

Rockchip Development Guide ISP20

文件标识：RK-KF-GX-601

发布版本：V1.6.8

日期：2020-04-24

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 © 2020 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

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

本文旨在描述RkAiq (Rk Auto Image Quality) 模块的作用，整体工作流程，及相关的API接口。主要给

使用RkAiq模块进行ISP功能开发的工程师提供帮助。

产品版本

| 芯片名称 | 内核版本 |
|---------------|------------|
| RV1126/RV1109 | Linux 4.19 |

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 |
|--------|-----------|--------|-------|---------|
| RV1126 | Y | N | N | N |
| RV1109 | Y | N | N | N |

修订记录

| 版本号 | 作者 | 修改日期 | 修改说明 |
|--------|------------------|------------|---|
| V1.0.0 | 钟以崇 张云龙 徐鸿飞 | 2020-06-09 | 初始版本 |
| V1.0.1 | 张云龙 徐鸿飞 朱林靖 池晓芳 | 2020-08-04 | 增加统计信息章节 |
| V1.1.0 | 张云龙 | 2020-08-14 | 增加背光补偿、强光抑制等接口说明 |
| v1.2.0 | 胡克俊 | 2020-9-25 | 增加AF统计值说明 |
| v1.3.0 | 胡克俊 | 2020-10-14 | 增加AF模块API说明 |
| v1.4.0 | 张云龙 胡克俊 钟以崇 朱林靖 | 2020-10-20 | 1. 增加暗区提升、畸变校正API说明 2. 增加AF统计框图，用户AF算法开发说明了 3. 增加3a统计值获取的帧同步阻塞接口 4. AE模块更新api及模块参数、增加常见问题debug方法 |
| v1.5.0 | 朱林靖 张云龙 欧阳亚凤 李仁奎 | 2020-11-2 | 1. AE模块api增加使用demo、更新模块级api数据类型 2. 增加去马赛克接口说明 3. 增加NR/Sharp 的模块api及数据类型 4. 增加Gamma说明 |
| v1.6.0 | 李仁奎 | 2020-11-11 | 1.增加DPCC说明 |
| v1.6.1 | 胡克俊 | 2020-11-18 | 1.修改AF说明中的一些错误 |
| v1.6.2 | 朱林靖 | 2020-12-07 | 添加AE模块api新增参数说明 |
| v1.6.3 | 钟以崇 | 2020-12-12 | 添加 rk_aiq_uapi_sysctl_updatelq 说明 |
| v1.6.4 | 钟以崇 | 2020-01-12 | 添加离线帧功能说明 |
| v1.6.5 | 欧阳亚凤 | 2021-03-18 | 补充sharp api说明 |
| v1.6.6 | 张云龙 | 2021-03-20 | 补充defog模块级api说明 更新AIQ版本获取说明 |
| v1.6.7 | 张云龙 | 2021-04-06 | 更新抓raw和yuv图章节 |

目录

概述

功能描述

RkAiq架构

软件架构

软件流程

API说明

系统控制

功能概述

API参考

`rk_aiq_uapi_sysctl_init`

`rk_aiq_uapi_sysctl_deinit`

`rk_aiq_uapi_sysctl_prepare`

`rk_aiq_uapi_sysctl_start`

`rk_aiq_uapi_sysctl_stop`

`rk_aiq_uapi_sysctl_getStaticMetas`

`rk_aiq_uapi_sysctl_enumStaticMetas`

`rk_aiq_uapi_sysctl_setModuleCtl`

`rk_aiq_uapi_sysctl_getModuleCtl`

`rk_aiq_uapi_sysctl_regLib`

`rk_aiq_uapi_sysctl_unRegLib`

`rk_aiq_uapi_sysctl_enableAxlib`

`rk_aiq_uapi_sysctl_getAxlibStatus`

`rk_aiq_uapi_sysctl_getEnabledAxlibCtx`

`rk_aiq_uapi_sysctl_setCpsLtCfg`

`rk_aiq_uapi_sysctl_getCpsLtInfo`

`rk_aiq_uapi_sysctl_queryCpsLtCap`

`rk_aiq_uapi_sysctl_getBindedSnsEntNmByVd`

`rk_aiq_uapi_sysctl_updateEq`

`rk_aiq_uapi_sysctl_setCrop`

`rk_aiq_uapi_sysctl_getCrop`

数据类型

`rk_aiq_working_mode_t`

`rk_aiq_static_info_t`

`rk_aiq_sensor_info_t`

`rk_aiq_module_id_t`

`rk_aiq_cpsl_cfg_t`

`rk_aiq_cpsl_info_t`

`rk_aiq_cpsl_cap_t`

`rk_aiq_rect_t`

离线帧处理

概述

功能框图

功能描述

如何获取RK RAW V2.0格式文件

支持的RAW格式

API参考

`rk_aiq_uapi_sysctl_prepareRkRaw`

`rk_aiq_uapi_sysctl_enqueueRkRawBuf`

`rk_aiq_uapi_sysctl_enqueueRkRawFile`

`rk_aiq_uapi_sysctl_registRkRawCb`

数据结构

`rk_aiq_raw_prop_t`

`rk_aiq_rawbuf_type_t`

注意事项

参考示例

AE

概述

[重要概念](#)

[功能描述](#)

[功能级API参考](#)

[rk_aiq_uapi_setExpMode](#)
[rk_aiq_uapi_getExpMode](#)
[rk_aiq_uapi_setAeMode](#)
[rk_aiq_uapi_getAeMode](#)
[rk_aiq_uapi_setManualExp](#)
[rk_aiq_uapi_setExpGainRange](#)
[rk_aiq_uapi_getExpGainRange](#)
[rk_aiq_uapi_setExpTimeRange](#)
[rk_aiq_uapi_getExpTimeRange](#)
[rk_aiq_uapi_setBLCMode](#)
[rk_aiq_uapi_setBLCStrength](#)
[rk_aiq_uapi_setHLCMode](#)
[rk_aiq_uapi_setHLCStrength](#)
[rk_aiq_uapi_setDarkAreaBoostStrth](#)
[rk_aiq_uapi_getDarkAreaBoostStrth](#)
[rk_aiq_uapi_setAntiFlickerMode](#)
[rk_aiq_uapi_getAntiFlickerMode](#)
[rk_aiq_uapi_setExpPwrLineFreqMode](#)
[rk_aiq_uapi_getExpPwrLineFreqMode](#)

[功能级API数据类型](#)

[opMode_t](#)
[aeMode_t](#)
[paRange_t](#)
[aeMeasAreaType_t](#)
[expPwrLineFreq_t](#)
[antiFlickerMode_t](#)

[模块级API参考](#)

[rk_aiq_user_api_ae_setExpSwAttr](#)
[rk_aiq_user_api_ae_getExpSwAttr](#)
[rk_aiq_user_api_ae_setLinAeDayRouteAttr](#)
[rk_aiq_user_api_ae_getLinAeDayRouteAttr](#)
[rk_aiq_user_api_ae_setHdrAeDayRouteAttr](#)
[rk_aiq_user_api_ae_getHdrAeDayRouteAttr](#)
[rk_aiq_user_api_ae_setLinAeNightRouteAttr](#)
[rk_aiq_user_api_ae_getLinAeNightRouteAttr](#)
[rk_aiq_user_api_ae_setHdrAeNightRouteAttr](#)
[rk_aiq_user_api_ae_getHdrAeNightRouteAttr](#)
[rk_aiq_user_api_ae_queryExpResInfo](#)
[rk_aiq_user_api_ae_setLinExpAttr](#)
[rk_aiq_user_api_ae_getLinExpAttr](#)
[rk_aiq_user_api_ae_setHdrExpAttr](#)
[rk_aiq_user_api_ae_getHdrExpAttr](#)

[模块级API数据类型](#)

[Uapi_ExpSwAttr_t](#)
[Uapi_ExpSwAttr_Advanced_t](#)
[Uapi_IrisAttr_t](#)
 [CalibDb_Pliris_Attr_t](#)
 [CalibDb_DCIris_Attr_t](#)
[Uapi_AntiFlicker_t](#)
[Uapi_AeAttr_t](#)
 [CalibDb_AeSpeed_t](#)
 [CalibDb_AeRange_t](#)
 [CalibDb_LinAeRange_t](#)
 [CalibDb_HdrAeRange_t](#)
 [CalibDb_AeFrmRateAttr_t](#)

Uapi_MeAttr_t
 CalibDb_LinMeAttr_t
 CalibDb_HdrMeAttr_t
Uapi_ExpInitExp_t
 CalibDb_LinExpInitExp_t
 CalibDb_HdrExpInitExp_t
Uapi_LinAeRouteAttr_t
 AecExpSeparateName_t
Uapi_HdrAeRouteAttr_t
Uapi_LinExpAttr_t
 CalibDb_AecDynamicSetpoint_t
Uapi_HdrExpAttr_t
Uapi_ExpQueryInfo_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
 rk_aiq_uapi_setWBMode
 rk_aiq_uapi_getWBMode
 rk_aiq_uapi_lockAWB
 rk_aiq_uapi_unlockAWB
 rk_aiq_uapi_setMWBSceen
 rk_aiq_uapi_getMWBSceen
 rk_aiq_uapi_setMWBGain
 rk_aiq_uapi_getMWBGain
 rk_aiq_uapi_setMWBC
 rk_aiq_uapi_getMWBC

功能级API数据类型

 rk_aiq_wb_op_mode_t
 rk_aiq_wb_scene_t
 rk_aiq_wb_gain_t
 rk_aiq_wb_cct_t

模块级API参考

 rk_aiq_user_api_awb_SetAttrib
 rk_aiq_user_api_awb_GetAttrib
 rk_aiq_user_api_awb_GetCCT
 rk_aiq_user_api_awb_QueryWBInfo

模块级API数据类型

 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_wb_awb_attrib_t
 rk_aiq_wb_attrib_t
 rk_aiq_wb_querry_info_t

AF

概述
重要概念
功能描述
开发用户AF算法
功能级API参考
 rk_aiq_uapi_setFocusMeasCfg

[rk_aiq_uapi_setFocusMode](#)
[rk_aiq_uapi_getFocusMode](#)
[rk_aiq_uapi_setFixedModeCode](#)
[rk_aiq_uapi_getFixedModeCode](#)
[rk_aiq_uapi_setFocusWin](#)
[rk_aiq_uapi_getFocusWin](#)
[rk_aiq_uapi_lockFocus](#)
[rk_aiq_uapi_unlockFocus](#)
[rk_aiq_uapi_oneshotFocus](#)
[rk_aiq_uapi_manualTrigerFocus](#)
[rk_aiq_uapi_trackingFocus](#)
[rk_aiq_uapi_setVcmCfg](#)
[rk_aiq_uapi_getVcmCfg](#)
[rk_aiq_uapi_setOpZoomPosition](#)
[rk_aiq_uapi_getOpZoomPosition](#)

功能级API数据类型

[rk_aiq_af_algo_meas_t](#)
[opMode_t](#)
[rk_aiq_lens_vcmcfg](#)

模块级API参考

[rk_aiq_user_api_af_SetAttrib](#)
[rk_aiq_user_api_af_GetAttrib](#)

模块级API数据类型

[RKAIQ_AF_MODE](#)
[rk_aiq_af_attrib_t](#)

其它说明

[VCM驱动验证](#)
[Normal模式提高对焦速度](#)

IMGPROC

概述

FEC

[功能描述](#)

[重要概念](#)

功能级API参考

[rk_aiq_uapi_setFecEn](#)
[rk_aiq_uapi_setFecCorrectDirection](#)
[rk_aiq_uapi_setFecBypass](#)
[rk_aiq_uapi_setFecCorrectLevel](#)

模块级API参考

[rk_aiq_user_api_afec_SetAttrib](#)
[rk_aiq_user_api_afec_GetAttrib](#)

模块级API数据类型

[fec_correct_direction_t](#)
[rk_aiq_fec_attrib_t](#)

性能优化

LDCH

[功能描述](#)

功能级API参考

[rk_aiq_uapi_setLdchEn](#)
[rk_aiq_uapi_setLdchCorrectLevel](#)

模块级API参考

[rk_aiq_user_api_aldch_SetAttrib](#)
[rk_aiq_user_api_aldch_GetAttrib](#)

模块级API数据类型

[rk_aiq_ldch_attrib_t](#)

HDR

[功能描述](#)

[重要概念](#)

功能级API参考

rk_aiq_uapi_setHDRMode
rk_aiq_uapi_getHDRMode
rk_aiq_uapi_setMHDRStrth
rk_aiq_uapi_getMHDRStrth

功能级API数据类型

hdr_OpMode_t
FastMode_t
DarkArea_t

模块级API参考

rk_aiq_uapi_ahdr_SetAttrib
rk_aiq_uapi_ahdr_GetAttrib

模块级API数据类型

hdr_OpMode_t
mgeCtrlData_t
amgeAttr_t
tmoCtrlData_t
AGlobalTmoData_t
atmoAttr_t
ahdrAttr_t
mmgeAttr_t
mtmoAttr_t
mhdrAttr_t
FastMode_t
DarkArea_t
CurrCtlData_t
CurrRegData_t
CalibDb_HdrMerge_t
TMO_en_t
GlobalLuma_t
DetailsHighLight_t
DetailsLowLight_t
LocalTMO_t
'GlobaTMO_t
CalibDb_HdrTmo_t
CalibDb_Ahdr_Para_t
hdrAttr_t

Noise Removal

功能描述

功能级API参考

rk_aiq_uapi_setNRMode
rk_aiq_uapi_getNRMode
rk_aiq_uapi_setANRStrth
rk_aiq_uapi_getANRStrth
rk_aiq_uapi_setMSpaNRStrth
rk_aiq_uapi_getMSpaNRStrth
rk_aiq_uapi_setMTNRStrth
rk_aiq_uapi_getMTNRStrth

模块级API参考

rk_aiq_user_api_anr_SetAttrib
rk_aiq_user_api_anr_GetAttrib
rk_aiq_user_api_anr_SetIQPara
rk_aiq_user_api_anr_GetIQPara

模块级API数据类型

rk_aiq_nr_attrib_t
ANROPMode_t
ANR_Auto_Attr_t
ANR_Manual_Attr_t

`rk_aiq_nr_IQPara_t`
`rk_aiq_nr_module_t`
`CalibDb_BayerNr_t`
`CalibDb_BayerNr_ModeCell_t`
`CalibDb_BayerNR_Params_t`
`CalibDb_MFNR_t`
`CalibDb_MFNR_ModeCell_t`
`CalibDb_MFNR_Dynamic_t`
`CalibDb_MFNR_Setting_t`
`CalibDb_MFNR_ISO_s`
`CalibDb_UVNR_t`
`CalibDb_UVNR_ModeCell_t`
`CalibDb_UVNR_Params_t`
`CalibDb_YNR_t`
`CalibDb_YNR_ModeCell_t`
`CalibDb_YNR_Setting_t`
`CalibDb_YNR_ISO_t`

Defog

[功能描述](#)

[功能级API参考](#)

`rk_aiq_uapi_setDhzMode`
`rk_aiq_uapi_getDhzMode`
`rk_aiq_uapi_setMDhzStrth`
`rk_aiq_uapi_getMDhzStrth`
`rk_aiq_uapi_enableDhz`
`rk_aiq_uapi_disableDhz`

[模块级API参考](#)

`rk_aiq_user_api_adehaze_setSwAttrib`
`rk_aiq_user_api_adehaze_getSwAttrib`

ACM

[功能描述](#)

[API参考](#)

`rk_aiq_uapi_setBrightness`
`rk_aiq_uapi_getBrightness`
`rk_aiq_uapi_setContrast`
`rk_aiq_uapi_getContrast`
`rk_aiq_uapi_setSaturation`
`rk_aiq_uapi_getSaturation`
`rk_aiq_uapi_setHue`
`rk_aiq_uapi_getHue`

Sharpen

[功能描述](#)

[功能级API参考](#)

`rk_aiq_uapi_setSharpness`
`rk_aiq_uapi_getSharpness`

[模块级API参考](#)

`rk_aiq_user_api_asharp_SetAttrib`
`rk_aiq_user_api_asharp_GetAttrib`
`rk_aiq_user_api_asharp_SetIQPara`
`rk_aiq_user_api_asharp_GetIQPara`

[模块级API数据类型](#)

`rk_aiq_sharp_attrib_t`
`AsharpOPMode_t`
`Asharp_Auto_Attr_t`
`Asharp_Manual_Attr_t`
`rk_aiq_sharp_IQpara_t`
`rk_aiq_sharp_module_t`
`CalibDb_Sharp_t`

CalibDb_Sharp_ModeCell_t
CalibDb_Sharp_Setting_t
CalibDb_Sharp_ISO_s
CalibDb_EdgeFilter_t
CalibDb_EdgeFilter_ModeCell_t
CalibDb_EdgeFilter_Setting_t
CalibDb_EdgeFilter_ISO_s

Gamma

功能描述

功能级API参考

rk_aiq_uapi_setGammaCoef

功能级API数据类型

rk_aiq_gamma_op_mode_t
rk_gamma_curve_type_t
rk_gamma_curve_usr_define1_para_t
rk_gamma_curve_usr_define2_para_t
Agamma_api_manual_t
CalibDb_Gamma_t
rk_aiq_gamma_attr_t

模块级API参考

rk_aiq_user_api_agamma_SetAttrib
rk_aiq_user_api_agamma_GetAttrib

DPCC

功能描述

模块级API参考

rk_aiq_user_api_adpcc_SetAttrib
rk_aiq_user_api_adpcc_GetAttrib

模块级API数据类型

AdpccOPMode_t
Adpcc_basic_params_select_t
Adpcc_basic_params_t
Adpcc_bpt_params_t
dpcc_pdaf_point_t
Adpcc_pdaf_params_t
CalibDb_Dpcc_Fast_Mode_t
CalibDb_Dpcc_Sensor_t
Adpcc_bpt_params_select_t
Adpcc_pdaf_params_select_t
Adpcc_Auto_Attr_t
Adpcc_fast_mode_attr_t
Adpcc_sensor_dpcc_attr_t
Adpcc_Manual_Attr_t
CalibDb_Dpcc_Pdaf_t
CalibDb_Dpcc_set_RK_t
CalibDb_Dpcc_set_LC_t
CalibDb_Dpcc_set_PG_t
CalibDb_Dpcc_set_RND_t
CalibDb_Dpcc_set_RG_t
CalibDb_Dpcc_set_RO_t
CalibDb_Dpcc_set_t
CalibDb_Dpcc_Expert_Mode_t
CalibDb_Dpcc_t
rk_aiq_dpcc_attrib_t

ASD

模块级API参考

rk_aiq_user_api_asd_GetAttrib

数据类型

asd_attrib_t

[Demosaic](#)

[功能描述](#)

[模块级API参考](#)

[rk_aiq_user_api_adebayer_SetAttrib](#)

[rk_aiq_user_api_adebayer_GetAttrib](#)

[数据类型](#)

[adebayer_attrib_t](#)

[其他](#)

[API参考](#)

[rk_aiq_uapi_setGrayMode](#)

[rk_aiq_uapi_getGrayMode](#)

[rk_aiq_uapi_setFrameRate](#)

[rk_aiq_uapi_getFrameRate](#)

[rk_aiq_uapi_setMirroFlip](#)

[rk_aiq_uapi_getMirroFlip](#)

[数据类型](#)

[rk_aiq_gray_mode_t](#)

[统计信息](#)

[概述](#)

[功能描述](#)

[AE统计信息](#)

[基于raw图的AE统计](#)

[基于RGB图的AE统计](#)

[AWB统计信息](#)

[AF统计信息](#)

[API参考](#)

[rk_aiq_uapi_sysctl_get3AStats](#)

[rk_aiq_uapi_sysctl_get3AStatsBlk](#)

[rk_aiq_uapi_sysctl_release3AStatsRef](#)

[数据类型](#)

[rk_aiq_isp_stats_t](#)

[RKAiqAecStats_t](#)

[RKAiqAecExpInfo_t](#)

[RkAiqExpParamComb_t](#)

[RkAiqAecHwStatsRes_t](#)

[Aec_Stat_Res_t](#)

[rawaebig_stat](#)

[rawaelite_stat](#)

[rawhist_stat](#)

[yuvae_stat](#)

[sihist_stat](#)

[rk_aiq_awb_stat_res_v200_t](#)

[rk_aiq_awb_stat_wp_res_light_v200_t](#)

[rk_aiq_awb_stat_wp_res_v200_t](#)

[rk_aiq_awb_stat_blk_res_v200_t](#)

[rk_aiq_af_algo_stat_t](#)

[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](#)

[Log开关](#)

[Log解读](#)

[AE](#)

[AF](#)

[AWB](#)

[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
AIQ动态截存方式
v4l2-ctl直接采集方式
RK IQ Tool(Tuner)抓图工具
使用场景介绍
错误码
缩略语

概述

ISP20 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、3DLUT、去噪（包括RAW域去噪，多帧降噪，颜色去噪等）、锐化等。

ISP20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

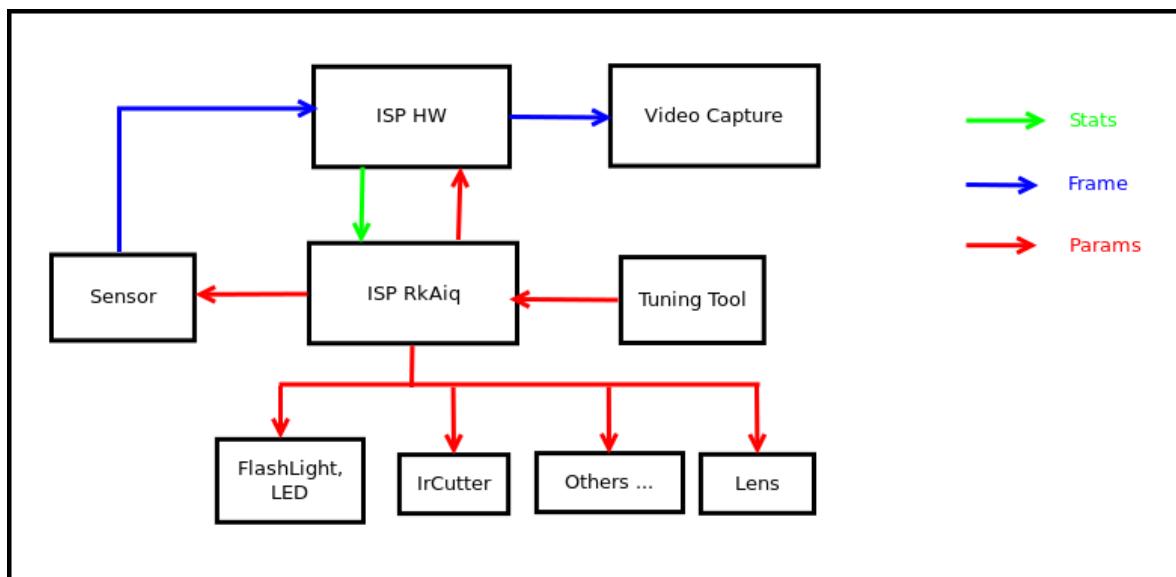


图1-1 ISP20 系统框图

ISP20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。Tuning 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

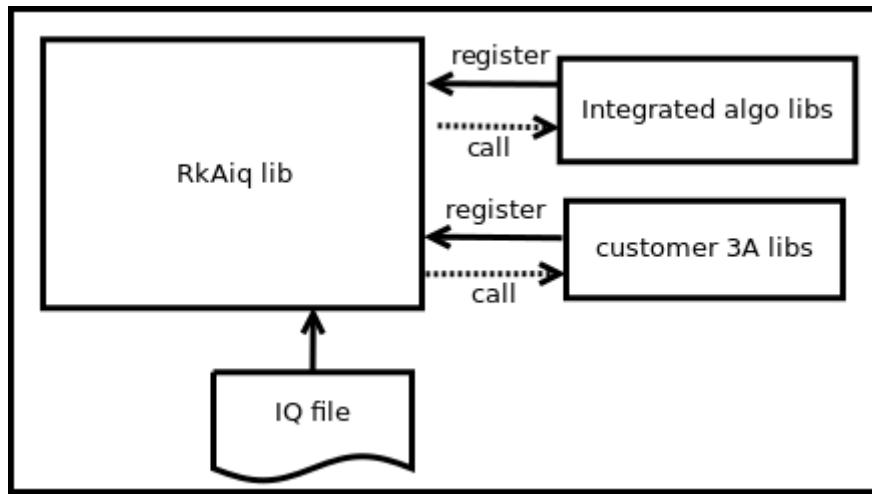


图1-2 RkAiq总体架构图

ISP20 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

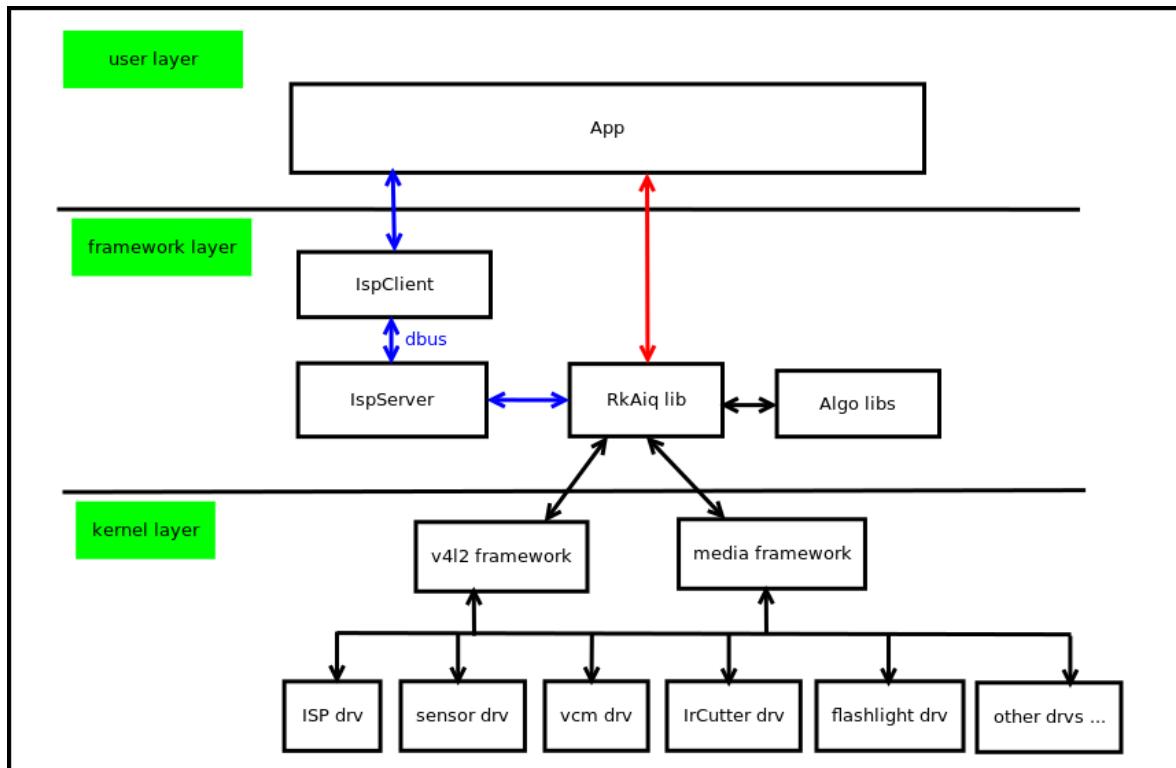


图1-3 软件架构框图

ISP20 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。

软件流程

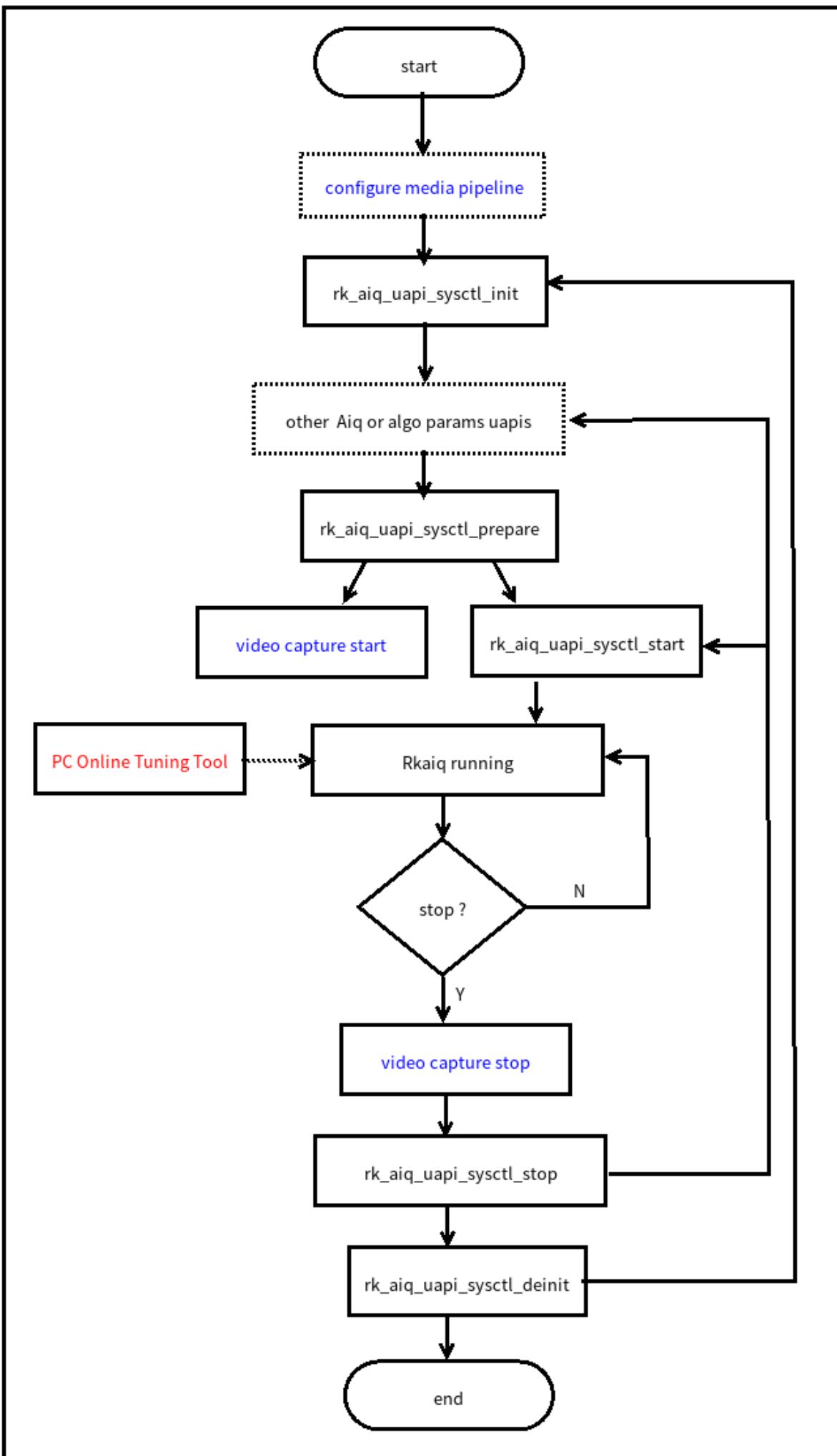


图1-4 流程图

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ISP20 pipeline，如sensor输出分辨率等等，驱动已有默认配置。
- rk_aiq_uapi_sysctl_init。初始化RkAiq，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_uapi_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_uapi_sysctl_prepare后调用。
- rk_aiq_uapi_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 Tuning Tool。PC端可通过Tuning 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_uapi_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_uapi_sysctl_deinit。反初始化RkAiq。

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。其中功能级别API是基于模块级别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。模块级别API提供对该模块的详细参数设置以及查询，未对功能进行API区分。

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。

API参考

`rk_aiq_uapi_sysctl_init`

【描述】

初始化AIQ上下文。

【语法】

```
rk_aiq_sys_ctx_t*  
rk_aiq_uapi_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_api_sysctl.h
- 库文件: librkaiq.so

【注意】

- 应先于其他函数调用。

rk_aiq_uapi_sysctl_deinit

【描述】

反初始化AIQ上下文环境。

【语法】

```
void
rk_aiq_uapi_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_api_sysctl.h
- 库文件: librkaiq.so

【注意】

- 不应在AIQ处于start状态调用。

rk_aiq_uapi_sysctl_prepare

【描述】

准备AIQ运行环境。

【语法】

```
XCamReturn  
rk_aiq_uapi_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，仅用于校验 | 输入 |
| height | sensor输出的分辨率高度，仅用于校验 | 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_api_sysctl.h
- 库文件: librkaiq.so

【注意】

- 应在rk_aiq_uapi_sysctl_start函数之前调用。
- 如果需要在rk_aiq_uapi_sysctl_start之后调用本函数，那么先调用rk_aiq_uapi_sysctl_stop函数，再调用rk_aiq_uapi_sysctl_prepare重新准备运行环境。

rk_aiq_uapi_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_uapi_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_api_sysctl.h
- 库文件: librkaiq.so

【注意】

- 应在rk_aiq_uapi_sysctl_prepare函数之后调用。

rk_aiq_uapi_sysctl_stop

【描述】

停止AIQ控制系统。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_stop(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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_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_uapi_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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_enumStaticMetas

【描述】

枚举AIQ获取到的静态信息。

【语法】

```
XCamReturn  
rk_aiq_uapi_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_api_sysctl.h

- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_setModuleCtl

【描述】

AIQ模块开关设置。

【语法】

```
XCamReturn
rk_aiq_uapi_sysctl_setModuleCtl(const rk_aiq_sys_ctx_t* ctx, rk_aiq_module_id_t
mId, bool mod_en);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|-------------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| mld | 模块ID | 输入 |
| mod_en | true为开启, false为关闭 | 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_getModuleCtl

【描述】

AIQ模块状态查询。

【语法】

```
XCamReturn
rk_aiq_uapi_sysctl_getModuleCtl(const rk_aiq_sys_ctx_t* ctx, rk_aiq_module_id_t
mId, bool *mod_en);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| mld | 模块ID | 输入 |
| mod_en | 当前状态 | 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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_regLib

【描述】

注册自定义算法库。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_regLib(const rk_aiq_sys_ctx_t* ctx,  
                           RkAiqAlgoDesComm* algo_lib_des);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------------|-------------------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| algo_lib_des | 算法描述结构体，字段id为AIQ生成的标识ID | 输入&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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_unRegLib

【描述】

注销自定义算法库。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_unRegLib(const rk_aiq_sys_ctx_t* ctx,  
                           const int algo_type,  
                           const int lib_id);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| algo_type | 要操作的算法模块类型 | 输入 |
| lib_id | 算法库标识ID | 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_enableAxlib

【描述】

设置自定义算法库运行状态。

【语法】

```
XCamReturn
rk_aiq_uapi_sysctl_enableAxlib(const rk_aiq_sys_ctx_t* ctx,
                                  const int algo_type,
                                  const int lib_id,
                                  bool enable);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| algo_type | 要操作的算法模块类型 | 输入 |
| lib_id | 算法库标识ID | 输入 |
| enable | 状态设置 | 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

【注意】

- 如果lib_id等同于当前运行的算法库，本函数可以在除未初始化外的任何状态下调用。
- 其他情况，仅在prepared状态下调用，并且algo_type所标识的算法库将被lib_id标识的新算法库替代。

rk_aiq_uapi_sysctl_getAxlibStatus

【描述】

获取算法库状态。

【语法】

```
bool rk_aiq_uapi_sysctl_getAxlibstatus(const rk_aiq_sys_ctx_t* ctx,  
                                         const int algo_type,  
                                         const int lib_id);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| algo_type | 要操作的算法模块类型 | 输入 |
| lib_id | 算法库标识ID | 输入 |

【返回值】

| 返回值 | 描述 |
|-------|------|
| false | 关闭状态 |
| true | 使能状态 |

【需求】

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

rk_aiq_uapi_sysctl_getEnabledAxlibCtx

【描述】

获取使能算法库的上下文结构体。

【语法】

```
const RkAiqAlgoContext* rk_aiq_uapi_sysctl_getEnabledAxlibCtx(const rk_aiq_sys_ctx_t* ctx, const int algo_type);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| algo_type | 要操作的算法模块类型 | 输入 |

【返回值】

| 返回值 | 描述 |
|-------|------|
| NULL | 获取失败 |
| 非NULL | 获取成功 |

【需求】

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

【注意】

- 返回的算法上下文结构体将被内部私有函数使用。对于用户自定义的算法库，该函数应在 rk_aiq_uapi_sysctl_enableAxlib 之后调用，否则将返回NULL。

rk_aiq_uapi_sysctl_setCpsLtCfg

【描述】

设置补光灯控制信息。

【语法】

```
XCamReturn
rk_aiq_uapi_sysctl_setCpsLtCfg(const rk_aiq_sys_ctx_t* ctx,
                                  rk_aiq_cpsl_cfg_t* cfg);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| cfg | 补光灯配置结构体指针 | 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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_getCpsLtInfo

【描述】

获取补光灯控制信息。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_getCpsLtInfo(const rk_aiq_sys_ctx_t* ctx,  
rk_aiq_cpsl_info_t* info);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|------|------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| info | 补光灯配置结构体指针 | 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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_queryCpsLtCap

【描述】

查询补光灯的支持能力。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_queryCpsLtCap(const rk_aiq_sys_ctx_t* ctx,  
rk_aiq_cpsl_cap_t* cap);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|------|----------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| cap | 补光灯支持能力查询结构体指针 | 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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_getBindedSnsEntNmByVd

【描述】

查询video结点所对应的sensor entity name。

【语法】

```
const char* rk_aiq_uapi_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ISPP scale结点路径。

【需求】

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

rk_aiq_uapi_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。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_setCrop

【描述】

设置crop参数。

【语法】

```
XCamReturn
rk_aiq_uapi_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 | 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】

- 最小crop分辨率为64x64。
- 要设置的resolution须存在于IQ XML效果文件中。
- 须在rk_aiq_uapi_sysctl_prepare之前调用。
- rect.width须为4的整数倍，对于不同的数据格式，对水平偏移量rect.left有不同的要求：
raw8和yuv422格式，rect.left须为8的整数倍。
raw10和raw12格式，rect.left须为4的整数倍。
rgb888格式，rect.left须为24的整数倍。

rk_aiq_uapi_sysctl_getCrop

【描述】

获取crop参数。

【语法】

```
XCamReturn
rk_aiq_uapi_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 |

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_module_id_t

【说明】

AIQ 模块ID

【定义】

```

typedef enum {
    RK_MODULE_INVAL = 0,
    RK_MODULE_DPCC,
    RK_MODULE_BLS,
    RK_MODULE_LSC,
    RK_MODULE_AWB_GAIN,
    RK_MODULE_CTK,
    RK_MODULE_GOC,
    RK_MODULE_SHARP,
    RK_MODULE_AE,
    RK_MODULE_AWB,
    RK_MODULE_NR,
    RK_MODULE_GIC,
    RK_MODULE_3DLUT,
    RK_MODULE_LDCH,
    RK_MODULE_TNR,
    RK_MODULE_FEC,
    RK_MODULE_MAX
} rk_aiq_module_id_t;

```

【成员】

| 成员名称 | 描述 |
|--------------------|---------|
| RK_MODULE_DPCC | 坏点检测与纠正 |
| RK_MODULE_BLS | 黑电平 |
| RK_MODULE_LSC | 镜头阴影校正 |
| RK_MODULE_AWB_GAIN | 白平衡增益 |
| RK_MODULE_CTK | 颜色校正 |
| RK_MODULE_GOC | 伽玛 |
| RK_MODULE_SHARP | 锐化 |
| RK_MODULE_AE | 曝光 |
| RK_MODULE_AWB | 白平衡 |
| RK_MODULE_NR | 去噪 |
| RK_MODULE_GIC | 绿平衡 |
| RK_MODULE_3DLUT | 3DLUT |
| RK_MODULE_LDCH | LDCH |
| RK_MODULE_TNR | 3D去噪 |
| RK_MODULE_FEC | 鱼眼校正 |

rk_aiq_cpsl_cfg_t

【说明】

补光灯设置信息结构体

【定义】

```
typedef struct rk_aiq_cpsl_cfg_s {
    RKAiqOPMode_t mode;
    rk_aiq_cpsls_t light_src;
    bool gray_on; /*!< force to gray if light on */
    union {
        struct {
            float sensitivity; /*!< Range [0-100] */
            uint32_t sw_interval; /*!< switch interval time, unit seconds */
        } a; /*!< auto mode */
        struct {
            uint8_t on; /*!< disable 0, enable 1 */
            float strength_led; /*!< Range [0-100] */
            float strength_ir; /*!< Range [0-100] */
        } m; /*!< manual mode */
    } u;
} rk_aiq_cpsl_cfg_t;
```

【成员】

| 成员名称 | 描述 |
|--------------|-------------------------|
| mode | 工作模式 |
| lght_src | 光源类型 |
| gray_on | 切换为夜晚模式后是否将画面切为黑白 |
| sensitivity | 自动模式下的切换灵敏度, 范围[0,100] |
| sw_interval | 自动模式下的切换间隔, 单位秒 |
| on | 手动模式下是否切换为夜晚模式 |
| strength_led | 手动模式下的LED灯强度, 范围[0,100] |
| strength_ir | 手动模式下的红外灯强度, 范围[0,100] |

rk_aiq_cpsl_info_t

【说明】

补光灯查询信息结构体

【定义】

```
typedef struct rk_aiq_cpsl_info_s {
    int32_t mode;
    uint8_t on;
    bool gray;
    float strength_led;
    float strength_ir;
    float sensitivity;
    uint32_t sw_interval;
    int32_t lght_src;
} rk_aiq_cpsl_info_t;
```

【成员】

| 成员名称 | 描述 |
|--------------|-------------------------|
| mode | 工作模式 |
| lght_src | 光源类型 |
| gray | 切换为夜晚模式后是否将画面切为黑白 |
| sensitivity | 自动模式下的切换灵敏度, 范围[0,100] |
| sw_interval | 自动模式下的切换间隔, 单位秒 |
| on | 手动模式下是否切换为夜晚模式 |
| strength_led | 手动模式下的LED灯强度, 范围[0,100] |
| strength_ir | 手动模式下的红外灯强度, 范围[0,100] |

rk_aiq_cpsl_cap_t

【说明】

补光灯支持能力结构体

【定义】

```
typedef struct rk_aiq_cpsl_cap_s {
    int32_t supported_modes[RK_AIQ_OP_MODE_MAX];
    uint8_t modes_num;
    int32_t supported_lght_src[RK_AIQ_CPSLS_MAX];
    uint8_t lght_src_num;
    rk_aiq_range_t strength_led;
    rk_aiq_range_t sensitivity;
    rk_aiq_range_t strength_ir;
} rk_aiq_cpsl_cap_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------|-------------------|
| supported_modes | 支持的工作模式 |
| modes_num | 支持的模式个数 |
| gray | 切换为夜晚模式后是否将画面切为黑白 |
| supported_lght_src | 支持的光源 |
| lght_src_num | 支持的光源个数 |
| strength_led | LED的强度范围 |
| sensitivity | 灵敏度范围 |
| strength_ir | 红外灯的强度范围 |

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】

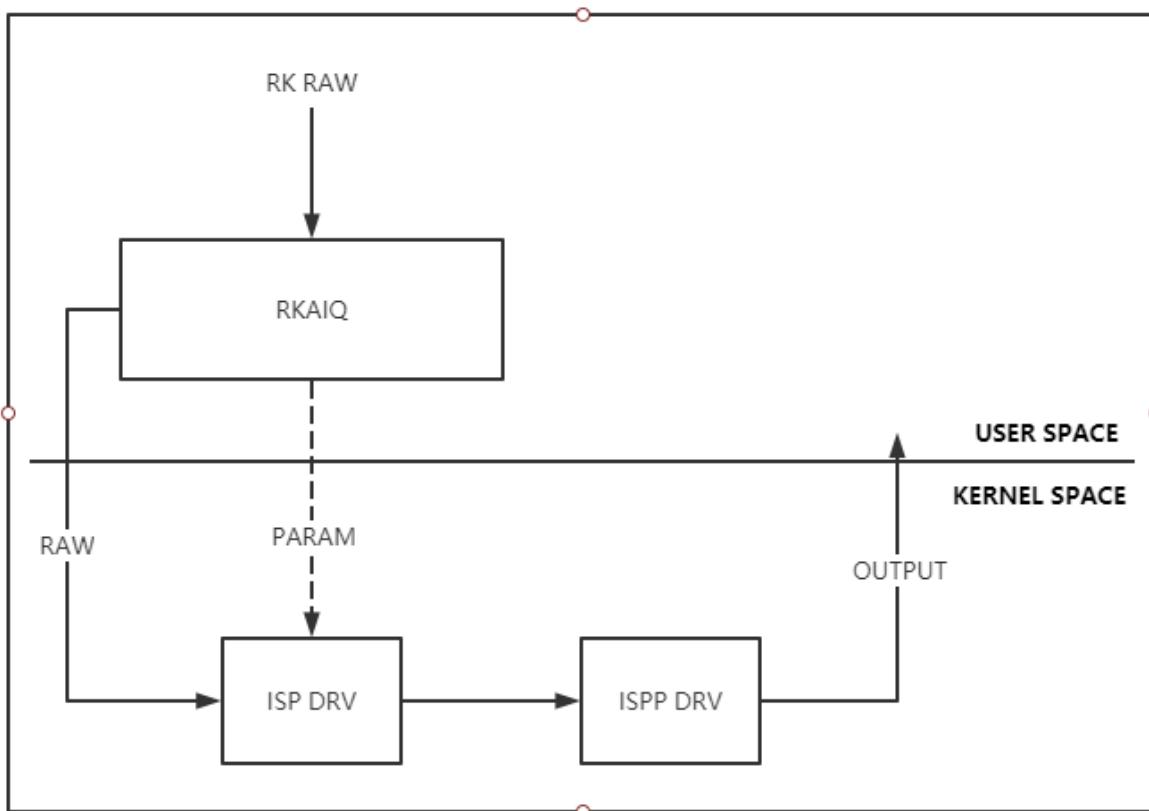
| 成员名称 | 描述 |
|--------|--------------------------|
| left | horizontal output offset |
| top | vertical output offset |
| width | horizontal output size |
| height | vertical output size |

离线帧处理

概述

RKAIQ提供离线RAW帧处理功能，即RK自定义的RK-Raw V2.0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 V2.0格式文件输入。
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 V2.0格式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 V2.0格式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。

如何获取RK RAW V2.0格式文件

目前仅供快速启动SDK内部使用，暂未实现RK RAW V2.0格式文件抓取的API接口。格式说明参考[RK-Raw V2.0](#)

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_uapi_sysctl_prepareRkRaw

【描述】

准备RK Raw格式数据处理的运行环境。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_prepareRkRaw(const rk_aiq_sys_ctx_t* ctx, rk_aiq_raw_prop_t  
prop);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|------|--------------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| prop | RK RAW V2.0格式的属性参数 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 该接口须在rk_aiq_uapi_sysctl_prepare之前调用。

rk_aiq_uapi_sysctl_enqueueRkRawBuf

【描述】

输入RK Raw格式的buffer

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_enqueueRkRawBuf(const rk_aiq_sys_ctx_t* ctx, void *rawdata,  
bool sync);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------------------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| rawdata | RK RAW V2.0格式的数据buffer | 输入 |
| sync | true: 同步模式。 false: 异步模式 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|------------|
| 0 | 成功 |
| 非0 | 失败, 详见错误码表 |

【注意】

- 若需要在异步模式下对使用结束的rawdata进行一些操作, 可以使用 rk_aiq_uapi_sysctl_registRkRawCb接口注册回调函数, 使用后的rawdata地址就会被传入该回调函数。

rk_aiq_uapi_sysctl_enqueueRkRawFile

【描述】

输入RK Raw 格式的文件

【语法】

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

【参数】

| 参数名称 | 描述 | 输入/输出 |
|------|-------------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| path | RK RAW V2.0格式文件路径 | 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】

- 该接口为同步接口。

rk_aiq_uapi_sysctl_registRkRawCb

【描述】

注册RK Raw处理完成后的回调接口

【语法】

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

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| callback | 回调函数指针 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 该接口不是必须的。
- 如果注册了回调，那么该回调只在rk_aiq_uapi_sysctl_enqueueRkRawBuf接口设置为异步模式时且Raw数据处理完成后被调用。

数据结构

rk_aiq_raw_prop_t

【说明】

RK Raw属性参数结构体

【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 | RK Raw图像宽 |
| frame_height | RK Raw图像高 |
| format | RK Raw的bayer pattern格式 |
| rawbuf_type | RK Raw格式的数据类型 |

rk_aiq_rawbuf_type_t

【说明】

RK Raw数据类型结构体

【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 | 表示输入的RK Raw格式buffer中'Raw数据'段存放的是本进程中DMA BUF的虚拟地址，而非RAW图像数据 |
| RK_AIQ_RAW_FD | 表示输入的RK Raw格式buffer中'Raw数据'段存放的是本进程中的BUF fd，而非RAW图像数据 |
| RK_AIQ_RAW_DATA | 表示输入的RK Raw格式buffer中'Raw数据'段存放的是Raw图像数据 |
| RK_AIQ_RAW_FILE | 表示输入的是RK Raw格式文件 |

注意事项

- 使用RK Raw数据处理功能，在创建AIQ Context时，rk_aiq_uapi_sysctl_init接口的参数sns_ent_name须为“FakeCamera”。
- Raw数据的处理依赖IQ XML效果文件，XML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.xml，放置于XML文件的加载路径下。
- 如果不接real camera，想要使用RK Raw数据处理功能，那么kernel dts中的ISP节点里不要配置输入port。
- 若当前已配置一个real camera sensor，假设real camera sensor连接在ISP0&ISPP0上；如果要使用RK Raw数据处理功能，那么需要在kernel中使能ISP1&ISPP1，使得创建的fake camera sensor能够正常使用。
- 如果real camera sensor和fake camera sensor需要同时使用并且其分辨率不同，那么kernel dts文件中的ISPP节点里需要配置最大分辨率信息，具体配置方式请参见文档《Rockchip_Driver_Guide_ISP2x_CN》。

参考示例

离线帧处理API的使用方法请参考rkisp_demo，路径为
YOUR_SDK_DIR/external/camera_engine_rkaiq/rkisp_demo。

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

rk_aiq_uapi_setExpMode

【描述】

设置曝光模式。

【语法】

```
XCamReturn rk_aiq_uapi_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 |

【需求】

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

rk_aiq_uapi_getExpMode

【描述】

获取曝光模式。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setAeMode

【描述】

设置AE工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_setAeMode(const rk_aiq_sys_ctx_t* ctx, aeMode_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 | 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】

- 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。如果切换手动模式同时需要设置曝光值，可以使用rk_aiq_uapi_setManualExp接口。

【需求】

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

rk_aiq_uapi_getAeMode

【描述】

获取AE工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_getAeMode(const rk_aiq_sys_ctx_t* ctx, aeMode_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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setManualExp

【描述】

使用手动曝光模式，并且设置增益和曝光时间。

【语法】

```
XCamReturn rk_aiq_uapi_setManualExp(const rk_aiq_sys_ctx_t* ctx, float gain,  
float time);
```

【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 |
| time | 曝光时间 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setExpGainRange

【描述】

设置增益范围。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getExpGainRange

【描述】

获取增益范围。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setExpTimeRange

【描述】

设置曝光时间范围。

【语法】

```
XCamReturn rk_aiq_uapi_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 |

【需求】

- 头文件: rk_aiq_user_api_imgproc.h

- 库文件: librkaiq.so

rk_aiq_uapi_getExpTimeRange

【描述】

获取曝光时间范围。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setBLCMode

【描述】

背光补偿开关、区域设置。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setBLCStrength

【描述】

设置暗区提升强度。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setHLCMode

【描述】

强光抑制开关。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_SetHLCStrength

【描述】

设置强光抑制强度。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setDarkAreaBoostStrth

【描述】

设置暗区提升强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 暗区提升强度, 范围[1,10] | 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有效。

【需求】

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

rk_aiq_uapi_getDarkAreaBoostStrth

【描述】

获取当前暗区提升强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 暗区提升强度 | 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】

- 该接口仅在线性模式下有效。

【需求】

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

rk_aiq_uapi_setAntiFlickerMode

【描述】

设置抗闪模式。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getAntiFlickerMode

【描述】

获取抗闪模式。

【语法】

```
XCamReturn rk_aiq_uapi_getAntiFlickerMode(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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setExpPwrLineFreqMode

【描述】

设置抗闪频率。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getExpPwrLineFreqMode

【描述】

获取抗闪频率。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

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_MANUAL1 = 1,  
    OP_INVAL  
} opMode_t;
```

【成员】

| 成员名称 | 描述 |
|------------|------|
| OP_AUTO | 自动模式 |
| OP_MANUAL1 | 手动模式 |
| OP_INVAL | 无效值 |

aeMode_t

【说明】

定义AE工作模式

【定义】

```
typedef enum aeMode_e {
    AE_AUTO = 0,
    AE_IRIS_PRIOR = 1,
    AE_SHUTTER_PRIOR = 2
} aeMode_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|------|
| OP_AUTO | 自动选择 |
| AE_IRIS_PRIOR | 光圈优先 |
| AE_SHUTTER_PRIOR | 快门优先 |

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 |

模块级API参考

rk_aiq_user_api_ae_setExpSwAttr

【描述】

设定 AE曝光软件属性。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_setExpSwAttr(const rk_aiq_sys_ctx_t* ctx,  
                                    const Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.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、光圈。设置手动模式之后，还需要分别设置各曝光分量的手动状态 (ManualGainEn、ManualTimeEn、ManuallspDgainEn、ManuallIrisEn) 及对应手动值。手动模式下，要求至少有一个曝光分量为手动状态，否则将报错退出。手动模式下设置的各曝光分量的值，会受到sensor及镜头的限制。如设置的曝光分量值超过sensor及镜头的限制，算法内部会自动进行校正。

```
Uapi_ExpSwAttr_t expSwAttr;  
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);  
expSwAttr.AecOpType = RK_AIQ_OP_MODE_MANUAL;  
//LinearAE  
expSwAttr.stManual.stLinMe.ManualGainEn = true;  
expSwAttr.stManual.stLinMe.ManualTimeEn = true;  
expSwAttr.stManual.stLinMe.ManualIspDgainEn = true;  
expSwAttr.stManual.stLinMe.ManualIrisEn = true;  
expSwAttr.stManual.stLinMe.GainValue = 1.0f; /*gain = 1x*/  
expSwAttr.stManual.stLinMe.TimeValue = 0.02f; /*time = 1/50s*/  
expSwAttr.stManual.stLinMe.IspDGainValue = 1.0f; /*IspDgain = 1x*/  
expSwAttr.stManual.stLinMe.PIrisGainValue = 512; /*p-iris F#=1.4*/
```

```

expSwAttr.stManual.stLinMe.DCIRisValue = 20; /*dc-iris PwmDuty=20*/
//HdrAE
expSwAttr.stManual.stHdrMe.ManualGainEn = true;
expSwAttr.stManual.stHdrMe.ManualTimeEn = true;
expSwAttr.stManual.stHdrMe.ManualIspDgainEn = true;
expSwAttr.stManual.stHdrMe.ManualIrisEn = true;
expSwAttr.stManual.stHdrMe.Gainvalue[0] = 1.0f; /*sframe gain = 1x*/
expSwAttr.stManual.stHdrMe.Timevalue[0] = 0.02f; /*sframe time = 1/50s*/
expSwAttr.stManual.stHdrMe.IspDGainValue[0] = 1.0f; /*sframe IspDgain = 1x*/
expSwAttr.stManual.stHdrMe.PIrisGainValue = 512; /*p-iris F#=1.4*/
expSwAttr.stManual.stHdrMe.DCIRisValue = 20; /*dc-iris PwmDuty=20*/

ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

```

- 设置自动曝光属性

```

Uapi_ExpSwAttr_t expSwAttr;
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.AecOpType = RK_AIQ_OP_MODE_AUTO;

//set gain range
expSwAttr.stAuto.SetAeRangeEn = true; /*必须使能*/
expSwAttr.stAuto.stLinAeRange.stGainRange.Max = 32.0f; /*gain_max = 32x*/
expSwAttr.stAuto.stLinAeRange.stGainRange.Min = 1.0f; /*gain_min = 1x*/
expSwAttr.stAuto.stHdrAeRange.stGainRange[0].Max = 32.0f; /*sframe gain_max = 32x*/
expSwAttr.stAuto.stHdrAeRange.stGainRange[0].Min = 1.0f; /*sframe gain_min = 1x*/
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

//set ae speed
expSwAttr.stAuto.stAeSpeed.DampOver = 0.2f;
expSwAttr.stAuto.stAeSpeed.DampUnder = 0.5f;
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

//set fixed framemode
expSwAttr.stAuto.stFrmRate.isFpsFix = true;
expSwAttr.stAuto.stFrmRate.FpsValue = 25; /*fps = 25*/
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);
//set auto framemode
expSwAttr.stAuto.stFrmRate.isFpsFix = false;
/*一般自动降帧模式由tuning人员事先配置好最低帧率和切换帧率对应的gain值*/

//set ae delay
expSwAttr.stAuto.BlackDelayFrame = 2;
expSwAttr.stAuto.WhiteDelayFrame = 4;
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

```

- 设置光圈控制参数

```

Uapi_ExpSwAttr_t expSwAttr;
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.stIris.enable = true; /*run AIris*/

//set P-iris attributes
expSwAttr.stIris.IrisType = IRIS_P_TYPE;

```

```

expSwAttr.stIris.PIrisAttr.TotalStep = 81;
expSwAttr.stIris.PIrisAttr.EffcStep = 44;
expSwAttr.stIris.PIrisAttr.ZeroIsMax = 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, 0};
memcpy(expSwAttr.stIris.PIrisAttr.StepTable, StepTable, sizeof(expSwAttr.stIris.PIrisAttr.StepTable));
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

//set DC-iris attributes
expSwAttr.stIris.IrisType = IRIS_DC_TYPE;
expSwAttr.stIris.DCIrisAttr.Kp= 0.5f;
expSwAttr.stIris.DCIrisAttr.Ki= 0.2f;
expSwAttr.stIris.DCIrisAttr.Kd = 0.3f;
expSwAttr.stIris.DCIrisAttr.OpenPwmDuty = 40;
expSwAttr.stIris.DCIrisAttr.ClosePwmDuty = 22;
expSwAttr.stIris.DCIrisAttr.MinPwmDuty = 0;
expSwAttr.stIris.DCIrisAttr.MaxPwmDuty = 100;
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

//set manual iris with auto ae
expSwAttr.AecOpType = RK_AIQ_OP_MODE_MANUAL;
expSwAttr.stManual.stLinMe.ManualGainEn = false;
expSwAttr.stManual.stLinMe.ManualTimeEn = false;
expSwAttr.stManual.stLinMe.ManualIspDgainEn = false;
expSwAttr.stManual.stLinMe.ManualIrisEn = true;
if(expSwAttr.stIris.IrisType == IRIS_P_TYPE);
    expSwAttr.stManual.stLinMe.PIrisGainValue = 512; /*p-iris F#=1.4*/
if(expSwAttr.stIris.IrisType == IRIS_DC_TYPE);
    expSwAttr.stManual.stLinMe.DCIrisValue = 20; /*dc-iris PwmDuty=20*/

```

- 设置抗闪功能

```

Uapi_ExpSwAttr_t expSwAttr;
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);

//set antifilicker mode
expSwAttr.stAntiFlicker.enable = true;
expSwAttr.stAntiFlicker.Frequency = AEC_FLICKER_FREQUENCY_50HZ;
expSwAttr.stAntiFlicker.Mode = AEC_ANTIFLICKER_AUTO_MODE;
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

```

- 设置AE权重

设置5X5权重，算法内部根据硬件实际分块规格，进行权重的扩展

```

Uapi_ExpSwAttr_t expSwAttr;
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);

```

```

//set aec 5x5 weight
uint8_t DayGridWeights[25]={
0, 1, 1, 1, 0,
1, 2, 3, 2, 1,
3, 5, 7, 5, 3,
2, 3, 4, 4, 2,
1, 2, 2, 2, 1
};
uint8_t NightGridWeights[25]={
0, 1, 1, 1, 0,
1, 2, 3, 2, 1,
3, 5, 7, 5, 3,
2, 3, 4, 4, 2,
1, 2, 2, 2, 1
};
for(int i=0;i<25;i++){
    expSwAttr.DayGridWeights.uCoeff[i] = DayGridWeights[i];
    expSwAttr.NightGridWeights.uCoeff[i] = NightGridWeights[i];
}
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

```

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

```

Uapi_ExpSwAttr_t expSwAttr;
ret = rk_aiq_user_api_ae_getExpSwAttr(ctx, &expSwAttr);
expSwAttr.stAdvanced.enable = true; //important! true means preferring to use
these parameters
uint8_t DayGridWeights[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,
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, 3, 5, 7, 8, 9, 10, 11, 10, 9, 8, 7, 5, 3, 2,
2, 3, 5, 7, 7, 8, 9, 10, 11, 10, 9, 8, 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
};

uint8_t NightGridWeights[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, 3, 5, 7, 8, 9, 10, 11, 10, 9, 8, 7, 5, 3, 2,
2, 3, 5, 7, 7, 8, 9, 10, 11, 10, 9, 8, 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
};

memcpy(expSwAttr.stAdvanced.DayGridWeights,DayGridWeights,sizeof(expSwAttr.stAdvanced.DayGridWeights));
memcpy(expSwAttr.stAdvanced.NightGridweights,NightGridweights,sizeof(expSwAttr.stAdvanced.DayGridweights));

ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);

```

rk_aiq_user_api_ae_getExpSwAttr

【描述】

获取 AE 曝光软件属性。

【语法】

```

XCamReturn
rk_aiq_user_api_ae_getExpSwAttr(const rk_aiq_sys_ctx_t* ctx,
                                  Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_setLinAeDayRouteAttr

【描述】

设置线性模式下AE的白天场景曝光分配策略。

【语法】

```

XCamReturn
rk_aiq_user_api_ae_setLinAeDayRouteAttr(const rk_aiq_sys_ctx_t* ctx, const
Uapi_LinAeRouteAttr_t linAeRouteAttr);

```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| linAeRouteAttr | AE曝光分配策略结构体 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

【举例】

```

Uapi_LinAeRouteAttr_t stLinDayRoute;
ret = rk_aiq_user_api_ae_getLinAeDayRouteAttr(ctx,&stLinDayRoute);

float TimeDot[6] = {0.0, 0.03, 0.03, 0.03, 0.03, 0.03};
float GainDot[6] = {1, 1, 4, 4, 8, 8};
float LinIspDGainDot[6] = {1, 1, 1, 5, 5, 5};
int LinPirisGainDot[6] = {1, 1, 1, 1, 1, 2};

stLinDayRoute.array_size = 6;
for(int i =0; i < stLinDayRoute.array_size; i++){
    stLinDayRoute.TimeDot[i]=TimeDot[i];
    stLinDayRoute.GainDot[i]=GainDot[i];
    stLinDayRoute.IspGainDot[i]=LinIspDGainDot[i];
    stLinDayRoute.PirisGainDot[i]=LinPirisGainDot[i];
}
ret = rk_aiq_user_api_ae_setLinAeDayRouteAttr(ctx,stLinDayRoute);

```

rk_aiq_user_api_ae_getLinAeDayRouteAttr

【描述】

获取线性模式下AE的白天场景曝光分配策略。

【语法】

```

XCamReturn
rk_aiq_user_api_ae_getLinAeDayRouteAttr(const rk_aiq_sys_ctx_t* ctx,
Uapi_LinAeRouteAttr_t* pLinAeRouteAttr);

```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|---------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| pLinAeRouteAttr | AE曝光分配策略结构体指针 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_setHdrAeDayRouteAttr

【描述】

设置HDR模式下AE的白天场景曝光分配策略。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_setHdrAeDayRouteAttr(const rk_aiq_sys_ctx_t* ctx, const  
uapi_HdrAeRouteAttr_t hdrAeRouteAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| hdrAeRouteAttr | AE曝光分配策略结构体 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

【举例】

```
uapi_HdrAeRouteAttr_t stHdrDayRoute;  
ret = rk_aiq_user_api_ae_getHdrAeDayRouteAttr(ctx,&stHdrDayRoute);  
float HdrTimeDot[3][6] = {0.0, 0.01, 0.01, 0.01, 0.01, 0.01,  
                          0.0, 0.02, 0.02, 0.02, 0.02, 0.02,  
                          0.0, 0.03, 0.03, 0.03, 0.03, 0.03  
};  
float HdrGainDot[3][6] = {1, 1, 4, 6, 8, 12,  
                         1, 1, 4, 6, 8, 12,  
                         1, 1, 4, 6, 8, 12
```

```

    };
float HdrIspDGainDot[3][6] = {1, 1, 1, 1, 1, 1,
                               1, 1, 1, 1, 1, 1,
                               1, 1, 1, 1, 1, 1
                           };
int HdrPIrisGainDot[6] = {1, 1, 1, 1, 1, 1};

stHdrDayRoute.array_size = 6;
for(int j = 0; j < stHdrDayRoute.array_size; j++){
    for(int i = 0; i < 3; i++){
        stHdrDayRoute.TimeDot[i][j]=HdrTimeDot[i][j];
        stHdrDayRoute.GainDot[i][j]=HdrGainDot[i][j];
        stHdrDayRoute.IspgainDot[i][j]=HdrIspDGainDot[i][j];
    }
    stHdrDayRoute.PIrisGainDot[j]=HdrPIrisGainDot[j];
}
ret = rk_aiq_user_api_ae_setHdrAeDayRouteAttr(ctx,stHdrDayRoute);

```

rk_aiq_user_api_ae_getHdrAeDayRouteAttr

【描述】

获取HDR模式下AE的白天场景曝光分配策略。

【语法】

```

XCamReturn
rk_aiq_user_api_ae_getHdrAeDayRouteAttr(const rk_aiq_sys_ctx_t* ctx,
Uapi_HdrAeRouteAttr_t* pHdrAeRouteAttr);

```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|---------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| pHdrAeRouteAttr | AE曝光分配策略结构体指针 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_setLinAeNightRouteAttr

【描述】

设置线性模式下AE的夜晚场景曝光分配策略。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_setLinAeNightRouteAttr(const rk_aiq_sys_ctx_t* ctx, const  
Uapi_LinAeRouteAttr_t linAeRouteAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| linAeRouteAttr | AE曝光分配策略结构体 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_getLinAeNightRouteAttr

【描述】

获取线性模式下AE的夜晚场景曝光分配策略。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_getLinAeNightRouteAttr(const rk_aiq_sys_ctx_t* ctx,  
Uapi_LinAeRouteAttr_t* pLinAeRouteAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|---------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| pLinAeRouteAttr | AE曝光分配策略结构体指针 | 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_api_ae.h、rk_aiq_uapi_ae_int.h

- 库文件: librkaiq.so

rk_aiq_user_api_ae_setHdrAeNightRouteAttr

【描述】

设置HDR模式下AE的夜晚场景曝光分配策略。

【语法】

```
XCamReturn
rk_aiq_user_api_ae_setHdrAeNightRouteAttr(const rk_aiq_sys_ctx_t* ctx, const
Uapi_HdrAeRouteAttr_t hdrAeRouteAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| hdrAeRouteAttr | AE曝光分配策略结构体 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_getHdrAeNightRouteAttr

【描述】

获取HDR模式下AE的夜晚场景曝光分配策略。

【语法】

```
XCamReturn
rk_aiq_user_api_ae_getHdrAeNightRouteAttr(const rk_aiq_sys_ctx_t* ctx,
Uapi_HdrAeRouteAttr_t* pHdrAeRouteAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|---------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| pHdrAeRouteAttr | AE曝光分配策略结构体指针 | 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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_queryExpResInfo

【描述】

获取 AE 内部状态信息。

【语法】

```
XCamReturn
rk_aiq_user_api_ae_queryExpResInfo(const rk_aiq_sys_ctx_t* ctx,
Uapi_ExpQueryInfo_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_setLinExpAttr

【描述】

设置AE线性模式曝光参数。

【语法】

```
XCamReturn
rk_aiq_user_api_ae_setLinExpAttr(const rk_aiq_sys_ctx_t* ctx, const
Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_getLinExpAttr

【描述】

获取AE线性模式曝光参数。

【语法】

```
XCamReturn
rk_aiq_user_api_ae_getLinExpAttr(const rk_aiq_sys_ctx_t* ctx, Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_setHdrExpAttr

【描述】

设置AE HDR模式曝光参数。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_setHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, const  
Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_ae_getHdrExpAttr

【描述】

获取AE HDR模式曝光参数。

【语法】

```
XCamReturn  
rk_aiq_user_api_ae_getHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, Uapi_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_api_ae.h、rk_aiq_uapi_ae_int.h
- 库文件: librkaiq.so

模块级API数据类型

Uapi_ExpswAttr_t

【说明】

AE公共功能控制参数结构体

【定义】

```
typedef struct Uapi_ExpswAttr_s {
    uint8_t enable;
    CalibDb_CamRawStatsMode_t RawStatsMode;
    CalibDb_CamHistStatsMode_t HistStatsMode;
    CalibDb_CamYRangeMode_t YRangeMode;
    uint8_t AecRunInterval;
    RKAiqOPMode_t AecOpType;
    Cam5x5UCharMatrix_t DayGridWeights;
    Cam5x5UCharMatrix_t NightGridWeights;
    uint8_t DNTrigger;
    Uapi_IrisAttr_t stIris;
    Uapi_AntiFlicker_t stAntiFlicker;
    Uapi_AeAttr_t stAuto;
    Uapi_MeAttr_t stManual;
    Uapi_ExpInitExp_t stInitExp;
    Uapi_ExpswAttr_Advanced_t stAdvanced; //special for Aeweight setting
} Uapi_ExpswAttr_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|--|
| enable | 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。 |
| RawStatsMode | Aec模块亮度统计模式。共四种模式分别为：Y/R/G/B，默认为Y模式。 |
| HistStatsMode | Aec模块直方图统计模式。共五种模式分别为：Y/R/G/B/RGB，默认为Y模式。 |
| YRangeMode | 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。 |
| AecRunInterval | 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。 |
| AecOpType | 曝光模式，分为自动曝光(AUTO)模式/手动(MANUAL)曝光模式。默认为AUTO模式。手动曝光模式需要与stManual一起配合，进行手动曝光值的设置。 |
| DayGridWeights | day模式下窗口(直方图)权重，包含5*5个数组元素 |
| NightGridWeights | night模式下窗口(直方图)权重，包含5*5个数组元素 |
| DNTrigger | AEC参数切换使能。0：AEC参数不随白天、夜晚模式进行切换，默认固定使用白天参数；1：AEC参数随白天、夜晚模式进行切换。 |
| stIris | 光圈控制参数结构体 |
| stAntiFlicker | 抗工频闪烁参数结构体 |
| stInitExp | 曝光初始值结构体 |
| stAuto | 自动曝光参数结构体 |
| stManual | 手动曝光参数结构体 |
| stAdvanced | 优先使用参数结构体 |

【相关数据类型】

- Uapi_ExpSwAttr_Advanced_t
- Uapi_IrisAttr_t
- Uapi_AntiFlicker_t
- Uapi_AeAttr_t
- Uapi_MeAttr_t
- Uapi_ExplInitExp_t

Uapi_ExpSwAttr_Advanced_t

【说明】

优先使用参数结构体

【定义】

```

#define RAWAEBIG_WIN_NUM      225

typedef struct Aec_uapi_advanced_attr_s {
    bool      enable;
    uint8_t  DayGridWeights[RAWAEBIG_WIN_NUM];
    uint8_t  NightGridWeights[RAWAEBIG_WIN_NUM];
} Aec_uapi_advanced_attr_t;

typedef Aec_uapi_advanced_attr_t Uapi_ExpSwAttr_Advanced_t;

```

【注意事项】

- Uapi_ExpSwAttr_t结构体中定义了一组AE权重，权重个数为5X5。算法内部根据硬件实际配置权重个数，进行权重的扩展。如需要设置精度更高的权重，可使用Uapi_ExpSwAttr_Advanced_t中定义的一组AE权重，其权重个数为15X15。算法内部根据硬件实际配置权重个数，进行权重的压缩。
- 使用Uapi_ExpSwAttr_Advanced_t中定义的一组AE权重，需要打开使能enable，将其置1。

Uapi_IrisAttr_t

【说明】

光圈控制参数

【定义】

```

typedef struct CalibDb_AecIrisCtrl_s {
    uint8_t          enable;
    CalibDb_IrisType_t   IrisType;
    CalibDb_PIris_Attr_t  PIRisAttr;
    CalibDb_DCiris_Attr_t  DCIrisAttr;
} CalibDb_AecIrisCtrl_t;
typedef CalibDb_AecIrisCtrl_t Uapi_IrisAttr_t;

```

【成员】

| 成员名称 | 描述 |
|------------|----------------------------------|
| Enable | 自动光圈功能的使能 |
| IrisType | 光圈类型，P（即P-iris光圈）或DC（即DC-iris光圈） |
| PIrisAttr | P光圈属性参数 |
| DCIrisAttr | DC光圈属性参数 |

CalibDb_PIris_Attr_t

【说明】

P光圈属性参数

【定义】

```

#define AEC_PIRIS_STAP_TABLE_MAX (1024)
typedef struct CalibDb_PIris_Attr_s {
    uint16_t          TotalStep;
    uint16_t          EffcStep;
    bool              ZeroIsMax;
    uint16_t          StepTable[AEC_PIRIS_STAP_TABLE_MAX];
} CalibDb_PIris_Attr_t;

```

【成员】

| 成员名称 | 描述 |
|-----------|--|
| TotalStep | P-iris步进电机总步数，具体大小与P-iris镜头有关。 |
| EffcStep | P-iris步进电机的可用步数，具体大小与P-iris镜头有关。 |
| ZerolsMax | 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。 |
| StepTable | 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。 |

CalibDb_DC Iris_Attr_t

【说明】

DC光圈属性

【定义】

```

typedef struct CalibDb_DC Iris_Attr_s {
    float      Kp;
    float      Ki;
    float      Kd;
    int        MinPwmDuty;
    int        MaxPwmDuty;
    int        OpenPwmDuty;
    int        ClosePwmDuty;
} CalibDb_DC Iris_Attr_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|---|
| 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。建议值为0.5。取值范围[0, 1]。 |
| 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。建议值为0.2。取值范围[0, 1]。 |
| 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。建议值为0.3。取值范围[0, 1]。 |
| MinPwmDuty | 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。 |
| MaxPwmDuty | 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。 |
| OpenPwmDuty | 光圈打开时的PWM占空比阈值, 当光圈PWM占空比高于(不含)OpenPwmDuty时, 光圈处于打开状态。具体大小与DC-iris镜头有关, 单位为%。 |
| ClosePwmDuty | 光圈关闭时的PWM占空比阈值, 当光圈PWM占空比小于(不含)ClosePwmDuty时, 光圈处于关闭状态。具体大小与DC-iris镜头有关, 单位为%。 |

【注意事项】

- 自动光圈功能关闭时, 对于DC-iris光圈, 默认会打开到最大; 对于P-iris光圈, 默认会打开到最大光圈所对应的步进电机位置。如想改变上述光圈位置, 可至AecInitValue模块中修改InitPIrisGainValue、InitDCIrisDutyValue。
- 自动光圈Airis算法的基本控制流程如下:

针对DC-iris镜头, Airis根据当前亮度与目标亮度的偏差值, 控制DC-iris镜头的光圈大小。当曝光达到最小值时, 且当前亮度超出目标亮度容忍度范围, 将退出AE控制, 曝光时间及曝光增益固定不变, 进入AIris控制范围。若当前画面亮度稳定且DC-iris的PWM占空值大于OpenPwmDuty时, 认为当前光圈达到最大, 退出AIris光圈控制, 控制权交由AE。

针对P-iris镜头, 光圈控制通过AecRoute模块进行。P-iris镜头的光圈大小换算为等效增益, 参与曝光分解计算。
- P-iris的步进电机位置与光圈等效增益映射表StepTable一般根据镜头厂家提供的步进电机位置与光圈孔径对应关系制作。P-iris的控制是通过AE的AecRoute模块来控制的, 该模块将光圈孔径大小换算成等效增益, 因此要求P-iris的光圈控制需要具有较好的线性度。等效增益的取值范围为[1,1024], 用等效增益1024表示F1.0, 等效增益512表示F1.4, 以此类推, 等效增益1表示F32.0。制作表时, 需要将步进电机位置对应的光圈孔径换算为等效增益, 填入StepTable中, 并固定按照步进电机位置递增(即step0、step1.....stepN)的顺序填入。
- TotalStep表示P-iris步进电机总步数, 具体大小与P-iris镜头有关。EffcStep表示P-iris步进电机的可用步数, 一般要求小于TotalStep。因为靠近光圈关闭端的位置, 其对应等效增益的值误差较大, 光圈调节过程中容易出现振荡, 所以通常不会使用光圈关闭端附近的步进位置。

- 表4-1为P-iris步进电机位置与光圈孔径和等效增益的对应表，以此表为例来说明StepTable该如何设置。表4-1中第1-2、4-5列的步进电机位置step和光圈孔径面积的对应关系为某镜头原厂提供。该款P-iris镜头的步进电机调节总步数为81，step0时对应的光圈孔径最大，标称最大光圈数为1.4。光圈数为1.4时对应的等效增益为512，故step0处对应的等效增益为512。其他孔径面积对应的等效增益，此处以step3为例，计算方式如下：step3的孔径面积为195.869，对应等效增益= $512 * (195.869 / 201.062) = 499$ （四舍五入）。以此类推，其他步进电机位置对应的等效增益值也可据此算出。从表1-1中可知，步进电机位置靠近关闭端时，对应的孔径面积很小，与最大的孔径面积相差可达几千倍，对应的等效增益值误差较大，因此建议靠近光圈关闭端的步进电机位置不要使用，以免因为误差导致曝光振荡。将表中各步进电机位置对应的等效增益按照步进电机位置递增（即step0、step1.....stepN）的顺序填入StepTable。
- DC-iris的OpenPwmDuty与ClosePwmDuty取值需要进行实测，其具体值与DC-iris镜头相关。对于部分镜头，存在当PWM占空比大于OpenPwmDuty时，光圈执行打开操作；当PWM占空比小于OpenPwmDuty时，光圈执行关闭操作；当PWM占空比大于等于ClosePwmDuty且小于等于OpenPwmDuty时，光圈稳定在当前位置，该区间内的值皆为HoldValue。另存在某些镜头，只存在一个光圈开关的阈值，即当PWM占空比大于该阈值时，光圈执行打开操作；当PWM占空比小于该阈值时，光圈执行关闭操作；当PWM占空比等于该阈值时，光圈稳定在当前位置，该阈值即为HoldValue。此时可令ClosePwmDuty = OpenPwmDuty = HoldValue。
- 光圈的手动模式参数设置与曝光的手动模式一致。需要使用手动光圈功能时，需将AecOpType设置为手动模式，并使能AecManualCtrl模块中的ManuallRisEn参数。当IrisType为P-iris时，仅PirisGainValue有效；当IrisType为P-iris时时，仅DCIrisValue有效。

表4-1 P-iris步进电机位置与光圈孔径和等效增益的对应表

| Step | 孔径面积(mm ²) | 等效增益 | Step | 孔径面积(mm ²) | 等效增益 |
|------|------------------------|------|------|------------------------|------|
| 0 | 201.062 | 512 | 41 | 56.653 | 144 |
| 1 | 200.759 | 511 | 42 | 53.438 | 136 |
| 2 | 198.583 | 506 | 43 | 50.282 | 128 |
| 3 | 195.869 | 499 | 44 | 47.188 | 120 |
| 4 | 192.879 | 491 | 45 | 44.159 | 112 |
| 5 | 189.677 | 483 | 46 | 41.197 | 105 |
| 6 | 186.293 | 474 | 47 | 38.307 | 98 |
| 7 | 182.744 | 465 | 48 | 35.49 | 90 |
| 8 | 179.035 | 456 | 49 | 32.751 | 83 |
| 9 | 175.271 | 446 | 50 | 30.093 | 77 |
| 10 | 171.484 | 437 | 51 | 27.519 | 70 |
| 11 | 167.681 | 427 | 52 | 25.034 | 64 |
| 12 | 163.865 | 417 | 53 | 22.642 | 58 |
| 13 | 160.036 | 408 | 54 | 20.347 | 52 |
| 14 | 156.198 | 398 | 55 | 18.154 | 46 |
| 15 | 152.351 | 388 | 56 | 16.068 | 41 |
| 16 | 148.499 | 378 | 57 | 14.096 | 36 |
| 17 | 144.642 | 368 | 58 | 12.245 | 31 |
| 18 | 140.783 | 359 | 59 | 10.522 | 27 |
| 19 | 136.925 | 349 | 60 | 8.935 | 23 |
| 20 | 133.069 | 339 | 61 | 7.484 | 19 |
| 21 | 129.217 | 329 | 62 | 6.169 | 16 |
| 22 | 125.371 | 319 | 63 | 4.987 | 13 |
| 23 | 121.535 | 309 | 64 | 3.936 | 10 |
| 24 | 117.709 | 300 | 65 | 3.014 | 8 |
| 25 | 113.897 | 290 | 66 | 2.22 | 6 |
| 26 | 110.1 | 280 | 67 | 1.55 | 4 |
| 27 | 106.321 | 271 | 68 | 1.003 | 3 |
| 28 | 102.562 | 261 | 69 | 0.577 | 1 |
| 29 | 98.826 | 252 | 70 | 0.268 | 1 |

| Step | 孔径面积(mm ²) | 等效增益 | Step | 孔径面积(mm ²) | 等效增益 |
|------|------------------------|------|------|------------------------|------|
| 30 | 95.115 | 242 | 71 | 0.075 | 0 |
| 31 | 91.431 | 233 | 72 | close | 0 |
| 32 | 87.777 | 224 | 73 | close | 0 |
| 33 | 84.156 | 214 | 74 | close | 0 |
| 34 | 80.569 | 205 | 75 | close | 0 |
| 35 | 77.02 | 196 | 76 | close | 0 |
| 36 | 73.51 | 187 | 77 | close | 0 |
| 37 | 70.043 | 178 | 78 | close | 0 |
| 38 | 66.621 | 170 | 79 | close | 0 |
| 39 | 63.247 | 161 | 80 | close | 0 |
| 40 | 59.923 | 153 | | | |

Uapi_AntiFlicker_t

【说明】

定义抗闪属性

【定义】

```
typedef struct CalibDb_AntiFlickerAttr_s {
    bool enable;
    CalibDb_FlickerFreq_t Frequency;
    CalibDb_AntiFlickerMode_t Mode;
} CalibDb_AntiFlickerAttr_t;
typedef CalibDb_AntiFlickerAttr_t Uapi_AntiFlicker_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|---|
| enable | 工频抗闪功能使能状态，0：关闭抗闪功能；1：开启抗闪功能 |
| Frequency | 抗闪频率属性，共包含3种帧率，分别为：AEC_FLICKER_FREQUENCY_OFF（抗闪使能位enable置0时有效）、AEC_FLICKER_FREQUENCY_50HZ、AEC_FLICKER_FREQUENCY_60HZ，默认为AEC_FLICKER_FREQUENCY_50HZ（工频50赫兹）。 |
| Mode | 抗闪模式，共包含两种抗闪模式：AEC_ANTIFLICKER_NORMAL_MODE（普通抗闪模式）、AEC_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enable置0，算法内部自动配置Frequency=AEC_FLICKER_FREQUENCY_OFF；如抗闪使能位enable置1时，抗闪频率配置为AEC_FLICKER_FREQUENCY_OFF，该频率值将配置无效，使用默认值AEC_FLICKER_FREQUENCY_50HZ。
- AEC_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。
- AEC_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。

Uapi_AeAttr_t

【说明】

定义AE自动曝光属性

【定义】

```
typedef struct CalibDb_AeAttr_s {
    CalibDb_AeSpeed_t          stAeSpeed;
    uint8_t                  BlackDelayFrame;
    uint8_t                  WhiteDelayFrame;
    bool                     SetAeRangeEn;
    CalibDb_LinAeRange_t      stLinAeRange;
    CalibDb_HdrAeRange_t      stHdrAeRange;
    CalibDb_AeFrmRateAttr_t   stFrmRate;
} CalibDb_AeAttr_t;
typedef CalibDb_AeAttr_t Uapi_AeAttr_t;
```

【成员】

通过api设置AE参数范围时，需要将SetAeRangeEn置1，否则默认使用系统自动曝光参数范围。曝光参数范围的设置，按照曝光模式的不同，分为stLinAeRange和stHdrAeRange两套。其中stHdrAeRange内支持各帧自动曝光参数范围的设置。

| 成员名称 | 描述 |
|-----------------|--|
| stAeSpeed | 自动曝光调节速度 |
| BlackDelayFrame | 自动曝光延时属性，图像亮度低于目标值超过BlackDelayFrame帧时，Ae开始调节 |
| WhiteDelayFrame | 自动曝光延时属性，图像亮度高于目标值超过WhiteDelayFrame帧时，Ae开始调节 |
| SetAeRangeEn | 是否设置自动曝光参数范围 |
| stLinAeRange | 线性模式自动曝光量范围结构体 |
| stHdrAeRange | Hdr模式自动曝光量范围结构体 |
| stFrmRate | 自动曝光帧率模式结构体，固定帧率模式或自动降帧模式 |

【注意事项】

- 通过api设置AE参数范围时，需要将SetAeRangeEn置1，否则默认使用系统自动曝光参数范围。曝光参数范围的设置，按照曝光模式的不同，分为stLinAeRange和stHdrAeRange两套。其中stHdrAeRange内支持各帧自动曝光参数范围的设置。
- 通过api查询到的AE参数范围，为算法内部经过校验校正后实际使用的参数范围。当api设置的AE参数范围超过sensor限制的参数范围时，会使用sensor限制的参数范围。sensor限制的参数范围，详见Rockchip_Tuning_Guide_ISP2x_CN_v1.x.x.pdf 文档中的AE模块sensorinfo参数。

CalibDb_AeSpeed_t

【说明】

定义AE条件速度属性

【定义】

```
typedef struct calibdb_aespeed_s {
    float DampOver;
    float DampUnder;
    float DampDark2Bright;
    float DampBright2Dark;
} CalibDb_AeSpeed_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------|--------------------------------------|
| DampOver | 环境亮度稳定，图像亮度高于目标值时对应的曝光调节速度，取值范围[0,1] |
| DampUnder | 环境亮度稳定，图像亮度低于目标值时对应的曝光调节速度，取值范围[0,1] |
| DampDark2Bright | 环境亮度突变，从暗到亮时对应的曝光调节速度，取值范围[0,1] |
| DampBright2Dark | 环境亮度突变，从亮到暗时对应的曝光调节速度，取值范围[0,1] |

CalibDb_AeRange_t

【说明】

定义AE参数范围

【定义】

```
typedef struct CalibDb_AeRange_s {
    float Min;
    float Max;
} CalibDb_AeRange_t;
```

【成员】

| 成员名称 | 描述 |
|------|-----|
| Min | 下限值 |
| Max | 上限值 |

CalibDb_LinAeRange_t

【说明】

定义AE线性模式的参数范围

【定义】

```
typedef struct CalibDb_LinAeRange_s {
    CalibDb_AeRange_t      stExpTimeRange;
    CalibDb_AeRange_t      stGainRange;
    CalibDb_AeRange_t      stIspDGainRange;
    CalibDb_AeRange_t      stPIrisRange;
} CalibDb_LinAeRange_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------|--------------------------|
| stExpTimeRange | 曝光时间范围，设置最大值和最小值，以毫秒为单位 |
| stGainRange | Sensor 模拟增益范围，设置最大值和最小值 |
| stIspDGainRange | ISP数字增益范围，设置最大值和最小值 |
| stPIrisRange | 光圈等效增益范围，仅支持P-Iris光圈大小控制 |

【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AecRoute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AecRoute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AecRoute曝光分解路线节点最大值和最小值为准。

CalibDb_HdrAeRange_t

【说明】

定义AE HDR模式的参数范围

【定义】

```
typedef struct CalibDb_HdrAeRange_s {
    CalibDb_AeRange_t      stExpTimeRange[3];
    CalibDb_AeRange_t      stGainRange[3];
    CalibDb_AeRange_t      stIspDGainRange[3];
    CalibDb_AeRange_t      stPIrisRange;
} CalibDb_HdrAeRange_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------|---|
| stExpTimeRange | 曝光时间范围，设置最大值和最小值，以秒为单位，数组0/1/2分别为短帧、中帧、长帧。 |
| stGainRange | Sensor 模拟增益范围，设置最大值和最小值，数组0/1/2分别为短帧、中帧、长帧。 |
| stIspDGainRange | ISP数字增益范围，设置最大值和最小值，数组0/1/2分别为短帧、中帧、长帧。 |
| stPlrisRange | 光圈等效增益值范围，设置最大值和最小值 |

【注意事项】

- stExpTimeRange、stGainRange、stIspDGainRange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，0-2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。
- 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/最小值以曝光分解路线节点最大值和最小值为准。

CalibDb_AeFrmRateAttr_t

【说明】

定义AE 帧率属性

【定义】

```
typedef struct CalibDb_AeFrmRateAttr_s {
    bool           isFpsFix;
    uint8_t        FpsValue;
} CalibDb_AeFrmRateAttr_t;
```

【成员】

| 成员名称 | 描述 |
|----------|--|
| isFpsFix | 自动曝光固定帧率模式的使能，默认值为FALSE, 即采用自动降帧模式；值为TRUE时，表示固定帧率模式。 |
| FpsValue | 仅在固定帧率时有效，默认值为0时，固定使用驱动默认的帧率；值不会0时，使用设定的帧率值。 |

Uapi_MeAttr_t

【说明】

手动曝光参数设置，根据曝光模式分为LinearAE和HdrAE两套参数。

【定义】

```

typedef struct CalibDb_MeAttr_s {
    CalibDb_LinMeAttr_t      stLinMe;
    CalibDb_HdrMeAttr_t     stHdrMe;
} CalibDb_MeAttr_t;
typedef CalibDb_MeAttr_t Uapi_MeAttr_t;

```

【相关数据类型】

- CalibDb_LinMeAttr_t
- CalibDb_HdrMeAttr_t

CalibDb_LinMeAttr_t

【说明】

LinearAE的手动曝光控制参数

【定义】

```

typedef struct CalibDb_LinMeAttr_s {
    bool                  ManualTimeEn;
    bool                  ManualGainEn;
    bool                  ManualIspDgainEn;
    bool                  ManualIrisEn;
    float                TimeValue;
    float                GainValue;
    float                IspDGainValue;
    int                  PIRisGainValue;
    int                  DCIrisValue;
} CalibDb_LinMeAttr_t;

```

【成员】

| 成员名称 | 描述 |
|------------------|---|
| ManualTimeEn | 手动曝光时间使能，默认值为1 |
| ManualGainEn | 手动sensor增益使能，默认值为1 |
| ManuallspDgainEn | 手动ISP数字增益使能，默认值为1 |
| ManuallirisEn | 手动光圈使能，默认值为1 |
| TimeValue | 手动曝光时间值，以s为单位，参数值受sensor限制 |
| GainValue | 手动sensor增益值，参数值受sensor限制 |
| IspDGainValue | 手动ISP数字增益值，参数值受ISP限制 |
| PIrisGainValue | 手动P光圈等效增益值，参数值受P光圈设备限制，取值范围为[1, 1024] |
| DCIrisValue | DC光圈HoldValue值，参数值与DC光圈设备有关，取值范围为[0, 100] |

【注意事项】

- 该模块仅在AeOptype = MANUAL时有效。ManualTimeEn,ManualGainEn, ManuallspDgainEn,ManuallirisEn 皆为1时，为手动模式；以上四者中只要任意一项不使能，则为半自动模式；以上四者皆为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自动模式下的最大/最小曝光时间和增益限制。超出自动曝光限制的范围之后，将使用自动模式下最大/最小值替代。
- ManualIrisEn，手动光圈使能。当光圈类型为P光圈时，仅PirisGainValue有效；当光圈类型为DC光圈时，仅DCIrisValue有效。
- DCIrisValue，手动模式下直接设置电机的PWM占空比值，取值范围[0, 100]。手动模式下若设置为HoldValue值（即AeIrisCtrl中ClosePwmDuty到OpenPwmDuty区间内的值），则DC光圈孔径维持在当前大小；若设置的值大于OpenPwmDuty，则光圈处于打开状态，该值越大打开的速度越大；若设置的值小于ClosePwmDuty，则光圈处于关闭状态，该值越小关闭的速度越大。
- RV1109/RV1126目前不支持ISP数字增益，故ManualIspDgainEn、IspDGainValue皆无效。

CalibDb_HdrMeAttr_t

【说明】

HdrAE的手动曝光控制参数

【定义】

```
typedef struct CalibDb_HdrMeAttr_s {
    bool           ManualTimeEn;
    bool           ManualGainEn;
    bool           ManualIspDgainEn;
    bool           ManualIrisEn;
    Cam3x1FloatMatrix_t TimeValue;
    Cam3x1FloatMatrix_t GainValue;
    Cam3x1FloatMatrix_t IspDGainValue;
    int             PIRisGainValue;
    int             DCIrisValue;
} CalibDb_HdrMeAttr_t;
```

【成员】

变量含义与CalibDb_LinMeAttr_t相同

【注意事项】

- 该模块仅在AeOptype = MANUAL时有效。ManualTimeEn,ManualGainEn, ManualIspDgainEn,ManualIrisEn 皆为1时，为手动模式；以上四者中只要任意一项不使能，则为半自动模式；以上四者皆为0，则等同自动模式，系统会报错提醒。
- TimeValue/GainValue/IspDGainValue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。
- 手动/半手动模式下，手动曝光时间和增益会受自动模式下的最大/最小曝光时间和增益限制。超出自动曝光限制的范围之后，将使用自动模式下最大/最小值替代。
- ManualIrisEn，手动光圈使能。当光圈类型为P光圈时，仅PirisGainValue有效；当光圈类型为DC光圈时，仅DCIrisValue有效。
- DCIrisValue，手动模式下直接设置电机的PWM占空比值，取值范围[0, 100]。手动模式下若设置为HoldValue值（即AeIrisCtrl中ClosePwmDuty到OpenPwmDuty区间内的值），则DC光圈孔径维持在当前大小；若设置的值大于OpenPwmDuty，则光圈处于打开状态，该值越大打开的速度越大；若设置的值小于ClosePwmDuty，则光圈处于关闭状态，该值越小关闭的速度越大。
- RV1109/RV1126目前不支持ISP数字增益，故ManualIspDgainEn、IspDGainValue皆无效。

Uapi_ExInitExp_t

【说明】

曝光初始值参数

【定义】

```
typedef struct CalibDb_ExpInitExp_s {
    CalibDb_LinExpInitExp_t          stLinExpInitExp;
    CalibDb_HdrExpInitExp_t         stHdrExpInitExp;
} CalibDb_ExpInitExp_t;
typedef CalibDb_ExpInitExp_t uapi_ExpInitExp_t;
```

【相关数据类型】

- CalibDb_LinExpInitExp_t
- CalibDb_HdrExpInitExp_t

CalibDb_LinExpInitExp_t

【说明】

线性曝光初始曝光参数

【定义】

```
typedef struct CalibDb_LinExpInitExp_s {
    float           InitTimeValue;
    float           InitGainValue;
    float           InitIspDGainValue;
    int             InitPIrisGainValue;
    int             InitDCIrisDutyValue;
} CalibDb_LinExpInitExp_t;
```

【成员】

| 成员名称 | 描述 |
|---------------------|---------------|
| InitTimeValue | 初始sensor曝光时间值 |
| InitGainValue | 初始sensor曝光增益值 |
| InitIspDGainValue | 初始ISP数字增益值 |
| InitPIrisGainValue | 初始P光圈等效增益值 |
| InitDCIrisDutyValue | 初始DC光圈占空比值 |

CalibDb_HdrExpInitExp_t

【说明】

Hdr曝光初始曝光参数

【定义】

```

typedef struct CalibDb_HdrExpInitExp_s {
    Cam3x1FloatMatrix_t    InitTimeValue;
    Cam3x1FloatMatrix_t    InitGainValue;
    Cam3x1FloatMatrix_t    InitIspDGainValue;
    int                   InitPIrisGainvalue;
    int                   InitDCIRisDutyValue;
    int                   array_size;
} CalibDb_HdrExpInitExp_t;

```

【成员】

成员描述同CalibDb_LinExplnItExp_t一致

【注意事项】

- InitTimeValue/InitGainValue/InitIspDGainValue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。

Uapi_LinAeRouteAttr_t

【说明】

定义AE线性策略属性

【定义】

```

#define AEC_ROUTE_MAX_NODES (10)
typedef struct CalibDb_LinAeRoute_Attr_s {
    AecExpSeparateName_t      name;
    float                    TimeDot[AEC_ROUTE_MAX_NODES];
    float                    GainDot[AEC_ROUTE_MAX_NODES];
    float                    IspgainDot[AEC_ROUTE_MAX_NODES];
    int                      PIRisGainDot[AEC_ROUTE_MAX_NODES];
    int                      array_size;
} CalibDb_LinAeRoute_Attr_t;
typedef CalibDb_LinAeRoute_Attr_t Uapi_LinAeRouteAttr_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|----------------------|
| name | 模式名称，分为day模式和night模式 |
| TimeDot | sensor曝光时间节点，单位为s |
| GainDot | sensor曝光增益节点 |
| IspgainDot | Isp数字增益节点 |
| PIrisGainDot | 光圈等效增益节点 |
| array_size | 曝光分解节点个数 |

【注意事项】

- 曝光分解曲线节点个数至多10个，建议至少设置6个节点

- 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。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AeIrisCtrl模块。
- 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。
- RV1109/RV1126目前不支持ISP数字增益，故IspGainDot无效。

【相关数据类型】

- AEC_ROUTE_MAX_NODES
- AecExpSeparateName_t

AecExpSeparateName_t

【说明】

定义Name字符串类型

【定义】

```
#define AEC_EXP_SEPARATE_NAME          ( 20U )
typedef char AecExpSeparateName_t[AEC_EXP_SEPARATE_NAME];
```

Uapi_HdrAeRouteAttr_t

【说明】

定义AE HDR策略属性

【定义】

```
typedef struct CalibDb_HdrAeRoute_Attr_s {
    AecExpSeparateName_t      name;
    float                    HdrTimeDot[3][AEC_ROUTE_MAX_NODES];
    float                    HdrGainDot[3][AEC_ROUTE_MAX_NODES];
    float                    HdrIspDGainDot[3][AEC_ROUTE_MAX_NODES];
    int                      PirisGainDot[AEC_ROUTE_MAX_NODES];
    int                      array_size;
} CalibDb_HdrAeRoute_Attr_t;
typedef CalibDb_HdrAeRoute_Attr_t Uapi_HdrAeRouteAttr_t;
```

【成员】

| 成员名称 | 描述 |
|----------------|--------------------------------------|
| name | 模式名称，分为day模式和night模式 |
| HdrTimeDot | sensor曝光时间节点，单位为s，数组0/1/2分别为短帧、中帧、长帧 |
| HdrGainDot | sensor曝光增益节点，数组0/1/2分别为短帧、中帧、长帧 |
| HdrIspDGainDot | Isp数字增益节点，数组0/1/2分别为短帧、中帧、长帧 |
| P IrisDot | 光圈节点，数组0/1/2分别为短帧、中帧、长帧 |
| array_size | 曝光分解节点个数，数组0/1/2分别为短帧、中帧、长帧 |

【注意事项】

- 曝光分解曲线节点个数至多10个，建议至少设置6个节点
- HdrTimeDot、HdrGainDot、HdrIspDGainDot定义为二维数组，第一维表示帧数，第二维表示各帧对应的曝光分量节点。2帧HDR模式下，第一维中仅0、1有效，分别表示短、长帧；3帧HDR模式下，第一维中0-2皆有效，分别表示短、中、长帧。
- 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。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AeIrisCtrl模块。
- 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。
- RV1109/RV1126目前不支持ISP数字增益，故HdrIspDGainDot无效。

【相关定义】

- AEC_ROUTE_MAX_NODES
- AecExpSeparateName_t

Uapi_LinExpAttr_t

【说明】

定义AE线性曝光调试参数

【定义】

```
typedef struct calibdb_LinearAE_Attr_s {
    uint8_t                    RawStatsEn;
    float                      SetPoint;
    float                      NightSetPoint;
```

```

    float          ToleranceIn;
    float          ToleranceOut;
    float          EbBias;
    CalibDb_AeStrategyMode_t      StrategyMode;
    bool           DySetPointEn;
    CalibDb_AecDynamicSetpoint_t DySetpoint[AEC_DNMODE_MAX];

    CalibDb_AecBacklight_t  BackLightConf;
    CalibDb_AecOverExpCtrl_t overExpCtrl;
} CalibDb_LinearAE_Attr_t;

typedef CalibDb_LinearAE_Attr_t Uapi_LinExpAttr_t;

```

【成员】

| 成员名称 | 描述 |
|-----------------|--|
| RawStatsEn | 线性曝光使用Raw域统计亮度功能使能 |
| SetPoint | day模式下，自动曝光调节时的目标亮度值，取值范围[0,255] |
| NightSetPoint | night模式下，自动曝光调节时的目标亮度值，取值范围[0,255] |
| EvBias | 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] |
| ToleranceIn/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] |
| StrategyMode | 自动曝光策略模式，高光优先或低光优先 |
| DySetPointEn | 自动曝光调节的动态目标亮度值使能开关。动态目标亮度值的使能，enable=TRUE时，采用动态目标亮度值，enable=FALSE时，使用固定目标亮度值，动态目标亮度值失效 |
| DySetpoint | 自动曝光调节的动态目标亮度值属性，随曝光量动态变化，分为normal模式和夜间/IR模式 |
| BackLightConf | 背光补偿功能 |
| OverExpCtrl | 强光抑制模块 |

【注意事项】

- SetPoint 代表normal模式下的目标亮度值，即未开启夜间模式或IR模式时使用的亮度值。NightSetPoint 代表夜间模式或IR模式下的目标亮度值。夜间模式与IR模式不可同时开启，IR模式的开启需要硬件上的支持。
- DySetPointEn = TRUE时，固定目标亮度值SetPoint、NightSetPoint无效，使用动态目标亮度值；DySetPointEn = FALSE时，动态目标亮度值无效，所有场景始终使用同一目标亮度。
- 曝光量偏差EvBias，用于特殊场景下对（固定/动态）目标亮度值（SetPoint/IRSetPoint）进行微调。真实生效目标亮度为 $(SetPoint/IRSetPoint) * (1+EvBias/100)$
- 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B应在 [真实生效目标亮度 X (1-Tolerance/100) , 真实生效目标亮度 X (1+Tolerance/100)] 范围内。
- StrategyMode暂无效。

【相关数据类型】

- CalibDb_AecDynamicSetpoint_t
- CalibDb_AecBacklight_t
- CalibDb_AecOverExpCtrl_t

CalibDb_AecDynamicSetpoint_t

【说明】

定义AE动态目标值

【定义】

```
#define AEC_SETPOINT_MAX_NODES 10

typedef struct CalibDb_AecDynamicSetpoint_s {
    AecDynamicSetpointName_t      name;
    float ExpValue[AEC_SETPOINT_MAX_NODES];
    float DySetpoint[AEC_SETPOINT_MAX_NODES];
    int   array_size;
} CalibDb_AecDynamicSetpoint_t;
```

【成员】

| 成员名称 | 描述 |
|------------|--|
| name | 模式名称，分为day模式和night模式 |
| ExpValue | 动态曝光量节点属性，节点值为当前曝光量与最大曝光量的比值，按照递增顺序设置，取值范围为[0,1] |
| DySetpoint | 动态目标亮度值节点属性，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 |
| array_size | 动态目标亮度值的节点数 |

Uapi_HdrExpAttr_t

【说明】

定义AE HDR曝光调试参数

【定义】

```
typedef struct CalibDb_HdrAE_Attr_s {
    float                      ToleranceIn;
    float                      ToleranceOut;
    CalibDb_AeHdrLongFrmMode_t LongfrmMode;
    CalibDb_AeStrategyMode_t   StrategyMode;
    float                      Ebias;
    CalibDb_HdraERatioType_t  ExpRatioType;
    Cam1x6FloatMatrix_t       RatioExpDot;
    Cam1x6FloatMatrix_t       M2SRatioFix;
    Cam1x6FloatMatrix_t       L2MRatioFix;
    Cam1x6FloatMatrix_t       M2SRatioMax;
    Cam1x6FloatMatrix_t       L2MRatioMax;
    CalibDb_LFrameCtrl_t     LframeCtrl;
    CalibDb_MFrameCtrl_t     MframeCtrl;
    CalibDb_SFrameCtrl_t     SframeCtrl;
} CalibDb_HdrAE_Attr_t;
typedef CalibDb_HdrAE_Attr_t Uapi_HdrExpAttr_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------|---|
| ToleranceIn/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] |
| LongfrmMode | 长帧模式参数 |
| StrategyMode | 自动曝光策略模式，高光优先或低光优先 |
| Evbias | 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] |
| ExpRatioType | 曝光比模式，仅在Hdr模式多帧合成下有效，AUTO：根据场景，自动计算长短帧的曝光比；FIX：长短帧采用固定曝光比 |
| RatioExpDot | 表示曝光量节点，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。节点个数固定为6 |
| M2SRatioFix | 节点个数固定为6。ExpRatioType为AUTO时，无效。ExpRatioType为FIX时，表示中帧与短帧的曝光比，与曝光量节点RatioExpDot一一对应 |
| L2MRatioFix | 节点个数固定为6。ExpRatioType为AUTO时，无效。ExpRatioType为FIX时，表示长帧与中帧的曝光比，与曝光量节点RatioExpDot一一对应。Hdr为2帧合成时无效，3帧合成时有效 |
| M2SRatioMax | 节点个数固定为6。ExpRatioType为AUTO时，表示中帧与短帧的曝光比动态最大值，与曝光量节点RatioExpDot一一对应。ExpRatioType为FIX时，无效 |
| L2MRatioMax | 节点个数固定为6。ExpRatioType为AUTO时，表示长帧与中帧的曝光比动态最大值，与曝光量节点RatioExpDot一一对应。Hdr为2帧合成时无效，3帧合成时有效。ExpRatioType为FIX时，无效 |
| LframeCtrl | 长帧控制参数 |
| MframeCtrl | 中帧控制参数，仅在HDR 3帧模式下有效 |
| SframeCtrl | 短帧控制参数 |

【注意事项】

- ExpRatioType为AUTO，采用自动曝光比模式。2帧模式下，长短帧的最大曝光比受M2SratioMax限制；3帧模式下，中短帧的最大曝光比受M2SratioMax限制，长中帧的最大曝光比受L2MratioMax限制。最小曝光比无限制，不得低于1。ExpRatioType为FIX，采用固定曝光比模式。2帧模式下，长短帧的曝光比为M2SRatioFix；3帧模式下，中短帧的曝光比为M2SRatioFix，长中帧的曝光比为L2MratioFix。

【相关数据类型】

- CalibDb_LFrameCtrl_t
- CalibDb_MFrameCtrl_t
- CalibDb_SFrameCtrl_t

Uapi_ExpQueryInfo_t

【说明】

定义AE曝光参数查询

【定义】

```

typedef struct Uapi_ExpQueryInfo_s {

    bool           IsConverged;
    bool           IsExpMax;
    float          LumaDeviation;
    float          HdrLumaDeviation[3];

    float          MeanLuma;
    float          HdrMeanLuma[3];

    float          GlobalEnvLux;
    float          BlockEnvLux[ISP2_RAWAE_WINNUM_MAX];

    RKAIqAecExpInfo_t CurExpInfo;
    unsigned short  Piris;
    float          LinePeriodsPerField;
    float          PixelPeriodsPerLine;
    float          PixelClockFreqMHZ;

} Uapi_ExpQueryInfo_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 |
| LumaDeviation | 线性模式下，AEC的目标值与实际画面亮度的差值，该值为正，表示实际亮度大于目标亮度；该值为负，表示实际亮度小于目标亮度；该值为0，表示实际画面亮度在目标值的容忍度范围内。 |
| HdrLumaDeviation | Hdr模式下，各帧的亮度目标值与实际画面亮度的差值，该值为正，表示实际亮度大于目标亮度；该值为负，表示实际亮度小于目标亮度；该值为0，表示实际画面亮度在目标值的容忍度范围内。2帧模式下，仅0、1有效，分别表示短、长帧；3帧模式下，0-2皆有效，分别表示短、中、长帧。 |
| MeanLuma | 线性模式下，当前画面的平均亮度。 |
| HdrMeanLuma | HDR模式下，各帧当前画面的平均亮度。2帧模式下，数组仅0、1成员有效，分别表示短、长帧；3帧模式下，数组0-2成员皆有效，分别表示短、中、长帧。 |
| CurExpInfo | 当前曝光参数，包括sensor曝光时间、sensor曝光增益值。按照曝光模式，分为LinearExp和HdrExp[3] 2套曝光参数。 |
| Piris | 光圈 |
| LinePeriodsPerField | sensor的VTS |
| PixelPeriodsPerLine | sensor的HTS |
| PixelClockFreqMHZ | sensor的像素时钟频率(单位：兆赫兹) |

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和System模块参数。这两个模块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AecSyncTest功能进行自测。sensorinfo、System参数含义说明参考《Rockchip_Tuning_Guide_ISP2x》。

AecSyncTest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。

AecSyncTest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 |
|-------------|----------------|
| Enable | 曝光与统计同步测试功能的使能 |
| IntervalFrm | 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LinearAE和HdrAE两套参数。

| 成员名称 | 描述 |
|----------------|-------------|
| TimeValue | 曝光时间值 |
| GainValue | 曝光增益值 |
| IspDgainValue | Isp数字增益值 |
| DcgMode | Dcg模式值 |
| PirisGainValue | 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和System参数设置正确。

```

>>> Framenum=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) EXP_DELAY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=50.342224,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。

(2) 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

rk_aiq_uapi_setWBMode

【描述】

设置白平衡工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_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场景下的增益值。若需要切换手动模式的同时设置增益值，可以使用rk_aiq_uapi_setMWBGain接口。

【需求】

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

rk_aiq_uapi_getWBMode

【描述】

获取白平衡工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_lockAWB

【描述】

锁定当前白平衡参数。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_unlockAWB

【描述】

解锁已被锁定的白平衡参数。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMWBScene

【描述】

设置白平衡场景。

【语法】

```
XCamReturn rk_aiq_uapi_setMWBScene(const rk_aiq_sys_ctx_t* ctx,  
rk_aiq_wb_scene_t scene);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| scene | 白平衡场景 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMWBScene

【描述】

获取白平衡场景。

【语法】

```
XCamReturn rk_aiq_uapi_getMWBScene(const rk_aiq_sys_ctx_t* ctx,  
rk_aiq_wb_scene_t *scene);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| scene | 白平衡场景 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMWBGain

【描述】

设置白平衡增益系数。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMWBGain

【描述】

获取白平衡增益系数。

【语法】

```
XCamReturn rk_aiq_uapi_getMWBGain(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMWBCT

【描述】

设置白平衡色温参数。

【语法】

```
XCamReturn rk_aiq_uapi_setMWBCT(const rk_aiq_sys_ctx_t* ctx, unsigned int ct);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| ct | 白平衡色温参数 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMWBCT

【描述】

获取白平衡增益系数。

【语法】

```
XCamReturn rk_aiq_uapi_getMWBCT(const rk_aiq_sys_ctx_t* ctx, unsigned int *ct);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| ct | 白平衡色温 | 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_api_imgproc.h
- 库文件: librkaiq.so

功能级API数据类型

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 | 手动模式 |
| RK_AIQ_WB_MODE_AUTO | 自动模式 |

rk_aiq_wb_scene_t

参见前述。

rk_aiq_wb_gain_t

参见前述。

rk_aiq_wb_cct_t

参见前述。

模块级API参考

rk_aiq_user_api_awb_SetAttrib

【描述】

获取白平衡属性。

【语法】

```
XCamReturn
rk_aiq_user_api_awb_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_wb_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_api_awb.h、rk_aiq_uapi_awb_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_awb_GetAttrib

【描述】

获取白平衡属性。

【语法】

```
XCamReturn  
rk_aiq_user_api_awb_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
rk_aiq_wb_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_api_awb.h、rk_aiq_uapi_awb_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_awb_GetCCT

【描述】

获取白平衡色温参数。

【语法】

```
XCamReturn  
rk_aiq_user_api_awb_GetCCT(const rk_aiq_sys_ctx_t* sys_ctx, rk_aiq_wb_cct_t  
*cct);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| cct | 白平衡的色温参数 | 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_api_awb.h、rk_aiq_uapi_awb_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_awb_QueryWBInfo

【描述】

获取白平衡增益系数，检测色温。

【语法】

```
XCamReturn
rk_aiq_user_api_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_api_awb.h、rk_aiq_uapi_awb_int.h
- 库文件: librkaiq.so

模块级API数据类型

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_wb_awb_attrib_t

【说明】

定义自动白平衡属性

【定义】

```
typedef struct rk_aiq_wb_awb_attrib_s {
    rk_aiq_wb_awb_alg_method_t algMethod;
    float tolerance;
    unsigned int runInterval;
    bool sceneAdjustEn;
    bool colorBalanceEn;
    bool cagaEn;
    bool wbGainAdjustEn;
    bool wbGainDaylightClipEn;
    bool wbGainInclipEn;
} rk_aiq_wb_awb_attrib_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|-----------------------------|
| algMethod | 白平衡策略选择 |
| tolerance | 容忍度 |
| runInterval | 运行帧间隔 |
| sceneAdjustEn | 场景调节 |
| colorBalanceEn | 色调调整使能 |
| cagaEn | 白平衡校正后的图像尽可能与人眼感知的色彩外貌一致的使能 |
| wbGainAdjustEn | 色调调整使能 |
| wbGainDaylightClipEn | 室外最低色温限制使能 |
| wbGainClipEn | 色温范围限制使能 |

rk_aiq_wb_attrib_t

【说明】

定义白平衡属性

【定义】

```
typedef struct rk_aiq_wb_attrib_s {
    bool byPass;
    rk_aiq_wb_op_mode_t mode;
    rk_aiq_wb_mwb_attrib_t stManual;
    rk_aiq_wb_awb_attrib_t stAuto;
} rk_aiq_wb_attrib_t;
```

【成员】

| 成员名称 | 描述 |
|----------|-----------|
| byPass | 跳过模块处理 |
| mode | 模式选择 |
| stManual | 手动模式下参数配置 |
| stAuto | 自动模式下参数配置 |

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;
    rk_aiq_wb_cct_t cctGloabl;
    bool awbConverged;
} rk_aiq_wb_querry_info_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|---------|
| gain | 增益 |
| cctGloabl | 全局色温参数 |
| awbConverged | 白平衡是否收敛 |

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。

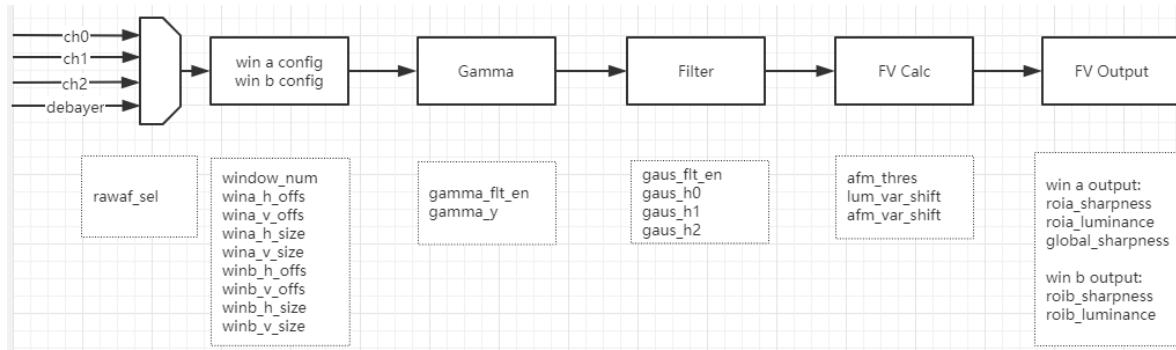
重要概念

- VCM: 音圈马达或音圈电机

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，ch0/ch1/ch2对应hdr模式下S/M/L帧，debayer为hdr模式下的合成帧。

FV Calc 使用固定算子计算图像水平方向和垂直方向的边缘信息。

windowA的输出主要包含15*15的FV信息，亮度信息未提供，需要将AE窗口设置成与AF一致，然后直接借用AE 15x15的亮度统计信息。

windowsB的输出主要包含一个FV信息和亮度信息。

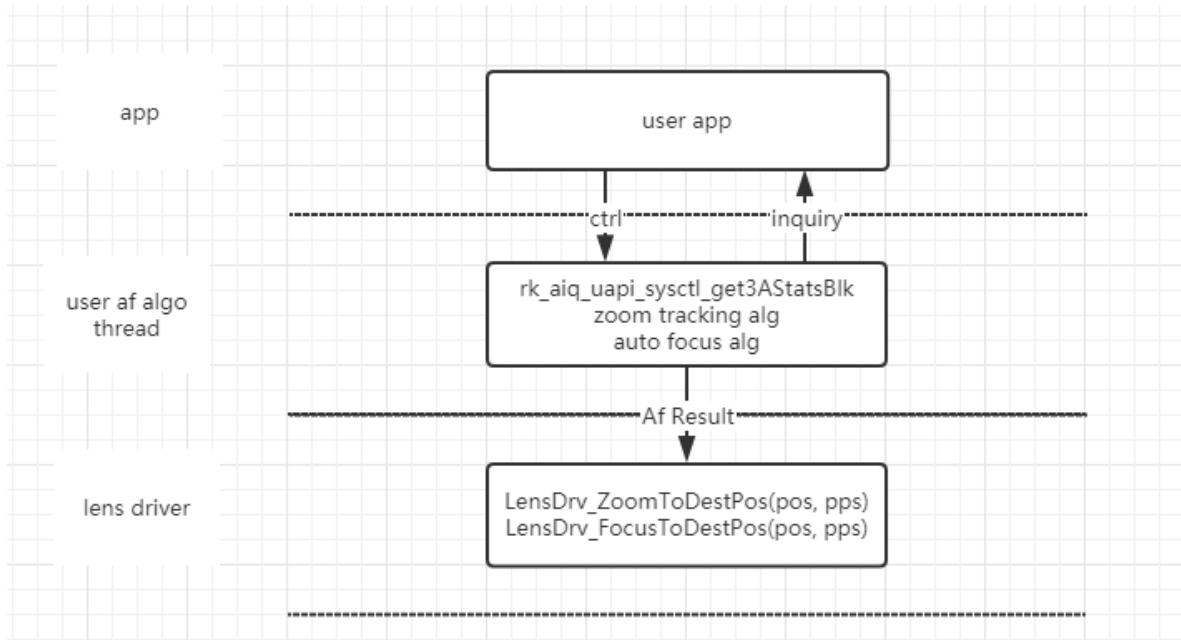
开发用户AF算法

用户实现AF算法时，可参考上图设置rk_aiq_af_attrib_t中的rk_aiq_af_algo_meas_t结构体，然后调用rk_aiq_user_api_af_SetAttrib进行af信息统计模块的配置。

之后可单独启动一个线程调用rk_aiq_uapi_sysctl_get3AStatsBlk获取af/ae/awb统计信息，af统计信息对应结构体为rk_aiq_af_algo_stat_t。

详细信息可参考rk_aiq_uapi_sysctl_get3AStatsBlk函数说明。

用户AF算法参考统计信息完成目标 zoom 和 focus 位置的计算，调用Lens Driver 将镜头镜片驱动到指定的位置即完成当前帧的操作。



参考代码

```
// Set AF meas config
rk_aiq_user_api_af_SetAttrib(ctx, &attr);

while (1)
{
    rk_aiq_isp_stats_t *stats_ref = NULL;

    // Get 3A stats, return 3a stats on each frame
    ret = rk_aiq_uapi_sysctl_get3AStatsBlk(ctx->aiq_ctx , &stats_ref, -1);
    if (ret == XCAM_RETURN_NO_ERROR && stats_ref != NULL) {
        {
            // User Auto Focus Alg Source
            UsrAfAlgRun();
            // Update Lens Position
            LensDrv_ZoomToDestPos(pos, pps);
            LensDrv_FocusToDestPos(pos, pps);
            // Release 3A stats
            rk_aiq_uapi_sysctl_release3AStatsRef(ctx->aiq_ctx, stats_ref);
        }
    }
}
```

功能级API参考

rk_aiq_uapi_setFocusMeasCfg

【描述】

配置AF信息统计，一般自行实现af算法时调用。

【语法】

```
XCamReturn rk_aiq_uapi_setFocusMeasCfg(const rk_aiq_sys_ctx_t* ctx,  
rk_aiq_af_algo_meas_t* meascfg);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| meascfg | AF信息统计配置 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFocusMode

【描述】 配置对焦模式。

【语法】

```
XCamReturn rk_aiq_uapi_setFocusMode(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 | 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_api_imgproc.h

- 库文件: librkaiq.so

rk_aiq_uapi_getFocusMode

【描述】 获取当前对焦模式。

【语法】

```
XCamReturn rk_aiq_uapi_getFocusMode(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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFixedModeCode

【描述】 设置手动对焦模式下的对焦位置。

【语法】

```
XCamReturn rk_aiq_uapi_setFixedModeCode(const rk_aiq_sys_ctx_t* ctx, unsigned short code);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|-------------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| code | 对焦code值, 取值范围0-64 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getFixedModeCode

【描述】 获取当前对焦位置。

【语法】

```
XCamReturn rk_aiq_uapi_getFixedModeCode(const rk_aiq_sys_ctx_t* ctx, unsigned short *code);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFocusWin

【描述】 设置自动对焦窗口。

【语法】

```
XCamReturn rk_aiq_uapi_SetFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_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 | 对焦窗口 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getFocusWin

【描述】 设置自动对焦窗口。

【语法】

```
XCamReturn rk_aiq_uapi_getFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_lockFocus

【描述】 锁定自动对焦，对焦暂停动作。

【语法】

```
XCamReturn rk_aiq_uapi_lockFocus(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_unlockFocus

【描述】 解锁自动对焦，对焦继续动作。

【语法】

```
XCamReturn rk_aiq_uapi_unlockFocus(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_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_uapi_oneshotFocus(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_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_uapi_manualTrigerFocus(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_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_uapi_oneshotFocus后使用。

【语法】

```
XCamReturn rk_aiq_uapi_trackingFocus(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setVcmCfg

【描述】 对vcm驱动进行配置。

【语法】

```
XCamReturn rk_aiq_uapi_setVcmCfg(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_lens_vcmcfg* cfg);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| cfg | vcm配置 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getVcmCfg

【描述】 获取当前vcm驱动配置。

【语法】

```
XCamReturn rk_aiq_uapi_getVcmCfg(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_lens_vcmcfg* cfg);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|----------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| cfg | vcm配置 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setOpZoomPosition

【描述】 设置zoom设备的位置。

【语法】

```
XCamReturn rk_aiq_uapi_setOpZoomPosition(const rk_aiq_sys_ctx_t* ctx, int pos);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|---------------------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| pos | zoom position, 取值范围0-2000 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getOpZoomPosition

【描述】 设置zoom设备的位置。

【语法】

```
XCamReturn rk_aiq_uapi_getOpZoomPosition(const rk_aiq_sys_ctx_t* ctx, int *pos);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|---------------------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| pos | zoom position, 取值范围0-2000 | 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_api_imgproc.h
- 库文件: librkaiq.so

功能级API数据类型

rk_aiq_af_algo_meas_t

【说明】

定义AF信息统计工作模式

【定义】

```
typedef struct {
    unsigned char contrast_af_en;
    unsigned char rawaf_sel;

    unsigned char window_num;
    unsigned short wina_h_offs;
    unsigned short wina_v_offs;
    unsigned short wina_h_size;
    unsigned short wina_v_size;

    unsigned short winb_h_offs;
    unsigned short winb_v_offs;
    unsigned short winb_h_size;
    unsigned short winb_v_size;

    unsigned char gamma_flt_en;
    unsigned short gamma_y[RKAIQ_RAWAF_GAMMA_NUM];

    unsigned char gaus_flt_en;
    unsigned char gaus_h0;
    unsigned char gaus_h1;
    unsigned char gaus_h2;

    unsigned char line_en[RKAIQ_RAWAF_LINE_NUM];
    unsigned char line_num[RKAIQ_RAWAF_LINE_NUM];

    unsigned short afm_thres;
```

```
unsigned char lum_var_shift[RKAIQ_RAWAF_WIN_NUM];  
unsigned char afm_var_shift[RKAIQ_RAWAF_WIN_NUM];  
} rk_aiq_af_algo_meas_t;
```

【成员】

| 成员名称 | 描述 |
|----------------|--|
| contrast_af_en | 是否使能AF信息统计，0为关闭，1为打开 |
| rawaf_sel | 选择AF信息统计的通道，取值范围0-3，对应hdr模式的长/中/短/合成帧通道选择，一般AF选择中帧通道，非hdr模式设置为0，hdr模式设置为1 |
| window_num | 生效的窗口数，window_num为1时，wina(主窗口)生效；window_num为2时，wina(主窗口)和winb(独立窗口)生效 |
| wina_h_offs | wina(主窗口)左上角第一个像素的水平坐标，该值必须大于等于2 |
| wina_v_offs | wina(主窗口)左上角第一个像素的垂直坐标，该值必须大于等于1 |
| wina_h_size | wina(主窗口)的窗口宽度，该值必须小于图像宽度-2-wina_h_offs；同时该值必须为15的倍数； |
| wina_v_size | wina(主窗口)的窗口高度，该值必须小于图像高度-2-wina_v_offs；同时该值必须为15的倍数； |
| winb_h_offs | winb(独立窗口)左上角第一个像素的水平坐标，该值必须大于等于2 |
| winb_v_offs | winb(独立窗口)左上角第一个像素的垂直坐标，该值必须大于等于1 |
| winb_h_size | winb(独立窗口)的窗口宽度，该值必须小于图像宽度-2-wina_h_offs |
| winb_v_size | winb(独立窗口)的窗口高度，该值必须小于图像高度-2-wina_v_offs |
| gamma_flt_en | gamma模块使能开关，0为关闭，1为打开 |
| gamma_y | gamma table的y值，取值范围0-1023；x坐标分段为0 to 1023: 16 16 16 16 32 32 32 32 64 64 64 128 128 128 128 128 |
| gaus_flt_en | 高斯滤波模块使能开关，0为关闭，1为打开 |
| gaus_h0 | 高斯滤波3x3系数h0，3x3系数最中间的系数值，取值范围为0-255 |
| gaus_h1 | 高斯滤波3x3系数h1，3x3系数最中间系数的上下左右四个系数值，取值范围为0-127 |
| gaus_h2 | 高斯滤波3x3系数h2，3x3系数中其它的系数值，取值范围为0-127 |
| line_en | 目前暂未生效 |
| line_num | 目前暂未生效 |
| afm_thres | AF统计阈值，计算出的sharpness值小于该值时，sharpness值改为0，可减少噪声的影响，取值范围为0-0xFFFF |
| lum_var_shift | luminance值的shift bit值，会按照该值将luminance值向右移位，避免得到的sharpness值溢出，取值范围为0-7 |
| afm_var_shift | sharpness值的shift bit值，会按照该值将sharpness值向右移位，避免得到的sharpness值溢出，取值范围为0-7 |

opMode_t

【说明】

定义AF信息统计工作模式

【定义】

```
typedef enum opMode_e {
    OP_AUTO = 0,
    OP_MANUAL = 1,
    OP_SEMI_AUTO = 2,
    OP_INVAL
} opMode_t;
```

【成员】

| 成员名称 | 描述 |
|--------------|--|
| OP_AUTO | 自动对焦模式，运行内置AF算法 |
| OP_MANUAL | 手动对焦模式，停止内置AF算法 |
| OP_SEMI_AUTO | 半自动对焦模式，set zoom position api调用后执行一次自动对焦 |

rk_aiq_lens_vcmcfg

【说明】 定义vcm驱动的配置

【定义】

```
typedef struct {
    int start_ma;
    int rated_ma;
    int step_mode;
} rk_aiq_lens_vcmcfg;
```

【成员】

| 成员名称 | 描述 |
|-----------|---|
| start_ma | vcm启动电流，单位为mA |
| rated_ma | vcm截止电流，单位为mA |
| step_mode | Linear slope control模式下使用的step control和step period值 |

模块级API参考

rk_aiq_user_api_af_SetAttrib

【描述】

设置对焦属性。

【语法】

```
XCamReturn
rk_aiq_user_api_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_api_awb.h、rk_aiq_uapi_awb_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_af_GetAttrib

【描述】

获取对焦属性。

【语法】

```
XCamReturn
rk_aiq_user_api_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_api_af.h、rk_aiq_uapi_af_int.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 |

rk_aiq_af_attrib_t

【说明】

对焦配置信息

【定义】

```
typedef struct rk_aiq_af_attrib_s {
    RKAIQ_AF_MODE AfMode;

    bool contrast_af;
    bool laser_af;
    bool pdaf;
    bool GammaEnable;
    bool GausEnable;

    int h_offs;
```

```

int v_offs;
unsigned int h_size;
unsigned int v_size;

unsigned short fixedModeDefCode;
unsigned short macroModeDefCode;
unsigned short infinityModeDefCode;

rk_aiq_af_algo_meas_t manual_meascfg;
} rk_aiq_af_attrib_t;

```

【成员】

| 成员名称 | 描述 |
|---------------------|----------------------------|
| 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-64 |
| manual_meascfg | 自定义对焦统计信息配置 |

其它说明

VCM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。

Normal模式提高对焦速度

sensor不通过vicap直接连到isp，且使用非HDR模式时，在应用启动前可设置export normal_no_read_back=1，启用直通模式，

这样sensor数据不用存到ddr再回读到isp，减小有效的AF统计值获取时间，提高对焦速度。

IMPROC

概述

imgproc是指影响图像效果的模块。

FEC

功能描述

光学系统、电子扫描系统失真而引起的斜视畸变、枕形、桶形畸变等，都可能使图像产生几何特性失真。图像的畸变校正是以某种变换方式将畸变图像转换为理想图像的过程。

该模块对x和y方向的图像畸变进行校正。

重要概念

- 畸变实际上指的是拍摄出来的物体相对于物体本身而言的失真。

功能级API参考

rk_aiq_uapi_setFecEn

【描述】 鱼眼畸变校正功能开关。

【语法】

```
XCamReturn rk_aiq_uapi_setFecEn(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 | 校正开关 | 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】

- 该接口只能在rk_aiq_uapi_sysctl_prepare之前调用有效，即不能在AIQ运行的状态下开关。若需要动态开关畸变校正效果，可以使用rk_aiq_uapi_setFecBypass接口。
- 开启鱼眼畸变校正后，会增加DDR带宽和CPU负载，对摄像头的采集帧率可能会影响。

【需求】

- 头文件: rk_aiq_user_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFecCorrectDirection

【描述】 设置鱼眼畸变校正方向。

【语法】

```
XCamReturn
rk_aiq_uapi_setFecCorrectDirection(const rk_aiq_sys_ctx_t* ctx, const
fec_correct_direction_t direction)
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| direction | 校正方向 | 输入 |

【定义】

```
typedef enum fec_correct_direction_e {
    FEC_CORRECT_DIRECTION_X = 0x1,
    FEC_CORRECT_DIRECTION_Y,
    FEC_CORRECT_DIRECTION_XY
} fec_correct_direction_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------------|---------|
| FEC_CORRECT_DIRECTION_X | 只校正X方向 |
| FEC_CORRECT_DIRECTION_Y | 只校正Y方向 |
| FEC_CORRECT_DIRECTION_XY | XY方向都校正 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 该接口只能在rk_aiq_uapi_sysctl_prepare之前调用有效，即不能在AIQ运行的状态下开关。

【需求】

- 头文件: rk_aiq_user_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFecBypass

【描述】 鱼眼畸变校正bypass开关，使能后，fec不进行校正。

【语法】

```
XCamReturn rk_aiq_uapi_setFecBypass(const rk_aiq_sys_ctx_t* ctx, bool bypass);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| bypass | 校正效果开关 | 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_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

rk_aiq_uapi_setFecCorrectLevel

【描述】 设置鱼眼畸变校正等级。

【语法】

```
XCamReturn rk_aiq_uapi_setFecCorrectLevel(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_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

模块级API参考

rk_aiq_user_api_afec_SetAttrib

【描述】

设置fec属性。

【语法】

```
XCamReturn
rk_aiq_user_api_afec_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_fec_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 | fec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_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_afec_GetAttrib

【描述】

获取fec属性。

【语法】

```
XCamReturn  
rk_aiq_user_api_afec_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
rk_aiq_fec_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 | fec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_api_afec.h、rk_aiq_uapi_afec_int.h
- 库文件: librkaiq.so

模块级API数据类型

`fec_correct_direction_t`

【说明】

fec校正方向

【定义】

```
typedef enum fec_correct_direction_e {  
    FEC_CORRECT_DIRECTION_X = 0x1,  
    FEC_CORRECT_DIRECTION_Y,  
    FEC_CORRECT_DIRECTION_XY  
} fec_correct_direction_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------------|---------|
| FEC_CORRECT_DIRECTION_X | 只校正X方向 |
| FEC_CORRECT_DIRECTION_Y | 只校正Y方向 |
| FEC_CORRECT_DIRECTION_XY | XY方向都校正 |

`rk_aiq_fec_attrib_t`

【说明】

fec属性配置

【定义】

```
typedef struct rk_aiq_fec_cfg_s {
    unsigned int en;
    int bypass;
    int correct_level;
    fec_correct_direction_t direction;
} rk_aiq_fec_cfg_t;
```

【成员】

| 成员名称 | 描述 |
|---------------|-------------------|
| en | 使能/关闭fec |
| bypass | bypass fec |
| correct_level | 设置fec校正级别 (0-255) |
| direction | 设置fec校正方向 |

性能优化

- fec模块的处理速度受ddr带宽影响，不优化的情况下，fec在5Mega和8Mega摄像头支持的帧率为

| sensor | Maximum fps |
|----------------------------------|-------------|
| imx415 3840*2160 Raw10bit NORMAL | 17fps |
| imx415 3840*2160 Raw10bit HDR2 | 15fps |
| imx335 2592*1944 Raw10bit NORMAL | 22fps |
| imx335 2592*1944 Raw10bit HDR2 | 20fps |

- 优化fec处理速度方案如下

1. ispp video buffer使用cma内存，修改如下
 - 内核dts修改ispp使用cma buffer，size根据实际sensor分辨率分配

```
diff --git a/arch/arm/boot/dts/rv1126-ipc.dtsi
b/arch/arm/boot/dts/rv1126-ipc.dtsi
index d9c69e9..3580f0b 100644
--- a/arch/arm/boot/dts/rv1126-ipc.dtsi
+++ b/arch/arm/boot/dts/rv1126-ipc.dtsi
@@ -169,7 +169,7 @@
};

&rkiispp_mmu {
-    status = "okay";
+    status = "disabled";
};

&rkvdec {
diff --git a/arch/arm/boot/dts/rv1126.dtsi
b/arch/arm/boot/dts/rv1126.dtsi
```

```

index 59b97244..77a8f81 100644
--- a/arch/arm/boot/dts/rv1126.dtsi
+++ b/arch/arm/boot/dts/rv1126.dtsi
@@ -320,7 +320,7 @@
    isp_reserved: isp {
        compatible = "shared-dma-pool";
        reusable;
-       size = <0x10000000>;
+       size = <0x20000000>;
    };

    ramoops: ramoops@8000000 {
@@ -1962,7 +1962,8 @@
        assigned-clock-rates = <5000000000>, <2500000000>,
                               <4000000000>;
        power-domains = <&power RV1126_PD_ISPP>;
-       iommus = <&rkippp_mmu>;
+       /* iommus = <&rkippp_mmu>; */
+       memory-region = <&isp_reserved>;
        status = "disabled";
    };

```

- app确认Streaming I/O 方式为Memory Mapping 方式

```

memset(&reqbuf, 0, sizeof(reqbuf));
reqbuf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
reqbuf.memory = V4L2_MEMORY_MMAP;
reqbuf.count = 20;

if (-1 == ioctl(fd, VIDIOC_REQBUFS, &reqbuf)) {
    if (errno == EINVAL)
        printf("Video capturing or mmap-streaming is not
supported\n");
    else
        perror("VIDIOC_REQBUFS");
    exit(EXIT_FAILURE);
}

```

2. fec只校正y方向的畸变

用户可以通过调用uapi配置，fec只校正y方向。

该方案的畸变校正方式为：ldch校x方向 + fec校正y方向

```
// 此函数需要在rkaiq 执行prepare之前调用，才会生效
rk_aiq_uapi_setFecCorrectDirection(ctx, FEC_CORRECT_DIRECTION_Y);
```

3. 确认isp/ispp频率

```
isp clk : 600M,  isp aclk: 500M,  qos: 0x101
ispp clk: 500M,  isp aclk: 500M , qos m0: 0x202,  m1:0x302
```

```
# cat /sys/kernel/debug/clk/clk_summary |grep isp
clk_isp_div      0      0      0  142857143      0      0
50000
clk_isp_np5      0      1      0  5000000000      0      0
50000
```

| | | | | | | |
|-----------------|---|---|---|------------|---|---|
| clk_ispp | 0 | 2 | 0 | 5000000000 | 0 | 0 |
| 50000 | | | | | | |
| clk_ispp_np5 | 0 | 0 | 0 | 111111112 | 0 | 0 |
| 50000 | | | | | | |
| clk_ispp_div | 0 | 1 | 0 | 5000000000 | 0 | 0 |
| 50000 | | | | | | |
| clk_ispp | 0 | 2 | 0 | 5000000000 | 0 | 0 |
| 50000 | | | | | | |
| aclk_ispp | 0 | 2 | 0 | 5940000000 | 0 | 0 |
| 50000 | | | | | | |
| hclk_ispp | 0 | 2 | 0 | 2970000000 | 0 | 0 |
| 50000 | | | | | | |
| aclk_pdispp_np5 | 0 | 0 | 0 | 4752000000 | 0 | 0 |
| 50000 | | | | | | |
| aclk_pdispp_div | 0 | 1 | 0 | 5940000000 | 0 | 0 |
| 50000 | | | | | | |
| aclk_pdispp | 0 | 2 | 0 | 5940000000 | 0 | 0 |
| 50000 | | | | | | |
| aclk_pdispp_niu | 0 | 0 | 0 | 5940000000 | 0 | 0 |
| 0 50000 | | | | | | |
| aclk_ispp | 0 | 4 | 0 | 5940000000 | 0 | 0 |
| 50000 | | | | | | |
| hclk_pdispp | 0 | 1 | 0 | 2970000000 | 0 | 0 |
| 50000 | | | | | | |
| hclk_pdispp_niu | 0 | 0 | 0 | 2970000000 | 0 | 0 |
| 0 50000 | | | | | | |
| hclk_ispp | 0 | 4 | 0 | 2970000000 | 0 | 0 |
| 50000 | | | | | | |

LDCH

功能描述

光学系统、电子扫描系统失真而引起的斜视畸变、枕形、桶形畸变等，都可能使图像产生几何特性失真。图像的畸变校正是以某种变换方式将畸变图像转换为理想图像的过程。

该模块只对x方向的图像畸变进行校正。

功能级API参考

`rk_aiq_uapi_setLdchEn`

【描述】 水平畸变校正功能开关。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 校正开关 | 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_api_aldch.h、rk_aiq_uapi_aldch_int.h
- 库文件: librkaiq.so

rk_aiq_uapi_setLdchCorrectLevel

【描述】 设置水平畸变校正等级。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_aldch.h、rk_aiq_uapi_aldch_int.h
- 库文件: librkaiq.so

模块级API参考

rk_aiq_user_api_aldch_SetAttrib

【描述】

设置fec属性。

【语法】

```
XCamReturn
rk_aiq_user_api_aldch_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_ldch_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 | ldch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_api_aldch.h、rk_aiq_uapi_aldch_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_aldch_GetAttrib

【描述】

获取fec属性。

【语法】

```
XCamReturn
rk_aiq_user_api_aldch_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_ldch_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 | ldch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_api_aldch.h、rk_aiq_uapi_aldch_int.h
- 库文件: librkaiq.so

模块级API数据类型

rk_aiq_ldch_attrib_t

【说明】

ldch属性配置

【定义】

```
typedef struct rk_aiq_ldch_cfg_s {
    unsigned int en;
    int correct_level;
} rk_aiq_ldch_cfg_t;
```

【成员】

| 成员名称 | 描述 |
|---------------|--------------------|
| en | 使能/关闭ldch |
| correct_level | 设置ldch校正级别 (0-255) |

HDR

功能描述

HDR(高动态范围成像, High Dynamic Range Imaging, HDRI 或 HDR), 在计算机图形学与摄影中, 是通过计算实现使用现有设备获得比普通数字图像技术更大曝光动态范围 (即更大的明暗差别) 图像的一种技术。HDR 的目的就是要正确地还原出超出现有设备动态范围的现实场景光亮比例。

重要概念

- 模块包含Merge和Tmo两个部分, Tmo可单独使用, Merge需要与Tmo一起使用

功能级API参考

`rk_aiq_uapi_setHDRMode`

【描述】 设置HDR工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_setHDRMode(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_imgproc.h

- 库文件: librkaiq.so

rk_aiq_uapi_getHDRMode

【描述】 获取HDR工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_getHDRMode(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMHDRStrth

【描述】 设置手动模式下的HDR强度。

【语法】

```
XCamReturn rk_aiq_uapi_setMHDRStrth(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 | 强度 取值范围: [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 |

【需求】

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

rk_aiq_uapi_getMHDRStrth

【描述】 获取手动模式下的HDR强度。

【语法】

```
XCamReturn rk_aiq_uapi_getMHDRStrth(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 | 强度 | 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_api_imgproc.h
- 库文件: librkaiq.so

功能级API数据类型

hdr_OpMode_t

【说明】

定义白平衡工作模式

【定义】

```

typedef enum hdr_OpMode_s {
    HDR_OpMode_Api_OFF    = 0,
    HDR_OpMode_Auto        = 1,
    HDR_OpMode_MANU        = 2,
    HDR_OpMode_SET_LEVEL   = 3,
    HDR_OpMode_DarkArea    = 4,
    HDR_OpMode_Tool         = 5,
} hdr_OpMode_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------|-----------------------|
| HDR_OpMode_Api_OFF | api关闭模式 |
| HDR_OpMode_Auto | 自动模式 |
| HDR_OpMode_MANU | 手动模式 |
| HDR_OpMode_SET_LEVEL | 快速模式，通过设置等级调整HDR效果 |
| HDR_OpMode_DarkArea | 暗区提升模式，linear、HDR均可使用 |
| HDR_OpMode_Tool | 工具模式 |

FastMode_t

【说明】

定义HDR快速模式属性

【定义】

```

typedef struct FastMode_s {
    int level;
} FastMode_t;

```

【成员】

| 成员名称 | 描述 |
|-------|--------|
| level | 快速模式等级 |

DarkArea_t

【说明】

定义HDR暗区提升模式属性

【定义】

```

typedef struct DarkArea_s {
    int level;
} DarkArea_t;

```

【成员】

| 成员名称 | 描述 |
|-------|--------|
| level | 暗区提升等级 |

模块级API参考

rk_aiq_uapi_ahdr_SetAttrib

【描述】

设定 HDR软件属性。

【语法】

```
XCamReturn
rk_aiq_uapi_ahdr_SetAttrib(RkAiqAlgoContext* ctx,
                             ahdr_attrib_t attr,
                             bool need_sync);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| attr | AHDR软件属性结构体 | 输入 |
| need_sync | 参数更新开关 | 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_uapi_ahdr_int.h
- 库文件: librkaiq.so

【说明】

rk_aiq_uapi_ahdr_GetAttrib

【描述】

获取 HDR软件属性。

【语法】

```
XCamReturn
rk_aiq_uapi_ahdr_GetAttrib(RkAiqAlgoContext* ctx,
                            ahdr_attrib_t attr,
                            bool need_sync);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|-------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| attr | AHDR软件属性结构体 | 输入 |
| need_sync | 参数更新开关 | 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_uapi_ahdr_int.h
- 库文件: librkaiq.so

【说明】

模块级API数据类型

hdr_OpMode_t

【说明】

定义HDR工作模式

【定义】

```
typedef enum hdr_OpMode_s {
    HDR_OpMode_Api_OFF = 0,
    HDR_OpMode_Auto     = 1,
    HDR_OpMode_MANU     = 2,
    HDR_OpMode_SET_LEVEL = 3,
    HDR_OpMode_DarkArea  = 4,
    HDR_OpMode_Tool      = 5,
} hdr_OpMode_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|-----------------------|
| HDR_OpMode_Api_OFF | api关闭模式 |
| HDR_OpMode_Auto | 自动模式 |
| HDR_OpMode_MANU | 手动模式 |
| HDR_OpMode_SET_LEVEL | 快速模式，通过设置等级调整HDR效果 |
| HDR_OpMode_DarkArea | 暗区提升模式，linear、HDR均可使用 |
| HDR_OpMode_Tool | 工具模式 |

mgeCtrlData_t

【说明】

定义自动Merge参数属性

【定义】

```
typedef struct mgeCtrlData_s {
    float stCoef;
    float stCoefMax;
    float stCoefMin;
    int stSmthMax;
    int stSmthMin;
    int stOfstMax;
    int stOfstMin;
} mgeCtrlData_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|----------|
| stCoef | 当前控制参数 |
| stCoefMax | 控制参数最大值 |
| stCoefMin | 控制参数最小值 |
| stSmthMax | 曲线斜率最大值 |
| stSmthMin | 曲线斜率最小值 |
| stOfstMax | 曲线偏移值最大值 |
| stOfstMin | 曲线偏移值最小值 |

amgeAttr_t

【说明】

定义自动Merge属性

【定义】

```
typedef struct amgeAttr_s {
    mgeCtrlData_t stMDCurveLM;
    mgeCtrlData_t stMDCurveMS;
    mgeCtrlData_t stOECurve;
} amgeAttr_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|-----------|
| stMDCurveLM | 长中帧运动曲线参数 |
| stMDCurveMS | 中短帧运动曲线参数 |
| stOECurve | 过曝曲线参数 |

tmoCtrlData_t

【说明】

定义自动Tmo参数属性

【定义】

```
typedef struct tmoCtrlData_s {
    float stCoef;
    float stCoefMax;
    float stCoefMin;
    int stMax;
    int stMin;
} tmoCtrlData_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|----------|
| stCoef | 当前控制参数 |
| stCoefMax | 控制参数最大值 |
| stCoefMin | 控制参数最小值 |
| stMax | 被控制参数最大值 |
| stMin | 被控制参数最小值 |

AGlobalTmoData_t

【说明】

定义自动Tmo中Global Tmo参数属性

【定义】

```
typedef struct AGlobalTmoData_s {
    bool en;
    float stCoef;
    float stCoefMax;
    float stCoefMin;
    int stMax;
    int stMin;
} AGlobalTmoData_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|----------|
| en | 开关功能 |
| stCoef | 当前控制参数 |
| stCoefMax | 控制参数最大值 |
| stCoefMin | 控制参数最小值 |
| stMax | 被控制参数最大值 |
| stMin | 被控制参数最小值 |

atmoAttr_t

【说明】

定义自动Tmo属性

【定义】

```
typedef struct atmoAttr_s {
    tmoCtrlData_t    stGlobeLuma;
    tmoCtrlData_t    stDtlsLL;
    tmoCtrlData_t    stDtlsHL;
    tmoCtrlData_t    stLocalTMO;
    AGlobalTmoData_t stGlobalTMO;
} atmoAttr_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|--------------|
| stGlobeLuma | 整体亮度参数 |
| stDtlsLL | 低亮区参数 |
| stDtlsHL | 高亮区参数 |
| stLocalTMO | Local Tmo参数 |
| stGlobalTMO | Global Tmo参数 |

ahdrAttr_t

【说明】

定义自动HDR属性

【定义】

```
typedef struct ahdrAttr_s {
    bool        bUpdateTmo;
    bool        bUpdateMge;
    amgeAttr_t  stMgeAuto;
    atmoAttr_t  stTmoAuto;
} ahdrAttr_t;
```

【成员】

| 成员名称 | 描述 |
|------------|-------------|
| bUpdateTmo | 更新Tmo参数开关 |
| bUpdateMge | 更新Merge参数开关 |
| stMgeAuto | 自动Merge参数 |
| stTmoAuto | 自动Tmo参数 |

mmgeAttr_t

【说明】

定义手动HDR中Merge工作模式

【定义】

```
typedef struct mmgeAttr_s {
    float OECurve_smooth;
    float OECurve_offset;
    float MDCurveLM_smooth;
    float MDCurveLM_offset;
    float MDCurveMS_smooth;
    float MDCurveMS_offset;
    float dampOE;
    float dampMDLM;
    float dampMDMS;
} mmgeAttr_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|-------------|
| OECurve_smooth | 过曝曲线斜率 |
| OECurve_offset | 过曝曲线偏移值 |
| MDCurveLM_smooth | 长中帧运动曲线斜率 |
| MDCurveLM_offset | 长中帧运动曲线偏移值 |
| MDCurveMS_smooth | 中短帧运动曲线斜率 |
| MDCurveMS_offset | 中短帧运动曲线偏移值 |
| dampOE | 过曝曲线平滑系数 |
| dampMDLM | 长中帧运动曲线平滑系数 |
| dampMDMS | 中短帧运动曲线平滑系数 |

mtmoAttr_t

【说明】

定义手动HDR中Tmo工作模式

【定义】

```
typedef struct mtmoAttr_s {
    float stGlobeLuma;
    float stDtlsHL;
    float stDtlsLL;
    float stLocalTMOStrength;
    float stGlobalTMOStrength;
    float damp;
} mtmoAttr_t;
```

【成员】

| 成员名称 | 描述 |
|---------------------|--------------|
| stGlobeLuma | 整体亮度参数 |
| stDtlsHL | 低光处亮度参数 |
| stDtlsLL | 高光处亮度参数 |
| stLocalTMOStrength | Local Tmo参数 |
| stGlobalTMOStrength | Global Tmo参数 |
| damp | Tmo参数平滑系数 |

mhdrAttr_t

【说明】

定义手动HDR工作模式

【定义】

```
typedef struct mhdrAttr_s {
    bool bUpdateTmo;
    bool bUpdateMge;
    mmgeAttr_t stMgeManual;
    mtmoAttr_t stTmoManual;
} mhdrAttr_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|-------------|
| bUpdateTmo | 更新Tmo参数开关 |
| bUpdateMge | 更新Merge参数开关 |
| stMgeManual | 手动Merge参数 |
| stTmoManual | 手动Tmo参数 |

FastMode_t

【说明】

定义HDR快速模式属性

【定义】

```
typedef struct FastMode_s {
    int level;
} FastMode_t;
```

【成员】

| 成员名称 | 描述 |
|-------|--------|
| level | 快速模式等级 |

DarkArea_t

【说明】

定义HDR暗区提升模式属性

【定义】

```
typedef struct DarkArea_s {
    int level;
} DarkArea_t;
```

【成员】

| 成员名称 | 描述 |
|-------|--------|
| level | 暗区提升等级 |

CurrCtlData_t

【说明】

定义当前控制量属性

【定义】

```
typedef struct CurrCtlData_s {
    int SceneMode;
    float GlobalLumaMode;
    float DetailsHighLightMode;
    float DetailsLowLightMode;
    float GlobalTMOMode;
    float LocalTMOMode;
    float EnvLv;
    float MoveCoef;
    float ISO;
    float OEPdf;
    float FocusLuma;
    float DarkPdf;
    float DynamicRange;
} CurrCtlData_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|-----------------|
| SceneMode | 场景模式 |
| GlobalLumaMode | 整体亮度参数控制值模式 |
| DetailsHighLightMode | 高光处亮度参数控制值模式 |
| DetailsLowLightMode | 低光处亮度参数控制值模式 |
| GlobalTmoMode | Global Tmo控制值模式 |
| LocalTMOMode | Local Tmo控制值模式 |
| Envlv | 环境亮度 |
| MoveCoef | 运动系数 |
| ISO | ISO |
| OEPdf | 过曝区域占比 |
| FocusLuma | 对焦处亮度 |
| DarkPdf | 暗区占比 |
| DynamicRange | 画面动态范围 |

CurrRegData_t

【说明】

定义当前被控制量属性

【定义】

```
typedef struct CurrRegData_s{
    float OECurve_smooth;
    float OECurve_offset;
    float MDCurveLM_smooth;
    float MDCurveLM_offset;
    float MDCurveMS_smooth;
    float MDCurveMS_offset;
    float GlobalLuma;
    float DetailsLowlight;
    float DetailsHighlight;
    float LocalTmoStrength;
    float GlobalTmoStrength;
} CurrRegData_t;
```

【成员】

| 成员名称 | 描述 |
|-------------------|--------------|
| OECurve_smooth | 过曝曲线斜率 |
| OECurve_offset | 过曝曲线偏移值 |
| MDCurveLM_smooth | 长中帧运动曲线斜率 |
| MDCurveLM_offset | 长中帧运动曲线偏移值 |
| MDCurveMS_smooth | 中短帧运动曲线斜率 |
| MDCurveMS_offset | 中短帧运动曲线偏移值 |
| GlobalLuma | 整体亮度 |
| DetailsLowlight | 低光处亮度 |
| DetailsHighlight | 高光处亮度 |
| LocalTmoStrength | Local Tmo力度 |
| GlobalTmoStrength | Global Tmo力度 |

CalibDb_HdrMerge_t

【说明】

定义stTool中Merge属性

【定义】

```
typedef struct CalibDb_HdrMerge_s{
    float envLevel[13];
    float oeCurve_smooth[13];
    float oeCurve_offset[13];
    float moveCoef[13];
    float mdCurveLm_smooth[13];
    float mdCurveLm_offset[13];
    float mdCurveMs_smooth[13];
    float mdCurveMs_offset[13];
    float oeCurve_damp;
    float mdCurveLm_damp;
    float mdCurveMs_damp;
} CalibDb_HdrMerge_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|-------------|
| envLevel | 环境亮度 |
| oeCurve_smooth | 过曝曲线斜率 |
| oeCurve_offset | 过曝曲线偏移值 |
| mdCurveLm_smooth | 长中帧运动曲线斜率 |
| mdCurveLm_offset | 长中帧运动曲线偏移值 |
| mdCurveMs_smooth | 中短帧运动曲线斜率 |
| mdCurveMs_offset | 中短帧运动曲线偏移值 |
| oeCurve_damp | 过曝曲线平滑系数 |
| mdCurveLm_damp | 长中帧运动曲线平滑系数 |
| mdCurveMs_damp | 中短帧运动曲线平滑系数 |

TMO_en_t

【说明】

定义stTool中Tmo的开关属性

【定义】

```
typedef struct TMO_en_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float en;
} TMO_en_t;
```

【成员】

| 成员名称 | 描述 |
|------|------|
| name | 模式名称 |
| en | 开关功能 |

GlobalLuma_t

【说明】

定义stTool中Tmo的整体亮度属性

【定义】

```
typedef struct GlobalLuma_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float GlobalLumaMode;
    float envLevel[13];
    float ISO[13];
    float Tolerance;
    float globalLuma[13];
} GlobalLuma_t;
```

【成员】

| 成员名称 | 描述 |
|----------------|-----------|
| name | 模式名称 |
| GlobalLumaMode | 整体亮度控制量模式 |
| envLevel | 环境亮度 |
| ISO | ISO |
| Tolerance | 控制量容忍值 |
| globalLuma | 整体亮度 |

DetailsHighLight_t

【说明】

定义stTool中Tmo的高光处亮度属性

【定义】

```
typedef struct DetailsHighLight_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float DetailsHighLightMode;
    float OEPdf[13];
    float EnvLv[13];
    float Tolerance;
    float detailsHighLight[13];
} DetailsHighLight_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|------------|
| name | 模式名称 |
| DetailsHighLightMode | 高光处亮度控制量模式 |
| OEPdf | 过曝区域占比 |
| EnvLv | 环境亮度 |
| Tolerance | 控制量容忍度 |
| detailsHighLight | 高光处亮度 |

DetailsLowLight_t

【说明】

定义stTool中Tmo的低光处亮度属性

【定义】

```

typedef struct DetailsLowLight_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float DetailsLowLightMode;
    float FocusLuma[13];
    float DarkPdf[13];
    float ISO[13];
    float Tolerance;
    float detailsLowLight[13];
} DetailsLowLight_t;

```

【成员】

| 成员名称 | 描述 |
|---------------------|------------|
| name | 模式名称 |
| DetailsLowLightMode | 低光处亮度控制量模式 |
| FocusLuma | 对角处亮度 |
| DarkPdf | 暗区亮度占比 |
| ISO | ISO |
| Tolerance | 控制量容忍度 |
| detailsLowLight | 低光处亮度 |

LocalTMO_t

【说明】

定义stTool中Tmo的Local Tmo属性

【定义】

```

typedef struct LocalTMO_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float LocalTMOMode;
    float DynamicRange[13];
    float EnvLv[13];
    float Tolerance;
    float Strength[13];
} LocalTMO_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|----------------|
| name | 模式名称 |
| LocalTMOMode | Local Tmo控制量模式 |
| DynamicRange | 画面动态范围 |
| EnvLv | 环境亮度 |
| Tolerance | 控制量容忍值 |
| Strength | Local Tmo力度 |

'GlobaTMO_t

【说明】

定义stTool中Tmo的Global Tmo属性

【定义】

```
typedef struct GlobaTMO_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float en;
    float iir;
    float mode;
    float DynamicRange[13];
    float EnvLv[13];
    float Tolerance;
    float Strength[13];
} GlobaTMO_t;
```

【成员】

| 成员名称 | 描述 |
|--------------|-----------------|
| name | 模式名称 |
| en | 开关功能 |
| iir | Global Tmo稳定帧数 |
| mode | Global Tmo控制量模式 |
| DynamicRange | 画面动态范围 |
| EnvLv | 环境亮度 |
| Tolerance | 控制量容忍值 |
| Strength | Global Tmo力度 |

CalibDb_HdrTmo_t

【说明】

定义stTool中Tmo属性

【定义】

```

typedef struct CalibDb_HdrTmo_s{
    TMO_en_t          en[CALIBDB_MAX_MODE_NUM];
    GlobalLuma_t      luma[CALIBDB_MAX_MODE_NUM];
    DetailsHighLight_t HighLight[CALIBDB_MAX_MODE_NUM];
    DetailsLowLight_t LowLight[CALIBDB_MAX_MODE_NUM];
    LocalTMO_t        LocalTMO[CALIBDB_MAX_MODE_NUM];
    GlobalTMO_t       GlobalTMO[CALIBDB_MAX_MODE_NUM];
    float             damp;
} CalibDb_HdrTmo_t;

```

【成员】

| 成员名称 | 描述 |
|-----------|--------------|
| en | 开关功能 |
| luma | 整体亮度参数 |
| HighLight | 高光处亮度参数 |
| LowLight | 低光处亮度参数 |
| LocalTMO | Local Tmo参数 |
| GlobalTMO | Global Tmo参数 |
| damp | Tmo参数平滑系数 |

CalibDb_Ahdr_Para_t

【说明】

定义stTool属性

【定义】

```

typedef struct CalibDb_Ahdr_Para_s{
    CalibDb_HdrMerge_t merge;
    CalibDb_HdrTmo_t tmo;
} CalibDb_Ahdr_Para_t;

```

【成员】

| 成员名称 | 描述 |
|-------|---------|
| merge | Merge参数 |
| tmo | Tmo参数 |

hdrAttr_t

【说明】

定义HDR属性

【定义】

```

typedef struct hdrAttr_s {
    hdr_OpMode_t          opMode;
    ahdrAttr_t             stAuto;
    mhdrAttr_t              stManual;
    FastMode_t                stSetLevel;
    DarkArea_t                  stDarkArea;
    CurrCtlData_t            CtlInfo;
    CurrRegData_t            RegInfo;
    CalibDb_Ahdr_Para_t stTool;
} hdrAttr_t;

```

【成员】

| 成员名称 | 描述 |
|------------|-----------------------|
| opMode | HDR模式 |
| stAuto | 自动HDR |
| stManual | 手动HDR |
| stSetLevel | HDR快速模式，通过设置等级调整HDR效果 |
| stDarkArea | 暗区提升模式，linear、HDR均可使用 |
| CtlInfo | 控制量信息 |
| RegInfo | 参数信息 |
| stTool | HDR工具模式 |

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。

功能级API参考

rk_aiq_uapi_setNRMode

【描述】 设置去噪模式。

【语法】

```
XCamReturn rk_aiq_uapi_setNRMode(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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getNRMode

【描述】 获取当前去噪模式。

【语法】

```
XCamReturn rk_aiq_uapi_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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setANRStrth

【描述】 设置普通去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getANRStrth

【描述】 获取普通去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMSpaNRStrth

【描述】 设置空域去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMSpaNRStrth

【描述】 获取空域去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMTNRStrth

【描述】 设置时域去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMTNRStrth

【描述】 获取时域去噪强度。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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_api_imgproc.h
- 库文件: librkaiq.so

模块级API参考

rk_aiq_user_api_anr_SetAttrib

【描述】

设置去噪算法属性。

【语法】

```
XCamReturn
rk_aiq_user_api_anr_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_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 | 去噪的参数属性 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 此attr属性参数为软件算法最终使用参数结构体，而非IQ xml对应参数结构体。IQ有些参数还需要转换成算法值。但是两者参数基本相同，差异较小。建议使用rk_aiq_user_api_anr_SetIQPara来设置自动参数，便于参数与IQ参数一一对应。

【需求】

- 头文件: rk_aiq_user_api_anr.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

rk_aiq_user_api_anr_GetAttrib

【描述】

获取去噪算法属性。

【语法】

```
XCamReturn
rk_aiq_user_api_anr_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_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 | 去噪的参数属性 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 此attr属性参数为软件算法最终使用参数结构体，而非IQ xml对应参数结构体。IQ有些参数还需要转换成算法值。但是两者参数基本相同，差异较小。建议使用rk_aiq_user_api_anr_GetIQPara来获取自动参数，便于参数与IQ参数一一对应。

【需求】

- 头文件: rk_aiq_user_api_anr.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

rk_aiq_user_api_anr_SetIQPara

【描述】

设置去噪IQxml参数。

【语法】

```
XCamReturn
rk_aiq_user_api_anr_SetIQPara(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_IQPara_t *para);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|--------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| para | 去噪的IQxml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 |

【注意】

- 此para属性参数为IQ xml对应参数结构体。

【需求】

- 头文件: rk_aiq_user_api_anr.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

rk_aiq_user_api_anr_GetIQPara

【描述】

获取去噪IQxml参数。

【语法】

```
XCamReturn
rk_aiq_user_api_anr_GetIQPara(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_IQPara_t *para);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|--------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| para | 去噪的IQxml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 |

【注意】

- 此para属性参数为IQ xml对应参数结构体。

【需求】

- 头文件: rk_aiq_user_api_anr.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

模块级API数据类型

rk_aiq_nr_attrib_t

【说明】

定义去噪模块的参数

【定义】

```

typedef struct rk_aiq_nr_attrib_s {
    ANROPMode_t eMode;
    ANR_Auto_Attr_t stAuto;
    ANR_Manual_Attr_t stManual;
} rk_aiq_nr_attrib_t;

```

【成员】

| 成员名称 | 描述 |
|----------|------------|
| eMode | 去噪模块属性 |
| stAuto | 去噪模块自动模式参数 |
| stManual | 去噪模块手动模式参数 |

ANROPMode_t

【说明】

定义去噪模块的模式

【定义】

```

typedef enum ANROPMode_e {
    ANR_OP_MODE_INVALID      = 0,
    ANR_OP_MODE_AUTO         = 1,
    ANR_OP_MODE_MANUAL       = 2,
    ANR_OP_MODE_MAX
} ANROPMode_t;

```

【成员】

| 成员名称 | 描述 |
|---------------------|-------------------|
| ANR_OP_MODE_INVALID | 去噪模块无效模式 |
| ANR_OP_MODE_AUTO | 去噪模块自动模式 |
| ANR_OP_MODE_MANUAL | 去噪模块手动模式 |
| ANR_OP_MODE_MAX | 去噪模块模式最大值，是一个无效模式 |

ANR_Auto_Attr_t

【说明】

定义去噪模块的自动属性

【定义】

```

typedef struct ANR_Auto_Attr_s
{
    //bayernr
    int bayernrEn;
}

```

```

RKAhr_Bayernr_Params_t stBayernrParams;
RKAhr_Bayernr_Params_Select_t stBayernrParamSelect;

//mfnr
int mfnrEn;
RKAhr_Mfnr_Params_t stMfnrParams;
RKAhr_Mfnr_Params_Select_t stMfnrParamSelect;

//ynr
int ynrEn;
RKAhr_Ynr_Params_t stYnrParams;
RKAhr_Ynr_Params_Select_t stYnrParamSelect;

//uvnr
int uvnrEn;
RKAhr_Uvnr_Params_t stUvnrParams;
RKAhr_Uvnr_Params_Select_t stUvnrParamSelect;

RKAhr_Mfnr_Dynamic_t stMfnr_dynamic;

} ANR_Auto_Attr_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------|--------------------------|
| int bayernrEn | bayernr模块使能位 |
| stBayernrParams | bayernr模块各个iso对应算法属性参数 |
| stBayernrParamSelect | bayernr模块根据当前iso计算出来属性参数 |
| mfnrEn | mfnr模块使能位 |
| stMfnrParams | mfnr模块各个iso对应算法属性参数 |
| stMfnrParamSelect | mfnr模块根据当前iso计算出来属性参数 |
| ynrEn | ynr模块使能位 |
| stYnrParams | ynr模块各个iso对应算法属性参数 |
| stYnrParamSelect | ynr模块根据当前iso计算出来属性参数 |
| uvnrEn | uvnr模块使能位 |
| stUvnrParams | uvnr模块各个iso对应算法属性参数 |
| stUvnrParamSelect | uvnr模块根据当前iso计算出来属性参数 |

ANR_Manual_Attr_t

【说明】

定义去噪模块的手动属性

【定义】

```

typedef struct ANR_Manual_Attr_s
{
    int bayernrEn;
    RKAnr_Bayernr_Params_Select_t stBayernrParamSelect;

    int mfnrEn;
    RKAnr_Mfnr_Params_Select_t stMfnrParamSelect;

    int ynrEn;
    RKAnr_Ynr_Params_Select_t stYnrParamSelect;

    int uvnrEn;
    RKAnr_Uvnr_Params_Select_t stUvnrParamSelect;

} ANR_Manual_Attr_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------|-----------------|
| int bayernrEn | bayernr模块使能位 |
| stBayernrParamSelect | bayernr手动设置算法参数 |
| mfnrEn | mfnr模块使能位 |
| stMfnrParamSelect | mfnr手动设置算法参数 |
| ynrEn | ynr模块使能位 |
| stYnrParamSelect | ynr手动设置算法参数 |
| uvnrEn | uvnr模块使能位 |
| stUvnrParamSelect | uvnr手动设置算法参数 |

rk_aiq_nr_IQPara_t

【说明】

定义去噪模块的IQ参数结构体

【定义】

```

typedef struct rk_aiq_nr_IQPara_s {
    int module_bits;

    CalibDb_BayerNr_t stBayernrPara;
    CalibDb_MFNR_t stMfnrPara;
    CalibDb_UVNR_t stUvnrPara;
    CalibDb_YNR_t stYnrPara;

} rk_aiq_nr_IQPara_t;

```

【成员】

| 成员名称 | 描述 |
|---------------|--------------------------------|
| module_bits | nr4个模块对应标志位，根据标志位来分别设置各个模块IQ参数 |
| stBayernrPara | bayernr模块对应IQ参数 |
| stMfnrPara | mfnr模块对应IQ参数 |
| stUvnrPara | uvnr模块对应IQ参数 |
| stYnrPara | ynr模块对应IQ参数 |

rk_aiq_nr_module_t

【说明】

定义去噪模块的设置IQ参数对应模块对应的标志位，各个模块可分开设置，也可统一设置，靠这个标志位识别。

【定义】

```
typedef enum rk_aiq_nr_module_e{
    ANR_MODULE_BAYERNR      = 0,
    ANR_MODULE_MFNR         = 1,
    ANR_MODULE_UVNR         = 2,
    ANR_MODULE_YNR          = 3,
} rk_aiq_nr_module_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------|----------------------|
| ANR_MODULE_BAYERNR | bayernr模块对应标志位为bit 0 |
| ANR_MODULE_MFNR | mfnr模块对应标志位为bit 1 |
| ANR_MODULE_UVNR | uvnr模块对应标志位为bit 2 |
| ANR_MODULE_YNR | ynr模块对应标志位为bit 3 |

CalibDb_BayerNr_t

【说明】

bayernr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_BayerNr_s {
    int enable;
    char version[64];
    CalibDb_BayerNr_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];
} CalibDb_BayerNr_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|----------------------------------|
| enable | bayernr模块使能位 |
| version | 硬件参数版本号 |
| mode_cell | 不同模式，如normal, hdr. gray模式对应去噪参数。 |

CalibDb_BayerNr_ModeCell_t

【说明】

bayernr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_BayerNr_ModeCell_s{
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    CalibDb_BayerNR_Params_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
} CalibDb_BayerNr_ModeCell_t;
```

【成员】

| 成员名称 | 描述 |
|---------|---------------------------------------|
| name | bayernr模块不同模式名称，normal, hdr, gray三种模式 |
| setting | bayernr一个模块下，还有lcg, hcg等对应2种配置。 |

CalibDb_BayerNR_Params_t

【说明】

定义bayernr模块IQ参数。

【定义】

```
typedef struct CalibDb_BayerNR_Params_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    float iso[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
    float filtPara[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
    float luLevel[8];
    float luLevelVal[8];
    float luRatio[8][CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
    float fixw[4][CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
    float lambda;
    unsigned char gauss_en;

    //下面参数目前暂未使用
    float RGainoff;
    float RGainFilp;
    float BGainoff;
    float BGainFilp;
    float edgeSoftness;
}
```

```

float gaussweight0;
float gaussweight1;
float bilEdgeFilter;
float bilFilterStreng[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
float bilEdgeSoft;
float bilEdgeSoftRatio;
float bilRegwgt;
} CalibDb_BayerNR_Params_t;

```

【成员】

| 成员名称 | 描述 |
|-------------|--|
| snr_mode | 噪声模式， HSNR, LSNR, 对应hcg, lcg模式 |
| sensor_mode | sensor hcg, lcg模式, 如未有此两个模式, 默认使用lcg里的参数 |
| iso | 不同iso对应的参数配置 |
| filtPara | 单帧降噪力度, 取值范围[0, 15.9]。此值越大, 降噪力度越大。 |
| luLevel | 根据像素亮度调整降噪强度, 此为x轴亮度, 0-255划分出8个点。 |
| luRatio | 与上面luLevel ——对应, 为y轴降噪强度。 |
| fixW | 分段噪声的4个等级的权重。取值范围[0 7.9]。 |
| lamda | 分段噪声阈值, lamda 越大, 调节范围越大。取值范围[0 16383]。 |
| gauss_en | 高斯导向滤波 3x3 使能位. 0: 关闭, 1: 打开。 |
| 其余参数 | 其余参数目前模块均未使用, 直接忽略。 |

CalibDb_MFNR_t

【说明】

mfnr模块IQ参数结构体

【定义】

```

typedef struct CalibDb_MFNR_s {
    int enable;
    char version[64];
    unsigned char local_gain_en;
    unsigned char mode_3to1;
    CalibDb_MFNR_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];

    //下面参数目前暂未使用
    unsigned char max_level;
    unsigned char max_level_uv;
    unsigned char back_ref_num;
    struct CalibDb_awb_uv_ratio_s uv_ratio[4];
} CalibDb_MFNR_t;

```

【成员】

| 成员名称 | 描述 |
|---------------|---|
| enable | mfnr模块使能位 |
| version | 硬件参数版本号 |
| local_gain_en | 是否支持采用前级的局部增益模式。 1: 前级局部增益模式, 0: 全局增益模式 |
| mode_3to1 | 是否使用3to1模式。 1: 使用3to1模式, 0: 使用2to1模式。 |
| mode_cell | 不同模式, 如normal, hdr, gray模式对应去噪参数。 |
| 其余参数 | 参数目前未使用 |

CalibDb_MFNR_ModeCell_t

【说明】

mfnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_MFNR_ModeCell_s {
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    CalibDb_MFNR_Setting_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
    CalibDb_MFNR_Dynamic_t dynamic;
} CalibDb_MFNR_ModeCell_t;
```

【成员】

| 成员名称 | 描述 |
|---------|-------------------------------------|
| name | mfnr模块不同模式名称, normal, hdr, gray三种模式 |
| setting | mfnr模块下, 还有lcg, hcg等对应2种配置。 |
| dynamic | 根据曝光动态打开或者关闭mfnr功能。 |

CalibDb_MFNR_Dynamic_t

【说明】

mfnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_MFNR_Dynamic_s {
    int enable;
    float lowth_iso;
    float lowth_time;
    float highth_iso;
    float highth_time;
} CalibDb_MFNR_Dynamic_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|---|
| enable | 动态开关mfnr模块功能的使能位，1：动态开关 0：保持常开或者常关状态 |
| lowth_iso | 曝光对应iso低阈值。当低于此曝光iso和下面曝光时间的低阈值时，才关闭mfnr。 |
| lowth_time | 曝光对应曝光时间低阈值。 |
| highth_iso | 曝光对应iso高阈值。当高于此曝光iso和下面曝光时间的高阈值时，mfnr才打开。 |
| highth_time | 曝光对应曝光时间高阈值。 |

CalibDb_MFNR_Setting_t

【说明】

mfnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_MFNR_Setting_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    struct CalibDb_MFNR_ISO_s mfnr_iso[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
} CalibDb_MFNR_Setting_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|---|
| snr_mode | mfnr模块下，有HSNR和LSNR，对应着hcg和lcg等对应2种配置。 |
| sensor_mode | sensor模式lcg, hcg等对应2种配置。如未有dcg模式，则默认使用lcg对应的参数。 |
| mfnr_iso | 不同iso下，对应不同去噪参数。 |

CalibDb_MFNR_ISO_s

【说明】

mfnr模块IQ参数结构体

【定义】

```
struct CalibDb_MFNR_ISO_s {
    float iso;
    float weight_limit_y[4];
    float weight_limit_uv[3];
    float ratio_frq[4];
    float luma_w_in_chroma[3];

    double noise_curve[5];
}
```

```
double noise_curve_x00;
float y_lo_noiseprofile[4];
float y_hi_noiseprofile[4];
float y_lo_bfscale[4];
float y_hi_bfscale[4];
float y_lumanrpoint[6];
float y_lumanrcurve[6];
float y_denoisestrength;

float uv_lo_noiseprofile[3];
float uv_hi_noiseprofile[3];
float uv_lo_bfscale[3];
float uv_hi_bfscale[3];
float uv_lumanrpoint[6];
float uv_lumanrcurve[6];
float uv_denoisestrength;

float y_lo_lv10_gfdelta[6];
float y_hi_lv10_gfdelta[6];
float y_lo_lv11_gfdelta[3];
float y_hi_lv11_gfdelta[3];
float y_lo_lv12_gfdelta[3];
float y_hi_lv12_gfdelta[3];
float y_lo_lv13_gfdelta[3];
float y_hi_lv13_gfdelta[3];

float uv_lo_lv10_gfdelta[6];
float uv_hi_lv10_gfdelta[6];
float uv_lo_lv11_gfdelta[3];
float uv_hi_lv11_gfdelta[3];
float uv_lo_lv12_gfdelta[3];
float uv_hi_lv12_gfdelta[3];

float lv10_gfsigma[6];
float lv11_gfsigma[3];
float lv12_gfsigma[3];
float lv13_gfsigma[3];

//下面参数目前暂未使用到
float y_lo_denoiseweight[4];
float y_hi_denoiseweight[4];
float uv_lo_denoiseweight[3];
float uv_hi_denoiseweight[3];
};
```

【成员】

| 成员名称 | 描述 |
|---------------------|---------------------------------------|
| iso | 不同iso等级，对应不同去噪参数。 |
| weight_limit_y | 亮度第0-3层前向权重值的最小值。 |
| weight_limit_uv | se度第0-2层前向权重值的最小值。 |
| ratio_frq | 亮度/色度的低频第0层纹理值判断时的低位和高位边界。 |
| luma_w_in_chroma | 色度第0层相似度判断中亮度指导色度去噪的权重值。 |
| noise_curve | y分量噪声sigma曲线，y值。工具标定出来的值。 |
| noise_curve_x00 | y分量噪声sigma曲线，y值。工具标定出来的值。 |
| y_lo_noiseprofile | y分量低频4层噪声曲线修正参数。工具标定出来的值。 |
| y_hi_noiseprofile | y分量高频4层噪声曲线修正参数。工具标定出来的值。 |
| y_lo_bfscale | y分量低频4层去噪力度。 |
| y_hi_bfscale | y分量高频4层去噪力度。 |
| y_lumanrpoint | y分量根据像素亮度，微调不同亮度pixel的去噪力度。此为x轴亮度。 |
| y_lumanrcurve | y分量根据像素亮度，微调不同亮度pixel的去噪力度。此为y轴去噪力度。 |
| y_denoisestrength | y分量高低频4层整体去噪力度调整分量。 |
| uv_lo_noiseprofile | uv分量低频3层噪声曲线修正参数。工具标定出来的值。 |
| uv_hi_noiseprofile | uv分量高频3层噪声曲线修正参数。工具标定出来的值。 |
| uv_lo_bfscale | uv分量低频3层去噪力度。 |
| uv_hi_bfscale | uv分量高频3层去噪力度。 |
| uv_lumanrpoint | uv分量根据像素亮度，微调不同亮度pixel的去噪力度。此为x轴亮度。 |
| uv_lumanrcurve | uv分量根据像素亮度，微调不同亮度pixel的去噪力度。此为y轴去噪力度。 |
| uv_denoisestrength | y分量高低频4层整体去噪力度调整分量。 |
| xxx_xxx_xxx_gfdelta | y和uv低频相似度算子。 |
| xxx_gfsgma | 高斯滤波算子参数。 |
| 其余参数 | 目前暂未使用到。 |

CalibDb_UVNR_t

【说明】

uvnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_UVNR_s {
    int enable;
    char version[64];
    CalibDb_UVNR_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];
} CalibDb_UVNR_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|----------------------------------|
| enable | uvnr模块使能位 |
| version | 硬件参数版本号 |
| mode_cell | 不同模式，如normal, hdr, gray模式对应去噪参数。 |

CalibDb_UVNR_ModeCell_t

【说明】

uvnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_UVNR_ModeCell_s {
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    CalibDb_UVNR_Params_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
} CalibDb_UVNR_ModeCell_t;
```

【成员】

| 成员名称 | 描述 |
|---------|------------------------------------|
| name | uvnr模块不同模式名称，normal, hdr, gray三种模式 |
| setting | uvnr一个模块下，还有lcg, hcg等对应2种配置。 |

CalibDb_UVNR_Params_t

【说明】

uvnr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_UVNR_Params_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    float ISO[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
    float step0_uvgrad_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
```

```

float step0_uvgrad_offset[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step1_median_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_bf_sigmaR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_bf_uvgain[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_bf_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step2_median_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_sigmaR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_uvgain[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step3_bf_sigmaR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step3_bf_uvgain[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step3_bf_ratio[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float kernel_3x3[3];
float kernel_5x5[5];
float kernel_9x9[8];
float kernel_9x9_num;

//下面参数暂未使用到
float step1_nonMed1[4];
float step1_nonBf1[4];
float step1_downSample_w[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_downSample_h[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_downSample_meanSize[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_median_size[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_median_IIR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_bf_size[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step1_bf_isRowIIR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step1_bf_isYcopy[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step2_nonExt_block[4];
float step2_nonMed[4];
float step2_nonBf[4];
float step2_downSample_w[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_downSample_h[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_downSample_meanSize[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_median_size[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_median_IIR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_size[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_sigmaD[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_isRowIIR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step2_bf_isYcopy[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float step3_nonBf3[4];
float step3_bf_size[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step3_bf_isRowIIR[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step3_bf_isYcopy[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;
float step3_bf_sigmaD[CALIBDB_NR_SHARP_MAX_ISO_LEVEL] ;

float sigma_adj_luma[9];
float sigma_adj_ratio[9];
float threshold_adj_luma[9];
float threshold_adj_thre[9];

```

```
} CalibDb_UVNR_Params_t;
```

【成员】

| 成员名称 | 描述 |
|---------------------|---|
| snr_mode | mfnr模块下，有HSNR和LSNR，对应着hcg和lcg等对应2种配置。 |
| sensor_mode | sensor模式lcg, hcg等对应2种配置。如未有dcg模式，则默认使用lcg对应的参数。 |
| ISO | 不同iso等级，对应不同去噪参数。 |
| step0_uvgrad_ratio | 梯度比例因子，值越小，梯度大地方颜色损失越多。取值范围[1, 63]。 |
| step0_uvgrad_offset | 梯度加权因子，值越大，图像颜色损失越多。取值范围[0, 1]。 |
| step1_median_ratio | 4x4下采样后滤波中值后图像和未去噪图像加权因子，取值范围[0, 1] |
| step1_bf_sigmaR | 双边1滤波sigma比例因子，值越大，去噪越大。 |
| step1_bf_uvgain | 双边1的uv比例因子，即y通道指导uv通道去噪权重，取值范围[0, 7.9]。 |
| step1_bf_ratio | 双边1去噪后图像和未去噪图像进行加权的权重，取值范围[0, 1]。 |
| step2_median_ratio | 32x32下采样后滤波中值后图像和未去噪图像加权因子，取值范围[0, 1] |
| step2_bf_sigmaR | 双边2滤波sigma比例因子，值越大，去噪越大。 |
| step2_bf_uvgain | 双边2的uv比例因子，即y通道指导uv通道去噪权重，取值范围[0, 7.9]。 |
| step2_bf_ratio | 双边2去噪后图像和未去噪图像进行加权的权重，取值范围[0, 1]。 |
| step3_bf_sigmaR | 双边3滤波sigma比例因子，值越大，去噪越大。 |
| step3_bf_uvgain | 双边3的uv比例因子，即y通道指导uv通道去噪权重，取值范围[0, 7.9]。 |
| step3_bf_ratio | 双边3去噪后图像和未去噪图像进行加权的权重，取值范围[0, 1]。 |
| kernel_3x3 | 双边3的，距离权重配置。 |
| kernel_5x5 | 双边2的，距离权重配置。 |
| kernel_9x9 | 双边1的，距离权重配置。 |
| kernel_9x9_num | 双边1的，双边使用点模式，共4种。取值范围[0-3]。 |
| 其余参数 | 目前暂未使用。 |

CalibDb_YNR_t

【说明】

ynr模块IQ参数结构体

【定义】

```

typedef struct CalibDb_YNR_s {
    int enable;
    char version[64];
    CalibDb_YNR_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];
} CalibDb_YNR_t;

```

【成员】

| 成员名称 | 描述 |
|-----------|----------------------------------|
| enable | ynr模块使能位 |
| version | 硬件参数版本号 |
| mode_cell | 不同模式，如normal, hdr, gray模式对应去噪参数。 |

CalibDb_YNR_ModeCell_t

【说明】

ynr模块IQ参数结构体

【定义】

```

typedef struct CalibDb_YNR_ModeCell_s {
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    CalibDb_YNR_Setting_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
} CalibDb_YNR_ModeCell_t;

```

【成员】

| 成员名称 | 描述 |
|---------|-----------------------------------|
| name | ynr模块不同模式名称，normal, hdr, gray三种模式 |
| setting | ynr模块下，有lcg, hcg等对应2种配置。 |

CalibDb_YNR_Setting_t

【说明】

ynr模块IQ参数结构体

【定义】

```

typedef struct CalibDb_YNR_Setting_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    CalibDb_YNR_ISO_t ynr_iso[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
} CalibDb_YNR_Setting_t;

```

【成员】

| 成员名称 | 描述 |
|-------------|---|
| snr_mode | ynr模块下，有HSNR和LSNR，对应着hcg和lcg等对应2种配置。 |
| sensor_mode | sensor模式lcg, hcg等对应2种配置。如未有dcg模式，则默认使用lcg对应的参数。 |
| ynr_iso | 不同iso下，对应不同去噪参数。 |

CalibDb_YNR_ISO_t

【说明】

ynr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_YNR_ISO_s {
    float iso;
    double sigma_curve[5];
    float ynr_lci[4];
    float ynr_lhci[4];
    float ynr_hlci[4];
    float ynr_hhci[4];

    float lo_lumaPoint[6];
    float lo_lumaRatio[6];
    float lo_directionStrength;
    float lo_bfscale[4];
    float imerge_ratio;
    float imerge_bound;
    float denoise_weight[4];

    float hi_lumaPoint[6];
    float hi_lumaRatio[6];
    float hi_bfscale[4];
    float hwith_d[4];
    float hi_denoiseStrength;
    float hi_detailMinAdjDnw;
    float hi_denoiseweight[4];

    float y_luma_point[6];
    float hgrad_y_level1[6];
    float hgrad_y_level2[6];
    float hgrad_y_level3[6];
    float hgrad_y_level4[6];
    float hi_soft_thresh_scale[4];
} CalibDb_YNR_ISO_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|--|
| iso | 不同iso等级，对应不同去噪参数。 |
| sigma_curve | 噪声sigma曲线。ynr模块标定工具标定的噪声曲线值。 |
| ynr_lci | 低频频 LL 噪声曲线的比例,1~4 分别对应四层. |
| ynr_lhci | 高频频 LH 噪声曲线的比例,1~4 分别对应四层. |
| ynr_hlci | 高频 HL噪声曲线的比例,1~4 分别对应四层. |
| ynr_hhci | 高频 HH 噪声曲线的比例,1~4 分别对应四层. |
| lo_lumaPoint | 在pixel亮度域上，增加低频sigma调节因子。对应pixel的亮度。取值范围[0 255]. |
| lo_lumaRatio | 在pixel亮度域上，增加低频sigma调节因子。对应调节因子。取值范围[0 2]. |
| lo_directionStrength | 低频sigma 调整因子最大值。取值范围[0 16)。 |
| lo_bfScale | 低频sigma调整因子最小值。分1-4层。取值范围[0-16). |
| imerge_ratio | 低频纹理区域，双边和高斯加权权重，值大，高斯占比多，值下，双边占比多。 |
| imerge_bound | 上面说的纹理阈值。 |
| denoise_weight | 低频去噪的结果和未去噪原图加权比例，值大，去噪占比大，值小，去噪占比小。 |
| hi_lumaPoint | 在pixel亮度域上，增加高频sigma调节因子。对应pixel的亮度。取值范围[0 255]。 |
| hi_lumaRatio | 在pixel亮度域上，增加高频sigma调节因子。对应调节因子。取值范围[0 2)。 |
| hi_bfScale | 高频双边滤波sigma影响因子，值越大，降噪强度越大。取值范围[0-16)。 |
| hwitd_d | 高频1-4层的双边滤波的空间权重。 |
| hi_denoiseStrength | 高频去噪的强度调节. 值越大，降噪强度越大。取值范围[0 16)。 |
| hi_detailMinAdjDnW | 高频限制阈值调整的范围. 取值范围[0 2)。 |
| hi_denoiseWeight | 高频去噪的结果和未去噪原图加权比例，值大，去噪占比大，值小，去噪占比小。 |
| y_luma_point | 梯度分区。梯度计算阈值调节参数 |
| hgrad_y_level1/2/3/4 | 梯度阈值调节因子。高频1-4层，梯度调节因子对应曲线y轴。取值范围[0-4) 。 |
| hi_soft_thresh_scale | 软阈值调节因子。对应高频1-4层整体调节因子。取值范围[0-1) 。 |

Defog

功能描述

Defog 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。

功能级API参考

rk_aiq_uapi_setDhzMode

【描述】

设置去雾工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_setDhzMode(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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getDhzMode

【描述】

获取当前去雾工作模式。

【语法】

```
XCamReturn rk_aiq_uapi_getDhzMode(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 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMDhzStrth

【描述】

设置去雾工作强度。

【语法】

```
XCamReturn rk_aiq_uapi_setMDhzStrth(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 | 强度 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMDhzStrth

【描述】

获取去雾工作强度。

【语法】

```
XCamReturn rk_aiq_uapi_getMDhzStrth(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 | 强度 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_enableDhz

【描述】

开启去雾功能。

【语法】

```
XCamReturn rk_aiq_uapi_enableDhz(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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_disableDhz

【描述】

关闭去雾功能。

【语法】

```
XCamReturn rk_aiq_uapi_disableDhz(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_api_imgproc.h
- 库文件: librkaiq.so

模块级API参考

rk_aiq_user_api_adehaze_setSwAttrib

【描述】

设置去雾参数。

【语法】

```
XCamReturn rk_aiq_user_api_adehaze_setSwAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
adehaze_sw_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 | 去雾参数 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

rk_aiq_user_api_adehaze_getSwAttrib

【描述】

获取当前去雾参数。

【语法】

```
XCamReturn rk_aiq_user_api_adehaze_getSwAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
adehaze_sw_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 | 去雾参数 | 输出 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

ACM

功能描述

ACM(Auto Color Management) 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。

API参考

rk_aiq_uapi_setBrightness

【描述】

设置亮度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 亮度值等级 取值范围：[0,255] 默认值为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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getBrightness

【描述】

获取亮度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 当前亮度等级 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setContrast

【描述】

设置对比度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 对比度等级 取值范围: [0,255] 默认值为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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getContrast

【描述】

获取对比度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 当前对比度等级 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setSaturation

【描述】

设置饱和度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 饱和度等级 取值范围: [0,255] 默认值为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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getSaturation

【描述】

获取饱和度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 当前饱和度等级 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setHue

【描述】

设置色度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 色度等级 取值范围: [0,255] 默认值为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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getHue

【描述】

获取色度等级。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 当前色度等级 | 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_api_imgproc.h
- 库文件: librkaiq.so

Sharpen

功能描述

Sharpen 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

rk_aiq_uapi_setSharpness

【描述】

设置锐化等级。

【语法】

```
XCamReturn rk_aiq_uapi_setSharpness(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 | 锐化等级 取值范围: [0,100] 默认值为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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getSharpness

【描述】

获取锐化等级。

【语法】

```
XCamReturn rk_aiq_uapi_getSharpness(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 | 锐化等级 | 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_api_imgproc.h
- 库文件: librkaiq.so

模块级API参考

rk_aiq_user_api_asharp_SetAttrib

【描述】

设置锐化算法属性。

【语法】

```
XCamReturn
rk_aiq_user_api_asharp_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
                                    rk_aiq_sharp_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 | 锐化的参数属性 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 此attr属性参数为软件算法最终使用参数结构体，而非IQ xml对应参数结构体。IQ有些参数还需要转换成算法值。但是两者参数基本相同，差异较小。建议使用rk_aiq_user_api_asharp_SetIQPara来设置自动参数，便于参数与IQ参数一一对应。

【需求】

- 头文件：rk_aiq_user_api_asharp.h、RkAiqHandleInt.h
- 库文件：librkaiq.so

rk_aiq_user_api_asharp_GetAttrib

【描述】

获取锐化算法属性。

【语法】

```
XCamReturn
rk_aiq_user_api_asharp_GetAttrib(const rk_aiq_sys_t* sys_ctx,
                                    rk_aiq_sharp_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 | 锐化的参数属性 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|-----------|
| 0 | 成功 |
| 非0 | 失败，详见错误码表 |

【注意】

- 此attr属性参数为软件算法最终使用参数结构体，而非IQ xml对应参数结构体。IQ有些参数还需要转换成算法值。但是两者参数基本相同，差异较小。建议使用rk_aiq_user_api_asharp_GetIQPara来获取自动参数，便于参数与IQ参数一一对应。

【需求】

- 头文件：rk_aiq_user_api_asharp.h、RkAiqHandleInt.h
- 库文件：librkaiq.so

rk_aiq_user_api_asharp_SetIQPara

【描述】

设置锐化IQxml参数。

【语法】

```
XCamReturn
rk_aiq_user_api_asharp_SetIQPara(const rk_aiq_sys_ctx_t* sys_ctx,
                                    rk_aiq_sharp_IQpara_t *para);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|--------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| para | 锐化的IQxml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 |

【注意】

- 此para属性参数为IQ xml对应参数结构体。

【需求】

- 头文件: rk_aiq_user_api_asharp.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

rk_aiq_user_api_asharp_GetIQPara

【描述】

获取锐化IQxml参数。

【语法】

```
XCamReturn
rk_aiq_user_api_asharp_GetIQPara(const rk_aiq_sys_ctx_t* sys_ctx,
                                    rk_aiq_sharp_IQpara_t *para);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|---------|--------------|-------|
| sys_ctx | AIQ上下文指针 | 输入 |
| para | 锐化的IQxml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 |

【注意】

- 此para属性参数为IQ xml对应参数结构体。

【需求】

- 头文件: rk_aiq_user_api_asharp.h、RkAiqHandleInt.h
- 库文件: librkaiq.so

模块级API数据类型

rk_aiq_sharp_attrib_t

【说明】

定义锐化模块的参数

【定义】

```
typedef struct rk_aiq_sharp_attrib_s {
    AsharpOPMode_t eMode;
    Asharp_Auto_Attr_t stAuto;
    Asharp_Manual_Attr_t stManual;
} rk_aiq_sharp_attrib_t;
```

【成员】

| 成员名称 | 描述 |
|----------|------------|
| eMode | 锐化模块属性 |
| stAuto | 锐化模块自动模式参数 |
| stManual | 锐化模块手动模式参数 |

AsharpOPMode_t

【说明】

定义锐化模块的模式

【定义】

```

typedef enum AsharpOPMode_e {
    ASHARP_OP_MODE_INVALID          = 0,
    ASHARP_OP_MODE_AUTO             = 1,
    ASHARP_OP_MODE_MANUAL           = 2,
    ASHARP_OP_MODE_MAX
} AsharpOPMode_t;

```

【成员】

| 成员名称 | 描述 |
|------------------------|-------------------|
| ASHARP_OP_MODE_INVALID | 锐化模块无效模式 |
| ASHARP_OP_MODE_AUTO | 锐化模块自动模式 |
| ASHARP_OP_MODE_MANUAL | 锐化模块手动模式 |
| ASHARP_OP_MODE_MAX | 锐化模块模式最大值，是一个无效模式 |

Asharp_Auto_Attr_t

【说明】

定义锐化模块的自动属性

【定义】

```

typedef struct Asharp_Auto_Attr_s
{
    //sharp
    int sharpEn;
    RKSharp_Sharp_Parms_t stSharpParam;
    RKSharp_Sharp_Parms_Select_t stSharpParamSelect;

    //edgefilter
    int edgeFltEn;
    RKSharp_EdgeFilter_Parms_t stEdgefilterParams;
    RKSharp_EdgeFilter_Parms_Select_t stEdgefilterParamSelect;
} Asharp_Auto_Attr_t;

```

【成员】

| 成员名称 | 描述 |
|-------------------------|-----------------------------|
| sharpEn | sharp模块使能位 |
| stSharpParam | sharp模块各个iso对应算法属性参数 |
| stSharpParamSelect | sharp模块根据当前iso计算出来属性参数 |
| edgeFltEn | edgefilter模块使能位 |
| stEdgefilterParams | edgefilter模块各个iso对应算法属性参数 |
| stEdgefilterParamSelect | edgefilter模块根据当前iso计算出来属性参数 |

Asharp_Manual_Attr_t

【说明】

定义锐化模块的手动属性

【定义】

```
typedef struct Asharp_Manual_Attr_s
{
    int sharpEn;
    RKASharp_Sharp_Params_Select_t stSharpParamSelect;

    int edgeFltEn;
    RKASharp_EdgeFilter_Params_Select_t stEdgefilterParamSelect;

} Asharp_Manual_Attr_t;
```

【成员】

| 成员名称 | 描述 |
|-------------------------|--------------------|
| sharpEn | sharp模块使能位 |
| stSharpParamSelect | sharp手动设置算法参数 |
| edgeFltEn | edgefilter模块使能位 |
| stEdgefilterParamSelect | edgefilter手动设置算法参数 |

rk_aiq_sharp_IQpara_t

【说明】

定义去噪模块的IQ参数结构体

【定义】

```
typedef struct rk_aiq_sharp_IQpara_s{
    int module_bits;
    calibDb_Sharp_t stSharpPara;
    calibDb_EdgeFilter_t stEdgeFltPara;
}rk_aiq_sharp_IQpara_t;
```

【成员】

| 成员名称 | 描述 |
|---------------|--------------------------------|
| module_bits | 锐化2个模块对应标志位，根据标志位来分别设置各个模块IQ参数 |
| stSharpPara | sharp模块对应IQ参数 |
| stEdgeFltPara | edgefilter模块对应IQ参数 |

rk_aiq_sharp_module_t

【说明】

定义锐化模块的设置IQ参数对应模块对应的标志位，各个模块可分开设置，也可统一设置，靠这个标志位识别。

【定义】

```
typedef enum rk_aiq_sharp_module_e{
    ASHARP_MODULE_SHARP      = 0,
    ASHARP_MODULE_EDGEFILTER = 1,
} rk_aiq_sharp_module_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------------|-------------------------|
| ASHARP_MODULE_SHARP | sharp模块对应标志位为bit 0 |
| ASHARP_MODULE_EDGEFILTER | edgefilter模块对应标志位为bit 1 |

CalibDb_Sharp_t

【说明】

bayernr模块IQ参数结构体

【定义】

```
typedef struct CalibDb_Sharp_s {
    int enable;
    char version[64];
    float luma_point[8];
    calibDb_Sharp_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];
} CalibDb_Sharp_t;
```

【成员】

| 成员名称 | 描述 |
|------------|----------------------------------|
| enable | sharp模块使能位 |
| version | 硬件参数版本号 |
| luma_point | 和像素亮度相关的分区 |
| mode_cell | 不同模式，如normal, hdr. gray模式对应去噪参数。 |

CalibDb_Sharp_ModeCell_t

【说明】

sharp模块IQ参数结构体

【定义】

```

typedef struct CalibDb_Sharp_ModeCell_s {
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float gauss_luma_coeff[9];
    float mbf_coeff[221];

    //v2
    float pbf_coeff_l[9];
    float pbf_coeff_h[9];
    float rf_m_coeff_l[25];
    float rf_m_coeff_h[25];
    float rf_h_coeff_l[25];
    float rf_h_coeff_h[25];
    float hbf_coeff_l[9];
    float hbf_coeff_h[9];

    CalibDb_Sharp_Setting_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
} CalibDb_Sharp_ModeCell_t;

```

【成员】

| 成员名称 | 描述 |
|------------------|---------------------------------------|
| name | bayernr模块不同模式名称，normal, hdr, gray三种模式 |
| setting | bayernr一个模块下，还有lcg, hcg等对应2种配置。 |
| gauss_luma_coeff | 高斯算子系数3x3，全为0则由使用代码内部定义的系数矩阵。 |
| mbf_coeff | 中频滤波算子系数17x13，全为0则由使用代码内部定义的系数矩阵。 |
| pbf_coeff_l | 预滤波高斯算子系数3x3，全为0则由使用代码内部定义的系数矩阵。 |
| pbf_coeff_h | 预滤波高斯算子系数3x3，全为0则由使用代码内部定义的系数矩阵。 |
| rf_m_coeff_l | 中频滤波算子系数5x5，全为0则由使用代码内部定义的系数矩阵。 |
| rf_m_coeff_h | 中频滤波算子系数5x5，全为0则由使用代码内部定义的系数矩阵。 |
| rf_h_coeff_l | 高频滤波算子系数5x5，全为0则由使用代码内部定义的系数矩阵。 |
| rf_h_coeff_h | 高频滤波算子系数5x5，全为0则由使用代码内部定义的系数矩阵。 |
| hbf_coeff_l | 高频滤波算子系数3x3，全为0则由使用代码内部定义的系数矩阵。 |
| hbf_coeff_h | 高频滤波算子系数3x3，全为0则由使用代码内部定义的系数矩阵。 |

CalibDb_Sharp_Setting_t

【说明】

sharp模块IQ参数结构体

【定义】

```

typedef struct CalibDb_Sharp_Setting_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    struct CalibDb_Sharp_ISO_s sharp_iso[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
} CalibDb_Sharp_Setting_t;

```

【成员】

| 成员名称 | 描述 |
|-------------|---|
| snr_mode | mfnr模块下，有HSNR和LSNR，对应着hcg和lcg等对应2种配置。 |
| sensor_mode | sensor模式lcg, hcg等对应2种配置。如未有dcg模式，则默认使用lcg对应的参数。 |
| sharp_iso | 不同iso下，对应不同sharp参数。 |

CalibDb_Sharp_ISO_s

【说明】

定义sharp模块IQ参数。

【定义】

```

struct CalibDb_Sharp_ISO_s {
    float iso;
    float hratio;
    float lratio;
    float mf_sharp_ratio;
    float hf_sharp_ratio;
    float luma_sigma[8];
    float pbf_gain;
    float pbf_ratio;
    float pbf_add;
    float mf_clip_pos[8];
    float mf_clip_neg[8];
    float hf_clip[8];
    float mbf_gain;
    float hbf_gain;
    float hbf_ratio;
    float mbf_add;
    float hbf_add;
    float local_sharp_strength;
    float pbf_coeff_percent;
    float rf_m_coeff_percent;
    float rf_h_coeff_percent;
    float hbf_coeff_percent;
};

```

【成员】

| 成员名称 | 描述 |
|----------------------|---|
| iso | 不同iso对应的参数配置 |
| hratio | range filter clip 的上限 ratio, 1~2, 值越大, range filter clip 的范围越大, 黑边重。 |
| lratio | range filter clip 的下限 ratio, 0~1, 值越小, range filter clip 的范围越大, 白边越重。 |
| mf_sharp_ratio | 中频锐化强度, 0~8, 值越大, 中频锐化强度越大。 |
| hf_sharp_ratio | 高频锐化强度, 0~8, 值越大, 高频锐化强度越大。 |
| luma_sigma | 与luma_point分区一一对应。预滤波sigma 曲线y轴, 0~255。 |
| pbf_gain | 预滤波 sigma 乘以的比例, 0~2, 值越大, 预滤波 sigma 越大。 |
| pbf_ratio | 预滤波融合权重, 0~1, 值越大, 预滤波强度越大 |
| pbf_add | 预滤波 sigma 叠加的偏移, 0~255, 值越大, 预滤波 sigma 越大。 |
| mf_clip_pos | range filter 计算中频时 clip 的最大值, 0~255。值越大, 白边越重。 |
| mf_clip_neg | range filter 计算中频时 clip 的最小值, -2~0。值越小, 黑边越重。 |
| hf_clip | range filter 计算高频时 clip 的范围, 0~255。值越大, 高频锐化越大。 |
| mbf_gain | 中频双边滤波 sigma 乘以的比例, 0~2, 值越大, 中频滤波 sigma 越大。 |
| hbf_gain | 高频双边滤波 sigma 乘以的比例, 0~2, 值越大, 高滤波 sigma 越大。 |
| hbf_ratio | 高频双边滤波融合权重, 0~1, 值越大, 高频双边滤波融合权重越大。 |
| mbf_add | 中频双边滤波 sigma 叠加的偏移, 0~255, 值越大, 中频滤波 sigma 越大。 |
| hbf_add | 高频双边滤波 sigma 叠加的偏移, 0~255, 值越大, 高频滤波 sigma 越大。 |
| local_sharp_strength | 计算高频叠加权重的比例, 0~255, 值越大, 高频叠加权重的比例越大 |
| pbf_coeff_percent | 预滤波两个高斯算子插值比例。 |
| rf_m_coeff_percent | 中频滤波两个算子插值比例。 |
| rf_h_coeff_percent | 高频滤波两个算子插值比例。 |
| hbf_coeff_percent | 高频滤波两个算子插值比例。 |

CalibDb_EdgeFilter_t

【说明】

edgefilter模块IQ参数结构体

【定义】

```

typedef struct CalibDb_EdgeFilter_s {
    int enable;
    char version[64];
    float luma_point[8];
    CalibDb_EdgeFilter_ModeCell_t mode_cell[CALIBDB_MAX_MODE_NUM];
} CalibDb_EdgeFilter_t;

```

【成员】

| 成员名称 | 描述 |
|------------|-----------------------------------|
| enable | edgefilter模块使能位 |
| version | 硬件参数版本号 |
| luma_point | 和像素亮度相关的分区 |
| mode_cell | 不同模式, 如normal, hdr, gray模式对应去噪参数。 |

CalibDb_EdgeFilter_ModeCell_t

【说明】

edgefilter模块IQ参数结构体

【定义】

```

typedef struct CalibDb_EdgeFilter_ModeCell_s {
    char name[CALIBDB_MAX_MODE_NAME_LENGTH];
    float dog_kernel_l[25];
    float dog_kernel_h[25];
    CalibDb_EdgeFilter_Setting_t setting[CALIBDB_NR_SHARP_SETTING_LEVEL];
} CalibDb_EdgeFilter_ModeCell_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|---|
| name | edgefilter模块不同模式名称, normal, hdr, gray三种模式 |
| setting | edgefilter一个模块下, 还有lcg, hcg等对应2种配置。 |
| dog_kernel_l | dog算子系数5x5, 全为0则由使用代码内部定义的系数矩阵。 |
| dog_kernel_h | dog算子系数5x5, 全为0则由使用代码内部定义的系数矩阵。 |

CalibDb_EdgeFilter_Setting_t

【说明】

edgefilter模块IQ参数结构体

【定义】

```

typedef struct CalibDb_EdgeFilter_Setting_s {
    char snr_mode[CALIBDB_NR_SHARP_NAME_LENGTH];
    char sensor_mode[CALIBDB_NR_SHARP_MODE_LENGTH];
    struct CalibDb_EdgeFilter_ISO_s
edgeFilter_iso[CALIBDB_NR_SHARP_MAX_ISO_LEVEL];
} CalibDb_EdgeFilter_Setting_t;

```

【成员】

| 成员名称 | 描述 |
|----------------|---|
| snr_mode | edgefilter模块下，有HSNR和LSNR，对应着hcg和lcg等对应2种配置。 |
| sensor_mode | sensor模式lcg, hcg等对应2种配置。如未有dcg模式，则默认使用lcg对应的参数。 |
| edgeFilter_iso | 不同iso下，对应不同edgefilter参数。 |

CalibDb_EdgeFilter_ISO_s

【说明】

定义edgefilter模块IQ参数。

【定义】

```

struct CalibDb_EdgeFilter_ISO_s {
    float iso;
    float edge_thed;
    float src_wgt;
    unsigned char alpha_adp_en;
    float local_alpha;
    float global_alpha;
    float noise_clip[8];
    float dog_clip_pos[8];
    float dog_clip_neg[8];
    float dog_alpha[8];
    float direct_filter_coeff[5];
    float dog_kernel_row0[5];
    float dog_kernel_row1[5];
    float dog_kernel_row2[5];
    float dog_kernel_row3[5];
    float dog_kernel_row4[5];
    float dog_kernel_percent;
};

```

【成员】

| 成员名称 | 描述 |
|---------------------|--|
| iso | 不同iso对应的参数配置 |
| edge_thed | 梯度 clip 的最小值， 1 ~255， 值越大， 边缘滤波效果越弱。 |
| src_wgt | 融合时叠加原图像的权重， 0~1， 值越大， 边缘滤波效果越弱。 |
| alpha_adp_en | 自适应融合开关。0：关闭，1：开启。 |
| local_alpha | 自适应融合关闭时，全局融合的权重，0~1，值越大，边缘滤波效果越强。 |
| global_alpha | 边缘滤波前后图像融合权重，0~1，值越大，边缘滤波效果越强。 |
| noise_clip | 边缘滤波 clip 曲线，0 ~255，值越大，边缘滤波效果越强。 |
| dog_clip_pos | dog clip 最大值，0 ~255，值越大，dog clip 的范围越大，叠加高频越多。 |
| dog_clip_neg | dog clip 最大值，0 ~255，值越大，dog clip 的范围越大，叠加高频越多。 |
| dog_alpha | dog 融合权重，0 ~3，值越大，叠加高频越多。 |
| direct_filter_coeff | 边缘滤波水平方向系数。 |
| dog_kernel_row0 | Dog滤波算子5x5，第一行系数。 |
| dog_kernel_row1 | Dog滤波算子5x5，第二行系数。 |
| dog_kernel_row2 | Dog滤波算子5x5，第三行系数。 |
| dog_kernel_row3 | Dog滤波算子5x5，第四行系数。 |
| dog_kernel_row4 | Dog滤波算子5x5，第五行系数。 |
| dog_kernel_percent | DOG算子插值比例。 |

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。

功能级API参考

rk_aiq_uapi_setGammaCoef

【描述】

设置伽玛。

【语法】

```
XCamReturn rk_aiq_uapi_setGammaCoef(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_gamma_attrib_t gammaAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------|--------------|-------|
| ctx | AIQ上下文指针 | 输入 |
| gammaAttr | Gamma软件属性结构体 | 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_api_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

rk_aiq_gamma_op_mode_t

【说明】

定义Gamma工作模式

【定义】

```
typedef enum rk_aiq_gamma_op_mode_s {
    RK_AIQ_GAMMA_MODE_OFF          = 0,
    RK_AIQ_GAMMA_MODE_MANUAL        = 1,
    RK_AIQ_GAMMA_MODE_TOOL          = 2,
} rk_aiq_gamma_op_mode_t;
```

【成员】

| 成员名称 | 描述 |
|--------------------------|---------|
| RK_AIQ_GAMMA_MODE_OFF | Api关闭模式 |
| RK_AIQ_GAMMA_MODE_MANUAL | Api自动模式 |
| RK_AIQ_GAMMA_MODE_TOOL | Api工具模式 |

rk_gamma_curve_type_t

【说明】

定义手动模式下Gamma曲线工作模式

【定义】

```

typedef enum rk_gamma_curve_type_s {
    RK_GAMMA_CURVE_TYPE_DEFUALT          = 0,
    RK_GAMMA_CURVE_TYPE_SRGB             = 1,
    RK_GAMMA_CURVE_TYPE_HDR              = 2,
    RK_GAMMA_CURVE_TYPE_USER_DEFINE1     = 3,
    RK_GAMMA_CURVE_TYPE_USER_DEFINE2     = 4,
} rk_gamma_curve_type_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------------------|---------------------|
| RK_GAMMA_CURVE_TYPE_DEFUALT | 使用IQ文件中Gamma曲线 |
| RK_GAMMA_CURVE_TYPE_SRGB | 使用sRGB标准Gamma 2.2曲线 |
| RK_GAMMA_CURVE_TYPE_HDR | 使用HDR模式Gamma曲线 |
| RK_GAMMA_CURVE_TYPE_USER_DEFINE1 | 使用用户定义Gamma曲线1 |
| RK_GAMMA_CURVE_TYPE_USER_DEFINE2 | 使用用户定义Gamma曲线2 |

rk_gamma_curve_usr_define1_para_t

【说明】

定义手动模式下用户定义Gamma曲线1属性

【定义】

```

typedef struct rk_gamma_curve_usr_define1_para_s {
    float coef1;
    float coef2;
} rk_gamma_curve_usr_define1_para_t;

```

【成员】

| 成员名称 | 描述 |
|-------|---------------|
| coef1 | 控制Gamma曲线形状 |
| coef2 | 控制Gamma曲线零点斜率 |

rk_gamma_curve_usr_define2_para_t

【说明】

定义手动模式下用户定义Gamma曲线2属性

【定义】

```

typedef struct rk_gamma_curve_usr_define2_para_s {
    int gamma_out_segnun;
    int gamma_out_offset;
    int gamma_table[45];
} rk_gamma_curve_usr_define2_para_t;

```

【成员】

| 成员名称 | 描述 |
|------------------|--------------------------------|
| gamma_out_segnum | 定义Gamma曲线X轴间距, 0: 非等间距, 1: 等间距 |
| gamma_out_offset | Gamma曲线偏移值 |
| gamma_table | Gamma曲线 |

Agamma_api_manual_t

【说明】

定义手动Gamma属性

【定义】

```
typedef struct Agamma_api_manual_s {
    bool en;
    rk_gamma_curve_type_t CurveType;
    rk_gamma_curve_usr_define1_para_t user1;
    rk_gamma_curve_usr_define2_para_t user2;
} Agamma_api_manual_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|--------------|
| en | 开关功能 |
| CurveType | 曲线种类 |
| user1 | 用户定义Gamma曲线1 |
| user2 | 用户定义Gamma曲线2 |

CalibDb_Gamma_t

【说明】

定义工具模式下Gamma属性

【定义】

```
typedef struct CalibDb_Gamma_s {
    unsigned char gamma_en;
    unsigned char gamma_out_segnum;
    unsigned char gamma_out_offset;
    float curve_normal[45];
    float curve_hdr[45];
    float curve_night[45];
} CalibDb_Gamma_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|--------------------------------|
| gamma_en | 开关功能 |
| gamma_out_segnum | 定义Gamma曲线X轴间距, 0: 非等间距, 1: 等间距 |
| gamma_out_offset | Gamma曲线偏移值 |
| curve_normal | 线性模式下Gamma曲线 |
| curve_hdr | HDR模式下Gamma曲线 |
| curve_night | 夜视模式下Gamma曲线 |

rk_aiq_gamma_attr_t

【说明】

定义Gamma属性

【定义】

```
typedef struct rk_aiq_gamma_attr_s {
    rk_aiq_gamma_op_mode_t mode;
    Agamma_api_manual_t stManual;
    CalibDb_Gamma_t     stTool;
    int                 Scene_mode;
} rk_aiq_gamma_attr_t;
```

【成员】

| 成员名称 | 描述 |
|------------|-----------|
| mode | Api模式 |
| stManual | 手动Gamma参数 |
| stTool | 工具Gamma参数 |
| Scene_mode | 场景模式 |

模块级API参考

rk_aiq_user_api_agamma_SetAttrib

【描述】

设定 Gamma 软件属性。

【语法】

```
XCamReturn
rk_aiq_user_api_agamma_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_gamma_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 | Gamma软件属性结构体 | 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_api_agamma.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。

rk_aiq_user_api_agamma_GetAttrib

【描述】

获取 Gamma软件属性。

【语法】

```
XCamReturn
rk_aiq_user_api_agamma_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_gamma_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 | Gamma软件属性结构体 | 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_api_agamma.h
- 库文件: librkaiq.so

【说明】

DPCC

功能描述

模块级API参考

rk_aiq_user_api_adpcc_SetAttrib

【描述】

设定 DPCC软件属性。

【语法】

```
XCamReturn  
rk_aiq_user_api_adpcc_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
rk_aiq_dpcc_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 | DPCC软件属性结构体 | 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_api_adpcc.h
- 库文件: librkaiq.so

【说明】

rk_aiq_user_api_adpcc_GetAttrib

【描述】

获取 DPCC软件属性。

【语法】

```
XCamReturn  
rk_aiq_user_api_adpcc_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
rk_aiq_dpcc_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 | DPCC软件属性结构体 | 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_api_adpcc.h
- 库文件: librkaiq.so

【说明】

模块级API数据类型

AdpccOPMode_t

【说明】

定义DPCC工作模式

【定义】

```
typedef enum AdpccOPMode_e {
    ADPCC_OP_MODE_INVALID      = 0,
    ADPCC_OP_MODE_AUTO         = 1,
    ADPCC_OP_MODE_MANUAL       = 2,
    ADPCC_OP_MODE_TOOL         = 3,
    ADPCC_OP_MODE_MAX
} AdpccOPMode_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------------|---------|
| ADPCC_OP_MODE_INVALID | Api无效模式 |
| ADPCC_OP_MODE_AUTO | Api自动模式 |
| ADPCC_OP_MODE_MANUAL | Api手动模式 |
| ADPCC_OP_MODE_TOOL | Api工具模式 |
| ADPCC_OP_MODE_MAX | |

Adpcc_basic_params_select_t

【说明】

定义DPCC基本参数属性

【定义】

```
typedef struct Adpcc_basic_params_select_s
{
    int          iso;
    unsigned char stage1_enable;
    unsigned char grayscale_mode;
    unsigned char enable;
    unsigned char sw_rk_out_sel;
    unsigned char sw_dpcc_output_sel;
    unsigned char stage1_rb_3x3;
    unsigned char stage1_g_3x3;
    unsigned char stage1_incl_rb_center;
    unsigned char stage1_incl_green_center;
    unsigned char stage1_use_fix_set;
    unsigned char stage1_use_set_3;
    unsigned char stage1_use_set_2;
    unsigned char stage1_use_set_1;
    unsigned char sw_rk_red_blue1_en;
    unsigned char rg_red_blue1_enable;
    unsigned char rnd_red_blue1_enable;
    unsigned char ro_red_blue1_enable;
    unsigned char lc_red_blue1_enable;
    unsigned char pg_red_blue1_enable;
    unsigned char sw_rk_green1_en;
    unsigned char rg_green1_enable;
    unsigned char rnd_green1_enable;
    unsigned char ro_green1_enable;
    unsigned char lc_green1_enable;
    unsigned char pg_green1_enable;
    unsigned char sw_rk_red_blue2_en;
    unsigned char rg_red_blue2_enable;
    unsigned char rnd_red_blue2_enable;
    unsigned char ro_red_blue2_enable;
    unsigned char lc_red_blue2_enable;
    unsigned char pg_red_blue2_enable;
    unsigned char sw_rk_green2_en;
    unsigned char rg_green2_enable;
    unsigned char rnd_green2_enable;
    unsigned char ro_green2_enable;
    unsigned char lc_green2_enable;
    unsigned char pg_green2_enable;
    unsigned char sw_rk_red_blue3_en;
    unsigned char rg_red_blue3_enable;
    unsigned char rnd_red_blue3_enable;
    unsigned char ro_red_blue3_enable;
    unsigned char lc_red_blue3_enable;
    unsigned char pg_red_blue3_enable;
    unsigned char sw_rk_green3_en;
    unsigned char rg_green3_enable;
    unsigned char rnd_green3_enable;
    unsigned char ro_green3_enable;
    unsigned char lc_green3_enable;
    unsigned char pg_green3_enable;
    unsigned char sw_mindis1_rb;
    unsigned char sw_mindis1_g;
    unsigned char line_thr_1_rb;
```

```
    unsigned char line_thr_1_g;
    unsigned char sw_dis_scale_min1;
    unsigned char sw_dis_scale_max1;
    unsigned char line_mad_fac_1_rb;
    unsigned char line_mad_fac_1_g;
    unsigned char pg_fac_1_rb;
    unsigned char pg_fac_1_g;
    unsigned char rnd_thr_1_rb;
    unsigned char rnd_thr_1_g;
    unsigned char rg_fac_1_rb;
    unsigned char rg_fac_1_g;
    unsigned char sw_mindis2_rb;
    unsigned char sw_mindis2_g;
    unsigned char line_thr_2_rb;
    unsigned char line_thr_2_g;
    unsigned char sw_dis_scale_min2;
    unsigned char sw_dis_scale_max2;
    unsigned char line_mad_fac_2_rb;
    unsigned char line_mad_fac_2_g;
    unsigned char pg_fac_2_rb;
    unsigned char pg_fac_2_g;
    unsigned char rnd_thr_2_rb;
    unsigned char rnd_thr_2_g;
    unsigned char rg_fac_2_rb;
    unsigned char rg_fac_2_g;
    unsigned char sw_mindis3_rb;
    unsigned char sw_mindis3_g;
    unsigned char line_thr_3_rb;
    unsigned char line_thr_3_g;
    unsigned char sw_dis_scale_min3;
    unsigned char sw_dis_scale_max3;
    unsigned char line_mad_fac_3_rb;
    unsigned char line_mad_fac_3_g;
    unsigned char pg_fac_3_rb;
    unsigned char pg_fac_3_g;
    unsigned char rnd_thr_3_rb;
    unsigned char rnd_thr_3_g;
    unsigned char rg_fac_3_rb;
    unsigned char rg_fac_3_g;
    unsigned char ro_lim_3_rb;
    unsigned char ro_lim_3_g;
    unsigned char ro_lim_2_rb;
    unsigned char ro_lim_2_g;
    unsigned char ro_lim_1_rb;
    unsigned char ro_lim_1_g;
    unsigned char rnd_offs_3_rb;
    unsigned char rnd_offs_3_g;
    unsigned char rnd_offs_2_rb;
    unsigned char rnd_offs_2_g;
    unsigned char rnd_offs_1_rb;
    unsigned char rnd_offs_1_g;

} Adpcc_basic_params_select_t;
```

Adpcc_basic_params_t

【说明】

定义DPCC基本参数属性

【定义】

```
typedef struct Adpcc_basic_params_s
{
    Adpcc_basic_params_select_t arBasic[DPCC_MAX_ISO_LEVEL];
} Adpcc_basic_params_t;
```

【成员】

| 成员名称 | 描述 |
|---------|-----------|
| arBasic | DPCC基本参数属 |

Adpcc_bpt_params_t

【说明】

定义自动DPCC属性

【定义】

```
typedef struct Adpcc_bpt_params_s
{
    unsigned char      bpt_rb_3x3;
    unsigned char      bpt_g_3x3;
    unsigned char      bpt_incl_rb_center;
    unsigned char      bpt_incl_green_center;
    unsigned char      bpt_use_fix_set;
    unsigned char      bpt_use_set_3;
    unsigned char      bpt_use_set_2;
    unsigned char      bpt_use_set_1;
    unsigned char      bpt_cor_en;
    unsigned char      bpt_det_en;
    unsigned short int bp_number;
    unsigned short int bp_table_addr;
    unsigned short int bpt_v_addr;
    unsigned short int bpt_h_addr;
    unsigned int        bp_cnt;
} Adpcc_bpt_params_t;
```

dpcc_pdaf_point_t

【说明】

【定义】

```
typedef struct dpcc_pdaf_point_s
{
    unsigned char y;
    unsigned char x;
} dpcc_pdaf_point_t;
```

该模块还未实现

Adpcc_pdaf_params_t

【说明】

定义自动模式下PDAF模式属性

【定义】

```
typedef struct Adpcc_pdaf_params_s
{
    unsigned char      sw_pdaf_en;
    unsigned char      pdaf_point_en[DPCC_PDAF_POINT_NUM];
    unsigned short int pdaf_offsety;
    unsigned short int pdaf_offsetx;
    unsigned char      pdaf_wrappy;
    unsigned char      pdaf_wrapx;
    unsigned short int pdaf_wrappy_num;
    unsigned short int pdaf_wrapx_num;
    dpcc_pdaf_point_t point[DPCC_PDAF_POINT_NUM];
    unsigned char      pdaf_forward_med;
} Adpcc_pdaf_params_t;
```

该模块还未实现

CalibDb_Dpcc_Fast_Mode_t

【说明】

定义自动模式下Fast mode属性

【定义】

```
typedef struct CalibDb_Dpcc_Fast_Mode_s
{
    int fast_mode_en;
    int ISO[CALIBDB_DPCC_MAX_ISO_LEVEL];
    int fast_mode_single_en;
    int fast_mode_single_level[CALIBDB_DPCC_MAX_ISO_LEVEL];
    int fast_mode_double_en;
    int fast_mode_double_level[CALIBDB_DPCC_MAX_ISO_LEVEL];
    int fast_mode_triple_en;
    int fast_mode_triple_level[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Fast_Mode_t;
```

【成员】

| 成员名称 | 描述 |
|------------------------|-----------------------------|
| Fast_mode_enable | Fast_mode开关功能, 0: 关闭, 1: 打开 |
| ISO | 环境ISO |
| fast_mode_single_en | 单坏点去除开关, 0: 关闭, 1: 打开 |
| fast_mode_single_level | 单坏点去除力度, 取值范围[0, 10] |
| fast_mode_double_en | 双坏点去除开关, 0: 关闭, 1: 打开 |
| fast_mode_double_level | 双坏点去除力度, 取值范围[0, 10] |
| fast_mode_triple_en | 多坏点去除开关, 0: 关闭, 1: 打开 |
| fast_mode_triple_level | 多坏点去除力度, 取值范围[0, 10] |

CalibDb_Dpcc_Sensor_t

【说明】

定义自动模式下Fast mode属性

【定义】

```
typedef struct CalibDb_Dpcc_Sensor_s
{
    float en;
    float max_level;
    float iso[CALIBDB_DPCC_MAX_ISO_LEVEL];
    float level_single[CALIBDB_DPCC_MAX_ISO_LEVEL];
    float level_multiple[CALIBDB_DPCC_MAX_ISO_LEVEL];
} calibDb_Dpcc_Sensor_t;
```

【成员】

| 成员名称 | 描述 |
|----------------|-------------------------------|
| en | sensor dpcc开关功能, 0: 关闭, 1: 打开 |
| max_level | 去除坏点最大力度 |
| iso | 环境ISO |
| level_single | 去除单个坏点力度 |
| level_multiple | 去除多个坏点力度 |

Adpcc_bpt_params_select_t

【说明】

定义自动模式下选择的Fast mode属性

【定义】

```
typedef Adpcc_bpt_params_t Adpcc_bpt_params_select_t;
```

Adpcc_pdaf_params_select_t

【说明】

定义自动模式下选择的PDAF模式属性

【定义】

```
typedef Adpcc_pdaf_params_t Adpcc_pdaf_params_select_t
```

Adpcc_Auto_Attr_t

【说明】

定义自动DPCC属性

【定义】

```
typedef struct Adpcc_Auto_Attr_s
{
    Adpcc_basic_params_t      stBasicParams;
    Adpcc_bpt_params_t        stBptParams;
    Adpcc_pdaf_params_t       stPdafParams;
    CalibDb_Dpcc_Fast_Mode_t stFastMode;
    CalibDb_Dpcc_Sensor_t    stSensorDpcc;
    Adpcc_basic_params_select_t stBasicSelect;
    Adpcc_bpt_params_select_t stBptSelect;
    Adpcc_pdaf_params_select_t stPdafSelect;

} Adpcc_Auto_Attr_t;
```

【成员】

| 成员名称 | 描述 |
|---------------|-------------------|
| stBasicParams | 自动模式下基本参数 |
| stBptParams | 自动模式下坏点参数 |
| stPdafParams | 自动模式下PDAF模式参数 |
| stFastMode | 自动模式下快速模式参数 |
| stSensorDpcc | 自动模式下Sensor坏点功能参数 |
| stBasicSelect | 自动模式下选择的基本参数 |
| stBptSelect | 自动模式下选择的坏点参数 |
| stPdafSelect | 自动模式下选择的PDAF模式参数 |

Adpcc_fast_mode_attr_t

【说明】

定义手动模式下快速模式属性

【定义】

```

typedef struct Adpcc_fast_mode_attr_s
{
    bool fast_mode_en;
    bool fast_mode_single_en;
    int fast_mode_single_level;
    bool fast_mode_double_en;
    int fast_mode_double_level;
    bool fast_mode_triple_en;
    int fast_mode_triple_level;
} Adpcc_fast_mode_attr_t;

```

【成员】

| 成员名称 | 描述 |
|------------------------|----------------------|
| Fast_mode_en | Fast_mode开关功能 |
| fast_mode_single_en | 单坏点去除开关 |
| fast_mode_single_level | 单坏点去除力度, 取值范围[0, 10] |
| fast_mode_double_en | 双坏点去除开关 |
| fast_mode_double_level | 双坏点去除力度, 取值范围[0, 10] |
| fast_mode_triple_en | 多坏点去除开关 |
| fast_mode_triple_level | 多坏点去除力度, 取值范围[0, 10] |

Adpcc_sensor_dpcc_attr_t

【说明】

定义手动模式下Sensor坏点功能属性

【定义】

```

typedef struct Adpcc_sensor_dpcc_attr_s
{
    bool en;
    int max_level;
    int single_level;
    int double_level;
} Adpcc_sensor_dpcc_attr_t;

```

【成员】

| 成员名称 | 描述 |
|--------------|-----------------|
| en | sensor dpcc开关功能 |
| max_level | 去除坏点最大力度 |
| single_level | 去除单个坏点力度 |
| double_level | 去除多个坏点力度 |

Adpcc_Manual_Attr_t

【说明】

定义手动DPCC属性

【定义】

```
typedef struct Adpcc_Manual_Attr_s
{
    Adpcc_basic_params_select_t stBasic;
    Adpcc_bpt_params_select_t   stBpt;
    Adpcc_pdaf_params_select_t  stPdaf;
    Adpcc_fast_mode_attr_t     stFastMode;
    Adpcc_sensor_dpcc_attr_t   stSensorDpcc;
} Adpcc_Manual_Attr_t;
```

【成员】

| 成员名称 | 描述 |
|---------------|-------------------|
| stBasicParams | 手动模式下基本参数 |
| stBptParams | 手动模式下坏点参数 |
| stPdafParams | 手动模式下PDAF模式参数 |
| stFastMode | 手动模式下快速模式参数 |
| stSensorDpcc | 手动模式下Sensor坏点功能参数 |

CalibDb_Dpcc_Pdaf_t

【说明】

定义工具PDAF模式属性

【定义】

```
typedef struct CalibDb_Dpcc_Pdaf_s
{
    unsigned char      en;
    unsigned char      point_en[16];
    unsigned short int offsetx;
    unsigned short int offsety;
    unsigned char      wrapx;
    unsigned char      wraphy;
    unsigned short int wrapx_num;
    unsigned short int wraphy_num;
    unsigned char      point_x[16];
    unsigned char      point_y[16];
    unsigned char      forward_med;
} CalibDb_Dpcc_Pdaf_t;
```

CalibDb_Dpcc_set_RK_t

【说明】

定义RK算法属性

【定义】

```

typedef struct CalibDb_Dpcc_Set_RK_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_sw_mindis[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_sw_mindis[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char sw_dis_scale_min[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char sw_dis_scale_max[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_RK_t;

```

【成员】

| 成员名称 | 描述 |
|------------------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_sw_mindis | 红、蓝通道坏点阈值1 |
| g_sw_mindis | 绿通道坏点阈值1 |
| sw_dis_scale_min | 坏点阈值2 |
| sw_dis_scale_max | 坏点阈值3 |

CalibDb_Dpcc_Set_LC_t

【说明】

定义LC算法属性

【定义】

```

typedef struct CalibDb_Dpcc_Set_LC_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_line_thr[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_line_thr[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_line_mad_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_line_mad_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_LC_t;

```

【成员】

| 成员名称 | 描述 |
|-----------------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_line_thr | 红、蓝通道坏点阈值1 |
| g_line_thr | 绿通道坏点阈值1 |
| rb_line_mad_fac | 红、蓝通道坏点阈值2 |
| g_line_mad_fac | 绿通道坏点阈值2 |

CalibDb_Dpcc_set_PG_t

【说明】

定义PG算法属性

【定义】

```
typedef struct CalibDb_Dpcc_Set_PG_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_pg_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_pg_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_PG_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_pg_fac | 红、蓝通道坏点阈值 |
| g_pg_fac | 绿通道坏点阈值 |

CalibDb_Dpcc_Set_RND_t

【说明】

定义RND算法属性

【定义】

```
typedef struct CalibDb_Dpcc_Set_RND_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_rnd_thr[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_rnd_thr[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_rnd_offs[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_rnd_offs[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_RND_t;
```

【成员】

| 成员名称 | 描述 |
|-------------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_rnd_thr | 红、蓝通道坏点阈值1 |
| g_rnd_thr | 绿通道坏点阈值1 |
| rb_rnd_offs | 红、蓝通道坏点阈值2 |
| g_rnd_offs | 绿通道坏点阈值2 |

CalibDb_Dpcc_set_RG_t

【说明】

定义RK算法属性

【定义】

```
typedef struct CalibDb_Dpcc_Set_RG_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_rg_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_rg_fac[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_RG_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_rg_fac | 红、蓝通道坏点阈值 |
| g_rg_fac | 绿通道坏点阈值 |

CalibDb_Dpcc_Set_RO_t

【说明】

定义RO算法属性

【定义】

```
typedef struct CalibDb_Dpcc_Set_RO_s
{
    unsigned char rb_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char rb_ro_lim[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char g_ro_lim[CALIBDB_DPCC_MAX_ISO_LEVEL];
} CalibDb_Dpcc_Set_RO_t;
```

【成员】

| 成员名称 | 描述 |
|-----------|-------------|
| rb_enable | 红、蓝通道坏点检测开关 |
| g_enable | 绿通道坏点检测开关 |
| rb_ro_lim | 红、蓝通道坏点阈值 |
| g_ro_lim | 绿通道坏点阈值 |

CalibDb_Dpcc_Set_t

【说明】

定义坏点判断条件属性

【定义】

```
typedef struct CalibDb_Dpcc_Set_s
{
    CalibDb_Dpcc_Set_RK_t rk;
    CalibDb_Dpcc_Set_LC_t lc;
    CalibDb_Dpcc_Set_PG_t pg;
    CalibDb_Dpcc_Set_RND_t rnd;
    CalibDb_Dpcc_Set_RG_t rg;
    CalibDb_Dpcc_Set_RO_t ro;
} CalibDb_Dpcc_Set_t;
```

【成员】

| 成员名称 | 描述 |
|------|-------|
| rk | RK算法 |
| lc | LC算法 |
| pg | PG算法 |
| rnd | RND算法 |
| rg | RG算法 |
| ro | RO算法 |

CalibDb_Dpcc_Expert_Mode_t

【说明】

定义工具专家模式属性

【定义】

```
typedef struct CalibDb_Dpcc_Expert_Mode_s
{
    float iso[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char stage1_Enable[CALIBDB_DPCC_MAX_ISO_LEVEL];
    unsigned char grayscale_mode;
    unsigned char rk_out_sel[CALIBDB_DPCC_MAX_ISO_LEVEL];
```

```

unsigned char dpcc_out_sel[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_rb_3x3[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_g_3x3[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_inc_rb_center[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_inc_g_center[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_use_fix_set[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_use_set3[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_use_set2[CALIBDB_DPCC_MAX_ISO_LEVEL];
unsigned char stage1_use_set1[CALIBDB_DPCC_MAX_ISO_LEVEL];
calibDb_Dpcc_Set_t set[3];
} CalibDb_Dpcc_Expert_Mode_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------|------------------------------------|
| iso | 环境ISO |
| stage1_Enable | 默认值1 |
| grayscale_mode | 黑白模式开关, 0: 关闭, 1: 打开 |
| rk_out_sel | RK坏点判断模式, 0: 模式1, 1: 模式2, 2: 模式3 |
| dpcc_out_sel | 坏点矫正模式, 0: 中值, 1: RK模式 |
| stage1_rb_3x3 | 默认值0 |
| stage1_g_3x3 | 默认值0 |
| stage1_inc_rb_center | 默认值1 |
| stage1_inc_g_center | 默认值1 |
| stage1_use_fix_set | 内置坏点判定条件开关, 0: 关闭, 1: 打开 |
| stage1_use_set3 | set_cell中第三种坏点判断条件开关, 0: 关闭, 1: 打开 |
| stage1_use_set2 | set_cell中第二种坏点判断条件开关, 0: 关闭, 1: 打开 |
| stage1_use_set1 | set_cell中第一种坏点判断条件开关, 0: 关闭, 1: 打开 |
| set | 坏点判断条件 |

CalibDb_Dpcc_t

【说明】

定义工具DPCC属性

【定义】

```

typedef struct CalibDb_Dpcc_s
{
    int enable;
    char version[64];
    CalibDb_Dpcc_Fast_Mode_t fast;
    CalibDb_Dpcc_Expert_Mode_t expert;
    CalibDb_Dpcc_Pdaf_t pdaf;
    CalibDb_Dpcc_Sensor_t sensor_dpcc;
} CalibDb_Dpcc_t;

```

【成员】

| 成员名称 | 描述 |
|-------------|----------------|
| enable | 开关功能 |
| version | 版本 |
| fast | 快速模式 |
| expert | 专家模式 |
| pdaf | PADF模式下坏点条件 |
| sensor_dpcc | Sensor自带坏点矫正设置 |

rk_aiq_dpcc_attrib_t

【说明】

定义DPCC属性

【定义】

```

typedef struct rk_aiq_dpcc_attrib_s
{
    AdpccOPMode_t eMode;
    Adpcc_Auto_Attr_t stAuto;
    Adpcc_Manual_Attr_t stManual;
    CalibDb_Dpcc_t stTool;
} rk_aiq_dpcc_attrib_t;

```

【成员】

| 成员名称 | 描述 |
|----------|----------|
| eMode | api模式 |
| stAuto | 自动DPCC模式 |
| stManual | 手动DPCC模式 |
| stTool | 工具DPCC模式 |

ASD

模块级API参考

rk_aiq_user_api_asd_GetAttrib

【描述】

获取当前环境亮度的计算结果。

【语法】

```
XCamReturn rk_aiq_user_api_asd_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,  
asd_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 | 计算结果 | 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_api_asd.h
- 库文件: librkaiq.so

数据类型

asd_attrib_t

【说明】

当前环境亮度的计算结果

【定义】

```
typedef struct asd_attrib_s {  
    float cur_m2r;  
} asd_attrib_t;
```

【成员】

| 成员名称 | 描述 |
|---------|---|
| cur_m2r | 当前平均亮度 计算方法： $\text{exp_val_ratio} = \text{cur_exp_val} / \text{max_exp_val}$ $\text{cur_m2r} = \text{mean_luma} / \text{exp_val_ratio}$ |

Demosaic

功能描述

去马赛克主要指将输入的 Bayer 数据转化成 RGB 域数据。

模块级API参考

rk_aiq_user_api_adebayer_SetAttrib

【描述】

设置去马赛克属性。

【语法】

```
XCamReturn
rk_aiq_user_api_adebayer_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
adebayer_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 | 去马赛克属性 | 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_api_adebayer.h、rk_aiq_uapi_adebayer_int.h
- 库文件: librkaiq.so

rk_aiq_user_api_adebayer_GetAttrib

【描述】

获取去马赛克属性。

【语法】

```
XCamReturn
rk_aiq_user_api_adebayer_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
adebayer_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 | 去马赛克属性 | 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_api_adebayer.h、rk_aiq_uapi_adebayer_int.h
- 库文件: librkaiq.so

数据类型

adebayer_attrib_t

【说明】

定义ISP去马赛克属性。

【定义】

```
typedef struct adebayer_attrib_s {
    unsigned char enable;
    unsigned char enhance_strength[9];
    unsigned char low_freq_thresh;
    unsigned char high_freq_thresh;
} adebayer_attrib_t;
```

【成员】

| 成员名称 | 描述 |
|---------------------|--|
| enable | Demosaic模块使能 0: 关闭 1: 使能 |
| enhance_strength[9] | 不同ISO下的细节纹理增强强度 index 0 - ISO 50 index 1 - ISO 100 index 2 - ISO 200 index 3 - ISO 400 index 4 - ISO 800 index 5 - ISO 1600 index 6 - ISO 3200 index 7 - ISO 6400 index 8 - ISO 12800 值越大细节细碎度和清晰度越好，同时伪细节也会相应增强 |
| low_freq_thresh | 低频权重选取阈值 值越大选取低频权重的概率越小 |
| high_freq_thresh | 高频权重选取阈值 值越大选取高频权重的概率越小 |

##

其他

API参考

rk_aiq_uapi_setGrayMode

【描述】

设置黑白图像模式的工作方式。

【语法】

```
XCamReturn rk_aiq_uapi_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 | 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 |

【需求】

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

rk_aiq_uapi_getGrayMode

【描述】

设置黑白图像模式的工作方式。

【语法】

```
rk_aiq_gray_mode_t rk_aiq_uapi_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 |

【返回值】

| 返回值 | 描述 |
|--------------------|------|
| rk_aiq_gray_mode_t | 工作模式 |

【需求】

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

rk_aiq_uapi_setFrameRate

【描述】

设置图像输出帧率。

【语法】

```
XCamReturn rk_aiq_uapi_setFrameRate(const rk_aiq_sys_ctx_t* ctx, frameRateInfo_t info);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| frameRateInfo_t | 帧率信息结构体 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getFrameRate

【描述】

获取图像输出帧率信息。

【语法】

```
XCamReturn rk_aiq_uapi_getFrameRate(const rk_aiq_sys_ctx_t* ctx, frameRateInfo_t* info);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|-----------------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| frameRateInfo_t | 帧率信息结构体 | 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_api_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_setMirroFlip

【描述】

设置图像镜像、翻转。

【语法】

```
XCamReturn rk_aiq_uapi_setMirroFlip(const rk_aiq_sys_ctx_t* ctx, bool mirror,  
bool flip);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| mirror | 是否镜像 | 输入 |
| flip | 是否翻转 | 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_imgproc.h
- 库文件: librkaiq.so

rk_aiq_uapi_getMirroFlip

【描述】

获取图像镜像、翻转信息。

【语法】

```
XCamReturn rk_aiq_uapi_getMirrorFlip(const rk_aiq_sys_ctx_t* ctx, bool* mirror, bool* flip);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|----------|-------|
| ctx | AIQ上下文指针 | 输入 |
| mirror | 是否镜像 | 输出 |
| flip | 是否翻转 | 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_api_imgproc.h
- 库文件: librkaiq.so

数据类型

rk_aiq_gray_mode_t

【说明】

黑白切换工作模式

【定义】

```
typedef enum rk_aiq_gray_mode_e {
    RK_AIQ_GRAY_MODE_CPSL,
    RK_AIQ_GRAY_MODE_OFF,
    RK_AIQ_GRAY_MODE_ON,
} rk_aiq_gray_mode_t;
```

【成员】

| 成员名称 | 描述 |
|-----------------------|-----------|
| RK_AIQ_GRAY_MODE_CPSL | 由cpsl算法控制 |
| RK_AIQ_GRAY_MODE_OFF | 关闭黑白模式 |
| RK_AIQ_GRAY_MODE_ON | 开启黑白模式 |

统计信息

概述

ISP提供的3A统计信息以及相关配置

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图的256段带权重直方图统计信息、基于raw图的分块R/G/B/Y 均值统计信息；基于gamma前RGB图的32段带权重直方图统计信息、基于gamma前RGB图的分块R/G/B/Y 均值统计信息。

基于raw图的AE统计

- 该模块统计分为分块亮度统计和直方图统计。根据支持的分块大小和是否含有子窗口统计，统计模式又可分为big模式、lite模式。
- big模式：最大支持全局15X15分块，最小支持1X1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；在全局分块的基础上，支持独立设置4个子窗口，每个子窗口均可输出29bit R/B通道亮度总和和32bit G通道总和，亮度均值需要在软件中除以每个子窗口的像素数求得。该模式下的带权重直方图统计，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。
- lite模式：最大支持5X5分块，最小支持1X1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5X5分块；不支持独立设置子窗口。该模式下的带权重直方图统计，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。

基于RGB图的AE统计

- 该模块统计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分块，最小支持1x1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；在全局分块的基础上，支持独立设置4个子窗口，每个子窗口均可输出32bit Y通道亮度总和，亮度均值需要在软件中除以每个子窗口的像素数求得。
- 直方图统计，最大支持15X15分块，最新支持5X5分块，该模式下的带权重直方图统计，根据分块数和对应分配的权重，进行32段8bit亮度统计，每个亮度分段内像素数的有效bit数为16bit。

AWB统计信息

AWB硬件统计信息包含全局统计信息和区域统计信息。

全局统计信息：图像全局AWB统计窗口内分色温区域的R,G,B均值，以及有效统计点的个数，色温区域支持7个色温。

分块统计信息：图像全局AWB统计窗口内15x15分块，每个分块的R,G,B均值。

AF统计信息

AF硬件统计信息包含2个主窗口统计信息以及1个主窗口中分块统计信息。

主窗口统计信息：AF统计主窗口内AF统计信息。

分块统计信息：AF统计主窗口内15x15分块的统计信息。

API参考

`rk_aiq_uapi_sysctl_get3AStats`

【描述】

获取3A统计信息。

【语法】

```
XCamReturn  
rk_aiq_uapi_sysctl_get3AStats(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_api_sysctl.h
- 库文件: librkaiq.so

rk_aiq_uapi_sysctl_get3AStatsBlk

【描述】

同步获取3A统计信息。

【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】

| 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

参考代码

```
// Set AF meas config
rk_aiq_user_api_af_SetAttrib(ctx, &attr);
while (1)
{
    rk_aiq_isp_stats_t *stats_ref = NULL;
```

```

// Get 3A stats, return 3a stats on each frame
ret = rk_aiq_uapi_sysctl_get3AStatsBlk(ctx->aiq_ctx , &stats_ref, -1);
if (ret == XCAM_RETURN_NO_ERROR && stats_ref != NULL) {
{
    // User Auto Focus Alg Source
    UsrAfAlgRun();
    // Update Lens Position
    LensDrv_ZoomToDestPos(pos, pps);
    LensDrv_FocusToDestPos(pos, pps);
    // Release 3A stats
    rk_aiq_uapi_sysctl_release3AStatsRef(ctx->aiq_ctx, stats_ref);
}
}

}

```

数据类型

rk_aiq_isp_stats_t

【说明】

AIQ 3A统计信息

【定义】

```

typedef struct {
    rk_aiq_isp_aec_stats_t aec_stats;
    rk_aiq_awb_stat_res_v200_t awb_stats_v200;
    rk_aiq_isp_af_stats_t af_stats;
} rk_aiq_isp_stats_t;

```

【成员】

| 成员名称 | 描述 |
|----------------|---------|
| aec_stats | ae统计信息 |
| awb_stats_v200 | awb统计信息 |
| af_stats | af统计信息 |

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 RKAiqAecStats_s {
    RkAiqAecHwStatsRes_t ae_data;
    RKAiqAecExpInfo_t ae_exp;
} RKAiqAecStats_t;

```

【成员】

| 成员名称 | 描述 |
|----------------------|----------------|
| RkAiqAecHwStatsRes_t | AE模块硬件统计信息 |
| RKAiqAecExpInfo_t | 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 RKAiqAecExpInfo_s {
    RkAiqExpParamComb_t LinearExp;
    RkAiqExpParamComb_t HdrExp[3];
    unsigned short line_length_pixels;
    unsigned short frame_length_lines;
    float pixel_clock_freq_mhz;
} RKAiqAecExpInfo_t;
```

【成员】

| 成员名称 | 描述 |
|----------------------|-------------------------------|
| LinearExp | 非HDR模式的曝光参数信息 |
| HdrExp | HDR模式的曝光参数信息 |
| 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

【定义】

```
typedef struct {
    RkAiqExpRealParam_t exp_real_params; //real value
    RkAiqExpSensorParam_t exp_sensor_params; //reg value
} RkAiqExpParamComb_t;
```

```

typedef struct RkAiqExpRealParam_s {
    float integration_time;
    float analog_gain;
    float digital_gain;
    float isp_dgain;
    int iso;
    int dcg_mode;
} RkAiqExpRealParam_t;

```

```

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】

| 成员名称 | 描述 |
|-------------------------|---------------------------------------|
| integration_time | 曝光积分时间，单位为秒 |
| analog_gain | sensor的模拟增益/Total增益 |
| digital_gain | sensor的数字增益，暂时无效。数字增益大小合并到analog_gain |
| isp_dgain | isp的数字增益，暂时无效 |
| iso | 感光度，暂时无效 |
| dcg_mode | dual conversion gain模式 |
| fine_integration_time | fine曝光积分时间寄存器值，暂时无效 |
| coarse_integration_time | 曝光积分时间寄存器值【行数】 |
| analog_gain_code_global | sensor模拟增益寄存器值 |
| digital_gain_global | sensor数字增益寄存器值，暂时无效 |
| 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精度。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

RkAiqAecHwStatsRes_t

【说明】

AE模块硬件统计信息

【定义】

```

typedef struct RkAiqAecHwStatsRes_s {
    Aec_Stat_Res_t chn[3];
    Aec_Stat_Res_t extra;
    struct yuvae_stat yuvae;
    struct sihist_stat sihist;
} RkAiqAecHwStatsRes_t;

```

【成员】

| 成员名称 | 描述 |
|----------------|---|
| Aec_Stat_Res_t | AE模块基于raw图的统计信息，兼容HDR与非HDR模式，至多支持HDR 3TO1 S/M/L的统计信息。 |
| yuvae_stat | AE模块基于gamma前RGB图的分块信息 |
| sihist_stat | AE模块基于gamma前RGB图的直方图信息 |

【注意事项】

- Aec_Stat_Res_t chn[3]: 代表HDR Merge模块前3个Raw数据通路的统计信息。非HDR模式，对应下标为0，其他下标均无效；HDR 2TO1模式，对应下标为0时表示短帧数据通路统计信息、下标1表示长帧数据通路统计信息，下标2无效；HDR 3TO1模式，对应下标为0时表示短帧数据通路统计信息、下标1表示中帧数据通路统计信息、下标2表示长帧数据通路统计信息。基于raw图的统计模块之前有BLC AWB模块，因此基于raw图的统计信息受BLC、AWB的增益值影响。
- Aec_Stat_Res_t extra: HDR模式下，extra表示HDR合成后经debayer的raw图统计信息。该统计模块之前有BLC、AWB、HDRMERGE、TMO模块，因此该模块的统计信息受BLC、AWB、HDRMERGE、TMO的增益影响。

Aec_Stat_Res_t

【说明】

AE模块基于raw图的统计信息

【定义】

```

typedef struct Aec_Stat_Res_s {
    //rawae
    struct rawaebig_stat rawae_big;
    struct rawaelite_stat rawae_lite;
    //rawhist
    struct rawhist_stat rawhist_big;
    struct rawhist_stat rawhist_lite;
} Aec_Stat_Res_t;

```

【成员】

| 成员名称 | 描述 |
|----------------|---------------------|
| rawaebig_stat | 基于raw图的big模式分块统计信息 |
| rawaelite_stat | 基于raw图的lite模式分块统计信息 |
| rawhist_stat | 基于raw图的直方图统计信息 |

【注意事项】

- 有关基于raw图统计的big、lite模式区别详见功能描述模块。由于big与lite模式的主要区别在于分块统计均值亮度的块数及是否支持子窗口均值亮度统计，故此处基于raw图的big、lite模式直方图统计具有相同的数据结构。

rawaebig_stat

【说明】

基于raw图的big模式统计信息，包含全局窗口分块R/G/B均值亮度、子窗口R/G/B亮度总和

【定义】

```
struct rawaebig_stat {  
    unsigned short channelr_xy[RAWAEBIG_WIN_NUM];  
    unsigned short channelg_xy[RAWAEBIG_WIN_NUM];  
    unsigned short channelb_xy[RAWAEBIG_WIN_NUM];  
    unsigned int   channely_xy[RAWAEBIG_WIN_NUM]; //not HW!  
    unsigned long int wndx_sumr[RAWAEBIG_SUBWIN_NUM];  
    unsigned long int wndx_sumg[RAWAEBIG_SUBWIN_NUM];  
    unsigned long int wndx_sumb[RAWAEBIG_SUBWIN_NUM];  
    unsigned short wndx_channelr[RAWAEBIG_SUBWIN_NUM]; //not HW!  
    unsigned short wndx_channelg[RAWAEBIG_SUBWIN_NUM]; //not HW!  
    unsigned short wndx_channelb[RAWAEBIG_SUBWIN_NUM]; //not HW!  
    unsigned char  wndx_channely[RAWAEBIG_SUBWIN_NUM]; //not HW!  
};  
#define RAWAEBIG_WIN_NUM      225  
#define RAWAEBIG_SUBWIN_NUM   4
```

【成员】

| 成员名称 | 描述 |
|-------------|------------------------------------|
| channelr_xy | big模式全局窗口分块的r通道均值亮度信息。有效比特数：10bit。 |
| channelg_xy | big模式全局窗口分块的g通道均值亮度信息。有效比特数：12bit。 |
| channelb_xy | big模式全局窗口分块的b通道均值亮度信息。有效比特数：10bit。 |
| wndx_sumr | big模式子窗口的r通道亮度和信息。有效比特数：29bit。 |
| wndx_sumg | big模式子窗口的g通道亮度和信息。有效比特数：32bit。 |
| wndx_sumb | big模式子窗口的b通道亮度和信息。有效比特数：29bit。 |

【注意事项】

- 基于raw图的big模式统计信息，仅包含R/G/B 3通道的统计信息，如需Y通道统计信息，可在软件中添加代码根据R/G/B统计值计算。
- 基于raw图的big模式全局窗口分块统计信息为做了除法的均值亮度统计信息，但子窗口为整个窗口的亮度和信息，需要在软件添加代码计算子窗口的均值亮度统计信息。
- 结构体中的channely_xy、wndx_channelr、wndx_channelg、wndx_channelb、wndx_channely参数皆为软件计算参数，需要添加代码，根据硬件统计值计算求得。

rawaelite_stat

【说明】

基于raw图的lite模式统计信息，包含全局窗口分块R/G/B均值亮度

【定义】

```
struct rawaelite_stat {
    unsigned short channelr_xy[RAWAELITE_WIN_NUM];
    unsigned short channelg_xy[RAWAELITE_WIN_NUM];
    unsigned short channelb_xy[RAWAELITE_WIN_NUM];
    unsigned int   channely_xy[RAWAELITE_WIN_NUM]; //not HW!
};

#define RAWAELITE_WIN_NUM 25
```

【成员】

| 成员名称 | 描述 |
|-------------|------------------------------------|
| channelr_xy | big模式全局窗口分块的r通道均值亮度信息。有效比特数：10bit。 |
| channelg_xy | big模式全局窗口分块的g通道均值亮度信息。有效比特数：12bit。 |
| channelb_xy | big模式全局窗口分块的b通道均值亮度信息。有效比特数：10bit。 |

【注意事项】

- 基于raw图的lite模式统计信息，仅包含R/G/B 3通道的统计信息，如需Y通道统计信息，可在软件中添加代码根据R/G/B统计值计算。
- 结构体中的channely_xy为软件计算参数，需要添加代码，根据硬件统计值计算求得。

rawhist_stat

【说明】

基于raw图的直方图统计信息

【定义】

```
struct rawhist_stat {
    unsigned int bins[RAWHIST_BIN_N_MAX];
};

#define RAWHIST_BIN_N_MAX 256
```

【成员】

| 成员名称 | 描述 |
|------|---------------------------|
| bins | 直方图的分段，共256段，有效bit数：28bit |

yuvae_stat

【说明】

基于gamma前RGB图的分块均值亮度统计信息，包含全局窗口分块Y通道均值亮度、子窗口Y通道亮度总和

【定义】

```

struct yuvae_stat {
    unsigned long int ro_yuvae_sumy[YUVAE_SUBWIN_NUM];
    unsigned char mean[YUVAE_WIN_NUM];
};

#define YUVAE_SUBWIN_NUM 4
#define YUVAE_WIN_NUM 225

```

【成员】

| 成员名称 | 描述 |
|---------------|---------------------------|
| ro_yuvae_sumy | 子窗口的Y通道亮度总和，有效bit数：32bit |
| mean | 全局窗口分块Y通道均值亮度，有效bit数：8bit |

【注意事项】

- 基于raw图的lite模式统计信息，仅包含R/G/B 3通道的统计信息，如需Y通道统计信息，可在软件中添加代码根据R/G/B统计值计算。
- 结构体中的channel_y_xy为软件计算参数，需要添加代码，根据硬件统计值计算求得。

sihist_stat

【说明】

基于gamma前RGB图的直方图计信息

【定义】

```

struct sihist_stat {
    unsigned int bins[SIHIST_BIN_N_MAX];
};

#define SIHIST_BIN_N_MAX 32

```

【成员】

| 成员名称 | 描述 |
|------|-------------------------|
| bins | 直方图的分段，共32段，有效比特数：16bit |

rk_aiq_awb_stat_res_v200_t

【说明】

定义白平衡硬件统计信息

【定义】

```

typedef struct rk_aiq_awb_stat_res_v200_s {
    rk_aiq_awb_stat_wp_res_light_v200_t light[RK_AIQ_AWB_MAX_WHITEREGIONS_NUM];
    rk_aiq_awb_stat_blk_res_v200_t blockResult[RK_AIQ_AWB_GRID_NUM_TOTAL];
    rk_aiq_awb_stat_wp_res_light_v200_t
        multiwindowLightResult[RK_AIQ_AWB_MAX_WHITEREGIONS_NUM];
    rk_aiq_awb_stat_wp_res_v200_t
        excWpRangeResult[RK_AIQ_AWB_STAT_WP_RANGE_NUM_V200];
} rk_aiq_awb_stat_res_v200_t;

```

【成员】

| 成员名称 | 描述 |
|------------------------|--|
| light | 主窗口下不同光源下的白点统计结果，最多 RK_AIQ_AWB_MAX_WHITEREGIONS_NUM个光源； |
| blockResult | 每个块的RGB累加，图像进行不重叠同尺寸的 15x15 (RK_AIQ_AWB_GRID_NUM_TOTAL) 分块 |
| multiwindowLightResult | 几个子窗口内不同光源下的白点统计结果，最多 RK_AIQ_AWB_MAX_WHITEREGIONS_NUM个光源； |
| excWpRangeResult | 落在非白点区域里的非白点统计结果，最多 RK_AIQ_AWB_STAT_WP_RANGE_NUM_V200个非白点区域 |

【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。

rk_aiq_awb_stat_wp_res_light_v200_t

【说明】

定义某个光源下的白点统计结果

【定义】

```
typedef struct rk_aiq_awb_stat_wp_res_light_v200_s {
    rk_aiq_awb_stat_wp_res_v200_t xYType[RK_AIQ_AWB_XY_TYPE_MAX_V200];
} rk_aiq_awb_stat_wp_res_light_v200_t;
```

【成员】

| 成员名 称 | 描述 |
|----------|--|
| xYType | 某个光源下不同大小的XY框的白点统计结果，最多 RK_AIQ_AWB_XY_TYPE_MAX_V200个框 |

rk_aiq_awb_stat_wp_res_v200_t

【说明】

定义某个光源某个大小的XY框下的白点统计结果，后非白点区域里的非白点统计结果

【定义】

```
typedef struct rk_aiq_awb_stat_wp_res_v200_s {
    unsigned int WpNo;
    unsigned int Rvalue;
    unsigned int Gvalue;
    unsigned int Bvalue;
} rk_aiq_awb_stat_wp_res_v200_t;
```

【成员】

| 成员名称 | 描述 |
|--------|---------------|
| WpNo | (非) 白点数量 |
| Rvalue | (非) 白点R通道的累加和 |
| Gvalue | (非) 白点G通道的累加和 |
| Bvalue | (非) 白点B通道的累加和 |

rk_aiq_awb_stat_blk_res_v200_t

【说明】

定义每个块的统计结果

【定义】

```
typedef struct rk_aiq_awb_stat_blk_res_v200_s {
    unsigned int Rvalue;
    unsigned int Gvalue;
    unsigned int Bvalue;
    bool isWP[RK_AIQ_AWB_STORE_LS_WPFLAG_NUM];
} rk_aiq_awb_stat_blk_res_v200_t;
```

【成员】

| 成员名称 | 描述 |
|--------|--|
| isWP | 块内是否包含某个光源白点的标志, 最多纪录 RK_AIQ_AWB_STORE_LS_WPFLAG_NUM个光源的标志 |
| Rvalue | 块内所有点R通道的累加和 |
| Gvalue | 块内所有点RG通道的累加和 |
| Bvalue | 块内所有点RB通道的累加和 |

rk_aiq_af_algo_stat_t

【说明】

定义AF统计信息

【定义】

```
typedef struct {
    unsigned int roia_sharpness;
    unsigned int roia_luminance;
    unsigned int roib_sharpness;
    unsigned int roib_luminance;
    unsigned int global_sharpness[RKAIQ_RAWAF_SUMDATA_NUM];
    struct timeval focus_starttim;
    struct timeval focus_endtim;
    int64_t sof_tim;
} rk_aiq_af_algo_stat_t;
```

【成员】

| 成员名称 | 描述 |
|------------------|--------------------|
| roia_sharpness | 主窗口的清晰度值; |
| roia_luminance | 主窗口的亮度值; |
| roib_sharpness | 独立窗口的清晰度值; |
| roib_luminance | 独立窗口的亮度值; |
| global_sharpness | 主窗口下15*15子窗口的清晰度值; |
| focus_starttim | 最近一次VCM移动的起始时间; |
| focus_endtim | 最近一次VCM移动的结束时间; |
| sof_tim | 本次数据帧的帧开始时间，单位ns; |

【注意事项】

roia_sharpness/roia_luminance/roib_sharpness/roib_luminance/global_sharpness为AF硬件统计信息。

focus_starttim/focus_endtim/sof_tim为VCM移动时间和数据帧的帧开始时间，辅助确认VCM是否移动结束，AF硬件统计信息是否可靠。

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: v0.1.6
```

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: 2020-06-05
    AIQ: v0.1.6
    IQ PARSER: v1.0.0
RK INTEGRATED ALGO MODULES:
    AWB: v0.0.9
    AEC: v0.1.1
    AF: v0.0.9
    AHDR: v0.0.9
    ANR: v0.0.9
    ASHARP: v0.0.9
    ADEHAZE: v0.0.9
    AGAMMA: v0.0.9
    A3DLUT: v0.0.9
    ABLC: v0.0.9
    ACCM: v0.0.9
    ACGC: v0.0.9
    ACP: v0.0.9
    ADEBAYER: v0.0.1
    ADPCC: v0.0.9
    AGIC: v0.0.9
    AIE: v0.0.1
    ALDCH: v0.0.9
    ALSC: v0.0.9
    AORB: v0.0.9
    AR2Y: v0.0.9
    ASD: v0.0.9
    AWDR: v0.0.9
*****
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

Log开关

1. 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] [AFEC] [ACGC] [AORB] [ASHARP]  
  
bit: 30      29      28      27      26      25      24      23      22  
mean: [AIE] [ACP] [AR2Y] [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
```

2. 模块log级别配置：

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。

例如开启af模块的log开关，并且级别为verbose，则bit[14] = 1, bit[3-0] = 4, 所以在应用程序执行前执行：

```
export persist_camera_engine_log=0x4014
```

查看当前log级别可通过如下命令：

```
[root@RV1126_RV1109:/]# echo $persist_camera_engine_log  
0x4014
```

3. 各模块开关列表

| 模块 | Debug log | Verbose log |
|-----|---|--|
| AE | export persist_camera_engine_log=0x1ff3 | export persist_camera_engine_log=0x1ff4 |
| AF | export persist_camera_engine_log=0x4ff3 | export persist_camera_engine_log=0x4ff4 |
| AWB | export persist_camera_engine_log=0x2ff3 | export persist_camera_engine_log=0x2ff4 |
| NR | export persist_camera_engine_log=0x40fff | |

Log解读

AE

由于篇幅限制，此处仅对debug等级的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,regain=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 |
| dcg | 当前帧对应的dcg模式。-1: sensor不支持dcg模式 或 sensor内部进行dcg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_algo_ae_ift.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_algo_ae_ift.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: RecRun: SMeanLuma=9.342692, MMeanLuma=37.698597, IMeanLuma=0.000000, IMoMeanLuma=37.571430, Isconverged=0, Longfrm=0
rk_aiq_ae_algo.cpp:6013: >>> FrameNum=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,Isconverg  
ed=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,lti  
me=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。 |

AF

1) 每帧的AF统计值

```
rk_aiq_algo_af_itf.cpp:465: AFProcessing: sharpness roia: 2454915276 - 16253362 roib: 4770628 -
7350891 vcm_stable: 1, 1618759, 1618375, 1618382, 30.006588, sof_tim 1618712, zoom_stable 1
```

| 成员名称 | 描述 |
|-------------|--|
| roia | window A的FV值和亮度值 |
| roib | window B的FV值和亮度值 |
| vcm_stable | vcm移动是否结束标记: 0: 未结束 1: 结束 当前时刻, 单位ms vcm移动起始时刻, 单位ms vcm移动结束时刻, 单位ms 当前帧曝光时间, 单位ms |
| sof_tim | 当前帧的帧数据传输起始时刻, 单位ms |
| zoom_stable | zoom缩放是否结束 |

vcm_stable的判断方法

```
if (sof_time >= vcm_end_time + exposure_time + extraDelay)
    vcm_stable = true;
else
    vcm_stable = false;
```

extraDelay在IQ xml中指定, 可正, 可负, 可为零

vcm_start_time、vcm_end_time从vcm驱动查询得到

zoom_stable的判断方法

```
if (cur_time >= zoom_end_time)
    zoom_stable = true;
else
    zoom_stable = false;
```

zoom_start_time、zoom_end_time从zoom驱动查询得到

2) 对焦触发:

af_trigger.cpp:158: curSharpness: 178.780685, triggered: 1, sceneChanged 1

当前帧FV值与上次成功对焦的FV值的变化大于TrigThers, 且连续StableFrames帧的FV变化小于StableThers时, 触发对焦。

TrigThers、StableFrames、StableThers由IQ xml指定

3) 搜索路径:

af_trigger.cpp:921: Search list is:

af_trigger.cpp:933: ->index: 0 pos: 64 stage: 1

af_trigger.cpp:938: index: 1 pos: 56 stage: 1

af_trigger.cpp:938: index: 2 pos: 48 stage: 1

```
af_trigger.cpp:938: index: 3 pos: 40 stage: 1
af_trigger.cpp:938: index: 4 pos: 32 stage: 1
af_trigger.cpp:938: index: 5 pos: 24 stage: 1
af_trigger.cpp:938: index: 6 pos: 16 stage: 1
af_trigger.cpp:938: index: 7 pos: 8 stage: 1
af_trigger.cpp:938: index: 8 pos: 0 stage: 1
```

4) 显示当前iso值，并根据iso值更新相关配置

```
af.cpp:1615: AfCalcMeasCfgByIso: current iso = 599, again 11.999999, dgain 1.000000!
af.cpp:171: AfUpdateMeasIsoCfg: iso = 800
```

5) 粗调结果

```
XCAM INFO (807) af_search.cpp:25: --> SearchIdx 1 route(search rough) is:
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 0, pos: 64, sharpness: 178.836884,
dSharpness: 0.000000, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 1, pos: 56, sharpness: 177.747253,
dSharpness: -0.003056, abs_dSharpness: -0.003056, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 2, pos: 48, sharpness: 178.024612,
dSharpness: 0.000780, abs_dSharpness: -0.002276, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 3, pos: 40, sharpness: 180.054565,
dSharpness: 0.005669, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 4, pos: 32, sharpness: 180.206558,
dSharpness: 0.000422, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 5, pos: 24, sharpness: 180.668610,
dSharpness: 0.001280, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 6, pos: 16, sharpness: 186.813889,
dSharpness: 0.016723, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 7, pos: 8, sharpness: 196.762360,
dSharpness: 0.025936, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 1, index: 8, pos: 0, sharpness: 206.649445,
dSharpness: 0.024509, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:38: MaxSharpnessPos 0, MaxSharpness: 206.649445,
MinSharpness: 177.747253
```

6) 细调结果

```
XCAM INFO (807) af_search.cpp:25: --> SearchIdx 1 route(search fine) is:
XCAM INFO (807) af_search.cpp:32: stage: 0, index: 0, pos: 8, sharpness: 196.762360,
dSharpness: 0.000000, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 0, index: 1, pos: 4, sharpness: 207.176315,
dSharpness: 0.025781, abs_dSharpness: 0.000000, skip: 0, quick_focus: 0
XCAM INFO (807) af_search.cpp:32: stage: 0, index: 2, pos: 0, sharpness: 206.649445,
dSharpness: -0.001273, abs_dSharpness: -0.001273, skip: 0, quick_focus: 1
XCAM INFO (807) af_search.cpp:38: MaxSharpnessPos 4, MaxSharpness: 207.176315,
MinSharpness: 177.747253
```

7) 对焦结果

af_search.cpp:1010: AfSearching: Found focus(pos: 4 sharpness: 207.176315, distance: 0.000 0)!

AWB

参考《Rockchip_Color_Optimization_Guide_ISP2x_CN》

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

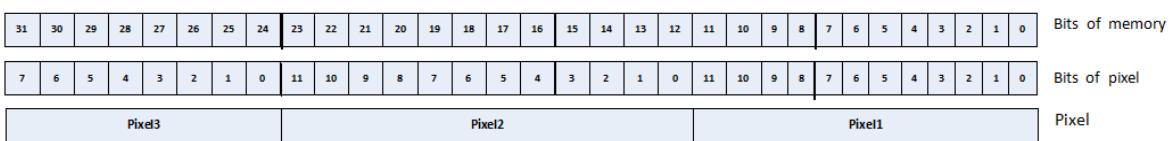
非紧凑型存储格式

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。

| 数据块定义 | | | | |
|-------|------|----------|--------|--------------------------|
| 项目 | 数据类型 | 长度(Byte) | 默认值 | 说明 |
| 起始标识符 | u16 | 2 | 0xFF00 | 固定值: 文件起始 |
| 块标识符 | u16 | 2 | 0xFF01 | 标识符: Raw信息 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF02 | 标识符: Normal模式 Raw数据 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF03 | 标识符: HDR2/HDR3短帧 Raw数据 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF04 | 标识符: HDR2长帧/HDR3中帧 Raw数据 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF05 | 标识符: HDR3长帧 Raw数据 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF06 | 标识符: 帧统计信息 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF07 | 标识符: ISP寄存器格式 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF08 | 标识符: ISP寄存器 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF09 | 标识符: ISPP寄存器格式 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF0a | 标识符: ISPP寄存器 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| 块标识符 | u16 | 2 | 0xFF0b | 标识符: 平台信息 |
| 块长度 | u32 | 4 | - | |
| 块数据 | | | | |
| ... | | | | |
| 结束标识符 | u16 | 2 | 0x00FF | 固定值: 文件结尾 |

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(RGBB); |
| 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](#)。

| 数据块定义: Raw数据 0xFF02 ~ 0xFF05 | | | |
|------------------------------|------|----------------------------------|-------|
| 项目 | 数据类型 | 长度(Byte) | 说明 |
| Raw数据 | - | 长*宽*N 非紧凑: N=2 紧凑: N=bpp/8 | Raw数据 |

帧统计信息块定义

| 数据块: 帧统计信息 0xFF06 | | | | |
|-------------------|-------|----------|--------|--|
| 项目 | 数据类型 | 长度(Byte) | 默认值 | 说明 |
| 版本号 | u16 | 2 | 0x0200 | Raw格式版本号: 0x0200=v2.0 |
| 帧号 | u32 | 4 | - | 帧号 |
| Normal_Exp | float | 4 | - | 线性模式: 快门时间 |
| Normal_Gain | float | 4 | - | 线性模式: 曝光增益 |
| Normal_Exp_REG | u32 | 4 | - | 线性模式: 快门时间的SENSOR寄存器值 |
| Normal_Gain_REG | u32 | 4 | - | 线性模式: 曝光增益的SENSOR寄存器值 |
| HDR_Exp_L | float | 4 | - | HDR3: 长帧快门时间 |
| HDR_Gain_L | float | 4 | - | HDR3: 长帧曝光增益 |
| HDR_Exp_L_REG | u32 | 4 | - | HDR3: 长帧快门时间的SENSOR寄存器值 |
| HDR_Gain_L_REG | u32 | 4 | - | HDR3: 长帧曝光增益的SENSOR寄存器值 |
| HDR_Exp_M | float | 4 | - | HDR3: 中帧快门时间 HDR2: 长帧快门时间 |
| HDR_Gain_M | float | 4 | - | HDR3: 中帧曝光增益 HDR2: 长帧曝光增益 |
| HDR_Exp_M_REG | u32 | 4 | - | HDR3: 中帧快门时间的SENSOR寄存器值 HDR2: 长帧快门时间的SENSOR寄存器值 |
| HDR_Gain_M_REG | u32 | 4 | - | HDR3: 中帧曝光增益的SENSOR寄存器值 HDR2: 长帧曝光增益的SENSOR寄存器值 |
| HDR_Exp_S | float | 4 | - | HDR3: 短帧快门时间 HDR2: 短帧快门时间 |
| HDR_Gain_S | float | 4 | - | HDR3: 短帧曝光增益 HDR2: 短帧曝光增益 |
| HDR_Exp_S_REG | u32 | 4 | - | HDR3: 短帧快门时间的SENSOR寄存器值 HDR2: 短帧快门时间的SENSOR寄存器值 |
| HDR_Gain_S_REG | u32 | 4 | - | HDR3: 短帧曝光增益的SENSOR寄存器值 HDR2: 短帧曝光增益的SENSOR寄存器值 |
| AWB_Rgain | float | 4 | - | 白平衡增益 |
| AWB_Bgain | float | 4 | - | 白平衡增益 |

寄存器格式块定义

| 数据块定义: 寄存器格式 0xFF07/0xFF09 | | | | |
|----------------------------|------|----------|--------|------------------|
| 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 |
| 帧号 | u32 | 4 | - | 寄存器基地址 |
| 基地址 | u32 | 4 | - | 寄存器基地址 |
| 偏移地址 | u32 | 4 | - | 起始偏移地址 |
| 数目 | u32 | 4 | - | 寄存器数目 |

寄存器块定义

| 数据块定义：寄存器 0xFF08/0xFF0a | | | |
|-------------------------|------|----------|-------|
| 项目 | 数据类型 | 长度(Byte) | 说明 |
| 寄存器数据 | - | - | 寄存器数据 |

平台信息块定义

| 数据块定义：平台信息 0xFF0b | | | |
|-------------------|--------|----------|----|
| 项目 | 数据类型 | 长度(Byte) | 说明 |
| 芯片型号 | string | 32 | |
| ISP版本 | string | 32 | |
| AIQ版本 | string | 32 | |

Raw/YUV数据采集方式

AIQ动态截存方式

目前RV1109/RV1126 AIQ 采用的数据流粗略流程如下：

CIS ----> csi tx ---raw---> ddr -----> csi-rx -> ... -> isp-> ... ->ispp -> ... -> out-yuv,

其中csi tx ---raw---> ddr -----> csi-rx 对于ISP硬件来说称为回读模式。

Raw:

AIQ根据/tmp/.capture_cnt中间文件获取用户想保存raw文件的帧数，AIQ将对应帧数的raw图写入/tmp目录下。

截存的Raw图为ISP IP实际处理的数据流，同时CIS 曝光以及ISP 各个图像处理模块都继续处于AIQ 的控制之下，所以通过截存RAW来分析实际预览过程中的问题.

YUV:

YUV数据是集成AIQ的应用直接操作v4l2 video获取，所以AIQ本身不支持截存YUV数据。截存YUV数据的功能需要应用实现。rkisp_demo是RK提供的一份集成了AIQ调用的参考应用，运行该应用可以实现同步截存YUV的功能。

集成AIQ的客户应用抓raw图步骤

1. 运行rkaiq。
2. echo要抓取的raw图帧数, 例如抓取3帧

```
linux系统下执行: echo 3 > /tmp/.capture_cnt
android系统下执行: echo 3 > /data/.capture_cnt
```

3. 在/tmp/capture_image或/data/capture_image目录下会生成抓取的raw图及对应的meta信息

linux系统下:

```
[root@RV1126_RV1109:/]# ls -l /tmp/capture_image/raw_2017-08-15_20-40-58/
total 35932
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_short.raw
-rw-r--r-- 1 root root 381 Aug 15 20:40 meta_data
```

android系统下:

```
[root@RV1126_RV1109:/]# ls -l /data/capture_image/raw_2017-08-15_20-40-58/
total 35932
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_short.raw
-rw-r--r-- 1 root root      381 Aug 15 20:40 meta_data
```

运行rkisp_demo,抓raw图及对应的yuv图像步骤

1. 加--sync-to-raw参数, 运行rkisp_demo, 只有rkisp_demo支持

```
rkisp_demo --device /dev/video14 --width 1280 --height 720 --vop --rkaiq --hdr 2
--sync-to-raw
```

2. echo要抓取的raw/yuv图帧数, 例如抓取3帧

```
linux系统下执行: echo 3 > /tmp/.capture_cnt
android系统下执行: echo 3 > /data/.capture_cnt
```

3. 在/tmp/capture_image或/data/capture_image目录下会生成抓取的raw图/meta信息/yuv图像

linux系统下:

```
[root@RV1126_RV1109:/]# ls -l /tmp/capture_image/raw_2017-08-15_20-40-58/
total 35932
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_short.raw
-rw-r--r-- 1 root root      381 Aug 15 20:40 meta_data
```

android系统下:

```
[root@RV1126_RV1109:/]# ls -l /data/capture_image/raw_2017-08-15_20-40-58/
total 35932
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame476_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame477_2688x1520_short.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_long.raw
-rw-r--r-- 1 root root 6128640 Aug 15 20:40 frame478_2688x1520_short.raw
-rw-r--r-- 1 root root      381 Aug 15 20:40 meta_data
```

4. 如上所示, raw图/meta信息/yuv图像是一一对应

v4l2-ctl直接采集方式

v4l2-ctl可以直接操作v4l2 video设备节点采集Raw/YUV数据, 但是是通过直接配置video的最终输出数据格式来实现的, 所以只能采集Raw 和采集YUV进行2选1.

具体配置方式及抓取命令参见《Rockchip_Driver_Guid_V1》的FAQ章节之“如何抓取CIS输出的RAW、YUV数据”。

RK IQ Tool(Tuner)抓图工具

RK IQ Tool(Tuner)抓图工具采集的Raw数据格式为非紧凑型Raw数据。IQ Tool也仅支持处理该Raw数据格式

使用场景介绍

| 方式 | 使用场景 | 描述 |
|-----------------------|---|---------------------------------|
| v4l2-ctrl直接采集方式 | 1. CIS驱动调试，确认输出Raw数据是否正常 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： 非紧凑型 Raw数据 紧凑型Raw |
| AIQ动态截存方式 | 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。 | Raw: RK-RAW V1.0。 |
| RK IQ Tool(Tuner)抓图工具 | 1. 使用RK IQ Tool(Tuner)抓图工具采集Raw图，直接使用RK IQ Tool 进行模组参数标定 | |

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 Tuning | Image Quality Tuning |