Rockchip_Driver_Guide_VI_CN

文件标识：RK-YH-GX-602

发布版本：V1.1.3

日期：2022-09-09

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

商标声明

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文旨在描述RKISP（Rockchip Image Signal Processing）模块的作用，整体工作流程，及相关的API接口。主要给
驱动工程师调试Camera提供帮助。

读者对象

本文档（本指南）主要适用于以下工程师：

驱动开发工程师

系统集成软件开发工程师

适用平台及系统

芯片名称	软件系统	支持情况
RV1126/RV1109	Linux(Kernel-4.19 and kernel-5.10)	Y
RK3566/RK3568	Linux(Kernel-4.19 and kernel-5.10)	Y
RK3588	Linux(Kernel-5.10)	Y
RV1106	Linux(Kernel-5.10)	Y

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	蔡艺伟	2020-06-11	初始版本
v1.0.0	陈泽发	2020-10-30	新增focus、zoom、iris、ircut说明
v1.0.1	陈泽发	2021-01-04	修改格式错误
v1.0.2	蔡艺伟	2021-01-21	rv1109/rv1126内存优化指南
v1.0.3	黄江龙	2021-02-04	新增VICAP LVDS/DVP/MIPI等接口设备节点注册说明
v1.0.4	蔡艺伟	2021-04-08	新增芯片版本差异说明和rk356x多sensor dts注册说明
v1.0.5	陈泽发	2021-04-24	新增MS41908步进马达驱动说明 完善抓RAW/YUV命令说明
v1.0.6	陈泽发	2021-07-21	vicap节点描述回退到和驱动一致
v1.0.7	蔡艺伟	2021-08-03	rv1109/rv1126时延优化指南
v1.0.8	陈泽发	2021-8-24	增加FAQ: 预览闪烁、光源紫色溢出问题
v1.0.9	蔡艺伟	2021-10-21	增加驱动单独更新方法说明
v1.1.0	陈泽发	2021-10-29	增加isp/vicap raw存储格式说明 增加vicap 异常复位说明 增加三摄dts配置说明 更新CIS驱动参考表/VCM驱动参考表
v1.1.1	陈泽发	2021-12-24	增加RK3588的说明 增加多摄同步机制章节

版本号	作者	修改日期	修改说明
v1.1.2	蔡艺伟	2022-8-29	增加rv1106说明 更新proc节点信息和debug模式使用说明
v1.1.3	陈泽发/林新泉	2022-9-9	增加vicap/isp特殊采集模式章节 描述8目方案中的多通道raw数据拼接采集的处理流程

目录

芯片VI差异

SOC	VI IP	VI Interface	Bayer CIS max resolution	Feature
RV1109	ISP20 (ISP + ISPP): 1 VICAP Full: 1 VICAP Lite: 1	MIPI DPHY: 2 x 4Lanes 2.5Gbps/Lane LVDS: 2 x 4Lanes 1.0Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	3072x2048	Upto 3 frames HDR

SOC	VI IP	VI Interface	Bayer CIS max resolution	Feature
RK3566	ISP21 Lite: 1 VICAP Full: 1	MIPI DPHY: 2 x 2Lanes or 1 x 4Lanes 2.5Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	4096x2304	No HDR
RK3568	ISP21：1 VICAP Full: 1	MIPI DPHY: 2 x 2Lanes or 1 x 4Lanes 2.5Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	4096x2304	Upto 2 frames HDR
RV1126	ISP20 (ISP + ISPP): 1 VICAP Full: 1 VICAP Lite: 1	MIPI DPHY: 2 x 4Lanes 2.5Gbps/Lane LVDS: 2 x 4Lanes 1.0Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	4416x3312	Upto 3 frames HDR
RK3588	VICAP：1 ISP30：2 FEC：2	MIPI DPHY: 4 x 2Lanes or 2 x 4Lanes 2.5Gbps/Lane MIPI DCPHY： 2 x 4Lanes dphy or 2 x 3Lanes cphy 2.5Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	4672x3504单isp 8192x6144双isp合成	Upto 3 frames HDR
RK3588S	VICAP：1 ISP30：2 FEC：2	MIPI DPHY: 2 x 2Lanes or 1 x 4Lanes 2.5Gbps/Lane MIPI DCPHY： 2 x 4Lanes dphy or 2 x 3Lanes cphy 2.5Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	4672x3504单isp 8192x6144双isp合成	Upto 3 frames HDR
RV1106	VICAP：1 ISP32：1	MIPI DPHY: 2 x 2Lanes or 1 x 4Lanes 2.5Gbps/Lane DVP: BT601 / BT656 / BT1120 pclk: 150MHz	3072x1728	Upto 2 frames HDR

Camera 软件驱动目录说明

Linux Kernel-4.19

|-- arch/arm/boot/dts DTS配置文件

|-- drivers/phy/rockchip

  |-- phy-rockchip-mipi-rx.c mipi dphy驱动
  |-- phy-rockchip-csi2-dphy-common.h
  |-- phy-rockchip-csi2-dphy-hw.c
  |-- phy-rockchip-csi2-dphy.c

|-- drivers/media

|-- platform/rockchip/cif RKCIF驱动

|-- platform/rockchip/isp RKISP驱动

|-- dev 包含 probe、异步注册、clock、pipeline、 iommu及media/v4l2 framework

|-- capture 包含 mp/sp/rawwr的配置及 vb2，帧中断处理

|-- dmarx 包含 rawrd的配置及 vb2，帧中断处理

|-- isp_params 3A相关参数设置

|-- isp_stats 3A相关统计

|-- isp_mipi_luma mipi数据亮度统计

|-- regs 寄存器相关的读写操作

|-- rkisp isp subdev和entity注册

|-- csi csi subdev和mipi配置

|-- bridge bridge subdev，isp和ispp交互桥梁

|-- platform/rockchip/ispp rkispp驱动

|-- dev 包含 probe、异步注册、clock、pipeline、 iommu及media/v4l2 framework

|-- stream 包含 4路video输出的配置及 vb2，帧中断处理

|-- rkispp ispp subdev和entity注册

|-- params TNR/NR/SHP/FEC/ORB参数设置

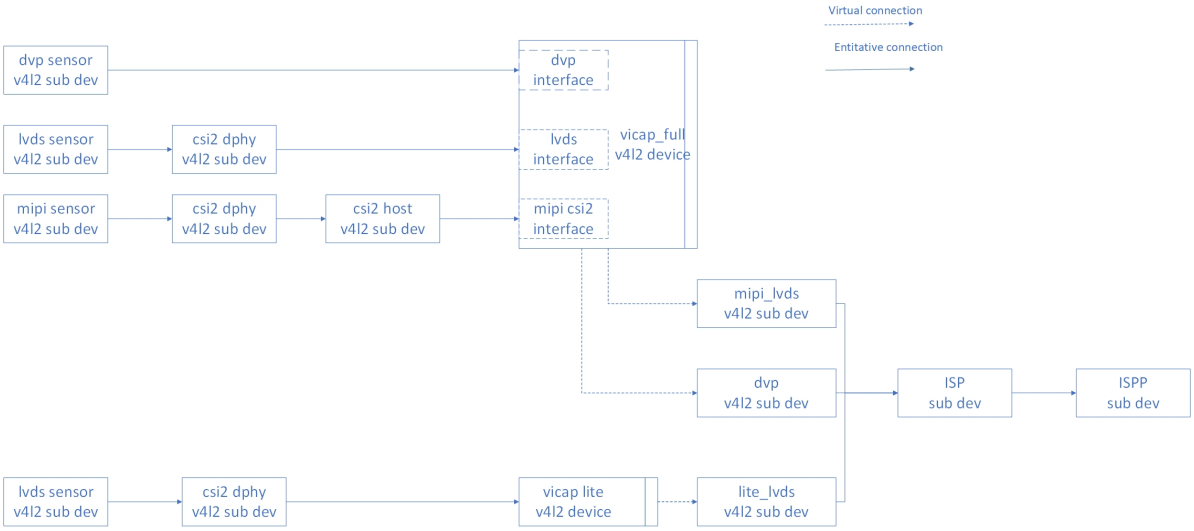
|-- stats ORB统计信息

|-- i2c

  |-- os04a10.c CIS(cmos image sensor)驱动

ISP和VICAP的链接关系

对于RV1126/RV1109及RK356X平台而言，VICAP和ISP是独立的两个图像处理IP, VICAP所采集图像若要通过ISP处理，在驱动层面需要生成VICAP对应接口的v4l2 sub device链接到ISP对应的节点，以提供参数给ISP驱动使用。ISP 驱动说明参见RKISP驱动，VICAP驱动说明参见RKVICAP驱动。具体的VICAP各接口与ISP链接的整体框图如下所示：

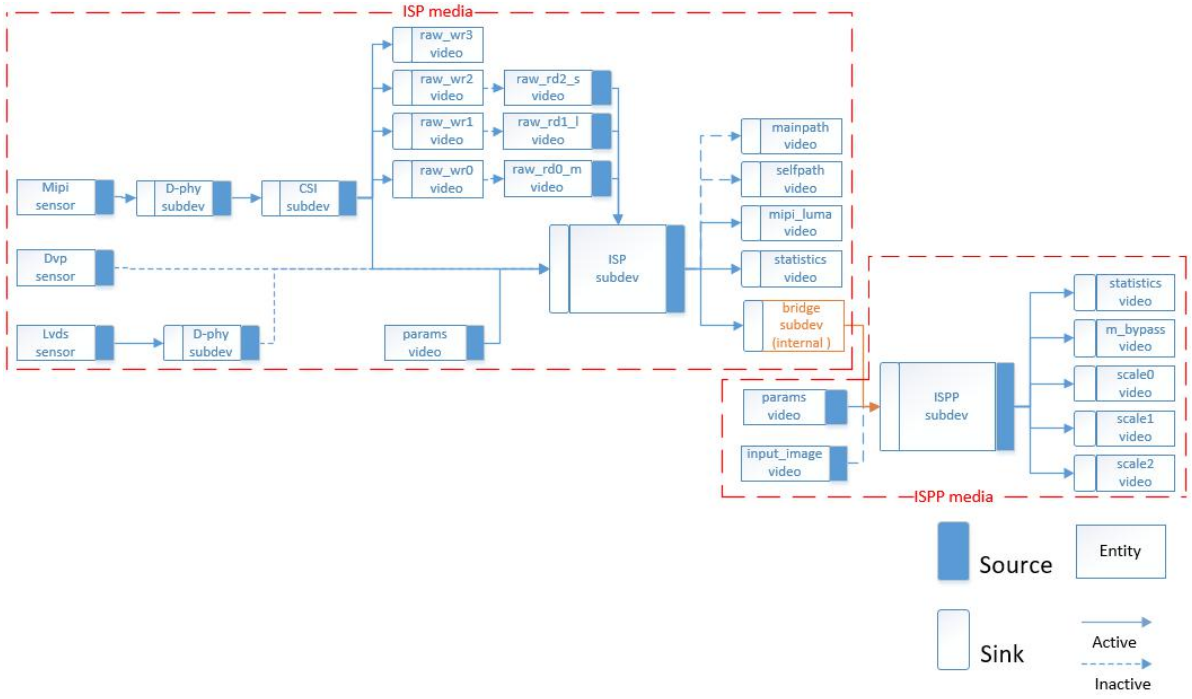


RKISP 驱动

框架简要说明

RKISP驱动主要是依据v4l2 / media framework实现硬件的配置、中断处理、控制 buffer轮转，以及控制subdevice(如 mipi dphy及sensor)的上下电等功能。

下面的框图描述了RKISP驱动的拓扑结构：



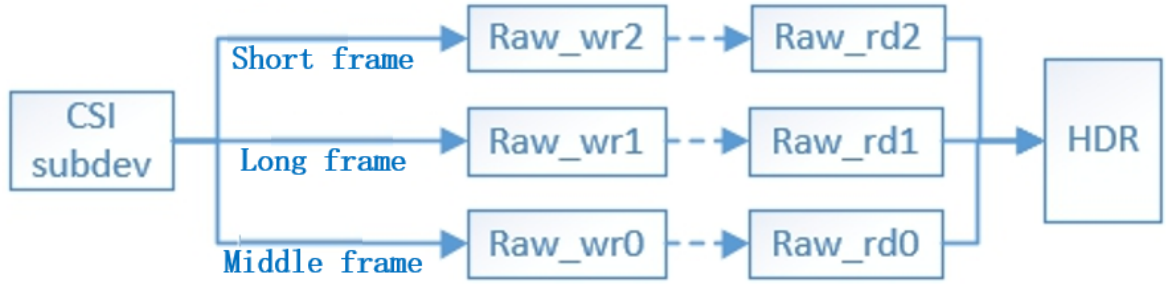
名称	类型	描述
rkisp_mainpath	v4l2_vdevcapture	Format: YUV, RAW Bayer; Support: Crop
rkisp_selfpath	v4l2_vdevcapture	Format: YUV, RGB; Support: Crop

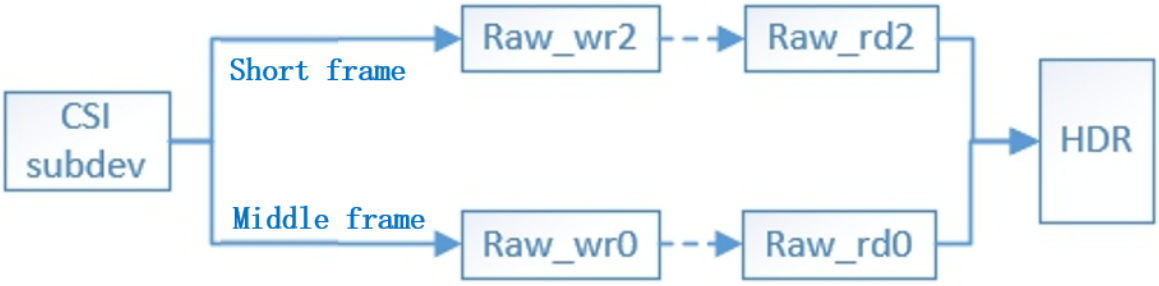
名称	类型	描述
rkisp-isp-subdev	v4l2_subdev	Internal isp blocks; Support: source/sink pad crop. The format on sink pad equal to sensor input format, the size equal to sensor input size. The format on source pad should be equal to vdev output format if output format is raw bayer, otherwise it should be YUYV2X8. The size should be equal/less than sink pad size.
rkisp-mipi-luma	v42_vdevcapture	Provice raw image luma
rkisp-statistics	v4l2_vdevcapture	Provide Image color Statistics information.
rkisp-input-params	v4l2_vdevoutput	Accept params for AWB, BLC..... Image enhancement blocks.
rkisp_rawrd0_m	v4l2_vdevoutput	Raw image read from ddr to isp,usually using for the hdr middle frame
rkisp_rawrd1_l	v4l2_vdevoutput	Raw image read from ddr to isp,usually using for the hdr long frame
rkisp_rawrd2_s	v4l2_vdevoutput	Raw image read from ddr to isp,usually using for the hdr short frame
rkisp-csi-subdev	v4l2_subdev	Mipi csi configure
rkisp_rawwr0	v4l2_vdevcapture	Raw image write to ddr from sensor,usually using for the hdr middle frame
rkisp_rawwr1	v4l2_vdevcapture	Raw image write to ddr from sensor,usually using for the hdr long frame
rkisp_rawwr2	v4l2_vdevcapture	Raw image write to ddr from sensor,usually using for the hdr short frame
rkisp_rawwr3	v4l2_vdevcapture	Raw image write to ddr from sensor
rockchip-mipi-dphy-rx	v4l2_subdev	MIPI-DPHY Configure.
rkisp-bridge-ispp	v4l2_subdev	Isp output yuv image to ispp
rkispp_input_image	v4l2_vdevoutput	Yuv image read from ddr to ispp
rkisp-isp-subdev	v4l2_subdev	The format and size on sink pad equal to isp outputThe support max size is 4416x3312, mix size is 66x258
rkispp_m_bypass	v4l2_vdev capture	Full resolution and yuv format

名称	类型	描述
rkispp_scale0	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[1 8] ratio 3264 max width for yuv422 2080 max width for yuv420
rkispp_scale1	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width
rkispp_scale2	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width

ISP HDR 模式说明

RKISP2支持接收mipi sensor输出hdr 3帧或2帧模式，硬件通过3路或2路dmatx采集数据到ddr，再通过3路或2路dmarx读到isp，isp做3帧或2帧合成，驱动链路如下所示：





csi subdev通过get_fmt获取sensor驱动多个pad格式的输出信息，对应csi的source pad。

Mipi sensor驱动具体配置请参考驱动移植步骤。

名称	名称	描述
rkisp-isp-subdev	Sensor pad0	Isp采集Sensor vc0(默认) 宽高格式输出，常用线性模式
rkisp_rawwr0	Sensor pad1	Rawwr0采集sensor vcX宽高格式输出
rkisp_rawwr1	Sensor pad2	Rawwr1采集sensor vcX宽高格式输出
rkisp_rawwr2	Sensor pad3	Rawwr2采集sensor vcX宽高格式输出
rkisp_rawwr3	Sensor pad4	Rawwr3采集sensor vcX宽高格式输出

RKVICAP 驱动

框架说明

RKVICAP驱动主要是基于v4l2 / media框架实现硬件的配置、中断处理、控制 buffer轮转，以及控制subdevice(如 mipi dphy及sensor)的上下电等功能。

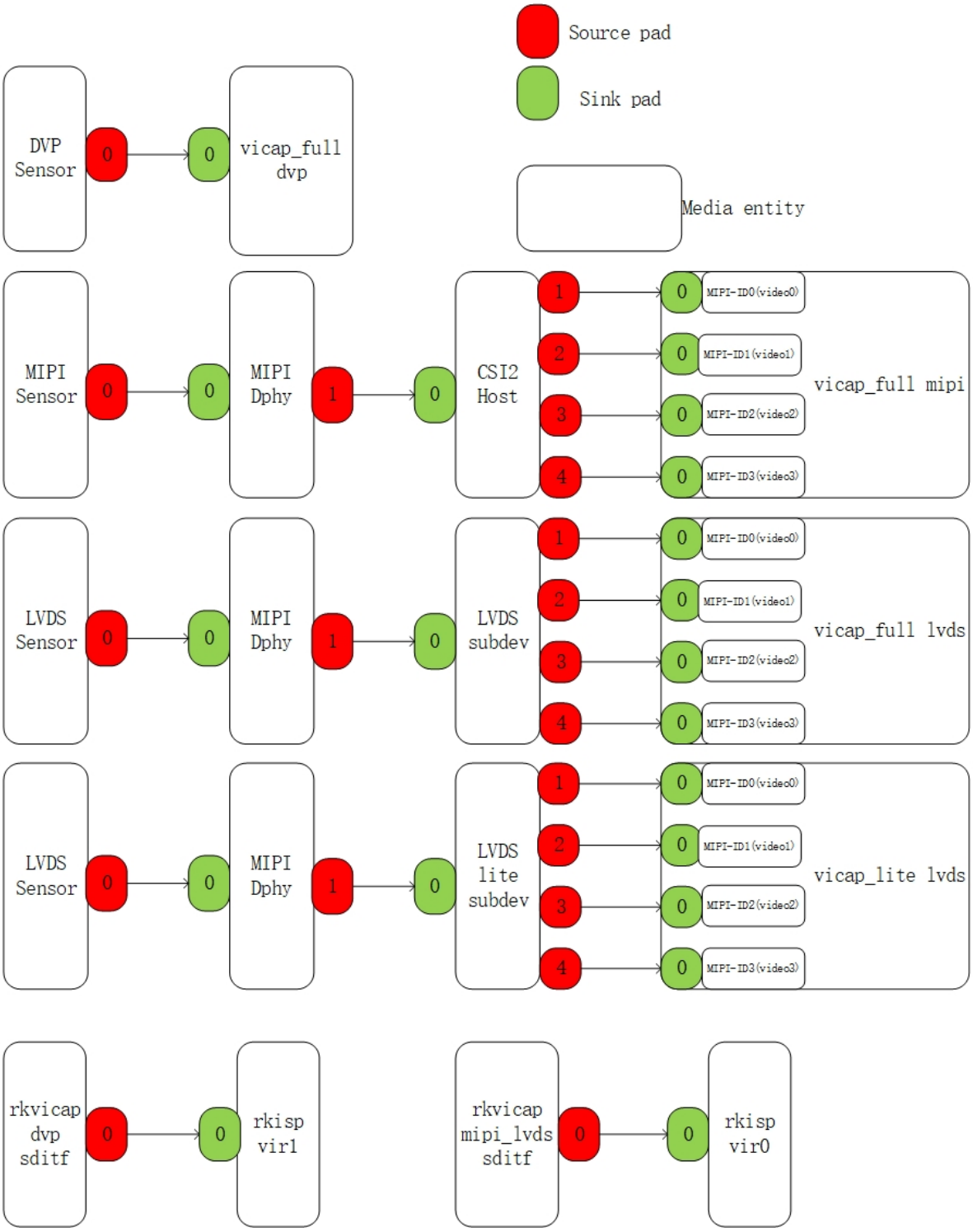
对于RV1126/RV1109而言，VICAP存在两个IP核，其中一个称之VICAP FULL,一个称之VICAP LITE。VICAP FULL拥有dvp/mipi/lvds三种接口，dvp可与mipi或者lvds接口同时工作，而mipi和lvds则不能同时工作，VICAP LITE 仅拥有lvds接口，可与VICAP FULL的接口同时工作；VICAP FULL dvp接口对应一个rkvicap_dvp节点，VICAP FULL mipi/lvds接口对应一个rkvicap_mipi_lvds节点，VICAP LITE 对应一个rkvicap_lite_mipi_lvds节点。各节点可独立采集。

对于RK356X芯片而言，VICAP只有单核，同时拥有dvp/mipi两种接口，dvp接口对应一个rkvicap_dvp节点，mipi接口对应一个rkvicap_mipi_lvds节点（与RV1126/RV1109的VICAP FULL同名），各节点可独立采集。

为了将VICAP采集数据信息同步给isp驱动，需要将VICAP驱动生成的逻辑sditf节点链接到isp所生成的虚拟节点设备。DVP接口对应rkvicap_dvp_sditf节点，VICAP FULL的mipi/lvds接口对应rkvicap_mipi_lvds_sditf节点，VICAP LITE对应rkvicap_lite_sditf。

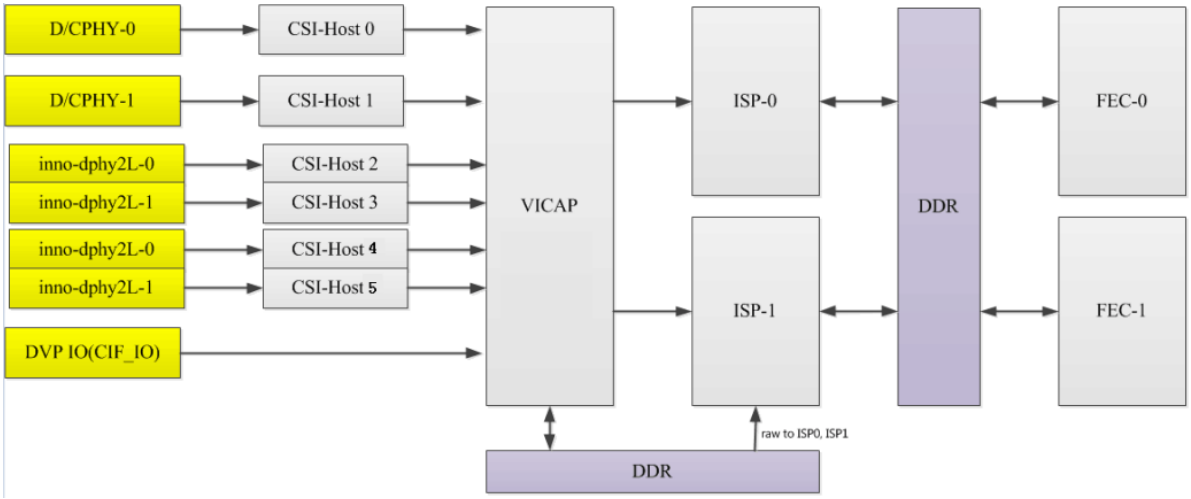
具体各接口的dts链接方式参见[CIS 设备注册(DTS)][#CIS 设备注册(DTS)]

下图描述了RKVICAP驱动的设备拓扑结构：



RK3588 VI结构

硬件通路框图



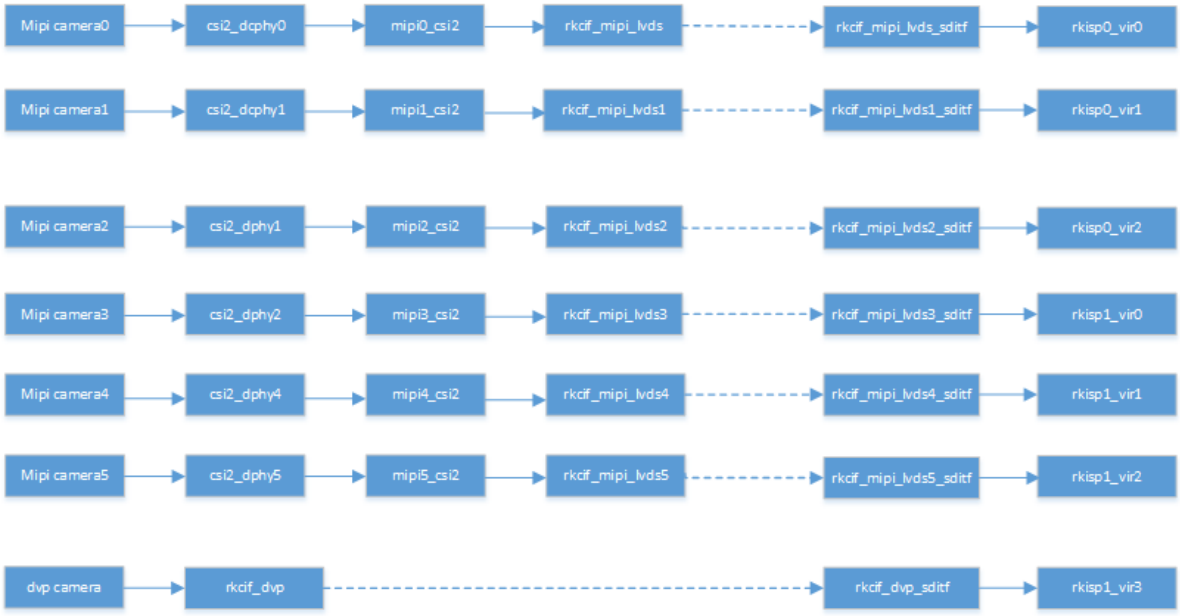
多sensor支持

单路硬件isp最多支持4路复用，isp复用情况支持分辨率如下：

2路复用：最大分辨率3840x2160，dts对应配置2路rkisp_vir设备。

3路或4路复用：最大分辨率2560x1536，dts对应配置3或4路rkisp_vir设备。

硬件支持最多采集7路sensor：6mipi + 1dvp，多sensor软件通路如下：



框图说明：

1. rk3588支持两个dcphy，节点名称分别为csi2_dcphy0/csi2_dcphy1。每个dcphy硬件支持RX/TX同时使用，对于camera输入使用的是RX。支持DPHY/CPHY协议复用；需要注意的是同一个dcphy的TX/RX只能同时使用DPHY或同时使用CPHY。其他dcphy参数请查阅rk3588数据手册。
2. rk3588支持2个dphy硬件，这里我们称之为dphy0_hw/dphy1_hw，两个dphy硬件都可以工作在full mode 和split mode两种模式下。
    1. dphy0_hw
        1. full mode：节点名称使用csi2_dphy0，最多支持4 lane。
        2. split mode： 拆分成2个phy使用，分别为csi2_dphy1（使用0/1 lane）、csi2_dphy2(使用2/3 lane)，每个phy最多支持2 lane。
        3. 当dphy0_hw使用full mode时，链路需要按照csi2_dphy1这条链路来配置，但是节点名称csi2_dphy1需要修改为csi2_dphy0，软件上是通过phy的序号来区分phy使用的模式。
    2. dphy1_hw
        1. full mode：节点名称使用csi2_dphy3，最多支持4 lane。

2. split mode： 拆分成2个phy使用，分别为csi2_dphy4（使用0/1 lane）、csi2_dphy5(使用2/3 lane)，每个phy最多支持2 lane。

3. 当dphy1_hw使用full mode时，链路需要按照csi2_dphy4这条链路来配置，但是节点名称csi2_dphy4需要修改为csi2_dphy3，软件上是通过phy的序号来区分phy使用的模式。

3. 使用上述mipi phy节点，需要把对应的物理节点配置上。（csi2_dcphy0_hw/csi2_dcphy1_hw/csi2_dphy0_hw/csi2_dphy1_hw）

4. 每个mipi phy都需要一个csi2模块来解析mipi协议，节点名称分别为mipi0_csi2~mipi5_csi2。

5. rk3588所有camera数据都需要通过vicap，再链接到isp。rk3588仅支持一个vicap硬件，这个vicap支持同时输入6路mipi phy，及一路dvp数据，所以我们将vicap分化成rkcif_mipi_lvds~rkcif_mipi_lvds5、rkcif_dvp等7个节点，各个节点的绑定关系需要严格按照框图的节点序号配置。

6. 每个vicap节点与isp的链接关系，通过对应虚拟出的XXX_sditf来指明链接关系。

7. rk3588支持2个isp硬件，每个isp设备可虚拟出多个虚拟节点，软件上通过回读的方式，依次从ddr读取每一路的图像数据进isp处理。对于多摄方案，建议将数据流平均分配到两个isp上。

8. 直通与回读模式：

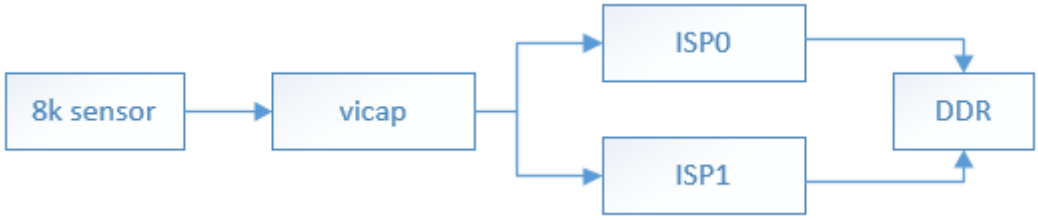
1. 直通：指数据经过vicap采集，直接发送给isp处理，不存储到ddr。需要注意的是hdr直通时，只有短帧是真正的直通，长帧需要存在ddr，isp再从ddr读取。

2. 回读：指数据经过vicap采集到ddr，应用获取到数据后，将buffer地址推送给isp，isp再从ddr获取图像数据。

3. 再dts配置时，一个isp硬件，如果只配置一个虚拟节点，默认使用直通模式，如果配置了多个虚拟节点默认使用回读模式。

双isp合成支持8K分辨率

vicap采集sensor 8k数据，然后左右图送给2个isp处理，再输出到ddr，流程如下：

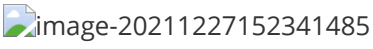


9. 分辨率大于16M（4672x3504）时需要同时使用两个isp来处理一张图像，只支持单摄。

1. 从rk3588s.dtsi文件中，可以找到rkisp0、rkisp1、rkisp_unite三种定义，当需要处理的分辨率大于16M，需要关闭rkisp0、rkisp1节点，并使能rkisp_unite，同时修改对应的iommu节点。
2. 使用rkisp_unite节点同样可以虚拟出多个节点，这个时候无论分辨率大小，同一张图像都是裁成左右两幅图，分别送2个isp处理，再合成一张输出。
3. dts配置参考rk3588单摄配置说明

8目MIPI sensor支持

RK3588硬件最多支持6路mipi sensor，可以通过RK1608拼接进行扩展，RK1608支持4路MIPI输入(1、2、4lane)，两路MIPI输出（1、2、4lane），通过RK1608拼接3路+原5路mipi即可实现8路MIPI sensor的输入，硬件连接如下：



软件上，需要在kernel的defconfig使能RK1608相关驱动：

```
CONFIG_VIDEO_ROCKCHIP_PREISP=y
CONFIG_VIDEO_PREISP_DUMMY_SENSOR=y
```

dts链路上，RK1608会抽象出RK1608 dphy节点，输入为虚拟的preisp_dmy sensor。RK1608 dphy同普通sensor一样，连接至RK3588 dcphy，形成RK1608->dcphy->csi->cif lvds链路。物理挂在RK1608的sensor在dts中直接挂在rkcif_mipi_lvdsx_sditf、rkcif_mipi_lvdsx_sditf_vir1、rkcif_mipi_lvdsx_sditf_vir2、rkcif_mipi_lvdsx_sditf_vir3的port0中，做为sditf设备的输入，sditf的输出同原通路一致，为对应的isp节点。dts链路示意如下：

RK1608 的dts配置主要在rk1608-dphy设备节点上，主要包括输入、输出通道及分辨率等信息，具体如下：

```
#define LINK_FREQ                700000000
    mipidphy0: mipidphy0 {
    compatible =  "rockchip,rk1608-dphy";
        status = "okay";
        //rockchip,grf = <&grf>;
        id = <0>;  //RK1608内部ID(0-2，按ID依次拼接)

        cam_nums = <1>;
        in_mipi = <1>;          //第一个sensor的输入mipi通道(0-3)
        out_mipi = <0>;         //输出mipi通道(0-1)
        link-freqs = /bits/ 64 <LINK_FREQ>; //MIPI速率

        sensor_i2c_bus = <5>;      //8目模式无效
        sensor_i2c_addr = <0x1a>; //8目模式无效
        sensor-name = "IMX464";

        rockchip,camera-module-index = <9>; //同普通sensor
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "TongJu";
        rockchip,camera-module-lens-name = "CHT842-MD";

        /* virtual-sensor mode */
        virtual-sub-sensor-config-0 {  //第二个sensor的配置信息
            id = <1>;  //RK1608内部ID(0-2，按ID依次拼接)
            in_mipi = <2>;
            out_mipi = <1>;
        };
        virtual-sub-sensor-config-1 { //第三个sensor的配置信息
            id = <2>;  //RK1608内部ID(0-2，按ID依次拼接)
            in_mipi = <3>;
            out_mipi = <1>;
        };
        /* multi-sensor mode end */

        format-config-0 {
            data_type = <0x2b>;
            mipi_lane = <2>;      //所接sensor的lane数
            mipi_lane_out = <4>; //RK1608输出的lane数
            field = <1>;       //以下同普通sensor配置
            colorspace = <8>;
            code = <MEDIA_BUS_FMT_SRGGB10_1X10>;
            width = <2712>;  //sensor输出分辨率
            height= <1538>;
            hactive = <2712>; //RK1608输出分辨率，宽相同，高为n*sensor
```

```
            vactive = <4614>;
            htotal = <3616>; //增加blank后的宽高，一般需要30%的blank
            vtotal = <4710>;
            inch0-info = <2712 1538 0x2b 0x2b 1>; //sensor输出分辨率
            outch0-info = <2712 4614 0x2b 0x2b 1>; //RK1608输出分辨率
            hcrop = <2560>; //crop后的单路分辨率
            vcrop = <1520>;
        };
        ...
```

sensor的dts配置参考如下：

```
&rkcif_mipi_lvds_sditf {
    #address-cells = <1>;
    #size-cells = <0>;
    status = "okay";
    rockchip,combine-index = <0>; //在RK1608拼接图中的顺序
    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
            mipi_lvds_sditf_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&imx464_out7>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;
            mipi_lvds_sditf: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&isp0_vir0>;
            };
        };
    };
};
```

CIS(cmos image sensor)驱动

CIS 设备注册(DTS)

MIPI接口

对于RV1126和RV1106平台而言，存在两个独立而完备的标准物理mipi csi2 dphy，对应于dts上的csi_dphy0和csi_dphy1（参见rv1126.dtsi），特性如下：

- data lane最大4 lanes；
- 最大速率2.5Gbps/lane；

对于RK356X平台而言，仅有一个标准物理mipi csi2 dphy，可以工作在两个模式：full mode 和split mode，拆分为csi2_dphy0/csi2_dphy1/csi2_dphy2三个逻辑dphy（参见rk3568.dtsi），特性如下：

Full mode

- 仅使用csi2_dphy0，csi2_dphy0与csi2_dphy1/csi2_dphy2互斥，不可同时使用；
- data lane最大4 lanes；
- 最大速率2.5Gbps/lane;

Split mode

- 仅使用csi2_dphy1和csi2_dphy2, 与csi2_dphy0互斥，不可同时使用；
- csi2_dphy1和csi2_dphy2可同时使用；
- csi2_dphy1和csi2_dphy2各自的data lane最大是2 lanes；
- csi2_dphy1对应物理dphy的lane0/lane1；
- csi2_dphy2对应物理dphy的lane2/lane3;
- 最大速率2.5Gbps/lane

具体dts用例，参见以下各示例。

链接ISP

RV1126/RV1106

下面以rv1126 isp和os04a10为例进行说明。

链接关系：sensor->csi_dphy->isp->ispp

arch/arm/boot/dts/rv1126-evb-v10.dtsi

配置要点

- data-lanes必须指明具体使用的lane数，否则无法识别为mipi 类型;

```
cam_ircut0: cam_ircut {
    status = "okay";
    compatible = "rockchip,ircut";
    ircut-open-gpios = <&gpio2 RK_PA7 GPIO_ACTIVE_HIGH>;
    ircut-close-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
    rockchip,camera-module-index = <1>;
    rockchip,camera-module-facing = "front";
};

os04a10: os04a10@36 {
    compatible = "ovti,os04a10";// 需要与驱动中的匹配字符串一致
    reg = <0x36>;// sensor I2C设备地址，7位
    clocks = <&cru CLK_MIPICSI_OUT>;// sensor clickin配置
    clock-names = "xvclk";
    power-domains = <&power RV1126_PD_VI>;
    pinctrl-names = "rockchip,camera_default";
    pinctrl-0 = <&mipi_csi_clk0>;// pinctl设置
    //电源
    avdd-supply = <&vcc_avdd>;
    dovdd-supply = <&vcc_dovdd>;
    dvdd-supply = <&vcc_dvdd>;
    // power管脚分配及有效电平
```

```
    pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
    // 模组编号，该编号不要重复
    rockchip,camera-module-index = <1>;
    // 模组朝向，有"back"和"front"
    rockchip,camera-module-facing = "front";
    // 模组名
    rockchip,camera-module-name = "CMK-OT1607-FV1";
    // lens名
    rockchip,camera-module-lens-name = "M12-4IR-4MP-F16";
    //ir cut设备
    ir-cut = <&cam_ircut0>;
    port {
        ucam_out0: endpoint {
            // mipi dphy端的port名
            remote-endpoint = <&mipi_in_ucam0>;
            // mipi lane数，1lane为 <1>, 4lane为 <1 2 3 4>
            data-lanes = <1 2 3 4>;
        };
    };
 };

&csi_dphy0 {
    status = "okay";
    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
            mipi_in_ucam0: endpoint@1 {
                reg = <1>;
                // sensor端的 port名
                remote-endpoint = <&ucam_out0>;
                // mipi lane数，1lane为 <1>, 4lane为 <1 2 3 4>
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;
            csidphy0_out: endpoint@0 {
                reg = <0>;
                // isp端的port名
                remote-endpoint = <&isp_in>;
            };
        };
    };
};

&rkisp {
    status = "okay";
};
```

```
&rkisp_vir0 {
    status = "okay";
    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
            isp_in: endpoint@0 {
                reg = <0>;
                // mipi dphy端的 port名
                remote-endpoint = <&csidphy0_out>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;
            isp0_out: endpoint@1 {
                reg = <1>;
                // ispp 端port名，isp输出给ispp
                remote-endpoint = <&ispp0_in>;
            };
        };
    };
};

&rkispp {
    status = "okay";
};

&rkispp_vir0 {
    status = "okay";
    port {
        #address-cells = <1>;
        #size-cells = <0>;
        Ispp0_in: endpoint@0 {
            reg = <0>;
            // isp端port名，ispp输入
            remote-endpoint = <&isp0_out>;
        };
    };
};
```

RK356X

下面以rk3566 isp和gc8034 4lane为例进行说明：

链接关系：sensor->csi2_dphy0->isp

配置要点

- 需要配置data-lanes
- 需要使能csi2_dphy_hw节点

```
    /* full mode: lane0-3 */
    gc8034: gc8034@37 {
        // 需要与驱动中的匹配字符串一致
        compatible = "galaxycore,gc8034";
        status = "okay";
        // sensor I2C设备地址，7位
        reg = <0x37>;
        // sensor mclk源配置
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        //sensor 相关电源域使能
        power-domains = <&power RK3568_PD_VI>;
        //sensor mclk pinctl设置
        pinctrl-names = "default";
        pinctrl-0 = <&cif_clk>;
        // reset管脚分配及有效电平
        reset-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_LOW>;
        // powerdown管脚分配及有效电平
        pwdn-gpios = <&gpio4 RK_PB2 GPIO_ACTIVE_LOW>;
        // 模组编号，该编号不要重复
        rockchip,camera-module-index = <0>;
        // 模组朝向，有"back"和"front"
        rockchip,camera-module-facing = "back";
        // 模组名
        rockchip,camera-module-name = "RK-CMK-8M-2-v1";
        // lens名
        rockchip,camera-module-lens-name = "CK8401";
        port {
            gc8034_out: endpoint {
                // csi2 dphy端的port名
                remote-endpoint = <&dphy0_in>;
                // csi2 dphy lane数，1lane为 <1>, 4lane为 <1 2 3 4>
                data-lanes = <1 2 3 4>;
            };
        };
    };

    &csi2_dphy_hw {
       status = "okay";
    };

    &csi2_dphy0 {
        //csi2_dphy0不与csi2_dphy1/csi2_dphy2同时使用，互斥
        status = "okay";
        /*
         * dphy0 only used for full mode,
         * full mode and split mode are mutually exclusive
         */
        ports {
            #address-cells = <1>;
            #size-cells = <0>;

            port@0 {
                reg = <0>;
                #address-cells = <1>;
```

```
                #size-cells = <0>;

                dphy0_in: endpoint@1 {
                    reg = <1>;
                    // sensor端的 port名
                    remote-endpoint = <&gc8034_out>;
                    // csi2 dphy lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor
端一致
                    data-lanes = <1 2 3 4>;
                };
            };

            port@1 {
                reg = <1>;
                #address-cells = <1>;
                #size-cells = <0>;

                dphy0_out: endpoint@1 {
                    reg = <1>;
                    // isp端的port名
                    remote-endpoint = <&isp0_in>;
                };
            };
        };
    };

&rkisp {
    status = "okay";
};

&rkisp_mmu {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp0_in: endpoint@0 {
            reg = <0>;
            // csi2 dphy端的 port名
            remote-endpoint = <&dphy0_out>;
        };
    };
};
```

链接VICAP

RV1126/RV1109

以mipi os04a10 4 lanes链接vicap为例：

链接关系：sensor->csi dphy->mipi csi host->vicap

配置要点：

- data-lanes必须指明具体使用的lane数，否则无法识别为mipi 类型;
- dphy需要链接到csi host节点。

```
os04a10: os04a10@36 {
        // 需要与驱动中的匹配字符串一致
        compatible = "ovti,os04a10";
        // sensor I2C设备地址，7位
        reg = <0x36>;
        // sensor mclk源配置
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        //sensor 相关电源域使能
        power-domains = <&power RV1126_PD_VI>;
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        //sensor mclk pinctl设置
        pinctrl-names = "rockchip,camera_default";
        pinctrl-0 = <&mipicsi_clk0>;
        // powerdown管脚分配及有效电平
        pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
        // 模组编号，该编号不要重复
        rockchip,camera-module-index = <1>;
        // 模组朝向，有"back"和"front"
        rockchip,camera-module-facing = "front";
        // 模组名
        rockchip,camera-module-name = "CMK-OT1607-FV1";
        // lens名
        rockchip,camera-module-lens-name = "M12-40IRC-4MP-F16";
        // ircut名
        ir-cut = <&cam_ircut0>;
        port {
            ucam_out0: endpoint {
                // csi2 dphy端的port名
                remote-endpoint = <&mipi_in_ucam0>;
                // csi2 dphy lane数，1lane为 <1>, 4lane为 <1 2 3 4>
                data-lanes = <1 2 3 4>;
            };
        };
    };

&csi_dphy0 {
    //csi2_dphy0不与csi2_dphy1/csi2_dphy2同时使用，互斥
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;
```

```
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_in_ucam0: endpoint@1 {
                reg = <1>;
                // sensor端的 port名
                remote-endpoint = <&ucam_out0>;
                // csi2 dphy lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor端一致
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                // csi2 host端的port名
                remote-endpoint = <&mipi_csi2_input>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                // csi2 dphy 端的port名
                remote-endpoint = <&csidphy0_out>;
                // csi2 host lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor端一致
                data-lanes = <1 2 3 4>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
```

```
                // vicap端的port名
                remote-endpoint = <&cif_mipi_in>;
                // csi2 host lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor端一致
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        /* MIPI CSI-2 endpoint */
        cif_mipi_in: endpoint {
            // csi2 host端的port名
            remote-endpoint = <&mipi_csi2_output>;
            // vicap 端 lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor端一致
            data-lanes = <1 2 3 4>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        /* sditf endpoint */
        mipi_lvds_sditf: endpoint {
            //isp 虚拟设备端port名
            remote-endpoint = <&isp_in>;
            //mipi csi2 dphy的lane数，与sensor一致
            data-lanes = <1 2 3 4>;
        };
    };
};

&rkisp {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in: endpoint@0 {
                reg = <0>;
                //vicap sditf的端点名
                remote-endpoint = <&mipi_lvds_sditf>;
            };
```

```
        };
    };
};
```

RK356X

以gc5025 2lane链接rk3566 evb2 mipi csi2 dphy的lane2/lane3为例：

链接关系：sensor->csi2 dphy->mipi csi host->vicap

配置要点

- data-lanes必须指明具体使用的lane数，否则无法识别为mipi 类型;
- dphy需要链接到csi host节点;
- 需要使能csi2 dphy hw节点。

```
/* split mode: lane:2/3 */
gc5025: gc5025@37 {
        status = "okay";
        // 需要与驱动中的匹配字符串一致
        compatible = "galaxycore,gc5025";
        // sensor I2C设备地址，7位
        reg = <0x37>;
        // sensor mclk源配置
        clocks = <&pmucru CLK_WIFI>;
        clock-names = "xvclk";
        //sensor mclk pinctl设置
        pinctrl-names = "default";
        pinctrl-0 = <&refclk_pins>;
        // reset管脚分配及有效电平
        reset-gpios = <&gpio3 RK_PA5 GPIO_ACTIVE_LOW>;
        // powerdown管脚分配及有效电平
        pwdn-gpios = <&gpio3 RK_PB0 GPIO_ACTIVE_LOW>;
        //sensor 相关电源域使能
        power-domains = <&power RK3568_PD_VI>;
        /*power-gpios = <&gpio0 RK_PC1 GPIO_ACTIVE_HIGH>;*/
        // 模组编号，该编号不要重复
        rockchip,camera-module-index = <1>;
        // 模组朝向，有"back"和"front"
        rockchip,camera-module-facing = "front";
        // 模组名
        rockchip,camera-module-name = "TongJu";
        // lens名
        rockchip,camera-module-lens-name = "CHT842-MD";
        port {
            gc5025_out: endpoint {
                // csi2 dphy端的port名
                remote-endpoint = <&dphy2_in>;
                // csi2 dphy lane数，2lane为 <1 2>, 4lane为 <1 2 3 4>
                data-lanes = <1 2>;
            };
        };
};

 &csi2_dphy_hw {
         status = "okay";
```

```
 };

&csi2_dphy2 {
    //csi2_dphy2不与csi2_dphy0同时使用，互斥;可与csi2_dphy1并行使用
    status = "okay";

    /*
     * dphy2 only used for split mode,
     * can be used  concurrently  with dphy1
     * full mode and split mode are mutually exclusive
     */
    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy2_in: endpoint@1 {
                reg = <1>;
                // sensor端的 port名
                remote-endpoint = <&gc5025_out>;
                // csi2 dphy lane数，2lane为 <1 2>, 4lane为 <1 2 3 4>,需与sensor端
一致
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy2_out: endpoint@1 {
                reg = <1>;
                // csi2 host端的port名
                remote-endpoint = <&mipi_csi2_input>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
```

```
            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                // csi2 dphy 端的port名
                remote-endpoint = <&dphy2_out>;
                // csi2 host lane数，2lane为 <1 2>, 4lane为 <1 2 3 4>,需与sensor端
一致
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                // vicap端的port名
                remote-endpoint = <&cif_mipi_in>;
                // csi2 host lane数，1lane为 <1>, 4lane为 <1 2 3 4>,需与sensor端一致
                data-lanes = <1 2>;
            };
        };
    };
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        cif_mipi_in: endpoint {
            // csi2 host端的port名
            remote-endpoint = <&mipi_csi2_output>;
            // vicap 端 lane数，2lane为 <1 2>, 4lane为 <1 2 3 4>,需与sensor端一致
            data-lanes = <1 2>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        /* MIPI CSI-2 endpoint */
        mipi_lvds_sditf: endpoint {
            //isp 虚拟设备端port名
            remote-endpoint = <&isp_in>;
            //mipi csi2 dphy的lane数，与sensor一致
            data-lanes = <1 2>;
        };
    };
};

&rkisp {
```

```
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in: endpoint@0 {
                reg = <0>;
                //vicap mipi sditf的端点名
                remote-endpoint = <&mipi_lvds_sditf>;
            };
        };
    };
};
```

RK3588

以imx464接dphy1为例

- data-lanes必须指明具体使用的lane数，否则无法识别为mipi 类型;
- dphy需要链接到csi host节点，csi2_dphy3对应使用mipi4_csi2；
- csi2_dphy3只是逻辑节点，需要依赖物理节点csi2_dphy1_hw。
- rkcif_mipi_lvds4是vicap的其中一个逻辑节点，物理节点rkcif及对应iommu需要配置上。
- rkcif_mipi_lvds4_sditf是虚拟子节点，是rkcif_mipi_lvds4的虚拟节点，用来链接isp。
- sensor驱动一般实现avdd/dvdd/dovdd三个电源操作，如果使用类似rk809分配出来的电源可以直接在sensor节点配置上，如果使用LDO等外部电源，使能脚是通过gpio控制，可以参考vcc_mipicsi1配置成电源节点，可以通过引用计数来上下电，适用于多设备使用同一电源。建议多摄时dvdd电源分开供应，avdd/dovdd可以共用，dvdd共用时，如果功率比较大，可能出现瞬时供应不足，电源有塌陷，影响到图像质量，甚至不出图。

```
/ {
    vcc_mipicsi1: vcc-mipicsi1-regulator {
        compatible = "regulator-fixed";
        gpio = <&gpio4 RK_PA6 GPIO_ACTIVE_HIGH>;
        pinctrl-names = "default";
        pinctrl-0 = <&mipicsi1_pwr>;
        regulator-name = "vcc_mipicsi1";
        enable-active-high;
    };

};

&csi2_dphy1_hw {
    status = "okay";
};

&csi2_dphy3 {
    status = "okay";
```

```
    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_in_ucam: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&imx464_out>;
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy3_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&mipi4_csi2_input>;
            };
        };
    };
};

&i2c4 {
    status = "okay";
    pinctrl-0 = <&i2c4m3_xfer>;

    imx464: imx464@36 {
        compatible = "sony,imx464";
        status = "okay";
        reg = <0x36>;
        clocks = <&cru CLK_MIPI_CAMARAOUT_M4>;
        clock-names = "xvclk";
        pinctrl-names = "default";
        pinctrl-0 = <&mipim0_camera4_clk>;
        avdd-supply = <&vcc_mipicsi1>;
        reset-gpios = <&gpio1 RK_PD6 GPIO_ACTIVE_HIGH>;
        pwdn-gpios = <&gpio3 RK_PC1 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "CMK-OT1980-PX1";
        rockchip,camera-module-lens-name = "SHG102";
        port {
            imx464_out: endpoint {
                remote-endpoint = <&mipi_in_ucam>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};
```

```
&mipi4_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi4_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&csidphy3_out>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi4_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in4>;
            };
        };
    };
};

&pinctrl {
    cam {
        mipicsi1_pwr: mipicsi1-pwr {
            rockchip,pins =
                /* camera power en */
                <4 RK_PA6 RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
};

&rkcif {
    status = "okay";
};

&rkcif_mipi_lvds4 {
    status = "okay";

    port {
        cif_mipi_in4: endpoint {
            remote-endpoint = <&mipi4_csi2_output>;
        };
    };
};
```

```
&rkcif_mipi_lvds4_sditf {
    status = "okay";

    port {
        mipi4_lvds_sditf: endpoint {
            remote-endpoint = <&isp0_vir0>;
        };
    };
};

&rkcif_mmu {
    status = "okay";
};

#if 1
&rkisp0 {
    status = "okay";
    /* the max input w h and fps of mulit sensor */
    //max-input = <2688 1520 30>;多摄sensor分辨率不一样，需要配置
};

&isp0_mmu {
    status = "okay";
};
#else //sensor分辨率大于16M(4672x3504)需要2个isp合成处理
/* dual isp case need width 32 align,  height 8 align */
&rkisp_unite_mmu {
    status = "okay";
};

&rkisp_unite {
       status = "okay";
};

&rkisp0_vir0 {
       status = "okay";
       rockchip,hw = <&rkisp_unite>;
};
#endif

&rkisp0_vir0 {
    status = "okay";
    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp0_vir0: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&mipi4_lvds_sditf>;
        };
    };
};
```

RV1106

dts配置参考arch/arm/boot/dts/rv1106-evb-cam.dtsi

LVDS接口

链接VICAP

RV1126/RV1109

以imx327 4lane为例, 链接关系如下:

链接关系：sensor->csi dphy->vicap

配置要点

- dphy不需要链接csi host节点，否则会导致收不到数据;
- data-lanes必须指明具体使用的lane数，否则会导致收不到数据;
- bus-type 必须配置为 3，否则无法识别为lvds接口，导致链路建立失败;

```
imx327: imx327@1a {
        // 需要与驱动中的匹配字符串一致
        compatible = "sony,imx327";
        // sensor I2C设备地址，7位
        reg = <0x1a>;
        // sensor mclk源配置
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        //sensor 相关电源域使能
        power-domains = <&power RV1126_PD_VI>;
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        //sensor mclk pinctl设置
        pinctrl-names = "default";
        pinctrl-0 = <&mipicsi_clk0>;
        // powerdown管脚分配及有效电平
        pwdn-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_HIGH>;
        // reset管脚分配及有效电平
        reset-gpios = <&gpio1 RK_PD5 GPIO_ACTIVE_HIGH>;
        // 模组编号，该编号不要重复
        rockchip,camera-module-index = <1>;
        // 模组朝向，有"back"和"front"
        rockchip,camera-module-facing = "front";
        // 模组名
        rockchip,camera-module-name = "CMK-OT1607-FV1";
        // lens名
        rockchip,camera-module-lens-name = "M12-4IR-4MP-F16";
        // ircut名
        ir-cut = <&cam_ircut0>;
        port {
            ucam_out0: endpoint {
                // csi2 dphy端的port名
                remote-endpoint = <&mipi_in_ucam0>;
                //csi2 dphy lvds lane数，1lane为 <1>, 4lane为 <4>，必须指定
                data-lanes = <4>;
                //lvds接口的类型，必须指定
```

```
                bus-type = <3>;
            };
        };
};

&csi_dphy0 {
    //csi2_dphy0不与csi2_dphy1/csi2_dphy2同时使用，互斥
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_in_ucam0: endpoint@1 {
                reg = <1>;
                // sensor端的 port名
                remote-endpoint = <&ucam_out0>;
                //csi2 dphy lvds lane数，1lane为 <1>, 4lane为 <4>，必须指定
                data-lanes = <4>;
                //lvds接口的类型，必须指定
                bus-type = <3>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                // vicap lite端的port名
                remote-endpoint = <&cif_lite_lvds_in>;
                //csi2 dphy lvds lane数，1lane为 <1>, 4lane为 <4>，必须指定
                data-lanes = <4>;
                //lvds接口的类型，必须指定
                bus-type = <3>;
            };
        };
    };
};

&rkcif_lite_mipi_lvds {
    status = "okay";

    port {
        /* lvds endpoint */
        cif_lite_lvds_in: endpoint {
            // csi2 dphy端的port名
            remote-endpoint = <&csidphy0_out>;
            //csi2 dphy lvds lane数，1lane为 <1>, 4lane为 <4>，必须指定
            data-lanes = <4>;
```

```
            //lvds接口的类型，必须指定
            bus-type = <3>;
        };
    };
};

&rkcif_lite_sditf {
    status = "okay";

    port {
        /* lvds endpoint */
        lite_sditf: endpoint {
            //isp 虚拟设备端port名
            remote-endpoint = <&isp_in>;
            //csi2 dphy的lane数，与sensor一致
            data-lanes = <4>;
        };
    };
};

&rkisp {
    status = "okay";
};

&isp0_mmu {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in: endpoint@0 {
                reg = <0>;
                //lite vicap lvds sditf的端点名
                remote-endpoint = <&lite_sditf>;
            };
        };
    };
};
```

需要使用双isp处理时，可以按照下面修改：

```
&rkisp {
    status = "disabled";
};

&isp0_mmu {
    status = "disabled";
};
```

```
&rkisp_unite {
    status = "okay";
};

&rkisp_unite_mmu {
    status = "okay";
};

&rkisp0_vir0 {
    status = "okay";

    /* hw 引用成unite节点*/
    rockchip,hw = <&rkisp_unite>;

    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp0_vir0: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&mipi4_lvds_sditf>;
        };
    };
};
```

DVP接口

链接VICAP

在RV1126/RV1109/RK356X/RK3588平台上，dvp各相关接口的dts配置是一样的。

BT601

以ar0230 bt601为例，链接关系如下：

链接关系：sensor->vicap

配置要点

- hsync-active/vsync-active必须配置，用于v4l2框架异步注册识别BT601接口,若不配置会识别为BT656接口；
- pclk-sample/bus-width可选；
- 必须在sensor驱动的g_mbus_config接口中，通过flag指明当前sensor的hsync-acitve/vsync-active/pclk-ative的有效极性，否则会导致无法收到数据；
- pinctrl需要引用对，以对bt601相关gpio做相应iomux，否则会导致无法收到数据；

g_mbus_config接口示例代码如下：

```
static int ar0230_g_mbus_config(struct v4l2_subdev *sd,
                struct v4l2_mbus_config *config)
{
    config->type = V4L2_MBUS_PARALLEL;
    config->flags = V4L2_MBUS_HSYNC_ACTIVE_HIGH |
            V4L2_MBUS_VSYNC_ACTIVE_HIGH |
            V4L2_MBUS_PCLK_SAMPLE_FALLING;
    return 0;
}
```

dts配置示例如下：

```
ar0230: ar0230@10 {
        // 需要与驱动中的匹配字符串一致
        compatible = "aptina,ar0230";
        // sensor I2C设备地址，7位
        reg = <0x10>;
        // sensor mclk源配置
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        //sensor 相关电源域使能
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        power-domains = <&power RV1126_PD_VI>;
        // powerdown管脚分配及有效电平
        pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
        /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
        //配置dvp相关数据管脚和时钟管脚
        pinctrl-names = "default";
        pinctrl-0 = <&cifm0_dvp_ctl>;
        // 模组编号，该编号不要重复
        rockchip,camera-module-index = <0>;
        // 模组朝向，有"back"和"front"
        rockchip,camera-module-facing = "back";
        // 模组名
        rockchip,camera-module-name = "CMK-OT0836-PT2";
        // lens名
        rockchip,camera-module-lens-name = "YT-2929";
        port {
            cam_para_out1: endpoint {
                remote-endpoint = <&cif_para_in>;
            };
        };
};

&rkcif_dvp {
    status = "okay";

    port {
        /* Parallel bus endpoint */
        cif_para_in: endpoint {
            //sensor端endpoint名
            remote-endpoint = <&cam_para_out1>;
```

```
            //sensor端相关配置参数
            bus-width = <12>;
            hsync-active = <1>;
            vsync-active = <1>;
            pclk-sample = <0>;
        };
    };
};

&rkcif_dvp_sditf {
    status = "okay";

    port {
        /* parallel endpoint */
        dvp_sditf: endpoint {
            //isp 虚拟设备端port名
            remote-endpoint = <&isp_in>;
            //sensor端相关配置参数
            bus-width = <12>;
            hsync-active = <1>;
            vsync-active = <1>;
            pclk-sample = <0>;
        };
    };
};

&rkisp {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in: endpoint@0 {
                reg = <0>;
                //dvp sditf的端点名
                remote-endpoint = <&dvp_sditf>;
            };
        };
    };
};
```

BT656/BT1120

BT656/BT1120的dts用法一致。

以ava fpga bt1120为例，链接关系如下：

链接关系： sensor->vicap

配置要点

- hsync-active/vsync-active不要配置，否则v4l2框架异步注册时会识别为BT601；
- pclk-sample/bus-width可选；
- 必须在sensor驱动的g_mbus_config接口中，通过flag变量指明当前sensor的pclk-ative的有效极性，否则会导致无法收到数据；
- 必须实现v4l2_subdev_video_ops中的querystd接口，指明当前接口为ATSC接口,否则会导致无法收到数据；
- 必须实现RKMODULE_GET_BT656_MBUS_INFO，BT656/BT1120都是调用这个ioctl，接口兼容，实现参考drivers/media/i2c/nvp6158_drv/nvp6158_v4l2.c
- pinctrl需要引用对，以对bt656/bt1120相关gpio做相应iomux，否则会导致无法收到数据。

g_mbus_config接口示例代码如下：

```
static int avafpga_g_mbus_config(struct v4l2_subdev *sd,
                 struct v4l2_mbus_config *config)
{
    config->type = V4L2_MBUS_BT656;
    config->flags = V4L2_MBUS_PCLK_SAMPLE_RISING;

    return 0;
}
```

querystd接口示例如下：

```
static int avafpga_querystd(struct v4l2_subdev *sd, v4l2_std_id *std)
{
    *std = V4L2_STD_ATSC;

    return 0;
}
```

dts配置示例如下：

```
avafpga: avafpga@70 {
    // 需要与驱动中的匹配字符串一致
    compatible = "ava,fpga";
    // sensor I2C设备地址，7位
    reg = <0x10>;
    // sensor mclk源配置
    clocks = <&cru CLK_CIF_OUT>;
    clock-names = "xvclk";
    //sensor 相关电源域使能
    avdd-supply = <&vcc_avdd>;
    dovdd-supply = <&vcc_dovdd>;
    dvdd-supply = <&vcc_dvdd>;
    // powerdown管脚分配及有效电平
    power-domains = <&power RV1126_PD_VI>;
    pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
    /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
    //配置dvp相关数据管脚和时钟管脚
    pinctrl-names = "default";
    pinctrl-0 = <&cifm0_dvp_ctl>;
    // 模组编号，该编号不要重复
```

```
    rockchip,camera-module-index = <0>;
    // 模组朝向，有"back"和"front"
    rockchip,camera-module-facing = "back";
    // 模组名
    rockchip,camera-module-name = "CMK-OT0836-PT2";
    // lens名
    rockchip,camera-module-lens-name = "YT-2929";
    port {
        cam_para_out2: endpoint {
            remote-endpoint = <&cif_para_in>;
        };
    };
};

&rkcif_dvp {
    status = "okay";

    port {
        /* Parallel bus endpoint */
        cif_para_in: endpoint {
            //sensor端endpoint名
            remote-endpoint = <&cam_para_out2>;
            //sensor端相关配置参数，可选
            bus-width = <16>;
            pclk-sample = <1>;
        };
    };
};

&rkcif_dvp_sditf {
    status = "okay";

    port {
        /* parallel endpoint */
        dvp_sditf: endpoint {
            //isp 虚拟设备端port名
            remote-endpoint = <&isp_in>;
            bus-width = <16>;
            pclk-sample = <1>;
        };
    };
};

&rkisp {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
```

```
            isp_in: endpoint@0 {
                reg = <0>;
                //dvp sditf的端点名
                remote-endpoint = <&dvp_sditf>;
            };
        };
    };
};
```

多sensor 注册

单个硬件isp通过虚拟多个设备，分时复用处理多路raw sensor数据。

对于rv1109/rv1126/rk356x vicap采集dvp raw数据只能按非紧凑存储, isp处理raw默认按紧凑存储，对于dvp raw数据进isp处理的注意事项，请参考如何配置ISP/VICAP RAW存储格式。

RV1126/RV1109

链接关系，isp0->ispp0和isp1->ispp1是固定配置rv1126.dtsi
rv1109/rv1126 isp/ispp，带宽允许的情况下最多可以4路复用，可自行在rv1126.dtsi增加
rkisp_vir0~rkisp_vir4/rkispp_vir0~rkispp_vir4
mipi进isp或cif进isp可选。

rv1109/rv1126支持2个phy接口，每个phy可复用为mipi/lvds，且最多支持4lane
rv1109/rv1126支持1个dvp接口，支持BT601/BT656/BT1120
isp支持mipi或dvp输入：mipi/dvp只能2选1，无法同时工作
vicap 支持mipi/lvds/dvp：mipi/lvds是复用关系，无法同时使用，dvp可以与前者同时使用
vicap lite 仅支持lvds
通过对以上硬件配置的了解，rv1109/rv1126目前最多支持3路raw sensor进isp处理

双摄进isp处理

sensor0（mipi）->csi_dphy0->csi2->vicap->isp0->ispp0
sensor1（mipi）->csi_dphy1->isp1->ispp1

实例参考：arch/arm/boot/dts/rv1109-evb-ddr3-v12-facial-gate.dts
gc2053->csi_dphy0->csi2->vicap->isp1->ispp1
ov2718->csi_dphy1->isp0->ispp0
对于不同分辨率如下配置很重要

```
&rkispp {
   status = "okay";
   /* the max input w h and fps of mulit sensor */
   max-input = <2688 1520 30>;//取不同sensor的最大宽和高及帧率
};
```

三摄进isp处理

sensor0（mipi）->csi_dphy0->csi2->vicap->isp0->ispp0
sensor1（mipi）->csi_dphy1->isp1->ispp1
sensor2（DVP）->vicap->isp2->ispp2
或
sensor0（mipi）->csi_dphy0->csi2->vicap->isp0->ispp0

sensor1（lvds）->csi_dphy1->vicap lite->isp1->ispp1
sensor2（DVP）->vicap->isp2->ispp2

实例参考：
bf2253-0(mipi)->dphy0->csi2->vicap(mipi)->isp0->ispp0
bf2253-1(mipi)->dphy1->isp1->ispp1
gc1054(dvp)->vicap(dvp)->isp2->ispp2

```
&i2c1 {
    status = "okay";
     clock-frequency = <400000>;

    gc1054: gc1054@21 {
               compatible = "galaxycore,gc1054";
            reg = <0x21>;
               clocks = <&cru CLK_CIF_OUT>;
            clock-names = "xvclk";
            power-domains = <&power RV1126_PD_VI>;

            pwdn-gpios  = <&gpio3 RK_PA5 GPIO_ACTIVE_HIGH>;
            reset-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_LOW>;

            rockchip,grf = <&grf>;
            pinctrl-names = "default";
            pinctrl-0 = <&cifm0_dvp_ctl>;

            rockchip,camera-module-index = <0>;
            rockchip,camera-module-facing = "back";
            rockchip,camera-module-name = "GC1054_B";
            rockchip,camera-module-lens-name = "GC1054_LEN";
               port {
                    cam_para_out1: endpoint {
                    remote-endpoint = <&cif_para_in>;
                        bus-width = <10>;
                        hsync-active = <1>;
                        vsync-active = <1>;
            };
        };
    };

    bf2253_isp0: bf2253_isp0@6d {
          compatible = "ovti,bf2253_isp0";
          reg = <0x6d>;
          clocks = <&cru CLK_MIPICSI_OUT>;
          clock-names = "xvclk";
          power-domains = <&power RV1126_PD_VI>;
          pinctrl-names = "rockchip,camera_default";
          pinctrl-0   = <&mipicsi_clk0>;
          power-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_HIGH>;
          pwdn-gpios  = <&gpio1 RK_PD4 GPIO_ACTIVE_LOW>;
          reset-gpios = <&gpio1 RK_PD5 GPIO_ACTIVE_HIGH>;

          avdd-supply  = <&vcc_3v3>;
          dovdd-supply = <&vcc_1v8>;
          dvdd-supply  = <&vcc_1v8>;
```

```
          rockchip,camera-module-index = <1>;
          rockchip,camera-module-facing = "front";
          rockchip,camera-module-name = "LA6110PA";
          rockchip,camera-module-lens-name = "YM6011P";
          port {
              cam_out1: endpoint {
                    remote-endpoint = <&mipi_in_ucam>;
                    data-lanes = <1>;
                };
          };
     };
};


&i2c3 {
    status = "okay";
    clock-frequency = <400000>;
    pinctrl-names = "default";
    pinctrl-0 = <&i2c3m2_xfer>;

    bf2253_isp1: bf2253_isp1@6d {
          compatible = "ovti,bf2253_isp1";
          reg = <0x6d>;
          clocks = <&cru CLK_MIPICSI_OUT>;
          clock-names = "xvclk";
          power-domains = <&power RV1126_PD_VI>;
          pinctrl-names = "rockchip,camera_default";
          //pinctrl-names = "rockchip,camera_sleep";
          pinctrl-0 = <&mipicsi_clk1>;

          power-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_HIGH>;
          pwdn-gpios  = <&gpio3 RK_PA4 GPIO_ACTIVE_LOW>;
          reset-gpios = <&gpio2 RK_PA0 GPIO_ACTIVE_HIGH>;

          avdd-supply  = <&vcc_3v3>;
          dovdd-supply = <&vcc_1v8>;
          dvdd-supply  = <&vcc_1v8>;

          rockchip,camera-module-index = <2>;
          rockchip,camera-module-facing = "front";
          rockchip,camera-module-name = "LA6110PA";
          rockchip,camera-module-lens-name = "YM6011P";
          port {
               cam_out0: endpoint {
                   remote-endpoint = <&csi_dphy1_input>;
                   data-lanes = <1>;
               };
          };
     };
};

&csi_dphy0 {
    status = "okay";
    ports {
```

```
        port@0 {
            mipi_in_ucam: endpoint@1 {
                remote-endpoint = <&cam_out1>;
                data-lanes = <1>;
            };
        };
        port@1 {
            csi_dphy0_out: endpoint@0 {
                remote-endpoint = <&mipi_csi2_input>;
                data-lanes = <1>;
            };
        };
    };

};

&csi_dphy1 {
    status = "okay";
    ports {
        port@0 {
            csi_dphy1_input: endpoint@1 {
                remote-endpoint = <&cam_out0>;
                data-lanes = <1>;
            };
        };
        port@1 {
            csi_dphy1_output: endpoint@0 {
                remote-endpoint = <&isp_in1>;
                data-lanes = <1>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";
    ports {
        port@0 {
            mipi_csi2_input: endpoint@1 {
                remote-endpoint = <&csi_dphy0_out>;
                data-lanes = <1>;
            };
        };

        port@1 {
            mipi_csi2_output: endpoint@0 {
                remote-endpoint = <&cif_mipi_in>;
                data-lanes = <1>;
            };
        };
    };
};

&rkcif_mipi_lvds {
    status = "okay";
```

```
    port {
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
            data-lanes = <1>;
        };
    };
};
&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        lvds_sditf: endpoint {
            remote-endpoint = <&isp_in0>;
            data-lanes = <1>;
        };
    };
};

&rkcif_dvp {
        status = "okay";
        //iommus = <&rkcif_mmu>;
        ///delete-property/ memory-region;
        port {
                /* Parallel bus endpoint */
                cif_para_in: endpoint {
                        remote-endpoint = <&cam_para_out1>;
                        bus-width = <8>;
                        hsync-active = <1>;
                        vsync-active = <1>;
                };
        };
};

&rkcif_dvp_sditf {
        status = "okay";
        port {
                /* Parallel bus endpoint */
        dvp_sditf: endpoint {
                        remote-endpoint = <&isp_in2>;
                };
        };
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in0: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&lvds_sditf>;
```

```
            };
        };
    };
};

&rkisp_vir1 {
    status = "okay";
    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in1: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&csi_dphy1_output>;

            };
        };
    };
};

&rkisp_vir2 {
    status = "okay";
    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp_in2: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&dvp_sditf>;
            };
        };
    };
};

&rkispp_vir0 {
    status = "okay";
};

&rkispp_vir1 {
    status = "okay";
};

&rkispp_vir2 {
        status = "okay";
};

&rkcif {
        status = "okay";
};

rkisp: rkisp@ffb50000 {
```

```
    status = "okay";
};

&rkispp {
    status = "okay";
    max-input = <1600 1200 30>;
    memory-region = <&isp_reserved>;
    /* the max input w h and fps of mulit sensor */
};

 rkcif_mmu: iommu@ffae0800{
    status = "disabled";
};

 rkisp_mmu: iommu@ffb51a00 {
    status = "disabled";
};

&rkispp_mmu {
    status = "disabled";
};
```

RK3566/RK3568

rk356x isp，带宽允许的情况下最多可以4路复用，可自行在rk3568.dtsi增加rkisp_vir0~rkisp_vir4
mipi进isp或cif进isp可选。

rk356x支持1个4lane phy接口，这个phy可分成2个2lane的phy使用
rk356x支持1个dvp接口，支持BT601/BT656/BT1120
isp支持mipi或dvp输入：mipi/dvp只能2选1，无法同时工作
vicap 支持mipi/dvp：mipi与dvp可以同时使用
通过对以上硬件配置的了解，rk356x目前最多支持3路raw sensor进isp处理

双摄进isp处理：

参考实例：
ov5695->dphy1->isp_vir0
gc5025->dphy2->csi2->vicap->isp_vir1

```
ov5695: ov5695@36 {
        status = "okay";
        ...
        port {
            ov5695_out: endpoint {
                remote-endpoint = <&dphy1_in>;
                data-lanes = <1 2>;
            };
        };
};

gc5025: gc5025@37 {
        status = "okay";
        ...
        port {
```

```
            gc5025_out: endpoint {
                remote-endpoint = <&dphy2_in>;
                data-lanes = <1 2>;
            };
        };
};

 &csi2_dphy_hw {
         status = "okay";
 };

&csi2_dphy1 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy1_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&ov5695_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy1_out: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&isp0_in>;
            };
        };
    };
};

&csi2_dphy2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
```

```
            dphy2_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&gc5025_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy2_out: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&mipi_csi2_input>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&dphy2_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in>;
                data-lanes = <1 2>;
            };
        };
    };
};

&rkcif_mipi_lvds {
    status = "okay";
```

```
    port {
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
            data-lanes = <1 2>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        mipi_lvds_sditf: endpoint {
            remote-endpoint = <&isp1_in>;
            data-lanes = <1 2>;
        };
    };
};

&rkisp {
    status = "okay";
    /* the max input w h and fps of mulit sensor */
    max-input = <2592 1944 30>;
};

&rkisp_vir0 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp0_in: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&dphy1_out>;
            };
        };
    };
};

&rkisp_vir1 {
    status = "okay";

    ports {
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            isp1_in: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&mipi_lvds_sditf>;
```

```
            };
        };
    };
};
```

三摄进isp处理：

参考实例：
gc2053(mipi)->dphy1->isp0
sc1330(dvp)->vicap(dvp)->isp1
ov5695(mipi)->dphy2->csi2->vicap(mipi)->isp2

```
&i2c2 {
    status = "okay";
    pinctrl-0 = <&i2c2m1_xfer>;

    /* split mode: lane0/1 */
    gc2053: gc2053@37 {
        status = "okay";
        compatible = "galaxycore,gc2053";
        reg = <0x37>;
        clocks = <&cru CLK_CAM0_OUT>;
        clock-names = "xvclk";
        /* Set pinctl of xvclk in &pinctl */
        power-domains = <&power RK3568_PD_VI>;
        reset-gpios = <&gpio4 RK_PB1 GPIO_ACTIVE_LOW>;
        pwdn-gpios = <&gpio3 RK_PD0 GPIO_ACTIVE_LOW>;
        /*power-gpios = <&gpio0 RK_PC1 GPIO_ACTIVE_HIGH>;*/
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "front";
        rockchip,camera-module-name = "rgbd";
        rockchip,camera-module-lens-name = "Optics";
        port {
            gc2053_out: endpoint {
                remote-endpoint = <&dphy1_in>;
                data-lanes = <1 2>;
            };
        };
    };
};

&i2c3 {
    status = "okay";
    pinctrl-0 = <&i2c3m0_xfer>;

    sc1330: sc1330@32 {
        status = "okay";
        compatible = "smartsens,sc1330";
        reg = <0x32>;
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        power-domains = <&power RK3568_PD_VI>;
        pinctrl-names = "default";
        /* conflict with gmac1m1_rgmii_pins & cif_clk*/
```

```
        pinctrl-0 = <&cif_clk &cif_dvp_clk &cif_dvp_bus10>;

        /*avdd-supply = <&vcc2v8_dvp>;*/
        /*dovdd-supply = <&vcc1v8_dvp>;*/
        /*dvdd-supply = <&vcc1v8_dvp>;*/

        reset-gpios = <&gpio4 RK_PA6 GPIO_ACTIVE_LOW>;
        pwdn-gpios = <&gpio3 RK_PC7 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <2>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "default";
        rockchip,camera-module-lens-name = "default";
        port {
            sc1330_out: endpoint {
                remote-endpoint = <&dvp_in_bcam>;
            };
        };
    };
};

&i2c4 {
    status = "okay";
    pinctrl-0 = <&i2c4m0_xfer>;
    clock-frequency = <1000000>;

    /* split mode: lane:2/3 */
    ov5695: ov5695@36 {
                status = "okay";
                compatible = "ovti,ov5695";
                reg = <0x36>;
                clocks = <&cru CLK_CAM0_OUT>;
                clock-names = "xvclk";
                power-domains = <&power RK3568_PD_VI>;
                pinctrl-names = "default";
                pinctrl-0 = <&cif_clk>;
                reset-gpios = <&gpio3 RK_PB0 GPIO_ACTIVE_HIGH>;
                pwdn-gpios = <&gpio4 RK_PC6 GPIO_ACTIVE_HIGH>;
                rockchip,camera-module-index = <1>;
                rockchip,camera-module-facing = "front";
                rockchip,camera-module-name = "TongJu";
                rockchip,camera-module-lens-name = "CHT842-MD";
                port {
                        ov5695_out: endpoint {
                                remote-endpoint = <&dphy2_in>;
                                data-lanes = <1 2>;
                        };
                };
    };
};

&csi2_dphy_hw {
    status = "okay";
};

&csi2_dphy1 {
```

```
    status = "okay";

    /*
     * dphy1 only used for split mode,
     * can be used  concurrently  with dphy2
     * full mode and split mode are mutually exclusive
     */
    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy1_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&gc2053_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy1_out: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&isp0_in>;
            };
        };
    };
};

&csi2_dphy2 {
    status = "okay";

    /*
     * dphy2 only used for split mode,
     * can be used  concurrently  with dphy1
     * full mode and split mode are mutually exclusive
     */
    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy2_in: endpoint@1 {
                reg = <1>;
```

```
                remote-endpoint = <&ov5695_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            dphy2_out: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&mipi_csi2_input>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&dphy2_out>;
                data-lanes = <1 2>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in>;
                data-lanes = <1 2>;
            };
        };
    };
};

&rkcif {
    status = "okay";
};
```

```
&rkcif_mmu {
    status = "okay";
};

&rkcif_mipi_lvds {
    status = "okay";

    /* csi2 link to rkcif, using rkcif to capture stream */
    port {
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
            data-lanes = <1 2>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        mipi_lvds_sditf: endpoint {
            remote-endpoint = <&isp2_in>;
        };
    };
};

&rkcif_dvp {
    status = "okay";

    port {
        dvp_in_bcam: endpoint {
            remote-endpoint = <&sc1330_out>;
            bus-width = <10>;
            vsync-active = <0>;
            hsync-active = <1>;
        };
    };
};

&rkcif_dvp_sditf {
    status = "okay";

    /* parallel endpoint */
    port {
        dvp_sditf: endpoint {
            remote-endpoint = <&isp1_in>;
            bus-width = <10>;
            pclk-sample = <1>;
        };
    };
};

&rkisp {
    status = "okay";
    /* the max input w h and fps of mulit sensor */
```

```
    max-input = <1920 1080 30>;
};

&rkisp_mmu {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp0_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy1_out>;
        };
    };
};

&rkisp_vir1 {
    status = "okay";

    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp1_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dvp_sditf>;
        };
    };
};

&rkisp_vir2 {
    status = "okay";

    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp2_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&mipi_lvds_sditf>;
        };
    };
};
```

RK3588

1. 参考RK3588多sensor支持说明
2. 参考arch/arm64/boot/dts/rockchip/rk3588-evb1-cam-6x.dtsi
3. 对于双isp unite模式

   1）支持双摄最大分辨率7616x2160

   2）支持3/4摄最大分辨率5056x1536

   dts配置居于上述unite模式单摄配置说明，再配置如下isp节点

   &rkisp0_vir1 {

   ```
       status = "okay";
       rockchip,hw = <&rkisp_unite>;
   ```

   //其它省略

   };

   &rkisp0_vir2 {

   ```
       status = "okay";
       rockchip,hw = <&rkisp_unite>;
   ```

   //其它省略

   };

   &rkisp0_vir3 {

   ```
       status = "okay";
       rockchip,hw = <&rkisp_unite>;
   ```

   //其它省略

   };

RV1106

dts参考arch/arm/boot/dts/rv1106-evb-dual-cam.dtsi

硬件isp支持双摄最大分辨率是1080p，如果大于此分辨率则需通过单帧2次回读来处理，缺点消耗更多isp吞吐率，带宽增加，输出帧率偏低。

CIS驱动说明

Camera Sensor采用I2C与主控进行交互，目前sensor driver按照I2C设备驱动方式实现，sensor driver同时采用v4l2 subdev的方式实现与host driver之间的交互。

数据类型简要说明

struct i2c_driver

[说明]

定义i2c 设备驱动信息

[定义]

```
struct i2c_driver {
    ……
    /* Standard driver model interfaces */
    int (*probe)(struct i2c_client *, const struct i2c_device_id *);
    int (*remove)(struct i2c_client *);
    ……
    struct device_driver driver;
    const struct i2c_device_id *id_table;
    ……
};
```

[关键成员]

成员名称	描述
@driver	Device driver model driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	List of I2C devices supported by this driver如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
#if IS_ENABLED(CONFIG_OF)
static const struct of_device_id os04a10_of_match[] = {
    { .compatible = "ovti,os04a10" },
    {},
};
MODULE_DEVICE_TABLE(of, os04a10_of_match);
#endif

static const struct i2c_device_id os04a10_match_id[] = {
    { "ovti,os04a10", 0 },
    { },
};

static struct i2c_driver os04a10_i2c_driver = {
    .driver = {
        .name = OS04A10_NAME,
        .pm = &os04a10_pm_ops,
        .of_match_table = of_match_ptr(os04a10_of_match),
    },
    .probe        = &os04a10_probe,
    .remove        = &os04a10_remove,
    .id_table    = os04a10_match_id,
};

static int __init sensor_mod_init(void)
{
```

```
    return i2c_add_driver(&os04a10_i2c_driver);
}

static void __exit sensor_mod_exit(void)
{
    i2c_del_driver(&os04a10_i2c_driver);
}

device_initcall_sync(sensor_mod_init);
module_exit(sensor_mod_exit);
```

struct v4l2_subdev_ops

[说明]

Define ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_ops {
    const struct v4l2_subdev_core_ops    *core;
    ……
    const struct v4l2_subdev_video_ops    *video;
    ……
    const struct v4l2_subdev_pad_ops    *pad;
};
```

[关键成员]

成员名称	描述
.core	Define core ops callbacks for subdevs
.video	Callbacks used when v4l device was opened in video mode.
.pad	v4l2-subdev pad level operations

[示例]

```
static const struct v4l2_subdev_ops os04a10_subdev_ops = {
    .core    = &os04a10_core_ops,
    .video    = &os04a10_video_ops,
    .pad    = &os04a10_pad_ops,
};
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ......
    int (*s_power)(struct v4l2_subdev *sd, int on);
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
        unsigned long arg);
#endif
    ......
};
```

[关键成员]

成员名称	描述
.s_power	puts subdevice in power saving mode (on == 0) or normal operation mode (on == 1).
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops os04a10_core_ops = {
    .s_power = os04a10_s_power,
    .ioctl = os04a10_ioctl,
#ifdef CONFIG_COMPAT
    .compat_ioctl32 = os04a10_compat_ioctl32,
#endif
};
```

目前使用了如下的私有ioctl实现模组信息的查询和OTP信息的查询设置。

私有ioctl	描述
RKMODULE_GET_MODULE_INFO	获取模组信息，详细参考struct rkmodule_inf;
RKMODULE_AWB_CFG	开关sensor对awb的补偿功能;若模组没有烧录golden awb值，可以在此设置;详细参考struct rkmodule_awb_cfg;
RKMODULE_LSC_CFG	开关sensor对lsc的补偿功能;详细参考struct rkmodule_lsc_cfg;
PREISP_CMD_SET_HDRAE_EXP	Hdr曝光设置详细参考struct preisp_hdrae_exp_s
RKMODULE_SET_HDR_CFG	设置Hdr模式，可实现normal和hdr模式切换，需要驱动适配normal和hdr 2组配置信息详细参考struct rkmodule_hdr_cfg

私有ioctl	描述
RKMODULE_GET_HDR_CFG	获取当前hdr模式详细参考struct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	设置线性模式的conversion gain，如imx347、os04a10 sensor带有conversion gain的功能，如sensor不支持conversion gain，可不实现

struct v4l2_subdev_video_ops

[说明]

Callbacks used when v4l device was opened in video mode.

[定义]

```
struct v4l2_subdev_video_ops {
    ......
    int (*s_stream)(struct v4l2_subdev *sd, int enable);
    ......
    int (*g_frame_interval)(struct v4l2_subdev *sd,
                struct v4l2_subdev_frame_interval *interval);
    int (*g_mbus_config)(struct v4l2_subdev *sd,
                struct v4l2_mbus_config *cfg);
 ......
};
```

[关键成员]

成员名称	描述
.g_frame_interval	callback for VIDIOC_SUBDEV_G_FRAME_INTERVAL ioctl handler code
.s_stream	used to notify the driver that a video stream will start or has stopped
.g_mbus_config	get supported mediabus configurations

[示例]

```
static const struct v4l2_subdev_video_ops os04a10_video_ops = {
    .s_stream = os04a10_s_stream,
    .g_frame_interval = os04a10_g_frame_interval,
    .g_mbus_config = os04a10_g_mbus_config,
};
```

struct v4l2_subdev_pad_ops

[说明]

v4l2-subdev pad level operations

[定义]

```
struct v4l2_subdev_pad_ops {
    ......
```

```
    int (*enum_mbus_code)(struct v4l2_subdev *sd,
                          struct v4l2_subdev_pad_config *cfg,
                          struct v4l2_subdev_mbus_code_enum *code);
    int (*enum_frame_size)(struct v4l2_subdev *sd,
                           struct v4l2_subdev_pad_config *cfg,
                           struct v4l2_subdev_frame_size_enum *fse);
    int (*get_fmt)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_format *format);
    int (*set_fmt)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_format *format);
    int (*enum_frame_interval)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_frame_interval_enum *fie);
    int (*get_selection)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_selection *sel);
    ......
};
```

[关键成员]

成员名称	描述
. enum_mbus_code	callback for VIDIOC_SUBDEV_ENUM_MBUS_CODE ioctl handler code.
. enum_frame_size	callback for VIDIOC_SUBDEV_ENUM_FRAME_SIZE ioctl handler code.
.s_fmt	callback for VIDIOC_SUBDEV_S_FMT ioctl handler code.
.g_fmt	callback for VIDIOC_SUBDEV_G_FMT ioctl handler code
.enum_frame_interval	callback for VIDIOC_SUBDEV_ENUM_FRAME_INTERVAL() ioctl handler code.
.get_selection	callback for VIDIOC_SUBDEV_G_SELECTION() ioctl handler code.

[示例]

```
static const struct v4l2_subdev_pad_ops os04a10_pad_ops = {
    .enum_mbus_code = os04a10_enum_mbus_code,
    .enum_frame_size = os04a10_enum_frame_sizes,
    .enum_frame_interval = os04a10_enum_frame_interval,
    .get_fmt = os04a10_get_fmt,
    .set_fmt = os04a10_set_fmt,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.s_ctrl	actually set the new control value.

[示例]

```
static const struct v4l2_ctrl_ops os04a10_ctrl_ops = {
    .s_ctrl = os04a10_set_ctrl,
};
```

RKISP驱动要求使用框架提供的user controls功能， cameras sensor驱动必须实现如下control功能，参考CIS驱动V4L2-controls列表1

struct xxxx_mode

[说明]

Sensor能支持各个模式的信息。

这个结构体在sensor驱动中常常可以见到，虽然它不是v4l2标准要求的。随着功能的增加，该结构体可根据需求增加变量。

[定义]

```
struct xxxx_mode {
    u32 bus_fmt;
    u32 width;
    u32 height;
    struct v4l2_fract max_fps;
    u32 hts_def;
    u32 vts_def;
    u32 exp_def;
    const struct regval *reg_list;
    u32 hdr_mode;
    u32 vc[PAD_MAX];
};
```

[关键成员]

成员名称	描述

成员名称	描述
.bus_fmt	Sensor输出格式，参考MEDIA_BUS_FMT 表
.width	有效图像宽度，需要和sensor当前配置的width输出一致
.height	有效图像高度，需要和sensor当前配置的height输出一致
.max_fps	图像FPS，denominator/numerator为fps
hts_def	默认HTS，为有效图像宽度 + HBLANK
vts_def	默认VTS，为有效图像高度 + VBLANK
exp_def	默认曝光时间
*reg_list	寄存器列表
.hdr_mode	Sensor工作模式，支持线性模式，两帧合成HDR,三帧合成HDR
.vc[PAD_MAX]	配置MIPI VC通道

[示例]

```
 enum os04a10_max_pad {
    PAD0, /* link to isp */
    PAD1, /* link to csi rawwr0 | hdr x2:L x3:M */
    PAD2, /* link to csi rawwr1 | hdr    x3:L */
    PAD3, /* link to csi rawwr2 | hdr x2:M x3:S */
    PAD_MAX,
};

static const struct os04a10_mode supported_modes[] = {
    {
        .bus_fmt = MEDIA_BUS_FMT_SBGGR12_1X12,
        .width = 2688,
        .height = 1520,
        .max_fps = {
            .numerator = 10000,
            .denominator = 300372,
        },
        .exp_def = 0x0240,
        .hts_def = 0x05c4 * 2,
        .vts_def = 0x0984,
        .reg_list = os04a10_linear12bit_2688x1520_regs,
        .hdr_mode = NO_HDR,
        .vc[PAD0] = V4L2_MBUS_CSI2_CHANNEL_0,
    }, {
        .bus_fmt = MEDIA_BUS_FMT_SBGGR12_1X12,
        .width = 2688,
        .height = 1520,
        .max_fps = {
            .numerator = 10000,
            .denominator = 225000,
        },
        .exp_def = 0x0240,
```

```
        .hts_def = 0x05c4 * 2,
        .vts_def = 0x0658,
        .reg_list = os04a10_hdr12bit_2688x1520_regs,
        .hdr_mode = HDR_X2,
        .vc[PAD0] = V4L2_MBUS_CSI2_CHANNEL_1,
        .vc[PAD1] = V4L2_MBUS_CSI2_CHANNEL_0,//L->csi wr0
        .vc[PAD2] = V4L2_MBUS_CSI2_CHANNEL_1,
        .vc[PAD3] = V4L2_MBUS_CSI2_CHANNEL_1,//M->csi wr2
    },
};
```

struct v4l2_mbus_framefmt

[说明]

frame format on the media bus

[定义]

```
struct v4l2_mbus_framefmt {
    __u32            width;
    __u32            height;
    __u32            code;
    __u32            field;
    __u32            colorspace;
    __u16            ycbcr_enc;
    __u16            quantization;
    __u16            xfer_func;
    __u16            reserved[11];
};
```

[关键成员]

成员名称	描述
width	Frame width
height	Frame height
code	参考MEDIA_BUS_FMT 表
field	V4L2_FIELD_NONE：帧输出方式V4L2_FIELD_INTERLACED：场输出方式

[示例]

struct rkmodule_base_inf

[说明]

模组基本信息，上层用此信息和IQ进行匹配

[定义]

```
struct rkmodule_base_inf {
    char sensor[RKMODULE_NAME_LEN];
    char module[RKMODULE_NAME_LEN];
    char lens[RKMODULE_NAME_LEN];
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
sensor	sensor名，从sensor驱动中获取
module	模组名，从DTS配置中获取，以模组资料为准
lens	镜头名，从DTS配置中获取，以模组资料为准

[示例]

struct rkmodule_fac_inf

[说明]

模组OTP 工厂信息

[定义]

```
struct rkmodule_fac_inf {
    __u32 flag;
    char module[RKMODULE_NAME_LEN];
    char lens[RKMODULE_NAME_LEN];
    __u32 year;
    __u32 month;
    __u32 day;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
flag	该组信息是否有效的标识
module	模组名，从OTP中获取编号，由编号得到模组名
lens	镜头名，从OTP中获取编号，由编号得到镜头名
year	生产年份,如12代表2012年
month	生产月份
day	生产日期

[示例]

struct rkmodule_awb_inf

[说明]

模组OTP awb测定信息

[定义]

```
struct rkmodule_awb_inf {
    __u32 flag;
    __u32 r_value;
    __u32 b_value;
    __u32 gr_value;
    __u32 gb_value;
    __u32 golden_r_value;
    __u32 golden_b_value;
    __u32 golden_gr_value;
    __u32 golden_gb_value;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
flag	该组信息是否有效的标识
r_value	当前模组的AWB R测定信息
b_value	当前模组的AWB B测定信息
gr_value	当前模组的AWB GR测定信息
gb_value	当前模组的AWB GB测定信息
golden_r_value	典型模组的AWB R测定信息，如没有烧录，设为0
golden_b_value	典型模组的AWB B测定信息，如没有烧录，设为0
golden_gr_value	典型模组的AWB GR测定信息，如没有烧录，设为0
golden_gb_value	典型模组的AWB GB测定信息，如没有烧录，设为0

[示例]

struct rkmodule_lsc_inf

[说明]

模组OTP lsc测定信息

[定义]

```
struct rkmodule_lsc_inf {
    __u32 flag;
    __u16 lsc_w;
    __u16 lsc_h;
    __u16 decimal_bits;
    __u16 lsc_r[RKMODULE_LSCDATA_LEN];
    __u16 lsc_b[RKMODULE_LSCDATA_LEN];
    __u16 lsc_gr[RKMODULE_LSCDATA_LEN];
    __u16 lsc_gb[RKMODULE_LSCDATA_LEN];
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
flag	该组信息是否有效的标识
lsc_w	lsc表实际宽度
lsc_h	lsc表实际高度
decimal_bits	lsc 测定信息的小数位数，无法获取的话，设为0
lsc_r	lsc r测定信息
lsc_b	lsc b测定信息
lsc_gr	lsc gr测定信息
lsc_gb	lsc gb测定信息

[示例]

struct rkmodule_af_inf

[说明]

模组OTP af测定信息

[定义]

```
struct rkmodule_af_inf {
    __u32 flag; // 该组信息是否有效的标识
    __u32 vcm_start; // vcm启动电流
    __u32 vcm_end; // vcm终止电流
    __u32 vcm_dir; // vcm测定方向
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
flag	该组信息是否有效的标识
vcm_start	vcm启动电流

成员名称	描述
vcm_end	vcm终止电流
vcm_dir	vcm测定方向

[示例]

struct rkmodule_inf

[说明]

模组信息

[定义]

```
struct rkmodule_inf {
    struct rkmodule_base_inf base;
    struct rkmodule_fac_inf fac;
    struct rkmodule_awb_inf awb;
    struct rkmodule_lsc_inf lsc;
    struct rkmodule_af_inf af;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
base	模组基本信息
fac	模组OTP 工厂信息
awb	模组OTP awb测定信息
lsc	模组OTP lsc测定信息
af	模组OTP af测定信息

[示例]

struct rkmodule_awb_cfg

[说明]

模组OTP awb配置信息

[定义]

```
struct rkmodule_awb_cfg {
    __u32 enable;
    __u32 golden_r_value;
    __u32 golden_b_value;
    __u32 golden_gr_value;
    __u32 golden_gb_value;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
enable	标识awb校正是否启用
golden_r_value	典型模组的AWB R测定信息
golden_b_value	典型模组的AWB B测定信息
golden_gr_value	典型模组的AWB GR测定信息
golden_gb_value	典型模组的AWB GB测定信息

[示例]

struct rkmodule_lsc_cfg

[说明]

模组OTP lsc配置信息

[定义]

```
struct rkmodule_lsc_cfg {
    __u32 enable;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
enable	标识lsc校正是否启用

[示例]

struct rkmodule_hdr_cfg

[说明]

hdr配置信息

[定义]

```
struct rkmodule_hdr_cfg {
    __u32 hdr_mode;
    struct rkmodule_hdr_esp esp;
} __attribute__ ((packed));
struct rkmodule_hdr_esp {
    enum hdr_esp_mode mode;
    union {
        struct {
            __u32 padnum;
            __u32 padpix;
        } lcnt;
        struct {
            __u32 efpix;
```

```
            __u32 obpix;
        } idcd;
    } val;
};
```

[关键成员]

成员名称	描述

hdr_mode	NO_HDR=0 //normal模式
	HDR_X2=5 //hdr 2帧模式
	HDR_X3=6 //hdr 3帧模式

struct rkmodule_hdr_esp	hdr especial mode

enum hdr_esp_mode	HDR_NORMAL_VC=0 //Normal virtual channel mode
	HDR_LINE_CNT=1 //Line counter mode (AR0239)
	HDR_ID_CODE=2 //Identification code mode(IMX327)

[示例]

struct preisp_hdrae_exp_s

[说明]

HDR曝光参数

[定义]

```
struct preisp_hdrae_exp_s {
    unsigned int long_exp_reg;
    unsigned int long_gain_reg;
    unsigned int middle_exp_reg;
    unsigned int middle_gain_reg;
    unsigned int short_exp_reg;
    unsigned int short_gain_reg;
    unsigned int long_exp_val;
    unsigned int long_gain_val;
    unsigned int middle_exp_val;
    unsigned int middle_gain_val;
    unsigned int short_exp_val;
    unsigned int short_gain_val;
    unsigned char long_cg_mode;
    unsigned char middle_cg_mode;
    unsigned char short_cg_mode;
};
```

[关键成员]

成员名称	描述
long_exp_reg	长帧曝光寄存器值
long_gain_reg	长帧增益寄存器值

成员名称	描述
middle_exp_reg	中帧曝光寄存器值
middle_gain_reg;	中帧增益寄存器值
short_exp_reg	短帧曝光寄存器值
short_gain_reg	短帧增益寄存器值
long_cg_mode	长帧conversion gain，0 LCG，1 HCG
middle_cg_mode	中帧conversion gain，0 LCG，1 HCG
short_cg_mode	短帧conversion gain，0 LCG，1 HCG

[说明]

preisp_hdrae_exp_s结构体内只需关注[关键成员]描述的几个参数，曝光、增益值转换成寄存器的公式在iq xml，具体如何转换请查阅iq xml格式说明，conversion gain需要Sensor本身支持这个功能，不支持的话，不需要关注conversion 参数，**HDR2X时，应将传下来的中帧、短帧参数设置进sensor的输出两帧对应的曝光参数寄存器。**

[示例]

struct rkmodule_channel_info

[说明]

channel信息，rk3588新增ioctl的参数

[定义]

```
struct rkmodule_channel_info {
        __u32 index;
        __u32 vc;
        __u32 width;
        __u32 height;
        __u32 bus_fmt;
        __u32 data_type;
        __u32 data_bit;
} __attribute__ ((packed));
```

[关键成员]

成员名称	描述
index	对应id0~id3的pad值
vc	当前video节点使用的vc通道
width	当前设备使用的采集宽度，比如SPD数据采集时，需要通过这边配置采集宽度，未配置将按sensor的输出分辨率采集
height	当前设备使用的采集高度，比如SPD数据采集时，需要通过这边配置采集宽度，未配置将按sensor的输出分辨率采集

成员名称	描述
bus_fmt	配置bus_fmt，主要用于MEDIA_BUS_FMT_EBD_1X8/MEDIA_BUS_FMT_EBD_1X8等特殊格式
data_type	采集的data type，根据输出端配置
data_bit	采集数据的位宽

API简要说明

xxxx_set_fmt

[描述]

设置sensor输出格式。这里指的是一个sensor驱动内支持多种分辨率。通过set_fmt下发分辨率，sensor驱动内遍历去获得最符合的分辨率。

HDR和线性模式的切换不通过这个函数配置，通过ioctl实现私有命令，实现切换。需要注意的是HDR和线性模式的分辨率要一致才能实现切换。

[语法]

```
static int xxxx_set_fmt(struct v4l2_subdev *sd,
                        struct v4l2_subdev_pad_config *cfg,
                        struct v4l2_subdev_format *fmt)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
*cfg	subdev pad information结构体指针	输入
*fmt	Pad-level media bus format结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_get_fmt

[描述]

获取sensor输出格式，获取到的是当前使用的配置，如前面描述的struct xxx_mode里面可能配置多个不同分辨率的配置，驱动初始化的时候会选中其中一个，或者set_fmt切换后，可通过get_fmt来确认当前的format。

对于RK3588以前的芯片，这边还会通过fmt->reserved[0]这个保留参数上传当前设备使用的vc通道信息，也就是可以通过这个参数来指定设备的vc通道，否则按默认值。

对于RK3588及后面的芯片，则通过ioctl实现RKMODULE_GET_CHANNEL_INFO，来获取通道信息。

[语法]

```
static int xxxx_get_fmt(struct v4l2_subdev *sd,
                        struct v4l2_subdev_pad_config *cfg,
                        struct v4l2_subdev_format *fmt)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
*cfg	subdev pad information结构体指针	输入
*fmt	Pad-level media bus format结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

参考MEDIA_BUS_FMT 表

xxxx_enum_mbus_code

[描述]

枚举sensor输出bus format，驱动中会根据struct xxx_mode结构定义一个静态结构体变量，里面填充sensor驱动支持的各个分辨率，这个函数可以根据结构体变量里面填充的bus format，返回枚举值。

[语法]

```
static int xxxx_enum_mbus_code(struct v4l2_subdev *sd,
                               struct v4l2_subdev_pad_config *cfg,
                               struct v4l2_subdev_mbus_code_enum *code)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
*cfg	subdev pad information结构体指针	输入
*code	media bus format enumeration结构体指针	输出

[返回值]

返回值	描述

返回值	描述
0	成功
非0	失败

下表总结了各种图像类型对应的format，参考MEDIA_BUS_FMT 表

xxxx_enum_frame_sizes

[描述]

枚举sensor输出大小。驱动中会根据struct xxx_mode结构定义一个静态结构体变量，里面填充sensor驱动支持的各个分辨率，这个函数可以根据结构体变量里面填充的分辨率，返回枚举值。

[语法]

```
static int xxxx_enum_frame_sizes(struct v4l2_subdev *sd,
                                 struct v4l2_subdev_pad_config *cfg,
                                 struct v4l2_subdev_frame_size_enum *fse)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
*cfg	subdev pad information结构体指针	输入
*fse	media bus frame size结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_g_frame_interval

[描述]

获取sensor输出帧间隔。驱动中会根据struct xxx_mode结构定义一个静态结构体变量，里面的配置的帧率是最高帧率，驱动通过增加vblank的值来降低帧率，如果需要获取实际帧率，这边返回给应用的值，需要将默认的vblank和当前的vblank值进行换算，以获取最新的帧间隔，上传给应用。

[语法]

```
static int xxxx_g_frame_interval(struct v4l2_subdev *sd,
                                 struct v4l2_subdev_frame_interval *fi)
```

[参数]

参数名称	描述	输入输出

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
*fi	pad-level frame rate结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_s_stream

[描述]

设置stream输入输出。

函数一般根据传入的参数on实现start_stream/stop_stream函数。

start_stream里面实现初始化寄存器数组的配置，初始化曝光参数配置，开启数据流。

stop_stream里面实现stream off的寄存器配置

[语法]

```
static int xxxx_s_stream(struct v4l2_subdev *sd, int on)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
on	1: 启动stream输出; 0: 停止stream输出	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_runtime_resume

[描述]

sensor上电时的回调函数。

函数里面主要做sensor上电操作，将上电操作放在这个函数，方便sensor驱动其他地方或者上层驱动调用pm_runtime_get_sync来上电。

[语法]

```
static int xxxx_runtime_resume(struct device *dev)
```

[参数]

参数名称	描述	输入输出
*dev	device结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_runtime_suspend

[描述]

sensor下电时的回调函数。

函数里面做sensor的下电操作，方便sensor驱动其他地方或者上层驱动调用pm_runtime_put来下电。

[语法]

```
static int xxxx_runtime_suspend(struct device *dev)
```

[参数]

参数名称	描述	输入输出
*dev	device结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_set_ctrl

[描述]

设置各个v4l2 control的值。

注册v4l2 control命令时，调用这个回调函数，应用设置control参数下来时，通过这个回调函数来配置sensor。

需要实现的v4l2 control命令可以参考 2.4 参考**struct v4l2_ctrl_ops**说明实现，主要实现以下回调

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2_ctrl结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxx_enum_frame_interval

[描述]

枚举sensor支持的帧间隔参数。

驱动中会根据struct xxx_mode结构定义一个静态结构体变量，可以将里面支持的几种分辨率下的帧间隔枚举上去。这边一般用途不大，对应帧率调整还是通过调整vblank。

[语法]

```
static int xxxx_enum_frame_interval(struct v4l2_subdev *sd,
                    struct v4l2_subdev_pad_config *cfg,
                    struct v4l2_subdev_frame_interval_enum *fie)
```

[参数]

参数名称	描述	输入输出
*sd	子设备实例	输入
*cfg	pad配置参数	输入
*fie	帧间隔参数	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_g_mbus_config

[描述]

获取支持的总线配置，总线类型存在并口/MIPI/LVDS等几种类型，并口里面存在BT601/BT656/BT1120，MIPI存在DPHY/CPHY协议，控制器通过这个接口获取当前sensor使用的总线参数，来确认控制器的采集方式。

比如使用mipi时，当Sensor支持多种MIPI传输模式时，可以根据Sensor当前使用的MIPI模式上传参数。

[语法]

```
static int xxxx_g_mbus_config(struct v4l2_subdev *sd,
                struct v4l2_mbus_config *config)
```

[参数]

参数名称	描述	输入输出
*config	总线配置参数	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_get_selection

[描述]

配置裁剪参数，isp输入的宽度要求16对齐，高度8对齐，对于sensor输出的分辨率不符合对齐或sensor输出分辨率不是标准分辨率，可实现这个函数对输入isp的分辨率做裁剪。

[语法]

```
static int xxxx_get_selection(struct v4l2_subdev *sd,
                struct v4l2_subdev_pad_config *cfg,
                struct v4l2_subdev_selection *sel)
```

[参数]

参数名称	描述	输入输出
*sd	子设备实例	输入
*cfg	pad配置参数	输入
*sel	裁剪参数	输出

[返回值]

返回值	描述

返回值	描述
0	成功
非0	失败

xxxx_ioctl

[描述]

ioctl 私有命令实现。

[语法]

```
static long xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)
```

[参数]

参数名称	描述	输入输出
*sd	子设备实例	输入
cmd	私有命名值	输入
*arg	传递的参数	输入/输出

[返回值]

返回值	描述
0	成功
非0	

驱动移植步骤

1.实现标准I2C子设备驱动部分.

1.1 根据**struct i2c_driver**说明实现以下成员:

struct driver.name

struct driver.pm

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述:
1). CIS 设备资源的获取,主要是解析DTS文件中定义资源, 参考Camera设备注册(DTS);

1.1) RK私有资源定义,命名方式如下rockchip,camera-module-xxx, 该部分资源会由驱动上传给用户态的camera_engine来决定IQ效果参数的匹配;

1.2) CIS设备资源定义,RK相关参考驱动一般包含以下几项:

成员名称	描述
CIS设备工作参考时钟	采用外部独立晶振方案无需获取,RK参考设计一般采用AP输出时钟，该方案需要获取，一般名称为xvclk
CIS设备控制GPIO	例如：Resst引脚,Powerdown引脚
CIS设备控制电源	根据实际硬件设计,获取匹配的软件电源控制资源,例如gpio,regulator

1.3) CIS设备ID号检查, 通过以上步骤获取必要资源后,建议驱动读取设备ID号以便检查硬件的准确性,当然该步骤非必要步骤.

1.4) CIS v4l2设备以及media实体的初始化;

v4l2子设备：v4l2_i2c_subdev_init，RK CIS驱动要求subdev拥有自己的设备节点供用户态rk_aiq访问，通过该设备节点实现曝光控制；

media实体：media_entity_init

2. 参考**struct v4l2_subdev_ops**说明实现v4l2子设备驱动，主要实现以下3个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_subdev_video_ops
struct v4l2_subdev_pad_ops
```

2.1 参考**struct v4l2_subdev_core_ops**说明实现其回调函数，主要实现以下回调：

.s_power

s_power实现sensor的上下电操作，对于一些寄存器数组较长的sensor，可以归纳出公共部分的寄存器，在上电后执行，s_stream只配置不同分辨率配置必须的寄存器，加快切换速度。

.ioctl

.compat_ioctl32

ioctl主要实现的RK私有控制命令，涉及：

成员名称	描述
RKMODULE_GET_MODULE_INFO	DTS文件定义的模组信息(模组名称等)，通过该命令上传给camera_engine，用于设备匹配
RKMODULE_AWB_CFG	模组OTP信息使能情况下，camera_engine通过该命令传递典型模组AWB标定值，CIS驱动负责与当前模组AWB标定值比较后，生成R/B Gain值设置到CIS MWB模块中；
RKMODULE_LSC_CFG	模组OTP信息使能情况下，camera_engine通过该命令控制LSC标定值生效使能；
PREISP_CMD_SET_HDRAE_EXP	HDR曝光设置详细参考struct preisp_hdrae_exp_s

成员名称	描述
RKMODULE_SET_HDR_CFG	设置HDR模式，可实现normal和hdr切换，需要驱动适配hdr和normal 2组配置信息详细参考struct rkmodule_hdr_cfg
RKMODULE_GET_HDR_CFG	获取当前HDR模式详细参考struct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	设置线性模式的conversion gain，如imx347、os04a10 sensor带有conversion gain的功能，高转换的conversion gain可以在低照度下获得更好的信噪比，如sensor不支持conversion gain，可不实现
RKMODULE_SET_QUICK_STREAM	仅配置sensor是stream on/off寄存器，用于异常复位。RK3588支持多摄同步，用于统一开启数据流，详细参考多摄同步机制
RKMODULE_GET_CHANNEL_INFO	RK3588新增获取通道信息，默认vicap的id0~id3对应vc0~vc3，如果有特殊需求可通过这个命令配置通道信息，如SPD/EBD数据采集。3588以前芯片仍通过get_fmt获取通道信息。
RKMODULE_GET_SYNC_MODE、RKMODULE_SET_SYNC_MODE	RK3588新增同步模式获取及配置，详细参考多摄同步机制

2.2 参考**struct v4l2_subdev_video_ops**说明实现其回调函数，主要实现以下回调函数：

成员名称	描述
.s_stream	开关数据流的函数，对于mipi clk是continuous的模式，必须在这个回调函数内开启数据流，若提前开数据流，会识别不到MIPI LP状态
.g_frame_interval	获取帧间隔参数（帧率）
.g_mbus_config	获取总线配置，对于MIPI接口，sensor驱动内若支持不同lane数配置或者支持HDR,通过这个接口返回当前sensor工作模式下的MIPI配置

2.3 参考**struct v4l2_subdev_pad_ops**说明实现其回调函数，主要实现以下回调函数：

成员名称	描述
.enum_mbus_code	枚举当前CIS驱动支持数据格式，实现参考xxxx_enum_mbus_code
.enum_frame_size	枚举当前CIS驱动支持分辨率，实现参考xxxx_enum_frame_sizes
.get_fmt	RKISP driver通过该回调获取CIS输出的数据格式，务必实现；针对Bayer raw sensor、SOC yuv sensor、BW raw sensor输出的数据类型定义参考MEDIA_BUS_FMT 表针对field 输出方式的支持，参考struct v4l2_mbus_framefmt定义； 实现参考xxxx_get_fmt
.set_fmt	设置CIS驱动输出数据格式以及分辨率，务必实现，实现参考xxxx_set_fmt

成员名称	描述
.enum_frame_interval	枚举sensor支持的帧间隔，包含分辨率。实现参考 xxx_enum_frame_interval
.get_selection	配置裁剪参数，isp输入的宽度要求16对齐，高度8对齐。实现参考 xxxx_get_selection

2.4 参考**struct v4l2_ctrl_ops**说明实现，主要实现以下回调

成员名称	描述
.s_ctrl	RKISP driver、camera_engine通过设置不同的命令来实现CIS 曝光控制；

参考CIS驱动V4L2-controls列表1实现各控制ID，其中以下ID属于信息获取类，这部分实现按照standard integer menu controls方式实现；

成员名称	描述
V4L2_CID_LINK_FREQ	参考CIS驱动V4L2-controls列表1中标准定义，目前RKISP driver根据该命令获取MIPI总线频率；
V4L2_CID_PIXEL_RATE	针对MIPI总线：pixel_rate = link_freq * 2 * nr_of_lanes / bits_per_sample
V4L2_CID_HBLANK	参考CIS驱动V4L2-controls列表1中标准定义
V4L2_CID_VBLANK	参考CIS驱动V4L2-controls列表1中标准定义

RK camera_engine会通过以上命令获取必要信息来计算曝光，其中涉及的公式如下：

公式
line_time = HTS / PIXEL_RATE;
PIXEL_RATE = HTS * VTS * FPS
HTS = sensor_width_out + HBLANK;
VTS = sensor_height_out + VBLANK;

其中以下ID属于控制类，RK camera_engine通过该类命令控制CIS

成员名称	描述
V4L2_CID_VBLANK	调整VBLANK，进而调整frame rate、Exposure time max；
V4L2_CID_EXPOSURE	设置曝光时间，单位：曝光行数
V4L2_CID_ANALOGUE_GAIN	设置曝光增益，实际为total gain = analog gain*digital gain; 单位：增益寄存器值
V4L2_CID_HFLIP	设置水平镜像，isp不带镜像功能，如果需要镜像，使用sensor的镜像机制

成员名称	描述
V4L2_CID_VFLIP	设置垂直镜像，isp不带镜像功能，如果需要镜像，使用sensor的镜像机制
V4L2_CID_TEST_PATTERN	实现test pattern，非必须，可用于调试，根据需要实现

2.5 HDR 驱动实现注意事项：

2.5.1 线性模式exposure、gain通过标准v4l2命令实现，当切换成HDR模式时，通过ioctl命令PREISP_CMD_SET_HDRAE_EXP传递参数，标准v4l2命令中的V4L2_CID_EXPOSURE、V4L2_CID_ANALOGUE_GAIN不使用。PREISP_CMD_SET_HDRAE_EXP实现时需要能够缓存参数，在start_stream写完初始化数组后，需要将初始曝光重新写进sensor，覆盖掉初始化数组里面的默认值，这样数据流输出的第一帧亮度才能和曝光参数匹配，才能尽快完成ae收敛。需要注意的是初始化数组里面不能配置开启数据流的寄存器，需要在start_stream返回前配置，同样是为了保证输出的第一帧亮度和初始化曝光参数匹配。

2.5.2 sensor与控制器的HDR/LINEAR交互，通过RKMODULE_SET_HDR_CFG/RKMODULE_GET_HDR_CFG，HDR sensor必须实现，才能正常进行模式切换。

3. CIS 驱动不涉及硬件数据接口信息定义，CIS设备与AP的接口连接关系由DTS设备节点的Port来体现其连接关系，参考 CIS 设备注册(DTS)中关于Port信息的描述。

4. CIS 参考驱动列表

VCM驱动

VCM设备注册(DTS)

RK VCM驱动私有参数说明：

名称	描述
启动电流	VCM 刚好能够推动模组镜头从模组镜头可移动行程最近端(模组远焦)移动，此时VCM driver ic的输出电流值定义为启动电流
额定电流	VCM刚好推动模组镜头至模组镜头可移动行程的最远端(模组近焦)，此时VCM driver ic的输出电流值定义为额定电流
VCM电流输出模式	VCM移动过程中会产生振荡,VCM driver ic电流输出变化需要考虑vcm的振荡周期，以便最大程度减小振荡，输出模式决定了输出电流改变至目标值的时间；

```
vm149c: vm149c@0c { // vcm驱动配置，支持AF时需要有这个设置
    compatible = "silicon touch,vm149c";
    status = "okay";
    reg = <0x0c>;
    rockchip,vcm-start-current = <0>; // 马达的启动电流
    rockchip,vcm-rated-current = <100>; // 马达的额定电流
    rockchip,vcm-step-mode = <4>; // 马达驱动ic的电流输出模式
    rockchip,camera-module-index = <0>; // 模组编号
    rockchip,camera-module-facing = "back"; // 模组朝向，有"back"和"front"
};
```

```
ov13850: ov13850@10 {
    ......
    lens-focus = <&vm149c>; // vcm驱动设置，支持AF时需要有这个设置
    ......
};
```

VCM驱动说明

数据类型简要说明

struct i2c_driver

[说明]

定义i2c 设备驱动信息

[定义]

```
struct i2c_driver {
    ……
    /* Standard driver model interfaces */
    int (*probe)(struct i2c_client *, const struct i2c_device_id *);
    int (*remove)(struct i2c_client *);
    ……
    struct device_driver driver;
    const struct i2c_device_id *id_table;
     ……
};
```

[关键成员]

成员名称	描述
@driver	Device driver model driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	List of I2C devices supported by this driver如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
static const struct i2c_device_id vm149c_id_table[] = {
    { VM149C_NAME, 0 },
    { { 0 } }
};
MODULE_DEVICE_TABLE(i2c, vm149c_id_table);
static const struct of_device_id vm149c_of_table[] = {
    { .compatible = "silicon touch,vm149c" },
    { { 0 } }
```

```
};
MODULE_DEVICE_TABLE(of, vm149c_of_table);
static const struct dev_pm_ops vm149c_pm_ops = {
    SET_SYSTEM_SLEEP_PM_OPS(vm149c_vcm_suspend, vm149c_vcm_resume)
    SET_RUNTIME_PM_OPS(vm149c_vcm_suspend, vm149c_vcm_resume, NULL)
};
static struct i2c_driver vm149c_i2c_driver = {
    .driver = {
        .name = VM149C_NAME,
        .pm = &vm149c_pm_ops,
        .of_match_table = vm149c_of_table,
    },
    .probe = &vm149c_probe,
    .remove = &vm149c_remove,
    .id_table = vm149c_id_table,
};
module_i2c_driver(vm149c_i2c_driver);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
                           unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops vm149c_core_ops = {
    .ioctl = vm149c_ioctl,
#ifdef CONFIG_COMPAT
    .compat_ioctl32 = vm149c_compat_ioctl32
#endif
};
```

目前使用了如下的私有ioctl实现马达移动时间信息的查询。

RK_VIDIOC_VCM_TIMEINFO

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*g_volatile_ctrl)(struct v4l2_ctrl *ctrl);
    int (*try_ctrl)(struct v4l2_ctrl *ctrl);
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.g_volatile_ctrl	Get a new value for this control. Generally only relevantfor volatile (and usually read-only) controls such as a control that returns the current signal strength which changes continuously.
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops vm149c_vcm_ctrl_ops = {
    .g_volatile_ctrl = vm149c_get_ctrl,
    .s_ctrl = vm149c_set_ctrl,
};
```

vm149c_get_ctrl和vm149c_set_ctrl对下面的control进行了支持

V4L2_CID_FOCUS_ABSOLUTE

API简要说明

xxxx_get_ctrl

[描述]

获取马达的移动位置。

[语法]

```
static int xxxx_get_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_set_ctrl

[描述]

设置马达的移动位置。

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_ioctl xxxx_compat_ioctl

[描述]

自定义ioctl的实现函数，主要包含获取马达移动的时间信息，

实现了自定义RK_VIDIOC_COMPAT_VCM_TIMEINFO。

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)
static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

驱动移植步骤

1.实现标准的i2c子设备驱动部分.

1.1 根据**struct i2c_driver**描述，主要实现以下几部分：

struct driver.name

struct driver.pm

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1）VCM设备资源获取，主要获取DTS资源，参考VCM设备注册（DTS）

1.1) RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。

1.2）VCM参数定义，命名方式如rockchip,vcm-xxx，主要涉及硬件参数启动电流、额定电流、移动模式，参数跟马达移动的范围和速度相关。

2） VCM v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，RK VCM驱动要求subdev拥有自己的设备节点供用户态camera_engine访问，通过该设备节点实现调焦控制；

media实体：media_entity_init；

3）RK AF算法将模组镜头整个可移动行程的位置参数定义为[0,64]，模组镜头整个可移动行程在VCM驱动电流上对应的变化范围为[启动电流，额定电流]，该函数中建议实现这2者间的映射换算关系；

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

.ioctl.compat_ioctl32

该回调主要实现RK私有控制命令，涉及：

成员名称	描述
RK_VIDIOC_VCM_TIMEINFO	camera_engine通过该命令获取此次镜头移动所需时间，据此来判断镜头何时停止以及CIS帧曝光时间段是否与镜头移动时间段有重叠；镜头移动时间与镜头移动距离、VCM driver ic电流输出模式相关。

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl和.s_ctrl以标准的v4l2 control实现了以下命令：

成员名称	描述
V4L2_CID_FOCUS_ABSOLUTE	camera_engine通过该命令来设置和获取镜头的绝对位置， RK AF算法中将镜头整个可移动行程的位置参数定义为[0,64]。

FlashLight驱动

FLASHLight设备注册(DTS)

SGM378 DTS 参考：

```
&i2c1 {
    ...
    sgm3784: sgm3784@30 {//闪光灯设备
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "sgmicro,gsm3784";
        reg = <0x30>;
        rockchip,camera-module-index = <0>;//闪光灯对应camera模组编号
        rockchip,camera-module-facing = "back";//闪光灯对应camera模组朝向
        enable-gpio = <&gpio2 RK_PB4 GPIO_ACTIVE_HIGH>;//enable gpio
        strobe-gpio = <&gpio1 RK_PA3 GPIO_ACTIVE_HIGH>;//flash触发gpio
        status = "okay";
        sgm3784_led0: led@0 {//led0设备信息
            reg = <0x0>;//index
            led-max-microamp = <299200>;//torch模式最大电流
            flash-max-microamp = <1122000>;//flash模式最大电流
            flash-max-timeout-us = <1600000>;//falsh最大时间
        };
        sgm3784_led1: led@1 {//led1设备信息
```

```
            reg = <0x1>;//index
            led-max-microamp = <299200>;//torch模式最大电流
            flash-max-microamp = <1122000>;//flash模式最大电流
            flash-max-timeout-us = <1600000>;//falsh最大时间
        };
    };
    ...
    ov13850: ov13850@10 {
        ...
        flash-leds = <&sgm3784_led0 &sgm3784_led1>;//闪光灯设备挂接到camera
        ...
    };
    ...
}
```

GPIO、PWM控制 dts 参考：

```
flash_ir: flash-ir {
        status = "okay";
        compatible = "led,rgb13h";
        label = "pwm-flash-ir";
        led-max-microamp = <20000>;
        flash-max-microamp = <20000>;
        flash-max-timeout-us = <1000000>;
        pwms=<&pwm3 0 25000 0>;
        //enable-gpio = <&gpio0 RK_PA1 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
};
&i2c1 {
    imx415: imx415@1a {
        ...
        flash-leds = <&flash_ir>;
        ...
    }
}
```

注意点：

1、软件上需要根据补光灯类型区分处理流程，如果是红外补光灯，dts 补光灯节点 label需要有ir字样用来识别硬件类型，led补光灯把ir字段去掉即可。

2、对于这种单个引脚控制的硬件电路，有两种情况，一种是固定亮度，直接使用gpio控制。另外一种是亮度可控，使用pwm，通过调节占空比设置亮度，dts pwms 或 enable-gpio，二选一配置。

FLASHLight驱动说明

数据类型简要说明

struct i2c_driver

[说明]

定义i2c 设备驱动信息

[定义]

```
struct i2c_driver {
    ……
    /* Standard driver model interfaces */
    int (*probe)(struct i2c_client *, const struct i2c_device_id *);
    int (*remove)(struct i2c_client *);
    ……
    struct device_driver driver;
    const struct i2c_device_id *id_table;
     ……
};
```

[关键成员]

成员名称	描述
@driver	Device driver model driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	List of I2C devices supported by this driver如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
static const struct i2c_device_id sgm3784_id_table[] = {
    { SGM3784_NAME, 0 },
    { { 0 } }
};
MODULE_DEVICE_TABLE(i2c, sgm3784_id_table);
static const struct of_device_id sgm3784_of_table[] = {
    { .compatible = "sgmicro,sgm3784" },
    { { 0 } }
};
MODULE_DEVICE_TABLE(of, sgm3784_of_table);
static const struct dev_pm_ops sgm3784_pm_ops = {
    SET_RUNTIME_PM_OPS(sgm3784_runtime_suspend, sgm3784_runtime_resume, NULL)
};
static struct i2c_driver sgm3784_i2c_driver = {
    .driver = {
        .name = sgm3784_NAME,
        .pm = &sgm3784_pm_ops,
        .of_match_table = sgm3784_of_table,
    },
    .probe = &sgm3784_probe,
    .remove = &sgm3784_remove,
    .id_table = sgm3784_id_table,
};
module_i2c_driver(vm149c_i2c_driver);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
              unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops sgm3784_core_ops = {
    .ioctl = sgm3784_ioctl,
#ifdef CONFIG_COMPAT
    .compat_ioctl32 = sgm3784_compat_ioctl32
#endif
};
```

目前使用了如下的私有ioctl实现闪光灯点亮时间信息的查询。

RK_VIDIOC_FLASH_TIMEINFO

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*g_volatile_ctrl)(struct v4l2_ctrl *ctrl);
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.g_volatile_ctrl	Get a new value for this control. Generally only relevantfor volatile (and usually read-only) controls such as a control that returns the current signal strength which changes continuously.
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops sgm3784_ctrl_ops[LED_MAX] = {
    [LED0] = {
        .g_volatile_ctrl = sgm3784_led0_get_ctrl,
        .s_ctrl = sgm3784_led0_set_ctrl,
    },
    [LED1] = {
        .g_volatile_ctrl = sgm3784_led1_get_ctrl,
        .s_ctrl = sgm3784_led1_set_ctrl,
    }
};
```

API简要说明

xxxx_set_ctrl

[描述]

设置闪光灯模式、电流和flash timeout时间。

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_get_ctrl

[描述]

获取闪光灯故障状态。

[语法]

```
static int xxxx_get_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_ioctl xxxx_compat_ioctl

[描述]

自定义ioctl的实现函数，主要包含获取闪光灯亮的时间信息，

实现了自定义RK_VIDIOC_COMPAT_FLASH_TIMEINFO。

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)

static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功

返回值	描述
非0	失败

驱动移植步骤

对于普通gpio直接控制led可参考使用kernel/drivers/leds/leds-rgb13h.c和

kernel/Documentation/devicetree/bindings/leds/leds-rgb13h.txt

对于flashlight driver IC可按如下步骤移植

1.实现标准的i2c子设备驱动部分.

1.1 根据**struct i2c_driver**描述，主要实现以下几部分：

struct driver.name

struct driver.pm

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1）flashlight设备资源获取，主要获取DTS资源，参考FLASHLIGHT设备注册(DTS)；

1.1) RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。

2)flash设备名:

对于双led闪光灯，使用led0、led1设备名进行区分。

```
/* NOTE: to distinguish between two led
 * name: led0 meet the main led
 * name: led1 meet the secondary led
 */
snprintf(sd->name, sizeof(sd->name),
         "m%02d_%s_%s_led%d %s",
         flash->module_index, facing,
         SGM3784_NAME, i, dev_name(sd->dev));
```

3)FLASH v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，RK flashlight驱动要求subdev拥有自己的设备节点供用户态camera_engine访问，通过该设备节点实现led控制；

media实体：media_entity_init；

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

.ioctl.compat_ioctl32

该回调主要实现RK私有控制命令，涉及：

成员名称	描述
RK_VIDIOC_FLASH_TIMEINFO	camera_engine通过该命令获取此次led亮的时间，据此来判断CIS帧曝光时间是否在闪光灯亮之后。

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl和.s_ctrl以标准的v4l2 control实现了以下命令：

成员名称	描述
V4L2_CID_FLASH_FAULT	获取闪光灯故障信息
V4L2_CID_FLASH_LED_MODE	设置Led模式 V4L2_FLASH_LED_MODE_NONE V4L2_FLASH_LED_MODE_TORCH V4L2_FLASH_LED_MODE_FLASH
V4L2_CID_FLASH_STROBE	控制闪光灯开
V4L2_CID_FLASH_STROBE_STOP	控制闪光灯关
V4L2_CID_FLASH_TIMEOUT	设置闪光灯模式最大持续亮时间
V4L2_CID_FLASH_INTENSITY	设置闪光灯模式电流
V4L2_CID_FLASH_TORCH_INTENSITY	设置火炬模式电流

FOCUS ZOOM P-IRIS驱动

这里驱动指的是由步进电机控制自动对焦(FOCUS)、变焦(ZOOM)、自动光圈(P-IRIS)。由于使用的步进电机控制方式一样及硬件设计的因素，将三个功能的驱动集成在一个驱动内。根据使用的驱动芯片，如SPI控制的芯片，可以将驱动封装成SPI框架子设备，本章节围绕MP6507、MS41908驱动芯片描述驱动需要实现的数据结构、框架及注意事项。

MP6507设备注册(DTS)

```
mp6507: mp6507 {
    status = "okay";
    compatible = "monolithicpower,mp6507";
    #pwm-cells = <3>;
    pwms = <&pwm6 0 25000 0>,
           <&pwm10 0 25000 0>,
           <&pwm9 0 25000 0>,
              <&pwm8 0 25000 0>;
    pwm-names = "ain1","ain2","bin1","bin2";
    rockchip,camera-module-index = <1>;
    rockchip,camera-module-facing = "front";
    iris_en-gpios  = <&gpio0 RK_PC2 GPIO_ACTIVE_HIGH>;
```

```
    focus_en-gpios  = <&gpio0 RK_PC3 GPIO_ACTIVE_HIGH>;
    zoom_en-gpios  = <&gpio0 RK_PC0 GPIO_ACTIVE_HIGH>;
    iris-step-max = <80>;
    focus-step-max = <7500>;
    zoom-step-max = <7500>;
    iris-start-up-speed = <1200>;
    focus-start-up-speed = <1200>;
    focus-max-speed = <2500>;
    zoom-start-up-speed = <1200>;
    zoom-max-speed = <2500>;
    focus-first-speed-step = <8>;
    zoom-first-speed-step = <8>;
    focus-speed-up-table = < 1176 1181 1188 1196
                             1206 1217 1231 1246
                             1265 1286 1309 1336
                             1365 1396 1429 1464
                             1500 1535 1570 1603
                             1634 1663 1690 1713
                             1734 1753 1768 1782
                             1793 1803 1811 1818>;
    focus-speed-down-table = < 1796 1788 1779 1768
                               1756 1743 1728 1712
                               1694 1674 1653 1630
                               1605 1580 1554 1527
                               1500 1472 1445 1419
                               1394 1369 1346 1325
                               1305 1287 1271 1256
                               1243 1231 1220 1211
                               1203 1195 1189 1184
                               1179 1175>;
    zoom-speed-up-table = < 1198 1205 1212 1220
                            1228 1238 1249 1260
                            1272 1285 1299 1313
                            1328 1343 1359 1375
                            1390 1406 1421 1436
                            1450 1464 1477 1489
                            1500 1511 1521 1529
                            1537 1544 1551>;
    zoom-speed-down-table = < 1547 1540 1531 1522
                              1511 1499 1487 1473
                              1458 1443 1426 1409
                              1392 1375 1357 1340
                              1323 1306 1291 1276
                              1262 1250 1238 1227
                              1218 1209 1202 1195
                              1189 1184 1179 1175
                              1171 1168>;
};

&i2c1 {
    imx334: imx334@1a {
        ...
        lens-focus = <&mp6507>;
        ...
    }
```

```
}

&pwm6 {
    status = "okay";
    pinctrl-names = "active";
    pinctrl-0 = <&pwm6m1_pins_pull_up>;
};

&pwm8 {
    status = "okay";
    pinctrl-names = "active";
    pinctrl-0 = <&pwm8m1_pins_pull_down>;
    center-aligned;
};

&pwm9 {
    status = "okay";
    pinctrl-names = "active";
    pinctrl-0 = <&pwm9m1_pins_pull_down>;
    center-aligned;
};

&pwm10 {
    status = "okay";
    pinctrl-names = "active";
    pinctrl-0 = <&pwm10m1_pins_pull_down>;
};
```

RK私有定义说明：

成员名称	描述
rockchip,camera-module-index	camera序号，和camera匹配的字段
rockchip,camera-module-facing	camera朝向，和camera匹配的字段
iris_en-gpios	IRIS使能GPIO
focus_en-gpios	focus使能GPIO
zoom_en-gpios	zoom使能GPIO
rockchip,iris-step-max	P-IRIS步进电机移动的最大步数
rockchip,focus-step-max	对焦步进电机移动的最大步数
zoom-step-max	变焦步进电机移动的最大步数
iris-start-up-speed	IRIS使用的步进电机的启动速度

成员名称	描述
focus-start-up-speed	focus使用的步进电机的启动速度
focus-max-speed	focus使用的步进电机的最大运行速度
zoom-start-up-speed	zoom使用的步进电机的启动速度
zoom-max-speed	zoom使用的步进电机的最大运行速度
focus-first-speed-step	focus启动速度运行的步数，后续加速区间等比例增加步数，使各个速度段运行的时间尽量接近一致
zoom-first-speed-step	zoom启动速度运行的步数，后续加速区间等比例增加步数，使各个速度段运行的时间尽量接近一致
focus-speed-up-table	focus加速曲线采用查表方式，调整参数生成加速曲线，将生成的梯形加速曲线或 S 型加速曲线的数据表配置进来，不配置或配置单个数据，则直接按启动速度匀速运行；加速曲线最小值不超过马达最大启动速度，最大值不超过步进马达最大运行速度。
focus-speed-down-table	focus减速曲线，减速曲线最大值需小于加速曲线最大值；若加速曲线无效，则减速曲线一样无效，全程按启动速度匀速运行；若没有配置减速曲线，则减速曲线由加速曲线对称得到。
zoom-speed-up-table	zoom加速曲线采用查表方式，调整参数生成加速曲线，将生成的梯形加速曲线或 S 型加速曲线的数据表配置进来，不配置或配置单个数据，则直接按启动速度匀速运行；加速曲线最小值不超过马达最大启动速度，最大值不超过步进马达最大运行速度。
zoom-speed-down-table	zoom减速曲线，减速曲线最大值需小于加速曲线最大值；若加速曲线无效，则减速曲线一样无效，全程按启动速度匀速运行；若没有配置减速曲线，则减速曲线由加速曲线对称得到。

数据类型简要说明

struct platform_driver

[说明]

定义平台设备驱动信息

[定义]

```
struct platform_driver {
        int (*probe)(struct platform_device *);
        int (*remove)(struct platform_device *);
        void (*shutdown)(struct platform_device *);
        int (*suspend)(struct platform_device *, pm_message_t state);
        int (*resume)(struct platform_device *);
        struct device_driver driver;
        const struct platform_device_id *id_table;
        bool prevent_deferred_probe;
};
```

[关键成员]

成员名称	描述
@driver	struct device_driver driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
#if defined(CONFIG_OF)
static const struct of_device_id motor_dev_of_match[] = {
    { .compatible = "monolithicpower,mp6507", },
    {},
};
#endif

static struct platform_driver motor_dev_driver = {
    .driver = {
        .name = DRIVER_NAME,
        .owner = THIS_MODULE,
        .of_match_table = of_match_ptr(motor_dev_of_match),
    },
    .probe = motor_dev_probe,
    .remove = motor_dev_remove,
};
module_platform_driver(motor_dev_driver);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
            unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops motor_core_ops = {
    .ioctl = motor_ioctl,
};
static const struct v4l2_subdev_ops motor_subdev_ops = {
    .core    = &motor_core_ops,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*g_volatile_ctrl)(struct v4l2_ctrl *ctrl);
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.g_volatile_ctrl	Get a new value for this control. Generally only relevantfor volatile (and usually read-only) controls such as a control that returns the current signal strength which changes continuously.

成员名称	描述
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops motor_ctrl_ops = {
    .s_ctrl = motor_s_ctrl,
};
```

API简要说明

xxxx_set_ctrl

[描述]

调用标准v4l2_control设置对焦、变焦、P光圈位置。

实现了以下v4l2标准命令：

成员名称	描述
V4L2_CID_FOCUS_ABSOLUTE	控制对焦，0表示焦距最小，近处清晰
V4L2_CID_ZOOM_ABSOLUTE	控制变焦倍数，0表示放大倍数最小，视场角最大
V4L2_CID_IRIS_ABSOLUTE	控制P光圈开口的大小，0表示光圈关闭

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_get_ctrl

[描述]

调用标准v4l2_control获取对焦、变焦、P光圈当前的位置。

实现了以下v4l2标准命令：

成员名称	描述
V4L2_CID_FOCUS_ABSOLUTE	控制对焦，0表示焦距最小，近处清晰
V4L2_CID_ZOOM_ABSOLUTE	控制变焦倍数，0表示放大倍数最小，视场角最大
V4L2_CID_IRIS_ABSOLUTE	控制P光圈开口的大小，0表示光圈关闭

[语法]

```
static int xxxx_get_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_ioctl xxxx_compat_ioctl

[描述]

自定义ioctl的实现函数，主要包含获取对焦、变焦、P光圈的时间信息（开始移动及结束移动的时间戳），由于使用的镜头没有定位装置，在必要的时候，需要对镜头马达位置进行复位。

实现了自定义：

成员名称	描述
RK_VIDIOC_VCM_TIMEINFO	对焦的时间信息，用来确认当前帧是否为对焦完成后的生效帧
RK_VIDIOC_ZOOM_TIMEINFO	变焦的时间信息，用来确认当前帧是否为变焦完成后的生效帧
RK_VIDIOC_IRIS_TIMEINFO	光圈的时间信息，用来确认当前帧是否为光圈调整后的生效帧
RK_VIDIOC_FOCUS_CORRECTION	对焦位置校正（复位）

成员名称	描述
RK_VIDIOC_ZOOM_CORRECTION	变焦位置校正（复位）
RK_VIDIOC_IRIS_CORRECTION	光圈位置校正（复位）

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)

static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

驱动移植步骤

对于SPI控制的驱动芯片，可以使用SPI框架进行设备驱动移植，RK参考驱动使用MP6507，直接使用pwm输出控制波形，通过MP6507进行功率放大，所以直接platform框架移植。
驱动参考：/kernel/drivers/media/i2c/mp6507.c

移植步骤如下：

1.实现标准的platform子设备驱动部分.

1.1 根据**struct platform_driver**描述，主要实现以下几部分：

struct driver.name

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1） 设备资源获取，主要获取DTS资源，参考FOCUS ZOOM P-IRIS设备注册(DTS);

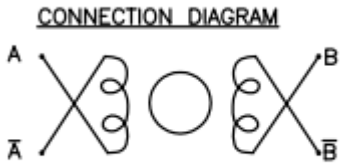
1.1) RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。

1.2) 获取pwm配置，根据马达的控制方式，AB相相差90度，可通过将B相的PWM设置中心对齐实现，在dts pwm节点配置center-aligned，详情见FOCUS ZOOM P-IRIS设备注册(DTS);

SEQUENCE OF EXCITATION

Step / Phase	1	2	3	4
A	+	+	−	−
$\bar{A}$	−	−	+	+
B	−	+	+	−
$\bar{B}$	+	−	−	+

Output Shaft Rotation CW



1.3) 获取使能引脚，MP6507需要使用4个pwm产生步进电机控制波形，由于硬件pwm有限，而对焦、变焦、P光圈三个步进电机各使用一个MP6507驱动器驱动，所以通过gpio来使能对应的MP6507驱动器，从而实现pwm分时复用，当然这也有个弊端，同一时刻只能驱动一个步进电机，其他两个步进电机需等待上一个操作结束才能继续操作;
1.4) 获取各个电机的最大步程、最大启动速度、最大运行速度、加速曲线数据等硬件相关限制条件及资源;

2. hrtimer_init，定时器初始化，pwm使用的是continuous模式，需要定时器定时，达到指定输出pwm波形个数后，进定时器中断关闭pwm，加速过程也需要在运行到指定波形个数后进入定时器中断修改pwm频率，从而实现步进电机的加速;

3）init_completion，通过completion实现同步机制，只有前面一个马达移动操作结束，下一个马达操作才能进行;

4. v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，驱动要求subdev拥有自己的设备节点供用户态rkaiq访问，通过该设备节点实现对马达的控制;

media实体：media_entity_init;

5. 设备名:

```
snprintf(sd->name, sizeof(sd->name), "m%02d_%s_%s",
         motor->module_index, facing,
         DRIVER_NAME);
```

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

```
.ioctl
.compat_ioctl32
```

该回调主要实现RK私有控制命令，涉及：

成员名称	描述

成员名称	描述
RK_VIDIOC_VCM_TIMEINFO	对焦的时间信息，用来确认当前帧是否为对焦完成后的生效帧
RK_VIDIOC_ZOOM_TIMEINFO	变焦的时间信息，用来确认当前帧是否为变焦完成后的生效帧
RK_VIDIOC_IRIS_TIMEINFO	光圈的时间信息，用来确认当前帧是否为光圈调整后的生效帧
RK_VIDIOC_FOCUS_CORRECTION	对焦位置校正（复位）
RK_VIDIOC_ZOOM_CORRECTION	变焦位置校正（复位）
RK_VIDIOC_IRIS_CORRECTION	光圈位置校正（复位）

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.g_volatile_ctrl

.s_ctrl

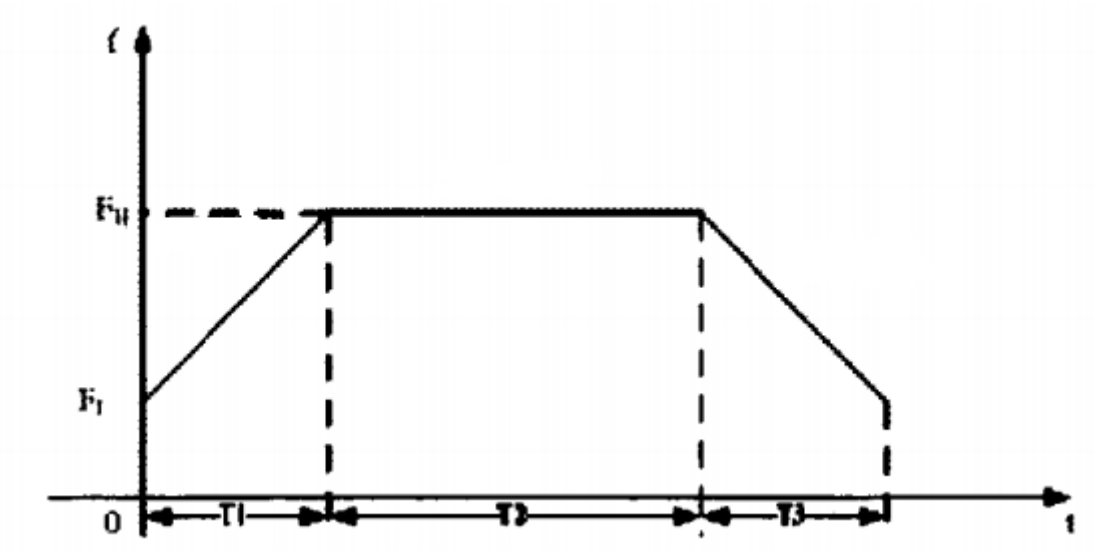
.g_volatile_ctrl和.s_ctrl以标准的v4l2 control实现了以下命令：

参数名称	描述
V4L2_CID_FOCUS_ABSOLUTE	控制对焦，0表示焦距最小，近处清晰
V4L2_CID_ZOOM_ABSOLUTE	控制变焦倍数，0表示放大倍数最小，视场角最大
V4L2_CID_IRIS_ABSOLUTE	控制P光圈开口的大小，0表示光圈关闭

3. 步进电机加速曲线参考:

3.1 梯形曲线

可以简单按图示等间隔等速度进行加速、减速操作。



3.2 S型曲线

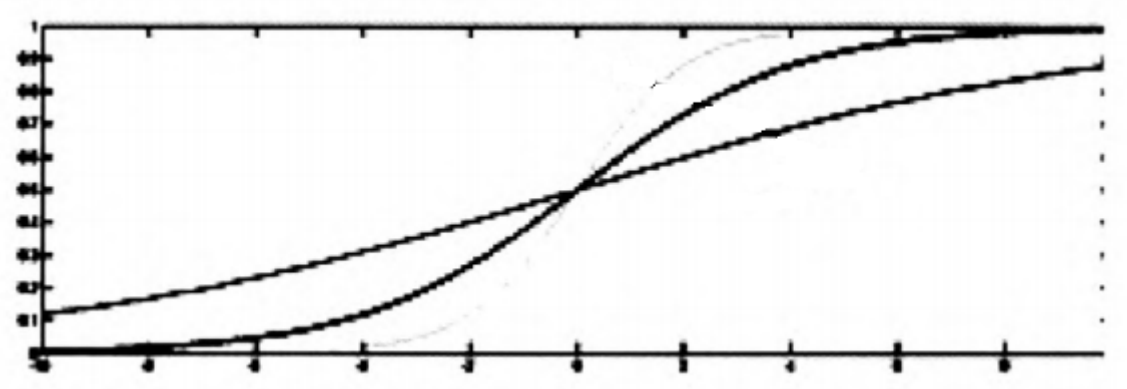
梯形加速如果不理想，可以考虑S型加速，可参考如下公式：

Speed = Vmin + ((Vmax - Vmin) / (1 + exp(-fac * (i - Num) / Num)));

其中，

Vmin指马达启动速度

Vmax指马达目标速度
fac是曲线系数，一般范围在4~6，值越大曲线中间越陡
i是速度分段序号，如分成32段加速，取值0~31
Num是速度分段数的一半，如分成32段，则num为16



MS41908设备注册(DTS)

由于部分镜头支持PIRIS、FOCUS、ZOOM、ZOOM1或DC-IRIS、FOCUS、ZOOM的组合，所以MS41908做成PIRIS、FOCUS、ZOOM、ZOOM1、DC-IRIS功能可配置，可以多次加载驱动，实现多个驱动芯片组合使用，dts会比较复杂，请仔细阅读各个参数的说明。

```
&spi0 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&spi0m0_clk &spi0m0_cs0n &spi0m0_miso &spi0m0_mosi>;
    //如果没有配置，要确认下默认的pinctrl是否为实际使用的pin组
    assigned-clocks = <&pmucru CLK_SPI0>;
    assigned-clock-rates = <100000000>;
    ms41908: ms41908@00 {
        status = "okay";
        compatible = "relmon,ms41908";
        reg = <0>;
        pinctrl-names = "default";
        focus-start-up-speed = <800>;
        zoom-start-up-speed = <800>;
        focus-step-max = <3160>;
        zoom-step-max = <1520>;
        focus-backlash = <18>;
        vd_fz-period-us = <10000>;
        vd_fz-gpios = <&gpio3 RK_PC6 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
        use-focus;
        use-zoom;
        focus-used-pin = "cd";
        zoom-used-pin = "ab";
    };
};
&i2c1 {
    imx335: imx335@1a{
        ...
        lens-focus = <&ms41908>;
        //多个设备注册，lens-focus = <&ms41908_0 &ms41908_1>;
        ...
```

```
    };
};
```

基础定义说明：

成员名称	描述
pinctrl-0	spi的pin定义，按实际使用的pin脚配置，引脚才能映射为spi功能 ag. pinctrl-0 = <&spi0m0_clk &spi0m0_cs0n &spi0m0_miso &spi0m0_mosi>;
assigned-clocks assigned-clock-rates	spi的时钟配置，建议按100MHz配置
reg	reg = <0>;表示使用cs0 reg = <1>;表示使用cs1
rockchip,camera-module-index	camera序号，和camera匹配的字段
rockchip,camera-module-facing	camera朝向，和camera匹配的字段
reset-gpios	ms41908的复位引脚，硬件固定上拉的情况下，可不配置
vd_fz-period-us	步进马达寄存器更新需要的脉冲信号周期，两个步进马达的脉冲信号用同一个，马达运行时间超过vd周期会导致失步，驱动会保证马达单个运动周期时间在vd周期内

FOCUS相关定义说明：

成员名称	描述
use-focus	是否使用focus的功能
focus-used-pin	每个ms41908芯片能驱动两个步进马达，对应的pin组称为”ab“、“cd”，根据实际硬件连接配置
focus-backlash	齿轮间隙产生的误差，马达方向变化时做补偿的步数，根据实际镜头测试得到数据
focus-start-up-speed	步进马达的启动速度，PPS为单位
focus-step-max	马达的有效运动范围，步数为单位
focus-ppw	设置ms41908输出pwm占空比，0-255，值越大驱动能力越强，根据电机负载调整
focus-phmode	设置ms41908输出pwm波形相位校正，一般不配置，视情况而定
focus-micro	设置微步数，分为64、128、256细分，默认256细分

成员名称	描述
focus-reback-distance	focus对焦曲线需要往同一个方向走，位置才会准确。 举例，当前位置为100，想要回到90，需要先回到80，再走到90的位置，位置才是准确的，这边配置的参数就是多回调的步数
focus-1-2phase-excitation	马达激励方式默认是2-2相激励，使用1-2相激励的方式可以配置这个参数 ag. focus-1-2phase-excitation;
focus-dir-opposite	当前马达的运动方向如果和实际的对焦曲线相反，可以配置这个参数实现马达运动方向的反转 ag. focus-dir-opposite;

光耦相关定义说明：

成员名称	描述
focus-pic	光耦的C引脚，用来检测电平变化，电平变化的交界点，就是光耦标记的原点
focus-pia	光耦的A引脚，驱动光电二极管，光耦校正时拉高，镜头正常工作时应拉低，不然光电二极管会影响成像
focus-pie	硬件设计的时候可直接接地，如果设计成gpio控制就需要配置引脚
focus-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些
focus-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些

注：未使用光耦定位的镜头无需配置光耦参数。

ZOOM相关定义说明：

成员名称	描述
use-zoom	是否使用zoom的功能
zoom-used-pin	每个ms41908芯片能驱动两个步进马达，对应的pin组称为”ab“、“cd”，根据实际硬件连接配置
zoom-backlash	齿轮间隙产生的误差，马达方向变化时做补偿的步数，根据实际镜头测试得到数据
zoom-start-up-speed	步进马达的启动速度，PPS为单位
zoom-step-max	马达的有效运动范围，步数为单位
zoom-ppw	设置ms41908输出pwm占空比，0-255，值越大驱动能力越强，根据电机负载调整
zoom-phmode	设置ms41908输出pwm波形相位校正，一般不配置，视情况而定

成员名称	描述
zoom-micro	设置微步数，分为64、128、256细分，默认256细分
zoom-1-2phase-excitation	马达激励方式默认是2-2相激励，使用1-2相激励的方式可以配置这个参数 ag. zoom-1-2phase-excitation;
zoom-dir-opposite	当前马达的运动方向如果和实际的对焦曲线相反，可以配置这个参数实现马达运动方向的反转 ag. zoom-dir-opposite;

光耦相关定义说明：

成员名称	描述
zoom-pic	光耦的C引脚，用来检测电平变化，电平变化的交界点，就是光耦标记的原点 需要注意当前的驱动zoom的光耦A/E脚和focus共用，如果使用的镜头zoom光耦脚是单独的，需新增A/E脚的控制。
zoom-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些
zoom-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些

注：未使用光耦定位的镜头无需配置光耦参数。

ZOOM1相关定义说明：

成员名称	描述
use-zoom1	是否使用zoom1的功能，有些镜头支持控制2个zoom
zoom1-used-pin	每个ms41908芯片能驱动两个步进马达，对应的pin组称为”ab“、“cd”，根据实际硬件连接配置
zoom1-backlash	齿轮间隙产生的误差，马达方向变化时做补偿的步数，根据实际镜头测试得到数据
zoom1-start-up-speed	步进马达的启动速度，PPS为单位
zoom1-step-max	马达的有效运动范围，步数为单位
zoom1-ppw	设置ms41908输出pwm占空比，0-255，值越大驱动能力越强，根据电机负载调整
zoom1-phmode	设置ms41908输出pwm波形相位校正，一般不配置，视情况而定
zoom1-micro	设置微步数，分为64、128、256细分，默认256细分
zoom1-1-2phase-excitation	马达激励方式默认是2-2相激励，使用1-2相激励的方式可以配置这个参数 ag. zoom1-1-2phase-excitation;

成员名称	描述
zoom1-dir-opposite	当前马达的运动方向如果和实际的对焦曲线相反，可以配置这个参数实现马达运动方向的反转 ag. zoom1-dir-opposite;

光耦相关定义说明：

成员名称	描述
zoom1-pic	光耦的C引脚，用来检测电平变化，电平变化的交界点，就是光耦标记的原点
zoom1-pia	光耦的A引脚，驱动光电二极管，光耦校正时拉高，镜头正常工作时应拉低，不然光电二极管会影响成像
zoom1-pie	硬件设计的时候可直接接地，如果设计成gpio控制就需要配置引脚
zoom1-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些
zoom1-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些

注：未使用光耦定位的镜头无需配置光耦参数。

PIRIS相关定义说明：

成员名称	描述
use-p-iris	是否使用P-IRIS的功能
piris-used-pin	每个ms41908芯片能驱动两个步进马达，对应的pin组称为”ab“、“cd”，根据实际硬件连接配置
piris-backlash	齿轮间隙产生的误差，马达方向变化时做补偿的步数，根据实际镜头测试得到数据
piris-start-up-speed	步进马达的启动速度，PPS为单位
piris-step-max	马达的有效运动范围，步数为单位
piris-ppw	设置ms41908输出pwm占空比，0-255，值越大驱动能力越强，根据电机负载调整
piris-phmode	设置ms41908输出pwm波形相位校正，一般不配置，视情况而定
piris-micro	设置微步数，分为64、128、256细分，默认256细分
piris-1-2phase-excitation	马达激励方式默认是2-2相激励，使用1-2相激励的方式可以配置这个参数 ag. piris-1-2phase-excitation;
piris-dir-opposite	当前马达的运动方向如果和实际的对焦曲线相反，可以配置这个参数实现马达运动方向的反转 ag. piris-dir-opposite;

光耦相关定义说明：

成员名称	描述
piris-pic	光耦的C引脚，用来检测电平变化，电平变化的交界点，就是光耦标记的原点
piris-pia	光耦的A引脚，驱动光电二极管，光耦校正时拉高，镜头正常工作时应拉低，不然光电二极管会影响成像
piris-pie	硬件设计的时候可直接接地，如果设计成gpio控制就需要配置引脚
piris-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些
piris-min-pos	光耦校正时没有标定步数，所以光耦原点左侧和右侧的步数需要实测，然后填写到dts，可比实际适当大些

注：未使用光耦定位的镜头无需配置光耦参数。

DCIRIS相关定义说明：

成员名称	描述
use-dc-iris	是否使用DC-IRIS的功能
vd_iris-gpios	DC光圈相关寄存器生效的同步脉冲引脚
dc-iris-reserved-polarity	DC光圈极性设置，如果出现0为光圈全开的情况，可以设置这个属性反转
dc-iris-max-log	DC光圈的目标值范围0~1023，实际使用的有效范围可能比较小，可以配置这个参数用来限制有效范围

数据类型简要说明

struct spi_driver

[说明]

定义平台设备驱动信息

[定义]

```
struct spi_driver {
        int (*probe)(struct spi_device *spi);
        int (*remove)(struct spi_device *spi);
        struct device_driver driver;
        const struct spi_device_id *id_table;
};
```

[关键成员]

成员名称	描述

成员名称	描述
@driver	struct device_driver driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
static const struct spi_device_id motor_match_id[] = {
    {"relmon,ms41908", 0 },
    { }
};

static struct spi_driver motor_dev_driver = {
    .driver = {
        .name = DRIVER_NAME,
        .of_match_table = of_match_ptr(motor_dev_of_match),
    },
    .probe        = &motor_dev_probe,
    .remove        = &motor_dev_remove,
    .id_table    = motor_match_id,
};
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
            unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.

成员名称	描述
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops motor_core_ops = {
    .ioctl = motor_ioctl,
};
static const struct v4l2_subdev_ops motor_subdev_ops = {
    .core    = &motor_core_ops,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*g_volatile_ctrl)(struct v4l2_ctrl *ctrl);
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.g_volatile_ctrl	Get a new value for this control. Generally only relevantfor volatile (and usually read-only) controls such as a control that returns the current signal strength which changes continuously.
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops motor_ctrl_ops = {
    .s_ctrl = motor_s_ctrl,
};
```

API简要说明

xxxx_set_ctrl

[描述]

调用标准v4l2_control设置对焦、变焦、P光圈位置。

MS41908实现v4l2标准命令：

成员名称	描述
V4L2_CID_FOCUS_ABSOLUTE	控制对焦，0表示焦距最小，近处清晰，不回调
V4L2_CID_ZOOM_ABSOLUTE	控制变焦倍数，0表示放大倍数最小，视场角最大，不回调
V4L2_CID_IRIS_ABSOLUTE	控制光圈开口的大小，0表示光圈关闭
V4L2_CID_ZOOM_CONTINUOUS	控制变焦倍数zoom1，多zoom组控制时使用

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

xxxx_ioctl xxxx_compat_ioctl

[描述]

自定义ioctl的实现函数，主要包含获取对焦、变焦、P光圈的时间信息（开始移动及结束移动的时间戳），由于使用的镜头没有定位装置，在必要的时候，需要对镜头马达位置进行复位。

实现了自定义：

成员名称	描述
RK_VIDIOC_VCM_TIMEINFO	对焦的时间信息，用来确认当前帧是否为对焦完成后的生效帧
RK_VIDIOC_ZOOM_TIMEINFO	变焦的时间信息，用来确认当前帧是否为变焦完成后的生效帧
RK_VIDIOC_IRIS_TIMEINFO	光圈的时间信息，用来确认当前帧是否为光圈调整后的生效帧
RK_VIDIOC_ZOOM1_TIMEINFO	多zoom组镜头时，zoom1的时间信息，用来确认当前帧是否为变焦完成后的生效帧
RK_VIDIOC_IRIS_CORRECTION	光圈位置复位，仅作用于P光圈
RK_VIDIOC_FOCUS_CORRECTION	对焦位置复位
RK_VIDIOC_ZOOM_CORRECTION	变焦位置复位

成员名称	描述
RK_VIDIOC_ZOOM1_CORRECTION	双变焦镜头，第二组变焦位置复位
RK_VIDIOC_ZOOM_SET_POSITION	设置跟焦参数，包含对焦、变焦参数，根据变焦曲线，实现多步变焦对焦联动
RK_VIDIOC_FOCUS_SET_POSITION	设置对焦位置

注：

1、为了解决齿轮间隙导致马达的绝对位置不准确的问题，通过固定一个方向为正方向，另一个方向为负方向，初始位置齿轮卡向正方向，当马达往负方向转动时，除了常规要转动的步数外，要多转动大于齿轮间隙步数n，再往正方向转动n步，这样齿轮可以保持往正方向卡，称之为回调。回调保证了绝对位置的准确性。但是回调的步数大于齿轮间隙，在手动对焦，或自动调焦过程，如果分多次往负方向运动，不断回调会导致画面抖动，所以不能每次往负方向转动都回调，故新增RK_VIDIOC_FOCUS_SET_POSITION、RK_VIDIOC_ZOOM_SET_POSITION接口，由af算法决定是否回调。标准v4l2命令V4L2_CID_FOCUS_ABSOLUTE、V4L2_CID_ZOOM_ABSOLUTE不做回调，仅用于手动模式。RK_VIDIOC_ZOOM_SET_POSITION包含对焦和变焦参数，变焦过程中同步调整对焦，使得画面过度较为自然。

2、早期为解决齿轮间隙，通过配置focus-backlash，当往负方向转动时多转齿轮间隙的步数，从而抵消掉齿轮间隙，但因镜头齿轮间隙存在个体差异，不标定有误差，标定工作量又大，所以废弃这个参数。驱动保留设计，若对马达位置准确度要求不高，仍可使用。

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)

static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

驱动移植步骤

对于SPI控制的驱动芯片，可以使用SPI框架进行设备驱动移植，MS41908作为参考。
驱动参考：/kernel/drivers/media/spi/ms41908.c

移植步骤如下：

1.实现标准的spi子设备驱动部分.

1.1 根据**struct spi_driver**描述，主要实现以下几部分：

struct driver.name

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1） 设备资源获取，主要获取DTS资源，参考MS41908设备注册(DTS);

1.1) RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。

1.2) 获取马达相关配置参数，具体根据芯片功能需求定义，尽量做到跟马达运动相关的参数可配置。

2. hrtimer_init，定时器初始化，ms41908是以vd信号作为触发信号，使用定时器来固定每次vd的周期，方便操作，寄存器是在vd信号后生效，每次需要修改寄存器值时，可在vd信号前，提前配置寄存器。需要注意的是寄存器配置的运动速度、运动步数，要在vd周期范围内，时间超过vd周期的步数会丢失。

3）init_completion，通过completion实现同步机制，对于同一个马达，只有前一次操作结束后，才能进行下一次操作。

4. v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，驱动要求subdev拥有自己的设备节点供用户态rkaiq访问，通过该设备节点实现对马达的控制；

media实体：media_entity_init；

5. 设备名:

```
snprintf(sd->name, sizeof(sd->name), "m%02d_%s_%s",
         motor->module_index, facing,
         DRIVER_NAME);
```

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

```
.ioctl
.compat_ioctl32
```

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.g_volatile_ctrl

.s_ctrl

DC-IRIS驱动

DC-IRIS相对于P-IRIS，无法准确知道光圈开口的大小，一般使用场景是默认全开，当曝光调节到最小时，图像还是过曝，则进入光圈调整，当曝光设置到最大，图像还是欠曝，进入光圈调整。DC-IRIS电机是直流电机，通过霍尔器件负反馈缓冲电机的调节速度。对于驱动而言，只要通过一个pwm控制电机转动，当pwm占空比小于20%，光圈会慢慢关闭，直到完全关闭，占空比越小，光圈关闭的速度越快；当占空比大于40%光圈会慢慢打开，占空比越大，打开速度越快；20%~40%区间光圈处于hold住的状态。这边的20%及40%不是定值，与pwm的频率及实际的硬件器件精度有关。
参考驱动： /kernel/drivers/media/i2c/hall-dc-motor.c

DC-IRIS设备注册(DTS)

```
hal_dc_motor: hal_dc_motor{
        status = "okay";
        compatible = "rockchip,hall-dc";
        pwms = <&pwm6 0 2500 0>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
};
&pwm6 {
        status = "okay";
        pinctrl-names = "active";
        pinctrl-0 = <&pwm6m0_pins_pull_down>;
};
&i2c1 {
    imx334: imx334@1a {
        ...
        lens-focus = <&hal_dc_motor>;
        ...
    }
}
```

数据类型简要说明

struct platform_driver

[说明]

定义平台设备驱动信息

[定义]

```
struct platform_driver {
        int (*probe)(struct platform_device *);
        int (*remove)(struct platform_device *);
        void (*shutdown)(struct platform_device *);
        int (*suspend)(struct platform_device *, pm_message_t state);
        int (*resume)(struct platform_device *);
        struct device_driver driver;
        const struct platform_device_id *id_table;
        bool prevent_deferred_probe;
};
```

[关键成员]

成员名称	描述
@driver	struct device_driver driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
#if defined(CONFIG_OF)
static const struct of_device_id motor_dev_of_match[] = {
    { .compatible = "rockchip,hall-dc", },
    {},
};
#endif

static struct platform_driver motor_dev_driver = {
    .driver = {
        .name = DRIVER_NAME,
        .owner = THIS_MODULE,
        .of_match_table = of_match_ptr(motor_dev_of_match),
    },
    .probe = motor_dev_probe,
    .remove = motor_dev_remove,
};
module_platform_driver(motor_dev_driver);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
            unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops motor_core_ops = {
    .ioctl = motor_ioctl,
};
static const struct v4l2_subdev_ops motor_subdev_ops = {
    .core    = &motor_core_ops,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops motor_ctrl_ops = {
    .s_ctrl = motor_s_ctrl,
};
```

API简要说明

xxxx_set_ctrl

[描述]

调用标准v4l2_control光圈位置，DC光圈实际上无法知道光圈的具体位置，这边设置的值是pwm的占空比。

实现了以下v4l2标准命令：

参数名称	描述
V4L2_CID_IRIS_ABSOLUTE	设置控制光圈的pwm的占空比，范围（0~100）

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_ioctl xxxx_compat_ioctl

[描述]

目前无私有定义需要实现，v4l2框架注册需要，实现空函数。

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)

static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

驱动移植步骤

驱动参考：/kernel/drivers/media/i2c/hall-dc-motor.c

移植步骤如下：

1.实现标准的platform子设备驱动部分.

1.1 根据**struct platform_driver**描述，主要实现以下几部分：

struct driver.name

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1） 设备资源获取，主要获取DTS资源，参考DC-IRIS设备注册(DTS);

> 1.1） RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。
>
> 1.2） 获取pwm资源，要注意pwm节点是否有使能。

2. v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，驱动要求subdev拥有自己的设备节点供用户态rkaiq访问，通过该设备节点实现对马达的控制；

media实体：media_entity_init；

3. flash设备名:

```
snprintf(sd->name, sizeof(sd->name), "m%02d_%s_%s",
         motor->module_index, facing,
         DRIVER_NAME);
```

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

```
ioctl
.compat_ioctl32
```

该回调目前不需要实现具体命令，但是作为v4l2子设备必须实现该操作函数，所以这边实现了一个空函数。

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl和.s_ctrl以标准的v4l2 control实现了以下命令：

成员名称	描述
V4L2_CID_IRIS_ABSOLUTE	设置控制光圈的pwm的占空比，范围（0~100）

RK-IRCUT驱动

IRCUT由两根线控制，对这两根线施加3.5v~6v的电源，通过对IRCUT供电电源的正负极对调，且满足通电时间100ms±10%，能够实现IRCUT的切换。驱动通过两个gpio控制电机驱动器的电流输出方向，gpio命令为open（红线）、close（黑线）。电流由open流向close，为红外截止滤光片，白天工作状态；电流由close流向open，为白玻璃片，夜晚工作状态。

RK-IRCUT设备注册(DTS)

```
cam_ircut0: cam_ircut {
        status = "okay";
        compatible = "rockchip,ircut";
        ircut-open-gpios = <&gpio2 RK_PA7 GPIO_ACTIVE_HIGH>;
        ircut-close-gpios  = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
};

&i2c1 {
    imx334: imx334@1a {
        ...
        ir-cut = <&cam_ircut0>;
        ...
    }
}
```

数据类型简要说明

struct platform_driver

[说明]

定义平台设备驱动信息

[定义]

```
struct platform_driver {
        int (*probe)(struct platform_device *);
        int (*remove)(struct platform_device *);
        void (*shutdown)(struct platform_device *);
        int (*suspend)(struct platform_device *, pm_message_t state);
        int (*resume)(struct platform_device *);
        struct device_driver driver;
        const struct platform_device_id *id_table;
        bool prevent_deferred_probe;
};
```

[关键成员]

成员名称	描述
@driver	struct device_driver driver主要包含驱动名称和与DTS注册设备进行匹配的of_match_table。当of_match_table中的compatible域和dts文件的compatible域匹配时，.probe函数才会被调用
@id_table	如果kernel没有使用of_match_table和dts注册设备进行进行匹配，则kernel使用该table进行匹配
@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
#if defined(CONFIG_OF)
static const struct of_device_id ircut_of_match[] = {
    { .compatible = "rockchip,ircut", },
    {},
};
#endif

static struct platform_driver ircut_driver = {
    .driver = {
        .name = RK_IRCUT_NAME,
        .of_match_table = of_match_ptr(ircut_of_match),
    },
    .probe = ircut_probe,
    .remove = ircut_drv_remove,
};

module_platform_driver(ircut_driver);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {
    ……
    long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);
#ifdef CONFIG_COMPAT
    long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,
              unsigned long arg);
#endif
    ……
};
```

[关键成员]

成员名称	描述
.ioctl	called at the end of ioctl() syscall handler at the V4L2 core.used to provide support for private ioctls used on the driver.
.compat_ioctl32	called when a 32 bits application uses a 64 bits Kernel, in order to fix data passed from/to userspace.in order to fix data passed from/to userspace.

[示例]

```
static const struct v4l2_subdev_core_ops ircut_core_ops = {
    .ioctl = ircut_ioctl,
};

static const struct v4l2_subdev_ops ircut_subdev_ops = {
    .core    = &ircut_core_ops,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

```
struct v4l2_ctrl_ops {
    int (*s_ctrl)(struct v4l2_ctrl *ctrl);
};
```

[关键成员]

成员名称	描述
.s_ctrl	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is held when these ops are called, so no one else can access controls owned by that handler.

[示例]

```
static const struct v4l2_ctrl_ops ircut_ctrl_ops = {
    .s_ctrl = ircut_s_ctrl,
};
```

API简要说明

xxxx_set_ctrl

[描述]

调用标准v4l2_control切换IRCUT。

实现了以下v4l2标准命令：

参数名称	描述
V4L2_CID_BAND_STOP_FILTER	0是CLOSE状态，红外光可进入； 3是OPEN状态，红外光不可进入；

[语法]

```
static int xxxx_set_ctrl(struct v4l2_ctrl *ctrl)
```

[参数]

参数名称	描述	输入输出
*ctrl	v4l2 control结构体指针	输入

[返回值]

返回值	描述
0	成功
非0	失败

xxxx_ioctl xxxx_compat_ioctl

[描述]

目前无私有定义需要实现，v4l2框架注册需要，实现空函数。

[语法]

```
static int xxxx_ioctl(struct v4l2_subdev *sd, unsigned int cmd, void *arg)

static long xxxx_compat_ioctl32(struct v4l2_subdev *sd, unsigned int cmd,
unsigned long arg)
```

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev结构体指针	输入
cmd	ioctl命令	输入
*arg/arg	参数指针	输出

[返回值]

返回值	描述
0	成功
非0	失败

驱动移植步骤

驱动参考：/kernel/drivers/media/i2c/rk_ircut.c

移植步骤如下：

1.实现标准的platform子设备驱动部分.

1.1 根据**struct platform_driver**描述，主要实现以下几部分：

struct driver.name

struct driver. of_match_table

probe函数

remove函数

1.2 probe函数实现细节描述：

1） 设备资源获取，主要获取DTS资源，参考RK-IRCUT设备注册(DTS)；

1.1) RK私有资源定义，命名方式如rockchip,camera-module-xxx，主要是提供设备参数和Camera设备进行匹配。

1.2) 获取open、close gpio资源；

2. init_completion，通过completion实现同步机制，由于切换IRCUT需要约100ms，需要completion同步机制来确保上一次IRCUT切换已经完成，才能再次进行操作；

3） 创建工作队列，将切换操作放在work queue，避免长时间阻塞；

4. v4l2设备以及media实体的初始化.

v4l2子设备：v4l2_i2c_subdev_init，驱动要求subdev拥有自己的设备节点供用户态rkaiq访问，通过该设备节点实现对IRCUT的控制；

media实体：media_entity_init；

```
sd->entity.function = MEDIA_ENT_F_LENS;
sd->entity.flags = 1;//flag固定为1，用于区分其他MEDIA_ENT_F_LENS类型的子设备
```

5. 设备名:

```
snprintf(sd->name, sizeof(sd->name), "m%02d_%s_%s",
         ircut->module_index, facing,
         RK_IRCUT_NAME);
```

2.实现v4l2子设备驱动，主要实现以下2个成员：

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 参考**v4l2_subdev_core_ops**说明实现回调函数，主要实现以下回调函数：

```
.ioctl
.compat_ioctl32
```

该回调目前不需要实现私有命令，但是v4l2框架注册有要求，故实现空函数，后续可根据需求补充函数内容。

2.2 参考**v4l2_ctrl_ops**说明实现回调函数，主要实现以下回调函数：

.s_ctrl

.s_ctrl以标准的v4l2 control实现了以下命令：

成员名称	描述
V4L2_CID_BAND_STOP_FILTER	0是CLOSE状态，红外光可进入； 3是OPEN状态，红外光不可进入；

media-ctl v4l2-ctl工具

media-ctl工具的操作是通过/dev/medio0等media 设备，它管理的是Media的拓扑结构中各个节点的format、大小、 链接。

v4l2-ctl工具则是针对/dev/video0，/dev/video1等 video设备，它在video设备上进行set_fmt、reqbuf、qbuf、dqbuf、stream_on、stream_off 等一系列操作。

具体用法可以参考命令的帮助信息，下面是常见的几个使用。

1、打印拓扑结构

```
media-ctl -p -d /dev/media0
```

注: isp2的设备节点较多，可能存在media0/media1/media2节点 ，需要逐个枚举查看设备信息。

2、链接

```
media-ctl -l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[0]'
media-ctl -l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[1]'
```

注： 把ispp的通路断开，链接到main_path，从main_path抓取raw图，media-ctl 没加-d指定设备，默认是/dev/media0设备，需要确认rkisp-isp-subdev挂在在哪个设备节点上，一般是/dev/media1。

3、修改fmt/size

```
media-ctl -d /dev/media0 \
--set-v4l2 '"ov5695 7-0036":0[fmt:SBGGR10_1X10/640x480]'
```

注：需要确认camera设备节点（ov5695 7-0036）挂载在哪个media设备。

4、设置fmt并抓帧

```
v4l2-ctl -d /dev/video0 \
--set-fmt-video=width=720,height=480,pixelformat=NV12 \
--stream-mmap=3 \
--stream-skip=3 \
--stream-to=/tmp/cif.out \
--stream-count=1 \
--stream-poll
```

5、设置曝光、gain等control

```
v4l2-ctl -d /dev/video3 --set-ctrl 'exposure=1216,analogue_gain=10'
```

注： isp驱动会调用camera子设备的control命令，所以指定设备为video3（main_path or self_path）可以设置到曝光，vicap不会去调用camera子设备的control命令，对采集节点直接设置control命令会失败。正确做法是找到camera设备节点是/dev/v4l-subdevX，对终端节点直接配置。

内存优化指南

rv1109/rv1126

MIPI -> DDR_1 -> ISP -> DDR_2 -> ISPP(TNR) -> DDR_3 -> ISPP(NR&Sharp) -> DDR_4 -> ISPP(FEC) -> DDR_5

1、DDR_1：vicap raw数据写到ddr，或者isp mipi raw数据写到ddr，isp再从ddr读取raw数据处理

占用内存：buf_cnt * buf_size * N，(N = 1:线性模式, 2:hdr2帧模式 3: hdr3帧模式)。

buf_size： ALIGN(width * bpp / 8, 256) * height; //bpp为位宽，raw8 raw10或raw12

buf_cnt： 默认4个，定义aiq库代码hwi/isp20/CamHwIsp20.h，最小需要3个。

#define ISP_TX_BUF_NUM 4

#define VIPCAP_TX_BUF_NUM 4

VICAP设备配置**ROCKCHIP_CIF_USE_NONE_DUMMY_BUF**去掉1个内部申请buf。

2、DDR_2：isp fbc yuv420和gain数据写到ddr，ispp再从ddr读取处理

占用内存：buf_size * buf_cnt

buf_size：ALIGN(width, 64) * ALIGN(height, 128) / 16 + ALIGN(width, 16) * ALIGN(hieght, 16) * 1.5625

buf_cnt：tnr 3to1模式4个buf，2to1模式3个buf，模式在iq xml中配置

动静判决开启增加一路缩略图输出，mxn下采样支持4x8和8x8 IQ xml配置选择

buf_size：ALIGN(width / m, 16) * (height / n) * 1.5

buf_cnt：默认6个，aiq使能self_path video配置buf个数

3、DDR_3： ispp tnr fbc yuv420和gain数据写到的ddr，ispp NR&Sharp再从ddr读取处理

占用内存：buf_size * buf_cnt

buf_size：ALIGN(width, 64) * ALIGN(height, 128) / 16 + ALIGN(width, 16) * ALIGN(hieght, 16) * 1.5625

buf_cnt：2个，已最小

4、DDR_4：ispp NR&Sharp yuyv数据写到ddr，ispp fec再从ddr读取处理

占用内存：buf_size * buf_cnt （fec功能不开不占用内存）

buf_size：width * height * 2

buf_cnt：2个，已最小

5、DDR_5：ispp 4路输出图像buffer，根据用户设置分辨率、格式和**buf_cnt**计算buffer大小

上述**buf_cnt**为内存可优化配置的地方

isp cma memory reserved size, can configure more memory and get the actual size after camera app running.

```
isp_reserved: isp {
    compatible = "shared-dma-pool";
    inactive;
    reusable;
    size = <0x10000000>;//256M and need 4M align
};

enable cma debug
+++ b/arch/arm/configs/rv1126_defconfig
@@ -62,6 +62,8 @@ CONFIG_IOSCHED_BFQ=y
 CONFIG_KSM=y
 CONFIG_DEFAULT_MMAP_MIN_ADDR=32768
 CONFIG_CMA=y
+CONFIG_CMA_DEBUG=y
+CONFIG_CMA_DEBUGFS=y

one page is 4K, 26091 page is 104364K and need 4M align, so config 104M to
isp_reserved
[root@RV1126_RV1109:/sys/kernel/debug/cma/cma-isp@0]# ls
alloc  base_pfn  bitmap  count free  maxchunk order_per_bit  used

[root@RV1126_RV1109:/sys/kernel/debug/cma/cma-isp@0]# cat used
26091
```

时延优化指南

rv1109/rv1126

1、配置vicap提前输出

通过dts rkcif_mipi_lvds节点配置wait-line，如图像高1520，配置wait-line=760，即图像采集一半后提前输出buffer给isp。根据isp读取buffer的速度调节wait-line。

```
&rkcif_mipi_lvds {
    wait-line = <760>;
};
```

也可通过配置echo 1000 > /sys/devices/platform/rkcif_mipi_lvds/wait_line调试，支持任意时刻配置。

注：wait-line配置太小，isp过早访问buffer内存，部分数据尚未采集，图像尾端会异常，可能表现为图像拼接，需要根据实际测试选择合适的wait-line。

2、配置isp提前输出

通过dts isp节点配置wait-line，如图像高1520，配置wait-line=760，即图像处理一半后提前输出buffer给后端。根据isp处理时间和ispp处理时间调节wait-line。

```
&rkisp_vir0 {
    wait-line = <760>;
};
```

也可通过配置/sys/module/video_rkisp/parameters/wait_line调试，启动isp（stream/aiq）前配置有效。

注：wait-line配置太小，且ispp处理速度比isp快，由于使用fbc压缩格式，会出现hold住情况。开动静判决和多sensor模式不支持。

3、配置ispp 4路提前输出

通过dts ispp节点配置wait-line，如图像高1520，配置wait-line=896，即图像处理896行后后提前输出buffer给后端。根据ispp处理时间（nr或fec）和后端处理时间调节wait-line。

```
&rkispp_vir0 {
    status = "okay";
    wait-line = <896>;
};
```

也可通过配置/sys/module/video_rkispp/parameters/wait_line调试，启动ispp（stream/aiq）前配置有效。

注：wait-line配置太小，且后端处理速度比ispp快，后端图像处理会异常。多sensor模式不支持。

4、提高硬件处理速度

1）提高isp/ispp clk

drivers/media/platform/rockchip/isp/hw.c

```
static const struct isp_clk_info rv1126_isp_clk_rate[] = {
	{
		.clk_rate = 20,
		.refer_data = 0,
	}, {
		.clk_rate = 600,
		.refer_data = 1920, //width
	}, {
		.clk_rate = 600,
		.refer_data = 2688,
	}, {
		.clk_rate = 600,
		.refer_data = 3072,
	}, {
		.clk_rate = 600,
		.refer_data = 3840,
	}
};
```

drivers/media/platform/rockchip/ispp/hw.c

```
static const struct ispp_clk_info rv1126_ispp_clk_rate[] = {
	{
		.clk_rate = 150,
		.refer_data = 0,
	}, {
		.clk_rate = 500,
		.refer_data = 1920 //width
	}, {
		.clk_rate = 500,
		.refer_data = 2688,
	}, {
		.clk_rate = 500,
		.refer_data = 3072,
	}, {
		.clk_rate = 500,
		.refer_data = 3840,
	}
};
```

2）关闭iommu使用预留内存，预留内存空间根据实际调整。

```
&rkisp_mmu {
	status = "disabled";
};

&rkisp {

	memory-region = <&isp_reserved>;

};
```

```
&rkispp_mmu {
    status = "disabled";
};

&rkispp {
    memory-region = <&isp_reserved>;
};
```

多摄曝光同步功能实现

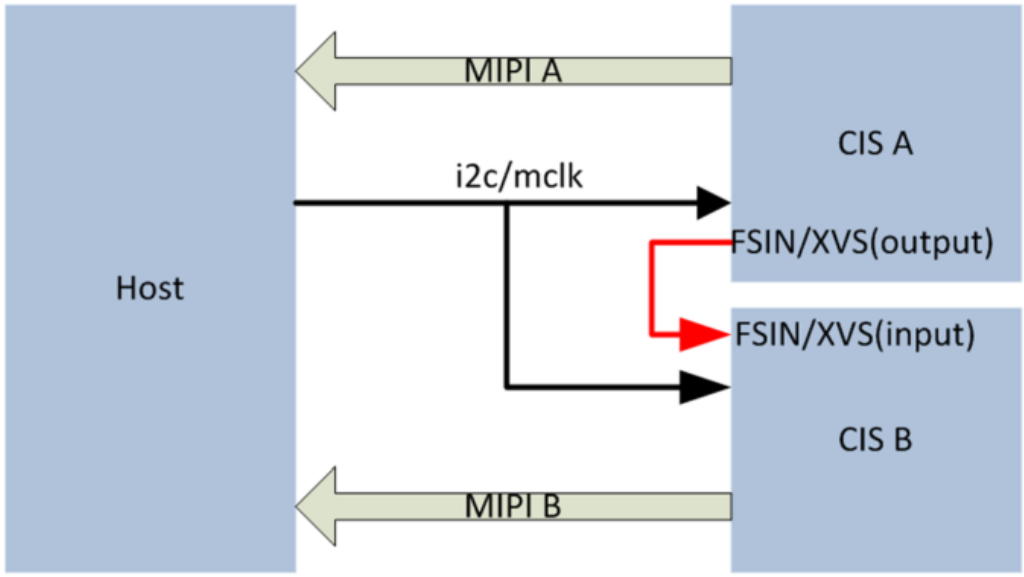
多摄曝光同步功能的实现涉及硬件和软件。

硬件相关：

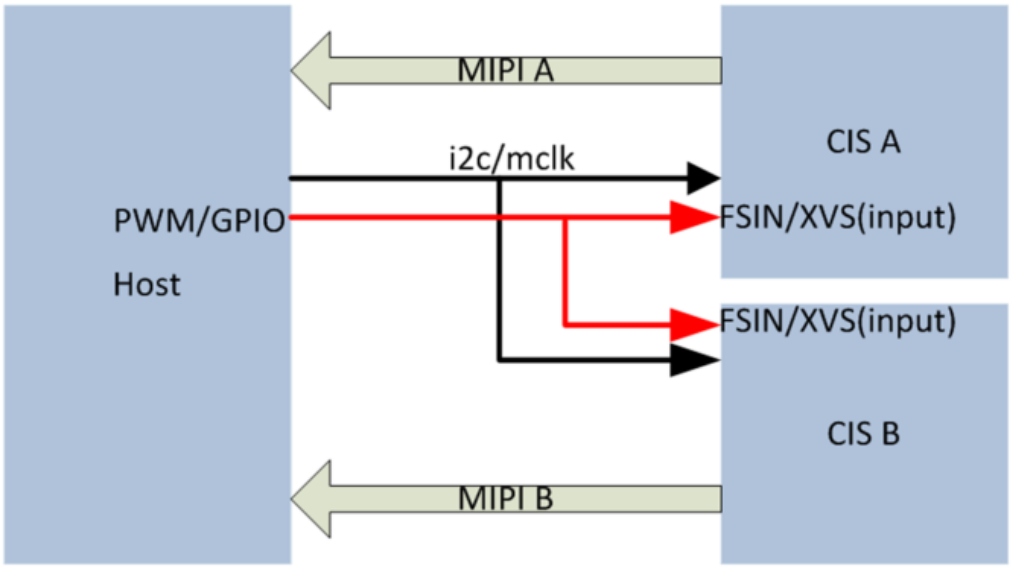
方案索引	方案名称	方案优缺点描述	RK Support
1	**sony:** master-master mode	**优点：** 1. 多摄之间仅需增加FSIN(OV) / XVS(Sony)单个信号的连接 **不足：** 1. Sony CIS采用该方案，接收同步信号CIS(B)相对于输出同步信号CIS(A)曝光起始时刻会延时1个曝光行的时间。一般在百us以内。	RK3588
2	**sony:** master-master(External signal sync) mode	**优点：** 1. Sony CIS采用该方案，CIS(B)相对于CIS(A)曝光起始时刻严格同步 **不足：** 1. 主控与CIS之间通过FSIN(OV) / XVS(Sony) 相连接。主控通过硬件PWM输出曝光场同步信号。PWM驱动与VI驱动需要同步实现	NO
3	**sony:** slave-slave(External signal sync) mode	**优点：** 1. Sony CIS采用该方案，CIS(B)相对于CIS(A)曝光起始时刻严格同步 **不足：** 1. 主控与CIS之间通过XVS， XHS相连接。主控通过硬件PWM输出曝光场同步信号。PWM驱动与VI驱动需要同步实现	NO

硬件方案示意图:

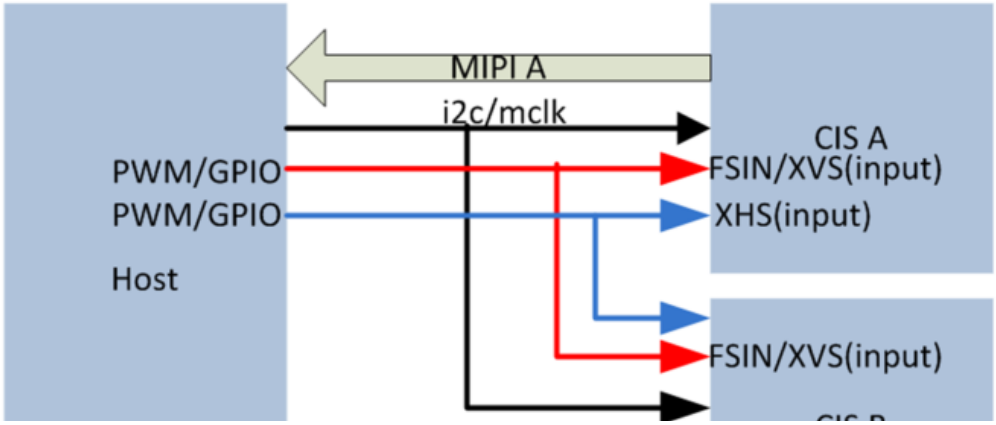
方案 **1**：**(sony:** master-master mode**)**



方案 **2**：**(sony:** master-master(External signal sync) mod



方案 **3**：**(sony:** slave-slave(External signal sync) mode**)**



软件相关

在硬件方案1的软件实现中，用户需要了解CIS驱动定义的曝光同步模式：

模式	FSIN/XVS 引脚	Stream 行为	约束条件
internal mater	输出	1.配置StreamOn寄存器使能之后，CIS 数据接口主动输出帧数据	1. 必须有并且仅有1个
external master	输入	1.配置StreamOn寄存器之后，CIS 数据接口主动输出帧数据 2.无FSIN/XVS 场同步信号：按照寄存器配置帧率，主动持续输出帧数据 有FSIN/XVS 场同步信号： 根据该信号同步曝光，持续输出帧数据	1. 允许多个
slave	输入	1.配置StreamOn寄存器之后，CIS 数据接口主动输出帧数据 2.无FSIN/XVS 场同步信号： 帧数据无输出 有FSIN/XVS 场同步信号： 根据该信号同步曝光，持续输出帧数据	1.允许多个
no sync	无效	1.配置StreamOn寄存器之后，CIS 数据接口主动输出帧数据 2.FSIN/XVS信号不影响数据流	

APP调用流程关键点：

1. App可以通过ioctl接口调用RK私有命令：RKMODULE_GET_SYNC_MODE / RKMODULE_SET_SYNC_MODE 配置所有需要同步的CIS驱动设备的曝光同步模式。同时也可以支持在内核DTS设备树中提前指定各CIS驱动设备的曝光同步模式。
2. 步骤1完成后，调用ioctl接口调用开启CIS设备数据流

CIS驱动实现注意事项：

1. CIS驱动必须实现RKMODULE_GET_SYNC_MODE/RKMODULE_SET_SYNC_MODE来配置曝光同步模式。
2. 由于internal master mode、external master mode 模式下，StreamOn寄存器使能设备会立即出流，为了避免App获取到不同步的数据流，

   2.1 CIS驱动需要在.s_stream回调中，配置CIS初始化Setting以及曝光同步模式对应的Setting，并且检查**仅在NO_SYNC_MODE**才使能StreamOn寄存器。

   2.2 实现RK私有的ioctl命令：**RKMODULE_SET_QUICK_STREAM，在该回调中使能StreamOn寄存器。**

3. 多摄曝光同步模式的参考CIS驱动：drivers/media/i2c/imx464.c

VICAP/ISP特殊采集模式

多通道raw数据拼接

多个raw sensor通过接入rk1608，rk1608打包一个mipi vc通道输出到rk3588主控端，由vicap作为整幅图像采集，vicap虚拟出3个子设备分别链接到3个虚拟的isp节点， rkaiq将buf分别送三个isp节点，isp节点根据偏移量读取raw数据，当三个isp节点都处理完raw数据，buf地址才会返回给vicap继续加入采集队列。

关键问题解答：

1、rk1608数据是如何拼接?

rk1608不做同步检测工作，同步工作要由sensor端做硬件同步，rk1608会检测是否每个摄像头都有数据，都有数据，rk1608的TX才会发送数据。数据排列方式如果下：

第1行： sensor0 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第2行： sensor1 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第3行： sensor2 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第4行： sensor0 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第5行： sensor1 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第6行： sensor2 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
...
第n-2行： sensor0 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第n-1行： sensor1 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|
第n行 ： sensor2 |--- 有效数据 ---|--- 256字节对齐(dummy数据补齐)---|

以上是3个摄像头拼接输入的参考，按行间隔输出。按紧凑存储且做256字节上取整对齐输出，这个是isp对输入数据的要求。rk1608的mipi接口都有对应的index，需要在dts上配置三个摄像头的输入输出，8目MIPI sensor支持的章节中有介绍rk1608节点注册，rk1608 dts上，每个sensor的in_mipi/out_mipi都要准确配置。且每个sensor的配置顺序对应着id0~id2，对应着数据的输出顺序。

2、isp如何拆解数据?

每个isp底下会挂着vicap的虚拟节点，假设使用的是rkcif_mipi_lvds节点，那么对应的虚拟节点分别是rkcif_mipi_lvds_sditf/rkcif_mipi_lvds_sditf_vir1/rkcif_mipi_lvds_sditf_vir2,节点里面需要配置rkckchip，combine-index = <0>来指定每个isp处理的摄像头数据，这边的index对应上个问题的sensor id0~id2。这样每个isp可以根据index按行偏移分别处理每个sensor的数据。

3、曝光控制如何对应到三个sensor?

前面两个问题，说明了数据如何发送，isp如何处理数据。那么曝光控制对应到每个sensor上呢?由于vicap的虚拟节点指定了每个isp节点处理sensor raw数据的顺序，所以直接sensor节点分别挂载到对应的vicap虚拟节点，rkaiq根据pipeline来分发曝光参数到sensor节点。也就是在v4l2 media链路上，原来是：

sensor->dphy->csi2 host->rkcif_mipi_lvds ... rkcif_mipi_lvds_sditf->rkisp

这个特殊模式下：

rk1608->dphy->csi2 host->rkcif_mipi_lvds
... sensor0->rkcif_mipi_lvds_sditf->rkisp0
... sensor1->rkcif_mipi_lvds_sditf_vir1->rkisp1
... sensor2->rkcif_mipi_lvds_sditf_vir2->rkisp2

这样每个rkaiq能够将每个isp节点处理的数据和每个sensor的曝光控制对应起来。

FAQ

如何单独更新驱动版本

通过git生成2个sdk**原生内核版本**驱动补丁，命令如下

新版本sdk内核下，生成补丁

4.19-kernel

git format-patch A..B drivers/media/common drivers/media/v4l2-core drivers/media/platform/rockchip drivers/media/i2c include/uapi/linux/rkisp2-config.h include/uapi/linux/rkisp21-config.h include/uapi/linux/rkispp-config.h include/uapi/linux/rkcif-config.h include/uapi/linux/rk-camera-module.h include/uapi/linux/rk_vcm_head.h include/uapi/linux/videodev2.h include/uapi/linux/media-bus-format.h -o tmp_patch

5.10-kernel

git format-patch A..B drivers/media/common drivers/media/v4l2-core drivers/media/platform/rockchip drivers/media/i2c include/uapi/linux/rkisp2-config.h include/uapi/linux/rkisp21-config.h include/uapi/linux/rkisp3-config.h include/uapi/linux/rkisp32-config.h include/uapi/linux/rkispp-config.h include/uapi/linux/fec-config.h include/uapi/linux/rkcif-config.h include/uapi/linux/rk-video-format.h include/uapi/linux/rk-camera-module.h include/uapi/linux/rk_vcm_head.h include/uapi/linux/videodev2.h include/uapi/linux/media-bus-format.h -o tmp_patch

老版本sdk内核下，打上补丁

git am tmp_patch/*

注：上述A为老版本内核commit id，B为新版本内核commit id，tmp_patch存放补丁目录。

旧版本内核需切到原生状态，否则会因为一些本地修改或已加补丁导致新版本生成补丁无法加到老版本上。

如何获取驱动版本号

从kernel启动log中获取

```
rkisp ffb50000.rkisp: rkisp driver version: v00.01.00
rkispp ffb60000.rkispp: rkispp driver version: v00.01.00
```

由以下命令获取

```
cat /sys/module/video_rkisp/parameters/version
cat /sys/module/video_rkispp/parameters/version
```

如何判断RKISP驱动加载状态

RKISP驱动如果加载成功，会有video及media设备存在于/dev/目录下。系统中可能存在多个/dev/video设备，通过/sys可以查询到 RKISP注册的video节点。

```
localhost ~ # grep '' /sys/class/video4linux/video*/name
```

还可以通过 media-ctl命令，打印拓扑结构查看pipeline是否正常。

判断camera驱动是否加载成功，当所有的camera都注册完毕，kernel会打印出如下的log。

```
localhost ~ # dmesg | grep Async
[ 0.682982] RKISP: Async subdev notifier completed
```

如发现kernel没有Async subdev notifier completed这行log，那么请首先查看sensor是否有相关的报错，I2C通讯是否成功。

如何配置ISP/VICAP RAW存储格式

isp：

rv1109/rv1126:
Three mode:
0: raw12/raw10/raw8 8bit memory compact
1: raw12/raw10 16bit memory one pixel
  big endian
  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
  | 3| 2| 1| 0| -| -| -| -|11|10| 9| 8| 7| 6| 5| 4|
2: raw12/raw10 16bit memory one pixel
  big align
  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
  |11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0| -| -| -| -|

rk356x:
Three mode:
0: raw12/raw10/raw8 8bit memory compact
1: raw12/raw10 16bit memory one pixel
  little align
  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
  | -| -| -| -|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
2: raw12/raw10 16bit memory one pixel
  big align
  |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
  |11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0| -| -| -| -|

控制说明：
  aiq增加对存储模式的控制，启动aiq程序会根据sensor型号配置支持的存储模式；
  rv1109/rv1126： aiq版本v0x1.0x71.1以上版本支持
  rk356x：aiq版本v2.60.01以上版本支持

用户也可通过对流设备执行ioctl来控制存储格式，include/uapi/linux/rkisp2-config.h有相关定义说明：

```
#define RKISP_CMD_GET_CSI_MEMORY_MODE \
    _IOR('V', BASE_VIDIOC_PRIVATE + 100, int)

#define RKISP_CMD_SET_CSI_MEMORY_MODE \
    _IOW('V', BASE_VIDIOC_PRIVATE + 101, int)

enum isp_csi_memory {
    CSI_MEM_COMPACT = 0,
    CSI_MEM_WORD_BIG_END = 1,
    CSI_MEM_WORD_LITTLE_ALIGN = 1,
    CSI_MEM_WORD_BIG_ALIGN = 2,
};
```

vicap：

rv1109/rv1126/rk356x:
Three mode:
0: raw12/raw10/raw8 8bit memory compact
1: raw12/raw10 16bit memory one pixel
low align for rv1126/rv1109/rk356x
|15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
| -| -| -| -|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
2: raw12/raw10 16bit memory one pixel
high align for rv1126/rv1109/rk356x
|15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0| -| -| -| -|

note: rv1109/rv1126/rk356x dvp only support uncompact mode,
and can be set low align or high align

控制说明：
aiq增加对存储模式的控制，启动aiq程序会根据sensor型号配置支持的存储模式；
rv1109/rv1126： aiq版本v0x1.0x71.1以上版本支持
rk356x： aiq版本v2.60.01以上版本支持

用户也可通过对流设备执行`ioctl`来控制存储格式，`include/uapi/linux/rkcif-config.h`有相关定义
说明：

```
#define RKCIF_CMD_GET_CSI_MEMORY_MODE \
    _IOR('V', BASE_VIDIOC_PRIVATE + 0, int)

#define RKCIF_CMD_SET_CSI_MEMORY_MODE \
    _IOW('V', BASE_VIDIOC_PRIVATE + 1, int)

enum cif_csi_lvds_memory {
    CSI_LVDS_MEM_COMPACT = 0,
    CSI_LVDS_MEM_WORD_LOW_ALIGN = 1,
    CSI_LVDS_MEM_WORD_HIGH_ALIGN = 2,
};
```

对于vicap可通过以下命令进行配置，但仅用于调试

配置成非紧凑，每个数字对应一个通道

```
echo 0 0 0 0 > /sys/devices/platform/rkcif_mipi_lvds/compact_test
```

配置成低对齐

```
echo 0 0 0 0 > /sys/devices/platform/rkcif_mipi_lvds/is_high_align
```

配置成高对齐

```
echo 0 0 0 0 > /sys/devices/platform/rkcif_mipi_lvds/is_high_align
```

如何配置VICAP 异常复位

当前vicap驱动存在复位机制，该机制用于当vicap出现异常情况时，对vicap进行cru复位操作。目前主要针对mipi sensor的异常复位，lvds/dvp sensor后续根据情况增加。具体用法如下：
若要启动复位机制，需要在dts上面cif相关的接口设备节点，添加rockchip,cif-monitor参数，若dts不设置该参数，则默认不使能复位机制。

```
&rkcif_mipi_lvds {
    status = "okay";
    /* rockchip,cif-monitor = <index0 index1 index2 index3 index4>; */
    rockchip,cif-monitor = <2 2 5 1000 5>;

    port {
        /* MIPI CSI-2 endpoint */
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
            data-lanes = <1 2 3 4>;
        };
    };
};
```

其中，
index0：用于说明复位的模式，目前主要有四种模式，状态如下表：

模式	说明
Idle	不启动复位模式
Continue	用于实时连续监测vicap是否mipi出错及断流，当发生出错及断流时进行vicap复位；监测定时器是在index1设定的帧数到达时，在帧尾进行初始化，进而开始监测，若未到达相应帧数，则不能触发监测；定时器以index2设定的周期进行采样检测；
Trigger	只在vicap出现csi2 协议层面的mipi报错时触发复位。通过index4来设定mipi 报错的次数，当达到index4的次数时，进而在帧尾中断触发监测定时器的初始化，在index2参数设定的周期到达后，实现一次vicap的复位；

模式	说明
Hotplug	热插拔模式，主要是针对类似n4/tp6188这类车机转接芯片实现的，用于解决插拔时图像割裂或者断流的问题；该模式具有continue模式的功能，即实时连续监测vicap是否mipi出错及断流，当发生出错及断流时进行vicap复位；监测定时器是在index1设定的帧数到达时，在帧尾进行初始化，进而开始监测，若未到达相应帧数，则不能触发监测；定时器以index2设定的周期进行采样检测；与continue的差别在于，在mipi 不报错和不断流的情况下，若是vicap所采集的sensor通过RKMODULE_SET_VICAP_RST_INFO命令置位复位使能，那么vicap在通过RKMODULE_GET_VICAP_RST_INFO获取到该信息后会触发复位操作。

index1：对continue或者hotplug而言，在采集到index1帧数据后，触发监测定时器；

Index2：监测定时器的周期，以一帧为单位，监测周期为index2帧；

Index3：延时复位的时间参数，在发现vicap csi2 报错后，在该定义时间内，持续对监测，当检测到错误不再增加，进行复位，超过该定义时间，不管是否还在增加错误，都立即进行复位操作，时间单位ms；

Index4：用于设定mipi csi err的出现次数，在达到该次数后，触发复位；

如何抓取CIS输出的RAW、YUV数据

驱动开发完成后，可以通过标准的v4l2-ctl命令直接操作驱动来获取CIS的输出数据。v4l2-ctl使用帮助可以参考：https://www.mankier.com/1/v4l2-ctl

示例：

```
v4l2-ctl -d /dev/video0  --set-fmt-video=width=1920,height=1080,pixelformat=RG10
--stream-mmap=4 --stream-count=1 --stream-to=/tmp/cap.raw  --stream-skip=2
```

-d：指定设备名称

--set-fmt-video：设置分辨率，需和sensor输出分辨率一致，sensor当前分辨率可通过media-ctl -p -d /dev/mediaX查看。

pixelformat：输出数据格式

--stream-mmap：mmap buffer数量。

--stream-count：抓取的帧数，多帧也是存在同一文件。

--stream-to：指定存储路径。

--stream-skip：跳掉的帧数。

设备支持情况列表

RV1109/RV1126

设备	输入接口	输入数据格式	设备节点名称	输出Raw	输出YUV
VICAP	DVP	RAW	video0~video3	非紧凑型Raw	no

设备	输入接口	输入数据格式	设备节点名称	输出Raw	输出YUV
VICAP	MIPI/LVDS	RAW	video0~video3	非紧凑型Raw 紧凑型Raw	no
VICAP	DVP / MIPI / LVDS	YUV	video0~video3	no	nv12 nv16
ISP	DVP / MIPI / LVDS	RAW	rkisp_rawwr0 rkisp_rawwr1 rkisp_rawwr2 rkisp_rawwr3	非紧凑型Raw 紧凑型Raw	no
ISP	MIPI / LVDS	YUV	rkisp_mainpath	非紧凑型Raw	nv12 nv16
ISPP	Read ddr only	YUV	rkispp_m_bypass rkispp_scale0 rkispp_scale1 rkispp_scale2	no	nv12 nv16

RK356X

设备	输入接口	输入数据格式	设备节点及名称	输出Raw	输出YUV
VICAP	DVP	RAW	video0~video3	非紧凑型Raw	no
VICAP	MIPI/LVDS	RAW	video0~video3	非紧凑型Raw 紧凑型Raw	no
VICAP	DVP / MIPI / LVDS	YUV	video0~video3	no	nv12 nv16
ISP	DVP / MIPI / LVDS	RAW	rkisp_rawwr0 rkisp_rawwr1 rkisp_rawwr2 rkisp_rawwr3	非紧凑型Raw 紧凑型Raw	no
ISP	MIPI / LVDS	YUV	rkisp_mainpath	非紧凑型Raw	nv12 nv16

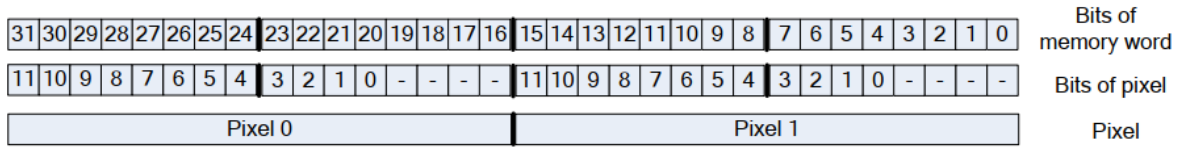
注:

1. 设备节点名称查询命令: media-ctl -p -d /dev/mediaX (其中X指0,1,2,3...)

Raw数据存储格式

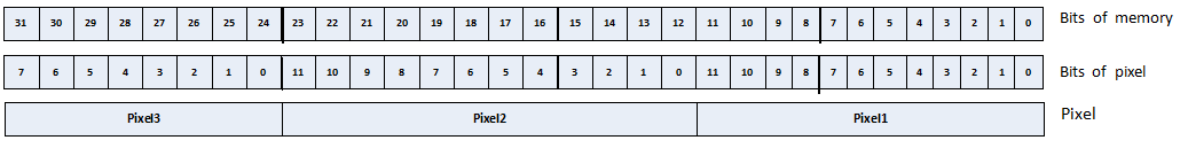
非紧凑型存储格式RAW

非紧凑型是指将sensor输出的raw10、raw12数据按16bit存储，高位对齐。对于raw12数据在内存中的存储排列方式，以4字节的内存片段为例，数据的存储方式如下所示：



紧凑型存储格式RAW

对于raw12数据在内存中的存储排列方式，以4字节的内存片段为例，数据的存储方式如下所示：



重要提醒：

ISP mainpath 设备，输入数据是Raw10 、Raw12时，统一是输出Raw12的非紧凑型存储格式RAW

参考用例：

VICAP输出Raw

1、默认紧凑型，通过如下命令可以实现紧凑与非紧凑格式的切换：

```
    echo 0 > /sys/devices/platform/rkcif_mipi_lvds/compact_test
```

其中，0表示非紧凑型，1表示紧凑型；对于同时使用多通道的设备或使用的通道不是vc0，命令可修改为

```
    echo 0 0 0 0 > /sys/devices/platform/rkcif_mipi_lvds/compact_test
```

其中，echo后面的数字依次对应vc0、vc1、vc2、vc3通道的数据存储类型。

2、 video0~3对应vc0~vc3。

3、执行v4l2-ctl

```
v4l2-ctl -d /dev/video0  --set-fmt-video=width=1920,height=1080,pixelformat=RG10
--stream-mmap=4 --stream-count=1 --stream-to=/tmp/cap.raw  --stream-skip=2
```

ISP maipath输出非紧凑Raw

1、需抓取mainpath的图像，isp默认输出链接是rkisp-bridge-ispp，需按如下命令切到mainpath：

```
    media-ctl -l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[0]'

    media-ctl -l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[1]'
```

注：没有使用-d表示默认使用media0节点，若rkisp-isp-subdev不在media0，需-d指定到所在media节点。

0表示pad0，sink，详细说明自行查阅v4l2相关文档。

2、isp output format默认是YUYV8_2X8，使用如下命令切换到bayer raw格式：

```
    media-ctl -d /dev/media0 --set-v4l2 '"rkisp-isp-
subdev":2[fmt:SBGGR12_1X12/2688x1520]'
```

注：rkisp-isp-subdev节点不一定在media0，-d指定设备，需要确认rkisp-isp-subdev在哪个media节点。

2表示pad2，source，详细说明自行查阅v4l2相关文档。

修改后必须media-ctl -p -d /dev/mediaX查看是否修改生效，生效后抓取的raw才是原始的raw数据。

3、执行v4l2-ctl

```
v4l2-ctl -d /dev/video0  --set-fmt-video=width=1920,height=1080,pixelformat=RG10
--stream-mmap=4 --stream-count=1 --stream-to=/tmp/cap.raw  --stream-skip=2
```

VICAP输出YUV：

只支持输入端数据为YUV格式，若输入端为RAW格式，vicap无法输出YUV格式。

```
v4l2-ctl -d /dev/video0  --set-fmt-video=width=1920,height=1080,pixelformat=NV12
--stream-mmap=4 --stream-count=1 --stream-to=/tmp/cap.raw  --stream-skip=2
```

ISP输出YUV：

```
v4l2-ctl -d /dev/video5  --set-fmt-video=width=1920,height=1080,pixelformat=NV12
--stream-mmap=4 --stream-count=1 --stream-to=/tmp/cap.raw  --stream-skip=2
```

注:

```
1. 对于isp，可抓取mainpath或selfpath，video5只是举例，请按实际参数设置。
2. ISP 输入数据是Raw情况下， ISP能够将Raw数据转换成YUV数据， 同时包含了各项图像处理操作， 此类
图像处理操作需要RK AIQ来控制ISP的各项图像处理模块,当前命令只是数据流部分,图像处理模块参数采用驱
动默认值,图像效果一般处于一个异常状态.
```

ISPP输出YUV：

ispp输入数据来源rkisp_mainpath、rkisp_selfpath和rkispp_input_image link关闭，rkisp-bridge-ispp link开启，rkisp-isp-subdev pad2: Source 格式必须是fmt:YUYV8_2X8，默认状态不需要配置link，参考命令如下，

```
media-ctl -l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[0]'

media-ctl -l '"rkisp-isp-subdev":2->"rkisp_selfpath":0[0]'

media-ctl -l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[1]'

media-ctl -d /dev/media1 -l '"rkispp_input_image":0->"rkispp-subdev":0[1]'

v4l2-ctl -d /dev/video13 \

--set-fmt-video=width=2688,height=1520,pixelformat=NV12 \

--stream-mmap=3 --stream-to=/tmp/nv12.out --stream-count=20 --stram-poll
```

注：-d 设备名称可根据抓图需求，选择以下几个节点，对应的节点名称通过media-ctl -p -d /dev/mediaX查看。

rkispp_m_bypass	Full resolution and yuv format
rkispp_scale0	Full or scale resolution and yuv formatScale range:[1 8] ratio, 3264 max width
rkispp_scale1	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width
rkispp_scale2	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width

如何切换CIS驱动输出分辨率

1、对于sensor驱动支持多个分辨率的驱动，需要抓取另外一个分辨率的raw数据时，可通过如下命令切换sensor当前使用的分辨率：

```
media-ctl -d /dev/media0 --set-v4l2 '"m01_f_os04a10 1-0036-
1":0[fmt:SBGGR12_1X12/2688x1520]'
```

注：m01_f_os04a10 1-0036-1是sensor节点的名称，后面跟需要的format，前提是sensor驱动内支持这个format配置。

2、对于vicap，只要设置sensor节点，对于isp还需要设置isp的输入输出格式，参考命令如下：

```
media-ctl -d /dev/media0 --set-v4l2 '"rkisp-isp-
subdev":0[fmt:SBGGR12_1X12/2688x1520]'
media-ctl -d /dev/media0 --set-v4l2 '"rkisp-isp-subdev":0[crop:(0,0)/2688x1520]'
media-ctl -d /dev/media0 --set-v4l2 '"rkisp-isp-
subdev":2[fmt:SBGGR12_1X12/2688x1520]'
media-ctl -d /dev/media0 --set-v4l2 '"rkisp-isp-subdev":2[crop:(0,0)/2688x1520]'
```

如何设置CIS的曝光参数

1、通过media-ctl -p -d /dev/mediaX找到sensor节点名称，节点名称格式为/dev/v4l-subdevX，参考命令如下：

```
v4l2-ctl -d /dev/v4l-subdev4 --set-ctrl 'exposure=1216,analogue_gain=10'
```

也可分开设置：

```
v4l2-ctl -d /dev/v4l-subdev4 --set-ctrl exposure=1216
v4l2-ctl -d /dev/v4l-subdev4 --set-ctrl analogue_gain=10
```

2、exposure最大值被sensor vts限制住，最大限制条件可能是vts-4或vts-10，不同sensor根据sensor手册的说明做限制。假设当前帧率为30fps，最大曝光时间为33.3ms，要设置40ms的曝光，就得加大vts才能设置40ms的曝光，可以等比例换算，vts_30fps * 30fps = vts_25fps * 25fps，从而换算出25fps对应的vts，（vts - height）为vblank，将换算后vblank设置到sensor驱动即可设置更大的曝光，命令参考如下：

```
v4l2-ctl -d /dev/v4l-subdev4 --set-ctrl vertical_blanking=200
```

如何支持黑白摄像头

CIS驱动需要将黑白sensor的输出format改为如下三种format之一，

```
MEDIA_BUS_FMT_Y8_1X8 （sensor 8bit输出）

MEDIA_BUS_FMT_Y10_1X10 （sensor 10bit输出）

MEDIA_BUS_FMT_Y12_1X12 （sensor 12bit输出）
```

即在函数xxxx_get_fmt和xxxx_enum_mbus_code返回上述format。

RKISP驱动会对这三种format进行特别设置，以支持获取黑白图像。

另外，如应用层需要获取Y8格式的图像，则只能使用SP Path，因为只有SP Path可以支持Y8格式输出。

如何支持奇偶场合成

RKISP 驱动支持奇偶场合成功能，限制要求：

1. MIPI接口： 支持输出frame count number (from frame start and frame end short packets)，RKISP驱动以此来判断当前场的奇偶；
2. BT656接口：支持输出标准SAV/EAV，即bit6为有奇场偶场标记信息，RKISP驱动以此来判断当前场的奇偶；
3. RKISP驱动中RKISP1_selfpath video设备节点具备该功能，其他video设备节点不具备该功能，app层误调用其他设备节点的话，驱动提示以下错误信息：

“only selfpath support interlaced”

RKISP_selfpath信息可以media-ctl -p查看:

```
entity 3: rkisp_selfpath (1 pad, 1 link)
     type Node subtype V4L flags 0
      device node name /dev/video1
    pad0: Sink
       <- "rkisp-isp-subdev":2 [ENABLED]
```

设备驱动实现方式如下：

设备驱动format.field需要设置为V4L2_FIELD_INTERLACED，表示此当前设备输出格式为奇偶场，即在函数xxxx_get_fmt返回format.field格式。可参考driver/media/i2c/tc35874x.c驱动；

如何查看debug信息

1、查看media pipeline信息，此对应dts camera配置

```
find /dev -name "media*" | xargs -i media-ctl -p -d {}
```

2、查看proc信息，此为vicap/isp/ispp当前状态和帧输入输出信息，可以多cat几次

```
cat /proc/rk*
```

3、查看驱动debug信息，设置debug level到isp和ispp节点，level数值越大信息越多

```
echo n > /sys/module/video_rkisp/parameters/debug (n = 0, 1, 2, 3, 4; 0为关)
echo n > /sys/module/video_rkispp/parameters/debug
```

4、查看寄存器信息，把isp.reg和ispp.reg pull出来

RV1109/RV1126

```
io -4 -l 0x10000 0xffb50000 > /tmp/isp.reg
io -4 -l 0x1000 0xffb60000 > /tmp/ispp.reg
```

RK3566/RK3568

```
io -4 -l 0x10000 0xfdff0000> /tmp/isp.reg
```

RK3588

```
io -4 -l 0x10000 0xfdcb0000 > /tmp/isp0.reg
io -4 -l 0x10000 0xfdcc0000 > /tmp/isp1.reg
```

RV1106

```
io -4 -l 0x10000 0xffa00000> /tmp/isp.reg
```

5、提供debug信息步骤

1. 有问题现场 1->2->4->3
2. 复现问题 3->启动->复现->1->2->4

6、isp/ispp/vicap proc信息说明

```
[root@RV1126_RV1109:/]# cat /proc/rkisp*
rkisp-vir0 Version:v01.09.00
clk_isp    400000000
aclk_isp   500000000
hclk_isp   250000000
Interrupt  Cnt:6195 ErrCnt:0
Input      rkcif_mipi_lvds Format:SBGGR10_1X10 Size:2688x1520@30fps Offset(0,0)
Isp Read   mode:frame1 (frame:4061 rate:66ms idle time:10ms frameloss:6077)
cnt(total:2026 X1:1969 X2:56 X3:-1)
           hw link:1 idle:1 vir(mode:0 index:0)
Output     rkispp0 Format:FBC420 Size:2688x1520 (frame:4061 rate:66ms
frameloss:2018)
Output     rkisp_selfpath Format:FBCG Size:672x190 Dcrop(0,0|2688x1520)
(frame:4061 rate:66ms delay:28ms frameloss:2017)
DPCC0      ON(0x5)
DPCC1      ON(0x5)
DPCC2      ON(0x5)
BLS        ON(0x1)
SDG        OFF(0x80446197)
LSC        ON(0x1)
AWBGAIN    ON(0x80446197) (gain: 0x01110111, 0x024d0219)
DEBAYER    ON(0x7000111)
CCM        ON(0x80000001)
GAMMA_OUT  ON(0x80000001)
CPROC      ON(0xf)
IE         OFF(0x0) (effect: BLACKWHITE)
WDR        OFF(0x30cf0)
HDRTMO     ON(0xa4f05a25)
HDRMGE     OFF(0x0)
RAWNR      ON(0x80100001)
GIC        ON(0x80000001)
DHAZ       ON(0x80101019)
3DLUT      OFF(0x2)
GAIN       ON(0x10110)
LDCH       OFF(0x0)
CSM        FULL(0x80446197)
SIAF       OFF(0x0)
SIAWB      OFF(0x0)
YUVAE      ON(0x400100f3)
SIHST      ON(0x38000107)
RAWAF      ON(0x7)
RAWAWB     ON(0x776887)
RAWAE0     ON(0x40000003)
RAWAE1     ON(0x400000f5)
RAWAE2     ON(0x400000f5)
RAWAE3     ON(0x400000f5)
RAWHIST0   ON(0x40000501)
RAWHIST1   ON(0x60000501)
RAWHIST2   ON(0x60000501)
RAWHIST3   ON(0x60000501)
Monitor    OFF Cnt:0
```

clk_isp： isp 时钟频率

Interrupt：包含mipi中断、isp内各模块的中断，数据有递增，说明有数据进isp，ErrCnt错误中断统计信息

Input　rkcif_mipi_lvds Format:SBGGR10_1X10 Size:2688x1520@30fps Offset(0,0)

输入源、输入格式、分辨率、帧率和裁剪信息

Isp Read　mode:frame1 (frame:4061 rate:66ms idle time:10ms frameloss:6077) cnt(total:2026 X1:1969 X2:56 X3:-1)

Read为回读模式（online对应直通模式）

frame1单帧线性（frame2 hdr2帧模式）

frame:帧号

rate:帧率，前后帧间隔

idle/working:isp硬件状态

time:isp硬件处理时间

frameloss:输入丢帧数

cnt: total:总回读次数 X1:1次回读次数 X2:2次回读次数 X3:3次回读次数

Output　rkispp0 Format:FBC420 Size:2688x1520 (frame:4061 rate:66ms frameloss:2018)
Output　rkisp_selfpath Format:FBCG Size:672x190 Dcrop(0,0|2688x1520) (frame:4061 rate:66ms delay:28ms frameloss:2017)

output输出流信息，输出buf有轮转才有更新信息

size:输出分辨率

Dcrop:输出裁剪信息

delay:对应时延信息（当前输出点时间搓-输入图像时间搓）

其他： isp各个模块的开关状态

ISP procfs debug功能使用说明

config isp procfs note to set debug mode:

BIT(0) for show isp reg

BIT(1) for dump bay3d iir/cur/ds buf once

BIT(8) for skip aiq params update

BIT(9) for skip hw params update, only for same w & h & bayer

如下举例5种功能，多摄isp复用情况下读取和操作寄存器比较实用，节点名对应虚拟isp名不同平台略有差异，具体ls /proc/rkisp*查看

1)show isp reg info

echo mode=0x1 > /proc/rkisp0-vir0

cat /proc/rkisp0-vir0

2)dump bay3d iir/cur/ds buf to /tmp

echo mode=0x2 > /proc/rkisp0-vir0

3)skip aiq params update

echo mode=0x100 > /proc/rkisp0-vir0

echo 0x2200=0 0x538=0x800080 > /proc/rkisp0-vir0

4)skip hw params update

echo mode=0x200 > /proc/rkisp0-vir0

5)close debug mode

echo mode=0 > /proc/rkisp0-vir0

case 3 or 4 also can will with 1 and 2, such as mode=0x103

```
[root@RV1126_RV1109:/]# cat /proc/rkispp*
rkispp0    Version:v00.01.08
clk_ispp   400000000
aclk_ispp  500000000
hclk_ispp  250000000
Interrupt  Cnt:13 ErrCnt:0
Input      rkisp0 Format:FBC420 Size:2688x1520 (frame:23 rate:67ms delay:49ms)
Output     rkispp_scale0 Format:NV12 Size:1920x1080 (frame:5 rate:42ms
delay:76ms frameloss:0)
TNR        ON(0x200000d) (mode: 2to1) (global gain: disable) (frame:21 time:7ms
idle) CNT:0x0 STATE:0x1e000000
NR         ON(0x57) (external gain: enable) (frame:5 time:9ms idle) 0x5f0:0x0
0x5f4:0x0
SHARP      ON(0x19) (YNR input filter: ON) (local ratio: OFF) 0x630:0x0
FEC        OFF(0x2) (frame:0 time:0ms idle) 0xc90:0x0
ORB        OFF(0x0)
Monitor    ON Cnt:0
```

ispp输入输出信息，流程isp->TNR->NR->FEC，rate/time/delay含义跟上面描述一样，对应当前节点的输入或输出信息

```
[root@RV1126_RV1109:/]# cat /proc/rkcif_mipi_lvds
Driver Version:v00.01.0a
Work Mode:ping pong
Monitor Mode:idle
aclk_cif:500000000
hclk_cif:250000000
dclk_cif:297000000
Input Info:
        src subdev:m01_f_os04a10 1-0036-1
        interface:mipi csi2
        lanes:4
        vc channel: 0 1
        hdr mode: hdr_x2
        format:SBGGR10_1X10/2688x1520@30
        crop.bounds:(0, 0)/2688x1520
Output Info:
        format:BG10/2688x1520(0,0)
        compact:enable
        frame amount:264
        early:10 ms
```

```
        single readout:30 ms
        total readout:30 ms
        rate:33 ms
        fps:30
        irq statistics:
                        total:515
                        csi over flow:0
                        csi bandwidth lack:0
                        all err count:0
                        frame dma end:515
```

Work Mode：rv1109之后默认使用ping pong，建议使用ping pong模式。

Monitor Mode: 监控模式，开启监控模式后，在mipi检查到异常的情况下，对vicap进行复位。

Input Info: 输入端信息汇总

src subdev:输入端设备，一般指sensor设备，包含摄像头朝向，索引号，设备名称，i2c总线，7bit slave地址等信息

interface:数据物理接口，mipi、lvds、dvp等。

vc channel: 实际使用的vc通道，指mipi协议上的多通道传输的虚拟通道。

hdr mode: sensor的工作模式，分为normal、 hdr_x2、hdr_x3。

format: 输入数据类型

crop.bounds：裁剪参数，可在sensor驱动配置.get_selection，从而对输入源的数据适当裁剪。

Output Info: 输出端信息汇总

format:输出数据类型

compact: 默认紧凑型输出，相关定义查阅以下章节：如何抓取CIS输出的RAW、YUV数据

frame amount:

early: wake up模式下，在采集到wait_line行数据后，提前将buffer送isp处理，默认模式是在采集到完整帧后送isp处理，这边显示的是相对采集到完整帧， buffer提前送isp的优化时间。 wake up模式配置说明在：时延优化指南

single readout: hdr模式下，单帧传输时间，也就是长帧的传输时间。

total readout: hdr模式下，长帧开始传输和短帧传输结束的时间差，也就是一帧合成帧的原始传输时间。

rate: 帧间隔时间。

fps: 帧率。

irq statistics:中断信息

total: frame end + err 总的中断数

csi over flow: overflow异常的中断数

csi bandwidth lack: bandwidth lack 异常的中断数

frame dma end: frame end的中断数，这个中断数等于从stream start开始，sensor输出的帧数。

如何排查预览闪烁问题

排查闪烁的原因，首先确认闪烁来源，可以从AE log上分析。

AE log打印开启方式如下：

1、终端（串口或adb shell）执行export persist_camera_engine_log=0x1ff3

2、在步骤1的同一个终端运行librkaiq.so，可以通过rkisp_demo、RkLunch.sh等程序。

3、在1、2步骤的基础上，还是无法打印出AE log，可能默认的编译方式没有将log编译进去，可参考如下修改：

```
czf@ISP:~/rk356x_sdk/external/camera_engine_rkaiq$ git diff
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 46fba20..f5ea67f 100755
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@ -6,9 +6,9 @ if(NOT CMAKE_BUILD_TYPE)
FORCE)
endif()

-if(NOT CMAKE_BUILD_TYPE STREQUAL "Release")
#if(NOT CMAKE_BUILD_TYPE STREQUAL "Release")
add_definitions(-DBUILD_TYPE_DEBUG)
-endif()
#endif()
```

AE log 包含统计值MeanLuma、TMO后的统计值TmoMeanLuma，曝光参数等信息，通过这些参数信息可以初步分析闪烁的原因。

```
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7028: ================================ HDR-AE (enter) ================================
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7049: AecRun: SMeanLuma=0.280734, MMeanLuma=4.009174,LMeanLuma=0.000000,TmoMeanluma=6.188991,Isconverged=0,Longfrm=0
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7058: >>> Framenum=5 Cur Piris=128, Sgain=1.000000,Stime=0.000505,mgain=1.000000,mtime=0.003005,lgain=1.000000,ltime=0.003000
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:3887: S-HighLightLuma=16.500000,S-Target=110.000000,S-GlobalLuma=0.280734,S-Target=15.652778
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:4264: L-LowLightLuma=3.848156,L-Target=80.000000,L-GlobalLuma=4.009174,L-Target=75.000000
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:5725: AecHdrClmExecute: sgain=1.000000,stime=0.002510,mgain=2.148907,mtime=0.020000,lgain=0.000000,ltime=0.000000
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7162: calc result:piris=128,sgain=1.000000,stime=0.002514,mgain=2.137962,mtime=0.020000,lgain=0.000000,ltime=0.000000
[AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7166: ================================ (exit) ================================
```

闪烁原因：

1、TMO合成导致的闪烁，如下图，log经过过滤，可以清楚看到短帧、中帧的统计值一直很稳定，但是TMO后的统计值却出现跳变，说明TMO的相关参数在某些场景下不适用，分析到这一步，可以参考tunning guide文档调整参数，仍无法解决请联系RK的IQ工程师协助处理。

```
AecRun: SMeanLuma=3.642202, MMeanLuma=59.557796,LMeanLuma=0.000000,TmoMeanluma=46.662384,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.638532, MMeanLuma=59.590824,LMeanLuma=0.000000,TmoMeanluma=46.708256,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.644037, MMeanLuma=59.631191,LMeanLuma=0.000000,TmoMeanluma=46.691742,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.642202, MMeanLuma=59.647705,LMeanLuma=0.000000,TmoMeanluma=46.713760,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.638532, MMeanLuma=59.598164,LMeanLuma=0.000000,TmoMeanluma=64.000000,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.640367, MMeanLuma=59.543118,LMeanLuma=0.000000,TmoMeanluma=46.702751,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.644037, MMeanLuma=59.620182,LMeanLuma=0.000000,TmoMeanluma=46.719265,Isconverged=1,Longfrm=0
AecRun: SMeanLuma=3.631193, MMeanLuma=59.620182,LMeanLuma=0.000000,TmoMeanluma=46.746788,Isconverged=1,Longfrm=0
```

2、raw上的统计值很稳定，TMO后的统计值也很稳定，但是画面上仍能看到闪烁，说明isp后续的模块中存在导致闪烁的问题，排查到这一步请联系RK工程师进一步分析。

3、闪烁出现在time、gain同时变化时，说明time gain的生效时刻配置可能存在问题，一般sensor的time是n+2生效，gain n+2 n+1都比较常出现，如果明确知道time、gain的生效帧，可以将参数填写到iq文件测试，以下是xml版本的iq文件说明，值2表示n+2生效，n表示第n帧的帧头将曝光参数设置下去。json版本参数类似请自行查阅文档配置。

```
<EXP_DELAY index="1" type="struct" size="[1 1]">
```

```
        <Normal index="1" type="struct" size="[1 1]">
             <time_delay index="1" type="double" size="[1 1]">
             [2 ]
         </time_delay>
             <gain_delay index="1" type="double" size="[1 1]">
             [2 ]
         </gain_delay>
             <dcg_delay index="1" type="double" size="[1 1]">
             [1 ]
         </dcg_delay>
       </Normal>
       <Hdr index="1" type="struct" size="[1 1]">
          <time_delay index="1" type="double" size="[1 1]">
             [2]
          </time_delay>
          <gain_delay index="1" type="double" size="[1 1]">
             [2]
          </gain_delay>
          <dcg_delay index="1" type="double" size="[1 1]">
             [1]
          </dcg_delay>
       </Hdr>
</EXP_DELAY>
```

如果无法确定生效帧，iq文件中AE模块中有AecSyncTest节点用来测试，这个模块的原理是两组曝光参数，间隔一定的帧数，来回切换。可以将两组参数的time设置成一样的值，gain设置成不一样的值，然后去分析AE log的MeanLuma统计值、及对应的time gain参数值。

4、如果闪烁的时候time是稳定的，gain值来回调，有可能gain的转换公式存在问题，也有可能sensor本身的线性度比较差。

4.1 转换公式和sensor的转换说明和驱动的写法相关，详细说明可以看下Sensor Info 填写指南。可以计算下AE log上的time gain转换到寄存器的值，和驱动打印出来的time gain寄存器值做比较，看自行计算的寄存器值和程序计算出来的值是否一致，不一致则需要确认转换公式和驱动的写法，看是否存在问题。

4.2 线性度问题，可以通过抓raw图，用看图工具获取图像统计值确认线性度。

4.2.1 time 线性度测试：

a. 盖上毛玻璃（可用薄纸巾替代，作用是让整副图像入光均匀，处于线性区），固定gain值为1，分别抓取10ms 20ms 30ms的raw图，获取统计值（一般软件的统计值为8bit，范围0~255），记录表格

b. 镜头完全遮黑，抓取一张raw图，这张图的统计值为黑电平值。（由于精度要求不高，这边的time gain值不做要求，不要太夸张即可，比如测试的gain都是按1x设置，抓黑电平的raw图用1000x，这样对统计值的影响就比较大，不可取）

c. 在表格中，将步骤a记录的统计值分别减掉步骤b的黑电平，将减掉黑电平的统计值和曝光时间做折线图，如为直线或接近直线，可认为线性度良好。

注：

1、驱动中有supported_modes配置表里有vts_def（默认配置下的帧长，包含场消隐）、帧率，通过两个参数可以简单换算曝光时间，假设帧率是30fps，vts_def是1200，帧间隔为1s/30fps=33.333ms 则10ms对应的曝光行为10/33.333*1200=360行，曝光参数设置参考 如何设置CIS的曝光参数

2、步骤a抓取的raw图统计值要大于黑电平，小于180

4.2.2 gain值 线性度

a. 盖上毛玻璃，固定time值为10ms，分别抓取1x 2x 4x 8x等gain值的raw图，获取统计值（一般软件的统计值为8bit，范围0~255），记录表格，若有些gain值下的统计值不在180以下的情况，可调整time值，分段测试。

b. 镜头完全遮黑，抓取一张raw图，这张图的统计值为黑电平值。

c. 在表格中，将步骤a记录的统计值分别减掉步骤b的黑电平，将减掉黑电平的统计值和gain做折线图，如为直线或接近直线，可认为线性度良好。

注：

1、步骤a抓取的raw图统计值要大于黑电平，小于180

2、如怀疑某段gain值下线性度有问题，可单独测试该段的线性度，不需要完整gain区间做线性度测试。

5、高亮环境下，如室外阳光比较强的情况下，出现闪烁，有可能是曝光值与寄存器转换存在问题，比如应用层认为的5行，通过寄存器转换，实际生效的可能是4行，这样就存在一行的亮度偏差，一行的亮度偏差在室外强光环境下就很容易导致闪烁，需要对比sensor手册关于曝光计算的描述，仔细检查驱动的实现是否正确。

如何排查光源处紫色溢出的问题

1. 线性模式

线性模式下，光源处发紫，有可能是sensor的gain值设置到非法值，导致图像异常，需要检查驱动换算的gain值寄存器是否符合sensor手册描述的限制条件。

2. HDR 模式

HDR 模式下，主要有以下两个原因：

2.1 短帧图像偏移导致，HDR合成错位。这种情况可以看下内核log是否有mipi相关报错。若没有mipi报错，进一步确认设置给sensor的曝光参数是否有问题， HDR sensor通常对长短帧曝光有较多限制，这些限制条件可以参考Sensor Info 填写指南。将驱动写入sensor的寄存器值打印出来，和sensor手册描述的限制条件做比较，看是否有设置不符合要求的寄存器值。

2.2 长短帧的曝光参数ratio和实际图像生效的ratio不匹配，这种情况下同样是参考2.1确认曝光参数的转换是否有问题。比较常见的问题是大部分sensor对短帧的曝光最大值有限制，假设某sensor短帧曝光最大值是2ms，而iq文件里面sensor info、AEC参数相关配置没有配置短帧最大值，或者短帧最大值限制条件设置得比驱动限制要大，比如说AEC有可能分解出短帧曝光3ms，设置到驱动，实际最大只能设置到2ms，但是驱动并没有直接返回错误给AEC，这样AEC认为3ms设置成功，并将曝光参数传递给TMO模块，导致合成的图像ratio不对，亮度不对。而短帧合进去的地方通常是过曝区域，通常表现在光源处，也就是常见光源处发紫。所以图像光源处发紫，重点在于排查AEC分解出来的曝光参数，和实际设置到sensor内的曝光参数是否存在差异。

Sensor Info 填写指南

以imx290为例：

[imx290]

CISAgainRange=1 31.6

CISDgainRange=1 125.89

单使用analog gain（again）亮度不足的时候，通常使用digital gain（dgain）来补偿，rk一般做法是将dgain混合到again下发，由驱动分解出again、dgain，分别设置到对应的sensor寄存器；

Imx290手册描述gain值分布如下：

0dB to 30 dB：Analog Gain 30 dB (step pitch 0.3dB)

30.3 dB to 72 dB: Analog Gain 30dB + Digital Gain 0.3 to 42dB (step pitch 0.3dB)

也就是again 30dB ， dgain 42dB

通过公式：

db = 20 * log10(gain倍数)

reg_gain = 20 * log10(gain倍数) * 10 / 3

算出倍数单位的again = 10^(30db/20)=31.6x

Dgain = 10^(42db/20) = 125.89x

CISExtraAgainRange=2 63.2

CISExtraAgainRange是again * dcg ratio的范围值，部分sensor支持HCG/LCG，HCG能够在暗环境获得更好的信噪比，若驱动实现相关功能，这边需要填写对应的转换gain值。Imx290手册描述Conversion efficiency ratio典型值为2，也就是当设置2x的again，且为HCG mode，实际生效的gain值为4x，所以CISExtraAgainRange=2*[1 31.6]，如果驱动没实现HCG/LCG，默认填[1 1]

CISIspDgainRange=1 1

Isp dgain，目前没有使用，按默认值即可

CISMinFps=10

允许的最小帧率，假设需要降帧的5fps，且sensor支持降帧到5fps的话，这边也要同步修改为5，才能通过iq配置或api降帧。

CISTimeRegMin=1

线性模式下，曝光行最小单位

CISLinTimeRegMaxFac=1.00 2.00

线性模式下最大曝光行

CISTimeRegOdevity=1 0

线性模式的曝光行奇偶性，从手册描述shs1可以逐1递增，曝光行对应也能逐1递增。

Imx290手册有如下描述Integration time = 1 frame period - (SHS1 + 1) X(1H period)

Rk框架目前从aiq下发给驱动的曝光单位都是以行时间为单位，如果部分sensor是半行单位，需要转化成行单位，从imx290的曝光公式可以看出是行单位，上述公式重新描述下为

曝光行time = vts -shs1 - 1

而从sensor 手册关于shs1的描述，限制范围为1~(Number of lines per frame - 2),也就是1~（vts-2）

所以CISTimeRegMin = vts -shs1 - 1 = vts -（vts-2）- 1 = 1

CISLinTimeRegMaxFac = vts - shs1 - 1 = vts - 1 - 1 = vts - 2

vts是一帧的总行数，包含场消隐，不同手册描述稍微有点差异，1 frame period、Number of lines per frame描述的都是vts。

CISHdrTimeRegMin=1

Hdr最小曝光行

CISHdrTimeRegMax=8 0 0

Hdr最大曝光行，增加这变量是因为有些sensor短帧最大曝光行有限制，并不能随着长帧曝光的降低而增加，也不能随着帧率的降低而增加，imx290就是这样的一个sensor，按照sony标准配置，短帧曝光行最大为8行，imx307 DOL文档有描述decreasing exposure ratio mode，按照里面的配置可以使短帧曝光行最大为222行，imx290 DOL文档未见描述，具体咨询sony是否支持。

CISHdrTimeRegOdevity=1.00 0.00

CISHdrTimeRegSumFac=1.00 6.00

Sony DOL 文档有如下描述：

List of DOL 2 frame Settings

Items	Symbol	Setting Register	Setting value / Condition
Frame Set Count	FSC	VMAX	VMAX × 2
Shutter timing of SEF1	SHS1	SHS1	2 or more and RHS1 - 2 or less
Readout timing of SEF1	RHS1	RHS1	2n+5 (n = 0, 1, 2 ⋯) and RHS1 ≤ FSC - BRL × 2 - 21
Shutter timing of LEF	SHS2	SHS2	RHS1 + 2 or more and FSC - 2 or less

Items	symbol	Formulas	Unit	Remarks
Exposure time of LEF	t_{LEF}	FSC - (SHS2 + 1)	H	-
Exposure time of SEF1	t_{SEF1}	RHS1 - (SHS1 + 1)		-
Exposure ratio	-	t_{LEF} / t_{SEF1}	-	Combining 2 frame

CISHdrTimeRegMin：

通过表格可以计算长帧最小曝光值=FSC-SHS2-1=FSC-(FSC-2)-1=1

短帧最小曝光值=RHS1-SHS1-1=RHS1-(SHS1-2)-1=1

所以hdr下最小曝光行为1

CISHdrTimeRegOdevity：从表格上看，shs1和shs2没有类似2n或2n+1的限制，所以对应的曝光行可以逐1递增

CISHdrTimeRegSumFac：

长短帧曝光和=（FSC-SHS2-1）+ （RHS1-SHS1-1）

SHS2、SHS1同时取最小值，使得长帧和短帧的曝光都最大

长短帧曝光和=（FSC-（RHS1+2）-1）+ （RHS1-2-1）=FSC-6

对于2帧DOL hdr，FSC=2*vts*，*故长短帧曝光和最大值为=2*vts-6

也就是CISHdrTimeRegSumFac=[2 6],但是为了计算方便，sony的DOL hdr驱动会将FSC作为vts上传的aec，也就是上传的vts实际上是已经乘2倍了，所以CISHdrTimeRegSumFac=[1 6]

CISTimeRegUnEqualEn=0

长短帧的time是否可以相等，由于imx290短帧限制，无法在任何情况下都能相等

CISHdrGainIndSetEn=1

长短帧的gain是否需要设置成一样，1表示可以设置成不一样的值，0表示长短帧gain要一样，具体看sensor描述，有些sensor长短帧共用一组寄存器，有些sensor虽然长短帧gain有分别的寄存器，但是设计原因需要两组寄存器设置成一样的值，为了曝光分解的正确性，需要准确填写这个参数。

注：imx290需要注意FPGC PFGC_1值的设置,DOL文档有具体描述。

FullResolution=1920x1080

GainRange=1 2 20 20 1 0 20 2 4 10 0 1 20 40 4 8 5 -20 1 40 60 8 16 2.5 -40 1 60 80 16 32 1.25 -60 1 80 100 32 64 0.625 -80 1 100 120 64 128 0.3125 -100 1 120 140 128 256 0.15625 -120 1 140 160 256 512 0.078125 -140 1 160 180 512 1024 0.0390625 -160 1 180 200

IsLinear=0

Rk平台支持倍数为单位的gain值设置和sony db方式的gain值设置，0表示使用db，对应imx290可以直接使用db方式，也可以使用上述的GainRange分解公式，GainRange分解公式会稍微有误差，毕竟非线性的曲线分解成多段线性曲线了。

NonLinear=DB_MODE

PatternMode=RGGB

TimeFactor=0 0 1 0.5

Time分解公式，建议保持这个公式，计算不符合公式的情况下，sensor驱动内做转换。

hdr_dcg_ratio=2

normal_dcg_ratio=2

Dcg ratio前面已经描述

SensorFlip=0

默认的mirror flip状态，bit0 mirror，bit1 flip

附录A CIS驱动V4L2-controls列表

CID	描述
V4L2_CID_VBLANK	Vertical blanking. The idle period after every frame during which no image data is produced. The unit of vertical blanking is a line. Every line has length of the image width plus horizontal blanking at the pixel rate defined by V4L2_CID_PIXEL_RATE control in the same sub-device.
V4L2_CID_HBLANK	Horizontal blanking. The idle period after every line of image data during which no image data is produced. The unit of horizontal blanking is pixels.
V4L2_CID_EXPOSURE	Determines the exposure time of the camera sensor. The exposure time is limited by the frame interval.
V4L2_CID_ANALOGUE_GAIN	Analogue gain is gain affecting all colour components in the pixel matrix. The gain operation is performed in the analogue domain before A/D conversion.

CID	描述
V4L2_CID_PIXEL_RATE	Pixel rate in the source pads of the subdev. This control is read-only and its unit is pixels / second. Ex mipi bus：pixel_rate = link_freq * 2 * nr_of_lanes / bits_per_sample
V4L2_CID_LINK_FREQ	Data bus frequency. Together with the media bus pixel code, bus type (clock cycles per sample), the data bus frequency defines the pixel rate (V4L2_CID_PIXEL_RATE) in the pixel array (or possibly elsewhere, if the device is not an image sensor). The frame rate can be calculated from the pixel clock, image width and height and horizontal and vertical blanking. While the pixel rate control may be defined elsewhere than in the subdev containing the pixel array, the frame rate cannot be obtained from that information. This is because only on the pixel array it can be assumed that the vertical and horizontal blanking information is exact: no other blanking is allowed in the pixel array. The selection of frame rate is performed by selecting the desired horizontal and vertical blanking. The unit of this control is Hz.

附录B MEDIA_BUS_FMT表

CIS sensor类型	Sensor输出format
Bayer RAW	MEDIA_BUS_FMT_SBGGR10_1X10 MEDIA_BUS_FMT_SRGGB10_1X10 MEDIA_BUS_FMT_SGBRG10_1X10 MEDIA_BUS_FMT_SGRBG10_1X10 MEDIA_BUS_FMT_SRGGB12_1X12 MEDIA_BUS_FMT_SBGGR12_1X12 MEDIA_BUS_FMT_SGBRG12_1X12 MEDIA_BUS_FMT_SGRBG12_1X12 MEDIA_BUS_FMT_SRGGB8_1X8 MEDIA_BUS_FMT_SBGGR8_1X8 MEDIA_BUS_FMT_SGBRG8_1X8 MEDIA_BUS_FMT_SGRBG8_1X8
YUV	MEDIA_BUS_FMT_YUYV8_2X8 MEDIA_BUS_FMT_YVYU8_2X8 MEDIA_BUS_FMT_UYVY8_2X8 MEDIA_BUS_FMT_VYUY8_2X8 MEDIA_BUS_FMT_YUYV10_2X10 MEDIA_BUS_FMT_YVYU10_2X10 MEDIA_BUS_FMT_UYVY10_2X10 MEDIA_BUS_FMT_VYUY10_2X10 MEDIA_BUS_FMT_YUYV12_2X12 MEDIA_BUS_FMT_YVYU12_2X12 MEDIA_BUS_FMT_UYVY12_2X12 MEDIA_BUS_FMT_VYUY12_2X12

CIS sensor类型	Sensor输出format
Only Y(黑白)即raw bw sensor	MEDIA_BUS_FMT_Y8_1X8 MEDIA_BUS_FMT_Y10_1X10 MEDIA_BUS_FMT_Y12_1X12

附录C CIS参考驱动列表

CIS 数据接口	CIS 输出数据类型	Frame/Field	参考驱动
MIPI	Bayer RAW	frame	0.3M ov7750.c gc0403.c 0.9M jx_h62.c 1.2M ov9750.c jx-h65.c 2M ov2685.c ov2680.c ov2735.c ov02g10.c ov02b10.c gc2385.c gc2355.c gc2053.c sc2232.c sc2239.c sc223a sc210iot.c sp250a.c 4M gc4c33.c jx_k04.c os04c10.c sc401ai.c 5M ov5695.c ov5648.c ov5670.c gc5024.c gc5025.c gc5035.c hm5040.c

CIS 数据接口	CIS 输出数据类型	Frame/Field	参考驱动
			sc5239.c hynix_hi556.c 8M ov8858.c os08a10.c os08a20.c sc8220 imx378.c imx317.c imx219.c gc8034.c hynix_hi846.c s5kgm1sp.c s5k4h7yx.c 13M ov13850.c ov13b10.c imx258.c ov12d2q.c

CIS 数据接口	CIS 输出数据类型	Frame/Field	参考驱动
MIPI	Bayer raw hdr	frame	2M imx307.c imx327.c imx462.c gc2093.c ov02k10 ov2718.c sc200ai.c sc2310.c jx-f37.c 4M ov4689.c os04a10.c imx347.c imx464.c sc4238.c 5M imx335.c os05a20.c sc500ai.c 8M imx334.c imx415.c
MIPI	YUV	frame	2M gc2145.c
MIPI	RAW BW	frame	0.3M ov7251.c 1M ov9281.c 1.3M sc132gs.c
MIPI	YUV	field	tc35874x.c
ITU.BT601	Bayer RAW		2M imx323.c ar0230.c

CIS 数据接口	CIS 输出数据类型	Frame/Field	参考驱动
ITU.BT601	YUV		0.3M gc0329.c gc0312.c gc032a.c 2M gc2145.c gc2155.c gc2035.c bf3925.c
ITU.BT601	RAW BW		0.3M sc031gs.c 1.3M sc032gs.c
ITU.BT656	Bayer RAW		2M imx323(可支持)

附录D VCM driver ic参考驱动列表

参考驱动
vm149c.c
dw9714.c
dw9718.c
fp5510.c
gt9760s.c
fp5501.c (step motor)
mp6507.c (step motor)
ms41908.c (step motor)

附录E Flash light driver ic参考驱动列表

参考驱动
sgm3784.c
leds-rgb13h.c(GPIO控制)

[*CIS 设备注册(DTS)]:*
[#CIS 设备注册(DTS)]:
[#CIS 设备注册(DTS)]: