Rockchip_Driver_Guide_ISP2x_CN

File identification: RK-YH-GX-602

Release version: V1.0.3

Date: 2021-02-23

Document Confidentiality: □ Top Secret □ Secret □ Internal Information ■ Public

Disclaimer

This document is provided "as is", Rockchip Microelectronics Co., Ltd. ("the company", the same below) does not make any statement, information and content of this document
The accuracy, reliability, completeness, merchantability, specific purpose and non-infringement of the company provide any express or implied statement or guarantee. This article
The file is only used as a reference for instructions.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without any notice.

Trademark statement

"Rockchip", "Rockchip" and "Rockchip" are all registered trademarks of our company and are owned by our company.

All other registered trademarks or trademarks that may be mentioned in this document are owned by their respective owners.

Rockchip Microelectronics Co., Ltd.

Rockchip Electronics Co., Ltd.

Address: No. 18, Area A, Software Park, Tongpan Road, Fuzhou City, Fujian Province

URL: www.rock-chips.com

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

Customer Service Fax: +86-591-83951833

Customer Service Email: fae@rock-chips.com

Preface

Overview

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

Product version``

Chip name	Kernel version
RV1126/RV1109	Linux 4.19

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 situation
RV1126	Linux (Kernel-4.19)	Y
RV1109	Linux (Kernel-4.19)	Y
RK3566	Linux (Kernel-4.19)	Y
RK3568	Linux (Kernel-4.19)	Y

Revision record

version number	author	Modified date	Modify the description
v0.1.0	Cai Yiwei	2020-06-11	initial version
v1.0.0	Chen Zefa	2020-10-30	Added description of focus, zoom, iris, ircut
v1.0.1	Chen Zefa	2021-01-04	Modify the format error
v1.0.2	Cai Yiwei	2021-01-21	rv1109/rv1126 memory optimization guide
v1.0.3	Huang Jianglong	2021-02-04	Added rkvicap driver description

content

Page 3

Page 4

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

Page 5

|-- drivers/media

|-- platform/rockchip/cif RKCIF driver

|-- platform/rockchip/isp RKISP driver

|-- dev includes 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 rkispp driver

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

|-- stream includes 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

The link relationship between ISP and VICAP

For the RV1126/RV1109 and RK356X platforms, VICAP and ISP are two independent image processing IPs. If the image collected by VICAP needs to be
Through ISP processing, the v4l2 sub device corresponding to the VICAP interface needs to be generated at the driver level to link to the node corresponding to the ISP to provide parameters
The number is used by the ISP driver. Please refer to RKISP driver for ISP driver description , and RKVICAP driver description for VICAP driver . Specific VICAP each connection
The overall block diagram of the connection between the port and the ISP is as follows:

Page 6

RKISP driver

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 Control the power-on and power-off functions of subdevices (such as mipi dphy and sensor).

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

Page 7

name	Types of	describe

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 ddr from sensor,usually using for the hdr long frame
rkisp_rawwr2	v4l2_vdevcapture	Raw image write to ddr from sensor,usually using for the hdr short frame
rkisp_rawwr3	v4l2_vdevcapture	Raw image write to ddr from sensor
rockchip-mipi-dphy-rx	v4l2_subdev	MIPI-DPHY Configure.
rkisp-bridge-ispp	v4l2_subdev	Isp output yuv image to ispp
rkispp_input_image	v4l2_vdevoutput	Yuv image read from ddr to ispp
rkisp-isp-subdev	v4l2_subdev	The format and size on sink pad equal to isp outputThe support max size is 4416x3312, mix size is 66x258
rkispp_m_bypass	v4l2_vdev capture	Full resolution and yuv format

Page 8

name	Types of	describe
rkispp_scale0	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[1 8] ratio, 3264 max width
rkispp_scale1	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width
rkispp_scale2	v4l2_vdev capture	Full or scale resolution and yuv formatScale range:[2 8] ratio, 1280 max width

ISP HDR mode description

RKISP2 supports receiving mipi sensor output hdr 3 frame or 2 frame mode, the hardware collects data to ddr through 3 or 2 dmatx, and then passes

3 or 2 dmarx reads the isp, the isp is combined with 3 or 2 frames, and the driving link 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 driverDriver migration steps .

name	name	describe
rkisp-isp-subdev	Sensor pad0	Isp acquisition 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

RKVICAP driver

Frame description

The RKVICAP driver is mainly based on the v4l2/media framework to implement hardware configuration, interrupt processing, control buffer rotation, and control Power-on and power-off functions 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 at the same time, but mipi and lvds are not the same
The VICAP LITE only has the lvds interface, which can work simultaneously with the VICAP FULL interface; the VICAP FULL dvp interface corresponds to one rkvicap_dvp node, VICAP FULL mipi/lvds interface corresponds to one rkvicap_mipi_lvds node, and VICAP LITE corresponds to one 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, the mipi interface corresponds to a rkvicap_mipi_lvds node (the same name as the VICAP FULL of RV1126/RV1109), each node can Collect 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 sditf node generated by the isp. To be node equipment. The DVP interface corresponds to the rkvicap_dvp_sditf node, and the mipi/lvds interface of VICAP FULL corresponds to rkvicap_mipi_lvds_sditf node, VICAP LITE corresponds to rkvicap_lite_sditf.

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

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

CIS (cmos image sensor) driver

CIS Device Registration (DTS)

Single registration

MIPI interface

For the RV1126 and RV1106 platforms, there are two independent and complete standard physical mipi csi2 dphy, corresponding to the dts csi_dphy0 and csi_dphy1 (see rv1126.dtsi), the characteristics are as follows:

The maximum data lane is 4 lanes;

The maximum rate is 2.5Gbps/lane;

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

Full mode

Only use csi2_dphy0, csi2_dphy0 and csi2_dphy1/csi2_dphy2 are mutually exclusive and cannot be used at the same time;

The maximum data lane is 4 lanes;

The maximum rate is 2.5Gbps/lane;

Split mode

Only use csi2_dphy1 and csi2_dphy2, mutually exclusive with csi2_dphy0, and cannot be used at the same time;

csi2_dphy1 and csi2_dphy2 can be used at the same time;

The maximum data lane of csi2_dphy1 and csi2_dphy2 is 2 lanes;

csi2_dphy1 corresponds to lane0/lane1 of the physical dphy;

csi2_dphy2 corresponds to lane2/lane3 of physical dphy;

Maximum rate 2.5Gbps/lane

For specific dts use cases, see the following examples.

Link to ISP

RV1126/RV1106 platform
Take rv1126 isp and os04a10 as examples below.

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

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

Configuration points

Data-lanes must specify the number of lanes used, otherwise it cannot be recognized as mipi type;

```
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";// Need to be consistent with 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 effective level
        pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
        // Module number, this number should not be repeated
```

```
        rockchip,camera-module-index = <1>;
        // Module orientation, there are "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 of the mipi dphy side
                        remote-endpoint = <&mipi_in_ucam0>;
                        // mipi lane number, 1lane 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 of the sensor
                                remote-endpoint = <&ucam_out0>;
                                // mipi lane number, 1lane 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 of the isp side
                                remote-endpoint = <&isp_in>;
                        };
                };
        };
};

&rkisp {
        status = "okay";
};

&rkisp_vir0 {
        status = "okay";
        ports {
                #address-cells = <1>;
                #size-cells = <0>;
```

Page 13

```
                port@0 {
                        reg = <0>;
                        #address-cells = <1>;
                        #size-cells = <0>;
                        isp_in: endpoint@0 {
                                reg = <0>;
                                // The port name of the mipi dphy side
                                remote-endpoint = <&csidphy0_out>;
                        };
                };
                port@1 {
                        reg = <1>;
                        #address-cells = <1>;
                        #size-cells = <0>;
                        isp0_out: endpoint@1 {
```

```
                                reg = <1>;
                                // ispp port name, isp output to ispp
                                remote-endpoint = <&ispp0_in>;
                        };
                };
        };
};

&rkispp {
        status = "okay";
};

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

RK356X platform

Let's take rk3566 isp and gc8034 4lane as examples for description:

Link relationship: ***sensor->csi2_dphy0->isp***

Configuration points

- Need to configure data-lanes
- Need to enable the csi2_dphy_hw node

```
/* full mode: lane0-3 */
gc8034: gc8034@37 {
        // Need to be consistent with the matching string in the driver
        compatible = "galaxycore,gc8034";
        status = "okay";
        // sensor I2C device address, 7 bits
        reg = <0x37>;
        // sensor mclk source configuration
```

Page 14

```
        clocks = <&cru CLK_CIF_OUT>;
        clock-names = "xvclk";
        //sensor related power domain enable
        power-domains = <&power RK3568_PD_VI>;
        //sensor mclk pinctl settings
        pinctrl-names = "default";
        pinctrl-0 = <&cif_clk>;
        // Reset pin assignment and effective level
        reset-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_LOW>;
        // powerdown pin assignment and effective level
        pwdn-gpios = <&gpio4 RK_PB2 GPIO_ACTIVE_LOW>;
        // Module number, this number should not be repeated
        rockchip,camera-module-index = <0>;
        // Module orientation, there are "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 {
                        // csi2 dphy port name
                        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 simultaneously with 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 of the sensor
            remote-endpoint = <&gc8034_out>;
            // csi2 dphy lane number, 1lane is <1>, 4lane is <1 2 3 4>, and sensor
Consistent
            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 of 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>;
                        // csi2 dphy port name
                        remote-endpoint = <&dphy0_out>;
                };
        };
};
```

Link to VICAP

RV1126/RV1109 platform

Take mipi os04a10 4lane link vicap as an example:

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

Configuration points:

Data-lanes must specify the number of lanes used, otherwise it cannot be recognized as mipi type;

Dphy needs to be linked to the csi host node.

```
os04a10: os04a10@36 {
        // Need to be consistent with 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>;
```

Page 16

```
        clock-names = "xvclk";
        //sensor related power domain enable
        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 effective level
        pwdn-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
        // Module number, this number should not be repeated
        rockchip,camera-module-index = <1>;
        // Module orientation, there are "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 {
                // csi2 dphy port name
                remote-endpoint = <&mipi_in_ucam0>;
                // csi2 dphy lane number, 1lane is <1>, 4lane is <1 2 3 4>
                data-lanes = <1 2 3 4>;
            };
        };
    };

&csi_dphy0 {
    //csi2_dphy0 is not used simultaneously with 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 of the sensor
                remote-endpoint = <&ucam_out0>;
                // csi2 dphy lane number, 1lane is <1>, 4lane is <1 2 3 4>, which must be consistent with the sensor
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
```

Page 17

```
                        // csi2 host port name
                        remote-endpoint = <&mipi_csi2_input>;
                    };
                };
        };
};

&mipi_csi2 {
        status = "okay";

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

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

                        mipi_csi2_input: endpoint@1 {
                                reg = <1>;
                                // csi2 dphy port name
                                remote-endpoint = <&csidphy0_out>;
                                // csi2 host lane number, 1lane is <1>, 4lane is <1 2 3 4>, which must 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 lane number, 1lane is <1>, 4lane is <1 2 3 4>, which must be consistent with the sensor side
                                data-lanes = <1 2 3 4>;
                        };
                };
        };
};

&rkvicap_mipi_lvds {
        status = "okay";

        port {
                /* MIPI CSI-2 endpoint */
                cif_mipi_in: endpoint {
                        // csi2 host port name
                        remote-endpoint = <&mipi_csi2_output>;
                        // The number of lanes on the vicap side, 1 lane is <1>, 4 lanes is <1 2 3 4>, which must be consistent with the sensor side
                        data-lanes = <1 2 3 4>;
                };
        };
};
```

Page 18

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

        port {
                /* sditf endpoint */
                mipi_lvds_sditf: endpoint {
                        //isp virtual device port name
                        remote-endpoint = <&isp_in>;
                        //The lane number of mipi csi2 dphy, consistent with 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>;
                                //vicap sditf endpoint name
                                remote-endpoint = <&mipi_lvds_sditf>;
                        };
                };
        };
};
```

RK356X platform

Take gc5025 2lane linking lane2/lane3 of rk3566 evb2 mipi csi2 dphy as an example:

Link relationship: ***sensor->csi2 dphy->mipi csi host->vicap***

Configuration points

- Data-lanes must specify the number of lanes used, otherwise it cannot be recognized as mipi type;
- 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";
                // Need to be consistent with 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";
```

Page 19

```
                //sensor mclk pinctl settings
                pinctrl-names = "default";
                pinctrl-0 = <&refclk_pins>;
                // Reset pin assignment and effective level
                reset-gpios = <&gpio3 RK_PA5 GPIO_ACTIVE_LOW>;
                // powerdown pin assignment and effective level
                pwdn-gpios = <&gpio3 RK_PB0 GPIO_ACTIVE_LOW>;
                //sensor related power domain enable
                power-domains = <&power RK3568_PD_VI>;
                /*power-gpios = <&gpio0 RK_PC1 GPIO_ACTIVE_HIGH>;*/
                // Module number, this number should not be repeated
                rockchip,camera-module-index = <1>;
                // Module orientation, there are "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 {
                                // csi2 dphy port name
                                remote-endpoint = <&dphy2_in>;
                                // csi2 dphy lane number, 2lane is <1 2>, 4lane is <1 2 3 4>
                                data-lanes = <1 2>;
                        };
                };
};

  &csi2_dphy_hw {
```

```
                status = "okay";
    };

&csi2_dphy2 {
        //csi2_dphy2 is not used simultaneously with 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 of the sensor
                                remote-endpoint = <&gc5025_out>;
                                // csi2 dphy lane number, 2lane is <1 2>, 4lane is <1 2 3 4>, need to contact the sensor
Unanimous
                                data-lanes = <1 2>;
                        };
```

Page 20

```
                };

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

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

&mipi_csi2 {
        status = "okay";

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

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

                        mipi_csi2_input: endpoint@1 {
                                reg = <1>;
                                // csi2 dphy port name
                                remote-endpoint = <&dphy2_out>;
                                // csi2 host lane number, 2lane is <1 2>, 4lane is <1 2 3 4>, need to contact the sensor
Unanimous
                                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 lane number, 1lane is <1>, 4lane is <1 2 3 4>, which must be consistent with the sensor side
                                data-lanes = <1 2>;
                        };
                };
        };
};

&rkvicap_mipi_lvds {
        status = "okay";

        port {
```

Page 21

```
                cif_mipi_in: endpoint {
                        // csi2 host port name
                        remote-endpoint = <&mipi_csi2_output>;
                        // The number of lanes on the vicap end, 2lane is <1 2>, 4lane is <1 2 3 4>, which must be consistent with the sensor end
                        data-lanes = <1 2>;
                };
        };
};

&rkvicap_mipi_lvds_sditf {
        status = "okay";

        port {
                /* MIPI CSI-2 endpoint */
                mipi_lvds_sditf: endpoint {
                        //isp virtual device port name
                        remote-endpoint = <&isp_in>;
                        //The lane number of mipi csi2 dphy, consistent with 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>;
                                //vicap mipi sditf endpoint name
                                remote-endpoint = <&mipi_lvds_sditf>;
                        };
                };
        };
};
```

LVDS interface

Link to VICAP

RV1126/RV1109 platform

Take imx327 4lane as an example, the link relationship is as follows:

Link relationship: ***sensor->csi dphy->vicap***

Configuration points

Dphy does not need to link to the csi host node, otherwise it will cause no data to be received;

Page 22

Data-lanes must specify the specific number of lanes used, otherwise the data will not be received;

The bus-type must be configured to 3, otherwise it will not be recognized as an lvds interface, resulting in link establishment failure;

```
imx327: imx327@1a {
        // Need to be consistent with 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";
        //sensor related power domain enable
        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 effective level
        pwdn-gpios = <&gpio3 RK_PA6 GPIO_ACTIVE_HIGH>;
        // Reset pin assignment and effective level
        reset-gpios = <&gpio1 RK_PD5 GPIO_ACTIVE_HIGH>;
        // Module number, this number should not be repeated
        rockchip,camera-module-index = <1>;
        // Module orientation, there are "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 {
                        // csi2 dphy port name
                        remote-endpoint = <&mipi_in_ucam0>;
                        //csi2 dphy lvds lane number, 1lane is <1>, 4lane is <4>, must be specified
                        data-lanes = <4>;
                        //The type of lvds interface, must be specified
                        bus-type = <3>;
                };
        };
};

&csi_dphy0 {
        //csi2_dphy0 is not used simultaneously with 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>;
```

Page 23

```
                        mipi_in_ucam0: endpoint@1 {
                                reg = <1>;
                                // The port name of the sensor
                                remote-endpoint = <&ucam_out0>;
                                //csi2 dphy lvds lane number, 1lane is <1>, 4lane is <4>, must be specified
                                data-lanes = <4>;
                                //The type of lvds interface, 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 vicap lite
                        remote-endpoint = <&cif_lite_lvds_in>;
                        //csi2 dphy lvds lane number, 1lane is <1>, 4lane is <4>, must be specified
                        data-lanes = <4>;
                        //The type of lvds interface, must be specified
                        bus-type = <3>;
                };
        };
    };
};

&rkvicap_lite_mipi_lvds {
    status = "okay";

    port {
        /* lvds endpoint */
        cif_lite_lvds_in: endpoint {
            // csi2 dphy port name
            remote-endpoint = <&csidphy0_out>;
            //csi2 dphy lvds lane number, 1lane is <1>, 4lane is <4>, must be specified
            data-lanes = <4>;
            //The type of lvds interface, must be specified
            bus-type = <3>;
        };
    };
};

&rkvicap_lite_sditf {
    status = "okay";

    port {
        /* lvds endpoint */
        lite_sditf: endpoint {
            //isp virtual device port name
            remote-endpoint = <&isp_in>;
            //csi2 dphy lane number, consistent with sensor
            data-lanes = <4>;
        };
    };
};
```

Page 24

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

&rkisp_vir0 {
    status = "okay";

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

            isp_in: endpoint@0 {
                reg = <0>;
                //Lite vicap lvds sditf endpoint name
                remote-endpoint = <&lite_sditf>;
            };
        };
    };
};
```

DVP interface

Link to VICAP

On the RV1126/RV1106/RK356X platform, the dts configuration of each related interface of dvp is the same.

BT601

Take ar0230 bt601 as an example, the link relationship is as follows:

Link relationship: ***sensor->vicap***

Configuration points

hsync-active/vsync-active must be configured for asynchronous registration of the v4l2 framework to recognize the BT601 interface. If not configured, it will be recognized as BT656 interface;

pclk-sample/bus-width optional;

You must specify the hsync-acitve/vsync- of the current sensor through the flag in the g_mbus_config interface of the sensor driver

The valid polarity of active/pclk-ative, otherwise the data will not be received;

The pinctrl needs to be quoted to do the corresponding iomux for the bt601 related gpio, otherwise the data will not be received;

The sample code of 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;
}
```

The dts configuration example is as follows:

```
ar0230: ar0230@10 {
```



```
                // Need to be consistent with 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";
                //sensor related power domain enable
                avdd-supply = <&vcc_avdd>;
                dovdd-supply = <&vcc_dovdd>;
                dvdd-supply = <&vcc_dvdd>;
                power-domains = <&power RV1126_PD_VI>;
                // powerdown pin assignment and effective level
                pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
                /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
                //Configure dvp related data pins and clock pins
                pinctrl-names = "default";
                pinctrl-0 = <&cifm0_dvp_ctl>;
                // Module number, this number should not be repeated
                rockchip,camera-module-index = <0>;
                // Module orientation, there are "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>;
                        };
                };
};

&rkvicap_dvp {
        status = "okay";

        port {
                /* Parallel bus endpoint */
                cif_para_in: endpoint {
                        //Sensor endpoint name
                        remote-endpoint = <&cam_para_out1>;
                        //sensor related configuration parameters
```

```
                        bus-width = <12>;
                        hsync-active = <1>;
                        vsync-active = <1>;
                        pclk-sample = <0>;
                    };
                };
        };

        &rkvicap_dvp_sditf {
                status = "okay";

                port {
                        /* parallel endpoint */
                        dvp_sditf: endpoint {
                                //isp virtual device port name
                                remote-endpoint = <&isp_in>;
```

Page 26

```
                                //sensor related configuration parameters
                                bus-width = <12>;
                                hsync-active = <1>;
                                vsync-active = <1>;
                                pclk-sample = <0>;
                        };
                };
        };

        &rkisp {
                status = "okay";
        };

        &rkisp_vir0 {
                status = "okay";

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

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

BT656/BT1120

The dts usage of BT656/BT1120 is the same.

Take ava fpga bt1120 as an example, the link relationship is as follows:

Link relationship: ***sensor->vicap***

Configuration points

Do not configure hsync-active/vsync-active, otherwise the v4l2 framework will recognize it as BT601 when registering asynchronously;

pclk-sample/bus-width optional;

It must be specified in the g_mbus_config interface of the sensor driver through the flag to indicate the effective polarity of the pclk-ative of the current sensor. No It will result in the inability to receive data;

The querystd interface in v4l2_subdev_video_ops must be implemented, indicating that the current interface is an ATSC interface, otherwise it will fail Received data;

The pinctrl needs to be quoted to do the corresponding iomux for the bt656/bt1120 related gpio, otherwise the data will not be received.

The sample code of the g_mbus_config interface is as follows:

Page 27

```
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;
}
```

The dts configuration example is as follows:

```
avafpga: avafpga@70 {
        // Need to be consistent with 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";
        //sensor related power domain enable
        avdd-supply = <&vcc_avdd>;
        dovdd-supply = <&vcc_dovdd>;
        dvdd-supply = <&vcc_dvdd>;
        // powerdown pin assignment and effective level
        power-domains = <&power RV1126_PD_VI>;
        pwdn-gpios = <&gpio2 RK_PA6 GPIO_ACTIVE_HIGH>;
        /*reset-gpios = <&gpio2 RK_PC5 GPIO_ACTIVE_HIGH>;*/
        //Configure dvp related data pins and clock pins
        pinctrl-names = "default";
        pinctrl-0 = <&cifm0_dvp_ctl>;
        // Module number, this number should not be repeated
        rockchip,camera-module-index = <0>;
        // Module orientation, there are "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>;
                };
        };
};

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

Page 28

```
        port {
                /* Parallel bus endpoint */
                cif_para_in: endpoint {
                        //Sensor endpoint name
                        remote-endpoint = <&cam_para_out2>;
                        //sensor related configuration parameters, optional
                        bus-width = <16>;
                        pclk-sample = <1>;
                };
```

```
        };
    };

    &rkvicap_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>;
                    //dvp sditf endpoint name
                    remote-endpoint = <&dvp_sditf>;
                };
            };
        };
    };
```

Multi-sensor registration

A single hardware isp/ispp virtualizes multiple devices and processes multiple sensor data in time division multiplexing.

Link relationship, isp0->ispp0 and isp1->ispp1 are fixed configuration rv1126.dtsi

Mipi into isp or cif into isp is optional.

Page 29

E.g:

sensor0->csi_dphy0->csi2->vicap->isp0->ispp0

sensor1->csi_dphy1->isp1->ispp1

Example reference arch/arm/boot/dts/rv1109-evb-ddr3-v12-facial-gate.dts

gc2053->csi_dphy0->csi2->cif->isp1->ispp1

ov2718->csi_dphy1->isp0->ispp0

The following configuration is very important for different resolutions

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

CIS driver description

Camera Sensor uses I2C to interact with the main control. At present, the sensor driver is implemented in accordance with the I2C device driver. The sensor driver At the same time, the v4l2 subdev method is used to realize the interaction with the host driver.

Brief description of data type

struct i2c_driver

[instruction]

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	describe
@driver	Device driver model driver mainly includes the driver name and the matching device with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registers the device for matching, then the kernel uses the table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

[Example]

Page 30

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

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

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

static int __init sensor_mod_init ( void )
{
        return i2c_add_driver ( & os04a10_i2c_driver );
}

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

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

struct v4l2_subdev_ops

[instruction]

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]

Page 31

Member name	describe
.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 ,
};
```

struct v4l2_subdev_core_ops

[instruction]

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	describe
.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]

Page 32

```
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
};
```

At present, the following private ioctl is used to implement module information query and OTP information query settings.

Private ioctl	describe
RKMODULE_GET_MODULE_INFO	Get module information, refer to detailsstruct rkmodule_inf ;
RKMODULE_AWB_CFG	Switch sensor's compensation function for awb; if the module does not burn golden The awb value can be set here; refer to struct for details rkmodule_awb_cfg ;
RKMODULE_LSC_CFG	Switch sensor's compensation function for lsc; refer to struct for details rkmodule_lsc_cfg ;
PREISP_CMD_SET_HDRAE_EXP	Hdr exposure setting detailed referencestruct preisp_hdrae_exp_s
RKMODULE_SET_HDR_CFG	Set the Hdr mode to switch between normal and hdr modes, and you need to drive For details on the configuration information of the dynamic adaptation normal and hdr 2 groups, please refer to struct rkmodule_hdr_cfg
RKMODULE_GET_HDR_CFG	Get the detailed reference of the current hdr modestruct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	Set the conversion gain of linear mode, such as imx347, The os04a10 sensor has a conversion gain function, such as sensor does not support conversion gain, may not be implemented

struct v4l2_subdev_video_ops

[instruction]

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]

Page 33

Member name	describe
.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 ,
};
```

struct v4l2_subdev_pad_ops

[instruction]

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]

Page 34

Member name	describe
. 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 ,
};
```

struct v4l2_ctrl_ops

[instruction]

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	describe
.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 user controls function provided by the framework, and the cameras sensor driver must implement the following control functions, Refer to CIS driver V4L2-controls list 1

Page 35

struct xxxx_mode

[instruction]

Sensor can support the information of each mode.

This structure can often be seen in the sensor driver, although it is not required by the v4l2 standard.

[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	describe
.bus_fmt	Sensor output format, refer to MEDIA_BUS_FMT table
.width	The effective image width, which needs to be consistent with the width output of the sensor currently configured
.height	The effective image height, which needs to be consistent with the height output of the sensor currently configured
.max_fps	Image FPS, denominator/numerator is fps
hts_def	Default HTS, which is effective image width + HBLANK
vts_def	The default VTS is the effective image height + VBLANK

exp_def	Default exposure time
*reg_list	Register list
.hdr_mode	Sensor working mode, support linear mode, two-frame synthesis HDR, three-frame synthesis HDR
.vc[PAD_MAX]	Configure 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
        },
};
```

struct v4l2_mbus_framefmt

[instruction]

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]

Page 37

Member name	describe
width	Frame width
height	Frame height
code	Refer to MEDIA_BUS_FMT table
field	V4L2_FIELD_NONE: Frame output mode V4L2_FIELD_INTERLACED: Field output mode

[Example]

struct rkmodule_base_inf

[instruction]

Basic module information, 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	describe
sensor	Sensor name, obtained from the sensor driver
module	Module name, obtained from DTS configuration, subject to module data
lens	Lens name, obtained from DTS configuration, subject to module data

[Example]

struct rkmodule_fac_inf

[instruction]

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]

Page 38

Member name	describe
flag	Identifies whether the group information is valid
module	Module name, get the number from OTP, get the module name from the number

lens	Lens name, get the number from OTP, get the lens name from the number
year	Production year, such as 12 for 2012
month	Production month
day	Production Date

[Example]

struct rkmodule_awb_inf

[instruction]

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	**describe**
flag	Identifies whether the group information is valid
r_value	AWB R measurement information of the current module
b_value	AWB B measurement information of the current module
gr_value	AWB GR measurement information of the current module
gb_value	AWB GB measurement information of the current module
golden_r_value	AWB R measurement information of a typical module, if not programmed, set to 0
golden_b_value	AWB B measurement information of a typical module, if not programmed, set to 0
golden_gr_value	AWB GR measurement information of a typical module, if not programmed, set to 0
golden_gb_value	AWB GB measurement information of a typical module, if not programmed, set to 0

[Example]

Page 39

struct rkmodule_lsc_inf

[instruction]

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	describe
flag	Identifies whether the group information is valid
lsc_w	lsc table actual width
lsc_h	lsc table actual height
decimal_bits	lsc The number of decimal places of the measurement information. If it cannot be obtained, set it to 0
lsc_r	lsc r measurement information
lsc_b	lsc b measurement information
lsc_gr	lsc gr measurement information
lsc_gb	lsc gb measurement information

[Example]

struct rkmodule_af_inf

[instruction]

Module OTP af measurement information

[definition]

```
struct rkmodule_af_inf {
        __u32 flag ; // Whether this group of information is a valid flag
        __u32 vcm_start ; // vcm start current
        __u32 vcm_end ; // vcm end current
        __u32 vcm_dir ; // vcm determination direction
} __attribute__ (( packed ));
```

[Key Member]

Page 40

Member name	describe
flag	Identifies whether the group information is valid
vcm_start	vcm starting current
vcm_end	vcm termination current
vcm_dir	vcm measurement direction

[Example]

struct rkmodule_inf

[instruction]

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	describe
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]

struct rkmodule_awb_cfg

[instruction]

Module OTP awb configuration information

[definition]

Page 41

```
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	**describe**
enable	Identifies whether awb correction is enabled
golden_r_value	AWB R measurement information of a typical module
golden_b_value	AWB B measurement information of a typical module
golden_gr_value	AWB GR measurement information of a typical module
golden_gb_value	AWB GB measurement information of a typical module

[Example]

struct rkmodule_lsc_cfg

[instruction]

Module OTP lsc configuration information

[definition]

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

[Key Member]

Member name	**describe**
enable	Identifies whether lsc correction is enabled

[Example]

struct rkmodule_hdr_cfg

[instruction]

hdr configuration information

[definition]

```
struct rkmodule_hdr_cfg {
        __u32 hdr_mode ;
        struct rkmodule_hdr_esp esp ;
} __attribute__ (( packed ));
struct rkmodule_hdr_esp {
```

Page 42

```
        enum hdr_esp_mode mode ;
        union {
                struct {
                        __u32 padnum ;
                        __u32 padpix ;
                } lcnt ;
                struct {
                        __u32 efpix ;
                        __u32 obpix ;
                } idcd ;
        } val ;
};
```

[Key Member]

Member name	describe

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]

struct preisp_hdrae_exp_s

[instruction]

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]

Page 43

Member name	describe
long_exp_reg	Long frame exposure register value

long_gain_reg	Long frame gain register value
middle_exp_reg	Middle frame exposure register value
middle_gain_reg;	Middle 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

[instruction]

In the preisp_hdrae_exp_s structure, you only need to pay attention to a few parameters described by [key member], and the formula for converting exposure and gain values into registers In iq xml, please refer to the iq xml format description for details on how to convert. Conversion gain requires the Sensor itself to support this function, not If you do, you don't need to pay attention to the conversion parameter. For **HDR2X, you should set the passed mid-frame and short-frame parameters into the sensor output Exposure parameter register corresponding to two frames** .

[Example]

API brief description

xxxx_set_fmt

[describe]

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]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
*cfg	Subdev pad information structure pointer	enter
*fmt	Pad-level media bus format structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_get_fmt

[describe]

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]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
*cfg	Subdev pad information structure pointer	enter
*fmt	Pad-level media bus format structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

refer toMEDIA_BUS_FMT table

xxxx_enum_mbus_code

[describe]

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]

Page 45

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
*cfg	Subdev pad information structure pointer	enter
*code	media bus format enumeration structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

The following table summarizes the format corresponding to various image types, refer toMEDIA_BUS_FMT table

xxxx_enum_frame_sizes

[describe]

Enumerate 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]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
*cfg	Subdev pad information structure pointer	enter
*fse	media bus frame size structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_g_frame_interval

[describe]

Get the sensor output fps.

[grammar]

Page 46

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

[parameter]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
*fi	pad-level frame rate structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_s_stream

[describe]

Set stream input and output.

[grammar]

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

[parameter]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
on	1: Start stream output; 0: Stop stream output	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_runtime_resume

[describe]

The callback function when the sensor is powered on.

[grammar]

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

[parameter]

Page 47

parameter name	describe	input Output
*dev	device structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_runtime_suspend

[describe]

The callback function when the sensor is powered off.

[grammar]

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

[parameter]

parameter name	describe	input Output
*dev	device structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_set_ctrl

[describe]

Set the value of each control.

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2_ctrl structure pointer	enter

[return value]

Page 48

return value	describe

0	success
Non-zero	fail

xxx_enum_frame_interval

[describe]

Enumerate 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]

parameter name	describe	input Output
*sd	Sub-device instance	enter
*cfg	pad configuration parameters	enter
*fie	Frame interval parameter	Output

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_g_mbus_config

[describe]

Obtain the supported bus configuration. For example, when using mipi, when the Sensor supports multiple MIPI transmission modes, you can
Upload parameters in MIPI mode.

[grammar]

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

[parameter]

parameter name	describe	input Output
*config	Bus configuration parameters	Output

[return value]

Page 49

return value	describe
0	success
Non-zero	fail

xxxx_get_selection

[describe]

Configure the cropping parameters, the width of the isp input requires 16 alignment, and the height 8 alignment. The resolution of the sensor output does not conform to the alignment or sensor
The output resolution is not the standard resolution, and 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]

parameter name	describe	input Output
*sd	Sub-device instance	enter
*cfg	pad configuration parameters	enter
*sel	Cropping parameters	Output

[return value]

return value	describe
0	success
Non-zero	fail

Driver migration steps

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

1.1 Implement the following members according to the **struct i2c_driver** description:

struct driver.name

struct driver.pm

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:
1). The acquisition of CIS device resources is mainly to parse the resources defined in the DTS file, refer to the Camera Device Registration (DTS) ;

Page 50

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

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

Member name	describe
CIS equipment working parameters	The external independent crystal oscillator solution does not need to be obtained. The RK reference design generally uses the AP output clock. This solution requires
Test clock	To obtain, the general name is xvclk
CIS device control GPIO	For example: Rest pin, Powerdown pin
CIS equipment control circuit source	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 to check the accuracy of the hardware. (This step is not necessary.

1.4) Initialization of CIS v4l2 equipment and media entities;

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

media entity: media_entity_init

2. Refer to the description of **struct v4l2_subdev_ops to** implement the v4l2 sub-device driver, which mainly implements the following 3 members:

struct v4l2_subdev_core_ops
struct v4l2_subdev_video_ops
struct v4l2_subdev_pad_ops

2.1 Refer to the description of **struct v4l2_subdev_core_ops to** implement its callback function, which mainly implements the following callbacks:

.s_power.ioctl

.compat_ioctl32

The RK private control commands mainly implemented by ioctl involve:

Page 51

Member name	describe
RKMODULE_GET_MODULE_INFO	The module information (module name, etc.) defined by the DTS file can be added by this command Upload camera_engine
RKMODULE_AWB_CFG	When the module OTP information is enabled, camera_engine passes this command Transmit the AWB calibration value of the typical module, the CIS driver is responsible for communicating with the current module After the AWB calibration value is compared, the R/B Gain value is generated and set to the CIS MWB Module
RKMODULE_LSC_CFG	When the module OTP information is enabled, camera_engine passes this command Control the LSC calibration value to take effect and enable;
PREISP_CMD_SET_HDRAE_EXP	For details on HDR exposure settings, please refer to struct preisp_hdrae_exp_s
RKMODULE_SET_HDR_CFG	Set HDR mode, can realize normal and hdr switch, need to drive suitable With hdr and normal 2 groups of configuration information, please refer to struct for details rkmodule_hdr_cfg
RKMODULE_GET_HDR_CFG	Get a detailed reference of the current HDR modestruct rkmodule_hdr_cfg
RKMODULE_SET_CONVERSION_GAIN	Set the conversion gain of linear mode, such as imx347, os04a10 sensor with conversion gain function, high conversion The conversion gain can get better signal-to-noise under low illumination Than, if the sensor does not support conversion gain, it may not be implemented

2.2 Refer to the description of **struct v4l2_subdev_video_ops to** implement its callback function, which mainly implements the following callback functions:

Member name	describe
.s_stream	The function of switching data flow, for mipi clk is continuous mode, must be in this callback function Open the data stream within a few minutes. If the data stream is opened in advance, the MIPI LP status will not be recognized
.g_frame_interval	Get the frame interval parameter (frame rate)
.g_mbus_config	Obtain the bus configuration. For the MIPI interface, if the sensor driver supports different lane configurations or supports Support HDR, return MIPI configuration in current sensor working mode through this interface

2.3 Refer to the description of **struct v4l2_subdev_pad_ops to** implement its callback function, which mainly implements the following callback functions:

Member name	describe
.enum_mbus_code	Enumerate data formats supported by the current CIS driver
.enum_frame_size	Enumerate the resolutions supported by the current CIS driver
	RKISP driver obtains the data format output by CIS through this callback, which must be realized;

.get_fmt	Data type output by Bayer raw sensor, SOC yuv sensor, and BW raw sensor Type definition referenceMEDIA_BUS_FMT table supports field output mode, please refer to struct v4l2_mbus_framefmt definition;
.set_fmt	Set the CIS driver output data format and resolution, which must be realized
.enum_frame_interval	Enumerate the frame interval supported by the sensor, including the resolution
.get_selection	Configure the cropping parameters, the width of the isp input requires 16 alignment, and the height 8 alignment

Page 52

2.4 Refer to the description of **struct v4l2_ctrl_ops to** implement, mainly implement the following callbacks

Member name	**describe**
.s_ctrl	RKISP driver and camera_engine realize CIS exposure control by setting different commands;

Refer to CIS driver V4L2-controls list 1Implement various control IDs. The following IDs belong to the information acquisition category. This part is implemented in accordance with the standard Realized by integer menu controls;

Member name	**describe**
V4L2_CID_LINK_FREQ	refer toThe CIS driver V4L2-controls list 1 is defined in the standard. Currently, the RKISP driver is based on this Command to get the MIPI bus frequency;
V4L2_CID_PIXEL_RATE	For MIPI bus: pixel_rate = link_freq * 2 * nr_of_lanes / bits_per_sample
V4L2_CID_HBLANK	refer toCIS driver V4L2-controls list 1 standard definition
V4L2_CID_VBLANK	refer toCIS driver V4L2-controls list 1 standard definition

RK camera_engine will obtain the necessary information through the above command to calculate the exposure, 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 category, and RK camera_engine controls CIS through this type of command

Member name	**describe**
V4L2_CID_VBLANK	Adjust VBLANK, and then adjust frame rate and Exposure time max;
V4L2_CID_EXPOSURE	Set exposure time, unit: number of exposure lines
V4L2_CID_ANALOGUE_GAIN	Set the exposure gain, actually total gain = analog gain*digital gain; single Bit: Gain register value

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

4. CIS reference driver list

VCM driver

Page 53

VCM Device Registration (DTS)

RK VCM driver private parameter description:

name	describe
Start power flow	VCM can just push the module lens to move from the nearest end of the movable stroke of the module lens (module far focus), at this time VCM The output current value of the driver ic is defined as the starting current
Rated electricity flow	The VCM just pushes the module lens to the far end of the movable stroke of the module lens (the module is near focus), at this time the VCM driver The output current value of ic is defined as the rated current
VCM electricity Stream output model	Oscillation will occur during the movement of VCM. VCM driver ic current output changes need to consider the oscillation period of vcm. To minimize the oscillation, the output mode determines the time for the output current to change 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 > ; // starting current of the motor
        rockchip , vcm - rated - current = < 100 > ; // rated current of the motor
        rockchip , vcm - step - mode = < 4 > ; // Current output mode of motor drive ic
        rockchip , camera - module - index = < 0 > ; // module number
        rockchip , camera - module - facing = "back" ; // Module facing, there are "back" and "front"
};

ov13850 : ov13850@10 {
        ......
        lens - focus = <& vm149c > ; // vcm driver setting, this setting is required when supporting AF
        ......
};
```

VCM driver description

Brief description of data type

struct i2c_driver

[instruction]

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 ;
        ...
};
```

Page 54

[Key Member]

Member name	describe
@driver	Device driver model driver mainly includes the driver name and the matching device with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registers the device for matching, then the kernel uses the 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 );
```

struct v4l2_subdev_core_ops

[instruction]

Define core ops callbacks for subdevs.

[definition]

Page 55

```
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	describe
.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
};
```

At present, the following private ioctl is used to query the time information of the motor movement.

RK_VIDIOC_VCM_TIMEINFO

struct v4l2_ctrl_ops

[instruction]

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]

Page 56

Member name	describe
.g_volatile_ctrl	Get a new value for this control. Generally only relevant for 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

API brief description

xxxx_get_ctrl

[describe]

Get the moving position of the motor.

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	Output

[return value]

return value	describe

0	success
Non-zero	fail

xxxx_set_ctrl

[describe]

Set the moving position of the motor.

[grammar]

Page 57

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_ioctl xxxx_compat_ioctl

[describe]

The realization function of custom ioctl mainly includes obtaining the time information of motor movement,

Implemented a 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	describe	input Output
*sd	v4l2 subdev structure pointer	enter
cmd	ioctl command	enter
*arg/arg	Parameter pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

Page 58

Driver migration steps

1.Implement the standard i2c sub-device driver part.

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

struct driver.name

struct driver.pm

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:

1) VCM equipment resource acquisition, mainly to obtain DTS resources, referenceVCM Device Registration (DTS)

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly to provide equipment parameters and Camera equipment
Make a match.

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

2) Initialization of VCM v4l2 device and media entity.

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

media entity: 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 entire movable stroke of the module lens is driven by the VCM.
The corresponding change range on the dynamic current is [starting current, rated current], and it is recommended to realize the mapping conversion relationship between the two in this function;

2. Implement v4l2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to the **v4l2_subdev_core_ops** description to implement the callback function, which mainly implements the following callback functions:

.ioctl.compat_ioctl32

This callback mainly implements RK private control commands, involving:

Member name	describe
RK_VIDIOC_VCM_TIMEINFO	The camera_engine obtains the time required for the lens movement through this command, and accordingly Determine when the lens is stopped and whether the CIS frame exposure time period is consistent with the lens movement time period Overlap; lens movement time and lens movement distance, VCM driver ic current output mode 式 related.

2.2 Refer to the **v4l2_ctrl_ops** description to implement the callback function, which mainly implements the following callback functions:

.g_volatile_ctrl.s_ctrl

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

Page 59

Member name	describe
V4L2_CID_FOCUS_ABSOLUTE	camera_engine uses this command to set and get the absolute position of the lens, RK The AF algorithm defines the position parameter of the entire movable stroke of the lens as [0,64].

FlashLight driver

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 camera module facing
                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 = < 299.2 thousand > ; // Maximum current mode Torch
                        Flash - max - MicroAmp = < 1.122 million > ; // Maximum Current Flash Mode
                        flash - max - timeout - us = < 1600000 > ; //falsh maximum time
                };
                sgm3784_led1 : led@1 { //led1 device information
                        reg = < 0x1 > ; //index
                        LED - max - MicroAmp = < 299.2 thousand > ; // Maximum current mode Torch
                        Flash - max - MicroAmp = < 1.122 million > ; // Maximum Current Flash Mode
                        flash - max - timeout - us = < 1600000 > ; //falsh maximum time
                };
        };
        ...
        ov13850 : ov13850@10 {
                ...
                flash - leds = <& sgm3784_led0 & sgm3784_led1 > ; //The flash device is hooked 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 > ;
      ...
    }
}
```

be careful:

1. The software needs to distinguish the processing flow according to the type of fill light. If it is an infrared fill light, the label of the dts fill light node needs to have the word ir Used to identify the hardware type, just remove the ir field from the led fill light.

2. There are two situations for this kind of single-pin controlled hardware circuit. One is to fix the brightness and directly use gpio to control. Another one One is the brightness controllable, use pwm, set the brightness by adjusting the duty cycle, dts pwms or enable-gpio, choose one of the two configurations.

FLASHLight driver description

Brief description of data type

struct i2c_driver

[instruction]

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	describe
@driver	Device driver model driver mainly includes the driver name and the matching device with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	List of I2C devices supported by this driver If the kernel does not use of_match_table and dts registers the device for matching, then the kernel uses the table for matching
@probe	Callback for device binding
@remove	Callback for device unbinding

Page 61

[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 );
```

struct v4l2_subdev_core_ops

[instruction]

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	describe
.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.

Page 62

[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 the flash lighting time information.

RK_VIDIOC_FLASH_TIMEINFO

struct v4l2_ctrl_ops

[instruction]

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	describe
.g_volatile_ctrl	Get a new value for this control. Generally only relevant for 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 ,
    }
};
```

Page 63

API brief description

xxxx_set_ctrl

[describe]

Set the flash mode, current and flash timeout time.

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_get_ctrl

[describe]

Get the flash fault status.

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_ioctl xxxx_compat_ioctl

[describe]

The realization function of custom ioctl mainly includes the time information of the flash light,

Implemented a custom RK_VIDIOC_COMPAT_FLASH_TIMEINFO.

Page 64

[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	describe	input Output

*sd	v4l2 subdev structure pointer	enter
cmd	ioctl command	enter
*arg/arg	Parameter pointer	Output

[return value]

return value	**describe**
0	success
Non-zero	fail

Driver migration steps

For ordinary gpio to directly control leds, please refer to kernel/drivers/leds/leds-rgb13h.c and

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

The flashlight driver IC can be transplanted as follows

1.Implement the standard i2c sub-device driver part.

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

struct driver.name

struct driver.pm

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:

1) Acquisition of flashlight device resources, mainly to obtain DTS resources, refer to FLASHLIGHT device registration (DTS);

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly to provide equipment parameters and Camera equipment Make a match.

2) Flash device name:

For dual led flash, use led0 and led1 device names 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) Initialization of FLASH v4l2 device and media entity.

v4l2 sub-device: v4l2_i2c_subdev_init, RK flashlight driver requires subdev to have its own device node for user mode Camera_engine access, through the device node to achieve led control;

media entity: media_entity_init;

2. Implement v4l2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to the **v4l2_subdev_core_ops** description to implement the callback function, which mainly implements the following callback functions:

.ioctl.compat_ioctl32

This callback mainly implements RK private control commands, involving:

Member name	describe
RK_VIDIOC_FLASH_TIMEINFO	The camera_engine obtains the time when the LED is on through this command, and judges accordingly Whether the CIS frame exposure time is after the flash is on.

2.2 Refer to the **v4l2_ctrl_ops** description to implement the callback function, which mainly implements 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	describe
V4L2_CID_FLASH_FAULT	Get flash fault information
V4L2_CID_FLASH_LED_MODE	Set Led mode V4L2_FLASH_LED_MODE_NONE V4L2_FLASH_LED_MODE_TORCH V4L2_FLASH_LED_MODE_FLASH
V4L2_CID_FLASH_STROBE	Control the flash
V4L2_CID_FLASH_STROBE_STOP	Control the flash off
V4L2_CID_FLASH_TIMEOUT	Set the maximum continuous light time of flash mode
V4L2_CID_FLASH_INTENSITY	Set flash mode current
V4L2_CID_FLASH_TORCH_INTENSITY	Set torch mode current

Page 66

FOCUS ZOOM P-IRIS driver

The drive here refers to the auto focus (FOCUS), zoom (ZOOM), and auto iris (P-IRIS) controlled by a stepping motor. Due to the stepping used
The motor control method is the same as the hardware design factor, and the three function drives are integrated into one drive. According to the driver chip used, such as SPI controlled chip, the driver can be packaged into SPI frame sub-device. This chapter describes the driver needs to be implemented around the MP6507 driver chip Data structure, framework and precautions.

FOCUS ZOOM P-IRIS 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
```

Page 67

```
                                      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:

Page 68

Member name	describe
rockchip,camera-module-index	Camera serial number, the field that matches the camera
rockchip,camera-module-facing	Camera orientation, field matching camera
iris_en-gpios	IRIS enable GPIO
focus_en-gpios	focus enable GPIO
zoom_en-gpios	zoom enable GPIO
rockchip,iris-step-max	P-IRIS stepper motor moves the maximum number of steps
rockchip,focus-step-max	The maximum number of steps that the focus stepper motor can move
zoom-step-max	The maximum number of steps that the zoom stepper motor can move
iris-start-up-speed	Starting speed of the stepper motor used by IRIS
focus-start-up-speed	Start speed of the stepper motor used by focus
focus-max-speed	Maximum operating speed of the stepper motor used by focus
zoom-start-up-speed	Start speed of stepper motor used by zoom
zoom-max-speed	Maximum operating speed of the stepper motor used by zoom
focus-first-speed-step	Focus is the number of steps to run at the start speed, and the number of steps is increased in proportion to the subsequent acceleration interval, so that each speed segi The running time is as close to the same as possible
zoom-first-speed-step	The number of steps in zoom start speed operation, the subsequent acceleration interval increases the number of steps proportionally, so that each speed segment The running time is as close to the same as possible
focus-speed-up-table	The focus acceleration curve uses a table lookup method, adjusts the parameters to generate an acceleration curve, and accelerates the generated trapezoid The data table configuration of the curve or S-shaped acceleration curve comes in. If you do not configure or configure a single data, press directly The starting speed runs at a constant speed; the minimum value of the acceleration curve does not exceed the maximum starting speed of the motor, and the maximum Over the maximum operating speed of the stepping motor.
focus-speed-down-table	focus deceleration curve, the maximum value of the deceleration curve must be less than the maximum value of the acceleration curve; if the acceleration curve has nc Effective, the deceleration curve is also invalid, and the whole process runs at a constant speed at the starting speed; if the deceleration curve is not configured Line, the deceleration curve is obtained symmetrically from the acceleration curve.
zoom-speed-up-table	The zoom acceleration curve adopts the table lookup method, adjusts the parameters to generate the acceleration curve, and accelerates the generated trapezoid The data table configuration of the curve or S-shaped acceleration curve comes in. If you do not configure or configure a single data, press directly The starting speed runs at a constant speed; the minimum value of the acceleration curve does not exceed the maximum starting speed of the motor, and the maximum Over the maximum operating speed of the stepping motor.

Page 69

Member name	describe
zoom-speed-down-table	zoom deceleration curve, the maximum value of the deceleration curve must be less than the maximum value of the acceleration curve; if the acceleration curve has nc Effective, the deceleration curve is also invalid, and the whole process runs at a constant speed at the starting speed; if the deceleration curve is not configured Line, the deceleration curve is obtained symmetrically from the acceleration curve.

Brief description of data type

struct platform_driver

[instruction]

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	describe
@driver	The struct device_driver driver mainly contains the driver name and matching with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	If the kernel does not use of_match_table and dts to register the device for matching, the kernel uses this table to match
@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 ,
```

Page 70

```
                . of_match_table = of_match_ptr ( motor_dev_of_match ),
        },
        . probe = motor_dev_probe ,
        . remove = motor_dev_remove ,
};
module_platform_driver ( motor_dev_driver );
```

struct v4l2_subdev_core_ops

[instruction]

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	describe
.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 ,
};
```

struct v4l2_ctrl_ops

[instruction]

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	describe
.g_volatile_ctrl	Get a new value for this control. Generally only relevant for 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 ,
};
```

API brief description

xxxx_set_ctrl

[describe]

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

The following v4l2 standard commands are implemented:

Member name	describe
V4L2_CID_FOCUS_ABSOLUTE	Control the focus, 0 means the minimum focal length, clear near
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the zoom factor is the smallest and the angle of view is the largest

V4L2_CID_IRIS_ABSOLUTE	Control the size of the P aperture opening, 0 means the aperture is closed

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	enter

[return value]

Page 72

return value	describe
0	success
Non-zero	fail

xxxx_get_ctrl

[describe]

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

The following v4l2 standard commands are implemented:

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

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_ioctl xxxx_compat_ioctl

[describe]

Customize the realization function of ioctl, which mainly includes the time information of obtaining focus, zoom, and P aperture (time to start and end the movement) Poke), because the lens used does not have a positioning device, it is necessary to reset the position of the lens motor when necessary.

Implemented customization:

Page 73

Member name	describe
RK_VIDIOC_VCM_TIMEINFO	Focusing time information, used to confirm whether the current frame is effective after focusing frame
RK_VIDIOC_ZOOM_TIMEINFO	Time information of zooming, used to confirm whether the current frame is valid after zooming is completed frame
RK_VIDIOC_IRIS_TIMEINFO	Time information of the aperture, used to confirm whether the current frame is effective after aperture adjustment frame
RK_VIDIOC_FOCUS_CORRECTION	Focus position correction (reset)
RK_VIDIOC_ZOOM_CORRECTION	Zoom position correction (reset)
RK_VIDIOC_IRIS_CORRECTION	Iris 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]

parameter name	describe	input Output
*sd	v4l2 subdev structure pointer	enter
cmd	ioctl command	enter
*arg/arg	Parameter pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

Driver migration steps

For SPI-controlled driver chips, you can use the SPI framework for device driver migration, and the RK reference driver uses MP6507, which can be used directly
The pwm outputs the control waveform, and the power is amplified by MP6507, so it is directly transplanted to the platform framework.
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 realized:

struct driver.name

Page 74

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:

1) Acquisition of equipment resources, mainly to obtain DTS resources, refer to FOCUS ZOOM P-IRIS Equipment Registration (DTS) ;

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly to provide equipment parameters and Camera equipment
Make a match.

1.2) Obtain the PWM configuration. According to the control mode of the motor, the phase difference of AB phase is 90 degrees, which can be achieved by aligning the center of the PWM setting of p
Configure center-aligned on the dts pwm node, see for detailsFOCUS ZOOM P-IRIS device registration (DTS) ;

1.3) To obtain the enable pin, MP6507 needs to use 4 PWMs to generate the stepping motor control waveform. Due to the limited hardware PWM, the focus,
The three stepping motors of zoom and P iris are driven by an MP6507 driver, so the corresponding MP6507 driver is enabled through gpio
In this way, pwm time-sharing multiplexing can be realized. Of course, this also has a drawback. Only one stepper motor can be driven at the same time, and the other two stepper motors can be driven
Need to wait for the end of the previous operation to continue the operation;
1.4) Obtain hardware-related restrictions and resources such as the maximum step, maximum starting speed, maximum operating speed, acceleration curve data of each motor, etc.
source;

2) hrtimer_init, timer initialization, pwm uses continuous mode, which requires timer timing to reach the specified output
After the number of pwm waveforms, enter the timer interrupt to turn off pwm, and the acceleration process also needs to enter the timer interrupt after running to the specified number of waveforms
Modify the pwm frequency to achieve the acceleration of the stepper motor;

3) init_completion, the synchronization mechanism is realized through completion, only the previous motor movement operation ends, and the next motor operation
Work in order to proceed;

4) Initialization of v4l2 device and media entity.

v4l2 sub-device: v4l2_i2c_subdev_init, the driver requires subdev to have its own device node for user mode rkaiq to access, through this
The equipment node realizes the control of the motor;

media entity: media_entity_init;

5) Flash device name:

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

2. Implement v4l2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to the **v4l2_subdev_core_ops** description to implement the callback function, which mainly implements the following callback functions:

Page 75

```
. ioctl
. compat_ioctl32
```

This callback mainly implements RK private control commands, involving:

Member name	describe
RK_VIDIOC_VCM_TIMEINFO	Focusing time information, used to confirm whether the current frame is effective after focusing frame
RK_VIDIOC_ZOOM_TIMEINFO	Time information of zooming, used to confirm whether the current frame is valid after zooming is completed frame
RK_VIDIOC_IRIS_TIMEINFO	Time information of the aperture, used to confirm whether the current frame is effective after aperture adjustment frame
RK_VIDIOC_FOCUS_CORRECTION	Focus position correction (reset)
RK_VIDIOC_ZOOM_CORRECTION	Zoom position correction (reset)
RK_VIDIOC_IRIS_CORRECTION	Iris position correction (reset)

2.2 Refer to the **v4l2_ctrl_ops** description to implement the callback function, which mainly implements the following callback functions:

.g_volatile_ctrl

.s_ctrl

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

parameter name	describe
V4L2_CID_FOCUS_ABSOLUTE	Control the focus, 0 means the minimum focal length, clear near
V4L2_CID_ZOOM_ABSOLUTE	Control the zoom factor, 0 means the zoom factor is the smallest and the angle of view is the largest
V4L2_CID_IRIS_ABSOLUTE	Control the size of the P aperture opening, 0 means the aperture is closed**

3. Reference for stepping motor acceleration curve:

3.1 Trapezoidal curve

You can simply accelerate and decelerate at equal intervals and speeds as shown in the figure.

3.2 S-curve

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

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

in,

Vmin refers to the motor starting speed

Vmax refers to the target speed of the motor

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

i is the speed segment number, if it is divided into 32 segments to accelerate, the value is 0~31

Num is half of the number of speed segments. If divided into 32 segments, num is 16

DC-IRIS driver

Compared with P-IRIS, DC-IRIS cannot accurately know the size of the aperture opening. The general use scene is fully opened by default. When the exposure is adjusted to the minimum
When the image is still over-exposed, enter the aperture adjustment. When the exposure is set to the maximum, the image is still under-exposed, enter the aperture adjustment. DC-IRIS motor
It is a DC motor that buffers the adjustment speed of the motor through the negative feedback of the Hall device. For the drive, as long as the motor is controlled by a pwm
When the PWM duty cycle is less than 20%, the iris will slowly close until it is completely closed. The smaller the duty cycle, the faster the iris closes;
The aperture will open slowly when the duty cycle is greater than 40%, the larger the duty cycle, the faster the opening speed; the aperture in the 20%~40% interval is in a hold state.
The 20% and 40% here are not fixed values, and are related to the frequency of pwm and the accuracy of the actual hardware devices.
Reference driver: /kernel/drivers/media/i2c/hall-dc-motor.c

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 > ;
                ...
         }
}
```

Page 77

Brief description of data type

struct platform_driver

[instruction]

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	describe
@driver	The struct device_driver driver mainly contains the driver name and matching with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	If the kernel does not use of_match_table and dts to register the device for matching, the kernel uses this table to match
@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 );
```

Page 78

struct v4l2_subdev_core_ops

[instruction]

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	describe
.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 ,
};
```

struct v4l2_ctrl_ops

[instruction]

The control operations that the driver has to provide.

[definition]

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

[Key Member]

Page 79

member name	describe
.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 ,
};
```

API brief description

xxxx_set_ctrl

[describe]

Call the standard v4l2_control iris position, the DC iris actually cannot know the specific position of the iris, the value set here is the duty of pwm Compare.

The following v4l2 standard commands are implemented:

parameter name	describe
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	describe	input Output
*ctrl	v4l2 control structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_ioctl xxxx_compat_ioctl

[describe]

Currently, there is no private definition to be implemented, and v4l2 framework registration is required to implement empty functions.

[grammar]

Page 80

```
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	describe	input Output
*sd	v4l2 subdev structure pointer	enter
cmd	ioctl command	enter
*arg/arg	Parameter pointer	Output

[return value]

return value	describe
0	success

Non-zero	fail

Driver migration steps

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 realized:

struct driver.name

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:

1) Acquisition of equipment resources, mainly to obtain DTS resources, refer to DC-IRIS Equipment Registration (DTS) ;

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly to provide equipment parameters and Camera equipment Make a match.

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

2) Initialization of v4l2 device and media entity.

v4l2 sub-device: v4l2_i2c_subdev_init, the driver requires subdev to have its own device node for user mode rkaiq to access, through this The equipment node realizes the control of the motor;

media entity: media_entity_init;

Page 81

3) Flash device name:

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

2. Implement v4l2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to the **v4l2_subdev_core_ops** description to implement the callback function, which mainly implements the following callback functions:

```
ioctl
. compat_ioctl32
```

The callback currently does not need to implement specific commands, but as a sub-device of v4l2, the operation function must be implemented, so an empty function is implemented here number.

2.2 Refer to the **v4l2_ctrl_ops** description to implement the callback function, which mainly implements 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	**describe**
V4L2_CID_IRIS_ABSOLUTE	Set the duty cycle of pwm that controls the aperture, range (0~100)

RK-IRCUT driver

The IRCUT is controlled by two wires. A 3.5v~6v power supply is applied to the two wires.

The electrical time is 100ms±10%, which can realize IRCUT switching. The drive controls the current output direction of the motor driver through two gpio, gpio The commands are open (red line) and close (black line). The current flows from open to close, which is an infrared cut-off filter, working during the day;

The flow flows from close to open. It is a white glass sheet and works at night.

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 > ;
        ...
    }
```



```
}
```

Brief description of data type

struct platform_driver

[instruction]

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	describe
@driver	The struct device_driver driver mainly contains the driver name and matching with the DTS registered device of_match_table. When the compatible field in of_match_table and the compatible field in the dts file When there is a match, the .probe function will be called
@id_table	If the kernel does not use of_match_table and dts to register the device for matching, the kernel uses this table to match
@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 ,
};
```

Page 83

```
module_platform_driver ( ircut_driver );
```

struct v4l2_subdev_core_ops

[instruction]

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	describe
.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 ,
};
```

struct v4l2_ctrl_ops

[instruction]

The control operations that the driver has to provide.

[definition]

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

[Key Member]

Page 84

member name	describe
	Actually set the new control value. s_ctrl is compulsory. The ctrl->handler->lock is

.s_ctrl	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 ,
};
```

API brief description

xxxx_set_ctrl

[describe]

Call standard v4l2_control to switch IRCUT.

The following v4l2 standard commands are implemented:

parameter name	describe
V4L2_CID_BAND_STOP_FILTER	0 is CLOSE state, infrared light can enter; 3 is OPEN state, infrared light cannot enter;

[grammar]

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

[parameter]

parameter name	describe	input Output
*ctrl	v4l2 control structure pointer	enter

[return value]

return value	describe
0	success
Non-zero	fail

xxxx_ioctl xxxx_compat_ioctl

[describe]

Currently, there is no private definition to be implemented, and v4l2 framework registration is required to implement empty functions.

Page 85

[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	describe	input Output
*sd	v4l2 subdev structure pointer	enter
cmd	ioctl command	enter
*arg/arg	Parameter pointer	Output

[return value]

return value	describe
0	success
Non-zero	fail

Driver migration steps

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 realized:

struct driver.name

struct driver. of_match_table

probe function

remove function

1.2 Detailed description of the probe function implementation:

1) Acquisition of equipment resources, mainly to obtain DTS resources, refer to RK-IRCUT Equipment Registration (DTS);

1.1) RK private resource definition, naming methods such as rockchip, camera-module-xxx, mainly to provide device parameters and Camera settings Prepare to match.

1.2) Obtain open and close gpio resources;

2) init_completion, the synchronization mechanism is realized through completion, since it takes about 100ms to switch IRCUT, completion is required Synchronization mechanism to ensure that the last IRCUT switch has been completed before the operation can be performed again;

3) Create a work queue and place the switching operation on the work queue to avoid long-term blocking;

4) Initialization of v4l2 device and media entity.

Page 86

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

media entity: media_entity_init;

```
sd -> entity . function = MEDIA_ENT_F_LENS ;
sd -> entity . flags = 1 ; //flag is fixed to 1, used to distinguish other sub-devices of MEDIA_ENT_F_LENS type
```

5) Device name:

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

2. Implement v4l2 sub-device driver, mainly implement the following 2 members:

```
struct v4l2_subdev_core_ops
struct v4l2_ctrl_ops
```

2.1 Refer to the **v4l2_subdev_core_ops** description to implement the callback function, which mainly implements the following callback functions:

```
. ioctl
. compat_ioctl32
```

This callback currently does not need to implement private commands, but v4l2 framework registration requires it, so it implements an empty function, and you can add functions according to your nee content.

2.2 Refer to the **v4l2_ctrl_ops** description to implement the callback function, which mainly implements the following callback functions:

.s_ctrl

.s_ctrl implements the following commands with standard v4l2 control:

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

media-ctl v4l2-ctl tool

The media-ctl tool operates through media devices such as /dev/medio0. It manages the nodes in the Media topology.
format, size, link.

The v4l2-ctl tool is for video devices such as /dev/video0 and /dev/video1. It performs set_fmt,
A series of operations such as reqbuf, qbuf, dqbuf, stream_on, and stream_off.

For specific usage, please refer to the help information of the command. The following are some common usages.

1. Print topology

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

Note: There are many device nodes in isp2, and media0/media1/media2 nodes may exist. You need to enumerate and view device information one by one.

2. Link

```
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 ispp, link to main_path, grab the raw image from main_path, media-ctl does not add -d to specify the device, default
If it is the /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 mounted on.

4. Set fmt and grab the frame

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

5. Set exposure, gain and other controls

```
V4L2 - CTL - D / dev / Video3 - SET - Ctrl 'Exposure = 1216, 10 = analogue_gain'
```

Note: The isp driver will call the control command of the camera sub-device, so the specified device is video3 (main_path or self_path)
It can be set to exposure, vicap will not call the control command of the camera sub-device, and setting the control command directly on the acquisition node will fail.
defeat. The correct way is to find the camera device node is /dev/v4l-subdevX and directly configure the terminal node.

rv1109/rv1126 memory optimization guide

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 reads raw data from ddr for processing

Occupied memory: 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 the aiq library code hwi/isp20/CamHwIsp20.h, 3 at least.

#define ISP_TX_BUF_NUM 4

```
#define VIPCAP_TX_BUF_NUM 4
```

2. DDR_2: isp fbc yuv420 and gain data are written to ddr, and ispp reads from ddr for processing

Occupied memory: buf_size * buf_cnt

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

buf_cnt : 4 bufs in tnr 3to1 mode, 3 bufs in 2to1 mode, the mode is configured in iq xml

3. DDR_3: ispp tnr fbc yuv420 and gain data written to ddr, ispp NR&Sharp reads and processes from ddr again

Occupied memory: buf_size * buf_cnt

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

buf_cnt: 2, which is the smallest

4. DDR_4: ispp NR&Sharp yuyv data is written to ddr, and ispp fec is read from ddr for processing

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

buf_size: width * height * 2

buf_cnt: 2, which is the smallest

5. DDR_5: ispp 4-channel output image buffer, the buffer size is calculated according to the resolution, format and **buf_cnt** set by the user

The above **buf_cnt** is where the memory can be optimally configured

FAQ

How to get the driver version number

Obtained from the kernel startup log

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

Obtained by

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

How to judge the loading status of RKISP driver

If the RKISP driver is successfully loaded, video and media devices will exist in the /dev/ directory. There may be multiple /dev/videos in the system For the device, the video node registered by RKISP can be queried through /sys.

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

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

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


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

If you find that the kernel does not have a log of Async subdev notifier completed, please check whether the sensor has related reports first.
Wrong, whether the I2C communication is successful.

How to capture yuv data output by ispp

Ispp input data sources rkisp_mainpath, rkisp_selfpath and rkispp_input_image link are closed, rkisp-bridge-ispp
Link is on, rkisp-isp-subdev pad2: Source format must be fmt:YUYV8_2X8, the default state does not need to configure link, please refer to
The command is as follows,

```
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 - the mmap = . 3 - Stream - to = / tmp / NV12 . OUT - Stream - COUNT = 20 is - STRAM - poll
```

How to capture Bayer Raw data output by Sensor

The reference command is as follows,

```
media - ctl - d / dev / media0 - set - v4l2 '"m01_f_os04a10 1-0036-
1":0[fmt:SBGGR12_1X12/2688x1520]'
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 - l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[0]'

media - ctl - l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[1]'
V4L2 - CTL - D / dev / video0 - SET - Ctrl 'Exposure = 1216, 10 = analogue_gain' \
- set - selection = target = crop , top = 0 , left = 0 , width = 2688 , height = 1520 \
- set - fmt - video = width = 2688 , height = 1520 , pixelformat = BG12 \
- Stream - the mmap = . 3 - Stream - to = / tmp / MP . RAW . OUT - Stream - COUNT = . 1 - Stream - poll
```

Note:

1. Specify the media node: -d /dev/media0 It depends on the media node configuration actually mounted on the subsequent nodes.

2. "m01_f_os04a10 1-0036-1" is to set the resolution of the sensor. If the sensor driver supports multiple resolutions, you can use this
Command to cut the resolution.

3. "rkisp-isp-subdev": 0, 0 refers to pad0, the input port of isp; "rkisp-isp-subdev": 2, 2 refers to pad2, the output of isp
Port, configure the input and output ports of the isp according to the actual format required.

4. It should be noted that although the ISP does not process the Raw image, it still fills the low bits of the 12-bit data with 0 into 16 bits. Regardless of
The sensor input is 10bit/12bit, and finally the upper layer gets 16bit per pixel.

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 8 bit output )

MEDIA_BUS_FMT_Y10_1X10 ( sensor 10 bit output )

MEDIA_BUS_FMT_Y12_1X12 ( sensor 12 bit output )
```

That is, the above format is returned in the functions xxxx_get_fmt and xxxx_enum_mbus_code.

RKISP driver will make 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, SP Path can only be used, because only SP Path can support Y8 fc nat output.

How to support odd and even field synthesis

RKISP driver supports odd and even field synthesis function, restriction requirements:

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

"Only selfpath support interlaced"

RKISP_selfpath information can be viewed with 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 driver format.field needs to be set to V4L2_FIELD_INTERLACED, which means that the current device output format is an even and odd field, that is,
The function xxxx_get_fmt returns the format.field format. Can refer to driver/media/i2c/tc35874x.c driver;

How to view debug information

1. Check the media pipeline information, this corresponds to the dts camera configuration

```
media - ctl - p - d / dev / mediaX ( X = 0 , 1 , 2 ..)
```

2. View the proc information, this is the pre-isp/ispp single state and frame input and output information, you can cat several times

```
cat / proc / rkisp *
```

Page 91

3. View the driver debug information, set the debug level to 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 ; 0 is off )
echo n > / sys / module / video_rkispp / parameters / debug
```

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

```
IO - . 4 - L 0x10000 0xffb50000 > / tmp / ISP . REG
IO - . 4 - L 0x1000 0xffb60000 > / tmp / the ISPP . REG
```

5. Steps to provide debug information

1) Problem site 1->2->4->3

2) Reproduce the problem 3->Start->Reproduce->1->2->4

6, proc information description

```
[ root@RV1126_RV1109 : / ] # cat / proc / rkisp0
rkisp0     Version : v00 . 01.07
Input      rkcif_mipi_lvds Format : SGBRG10_1X10 Size : 3840 x2160@20fps Offset ( 0 , 0 )
| RDBK_X2 ( frame : 1378 rate : 49 ms )
Output     rkispp0 Format : FBC420 Size : 3840 x2160 ( frame : 1377 rate : 51 ms )
Interrupt  Cnt : 6550 ErrCnt : 0
clk_isp 594000000
aclk_isp 500000000
hclk_isp 250000000
DPCC0 ON ( 0x40000005 )
DPCC1 ON ( 0x40000005 )
DPCC2 ON ( 0x40000005 )
BLS ON ( 0x40000001 )
SDG OFF ( 0x80446197 )
```

```
LSC ON ( 0x1 )
AWBGAIN ON ( 0x80446197 ) ( gain : 0x010d010d , 0x01f20218 )
DEBAYER ON ( 0xf000111 )
CCM ON ( 0xc0000001 )
GAMMA_OUT ON ( 0xc0000001 )
CPROC ON ( 0xf )
IE OFF ( 0x0 ) ( effect : BLACKWHITE )
WDR OFF ( 0x30cf0 )
HDRTMO ON ( 0xc7705a23 )
HDRMGE ON ( 0xc0000005 )
RAWNR ON ( 0xc0100001 )
GIC OFF ( 0x0 )
DHAZ ON ( 0xc0101019 )
3 DLUT OFF ( 0x2 )
GAIN ON ( 0xc0010111 )
LDCH OFF ( 0x0 )
CSM FULL ( 0x80446197 )
SIAF OFF ( 0x0 )
SIAWB OFF ( 0x0 )
YUVAE ON ( 0x400100f3 )
SIHST ON ( 0x38000107 )
RAWAF ON ( 0x7 )
RAWAWB ON ( 0x4037e887 )
RAWAE0 ON ( 0x40000003 )
RAWAE1 ON ( 0x400000f5 )
```

Page 92

```
RAWAE2 ON ( 0x400000f5 )
RAWAE3 ON ( 0x400000f5 )
RAWHIST0 ON ( 0x40000501 )
RAWHIST1 ON ( 0x60000501 )
RAWHIST2 ON ( 0x60000501 )
RAWHIST3 ON ( 0x60000501 )
```

Input: input source, input format, resolution, DDR readback times, current frame number, actual frame interval

Output: output object, output format, resolution, current frame number, actual frame interval

Interrupt: Including mipi interrupts, interrupts of modules in the isp, the data is incremented, indicating that there is data entering the isp, ErrCnt error interrupt statistics information

clk_isp: isp clock frequency

Other: Switch status of each module of isp

```
[ root@RV1126_RV1109 : / ] # cat / proc / rkispp0
rkispp0    Version : v00 . 01.07
Input      rkisp0 Format : FBC420 Size : 3840 x2160 ( frame : 1656 rate : 51 ms delay : 85 ms )
The Output     rkispp_scale0 the Format : in NV12 Size : 1920 x1080 ( Frame : 1655 Rate : 51 is MS
delay : 108 ms )
TNR ON ( 0xd00000d ) ( mode : 2 to1 ) ( global gain : disable ) ( frame : 1656
time : 13 ms ) CNT : 0x0 STATE : 0x1e000000
NR ON ( 0x47 ) ( external gain : enable ) ( frame : 1656 time : 9 ms ) 0x5f0 : 0x0
0x5f4 : 0x0
SHARP ON ( 0x1d ) ( YNR input filter : ON ) ( local ratio : OFF ) 0x630 : 0x0
FEC OFF ( 0x2 ) ( frame : 0 time : 0 ms ) 0xc90 : 0x0
ORB OFF ( 0x0 )
Interrupt  Cnt : 5300 ErrCnt : 0
clk_ispp 500000000
aclk_ispp 500000000
hclk_ispp 250000000
```

Input: input source, input format, resolution, current frame number, actual frame interval

Output: output object, output format, resolution, current frame number, actual frame interval

Interrupt: The processing in ispp is interrupted, and the data increment indicates that there is data entering ispp, ErrCnt error interrupt statistics information

clk_ispp: ispp clock frequency

Other: Switch status of each module of ispp

Appendix A CIS driver V4L2-controls list

CID	describe
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.

Appendix B MEDIA_BUS_FMT table

CIS sensor type	Sensor output 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 raw bw sensor	MEDIA_BUS_FMT_Y8_1X8 MEDIA_BUS_FMT_Y10_1X10 MEDIA_BUS_FMT_Y12_1X12

Appendix C CIS Reference Driver List

Page 95

CIS data interface	CIS output data type	Frame/Field	Reference drive
			0.3M ov7750.c gc0403.c 1.2M ov9750.c jx-h65.c 2M ov2685.c ov2680.c ov2735.c

			gc2385.c gc2355.c gc2053.c sc2239.c sc210iot.c
MIPI	Bayer RAW	frame	4M gc4c33.c
			5M ov5695.c ov5648.c ov5670.c gc5024.c gc5025.c gc5035.c
			8M ov8858.c imx378.c imx317.c imx219.c gc8034.c
			13M ov13850.c imx258.c

Page 96

CIS data interface	CIS output data type	Frame/Field	Reference drive
			2M imx307.c imx327.c gc2093.c ov02k10 ov2718.c sc200ai.c sc2310.c jx-f37.c
MIPI	Bayer raw hdr	frame	4M ov4689.c os04a10.c imx347.c sc4238.c
			5M imx335.c
			8M imx334.c imx415.c
MIPI	YUV	frame	2M gc2145.c

			0.3M ov7251.c
MIPI	RAW BW	frame	1M ov9281.c
			1.3M sc132gs.c
MIPI	YUV	field	tc35874x.c
ITU.BT601	Bayer RAW		2M imx323.c ar0230.c

Page 97

CIS data interface	CIS output data type	Frame/Field	Reference drive
			0.3M gc0329.c gc0312.c gc032a.c
ITU.BT601	YUV		2M gc2145.c gc2155.c gc2035.c bf3925.c
ITU.BT601	RAW BW		
ITU.BT656	Bayer RAW		2M imx323 (supportable)

Appendix D VCM driver ic reference driver list

Reference drive
vm149c.c
dw9714.c
fp5510.c

Appendix E Flash light driver ic reference driver list

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