密级状态：绝密(　　)　　秘密(　　)　　内部资料()　　公开(√)

RK_ISP10_Camera_User_Manual

s 文件状态： [　] 草稿 [　] 正式发布 [√] 正在修改	文件标识：	
	当前版本：	2.1
	作　　者：	邓达龙、钟以崇、欧阳亚凤、张云龙、叶志明、黄春成
	完成日期：	2018-5-22

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

历 史 版 本

版本	日 期	描 述	作 者	审核
V1.0	2015-3-17	建立文档，主要介绍 RK3288/RK3368Camera 的注意事项	张云龙	
V2.0	2016-8-19	添加 RK3399 Camera 的注意事项	黄春成	
V2.1	2017-10-24	添加 camera 驱动移植指导	张云龙	
V2.2	2018-05-22	完善 camera 驱动移植指导	陈泽发	

目录

1. 文档适用平台

该文档适用于 RK3288、RK3368 和 RK3399 平台。

1.1. 平台说明

1) RK3288

两个 PHY， PHY0 以及 PHY1 都支持 1lane、2lane、4lane，最大支持 13M pixel raw sensor。

2) RK3368

一个 PHY，PHY 支持 1lane、2lane、4lane，最大支持 8M pixel raw sensor。

3) RK3399

两个 PHY，PHY 支持 1lane、2lane、4lane，最大支持 13M pixel raw sensor。

2. 硬件说明

2.1. DVP SOC Camera Sensor

1) RK3288

建议将该类 Sensor 输出的 YUV 数据 bit0-bit7 对应连接至 RK3288 CIF_D2 - CIF_D9

2) RK3368

建议将该类 Sensor 输出的 YUV 数据 bit0-bit7 对应连接至 RK3368 CIF_D4 - CIF_D11

3) RK3399

建议将该类 Sensor 输出的 YUV 数据 bit0-bit7 对应连接至 RK3399 CIF_D0 - CIF_D7

2.2. MIPI Camera Sensor

(模组的 MIPI Lane 数 >= PHY 支持的 MIPI Lane 数）满足这一条件都可以连接到对应的 PHY，但是最后实际使用的 Lane 数以 PHY 支持的 Lane 数为准；

MIPI Camera Sensor 在选用时，建议事先查阅 RockChip 的认证列表：《RKISPV1_Camera_Module_AVL》，确认是否调试通过.

2.3. 2 个 Camera Sensor 同时工作的限制说明

1) RK3288、RK3368

1、2 个 Sensor 只能有一个是 RAW Sensor；

2、必须有一个是 MIPI Sensor；

2) RK3399

1、2 个 Sensor 都为 RAW Sensor 或者 mipi sensor；

2.4. RAW Camera Sensor 选型说明

1、事先获取 RockChip 的认证列表：《RKISPV1_Camera_Module_AVL》；

2、列表中已经有相关型号，并且状态显示 Ready，那么建议按照列表中的模组配置信息让模组厂进行打样；

3、列表中没有相关型号，或是想选择不同配置（镜头、VCM）的模组，那么建议填写《RockChip 摄像头模组调试需求申请表》，同时发给 RockChip。

注：RAW Camera Sensor 调试周期在 4 周左右；
模组配置更换 调试周期在 3 周左右；

3. 文件目录说明

3288 Android：

```
|
| hardware\rk29\camera
    |
    |CameraHal                    CameraHal 源码
    |Config                       Camera 配置文件信息及 isp 库
    |SiliconImage                 ISP 库相关头文件信息
      |
      |isi\drv                    Sensor 驱动源码
        |
        |OV8825\calib             Sensor 模组 tunning 参数
```

3368 Android：

```
|
| hardware\rockchip\camera
    |
    |CameraHal                    CameraHal 源码
    |Config                       Camera 配置文件信息及 isp 库
    |SiliconImage                 ISP 库相关头文件信息
      |
      |isi\drv                    Sensor 驱动源码
        |
        |OV8825\calib             Sensor 模组 tunning 参数
```

Kernel:

```
|
|drivers\media\video\rk_camsys     CamSys 驱动源码
|include\media\camsys_head.h
```

3399 Android：

```
|
| hardware\rockchip\camera
    |
    |CameraHal                    CameraHal 源码
    |Config                       Camera 配置文件信息及 isp 库
    |SiliconImage                 ISP 库相关头文件信息
      |
      |isi\drv                    Sensor 驱动源码
        |
        |OV8825\calib             Sensor 模组 tunning 参数
```

Kernel:

```
|
```

|drivers\media\video\rk_camsys　　CamSys 驱动源码
|include\media\camsys_head.h

4. 版本说明

4.1. 版本获取方式

在机器的 shell 中执行以下命令：

```
root@rk3288:/ # getprop
[sys_graphic.cam_camboard.ver]: [0.2.0]          支持 cam_board.xml  的版本
[sys_graphic.cam_drv_camsys.ver]: [0.8.0]        camsys 驱动版本
[sys_graphic.cam_hal.ver]: [0.9.0]               CameraHal 版本
[sys_graphic.cam_isi.ver]: [0.1.0]               ISI 接口版本
[sys_graphic.cam_libisp.ver]: [0.4.0]            ISP 库版本
[sys_graphic.OV8825.ver]: [0.9.0]              sensor 驱动版本号
```

由于各个源码以及库之间版本需要匹配使用，所以在代码中已经做了版本校验规则，如果出现 panic 等信息，麻烦先关注是否是版本之间的不匹配导致！！

例如：

```
D/CameraHal( 1739):  CamSys_Head.h Version Check:
D/CameraHal( 1739):     Kernel camsys_head.h: v0.6.0
D/CameraHal( 1739):     Kernel camsys_drv :   v0.8.0
D/CameraHal( 1739):     CameraHal camsys_head.h : v0.7.0
D/CameraHal( 1739):
D/CameraHal( 1739):
D/CameraHal( 1739):
F/CameraHal( 1739): static int
camera_board_profiles::RegisterSensorDevice(rk_cam_total_info*):
F/CameraHal( 1739): VERSION-WARNING: camsys_head.h version isn't
match in Kernel and CameraHal
```

5. 如何注册 DVP/MIPI Sensor

注册 DVP/MIPI Sensor 方式通过填写 cam_board.xml 来实现，该文件使用简要说明如下：

注：如果机器中没有 DVP/MIPI Sensor，删除 cam_board.xml 文件即可；

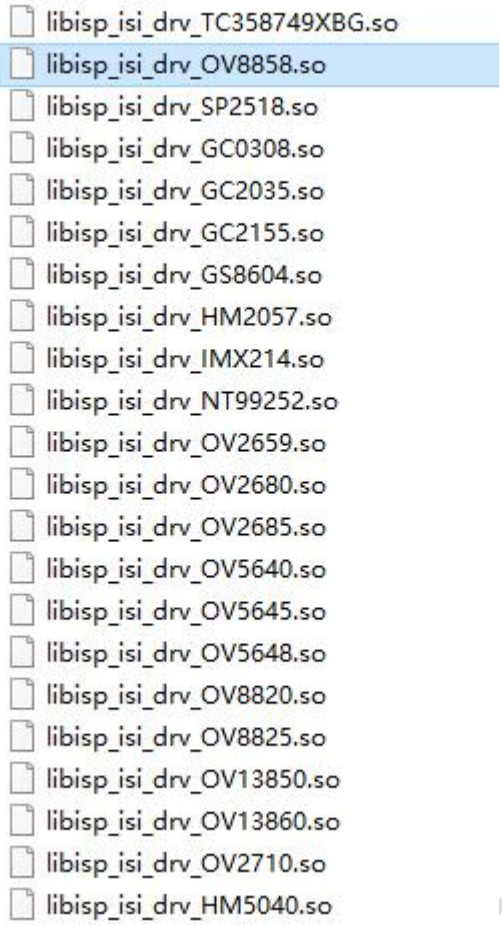
```
<BoardXmlVersion version="v0.2.0">
```

以上标识的为当前 xml 文件的版本号，如果与 sys_graphic.cam_camboard.ver 不一致，可能导致错误，麻烦更新 cam_board.xml。

5.1. Sensor 注册信息

```
<SensorName name="OV8858" ></SensorName>
```

填写 Sensor 名字，该名字必须与 Sensor 驱动的名字一致，目前提供的 Sensor 驱动如下：



<SensorLens name="LG-9569A2"></SensorLens>

填写模组所配置的镜头型号，镜头型号必须根据模组实际配置填写，这个将直接影响到最后的成像质量。

注意：非 OTP 模组及有 OTP 但读取不到 lens ID 则以这里配置的为准；有 OTP 且能读取到 lens ID 则以读取到的镜头型号为准。

目前 tuning 过的 sensor 及可配置镜头型号如下：

OV8825：

LG-5008A7

OV8820：

LG-5008A7

OV8858：

SUNNY-3813A
LG-9569A2
R5AV08

OV5648：

CHT-842B-MD
XY-LE001B1

<SensorDevID IDname="CAMSYS_DEVID_SENSOR_1A"></SensorDevID>

填写 Sensor 软件 ID，注册的 ID 只需要不一致即可，可填写以下值：

CAMSYS_DEVID_SENSOR_1A

CAMSYS_DEVID_SENSOR_1B
CAMSYS_DEVID_SENSOR_2

<SensorHostDevID busnum="CAMSYS_DEVID_MARVIN" ></SensorHostDevID>
填写采集控制器名称，目前只支持填写：
CAMSYS_DEVID_MARVIN

<SensorI2cBusNum busnum="3"></SensorI2cBusNum>
填写 Sensor 所连接的主控 I2C 通道号

<SensorI2cAddrByte byte="2"></SensorI2cAddrByte>
填写 Sensor 寄存器地址长度，单位：Byte

<SensorI2cRate rate="100000"></SensorI2cRate>
填写 Sensor 的 I2C 频率，单位：Hz

<SensorMclk mclk="24000000" delay="1000"></SensorMclk>
填写 Sensor 输入时钟频率， 单位:Hz

<SensorAvdd name="NC" min="28000000" max="28000000" delay="0"></SensorAvdd>
填写 Sensor AVDD 的 PMU LDO 名称，如果不是连接到 PMU，那么只需填写 NC

<SensorDovdd name="NC" min="18000000" max="18000000"
delay="5000"></SensorDovdd>
填写 Sensor DOVDD 的 PMU LDO 名称，如果不是连接到 PMU，那么只需填写 NC，注意 min 以及 max 值必须填写，这决定了 Sensor 的 IO 电压；RK3399 中有 delay，调整上电时序；

<SensorDvdd name="NC" min="12000000" max="12000000" delay="0"></SensorDvdd>
填写 Sensor DVDD 的 PMU LDO 名称，如果不是连接到 PMU，那么只需填写 NC

<SensorGpioPwdn ioname="RK30_PIN1_PC2" active="0"
delay="0"></SensorGpioPwdn>
填写 Sensor PowerDown 引脚，直接填写名称即可， active 填写休眠的有效电平；
RK3399 中 phy0、phy1 有单独的“SensorGpioPwdn ”，分别 为“SensorGpioPwdn 0”、“SensorGpioPwdn 1”；

<SensorGpioRst ioname="NC" active="0" delay="1000"></SensorGpioRst>
填写 Sensor Reset 引脚，直接填写名称即可， active 填写复位的有效电平

<SensorGpioPwen ioname="NC" active="1" delay="1000"></SensorGpioPwen>
填写 Sensor Power 引脚，直接填写名称即可， active 填写电源有效电平

<SensorFacing facing="front"></SensorFacing>

填写 Sensor 作为前置还是后置，可填写如下值：
front
back

<SensorInterface mode="CCIR601"></SensorInterface>
填写 Sensor 的接口方式，可填写如下值：
CCIR601
CCIR656,
MIPI,
SMIA

<SensorMirrorFlip mirror="0"></SensorMirrorFlip>
暂不支持

<SensorOrientation orientation="0"></SensorOrientation>
填写 Sensor 的角度信息

<SensorPowerupSequence seq="1234"></SensorPowerupSequence>
暂不支持
<SensorFovParemeter h="60.0" v="60.0"></SensorFovParemeter>
FOV 配置选项， h 代表水平视角度数，v 代表垂直视角度数
理论上,FOV 值可以由模组规格书中获得,由于可能不精确,在测试 Cts_Verify FOV 选项时，可以先测试一张全分辨率照片，查看具体的 FOV 值，然后将测试出的 FOV 值重新填入该处，重新烧写固件测试。
<SensorAWB_Frame_Skip fps="15"></SensorAWB_Frame_Skip>
设置 Camera 进入时，过滤 awb 不稳定的最大帧数
如果 sensor 帧率可以达到 30 帧，建议设置成 15 帧；
如果 sensor 帧率只在 15 帧左右，建议跳桢数减少，避免刚进入黑屏时间较长。

DVP Sensor：
<SensorPhy phyMode="CamSys_Phy_Cif" sensor_d0_to_cif_d ="2" cif_num="0" sensorFmt="CamSys_Fmt_Raw_10b"></SensorPhy>
phyMode：
Sensor 接口硬件连接方式，可填写如下值：
CamSys_Phy_Cif

sensor_d0_to_cif_d:
Sensor DVP 输出数据位 D0 对应连接的主控 DVP 接口的数据位号码

cif_num:
Sensor DVP 连接到主控 DVP 接口编号

sensorFmt：
Sensor 输出的数据格式，目前支持 CamSys_Fmt_Raw_10b 和 CamSys_Fmt_Raw

_12b

MIPI Sensor：
<SensorPhy phyMode="CamSys_Phy_Mipi" lane="1" phyIndex="0" sensorFmt="CamSys_Fmt_Raw_10b"></SensorPhy>

phyMode：
Sensor 接口硬件连接方式，可填写如下值：
CamSys_Phy_Mipi

lane:
Sensor mipi 接口数据通道数

phyindex:
Sensor mipi 连接的主控 mipi phy 编号
RK3368 仅支持 phyIndex="0"

sensorFmt
Sensor 输出数据格式，目前仅支持 CamSys_Fmt_Raw_10b

5.2. VCM 注册信息

<VCMDrvName name="NC"></VCMDrvName>
填写马达驱动 IC 的名称，如果 Sensor 集成马达驱动 IC 的话，请填写：BuiltInSensor

<VCMName name="NC"></VCMName>
填写马达的名称

<VCMI2cBusNum busnum="0"></VCMI2cBusNum>
填写马达驱动 IC 的连接的主控 I2C 通道号，一般与 Sensor 同一个通道

<VCMI2cAddrByte byte="0"></VCMI2cAddrByte>
填写马达驱动 IC 的 i2c 地址字节数

<VCMI2cRate rate="0"></VCMI2cRate>
填写马达驱动 IC 的 i2c 速率

<VCMVdd name="NC" min="0" max="0"></VCMVdd>
填写模组上连接 AF_VCC(马达电源)的 PMU_LDO 名称

<VCMGpioPwdn ioname="NC" active="0"></VCMGpioPwdn>
填写模组上马达驱动 IC 的休眠使能 IO，一般与 Sensor 的休眠使能 IO 一致

<VCMGpioPower ioname="NC" active="0"></VCMGpioPower>
填写使能模组 AF_VCC 的使能 IO

<VCMCurrent start="20" rated="80" vcmmax="100" stepmode="13" drivermax="100"></VCMCurrent>

填写马达的电流参数：

start: 马达的启动电流

rated: 马达的额定电流

vcmmax: 马达的最大电流

stepmode: 马达驱动 ic 的电流输出方式，该指标关系到马达的移动速度，麻烦参考驱动 icdatasheet；

drivermax: 马达驱动 ic 的最大输出电流

注意事项：start、rated、stepmode 这 3 项指标有可能会导致马达在对焦过程中的异响问题；

如果出现模组对焦远处无法清晰，近处可以清晰，麻烦确认启动电流相对马达实际启动电流是否配置过大；

5.3. 软件功能配置信息

```
<AWB>
    <AWB_Auto support="1"></AWB_Auto>
    <AWB_Incandescent support="1"></AWB_Incandescent>
    <AWB_Fluorescent support="1"></AWB_Fluorescent>
    <AWB_Warm_Fluorescent support="1"></AWB_Warm_Fluorescent>
    <AWB_Daylight support="1"></AWB_Daylight>
    <AWB_Cloudy_Daylight support="1"></AWB_Cloudy_Daylight>
    <AWB_Twilight support="1"></AWB_Twilight>
    <AWB_Shade support="1"></AWB_Shade>
</AWB>
```

配置 AWB 模式

1: 使能该功能

0: 屏蔽该功能

```
<Sence>
    <Sence_Mode_Auto support="1"></Sence_Mode_Auto>
    <Sence_Mode_Action support="1"></Sence_Mode_Action>
    <Sence_Mode_Portrait support="1"></Sence_Mode_Portrait>
    <Sence_Mode_Landscape support="1"></Sence_Mode_Landscape>
    <Sence_Mode_Night support="1"></Sence_Mode_Night>
    <Sence_Mode_Night_Portrait support="1"></Sence_Mode_Night_Portrait>
    <Sence_Mode_Theatre support="1"></Sence_Mode_Theatre>
    <Sence_Mode_Beach support="1"></Sence_Mode_Beach>
    <Sence_Mode_Snow support="1"></Sence_Mode_Snow>
    <Sence_Mode_Sunset support="1"></Sence_Mode_Sunset>
    <Sence_Mode_Steayphoto support="1"></Sence_Mode_Steayphoto>
    <Sence_Mode_Pireworks support="1"></Sence_Mode_Pireworks>
    <Sence_Mode_Sports support="1"></Sence_Mode_Sports>
    <Sence_Mode_Party support="1"></Sence_Mode_Party>
    <Sence_Mode_Candlelight support="1"></Sence_Mode_Candlelight>
```

```
    <Sence_Mode_Barcode support="1"></Sence_Mode_Barcode>
    <Sence_Mode_HDR support="1"></Sence_Mode_HDR>
</Sence>
```

配置 Scence 功能，暂不支持

```
<Effect>
    <Effect_None support="1"></Effect_None>
    <Effect_Mono support="1"></Effect_Mono>
    <Effect_Solarize support="1"></Effect_Solarize>
    <Effect_Negative support="1"></Effect_Negative>
    <Effect_Sepia support="1"></Effect_Sepia>
    <Effect_Posterize support="1"></Effect_Posterize>
    <Effect_Whiteboard support="1"></Effect_Whiteboard>
    <Effect_Blackboard support="1"></Effect_Blackboard>
    <Effect_Aqua support="1"></Effect_Aqua>
</Effect>
```

配置 Effect 功能，暂不支持

```
<FocusMode>
    <Focus_Mode_Auto support="1"></Focus_Mode_Auto>
```

暂不支持

```
    <Focus_Mode_Infinity support="1"></Focus_Mode_Infinity>
```

暂不支持

```
    <Focus_Mode_Marco support="1"></Focus_Mode_Marco>
```

暂不支持

```
    <Focus_Mode_Fixed support="1"></Focus_Mode_Fixed>
```

暂不支持

```
    <Focus_Mode_Edof support="1"></Focus_Mode_Edof>
```

暂不支持

```
    <Focus_Mode_Continuous_Video support="1"></Focus_Mode_Continuous_Video>
```

配置是否使能录像时预览界面的连续对焦功能

1： 使能该功能

0： 屏蔽该功能

```
    <Focus_Mode_Continuous_Picture
support="1"></Focus_Mode_Continuous_Picture>
```

配置是否使能拍照预览界面的连续对焦功能

1： 使能该功能

0： 屏蔽该功能

```
</FocusMode>

<FlashMode>
    <Flash_Mode_Off support="1"></Flash_Mode_Off>
    <Flash_Mode_On support="1"></Flash_Mode_On>
    <Flash_Mode_Torch support="1"></Flash_Mode_Torch>
```

```
    <Flash_Mode_Auto support="1"></Flash_Mode_Auto>
    <Flash_Mode_Red_Eye support="1"></Flash_Mode_Red_Eye>
</FlashMode>
```

配置 Flash 功能，暂不支持

```
<AntiBanding>
    <Anti_Banding_Auto support="1"></Anti_Banding_Auto>
    <Anti_Banding_50HZ support="1"></Anti_Banding_50HZ>
    <Anti_Banding_60HZ support="1"></Anti_Banding_60HZ>
    <Anti_Banding_Off support="1"></Anti_Banding_Off>
</AntiBanding>
```

配置 AntiBanding 功能，暂不支持

```
<HDR support="0"></HDR>
```

配置 HDR 功能，暂不支持

```
<ZSL support="0"></ZSL>
```

配置 ZSL 功能，暂不支持

```
<DigitalZoom support="1"></DigitalZoom>
```

配置是否使能数码变焦功能

1： 使能该功能

0： 屏蔽该功能

```
<Continue_SnapShot support="1"></Continue_SnapShot>
```

配置是否使能连拍功能

1： 使能该功能

0： 屏蔽该功能

```
<InterpolationRes resolution="0"></InterpolationRes>
```

配置插值分辨率，目前支持的插值像素 1M/2M/3M/5M/8M。

比如想插值到 5M，那么设置 resolution="5000000"。

```
<PreviewSize width="0" height="0"></PreviewSize>
```

配置客户强制需求的预览分辨率，一般来说，宽高各设置成 0，由系统来进行选择；但是有可能系统选择出来的分辨率帧率过低，那么可以指定你所需要的分辨率；

注：目前 ov8825，建议将该项设置成 1920x1080；

```
<FaceDetect support="1" MaxNum="1"></FaceDetect>
```

配置是否支持人脸检测功能

1： 使能该功能

0： 屏蔽该功能

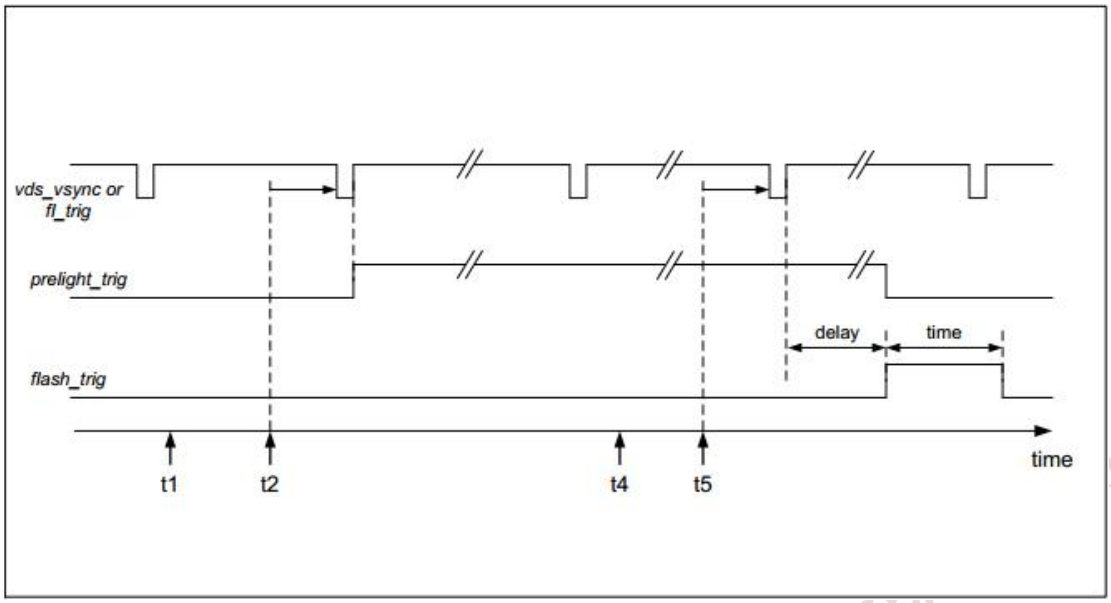
```
<Cproc support="1" contrast="1.1" saturation="1.0" hue="0"
```

brightness="0"></Cproc>
配置是否调整色彩效果；
1： 使能该功能
0： 屏蔽该功能
Contras(对比度)：（0.0， 1.992）
Saturation(饱和度)：（0.0， 1.992）
Hue（色相）：（-90 ， 87.188）
Brightness（亮度）：（-128 ， 127）

<Gammaout support = "0" gamma = "1.0" offset = "0"></Gammaout>
配置 gamma 值；

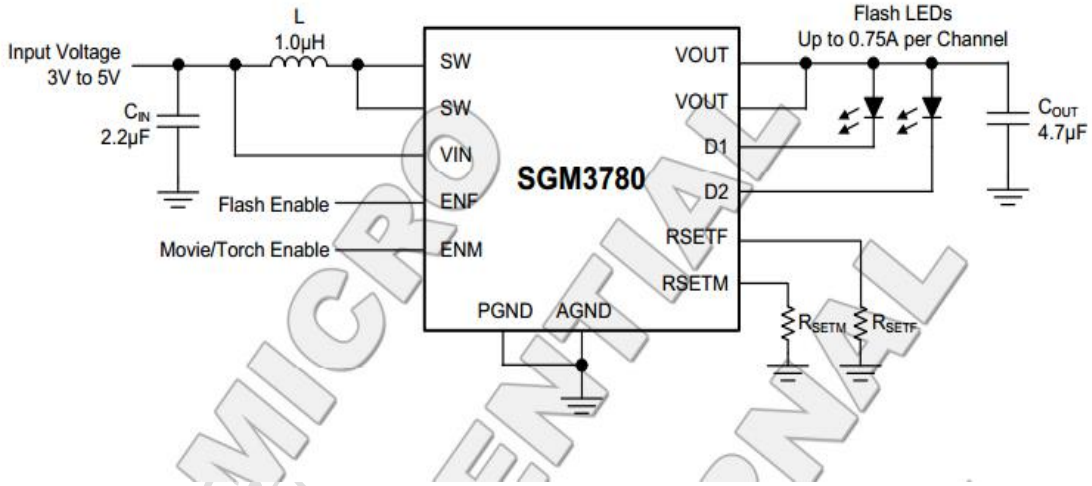
5.4. FLASH 注册信息

<FlashName name="Internal"></FlashName>
Flash 的名称，采用默认值
<FlashI2cBusNum busnum="0"></FlashI2cBusNum>
暂不支持
<FlashI2cAddrByte byte="0"></FlashI2cAddrByte>
暂不支持
<FlashI2cRate rate="0"></FlashI2cRate>
暂不支持
<FlashTrigger ioname="NC" active="0"></FlashTrigger>
填写 ISP 的 FLASHTRIGOUT 使能的有效电平
rk3288：对应 GPIO7-B5
rk3368：对应 GPIO3-C4
rk3399：对应 GPIO1-A3
<FlashEn ioname="NC" active="0"></FlashEn>
填写 ISP 的 PRILIGHTTRIG 使能的有效电平
rk3288：对应 GPIO7-B6
rk3368：对应 GPIO3-C5
rk3399：对应 GPIO1-A4
<FlashLuminance luminance="0"></FlashLuminance>
暂不支持
<FlashColorTemp colortemp="0"></FlashColorTemp>
暂不支持
<FlashModeType mode="1"></FlashModeType>
填写 Flash 的工作方式,目前支持以下两种 flash 工作模式：
Mode 1:
该模式下 prelight_trig 和 flash_trig 的时序图如下：



prelight_trig 为高，flash_trig 为低时进入 movie/torch mode；prelight_trig 为低，flash_trig 为高时进入 flash mode。

以 SGN3780 芯片为例：



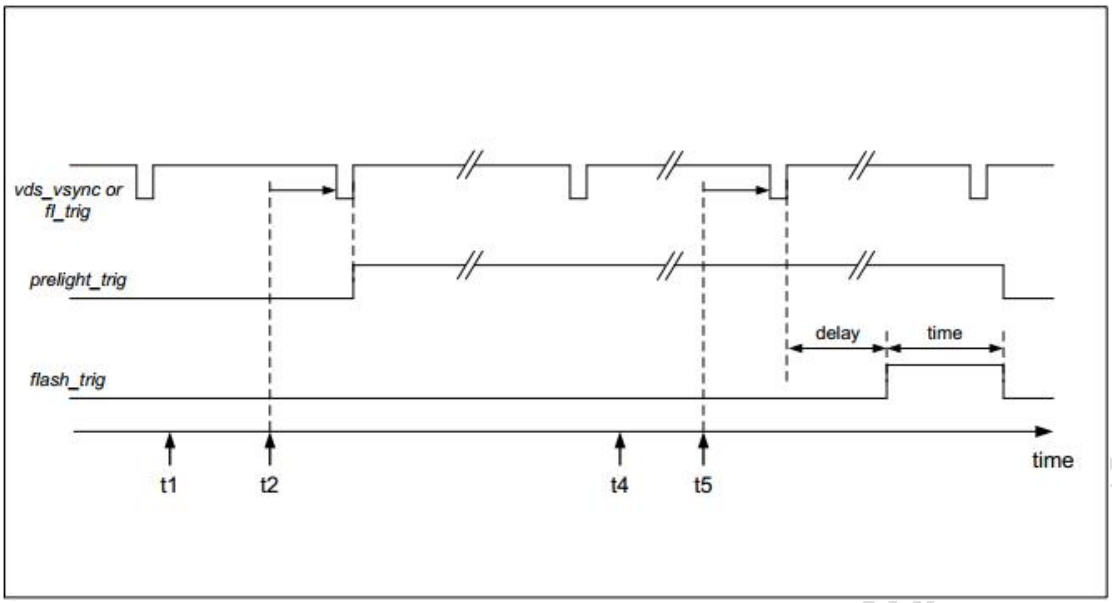
ENF <------> FlashTrigger <------> GPIO7-B5

ENM <------> FlashEn <------> GPIO7-B6

ENM 为低，ENF 为高时进入 flash 模式；ENM 为高，ENF 为低时进入 Movie/Torch 模式。

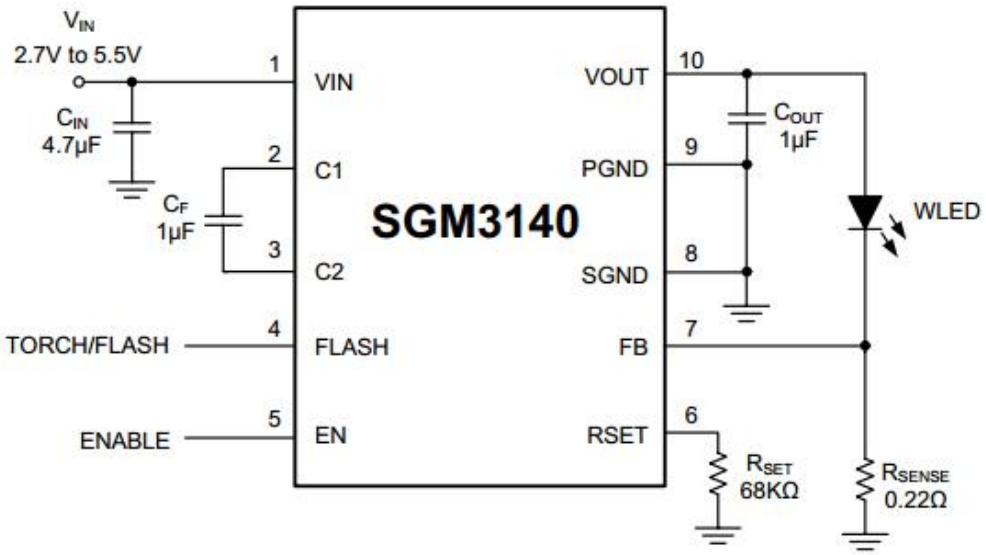
Mode 2:

该模式下 prelight_trig 和 flash_trig 的时序图如下：



prelight_trig 为高，flash_trig 为低进入 movie/torch mode；prelight_trig 为高，flash_trig 为高时进入 flash mode。

以 SGM3140 芯片为例：



FLASH <------> FlashTrigger<------> GPIO3-C4

EN <------> FlashEn <------> GPIO3-C5

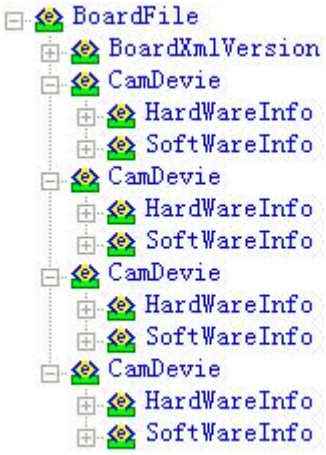
EN 为高，FLASH 为高时进入 flash 模式；EN 为高，FLASH 为低时进入 torch 模式。

注意：在 mode2 情况下，FlashTrigger 和 FlashEn 的有效电平须配置一致，否则会导致 panic 错误。

5. 5. cam_board.xml 支持多个 sensor 配置

Cam_board.xml 支持多个 sensor device 配置，在 xml 里添加自己可能用到的<CamDevie>，填写上面所述相应所需的硬件信息即可。

例如下图：



5.6. 如何测试 CTS_Verify FOV

麻烦参考 5.1 章节（Sensor 注册信息）中关于<SensorFovParameter>的说明

5.7. 如何解决开启 Camera 最初几帧的偏色问题

麻烦参考 5.1 章节（Sensor 注册信息）中关于<SensorAWB_Frame_Skip >的说明；

5.8. Camera 插值说明

麻烦参考 5.3 章节(软件功能配置信息)中关于<InterpolationRes>的说明。

6. SOC Sensor 支持列表

Camera Sensor	Type	Optical format	VCM	VCM driver	IR-cut filter	Dimension(mm)	Lens	Module Vendor and Module number
raw sensor 参见文件《**RKISPV1_Camera_Module_AVL**》								
MIPI soc SENSOR								
2Mega								
Ov2685								
GC2155								
DVP soc SENSOR								
5Mega								
OV5640								
HM5065								
2Mega								
GC2035								
HM2057								
NT99252								
SP2518								
OV2659								
0.3Mega								

GC0308								

7. Sensor 驱动移植指导

7.1 基本概念

7.1.1 MIPI

MIPI 的全称是 Mobile Industry Processor Interface(移动行业处理器接口)，本文描述的 MIPI 接口特指物理层使用 D-PHY 传输规范，协议层使用 CSI-2 的通信接口。

7.1.2 Lane

用于连接发送端和接收端的一对高速差分线，既可以是时钟 Lane，也可以是数据 Lane。

7.2 常用数据类型

IsiRegisterFlags_t

[说明]

寄存器配置结构体中的 Flag 枚举类型

[定义]

```
typedef enum IsiRegisterFlags_e
{
    // basic features
    eTableEnd   = 0x00,  /**< special flag for end of register table */
    eReadable   = 0x01,
    eWritable   = 0x02,
    eVolatile   = 0x04,  /**< register can change even if not written by I2C */
    eDelay      = 0x08,  /**< wait n ms */
    eReserved   = 0x10,
    eNoDefault  = 0x20,  /**< no default value specified */
    eTwoBytes   = 0x40,  /**< SMIA sensors use 8-, 16- and 32-bit registers */
    eFourBytes  = 0x80,  /**< SMIA sensors use 8-, 16- and 32-bit registers */

    // combined features
    eReadOnly           = eReadable,
```

```
    eWriteOnly          = eWritable,
    eReadWrite          = eReadable | eWritable,
    eReadWriteDel       = eReadable | eWritable | eDelay,
    eReadWriteVolatile  = eReadable | eWritable | eVolatile,
    eReadWriteNoDef     = eReadable | eWritable | eNoDefault,
    eReadWriteVolNoDef  = eReadable | eWritable | eVolatile | eNoDefault,
    eReadVolNoDef       = eReadable | eVolatile | eNoDefault,
    eReadOnlyVolNoDef   = eReadOnly | eVolatile | eNoDefault,

    // additional SMIA features
    eReadOnly_16        = eReadOnly         | eTwoBytes,
    eReadWrite_16       = eReadWrite        | eTwoBytes,
    eReadWriteDel_16    = eReadWriteDel     | eTwoBytes,
    eReadWriteVolatile_16  = eReadWriteVolatile | eTwoBytes,
    eReadWriteNoDef_16     = eReadWriteNoDef    | eTwoBytes,
    eReadWriteVolNoDef_16  = eReadWriteVolNoDef | eTwoBytes,
    eReadOnlyVolNoDef_16   = eReadOnly_16 | eVolatile | eNoDefault,
    eReadOnly_32        = eReadOnly         | eFourBytes,
    eReadWrite_32       = eReadWrite        | eFourBytes,
    eReadWriteVolatile_32  = eReadWriteVolatile | eFourBytes,
    eReadWriteNoDef_32     = eReadWriteNoDef    | eFourBytes,
    eReadWriteVolNoDef_32  = eReadWriteVolNoDef | eFourBytes
} IsiRegisterFlags_t;
```

[关键成员]

成员名称	描述
@eReadWrite	读写标志
@eTableEnd	寄存器数组结束的标志
@eDelay	延时标志，若寄存器序列需要延时可用这个标志来实现延时

[示例]

```
{0x06, 0x16,"0x0100",eReadWrite},
{0x00, 0x05,"0x0100",eDelay},//延时 5ms
{0x00, 0x00,"0x0100",eTableEnd}//数组结束
```

IsiRegDescription_t

[说明]

寄存器配置信息结构体

[定义]

```
typedef struct IsiRegisterFlags_s
{
    uint32_t    Addr;   /* register address */
    uint32_t    DefaultValue; /* register value */
    const char  * pName;
    uint32_t    Flags; /*see IsiRegisterFlags_t */
} IsiRegDescription_t;
```

[关键成员]

成员名称	描述
@Addr	寄存器地址，具体字节数根据实际写；
@DefaultValue	寄存器数组结束的标志
@pName	给寄存器起名称，不用太关心；
@Flags	寄存器类型的标志，用于标记当前寄存器是读写使能，还是用于延时，或是寄存器数组结束的标志等；

[示例]

```
{0x06, 0x16,"0x0100",eReadWrite},
```

IsiSensorHandle_t

[说明]

Sensor 驱动 handle 的定义

[定义]

typedef void *IsiSensorHandle_t;

[关键成员]

成员名称	描述
@IsiSensorHandle_t	空类型指针，赋值时匹配具体类型；

[示例]

```
IsiSensorHandle_t      handle;
```

IsiSensorCaps_t

[说明]

Sensor 配置信息结构体

[定义]

```
typedef struct IsiSensorCaps_s
{
    uint32_t BusWidth;                  /**< supported bus-width */
    uint32_t Mode;                      /**< supported operating modes */
    uint32_t FieldSelection;            /**< sample fields */
    uint32_t YCSequence;
    uint32_t Conv422;
    uint32_t BPat;                      /**< bayer pattern */
    uint32_t HPol;                      /**< horizontal polarity */
    uint32_t VPol;                      /**< vertical polarity */
    uint32_t Edge;                      /**< sample edge */
    uint32_t Bls;                       /*< black level substraction */
    uint32_t Gamma;                     /**< gamma */
    uint32_t CConv;
    uint32_t Resolution;                /**< supported resolutions */
    uint32_t DwnSz;
    uint32_t BLC;
    uint32_t AGC;
```

```
    uint32_t AWB;
    uint32_t AEC;
    uint32_t DPCC;
    uint32_t CieProfile;
    uint32_t SmiaMode;
    uint32_t MipiMode;
    uint32_t AfpsResolutions;          /**< resolutions supported by Afps */
  uint32_t SensorOutputMode;
  uint32_t Index;
} IsiSensorCaps_t;
```

[关键成员]

成员名称	描述
BusWidth	ISI_BUSWIDTH_8BIT_ZZ ISI_BUSWIDTH_8BIT_EX ISI_BUSWIDTH_10BIT_EX ISI_BUSWIDTH_10BIT_ZZ ISI_BUSWIDTH_12BIT ISI_BUSWIDTH_10BIT(ISI_BUSWIDTH_10BIT_EX) 总线宽度，用于配置 ISP 采集位宽，后缀 ZZ 表示低位补零，EX 表示高位补零
Mode	ISI_MODE_BT601 ISI_MODE_BT656 ISI_MODE_BAYER ISI_MODE_DATA ISI_MODE_PICT ISI_MODE_RGB565 ISI_MODE_MIPI ISI_MODE_BAY_BT656 ISI_MODE_RAW_BT656 配置 ISP 采集的数据格式，一般 MIPI 用的比较多；DVP 使用 ISI_MODE_PICT 或 ISI_MODE_BT601，具体自行查找格式定义；
FieldSelection	ISI_FIELDSEL_BOTH ISI_FIELDSEL_EVEN ISI_FIELDSEL_ODD 域选择，一般情况下使用 ISI_FIELDSEL_BOTH
YCSequence	ISI_YCSEQ_YCBYCR ISI_YCSEQ_YCRYCB ISI_YCSEQ_CBYCRY

	ISI_YCSEQ_CRYCBY YUV 的 UV 分量排列方式
Conv422	ISI_CONV422_COSITED ISI_CONV422_INTER ISI_CONV422_NOCOSITED YUV422 的三种排列方式
BayerPatttern	ISI_BPAT_RGRGGBGB ISI_BPAT_GRGRBGBG ISI_BPAT_GBGBRGRG ISI_BPAT_BGBGGRGR Bayer 格式，根据 Sensor 手册提供的格式配置，仅输出数据格式为 Bayer 时设置有效
HPolarity	ISI_HPOL_SYNCPOS ISI_HPOL_SYNCNEG ISI_HPOL_REFPOS ISI_HPOL_REFNEG 行同步信号极性
VPolarity	ISI_VPOL_POS ISI_VPOL_NEG 场同步信号极性
Edge	ISI_EDGE_RISING ISI_EDGE_FALLING 同步时钟极性
Bls	ISI_BLS_OFF only now 帧前添加黑行数
Gamma	ISI_GAMMA_OFF only now 伽马校验
ColorConv	ISI_CCONV_OFF only now 颜色校验
Resolution	Such as ISI_RES_2592_1944P30 所有已支持的分辨率可以在 hardware/rockchip/camera/SiliconImage/include/isi/isi_common.h 中查看。如果没有你想要的分辨率，请联系我们添加(自行在/isi_common.h 中添加是不够的)。
DwnSz	ISI_DWNSZ_SUBSMPL ISI_DWNSZ_SCAL_BAY ISI_DWNSZ_SCAL_COS 降低图像输出尺寸的方式，常用的是二分采样（ISI_DWNSZ_SUBSMPL）
BLC	ISI_BLC_OFF ISI_BLC_AUTO 黑电平校验
AGC	ISI_AGC_OFF 自动增益控制
AWB	ISI_AWB_OFF

	自动白平衡控制
AEC	ISI_AEC_OFF 自动曝光控制
DPCC	ISI_DPCC_OFF 坏点校正
AFPS	ISI_AFPS_NOTSUPP 自动帧率调整
Index	Default 0 Sensor 索引号

更多信息请查看 hardware/rockchip/camera/SiliconImage/include/isi/isi_common.h。

[示例]

```
static RESULT OV8858_IsiGetCapsIssInternal(
      IsiSensorCaps_t     *pIsiSensorCaps, uint32_t    mipi_lanes)
{
      RESULT result = RET_SUCCESS;

      if ( pIsiSensorCaps == NULL )
      {
            return ( RET_NULL_POINTER );
      }
      else
      {
            if (mipi_lanes == SUPPORT_MIPI_FOUR_LANE) {
                  ......
            } else if(mipi_lanes == SUPPORT_MIPI_TWO_LANE) {
                  ......
            }    else if(mipi_lanes == SUPPORT_MIPI_ONE_LANE) {
                  ......
            }
            pIsiSensorCaps->BusWidth            = ISI_BUSWIDTH_10BIT;
            pIsiSensorCaps->Mode                 = ISI_MODE_MIPI;
            pIsiSensorCaps->FieldSelection    = ISI_FIELDSEL_BOTH;
            pIsiSensorCaps->YCSequence         = ISI_YCSEQ_YCBYCR;
            pIsiSensorCaps->Conv422             = ISI_CONV422_NOCOSITED;
            pIsiSensorCaps->BPat                = ISI_BPAT_BGBGGRGR;
            pIsiSensorCaps->HPol                = ISI_HPOL_REFPOS;
            pIsiSensorCaps->VPol                = ISI_VPOL_POS;
            pIsiSensorCaps->Edge                = ISI_EDGE_RISING;
            pIsiSensorCaps->Bls                 = ISI_BLS_OFF;
            pIsiSensorCaps->Gamma                = ISI_GAMMA_OFF;
            pIsiSensorCaps->CConv               = ISI_CCONV_OFF;
            pIsiSensorCaps->BLC                 = ( ISI_BLC_AUTO | ISI_BLC_OFF);
            pIsiSensorCaps->AGC                = ( ISI_AGC_OFF );
            pIsiSensorCaps->AWB                = ( ISI_AWB_OFF );
            pIsiSensorCaps->AEC                = ( ISI_AEC_OFF );
```

```
            pIsiSensorCaps->DPCC            = ( ISI_DPCC_AUTO | ISI_DPCC_OFF );
            pIsiSensorCaps->DwnSz               = ISI_DWNSZ_SUBSMPL;
            pIsiSensorCaps->CieProfile      = ( ISI_CIEPROF_A
                                                | ISI_CIEPROF_D50
                                                | ISI_CIEPROF_D65
                                                | ISI_CIEPROF_D75
                                                | ISI_CIEPROF_F2
                                                | ISI_CIEPROF_F11 );
            pIsiSensorCaps->SmiaMode            = ISI_SMIA_OFF;
            pIsiSensorCaps->MipiMode            = ISI_MIPI_MODE_RAW_10;
            pIsiSensorCaps->AfpsResolutions = ( ISI_AFPS_NOTSUPP );
  pIsiSensorCaps->SensorOutputMode = ISI_SENSOR_OUTPUT_MODE_RAW;
        }
  end:
        return result;
  }
```

IsiAfpsInfo_t

[说明]

Sensor 的 AFPS 配置信息结构体，用于调整帧率时的参数配置，因曝光调节的最大值为帧输出时间（即场信号有效时），当曝光调到最大时，图像的亮度效果未达到合适的值时，会通过调整帧率，从而改变曝光时间范围，使输出的图像亮度达到较合适的值；

[定义]

```
typedef struct IsiAfpsInfo_s
{
    float   AecMinGain;   /*< minimum gain for AEC in Afps mode */
    float   AecMaxGain;   /*< maximum gain for AEC in Afps mode */
    float   AecMinIntTime; /*< minimum integration time for AEC in Afps mode */
    float   AecMaxIntTime; /*< maximum integration time for AEC in Afps mode */
    uint32_t   AecSlowestResolution; /*< slowst resolution for AEC in Afps mode */
    IsiAfpsResInfo_t Stage[ISI_NUM_AFPS_STAGES];      /*< the list of supported resolutions with .MaxIntTime in ascending(!) order;Resolution = 0 marks end of list if not all array elements are used */
    uint32_t   CurrResolution;    /*< current resolution */
    float   CurrMinIntTime;   /*< minimum integration time of current resolution */
```

```
    float   CurrMaxIntTime;   /*< maximum integration time of current resolution
*/
} IsiAfpsInfo_t;
```

其中 IsiAfpsResInfo_t 结构如下：

```
typedef struct IsiAfpsResInfo_s
{
    uint32_t Resolution;          /**< the corresponding resolution ID */
    float    MaxIntTime;          /**< the maximum supported integration time */
} IsiAfpsResInfo_t;
```

[关键成员]

成员名称	描述
@AecMinGain	最小增益
@AecMaxGain	最大增益
@AecMinIntTime	最小曝光时间
@AecMaxIntTime	最大曝光时间
@AecSlowestResolution	帧率最低的分辨率
@IsiAfpsResInfo_t	分辨率信息结构数组
@CurrResolution	当前分辨率
@CurrMinIntTime	当前分辨率下最小曝光时间
@CurrMaxIntTime	当前分辨率下最大曝光时间

[示例]

```
static RESULT OV8858_IsiGetAfpsInfoHelperIss(
    OV8858_Context_t     *pOV8858Ctx,
    uint32_t              Resolution,
    IsiAfpsInfo_t*        pAfpsInfo,
    uint32_t              AfpsStageIdx
)
{
    RESULT result = RET_SUCCESS;
    TRACE( OV8858_INFO, "%s: (enter)\n", __FUNCTION__);
    DCT_ASSERT(pOV8858Ctx != NULL);
    DCT_ASSERT(pAfpsInfo != NULL);
    DCT_ASSERT(AfpsStageIdx <= ISI_NUM_AFPS_STAGES);
    // update resolution in copy of config in context
```

```
        pOV8858Ctx->Config.Resolution = Resolution;
        // tell sensor about that
        result = OV8858_SetupOutputWindowInternal( pOV8858Ctx,
&pOV8858Ctx->Config,BOOL_FALSE,BOOL_FALSE );
        if ( result != RET_SUCCESS )
        {
            TRACE( OV8858_ERROR, "%s: SetupOutputWindow failed for resolution
ID %08x.\n", __FUNCTION__, Resolution);
            return ( result );
        }

        // update limits & stuff (reset current & old settings)
        result = OV8858_AecSetModeParameters( pOV8858Ctx, &pOV8858Ctx->Config );
        if ( result != RET_SUCCESS )
        {
            TRACE( OV8858_ERROR, "%s: AecSetModeParameters failed for resolution
ID %08x.\n", __FUNCTION__, Resolution);
            return ( result );
        }
        // take over params
        pAfpsInfo->Stage[AfpsStageIdx].Resolution = Resolution;
        pAfpsInfo->Stage[AfpsStageIdx].MaxIntTime =
pOV8858Ctx->AecMaxIntegrationTime;
        pAfpsInfo->AecMinGain          = pOV8858Ctx->AecMinGain;
        pAfpsInfo->AecMaxGain          = pOV8858Ctx->AecMaxGain;
        pAfpsInfo->AecMinIntTime       = pOV8858Ctx->AecMinIntegrationTime;
        pAfpsInfo->AecMaxIntTime       = pOV8858Ctx->AecMaxIntegrationTime;
        pAfpsInfo->AecSlowestResolution = Resolution;
        TRACE( OV8858_INFO, "%s: (exit)\n", __FUNCTION__);
        return ( result );
    }
```

7.3 API 参考

OV8858_IsiCreateSensorIss

[描述]

创建一个 Sensor 实例，包含初始化 hSensor，Sensor 的 I2C 配置信息、一些 Sensor 状态参数、VCM 配置等；

[语法]

Static RESULT OV8858_IsiCreateSensorIss(IsiSensorInstanceConfig_t *pConfig)

[参数]

参数名称	描述	输入输出
IsiSensorInstanceConfig	IsiSensorInstanceConfig_t 结构体指针	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针
RET_OUTOFMEM	失败，内存溢出

OV8858_IsiReleaseSensorIss

[描述]

释放 Sensor 实例，包含停止数据流、掉电、释放 Sensor handle；

[语法]

static RESULT OV8858_IsiReleaseSensorIss(IsiSensorHandle_t handle)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle_t 结构体句柄	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误句柄

OV8858_IsiGetCapsIssInternal

[描述]

为 Sensor 描述结构填写正确的指针，包含 Resolution、BusWidth、Mode 等，具体可以参考 7.2.4 IsiSensorCaps_t 结构中的介绍；

[语法]

static RESULT OV8858_IsiGetCapsIssInternal(IsiSensorCaps_t *pIsiSensorCaps, uint32_t mipi_lanes)

[参数]

参数名称	描述	输入输出
pIsiSensorCaps	IsiSensorCaps_t 结构体指针	输出
mipi_lanes	MIPI 的通道数	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针

OV8858_SetupOutputFormat

[描述]

根据配置设置图像输出格式，目前没有做什么处理，比如 pConfig->BusWidth 一般是根据外部输入配置为固定值，不能动态调整，但是 pConfig->BusWidth 的 switch 语句中 case 没有 ISI_BUSWIDTH_8BIT_ZZ，而外部输入又是这个格式，则需要补上，否则 CameraHal 程序有可能报错；比如 BLC、AGC 这类设置一般都处于 OFF 状态，所以移植的时候不需要做多少改动，除非有需要实现对应功能；

[语法]

RESULT OV8858_SetupOutputFormat(OV8858_Context_t *pOV8858Ctx, const IsiSensorConfig_t *pConfig)

[参数]

参数名称	描述	输入输出
pOV8858Ctx	OV8858_Context_t 结构体指针	输入
pConfig	IsiSensorConfig_t 的结构体指针	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针

OV8858_get_PCLK

[描述]

获取 Sensor 的 PLL 输出像素时钟，一般通过获取 Sensor 的寄存器值，根据公式去换算；若应用不需要更改 PLL 输出，也可直接返回定值，减少计算步骤；

[语法]

int OV8858_get_PCLK(OV8858_Context_t *pOV8858Ctx,int XVCLK)

[参数]

参数名称	描述	输入输出
pOV8858Ctx	OV8858_Context_t 结构体指针	输入
XVCLK	外部输入时钟	输入

[返回值]

返回值	描述
PCLK	返回像素时钟值

OV8858_SetupOutputWindowInternal

[描述]

根据 CameraHal 传进来的分辨率去配置 Sensor 的寄存器，移植的时候根据 Sensor 支持的分辨率去修改；目前 8M 及以下的 Sensor 可只实现全分辨率，减少工作量，需要更低的分辨率可以通过 ISP 裁剪；8M 以上无法实现全分辨率 30fps 预览，需要实现 binning；

[语法]

RESULT OV8858_SetupOutputWindowInternal(OV8858_Context_t *pOV8858Ctx,const IsiSensorConfig_t *pConfig,bool_t set2Sensor, bool_t res_no_chg)

[参数]

参数名称	描述	输入输出
pOV8858Ctx	OV8858_Context_t 结构体指针	输入
pConfig	IsiSensorConfig_t 的结构体指针	输入
set2Sensor	true 表示设置到 Sensor，false 不做操作	输入

res_no_chg	false 表示更改分辨率，true 不做操作	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针
RET_NOTSUPP	失败，不支持的设置

OV8858_SetupImageControl

[描述]

设置传感器的 BLC, AGC, AWB, AEC, DPCC 等功能，一般没有用上，都是 OFF 状态，移植不需要修改；若需要用，请自行修改程序，并且跟前面的 IsiSensorCaps_t 结构中配置的参数对应；

[语法]

RESULT OV8858_SetupImageControl(OV8858_Context_t *pOV8858Ctx, const IsiSensorConfig_t *pConfig)

[参数]

参数名称	描述	输入输出
pOV8858Ctx	OV8858_Context_t 结构体指针	输入
pConfig	IsiSensorConfig_t 的结构体指针	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NOTSUPP	失败，不支持的设置

OV8858_IsiSetupSensorIss

[描述]

根据获取的配置信息去配置 Sensor，基本上是一些宏及外部传进来的参数，移植的时候程序不需要怎么修改，具体看程序的实现；

[语法]

RESULT OV8858_IsiSetupSensorIss(IsiSensorHandle_t handle,const IsiSensorConfig_t *pConfig)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
pConfig	IsiSensorConfig_t 的结构体指针	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针

OV8858_IsiChangeSensorResolutionIss

[描述]

更改 Sensor 输出分辨率，调用 OV8858_SetupOutputWindowInternal 用于设置分辨率，OV8858_AecSetModeParameters 用于重新设置曝光取值范围，OV8858_IsiExposureControlIss 用于重新设置曝光、增益值；移植时，此函数不需要修改；

[语法]

RESULT OV8858_IsiChangeSensorResolutionIss(IsiSensorHandle_t handle, uint32_t Resolution,uint8_t *pNumberOfFramesToSkip)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
Resolution	需要设置的分辨率	输入
pNumberOfFramesToSkip	需要丢掉的帧数	输出

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_NULL_POINTER	失败，空指针

OV8858_IsiSensorSetStreamingIss

[描述]

开启或关闭数据流，直接通过寄存器写入；移植时不需要修改函数实现，修改对应寄存器 addr 及 value 的宏定义；

[语法]

static RESULT OV8858_IsiSensorSetStreamingIss(IsiSensorHandle_t handle, bool_t on)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
on	true 表示开启数据流	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_WRONG_STATE	失败，错误的状态

OV8858_IsiSensorSetPowerIss

[描述]

Sensor 上电或掉电，根据 Sensor 手册的上电时序编写；很少需要修改；

[语法]

static RESULT OV8858_IsiSensorSetPowerIss(IsiSensorHandle_t handle, bool_t on)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
on	True 表示上电	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄

OV8858_IsiCheckSensorConnectionIss

[描述]

检测 Sensor 的连接状态，通过 i2c 获取 Sensor 的 Chip ID，并与 Sensor 驱动中的宏定义比较，匹配则返回 RET_SUCCESS；移植的时候只需修改宏定义；另外 Chip ID 有分 3 个字节和 2 个字节，若原来的程序是 3 个字节，而 Sensor 实际只有 2 个字节的 Chip ID，需进行适当修改；

[语法]

static RESULT OV8858_IsiCheckSensorConnectionIss(IsiSensorHandle_t handle)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_FAILURE	失败
RET_NULL_POINTER	失败，空指针

OV8858_IsiGetGainIss

[描述]

获取增益值;每次设置 gain 的时候都会将 gain 值缓存在 Sensor handle 的变量中，若是返回这个变量的值，程序不需要修改；若是通过 i2c 向 Sensor 读取，则根据不同的 Sensor 换算公式，需要进行适当的修改；

[语法]

RESULT OV8858_IsiGetGainIss(IsiSensorHandle_t handle,float *pSetGain)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
pSetGain	获取的增益倍数	输出

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

OV8858_IsiSetGainIss

[描述]

设置增益倍数，根据 Sensor 手册提供的换算方式修改程序，以倍数为单位；

[语法]

RESULT OV8858_IsiSetGainIss(IsiSensorHandle_t handle,float NewGain, float *pSetGain)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
NewGain	要设置的增益倍数	输入
pSetGain	实际设置到 Sensor 的增益倍数	输出

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

OV8858_IsiGetIntegrationTimeIss

[描述]

获取曝光时间，与获取增益值的方式同，若是从 Sensor handle 缓存值获取，不需要修改程序；

[语法]

RESULT OV8858_IsiGetIntegrationTimeIss(IsiSensorHandle_t handle,float *pSetIntegrationTime)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
pSetIntegrationTime	获取到的曝光时间，浮点型，秒为单位	输出

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

OV8858_IsiSetIntegrationTimeIss

[描述]

设置曝光时间，需要将曝光时间由秒为单位转换成行时间单位（line length）；并根据所占的寄存器位将值写入寄存器；根据 Sensor 手册进行修改；

[语法]

RESULT OV8858_IsiSetIntegrationTimeIss(IsiSensorHandle_t handle, float NewIntegrationTime, float *pSetIntegrationTime, uint8_t *pNumberOfFramesToSkip)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle 结构体句柄	输入
NewIntegrationTime	要设置的曝光时间，浮点型，秒为单位	输入
pSetIntegrationTime	实际设置的曝光时间，浮点型，秒为单位	输出
pNumberOfFramesToSkip	需要丢掉的帧数	输出

[返回值]

返回值	描述

RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

OV8858_IsiGetAfpsInfoHelperIss

[描述]

返回给定分辨率的可能的 AFPS 分辨率信息，也就是返回分辨率、最小曝光时间、最大曝光时间、最小增益倍数、最大增益倍数等信息，通过 pAfpsInfo 返回；每次调用只能获取一个分辨率的信息，通过 AfpsStageIdx 索引来分别存储信息；

[语法]

static RESULT OV8858_IsiGetAfpsInfoHelperIss(OV8858_Context_t *pOV8858Ctx, uint32_t Resolution, IsiAfpsInfo_t* pAfpsInfo, uint32_t AfpsStageIdx)

[参数]

参数名称	描述	输入输出
pOV8858Ctx	OV8858_Context_t 结构体指针	输入
Resolution	需要获取详细信息的分辨率	输入
pAfpsInfo	IsiAfpsInfo_t 结构体指针，用于返回信息	输出
AfpsStageIdx	分辨率列表的索引	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针
RET_NOTSUPP	失败，不支持的设置

OV8858_IsiGetAfpsInfoIss

[描述]

对 OV8858_IsiGetAfpsInfoHelperIss 多做了层封装；假设是 MIPI 接口的会根据 lane

数来区分不同 lane 下的不同分辨率；

[语法]

RESULT OV8858_IsiGetAfpsInfoIss(IsiSensorHandle_t handle,uint32_t Resolution,IsiAfpsInfo_t* pAfpsInfo)

[参数]

参数名称	描述	输入输出
handle	IsiSensorHandle_t 结构体句柄	输入
Resolution	需要获取详细信息的分辨率	输入
pAfpsInfo	IsiAfpsInfo_t 结构体指针，用于返回信息	输出

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

OV8858_IsiGetSensorI2cInfo

[描述]

用于上层代码获取 Sensor i2c 信息，可能有些地方会用到，所以一般根据寄存器字节数进行适当的修改，防止出错；

[语法]

static RESULT OV8858_IsiGetSensorI2cInfo(sensor_i2c_info_t** pdata)

[参数]

参数名称	描述	输入输出
pdata	sensor_i2c_info_t 结构体的双重指针	输入

[返回值]

返回值	描述
RET_SUCCESS	成功
RET_WRONG_HANDLE	失败，错误的句柄
RET_NULL_POINTER	失败，空指针

注：

1、Sensor 驱动还有些较为简单的接口或暂时没用上的接口，不做介绍，自行阅读；

2、Sensor 驱动中的 OTP 接口请参考现有程序及厂家资料修改；驱动移植时优先点亮 Sensor，再根据模组是否有烧录 OTP，决定是否实现或修改 OTP 程序；

3、Sensor 驱动的 VCM 程序请根据模组资料实现或修改，同样优先实现 Sensor 点亮，再根据有无 VCM 去实现或修改对应程序；VCM 接口如下：

```
/* AF functions */
pIsiSensor->pIsiMdiInitMotoDriveMds          = OV8858_IsiMdiInitMotoDriveMds;
pIsiSensor->pIsiMdiSetupMotoDrive            = OV8858_IsiMdiSetupMotoDrive;
pIsiSensor->pIsiMdiFocusSet                  = OV8858_IsiMdiFocusSet;
pIsiSensor->pIsiMdiFocusGet                  = OV8858_IsiMdiFocusGet;
pIsiSensor->pIsiMdiFocusCalibrate            = OV8858_IsiMdiFocusCalibrate;
```

7.4 移植步骤

驱动目录结构

以 OV8858 的驱动为例：

```
hardware\rockchip\camera\SiliconImage\isi\drv\OV8858
                                                |
                                                |--calib
                                                        |--OV8858_lens_LG-9569A2.xml
                                                |--include_priv
                                                        |--OV8858_MIPI_priv.h
                                                |--source
                                                        |--OV8858_MIPI.c
                                                        |--OV8858_tables.c
                                                |--Android.mk
```

准备工作

开始移植驱动之前，你需要拿到以下资料：

1.摄像头模组规格书。

2.VCM driver-IC datasheet(如果摄像头模组带 VCM)。

3.摄像头 sensor datasheet 和 application note(例如,OV 一般会提供)。

4.所需要的分辨率的寄存器配置表。

开始移植

你可以从零开始，新建文件、添加函数…等，但是我建议最好是以 SDK 中已有的驱动为模板进行移植。例如，如果你当前你要驱动的摄像头是 DVP 接口的，那么可以参考 OV2659/GC2155 等；如果是 MIPI RAW 的，可以参考 OV5648/OV8858/IMX214 等；如果是 MIPI YUV 的，可以参考 OV2685。

下面以 OV8858 为例：

首先从 OV8858 目录拷贝一份，重命名成你要驱动的 sensor 名字，目录内的各个文件名、源码中引用的 sensor 名都要进行修改，包括 Android.mk 中引用的文件名以及生成库的名字。

代码中涉及到的宏：

名称	说明
OV8858_MODE_SELECT	Stream(enable)控制寄存器、使能寄存器
OV8858_MODE_SELECT_OFF	Stream off 的寄存器值
OV8858_MODE_SELECT_ON	Stream on 的寄存器值
OV8858_SOFTWARE_RST	Software reset 寄存器
OV8858_SOFTWARE_RST_VALUE	Software reset 使能的寄存器值
OV8858_CHIP_ID_HIGH_BYTE	Chip id(或 Model id)的 high-byte 寄存器(如果有)
OV8858_CHIP_ID_HIGH_BYTE_DEFAULT	默认的 high-byte 的寄存器值(用以跟实际读出的值进行校对)
OV8858_CHIP_ID_MIDDLE_BYTE	Chip id(或 Model id)的 middle-byte 寄存器(如果有)
OV8858_CHIP_ID_MIDDLE_BYTE_DEFAULT	默认的 middle-byte 寄存器值
OV8858_CHIP_ID_LOW_BYTE	CHIP ID 的 low-byte 寄存器
OV8858_CHIP_ID_LOW_BYTE_DEFAULT	默认的 low-byte 寄存器值
OV8858_AEC_AGC_ADJ_H	Analog gain 寄存器的 high-byte
OV8858_AEC_AGC_ADJ_L	Analog gain 寄存器的 low-byte
OV8858_AEC_EXPO_H	Integration time 寄存器的 high-byte
OV8858_AEC_EXPO_M	Integration time 寄存器的 middle-byte
OV8858_AEC_EXPO_L	Integration time 寄存器的 low-byte
OV8858_SLAVE_ADDR	IIC address
OV8858_SLAVE_ADDR2	IIC addres(同一款 sensor，模组硬件接法不同，会有不同的 address，作为备选)
OV8858_SLAVE_AF_ADDR	VCM driver IC 的 slave address
Sensor_OTP_SLAVE_ADDR	读取 OTP 信息的 slave address
OV8858_MAXN_GAIN	增益寄存器最大写入值
OV8858_MIN_GAIN_STEP	最小增益增量
OV8858_MAX_GAIN_AEC	最大增益倍数
MAX_VCMDRV_CURRENT	VCM 最大电流
MAX_VCMDRV_REG	VCM 寄存器最大写入值
OV8858_I2C_NR_ADR_BYTES	寄存器地址的字节数
OV8858_I2C_NR_DAT_BYTES	寄存器值的字节数

以上宏在代码中的赋值，需要阅读相关的数据手册进行修改。
注意：如果寄存器没有分 hight-byte、middle-byte、low-byte 的话，那么只使用 low-byte 即可，当然，你也完全可以根据自己的喜好进行修改。

寄存器配置

Sensor 的寄存器配置序列需要从 sensor 的 datasheet 或者由原厂提供的寄存器配置文件中整理后应用在代码中。

根据应用场景及 sensor 的支持情况，寄存器序列可分为 1lane，2lane，4lane 三组，每组有一个 global setting 或者叫 initial setting，然后还有 binning size 和 full size 的 setting（OV 的 sensor 一般是这样）,以 ov8858 2lane 为例：

Global setting：

```
const IsiRegDescription_t OV8858_g_aRegDescription_twolane[] =
{
    {0x0100, 0x00,"0x0100",eReadWrite},
    {0x0100, 0x00,"0x0100",eReadWrite},
    {0x0100, 0x00,"0x0100",eReadWrite},
    {0x0100, 0x00,"0x0100",eReadWrite},
    {0x0302, 0x1e,"0x0100",eReadWrite},
    {0x0303, 0x00,"0x0100",eReadWrite},
    {0x0304, 0x03,"0x0100",eReadWrite},
    {0x030e, 0x00,"0x0100",eReadWrite},
    {0x030f, 0x09,"0x0100",eReadWrite},
    {0x0312, 0x01,"0x0100",eReadWrite},
    ...
    ...
    ...
    {0x0000 ,0x00,"eTableEnd",eTableEnd}
};
```

Bining size setting:

```
const IsiRegDescription_t OV8858_g_1632x1224_twolane[] =
{
    {0x030e, 0x00,"0x0100",eReadWrite}, // pll2_rdiv
    {0x030f, 0x09,"0x0100",eReadWrite}, // pll2_divsp
    {0x0312, 0x01,"0x0100",eReadWrite}, // pll2_pre_div0, pll2_r_divdac
    {0x3015, 0x01,"0x0100",eReadWrite}, //
    {0x3501, 0x4d,"0x0100",eReadWrite}, // exposure M
    {0x3502, 0x40,"0x0100",eReadWrite}, // exposure L
    {0x3706, 0x35,"0x0100",eReadWrite}, //
    {0x370a, 0x00,"0x0100",eReadWrite}, //
    {0x370b, 0xb5,"0x0100",eReadWrite}, //
    {0x3778, 0x1b,"0x0100",eReadWrite}, //
    {0x3808, 0x06,"0x0100",eReadWrite}, // x output size H
    {0x3809, 0x60,"0x0100",eReadWrite}, // x output size L
    {0x380a, 0x04,"0x0100",eReadWrite}, // y output size H
    {0x380b, 0xc8,"0x0100",eReadWrite}, // y output size L
    {0x380c, 0x07,"0x0100",eReadWrite}, // HTS H
    {0x380d, 0x88,"0x0100",eReadWrite}, // HTS L
    {0x380e, 0x04,"0x0100",eReadWrite}, // VTS H
    {0x380f, 0xdc,"0x0100",eReadWrite}, // VTS L
    {0x3814, 0x03,"0x0100",eReadWrite}, // x odd inc
    {0x3821, 0x67,"0x0100",eReadWrite}, // mirror on, bin on
    {0x382a, 0x03,"0x0100",eReadWrite}, // y odd inc
    ...
    ...
    ...
    {0x0000 ,0x00,"eTableEnd",eTableEnd}
};
```

Full size setting:

```
const IsiRegDescription_t OV8858_g_3264x2448_twolane[] =
{
    {0x030e, 0x02,"0x0100",eReadWrite}, // pll2_rdiv
    {0x030f, 0x04,"0x0100",eReadWrite}, // pll2_divsp
    {0x0312, 0x03,"0x0100",eReadWrite}, // pll2_pre_div0, pll2_r_divdac
    {0x3015, 0x00,"0x0100",eReadWrite}, //
    {0x3501, 0x9a,"0x0100",eReadWrite}, //
    {0x3502, 0x20,"0x0100",eReadWrite}, //
    {0x3706, 0x6a,"0x0100",eReadWrite}, //
    {0x370a, 0x01,"0x0100",eReadWrite}, //
    {0x370b, 0x6a,"0x0100",eReadWrite}, //
    {0x3778, 0x32,"0x0100",eReadWrite}, //
    {0x3808, 0x0c,"0x0100",eReadWrite}, // x output size H
    {0x3809, 0xc0,"0x0100",eReadWrite}, // x output size L
    {0x380a, 0x09,"0x0100",eReadWrite}, // y output size H
    {0x380b, 0x90,"0x0100",eReadWrite}, // y output size L
    {0x380c, 0x07,"0x0100",eReadWrite}, // HTS H
    {0x380d, 0x94,"0x0100",eReadWrite}, // HTS L
    {0x380e, 0x09,"0x0100",eReadWrite}, // VTS H
    {0x380f, 0xaa,"0x0100",eReadWrite}, // VTS L
    {0x3814, 0x01,"0x0100",eReadWrite}, // x odd inc
    {0x3821, 0x46,"0x0100",eReadWrite}, // mirror on, bin off
    {0x382a, 0x01,"0x0100",eReadWrite}, // y odd inc
    ...
    ...
    ...
    {0x0000 ,0x00,"eTableEnd",eTableEnd}
};
```

Fpschg setting:

通过设定不同的 VTS 寄存器的值来调整帧率。

```
const IsiRegDescription_t OV8858_g_1632x1224P30_twolane_fpschg[] =
{
    {0x380e, 0x04,"0x0100",eReadWrite}, // VTS H
    {0x380f, 0xdc,"0x0100",eReadWrite}, // VTS L
    {0x0000 ,0x00,"eTableEnd",eTableEnd}

};

const IsiRegDescription_t OV8858_g_1632x1224P25_twolane_fpschg[] =
{
    {0x380e, 0x05,"0x0100",eReadWrite}, // VTS H
    {0x380f, 0xd4,"0x0100",eReadWrite}, // VTS L
    {0x0000 ,0x00,"eTableEnd",eTableEnd}
};


const IsiRegDescription_t OV8858_g_1632x1224P20_twolane_fpschg[] =
{
    {0x380e, 0x07,"0x0100",eReadWrite}, // VTS H
    {0x380f, 0x4a,"0x0100",eReadWrite}, // VTS L
    {0x0000 ,0x00,"eTableEnd",eTableEnd}

};
```

计算方法：例如，初始化序列帧率为 30fps，VTS 为 0x04dc 时，那么 25fps 时的 VTS 为 0x04dc*30/25=0x05d4。

注意：

1、数组要以{0x0000, 0x00, "eTableEnd",eTableEnd}为结束标志。

2、如果寄存器值是两个字节，那么 IsiRegDescription_t 结构体的 Flags 值应为 eReadWrite_16，如：

```
//XVCLK=24Mhz, SCLK=4x120Mhz, MIPI 640Mbps, DACCLK=240Mhz
{0x0103 ,0x10120 ,"0x0100",eReadWrite_16},// sc ctrl (software reset)
{0x3638 ,0x20102 ,"0x0100",eReadWrite_16},//
{0x0300 ,0x30230 ,"0x0100",eReadWrite_16},// PLL CTRL 0(pll1_pre_div)
```

3、特别要注意的是，由于主控时序的要求，任何一个寄存器 setting 数组里面都不要 stream on sensor 或者叫 wake up sensor，比如，一般 OV 的 sensor 的 stream 寄存器是 0x0100，那么寄存器 setting 数组里不要对 0x0100 寄存器置 1，驱动的

IsiSensorSetStreamingIss 函数中会去操作 stream 寄存器，其他厂商的 sensor 的 stream 寄存器请参阅其 datasheet。

4、如果序列中需要延时操作，可以使用 eDelay 标志，如：

```
    {0x3706, 0x6a,"0x0100",eReadWrite}, //
    {0x370a, 0x01,"0x0100",eReadWrite}, //
    {0x370b, 0x6a,"0x0100",eReadWrite}, //
    {0x0000, 0x05,"0x0100",eDelay}, //delay 5ms
    {0x3808, 0x0c,"0x0100",eReadWrite}, // x output size H
    {0x3809, 0xc0,"0x0100",eReadWrite}, // x output size L
    {0x380a, 0x09,"0x0100",eReadWrite}, // y output size H
    {0x380b, 0x90,"0x0100",eReadWrite}, // y output size L
```

5、关于结构体的更多信息参见《常用数据类型》章节中的相关说明。

6、有的 sensor 比如 sony 的，没有 global setting，这样的话将数组留空即可：

```
const IsiRegDescription_t OV8858_g_aRegDescription_twolane[] =
{
    {0x0000 ,0x00,"eTableEnd",eTableEnd}

};
```

此外，驱动代码中函数 OV8858_IsiRegReadIss 和 OV8858_IsiRegWriteIss 对其的使用要考虑修改。

7.5 编译及验证

以 RK3399 Android7.1 为例：

1、初始化环境

```
camera@ISP:~/camera/rk3399_android7.1$ source build/envsetup.sh
including device/asus/fugu/vendorsetup.sh
including device/generic/mini-emulator-arm64/vendorsetup.sh
including device/generic/mini-emulator-armv7-a-neon/vendorsetup.sh
including device/generic/mini-emulator-mips64/vendorsetup.sh
including device/generic/mini-emulator-mips/vendorsetup.sh
including device/generic/mini-emulator-x86_64/vendorsetup.sh
including device/generic/mini-emulator-x86/vendorsetup.sh
including device/google/dragon/vendorsetup.sh
including device/htc/flounder/vendorsetup.sh
including device/huawei/angler/vendorsetup.sh
including device/lge/bullhead/vendorsetup.sh
including device/linaro/hikey/vendorsetup.sh
including device/moto/shamu/vendorsetup.sh
including device/rockchip/rk3399/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

2、根据板子型号选择

```
camera@ISP:~/camera/rk3399_android7.1$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. aosp_arm-eng
     2. aosp_arm64-eng
     3. aosp_mips-eng
     4. aosp_mips64-eng
     5. aosp_x86-eng
     6. aosp_x86_64-eng
     7. full_fugu-userdebug
     8. aosp_fugu-userdebug
     9. mini_emulator_arm64-userdebug
     10. m_e_arm-userdebug
     11. m_e_mips64-eng
     12. m_e_mips-userdebug
     13. mini_emulator_x86_64-userdebug
     14. mini_emulator_x86-userdebug
     15. aosp_dragon-userdebug
     16. aosp_dragon-eng
     17. aosp_flounder-userdebug
     18. aosp_angler-userdebug
     19. aosp_bullhead-userdebug
     20. hikey-userdebug
     21. aosp_shamu-userdebug
     22. rk3399-userdebug
     23. rk3399_32-userdebug
     24. rk3399_box-userdebug
     25. rk3399_box-user
     26. rk3399_64-userdebug
     27. rk3399_64-user
     28. rk3399_64_vr-userdebug
     29. rk3399_64_vr-user
     30. rk3399_disvr-userdebug
     31. rk3399_disvr-user
     32. rk3399_mid-userdebug
     33. rk3399_mid-user
     34. rk3399_laptop-userdebug
     35. rk3399_laptop-user

which would you like? [aosp_arm-eng]
```

输入对应编号；

3、编译

```
camera@ISP:~/camera/rk3399_android7.1/hardware/rockchip/camera/SiliconImage/isi/drv/OV8858$ mm -B
```

到 Sensor 驱动目录下执行 mm -B 编译驱动；或在源码的根目录执行 mmm /驱动目录 -B

4、修改 cam_board.xml

从板子取出原有 cam_board.xml 来修改，在 cmd 命令行输入：

adb pull /etc/cam_board.xml

修改完重新打进开发板，在命令行输入：

adb root

adb remount

adb push cam_board.xml /etc/

5、将 IQ 文件 push 进开发板，cam_board.xml 的配置决定 IQ 文件的名称，以 OV8858 为例，若 SensorLens 为空，则 IQ 文件名称为 OV8858.xml，若 SensorLens 为 NC，则 IQ 文件名称为 OV8858_lens_NC.xml，若 SensorLens 为具体型号，则 OV8858_lens_型号.xml；在 cmd 命令行输入：

adb push /文件目录/OV8858.xml /etc/

6、编译生成的 Sensor 驱动输出到如下路径：

rk3399_android7.1\out\target\product\rk3399_mid\system\lib\hw
Rk3399 的挖掘机版本的目录是 rk3399_mid，其他版本根据实际选目录；
在 cmd 命令行执行 push 命令将 so 库打进开发板：

adb push /文件目录/libisp_isi_drv_OV8858.so /system/lib/hw/

7、在 cmd 命令行输入 adb shell 进入 adb；

```
C:\Users\rockchip>adb shell
rk3399_mid:/ $ sync
rk3399_mid:/ $ reboot
```

同步文件，然后重启，以便新 push 进去的文件生效；

8、若相机 app 打开相机报错，可在 adb 命令行下通过 logcat 命令打印日志，查看错误；
可通过 logcat | grep CameraHal 仅打印相机相关的 log；
通过 logcat -c 清空 log 缓存；
另外可设置 log 打印等级，如下：

```
rk3399_mid:/ $ getprop | grep cam
[camera2.portability.force_api]: [1]
[init.svc.cameraserver]: [running]
[sys.camera.callprocess]: [cameraserver]
[sys.cts_camera.status]: [false]
[sys_graphic.cam_back.iq]: [/etc/GC8034.xml]
[sys_graphic.cam_back.iq.ver]: [2018-05-10_lsl_GC8034_matic_v1.0.1]
[sys_graphic.cam_back.len]: []
[sys_graphic.cam_back.modulename]: []
[sys_graphic.cam_camboard.ver]: [0x0.0xf.0x0]
[sys_graphic.cam_drv_camsys.ver]: [0x0.0x0.0x1]
[sys_graphic.cam_hal.ver]: [0x1.0x51.0x0]
[sys_graphic.cam_isi.ver]: [0x0.0xd.0x0]
[sys_graphic.cam_libisp.ver]: [0x2.0x3.0x0]
[sys_graphic.cam_otp_awb]: [false]
[sys_graphic.cam_otp_awb_enable]: [false]
[sys_graphic.cam_otp_lsc]: [false]
[sys_graphic.cam_otp_lsc_enable]: [false]
[sys_graphic.cam_trace]: [0]
rk3399_mid:/ $ setprop sys_graphic.cam_trace 2
rk3399_mid:/ $ getprop | grep cam
[camera2.portability.force_api]: [1]
[init.svc.cameraserver]: [running]
[sys.camera.callprocess]: [cameraserver]
[sys.cts_camera.status]: [false]
[sys_graphic.cam_back.iq]: [/etc/GC8034.xml]
[sys_graphic.cam_back.iq.ver]: [2018-05-10_lsl_GC8034_matic_v1.0.1]
[sys_graphic.cam_back.len]: []
[sys_graphic.cam_back.modulename]: []
[sys_graphic.cam_camboard.ver]: [0x0.0xf.0x0]
[sys_graphic.cam_drv_camsys.ver]: [0x0.0x0.0x1]
[sys_graphic.cam_hal.ver]: [0x1.0x51.0x0]
[sys_graphic.cam_isi.ver]: [0x0.0xd.0x0]
[sys_graphic.cam_libisp.ver]: [0x2.0x3.0x0]
[sys_graphic.cam_otp_awb]: [false]
[sys_graphic.cam_otp_awb_enable]: [false]
[sys_graphic.cam_otp_lsc]: [false]
[sys_graphic.cam_otp_lsc_enable]: [false]
[sys_graphic.cam_trace]: [2]
```