密级状态：绝密（　　）　秘密（　　）　内部（　）　公开（√ ）

RKISP_Driver_User_Manual

（ISP 部）

文件状态： [] 正在修改 [√] 正式发布	当前版本：	V1.1
	作　　者：	胡克俊
	完成日期：	2019-03-18
	审　　核：	邓达龙
	完成日期：	2019-03-18

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Electronics　Co . , Ltd

(版本所有,翻版必究)

版 本 历 史

版本号	作者	修改日期	修改说明	审核	备注
V1.0	胡克俊	2018-11-13	发布初版		
V1.1	胡克俊	2019-03-18	增加模组/OTP 信息说明；增加 VCM 驱动的 DTS 说明		

目 录

1. 文档适用平台 1

2. CAMERA 文件目录说明 1

3. RKISP 驱动说明 2

4. CAMERA 设备注册(DTS) 3

4.1 MIPI SENSOR 注册 3

4.2 DVP SENSOR 注册 6

5. CAMERA 设备驱动 9

5.1 数据类型简要说明 10

struct i2c_driver *10*

struct v4l2_subdev_core_ops *12*

struct v4l2_subdev_video_ops *13*

struct v4l2_subdev_pad_ops *14*

struct v4l2_ctrl_ops *16*

struct xxxx_mode *18*

5.2 API 简要说明 20

xxxx_set_fmt *20*

xxxx_get_fmt *20*

xxxx_enum_mbus_code *21*

xxxx_enum_frame_sizes *23*

xxxx_g_frame_interval *24*

xxxx_s_stream *24*

xxxx_runtime_resume *25*

xxxx_runtime_suspend *25*

xxxx_set_ctrl......26

6. MEDIA-CTL / V4L2-CTL 工具......**26**

7. RKISP 调试及常见的问题......**27**

1. 文档适用平台

芯片平台	软件系统	支持情况
RK3399	Linux(Kernel-4.4)	Y
RK3288	Linux(Kernel-4.4)	Y
RK3326	Linux(Kernel-4.4) Android-9.0	Y

2. Camera 文件目录说明

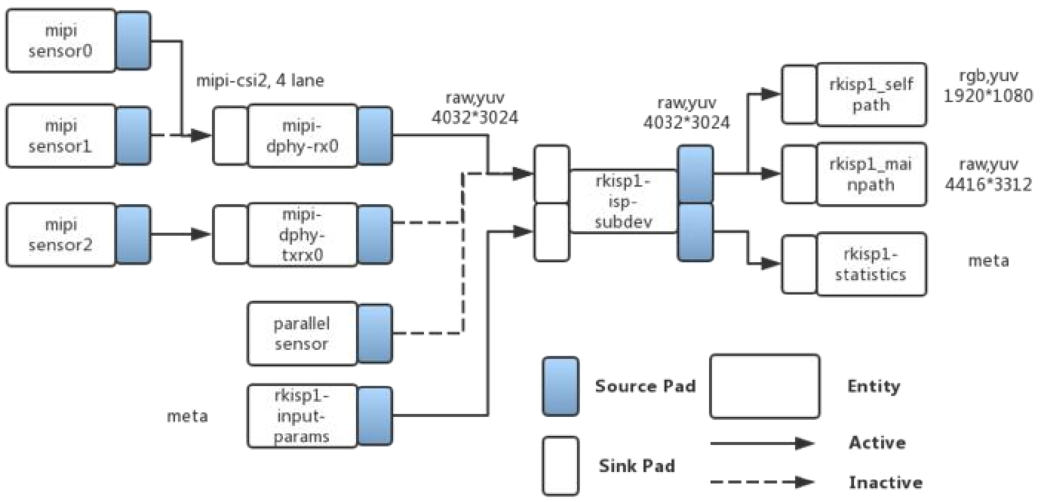
Linux Kernel-4.4：

```
  |
  |arch/arm64/boot/dts/rockchip       DTS 配置文件
  |drivers/media
    |
    |platform/rockchip/isp1           RKISP 驱动
      |—— capture.c                   包含 mp/sp 的配置及 vb2，帧中断处理
      |—— dev.c                       包含 probe、异步注册、clock、pipeline、
                                      iommu 及 media/v4l2 framework
      |—— isp_params.c                3A 相关参数设置
      |—— isp_stats.c                 3A 相关统计
      |—— mipi_dphy_sy.c              mipi dphy 驱动，它独立于 isp 驱动，
                                      将计划放在其它目录
      |—— regs.c                      寄存器相关的读写操作
      |—— rkisp1.c                    对应 isp_sd entity 节点，
                                      包含从 mipi 接收数据，并有 crop 功能
    |i2c/                             Camera Sensor 驱动
```

3. RKISP 驱动说明

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

下面的框图描述了 RKISP1 驱动的拓扑结构。



名称	类型	描述
rkisp1_mainpath	v4l2_vdev, capture	Format: YUV, RAW Bayer; Support: Crop
rkisp1_selfpath	v4l2_vdev, capture	Format: YUV, RGB; Support: Crop
rkisp1-isp-subdev	v4l2_subdev	Internal isp blocks; Support: source/sink pad crop. The format on sink pad should be equal to sensor input format, the size should be equal/less than 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.
rockchip-sy-mipi-dphy	v4l2_subdev	MIPI-DPHY Configure.
rkisp1-statistics	v4l2_vdev, capture	Provide Image color Statistics information.
rkisp1-input-params	v4l2_vdev, output	Accept params for AWB, BLC...... Image enhancement blocks.

4. Camera 设备注册(DTS)

RKISP 的 DTS 节点在 kernel 源码中有文档说明,

它位于 Documentation/devicetree/bindings/media/rockchip-isp1.txt。

mipi dphy 驱动节点也在 kernel 源码中有文档说明,

它位于 Documentation/devicetree/bindings/media/rockchip-mipi-dphy.txt

4.1 MIPI Sensor 注册

下面以 rk3399 isp0 和 ov13850 为例进行说明。

```
vm149c: vm149c@0c {
        compatible = "silicon touch,vm149c";
        status = "okay";
        reg = <0x0c>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";

};

ov13850: ov13850@10 {

        compatible = "ovti,ov13850"; // 需要与驱动中的匹配字符串一致

        status = "okay";

        reg = <0x10>; // sensor I2C 设备地址

        clocks = <&cru SCLK_CIF_OUT>; // sensor clickin 配置

        clock-names = "xvclk";

        reset-gpios = <&gpio2 10 GPIO_ACTIVE_HIGH>;

        // reset 管脚分配及有效电平

        pwdn-gpios = <&gpio1 4 GPIO_ACTIVE_HIGH>;

        // power 管脚分配及有效电平

        pinctrl-names = "rockchip,camera_default";

        pinctrl-0 = <&cif_clkout>; // pinctl 设置

        rockchip,camera-module-index = <0>; // 模组编号，该编号不要重复

        rockchip,camera-module-facing = "back"; // 模组朝向，有"back"和"front"

        rockchip,camera-module-name = "CMK-CT0116"; // 模组名

        rockchip,camera-module-lens-name = "Largan-50013A1"; // lens 名

        // 模组名和 lens 名被用来和 IQ xml 文件做匹配

        lens-focus = <&vm149c>; // vcm 驱动设置，支持 AF 时需要有这个设置

        port {

            ucam_out0: endpoint {

                remote-endpoint = <&mipi_in_ucam0>;

                // mipi dphy 端的 port 名

                data-lanes = <1 2>;

                // mipi lane 数，1lane 为 <1>, 4lane 为 <1 2 3 4>
            };

        };

    };

&mipi_dphy_rx0 {

    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>;

                remote-endpoint = <&ucam_out0>;

                // sensor 端的 port 名

                data-lanes = <1 2>;

                // mipi lane 数，1lane 为 <1>, 4lane 为 <1 2 3 4>

            };

        };

        port@1 {

            reg = <1>;

            #address-cells = <1>;

            #size-cells = <0>;

            dphy_rx0_out: endpoint@0 {

                reg = <0>;
                remote-endpoint = <&isp0_mipi_in>;
                // isp 端的 port 名
            };
        };
    };
};
&rkisp1_0 {
    status = "okay";
    port {
        #address-cells = <1>;
        #size-cells = <0>;
        isp0_mipi_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy_rx0_out>;
            // mipi dphy 端的 port 名
        };
    };
};
&isp0_mmu {
    status = "okay"; // isp 驱动使用了 iommu，所以 isp iommu 也需要打开
};
```

4.2 DVP Sensor 注册

以 rk3326 isp 和 gc0312/gc2145 为例进行说明。

```
&i2c2 {
    status = "okay";
```

```
gc0312@21 {
    status = "okay";
    compatible = "galaxycore,gc0312"; // 需要与驱动中的匹配字符串一致
    reg = <0x21>; // sensor I2C 设备地址
    pinctrl-names = "default";
    pinctrl-0 = <&cif_clkout_m0>; // pinctl 设置
    clocks = <&cru SCLK_CIF_OUT>; // sensor clickin 配置
    clock-names = "xvclk";
    avdd-supply = <&vcc2v8_dvp>; // sensor 电源配置
    dovdd-supply = <&vcc1v8_dvp>;
    dvdd-supply = <&vcc1v8_dvp>;
    pwdn-gpios = <&gpio2 14 GPIO_ACTIVE_HIGH>;
    // power 管脚分配及有效电平
    rockchip,camera-module-index = <1>; // 模组编号，该编号不要重复
    rockchip,camera-module-facing = "front"; // 模组朝向，有"back"和"front"
    rockchip,camera-module-name = "CameraKing"; // 模组名
    rockchip,camera-module-lens-name = "Largan"; // lens 名
    port {
        gc0312_out: endpoint {
            remote-endpoint = <&dvp_in_fcam>;
            // isp 端的 port 名
        };
    };
};
gc2145@3c {
    status = "okay";
```

```
        compatible = "galaxycore,gc2145"; // 需要与驱动中的匹配字符串一致

        reg = <0x3c>; // sensor I2C 设备地址

        pinctrl-names = "default";

        pinctrl-0 = <&cif_clkout_m0>; // pinctl 设置

        clocks = <&cru SCLK_CIF_OUT>; // sensor clickin 配置

        clock-names = "xvclk";

        avdd-supply = <&vcc2v8_dvp>; // sensor 电源配置

        dovdd-supply = <&vcc1v8_dvp>;

        dvdd-supply = <&vcc1v8_dvp>;

        pwdn-gpios = <&gpio2 13 GPIO_ACTIVE_HIGH>;

        // power 管脚分配及有效电平

        rockchip,camera-module-index = <0>; // 模组编号，该编号不要重复

        rockchip,camera-module-facing = "back"; // 模组朝向，有"back"和"front"

        rockchip,camera-module-name = "CameraKing"; // 模组名

        rockchip,camera-module-lens-name = "Largan"; // lens 名

        port {

            gc2145_out: endpoint {

                remote-endpoint = <&dvp_in_bcam>;

                // isp 端的 port 名

            };

        };

    };

};

&isp_mmu {

    status = "okay";

};
```

```
&rkisp1 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&cif_clkout_m0 &dvp_d0d1_m0 &dvp_d2d9_m0 &dvp_d10d11_m0>;
    // pinctl 设置，增加 dvp pin 脚相关配置
    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;
            dvp_in_fcam: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&gc0312_out>; // sensor 端的 port 名
            };
            dvp_in_bcam: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&gc2145_out>; // sensor 端的 port 名
            };
        };
    };
};
```

5. Camera 设备驱动

Camera Sensor 采用 I2C 与主控进行交互，目前 sensor driver 按照 I2C 设备驱动方式实现，

sensor driver 同时采用 v4l2 subdev 的方式实现与 host driver 之间的交互。

5.1 数据类型简要说明

struct i2c_driver

[说明]

定义 i2c 设备驱动信息

[定义]

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

[关键成员]

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

@probe	Callback for device binding
@remove	Callback for device unbinding

[示例]

```
#if IS_ENABLED(CONFIG_OF)
static const struct of_device_id ov13850_of_match[] = {
  { .compatible = "ovti,ov13850" },
  {},
};
MODULE_DEVICE_TABLE(of, ov13850_of_match);
#endif

static const struct i2c_device_id ov13850_match_id[] = {
  { "ovti,ov13850", 0 },
  { },
};

static struct i2c_driver ov13850_i2c_driver = {
  .driver = {
      .name = "ov13850",
      .pm = &ov13850_pm_ops,
      .of_match_table = of_match_ptr(ov13850_of_match),
  },
  .probe       = &ov13850_probe,
  .remove      = &ov13850_remove,
  .id_table    = ov13850_match_id,
};


static int __init sensor_mod_init(void)

{

  return i2c_add_driver(&ov13850_i2c_driver);

}


static void __exit sensor_mod_exit(void)

{

  i2c_del_driver(&ov13850_i2c_driver);

}


device_initcall_sync(sensor_mod_init);

module_exit(sensor_mod_exit);
```

struct v4l2_subdev_core_ops

[说明]

Define core ops callbacks for subdevs.

[定义]

```
struct v4l2_subdev_core_ops {

  ……

  long (*ioctl)(struct v4l2_subdev *sd, unsigned int cmd, void *arg);

#ifdef CONFIG_COMPAT

  long (*compat_ioctl32)(struct v4l2_subdev *sd, unsigned int cmd,

                          unsigned long arg);

#endif
```

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

[关键成员]

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

[示例]

```
static const struct v4l2_subdev_core_ops ov13850_core_ops = {
  .ioctl = ov13850_ioctl,
#ifdef CONFIG_COMPAT
  .compat_ioctl32 = ov13850_compat_ioctl32,
#endif
};
```

目前使用了如下的私有 ioctl 实现模组信息的查询和 OTP 信息的查询设置。

RKMODULE_GET_MODULE_INFO / RKMODULE_AWB_CFG

struct v4l2_subdev_video_ops

[说明]

Callbacks used when v4l device was opened in video mode.

[定义]

```
struct v4l2_subdev_video_ops {
  ……
  int (*s_stream)(struct v4l2_subdev *sd, int enable);
  ……
  int (*g_frame_interval)(struct v4l2_subdev *sd,
              struct v4l2_subdev_frame_interval *interval);
  int (*s_frame_interval)(struct v4l2_subdev *sd,
              struct v4l2_subdev_frame_interval *interval);
  ……
};
```

[关键成员]

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

[示例]

```
static const struct v4l2_subdev_video_ops ov13850_video_ops = {
  .s_stream = ov13850_s_stream,
  .g_frame_interval = ov13850_g_frame_interval,
};
```

struct v4l2_subdev_pad_ops

[说明]

v4l2-subdev pad level operations

[定义]

```
struct v4l2_subdev_pad_ops {
    ……


    int (*enum_mbus_code)(struct v4l2_subdev *sd,
                          struct v4l2_subdev_pad_config *cfg,
                          struct v4l2_subdev_mbus_code_enum *code);
    int (*enum_frame_size)(struct v4l2_subdev *sd,
                           struct v4l2_subdev_pad_config *cfg,
                           struct v4l2_subdev_frame_size_enum *fse);
    int (*get_fmt)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_format *format);
    int (*set_fmt)(struct v4l2_subdev *sd,
                   struct v4l2_subdev_pad_config *cfg,
                   struct v4l2_subdev_format *format);
    ……


};
```

[关键成员]

成员名称	描述
. enum_mbus_code	callback for VIDIOC_SUBDEV_ENUM_MBUS_CODE ioctl handler code.
. enum_frame_size	callback for VIDIOC_SUBDEV_ENUM_FRAME_SIZE ioctl handler code.
.s_fmt	callback for VIDIOC_SUBDEV_S_FMT ioctl handler code.
.g_fmt	callback for VIDIOC_SUBDEV_G_FMT ioctl handler code

[示例]

```
static const struct v4l2_subdev_pad_ops ov13850_pad_ops = {
  .enum_mbus_code = ov13850_enum_mbus_code,
  .enum_frame_size = ov13850_enum_frame_sizes,
  .get_fmt = ov13850_get_fmt,
  .set_fmt = ov13850_set_fmt,
};
```

struct v4l2_ctrl_ops

[说明]

The control operations that the driver has to provide.

[定义]

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

[关键成员]

成员名称	描述
.g_volatile_ctrl	get a new value for this control, generally only relevant for volatile (and usually read-only) controls .
.try_ctrl	test whether the control's value is valid.
.s_ctrl	actually set the new control value.

[示例]

```
static const struct v4l2_ctrl_ops ov13850_ctrl_ops = {
  .s_ctrl = ov13850_set_ctrl,
};
```

Rkisp 驱动要求使用框架提供的 user controls 功能， cameras sensor 驱动必须实现如下 control 功能。

成员名称	描述
V4L2_CID_VBLANK	Vertical blanking. The idle period after every frame during which no image data is produced. The unit of vertical blanking is a line. Every line has length of the image width plus horizontal blanking at the pixel rate defined by V4L2_CID_PIXEL_RATE control in the same sub-device.
V4L2_CID_HBLANK	Horizontal blanking. The idle period after every line of image data during which no image data is produced. The unit of horizontal blanking is pixels.
V4L2_CID_EXPOSURE	Determines the exposure time of the camera sensor. The exposure time is limited by the frame interval.
V4L2_CID_ANALOGUE_GAIN	Analogue gain is gain affecting all colour components in the pixel matrix. The gain operation is performed in the analogue domain before A/D conversion.
V4L2_CID_PIXEL_RATE	Pixel rate in the source pads of the subdev. This control is read-only and its unit is pixels / second.
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.

struct xxxx_mode

[说明]

Sensor 能支持各个模式的信息。

这个结构体在 sensor 驱动中常常可以见到，虽然它不是 v4l2 标准要求的。

[定义]

```
struct xxxx_mode {
  u32 width;
  u32 height;
  struct v4l2_fract max_fps;
  u32 hts_def;
  u32 vts_def;
  u32 exp_def;
  const struct regval *reg_list;
};
```

[关键成员]

成员名称	描述
.width	有效图像宽度
.height	有效图像高度

.max_fps	图像 FPS，denominator/numerator 为 fps
hts_def	默认 HTS，为有效图像宽度 + HBLANK
vts_def	默认 VTS，为有效图像高度 + VBLANK
exp_def	默认曝光时间
*reg_list	寄存器列表

[示例]

```
static const struct ov13850_mode supported_modes[] = {
  {
        .width = 2112,
        .height = 1568,
        .max_fps = {
            .numerator = 10000,
            .denominator = 300000,
        },
        .exp_def = 0x0600,
        .hts_def = 0x12c0,
        .vts_def = 0x0680,
        .reg_list = ov13850_2112x1568_regs,
  },{
        .width = 4224,
        .height = 3136,
        .max_fps = {
            .numerator = 20000,
            .denominator = 150000,
        },
        .exp_def = 0x0600,
        .hts_def = 0x12c0,

        .vts_def = 0x0d00,

        .reg_list = ov13850_4224x3136_regs,

    },

};
```

5.2 API 简要说明

xxxx_set_fmt

[描述]

设置 sensor 输出格式。

[语法]

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

[参数]

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

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_get_fmt

[描述]

获取 sensor 输出格式。

[语法]

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

[参数]

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

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_enum_mbus_code

[描述]

枚举 sensor 输出 bus format。

[语法]

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

[参数]

参数名称	描述	输入输出
*sd	v4l2 subdev 结构体指针	输入
*cfg	subdev pad information 结构体指针	输入

*code	media bus format enumeration 结构体指针	输出

[返回值]

返回值	描述
0	成功
非 0	失败

下表总结了各种图像类型对应的 format

图像类型	Sensor 输出 format
Bayer RAW	MEDIA_BUS_FMT_SBGGR10_1X10 MEDIA_BUS_FMT_SRGGB10_1X10 MEDIA_BUS_FMT_SGBRG10_1X10 MEDIA_BUS_FMT_SGRBG10_1X10 MEDIA_BUS_FMT_SRGGB12_1X12 MEDIA_BUS_FMT_SBGGR12_1X12 MEDIA_BUS_FMT_SGBRG12_1X12 MEDIA_BUS_FMT_SGRBG12_1X12 MEDIA_BUS_FMT_SRGGB8_1X8 MEDIA_BUS_FMT_SBGGR8_1X8 MEDIA_BUS_FMT_SGBRG8_1X8 MEDIA_BUS_FMT_SGRBG8_1X8
YUV	MEDIA_BUS_FMT_YUYV8_2X8 MEDIA_BUS_FMT_YVYU8_2X8 MEDIA_BUS_FMT_UYVY8_2X8 MEDIA_BUS_FMT_VYUY8_2X8 MEDIA_BUS_FMT_YUYV10_2X10 MEDIA_BUS_FMT_YVYU10_2X10

	MEDIA_BUS_FMT_UYVY10_2X10 MEDIA_BUS_FMT_VYUY10_2X10 MEDIA_BUS_FMT_YUYV12_2X12 MEDIA_BUS_FMT_YVYU12_2X12 MEDIA_BUS_FMT_UYVY12_2X12 MEDIA_BUS_FMT_VYUY12_2X12
Only Y(黑白)	MEDIA_BUS_FMT_Y8_1X8 MEDIA_BUS_FMT_Y10_1X10 MEDIA_BUS_FMT_Y12_1X12

xxxx_enum_frame_sizes

[描述]

枚举 sensor 输出大小。

[语法]

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

[参数]

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

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_g_frame_interval

[描述]

获取 sensor 输出 fps。

[语法]

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

[参数]

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

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_s_stream

[描述]

设置 stream 输入输出。

[语法]

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

[参数]

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

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_runtime_resume

[描述]

sensor 上电时的回调函数。

[语法]

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

[参数]

参数名称	描述	输入输出
*dev	device 结构体指针	输入

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_runtime_suspend

[描述]

sensor 下电时的回调函数。

[语法]

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

[参数]

参数名称	描述	输入输出
*dev	device 结构体指针	输入

[返回值]

返回值	描述
0	成功
非 0	失败

xxxx_set_ctrl

[描述]

设置各个 control 的值。

[语法]

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

[参数]

参数名称	描述	输入输出
*ctrl	v4l2_ctrl 结构体指针	输入

[返回值]

返回值	描述
0	成功
非 0	失败

6. media-ctl / v4l2-ctl 工具

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

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

具体用法可以参考命令的帮助信息, 下面是常见的几个使用。

1) 打印拓扑结构

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

2) 修改 fmt/size

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

3） 设置 fmt 并抓帧

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

4） 设置曝光、gain 等 control

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

7. RKISP 调试及常见的问题

1） 判断 rkisp 驱动加载状态

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

```
localhost ~ # grep '' /sys/class/video4linux/video*/name
/sys/class/video4linux/video3/name:rkisp1_selfpath
/sys/class/video4linux/video4/name:rkisp1_mainpath
/sys/class/video4linux/video5/name:rkisp1-statistics
/sys/class/video4linux/video6/name:rkisp1-input-params
```

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

2） 判断 camera 驱动是否加载成功

当所有的 camera 都注册完毕，kernel 会打印出如下的 log。

```
localhost ~ # dmesg | grep Async
[ 0.682982] rkisp1: Async subdev notifier completed
```

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

3） 抓取 isp 输出的 yuv 数据

参考命令如下，

```
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:YUYV8_2X8/2592x1944]'

media-ctl -d /dev/media0 --set-v4l2 '"rkisp1-isp-subdev":2[crop:(0,0)/2592x1944]'

v4l2-ctl -d /dev/video4 \

--set-selection=target=crop,top=336,left=432,width=1920,height=1080 \

--set-fmt-video=width=1280,height=720,pixelformat=NV21 \

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

4） 抓取 Sensor 输出的 Raw Bayer 原始数据

参考命令如下，

```
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=SBGGR10 \

--stream-mmap=3 --stream-to=/tmp/mp.raw.out --stream-count=1 --stream-poll
```

需要注意的是， ISP 虽然不对 Raw 图处理， 但它仍然会将 10bit 的数据低位补 0 成 16bit。 不管 Sensor 输入的是 10bit/12bit， 最终上层得到的都是 16bit 每像素。

5） 如何支持黑白摄像头

Camera sensor 驱动需要将黑白 sensor 的输出 format 改为如下三种 format 之一，

MEDIA_BUS_FMT_Y8_1X8 (sensor 8bit 输出)

MEDIA_BUS_FMT_Y10_1X10 (sensor 10bit 输出)

MEDIA_BUS_FMT_Y12_1X12 (sensor 12bit 输出)

即在函数 xxxx_get_fmt 和 xxxx_enum_mbus_code 返回上述 format。

Rkisp 驱动会对这三种 format 进行特别设置，以支持获取黑白图像。

另外，如应用层需要获取 Y8 格式的图像，则只能使用 SP Path，因为只有 SP Path 可以支持 Y8 格式输出。