值范围[0, 255]，建议至少设置6个节点，以防曝光过渡不平滑。节点个数需要与expLevel一致，节点值需要与expLevel一一对应，随着expLevel增大而减小。

## ae\_overExpCtrl\_t

### 【说明】

定义强光抑制功能参数结构体。

### 【定义】

```
typedef struct ae_overExpSetpoint_s {
    uint8_t sw_aeT_overExpSetpoint_len;
    float sw_aeT_oePdf_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_loLitWgt_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_hiLitWgt_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_overExpSetpoint_t;

typedef struct ae_overExpCtrl_s {
    bool      sw_aeT_overExp_en;
    float    sw_aeT_overExpBias_strg;
    float    sw_aeT_overExpWgt_max;
    float    sw_aeT_loLit_thred;
    float    sw_aeT_hiLit_thred;
    ae_overExpSetpoint_t overExpSetpoint;
} ae_overExpCtrl_t;
```

### 【成员】

成员名称	描述
sw_aeT_overExp_en	模块使能位，1：使能，0：关闭。
sw_aeT_overExpBias_strg	强光抑制时，调整亮度力度，单位为%，取值范围为[-200, +200]。
sw_aeT_overExpWgt_max	最大权重值，取值范围[1, 8]，用于约束过曝抑制对亮度的影响力。值越大，约束力度越小。
sw_aeT_loLit_thred	低亮区域的亮度阈值，取值范围[0, 255]。
sw_aeT_hiLit_thred	高亮区域的亮度阈值，取值范围[0, 255]。
overExpSetpoint	强光抑制目标值。

- ae\_overExpSetpoint\_t 结构体成员如下：

成员名称	描述
sw_aeT_overExpSetpoint_len	overExpSetpoint节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_oePdf_dot	过曝区域占比，建议至少设置6个节点，以防曝光过渡不平滑，占比值从小到大变化，取值范围[0, 1]。
sw_aeT_loLitWgt_dot	低亮区域权重，取值范围[1, 20]，建议至少设置6个节点，以防曝光过渡不平滑，与过曝区域占比节点相对应。建议值为1
sw_aeT_hiLitWgt_dot	高亮区域权重，取值范围[1, 20]，建议至少设置6个节点，以防曝光过渡不平滑，与过曝区域占比节点相对应。 sw_aeT_oePdf_dot越大，sw_aeT_hiLitWgt_dot越小。

## ae\_api\_hdrExpAttr\_t

### 【说明】

定义AE HDR曝光调试参数。

### 【定义】

```
typedef struct ae_hdrAeCtrl_s {
    float      sw_aeT_tolerance_in;
    float      sw_aeT_tolerance_out;
    float      sw_aeT_evBias_strg;
    ae_strategy_mode_t  sw_aeT_strategy_mode;
    float      sw_aeT_lumaDist_thred;
    ae_hdrInitExp_t   initExp;
    ae_hdrRoute_t     route;
    ae_hdrExpRatioCtrl_t  expRatioCtrl;
    ae_hdrLfrmMode_t   lfrmMode;
    ae_hdrLfrmCtrl_t   lfrmCtrl;
    ae_hdrMfrmCtrl_t   mfrmCtrl;
    ae_hdrSfrmCtrl_t   sfrmCtrl;
} ae_hdrAeCtrl_t;
```

```
typedef ae_hdrAeCtrl_t ae_api_hdrExpAttr_t;
```

## 【成员】

成员名称	描述
sw_aeT_tolerance_in/out	自动曝光调节时，画面亮度的容忍度。单位为%，取值范围为[0, 100]。
sw_aeT_evBias_strg	自动曝光调节时，曝光量的偏差百分比，单位为%，取值范围为[-200, +200]。
sw_aeT_strategy_mode	自动曝光策略模式，高光优先或低光优先。 (ae_strategy_highlight_mode/ae_strategy_lowlight_mode)
sw_aeT_lumaDist_thred	区域增长容忍度。
initExp	Hdr曝光模式初始值设置。
route	自动曝光分解策略属性。
expRatioCtrl	HdrAE曝光比控制模块，仅在Hdr模式多帧合成下有效。
lfrmMode	长帧模式参数。
lfrmCtrl	长帧控制参数。
mfrmCtrl	中帧控制参数，仅在HDR 3帧模式下有效。
sfrmCtrl	短帧控制参数。

## ae\_hdrInitExp\_t

### 【说明】

定义Hdr AE曝光初始值参数结构体。

### 【定义】

```
typedef struct ae_hdrInitExp_s {
    float sw_aeT_initTime_val[AE_HDRFRAME_MAX_NUM];
    float sw_aeT_initGain_val[AE_HDRFRAME_MAX_NUM];
    float sw_aeT_initIspDGain_val[AE_HDRFRAME_MAX_NUM];
} ae_hdrInitExp_t;
```

## 【成员】

成员名称	描述
sw_aeT_initTime_val	初始曝光时间值，单位为秒。
sw_aeT_initGain_val	初始sensor增益值，此处增益值为实际值，单位为1x。
sw_aeT_initIspDGain_val	初始ISP数字增益值，此处增益值为实际值，单位为1x。

## ae\_hdrRoute\_t

### 【说明】

定义Hdr AE曝光分解路径属性。

### 【定义】

```

typedef struct ae_hdrRoute_s {
    uint8_t sw_aeT_route_len;
    float sw_aeT_frm0Time_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm0Gain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm0IspDGain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm1Time_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm1Gain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm1IspDGain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm2Time_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm2Gain_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_frm2IspDGain_dot[AE_ROUTE_DOT_MAX_NUM];
    int sw_aeT_pIrisGain_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_hdrRoute_t;

```

### 【成员】

成员名称	描述
sw_aeT_route_len	route节点的长度，每个曝光分量的节点个数需要一致，最多支持12个分解节点。
sw_aeT_frm0Time_dot sw_aeT_frm1Time_dot sw_aeT_frm2Time_dot	曝光时间节点，单位为秒。Hdr 2帧模式时，仅frm0/1有效；Hdr 3帧模式时，frm0/1/2皆有效。frm0/1/2依次为曝光量从短至长的帧序号。
sw_aeT_frm0Gain_dot sw_aeT_frm1Gain_dot sw_aeT_frm2Gain_dot	sensor增益节点，此处增益值为实际值，单位为1x。Hdr 2帧模式时，仅frm0/1有效；Hdr 3帧模式时，frm0/1/2皆有效。frm0/1/2依次为曝光量从短至长的帧序号。
sw_aeT_frm0IspDGain_dot sw_aeT_frm1IspDGain_dot sw_aeT_frm2IspDGain_dot	Isp数字增益节点，此处增益值为实际值，单位为1x。Hdr 2帧模式时，仅frm0/1有效；Hdr 3帧模式时，frm0/1/2皆有效。frm0/1/2依次为曝光量从短至长的帧序号。
sw_aeT_pIrisGain_dot	光圈等效增益节点，此处增益值为实际值，单位为1x。

### 【注意事项】

- 曝光分解曲线节点最多12个，**建议至少设置6个节点**，才可实现曝光分解的平滑。
- 需要注意的是：HDR 2帧模式下，仅需设置frm0/1TimeDot、frm0/1GainDot、frm0/1IspDGainDot，分别对应实际的短、长帧；HDR 3帧模式下，需设置frm0/1/2TimeDot、frm0/1/2GainDot、frm0/1/2IspDGainDot，分别对应短、中、长帧。设置HDR模式下各帧的sensor曝光时间时，需要合理分配曝光时间，各帧**曝光时间的总和不能超过帧率所允许的最大曝光时间！**
- 节点的曝光量是曝光时间、sensor增益、ISP数字增益、光圈等效增益等各分量的乘积。节点曝光量必须单调递增，即后一个节点的曝光量必须大于前一个节点的曝光量。第一个节点的曝光量最小，第二个节点的曝光量最大。
- 节点中曝光时间分量的单位为秒，最小值允许为0，实际最小曝光时间代码内部会根据sensor限制进行校正。
- 光圈分量仅支持P-Iris，不支持DC-Iris和HDC-iris。P-iris等效增益分量仅在Airis自动光圈功能使能时有效，否则默认光圈固定为初始值大小。P-iris等效增益的计算详见ae\_api\_irisAttr\_t结构体。
- 设置的曝光分解路线节点不是最终生效的曝光分解路线。系统最终各曝光分量的实际最大/小值由曝光分解节点和手动配置的曝光分量最大/小值共同决定。先对曝光分解路线节点最大/小值做第一次校正，当节点最大/小值不超过sensor或isp的限制时，节点最大/小值不变；当节点最大/小值超过sensor或isp的限制时，节点最大/小值以sensor或isp的限制为准。当手动配置的曝光分量最大/

小值为0时，最终生效的曝光分解路线以第一次校正的分解路线为准；当手动配置的曝光分量最大/小值不为0时，且设置的最大/小值不超过sensor或isp的限制时，对曝光分解路线做第二次校正，节点最大/小值以手动设置的范围为准；若设置曝光分量的最大/小值超过sensor或isp的限制时，曝光分解路线曝光分量的节点最大/小值以第一次校正结果为准。

- 如果相邻节点的曝光量增加，则应该只有一个曝光分量增加，其他曝光分量固定。增加的分量决定该段路线的分配策略。例如增益分量增加，其他分量固定，那么该段路线的分配策略是增益优先。

## ae\_hdrExpRatioCtrl\_t

### 【说明】

定义Hdr AE曝光比控制模块结构体。

### 【定义】

```
typedef struct ae_hdrExpRatio_s {
    uint8_t sw_aeT_expRatio_len;
    float sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_m2sRatioFix_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_l2mRatioFix_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_m2sRatioMax_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_l2mRatioMax_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_hdrExpRatio_t;

typedef struct ae_hdrExpRatioCtrl_s {
    ae_expRatio_mode_t sw_aeT_expRatio_mode;
    ae_hdrExpRatio_t expRatio;
} ae_hdrExpRatioCtrl_t;
```

### 【成员】

成员名称	描述
sw_aeT_expRatio_mode	曝光比模式，仅在Hdr模式多帧合成下有效。 - ae_expRatio_auto_mode：根据场景，自动计算长短帧的曝光比 - ae_expRatio_fix_mode：长短帧采用固定曝光比
expRatio	长短帧曝光比配置。

- ae\_hdrExpRatio\_t 结构体成员如下：

成员名称	描述
sw_aeT_expRatio_len	expRatio节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	曝光量节点， <b>建议至少设置6个节点</b> ，才可实现曝光过渡的平滑，根据曝光量，动态设置曝光比固定值或曝光比最大值，二者一一对应。
sw_aeT_m2sRatioFix_dot	中帧与短帧的曝光比（Mexp/Sexp）固定值， <b>建议至少设置6个节点</b> ，才可实现曝光过渡的平滑，节点个数需要与 RatioExpDot节点个数一致。 - sw_aeT_expRatio_mode= AUTO时，无效。 - sw_aeT_expRatio_mode= FIX时，表示中帧与短帧的曝光比，与曝光量节点sw_aeT_expLevel_dot一一对应。可根据应用实际需求，支持根据曝光量节点sw_aeT_expLevel_dot设置不同的固定曝光比，也可所有曝光量节点统一设置为一样的曝光比。
sw_aeT_l2mRatioFix_dot	长帧帧与中帧的曝光比（Lexp/Mexp）固定值， <b>建议至少设置6个节点</b> ，才可实现曝光过渡的平滑，节点个数需要与 RatioExpDot节点个数一致。 - sw_aeT_expRatio_mode= AUTO时，无效。 - sw_aeT_expRatio_mode= FIX时，表示长帧与中帧的曝光比，与曝光量节点sw_aeT_expLevel_dot一一对应。可根据应用实际需求，支持根据曝光量节点sw_aeT_expLevel_dot设置不同的固定曝光比，也可所有曝光量节点统一设置为一样的曝光比。 - Hdr为2帧合成时无效，3帧合成时有效。
sw_aeT_m2sRatioMax_dot	中帧与短帧的曝光比（Mexp/Sexp）最大值， <b>建议至少设置6个节点</b> ，才可实现曝光过渡的平滑，节点个数需要与 RatioExpDot节点个数一致。 - sw_aeT_expRatio_mode= AUTO时，表示中帧与短帧的曝光比动态最大值，与曝光量节点sw_aeT_expLevel_dot一一对应。 - sw_aeT_expRatio_mode= FIX时，无效。
sw_aeT_l2mRatioMax_dot	长帧与中帧的曝光比（Lexp/Mexp）最大值， <b>建议至少设置6个节点</b> ，才可实现曝光过渡的平滑，节点个数需要与 RatioExpDot节点个数一致。 - sw_aeT_expRatio_mode= AUTO时，表示长帧与中帧的曝光比动态最大值，与曝光量节点sw_aeT_expLevel_dot一一对应。 - sw_aeT_expRatio_mode= FIX时，无效。 - Hdr为2帧合成时无效，3帧合成时有效。

### **ae\_hdrLfrmMode\_t**

#### **【说明】**

定义Hdr AE长帧模式功能控制模块结构体。

#### **【定义】**

```

typedef struct ae_hdrLfrmMode_s {
    ae_longFrm_mode_t sw_aeT_lfrm_mode;
    uint16_t     sw_aeT_sfrmTimeReg_min;
    float        sw_aeT_lfrmModeExp_thred;
} ae_hdrLfrmMode_t;

```

## 【成员】

成员名称	描述
sw_aeT_lfrm_mode	长帧模式，包括： ae_longFrm_disable_mode (auto/enable) 三种模式。 - disable：正常Hdr，不开启长帧模式，Ae和Hdr合成模块按照手动/自动曝光比进行工作。 - auto：自动长帧模式，在长帧曝光超过设定的 sw_aeT_lfrmModeExp_thred阈值时，长帧曝光时间接近HDR单帧所允许的最大曝光值，合成模块只输出长帧。 - enable：开启长帧模式，AE将短帧曝光时间固定设为最小值，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。
sw_aeT_sfrmTimeReg_min	长帧模式/自动长帧模式 (auto/enable) 下有效，短帧最小曝光行。由于sensor的一些限制，长帧模式下，短帧的最小曝光行可能无法达到sensor允许的最小曝光行，因此需要另行设置。
sw_aeT_lfrmModeExp_thred	自动长帧模式下有效，当长帧曝光超过 sw_aeT_lfrmModeExp_thred，切换为长帧模式。

## ae\_hdrLfrmCtrl\_t

### 【说明】

定义长帧调试参数结构体。HdrAE策略中，2帧模式下，长帧需要兼容一般动态范围场景和背光场景，因此具有两个亮度约束条件：全局目标亮度和暗区目标亮度。在保证长帧全局亮度位于全局目标亮度的容忍区间内的同时，要求暗区亮度大于等于暗区目标亮度。详细可见《Rockchip\_Tuning\_Guide》文档。

### 【定义】

```

typedef struct ae_hdrLfrmSetpoint_s {
    uint8_t sw_aeT_lfrmSetpoint_len;
    float  sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float  sw_aeT_nonOEPdfTh_dot[AE_ROUTE_DOT_MAX_NUM];
    float  sw_aeT_loLitPdfTh_dot[AE_ROUTE_DOT_MAX_NUM];
    float  sw_aeT_lfrmSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
    float  sw_aeT_loLitSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_hdrLfrmSetpoint_t;

typedef struct ae_hdrLfrmCtrl_s {
    float    sw_aeT_oeROI_low_thred;
    float    sw_aeT_loLv_thred;
    float    sw_aeT_hiLv_thred;
    ae_hdrLfrmSetpoint_t lfrmSetpoint;
} ae_hdrLfrmCtrl_t;

```

## 【成员】

成员名称	描述
sw_aeT_oeROI_low_thred	过曝区域亮度最低值，用于区分过曝区域与非过曝区域，取值范围[0, 255]。
sw_aeT_loLv_thred	环境亮度高阈值，无量纲，取值范围[0, 15]。
sw_aeT_hiLv_thred	环境亮度低阈值，无量纲，取值范围[0, 15]。
lfrmSetpoint	动态长帧目标值配置。

- ae\_hdrLfrmSetpoint\_t 结构体成员如下：

成员名称	描述
sw_aeT_lfrmSetpoint_len	lfrmSetpoint节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	动态长帧曝光值节点参数，建议至少设置6个节点，才可实现曝光过渡的平滑。expLevel =gain*time，time单位为s。
sw_aeT_nonOEPdfTh_dot	非过曝区域占比阈值，取值范围[0, 1]，节点个数需要与expLevel保持一致，节点值与expLevel一一对应。
sw_aeT_loLitPdfTh_dot	暗区占比阈值，取值范围[0, 1]，节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应,随着expLevel增大而增大。
sw_aeT_lfrmSetpoint_dot	动态长帧全局目标亮度值，取值范围[0, 255]，节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应。
sw_aeT_loLitSetpoint_dot	动态长帧暗区亮度目标值，取值范围[0, 255]，节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应，随着expLevel增大而减小。

## ae\_hdrMfrmCtrl\_t

### 【说明】

定义中帧调试参数结构体。

### 【定义】

```
typedef struct ae_hdrMfrmCtrl_s {
    uint8_t sw_aeT_mfrmCtrl_len;
    float sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_mfrmSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_hdrMfrmCtrl_t;
```

## 【成员】

成员名称	描述
sw_aeT_mfrmCtrl_len	mfrmCtrl节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	动态中帧曝光值节点参数，建议至少设置6个节点，才可实现曝光过渡的平滑。expLevel=gain*time (time 单位为s)。
sw_aeT_mfrmSetpoint_dot	动态中帧全局目标亮度值，取值范围[0, 255]。节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应。随曝光增长，目标值降低。

## ae\_hdrSfrmCtrl\_t

### 【说明】

定义短帧调试参数结构体。

### 【定义】

```
typedef struct ae_hdrSfrmSetpoint_s {
    uint8_t sw_aeT_sfrmSetpoint_len;
    float sw_aeT_expLevel_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_sfrmSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
    float sw_aeT_hiLitSetpoint_dot[AE_ROUTE_DOT_MAX_NUM];
} ae_hdrSfrmSetpoint_t;

typedef struct ae_hdrSfrmCtrl_s {
    bool sw_aeT_hiLitROIExpd_en;
    float sw_aeT_hiLit_tolerance;
    ae_hdrSfrmSetpoint_t sfrmSetpoint;
} ae_hdrSfrmCtrl_t;
```

### 【成员】

成员名称	描述
sw_aeT_hiLitROIExpd_en	短帧高亮区扩展使能。 - 1：忽略占比较小的高亮区，减小高亮区灵敏度。 - 0：对所有高亮区进行亮度抑制，增大高亮区灵敏度。
sw_aeT_hiLit_tolerance	设置短帧高亮区目标容忍百分比，单位为%，取值范围[0, 100]。
sfrmSetpoint	动态短帧目标值配置。

- ae\_hdrSfrmSetpoint\_t 结构体成员如下：

成员名称	描述
sw_aeT_sfrmSetpoint_len	sfrmSetpoint节点的长度，每个分量的节点个数需要一致，最多支持12个节点。
sw_aeT_expLevel_dot	动态短帧最大曝光值节点参数，建议至少设置6个节点，才可实现曝光过渡的平滑。expLevel=gain*time (time 单位为s)。
sw_aeT_sfrmSetpoint_dot	动态短帧全局平均亮度目标值，取值范围[0, 255]。节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应。同区间内的亮区亮度目标值要求高于对应全局亮度目标值。
sw_aeT_hiLitSetpoint_dot	动态短帧高亮区均值目标值，取值范围[0, 255]。节点个数需要与expLevel保持一致，节点值与expLevel各节点值一一对应。

## ae\_api\_irisAttr\_t

### 【说明】

定义光圈控制参数。

### 【定义】

```
typedef struct ae_irisCtrl_s{
    bool      sw_aeT_iris_en;
    ae_iris_type_t  sw_aeT_iris_type;
    ae_initIris_t   initIris;
    ae_manIris_t   manIris;
    ae_pIrisCtrl_t pIrisCtrl;
    ae_dCIrisCtrl_t dCIrisCtrl;
    ae_hdCIrisCtrl_t hdCIrisCtrl;
} ae_irisCtrl_t;

typedef ae_irisCtrl_t ae_api_irisAttr_t;
```

### 【成员】

成员名称	描述
sw_aeT_iris_en	自动光圈控制功能的使能。
sw_aeT_iris_type	光圈类型，包含有P（即P-iris光圈，ae_iris_p_type）或DC（即DC-iris光圈，ae_iris_dc_type）或HDC（即HDC-iris光圈，ae_iris_hdc_type）。
initIris	光圈初始值参数。
manIris	手动光圈控制参数。
pIrisCtrl	P光圈属性参数。
dCIrisCtrl	DC光圈属性参数。
hdCIrisCtrl	HDC光圈属性参数。

## ae\_initIris\_t

### 【说明】

定义光圈初始值参数。

### 【定义】

```
typedef struct ae_initIris_s {
    int sw_aeT_initPIrisGain_val;
    int sw_aeT_initDCIrisHold_val;
    int sw_aeT_initHDCIrisGain_val;
} ae_initIris_t;
```

### 【成员】

成员名称	描述
sw_aeT_initPIrisGain_val	P光圈等效增益初始值，取值范围为[1, 1024]，仅在光圈类型为P光圈时有效，默认初始值为P光圈所支持的最大光圈对应的等效增益值。等效增益的含义说明详见光圈调试参数pIrisCtrl模块。
sw_aeT_initDCIrisHold_val	DC光圈占空比初始值，取值范围为[0, 100]，仅在光圈类型为DC光圈时有效，默认初始值为DC光圈的最大占空比值，此时DC-iris将以最大速度打开光圈。占空比的含义说明详见光圈调试参数dclIrisCtrl模块。
sw_aeT_initHDCIrisGain_val	HDC光圈目标初始值，取值范围为[1, 512]，仅在光圈类型为HDC光圈时有效，默认初始值与霍尔DC光圈的zeroIsMax有关，设置为最大光圈的位置。zeroIsMax的含义说明详见hdclIrisCtrl模块。

## ae\_manIris\_t

### 【说明】

定义手动光圈控制参数。

### 【定义】

```
typedef struct ae_manIris_s {
    bool sw_aeT_manIris_en;
    int sw_aeT_manPIrisGain_val;
    int sw_aeT_manDCIrisHold_val;
    int sw_aeT_manHDCIrisGain_val;
} ae_manIris_t;
```

### 【成员】

成员名称	描述
sw_aeT_manIris_en	手动光圈使能，默认值为1。
sw_aeT_manPIrisGain_val	手动P光圈等效增益值，取值范围为[1, 1024]，仅在光圈类型为P光圈时有效，默认初始值为P光圈所支持的最大光圈对应的等效增益值。等效增益的含义说明详见光圈调试参数plirisCtrl模块。
sw_aeT_manDCIrisHold_val	手动DC光圈占空比值，取值范围为[0, 100]，仅在光圈类型为DC光圈时有效，默认初始值为DC光圈的最大占空比值，此时DC-iris将以最大速度打开光圈。占空比的含义说明详见光圈调试参数dcirisCtrl模块。
sw_aeT_manHDCIrisGain_val	手动HDC光圈目标值，取值范围为[1, 512]，仅在光圈类型为HDC光圈时有效，默认初始值与霍尔DC光圈的zerolsMax有关，设置为最大光圈的位置。zerolsMax的含义说明详见hdclirisCtrl模块。

## ae\_plirisCtrl\_t

### 【说明】

定义P光圈 (precise iris) 控制参数。

### 【定义】

```
typedef struct ae_plirisCtrl_s {
    uint16_t sw_aeT_totalStep_val;
    uint16_t sw_aeT_effcStep_val;
    bool    sw_aeT_zerolsMax_en;
    uint16_t sw_aeT_step2Gain_table[AE_PIRIS_TABLE_MAX_STEP];
} ae_plirisCtrl_t;
```

### 【成员】

成员名称	描述
sw_aeT_totalStep_val	P-iris步进电机总步数，具体大小与P-iris镜头有关。
sw_aeT_effcStep_val	P-iris步进电机的可用步数，具体大小与P-iris镜头有关。
sw_aeT_zerolsMax_en	P-iris步进电机step0是否对应最大光圈位置，具体取值与P-iris镜头有关。该值为0，代表步进电机位置为step0时，光圈开到最小；该值为1，代表步进电机位置为step0时，光圈开到最大。
sw_aeT_step2Gain_table	P-iris步进电机位置与光圈等效增益的映射表，具体数值与P-iris镜头有关。

## ae\_dcIrisCtrl\_t

### 【说明】

定义dc光圈 (dc iris without hall) 控制参数。

### 【定义】

```

typedef struct ae_dclrisCtrl_s {
    float sw_aeT_dclris_Kp;
    float sw_aeT_dclris_Ki;
    float sw_aeT_dclris_Kd;
    int sw_aeT_pwmDuty_min;
    int sw_aeT_pwmDuty_max;
    int sw_aeT_pwmDuty_open;
    int sw_aeT_pwmDuty_close;
} ae_dclrisCtrl_t;

```

### 【成员】

成员名称	描述
sw_aeT_dclris_Kp	比例系数，用于限制光圈剧烈变化时光圈的开关速度，该值越大，光线剧烈变化时光圈打开和关闭的速度越慢。该值过大，调节过程制动就会超前，致使调节时间过长；该值过小，调节过程制动就会落后，从而导致超调增加。该值的合理设置与DC-iris镜头及电路特性有关，取值范围[0, 1]，建议值为0.5。
sw_aeT_dclris_Ki	积分系数，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越大。该值过大，容易出现超调导致振荡；该值过小，光圈调节速度较慢、环境亮度变化较剧烈时容易发生振荡。取值范围[0, 1]，建议值为0.2。
Kdsw_aeT_dclris_Kd	微分系数，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越大。取值范围[0, 1]，建议值为0.3。
sw_aeT_pwmDuty_min	最小PWM占空比，具体大小与DC-iris镜头、电路特性有关，单位为%。该值越小，所支持的光圈关闭速度越快，但容易导致光圈振荡。取值范围[0, 100]，默认值为0。
sw_aeT_pwmDuty_max	最大PWM占空比，具体大小与DC-iris镜头、电路特性有关，单位为%。该值越大，所支持的光圈打开速度越快，该值过小，可能导致光圈尚未达到最大时就退出光圈控制。取值范围[0, 100]，默认值为100。
sw_aeT_pwmDuty_open	光圈打开时的PWM占空比阈值，当光圈PWM占空比高于（不含）pwmDuty_open时，光圈处于打开状态。具体大小与DC-iris镜头有关，单位为%，取值范围[0, 100]。
sw_aeT_pwmDuty_close	光圈关闭时的PWM占空比阈值，当光圈PWM占空比小于（不含）pwmDuty_close时，光圈处于关闭状态。具体大小与DC-iris镜头有关，单位为%，取值范围[0, 100]。

### ae\_hdclrisCtrl\_t

#### 【说明】

定义hdclris光圈 (dc iris with hall) 控制参数。

#### 【定义】

```

typedef struct ae_hdclrisZoom_s {
    int sw_aeC_zoom2Iris_len;
    int sw_aeC_zoom2Iris_idx[AE_HDCIRIS_DOT_MAX_NUM];
}

```

```

int sw_aeC_zoom2Iris_val[AE_HDCIRIS_DOT_MAX_NUM];
} ae_hdclirisZoom_t;

typedef struct ae_hdclirisGain_s {
    int sw_aeC_iris2Gain_len;
    int sw_aeC_iris2Gain_idx[AE_HDCIRIS_DOT_MAX_NUM];
    int sw_aeC_iris2Gain_val[AE_HDCIRIS_DOT_MAX_NUM];
} ae_hdclirisGain_t;

typedef struct ae_hdclirisCtrl_s {
    float      sw_aeT_damp_over;
    float      sw_aeT_damp_under;
    bool       sw_aeT_zerolsMax_en;
    int        sw_aeT_target_min;
    int        sw_aeT_target_max;
    ae_hdclirisZoom_t  zoom2Iris;
    ae_hdclirisGain_t  iris2Gain;
} ae_hdclirisCtrl_t;

```

## 【成员】

成员名称	描述
sw_aeT_damp_over	自动光圈打开的阻尼系数，该值越小，所支持的光圈打开速度越快，取值范围[0, 1]。
sw_aeT_damp_under	自动光圈闭合的阻尼系数，该值越小，所支持的光圈闭合速度越快，取值范围[0, 1]。
sw_aeT_zerolsMax_en	光圈目标值为0时光圈是否全开，全开为1，全闭为0，该值与光圈驱动设置有关。
sw_aeT_target_min	光圈目标值可调节的最小值，该值与光圈驱动设置有关，取值范围[0, 1023]。
sw_aeT_target_max	光圈目标值可调节的最大值，该值与光圈驱动设置有关，取值范围[0, 1023]。
zoom2Iris	zoom步进电机位置与最佳光圈目标值的映射表。
iris2Gain	光圈目标值与光圈等效增益的映射表。

- ae\_hdclirisZoom\_t 和ae\_hdclirisGain\_t 结构体成员如下：

成员名称	描述
sw_aeC_zoom2Iris_len	zoom2Iris映射表长度，每个分量的节点个数需要一致，最大支持256个节点。
sw_aeC_zoom2Iris_idx	各个倍率下对应的zoom步进电机位置，可参照AF模块中的zoomPosition参数填写，最多支持设置256个值。
sw_aeC_zoom2Iris_val	各个倍率下对应的最佳光圈目标值，可根据镜头厂商提供的映射表填写，与sw_aeC_zoom2Iris_idx的节点一一对应，最多支持设置256个值。
sw_aeC_iris2Gain_len	iris2Gain映射表长度，每个分量的节点个数需要一致，最大支持256个节点。
sw_aeC_iris2Gain_idx	光圈目标值调节的节点，需要标定，按照光圈通光面积从大到小填写，最多支持设置256个值。
sw_aeC_iris2Gain_val	光圈目标值对应的光圈等效增益值，与sw_aeC_iris2Gain_idx的节点一一对应，最多支持设置256个值。sw_aeC_iris2Gain_val按照 <b>从大到小的顺序</b> 填写，第一个节点默认是光圈全开时对应的光圈等效增益512。

## ae\_api\_queryInfo\_t

### 【说明】

定义AE曝光参数查询。

### 【定义】

```

typedef struct ae_expParam_s {
    float integration_time;
    float analog_gain;
    float digital_gain;
    float isp_dgain;
    int iso;
    int dcg_mode;
    int longfrm_mode;
} ae_expParam_t;

typedef struct ae_queryInfo_s {
    bool      isConverged;
    bool      isExpMax;
    bool      envChange;
    ae_linExpInfo_t linExpInfo;
    ae_hdrExpInfo_t hdrExpInfo;
    float     gblEnvLv;
    float     oeROIPdf;
    float     hiLitROIPdf;
    float     loLitROIPdf;
    float     vts;
    float     hts;
    float     pclk;
    float     fps;
} ae_queryInfo_t;

typedef ae_queryInfo_t ae_api_queryInfo_t;

```

## 【成员】

成员名称	描述
isConverged	自动曝光是否收敛。
isExpMax	ISP曝光是否达到最大值。
envChange	环境是否变化。
linExpInfo	线性曝光信息。
hdrExpInfo	Hdr曝光信息。
gblEnvLv	全局的环境亮度。
oeROIPdf	过曝区域占比。
hiLitROIPdf	高光区域占比。
loLitROIPdf	低光区域占比。
vts	sensor当前的VTS。
hts	sensor当前的HTS。
pclk	sensor的像素时钟频率(单位：兆赫兹)。
fps	sensor当前的出图帧率，注意该值与系统实际出流帧率定义不同，值可能存在偏差。

## ae\_linExpInfo\_t

### 【说明】

定义线性曝光信息结构体。

### 【定义】

```
typedef struct ae_linExpInfo_s {
    float    devLuma;
    float    meanLuma;
    ae_expRange_t expRange;
    ae_expParam_t expParam;
} ae_linExpInfo_t;
```

## 【成员】

成员名称	描述
devLuma	当前均值亮度与目标亮度的亮度差值比， $devLuma = (meanLuma - SetPoint)/SetPoint$ 。
meanLuma	当前均值亮度，取值范围[0, 255]。
expRange	当前线性曝光分量的range，包含增益range与曝光时间range。
expParam	当前线性曝光分量值，包含sensor的total gain与曝光时间。

- ae\_expParam\_t 结构体成员如下：

成员名称	描述
integration_time	曝光积分时间值，以秒为单位。
analog_gain	sensor模拟增益值，此处增益值为实际值，即sensor total gain，单位为1x。
digital_gain	sensor数字增益值，单位为1x，当sensor的数字增益起弥补模拟增益精度作用时，配置为1x，整体增益值写入analog_gain中。
isp_dgain	isp数字增益值，单位为1x。
iso	总增益值，iso表示系统增益，以常数50乘以倍数为单位，iso = analog_gain * digital_gain * isp_dgain*50。
dcg_mode	dcg模式，-1表示不支持dcg模式，0表示LCG，1表示HCG。

- ae\_expRange\_t 结构体参考前文介绍。

### ae\_hdrExplInfo\_t

#### 【说明】

定义Hdr曝光信息结构体。

#### 【定义】

```
typedef struct ae_hdrExplInfo_s {
    float    devLuma[AE_HDRFRAME_MAX_NUM];
    float    frm0Luma;
    float    frm1Luma;
    float    frm2Luma;
    ae_expRange_t expRange[AE_HDRFRAME_MAX_NUM];
    ae_expParam_t expParam[AE_HDRFRAME_MAX_NUM];
} ae_hdrExplInfo_t;
```

#### 【成员】

成员名称	描述
devLuma	HDR模式下，当前各帧均值亮度与各帧目标亮度的亮度差值比。2帧模式下，元素0与元素1，分别代表短帧与长帧的亮度差值比；3帧模式下，元素0、元素1、元素2，分别代表短帧、中帧、长帧的亮度差值比。
frm0Luma	HDR曝光2/3帧模式下，代表短帧均值亮度，取值范围[0, 255]。
frm1Luma	HDR曝光2帧模式下，代表长帧帧均值亮度；HDR曝光3帧模式下，代表中帧均值亮度。取值范围[0, 255]。
frm2Luma	仅在HDR曝光3帧模式下有效，代表长帧帧均值亮度，取值范围[0, 255]。
expRange	HDR曝光模式下的曝光分量range，2帧模式下，元素0与元素1，分别代表短帧与长帧的曝光分量range；3帧模式下，元素0、元素1、元素2，分别代表短帧、中帧、长帧的曝光分量range。
expParam	HDR曝光模式下的当前曝光分量值，包含sensor的total gain与曝光时间。2帧模式下，元素0与元素1，分别代表短帧与长帧的曝光分量值；3帧模式下，元素0、元素1、元素2，分别代表短帧、中帧、长帧的曝光分量值。

- ae\_expParam\_t 结构体参考前文介绍。
- ae\_expRange\_t 结构体参考前文介绍。

## Uapi\_ExpWin\_t

### 【说明】

定义AE统计窗口属性参数。

### 【定义】

```
typedef struct window {
    uint16_t h_offs;
    uint16_t v_offs;
    uint16_t h_size;
    uint16_t v_size;
} window_t;

typedef struct Uapi_ExpWin_s {
    rk_aiq_uapi_sync_t sync;
    window      Params;
} Uapi_ExpWin_t;
```

### 【成员】

成员名称	描述
h_offs	窗口左上角相对坐标原点的水平偏移值，这里的坐标原点指sensor感光区域左上角。
v_offs	窗口左上角相对坐标原点的垂直偏移值，这里的坐标原点指sensor感光区域左上角。
h_size	窗口水平方向尺寸。
v_size	窗口竖直方向尺寸。

## **Uapi\_AecStatsCfg\_t**

### **【说明】**

定义AE统计值通道配置结构体。

### **【定义】**

```
typedef struct Uapi_AecStatsCfg_s {
    rk_aiq_uapi_sync_t sync;
    bool updateStats;
    bool YChannelEn;
    bool RChannelEn;
    bool GChannelEn;
    bool BChannelEn;
} Uapi_AecStatsCfg_t;
```

### **【成员】**

成员名称	描述
updateStats	是否每帧都更新AE统计值。若设置为false，在RK AE收敛后，不再更新AE统计值。
YChannelEn	是否赋值Y通道AE统计值。
RChannelEn	是否赋值R通道AE统计值。
GChannelEn	是否赋值G通道AE统计值。
BChannelEn	是否赋值B通道AE统计值。

## **常见问题定位及debug方法**

若出现画面亮度闪烁、过冲、亮度不符合预期等问题时，建议通过抓取有问题场景的AE LOG进行分析，有助于快速定位问题、提高工作效率。

## **曝光统计同步测试功能**

标定ISP模块前，需要驱动人员或tuning人员填写调试IQ XML中的SensorInfo参数。这个模块的涉及到曝光参数的设置，如设置错误可能发生曝光出错、闪烁等现象。建议配置完sensorinfo参数之后，开启SyncTest功能进行自测。sensorinfo参数含义说明参考《Rockchip\_Tuning\_Guide\_ISP33》。

SyncTest功能通过循环设置N组不同曝光值，可测试sensor的曝光时间和曝光增益、及DCG切换生效帧数是否正确，还可用于测试曝光的线性度，从而确认曝光时间和曝光增益的寄存器值转换公式及相关参数是否正确。

SyncTest功能参数介绍如下：

### **【描述】**

曝光与统计的同步测试功能，支持按照给定间隔帧数，循环设置N组不同的曝光值，用于debug及验证曝光分量（曝光时间、曝光增益）的生效帧数及sensor曝光参数设置是否正确。

### **【成员】**

成员名称	描述
sw_aeT_syncTest_en	曝光与统计同步测试功能的使能
sw_aeT_syncTest_interval	曝光切换间隔帧数
alterExp	曝光切换参数

- alterExp

根据模式的不同，分为linAlterExp和hdrAlterExp两套参数。

成员名称	描述
sw_aeT_time_val	曝光时间值
sw_aeT_gain_val	曝光增益值
sw_aeT_ispDGain_val	Isp数字增益值
sw_aeT_dcg_mode	Dcg模式值
sw_aeT_pirisGain_val	P-iris等效增益值

如参数设置正确，LOG示例如下（仅截取LOG中的关键信息）。红框所示为曝光切换位置，可见亮度并无发生突变且与曝光值相匹配，无延迟或提前线性，此时可基本判断sensorinfo参数设置正确。

```
>>> rramenum=116 Cur gain=5.988304,time=0.019991,meanluma=44.751110,piris=0
>>> Framenum=117 Cur gain=5.988304,time=0.019991,Meanluma=44.755554,piris=0
>>> Framenum=118 Cur gain=5.988304,time=0.019991,Meanluma=44.755554,piris=0
>>> Framenum=119 Cur gain=5.988304,time=0.019991,Meanluma=44.764446,piris=0
>>> Framenum=120 Cur gain=5.988304,time=0.019991,Meanluma=44.755554,piris=0
>>> Framenum=121 Cur gain=5.988304,time=0.019991,Meanluma=44.755554,piris=0
>>> Framenum=122 Cur gain=5.988304,time=0.019991,Meanluma=44.768890,piris=0
>>> Framenum=123 Cur gain=5.988304,time=0.019991,Meanluma=44.782223,piris=0
>>> Framenum=124 Cur gain=1.000000,time=0.019991,Meanluma=24.568890,piris=0
>>> Framenum=125 Cur gain=1.000000,time=0.019991,Meanluma=24.484444,piris=0
>>> Framenum=126 Cur gain=1.000000,time=0.019991,Meanluma=24.484444,piris=0
>>> Framenum=127 Cur gain=1.000000,time=0.019991,Meanluma=24.484444,piris=0
>>> Framenum=128 Cur gain=1.000000,time=0.019991,Meanluma=24.480000,piris=0
>>> Framenum=129 Cur gain=1.000000,time=0.019991,Meanluma=24.480000,piris=0
>>> Framenum=130 Cur gain=1.000000,time=0.019991,Meanluma=24.471111,piris=0
>>> Framenum=131 Cur gain=1.000000,time=0.019991,Meanluma=24.475555,piris=0
```

## 曝光变化时出现闪烁

可能导致曝光变化时出现闪烁的几种原因：

(1) CISExpUpdate模块中的gain、time生效时刻帧数错误。常见LOG示例如下（仅截取每帧关键LOG行）：

```
Cur gain=1.937500,time=0.010015,RawMeanluma=24.622223,YuvMeanluma=34.124443,IsConverged=0
Cur gain=1.937500,time=0.010015,RawMeanluma=37.795555,YuvMeanluma=54.217777,IsConverged=0
Cur gain=1.937500,time=0.010015,RawMeanluma=37.257778,YuvMeanluma=52.435555,IsConverged=0
Cur gain=1.328125,time=0.020000,RawMeanluma=37.288887,YuvMeanluma=52.480000,IsConverged=0
Cur gain=1.390625,time=0.020000,RawMeanluma=60.342224,YuvMeanluma=82.528893,IsConverged=0
Cur gain=1.453125,time=0.020000,RawMeanluma=46.471111,YuvMeanluma=63.831112,IsConverged=0
Cur gain=1.000000,time=0.030015,RawMeanluma=48.048889,YuvMeanluma=66.195557,IsConverged=0
Cur gain=1.187500,time=0.020000,RawMeanluma=65.511108,YuvMeanluma=87.622223,IsConverged=0
Cur gain=1.125000,time=0.020000,RawMeanluma=38.071110,YuvMeanluma=53.022221,IsConverged=0
Cur gain=1.062500,time=0.020000,RawMeanluma=42.928890,YuvMeanluma=60.355556,IsConverged=0
Cur gain=1.640625,time=0.010015,RawMeanluma=41.328888,YuvMeanluma=57.666668,IsConverged=0
Cur gain=1.593750,time=0.010015,RawMeanluma=25.453333,YuvMeanluma=35.293335,IsConverged=0
Cur gain=1.562500,time=0.010015,RawMeanluma=33.360001,YuvMeanluma=47.897778,IsConverged=0
Cur gain=1.531250,time=0.010015,RawMeanluma=32.595554,YuvMeanluma=46.084446,IsConverged=0
```

红框所标为亮度出错位置，可见随着曝光增长或降低，对应亮度与曝光变化趋势相反。通过观察，可发现亮度突变都发生在曝光时间和曝光增益同时变化后的第二帧。亮度的变化趋势与曝光时间变化趋势一致，因此可以判断gain、time的生效帧数出错，曝光时间和曝光增益的变化并未同时生效，导致亮度和曝光不匹配。可以通过修改gain、time生效帧数解决此问题。上述问题可以使用AE的syncTest功能进行复现，设置两组曝光（曝光增益和曝光时间不同），令其来回切换，查看LOG可知是否复现。

(2) 驱动错误导致的亮度来回震荡无法收敛，常发生在高亮场景，常见LOG示例如下（仅截取每帧关键LOG行）

```
Framenum=3378 Cur Piris=128, Sgain=1.188502,Stime=0.000044,m
Framenum=3379 Cur Piris=128, Sgain=1.148154,Stime=0.000044,m
Framenum=3380 Cur Piris=128, Sgain=1.148154,Stime=0.000044,m
Framenum=3381 Cur Piris=128, Sgain=1.230269,Stime=0.000044,m
Framenum=3382 Cur Piris=128, Sgain=1.000000,Stime=0.000059,:  

Framenum=3383 Cur Piris=128, Sgain=1.035142,Stime=0.000059,:  

Framenum=3384 Cur Piris=128, Sgain=1.035142,Stime=0.000059,:  

Framenum=3385 Cur Piris=128, Sgain=1.071519,Stime=0.000059,:  

Framenum=3386 Cur Piris=128, Sgain=1.273503,Stime=0.000044,:  

Framenum=3387 Cur Piris=128, Sgain=1.188502,Stime=0.000044,:  

Framenum=3388 Cur Piris=128, Sgain=1.148154,Stime=0.000044,:  

Framenum=3389 Cur Piris=128, Sgain=1.148154,Stime=0.000044,:  

Framenum=3390 Cur Piris=128, Sgain=1.230269,Stime=0.000044,:  

Framenum=3391 Cur Piris=128, Sgain=1.000000,Stime=0.000059,:  

Framenum=3392 Cur Piris=128, Sgain=1.000000,Stime=0.000059,m
Framenum=3393 Cur Piris=128, Sgain=1.000000,Stime=0.000059,m
Framenum=3394 Cur Piris=128, Sgain=1.035142,Stime=0.000059,:  

Framenum=3395 Cur Piris=128, Sgain=1.273503,Stime=0.000044,:  

Framenum=3396 Cur Piris=128, Sgain=1.188502,Stime=0.000044,:  

Framenum=3397 Cur Piris=128, Sgain=1.148154,Stime=0.000044,:  

Framenum=3398 Cur Piris=128, Sgain=1.148154,Stime=0.000044,:  

Framenum=3399 Cur Piris=128, Sgain=1.230269,Stime=0.000044,:  


```

如图，可观察到曝光在59ms和44ms之间来回震荡，具体查看59ms和44ms对应的亮度均值会发现，二者的亮度之比与曝光之比差距较大，线性度有问题。第一步需要进行sensor的驱动检测，在驱动打印曝光寄存器值，查看寄存器值是否与log中的曝光一致。上述问题可以使用AE的syncTest功能进行复现，设置多组曝光（包含出现问题的曝光），令其来回切换，查看LOG中每帧曝光对应的亮度变化是否满足线性。

```
rk_aiq_ae_algo.cpp:6405: ===== HDR-AE (enter) =====
rk_aiq_ae_algo.cpp:6425: AecRun: SMeanLuma=25.166803 MMeanLuma=85.886871,LMeanLuma=0.000000,TmoMeanluma=35.152767,Isconverged=0.
rk_aiq_ae_algo.cpp:6434: >>> Framenum=3385 Cur Piris=128, Sgain=1.071519,Stime=0.000059,mgain=1.071519,mtime=0.000207,lgain=0.
rk_aiq_ae_algo.cpp:3564: S-HighLightLuma=72.000000,S-Target=100.000000,S-GlobalLuma=25.166803,S-Target=19.998999
rk_aiq_ae_algo.cpp:3909: L-LowLightLuma=56.696957,L-Target=49.991318,L-GlobalLuma=85.886871,L-Target=79.985535
rk_aiq_ae_algo.cpp:5385: AecHdrlClmExecute: sgain=1.000000,stime=0.000051,mgain=1.000000,mtime=0.000220,lgain=0.000000,ltime=0.
rk_aiq_ae_algo.cpp:6555: calc result:piris=128,sgain=1.148154,stime=0.000044,mgain=1.071519,mtime=0.000207,lgain=0.000000,ltime=0.
rk_aiq_ae_algo.cpp:6559: ===== (exit) =====

rk_aiq_algo_ae_itf.cpp:256: Cur-Exp: FrmId=3386,S-gain=0x7,S-time=0xc,M-gain=0x2,M-time=0x38,L-gain=0x0,L-time=0x0,envChange=1
rk_aiq_algo_ae_itf.cpp:264: Last-Res:FrmId=3385,S-gain=0x4,S-time=0xc,M-gain=0x2,M-time=0x38,L-gain=0x0,L-time=0x0

rk_aiq_ae_algo.cpp:6405: ===== HDR-AE (enter) =====
rk_aiq_ae_algo.cpp:6425: AecRun: SMeanLuma=15.018992, MMeanLuma=85.981834,LMeanLuma=0.000000,TmoMeanluma=31.430223,Isconverged=0.
rk_aiq_ae_algo.cpp:6434: >>> Framenum=3386 Cur Piris=128, Sgain=1.273503,Stime=0.000044,mgain=1.071519,mtime=0.000207,lgain=0.
rk_aiq_ae_algo.cpp:3564: S-HighLightLuma=43.000000,S-Target=100.000000,S-GlobalLuma=15.018992,S-Target=19.999107
rk_aiq_ae_algo.cpp:3909: L-LowLightLuma=56.738033,L-Target=49.991318,L-GlobalLuma=85.981834,L-Target=79.985535
rk_aiq_ae_algo.cpp:5385: AecHdrlClmExecute: sgain=1.000000,stime=0.000051,mgain=1.000000,mtime=0.000220,lgain=0.000000,ltime=0.
rk_aiq_ae_algo.cpp:6555: calc result:piris=128,sgain=1.273503,stime=0.000044,mgain=1.071519,mtime=0.000207,lgain=0.000000,ltime=0.
rk_aiq_ae_algo.cpp:6559: ===== (exit) =====
```

(3) AE后续模块导致的闪烁，常见LOG示例如下（仅截取每帧关键LOG行）：

```
AecRun: SMeanLuma=12.698849, MMeanLuma=240.257675,LMeanLuma=0.000000,TmoMeanluma=104.390663,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=12.944373, MMeanLuma=244.828003,LMeanLuma=0.000000,TmoMeanluma=104.161766,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=12.950768, MMeanLuma=245.728897,LMeanLuma=0.000000,TmoMeanluma=102.967392,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=10.402813, MMeanLuma=222.482101,LMeanLuma=0.000000,TmoMeanluma=79.687981,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=8.134911, MMeanLuma=159.046036,LMeanLuma=0.000000,TmoMeanluma=60.763428,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=6.737852, MMeanLuma=127.505112,LMeanLuma=0.000000,TmoMeanluma=54.783249,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=6.527493, MMeanLuma=123.283249,LMeanLuma=0.000000,TmoMeanluma=54.739769,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=6.310742, MMeanLuma=119.683502,LMeanLuma=0.000000,TmoMeanluma=63.102303,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=5.210998, MMeanLuma=95.413681,LMeanLuma=0.000000,TmoMeanluma=53.315216,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=5.067775, MMeanLuma=86.629799,LMeanLuma=0.000000,TmoMeanluma=49.849743,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=4.646420, MMeanLuma=78.632355,LMeanLuma=0.000000,TmoMeanluma=46.617645,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=4.376598, MMeanLuma=76.865089,LMeanLuma=0.000000,TmoMeanluma=45.372761,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=4.205883, MMeanLuma=74.497444,LMeanLuma=0.000000,TmoMeanluma=43.842072,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=3.946291, MMeanLuma=74.496162,LMeanLuma=0.000000,TmoMeanluma=43.543480,Isconverged=0,Longfrm=0
AecRun: SMeanLuma=3.789642, MMeanLuma=74.514069,LMeanLuma=0.000000,TmoMeanluma=43.231457,Isconverged=0,Longfrm=0
```

从LOG中可知左侧SMeanLuma和MMeanLuma递减的过程中，TmoMeanluma模块输出的亮度发生了突变，发生这种情况时需至TMO模块进行debug。

## AWB

### 概述

AWB模块的功能是通过改变拍摄设备的色彩通道的增益，对色温环境所造成颜色偏差和拍摄设备本身所固有的色彩通道增益的偏差进行统一补偿，从而让获得的图像能正确反映物体的真实色彩。

### 重要概念

- 色温：色温是按绝对黑体来定义的，光源的辐射在可见区和绝对黑体的辐射完全相同时，此时黑体的温度就称此光源的色温。
- 白平衡：在不同色温的光源下，白色在传感器中的响应会偏蓝或偏红。白平衡算法通过调整 R, G, B 三个颜色通道的强度，使白色真实呈现。

### 功能描述

AWB 模块有WB 信息统计及 AWB 策略控制算法两部分组成。

### 功能级API参考

参考章节[IMGPROC功能级API](#)AWB相关

### 模块级API参考

#### rk\_aiq\_user\_api2\_awb\_SetAttrib

##### 【描述】

设置白平衡API支持的全部参数。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	白平衡API支持的全部参数，同json结构体定义	输出

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件: librkaiq.so

## 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_GetAttrib**

#### 【描述】

获取白平衡API支持的全部参数。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	白平衡API支持的全部参数	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件: librkaiq.so

## 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_QueryWBInfo**

#### 【描述】

获取白平衡增益系数，色温。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_QueryWBInfo(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_wb_querry_info_t  
*wb_querry_info);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
wb_querry_info	颜色相关状态参数	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_Lock**

#### 【描述】

锁定当前白平衡参数。

#### 【语法】

```
XCamReturn
rk_aiq_user_api2_awb_Lock(const rk_aiq_sys_ctx_t* sys_ctx);
```

#### 【参数】

参数名称	描述	输入/输出
sys_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

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_Unlock**

### 【描述】

解锁已被锁定的白平衡参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_Lock(const rk_aiq_sys_ctx_t* sys_ctx);
```

### 【参数】

参数名称	描述	输入/输出
sys_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

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

### 【示例】

- 参考sample\_awb\_module2.c

## rk\_aiq\_user\_api2\_awb\_SetWbGainCtrlAttrib

### 【描述】

设置白平衡增益控制参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_SetWbGainCtrlAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_gainCtrl_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
awb_gainCtrl_t	白平衡增益控制参数	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件: librkaiq.so

## 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_GetWbGainCtrlAttrib**

#### 【描述】

获取白平衡增益控制参数。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_awb_GetWbGainCtrlAttrib(const rk_aiq_sys_t* sys_ctx, awb_gainCtrl_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	白平衡增益控制参数	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件: librkaiq.so

## 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_SetAwbStatsAttrib**

#### 【描述】

设置自动白平衡模式下统计相关的参数。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_awb_SetAwbStatsAttrib(const rk_aiq_sys_t* sys_ctx, awb_Stats_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡模式下统计相关的参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module2.c

### **rk\_aiq\_user\_api2\_awb\_GetAwbStatsAttrib**

#### 【描述】

获取自动白平衡模式下统计相关的参数。

#### 【语法】

```
XCamReturn
rk_aiq_user_api2_awb_GetAwbStatsAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_Stats_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡模式下统计相关的参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module2.c

## **rk\_aiq\_user\_api2\_awb\_SetAwbGnCalcStepAttrib**

### **【描述】**

设置自动白平衡gain计算相关参数。

### **【语法】**

```
XCamReturn
rk_aiq_user_api2_awb_SetAwbGnCalcStepAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_gainCalcStep_t* attr);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡gain计算相关参数	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

### **【示例】**

- 参考sample\_awb\_module2.c

## **rk\_aiq\_user\_api2\_awb\_GetAwbGnCalcStepAttrib**

### **【描述】**

获取自动白平衡gain计算相关参数。

### **【语法】**

```
XCamReturn
rk_aiq_user_api2_awb_GetAwbGnCalcStepAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_gainCalcStep_t* attr);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡gain计算相关参数	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

### 【示例】

- 参考sample\_awb\_module2.c

## **rk\_aiq\_user\_api2\_awb\_SetAwbGnCalcOthAttrib**

### 【描述】

设置自动白平衡增益计算其他参数。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_awb_SetAwbGnCalcOthAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_gainCalcOth_t*
attr);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡增益计算其他参数	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

### 【示例】

- 参考sample\_awb\_module2.c

## **rk\_aiq\_user\_api2\_awb\_GetAwbGnCalcOthAttrib**

### 【描述】

获取自动白平衡增益计算其他参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_awb_GetAwbGnCalcOthAttrib(const rk_aiq_sys_ctx_t* sys_ctx, awb_gainCalcOth_t*  
attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	自动白平衡增益计算其他参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_awb\_v3.h、rk\_aiq\_uapiv3\_awb\_int.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module2.c

## 模块级API数据类型

### **rk\_aiq\_wb\_querry\_info\_t**

#### 【说明】

定义白平衡查询信息

#### 【定义】

```
typedef struct rk_aiq_wb_querry_info_s {  
    rk_aiq_wb_gain_t gain; //effective gain  
    rk_aiq_wb_cct_t cctGloabl;  
    bool awbConverged;  
    uint32_t LVValue;  
    rk_aiq_wb_gain_t stat_gain_glb;  
    rk_aiq_wb_gain_t stat_gain_blk;  
    rk_aiq_op_mode_t opMode;  
} rk_aiq_wb_querry_info_t;
```

#### 【成员】

成员名称	描述
gain	生效的白平衡增益
cctGloabl	全局色温参数
awbConverged	白平衡是否收敛
LVValue	相关环境亮度
stat_gain_glb	自动白平衡增益
stat_gain_blk	块平均的白平衡增益
opMode	生效的白平衡增益控制模式

## awb\_gainCtrl\_t

### 【说明】

白平衡增益控制参数，包括自动/手动白平衡模式及手动白平衡增益配置

### 【定义】

```
typedef struct awb_gainCtrl_s {
    rk_aiq_op_mode_t opMode;
    wb_mwb_t manualPara;
} awb_gainCtrl_t;
```

### 【成员】

参考 文档《Rockchip\_Color\_Optimization\_Guide》章节“白平衡增益控制(wbGainCtrl)”

## awb\_Stats\_t

### 【说明】

定义自动白平衡模式下统计相关的参数，包括白点检测参数，输入源配置等

### 【定义】

```
typedef struct awb_Stats_s {
    awbStats_src_mode_t hw_awbCfg_statsSrc_mode;
    bool hw_awbCfg_lsc_en;
    bool hw_awbCfg_uvDct_en;
    bool hw_awbCfg_xyDct_en;
    bool sw_awbCfg_lgtPrefer_en;
    bool sw_awbCfg_lgtSrcWgt_en;
    awbStats_ds_mode_t hw_awbCfg_ds_mode;
    awbStats_win_t mainWin;
    awb_rgbySpace_t rgbyLimit;
    awb_rgb2xy_para_t rgb2xy;
    awb_extRange_t extraWpRange;
    awb_luma2WpWgt_t luma2WpWgt;
    awb_earlAct_t earlierAwbAct;
    awb_blc_t blc2ForAwb;
    int lightSources_len;
    awb_lgtSrc_t lightSources[CALD_AWB_LS_NUM_MAX];
    awb_zoneWgt_t zoneWgt;
    bool hw_awbCfg_zoneStats_en;
}
```

```
    awbStats_pixEngineSrc_mode_t hw_awbCfg_zoneStatsSrc_mode;
} awb_Stats_t;
```

### 【成员】

参考 文档《Rockchip\_Color\_Optimization\_Guide》章节“自动白平衡统计相关 (awbStats) ”

## awb\_gainCalcStep\_t

### 【说明】

定义自动白平衡增益计算其他参数。

### 【定义】

```
typedef struct awb_gainCalcStep_s {
    awb_ganCalcMethod_e gnCalc_method;
    awb_div_t division;
    awb_wbGnType1Calc_t wbGnType1;
    awb_wbGnType3Calc_t wbGnType3;
    int wbGnExt_len;
    awb_wbGnExtrCalc_t wbGnExt[CALID_AWB_GNEXT_GN_NUM_MAX];
    awb_dayLgtClip_Cfg_t wbGainDaylightClip;
    awb_gainClip_t wbGainClip;
    awb_gainAdjust_t wbGainAdjust;
    awb_gainOffset_t wbGainOffset;
    awb_dampFactor_t dampFactor;
} awb_gainCalcStep_t;
```

### 【成员】

参考 文档《Rockchip\_Color\_Optimization\_Guide》章节“自动白平衡增益计算相关 (awbGnCalcStep) ”

## awb\_gainCalcOth\_t

### 【说明】

定义自动白平衡增益计算其他参数。

### 【定义】

```
typedef struct awb_gainCalcOth_s{
    float fstFrm_wbgain[4];
    awb_advSiteRec_t advSiteRec;
    awb_smartRun_t tolerance;
    awb_runinterval_t runInterval;
    awb_converge_t converged;
    awb_ctCalc_t ctCalc;
}awb_gainCalcOth_t;
```

### 【成员】

参考 文档《Rockchip\_Color\_Optimization\_Guide》章节“其他 (awbGnCalcOth) ”

## awb\_api\_attrib\_t

### 【说明】

定义白平衡API支持的全部参数。

## 【定义】

```
typedef struct{
    awb_gainCtrl_t wbGainCtrl;
    awb_Stats_t awbStats;
    awb_gainCalcStep_t awbGnCalcStep;
    awb_gainCalcOth_t awbGnCalcOth;
} awb_api_attrib_t;
```

## 【成员】

各成员的结构体定义参考本文档中“awb\_gainCtrl\_t”，“awb\_Stats\_t”，“awb\_gainCalcStep\_t”，“awb\_gainCalcOth\_t”章节描述

# AF

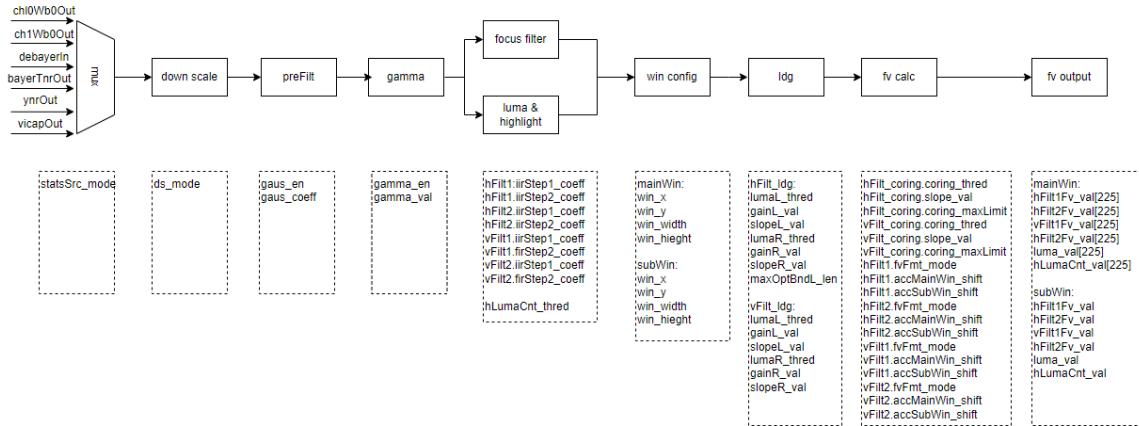
## 概述

AF模块的功能是指调整相机镜头，使被拍物成像清晰的过程。

## 功能描述

AF模块由AF信息统计及AF控制算法两部分组成。

下图为AF信息统计模块框图

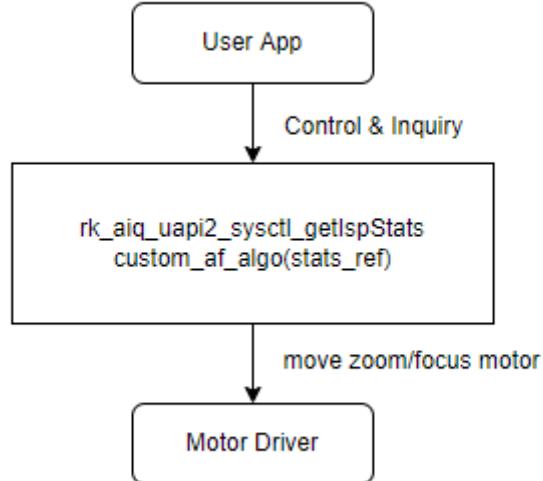


其中，ch0Wb0Out/ch1Wb0Out对应hdr模式下S/L帧，debayerIn为hdr模式下的合成帧，bayerTnrOut为tnr模块的输出，ynrOut为ynr模块的输出，vicapOut为vicap的输出。

windowA的输出主要包含15\*15的V1/V2/H1/H2 FV信息、亮度信息和高亮计数。

windowB的输出主要包含单独的V1/V2/H1/H2 FV信息、亮度信息和高亮计数。

## 开发用户AF算法



参考代码位置：sdk: external/camera\_engine\_rkaiq/rkisp\_demo/demo/af\_algo\_demo

## 功能级API参考

参考章节 [IMGPROC功能级API AF 相关](#)

## 模块级API参考

### **rk\_aiq\_user\_api2\_af\_SetAttrib**

#### 【描述】

设置对焦属性。

#### 【语法】

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

#### 【参数】

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

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_af.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_af\_GetAttrib**

### 【描述】

获取对焦属性。

### 【语法】

```
XCamReturn rk_aiq_user_api2_af_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_af_attrib_t *attr);
```

### 【参数】

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	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_af.h
- 库文件：librkaiq.so

## 模块级API数据类型

### RKAIQ\_AF\_MODE

#### 【说明】

定义对焦工作模式

#### 【定义】

```
typedef enum _RKAIQ_AF_MODE
{
    RKAIQ_AF_MODE_NOT_SET = -1,
    RKAIQ_AF_MODE_AUTO,
    RKAIQ_AF_MODE_MACRO,
    RKAIQ_AF_MODE_INFINITY,
    RKAIQ_AF_MODE_FIXED,
    RKAIQ_AF_MODE_EDOF,
    RKAIQ_AF_MODE_CONTINUOUS_VIDEO,
    RKAIQ_AF_MODE_CONTINUOUS_PICTURE,
    RKAIQ_AF_MODE_ONESHOT_AFTER_ZOOM,
} RKAIQ_AF_MODE;
```

#### 【成员】

成员名称	描述
RKAIQ_AF_MODE_NOT_SET	对焦模式未设置
RKAIQ_AF_MODE_AUTO	自动对焦模式
RKAIQ_AF_MODE_MACRO	微距对焦模式
RKAIQ_AF_MODE_INFINITY	远距对焦模式
RKAIQ_AF_MODE_FIXED	固定对焦模式
RKAIQ_AF_MODE_EDOF	景深对焦模式
RKAIQ_AF_MODE_CONTINUOUS_VIDEO	平滑持续对焦模式
RKAIQ_AF_MODE_CONTINUOUS_PICTURE	快速持续对焦模式
RKAIQ_AF_MODE_ONESHOT_AFTER_ZOOM	半自动对焦模式，set zoom position调用后，自动触发一次对焦

## RKAIQ\_AF\_HWVER

### 【说明】

定义对焦工作模式

### 【定义】

```
typedef enum _RKAIQ_AF_HWVER
{
    RKAIQ_AF_HW_V20 = 0,
    RKAIQ_AF_HW_V30,
    RKAIQ_AF_HW_V31,
    RKAIQ_AF_HW_V32_LITE,
    RKAIQ_AF_HW_V33,
    RKAIQ_AF_HW_VMAX
} RKAIQ_AF_HWVER;
```

### 【成员】

成员名称	描述
RKAIQ_AF_HW_V20	AF硬件版本 2.0
RKAIQ_AF_HW_V30	AF硬件版本 3.0
RKAIQ_AF_HW_V31	AF硬件版本 3.1
RKAIQ_AF_HW_V32_LITE	AF硬件版本 3.2
RKAIQ_AF_HW_V33	AF硬件版本 3.3

## manual\_afStats\_cfg

### 【说明】

定义AF信息统计工作模式

### 【定义】

```

typedef struct afStats_cfg_s {
    /* M4_GENERIC_DESC(
        M4_ALIAS(hw_afCfg_stats_en),
        M4_TYPE(bool),
        M4_DEFAULT(1),
        M4_HIDE_EX(1),
        M4_RO(0),
        M4_ORDER(0),
        M4_NOTES(Enable af statics.\nFreq of use: low)) */
    //reg: sw_rawaf_en
    bool hw_afCfg_stats_en;
    /* M4_GENERIC_DESC(
        M4_ALIAS(hw_afCfg_ldg_en),
        M4_TYPE(bool),
        M4_DEFAULT(0),
        M4_HIDE_EX(0),
        M4_RO(0),
        M4_ORDER(1),
        M4_GROUP_CTRL(ldg_en_group),
        M4_NOTES(Enable ldg function.\nFreq of use: high)) */
    //reg: sw_rawaf_ldg_en
    bool hw_afCfg_ldg_en;
    /* M4_GENERIC_DESC(
        M4_ALIAS(hw_afCfg_statsSrc_mode),
        M4_TYPE(enum),
        M4_ENUM_DEF(afStats_src_mode_t),
        M4_DEFAULT(afStats_chl0Wb0Out_mode),
        M4_HIDE_EX(0),
        M4_RO(0),
        M4_ORDER(2),
        M4_NOTES(Input source selection. Reference enum types.\nFreq of use: high)) */
    afStats_src_mode_t hw_afCfg_statsSrc_mode;
    /* M4_GENERIC_DESC(
        M4_ALIAS(hw_afCfg_statsBtnrOut_shift),
        M4_TYPE(u8),
        M4_SIZE_EX(1,1),
        M4_RANGE_EX(0,15),
        M4_DEFAULT(0),
        M4_HIDE_EX(0),
        M4_RO(0),
        M4_ORDER(3),
        M4_NOTES(Right shift bit number when use af_statsBtnrOut_mode.\nFreq of use: high)) */
    //reg: sw_rawaf_tnrin_shift
    uint8_t hw_afCfg_statsBtnrOut_shift;
    /* M4_GENERIC_DESC(
        M4_ALIAS(mainWin),
        M4_TYPE(struct),
        M4_UI_MODULE(normal_ui_style),
        M4_HIDE_EX(1),
        M4_RO(0),
        M4_ORDER(4),
        M4_NOTES(Roi of window A)) */
    afStats_mainWin_t mainWin;
    /* M4_GENERIC_DESC(
        M4_ALIAS(subWin),
        M4_TYPE(struct),
        M4_UI_MODULE(normal_ui_style),

```

```

M4_HIDE_EX(1),
M4_RO(0),
M4_ORDER(5),
M4_NOTES(Roi of window B)) */

afStats_subWin_t subWin;
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_afCfg_ds_mode),
    M4_TYPE(enum),
    M4_ENUM_DEF(afStats_ds_mode_t),
    M4_DEFAULT(afStats_ds_disable_mode),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(6),
    M4_NOTES(Down scale mode. Reference enum types.\nFreq of use: high)) */
afStats_ds_mode_t hw_afCfg_ds_mode;
/* M4_GENERIC_DESC(
    M4_ALIAS(gamma),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(7),
    M4_NOTES(Gamma setting)) */
afStats_gamma_t gamma;
/* M4_GENERIC_DESC(
    M4_ALIAS(preFilt),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(8),
    M4_NOTES(Gause setting)) */
afStats_gaus_t preFilt;
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_afCfg_hLumaCnt_thred),
    M4_TYPE(f32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0,1),
    M4_DEFAULT(0.8915),
    M4_HIDE_EX(0),
    M4_DIGIT_EX(10f10b),
    M4_RO(0),
    M4_ORDER(9),
    M4_NOTES(Hightlight threshold value.\nFreq of use: high)) */
//reg: sw_rawaf_highlit_thresh
float hw_afCfg_hLumaCnt_thred;
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_afCfg_hFiltLnBnd_mode),
    M4_TYPE(enum),
    M4_ENUM_DEF(afStats_hFiltLnBnd_mode_t),
    M4_DEFAULT(afStats_hFiltLnBnd_curLn_mode),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(10),
    M4_NOTES(Weather left border of horizontal filter use above line right border out.\nFreq of use:
low)) */
//reg: sw_rawaf_hiir_left_border_mode
afStats_hFiltLnBnd_mode_t hw_afCfg_hFiltLnBnd_mode;

```

```
/* M4_GENERIC_DESC(
    M4_ALIAS(hFilt1),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(11),
    M4_NOTES(Horizontal filter 1 setting)) */
afStats_hFilt_t hFilt1;
/* M4_GENERIC_DESC(
    M4_ALIAS(hFilt2),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(12),
    M4_NOTES(Horizontal filter 2 setting)) */
afStats_hFilt_t hFilt2;
/* M4_GENERIC_DESC(
    M4_ALIAS(hFiltCoring),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(13),
    M4_NOTES(Horizontal filter coring setting)) */
afStats_coring_t hFilt_coring;
/* M4_GENERIC_DESC(
    M4_ALIAS(hFiltLdg),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(14),
    M4_GROUP(ldg_en_group),
    M4_NOTES(Horizontal filter ldg setting)) */
afStats_hLdg_t hFilt_ldg;
/* M4_GENERIC_DESC(
    M4_ALIAS(vFilt1),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(15),
    M4_NOTES(Vertical filter 1 setting)) */
afStats_vFilt_t vFilt1;
/* M4_GENERIC_DESC(
    M4_ALIAS(vFilt2),
    M4_TYPE(struct),
    M4_UI_MODULE(normal_ui_style),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(16),
    M4_NOTES(Vertical filter 2 setting)) */
afStats_vFilt_t vFilt2;
/* M4_GENERIC_DESC(
    M4_ALIAS(vFiltCoring),
    M4_TYPE(struct),
```

```

M4_UI_MODULE(normal_ui_style),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(17),
M4_NOTES(Vertical filter coring setting)) */
afStats_coring_t vFilt_coring;
/* M4_GENERIC_DESC(
M4_ALIAS(vFiltLdg),
M4_TYPE(struct),
M4_UI_MODULE(normal_ui_style),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(18),
M4_GROUP(ldg_en_group),
M4_NOTES(Vertical filter ldg setting)) */
afStats_vLdg_t vFilt_ldg;
} afStats_cfg_t;

```

## 【成员】

成员名称	描述
hw_afCfg_stats_en	AF统计模块开关
hw_afCfg_ldg_en	LDG功能开关
hw_afCfg_statsSrc_mode	AF统计输入源的选择
hw_afCfg_statsBtnrOut_shift	统计输入源选择btnrOut时，每个像素右移的比特数
mainWin	统计主窗口的位置尺寸配置
subWin	统计独立窗口的位置尺寸配置
hw_afCfg_ds_mode	输入源的下采样配置
gamma	Gamma模块的配置
preFilt	预滤波模块的配置
hw_afCfg_hLumaCnt_thred	统计高亮像素个数时的阈值
hw_afCfg_hFiltLnBnd_mode	水平滤波器的边界处理模式
hFilt1	第一套水平滤波器的滤波器配置
hFilt2	第二套水平滤波器的滤波器配置
hFilt_coring	水平滤波器coring功能的相关配置
hFilt_ldg	水平滤波器ldg功能的相关配置
vFilt1	第一套垂直滤波器的滤波器配置
vFilt2	第二套垂直滤波器的滤波器配置
vFilt_coring	垂直滤波器coring功能的相关配置
vFilt_ldg	垂直滤波器ldg功能的相关配置

## rk\_aiq\_af\_attrib\_t

### 【说明】

对焦配置信息

### 【定义】

```
typedef struct rk_aiq_af_attrib_s {
    rk_aiq_uapi_sync_t sync;

    RKAIQ_AF_MODE AfMode;
    RKAIQ_AF_HWVER AfHwVer;

    bool contrast_af;
    bool laser_af;
    bool pdaf;

    int h_offs;
    int v_offs;
    unsigned int h_size;
    unsigned int v_size;

    short fixedModeDefCode;
    short macroModeDefCode;
    short infinityModeDefCode;

    union {
        rk_aiq_af_algo_meas_v20_t manual_meascfg;
        rk_aiq_af_algo_meas_v30_t manual_meascfg_v30;
        rk_aiq_af_algo_meas_v31_t manual_meascfg_v31;
        rk_aiq_af_algo_meas_v32_t manual_meascfg_v32;
        rk_aiq_af_algo_meas_v33_t manual_meascfg_v33;
        afStats_cfg_t manual_afStats_cfg;
    };
} rk_aiq_af_attrib_t;
```

### 【成员】

成员名称	描述
sync	同步异步API相关信息，参考rk_aiq_uapi_sync_t定义，参见“ <b>概述/API说明</b> ”章节
AfMode	对焦模式
contrast_af	使能反差对焦
laser_af	使能激光对焦
pdaf	使能相位对焦
h_offs	对焦窗口起始水平坐标
v_offs	对焦窗口起始垂直坐标
h_size	对焦窗口宽度
v_size	对焦窗口高度
fixedModeDefCode	固定对焦模式下对焦code值
macroModeDefCode	微距对焦模式下终止code值，对焦范围为0-该值
infinityModeDefCode	远距距对焦模式下起始code值，对焦范围为该值-MaxValue
manual_meascfg	AF2.0 自定义对焦统计信息配置
manual_meascfg_v30	AF3.0 自定义对焦统计信息配置
manual_meascfg_v31	AF3.1 自定义对焦统计信息配置
manual_meascfg_v32	AF3.2 自定义对焦统计信息配置
manual_meascfg_v33	AF3.3 自定义对焦统计信息配置，对应旧版头文件
manual_afStats_cfg	AF3.3 自定义对焦统计信息配置，对应新版头文件

## 其它说明

### VCM马达模组驱动验证

1) vcm驱动的起动电流、终止电流是否设置正确，

首先要从模组厂获取起动电流、终止电流的相关信息。

其次选取几个模组，确认这些信息是否正确，方法如下：

dts中将起始电流，终止电流设置为VCM可支持的最大范围。

对焦模式切换为手动模式，从64开始逐步调整vcm position，当lens开始移动，远焦物体(10米以上)清晰时，记录当前的position值，

起始电流为 $(vcm\_max\_mA - vcm\_min\_mA) * (64 - curPos) / 64$ 。

继续调整vcm位置，当近焦物体(10cm或20cm)清晰时，记录当前的position值，

终止电流为 $(vcm\_max\_mA - vcm\_min\_mA) * (64 - curPos) / 64$ 。

2) 不同方向移动vcm，最终停留位置是否稳定。

对焦模式切换为手动模式，选取一个position，从0移动到该位置和从64移动到该位置，比较两次的图像清晰程度是否一致，AF统计值是否接近。

3. 移动镜头所需要的时间是否正确。

## 电动马达模组驱动验证

1) 多次单步移动和一次多步移动最终停留位置是否相同

1) 首先选定一个zoom/focus马达位置，使图像达到最清晰的状态，记录当前马达位置，并抓取一张图像A；

2) 让zoom或focus马达后退一定的步数，然后单步移动zoom或focus马达，直到到达记录的zoom/focus马达位置，抓取一张图像B；

3) 让zoom或focus马达再次后退一定的步数，一次移动zoom或focus马达，到达记录的zoom/focus马达位置，抓取一张图像C；

4) 比较图像A、图像B和图像C的清晰程度是否一致，视野范围是否一致；

2) 马达随机移动最终停留位置是否相同

1) 首先选定一个zoom/focus马达位置，使图像达到最清晰的状态，记录当前马达位置，并抓取一张图像A；

2) 其次通过脚本让zoom或focus马达随机移动，移动400到500次后，回到最初的zoom/focus马达位置，抓取一张图像B；

3) 比较图像A与图像B，判断清晰程度是否一致，视野范围是否一致；

3) 马达同时移动最终停留位置是否相同

1) 首先选定一个zoom/focus马达位置，使图像达到最清晰的状态，记录当前马达位置，并抓取一张图像A；

2) 其次通过脚本让zoom和focus马达同时进行随机移动，移动400到500次后，回到最初的zoom/focus马达位置，抓取一张图像B；

3) 比较图像A与图像B，判断清晰程度是否一致，视野范围是否一致；

# IMGPROC

---

## 概述

imgproc 是指影响图像效果的模块。

## Merge

### 功能描述

Merge是将多帧图像合成为一帧的模块。

### 重要概念

- 在且仅在HDR模式下生效。

### 模块级API参考

**rk\_aiq\_user\_api2\_merge\_SetAttrib**

#### 【描述】

设置属性。

#### 【语法】

XCamReturn

```
rk_aiq_user_api2_merge_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, mge_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	属性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_amerge\_v12\_GetAttrib**

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

XCamReturn

```
rk_aiq_user_api2_merge_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, mge_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	属性	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_merge\_QueryStatus**

### 【描述】

获取当前生效参数及状态。

### 【语法】

XCamReturn

```
rk_aiq_user_api2_merge_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, mge_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	状态参数	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_merge.h 及 isp/rk\_aiq\_isp\_merge22.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## DRC

### 功能描述

DRC(动态范围压缩, High Dynamic Range Compression)，其作用是将高比特位的图像压缩到低比特位图像。

### 重要概念

- 在线性或者HDR模式下均可使用DRC。

## 功能级API参考

参考章节[IMGPROC功能级API](#)DRC相关。

## 模块级API参考

### `rk_aiq_user_api2_drc_SetAttrib`

### 【描述】

设置属性。

### 【语法】

### XCamReturn

```
rk_aiq_user_api2_drc_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, drc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### rk\_aiq\_user\_api2\_drc\_GetAttrib

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

### XCamReturn

```
rk_aiq_user_api2_drc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, drc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_drc\_QueryStatus**

#### 【描述】

获取生效参数及状态。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_drc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, drc_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	状态指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

### **rk\_aiq\_user\_api2\_drc\_SetStrength**

#### 【描述】

设置力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_drc_SetStrength(const rk_aiq_sys_ctx_t* sys_ctx, adrc_strength_t ctrl);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
ctrl	力度控制参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_drc\_SetStrength**

### 【描述】

获取当前力度，只适用于Auto模式。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_drc_GetStrength(const rk_aiq_sys_ctx_t* sys_ctx, adrc_strength_t ctrl);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
ctrl	力度控制参数	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_drc.h 及 isp/rk\_aiq\_isp\_drc40.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **adrc\_strength\_t**

### 【说明】

drc力度控制参数

### 【定义】

```

typedef struct {
    bool darkAreaBoostEn;
    unsigned int darkAreaBoostStrength;
    bool hdrStrengthEn;
    unsigned int hdrStrength;
} adrc_strenght_t;

```

### 【成员】

成员名称	描述
darkAreaBoostEn	暗区提升使能
darkAreaBoostStrength	暗区提升力度
hdrStrengthEn	高动态力度使能
hdrStrength	高动态力度

## Noise Removal

### 功能描述

图像噪声是指存在于图像数据中的不必要的或多余的干扰信息。图像去噪是减少数字图像中噪声的过程。

包括TNR（时域降噪）、YNR（Y域降噪）及CNR（UV域降噪）。

### 功能级API参考

参考章节[IMGPROC功能级API](#)去噪相关。

### 模块级API参考

#### rk\_aiq\_user\_api2\_btnr\_SetAttrib

##### 【描述】

设置降噪属性。

##### 【语法】

```

XCamReturn
rk_aiq_user_api2_btnr_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, btnr_api_attrib_t* attr);

```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_btnr\_GetAttrib**

##### 【描述】

获取默认属性参数或者最新设置参数。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_btnr_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, btnr_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_btnr\_QueryStatus**

##### 【描述】

获取当前参数及状态。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_btnr_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, btnr_status_t* status);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_btnr\_SetStrength**

##### 【描述】

设置力度，只适用于Auto模式。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_btnr_SetStrength(const rk_aiq_sys_ctx_t* sys_ctx, abtnr_strength_t *strg);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_btnr\_GetStrength**

### 【描述】

获取力度，只适用于Auto模式。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_btnr_GetStrength(const rk_aiq_sys_ctx_t* sys_ctx, abtnr_strength_t *strg);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

## rk\_aiq\_user\_api2\_ynr\_SetAttrib

### 【描述】

设置属性。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ynr_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ynr_api_attrib_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_ynr\_GetAttrib**

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ynr_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ynr_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_ynr\_QueryStatus**

#### 【描述】

获取当前参数及状态。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ynr_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, ynr_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_ynr\_SetStrength**

#### 【描述】

设置力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ynr_SetStrength(const rk_aiq_sys_ctx_t* sys_ctx, aynr_strength_t *strg);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### **rk\_aiq\_user\_api2\_ynr\_GetStrength**

#### 【描述】

获取力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_ynr_GetStrength(const rk_aiq_sys_ctx_t* sys_ctx, aynr_strength_t *strg);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cnr\_SetAttrib**

#### 【描述】

设置属性。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_cnr_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, cnr_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cnr\_GetAttrib**

### 【描述】

获取默认属性参数或者最新设置参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_cnr_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, cnr_api_attrib_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

## [rk\\_aiq\\_user\\_api2\\_cnr\\_QueryStatus](#)

### 【描述】

获取当前参数及状态。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_cnr_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, cnr_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cnr\_SetStrength**

#### 【描述】

设置力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_cnr_SetStrength(const rk_aiq_sys_ctx_t* sys_ctx, acnr_strength_t *strg);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cnr\_GetStrength**

#### 【描述】

获取力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_cnr_GetStrength(const rk_aiq_sys_ctx_t* sys_ctx, acnr_strength_t *strg);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### 模块级API数据类型

参数涉及头文件:

- TNR: algos/rk\_aiq\_api\_types\_btnr.h, algos/rk\_aiq\_api\_types\_btnr41.h, isp/rk\_aiq\_isp\_btnr41.h, 具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。
- YNR: algos/rk\_aiq\_api\_types\_ynr.h, isp/rk\_aiq\_isp\_ynr40.h, 具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。
- CNR: algos/rk\_aiq\_api\_types\_cnr.h, isp/rk\_aiq\_isp\_cnr32.h, 具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

### abtnr\_strength\_t

#### 【说明】

TNR力度控制参数

#### 【定义】

```
typedef struct {
    bool en;
    float percent;
} abtnr_strength_t;
```

#### 【成员】

成员名称	描述
en	力度控制使能
percent	力度百分比, 0~100

### aynr\_strength\_t

#### 【说明】

YNR力度控制参数

#### 【定义】

```
typedef struct {
    bool en;
    float percent;
} aynr_strength_t;
```

### 【成员】

成员名称	描述
en	力度控制使能
percent	力度百分比, 0~100

## acnr\_strength\_t

### 【说明】

CNR力度控制参数

### 【定义】

```
typedef struct {
    bool en;
    float percent;
} acnr_strength_t;
```

### 【成员】

成员名称	描述
en	力度控制使能
percent	力度百分比, 0~100

## BLC

### 功能描述

BLC(Black Levele Correction)为暗电平矫正模块。ISP39中包含 BLC0 和 BLC1 2个不同位置的BLC模块。其中HDR模式时只使用BLC0，线性模式时 BLC1 通常配合predgain和ob offset 功能使用。

### 模块级API参考

#### rk\_aiq\_user\_api2\_blc\_SetAttrib

##### 【描述】

设置属性。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_blc_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, blc_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_blc\_GetAttrib**

##### 【描述】

获取默认属性参数或者最新设置参数。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_blc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, blc_api_attrib_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_blc\_QueryStatus**

### 【描述】

获取当前参数及状态。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_blc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, blc_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

## 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_blc.h 及 isp/rk\_aiq\_isp\_blc30.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## Dehaze&Enhance

### 功能描述

Dehaze&Enhance包含2种模式：

- Dehaze 是通过动态的改变图象的对比度和亮度来实现的去雾增强。
- Enhance提升图像局部区域的对比度。

其中 Dehaze 与 Enhance功能互斥

### 功能级API参考

参考章节[IMGPROC功能级API](#)去雾及对比度。

## rk\_aiq\_user\_api2\_dehaze\_SetAttrib

### 【描述】

设置属性。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_dehaze_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dehaze_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_dehaze\_GetAttrib**

#### 【描述】

获取默认属性参数或者最新设置参数。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_dehaze_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dehaze_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

#### 【说明】

### **rk\_aiq\_user\_api2\_dehaze\_QueryStatus**

#### 【描述】

获取当前参数及状态。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_dehaze_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, dehaze_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

#### 【说明】

### **rk\_aiq\_user\_api2\_setDehazeEnhanceStrth**

#### 【描述】

设置力度参数，只适用于Auto模式。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_setDehazeEnhanceStrth(const rk_aiq_sys_ctx_t* sys_ctx, adehaze_strength_t ctrl);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_getDehazeEnhanceStrth**

##### 【描述】

获取力度参数，只适用于Auto模式。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_getDehazeEnhanceStrth(const rk_aiq_sys_ctx_t* sys_ctx, adehaze_strength_t* ctrl);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度参数	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_dehaze.h 及 isp/rk\_aiq\_isp\_dehaze23.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## HistEq

### 功能描述

HistEq(Histogram Equalization)直方图均衡化用于增强图像对比度。

### 功能级API参考

## 模块级API参考

### rk\_aiq\_user\_api2\_histeq\_SetAttrib

#### 【描述】

设置属性。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_histeq_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, histeq_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### rk\_aiq\_user\_api2\_histeq\_GetAttrib

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_histeq_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, histeq_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_histeq\_QueryStatus**

##### 【描述】

获取当前参数及状态。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_histeq_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, histeq_status_t* status);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_histeq.h 及 isp/rk\_aiq\_isp\_histeq23.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **CPROC**

### **功能描述**

CPROC(Color Processing) 提供基本的喜好色调节功能，通过对一定区间内的亮度、对比度、饱和度、色度的调节，达到对喜好色的调节，该模块作用于YUV域图像。

## 模块级API参考

### **rk\_aiq\_user\_api2\_cp\_SetAttrib**

#### 【描述】

设置cproc模块属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_cp_SetAttrib(const rk_aiq_sys_t* sys_ctx, cp_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	cproc模块属性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_cp\_GetAttrib**

#### 【描述】

获取cproc模块当前属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_cp_GetAttrib(const rk_aiq_sys_t* sys_ctx, cp_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	cproc模块属性	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_cp\_QueryStatus**

### 【描述】

查询cproc模块当前状态

### 【语法】

```
XCamReturn rk_aiq_user_api2_cp_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx,
                                              cp_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	IE模块状态指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **模块级API数据类型**

### **cp\_params\_static\_t**

#### 【说明】

定义cproc模块static属性参数

#### 【定义】

```
typedef struct {
    unsigned char brightness;
    unsigned char contrast;
    unsigned char saturation;
    unsigned char hue;
} cp_params_static_t;
```

### 【成员】

成员名称	描述
brightness	亮度, 范围: [0,255], 对应范围: [-128,127], 默认值 128
contrast	对比度, 范围: [0,255] , 对应倍数: [0, 1.992], 默认值为 128
saturation	饱和度, 范围: [0,255], 对应倍数: [0, 1.992], 默认值为 128
hue	色度, 范围: [0,255] , 对应度数[-90,87.188] , 默认值 128

## cp\_param\_auto\_t

### 【说明】

定义cproc模块自动参数

### 【定义】

```
typedef struct {
    cp_params_static_t sta;
} cp_param_auto_t;
```

### 【成员】

成员名称	描述
sta	cproc模块static属性

## cp\_param\_t

### 【说明】

定义cproc模块手动参数

### 【定义】

```
typedef struct {
    cp_params_static_t sta;
} cp_param_t;
```

### 【成员】

成员名称	描述
sta	cproc模块static属性

## cp\_api\_attrib\_t

## 【说明】

定义cproc模块的参数

## 【定义】

```
typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    cp_param_auto_t stAuto;
    cp_param_t stMan;
} cp_api_attrib_t;
```

## 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stAuto	自动cproc模块参数
stMan	手动cproc模块参数

## cp\_status\_t

### 【说明】

定义cproc模块的状态参数

### 【定义】

```
typedef struct cp_status_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    cp_param_t stMan;
} cp_status_t;
```

## 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	cproc模块当前生效参数

## IE

## 功能描述

IE(Image Effect) 提供图像的特殊效果设置，如黑白效果等。该模块作用于YUV域图像。

## 模块级API参考

### **rk\_aiq\_user\_api2\_ie\_SetAttrib**

#### 【描述】

设置IE模块属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_ie_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ie_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	IE模块属性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_ie\_GetAttrib**

#### 【描述】

获取IE模块当前属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_ie_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
                                         ie_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	IE模块属性	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_ie\_QueryStatus**

### 【描述】

查询IE模块当前状态

### 【语法】

```
XCamReturn rk_aiq_user_api2_ie_QueryStatus(const rk_aiq_sys_t* sys_ctx,
                                              ie_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	IE模块状态指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **模块级API数据类型**

### **ie\_params\_static\_t**

#### 【说明】

定义ie模块静态参数

#### 【定义】

```
typedef struct {
    unsigned char mode;
} ie_params_static_t;
```

#### 【成员】

成员名称	描述
mode	IE效果类型，0：无特殊效果，1：黑白效果，默认值0， <b>预留，暂不使用</b>

## ie\_param\_auto\_t

### 【说明】

定义自动ie模块参数

### 【定义】

```
typedef struct {
    ie_params_static_t sta;
} ie_param_auto_t;
```

成员名称	描述
sta	ie模块static参数

## ie\_param\_t

### 【说明】

定义手动ie模块参数

### 【定义】

```
typedef struct {
    ie_params_static_t sta;
} ie_param_t;
```

## ie\_api\_attrib\_t

### 【说明】

定义IE模块的参数

### 【定义】

```
typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    ie_param_auto_t stAuto;
    ie_param_t stMan;
} ie_api_attrib_t;
```

### 【成员】

成员名称	描述
opMode	工作模式
en	模块使能，模块使能即启用黑白效果
bypass	模块bypass，预留，暂不使用
stAuto	自动ie模块参数
stMan	手动ie模块参数

## ie\_status\_t

### 【说明】

定义ie模块的状态参数

### 【定义】

```
typedef struct ie_status_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    ie_param_t stMan;
} ie_status_t;
```

### 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	ie模块当前生效参数

## CSM

### 功能描述

CSM(Color Space Matrix) 可设置RGB到YUV转换时的参数。

### 模块级API参考

#### rk\_aiq\_user\_api2\_csm\_SetAttrib

##### 【描述】

设置CSM模块属性

##### 【语法】

```
XCamReturn rk_aiq_user_api2_csm_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, csm_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	CSM模块属性指针	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_csm\_GetAttrib**

#### 【描述】

获取CSM模块当前属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_csm_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, csm_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	CSM模块属性指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_csm\_QueryStatus**

#### 【描述】

查询CSM模块当前状态

#### 【语法】

```
XCamReturn rk_aiq_user_api2_csm_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, csm_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	CSM模块状态指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 模块级API数据类型

#### csm\_params\_static\_t

##### 【说明】

定义CSM模块静态参数

##### 【定义】

```
typedef struct {
    bool hw_csmT_full_range;
    uint8_t hw_csmT_y_offset;
    uint8_t hw_csmT_c_offset;
    float sw_csmT_coeff[9];
} csm_params_static_t;
```

#### 【成员】

成员名称	描述
hw_csmT_full_range	是否输出 full range， 默认值： true
hw_csmT_y_offset	亮度偏移值， 范围： [0,63]， 默认值： 0
hw_csmT_c_offset	色度偏移值， 范围： [0,255]， 默认值： 0
sw_csmT_coeff	RGB到YUV的3x3转换矩阵， 范围： [-2, 1.992]， 默认值： [ 0.299, 0.587, 0.114, -0.169, -0.331, 0.5, 0.5, -0.419, -0.081]

#### csm\_param\_t

### 【说明】

定义手动CSM模块参数

### 【定义】

```
typedef struct {
    csm_params_static_t
} csm_param_t;
```

## csm\_api\_attrib\_t

### 【说明】

定义CSM模块参数

### 【定义】

```
typedef struct csm_api_attrib_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    csm_param_t stMan;
} csm_api_attrib_t;
```

### 【成员】

成员名称	描述
opMode	工作模式，自动模式下使用BT601 Full标准
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	手动csm模块参数

## csm\_status\_t

### 【说明】

定义csm模块的状态参数

### 【定义】

```
typedef struct csm_status_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    csm_param_t stMan;
} csm_status_t;
```

### 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	csm模块当前生效参数

## Sharp

### 功能描述

Sharp 模块用于增强图像的清晰度，包括调节图像边缘的锐化属性和增强图像的细节和纹理。

### 功能级API参考

参考章节[IMGPROC功能级API](#)sharp相关。

### 模块级API参考

#### `rk_aiq_user_api2_sharp_SetAttrib`

##### 【描述】

设置属性。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_sharp_SetAttrib(const rk_aiq_sys_t* sys_ctx, sharp_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

##### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

##### 【说明】

#### `rk_aiq_user_api2_sharp_GetAttrib`

### 【描述】

获取默认属性参数或者最新设置参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_sharp_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, sharp_api_attrib_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

### 【说明】

## **rk\_aiq\_user\_api2\_sharp\_QueryStatus**

### 【描述】

获取当前参数及状态。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_sharp_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, sharp_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_sharp\_SetStrength**

#### 【描述】

设置力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_sharp_SetStrength(const rk_aiq_sys_ctx_t* sys_ctx, asharp_strength_t *strg);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_sharp\_GetStrength**

#### 【描述】

获取力度，只适用于Auto模式。

#### 【语法】

```
XCamReturn
```

```
rk_aiq_user_api2_sharp_GetStrength(const rk_aiq_sys_ctx_t* sys_ctx, asharp_strength_t *strg);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strg	力度	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_sharp.h, isp/rk\_aiq\_isp\_sharp40.h, 具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

#### asharp\_strength\_t

##### 【说明】

力度控制参数

##### 【定义】

```
typedef struct {
    bool en;
    float percent;
} asharp_strength_t;
```

##### 【成员】

成员名称	描述
en	力度控制使能
percent	力度百分比，0~100

## Gamma

### 功能描述

Gamma 模块对图像进行亮度空间非线性转换以适配输出设备。ISP39支持49点log域gamma曲线，其横坐标为：

```
int gamma_X_v11[49] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64, 80, 96, 112, 128,
160, 192, 224, 256, 320, 384, 448, 512, 640, 768, 896, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072,
3328, 3584, 3840, 4095};
```

### 功能级API参考

参考章节[IMGPROC功能级API](#)GAMMA相关。

### 模块级API参考

## **rk\_aiq\_user\_api2\_gamma\_SetAttrib**

### **【描述】**

设置属性。

### **【语法】**

```
XCamReturn
```

```
rk_aiq_user_api2_gamma_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gamma_api_attrib_t* attr);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **【说明】**

## **rk\_aiq\_user\_api2\_gamma\_GetAttrib**

### **【描述】**

获取默认属性参数或者最新设置参数。

### **【语法】**

```
XCamReturn
```

```
rk_aiq_user_api2_gamma_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gamma_api_attrib_t* attr);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_gamma\_QueryStatus**

##### 【描述】

获取当前参数及状态。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_gamma_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, gamma_status_t* status);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_gamma.h, isp/rk\_aiq\_isp\_gamma21.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **CCM**

### **功能描述**

CCM (Color Correction Matrix) 模块对图像进行颜色校正处理。

## 模块级API参考

### rk\_aiq\_user\_api2\_ccm\_SetAttrib

#### 【描述】

设置CCM属性。

#### 【语法】

```
XCamReturn rk_aiq_user_api2_ccm_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ccm_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	CCM的属性参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【示例】

- 参考 sample\_accm\_module.c 中 "sample\_ccm\_test" 设置CCM属性。

### rk\_aiq\_user\_api2\_ccm\_GetAttrib

#### 【描述】

获取CCM属性。

#### 【语法】

```
XCamReturn rk_aiq_user_api2_ccm_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ccm_api_attrib_t* attr);
```

#### 【参数】

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

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_accm\_module.c 中 "sample\_ccm\_test" 获取CCM属性。

## **rk\_aiq\_user\_api2\_ccm\_QueryStatus**

### 【描述】

查询CCM模块当前状态。

### 【语法】

```
XCamReturn rk_aiq_user_api2_ccm_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, ccm_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	CCM模块状态指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_accm\_module.c 中 "sample\_query\_ccm\_status" 查询CCM模块状态。

## **rk\_aiq\_user\_api2\_ccm\_SetCalib**

### 【描述】

设置CCM模块Calib。

### 【语法】

```
XCamReturn rk_aiq_user_api2_ccm_SetCalib(const rk_aiq_sys_ctx_t* sys_ctx, accm_ccmCalib_t* calib);
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	CCM模块Calib指针	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_ccm.h
- 库文件：librkaiq.so

## 【示例】

- 参考 sample\_accm\_module.c 中 "sample\_ccm\_setCalib\_test" 查询CCM模块状态。

### **rk\_aiq\_user\_api2\_ccm\_GetCalib**

## 【描述】

获取CCM模块Calib。

## 【语法】

```
XCamReturn rk_aiq_user_api2_ccm_GetCalib(const rk_aiq_sys_ctx_t* sys_ctx, accm_ccmCalib_t* calib);
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	CCM模块Calib指针	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【示例】

- 参考 sample\_accm\_module.c 中 "sample\_ccm\_setCalib\_test" 查询CCM模块状态。

## 模块级API数据类型

### ccm\_ccmAlpha\_yFac\_t

#### 【说明】

定义CCM模块局部亮度相关alpha调整曲线参数。

#### 【定义】

```
typedef struct {
    uint8_t hw_ccmT_facMax_minThred;
    uint8_t hw_ccmT_facMax_maxThred;
    uint16_t hw_ccmT_loY2Alpha_fac0[17];
    uint16_t hw_ccmT_hiY2Alpha_fac0[17];
} ccm_ccmAlpha_satFac_t;
```

#### 【成员】

成员名称	描述
hw_ccmT_facMax_minThred	当y <= 2^hw_ccmT_facMax_minThred时，认定为暗区，使用hw_ccmT_loY2Alpha_fac0曲线； 取值范围：[0, 11]
hw_ccmT_facMax_maxThred	当y >= 4095 - 2^hw_ccmT_facMax_maxThred时，认定为高亮区，使用hw_ccmT_hiY2Alpha_fac0曲线； 取值范围：[0, 11]
hw_ccmT_loY2Alpha_fac0	暗区调整曲线纵坐标，对应横坐标为y，递增曲线； 取值范围：[0, 1024]，
hw_ccmT_hiY2Alpha_fac0	曲线纵坐标，对应横坐标为4095-y，递增曲线； 取值范围：[0, 1024]

### ccm\_ccmAlpha\_satFac\_t

#### 【说明】

定义CCM模块局部饱和度相关alpha调整曲线参数。

#### 【定义】

```
typedef struct {
    uint8_t hw_ccmT_satIdx_maxLimit;
    uint8_t hw_ccmT_facMax_thred;
    float hw_ccmT_satIdx_scale;
    uint16_t hw_ccmT_sat2Alpha_fac1[17];
} ccm_ccmAlpha_satFac_t;
```

#### 【成员】

成员名称	描述
hw_ccmT_satIdx_maxLimit	当color diff <= -hw_ccmT_satIdx_maxLimit时，对应像素点的sat2Alpha_fac1 = hw_ccmT_sat2Alpha_fac1[16]; 取值范围：[0, 255]
hw_ccmT_facMax_thred	当color diff >= hw_ccmT_facMax_thred时，对应像素点的sat2Alpha_fac1 = hw_ccmT_sat2Alpha_fac1[0]; 取值范围：[0, 255]
hw_ccmT_satIdx_scale	曲线横坐标拉伸系数。； 取值范围：[0, 63]，越大对应的sat2Alpha_fac1越小，饱和度降低
hw_ccmT_sat2Alpha_fac1	曲线纵坐标，递减曲线； 取值范围：[0, 1024]

## ccm\_enhance\_t

### 【说明】

定义CCM模块enhance参数。

### 【定义】

```
typedef struct {
    bool hw_ccmT_enhance_en;
    float hw_ccmT_enhanceRat_maxLimit;
} ccm_enhance_t;
```

### 【成员】

成员名称	描述
hw_ccmT_enhance_en	是否启用颜色增强功能，取值范围：TRUE或FALSE
hw_ccmT_enhanceRat_maxLimit	动态范围的压缩比最大值限制，大于1，增强饱和度，小于1，降低饱和度； 取值范围：[0, 15.9986]

## ccm\_matrix\_t

### 【说明】

定义CCM模块标定矩阵参数。

### 【定义】

```
typedef struct {
    float hw_ccmC_matrix_coeff[9];
    float hw_ccmC_matrix_offset[3];
} ccm_matrix_t;
```

### 【成员】

成员名称	描述
hw_ccmC_matrix_coeff	色彩校正矩阵； 取值范围：[-8, 7.992]
hw_ccmC_matrix_offset	R\G\B分量偏移； 取值范围：[-2048, 2047]

## ccm\_param\_static\_t

### 【说明】

定义CCM模块static属性。

### 【定义】

```
typedef struct {
    uint16_t hw_ccmCfg_rgb2y_coeff[3];
} ccm_param_static_t;
```

### 【成员】

成员名称	描述
hw_ccmCfg_rgb2y_coeff	RGB到Y的转换系数，取值范围：[0, 128]

## ccm\_param\_dyn\_t

### 【说明】

定义手动CCM模块dynamic属性参数。

### 【定义】

```
typedef struct {
    ccm_enhance_t enhance;
    ccm_ccmAlpha_yFac_t ccmAlpha_yFac;
    ccm_ccmAlpha_satFac_t ccmAlpha_satFac;
    ccm_matrix_t ccMatrix;
} ccm_param_dyn_t;
```

### 【成员】

成员名称	描述
enhance	CCM模块enhance属性
ccm_ccmAlpha_yFac_t	CCM模块局部亮度相关alpha调整曲线属性
ccm_ccmAlpha_satFac_t	CCM模块局部饱和度相关alpha调整曲线属性
ccMatrix	CCM模块CCM矩阵属性

## ccm\_param\_t

### 【说明】

定义手动CCM模块属性参数。

## 【定义】

```
typedef struct {
    ccm_param_static_t sta;
    ccm_param_dyn_t dyn;
} ccm_param_t;
```

## 【成员】

成员名称	描述
sta	手动CCM模块static属性
dyn	手动CCM模块dynamic属性

## accm\_gain2SatCurve\_t

### 【说明】

定义自动CCM模块各个光源增益-饱和度曲线配置参数。

### 【定义】

```
typedef struct {
    float sw_ccmT_isoldx_val[4];
    float sw_ccmT_glbSat_val[4];
} accm_gain2SatCurve_t;
```

## 【成员】

成员名称	描述
sw_ccmT_isoldx_val	增益, 取值范围: [1, 4096]
sw_ccmT_glbSat_val	饱和度, 取值范围: [0, 200]

## accm\_param\_illuLink\_t

### 【说明】

定义自动CCM光源相关属性参数。

### 【定义】

```
typedef struct {
    char sw_ccmC_illu_name[8];
    float sw_ccmC_wbGainR_val;
    float sw_ccmC_wbGainB_val;
    accm_gain2SatCurve_t gain2SatCurve;
} accm_param_illuLink_t;
```

## 【成员】

成员名称	描述
sw_ccmC_illu_name	光源名
sw_ccmC_wbGainR_val	该光源对应的Rgain
sw_ccmC_wbGainB_val	该光源对应的Bgain
gain2SatCurve	增益-饱和度曲线配置

## accm\_param\_isoLink\_t

### 【说明】

定义自动CCM iso相关属性参数。

### 【定义】

```
typedef struct {
    float sw_ccmT_glbCcm_scale;
    ccm_enhance_t enhance;
    ccm_ccmAlpha_yFac_t ccmAlpha_yFac;
    ccm_ccmAlpha_satFac_t ccmAlpha_satFac;
} accm_param_isoLink_t;
```

### 【成员】

成员名称	描述
sw_ccmT_glbCcm_scale	全局饱和度调整系数，取值范围: [0, 1]
enhance	CCM模块enhance属性
ccm_ccmAlpha_yFac_t	CCM模块局部亮度相关alpha调整曲线属性
ccm_ccmAlpha_satFac_t	CCM模块局部饱和度相关alpha调整曲线属性

## accm\_param\_dyn\_t

### 【说明】

定义自动CCM模块dynamic属性

### 【定义】

```
typedef struct {
    accm_param_illuLink_t illuLink[9];
    uint8_t sw_ccmT_illuLink_len;
    accm_param_isoLink_t isoLink[13];
} accm_param_dyn_t;
```

### 【成员】

成员名称	描述
illuLink	自动CCM光源相关属性
sw_ccmT_illuLink_len	自动CCM光源相关属性实际个数, 取值范围: [0, 9]
isoLink	自动CCM iso相关属性

### accm\_param\_static\_t

#### 【说明】

定义自动CCM模块static属性参数。

#### 【定义】

```
typedef struct {
    bool sw_ccmT_damp_en;
    ccm_param_static_t ccmCfg;
} accm_param_static_t;
```

#### 【成员】

成员名称	描述
sw_ccmT_damp_en	自动CCM模块平滑使能
ccmCfg	CCM模块static属性参数

### ccm\_param\_auto\_t

#### 【说明】

定义自动CCM模块属性

#### 【定义】

```
typedef struct {
    accm_param_static_t sta;
    accm_param_dyn_t dyn;
} ccm_param_auto_t;
```

#### 【成员】

成员名称	描述
sta	自动CCM模块static属性
dyn	自动CCM模块dynamic属性

### ccm\_api\_attrib\_t

#### 【说明】

定义CCM模块属性

#### 【定义】

```

typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    ccm_param_auto_t stAuto;
    ccm_param_t stMan;
} accm_api_attrib_t;

```

### 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stAuto	自动CCM模块参数
stMan	手动CCM模块参数

### accm\_matrixAll\_t

#### 【说明】

定义CCM模块各个光源标定参数属性。

#### 【定义】

```

typedef struct {
    char sw_ccmC_illu_name[8];
    float sw_ccmC_ccmSat_val;
    ccm_matrix_t ccMatrix;
} accm_matrixAll_t;

```

### 【成员】

成员名称	描述
sw_ccmC_illu_name	光源名
sw_ccmC_ccmSat_val	饱和度, 取值范围: [0, 200]
ccMatrix	标定矩阵

### accm\_ccmCalib\_t

#### 【说明】

定义CCM模块Calib属性。

#### 【定义】

```

typedef struct {
    accm_matrixAll_t matrixAll[36];
    uint8_t sw_ccmC_matrixAll_len;
} accm_ccmCalib_t;

```

## 【成员】

成员名称	描述
matrixAll	CCM标定矩阵集合
sw_ccmC_matrixAll_len	实际标定矩阵个数, 取值范围: [0, 36]

## accm\_status\_t

### 【说明】

定义自动CCM模块状态参数

### 【定义】

```
typedef struct {
    char sw_ccmC_illuUsed_name[8];
    float sw_ccmC_ccmSat_val;
    float sw_ccmT_glbCcm_scale;
} accm_status_t;
```

## 【成员】

成员名称	描述
sw_ccmC_illuUsed_name	自动模式下CCM模块当前参数对应的光源
sw_ccmC_ccmSat_val	自动模式下CCM模块当前参数对应的饱和度
sw_ccmT_glbCcm_scale	自动模式下CCM模块当前参数对应的global scale

## ccm\_status\_t

### 【说明】

定义CCM模块状态参数

### 【定义】

```
typedef struct ccm_status_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    ccm_param_t stMan;
    accm_status_t accmStatus;
} ccm_status_t;
```

## 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	CCM模块当前生效参数
accmStatus	自动CCM模块状态参数

## LDC

### 功能描述

LDC模块包含两个子模块，LDCH和LDCV，分别对x方向和y方向进行图像小畸变校正。不适用鱼眼镜头畸变矫正。

如图为LDCH畸变校正前后的图像效果，该模块只对x方向小畸变进行校正。



#### 【LDC矫正表更新方式】

- IQ工具标定出镜头畸变系数，通过iq json或者API的方式传入到AIQ。AIQ内部根据畸变系数，自动生成矫正表应用到LDC硬件。
- 将矫正表数据保存成文件，通过iq json或者API的方式给AIQ传入文件路径及文件名。AIQ从文件读取矫正表，应用到LDC硬件。
- 将矫正表存到buffer，通过 `rk_aiq_user_api2_ldc_SetManualAttrib` 给AIQ传入buffer地址及大小。AIQ从buffer读取矫正表，应用到LDC硬件。

#### 【注意】

- LDC模块只支持两种模式，一种是只开LDCH，另一种是LDCH+LDCV都开，暂不支持LDCV的单独开关。
- 硬件上，数据流通路rkisp\_mainpath/rkisp\_selfpath和rkisp\_ldcpath互斥。要打开LDCV模块，应用也必须切换获取图像的video节点，即需要先关闭MP和SP节点，再开启LDC节点，从LDC节点获取图像数据。

mainpath	selfpath	ldcpath
关	关	开
开	开	关
开	关	关
关	开	关

### 功能级API参考

#### `rk_aiq_uapi2_setLdchEn`

##### 【描述】

LDCH开关。

##### 【语法】

```
XCamReturn rk_aiq_uapi2_setLdchEn(const rk_aiq_sys_ctx_t* ctx, bool en)
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
en	模块开关	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

#### 【示例】

源码路径: rkisp\_demo/demo/sample/sample\_aldc\_module.c

## rk\_aiq\_uapi2\_setLdchCorrectLevel

#### 【描述】

LDCH校正强度调节。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setLdchCorrectLevel(const rk_aiq_sys_ctx_t* ctx, int correctLevel)
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
correctLevel	校正强度，取值范围：[0~255]	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

#### 【示例】

源码路径: rkisp\_demo/demo/sample/sample\_aldc\_module.c

### **rk\_aiq\_uapi2\_setLdchLdcvEn**

#### 【描述】

LDCH和LDCV同时开启的模式，功能开关。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setLdchLdcvEn(const rk_aiq_sys_ctx_t* ctx, bool en)
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
en	模块开关	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

#### 【注意】

- LDCH + LDCV开启的模式下，不支持模块关，所以en传入false，配置无效。

### **rk\_aiq\_uapi2\_setLdchLdcvCorrectLevel**

#### 【描述】

LDCH和LDCV同时开启的模式，矫正强度调节。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setLdchLdcvCorrectLevel(const rk_aiq_sys_ctx_t* ctx, int correctLevel)
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
correctLevel	校正强度，取值范围：[0~255]	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 模块级API参考

#### **rk\_aiq\_user\_api2\_ldc\_SetAttrib**

##### 【描述】

设置LDC属性。

##### 【语法】

```
XCamReturn rk_aiq_user_api2_ldc_SetAttrib(const rk_aiq_sys_t* sys_ctx, ldc_api_attrib_t* attr)
```

##### 【参数】

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

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

##### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ldc.h
- 库文件：librkaiq.so

##### 【示例】

源码路径：rkisp\_demo/demo/sample/sample\_aldc\_module.c

##### 【注意】

- LDC的自动/手动模式在rk\_aiq\_uapi2\_sysctl\_prepare调用后就确定。所以调用此api修改模式的话，rk\_aiq\_user\_api2\_ldc\_SetAttrib需要在rk\_aiq\_uapi2\_sysctl\_prepare之前配置，否则模式切换，效果会有突变。
- opMode配置为手动模式的情况：必须配置tunning -> extMeshFile，并且放入mesh文件，否则LDC不开启。

## **rk\_aiq\_user\_api2\_ldc\_GetAttrib**

### **【描述】**

获取LDC当前配置的属性。

### **【语法】**

```
XCamReturn rk_aiq_user_api2_ldc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ldc_api_attrib_t* attr)
```

### **【参数】**

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

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_ldc.h
- 库文件：librkaiq.so

### **【示例】**

```
源码路径：rkisp_demo/demo/sample/sample_ldc_module.c
```

## **rk\_aiq\_user\_api2\_ldc\_QueryStatus**

### **【描述】**

查询LDC当前配置的属性。

### **【语法】**

```
XCamReturn rk_aiq_user_api2_ldc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, ldc_status_t* status)
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	LDC的参数属性	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ldc.h
- 库文件：librkaiq.so

### 【示例】

源码路径：rkisp\_demo/demo/sample/sample\_aldc\_module.c

## **rk\_aiq\_user\_api2\_ldc\_GetManualAttrib**

### 【描述】

获取应用传入校正表的模式下，当前的参数属性

### 【语法】

```
XCamReturn rk_aiq_user_api2_ldc_GetManualAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ldc_param_t* attr)
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	LDC外部表buffer更新矫正表模式属性	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_ldc.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_ldc\_SetManualAttrib**

## 【描述】

配置通过虚拟地址传入应用生成畸变校正表模式的接口属性

## 【语法】

```
XCamReturn rk_aiq_user_api2_ldc_SetManualAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ldc_param_t* attr)
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	LDC外部表buffer更新矫正表模式属性	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_ldc.h
- 库文件：librkaiq.so

## 【注意】

- LDC的自动/手动模式在rk\_aiq\_uapi2\_sysctl\_prepare调用后就确定。所以调用此api修改传入应用生成的校正表的话，rk\_aiq\_user\_api2\_ldc\_SetManualAttrib需要在rk\_aiq\_uapi2\_sysctl\_prepare之前配置，否则模式切换，效果会有突变。
- 调用此接口后，aiq内部会自动切成使用外部表，不需要再调用rk\_aiq\_user\_api2\_ldc\_SetAttrib接口。
- 应用传入表的模式下，模块开关也通过这个接口控制。

## 【参考】

源码路径：rkisp\_demo/demo/sample/sample\_aldc\_module.c

```
#include "uAPI2/rk_aiq_user_api2_ldc.h"

#define LDCH_MESH_1 "/tmp/ldch_mesh_1.bin"
#define LDCH_MESH_2 "/tmp/ldch_mesh_2.bin"
XCamReturn sample_aldc_SetManualAttrib(const rk_aiq_sys_ctx_t* ctx, bool en) {
    XCamReturn ret = XCAM_RETURN_NO_ERROR;
    if (ctx == NULL) {
        ret = XCAM_RETURN_ERROR_PARAM;
        RKAIQ_SAMPLE_CHECK_RET(ret, "param error!");
    }

    ldc_param_t attr;
    if (rk_aiq_user_api2_ldc_GetManualAttrib(ctx, &attr) < 0)
        ERR("Failed to get manual Attr from LDC");
```

```

if (en) {
    static bool flag = false;
    char filename[512] = "\0";
    if (flag)
        sprintf(filename, "%s", LDCH_MESH_1);
    else
        sprintf(filename, "%s", LDCH_MESH_2);
    flag = !flag;

    uint32_t size = 0;
    if (read_lut_from_file(filename, &attr.sta.ldchCfg.lutMapCfg.sw_ldcT_lutMapBuf_vaddr[0],
                           &size)) {
        attr.sta.ldchCfg.en = true;
        attr.sta.ldchCfg.lutMapCfg.sw_ldcT_lutMap_size = size;

        ret = rk_aiq_user_api2_ldc_SetManualAttrib(ctx, &attr);
        RKAIQ_SAMPLE_CHECK_RET(ret, "Ldc set manual attrib error!\n");
    }
} else {
    attr.sta.ldchCfg.en = false;
    ret = rk_aiq_user_api2_ldc_SetManualAttrib(ctx, &attr);
}

return ret;
}

```

## 模块级API数据类型

### ldc\_api\_attrib\_t

#### 【说明】

LDC属性配置

#### 【类型】

```

typedef struct ldc_api_attrib_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    ldc_param_auto_t tunning;
} ldc_api_attrib_t;

```

#### 【成员】

成员名称	描述
opMode	自动和手动模式配置
en	模块开关
bypass	此功能ldc模块不支持
tunning	详细的配置参数

### ldc\_param\_auto\_t

## 【说明】

LDC模块详细参数配置

## 【类型】

```
typedef struct ldc_param_auto_s {
#if defined(ISP_HW_V39)
    ldc_enMode_static_t enMode;
#endif
    ldc_autoGenMesh_static_t autoGenMesh;
    ldc_extMeshFile_static_t extMeshFile;
} ldc_param_auto_t;
```

## 【成员】

成员名称	描述
enMode	LDC_LDCH_EN：开启LDCH模块； LDC_LDCH_LDCV_EN：LDCH和LDCV模块同时开启；
autoGenMesh	IQ工具标定出镜头畸变系数，AIQ内部根据系数自动生成矫正表。使用此模式的话，ldc_api_attrib_t.opMode 必须配置为RK_AIQ_OP_MODE_AUTO。
extMeshFile	应用将矫正表保存都文件，AIQ从文件读取矫正表。使用此模式的话，ldc_api_attrib_t.opMode 必须配置为RK_AIQ_OP_MODE_MANUAL。

## 【注意】

- enMode必须配置正确，否则可能出现模式配置错误，模块没有开启的情况。

## ldc\_autoGenMesh\_static\_t

### 【说明】

RK畸变标定模型，AIQ内部自动生成矫正表模式下相关参数

### 【类型】

```
typedef struct ldc_autoGenMesh_static_s {
    ldc_lensDistorCoeff_static_t lensDistorCoeff;
    unsigned char sw_ldcT_correctStrg_val;
    unsigned char sw_ldcC_correctStrg_maxLimit;
    bool sw_ldcT_saveMaxFovX_bit;
} ldc_autoGenMesh_static_t;
```

## 【成员】

成员名称	描述
lensDistorCoeff	RK畸变标定模型标定出来的镜头畸变系数，IQ工具标定生成。
sw_ldcT_correctStrg_val	矫正强度配置，范围0~255。
sw_ldcC_correctStrg_maxLimit	LDC硬件可支持的最大矫正强度，IQ工具标定生成。
sw_ldcT_saveMaxFovX_bit	是否保存最大的fov，此配置开启，中心图像可能会压缩。矫正后图像fov满足需求的情况下，建议关闭。

## ldc\_lensDistorCoeff\_static\_t

### 【说明】

RK畸变标定模型标定出来的镜头畸变系数

### 【类型】

```
typedef struct ldc_lensDistorCoeff_static_s {
    double sw_ldcC_opticCenter_x;
    double sw_ldcC_opticCenter_y;
    double sw_ldcC_lensDistor_coeff[4];
} ldc_lensDistorCoeff_static_t;
```

### 【成员】

成员名称	描述
sw_ldcC_opticCenter_x	镜头光心在水平方向的位置。IQ工具标定生成。
sw_ldcC_opticCenter_y	镜头光心在垂直方向的位置。IQ工具标定生成。
sw_ldcC_lensDistor_coeff	RK畸变标定模型标定出来的镜头畸变系数。IQ工具标定生成。

### 【注意】

- 有些模组没有对LDC进行标定，直接打开LDC模块导致ISP卡住。可尝试将 sw\_ldcC\_opticCenter\_x/sw\_ldcC\_opticCenter\_y参数配置为图像中心，再打开camera。这种方式可以打开camera，但是矫正效果不佳，实际使用此模块需要用IQ工具进行标定。
- 使用此模式的话，ldc\_api\_attrib\_t.opMode 必须配置为RK\_AIQ\_OP\_MODE\_AUTO。

## ldc\_extMeshFile\_static\_t

### 【说明】

外部指定矫正表路径，AIQ读取矫正表应用到LDC方式的参数描述

### 【类型】

```
typedef struct ldc_extMeshFile_static_s {
    char sw_ldcT_extMeshFile_path[256];
    char sw_ldcT_ldchExtMeshFile_name[256];
#if defined(ISP_HW_V39)
    char sw_ldcT_ldcvExtMeshFile_name[256];
#endif
} ldc_extMeshFile_static_t;
```

## 【成员】

成员名称	描述
sw_ldcT_extMeshFile_path	矫正表在文件系统的目录，如 /data/
sw_ldcT_ldchExtMeshFile_name	LDCH模块矫正表文件名，如 ldch_mesh.bin
sw_ldcT_ldcvExtMeshFile_name	LDCV模块矫正表文件名，如 ldcv_mesh.bin

## 【注意】

- 使用此模式的话，ldc\_api\_attrib\_t.opMode 必须配置为RK\_AIQ\_OP\_MODE\_MANUAL。

## ldc\_param\_t

### 【说明】

通过buffer虚拟地址传入畸变矫正表方式的属性描述

### 【类型】

```
typedef struct {
    ldc_params_static_t sta;
} ldc_param_t;

typedef struct {
    ldc_ldch_params_cfg_t ldchCfg;
#ifndef ISP_HW_V39
    ldc_ldcv_params_cfg_t ldcvCfg;
#endif
} ldc_params_static_t;
```

## 【成员】

成员名称	描述
ldchCfg	LDCH模块参数配置
ldcvCfg	LDCV模块参数配置

## ldc\_ldch\_params\_cfg\_t

### 【说明】

通过buffer虚拟地址传入畸变矫正表方式的属性描述

### 【类型】

```
typedef struct ldc_ldch_params_cfg_s {
    bool en;
    ldc_lutMapCfg_t lutMapCfg;
} ldc_ldch_params_cfg_t;

typedef struct ldc_ldcv_params_cfg_s {
    bool en;
    ldc_lutMapCfg_t lutMapCfg;
} ldc_ldcv_params_cfg_t;
```

## 【成员】

成员名称	描述
en	模块开关
lutMapCfg	畸变矫正表地址、大小配置

## 【注意】

- LDC模块只支持两种模式，一种是只开LDCH，另一种是LDCH+LDCV都开，暂不支持LDCV的单独开关。

## ldc\_lutMapCfg\_t

### 【说明】

矫正表属性描述

### 【类型】

```
typedef struct ldc_lutMapCfg_s {
    uint32_t sw_ldcT_lutMap_size;
    void* sw_ldcT_lutMapBuf_vaddr[2];
    uint32_t sw_ldcT_lutMapBuf_fd[2];
} ldc_lutMapCfg_t;
```

## 【成员】

成员名称	描述
sw_ldcT_lutMap_size	矫正表大小
sw_ldcT_lutMapBuf_vaddr[2]	sw_ldcT_lutMapBuf_vaddr[0]: 应用存储矫正表buffer地址; sw_ldcT_lutMapBuf_vaddr[1]: ISP联合模式下，右摄畸变矫正表buffer地址；非联合模式下，不需要配置。
sw_ldcT_lutMapBuf_fd[2];	内部使用，不需要配置。

## DeBayer

### 功能描述

DeBayer(Demosic)完成将由 sensor 采集到的，带有 CFA 属性的图像通过插值算法还原成为具有完整像素信息的RGB图像。

该模块支持Bayer raw数据，包含 RGGB、BGGR、GRBG、GBRG 四种 pattern 模式。暂不支持 其他CFA数据，例如：RCCB, RGB-IR等CFA。

### 模块级API参考

#### rk\_aiq\_user\_api2\_dm\_SetAttrib

##### 【描述】

设置属性。

##### 【语法】

### XCamReturn

```
rk_aiq_user_api2_dm_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dm_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### rk\_aiq\_user\_api2\_dm\_GetAttrib

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

### XCamReturn

```
rk_aiq_user_api2_dm_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dm_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_dm\_QueryStatus**

#### 【描述】

获取当前参数及状态。

#### 【语法】

XCamReturn

```
rk_aiq_user_api2_dm_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, dm_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_dm.h, isp/rk\_aiq\_isp\_dm24.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **DPCC**

### **功能描述**

检测并消除图像中的坏点。建议详见《Rockchip\_Tuning\_Guide\_ISP39》文档中DPCC章节中“功能描述”。

### **模块级API参考**

#### **rk\_aiq\_user\_api2\_dpc\_SetAttrib**

#### 【描述】

设置属性。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_dpc_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dpc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### **rk\_aiq\_user\_api2\_dpc\_GetAttrib**

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_dpc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, dpc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### rk\_aiq\_user\_api2\_dpc\_QueryStatus

#### 【描述】

获取当前参数及状态。

#### 【语法】

XCamReturn

```
rk_aiq_user_api2_dpc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, dpc_status_t* status);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

#### 【说明】

### 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_dpc.h, isp/rk\_aiq\_isp\_dpc21.h, 具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## LSC

### 功能描述

镜头阴影校正（Lens Shading Correction）是为了解决由于lens的光学特性，由于镜头对于光学折射不均匀导致的镜头周围出现阴影的情况。

### 模块级API参考

#### rk\_aiq\_user\_api2\_lsc\_SetAttrib

#### 【描述】

设置LSC模块属性。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_lsc_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lsc_api_attrib_t attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	LSC模块属性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_lsc\_GetAttrib**

#### 【描述】

获取LSC模块属性。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_lsc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lsc_api_attrib_t *attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	LSC模块属性	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_lsc\_QueryStatus**

### 【描述】

查询LSC模块状态属性。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_lsc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, lsc_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	LSC模块状态指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## rk\_aiq\_user\_api2\_lsc\_SetCalib

### 【描述】

设置LSC模块Calib参数。

### 【语法】

```
XCamReturn  
rk_aiq_user_api2_lsc_SetCalib(const rk_aiq_sys_ctx_t* sys_ctx, alsc_lscCalib_t* calib);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	LSC模块Calib指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## **rk\_aiq\_user\_api2\_lsc\_GetCalib**

### **【描述】**

获取LSC模块Calib参数。

### **【语法】**

XCamReturn

```
rk_aiq_user_api2_lsc_GetCalib(const rk_aiq_sys_ctx_t* sys_ctx, alsc_lscCalib_t* calib);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	LSC模块Calib指针	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

## **数据类型**

### **lsc\_meshGrid\_t**

#### **【说明】**

定义LSC模块采样点坐标

#### **【定义】**

```
typedef enum lsc_meshGrid_s {
    float posX_f[17];
    float posY_f[17];
} lsc_meshGrid_t;
```

#### **【成员】**

成员名称	描述
posX_f	LSC模块采样点水平方向归一化坐标, 取值范围: [0, 1], posX_f[0] = 0, posX_f[16] = 1
posY_f	LSC模块采样点垂直方向归一化坐标, 取值范围: [0, 1], posY_f[0] = 0, posY_f[16] = 1

## **lsc\_meshGrid\_mode\_t**

### **【说明】**

定义LSC模块下采样模式

### **【定义】**

```
typedef enum lsc_meshGrid_mode_e {
    lsc_usrConfig_mode = 0,
    lsc_equalSector_mode = 1,
    lsc_vendorDefault_mode,
} lsc_meshGrid_mode_t;
```

### **【成员】**

成员名称	描述
lsc_usrConfig_mode	LSC模块用户自定义采样模式, 选择该模式, 即可通过meshGrid配置自定义采样点坐标
lsc_equalSector_mode	LSC模块等距采样模式
lsc_vendorDefault_mode	LSC模块平台默认下采样模式, 默认使用等距采样

## **lsc\_param\_static\_t**

### **【说明】**

定义LSC模块static属性参数

### **【定义】**

```
typedef struct {
    lsc_meshGrid_mode_t sw_lscT_meshGrid_mode;
    lsc_meshGrid_t meshGrid;
} lsc_param_static_t;
```

### **【成员】**

成员名称	描述
sw_lscT_meshGrid_mode	LSC模块下采样模式
meshGrid	LSC模块用户自定义采样点坐标, sw_lscT_meshGrid_mode = lsc_usrConfig_mode, 启用该配置

## **lsc\_meshGain\_t**

### 【说明】

定义LSC模块四个通道校正表参数

### 【定义】

```
typedef struct {
    uint16_t hw_lscC_gainR_val[289];
    uint16_t hw_lscC_gainGr_val[289];
    uint16_t hw_lscC_gainB_val[289];
    uint16_t hw_lscC_gainGb_val[289];
} lsc_meshGain_t;
```

### 【成员】

成员名称	描述
hw_lscC_gainR_val	LSC模块R通道校正表，取值范围: [1024, 8191]，高3bit整数位低8bit小数位
hw_lscC_gainGr_val	LSC模块Gr通道校正表，取值范围: [1024, 8191]，高3bit整数位低8bit小数位
hw_lscC_gainB_val	LSC模块B通道校正表，取值范围: [1024, 8191]，高3bit整数位低8bit小数位
hw_lscC_gainGb_val	LSC模块Gb通道校正表，取值范围: [1024, 8191]，高3bit整数位低8bit小数位

## lsc\_param\_dyn\_t

### 【说明】

定义LSC模块dynamic属性参数

### 【定义】

```
typedef struct {
    lsc_meshGain_t meshGain;
} lsc_param_dyn_t;
```

### 【成员】

成员名称	描述
meshGain	LSC模块四个通道校正表

## lsc\_param\_t

### 【说明】

定义手动LSC模块属性

### 【定义】

```
typedef struct {
    lsc_param_static_t sta;
    lsc_param_dyn_t dyn;
} lsc_param_t;
```

## 【成员】

成员名称	描述
sta	手动LSC模块static属性参数
dyn	手动LSC模块dynamic属性参数

## alsc\_gain2VigCurve\_t

### 【说明】

定义自动LSC模块各光源的增益-强度曲线属性参数

### 【定义】

```
typedef struct {
    float sw_lscT_isoldx_val[4];
    float sw_lscT_vignetting_val[4];
} alscl_gain2VigCurve_t;
```

## 【成员】

成员名称	描述
sw_lscT_isoldx_val	增益，取值范围: [1, 4096]
sw_lscT_vignetting_val	强度，取值范围: [0, 100]

## alsc\_param\_illuLink\_t

### 【说明】

定义自动LSC模块光源相关属性参数

### 【定义】

```
typedef struct {
    char sw_lscC_illu_name[8];
    float sw_lscC_wbGainR_val;
    float sw_lscC_wbGainB_val;
    alscl_gain2VigCurve_t gain2VigCurve;
} alscl_param_illuLink_t;
```

## 【成员】

成员名称	描述
sw_lscC_illu_name	光源名
sw_lscC_wbGainR_val	该光源对应的Rgain
sw_lscC_wbGainB_val	该光源对应的Bgain
gain2VigCurve	该光源的增益-强度曲线属性

## alsc\_param\_dyn\_t

## 【说明】

定义自动LSC模块dynamic属性参数

## 【定义】

```
typedef struct {
    alscl_param_illuLink_t illuLink[14];
    uint8_t sw_lscT_illuLink_len;
} ahsv_param_dyn_t;
```

## 【成员】

成员名称	描述
illuLink	自动LSC模块光源相关属性集合
sw_lscT_illuLink_len	实际光源相关属性个数，取值范围: [0, 14]

## alsc\_param\_static\_t

### 【说明】

定义自动LSC模块static属性参数

### 【定义】

```
typedef struct {
    bool sw_lscT_damp_en;
    lsc_param_static_t lscCfg;
} alscl_param_static_t;
```

## 【成员】

成员名称	描述
sw_lscT_damp_en	自动LSC模块平滑使能
lscCfg	LSC模块static属性

## lsc\_param\_auto\_t

### 【说明】

定义自动LSC模块属性参数

### 【定义】

```
typedef struct {
    alscl_param_static_t sta;
    alscl_param_dyn_t dyn;
} lsc_param_auto_t;
```

## 【成员】

成员名称	描述
sta	自动LSC模块static属性参数
dyn	自动LSC模块dynamic属性参数

### **lsc\_api\_attrib\_t**

#### **【说明】**

定义LSC模块属性参数

#### **【定义】**

```
typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    lsc_param_auto_t stAuto;
    lsc_param_t stMan;
} lsc_api_attrib_t;
```

#### **【成员】**

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stAuto	自动LSC模块属性参数
stMan	手动LSC模块属性参数

### **alsc\_tableAll\_t**

#### **【说明】**

定义LSC模块各个光源Calib参数

#### **【定义】**

```
typedef struct {
    char sw_lscC_illu_name[8];
    float sw_lscC_vignetting_val;
    lsc_meshGain_t meshGain;
} alscl_tableAll_t;
```

#### **【成员】**

成员名称	描述
sw_lscC_illu_name	光源名
sw_lscC_vignetting_val	LSC模块校正表对应强度，标定取得
meshGain	LSC模块校正表参数，标定取得

## alsc\_lscCalib\_t

### 【说明】

定义LSC模块Calib参数

### 【定义】

```
typedef struct {
    alsc_tableAll_t tableAll[42];
    int sw_lscC_tblAll_len;
} alsclscCalib_t;
```

### 【成员】

成员名称	描述
tableAll	LSC模块各个光源Calib集合
sw_lscC_tblAll_len	LSC模块Calib实际个数，取值范围: [0, 42]

## alsc\_status\_t

### 【说明】

定义自动LSC模块状态参数

### 【定义】

```
typedef struct {
    char sw_lscC_illuUsed_name[8];
    float sw_lscC_vignetting_val;
} alsclscStatus_t;
```

### 【成员】

成员名称	描述
sw_lscC_illuUsed_name	光源名
sw_lscC_vignetting_val	自动LSC模块vignetting生效值

## lsc\_status\_t

### 【说明】

定义LSC模块状态参数

### 【定义】

```
typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    lsc_param_t stMan;
    alsclscStatus_t alsclscStatus;
} lsc_status_t;
```

### 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	LSC模块当前生效参数
alscStatus	自动LSC模块状态参数

## GIC

### 功能描述

GIC 模块用于矫正Gr与Gb两个通道的失衡，提高部分场景的图像质量。

### 模块级API参考

#### **rk\_aiq\_user\_api2\_gic\_SetAttrib**

##### 【描述】

设置属性。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_gic_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gic_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

##### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

##### 【说明】

#### **rk\_aiq\_user\_api2\_gic\_GetAttrib**

##### 【描述】

获取默认属性参数或者最新设置参数。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_gic_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gic_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_gic\_QueryStatus**

#### 【描述】

获取当前参数及状态。

## 【语法】

```
XCamReturn  
rk_aiq_user_api2_gic_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, gic_status_t* status);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

#### 【说明】

#### **rk\_aiq\_user\_api2\_gic\_SetAttrib**

##### 【描述】

设置属性。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_gic_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gic_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

##### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

#### 【说明】

#### **rk\_aiq\_user\_api2\_gic\_GetAttrib**

##### 【描述】

获取默认属性参数或者最新设置参数。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_gic_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, gic_api_attrib_t* attr);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

##### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **rk\_aiq\_user\_api2\_gic\_QueryStatus**

##### 【描述】

获取当前参数及状态。

##### 【语法】

```
XCamReturn  
rk_aiq_user_api2_gic_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, gic_status_t* status);
```

##### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

#### **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_gic.h, isp/rk\_aiq\_isp\_gic30.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **CGC**

### **功能描述**

CGC(Color Gamut Compression) 可设置色域压缩相关参数。

## 模块级API参考

### **rk\_aiq\_user\_api2\_cgc\_SetAttrib**

#### 【描述】

设置CGC模块属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_cgc_SetAttrib(const rk_aiq_sys_t* sys_ctx, cgc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	CGC模块属性指针	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **rk\_aiq\_user\_api2\_cgc\_GetAttrib**

#### 【描述】

获取CGC模块当前属性

#### 【语法】

```
XCamReturn rk_aiq_user_api2_cgc_GetAttrib(const rk_aiq_sys_t* sys_ctx, cgc_api_attrib_t* attr);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	CGC模块属性指针	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_cgc.h
- 库文件：librkaiq.so

## **rk\_aiq\_user\_api2\_cgc\_QueryStatus**

### 【描述】

查询CGC模块状态参数

### 【语法】

```
XCamReturn rk_aiq_user_api2_cgc_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, cgc_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	CGC模块状态指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## **模块级API数据类型**

### **cgc\_params\_static\_t**

#### 【说明】

定义CGC模块static属性

#### 【定义】

```
typedef struct {
    bool cgc_yuv_limit;
    bool cgc_ratio_en;
} cgc_params_static_t;
```

#### 【成员】

成员名称	描述
cgc_yuv_limit	CGC模块yuv limit range使能
cgc_ratio_en	CGC模块色域压缩使能，仅在yuv limit range下可配置

### cgc\_param\_auto\_t

#### 【说明】

定义自动CGC模块属性

#### 【定义】

```
typedef struct {
    cgc_params_static_t sta;
} cgc_param_auto_t;
```

#### 【成员】

成员名称	描述
sta	CGC模块static属性参数

### cgc\_param\_t

#### 【说明】

定义手动CGC模块属性

#### 【定义】

```
typedef struct {
    cgc_params_static_t sta;
} cgc_param_t;
```

#### 【成员】

成员名称	描述
sta	CGC模块static属性参数

### cgc\_api\_attrib\_t

#### 【说明】

定义CGC模块属性参数

#### 【定义】

```
typedef struct {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    cgc_param_auto_t stAuto;
    cgc_param_t stMan;
} lsc_api_attrib_t;
```

## 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stAuto	自动CGC模块属性参数
stMan	手动CGC模块属性参数

## cgc\_status\_t

### 【说明】

定义CGC模块状态参数

### 【定义】

```
typedef struct cgc_status_s {
    rk_aiq_op_mode_t opMode;
    bool en;
    bool bypass;
    cgc_param_t stMan;
} cgc_status_t;
```

## 【成员】

成员名称	描述
opMode	工作模式
en	模块使能
bypass	模块bypass，预留，暂不使用
stMan	CGC模块当前生效参数

## CAC

### 功能描述

CAC 模块用于矫正由于光波长折射率差异引起的物体边缘紫边（或者其他颜色）问题。

### 模块级API参考

#### rk\_aiq\_user\_api2\_cac\_SetAttrib

##### 【描述】

设置属性。

##### 【语法】

```
XCamReturn
rk_aiq_user_api2_cac_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, cac_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cac\_GetAttrib**

#### 【描述】

获取默认属性参数或者最新设置参数。

#### 【语法】

```
XCamReturn  
rk_aiq_user_api2_cac_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, cac_api_attrib_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
attr	属性结构体	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

## 【说明】

### **rk\_aiq\_user\_api2\_cac\_QueryStatus**

### 【描述】

获取当前参数及状态。

### 【语法】

```
XCamReturn rk_aiq_user_api2_cac_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, cac_status_t* status);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
status	当前参数及状态	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【说明】

## 模块级API数据类型

参数涉及头文件 algos/rk\_aiq\_api\_types\_cac.h, isp/rk\_aiq\_isp\_cac30.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## 3DLut

### 功能描述

3DLut 模块对图像进行HSV空间的颜色映射处理。

### 模块级API参考

#### **rk\_aiq\_user\_api2\_3dlut\_SetAttrib**

### 【描述】

设置3dlut属性。

### 【语法】

```
XCamReturn rk_aiq_user_api2_3dlut_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_api_attrib_t* attr);  
;
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	3dlut模块参数属性	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.c 中 "sample\_3dlut\_test" 等设置3dlut属性。

## **rk\_aiq\_user\_api2\_3dlut\_GetAttrib**

### 【描述】

获取3dlut属性。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_3dlut_GetAttrib(const rk_aiq_sys_t* sys_ctx, lut3d_api_attrib_t* attr);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
attr	3dlut模块参数属性	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.c 中 "sample\_3dlut\_test" 等获取3dlut属性。

## **rk\_aiq\_user\_api2\_3dlut\_QueryStatus**

### **【描述】**

查询3dlut模块状态。

### **【语法】**

```
XCamReturn rk_aiq_user_api2_3dlut_QueryStatus(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_status_t* status);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
status	3dlut模块状态属性	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### **【示例】**

- 参考 sample\_a3dlut\_module.c 中 "sample\_query\_3dlut\_status" 查询3dlut状态。

## **rk\_aiq\_user\_api2\_3dlut\_SetCalib**

### **【描述】**

设置3dlut模块Calib参数。

### **【语法】**

```
XCamReturn
rk_aiq_user_api2_3dlut_SetCalib(const rk_aiq_sys_ctx_t* sys_ctx, alut3d_lut3dCalib_t* calib);
```

### **【参数】**

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	3dlut模块Calib指针	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.c 中 "sample\_3dlut\_setCalib\_test" 设置3dlut Calib。

## **rk\_aiq\_user\_api2\_3dlut\_GetCalib**

### 【描述】

获取3dlut模块Calib参数。

### 【语法】

```
XCamReturn
rk_aiq_user_api2_3dlut_GetCalib(const rk_aiq_sys_ctx_t* sys_ctx, alut3d_lut3dCalib_t* calib);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
calib	3dlut模块Calib指针	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_3dlut.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.c 中 "sample\_3dlut\_setCalib\_test" 获取3dlut Calib。

## **模块级API数据类型**

参数涉及头文件 algos/rk\_aiq\_api\_types\_3dlut.h, isp/rk\_aiq\_isp\_3dlut20.h，具体描述参考文档《Rockchip\_Tuning\_Guide\_ISP39》。

## **IMGPROC功能级API**

## AE相关

### rk\_aiq\_uapi2\_setAeLock

#### 【描述】

设置ae曝光锁定功能。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setAeLock (const rk_aiq_sys_ctx_t* ctx, bool on);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
on	锁定使能	输入

#### 【返回值】

返回值	描述
0	成功
非0	

### rk\_aiq\_uapi2\_setExpMode

#### 【描述】

设置曝光模式，支持设置自动曝光和手动曝光。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	曝光模式	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】

- 曝光模式切为手动模式时增益和曝光时间都切换为手动模式。

- 曝光模式切为手动模式时的增益和曝光时间采用图像效果文件中定义的初始值。
- 曝光模式切为手动模式时需要设置曝光值，可以使用rk\_aiq\_uapi2\_setExpManualGain和rk\_aiq\_uapi2\_setExpManualTime接口进行设置，在切换模式前调用这两个API配置手动增益和曝光时长，切换模式后API配置的值会直接生效。

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getExpMode**

### 【描述】

获取曝光模式。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	曝光模式	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setExpTimeMode**

### 【描述】

设置曝光时长模式，支持设置自动曝光时长和手动曝光时长。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpTimeMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	曝光时长模式	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】

- 需要先调用API rk\_aiq\_uapi2\_setExpMode将AE配置为手动模式，才可以单独切换曝光时长的模式。

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getExpTimeMode**

### 【描述】

获取曝光时长模式。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpTimeMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	曝光时长模式	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setExpGainMode**

### 【描述】

设置增益模式，支持设置自动增益和手动增益。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpGainMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	增益模式	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】

- 需要先调用API rk\_aiq\_uapi2\_setExpMode将AE配置为手动模式，才可以单独切换增益的模式。

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getExpGainMode**

### 【描述】

获取增益模式。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpGainMode(const rk_aiq_sys_t* ctx, opMode_t *mode);
```

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	增益模式	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setExpManualGain**

### 【描述】

设置手动增益。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpManualGain(const rk_aiq_sys_ctx_t* ctx, float gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
gain	曝光增益	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_setExpManualTime

### 【描述】

设置手动曝光时长。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpManualTime(const rk_aiq_sys_ctx_t* ctx, float gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
time	曝光时长	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h

- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setExpGainRange**

### 【描述】

设置增益范围。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpGainRange(const rk_aiq_sys_ctx_t* ctx, paRange_t *gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
gain	曝光增益范围	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getExpGainRange**

### 【描述】

获取增益范围。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpGainRange(const rk_aiq_sys_ctx_t* ctx, paRange_t *gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
gain	曝光增益范围	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setExpTimeRange**

### 【描述】

设置曝光时间范围。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpTimeRange(const rk_aiq_sys_ctx_t* ctx, paRange_t *time);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
time	曝光时间范围	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】

- 该Range限制在自动模式和手动模式下均生效

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getExpTimeRange**

### 【描述】

获取曝光时间范围。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpTimeRange(const rk_aiq_sys_ctx_t* ctx, paRange_t *time);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
time	曝光时间范围	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setBLCMode**

### 【描述】

背光补偿开关、区域设置。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setBLCMode(const rk_aiq_sys_ctx_t* ctx, bool on, aeMeasAreaType_t areaType);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输入
areaType	补偿区域选择	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】

- 该接口仅在线性模式下可用。

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setBLCStrength**

### 【描述】

设置暗区提升强度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setBLCStrength(const rk_aiq_sys_ctx_t* ctx, int strength);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strength	提升强度，范围[1,100]	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】

- 该接口仅在线性模式下可用。

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setHLCMode**

### 【描述】

强光抑制开关。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setHLCMode(const rk_aiq_sys_ctx_t* ctx, bool on);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	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】

- 该接口仅在线性模式下可用。

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h

- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setHLCStrength**

### 【描述】

设置强光抑制强度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setHLCStrength(const rk_aiq_sys_ctx_t* ctx, int strength);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
strength	抑制强度, 范围[1,100]	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】

- 该接口仅在线性模式下可用。

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setAntiFlickerEn**

### 【描述】

设置抗工频闪烁开关。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setAntiFlickerEn(const rk_aiq_sys_ctx_t* ctx, bool on);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
on	功能开关参数	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getAntiFlickerEn**

#### 【描述】

设置抗工频闪烁开关。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getAntiFlickerEn(const rk_aiq_sys_ctx_t* ctx, bool* on);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
on	功能开关参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_setAntiFlickerMode**

#### 【描述】

设置抗闪模式。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setAntiFlickerMode(const rk_aiq_sys_ctx_t* ctx, antiFlickerMode_t mode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	抗闪模式 ANTIFLICKER_AUTO_MODE：自动抗闪模式 ANTIFLICKER_NORMAL_MODE：普通抗闪模式	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getAntiFlickerMode**

#### 【描述】

获取抗闪模式。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getAntiFlickerMode(const rk_aiq_sys_t* ctx, antiFlickerMode_t *mode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	抗闪模式 ANTIFLICKER_AUTO_MODE：自动抗闪模式 ANTIFLICKER_NORMAL_MODE：普通抗闪模式	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_setExpPwrLineFreqMode**

### 【描述】

设置抗闪频率。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setExpPwrLineFreqMode(const rk_aiq_sys_ctx_t* ctx, expPwrLineFreq_t freq);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
freq	抗闪频率	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_getExpPwrLineFreqMode

### 【描述】

获取抗闪频率。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getExpPwrLineFreqMode(const rk_aiq_sys_ctx_t* ctx, expPwrLineFreq_t *freq);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
freq	抗闪频率	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## AWB相关

### **rk\_aiq\_uapi2\_setWBMode**

#### 【描述】

设置白平衡工作模式。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setWBMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	白平衡工作模式	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】

- 若设置为手动模式，白平衡增益值为当前手动白平衡参数控制。若需要切换手动模式的同时设置特定增益值，可以使用rk\_aiq\_uapi2\_SetMWBGain接口。

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

#### 【示例】

- 参考sample\_awb\_module.cpp

### **rk\_aiq\_uapi2\_getWBMode**

#### 【描述】

获取白平衡工作模式。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getWBMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

#### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
mode	白平衡工作模式	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module.cpp

## rk\_aiq\_uapi2\_lockAWB

#### 【描述】

锁定当前白平衡参数。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_lockAWB(const rk_aiq_sys_ctx_t* ctx);
```

#### 【参数】

参数名称	描述	输入/输出
sys_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

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

#### 【示例】

- 参考sample\_awb\_module.cpp

## rk\_aiq\_uapi2\_unlockAWB

### 【描述】

解锁已被锁定的白平衡参数。

### 【语法】

```
XCamReturn rk_aiq_uapi2_unlockAWB(const rk_aiq_sys_ctx_t* ctx);
```

### 【参数】

参数名称	描述	输入/输出
sys_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

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

### 【示例】

- 参考sample\_awb\_module.cpp

## **rk\_aiq\_uapi2\_setMWBScene**

### 【注意事项】

不支持该API

## **rk\_aiq\_uapi2\_getMWBScene**

### 【注意事项】

不支持该API

## **rk\_aiq\_uapi2\_setMWBGain**

### 【描述】

设置手动白平衡增益系数。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setMWBGain(const rk_aiq_sys_ctx_t* ctx, rk_aiq_wb_gain_t *gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
gain	白平衡增益系数	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考sample\_awb\_module.cpp

## rk\_aiq\_uapi2\_getWBGain

### 【描述】

获取白平衡增益系数。手动白平衡和自动白平衡均用该函数

### 【语法】

```
XCamReturn rk_aiq_uapi2_getWBGain(const rk_aiq_sys_ctx_t* ctx, rk_aiq_wb_gain_t *gain);
```

### 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
gain	白平衡增益系数	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考sample\_awb\_module.cpp

## **rk\_aiq\_uapi2\_setWBCT**

### **【注意事项】**

不支持该API

## **rk\_aiq\_uapi2\_getWBCT**

### **【注意事项】**

不支持该API

## **rk\_aiq\_uapi2\_setAwbGainOffsetAttrib**

### **【注意事项】**

不支持该API

## **rk\_aiq\_uapi2\_getAwbGainOffsetAttrib**

### **【注意事项】**

不支持该API

## **AF相关**

### **rk\_aiq\_uapi2\_setFocusMode**

**【描述】** 配置对焦模式。

#### **【语法】**

```
XCamReturn rk_aiq_uapi2_setFocusMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

#### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	对焦模式	输入

#### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

#### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getFocusMode**

**【描述】** 获取当前对焦模式。

## 【语法】

```
XCamReturn rk_aiq_uapi2_getFocusMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	对焦模式	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setFocusWin**

**【描述】** 设置自动对焦窗口。

## 【语法】

```
XCamReturn rk_aiq_uapi2_setFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_t *rect);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
rect	对焦窗口，取值范围由sensor输入大小确定	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getFocusWin**

**【描述】** 设置自动对焦窗口。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_t *rect);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
rect	对焦窗口	输出

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

**【需求】**

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_lockFocus**

**【描述】** 锁定自动对焦，对焦暂停动作。

**【语法】**

```
XCamReturn rk_aiq_uapi2_lockFocus(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

**【需求】**

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_unlockFocus**

**【描述】** 解锁自动对焦，对焦继续动作。

**【语法】**

```
XCamReturn rk_aiq_uapi2_unlockFocus(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

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_oneshotFocus**

**【描述】** 触发单次对焦，对焦完成后，不会继续对焦。

**【语法】**

```
XCamReturn rk_aiq_uapi2_oneshotFocus(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

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_manualTrigerFocus**

**【描述】** 手动触发对焦，对焦完成后，继续监视画面是否模糊，如果画面模糊，再次执行对焦。

## 【语法】

```
XCamReturn rk_aiq_uapi2_manualTrigerFocus(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

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_trackingFocus**

**【描述】** 继续监视画面是否模糊，如果画面模糊，再次执行对焦，一般执行rk\_aiq\_uapi2\_oneshotFocus后使用。

## 【语法】

```
XCamReturn rk_aiq_uapi2_trackingFocus(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

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getSearchResult**

**【描述】** 获取对焦结果。

## 【语法】

```
XCamReturn rk_aiq_uapi2_getSearchResult(const rk_aiq_sys_ctx_t* ctx, rk_aiq_af_result_t* result);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
result	对焦结果，具体参考结构体rk_aiq_af_result_t的说明	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_getZoomRange

**【描述】** 获取变焦范围值，用于限制rk\_aiq\_uapi\_setOpZoomPosition输入的zoom code参数。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getZoomRange(const rk_aiq_sys_ctx_t* ctx, rk_aiq_af_zoomrange* range);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
range	zoom可移动范围，具体参考结构体rk_aiq_af_zoomrange	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_setOpZoomPosition

**【描述】** 设置zoom位置，修改变焦位置。

## 【语法】

```
XCamReturn rk_aiq_uapi2_setOpZoomPosition(const rk_aiq_sys_ctx_t* ctx, int pos);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pos	zoom position, 取值范围由rk_aiq_uapi2_getZoomRange确定	输入

## 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getOpZoomPosition**

**【描述】** 获取zoom位置。

## 【语法】

```
XCamReturn rk_aiq_uapi2_getOpZoomPosition(const rk_aiq_sys_ctx_t* ctx, int *pos);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
pos	zoom position	输出

## 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_endOpZoomChange**

**【描述】** 结束zoom设备位置的设置，在rk\_aiq\_uapi\_setOpZoomPosition之后调用，被调用后执行聚焦动作。

**【语法】**

```
XCamReturn rk_aiq_uapi2_endOpZoomChange(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

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getFocusRange**

**【描述】** 获取聚焦范围值，用于限制rk\_aiq\_uapi\_setFocusPosition输入的focus code参数。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getFocusRange(const rk_aiq_sys_ctx_t* ctx, rk_aiq_af_focusrange* range);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
range	focus可移动范围，具体参考结构体rk_aiq_af_focusrange	输出

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_setFocusPosition**

**【描述】** 设置手动对焦模式下的对焦位置。

**【语法】**

```
XCamReturn rk_aiq_uapi2_setFocusPosition(const rk_aiq_sys_ctx_t* ctx, unsigned short code);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
code	对焦code值，取值范围由rk_aiq_uapi2_getFocusRange确定	输入

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getFocusPosition**

**【描述】** 获取当前对焦位置。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getFocusPosition(const rk_aiq_sys_ctx_t* ctx, unsigned short *code);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
code	对焦code值	输出

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_startZoomCalib**

**【描述】** 执行电动马达模组校正。

**【语法】**

```
XCamReturn rk_aiq_uapi2_startZoomCalib(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

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_resetZoom**

**【描述】** 执行电动马达模组复位。

**【语法】**

```
XCamReturn rk_aiq_uapi2_resetZoom(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

**【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **去雾及对比度**

## **rk\_aiq\_uapi2\_setMDehazeStrth**

### **【描述】**

设置去雾力度。

### **【语法】**

```
XCamReturn rk_aiq_uapi2_setMDehazeStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，取值范围[1,100]，默认值为50，精度1。	输入

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getMDehazeStrth**

### **【描述】**

获取去雾力度。

### **【语法】**

```
XCamReturn rk_aiq_uapi2_getMDehazeStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int* level);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，取值范围[1,100]，默认值为50，精度1。	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setMEnhanceStrth**

### 【描述】

设置Enhance局部对比度力度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setMEnhanceStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，，取值范围[1,100]，默认值为50，精度1。	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getMEnhanceStrth**

### 【描述】

获取Enhance局部对比度力度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getMEnhanceStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，，取值范围[1,100]，默认值为50，精度1。	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_setMEnhanceChromeStrth**

#### 【描述】

设置Enhance局部对比度调整中的颜色饱和度调整力度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setMEnhanceChromeStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，，取值范围[1,100]，默认值为50，精度1。	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getMEnhanceChromeStrth**

#### 【描述】

获取设置Enhance局部对比度调整中的颜色饱和度调整力度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getMEnhanceChromeStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	等级，，取值范围[1,100]，默认值为50，精度1。	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## DRC相关

### **rk\_aiq\_uapi2\_setHDRStrth**

**【描述】** 设置动态范围。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setHDRStrth(const rk_aiq_sys_ctx_t* ctx, bool on, unsigned int level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输入
level	动态范围，取值范围0.0-100.0, 默认值50。	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getHDRStrth**

**【描述】** 获取动态范围。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getHDRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on, unsigned int *level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输出
level	动态范围, 取值范围0.0-100.0, 默认值50。	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_setDarkAreaBoostStrth**

**【描述】** 获取暗区增强力度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setDarkAreaBoostStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int level)
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	动态范围, 取值范围0.0-100.0, 默认值50。	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_getDarkAreaBoostStrth**

**【描述】** 获取暗区增强力度。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getDarkAreaBoostStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int* level)
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	动态范围, 取值范围0.0-100.0, 默认值50。	输出

**【返回值】**

返回值	描述
0	成功
非0	失败, 详见错误码表

**【需求】**

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## 去噪

### **rk\_aiq\_uapi2\_setNRMode**

**【注意事项】**

不支持该API

### **rk\_aiq\_uapi2\_getNRMode**

**【描述】** 获取当前去噪模式。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getNRMode(const rk_aiq_sys_ctx_t* ctx, opMode_t* mode);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	工作模式	输出

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_setANRStrth**

**【描述】** 设置NR子系统整体去噪力度等级。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setANRStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	去噪强度，取值范围0.0-100.0, 默认值50。	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】

- ISP39 NR子系统包含：时域降噪、空域亮度降噪等。

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getANRStrth**

**【描述】** 获取NR子系统整体去噪力度等级。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getANRStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int* level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	去噪强度，取值范围0.0-100.0, 默认值50。	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_setMSpaNRStrth**

**【描述】** 设置空域亮度去噪强度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setMSpaNRStrth(const rk_aiq_sys_ctx_t* ctx, bool on, unsigned int level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输入
level	去噪强度，取值范围0.0-100.0, 默认值50。	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getMSpaNRStrth**

**【描述】** 获取空域亮度去噪强度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getMSpaNRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on, unsigned int *level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输出
level	去噪强度，取值范围0.0-100.0, 默认值50。	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_setMTNRStrth**

**【描述】** 设置时域去噪强度。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setMTNRStrth(const rk_aiq_sys_ctx_t* ctx, bool on, unsigned int level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输入
level	去噪强度，取值范围0-100, 默认值50。	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getMTNRStrth**

**【描述】** 获取时域去噪强度。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getMTNRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on, unsigned int *level);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
on	开关	输出
level	去噪强度，取值范围0-100, 默认值50。	输出

**【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码表

**【需求】**

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **CCM相关**

### **rk\_aiq\_uapi2\_setCCMode**

**【注意事项】**

不支持该API

**【模块级API替代】**

```
XCamReturn rk_aiq_user_api2_ccm_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ccm_api_attrib_t* attr);
```

### **rk\_aiq\_uapi2\_getCCMode**

**【描述】**

获取CCM工作模式。

**【语法】**

```
XCamReturn rk_aiq_uapi2_getCCMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

**【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	工作模式; 取值范围: {OP_AUTO, OP_MANUAL}	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

#### 【示例】

- 参考 sample\_accm\_module.cpp 中 "sample\_get\_ccm\_mode" 获取工作模式。

### **rk\_aiq\_uapi2\_setMCCoef**

#### 【注意事项】

不支持该API

#### 【模块级API替代】

```
XCamReturn rk_aiq_user_api2_ccm_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, ccm_api_attrib_t* attr);
```

### **rk\_aiq\_uapi2\_getMCCoef**

#### 【描述】

获取CCM矩阵，包括色彩校正矩阵和R/G/B通道偏移。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getMCCoef(const rk_aiq_sys_ctx_t* ctx, rk_aiq_ccm_matrix_t *mccm);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mccm	CCM矩阵，包括色彩校正矩阵和R/G/B通道偏移	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_accm\_module.cpp 中 "sample\_get\_ccm\_manual\_matrix" 获取CCM矩阵。

## **rk\_aiq\_uapi2\_getACcmSat**

### 【描述】

获取自动模式下的CCM饱和度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getACcmSat(const rk_aiq_sys_ctx_t* ctx, float *finalsat);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
finalsat	饱和度，手动模式为0	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_accm\_module.cpp 中 "sample\_get\_accm\_sat" 获取自动模式下CCM饱和度。

## **rk\_aiq\_uapi2\_getACcmMatrixName**

### 【描述】

获取自动模式下CCM矩阵名。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getACcmMatrixName(const rk_aiq_sys_ctx_t* ctx, char **ccm_name);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
ccm_name	CCM矩阵名	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## 【示例】

- 参考 sample\_accm\_module.cpp 中 "sample\_get\_accm\_matrix\_name" 获取自动模式下CCM饱和度。

## 3DLUT相关

### rk\_aiq\_uapi2\_setLut3dMode

#### 【注意事项】

不支持该API

#### 【模块级API替代】

```
XCamReturn rk_aiq_user_api2_3dlut_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_api_attrib_t* attr);
```

### rk\_aiq\_uapi2\_getLut3dMode

#### 【描述】

获取3DLUT工作模式。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getLut3dMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

## 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	工作模式	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.cpp 中 "sample\_get\_a3dlut\_mode" 设置手动/自动模式。

## **rk\_aiq\_uapi2\_setM3dLut**

### 【注意事项】

不支持该API

### 【模块级API替代】

```
XCamReturn rk_aiq_user_api2_3dlut_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_api_attrib_t* attr);
```

## **rk\_aiq\_uapi2\_getM3dLut**

### 【描述】

获取3DLUT 3D查找表。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getM3dLut(const rk_aiq_sys_ctx_t* ctx, rk_aiq_lut3d_table_t *mlut);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mlut	3D查找表	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### 【示例】

- 参考 sample\_a3dlut\_module.cpp 中 "sample\_get\_a3dlut\_lut" 获取3DLUT 3D查找表。

## rk\_aiq\_uapi2\_getA3dLutStrth

### 【注意事项】

不支持该API

### 【模块级API替代】

```
XCamReturn rk_aiq_user_api2_3dlut_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_api_attrib_t* attr);
```

## rk\_aiq\_uapi2\_getA3dLutName

### 【注意事项】

不支持该API

### 【模块级API替代】

```
XCamReturn rk_aiq_user_api2_3dlut_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx, lut3d_api_attrib_t* attr);
```

## 对比度

## rk\_aiq\_uapi2\_setContrast

### 【描述】

设置对比度强度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setContrast(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_getContrast

### 【描述】

获取对比度强度。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getContrast(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## 亮度

### **rk\_aiq\_uapi2\_setBrightness**

### 【描述】

设置输出图像的全局亮度等级。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setBrightness(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级， 默认等级128	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getBrightness**

### 【描述】

获取输出图像的全局亮度等级。

### 【语法】

```
XCamReturn rk_aiq_uapi2_getBrightness(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **饱和度**

## **rk\_aiq\_uapi2\_setSaturation**

### 【描述】

设置输出图像的全局饱和度等级。

### 【语法】

```
XCamReturn rk_aiq_uapi2_setSaturation(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级， 默认等级128	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】

- 该亮度调整是ISP图像输出的后处理模块，不包含在图像效果调试范畴。

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **rk\_aiq\_uapi2\_getSaturation**

#### 【描述】

获取输出图像的全局饱和度等级。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getSaturation(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【注意】

- 该亮度调整是ISP图像输出的后处理模块，不包含在图像效果调试范畴。

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

### **色度**

### **rk\_aiq\_uapi2\_setHue**

#### 【描述】

设置输出图像的全局色度等级。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setHue(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级， 默认等级128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】

- 该色度调整是ISP图像输出的后处理模块， 不包含在图像效果调试范畴。

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## rk\_aiq\_uapi2\_getHue

#### 【描述】

获取输出图像的全局色度等级。

#### 【语法】

```
XCamReturn rk_aiq_uapi2_getHue(const rk_aiq_sys_ctx_t* ctx, unsigned int *level);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
level	强度等级	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败， 详见错误码表

#### 【注意】

- 该色度调整是ISP图像输出的后处理模块， 不包含在图像效果调试范畴。

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

# 色彩模式配置

## **rk\_aiq\_uapi2SetColorMode**

### 【描述】

设置颜色模式。

### 【语法】

```
XCamReturn rk_aiq_uapi2SetColorMode(const rk_aiq_sys_ctx_t* ctx, unsigned int mode);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	颜色模式： 0 NONE 1 BW 2 NEGATIVE 3 SEPIA 4 BW 5 NEGATIVE 6 SEPIA	输入

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## **rk\_aiq\_uapi2\_getColorMode**

### 【描述】

获取颜色模式

### 【语法】

```
XCamReturn rk_aiq_uapi2_getColorMode(const rk_aiq_sys_ctx_t* ctx, unsigned int *mode);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	模式: 0 NONE 1 BW 2 NEGATIVE 3 SEPIA 4 BW 5 NEGATIVE 6 SEPIA	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api2\_imgproc.h
- 库文件: librkaiq.so

## 灰度范围、色彩空间配置

### **rk\_aiq\_uapi2SetColorSpace**

#### 【描述】

设置灰度范围、色彩空间。

#### 【语法】

```
XCamReturn rk_aiq_uapi2SetColorSpace(const rk_aiq_sys_ctx_t* ctx, int Cspace);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
Cspace	色彩空间模式: 0 BT.601 FULL 1 BT.601 LIMIT 2 BT.709 FULL 3 BT.709 LIMIT 253 OTHER FULL 254 OTHER LIMIT 255 OTHER	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】

- OTHER FULL/OTHER LIMIT 指在色彩空间为非BT.601和BT.709时下的灰度范围配置

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getColorSpace**

### 【描述】

获取灰度范围、色彩空间

### 【语法】

```
XCamReturn rk_aiq_uapi2_getColorSpace(const rk_aiq_sys_ctx_t* ctx, int *Cspace);
```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
Cspace	色彩空间模式： 0 BT.601 FULL 1 BT.601 LIMIT 2 BT.709 FULL 3 BT.709 LIMIT 253 OTHER FULL 254 OTHER LIMIT 255 OTHER	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

### 【注意】

- OTHER FULL/OTHER LIMIT 指在色彩空间为非BT.601和BT.709时下的灰度范围配置

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **灰度模式配置**

## **rk\_aiq\_uapi2\_setGrayMode**

### **【描述】**

设置灰度模式。

### **【语法】**

```
XCamReturn rk_aiq_uapi2_setGrayMode(const rk_aiq_sys_ctx_t* ctx, rk_aiq_gray_mode_t mode);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
mode	模式选择	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】**

- 彩色CIS对接ISP，环境可见光不足采用红外补光情况下，该接口切换ISP工作在红外灰度模式。

### **【需求】**

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## **rk\_aiq\_uapi2\_getGrayMode**

### **【描述】**

获取灰度模式

### **【语法】**

```
rk_aiq_gray_mode_t rk_aiq_uapi2_getGrayMode(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
Cspace	色彩空间模式	输出

### **【返回值】**

返回值	描述
RK_AIQ_GRAY_MODE_CPSL	由cpsl控制模式
RK_AIQ_GRAY_MODE_OFF	彩色模式
RK_AIQ_GRAY_MODE_ON	灰度模式

### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

## GAMMA相关

### rk\_aiq\_uapi2\_setGammaCoef

#### • 【描述】

通过GammaCoef和SlopeAtZero快速设置gamma曲线。其gamma曲线生成方式如下：

```
for(int i = 0; i < 49; i++) {
    gamma_Y_v11[i] = 4095 * pow(gamma_Y_v11[i] / 4095, 1 / GammaCoef + SlopeAtZero);
    gamma_Y_v11[i] = LIMIT_VALUE(gamma_Y_v11[i], 4095, 0);
}
```

#### 【语法】

```
XCamReturn rk_aiq_uapi2_setGammaCoef(const rk_aiq_sys_ctx_t* ctx, float GammaCoef, float SlopeAtZero);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
GammaCoef	gamma系数，取值范围[0,100]，默认值2.2，精度0.01	输入
SlopeAtZero	暗区斜率，取值范围[-0.05,0.05]，默认值0，精度0.001	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

#### 【需求】

- 头文件：rk\_aiq\_user\_api2\_imgproc.h
- 库文件：librkaiq.so

#### 【说明】

Api中Gamma曲线未按照场景进行切换，若场景变化，请重新通过api设置gamma曲线。

## 功能级API数据类型

### opMode\_t

#### 【说明】

定义自动手动模式。

#### 【定义】

```
typedef enum opMode_e {
    OP_AUTO = 0,
    OP_MANUAL = 1,
    OP_INVAL
} opMode_t;
```

#### 【成员】

成员名称	描述
OP_AUTO	自动模式
OP_MANUAL	手动模式
OP_INVAL	无效值

### paRange\_t

#### 【说明】

定义参数范围。

#### 【定义】

```
typedef struct paRange_s {
    float max;
    float min;
} paRange_t;
```

#### 【成员】

成员名称	描述
max	上限值
min	下限值

### aeMeasAreaType\_t

#### 【说明】

定义AE测量区域类型。

#### 【定义】

```

typedef enum aeMeasAreaType_e {
    AE_MEAS_AREA_AUTO = 0,
    AE_MEAS_AREA_UP,
    AE_MEAS_AREA_BOTTOM,
    AE_MEAS_AREA_LEFT,
    AE_MEAS_AREA_RIGHT,
    AE_MEAS_AREA_CENTER,
} aeMeasAreaType_t;

```

### 【成员】

成员名称	描述
AE_MEAS_AREA_AUTO	自动
AE_MEAS_AREA_UP	上方区域
AE_MEAS_AREA_BOTTOM	下方区域
AE_MEAS_AREA_LEFT	左边区域
AE_MEAS_AREA_RIGHT	右边区域
AE_MEAS_AREA_CENTER	中心区域

## expPwrLineFreq\_t

### 【说明】

定义抗闪频率。

### 【定义】

```

typedef enum expPwrLineFreq_e {
    EXP_PWR_LINE_FREQ_DIS = 0,
    EXP_PWR_LINE_FREQ_50HZ = 1,
    EXP_PWR_LINE_FREQ_60HZ = 2,
} expPwrLineFreq_t;

```

### 【成员】

成员名称	描述
EXP_PWR_LINE_FREQ_DIS	
EXP_PWR_LINE_FREQ_50HZ	50赫兹
EXP_PWR_LINE_FREQ_60HZ	60赫兹

## antiFlickerMode\_t

### 【说明】

定义抗闪模式。

### 【定义】

```
typedef enum antiFlickerMode_e {
    ANTIFLICKER_NORMAL_MODE = 0,
    ANTIFLICKER_AUTO_MODE = 1,
} antiFlickerMode_t;
```

### 【成员】

成员名称	描述
ANTIFLICKER_NORMAL_MODE	普通模式
ANTIFLICKER_AUTO_MODE	自动选择模式

## rk\_aiq\_wb\_op\_mode\_t

### 【说明】

定义白平衡工作模式

### 【定义】

```
typedef enum rk_aiq_wb_op_mode_s {
    RK_AIQ_WB_MODE_INVALID = 0,
    RK_AIQ_WB_MODE_MANUAL = 1,
    RK_AIQ_WB_MODE_AUTO = 2,
    RK_AIQ_WB_MODE_MAX
} rk_aiq_wb_op_mode_t;
```

### 【成员】

成员名称	描述
RK_AIQ_WB_MODE_MANUAL	白平衡手动模式
RK_AIQ_WB_MODE_AUTO	白平衡自动模式

## rk\_aiq\_wb\_mwb\_mode\_t

### 【说明】

定义手动白平衡模式类型

### 【定义】

```
typedef enum rk_aiq_wb_mwb_mode_e {
    RK_AIQ_MWB_MODE_INVALID = 0,
    RK_AIQ_MWB_MODE_CCT = 1,
    RK_AIQ_MWB_MODE_WBGAIN = 2,
    RK_AIQ_MWB_MODE_SCENE = 3,
} rk_aiq_wb_mwb_mode_t;
```

### 【成员】

成员名称	描述
RK_AIQ_MWB_MODE_CCT	色温
RK_AIQ_MWB_MODE_WBGAIN	增益系数
RK_AIQ_MWB_MODE_SCENE	场景

## rk\_aiq\_wb\_gain\_t

### 【说明】

定义白平衡增益参数

### 【定义】

```
typedef struct rk_aiq_wb_gain_s {
    float rgain;
    float grgain;
    float gbgain;
    float bgain;
} rk_aiq_wb_gain_t;
```

### 【成员】

成员名称	描述
rgain	R通道增益
grgain	G通道增益
gbgain	GB通道增益
bgain	B通道增益

## rk\_aiq\_wb\_scene\_t

### 【说明】

定义白平衡增益参数

### 【定义】

```
typedef enum rk_aiq_wb_scene_e {
    RK_AIQ_WBCT_INCANDESCENT = 0,
    RK_AIQ_WBCT_FLUORESCENT,
    RK_AIQ_WBCT_WARM_FLUORESCENT,
    RK_AIQ_WBCT_DAYLIGHT,
    RK_AIQ_WBCT_CLOUDY_DAYLIGHT,
    RK_AIQ_WBCT_TWILIGHT,
    RK_AIQ_WBCT_SHADE
} rk_aiq_wb_scene_t;
```

### 【成员】

成员名称	描述
RK_AIQ_WBCT_INCANDESCENT	白炽灯
RK_AIQ_WBCT_FLUORESCENT	荧光灯
RK_AIQ_WBCT_WARM_FLUORESCENT	暖荧光灯
RK_AIQ_WBCT_DAYLIGHT	日光
RK_AIQ_WBCT_CLOUDY_DAYLIGHT	阴天
RK_AIQ_WBCT_TWILIGHT	暮光
RK_AIQ_WBCT_SHADE	阴影

## rk\_aiq\_wb\_cct\_t

### 【说明】

定义白平衡增益参数

### 【定义】

```
typedef struct rk_aiq_wb_cct_s {
    float CCT;
    float CCRI;
} rk_aiq_wb_cct_t;
```

### 【成员】

成员名称	描述
CCT	相关色温
CCRI	相关显色指数

## rk\_aiq\_wb\_mwb\_attrib\_t

### 【说明】

定义手动白平衡属性

### 【定义】

```
typedef struct rk_aiq_wb_mwb_attrib_s {
    rk_aiq_wb_mwb_mode_t mode;
    union MWBPara_u {
        rk_aiq_wb_gain_t gain;
        rk_aiq_wb_scene_t scene;
        rk_aiq_wb_cct_t cct;
    } para;
} rk_aiq_wb_mwb_attrib_t;
```

### 【成员】

成员名称	描述
mode	模式选择
para	模式对应的参数配置

## rk\_aiq\_uapiV2\_wb\_awb\_wbGainOffset\_t

### 【说明】

定义自动白平衡gain偏移

### 【定义】

```
typedef struct rk_aiq_uapiV2_wb_awb_wbGainOffset_s{
    rk_aiq_uapi_sync_t sync;
    CalibDbV2_Awb_gain_offset_cfg_t gainOffset;
}rk_aiq_uapiV2_wb_awb_wbGainOffset_t;
```

### 【成员】

成员名称	描述
sync	参考rk_aiq_uapi_sync_t定义说明，参见“概述/API说明”章节
gainOffset	参看后面CalibDbV2_Awb_gain_offset_cfg_t描述

## CalibDbV2\_Awb\_gain\_offset\_cfg\_t

### 【说明】

定义自动白平衡gain偏移

### 【定义】

```
typedef struct CalibDbV2_Awb_gain_offset_cfg_s{
    bool enable;
    float offset[4];
}CalibDbV2_Awb_gain_offset_cfg_t;
```

### 【成员】

成员名称	描述
enable	使能开关 取值0或1，分别代表不使能、使能
offset	wbgain与offset相加，对应R GR GB B通道的偏移 取值范围由wbgain与offset相加值确定，wbgain与offset相加后范围在[0,8] (ISP21)

## rk\_aiq\_uapiV2\_wbV32\_awb\_gainAdjust\_t

### 【说明】

定义自动白平衡色调调整参数

## 【定义】

```
typedef struct rk_aiq_uapiV2_wbV32_awb_gainAdjust_s {
    bool enable;
    CalibDbV2_Awb_Ctrl_Dat_Selt_t ctrlDataSelt;
    CalibDbV2_Awb_Gain_Adj_Dat_Sl_t adjDataSelt;
    CalibDbV2_Awb_Cct_Lut_Cfg_Lv2_t lutAll[RK_UAPI_AWB_CT_LUT_NUM];
    int lutAll_len;
} rk_aiq_uapiV2_wbV32_awb_gainAdjust_t;

typedef enum CalibDbV2_Awb_Ctrl_Dat_Selt_e{
    AWB_CTRL_DATA_ISO = 0,
    AWB_CTRL_DATA_LV = 1,
}CalibDbV2_Awb_Ctrl_Dat_Selt_t;

typedef enum CalibDbV2_Awb_Gain_Adj_Dat_Sl_e{
    AWB_GAIN_ADJ_DATA_GAIN = 0,
    AWB_GAIN_ADJ_DATA_CT = 1,
}CalibDbV2_Awb_Gain_Adj_Dat_Sl_t;

typedef struct CalibDbV2_Awb_Cct_Lut_Cfg_Lv2_s {
    float ctlData;
    float rgct_in_ds[CALD_AWB_RGCT_GRID_NUM];
    float bgcri_in_ds[CALD_AWB_BGCRI_GRID_NUM];
    float rgct_lut_out[CALD_AWB_RGCT_GRID_NUM*CALD_AWB_BGCRI_GRID_NUM];
    float bgcri_lut_out[CALD_AWB_RGCT_GRID_NUM*CALD_AWB_BGCRI_GRID_NUM];
} CalibDbV2_Awb_Cct_Lut_Cfg_Lv2_t;
```

## 【成员】

名称	描述
enable	色调调整使能 取值0或1，分别代表不使能、使能
ctrlDataSel	指定色调调整表的控制量，即lutAll.ctrlData 取值AWB_CTRL_DATA_LV或AWB_CTRL_DATA_ISO ctrlDataSel=AWB_CTRL_DATA_LV时，表示不同的环境亮度下可以配置不同的lut用于色调调整 ctrlDataSel=AWB_CTRL_DATA_ISO时，表示不同的ISO下可以配置不同的lut用于色调调整
adjDataSel	指定色调调整表rgct, bgcri的变量含义 取值AWB_GAIN_ADJ_DATA_GAIN或AWB_GAIN_ADJ_DATA_CT; 取值为AWB_GAIN_ADJ_DATA_GAIN时，2维lut的两个维度分别为Rgain(白平衡增益之红色通道), Bgain(白平衡增益之蓝色通道); 取值为AWB_GAIN_ADJ_DATA_CT时，2维lut的两个维度分别为CT(相关色温), CRI(显色指数)
lutAll	lutAll中所有成员概述看这里： 不同的lutAll.ctrlData可以配置不同的lut用于色调调整 色调调整通过2维线性插值实现，配置输入的2维表lut_in和输出的2维表lut_out，代入场景中 (rgct, bgcri) 值即可得到调整后的wbgain值，其中2维表为11行9列的表。因此通过修改lut_out表格中每个位置的 (rgct, bgcri) 值就可以实现色调调整（参考下面的配置2） lut_out的rgct分量存在lutAll.rgct_lut_out中, lut_out表的bgcri分量存在lutAll.bgcri_lut_out中 lut_in的rgct分量和bgcri分量没有直接存下来，通过下面的方式生成： lut_in的rgct分量每一行都是复制lutAll.rgct_in_ds的值，lut_in的bgcri分量的每一列都是复制lutAll.bgcri_in_ds的值（参考下面的配置1）
lutAll.ctrlData	色调调整表的控制量 ctrlDataSel=AWB_CTRL_DATA_LV时，表示环境亮度 ctrlDataSel=AWB_CTRL_DATA_ISO时，表示ISO 取值范围0-255000
lutAll.rgct_in_ds	用于生成输入的2维表lut_in，参考前面lutAll描述 有9个元素，从左到右值从小到大 若adjDataSel=AWB_GAIN_ADJ_DATA_GAIN，存rgain的值，取值范围0-8 若adjDataSel=AWB_GAIN_ADJ_DATA_CT，存ct的值，取值范围0-10000
lutAll.bgcri_in_ds	用于生成输入的2维表lut_in，参考前面lutAll描述 有11个元素，从左到右值从大到小 若adjDataSel=AWB_GAIN_ADJ_DATA_GAIN，存bgain的值，取值范围0-8 若adjDataSel=AWB_GAIN_ADJ_DATA_CT，存cri的值，取值范围-2-4
lutAll.rgct_lut_out	输出的2维表，参考前面lutAll描述 若adjDataSel=AWB_GAIN_ADJ_DATA_CT，存ct的值，取值范围0-10000 若adjDataSel=AWB_GAIN_ADJ_DATA_GAIN，存rgain的值，取值范围0-8

名称	描述
lutAll.bgcri_lut_out	输出的2维表，参考前面lutAll描述 若adjDataSel=AWB_GAIN_ADJ_DATA_CT，存cri的值，取值范围0-10000 若adjDataSel=AWB_GAIN_ADJ_DATA_GAIN，存bgain的值，取值范围0-8

更多说明参考《Rockchip\_Color\_Optimization\_Guide》文档里面的WBGain色调调整章节

## rk\_aiq\_uapiV2\_wbV32\_awb\_mulWindow\_t

### 【说明】

定义子窗口参数，落入子窗口内的点将不进行白点检测，通常将该子窗口配置为人脸的位置。

### 【定义】

```
typedef struct rk_aiq_uapiV2_wbV32_awb_mulWindow_s {
    rk_aiq_uapi_sync_t sync;
    bool enable;
    float window[4][4];
} rk_aiq_uapiV2_wbV32_awb_mulWindow_t;
```

### 【成员】

成员名称	描述
sync	参考rk_aiq_uapi_sync_t说明，参见“概述/API说明”章节
enable	多窗口使能 取值0或1，分别代表不使能、使能
window	第一个维度4表示，最多可以配置4个子窗口 第二维度4对应于窗口的[hoffset,voffset,hszie,vszie]，即窗口的水平方向偏移 =width* hoffset，垂直方向偏移= height* voffset，宽=width * hsize，高= height * vszie 取值范围[0-1]

## rk\_aiq\_wb\_awb\_alg\_method\_t

### 【说明】

定义两种自动白平衡方法。

### 【定义】

```
typedef enum rk_aiq_wb_awb_alg_method_s {
    RK_AIQ_AWB_ALG_TYPE_INVALID = 0,
    RK_AIQ_AWB_ALG_TYPE_GLOABL = 1,
    RK_AIQ_AWB_ALG_TYPE_GRAYWORD = 2,
} rk_aiq_wb_awb_alg_method_t;
```

### 【成员】

枚举名称	描述
RK_AIQ_AWB_ALG_TYPE_INVALID	非法方法
RK_AIQ_AWB_ALG_TYPE_GLOABL	rk自动白平衡算法，具体算法参考 《Rockchip_Color_Optimization_Guide》文档中自动 白平衡部分
RK_AIQ_AWB_ALG_TYPE_GRAYWORD	白点选取参考灰度世界法，即所有点都参与StaGain计 算，而RK_AIQ_AWB_ALG_TYPE_GLOABL的白点选取 要同时满足三个域的白点检测条件

## rk\_aiq\_uapiV2\_wb\_awb\_dampFactor\_t

### 【说明】

定义自动白平衡收敛参数。

### 【定义】

```
typedef struct rk_aiq_uapiV2_wb_awb_dampFactor_s {
    float dFStep;
    float dFMin;
    float dFMax;
} rk_aiq_uapiV2_wb_awb_dampFactor_t;
```

### 【成员】

名称	描述
dFStep	wbGainDampFactor变化的步长 取值范围0-1
dFMin	wbGainDampFactor最小值 取值范围0-1
dFMax	wbGainDampFactor最大值 取值范围0-1

## CalibDbV2\_Awb\_DaylgtClip\_Cfg\_t

### 【说明】

定义室外最低色温参数。

### 【定义】

```
typedef struct CalibDbV2_Awb_DaylgtClip_Cfg_s {
    bool enable;
    float outdoor_cct_min;
} CalibDbV2_Awb_DaylgtClip_Cfg_t;
```

### 【成员】

名称	描述
enable	室外最低色温限制使能 取值0或1，分别代表不使能、使能
outdoor_cct_min	室外最低色温取值不限

更多说明参考《Rockchip\_Color\_Optimization\_Guide》文档里面的WBGain范围限制章节

## **rk\_aiq\_uapiV2\_wb\_awb\_cctClipCfg\_t**

### **【说明】**

定义室外最低色温参数。

### **【定义】**

```
typedef struct rk_aiq_uapiV2_wb_awb_cctClipCfg_s {
    bool enable;
    float cct[RK_UAPI_AWB_CT_GRID_NUM];
    int cct_len;
    float cri_bound_up[RK_UAPI_AWB_CT_GRID_NUM];
    float cri_bound_low[RK_UAPI_AWB_CT_GRID_NUM];
} rk_aiq_uapiV2_wb_awb_cctClipCfg_t;
```

### **【成员】**

名称	描述
enable	色温范围限制使能 取值0或1，分别代表不使能、使能
cct	对应图上围成区域的圆点cct坐标 上下边界采用相同的cct采样坐标 取值[1000-10000]
cri_bound_up	对应图上围成区域的下边界圆点cri分量 对于位于区域内的点即cri0>=-cri_bound_up，否则 取值-1到1
cri_bound_low	对应图上围成区域的上边界圆点cri分量 对于位于区域内的点有cri0<=cri_bound_low 取值-1到1

更多说明参考《Rockchip\_Color\_Optimization\_Guide》文档里面的WBGain范围限制章节

## **rk\_aiq\_uapiV2\_wbV32\_awb\_attrib\_t**

### **【说明】**

定义自动白平衡api参数

### **【定义】**

```
typedef struct rk_aiq_uapiV2_wbV32_awb_attrib_s {
    rk_aiq_wb_awb_alg_method_t algMtdTp;
    bool algMtdTp_valid;
    rk_aiq_uapiV2_wb_awb_dampFactor_t dampFactor;
```

```

bool dampFactor_valid;
CalibDbV2_Awb_gain_offset_cfg_t wbGainOffset;
bool wbGainOffset_valid;
rk_aiq_uapiV2_wbV32_awb_mulWindow_t multiWindow;
bool multiWindow_valid;
CalibDbV2_Awb_DaylgtClip_Cfg_t wbGainDaylightClip;
bool wbGainDaylightClip_valid;
rk_aiq_uapiV2_wb_awb_cctClipCfg_t wbGainClip;
bool wbGainClip_valid;
rk_aiq_uapiV2_wbV32_awb_gainAdjust_t wbGainAdjust;
bool wbGainAdjust_valid;
} rk_aiq_uapiV2_wbV32_awb_attrib_t;

```

## 【成员】

成员名称	描述
algMtdTp	参考前述rk_aiq_wb_awb_alg_method_t
algMtdTp_valid	取值true时，algMtdTp参数生效，否则不生效
dampFactor	参考前述rk_aiq_uapiV2_wb_awb_dampFactor_t
dampFactor_valid	取值true时，dampFactor参数生效，否则不生效
wbGainOffset	参考前述rk_aiq_uapiV2_wb_awb_wbGainOffset_t
wbGainOffset_valid	取值true时，wbGainOffset参数生效，否则不生效
multiWindow	参考前述rk_aiq_uapiV2_wbV32_awb_mulWindow_t
multiWindow_valid	取值true时，multiWindow参数生效，否则不生效
wbGainDaylightClip	参考前述CalibDbV2_Awb_DaylgtClip_Cfg_t
wbGainDaylightClip_valid	取值true时，wbGainDaylightClip参数生效，否则不生效
wbGainClip	参考前述rk_aiq_uapiV2_wb_awb_cctClipCfg_t
wbGainClip_valid	取值true时，wbGainClip_valid参数生效，否则不生效
wbGainAdjust	参考前述CalibDbV2_Awb_gain_offset_cfg_t
wbGainAdjust_valid	取值true时，wbGainAdjust_valid参数生效，否则不生效

## rk\_aiq\_uapiV2\_wbV32\_attrib\_t

### 【说明】

定义白平衡API支持的全部参数。

### 【定义】

```

typedef struct rk_aiq_uapiV2_wbV32_attrib_s {
    rk_aiq_uapi_sync_t sync;
    bool byPass;
    rk_aiq_wb_op_mode_t mode;
    rk_aiq_wb_mwb_attrib_t stManual;
    rk_aiq_uapiV2_wbV32_awb_attrib_t stAuto;
} rk_aiq_uapiV2_wbV32_attrib_t;

```

### 【成员】

成员名称	描述
sync	参考rk_aiq_uapi_sync_t说明，参见“概述/API说明”章节
bypass	取值0或1 0表示做白平衡校正，使用的白平衡增益由表格后面的参数控制控制 1表示不执行白平衡校正
mode	自动或手动白平衡模式控制参数，参考前述rk_aiq_wb_op_mode_t
stManual	手动白平衡参数，参考前述rk_aiq_wb_mwb_attrib_t
stAuto	自动白平衡参数，参考前述rk_aiq_uapiV2_wbV21_awb_attrib_t

## rk\_aiq\_af\_zoomrange

【说明】 定义zoom取值范围

### 【定义】

```

typedef struct {
    int min_pos;
    int max_pos;
    float min_fl;
    float max_fl;
} rk_aiq_af_zoomrange;

```

### 【成员】

成员名称	描述
min_pos	zoom最小值
max_pos	zoom最大值
min_fl	最小焦距
max_fl	最大焦距

## rk\_aiq\_af\_focusrange

【说明】 定义focus取值范围

### 【定义】

```
typedef struct {
    int min_pos;
    int max_pos;
} rk_aiq_af_focusrange;
```

### 【成员】

成员名称	描述
min_pos	focus最小值
max_pos	focus最大值

## rk\_aiq\_af\_result\_t

【说明】 定义af搜索结果

### 【定义】

```
typedef enum rk_aiq_af_sec_stat_e
{
    RK_AIQ_AF_SEARCH_INVAL = 0,
    RK_AIQ_AF_SEARCH_RUNNING = 1,
    RK_AIQ_AF_SEARCH_END = 2
} rk_aiq_af_sec_stat_t;

typedef struct {
    rk_aiq_af_sec_stat_t stat;
    int32_t final_pos;
} rk_aiq_af_result_t;
```

### 【成员】

成员名称	描述
stat	对焦状态
final_pos	最终focus位置

## rk\_aiq\_ccm\_matrix\_t

### 【说明】

定义CCM矩阵。

### 【定义】

```
typedef struct rk_aiq_ccm_matrix_s {
    float ccMatrix[9];
    float ccOffsets[3];
} rk_aiq_ccm_matrix_t;
```

### 【成员】

成员名称	描述
ccMatrix	手动模式下色彩校正矩阵； 取值范围：[-8, 7.992]
ccOffsets	手动模式下R\G\B分量偏移； 取值范围：[-4096, 4095]

## rk\_aiq\_lut3d\_table\_t

### 【说明】

定义3DLUT 3D查找表。

### 【定义】

```
typedef struct rk_aiq_lut3d_table_s{
    unsigned short look_up_table_r[729];
    unsigned short look_up_table_g[729];
    unsigned short look_up_table_b[729];
} rk_aiq_lut3d_table_t;
```

### 【成员】

成员名称	描述
look_up_table_r	手动模式下R通道查找表； 取值范围：[0x0, 0x3ff]
look_up_table_g	手动模式下G通道查找表； 取值范围：[0x0, 0xffff]
look_up_table_b	手动模式下B通道查找表； 取值范围：[0x0, 0x3ff]

## 统计信息

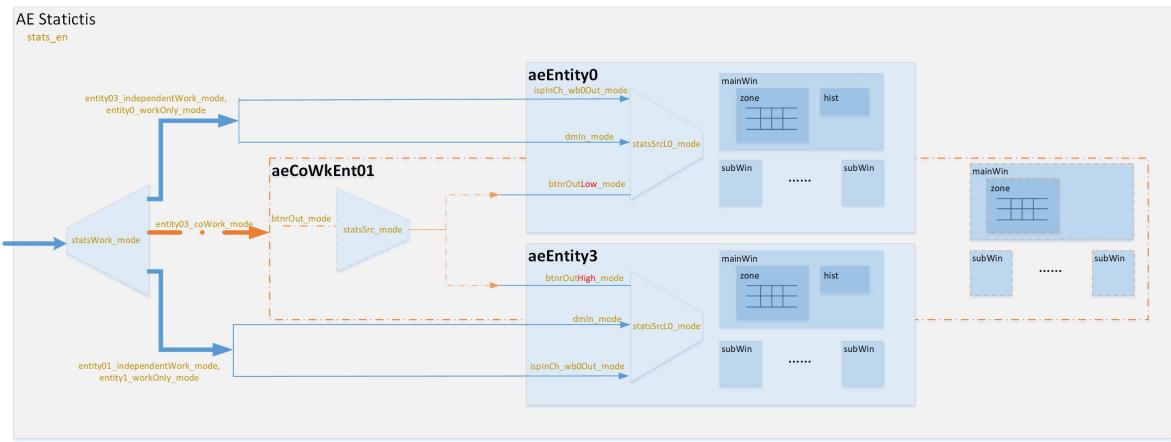
### 概述

#### 概述

ISP39支持对图像数据处理获取到AWB / AE / AF 3A控制算法需要的相关的统计信息

### 功能描述

#### AE统计信息



AE硬件统计信息主要包含以下几个部分：

基于raw域的256段加权直方图统计信息、分块R/G/B/Y 均值统计信息；

### 基于raw域的AE统计

- 统计信息分为分块亮度统计和直方图统计。
- 分块亮度统计：最大支持15X15非独立子窗口分块，最小支持1X1非独立子窗口分块，每个分块均可输出10bit R/B通道亮度均值和12bit G通道均值，默认采用15X15分块；
- 加权直方图统计，根据分块数和对应分配的权重，进行256段8bit亮度统计，每个亮度分段内像素数的有效bit数为28bit。
- 3576平台共包含两个AE统计模块，每个统计模块的统计信息内容如上所述，区别在于统计模块的输入数据源不同。两个统计模块分别用entity0及entity3进行表示。

## AWB统计信息

**AWB硬件统计数据源：**

- hdr*模式下用于选择进入rawawb统计的数据来源，共有以下几种：
  - raw\_in\_short--->blc--->offset--->lsc--->rawawb\_statistics
  - raw\_in\_long--->blc--->offset--->lsc--->rawawb\_statistics
  - bay2dnr\_output--->lsc--->rawawb\_statistics
  - hdr\_drc\_output--->rawawb\_statistics
- linear*模式下用于选择进入rawawb统计的数据来源，共有以下几种：
  - raw\_in--->blc--->offset--->lsc--->rawawb\_statistics
  - bay2dnr\_output--->blc1-->offset--->lsc--->rawawb\_statistics
  - hdr\_drc\_output--->rawawb\_statistics

其中：

由json中wb模块的**hw\_awbCfg\_statsSrc\_mode**结构体参数指定

blc指的是主通路的blc(包括blc0和blc1)

blc1指的是主通路的blc1

offset为相对于主通路的blc差异值，由json中wb模块的blc2ForAwb结构体参数指定；

lsc值rawawb通路上的lsc, 该参数同主通路lsc，只可以通过hw\_awbCfg\_lsc\_en去控制是否执行lsc;

raw\_in\_xxx指的是sensor输出的raw；

bay2dnr\_output指是bayer2dnr模块的输出；

hdr\_drc\_output指的是drc模块的输出；

**AWB硬件统计信息包含白点统计信息和分块统计信息**

- 白点统计信息

记录主窗口内4个光源2种大小的白区里的RGin,BGain累加值及个数；

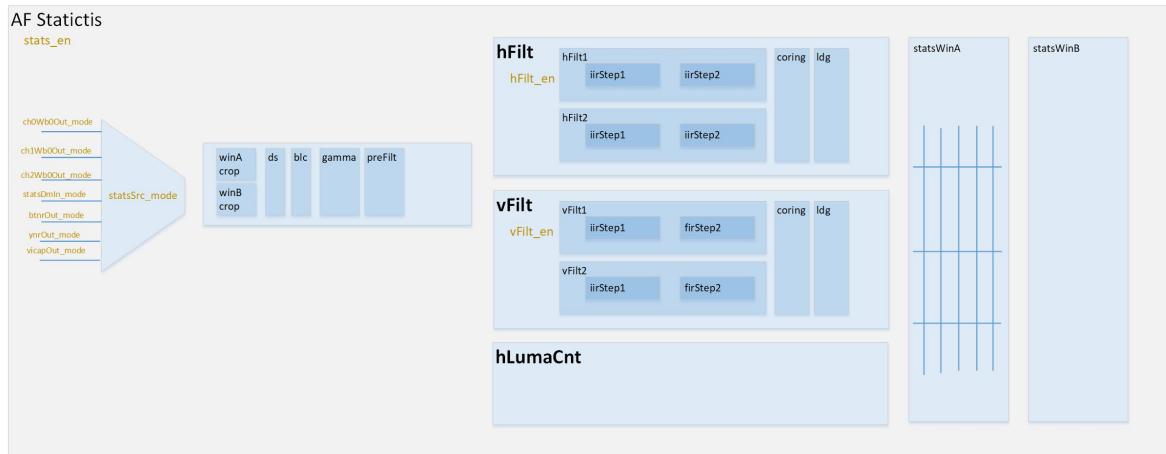
4个UV或XY域附加框里的RGin,BGain累加值及个数；

白点亮度直方图

- 分块统计信息

图像15x15分块，每个块所有点或白点的R,G,B均值

## AF统计信息



### AFHW Stats硬件统计：

- HDR多帧模式下，支持选择L / S 2个通道中的其中1个通道。同时支持2通道合成后的数据作为输入。
- 线性模式下，仅有1个通道数据。
- 支持水平方向2个可配置频段的FV输出，垂直方向上2个可配置频段的FV输出

## API参考

### **rk\_aiq\_uapi\_sysctl\_get3AStatsBlk**

#### 【描述】

同步获取3A统计信息。除SmartIr应用外，建议使用 rk\_aiq\_uapi2\_stats\_getIspStats 替代。

#### 【语法】

```
XCamReturn
rk_aiq_uapi_sysctl_get3AStatsBlk(const rk_aiq_sys_ctx_t* ctx, rk_aiq_isp_stats_t **stats, int
timeout_ms);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
stats	统计信息结构体指针	输出
timeout_ms	超时时间, -1意思是无限等待, 直到有统计数据	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

#### 【需求】

- 头文件: rk\_aiq\_user\_api\_sysctl.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi\_sysctl\_release3AStatsRef**

#### 【描述】

释放获取的3A统计信息, 与rk\_aiq\_uapi\_sysctl\_get3AStatsBlk配套使用。

#### 【语法】

```
void
rk_aiq_uapi_sysctl_release3AStatsRef(const rk_aiq_sys_ctx_t* ctx, rk_aiq_isp_stats_t *stats);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
stats	统计信息结构体指针	输入

#### 【返回值】

无

#### 【需求】

- 头文件: rk\_aiq\_user\_api\_sysctl.h
- 库文件: librkaiq.so

### **rk\_aiq\_uapi2\_stats\_getIspStats**

#### 【描述】

获取3A统计信息。

#### 【语法】

```

XCamReturn
rk_aiq_uapi2_stats_getIspStats(const rk_aiq_sys_ctx_t* ctx,
                                rk_aiq_isp_statistics_t *stats, int timeout_ms);

```

### 【参数】

参数名称	描述	输入/输出
ctx	AIQ上下文指针	输入
stats	统计信息结构体指针	输入
timeout_ms	超时参数 -1: 无限等待, 直到获取到下一帧统计 0: 立即返回, 获取当前帧统计 >0: 超时等待, 直到获取到下一帧统计或者超时	输入

### 【返回值】

返回值	描述
0	成功
非0	失败, 详见错误码表

### 【需求】

- 头文件: rk\_aiq\_user\_api2\_isp39.h 或者 rk\_aiq\_user\_api2\_isp.h
- 库文件: librkaiq.so

### 参考代码

/media/isp//camera\_engine\_rkaiq/rkisp\_demo/demo/rkisp\_demo.c

## 数据类型

### **rk\_aiq\_isp\_stats\_t**

#### 【说明】

AIQ 3A统计信息

#### 【定义】

```

typedef struct {
    uint32_t frame_id;
    union {
        rk_aiq_isp_aec_stats_t aec_stats; /* rv1106 ... */
        RKAiqAecStatsV25_t  aec_stats_v25; /* rv1103b rk3576 */
    };
    bool bValid_aec_stats;
    int awb_hw_ver;
    union {
        rk_aiq_awb_stat_res_v200_t awb_stats_v200;
        rk_aiq_awb_stat_res2_v201_t awb_stats_v21; /* rk356x */
        rk_aiq_isp_awb_stats2_v3x_t awb_stats_v3x; /* rk3588 */
        rk_aiq_isp_awb_stats_v32_t awb_stats_v32; /* rv1106, rk3562 */
        awbStats_stats_priv_t awb_stats_v39; /* rv1103b rk3576 */
    };
}

```

```

};

bool bValid_awb_stats;
int af_hw_ver;
union {
    rk_aiq_isp_af_stats_t af_stats;
    rk_aiq_isp_af_stats_v3x_t af_stats_v3x; /* rk3588 */
    afStats_stats_t afStats_stats; /* rk3576 */
};

bool bValid_af_stats;
} rk_aiq_isp_stats_t;

```

### 【成员】

成员名称	描述
frame_id	统计帧ID
aec_stats_v25	ae统计信息
bValid_aec_stats	ae统计信息是否有效
awb_stats_v39	awb统计信息
bValid_awb_stats	ae统计信息是否有效
afStats_stats	af统计信息
bValid_af_stats	af统计信息是否有效
其他	ISP39不使用

## AE统计数据类型

### RKAiqAecStats\_t

#### 【说明】

定义AE数据信息，详细内容参见AE章节的功能描述。

#### 【定义】

```

typedef struct RKAiqAecStatsV25_s {
    aeStats_stats_t ae_data; //AeHwMeas_Res
    RKAiqAecExpInfo_t ae_exp; //AeExp_Info
} RKAiqAecStatsV25_t;

```

### 【成员】

成员名称	描述
ae_data	AE模块硬件统计信息
ae_exp	AE模块sensor曝光信息

### RKAiqAecExpInfo\_t

#### 【说明】

AE模块曝光参数信息

## 【定义】

```
typedef struct RKAiqAecExplInfo_s {
    RkAiqExpParamComb_t LinearExp;
    RkAiqExpParamComb_t HdrExp[3];
    RkAiqIrisParamComb_t Iris;
    uint16_t line_length_pixels;
    uint32_t frame_length_lines;
    float pixel_clock_freq_mhz;
    CISFeature_t CISFeature;
    RKAiqExpl2cParam_t exp_i2c_params;
} RKAiqAecExplInfo_t;
```

## 【成员】

成员名称	描述
LinearExp	非HDR模式的曝光参数信息，包含曝光实际值和RK格式的寄存器值
HdrExp	HDR模式的曝光参数信息，包含曝光实际值和RK格式的寄存器值，至多支持3帧曝光
exp_i2c_params	曝光参数的sensor寄存器值，第三方AE方案使用的曝光寄存器值参数
line_length_pixels	hts，其值由sensor的配置序列决定
frame_length_lines	vts，其值由sensor的配置序列决定
pixel_clock_freq_mhz	pclk，单位MHz，其值由sensor的配置序列决定

## 【注意事项】

- HdrExp表示HDR模式下的曝光参数信息，至多支持3TO1。HDR 2TO1：下标0表示短帧曝光参数，下标1表示长帧曝光参数，下标2无效；HDR 3TO1：下标0表示短帧曝光参数，下标1表示中帧曝光参数，下标2表示长帧曝光参数。

## RkAiqExpParamComb\_t

### 【说明】

AE模块曝光参数信息详细内容

## 【成员】

```
typedef struct {
    RkAiqExpRealParam_t exp_real_params; //real value
    RkAiqExpSensorParam_t exp_sensor_params;//RK reg value
} RkAiqExpParamComb_t;
```

成员名称	描述
exp_real_params	曝光分量的实际物理值
exp_sensor_params	曝光分量的sensor寄存器值，遵循RK曝光设置方式

```

typedef struct RkAiqExpRealParam_s {
    float integration_time;
    float analog_gain;
    float digital_gain;
    float isp_dgain;
    int iso;
    int dcg_mode;
} RkAiqExpRealParam_t;

```

成员名称	描述
integration_time	曝光积分时间，单位为秒(s)
analog_gain	sensor的模拟增益/Total增益，单位为倍数
digital_gain	sensor的数字增益，单位为倍数，当sensor的数字增益起弥补模拟增益精度作用时，配置为1x，整体增益值写入analog_gain中
isp_dgain	isp数字增益，单位为倍数，目前暂无效，默认值为1x
iso	总增益值，iso表示系统增益，以常数50乘以倍数为单位，iso = total增益 *50
dchg_mode	dual conversion gain模式

```

typedef struct RkAiqExpSensorParam_s {
    unsigned short fine_integration_time;
    unsigned short coarse_integration_time;
    unsigned short analog_gain_code_global;
    unsigned short digital_gain_global;
    unsigned short isp_digital_gain;
} RkAiqExpSensorParam_t;

```

成员名称	描述
fine_integration_time	sensor曝光积分时间的小数行寄存器值，仅当sensor支持小数行时，需要填写
coarse_integration_time	sensor曝光积分时间对应的寄存器值，单位为行数
analog_gain_code_global	sensor模拟增益对应的寄存器值
digital_gain_global	sensor数字增益对应的寄存器值
isp_digital_gain	isp数字增益寄存器值，暂时无效

### 【注意事项】

- 不同sensor的数字增益作用不同，有的是用于增大感光度范围，有的是用于补足模拟增益的程度。因此目前先不将数字增益单独列出，其大小和对应寄存器值全部并入模拟增益中。
- dual conversion gain模式共有三种状态，值为-1代表sensor不支持dcg，值为0代表LCG，值为1代表HCG

**RKAiqExpl2cParam\_t**

## 【说明】

AE模块I2c曝光参数(一般为第三方AE使用)

## 【定义】

```
#define MAX_I2CDATA_LEN 64
typedef struct RKAiqExpl2cParam_s {
    bool      bValid;
    unsigned int nNumRegs;
    unsigned int RegAddr[MAX_I2CDATA_LEN];
    unsigned int AddrByteNum[MAX_I2CDATA_LEN];
    unsigned int RegValue[MAX_I2CDATA_LEN];
    unsigned int ValueByteNum[MAX_I2CDATA_LEN];
    unsigned int DelayFrames[MAX_I2CDATA_LEN];
} RKAiqExpl2cParam_t;
```

## 【成员】

成员名称	描述
bValid	I2c参数生效使能位 true：使用RKAiqExpl2cParam_t设置寄存器值（一般为第三方AE使用）； false：使用RK格式的寄存器参数进行寄存器值设置
nNumRegs	设置的寄存器值数量
RegAddr	寄存器地址数组，元素最大个数为64
AddrByteNum	寄存器地址的比特长度，元素最大个数为64
RegValue	寄存器值数组，元素最大个数为64
ValueByteNum	寄存器值的比特长度，元素最大个数为64
DelayFrames	寄存器值延迟生效帧数数组，元素最大个数为64

## 【注意事项】

- 该寄存器参数仅在第三方AE应用，且bValid为true时才有效，否则默认使用RkAiqExpParamComb\_t中的RK格式寄存器参数
- 可支持设置的寄存器值最大个数为64

## RkAiqIrisParamComb\_t

### 【说明】

AE模块光圈参数

### 【定义】

```
typedef struct {
    RkAiqPIrisParam_t PIris;
    RkAiqDCIrisParam_t DCIris;
    RkAiqHDCIrisParam_t HDCIris;
} RkAiqIrisParamComb_t;

typedef struct {
    int      step;
    int      gain;
    bool     update;
}
```

```

} RkAiqPIrisParam_t;
typedef struct {
    int    pwmDuty; //percent value,range = 0-100
    bool   update;
} RkAiqDCIrisParam_t;
typedef struct {
    int    target;
    bool   update;
    int    adc;
    int    zoomPos;
} RkAiqHDCIrisParam_t;

```

### 【成员】

成员名称	子成员	描述
PIris	step gain update	P光圈参数： P光圈步进寄存器值 P光圈步进等效增益值 P光圈步进更新标志
DCIris	pwmDuty update	DC光圈参数： PWM占空比值，取值范围：1~100 DC光圈参数更新标志
DCIris	target update adc zoomPos	HDC光圈参数： HDC光圈参数更新标志 HDC光圈驱动中的ADC值 镜头中的zoom位置

## RkAiqAecHwStatsRes\_t

### 【说明】

AE模块硬件统计信息

### 【定义】

```

typedef struct RkAiqAecHwStatsRes_s {
    aeStats_entityGroupWk_mode_t hw_ae_entityGroup_mode;
    aeStats_entityGrpStats_t entityGroup;
} RkAiqAecHwStatsRes_t;

typedef struct aeStats_entityGrpStats_s {
    aeStats_entityStats_t coWkEnt03;
    aeStats_entitiesStats_t entities;
} aeStats_entityGrpStats_t;

typedef struct aeStats_entitiesStats_s {
    aeStats_entityStats_t entity0;
    aeStats_entityStats_t entity3;
} aeStats_entitiesStats_t;

```

### 【成员】

成员名称	描述
hw_ae_entityGroup_mode	AE统计模块的工作模式，用于判断工作的统计模块进行统计获取： aeStats_entity03_coWk_mode，即coWkEnt03模块有效，统计需从coWkEnt03中获取； aeStats_entity03_independentWk_mode，即entity0 entity3模块有效，统计可从entity0 entity3中获取； aeStats_entity0_wkOnly_mode，仅entity0模块有效，统计可从entity0中获取； aeStats_entity3_wkOnly_mode，仅entity3模块有效，统计可从entity3中获取；
entityGroup	AE各统计模块对应的统计结果

#### 【注意】

- entityGroup中包含了3个统计模块的统计信息。其中entity0与entity3为硬件统计模块，coWkEnt03为软件虚拟出的统计模块。hw\_ae\_entityGroup\_mode = aeStats\_entity03\_coWk\_mode时，coWkEnt03有效，其主要针对built-in-HDR应用。

### aeStats\_entityStats\_t

#### 【说明】

AE硬件计模块对应的统计结果

#### 【定义】

```
typedef struct aeStats_entityStats_s {
    aeStats_mainWinStats_t mainWin;
    aeStats_subWinStats_t subWin[AESTATS_SUBWIN_NUM];
    aeStats_histStats_t hist;
} aeStats_entityStats_t;
```

#### 【成员】

成员名称	描述
mainWin	基于raw域的分块亮度主窗口统计信息
subWin	基于raw域的独立子窗口统计信息
hist	基于raw域 直方图统计信息

#### 【注意事项】

- 3576平台不支持设置独立子窗口，故subWin中的统计信息无效

### aeStats\_mainWinStats\_t

#### 【说明】

基于raw域的分块亮度主窗口统计信息，包含15X15个非独立子窗口的R/G/B均值亮度

#### 【定义】

```

typedef struct aeStats_mainWinStats_s {
    uint16_t hw_ae_meanBayerR_val[AESTATS_ZONE_15x15_NUM];
    uint16_t hw_ae_meanBayerGrGb_val[AESTATS_ZONE_15x15_NUM];
    uint16_t hw_ae_meanBayerB_val[AESTATS_ZONE_15x15_NUM];
} aeStats_mainWinStats_t;

```

### 【成员】

成员名称	描述
hw_ae_meanBayerR_val	非独立子窗口分块的r通道均值亮度信息。有效比特数：10bit。
hw_ae_meanBayerGrGb_val	非独立子窗口分块的g通道均值亮度信息。有效比特数：12bit。
hw_ae_meanBayerB_val	非独立子窗口分块的b通道均值亮度信息。有效比特数：10bit。

### 【注意事项】

- 基于raw域的分块亮度主窗口统计信息，包含15X15个非独立子窗口，非独立子窗口有最小尺寸限制，要求最小尺寸为16X4。

## aeStats\_subWinStats\_t

### 【说明】

基于raw域的独立子窗口统计信息

### 【定义】

```

typedef struct aeStats_subWinStats_s {
    uint32_t hw_ae_sumBayerR_val;
    uint32_t hw_ae_sumBayerGrGb_val;
    uint32_t hw_ae_sumBayerB_val;
} aeStats_subWinStats_t;

```

### 【成员】

成员名称	描述
hw_ae_sumBayerR_val	独立子窗口的r通道亮度总和信息。有效比特数：29bit。
hw_ae_sumBayerGrGb_val	独立子窗口的g通道亮度总和信息。有效比特数：32bit。
hw_ae_sumBayerB_val	独立子窗口的b通道亮度总和信息。有效比特数：29bit。

### 【注意】

- 如需得到独立子窗口的亮度均值信息，需要自行软件计算，用亮度总和除以子窗口中的像素数总和
- 3576平台不支持设置独立子窗口，该统计信息无效

## aeStats\_histStats\_t

### 【说明】

基于raw图的直方图统计信息

### 【定义】

```
typedef struct aeStats_histStats_s {
    unsigned int hw_ae_histBin_val[AESTATS_HIST_BIN_NUM];
} aeStats_histStats_t;
```

## 【成员】

成员名称	描述
hw_ae_histBin_val	直方图的分段，共256段，每个分段内有效bit数：28bit

## AWB统计数据类型

### awbStats\_stats\_t

#### 【说明】

定义白平衡硬件统计信息，主要包含白点统计信息和分块统计信息

- 白点统计信息

记录主窗口内4个光源2种大小的白区里的RGin,BGain累加值及个数；

4个UV或XY域附加框里的RGin,BGain累加值及个数；

白点亮度直方图；

- 分块统计信息

图像15x15分块(如AWB分块示意图所示)，每个块所有点或白点的R,G,B均值



图 AWB分块示意图

#### 【定义】

```
typedef struct awbStats_wpEngineStats_s {
    awbStats_wpStats_t norWp[AWBSTATS_WPDCT_LS_NUM];
    awbStats_wpStats_t bigWp[AWBSTATS_WPDCT_LS_NUM];
    uint32_t hw_awbCfg_wpXyUvSpcRaw_cnt[AWBSTATS_WPDCT_LS_NUM];
    uint32_t hw_awb_wpHistBin_val[AWBSTATS_WP_HIST_BIN_NUM];
} awbStats_wpEngineStats_t;

typedef struct awbStats_wpfiltEngineStats_s {
    awbStats_wpStats_t fltPix[AWBSTATS_WPFLTOUTFULL_ENTITY_NUM];
} awbStats_wpfiltEngineStats_t;

typedef struct awbStats_pixEngineStats_s {
    awbStats_pixStats_t zonePix[AWBSTATS_ZONE_15x15_NUM];
} awbStats_pixEngineStats_t;

typedef struct awbStats_stats_s {
    awbStats_wpEngineStats_t wpEngine;
    awbStats_wpfiltEngineStats_t wpFltOutFullEngine;
    awbStats_pixEngineStats_t pixEngine;
} awbStats_stats_t;
```

## 【成员】

成员名称	描述
wpEngine	主窗口下不同光源下的白点统计结果
wpEngine.norWp	主窗口下xy中框统计的白点统计结果
wpEngine.bigWp	主窗口下xy大框统计的白点统计结果
wpEngine.hw_awb_wpHistBin_val	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。wpEngine.norWp
wpEngine.wpEngine.norWp	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。
pixEngine	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，如AWB 分块示意图所示
wpFltOutFullEngine.fltPix	落在extraWpRange区域里的点的统计结果（只会记录 wpRegionSet前四个区域），最多4个区域

### 【注意事项】

如果用户希望获取主窗口全局的白点统计结果，根据所有光源下的白点统计结果可以简单换算得到。

### awbStats\_wpStats\_t

#### 【说明】

定义某个光源某个大小的XY框下的白点统计结果，或非白点区域里的统计结果

#### 【定义】

```
typedef struct awbStats_wpStats_s {
    uint32_t hw_awbCfg_statsWp_count;
    uint32_t hw_awbCfg_rGainSum_val;
    uint32_t hw_awbCfg_bGainSum_val;
} awbStats_wpStats_t;
```

#### 【成员】

成员名称	描述
hw_awbCfg_statsWp_count	白点数量，22bit整数位，10bit小数位。其中 WpNo = ISP硬件 统计白点数 / 2^10.
hw_awbCfg_rGainSum_val	白点R通道的累加和，22bit整数位，10bit小数位
hw_awbCfg_bGainSum_val	白点G通道的累加和，22bit整数位，10bit小数位

### awbStats\_pixStats\_t

#### 【说明】

定义每个块的统计结果。

若hw\_awbCfg\_zoneStatsSrc\_mode == awbStats\_pixAll\_mode, 记录的结果为块内所有点的累加和；

若hw\_awbCfg\_zoneStatsSrc\_mode ==awbStats\_norWpAll\_mode, 记录的结果为块内所有中框白点的累加和；

若hw\_awbCfg\_zoneStatsSrc\_mode ==awbStats\_bigWpAll\_mode, 记录的结果为块内所有大框白点的累加和；

其他模式说明参考文档《Rockchip\_Color\_Optimization\_Guide》中章节“hw\_awbCfg\_zoneStatsSrc\_mode”

### 【定义】

```
typedef struct awbStats_pixStats_s {
    uint32_t hw_awbCfg_statsPix_count;
    uint32_t hw_awbCfg_rSum_val;
    uint32_t hw_awbCfg_gSum_val;
    uint32_t hw_awbCfg_bSum_val;
} awbStats_pixStats_t;
```

### 【成员】

成员名称	描述
hw_awbCfg_statsPix_count	块内点的个数
hw_awbCfg_rSum_val	块内所有点R通道的累加和
hw_awbCfg_gSum_val	块内所有点RG通道的累加和
hw_awbCfg_bSum_val	块内所有点RB通道的累加和

## afStats\_stats\_t

### 【说明】

定义AF统计信息

### 【定义】

```
typedef struct afStats_mainWinStats_s {
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_hFilt1Fv_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(0),
    M4_NOTES(H1 fv value\nFreq of use: high)) */
    uint32_t hw_af_hFilt1Fv_val[AFSTATS_ZONE_NUM];
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_hFilt2Fv_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
```

```

M4_ORDER(1),
M4_NOTES(H2 fv value\nFreq of use: high) */
uint32_t hw_af_hFilt2Fv_val[AFSTATS_ZONE_NUM];
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_vFilt1Fv_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(2),
    M4_NOTES(V1 fv value\nFreq of use: high) */
uint32_t hw_af_vFilt1Fv_val[AFSTATS_ZONE_NUM];
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_vFilt2Fv_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(3),
    M4_NOTES(V2 fv value\nFreq of use: high) */
uint32_t hw_af_vFilt2Fv_val[AFSTATS_ZONE_NUM];
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_luma_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(4),
    M4_NOTES(Luma value.\nFreq of use: high) */
uint32_t hw_af_luma_val[AFSTATS_ZONE_NUM];
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_hLumaCnt_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),
    M4_ORDER(5),
    M4_NOTES(High light value.\nFreq of use: high) */
uint32_t hw_af_hLumaCnt_val[AFSTATS_ZONE_NUM];
} afStats_mainWinStats_t;

typedef struct afStats_subWinStats_s {
/* M4_GENERIC_DESC(
    M4_ALIAS(hw_af_hFilt1Fv_val),
    M4_TYPE(u32),
    M4_SIZE_EX(1,1),
    M4_RANGE_EX(0, 0xFFFFFFFF),
    M4_DEFAULT(0),
    M4_HIDE_EX(0),
    M4_RO(0),

```

```
M4_ORDER(0),
M4_NOTES(H1 fv value\nFreq of use: high)) */
uint32_t hw_af_hFilt1Fv_val;
/* M4_GENERIC_DESC(
M4_ALIAS(hw_af_hFilt2Fv_val),
M4_TYPE(u32),
M4_SIZE_EX(1,1),
M4_RANGE_EX(0, 0xFFFFFFFF),
M4_DEFAULT(0),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(1),
M4_NOTES(H2 fv value\nFreq of use: high)) */
uint32_t hw_af_hFilt2Fv_val;
/* M4_GENERIC_DESC(
M4_ALIAS(hw_af_vFilt1Fv_val),
M4_TYPE(u32),
M4_SIZE_EX(1,1),
M4_RANGE_EX(0, 0xFFFFFFFF),
M4_DEFAULT(0),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(2),
M4_NOTES(V1 fv value\nFreq of use: high)) */
uint32_t hw_af_vFilt1Fv_val;
/* M4_GENERIC_DESC(
M4_ALIAS(hw_af_vFilt2Fv_val),
M4_TYPE(u32),
M4_SIZE_EX(1,1),
M4_RANGE_EX(0, 0xFFFFFFFF),
M4_DEFAULT(0),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(3),
M4_NOTES(V2 fv value\nFreq of use: high)) */
uint32_t hw_af_vFilt2Fv_val;
/* M4_GENERIC_DESC(
M4_ALIAS(hw_af_luma_val),
M4_TYPE(u32),
M4_SIZE_EX(1,1),
M4_RANGE_EX(0, 0xFFFFFFFF),
M4_DEFAULT(0),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(4),
M4_NOTES(Luma value.\nFreq of use: high)) */
uint32_t hw_af_luma_val;
/* M4_GENERIC_DESC(
M4_ALIAS(hw_af_hLumaCnt_val),
M4_TYPE(u32),
M4_SIZE_EX(1,1),
M4_RANGE_EX(0, 0xFFFF),
M4_DEFAULT(0),
M4_HIDE_EX(0),
M4_RO(0),
M4_ORDER(5),
M4_NOTES(High light value.\nFreq of use: high)) */
uint32_t hw_af_hLumaCnt_val;
```

```

} afStats_subWinStats_t;

typedef struct afStats_stats_s {
    /* M4_GENERIC_DESC(
        M4_ALIAS(statsWinA),
        M4_TYPE(struct),
        M4_UI_MODULE(normal_ui_style),
        M4_HIDE_EX(0),
        M4_RO(0),
        M4_ORDER(0),
        M4_NOTES(Window A stats)) */
    afStats_mainWinStats_t mainWin;
    /* M4_GENERIC_DESC(
        M4_ALIAS(statsWinB),
        M4_TYPE(struct),
        M4_UI_MODULE(normal_ui_style),
        M4_HIDE_EX(0),
        M4_RO(0),
        M4_ORDER(1),
        M4_NOTES(Window B stats)) */
    afStats_subWinStats_t subWin;
} afStats_stats_t;

```

### 【成员】

成员名称	描述
mainWin.hw_af_hFilt1Fv_val	主窗口下15*15子窗口的H1清晰度值
mainWin.hw_af_hFilt2Fv_val	主窗口下15*15子窗口的H2清晰度值
mainWin.hw_af_vFilt1Fv_val	主窗口下15*15子窗口的V1清晰度值
mainWin.hw_af_vFilt2Fv_val	主窗口下15*15子窗口的V2清晰度值
mainWin.hw_af_luma_val	主窗口下15*15子窗口的亮度值
mainWin.hw_af_hLumaCnt_val	主窗口下15*15子窗口的高亮计数值
subWin.hw_af_hFilt1Fv_val	独立窗口的H1清晰度值
subWin.hw_af_hFilt2Fv_val	独立窗口的H2清晰度值
subWin.hw_af_vFilt1Fv_val	独立窗口的V1清晰度值
subWin.hw_af_vFilt2Fv_val	独立窗口的V2清晰度值
subWin.hw_af_luma_val	独立窗口的亮度值
subWin.hw_af_hLumaCnt_val	独立窗口的高亮计数值

##

## ISP效果参数封装

### JSON-IQ与场景切换

在最新的**RKAIQ**中引入了多场景、差分**IQ**文件支持，使得用户可以在一份**IQ**文件中实现多组图像效果参数，配合**RKAIQ**提供的**UAPI**接口做到无缝切换。

## IQ文件格式说明

下面是一个标准的**IQ**文件**JSON**层级视图，其中**main\_scene**为顶级数组，**sub\_scene**为**main\_scene**的子级，核心的**IQ**参数保存在每一个**scene\_isp39**结构中，当需要切换参数的时候，我们只需要指定参数所在的**main\_scene**的名称和**sub\_scene**的名称，程序即可找到对应的参数并完成切换。

```
▼ object {7}
  ► sensor_calib {10}
  ► module_calib {1}
  ▼ main_scene [2]
    ▼ 0 {3}
      name : normal
      ▼ sub_scene [2]
        ▼ 0 {2}
          name : day
          ► scene_isp32 {29}
        ▼ 1 {2}
          name : night
          ▼ scene_isp32 {4}
            ► colorAsGrey {1}
            ► sharp_v33 {2}
            ▼ csm {1}
              ► TuningPara {5}
            ► cgcc {1}
          sub_scene_len : 2
        ▼ 1 {3}
          name : hdr
          ► sub_scene [2]
          sub_scene_len : 2
        main_scene_len : 2
  ► uapi [0]
  uapi_len : 0
  ► sys_static_cfg {1}
```

### 模块说明：

OBJECT名称	说明	备注
<b>sensor_calib</b>	与SENSOR型号相关的参数	多个场景共用
<b>module_calib</b>	与模组型号相关的参数	多个场景共用
<b>uapi</b>	调试相关结构	用户无需关心
<b>sys_static_cfg</b>	调试相关结构	用户无需关心
<b>main_scene</b>	主场景集合，该结构体对应的数据类型为数组	如上图，包含两个主场景normal和HDR
<b>main_scene_len</b>	主场景个数， <b>main_scene</b> 数组中的成员个数	
<b>sub_scene</b>	子场景集合， <b>main_scene</b> 的成员	如上图，normal主场景下包含day和night两个子场景
<b>sub_scene_len</b>	主场景个数， <b>sub_scene</b> 数组中的成员个数	
<b>scene_isp32</b>	场景的最小单元，包含 <b>AE</b> 等ISP模块参数	

## 差分参数特性：

由于目前单个IQ文件体积已经超过100KB，当需要的场景越来越多时，IQ文件的体积将成倍增加，对于存储空间有限的设备来说并不友好，于是RKAIQ引入了差分参数的支持，使得在使用多套参数的时候，只需要保留每个场景中与基础场景的差异部分即可，从而节省大量空间。

例如我们有以下A、B两个场景：

场景A

```
{
  "name": "day",
  "ae_calib": {
    "enable": true,
    "value": 1.4
  },
  "af_calib": {
    "enable": true,
    "value": 1.4
  },
  "awb_calib": {
    "enable": true,
    "value": 1.4
  },
  "colorAsGrey": {
    "enable": false,
    "value": 1.4
  }
}
```

场景B

```
{  
    "name": "night",  
    "ae_calib": {  
        "enable": true,  
        "value": 1.2  
    },  
    "af_calib": {  
        "enable": true,  
        "value": 1.4  
    },  
    "awb_calib": {  
        "enable": false,  
        "value": 1.4  
    },  
    "colorAsGrey": {  
        "enable": true,  
        "value": 1.4  
    }  
}
```

以day为基础场景，经过分析场景B与场景A的差异，仅在场景B中保留差异部分，则场景B可以改写为：

场景B

```
{  
    "name": "night",  
    "ae_calib": {  
        "value": 1.2  
    },  
    "awb_calib": {  
        "enable": false  
    },  
    "colorAsGrey": {  
        "enable": true  
    }  
}
```

以此类推，多场景IQ文件在既保证JSON格式完整性的情况下减少了体积占用。

需要注意的是：当我们的差异部分为数组元素的时候，需要保证同级的元素都被保留，且顺序保持一致，否则可能出现错误。

### 编程接口：

在RKAIQ初始化完成后，我们可以通过调用以下接口来完成场景切换：

```
int rk_aiq_uapi_sysctl_switch_scene(const rk_aiq_sys_ctx_t* sys_ctx,  
                                     const char* main_scene,  
                                     const char* sub_scene)
```

接口描述：

参数名称	说明	类型
<b>sys_ctx</b>	RKAIQ实例指针	指针
<b>main_scene</b>	主场景名称	字符串
<b>sub_scene</b>	子场景名称	字符串

## JSON-IQ转BINARY-IQ

### 概述

- JSON转BIN方案旨在为用户提供一个体积小且快速加载IQ的方案，推荐使用在小存储单元和时间敏感类应用场景中；
- JSON与BIN同时存在的情况下，JSON的优先级高于BIN；
- JSON转BIN工具需要和AIQ版本一一对应，版本不同将导致无法预估的错误；
- IQ调试和Calib释放相关逻辑依赖J2S框架，当选择仅支持BIN模式，某些特殊功能不支持；

J2S Parser	BIN Parser	在线调试	场景切换	在线更新IQ
ENABLE	ENABLE	Y	Y	Y
ENABLE	DISABLE	Y	Y	Y
DISABLE	ENABLE	N	Y	N
DISABLE	DISABLE	N	Y	N

### 转换工具使用说明

转换工具源码路径：`camera_engine_rkaiq/tool/j2s4b`

#### 编译：

```
# 进入转换工具源码路径
cd tool/j2s4b/
# 执行转换工具编译脚本
./build.sh
```

编译完成将在`output/j2s4b`路径生成转换工具

#### 转换说明：

```
output/j2s4b <input.json> <output.bin>
```

#### 参数说明

参数名称	说明	备注
<b>input.json</b>	输入的 <b>JSON-IQ</b> 文件	必须是完整参数的 <b>IQ</b> 文件，不能含有差分参数，否则运行时会导致严重错误
<b>output.bin</b>	输出的 <b>BINARY-IQ</b> 文件	需要和 <b>RKAIQ</b> 版本一一对应，否则运行时会导致严重错误

转换完成工后工具会自行将转换输出与原输入JSON进行校验，校验一致的话，会输出**YES**标识  
将转换完成的bin文件重命名，文件名除后缀为.bin以外其它与原输入JSON一致  
将输出的BIN文件放到设备IQ目录中，并将同目录下同名的JSON文件删除/备份  
如果输入的**IQ**文件中包含多个完整的场景参数，则转换完后依然可以调用**UAPI**来实现场景切换。

## SmartIr

### 概述

实现软光敏功能，为应用提供日夜判断结果。

### 功能描述

SmartIr是通过 AE、AWB统计来计算环境的亮度以及环境的红外光占比，根据这些信息综合判断是否进行日夜切换。SmartIr 是基于AIQ的3A统计值和相关API实现，编译后生成独立的SmartIr库，独立于AIQ库。

### API参考

#### rk\_smart\_ir\_init

##### 【描述】

初始化 SmartIr 运行环境。

##### 【语法】

```
rk_smart_ir_ctx_t* rk_smart_ir_init(const rk_aiq_sys_ctx_t* aiq_ctx);
```

##### 【参数】

参数名称	描述	输入/输出
aiq_ctx	AIQ上下文指针	输入

##### 【返回值】

返回值	描述
非NULL	成功，返回对应于 AIQ ctx 的 SmartIr ctx
NULL	失败

##### 【需求】

- 头文件：rk\_smart\_ir\_api.h
- 库文件：libsmartIr.so librkaiq.so

#### rk\_smart\_ir\_deinit

##### 【描述】

反初始化 SmartIr。

## 【语法】

```
XCamReturn rk_smart_ir_deinit(const rk_smart_ir_ctx_t* ctx);
```

## 【参数】

参数名称	描述	输入/输出
ctx	smartIr 上下文指针	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码说明

## 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartIr.so librkaiq.so

## **rk\_smart\_ir\_setAttr**

### 【描述】

配置smartIr参数。

### 【语法】

```
XCamReturn rk_smart_ir_setAttr(rk_smart_ir_ctx_t* ctx, rk_smart_ir_attr_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	smartIr 上下文指针	输入
attr	配置日夜判断参数	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码说明

## 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartIr.so librkaiq.so

## **rk\_smart\_ir\_getAttr**

## 【描述】

获取smartlr配置参数。

## 【语法】

```
XCamReturn rk_smart_ir_getAttr(rk_smart_ir_ctx_t* ctx, rk_smart_ir_attr_t* attr);
```

## 【参数】

参数名称	描述	输入/输出
ctx	smartlr 上下文指针	输入
attr	获取日夜判断参数	输出

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码说明

## 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

## **rk\_smart\_ir\_queryInfo**

### 【描述】

获取smartlr信息。

### 【语法】

```
XCamReturn rk_smart_ir_queryInfo(rk_smart_ir_ctx_t* ctx, rk_smart_ir_query_info_t* query_info);
```

### 【参数】

参数名称	描述	输入/输出
ctx	smartlr 上下文指针	输入
query_info	获取smartlr信息	输出

### 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码说明

### 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

## **rk\_smart\_ir\_runOnce**

### **【描述】**

执行日夜判断流程。

### **【语法】**

```
XCamReturn rk_smart_ir_runOnce(rk_smart_ir_ctx_t* ctx, rk_aiq_isp_stats_t* stats_ref,
rk_smart_ir_result_t* result);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	SmartIr 上下文指针	输入
stats_ref	3A 统计信息	输入
result	日夜判断结果	输出

### **【返回值】**

返回值	描述
0	成功
非0	失败，详见错误码说明

### **【需求】**

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

## **rk\_smart\_ir\_groupRunOnce**

### **【描述】**

group camera执行日夜判断流程。

### **【语法】**

```
XCamReturn rk_smart_ir_groupRunOnce(rk_smart_ir_ctx_t* ctx, rk_aiq_isp_stats_t** grp_stats, int
cam_num, rk_smart_ir_result_t* result);
```

### **【参数】**

参数名称	描述	输入/输出
ctx	SmartIr 上下文指针	输入
grp_stats	各个camera的3A 统计信息	输入
cam_num	group camera个数	输入
result	日夜判断结果	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败， 详见错误码说明

#### 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

### **rk\_smart\_ir\_config**

#### 【描述】

配置参数。 (smartlr v2.0.0开始不再使用)

#### 【语法】

```
XCamReturn rk_smart_ir_config(rk_smart_ir_ctx_t* ctx, rk_smart_ir_params_t* config);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	SmartIr 上下文指针	输入
config	配置日夜判断参数	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败， 详见错误码说明

#### 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

### **rk\_smart\_ir\_set\_status**

#### 【描述】

初始化 smartIr 日夜状态。 (smartIr v2.0.0开始不再使用)

#### 【语法】

```
XCamReturn rk_smart_ir_set_status(rk_smart_ir_ctx_t* ctx, rk_smart_ir_result_t result);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	SmartIr 上下文指针	输入
result	配置日夜状态	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败， 详见错误码说明

#### 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartIr.so librkaiq.so

### **rk\_smart\_ir\_auto\_irled**

#### 【描述】

IR补光灯亮度自动控制，仅在IR补光灯支持亮度可调（设置PWM占空比）才有效。该API会调用AE API进行曝光控制。 (smartIr v2.0.0开始不再使用)

#### 【语法】

```
XCamReturn rk_smart_ir_auto_irled(rk_smart_ir_ctx_t* ctx, rk_smart_ir_autoled_t* auto_irled);
```

#### 【参数】

参数名称	描述	输入/输出
ctx	SmartIr 上下文指针	输入
auto_irled	红外补光灯信息	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败， 详见错误码说明

#### 【需求】

- 头文件: rk\_smart\_ir\_api.h
- 库文件: libsmartlr.so librkaiq.so

## 数据类型

### 枚举型数据说明

```
// 日夜状态类型
typedef enum RK_SMART_IR_STATUS_e {
    RK_SMART_IR_STATUS_INVALID      = 0,
    RK_SMART_IR_STATUS_DAY         = 1, /*状态为白天*/
    RK_SMART_IR_STATUS_NIGHT       = 2, /*状态为黑夜*/
    RK_SMART_IR_STATUS_MAX
} RK_SMART_IR_STATUS_t;

// 日夜转化模式
typedef enum RK_SMART_IR_SWTICH_MODE_e {
    RK_SMART_IR_SWITCH_MODE_INVALID  = 0, /*无效模式*/
    RK_SMART_IR_SWITCH_MODE_AUTO    = 1, /*自动模式，补光灯会根据环境照度自动开启或关闭*/
    RK_SMART_IR_SWITCH_MODE_DAY     = 2, /*白天模式，补光灯不会开启（图像为彩色）*/
    RK_SMART_IR_SWITCH_MODE_NIGHT   = 3, /*夜晚模式，红外或白光灯开启（若选择红外模式则图像为黑白，若选择可见光模式则图像为彩色）*/
    RK_SMART_IR_SWITCH_MODE_TIME    = 4, /*定时模式，设置时间段内为补光灯不开启时间段（暂未使用）*/
    RK_SMART_IR_SWITCH_MODE_MAX
} RK_SMART_IR_SWTICH_MODE_t;

// 补光灯光源类型
typedef enum RK_SMART_IR_LIGHT_TYPE_e {
    RK_SMART_IR_LIGHT_TYPE_INVALID  = 0, /*无效模式*/
    RK_SMART_IR_LIGHT_TYPE_VIS       = 1, /*可见光模式，开启白光灯或者暖光灯*/
    RK_SMART_IR_LIGHT_TYPE_IR        = 2, /*红外模式，开启红外灯*/
    RK_SMART_IR_LIGHT_TYPE_NONE      = 3, /*关闭模式，没有补光灯或者不开补光灯*/
    RK_SMART_IR_LIGHT_TYPE_MIX       = 4, /*混合模式，白光灯和红外灯同时开启（暂未使用）*/
    RK_SMART_IR_LIGHT_TYPE_MAX
} RK_SMART_IR_LIGHT_TYPE_t;

// 补光灯亮度调节模式
typedef enum RK_SMART_IR_LIGHT_MODE_e {
    RK_SMART_IR_LIGHT_MODE_INVALID  = 0, /*无效模式*/
    RK_SMART_IR_LIGHT_MODE_AUTO     = 1, /*自动模式，设置成自动模式后，可填写一个亮度数值，补光灯亮度会根据环境照度自动调节亮度，但不会高于设置的最大值*/
    RK_SMART_IR_LIGHT_MODE_MANUAL   = 2, /*手动模式，设置成手动模式后，可填写一个亮度数值，补光灯亮度不会根据环境照度调节，一直是这个固定的亮度值*/
    RK_SMART_IR_LIGHT_MODE_MAX
} RK_SMART_IR_LIGHT_MODE_t;
```

### **rk\_smart\_ir\_params\_t**

#### 【说明】

算法参数配置。

#### 【定义】

```

typedef struct rk_smart_ir_params_s {
    float d2n_envL_th;
    float n2d_envL_th;
    float rggain_base;
    float bggain_base;
    float awbgain_rad;
    float awbgain_dis;
    uint16_t switch_cnts_th;
} rk_smart_ir_params_t;

```

### 【成员】

成员名称	描述
d2n_envL_th	日转夜亮度阈值，计算公式为：envL = Raw域AE亮度统计值 / (曝光时间 × 增益)，曝光时间单位为毫秒，增益为倍数。
n2d_envL_th	夜转日亮度阈值，计算公式为：envL = Raw域AE亮度统计值 / (曝光时间 × 增益)，曝光时间单位为毫秒，增益为倍数。
rggain_base	黑夜切白天的Rgain/Ggain基准值，默认值为1。
bggain_base	黑夜切白天的Bgain/Ggain基准值，默认值为1。
awbgain_rad	黑夜切白天的awbgain滤波半径，默认值为0。
awbgain_dis	黑夜切白天的awbgain离散度阈值，需要调试。
switch_cnts_th	切换阈值，保持相同状态次数大于该阈值时才允许状态切换。

## rk\_smart\_ir\_attr\_t

### 【说明】

模式选择和算法参数配置。

### 【定义】

```

typedef struct rk_smart_ir_attr_s {
    RK_SMART_IR_STATUS_t    init_status;
    RK_SMART_IR_SWITCH_MODE_t  switch_mode;
    RK_SMART_IR_LIGHT_MODE_t   light_mode;
    RK_SMART_IR_LIGHT_TYPE_t   light_type;
    uint8_t                  light_value;
    rk_smart_ir_params_t     params;
    bool                     en_quick_switch;
    bool                     en_grid_weight;
    bool                     en_auto_n2dth;
} rk_smart_ir_attr_t;

```

### 【成员】

成员名称	描述
init_status	配置日夜状态，可以设置初始的smartIr日夜状态，或者在smartIr运行过程中改变内部日夜状态。
switch_mode	日夜转化模式，包括自动切换、固定白天、固定黑夜、定时切换四种模式。
light_mode	补光灯亮度调节模式，包括自动亮度补光模式和固定亮度补光模式。
light_type	补光灯光源类型，包括可见光、红外光、无补光、混合光四种类型。
light_value	补光灯亮度值。当补光灯为手动模式时，配置light_value为固定的补光灯亮度；当补光灯为自动模式时，补光灯的最大亮度不会超过light_value。补光灯亮度由PWM占空比控制，用于控制补光灯的强度，取值范围为0~100，步进为1，默认值为100。
params	日夜切换的配置参数。
en_quick_switch	是否加快日夜转化，对转化速度有要求的可以将此参数配置为true。值得注意的是，切换速度提升了，相应的误切概率也会增加。
en_grid_weight	是否启用带分块权重的日夜转化，该权重值与AE模块中的GridWeights参数是一致的。
en_auto_n2dth	补光灯类型配置为自动，黑夜切白天时，是否自动调整的黑切白阈值。true为自动调整，false为不调整。

## rk\_smart\_ir\_result\_t

### 【说明】

smartIr结果参数。

### 【定义】

```
typedef struct rk_smart_ir_result_s {
    RK_SMART_IR_STATUS_t status;
    bool      gray_on;
    float     fill_value;
} rk_smart_ir_result_t;
```

### 【成员】

成员名称	描述
status	白天或者黑夜的状态。
gray_on	图像为黑白或者彩色，true为黑白，false为彩色。
fill_value	需要调节的补光灯亮度，补光灯亮度由PWM占空比控制，取值范围为0~100，步进为0.1。

## rk\_smart\_ir\_query\_info\_t

### 【说明】

smartIr信息查询。

### 【定义】

```

typedef struct rk_smart_ir_query_info_s {
    float rggain;
    float bggain;
    float ir_strength;
} rk_smart_ir_query_info_t;

```

### 【成员】

成员名称	描述
rggain	Rgain/Ggain。
bggain	Bgain/Ggain。
ir_strength	当前环境的红外强度，取值范围为0~1，该值越大表示红外占比越大。

## rk\_smart\_ir\_autoled\_t

### 【说明】

IR补光灯亮度自动控制信息。

### 【定义】

```

typedef struct rk_smart_ir_autoled_s {
    bool is_smooth_convert;
    int auto_irled_val;
    int auto_irled_min;
    int auto_irled_max;
} rk_smart_ir_autoled_t;

```

### 【成员】

成员名称	描述
is_smooth_convert	快速启动初始状态是黑夜模式的情况下，需将此参数配置为true，实现平滑过渡。
auto_irled_val	黑夜模式下，自动红外补光灯亮度值，归一化至0~100%，单位为百分比，取值范围为 (auto_irled_min, auto_irled_max)，默认值为100。
auto_irled_min	黑夜模式下，自动红外补光灯最小值，归一化至0~100%，单位为百分比，取值范围为 (1,100)，默认值为10。该值不能设置过小，否则线性度太差导致画面来回闪烁。
auto_irled_max	黑夜模式下，自动红外补光灯最大值，归一化至0~100%，单位为百分比，取值范围为 (1,100)，默认值为100。

## 功能集成

### 参考

YOUR\_SDK/media/isplisp/camera\_engine\_rkaiq/rkisp\_demo/demo/sample/sample\_smalrlr.cpp

## 参数调试

由于各设备使用的红外灯及Sensor差异，需要对 `rk_smart_ir_params_t` 中的配置参数做调试才能达到比较理想的效果。调试步骤如下：（1）红外光白平衡参数标定

- 启动预览，将红外灯打开，ircutter关闭，并将设备置于无可见光环境；
  - 按照sample\_smartiR中的calibration模式，获得sensor在无可见光时的RGgain及BGgain；
  - 根据获得的 RGgain 及 BGgain 均值，确定 rggain\_base、bggain\_base 基准值，一般无可见光环境下可设置为 0.9~1.1 左右，可根据实际情况调整；
  - awbgain\_rad 为 awbgain 的滤波参数，该值越大过滤的范围越宽，黑夜转白天就越不容易，参考范围为 0~1.0；
  - awbgain\_dis 为 awbgain 的离散度阈值，衡量数据分布离散程度，该值越大允许的离散程度越大，黑夜转白天就越不容易。

## (2) 亮度切换阈值设置

假设 sensor 限定的最大曝光为 30 毫秒，增益为 32 倍，预设的目标亮度为 40（基于8bit），那么：

- d2n\_envL\_th = 40 / (30 \* 32) 约为0.04，此时，假定目标亮度为40（基于8bit），曝光量为最大时，白天切成黑夜；
  - n2d\_envL\_th = 12 / (15 \* 4) 约为0.2，此时，假定等同于目标亮度为12（基于8bit），曝光量为15毫秒 x 4倍，黑夜切白天；
  - 可通过 API rk\_aiq\_user\_api2\_ae\_queryExpResInfo 查询当前环境亮度，该 API 返回的环境亮度基于8bit亮度统计计算得来。

### (3) 灵敏度控制

`switch_cnts_th` 可用于控制切换灵敏度，可根据帧率将该值换算成时间。公式为：状态保持时长 (ms) =  $1 / \text{fps} \times 1000 \times \text{switch\_cnts\_th}$ ，假设帧率为30fps，探测到日夜切换时需要保持100次以上时才允许切换，那切换延时大概在3.3 秒。

AIISP

ISP39支持AIISP降噪

## 功能描述

AIISP是将ISP的中间RAW数据写入DDR，然后通过CPU和NN的配合，使用软件AINR算法将DDR的数据进行AI运算，

具备了RAW域图像2D去噪功能，在高ISO场景具备比传统ISP（bayerNR，Ynr，Cnr）更强的去噪能力，能够更好的保留细节，去除噪声。

# 约束

- 与HDR功能、predgain 等互斥
  - isp必须工作在离线模式

API参考

## rk\_aiq\_uapi2\_sysctl\_initAiisp

## 【描述】

## 设置aiisp初始化信息

## 【语法】

## XCamReturn

rk\_aiq\_uapi2\_sysctl\_initAiisp(rk\_aiq\_sys\_ctx\_t\* sys\_ctx, rk\_aiq\_aiisp\_cfg\_t\* aiisp\_cfg, rk\_aiq\_aiisp\_chaiisp\_ch):

## 【参数】

参数名称	描述	输入/输出
sys_ctx	AIQ上下文指针	输入
aiisp_cfg	保留，设置为 NUL	输入
aiisp_cb	保留，设置为 NUL	输入

## 【返回值】

返回值	描述
0	成功
非0	失败，详见错误码表

## 【需求】

- 头文件：rk\_aiq\_user\_api2\_sysctl.h
- 库文件：librkaiq.so

## 【注意】

- 需要在 rk\_aiq\_uapi2\_sysctl\_ini 后，rk\_aiq\_uapi2\_sysctl\_prepare 前调用
- 与HDR功能、predgain 等互斥
- isp必须工作在离线模式
- 依赖aiisp算法库，不同型号sensor需要单独调试 aiisp 算法效果

## log解读

在跑应用之前，打开smartIir log动态开关，命令如下：

```
(Linux)  
export persist_camera_engine_log=0x800000000  
(Android)  
setprop persist.vendor.rkisp.log 0x800000000
```

其中一帧的log信息如下：

Line1：

```
[SMARTIR] Switch: FraID=2750,AEConv=1,Luma=2.0992,Exp=  
[0.0398,128.0000],AllDis=0.0390,PartDis=0.1018,Blk=[0,0,225],Cnts=0,Cur=2
```

当前帧的日夜切换信息。

成员名称	描述
FraID	当前帧的帧号。
AEConv	当前帧曝光是否已经收敛。0：曝光未收敛；1：曝光已收敛。
Luma	当前帧对应的Raw图亮度。
Exp	当前帧对应的sensor曝光时间值和增益值。
AllDis	当前帧对应的所有block的R/Gain和B/Gain离散值。
PartDis	当前帧对应的滤波后的R/Gain和B/Gain离散值。
Blk	统计属于keep/day/night的block数量。白天状态下，当night block大于180后切到黑夜状态；黑夜状态下，当day block大于135后切到白天。
Cnts	切换时，保持相同状态的次数。
Cur	当前白天/黑夜状态信息，1：白天，2：黑夜。

Line2:

```
[SMARTIR]
AutoLed:LumaDeviation=-0.9300,IsInit=1,ConvDone=1,CtrlPrior=2,LedPos=2,PosCnt=0,LedVal=100
```

黑夜状态下，开启补光灯亮度自动控制。

成员名称	描述
LumaDeviation	当前均值亮度与目标亮度的亮度差值比。
IsInit	黑夜状态下补光灯亮度自动控制初始化完成标志。0：未初始化；1：完成初始化。
ConvDone	黑夜状态下场景识别完成标志。0：未完成；1：完成。
CtrlPrior	当前帧对应的曝光控制优先级。0：无效；1：曝光时间优先；2：自动补光优先；3：曝光增益优先；4：黑切白优先。
LedPos	补光灯当前的状态。-2：最小亮度值；-1：减小亮度值；0：亮度值保持不变；1：增大亮度值；2：最大值亮度。
PosCnt	补光灯亮度值保持相同状态的标志。
LedVal	输出补光灯的亮度值。

## Debug & FAQ

### 如何获取版本号

aiq提供了版本发布日期、aiq版本、iq解析器版本及isp各个算法模块的版本信息；

[获取简略版本信息](#)

```
strings librkaiq.so | grep -w AIQ
AIQ v6.0x6.3-rc2
```

## 获取完整版本信息

1. SDK中aiq库默认编译为Release版本，需要改成Debug版本，重新编译aiq库后更新到设备。

```
SDK/external/camera_engine_rkaiq/CMakeLists.txt:
```

```
8
9 if(NOT CMAKE_BUILD_TYPE STREQUAL "Release")
10 add_definitions(-DBUILD_TYPE_DEBUG)
11 endif()
```

改成：

```
8
9 #if(NOT CMAKE_BUILD_TYPE STREQUAL "Release")
10 add_definitions(-DBUILD_TYPE_DEBUG)
11 #endif()
```

2. 默认打印级别下，加载运行的aiq库不会打印，可以设置xcore模块的log级别，以打印aiq版本信息：

```
export persist_camera_engine_log=0x1000000ff2
```

3. aiq启动时打印版本信息如下所示：

```
***** VERSION INFOS *****
version release date: 2024-07-18
    AIQ: AIQ v6.0x6.3-rc2
git logs:
8532661 docs: add isp33/isp39 developement docs
ef2a5ba Support for easy compilation of each SOC platform
957a3e8c AE: OverExp max_wgt support str_bias
3cc2426 uapi helper: fix ae set exp attr.
aa4daed algos: ldch: remove some unused code

***** VERSION INFOS END *****
```

## 版本号匹配规则说明

AIQ与IQ Tool、ISP Driver的版本匹配规则如下：

v A . B . C

其中B为16进制表示，bit[0:3] 标识 AIQ与IQ Tool的匹配版本，bit[4:7]标识AIQ与ISP driver的匹配版本，例如：

ISP driver: v 1.0x3.0 与 AIQ: v1.0x30.0匹配，与AIQ: v1.0x40.0不匹配。

IQ tool: v 1.0x3.0 与 AIQ: v1.0x33.0匹配，与AIQ: v1.0x30.0不匹配，其中AIQ 版本号C不为0，有可能出现版本不匹配的情况，针对IQ Tool匹配建议优先采用C版本号为0的AIQ版本。

# AIQ Log

## 概述

aiq采用64bits表示所有模块的log级别，表示各个模块的位图及说明如下：

bit: [63-39] 38 37 36 35 34 33 32 31  
mean: [U] [CAMHW][ANALYZER][XCORE][ASD][ACGC][AFEC][AORB][ASHARP]

bit: 30 29 28 27 26 25 24 23 22  
mean:[AIE][ACP][ACSM][ALDCH][A3DLUT][ADEHAZE][AWDR][AGAMMA][ACCM]

bit: 21 20 19 18 17 16 15 14 13 12  
mean:[ADEBAYER][AGIC][ALSC][ANR][AHDR][ADPCC][ABLC][AF][AWB][AEC]

bit: 11-4 3-0  
mean:[sub modules][level]

[U] means unused now.

[level] : use 4 bits to define log levels.

each module log has following ascending levels:

0: error

1: warning

2: info

3: debug

4: verbose

5: low1

6-7: unused, now the same as debug

[sub modules] : use bits 4-11 to define the sub modules of each module, the specific meaning of each bit is decided by the module itself. These bits are designed to implement the sub module's log switch.

[modules] : AEC, AWB, AF ...

set debug level example:

eg. set module af log level to debug, and enable all sub modules of af:

Android:

setprop persist.vendor.rkisp.log 0x4ff4

Linux:

export persist\_camera\_engine\_log=0x4ff4

And if only want enable the sub module 1 log of af:

Android:

setprop persist.vendor.rkisp.log 0x4014

Linux:

export persist\_camera\_engine\_log=0x4014

如上说明，linux环境下通过设置环境变量persist\_camera\_engine\_log来控制各个模块的开关级别。

模块	Debug log	Verbose log
AE	export persist_camera_engine_log=0x1ff3	export persist_camera_engine_log=0x1ff3
AWB	export persist_camera_engine_log=0x2ff3	export persist_camera_engine_log=0x2ff3
AF	export persist_camera_engine_log=0x4ff3	export persist_camera_engine_log=0x4ff3
HDR	export persist_camera_engine_log=0x20ff3	export persist_camera_engine_log=0x20ff3
NR	export persist_camera_engine_log=0x40ff3	export persist_camera_engine_log=0x40ff3
Dehaze	export persist_camera_engine_log=0x2000ff3	export persist_camera_engine_log=0x2000ff3
Sharp	export persist_camera_engine_log=0x80000ff3	export persist_camera_engine_log=0x80000ff3
CAMHW	export persist_camera_engine_log=0x4000000ff3	export persist_camera_engine_log=0x4000000ff3

查看当前log级别可通过如下命令：

```
[root@RV1126_RV1109:/]# echo $persist_camera_engine_log
0x4014
```

## AE

### 配置指南

Log level	描述以及使能建议
Error	默认使能 严重错误
Warning	默认未使能 警告错误，算法内部可能针对该错误进行了异常处理，算法仍可继续运行
Info	默认未使能 <b>基本调试信息：</b> 逐帧输出信息：无 事件信息：算法中运行状态信息及帧率相关信息，一般仅在初始化、切换分辨率、修改配置参数时打印 <b>建议：</b> 用于获知算法运行状态及帧率值
Debug	默认未使能 <b>基本调试信息(相对Info等级增加)：</b> 逐帧输出信息：帧号、帧匹配曝光量、帧匹配AE统计信息(图像亮度)及曝光结果值 事件信息： <b>建议：</b> 通过连续的图像亮度、曝光量等信息来调试图像亮度闪烁等问题
Verbose	默认未使能 <b>基本调试信息(相对Debug等级增加)：</b> 逐帧输出信息：算法相关运算结果信息 事件信息： <b>建议：</b> 一般不开启，当debug等级无法判断问题时再开启

Sub modules bit	描述以及使能建议
bit[4]	与AE参数配置相关的log等级 <b>建议：</b> 读取IQ或API设置参数时，可根据该等级log查看配置参数是否正确。
bit[5]	AE信息处理模块日志开关，该模块根据AE统计计算环境以及图像相关指标。 <b>建议：</b> 读取和统计值相关的亮度信息
bit[6]	AE曝光量计算模块日志开关。 <b>建议：</b> 读取和曝光结果相关的信息
bit[7]	与Airis光圈算法相关的log等级， <b>建议：</b> 可变光圈相关问题。
bit[8:11]	无效

## log解读

由于篇幅限制，此处仅对debug等级（0x1ff3，对应AE所有子模块的debug级信息）的log进行内容解读。

- 线性模式AE LOG

```
rk_aiq_algo_ae_itf.cpp:262: Cur-Exp: FrmId=270,gain=0x36a,time=0x576,envChange=0,dcg=-1,pirs=0
rk_aiq_algo_ae_itf.cpp:266: Last-Res:FrmId=269,gain=0x356,time=0x576,pirs=0
rk_aiq_ae_algo.cpp:5861: ===== Linear-AE (enter)=====
rk_aiq_ae_algo.cpp:5881: >>> Framenum=270 Cur gain=6.826667,time=0.029987,pirisGain=0,RawMeanluma=29.564444,YuvMeanluma=34.875557,IsConverged=0
rk_aiq_ae_algo.cpp:2961: AecClimExecute: NewExposure(0.172051) SplitGain(5.735024) SplitIntegrationTime(0.030000) SplitPirisGain(0)
rk_aiq_ae_algo.cpp:5952: calc result:SetPoint=22.000000,gain=5.720671,time=0.029987,piris=0,reggain=845,regtime=1398
rk_aiq_ae_algo.cpp:6133: ===== (exit)=====
```

图3-1 线性模式AE LOG

如图3-1所示为线性模式的AE LOG示例。

Line1:

```
Cur-Exp: FrmId=270,gain=0x36a,time=0x576,envChange=0,dcg=-1,pirs=0
```

当前帧的曝光参数信息。

成员名称	描述
FrmId	当前帧的帧号
gain	当前帧对应的sensor曝光增益寄存器值
time	当前帧对应的sensor曝光时间寄存器值
envChange	当前帧是否发生环境亮度突变。0：环境亮度无突变；1：环境亮度突变
dchg	当前帧对应的dchg模式。-1：sensor不支持dchg模式 或 sensor内部进行dchg模式的切换；0：LCG模式；1：HCG模式
pirs	当前帧对应的p-iris光圈步进电机位置。若Airis功能关闭，该参数无效，无意义。

Line2:

```
Last-Res:FrmId=269,gain=0x356,time=0x576,pirs=0
```

上次运行AE设置的新曝光参数，部分参数与（1）中含义一致，此处不再赘述。通过比较Line1与Line2的曝光参数LOG信息，可知当前曝光是否与新曝光一致，即新曝光是否已经生效。

Line3:

```
===== Linear-AE
(enter)=====
```

进入AE控制算法模块，Linear-AE表示当前为线性曝光模式。

Line4:

```
Framenum=270
Cur
gain=6.826667,time=0.029987,pirisGain=0,RawMeanluma=29.564444,YuvMeanluma=3
4.875557,IsConverged=0
```

成员名称	描述
Framenum	当前帧的帧号
gain	当前帧对应的sensor曝光增益值
time	当前帧对应的sensor曝光时间值
pirisGain	当前帧对应的p-iris光圈等效增益值。若Airis功能关闭，该参数无效，无意义。
RawMeanluma	当前帧对应的debayer前raw图亮度，已扣除黑电平，并乘上awb gain。
YuvMeanluma	当前帧对应的gamma前RGB图亮度，用于判断debayer后的模块对亮度的影响。
IsConverged	当前帧曝光是否已经收敛。0：曝光未收敛；1：曝光已收敛

Line 5:

```
AecCImExecute: NewExposure(0.180993) SplitGain(6.033096)
SplitIntegrationTime(0.030000) SplitPirisGain(0)
```

成员名称	描述
NewExposure	AE控制算法得出的新曝光量值
SplitGain	新sensor曝光增益
SplitIntegrationTime	新sensor曝光时间
SplitPirisGain	新p-iris光圈等效增益值。若Airis功能关闭，该参数无效，无意义。

Line6:

```
calc
result:SetPoint=22.000000,gain=6.023529,time=0.029987,piris=0,reggain=854,regtime=
1398
```

最终设置的新曝光

成员名称	描述
SetPoint	目标亮度值
gain	最终的新曝光增益值
time	最终的新曝光时间值
piris	最终的新p-iris光圈等效增益值。若Airis功能关闭，该参数无效，无意义。
regain	最终的新曝光增益值对应的寄存器值
regtime	最终的新曝光时间值对应的寄存器值

综上，可得知当前帧的画面亮度RawMeanLuma，以及对应的目标亮度setpoint。通过比较画面亮度和目标亮度，计算新曝光。

- Hdr模式AE LOG:

```

rk_aiq_ae_algo_ae_itf.cpp:246: Cur-Exp: FrmId=22,S-gain=0x0,S-time=0x2b6,M-gain=0xb,M-time=0x1a5e,L-gain=0x0,L-time=0x0,envChange=1,dcg=-1--1--1,Piris=0
rk_aiq_ae_algo_ae_itf.cpp:254: Last-Res:FrmId=20,S-gain=0x5,S-time=0x8ca,M-gain=0x11,M-time=0x1a5e,L-gain=0x0,L-time=0x0

rk_aiq_ae_algo.cpp:5983: ===== HDR-AE (enter) =====
rk_aiq_ae_algo.cpp:6004: AecRun: SMeanLuma=9.342692, MMeanLuma=37.698597, LMeanLuma=0.000000, TmoMeanluma=37.571430, lsconverged=0, Longfrm=0
rk_aiq_ae_algo.cpp:6013: >>> Framerun=22 Cur_Piris=0, Sgain=1.000000, Stime=0.002570, mgain=1.462177, mtime=0.025000, lgain=1.000000, ltime=0.000000
rk_aiq_ae_algo.cpp:3308: S-HighLightLuma=197.250000, S-Target=100.000000, S-GlobalLuma=9.342692, S-Target=19.959433
rk_aiq_ae_algo.cpp:3733: L-LowLightLuma=29.626642, L-Target=48.572094, L-GlobalLuma=37.698597, L-Target=77.620155
rk_aiq_ae_algo.cpp:6110: calc result:piris=0,sgain=1.000000,stime=0.005081,mgain=1.862087,mtime=0.025000,lgain=0.000000,ltime=0.000000
rk_aiq_ae_algo.cpp:6133: ===== (exit) =====

```

图3-2 Hdr模式AE LOG

Line1:

Cur-Exp: FrmId=22,S-gain=0x0,S-time=0x2b6,M-gain=0xb,M-time=0x1a5e,L-gain=0x0,L-time=0x0,envChange=1,dcg=-1--1--1,Piris=0

当前帧的曝光参数信息。

成员名称	描述
FrmId	当前帧的帧号
S/M/L-gain	当前Hdr各帧对应的sensor曝光增益寄存器值。HDR 2帧模式时，S/M有效；HDR 3帧模式时，S/M/L皆有效。
S/M/L-time	当前Hdr各帧对应的sensor曝光时间寄存器值。HDR 2帧模式时，S/M有效；HDR 3帧模式时，S/M/L皆有效。
envChange	当前帧是否发生环境亮度突变。0：环境亮度无突变；1：环境亮度突变
dchg	当前帧对应的dchg模式，分别对应短中长3帧。Hdr 2帧模式时，前两个数值有效，分别代表短长帧的dchg模式；Hdr 3帧模式时，三个数值分别代表短中长的dchg模式。-1：sensor不支持dchg模式 或 sensor内部进行dchg模式的切换；0：LCG模式；1：HCG模式
pirs	当前帧对应的p-iris光圈步进电机位置。若Airis功能关闭，该参数无效，无意义。

Line2:

Last-Res:FrmId=20,S-gain=0x5,S-time=0x8ca,M-gain=0x11,M-time=0x1a5e,L-gain=0x0,L-time=0x0

上次运行AE设置的新曝光参数，部分参数与（1）中含义一致，此处不再赘述。通过比较Line1与Line2的曝光参数LOG信息，可知当前曝光是否与新曝光一致，即新曝光是否已经生效。

Line3:

```

=====
HDR-AE
(enter)=====

```

进入AE控制算法模块，HDR-AE表示当前为HDR曝光模式。

Line4:

AecRun: SMeanLuma=9.342692,  
MMeanLuma=37.698597,LMeanLuma=0.000000,TmoMeanluma=37.571430,lsconverge  
d=0,Longfrm=0

成员名称	描述
S/M/LMeanLuma	当前Hdr各帧的亮度均值。HDR 2帧模式时，S/M有效；HDR 3帧模式时，S/M/L皆有效。
TmoMeanluma	当前帧TMO模块输出的亮度均值。
Isconverged	当前Hdr各帧曝光量是否收敛。0：曝光未收敛；1：曝光已收敛。
Longfrm	当前帧的长帧模式状态。0：长帧模式关闭；1：长帧模式开启。

Line5:

```
Framenum=22 Cur Piris=0,
Sgain=1.000000,Stime=0.002570,mgain=1.462177,mtime=0.025000,lgain=1.000000,ltime=0.000000
```

成员名称	描述
Framenum	当前帧的帧号
s/m/lgain	当前帧对应的sensor曝光增益值。HDR 2帧模式时，s/m有效；HDR 3帧模式时，s/m/l皆有效。
s/m/ltime	当前帧对应的sensor曝光时间值。HDR 2帧模式时，s/m有效；HDR 3帧模式时，s/m/l皆有效。
piris	当前帧对应的p-iris光圈等效增益值。若Airis功能关闭，该参数无效，无意义。

Line6:

```
S-HighLightLuma=197.250000,S-Target=100.000000,S-GlobalLuma=9.342692,S-Target=19.959433
```

短帧控制算法LOG

成员名称	描述
S-HighLightLuma	当前短帧高亮区域亮度。
S-Target	当前短帧高亮区域目标亮度。
S-GlobalLuma	当前短帧全局区域平均亮度。
S-Target	当前短帧全局区域目标亮度。

Line7:

```
L-LowLightLuma=29.626642,L-Target=48.572094,L-GlobalLuma=37.698597,L-Target=77.620155
```

长帧控制算法LOG

成员名称	描述
L-LowLightLuma	当前长帧暗区亮度
L-Target	当前长帧暗区目标亮度。
L-GlobalLuma	当前长帧全局区域平均亮度。
L-Target	当前长帧全局区域目标亮度。

Line8:

```
calc
result:piris=0,sgain=1.000000,stime=0.005081,mgain=1.862087,mtime=0.025000,lgain
=0.000000,ltime=0.000000
```

AE控制算法输出的曝光结果

成员名称	描述
s/m/lgain	最终新sensor曝光增益值。HDR 2帧模式时，s/m有效；HDR 3帧模式时，s/m/l皆有效。
s/m/ltime	最终新sensor曝光时间值。HDR 2帧模式时，s/m有效；HDR 3帧模式时，s/m/l皆有效。
piris	最终新p-iris光圈等效增益值。若Airis功能关闭，该参数无效，无意义。

## AWB

### 配置指南

Log level	描述以及使能建议
Error	默认使能 严重错误
Warning	默认使能 警告错误， 算法内部可能针对该错误进行了异常处理.
Info	默认未使能 <b>基本调试信息:</b> 逐帧输出信息：wbgain结果 事件信息： 建议：
debug	默认未使能 <b>基本调试信息(相对Info等级增加):</b> 逐帧输出信息：白平衡收敛状态，环境亮度，白点数量，总的色温信息，概率大的前四个光源的概率等等 事件信息:awb的api调用后的相关属性信息打印 建议：当对输出log大小有限制时，可以采用该等级获取关键log信息用于debug
Verbose	默认未使能 <b>基本调试信息(相对Debug等级增加):</b> 逐帧输出信息：各个光源白点统计信息（白平衡增益，白点数量），及策略相关的中间量结果等等 事件信息： 建议：推荐使用该等级抓完整的log用于debug

Sub modules bit	描述以及使能建议
bit[4:11]	无效

## log解读

参考《Rockchip\_Color\_Optimization\_Guide\_ISP2x\_CN》

## AF

### 配置指南

Log level	描述以及使能建议
Error	默认使能 严重错误
Warning	默认未使能 警告错误，算法内部可能针对该错误进行了异常处理.
Info	默认未使能 <b>基本调试信息:</b> 逐帧输出信息：对焦搜索算法最终结果 事件信息： <b>建议：</b>
debug	默认未使能 <b>基本调试信息(相对Info等级增加):</b> 逐帧输出信息：FV统计值，马达移动位置信息 事件信息: 对焦触发、清晰位置搜索、变焦、跟焦、对焦 <b>建议：</b> 推荐使用该等级抓完整的log用于debug
Verbose	默认未使能 <b>基本调试信息(相对Debug等级增加):</b> 无 逐帧输出信息： 事件信息： <b>建议：</b> 无

Sub modules bit	描述以及使能建议
bit[4:11]	无效

**log解读**

**MERGE**

**配置指南**

Log level	描述以及使能建议
Info	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：暂未使用</p> <p>事件信息：暂未使用</p> <p><b>建议：</b>关闭</p>
debug	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：Merge调试信息</p> <p>事件信息：暂未使用</p> <p><b>建议：</b>当画面亮度不符合预期，或者TMO之后对比度不够时候开启</p>
Verbose	<p>默认未使能</p> <p><b>基本调试信息(相对Verbose等级增加):</b></p> <p>逐帧输出信息：Mrerge曝光相关信息</p> <p>事件信息：暂未使用</p> <p><b>建议：</b>当画面出现闪烁，且从AE log中确认是TMO在闪烁时开启，供RK人员debug 使用</p>

Sub modules bit	描述以及使能建议
bit[4:11]	无效

## log解读

当模式为线性时，merge日志等级如下：

```
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:0, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:1, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:2, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:3, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:4, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:5, It's in Linear Mode, Merge function bypass
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:256: AmergeProcess FrameID:6, It's in Linear Mode, Merge function bypass
```

当模式为HDR且基准帧为长帧时，merge日志等级为10000000ff3时

```
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:143: AmergeProcess:/#####Amerge Start#####
[AMERGE]:XCAM DEBUG rk_aiq_amege_algo.cpp:1274: AmergeByPassProcessing: FrameID:0 HDRFrameNum:2 LongFrmMode:0 MergeApiMode:0 EnvLv:0.000000 MoveCoef:0.000000 bypass:0
[AMERGE]:XCAM DEBUG rk_aiq_amege_algo.cpp:950: MergeDamping: Current BaseFrm:0 OECurve_smooth:80.000000 OECurve_offset:280.000000
[AMERGE]:XCAM DEBUG rk_aiq_amege_algo.cpp:953: MergeDamping: Current MDCurveMS_smooth:80.000000 MDCurveMS_offset:38.000000 MDCurveLM_smooth:80.000000 MDCurveLM_offset:38.000000
[AMERGE]:XCAM DEBUG rk_aiq_algo_amege_itf.cpp:253: AmergeProcess:/#####Amerge Over#####

```

名称	描述
FrameID	帧号
HDRFrameNum	HDR帧数
LongFrmMode	长帧模式开关，0：关闭，1：开启
MergeApiMode	API模式
EnvLv	环境亮度
MoveCoef	运动变量
bypass	当前帧bypass开关，0：关闭，1：开启
BaseFrm	基准帧模式，0：长帧模式，1：短帧模式
OECurve_smooth	当前帧过曝曲线的斜率，取值范围[0,200]，由json中参数逆归一化得到，系数200。
OECurve_offset	当前帧过曝曲线的偏移值，取值范围[108,280]。
MDCurveMS_smooth	当前帧中帧和短帧之间运动曲线斜率，取值范围为[0,200]，由json中参数逆归一化得到，系数200。
MDCurveMS_offset	当前帧中帧和短帧之间运动曲线偏移值，取值范围为[0.26,100]，由json中参数逆归一化得到，系数100。
MDCurveLM_smooth	当前帧长帧和中帧之间运动曲线斜率，取值范围为[0,200]，由json中参数逆归一化得到，系数200。
MDCurveLM_offset	当前帧长帧和中帧之间运动曲线偏移值，取值范围为[0.26,100]，由json中参数逆归一化得到，系数100。

当模式为HDR且基准帧为短帧时，merge日志等级为10000000ff3时

```
AmergeProcess:/#####
Amerge Start#####
AmergeByPassProcessing: FrameID:1 HDRFrameNum:2 LongFrmMode:0 MergeApiMode:0 EnvLv:0.000000 MoveCoef:0.000000 bypass:0
MergeDampingV12: Current BaseFrm:1 OECurve_smooth:0.400000 OECurve_offset:210.000000
MergeDampingV12: Current MDCurve_Coef:0.050000 MDCurve_ms_thd:0.000000 MDCurve_lm_thd:0.000000
AmergeProcess:/#####
Amerge Over#####

```

名称	描述
FrameID	帧号
HDRFrameNum	HDR帧数
LongFrmMode	长帧模式开关，0：关闭，1：开启
MergeApiMode	API模式
EnvLv	环境亮度
MoveCoef	运动变量
bypass	当前帧bypass开关，0：关闭，1：开启
BaseFrm	基准帧模式，0：长帧模式，1：短帧模式
OECurve_smooth	当前帧过曝曲线的斜率，取值范围[0,200]，由json中参数逆归一化得到，系数200。
OECurve_offset	当前帧过曝曲线的偏移值，取值范围[108,280]。
MDCurve_Coef	当前帧控制系数，取值范围[0,1]，默认值为0.05，精度0.0001。
MDCurve_ms_thd0	当前帧中短帧控制系数，取值范围[0,1]，默认值为0.0，精度0.1。
MDCurve_lm_thd0	当前帧长中帧控制系数，取值范围[0,1]，默认值为0.0，精度0.1

## DRC

### 配置指南

Log level	描述以及使能建议
Info	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：暂未使用 事件信息：暂未使用 <b>建议：</b>关闭</p>
debug	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：DRC调试信息 事件信息：暂未使用 <b>建议：</b>当画面亮度不符合预期，或者TMO之后对比度不够时候开启</p>
Verbose	<p>默认未使能</p> <p><b>基本调试信息(相对Verbose等级增加):</b></p> <p>逐帧输出信息：DRC曝光相关信息 事件信息：暂未使用 <b>建议：</b>当画面出现闪烁，且从AE log中确认是TMO在闪烁时开启，供RK人员debug 使用</p>

Sub modules bit	描述以及使能建议
bit[4:11]	无效

## log解读

DRC日志等级为20ff3时

```
processing:///////////////ADRC Start///////////////
AdrcBypassProcessing: FrameID:1 HDRFrameNum:2 LongFrmMode:0 DRCApiMode:0 EnvLv:0.000000 ISO:0.000000 bypass:0
AdrcTuningParaProcessing: Current DrcGain:1.000000 Alpha:0.200000 clip:16.000000 CompressMode:0
AdrcTuningParaProcessing: Current HiLight Strength:0.000000 gas_t:0.000000
AdrcTuningParaProcessing: Current LocalWeit:1.000000 LocalAutoEnable:1 LocalAutoWeit:0.000000 GlobalContrast:0.000000 LoLitContrast:0.000000 MotionStr:0.000000
AdrcDampingV12: Current damp DrcGain:1.000000 Alpha:0.200000 clip:16.000000 CompressMode:0
AdrcDampingV12: Current damp HiLight Strength:0.000000 gas_t:0.000000
AdrcDampingV12: current damp LocalWeit:1.000000 LocalAutoEnable:1 LocalAutoWeit:0.000000 GlobalContrast:0.000000 LoLitContrast:0.000000 MotionStr:0.000000
processing:///////////////ADRC Over///////////////
```

名称	描述
FrameID	帧号
HDRFrameNum	HDR帧数
LongFrmMode	长帧模式开关，0：关闭，1：开启
DRCApiMode	API模式
EnvLv	环境亮度
bypass	当前帧bypass开关，0：关闭，1：开启
Enable	DRC开关
DrcGain	当前帧DRC模块增益，取值范围[1,8]。
Alpha	当前帧DrcGain Alpha值，取值范围[0,1]。
Clip	当前帧DrcGain Clip值，取值范围[0,64]。
Strength	当前帧HiLight高光区域细节，取值范围[0,1]。
CompressMode	当前帧CompressSetting模式。
LocalWeit	当前帧Local权重，取值范围[0,1]，0：Global，1：全Local，默认值0。
LocalAutoEnable	当前帧自动LocalWeit开关，取值范围[0,1]，默认值为1，精度1。
LocalAutoWeit	当前帧自动LocalWeit值，取值范围[0,1]，默认值为0.4，精度0.01。
GlobalContrast	当前帧全局对比度，取值范围[0,1]，默认值为0，精度0.01。
LoLitContrast	当前帧低量区对比度，取值范围[0,1]，默认值为0，精度0.01。

## NR&Sharp

### 配置指南

Log level	描述以及使能建议
error	<p>默认使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：严重错误地方，可能会导致软件崩溃的打印 事件信息：暂未使用 <b>建议：</b>开启</p>
Info	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：所有函数调用信息，没有寄存器值打印。整体信息较多。 事件信息：暂未使用 <b>建议：</b>效果没有异常出错时候，不建议打开此项。一般建议使用io命令打印寄存器值。</p>
debug	<p>默认未使能</p> <p><b>基本调试信息(相对Info等级增加):</b></p> <p>逐帧输出信息：所有函数调用信息，包括寄存器值打印，寄存器值打印信息较多。 事件信息:暂未使用 <b>建议：</b>当效果出现严重错误时候，建议打开此项，提供给rk debug使用。</p>

Sub modules bit	描述以及使能建议
bit[4:11]	无效

## log解读

### Dhz&Ehz

#### 配置指南

Log level	描述以及使能建议
Info	<p>默认未使能</p> <p><b>基本调试信息:</b></p> <p>逐帧输出信息：暂未使用 事件信息：暂未使用 <b>建议：</b></p>
debug	<p>默认未使能</p> <p><b>基本调试信息(相对Info等级增加):</b></p> <p><b>逐帧输出信息：</b>Dhz&amp;Ehz调试信息 <b>事件信息:</b> <b>建议：</b>**</p>
Verbose	<p>默认未使能</p> <p><b>基本调试信息(相对Verbose等级增加):</b></p> <p>逐帧输出信息：暂未使用 事件信息：暂未使用 <b>建议：</b></p>

Sub modules bit	描述以及使能建议
bit[4:11]	无效

## Log解读

当Enhance功能开启时，Dehaze日志等级为2000ff3时

```
[ADEHAZE]:XCAM DEBUG rk_aiq_algo_adhaz_itf.cpp:145: ****Adehaze Start*****
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:2402: AdehazeByPassProcessing:FrameID:5 byPassProc:0 ISO:50.000000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:1423: AdehazeEnhanceApiBypassV30Process: Adehaze Api off!!!
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:223: EnableSettingV30: Dehaze module en:1 Dehaze en:0, Enhance en:1, Hist en:0
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:1022: GetEnhanceParamsV30 EnvLv:0.457436 enhance_value:1.300000 enhance_chroma:1.000000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:1024: GetEnhanceParamsV30 enhance_value_reg:0x533 enhance_chroma_reg:0x400
[ADEHAZE]:XCAM DEBUG rk_aiq_adhaz_itf.cpp:188: ****Adehaze over*****
```

名称	描述
FrameID	帧号
byPassProc	当前帧bypass开关，0：关闭，1：开启
ISO	ISO
Dehaze module en	Dehaze模块开关
Dehaze en	Dehaze功能开关
Enhance en	Enhance功能开关
Hist en	Hist功能开关
EnvLv	当前帧环境亮度
enhance_value	当前帧通用对比度力度，取值范围[0, 16]，推荐范围[1, 2]。
enhance_chroma	当前帧色度的增强调节参数，取值范围[0, 16]，推荐范围[1, 2]。
enhance_value_reg	当前帧通用对比度力度reg值。
enhance_chroma_reg	当前帧色度的增强调节参数reg值。

当Dehaze功能开启且cfg\_alpha=0时，Dehaze日志等级为2000ff3时

```
[ADEHAZE]:XCAM DEBUG rk_aiq_adhaz_itf.cpp:145: ****Adehaze Start*****
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:2402: AdehazeByPassProcessing:FrameID:3 byPassProc:0 ISO:50.000000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:1423: AdehazeEnhanceApiBypassV30Process: Adehaze Api off!!!
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:223: EnableSettingV30: Dehaze module en:1 Dehaze en:1, Enhance en:0, Hist en:0
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:960: GetDehazeParamsV30 cfg_alpha:0 EnvLv:0.433231 air_max:250.000000 air_min:200.000000 tmax_base:125.000000 wt_max:0.900000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:961: GetDehazeParamsV30 cfg_alpha_reg:0x0 air_max:0xfa air_min:0xc8 tmax_base:0x7d wt_max:0xe6
[ADEHAZE]:XCAM DEBUG rk_aiq_adhaz_itf.cpp:188: ****Adehaze over*****
```

名称	描述
FrameID	帧号
byPassProc	当前帧bypass开关， 0：关闭， 1：开启
ISO	ISO
Dehaze module en	Dehaze模块开关
Dehaze en	Dehaze功能开关
Enhance en	Enhance功能开关
Hist en	Hist功能开关
cfg_alpha	软件配置占比， 取值范围[0,1]， 默认值1， 精度0.01。
EnvLv	当前帧环境亮度
air_max	当前帧air自适应的最大值限制， 取值范围[230, 250]， 默认值250。
air_min	当前帧air自适应的最小值限制， 取值范围[230, 250]， 默认值200。
tmax_base	当前帧tmax自适应基础值， 默认125， 对应配置如下， 200(131), 210(125), 220(119), 230(114), 240(109), 250(105)， 推荐131-105
wt_max	当前帧wt自适应的最大值限制， 取值范围[0.75, 0.9]， 默认值0.9。

当Dehaze功能开启且cfg\_alpha=1时， Dehaze日志等级为2000ff3时

```
[ADEHAZE]:XCAM DEBUG rk_aiq_algo_adhaz_itf.cpp:145: ****Adehaze Start*****
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:2402: AdehazeByPassProcessing:FrameID:4 byPassProc:0 ISO:50.000000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:1423: AdehazeEnhanceApiBypassV30Process: Adehaze Api off!!!
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:223: EnableSettingV30: Dehaze module en:1 Dehaze en:1, Enhance en:0, Hist en:0
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:954: GetDehazeParamsV30 cfg_alpha:1 EnvLv:0.467077 cfg_air:210.000000 cfg_tmax:0.200000 cfg_wt:0.800000
[ADEHAZE]:XCAM DEBUG rk_aiq_adehaze_algo.cpp:955: GetDehazeParamsV30 cfg_alpha_reg:0x255 cfg_air:0xd2 cfg_tmax:0xcc cfg_wt:0xcc
[ADEHAZE]:XCAM DEBUG rk_aiq_algo_adhaz_itf.cpp:188: ****Adehaze over*****
```

名称	描述
FrameID	帧号
byPassProc	当前帧bypass开关，0：关闭，1：开启
ISO	ISO
Dehaze module en	Dehaze模块开关
Dehaze en	Dehaze功能开关
Enhance en	Enhance功能开关
Hist en	Hist功能开关
cfg_alpha	软件配置占比，取值范围[0,1]，默认值1，精度0.01。
EnvLv	当前帧环境亮度
cfg_air	当前帧软件配置air，大气光系数，取值范围[0, 255]，默认值210。
cfg_tmax	当前帧软件配置tmax，去雾的最大值，取值范围[0, 1]，默认值0.2。
cfg_wt	当前帧软件配置wt，图像去雾力度，取值范围[0, 1]，默认值0.8。

当Hist功能开启时，Dehaze日志等级为2000ff3时

```
[ADEHAZE]:XCAM DEBUG rk_aiq_algo_dehaz_itf.cpp:145: /*****Adehaze Start*****/
[ADEHAZE]:XCAM DEBUG rk_aiq_dehaze_algo.cpp:2402: AdehazeByPassProcessing:FrameID:3 byPassProc:0 ISO:50.000000
[ADEHAZE]:XCAM DEBUG rk_aiq_dehaze_algo.cpp:1423: AdehazeEnhanceApiByPassV30Process: Adehaze Api off!!!
[ADEHAZE]:XCAM DEBUG rk_aiq_dehaze_algo.cpp:223: EnableSettingV30: Dehaze module en:0, Enhance en:0, Hist en:1
[ADEHAZE]:XCAM DEBUG rk_aiq_dehaze_algo.cpp:1137: GetHistParamsV30 cfg_alpha:0.000000 EnvLv:0.452185 hist_para_en:1 hist_gratio:2.000000 hist_th_off:64.000000 hist_k:2.000000 hist_min:0.015000 hist_scale:0.090000 cfg_gratio:2.0000
[ADEHAZE]:XCAM DEBUG rk_aiq_dehaze_algo.cpp:1139: GetHistParamsV30 cfg_alpha_reg:0x0 hist_gratio_reg:0x10 hist_th_off_reg:0x40 hist_k_reg:0x8 hist_min_reg:0x3 hist_scale_reg:0x17 cfg_gratio_reg:0x200
[ADEHAZE]:XCAM DEBUG rk_aiq_algo_edhaz_itf.cpp:188: /*****Adehaze over*****/
```

名称	描述
FrameID	帧号
byPassProc	当前帧bypass开关，0：关闭，1：开启
ISO	ISO
Dehaze module en	Dehaze模块开关
Dehaze en	Dehaze功能开关
Enhance en	Enhance功能开关
Hist en	Hist功能开关
cfg_alpha	软件配置占比，取值范围[0,1]，默认值1，精度0.01。
EnvLv	当前帧环境亮度
hist_para_en	当前帧直方图拉伸控制开关。
hist_gratio	当前帧直方图拉伸倍数，直方图均衡控制系数，取值范围[0, 32]。
hist_th_off	当前帧直方图统计阈值，取值范围[0, 255]，默认值64。
hist_k	当前帧直方图自适应阈值放大倍数，取值范围[0, 7]，默认值2。
hist_min	当前帧直方图统计阈值的最小值，取值范围[0,2)，默认值0.016。
hist_scale	当前帧直方图均衡控制系数，取值范围[0, 32]。
cfg_gratio	当前帧软件配置直方图拉伸倍数，直方图均衡控制系数，取值范围[0, 32)。

## CamHW

### 配置指南

Log level	描述以及使能建议
Error	默认使能 严重错误
Warning	默认未使能 警告错误，算法内部可能针对该错误进行了异常处理.
Info	默认未使能 <b>基本调试信息:</b> 逐帧输出信息： 事件信息： <b>建议:</b>
Debug	默认未使能 <b>基本调试信息(相对Info等级增加):</b> 逐帧输出信息： 事件信息： <b>建议:</b>
Verbose	默认未使能 <b>基本调试信息(相对Debug等级增加):</b> 逐帧输出信息： 事件信息： <b>建议:</b> 存在单次大量信息输出，谨慎使用
Low1	默认未使能 <b>基本调试信息(相对Debug等级增加):</b> 逐帧输出信息： 事件信息：函数调用路径信息 <b>建议:</b>

Sub modules bit	描述以及使能建议
bit[4]	30 子模块log开关。 负责 HWI 流程控制部分。 <b>建议：</b> 有以下各子模块问题时建议一起使能。
bit[5]	Isp30Params 及 Isp30PolThread 子模块log开关。 负责 isp 参数、数据流、事件等处理部分。 <b>建议：</b> 疑似参数与视频帧未匹配、时序设置异常等问题 回读离线模式下，数据流断流等问题
bit[6]	SensorHw 子模块log开关。 负责 camera sensor 相关部分。 <b>建议：</b> 疑似CIS曝光设置等问题
bit[7]	FlashLight 子模块log开关。 负责补光灯控制部分。 <b>建议：</b> 补光灯控制问题。
bit[8]	LensHw 子模块log开关。 负责镜头马达控制部分。 <b>建议：</b> 马达控制等问题
bit[9]	30SpThread 子模块开关。 内部模块保留使用。
bit[10:11]	无效

## log解读

- SensorHw 子模块

```
[CAMHW]:XCAM DEBUG SensorHw.cpp:685: handle_sof: frameid=13, exp_list size=1, gain_list size=0
[CAMHW]:XCAM DEBUG SensorHw.cpp:726: handle_sof: working_mode=0,frameid=13, status: set_time=1, set_gain=0
[CAMHW]:XCAM DEBUG SensorHw.cpp:180: setLinearSensorExposure: frameid: 13, a-gain: 17, time: 2024, dcg: -1, snr: 0
```

上图log为线性模式时设置新曝光到sensor驱动流程，有新曝光时才会调用，具体含义如下表：

成员	描述
frameid	sof事件帧号
handle_sof	sof事件回调函数，每一帧回调一次，曝光设置在该回调中处理
exp_list size	曝光列表中还有多少个新曝光时间未设置到驱动
gain_list size	曝光列表中有多少个新曝光增益未设置到驱动
working_mode	当前工作模式，0为normal
set_time	该sof消息中是否有新曝光时间需要设置到驱动
set_gain	该sof消息中是否有新曝光增益需要设置到驱动
a-gain, time	新的曝光时间、增益寄存器值

## 如何采集Raw/YUV图像

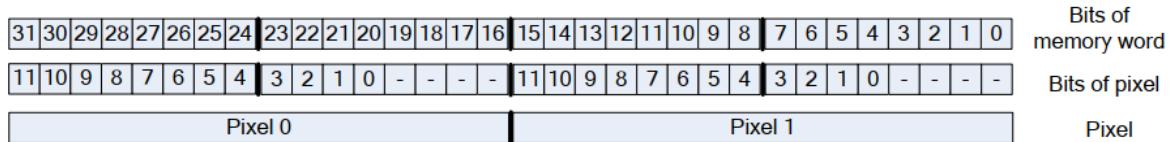
Raw数据特指CIS原始输出的数据，未经过任何图像处理。针对Bayer raw sensor，Raw数据即8/12/14 bit bayer rgb 数据。一般情况下，以下几种情况需要获取CIS Raw数据：

1. ISP处理后的YUV数据输出异常的情况下，希望动态截存此时ISP输入的Raw数据分析问题是否是CIS问题
2. 图像质量效果调试前，需要采集CIS Raw数据进行模组参数的标定

## Raw数据存储格式

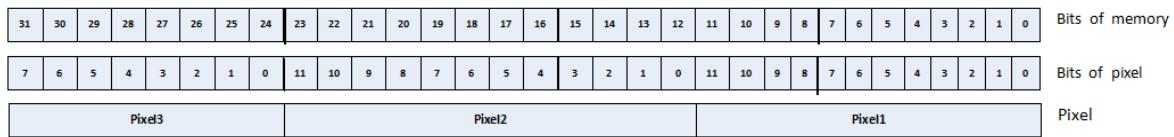
### 非紧凑型存储格式

对于raw12数据在内存中的存储排列方式，以4字节的内存片段为例，数据的存储方式如下所示：



### 紧凑型存储格式

对于raw12数据在内存中的存储排列方式，以4字节的内存片段为例，数据的存储方式如下所示：



## RK-Raw V1.0

项目	参数名称	数据类型定义	长度	描述
文件头	Identifier	unsigned short	2	固定 0x8080
	Header length	unsigned short	2	固定 128
	Frame index	unsigned int	4	帧索引号
	Width	unsigned short	2	图像宽度
	Height	unsigned short	2	图像高度
	Bit depth	unsigned char	1	图像位宽
	Bayer format	unsigned char	1	0: BGGR; 1: GBRG; 2: GRBG; 3: RGGB;
	Number of HDR Frame	unsigned char	1	帧类型： 1: 表示线性模式，短帧 2: 表示 2 帧 HDR，长帧+短帧 3: 表示 3 帧 HDR，长帧+中帧+短帧
	Current Frame type	unsigned char	1	当前帧类型： 1: 短帧 2: 中帧 3: 长帧
	Storage type	unsigned char	1	0: 紧凑型 1: 非紧凑型
RAW DATA	Line stride	unsigned short	2	单位为字节
	Effective line stride	unsigned short	2	单位为字节
	Reserved	unsigned char	107	保留段
RAW DATA	RAW DATA	RAW	W * H * BPP	RAW DATA

## RK-Raw V2.0

RK-Raw V2.0 格式由起始标识符、数据块以及结束标识符构成。

### 格式数据块定义

数据块有三部分组成：标识符ID，块长度，块数据。每个标识符ID都有唯一的固定值。

图表中的数据块并不都是必需的，可以选择其中的几个数据块组合使用。例如，一个RKRawV2文件可能只包含'Raw信息块'、'Raw数据块'和'帧统计信息块'，我们建议至少包含这三个数据块，因为这样的文件才有意义。

### Raw信息块定义

数据块定义: Raw信息 0xFF01				
项目	数据类型	长度(Byte)	默认值	说明
版本号	u16	2	0x0200	版本号: 0x0200=v2.0
Sensor 名	string	32	-	Sensor型号
场景/光源	string	32	-	采集场景/光源
帧号	u32	4	-	帧号
宽度	u16	2	-	图像宽度, 单位为像素
高度	u16	2	-	图像高度, 单位为像素
位宽	u8	1	-	图像位宽
Bayer格式	u8	1	-	0(BGGR);1(GBRG); 2(GRBG);3(RGGB);
HDR合成帧数	u8	1	-	1(线性模式);2(长+短帧);3(长+中+短帧);
存储格式	u8	1	-	0(紧凑);1(非紧凑);
行长	u16	2	-	单位为字节
有效行长	u16	2	-	单位为字节
大小端	u8	1	-	0(大端);1(小端);

### Raw数据块定义

该数据段中可以存放Raw图像数据，也可以存放指向Raw图像数据的buffer fd或虚拟地址。在使用相关API的时候需要传参指明该数据块中存储的类型，具体请参见[rk\\_aiq\\_rawbuf\\_type\\_t](#)。

### 帧统计信息块定义

## 寄存器格式块定义

### 寄存器块定义

### 平台信息块定义

## Raw/YUV数据采集方式

## 错误码

错误代码	描述
0	成功
-1	失败
-2	参数无效
-3	内存不足
-4	文件操作失败
-5	ANALYZER模块出错
-6	ISP模块出错
-7	sensor驱动出错
-8	线程操作出错
-9	IOCTL操作出错
-10	时序出错
-20	超时
-21	超出范围
-255	未知错误

## 缩略语

缩写	全称
CIS	Camera Image Sensor
RkAiq	Rockchip Automatical Image Quality
ISP	Image Signal Process
IQ Tunning	Image Quality Tunning