Rockchip Linux 4.4 Camera Developer Guide

ID: RK-KF-YF-347

Release Version: V2.0.0

Release Date: 2020-03-18

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

Rockchip Electronics Co., Ltd.

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

Website: www.rock-chips.com

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

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document mainly introduces the new driver structure of CIF and ISP of Rockchip chipset, and how to write/port sensor driver, application layer debugging method, application development interface, 3A integration, etc.

The CIF, ISP and camera drivers described in this document all meet V4L2 standard interface and also provide compatible interface, providing compatible interface. At the same time, try to simplify the difficulty of writing and porting the sensor driver. However, users still need to understand the usage of a series of V4L2 tools and related concepts.

Product Version

Chipset	Kernel Version	ISP	CIF
RK3399	4.4	Yes, two	No
RK3326/PX30	4.4	Yes, one	Yes, one
RK3288	4.4	Yes, one	Yes, one
RK312x/PX3SE	4.4	No	Yes, one
RK180x	4.4	Yes, one	Yes, one

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	ZhengSQ	2018-07-10	Initial version
V2.0.0	ZhengSQ	2020-03-18	Add application interface, update 3A application method, adjust the order of chapters, revise errors

Contents

1. Historical Version Introduction and Acronyms

1.1 Acronyms

- 3A, refers to auto focus (AF), auto exposure (AE) and auto white balance (AWB) algorithms, or algorithms.so library.
- Async Sub Device, refers to V4L2 sub-device under Media Controller structure,such as sensor,MIPI DPHY.
- Bayer Raw, or Raw Bayer, represents the data frame format of RGGB, BGGR, GBRG, GRBG output from device (sensor or ISP).
- Buildroot, refers to a series of Linux SDKs base on Buildroot released by Rockchip.
- Camera, refers to a complete system composed of VIP or ISP in Rockchip chipset and its connected sensor, and their drivers in this document.
- CIF, VIP module of RK chips which is used to receive data from sensor and save it to memory. Only used to transfer data, no ISP function.
- DVP, is a parallel data transfer interface, is the abbreviation of Digital Video Port.
- Entity, refers to each node under Media Controller framework.
- FCC or FourCC, that is Four Character (FCC) codes, and it refers to the image format represented by 4 characters in Linux Kernel. For details, see FourCC chapter.
- HSYNC, refers to the line synchronization signal of the DVP interface.
- ISP, Image Signal Processing, used to receive and process images. It refers to both hardware and ISP driver in general.
- IOMMU, Input-Output Memory Management Unit, in this document, it means iohmu module in Rockchip chips, it is used to map physically fragmented memory pages into visible contiguous memory of cif or isp. It refers to both hardware and IOMMU driver in general.
- IQ, Image Quality, in this document, it refers to the IQ xml debugged by Bayer Raw Camera. It is used for 3A tunning.
- Media Controller, a media framework for Linux kernel, mainly used for topology management.
- MIPI, refers to MIPI protocol in general in this document.
- MIPI-DPHY refers to MIPI-DPHY protocol, or the controller that complies with MIPI-DPHY protocol of Rockchip chips in this document.
- MP, Main Path, refers to an output node of Rockchip ISP, which can output full-resolution images and generally used for taking pictures and capturing raw images.
- PCLK is Pixel clock output from sensor.
- Pipeline refers to the link formed by the interconnection of each entity in media controller in this document.
- RKCIF, refers to the driver name of CIF.
- RKISP1, refers to the name of ISP driver.
- SP, Self Path, refers to an output node of Rockchip ISP, which can only output 1080p resolution.
- Userspace, that is Linux user space (relative to Linux kernel space).
- V4L2, Video4Linux2, the video processing module of Linux kernel.
- VIP refers to Video Input Processor in Rockchip chips, it once used as an alias for CIF.
- VSYNC, Field sync signal of DVP interface.

1.2 Historical Versions of ISP and CIF Drivers

The RKISP1 and RKCIF drivers described in this document are based on Media Controller, V4L2 Framework, VB2, and sensor is registered as an Async Sub Device asynchronously. Their codes are located in the `drivers/media/platform/rockchip/isp1/` and `drivers/media/platform/rockchip/cif/` directories, respectively; sensor code is located in the `drivers/media/i2c` directory.

The previous versions are no longer updated continuously or no longer supported. The details are as follows:

Drive name	Type	Kernel	Whether in the scope of this article	Code location
RKISP1	ISP	4.4	Yes	drivers/media/platform/rockchip/isp1
RKCIF	CIF	4.4	Yes	drivers/media/platform/rockchip/cif
RK-ISP10	ISP	4.4	No	drivers/media/platform/rk-isp10/
RK-CAMSYS	CIF	4.4	No	drivers/media/video

1.3 FAQ Documents

The FAQ document named "Rockchip_Trouble_Shooting_Linux4.4_Camera_CN" which is generally located in the same directory as this document is recommended for customers debugging sensor faster and more conveniently.

2. Sensor Driver Development and Porting

Sensor drivers are located in the drivers/media/i2c directory. Note that the sensor drivers described in this chapter are with Media Controller attribute, so the drivers in the drivers/media/i2c/soc_camera directory are not applicable.
Sensor driver and RKCIF or RKISP1 driver are independent to the greatest extent, the two are registered asynchronously, and the connection relationship is declared by the remote-endpoint in dts. Therefore, the sensor driver described in this chapter is applicable to both RKCIF and RKISP1.
In the Media Controller structure, sensor is generally used as a Sub Device and linked with the Rkcif, Rkisp1 or Mipi Dphy driver through Pad. This chapter mainly introduces the sensor driver code, dts configuration, and how to debug when porting a sensor driver.

This chapter summarizes the development and porting of sensor driver into 5 parts,

- Write the power-on sequence according to the datasheet, including vdd, reset, powerdown, clk, etc.
- Configure sensor register to output the required resolution and format.
- Write the callback functions needed by struct v4l2_subdev_ops, generally including set_fmt, get_fmt, s_stream, s_power.
- Added v4l2 controller for setting such as fps, exposure, gain, test pattern.
- Write the probe() function and add Media Control and Sub Device initialization code.

As a good habit, after completing the driver coding, you also need to add the corresponding documentation. You can refer to Documentation/devicetree/bindings/media/i2c/. In this way, the board-level dts can be quickly configured according to this document.

Generally, in the board-level dts, the followings are necessary for sensor driver reference:

- Configure the correct clk and io mux.
- Set the regulator and gpio required for the power-on sequence according to the schematic.
- Add the port sub-node to establish connection with cif or isp.

This chapter takes ov5695 and ov2685 as examples to analyze the sensor driver.

2.1 Power-on Sequence

Different sensors have different power-on sequence requirements. For example, most OV sensors may not have strict timing requirements. As long as mclk, vdd, reset, and powerdown states are correct, I2C communicate can be performed correctly and images can be output without worrying about the sequence and delay of electricity. However, there are still a few number of sensors that have very strict power-on requirements. For example, OV2685 must be powered on strictly follow the sequence.

In the datasheet provided by the sensor manufacturer, there is usually a power-on sequence diagram, which only needs to be configured in order. Take drivers/media/i2c/ov5695.c as an example, where `__ov5695_power_on()` is used to power on sensor. The code is as follows (with deletions).

```
static int __ov5695_power_on(struct ov5695 *ov5695)
{
    int ret;
    u32 delay_us;
    struct device *dev = &ov5695->client->dev;

    ret = clk_set_rate(ov5695->xvclk, OV5695_XVCLK_FREQ);

    if (clk_get_rate(ov5695->xvclk) != OV5695_XVCLK_FREQ)
        dev_warn(dev, "xvclk mismatched, modes are based on 24MHz\n");
    ret = clk_prepare_enable(ov5695->xvclk);

    if (!IS_ERR(ov5695->reset_gpio))
        gpiod_set_value_cansleep(ov5695->reset_gpio, 1);

    ret = regulator_bulk_enable(OV5695_NUM_SUPPLIES, ov5695->supplies);

    if (!IS_ERR(ov5695->reset_gpio))
        gpiod_set_value_cansleep(ov5695->reset_gpio, 0);

    if (!IS_ERR(ov5695->pwdn_gpio))
        gpiod_set_value_cansleep(ov5695->pwdn_gpio, 1);

    /* 8192 cycles prior to first SCCB transaction */
    delay_us = ov5695_cal_delay(8192);
    usleep_range(delay_us, delay_us * 2);

    return 0;
}
```

The power-on sequence of OV5695 is briefly described as follows,

- Provide xvclk (that is mclk) first.
- Next, enable reset pin.
- Power on vdd of each channel. The `regulator_bulk` is used here, because vdd, vodd, and avdd have no strict order. If there are strict requirements between vdd, you have to be handled separately, please refer to OV2685 driver code.
- Set the reset and powerdown pin of sensor to work status. Reset or powerdown, may be one of them is needed . Configure according to the actual needs of the sensor package and hardware schematic.
- Finally, according to the timing requirements of ov5695, 8192 clk cycles are required to be delayed before power-on is completed.

Note that although many sensors can work normally without powering on as required by datasheet, it is undoubtedly that the most reliable to operate according to the sequence recommended by vendors.

Similarly, there will be a power down sequence in the datasheet, which also needs to be implemented as required.

2.1.1 Check Whether the Power-on Sequence is Correct

In the .probe() stage, it will try to read the chip id, such as `ov5695_check_sensor_id()` of ov5695. If the chip id can be read correctly, it is generally considered that the power-on sequence is correct and sensor can perform i2c communication normally.

2.2 Sensor Initialization Register List

Both of OV5695 and OV2685 define `struct ov5695_mode` and `struct ov2685_mode`, which are used to represent different initialization modes of sensors, that is, sensor can output images of different resolutions, different fps, etc. Mode can include such as resolution, Mbus Code, fps, register initialization list, etc.
Register initialization list, please fill in as it provided by the vendor directly. It should be noted that the list ends with `REG_NULL` at the end. **Note that REG_NULL should not conflict with the register address**.

2.3 The v4l2_subdev_ops Callback Function

The v4l2_subdev_ops callback function is the core of the logic control in the sensor driver. The callback function includes rich interfaces, please check the kernel code `include/media/v4l2-subdev.h` for details. It is recommended that the sensor driver include the following callback functions at least.

- .open(), it will call the .open() function when opening the /dev/v4l-subdev? node through Userspace. When the application layer needs to set control of the sensor separately, .open() must be implemented.
- .s_power(), including power on and power off.
- .s_stream(), that is setting stream. Including stream on and stream off. Generally configure the register here to make it output images.
- .enum_mbus_code(), enumerate the mbus_code supported by driver.
- .enum_frame_size(), enumerate the resolutions supported by driver.
- .get_fmt(), returns the format/size selected by the current sensor. If .get_fmt() is missing, you cannot find the currently configured format of sensor entity by media-ctl tool.
- .set_fmt(), set the format/size of sensor.

In the above callbacks, .s_power() and .s_stream() will be more complicated. In the ov5695 driver code, pm_runtime is used to manage power. Use `v4l2_ctrl_handler_setup()` in .s_stream() to configure control information actually (v4l2 control may be updated when the sensor is powered off) and write to the register.

2.4 V4l2 Controller

It is necessary to implement the v4l2 controller function to update exposure, gain, and blanking dynamically, especially for RAW BYAYER sensor. General Raw Bayer Sensors are required.

In the OV5695 driver code,

- ov5695_initialize_controls(), used to declare which controls are supported and set the maximum and minimum value.
- struct v4l2_ctrl_ops, specifies the `ov5695_set_ctrl()` callback function, application can set value to each control through this function.

2.5 Probe Function and Registration Media Entity and v4l2 subdev

In the probe function, firstly, parse dts to obtain information such as regulator, gpio, clk and so on to power on and off the sensor. Next, register media entity, v4l2 subdev, and v4l2 controller information. Note that the registration of v4l2 subdev is asynchronous. Several key functions are as follows.

- v4l2_i2c_subdev_init(), registered as a v4l2 subdev, callback function is provided in the parameter.
- ov5695_initialize_controls(), initialize v4l2 controls.
- media_entity_init(), registered as a media entity, OV5695 has only one output called Source Pad.
- v4l2_async_register_subdev(), declares that sensor needs to be registered asynchronously. This call is necessary, because both of RKISP1 and RKCIF register Sub Device asynchronously.

2.6 dts Example: MIPI Sensor

According to the hardware design, the key is to configure pinctl (iomux), clk, gpio, remote port.
The following example is the OV5695 dts node in rk3326-evb-lp3-v10-linux.dts.

```
    ov5695: ov5695@36 {
        compatible = "ovti,ov5695";
        reg = <0x36>;

        clocks = <&cru SCLK_CIF_OUT>;
        clock-names = "xvclk";

        avdd-supply = <&vcc2v8_dvp>;
        dovdd-supply = <&vcc1v8_dvp>;
        dvdd-supply = <&vdd1v5_dvp>;

        /*reset-gpios = <&gpio2 14 GPIO_ACTIVE_HIGH>;*/
        pwdn-gpios = <&gpio2 14 GPIO_ACTIVE_HIGH>;

        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "TongJu";
        rockchip,camera-module-lens-name = "CHT842-MD";
```

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

Note:

- pinctrl, initialize the necessary pin iomux, this example includes reset pin initialization and clk iomux.
- clock, specify the name as xvclk (driver will read clock named xvclk), that is, 24M clock.
- vdd supply, OV5695 need three power supplies.
- The port subnode defines an endpoint that declares a connection to mipi_in_wcam. Similarly, mipi dphy will reference wcam_out.
- data-lanes specifies that OV5695 uses two lanes. **In the wcam_out node, data-lanes should match it**.

2.7 dts Example: DVP Sensor

Compared with Mipi sensor, it has no configure data-lanes of DVP sensor's dts , the endpoint is connected to cif, and rest parts are the same.

Take gc2155 in `arch/arm64/boot/dts/rockchip/rk3326-evb-lp3-v10-linux.dts` as an example,

- In the dts node of gc2155, the remote-endpoint points to cif_in.
- There's no need to configure data-lanes parameter.

2.8 Sensor Debugging

After completing sensor driver porting, you have to check whether it works normally.

If you encounter problems during the debugging process, please troubleshoot according to the FAQ document first.

2.8.1 Whether the Sensor is Registered Successfully

The first key point for sensor debugging is whether i2c can communicate successfully and check the chip id is correct or not. If it is correct, indicating that there is no problem with power-on sequence. The related log will usually be printed out in driver. Logs of different sensors are not the same, it will not take examples here. Second, check `media-ctl` topology to see if sensor is already registered as an entity. If so, sensor has been registered successfully.

2.8.2 Is there Any Output from Capture Image Tools

Please capture images through capture tools such as `v4l2-ctl`, `gstreamer`, camera app, etc.

2.8.3 Check Whether the Control Is Effective

Use v4l2-ctl to set relevant parameters, such as gain, exposure, blanking, and generate pictures to check whether controls of the sensor are effective. For example, increase gain or exposure whether the brightness of the picture increases; increase blanking whether the frame rate decreases.

3. Debugging Tools and Commonly Used Commands

This chapter mainly introduces commonly used capture tools.
Because most of the commands are relatively long, in order to read more friendly, the escape character \ is used to split a line of commands into several lines. Users can directly copy and paste when using, but if users put commands on one line, please remove the escape character \ .

3.1 v4l-utils

In Linux SDKs released by Rockchip, the v4l-utils package has been integrated by default. Users can turn on or off the v4l-utils package through the building switch of buildroot. such as:

```
# grep -rn LIBV4L_UTILS - buildroot/configs/rockchip/camera.config
BR2_PACKAGE_LIBV4L_UTILS=y
```

Users can also get the source code for building on the official website of www.linuxtv.org.

The v4l-utils package can be installed directly through the apt tool under Ubuntu system, as follows:

```
# sudo apt-get install v4l-utils
```

3.2 Use media-ctl to Get Topology

The media-ctl is a tool in the v4l-utils package. It is mainly used to check and configure the information of each entity of Media Framework, such as format, crop, link enable, etc. This tool can expand camera functions more flexibly.

For regular application cases, users do not need to configure specific Entity information, just use the default.

3.2.1 Display Topology

The following command is used to display topology. Note: **When both cif and isp are enabled, or there are multiple isps enabled, or USB camera is inserted, there may be multiple media devices, such as /dev/media0, /dev/media1, /dev/media2.**

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

The main concern of developers is whether to find the Entity of sensor. **If the Entity of sensor is not found, it means that there is a problem with sensor registration. Please check by FAQ document.**

For example, after connecting an ov5695 camera to RK3326 SDK board, you can see the following output (with deletion).

```
# media-ctl -p -d /dev/media1
- entity 9: m00_b_ov5695 2-0036 (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:SBGGR10_1X10/2592x1944@10000/300000 field:none]
                -> "rockchip-mipi-dphy-rx":0 [ENABLED,DYNAMIC]
```

You can see from the ov5695 entity information:

- The full name of the Entity is: `m00_b_ov5695 2-0036`.
- It is a `V4L2 subdev` (Sub-Device) `Sensor`.
- Its corresponding node is `/dev/v4l-subdev2`, applications (such as v4l2-ctl) can open it and configure.
- It has only one output (`Source`) node, that is `pad0`.
- Its output format is `[fmt:SBGGR10_1X10/2592x1944@10000/300000 field:none]`, where SBGGR10_1X10 is the abbreviation of one mbus-code, the next section will list common mbus-code.
- Its Source pad0 is linked to (`->`) `pad0` of `"rockchip-mipi-dphy-rx"`, and its current status is `ENABLED`. `DYNAMIC` means that the status can be changed to DISABLED.

If the same ISP or CIF is connected to two sensors at the same time, only one of them is ENABLED, as shown in the following example (with deletion).

```
[root@rk3326_64:/]# media-ctl -p -d /dev/media1
-entity 9: irs16x5c 1-003d (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:SBGGR12_1X12/224x1557@10000/300000 field:none]
                -> "rockchip-mipi-dphy-rx":0 [ENABLED,DYNAMIC]

-entity 10: irs16x5c 2-003d (1 pad, 1 link)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev3
        pad0: Source
                [fmt:SBGGR12_1X12/224x1557@10000/300000 field:none]
                -> "rockchip-mipi-dphy-rx":0 [DYNAMIC]
```

In the above example:

- The two sensors irs16x5c both received "rockchip-mipi-dphy-rx":0, but only entity 9 is ENABLED.
- You have to switch sensor only when the entire link stops working, that is: you cannot change the configuration of each Entity in the pipeline during capturing process.

3.2.2 Switch Sensor

If multiple sensors are connected, you can switch them with the following commands.

```
# media-ctl -d /dev/media0 \
    -l '"ov5695 7-0036":0->"rockchip-sy-mipi-dphy":0[0]'
# media-ctl -d /dev/media0 \
    -l '"ov2685 7-003c":0->"rockchip-sy-mipi-dphy":0[1]'
```

- The command format is media-ctl -l'"entity name":pad->"entity name":pad[Status]'.
- The entire link needs to use single quotes because of special characters: > [ ].
- Entity name needs to use double quotes because there are spaces in between.
- Status can be 0 or 1 to indicate Active or In-Active.

3.2.3 Modify the Format and Sze of Entity

First example: OV5695 supports multiple resolution output, it is 2592x1944 by default. Now change the output resolution to 1920x1080.

```
# media-ctl -d /dev/media1 \
    --set-v4l2 '"m00_b_ov5695 2-0036":0[fmt:SBGGR10_1X10/1920x1080]'
```

After modifying the output of OV5695, the size of rkisp1-isp-subdev and the video device crop should be modified accordingly. Because the size of the subsequent stage cannot be greater than the size of the previous stage.

```
# media-ctl -d /dev/media1 \
    --set-v4l2 '"rkisp1-isp-subdev":0[fmt:SBGGR10_1X10/1920x1080]'
# media-ctl -d /dev/media1 \
    --set-v4l2 '"rkisp1-isp-subdev":0[crop:(0,0)/1920x1080]'
# media-ctl -d /dev/media1 \
    --set-v4l2 '"rkisp1-isp-subdev":2[crop:(0,0)/1920x1080]'
# v4l2-ctl -d /dev/video1 \
    --set-selection=target=crop,top=0,left=0,width=1920,height=1080
```

Second example: for raw bayer sensors, rkisp1 outputs yuv format by default, and the fmt of rkisp1-isp-subdev is changed to the fmt of sensor, so that MP node can output raw images.

```
# media-ctl -d /dev/media1 \
    --set-v4l2 '"rkisp1-isp-subdev":2[fmt:SBGGR10/2592x1944]'
```

In the above example, there are some points to be noted:

- Pay attention to special characters, you need to use single or double quotes.
- Don't add or delete spaces in quotation marks.
- Please use `media-ctl --help` to view more detailed help.

3.2.4 Commonly Used mbus-code Formats

Mbus-code is short for Media Bus Pixel Codes, which describes the format used for transmission on the physical bus, such as the image format transmitted by sensor to isp through mipi dphy, or the format transmitted between sub-modules within ISP. In particular, it is necessary to distinguish Mbus-code from FourCC. The latter refers to the image format stored in Memory.

Mbus-code is defined in `include/uapi/linux/media-bus-format.h` of kernel.

The following table lists several Mbus-codes commonly used in this document.

Macros defined in Kernel	Mubs-code abbreviation	Type	Bpp	Bus width	Samples
MEDIA_BUS_FMT_SBGGR8_1X8	SBGGR8_1X8	Bayer Raw	8	8	1
MEDIA_BUS_FMT_SRGGB8_1X8	SRGGB8_1X8	Bayer Raw	8	8	1
MEDIA_BUS_FMT_SBGGR10_1X10	SBGGR10_1X10	Bayer Raw	10	10	1
MEDIA_BUS_FMT_SRGGB10_1X10	SRGGB10_1X10	Bayer Raw	10	10	1
MEDIA_BUS_FMT_SBGGR12_1X12	SBGGR12_1X12	Bayer Raw	12	12	1
MEDIA_BUS_FMT_SRGGB12_1X12	SRGGB12_1X12	Bayer Raw	12	12	1
MEDIA_BUS_FMT_YUYV8_2X8	YVYU8_2X8	YUV 422	16	8	2
MEDIA_BUS_FMT_UYUV8_2X8	UYUV8_2X8	YUV 422	16	8	2
MEDIA_BUS_FMT_Y8_1X8	Y8_1X8	YUV GREY	8	8	1
MEDIA_BUS_FMT_RGB888_1X24	RGB888_1X24	RGB 888	24	24	1

The media-ctl can list the supported mbus codes.

```
# media-ctl --known-mbus-fmts
```

3.2.5 Find Video Devices

There are multiple entities in the topology, some are sub devices and some are video devices. The device node corresponding to the former is /dev/v4l-subdev, and the latter corresponds to /dev/video. In the case of multiple video devices, users are most concerned about which device can output images.

```
# media-ctl -d /dev/media1 -e "rkisp1_selfpath"
/dev/video2
# media-ctl -d /dev/media1 -e "rkisp1_mainpath"
/dev/video1
```

The above two commands display the device path of SP and MP nodes of RKISP1 in the link /dev/media1 respectively. RKISP1 has two video output devices, both of which can output images.

It is similar to RKCIF:

```
# media-ctl -d /dev/media0 -e "stream_cif"
/dev/video0
```

The above command shows the path of RKCIF video device in the link /dev/media0. RKCIF has only one video output node.

```
# v4l2-ctl -d /dev/video1 --all
```

The above command shows some main parameters of /dev/video1, such as crop, fmt, v4l2 controls, etc.

3.3 Capture Pictures by v4l2-ctl

The operation of the Media-ctl tool is through media devices such as /dev/medio0, and it manages format, size, and link of each node in the Media topology. The V4l2-ctl tool is for video devices such as /dev/video0 and /dev/video1. It performs a series of operations such as set_fmt, reqbuf, qbuf, dqbuf, stream_on, stream_off, etc. on video devices. This document mainly uses v4l2-ctl to collect frame data, set exposure, gain, VTS, etc. v4l2_control.

It is recommended to read the help file of v4l2-ctl first. The help document has rich contents and is divided into many parts. What we should pay more attention to are streaming and vidcap.

The summary of the help file is as follows:

```
# v4l2-ctl --help
```

To check the complete help document as follows, with rich content relatively.

```
# v4l2-ctl --help-all
```

To check parameters related to streaming as follows.

```
# v4l2-ctl --help-streaming
```

To check parameters related to vidcap as follows. It mainly includes get-fmt, set-fmt, etc.

```
# v4l2-ctl --help-vidcap
```

3.3.1 Capture Frames by v4l2-ctl

Example 1: capture one frame of NV12 data output by RKCIF and save it to /tmp/nv12.bin with a resolution of 640x480. Before saving data, discard the first 3 frames (that is, although the first 3 frames are returned to userspace, they are not saved to the file).

```
# v4l2-ctl -d /dev/video0 \
    --set-fmt-video=width=640,height=480,pixelformat=NV12 \
    --stream-mmap=3 \
    --stream-skip=3 \
    --stream-to=/tmp/nv12.bin \
    --stream-count=1 \
    --stream-poll
```

Example 2: capture 10 frames of NV12 data output by RKISP and save it to /tmp/nv12.bin with a resolution of 1920x1080.

```
# v4l2-ctl -d /dev/video1 \
    --set-selection=target=crop,top=0,left=0,width=1920,height=1080
    --set-fmt-video=width=1920,height=1080,pixelformat=NV12 \
    --stream-mmap=3 \
    --stream-to=/tmp/nv12.bin \
    --stream-count=10 \
    --stream-poll
```

Note of parameters:

- -d, specify the operation object as /dev/video0 device.
- --set-selection, specify to crop input images. Especially when the size of the previous stage of RKISP1 changes, make sure that the selection is not greater than the output size of the previous stage. The crop of RKCIF is set by the --set-crop parameter.
- --set-fmt-video, specifies the width and height and pxielformat (represented by FourCC). NV12 is the pixelformat represented by FourCC.
- --stream-mmap, specify the type of buffer as mmap, that is, a physically continuous or iommu mapped buffer allocated by kernel.
- --stream-skip, specify to discard (not save to file) the first 3 frames.
- --stream-to, specify the file path for saving frame data.
- --stream-count, specifies the number of frames captured, excluding the number of discarded by --stream-skip.
- --stream-poll, this option indicates v4l2-ctl use asynchronous IO, that is, use "select" to wait for frame data complete before dqbuf to ensure that dqbuf is not blocked. Otherwise, dqbuf will block until a data frame arrives.

3.3.2 Set Controls like Exposure and Gain

If the sensor driver implements v4l2 control, v4l2-ctl can be used to set the exposure, gain, etc. before capturing images.
RKCIF or RKISP will inherit the control of sub device, so you can see the v4l2 control of sensor through /dev/video3.

The following are the relevant settings of OV5695 on RK3326 SDK device, including exposure, gain, blanking, test_pattern, etc.

```
# v4l2-ctl -d /dev/video1 -l

User Controls

          exposure 0x00980911 (int) : min=4 max=2228 step=1 default=1104
value=1104
              gain 0x00980913 (int) : min=0 max=16383 step=1 default=1024
value=1024

Image Source Controls

 vertical_blanking 0x009e0901 (int) : min=1152 max=31687 step=1 default=1152
value=1152
     analogue_gain 0x009e0903 (int) : min=16 max=248 step=1 default=248 value=248

Image Processing Controls

      test_pattern 0x009f0903 (menu): min=0 max=4 default=0 value=0
```

These controls can be modified by v4l2-ctl. For example, modify exposure and analogue_gain as follows.

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

3.3.3 Capture Raw Images

The MP node of RKISP1 can capture raw images. At this time, ISP is in the by-pass state and does not tune data.

For example, capture Raw Bayer data output from sensor OV5695. The format is SBGGR10_1X10 and the size is 2592x1944.

```
# media-ctl -d /dev/media0 \
    --set-v4l2 '"ov5695 7-0036":0[fmt:SBGGR10_1X10/2592x1944]'
# media-ctl -d /dev/media0 \
    --set-v4l2 '"rkisp1-isp-subdev":0[fmt:SBGGR10_1X10/2592x1944]'
# media-ctl -d /dev/media0 \
    --set-v4l2 '"rkisp1-isp-subdev":0[crop:(0,0)/2592x1944]'
# media-ctl -d /dev/media0 \
    --set-v4l2 '"rkisp1-isp-subdev":2[fmt:SBGGR10_1X10/2592x1944]'
# media-ctl -d /dev/media0 \
    --set-v4l2 '"rkisp1-isp-subdev":2[crop:(0,0)/2592x1944]'
# v4l2-ctl -d /dev/video4 \
    --set-ctrl 'exposure=1216,analogue_gain=10' \
    --set-selection=target=crop,top=0,left=0,width=2592,height=1944 \
    --set-fmt-video=width=2592,height=1944,pixelformat=BG10 \
    --stream-mmap=3 \
    --stream-to=/tmp/mp.raw.out \
    --stream-count=1 \
    --stream-poll
```

Note:

- Media-ctl in line 4 sets isp-subdev output format to be consistent with sensor.
- Lines 3 and 5 set the size of crop and sensor to be the same, that is, no cropping.

- In line 6, if the picture is too dark, you can adjust expo or gain to increase brightness optionally. And sensor driver needs to implement v4l2 control.
- In line 7 and line 8, v4l2-ctl sets the selection without cropping and the output pixel format FourCC is BG10.
- It should be noted that although ISP does not process raw images, it still fills the low bits of 10bit data with 0 into 16bit. Regardless of the sensor input is 10bit or 12bit, the final upper layer is 16bit per pixel.

Add PGM symbol to the header of Bayer Raw file, and it will be converted into pgm images that can be directly opened and viewed in Ubuntu. Just add three lines of PGM header.

For example, convert raw images to pgm format:

```
# cat > /tmp/raw.pgm << EOF
P5
2592 1944
65535
EOF

# cat /tmp/mp.raw.out >> /tmp/raw.pgm
```

Note:

- Line 2, P5 is the fixed identifier.
- Line 3 indicates the resolution of raw images, that is, the length and width, separated by a space character.
- Line 4 indicates the depth, 65535 is 16bit. If it is 8bit, change it to 255 accordingly.
- Line 7, add the raw data to the end of the pgm file header.
- Note that there are only three lines in the pgm header, do not add extra blank lines.

3.3.4 Commonly used FourCC Formats

FourCC is short for Four Character Codes, which uses 4 characters (that is 32bit) to name the image format. In Linux Kernel, the macro definition is as follows:

```
#define v4l2_fourcc(a,b,c,d) \
    (((__u32)(a)<<0)|((__u32)(b)<<8)|((__u32)(c)<<16)|((__u32)(d)<<24))
```

The format defined by FourCC is the format in which images and videos are stored in the memory. This should be distinguished from mbus-code.

Several formats commonly used in this document are listed below. For more detailed definition, please refer to `videodev2.h` of the kernel code.

Macro defined in kernel	FourCC
V4L2_PIX_FMT_NV12	NV12
V4L2_PIX_FMT_NV21	NV21
V4L2_PIX_FMT_NV16	NV16
V4L2_PIX_FMT_NV61	NV61
V4L2_PIX_FMT_NV12M	NM12
V4L2_PIX_FMT_YUYV	YUYV
V4L2_PIX_FMT_YUV420	YU12
V4L2_PIX_FMT_SBGGR10	BG10
V4L2_PIX_FMT_SGBRG10	GB10
V4L2_PIX_FMT_SGRBG10	BA10
V4L2_PIX_FMT_SRGGB10	RG10
V4L2_PIX_FMT_GREY	GREY

3.4 Display YUV Images by mplayer in Ubuntu

Some commands in the previous section capture frame data and save them as a file, which can be parsed by mplayer in the Ubuntu environment.

The mplayer can be installed via apt as follows.

```
# sudo apt-get install mplayer
```

Play NV12 images with 640x480 size as follows.

```
# W=640; H=480; mplayer /tmp/nv12.bin -loop 0 -demuxer rawvideo -fps 30 \
    -rawvideo w=${W}:h=${H}:size=$((${W}*${H}*3/2)):format=NV12
```

Play YUYV images with 640x480 size, as follows.

```
# W=640; H=480; mplayer /tmp/yuyv.bin -loop 0 -demuxer rawvideo -fps 30 \
    -rawvideo w=${W}:h=${H}:size=$((${W}*${H}*2)):format=YUY2
```

In the above example:

- W and H are variables, specify the width and height to facilitate subsequent reference.
- fps specifies playback rate. If fps is 1, then one frame will be played in 1 second.
- size refers to the size of each frame.
- format refers to format, `mplayer -rawvideo format=help` can display all supported formats.

In Windows, tools such as 7yuv can be used to parse images.

3.5 GStreamer Usage

In Linux SDKs released by Rockchip, you can use GStreamer to preview Camera's images and encoding.

The v4l2src plugin can be used to get images from video device. By default, `rkisp_3A_server` will also start tunning, so that images with normal brightness and color can be obtained.

3.5.1 Display Images with GStreamer

The following command will display camera images on screen.

```
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0
# export XDG_RUNTIME_DIR=/tmp/.xdg
# gst-launch-1.0 v4l2src device=/dev/video1 ! \
    video/x-raw,format=NV12,width=2592,height=1944,framerate=30/1 ! kmssink
```

3.5.2 Videos and Images Encoding with GStreamer

Linux SDKs also have hardware encoding. The following commands can encode and save camera data stream into a file.

```
# gst-launch-1.0 v4l2src device=/dev/video1 num-buffers=100 ! \
    video/x-raw,format=NV12,width=1920,height=1088,framerate=30/1 ! \
    videoconvert ! mpph264enc ! h264parse ! mp4mux ! \
    filesink location=/tmp/h264.mp4
```

```
# gst-launch-1.0 -v v4l2src device=/dev/video1 num-buffers=10 ! \
    video/x-raw,format=NV12,width=1920,height=1080 ! mppjpegenc ! \
    multifilesink location=/tmp/test%05d.jpg
```

Note:

- mppjpegenc and mpph264enc encoder are hardware encoding provided by rockchipmpp plugin.
- The mpp encoder needs height with 16 alignment, so the kernel needs to include this patch: `fea937b015e7 media: rockchip: isp1/cif: set height alignment to 16 in queue_setup`.

3.6 Boards Debugging Without Screen

Rockchip Linux SDKs provide librkuvc.so, by this interface, boards can be used as an uvc camera and connect to a PC via a USB OTG cable.

In the chapter of application development, sample codes are given for reference.

3.7 Turn on Debugging Switch

Both RKISP1 or RKCIF drivers contain some v4l2_dbg() logs, which can be turned on by commands, as follows.

```
# echo 1 > /sys/module/video_rkcif/parameters/debug
```

Or

```
# echo 1 > /sys/module/video_rkisp1/parameters/debug
```

In addition, VB2 and V4L2 also have debugging switches.

Open VB2 related log as follows.

```
# echo 7 > /sys/module/videobuf2_core/parameters/debug
```

VB2 log mainly includes buffer rotation, such as reqbuf, qbuf, dqbuf and buffer status changes. It should be noted that the vb2 module switch is a general switch, and other related logs that use vb2(such as VPU/ISP, etc.) will also enable output.

Open v4l2 related log, such as ioctl call. The following command will open all v4l2 related logs.

```
# echo 0x1f > /sys/class/video4linux/video0/dev_debug
```

You can also open only a small part of log. The following Kernel macro defines which log will be enabled by each bit. Just turn on the bit corresponding to the required log. These macros are defined in the kernel header file `include/media/v4l2-ioctl.h`.

```
/* Just log the ioctl name + error code */
#define V4L2_DEV_DEBUG_IOCTL            0x01
/* Log the ioctl name arguments + error code */
#define V4L2_DEV_DEBUG_IOCTL_ARG        0x02
/* Log the file operations open, release, mmap and get_unmapped_area */
#define V4L2_DEV_DEBUG_FOP              0x04
/* Log the read and write file operations and the VIDIOC_(D)QBUF ioctls */
#define V4L2_DEV_DEBUG_STREAMING        0x08
/* Log poll() */
#define V4L2_DEV_DEBUG_POLL             0x10
```

4. 3A Integration Method

The contents of this section only applies to RKISP1. RKCIF is without ISP function and does not require 3A.

In order to support 3A, many ways have been added in the past version to support it, such as:

- Provide application program link of librkisp.so library.
- Write rkisp, rkv4l2src gstreamer plug-ins to support GStreamer.

However, the above ways are troublesome for applications and does not effectively adapt to existing programs, such as VLC and Chrome browsers.

After the RKISP1 v0.1.5 driver and camera_engine_rkisp v2.2.0 version, the `rkisp_3A_server` process has been added, which will trigger and complete 3A tunning automatically. In this way,

- The application no longer needs 3A library librkisp.so, but is only responsible for data flow.
- The v4l2src plug-in of GStreamer can be used directly, open source tools such as vlc can also be used directly, and normal 3A images can be obtained.

4.1 rkisp_3A_server

Including Kernel's RKISP1 driver, camera_engine_rkisp, and related startup scripts. If you have updated to the latest Rockchip Linux SDK, 3A is already integrated by default, including the following three main parts:

- The driver version of RKISP1 is v0.1.5 or later versions. The version number is defined in: `kernel/drivers/media/platform/rockchip/isp1/version.h`
- The camera_engine_rkisp package is updated to v2.2.0 or later. The path is in `external/camera_engine_rkisp`.
- The path of camera_engine_rkisp building script and self-starting script are in: `buildroot/package/rockchip/camera_engine_rkisp`

The file built by camera_engine_rkisp is as follows:

```
/usr/bin/rkisp_3A_server
/usr/lib/librkisp.so
/usr/lib/librkisp_api.so
/usr/lib/rkisp/
              ├── ae
              │   └── librkisp_aec.so
              ├── af
              │   └── librkisp_af.so
              └── awb
                  └── librkisp_awb.so
/etc/iqfiles/
/etc/init.d/S40rkisp_3A
```

Note:

- rkisp_3A_server, is an executable file, responsible for monitoring and starting 3A tunning as needed
- librkisp.so, which implements the main interface of 3A, and calls aec, af, awb link libraries respectively. The latter is not open source
- iqfiles directory, storing iq parameters and xml files of sensors
- S40rkisp_3A is the boot script of rkisp_3A_server so that it can be started with boot

Except different booting way, the other logic of Debian system provided by Rockchip is the same as Linux SDKs.

4.2 Enable rkisp_3A_server log

Enable log by declaring environment variables, for example, open AEC log as follows:

```
# export persist_camera_engine_log=0x40
```

Add the above line to /etc/init.d/S40rkisp_3A, the log will be saved to /var/log/messages. If there are many logs, the messages file will be split into multiple files, such as /var/log/messages.0 and /var/log/messages.1. **When users pack logs, remember to pack all logs. such as:**

```
# tar zcvf /tmp/camera-log.tar.gz /var/log/messages*
```

Logs can be opened according to modules as follows,

```
      bits:   31-28  27-24   23-20   19-16 15-12  11-8  7-4   3-0
      module: [u]    [u]    [xcore]  [ISP] [AF]   [AWB] [AEC] [NO]
             *[u] means unused now.
      each module log has following ascending levels:
             0: error
             1: warning
             2: info
             3: verbose
             4: debug
             5: low1
             6-7: unused, now the same as debug
```

Special attention: if users needs to start rkisp_3A_server in the terminal and open log, it is recommended to redirect logs to a file, because logs are too large, it cannot be directly printed to a slow device (such as a serial port, shell default output).

```
# /etc/init.d/S40rkisp_3A stop
# export persist_camera_engine_log=0x40
# /usr/bin/rkisp_3A_server --mmedia=/dev/media0 > /tmp/log 2>&1
```

As shown above, first stop service, set environment variables, and then manually execute the command to start rkisp_3A_server, and redirect logs to /tmp/log file.

4.3 XML Loading Acceleration Function

For some products that need to boot up and display images faster, we provide xml loading acceleration. By opening the macro definition

```
# make menuconfig

 | | --- Rockchip BSP packages                                          | |
 | | [*]  Rockchip Camera Engine for linux                              | |
 | | [*]   Rockchip Camera Engine 3A service run in booting             | |
 | |        Specify a directory to store xml speed up bin (disabled)  --->  | |
 | | ()    Rockchip Camera Engine IQ xml file                           | |
```

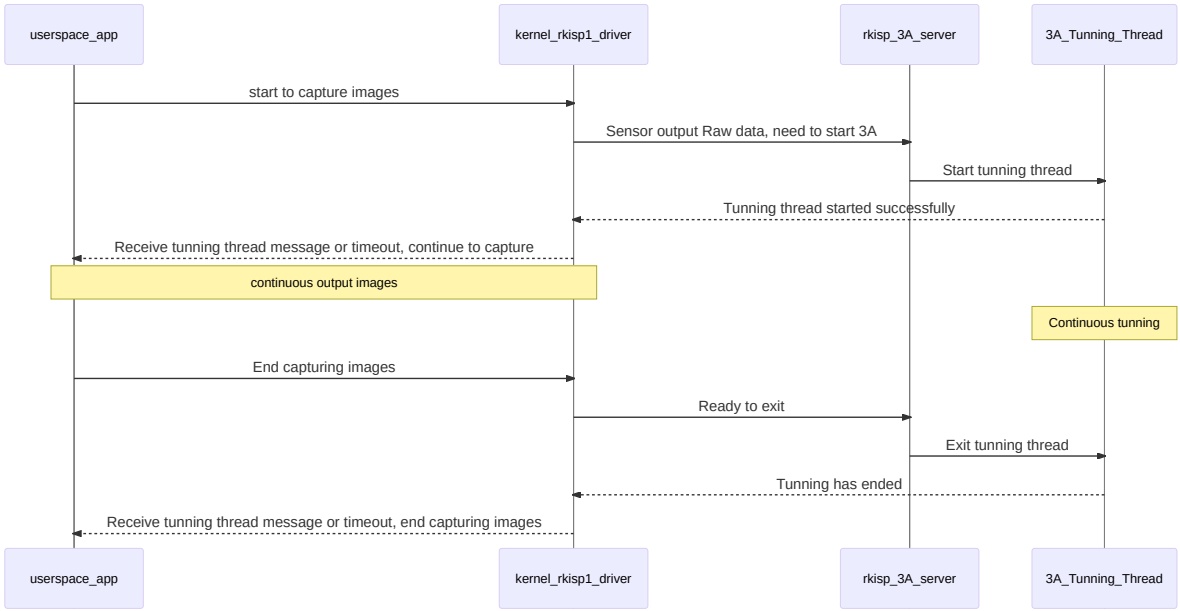
According to whether rootfs is writable, select the option `Specify a directory to store xml speed up bin`.

```
             | |                (X) disabled                              | |
             | |                ( ) /etc/iqfiles-db                       | |
             | |                ( ) /userdata/iqfiles-db                  | |
```

If rootfs is read only, you can only choose `/userdata/iqfiles-db`. This option specifies a directory for storing xml bin, and the file system where it is located needs to be writable.

4.4 Execution Sequence of rkisp_3A_server

The following shows the sequence diagram of rkisp_3A_server.



5. Applications Development

In addition to directly capturing and previewing images through v4l2-ctl, GStreamer, VLC, etc., users can also write programs based on the V4L2 interface to capture images and then process them.

Applications can also link librkisp.so library by itself to get richer image information, such as:

- Manually control exposure time and gain.
- Set exposure weight and spot metering for a certain area.
- Obtain brightness statistics, such as the average brightness of each area, the histogram of the entire picture.
- Control the maximum fps and gain to improve image quality.

All of the above functions can be obtained after applications are linked to librkisp.so. Therefore, the rkisp_3A_server process needs to be stopped first, and 3A initialization and start are completed by applications.

```
# /etc/init.d/S40rkisp_3A stop
```

Or delete S40rkisp_3A directly.

```
# rm /etc/init.d/S40rkisp_3A
```

5.1 librkisp_api.so Interface Usage

In order to use it faster, camera_engine_rkisp provides the rkisp_api interface, which is built into librkisp_api.so and can be called directly. It can also be used as a reference. Its main API declaration is in the rkisp_api.h header file.

The first example, open and get a 1920x1080 image.

```
int test_capture_mmap_quick()
{
    const struct rkisp_api_ctx *ctx;
    const struct rkisp_api_buf *buf;
    int count = 10;

    ctx = rkisp_open_device("/dev/video1", 0);
    if (ctx == NULL)
        return -1;

    rkisp_set_fmt(ctx, 1920, 1080, ctx->fcc);

    if (rkisp_start_capture(ctx))
        return -1;

    do {
        buf = rkisp_get_frame(ctx, 0);

        /* Deal with the buffer */

        rkisp_put_frame(ctx, buf);
    } while (count--);

    rkisp_stop_capture(ctx);
    rkisp_close_device(ctx);

    return 0;
}
```

In the above code, rkisp_3A_server still needs to be used for 3A tunning. If applications want to obtain more 3A related information, just modify the parameter when open:

```
    ctx = rkisp_open_device("/dev/video1", 1);
```

The obtained buf will contain more statistical information.

The second example, modify the resolution of sensor, for example, modify the default output resolution of ov5695 to 1920x1080.

```
    const struct rkisp_api_ctx *ctx;

    ctx = rkisp_open_device("/dev/video1", 0);
    if (ctx == NULL)
        return -1;
    rkisp_set_sensor_fmt(ctx, 1920, 1080, MEDIA_BUS_FMT_SBGGR10_1X10);
```

5.2 Share Memory by DMA Buffer

Sharing Buffer between multiple modules can reduce memory copy and improve efficiency. DMA Buffer is a detailed method. DMA Buffer can share memory between camera, RGA (image processing module), DRM (display), and MPP (coding).

By default, rkisp_api uses mmap to allocate memory from kernel. It can export memory to userspace and use it as a DMA Buffer for other modules. Conversely, it can also accept DMA Buffers from other modules and write directly to the target Buffer when Kernel collects data frames, so as to avoid once copy.

The first example, use MMAP to export DMA Buffer to other modules.

```
    const struct rkisp_api_ctx *ctx;
    const struct rkisp_api_buf *buf;

    ctx = rkisp_open_device("/dev/video1", 0);
    if (ctx == NULL || rkisp_start_capture(ctx))
        return -1;

    rkisp_get_frame(ctx, 0);
    printf("size: %d, dmabuf fd: %d\n", buf->size, buf->fd);
```

In the above example, `buf->fd` is the DMA Buffer descriptor, which can be directly used by other modules. But you have to pay attention to that:

- Before the buffer is used up, rkisp_put_frame() cannot be called.
- After using the buffer, don’t forget to call rkisp_put_frame().

The second example: using DMA Buffer of other modules, the camera data collected by kernel is directly filled into the target Buffer.

```
int test_capture_ext_dmabuf()
{
    ctx = rkisp_open_device("/dev/video1", 0);
    if (ctx == NULL)
        return -1;

    for (i = 0, ret = 0; i < buf_count; i++) {
        if (drmGetBuffer(dev.drm_fd, width, height, FORMAT, &buf[i]))
            goto out;
```

```
        dmabuf_fd[i] = buf[i].dmabuf_fd;
    }
    rkisp_set_fmt(ctx, width, height, ctx->fcc);
    rkisp_set_buf(ctx, buf_count, dmabuf_fd, buf[0].size);

    if (rkisp_start_capture(ctx))
        goto out;

    buf = rkisp_get_frame(ctx, 0);
    printf("The ext buf fd is: %d\n", buf->fd);
    rkisp_put_frame(ctx, buf);

    rkisp_stop_capture(ctx);

out:
    while (--i >= 0)
        drmPutBuffer(dev.drm_fd, &buf[i]);

    rkisp_close_device(ctx);
}
```

In the above example, `rkisp_set_buf()` directly give DRM DMA Buffer to camera for use, and Buffer is allocated through the drm interface. Here we mainly introduce the usage of rkisp_api interface. The functions such as `drmGetBuffer()` and `drmPutBuffer()` will not be listed in details.

5.3 Set Area Exposure Weight

In some special cases, the default global exposure cannot get the best effect. Users can change the key area of metering and increase the weight value of this area to get the best brightness of the area. Of course, other areas may be overexposed or dark at this time. For example, the following two cases:

- Face recognition under backlight. When there is no HDR, backlight makes the face area darker and is not beneficial recognition; at this time, after the face area is detected, the weight of the area can be increased to make the face clearer.
- The sweeper recognizes objects on the ground. At this time, the lower part of the image is concerned by the application, and due to the difference in object reflection, the brightness of the upper and lower parts is often different; at this time, the metering weight of the lower part of the image can be increased.

For example, call the setting weight interface:

```
    unsigned char weights[] = {
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
         1,  1,  5,  9, 15, 31, 15,  5,  1,
    };
    rkisp_set_expo_weights(ctx, weights, 81);
```

In the above example, a 9x9 array is defined, and the value range of each element is [1, 31] corresponding to 81 regions of the image. The larger the value, the greater the weight of the region.

5.4 Simulate to an UVC Camera by librkuvc.so

Rockchip Linux SDKs integrate librkuvc.so library, the source code of which is located in `external/uvc_app`. After obtaining camera images, it can be compressed and transmitted to PC via librkuvc.so.

For example, call librkuvc.so:

```
#include "uvc_control.h"
#include "uvc_video.h"

int test_uvc_mmap()
{
    const struct rkisp_api_ctx *ctx;
    const struct rkisp_api_buf *buf;
    uint32_t flags = 0;
    int extra_cnt = 0;

    ctx = rkisp_open_device("/dev/video1", 0);

    if (ctx == NULL)
        return -1;

    if (rkisp_set_fmt(ctx, 640, 480, V4L2_PIX_FMT_NV12))
        return -1;

    if (rkisp_start_capture(ctx))
        return -1;

    flags = UVC_CONTROL_LOOP_ONCE;
    uvc_control_run(flags);

    do {
        buf = rkisp_get_frame(ctx, 0);
        extra_cnt++;
        uvc_read_camera_buffer(buf->buf, buf->fd, buf->size, &extra_cnt,
sizeof(extra_cnt));

        rkisp_put_frame(ctx, buf);
    } while (1);

    uvc_control_join(flags);

    rkisp_stop_capture(ctx);
    rkisp_close_device(ctx);

    return 0;
}
```

In the above example, memory is allocated by camera driver and the DMA Buffer is given to UVC encoding.

```
# GCC=buildroot/output/rockchip_rk3326_64/host/bin/aarch64-buildroot-linux-gnu-gcc
# SYSROOT=buildroot/output/rockchip_rk3326_64/staging
# ENGINE=external/camera_engine_rkisp
# $GCC --sysroot=$SYSROOT camera_uvc.c \
        -L$ENGINE/build/lib/ -lrkisp -lrkisp_api \
        -L $SYSROOT/usr/lib/ -lrkuvc \
        -I $SYSROOT/usr/include/uvc/ \
        -I./ \
        -o camera_uvc
```

Build and copy camera_uvc to the /usr/bin directory, and call it on the development board as follows:

```
# /usr/bin/uvc_MJPEG.sh
# /usr/bin/camera_uvc
```

A few points to note:

- uvc_MJPEG.sh only needs to be initialized once after booting.
- Support resolutions of 640x480, 1280x720, 1920x1080, 2560x1440, please check /usr/bin/uvc_MJPEG.sh for updates.

Hardware requirements: mpp encoding requires the height of buffer to be aligned to 16, otherwise the memory access overrun will occur.

6. RKISP1 Driver Introduction

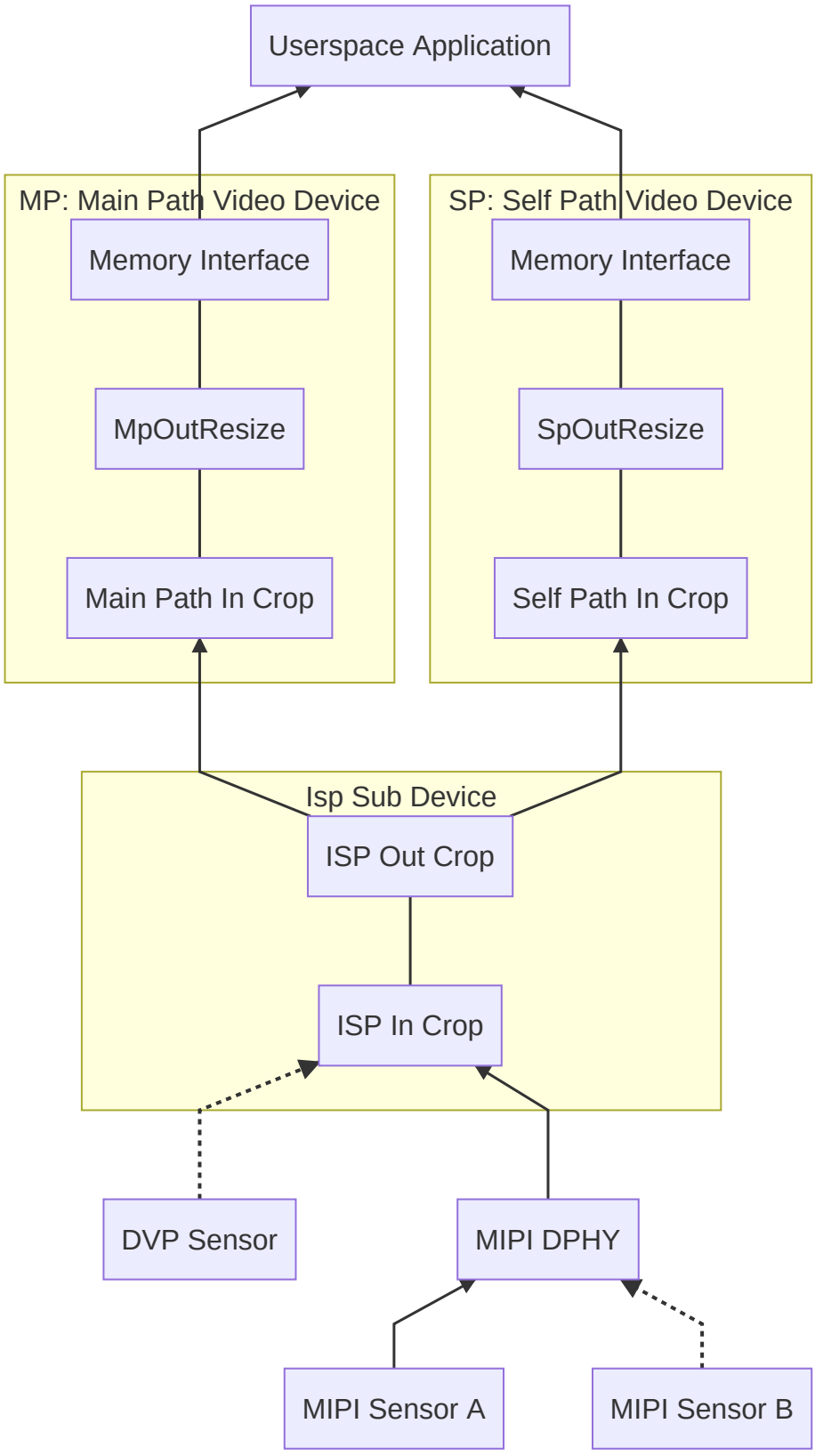
The driver code of RKISP1 is located in the `drivers/media/platform/rockchip/isp1` directory. It is mainly based on v4l2 / media framework to implement hardware configuration, interrupt processing, control buffer rotation, control subdevice (such as mipi dphy and sensor) power on and off functions.

A brief introduction to the contents of each file in the driver is as follows:

```
# tree drivers/media/platform/rockchip/isp1/
drivers/media/platform/rockchip/isp1/
├── capture.c    #Contains the configuration of mp/sp and vb2, frame interrupt processing
├── dev.c        #Include probe, sensor registration, clock, pipeline, iommu
├── isp_params.c #3A Related parameter settings
├── isp_stats.c  #3A related statistics
├── regs.c       #register related read and write operations
├── rkisp1.c     #corresponding to rkisp-isp-sd entity, including receiving data from mipi/dvp and crop function
```

The code for Mipi Dphy is located in `drivers/phy/rockchip/phy-rockchip-mipi-rx.c`. It is also a v4l2 sub-device.

The nodes in the figure below are the connection diagrams that users often encounter during development and usage, from sensor to ISP output.



As shown above, RKISP1 has the following features:

- It can be adapted to MIPI or DVP interface, MIPI DPHY is required when connecting to MIPI sensor.
- Multiple sensors can be connected, but only one is active at the same time.
- After an image is input to ISP, it can be divided into two outputs of MP and SP. After MP and SP are processed based on the same raw image, the two channels can be output simultaneously.
- MP stands for Main Path. it can output full resolution images, up to 4416x3312. MP can output yuv or raw images, and only MP can output raw images.
- SP stands for Self Path. Supports up to 1920x1080 resolution. SP can output yuv or rgb images, but not raw images.

- Both MP and SP have crop and resize functions, which do not affect each other.

The comparison of MP and SP output functions is as follows:

Output Device	Max Resolution	Supported Format	Crop/Resize
SP	1920x1080	YUV, RGB	Support
MP	4416x3312	YUV, RAW	Support

RKISP1 also has some other nodes such as rkisp1-input-params, rkisp1-statistics, which are specially used for 3A tunning; rkisp1_rawpath and rkisp1_dmapath are used in special cases, and generally will not be used during App development.

6.1 Rkisp1 dts Board Configuration

When RK Linux SDKs are released, if the chip supports ISP, the rkisp1 node has been defined in its dtsi, such as the isp node in rk3288-rkisp1.dtsi, and the rkisp1_0 and rkisp1_1 nodes in rk3399.dtsi. The following table describes ISP information of each chip.

Chip name	dts node name	corresponding mipi dphy	corresponding iommu
RK3399	rkisp1_0	mipi_dphy_rx0	isp0_mmu
RK3399	rkisp1_1	mipi_dphy_tx1rx1	isp1_mmu
RK3288	rkisp1	mipi_phy_rx0 or mipi_phy_tx1rx1	isp_mmu
PX30/RK3326	rkisp1	mipi_dphy_rx0	isp_mmu
RK1808	rkisp1	mipi_dphy_rx	isp_mmu

From the above table:

- RK3399 has two isps corresponding to different dphy and mmu respectively
- RK3288 has only one isp, but hardware dphy can choose rx0 or tx1rx1

During board-level configuration, you only have to enable the corresponding nodes separately and establish a remote-endpoint link relationship. Please refer to the existing configuration of kernel, such as: `arch/arm64/boot/dts/rockchip/rk3326-evb-lp3-v10-linux.dts`, enable sensor, mipi_dphy_rx0, rkisp1, isp_mmu respectively, and set remote-endpoint associate node.

It is especially important to note that for MIPI sensors, both of the data-lane parameter sensor and mipi_dphy_rx0 need to be configured correctly with the same value.

7. RKCIF Driver Introduction

TODO.