Rockchip Developement Guide ISP20

ID: RK-SM-YF-601

Release Version: V1.6.4

Release Date: 2020-1-12

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: [fae@rock-chips.com]

Foreword

Overview

This article aims to describe the role of the RkAiq (Rk Auto Image Quality) module, the overall workflow, and related API interfaces. Mainly to
Engineers who use RkAiq module for ISP function development provide help.

Product Version''

Chip Name	Kernel Version
RV1126/RV1109	Linux 4.19

Target Audience

This document (this guide) is mainly applicable to the following engineers:

ISP module software development engineer

System Integration Software Development Engineer

Support status of each chip system

Chip Name	BuildRoot	Debian	Yocto	Android
RV1126	Y	N	N	N
RV1109	Y	N	N	N

Revision History

Version Number	Author	Revision Date	Revision Description
V1.0.0	Alan Zhong， Dalon Zhang, Hongfei Xu	2020-06-09	Initial version
V1.0.1	Dalon Zhang, Hongfei Xu, Jayne Zhu, Emily Chi	2020-08-04	Add statistical information chapter
V1.1.0	Dalon Zhang	2020-08-14	Add interface descriptions such as backlight compensation and glare suppression
v1.2.0	William Hu	2020-9-25	Added AF statistical value description
v1.3.0	William Hu	2020-10-14	Add AF module API description

Version Number	Author	Revision Date	Revision Description
v1.5.0	Jayne Zhu, Dalon Zhang , Ouyang Yafeng, Liquid Li	2020-11-2	1. AE module api adds demo and update module-level api data type 2. Add demosaicing interface description 3. Add NR/Sharp module api and data type 4. Add Gamma description
v1.6.0	Liquid Li	2020-11-11	1. Add DPCC description
v1.6.1	William Hu	2020-11-18	1. Modify some errors in the AF description
v1.6.2	Jayne Zhu	2020-12-07	Add AE module api new parameter description
v1.6.3	Alan Zhong	2020-12-12	Add rk_aiq_uapi_sysctl_updateIq description
v1.6.4	Alan Zhong	2020-01-12	Add offline frame function description

Content

Content

Overview

ISP20 contains a series of image processing algorithm modules, mainly including: dark current correction, dead pixel correction, 3A, HDR, lens shading correction, lens distortion correction, 3DLUT, denoising (including RAW domain denoising, multi-frame denoising, Color denoising, etc.), sharpening, etc.

ISP20 includes hardware algorithm realization and software logic control part, RkAiq is the realization of software logic control part.

The main functions of the RkAiq software module are: obtaining image statistics from the ISP driver, combining with IQ Tuning parameters, using a series of algorithms to calculate new ISP, Sensor and other hardware parameters, and continuously iterating the process to finally achieve the optimal image effect.

Function description

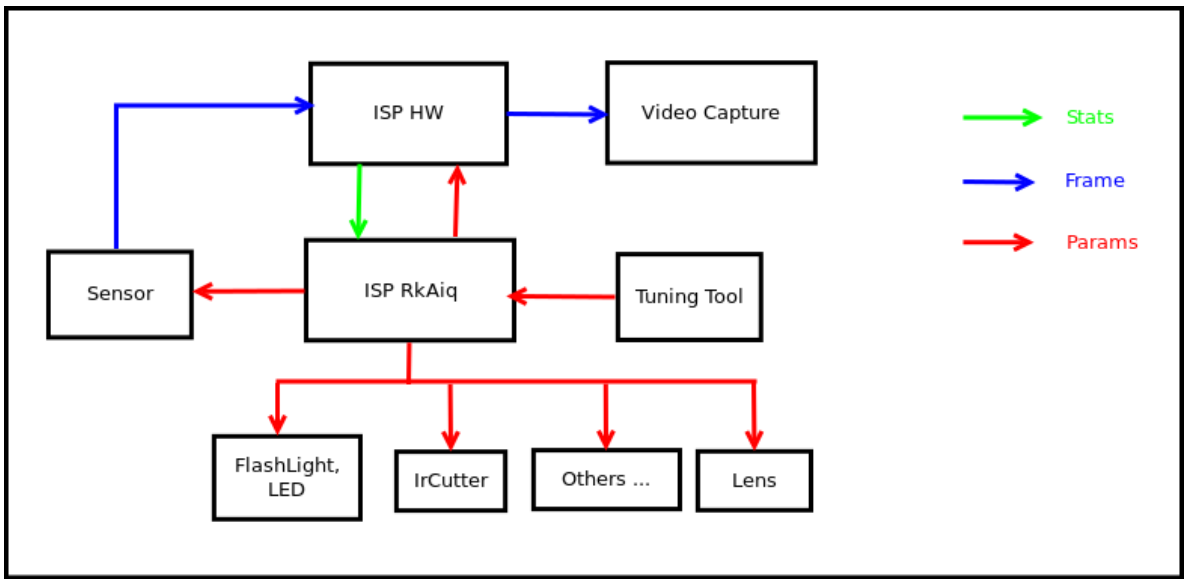


Figure 1-1 ISP20 system block diagram

The overall software and hardware block diagram of ISP20 is shown in Figure 1-1. The Sensor outputs the data stream to the ISP HW, and the ISP HW outputs the image after a series of image processing algorithms. RkAiq continuously obtains statistical data from ISP HW, and generates new parameters through 3A and other algorithms to feed back to each hardware module. Tuning tool can debug the parameters online in real time, and save and generate a new iq parameter file after debugging.

RkAiq Architecture

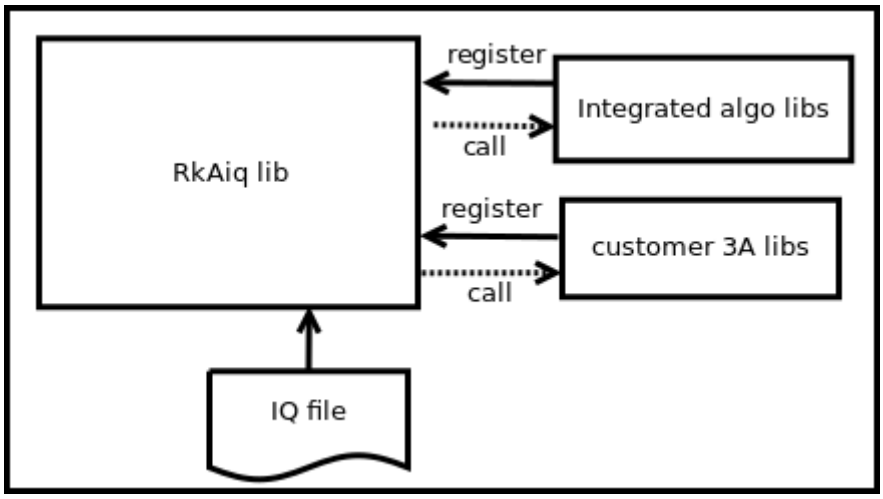


Figure 1-2 RkAiq overall architecture diagram

The design idea of ISP20 RkAiq software is shown in Figure 1-2. Mainly divided into the following four parts:

1. RkAiq lib dynamic library. The library contains the main logic part, which is responsible for obtaining statistics from the driver and transmitting them to each algorithm library.
2. Integrated algo libs. The static algorithm library provided by Rk has been registered to the RkAiq lib dynamic library by default.
3. customer 3A libs. Customers can implement their own 3A algorithm library or other algorithm libraries according to the algorithm library interface definition. After registering the custom algorithm library to the RkAiq lib dynamic library, you can choose to run the custom library or the Rk library according to the provided interface.
4. IQ file. The iq tuning result file saves algorithm related parameters and some system static parameters such as CIS.

Software Architecture

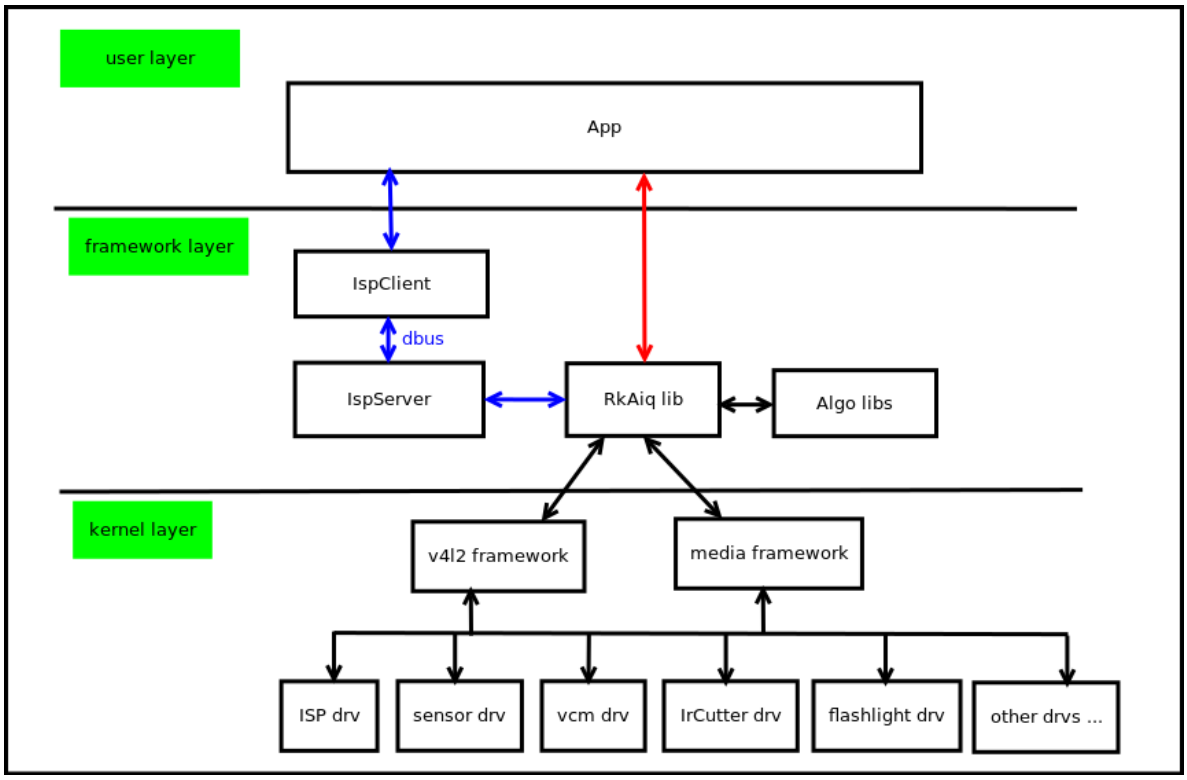


Figure 1-3 Software architecture block diagram

The ISP20 software block diagram is shown in Figure 1-3. Mainly divided into the following three layers:

1. Kernel layer. This layer contains all the hardware drivers of the Camera system, mainly including ISP driver, sensor driver, vcm driver, flashlight driver, IrCutter driver and so on. The drivers are implemented based on the V4L2 and Media framework.
2. framework layer. This layer is the integration layer of RkAiq lib. Rkaiq lib has two integration methods:
    - IspServer method.

      In this way, Rkaiq lib runs in an independent process of IspServer, and the client communicates with it through dbus. In addition, this method can provide images with ISP debugging effects for existing third-party applications such as v4l-ctl without modifying the source code
    - Direct integration method
      RkAiq lib can be directly integrated into the application.
3. User layer. User application layer.

Software Process

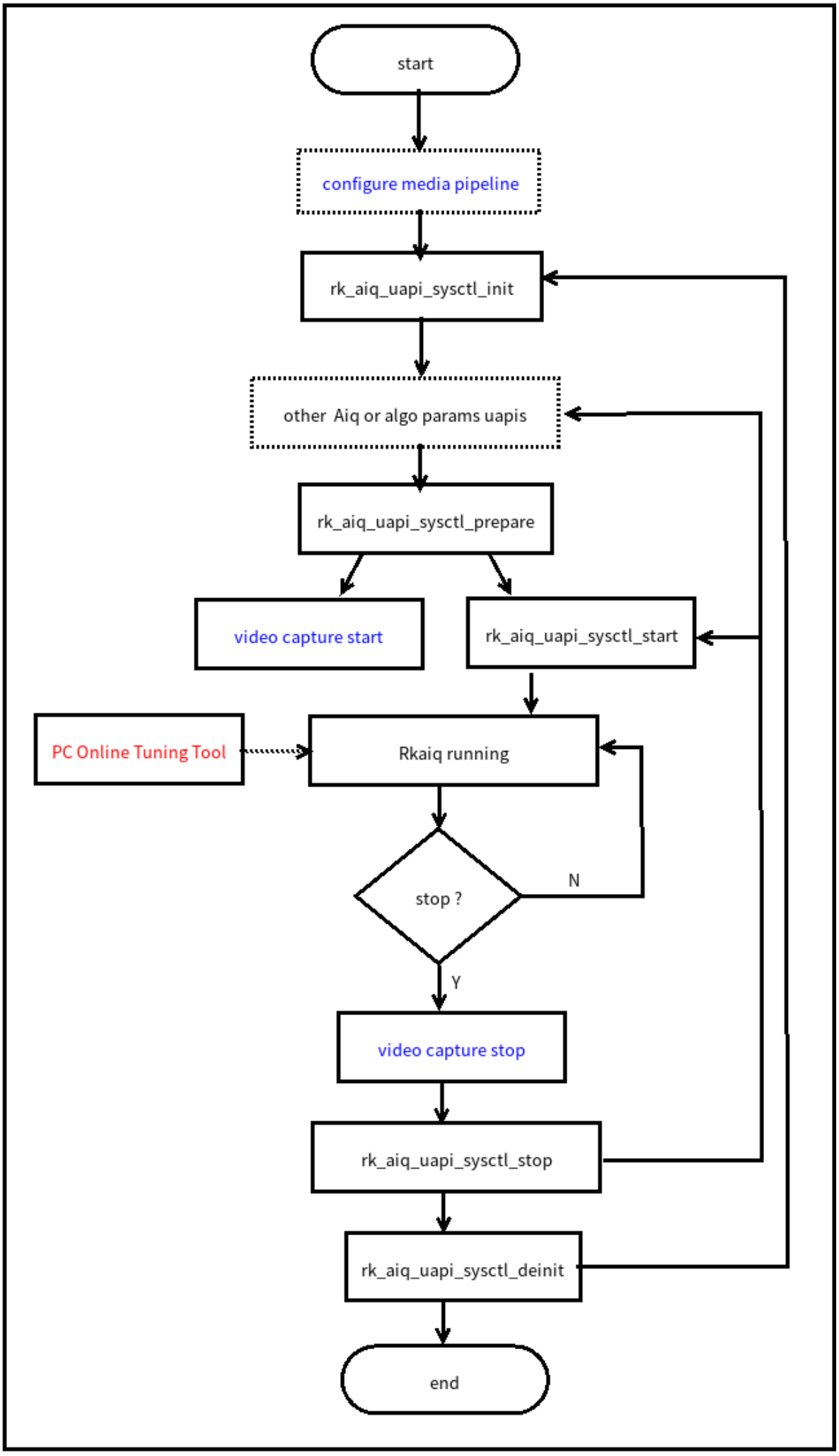

start
configure media pipeline
rk_aiq_uapi_sysctl_init
other Aiq or algo params uapis
rk_aiq_uapi_sysctl_prepare
video capture start
rk_aiq_uapi_sysctl_start
PC Online Tuning Tool
Rkaiq running
stop ?
N
Y
video capture stop
rk_aiq_uapi_sysctl_stop
rk_aiq_uapi_sysctl_deinit
end

Figure 1-4 Flow chart

The calling process of the RkAiq interface is shown in Figure 1-4. The dotted frame in the figure is optional, and the blue font is the configuration that the application needs to cooperate with the RkAiq process.

- configure media pipeline. Optionally, configure ISP20 pipeline, such as sensor output resolution, etc. The driver has default configuration.
- rk_aiq_uapi_sysctl_init. Initialize RkAiq, including IQ tuning parameters and initialization of each algorithm library.
- other Aiq or algo params uapis. Optional, you can configure the required parameters through the API interface provided by each algorithm, and register a third-party algorithm library, etc.
- rk_aiq_uapi_sysctl_prepare. Prepare the initialization parameters of each algorithm library and each hardware module, and set them to the drive.
- video capture start. This process is the start of the ISP data stream on the application side, and this process needs to be called after rk_aiq_uapi_sysctl_prepare.
- rk_aiq_uapi_sysctl_start. Start the internal process of RkAiq. After the interface is successfully called, the sensor starts to output data, the ISP starts to process the data, and output the processed image.
- Rkaiq running. RkAiq continuously obtains statistical data from the ISP driver, calls 3A and other algorithms to calculate new parameters, and applies the new parameters to the driver.
- PC Online Tuning Tool. On the PC side, the parameters can be adjusted online through the Tuning Tool.
- video capture stop. You need to stop the data flow part before stopping the RkAiq process.
- rk_aiq_uapi_sysctl_stop. Stop the RkAiq running process. The parameters can be adjusted and then restarted or restarted directly.
- rk_aiq_uapi_sysctl_deinit. Uninitialize RkAiq.

API description

The API provided by RKAiq is divided into two levels: function level API and module level API. Among them, the function-level API is packaged based on the module-level API, mainly for some simple functional designs based on the module for product applications. The module-level API provides detailed parameter settings and queries for the module, and does not differentiate between functions.

System Control

Functional Overview

The system control part includes AIQ public attribute configuration, initialize AIQ, run AIQ, exit AIQ, set AIQ modules and other functions.

API Reference

rk_aiq_uapi_sysctl_init

【Description】
Initialize the AIQ context.

【Syntax】

```
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);
```

【Parameter】

Parameter Name	Description	Input/Output
sns_ent_name	sensor entity name	input
iq_file_dir	Calibration parameter file path	Input
err_cb	Error callback function, can be NULL	Input
metas_cb	Meta data callback function, can be NULL	Input

【Return value】

Return Value	Description
rk_aiq_sys_ctx_t*	AIQ context pointer

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-Should be called before other functions.

rk_aiq_uapi_sysctl_deinit

【Description】
Deinitialize the AIQ context environment.

【Syntax】

```
void
rk_aiq_uapi_sysctl_deinit( rk_aiq_sys_ctx_t* ctx);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
None	None

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-It should not be called when the AIQ is in the start state.

rk_aiq_uapi_sysctl_prepare

【Description】
Prepare the AIQ operating environment.

【Syntax】

```
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);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
width	The resolution width of sensor output, only for verification	Input
height	The resolution height output by the sensor, only for verification	Input
mode	ISP Pipeline working mode (NORMAL/HDR)	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-It should be called before the rk_aiq_uapi_sysctl_start function.
-If you need to call this function after rk_aiq_uapi_sysctl_start, first call the rk_aiq_uapi_sysctl_stop function, and then call rk_aiq_uapi_sysctl_prepare to prepare the operating environment again.

rk_aiq_uapi_sysctl_start

【Description】
Start the AIQ control system. After the AIQ is started, it will continuously obtain 3A statistics from the ISP driver, run the 3A algorithm, and apply the calculated new parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-It should be called after the rk_aiq_uapi_sysctl_prepare function.

rk_aiq_uapi_sysctl_stop

【Description】
Stop the AIQ control system.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_getStaticMetas

【Description】

Query sensor corresponding static information, such as resolution, data format, etc.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sns_ent_name	sensor entity name	input
static_info	Static information structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

Header file: rk_aiq_user_api_sysctl.h

Library file: librkaiq.so

rk_aiq_uapi_sysctl_enumStaticMetas

【Description】

Enumerate the static information obtained by AIQ.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
index	Index number, starting from 0	Enter
static_info	Static information structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_setModuleCtl

【Description】
AIQ module switch settings.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mId	Module ID	Input
mod_en	true to enable, false to disable	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_getModuleCtl

【Description】
AIQ module status query.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mId	Module ID	Input
mod_en	Current status	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_regLib

【Description】
Register a custom algorithm library.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
algo_lib_des	Algorithm description structure, the field id is the identification ID generated by AIQ	Input & Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_unRegLib

【Description】

Log out of the custom algorithm library.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
algo_type	Type of algorithm module to be operated	Input
lib_id	Algorithm library identification ID	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_enableAxlib

【Description】

Set the running status of the custom algorithm library.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
algo_type	Type of algorithm module to be operated	Input
lib_id	Algorithm library identification ID	Input
enable	Status setting	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-If lib_id is equal to the currently running algorithm library, this function can be called in any state except uninitialized.
-In other cases, it is only called in the prepared state, and the algorithm library identified by algo_type will be replaced by the new algorithm library identified by lib_id.

rk_aiq_uapi_sysctl_getAxlibStatus

【Description】
Get the state of the algorithm library.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
algo_type	Type of algorithm module to be operated	Input
lib_id	Algorithm library identification ID	Input

【Return value】

Return Value	Description
false	Closed state
true	Enable state

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_getEnabledAxlibCtx

【Description】
Get the context structure of the enabled algorithm library.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
algo_type	Type of algorithm module to be operated	Input

【Return value】

Return Value	Description
NULL	Get failed
Not NULL	Successfully obtained

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

【Note】

-The returned algorithm context structure will be used by internal private functions. For user-defined algorithm libraries, this function should be called after rk_aiq_uapi_sysctl_enableAxlib, otherwise it will return NULL.

rk_aiq_uapi_sysctl_setCpsLtCfg

【Description】
Set the fill light control information.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
cfg	Fill light configuration structure pointer	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_getCpsLtInfo

【Description】
Obtain the fill light control information.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
info	Fill light configuration structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_queryCpsLtCap

【Description】
Query the support capability of the fill light.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
cap	Fill light support ability query structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_getBindedSnsEntNmByVd

【Description】
Query the sensor entity name corresponding to the video node.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
vd	video path, such as /dev/video20	input

【Return value】

Return Value	Description
sensor entity name	String pointer

【Note】

-The parameter must be the path of the ISPP scale node.

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h

-Library file: librkaiq.so

rk_aiq_uapi_sysctl_updateIq

【Description】

Dynamically update the currently used iq parameter file without stopping the data flow.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
iqfile	new iq file	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-iqfile needs to be a full path.
-Updating the iq parameter does not mean that the operating mode can be switched. If you need to switch between hdr and normal, it cannot be updated
iq file implementation; but the switching of some functions can be achieved through different configurations of iq parameters, such as day and night switching
Switching is achieved entirely through iq configuration.
-When switching iq, the configuration parameters in iq will override the user API settings. Such as AWB module, manual and automatic can be configured in iq
Mode, after executing this function, no matter what mode the current AWB is in, it will eventually be overwritten by the default configuration in the new iq.

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h

-Library file: librkaiq.so

rk_aiq_uapi_sysctl_setCrop

【Description】

Set crop parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
rect	crop parameter	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-The minimum crop resolution is 64x64.
-The resolution to be set must exist in the IQ XML effect file.
-Must be called before rk_aiq_uapi_sysctl_prepare.
-rect.width must be an integer multiple of 4. For different data formats, there are different requirements for the horizontal offset rect.left:
For raw8 and yuv422 formats, rect.left must be an integer multiple of 8.
In raw10 and raw12 formats, rect.left must be an integer multiple of 4.
In rgb888 format, rect.left must be an integer multiple of 24.

rk_aiq_uapi_sysctl_getCrop

【Description】

Get crop parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
rect	crop parameter structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

Type of Data

rk_aiq_working_mode_t

【Description】
AIQ pipeline working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_WORKING_MODE_NORMAL	Normal mode
RK_AIQ_WORKING_MODE_ISP_HDR2	Two-frame HDR mode
RK_AIQ_WORKING_MODE_ISP_HDR3	Three-frame HDR mode

【Precautions】

-You need to query the modes supported by the sensor and AIQ first. If the set mode does not support the setting, the setting will be invalid.

rk_aiq_static_info_t

【Description】
AIQ static information

【Definition】

```
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;
```

【Member】

Member Name	Description
sensor_info	Description of sensor name, supported resolution, etc.
lens_info	Lens Information
has_lens_vcm	Whether to bring vcm
has_fl	Whether with flash
fl_strth_adj_sup	Is it adjustable with flashlight
bool has_irc	With IR-CUT
bool fl_ir_strth_adj_sup	

rk_aiq_sensor_info_t

【Description】

sensor information

【Definition】

```
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;
```

【Member】

Member Name	Description
sensor_name	The name of the sensor
support_fmt	Supported formats
num	Number of supported formats
has_fl	Whether with flash
binded_strm_media_idx	Media node number mounted on the sensor

rk_aiq_module_id_t

【Description】

AIQ module ID

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_MODULE_DPCC	Bad pixel detection and correction
RK_MODULE_BLS	Black level
RK_MODULE_LSC	Lens shading correction
RK_MODULE_AWB_GAIN	White balance gain
RK_MODULE_CTK	Color correction
RK_MODULE_GOC	Gamma
RK_MODULE_SHARP	Sharpen
RK_MODULE_AE	Exposure
RK_MODULE_AWB	White balance
RK_MODULE_NR	Denoising
RK_MODULE_GIC	Green Balance
RK_MODULE_3DLUT	3DLUT
RK_MODULE_LDCH	LDCH
RK_MODULE_TNR	3D denoising
RK_MODULE_FEC	Fisheye correction

rk_aiq_cpsl_cfg_t

【Description】

Fill light setting information structure

【Definition】

```
typedef struct rk_aiq_cpsl_cfg_s {
    RKAiqOPMode_t mode;
    rk_aiq_cpsls_t lght_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;
```

【Member】

Member Name	Description
mode	Working mode
lght_src	Type of light source
gray_on	Whether to cut the screen to black and white after switching to night mode
sensitivity	Switching sensitivity in automatic mode, range [0,100]
sw_interval	Switching interval in automatic mode, in seconds
on	Whether to switch to night mode in manual mode
strength_led	LED light intensity in manual mode, range [0,100]
strength_ir	Infrared light intensity in manual mode, range [0,100]

rk_aiq_cpsl_info_t

【Description】

Fill light query information structure

【Definition】

```
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;
```

【Member】

Member Name	Description
mode	Working mode
lght_src	Type of light source
gray	Whether to cut the screen to black and white after switching to night mode
sensitivity	Switching sensitivity in automatic mode, range [0,100]
sw_interval	Switching interval in automatic mode, in seconds
on	Whether to switch to night mode in manual mode
strength_led	LED light intensity in manual mode, range [0,100]
strength_ir	Infrared light intensity in manual mode, range [0,100]

rk_aiq_cpsl_cap_t

【Description】

Supplement light support capability structure

【Definition】

```
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;
```

【Member】

Member Name	Description
supported_modes	Supported working modes
modes_num	Number of supported modes
gray	Whether to cut the screen to black and white after switching to night mode
supported_lght_src	Supported light sources
lght_src_num	Number of supported light sources
strength_led	LED intensity range
sensitivity	sensitivity range
strength_ir	Intensity range of infrared lamp

rk_aiq_rect_t

【Description】

Define crop parameter structure

【Definition】

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

【Member】

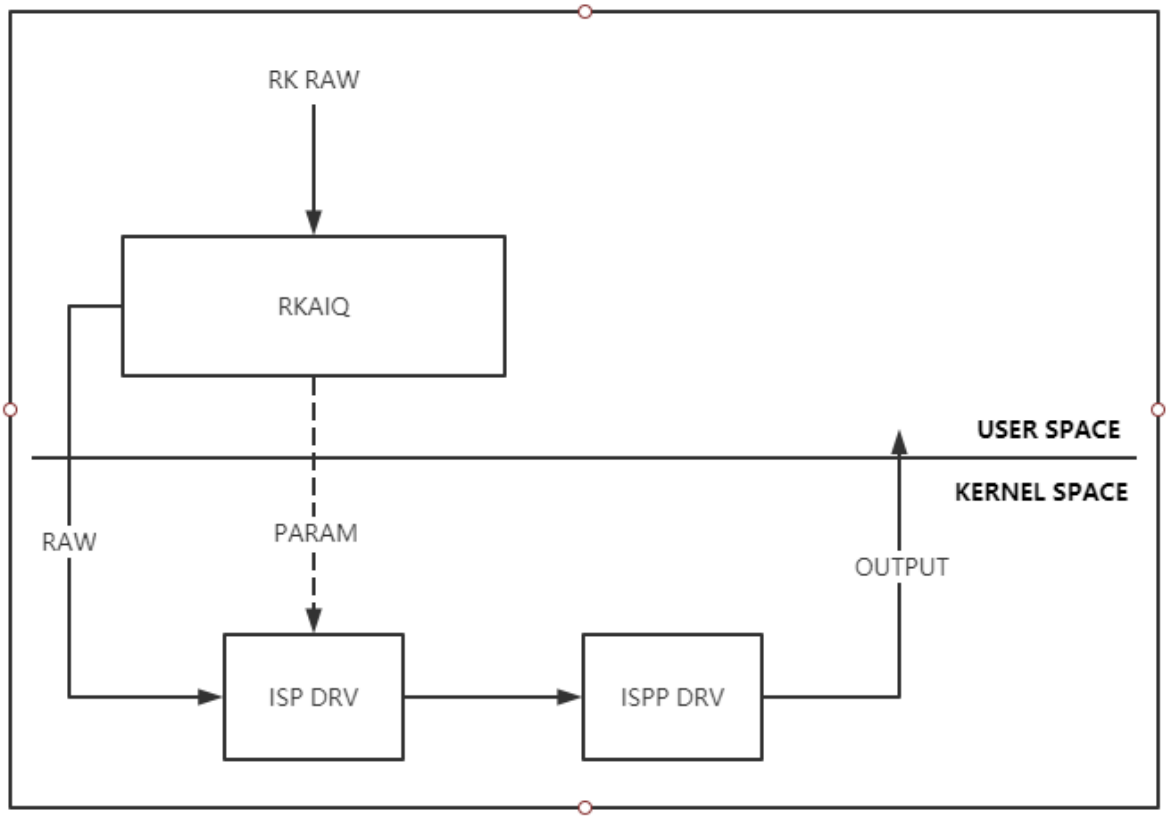
Member Name	Description
left	horizontal output offset
top	vertical output offset
width	horizontal output size
height	vertical output size

Offline frame processing

Overview

RKAIQ provides offline RAW frame processing function, that is, RK customized RAW format files are parsed by RKAIQ and then sent into ISP for processing, and output as an image that can be normally displayed.

Functional block diagram



Offline frame processing block diagram

Function description

- Support RK-RAW format file input. Calling the file input interface, the calling process will be blocked until the file is successfully processed and output .
- Support RK-RAW format buffer input and asynchronous working mode. Calling the buffer input interface, the calling process will not be blocked, and the callback function will be called after the buffer processing is completed (if there is a registered callback function).
- Support RK-RAW format buffer input, synchronous working mode. Calling the buffer input interface, the calling process will be blocked until the buffer processing is successfully processed and output .

RK-RAW format description

Please refer to "RK RAW File Format" documentation.

How to get RK RAW files

Please refer to the FAQ chapter.

Supported RAW format

Support raw8/raw10/raw12, BGGR/GBRG/GRBG/RGGB.

API Reference

rk_aiq_uapi_sysctl_prepareRkRaw

【Description】

Prepare the environment for RK Raw format data processing.

【Prototype】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
prop	RK Raw format property parameters	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface must be called before rk_aiq_uapi_sysctl_prepare.

rk_aiq_uapi_sysctl_enqueueRkRawBuf

【Description】

input RK Raw format buffer

【Prototype】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
rawdata	RK Raw format data buffer	input
sync	true: Synchronous mode. false: Asynchronous mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

If you need extra operation on the rawdata that has been processed in asynchronous mode, you can use the rk_aiq_uapi_sysctl_registRkRawCb interface to register the callback function, and the rawdata buffer will be passed into the callback function.

rk_aiq_uapi_sysctl_enqueueRkRawFile

【Description】

input RK Raw format file

【Prototype】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
path	RK Raw format file path	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-The interface is synchronous.

rk_aiq_uapi_sysctl_registRkRawCb

【Description】

Register callback function

【Prototype】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
callback	Callback function pointer	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is not necessary.
-If a callback is registered, the callback will only be called when the rk_aiq_uapi_sysctl_enqueueRkRawBuf interface is called in asynchronous mode and the Raw data processing is completed.

Data structure

rk_aiq_raw_prop_t

【Description】
RK Raw parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
frame_width	RK Raw image width
frame_height	RK Raw image height
format	RK Raw bayer pattern
rawbuf_type	RK Raw type

rk_aiq_rawbuf_type_t

【Description】
RK Raw type structure

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_RAW_ADDR	Indicates that the'Raw data' section of the input RK Raw format buffer stores the virtual address of the DMA BUF in this process, not the raw data itself.
RK_AIQ_RAW_FD	Indicates that the'Raw data' section in the input RK Raw format buffer stores the BUF fd in this process, not the RAW data itself.
RK_AIQ_RAW_DATA	Indicates that the raw image data is stored in the'Raw data' section of the input RK Raw format buffer.
RK_AIQ_RAW_FILE	Indicates that the input is a RK Raw format file.

Note

- Using RK Raw data processing function, when creating AIQ Context, the parameter sns_ent_name of the rk_aiq_uapi_sysctl_init interface must be "FakeCamera".
- The processing of raw data depends on IQ XML file, and make sure that the XML file is generated with the same resolution of the offline raw file. The IQ XML file must be named as FakeCamera.xml and placed under the loading path of the Aiq configuration files.
- If you do not connect to the real camera and want to use the RK Raw data processing function, then do not configure the input port in the ISP node in the kernel DTS file.
- If a real camera sensor is currently configured, suppose that the real camera sensor is connected to ISP0&ISPP0; if you want to use the RK Raw data processing function, you need to enable ISP1&ISPP1 in DTS , so that the created fake camera sensor can be used normally.
- If the real camera sensor and fake camera sensor need to be used concurrently and their resolutions are different, then the ISPP node in the DTS file SHOULD configure the maximum resolution information. For the specific configuration method, please refer to the document "Rockchip_Driver_Guide_ISP2x_CN".

Reference example

For how to use the offline frame processing API, please refer to rkisp_demo, the path is YOUR_SDK_DIR/external/camera_engine_rkaiq/rkisp_demo.

AE

Overview

The function of the AE module is to obtain the current image exposure according to the automatic metering system, and then automatically configure the lens aperture, sensor shutter and gain to obtain the best image quality.

Important concepts

-Exposure time: The time for the sensor to accumulate charge, which is the period from when the sensor pixel is exposed to when the charge is read out.

-Exposure gain: The total amplification factor of the output charge of the sensor. Generally, there are digital gain and analog gain. The noise introduced by analog gain will be slightly smaller, so analog gain is generally preferred.

-Aperture: Aperture is a mechanical device in the lens that can change the size of the clear aperture.

-Anti-flicker: The screen flicker caused by the mismatch between the power frequency of the lamp and the frame rate of the sensor, generally by limiting the exposure time and modifying the frame rate of the sensor to achieve the anti-flicker effect.

Function description

The AE module is composed of AE statistical information and the algorithm of AE control strategy.

Functional API Reference

rk_aiq_uapi_setExpMode

【Description】
Set the exposure mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setExpMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Exposure mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getExpMode

【Description】
Get the exposure mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getExpMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Exposure mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setAeMode

【Description】
Set AE working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setAeMode(const rk_aiq_sys_ctx_t* ctx, aeMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Working mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

When the exposure mode is switched to manual mode, the gain and exposure time adopt the initial values defined in the image effect file. If you need to set the exposure value while switching to manual mode, you can use the rk_aiq_uapi_setManualExp interface.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getAeMode

【Description】
Get the AE working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getAeMode(const rk_aiq_sys_ctx_t* ctx, aeMode_t *mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Working mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setManualExp

【Description】
Use manual exposure mode, and set the gain and exposure time.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
gain	exposure gain	input
time	Exposure time	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setExpGainRange

【Description】
Set the gain range.

【Syntax】

```
XCamReturn rk_aiq_uapi_setExpGainRange(const rk_aiq_sys_ctx_t* ctx, paRange_t
*gain);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
gain	Exposure gain range	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getExpGainRange

【Description】
Get the gain range.

【Syntax】

```
XCamReturn rk_aiq_uapi_getExpGainRange(const rk_aiq_sys_ctx_t* ctx, paRange_t
*gain);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
gain	Exposure gain range	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setExpTimeRange

【Description】
Set the exposure time range.

【Syntax】

```
XCamReturn rk_aiq_uapi_setExpTimeRange(const rk_aiq_sys_ctx_t* ctx, paRange_t
*time);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
time	Exposure time range	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getExpTimeRange

【Description】
Get the exposure time range.

【Syntax】

```
XCamReturn rk_aiq_uapi_getExpTimeRange(const rk_aiq_sys_ctx_t* ctx, paRange_t *time);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
time	Exposure time range	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setBLCMode

【Description】
Backlight compensation switch, area setting.

【Syntax】

```
XCamReturn rk_aiq_uapi_setBLCMode(const rk_aiq_sys_ctx_t* ctx, bool on, aeMeasAreaType_t areaType);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input
areaType	Compensation area selection	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only available in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setBLCStrength

【Description】
Set the dark area to enhance the intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setBLCStrength(const rk_aiq_sys_ctx_t* ctx, int
strength);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
strength	Increase strength, range [1,100]	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only available in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setHLCMode

【Description】
Strong light suppression switch.

【Syntax】

```
XCamReturn rk_aiq_uapi_setHLCMode(const rk_aiq_sys_ctx_t* ctx, bool on);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only available in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setHLCStrength

【Description】
Set the intensity of strong light suppression.

【Syntax】

```
XCamReturn rk_aiq_uapi_setHLCStrength(const rk_aiq_sys_ctx_t* ctx, int
strength);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
strength	Inhibition strength, range [1,100]	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only available in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setDarkAreaBoostStrth

【Description】

Set the dark area to enhance the intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setDarkAreaBoostStrth(const rk_aiq_sys_ctx_t* ctx,
unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Dark area boost strength, range [1,10]	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only valid in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getDarkAreaBoostStrth

【Description】

Get the current dark area boost intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setDarkAreaBoostStrth(const rk_aiq_sys_ctx_t* ctx,
unsigned int *level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Dark area boost intensity	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface is only valid in linear mode.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setAntiFlickerMode

【Description】
Set the anti-flash mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setAntiFlickerMode(const rk_aiq_sys_ctx_t* ctx,
antiFlickerMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Anti-flash mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getAntiFlickerMode

【Description】
Get anti-flash mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getAntiFlickerMode(const rk_aiq_sys_ctx_t* ctx,
antiFlickerMode_t *mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Anti-flash mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setExpPwrLineFreqMode

【Description】
Set the anti-flash frequency.

【Syntax】

```
XCamReturn rk_aiq_uapi_setExpPwrLineFreqMode(const rk_aiq_sys_ctx_t* ctx,
expPwrLineFreq_t freq);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
freq	Anti-flash frequency	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getExpPwrLineFreqMode

【Description】
Obtain the anti-flash frequency.

【Syntax】

```
XCamReturn rk_aiq_uapi_getExpPwrLineFreqMode(const rk_aiq_sys_ctx_t* ctx,
expPwrLineFreq_t *freq);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
freq	Anti-flash frequency	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Function-level API data types

opMode_t

【Description】
Define automatic manual mode

【Definition】

```
typedef enum opMode_e {
    OP_AUTO = 0,
    OP_MANUALl = 1,
    OP_INVAL
} opMode_t;
```

【Member】

Member Name	Description
OP_AUTO	Automatic mode
OP_MANUALl	Manual mode
OP_INVAL	Invalid value

aeMode_t

【Description】
Define AE working mode

【Definition】

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

【Member】

Member Name	Description
OP_AUTO	Automatic selection
AE_IRIS_PRIOR	Aperture Priority
AE_SHUTTER_PRIOR	Shutter priority

paRange_t

【Description】
Define the parameter range

【Definition】

```
typedef struct paRange_s {
    float max;
    float min;
} paRange_t;
```

【Member】

Member Name	Description
max	Upper limit value
min	Lower limit value

aeMeasAreaType_t

【Description】
Define AE measurement area type

【Definition】

```
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;
```

【Member】

Member Name	Description
AE_MEAS_AREA_AUTO	Automatic
AE_MEAS_AREA_UP	Upper area
AE_MEAS_AREA_BOTTOM	Lower area
AE_MEAS_AREA_LEFT	Left area
AE_MEAS_AREA_RIGHT	Right area
AE_MEAS_AREA_CENTER	Central area

expPwrLineFreq_t

【Description】
Define anti-flash frequency

【Definition】

```
typedef enum expPwrLineFreq_e {
        EXP_PWR_LINE_FREQ_DIS = 0,
        EXP_PWR_LINE_FREQ_50HZ = 1,
        EXP_PWR_LINE_FREQ_60HZ = 2,
} expPwrLineFreq_t;
```

【Member】

Member Name	Description
EXP_PWR_LINE_FREQ_DIS	
EXP_PWR_LINE_FREQ_50HZ	50 Hz
EXP_PWR_LINE_FREQ_60HZ	60 Hz

antiFlickerMode_t

【Description】
Define anti-flash mode

【Definition】

```
typedef enum antiFlickerMode_e {
    ANTIFLICKER_NORMAL_MODE = 0,
    ANTIFLICKER_AUTO_MODE = 1,
} antiFlickerMode_t;
```

【Member】

Member Name	Description
ANTIFLICKER_NORMAL_MODE	Normal mode
ANTIFLICKER_AUTO_MODE	Automatic selection mode

Module-level API reference

rk_aiq_user_api_ae_setExpSwAttr

【Description】

Set AE exposure software properties.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_setExpSwAttr(const rk_aiq_sys_ctx_t* ctx,
                                const Uapi_ExpSwAttr_t expSwAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
expSwAttr	AE public function control parameter structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

【Example】

- Set manual exposure properties

Exposure components include sensor exposure time, sensor exposure gain, isp digital gain, and aperture. After setting the manual mode, you also need to set the manual state of each exposure component (ManualGainEn, ManualTimeEn, ManualIspDgainEn, ManualIrisEn) and the corresponding manual value. In manual mode, at least one exposure component is required to be in manual state, otherwise it will report an error and exit. The value of each exposure component set in manual mode will be limited by the sensor and lens. If the set exposure component value exceeds the limit of the sensor and lens, the algorithm will automatically correct it.

```
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);
```

- Set auto exposure properties

```
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;/*must enable*/
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;
/*Generally, the automatic frame reduction mode is configured by the tuning
personnel in advance to configure the minimum frame rate and the gain value
corresponding to the switching frame rate.*/

//set ae delay
```

```
expSwAttr.stAuto.BlackDelayFrame = 2;
expSwAttr.stAuto.WhiteDelayFrame = 4;
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);
```

- Set aperture control parameters

```
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.PI
risAttr.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*/
```

- Set anti-flash function

```
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);
```

- Set AE weight

  Set the 5X5 weight, and expand the weight according to the actual hardware block specifications within the algorithm

```
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.DayGridWeights.uCoeff[i] = DayGridWeights[i];
}
ret = rk_aiq_user_api_ae_setExpSwAttr(ctx, expSwAttr);
```

Set the 15X15 weight, and the algorithm will compress the weight according to the actual block specifications of the hardware.

```
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,
    3,  5 ,7, 10, 11, 12, 13, 14, 13,12, 11, 10,  7,  5,  3,
    2,  4, 6,  9, 10, 11, 12, 13, 12,11, 10,  9,  6,  4,  2,
    2,  4, 6,  8,  9, 10, 11, 12, 11,10,  9,  8,  6,  4,  2,
```

```
    2,  3, 5,  7,  8,  9, 10, 11, 10, 9,  8,  7,  5,  3,  2,
    2,  3, 5,  7,  7,  8,  9, 10,  9, 8,  7,  7,  5,  3,  2,
    1,  2, 4,  6,  6,  7,  8,  9,  8, 7,  6,  6,  4,  2,  1,
    0,  1, 3,  5,  5,  6,  7,  8,  7, 6,  5,  5,  3,  1,  0,
    0,  1, 3,  5,  5,  6,  7,  8,  7, 6,  5,  5,  3,  1,  0
};
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,  4, 6,  8,  9, 10, 11, 12, 11,10,  9,  8,  6,  4,  2,
    2,  3, 5,  7,  8,  9, 10, 11, 10, 9,  8,  7,  5,  3,  2,
    2,  3, 5,  7,  7,  8,  9, 10,  9, 8,  7,  7,  5,  3,  2,
    1,  2, 4,  6,  6,  7,  8,  9,  8, 7,  6,  6,  4,  2,  1,
    0,  1, 3,  5,  5,  6,  7,  8,  7, 6,  5,  5,  3,  1,  0,
    0,  1, 3,  5,  5,  6,  7,  8,  7, 6,  5,  5,  3,  1,  0
};
memcpy(expSwAttr.stAdvanced.DayGridWeights,DayGridWeights,sizeof(expSwAttr.stAdv
anced.DayGridWeights));
memcpy(expSwAttr.stAdvanced.NightGridWeights,NightGridWeights,sizeof(expSwAttr.s
tAdvanced.DayGridWeights));

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

rk_aiq_user_api_ae_getExpSwAttr

【Description】

Obtain AE exposure software properties.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_getExpSwAttr(const rk_aiq_sys_ctx_t* ctx, Uapi_ExpSwAttr_t*
pExpSwAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pExpSwAttr	AE exposure software attribute structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setLinAeDayRouteAttr

【Description】

Set the AE's daytime scene exposure allocation strategy in linear mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
linAeRouteAttr	AE exposure allocation strategy structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

【Example】

```
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, 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

【Description】

Obtain the daytime scene exposure allocation strategy of AE in linear mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pLinAeRouteAttr	AE exposure allocation strategy structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setHdrAeDayRouteAttr

【Description】

Set the AE's daytime scene exposure allocation strategy in HDR mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
hdrAeRouteAttr	AE exposure allocation strategy structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

【Example】

```
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

【Description】

Obtain the daytime scene exposure allocation strategy of AE in HDR mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pHdrAeRouteAttr	AE exposure allocation strategy structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setLinAeNightRouteAttr

【Description】
Set the AE night scene exposure allocation strategy in linear mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
linAeRouteAttr	AE exposure allocation strategy structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_getLinAeNightRouteAttr

【Description】
Get the AE night scene exposure allocation strategy in linear mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pLinAeRouteAttr	AE exposure allocation strategy structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setHdrAeNightRouteAttr

【Description】
Set the AE night scene exposure allocation strategy in HDR mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
hdrAeRouteAttr	AE exposure allocation strategy structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_getHdrAeNightRouteAttr

【Description】
Get the AE night scene exposure allocation strategy in HDR mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pHdrAeRouteAttr	AE exposure allocation strategy structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_queryExpResInfo

【Description】
Obtain AE internal status information.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_queryExpResInfo(const rk_aiq_sys_ctx_t* ctx,
Uapi_ExpQueryInfo_t* pExpResInfo);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pExpResInfo	AE internal state information structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setLinExpAttr

【Description】
Set the AE linear mode exposure parameters.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_setLinExpAttr(const rk_aiq_sys_ctx_t* ctx, const
Uapi_LinExpAttr_t linExpAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
linExpAttr	AE exposure parameter structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_getLinExpAttr

【Description】
Get AE linear mode exposure parameters.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_getLinExpAttr(const rk_aiq_sys_ctx_t* ctx, Uapi_LinExpAttr_t*
pLinExpAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pLinExpAttr	AE exposure parameter structure pointer	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_setHdrExpAttr

【Description】
Set AE HDR mode exposure parameters.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_setHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, const
Uapi_HdrExpAttr_t hdrExpAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
hdrExpAttr	AE exposure parameter structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

rk_aiq_user_api_ae_getHdrExpAttr

【Description】
Get AE HDR mode exposure parameters.

【Syntax】

```
XCamReturn
rk_aiq_user_api_ae_getHdrExpAttr(const rk_aiq_sys_ctx_t* ctx, Uapi_HdrExpAttr_t*
pHdrExpAttr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
pHdrExpAttr	AE exposure parameter structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_ae.h, rk_aiq_uapi_ae_int.h
-Library file: librkaiq.so

Module-level API data types

Uapi_ExpSwAttr_t

【Description】

AE public function control parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
enable	Aec enable switch. 1. Turn on the Aec algorithm; 0, turn off the Aec algorithm, and the exposure remains at the value before turning off.
RawStatsMode	Aec module brightness statistics mode. There are four modes: Y/R/G/B, the default is Y mode.
HistStatsMode	Aec module histogram statistics mode. There are five modes: Y/R/G/B/RGB, and the default is Y mode.
YRangeMode	Range mode of Y channel of Aec module. The two modes are FULL/LIMITED, and the default is FULL mode.
AecRunInterval	Ae algorithm run interval, the value range is [0,255], and the default value is 0. When the value is 0, AE is run every frame; when the value is 1, AE is run every other frame; and so on.
AecOpType	Exposure mode, divided into automatic exposure (AUTO) mode / manual (MANUAL) exposure mode. The default is AUTO mode. Manual exposure mode needs to cooperate with stManual to set the manual exposure value.
DayGridWeights	The weight of the window (histogram) in day mode, containing 5*5 array elements
NightGridWeights	Window (histogram) weight in night mode, including 5*5 array elements
DNTrigger	AEC parameter switch enable. 0: AEC parameters are not switched with the day and night modes, and the day parameters are fixed by default; 1: AEC parameters are switched with the day and night modes.
stIris	Iris control parameter structure
stAntiFlicker	Anti-power frequency flicker parameter structure
stInitExp	Exposure initial value structure
stAuto	Automatic exposure parameter structure
stManual	Manual exposure parameter structure
stAdvanced	Use parameter structure first

【Related data type】

-Uapi_ExpSwAttr_Advanced_t

-Uapi_IrisAttr_t
-Uapi_AntiFlicker_t
-Uapi_AeAttr_t
-Uapi_MeAttr_t
-Uapi_ExpInitExp_t

Uapi_ExpSwAttr_Advanced_t

【Description】

Prefer parameter structure

【Definition】

```
#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;
```

【Precautions】

-A set of AE weights is defined in the Uapi_ExpSwAttr_t structure, and the number of weights is 5X5. The algorithm internally expands the weights according to the actual hardware configuration weights. If you need to set a higher-precision weight, you can use a set of AE weights defined in Uapi_ExpSwAttr_Advanced_t, and the number of weights is 15X15. In the algorithm, the weights are compressed according to the actual hardware configuration weights.
-To use a set of AE weights defined in Uapi_ExpSwAttr_Advanced_t, you need to turn on enable and set it to 1.

Uapi_IrisAttr_t

【Description】

Aperture control parameters

【Definition】

```
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;
```

【Member】

Member name	Description
Enable	Enable auto iris function
IrisType	Aperture type, P (ie P-iris aperture) or DC (ie DC-iris aperture)
PIrisAttr	P Aperture Attribute Parameters
DCIrisAttr	DC Aperture Attribute Parameters

CalibDb_PIris_Attr_t

【Description】

P aperture attribute parameters

【Definition】

```
#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;
```

【Member】

Member name	Description
TotalStep	The total number of steps of the P-iris stepper motor, the specific size is related to the P-iris lens.
EffcStep	The number of steps available for the P-iris stepper motor, the specific size is related to the P-iris lens.
ZeroIsMax	Whether P-iris stepping motor step0 corresponds to the maximum aperture position, the specific value is related to the P-iris lens. The value is 0, which means that when the position of the stepper motor is step0, the iris is opened to the minimum; the value is 1, which means that when the position of the stepper motor is step0, the iris is opened to the maximum.
StepTable	The mapping table of P-iris stepper motor position and iris equivalent gain. The specific value is related to the P-iris lens.

CalibDb_DCIris_Attr_t

【Description】

DC aperture properties

【Definition】

```
typedef struct CalibDb_DCIris_Attr_s {
    float Kp;
    float Ki;
    float Kd;
    int MinPwmDuty;
    int MaxPwmDuty;
    int OpenPwmDuty;
    int ClosePwmDuty;
} CalibDb_DCIris_Attr_t;
```

【Member】

Member name	Description
Kp	The proportional coefficient is used to limit the opening and closing speed of the aperture when the aperture changes drastically. The larger the value, the slower the aperture opening and closing speed when the light changes drastically. If the value is too large, the braking will be advanced during the adjustment process, resulting in too long adjustment time; if the value is too small, the braking during the adjustment process will lag behind, resulting in increased overshoot. The reasonable setting of this value is related to the characteristics of the DC-iris lens and circuit. The recommended value is 0.5. The value range is [0, 1].
Ki	Integral coefficient, used to adjust the opening and closing speed of the iris, the greater the value, the greater the opening and closing speed of the iris. If the value is too large, overshoot is likely to cause oscillation; if the value is too small, the iris adjustment speed is slow and the environment brightness changes drastically, it is prone to oscillation. The recommended value is 0.2. The value range is [0, 1].
Kd	Differential coefficient, used to adjust the opening and closing speed of the iris, the larger the value, the greater the opening and closing speed of the iris. The recommended value is 0.3. The value range is [0, 1].
MinPwmDuty	The minimum PWM duty cycle, the specific size is related to the DC-iris lens and circuit characteristics, and the unit is %. The smaller the value, the faster the supported aperture closing speed, but it is easy to cause the aperture to oscillate. The value range is [0,100], and the default value is 0.
MaxPwmDuty	The maximum PWM duty cycle, the specific size is related to the DC-iris lens and circuit characteristics, and the unit is %. The larger the value, the faster the supported aperture opening speed. If the value is too small, it may cause the aperture control to exit before the maximum aperture. The value range is [0,100], and the default value is 100.
OpenPwmDuty	The PWM duty cycle threshold when the iris is open. When the PWM duty cycle of the iris is higher than (excluding) OpenPwmDuty, the iris is in the open state. The specific size is related to the DC-iris lens, and the unit is %.
ClosePwmDuty	The PWM duty cycle threshold when the iris is closed. When the PWM duty cycle of the iris is less than (excluding) ClosePwmDuty, the iris is closed. The specific size is related to the DC-iris lens, and the unit is %.

【Precautions】

- When the auto iris function is turned off, the DC-iris iris will be opened to the maximum by default; for the P-iris iris, it will be opened to the stepper motor position corresponding to the maximum iris by default. If you want to change the above aperture position, you can go to the AecInitValue module to modify InitPIrisGainValue and InitDCIrisDutyValue.
- The basic control process of the automatic iris Airis algorithm is as follows:

For the DC-iris lens, Airis controls the aperture size of the DC-iris lens according to the deviation between the current brightness and the target brightness. When the exposure reaches the minimum value and the current brightness exceeds the target brightness tolerance range, the AE control will be exited, the exposure time and exposure gain will be fixed, and the AIris control range will be entered. If the current image brightness is stable and the PWM duty value of DC-iris is greater than OpenPwmDuty, it is considered that the current aperture has reached the maximum, exit AIris aperture control, and the control will be transferred to AE.

For P-iris lenses, the iris is controlled through the AecRoute module. The aperture size of the P-iris lens is converted into an equivalent gain, which participates in the exposure decomposition calculation.

- P-iris stepper motor position and iris equivalent gain mapping table StepTable is generally made according to the corresponding relationship between the step motor position and the iris aperture provided by the lens manufacturer. The control of P-iris is controlled by the AecRoute module of AE, which converts the size of the iris aperture into an equivalent gain. Therefore, the iris control of P-iris is required to have good linearity. The value range of the equivalent gain is [1,1024], the equivalent gain 1024 represents F1.0, the equivalent gain 512 represents F1.4, and so on, the equivalent gain 1 represents F32.0. When making the table, it is necessary to convert the aperture aperture corresponding to the position of the stepper motor into an equivalent gain, fill it in the StepTable, and fill it in in the order of increasing the position of the stepper motor (ie step0, step1...stepN).
- TotalStep represents the total number of steps of the P-iris stepper motor, the specific size is related to the P-iris lens. EffcStep represents the number of steps available for the P-iris stepper motor, which is generally required to be less than TotalStep. Because the position is close to the closing end of the iris, the value of the equivalent gain error is relatively large, and oscillation is prone to occur during the adjustment of the iris, so the step position near the closing end of the iris is usually not used.
- Table 4-1 is the correspondence table of P-iris stepper motor position, iris aperture and equivalent gain. Take this table as an example to explain how to set StepTable. The corresponding relationship between the step motor position step and the aperture area of the 1-2 and 4-5 columns in Table 4-1 is provided by a certain lens manufacturer. The stepper motor of this P-iris lens has a total number of steps of 81. The corresponding aperture at step0 is the largest, and the nominal maximum number of apertures is 1.4. When the aperture is 1.4, the corresponding equivalent gain is 512, so the corresponding equivalent gain at step 0 is 512. The equivalent gain corresponding to other aperture areas, here is step 3 as an example, the calculation method is as follows: the aperture area of step 3 is 195.869, the corresponding equivalent gain = 512* (195.869/201.062) = 499 (rounded). By analogy, the equivalent gain values corresponding to other stepper motor positions can also be calculated based on this. It can be seen from Table 1-1 that when the stepping motor is close to the closed end, the corresponding aperture area is very small, and the difference from the largest aperture area can be several thousand times, and the corresponding equivalent gain value error is large, so it is recommended to be close to the aperture. Do not use the stepping motor position at the closed end to avoid exposure oscillation due to errors. The equivalent gain corresponding to each stepper motor position in the table is filled into StepTable in the order of stepper motor position increment (ie step0, step1...stepN).
- The values of OpenPwmDuty and ClosePwmDuty of DC-iris need to be measured, and their specific values are related to the DC-iris lens. For some lenses, when the PWM duty cycle is greater than OpenPwmDuty, the iris is opened; when the PWM duty cycle is less than OpenPwmDuty, the iris is closed; when the PWM duty cycle is greater than or equal to ClosePwmDuty and less than or equal to OpenPwmDuty, the iris is stable The current position and the value in this interval are all HoldValue. There are also some lenses, there is

only a threshold of the aperture switch, that is, when the PWM duty cycle is greater than the threshold, the iris performs an opening operation; when the PWM duty cycle is less than the threshold, the iris performs a closing operation; when the PWM duty cycle When it is equal to the threshold, the iris is stable at the current position, and the threshold is HoldValue. At this time, you can set ClosePwmDuty = OpenPwmDuty = HoldValue.

- The manual mode parameter setting of the aperture is consistent with the manual mode of exposure. When you need to use the manual iris function, you need to set AecOpType to manual mode and enable the ManualIrisEn parameter in the AecManualCtrl module. When IrisType is P-iris, only PirisGainValue is valid; when IrisType is P-iris, only DCIrisValue is valid.

Table 4-1 Correspondence table of P-iris stepper motor position, iris aperture and equivalent gain

Step	Aperture area(mm^2^)	Equivalent gain	Step	Aperture area(mm^2^)	Equivalent gain
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	Aperture area(mm^2^)	Equivalent gain	Step	Aperture area(mm^2^)	Equivalent gain
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

- **【Description】**
  Define anti-flash properties

  【Definition】

```
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;
```

  【Member】

Member Name	Description
enable	Power frequency anti-flash function enable state, 0: disable anti-flash function; 1: open anti-flash function

Frequency | Anti-flicker frequency properties, including 3 frame rates, namely: AEC_FLICKER_FREQUENCY_OFF (valid when the anti-flicker enable bit is set to 0), AEC_FLICKER_FREQUENCY_50HZ, AEC_FLICKER_FREQUENCY_60HZ, the default is AEC_FLICKER_FREQUENCY_50 Hz. |

| Mode | Anti-flash mode, including two anti-flash modes: AEC_ANTIFLICKER_NORMAL_MODE (normal anti-flash mode), AEC_ANTIFLICKER_AUTO_MODE (automatic anti-flash mode) |

  【Precautions】

    - To turn off the anti-flash function, you need to set the anti-flash enable bit enable to 0, and the algorithm will automatically configure Frequency=AEC_FLICKER_FREQUENCY_OFF; if the anti-flicker enable bit is set to 1, the anti-flash frequency is configured as AEC_FLICKER_FREQUENCY_OFF, and the frequency value will be invalid. Use the default value AEC_FLICKER_FREQUENCY_50HZ.
- AEC_ANTIFLICKER_NORMAL_MODE is a normal anti-flash mode. The exposure time can be adjusted according to the brightness. The minimum exposure time is fixed at 1/120 sec (60Hz) or 1/100 sec (50Hz), which is not limited by the minimum exposure time.

Lighted environment: The exposure time can be matched with the frequency of the light source, which can prevent the image from flickering.

High-brightness environment: The higher the brightness, the shortest exposure time is required. The minimum exposure time of the normal anti-flash mode cannot match the frequency of the light source, resulting in overexposure.

- AEC_ANTIFLICKER_AUTO_MODE is an automatic anti-flash mode, the exposure time can be adjusted according to the brightness, and the minimum exposure time can reach the minimum exposure time of the sensor. The difference from the normal anti-flash mode is mainly reflected in the high-brightness environment. High-brightness environment: The minimum exposure time can reach the minimum exposure time of the sensor, which can effectively inhibit overexposure, but the anti-flashing failure at this time.

Uapi_AeAttr_t

【Description】

Define AE auto exposure properties

【Definition】

```
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;
```

【Member】

When setting the AE parameter range through the api, you need to set SetAeRangeEn to 1, otherwise the system auto exposure parameter range is used by default. The setting of the exposure parameter range is divided into two sets of stLinAeRange and stHdrAeRange according to the different exposure modes. Among them, stHdrAeRange supports the setting of the automatic exposure parameter range of each frame.

Member Name	Description
stAeSpeed	Automatic exposure adjustment speed
BlackDelayFrame	Automatic exposure delay property, when the image brightness is lower than the target value and exceeds the BlackDelayFrame frame, Ae starts to adjust
WhiteDelayFrame	Automatic exposure delay property, when the image brightness is higher than the target value and exceeds the WhiteDelayFrame frame, Ae starts to adjust
SetAeRangeEn	Whether to set the auto exposure parameter range
stLinAeRange	Linear mode auto exposure range structure
stHdrAeRange	Hdr mode auto exposure range structure
stFrmRate	Automatic exposure frame rate mode structure, fixed frame rate mode or automatic frame rate reduction mode

【Precautions】

When setting the AE parameter range through the api, you need to set SetAeRangeEn to 1, otherwise the system auto exposure parameter range is used by default. The setting of the exposure parameter range is divided into two sets of stLinAeRange and stHdrAeRange according to the different exposure modes. Among them, stHdrAeRange supports the setting of the automatic exposure parameter range of each frame.

The AE parameter range queried through api is the parameter range actually used after verification and correction inside the algorithm. When the AE parameter range set by the api exceeds the parameter range limited by the sensor, the parameter range limited by the sensor will be used. For the parameter range restricted by sensor, see the sensorinfo parameter of the AE module in the Rockchip_Tuning_Guide_ISP2x_CN_v1.x.x.pdf document.

CalibDb_AeSpeed_t

【Description】

Define AE conditional speed attributes

【Definition】

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

【Member】

Member name	Description
DampOver	The environment brightness is stable, the corresponding exposure adjustment speed when the image brightness is higher than the target value, the value range is [0,1]
DampUnder	The environment brightness is stable, the corresponding exposure adjustment speed when the image brightness is lower than the target value, the value range [0,1]
DampDark2Bright	Sudden changes in environmental brightness, the corresponding exposure adjustment speed from dark to bright, value range [0,1]
DampBright2Dark	Abrupt environment brightness, the corresponding exposure adjustment speed from bright to dark, value range [0,1]

CalibDb_AeRange_t

【Description】
Define AE parameter range

【Definition】

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

【Member】

Member Name	Description
Min	Lower limit value
Max	Upper limit value

CalibDb_LinAeRange_t

【Description】
Define the parameter range of AE linear mode

【Definition】

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

【Member】

Member Name	Description
stExpTimeRange	Exposure time range, set the maximum and minimum values, in milliseconds
stGainRange	Sensor analog gain range, set the maximum and minimum values
stIspDGainRange	ISP digital gain range, set the maximum and minimum values
stPIrisRange	Iris equivalent gain range, only supports P-Iris aperture size control

【Precautions】

-When the maximum/minimum value of each exposure component is the default value of 0, the set exposure component range will not take effect. The actual maximum/minimum value of each exposure component is determined by the minimum and maximum values of the AecRoute exposure decomposition route node.
-When the maximum/minimum value of each exposure component is not 0, the set exposure component range takes effect. When the set exposure component range does not exceed the limit of the sensor or ISP, the actual maximum/minimum value of each exposure component is based on the set exposure component The range shall prevail, and the AecRoute exposure decomposition route shall be corrected. The maximum/minimum value of the node is changed to the maximum/minimum value of the exposure component; if the limit of the sensor or ISP is exceeded, the actual maximum/minimum value of each exposure component is The maximum and minimum values of AecRoute exposure decomposition route nodes shall prevail.

CalibDb_HdrAeRange_t

【Description】
Define the parameter range of AE HDR mode

【Definition】

```
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;
```

【Member】

Member Name	Description
stExpTimeRange	Exposure time range, set the maximum and minimum values, in seconds, the array 0/1/2 is short frame, medium frame, and long frame respectively.
stGainRange	Sensor simulation gain range, set the maximum and minimum values, the array 0/1/2 is short frame, medium frame, and long frame respectively.
stIspDGainRange	ISP digital gain range, set the maximum and minimum values, the array 0/1/2 is short frame, medium frame and long frame respectively.
stPIrisRange	Equivalent gain value range of the iris, set the maximum and minimum values

【Precautions】

-stExpTimeRange, stGainRange, stIspDGainRange are predefined as an array containing 3 members. In the 2-frame HDR mode, only members 0 and 1 are valid, which respectively indicate the exposure component range corresponding to the short and long frames; in the 3-frame HDR mode, 0-2 are all valid, which respectively indicate the exposure components corresponding to the short, medium, and long frames range.
-When the maximum/minimum value of each exposure component is the default value of 0, the set exposure component range is not effective. At this time, the actual maximum/minimum value of each exposure component is determined by the minimum and maximum value of the exposure decomposition route node corrected by the algorithm.
-When the maximum/minimum value of each exposure component is not 0, the set exposure component range takes effect. When the set exposure component range does not exceed the limit of the sensor or ISP, the actual maximum/minimum value of each exposure component is based on the set exposure component The range shall prevail, and the exposure decomposition route shall be corrected. The maximum/minimum value of the node is changed to the maximum/minimum value of the exposure component; if the limit of the sensor or ISP is exceeded, the actual maximum/minimum value of each exposure component will be exposed The maximum and minimum values of the decomposition route nodes shall prevail.

CalibDb_AeFrmRateAttr_t

【Description】
Define AE frame rate attributes

【Definition】

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

【Member】

Member Name	Description
isFpsFix	Enable the auto exposure fixed frame rate mode, the default value is FALSE, that is, the automatic frame rate reduction mode is adopted; when the value is TRUE, it indicates the fixed frame rate mode.
FpsValue	It is only valid when the frame rate is fixed. When the default value is 0, the driver's default frame rate is always used; when the value is not 0, the set frame rate value is used.

Uapi_MeAttr_t

【Description】

Manual exposure parameter setting is divided into two sets of parameters: LinearAE and HdrAE according to the exposure mode.

【Definition】

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

[Related data types]

-CalibDb_LinMeAttr_t

-CalibDb_HdrMeAttr_t

CalibDb_LinMeAttr_t

【Description】

LinearAE's manual exposure control parameters

【Definition】

```
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;
```

【Member】

Member name	Description
ManualTimeEn	Manual exposure time enable, the default value is 1
ManualGainEn	Manual sensor gain enable, the default value is 1
ManualIspDgainEn	Manual ISP digital gain enable, the default value is 1
ManualIrisEn	Manual iris enable, the default value is 1
TimeValue	Manual exposure time value, in s, the parameter value is limited by sensor
GainValue	Manual sensor gain value, parameter value is limited by sensor
IspDGainValue	Manual ISP digital gain value, parameter value is limited by ISP
PIrisGainValue	Manual P iris equivalent gain value, the parameter value is limited by the P iris device, the value range is [1, 1024]
DCIrisValue	DC iris HoldValue value, the parameter value is related to the DC iris device, the value range is [0, 100]

【Precautions】

-This module is only valid when AeOptype = MANUAL. When ManualTimeEn, ManualGainEn, ManualIspDgainEn, and ManualPIrisEn are all 1, it is manual mode; as long as any one of the above four is disabled, it is semi-automatic mode; the above four are all 0, it is equivalent to automatic mode, and the system will report an error.
-In manual/semi-manual mode, manual exposure time and gain will be limited by the maximum/minimum exposure time and gain in automatic mode. After the range of auto exposure limit is exceeded, the maximum/minimum value in auto mode will be used instead.
-ManualIrisEn, manual iris enable. When the aperture type is P aperture, only PirisGainValue is valid; when the aperture type is DC aperture, only DCIrisValue is valid.
-DCIrisValue, directly set the PWM duty cycle value of the motor in manual mode, the value range is [0, 100]. In manual mode, if it is set to the HoldValue value (that is, the value in the range from ClosePwmDuty to OpenPwmDuty in AecIrisCtrl), the DC iris aperture remains at the current size; if the set value is greater than OpenPwmDuty, the iris is in the open state, and the greater the value, the opening speed The larger the value is; if the set value is less than ClosePwmDuty, the iris is closed. The smaller the value, the higher the closing speed.
-RV1109/RV1126 currently does not support ISP digital gain, so ManualIspDgainEn and IspDGainValue are invalid.

CalibDb_HdrMeAttr_t

【Description】

Manual exposure control parameters for HdrAE

【Definition】

```
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;
```

【member】

The variable meaning is the same as CalibDb_LinMeAttr_t

【Precautions】

-This module is only valid when AeOptype = MANUAL. When ManualTimeEn, ManualGainEn, ManualIspDgainEn, and ManualPIrisEn are all 1, it is manual mode; as long as any one of the above four is disabled, it is semi-automatic mode; the above four are all 0, it is equivalent to automatic mode, and the system will report an error.
-TimeValue/GainValue/IspDGainValue are all arrays with 3 members. In Hdr 2 frame mode, the members of array [0-1] are valid, representing short and long frames respectively; in Hdr 3 frame mode, the members of array [0-2] are all valid, corresponding to short, medium, and long frames respectively.
-In manual/semi-manual mode, manual exposure time and gain will be limited by the maximum/minimum exposure time and gain in automatic mode. After the range of auto exposure limit is exceeded, the maximum/minimum value in auto mode will be used instead.
-ManualIrisEn, manual iris enable. When the aperture type is P aperture, only PirisGainValue is valid; when the aperture type is DC aperture, only DCIrisValue is valid.
-DCIrisValue, directly set the PWM duty cycle value of the motor in manual mode, the value range is [0, 100]. In manual mode, if it is set to the HoldValue value (that is, the value in the range from ClosePwmDuty to OpenPwmDuty in AecIrisCtrl), the DC iris aperture remains at the current size; if the set value is greater than OpenPwmDuty, the iris is in the open state, and the greater the value, the opening speed The larger the value is; if the set value is less than ClosePwmDuty, the iris is closed. The smaller the value, the higher the closing speed.
-RV1109/RV1126 currently does not support ISP digital gain, so ManualIspDgainEn and IspDGainValue are invalid.

Uapi_ExpInitExp_t

【Description】

Exposure initial value parameter

【Definition】

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

[Related data types]

-CalibDb_LinExpInitExp_t

-CalibDb_HdrExpInitExp_t

CalibDb_LinExpInitExp_t

【Description】

Linear exposure initial exposure parameters

【Definition】

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

【member】

Member name	Description
InitTimeValue	Initial sensor exposure time value
InitGainValue	Initial sensor exposure gain value
InitIspDGainValue	Initial ISP digital gain value
InitPIrisGainValue	Initial P-iris equivalent gain value
InitDCIrisDutyValue	Initial DC Iris Duty Cycle Value

CalibDb_HdrExpInitExp_t

【Description】

Hdr exposure initial exposure parameters

【Definition】

```
typedef struct CalibDb_HdrExpInitExp_s {
    Cam3x1FloatMatrix_t InitTimeValue;
    Cam3x1FloatMatrix_t InitGainValue;
    Cam3x1FloatMatrix_t InitIspDGainValue;
    int InitPIrisGainValue;
    int InitDCIrisDutyValue;
    int array_size;
} CalibDb_HdrExpInitExp_t;
```

【Member】

The member description is the same as CalibDb_LinExpInitExp_t

【Precautions】

-InitTimeValue/InitGainValue/InitIspDGainValue are all arrays with 3 members. In Hdr 2 frame mode, the members of array [0-1] are valid, representing short and long frames respectively; in Hdr 3 frame mode, the members of array [0-2] are all valid, corresponding to short, medium, and long frames respectively.

Uapi_LinAeRouteAttr_t

【Description】

Define AE linear strategy attributes

【Definition】

```
#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;
```

【Member】

Member Name	Description
name	Mode name, divided into day mode and night mode
TimeDot	Sensor exposure time node, the unit is s
GainDot	sensor exposure gain node
IspgainDot	Isp Digital Gain Node
PIrisGainDot	Iris Equivalent Gain Node
array_size	Number of exposure decomposition nodes

【Precautions】

- The number of nodes in the exposure decomposition curve is at most 10, and it is recommended to set at least 6 nodes
- The exposure of the node is the product of the exposure time, sensor gain, ISP digital gain, and iris equivalent gain. The node exposure must increase monotonically, that is, the exposure of the next node must be greater than the exposure of the previous node. The first node has the smallest exposure, and the second node has the largest exposure.
- The unit of the exposure time component in the node is seconds, and the minimum allowed value is 0. The actual minimum exposure time code will be internally corrected according to the sensor limit.
- The aperture component only supports P-Iris, not DC-Iris. The P-iris equivalent gain component is only valid when the Airis automatic iris function is enabled, otherwise the default iris is fixed to the initial value. The calculation of P-iris equivalent gain is detailed in the AecIrisCtrl module.

- The set exposure decomposition route node is not the final exposure decomposition route. The actual maximum/minimum value of each exposure component in the system is determined by the exposure decomposition node and the maximum/minimum value of the manually configured exposure component. First calibrate the maximum/minimum value of the node of the exposure decomposition route. When the maximum/minimum value of the node does not exceed the limit of the sensor or isp, the maximum/minimum value of the node does not change; when the maximum/minimum value of the node exceeds the sensor or isp When limiting, the maximum/minimum value of the node is subject to the limit of the sensor or isp. When the maximum/minimum value of the manually configured exposure component is 0, the final effective exposure decomposition route is based on the decomposition route of the first correction; when the maximum/minimum value of the manually configured exposure component is not 0, and the maximum value is set /When the small value does not exceed the limit of the sensor or isp, the exposure decomposition route is corrected for the second time, and the maximum/minimum value of the node is subject to the manually set range; if the maximum/minimum value of the set exposure component exceeds the limit of the sensor or isp When, the maximum/minimum value of the node of the exposure component of the exposure decomposition route is subject to the first correction result.
- If the exposure of adjacent nodes increases, only one exposure component should increase, and the other exposure components should be fixed. The added weight determines the allocation strategy of the route. For example, if the gain component increases and other components are fixed, then the allocation strategy for this route is gain priority.
- RV1109/RV1126 currently does not support ISP digital gain, so IspgainDot is invalid.

【Related data type】

-AEC_ROUTE_MAX_NODES
-AecExpSeparateName_t

AecExpSeparateName_t

【Description】
Define the name string type

【Definition】

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

Uapi_HdrAeRouteAttr_t

【Description】
Define AE HDR policy attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
name	Mode name, divided into day mode and night mode
HdrTimeDot	Sensor exposure time node, the unit is s, and the array 0/1/2 is short frame, medium frame, and long frame respectively
HdrGainDot	sensor exposure gain node, array 0/1/2 are short frame, medium frame, long frame respectively
HdrIspDGainDot	Isp digital gain node, array 0/1/2 are short frame, medium frame, long frame
PIrisDot	Aperture node, array 0/1/2 are short frame, medium frame, long frame respectively
array_size	Number of exposure decomposition nodes, array 0/1/2 are short frame, medium frame, and long frame respectively

【Precautions】

- The number of nodes in the exposure decomposition curve is at most 10, and it is recommended to set at least 6 nodes
  -HdrTimeDot, HdrGainDot, HdrIspDGainDot are defined as a two-dimensional array, the first dimension represents the number of frames, and the second dimension represents the exposure component node corresponding to each frame. In the 2-frame HDR mode, only 0 and 1 in the first dimension are valid, indicating short and long frames respectively; in the 3-frame HDR mode, 0-2 in the first dimension are all valid, indicating short, medium, and long frames respectively.
- The exposure of the node is the product of the exposure time, sensor gain, ISP digital gain, and iris equivalent gain. The node exposure must increase monotonically, that is, the exposure of the next node must be greater than the exposure of the previous node. The first node has the smallest exposure, and the second node has the largest exposure.
- The unit of the exposure time component in the node is seconds, and the minimum allowed value is 0. The actual minimum exposure time code will be internally corrected according to the sensor limit.
- The aperture component only supports P-Iris, not DC-Iris. The P-iris equivalent gain component is only valid when the Airis automatic iris function is enabled, otherwise the default iris is fixed to the initial value. The calculation of P-iris equivalent gain is detailed in the AecIrisCtrl module.
- The set exposure decomposition route node is not the final exposure decomposition route. The actual maximum/minimum value of each exposure component in the system is determined by the exposure decomposition node and the maximum/minimum value of the manually configured exposure component. First calibrate the maximum/minimum value of the node of the exposure decomposition route. When the maximum/minimum value of the node does not exceed the limit of the sensor or isp, the maximum/minimum value of the node does not change; when the maximum/minimum value of the node exceeds the sensor or isp When limiting, the maximum/minimum value of the node is subject to the limit of the sensor or isp. When the maximum/minimum value of the manually configured exposure component is 0, the final effective exposure decomposition route is based on the decomposition route of the first correction; when the maximum/minimum value of the manually configured exposure component is not 0, and the maximum value is set /When

the small value does not exceed the limit of the sensor or isp, the exposure decomposition route is corrected for the second time, and the maximum/minimum value of the node is subject to the manually set range; if the maximum/minimum value of the set exposure component exceeds the limit of the sensor or isp When, the maximum/minimum value of the node of the exposure component of the exposure decomposition route is subject to the first correction result.

- If the exposure of adjacent nodes increases, only one exposure component should increase, and the other exposure components should be fixed. The added weight determines the allocation strategy of the route. For example, if the gain component increases and other components are fixed, then the allocation strategy for this route is gain priority.
- RV1109/RV1126 currently does not support ISP digital gain, so HdrIspDGainDot is invalid.

【Related definitions】

-AEC_ROUTE_MAX_NODES
-AecExpSeparateName_t

Uapi_LinExpAttr_t

【Description】

Define AE linear exposure debugging parameters

【Definition】

```
typedef struct CalibDb_LinearAE_Attr_s {
    uint8_t RawStatsEn;
    float SetPoint;
    float NightSetPoint;
    float ToleranceIn;
    float ToleranceOut;
    float Evbias;
    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;
```

【Member】

Member Name	Description
RawStatsEn	Linear exposure uses the Raw domain statistical brightness function to enable
SetPoint	In day mode, the target brightness value during automatic exposure adjustment, the value range is [0,255]
NightSetPoint	In night mode, the target brightness value during automatic exposure adjustment, the value range is [0,255]
EvBias	The deviation percentage of the exposure during automatic exposure adjustment, the unit is %, and the value range is [-200,+200]
ToleranceIn/Out	Tolerance of screen brightness during automatic exposure adjustment. The unit is %, and the value range is [0,100]
StrategyMode	Automatic exposure strategy mode, high light priority or low light priority
DySetPointEn	Dynamic target brightness value enable switch for automatic exposure adjustment. Enable the dynamic target brightness value, when enable=TRUE, the dynamic target brightness value is used, when enable=FALSE, the fixed target brightness value is used, and the dynamic target brightness value is invalid
DySetpoint	Dynamic target brightness value attribute of automatic exposure adjustment, which dynamically changes with exposure, divided into normal mode and night/IR mode
BackLightConf	Backlight Compensation Function
OverExpCtrl	Strong Light Suppression Module

【Precautions】

-SetPoint represents the target brightness value in normal mode, that is, the target brightness value used when night mode or IR mode is not turned on. NightSetPoint represents the target brightness value in night mode or IR mode. Night mode and IR mode cannot be turned on at the same time. The opening of IR mode requires hardware support.
-When DySetPointEn = TRUE, the fixed target brightness values SetPoint and NightSetPoint are invalid, and the dynamic target brightness value is used; when DySetPointEn = FALSE, the dynamic target brightness value is invalid, and all scenes always use the same target brightness.
-Exposure deviation EvBias, used to fine-tune the (fixed/dynamic) target brightness value (SetPoint/IRSetPoint) in special scenes. The actual effective target brightness is (SetPoint/IRSetPoint)*(1+EvBias/100)
-The tolerance of the automatic exposure screen brightness is Tolerance. When the automatic exposure converges, the screen brightness value B should be in the range of [real effective target brightness X (1-Tolerance/100), real effective target brightness X (1+Tolerance/100)] Inside.
-StrategyMode is temporarily invalid.

[Related data types]

-CalibDb_AecDynamicSetpoint_t
-CalibDb_AecBacklight_t
-CalibDb_AecOverExpCtrl_t

CalibDb_AecDynamicSetpoint_t

【Description】

Define AE dynamic target value

【Definition】

```
#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;
```

【Member】

Member Name	Description
name	Mode name, divided into day mode and night mode
ExpValue	Dynamic exposure node attribute, the node value is the ratio of the current exposure to the maximum exposure, set in increasing order, the value range is [0,1]
DySetpoint	Dynamic target brightness value node attribute, the node value changes dynamically with the exposure, the larger the exposure node value, the smaller the target brightness node value, and it corresponds to the exposure node one-to-one
array_size	Number of nodes for dynamic target brightness value

Uapi_HdrExpAttr_t

【Description】

Define AE HDR exposure debugging parameters

【Definition】

```
typedef struct CalibDb_HdrAE_Attr_s {
    float ToleranceIn;
    float ToleranceOut;
    CalibDb_AeHdrLongFrmMode_t LongfrmMode;
    CalibDb_AeStrategyMode_t StrategyMode;
    float Evbias;
    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;
```

【Member】

Member Name	Description
ToleranceIn/Out	Tolerance of screen brightness during automatic exposure adjustment. The unit is %, and the value range is [0,100]
LongfrmMode	Long Frame Mode Parameters
StrategyMode	Automatic exposure strategy mode, high light priority or low light priority
Evbias	The deviation percentage of the exposure during automatic exposure adjustment, the unit is %, and the value range is [-200,+200]
ExpRatioType	Exposure ratio mode, only valid in Hdr mode multi-frame synthesis, AUTO: Automatically calculate the exposure ratio of long and short frames according to the scene; FIX: Use a fixed exposure ratio for long and short frames
RatioExpDot	Represents the exposure amount node, according to the exposure amount, dynamically set the fixed value of the exposure ratio or the maximum value of the exposure ratio, and the two correspond one to one. The number of nodes is fixed at 6
M2SRatioFix	The number of nodes is fixed at 6. When ExpRatioType is AUTO, it is invalid. When ExpRatioType is FIX, it indicates the exposure ratio of the medium frame to the short frame, which corresponds to the exposure node RatioExpDot one-to-one
L2MRatioFix	The number of nodes is fixed at 6. When ExpRatioType is AUTO, it is invalid. When ExpRatioType is FIX, it indicates the exposure ratio of the long frame to the medium frame, and corresponds to the exposure node RatioExpDot one-to-one. Hdr is invalid when 2 frames are combined, and effective when 3 frames are combined
M2SRatioMax	The number of nodes is fixed at 6. When ExpRatioType is AUTO, it indicates the dynamic maximum value of the exposure ratio between the medium frame and the short frame, which corresponds to the exposure node RatioExpDot one-to-one. When ExpRatioType is FIX, invalid
L2MRatioMax	The number of nodes is fixed at 6. When ExpRatioType is AUTO, it indicates the dynamic maximum value of the exposure ratio between the long frame and the medium frame, which corresponds to the exposure node RatioExpDot one-to-one. Hdr is invalid when 2 frames are combined, and valid when 3 frames are combined. When ExpRatioType is FIX, invalid
LframeCtrl	Long frame control parameters
MframeCtrl	Medium frame control parameters, only valid in HDR 3 frame mode
SframeCtrl	Short frame control parameters

【Precautions】

-ExpRatioType is AUTO, using automatic exposure ratio mode. In 2 frame mode, the maximum exposure ratio of long and short frames is limited by M2SratioMax; in 3 frame mode, the maximum exposure ratio of short and medium frames is limited by M2SratioMax, and the maximum exposure ratio of long and medium frames is limited by L2MratioMax. The minimum exposure ratio is unlimited and must not be less than 1. ExpRatioType is FIX, using a fixed exposure ratio mode. In 2 frame mode, the exposure ratio of long and short frames is M2SRatioFix; in 3 frame mode, the exposure ratio of short and medium frames is M2SRatioFix, and the exposure ratio of long and medium frames is L2MratioFix.

【Related data type】

-CalibDb_LFrameCtrl_t
-CalibDb_MFrameCtrl_t
-CalibDb_SFrameCtrl_t

Uapi_ExpQueryInfo_t

【Description】
Define AE exposure parameter query

【Definition】

```
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;
```

【Member】

Member Name	Description
IsConverged	Is Auto Exposure Converged
IsExpMax	Whether the ISP exposure reaches the maximum value
LumaDeviation	In linear mode, the difference between the target value of AEC and the actual screen brightness. This value is positive, indicating that the actual brightness is greater than the target brightness; the value is negative, indicating that the actual brightness is less than the target brightness; the value is 0, indicating the actual The screen brightness is within the tolerance range of the target value.
HdrLumaDeviation	In Hdr mode, the difference between the brightness target value of each frame and the actual screen brightness. This value is positive, indicating that the actual brightness is greater than the target brightness; if the value is negative, the actual brightness is less than the target brightness; the value is 0, Indicates that the actual screen brightness is within the tolerance range of the target value. In the 2-frame mode, only 0 and 1 are valid, indicating short and long frames respectively; in the 3-frame mode, 0-2 are all valid, indicating short, medium, and long frames respectively.
MeanLuma	In linear mode, the average brightness of the current picture.
HdrMeanLuma	In HDR mode, the average brightness of the current picture in each frame. In the 2-frame mode, only members 0 and 1 of the array are valid, representing short and long frames respectively; in the 3-frame mode, members of the array 0-2 are all valid, representing short, medium, and long frames respectively.
CurExpInfo	Current exposure parameters, including sensor exposure time and sensor exposure gain value. According to the exposure mode, there are 2 sets of exposure parameters, LinearExp and HdrExp[3].
Piris	Aperture
LinePeriodsPerField	Sensor's VTS
PixelPeriodsPerLine	Sensor's HTS
PixelClockFreqMHZ	sensor's pixel clock frequency (unit: megahertz)

Common problem location and debug method

If there are problems such as screen brightness flickering, overshoot, and brightness that does not meet expectations, it is recommended to capture the AE LOG of the problematic scene for analysis, which helps to quickly locate the problem and improve work efficiency.

Exposure statistics synchronization test function

Before calibrating the ISP module, the driver or tuning personnel is required to fill in the SensorInfo and System module parameters in the IQ XML for debugging. These two modules involve the setting of exposure parameters, such as setting errors, exposure errors, flickering, etc. may occur. It is recommended that after configuring the sensorinfo parameters, enable the AecSyncTest function for self-testing. For the meaning of sensorinfo and System parameters, please refer to "Rockchip_Tuning_Guide_ISP2x".

The AecSyncTest function sets N groups of different exposure values cyclically, which can test whether the sensor's exposure time, exposure gain, and the number of effective frames of DCG switching are correct. It can also be used to test the linearity of exposure to confirm the register value conversion of exposure time and exposure gain. Whether the formula and related parameters are correct.

The function parameters of AecSyncTest are introduced as follows:

【Description】

The synchronization test function of exposure and statistics supports the setting of N groups of different exposure values cyclically according to the number of frames at a given interval, which is used for debugging and verifying the effective number of frames of exposure components (exposure time, exposure gain) and whether the sensor exposure parameter settings are correct .

【member】

Member name	Description
Enable	Enable the synchronization test function of exposure and statistics
IntervalFrm	Exposure switching interval frame number
AlterExp	Exposure switching parameters

-AlterExp

According to the different modes, there are two sets of parameters, LinearAE and HdrAE.

Member name	Description
TimeValue	Exposure time value
GainValue	Exposure gain value
IspDgainValue	Isp digital gain value
DcgMode	Dcg Mode Value
PIrisGainValue	P-iris equivalent gain value

If the parameters are set correctly, the LOG example is as follows (only the key information in the LOG is intercepted). The red frame shows the exposure switching position. It can be seen that there is no sudden change in the brightness and it matches the exposure value, there is no delay or linearity in advance. At this time, it can be basically judged that the sensorinfo and System parameters are set correctly.

```
>>> 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
```

Flickers when exposure changes

There may be several reasons for flickering when exposure changes:

(1) The number of frames at the effective moment of gain and time in the EXP_DELAY module is wrong. Common LOG examples are as follows (only the key LOG lines of each frame are intercepted):

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

The red frame marks the location of the brightness error. It can be seen that as the exposure increases or decreases, the corresponding brightness changes in the opposite trend to the exposure. Through observation, it can be found that the sudden changes in brightness all occur in the second frame after the exposure time and the exposure gain are changed at the same time. The change trend of the brightness is consistent with the change trend of the exposure time, so it can be judged that the effective frame number of gain and time is wrong, and the changes of the exposure time and the exposure gain do not take effect at the same time, resulting in the brightness and exposure mismatch. This problem can be solved by modifying the number of effective frames for gain and time.

(2) The flicker caused by the AE follow-up module, the common LOG example is as follows (only the key LOG line of each frame is intercepted):

```
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
```

It can be seen from the LOG that during the decreasing process of SMeanLuma and MMeanLuma on the left, the brightness output by the TmoMeanluma module has undergone abrupt changes. When this happens, it is necessary to debug in the TMO module.

AWB

Overview

The function of the AWB module is to make the appearance of obtained images under different colored light sources to be identical to the canonical image under a white light source, by correcting the images with the white balance gain of the red, green, blue channel .

Important concepts

-Color temperature: The color temperature of a light source is the temperature of an ideal black-body radiator that radiates light of a color comparable to that of the light source.Color temperature is a characteristic of visible light . Color temperature is conventionally expressed in kelvins, using the symbol K, a unit of measure for absolute temperature
 When all are the same, the temperature of the black body at this time is called the color temperature of the light source.
-White balance: Under light sources with different color temperatures, the white response in the sensor will be blue or red. White balance calculation
 The method adjusts the intensity of the three color channels R, G, B to make the white truly appear.

Function description

The AWB module consists of two parts: WB information statistics and AWB strategy control algorithm.

Functional API Reference

rk_aiq_uapi_setWBMode

【Description】
Set the white balance working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setWBMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	White balance working mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-If set to manual mode, the white balance gain value is the gain value at present. If you need to switch the manual mode and set the gain value at the same time, you should use the rk_aiq_uapi_setMWBGain interface.

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getWBMode

【Description】
Get the white balance working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getWBMode(const rk_aiq_sys_ctx_t* ctx, opMode_t *mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	White balance working mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_lockAWB

【Description】
Lock the current white balance parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_unlockAWB

【Description】
Unlock the locked white balance parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMWBScene

【Description】
Switch to the manual white balance mode and set the white balance scene.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
scene	White balance scene	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMWBScene

【Description】
Get the white balance scene.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
scene	White balance scene	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMWBGain

【Description】
Switch to the manual white balance mode and set manual white balance value at the same time

【Syntax】

```
XCamReturn rk_aiq_uapi_setMWBGain(const rk_aiq_sys_ctx_t* ctx, rk_aiq_wb_gain_t
*gain);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
gain	white balance gain coefficient	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMWBGain

【Description】
Get the white balance gain coefficient.

【Syntax】

```
XCamReturn rk_aiq_uapi_getMWBGain(const rk_aiq_sys_ctx_t* ctx, rk_aiq_wb_gain_t
*gain);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
gain	White balance gain factor	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMWBCT

【Description】

Switch to the manual white balance mode and set the white balance color temperature parameter.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
ct	White balance color temperature parameter	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMWBCT

【Description】

Get the white balance gain coefficient.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
ct	White balance color temperature	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

- 【Requirement】

  -Header file: rk_aiq_user_api_imgproc.h
  -Library file: librkaiq.so

Function-level API data types

rk_aiq_wb_op_mode_t

【Description】
Define white balance working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_WB_MODE_MANUAL	Manual mode
RK_AIQ_WB_MODE_AUTO	Automatic mode

rk_aiq_wb_scene_t

See above.

rk_aiq_wb_gain_t

See above.

rk_aiq_wb_cct_t

See above.

Module-level API reference

rk_aiq_user_api_awb_SetAttrib

【Description】
Get white balance properties.

【Syntax】

```
XCamReturn
rk_aiq_user_api_awb_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_wb_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of white balance	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_awb.h, rk_aiq_uapi_awb_int.h
-Library file: librkaiq.so

rk_aiq_user_api_awb_GetAttrib

【Description】

Get white balance attributes.

【Syntax】

```
XCamReturn
rk_aiq_user_api_awb_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_wb_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of white balance	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_awb.h, rk_aiq_uapi_awb_int.h
-Library file: librkaiq.so

rk_aiq_user_api_awb_GetCCT

【Description】

Get the white balance color temperature parameter.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
cct	White balance color temperature parameter	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_awb.h, rk_aiq_uapi_awb_int.h
-Library file: librkaiq.so

rk_aiq_user_api_awb_QueryWBInfo

【Description】
Obtain the white balance gain coefficient and color temperature.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
wb_querry_info	Color related status parameters	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_awb.h, rk_aiq_uapi_awb_int.h
-Library file: librkaiq.so

Module-level API data types

rk_aiq_wb_op_mode_t

【Description】
Define white balance working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_WB_MODE_MANUAL	White balance manual mode
RK_AIQ_WB_MODE_AUTO	White balance auto mode

rk_aiq_wb_mwb_mode_t

【Description】
Define the type of manual white balance mode

【Definition】

```
typedef enum rk_aiq_wb_mwb_mode_e {
    RK_AIQ_MWB_MODE_INVAILD = 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;
```

【Member】

Member Name	Description
RK_AIQ_MWB_MODE_CCT	Color temperature
RK_AIQ_MWB_MODE_WBGAIN	Gain factor
RK_AIQ_MWB_MODE_SCENE	Scene

rk_aiq_wb_gain_t

【Description】
Define white balance gain parameters

【Definition】

```
typedef struct rk_aiq_wb_gain_s {
    float rgain;
    float grgain;
    float gbgain;
    float bgain;
} rk_aiq_wb_gain_t;
```

【Member】

Member Name	Description
rgain	R channel gain
grgain	G channel gain
gbgain	GB channel gain
bgain	B channel gain

rk_aiq_wb_scene_t

【Description】
Define white balance gain parameters

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_WBCT_INCANDESCENT	Incandescent lamp
RK_AIQ_WBCT_FLUORESCENT	Fluorescent lamp
RK_AIQ_WBCT_WARM_FLUORESCENT	Warm fluorescent lamp
RK_AIQ_WBCT_DAYLIGHT	Daylight
RK_AIQ_WBCT_CLOUDY_DAYLIGHT	Cloudy
RK_AIQ_WBCT_TWILIGHT	Twilight
RK_AIQ_WBCT_SHADE	Shadow

rk_aiq_wb_cct_t

【Description】

Define white balance gain parameters

【Definition】

```
typedef struct rk_aiq_wb_cct_s {
    float CCT;
    float CCRI;
} rk_aiq_wb_cct_t;
```

【Member】

Member Name	Description
CCT	Correlated Color Temperature
CCRI	Correlated Color Rendering Index

rk_aiq_wb_mwb_attrib_t

【Description】

Define manual white balance attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
mode	Mode selection
para	Parameter configuration corresponding to the mode

rk_aiq_wb_awb_attrib_t

【Description】

Define auto white balance attributes

【Definition】

```
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 wbGainClipEn;
} rk_aiq_wb_awb_attrib_t;
```

【Member】

Member Name	Description
algMethod	Useless
tolerance	tolerance
runInterval	Run frame interval
sceneAdjustEn	Scene Adjustment
colorBalanceEn	Useless
cagaEn	Enable that the white balance corrected image is as consistent as possible with the color appearance perceived by the human eye
wbGainAdjustEn	Hue adjustment enable
wbGainDaylightClipEn	Outdoor minimum color temperature limit enable
wbGainClipEn	Color temperature range limit enable

rk_aiq_wb_attrib_t

【Description】

Define white balance properties

【Definition】

```
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;
```

【Member】

Member Name	Description
byPass	Skip module processing
mode	Working Mode selection
stManual	Parameter configuration in manual mode
stAuto	Parameter configuration in automatic mode

rk_aiq_wb_querry_info_t

【Description】
Define white balance query information

【Definition】

```
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;
```

【Member】

Member Name	Description
gain	gain
cctGloabl	Global color temperature parameter
awbConverged	Is the white balance converged

AF

Overview

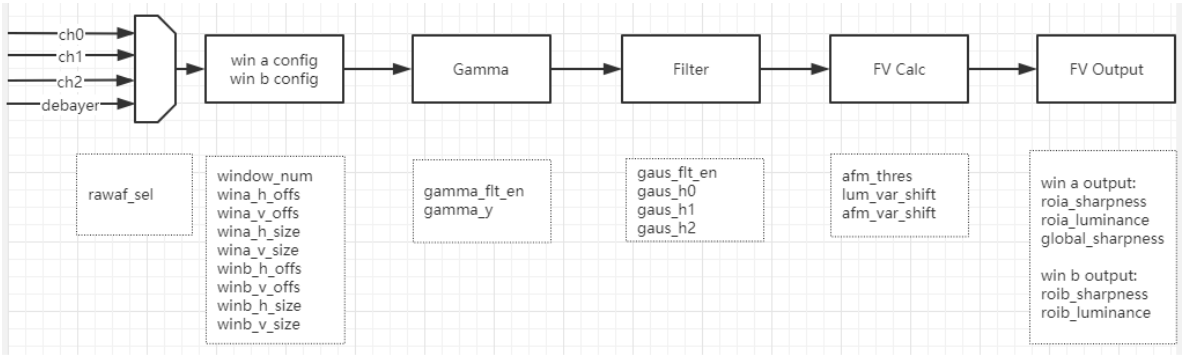
The function of the AF module refers to the process of adjusting the camera lens to make the image of the object clear.

Important concepts

-VCM: Voice coil motor

Function description

The AF module is composed of AF information statistics and AF control algorithm.
The following figure shows the block diagram of the AF information statistics module



Among them, ch0/ch1/ch2 correspond to S/M/L frames in hdr mode, and debayer is a merged frame in hdr mode.
FV Calc uses fixed operators to calculate the horizontal and vertical edge information of the image.

The output of window A mainly contains 15*15 FV information. The luminance information is not provided. You need to set the AE window to be consistent with AF, and then directly use the AE 15x15 luminance statistics.

The output of windows B mainly contains an FV information and luminance information.

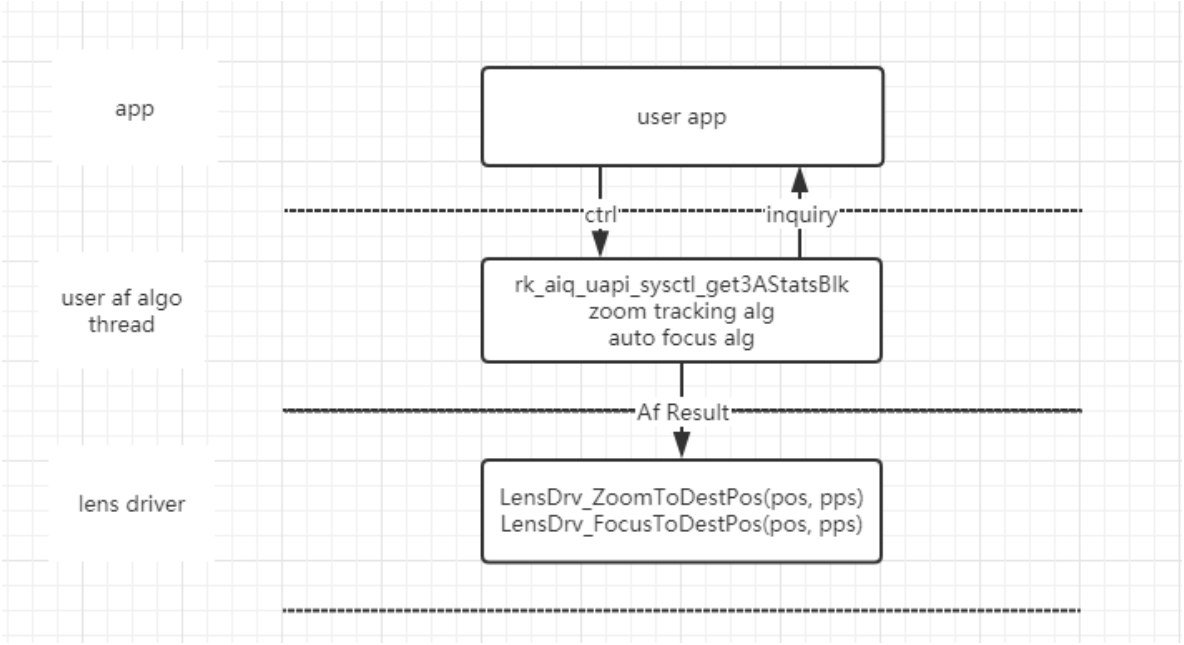
Porting user AF algorithm

When users implement the AF algorithm, they can refer to the figure above to set the rk_aiq_af_algo_meas_t structure in rk_aiq_af_attrib_t, and then call rk_aiq_user_api_af_SetAttrib to configure the af hardware statistics module.

After that, you can start a separate thread to call rk_aiq_uapi_sysctl_get3AStatsBlk to obtain af/ae/awb statistics. The corresponding structure of af statistics is rk_aiq_af_algo_stat_t.

For more information, please refer to the rk_aiq_uapi_sysctl_get3AStatsBlk function description.

The user AF algorithm refers to the af statistics to complete the calculation of the target zoom and focus positions, and then calls the Lens Driver to move the lens to the specified position.



Reference Code

// 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);
}
```

}

Functional API Reference

rk_aiq_uapi_setFocusMeasCfg

【Description】
Configure AF hardware statistics module, which is generally called when the af algorithm is implemented by itself.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
meascfg	AF information statistics configuration	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setFocusMode

【Description】 Configure the working mode of the af algorithm.

【Syntax】

```
XCamReturn rk_aiq_uapi_setFocusMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Focus mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getFocusMode

【Description】 Get the current working mode of the af algorithm.

【Syntax】

```
XCamReturn rk_aiq_uapi_getFocusMode(const rk_aiq_sys_ctx_t* ctx, opMode_t
*mode);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
mode	Focus mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setFixedModeCode

【Description】 Set the lens position in manual focus mode.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
code	Focus code value, the value range is 0-64	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getFixedModeCode

【Description】 Get the current lens position.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
code	Focus code value	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setFocusWin

【Description】 Set the af hardware statistics window.

【Syntax】

```
XCamReturn rk_aiq_uapi_setFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_t *rect);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
rect	Focus window	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getFocusWin

【Description】 Set the af hardware statistics window.

【Syntax】

```
XCamReturn rk_aiq_uapi_getFocusWin(const rk_aiq_sys_ctx_t* ctx, paRect_t *rect);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
rect	Focus window	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_lockFocus

【Description】 Auto focus is locked and focus pauses.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_unlockFocus

【Description】 Auto focus is unlocked, and focus continues.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_oneshotFocus

[Description] Trigger a single focus search. The af will not tracking after the operation.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_manualTrigerFocus

[Description] Manually trigger the focus. The af will tracking after the operation.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_trackingFocus

【Description】

Config the af to enter the tracking state.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setVcmCfg

[Description] Configure the vcm driver.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
cfg	vcm configuration	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getVcmCfg

[Description] Get the current vcm driver configuration.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
cfg	vcm configuration	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setOpZoomPosition

【Description】 Set the position of the zoom lens.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
pos	zoom position, the value range is 0-2000	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getOpZoomPosition

【Description】 Set the position of the zoom lens.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
pos	zoom position, value range 0-2000	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Function-level API data types

rk_aiq_af_algo_meas_t

【Description】
Define AF hardware statistics configuration

【Definition】

```
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;
```

【Member】

Member Name	Description
contrast_af_en	enable AF information statistics, 0 is disable, 1 is enable.
rawaf_sel	Select the source channel for AF hardware statistics, the value range is 0-3, which corresponds to the long/medium/short/composite video frame channel selection of hdr mode. Generally, the medium frame channel is selected for AF, and the non-hdr mode is set to 0, and the hdr mode is set to 1
window_num	The number of effective windows. When window_num is 1, wina (main window) is effective; when window_num is 2, wina (main window) and winb (independent window) are effective
wina_h_offs	The horizontal coordinate of the first pixel in the upper left corner of wina (main window), the value must be greater than or equal to 2
wina_v_offs	The vertical coordinate of the first pixel in the upper left corner of wina (main window), the value must be greater than or equal to 1
wina_h_size	window width of wina (main window), this value must be smaller than image width-2-wina_h_offs and must be a multiple of 15;
wina_v_size	window height of wina (main window), this value must be smaller than image height-2-wina_v_offs and must be a multiple of 15;
winb_h_offs	The horizontal coordinate of the first pixel in the upper left corner of winb (independent window), the value must be greater than or equal to 2
winb_v_offs	The vertical coordinate of the first pixel in the upper left corner of winb (independent window), the value must be greater than or equal to 1
winb_h_size	The window width of winb (independent window), this value must be smaller than the image width -2-wina_h_offs
winb_v_size	The window height of winb (independent window), this value must be smaller than the image height-2-wina_v_offs
gamma_flt_en	gamma function enable, 0 is disable, 1 is enable.
gamma_y	The y value of gamma table, the value range is 0-1023; the x coordinate segment is 0 to 1023: 16 16 16 16 32 32 32 32 64 64 64 128 128 128 128 128
gaus_flt_en	Gaussian filter module enable, 0 is disable, 1 is enable.
gaus_h0	Gaussian filter 3x3 coefficient h0, the middle coefficient value of the 3x3 coefficient, the value range is 0-255
gaus_h1	Gaussian filter 3x3 coefficient h1, the four coefficient values of the upper, lower, left and right of the middle coefficient of the 3x3 coefficient, the value range is 0-127
gaus_h2	Gaussian filter 3x3 coefficient h2, other coefficient values in the 3x3 coefficient, the value range is 0-127

Member Name	Description
line_en	Currently not effective yet
line_num	Currently not effective yet
afm_thres	AF statistical sharpness threshold. When the statistical value is less than this value, the statistical value is changed to 0, which can reduce the influence of noise. The value range is 0-0xFFFF
lum_var_shift	The shit bit value of the luminance value will shift the luminance value to the right according to this value to avoid overflow of the obtained sharpness value. The value range is 0-7
afm_var_shift	The shit bit value of the sharpness value. The sharpness value will be shifted to the right according to this value to avoid overflow of the obtained sharpness value. The value range is 0-7

opMode_t

【Description】
Define AF work mode

【Definition】

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

【Member】

Member Name	Description
OP_AUTO	Auto focus mode, run built-in AF algorithm
OP_MANUAL	Manual focus mode, stop the built-in AF algorithm
OP_SEMI_AUTO	Semi-auto focus mode, perform auto focus once after calling set zoom position api

rk_aiq_lens_vcmcfg

【Description】 Define the configuration of the vcm driver

【Definition】

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

【Member】

Member Name	Description
start_ma	vcm start current in mA
rated_ma	vcm cut-off current in mA
step_mode	The step control and step period values used in linear slope control mode

Module level API Reference

rk_aiq_user_api_af_SetAttrib

【Description】
Set the AF attribute

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	the AF attribute	input

【Return value】

Return Value	Description
0	Success
Other values	Failure, see error code table for details

【Requirement】

- Header files：rk_aiq_user_api_awb.h、rk_aiq_uapi_awb_int.h
- Library files：librkaiq.so

rk_aiq_user_api_af_GetAttrib

【Description】
Get the AF attribute

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	the AF attribute	output

【Return value】

Return Value	Description
0	Success
Other values	Failure, see error code table for details

【Requirement】

- Header files：rk_aiq_user_api_af.h、rk_aiq_uapi_af_int.h
- Library files：librkaiq.so

Module-level API data types

RKAIQ_AF_MODE

【Description】
Define the af working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RKAIQ_AF_MODE_NOT_SET	Focus mode is not set
RKAIQ_AF_MODE_AUTO	Auto focus mode
RKAIQ_AF_MODE_MACRO	Macro focus Mode
RKAIQ_AF_MODE_INFINITY	Infinity focus mode
RKAIQ_AF_MODE_FIXED	Fixed focus mode
RKAIQ_AF_MODE_EDOF	Depth of Field Focus Mode
RKAIQ_AF_MODE_CONTINUOUS_VIDEO	Smooth continuous focus mode
RKAIQ_AF_MODE_CONTINUOUS_PICTURE	Fast continuous focus mode
RKAIQ_AF_MODE_ONESHOT_AFTER_ZOOM	Semi-automatic focus mode, after calling set zoom position, automatically trigger a focus

rk_aiq_af_attrib_t

【Description】

Focus configuration information

【Definition】

```
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;
```

【Member】

Member Name	Description
AfMode	Focus Mode
contrast_af	Enable contrast focus
laser_af	Enable laser focus
pdaf	Enable phase focus
h_offs	The start horizontal coordinate of the focus window
v_offs	Starting vertical coordinate of the focus window
h_size	Focus window width
v_size	Focus window height
fixedModeDefCode	Focus code value in fixed focus mode
macroModeDefCode	End code value in macro focus mode, the focus range is 0-this value
infinityModeDefCode	The initial code value in infinity focus mode, the focus range is -64
manual_meascfg	Custom focus statistics configuration

other instructions

VCM driver verification

1) Whether the starting current and termination current of the vcm drive are set correctly,

First, obtain the relevant information about the starting current and the termination current from the module factory.

Next, select several modules to confirm whether the information is correct, the method is as follows:

Set the starting current and ending current in dts to the maximum range that VCM can support.

The focus mode is switched to manual mode, and the vcm position is gradually adjusted from 64. When the lens starts to move and the far focus object (above 10 meters) is clear, the current position value is recorded.

The initial current is (vcm_max_mA-vcm_min_mA) * (64-curPos) / 64.

Continue to adjust the vcm position, when the close focus object (10cm or 20cm) is clear, record the current position value,

The termination current is (vcm_max_mA-vcm_min_mA) * (64-curPos) / 64.

2) Move vcm in different directions, and whether the final stay position is stable.

Switch the focus mode to manual mode, select a position, move from 0 to this position and from 64 to this position, compare whether the two image sharpness is the same, and whether the AF statistics are close.

Normal mode improves focus speed

When the sensor is not directly connected to the isp through vicap and uses non-HDR mode, you can set export normal_no_read_back=1 before the application starts to enable the pass-through mode,

In this way, the sensor data does not need to be stored in the ddr and then read back to the isp, which reduces the effective AF statistical value acquisition time and improves the focusing speed.

IMGPROC

Overview

imgproc refers to the module that affects the image effect.

FEC

Function description

The squint distortion, pincushion, barrel distortion, etc. caused by the distortion of the optical system and the electronic scanning system may cause the geometric characteristics of the image to be distorted. Image distortion correction is a process of transforming a distorted image into an ideal image in a certain transformation manner.

This module corrects image distortion in the x and y directions.

Important concepts

-Distortion actually refers to the distortion of the photographed object relative to the object itself.

Functional API Reference

rk_aiq_uapi_setFecEn

【Description】 Enable fisheye distortion correction function.

【Syntax】

```
XCamReturn rk_aiq_uapi_setFecEn(const rk_aiq_sys_ctx_t* ctx, bool en);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
en	enable	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface can only be called before rk_aiq_uapi_sysctl_prepare, that is, it cannot support dynamic switch while AIQ is running. If you need dynamic switch distortion correction effect, you can use the rk_aiq_uapi_setFecBypass interface.
-After enabling fisheye distortion correction, DDR bandwidth and CPU load will increase, which may affect the camera's capture frame rate.

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

rk_aiq_uapi_setFecCorrectDirection

【Description】 Set the direction of fisheye distortion correction.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
direction	Correction direction	Input

【Definition】

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

【Member】

Member Name	Description
FEC_CORRECT_DIRECTION_X	Only correct the X direction
FEC_CORRECT_DIRECTION_Y	Only correct the Y direction
FEC_CORRECT_DIRECTION_XY	Correction in both XY directions

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This interface can only be called before rk_aiq_uapi_sysctl_prepare, that is, it cannot be call while AIQ is running.

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

rk_aiq_uapi_setFecBypass

【Description】 Bypass fisheye distortion correction function. The data also be processd by FEC, but correction strength is equivalent to no correction.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
bypass	Correction effect switch	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

rk_aiq_uapi_setFecCorrectLevel

【Description】 Set the fisheye distortion correction level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setFecCorrectLevel(const rk_aiq_sys_ctx_t* ctx, int
correctLevel);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
correctLevel	Correction level	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

Module-level API reference

rk_aiq_user_api_afec_SetAttrib

【Description】
Set the fec attribute.

【Syntax】

```
XCamReturn
rk_aiq_user_api_afec_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_fec_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of fec	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

rk_aiq_user_api_afec_GetAttrib

【Description】
Get the fec attribute.

【Syntax】

```
XCamReturn
rk_aiq_user_api_afec_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_fec_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of fec	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_afec.h, rk_aiq_uapi_afec_int.h
-Library file: librkaiq.so

Module-level API data types

fec_correct_direction_t

【Description】
fec correction direction

【Definition】

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

- 【Member】

Member Name	Description
FEC_CORRECT_DIRECTION_X	Only correct the X direction
FEC_CORRECT_DIRECTION_Y	Only correct the Y direction
FEC_CORRECT_DIRECTION_XY	Correction in both XY directions

rk_aiq_fec_attrib_t

【Description】
fec attribute configuration

【Definition】

```
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;
```

【Member】

Member Name	Description
en	Enable/Disable fec
bypass	Bypass fec
correct_level	Set fec correction level (0-255)
direction	Set fec correction direction

Performance optimization

-The throughput rate of the fec module is affected by the ddr bandwidth. The throughput rate of the fec module that without optimization is show in table bellow:

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	20fp

The optimized solution for FEC is as follows

1. ispp video buffer uses cma memory, and modified as follows
    - The kernel dts modifies ispp to use cma buffer, and the size is allocated according to the actual sensor resolution

```
The optimized fec processing speed scheme is as follows

1. Ispp video buffer uses cma memory, modified as follows

    -Kernel dts modify ispp to use cma buffer, and size is allocated
according to the actual sensor resolutiondiff --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 @@
 };

 &rkispp_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 = <500000000>, <250000000>,
                       <400000000>;
        power-domains = <&power RV1126_PD_ISPP>;
-       iommus = <&rkispp_mmu>;
+       /* iommus = <&rkispp_mmu>; */
+       memory-region = <&isp_reserved>;
        status = "disabled";
    };
```

- The app confirms that the Streaming I/O method is the Memory Mapping method

```
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 only corrects the distortion in the y direction

   Users can configure fec to corrects the y direction only by calling rk_aiq_uapi_setFecCorrectDirection,

   The distortion correction method of this scheme is: ldch correction x direction + fec correction y direction

```
// This function needs to be called before rkaiq executes prepare to take
effect
rk_aiq_uapi_setFecCorrectDirection(ctx, FEC_CORRECT_DIRECTION_Y);
```

3. Increase isp/ispp process clock frequency

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   500000000          0     0  50000
     clk_isp        0        2        0   500000000          0     0  50000
  clk_ispp_np5      0        0        0   111111112          0     0  50000
  clk_ispp_div      0        1        0   500000000          0     0  50000
     clk_ispp       0        2        0   500000000          0     0  50000
        aclk_isp    0        2        0   594000000          0     0  50000
           hclk_isp 0        2        0   297000000          0     0  50000
  aclk_pdispp_np5   0        0        0   475200000          0     0  50000
  aclk_pdispp_div   0        1        0   594000000          0     0  50000
     aclk_pdispp    0        2        0   594000000          0     0  50000
        aclk_pdispp_niu 0        0        0    594000000          0     0
50000
        aclk_ispp   0        4        0   594000000          0     0  50000
        hclk_pdispp 0        1        0   297000000          0     0  50000
           hclk_pdispp_niu 0        0        0   297000000          0     0
50000
           hclk_ispp 0        4        0   297000000          0     0  50000
```

LDCH

Function description

The squint distortion, pincushion, barrel distortion, etc. caused by the distortion of the optical system and the electronic scanning system may cause the geometric characteristics of the image to be distorted. Image distortion correction is a process of transforming a distorted image into an ideal image in a certain transformation manner.

This module only corrects the image distortion in the x direction.

- ## Functional API Reference

  rk_aiq_uapi_setLdchEn

  【Description】 Enable horizontal distortion correction function.

  【Syntax】

  ```
  XCamReturn rk_aiq_uapi_setLdchEn(const rk_aiq_sys_ctx_t* ctx, bool en);
  ```

  【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
en	enable	Input

  【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_aldch.h, rk_aiq_uapi_aldch_int.h
-Library file: librkaiq.so

rk_aiq_uapi_setLdchCorrectLevel

【Description】 Set the level of horizontal distortion correction.

【Syntax】

```
XCamReturn rk_aiq_uapi_setLdchCorrectLevel(const rk_aiq_sys_ctx_t* ctx, int
correctLevel);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
correctLevel	Correction level	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_aldch.h, rk_aiq_uapi_aldch_int.h
-Library file: librkaiq.so

Module-level API reference

rk_aiq_user_api_aldch_SetAttrib

【Description】
Set the fec attribute.

【Syntax】

```
  XCamReturn
  rk_aiq_user_api_aldch_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_ldch_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of ldch	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_aldch.h, rk_aiq_uapi_aldch_int.h
-Library file: librkaiq.so

rk_aiq_user_api_aldch_GetAttrib

【Description】
Get the fec attribute.

【Syntax】

```
  XCamReturn
  rk_aiq_user_api_aldch_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_ldch_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Parameter attributes of ldch	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_aldch.h, rk_aiq_uapi_aldch_int.h
-Library file: librkaiq.so

Module-level API data types

rk_aiq_ldch_attrib_t

【Description】
ldch attribute configuration

【Definition】

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

【Member】

Member Name	Description
en	Enable/Disable ldch
correct_level	Set ldch correction level (0-255)

HDR

Function description

HDR (High Dynamic Range Imaging, HDRI or HDR), in computer graphics and photography, is achieved through calculation using existing equipment to obtain a larger dynamic range than ordinary digital imaging technology (that is, larger The difference between light and shade) is a technique for images. The purpose of HDR is to correctly restore the luminance ratio of the real scene that exceeds the dynamic range of the existing equipment.

Important concepts

-The module contains two parts: Merge and Tmo, Tmo can be used alone, Merge needs to be used together with Tmo

Functional API Reference

rk_aiq_uapi_setHDRMode

【Description】 Set HDR working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setHDRMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	Working mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getHDRMode

【Description】 Get HDR working mode.

【Syntax】

```
  XCamReturn rk_aiq_uapi_getHDRMode(const rk_aiq_sys_ctx_t* ctx, opMode_t
*mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	Working mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMHDRStrth

【Description】 Set the HDR strength in manual mode.

【Syntax】

```
  XCamReturn rk_aiq_uapi_setMHDRStrth(const rk_aiq_sys_ctx_t* ctx, bool on,
unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input
level	Strength Value range: [1,100]	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMHDRStrth

【Description】 Get the HDR strength in manual mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getMHDRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on,
unsigned int* level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	output
level	strength	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Function-level API data types

hdr_OpMode_t

【Description】
Define white balance working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
HDR_OpMode_Api_OFF	api off mode
HDR_OpMode_Auto	Automatic mode
HDR_OpMode_MANU	Manual mode
HDR_OpMode_SET_LEVEL	Fast mode, adjust the HDR effect by setting the level
HDR_OpMode_DarkArea	Dark area boost mode, can be used in both linear and HDR
HDR_OpMode_Tool	Tool Mode

FastMode_t

【Description】
Define HDR fast mode attributes

【Definition】

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

【Member】

Member Name	Description
level	Fast mode level

DarkArea_t

【Description】
Define HDR dark area boost mode attributes

【Definition】

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

【Member】

Member Name	Description
level	Dark area boost level

Module-level API reference

rk_aiq_uapi_ahdr_SetAttrib

【Description】

Set HDR software properties.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	AHDR software attribute structure	input
need_sync	Parameter update switch	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_ahdr_int.h
-Library file: librkaiq.so

【Description】

rk_aiq_uapi_ahdr_GetAttrib

【Description】

Obtain HDR software properties.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	AHDR software attribute structure	input
need_sync	Parameter update switch	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_ahdr_int.h
-Library file: librkaiq.so

【Description】

Module-level API data types

hdr_OpMode_t

【Description】
Define HDR working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
HDR_OpMode_Api_OFF	api off mode
HDR_OpMode_Auto	Automatic mode
HDR_OpMode_MANU	Manual mode
HDR_OpMode_SET_LEVEL	Fast mode, adjust the HDR effect by setting the level
HDR_OpMode_DarkArea	Dark area boost mode, both linear and HDR can be used
HDR_OpMode_Tool	Tool Mode

mgeCtrlData_t

【Description】

Define automatic Merge parameter attributes

【Definition】

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

【Member】

Member Name	Description
stCoef	Current controlling value
stCoefMax	Maximum controlling value
stCoefMin	Minimum controlling value
stSmthMax	Maximum controlling value
stSmthMin	Minimum controlling value
stOfstMax	Maximum curve offset value
stOfstMin	Minimum curve offset value

amgeAttr_t

【Description】

Define automatic merge properties

【Definition】

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

【Member】

Member Name	Description
stMDCurveLM	Motion curve between long and medium frame
stMDCurveMS	Motion curve between medium and short frame
stOECurve	Overexposure curve

tmoCtrlData_t

【Description】

Define automatic Tmo parameter properties

【Definition】

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

【Member】

Member Name	Description
stCoef	Current controlling value
stCoefMax	Maximum controlling value
stCoefMin	Minimum controlling value
stMax	Maximum value of controlled parameter
stMin	Minimum value of controlled parameter

AGlobalTmoData_t

【Description】

Define Global Tmo parameter attributes in automatic Tmo

【Definition】

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

【Member】

Member Name	Description
en	Function switch
stCoef	Current controlling value
stCoefMax	Maximum controlling value
stCoefMin	Minimum controlling value
stMax	Maximum value of controlled parameter
stMin	Minimum value of the controlled parameter

atmoAttr_t

【Description】
Define automatic Tmo attributes

【Definition】

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

【Member】

Member Name	Description
stGlobeLuma	Global luma parameters
stDtlsLL	Low light area parameters
stDtlsHL	Highlight area parameters
stLocalTMO	Local Tmo parameters
stGlobalTMO	Global Tmo parameters

ahdrAttr_t

【Description】
Define automatic HDR attributes

【Definition】

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

【Member】

Member Name	Description
bUpdateTmo	Update Tmo parameter switch
bUpdateMge	Update Merge parameter switch
stMgeAuto	Automatic Merge parameters
stTmoAuto	Automatic Tmo parameters

mmgeAttr_t

【Description】
Define the Merge working mode in manual HDR

【Definition】

```
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;
```

【Member】

Member Name	Description
OECurve_smooth	Overexposure curve slope
OECurve_offset	Overexposure curve offset
MDCurveLM_smooth	Long and medium frame motion curve slope
MDCurveLM_offset	Long and medium frame motion curve offset
MDCurveMS_smooth	Short and medium frame motion curve slope
MDCurveMS_offset	Middle and short frame motion curve offset
dampOE	Smoothing coefficient of overexposure curve
dampMDLM	Smoothing coefficient of long and medium frame motion curve
dampMDMS	Smoothing coefficient of short and medium frame motion curve

mtmoAttr_t

【Description】
Define Tmo working mode in manual HDR

【Definition】

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

【Member】

Member Name	Description
stGlobeLuma	Overall brightness parameters
stDtlsHL	Luminance parameters in low light
stDtlsLL	Brightness parameters in highlights
stLocalTMOStrength	Local Tmo parameters
stGlobalTMOStrength	Global Tmo parameters
damp	Tmo parameter smoothing coefficient

mhdrAttr_t

【Description】
Define manual HDR working mode

【Definition】

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

【Member】

Member Name	Description
bUpdateTmo	Update Tmo parameter switch
bUpdateMge	Update Merge parameter switch
stMgeManual	Manual Merge parameters
stTmoManual	Manual Tmo parameters

FastMode_t

【Description】
Define HDR fast mode attributes

【Definition】

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

【Member】

Member Name	Description
level	Quick mode level

DarkArea_t

【Description】
Define HDR dark area boost mode attributes

【Definition】

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

【Member】

Member Name	Description
level	Dark area upgrade level

CurrCtlData_t

【Description】
Define the attributes of the current control quantity

【Definition】

```
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;
```

【Member】

Member Name	Description
SceneMode	Scene Mode
GlobalLumaMode	Overall brightness parameter control value mode
DetailsHighLightMode	Brightness parameter control value mode in highlights
DetailsLowLightMode	Brightness parameter control value mode in low light
Global TmoMode	Global Tmo control value mode
LocalTMOMode	Local Tmo control value mode
Envlv	Ambient Brightness
MoveCoef	Motion Coef
ISO	ISO
OEPdf	Percentage of overexposed area
FocusLuma	Brightness at focus
DarkPdf	Percentage of Dark Area
DynamicRange	Dynamic Range of Screen

CurrRegData_t

【Description】

Define the attributes of the current controlled quantity

【Definition】

```
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;
```

【Member】

Member Name	Description
OECurve_smooth	Overexposure curve slope
OECurve_offset	Overexposure curve offset value
MDCurveLM_smooth	Long and medium frame motion curve slope
MDCurveLM_offset	Long and medium frame motion curve offset value
MDCurveMS_smooth	Short and medium frame motion curve slope
MDCurveMS_offset	Middle and short frame motion curve offset value
GlobalLuma	Global luma
DetailsLowlight	Luminance in low light
DetailsHighlight	Luminance in highlights
LocalTmoStrength	Local TmoStrength
GlobaltmoStrength	Global Tmo Strength

CalibDb_HdrMerge_t

【Description】

Define the Merge attribute in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
envLevel	Environment brightness
oeCurve_smooth	Overexposure curve slope
oeCurve_offset	Overexposure curve offset
mdCurveLm_smooth	Long and medium frame motion curve slope
mdCurveLm_offset	Long and medium frame motion curve offset value
mdCurveMs_smooth	Short and medium frame motion curve slope
mdCurveMs_offset	Middle and short frame motion curve offset value
oeCurve_damp	Overexposure curve smoothing coefficient
mdCurveLm_damp	Long and medium frame motion curve smoothing coefficient
mdCurveMs_damp	Smooth coefficient of short and medium frame motion curve

TMO_en_t

【Description】
Define the switch attributes of Tmo in stTool

【Definition】

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

【Member】

Member Name	Description
name	name
en	Function swtich

GlobalLuma_t

【Description】
Define the overall brightness properties of Tmo in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
name	name
GlobalLumaMode	Global Luma control mode
envLevel	Environment Luminance
ISO	ISO
Tolerance	Control amount tolerance
globalLuma	Global luma strength

DetailsHighLight_t

【Description】
Define the brightness attribute of the highlight of the Tmo in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
name	name
DetailsHighLightMode	Details high light control mode
OEPdf	Percentage of overexposed area
EnvLv	Environment Brightness
Tolerance	Control parameter tolerance
detailsHighLight	Luminance in highlights

DetailsLowLight_t

【Description】
Define the low-light brightness properties of Tmo in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
name	name
DetailsLowLightMode	Details low light control mode
FocusLuma	Luminance at the focus area
DarkPdf	Percentage of Dark area
ISO	ISO
Tolerance	Control parameter tolerance
detailsLowLight	Luminance in low light

LocalTMO_t

【Description】

Define the Local Tmo attribute of Tmo in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
name	name
LocalTMOMode	Local Tmo control mode
DynamicRange	Dynamic Range of image
EnvLv	Environment Brightness
Tolerance	Control parameter tolerance
Strength	Local Tmo Strength

'GlobaTMO_t

【Description】
Define the Global Tmo attribute of Tmo in stTool

【Definition】

```
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;
```

【Member】

Member Name	Description
name	name
en	Function switch
iir	Global Tmo IIR frames
mode	Global Tmo control mode
DynamicRange	Dynamic Range of image
EnvLv	Environment Brightness
Tolerance	Control amount tolerance
Strength	Global Tmo Strength

CalibDb_HdrTmo_t

【Description】
Define the Tmo attribute in stTool

【Definition】

```
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];
    GlobaTMO_t          GlobaTMO[CALIBDB_MAX_MODE_NUM];
    float               damp;
} CalibDb_HdrTmo_t;
```

【Member】

Member Name	Description
en	Function switch
luma	Global luma parameters
HighLight	Luminance parameters at high light
LowLight	Luminance parameters in low light
LocalTMO	Local Tmo Parameters
GlobaTMO	Global Tmo Parameters
damp	Tmo parameter smoothing coefficient

CalibDb_Ahdr_Para_t

【Description】
Define stTool properties

【Definition】

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

【Member】

Member Name	Description
merge	Merge parameters
tmo	Tmo parameters

hdrAttr_t

【Description】
Define HDR attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
opMode	HDR mode
stAuto	Automatic HDR
stManual	Manual HDR
stSetLevel	HDR fast mode, adjust the HDR effect by setting the level
stDarkArea	Dark area enhancement mode, both linear and HDR can be used
CtlInfo	Controlling amount information
RegInfo	Parameter Information
stTool	HDR tool mode

Noise Removal

Function description

Image noise refers to unnecessary or redundant interference information existing in image data. Image denoising is the process of reducing noise in digital images.

Functional API Reference

rk_aiq_uapi_setNRMode

【Description】 Set the denoising mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setNRMode(const rk_aiq_sys_ctx_t* ctx, opMode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	Working mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getNRMode

【Description】 Get the current denoising mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getNRMode(const rk_aiq_sys_ctx_t* ctx, opMode_t* mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	Working mode	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setANRStrth

【Description】 Set the normal denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setANRStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Denoising intensity	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getANRStrth

【Description】 Get the normal denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_getANRStrth(const rk_aiq_sys_ctx_t* ctx, unsigned int*
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Denoising intensity	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMSpaNRStrth

【Description】 Set the airspace denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setMSpaNRStrth(const rk_aiq_sys_ctx_t* ctx, bool on,
unsigned int level);
```

- 【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input
level	Denoising intensity	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMSpaNRStrth

【Description】 Get the airspace denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_getMSpaNRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on,
unsigned int *level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	output
level	Denoising intensity	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMTNRStrth

【Description】 Set the time domain denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_setMTNRStrth(const rk_aiq_sys_ctx_t* ctx, bool on,
unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input
level	Denoising intensity	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMTNRStrth

【Description】 Get the time domain denoising intensity.

【Syntax】

```
XCamReturn rk_aiq_uapi_getMTNRStrth(const rk_aiq_sys_ctx_t* ctx, bool *on,
unsigned int *level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	output
level	Denoising intensity	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Module-level API reference

rk_aiq_user_api_anr_SetAttrib

【Description】

Set the properties of the denoising algorithm.

【Syntax】

```
XCamReturn
rk_aiq_user_api_anr_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Denoising parameter attributes	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This attr attribute parameter is the final parameter structure used by the software algorithm, not the corresponding parameter structure of IQ xml. Some parameters of IQ also need to be converted into algorithm values. But the parameters of the two are basically the same, the difference is small. It is recommended to use rk_aiq_user_api_anr_SetIQPara to set the automatic parameters, so that the parameters correspond to the IQ parameters one-to-one.

【Requirement】

-Header files: rk_aiq_user_api_anr.h, RkAiqHandleInt.h
-Library file: librkaiq.so

rk_aiq_user_api_anr_GetAttrib

【Description】

Get the properties of the denoising algorithm.

【Syntax】

```
XCamReturn
rk_aiq_user_api_anr_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
rk_aiq_nr_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Denoising parameter attributes	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This attr attribute parameter is the final parameter structure used by the software algorithm, not the corresponding parameter structure of IQ xml. Some parameters of IQ also need to be converted into algorithm values. But the parameters of the two are basically the same, the difference is small. It is recommended to use rk_aiq_user_api_anr_GetIQPara to obtain automatic parameters, so that the parameters and IQ parameters are one-to-one correspondence.

【Requirement】

-Header files: rk_aiq_user_api_anr.h, RkAiqHandleInt.h
-Library file: librkaiq.so

rk_aiq_user_api_anr_SetIQPara

【Description】
Set the denoising IQxml parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
para	Denoising IQxml parameter attributes	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This para attribute parameter is the corresponding parameter structure of IQ xml.

【Requirement】

-Header files: rk_aiq_user_api_anr.h, RkAiqHandleInt.h
-Library file: librkaiq.so

rk_aiq_user_api_anr_GetIQPara

【Description】
Get denoising IQxml parameters.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
para	Denoising IQxml parameter attributes	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Note】

-This para attribute parameter is the corresponding parameter structure of IQ xml.

【Requirement】

-Header files: rk_aiq_user_api_anr.h, RkAiqHandleInt.h
-Library file: librkaiq.so

Module-level API data types

rk_aiq_nr_attrib_t

【Description】
Define the parameters of the denoising module

【Definition】

```
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;
```

【Member】

Member Name	Description
eMode	Denoising module properties
stAuto	Denoising module automatic mode parameters
stManual	Manual mode parameters of denoising module

ANROPMode_t

【Description】

Define the mode of the denoising module

【Definition】

```
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;
```

【Member】

Member Name	Description
ANR_OP_MODE_INVALID	Invalid mode of denoising module
ANR_OP_MODE_AUTO	Automatic mode of denoising module
ANR_OP_MODE_MANUAL	Manual mode of denoising module
ANR_OP_MODE_MAX	Maximum value of denoising module mode, which is an invalid mode

ANR_Auto_Attr_t

【Description】

Define the automatic properties of the denoising module

【Definition】

```
typedef struct ANR_Auto_Attr_s
{
    //bayernr
```

```
    int bayernrEn;
    RKAnr_Bayernr_Params_t stBayernrParams;
    RKAnr_Bayernr_Params_Select_t stBayernrParamSelect;

    //mfnr
    int mfnrEn;
    RKAnr_Mfnr_Params_t stMfnrParams;
    RKAnr_Mfnr_Params_Select_t stMfnrParamSelect;

    //ynr
    int ynrEn;
    RKAnr_Ynr_Params_t stYnrParams;
    RKAnr_Ynr_Params_Select_t stYnrParamSelect;

    //uvnr
    int uvnrEn;
    RKAnr_Uvnr_Params_t stUvnrParams;
    RKAnr_Uvnr_Params_Select_t stUvnrParamSelect;

    RKAnr_Mfnr_Dynamic_t stMfnr_dynamic;

} ANR_Auto_Attr_t;
```

【Member】

Member Name	Description
int bayernrEn	bayernr module enable bit
stBayernrParams	The corresponding algorithm attribute parameters of each iso of the bayernr module
stBayernrParamSelect	The bayernr module calculates the attribute parameters based on the current iso
mfnrEn	mfnr module enable bit
stMfnrParams	The corresponding algorithm attribute parameters of each iso of the mfnr module
stMfnrParamSelect	The mfnr module calculates the attribute parameters according to the current iso
ynrEn	ynr module enable bit
stYnrParams	ynr module each iso corresponding algorithm attribute parameter
stYnrParamSelect	The ynr module calculates the attribute parameters based on the current iso
uvnrEn	uvnr module enable bit
stUvnrParams	uvnr module each iso corresponding algorithm attribute parameter
stUvnrParamSelect	The uvnr module calculates the attribute parameters based on the current iso

ANR_Manual_Attr_t

【Description】

Define the manual properties of the denoising module

【Definition】

```
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;
```

【Member】

Member Name	Description
int bayernrEn	bayernr module enable bit
stBayernrParamSelect	bayernr manual setting of algorithm parameters
mfnrEn	mfnr module enable bit
stMfnrParamSelect	mfnr manually set algorithm parameters
ynrEn	ynr module enable bit
stYnrParamSelect	ynr manually set algorithm parameters
uvnrEn	uvnr module enable bit
stUvnrParamSelect	uvnr manually set algorithm parameters

rk_aiq_nr_IQPara_t

【Description】

Define the IQ parameter structure of the denoising module

【Definition】

```
typedef struct rk_aiq_nr_IQPara_s {
    int module_bits;
To
CalibDb_BayerNr_t stBayernrPara;
CalibDb_MFNR_t stMfnrPara;
CalibDb_UVNR_t stUvnrPara;
CalibDb_YNR_t stYnrPara;

} rk_aiq_nr_IQPara_t;
```

【Member】

Member Name	Description
module_bits	nr4 modules corresponding to the flag bit, according to the flag bit to set each module IQ parameters
stBayernrPara	bayernr module corresponding to IQ parameters
stMfnrPara	mfnr module corresponding to IQ parameters
stUvnrPara	IQ parameters corresponding to uvnr module
stYnrPara	IQ parameters corresponding to ynr module

rk_aiq_nr_module_t

【Description】

Define the setting of the denoising module IQ parameters correspond to the corresponding flag bit of the module, each module can be set separately or set uniformly, and it is recognized by this flag bit.

【Definition】

```
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;
```

【Member】

Member Name	Description
ANR_MODULE_BAYERNR	The corresponding flag bit of the bayernr module is bit 0
ANR_MODULE_MFNR	The corresponding flag bit of the mfnr module is bit 1
ANR_MODULE_UVNR	The corresponding flag bit of the uvnr module is bit 2
ANR_MODULE_YNR	The corresponding flag bit of the ynr module is bit 3

CalibDb_BayerNr_t

【Description】

IQ parameter structure of bayernr module

【Definition】

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

【Member】

Member Name	Description
enable	bayernr module enable bit
version	Hardware parameter version number
mode_cell	Different modes, such as normal, hdr. gray mode correspond to denoising parameters.

CalibDb_BayerNr_ModeCell_t

【Description】

IQ parameter structure of bayernr module

【Definition】

```
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;
```

【Member】

Member Name	Description
name	The names of the different modes of the bayernr module, three modes of normal, hdr, and gray
setting	Under one bayernr module, there are two configurations corresponding to lcg and hcg.

CalibDb_BayerNR_Params_t

【Description】

Define the IQ parameters of the bayernr module.

【Definition】

```
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 lamda;
    unsigned char gauss_en;

    //The following parameters are currently not used
    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;
```

【Member】

Member Name	Description
snr_mode	Noise mode, HSNR, LSNR, corresponding to hcg, lcg mode
sensor_mode	sensor hcg, lcg mode, if these two modes are not available, the parameters in lcg are used by default
iso	Parameter configuration corresponding to different iso
filtPara	Single frame noise reduction intensity, the value range is [0, 15.9]. The larger the value, the stronger the noise reduction.
luLevel	Adjust the noise reduction intensity according to the pixel brightness, this is the x-axis brightness, and 8 points are divided from 0-255.
luRatio	One-to-one correspondence with the above luLevel, which is the y-axis noise reduction intensity.
fixW	Weights of 4 levels of segmented noise. Value range [0 7.9].
lamda	Segmented noise threshold, the larger the lamda, the larger the adjustment range. The value range is [0 16383].
gauss_en	Gaussian guided filter 3x3 enable bit. 0: disable, 1: enable.
The remaining parameters	The remaining parameters are not used by the module at present and are ignored directly.

CalibDb_MFNR_t

【Description】

mfnr module IQ parameter structure

【Definition】

```
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];

    //The following parameters are currently not used
    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;
```

【Member】

Member Name	Description
enable	mfnr module enable bit
version	Hardware parameter version number
local_gain_en	Whether to support the local gain mode of the previous stage. 1: Pre-stage local gain mode, 0: Global gain mode
mode_3to1	Whether to use 3to1 mode. 1: Use 3to1 mode, 0: Use 2to1 mode.
mode_cell	Different modes, such as normal, hdr. gray mode correspond to denoising parameters.
Other parameters	Parameters are not currently used

CalibDb_MFNR_ModeCell_t

【Description】

mfnr module IQ parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
name	Different mode names of mfnr module, three modes of normal, hdr, gray
setting	Under the mfnr module, there are two configurations corresponding to lcg, hcg, etc.
dynamic	Turn on or off the mfnr function dynamically according to the exposure.

CalibDb_MFNR_Dynamic_t

【Description】
mfnr module IQ parameter structure

【Definition】

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

【Member】

Member Name	Description
enable	Enable bit of dynamic switch mfnr module function, 1: dynamic switch 0: keep normally open or normally closed state
lowth_iso	Exposure corresponds to the low iso threshold. When it is lower than the low threshold of the exposure iso and the following exposure time, mfnr is turned off.
lowth_time	The exposure corresponds to the low threshold of the exposure time.
highth_iso	Exposure corresponds to the iso high threshold. When it is higher than the high threshold of the exposure iso and the following exposure time, mfnr is turned on.
highth_time	Exposure corresponds to the high threshold of the exposure time.

CalibDb_MFNR_Setting_t

【Description】
mfnr module IQ parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
snr_mode	Under the mfnr module, there are HSNR and LSNR, corresponding to two configurations such as hcg and lcg.
sensor_mode	sensor mode lcg, hcg, etc. correspond to 2 configurations. If there is no dcg mode, the parameters corresponding to lcg will be used by default.
mfnr_iso	Different iso, corresponding to different denoising parameters.

CalibDb_MFNR_ISO_s

【Description】

mfnr module IQ parameter structure

【Definition】

```
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_lvl0_gfdelta[6];
    float y_hi_lvl0_gfdelta[6];
```

```
    float y_lo_lvl1_gfdelta[3];
    float y_hi_lvl1_gfdelta[3];
    float y_lo_lvl2_gfdelta[3];
    float y_hi_lvl2_gfdelta[3];
    float y_lo_lvl3_gfdelta[3];
    float y_hi_lvl3_gfdelta[3];

    float uv_lo_lvl0_gfdelta[6];
    float uv_hi_lvl0_gfdelta[6];
    float uv_lo_lvl1_gfdelta[3];
    float uv_hi_lvl1_gfdelta[3];
    float uv_lo_lvl2_gfdelta[3];
    float uv_hi_lvl2_gfdelta[3];

    float lvl0_gfsigma[6];
    float lvl1_gfsigma[3];
    float lvl2_gfsigma[3];
    float lvl3_gfsigma[3];

    //The following parameters are currently not used
    float y_lo_denoiseweight[4];
    float y_hi_denoiseweight[4];
    float uv_lo_denoiseweight[3];
    float uv_hi_denoiseweight[3];
};
```

【Member】

Member Name	Description
iso	Different iso levels correspond to different denoising parameters.
weight_limit_y	The minimum value of the forward weight value of the brightness layer 0-3.
weight_limit_uv	The minimum value of the forward weight value of the se degree 0-2.
ratio_frq	The low and high boundary when judging the low-frequency layer 0 texture value of luminance/chroma.
luma_w_in_chroma	The weight value of luminance to guide chroma denoising in the judgment of the similarity of the chroma 0th layer.
noise_curve	y component noise sigma curve, y value. The value calibrated by the tool.
noise_curve_x00	y component noise sigma curve, y value. The value calibrated by the tool.
y_lo_noiseprofile	y component low-frequency 4-layer noise curve correction parameter. The value calibrated by the tool.
y_hi_noiseprofile	y component high frequency 4-layer noise curve correction parameter. The value calibrated by the tool.
y_lo_bfscale	y component low frequency 4-layer denoising intensity.
y_hi_bfscale	y component high frequency 4-layer denoising strength.
y_lumanrpoint	The y component fine-tunes the denoising strength of pixels of different brightness according to the brightness of the pixel. This is the x-axis brightness.
y_lumanrcurve	The y component fine-tunes the denoising strength of pixels of different brightness according to the brightness of the pixel. This is the y-axis denoising strength.
y_denoisestrength	y component high and low frequency 4 layers of overall denoising intensity adjustment component.
uv_lo_noiseprofile	uv component low-frequency 3-layer noise curve correction parameter. The value calibrated by the tool.
uv_hi_noiseprofile	uv component high-frequency 3-layer noise curve correction parameter. The value calibrated by the tool.
uv_lo_bfscale	uv component low-frequency 3-layer denoising intensity.
uv_hi_bfscale	uv component high frequency 3-layer denoising strength.
uv_lumanrpoint	The uv component fine-tunes the denoising strength of pixels with different brightness according to the brightness of the pixel. This is the x-axis brightness.

Member Name	Description
uv_lumanrcurve	The uv component fine-tunes the denoising strength of pixels of different brightness according to the brightness of the pixel. This is the y-axis denoising strength.
uv_denoisestrength	y component high and low frequency 4 layers of overall denoising intensity adjustment component.
xxx_xxx_xxx_gfdelta	y and uv low-frequency similarity operator.
xxx_gfsigma	Gaussian filter operator parameters.
The remaining parameters	Currently not used.

CalibDb_UVNR_t

【Description】
uvnr module IQ parameter structure

【Definition】

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

【Member】

Member Name	Description
enable	uvnr module enable bit
version	Hardware parameter version number
mode_cell	Different modes, such as normal, hdr. gray mode correspond to denoising parameters.

CalibDb_UVNR_ModeCell_t

【Description】
uvnr module IQ parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
name	Different mode names of uvnr module, three modes of normal, hdr, gray
setting	Under one module of uvnr, there are two configurations corresponding to lcg, hcg, etc.

CalibDb_UVNR_Params_t

【Description】
uvnr module IQ parameter structure

【Definition】

```
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;


    //The following parameters are not used yet
    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;
```

【Member】

Member Name	Description
snr_mode	Under the mfnr module, there are HSNR and LSNR, corresponding to two configurations such as hcg and lcg.
sensor_mode	sensor mode lcg, hcg, etc. correspond to 2 configurations. If there is no dcg mode, the parameters corresponding to lcg will be used by default.
ISO	Different iso levels correspond to different denoising parameters.
step0_uvgrad_ratio	Gradient scale factor, the smaller the value, the more color loss is in the place where the gradient is large. The value range is [1 63].
step0_uvgrad_offset	Gradient weighting factor, the larger the value, the more the color loss of the image. The value range is [0, 1].
step1_median_ratio	4x4 down-sampling and weighting factor of filtered median image and un-denoised image, value range [0, 1]
step1_bf_sigmaR	Bilateral 1 filter sigma scale factor, the greater the value, the greater the denoising.
step1_bf_uvgain	The uv scale factor of bilateral 1, that is, the y channel guides the uv channel denoising weight, the value range is [0, 7.9].
step1_bf_ratio	The weight of the two-sided 1 denoised image and the un-denoised image. The value range is [0, 1].
step2_median_ratio	32x32 downsampling and the weighting factor of the filtered median image and the un-denoised image, the value range is [0, 1]
step2_bf_sigmaR	Bilateral 2 filter sigma scale factor, the greater the value, the greater the denoising.
step2_bf_uvgain	Bilateral 2 uv scale factor, that is, the y channel guides the uv channel denoising weight, the value range is [0, 7.9].
step2_bf_ratio	Bilateral 2 The weight of the denoised image and the undenoised image. The value range is [0, 1].
step3_bf_sigmaR	Bilateral 3 filter sigma scale factor, the greater the value, the greater the denoising.
step3_bf_uvgain	The uv scale factor of bilateral 3, that is, the y channel guides the uv channel denoising weight, the value range is [0, 7.9].
step3_bf_ratio	Bilateral 3 denoised image and un-denoised image weighted weight, the value range is [0, 1].
kernel_3x3	Bilateral 3, distance weight configuration.
kernel_5x5	Bilateral 2, distance weight configuration.
kernel_9x9	Bilateral 1, distance weight configuration.
kernel_9x9_num	Two-sided 1, two-sided use point mode, a total of 4 types. The value range is [0-3].

Member Name	Description
The remaining parameters	Currently not used.

CalibDb_YNR_t

【Description】
ynr module IQ parameter structure

【Definition】

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

【Member】

Member Name	Description
enable	ynr module enable bit
version	Hardware parameter version number
mode_cell	Different modes, such as normal, hdr. gray mode correspond to denoising parameters.

CalibDb_YNR_ModeCell_t

【Description】
ynr module IQ parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
name	Different mode names of ynr module, three modes of normal, hdr, gray
setting	Under the ynr module, there are two configurations corresponding to lcg and hcg.

CalibDb_YNR_Setting_t

【Description】

ynr module IQ parameter structure

【Definition】

```
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;
```

【Member】

Member Name	Description
snr_mode	Under the ynr module, there are HSNR and LSNR, corresponding to two configurations such as hcg and lcg.
sensor_mode	sensor mode lcg, hcg, etc. correspond to 2 configurations. If there is no dcg mode, the parameters corresponding to lcg will be used by default.
ynr_iso	Different iso, corresponding to different denoising parameters.

CalibDb_YNR_ISO_t

【Description】

ynr module IQ parameter structure

【Definition】

```
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;
```

- **【Member】**

Member Name	Description
iso	Different iso levels correspond to different denoising parameters.
sigma_curve	Noise sigma curve. The noise curve value calibrated by the ynr module calibration tool.
ynr_lci	The ratio of low frequency LL noise curve, 1~4 correspond to four layers respectively.
ynr_lhci	The ratio of high frequency LH noise curve, 1~4 correspond to four layers respectively.
ynr_hlci	The ratio of high frequency HL noise curve, 1~4 correspond to four layers respectively.
ynr_hhci	The ratio of high frequency HH noise curve, 1~4 correspond to four layers respectively.
lo_lumaPoint	In the pixel brightness domain, increase the low-frequency sigma adjustment factor. Corresponds to the brightness of the pixel. Value range [0 255].
lo_lumaRatio	In the pixel brightness domain, increase the low-frequency sigma adjustment factor. Corresponding adjustment factor. Value range [0 2].
lo_directionStrength	The maximum value of the low frequency sigma adjustment factor. The value range is [0 16).
lo_bfScale	The minimum value of the low frequency sigma adjustment factor. Divided into 1-4 layers. Value range [0-16).
imerge_ratio	Low-frequency texture area, double-sided and Gaussian weighted weight, large value, Gaussian proportion, lower value, bilateral proportion.
imerge_bound	The texture threshold mentioned above.
denoise_weight	The weighted ratio of the result of low-frequency denoising and the original image without denoising, the larger the value, the larger the proportion of denoising, the smaller the value, the smaller the proportion of denoising.
hi_lumaPoint	In the pixel brightness domain, increase the high frequency sigma adjustment factor. Corresponds to the brightness of the pixel. The value range is [0 255].
hi_lumaRatio	In the pixel brightness domain, increase the high-frequency sigma adjustment factor. Corresponding adjustment factor. The value range is [0 2).
hi_bfScale	High-frequency bilateral filter sigma influence factor, the larger the value, the greater the noise reduction intensity. Value range [0-16).

Member Name	Description
hwith_d	The spatial weight of the bilateral filtering of the high frequency 1-4 layers.
hi_denoiseStrength	High-frequency denoising intensity adjustment. The larger the value, the greater the denoising intensity. The value range is [0 16).
hi_detailMinAdjDnW	Range of high frequency limit threshold adjustment. Value range [0 2).
hi_denoiseWeight	The weighted ratio between the result of high-frequency denoising and the original image without denoising, the larger the value, the larger the proportion of denoising, the smaller the value, the smaller the proportion of denoising.
y_luma_point	Gradient partition. Gradient calculation threshold adjustment parameters
hgrad_y_level1/2/3/4	Gradient threshold adjustment factor. High frequency 1-4 layers, the gradient adjustment factor corresponds to the y-axis of the curve. Value range [0-4).
hi_soft_thresh_scale	Soft threshold adjustment factor. Corresponding to the overall adjustment factor of high frequency 1-4 layers. Value range [0-1).

Defog

Function description

Defog is a defogging enhancement by dynamically changing the contrast and brightness of the image.

Functional API Reference

rk_aiq_uapi_setDhzMode

【Description】

Set the defogging working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_setDhzMode(const rk_aiq_sys_ctx_t* ctx, opMode_t
mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	mode	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getDhzMode

【Description】

Get the current defogging working mode.

【Syntax】

```
XCamReturn rk_aiq_uapi_getDhzMode(const rk_aiq_sys_ctx_t* ctx, opMode_t*
mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	mode	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMDhzStrth

【Description】

Set the intensity of defogging work.

【Syntax】

```
XCamReturn rk_aiq_uapi_setMDhzStrth(const rk_aiq_sys_ctx_t* ctx, bool on,
unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	input
level	intensity	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMDhzStrth

【Description】

Get the intensity of dehazing work.

【Syntax】

```
XCamReturn rk_aiq_uapi_getMDhzStrth(const rk_aiq_sys_ctx_t* ctx, bool *on,
unsigned int *level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
on	switch	output
level	intensity	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_enableDhz

【Description】

Turn on the defogging function.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_disableDhz

【Description】

Turn off the defogging function.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

ACM

Function description

ACM (Auto Color Managment) provides basic preference color adjustment function,
Adjust the degree, contrast, saturation and chroma to achieve the adjustment of the color preference.

API Reference

rk_aiq_uapi_setBrightness

【Description】

Set the brightness level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setBrightness(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Brightness value level Value range: [0,255] The default value is 128	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getBrightness

【Description】

Get the brightness level.

【Syntax】

```
XCamReturn rk_aiq_uapi_getBrightness(const rk_aiq_sys_ctx_t* ctx, unsigned int* level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Current brightness level	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setContrast

【Description】

Set the contrast level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setContrast(const rk_aiq_sys_ctx_t* ctx, unsigned int
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Contrast level Value range: [0,255] The default value is 128	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getContrast

【Description】

Get the contrast level.

【Syntax】

```
XCamReturn rk_aiq_uapi_getContrast(const rk_aiq_sys_ctx_t* ctx, unsigned int
*level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Current contrast level	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setSaturation

【Description】

Set the saturation level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setSaturation(const rk_aiq_sys_ctx_t* ctx, unsigned int
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Saturation level Value range: [0,255] The default value is 128	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getSaturation

【Description】

Get the saturation level.

【Syntax】

```
XCamReturn rk_aiq_uapi_getSaturation(const rk_aiq_sys_ctx_t* ctx, unsigned int* level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Current saturation level	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setHue

【Description】

Set the chroma level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setHue(const rk_aiq_sys_ctx_t* ctx, unsigned int level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Chroma level Value range: [0,255] The default value is 128	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getHue

【Description】

Get the chromaticity level.

【Syntax】

```
XCamReturn rk_aiq_uapi_getHue(const rk_aiq_sys_ctx_t* ctx, unsigned int* level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Current chroma level	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Sharpen

Function description

The Sharpen module is used to enhance the sharpness of the image, including adjusting the sharpening properties of the edge of the image and enhancing the details of the image
And texture.

Functional API Reference

rk_aiq_uapi_setSharpness

【Description】

Set the sharpening level.

【Syntax】

```
XCamReturn rk_aiq_uapi_setSharpness(const rk_aiq_sys_ctx_t* ctx, unsigned int
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	Sharpening level Value range: [0,100] The default value is 50	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getSharpness

【Description】

Get the sharpening level.

【Syntax】

```
XCamReturn rk_aiq_uapi_getSharpness(const rk_aiq_sys_ctx_t* ctx, unsigned int*
level);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
level	sharpening level	output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

Gamma

Function description

Gamma performs a non-linear conversion of the luminance space of the image to adapt to the output device.

Functional API Reference

rk_aiq_uapi_setGammaCoef

【Description】

Set gamma.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
gammaAttr	Gamma software attribute structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

【Description】

The gamma curve in Api is not switched according to the scene. If the scene changes, please set the gamma curve through the api again.

Function-level API data types

rk_aiq_gamma_op_mode_t

【Description】

Define Gamma working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_GAMMA_MODE_OFF	Api off mode
RK_AIQ_GAMMA_MODE_MANUAL	Api automatic mode
RK_AIQ_GAMMA_MODE_TOOL	Api tool mode

rk_gamma_curve_type_t

【Description】

Define the Gamma curve working mode in manual mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_GAMMA_CURVE_TYPE_DEFUALT	Use Gamma curve in IQ file
RK_GAMMA_CURVE_TYPE_SRGB	Use sRGB standard Gamma 2.2 curve
RK_GAMMA_CURVE_TYPE_HDR	Use HDR mode Gamma curve
RK_GAMMA_CURVE_TYPE_USER_DEFINE1	Use user-defined Gamma curve 1
RK_GAMMA_CURVE_TYPE_USER_DEFINE2	Use user-defined Gamma curve 2

rk_gamma_curve_usr_define1_para_t

【Description】

Define user-defined Gamma curve 1 attributes in manual mode

【Definition】

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

【Member】

Member Name	Description
coef1	The Gamma curve slope
coef2	The zero slope of the Gamma curve

rk_gamma_curve_usr_define2_para_t

【Description】

Define user-defined Gamma curve 2 attributes in manual mode

【Definition】

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

【Member】

Member Name	Description
gamma_out_segnum	Define the X-axis spacing of the Gamma curve, 0: Log space, 1: Linear space
gamma_out_offset	Gamma curve offset
gamma_table	Gamma curve

Agamma_api_manual_t

【Description】

Define manual Gamma attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
en	Function switch
CurveType	Curve Type
user1	User-defined Gamma Curve 1
user2	User-defined Gamma Curve 2

CalibDb_Gamma_t

【Description】
Define Gamma attributes in tool mode

【Definition】

```
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;
```

【Member】

Member Name	Description
gamma_en	Function switch
gamma_out_segnum	Define the X-axis spacing of the Gamma curve, 0: Log Space, 1: Linear space
gamma_out_offset	Gamma curve offset
curve_normal	Gamma curve in linear mode
curve_hdr	Gamma curve in HDR mode
curve_night	Gamma curve in night mode

rk_aiq_gamma_attr_t

【Description】
Define Gamma attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
mode	Api mode
stManual	Manual Gamma parameter
stTool	Tool Gamma Parameters
Scene_mode	Scene Mode

Module-level API reference

rk_aiq_uapi_agamma_SetAttrib

【Description】

Set Gamma software properties.

【Syntax】

```
XCamReturn
rk_aiq_uapi_agamma_SetAttrib(RkAiqAlgoContext *ctx,
                             rk_aiq_gamma_attrib_t attr,
    bool need_sync);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	Gamma software attribute structure	Input
need_sync	Parameter update switch	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_agamma_int.h
-Library file: librkaiq.so

【Description】

The gamma curve in Api is not switched according to the scene. If the scene changes, please set the gamma curve through the api again.

rk_aiq_uapi_agamma_GetAttrib

【Description】
Get Gamma software properties.

【Syntax】

```
XCamReturn
rk_aiq_uapi_agamma_GetAttrib(RkAiqAlgoContext *ctx,
                             rk_aiq_gamma_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	Gamma software attribute structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_agamma_int.h
-Library file: librkaiq.so

【Description】

DPCC

Function description

Module-level API reference

rk_aiq_uapi_adpcc_SetAttrib

【Description】
Set the DPCC software properties.

【Syntax】

```
XCamReturn
rk_aiq_uapi_agamma_SetAttrib(RkAiqAlgoContext *ctx,
                             rk_aiq_dpcc_attrib_t attr,
                             bool need_sync);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	DPCC software attribute structure	Input
need_sync	Parameter update switch	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_adpcc_int.h
-Library file: librkaiq.so

【Description】

rk_aiq_uapi_adpcc_GetAttrib

【Description】
Obtain DPCC software properties.

【Syntax】

```
XCamReturn
rk_aiq_uapi_adpcc_GetAttrib(RkAiqAlgoContext *ctx,
                            rk_aiq_dpcc_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
attr	DPCC software attribute structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_uapi_adpcc_int.h
-Library file: librkaiq.so

【Description】

Module-level API data types

AdpccOPMode_t

【Description】

Define DPCC working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
ADPCC_OP_MODE_INVALID	Api invalid mode
ADPCC_OP_MODE_AUTO	Api automatic mode
ADPCC_OP_MODE_MANUAL	Api manual mode
ADPCC_OP_MODE_TOOL	Api tool mode
ADPCC_OP_MODE_MAX	

Adpcc_basic_params_select_t

【Description】

Define the basic parameter attributes of DPCC

【Definition】

```
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

【Description】

Define the basic parameter attributes of DPCC

【Definition】

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

【Member】

Member Name	Description
arBasic	DPCC basic parameters

Adpcc_bpt_params_t

【Description】

Define automatic DPCC properties

【Definition】

```
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

【Description】

【Definition】

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

The module has not yet been implemented

Adpcc_pdaf_params_t

【Description】

Define PDAF mode attributes in automatic mode

【Definition】

```
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_wrapy;
    unsigned char pdaf_wrapx;
    unsigned short int pdaf_wrapy_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;
```

The module has not yet been completed

CalibDb_Dpcc_Fast_Mode_t

【Description】

Define Fast mode attributes in automatic mode

【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;
```

【Member】

Member name	Description
Fast_mode_enable	Fast_mode function switch, 0: off, 1: on
ISO	ISO
fast_mode_single_en	Single dead pixel correction switch, 0: off, 1: on
fast_mode_single_level	Single dead pixel correction strength, value range [0, 10]
fast_mode_double_en	Double dead pixel correction switch, 0: off, 1: on
fast_mode_double_level	Double dead pixel correction strength, value range [0, 10]
fast_mode_triple_en	Multiple dead pixel correction switch, 0: off, 1: on
fast_mode_triple_level	Multi-dead pixel correction strength, value range [0, 10]

CalibDb_Dpcc_Sensor_t

【Description】

Define Fast mode attributes in automatic mode

【Definition】

```
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;
```

【Member】

Member name	Description
en	sensor dpcc switch function, 0: closed, 1: open
max_level	Maximum effort to remove dead pixels
iso	Environmental ISO
level_single	Removal of a single dead pixel intensity
level_multiple	Removal of multiple dead pixels strength

Adpcc_bpt_params_select_t

【Description】
Define the Fast mode attribute selected in automatic mode

【Definition】

```
typedef Adpcc_bpt_params_t Adpcc_bpt_params_select_t;
```

Adpcc_pdaf_params_select_t

【Description】
Define the attributes of PDAF mode selected in automatic mode

【Definition】

```
typedef Adpcc_pdaf_params_t Adpcc_pdaf_params_select_t
```

Adpcc_Auto_Attr_t

【Description】
Define automatic DPCC properties

【Definition】

```
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;
```

【Member】

Member Name	Description
stBasicParams	Basic parameters in automatic mode
stBptParams	Dead pixel parameters in automatic mode
stPdafParams	PDAF mode parameters in automatic mode
stFastMode	Fast mode parameters in automatic mode
stSensorDpcc	Sensor dead pixel function parameters in automatic mode
stBasicSelect	Basic parameters selected in automatic mode
stBptSelect	Dead pixel parameters selected in automatic mode
stPdafSelect	PDAF mode parameters selected in automatic mode

Adpcc_fast_mode_attr_t

【Description】
Define quick mode properties in manual mode

【Definition】

```
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;
```

【Member】

Member name	Description
Fast_mode_en	Fast_mode function switch
fast_mode_single_en	Single dead pixel correction switch
fast_mode_single_level	Single dead pixel correction strength, value range [0, 10]
fast_mode_double_en	Double dead pixel correction switch
fast_mode_double_level	Double dead pixel correction strength, value range [0, 10]
fast_mode_triple_en	Multi-dead pixel correction switch
fast_mode_triple_level	Multi-dead pixel correction strength, value range [0, 10]

Adpcc_sensor_dpcc_attr_t

【Description】
Define sensor dead pixel function attributes in manual mode

【Definition】

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

【Member】

Member name	Description
en	Sensor dpcc function switch
max_level	Maximum dead pixels correction level
single_level	Single dead pixel correction strength
double_level	Multiple dead pixels correction strength

Adpcc_Manual_Attr_t

【Description】
Define manual DPCC attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
stBasicParams	Basic parameters in manual mode
stBptParams	Dead pixel parameters in manual mode
stPdafParams	PDAF mode parameters in manual mode
stFastMode	Fast mode parameters in manual mode
stSensorDpcc	Sensor dead pixel function parameters in manual mode

CalibDb_Dpcc_Pdaf_t

【Description】
Define tool PDAF mode attributes

【Definition】

```
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 wrapy;
    unsigned short int wrapx_num;
    unsigned short int wrapy_num;
    unsigned char point_x[16];
    unsigned char point_y[16];
    unsigned char forward_med;
} CalibDb_Dpcc_Pdaf_t;
```

CalibDb_Dpcc_set_RK_t

【Description】
Define RK algorithm properties

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_sw_mindis	Red and blue channel dead pixel threshold 1
g_sw_mindis	Green channel dead pixel threshold 1
sw_dis_scale_min	Dead pixel threshold 2
sw_dis_scale_max	Dead pixel threshold 3

CalibDb_Dpcc_set_LC_t

【Description】
Define LC algorithm properties

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_line_thr	Red and blue channel dead pixel threshold 1
g_line_thr	Green channel dead pixel threshold 1
rb_line_mad_fac	Red and blue channel dead pixel threshold 2
g_line_mad_fac	Green channel dead pixel threshold 2

CalibDb_Dpcc_set_PG_t

【Description】
Define PG algorithm attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_pg_fac	Red and blue channel dead pixel threshold
g_pg_fac	Green channel dead pixel threshold

CalibDb_Dpcc_set_RND_t

【Description】
Define RND algorithm properties

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_rnd_thr	Red and blue channel dead pixel threshold 1
g_rnd_thr	Green channel dead pixel threshold 1
rb_rnd_offs	Red and blue channel dead pixel threshold 2
g_rnd_offs	Green channel dead pixel threshold 2

CalibDb_Dpcc_set_RG_t

【Description】
Define RK algorithm properties

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_rg_fac	Red and blue channel dead pixel threshold
g_rg_fac	Green channel dead pixel threshold

CalibDb_Dpcc_set_RO_t

【Description】

Define RO algorithm properties

【Definition】

```
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;
```

【Member】

Member Name	Description
rb_enable	Red and blue channel switch of dead pixel detection
g_enable	Green channel switch of dead pixel detection
rb_ro_lim	Red and blue channel dead pixel threshold
g_ro_lim	Green channel dead pixel threshold

CalibDb_Dpcc_set_t

【Description】

Define dead pixel judgment condition attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
rk	RK Algorithm
lc	LC algorithm
pg	PG algorithm
rnd	RND Algorithm
rg	RG algorithm
ro	RO algorithm

CalibDb_Dpcc_Expert_Mode_t

【Description】

Define tool expert mode attributes

【Definition】

```
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;
```

【Member】

Member name	Description
iso	ISO
stage1_Enable	Default value 1
grayscale_mode	Black and white mode switch, 0: off, 1: on
rk_out_sel	RK dead pixel correction mode, 0: mode 1, 1: mode 2, 2: mode 3
dpcc_out_sel	Dead pixel correction mode, 0: median, 1: RK mode
stage1_rb_3x3	Default value 0
stage1_g_3x3	Default value 0
stage1_inc_rb_center	Default value 1
stage1_inc_g_center	Default value 1
stage1_use_fix_set	Built-in dead pixel determination condition switch, 0: closed, 1: open
stage1_use_set3	The third type of dead pixel judgment condition switch in set_cell, 0: closed, 1: open
stage1_use_set2	The second type of dead pixel judgment condition switch in set_cell, 0: closed, 1: open
stage1_use_set1	The first type of dead pixel judgment condition switch in set_cell, 0: closed, 1: open
set	Dead pixel judgment condition

CalibDb_Dpcc_t

【Description】
Define tool DPCC attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
enable	Switch function
version	Version
fast	fast mode
expert	Expert mode
pdaf	PADF
sensor_dpcc	Sensor dpcc setting

rk_aiq_dpcc_attrib_t

【Description】
Define DPCC attributes

【Definition】

```
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;
```

【Member】

Member Name	Description
eMode	api mode
stAuto	Automatic DPCC mode
stManual	Manual DPCC mode
stTool	Tool DPCC mode

ASD

Functional API Reference

rk_aiq_user_api_asd_GetAttrib

【Description】

Get the calculation result of the current environment brightness.

【Syntax】

```
XCamReturn rk_aiq_user_api_asd_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
asd_attrib_t* attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Calculation result	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_asd.h
-Library file: librkaiq.so

type of data

asd_attrib_t

【Description】

Calculation result of current ambient brightness

【Definition】

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

【Member】

Member Name	Description
cur_m2r	Current average brightness The calculation method is: exp_val_ratio = cur_exp_val / max_exp_va cur_m2r = mean_luma / exp_val_ratio

Demosaic

Function description

Demosaicing mainly refers to converting the input Bayer data into RGB domain data.

Module-level API reference

rk_aiq_user_api_adebayer_SetAttrib

【Description】
Set demosaicing properties.

【Syntax】

```
XCamReturn
rk_aiq_user_api_adebayer_SetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
adebayer_attrib_t attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Demosaicing attributes	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_adebayer.h, rk_aiq_uapi_adebayer_int.h
-Library file: librkaiq.so

rk_aiq_user_api_adebayer_GetAttrib

【Description】
Get demosaicing properties.

【Syntax】

```
XCamReturn
rk_aiq_user_api_adebayer_GetAttrib(const rk_aiq_sys_ctx_t* sys_ctx,
adebayer_attrib_t *attr);
```

【Parameter】

Parameter Name	Description	Input/Output
sys_ctx	AIQ context pointer	input
attr	Demosaicing attributes	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header files: rk_aiq_user_api_adebayer.h, rk_aiq_uapi_adebayer_int.h
-Library file: librkaiq.so

type of data

adebayer_attrib_t

【Description】

Define ISP demosaicing attributes.

【Definition】

```
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;
```

【Member】

Member Name	Description
enable	Demosaic module enable 0: disable 1: enable
enhance_strength[9]	Detail texture enhancement strength under different ISO 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 The larger the value, the better the fineness and clarity of detail , And the pseudo-details will be enhanced accordingly
low_freq_thresh	Low frequency weight selection threshold The larger the value, the lower the probability of selecting low frequency weights
high_freq_thresh	High frequency weight selection threshold The larger the value, the lower the probability of selecting high frequency weights

Other

API Reference

rk_aiq_uapi_setGrayMode

【Description】

Set how the black and white image mode works.

【Syntax】

```
XCamReturn rk_aiq_uapi_setGrayMode(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_gray_mode_t mode);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mode	Working mode	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getGrayMode

【Description】

Set how the black and white image mode works.

【Syntax】

```
rk_aiq_gray_mode_t rk_aiq_uapi_setGrayMode(const rk_aiq_sys_ctx_t* ctx);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input

【Return value】

Return Value	Description
rk_aiq_gray_mode_t	Working mode

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setFrameRate

【Description】

Set the image output frame rate.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
frameRateInfo_t	Frame Rate Information Structure	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getFrameRate

【Description】

Get image output frame rate information.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
frameRateInfo_t	Frame Rate Information Structure	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_setMirroFlip

【Description】

Set image mirroring and flipping.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mirror	Whether to mirror	Input
flip	Whether to flip	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

rk_aiq_uapi_getMirroFlip

【Description】

Obtain image mirroring and flipping information.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
mirror	Whether to mirror	Output
flip	Whether to flip	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_imgproc.h
-Library file: librkaiq.so

type of data

rk_aiq_gray_mode_t

【Description】

Black and white switching working mode

【Definition】

```
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;
```

【Member】

Member Name	Description
RK_AIQ_GRAY_MODE_CPSL	Controlled by cpsl algorithm
RK_AIQ_GRAY_MODE_OFF	Turn off black and white mode
RK_AIQ_GRAY_MODE_ON	Turn on black and white mode

Statistics

Overview

3A statistical information and related configuration provided by ISP

Function description

AE Statistics

- AE hardware statistics mainly include the following parts: 256-segment weighted histogram statistics based on raw graphs, block R/G/B/Y average statistics based on raw graphs; 32-segment bands based on RGB graph before gamma Weight histogram statistics, block R/G/B/Y average statistics based on the RGB image before gamma.

  AE statistics based on raw graphs

  -The module statistics are divided into block brightness statistics and histogram statistics. According to the supported block size and whether it contains sub-window statistics, the statistical mode can be divided into big mode and lite mode.

  -big mode: maximum support global 15X15 block, minimum support 1X1 block, each block can output 10bit R/B channel brightness average and 12bit G channel average, default 15X15 block; on the basis of global block , Supports independent setting of 4 sub-windows, each sub-window can output the sum of 29bit R/B channel brightness and 32bit G channel sum. The average brightness value needs to be obtained by dividing the number of pixels in each sub-window in the software. In this mode, the weighted histogram statistics, according to the number of blocks and the corresponding assigned weights, perform 256-segment 8bit brightness statistics, and the effective number of pixels in each brightness segment is 28bit.

  -Lite mode: Maximum support 5X5 block, minimum support 1X1 block, each block can output 10bit R/B channel average brightness and 12bit G channel average, default 5X5 block; does not support independent setting of sub-windows. In this mode, the weighted histogram statistics, according to the number of blocks and the corresponding assigned weights, perform 256-segment 8bit brightness statistics, and the effective number of pixels in each brightness segment is 28bit.

  AE statistics based on RGB graph

  -The module statistics are divided into block brightness statistics and histogram statistics.
  -Block brightness statistics, maximum support 15X15 block, minimum support 1x1 block, each block can output 10bit R/B channel brightness average and 12bit G channel average, default 15X15 block; based on global block Above, it supports independent setting of 4 sub-windows. Each sub-window can output the sum of 32-bit Y channel brightness. The average brightness value needs to be obtained by dividing the number of pixels in each sub-window in the software.
  -Histogram statistics, the largest support 15X15 block, the latest support 5X5 block, the weighted histogram statistics in this mode, according to the number of blocks and the corresponding assigned weight, 32 segments of 8bit brightness statistics, within each brightness segment The effective bit number of the number of pixels is 16 bits.

AWB statistics

AWB hardware statistics include global statistics and regional statistics.

Global statistical information: the average value of R, G, B of the color temperature area in the global AWB statistical window of the image, and the number of effective statistical points. The color temperature area supports 7 color temperatures.

Block statistics: 15x15 blocks in the global AWB statistics window of the image, and the average value of R, G, B for each block.

AF statistics

The AF hardware statistics include 2 main window statistics and 1 main window block statistics.

Main window statistical information: AF statistical information in the main window.

Block statistics: 15x15 block statistics in the AF statistics main window.

API Reference

rk_aiq_uapi_sysctl_get3AStats

【Description】

Get 3A statistics.

【Syntax】

```
XCamReturn
rk_aiq_uapi_sysctl_get3AStats(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_isp_stats_t *stats);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
stats	Statistics structure pointer	Output

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

rk_aiq_uapi_sysctl_get3AStatsBlk

【Description】

Obtain 3A statistics simultaneously.

【Syntax】

```
XCamReturn
rk_aiq_uapi_sysctl_get3AStatsBlk(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_isp_stats_t **stats, int timeout_ms);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
stats	Statistics structure pointer	Output
timeout_ms	Timeout, -1 means wait indefinitely until there are statistics	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h

-Library file: librkaiq.so

rk_aiq_uapi_sysctl_release3AStatsRef

【Description】

Release the acquired 3A statistics, and use it with rk_aiq_uapi_sysctl_get3AStatsBlk.

【Syntax】

```
void
rk_aiq_uapi_sysctl_release3AStatsRef(const rk_aiq_sys_ctx_t* ctx,
rk_aiq_isp_stats_t *stats);
```

【Parameter】

Parameter Name	Description	Input/Output
ctx	AIQ context pointer	input
stats	Statistics structure pointer	Input

【Return value】

Return Value	Description
0	Success
Not 0	Failure, see error code table for details

【Requirement】

-Header file: rk_aiq_user_api_sysctl.h
-Library file: librkaiq.so

Reference Code

// 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);
}
```

}

Type of data

rk_aiq_isp_stats_t

【Description】
AIQ 3A statistics

【Definition】

```
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;
```

【Member】

Member Name	Description
aec_stats	ae statistics
awb_stats_v200	awb statistics
af_stats	af statistics

RKAiqAecStats_t

【Description】
Define AE data information. For details, see the function description in the AE chapter.

【Definition】

```
typedef struct RKAiqAecStats_s {
    RkAiqAecHwStatsRes_t ae_data;
    RKAiqAecExpInfo_t ae_exp;
} RKAiqAecStats_t;
```

【Member】

Member Name	Description
RkAiqAecHwStatsRes_t	AE module hardware statistics
RKAiqAecExpInfo_t	AE module sensor exposure information

RKAiqAecExpInfo_t

【Description】
AE module exposure parameter information

【Definition】

```
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;
```

【Member】

Member Name	Description
LinearExp	Exposure parameter information for non-HDR mode
HdrExp	Exposure parameter information in HDR mode
line_length_pixels	hts, its value is determined by the sensor configuration sequence
frame_length_lines	vts, its value is determined by the sensor configuration sequence
pixel_clock_freq_mhz	pclk, unit MHz, its value is determined by the sensor configuration sequence

【Precautions】

HdrExp represents the exposure parameter information in HDR mode, and supports 3TO1 at most. HDR 2TO1: subscript 0 means short frame exposure parameters, subscript 1 means long frame exposure parameters, subscript 2 is invalid; HDR 3TO1: subscript 0 means short frame exposure parameters, subscript 1 means medium frame exposure parameters, subscript 2 Represents long frame exposure parameters.

RkAiqExpParamComb_t

【Description】

AE module exposure parameter information details

【Definition】

```
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;
```

【Member】

Member Name	Description
integration_time	Exposure integration time, in seconds
analog_gain	Sensor's analog gain/Total gain
digital_gain	The digital gain of the sensor is temporarily invalid. The digital gain size is merged into analog_gain
isp_dgain	isp's digital gain, temporarily invalid
iso	Sensitivity, temporarily invalid
dcg_mode	dual conversion gain mode
fine_integration_time	fine exposure integration time register value, temporarily invalid
coarse_integration_time	Exposure integration time register value [number of rows]
analog_gain_code_global	sensor analog gain register value
digital_gain_global	sensor digital gain register value, temporarily invalid
isp_digital_gain	isp digital gain register value, temporarily invalid

【Precautions】

-Different sensors have different functions of digital gain, some are used to increase the sensitivity range, and some are used to complement the accuracy of analog gain. Therefore, the digital gain is not listed separately at present, and its size and corresponding register value are all incorporated into the analog gain.
-The dual conversion gain mode has three states. A value of -1 means that the sensor does not support dcg, a value of 0 means LCG, and a value of 1 means HCG.

RkAiqAecHwStatsRes_t

【Description】
AE module hardware statistics

【Definition】

```
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;
```

【Member】

Member Name	Description
Aec_Stat_Res_t	The AE module is based on raw graph statistics, compatible with HDR and non-HDR modes, and supports at most HDR 3TO1 S/M/L statistics.
yuvae_stat	AE module based on the block information of the RGB image before gamma
sihist_stat	AE module based on histogram information of RGB image before gamma

【Precautions】

Aec_Stat_Res_t chn[3]: Represents the statistical information of the first 3 Raw data channels of the HDR Merge module. In non-HDR mode, the corresponding subscript is 0, and other subscripts are invalid; HDR 2TO1 mode, when the corresponding subscript is 0, it means short-frame data path statistics, subscript 1 means long-frame data path statistics, and subscript 2 is invalid; In HDR 3TO1 mode, when the corresponding subscript is 0, it indicates the statistics of the short frame data path, the subscript 1 indicates the statistics of the medium frame data path, and the subscript 2 indicates the statistics of the long frame data path. The statistics module based on raw graphs has the BLC AWB module before, so the statistics based on raw graphs are affected by the gain values of BLC and AWB.

Aec_Stat_Res_t extra: In HDR mode, extra represents the raw image statistics of debayer after HDR synthesis. Before this statistical module has BLC, AWB, HDRMERGE, TMO module, so the statistical information of this module is affected by the gain of BLC, AWB, HDRMERGE, TMO.

Aec_Stat_Res_t

【Description】
The AE module is based on the statistical information of the raw graph

【Definition】

```
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;
```

【Member】

Member Name	Description
rawaebig_stat	Big mode block statistics based on raw graphs
rawaelite_stat	Lite mode block statistics based on raw graph
rawhist_stat	Histogram statistics based on raw graphs

【Precautions】

For the difference between big and lite modes based on raw graph statistics, see the function description module. Since the main difference between the big and lite modes is the number of blocks to count the average brightness of the block and whether to support the sub-window average brightness statistics, the histogram statistics of the big and lite modes based on the raw image here have the same data structure.

rawaebig_stat

【Description】

Big mode statistics based on raw images, including global window block R/G/B average brightness, sub-window R/G/B brightness sum

【Definition】

```
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
```

【Member】

Member Name	Description
channelr_xy	The average brightness information of the r channel of the global window block in big mode. Number of effective bits: 10bit.
channelg_xy	The average brightness information of the g channel of the global window block in big mode. Number of effective bits: 12bit.
channelb_xy	The b-channel mean brightness information of the big mode global window block. Number of effective bits: 10bit.
wndx_sumr	R channel brightness and information of the big mode sub-window. Effective number of bits: 29bit.
wndx_sumg	The g channel brightness and information of the big mode sub-window. Number of effective bits: 32bit.
wndx_sumb	The b channel brightness and information of the big mode sub-window. Effective number of bits: 29bit.

【Precautions】

The big mode statistics information based on the raw graph only contains the statistics information of the R/G/B 3 channels. If you need the statistics information of the Y channel, you can add code to the software to calculate based on the R/G/B statistics.

The big mode global window block statistics based on the raw graph is the divided average brightness statistics, but the sub-window is the brightness and information of the entire window. You need to add code in the software to calculate the average brightness statistics of the sub-window.

The channely_xy, wndx_channelr, wndx_channelg, wndx_channelb, wndx_channely parameters in the structure are all software calculation parameters, and you need to add code and calculate them based on hardware statistics.

rawaelite_stat

【Description】

Lite mode statistics based on raw image, including global window block R/G/B average brightness

【Definition】

```
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
```

【Member】

Member Name	Description
channelr_xy	The average brightness information of the r channel of the global window block in big mode. Number of effective bits: 10bit.
channelg_xy	The average brightness information of the g channel of the global window block in big mode. Number of effective bits: 12bit.
channelb_xy	The b-channel mean brightness information of the big mode global window block. Number of effective bits: 10bit.

【Precautions】

The lite mode statistical information based on the raw graph only contains the statistical information of the R/G/B 3 channels. If you need the Y channel statistical information, you can add code to the software to calculate based on the R/G/B statistical value.

The channely_xy in the structure is a software calculation parameter, and you need to add code to calculate it based on the hardware statistics.

rawhist_stat

【Description】

Histogram statistics based on raw graphs

【Definition】

```
struct rawhist_stat {
    unsigned int bins[RAWHIST_BIN_N_MAX];
};
#define RAWHIST_BIN_N_MAX 256
```

【Member】

Member Name	Description
bins	Segmentation of the histogram, a total of 256 segments, effective bit number: 28bit

yuvae_stat

【Description】

Based on the block average brightness statistics of the RGB map before gamma, including the block Y channel average brightness of the global window and the sum of the sub-window Y channel brightness

【Definition】

```
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
```

【Member】

Member Name	Description
ro_yuvae_sumy	The sum of the Y channel brightness of the sub-window, the number of effective bits: 32bit
mean	Global window block Y channel mean brightness, effective bit number: 8bit

【Precautions】

-The lite mode statistical information based on the raw graph only contains the statistical information of the R/G/B 3 channels. If you need the Y channel statistical information, you can add code to the software to calculate based on the R/G/B statistical value.
-The channely_xy in the structure is a software calculation parameter, and you need to add code to calculate it based on the hardware statistics.

sihist_stat

【Description】
Histogram meter information based on RGB image before gamma

【Definition】

```
struct sihist_stat {
    unsigned int bins[SIHIST_BIN_N_MAX];
};
#define SIHIST_BIN_N_MAX 32
```

【Member】

Member Name	Description
bins	Segmentation of the histogram, a total of 32 segments, the number of effective bits: 16bit

rk_aiq_awb_stat_res_v200_t

【Description】
Define white balance hardware statistics

【Definition】

```
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;
```

【Member】

Member Name	Description
light	White point statistics under different light sources in the main window, up to RK_AIQ_AWB_MAX_WHITEREGIONS_NUM light sources;
blockResult	The RGB accumulation of each block, and the image is divided into 15x15 (RK_AIQ_AWB_GRID_NUM_TOTAL) blocks of the same size without overlapping
multiwindowLightResult	White point statistical results under different light sources in several sub-windows, up to RK_AIQ_AWB_MAX_WHITEREGIONS_NUM light sources;
excWpRangeResult	The statistical results of non-white dots falling in the non-white dot area, up to RK_AIQ_AWB_STAT_WP_RANGE_NUM_V200 non-white dot areas

【Precautions】

If the user wants to obtain the global white point statistical results of the main window, it can be simply converted according to the white point statistical results under all light sources.

rk_aiq_awb_stat_wp_res_light_v200_t

【Description】
Define the white point statistics under a certain light source

【Definition】

```
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;
```

【Member】

Member Name	Description
xYType	White point statistical results of XY boxes of different sizes under a certain light source, up to RK_AIQ_AWB_XY_TYPE_MAX_V200 boxes

rk_aiq_awb_stat_wp_res_v200_t

【Description】
Define the statistical result of the white point under the XY frame of a certain size of a certain light source, and the statistical result of the non-white point in the back non-white point area

【Definition】

```
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;
```

【Member】

Member Name	Description
WpNo	Number of (non-)white dots
Rvalue	The cumulative sum of the (non-)white point R channel
Gvalue	The cumulative sum of the (non-)white point G channel
Bvalue	The cumulative sum of the (non-)white point B channel

rk_aiq_awb_stat_blk_res_v200_t

【Description】

Define the statistical results of each block

【Definition】

```
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;
```

【Member】

Member Name	Description
isWP	Whether the block contains a white point flag of a light source, and records the flags of RK_AIQ_AWB_STORE_LS_WPFLAG_NUM light sources at most
Rvalue	The cumulative sum of the R channel of all points in the block
Gvalue	The cumulative sum of RG channels of all points in the block
Bvalue	Cumulative sum of RB channels of all points in the block

rk_aiq_af_algo_stat_t

【Description】

Define AF statistics

【Definition】

```
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;
```

【Member】

Member Name	Description
roia_sharpness	The sharpness value of the main window;
roia_luminance	The brightness value of the main window;
roib_sharpness	The sharpness value of the independent window;
roib_luminance	The brightness value of the independent window;
global_sharpness	The sharpness value of the 15*15 sub-window under the main window;
focus_starttim	The start time of the most recent VCM move;
focus_endtim	The end time of the most recent VCM move;
sof_tim	The frame start time of this data frame, in ns;

【Precautions】

roia_sharpness/roia_luminance/roib_sharpness/roib_luminance/global_sharpness is the AF hardware statistics.
focus_starttim/focus_endtim/sof_tim are the VCM movement time and the frame start time of the data frame, to assist in confirming whether the VCM movement is over, and the AF hardware statistics are reliable.

Debug & FAQ

How to get the version number

1. aiq provides version release date, aiq version, iq parser version and version information of each algorithm module of isp;
2. Under the default printing level, loading and running the aiq library will not print. You can set the log level of the xcore module to print the aiq version information:

```
export persist_camera_engine_log=0x1000000ff2
```

3. The printed version information is as follows:

```
************************* 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 ************************
```

Version number matching rule description

The version matching rules of AIQ, IQ Tool and ISP Driver are as follows:

v A. B. C

Where B is hexadecimal representation, bit[0:3] indicates the matching version of AIQ and IQ Tool, bit[4:7] indicates the matching version of AIQ and ISP driver, for example:

ISP driver: v 1. 0x3.0 matches AIQ: v1.0x30.0, but does not match AIQ: v1.0x40.0.

IQ tool: v 1. 0x3.0 matches AIQ: v1.0x33.0, but does not match AIQ: v1.0x30.0, where the AIQ version number C is not 0, and there may be version mismatches. It is recommended to use the AIQ version with C version number 0 first for IQ Tool matching.

AIQ Log

Log switch

1. aiq uses 64bits to represent the log level of all modules, and the bitmap and description of each module are as follows:

```
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,
thespecific meaning of each bit is decided by the module itself. These bitsis
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. Module log level configuration:

   As explained above, in the Linux environment, the switch level of each module is controlled by setting the environment variable persist_camera_engine_log.

   For example, if the log switch of the af module is turned on and the level is verbose, then bit[14] = 1, bit[3-0] = 4, so execute before the application is executed:

```
export persist_camera_engine_log=0x4014
```

To view the current log level, you can use the following command:

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

3. List of each module switch

Module	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 interpretation

AE

Due to space limitations, only the debug level log is interpreted here.

-Linear mode 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: AecClmExecute: NewExposure(0.172051) SplitGain(5.735024) SplitIntegrationTime(0.030000) SplitPirisGain(0)
  rk_aiq_ae_algo.cpp:5952: calc result:SetPoint=22.000000,gain=5.720671,time=0.029987,piris=0,reggain=845,regtime=1398
  rk_aiq_ae_algo.cpp:6133: ================================ (exit)==============================
  ```

  Figure 3-1 Linear mode AE LOG

  -Figure 3-1 shows an example of AE LOG in linear mode.

  Line1:

  > Cur-Exp: FrmId=270,gain=0x36a,time=0x576,envChange=0,dcg=-1,pirs=0

  The exposure parameter information of the current frame.
  ```

Member name	Description
FrmId	Frame number of the current frame
gain	The sensor exposure gain register value corresponding to the current frame
time	The sensor exposure time register value corresponding to the current frame
envChange	Whether a sudden change in environment brightness occurs in the current frame. 0: no sudden change in environmental brightness; 1: sudden change in environmental brightness
dcg	The dcg mode corresponding to the current frame. -1: The sensor does not support the dcg mode or the dcg mode is switched inside the sensor; 0: LCG mode; 1: HCG mode
pirs	The p-iris iris stepper motor position corresponding to the current frame. If the Airis function is turned off, this parameter is invalid and meaningless.

Line2:

> Last-Res:FrmId=269,gain=0x356,time=0x576,pirs=0

Some of the new exposure parameters set by the last AE run have the same meaning as in (1), so I won't repeat them here. By comparing the exposure parameter LOG information of Line1 and Line2, it can be known whether the current exposure is consistent with the new exposure, that is, whether the new exposure has taken effect.

Line3:

> ================================ Linear-AE (enter)==========
> ==========
> ==========
> ========== ==========

Enter the AE control algorithm module, Linear-AE means the current linear exposure mode.

Line4:

> Framenum=270
>
> Cur gain=6.826667, time=0.029987, pirisGain=0, RawMeanluma=29.564444, YuvMeanluma=34.875557, IsConverged=0

Member name	Description
Framenum	The frame number of the current frame
gain	The sensor exposure gain value corresponding to the current frame
time	The sensor exposure time value corresponding to the current frame
pirisGain	The equivalent gain value of the p-iris aperture corresponding to the current frame. If the Airis function is turned off, this parameter is invalid and meaningless.
RawMeanluma	The brightness of the raw image before the debayer corresponding to the current frame, after deducting the black level, and multiplying it by awb gain.
YuvMeanluma	The brightness of the RGB image before gamma corresponding to the current frame is used to determine the effect of the module after the debayer on the brightness.
IsConverged	Whether the current frame exposure has converged. 0: The exposure has not converged; 1: The exposure has converged

Line 5:

> AecClmExecute: NewExposure(0.180993) SplitGain(6.033096) SplitIntegrationTime(0.030000) SplitPirisGain(0)

Member name	Description
NewExposure	New exposure value obtained by AE control algorithm
SplitGain	New sensor exposure gain
SplitIntegrationTime	New sensor exposure time
SplitPirisGain	New p-iris iris equivalent gain value. If the Airis function is turned off, this parameter is invalid and meaningless.

Line6:

> calc result: SetPoint=22.000000,gain=6.023529,time=0.029987,piris=0,reggain=854,regtime=1398

New exposure finally set

Member name	Description
SetPoint	Target brightness value
gain	The final new exposure gain value
time	Final new exposure time value
piris	The final new p-iris iris equivalent gain value. If the Airis function is turned off, this parameter is invalid and meaningless.
reggain	Register value corresponding to the final new exposure gain value
regtime	Register value corresponding to the final new exposure time value

In summary, we can know the current frame's screen brightness RawMeanLuma and the corresponding target brightness setpoint. Calculate the new exposure by comparing the brightness of the screen and the target brightness.

-Hdr mode AE LOG:

```
rk_aiq_algo_ae_itf.cpp:246: Cur-Exp: FrmId=22,S-gain=0x0,S-time=0x2b6,M-gain=0xb,M-time=0x1a5e,L-gain=0x0,L-time=0x0,envChange=1,dcg=-1--1--1,Piris=0
rk_aiq_algo_ae_itf.cpp:254: Last-Res:FrmId=20,S-gain=0x5,S-time=0x8ca,M-gain=0x11,M-time=0x1a5e,L-gain=0x0,L-time=0x0

rk_aiq_ae_algo.cpp:5983: ================================ HDR-AE (enter)==============================
rk_aiq_ae_algo.cpp:6004: AecRun: SMeanLuma=9.342692, MMeanLuma=37.698597,LMeanLuma=0.000000,TmoMeanluma=37.571430,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)==============================
```

Figure 3-2 AE LOG in Hdr mode

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

The exposure parameter information of the current frame.

Member name	Description
FrmId	Frame number of the current frame
S/M/L-gain	The sensor exposure gain register value corresponding to each frame of the current Hdr. S/M is valid in HDR 2 frame mode; S/M/L is valid in HDR 3 frame mode.
S/M/L-time	The sensor exposure time register value corresponding to each frame of the current Hdr. S/M is valid in HDR 2 frame mode; S/M/L is valid in HDR 3 frame mode.
envChange	Whether a sudden change in environment brightness occurs in the current frame. 0: no sudden change in environmental brightness; 1: sudden change in environmental brightness
dcg	The dcg mode corresponding to the current frame, corresponding to 3 short, medium and long frames respectively. In the Hdr 2 frame mode, the first two values are valid and represent the dcg mode of short and long frames respectively; in the Hdr 3 frame mode, the three values represent the short, medium and long dcg modes respectively. -1: The sensor does not support the dcg mode or the dcg mode is switched inside the sensor; 0: LCG mode; 1: HCG mode
pirs	The p-iris iris stepper motor position corresponding to the current frame. If the Airis function is turned off, this parameter is invalid and meaningless.

Line2:

> Last-Res:FrmId=20,S-gain=0x5,S-time=0x8ca,M-gain=0x11,M-time=0x1a5e,L-gain=0x0,L-time=0x0

Some of the new exposure parameters set by the last AE run have the same meaning as in (1), so I won't repeat them here. By comparing the exposure parameter LOG information of Line1 and Line2, it can be known whether the current exposure is consistent with the new exposure, that is, whether the new exposure has taken effect.

Line3:

> ================================= HDR-AE (enter)== =====================

Enter the AE control algorithm module, HDR-AE means the current HDR exposure mode.

Line4:

> AecRun: SMeanLuma=9.342692, MMeanLuma=37.698597,LMeanLuma=0.000000,TmoMeanluma=37.571430,Isconverged=0,Longfrm=0

Member name	Description
S/M/LMeanLuma	The average brightness of each frame of the current Hdr. S/M is valid in HDR 2 frame mode; S/M/L is valid in HDR 3 frame mode.
TmoMeanluma	The average value of the brightness output from the TMO module of the current frame.
Isconverged	Whether the current exposure of each frame of Hdr has converged. 0: The exposure has not converged; 1: The exposure has converged.
Longfrm	The long frame mode status of the current frame. 0: Long frame mode is off; 1: Long frame mode is on.

Line5:

> Framenum=22 Cur Piris=0, Sgain=1.000000, Stime=0.002570, mgain=1.462177, mtime=0.025000, lgain=1.000000, ltime=0.000000

Member name	Description
Framenum	The frame number of the current frame
s/m/lgain	The sensor exposure gain value corresponding to the current frame. In HDR 2 frame mode, s/m is valid; in HDR 3 frame mode, s/m/l is valid.
s/m/ltime	The sensor exposure time value corresponding to the current frame. In HDR 2 frame mode, s/m is valid; in HDR 3 frame mode, s/m/l is valid.
piris	The p-iris iris equivalent gain value corresponding to the current frame. If the Airis function is turned off, this parameter is invalid and meaningless.

Line6:

> S-HighLightLuma=197.250000, S-Target=100.000000, S-GlobalLuma=9.342692, S-Target=19.959433

Short frame control algorithm LOG

Member name	Description
S-HighLightLuma	Brightness of the highlight area of the current short frame.
S-Target	The target brightness of the current short frame highlight area.
S-GlobalLuma	The average brightness of the global area of the current short frame.
S-Target	Current short frame global area target brightness.

Line7:

> L-LowLightLuma=29.626642, L-Target=48.572094, L-GlobalLuma=37.698597, L-Target=77.620155

Long frame control algorithm LOG

Member name	Description
L-LowLightLuma	Current long frame dark area brightness
L-Target	The current target brightness in the dark area of the long frame.
L-GlobalLuma	The average brightness of the global area of the current long frame.
L-Target	The current long frame global area target brightness.

Line8:

> calc result:piris=0,sgain=1.000000,stime=0.005081,mgain=1.862087,mtime=0.025000,lgain=0.000000,ltime=0.000000

-Exposure results output by AE control algorithm

Member name	Description
s/m/lgain	The final exposure gain value of the new sensor. In HDR 2 frame mode, s/m is valid; in HDR 3 frame mode, s/m/l is valid.
s/m/ltime	The final exposure time value of the new sensor. In HDR 2 frame mode, s/m is valid; in HDR 3 frame mode, s/m/l is valid.
piris	The final new p-iris iris equivalent gain value. If the Airis function is turned off, this parameter is invalid and meaningless.

AF

1) AF statistics of each frame

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

Member name	Description
roia	FV value and brightness value of window A
roib	FV value and brightness value of window B
vcm_stable	Whether the vcm movement end flag: 0: not finished 1: end current time, unit ms vcm movement start time, unit ms vcm movement end time, unit ms Exposure time of the current frame, in ms
sof_tim	The start time of frame data transmission of the current frame, in ms
zoom_stable	Whether zoom is over

Judgment method of vcm_stable

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

extraDelay is specified in IQ xml, can be positive, negative, or zero

vcm_start_time, vcm_end_time are obtained from vcm driver query

How to judge zoom_stable

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

zoom_start_time, zoom_end_time are obtained from zoom driver query

2) Focus trigger:

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

When the change of the FV value of the current frame and the FV value of the last successful focus is greater than TrigThers, and the FV change of consecutive StableFrames frames is less than StableThers, the focus is triggered.

TrigThers, StableFrames, StableThers are specified by IQ xml

3) Search path:

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) Display the current iso value, and update the relevant configuration according to the iso value

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

5) Rough-tune result

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) Fine-tune results

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) Focus result

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

AWB

Refer to "Rockchip_Color_Optimization_Guide_ISP2x_CN"

Fetch raw/yuv images dynamically

The principle of crawling raw images

1. The rough data flow process of the current software isp is:

sensor(raw) -> csi-tx -> isp-rx -> ... -> isp-> ... ->ispp -> ... -> out-yuv, where csi-tx -> isp- The raw image data of rx can be obtained in the hwi layer of aiq.

aiq obtains the number of frames the user wants to save the raw file according to the /tmp/.capture_cnt intermediate file, and aiq writes the raw image corresponding to the number of frames into the /tmp directory.

2. This raw method is only supported in isp readback mode.

Steps to Grab the Raw Picture

1. Run rkaiq.
2. The number of raw image frames to be captured by echo, for example, capture 3 frames

```
Execute under linux system: echo 3> /tmp/.capture_cnt
Execute under android system: echo 3> /data/.capture_cnt
```

3. The captured raw image and corresponding meta information will be generated in the /tmp/capture_image or /data/capture_image directory

```
Under linux system:
[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

Under android system:
[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
```

Run rkisp_demo, grab raw and corresponding yuv image steps

1. Add --sync-to-raw parameter, run rkisp_demo, only rkisp_demo supports

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

2. The number of raw/yuv image frames to be captured by echo, for example, 3 frames are captured

```
Execute under linux system: echo 3> /tmp/.capture_cnt
Execute under android system: echo 3> /data/.capture_cnt
```

3. The captured raw image/meta information/yuv image will be generated in the /tmp/capture_image or /data/capture_image directory

```
Under linux system:
[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

Under android system:
[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. As shown above, the raw image/meta information/yuv image has a one-to-one correspondence

---

error code

Error Code	Description
0	Success
-1	Failure
-2	Invalid parameter
-3	Insufficient memory
-4	File operation failed
-5	ANALYZER module error
-6	ISP module error
-7	Sensor driver error
-8	Thread operation error
-9	IOCTL operation error
-10	Timing error
-20	Timeout
-21	Out of range
-255	Unknown error

Acronyms

Abbreviation	Full name
CIS	Camera Image Sensor
RkAiq	Rockchip Automatical Image Quality
ISP	Image Signal Process
IQ Tuning	Image Quality Tuning