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
(版本所有,翻版必究)

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

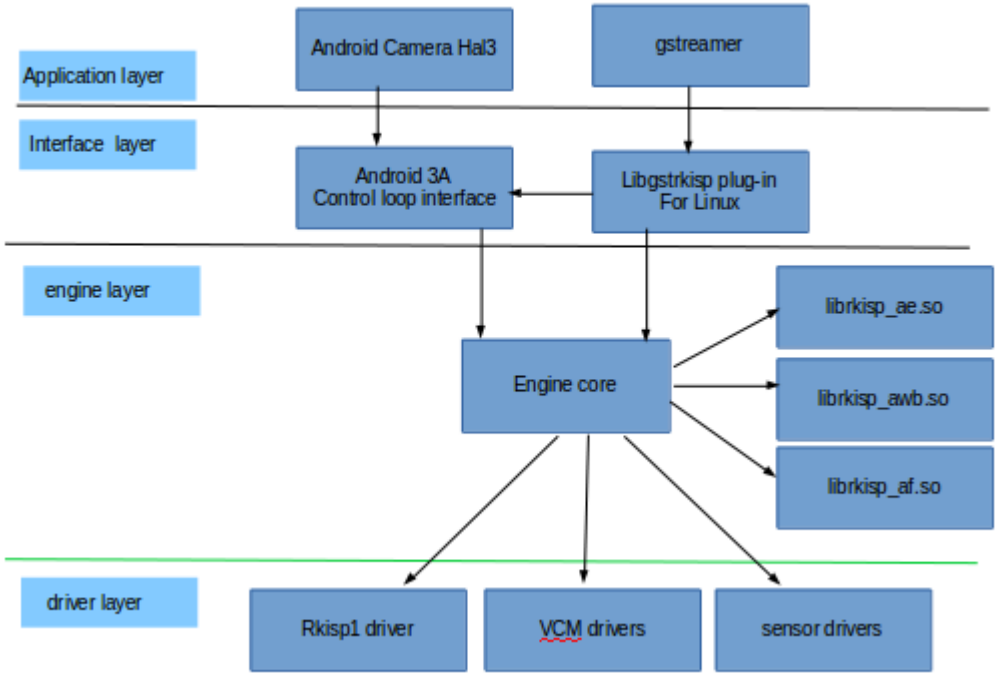

图 1 camera engine 模块结构

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