Rockchip_Driver_Guide_VI_EN

ID: RK-YH-GX-604

Release Version: V1.1.3

Release Date: 2022-09-09

Security Level: □Top-Secret □Secret □Internal ■Public

DISCLAIMER

Trademark Statement

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: [fae@rock-chips.com]mailto:fae@rock-chips.com)

Foreword

Overview

This article aims to describe the role of the RKISP (Rockchip Image Signal Processing) module, the overall workflow, and related API interfaces. Mainly to driver engineers provide assistance in debugging Camera.

Product Version

Chip Name	Kernel Version
RV1126/RV1109	Linux 4.19

Target Audience

This document (this guide) is mainly applicable to the following engineers:

Drive development engineer

System Integration Software Development Engineer

Applicable platforms and systems

Chip Name	Software System	Support Status
RV1126	Linux(Kernel-4.19) and kernel-5.10	Y
RV1109	Linux(Kernel-4.19) and kernel-5.10	Y
RK3566	Linux(Kernel-5.10)	Y
RK3568	Linux(Kernel-5.10)	Y

Revision History

Version Number	Author	Revision Date	Revision Description
v0.1.0	Cain Cai	2020-06-11	Initial version
v1.0.0	Zefa Chen	2020-10-30	Added focus, zoom, iris, ircut descriptions
v1.0.1	Zefa Chen	2021-01-04	Modified format error
v1.0.2	Cain Cai	2021-01-21	RV1109/RV1126 memory optimization guide
v1.0.3	Allon Huang	2021-02-04	Added VICAP LVDS/DVP/MIPI and other interface device node registration instructions
v1.0.4	Cain Cai	2021-04-08	Add chip version different and mulit sensor dts for rk356x
v1.0.5	Zefa Chen	2021-04-24	Added MS41908 stepper motor driver description Improve the collection of RAW/YUV command instructions
v1.0.6	Zefa Chen	2021-07-23	The description of the vicap node falls back to be consistent with the driver
v1.0.7	Cain Cai	2021-08-03	RV1109/RV1126 delay optimization guide
v1.0.8	Zefa Chen	2021-08-24	Added FAQ: preview flickering, purple light source overflow problem
v1.0.9	Cain cain	2021-10-21	Added driver update method description
v1.1.0	Zefa Chen	2021-10-29	Add description of isp/vicap raw storage format Add description of vicap abnormal reset Add description of three camera dts configuration Update CIS driver reference table/VCM driver reference table
v1.1.1	Zefa Chen	2021-12-24	Added RK3588 description added multi-camera synchronization mechanism chapter
v1.1.2	Cain Cai	2022-08-29	Added rv1106 instructions, update proc node information and debug mode usage instructions
v1.1.3	Zefa Chen/Xinquan Chen	2022-9-9	The section on VICAP/ISP Special Acquisition Mode describes the processing process of multi-channel RAW data stitching and acquisition in the 8 mesh solution

Contents

1. VI difference of the chip

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
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 single ISP 8192x6144 dual ISP synthesis	Upto 3 frames HDR

SOC	VI IP	VI Interface	Bayer CIS max resolution	Feature
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 single ISP 8192x6144 dual ISP synthesis	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

2. Camera software driver catalog description

Linux Kernel-4.19

|-- arch/arm/boot/dts DTS configuration file

|-- drivers/phy/rockchip

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

|-- drivers/media

|-- platform/rockchip/cif

|-- platform/rockchip/isp

|-- dev Including probe, asynchronous registration, clock, pipeline, iommu and media/v4l2 framework

|-- capture Including mp/sp/rawwr configuration and vb2, frame interrupt processing

|-- dmarx Including rawrd configuration and vb2, frame interrupt processing

|-- isp_params 3A related parameter settings

|-- isp_stats 3A related statistics

|-- isp_mipi_luma mipi data brightness statistics

|-- regs Register-related read and write operations

|-- rkisp isp subdev and entity registration

|-- csi csi subdev and mipi configuration

|-- bridge bridge subdev, isp and ispp interactive bridge

|-- platform/rockchip/ispp

|-- dev Including probe, asynchronous registration, clock, pipeline, iommu and media/v4l2 framework

|-- stream Including 4 video output configuration and vb2, frame interrupt processing

|-- rkispp ispp subdev and entity registration

|-- params TNR/NR/SHP/FEC/ORB parameter setting

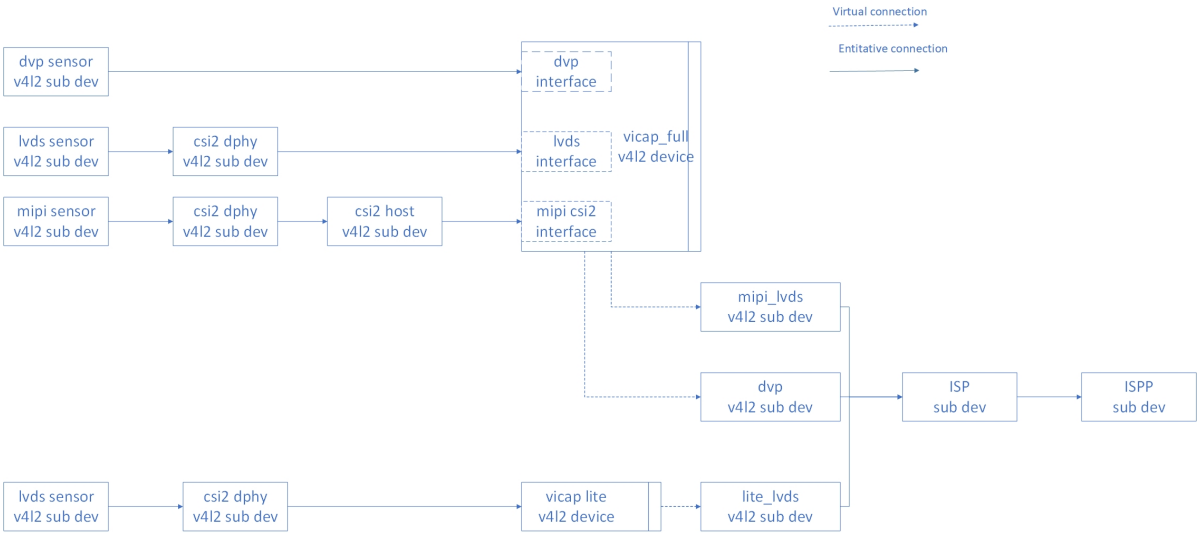
|-- stats ORB statistics

|-- i2c

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

3. Link relationship between ISP and VICAP

For the RV1126/RV1109 and RK356X platforms, VICAP and ISP are two independent image processing IPs. If the images collected by VICAP are processed by ISP, the v4l2 sub device of the VICAP corresponding interface needs to be generated at the driver level to link to the node corresponding to the ISP. , To provide parameters for the ISP driver to use. Please refer to RKISP driver for ISP driver description and RKVICAP driver for VICAP driver description. The overall block diagram of the specific VICAP interfaces and the ISP link is as follows:

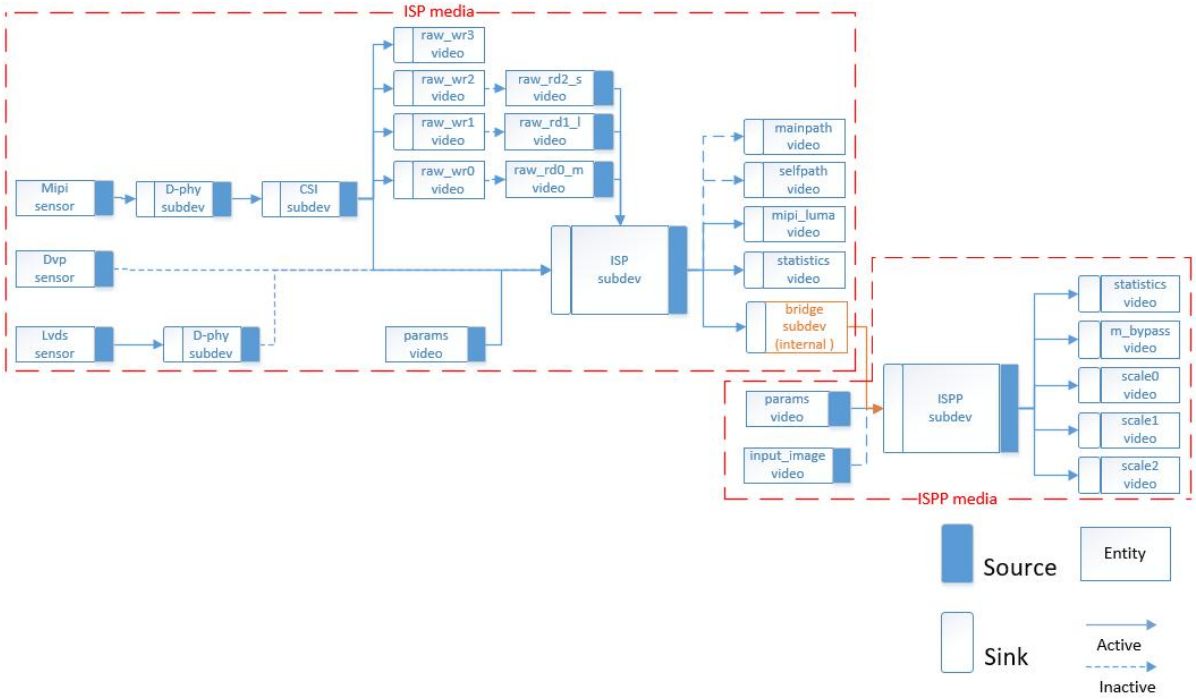


4. RKISP driver

4.1 Brief description of the framework

The RKISP driver is mainly based on the v4l2/media framework to implement hardware configuration, interrupt processing, control buffer rotation, and control the power on and off of subdevices (such as MIPI DPHY and sensor).

The following block diagram describes the topology of the RKISP driver:

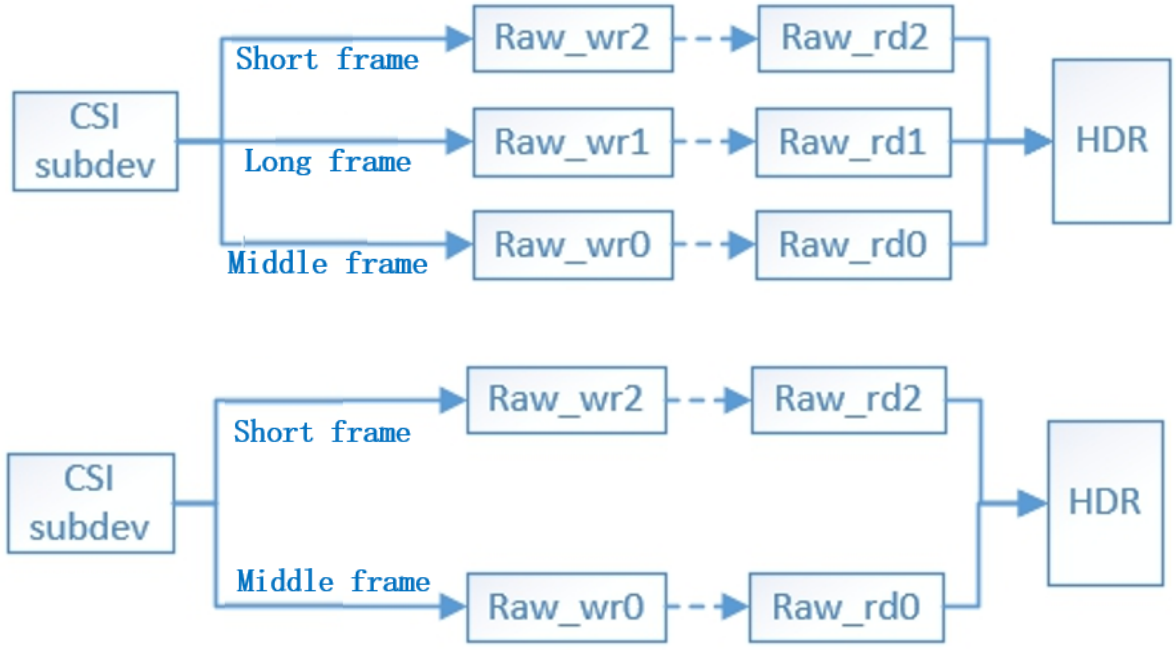


Name	Type	Description
rkisp_mainpath	v4l2_vdevcapture	Format: YUV, RAW Bayer; Support: Crop
rkisp_selfpath	v4l2_vdevcapture	Format: YUV, RGB; Support: Crop
rkisp-isp-subdev	v4l2_subdev	Internal isp blocks; Support: source/sink pad crop. The format on sink pad equal to sensor input format, the size equal 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 DDRfrom 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 output. The support max size is 4416x3312, min size is 66x258
rkispp_m_bypass	v4l2_vdev capture	Full resolution and yuv format

Name	Type	Description
rkispp_scale0	v4l2_vdev capture	Full or scale resolution and yuv format Scale 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 format Scale range:[2 8] ratio 1280 max width
rkispp_scale2	v4l2_vdev capture	Full or scale resolution and yuv format Scale range:[2 8] ratio 1280 max width

4.2 ISP HDR mode description

RKISP2 supports receiving MIPI sensor to output HDR 3 frames or 2 frames mode, the hardware collects data to DDR through 3 or 2 dmatx, and then reads the ISP through 3 or 2 dmarx, and the ISP does 3 or 2 frames synthesis, drive chain The road is as follows:



The CSI subdev obtains the output information of the sensor driver in multiple pad formats through get_fmt, which corresponds to the source pad of the CSI.

Please refer to the specific configuration of MIPI sensor driver Driver migration steps

Name	Name	Description
rkisp-isp-subdev	Sensor pad0	ISP capture Sensor vc0 (default) wide and high format output, commonly used linear mode
rkisp_rawwr0	Sensor pad1	Rawwr0 capture sensor vcX wide and high format output
rkisp_rawwr1	Sensor pad2	Rawwr1 capture sensor vcX wide and high format output
rkisp_rawwr2	Sensor pad3	Rawwr2 capture sensor vcX wide and high format output
rkisp_rawwr3	Sensor pad4	Rawwr3 capture sensor vcX wide and high format output

5. RKVICAP driver

5.1 Frame description

The RKVICAP driver is mainly based on the v4l2/media framework to implement hardware configuration, interrupt processing, control buffer rotation, and control the power on and off of subdevices (such as mipi dphy and sensor).

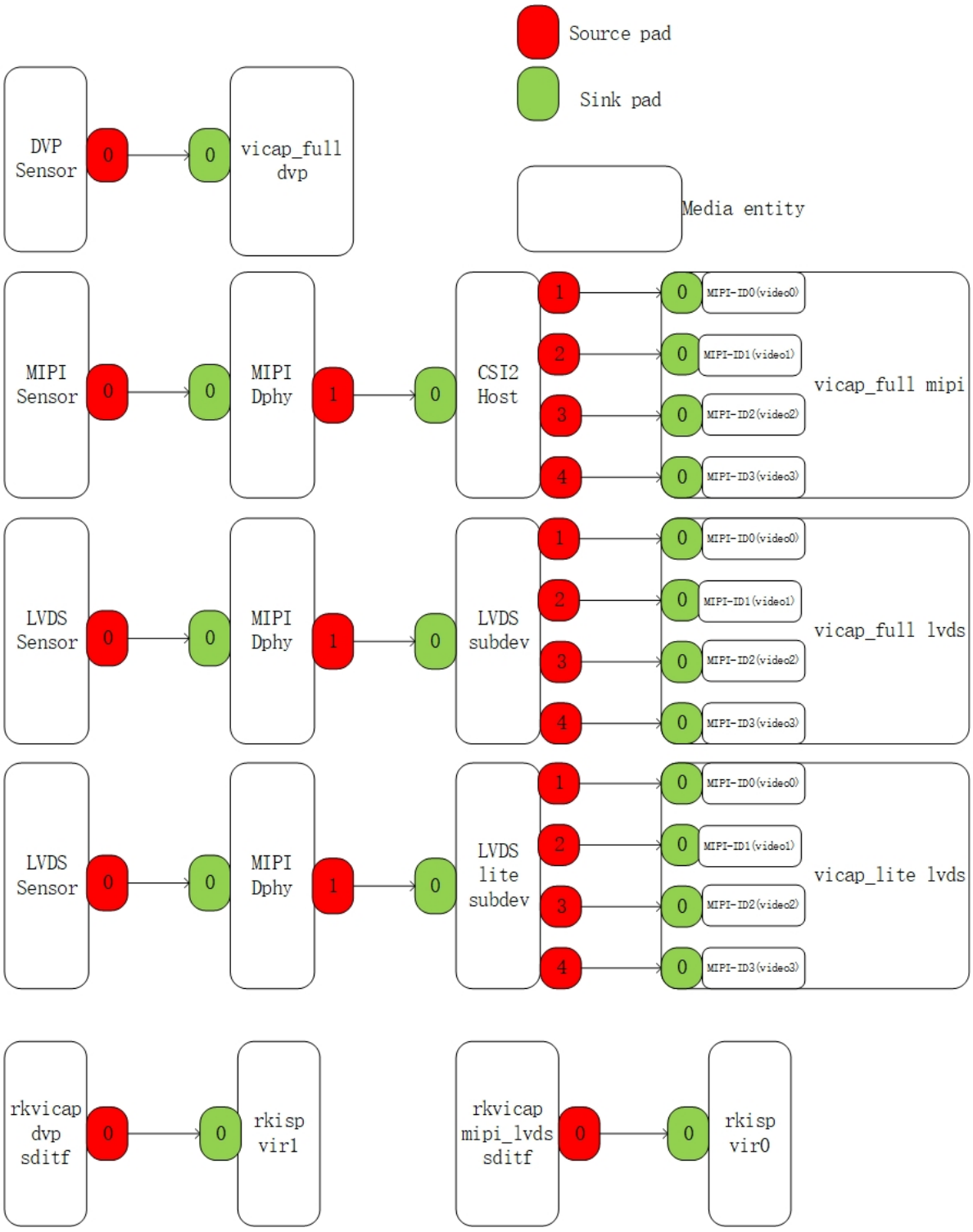
For RV1126/RV1109, VICAP has two IP cores, one of which is called VICAP FULL and the other is called VICAP LITE. VICAP FULL has three interfaces: dvp/mipi/lvds, dvp can work with mipi or lvds interface at the same time, but mipi and lvds cannot work at the same time, VICAP LITE only has lvds interface, which can work with VICAP FULL interface at the same time; VICAP FULL dvp The interface corresponds to a rkvicap_dvp node, the VICAP FULL mipi/lvds interface corresponds to a rkvicap_mipi_lvds node, and the VICAP LITE corresponds to a rkvicap_lite_mipi_lvds node. Each node can be collected independently.

For the RK356X chip, VICAP has only a single core and has two interfaces, dvp and mipi. The dvp interface corresponds to a rkvicap_dvp node, and the mipi interface corresponds to a rkvicap_mipi_lvds node (the same name as the VICAP FULL of RV1126/RV1109), and each node can be collected independently.

In order to synchronize the data collected by VICAP to the isp driver, it is necessary to link the logical sditf node generated by the VICAP driver to the virtual node device generated by the isp. The DVP interface corresponds to the rkvicap_dvp_sditf node, the mipi/lvds interface of VICAP FULL corresponds to the rkvicap_mipi_lvds_sditf node, and the VICAP LITE corresponds to rkvicap_lite_sditf.

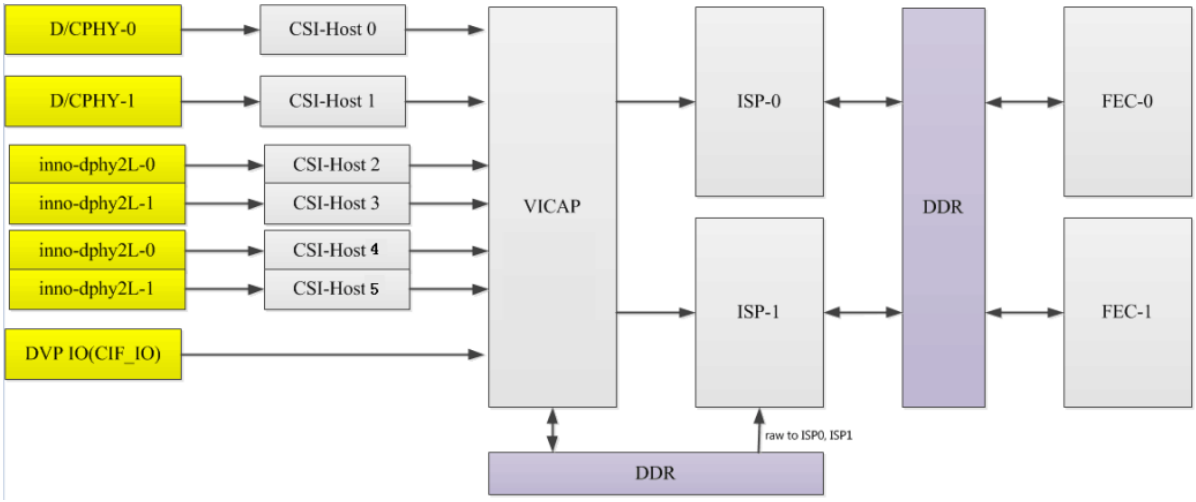
Please refer to the specific dts link method of each interface CIS Device Registration (DTS)

The following figure describes the topology of the device driven by RKVICAP:



6. RK3588 VI structure

6.1 Block diagram of the hardware path



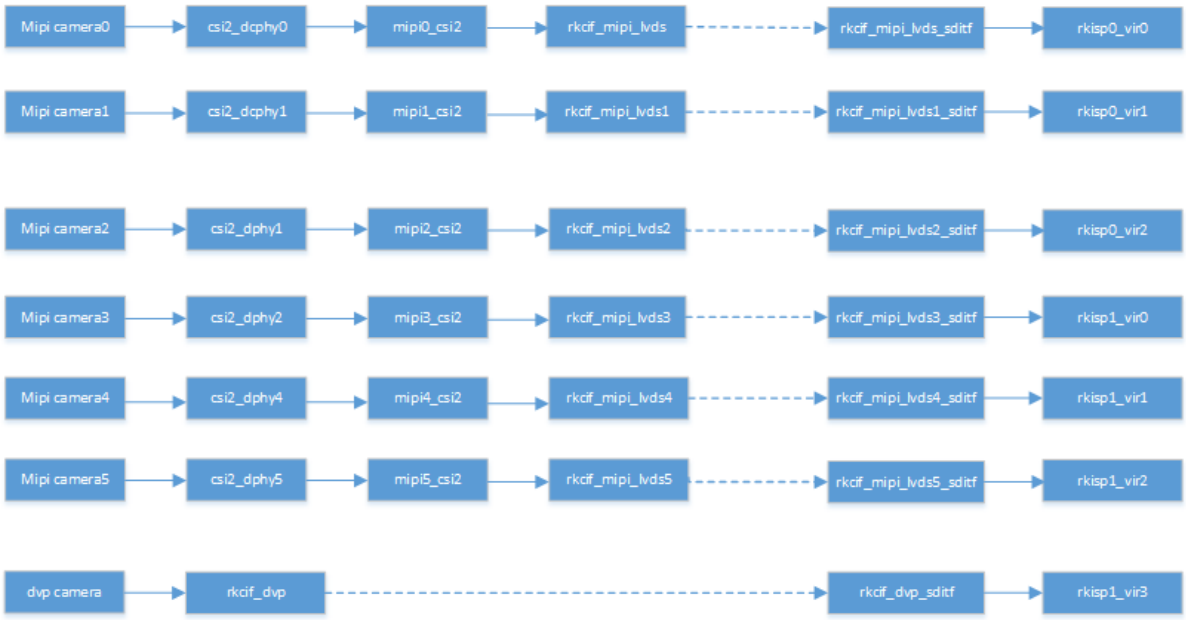
6.2 Multi-sensor support

A single hardware ISP supports up to four multiplexes, and the supported resolutions for ISP multiplexing are as follows:

2-way multiplexing: The maximum resolution is 3840x2160, and DTS corresponds to the configuration of 2-way rkisp_vir devices.

3-way or 4-way multiplexing: The maximum resolution is 2560x1536, and DTS corresponds to the configuration of 3 or 4-way rkisp_vir devices.

The hardware supports the acquisition of up to 7 sensors: 6MIPI + 1DVP, and the multi-sensor software path is as follows:



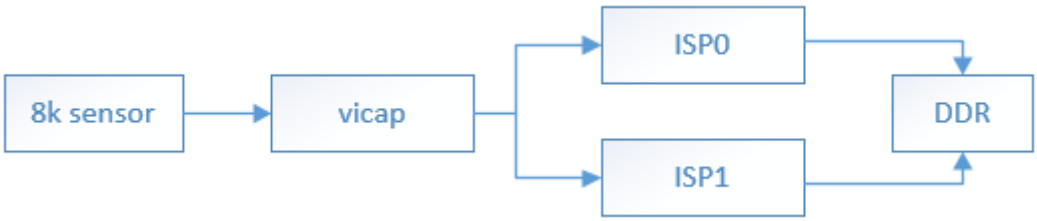
Block diagram description:

```
 1. The RK3588 supports two DCPHYs with node names of csi2_dcphy0/csi2_dcphy1.
Each dcphy hardware supports RX/TX at the same time, and RX is used for camera
inputs. Support DPHY/CPHY protocol multiplexing; It should be noted that TX/RX of
the same dcphy can only use DPHY at the same time or CPHY at the same time. Other
DCPHY parameters can be found in the RK3588 data sheet.
 2. RK3588 supports 2 DPY hardware, here we call it dphy0_hw/dphy1_hw, both DPHY
hardware can work in Full Mode and Split Mode modes.
```

```
        1. dphy0_hw
            1. Full Mode: The node name is csi2_dphy0, and up to 4 lanes are
supported.
            2. Split Mode: Split into 2 PHYs, which are csi2_dphy1 (using 0/1
lane) and csi2_dphy2 (using 2/3 LANE), each PHY supports up to 2 lanes.
            3. When dphy0_hw uses full mode, the link needs to be configured
according to the csi2_dphy1 link, but the node name csi2_dphy1 needs to be
modified to csi2_dphy0, and the software is distinguished by the sequence number
of PHY to distinguish the mode used by PHY.
        2. dphy1_hw
            1. Full Mode: The node name uses csi2_dphy3, and supports up to 4
lanes.
            2. Split Mode: Split into 2 PHYs, which are csi2_dphy4 (using 0/1
lane) and csi2_dphy5 (using 2/3 LANE), each PHY supports up to 2 lanes.
            3. When dphy1_hw uses full mode, the link needs to be configured
according to the csi2_dphy4 link, but the node name csi2_dphy4 need to be
modified to csi2_dphy3, and the software is to distinguish the mode used by the
PHY sequence number.
 3. To use the above MIPI PHY node, you need to configure the corresponding
physical node. （csi2_dcphy0_hw/csi2_dcphy1_hw/csi2_dphy0_hw/csi2_dphy1_hw）
 4. Each MIPI PHY requires a CSI2 module to parse the MIPI protocol, and the node
names are mipi0_csi2~mipi5_csi2.
 5. All camera data in RK3588 needs to be linked to the ISP through ViCap. RK3588
only supports one VICAP hardware, this VICAP supports simultaneous input of 6
MIPI PHYs, and one DVP data, so we divide VICAP into 7 nodes such as
rkcif_mipi_lvds~rkcif_mipi_lvds5, rkcif_dvp, etc., and the binding relationship
of each node needs to be configured in strict accordance with the node serial
number of the block diagram.
 6. The link relationship between each VICAP node and the ISP indicates the link
relationship by corresponding to the virtual XXX_sditf.
 7. The RK3588 supports 2 ISP hardware, each ISP device can virtualize multiple
virtual nodes, and the software reads each image data from the DDR in turn into
the ISP for processing. For multi-camera scenarios, it is recommended to
distribute the data stream evenly between the two ISPs.
 8. Pass-through and readback modes:
       1. Pass-through: means that the data is collected by VICAP, sent directly
to the ISP for processing, and is not stored in DDR. It should be noted that when
HDR pass-through, only short frames are true pass-through, long frames need to
exist DDR, and ISP reads from DDR.
       2. Read back: It means that the data is collected to the DDR through
VICAP, and after the application obtains the data, it pushes the buffer address
to the ISP, and the ISP obtains the image data from the DDR.
       3. When configuring DTS again, if only one virtual node is configured,
passthrough mode is used by default for an ISP hardware, and readback mode is
used by default if multiple virtual nodes are configured.
```

6.3 Dual ISP synthesis supports 8K resolution

VICAP collects sensor 8K data, then sends the left and right images to two ISPs for processing, and then outputs them to DDR, the process is as follows:

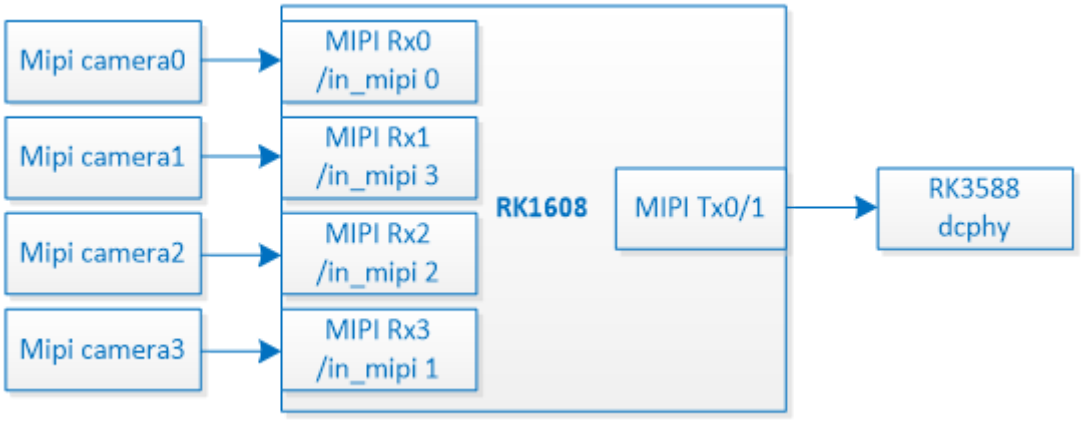


When the resolution is greater than 16M (4672x3504), two ISPs need to be used at the same time to process one image, and only single camera is supported.

1. From the rk3588s.dtsi file, you can find the three definitions of rkisp0, rkisp1, and rkisp_unite, when the resolution to be processed is greater than 16M, you need to close the rkisp0 and rkisp1 nodes, and enable the rkisp_unite, while modifying the corresponding iommu node.
2. Using rkisp_unite nodes can also virtualize multiple nodes, at this time, regardless of the resolution, the same image is cropped into two pictures, left and right, sent 2 ISPs for processing, and then synthesized into an output.
3. DTS Configuration refer to[rk3588单摄配置说明]

6.4 8 mesh MIPI sensor support

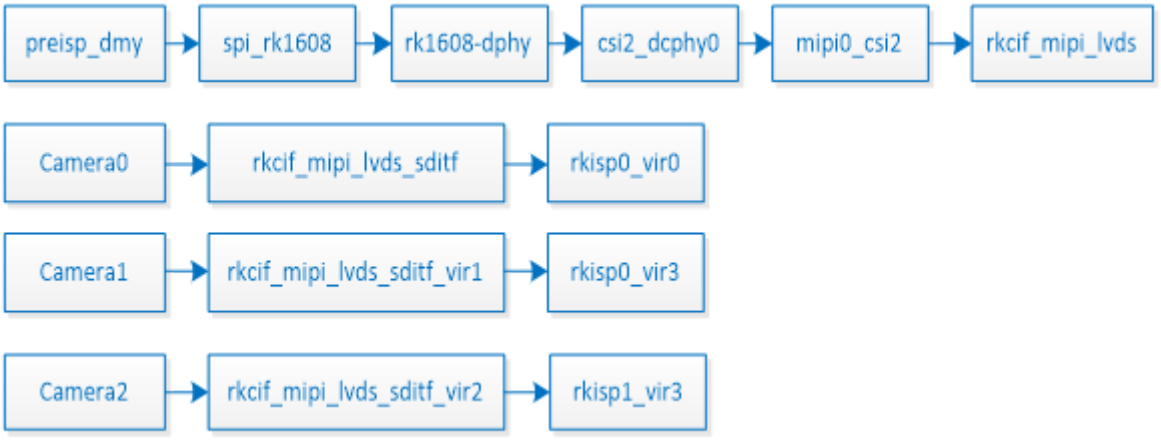
RK3588 hardware supports up to 6 mipi sensors, which can be expanded by RK1608 splicing, RK1608 supports 4 MIPI inputs (1, 2, 4lane), two MIPI outputs (1, 2, 4lane), through RK1608 splicing 3 + original 5 MIPI can achieve 8 MIPI sensor inputs, the hardware connection is as follows:



Software, you need to enable the RK1608 driver in the kernel's defconfig:

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

On the dts link, RK1608 abstracts the RK1608 dphy node and enters it as a virtual preisp_dmy sensor. RK1608 dphy, like ordinary sensors, is connected to RK3588 dcphy to form an RK1608->dcphy->csi->cif lvds link. The sensor physically hung in the RK1608 is directly hung in the rkcif_mipi_lvdsx_sditf, rkcif_mipi_lvdsx_sditf_vir1, rkcif_mipi_lvdsx_sditf_vir2, and rkcif_mipi_lvdsx_sditf_vir3 port0 in the dts as the input of the SDITF device. The output of SDITF is consistent with the original path, and is the corresponding ISP node. The DTS link is shown as follows:



The dts configuration of RK1608 is mainly on the rk1608-dpy device node, which mainly includes input and output channels and resolution, as follows:

```
#define LINK_FREQ               700000000
    mipidphy0: mipidphy0 {
    compatible =  "rockchip,rk1608-dphy";
        status = "okay";
        //rockchip,grf = <&grf>;
        id = <0>;  //RK1608 internal ID (0-2, spliced sequentially by ID)

        cam_nums = <1>;
        in_mipi = <1>;          //Input MIPI channel of the first sensor (0-3)
        out_mipi = <0>;         //Output MIPI channel (0-1)
        link-freqs = /bits/ 64 <LINK_FREQ>; //MIPI rate

        sensor_i2c_bus = <5>;      //The 8-mesh pattern is invalid
        sensor_i2c_addr = <0x1a>; //The 8-mesh pattern is invalid
        sensor-name = "IMX464";

        rockchip,camera-module-index = <9>; //Same as ordinary 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 {  //Configuration information for the second
sensor
            id = <1>;  //RK1608 internal ID (0-2, spliced by ID)
            in_mipi = <2>;
            out_mipi = <1>;
        };
        virtual-sub-sensor-config-1 { //Configuration information for the third
sensor
            id = <2>;  //RK1608 internal ID (0-2, spliced by ID)
            in_mipi = <3>;
            out_mipi = <1>;
        };
        /* multi-sensor mode end */

        format-config-0 {
            data_type = <0x2b>;
            mipi_lane = <2>;      //The number of lanes connected to the sensor
            mipi_lane_out = <4>; //The number of lanes output by RK1608
```

```
            field = <1>;      //The following is the same as the normal sensor
configuration
            colorspace = <8>;
            code = <MEDIA_BUS_FMT_SRGGB10_1X10>;
            width = <2712>;  //Sensor output resolution
            height= <1538>;
            hactive = <2712>; //RK1608 output resolution, the same width, height
is n*sensor
            vactive = <4614>;
            htotal = <3616>; //After increasing the width and height of blank,
30% of the blank is generally required
            vtotal = <4710>;
            inch0-info = <2712 1538 0x2b 0x2b 1>; //Sensor output resolution
            outch0-info = <2712 4614 0x2b 0x2b 1>; //RK1608 output resolution
            hcrop = <2560>; //Single resolution after cropping
            vcrop = <1520>;
        };
        ...
```

The DTS configuration reference of the sensor is as follows:

```
&rkcif_mipi_lvds_sditf {
    #address-cells = <1>;
    #size-cells = <0>;
    status = "okay";
    rockchip,combine-index = <0>; //Order in RK1608 stitching diagram
    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>;
            };
        };
    };
};
```

7. CIS (CMOS image sensor) driver

7.1 CIS Device Registration (DTS)

7.1.1 MIPI interface

For the RV1126 and RV1106 platforms, there are two separate and complete standard physical MIPI CSI2 DPYs, corresponding to the csi_dphy0 and csi_dphy1 on the DTS (see RV1126.DTSI), with the following characteristics:

- Data lanes are up to 4 lanes;
- Maximum speed 2.5Gbps/lane;

For the RK356X platform, there is only one standard physical mipi csi2 dphy, which can operate in two modes: full mode and split mode, split into three logical dphys csi2_dphy0/csi2_dphy1/csi2_dphy2 (see rk3568.dtsi), with the following characteristics:

Full mode

- Only use csi2_dphy0, csi2_dphy0 and csi2_dphy1/csi2_dphy2 mutually exclusive, not at the same time;
- Data lanes up to 4 lanes;
- Maximum speed 2.5Gbps/lane;

Split mode

- Use only csi2_dphy1 and csi2_dphy2, mutually exclusive with csi2_dphy0, not at the same time;
- csi2_dphy1 and csi2_dphy2 can be used at the same time;
- The maximum data lanes of csi2_dphy1 and csi2_dphy2 are 2 lanes;
- csi2_dphy1 lane0/lane1 corresponding to physical DPY;
- csi2_dphy2 Lane2/LANE3 corresponding to physical DPY;
- Maximum rate 2.5Gbps/lane

For specific DTS use cases, see the following examples.

7.1.1.1 Link ISP

7.1.1.1.1 *RV1126/RV1106*

The following is an example of RV1126 ISP and OS04A10.

Link relationship: sensor->csi_dphy->isp->ispp

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

Configuration Essentials

- Data-lanes must indicate the specific number of lanes used, otherwise they cannot be recognized as mipi types;

```
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";// It needs to match the matching string in the
driver
    reg = <0x36>;// sensor I2C device address, 7 bits
    clocks = <&cru CLK_MIPICSI_OUT>;// Sensor clickin configuration
    clock-names = "xvclk";
    power-domains = <&power RV1126_PD_VI>;
    pinctrl-names = "rockchip,camera_default";
    pinctrl-0 = <&mipi_csi_clk0>;// PINCTL settings
    //power supply
    avdd-supply = <&vcc_avdd>;
    dovdd-supply = <&vcc_dovdd>;
    dvdd-supply = <&vcc_dvdd>;
    // Power pin assignment and active level
    pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
    // Module number, which should not be repeated
    rockchip,camera-module-index = <1>;
    // Module orientation, with "back" and "front"
    rockchip,camera-module-facing = "front";
    // Module name
    rockchip,camera-module-name = "CMK-OT1607-FV1";
    // Lens name
    rockchip,camera-module-lens-name = "M12-4IR-4MP-F16";
    //IR CUT equipment
    ir-cut = <&cam_ircut0>;
    port {
        ucam_out0: endpoint {
            // The port name on the mipi dphy side
            remote-endpoint = <&mipi_in_ucam0>;
            // MIPI lanes number, 1 lane is <1>,
            4lane is <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>;
                // The port name on the sensor side
                remote-endpoint = <&ucam_out0>;
```

```
                // MIPI lanes number, 1 lane is <1>, 4lane is
<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>;
                // The port name on the ISP side
                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>;
                // The port name on the mipi dphy side
                remote-endpoint = <&csidphy0_out>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;
            isp0_out: endpoint@1 {
                reg = <1>;
                // ISP port name, ISP output to ISP
                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 port name, ISPP input
            remote-endpoint = <&isp0_out>;
        };
    };
};
```

7.1.1.1.2 *RK356X*

The following is an example of RK3566 ISP and GC8034 4LANE:

Link relationship: sensor->csi2_dphy0->isp

Configuration Essentials

- Data-lanes need to be configured
- Requires csi2_dphy_hw node to be enabled

```
/* full mode: lane0-3 */
   gc8034: gc8034@37 {
        // It needs to match the matching string in the driver
        compatible = "galaxycore,gc8034";
        status = "okay";
        // sensor I2C device address, 7 bits
        reg = <0x37>;
        // Sensor mclk source configuration
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        //The sensor dependent power domain is enabled
        power-domains = <&power RK3568_PD_VI>;
        //Sensor McLk pinctl settings
        pinctrl-names = "default";
        pinctrl-0 = <&cif_clk>;
        // Reset pin assignment and active level
        reset-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_LOW>;
        // Powerdown pin assignment and active level
        pwdn-gpios = <&gpio4 RK_PB2 GPIO_ACTIVE_LOW>;
        // Module number, which should not be repeated
        rockchip,camera-module-index = <0>;
        // Module orientation, with "back" and "front"
        rockchip,camera-module-facing = "back";
        // Module name
        rockchip,camera-module-name = "RK-CMK-8M-2-v1";
        // Lens name
        rockchip,camera-module-lens-name = "CK8401";
        port {
            gc8034_out: endpoint {
                // The port name of the CSI2 dphy end
                remote-endpoint = <&dphy0_in>;
                // CSI2 dphy lane number, 1lane is <1>,
                4lane is <1 2 3 4>, data-lanes = <1 2 3 4>;
            };
        };
```

```
    };

    &csi2_dphy_hw {
       status = "okay";
    };

    &csi2_dphy0 {
        //csi2_dphy0 is not used at the same time as
        csi2_dphy1/csi2_dphy2, mutually exclusive
        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>;
                    // The port name on the sensor side
                    remote-endpoint = <&gc8034_out>;
                    // CSI2 dphy lanes, 1 lane is <1 >, 4 lanes
is <1 2 3 4 >, which needs to be                              consistent with the
sensor side
                        data-lanes = <1 2 3 4>;
                };
            };

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

                dphy0_out: endpoint@1 {
                    reg = <1>;
                    // The port name on the ISP side
                    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>;
            // The port name of the CSI2 dphy side
            remote-endpoint = <&dphy0_out>;
        };
    };
};
```

7.1.1.2 Link VICAP

7.1.1.2.1 *RV1126/RV1109*

Take MIPI OS04A10 4 Lanes Link VICAP as an example:

Link relationship: sensor->csi dphy->mipi csi host->vicap

Configuration Points:

- Data-lanes must indicate the specific number of lanes used, otherwise they cannot be recognized as MIPI types;
- dphy needs to be linked to the CSI Host node.

```
os04a10: os04a10@36 {
        // It needs to match the matching string in the driver
        compatible = "ovti,os04a10";
        // sensor I2C device address, 7 bits
        reg = <0x36>;
        // Sensor mclk source configuration
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        //The sensor dependent power domain is enabled
        power-domains = <&power RV1126_PD_VI>;
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        //Sensor McLk pinctl settings
        pinctrl-names = "rockchip,camera_default";
        pinctrl-0 = <&mipicsi_clk0>;
        // Powerdown pin assignment and active level
        pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
        // Module number, which should not be repeated
        rockchip,camera-module-index = <1>;
        // Module orientation, with "back" and "front"
        rockchip,camera-module-facing = "front";
        // Module name
        rockchip,camera-module-name = "CMK-OT1607-FV1";
        // Lens name
```

```
        rockchip,camera-module-lens-name = "M12-40IRC-4MP-F16";
        // Ircut name
        ir-cut = <&cam_ircut0>;
        port {
            ucam_out0: endpoint {
                // The port name of the CSI2 dphy end
                remote-endpoint = <&mipi_in_ucam0>;
                // CSI2 dphy lanes number, 1lane is <1>,
4lane is <1 2 3 4 >
                data-lanes = <1 2 3 4>;
            };
        };
    };

&csi_dphy0 {
    //csi2_dphy0 is not used at the same time as csi2_dphy1/csi2_dphy2, mutually
exclusive
    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>;
                // The port name on the sensor side
                remote-endpoint = <&ucam_out0>;
                // CSI2 dphy lanes, 1 lane is <1 >, 4 lanes is <1 2 3 4 >, which
needs to be consistent with the sensor side
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                // The port name of the CSI2 host
                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>;
                // The port name of the CSI2 host
                remote-endpoint = <&csidphy0_out>;
                // CSI2 host lanes, 1 lane is <1>,
                4lane is <1 2 3 4 >, which needs to be
                consistent with the sensor side
                data-lanes = <1 2 3 4>;
            };
        };

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

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                // The port name on the vicap side
                remote-endpoint = <&cif_mipi_in>;
                // CSI2 host lanes, 1 lane is <1>,
                4lane is <1 2 3 4 >, which needs to be
                consistent with the sensor side
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        /* MIPI CSI-2 endpoint */
        cif_mipi_in: endpoint {
            // The port name of the CSI2 host
            remote-endpoint = <&mipi_csi2_output>;
            // The number of lanes on the vicap side,
            1 lane is <1 >, 4lane is < 1 2 3 4 >,
            which needs to be consistent with the sensor side
            data-lanes = <1 2 3 4>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        /* sditf endpoint */
```

```
        mipi_lvds_sditf: endpoint {
            //ISP virtual device port name
            remote-endpoint = <&isp_in>;
            //The number of lanes of MIPI CSI2 dphy,
            consistent with the sensor
            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>;
                //The endpoint name of the Vicap SDITF
                remote-endpoint = <&mipi_lvds_sditf>;
            };
        };
    };
};
```

7.1.1.2.2 *RK356X*

Take the example of lane2/lane3 of GC5025 2lane link rk3566 EVB2 mipi CSI2 dphy:

Link relationship: sensor->csi2 dphy->mipi CSI host->vicap

Configuration Essentials

- Data-lanes must indicate the specific number of lanes used, otherwise they cannot be recognized as MIPI types;
- dphy needs to be linked to the CSI Host node;
- The CSI2 dphy HW node needs to be enabled.

```
/* split mode: lane:2/3 */
gc5025: gc5025@37 {
        status = "okay";
        // It needs to match the matching string in the driver
        compatible = "galaxycore,gc5025";
        // sensor I2C device address, 7 bits
        reg = <0x37>;
        // Sensor mclk source configuration
        clocks = <&pmucru CLK_WIFI>;
        clock-names = "xvclk";
```

```
        //Sensor McLk pinctl settings
        pinctrl-names = "default";
        pinctrl-0 = <&refclk_pins>;
        // Reset pin assignment and active level
        reset-gpios = <&gpio3 RK_PA5 GPIO_ACTIVE_LOW>;
        // Powerdown pin assignment and active level
        pwdn-gpios = <&gpio3 RK_PB0 GPIO_ACTIVE_LOW>;
        // The sensor dependent power domain is enabled
        power-domains = <&power RK3568_PD_VI>;
        /*power-gpios = <&gpio0 RK_PC1 GPIO_ACTIVE_HIGH>;*/
        // Module number, which should not be repeated
        rockchip,camera-module-index = <1>;
        // Module orientation, with "back" and "front"
        rockchip,camera-module-facing = "front";
        // Module name
        rockchip,camera-module-name = "TongJu";
        // Lens name
        rockchip,camera-module-lens-name = "CHT842-MD";
        port {
            gc5025_out: endpoint {
                // The port name of the CSI2 dphy end
                remote-endpoint = <&dphy2_in>;
                // CSI2 dphy lanes number, 2lanes are <1 2>, 4lanes are <1 2 3 4
>
                data-lanes = <1 2>;
            };
        };
};

 &csi2_dphy_hw {
         status = "okay";
 };

&csi2_dphy2 {
    //csi2_dphy2 is not used at the same time as csi2_dphy0, mutually exclusive;
Can be used in parallel with csi2_dphy1
    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>;
                // The port name on the sensor side
                remote-endpoint = <&gc5025_out>;
```

```
                // CSI2 dphy lanes, 2lane is <1 2>, 4lane is <1 2 3 4 >, which
needs to be consistent with the sensor side
                data-lanes = <1 2>;
            };
        };

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

            dphy2_out: endpoint@1 {
                reg = <1>;
                // The port name of the CSI2 host
                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>;
                // The port name on the CSI2 dphy side
                remote-endpoint = <&dphy2_out>;
                // CSI2 host lanes, 2 lanes are <1 2>, 4lanes are <1 2 3 4 >,
which need to be consistent with the sensor side
                data-lanes = <1 2>;
            };
        };

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

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                // The port name on the vicap side
                remote-endpoint = <&cif_mipi_in>;
                // CSI2 host lanes, 1 lane is <1>, 4lane is <1 2 3 4 >, which
needs to be consistent with the sensor side
                data-lanes = <1 2>;
            };
        };
```

```
    };
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        cif_mipi_in: endpoint {
            // The port name of the CSI2 host
            remote-endpoint = <&mipi_csi2_output>;
            // The number of lanes on the vicap side, 2lane is <1 2>, 4lane is <1
2 3 4 >, which needs to be consistent with the sensor side
            data-lanes = <1 2>;
        };
    };
};

&rkcif_mipi_lvds_sditf {
    status = "okay";

    port {
        /* MIPI CSI-2 endpoint */
        mipi_lvds_sditf: endpoint {
            //ISP virtual device port name
            remote-endpoint = <&isp_in>;
            //The number of lanes of MIPI CSI2 dphy, consistent with the sensor
            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>;
                //The endpoint name of Vicap Mipi SDITF
                remote-endpoint = <&mipi_lvds_sditf>;
            };
        };
    };
};
```

7.1.1.2.3 *RK3588*

Take IMX464 connected to DPY1 as an example

- Data-lanes must indicate the specific number of lanes used, otherwise they cannot be recognized as mipi types;
- dphy needs to be linked to the CSI Host node, csi2_dphy3 corresponds to the mipi4_csi2 used;
- csi2_dphy3 are just logical nodes and need to rely on physical nodes csi2_dphy1_hw.
- rkcif_mipi_lvds4 is one of the logical nodes of vicap, and the physical node rkcif and corresponding iommu need to be configured.
- rkcif_mipi_lvds4_sditf is a virtual child node, which is a rkcif_mipi_lvds4 virtual node that is used to link ISPs.
- The sensor driver generally realizes AVDD/DVDD/DOVDD three power supply operation, if the power supply similar to the RAK809 allocated can be directly configured on the sensor node, if the LDO and other external power supplies, the enabling pin is controlled by GPIO, you can refer to the vcc_mipicsi1 configured as a power node, can be powered up and down by reference counting, suitable for multiple devices using the same power supply. It is recommended that the DVD power supply is supplied separately when multi-camera, AVDD/DOVDD can be shared, and when the DVD is shared, if the power is relatively large, there may be an instantaneous shortage of supply, and the power supply will collapse, affecting the image quality, or even not drawing.

```
/ {
    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>;
        };
    };
};
```

7.1.1.2.4 *RV1106*

DTS configuration refer to arch/arm/boot/dts/rv1106-evb-cam.dtsi

7.1.2 LVDS interface

7.1.2.1 Link VICAP

7.1.2.1.1 *RV1126/RV1109*

Taking IMX327 4lane as an example, the link relationship is as follows:

Link relationship: sensor->csi dphy->vicap

Configuration Essentials

- dphy does not need to link the CSI Host node, otherwise it will not receive data;

- Data-lanes must indicate the specific number of lanes used, otherwise data will not be received;
- bus-type must be configured as 3, otherwise it cannot be recognized as an LVDS interface, resulting in link establishment failure;

```
imx327: imx327@1a {
        // It needs to match the matching string in the driver
        compatible = "sony,imx327";
        // sensor I2C device address, 7 bits
        reg = <0x1a>;
        // Sensor mclk source configuration
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        //The sensor dependent power domain is enabled
        power-domains = <&power RV1126_PD_VI>;
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        //Sensor McLk pinctl settings
        pinctrl-names = "default";
        pinctrl-0 = <&mipicsi_clk0>;
        // Powerdown pin assignment and active level
        pwdn-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_HIGH>;
        // Reset pin assignment and active level
        reset-gpios = <&gpio1 RK_PD5 GPIO_ACTIVE_HIGH>;
        // Module number, which should not be repeated
        rockchip,camera-module-index = <1>;
        // Module orientation, with "back" and "front"
        rockchip,camera-module-facing = "front";
        // Module name
        rockchip,camera-module-name = "CMK-OT1607-FV1";
        // Lens name
        rockchip,camera-module-lens-name = "M12-4IR-4MP-F16";
        // Ircut name
        ir-cut = <&cam_ircut0>;
        port {
            ucam_out0: endpoint {
                // The port name of the CSI2 dphy end
                remote-endpoint = <&mipi_in_ucam0>;
                //CSI2 dphy lvds number of lanes, 1 lane is <1>, 4lane is <4 >,
must be specified
                data-lanes = <4>;
                //The type of LVDS interface, which must be specified
                bus-type = <3>;
            };
        };
};

&csi_dphy0 {
    //csi2_dphy0 is not used at the same time as csi2_dphy1/csi2_dphy2, mutually
exclusive
    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>;
                // The port name on the sensor side
                remote-endpoint = <&ucam_out0>;
                //CSI2 dphy lvds number of lanes, 1 lane is <1>, 4lane is <4 >,
must be specified
                data-lanes = <4>;
                //The type of LVDS interface, which must be specified
                bus-type = <3>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                // The port name of the vicap lite side
                remote-endpoint = <&cif_lite_lvds_in>;
                // CSI2 dphy lvds number of lanes, 1 lane is <1>, 4lane is <4 >,
must be specified
                data-lanes = <4>;
                // The type of LVDS interface, which must be specified
                bus-type = <3>;
            };
        };
    };
};

&rkcif_lite_mipi_lvds {
    status = "okay";

    port {
        /* lvds endpoint */
        cif_lite_lvds_in: endpoint {
            // The port name of the CSI2 dphy end
            remote-endpoint = <&csidphy0_out>;
            // CSI2 dphy lvds number of lanes, 1 lane is <1>, 4lane is <4 >, must
be specified
            data-lanes = <4>;
            //The type of LVDS interface, which must be specified
            bus-type = <3>;
        };
    };
};

&rkcif_lite_sditf {
    status = "okay";

    port {
```

```
        /* lvds endpoint */
        lite_sditf: endpoint {
            //ISP virtual device port name
            remote-endpoint = <&isp_in>;
            //CSI2 dphy has the number of lanes, which is the same as the sensor
            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>;
                //The endpoint name of the lite vicap lvds sditf
                remote-endpoint = <&lite_sditf>;
            };
        };
    };
};
```

If you need to use dual ISP processing, you can modify it as follows:

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

&isp0_mmu {
    status = "disabled";
};

&rkisp_unite {
    status = "okay";
};

&rkisp_unite_mmu {
    status = "okay";
};

&rkisp0_vir0 {
```

```
    status = "okay";

    /* hw is referenced as a unite node*/
    rockchip,hw = <&rkisp_unite>;

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

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

7.1.3 DVP interface

7.1.3.1 Link VICAP

On the RV1126/RV1109/RK356X/RK3588 platforms, the dts configuration of each related interface of DVP is the same.

7.1.3.1.1 BT601

Taking AR0230 BT601 as an example, the link relationship is as follows:

Link relationship: sensor->vicap

Configuration Essentials

- hsync-active/vsync-active must be configured for the v4l2 framework to register and identify the BT601 interface asynchronously, if not configured, it will be recognized as the BT656 interface;
- pclk-sample/bus-width optional;
- The valid polarity of the current sensor's hsync-acitve/vsync-active/pclk-ative must be indicated by flag in the sensor-driven g_mbus_config interface, otherwise the data will not be received;
- Pinctrl needs to reference pairs to make the corresponding IOMUX for BT601-related GPIOs, otherwise it will cause data to not be received;

The sample code for the g_mbus_config interface is as follows:

```
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;
}
```

An example DTS configuration is as follows:

```
ar0230: ar0230@10 {
        // It needs to match the matching string in the driver
        compatible = "aptina,ar0230";
        // sensor I2C device address, 7 bits
        reg = <0x10>;
        // Sensor mclk source configuration
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        //The sensor dependent power domain is enabled
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        power-domains = <&power RV1126_PD_VI>;
        // Powerdown pin assignment and active level
        pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
        /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
        //Configure the DVP-related data pins and clock pins
        pinctrl-names = "default";
        pinctrl-0 = <&cifm0_dvp_ctl>;
        // Module number, which should not be repeated
        rockchip,camera-module-index = <0>;
        // Module orientation, with "back" and "front"
        rockchip,camera-module-facing = "back";
        // Module name
        rockchip,camera-module-name = "CMK-OT0836-PT2";
        // Lens name
        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 {
            //The name of the sensor endpoint
            remote-endpoint = <&cam_para_out1>;
            //Relevant configuration parameters on the sensor side
            bus-width = <12>;
            hsync-active = <1>;
            vsync-active = <1>;
            pclk-sample = <0>;
        };
    };
};

&rkcif_dvp_sditf {
    status = "okay";

    port {
```

```
        /* parallel endpoint */
        dvp_sditf: endpoint {
            //ISP virtual device port name
            remote-endpoint = <&isp_in>;
            //Relevant configuration parameters on the sensor side
            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>;
                //The endpoint name of the DVD SDITF
                remote-endpoint = <&dvp_sditf>;
            };
        };
    };
};
```

7.1.3.1.2 BT656/BT1120

The BT656/BT1120 DTS usage is consistent.

Taking the Ava FPGA BT1120 as an example, the link relationship is as follows:

*链接关系： **sensor->vicap***

配置要点

- hsync-active/vsync-active do not configure, otherwise the v4l2 framework will recognize it as BT601 when registering asynchronously;
- pclk-sample/bus-width optional;
- The effective polarity of the current sensor's pclk-ative must be indicated by the flag variable in the sensor-driven g_mbus_config interface, otherwise the data will not be received;
- The querystd interface in the v4l2_subdev_video_ops must be implemented, indicating that the current interface is an ATSC interface, otherwise the data will not be received;

- Must implement RKMODULE_GET_BT656_MBUS_INFO, BT656/BT1120 are called this ioctl, interface compatible, implementation reference drivers/media/i2c/nvp6158_drv/nvp6158_v4l2.c
- Pinctrl needs to reference pairs to make IOMUX for BT656/BT1120-related GPIOs, otherwise data cannot be received.

The sample code for the g_mbus_config interface is as follows:

```
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;
}
```

An example of the querystd interface is as follows:

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

    return 0;
}
```

An example DTS configuration is as follows:

```
avafpga: avafpga@70 {
    // It needs to match the matching string in the driver
    compatible = "ava,fpga";
    // sensor I2C device address, 7 bits
    reg = <0x10>;
    // Sensor mclk source configuration
    clocks = <&cru CLK_CIF_OUT>;
    clock-names = "xvclk";
    //The sensor dependent power domain is enabled
    avdd-supply = <&vcc_avdd>;
    dovdd-supply = <&vcc_dovdd>;
    dvdd-supply = <&vcc_dvdd>;
    // Powerdown pin assignment and active level
    power-domains = <&power RV1126_PD_VI>;
    pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
    /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
    //Configure the DVP-related data pins and clock pins
    pinctrl-names = "default";
    pinctrl-0 = <&cifm0_dvp_ctl>;
    // Module number, which should not be repeated
    rockchip,camera-module-index = <0>;
    // Module orientation, with "back" and "front"
    rockchip,camera-module-facing = "back";
    // Module name
    rockchip,camera-module-name = "CMK-OT0836-PT2";
    // Lens name
    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 {
            //The name of the sensor endpoint
            remote-endpoint = <&cam_para_out2>;
            //Sensor-side related configuration parameters, optional
            bus-width = <16>;
            pclk-sample = <1>;
        };
    };
};

&rkcif_dvp_sditf {
    status = "okay";

    port {
        /* parallel endpoint */
        dvp_sditf: endpoint {
            //ISP virtual device port name
            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>;
                //The endpoint name of the DVD SDITF
                remote-endpoint = <&dvp_sditf>;
            };
        };
    };
```

```
};
```

7.1.3.1.3 RV1106 DVP DTS Notes

Refer to the previous configuration description, basically unchanged, pay attention to MCLK and other hardware resources according to the actual hardware link configuration.

The RV1106 supports PINCTRLs for both DVPs, and the correct PINCTRL needs to be referenced in the DTS based on the hardware usage.
M0:

Support BT1120, 16bit data
Support BT656, 8bit data
Support BT601, 8/10/12bit data
This set of pins exists and mipi dphy pin multiplexing, mipi dphy working state to be configured in TTL mode, so rkcif_dvp node needs to reference mipi dphy node, refer to the following:

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

    rockchip,dphy_hw = <&csi2_dphy_hw>;

        ......
};
```

M1:
Support BT656, 8bit data
Support BT601, 8/10 data
Can be used at the same time as MIPI.

Note: At present, the DVD data of the RK platform is highly aligned, and you must read the RK hardware design reference guide in detail when designing the hardware to prevent the hardware from being not designed as required and cannot be collected.

7.1.4 Multi-sensor registration

A single hardware ISP processes multiplexed multiple raw sensor data by virtualizing multiple devices.

For RV1109/RV1126/RK356X VICAP to collect DVP RAW data can only be stored as non-compact, ISP processing RAW is based on compact storage by default, for the precautions of DVP RAW data into ISP processing, please refer to如何配置ISP/VICAP RAW存储格式.

7.1.4.1 RV1126/RV1109

The link relationship, isp0->ispp0 and isp1->ispp1 is the fixed configuration rv1126.dtsi

RV1109/RV1126 ISP/ISPP, up to 4-way multiplexing when bandwidth permits, can be added by rkisp_vir0~rkisp_vir4/rkispp_vir0~rkispp_vir4 in RV1126.DTSI

MIPI into ISP or CIF into ISP is optional.

- RV1109/RV1126 supports 2 PHY interfaces, each PHY can be multiplexed as MIPI/LVDS, and supports up to 4LANE.
- RV1109/RV1126 supports one DVP interface and supports BT601/BT656/BT1120.
- ISP supports MIPI or DVP input: MIPI/DVP can only choose 1 of 2 and cannot work at the same time.
- VICAP supports MIPI/LVDS/DVP: MIPI/LVDS is a multiplexing relationship and cannot be used at the same time, and DVP can be used at the same time as the former.
- Vicap Lite only supports LVDS.
- Through the understanding of the above hardware configuration, RV1109/RV1126 currently supports up to 3 RAW SENSOR input ISP processing.

7.1.4.1.1 Dual-entry ISP processing

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

Example reference: arch/arm/boot/dts/rv1109-evb-ddr3-v12-facial-gate.dts
gc2053->csi_dphy0->csi2->vicap->isp1->ispp1
ov2718->csi_dphy1->isp0->ispp0
For different resolutions, it is important to configure the following

```
&rkispp {
   status = "okay";
   /* the max input w h and fps of mulit sensor */
   max-input = <2688 1520 30>;// Maximum width and height and frame rate for different sensors
};
```

7.1.4.1.2 Three-shot ISP treatment

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

Example reference:

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";
};
```

7.1.4.2 RK3566/RK3568

RK356X ISP, up to 4 multiplexed when bandwidth allows, can be added by rkisp_vir0~rkisp_vir4 in RK3568.DTSI

MIPI into ISP or CIF into ISP is optional.

The RK356X supports 1 4-lane PHY interface, which can be divided into 2 2-lane PHYs.
The rk356x supports one DVD interface and supports BT601/BT656/BT1120
ISP supports MIPI or DVP input: MIPI/DVP can only choose 1 of 2 and cannot work at the same time.
VICAP supports MIPI/DVP: MIPI and DVP can be used at the same time.
Through the understanding of the above hardware configuration, the RK356X currently supports up to 3 RAW SENSORS into ISP processing.

7.1.4.2.1 Dual Intake ISP Processing:

Reference examples:

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>;
            };
        };
    };
};
```

7.1.4.2.2 Three-shot ISP Processing:

Reference examples:

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>;
        };
    };
};
```

7.1.4.3 RK3588

1. Refer to RK3588多sensor支持
2. Refer to arch/arm64/boot/dts/rockchip/rk3588-evb1-cam-6x.dtsi
3. For dual ISP unite mode

   1）Support dual camera, maximum resolution 7616x2160

   2）Support 3/4 camera, maximum resolution 5056x1536

The DTS configuration is described in the above unite mode single-camera configuration description, and then configured as follows: ISP nodes

&rkisp0_vir1 {

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

  //Other omissions

};

&rkisp0_vir2 {

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

  //Other omissions

};

7.1.4.4 RV1106

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

Hardware ISP support dual camera, the maximum resolution is 1080p, if it is larger than this resolution, it needs to be processed by a single frame 2 readback, the disadvantages consume more ISP throughput, bandwidth increases, and the output frame rate is low.

7.2 CIS driver description

Camera sensor uses I2C to interact with the main control, and the sensor driver is currently implemented in accordance with the I2C device driver mode, and the sensor driver also uses the V4l2 subdev method to realize the interaction with the host driver.

7.2.1 A brief description of the data type

7.2.1.1 struct i2c_driver

[Description]

Define I2C device driver information

[Definition]

```
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;
    ……
};
```

[Key Member]

Member name	description
@driver	The Device driver model driver mainly contains the driver name and the of_match_table that matches the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the DTS file
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
#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);
```

7.2.1.2 struct v4l2_subdev_ops

[Description]

Define ops callbacks for subdevs.

[Definition]

```
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;
};
```

[Key Member]

Member name	description
.core	Define core ops callbacks for subdevs
.video	Callbacks used when v4l device was opened in video mode.
.pad	v4l2-subdev pad level operations

[Example]

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

7.2.1.3 struct v4l2_subdev_core_ops

[Description]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	description
.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.

[Example]

```
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
};
```

Currently, the following private IOCTL is used to query module information and set up OTP information.

Private ioctl	description
RKMODULE_GET_MODULE_INFO	For module information, refer to struct rkmodule_inf;
RKMODULE_AWB_CFG	The compensation function of the switch sensor to AWB; If the module does not flash the golden awb value, you can set this setting; Refer to struct rkmodule_awb_cfg;
RKMODULE_LSC_CFG	Switch sensor compensation function for LSC; Detailed refer tostruct rkmodule_lsc_cfg;
PREISP_CMD_SET_HDRAE_EXP	HDR exposure settings are detailed refer tostruct preisp_hdrae_exp_s
RKMODULE_SET_HDR_CFG	Setting HDR mode can switch between normal and HDR mode, and you need to refer to the configuration information of the driver adaptation normal and HDR 2 groups for detailed refer tostruct rkmodule_hdr_cfg
RKMODULE_GET_HDR_CFG	Get a detailed reference to the current HDR mode struct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	Set the conversion gain of linear mode, such as IMX347, OS04A10 sensor with conversion gain function, such as the sensor does not support conversion gain, can not be implemented

7.2.1.4 struct v4l2_subdev_video_ops

[Description]

Callbacks used when v4l device was opened in video mode.

[Definition]

```
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);
 ……
};
```

[Key Member]

Member name	description
.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

[Example]

```
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,
};
```

7.2.1.5 struct v4l2_subdev_pad_ops

[Description]

v4l2-subdev pad level operations

[Definition]

```
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);
    ……
};
```

[Key Member]

Member name	description
. 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.

[Example]

```
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,
};
```

7.2.1.6 struct v4l2_ctrl_ops

[Description]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	description
.s_ctrl	actually set the new control value.

[Example]

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

The RKISP driver requires the use of the user controls function provided by the framework, and the cameras sensor driver must implement the following control functions,refer to:CIS驱动V4L2-controls列表1

7.2.1.7 struct xxxx_mode

[Description]

The Sensor can support information in each mode.

This structure is often seen in sensor drivers, although it is not required by the V4L2 standard. As functionality increases, the structure can add variables according to demand.

[Definition]

```
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];
};
```

[Key Member]

Member name	description
.bus_fmt	Sensor output format, refer toMEDIA_BUS_FMT 表
.width	The effective image width needs to be consistent with the width output currently configured by the sensor
.height	The effective image height needs to be consistent with the HEIGHT output currently configured by the sensor
.max_fps	Image FPS, denominator/numerator is fps
hts_def	The default HTS is effective image width + HBLANK
vts_def	The default VTS is valid image height + VBLANK
exp_def	Default exposure time
*reg_list	List of registers
.hdr_mode	Sensor working mode, support linear mode, two-frame composite HDR, three-frame synthetic HDR
.vc[PAD_MAX]	Configure the MIPI VC channel

[Example]

```
 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
    },
};
```

7.2.1.8 struct v4l2_mbus_framefmt

[Description]

frame format on the media bus

[Definition]

```
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];
};
```

[Key Member]

Member name	description
width	Frame width
height	Frame height
code	Refer toMEDIA_BUS_FMT 表
field	V4L2_FIELD_NONE: Frame output mode V4L2_FIELD_INTERLACED: Field output mode

[Example]

7.2.1.9 struct rkmodule_base_inf

[Description]

The basic information of the module, the upper layer uses this information to match with IQ

[Definition]

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

[Key Member]

Member name	description
sensor	The name of the sensor, obtained from the sensor driver
module	The module name is obtained from the DTS configuration, and the module information shall prevail
lens	The lens name, obtained from the DTS configuration, is subject to the module data

[Example]

7.2.1.10 struct rkmodule_fac_inf

[Description]

Module OTP factory information

[Definition]

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

[Key Member]

Member name	description
flag	The identity of whether the group information is valid
module	Module name, get the number from the OTP, and get the module name from the number
lens	Lens name, get the number from the OTP, and get the lens name from the number
year	Year of production, such as 12 for 2012
month	Production month
day	Production date

[Example]

7.2.1.11 struct rkmodule_awb_inf

[Description]

Module OTP awb measurement information

[Definition]

```
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));
```

[Key Member]

Member name	description
flag	The identity of whether the group information is valid
r_value	AWB R measurement information for the current module
b_value	AWB B measurement information for the current module
gr_value	AWB GR measurement information for the current module
gb_value	AWB GB measurement information for the current module
golden_r_value	The AWB R measurement information of a typical module, if not burned, is set to 0
golden_b_value	The AWB B measurement information of a typical module, if not burned, is set to 0
golden_gr_value	The AWB GR measurement information of a typical module, if not burned, is set to 0
golden_gb_value	The AWB GB measurement information of a typical module, if not burned, is set to 0

[Example]

7.2.1.12 struct rkmodule_lsc_inf

[Description]

Module OTP lsc measurement information

[Definition]

```
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));
```

[Key Member]

Member name	description
flag	The identity of whether the group information is valid
lsc_w	The actual width of the LSC table
lsc_h	LSC table actual height
decimal_bits	The number of decimal places in the LSC measurement information is set to 0 if it cannot be obtained
lsc_r	LSC R assay information
lsc_b	LSC B assay information
lsc_gr	LSC GR assay information
lsc_gb	LSC GB assay information

[Example]

7.2.1.13 struct rkmodule_af_inf

[Description]

Module OTP af measurement information

[Definition]

```
struct rkmodule_af_inf {
    __u32 flag; // The identity of whether the group information is valid
    __u32 vcm_start; // VCM start-up current
    __u32 vcm_end; // VCM terminates the current
    __u32 vcm_dir; // VCM determination direction
} __attribute__ ((packed));
```

[Key Member]

Member name	description
flag	The identity of whether the group information is valid
vcm_start	VCM start-up current
vcm_end	VCM terminates the current
vcm_dir	VCM determination direction

[Example]

7.2.1.14 struct rkmodule_inf

[Description]

Module information

[Definition]

```
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));
```

[Key Member]

Member name	description
base	Basic module information
fac	Module OTP factory information
awb	Module OTP awb measurement information
lsc	Module OTP lsc measurement information
af	Module OTP af measurement information

[Example]

7.2.1.15 struct rkmodule_awb_cfg

[Description]

Module OTP awb configuration information

[Definition]

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

[Key Member]

Member name	description
enable	Identifies whether AWB correction is enabled
golden_r_value	AWB R measurement information for a typical module
golden_b_value	AWB B determination information for a typical module
golden_gr_value	Information on AWB GR determination of a typical module
golden_gb_value	AWB GB measurement information for typical modules

[Example]

7.2.1.16 struct rkmodule_lsc_cfg

[Description]

Module OTP lsc configuration information

[Definition]

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

[Key Member]

Member name	Description
enable	Identifies whether LSC correction is enabled

[Example]

7.2.1.17 struct rkmodule_hdr_cfg

[Description]

HDR configuration information

[Definition]

```
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;
};
```

[Key Member]

Member name	Description

hdr_mode	NO_HDR=0 //normal mode
	HDR_X2=5 //hdr 2 frame mode
	HDR_X3=6 //hdr 3 frame mode

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)

[Example]

7.2.1.18 struct preisp_hdrae_exp_s

[Description]

HDR exposure parameters

[Definition]

```
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;
};
```

[Key Member]

Member name	Description
long_exp_reg	Long frame exposure register value
long_gain_reg	Long frame gain register value
middle_exp_reg	Medium frame exposure register value
middle_gain_reg;	Medium frame gain register value
short_exp_reg	Short frame exposure register value
short_gain_reg	Short frame gain register value
long_cg_mode	Long frame conversion gain，0 LCG，1 HCG
middle_cg_mode	Medium frame conversion gain，0 LCG，1 HCG
short_cg_mode	Short frame conversion gain，0 LCG，1 HCG

[Description]

preisp_hdrae_exp_s only need to pay attention to a few parameters described by [key members] in the structure, the formula for converting exposure and gain values into registers is in IQ XML, please refer to IQ XML format description for specific conversion, conversion gain requires the sensor itself to support this function, if it is not supported, there is no need to pay attention to conversion parameters, **HDR2X should be passed down the medium frame, The short frame parameter sets the exposure parameter register corresponding to the output two frames of the sensor.**

[Example]

7.2.2 API简要说明

7.2.2.1 xxxx_set_fmt

[Description]

Set the sensor output format.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
*cfg	subdev pad information struct pointer	input
*fmt	Pad-level media bus format struct pointer	input

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.2 xxxx_get_fmt

[Description]

Get the sensor output format.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
*cfg	subdev pad information struct pointer	input
*fmt	Pad-level media bus format struct pointer	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

refer to MEDIA_BUS_FMT 表

7.2.2.3 xxxx_enum_mbus_code

[Description]

Enumerate sensor output bus format.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
*cfg	subdev pad information struct pointer	input
*fmt	media bus format enumeration struct pointer	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

The following table summarizes the formats corresponding to each image type, refer toMEDIA_BUS_FMT 表

7.2.2.4 xxxx_enum_frame_sizes

[Description]

Enumerate the sensor output size.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
*cfg	subdev pad information struct pointer	input
*fmt	media bus format size struct pointer	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.5 xxxx_g_frame_interval

[Description]

Get the sensor output fps.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
*fi	pad-level frame rate struct pointer	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.6 xxxx_s_stream

[Description]

Set the stream input and output.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
on	1: Start the stream output; 0: Stop stream output	input

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.7 xxxx_runtime_resume

[Description]

The callback function when the sensor is powered on.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*dev	device struct pointer	input

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.8 xxxx_runtime_suspend

[Description]

Callback function when the sensor is powered down.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*dev	device struct pointer	input

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.9 xxxx_set_ctrl

[Description]

Set the values of each control.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*ctrl	v4l2_ctrl struct pointer	input

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.10 xxx_enum_frame_interval

[Description]

Enumerates the frame interval parameters supported by the sensor.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	Child device instances	input
*cfg	pad configuration parameters	input
*fie	Frame interval parameters	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.11 xxxx_g_mbus_config

[Description]

Get the supported bus configuration, for example, when using mipi, when the sensor supports multiple MIPI transmission modes, you can upload parameters according to the MIPI mode currently used by the sensor.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*config	Bus configuration parameters	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.12 xxxx_get_selection

[Description]

Configure the cropping parameters, the width of the ISP input requires 16 alignment, the height 8 alignment, for the resolution of the sensor output does not meet the alignment or the sensor output resolution is not a standard resolution, this function can be implemented to crop the resolution of the input ISP.

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*sd	Child device instances	input
*cfg	pad configuration parameters	input
*fie	Crop parameters parameters	output

[Return value]

Return value	Description
0	Success
Non-0	Failure

7.2.2.13 xxxx_ioctl

[Description]

ioctl private command implementation.

[Grammar]

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

[Parameter]

Parameter name	description	Input and output
*sd	Child device instances	input
cmd	Private named values	input
*arg	The parameters passed	input/output

[Return value]

返回值	描述
0	Success
Non-0	

7.2.3 Drive the migration step

1.Implement the standard I2C sub-device driver part.

1.1 Implement the following members according to the struct i2c_driver instructions:

struct driver.name

struct driver.pm

struct driver. of_match_table

probe function

remove function

1.2 Probe function implementation details:
1). CIS device resources are obtained, mainly to parse the resources defined in DTS files, refer to [Camera Device Registration (DTS)] (#_CIS Device Registration (DTS));

1.1) RK private resource definition, named as follows rockchip, camera-module-xxx, this part of the resource will be uploaded by the driver to the user-mode camera_engine to determine the matching of IQ effect parameters;

1.2) CIS device resource definition, RK related reference drivers generally include the following items:

Member name	Description
CIS devices operate as reference clocks	The use of external independent crystal oscillator scheme does not need to be obtained, RK reference design generally uses AP output clock, this scheme needs to be obtained, generally known as xvclk
CIS devices control GPIOs	For example: Resst pin, Powerdown pin
CIS devices control the power supply	According to the actual hardware design, obtain matching software power control resources, such as GPIO, regulator

1.3) CIS device ID number check, after obtaining the necessary resources through the above steps, it is recommended that the driver read the device ID number in order to check the accuracy of the hardware, of course, this step is not necessary.

1.4) Initialization of CIS v4l2 devices and media entities;

v4l2 sub-device: v4l2_i2c_subdev_init, the RK CIS driver requires subdev to have its own device node for user-mode rk_aiq to access, and realize exposure control through this device node;

Media entity: media_entity_init

2.Refer to struct v4l2_subdev_ops to implement the V4L2 sub-device driver, which mainly implements the following three members:

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

2.1 Refer to **struct v4l2_subdev_core_ops** to implement its callback function, mainly implementing the following callbacks:

.s_power

s_power realize the power-up and down operation of the sensor, for some sensors with long register arrays, the registers of the common part can be summarized and executed after power-on, s_stream only configure the registers necessary for different resolution configurations to speed up the switching speed.

.ioctl

.compat_ioctl32

ioctl mainly implements RK private control commands, which involve:

Member name	Description
RKMODULE_GET_MODULE_INFO	The module information (module name, etc.) defined by the DTS file is uploaded to the camera_engine through this command for device matching
RKMODULE_AWB_CFG	When the module OTP information is enabled, the typical module AWB calibration value camera_engine transmitted through this command, and the CIS driver is responsible for comparing it with the current module AWB calibration value, and then generating the R/B Gain value to be set to the CIS MWB module.
RKMODULE_LSC_CFG	When the module OTP information is enabled, the LSC calibration value camera_engine controlled by this command to take effect;
PREISP_CMD_SET_HDRAE_EXP	HDR exposure settings detailed refer tostruct preisp_hdrae_exp_s
RKMODULE_SET_HDR_CFG	Setting the HDR mode can realize normal and HDR switching, and you need to refer to the configuration information of the driver adaptation HDR and normal 2 groups for detailed refer tostruct rkmodule_hdr_cfg
RKMODULE_GET_HDR_CFG	Get a detailed reference for the current HDR modestruct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	Set linear mode conversion gain, such as IMX347, OS04A10 sensor with conversion gain function, high conversion conversion gain can obtain a better signal-to-noise ratio in low illumination, such as the sensor does not support conversion gain, can not be achieved
RKMODULE_SET_QUICK_STREAM	The only configuration sensor is the stream on/off register, which is used for abnormal reset. RK3588 supports multi-camera synchronization for unified data streaming, please refer to it for details多摄同步机制
RKMODULE_GET_CHANNEL_INFO	RK3588 adds to obtain channel information, the default vicap id0~id3 corresponds to vc0~vc3, if there are special requirements, you can configure channel information through this command, such as SPD/EBD data acquisition. Prior to 3588, the chip still obtained channel information through the get_fmt.
RKMODULE_GET_SYNC_MODE、RKMODULE_SET_SYNC_MODE	RK3588 adds synchronization mode acquisition and configuration, refer to for details多摄同步机制

2.2 Refer to **struct v4l2_subdev_video_ops** to implement its callback functions, mainly implementing the following callback functions:

Member name	Description
.s_stream	The function of switching data flow, which is continuous mode for mipi clk, must be enabled in this callback function, if the data flow is opened in advance, the MIPI LP state will not be recognized
.g_frame_interval	Get frame interval parameters (frame rate)
.g_mbus_config	Get the bus configuration, for the MIPI interface, if the sensor driver supports different lane configurations or supports HDR, return the MIPI configuration in the current sensor working mode through this interface

2.3 Refer to **struct v4l2_subdev_pad_ops** to implement its callback functions, mainly implement the following callback functions:

Member name	Description
.enum_mbus_code	Enumerates the data formats supported by the current CIS driver and implements refer toxxxx_enum_mbus_code
.enum_frame_size	Enumerate the current CIS driver support resolution and implement refer toxxxx_enum_frame_sizes
.get_fmt	The RKISP driver obtains the data format of CIS output through this callback, which must be implemented; Define references for data types output by Bayer raw sensor, SOC yuv sensor, BW raw sensorMEDIA_BUS_FMT 表For support for field output methods, refer to struct v4l2_mbus_framefmtDefinition; Implementation refer toxxxx_get_fmt
.set_fmt	Set the CIS drive output data format and resolution, be sure to implement it, and implement the refer toxxxx_set_fmt
.enum_frame_interval	Enumerates the frame intervals supported by the sensor, including resolution. Implementation refer toxxx_enum_frame_interval
.get_selection	To configure the cropping parameters, the width input by the ISP requires 16 alignment and height 8 alignment. Implementation refer toxxxx_get_selection

2.4 Refer to **struct v4l2_ctrl_ops** to describe the implementation, mainly implement the following callbacks

Member name	Description
.s_ctrl	RKISP driver, camera_engine to achieve CIS exposure control by setting different commands;

Refer to[CIS驱动V4L2-controls列表1](Implement each control ID, of which the following ID belongs to the information acquisition class, and this part of the implementation is implemented in accordance with the standard integer menu controls;

Member name	Description
V4L2_CID_LINK_FREQ	Refer toCIS驱动V4L2-controls列表1The standard definition currently obtains the MIPI bus frequency according to this command;
V4L2_CID_PIXEL_RATE	For MIPI buses：pixel_rate = link_freq * 2 * nr_of_lanes / bits_per_sample
V4L2_CID_HBLANK	Refer toCIS驱动V4L2-controls列表1in the standard definition
V4L2_CID_VBLANK	Refer toCIS驱动V4L2-controls列表1in the standard definition

RK camera_engine will obtain the necessary information to calculate the exposure through the above command, and the formula involved is as follows:

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

Among them, the following IDs belong to the control class, and RK camera_engine control CIS through this class command

Member name	Description
V4L2_CID_VBLANK	Adjust VBLANK, and then adjust frame rate, exposure time max;
V4L2_CID_EXPOSURE	Set the exposure time in the number of exposure lines
V4L2_CID_ANALOGUE_GAIN	Set the exposure gain, which is actually total gain = analog gain*digital gain; Unit: Gain register value
V4L2_CID_HFLIP	Set horizontal mirroring, ISP does not have mirroring function, if mirroring is required, use the mirroring mechanism of sensor
V4L2_CID_VFLIP	Set vertical mirroring, ISP does not have mirroring function, if mirroring is required, use the mirroring mechanism of sensor
V4L2_CID_TEST_PATTERN	Implement the test pattern, which is optional, available for debugging, as needed

2.5 HDR driver implementation considerations:

2.5.1 Linear mode exposure and gain are realized through the standard v4l2 command, when switching to HDR mode, parameters are passed through the ioctl command PREISP_CMD_SET_HDRAE_EXP, and the V4L2_CID_EXPOSURE and V4L2_CID_ANALOGUE_GAIN in the standard v4l2 command are not used. PREISP_CMD_SET_HDRAE_EXP implementation needs to be able to cache parameters, after the start_stream write the initialization array, the initial exposure needs to be written back into the sensor, overriding the default value in the initialization array, so that the brightness of the first frame output by the data stream can match the exposure parameters, so that AE convergence can be completed as soon as possible. It should be noted that the register that opens the data stream cannot be configured in the initialization array, and needs to be configured

before the start_stream is returned, also to ensure that the brightness of the first frame of the output matches the initialization exposure parameters.

2.5.2 HDR/LINEAR interaction between the sensor and the controller, through RKMODULE_SET_HDR_CFG/RKMODULE_GET_HDR_CFG, the HDR sensor must be implemented in order to switch modes normally.

3. The CIS driver does not involve the definition of hardware data interface information, and the interface connection relationship between CIS device and AP is reflected by the port of the DTS device node, refer to the description of port information in [CIS Device Registration (DTS)] (#_CIS Device Registration (DTS)).

4. CIS 参考驱动列表

8. VCM driver

8.1 VCM Device Registration (DTS)

RK VCM driver private parameter description:

Name	Description
Starting current	VCM just moves the module lens from the closest end of the module lens's movable travel (module telephoto), where the output current value of the VCM driver IC is defined as the start-up current
Rated current	VCM pushes the module lens just to the farthest end of the module lens's movable travel (module close-up), at which point the output current value of the VCM driver IC is defined as the rated current
VCM current output mode	VCM driver IC current output changes need to consider the oscillation period of VCM to minimize oscillation, and the output mode determines the time when the output current changes to the target value;

```
vm149c: vm149c@0c { // VCM driver configuration, this setting is required when
supporting AF
    compatible = "silicon touch,vm149c";
    status = "okay";
    reg = <0x0c>;
    rockchip,vcm-start-current = <0>; // The starting current of the motor
    rockchip,vcm-rated-current = <100>; // The rated current of the motor
    rockchip,vcm-step-mode = <4>; // Current output mode of the motor drive IC
    rockchip,camera-module-index = <0>; // Module number
    rockchip,camera-module-facing = "back"; // Module orientation,
有"back"和"front"
};

ov13850: ov13850@10 {
    ......
    lens-focus = <&vm149c>; // VCM driver settings, which are required when
supporting AF
```

```
    ......
};
```

8.2 VCM driver description

8.2.1 A brief description of the data type

8.2.1.1 struct i2c_driver

[illustrate]

Define I2C device driver information

[Definition]

```
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;
     ……
};
```

[Key Member]

Member name	Description
@driver	The Device driver model driver mainly contains the driver name and the of_match_table that matches the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the DTS file
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
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);
```

8.2.1.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

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

Currently, the following private IOCTL is used to query motor movement time information.

RK_VIDIOC_VCM_TIMEINFO

8.2.1.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

```
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);
};
```

[Key Member]

Member name	Description
.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.

[Example]

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

vm149c_get_ctrl and vm149c_set_ctrl support the following controls

V4L2_CID_FOCUS_ABSOLUTE

8.2.2 A brief description of the API

8.2.2.1 xxxx_get_ctrl

[Description]

Gets the position of the motor to move.

[Grammar]

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

[Parameter]

Member name	Description	Input/Output
*ctrl	v4l2 control Struct pointer	Output

[Return value]

Return value	Description
0	success
Non-0	fail

8.2.2.2 xxxx_set_ctrl

[Description]

Set the movement position of the motor.

[Grammar]

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

[Parameter]

Member name	Description	Input/Output
*ctrl	v4l2 control Struct pointer	Input

[Return value]

Return value	Description
0	success
Non-0	fail

8.2.2.3 xxxx_ioctl xxxx_compat_ioct

[Description]

The implementation function of custom ioctl, mainly contains to obtain the time information of motor movement,

Implemented custom RK_VIDIOC_COMPAT_VCM_TIMEINFO.

[Grammar]

```
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)
```

[Parameter]

Parameter name	Description	Input and output
*sd	v4l2 subdev Struct pointer	Input
cmd	ioctl command	Input
*arg/arg	Parameter pointer	output

[Return value]

Return value	Description
0	success
Non-0	fail

8.2.3 Drive the migration step

1.Implement the standard I2C sub-device driver part.

1.1 According to the description of **struct i2c_driver**, the following parts are mainly implemented:

struct driver.name

struct driver.pm

struct driver. of_match_table

probe function

remove function

1.2 Probe function implementation details:

1)VCM device resource acquisition, mainly to obtain DTS resources, refer to [VCM Device Registration (DTS)](#_VCM Device Registration (DTS))

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly provide device parameters and Camera device for matching.

1.2) VCM parameter definition, naming methods such as rockchip, VCM-xxx, mainly involving hardware parameters starting current, rated current, mobile mode, parameters related to the range and speed of motor movement.

2)Initialization of VCM v4l2 devices and media entities.

v4l2 sub-device: v4l2_i2c_subdev_init, the RK VCM driver requires subdev to have its own device node for user-mode camera_engine to access, and realize focus control through this device node;

Media entities: media_entity_init;

3)The RK AF algorithm defines the position parameter of the entire movable stroke of the module lens as [0,64], and the corresponding change range of the entire movable stroke of the module lens in the VCM driving current is [starting current, rated current], and it is recommended to realize the mapping conversion relationship between these two in this function;

2.Implement the V4L2 sub-device driver, mainly implementing the following two members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to **v4l2_subdev_core_ops** to describe the implementation of callback functions, mainly implement the following callback functions:

.ioctl.compat_ioctl32

This callback mainly implements the RK private control command, which involves:

Member name	Description
RK_VIDIOC_VCM_TIMEINFO	camera_engine use this command to obtain the time required for this lens movement, and determine when the lens stops and whether the CIS frame exposure period overlaps with the lens movement time period; Lens movement time is related to lens movement distance and VCM driver IC current output mode.

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl and .s_ctrl implement the following commands with standard v4l2 control:

Member name	Description
V4L2_CID_FOCUS_ABSOLUTE	camera_engine the absolute position of the lens is set and obtained by this command, the position parameter of the entire movable stroke of the lens is defined in the RK AF algorithm as [0,64].

9. FlashLight driver

9.1 FLASHLight Device Registration (DTS)

SGM378 DTS Reference:

```
&i2c1 {
    ...
    sgm3784: sgm3784@30 {//Flash device
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "sgmicro,gsm3784";
        reg = <0x30>;
        rockchip,camera-module-index = <0>;//The flash corresponds to the camera
module number
        rockchip,camera-module-facing = "back";//The flash corresponds to the
orientation of the camera module
        enable-gpio = <&gpio2 RK_PB4 GPIO_ACTIVE_HIGH>;//enable gpio
        strobe-gpio = <&gpio1 RK_PA3 GPIO_ACTIVE_HIGH>;//Flash triggers GPIO
        status = "okay";
        sgm3784_led0: led@0 {//led0 Device information
            reg = <0x0>;//index
            led-max-microamp = <299200>;//Torch mode maximum current
            flash-max-microamp = <1122000>;//flash mode maximum current
            flash-max-timeout-us = <1600000>;//falsh Maximum time
        };
        sgm3784_led1: led@1 {//led1 Device information
            reg = <0x1>;//index
            led-max-microamp = <299200>;//Torch mode maximum current
            flash-max-microamp = <1122000>;//flash mode maximum current
            flash-max-timeout-us = <1600000>;//falsh Maximum time
        };
    };
    ...
    ov13850: ov13850@10 {
        ...
        flash-leds = <&sgm3784_led0 &sgm3784_led1>;//The flash device is hooked
up to the camera
        ...
    };
    ...
}
```

GPIO, PWM control dts reference:

```
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>;
        ...
    }
}
```

Note:

1. The software needs to distinguish the processing process according to the type of fill light, if it is an infrared fill light, the DTS fill light node label needs to have the word IR to identify the hardware type, and the LED fill light can remove the IR field.
2. For this hardware circuit controlled by a single pin, there are two cases, one is fixed brightness, which is directly controlled by GPIO. The other is brightness controllable, using PWM, setting the brightness by adjusting the duty cycle, DTS PWMS or Enable-GPIO, one of the two configurations.

9.2 FLASHLight driver description

9.2.1 A brief description of the data type

9.2.1.1 struct i2c_driver

[illustrate]

Define I2C device driver information

[Definition]

```
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;
     ……
};
```

[Key Member]

Member name	Description
@driver	The Device driver model driver mainly contains the driver name and the of_match_table that matches the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the DTS file
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
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);
```

9.2.1.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

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

Currently, the following private IOCTL is used to query flash lighting time information.

RK_VIDIOC_FLASH_TIMEINFO

9.2.1.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	Description
.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.

[Example]

```
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,
    }
};
```

9.2.2 A brief description of the API

9.2.2.1 xxxx_set_ctrl

[Description]

Set the flash mode, current, and flash timeout time.

[Grammar]

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

[Parameter]

Parameter name	Description	Input and output
*ctrl	v4l2 control Struct pointer	input

[Return Value]

Return value	Description
0	Success
Non-0	Fail

9.2.2.2 xxxx_get_ctrl

[Description]

Gets the flash fault status.

[Grammar]

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

[Parameter]

Parameter name	Description	Input and output
*ctrl	v4l2 control Struct pointer	output

[Return Value]

Return value	Description
0	Success
Non-0	Fail

9.2.2.3 xxxx_ioctl xxxx_compat_ioctl

[Description]

The implementation function of custom ioctl, mainly contains the time information of obtaining the flash on,

Implemented custom RK_VIDIOC_COMPAT_FLASH_TIMEINFO.

[Grammar]

```
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)
```

[Parameter]

Parameter name	Description	Input and output
*sd	v4l2 subdev Struct pointer	input
cmd	ioctl command	input
*arg/arg	Parameter pointer	output

[Return Value]

Return value	Description
0	Success
Non-0	Fail

9.2.3 Drive the migration step

For ordinary GPIO direct control LEDs, refer to kernel/drivers/leds/leds-rgb13h.c and

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

For flashlight driver ICs, the following steps can be ported

1.Implement the standard I2C sub-device driver part.

1.1 According to the description of **struct i2c_driver**, the following parts are mainly implemented:

1.2 Probe function implementation details:

1）FLASHLIGHT DEVICE RESOURCE ACQUISITION, MAINLY TO OBTAIN DTS RESOURCES, REFER TO FLASHLIGHT设备注册(DTS);

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly provide device parameters and Camera device for matching.

2）Flash device name:

For dual LED flashes, use the device name of LED 0 and LED 1 to distinguish.

```
/* 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 DEVICE AND INITIALIZATION OF MEDIA ENTITIES.

v4l2 sub-device: v4l2_i2c_subdev_init, the RK flashlight driver requires subdev to have its own device node for user-mode camera_engine to access, and realize LED control through this device node;

Media entities: media_entity_init;

2. Implement the V4L2 sub-device driver, mainly implement the following 2 members:

.ioctl.compat_ioctl32

This callback mainly implements the RK private control command, which involves:

Member name	Description
RK_VIDIOC_FLASH_TIMEINFO	The time of the LED is camera_engine obtained through this command, and it is used to determine whether the exposure time of the CIS frame is after the flash.

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl and .s_ctrl implement the following commands with standard v4l2 control:

Member name	**Description**
V4L2_CID_FLASH_FAULT	Get flash fault information
V4L2_CID_FLASH_LED_MODE	Set the LED mode V4L2_FLASH_LED_MODE_NONE V4L2_FLASH_LED_MODE_TORCH V4L2_FLASH_LED_MODE_FLASH
V4L2_CID_FLASH_STROBE	Control flash on
V4L2_CID_FLASH_STROBE_STOP	Control flash off
V4L2_CID_FLASH_TIMEOUT	Set flash mode maximum duration
V4L2_CID_FLASH_INTENSITY	Set flash mode current
V4L2_CID_FLASH_TORCH_INTENSITY	Set the flare mode current

10. FOCUS ZOOM P-IRIS DRIVEN

The drive here refers to the stepper motor controlled by the autofocus (FOCUS), zoom (ZOOM), and automatic aperture (P-IRIS). Due to the same stepper motor control method and hardware design factors, the drive with three functions is integrated into one drive. According to the driver chip used, such as the SPI control chip, the driver can be packaged into SPI framework sub-devices, this section around MP6507, MS41908 driver chip to describe the driver to implement the data structure, framework and precautions.

10.1 MP6507 Device Registration (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 Private Definition Description:

Member name	Description
rockchip,camera-module-index	The camera ordinal number, the field that matches the camera
rockchip,camera-module-facing	Camera orientation, the field that matches the camera
iris_en-gpios	IRIS enables GPIO
focus_en-gpios	Focus enables GPIO
zoom_en-gpios	zoom enables GPIO
rockchip,iris-step-max	Maximum number of steps to be moved by a P-IRIS stepper motor
rockchip,focus-step-max	Maximum number of steps moved by the focusing stepper motor
zoom-step-max	Maximum number of steps moved by a zoom stepper motor
iris-start-up-speed	Starting speed of stepper motors used by IRIS
focus-start-up-speed	The starting speed of the stepper motor used by the Focus
focus-max-speed	The maximum operating speed of the stepper motor used by the focus
zoom-start-up-speed	Start-up speed of stepper motors used by Zoom
zoom-max-speed	Maximum operating speed of stepper motors used by Zoom
focus-first-speed-step	Focus starts the number of steps running at the speed, and increases the number of steps in proportion to the subsequent acceleration interval to make the running time of each speed segment as close as possible to the same.
zoom-first-speed-step	The number of steps run by zoom startup speed, and the number of steps in proportion to the subsequent acceleration interval is increased so that the running time of each speed segment is as close as possible.
focus-speed-up-table	The focus acceleration curve adopts the table lookup method, adjusts the parameters to generate the acceleration curve, configures the data table of the generated trapezoidal acceleration curve or S-shaped acceleration curve, and does not configure or configure a single data, it directly runs at a uniform speed according to the starting speed; The minimum value of the acceleration curve does not exceed the maximum starting speed of the motor, and the maximum value does not exceed the maximum operating speed of the stepper motor.
focus-speed-down-table	Focus deceleration curve, the maximum value of the deceleration curve should be less than the maximum value of the acceleration curve; If the acceleration curve is invalid, the deceleration curve is also invalid, and the whole process runs at a uniform speed according to the starting speed; If no deceleration curve is configured, the deceleration curve is obtained symmetrically from the acceleration curve.

Member name	Description
zoom-speed-up-table	The zoom acceleration curve adopts the table lookup method, adjusts the parameters to generate the acceleration curve, configures the data table of the generated trapezoidal acceleration curve or S-shaped acceleration curve, and does not configure or configure a single data, it directly runs at a uniform speed according to the startup speed; The minimum value of the acceleration curve does not exceed the maximum starting speed of the motor, and the maximum value does not exceed the maximum operating speed of the stepper motor.
zoom-speed-down-table	Zoom deceleration curve, the maximum value of the deceleration curve should be less than the maximum value of the acceleration curve; If the acceleration curve is invalid, the deceleration curve is also invalid, and the whole process runs at a uniform speed according to the starting speed; If no deceleration curve is configured, the deceleration curve is obtained symmetrically from the acceleration curve.

10.1.1 A brief description of the data type

10.1.1.1 struct platform_driver

[illustrate]

Define platform device driver information

[Definition]

```
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;
};
```

[Key Member]

Member name	Description
@driver	The struct device_driver driver mainly contains the driver name and the of_match_table that match the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the dts file
@id_table	If the kernel does not use of_match_table and DTS registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
#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);
```

10.1.1.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

```
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,
};
```

10.1.1.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	Description
.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.

[Example]

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

10.1.2 A brief description of the API

10.1.2.1 xxxx_set_ctrl

[Description]

Call the standard v4l2_control to set focus, zoom, P aperture position.

The following V4L2 standard commands are implemented:

Member name	Description
V4L2_CID_FOCUS_ABSOLUTE	Control focus, 0 means minimum focal length and clear near
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the smallest magnification and the largest field of view
V4L2_CID_IRIS_ABSOLUTE	Controls the size of the P aperture opening, 0 means the aperture is closed

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*ctrl	v4l2 control Struct pointer	Input

[Return Value]

Return value	Description
0	Success
Non-0	Fail

10.1.2.2 xxxx_get_ctrl

[Description]

Call the standard v4l2_control to get the current position of focus, zoom, and P aperture.

The following V4L2 standard commands are implemented:

Member name	Description
V4L2_CID_FOCUS_ABSOLUTE	Control focus, 0 means minimum focal length and clear near
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the smallest magnification and the largest field of view
V4L2_CID_IRIS_ABSOLUTE	Controls the size of the P aperture opening, 0 means aperture off

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*ctrl	v4l2 control Struct pointer	output

[Return Value]

Return value	Description
0	Success
Non-0	Fail

10.1.2.3 xxxx_ioctl xxxx_compat_ioctl

[Description]

The implementation function of custom ioctl mainly contains the time information of obtaining focus, zoom, and P aperture (timestamps of starting and ending movement), because the lens used does not have a positioning device, and when necessary, the position of the lens motor needs to be reset.

Customization implemented:

Member name	Description
RK_VIDIOC_VCM_TIMEINFO	Time information for focus, used to confirm whether the current frame is the effective frame after the focus is completed.
RK_VIDIOC_ZOOM_TIMEINFO	Zoom time information, used to confirm whether the current frame is the effective frame after zoom is completed.
RK_VIDIOC_IRIS_TIMEINFO	The time information of the aperture, which is used to confirm whether the current frame is the effective frame after the aperture adjustment
RK_VIDIOC_FOCUS_CORRECTION	Focus position correction (reset)
RK_VIDIOC_ZOOM_CORRECTION	Zoom position correction (reset)
RK_VIDIOC_IRIS_CORRECTION	Aperture position correction (reset)

[Grammar]

```
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)
```

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev Struct pointer	Input
cmd	ioctl command	Input
*arg/arg	Parameter pointer	output

[Return Value]

Return Value	Description
0	success
Non-0	fail

10.1.3 Drive the migration step

For the driver chip controlled by SPI, the SPI framework can be used for device driver porting, and the RK reference driver uses MP6507, which directly uses PWM output control waveform, and power amplification is performed through MP6507, so the platform framework is directly ported.
Driver reference: /kernel/drivers/media/i2c/mp6507.c

The migration steps are as follows:

1.Implement the standard platform sub-device driver part.

1.1 According to the description of **struct platform_driver**, the following parts are mainly implemented:

struct driver.name

struct driver. of_match_table

probe function

Remove function

1.2 Probe function implementation details:

1)DEVICE RESOURCE ACQUISITION, MAINLY TO OBTAIN DTS RESOURCES, REFER TO [FOCUS ZOOM P-IRIS DEVICE REGISTRATION (DTS)] (#_FOCUS ZOOM P-IRIS DEVICE REGISTRATION (DTS));

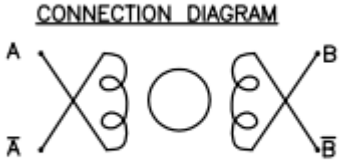
1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly provide device parameters and Camera device for matching.

1.2) OBTAIN THE PWM CONFIGURATION, ACCORDING TO THE CONTROL MODE OF THE MOTOR, THE AB PHASE IS 90 DEGREES APART, WHICH CAN BE ACHIEVED BY ALIGNING THE PW SETTING CENTER OF THE B PHASE, AND CONFIGURE CENTER-ALIGNED IN THE DTS PWM NODE, SEE [FOCUS ZOOM P-IRIS DEVICE REGISTRATION (DTS)] (#_FOCUS ZOOM P-IRIS DEVICE REGISTRATION (DTS)) for details;

SEQUENCE OF EXCITATION

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

Output Shaft Rotation CW



1.3) Get the enable pin, MP6507 needs to use 4 PWM to generate stepper motor control waveform, due to the limited hardware PWM, and focus, zoom, P aperture three stepper motors each use an MP6507 driver drive, so through GPIO to enable the corresponding MP6507 driver, so as to achieve PWM time-sharing multiplexing, of course, this also has a disadvantage, only one stepper motor can be driven at the same time, the other two stepper motors need to wait for the end of the previous operation to continue operation;

1.4) Obtain hardware-related constraints and resources such as maximum step, maximum starting speed, maximum operating speed, acceleration curve data and other maximum step of each motor;

2. hrtimer_init, the timer is initialized, PWM is used in continuous mode, the timer timing is required, after reaching the specified number of output PWM waveforms, the advance timer interrupts to close the PWM, and the acceleration process also needs to enter the timer to interrupt and modify the PWM frequency after running to the specified number of waveforms, so as to realize the acceleration of the stepper motor;
3. init_completion, the synchronization mechanism is realized through completion, only when the previous motor moving operation ends, the next motor operation can be carried out;
4. Initialization of the v4l2 device and media entity.

V4L2 sub-device: v4l2_i2c_subdev_init, the driver requires Subdev to have its own device node for user state rkaiq access, through which the motor control is realized;

Media entities: media_entity_init;

5. Device Name:

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

2. Implement the V4L2 sub-device driver, mainly implement the following two members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to **v4l2_subdev_core_ops** to describe the implementation of callback functions, mainly implement the following callback functions:

```
.ioctl
.compat_ioctl32
```

This callback mainly implements the RK private control command, which involves:

Member name	Description
RK_VIDIOC_VCM_TIMEINFO	Time information for focus, used to confirm whether the current frame is the effective frame after the focus is completed
RK_VIDIOC_ZOOM_TIMEINFO	Zoom time information, used to confirm whether the current frame is the effective frame after zoom is completed
RK_VIDIOC_IRIS_TIMEINFO	Time information of the aperture, used to confirm whether the current frame is the effective frame after the aperture adjustment
RK_VIDIOC_FOCUS_CORRECTION	Focus position correction (reset)
RK_VIDIOC_ZOOM_CORRECTION	Zoom position correction (reset)
RK_VIDIOC_IRIS_CORRECTION	Aperture position correction (reset)

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.g_volatile_ctrl

.s_ctrl

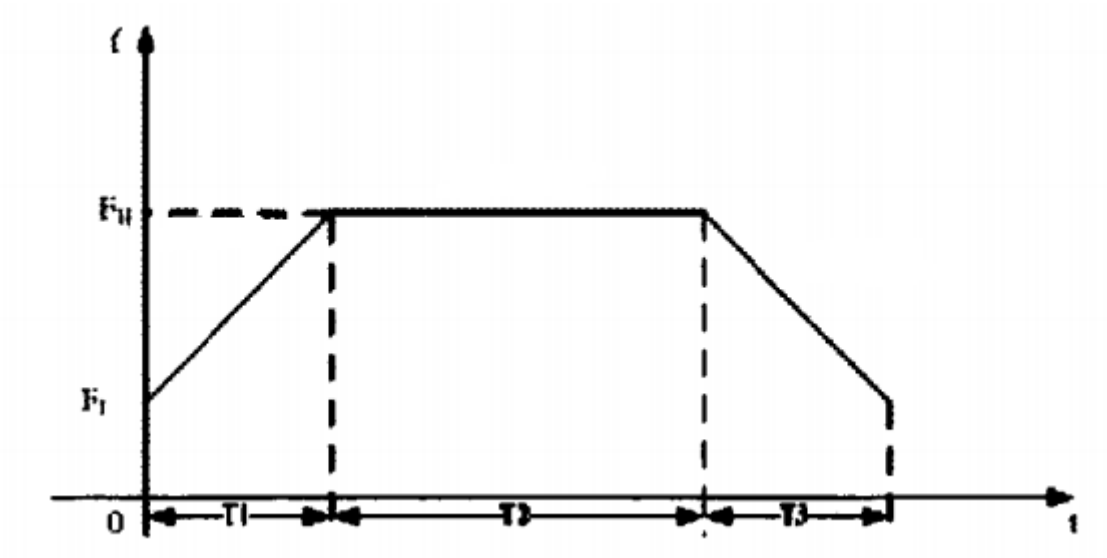
.g_volatile_ctrl and .s_ctrl implement the following commands with standard v4l2 control:

Member name	Description
V4L2_CID_FOCUS_ABSOLUTE	Control focus, 0 means minimum focal length and clear near
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the smallest magnification and the largest field of view
V4L2_CID_IRIS_ABSOLUTE	Controls the size of the P aperture opening, 0 means aperture off

3. Stepper motor acceleration curve reference:

3.1 Trapezoidal curves

It is possible to accelerate and decelerate at equal intervals and speeds as shown.



3.2 S-shaped curve

If the trapezoidal acceleration is not ideal, you can consider the S-type acceleration, which can refer to the following formula:

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

thereinto

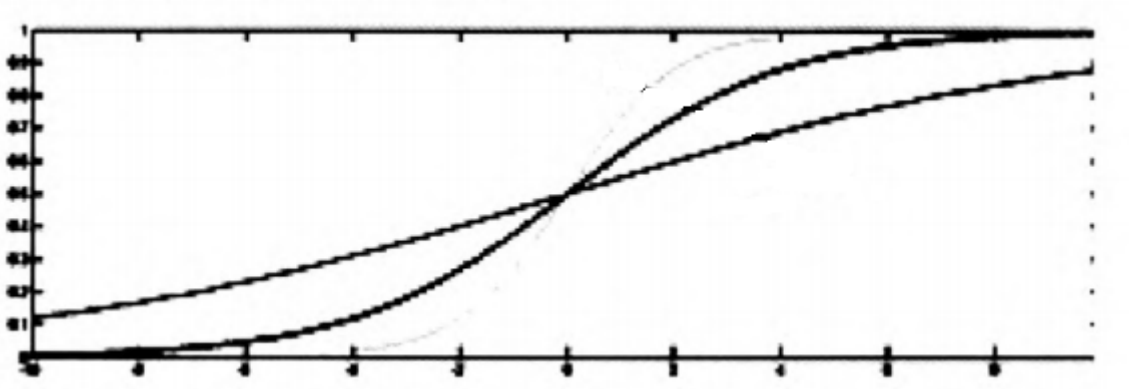
Vmin refers to the motor starting speed

Vmax refers to the target speed of the motor

fac is the curve coefficient, the general range is 4~6, the larger the value, the steeper the middle of the curve

i is the speed segment serial number, such as divided into 32 segments of acceleration, the value is 0~31

Num is half the number of velocity segments, and if divided into 32 segments, NUM is 16



10.2 MS41908 Device Registration (DTS)

Since some lenses support the combination of PIRIS, FOCUS, ZOOM, ZOOM1 or DC-IRIS, FOCUS, ZOOM, so MS41908 MAKES PIRIS, FOCUS, ZOOM, ZOOM1, DC-IRIS function configurable, can load the driver multiple times, to achieve the combination of multiple driver chips, dts will be more complicated, please read the description of each parameter carefully.

```
&spi0 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&spi0m0_clk &spi0m0_cs0n &spi0m0_miso &spi0m0_mosi>;
```

```
    //If not, check whether the default pinctrl is the actual PIN group used
    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>;
        //Multiple device registration, lens-focus = <&ms41908_0 &ms41908_1>;
        ...
    };
};
```

10.2.1 Description of the basic definition:

Member name	Description
pinctrl-0	The SPI pin definition can be mapped to the SPI function ag according to the actual pin configuration. pinctrl-0 = <&spi0m0_clk &spi0m0_cs0n &spi0m0_miso &spi0m0_mosi>;
assigned-clocks assigned-clock-rates	SPI clock configuration, recommended to configure at 100MHz
reg	reg = <0>; indicates the use of cs0 reg = <1>; Indicates the use of cs1
rockchip,camera-module-index	The camera ordinal number, the field that matches the camera
rockchip,camera-module-facing	Camera orientation, the field that matches the camera
reset-gpios	The reset pin of the MS41908 can be left without configuration when the hardware is fixed pull-up
vd_fz-period-us	The stepper motor register updates the required pulse signal period, the pulse signal of the two stepper motors is the same, the motor running time exceeds the VD cycle will cause step loss, the drive will ensure that the single motor movement cycle time is within the VD cycle

10.2.2 Description of FOCUS related definitions:

Member name	Description
use-focus	Whether to use the focus function
focus-used-pin	Each MS41908 chip can drive two stepper motors, and the corresponding PIN group is called "AB" and "CD", according to the actual hardware connection configuration
focus-backlash	The error caused by the gear clearance and the number of steps to compensate for the change of motor direction are obtained according to the actual lens test
focus-start-up-speed	The starting speed of the stepper motor, in PPS
focus-step-max	The effective range of motion of the motor, in steps
focus-ppw	Set the MS41908 output PWM duty cycle, 0-255, the larger the value, the stronger the drive capacity, adjust according to the motor load
focus-phmode	Set the MS41908 output PWM waveform phase correction, generally not configured, depending on the situation
focus-micro	Set the number of microsteps, divided into 64, 128, 256 subdivisions, default 256 subdivisions
focus-reback-distance	The focus curve needs to go in the same direction for the position to be accurate. for example, the current position is 100, if you want to return to 90, you need to return to 80 first, and then go to the 90 position, the position is accurate, and the parameter configured here is the number of steps of multiple callbacks
focus-1-2phase-excitation	The motor excitation method defaults to 2-2 phase excitation, and this parameter can be configured ag using 1-2 phase excitation. focus-1-2phase-excitation;
focus-dir-opposite	If the current motor movement direction is opposite to the actual focusing curve, this parameter can be configured to reverse the motor movement direction ag. focus-dir-opposite;

Description of optocoupler related definitions:

Member name	Description
focus-pic	The C pin of the optocoupler is used to detect the level change, and the junction point of the level change is the origin of the optocoupler mark
focus-pia	The A pin of the photocoupler, drive the photodiode, pull high when the photocoupler is corrected, and the lens should be pulled low when it is working normally, otherwise the photodiode will affect the imaging
focus-pie	The hardware can be directly grounded when designing, and if it is designed for GPIO control, it is necessary to configure the pin
focus-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual
focus-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual

Note: Lenses that do not use optocoupler positioning do not need to configure optocoupler parameters.

10.2.3 Description of ZOOM related definitions:

Member name	Description
use-zoom	Whether to use the function of zoom
zoom-used-pin	Each MS41908 chip can drive two stepper motors, and the corresponding PIN group is called "AB" and "CD", according to the actual hardware connection configuration
zoom-backlash	The error caused by the gear clearance and the number of steps to compensate for the change of motor direction are obtained according to the actual lens test
zoom-start-up-speed	The starting speed of the stepper motor, in PPS
zoom-step-max	The effective range of motion of the motor, in steps
zoom-ppw	Set the MS41908 output PWM duty cycle, 0-255, the larger the value, the stronger the drive capacity, adjust according to the motor load
zoom-phmode	Set the MS41908 output PWM waveform phase correction, generally not configured, depending on the situation
zoom-micro	Set the number of microsteps, divided into 64, 128, 256 subdivisions, default 256 subdivisions
zoom-1-2phase-excitation	The motor excitation method defaults to 2-2 phase excitation, and this parameter can be configured ag using 1-2 phase excitation. zoom-1-2phase-excitation;
zoom-dir-opposite	If the current motor movement direction is opposite to the actual focusing curve, this parameter can be configured to reverse the motor movement direction ag. zoom-dir-opposite;

Description of optocoupler related definitions:

Member name	Description
zoom-pic	The C pin of the optocoupler is used to detect the level change, and the junction point of the level change is the origin of the photocoupler mark need to pay attention to the current drive zoom's optocoupler A / E pin and focus are shared, if the lens zoom pin used is separate, the control of the A / E pin needs to be added.
zoom-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual
zoom-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual

Note: Lenses that do not use optocoupler positioning do not need to configure optocoupler parameters.

10.2.4 ZOOM1 related definitions:

Member name	Description
use-zoom1	Whether to use the function of zoom1, some lenses support control 2 zoom
zoom1-used-pin	Each MS41908 chip can drive two stepper motors, and the corresponding PIN group is called "AB" and "CD", according to the actual hardware connection configuration
zoom1-backlash	The error caused by the gear clearance and the number of steps to compensate for the change of motor direction are obtained according to the actual lens test
zoom1-start-up-speed	The starting speed of the stepper motor, in PPS
zoom1-step-max	he effective range of motion of the motor, in steps
zoom1-ppw	Set the MS41908 output PWM duty cycle, 0-255, the larger the value, the stronger the drive capacity, adjust according to the motor load
zoom1-phmode	Set the MS41908 output PWM waveform phase correction, generally not configured, depending on the situation
zoom1-micro	Set the number of microsteps, divided into 64, 128, 256 subdivisions, default 256 subdivisions
zoom1-1-2phase-excitation	The motor excitation method defaults to 2-2 phase excitation, and this parameter can be configured ag using 1-2 phase excitation. zoom1-1-2phase-excitation;
zoom1-dir-opposite	If the current motor movement direction is opposite to the actual focusing curve, this parameter can be configured to reverse the motor movement direction ag. zoom1-dir-opposite;

Description of optocoupler related definitions:

Member name	Description
zoom1-pic	The C pin of the optocoupler is used to detect the level change, and the junction point of the level change is the origin of the optocoupler mark
zoom1-pia	The A pin of the photocoupler, drive the photodiode, pull high when the photocoupler is corrected, and the lens should be pulled low when it is working normally, otherwise the photodiode will affect the imaging
zoom1-pie	The hardware can be directly grounded when designing, and if it is designed for GPIO control, it is necessary to configure the pin
zoom1-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual
zoom1-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual

Note: Lenses that do not use optocoupler positioning do not need to configure optocoupler parameters.

10.2.5 Description of PIRIS:

Member name	Description
use-p-iris	Whether to use the functions of P-IRIS
piris-used-pin	Each MS41908 chip can drive two stepper motors, and the corresponding PIN group is called "AB" and "CD", according to the actual hardware connection configuration
piris-backlash	The error caused by the gear clearance and the number of steps to compensate for the change of motor direction are obtained according to the actual lens test
piris-start-up-speed	The starting speed of the stepper motor, in PPS
piris-step-max	The effective range of motion of the motor, in steps
piris-ppw	Set the MS41908 output PWM duty cycle, 0-255, the larger the value, the stronger the drive capacity, adjust according to the motor load
piris-phmode	Set the MS41908 output PWM waveform phase correction, generally not configured, depending on the situation
piris-micro	Set the number of microsteps, divided into 64, 128, 256 subdivisions, default 256 subdivisions
piris-1-2phase-excitation	The motor excitation method defaults to 2-2 phase excitation, and this parameter can be configured ag using 1-2 phase excitation. piris-1-2phase-excitation;
piris-dir-opposite	If the current motor movement direction is opposite to the actual focusing curve, this parameter can be configured to reverse the motor movement direction ag. piris-dir-opposite;

Description of optocoupler related definitions:

Member name	Description
piris-pic	The C pin of the optocoupler is used to detect the level change, and the junction point of the level change is the origin of the optocoupler mark
piris-pia	he A pin of the photocoupler, drive the photodiode, pull high when the photocoupler is corrected, and the lens should be pulled low when it is working normally, otherwise the photodiode will affect the imaging
piris-pie	The hardware can be directly grounded when designing, and if it is designed for GPIO control, it is necessary to configure the pin
piris-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual
piris-min-pos	There is no calibration step count during optocoupler correction, so the number of steps on the left and right side of the photocoupler origin needs to be measured, and then filled in to dts, which can be larger than the actual

Note: Lenses that do not use optocoupler positioning do not need to configure optocoupler parameters.

10.2.6 DCIRIS related definitions are explained:

Member name	Description
use-dc-iris	Whether to use DC-IRIS features
vd_iris-gpios	The sync pulse pin where the DC aperture correlation register is active
dc-iris-reserved-polarity	DC aperture polarity setting, if 0 is the aperture is wide open, you can set this property to reverse
dc-iris-max-log	The target value range of DC aperture is 0~1023, the actual effective range used may be relatively small, you can configure this parameter to limit the effective range

10.2.7 A brief description of the data type

10.2.7.1 struct spi_driver

[illustrate]

Define platform device driver information

[Definition]

```
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;
};
```

[Key Member]

Member name	Description
@driver	The struct device_driver driver mainly contains the driver name and the of_match_table that match the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the DTS file
@id_table	If the kernel does not use of_match_table and DTS registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
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,
};
```

10.2.7.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

```
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,
};
```

10.2.7.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	Description
.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.

[Example]

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

10.2.8 A brief description of the API

10.2.8.1 xxxx_set_ctrl

[Description]

Call the standard v4l2_control to set focus, zoom, P aperture position.

MS41908 implements the v4l2 standard command:

Member name	Description
V4L2_CID_FOCUS_ABSOLUTE	Control focus, 0 means that the focal length is minimal, the near is clear, and there is no callback
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the smallest magnification, the largest field of view, no callback
V4L2_CID_IRIS_ABSOLUTE	Controls the size of the aperture opening, 0 means aperture off
V4L2_CID_ZOOM_CONTINUOUS	Control zoom factor zoom1, multi-zoom group control when using

[Grammar]

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

[Parameter]

Member name	Description	Input and output
*ctrl	v4l2 control Struct pointer	input

10.2.8.2 xxxx_ioctl xxxx_compat_ioctl

[Description]

The implementation function of custom ioctl mainly contains the time information of obtaining focus, zoom, and P aperture (timestamps of starting and ending movement), because the lens used does not have a positioning device, and when necessary, the position of the lens motor needs to be reset.

Customization implemented:

Member name	Description
RK_VIDIOC_VCM_TIMEINFO	Time information for focus, used to confirm whether the current frame is the effective frame after the focus is completed
RK_VIDIOC_ZOOM_TIMEINFO	Zoom time information, used to confirm whether the current frame is the effective frame after zoom is completed
RK_VIDIOC_IRIS_TIMEINFO	Time information of the aperture, used to confirm whether the current frame is the effective frame after the aperture adjustment
RK_VIDIOC_ZOOM1_TIMEINFO	When multi-zoom group lenses, the time information of zoom1 is used to confirm whether the current frame is the effective frame after zooming is completed
RK_VIDIOC_IRIS_CORRECTION	The aperture position is reset, acting only on the P aperture
RK_VIDIOC_FOCUS_CORRECTION	Focus position reset
RK_VIDIOC_ZOOM_CORRECTION	Zoom position reset
RK_VIDIOC_ZOOM1_CORRECTION	Dual zoom lens, second set zoom position reset
RK_VIDIOC_ZOOM_SET_POSITION	Set the focus parameters, including focus and zoom parameters, and realize multi-step zoom focus linkage according to the zoom curve
RK_VIDIOC_FOCUS_SET_POSITION	Set focus position

Concentrate:

1. In order to solve the problem of inaccurate absolute position of the motor caused by gear clearance, by fixing one direction for positive direction and the other direction for negative direction, the initial position gear card to positive direction, when the motor rotates in the negative direction, in addition to the conventional number of steps to rotate, to rotate more than the gear clearance step n, and then turn n steps in the positive direction, so that the gear can keep stuck in the positive direction, called callback. Callbacks guarantee the accuracy of the absolute position. However, the number of steps of the callback is greater than the gear gap, in the manual focusing, or automatic focus process, if it moves in the negative direction many times, the continuous callback will cause the picture to shake, so it cannot be called back every time it turns in the negative direction, so the new RK_VIDIOC_FOCUS_SET_POSITION, RK_VIDIOC_ZOOM_SET_POSITION interface, the AF algorithm decides whether to callback. Standard v4l2 commands V4L2_CID_FOCUS_ABSOLUTE and V4L2_CID_ZOOM_ABSOLUTE do not make callbacks, and are only used in manual mode. The RK_VIDIOC_ZOOM_SET_POSITION includes focus

and zoom parameters, and the focus is adjusted synchronously during the zoom process, making the image overly natural.

2. In the early stage, in order to solve the gear gap, by configuring focus-backlash, when rotating in the negative direction, the number of steps of the gear gap is rotated, so as to offset the gear gap, but due to the individual differences in the lens gear gap, there is no calibration error, and the calibration workload is large, so this parameter is abandoned. The drive retention design can still be used if the motor position accuracy is not required.

[Grammar]

```
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)
```

[Parameter]

Member name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
cmd	ioctl command	input
*arg/arg	parameter pointer	output

[Return Value]

Return Value	Description
0	success
Non-0	fail

10.2.9 Drive the migration step

For SPI-controlled driver chips, the SPI framework can be used for device driver porting, MS41908 as a reference.
Driver reference: /kernel/drivers/media/spi/ms41908.c

The migration steps are as follows:

1.Implement the standard SPI sub-device driver section.

1.1 According to the description of struct spi_driver**, the following parts are mainly implemented:

struct driver.name

struct driver. of_match_table

probe function

Remove function

1.2 Probe function implementation details:

1)Device resource acquisition, mainly to obtain DTS resources, refer to [MS41908 Device Registration (DTS)](#_MS41908 Device Registration (DTS));

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly provide device parameters and Camera device for matching.

1.2) Obtain motor-related configuration parameters, define them according to the functional requirements of the chip, and try to make the parameters related to motor movement configurable.

2)hrtimer_init, timer initialization, MS41908 is to VD signal as trigger signal, use timer to fix the cycle of each VD, easy to operate, the register is effective after the VD signal, each time you need to modify the register value, you can configure the register in advance before the VD signal. It should be noted that the movement speed and number of steps configured by the register, within the VD period range, the number of steps that exceed the VD period will be lost.

3)init_completion, the synchronization mechanism is realized through completion, for the same motor, the next operation can only be carried out after the previous operation has ended.

4)Initialization of the v4l2 device and media entity.

V4L2 sub-device: v4l2_i2c_subdev_init, the driver requires Subdev to have its own device node for user state rkaiq access, through which the motor control is realized;

Media entities: media_entity_init;

5)Device Name:

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

2. Implement the V4L2 sub-device driver, mainly implement the following 2 members:

```
.ioctl
.compat_ioctl32
```

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.g_volatile_ctrl

.s_ctrl

11. DC-IRIS drive

DC-IRIS relative to P-IRIS, can not accurately know the size of the aperture opening, the general use scenario is the default full open, when the exposure is adjusted to the minimum, the image is still overexposed, then enter the aperture adjustment, when the exposure is set to the maximum, the image is still underexposed, enter the aperture adjustment. DC-IRIS motors are DC motors that buffer the regulation speed of the motor through negative feedback from Hall devices. For the drive, as long as the motor rotation is controlled by a PWM, when the PWM duty cycle is less than 20%, the aperture will slowly close until it is completely closed, the smaller the duty cycle, the faster the aperture closes; When the duty cycle is greater than 40%, the aperture will slowly open, the larger

the duty cycle, the faster the opening speed; The 20%~40% interval aperture is in a holding state. The 20% and 40% here are not fixed values, but are related to the frequency of PWM and the actual hardware device accuracy.
Reference drivers: /kernel/drivers/media/i2c/hall-dc-motor.c

11.1 DC-IRIS Device Registration (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>;
        ...
    }
}
```

11.1.1 A brief description of the data type

11.1.1.1 struct platform_driver

[illustrate]

Define platform device driver information

[Definition]

```
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;
};
```

[Key Member]

Member name	Description
@driver	The struct device_driver driver mainly contains the driver name and the of_match_table that match the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the dts file
@id_table	If the kernel does not use of_match_table and DTS registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
#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);
```

11.1.1.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

```
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,
};
```

11.1.1.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	Description
.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.

[Example]

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

11.1.2 A brief description of the API

11.1.2.1 xxxx_set_ctrl

[Description]

Call the standard v4l2_control aperture position, DC aperture can not actually know the specific position of the aperture, the value set here is the duty cycle of PWM.

The following V4L2 standard commands are implemented:

Member name	Description
V4L2_CID_IRIS_ABSOLUTE	Set the duty cycle of PWM that controls the aperture, range (0~100)

[Grammar]

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

[Parameter]

Parameter name	Description	Input and output
*ctrl	v4l2 control Struct pointer	input

[Return Value]

Return Value	Description
0	success
Non-0	fail

11.1.2.2 xxxx_ioctl xxxx_compat_ioctl

[Description]

At present, there is no private definition to be implemented, and the v4l2 framework registration needs to implement empty functions.

[Grammar]

```
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)
```

[Parameter]

Parameter name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
cmd	ioctl command	input
*arg/arg	Parameter pointer	output

[Return Value]

Return Value	Description
0	success
Non-0	fail

11.1.3 Drive the migration step

Driver reference: /kernel/drivers/media/i2c/hall-dc-motor.c

The migration steps are as follows:

1.Implement the standard platform sub-device driver part.

1.1 According to the description of **struct platform_driver**, the following parts are mainly implemented:

struct driver.name

struct driver. of_match_table

probe function

Remove function

1.2 Probe function implementation details:

1)Device resource acquisition, mainly to obtain DTS resources, refer toDC-IRIS设备注册(DTS);

```
1.1) RK private resource definition, naming methods such as rockchip, camera-
module-xxx, mainly provide device parameters and Camera device for matching.

1.2) To obtain PWM resources, pay attention to whether the PWM node is enabled.
```

2.Initialization of the v4l2 device and the media entity.

V4L2 sub-device: v4l2_i2c_subdev_init, the driver requires Subdev to have its own device node for user state rkaiq access, through which the motor control is realized;

Media entities: media_entity_init;

3. Flash device name:

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

2.Implement the V4L2 sub-device driver, mainly implementing the following two members:

```
ioctl
.compat_ioctl32
```

This callback does not need to implement specific commands at present, but as a v4l2 child device must implement this operation function, so an empty function is implemented here.

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.g_volatile_ctrl.s_ctrl

.g_volatile_ctrl and .s_ctrl implement the following commands with standard v4l2 control:

Member name	Description
V4L2_CID_IRIS_ABSOLUTE	Set the duty cycle of PWM that controls the aperture, range (0~100)

12. RK-IRCUT drive

IRCUT is controlled by two wires, 3.5v~6v power supply is applied to these two lines, and IRCUT switching can be realized by adjusting the positive and negative poles of the IRCUT power supply and meeting the power-on time of 100ms±10%. The drive controls the current output direction of the motor driver through two GPIOs, and the GPIO commands are Open (red line) and Close (black line). The current flows from open to close, which is an infrared cut-off filter and works during the day; The current flows from close to open, which is a white glass sheet and works at night.

12.1 RK-IRCUT Device Registration (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>;
        ...
    }
}
```

12.1.1 A brief description of the data type

12.1.1.1 struct platform_driver

[illustrate]

Define platform device driver information

[Definition]

```
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;
};
```

[Key Member]

Member name	Description
@driver	The struct device_driver driver mainly contains the driver name and the of_match_table that match the DTS registered device. The .probe function is called when the compatible field in the of_match_table matches the compatible field of the DTS file
@id_table	If the kernel does not use of_match_table and DTS registered devices for matching, the kernel uses that table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

```
#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);
```

12.1.1.2 struct v4l2_subdev_core_ops

[illustrate]

Define core ops callbacks for subdevs.

[Definition]

```
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
    ……
};
```

[Key Member]

Member name	Description
.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.

[Example]

```
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,
};
```

12.1.1.3 struct v4l2_ctrl_ops

[illustrate]

The control operations that the driver has to provide.

[Definition]

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

[Key Member]

Member name	Description
.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.

[Example]

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

12.1.2 A brief description of the API

12.1.2.1 xxxx_set_ctrl

[Description]

Call the standard v4l2_control switch IRCUT.

The following V4L2 standard commands are implemented:

Member name	Description
V4L2_CID_BAND_STOP_FILTER	0 is the CLOSE state, infrared light can enter; 3 is the OPEN state, infrared light cannot enter;

[Grammar]

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

[Parameter]

Parameter name	Description	Input and output
*ctrl	v4l2 control struct pointer	input

[Return Value]

Return Value	Description
0	success
Non-0	fail

12.1.2.2 xxxx_ioctl xxxx_compat_ioctl

[Description]

At present, there is no private definition to be implemented, and the v4l2 framework registration needs to implement empty functions.

[Grammar]

```
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)
```

[Parameter]

Parameter name	Description	Input and output
*sd	v4l2 subdev struct pointer	input
cmd	ioctl command	input
*arg/arg	struct pointer	output

[Return Value]

Return Value	Description
0	success
Non-0	fail

12.1.3 Drive the migration step

Driver reference: /kernel/drivers/media/i2c/rk_ircut.c

The migration steps are as follows:

1.Implement the standard platform sub-device driver part.

1.1 According to the description of **struct platform_driver**, the following parts are mainly implemented:

struct driver.name

struct driver. of_match_table

probe function

Remove function

1.2 Probe function implementation details:

1)Device resource acquisition, mainly to obtain DTS resources, refer to 参考RK-IRCUT设备注册(DTS);

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly provide device parameters and Camera device for matching.

1.2) Get open, close GPIO resources;

2. init_completion, through the completion to implement the synchronization mechanism, since it takes about 100ms to switch IRCUT, the completion synchronization mechanism is required to ensure that the last IRCUT switch has been completed before it can be operated again;
3. Create a work queue and put the switching operation on the work queue to avoid blocking for a long time;
4. Initialization of the v4l2 device and media entity.

v4l2 sub-device: v4l2_i2c_subdev_init, the driver requires subdev to have its own device node for user rkaiq access, and realize IRCUT control through this device node;

Media entities: media_entity_init;

```
sd->entity.function = MEDIA_ENT_F_LENS;
sd->entity.flags = 1;//flag is fixed to 1 and is used to distinguish subdevices
of other MEDIA_ENT_F_LENS types
```

5. Device Name:

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

2. Implement the V4L2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to **v4l2_subdev_core_ops** to describe the implementation of callback functions, mainly implement the following callback functions:

```
.ioctl
.compat_ioctl32
```

This callback does not need to implement a private command at present, but the v4l2 framework registration requirements, so it implements an empty function, and the function content can be supplemented according to the requirements in the future.

2.2 Refer to **v4l2_ctrl_ops** to implement callback functions, mainly implementing the following callback functions:

.s_ctrl

.s_ctrl implement the following commands with standard v4l2 control:

Member name	**Description**
V4L2_CID_BAND_STOP_FILTER	0 is the CLOSE state, infrared light can enter; 3 is the OPEN state, infrared light cannot enter;

13. Media-CTL v4L2-CTL tool

The media-ctl tool operates through media devices such as /dev/medio0, which manages the format, size, and links of each node in the Media topology.

The v4l2-ctl tool is for video devices such as /dev/video0, /dev/video1, which performs a series of operations such as set_fmt, reqbuf, qbuf, dqbuf, stream_on, stream_off and so on on the video device.

For specific usage, please refer to the help information of the command, and the following are several common uses.

1. Print the topology

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

Note: ISP2 has many device nodes, and there may be media0/media1/media2 nodes, so you need to enumerate the device information one by one.

2. Links

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

Note: Disconnect the path of the ISPP, link to the main_path, grab the RAW graph from the main_path, media-ctl does not add -d to specify the device, the default is /dev/media0 device, you need to confirm which device node rkisp-isp-subdev is hung on, usually /dev/media1.

3. Modify FMT/size

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

Note: You need to confirm which media device the camera device node (ov5695 7-0036) is attached to.

4. Set up FMT and capture frames

```
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
```

4. Set controls such as exposure and gain

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

Note: The ISP driver will call the control command of the camera sub-device, so specifying the device as video3 (main_path or self_path) can be set to the exposure, vicap will not call the control command of the camera sub-device, and the control command will fail to set directly on the acquisition node. The correct way to do this is to find the camera device node is /dev/v4l-subdevX, and configure the endpoint directly.

14. Memory optimization guidance

14.1 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 data is written to DDR, or ISP MIPI RAW data is written to DDR, and ISP then reads RAW data processing from DDR

Memory occupancy: buf_cnt * buf_size * N, (N = 1: linear mode, 2: HDR2 frame mode 3: HDR3 frame mode).

buf_size： ALIGN(width * bpp / 8, 256) * height; BPP is the bit width, RAW8 RAW10 or RAW12

buf_cnt: 4 by default, define aiq library code hwi/isp20/CamHwIsp20.h, minimum 3.

#define ISP_TX_BUF_NUM 4

#define VIPCAP_TX_BUF_NUM 4

VICAP device configuration **ROCKCHIP_CIF_USE_NONE_DUMMY_BUF** remove 1 internal application buf.

2. DDR_2: ISP FBC YUV420 and GAIN data are written to DDR, and ISPP is read and processed from DDR

Memory occupied: 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 mode 4 buf, 2to1 mode 3 buf, mode configured in IQ XML

Dynamic and static judgment is on Added a thumbnail output, MXN downsampling supports 4x8 and 8x8 IQ xml configuration selection

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

buf_cnt: By default, 6, AIQ enabled self_path video configuration number of bufs

3, DDR_3: ISPP TNR FBC YUV420 and gain data written to the DDR, ISPP NR & Sharp then read from the DDR for processing

Memory occupied: buf_size * buf_cnt

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

buf_cnt: 2, the smallest

4. DDR_4: ispp NR & Sharp yuyv data is written to DDR, and ISPP FEC is read and processed from DDR

Occupied memory: buf_size * buf_cnt (FEC function does not open and does not occupy memory)

buf_size：width * height * 2

buf_cnt: 2, the smallest

5. DDR_5: ISPP 4-way output image buffer, according to the user setting resolution, format and **buf_cnt** calculate the buffer size

The above buf_cnt are where memory can be optimally configured

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
```

15. Latency optimization guide

15.1 rv1109/rv1126

1. Configure vicap to output in advance

Configure wait-line through the DTS rkcif_mipi_lvds node, such as image height 1520, configure wait-line=760, that is, output buffer to ISP in advance after half of the image acquisition. Adjust the wait-line according to the speed at which the ISP reads the buffer.

&rkcif_mipi_lvds {

```
    wait-line = <760>;
```

};

You can also configure the echo 1000 > /sys/devices/platform/rkcif_mipi_lvds/wait_line to debug, which supports configuration at any time.

Note: The wait-line configuration is too small, the ISP accesses the buffer memory too early, some data has not yet been collected, and the end of the image will be abnormal, which may be manifested as image stitching, and you need to select the appropriate wait-line according to the actual test.

2、Configure ISP early output

Configure wait-line through the DTS ISP node, such as image height 1520, configure wait-line=760, that is, output buffer to the backend in advance after half of the image processing. Adjust the wait-line according to ISP processing time and ISPP processing time.

&rkisp_vir0 {

```
    wait-line = <760>;
```

};

You can also debug by configuring /sys/module/video_rkisp/parameters/wait_line, and the configuration is valid before starting ISP (stream/aiq).

Note: The wait-line configuration is too small, and ISPP processing speed is faster than ISP, due to the use of FBC compression format, a hold situation will occur. Start-static judgment and multi-sensor mode are not supported.

3. Configure ISPP 4-way advance output

Configure wait-line through the DTS ISPP node, such as image height 1520, configure wait-line=896, that is, output buffer to the backend in advance after 896 lines of image processing. Adjust the wait-line based on ISPP processing time (NR or FEC) and back-end processing time.

&rkispp_vir0 {

```
status = "okay";

wait-line = <896>;
```

};

You can also debug by configuring /sys/module/video_rkispp/parameters/wait_line, and the configuration is valid before starting ispp(stream/aiq).

Note: If the wait-line configuration is too small and the back-end processing speed is faster than the ISPP, the back-end image processing will be abnormal. Multisensor mode is not supported.

4、Improve hardware processing speed

1. Improve 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. Turn off iommu using reserved memory, and adjust the reserved memory space according to the actual situation.

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

&rkisp {

  memory-region = <&isp_reserved>;

};

&rkispp_mmu {
        status = "disabled";
  };

&rkispp {

        memory-region = <&isp_reserved>;

};
```

16. Multi-camera exposure synchronization function realized

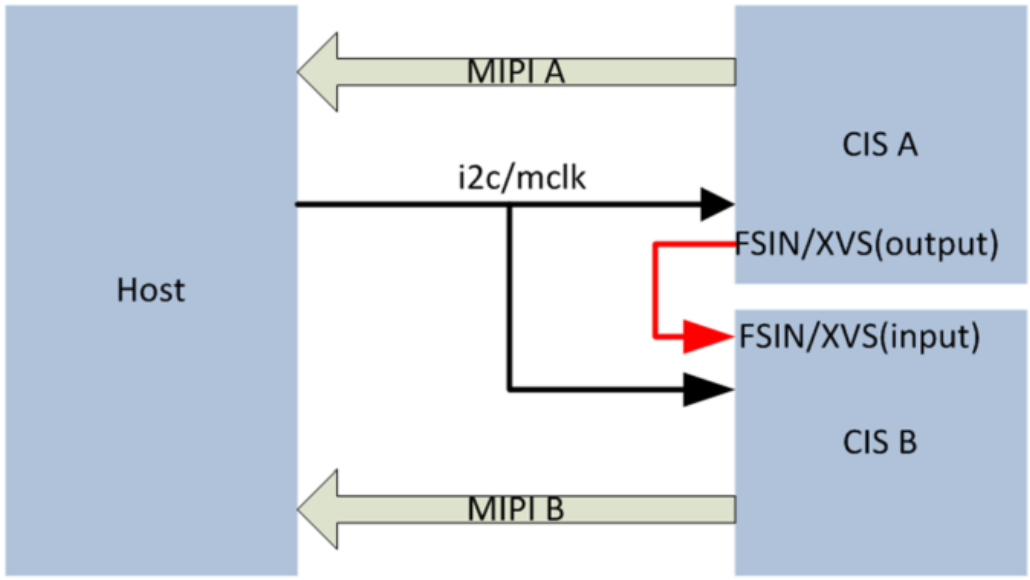
The implementation of the multi-camera exposure synchronization function involves hardware and software.

16.1 Hardware related:

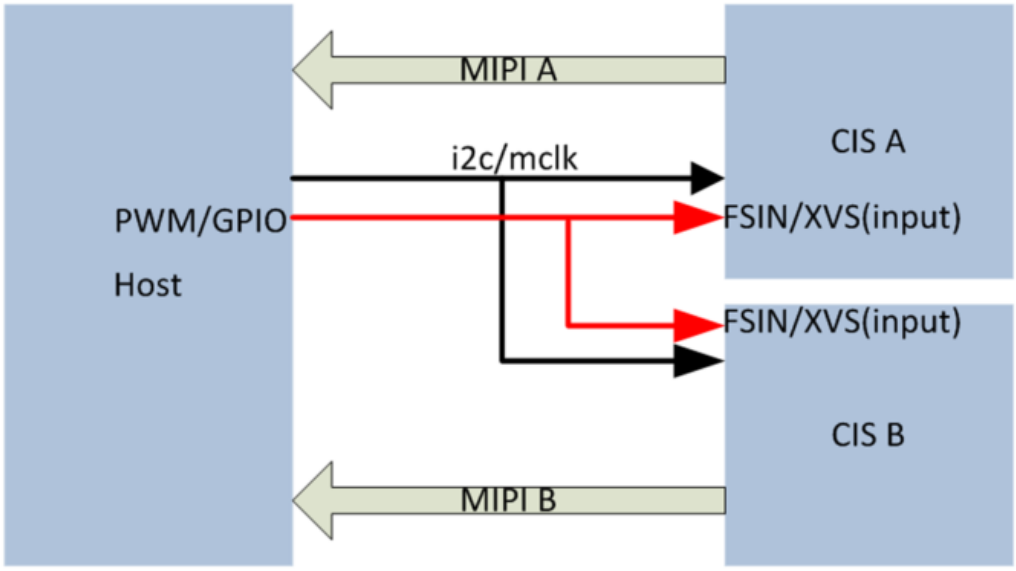
Scenario index	Scheme name	Description of the advantages and disadvantages of the scheme	RK Support
1	**sony:** master-master mode	**Advantages:** 1. Only need to increase the connection of FSIN(OV)/XVS(Sony) single signal between multiple cameras **Deficiency:** 1. Sony CIS adopts this scheme, and the reception of the sync signal CIS(B) is delayed by one exposure line relative to the exposure start time of the output sync signal CIS(A). Generally within 100us.	RK3588
2	**sony:** master-master(External signal sync) mode	**Advantages:** 1. Sony CIS adopts this scheme, CIS(B) is strictly synchronized relative to the CIS(A) exposure initiation moment **Insufficient** 1 . The master and CIS are connected via FSIN(OV) / XVS(Sony). The master outputs the exposure field synchronization signal via hardware PWM. PWM drivers and VI drivers need to be implemented synchronously	NO
3	**sony:** slave-slave(External signal sync) mode	**Advantages:** 1. Sony CIS adopts this scheme, CIS(B) is strictly synchronized relative to the CIS(A) exposure initiation moment **Insufficient:** 1.The master and CIS are connected through XVS, XHS. The master outputs the exposure field synchronization signal via hardware PWM. PWM drivers and VI drivers need to be implemented synchronously	NO

Hardware solution diagram:

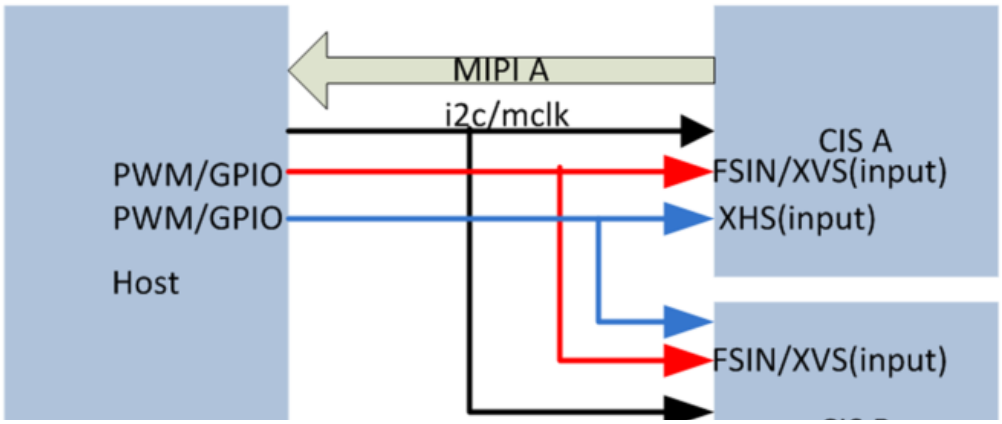
方案 **1**：**(sony:** master-master mode**)**



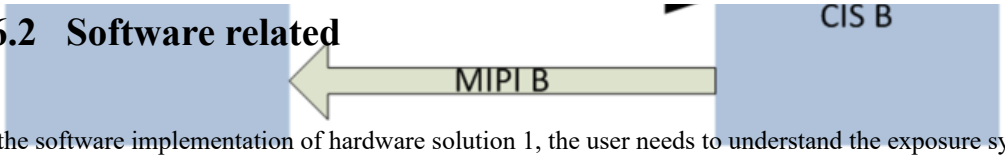
方案 **2**：**(sony:** master-master(External signal sync) mod



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



16.2 Software related



In the software implementation of hardware solution 1, the user needs to understand the exposure synchronization mode defined by the CIS driver:

Mode	FSIN/XVS pin	Stream behavior	Constraints
internal mater	output	1. After configuring the StreamOn register to be enabled, the CIS data interface actively outputs frame data	1. There must be and only 1
external master	input	1. After configuring the StreamOn register, the CIS data interface actively outputs frame data 2. No FSIN/XVS field synchronization signal: according to the register configuration frame rate, actively and continuously output frame data With FSIN/XVS field synchronization signal: according to this signal, synchronize exposure, continuously output frame data	1. Allow multiple
slave	input	1. After configuring the StreamOn register, the CIS data interface actively outputs frame data 2. No FSIN/XVS field synchronization signal: No output of frame data There is a FSIN/XVS field synchronization signal: According to this signal, the exposure is synchronized and the frame data is continuously output	1. Allow multiple
no sync	invalid	1. After configuring the StreamOn register, the CIS data interface actively outputs frame data 2. The FSIN/XVS signal does not affect the data flow	

16.2.1 Key points of the APP call process:

1. The App can call the RK private command through the ioctl interface: RKMODULE_GET_SYNC_MODE / RKMODULE_SET_SYNC_MODE Configure the exposure synchronization mode of all CIS drive devices that need to be synchronized. At the same time, it can also support specifying the exposure synchronization mode of each CIS driver device in advance in the kernel DTS device tree.
2. After Step 1 is completed, call the ioctl interface to enable the CIS device data flow

16.2.2 CIS driver implementation considerations:

1. The CIS driver must implement RKMODULE_GET_SYNC_MODE/RKMODULE_SET_SYNC_MODE to configure the exposure synchronization mode.
2. Due to internal master mode and external master mode mode, the StreamOn register enable device will immediately flow out, in order to avoid the App from obtaining an out-of-sync data stream.

2.1 The CIS driver needs to configure the CIS Initialization Setting and the Corresponding Setting of the exposure synchronization mode in the .s_stream callback, and check that the StreamOn register is enabled only in NO_SYNC_MODE.

2.2 Implement RK's private ioctl command: **RKMODULE_SET_QUICK_STREAM, enable the StreamOn register** in this callback.

3. Reference CIS driver for multi-camera exposure sync mode: drivers/media/i2c/imx464.c

17. VICAP/ISP special collection mode

17.1 Multi-channel RAW data stitching

Multiple RAW SENSORS are connected to RK1608, RK1608 packs a MIPI VC channel output to the RK3588 main control end, and VICAP is used as the whole image acquisition, VICAP virtualizes 3 sub-devices to link to 3 virtual ISP nodes, RKAIQ sends BUf to three ISP nodes, ISP nodes read RAW data according to the offset, when the three ISP nodes have processed RAW data, The buf address will be returned to vicap to continue joining the collection queue.

Key Questions Answered:

1. How is the RK1608 data stitched?

   rk1608 does not do synchronization detection work, synchronization work by the sensor end to do hardware synchronization, rk1608 will detect whether each camera has data, there is data, rk1608 TX will send data. The data is arranged as follows:

   Line 1: sensor0 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line 2: sensor1 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line 3: sensor2 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line 4: sensor0 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line 5: sensor1 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line 6: sensor2 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   ...
   Line n-2: sensor0 |--- valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line n-1: sensor1 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|
   Line n: sensor2 |--- Valid data ---|--- 256-byte alignment (dummy data completion) ---|

The above is a reference for the 3 camera stitching inputs, output at row intervals. Compact storage and 256 bytes rounded up aligned output, which is the ISP's requirement for input data. The MIPI interface of the rk1608 has a corresponding index, you need to configure the input and output of three cameras on the DTS, the section of 8目 MIPI sensor支持introduces the RK1608 node registration, and on the RK1608 DTS, the in_mipi/out_mipi of each sensor must be accurately configured. And the configuration order of each sensor corresponds to ID0~ID2, corresponding to the output order of data.

2. How does ISP disassemble data?

   Each ISP will have a virtual node of VICAP hanging under it, assuming that rkcif_mipi_lvds node is used, then the corresponding virtual nodes are rkcif_mipi_lvds_sditf/rkcif_mipi_lvds_sditf_vir1/rkcif_mipi_lvds_sditf_vir2, and the node needs to be configured with rkckchip, combine-index = <0> to specify the camera data processed by each ISP, the index here corresponds to the sensor ID0~ID2 of the previous problem. This allows each ISP to process the data for each sensor separately according to the row offset according to the index.

3. How does the exposure control correspond to the three sensors?

   The first two questions illustrate how data is sent and how ISPs handle data. So what about the exposure control corresponding to each sensor?

   Since the virtual node of VICAP specifies the order in which each ISP node processes Sensor RAW data, the direct sensor node is mounted to the corresponding Vicap virtual node, and Rkaiq distributes the exposure parameters to the sensor node according to the pipeline. That is, on the V4L2 Media link, it turned out to be: sensor->dphy->csi2 host->rkcif_mipi_lvds ... rkcif_mipi_lvds_sditf->rkisp

In this special mode:

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

In this way, each RKAIQ can map the data processed by each ISP node to the exposure control of each sensor.

18. FAQ

18.1 How to update the driver version separately

Generate 2 SDK **native kernel version**driver patches via git, the command is as follows

Under the new version of the SDK kernel, generate a patch

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

Under the old SDK kernel, patch it

git am tmp_patch/*

Note: The above A is the commit id of the old version kernel, B is the commit id of the new version kernel, and the patch directory is stored tmp_patch.

The old kernel needs to be switched to the native state, otherwise the new version cannot be added to the old version due to some local modifications or patches.

18.2 How to get the driver version number

Get from the kernel startup log

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

The value can be obtained by the following command

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

18.3 How to determine the RKISP driver loading state

If the RKISP driver is successfully loaded, video and media devices are stored in the /dev/directory. Multiple /dev/video devices may exist in the system. Run the /sys command to query the video node registered with RKISP.

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

You can also run the media-ctl command to print the topology to check whether the pipeline is normal.

Check whether the camera driver is successfully loaded. When all cameras are registered, the kernel will print the following log.

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

If the Async subdev notifier completed log line is not displayed in kernel, check whether related errors are reported from the sensor and whether the I2C communication succeeds.

18.4 How do I configure the ISP/VICAP RAW storage format

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| -| -| -| -|

Control instructions:

aiq increases the control of the storage mode. Starting aiq program will configure the storage mode supported according to the sensor model.

rv1109/rv1126: aiq version v0x1.0x71.1 or later supported

rk356x: aiq version v2.60.01 or later supported

Users also can carry out the ioctl by convection equipment to control storage format, include/uapi/Linux/rkisp2 - config. H relevant definition description:

```
#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

Control instructions:
aiq increases the control of the storage mode. Starting aiq program will configure the storage mode supported according to the sensor model.
rv1109/rv1126: aiq version v0x1.0x71.1 or later supported
rk356x: aiq version v2.60.01 or later supported

```
You can also control the storage format by running ioctl on the convection
device, include/uapi/linux/rkcif-config.h has related definitions:
```

```
#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,
};
```

The following commands can be used to configure vicap, but only for debugging

Configured as non-compact, each number corresponds to one channel

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

Configure low alignment

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

Configure high alignment

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

18.5 How do I configure VICAP Abnormal reset

The vicap driver has a reset mechanism. This mechanism is used to reset the cru of the vicap when the vicap is abnormal. At present, it is mainly aimed at abnormal reset of mipi sensor. lvds/dvp sensor will be increased in the future according to the situation. The specific usage is as follows:
To enable the reset mechanism, you need to add rockchip and cif-monitor parameters to the cif-related interface device node on the dts. If the parameters are not set for the dts, the reset mechanism is disabled by default.

```
&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>;
        };
    };
};
```

Among them,
index0: Used to indicate the mode of reset. Currently, there are four main modes, the status of which is as follows:

Mode	illustrate
Idle	don't start the reset mode
Continue	for real-time continuous monitoring error whether vicap mipi and blocks the flow, when the error occurred and the cutoff for vicap reduction; When the number of frames set by index1 reaches, the monitoring timer initializes at the end of the frame and then starts monitoring. If the number of frames does not reach, the monitoring cannot be triggered. The timer is sampled and detected by the period set by index2;
Trigger	only appear on vicap csi2 protocol level mipi Trigger reset error. The number of errors reported by mipi was set with index4. When the number of times of index4 was reached, the initialization of the monitoring timer was triggered by the interruption at the end of the frame. After the period set by the index2 parameters was reached, a vicap reset was realized.
Hotplug	Hotplug mode, mainly for similar n4 interchange/tp6188 switching chip implementation, this kind of car machine is used to solve the problem of plug image lacerate or block; This mode has the function of continue mode, that is, real-time continuous monitoring of vicap mipi error and disconnection, and vicap reset in case of error and disconnection. When the number of frames set by index1 reaches, the monitoring timer initializes at the end of the frame and then starts monitoring. If the number of frames does not reach, the monitoring cannot be triggered. The timer is sampled and detected by the period set by index2; The difference with continue is that if the sensor collected by vicap is enabled with the RKMODULE_SET_VICAP_RST_INFO command, under the condition that mipi does not report errors or interrupt flow, Then vicap will trigger the reset operation after obtaining the information through RKMODULE_GET_VICAP_RST_INFO.

index1: For continue or hotplug, after collecting index1 frame data, the monitoring timer is triggered;

Index2: monitoring period of timer, in the unit of one frame, monitoring period is index2 frame;

Index3: the time parameter of delayed reset. After vicap csi2 reports an error, the monitoring is continued within the defined time. When the error is detected and no longer increases, the reset operation is performed immediately after the defined time, no matter whether the error is increasing or not.

Index4: Used to set the number of occurrences of mipi csi err, and trigger reset when this number is reached;

18.6 How to grab RAW and YUV data output by CIS

After the driver is developed, the CIS output data can be obtained by manipulating the driver directly through the standard v4l2-ctl command. V4l2 - CTL use help can refer to: https://www.mankier.com/1/v4l2-ctl

Example :

```
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 : Specifies the device name

--set-fmt-video : set the resolution that is consistent with the output resolution of the sensor. You can run the media-ctl -p -d /dev/mediaX command to view the current resolution of the sensor.

pixelformat : Output data format

--stream-mmap : Number of mmap buffers.

--stream-count : Indicates the number of frames captured. Multiple frames exist in the same file.

-- stream-to- : Specifies the storage path.

--stream-skip : The number of frames that are skipped.

Device Support list

RV1109/RV1126

equipment	Input interface	Input data format	Device node name	Output Raw	Output YUV
VICAP	DVP	RAW	video0~video3	Non-compact Raw	no
VICAP	MIPI/LVDS	RAW	video0~video3	Non-compact Raw Compact 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	Non-compact Raw Compact Raw	no
ISP	MIPI / LVDS	YUV	rkisp_mainpath	Non-compact Raw	nv12 nv16
ISPP	Read ddr only	YUV	rkispp_m_bypass rkispp_scale0 rkispp_scale1 rkispp_scale2	no	nv12 nv16

RK356X

equipment	Input interface	Input data format	Device node name	Output Raw	Output 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

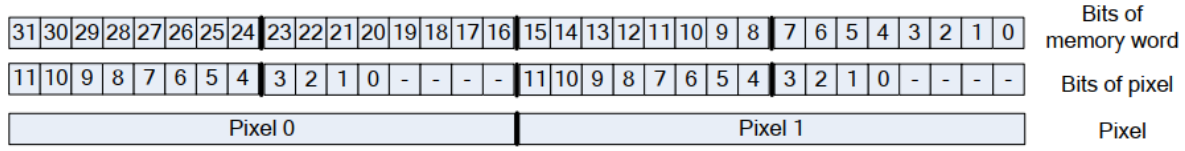
Note:

1. Run the media-ctl -p -d /dev/mediaX command to query the node name (where X indicates 0,1,2,3...).

Raw data storage format

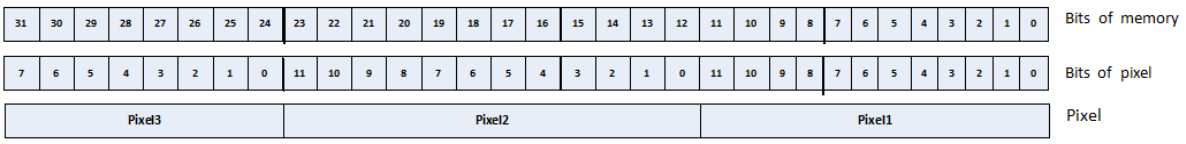
Non-compact storage Format RAW

Non-compact type: Stores the raw10 and raw12 data output by the sensor at 16 bits, aligned at high levels. raw12 data is stored in memory. For example, 4-byte memory fragments are stored as follows:



Compact storage Format RAW

raw12 data is stored in memory. For example, 4-byte memory fragments are stored as follows:



Important reminder:

ISP mainpath device. If the input data is Raw10 or Raw12, the output data is Raw12 in the non-compact format

Reference Use Case :

VICAP outputs Raw

1. The default compact format can be switched between compact and non-compact formats through the following commands:

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

Where, 0 indicates non-compact and 1 indicates compact. For a device that uses multiple channels or uses channels other than vc0, the command can be changed to

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

The numbers following echo correspond to the data store types of channels vc0, vc1, vc2, and vc3 in sequence.

2. video0~3 corresponds to vc0~vc3.
3. Run the v4l2-ctl command

```
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 outputs non-compact Raw

1. The image of mainpath needs to be captured. The default isp output link is rkip-bridge-ispp.

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

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

Note: If -d is not used, media0 is used by default. If rkip-isp-subdev is not used in media0, you need to specify -d to the media node where rkip-isp-subdev resides.

0 indicates pad0, sink. For details, see v4l2 documents.

2. isp output format The default output format is YUYV8_2X8. Run the following command to switch to bayer raw format:

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

Note: The rkip-isp-subdev node does not necessarily reside in media0. -d specifies the device. You need to confirm the media node where rkip-isp-subdev resides.

2 indicates pad2, source. For details, refer to v4l2 documents.

After the modification, you must run the media-ctl -p -d /dev/mediaX command to check whether the modification takes effect. The raw data captured after the modification takes effect is the original raw data.

3. Run the v4l2-ctl command

```
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 output YUV:

Only the input data can be in YUV format. If the input data is in RAW format, vicap cannot output YUV format.

```
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 output 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
```

Note:

```
1. For isp, you can grab mainpath or selfpath. video5 is only an example.
2. When ISP input data is Raw, ISP can convert Raw data into YUV data, including
various image processing operations. This kind of image processing operation
requires RK AIQ to control the various image processing modules of ISP. The
current command is only part of the data flow, and the parameters of the image
processing module adopt the default values of the driver. The image effect is
generally in an abnormal state.
```

ISPP output YUV:

ispp input data sources rkisp_mainpath, rkisp_selfpath, and rkispp_input_image link are disabled. Rkip-bridge-ispp link is enabled. Rkip-isp-subdev pad2: The Source format must be fmt:YUYV8_2X8. link does not need to be configured by default. For details, see the following command.

```
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
```

Note: -d Device name You can select the following nodes based on the capture requirements. Run the media-ctl -p -d /dev/mediaX command to view the node name.

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

18.7 How to switch CIS driver output resolution

1.The sensor driver supports drivers with multiple resolutions. When the raw data of another resolution needs to be captured, you can run the following command to switch the current resolution of the sensor:

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

Note: m01_f_os04a10 1-0036-1 Indicates the name of the sensor node, followed by the format required, provided that the sensor driver supports the format.

2.For vicap, as long as the sensor node is set, the isp input/output format needs to be set. The reference command is as follows:

```
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]'
```

18.8 How to set exposure parameters for CIS

1、Find the sensor node name through media-ctl -p -d /dev/mediaX, the node name format is /dev/v4l-subdevX, refer to the following command:

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

It can also be set separately:

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

2, the maximum exposure is limited by the sensor VTS, the maximum restriction may be VTS-4 or VTS-10, different sensors according to the instructions in the sensor manual to make restrictions. Assuming that the current frame rate is 30fps, the maximum exposure time is 33.3ms, to set 40ms exposure, you have to increase VTS to set 40ms exposure, you can convert it in equal proportions, vts_30fps * 30fps = vts_25fps * 25fps, so as to convert the 25fps corresponding to vts, (vts - height) is vblank, set the converted vblank to the sensor driver to set a larger exposure, The command reference is as follows:

```
v4l2-ctl -d /dev/v4l-subdev4 --set-ctrl vertical_blanking=200
```

18.9 How to support black and white cameras

The CIS driver needs to change the output format of the black and white sensor to one of the following three formats:

```
MEDIA_BUS_FMT_Y8_1X8 (sensor 8bit output)

MEDIA_BUS_FMT_Y10_1X10 (sensor 10bit output)

MEDIA_BUS_FMT_Y12_1X12 (sensor 12bit output)
```

That is, the above format is returned xxxx_get_fmt and xxxx_enum_mbus_code the function.

The RKISP driver makes special settings for these three formats to support the acquisition of black and white images.

In addition, if the application layer needs to obtain images in Y8 format, it can only use SP Path, because only SP Path can support Y8 format output.

18.10 How to support parity field synthesis

The RKISP driver supports parity field synthesis functions, and the restrictions require:

1. MIPI interface: support output frame count number (from frame start and frame end short packets), RKISP drive to determine the parity of the current field;
2. BT656 interface: support output standard SAV/EAV, that is, bit6 is the odd field even field marker information, RKISP drive to judge the parity of the current field;
3. In the RKISP driver RKISP1_selfpath video device node has this function, but other video device nodes do not have this function, and if the app layer mistakenly calls other device nodes, the driver prompts the following error message:

“only selfpath support interlaced”

RKISP_selfpath information can be viewed in media-ctl -p:

```
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]
```

The device driver is implemented as follows:**

The device-driven format.field needs to be set to V4L2_FIELD_INTERLACED, indicating that this current device output format is parity field, that is, the format.field format is returned xxxx_get_fmt the function. Please refer to driver/media/i2c/tc35874x.c driver;

VICAP supports parity field synthesis function limitations:

1. MIPI interface: using virtual width synthesis, the virtual width configuration is twice the real width, and it is received according to two frames. That is, when receiving odd frames, each row of data is stored, and a row of memory is vacated to store even frames, and the buffer address of the even frame is offset by one row of memory from the buffer address of the odd frame.

1.1 It is required to output odd frames first, and even frames are output, otherwise the parity order is reversed and the image is abnormal.

1.2 Require that parity frames have their own FS/FE short packets

2. BT656/BT1120 interface: the module supports parity field synthesis internally, and there are no special requirements.
3. The driver implementation is consistent with RKISP, as long as the sensor driver returns format.field format xxxx_get_fmt the function, ViCap will automatically collect it in this format.

18.11 How to view debug information

1. View the media pipeline information, which corresponds to the DTS camera configuration

```
find /dev -name "media*" | xargs -i media-ctl -p -d {}
```

2. Check the proc information, which is the current status and frame input and output information of vicap/isp/ispp, you can cat several times

```
cat /proc/rk*
```

3. View the driver debug information, set the debug level to the ISP and ISPP nodes, the larger the level value, the more information

```
echo n > /sys/module/video_rkisp/parameters/debug (n = 0, 1, 2, 3, 4; 0 is off)
echo n > /sys/module/video_rkispp/parameters/debug
```

4. Check the register information and pull out isp.reg and ispp.reg

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. Provide debug information steps

    1. Problematic scene 1->2->4->3

2. Reproduction issues 3->-> reproduction->1->2->4

6. ISP/ISPP/VICAP PROC information description

```
[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 clock frequency

Interrupt: Contains MIPI interrupts, interrupts of each module in the ISP, and the data is incremented, indicating that there is data entering the ISP, and ErrCnt error interrupt statistics

Input rkcif_mipi_lvds Format:SBGGR10_1X10 Size:2688x1520@30fps Offset(0,0)

Input source, input format, resolution, frame rate, and crop information

Isp Read mode:frame1 (frame:4061 rate:66ms idle time:10ms frameloss:6077) cnt(total:2026 X1:1969 X2:56 X3:-1)

Read is readback mode (online corresponds to passthrough mode)

Frame1 single-frame linear (frame2 HDR2 frame mode)

frame: The frame number

rate: frame rate, before and after frame interval

IDLE/Working: The status of the ISP hardware

time: ISP hardware processing time

frameloss: Enter the number of dropped frames

cnt: total: total readbacks X1:1 readbacks X2:2 readbacks X3:3 readbacks

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 output stream information, output buf has rotation to update information

size: Output resolution

Dcrop: Outputs cropping information

delay: corresponding delay information (current output point time rub - input image time rub)

Other: The switching status of each module of the ISP

ISP procfs debug function usage instructions

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

The following example of 5 functions, multi-camera ISP multiplexing is more practical to read and operate registers, node name corresponding to virtual ISP name slightly different platforms, specific ls /proc/rkisp* view

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 input and output information, process isp->TNR->NR->FEC, rate/time/delay meaning is the same as described above, corresponding to the input or output information of the current node

```
[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: After rv1109, ping pong is used by default, and it is recommended to use ping pong.

Monitor Mode: Monitor mode, after enabling the monitoring mode, vicap is reset in the event that MIPI detects abnormalities.

Input Info: A summary of input information

SRC Subdev: Input device, generally refers to the sensor device, containing camera orientation, index number, device name, I2C bus, 7bit slave address and other information

Interface: Data physical interface, MIPI, LVDS, DVP, etc.

VC Channel: The actual VC channel refers to the virtual channel of multi-channel transmission on the MIPI protocol.

HDR Mode: The working mode of the sensor, divided into normal, hdr_x2, hdr_x3.

format: The input data type

crop.bounds: Clipping parameter, which can be configured .get_selection the sensor driver to properly trim the data from the input source.

Output Info: A summary of the output information

format: The output data type

compact: Default compact output, the relevant definition can be found in the following section:：如何抓取CIS输出的RAW、YUV数据

frame amount:

Early: In the wake up mode, after collecting wait_line rows of data, the buffer is sent to the ISP in advance, the default mode is to collect the full frame and send the ISP for processing, which shows the optimization time of the buffer to the ISP in advance. Wake up mode configuration instructions can be found at: [Delay Optimization Guide] (Delay Optimization Guide)

Single readout: In HDR mode, the transmission time of a single frame, that is, the transmission time of a long frame.

Total Readout: In HDR mode, the time difference between the start of transmission of long frames and the end of transmission of short frames, that is, the original transmission time of a composite frame.

rate: The time between frames.

fps: Frame rate.

IRQ statistics: Interrupt information

total: frame end + err The total number of interrupts

CSI over flow: The number of interrupts for the overflow exception

CSI Bandwidth Lack: The number of interrupts for bandwidth lack exceptions

Frame DMA end: The number of interrupts in the frame end, which is equal to the number of frames output by the sensor from the start of the stream.

18.12 How to troubleshoot preview flickering

To troubleshoot the cause of flickering, first confirm the source of flicker, which can be analyzed from AE log.

AE log printing is turned on as follows:

1. The terminal (serial port or adb shell) executes export persist_camera_engine_log=0x1ff3
2. Run librkaiq.so in the same terminal in step 1, you can go through rkisp_demo, RkLunch.sh and other programs.
3. On the basis of steps 1 and 2, AE log cannot be printed, and the default compilation method may not compile the log in, you can refer to the following modifications:

```
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 contains statistical value MeanLuma, statistical value TmoMeanLuma after TMO, exposure parameters and other information, through which the cause of flicker can be preliminarily analyzed.

```
[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=50.000000,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) ================================
```

Flashing cause:

1、 TMO synthesis caused by flickering, as shown in the figure below, log filtered, you can clearly see that the statistical value of short frame, medium frame has been very stable, but the statistical value after TMO jumps, indicating that TMO related parameters are not applicable in some scenarios, analysis to this step, you can refer to tunning guide document to adjust the parameters, still can not be solved, please contact RK's IQ engineer to assist in processing.

```
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、The statistical value on raw is very stable, and the statistical value after TMO is also stable, but you can still see flickering on the screen, indicating that there is a problem that causes flickering in the subsequent modules of isp, please contact RK engineers for further analysis in this step.

3、flashing appears in time, gain at the same time change, indicating that time gain effective time configuration may have problems, the general sensor time is n+2 effective, gain n+2 n+1 are more common, if you clearly know the effective frame of time, gain, you can fill in the parameters to the iq file test, the following is the xml version of the iq file description, the value 2 means n+2 takes effect, n means that the nth frame header will set the exposure parameters. The JSON version parameters are similar, please consult the document configuration.

```
<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>
```

If the effective frame cannot be determined, there is an AecSyncTest node in the AE module in the iq file for testing, the principle of this module is two sets of exposure parameters, a certain number of frames apart, switching back and forth. You can set the time of the two sets of parameters to the same value and gain to a different value, and then analyze the MeanLuma statistic value of AE log and the corresponding time gain parameter value.

4. If the time is stable when flashing, and the gain value is called, there may be a problem with the conversion formula of gain, and the linearity of the sensor itself may be poor.

4.1 The conversion formula and the conversion description of the sensor are related to the writing of the driver, and the detailed description can be found in Sensor Info 填写指南. You can calculate the time gain conversion to register value on the AE log, and compare it with the time gain register value printed by the driver to see whether the self-calculated register value is consistent with the value calculated by the program, and if it is inconsistent, you need to confirm the conversion formula and the writing method of the driver to see if there is a problem.

4.2 For linearity problems, you can confirm the linearity by grabbing the RAW diagram and obtaining the statistical value of the image with the image viewing tool.

4.2.1 Time Linearity Test:

a. Cover the ground glass (can be replaced by a thin paper towel, the function is to make the whole image evenly lit, in the linear area), fix the gain value of 1, grab the RAW graph of 10ms 20ms 30ms respectively, obtain the statistical value (the statistical value of the general software is 8bit, the range is 0~255), and record the table

b. The lens is completely blackened, grab a RAW plot, the statistical value of this graph is the black level value. (Because the accuracy requirements are not high, the time gain value here is not required, do not exaggerate too much, for example, the gain of the test is set according to 1x, and the RAW graph of the black level is 1000x, so that the impact on the statistical value is relatively large, not desirable)

c. In the table, subtract the black level of step b separately from the statistical values recorded in step a, and make a line chart of the statistical values and exposure time of the minus black level, if it is straight or close to a straight line, the linearity can be considered to be good.

Concentrate:

1.There are supported_modes configuration tables in the driver with vts_def (frame length under default configuration, including field blanking), frame rate, through two parameters can be easily converted exposure time, assuming the frame rate is 30fps, the vts_def is 1200, the frame interval is 1s/30fps=33.333ms then the exposure behavior corresponding to 10ms is 10/33.333*1200=360 lines, and the exposure parameter setting reference [How to set CIS exposure parameters] (How to set CIS exposure parameters)

2.The statistical value of the RAW graph captured by step A should be greater than the black level and less than 180

4.2.2 gain value linearity

a. Cover the ground glass, fix the time value for 10ms, grab the RAW graph of 1x 2x 4x 8x and other gain values respectively, obtain the statistical value (the statistical value of the general software is 8bit, the range is 0~255), record the table, if the statistical value under some gain values is not below 180, you can adjust the time value and test in segments.

b. The lens is completely blackened, grab a RAW plot, the statistical value of this graph is the black level value.

c. In the table, the statistical value recorded in step A is subtracted from the black level of step B, and the statistical value of the black level and gain are made into a line chart, if it is a straight line or close to a straight line, the linearity can be considered to be good.

Concentrate:

1. The statistical value of the RAW graph captured by step A should be greater than the black level and less than 180
2. If you suspect that there is a problem with the linearity under the GAIN value of a certain segment, you can test the linearity of the segment separately, and the linearity test of the complete GAIN interval is not required.

5、 bright environment, such as outdoor sunlight is relatively strong, flickering, there may be a problem with the exposure value and register conversion, such as the application layer believes that 5 lines, through the register conversion, the actual effective may be 4 lines, so that there is a line of brightness deviation, a line of brightness deviation in the outdoor strong light environment is easy to lead to flickering, need to compare the description of the sensor manual on exposure calculation, carefully check whether the implementation of the driver is correct.

18.13 How to troubleshoot purple spillage at light sources

1. Linear mode

In linear mode, the light source is purple, it is possible that the gain value of the sensor is set to an illegal value, causing the image to be abnormal, and it is necessary to check whether the gain value register of the drive conversion meets the restrictions described in the sensor manual.

2. HDR mode

In HDR mode, there are two main reasons:

2.1 Short frame image offset causes HDR compositing to be misaligned. In this case, you can check whether the kernel log has MIPI-related errors. If there is no MIPI error, further confirm whether there is a problem with the exposure parameters set to the sensor, HDR sensors usually have more restrictions on long and short frame exposure, these restrictions can be referred toSensor Info 填写指南. Print out the register values that drive to the sensor and compare them with the restrictions described in the sensor manual to see if any register values are set that do not meet the requirements.

2.2 The ratio of the exposure parameters of the long and short frames does not match the ratio of the actual image, in this case, it is also referred to 2.1 to confirm whether there is a problem with the conversion of the exposure parameters. The more common problem is that most sensors have restrictions on the maximum exposure of short frames, assuming that the maximum exposure of a sensor short frame is 2ms, and the sensor info, AEC parameter related configuration in the IQ file does not configure the maximum short frame, or the short frame maximum limit is set larger than the driver limit, for example, AEC may decompose the short frame exposure 3ms, set to the driver, the actual maximum can only be set to 2ms, but the driver does not directly return an error to AEC , so that AEC thinks that the 3ms setting is successful and passes the exposure parameters to the TMO module, resulting in the composite image ratio is wrong and the brightness is wrong. The place where the short frame is combined is usually the overexposed area, usually manifested at the light source, that is, the purple color at the common light source. Therefore, the image light source is purple, and the focus is to check whether there is a difference between the exposure parameters decomposed by AEC and the exposure parameters actually set to the sensor.

18.14 Sensor Info Fill out the guide

Take IMX290 as an example:

[imx290]

CISAgainRange=1 31.6

CISDgainRange=1 125.89

When the brightness of analog gain(again) is insufficient, digital gain(dgain) is usually used to compensate, and the general practice of rk is to mix dgain to again, and the driver decomposes again and dgain, respectively set to the corresponding sensor register;

The IMX290 manual describes the GAIN value distribution as follows:

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)

That's again 30dB and dgain 42dB

By formula:

db = 20 * log10 (gain multiple)

reg_gain = 20 * log10 (gain multiple) * 10 / 3

Calculate the multiple unit again = 10^(30db/20)=31.6x

Dgain = 10^(42db/20) = 125.89x

CISExtraAgainRange=2 63.2

CISExtraAgainRange is the range value of again * dcg ratio, some sensors support HCG/LCG, HCG can obtain a better signal-to-noise ratio in dark environments, if the driver implements related functions, you need to fill in the corresponding conversion gain value. The IMX290 manual describes that the typical value of conversion efficiency ratio is 2, that is, when setting 2x again, and HCG mode, the actual gain value is 4x, so CISExtraAgainRange=2*[1 31.6], if the driver does not implement HCG/LCG, default fill in [1 1]

CISIspDgainRange=1 1

ISP dgain, which is not currently used, can be used by default

CISMinFps=10

The minimum allowable frame rate, assuming that 5fps needs to be downgraded, and the sensor supports frame reduction to 5fps, this side must also be changed to 5 synchronously to reduce frames through IQ configuration or API.

CISTimeRegMin=1

In linear mode, the smallest unit of exposure line

CISLinTimeRegMaxFac=1.00 2.00

Maximum exposure line in linear mode

CISTimeRegOdevity=1 0

The parity of exposure lines in linear mode can be increased by 1 according to the manual description of SHS1, and the exposure line can also be increased by 1.

The IMX290 manual describes Integration time = 1 frame period - (SHS1 + 1) X(1H period)

The RK framework currently sends the exposure units to the driver from AIQ in unit of line time, if part of the sensor is a half-line unit, it needs to be converted to a line unit, which can be seen from the exposure formula of IMX290 as a row unit, and the above formula is re-described below

Exposure line time = vts -shs1 - 1

From the description of SHS1 in the sensor manual, the limit range is 1~ (Number of lines per frame - 2), that is, 1~(vts-2)

So CISTimeRegMin = vts -shs1 - 1 = vts - (vts-2) - 1 = 1

CISLinTimeRegMaxFac = vts - shs1 - 1 = vts - 1 - 1 = vts - 2

vts is the total number of lines in a frame, including field blanking, different manual descriptions are slightly different, 1 frame period, Number of lines per frame description are vts.

CISHdrTimeRegMin=1

HDR minimum exposure line

CISHdrTimeRegMax=8 0 0

HDR maximum exposure line, increase this variable because some sensors short frame maximum exposure line has restrictions, can not increase with the reduction of long frame exposure, can not increase with the decrease of frame rate, imx290 is such a sensor, according to the Sony standard configuration, short frame exposure line maximum 8 lines, imx307 DOL document has a description of decreasing exposure ratio mode, According to the configuration inside, the maximum exposure line of short frame can be 222 lines, imx290 DOL document is not described, specifically consult Sony whether it is supported.

CISHdrTimeRegOdevity=1.00 0.00

CISHdrTimeRegSumFac=1.00 6.00

The Sony DOL documentation is described as follows:

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：

The table can calculate the minimum exposure value for long frames = FSC-SHS2-1=FSC-(FSC-2)-1=1

Short frame minimum exposure = RHS1-SHS1-1=RHS1-(SHS1-2)-1=1

So the minimum exposure behavior at HDR1

CISHdrTimeRegOdevity: From the table, shs1 and shs2 do not have a similar 2n or 2n+1 limit, so the corresponding exposure rows can be incremented by 1

CISHdrTimeRegSumFac：

Long and short frame exposure sum = (FSC-SHS2-1) + (RHS1-SHS1-1)

SHS2 and SHS1 take the minimum value at the same time, so that the exposure of both long and short frames is maximum

Long and short frame exposures and = (FSC-(RHS1+2)-1) + (RHS1-2-1)=FSC-6

For 2 frames of DOL hdr, FSC=2*vts, so the maximum exposure and maximum value of long and short frames is =2*vts-6

That is, CISHdrTimeRegSumFac=[2 6], but for the convenience of calculation, sony's DOL hdr driver will FSC as the aec uploaded by vts, that is, the uploaded vts has actually been multiplied by 2 times, so CISHdrTimeRegSumFac=[1 6]

CISTimeRegUnEqualEn=0

Whether the time of long and short frames can be equal, due to the IMX290 short frame limit, it cannot be equal in any case

CISHdrGainIndSetEn=1

Whether the gain of long and short frames needs to be set to the same, 1 means that it can be set to different values, 0 means that the long and short frames gain should be the same, depending on the sensor description, some sensor long and short frames share a set of registers, some sensors although the long and short frames gain have separate registers, but the design reason requires two sets of registers to set to the same value, in order to expose the correctness of decomposition, you need to accurately fill in this parameter.

Note: imx290 needs to pay attention to the setting of FPGC PFGC_1 value, which is specifically described in the DOL document.

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

The RK platform supports the multiple of the GAIN value setting and the SONY DB mode of the GAIN value setting, 0 means to use db, corresponding to IMX290 can directly use db mode, you can use the above GainRange decomposition formula, GainRange decomposition formula will be slightly error, after all, the nonlinear curve is decomposed into a multi-segment linear curve.

NonLinear=DB_MODE

PatternMode=RGGB

TimeFactor=0 0 1 0.5

Time decomposition formula, it is recommended to keep this formula, the calculation does not conform to the formula, the sensor drive to do the conversion.

hdr_dcg_ratio=2

normal_dcg_ratio=2

DCG ratio has been described earlier

SensorFlip=0

The default mirror flip state, bit0 mirror, bit1 flip

18.15 Sensor index considerations

The index of the sensor is filled in the DTS, rockchip, camera-module-index
In order to facilitate device finding, librkaiq and rockit preferentially support the index of the sensor as the device index. The tool_server that comes with the Tuning tool is also to find devices according to the sensor index. If you use a similar tool without adding -d to specify the device number, the default is 0.

So it is recommended that

When shooting a single camera, the camera index is set to 0

When dual cameras, the sensor index is equipped with 0/1

When multi-camera, the sensor index is equipped with 0/1/2/3...

19. Appendix A List of CIS drivers V4L2-controls

CID	Description
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.
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.

20. Appendix B MEDIA_BUS_FMT table

CIS sensor type	Sensor outputs 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
Only Y (black and white) is the raw bw sensor	MEDIA_BUS_FMT_Y8_1X8 MEDIA_BUS_FMT_Y10_1X10 MEDIA_BUS_FMT_Y12_1X12

21. Appendix C CIS Reference Driver List

CIS data interface	CIS output data type	Frame/Field	Reference driver
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 sc5239.c hynix_hi556.c 8M ov8858.c

CIS data interface	CIS output data type	Frame/Field	Reference driver
			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
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

CIS data interface	CIS output data type	Frame/Field	Reference driver
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
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(supported)

22. Appendix D VCM driver IC reference driver list

Reference driver
vm149c.c
dw9714.c
dw9718.c
fp5510.c
gt9760s.c
fp5501.c (step motor)
mp6507.c (step motor)
ms41908.c (step motor)

23. Appendix E Flash light driver reference driver list

Reference driver
sgm3784.c
leds-rgb13h.c (GPIO control)

[*CIS Device Registration (DTS)]:*
*[#*CIS Device Registration (DTS)]:
[#CIS Device Registration (DTS)]: