密级状态：绝密(　　)　　秘密(　　)　　内部(　)　　公开(√)

Camera_Engine_Rkisp_User_Manual

（ISP 部）

文件状态： [] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	钟以崇
	完成日期：	2018-11-08
	审　　核：	邓达龙
	完成日期：	2018-11-14

福州瑞芯微电子股份有限公司
Fuzhou Rockchips Electronics Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	审核	备注
V1.0	钟以崇	2018-11-08	发布初版		

目 录

Camera engine 主要实现的是 Raw senosr 的 3A 控制，对于 Linux 系统来说，还可通过在它基础上实现的 libgstrkisp 插件来实现数据流获取等。除了 3A 库源码不开放外，其他部分的代码都是开源的。该文档主要描述了 camera engine 的模块组成，简要 API 说明，编译步骤，及调试方面的注意事项。

文档适用平台

芯片平台	驱动	操作系统	支持情况
RK3288	Linux(Kernel-4.4):rkisp1 driver	Linux	Y
RK3399	Linux(Kernel-4.4):rkisp1 driver	Linux	Y
RV3326	Linux(Kernel-4.4):rkisp1 driver	Android-9.0 Linux	Y

Camera engine 基本框架

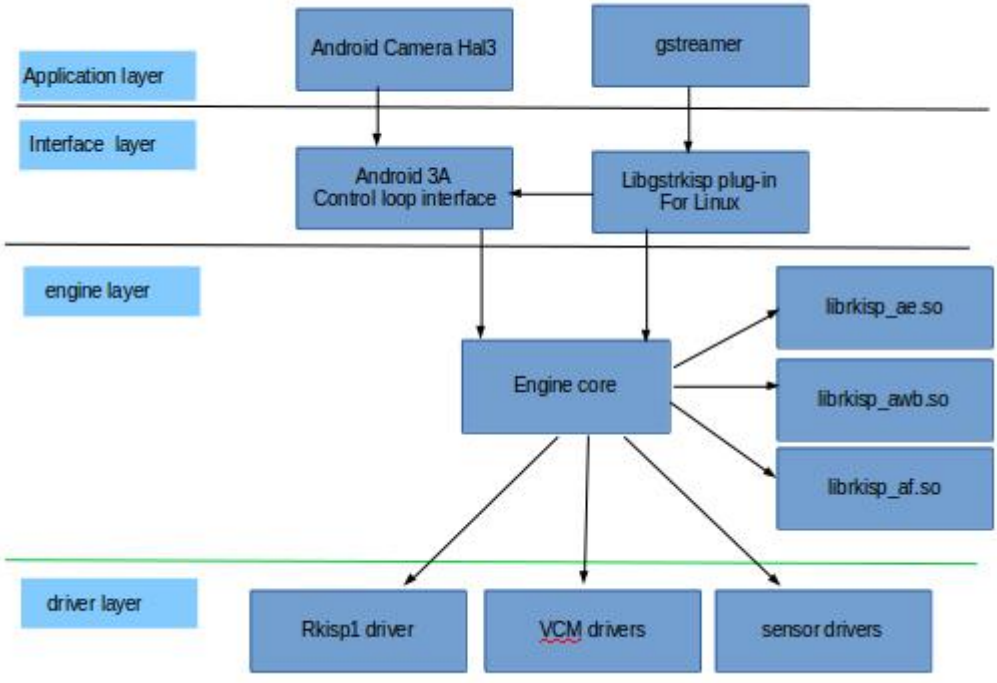

图 1 camera engine 模块结构

上图各模块简要说明如下：

driver layer 为驱动层，不在本文描述范围内，具体参考《RKISP_Driver_User_Manual_v1.0》。

Engine layer

包括 core engine库（librkisp.so）及 3A库。Core engine主体功能为获取驱动数据流，实现上层帧参数控制，如 3A模式等，从ISP驱动获取 3A统计，调用 3A库实现 3A调整。为上层主要提供的类接口为 DeviceManager。librkisp_ae.so，librkisp_awb.so及librkisp_af.so为RK实现的 3A库，实现为动态加载库，且有标准接口，用户如有需求，可实现自己的 3A库进行替换。

Interface layer

在 engine 层基础上为 Android 及 Linux 封装了不同接口。Android 层不需要数据流部分，只需要 3A 控制部分，控制接口及说明请参考头文件 rkisp_control_loop.h，该文件中对实现的接口以及基本调用流程都有详细说明及注释。libgstrkisp 是为 gstreamer 实现的插件，通过该插件，用户可通过 gsreamer 获取数据流以及控制 3A。如用户有其他需求，可封装满足自己需求的接口层。

Application layer

应用层，目前有适配 Android 的 Camera Hal3 及 Linux 平台的 gstreamer。

API 简要说明

Android CL API

接口在 rkisp_control_loop.h 中已有详细说明，简要说明如下：

rkisp_cl_init

[描述]

初始化 control loop。

[语法]

```
int rkisp_cl_init(void** cl_ctx, const char* tuning_file_path,
                  const cl_result_callback_ops_t *callback_ops);
```

[参数]

参数名称	描述	输入输出
cl_ctx	成功返回 control loop context	输出
tuning_file_path	RAW sensor 使用的 tunning xml 文件	输入
callback_ops	接收 result metadata 的回调,可返回 3A state，当前帧的生效参数等	

[返回值]

返回值	描述
0	成功
非 0	失败

rkisp_cl_prepare

[描述]

prepare control loop。

[语法]

```
int rkisp_cl_prepare(void* cl_ctx,
                     const struct rkisp_cl_prepare_params_s*
                     prepare_params);
```

[参数]

参数名称	描述	输入输出
cl_ctx	control loop context	输入
prepare_params	所需控制的设备路径集	输入

[返回值]

返回值	描述
0	成功
非 0	失败

rkisp_cl_start

[描述]

start control loop，调用成功后 control loop 开始运行，3A 开始工作。

[语法]

int rkisp_cl_start(void* cl_ctx)

[参数]

参数名称	描述	输入输出
cl_ctx	control loop context	输入

[返回值]

返回值	描述
0	成功
非 0	失败

rkisp_cl_stop

[描述]

stop control loop

[语法]

int rkisp_cl_stop(void* cl_ctx)

[参数]

参数名称	描述	输入输出
cl_ctx	control loop context	输入

[返回值]

返回值	描述
0	成功
非 0	失败

rkisp_cl_deinit

[描述]

反初始化 control loop

[语法]

void rkisp_cl_deinit(void* cl_ctx)

[参数]

参数名称	描述	输入输出
cl_ctx	control loop context	输入

rkisp_cl_set_frame_params

[描述]

设置新的帧参数，主要包括 3A 模式等

[语法]

```
int rkisp_cl_set_frame_params(const void* cl_ctx,
            const struct rkisp_cl_frame_metadata_s* frame_params);
```

[参数]

参数名称	描述	输入输出
cl_ctx	control loop context	输入
frame_params	新的帧参数	输入

[返回值]

返回值	描述
0	成功
非 0	失败

[注意]

参数结构体直接使用 Android 的 camera_metadata_t 结构，关于如何通过 metadata 设置新参数，请参考 google 官方文档：

https://developer.android.com/reference/android/hardware/camera2/CameraMetadata

设置固定帧率的 sample code 如下：

```
camera_metadata_t * _meta = allocate_camera_metadata(DEFAULT_ENTRY_CAP,
                                                     DEFAULT_DATA_CAP);
CameraMetadata* _metadata = new CameraMetadata(_meta);
int32_t fpsRange[] = {30, 30};
_metadata->update( ANDROID_CONTROL_AE_TARGET_FPS_RANGE,   fpsRange, 2);
struct rkisp_cl_frame_metadata_s frame_params = {0, _meta};
rkisp_cl_set_frame_params(cl_ctx, &frame_params);
```

Linux gstrkisp API

TODO

编译

TODO

Android 平台编译

1. 将 camera engine 源码放至 <android 工程根目录>/hardware/rockchip/
2. 配置 productConfigs.mk
   productConfigs.mk 位于 camera engine 源码根目录下。
   RK3326：
   IS_RKISP_v12 = false 改成 IS_RKISP_v12 = true
   Rk3399,Rk3288：
   不需修改该文件
3. 工程编译环境设置好后，camera engine 源码目录执行 mm 编译

编译后生成 librkisp.so, 3A 库不提供源码，随工程提供编好的库在 plugins/rkiq/<aec/af/awb>/<lib32/lib64>

Linux 平台编译

1. 配置 productConfigs.mk
   配置编译工具链路径： CROSS_COMPILE，如果使用的是 linux sdk 工程则不需要该步骤。
2. 编译

rk3288:

make ARCH=arm

rk3326:

可通过 ARCH=arm 或者 aarch64 来指定编译 32 位或者 64 位库

make ARCH=aarch64 IS_RKISP_v12=true

rk3399:

可通过 ARCH=arm 或者 aarch64 来指定编译 32 位或者 64 位库

make ARCH=aarch64

编译成功后库文件生成在 camera engine 工程目录 build 文件夹下。3A 库不提供源码，随工程提供编好的库在 plugins/rkiq/<aec/af/awb>/<lib32/lib64>

调试

Android 平台调试

log 开关

setprop persist.vendor.rkisp.log <level>

level：

0: error level, defalut level

1: warniing level
2: info level
3: verbose level

更新库

android 8.x 及以上库路径：

librkisp : /vendor/lib<64>
3a: /vendor/lib<64>/rkisp/<ae/awb/af>/
iq: /vendor/etc/camera/rkisp/
更新库后重启 camera 服务：
pkill provider && pkill camera

android 7.x 及以下库路径：

librkisp: /system/lib<64>/
3a: /system/lib<64>/rkisp/<ae/awb/af>/
iq: /system/etc/camera/rkisp/
更新库后重启 camera 服务：
pkill camera*

Linux 平台调试

log 开关

export persist_camera_engine_log=<level>
level：
0: error level, defalut level
1: warniing level
2: info level
3: verbose level

更新库

库路径：

librkisp: /usr/lib/
3a: /usr/lib/rkisp/<ae/awb/af>/
iq: /etc/cam_iq.xml
需要注意的是 iq 文件必须要命名成 cam_iq.xml

Linux 平台使用

1、库及 IQ 文件

1.1 库文件

参考《编译》的章节，camera_engine_rkisp 需要将 5 个库文件 push 到板子里。

1. librkisp.so push 到板子的/usr/lib/
2. librkisp_aec.so push 到板子的/usr/lib/rkisp/ae/
3. librkisp_awb.so push 到板子的/usr/lib/rkisp/awb/
4. librkisp_af.so push 到板子的/usr/lib/rkisp/af/
5. libgstrkisp.so push 到板子的/usr/lib/gstreamer-1.0/

(注：若不使用 gstreamer 可以不用 push libgstrkisp.so)

在 buildroot 系统中，已自动将全部的库拷贝到系统中，如图 1.1-1。

(buildroot/package/rockchip/camera_engine_rkisp/camera_engine_rki sp.mk)

```
RKgstDir = $(TARGET_DIR)/usr/lib/gstreamer-1.0
RKafDir = $(TARGET_DIR)/usr/lib/rkisp/af
RKaeDir = $(TARGET_DIR)/usr/lib/rkisp/ae
RKawbDir = $(TARGET_DIR)/usr/lib/rkisp/awb

define CAMERA_ENGINE_RKISP_INSTALL_TARGET_CMDS
    mkdir -p $(RKgstDir)
    mkdir -p $(RKafDir)
    mkdir -p $(RKaeDir)
    mkdir -p $(RKawbDir)
    mkdir -p $(TARGET_DIR)/etc/iqfiles
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/set_pipeline.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/camera_rkisp.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/S50set_pipeline $(TARGET_DIR)/etc/init.d/
    $(INSTALL) -D -m 755 $(@D)/build/bin/rkisp_demo $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 644 $(@D)/iqfiles/*.xml $(TARGET_DIR)/etc/iqfiles/
    $(INSTALL) -D -m 644 $(@D)/build/lib/librkisp.so $(TARGET_DIR)/usr/lib/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/af/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_af.so $(RKafDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/aec/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_aec.so $(RKaeDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/awb/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_awb.so $(RKawbDir)/
    $(INSTALL) -D -m 644 $(@D)/build/lib/libgstrkisp.so $(RKgstDir)/
endef
```

图 1.1-1

1.2 IQ 文件

在 SDK 工程目录中，在 etc/external/camera_engine_rkisp/iqfiles 目录下统一存放 IQ 文件。如果需要加入新的 IQ 文件，就放在此目录下，并且 IQ 名字规范大写（例如 OV5695.xml），然后删除以下目录 buildroot/output/rockchip_rkxxxx_xx/build/camera_engine_rkisp-1.0，最后重新编译 buildroot。

在 buildroot 系统中，IQ 文件会统一拷贝到板子的/etc/iqfiles/目录下，如图 1.2-1。(buildroot/package/rockchip/camera_engine_rkisp/camera_engine_rkisp.mk)

```
RKgstDir = $(TARGET_DIR)/usr/lib/gstreamer-1.0
RKafDir = $(TARGET_DIR)/usr/lib/rkisp/af
RKaeDir = $(TARGET_DIR)/usr/lib/rkisp/ae
RKawbDir = $(TARGET_DIR)/usr/lib/rkisp/awb

define CAMERA_ENGINE_RKISP_INSTALL_TARGET_CMDS
    mkdir -p $(RKgstDir)
    mkdir -p $(RKafDir)
    mkdir -p $(RKaeDir)
    mkdir -p $(RKawbDir)
    mkdir -p $(TARGET_DIR)/etc/iqfiles
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/set_pipeline.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/camera_rkisp.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/S50set_pipeline $(TARGET_DIR)/etc/init.d/
    $(INSTALL) -D -m 755 $(@D)/build/bin/rkisp_demo $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 644 $(@D)/iqfiles/*.xml $(TARGET_DIR)/etc/iqfiles/
    $(INSTALL) -D -m 644 $(@D)/build/lib/librkisp.so $(TARGET_DIR)/usr/lib/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/af/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_af.so $(RKafDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/aec/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_aec.so $(RKaeDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/awb/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_awb.so $(RKawbDir)/
    $(INSTALL) -D -m 644 $(@D)/build/lib/libgstrkisp.so $(RKgstDir)/
endef
```

图 1.2-1

当系统启动后，会运行/etc/init.d/S50set_pipeline start，这里会匹配当前连接的 sensor，如图 1.2-2 所示，

```
if [[ $MP_NODE =~ "/dev/video" ]]
then
  set_pipeline.sh --sensorbayer $BAYER --sensorname "$NAME"
  if [[ $SENSOR ]]
  then
      ln -fs /etc/iqfiles/$SENSOR*.xml /etc/cam_iq.xml
  fi
fi
done
```

图 1.2-2

通过名字找到/etc/iqfiles/目录下匹配的 xml 文件，链接成/etc/cam_iq.xml，如图 1.2-3 所示，当前 cam_iq.xml 链接的是 OV5695.xml。

```
- entity 7: ov5695 2-0036 (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:SBGGR10_1X10/2592x1944@10000/300000 field:none]
                -> "rockchip-mipi-dphy-rx":0 [ENABLED]

/ # ls -l /etc/cam_iq.xml
lrwxrwxrwx    1 root     root           23 Aug  5 09:12 /etc/cam_iq.xml -> /etc/iqfiles/OV5695.xml
```

图 1.2-3

2 、使用方法

Camera_engine_rkisp 使用方式有两种：1、通过 gstreamer ，2、V4L2 应用编程。

2.1 通过 gstreamer

Camera_engine_rkisp 的使用通过以 plugin 的形式通过 gst-launch-1.0 实现。
测试前我们需要指明动态库的路径：

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0

通过以下命令可以测试

gst-launch-1.0 rkisp device=/dev/video1 io-mode=1 analyzer=1 enable-3a=1 path-iqf=/etc/cam_iq.xml ! video/x-raw,format=NV12,width=640,height=480, framerate=30/1 ! videoconvert ! autovideosink

若没有显示设备，需要 dump 图像，可以将以上命令‘autovideosink’修改为‘filesink location=/tmp/streamer.yuv’，最后通过 yuv 工具预览。

Buildroot 中可以直接使用 camera_rkisp.sh 测试。

2.2 通过 v4l2 应用编程

我们提供了 rkisp_demo 供客户参考测试。如图 2.2-1 代码在工程的 tests/下 rkisp_dmeo 随工程生成在目录 build/bin/

```
leo@leo:~/rk3326/external/camera_engine_rkisp$ ls tests/
Android.mk  build  rkisp_demo.cpp  rkisp_demo.o  test_camcl.cpp  test_camcl.o
leo@leo:~/rk3326/external/camera_engine_rkisp$ ls build/bin/
rkisp_demo  test_ispcl
```

图 2.2-1

Buildroot 系统中，已经将 rkisp_dmeo 拷贝到/usr/bin/下 (buildroot/package/rockchip/camera_engine_rkisp/camera_engine_rkisp.mk)

```
RKgstDir = $(TARGET_DIR)/usr/lib/gstreamer-1.0
RKafDir = $(TARGET_DIR)/usr/lib/rkisp/af
RKaeDir = $(TARGET_DIR)/usr/lib/rkisp/ae
RKawbDir = $(TARGET_DIR)/usr/lib/rkisp/awb

define CAMERA_ENGINE_RKISP_INSTALL_TARGET_CMDS
    mkdir -p $(RKgstDir)
    mkdir -p $(RKafDir)
    mkdir -p $(RKaeDir)
    mkdir -p $(RKawbDir)
    mkdir -p $(TARGET_DIR)/etc/iqfiles
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/set_pipeline.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/camera_rkisp.sh $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 755 $(TOPDIR)/package/rockchip/camera_engine_rkisp/S50set_pipeline $(TARGET_DIR)/etc/init.d/
    $(INSTALL) -D -m 755 $(@D)/build/bin/rkisp_demo $(TARGET_DIR)/usr/bin/
    $(INSTALL) -D -m 644 $(@D)/iqfiles/*.xml $(TARGET_DIR)/etc/iqfiles/
    $(INSTALL) -D -m 644 $(@D)/build/lib/librkisp.so $(TARGET_DIR)/usr/lib/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/af/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_af.so $(RKafDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/aec/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_aec.so $(RKaeDir)/
    $(INSTALL) -D -m 644 $(@D)/plugins/3a/rkiq/awb/$(CAMERA_ENGINE_RKISP_LIB)/librkisp_awb.so $(RKawbDir)/
    $(INSTALL) -D -m 644 $(@D)/build/lib/libgstrkisp.so $(RKgstDir)/
endef
```

图 2.2-2

使用方法：如图 2.2-3，可以通过 rkisp_demo -h 查看，最后会在指定的 ouput 目录下生成图像数据，再通过 yuv 工具预览。

```
rkisp_demo: Add some control parameters(device, output...) for the rkisp_demo

        --width,  default 640,            optional, width of image
        --height, default 480,            optional, height of image
        --format, default NV12,           optional, fourcc of format
        --count,  default    5,           optional, how many frames to capture
        --iqfile, default /etc/cam_iq.xml, optional, camera IQ file
        --device,                         required, path of video device
        --output,                         required, output file path
        --verbose,                        optional, print more log

Example:
        rkisp_demo  --device=/dev/video1 --output=/tmp/test.yuv \
        --width=1920 --height=1080 --count=10
```

图 2.2-3

3、 常见问题

1、如图 3.1，若遇到以下错误提示，是/dev/video 没有指定正确。

```
[root@rk3326_64:/etc]# camera_rkisp.sh
Setting pipeline to PAUSED ...
media get entity by name: rkisp1_mainpath is null
media get entity by name: rkisp1_selfpath is null
media get entity by name: rkisp1-isp-subdev is null
media get entity by name: rkisp1-input-params is null
media get entity by name: rkisp1-statistics is null
media get entity by name: rockchip-sy-mipi-dphy is null
media get entity by name: lens is null
Caught SIGSEGV
exec gdb failed: No such file or directory
Spinning.  Please run 'gdb gst-launch-1.0 578' to continue debugging, Ctrl-C to quit, or Ctrl-\ to dump core.
```

图 3.1

2、如图 3.2，若遇到以下错误提示，是动态库的路径没有指定。

```
/ # gst-launch-1.0 rkisp device=/dev/video1 io-mode=1 analyzer=1 enable-3a=1 pat
h-iqf=/etc/cam_iq.xml ! video/x-raw,format=NV12,width=640,height=480, framerate=
30/1 ! videoconvert ! autovideosink

(gst-launch-1.0:583): GStreamer-WARNING **: Failed to load plugin '/usr/lib/gstreamer-1.0/libgstrkisp.so': libgstvideo4linux2.so: cannot open shared object file: No such file or directory
WARNING: erroneous pipeline: no element "rkisp"
```

图 3.2