Rockchip_Android_Fast_Reverse_Image_System_Developer_Guide

文件标识：RK-KF-YF-319

发布版本：V1.0.3

日期：2023-03-30

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

本文档是基于RK3588/RK356x/RK3368 ANDROID平台开发快速倒车影像功能的帮助文档。

概述

本快速倒车影像系统基于Android平台开发，内核版本为Linux 4.19（RK356x/rk3368），Linux5.10（RK3588）。适用于rk356x、rk3588平台的车机中控倒车影像和流媒体后视镜等产品的快速倒车开发需求。

产品版本

芯片名称	内核版本	**Android**版本
rk3368/px5	Linux 4.19	Android12.0
rk3566/rk3568	Linux 4.19	Android11.0
rk3588	Linux 5.10	Android12.0
rk3562	Linux 5.10	Android13.0

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	范建威	2022-3-31	初始版本
V1.0.1	范建威	2022-4-20	修改格式错误 增加显示相关配置
V1.0.2	范建威	2023-3-28	修改相关错误
V1.0.3	范建威	2023-3-30	增加rk3562相关说明

目录

1. 快速倒车影像系统介绍

1.1 简要说明

倒车视频影像(简称 RIS)就是在车尾安装了倒车摄像头，当挂入倒档时，该系统会自动接通位于车尾的摄像头，将车后状况显示于中控或后视镜的液晶显示屏上。

1.2 特点

快速进入：上电最快约 3.0 秒进入倒车影像界面。

快速切换：apk直接对vehicle节点控制，快速切换dvd与倒车视频。

后台开机：倒车影像系统在开机过程中，安卓系统仍能保持后台开机。

快速切换：倒车影像系统在开机过程，倒车影像界面和Android系统界面可快速切换。

控制方便：通过检测 IO 口电平变化，进行倒车影像系统与安卓系统之间的切换。

1.3 camera类型

1.3.1 MIPI camera

MIPI为低压差分信号，传输速度快，抗干扰能力强，一般支持800万及以上像素的camera。目前主流手机camera模组都使用MIPI传输，传输时使用4对差分数据信号和1对差分时钟信号 。

1.3.2 DVP camera

DVP Camera或称为并口Camera，接口如下图所示，一般支持BT601/BT656/BT1120数据的传输。DVP是并口传输，速度相对较慢，传输的带宽较低，一般用于500万像素以下的camera。需要使用PCLK时钟、VSYNC场同步、HSYNC行同步、D[0：11]并口数据，可以是8/10/12bit/16bit数据位大小。

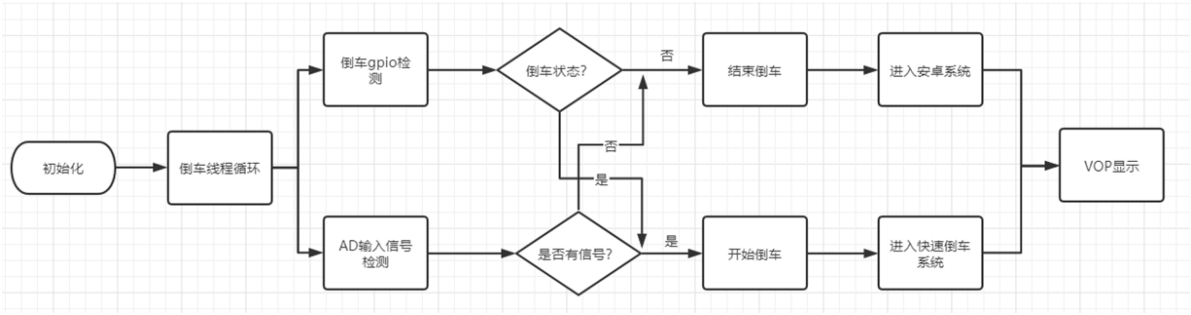
1.3.3 各平台支持camera情况

芯片名称	**MIPI接口**	**DVP接口**
rk3588/rk356x	支持	BT601/BT656/BT1120
rk3368	不支持	BT601/BT656
rk3562	支持	不支持

1.4 快速倒车流程

本快速倒车系统硬件流程主要是获取外部摄像头输入的信号，将信号送至主控芯片的CIF控制器进行捕获处理，最后把图像数据经过IPP/IEP（硬件加速处理）以及RGA/IPP（旋转镜像）等处理后送至显示器显示。总体流程图如图所示。

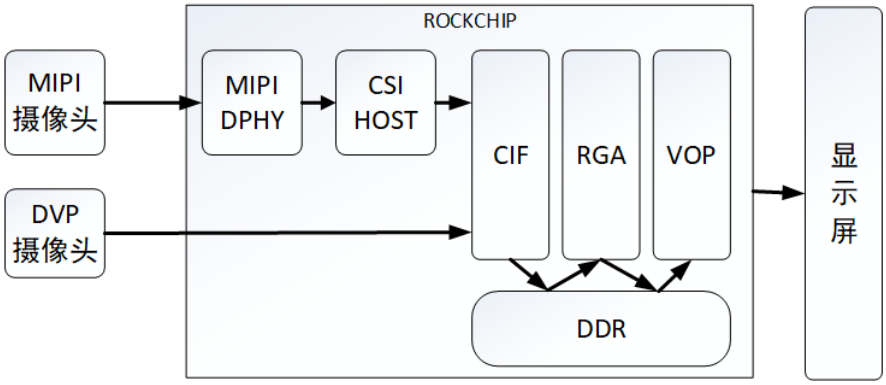
rk356x/rk3588平台支持直接获取MIPI接口和DVP接口的摄像头信号。rk3368支持获取DVP接口的信号，一般是从摄像头获取的模拟CVBS信号，经过TP2825AD7181D等转换芯片的AD转换模块处理后，输出BT656视频信号。



1.5 数据流处理框图

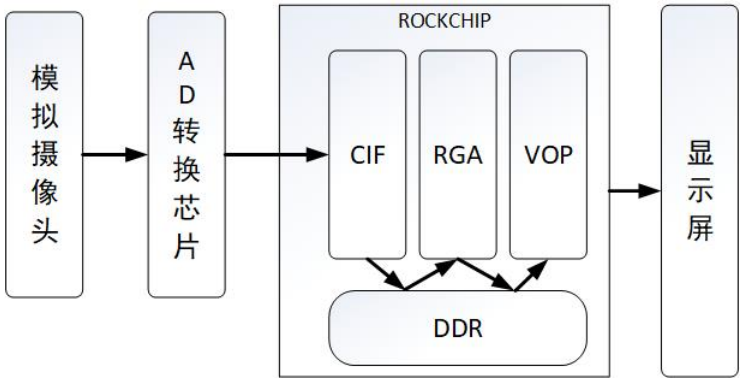
1.5.1 rk3588/rk356x数据处理框图

rk3588/rk356x快速倒车信号处理流程如图所示。



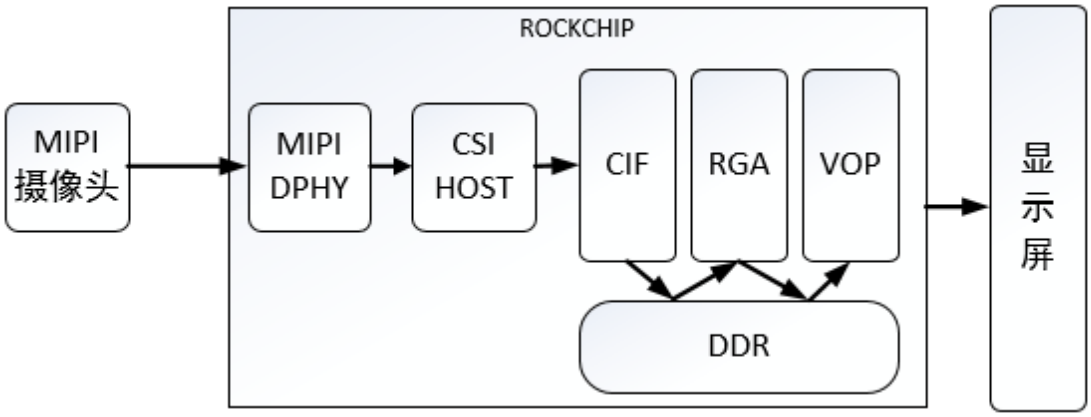
1.5.2 rk3368数据处理框图

rk3368平台快速倒车数据处理流程如图所示。



1.5.3 rk3562数据处理框图

rk3562快速倒车数据处理流程如图所示。



1.6 相关代码

vehicle驱动相关代码：

```
drivers/video/rockchip/vehicle/vehicle_samsung_dcphy_common.h
drivers/video/rockchip/vehicle/vehicle-csi2-dphy-common.h
drivers/video/rockchip/vehicle/vehicle_ad.h
drivers/video/rockchip/vehicle/vehicle_ad_7181.c
drivers/video/rockchip/vehicle/vehicle_ad_7181.h
drivers/video/rockchip/vehicle/vehicle_ad_gc2145.c
drivers/video/rockchip/vehicle/vehicle_ad_gc2145.h
drivers/video/rockchip/vehicle/vehicle_ad_nvp6324.c
drivers/video/rockchip/vehicle/vehicle_ad_nvp6324.h
drivers/video/rockchip/vehicle/vehicle_ad_tp2825.c
drivers/video/rockchip/vehicle/vehicle_ad_tp2825.h
drivers/video/rockchip/vehicle/vehicle_cfg.h
drivers/video/rockchip/vehicle/vehicle_cif.c
drivers/video/rockchip/vehicle/vehicle_cif.h
drivers/video/rockchip/vehicle/vehicle_cif_format.h
drivers/video/rockchip/vehicle/vehicle_dev.c
drivers/video/rockchip/vehicle/vehicle_flinger.c
drivers/video/rockchip/vehicle/vehicle_flinger.h
drivers/video/rockchip/vehicle/vehicle_generic_sensor.c
drivers/video/rockchip/vehicle/vehicle_gpio.c
drivers/video/rockchip/vehicle/vehicle_gpio.h
drivers/video/rockchip/vehicle/vehicle_main.c
drivers/video/rockchip/vehicle/vehicle_main.h
```

2. 关键配置

2.1 kernel配置

rockchip_config文件需要打开如下倒车配置

```
CONFIG_VIDEO_REVERSE_IMAGE=y
```

kernel-5.10平台还需要打开如下drm direct show的配置：

```
CONFIG_ROCKCHIP_DRM_DIRECT_SHOW=y
```

sensor 相关需要打开（以NVP6188为例，需要支持快速倒车切换到v4l2框架）

```
CONFIG_VIDEO_NVP6188=y
CONFIG_VIDEO_REVERSE_NVP6188=y
```

2.2 倒车脚配置

只需要知道倒车脚接主控的引脚号，在dts中进行配置即可。如40号引脚就等于1x32+1x8+0，所以配置dts时就是gpio1 RK_PB0 GPIO_ACTIVE_HIGH。

```
gpio_det: gpio-det {
    status = "okay";

    pinctrl-names = "default";
    pinctrl-0 = <&vehicle_gpios>;

    car-reverse {
        car-reverse-gpios = <&gpio1 RK_PB0 GPIO_ACTIVE_HIGH>;
        linux,debounce-ms = <5>;
        label = "car-reverse";
        gpio,wakeup;
    };
};
```

配置对应的pinctrl:

```
&pinctrl {
    vehicle {
        vehicle_gpios: vehicle-gpios {
            /* gpios */
            rockchip,pins =
                /* car-reverse */
                <1 RK_PB0 RK_FUNC_GPIO &pcfg_pull_up>;
        };
    };
};
```

2.3 vehicle配置

2.3.1 rk356x

以rk3588 kernel-5.10为例，rk356x kernel-4.19配置略有不同，可以参考对应的dts示例rk356x-vehicle-and-sensor.dtsi

2.3.2 rk3588

rk3588 vehicle设备配置示例如下：

```
vehicle: vehicle {
    compatible = "rockchip,vehicle";
    status = "okay";

    //vehicle,reset-gpio = <&gpio3 12 GPIO_ACTIVE_LOW>;
    //vehicle,reset-active-low;
    pinctrl-names = "vehicle";
    // pinctrl-0 = <&mipim1_camera2_clk>; //dcphy1
    pinctrl-0 = <&mipim1_camera1_clk>; //dcphy0
    // pinctrl-0 = <&mipim1_camera3_clk>;
    // pinctrl-0 = <&mipim0_camera4_clk>;
    // pinctrl-0 = <&cif_clk &cif_dvp_clk &cif_dvp_bus16>;

    clocks = <&cru ACLK_VICAP>,
             <&cru HCLK_VICAP>,
             <&cru DCLK_VICAP>;
    clock-names = "aclk_cif",
                  "hclk_cif",
                  "dclk_cif";
    resets = <&cru SRST_A_VICAP>,
             <&cru SRST_H_VICAP>,
             <&cru SRST_D_VICAP>;
    reset-names = "rst_cif_a",
                  "rst_cif_h",
                  "rst_cif_d";
    assigned-clocks = <&cru DCLK_VICAP>;
    assigned-clock-rates = <600000000>;
    power-domains = <&power RK3588_PD_VI>;
    cif,drop-frames = <4>; //frames to drop
    cif,chip-id = <1>; /*0:rk3568 1:rk3588*/
    rockchip,grf = <&sys_grf>;
    rockchip,cru = <&cru>;
    rockchip,cif = <&rkcif>;
    rockchip,gpio-det = <&gpio_det>;
    rockchip,cif-sensor = <&cif_sensor>;
    rockchip,cif-phy = <&cif_phy>;
    ad,fix-format = <0>;//0:auto detect,1:pal;2:ntsc;3:720p50;4:720p30;5:720p25
    /*0:no, 1:90; 2:180; 4:270; 0x10:mirror-y; 0x20:mirror-x*/
    vehicle,rotate-mirror = <0x00>;
    vehicle,crtc_name = "video_port3";
    vehicle,plane_name = "Esmart0-win0";
};
```

2.3.3 rk3368

引用rk3368对应cif相关的资源

```
    vehicle: vehicle {
        compatible = "rockchip,vehicle";
        status = "okay";

        pinctrl-names = "vehicle";
        pinctrl-0 = <&cif_clkout &isp_dvp_d2d9 &isp_dvp_d10d11>;

        clocks = <&cru PCLK_VIP>, <&cru ACLK_VIP>,
             <&cru HCLK_VIP>, <&cru SCLK_VIP_SRC>,
             <&cru SCLK_VIP_OUT>;
        clock-names = "pclk_cif", "aclk_cif0",
                  "hclk_cif0", "cif0_in",
                 "xvclk";
        resets = <&cru SRST_VIP>;
        reset-names = "rst_cif";
        power-domains = <&power RK3368_PD_VIO>;
        cif,drop-frames = <4>; //frames to drop
        rockchip,grf = <&grf>;
        rockchip,cru = <&cru>;
        rockchip,cif = <&cif_new>;
        rockchip,gpio-det = <&gpio_det>;
        rockchip,cif-sensor = <&cif_sensor>;
        ad,fix-format = <0>;//0:auto
detect,1:pal;2:ntsc;3:720p50;4:720p30;5:720p25
        /*0:no, 1:90; 2:180; 4:270; 0x10:mirror-y; 0x20:mirror-x*/
        vehicle,rotate-mirror = <0x00>;
    };
```

2.3.4 rk3562

rk3562是kernel-5.10平台开发，配置与rk3588类似，修改对应的clk和rst资源，cif,chip-id等配置即可。

```
    vehicle: vehicle {
        compatible = "rockchip,vehicle";
        status = "okay";

        // pinctrl-names = "default";
        // pinctrl-0 = <&camm0_clk0_out>;

        clocks = <&cru ACLK_VICAP>,
            <&cru HCLK_VICAP>,
            <&cru DCLK_VICAP>,
            <&cru CSIRX0_CLK_DATA>,
            <&cru CSIRX1_CLK_DATA>,
            <&cru CSIRX2_CLK_DATA>,
            <&cru CSIRX3_CLK_DATA>;
        clock-names = "aclk_cif",
                  "hclk_cif",
                  "dclk_cif",
                  "csirx0_data",
```

```
                  "csirx1_data",
                  "csirx2_data",
                  "csirx3_data";
        resets = <&cru SRST_A_VICAP>,
            <&cru SRST_H_VICAP>,
            <&cru SRST_D_VICAP>,
            <&cru SRST_I0_VICAP>,
            <&cru SRST_I1_VICAP>,
            <&cru SRST_I2_VICAP>,
            <&cru SRST_I3_VICAP>;
        reset-names = "rst_cif_a",
                  "rst_cif_h",
                  "rst_cif_d",
                  "rst_cif_i0",
                  "rst_cif_i1",
                  "rst_cif_i2",
                  "rst_cif_i3";
        power-domains = <&power RK3562_PD_VI>;
        cif,drop-frames = <4>; //frames to drop
        cif,chip-id = <2>; /*0:rk3568 1:rk3588 2:rk3562*/
        rockchip,grf = <&sys_grf>;
        rockchip,cru = <&cru>;
        rockchip,cif = <&rkcif>;
        rockchip,gpio-det = <&gpio_det>;
        rockchip,cif-sensor = <&cif_sensor>;
        rockchip,cif-phy = <&cif_phy>;
        ad,fix-format = <0>;//0:auto
detect,1:pal;2:ntsc;3:720p50;4:720p30;5:720p25
        /*0:no, 1:90; 2:180; 4:270; 0x10:mirror-y; 0x20:mirror-x*/
        vehicle,rotate-mirror = <0x00>;
        vehicle,crtc_name = "video_port0";
        vehicle,plane_name = "Esmart0-win0";
    };
```

2.4 sensor配置

以nvp6324为例，需要配置sensor的gpio、i2c地址等，关键配置如下：

```
cif_sensor: cif_sensor {
    compatible = "rockchip,sensor";
    status = "okay";

    nvp6324 {
        status = "okay";
        //power-gpios = <&gpio0 RK_PC1 GPIO_ACTIVE_HIGH>;
        //pwr_active = <PWR_ACTIVE_HIGH>;
        powerdown-gpios = <&gpio1 RK_PA1 GPIO_ACTIVE_HIGH>;
        pwdn_active = <1>;
        // reset-gpios = <&gpio4 RK_PA6 GPIO_ACTIVE_HIGH>;
        // rst_active = <PWR_ACTIVE_HIGH>;
        orientation = <90>;
        i2c_add = <0x60>;                                   //i2c地址，这里是8位地址
        i2c_chl = <5>;                                      //i2c总线地址
        cif_chl = <0>;                                      //cif通道
        ad_chl = <0>;
```

```
        mclk_rate = <24>;
        rockchip,camera-module-defrect0 = <1920 1080 0 0 1920 1080>;
    };
};
```

2.5 mipi/dvp接口配置

2.5.1 rk356x/rk3368

rk356x仅支持一个mipi-dphy，因此不需要针对mipi接口和dvp接口单独配置。

rk3368仅支持dvp接口，不需要这个配置。

2.5.2 rk3588

rk3588共支持2个mipi-dphy，2个mipi-dcphy，1个dvp接口，因此需要在dts配置。mipi-dphy和mipi-dcphy的配置略有不同，具体配置如下，将实际使用的接口状态打开即可。

```
cif_phy: cif_phy {
    status = "okay";

    csi2_dcphy0 {
        status = "okay";
        clocks = <&cru CLK_MIPI_CAMARAOUT_M1>,
                 <&cru PCLK_MIPI_DCPHY0>,
                 <&cru PCLK_CSI_HOST_0>,
                 <&cru ICLK_CSIHOST0>;
        clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host",
                      "iclk_csi2host";
        resets = <&cru SRST_P_CSI_HOST_0>,
                 <&cru SRST_CSIHOST0_VICAP>;
        reset-names = "srst_csihost_p",
                      "srst_csihost_vicap";
        csihost-idx = <0>;
        rockchip,csi2 = <&mipi0_csi2>;
        phys = <&mipi_dcphy0>;
        phy-names = "dcphy";
    };
    csi2_dcphy1 {
        status = "disabled";
        clocks = <&cru CLK_MIPI_CAMARAOUT_M2>,
                 <&cru PCLK_MIPI_DCPHY1>,
                 <&cru PCLK_CSI_HOST_1>,
                 <&cru ICLK_CSIHOST1>;
        clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host",
                      "iclk_csi2host";
        resets = <&cru SRST_P_CSI_HOST_1>,
                 <&cru SRST_CSIHOST1_VICAP>;
        reset-names = "srst_csihost_p",
```

```
                      "srst_csihost_vicap";
        csihost-idx = <1>;
        rockchip,csi2 = <&mipi1_csi2>;
        phys = <&mipi_dcphy1>;
        phy-names = "dcphy";
    };
    csi2_dphy0 {
        status = "disabled";
        clocks = <&cru CLK_MIPI_CAMARAOUT_M3>,
                 <&cru PCLK_CSIPHY0>,
                 <&cru PCLK_CSI_HOST_2>;
        clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host";
        resets = <&cru SRST_CSIPHY0>,
                 <&cru SRST_P_CSIPHY0>,
                 <&cru SRST_P_CSI_HOST_2>,
                 <&cru SRST_CSIHOST2_VICAP>;
        reset-names = "srst_csiphy",
                      "srst_p_csiphy",
                      "srst_csihost_p",
                      "srst_csihost_vicap";
        csihost-idx = <2>;
        rockchip,dphy-grf = <&mipidphy0_grf>;
        rockchip,csi2-dphy = <&csi2_dphy0_hw>;
        rockchip,csi2 = <&mipi2_csi2>;
    };
    /* only rk3588 */
    csi2_dphy3 {
        status = "disabled";
        clocks = <&cru CLK_MIPI_CAMARAOUT_M4>,
                 <&cru PCLK_CSIPHY1>,
                 <&cru PCLK_CSI_HOST_4>;
        clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host";
        resets = <&cru SRST_CSIPHY1>,
                 <&cru SRST_P_CSIPHY1>,
                 <&cru SRST_P_CSI_HOST_4>,
                 <&cru SRST_CSIHOST4_VICAP>;
        reset-names = "srst_csiphy",
                      "srst_p_csiphy",
                      "srst_csihost_p",
                      "srst_csihost_vicap";
        csihost-idx = <4>;
        rockchip,dphy-grf = <&mipidphy1_grf>;
        rockchip,csi2-dphy = <&csi2_dphy1_hw>;
        rockchip,csi2 = <&mipi4_csi2>;
    };
    rkcif_dvp {
        status = "disabled";
        clocks = <&cru CLK_CIFOUT_OUT>;
        clock-names = "xvclk";
    };
};
```

若使用的是mipi_dcphy，则需要打开对应的节点：

```
/*if use rk3588-dcphy0 need enable this node*/
&mipi_dcphy0 {
    status = "okay";
};

/*if use rk3588-dcphy1 need enable this node*/
&mipi_dcphy1 {
    status = "okay";
};
```

需要注意的是，mipi_dcphy节点是与显示相关的共用，因此不需要使用的时候，不要单独配置即可，需要使用的时候再添加打开。

2.5.3 rk3562

rk3562有两路mipi dphy，不支持dvp接口，需要按需选择配置：

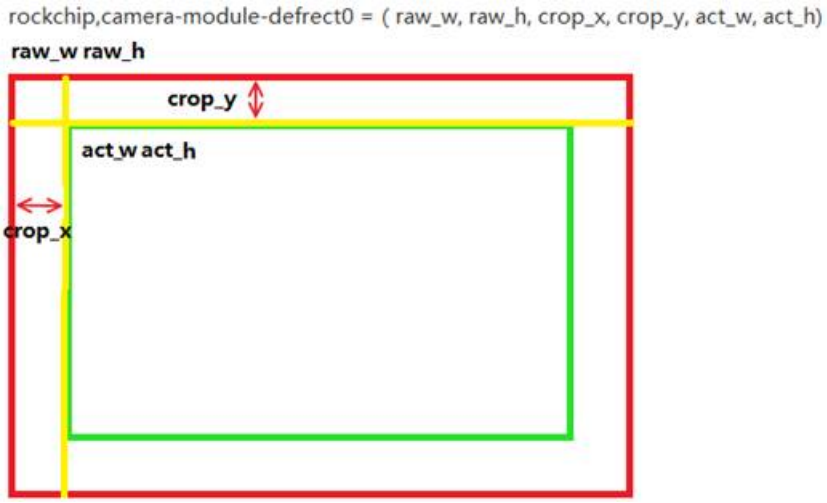
```
    cif_phy: cif_phy {
        status = "okay";

        csi2_dphy0 {
            status = "okay";
            clocks = <&cru CLK_CAM0_OUT2IO>,
                 <&cru PCLK_CSIPHY0>,
                 <&cru PCLK_CSIHOST0>;
            clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host";
            resets = <&cru SRST_P_CSIPHY0>,
                 <&cru SRST_P_CSIHOST0>;
            reset-names = "srst_p_csiphy",
                      "srst_csihost_p";
            csihost-idx = <0>;
            rockchip,csi2-dphy = <&csi2_dphy0_hw>;
            rockchip,csi2 = <&mipi0_csi2>;
        };
        csi2_dphy3 {
            status = "disabled";
            clocks = <&cru CLK_CAM2_OUT2IO>,
                <&cru PCLK_CSIPHY1>,
                <&cru PCLK_CSIHOST2>;
            clock-names = "xvclk",
                      "pclk",
                      "pclk_csi2host";
            resets = <&cru SRST_P_CSIPHY1>,
                 <&cru SRST_P_CSIHOST2>;
            reset-names = "srst_p_csiphy",
                      "srst_csihost_p";
            csihost-idx = <2>;
            rockchip,csi2-dphy = <&csi2_dphy1_hw>;
            rockchip,csi2 = <&mipi2_csi2>;
        };
    };
```

2.6 裁减配置

在dts配置vehicle设备的时候设置，具体的裁减方式如图所示。

```
rockchip,camera-module-defrect0 = <1920 1080 0 0 1920 1080>;
```



2.7 旋转、镜像配置

目前旋转镜像同时只能配置一个，不能同时配置旋转和镜像。dts配置的时候设置：

```
/*0:no, 1:90; 2:180; 4:270; 0x10:mirror-y; 0x20:mirror-x*/
vehicle,rotate-mirror = <0x04>;
```

同时驱动代码vehicle_flinger_I2O1.c配置旋转角度：

```
diff --git a/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
b/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
index 793cee293658..9f1f497a32e8
--- a/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
+++ b/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
@@ -57,7 +57,7 @@ enum force_value {
        FORCE_XOFFSET = 0,
        FORCE_YOFFSET = 0,
        FORCE_FORMAT = HAL_PIXEL_FORMAT_YCrCb_NV12,
-       FORCE_ROTATION = RGA_TRANSFORM_ROT_0,
+       FORCE_ROTATION = RGA_TRANSFORM_ROT_270,
 };

 enum {
```

2.8 显示设置

2.8.1 kernel设置图层

2.8.1.1 RK3588/RK3562

kernel-5.10支持从dts配置显示图层和video_port，配置如下：

```
vehicle,crtc_name = "video_port3";
vehicle,plane_name = "Esmart0-win0";
```

上述配置说明显示在video_port3，图层Esmart0-win0。其中，crtc可以根据实际dts的配置来选择，多屏场景可以根据这个配置显示倒车画面的屏幕。

图层还需要再HWC设置预留对应的图层。

2.8.1.2 RK356x/RK3368

kernel4.19可以直接在驱动中设置：

图层设置，rockchip_drm_drv.c中设置，下面示例为配置Esmart0-win0图层。

```
    drm_for_each_plane(plane, drm_dev) {

        DRM_DBG("plane id: %d, plane name:%s, type=%d\n", plane->base.id, plane-
>name, plane->type);
        if (/*plane->type == DRM_PLANE_TYPE_PRIMARY &&
            (plane->possible_crtcs & drm_crtc_mask(crtc)) && */
            !strncmp(plane->name, "Esmart0-win0", DRM_PROP_NAME_LEN)) {
            break;
        }
    }
```

crtc设置，参考如下修改：

```
diff --git a/kernel-4.19/drivers/gpu/drm/rockchip/rockchip_drm_drv.c b/kernel-
4.19/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
index 6e19ded..5c47b86 100755
--- a/kernel-4.19/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
+++ b/kernel-4.19/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
@@ -1379,7 +1379,7 @@ void drm_direct_show_image(int dmabuf_fd, int width, int
height, int format, uin

        drm_modeset_lock_all(drm_dev);
        drm_for_each_crtc(crtc, drm_dev) {
-               if (crtc->state &&  crtc->state->active) {
+               if (crtc->state &&  crtc->state->active && (!strncmp(crtc->name,
"video_port1", DRM_PROP_NAME_LEN))) {
                        crtc_active = true;
                        break;
                }
```

2.8.2 HWC预留图层

2.8.2.1 RK3588/RK356x/RK3562

HWC预留Esmart0-win0图层作为快速倒车使用，最新的代码已经支持预留图层，/device/rockchip/rk3588/device.mk配置如下属性即可：

```
diff --git a/device.mk b/device.mk
index 01c115b..ebdc695 100644
--- a/device.mk
+++ b/device.mk
@@ -10,7 +10,8 @@ PRODUCT_PACKAGES += \

 # Disable partial updates
 PRODUCT_PROPERTY_OVERRIDES += \
-    debug.hwui.use_partial_updates=false
+    debug.hwui.use_partial_updates=false \
+    vendor.hwc.reserved_plane_name=Esmart0-win0
```

临时调试的时候，可以先将设备上的/vendor/build.prop文件使用adb pull上来，在其中添加属性：

```
vendor.hwc.reserved_plane_name=Esmart0-win0
```

再推到设备生效即可。

2.8.2.2 rk3368

默认SDK已经支持预留图层，HWC设置属性即可。

HWC通过设置属性hwc.win1.reserved设置预留的图层。

hardware/rockchip/hwcomposer/drmhwc/修改如下：

```
diff --git a/Android.mk b/Android.mk
index 72f3426..2468e6a 100755
--- a/Android.mk
+++ b/Android.mk
@@ -504,7 +504,7 @@ LOCAL_CPPFLAGS += -DDUAL_VIEW_MODE=0
 endif

 #USE_PLANE_RESERVED enable
-#LOCAL_CPPFLAGS += -DUSE_PLANE_RESERVED
+LOCAL_CPPFLAGS += -DUSE_PLANE_RESERVED

 # RK_RGA_PREPARE_ASYNC and RK_RGA_COMPSITE_SYNC are exclusive.
 #  RK_RGA_PREPARE_ASYNC: use async rga in hwc_prepare.
diff --git a/drmdisplaycompositor.cpp b/drmdisplaycompositor.cpp
index b09377b..5d348c4 100755
--- a/drmdisplaycompositor.cpp
+++ b/drmdisplaycompositor.cpp
@@ -856,7 +856,7 @@ int DrmDisplayCompositor::DisablePlanes(DrmDisplayComposition
*display_comp) {
```

```
   int ret;
 #ifdef USE_PLANE_RESERVED
-  int win1_reserved = hwc_get_int_property( PROPERTY_TYPE ".hwc.win1.reserved",
"0");
+  int win1_reserved = hwc_get_int_property( PROPERTY_TYPE ".hwc.win1.reserved",
"1");
 #endif
   std::vector<DrmCompositionPlane> &comp_planes =
       display_comp->composition_planes();
@@ -1186,7 +1186,7 @@ int DrmDisplayCompositor::CommitFrame(DrmDisplayComposition
*display_comp,
     //Find out the fb target for clone layer.
     int fb_target_fb_id = -1;
 #ifdef USE_PLANE_RESERVED
-    int win1_reserved = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.reserved", "0");
+    int win1_reserved = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.reserved", "1");
 #endif

 #if RK_3D_VIDEO
diff --git a/hwc_rockchip.cpp b/hwc_rockchip.cpp
index d35cb1c..a963b1a 100755
--- a/hwc_rockchip.cpp
+++ b/hwc_rockchip.cpp
@@ -1892,8 +1892,8 @@ bool MatchPlanes(
     bool bMatch = false;

 #ifdef USE_PLANE_RESERVED
-        uint64_t win1_reserved = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.reserved", "0");
-        uint64_t win1_zpos = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.zpos", "0");
+        uint64_t win1_reserved = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.reserved", "1");
+        uint64_t win1_zpos = hwc_get_int_property( PROPERTY_TYPE
".hwc.win1.zpos", "4");
 #endif

diff --git a/hwcomposer.cpp b/hwcomposer.cpp
index aa0af3d..66f05ce 100755
--- a/hwcomposer.cpp
+++ b/hwcomposer.cpp
@@ -2536,7 +2536,7 @@ static int hwc_prepare(hwc_composer_device_1_t *dev, size_t
num_displays,
   int ret = -1;

 #ifdef USE_PLANE_RESERVED
-  int win1_reserved = hwc_get_int_property( PROPERTY_TYPE ".hwc.win1.reserved",
"0");
+  int win1_reserved = hwc_get_int_property( PROPERTY_TYPE ".hwc.win1.reserved",
"1");
 #endif

 #ifdef USE_HWC2
```

调试的时候可以单独使用mm编译HWC，将编译生成的以下库push到设备即可：

```
adb push
SDK/out/target/product/rk3368_Android12/vendor/lib/hw/hwcomposer.rk30board.so
/vendor/lib/hw
adb push
SDK/out/target/product/rk3368_Android12/vendor/lib64/hw/hwcomposer.rk30board.so
/vendor/lib64/hw
```

2.9 预留CMA内存

快速倒车需要预留CMA内存，用于DRM接口分配buf，可根据分辨率按需分配内存大小，配置参考如下：

```
    reserved-memory {
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        drm_vehicle: drm-vehicle@0{
            compatible = "shared-dma-pool";
            inactive;
            reusable;
            reg = <0x0 (512 * 0x100000) 0x0 (256 * 0x100000)>;//512M ~ 512M+256M
            linux,cma-default;
        };
    };
```

DRM引用：

```
&display_subsystem {
    memory-region = <&drm_logo>, <&drm_vehicle>;
    memory-region-names = "drm-logo", "drm-vehicle";
};
```

3. 调试说明

3.1 打开log开关

3.1.1 打开倒车相关log

drivers/video/rockchip/vehicle/vehicle_cfg.h 中
根据static int debug = 0x1f;调整 debug的LOG输出，分别如下：

```
#define VEHICLE_DG1(format, ...) dprintk(0x1, format, ## __VA_ARGS__)
//ad 转换芯片debug信息输出，对应vehicle_ad_xxxx.c/vehicle_generic_sensor.c文件
#define VEHICLE_DG2(format, ...) dprintk(0x2, format, ## __VA_ARGS__)
//cif 控制器debug信息输出，对应vehicle_cif.c文件
#define VEHICLE_DG3(format, ...) dprintk(0x4, format, ## __VA_ARGS__)
//flinger debug信息输出，对应vehicle_flinger_I2O1.c文件
#define VEHICLE_DG4(format, ...) dprintk(0x8, format, ## __VA_ARGS__)
//vehicle debug信息输出，对应vehicle_main.c文件
#define VEHICLE_DG5(format, ...) dprintk(0x10, format, ## __VA_ARGS__)
//vehicle gpio debug信息输出，vehicle_gpio.c文件
```

3.1.2 打开DRM相关log

3.1.2.1 rk3588/rk3562

rk3588 kernel5.10将drm显示部分独立实现，打开log开关如下：

```
diff --git a/drivers/gpu/drm/rockchip/rockchip_drm_direct_show.c
b/drivers/gpu/drm/rockchip/rockchip_drm_direct_show.c
old mode 100644
new mode 100755
index a4799985add0..1bdaddbb4cf8
--- a/drivers/gpu/drm/rockchip/rockchip_drm_direct_show.c
+++ b/drivers/gpu/drm/rockchip/rockchip_drm_direct_show.c
@@ -10,7 +10,7 @@
 #include "../drm_internal.h"
 #include "rockchip_drm_direct_show.h"

-static int drm_ds_debug;
+static int drm_ds_debug = 1;
 #define DRM_DS_DBG(format, ...) do {    \
        if (drm_ds_debug)       \
                pr_info("DRM_DS: %s(%d): " format, __func__, __LINE__, ##
__VA_ARGS__); \
```

3.1.2.2 rk356x/rk3368

```
diff --git a/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
b/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
old mode 100644
new mode 100755
index 1125a4c..90656df
--- a/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
+++ b/drivers/gpu/drm/rockchip/rockchip_drm_drv.c
@@ -57,7 +57,7 @@
#define DRIVER_MINOR 0
#define DRIVER_PATCH 0
-static int debug = 0;
+static int debug = 1;
#define DRM_DBG(format, ...) do { \
if (debug) \
printk(KERN_INFO "%s %s(%d): " format,DRIVER_DESC,__func__,__LINE__,##
```

```
__VA_ARGS__); } while (0)
```

3.1.3 打开rkcif控制器log

rk3588/rk356x/rk3562支持在退出vehicle倒车后，重新注册v4l2 camera框架，通过如下命令打开 debug 开关：

```
echo 1 > /sys/module/video_rkcif/parameters/debug
echo 1 > /sys/module/video_rkcif/parameters/debug_csi2
```

或者修改 SDK/kernel/drivers/media/platform/rockchip/cif/dev.c 中 int rkcif_debug = 1;

3.2 常用调试命令

3.2.1 查看控制器的寄存器

- 打开io命令配置，在配置文件rockchip_defconfig中添加：

```
CONFIG_DEVMEM=y
```

- rk3588支持mipi和dvp接口的输入，对应查看vicap、csi-host、phy的寄存器命令如下：

```
//vicap
io -4 -l 0x1000 0xfdce0000
//mipi-csi host
io -4 -l 0x100 0xfdd10000
io -4 -l 0x100 0xfdd20000
io -4 -l 0x100 0xfdd30000
io -4 -l 0x100 0xfdd40000
io -4 -l 0x100 0xfdd50000
io -4 -l 0x100 0xfdd60000
//csi-dphy
io -4 -l 0x100 0xfeda0000
io -4 -l 0x100 0xfedb0000
io -4 -l 0x100 0xfedc0000
io -4 -l 0x100 0xfedc8000
```

- rk356x查看vicap和csi-host寄存器

```
//vicap
io -4 -l 0x200 0xfdfe0000
//mipi-csi host
io -4 -l 0x100 0xfdfb0000
```

- rk3368查看cif控制器命令如下：

```
io -4 -l 0x100 0xff950000
```

- rk3562对应vicap寄存器和csi寄存器

```
// vicap
io -4 -l 0x1000 0xff3e0000
//csi host
io-4 -l 0x100 0xff380000    //mipi0
io-4 -l 0x100 0xff390000
io-4 -l 0x100 0xff3a0000
io-4 -l 0x100 0xff3b0000
```

3.2.2 查看中断

查看vehicle-cif中断：

```
cat /proc/interrupts | grep vehicle_cif
```

3.2.3 保存hwc log

抓取HWC相关log：

```
adb root && adb shell logcat -G 128M
adb shell "setprop vendor.hwc.log debug"
adb shell logcat -c
adb shell logcat > hwc.log
```

3.2.4 查看图层状态

查看图层状态命令：

```
cat /d/dri/0/summary
```

3.3 快速倒车与v4l2框架切换

3.3.1 调试命令

```
echo 88 > /dev/vehicle
```

可以退出快速倒车系统，切换到v4l2框架，重新注册v4l2框架的驱动链路。

3.3.2 驱动对应配置

需要实现快速倒车和v4l2框架的切换，需要驱动实现相应的接口，可参考nvp6188.

media/i2c/nvp6188.c：使用宏定义CONFIG_VIDEO_REVERSE_IMAGE区分sensor注册的入口

```
int nvp6188_sensor_mod_init(void)
{
    return i2c_add_driver(&nvp6188_i2c_driver);
}

#ifndef CONFIG_VIDEO_REVERSE_IMAGE
device_initcall_sync(nvp6188_sensor_mod_init);
#endif
```

vehicle实现相应的接口：

vehicle_generic_sensor.c文件，需要实现如下接口：

```
.sensor_mod_init = nvp6188_sensor_mod_init
```

config文件需要将sensor的都打开：

```
CONFIG_VIDEO_NVP6188=y
CONFIG_VIDEO_REVERSE_NVP6188=y
```

4. 常见问题与解决

4.1 支持摄像头接口类型？

4.1.1 rk3588/rk356x

rk3588和rk356x快速倒车影像系统，支持输出yuv的MIPI摄像头或者转换芯片，如gc2145 mipi 摄像头、nvp6324转接芯片等；

MIPI RAW sensor需要调试3A效果，暂时不支持应用于快速倒车影像。

rk3588和rk356x支持dvp接口的YUV摄像头或者AD转换芯片，如gc2145 dvp摄像头、nvp6158转接芯片等等

4.1.2 rk3368

支持dvp接口sensor，输入为bt601/bt656信号，如gc2145 dvp摄像头、tp2825/ad7181等AD芯片。

4.1.3 rk3562

rk3562支持MIPI YUV摄像头用于快速倒车系统。

4.2 调试过程中如何抓图？

在文件drivers/video/rockchip/vehicle/vehicle_flinger.c中设置：

```
diff --git a/kernel-5.10/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
b/kernel-5.10/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
old mode 100644
new mode 100755
index 81b41eb..016470f
--- a/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
+++ b/drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c
@@ -46,9 +46,9 @@
 #include "vehicle_flinger.h"
 #include "../../../gpu/drm/rockchip/rockchip_drm_direct_show.h"

-static int vehicle_dump_cif = 0;
-static int vehicle_dump_rga = 0;
-static int vehicle_dump_vop = 0;
+static int vehicle_dump_cif = 1;
+static int vehicle_dump_rga = 1;
+static int vehicle_dump_vop = 1;

 enum force_value {
        FORCE_WIDTH = 1920,
```

打开该设置后，会保存cif输出、rga输入输出、vop输出的图像，将图像保存在：

```
/data/rga_scaler_in_width_height.yuv
/data/rga_scaler_out_width_height.yuv
/data/rga_render_width_height.yuv
/data/vop_show_width_height.yuv
/data/cif_out_width_height.yuv
```

4.3 倒车开关无效？

首先查看dts文件是否有对快速倒车的gpio口的描述，如果没有就添加，如果有，则查看gpio口是否冲突：分析log查看倒车GPIO口，然后从代码中查询该口用到的地方，从而分析该口是否被占用，以及占用的是哪个硬件，是否有用，从而考虑更改io口解决问题。

4.4 如何添加新的camera或者转换芯片的支持？

参考现有的sensor，如tp2825、gc2145、nvp6324等进行实现，主要有以下几点：

1. dts添加相关sensor的配置
2. kernel/drivers/video/rockchip/vehicle/vehicle_generic_sensor.c文件中的接口调用进行修改与实现

```
107         {
108                 .name = "nvp6188",
109 #ifdef CONFIG_VIDEO_REVERSE_NVP6188
110                 .sensor_init = nvp6188_ad_init,
111                 .sensor_deinit = nvp6188_ad_deinit,
112                 .sensor_stream = nvp6188_stream,
113                 .sensor_get_cfg = nvp6188_ad_get_cfg,
114                 .sensor_check_cif_error = nvp6188_ad_check_cif_error,
115                 .sensor_check_id_cb = nvp6188_check_id,
116                 .sensor_set_channel = nvp6188_channel_set,
117 #ifdef CONFIG_VIDEO_NVP6188
118                 .sensor_mod_init = nvp6188_sensor_mod_init
119 #endif
120 #endif
121         }
122 };
123
```

3. 对ad初始化等函数进行实现，参考如下两个文件，并在Makefile文件中进行配置。

```
-rw-rw-r-- 1 fjw fjw   1801 Mar  4 21:05 vehicle_ad.h
-rw-rw-r-- 1 fjw fjw  16824 Mar  7 11:28 vehicle_ad_7181.c
-rw-rw-r-- 1 fjw fjw    437 Mar  4 21:05 vehicle_ad_7181.h
-rw-rw-r-- 1 fjw fjw  24401 Mar  4 21:05 vehicle_ad_gc2145.c
-rw-rw-r-- 1 fjw fjw    435 Mar  4 21:05 vehicle_ad_gc2145.h
-rw-rw-r-- 1 fjw fjw  30240 Mar  4 21:05 vehicle_ad_nvp6324.c
-rw-rw-r-- 1 fjw fjw    444 Mar  4 21:05 vehicle_ad_nvp6324.h
-rw-rw-r-- 1 fjw fjw  22264 Mar  4 21:05 vehicle_ad_tp2825.c
-rw-rw-r-- 1 fjw fjw    435 Mar  4 21:05 vehicle_ad_tp2825.h
```

4. makefile添加对应的配置：

video/rockchip/vehicle目录：

Makefile文件的修改：

```
video_rkvehicle-$(CONFIG_VIDEO_REVERSE_NVP6188) += \
            vehicle_ad_nvp6188.o
```

kconfig添加：

```
config VIDEO_REVERSE_NVP6188
    bool "nvp6188 for reverse sensor"
    help
      Say y if use nvp6188.
```

并在config文件打开对应的配置。

5. 如果需要与正常的v4l2 camera框架代码进行切换，例如开机之后正常进入快速倒车，之后执行echo 88 > /dev/vehicle，退出快速倒车之后，调用nvp6188_sensor_mod_init切换到v4l2的camera框架，需要实现nvp6188_sensor_mod_init接口，v4l2框架的驱动也要做相应的修改：

```
2735
2736  static struct i2c_driver nvp6188_i2c_driver = {
2737          .driver = {
2738                  .name = NVP6188_NAME,
2739                  .pm = &nvp6188_pm_ops,
2740                  .of_match_table = of_match_ptr(nvp6188_of_match),
2741          },
2742          .probe          = &nvp6188_probe,
2743          .remove         = &nvp6188_remove,
2744          .id_table       = nvp6188_match_id,
2745  };
2746
2747  int nvp6188_sensor_mod_init(void)
2748  {
2749          return i2c_add_driver(&nvp6188_i2c_driver);
2750  }
2751
2752  #ifndef CONFIG_VIDEO_REVERSE_IMAGE
2753  device_initcall_sync(nvp6188_sensor_mod_init);
2754  #endif
2755
2756  static void __exit sensor_mod_exit(void)
2757  {
2758          i2c_del_driver(&nvp6188_i2c_driver);
2759  }
```

注意：如果需要添加新的camera或者转接芯片的支持，建议现在正常的v4l2 camera的框架下点亮sensor，再行移植到快速倒车的框架下。

4.5 快速倒车不出图

快速倒车正常注册，打到倒车脚，没有出图。排查如下：

- 1.确认sensor是否有数据输出

可以直接使用示波器测量sensor是否有输出，如果没有输出，查看log是否是sensor的i2c通讯异常，如果i2c通讯正常，则可能是对应的配置序列的问题。

建议在调试快速倒车之前，将sensor正常跑V4l2 camera框架的先调试起来，这样就基本可以避免sensor的问题。

- 2.查看cif控制器是否收到数据

可以直接查看cif控制器是否有中断：

```
cat /proc/interrupt | grep vehicle
```

如果是mipi接口的sensor，可以查看对应的dphy的stopstate是否正常，连续查询对应的寄存器，查看对应的data lane和clk lane的状态位是否变化，具体的寄存器地址可以查阅对应芯片平台的TRM手册获取。这里以rk3588为例，首先查看对应CSI-HOST的基地址（RK3588共有6个CSI-HOST），如下所示。

CSI HOST0	FDD10000	64KB
CSI HOST1	FDD20000	64KB
CSI HOST2	FDD30000	64KB
CSI HOST3	FDD40000	64KB
CSI HOST4	FDD50000	64KB
CSI HOST5	FDD60000	64KB

查看对应的CSI-HOST的phy-stopstate的偏移地址，可知对应的偏移地址为0x0014

1.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CSI2HOST_VERSION	0x0000	W	0x00000000	Controller version identification
CSI2HOST_N_LANES	0x0004	W	0x00000000	Number of active data lanes
CSI2HOST_CSI2_RESETN	0x0010	W	0x00000000	CSI2 controller reset
CSI2HOST_PHY_STATE	0x0014	W	0x00000000	General settings for all blocks
CSI2HOST_DATA_IDS_1	0x0018	W	0x00000000	Data IDS for which IDI reports line boundary matching errors
CSI2HOST_DATA_IDS_2	0x001C	W	0x00000000	Data IDS for which IDI reports line boundary matching errors
CSI2HOST_ERR1	0x0020	W	0x00000000	Error state register 1
CSI2HOST_ERR2	0x0024	W	0x00000000	Error state register 2
CSI2HOST_MSK1	0x0028	W	0x00000000	Masks for errors 1
CSI2HOST_MSK2	0x002C	W	0x00000000	Masks for errors 2
CSI2HOST_CONTROL	0x0040	W	0x0C204000	Control

state状态寄存器码，描述如下：

CSI2HOST_PHY_STATE

Address: Operational Base + offset (0x0014)

Bit	Attr	Reset Value	Description
31:12	RO	0x00000	reserved
11	RW	0x0	bypass_2ecc_tst Payload Bypass test mode for double ECC errors.
10	RO	0x0	phy_stopstateclk Clock lane in Stop state.
9	RO	0x0	phy_rxulpsclknot This signal indicates that the clock lane module has entered the Ultra Low Power state. Active low.
8	RO	0x0	phy_rxclkactivehs Indicates that the clock lane is actively receiving a DDR clock.
7	RO	0x0	phy_stopstatedata_3 Data lane 3 in Stop state.
6	RO	0x0	phy_stopstatedata_2 Data lane 2 in Stop state.
5	RO	0x0	phy_stopstatedata_1 Data lane 1 in Stop state.
4	RO	0x0	phy_stopstatedata_0 Data lane 0 in Stop state.
3	RO	0x0	phy_rxulpsesc_3 Lane module0 has entered the Ultra Low Power mode.
2	RO	0x0	phy_rxulpsesc_2 Lane module2 has entered the Ultra Low Power mode.
1	RO	0x0	phy_rxulpsesc_1 Lane module1 has entered the Ultra Low Power mode.

根据基地址+偏移地址就可以查询对应的phy-state状态，并根据状态是否变化来判断mipi-phy是否有接收到数据。

```
io -4 0xfdd10014    //mipi0
io -4 0xfdd20014    //mipi1
io -4 0xfdd30014    //mipi2
io -4 0xfdd40014    //mipi3
io -4 0xfdd50014    //mipi4
io -4 0xfdd60014    //mipi5
```

- 3.根据log的其他报错或者添加log来定位问题。

4.6 drm内存不够报错

分配内存的时候出现类似如下报错，导致内存分配失败的，是预留给drm的CMA的内存不够导致的，适当将预留的CMA内存增加，即可解决。

```
3.736468][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1158): drm_direct_alloc_dma_buf, 1
3.740562][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1166): drm_direct_alloc_dma_buf, 2
3.741378][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1177): drm_direct_alloc_dma_buf, 3
3.742173][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1183): drm_direct_alloc_dma_buf, 4
3.742949][  T113] drm_direct_alloc_dma_buf, 5, fd:0, phy_addr:0x4ff900000, vir_addr:0x0000000021a2644c; w:1920,h:1080
3.743899][  T113] Vehicle vehicle_flinger_I2O1(342): -----creat buffer over
3.745176][  T113] Vehicle vehicle_flinger_I2O1(323): ----------------alloc buffer start---------
3.745977][  T113] Vehicle vehicle_flinger_I2O1(331): -----------bpp over s = 1920 h = 1080 bpp = 2
3.746789][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1158): drm_direct_alloc_dma_buf, 1
3.747893][  T113] cma: cma_alloc: cma: alloc failed, req-size: 1013 pages, ret: -12
3.750903][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1166): drm_direct_alloc_dma_buf, 2
3.751725][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1177): drm_direct_alloc_dma_buf, 3
3.752511][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1183): drm_direct_alloc_dma_buf, 4
3.753292][  T113] drm_direct_alloc_dma_buf, 5, fd:1, phy_addr:0x405000000, vir_addr:0x00000000f28377fb; w:1920,h:1080
3.754243][  T113] Vehicle vehicle_flinger_I2O1(342): -----creat buffer over
3.756170][  T113] Vehicle vehicle_flinger_I2O1(323): ----------------alloc buffer start---------
3.756957][  T113] Vehicle vehicle_flinger_I2O1(331): -----------bpp over s = 1920 h = 1080 bpp = 2
3.757768][  T113] RockChip Soc DRM drm_direct_alloc_dma_buf(1158): drm_direct_alloc_dma_buf, 1
3.758960][  T113] cma: cma_alloc: cma: alloc failed, req-size: 1013 pages, ret: -12
```

解决办法：dts配置预留更大的内存，修改如下配置预留的内存即可：

```
drm_vehicle: drm-vehicle@20000000{
        compatible = "shared-dma-pool";
        inactive;
        reusable;
        reg = <0x0 (512 * 0x100000) 0x0 (256 * 0x100000)>;//512M ~ 512M+256M
        linux,cma-default;
    };
```

4.7 快速倒车与安卓页面闪烁

出现这个问题，是上层HWC没有预留图层导致的，确认HWC有预留图层即可。

设置了预留图层，但是还是没有生效，确保正确预留图层且预留的图层与内核提交的一致。

1.调试常用方法：

a.看plane显示状态

```
 cat /d/dri/0/summary
```

b. 查看log

```
setprop sys.hwc.log 511 ;logcat -c ;logcat | grep hwc >hwc.log
```

确认是否有plane预留成功的log：

```
hwcomposer-drm: Enable **USE_PLANE_RESERVED,** plane **share_id** = 57
```

这个可以通过log看到预留成功的plane id。相关代码在

```
hardware/rockchip/hwcomposer/hwcomposer.cpp
```

c. 确认快速倒车送显的plane id与预留的id一致。

rk3588在drivers/video/rockchip/vehicle/vehicle_flinger_I2O1.c修改rk_flinger_vop_show函数：

```
flinger->plane = rockchip_drm_direct_show_get_plane(flinger->drm_dev, "Esmart0-
win0");
if (flinger->plane == NULL) {
    printk("%s: error: failed to get plane\n", __func__);
    return -EINVAL;
}
```

rk356x和rk3368在kernel/drivers/gpu/drm/rockchip/rockchip_drm_drv.c文件中修改。

drm_direct_show_image函数中：如下部分的break前加log，把plane->base.id这个值打出来，看下要送显的id是否与预留的id一致，不一致的话可以考虑把如下的if判断条件中改为plane->base.id == 预留的id。参考如下修改：

```
drm_for_each_plane(plane, drm_dev) {
    DRM_DBG("plane id: %d, plane name:%s, type=%d\n", plane->base.id, plane-
>name, plane->type);
    if (!strncmp(plane->name, "plane-2", DRM_PROP_NAME_LEN)) //plane-2 对应win1-0
    {
        break;
    }
}
```

4.8 如何进入快速倒车画面

1. dts配置倒车脚gpio，开机后，拉高对应的gpio即可进入

```
car-reverse {
    car-reverse-gpios = <&gpio1 RK_PB0 GPIO_ACTIVE_HIGH>;
    linux,debounce-ms = <5>;
    label = "car-reverse";
    gpio,wakeup;
};
```

2. 没有中断脚，开机默认进入快速倒车画面

为了调试使用，驱动可以设置TEST_GPIO，可以让系统开机默认进入倒车画面

```
diff --git a/drivers/video/rockchip/vehicle/vehicle_main.c
b/drivers/video/rockchip/vehicle/vehicle_main.c
old mode 100644
new mode 100755
index cf7d686..26931dd
--- a/kernel-5.10/drivers/video/rockchip/vehicle/vehicle_main.c
+++ b/kernel-5.10/drivers/video/rockchip/vehicle/vehicle_main.c
@@ -219,7 +219,7 @@ static int vehicle_state_change(struct vehicle *v)
        flinger_inited = true;

        gpio_reverse_on = vehicle_gpio_reverse_check(gpiod);
-       gpio_reverse_on = TEST_GPIO & gpio_reverse_on;
+       gpio_reverse_on = TEST_GPIO;
        VEHICLE_DG4("%s, gpio = reverse %s, ad width = %d,ad_sensor_ready = %d,
state=%d dvr_apk_need_start = %d\n",
```

```
                        __func__, gpio_reverse_on ? "on" : "over",
                        v_cfg->width,v_cfg->ad_ready, v->state,
dvr_apk_need_start);
```

3. 没有设置中断脚，系统起来后进入快速倒车

直接往vehicle节点写入属性值，可设置进入倒车画面和退出画面

进入倒车影像：

```
echo 11 > /dev/vehicle
```

关闭倒车影像：

```
echo 10 > /dev/vehicle
```

4. 退出倒车

往vehicle节点写入对应的值，可以退出快速倒车系统，执行如下命令退出倒车后，将释放cif的资源，将不能再次进入倒车影像画面

```
echo 88 > /dev/vehicle
```

4.9 相关log分析

1.倒车驱动正常注册log

```
[    1.645555][    T1] vehicle vehicle: driver version: 00.02.0e
[    1.645641][    T1] vehicle_generic_sensor(292): Get nvp6324 rst_active
failed!
[    1.645668][    T1] vehicle vehicle_main: vehicle driver probe success
[    1.645740][  T114] Vehicle vehicle_main(483):(483) g_vehicle((einval))!
[    1.645831][  T114] vehicle vehicle_main: vehicle_gpio_init: request irq
gpio(40)
[    1.645947][  T114] vehicle vehicle_main: vehicle_gpio_init_check: gpiod-
>atv_val(1), gpiod->val(1)
[    1.645959][  T114] Vehicle vehicle_main(164):vehicle_gpio_stat_change_notify
enter!
[    1.652240][  T114] vehicle_generic_sensor(375): vehicle_ad_init(375)
ad_name:nvp6324!
[    2.332668][  T114] vehicle vehicle_main: use mipi dcphy, no need request rst
[    2.332685][  T114] vehicle vehicle_main: clk get pclk
[    2.332710][  T114] vehicle vehicle_main: clk get pclk_csi2host
[    2.332719][  T114] vehicle vehicle_main: clk get iclk_csi2host
[    2.412892][  T114] vehicle vehicle_main: vehicle_flinger_parse_dt: get crtc
name from dts, crtc-name = video_port3
[    2.412914][  T114] vehicle vehicle_main: vehicle_flinger_parse_dt: get crtc
name from dts, crtc-name = Esmart0-win0
```

2.等待drm设备报错

在快速倒车过程中可能会出现如下报错的log，该log并非真的报错log，而是快速倒车在等待drm设备注册成功的log，只有等待drm注册成功后才会继续流程。

```
[    3.923492][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    3.940157][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    3.956824][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    3.973491][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    3.990157][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.006825][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.023491][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.040157][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.056824][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.073491][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.090157][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
[    4.106824][  T114] vehicle_flinger_init: ------drm device is not ready!!!----
-
```

3.找不到crtc，提示这个报错可能是dts配置的crtc错误，注意检查。也有可能是该crtc对应的显示屏幕被关闭。

```
[  123.478773][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
[  123.479577][    T7] rk_flinger_vop_show: error: failed to get crtc
[  123.518649][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
[  123.519445][    T7] rk_flinger_vop_show: error: failed to get crtc
[  123.538423][    T1] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_disable]
Crtc atomic disable vp3
[  123.564958][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
[  123.565752][    T7] rk_flinger_vop_show: error: failed to get crtc
[  123.598755][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
[  123.599549][    T7] rk_flinger_vop_show: error: failed to get crtc
[  123.638744][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
[  123.639539][    T7] rk_flinger_vop_show: error: failed to get crtc
[  123.678812][    T7] ERR: DRM_DS: rockchip_drm_direct_show_get_crtc(201):
failed to find active crtc
```