

密级状态：绝密() 秘密() 内部() 公开(√)

Camera_Hal3_User_Manual

(ISP 部)

文件状态： [√] 正在修改 [] 正式发布	当前版本：	V2.3
	作 者：	付祥, 钟以崇
	完成日期：	2019-12-03
	审 核：	
	完成日期：	

福州瑞芯微电子有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有,翻版必究)

版本历史

		修改日期	修改说明	审核	备注
V1.0	付祥	2018-11-12	发布初版		
V1.0.1	付祥	2018-11-13			
V2.0	付祥	2018-11-16	增加初版 hal3 框架说明		
V2.1	付祥	2019-03-14	修改部分 sensor 配置说明，以及增加个调试用例（适配 hal 3 V1.9.0）		
V2.2	钟以崇，王潘祯撰	2019-07-01	增加 flashlight 配置说明； 修改 camera3_profiles.xml 配置说明		
V2.3	张云龙	2019-12-03	增加 camera3_profiles.xml 中自动生成方式的说明		

目录

目录	4
ROCKCHIP CAMERAHAL3 框架与新增 CAMERA 配置及调试说明	7
文档适用平台	7
1. CAMERA HAL3 框架	7
1.1 CAMERA HAL3 基本框架:	7
1.2 代码目录简要说明	8
1.3 CAMERA HAL3 基本组件:	9
1.4 CAMERA HAL3 与 FRAME WORK 交互时序:	10
1.5 CAMERA HAL3 实现详细时序:	11
1.6 GRAPH 与 MEDIACTL PIPELINE:	11
1.7 CAMERA BUFFER 与 METADATA 管理:	11
2. SENSOR 适配简要步骤说明:	12
2.1 获取 TUNNING XML	12
2.2 配置 CAMERA3_PROFILES.XML	12
2.2.1 camera3_profiles.xml 说明:	12
2.2.2 自动生成方式的说明:	13
2.2.3 手动配置方式的说明	14
Profiles	14
control.aeAvailableAntibandingModes	15
control.aeAvailableModes	15
control.aeAvailableTargetFpsRanges	15
control.afAvailableModes	16
control.awbAvailableModes	16
jpeg.maxSize	16
lens.info.availableApertures	16
lens.info.availableFocalLengths	16
lens.info.minimumFocusDistance	16
scaler.availableMaxDigitalZoom	17
scaler.availableStreamConfigurations	17
scaler.availableMinFrameDurations	17
scaler.availableStallDurations	18

sensor.info.activeArraySize	18
sensor.info.physicalSize	18
sensor.info.pixelArraySize	18
sensor.orientation	18
flash.info.available	19
supportTuningSize	19
SensorType	19
frame.initialSkip	19
2.2.4 FOV 设置方法:	19
2.2.5 xml 运行生效:	20
3. 编译运行调试:	20
3.1 编译:	20
3.2 生成库:	21
3.3 运行:	21
4. DUMP 说明	22
4.1 属性说明:	22
4.2 未生成 DUMP 文件问题:	23
5. 版本说明:	23
HAL3 版本获取:	23
6. 调试案例:	24
ANDROID.HARDWARE.CAMERA2.CTS.RECORDINGTEST#TESTBASICRECORDING FAILED	24
ANDROID.HARDWARE.CAMERA2.CTS.PERFORMANCETEST#TESTMULTIPLECAPTURE FAILED	24
ANDROID.HARDWARE.CTS.CAMERA2TEST#TESTVIDEOSNAPSHOT FAILED	25
ANDROID.HARDWARE.CAMERA2.CTS.STILLCAPTURETEST#TESTJPEGEXIF FAILED	26
修改 FOV 中的 LENS.INFO.AVAILABLEFOCALLENGTHS 值后导致 CTS 几项测试不过问题	27
问题描述:	27
问题分析:	27
解决办法:	27
问题解决提交点:	28
ITS 常见 FAILED 项及解决办法	28
ITS 测试相关	28
Camera 测试 ITS 输出的 LOG 位置:	28
Camera 测试 ITS 对应的 Python 脚本位置:	29
ITS 测试出现测试 test_meta 失败问题分析示例	29

Rockchip CameraHal3 框架与新增 camera 配置及调试说明

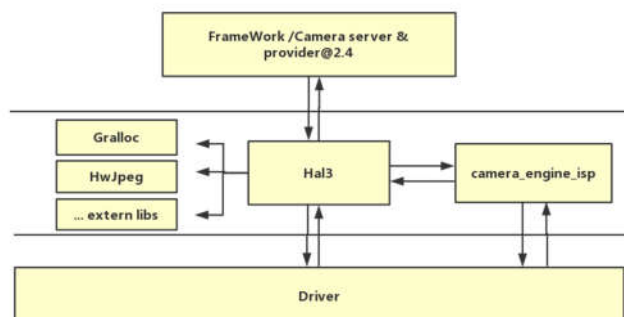
Hal3 基于新框架的 rkisp1 及 cif 驱动，新框架驱动介绍可参考文档《Rockchip Linux Camera 开发指南.pdf》。
(Hal3 代码目录位于 <工程根目录>/hardware/rockchip/camera，以下使用<hal3_camera>来代替)

文档适用平台

芯片平台	驱动	操作系统	HAL3 代码版本
RK3326 RK3399 RK3328 RK3368 RK1808	Linux (Kernel-4.4):rkisp1 driver	Android9.0	V2.0.0

1. Camera Hal3 框架

1.1 Camera Hal3 基本框架:



Camera hal3 在 android 框架中所处的位置如上图，对上，主要实现 Framework 一整套 API 接口，响应其控制命令，返回数据与控制参数结果。对下，主要是通 V4l2 框架实现与 kernel 的交互。3a 控制则是通 control loop 接口与 camera_engine_isp 交互。另外，其中一些组件或功能的实现也会调用到其他一些第三方库，如 cameraBuffer 相关，会调用到 Galloc 相关库，jpeg 编码则会调用到 Hwjpeg 相关库。

驱动框架文档参考：《RKISP_Driver_User_Manual_v1.0》

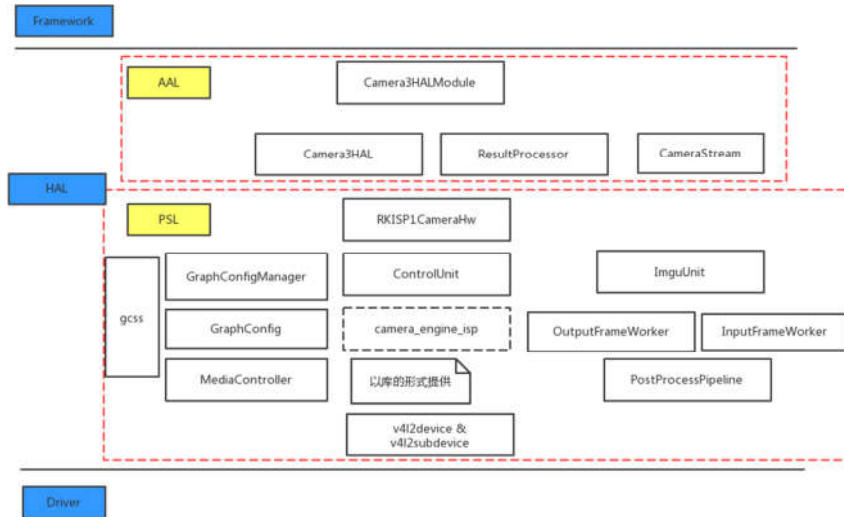
Camera_engine_isp 参考：《Camera_Engine_Rkisp_User_Manual》

1.2 代码目录简要说明

hal3_camera :

— AAL	Android Abstraction Layer, 负责与 framework 交互
— common	公用文件, 如线程, 消息处理, Log 打印等实现
— gcss	xml 解析相关
— imageProcess	图像处理相关, 如 scale
— jpeg	jpeg 编码相关
— mediacontroller	media pipeline 相关
— platformdata	hal3 能力支持的属性, 主要是管理从 xml 获取到的属性
— utils	目前只有一个 Error.h, 定义了一些返回值
— v4l2dev	封装了一些与 v4l2 驱动交互的具体 io
— etc	配置文件目录
— include	Control loop 的头文件, buffer_manager 相关头文件
— lib	3a engine 相关库
— psl	Physical Layer, 物理实现层, 所有的实现逻辑基本都在这里
— rkisp1	目前只有 Rkisp1 一套实现方案
— tasks	基本只用到了里面的几个 Notify 的接口类和 JpegEncodeTask
— workers	数据的获取处理都在这里
— tools	包含了一个自动生成 graph setting xml 的 Python 脚本

1.3 Camera Hal3 基本组件:

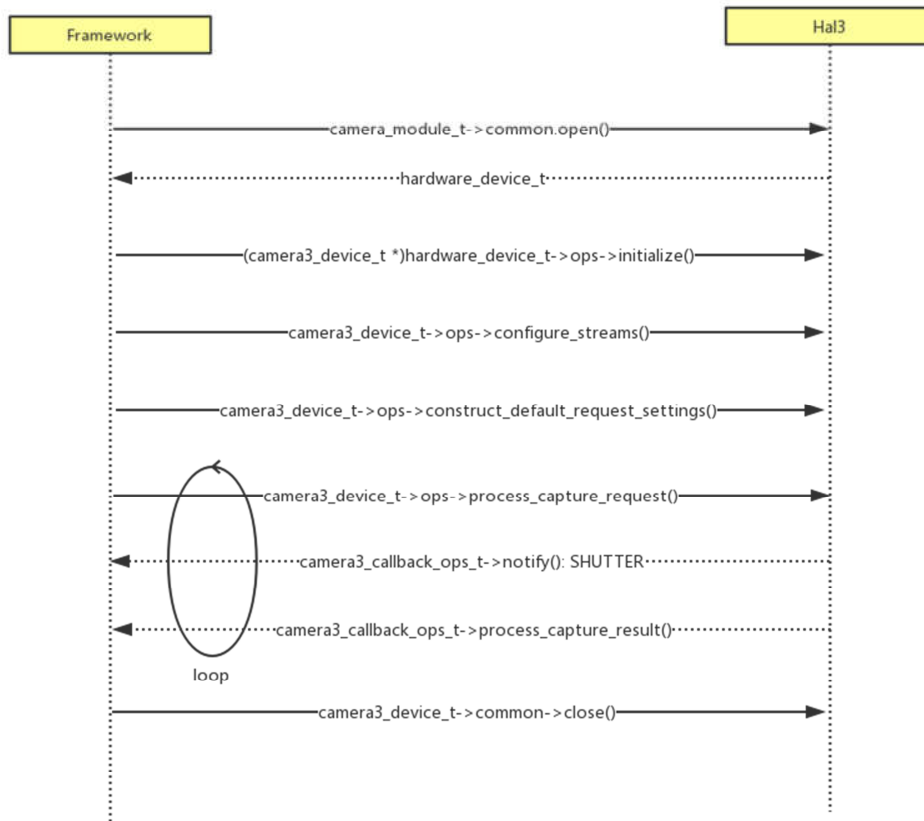


Camera hal3 中的模块主要包括 AAL 与 PSL。

AAL: 主要负责与 framework 交互, camera_module 与 API 接口实例 camera3_device_ops 在此模块定义。该模块对此 API 加以封装, 并将请求发往 PSL, 并等待接收 PSL 返回相应数据流与控制参数。

PSL: 则是物理层的具体实现, 基中 gcss、GraphConifg、MediaController 主要负责配置文件 xml 的解析, 底层 pipeline 的配置, ControlUnit 主要负责与 camera_engine_isp 的交互, 以实现 3a 的控制, 并中转一些请求以及 Metadata 的处理收集上报。 , ImguUnit、OutputFrameWork、postProcessPipeline 则主要负责获取数据帧并做相应处理以及上报。 V4l2device、V4l2Subdevice 则是负责与 v4l2 驱动交互, 实现具体的 io 操作。

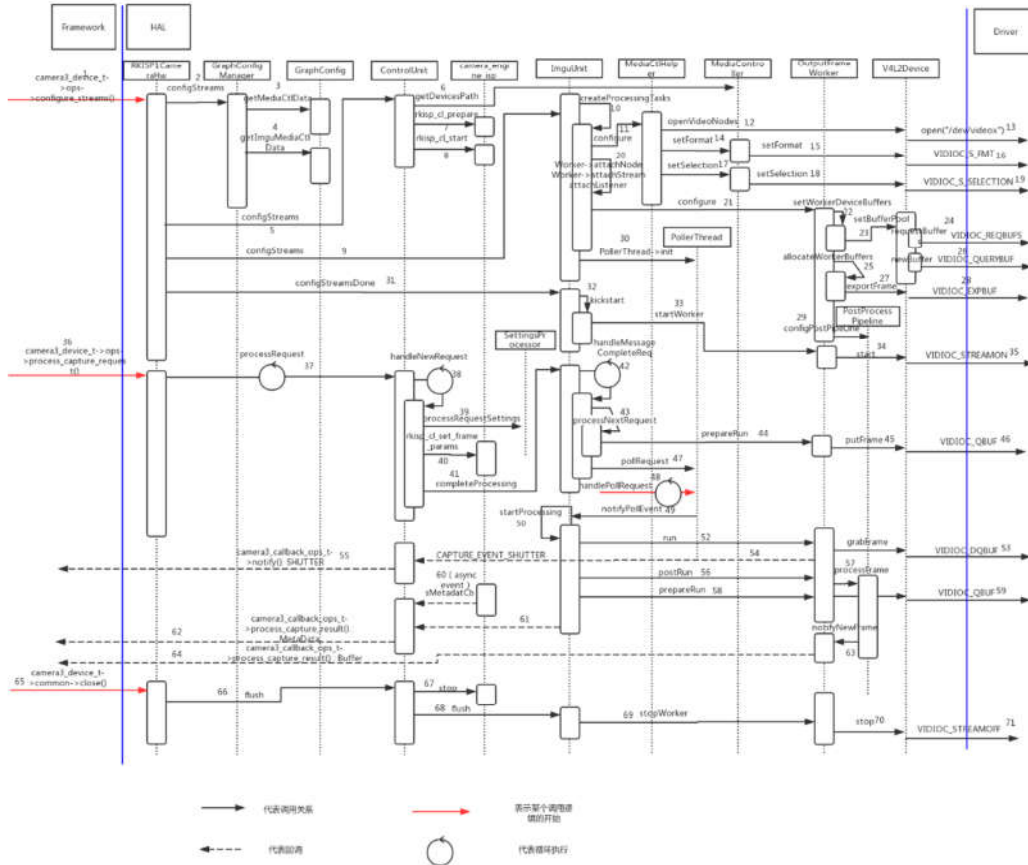
1.4 Camera hal3 与 Framework 交互时序:



关于 framework 与 hal 交互的 API 详细说明文档可以参考:

`<android_root>/hardware/libhardware/include/hardware/camera3.h`

1.5 Camera Hal3 实现详细时序:



此图主要描绘了 `configure_streams` 流程, `process_capture_request` 流程在 Hal3 中的具体实现逻辑, 时序图以这两个 API 接口为起始点, 直到 hal3 下发 v4l2 相关 `ioctl` 并返回相关数据结果为止。该流程图基本涵盖了 Hal3 中的主要模块

上图中 ○ 符号代表循环执行, 也表示此处有线程正在等待事件的到来。Hal 层的运行也正是由这些事件驱动 (即上面的红色键头)。

1.6 Graph 与 mediactl pipeline:

TODO

1.7 Camera buffer 与 MetaData 管理:

TODO

2. Sensor 适配简要步骤说明:

在 sensor 驱动已经调通的基础上, HAL 中添加新 sensor 支持需要配置如下文件: (Hal3 代码目录位于 <工程根目录>/hardware/rockchip/camera, 以下使用<hal3_camera>来代替)

- 1) 获取 tuning 文件, SOC sensor 可略过此步骤
- 2) 配置 camera3_profiles.xml
- 3) 将配置文件 push 到板子, 并重新启动 camera 进程

以下章节是各个步骤详细说明。

2.1 获取 tuning xml

tuning 文件是效果参数文件,只有 Raw sensor 才需该此文件。该文件如何获取可以联系 FAE。

该文件需要以如下方式命令: <sensor_name>_<module_name>_<lens_name>.xml, 并将该文件 push 到板子的 /vendor/etc/camera/rkisp1 目录下。最终 3a 库会从该目录中读取符合规则的 tuning 文件。

另外, 调试 Raw sensor 数据通路时, 也可先 bypass isp。只需要将 sensor 类型设置为 SOC 即可 (参见后续章节中 sensor 类型的设置。), 此时, tuning 文件可暂不配置。

2.2 配置 camera3_profiles.xml

2.2.1 camera3_profiles.xml 说明:

在<hal3_camera>/etc/camera 目录下有多个 camera3_profiles_<platform>.xml, 最终会有一个文件 push 到 /vendor/etc/camera/camera3_profiles.xml. 选择一个适用的 camera3_profiles_<platform>.xml 文件, 参照前面 sensor 的配置添加新 sensor。

camera3_profiles.xml 中包含了多个 Profiles 节点, Profiles 节点包含一个 camera 完整属性列表。开发板上接了几个 sensor, 即需要配置几个 Profiles 节点。

Profiles 节点下又包含了如下四个子节点。

```
<Profiles cameraId="0" name="ov5695" moduleId="m00">
```

```
  <Supported_hardware>
```

```
</Supported_hardware>
```

```
  <Android_metadata>
```

```
</Android_metadata>
```

```
<!-- **** PSL specific section start ****-->
```

```
<Hal_tuning_RKISP1>
```

```
</Hal_tuning_RKISP1>
```

```
<Sensor_info_RKISP1>
```

```
</Sensor_info_RKISP1>
```

```
<!-- **** PSL specific section end ****-->
```

```
</Profiles>
```

`<Android_metadata>` 节点包含的信息主要是 camera 的能力支持，该字段的信息上层将通过 camera_module 的 API: `get_camera_info()` 获取到。Camera 运行时也可以通过如下命令获取到相关的信息。

```
$ adb shell dumpsys media.camera
```

该节点中详细字段的定义可以参见 android 开发者网站：（CTS 中的一些问题需要详细查看该网站中字段的定义）

<https://developer.android.com/reference/android/hardware/camera2/CameraCharacteristics>

其他的几个子节点 主要是平台实现所需要的一些信息， 这些对上层是透明的。

2.2.2 自动生成方式的说明：

从 V2.1 版本开始，camera3_profiles.xml 中增加 forceAutoGenAndroidMetas 配置选项，以选择是否使用自动生成配置项的方式。

forceAutoGenAndroidMetas 在 camera3_profiles.xml 中所处位置：

```
<Hal_tuning_RKISP1>
```

```
<forceAutoGenAndroidMetas value="false"/>
```

```
</Hal_tuning_RKISP1>
```

forceAutoGenAndroidMetas 配置为" true" 时，表示使用自动生成的方式；

forceAutoGenAndroidMetas 配置为" false" 时，表示使用手动配置的方式；

未配置 forceAutoGenAndroidMetas 时，默认使用手动配置的方式。

可自动生成的配置项如下：

```
control.aeAvailableTargetFpsRanges
```

```
peg.maxSize
```

```
jpeg.availableThumbnailSizes
```

scaler.availableStreamConfigurations

scaler.availableMinFrameDurations

scaler.availableStallDurations

sensor.info.activeArraySize

sensor.info.sensitivityRange

sensor.info.exposureTimeRange

sensor.info.maxFrameDuration

sensor.info.pixelArraySize

sensor.maxAnalogSensitivity

未上面列出的项仍需采用手动配置的方式。

手动配置的说明见 2.2.3 章节。

采用自动生成的方式注意事项：

1、需要 sensor 驱动实现 enum_frame_interval 接口。

2、Sensor 驱动中建议只配置两种分辨率，一般为 full size 和 binning size，且 binning size 的最大帧率大于等于 full size 的最大帧率。

3、使用 IQ Tool 调试工具时，建议采用自动生成的方式。

2.2.3 手动配置方式的说明

各平台已有提供参考的 xml 配置文件，另外 camera3_profiles_default.xml 中有提供 RAW sensor 及 SOC sensor 的参考配置，配置时务清楚是 SOC 还是 RAW sensor，然后根据对应的参考配置来进行配置，否则会导致非常多的 CTS 问题。该配置文件会随 HAL3 版本更新而更新，主要是会增加新功能的支持等，客户拿到新 SDK 时需要同步更新具体产品 xml。如下属性需要客户修改以适应不同 sensor：

Profiles

该项配置所需要上报给应用的 Camera 项，每个 Camera 对应一项 Profiles 字段配置。该字段主要的配置项如下：

```
cameraId // 不再作为关键项，可配置 0 或者 1
name     // 需要与驱动名称一致，注意有大小写区别
moduleId
```

关键配置项，值格式为 “mxx”，其中 “m” 为 “module” 缩写，“xx” 为十进制数字，标示 camera 唯一编号，moduleId 需要与驱动 DTS 中配置相一致，否则将探测错误。另外，配置多个 camera 时，多个 camera 的<profiles>项需要按照 moduleId 升序排列。如下图：

```

<!-- RAW SENSOR REFERENCE SETTING -->
+<Profiles cameraId="0" name="imx258" moduleId="m00"></Profiles>
<!-- SOC SENSOR REFERENCE SETTING -->
+<Profiles cameraId="1" name="gc2145" moduleId="m01"></Profiles>

```

此外，为了做到一套固件兼容多种硬件配置，可将所有有可能使用到的 camera 都配置上，只需注意 moduleId 升序排列即可，允许存在相同的 moduleId，但不允许有相同 moduleId 的 camera 能同时 probe 到的情况。

通过如下命令：`adb shell cat /sys/class/video4linux/*/name` 可以获取所有 v4l2 设备节点的名字，其中形如 `m00_b_ov5695 2-0036` 为 sensor 节点名称。该命令规则中，`m00` 代表 moduleId，主要为匹配 `len,flash` 之用，‘`b`’代表 camera 方向为后置，如果是前置则为‘`f`’，‘`ov5695`’代表 sensor name，‘`2-0036`’代表 I2c 地址。

以下 `Android_metadata` 设置项主要为 Android 相关配置项，各字段具体可参考 `<SDK>/system/media/camera/docs/docs.html` 说明。

control.aeAvailableAntibandingModes

SOC: AUTO

RAW: 50HZ,60Hz // 以排在首位的作为初始化配置

control.aeAvailableModes

ON // 不支持 flash 时

ON, ON_AUTO_FLASH, ON_ALWAYS_FLASH // 支持 flash 时

control.aeAvailableTargetFpsRanges

该设置项有多个限制需要注意:

- 1) 录像必需要有一组恒定帧率, 假如帧率为 x , 那就要包含 (x,x)
- 2) 录像帧率必需至少要一组大于 24 帧
- 3) 第一组必需 $\text{Min} \leq 15$. 所以第一组一般为 $(15, x)$
- 4) 各组帧率需要按升序排列

升序具体意义为，假设有定义有两组帧率： $(\text{min1}, \text{max1}), (\text{min2}, \text{max2})$ ，则 $\text{max2} \geq \text{max1}$ ， $\text{max2} = \text{max1}$ 时，还需要满足 $\text{min1} \leq \text{min2}$ 。

一般情况下 sensor 驱动只会输出两组分辨率，全分辨率及 binning 分辨率，其他分辨率即使有调试也一般不使用（可由 ISP 裁剪及缩放得到）。

假设:

$\text{max2} = \text{max_fps_binning}$

$\text{max1} = \text{max_fps_full}$

且 $\text{max2} \geq \text{max1}$

那么

1) 如果 $\text{max1} > 15$, 可按如下配置

$(\text{min1}, \text{max1}), (\text{max1}, \text{max1}), (\text{min2}, \text{max2}), (\text{max2}, \text{max2})$

其中 $\text{min1} \leq 15$, $\text{min2} > 0$, $\text{max2} \geq 24$ 。如果需要增加录像的固定帧率, 则按上述升序列规则添加即可。

示例如下:

假如: $\text{max1} = 20$, $\text{max2} = 30$, 且需要有 15 fps 的固定录像帧率, 那么可按如下配置:

$(15, 15), (10, 20), (20, 20), (10, 30), (30, 30)$

2) 如果 $\text{max1} \leq 15$, 可按如下配置

$(\text{max1}, \text{max1}), (\text{min2}, \text{max2}), (\text{max2}, \text{max2})$

其中 $\text{min2} > 0$, $\text{max2} \geq 24$ 。如果需要增加录像的固定帧率, 则按上述升序列规则添加即可。示例如下:

假如: $\text{max1} = 7$, $\text{max2} = 30$, 且需要有 15 fps 的固定录像帧率, 那么可按如下配置:

$(7, 7), (15, 15), (10, 30), (30, 30)$

注: min fps 可用于控制拍照预览时的最小帧率, 也即控制了最大曝光时间, 可以根据需要进行调整, 但设置过小会影响拍照速度, 但在较暗情况下能获得更好的预览效果。

control.afAvailableModes

SOC: OFF //soc camera 目前不支持 af

RAW: OFF // 如果 camera 没有 af 功能

RAW: AUTO,CONTINUOUS_VIDEO,CONTINUOUS_PICTURE,OFF // camera 具有 af 功能

control.awbAvailableModes

SOC: AUTO

RAW: AUTO,INCANDESCENT,FLUORESCENT,DAYLIGHT,CLOUDY_DAYLIGHT

jpeg.maxSize

计算公式如下:

最大分辨率为: `scaler.availableStreamConfigurations` 中 BLOB 项最大分辨率项

$\text{jpeg.maxSize} \geq \text{max_blob_w} * \text{max_blob_h} * 3 / 2$

lens.info.availableApertures

可选光圈, 目前只支持一个, 可从模组规格书中获取。

lens.info.availableFocalLengths

可选焦长, 目前只支持一个, 可从模组规格书中获取, 与 FOV 计算相关, 配置参考 2.2.3 节。

lens.info.minimumFocusDistance

0.0 // 不支持 af 时

非 0 // 支持 af 时, 务必配置配置成非 0, 具体需要根据模组规格书来设置

scaler.availableMaxDigitalZoom

默认值为 4.0，根据芯片平台及需要可增大或减小放大倍数；注意增大放大倍数时，在放大预览情况下，在不同平台上可能会影响预览帧率，主要是由平台的 2D 加速器引起的，如果发现存在该种情况，请减小放大倍数。

scaler.availableStreamConfigurations

HAL 层支持的分辨率列表，有如下限制：

- 1) 需要按照分辨率依次降序排列
- 2) 为了满足 CTS 要求，需要包含 352x288,320x240,176x144 配置项
- 3) 如果在 media_profiles_V1_0.xml 中有指定录像分辨率，那么该列表中需要包含该分辨率
- 4) 列表中需要支持 BLOB, YCbCr_420_888, IMPLEMENTATION_DEFINED 三种格式输出配置，三种格式中支持 的分辨率都要相同
- 5) 为了不影响拍照速度，如果 sensor 最大输出尺寸宽度大于 4096 时，需将最大分辨率宽度限制在 4096。

满足以上限制条件后，可根据需要增减配置项。以下是 IMX258 参考配置项，IMX258 驱动输出的最大分辨率为 4208x3120。

```
<scaler.availableStreamConfigurations value="
BLOB,4096x3072,OUTPUT,
BLOB,2096x1560,OUTPUT,
BLOB,1920x1080,OUTPUT,
BLOB,1280x960,OUTPUT,
BLOB,1280x720,OUTPUT,
BLOB,640x480,OUTPUT,
BLOB,352x288,OUTPUT,
BLOB,320x240,OUTPUT,
BLOB,176x144,OUTPUT,
YCbCr_420_888,4096x3072,OUTPUT,
YCbCr_420_888,2096x1560,OUTPUT,
YCbCr_420_888,1920x1080,OUTPUT,
YCbCr_420_888,1280x960,OUTPUT,
YCbCr_420_888,1280x720,OUTPUT,
YCbCr_420_888,640x480,OUTPUT,
YCbCr_420_888,352x288,OUTPUT,
YCbCr_420_888,320x240,OUTPUT,
YCbCr_420_888,176x144,OUTPUT,
IMPLEMENTATION_DEFINED,4096x3072,OUTPUT,
IMPLEMENTATION_DEFINED,2096x1560,OUTPUT,
IMPLEMENTATION_DEFINED,1920x1080,OUTPUT,
IMPLEMENTATION_DEFINED,1280x960,OUTPUT,
IMPLEMENTATION_DEFINED,1280x720,OUTPUT,
IMPLEMENTATION_DEFINED,640x480,OUTPUT,
IMPLEMENTATION_DEFINED,352x288,OUTPUT,
IMPLEMENTATION_DEFINED,320x240,OUTPUT,
IMPLEMENTATION_DEFINED,176x144,OUTPUT" /> <!-- critical field, refer to doc camera_hal3_user_manual_vx.x.pdf -->
```

scaler.availableMinFrameDurations

配置 scaler.availableStreamConfigurations 中各分辨率下最小帧间隔（即最大帧率），需要满足以下条件：

- 1) 需要包含 scaler.availableStreamConfigurations 中定义的所有格式，分辨率
- 2) 各分辨率最大帧率可从 sensor 驱动获取，一般 sensor 只输出 full 及 binning 两种分辨率，列表上报的支持分辨率如果 sensor 驱动不能直接支持，那么会由 ISP 裁剪及缩放得到，因此非 sensor 直接输出的分辨率帧率与比之更大的最接近的 sensor 输出分辨率相同。

以下是 IMX258 参考配置项，IMX258 驱动输出的最大分辨率为 4208x3120@20fps,2096x1560@30fps:

```

<scaler.availableMinFrameDurations value="
BLOB,4096x3072,50000000,
BLOB,2096x1560,33333333,
BLOB,1920x1080,33333333,
BLOB,1280x960,33333333,
BLOB,1280x720,33333333,
BLOB,640x480,33333333,
BLOB,352x288,33333333,
BLOB,320x240,33333333,
BLOB,176x144,33333333,
YCbCr_420_888,4096x3072,50000000,
YCbCr_420_888,2096x1560,33333333,
YCbCr_420_888,1920x1080,33333333,
YCbCr_420_888,1280x960,33333333,
YCbCr_420_888,1280x720,33333333,
YCbCr_420_888,640x480,33333333,
YCbCr_420_888,352x288,33333333,
YCbCr_420_888,320x240,33333333,
YCbCr_420_888,176x144,33333333,
IMPLEMENTATION_DEFINED,4096x3072,50000000,
IMPLEMENTATION_DEFINED,2096x1560,33333333,
IMPLEMENTATION_DEFINED,1920x1080,33333333,
IMPLEMENTATION_DEFINED,1280x960,33333333,
IMPLEMENTATION_DEFINED,1280x720,33333333,
IMPLEMENTATION_DEFINED,640x480,33333333,
IMPLEMENTATION_DEFINED,352x288,33333333,
IMPLEMENTATION_DEFINED,320x240,33333333,
IMPLEMENTATION_DEFINED,176x144,33333333" /> <!-- critical field, refer to doc camera_hal3_user_manual_vx.x.pdf -->

```

scaler.availableStallDurations

配置 *scaler.availableStreamConfigurations* 中 BLOB 格式各分辨率的允许的最大间隔时长，可直接复制 *scaler.availableMinFrameDurations* 中 BLOB 的配置项，也可设置大于 *scaler.availableMinFrameDurations* 中的值，只需满足小于 *sensor.info.maxFrameDuration* 中配置的最大间隔即可。设置大点有利于 CTS 拍照相关测试项的稳定性。

以下是 IMX258 参考配置项：

```

<scaler.availableStallDurations value="
BLOB,4096x3072,66666666,
BLOB,2096x1560,33333333,
BLOB,1920x1080,33333333,
BLOB,1280x960,33333333,
BLOB,1280x720,33333333,
BLOB,640x480,33333333,
BLOB,352x288,33333333,
BLOB,320x240,33333333,
BLOB,176x144,33333333" /> <!-- critical field, refer to doc camera_hal3_user_ma

```

sensor.info.activeArraySize

设置成 sensor 驱动输出的最大分辨率，可从 sensor 驱动得到。

sensor.info.physicalSize

sensor 物理尺寸，可从模组规格书中得到。与 FOV 计算相关，配置参考 2.2.3 节。

sensor.info.pixelArraySize

设置成 sensor 驱动输出的最大分辨率，可从 sensor 驱动得到。

sensor.orientation

模组的安装方向，可设置 0, 90, 180, 270。客户需根据模组在机器上的安装方向进行调整，需要通过 CTS verifier 相关项的测试；如果方向有水平或者垂直等镜像问题，则可能需要调整 sensor 驱动的输出方

向。

flash.info.available

FALSE // 不支持闪光灯

TRUE // 支持闪光灯

supportTuningSize

SOC: 不需设置此项

RAW: 需要从对应的 IQ 效果文件中获支持的分辨率，一般来说包括 sensor 的全分辨率和 binning 分辨率。如不设置此项，那么预览时也会使用 sensor 驱动输出的最大分辨率。

SensorType

SOC: SENSOR_TYPE_SOC

RAW: SENSOR_TYPE_RAW // 测试数据流时，可将 RAW 的设置成 SENSOR_TYPE_SOC，只是输出图像无 3A 效果，注意 RAW 摄像头设置成 SOC 仅仅只用于调试。

frame.initialSkip

打开 Camera 应用时，预览前几帧 3A 未收敛，可能在不同场景下存在前几帧偏色等情况，通过该选项可设置合适的过滤帧数来避免该现象。此外，由于 CTS 很多帧率相关项是以发送固定的 request 数量，然后得到相应的帧数量时间来统计的，目前存在第一个 request 结果帧返回过慢的问题，也可通过设置合适的过滤帧数来规避该问题。

2.2.4 FOV 设置方法:

在: frameworks/av/services/camera/libcameraservice/api1/client2/Parameters.cpp 中调用如下函数计算

FOV: res = calculatePictureFovs(&horizFov, &vertFov);

FOV 计算公式:

```
/**
 * Basic field of view formula is:
 * angle of view = 2 * arctangent ( d / 2f )
 * where d is the physical sensor dimension of interest, and f is
 * the focal length. This only applies to rectilinear sensors, for focusing
 * at distances >> f, etc.
 */
```

需要修改的 camera3_profiles.xml 中字段:

```
<lens.info.availableFocalLengths value="x.xx"/>
```

<sensor.info.physicalSize value="x.xx,x.xx"/><!-- 1600x1.75um 1200x1.75um -->

上述两个值获取方法:

availableFocalLengths:

根据模组规格书提供,例如 GC2145 的一个模组规格书:



则: <lens.info.availableFocalLengths value="2.38"/>

physicalSize value:

参考 Sensor 的 Datasheet, 例如 GC2145:

Pixel Size 1.75 μ m x 1.75 μ m

Pixel Size	1.75 μ m x 1.75 μ m
Active pixel array	1616 x 1232

则 physicalSize value 中 hori = 1600x1.75um = 2.8 ;vertical_size = 1200x1.75um =2.1;

<sensor.info.physicalSize value=" 2.8,2.1"/><!-- 1600x1.75um 1200x1.75um -->

根据 FOV 计算公式: $\text{horizFov} = \arctan(2.8/2.38) = 49.6$ 度;

注意:

如果实际 CTS-Verify 测量所需的值为: 56 度; 则需要反推修改 availableFocalLengths

因为 physicalSize value 是固定的, 则根据 FOV 计算公式,

反推出 availableFocalLengths 值 = $2.8/\tan((56^\circ)) = 2.8/1.48 = 1.89$;

则可以上述公式, 反推出修改成<lens.info.availableFocalLengths value="1.89"/>即可;

2.2.5 xml 运行生效:

参照章节 3.3。

3. 编译运行调试:

3.1 编译:

- 1) 确认 <android_root>/device/rockchip/common/BoardConfig.mk 文件中, 是否有定义宏 BOARD_DEFAULT_CAMERA_HAL_VERSION, 如无定义, 请在文件末添加如下:
BOARD_DEFAULT_CAMERA_HAL_VERSION := 3.3
(android 9.0 以前 Sdk 发布可能带有 hal1 和 hal3 两套源码, 该两套源码编译目标是相同的, 所以同时

编译会产生冲突，因此在编译时加一个宏来判断编译哪一套 hal 源码，如下：`ifeq (1,$(strip $(shell expr $(BOARD_DEFAULT_CAMERA_HAL_VERSION) \>= 3.0)))`，只有当该宏 `>=3.0` 时才会编译 hal3.)

- 1) 在<android_root> 目录

```
$ source build/envsetup.sh
```

```
$ lunch
```

- 2) 进入 hal3 源码目录

```
$ mma -j8
```

3.2 生成库：

- 2) Hal3 库： /vendor/lib<64>/hw/camera.rk30board.so

- 3) librkisp : /vendor/lib<64>/librkisp.so

- 4) 3a lib : /vendor/lib<64>/rkisp/<ae/awb/af>/

- 5) 配置文件： /vendor/etc/camera/

上述配置文件是通过预编译将<hal3>/etc/camera 中的文件 copy 到 android out 目录。 当修改源码编译后，只需 push camera.rk30board.so 即可， 如修改配置文件， 也只需要 push 相应配置文件。

3.3 运行：

1. 将需要更新的库或者 xm 配置文件 push 到板子相应的目录。

```
$ adb root && adb remount
```

```
$ adb push <hal3_camera>/etc/camera /vendor/etc/ (android version >= 8.0)
```

```
$ adb push <hal3_camera>/etc/camera /system/etc/ (android version < 8.0)
```

2. 重新启动 camera 服务进程

```
$ adb shell pkill camera && adb shell pkill provider
```

3. 通过如下命令查看 camera 是否加载成功。

```
$ adb shell dumpsys media.camera
```

4. 如果没有打印出 camera 相关信息(camera 正常信息有好几百行)，则加载失败。

此时：

5. 再次确认配置文件是否有 Push 到板子（普遍是这个问题，请再三确认）：

```
$ adb shell
```

```
$ cat /vendor/camera/camera3_profiles.xml //查看该文件是否是修改过后的文件，
```

\$ adb logcat|grep "E RkCamera" 查看是否是致命错误，定位分析。

6. 如果前三步都没有问题，底层驱动正常，可用 v4l2-ctl 抓到数据，此时 camera 应该可以打开了。

Camera 如果打不开，可以打开相关 camera log 的开关来定位问题。

```
$ adb shell setprop persist.vendor.camera.hal.debug 5
```

4. Dump 说明

为了方便调试，hal3 增加了几个属性值，可以将预览，录像，拍照等数据流直接 dump 到文件。以下是详细说明：

4.1 属性说明：

■ persist.vendor.camera.dump: 表示相关数据的 dump 开关，属性值对应不同数据流
例：

```
adb shell setprop persist.vendor.camera.dump 1 #dump preview
adb shell setprop persist.vendor.camera.dump 2 # dump video
adb shell setprop persist.vendor.camera.dump 4 # dump zsl
adb shell setprop persist.vendor.camera.dump 8 # dump jpeg
adb shell setprop persist.vendor.camera.dump 16 # dump raw
adb shell setprop persist.vendor.camera.dump 128 # dump pure data that not processed in hal
adb shell setprop persist.vendor.camera.dump 11 # dump preview + video + jpeg
```

■ persist.vendor.camera.dump.skip 属性表示跳过前面 n 帧
例：\$ adb shell setprop persist.vendor.camera.dump.skip 10
表示前面 10 帧不 dump

■ persist.vendor.camera.dump.gap 属性是表示 dump 帧的间隔
例：\$ adb shell setprop persist.vendor.camera.dump.gap 10
表示隔 10 帧 dump 一帧

■ persist.vendor.camera.dump.cnt 属性表示 dump 帧的总帧数
例：\$ adb shell setprop persist.vendor.camera.dump.cnt 100
表示总共只 dump 100 帧

■ persist.vendor.camera.dump.path 属性表示 dump 帧的路径
例：\$ adb shell setprop persist.vendor.camera.dump.path /data/dump/
表示 dump 的路径为 /data/dump/ （最后的” /” 不能省）

以下是一个完整例子，表示 dump 预览帧，前面 10 帧不 dump，每隔 10 帧 dump 一次，总共 dump 100 帧，路径为/data/dump/

```
adb shell setprop persist.vendor.camera.dump 1
adb shell setprop persist.vendor.camera.dump.skip 10
adb shell setprop persist.vendor.camera.dump.gap 10
adb shell setprop persist.vendor.camera.dump.cnt 100
adb shell setprop persist.vendor.camera.dump.path /data/dump/
```

4.2 未生成 dump 文件问题:

Dump 属性设置完成，打开相机，预览后，板子 /data/dump/目录下应该会有如下 dump 文件生成。

```
ek3399_all:/data/dump # ls
dump_1280x960_00000160_PREVIEW_0 dump_1280x960_00000274_PREVIEW_0 dump_1280x960_00000388_PREVIEW_0 dump_1280x960_00000502_PREVIEW_0 dump_1280x960_00000616_PREVIEW_0
dump_1280x960_00000161_PREVIEW_0 dump_1280x960_00000275_PREVIEW_0 dump_1280x960_00000389_PREVIEW_0 dump_1280x960_00000503_PREVIEW_0 dump_1280x960_00000617_PREVIEW_0
dump_1280x960_00000162_PREVIEW_0 dump_1280x960_00000276_PREVIEW_0 dump_1280x960_00000390_PREVIEW_0 dump_1280x960_00000504_PREVIEW_0 dump_1280x960_00000618_PREVIEW_0
dump_1280x960_00000163_PREVIEW_0 dump_1280x960_00000277_PREVIEW_0 dump_1280x960_00000391_PREVIEW_0 dump_1280x960_00000505_PREVIEW_0 dump_1280x960_00000619_PREVIEW_0
dump_1280x960_00000164_PREVIEW_0 dump_1280x960_00000278_PREVIEW_0 dump_1280x960_00000392_PREVIEW_0 dump_1280x960_00000506_PREVIEW_0 dump_1280x960_00000620_PREVIEW_0
```

如果未生成 dump 文件，请参照前面调试说明章节，打开 log 开关。并查看 log 是否有以下错误

```
$ adb logcat |grep "open file failed"
```

```
I RkCamera: <HAL> CameraBuffer: dumpImage filename is /data/dump_1280x960_00000015_PREVIEW_0
E RkCamera: <HAL> CameraBuffer: open file failed
```

该错误是由于 dump 路径无权限访问，或者不存在导致的。可以尝试以下步骤解决。

确认 dump 路径是否存在，不存在请更改目录，或创建目录

目录存在但依然无权限访问，可以使用如下命令暂时关闭 selinux

```
$ adb root && adb shell setenforce 0
```

5. 版本说明:

Hal3 版本获取:

通过读取属性值获取

```
$ adb shell getprop |grep cam.hal3.ver
```

也可以通过查看 logcat 获取

```
$ adb logcat |grep "Hal3 Release version"
```


6. 调试案例:

android.hardware.camera2.cts.RecordingTest#testBasicRecording

failed

```
android.hardware.camera2.cts.RecordingTest#testBasicRecording fail junit.framework.AssertionFailedError: Video size 176x144 for profile ID 2 must be one of the camera device supported video sizes
```

分析: 该问题是因为在 Camera3_profiles.xml 中没有配置相应分辨率

解决方法: 参照 2.3.2, 将相应 size(这里为 176x144) 添加到 scaler.availableStreamConfigurations、scaler.availableMinFrameDurations、scaler.availableStallDurations 中。如果 BLOB 不需要配置此分辨率, 对 BLOB 相关的流可不添加此分辨率。

android.hardware.camera2.cts.PerformanceTest#testMultipleCapture

failed

分析: 该项要求, fps range 中必需含一组固定帧率 fps [minfps, minfps], minfps 为 scaler.availableMinFrameDurations 中 YCbCr_420_888 的最大尺寸对应的帧率

```
1) testMultipleCapture(android.hardware.camera2.cts.PerformanceTest)
junit.framework.AssertionFailedError: Cam 0: Target FPS range of (18, 18) must be supported
    at junit.framework.Assert.fail(Assert.java:50)
    at junit.framework.Assert.assertTrue(Assert.java:20)
    at android.hardware.camera2.cts.testcases.Camera2SurfaceViewTestCase.getSuitableFpsRangeForDuration(Camera2SurfaceViewTestCase.java:857)
    at android.hardware.camera2.cts.PerformanceTest.testMultipleCapture(PerformanceTest.java:478)
    at java.lang.reflect.Method.invoke(Native Method)
    at android.test.InstrumentationTestCase.runMethod(InstrumentationTestCase.java:220)
    at android.test.InstrumentationTestCase.runTest(InstrumentationTestCase.java:205)
    at android.test.ActivityInstrumentationTestCase2.runTest(ActivityInstrumentationTestCase2.java:192)
    at junit.framework.TestCase.runBare(TestCase.java:134)
    at junit.framework.TestResult$1.protect(TestResult.java:115)
    at android.support.test.internal.runner.junit3.AndroidTestResult.runProtected(AndroidTestResult.java:73)
    at junit.framework.TestResult.run(TestResult.java:118)
    at android.support.test.internal.runner.junit3.AndroidTestResult.run(AndroidTestResult.java:51)
    at junit.framework.TestCase.run(TestCase.java:124)
    at android.support.test.internal.runner.junit3.NonLeakyTestSuite$NonLeakyTest.run(NonLeakyTestSuite.java:62)
    at junit.framework.TestSuite.runTest(TestSuite.java:243)
    at junit.framework.TestSuite.run(TestSuite.java:238)
    at android.support.test.internal.runner.junit3.DelegatingTestSuite.run(DelegatingTestSuite.java:97)
    at android.support.test.internal.runner.junit3.AndroidTestSuite.run(AndroidTestSuite.java:65)
    at android.support.test.internal.runner.junit3.JUnit38ClassRunner.run(JUnit38ClassRunner.java:115)
    at org.junit.runners.Suite.runChild(Suite.java:128)
    at org.junit.runners.Suite.runChild(Suite.java:27)
    at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
```



```

<scaler.availableMinFrameDurations value="BLOB,2592x1944,55555555,
    BLOB,1920x1080,55555555,
    BLOB,1280x960,33333333,
    BLOB,1280x720,33333333,
    BLOB,640x480,33333333,
    BLOB,320x240,33333333,
    BLOB,176x144,33333333,
    YCbCr_420_888,2592x1944,55555555,
    YCbCr_420_888,1920x1080,55555555,
    YCbCr_420_888,1280x960,33333333,
    YCbCr_420_888,1280x720,33333333,
    YCbCr_420_888,640x480,33333333,
    YCbCr_420_888,320x240,33333333,
    YCbCr_420_888,176x144,33333333,
    IMPLEMENTATION_DEFINED,2592x1944,55555555,
    IMPLEMENTATION_DEFINED,1920x1080,55555555,
    IMPLEMENTATION_DEFINED,1280x960,33333333,
    IMPLEMENTATION_DEFINED,1280x720,33333333,
    IMPLEMENTATION_DEFINED,640x480,33333333,
    IMPLEMENTATION_DEFINED,320x240,33333333,
    IMPLEMENTATION_DEFINED,176x144,33333333" />

```

Durations 为 55555555 时，帧率为 18

解决方法：修改 `<control.acAvailableTargetFpsRanges value="15,18,18,18,18,30,30,30"/>`

android.hardware.cts.CameraTest#testVideoSnapshot failed

android.hardware.cts.CameraTest#testJpegThumbnailSize Failed

android.hardware.cts.CameraTest#testVideoSnapshot	fail	junit.framework.AssertionFailedError
android.hardware.cts.CameraTest#testJpegThumbnailSize	fail	junit.framework.AssertionFailedError

分析：失败的 stack 如下：

```

There was 1 failure:
1) testVideoSnapshot (android.hardware.cts.CameraTest)
junit.framework.AssertionFailedError
    at junit.framework.Assert.fail (Assert.java:48)
    at junit.framework.Assert.assertTrue (Assert.java:20)
    at junit.framework.Assert.assertTrue (Assert.java:27)
    at android.hardware.cts.CameraTest.testJpegThumbnailSizeByCamera (CameraTest.java:795)
    at android.hardware.cts.CameraTest.testVideoSnapshotByCamera (CameraTest.java:3075)
    at android.hardware.cts.CameraTest.testVideoSnapshot (CameraTest.java:3014)
    at java.lang.reflect.Method.invoke (Native Method)
    at android.test.InstrumentationTestCase.runMethod (InstrumentationTestCase.java:220)
    at android.test.InstrumentationTestCase.-wrap0 (InstrumentationTestCase.java)
    at android.test.InstrumentationTestCase$2.run (InstrumentationTestCase.java:195)
    at android.app.Instrumentation$SyncRunnable.run (Instrumentation.java:1950)
    at android.os.Handler.handleCallback (Handler.java:755)
    at android.os.Handler.dispatchMessage (Handler.java:95)
    at android.os.Looper.loop (Looper.java:154)
    at android.app.ActivityThread.main (ActivityThread.java:6141)
    at java.lang.reflect.Method.invoke (Native Method)
    at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run (ZygoteInit.java:912)
    at com.android.internal.os.ZygoteInit.main (ZygoteInit.java:802)

FAILURES!!!
Tests run: 1, Failures: 1
android.hardware.cts.CameraTest#testVideoSnapshot failed!!!

```

```
private void testJpegThumbnailSizeByCamera(boolean recording,
    int recordingWidth, int recordingHeight) throws Exception {
    // Thumbnail size parameters should have valid values.
    Parameters p = mCamera.getParameters();
    Size size = p.getJpegThumbnailSize();
    assertTrue(size.width > 0 && size.height > 0);
    List<Size> sizes = p.getSupportedJpegThumbnailSizes();
}
```

由 stack 查找源码，可知道，是由于 thumb size 为 0 引起的。通过如下命令确认，发现确实在 api1 中上报的 thumbnail size static meta 确实为 0，

```
bob@ubuntu:~$ adb shell dumpsys media.camera|grep thumb
    jpeg-thumbnail-height: 0
    jpeg-thumbnail-quality: 90
    jpeg-thumbnail-size-values: 0x0,160x120,320x180,320x240
    jpeg-thumbnail-width: 0
```

但从 jpeg-thumbnail-size-values 可知，我们上报的 support size 有四组，0x0, 160x120, 320x180, 320x240。那为什么上报的 width、height 为 0 呢，在 framework 中查看代码发现，如果最大一组的 thumbnail size 与上报的最大拍照尺寸宽高比不同，则上报 thumb width 与 thumb height 设置为 0

解决方法：修改最大一组的 thumbnail size 使其与最大拍照尺寸宽高比相同（thumbnail size 大小不能超过 320x240）

android.hardware.camera2.cts.StillCaptureTest#testJpegExif failed

```
android.hardware.camera2.cts.StillCaptureTest#testJpegExif fail java.lang.Exception: Test failed for camera 0. Jpeg must have thumbnail for thumbnail size 320x240
```

分析：从 Cts 源码看，是因为返回去的拍照 Buffer 没有 thumbnail，因此查看 hal 最后一次编码过程，发现是因为编码输出 buffer size 太小，导致编码失败。查看 camera3_profiles.xml 中 <jpeg.maxSize value="4967424"/> <!-- 2112*1568*1.5 --> 发现此处限制了 jpeg out buffer 的大小。

解决办法：修改 xml 中 jpeg.maxsize 为：

```
<jpeg.maxSize value="19267584"/> <!-- 4096*3136*1.5 -->
```

1. 曝光很暗，预览非常黑，遇到过的几种情况。
 - a) raw sensor 设置成了 soc sensor 导致 3a 未运行
设置为 <sensorType value="SENSOR_TYPE_RAW"/> 即可
 - b) hal 与 camera engine 版本不匹配
使用最新版本编译，更新相应的 xml 和库文件
 - c) sensor.info.exposureTimeRange 最大值设置太小。
(参照 2.3 节) 修改如下设置，恢复正常曝光。
<sensor.info.exposureTimeRange value="0, 66666666"/>

该值设置的是 sensor 支持曝光时长的属性，单位是 ns，最大曝光时间和 app 设置的 fpsRange 最小帧率会共同限制 aec 算法中算出的曝光值。上述值对应的最小帧率是 15 帧，如果需要支持更小的帧率，可以相应改大该值。

修改 FOV 中的 lens.info.availableFocalLengths 值后导致 CTS 几项测试不过问题

问题描述：

修改 camera3_profiles.xml 中的两个 Sensor 的<lens.info.availableFocalLengths value="2.04"/>值一个改成<lens.info.availableFocalLengths value="3.73"/>

另外一个改成：<lens.info.availableFocalLengths value="3.04"/>后

测试 CTS 出现如下错误：

Test	Result	Details
android.hardware.camera2.cts.StillCaptureTest#testFocalLengths	fail	java.lang.Exception: There were 2 errors:
android.hardware.camera2.cts.StillCaptureTest#testJpegExif	fail	java.lang.Exception: There were 6 errors:
android.hardware.cts.CameraTest#testJpegExif	fail	junit.framework.AssertionFailedError: expected <2.5999999046325684> but was: <3.57>

android.hardware.camera2.cts.StillCaptureTest#testFocalLengths

android.hardware.camera2.cts.StillCaptureTest#testJpegExif

错误信息：

TestRunner: java.lang.Exception: There were 6 errors:

Test failed for camera 1: Focal length should match (expected = 3.73, actual = 3.04, tolerance = 0.001)

Test failed for camera 1: Exif focal length should match capture result (expected = 3.73, actual = 3.04, tolerance = 0.001)

问题分析：

SDK 中 fov 测试代码位置

SDK/cts/apps/CtsVerifier/src/com/android/cts/verifier/camera/fov

解决办法：

如下修改：前面传给 JPEGEncode 的信息错误了；直接就传 CameraID 为：0

```
--- a/psl/rkisp1/workers/PostProcessPipeline.cpp
```

```
+++ b/psl/rkisp1/workers/PostProcessPipeline.cpp
```

```
@@@ -1037,7 +1037,7 @@ PostProcessUnitJpegEnc::prepare(const FrameInfo& outfmt, int bufNum) {
```

```
    if (!mJpegTask.get()) {
```

```
        LOGI("Create JpegEncodeTask");
```

- mJpegTask.reset(new JpegEncodeTask(0)); // ignore camId
- + mJpegTask.reset(new JpegEncodeTask(mPipeline->getCameraId())); // ignore camId

android.hardware.cts.CameraTest#testJpegExif

问题解决提交点:

commit f8e90ea13add9037f97747c20dc86003203a87e6

Author: Bob <bob.fu@rock-chips.com>

Date: Thu Mar 28 15:39:29 2019 +0800

fix exif data error

CTS: android.hardware.camera2.cts.StillCaptureTest#testFocalLengths may failed due to the exif info mismatch. JpegEncodeTask should hold the correct cameraId but always get 0. fix it.

Change-Id: I5c775cc13708169a30e222a2dbbd1da6e3e6d51d

Signed-off-by: Bob bob.fu@rock-chips.com

对比查看当前代码是否包含该提交点;

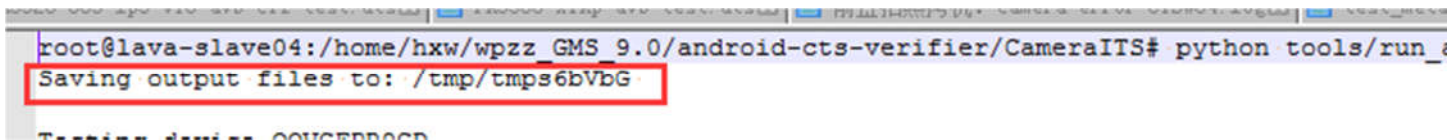
ITS 常见 Failed 项及解决办法

ITS 测试相关

Camera 测试 ITS 输出的 LOG 位置:

参考 ITS 测试文档: These files are all saved to a new temporary directory, the path of which is printed when the run_all_tests.py script begins

开始测试打印出 LOG 保存的位置如如图:



```
root@lava-slave04:/home/hxw/wpzz_GMS_9.0/android-cts-verifier/CameraITS# python tools/run_all_tests.py
Saving output files to: /tmp/tmps6bVbG
```

即 PC 中的/tmp/tmps6bVbG 保存了测试 LOG, 用于 debug



```
root@lava-slave04:/tmp/tmps6bVbG/0# ls
scene0 scene1 scene1.jpg scene2 scene2.jpg scene3 scene3.jpg scene4 scene4.jpg scene4.jpg
root@lava-slave04:/tmp/tmps6bVbG/0#
```


Camera 测试 ITS 对应的 Python 脚本位置:

CameraITS\tests\scene*目录下有对应测试项 Python 脚本:

例如: test_meta 这项对应的脚本为:

CameraITS\tests\scene0 中的对应测试项 Python 脚本: test_metadata.py

```
TS_9.0/android-cts-verifier/CameraITS$ ls tests/scene0/
test_capture_result_dump.py  test_jitter.py  test_param_sensitivity_burst.p
test_gyro_bias.py          test_metadata.py  test_read_write.py
```

ITS 测试出现测试 test_meta 失败问题分析示例

问题描述:

```
SKIP scene0/test_gyro_bias [1.8s]
SKIP scene0/test_jitter [1.8s]
FAIL scene0/test_metadata [2.1s]
```

问题分析:

查看 ITS 测试 LOG:

是计算的 pixel_pitch_height 为: 0.00um, 异常了;

```
11 > props.has_key("android.info.supportedHardwareLevel")
12 > props["android.info.supportedHardwareLevel"] is not None
13 > props["android.info.supportedHardwareLevel"] in [0,1,2,3]
14 > md.has_key("android.sensor.rollingShutterSkew")
15 > md["android.sensor.rollingShutterSkew"] is not None
16 > props.has_key("android.sensor.info.timestampSource")
17 > props["android.sensor.info.timestampSource"] is not None
18 > props["android.sensor.info.timestampSource"] in [0,1]
19 > props.has_key("android.scaler.croppingType")
20 > props["android.scaler.croppingType"] is not None
21 > props["android.scaler.croppingType"] in [0,1]
22 pixel_pitch WxH: 1.75 um, 1.75 um
23 field of view: 64.0 degrees
24

11 Passed> props.has_key("android.info.supportedHardwareLevel")
12 Passed> props["android.info.supportedHardwareLevel"] in [0,1,2,3]
13 Passed> md.has_key("android.sensor.rollingShutterSkew")
14 Passed> md["android.sensor.rollingShutterSkew"] is not None
15 Passed> props.has_key("android.sensor.info.timestampSource")
16 Passed> props["android.sensor.info.timestampSource"] is not None
17 Passed> props["android.sensor.info.timestampSource"] in [0,1]
18 Passed> props.has_key("android.scaler.croppingType")
19 Passed> props["android.scaler.croppingType"] is not None
20 Passed> props["android.scaler.croppingType"] in [0,1]
21 Passed> props["android.scaler.croppingType"] in [0,1]
22 Assert pixel_pitch WxH: 2.19 um, 0.00 um
23
```

查看对应的 Python 脚本:

CameraITS\tests\scene0 中的对应测试项 Python 脚本: test_metadata.py 如下

计算 pixel_pitch_h 如下:

```
if not its.caps.legacy(props):
    # Test: pixel_pitch, FOV, and hyperfocal distance are reasonable
    fmts = props["android.scaler.streamConfigurationMap"]["availableStreamConfigurations"]
    fmts = sorted(fmts, key=lambda k: k["width"]*k["height"], reverse=True)
    sensor_size = props["android.sensor.info.physicalSize"]
    pixel_pitch_h = (sensor_size["height"] / fmts[0]["height"]) * 1E3
    pixel_pitch_w = (sensor_size["width"] / fmts[0]["width"]) * 1E3
    print "Assert pixel_pitch WxH: %.2f um, %.2f um" % (pixel_pitch_w, pixel_pitch_h)
    assert 0.9 <= pixel_pitch_w <= 10
    assert 0.9 <= pixel_pitch_h <= 10
    assert 0.333 <= pixel_pitch_w/pixel_pitch_h <= 3.0
```

怀疑是计算的时候把<sensor.info.physicalSize value="1.4,1.0"/>中 height 的 1.0 当做整数计算,

导致最后算出的 pixel_pitch_h 为 0;

解决办法:

```
<sensor.info.physicalSize value="1.4,1.0"/> <!-- 4224x1.12um 3136x1.12um -->
```

改成

```
<sensor.info.physicalSize value="1.4,1.04"/> <!-- 4224x1.12um 3136x1.12um -->
```