Rockchip RK628 For All Porting Guide

文件标识：RK-YH-YF-287

发布版本：V2.0

日期：2022-04-10

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

RK628输入支持RGB/HDMI/BT1120，输出支持DSI/LVDS/GVI/CSI/BT1128/RGB/HDMI等，具体功能描述参考datasheet。本文档主要描述 rk628-for-all代码软件配置以及调试手段，以及常见问题处理方式， rk628-for-all代码期望做到与硬件平台和软件版本无关，还在不断完善中。目前代码包括MISC（用于代码路径drivers/misc/rk628/，支持DSI/LVDS/GVI等）和MEDIA（代码路径drivers/media/i2c/rk628/，支持CSI/CSI-V4L2/BT1120）两大部分，分别进行维护。本文档按照不同的模块分别进行介绍和说明。

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	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-08-28	初始发布
V1.1	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-09-03	新增DTS DEMO
V1.2	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-09-17	补充FAQ部分
V1.3	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-09-29	补充AUDIO的FAQ
V1.4	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-10-08	MISC新增HDMIOUT说明
V1.5	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠	2021-10-27	增加DSI->CSI方案说明和pinctrl支持
V1.6	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠/黄雄山	2021-11-02	增加硬件基础信息
V1.7	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠/黄雄山	2021-11-26	支持根据输入源配置src_mode参数自动生成hdmirx的edid数据
V1.8	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠/黄雄山	2022-01-15	新增部分FAQ及BT1120的使用说明
V1.9	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠/黄雄山	2022-03-05	补充说明HDMIRX支持的频点列表，MISC新增多RK628功能使用说明。
V2.0	黄国椿/陈顺庆/兰顺华/温定贤/罗伟/让凌鹏/郭立楠/黄雄山	2022-04-10	补充声卡注册失败问题处理和HDMI-IN上层问题排查方法。

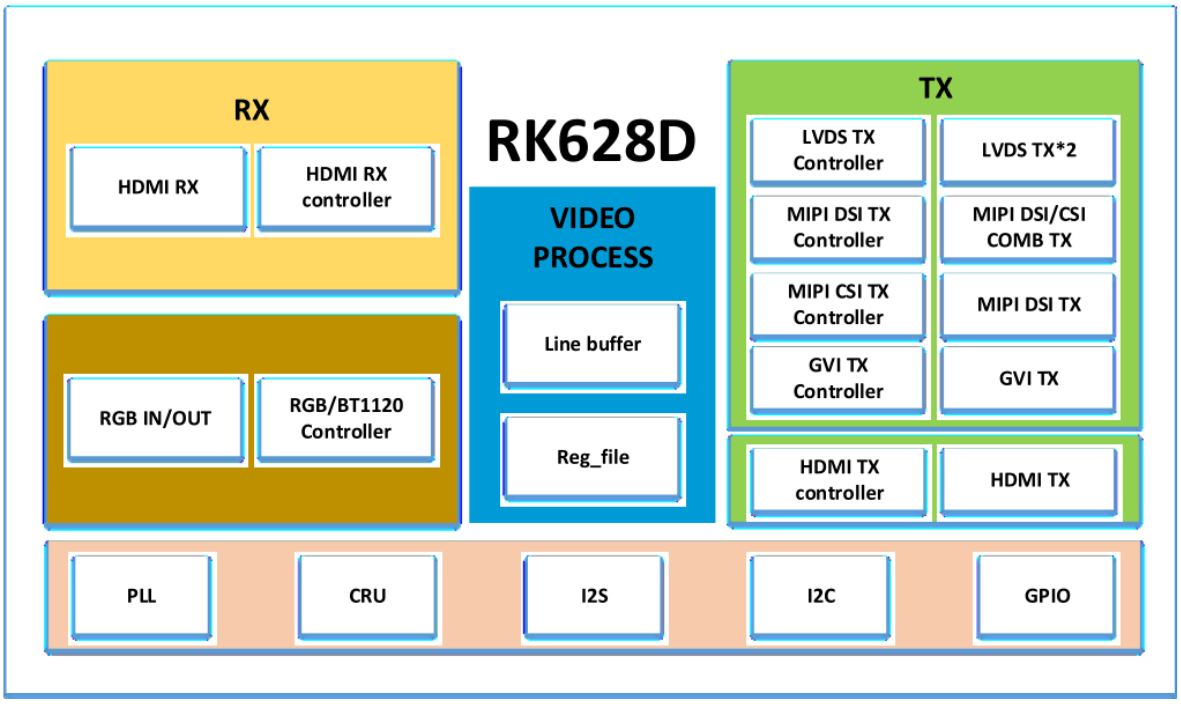
目录

前言

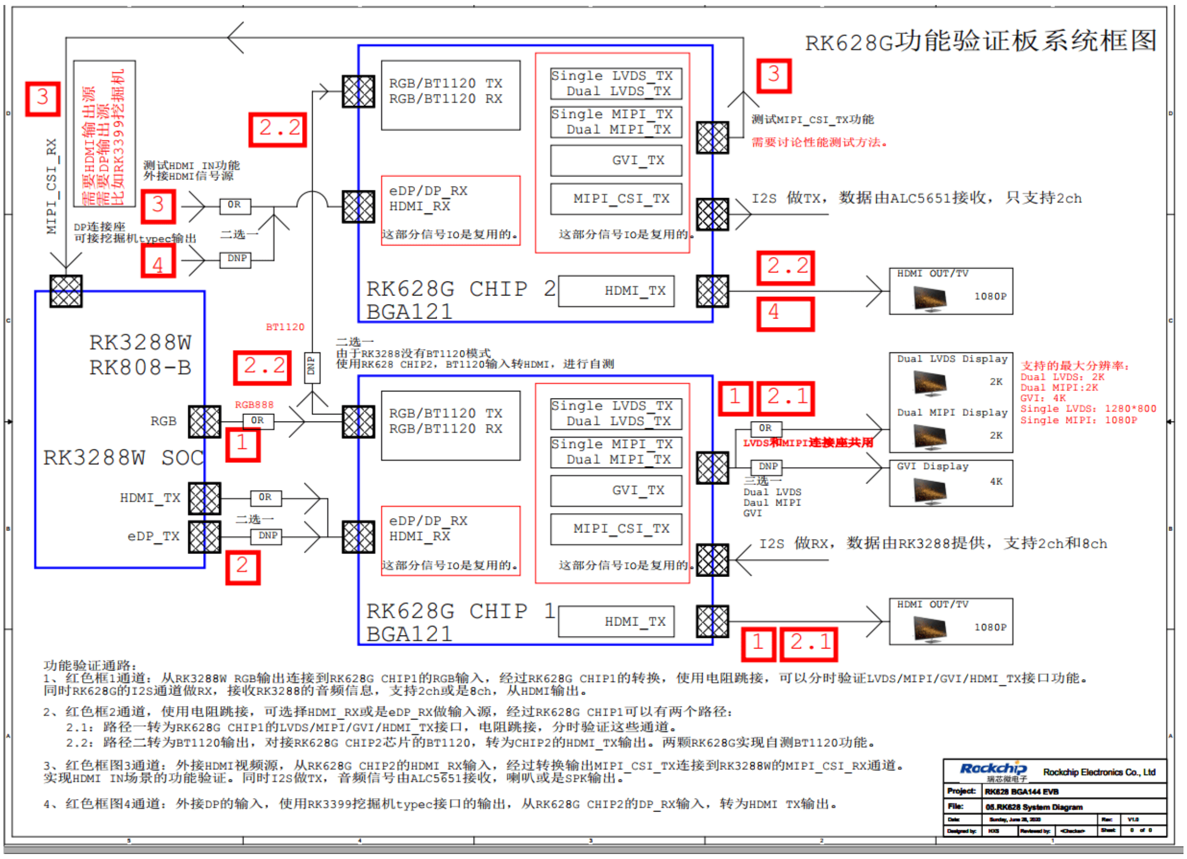
RK628 分为 Display 通路和 HDMI IN 通路，SDK 版本 Display 通路基于DRM框架，HDMI IN 通路基于V4L2框架，不同的框架或是不同的内核版本需要不同的驱动去适配，而且只适用于RK平台。为了适配没的平台，方便驱动移植，推出 For-All 版本驱动。

For-All 版本驱动一样也分为Display 通路和 HDMI IN 通路，Display 通路的驱动于drivers/misc/rk628/下，HDMI IN 通路的驱动于drivers/media/i2c/rk628/下，下面我们Misc和Media分别代表这两套驱动，两套代码相互独立，都可单独编译运行。

RK628芯片框图如下:



RK628典型设计硬件框图如下:



功能验证通路：
1、红色框1通道：从RK3288W RGB输出连接到RK628G CHIP1的RGB输入，经过RK628G CHIP1的转换，使用电阻跳接，可以分时验证LVDS/MIPI/GVI/HDMI_TX接口功能。
同时RK628G的I2S通道做RX，接收RK3288的音频信息，支持2ch或是8ch，从HDMI输出。

2、红色框2通道，使用电阻跳接，可选择HDMI_RX或是eDP_RX做输入源，经过RK628G CHIP1可以有两个路径：
2.1：路径一转为RK628G CHIP1的LVDS/MIPI/GVI/HDMI_TX接口，电阻跳接，分时验证这些通道。
2.2：路径二转为BT1120输出，对接RK628G CHIP2芯片的BT1120，转为CHIP2的HDMI_TX输出。两颗RK628G实现自测BT1120功能。

3、红色框图3通道：外接HDMI视频源，从RK628G CHIP2的HDMI_RX输入，经过转换输出MIPI_CSI_TX连接到RK3288W的MIPI_CSI_RX通道。
实现HDMI IN场景的功能验证。同时I2S做TX，音频信号由ALC5651接收，喇叭或是SPK输出。

4、红色框图4通道：外接DP的输入，使用RK3399挖掘机typec接口的输出，从RK628G CHIP2的DP_RX输入，转为HDMI TX输出。

硬件部分请查阅文档“RK628D硬件设计注意事项_V10_20201130.pdf”或更新版本。

RK628主要规格如下:

Video Iutput Interface	**HDMI/BT1120/RGB**	**Typical Format**
HDMI IN	4K@30	YUV420(4K@60) YUV422/YUV444/RGB888
BT1120 IN	1080P@60	YUV422 8bit
RGB IN	1080P@60	RGB888
VIDEO Output Interface	**HDMI/GVI/MIPI/LVDS/RGB/BT1120**	**Typical Format**
GVI OUT	4K@60	RGB888
MIPI CSI	4K@30	YUV422 8bit
MIPI DSI	1080P@60/2.5K@60	RGB888
LVDS OUT	720P@60/1080P@60	RGB888
BT1120 OUT	1080P@60	YUV422 8bit
RGB OUT	1080P@60	RGB888
HDMI OUT	1080P@60	RGB888/YUV444 8bit

更多的应用注意事项请查阅文档Rockchip_RK628D_Application_Notes_CN_V1.4.pdf或更新版本。

Misc

驱动介绍

驱动目录结构

```
drivers/misc/rk628/
├── Makefile
├── panel.c
├── panel.h
├── rk628.c
├── rk628_combrxphy.c
├── rk628_combrxphy.h
├── rk628_combtxphy.c
├── rk628_combtxphy.h
├── rk628_config.c   //V19版本已删除
├── rk628_config.h   //V19版本已删除
├── rk628_cru.c
├── rk628_cru.h
├── rk628_csi.c
├── rk628_csi.h
├── rk628_dsi.c
├── rk628_dsi.h
├── rk628_gpio.h
├── rk628_grf.h
├── rk628_gvi.c
├── rk628_gvi.h
├── rk628.h
├── rk628_hdmirx.c
├── rk628_hdmirx.h
├── rk628_hdmitx.c
├── rk628_hdmitx.h
├── rk628_lvds.c
├── rk628_lvds.h
├── rk628_pinctrl.c
├── rk628_pinctrl.h
├── rk628_post_process.c
├── rk628_post_process.h
├── rk628_rgb.c
└── rk628_rgb.h
```

配置 rk628d 输入输出的方法

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于RGB2DSI的参考配置方法。

```
drivers/misc/rk628/rk628.h
#define RK628_RGB_IN
#undef RK628_HDMI_IN
#define RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制，关键字是“rk628,rgb-in;”和“rk628-dsi”，举例如下：

```
--- a/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts
+++ b/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts
+               rk628,rgb-in;
+               rk628-dsi {
+                       //rockchip,dual-channel;
+                       dsi,eotp;
+                       dsi,video-mode;
+                       dsi,format = "rgb888";
+                       dsi,lanes   = <4>;
+                       status = "okay";
                }
```

目前输入有HDMI IN和RGB IN, 输出有RGB, LVDS, DSI, CSI, 所以需要配置对应的输入输出，如上配置的是RGB + DSI OUT.

MISC驱动核心DTS配置说明

```
&i2c1 {
    clock-frequency = <400000>;
    status = "okay";

    rk628: rk628@50 {
        compatible = "rockchip,rk628";
        status = "okay";
        reg = <0x50>;
        interrupt-parent = <&gpio7>;
        interrupts = <15 IRQ_TYPE_LEVEL_HIGH>;
        enable-gpios = <&gpio5 RK_PC2 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio7 RK_PB6 GPIO_ACTIVE_LOW>;

        /*panel option*/
        panel-enable-gpios = <&gpio7 RK_PA2 GPIO_ACTIVE_HIGH>;
        panel-backlight = <&backlight>;

        /* soc_24M option, only for gvi */
        /*
        pinctrl-names = "default";
        pinctrl-0 = <&test_clkout>;
        clocks = <&cru SCLK_TESTOUT>;
        clock-names = "soc_24M";
        assigned-clocks = <&cru SCLK_TESTOUT_SRC>;
        assigned-clock-parents = <&xin24m>;
        */
    };
};

/*
&pinctrl {
    test {
        test_clkout: test-clkout {
            rockchip,pins = <0 17 RK_FUNC_1 &pcfg_pull_none>;
        };
    };
};
*/
```

DTS属性说明

Property	Description	Option Value
reg	I2C slave address	0x50/0x51
interrupt-parent/interrupts	interrupt	depend on hardware
enable-gpios	rk628d enable pin	depend on hardware
reset-gpios	rk628d reset pin	depend on hardware
panel-enable-gpios	panel enable pin, option	depend on hardware
panel-backlight	panel backlight, option	depend on hardware
soc_24M	soc 为 gvi tx提供同源的时钟描述, option	depend on hardware

panel 端配置

1. timing配置

V19版本开始在DTS里配置，可忽略这部分内容，而早期版本在 rk628_config.c修改配置。

2. 属性说明

Property	Description
clock	pixel clk in KHz
hdisplay	hactive
hsync_start	hactive+hfp
hsync_end	hactive+hfp+hsync
htotal	hactive+hfp+hsync+hbp
vdisplay	vactive
vsync_start	vactive+vfp
vsync_end	vactive+vfp+vsync
vtotal	vactive+vfp+vsync+vbp
flags	DRM_MODE_FLAG_NVSYNC | DRM_MODE_FLAG_NVSYNC 或 DRM_MODE_FLAG_PVSYNC | DRM_MODE_FLAG_PVSYNC

```
static struct drm_display_mode src_mode = {
    /* .clock = */ 132000,
    /* .hdisplay = */ 1080,
    /* .hsync_start = */ 1080 + 15,
    /* .hsync_end = */ 1080 + 15 + 2,
    /* .htotal = */ 1080 + 15 + 2 + 30,
    /* .vdisplay = */ 1920,
    /* .vsync_start = */ 1920 + 15,
    /* .vsync_end = */ 1920 + 15 + 2,
    /* .vtotal = */ 1920 + 15 + 2 + 15,
    /* .flags = */ DRM_MODE_FLAG_NVSYNC | DRM_MODE_FLAG_NVSYNC,
};

static struct drm_display_mode dst_mode = {
    /* .clock = */ 132000,
    /* .hdisplay = */ 1080,
    /* .hsync_start = */ 1080 + 15,
    /* .hsync_end = */ 1080 + 15 + 2,
    /* .htotal = */ 1080 + 15 + 2 + 30,
    /* .vdisplay = */ 1920,
    /* .vsync_start = */ 1920 + 15,
    /* .vsync_end = */ 1920 + 15 + 2,
    /* .vtotal = */ 1920 + 15 + 2 + 15,
    /* .flags = */ DRM_MODE_FLAG_NHSYNC | DRM_MODE_FLAG_NVSYNC,
};
```

如果有用到 scaler，将 scaler source timing 配置到 src_mode，将 scaler out timing 配置到 dst_mode。如果没有用到 scaler，则 src_mode 和 dst_mode 配置相同的目标 timing。

如果用到两个同分辨率的 DSI 单屏或两个通分辨率的 LVDS 单屏，需要对 src_mode 和 dst_mode 中的 clock、hdisplay、hsync_start、hsync_end 和 htotal 在原有单屏配置基础上乘以2。

参考DTS

目前已支持以下DTS DEMO，其他的还在逐步添加中。

```
arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts
arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2gvi-avb.dts
arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2lvds-avb.dts
arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2lvds-dual-avb.dts
```

输入模块介绍

RGB 输入

配置RGB输入

V19版本开始在DTS里配置

```
rk628,hdmi-in;
```

更早版本是在rk628.h中配置

```
#define RK628_RGB_IN
```

注：BT1120和RGB共用一套驱动代码，通过宏定义隔离。

BT1120 输入

配置BT1120输入

V19版本开始在DTS里配置

```
rk628,bt1120-in;
```

更早版本是在rk628.h中配置

```
#define RK628_BT1120_IN
```

注：BT1120和RGB共用一套驱动代码，通过宏定义隔离。

配置csc

V19版本开始在DTS里配置，更早版本是在rk628.h中配置。

```
rk628_config.c

输入:

enum bus_format input_bus_format = BUS_FMT_YUV420;

按需配置输出，例如RGB:

enum bus_format output_bus_format = BUS_FMT_RGB;
```

按需配置双边沿

V19版本开始在DTS里配置，更早版本是在rk628.h中配置。

```
rk628.h

#define BT1120_DUAL_EDGE
```

注：该部分代码未完全验证完。

HDMI 输入

配置HDMI输入

V19版本开始在DTS里配置

```
rk628,hdmi-in;
```

更早版本是在rk628.h中配置

```
#define RK628_HDMI_IN
```

HDMIRX dts 配置

目前 HDMI 输入有两种形态，一种是与 Soc 直连，一种是通过 HDMI 线缆连接。

1. **与 Soc 直连的模式**

与 Soc 直连的模式，需要 Source 端默认输出，所以不需要热插拔检测脚，这种情况下一般是不支持分辨率切换，所以在该情况下，驱动采用轮询方式去检测 HDMI 信号的输入，所以在该情况下，dts 不需要配置 plugin-det-gpios.

在该模式下，HDMI 4K YUV420 输入需要配置 src-mode-4k-yuv420，因为HDMIRX无法提前判断HDMI Source是否是 YUV420 输出。

2. **通过 HDMI 线缆连接**

   HDMI 线缆连接需要热插拔检测，同时也有分辨率/颜色格式切换的需求，所以该模式下需要配置plugin-det-gpios，默认采用中断方式。

   hpd-output-inverted：HPD输出取反配置，若HPD输出电平在电路上做了取反，则需要使能此配置项，配置如下：

```
rk628: rk628@50 {
    compatible = "rockchip,rk628";
    reg = <0x50>;
    interrupt-parent = <&gpio2>;
    interrupts = <2 IRQ_TYPE_LEVEL_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&hdmirx_plugin_det>;
    reset-gpios = <&gpio2 RK_PA3 GPIO_ACTIVE_LOW>;
+   plugin-det-gpios = <&gpio2 RK_PA4 GPIO_ACTIVE_LOW>;
+   /* hpd-output-inverted */
    panel-enable-gpios = <&gpio2 RK_PA5 GPIO_ACTIVE_HIGH>;
    panel-backlight = <&backlight>;
    status = "okay";
};
```

修改分辨率

1. HDMI 线缆连接模式，修改HDMIRX的分辨率，参考EDID的配置方法。
2. 与 Soc 直接模式，修改输入源分辨率，以RK3399为例：

```
diff --git a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
index 0808c203b9d1..b012ba63b8a5 100644
--- a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
+++ b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
@@ -2617,7 +2617,7 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                                             connector);
        struct edid *edid;
        struct drm_display_mode *mode;
-       const u8 def_modes[6] = {4, 16, 31, 19, 17, 2};
+       const u8 def_modes[6] = {108, 4, 16, 31, 19, 17};
        struct drm_display_info *info = &connector->display_info;
        struct hdr_static_metadata *metedata =
                        &connector->display_info.hdmi.hdr_panel_metadata;
@@ -2627,6 +2627,7 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                return 0;

        edid = drm_get_edid(connector, hdmi->ddc);
+       edid = NULL;
        if (edid) {
                dev_dbg(hdmi->dev, "got edid: width[%d] x height[%d]\n",
                        edid->width_cm, edid->height_cm);
@@ -2641,8 +2642,6 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                drm_mode_connector_update_hdr_property(connector, metedata);
                kfree(edid);
        } else {
+               hdmi->sink_is_hdmi = true;
+               hdmi->sink_has_audio = true;
                for (i = 0; i < sizeof(def_modes); i++) {
                        mode = drm_display_mode_from_vic_index(connector,
                                                                def_modes,
diff --git a/drivers/gpu/drm/drm_edid.c b/drivers/gpu/drm/drm_edid.c
index a61d52772dc1..e0e0d9cdf212 100644
--- a/drivers/gpu/drm/drm_edid.c
+++ b/drivers/gpu/drm/drm_edid.c
@@ -1220,6 +1220,11 @@ static const struct drm_display_mode edid_cea_modes[]
= {
                4104, 4400, 0, 2160, 2168, 2178, 2250, 0,
                DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
          .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_64_27,
},
+       /* 108 - 1080x1920p@60Hz */
+       { DRM_MODE("1080x1920", DRM_MODE_TYPE_DRIVER, 148500, 1080, 1120,
+               1140, 1220, 0, 1920, 1940, 1950, 1970, 0,
+               DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
+         .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_16_9,
},
 };

 /*
```

添加竖屏分辨率 1080x1920, 然后需要注意的是RK628不支持DVI，所以需要把sink_is_hdmi置成true.

HDMIRX 频点支持

目前 HDMIRX 支持以下这些频点，所以要求前端 HDMI 输入的分辨率需要满足以下频点，比如上面添加的1080x1920@60的分辨率，频点是 148500，有在下面列表中。

如果屏的分辨率对应的频点不在下面列表中，那么可以下面列表中找到相近的频点进行替换，影响就是屏的刷新率可能大于6060Hz，也可能小于60Hz.

```
25175, 27000, 33750, 40000, 59400, 65000, 68250, 74250,
75000, 83500, 85500, 88750, 928125, 101000, 102250, 108000,
118800, 119000, 135000, 148500, 150000, 162000, 165000, 297000
```

频点的计算的公式：`htotal * vtotal * fps = clock`

举个例子：屏的 `htotal: 1354, vtotal: 816, fps = 60`, 所以 `clock = 1354 * 816 * 60 = 66300 KHz`，不在上面的列表中，然后找一个相近的 65000KHz，换算之后的 `fps = 65000KHz / 1354 / 816 = 59Hz`.

HDMI2GVI 4K60 输出

因为 HDMIRX 4K60只支持YUV420，所以要求前端HDMI OUT 输出4K60 YUV420 的分辨率，所以需要有以下这些配置：

1. RK628 HDMIRX dts 需要配置如下配置：

```
rk628: rk628@50 {
    compatible = "rockchip,rk628";
    reg = <0x50>;
    interrupt-parent = <&gpio2>;
    interrupts = <2 IRQ_TYPE_LEVEL_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&hdmirx_plugin_det>;
    reset-gpios = <&gpio2 RK_PA3 GPIO_ACTIVE_LOW>;
+   src-mode-4k-yuv420;
    panel-enable-gpios = <&gpio2 RK_PA5 GPIO_ACTIVE_HIGH>;
    panel-backlight = <&backlight>;
    status = "okay";
};
```

2. 主控 HDMI OUT 需要如下配置，（参考代码为 kernel 4.4，其他版本可能代码有差异，但修改方法是一样的）：

```
From 43a892071c4f121551ba118e46e3fe0f0541849e Mon Sep 17 00:00:00 2001
From: Chen Shunqing <csq@rock-chips.com>
Date: Sat, 8 Jan 2022 16:03:09 +0800
Subject: [PATCH 1/2] drm/bridge: synopsys: dw-hdmi: 4k60 yuv420 output
default

Signed-off-by: Chen Shunqing <csq@rock-chips.com>
---
 drivers/gpu/drm/bridge/synopsys/dw-hdmi.c    | 12 +++++++-----
 drivers/gpu/drm/rockchip/dw_hdmi-rockchip.c |  4 ++++
 2 files changed, 11 insertions(+), 5 deletions(-)

diff --git a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
index 0808c203b9d1..8e66876abc3e 100644
--- a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
+++ b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
@@ -1652,6 +1652,7 @@ static void dw_hdmi_phy_disable(struct dw_hdmi *hdmi,
void *data)
 enum drm_connector_status dw_hdmi_phy_read_hpd(struct dw_hdmi *hdmi,
                               void *data)
 {
+   return connector_status_connected;
    return hdmi_readb(hdmi, HDMI_PHY_STAT0) & HDMI_PHY_HPD ?
        connector_status_connected : connector_status_disconnected;
 }
@@ -2617,7 +2618,7 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                         connector);
    struct edid *edid;
    struct drm_display_mode *mode;
-   const u8 def_modes[6] = {4, 16, 31, 19, 17, 2};
+   const u8 def_modes[6] = {97, 16, 31, 19, 17, 2};
    struct drm_display_info *info = &connector->display_info;
    struct hdr_static_metadata *metedata =
            &connector->display_info.hdmi.hdr_panel_metadata;
@@ -2626,7 +2627,8 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
    if (!hdmi->ddc)
```

```
        return 0;

-   edid = drm_get_edid(connector, hdmi->ddc);
+   /* edid = drm_get_edid(connector, hdmi->ddc); */
+   edid = NULL;
    if (edid) {
        dev_dbg(hdmi->dev, "got edid: width[%d] x height[%d]\n",
            edid->width_cm, edid->height_cm);
@@ -2657,8 +2659,8 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                ret++;
            }
        }
-       info->edid_hdmi_dc_modes = 0;
-       info->hdmi.y420_dc_modes = 0;
+       info->edid_hdmi_dc_modes = 1;
+       info->hdmi.y420_dc_modes = 1;
        info->color_formats = 0;

        dev_info(hdmi->dev, "failed to get edid\n");
@@ -3794,7 +3796,7 @@ int dw_hdmi_bind(struct device *dev, struct device
*master,

    hdmi->initialized = false;
    ret = hdmi_readb(hdmi, HDMI_PHY_STAT0);
-   if ((ret & HDMI_PHY_TX_PHY_LOCK) && (ret & HDMI_PHY_HPD) &&
+   if ((ret & HDMI_PHY_TX_PHY_LOCK) /*&& (ret & HDMI_PHY_HPD)*/ &&
        hdmi_readb(hdmi, HDMI_FC_EXCTRLDUR)) {
        hdcp_stat = hdmi_readb(hdmi, HDMI_A_APIINTSTAT);
        hdmi->mc_clkdis = hdmi_readb(hdmi, HDMI_MC_CLKDIS);
diff --git a/drivers/gpu/drm/rockchip/dw_hdmi-rockchip.c
b/drivers/gpu/drm/rockchip/dw_hdmi-rockchip.c
index 3c98573ca01b..3bb4a8e9379d 100644
--- a/drivers/gpu/drm/rockchip/dw_hdmi-rockchip.c
+++ b/drivers/gpu/drm/rockchip/dw_hdmi-rockchip.c
@@ -534,11 +534,13 @@ dw_hdmi_rockchip_mode_valid(struct drm_connector
*connector,
     * clock > 340MHz is YCbCr420 or not and whether the platform supports
     * YCbCr420.
     */
+#if 0
    if (mode->clock > 340000 &&
        connector->display_info.max_tmds_clock < 340000 &&
        (!drm_mode_is_420(&connector->display_info, mode) ||
         !connector->ycbcr_420_allowed))
        return MODE_BAD;
+#endif

    if (!encoder) {
        const struct drm_connector_helper_funcs *funcs;
@@ -726,6 +728,8 @@ dw_hdmi_rockchip_select_output(struct
drm_connector_state *conn_state,
        break;
    }

+   *color_format = DRM_HDMI_OUTPUT_YCBCR420;
+
    if (*color_format == DRM_HDMI_OUTPUT_DEFAULT_RGB &&
        info->edid_hdmi_dc_modes & DRM_EDID_HDMI_DC_30)
        support_dc = true;

2.17.1
```

输出模块介绍

DSI输出

DSI 输出配置

V19版本开始在DTS里配置DSI类型，更早版本则在 rk628_config.c 中配置 DSI 输出类型。

属性说明

Property	Description	Option Value
bpp	bits for a pixel	16/18/24
bus_format	color mapping	MIPI_DSI_FMT_RGB888/RGB666/RGB666_PACKED/RGB565
lanes	select dsi host lanes	1/2/4
slave	dual channel dsi	true/false
master	dual channel dsi	true/false

注：除了表格中描述的属性可以根据 option value 做修改外，其他属性不建议修改。

单 DSI 输出

```
/* config dsi0 */
const static struct rk628_dsi rk628_dsi0 = {
    /* .bpp = */ 24,
    /* .bus_format */ MIPI_DSI_FMT_RGB888,
    /* .slave = */ false,
    /* .master = */ false,
    /* .channel = */ 0,
    /* .lanes */ 4,
    /* .id */ 0,
    /* .mode_flags */ MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
              MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET,
};

/* config dsi1 for dual channel dsi */
const static struct rk628_dsi rk628_dsi1 = {
    /* .bpp = */ 24,
    /* .bus_format */ MIPI_DSI_FMT_RGB888,
    /* .slave = */ false,
    /* .master = */ false,
    /* .channel = */ 0,
    /* .lanes */ 4,
    /* .id */ 0,
    /* .mode_flags */ MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
              MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET,
};
```

双 DSI 输出

修改 rk628_dsi0 slave 为 true，rk628_dsi1 master 为 true:

```
/* config dsi0 */
const static struct rk628_dsi rk628_dsi0 = {
    /* .bpp = */ 24,
    /* .bus_format */ MIPI_DSI_FMT_RGB888,
    /* .slave = */ true,
    /* .master = */ false,
    /* .channel = */ 0,
    /* .lanes */ 4,
    /* .id */ 0,
    /* .mode_flags */ MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
              MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET,
};

/* config dsi1 for dual channel dsi */
const static struct rk628_dsi rk628_dsi1 = {
    /* .bpp = */ 24,
    /* .bus_format */ MIPI_DSI_FMT_RGB888,
    /* .slave = */ false,
    /* .master = */ true,
    /* .channel = */ 0,
    /* .lanes */ 4,
    /* .id */ 0,
    /* .mode_flags */ MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
              MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET,
};
```

DSI panel 初始化序列配置

根据 panel vendor 提供的初始化序列按如下格式配置:

```
struct panel_cmd_seq {
    u16 cmd_cnt;
    u8 init_sequence[15][5];
} panel_cmd_init_seq = {
    15,
    {
        { 0x23, 0x00, 0x02, 0xFE, 0x21 },
        { 0x23, 0x00, 0x02, 0x04, 0x00 },
        { 0x23, 0x00, 0x02, 0x00, 0x64 },
        { 0x23, 0x00, 0x02, 0x2A, 0x00 },
        { 0x23, 0x00, 0x02, 0x26, 0x64 },
        { 0x23, 0x00, 0x02, 0x54, 0x00 },
        { 0x23, 0x00, 0x02, 0x50, 0x64 },
        { 0x23, 0x00, 0x02, 0x7B, 0x00 },
        { 0x23, 0x00, 0x02, 0x77, 0x64 },
        { 0x23, 0x00, 0x02, 0xA2, 0x00 },
        { 0x23, 0x00, 0x02, 0x9D, 0x64 },
        { 0x23, 0x00, 0x02, 0xC9, 0x00 },
        { 0x23, 0x00, 0x02, 0xC5, 0x64 },
        { 0x05, 0x78, 0x01, 0x11 },
        { 0x05, 0x1E, 0x01, 0x29 },
```

panel_init_sequence 是定义的二维数组，第一列表示 data type，第二列表示 mdelays，第三列表示发送每条命令的 payload_lenth，后面几列都是表示每条命令的 payload。

常见数据类型

data type	description	packet size
0x03	Generic Short WRITE, no parameters	short
0x13	Generic Short WRITE, 1 parameters	short
0x23	Generic Short WRITE, 2 parameters	short
0x29	Generic long WRITE,	long
0x05	DCS Short WRITE, no parameters	short
0x15	DCS Short WRITE, 1 parameters	short

LVDS输出

LVDS输出配置

V19版本开始在DTS里配置LVDS类型，更早版本则在 rk628_config.c 中配置 LVDS 输出类型。

配置属性说明

property	description
LVDS_SINGLE_LINK	单通道LVDS
LVDS_DUAL_LINK_ODD_EVEN_PIXELS	双通道LVDS，左右通道为奇偶通道
LVDS_DUAL_LINK_EVEN_ODD_PIXELS	双通道LVDS，左右通道为偶奇通道
LVDS_DUAL_LINK_LEFT_RIGHT_PIXELS	双通道LVDS，左右通道为左右屏
LVDS_DUAL_LINK_RIGHT_LEFT_PIXELS	双通道LVDS，左右通道为右左屏

单 LVDS 输出

```
uint32_t rk628_lvds_get_link_type(void)
{
    return LVDS_SINGLE_LINK;
}
```

双 LVDS 输出

```
u32 rk628_lvds_get_link_type(void)
{
    return LVDS_DUAL_LINK_ODD_EVEN_PIXELS;
}
```

注：根据屏规格可以选择配置双通道 LVDS 的奇偶特性。

双 LVDS 左右屏

```
u32 rk628_lvds_get_link_type(void)
{
    return LVDS_DUAL_LINK_LEFT_RIGHT_PIXELS;
}
```

LVDS 数据格式配置

```
u32 rk628_lvds_get_format(void)
{
    /*
     * LVDS_FORMAT_VESA_24BIT
     * LVDS_FORMAT_JEIDA_24BIT
     * LVDS_FORMAT_JEIDA_18BIT
     * LVDS_FORMAT_VESA_18BIT
     */
    return LVDS_FORMAT_JEIDA_24BIT;
}
```

GVI输出

GVI输出配置

V19版本开始在DTS里配置GVI类型，更早版本则在 rk628_config.c 中配置 GVI 输出类型

GVI属性说明

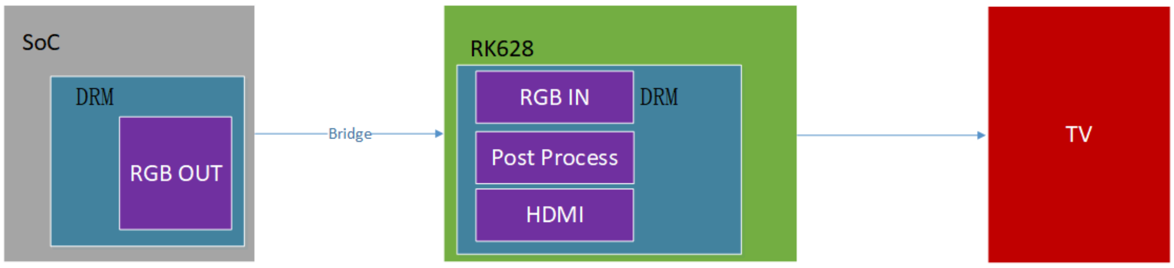
Property	Description	Option Value
bus_format	color mapping	MIPI_DSI_FMT_RGB888/RGB666/RGB666_PACKED/RGB565
lanes	select gvi host lanes	1/2/4/8
byte_mode	byte mode	3/4/5
color_depth	color depth	COLOR_DEPTH_RGB_YUV444_30BIT/24BIT/18BIT

```
static struct rk628_gvi rk628_gvi = {
    /* .bus_format = */ MEDIA_BUS_FMT_RGB888_1X24,
    /* .lanes = */ 8,
    /* .division_mode = */ false,
    /* .byte_mode = */ 3,
    /* .color_depth = */ COLOR_DEPTH_RGB_YUV444_24BIT,
};
```

RGB输出

HDMI输出

考虑到 HDMI 需要支持切不同分辨率的需求，当前 Misc 的 RK628 驱动无法支持，所以 HDMI OUT 驱动用 DRM 框架实现。所以对于非 DRM 框架 RK628 不支持 HDMI OUT.



目前支持的几种组合

RGB2DSI转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于RGB2DSI的参考配置方法。

```
drivers/misc/rk628/rk628.h
#define RK628_RGB_IN
#undef RK628_HDMI_IN
#define RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts
+++ b/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts
+               rk628,rgb-in;
+               rk628-dsi {
+                       //rockchip,dual-channel;
+                       dsi,eotp;
+                       dsi,video-mode;
+                       dsi,format = "rgb888";
+                       dsi,lanes  = <4>;
+                       status = "okay";
                }
```

2. dts修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2dsi-avb.dts，其中rk3288-evb-rk628-misc-rgb2dsi-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts 增加RK628支持RGB转MIPI DSI的功能，具体差异请自行对比代码。

RGB2LVDS转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于RGB2LVDS的参考配置方法。

```
drivers/misc/rk628/rk628.h
#define RK628_RGB_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#define RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2lvds-avb.dts
+++ b/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2lvds-avb.dts
@@ -111,6 +111,54 @@
                reset-gpios = <&gpio7 RK_PB6 GPIO_ACTIVE_LOW>;
                //panel-enable-gpios = <&gpio7 RK_PA2 GPIO_ACTIVE_HIGH>;
                status = "okay";
+
+               rk628,rgb-in;
+               rk628-lvds {
+                       /* "jeida_18","vesa_24","vesa_18" */
+                       bus-format = "jeida_24";
+
+                       /* "dual_link_odd_even_pixels"
+                        * "dual_link_even_odd_pixels"
+                        * "dual_link_left_right_pixels"
+                        * "dual_link_right_left_pixels"
+                        */
+                       link-type = "single_link";
+                       status = "okay";
+               };
```

2. dts修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2lvds-avb.dts，其中rk3288-evb-rk628-misc-rgb2lvds-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts 增加RK628支持RGB转LVDS的功能，具体差异请自行对比代码。

RGB2GVI转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于RGB2GVI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#define RK628_RGB_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#undef RK628_LVDS_OUT
#define RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2gvi-avb.dts
+++ b/arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2gvi-avb.dts
@@ -122,6 +122,56 @@
```

```
                reset-gpios = <&gpio7 RK_PB6 GPIO_ACTIVE_LOW>;
                //panel-enable-gpios = <&gpio7 RK_PA2 GPIO_ACTIVE_HIGH>;
                status = "okay";
+
+               rk628,rgb-in;
+
+               rk628-gvi {
+                       /* "rgb666"
+                        * "rgb888"
+                        * "rgb101010"
+                        * "yuyv8"
+                        * "yuyv10"
+                        */
+                       bus-format = "rgb888";
+                       gvi,lanes = <8>;
+                       //"rockchip,division-mode";
+                       //"rockchip, gvi-frm-rst";
+                       status = "okay";
+               };
```

2. dts修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2gvi-avb.dts，其中rk3288-evb-rk628-misc-rgb2gvi-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts 增加RK628支持RGB转GVI的功能，具体差异请自行对比代码。

RGB2HDMI转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于RGB2HDMI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#define RK628_RGB_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
#define RK628_HDMI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制，需要在RK628的DTS里新增“rk628,rgb-in;”配置。

2. dts 修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-rgb2hdmi-avb.dts，其中rk3288-evb-rk628-misc-rgb2hdmi-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts 增加RK628支持RGB转GVI的功能，具体差异请自行对比代码，这边 HDMI OUT 与其他输出不一样的地方是，HDMI 基于 DRM 框架来实现，所以 dts 需要如下配置：

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

        ports {
                port@1 {
                        reg = <1>;
                        rgb_out_hdmi: endpoint {
                                remote-endpoint = <&rgb_in_hdmi>;
                        };
                };
        };
};

&i2c1 {
        clock-frequency = <400000>;
        status = "okay";

        rk628: rk628@50 {
                compatible = "rockchip,rk628";
                reg = <0x50>;
                pinctrl-names = "default";
                pinctrl-0 = <&test_clkout>;
                assigned-clocks = <&cru SCLK_TESTOUT_SRC>;
                assigned-clock-parents = <&xin24m>;
                clocks = <&cru SCLK_TESTOUT>;
                clock-names = "soc_24M";
                interrupt-parent = <&gpio7>;
                interrupts = <15 IRQ_TYPE_LEVEL_HIGH>;
                enable-gpios = <&gpio5 RK_PC2 GPIO_ACTIVE_HIGH>;
                reset-gpios = <&gpio7 RK_PB6 GPIO_ACTIVE_LOW>;
                status = "okay";

                port {
                        rgb_in_hdmi: endpoint {
                                remote-endpoint = <&rgb_out_hdmi>;
                        };
                };
        };
};
```

BT1120-DSI转换

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于BT1120转DSI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_BT1120_IN
#undef RK628_HDMI_IN
#define RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
#undef RK628_HDMI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

DTS参考RGB2DSI的即可，其中"rk628,rgb-in;" 改成 “rk628,bt1120-in;”。

BT1120-LVDS转换

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于BT1120转LVDS的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_BT1120_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#define RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
#undef RK628_HDMI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

DTS参考RGB2LVDS的即可，其中"rk628,rgb-in;" 改成 “rk628,bt1120-in;”。

BT1120-GVI转换

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于BT1120转GVI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_BT1120_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#undef RK628_LVDS_OUT
#define RK628_GVI_OUT
#undef RK628_CSI_OUT
#undef RK628_HDMI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

DTS参考RGB2GVI的即可，其中"rk628,rgb-in;" 改成 “rk628,bt1120-in;”。

BT1120-HDMI转换

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于BT1120转HDMI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_BT1120_IN
#undef RK628_HDMI_IN
#undef RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
#define RK628_HDMI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

DTS参考RGB2HDMI的即可，其中"rk628,rgb-in;" 改成 “rk628,bt1120-in;”。

HDMI2DSI转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于HDMI2DSI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_HDMI_IN
#define RK628_DSI_OUT
#undef RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2dsi-avb.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2dsi-avb.dts
@@ -231,6 +231,322 @@
                panel-enable-gpios = <&gpio2 RK_PA5 GPIO_ACTIVE_HIGH>;
                panel-backlight = <&backlight>;
                status = "okay";
+               rk628,hdmi-in;
+               rk628-dsi {
+                       //rockchip,dual-channel;
+                       dsi,eotp;
+                       dsi,video-mode;
+                       dsi,format = "rgb888";
+                       dsi,lanes  = <4>;
+                       status = "okay";
```

2. dts 修改

可以参考以下DTS arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2dsi-avb.dts，其中rk3399-evb-ind-lpddr4-rk628-hdmi2dsi-avb.dts是基于rockchip/rk3399-evb-ind-lpddr4-android-avb.dts增加RK628支持HDMI转DSI的功能，具体差异请自行对比代码。

HDMI2LVDS转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于HDMI2LVDS的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_HDMI_IN
#undef RK628_DSI_OUT
#define RK628_LVDS_OUT
#undef RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm/boot/dts/rk3288-evb-rk628-misc-hdmi2lvds-avb.dts
+++ b/arch/arm/boot/dts/rk3288-evb-rk628-misc-hdmi2lvds-avb.dts
@@ -56,6 +56,53 @@
                reset-gpios = <&gpio7 RK_PB6 GPIO_ACTIVE_LOW>;
                panel-enable-gpios = <&gpio7 RK_PA2 GPIO_ACTIVE_HIGH>;
                status = "okay";
+               rk628,hdmi-in;
+               rk628-lvds {
+                       /* "jeida_18","vesa_24","vesa_18" */
+                       bus-format = "jeida_24";
+
+                       /* "dual_link_odd_even_pixels"
+                        * "dual_link_even_odd_pixels"
+                        * "dual_link_left_right_pixels"
+                        * "dual_link_right_left_pixels"
+                        */
+                       link-type = "single_link";
+                       status = "okay";
+               };
```

2. dts 修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-hdmi2lvds-avb.dts，其中rk3288-evb-rk628-misc-hdmi2lvds-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts增加RK628支持HDMI转LVDS的功能，具体差异请自行对比代码。

HDMI2GVI转换

1. 代码修改

需要支持的用#define,不需要支持的用#undef，例如下面这个是用于HDMI2GVI的参考配置方法：

```
drivers/misc/rk628/rk628.h
#undef RK628_RGB_IN
#define RK628_HDMI_IN
#undef RK628_DSI_OUT
#undef RK628_LVDS_OUT
#define RK628_GVI_OUT
#undef RK628_CSI_OUT
```

从V19版本开始取消了上面的配置方案，改由完全在DTS里控制。

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2gvi-avb.dts
+++ b/arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2gvi-avb.dts
@@ -185,6 +185,56 @@
                clocks = <&cru SCLK_CIF_OUT>;
                clock-names = "soc_24M";
                status = "okay";
+               rk628,hdmi-in;
+
+               rk628-gvi {
+                       /* "rgb666"
+                        * "rgb888"
+                        * "rgb101010"
+                        * "yuyv8"
+                        * "yuyv10"
+                        */
+                       bus-format = "rgb888";
+                       gvi,lanes = <8>;
+                       //"rockchip,division-mode";
+                       //"rockchip, gvi-frm-rst";
+                       status = "okay";
+               };
```

2. dts修改

可以参考以下DTS arch/arm/boot/dts/rk3288-evb-rk628-misc-hdmi2gvi-avb.dts，其中rk3288-evb-rk628-misc-hdmi2gvi-avb.dts是基于rk3288-evb-android-rk808-edp-avb.dts 增加RK628支持HDMI转GVI的功能，具体差异请自行对比代码。

这里以RK356X ANDROID11为例，新增以下DTS可以支持HDMI2GVI功能（引脚请自行修改）。

```
&hdmi {
    status = "okay";
    rockchip,phy-table =
        <92812500  0x8009 0x0000 0x0270>,
        <165000000 0x800b 0x0000 0x026d>,
        <185625000 0x800b 0x0000 0x01ed>,
        <297000000 0x800b 0x0000 0x01ad>,
        <594000000 0x8029 0x0000 0x0088>,
        <000000000 0x0000 0x0000 0x0000>;

};

&hdmi_in_vp0 {
    status = "okay";
};

&hdmi_in_vp1 {
    status = "disabled";
};


&i2c3 {
    status = "okay";
    rk628: rk628@50 {
      compatible = "rockchip,rk628";
        reg = <0x50>;
        interrupt-parent = <&gpio2>;
        interrupts = <RK_PA0 IRQ_TYPE_LEVEL_HIGH>;
        enable-gpios = <&gpio0 RK_PA4 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio1 RK_PD7 GPIO_ACTIVE_LOW>;
    pinctrl-0 = <&refclk_pins>;
    pinctrl-names = "default";
        status = "okay";
    assigned-clocks = <&pmucru CLK_WIFI>;
    assigned-clock-rates = <24000000>;
    clocks = <&pmucru CLK_WIFI>;
        clock-names = "soc_24M";
    };

};
```

基础调试命令

1. 寄存器调试节点

```
console:/ # ls /d/regmap/
0-0010/                   4-0050-combtxphy/         4-0050-gpio3/
0-001c/                   4-0050-cru/               4-0050-grf/
0-0020-rk817-codec/       4-0050-csi/               4-0050-gvi/
0-0020/                   4-0050-dsi0/              4-0050-hdmi/
1-000f/                   4-0050-dsi1/              4-0050-hdmirx/
4-0022/                   4-0050-gpio0/             ff870000.spdif/
4-0050-adapter/           4-0050-gpio1/             ff880000.i2s/
4-0050-combrxphy/         4-0050-gpio2/             ff8a0000.i2s/
```

其中，4 表示I2C总线号，0050表示RK628 SLAVE地址。

2. 读GRF寄存器

```
console:/ # cat /d/regmap/4-0050-grf/registers
000: 0600062b
004: ffffffff
008: 00000000
00c: 00000000
010: 00000001
014: 00000000
018: 00050000
01c: 000a032a
020: 00320302
…
```

3. 写GRF寄存器

```
console:/ # echo 0x000 0xffffffff > /d/regmap/4-0050-grf/registers
```

4. 读hdmirx寄存器

```
console:/ # cat /sys/kernel/debug/regmap/4-0050-hdmirx/registers
```

5. 自测模式命令

在调试过程中，可以通过以下命令测试输出模块的控制器、对应的 phy、屏端这条链路是否正常工作，如果 color bar 能正常显示，请检查主控输出、RK628 input、RK628 Process 的配置，反之请检查对应输出接口和屏端的配置：

1. HDMITX color bar

```
console:/ # echo 0x70324 0x00 > /d/regmap/1-0050-hdmi/registers
```

2. DSI color bar

```
console:/ # echo 0x50038 0x13f02 > /d/regmap/1-0050-dsi0/registers
```

3. GVI color bar

```
console:/ # echo  0x80060 0x1 > /d/regmap/1-0050-gvi/registers
```

显示常见问题处理

没有生成regmap节点

```
rk3568_r:/ # dmesg | grep rk628
[    0.294331] rk628 3-0050: RK628 misc driver version: 0.0.1
[    0.294426] rk628 3-0050: failed to request panel enable GPIO: -16
[    0.294463] rk628: probe of 3-0050 failed with error -16
```

I2C通信异常

如下log表示RK628的I2C通信异常导致RK628的各个模块注册不上，需要检查RK628的时序以及24MHz的基准时钟（见常见需求处理章节），以及相关pin的iomux（见基础调试命令章节）。

```
[    0.960609] rk628 1-0050: failed to access register: -6
```

hdmi-rx clock detected failed

```
[    3.746543] rk628 1-0050: clk det over cnt:10, reg_0x6654:0x501f0000
[    3.746570] rk628 1-0050: Clock detection anomaly
[    3.747726] rk628 1-0050: clock detected failed, cfg resistance manual!
[    3.873552] rk628 1-0050: clk not stable, reg_0x6630:0x87000d,
reg_0x6608:0x110100
[    3.873578] hdmi rxphy power on failed
```

对于此问题可以按以下步骤排查：

1 读取rk628寄存器的值，如果寄存器值类似于下面这种情况，则表示I2C没通信成功，可以检查下24M晶体或者RESET脚的状态是否正常。

```
rk322x_box:/ # cat /d/regmap/1-0050-dsi0/registers
50000: xxxxxxxx
50004: xxxxxxxx
50008: xxxxxxxx
5000c: xxxxxxxx
50010: xxxxxxxx
50014: xxxxxxxx
```

2 修改DTS，尝试降低I2C的速度为100KHZ或更低。

3 检查HPD极性是否正确，补丁类似修改如下：

```
--- a/drivers/misc/rk628/rk628_hdmirx.c
+++ b/drivers/misc/rk628/rk628_hdmirx.c
@@ -143,7 +143,7 @@ static void rk628_hdmirx_disable_edid(struct rk628 *rk628)
 static void rk628_hdmirx_enable_edid(struct rk628 *rk628)
 {
        rk628_i2c_update_bits(rk628, HDMI_RX_HDMI_SETUP_CTRL,
HOT_PLUG_DETECT_MASK,

-              HOT_PLUG_DETECT(1));
+              HOT_PLUG_DETECT(0));
               }
```

4 尝试固定HDMI-OUT的分辨率

对于HDMI直连的方式，需要根据RK628送显的分辨率来调整HDMI-TX端的分辨率，并输出固定的HDMI分辨率，对于RK平台的KERNEL4.4代码，可以参补丁包的补丁或下面的范例，

```
--- a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
+++ b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
@@ -2617,7 +2617,11 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                                                     connector);
        struct edid *edid;
        struct drm_display_mode *mode;
+#if HDMI_KEEP_RESOLUTION
+       const u8 def_modes[6] = {108, 16, 31, 19, 17, 2};
+#else
        const u8 def_modes[6] = {4, 16, 31, 19, 17, 2};
+#endif
        struct drm_display_info *info = &connector->display_info;
        struct hdr_static_metadata *metedata =
                        &connector->display_info.hdmi.hdr_panel_metadata;
@@ -2627,6 +2631,9 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                return 0;

        edid = drm_get_edid(connector, hdmi->ddc);
+#if HDMI_KEEP_RESOLUTION
+       edid = NULL;
+#endif
        if (edid) {
                dev_dbg(hdmi->dev, "got edid: width[%d] x height[%d]\n",
                        edid->width_cm, edid->height_cm);
@@ -2657,6 +2664,10 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                                ret++;
                        }
                }
+#if HDMI_KEEP_RESOLUTION
+               hdmi->sink_is_hdmi  = true;
+               hdmi->sink_has_audio = true;
+#endif
                info->edid_hdmi_dc_modes = 0;
                info->hdmi.y420_dc_modes = 0;
                info->color_formats = 0;
diff --git a/drivers/gpu/drm/drm_edid.c b/drivers/gpu/drm/drm_edid.c
old mode 100644
new mode 100755
index a61d52772dc1..430a92fec29f
--- a/drivers/gpu/drm/drm_edid.c
+++ b/drivers/gpu/drm/drm_edid.c
@@ -1220,6 +1220,13 @@ static const struct drm_display_mode edid_cea_modes[] = {
                4104, 4400, 0, 2160, 2168, 2178, 2250, 0,
                DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
          .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_64_27, },
+#if HDMI_KEEP_RESOLUTION
+        /* 108 - 1080x1920@60Hz */
+       { DRM_MODE("1080x1920", DRM_MODE_TYPE_DRIVER, 148500, 1080, 1120,
+               1140, 1220, 0, 1920, 1920+20, 1920+20+10, 1920+20+10+20, 0,
+                DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
+                .vrefresh = 60, .picture_aspect_ratio =
HDMI_PICTURE_ASPECT_64_27, },
+#endif
 };

 /*
diff --git a/include/linux/hdmi.h b/include/linux/hdmi.h
index a8095e1d62b5..29a2bb122c9f 100644
--- a/include/linux/hdmi.h
+++ b/include/linux/hdmi.h
@@ -41,6 +41,7 @@ enum hdmi_infoframe_type {
 #define HDMI_AVI_INFOFRAME_SIZE    13
 #define HDMI_SPD_INFOFRAME_SIZE    25
 #define HDMI_AUDIO_INFOFRAME_SIZE  10
+#define HDMI_KEEP_RESOLUTION    1
```

对于RK平台的KERNEL 4.19代码可以类似修改：

```
--- a/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
+++ b/drivers/gpu/drm/bridge/synopsys/dw-hdmi.c
@@ -202,6 +202,13 @@ static const u16 csc_coeff_full_to_limited[3][4] = {
 };

 static const struct drm_display_mode dw_hdmi_default_modes[] = {
+#if HDMI_KEEP_RESOLUTION
+       /* 16 - 1920x1080@60Hz 16:9 */
+       { DRM_MODE("1920x1080", DRM_MODE_TYPE_DRIVER, 148500, 1920, 2008,
+                  2052, 2200, 0, 1080, 1084, 1089, 1125, 0,
+                  DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
+       .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_16_9, },
+#else
        /* 4 - 1280x720@60Hz 16:9 */
        { DRM_MODE("1280x720", DRM_MODE_TYPE_DRIVER, 74250, 1280, 1390,
                   1430, 1650, 0, 720, 725, 730, 750, 0,
@@ -236,6 +243,7 @@ static const struct drm_display_mode dw_hdmi_default_modes[]
= {
                   798, 858, 0, 480, 489, 495, 525, 0,
                   DRM_MODE_FLAG_NHSYNC | DRM_MODE_FLAG_NVSYNC),
          .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_4_3, },
+#endif
 };

 struct hdmi_vmode {
@@ -2751,6 +2759,9 @@ static int dw_hdmi_connector_get_modes(struct
drm_connector *connector)
                return 0;

        edid = drm_get_edid(connector, hdmi->ddc);
+#if HDMI_KEEP_RESOLUTION
+       edid = NULL;
+#endif
        if (edid) {
                dev_dbg(hdmi->dev, "got edid: width[%d] x height[%d]\n",
                        edid->width_cm, edid->height_cm);
```

其中需要注意下，不同的主控采用的代码不用，例如RK3588采用的是dw-hdmi-qp.c，而RK3036/RK3128采用的是inno_hdmi.c，固定分辨率的补丁如下：

```
--- a/drivers/gpu/drm/rockchip/inno_hdmi.c
+++ b/drivers/gpu/drm/rockchip/inno_hdmi.c
@@ -623,7 +623,11 @@ static int inno_hdmi_connector_get_modes(struct
drm_connector *connector)
    struct inno_hdmi *hdmi = to_inno_hdmi(connector);
    struct drm_display_mode *mode;
    struct drm_display_info *info = &connector->display_info;
+#if HDMI_KEEP_RESOLUTION
+   const u8 def_modes[6] = {108, 16, 31, 19, 17, 2};
+#else
    const u8 def_modes[6] = {4, 16, 31, 19, 17, 2};
+#endif
    struct edid *edid;
    int ret = 0;
    u8 i;
@@ -632,6 +636,9 @@ static int inno_hdmi_connector_get_modes(struct
drm_connector *connector)
        return 0;

    edid = drm_get_edid(connector, hdmi->ddc);
+#if HDMI_KEEP_RESOLUTION
+   edid = NULL;
+#endif
    if (edid) {
        hdmi->hdmi_data.sink_is_hdmi = drm_detect_hdmi_monitor(edid);
        hdmi->hdmi_data.sink_has_audio = drm_detect_monitor_audio(edid);
```

hdmi2dsi显示异常

检查过mipi屏端上电时序和初始化命令都是正常的，屏显示会闪屏或者黑屏，可以尝试保持rk628_hdmirx.c文件edid数组的屏参timing不变的情况下，对于V19之前的版本可以增加rk628_config.c文件dst_mode的h-backporch，h-frontporch，h-sync-pulse这几个值，而V19版本则挪到了DTS里。修改方法可以参考以下补丁：

```
diff --git a/kernel/drivers/misc/rk628/rk628_config.c
b/kernel/drivers/misc/rk628/rk628_config.c
index 4c80763..792cbc8 100755
--- a/kernel/drivers/misc/rk628/rk628_config.c
+++ b/kernel/drivers/misc/rk628/rk628_config.c
@@ -22,11 +22,11 @@ static struct drm_display_mode src_mode = {
 };

 static struct drm_display_mode dst_mode = {
-       /* .clock = */ 132000,
+       /* .clock = */ 142000,
        /* .hdisplay = */ 1080,
-       /* .hsync_start = */ 1080 + 15,
-       /* .hsync_end = */ 1080 + 15 + 2,
-       /* .htotal = */ 1080 + 15 + 2 + 30,
+       /* .hsync_start = */ 1080 + 65,
+       /* .hsync_end = */ 1080 + 65 + 12,
+       /* .htotal = */ 1080 + 65 + 12 + 60,
        /* .vdisplay = */ 1920,
        /* .vsync_start = */ 1920 + 15,
        /* .vsync_end = */ 1920 + 15 + 2,
```

或者尝试手动修改mipi clk的大小。默认代码中已经根据TIMING自动计算出了mipi clk，但自动计算值可能与屏幕CLK要求并不匹配，这样可能导致屏幕不亮，因此可以尝试通过下面方法手动修改mipi clk输出大小。以下补丁的修改表示将mipi clk强制设置成1000M（注意mipi是双边沿采样，这个值是双边沿采样后的值，实际示波器测量到的clk值应该是这个值的一半），这或许可以解决黑屏问题。

```
diff --git a/kernel/drivers/misc/rk628/rk628_dsi.c
b/kernel/drivers/misc/rk628/rk628_dsi.c
index f4633ae..49c5fa6 100755
--- a/kernel/drivers/misc/rk628/rk628_dsi.c
+++ b/kernel/drivers/misc/rk628/rk628_dsi.c
@@ -1429,7 +1429,7 @@ void rk628_dsi_enable(struct rk628 *rk628)
 {
        const struct rk628_dsi *dsi = rk628_display_get_dsi0_host();
        const struct rk628_dsi *dsi1 = rk628_display_get_dsi1_host();
-       u32 rate = rk628_dsi_get_lane_rate(dsi);
+       u32 rate = 1000;
        int bus_width;

        rk628_i2c_update_bits(rk628, GRF_SYSTEM_CON0, SW_OUTPUT_MODE_MASK,
```

另外，如果遇到双mipi屏纯色界面下中线位置出现色差，V19之前的版本可以尝试修改行场信号的Hback-porch，Hfront-porch，H-pulse值看是否有改善，V19版本则挪到了DTS里进行配置，另外要注意修改这些值时可能要重新修改pixel-clk或帧场信号porch值，来保证前后帧率一致。

```
diff --git a/drivers/misc/rk628/rk628_config.c
b/drivers/misc/rk628/rk628_config.c
index af78b24..cc5c54b 100755
--- a/drivers/misc/rk628/rk628_config.c
+++ b/drivers/misc/rk628/rk628_config.c
@@ -24,9 +24,9 @@ static struct rk628_display_mode src_mode = {
 static struct rk628_display_mode dst_mode = {
        /* .clock = */ 226000,
        /* .hdisplay = */ 1536,
-       /* .hsync_start = */ 1536 + 90,
-       /* .hsync_end = */ 1536 + 90 + 4,
-       /* .htotal = */ 1536 + 90 + 4 + 90,
+       /* .hsync_start = */ 1536 + 50,
+       /* .hsync_end = */ 1536 + 50 + 14,
+       /* .htotal = */ 1536 + 50 + 14 + 120,
        /* .vdisplay = */ 2048,
        /* .vsync_start = */ 2048 + 25,
        /* .vsync_end = */ 2048 + 25 + 2,
```

DSI或GVI显示有内容但花屏

V19之前的版本可以尝试修改 rk628_config.c文件中 src_mode flags 的正负极性，而V19版本则挪到了DTS里，方法对DSI和GVI均有效。

例如以下修改：

```
--- a/drivers/misc/rk628/rk628_config.c
+++ b/drivers/misc/rk628/rk628_config.c
@@ -104,7 +104,7 @@ static struct drm_display_mode src_mode = {
        /* .vsync_start = */ 1080 + 4,
        /* .vsync_end = */ 1080 + 4 + 5,
        /* .vtotal = */ 1080 + 4 + 5 + 36,

-       /* .flags = */ DRM_MODE_FLAG_NHSYNC | DRM_MODE_FLAG_NVSYNC,
+       /* .flags = */ DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC,
         };
```

提高clock后DSI无法显示

为了提高刷新率，一般会提高CLK，例如下面的修改：

```
--- a/drivers/misc/rk628/rk628_config.c
+++ b/drivers/misc/rk628/rk628_config.c
@@ -22,7 +22,7 @@ static struct rk628_display_mode src_mode = {
 };
 static struct rk628_display_mode dst_mode = {
-       /* .clock = */ 132000,
+       /* .clock = */ 148000,
```

但提高CLK可能导致屏幕闪屏或无法显示，如果出现这种情况，则可以尝试按如下方式调整LANE的速率解决此问题，最后一个参数9也可以微调试试。

```
--- a/drivers/misc/rk628/rk628_dsi.c
+++ b/drivers/misc/rk628/rk628_dsi.c
@@ -1049,7 +1049,7 @@ static u32 rk628_dsi_get_lane_rate(const struct rk628_dsi
*dsi)
        bpp = dsi->bpp;
        lanes = dsi->slave ? dsi->lanes *2 : dsi->lanes;
        lane_rate = mode->clock / 1000 * bpp / lanes;
-       lane_rate = DIV_ROUND_UP(lane_rate * 5, 4);
+       lane_rate = DIV_ROUND_UP(lane_rate * 10, 9);
```

显示有偏移问题

例如，HDMI 做为输入，后端显示有偏移，根据偏移方向，调整屏端的timing 消隐（对于V19及更新版本是对应DTS中，而早期版本则是rk628_config.c文件中的 dst_mode）。同时，要确保SRC和DST的CLOCK保持相同，例如都是148500。

如何操作RK628的GPIO

在需要控制GPIO的位置直接调用GPIO接口即可，该接口包括IOMUX设置，GPIO方向设置和电平设置，例如以下操作将I2C_SDA_HDMI和I2C_SCL_HDMI接口设为GPIO。

```
rk628_misc_gpio_direction_output(rk628, GPIO1_B1, GPIO_REG_HIGH);
rk628_misc_gpio_direction_output(rk628, GPIO1_B2, GPIO_REG_HIGH);
```

若需要恢复则需要调用以下接口。

```
rk628_misc_pinctrl_set_mux(rk628, GPIO1_B1, DDCM0SDARX);
rk628_misc_pinctrl_set_mux(rk628, GPIO1_B2, DDCM0SCLRX);
```

如何降低D-LVDS的功耗

默认代码以LVDS的信号完整性优先，如果需要进一步降低功耗，可以尝试增加以下补丁：

```
--- a/drivers/misc/rk628/rk628_combtxphy.c
+++ b/drivers/misc/rk628/rk628_combtxphy.c
@@ -70,7 +70,7 @@ static void rk628_combtxphy_lvds_power_on(struct rk628 *rk628)
        rk628_i2c_update_bits(rk628, COMBTXPHY_CON7,
                              SW_TX_RTERM_MASK | SW_TX_MODE_MASK |
                              BYPASS_095V_LDO_MASK | TX_COM_VOLT_ADJ_MASK,
-                             SW_TX_RTERM(6) | SW_TX_MODE(3) |
+                             SW_TX_RTERM(7) | SW_TX_MODE(3) |
                              BYPASS_095V_LDO(1) | TX_COM_VOLT_ADJ(0));
```

如何修改输入源的分辨率

从V17-1126版本开始，支持通过配置的源数据分辨率src_mode自动写到hdmirx的edid_init_data里面，省去通过EDID工具手动生成EDID数据（该数据会写入RK628寄存器中）的过程，开发人员需要做的是就是调整src_mode数据即可，例如下面参数。对于V19版本，以下参数全部挪到了DTS中了。

```
static struct rk628_display_mode src_mode = {
    /* .clock = */ 148500,
    /* .hdisplay = */ 1920,
    /* .hsync_start = */ 1920 + 80,
    /* .hsync_end = */ 1920 + 80 + 40,
    /* .htotal = */ 1920 + 80 + 40 + 160,
    /* .vdisplay = */ 1080,
    /* .vsync_start = */ 1080 + 10,
    /* .vsync_end = */ 1080 + 10 + 5,
    /* .vtotal = */ 1080 + 10 + 5 + 30,
    /* .flags = */ DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC,
};
```

HDMIRX 分辨率锁不住

大概是因为前端HDMI输出的分辨率频点不在 RK628 HDMIRX 支持的范围内，可以参考 <HDMIRX 频点支持>

HDMI2GVI 概率性显示异常

1. 如果出现概率性出现条纹的情况，可以dump GVI寄存器，如果无异常，可以关闭主控的DDR变频看看能否改善；
2. 如果出现概率性开机不亮，可以看看手动断开RK628的24M CLK能否改善，如果可以，可以检查下24M CLK的稳定性。

RGB2HDMI不出图问题

1. 先确认HDMI是否被检测到了，例如以下链接状态为正常，如异常则需要排查原因。

```
rk3568_r:/ # cat /sys/class/drm/card0-HDMI-A-1/status
connected
rk3568_r:/ # cat /sys/class/drm/card0-HDMI-A-1/modes
1920x1080
1920x1080
1920x1080
1920x1080
```

```
1920x1080
1920x1080
1920x1080
1920x1080
1280x720
1280x720
1280x720
1280x720
```

2. 确认DCLK是否正常，clk需要和real_clk保持一致，即能分到所需要的频率用于显示。

```
rk3568_r:/ # cat /d/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: ACTIVE
Connector: HDMI-A-1
bus_format[100a]: RGB888_1X24
overlay_mode0 output_mode0 color_space0
Display mode: 1920x1080p60
clk148500 real_clk148500 type48 flag5
H: 1920 2008 2052 2200
V: 1080 1084 1089 1125
Cluster0-win0: ACTIVE
win_id: 4
format: AB24 little-endian (0x34324241)[AFBC] SDR0 color_space0 glb_alpha[0xff]
rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
csc: y2r0 r2y0 csc mode0
zpos: 0
src: pos[0, 0] rect[1920 x 1080]
dst: pos[0, 0] rect[1920 x 1080]
buf0: addr: 0x0000000004f2b000 pitch: 7680 offset: 0

rk3568_r:/ # cat /d/clk/clk_summary | grep dclk
dclk_vop1 0 1 0 12000000 0 0 50000
dclk_vicap 0 0 0 250000000 0 0 50000
dclk_vop2 1 2 0 148500000 0 0 50000
dclk_sdmmc_buffer 0 0 0 99000000 0 0 50000
dclk_ebc 0 0 0 396000000 0 0 50000
dclk_vop0
```

3. 确认主控是否真正送出显示数据，ADB截图方法即可确认。

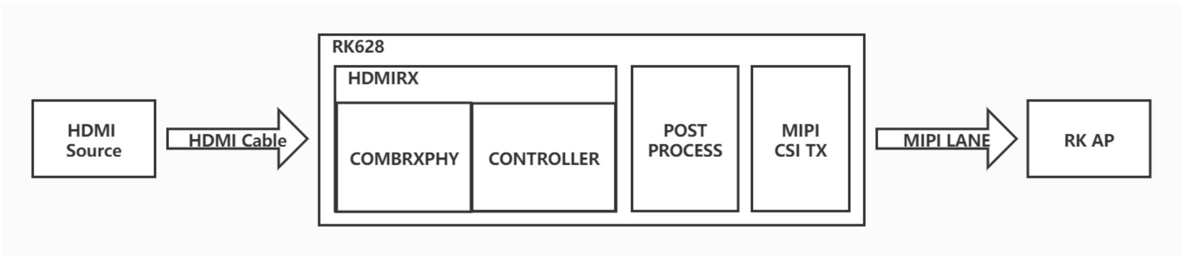
```
adb shell /system/bin/screencap -p /sdcard/screenshot2.png
adb pull /sdcard/screenshot2.png .
```

4. 尝试修改极性flags配置或代码中强制修改。

```
--- a/drivers/misc/rk628/rk628_post_process.c
+++ b/drivers/misc/rk628/rk628_post_process.c
@@ -213,14 +213,14 @@ void rk628_post_process_init(struct rk628 *rk628)

        rk628_cru_clk_set_rate(rk628, CGU_CLK_RX_READ, src->clock * 1000);
        rk628_cru_clk_set_rate(rk628, CGU_SCLK_VOP, dst_rate * 1000);
+#if 0
        if (src->flags & DRM_MODE_FLAG_PVSYNC)
                rk628_i2c_update_bits(rk628, GRF_SYSTEM_CON0,
                                        SW_VSYNC_POL_MASK, SW_VSYNC_POL(1));
        if (src->flags & DRM_MODE_FLAG_PHSYNC)
                rk628_i2c_update_bits(rk628, GRF_SYSTEM_CON0, SW_HSYNC_POL_MASK,
                                        SW_HSYNC_POL(1));
+#endif
        rk628_post_process_scaler_init(rk628, src, dst);
 }
```

Media

驱动介绍



Media 为 RK628 HDMI IN 通路的驱动代码，将RK628D作为类camera设备使用，实现如上功能。

驱动目录结构如下：

```
drivers/media/i2c/rk628
├── Makefile
├── Makefile.orig
├── rk628_bt1120_v4l2.c
├── rk628.c
├── rk628_combrxphy.c
├── rk628_combrxphy.h
├── rk628_combtxphy.c
├── rk628_combtxphy.h
├── rk628.c.orig
├── rk628_cru.c
├── rk628_cru.h
├── rk628_csi.c
├── rk628_csi.h
├── rk628_csi_v4l2.c
├── rk628_dsi.c
├── rk628_dsi.h
├── rk628_gpio.h
├── rk628_grf.h
├── rk628.h
├── rk628_hdmirx.c
├── rk628_hdmirx.h
├── rk628.h.orig
├── rk628_mipi_dphy.h
├── rk628_pinctrl.c
├── rk628_pinctrl.h
└── rk628_v4l2_controls.h
```

目前主要的几个驱动文件及所适用的场景如下：

rk628_csi.c：HDMI2CSI 应用场景，适用于CameraHal1及第三方SOC；

rk628_csi_v4l2.c: HDMI2CSI / HDMI2DSI 应用场景，适用于CameraHal3等基于v4l2框架的平台；

rk628_bt1120_v4l2.c: HDMI2BT1120 应用l场景，适用于CameraHal3等基于v4l2框架的平台；

移植说明

把Media相关的rk628驱动直接拷贝到drivers/media/i2c/rk628/下，然后修改drivers/media/i2c/Makefile，添加rk628的编译及可：

```
obj-y       += rk628/
```

Media的驱动默认都会编译，只需要通过配置dts选择运行对应的驱动。

上层软件包的选择：

查看以下文件选择版本：`hardware/rockchip/camera/SiliconImage/include/isp_cam_api/cam_api/cam_engine_interface.h`

如果是下面宏定义，则选择camerahal1-isplib-2.4这包软件

```
#define CONFIG_SILICONIMAGE_LIBISP_VERSION KERNEL_VERSION(2, 0x04, 0)
```

如果是是下面宏定义，则选择camerahal1-isplib-3.1这包软件

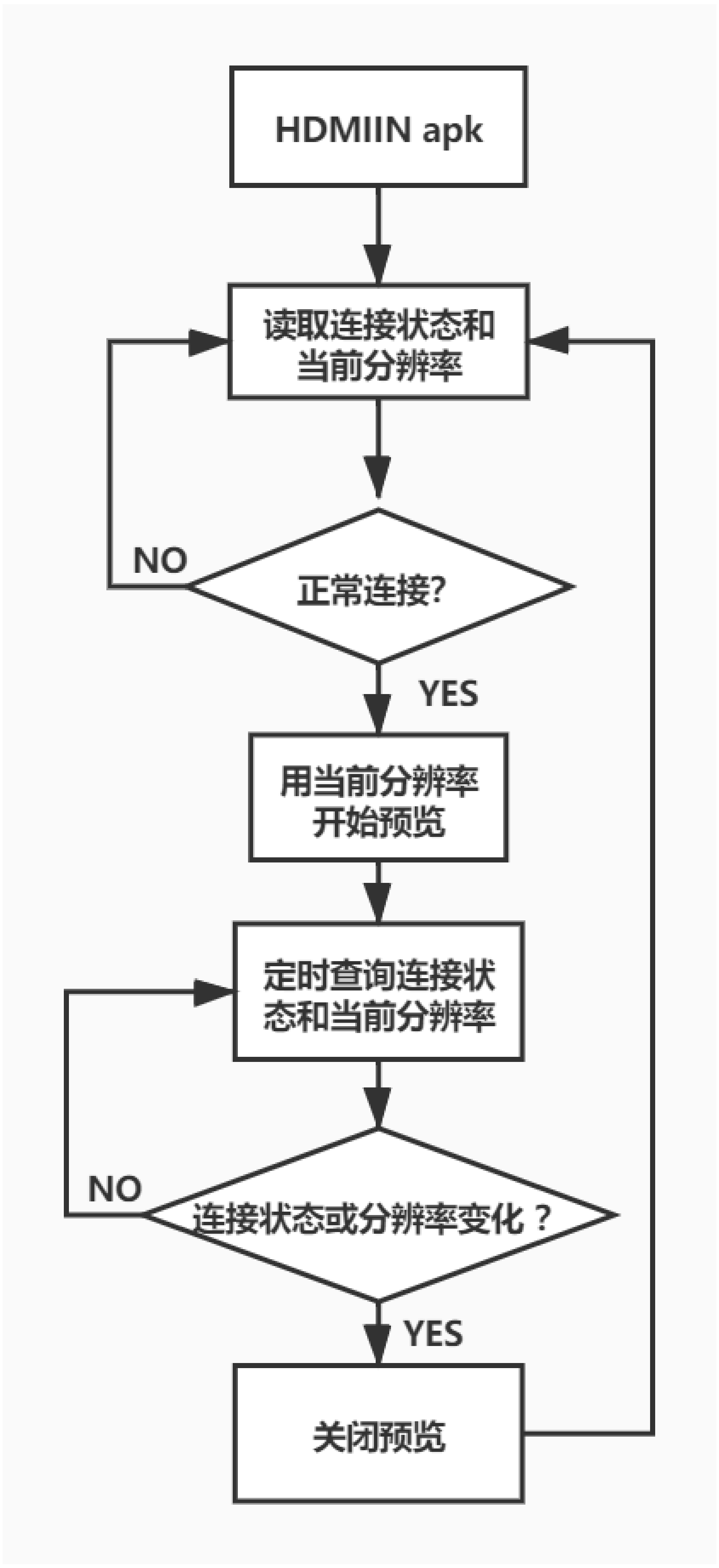
```
#define CONFIG_SILICONIMAGE_LIBISP_VERSION KERNEL_VERSION(3, 0x1, 0)
```

如果文件不存在（ANDROID9.0及更高版本），则选择camerahal3这包软件

HDMI IN VIDEO框架说明

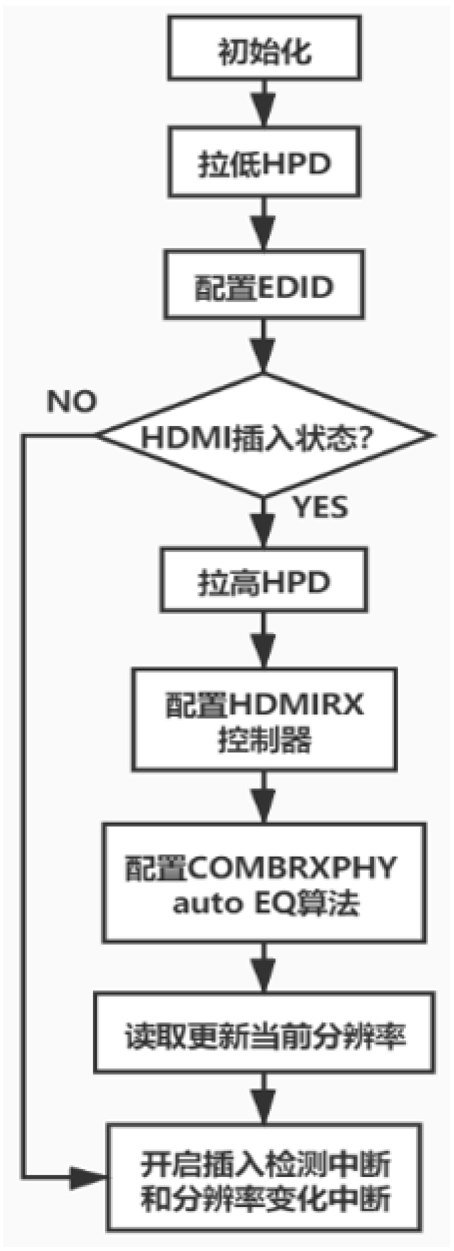
HDMI IN video部分的软件实现方案是将RK628D模拟成一个MIPI SOC camera设备，通过camera框架接收video数据并在APK进行显示，同时基于HDMI IN的应用场景需要，增加HDMI IN热拔插和HDMI IN分辨率自适应支持。

HDMI IN APK工作流程

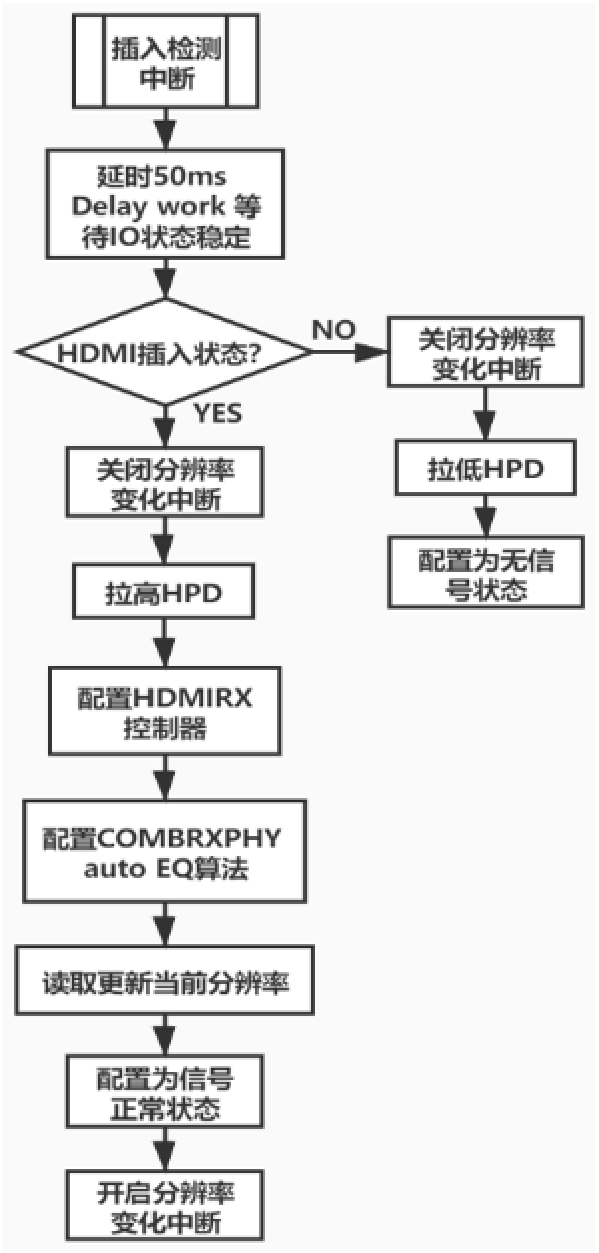


RK628D驱动架构

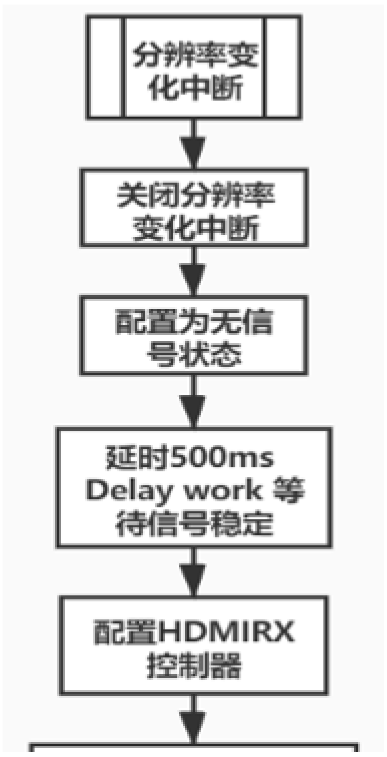
RK628D驱动需要重点关注的主要有三个部分：初始化、热拔插中断处理、分辨率切换中断处理。流程图参考如下：

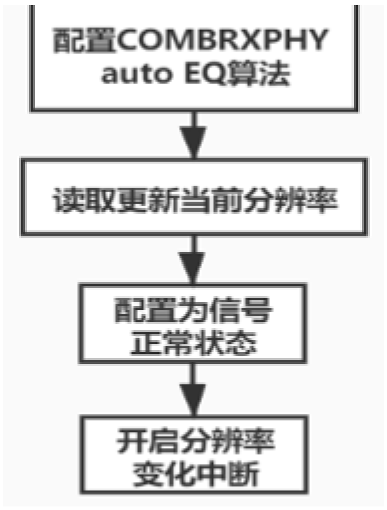


驱动初始配置流程



热拔插中断处理程序





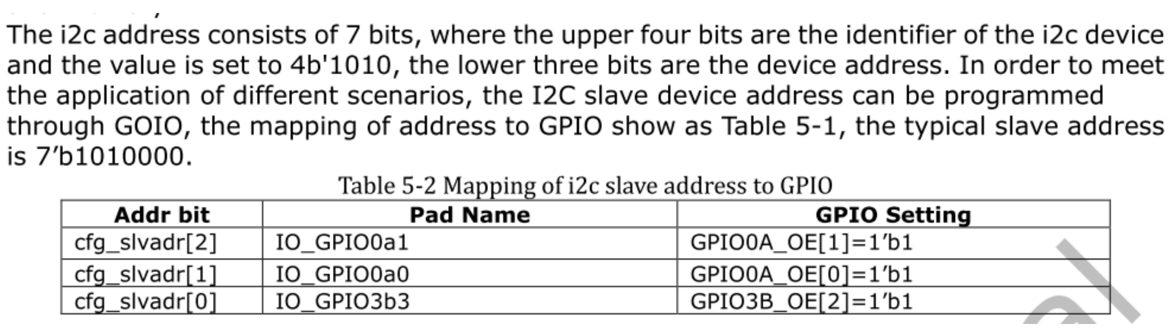
分辨率变化中断处理程序

dts配置说明

RK628节点配置

```
&i2c1 {
        clock-frequency = <400000>;
        status = "okay";
        rk628_csi_v4l2: rk628_csi_v4l2@51 {
                reg = <0x50>;
                compatible = "rockchip,rk628-csi-v4l2";
                interrupt-parent = <&gpio7>;
                interrupts = <11 IRQ_TYPE_LEVEL_HIGH>;
                enable-gpios = <&gpio5 RK_PC3 GPIO_ACTIVE_HIGH>;
                reset-gpios = <&gpio7 RK_PB4 GPIO_ACTIVE_LOW>;
                /* hpd-output-inverted; */
                plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
                power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
                hdcp-enable = <1>;
        };
};
```

- **reg**: I2C地址；RK628D典型的7bit I2C地址为0x50，使用多片RK628D时，可通过RK628D的GPIO改变I2C地址，参考如下：



The i2c address consists of 7 bits, where the upper four bits are the identifier of the i2c device and the value is set to 4b'1010, the lower three bits are the device address. In order to meet the application of different scenarios, the I2C slave device address can be programmed through GOIO, the mapping of address to GPIO show as Table 5-1, the typical slave address is 7'b1010000.

Table 5-2 Mapping of i2c slave address to GPIO

Addr bit	Pad Name	GPIO Setting
cfg_slvadr[2]	IO_GPIO0a1	GPIO0A_OE[1]=1'b1
cfg_slvadr[1]	IO_GPIO0a0	GPIO0A_OE[0]=1'b1
cfg_slvadr[0]	IO_GPIO3b3	GPIO3B_OE[2]=1'b1

- **compatible**:

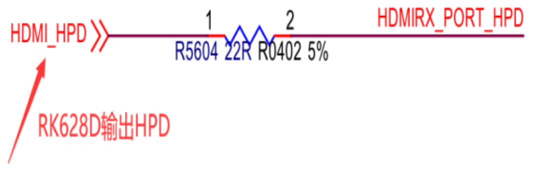
  compatible = "rockchip,rk628-csi" 对应 rk628_csi.c;

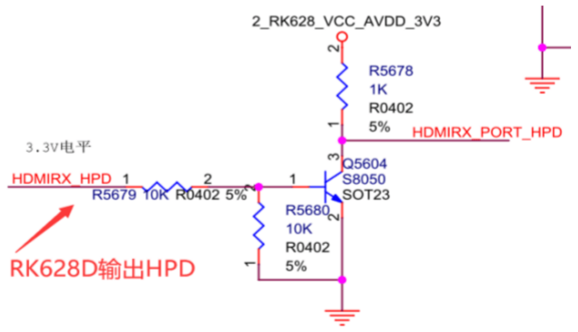
  compatible = "rockchip,rk628-csi-v4l2" 对应 rk628_csi_v4l2.c;

  compatible = "rockchip,rk628-bt1120-v4l2" 对应 rk628_bt1120_v4l2.c;

interrupt-parent/ interrupts：连接RK628D中断的GPIO引脚；
enable-gpios：RK628D供电控制GPIO引脚（若为常供电可不配置）；
reset-gpios：RK628D复位控制GPIO引脚；
hpd-output-inverted：HPD输出取反配置，若HPD输出电平在电路上做了取反，则需要使能此配置项；

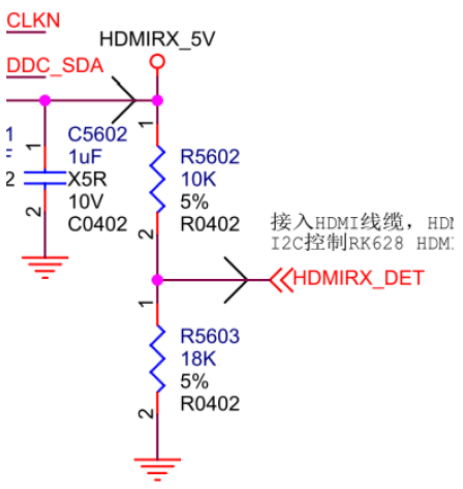
- **◆ HPD未取反设计：** **◆ HPD取反设计：**

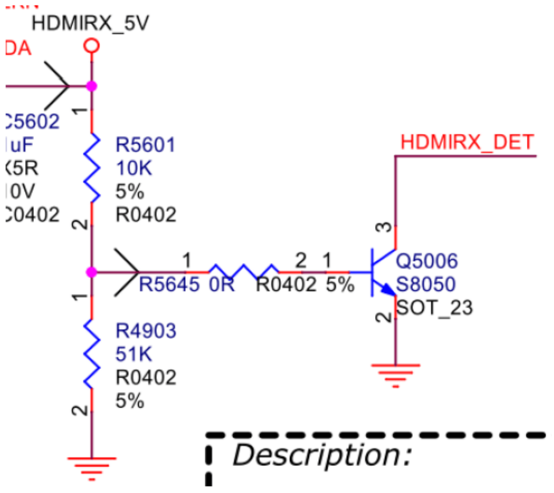




- **plugin-det-gpios**：HDMI插入检测GPIO引脚，注意电路上是否有对电平取反，有效电平需要配置正确；

◆ HDMIRX_DET未取反设计： **◆ HDMIRX_DET取反设计：**





power-gpios：RK主控端MIPI RX电源域供电控制GPIO引脚（若为常供电可不配置）；

hdcp-enable: 使能HDCP功能；

图像接收链路组合

将RK628转换芯片作为类camera设备进行开发，与camera sensor一样需要实现基于V4L2框架的驱动，数据链路配置的方法与MIPI SOC Sensor一致。

HDMI2CSI转换

以RK3288 + CSI_V4L2为例,，rk628 + isp1链路配置

```
&rk628_csi {
    status = "okay";
    /*
     * If the hpd output level is inverted on the circuit,
     * the following configuration needs to be enabled.
     */
    /* hpd-output-inverted; */
    plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
    power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
    rockchip,camera-module-index = <0>;
    rockchip,camera-module-facing = "back";
    rockchip,camera-module-name = "RK628-CSI";
    rockchip,camera-module-lens-name = "NC";

    port {
        hdmiin_out0: endpoint {
            remote-endpoint = <&mipi_in>;
            data-lanes = <1 2 3 4>;
        };
    };
};

&mipi_phy_rx0 {
    status = "okay";

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

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

            mipi_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&hdmiin_out0>;
                data-lanes = <1 2 3 4>;
            };
        };

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

            dphy_rx_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&isp_mipi_in>;
            };
        };
    };
};

&rkisp1 {
    status = "okay";
    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp_mipi_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy_rx_out>;
        };
    };
};
```

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

以RK3568 + CSI_V4L2为例，rk628 + isp2链路配置：

```
&rk628_csi {
    status = "okay";
    /*
     * If the hpd output level is inverted on the circuit,
     * the following configuration needs to be enabled.
     */
    /* hpd-output-inverted; */
    plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
    power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
    rockchip,camera-module-index = <0>;
    rockchip,camera-module-facing = "back";
    rockchip,camera-module-name = "RK628-CSI";
    rockchip,camera-module-lens-name = "NC";

    port {
        hdmiin_out0: endpoint {
            remote-endpoint = <&mipi_in>;
            data-lanes = <1 2 3 4>;
        };
    };
};

&csi2_dphy_hw {
    status = "okay";
};

&csi2_dphy0 {
    status = "okay";

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

            mipi_in: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&hdmiin_out0>;
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&isp0_in>;
            };
        };
    };
};

&rkisp {
    status = "okay";
};

&rkisp_mmu {
    status = "okay";
};

&rkisp_vir0 {
    status = "okay";

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

        isp0_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&csidphy0_out>;
        };
    };
};
```

以RK3568为例，rk628 + vicap链路（RK628 DSI TX接SOC CSI RX的方式）配置：

```
&rk628_csi {
    status = "okay";
    /*
     * If the hpd output level is inverted on the circuit,
     * the following configuration needs to be enabled.
     */
    /* hpd-output-inverted; */
    plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
    power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
    rockchip,camera-module-index = <0>;
    rockchip,camera-module-facing = "back";
    rockchip,camera-module-name = "RK628-CSI";
    rockchip,camera-module-lens-name = "NC";

    port {
        hdmiin_out0: endpoint {
            remote-endpoint = <&mipi_in>;
            data-lanes = <1 2 3 4>;
        };
    };
};

&csi2_dphy_hw {
    status = "okay";
};

&csi2_dphy0 {
    status = "okay";

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

            mipi_in: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&hdmiin_out0>;
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&mipi_csi2_input>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

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

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

            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&csidphy0_out>;
                data-lanes = <1 2 3 4>;
            };
        };

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

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&rkcif {
    status = "okay";
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
            data-lanes = <1 2 3 4>;
        };
    };
};

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

HDMI2DSI转换

HDMI To MIPI CSI 应用场景存在色域空间转换或图像格式的上下采样，图像色彩少量细节可能会有轻微偏差，可以采用HDMI To MIPI DSI替代HDMI To MIPI CSI方案，需要注意的是：

最大支持分辨率1080P60；

目前支持接收MIPI DSI的AP：RK1109、RK1126、RK3566、RK3568；

不支持接收MIPI DSI的AP：RK3288、RK3326、RK3368、RK3399等早期芯片。

CSI 与 DSI 代码兼容，只是dts的compatible做区分：

CSI 为 compatible = "rockchip,rk628-csi-v4l2";

DSI 为 compatible = "rockchip,rk628-dsi-v4l2";

以RK3568为例

```
&i2c2 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&i2c2m1_xfer>;

    rk628_csi: rk628_csi@32 {
        compatible = "rockchip,rk628-dsi-v4l2";
        reg = <0x50>;
        //clocks = <&ext_cam_clk>;
        //clock-names = "xvclk";

        interrupt-parent = <&gpio4>;
        interrupts = <16 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&rk628_irq>;
        //power-gpios = <&gpio0 RK_PD5 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio4 RK_PD2 GPIO_ACTIVE_LOW>;
        plugin-det-gpios = <&gpio0 RK_PD6 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "RK628-CSI";
        rockchip,camera-module-lens-name = "NC";
        port {
            rk628_out: endpoint {
                remote-endpoint = <&hdmi2mipi_in>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&mipi_csi2 {
    status = "okay";

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

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

            mipi_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&csidphy_out>;
                data-lanes = <1 2 3 4>;
            };
        };

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

            mipi_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&rkcif {
    status = "okay";
};

&rkcif_mipi_lvds {
    status = "okay";

    port {
        cif_mipi_in: endpoint {
            remote-endpoint = <&mipi_csi2_output>;
```

```
                data-lanes = <1 2 3 4>;
            };
        };
};

&rkcif_mmu {
        status = "okay";
};

&rkcif_dvp_sditf {
        status = "disabled";
};

&csi2_dphy_hw {
        status = "okay";
};

&csi2_dphy0 {
        status = "okay";

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

                hdmi2mipi_in: endpoint@0 {
                    reg = <1>;
                    remote-endpoint = <&rk628_out>;
                    data-lanes = <1 2 3 4>;
                };
                mipi_in_ucam0: endpoint@1 {
                    reg = <1>;
                    remote-endpoint = <&ov8856_out>;
                    data-lanes = <1 2 3 4>;
                };
                mipi_in_ucam1: endpoint@2 {
                    reg = <2>;
                    remote-endpoint = <&gc8034_out>;
                    data-lanes = <1 2 3 4>;
                };
                mipi_in_ucam2: endpoint@3 {
                    reg = <3>;
                    remote-endpoint = <&ov5695_out>;
                    data-lanes = <1 2>;
                };
            };
            port@1 {
                reg = <1>;
                #address-cells = <1>;
                #size-cells = <0>;

                csidphy_out: endpoint@0 {
                    reg = <0>;
                    remote-endpoint = <&mipi_csi2_input>;
                };
        };
    };
};
```

CIF 需要包含下面的提交:

```
From e925e385a41a857e8e50020b5df9a2d41b166d83 Mon Sep 17 00:00:00 2001
From: Zefa Chen <zefa.chen@rock-chips.com>
Date: Thu, 02 Sep 2021 15:07:13 +0800
Subject: [PATCH] media: rockchip: cif fix errors in rgb24 data format

Signed-off-by: Zefa Chen <zefa.chen@rock-chips.com>
Change-Id: I0766997860f06dc25e604c2ea8425049a987fdc5
---

diff --git a/drivers/media/platform/rockchip/cif/capture.c
b/drivers/media/platform/rockchip/cif/capture.c
index 6c0aa7d..36bd2c9 100644
--- a/drivers/media/platform/rockchip/cif/capture.c
+++ b/drivers/media/platform/rockchip/cif/capture.c
@@ -1785,13 +1785,11 @@
     * needs aligned with :ALIGN(bits_per_pixel * width * 2, 8), to optimize
reading and
     * writing of ddr, aliged with 256
     */
-   if (fmt->fmt_type == CIF_FMT_TYPE_RAW && stream->is_compact) {
+   if (fmt->fmt_type == CIF_FMT_TYPE_RAW && stream->is_compact &&
+       fmt->csi_fmt_val != CSI_WRDDR_TYPE_RGB888) {
        channel->virtual_width = ALIGN(channel->width * fmt->raw_bpp / 8, 256);
    } else {
-       if (fmt->fmt_type == CIF_FMT_TYPE_RAW && fmt->csi_fmt_val !=
CSI_WRDDR_TYPE_RAW8)
-           channel->virtual_width = ALIGN(channel->width * 2, 8);
-       else
-           channel->virtual_width = ALIGN(channel->width * fmt->bpp[0] / 8, 8);
+       channel->virtual_width = ALIGN(channel->width * fmt->bpp[0] / 8, 8);
    }

    if (channel->fmt_val == CSI_WRDDR_TYPE_RGB888)
@@ -3240,7 +3238,8 @@

        if (fmt->fmt_type == CIF_FMT_TYPE_RAW && stream->is_compact &&
            (dev->active_sensor->mbus.type == V4L2_MBUS_CSI2 ||
-            dev->active_sensor->mbus.type == V4L2_MBUS_CCP2)) {
+            dev->active_sensor->mbus.type == V4L2_MBUS_CCP2) &&
+            fmt->csi_fmt_val != CSI_WRDDR_TYPE_RGB888) {
            bpl = ALIGN(width * fmt->raw_bpp / 8, 256);
        } else {
            bpp = rkcif_align_bits_per_pixel(stream, fmt, i);
@@ -4659,13 +4658,11 @@
     * needs aligned with :ALIGN(bits_per_pixel * width * 2, 8), to optimize
reading and
     * writing of ddr, aliged with 256
     */
-   if (fmt->fmt_type == CIF_FMT_TYPE_RAW && stream->is_compact) {
+   if (fmt->fmt_type == CIF_FMT_TYPE_RAW && stream->is_compact &&
+       fmt->csi_fmt_val != CSI_WRDDR_TYPE_RGB888) {
        *crop_vwidth = ALIGN(raw_width * fmt->raw_bpp / 8, 256);
    } else {
-       if (fmt->fmt_type == CIF_FMT_TYPE_RAW)
-           *crop_vwidth = ALIGN(raw_width * 2, 8);
-       else
-           *crop_vwidth = ALIGN(raw_width * fmt->bpp[0] / 8, 8);
+       *crop_vwidth = ALIGN(raw_width * fmt->bpp[0] / 8, 8);
    }

    if (channel->fmt_val == CSI_WRDDR_TYPE_RGB888)
```

rk628-dsi输出默认是command mode，cif接收也需要使用command mode，修改如下:

```
diff --git a/drivers/media/platform/rockchip/cif/capture.c
b/drivers/media/platform/rockchip/cif/capture.c
index 36bd2c99c707..fc4181daed2c 100644
--- a/drivers/media/platform/rockchip/cif/capture.c
+++ b/drivers/media/platform/rockchip/cif/capture.c
@@ -1753,7 +1753,7 @@ static int rkcif_csi_channel_init(struct rkcif_stream
*stream,

        channel->fmt_val = stream->cif_fmt_out->csi_fmt_val;

-       channel->cmd_mode_en = 0; /* default use DSI Video Mode */
+       channel->cmd_mode_en = 1; /* default use DSI Video Mode */

        if (stream->crop_enable) {
                channel->crop_en = 1;
```

CameraHal 需要加上rgb2yuv算子库，若SDK代码版本是R9，则可以直接加上下面的补丁，具体见补丁rk356x_HAL3_support_rgb2yuv_patch.rar

若代码版本是R10或者R11的，在hardware/rockchip/camera下先添加如下修改，再打上上述补丁，

```
diff --git a/psl/rkisp2/RKISP2GraphConfig.cpp b/psl/rkisp2/RKISP2GraphConfig.cpp
index 2a5fa5a..3d2409c 100755
--- a/psl/rkisp2/RKISP2GraphConfig.cpp
+++ b/psl/rkisp2/RKISP2GraphConfig.cpp
@@ -76,6 +76,7 @@ const string MEDIACTL_POSTVIEWNAME = "postview";

 const string MEDIACTL_STATNAME = "rkisp1-statistics";
 const string MEDIACTL_VIDEONAME_CIF = "stream_cif_dvp_id0";
+const string MEDIACTL_VIDEONAME_CIF_MIPI_ID0 = "stream_cif_mipi_id0";

 RKISP2GraphConfig::RKISP2GraphConfig() :
         mManager(nullptr),
@@ -2620,6 +2621,13 @@ status_t RKISP2GraphConfig::getImguMediaCtlConfig(int32_t
cameraId,
                 addLinkParams("rkisp-isp-subdev", 2, "rkisp_mainpath", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
                 addLinkParams("rkisp-isp-subdev", 2, "rkisp_selfpath", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
             }
+       } else if(mipName2.find("mipi") != std::string::npos) {
```

```
+             addLinkParams(mipName, mipSrcPad, mipName2, csiSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             addLinkParams(mipName2, 1, "stream_cif_mipi_id0", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             addLinkParams(mipName2, 2, "stream_cif_mipi_id1", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             addLinkParams(mipName2, 3, "stream_cif_mipi_id2", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             addLinkParams(mipName2, 4, "stream_cif_mipi_id3", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             mSensorLinkedToCIF = true;
        } else {
             addLinkParams(mipName, mipSrcPad, csiName, csiSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
             addLinkParams(csiName, csiSrcPad, IspName, ispSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
@@ -2628,6 +2636,12 @@ status_t RKISP2GraphConfig::getImguMediaCtlConfig(int32_t
cameraId,
             addLinkParams(csiName, 5, "rkisp_rawwr3", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
        }
     }
+    if(mSensorLinkedToCIF){
+       addImguVideoNode(IMGU_NODE_VIDEO, MEDIACTL_VIDEONAME_CIF_MIPI_ID0,
mediaCtlConfig);
+       addFormatParams(MEDIACTL_VIDEONAME_CIF_MIPI_ID0, mCurSensorFormat.width,
mCurSensorFormat.height,
+               0, V4L2_PIX_FMT_NV12, 0, 0, mediaCtlConfig);
+        return OK;
+    }
     // isp input pad format and selection config
     addFormatParams(IspName, ispInWidth, ispInHeight, ispSinkPad, ispInFormat,
0, 0, mediaCtlConfig);
     addSelectionParams(IspName, ispInWidth, ispInHeight, 0, 0,
V4L2_SEL_TGT_CROP, ispSinkPad, mediaCtlConfig);
--
2.17.1
```

请注意，该补丁仅仅适用于RK3568+ANDROID11平台使用。

HDMI2BT1120转换

以RK3568 + BT1120_V4L2为例

```
&i2c2 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&i2c2m1_xfer>;

    rk628_bt1120: rk628_bt1120@32 {
        compatible = "rockchip,rk628-bt1120-v4l2";
        reg = <0x50>;
        //clocks = <&ext_cam_clk>;
        //clock-names = "xvclk";
        interrupt-parent = <&gpio4>;
        interrupts = <16 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&rk628_irq>;
        //power-gpios = <&gpio0 RK_PD5 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio4 RK_PD2 GPIO_ACTIVE_LOW>;
        plugin-det-gpios = <&gpio0 RK_PD6 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "RK628-BT1120";
        rockchip,camera-module-lens-name = "NC";
        port {
            cam_para_out1: endpoint {
                remote-endpoint = <&cif_para_in>;
                bus-width = <16>;
                pclk-sample = <1>;
                /* hsync-active = <1>; */
                /* vsync-active = <0>; */
            };
        };
    };
};

rkcif_dvp {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = </*&cif_clk */&cif_dvp_clk &cif_dvp_bus16 &cif_dvp_bus8>;

    port {
        /* Parallel bus endpoint */
        cif_para_in: endpoint {
            remote-endpoint = <&cam_para_out1>;
        };
    };
};

&rkcif {
    status = "okay";
};

&rkcif_mmu {
    status = "okay";
};

&rkcif_dvp_sditf {
    status = "okay";
};

&csi2_dphy_hw {
    status = "disabled";
};

&csi2_dphy0 {
    status = "disabled";

    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>;
                data-lanes = <1 2 3 4>;
            };
            mipi_in_ucam1: endpoint@2 {
                reg = <2>;
                remote-endpoint = <&gc8034_out>;
                data-lanes = <1 2 3 4>;
            };
            mipi_in_ucam2: endpoint@3 {
                reg = <3>;
                remote-endpoint = <&ov5695_out>;
                data-lanes = <1 2>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&isp0_in>;
            };
        };
    };
};
```

开启HDCP功能

RK628 HDMIRX支持HDCP1.4，HDCP功能默认关闭，需要时打开：

1. dtsRK628节点使能HDCP

```
                 reset-gpios = <&gpio7 RK_PB4 GPIO_ACTIVE_LOW>;
                 plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
                 power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
+                hdcp-enable = <1>;
```

2. 烧录HDCP RX Key

   HDCP Key需要从HDCP协会购买，RK不提供Key，请注意HDCP有区分 TX 和 RK Key，我们需要的是RK Key；

   拿到Key 源文件之后，用 KeyConverter 工具拆分Key，并转成skf后缀的烧录文件，具体请看工具使用说明；

   转成.skf文件之后，使用 RKDevInfoWriteTool 工具进行烧写，工具会对 Key 进行加密，具体请看工具使用说明；

开启 scaler 功能

有些平台不支持 MIPI CSI 4K接收，但是RK628 HDMI IN可以支持4K输入， 所以有客户需求，无论什么分辨率输入，CSI 都按1080P输出，dts只需要添加如下配置：

```
                reset-gpios = <&gpio7 RK_PB4 GPIO_ACTIVE_LOW>;
                plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
                power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
+               scaler-en = <1>;
```

csi 支持 2 lanes

csi 默认配置是 4 lanes：

```
csi->csi_lanes_in_use = USE_4_LANES;
```

如果需要配置成 2 lanes，那么把 csi->csi_lanes_in_use 配置成 2；

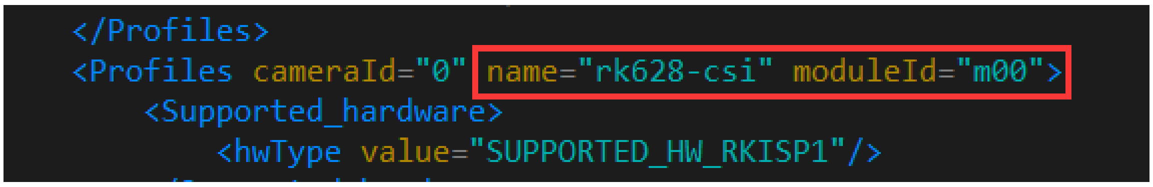
camera3_profiles.xml配置文件说明

camera3_profiles.xml文件对应SDK目录下具体芯片平台的文件：

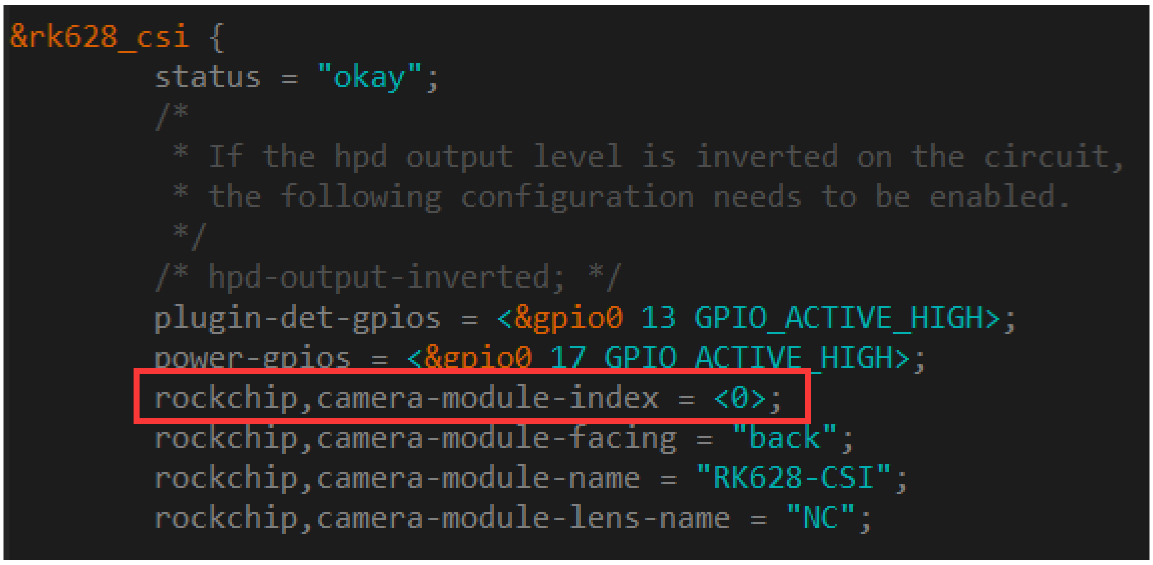
```
hardware/rockchip/camera/etc/camera/camera3_profiles_rk3xxx.xml
```

主要配置注意事项如下，详情可参考SOC Sensor的配置方法：

- **name**：需要与驱动名称一致，有大小写区别；
- **moduleId**：需要与驱动dts中配置的index一致；



```
    </Profiles>
    <Profiles cameraId="0" name="rk628-csi" moduleId="m00">
        <Supported_hardware>
            <hwType value="SUPPORTED_HW_RKISP1"/>
```



```
&rk628_csi {
        status = "okay";
        /*
         * If the hpd output level is inverted on the circuit,
         * the following configuration needs to be enabled.
         */
        /* hpd-output-inverted; */
        plugin-det-gpios = <&gpio0 13 GPIO_ACTIVE_HIGH>;
        power-gpios = <&gpio0 17 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "RK628-CSI";
        rockchip,camera-module-lens-name = "NC";
```

- scaler.availableStreamConfigurations/scaler.availableMinFrameDurations/ scaler.availableStallDurations：需要正确配置驱动支持的分辨率和最小的帧间隔时间，若驱动中需要增加新的分辨率支持，在这里也要相应地增加配置；

```
<scaler.availableStreamConfigurations value="BLOB,3840x2160,OUTPUT,
    BLOB,1920x1080,OUTPUT,
    BLOB,1280x720,OUTPUT,
    BLOB,720x576,OUTPUT,
    BLOB,720x480,OUTPUT,
    YCbCr_420_888,3840x2160,OUTPUT,
    YCbCr_420_888,1920x1080,OUTPUT,
    YCbCr_420_888,1280x720,OUTPUT,
    YCbCr_420_888,720x576,OUTPUT,
    YCbCr_420_888,720x480,OUTPUT,
    IMPLEMENTATION_DEFINED,3840x2160,OUTPUT,
    IMPLEMENTATION_DEFINED,1920x1080,OUTPUT,
    IMPLEMENTATION_DEFINED,1280x720,OUTPUT,
    IMPLEMENTATION_DEFINED,720x576,OUTPUT,
    IMPLEMENTATION_DEFINED,720x480,OUTPUT" />
<scaler.availableMinFrameDurations value="BLOB,3840x2160,33333333,
    BLOB,1920x1080,16666667,
    BLOB,1280x720,16666667,
    BLOB,720x576,20000000,
    BLOB,720x480,16666667,
    YCbCr_420_888,3840x2160,33333333,
    YCbCr_420_888,1920x1080,16666667,
    YCbCr_420_888,1280x720,16666667,
    YCbCr_420_888,720x576,20000000,
    YCbCr_420_888,720x480,16666667,
    IMPLEMENTATION_DEFINED,3840x2160,33333333,
    IMPLEMENTATION_DEFINED,1920x1080,16666667,
    IMPLEMENTATION_DEFINED,1280x720,16666667,
    IMPLEMENTATION_DEFINED,720x576,20000000,
    IMPLEMENTATION_DEFINED,720x480,16666667"/>
<scaler.availableStallDurations value="BLOB,3840x2160,33333333,
    BLOB,1920x1080,16666667,
    BLOB,1280x720,16666667,
    BLOB,720x576,20000000,
    BLOB,720x480,16666667"/>
```

- **sensor.orientation**：图像旋转角度，支持0、90、180、270；

```
<sensor.maxAnalogSensitivity value="2400"/> <!-- HAL
<sensor.orientation value="0"/>
<sensor.profileHueSatMapDimensions value="0,0,0"/>
```

不同芯片平台的接收能力

由于各个芯片平台isp/vicap的性能不同，对图像的最大接收能力也不同。可参考下表：

芯片平台	接收控制器	支持最大分辨率
RK3288/RK3326/RK3368	isp	1920x1080P60
RK3399	isp	3840x2160P30（注：isp需要超频）
RK3566/RK3568	vicap/isp	3840x2160P30

配置isp超频的方法

- RK3399配置PLL_NPLL为650M：

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-vop-clk-set.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399-vop-clk-set.dtsi
@@ -148,7 +148,7 @@
                          <50000000>, <100000000>,
                          <75000000>, <75000000>,
                          <816000000>, <816000000>,
-                         <600000000>, <200000000>,
+                         <650000000>, <200000000>,
                          <800000000>, <150000000>,
                          <75000000>, <37500000>,
                          <300000000>, <100000000>,
```

- 修改rk3399 isp最大支持频率为650M：

```
--- a/drivers/media/platform/rockchip/isp1/dev.c
+++ b/drivers/media/platform/rockchip/isp1/dev.c
@@ -757,7 +757,7 @@ static const unsigned int rk3368_isp_clk_rate[] = {

 /* isp clock adjustment table (MHz) */
 static const unsigned int rk3399_isp_clk_rate[] = {
-       300, 400, 600
+       300, 400, 650
 };

 static struct isp_irqs_data rk1808_isp_irqs[] = {
```

- 如果rk3288 概率性isp报错，isp也可以抬频尝试，一般只需留下最高频即可，该修改仅用于定位问题即可。

```
--- a/drivers/media/platform/rockchip/isp1/dev.c
+++ b/drivers/media/platform/rockchip/isp1/dev.c
@@ -849,7 +849,7 @@ static const unsigned int rk1808_isp_clk_rate[] = {

 /* isp clock adjustment table (MHz) */
static const unsigned int rk3288_isp_clk_rate[] = {
-       150, 384, 500, 594
+       594
 };
```

- 转换芯片驱动中配置isp频率

```
#define RK628_CSI_PIXEL_RATE_HIGH       600000000
...

static int rk628_csi_set_fmt(struct v4l2_subdev *sd,
                struct v4l2_subdev_pad_config *cfg,
                struct v4l2_subdev_format *format)
{
...
        if ((mode->width == 3840) && (mode->height == 2160)) {
                v4l2_dbg(1, debug, sd,
                        "%s res wxh:%dx%d, link freq:%llu, pixrate:%u\n",
                        __func__, mode->width, mode->height,
                        link_freq_menu_items[1], RK628_CSI_PIXEL_RATE_HIGH);
                __v4l2_ctrl_s_ctrl(csi->link_freq, 1);
                __v4l2_ctrl_s_ctrl_int64(csi->pixel_rate,
                        RK628_CSI_PIXEL_RATE_HIGH);
        }
...
}
```

isp驱动中会对配置的频率再加25%的余量，所以在驱动中配置适当的频率RK628_CSI_PIXEL_RATE_HIGH即可。

```
drivers/media/platform/rockchip/isp1/dev.c

static int __isp_pipeline_s_isp_clk(struct rkisp1_pipeline *p)
{
...
        ctrl = v4l2_ctrl_find(sd->ctrl_handler, V4L2_CID_PIXEL_RATE);
        if (!ctrl) {
                v4l2_warn(sd, "No pixel rate control in subdev\n");
                return -EPIPE;
        }

        /* calculate data rate */
        data_rate = v4l2_ctrl_g_ctrl_int64(ctrl) *
                    dev->isp_sdev.in_fmt.bus_width;
        data_rate >>= 3;
        do_div(data_rate, 1000 * 1000);

        /* increase 25% margin */
        data_rate += data_rate >> 2;
...
}
```

配置ISP使用CMA内存的方法

部分平台HDMI IN接收图像数据时，根据实际系统负载，可能会存在带宽不足导致丢帧或MIPI接收异常等问题，参考异常log如下：

```
[ 228.999567] rkisp1: MIPI mis error: 0x00800000
[ 228.999925] rkisp1: CIF_ISP_PIC_SIZE_ERROR (0x00000001)
[ 228.999976] rkisp1: CIF_ISP_PIC_SIZE_ERROR (0x00000001)rkisp1:
CIF_ISP_PIC_SIZE_ERROR (0x00000001)
[ 229.000081] rkisp1: CIF_ISP_PIC_SIZE_ERROR (0x00000001)rkisp1:
CIF_ISP_PIC_SIZE_ERROR (0x00000001)
[ 229.000187] rkisp1: CIF_ISP_PIC_SIZE_ERROR (0x00000001)rkisp1:
CIF_ISP_PIC_SIZE_ERROR (0x00000001)
[ 229.000294] rkisp1: CIF_ISP_PIC_SIZE_ERROR (0x00000001)rkisp1:
CIF_ISP_PIC_SIZE_ERROR (0x00000001)
```

此时需要提高DDR频率，若仍无改善，可给ISP预留使用CMA内存，以改善解决此问题：

- 在rockchip_defconfig配置预留CMA内存128MB

```
CONFIG_CMA=y
CONFIG_CMA_SIZE_MBYTES=128
```

- 在dts配置ISP关闭IOMMU，使用CMA内存

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

EDID的配置方法

RK628D支持EDID配置，目前驱动代码中EDID支持的分辨率为：

3840x2160P30、1920x1080P60、1920x1080P30、1280x720P60、720x576P50、720x480P60

若需要修改分辨率支持，可直接在驱动代码中修改EDID：

```
static u8 edid_init_data[] = {
        0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
        0x49, 0x73, 0x8D, 0x62, 0x00, 0x88, 0x88, 0x88,
        0x08, 0x1E, 0x01, 0x03, 0x80, 0x00, 0x00, 0x78,
        0x0A, 0x0D, 0xC9, 0xA0, 0x57, 0x47, 0x98, 0x27,
        0x12, 0x48, 0x4C, 0x00, 0x00, 0x00, 0x01, 0x01,
        0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
        0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x02, 0x3A,
        0x80, 0x18, 0x71, 0x38, 0x2D, 0x40, 0x58, 0x2C,
        0x45, 0x00, 0xC4, 0x8E, 0x21, 0x00, 0x00, 0x1E,
        0x01, 0x1D, 0x00, 0x72, 0x51, 0xD0, 0x1E, 0x20,
        0x6E, 0x28, 0x55, 0x00, 0xC4, 0x8E, 0x21, 0x00,
        0x00, 0x1E, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x54,
        0x37, 0x34, 0x39, 0x2D, 0x66, 0x48, 0x44, 0x37,
        0x32, 0x30, 0x0A, 0x20, 0x00, 0x00, 0x00, 0xFD,
        0x00, 0x14, 0x78, 0x01, 0xFF, 0x1D, 0x00, 0x0A,
        0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x01, 0x18,

        0x02, 0x03, 0x1A, 0x71, 0x47, 0x5F, 0x90, 0x22,
        0x04, 0x11, 0x02, 0x01, 0x23, 0x09, 0x07, 0x01,
        0x83, 0x01, 0x00, 0x00, 0x65, 0x03, 0x0C, 0x00,
        0x10, 0x00, 0x02, 0x3A, 0x80, 0x18, 0x71, 0x38,
        0x2D, 0x40, 0x58, 0x2C, 0x45, 0x00, 0x20, 0xC2,
        0x31, 0x00, 0x00, 0x1E, 0x01, 0x1D, 0x00, 0x72,
        0x51, 0xD0, 0x1E, 0x20, 0x6E, 0x28, 0x55, 0x00,
        0x20, 0xC2, 0x31, 0x00, 0x00, 0x1E, 0x02, 0x3A,
        0x80, 0xD0, 0x72, 0x38, 0x2D, 0x40, 0x10, 0x2C,
        0x45, 0x80, 0x20, 0xC2, 0x31, 0x00, 0x00, 0x1E,
        0x01, 0x1D, 0x80, 0x18, 0x71, 0x38, 0x2D, 0x40,
        0x58, 0x2C, 0x45, 0x00, 0xC0, 0x6C, 0x00, 0x00,
        0x00, 0x18, 0x01, 0x1D, 0x80, 0x18, 0x71, 0x1C,
        0x16, 0x20, 0x58, 0x2C, 0x25, 0x00, 0xC0, 0x6C,
        0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC1,
```

推荐一个EDID编辑工具：http://www.quantumdata.com/support/downloads/980/release_5_05/R_980mgr_5.05_Win32.msi

HDMI IN APK适配方法

获取和编译APK源码

APK源码提供在SDK目录：

```
RKDocs/common/hdmi-in/apk/rkCamera2_based_on_CameraHal3_V1.3.tar.gz
```

需要将源码拷贝并解压到目录：

```
packages/apps/
```

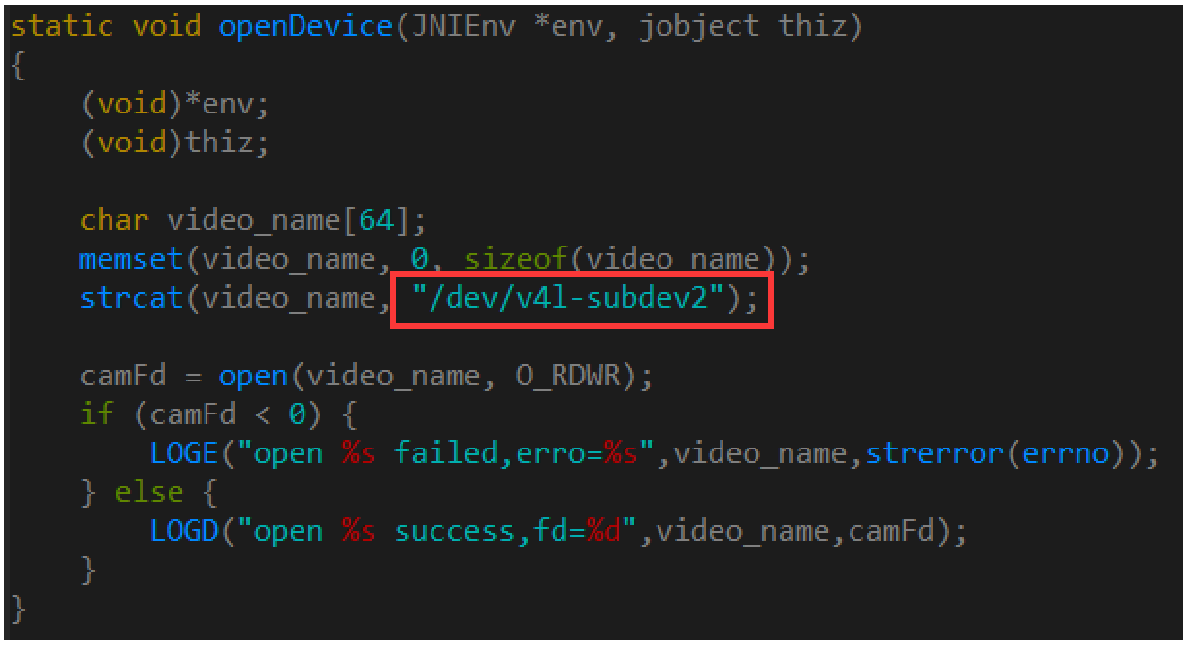
以RK3288平台为例，在device/rockchip/rk3288/目录参照如下修改，增加rkCamera2 APK编译：

```
diff --git a/device.mk b/device.mk
index 6667b5c..96f08f1 100644
--- a/device.mk
+++ b/device.mk
@@ -17,7 +17,8 @@
 PRODUCT_PACKAGES += \
     memtrack.$(TARGET_BOARD_PLATFORM) \
     WallpaperPicker \
-    Launcher3
+    Launcher3 \
+    rkCamera2

 #$_rbox_$_modify_$_zhengyang: add displayd
 PRODUCT_PACKAGES += \
```

APK源码的适配

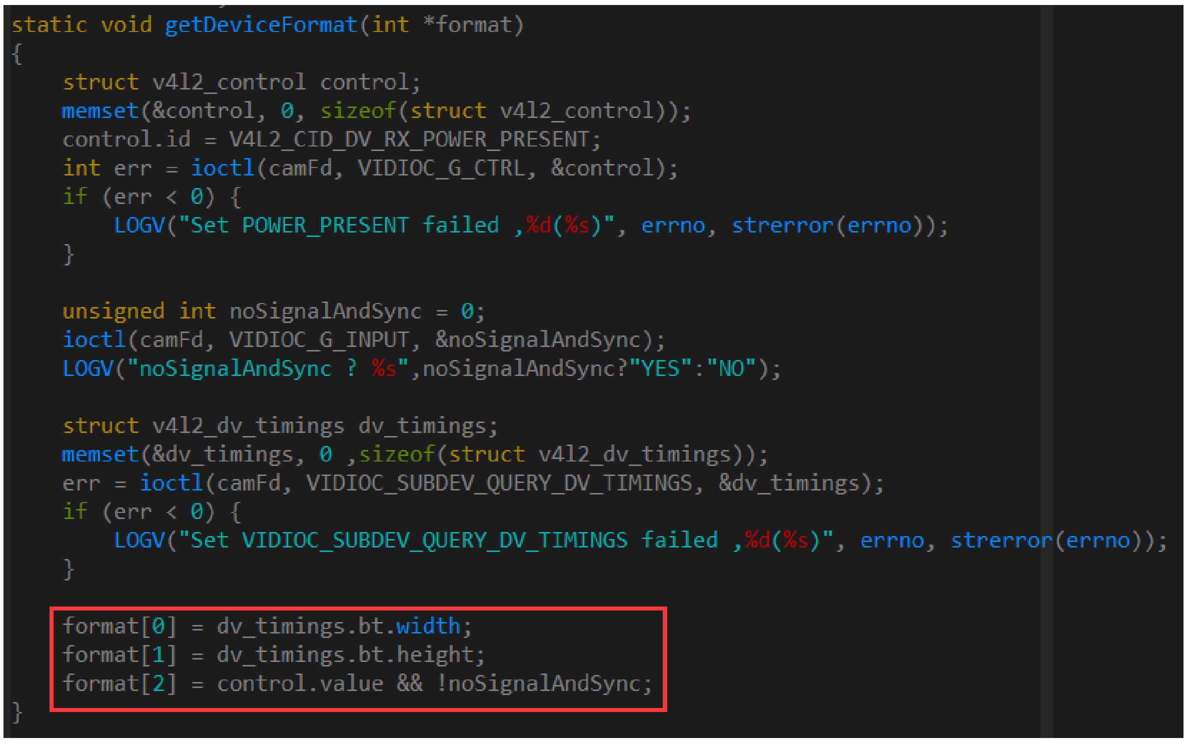
APK通过ioctl的方式访问RK628D设备节点，获取当前的连接状态和分辨率。RK628D设备节点在isp1/isp2/vicap链路上可能会差异。需要根据实际情况修改APK源码，获取设备节点的方法可参考调试命令举例章节。

```
rkCamera2/jni/native.cpp
```



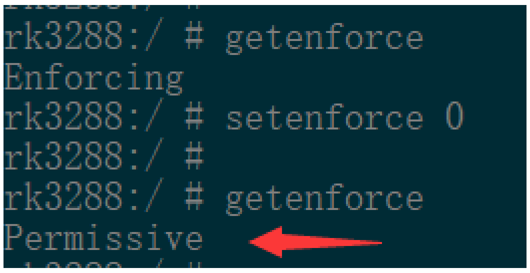
```
static void openDevice(JNIEnv *env, jobject thiz)
{
    (void)*env;
    (void)thiz;

    char video_name[64];
    memset(video_name, 0, sizeof(video_name));
    strcat(video_name, "/dev/v4l-subdev2");

    camFd = open(video_name, O_RDWR);
    if (camFd < 0) {
        LOGE("open %s failed,erro=%s",video_name,strerror(errno));
    } else {
        LOGD("open %s success,fd=%d",video_name,camFd);
    }
}
```

获取当前的连接状态和分辨率代码如下：



```
static void getDeviceFormat(int *format)
{
    struct v4l2_control control;
    memset(&control, 0, sizeof(struct v4l2_control));
    control.id = V4L2_CID_DV_RX_POWER_PRESENT;
    int err = ioctl(camFd, VIDIOC_G_CTRL, &control);
    if (err < 0) {
        LOGV("Set POWER_PRESENT failed ,%d(%s)", errno, strerror(errno));
    }

    unsigned int noSignalAndSync = 0;
    ioctl(camFd, VIDIOC_G_INPUT, &noSignalAndSync);
    LOGV("noSignalAndSync ? %s",noSignalAndSync?"YES":"NO");

    struct v4l2_dv_timings dv_timings;
    memset(&dv_timings, 0 ,sizeof(struct v4l2_dv_timings));
    err = ioctl(camFd, VIDIOC_SUBDEV_QUERY_DV_TIMINGS, &dv_timings);
    if (err < 0) {
        LOGV("Set VIDIOC_SUBDEV_QUERY_DV_TIMINGS failed ,%d(%s)", errno, strerror(errno));
    }

    format[0] = dv_timings.bt.width;
    format[1] = dv_timings.bt.height;
    format[2] = control.value && !noSignalAndSync;
}
```

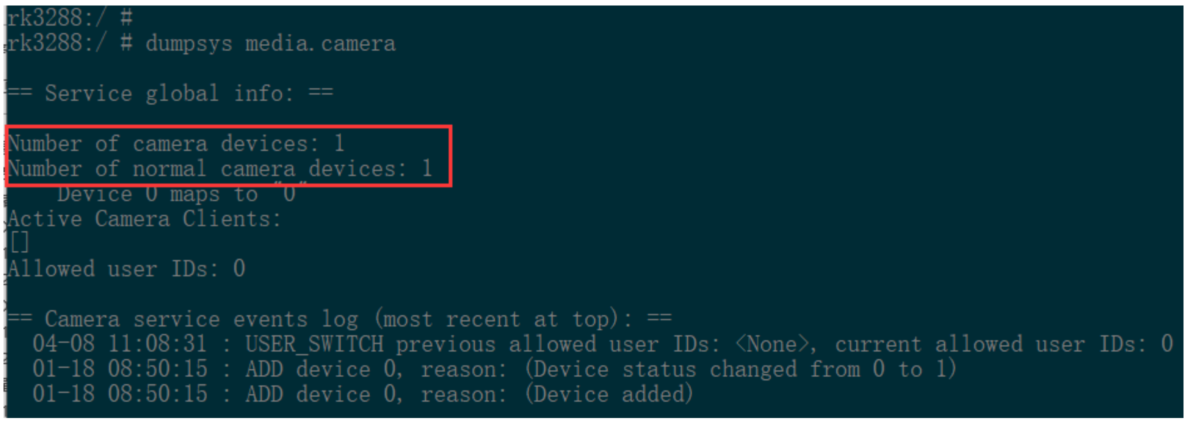
由于在APK访问了设备节点，所以需要确认是否关闭了selinux，可通过getenforce命令查看：



```
rk3288:/ # getenforce
Enforcing
rk3288:/ # setenforce 0
rk3288:/ #
rk3288:/ # getenforce
Permissive
```

APK调试前的准备

APK调试前首先需要先将驱动调试完成，参考驱动调试方法章节。第二步需要确认camera设备是否正确注册到CameraHal，可通过以下命令查看，若未正确注册，则需要检查camera3_profiles.xml配置，参考camera3_profiles.xml配置文件说明章节。



```
rk3288:/ #
rk3288:/ # dumpsys media.camera

== Service global info: ==

Number of camera devices: 1
Number of normal camera devices: 1
    Device 0 maps to "0"
Active Camera Clients:
[]
Allowed user IDs: 0

== Camera service events log (most recent at top): ==
  04-08 11:08:31 : USER_SWITCH previous allowed user IDs: <None>, current allowed user IDs: 0
  01-18 08:50:15 : ADD device 0, reason: (Device status changed from 0 to 1)
  01-18 08:50:15 : ADD device 0, reason: (Device added)
```

驱动调试方法

驱动调试方法与SOC Sensor的调试方法一致，更多信息请参考redmine文档：

https://redmine.rock-chips.com/documents/53

调试工具获取

需要使用media-ctl和v4l2-ctl工具，目前SDK编译固件时会自动拷贝集成，具体是放置在SDK目录：

```
hardware/rockchip/camera/etc/tools/
```

若SDK版本较老，可通过redmine获取：

https://redmine.rock-chips.com/documents/104

将media-ctl和v4l2-ctl用adb push 到设备的 /vendor/bin/ 目录。

调试命令举例

以RK3288 + RK628D 接收1920x1080P分辨率为例，具体调试时需要根据实际情况修改。请注意，以下调试命令需要从前往后逐条输入，否则有可能出现缺少配置导致无法工作的问题。

- 查看链路拓扑结构：

执行命令查看media节点的拓扑结构，根据不同芯片平台具体的链路连接，有可能是 /dev/media0 或 /dev/media1。

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

截取RK628D部分为例，可知RK628D设备为：/dev/v4l-subdev2，识别到HDMI IN分辨率为：1920x1080。

```
...
- entity 8: m00_b_rk628-csi rk628-csi (1 pad, 1 link)
            type V4L2 subdev subtype Sensor
            device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:UYVY2X8/1920x1080]
                -> "rockchip-mipi-dphy-rx":0 [ENABLED]
...
```

- 配置链路连接：

设备复位启动后，链路默认是连接上的。用HDMI IN APK打开再退出时，链路会被断开，查看拓扑结构，没有 [ENABLED] 时才需要重新链接。

```
media-ctl -d /dev/media0 -l \
'"m00_b_rk628-csi rk628-csi":0->"rockchip-mipi-dphy-rx":0 [1]'
media-ctl -d /dev/media0 -l \
'"rockchip-mipi-dphy-rx":1->"rkisp1-isp-subdev":0 [1]'
media-ctl -d /dev/media0 -l '"rkisp1-input-params":0->"rkisp1-isp-subdev":1 [1]'
media-ctl -d /dev/media0 -l '"rkisp1-isp-subdev":2->"rkisp1_mainpath":0 [1]'
media-ctl -d /dev/media0 -l '"rkisp1-isp-subdev":2->"rkisp1_selfpath":0 [1]'
media-ctl -d /dev/media0 -l '"rkisp1-isp-subdev":3->"rkisp1-statistics":0 [1]'
```

- 配置分辨率：

```
media-ctl -d /dev/media0 \
--set-v4l2 '"rkisp1-isp-subdev":0[fmt:UYVY2X8/1920x1080]'
media-ctl -d /dev/media0 \
--set-v4l2 '"rkisp1-isp-subdev":0[crop:(0,0)/1920x1080]'
media-ctl -d /dev/media0 \
--set-v4l2 '"rkisp1-isp-subdev":2[fmt:UYVY2X8/1920x1080]'
media-ctl -d /dev/media0 \
--set-v4l2 '"rkisp1-isp-subdev":2[crop:(0,0)/1920x1080]'
```

- 获取图像数据流：

查看配置结果：

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

获取图像数据流：

```
v4l2-ctl --verbose -d /dev/video0 \
--set-fmt-video=width=1920,height=1080,pixelformat='NV12' \
--stream-mmap=4 \
--set-selection=target=crop,flags=0,top=0,left=0,width=1920,height=1080
```

抓取图像yuv文件，可adb pull上来用7yuv等工具查看：

```
v4l2-ctl --verbose -d /dev/video0 \
--set-fmt-video=width=1920,height=1080,pixelformat='NV12' \
--stream-mmap=4 --stream-skip=5 --stream-count=10 \
--stream-to=/data/rk628_1920x1080.yuv --stream-poll
```

若一切正常，能接收到图像数据，会打出帧率，参考log如下：

```
VIDIOC_QUERYCAP: ok
VIDIOC_G_FMT: ok
VIDIOC_S_FMT: ok
Format Video Capture Multiplanar:
        Width/Height      : 1920/1080
        Pixel Format      : 'NV12'
        Field             : None
        Number of planes  : 1
        Flags             :
        Colorspace        : Default
        Transfer Function : Default
        YCbCr Encoding    : Default
        Quantization      : Full Range
        Plane 0           :
           Bytes per Line : 1920
           Size Image     : 3110400
VIDIOC_G_SELECTION: ok
VIDIOC_S_SELECTION: ok
VIDIOC_REQBUFS: ok
VIDIOC_QUERYBUF: ok
VIDIOC_QUERYBUF: ok
VIDIOC_QBUF: ok
VIDIOC_QUERYBUF: ok
VIDIOC_QBUF: ok
```

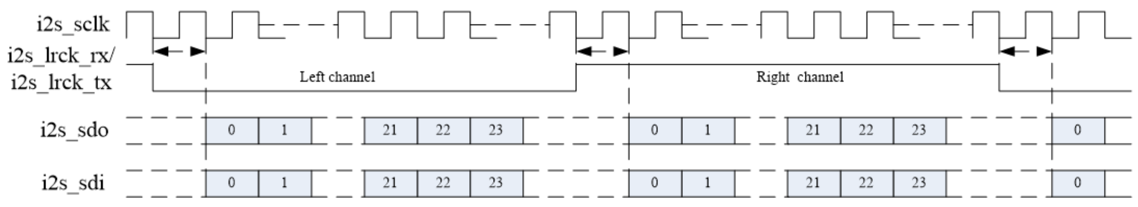
```
VIDIOC_QUERYBUF: ok
VIDIOC_QBUF: ok
VIDIOC_QUERYBUF: ok
VIDIOC_QBUF: ok
VIDIOC_STREAMON: ok
idx: 0 seq:      0 bytesused: 3110400 ts: 131.560377
idx: 1 seq:      1 bytesused: 3110400 ts: 131.577023 delta: 16.646 ms
idx: 2 seq:      2 bytesused: 3110400 ts: 131.593697 delta: 16.674 ms
idx: 3 seq:      3 bytesused: 3110400 ts: 131.610363 delta: 16.666 ms
idx: 0 seq:      4 bytesused: 3110400 ts: 131.627033 delta: 16.670 ms fps: 60.01
idx: 1 seq:      5 bytesused: 3110400 ts: 131.643721 delta: 16.688 ms fps: 59.99
idx: 2 seq:      6 bytesused: 3110400 ts: 131.660390 delta: 16.669 ms fps: 59.99
idx: 3 seq:      7 bytesused: 3110400 ts: 131.677058 delta: 16.668 ms fps: 59.99
```

音频模块介绍

音频部分会根据使用HDMIRX或者HDMITX的使用场景，可以分别设置为输出与输入。在HDMIRX使用场景中，RK628 HDMI接收的音频数据被解包出来后通过I2S接口输出；而HDMITX使用场景中，RK628D I2S是作为数据输入端，接收音频数据，打包后通过HDMITX输出。下面是两种场景的配置方式：

HDMIRX

HDMIRX一般是在驱动加载的时候，就会去初始化音频模块，不再需要额外的配置了。HDMITX播放音频数据时候，音频信号会通过RK628 I2S输出（RK628必须是master）。RK628 I2S 可以直接与支持I2S接口的DAC、SOC连接。I2S格式如下：



I2S normal mode timing format

与SOC连接时候，RK628 I2S不需要额外的配置，可以使用dummy_codec创建一个声卡设备，供系统使用：

```
rk628_dc: rk628-dc {
        compatible = "rockchip,dummy-codec";
        #sound-dai-cells = <0>;
};

hdmiin-sound {
        compatible = "simple-audio-card";
        simple-audio-card,format = "i2s";
        simple-audio-card,name = "rockchip,hdmiin";
        simple-audio-card,bitclock-master = <&dailink0_master>;
        simple-audio-card,frame-master = <&dailink0_master>;
        status = "okay";
        simple-audio-card,cpu {
                sound-dai = <&i2s0>;
        };
        dailink0_master: simple-audio-card,codec {
                sound-dai = <&rk628_dc>;
        };
};
```

与DAC连接的时候，软件上面不要上面的配置了，DAC直接就是能输出模拟信号。但是目前大部分的DAC都是需要有提供一个MCLK，RK628设计的时候，没有预留MCLK，目前使用一个兼容的方法，使用RK628 GPIO1_A0 (H7)作为TEST_CLKO，可以输出128FS的时钟。兼容目前大部分的DAC，有验证过的包括：CS4344、ES7144。使能GPIO1_A0 输出MCLK需要在初始化的时候，调用以下接口，修改如下（请注意自己项目用的是哪个驱动）：

```
--- a/drivers/media/i2c/rk628/rk628_hdmirx.h
+++ b/drivers/media/i2c/rk628/rk628_hdmirx.h
@@ -392,5 +392,5 @@ bool rk628_audio_ctsnints_enabled(HAUDINFO info);
 void rk628_csi_isr_ctsn(HAUDINFO info, u32 pdec_ints);
 void rk628_csi_isr_fifoints(HAUDINFO info, u32 fifo_ints);
 int rk628_is_avi_ready(struct rk628 *rk628, bool avi_rcv_rdy);
+void rk628_hdmirx_audio_set_mclk_output(HAUDINFO info);

--- a/drivers/media/i2c/rk628/rk628_csi.c
+++ b/drivers/media/i2c/rk628/rk628_csi.c
@@ -957,6 +957,9 @@ static void rk628_csi_initial_setup(struct rk628_csi *csi)
 {
        struct v4l2_subdev_edid def_edid;

+       //enable rk628 mclk
+       rk628_hdmirx_audio_set_mclk_output(csi->audio_info);

--- a/drivers/media/i2c/rk628/rk628_csi_v4l2.c
+++ b/drivers/media/i2c/rk628/rk628_csi_v4l2.c
@@ -1096,6 +1096,9 @@ static void rk628_csi_initial_setup(struct v4l2_subdev
*sd)
        struct rk628_csi *csi = to_csi(sd);
        struct v4l2_subdev_edid def_edid;

+       //enable rk628 mclk
+       rk628_hdmirx_audio_set_mclk_output(csi->audio_info);
```

对于RK3288 EVB，使用的是RK3288 I2S与RT5651的I2S2，RK628 I2S与RT5651的I2S2连接。在使用过程中，通过切换RT5651内部的route，达到切换不同通路录音，播放的功能，对应dts配置如下：

```
hdmiin-sound {
        compatible = "rockchip,rockchip-rt5651-rk628-sound";
        rockchip,cpu = <&i2s>;
        rockchip,codec = <&rt5651>;
        status = "okay";
};
```

在默认情况下，i2s只有在预览的时候，才会有输出，如果需要一直输出，DTS需要添加i2s-enable-default配置

```
&i2c4 {
        clock-frequency = <400000>;
        status = "okay";
        rk628_csi_v4l2: rk628_csi_v4l2@50 {
                reg = <0x50>;
                compatible = "rockchip,rk628-csi-v4l2";
                ....
                i2s-enable-default;
                ....
        };
};
```

HDMITX

HDMITX 配置比较简单，只需要配置好HDMI声卡就好了。如下，RK628 I2S与SOC的I2S0连接：

```
hdmi_sound: hdmi-sound {
        compatible = "simple-audio-card";
        simple-audio-card,format = "i2s";
        simple-audio-card,name = "hdmi-sound";
        status = "okay";
        simple-audio-card,cpu {
                sound-dai = <&i2s0>;
        };
        simple-audio-card,codec {
                sound-dai = <&rk628_hdmi>;
        };
};
```

音频常见问题处理

I2S没有输出

```
echo 0x000 0x6000220 > /d/regmap/0-0050-grf/registers   //i2s的IOMUX
echo 0x70 0xfffff55c > /d/regmap/0-0050-grf/registers
echo 0x70 0x155c155c > /d/regmap/0-0050-grf/registers   //设置输出
```

使能打印v4l2_dbg

```
diff --git a/drivers/media/i2c/rk628_csi.c b/drivers/media/i2c/rk628_csi.c
index 5e8e3710a82f..638ac2acc472 100644
--- a/drivers/media/i2c/rk628_csi.c
+++ b/drivers/media/i2c/rk628_csi.c
@@ -34,7 +34,7 @@
 #include <video/videomode.h>
 #include "rk628_csi.h"

-static int debug;
+static int debug = 3;
 module_param(debug, int, 0644);
 MODULE_PARM_DESC(debug, "debug level (0-3)");

diff --git a/include/media/v4l2-common.h b/include/media/v4l2-common.h
index cdc87ec61e54..f159118d4a6b 100644
--- a/include/media/v4l2-common.h
```

```
+++ b/include/media/v4l2-common.h
@@ -82,7 +82,7 @@
 #define v4l2_dbg(level, debug, dev, fmt, arg...)                        \
        do {                                                             \
                if (debug >= (level))                                    \
-                       v4l2_printk(KERN_DEBUG, dev, fmt , ## arg);      \
+                       v4l2_printk(KERN_INFO, dev, fmt , ## arg);       \
        } while (0)

 /**
```

关于应用录音数据杂音问题

这个可能是HDMTX采样率与HAL录音采样率不一致导致的

1. 查看HDMITX采样率

打开内核log: `echo 3 > /sys/module/rk628_csi/parameters/debug`

2. 打开之后，从下面log看出TX的采样率

```
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfe, single offset:0, total offset:-2
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfc, single offset:-2, total offset:-4
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfc, single offset:0, total offset:-4
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfc, single offset:0, total offset:-4
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfa, single offset:-2, total offset:-6
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfc, single offset:2, total offset:-4
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
```

```
m00_b_rk628-csi rk628-csi: rk628_hdmirx_audio_fs: clkrate:1500 tmdsclk:74250000,
n_decoded:6144, cts_decoded:74250, fs_audio:48000
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_FILLSTS1:0xfa, single offset:-2, total offset:-6
m00_b_rk628-csi rk628-csi: rk628_csi_delayed_work_audio:
HDMI_RX_AUD_FIFO_ISTS:0x4
```

3. HAL录音的采样率：

```
getprop vendor.hdmiin.audiorate
```

直接设置IOMUX

```
 static void rk628_hdmirx_initial_setup(struct rk628_hdmirx *hdmirx)
 {
        struct v4l2_subdev_edid def_edid;
@@ -783,6 +937,8 @@ static void rk628_hdmirx_initial_setup(struct rk628_hdmirx
*hdmirx)

        // ddc and hpd pinctrl
        regmap_write(hdmirx->grf, GRF_GPIO1AB_SEL_CON, 0x07000700);
+       //i2s pinctrl
+       regmap_write(hdmirx->grf, GRF_GPIO0AB_SEL_CON, 0x155c155c);

        rk628_hdmirx_controller_reset(hdmirx);
```

关于tmdsclk计算错误

1 bitmask 与时钟对齐

```
git sholsh@rk-intel-1:~/rk-sdk/android11-rk3399/kernel$ git show
007131063748ea69facbf4bbe7aaee71c34fd921
commit 007131063748ea69facbf4bbe7aaee71c34fd921
Author: Shunhua Lan <lsh@rock-chips.com>
Date:   Fri Apr 23 18:43:59 2021 +0800

    media: i2c: rk628csi: fix mask for clkrate and fs audio align to 100

    Signed-off-by: Shunhua Lan <lsh@rock-chips.com>
    Change-Id: I15b290319463f1b41e6908e54caa99ef9c6db4f4

diff --git a/drivers/media/i2c/rk628_csi.c b/drivers/media/i2c/rk628_csi.c
index 89c4a926a985..a970bc621e0e 100644
--- a/drivers/media/i2c/rk628_csi.c
+++ b/drivers/media/i2c/rk628_csi.c
@@ -1064,7 +1064,7 @@ static void rk628_csi_delayed_work_audio(struct
work_struct *work)

        /* fout=128*fs=ftmds*N/CTS */
        regmap_read(csi->hdmirx_regmap, HDMI_RX_HDMI_CKM_RESULT, &clkrate);
-       clkrate = clkrate & 0xfff;
+       clkrate = clkrate & 0xffff;
        /* tmdsclk = (clkrate/1000) * 49500000 */
        tmdsclk = clkrate * (49500000 / 1000);
        regmap_read(csi->hdmirx_regmap, HDMI_RX_PDEC_ACR_CTS, &cts_decoded);
```

```
@@ -1073,6 +1073,8 @@ static void rk628_csi_delayed_work_audio(struct
work_struct *work)
        if (cts_decoded != 0) {
                fs_audio = div_u64((tmdsclk * n_decoded), cts_decoded);
                fs_audio = div_u64(fs_audio, 128);
+               fs_audio = div_u64(fs_audio + 50, 100);
+               fs_audio *= 100;
        }
        v4l2_dbg(2, debug, sd,
                "%s: clkrate:%d tmdsclk:%llu, n_decoded:%d, cts_decoded:%d,
fs_audio:%llu\n",
```

问题log如下：

```
[67.440094] m00_b_rk628-csi rk628-csi:rk628_csi_delayed_work_audio:clkrate:1904
tmdsclk:94248000,n_decoded:6144,cts_decoded:297000,fs_audio:15232
```

4K频，计算的tmdsclk 频率应该是297M

tmdsclk = clkrate * (49500000 / 1000);

1904 ---> 0x770

6000 ---> 0x1770

tmdsclk 6*49500000 = 297000000

6 设置GPIO输出test clk

快捷设置

```
grf: gpio0 相关iomux切换，配置为IO口：
echo 0x70 0x0fff0000 > registers

gpio0 配置为输出口, 1-0051-rk628-pinctrl:
echo 0xd0008 0x00EC00EC > registers

grf，选择在哪一级拉出信号：
HDMI_RX:
echo 0x300 0x11 > registers
ASYNC_IN:
echo 0x300 0x14 > registers
ASYNC_OUT:
echo 0x300 0x15 > registers
SCALER:
echo 0x300 0x16 > registers
```

rk356x 的IOMUX特殊处理

对于RK356X 因为每路I2S可以复用多组pin，需要额外在配置GRF寄存器GRF_IOFUNC_SEL4：

Bit	Attr	Reset Value	Description
14	RW	0x0	i2s3_iomux_sel I2S3 IO mux selection 1'b0:M0 mux solution 1'b1:M1 mux solution
12	RW	0x0	i2s2_iomux_sel I2S2 IO mux selection 1'b0:M0 mux solution 1'b1:M1 mux solution
11:10	RW	0x0	i2s1_iomux_sel I2S1 IO mux selection 2'b00:M0 mux solution 2'b01:M1 mux solution 2'b10:M2 mux solution 2'b11: Reserved

pinctrl驱动里面，会根据用户选择的mclk来配置GRF对应的bit，代码如下：

```
static struct rockchip_mux_route_data rk3568_mux_route_data[] = {
    ....
    RK_MUXROUTE_GRF(1, RK_PA2, 1, 0x0310, WRITE_MASK_VAL(11, 10, 0)), /* I2S1 IO mux M0 */
    RK_MUXROUTE_GRF(3, RK_PC6, 4, 0x0310, WRITE_MASK_VAL(11, 10, 1)), /* I2S1 IO mux M1 */
    RK_MUXROUTE_GRF(2, RK_PD0, 5, 0x0310, WRITE_MASK_VAL(11, 10, 2)), /* I2S1 IO mux M2 */
    RK_MUXROUTE_GRF(2, RK_PC1, 1, 0x0310, WRITE_MASK_VAL(12, 12, 0)), /* I2S2 IO mux M0 */
    RK_MUXROUTE_GRF(4, RK_PB6, 5, 0x0310, WRITE_MASK_VAL(12, 12, 1)), /* I2S2 IO mux M1 */
    RK_MUXROUTE_GRF(3, RK_PA4, 4, 0x0310, WRITE_MASK_VAL(14, 14, 0)), /* I2S3 IO mux M0 */
    RK_MUXROUTE_GRF(4, RK_PC4, 5, 0x0310, WRITE_MASK_VAL(14, 14, 1)), /* I2S3 IO mux M1 */
    ....
}
```

对于类似rk628这类不需要mclk应用，因为没有配置mclk，所以边不会设置对应的bit，这里需要手动添加下，一般i2s都会使用sclk的，这里把sclk加上去（后续新的RK356X SDK会增加此补丁）：

```
diff --git a/drivers/pinctrl/pinctrl-rockchip.c b/drivers/pinctrl/pinctrl-rockchip.c
index 871c49ef0c2a..3d10e6c2ed27 100644
--- a/drivers/pinctrl/pinctrl-rockchip.c
+++ b/drivers/pinctrl/pinctrl-rockchip.c
@@ -1046,13 +1046,26 @@ static struct rockchip_mux_route_data rk3568_mux_route_data[] = {
        RK_MUXROUTE_GRF(2, RK_PB0, 3, 0x0310, WRITE_MASK_VAL(9, 8, 0)), /* UART9 IO mux M0 */
        RK_MUXROUTE_GRF(4, RK_PC5, 4, 0x0310, WRITE_MASK_VAL(9, 8, 1)), /* UART9 IO mux M1 */
        RK_MUXROUTE_GRF(4, RK_PA4, 4, 0x0310, WRITE_MASK_VAL(9, 8, 2)), /* UART9 IO mux M2 */
+
        RK_MUXROUTE_GRF(1, RK_PA2, 1, 0x0310, WRITE_MASK_VAL(11, 10, 0)), /*
I2S1 IO mux M0 */
        RK_MUXROUTE_GRF(3, RK_PC6, 4, 0x0310, WRITE_MASK_VAL(11, 10, 1)), /*
I2S1 IO mux M1 */
        RK_MUXROUTE_GRF(2, RK_PD0, 5, 0x0310, WRITE_MASK_VAL(11, 10, 2)), /*
I2S1 IO mux M2 */
+       RK_MUXROUTE_GRF(1, RK_PA3, 1, 0x0310, WRITE_MASK_VAL(11, 10, 0)), /*
I2S1 IO mux sclk tx M0 */
+       RK_MUXROUTE_GRF(1, RK_PA4, 1, 0x0310, WRITE_MASK_VAL(11, 10, 0)), /*
I2S1 IO mux sclk rx M0 */
+       RK_MUXROUTE_GRF(3, RK_PC7, 4, 0x0310, WRITE_MASK_VAL(11, 10, 1)), /*
I2S1 IO mux sclk tx M1 */
+       RK_MUXROUTE_GRF(4, RK_PA6, 5, 0x0310, WRITE_MASK_VAL(11, 10, 1)), /*
I2S1 IO mux sclk rx M1 */
+       RK_MUXROUTE_GRF(2, RK_PD1, 5, 0x0310, WRITE_MASK_VAL(11, 10, 2)), /*
I2S1 IO mux sclk tx M2 */
+       RK_MUXROUTE_GRF(3, RK_PC3, 5, 0x0310, WRITE_MASK_VAL(11, 10, 2)), /*
I2S1 IO mux sclk rx M2 */
        RK_MUXROUTE_GRF(2, RK_PC1, 1, 0x0310, WRITE_MASK_VAL(12, 12, 0)), /*
I2S2 IO mux M0 */
        RK_MUXROUTE_GRF(4, RK_PB6, 5, 0x0310, WRITE_MASK_VAL(12, 12, 1)), /*
I2S2 IO mux M1 */
+       RK_MUXROUTE_GRF(2, RK_PC2, 1, 0x0310, WRITE_MASK_VAL(12, 12, 0)), /*
I2S2 IO mux sclk tx M0 */
+       RK_MUXROUTE_GRF(2, RK_PB7, 1, 0x0310, WRITE_MASK_VAL(12, 12, 0)), /*
I2S2 IO mux sclk rx M0 */
+       RK_MUXROUTE_GRF(4, RK_PB7, 4, 0x0310, WRITE_MASK_VAL(12, 12, 1)), /*
I2S1 IO mux sclk tx M1 */
+       RK_MUXROUTE_GRF(4, RK_PC1, 5, 0x0310, WRITE_MASK_VAL(12, 12, 1)), /*
I2S1 IO mux sclk rx M1 */
        RK_MUXROUTE_GRF(3, RK_PA2, 4, 0x0310, WRITE_MASK_VAL(14, 14, 0)), /*
I2S3 IO mux M0 */
        RK_MUXROUTE_GRF(4, RK_PC2, 5, 0x0310, WRITE_MASK_VAL(14, 14, 1)), /*
I2S3 IO mux M1 */
+       RK_MUXROUTE_GRF(3, RK_PA3, 4, 0x0310, WRITE_MASK_VAL(14, 14, 0)), /*
I2S3 IO mux sclk M0 */
+       RK_MUXROUTE_GRF(4, RK_PC3, 5, 0x0310, WRITE_MASK_VAL(14, 14, 1)), /*
I2S3 IO mux sclk M1 */
        RK_MUXROUTE_GRF(1, RK_PA4, 3, 0x0314, WRITE_MASK_VAL(1, 0, 0)), /* PDM
IO mux M0 */
        RK_MUXROUTE_GRF(1, RK_PA6, 3, 0x0314, WRITE_MASK_VAL(1, 0, 0)), /* PDM
IO mux M0 */
        RK_MUXROUTE_GRF(3, RK_PD6, 5, 0x0314, WRITE_MASK_VAL(1, 0, 1)), /* PDM
IO mux M1 */
```

RK3399的LRCK的特殊处理

如果方向配置错误可能导致声卡打开后收不到任何数据，LOG大概如下（节点路径需要根据实际情况修改）：

```
console:/ # cat /proc/asound/cards
 0 [rockchiphdmi   ]: rockchip_hdmi - rockchip,hdmi
                      rockchip,hdmi
 1 [rockchiphdmiin ]: rockchip_hdmiin - rockchip,hdmiin
                      rockchip,hdmiin

console:/ # cat /proc/asound/card1/pcm0c/sub0/status
```

```
state: RUNNING
owner_pid   : 1415
trigger_time: 1632716135.151992694
tstamp      : 0.000000000
delay       : 0
avail       : 0
avail_max   : 0
-----
hw_ptr      : 0
appl_ptr    : 0
```

正常情况应该类似下面的信息：

```
console:/ # cat /proc/asound/card1/pcm0c/sub0/status
state: RUNNING
owner_pid   : 1421
trigger_time: 1632892868.412331044
tstamp      : 1632892881.181113577
delay       : 232
avail       : 232
avail_max   : 264
-----
hw_ptr      : 543736
appl_ptr    : 543504
```

这是因为RK628做主，RK3399当从设备，需要根据电路连接方式进行配置，如果LRCK_TX连接RK628，则“rockchip,clk-trcm”配置如下：

```
+&i2s0 {
+        rockchip,clk-trcm = <1>;
+        status = "okay";
+};

console:/ # io -4 0xff880008
ff880008:  18071f1f
```

如果LRCK_RX连接RK628，则“rockchip,clk-trcm”配置如下：

```
+&i2s0 {
+        rockchip,clk-trcm = <2>;
+        status = "okay";
+};
正常寄存器如下：
console:/ # io -4 0xff880008
ff880008:  28071f1f
```

I2S_CKR寄存器解释如下bit29:28解释如下：

```
Tx and Rx Common Use
2'b00/2'b11:tx_lrck/rx_lrck are used as synchronous signal for TX
/RX respectively.
2'b01:only tx_lrck is used as synchronous signal for TX and RX.
2'b10:only rx_lrck is used as synchronous signal for TX and RX.
```

RK628代码可以更新到v15-210926或更新，然后参考这个DTS的音频配置
arch/arm64/boot/dts/rockchip/rk3399-evb-ind-lpddr4-rk628-hdmi2csi-v4l2-avb.dts

HDMI-IN声卡选择错误

APK打开时会去根据声卡名称去匹配，早期代码可能存在名称匹配错误问题，从而导致HDMI-IN声卡无法打开的问题，错误信息如下：

```
09-28 07:03:13.341   261  1386 D AudioHardwareTiny: card0 id:rockchiphdmi
09-28 07:03:13.341   261  1386 D AudioHardwareTiny: SPEAKER card, got
card=0,device=0
09-28 07:03:13.341   261  1386 D AudioHardwareTiny: HDMI card, got
card=0,device=0
09-28 07:03:13.341   261  1386 D AudioHardwareTiny: SPDIF card, got
card=0,device=0
09-28 07:03:13.341   261  1386 D AudioHardwareTiny: BT card, got card=0,device=0
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: card1 id:rockchiphdmiin
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: SPEAKER card, got
card=1,device=0
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: HDMI card, got
card=1,device=0
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: SPDIF card, got
card=1,device=0
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: BT card, got card=1,device=0
09-28 07:03:13.342   261  1386 D AudioHardwareTiny: No exist
proc/asound/card2/id, break and finish parsing
09-28 07:03:13.343   261  1386 D AudioHardwareTiny: dump out device info
09-28 07:03:13.343   261  1386 D AudioHardwareTiny: dev_info SPEAKER  card=1,
device:0
09-28 07:03:13.343   261  1386 D AudioHardwareTiny: dev_info HDMI  card=1,
device:0
09-28 07:03:13.343   261  1386 D AudioHardwareTiny: dev_info SPDIF  card=1,
device:0
09-28 07:03:13.343   261  1386 D AudioHardwareTiny: dev_info BT  card=1,
device:0
```

HDMI card, got card=1,device=0 信息与实际不符，因此声卡无法打开。

需要更新以下补丁解决或者将RK628代码更新到V15-210926或更新版本解决。

```
hardware/rockchip/audio
commit 1b51a62fc62e9d63850e4e5a39f1e3cff0aa9b88 (HEAD)
Author: Shunhua Lan <lsh@rock-chips.com>
Date:   Tue Sep 28 17:25:44 2021 +0800

    [audio hal] use levenshtein distance for sound card matching

    Signed-off-by: Shunhua Lan <lsh@rock-chips.com>
    Change-Id: I16f1692715d710a0693ae74875b0272669a04ba2
```

其他音频文档补充

如果RK628直接连SOC，可以参考文档“RK628-DIRECT-TO-SOC.pdf”。

如果RK628通过AUDIO CODEC再连SOC，可以参考文档“RK628-RT5640-AUDIO-CONFIG.pdf”。

和HDMI-IN APK相关的音频问题，可以参考文档“RK628-HDMIIN-APP-AUDIO.pdf”。

若其他文档与这个文档有不符的地方或者冲突，以这个文档为准。

常见问题排查方法

打开log开关

可通过如下命令打开RK628D的log开关，然后通过demsg命令获取kernel log：

```
echo 1 > /sys/module/rk628_csi/parameters/debug
```

若要抓取上电开机过程的log，建议直接修改代码并重新编译烧写kernel相关部分：

```
diff --git a/drivers/media/i2c/rk628_csi.c b/drivers/media/i2c/rk628_csi.c
index c763a9558169..bd7f3effb45a 100644
--- a/drivers/media/i2c/rk628_csi_v4l2.c
+++ b/drivers/media/i2c/rk628_csi_v4l2.c
@@ -34,7 +34,7 @@
 #include <video/videomode.h>
 #include "rk628_csi.h"

-static int debug;
+static int debug = 1;
 module_param(debug, int, 0644);
 MODULE_PARM_DESC(debug, "debug level (0-3)");

diff --git a/include/media/v4l2-common.h b/include/media/v4l2-common.h
index 1cc0c5ba16b3..e74f3a85f0b8 100644
--- a/include/media/v4l2-common.h
+++ b/include/media/v4l2-common.h
@@ -75,7 +75,7 @@
 #define v4l2_dbg(level, debug, dev, fmt, arg...)                   \
        do {                                                        \
                if (debug >= (level))                               \
-                       v4l2_printk(KERN_DEBUG, dev, fmt , ## arg);     \
+                       v4l2_printk(KERN_INFO, dev, fmt , ## arg);      \
        } while (0)
```

寄存器读写

RK628寄存器调试节点如下，当前示例RK628接在I2C1，地址为0x51：

```
rk3288:/ # ls -l /d/regmap/
total 0
drwxr-xr-x 2 root root 0 1970-01-01 00:00 0-001b
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-adapter
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-combrxphy
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-combtxphy
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-cru
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-csi
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-grf
drwxr-xr-x 2 root root 0 1970-01-01 00:00 1-0051-hdmirx
drwxr-xr-x 2 root root 0 1970-01-01 00:00 2-001a
drwxr-xr-x 2 root root 0 1970-01-01 00:00 ff890000.i2s
drwxr-xr-x 2 root root 0 1970-01-01 00:00 ff96c000.video-phy
```

寄存器节点默认只读，如果需要寄存器可写，需要添加如下修改：

```
diff --git a/drivers/base/regmap/regmap-debugfs.c b/drivers/base/regmap/regmap-
debugfs.c
index 3f0a7e262d69..b819645edd84 100644
--- a/drivers/base/regmap/regmap-debugfs.c
+++ b/drivers/base/regmap/regmap-debugfs.c
@@ -259,7 +259,7 @@ static ssize_t regmap_map_read_file(struct file *file, char
__user *user_buf,
                                   count, ppos);
 }

-#undef REGMAP_ALLOW_WRITE_DEBUGFS
+#define  REGMAP_ALLOW_WRITE_DEBUGFS
 #ifdef REGMAP_ALLOW_WRITE_DEBUGFS
 /*
  * This can be dangerous especially when we have clients such as
```

读寄存器：

```
rk3288:/ # cat /d/regmap/1-0051-cru/registers
c0000: 00001063
c0004: 00001442
c0008: 00000000
c000c: 00000007
c0010: 00007f00
c0020: 00006010
c0024: 00000581
c0028: 00ef348b
c002c: 00000007
c0030: 00007f00
...
```

写寄存器：

```
rk3288:/ # echo 0x000 0xffffffff > /d/regmap/1-0051-grf/registers
```

clk det 异常问题

COMBRXPHY没有检测到信号，有可能是HDMI插入有效电平或是HPD有效电平配置出错，导致输入源HDMI信号没有正常输入。需要检查rk628_csi节点下plugin-det-gpios和hpd-output-inverted配置项，同时可以用万用表测试HPD电平状态。

注意异常时会进行retry，最多到retry 2。若retry后能够正常，则可认为信号正常。

```
rk628-csi-v4l2 1-0051: clk det over cnt:10, reg_0x6654:0x403f0000
rk628-csi-v4l2 1-0051: |d2_p| level detection anomaly
rk628-csi-v4l2 1-0051: clock detected failed, cfg resistance manual!
rk628-csi-v4l2 1-0051: err, clk not stable, reg_0x6630:0x87000d,
reg_0x6608:0x110100
m00_b_rk628-csi 1-0051: hdmi rxphy power on failed
```

如上log，可以看出是data2_p 电压检测异常，可能是没信号或是电平幅值比较低。

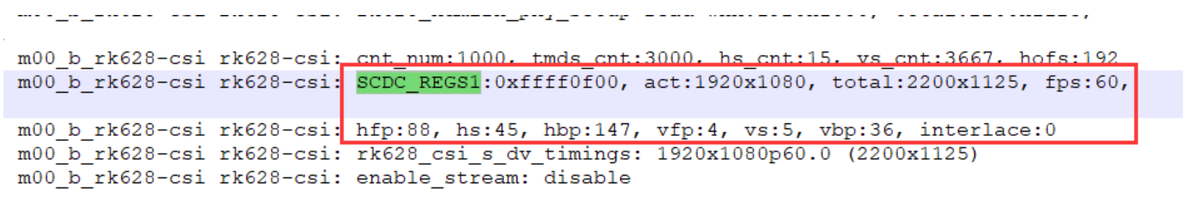
```
rk628-csi-v4l2 1-0051: clk det over cnt:10, reg_0x6654:0x403f0000
rk628-csi-v4l2 1-0051: Clock detection anomaly
rk628-csi-v4l2 1-0051: clock detected failed, cfg resistance manual!
rk628-csi-v4l2 1-0051: err, clk not stable, reg_0x6630:0x87000d,
reg_0x6608:0x110100
m00_b_rk628-csi 1-0051: hdmi rxphy power on failed
```

如上log，表示频点检测异常，可能是频点不在RK628的支持范围内，目前 HDMIRX 支持以下频点：

```
25175, 27000, 33750, 40000, 59400, 65000, 68250, 74250,
75000, 83500, 85500, 88750, 928125, 101000, 102250, 108000,
118800, 119000, 135000, 148500, 150000, 162000, 165000, 297000
```

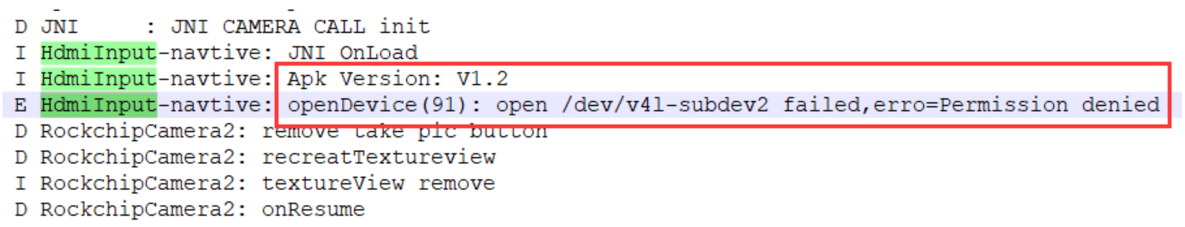
HDMI RX正常的判断方法

COMBRXPHY正常锁定后，HDMI RX控制器能正常解析到Timing，Timing是计算出来，可能会有一些小误差，一般可能会差1。详细的Timing可参考CEA标准文档。



```
m00_b_rk628-csi rk628-csi: cnt_num:1000, tmds_cnt:3000, hs_cnt:15, vs_cnt:3667, hofs:192
m00_b_rk628-csi rk628-csi: SCDC_REGS1:0xffff0f00, act:1920x1080, total:2200x1125, fps:60,

m00_b_rk628-csi rk628-csi: hfp:88, hs:45, hbp:147, vfp:4, vs:5, vbp:36, interlace:0
m00_b_rk628-csi rk628-csi: rk628_csi_s_dv_timings: 1920x1080p60.0 (2200x1125)
m00_b_rk628-csi rk628-csi: enable_stream: disable
```

Open subdev 权限异常

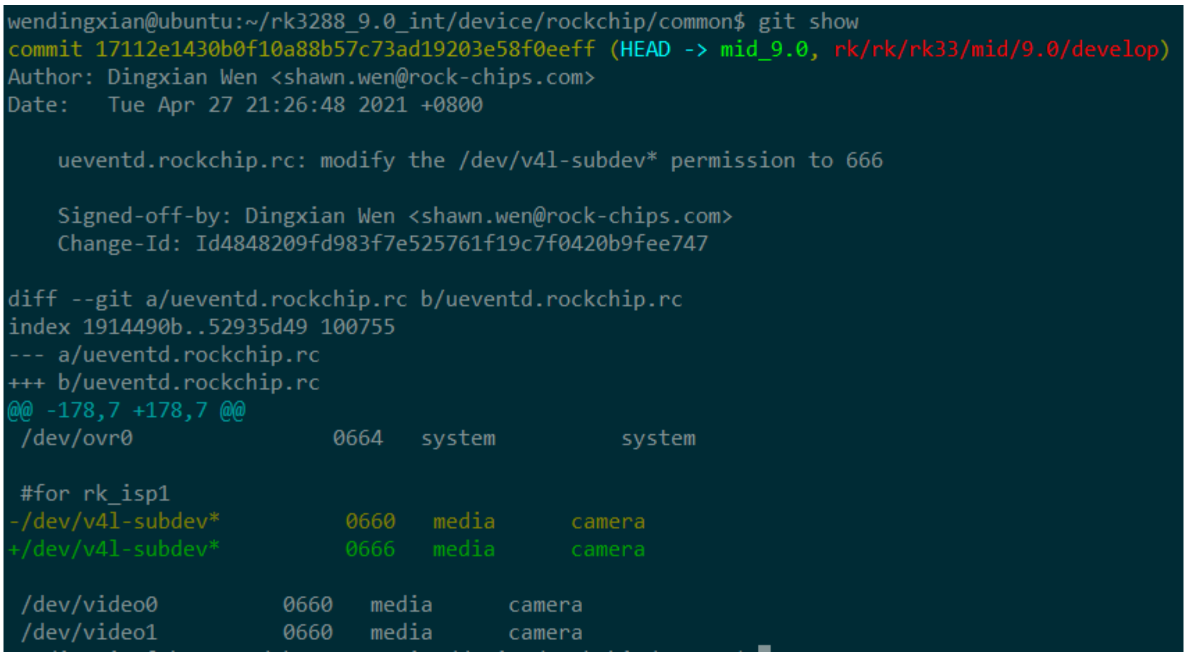


```
D JNI      : JNI CAMERA CALL init
I HdmiInput-navtive: JNI OnLoad
I HdmiInput-navtive: Apk Version: V1.2
E HdmiInput-navtive: openDevice(91): open /dev/v4l-subdev2 failed,erro=Permission denied
D RockchipCamera2: remove take pic button
D RockchipCamera2: recreatTextureview
I RockchipCamera2: textureView remove
D RockchipCamera2: onResume
```

需要确认是否已经给/dev/v4l-subdev*提供666的权限，在设备命令行查询：

```
rk3288:/ # cat /vendor/ueventd.rc | grep subdev
/dev/v4l-subdev*          0666   media      camera
```

目前SDK代码已经包含了本提交，若未包含，可在device/rockchip/common/参考如下修改：



```
wendingxian@ubuntu:~/rk3288_9.0_int/device/rockchip/common$ git show
commit 17112e1430b0f10a88b57c73ad19203e58f0eeff (HEAD -> mid_9.0, rk/rk/rk33/mid/9.0/develop)
Author: Dingxian Wen <shawn.wen@rock-chips.com>
Date:   Tue Apr 27 21:26:48 2021 +0800

    ueventd.rockchip.rc: modify the /dev/v4l-subdev* permission to 666

    Signed-off-by: Dingxian Wen <shawn.wen@rock-chips.com>
    Change-Id: Id4848209fd983f7e525761f19c7f0420b9fee747

diff --git a/ueventd.rockchip.rc b/ueventd.rockchip.rc
index 1914490b..52935d49 100755
--- a/ueventd.rockchip.rc
+++ b/ueventd.rockchip.rc
@@ -178,7 +178,7 @@
 /dev/ovr0                 0664   system          system

 #for rk_isp1
-/dev/v4l-subdev*           0660   media      camera
+/dev/v4l-subdev*           0666   media      camera

 /dev/video0           0660   media      camera
 /dev/video1           0660   media      camera
```

信号识别不到

步骤1. 拔插HDMI，串口没有RK628相关的log打印，说明det脚的gpio配置错误。

步骤2. 拔出HDMI有RK628的log打印，而插入没有RK628的log打印，说明plugin-det-gpios脚极性设置反了，如果中间加了三极管反向那么要设置GPIO_ACTIVE_LOW。 判断拔插状态是否吻合，可以把rk628_csi_v4l2.c 里面，static int debug;改成static int debug =3;
插入的时候会打印tx_5v_power_present: 1 ， 拔出的时候tx_5v_power_present: 0 就算对了

步骤3. 插入一直检测不到正常信号，可能有下面几种情况：

情况1. hpd脚反向问题，RK628给hdmiin座子那边要有高电平，hdmi才会有输出，如果加了反向器，那么RK628管脚需要输出低，反向之后到座子那边才会变高。那么dts要加上hpd-output-inverted;

情况2. phy锁不住，请参考clk det 异常问题；

情况3. 分辨率锁不住，可能是HDMIRX检测到码率超出规定范围；

情况4. 分辨率能锁住。avi_rcv_rdy在信号锁住后会设置为1，但是如果报avi信号没有ready，这种可能是中断脚没有配置。例如以下LOG一直打印avi_rcv_rdy:0，表示能识别到正常分辨率，但是avi_rcv_rdy为0，信号没锁住，优先检查下中断脚配置是否正确。

```
rk628d 1-0050: rk628_is_avi_ready PDEC_AVI_PB:0x10000840, avi_rcv_rdy:0
rk628d 1-0050: SCDC_REGS1:0x80000f00, act:1920x1080, total:2200x1125, fps:60,
pixclk:148500000
```

这种情况下去抓图，测试过rk3288 抓图会概率性出现图像分割界面

显示异常

1. 确认输入源是否是DVI Mode，目前RK628不支持DVI，可以查看log:

   ```
   rk628-csi-v4l2 1-0051: DVI mode detected
   ```

2. 确认输入源是否有HDCP加密，如果有加密，需要打开RK628 HDMIRX的HDCP功能。

抓图失败

1. Android9.0 之后及 linux 平台

RK平台 Android9.0 之后或者 linux 可以通过v4l2-ctl抓图。
先看`media-ctl -d /dev/mediaX -p`，media 节点编号根据各个平台不同而不同。如果不清楚，一般情况把cif控制器关掉，会是media0，rk3399如果接phy1，可能就是media1等等,请了解平台camera配置.
如果media-ctl拓扑下面有rk628的节点，如果没有应该就是dts配置有问题，自行检查，另外拓扑结构所有分辨率并不和你输入的分辨率匹配，那么也得通过media-ctl设置成一致。

然后执行

```
v4l2-ctl -d /dev/video0 --set-fmt-
video=width=1920,height=1080,pixelformat=NV12 --stream-mmap=3`
```

如果一直打印<<<<<<<<<< 说明抓图正常，这种情况驱动一般已经正常。
如果一直报错，或者概率报错：

```
rkisp1: MIPI mis error:
```

一般可能跟硬件相关，检查硬件hdmi线有无分叉，阻抗匹配是否一致，另外比较多客户喜欢hdmi线上接电阻（默认原理图上不接），这种我们默认非必要情况下阻值为0，如果不为0，可能概率性出现报错情况。要具体分析，检查下rk628到主控这端的信号；

或者出现

```
rkisp-vir0: MIPI error: overflow: 0x00000001
```

这个log是rk356x rv11xx容易在isp 使用4k输入的时候出现的问题，这种情况建议走rk628+vicap链路，请搜索本文档vicap配置方式说明

另外打印如下这种SIZE_ERROR，有可能是输入720p，却用1080p采集，这种分辨率不匹配的情况，要检查拓扑结构所有分辨率，以及v4l2-ctl采集分辨率是否对的问题，当然建议是刚开始先都用1080p去测试

```
rkisp1 ff910000.rkisp1: CIF_ISP_PIC_SIZE_ERROR
```

另外系统默认用的是/dev/v4l-subdev2去判断分辨率，但是如果用`media-ctl -p`看的拓扑结构为/dev/v4l-subdev3，如下：

```
 - entity 70: m00_b_rk628-csi rk628-csi (1 pad, 1 link)
          type V4L2 subdev subtype Sensor
          device node name /dev/v4l-subdev3
pad0: Source
    [fmt:UYVY2X8/1280x720]
    -> "rockchip-csi2-dphy0":0 []
```

这样就会识别不到分辨率，应用在判断分辨率会识别不到，也会打印CIF_ISP_PIC_SIZE_ERROR，为了解决这个问题，需要将android的应用jni/native.cpp文件做如下修改：

```
strcat(video_name, "/dev/v4l-subdev2");
```

改成

```
strcat(video_name, "/dev/v4l-subdev3");
```

2. 4K分辨率不收图

 4K因为频率更高，也有可能是硬件信号质量不好的缘故，另外3399 isp需要超频，参考RK3399配置isp超频的方法

APK打开失败

发布的补丁会自带rkCamera2源码，请在系统里面编译(可以直接放在根目录下，然后mmm apk目录)。
应用闪退可能是以下原因导致的：

1. 依赖库问题

   如果只将apk安装到系统，那么会报

   ```
   AndroidRuntime: java.lang.UnsatisfiedLinkError: dlopen failed: library
   "/system/lib64/libhdmiinput_jni.so" needed or dlopened by
   "/apex/com.android.runtime/lib64/libnativeloader.so" is not accessible for
   the namespace "classloader-namespace"
   ```

   这种情况建议直接在SDK编译并打包，依赖库会打包到img里面。

2. 文件权限问题

   APK通过ioctl的方式访问RK628D设备节点，比如：

   ```
   HdmiInput-navtive: openDevice(113): open /dev/v4l-subdev2
   failed,erro=Permission denied
   ```

   首先确定文件权限，补丁有设置成666，由于在APK访问了设备节点，所以需要确认是否关闭了selinux，可通过getenforce命令查看。

3. 上层配置问题

   Android 9.0 之后配置camera3_profiles.xml，Android 9.0之前配置cam_board.xml，可以通过dumpsys media.camera查看，如果没有，可能就是没有配置：

   ```
   $ dumpsys media.camera

   == Service global info: ==

   Number of camera devices: 1
   Number of normal camera devices: 1
   ```

4. camera库crash

   例如出现以下错误，可能是camera名称错误导致，因为getDataFromXmlFile获取到的名字和驱动的不吻合，导致CRASH。

```
09-29 04:43:30.673   405   405 W ServiceManagement: Waited one second for
android.hardware.camera.provider@2.4::ICameraProvider/legacy/0. Waiting
another...
09-29 04:43:31.673   405   405 W ServiceManagement: Waited one second for
android.hardware.camera.provider@2.4::ICameraProvider/legacy/0. Waiting
another...
...
09-29 04:43:31.904  1404  1404 F DEBUG   : backtrace:
09-29 04:43:31.904  1404  1404 F DEBUG   :     #00 pc 0007214e
/vendor/lib/hw/camera.rk30board.so
(android::camera2::ChromeCameraProfiles::handleAndroidStaticMetadata(char
const*, char const**)+546)
09-29 04:43:31.904  1404  1404 F DEBUG   :     #01 pc 00007895
/system/lib/vndk-28/libexpat.so (doContent+432)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #02 pc 0000637b
/system/lib/vndk-28/libexpat.so (contentProcessor+40)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #03 pc 00003825
/system/lib/vndk-28/libexpat.so (XML_ParseBuffer+84)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #04 pc 000711ad
/vendor/lib/hw/camera.rk30board.so
(android::camera2::CameraProfiles::getDataFromXmlFile()+148)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #05 pc 00071e97
/vendor/lib/hw/camera.rk30board.so
(android::camera2::ChromeCameraProfiles::init()+86)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #06 pc 000734c7
/vendor/lib/hw/camera.rk30board.so (android::camera2::PlatformData::init()+198)
09-29 04:43:31.905  1404  1404 F DEBUG   :     #07 pc 000cecb3
/vendor/lib/hw/camera.rk30board.so (initCameraHAL()+46)
...
```

对于需要v4l2（android9.0及以上）的需要将camera3_profiles_rk3399.xml里面的name="rk628-csi"改名成name="rk628-csi-v4l2"，修改如下：

```
hardware/rockchip/camera
--- a/etc/camera/camera3_profiles_rk3399.xml
+++ b/etc/camera/camera3_profiles_rk3399.xml
@@ -769,7 +769,7 @@

 <!-- ******************PSL specific section end
*****************************************************************-->
     </Profiles>
-    <Profiles cameraId="0" name="rk628-csi" moduleId="m00">
+    <Profiles cameraId="0" name="rk628-csi-v4l2" moduleId="m00">
```

5. APK权限导致的闪退

   出现该问题的LOG大概如下：

   对于需要v4l2（android9.0及以上）的需要将"rk628-csi"改名成"rk628-csi-v4l2"，修改如下：

```
09-24 10:21:20.154  1552  1570 E AndroidRuntime: FATAL EXCEPTION: Thread-2
09-24 10:21:20.154  1552  1570 E AndroidRuntime: Process:
com.android.rockchip.camera2, PID: 1552
09-24 10:21:20.154  1552  1570 E AndroidRuntime:
java.lang.IllegalStateException: startRecording() called on an uninitialized
AudioRecord.
09-24 10:21:20.154  1552  1570 E AndroidRuntime:        at
android.media.AudioRecord.startRecording(AudioRecord.java:983)
09-24 10:21:20.154  1552  1570 E AndroidRuntime:        at
com.android.rockchip.camera2.AudioStream$recordSound.run(AudioStream.java:199)
09-24 10:21:20.154  1552  1570 E AndroidRuntime:        at
java.lang.Thread.run(Thread.java:764)
09-24 10:21:20.164   464   820 W ActivityManager:   Force finishing activity
com.android.rockchip.camera2/.RockchipCamera2
09-24 10:21:20.183  1552  1552 D RockchipCamera2: onPause
```

可以采用以下补丁解决（RK628代码更新到V15-210926或更新也可以）

```
packages/apps/rkCamera2
--- a/AndroidManifest.xml
+++ b/AndroidManifest.xml
@@ -4,7 +4,8 @@
     package="com.android.rockchip.camera2">

         <uses-permission android:name="android.permission.CAMERA" />
-        <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
+       <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
+       <uses-permission android:name="android.permission.RECORD_AUDIO" />
     <application
         android:allowBackup="true"
         android:icon="@mipmap/ic_launcher"
diff --git a/src/com/android/rockchip/camera2/RockchipCamera2.java
b/src/com/android/rockchip/camera2/RockchipCamera2.java
index 9dc29b8..02ee104 100755
--- a/src/com/android/rockchip/camera2/RockchipCamera2.java
+++ b/src/com/android/rockchip/camera2/RockchipCamera2.java
@@ -100,9 +100,11 @@ public class RockchipCamera2 extends Activity {
        if (ActivityCompat.checkSelfPermission(this,
            Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission(this,
-                   Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {
+                   Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED
+           && ActivityCompat.checkSelfPermission(this,
+                   Manifest.permission.RECORD_AUDIO) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(RockchipCamera2.this,
-               new String[] { Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE },
+               new String[] { Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE, Manifest.permission.RECORD_AUDIO},
                REQUEST_CAMERA_PERMISSION);
            return;
        }
```

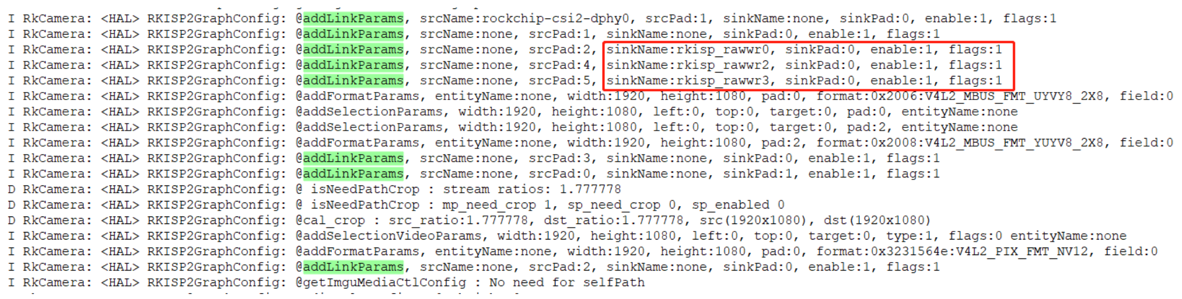
dts配置连接到rkcif，apk预览失败

RK356X，android11代码版本是R10/R11，dts配置RK628连接到rkcif，抓图正常，apk打开预览失败，可能是R10和R11的代码cameraHAL部分不支持该pipeline链路造成的，R9没有这个问题。排查方法和解决方法如下：

1. 打开cameraHAL开关：

```
setprop persist.vendor.camera.hal.debug 5
```

2. 抓取打开apk出错的logcat，log关键信息如下：



```
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:rockchip-csi2-dphy0, srcPad:1, sinkName:none, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:1, sinkName:none, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:2, sinkName:rkisp_rawwr0, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:4, sinkName:rkisp_rawwr2, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:5, sinkName:rkisp_rawwr3, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addFormatParams, entityName:none, width:1920, height:1080, pad:0, format:0x2006:V4L2_MBUS_FMT_UYVY8_2X8, field:0
I RkCamera: <HAL> RKISP2GraphConfig: @addSelectionParams, width:1920, height:1080, left:0, top:0, target:0, pad:0, entityName:none
I RkCamera: <HAL> RKISP2GraphConfig: @addSelectionParams, width:1920, height:1080, left:0, top:0, target:0, pad:2, entityName:none
I RkCamera: <HAL> RKISP2GraphConfig: @addFormatParams, entityName:none, width:1920, height:1080, pad:2, format:0x2008:V4L2_MBUS_FMT_YUYV8_2X8, field:0
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:3, sinkName:none, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:0, sinkName:none, sinkPad:1, enable:1, flags:1
D RkCamera: <HAL> RKISP2GraphConfig: @ isNeedPathCrop : stream ratios: 1.777778
D RkCamera: <HAL> RKISP2GraphConfig: @ isNeedPathCrop : mp_need_crop 1, sp_need_crop 0, sp_enabled 0
D RkCamera: <HAL> RKISP2GraphConfig: @cal_crop : src_ratio:1.777778, dst_ratio:1.777778, src(1920x1080), dst(1920x1080)
I RkCamera: <HAL> RKISP2GraphConfig: @addSelectionVideoParams, width:1920, height:1080, left:0, top:0, target:0, type:1, flags:0 entityName:none
I RkCamera: <HAL> RKISP2GraphConfig: @addFormatParams, entityName:none, width:1920, height:1080, pad:0, format:0x3231564e:V4L2_PIX_FMT_NV12, field:0
I RkCamera: <HAL> RKISP2GraphConfig: @addLinkParams, srcName:none, srcPad:2, sinkName:none, sinkPad:0, enable:1, flags:1
I RkCamera: <HAL> RKISP2GraphConfig: @getImguMediaCtlConfig : No need for selfPath
```

如果看到上述log，则是cameraHAL默认将对应的pipeline配置到rkisp了，而dts对应的是配置到rkcif

3. 解决办法

   hardware/rockchip/camera目录下，添加如下修改：

```
diff --git a/psl/rkisp2/RKISP2GraphConfig.cpp b/psl/rkisp2/RKISP2GraphConfig.cpp
index 2a5fa5a..3d2409c 100755
--- a/psl/rkisp2/RKISP2GraphConfig.cpp
+++ b/psl/rkisp2/RKISP2GraphConfig.cpp
@@ -76,6 +76,7 @@ const string MEDIACTL_POSTVIEWNAME = "postview";

 const string MEDIACTL_STATNAME = "rkisp1-statistics";
 const string MEDIACTL_VIDEONAME_CIF = "stream_cif_dvp_id0";
+const string MEDIACTL_VIDEONAME_CIF_MIPI_ID0 = "stream_cif_mipi_id0";

 RKISP2GraphConfig::RKISP2GraphConfig() :
         mManager(nullptr),
@@ -2620,6 +2621,13 @@ status_t RKISP2GraphConfig::getImguMediaCtlConfig(int32_t
cameraId,
                  addLinkParams("rkisp-isp-subdev", 2, "rkisp_mainpath", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
                  addLinkParams("rkisp-isp-subdev", 2, "rkisp_selfpath", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
             }
+        } else if(mipName2.find("mipi") != std::string::npos) {
+            addLinkParams(mipName, mipSrcPad, mipName2, csiSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+            addLinkParams(mipName2, 1, "stream_cif_mipi_id0", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+            addLinkParams(mipName2, 2, "stream_cif_mipi_id1", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+            addLinkParams(mipName2, 3, "stream_cif_mipi_id2", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
```

```
+             addLinkParams(mipName2, 4, "stream_cif_mipi_id3", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
+             mSensorLinkedToCIF = true;
        } else {
             addLinkParams(mipName, mipSrcPad, csiName, csiSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
             addLinkParams(csiName, csiSrcPad, IspName, ispSinkPad, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
@@ -2628,6 +2636,12 @@ status_t RKISP2GraphConfig::getImguMediaCtlConfig(int32_t
cameraId,
             addLinkParams(csiName, 5, "rkisp_rawwr3", 0, 1,
MEDIA_LNK_FL_ENABLED, mediaCtlConfig);
        }
     }
+    if(mSensorLinkedToCIF){
+        addImguVideoNode(IMGU_NODE_VIDEO, MEDIACTL_VIDEONAME_CIF_MIPI_ID0,
mediaCtlConfig);
+        addFormatParams(MEDIACTL_VIDEONAME_CIF_MIPI_ID0, mCurSensorFormat.width,
mCurSensorFormat.height,
+                0, V4L2_PIX_FMT_NV12, 0, 0, mediaCtlConfig);
+         return OK;
+    }
     // isp input pad format and selection config
     addFormatParams(IspName, ispInWidth, ispInHeight, ispSinkPad, ispInFormat,
0, 0, mediaCtlConfig);
     addSelectionParams(IspName, ispInWidth, ispInHeight, 0, 0,
V4L2_SEL_TGT_CROP, ispSinkPad, mediaCtlConfig);
```

如何操作RK628的GPIO

在需要控制GPIO的位置直接调用GPIO接口即可，该接口包括IOMUX设置，GPIO方向设置和电平设置，例如以下操作将I2C_SDA_HDMI和I2C_SCL_HDMI接口设为GPIO。

```
rk628_gpio_direction_output(rk628, GPIO1_B1, GPIO_REG_HIGH);
rk628_gpio_direction_output(rk628, GPIO1_B2, GPIO_REG_HIGH);
```

若需要恢复则需要调用以下接口。

```
rk628_pinctrl_set_mux(rk628, GPIO1_B1, DDCM0SDARX);
rk628_pinctrl_set_mux(rk628, GPIO1_B2, DDCM0SCLRX);
```

HDMI-IN色域问题处理

对于HDMI-IN场景，显示画面如果不够亮，可能是色域范围不够，除了驱动要更新到V20版本之外，HAL层也需要相应处理。对于HAL版本（ANDROID8及更早版本）的修改如下：

```
hardware/rockchip/camera
--- a/psl/rkisp1/RKISP1CameraHw.cpp
+++ b/psl/rkisp1/RKISP1CameraHw.cpp
@@ -389,13 +389,15 @@
RKISP1CameraHw::configStreams(std::vector<camera3_stream_t*> &activeStreams,
             GRALLOC_USAGE_HW_VIDEO_ENCODER |
             GRALLOC_USAGE_HW_CAMERA_WRITE |
             RK_GRALLOC_USAGE_SPECIFY_STRIDE|
-            GRALLOC_USAGE_PRIVATE_1; // full range
+            // GRALLOC_USAGE_PRIVATE_1; // full range
+            GRALLOC_USAGE_PRIVATE_0; //limit range
     camera3_stream_t* stillStream = findStreamForStillCapture(activeStreams);
     for (unsigned int i = 0; i < activeStreams.size(); i++) {
         activeStreams[i]->max_buffers = maxBufs;
         activeStreams[i]->usage |= usage;
+        activeStreams[i]->data_space = HAL_DATASPACE_RANGE_LIMITED;
         if (activeStreams[i] != stillStream) {
             mStreamsVideo.push_back(activeStreams[i]);
```

而对于HAL3版本（ANDROID9或更新）补丁如下：

```
hardware/rockchip/camera
--- a/CameraHal/DisplayAdapter.cpp
+++ b/CameraHal/DisplayAdapter.cpp
int DisplayAdapter::cameraDisplayBufferCreate(int width, int height, const char
*fmt,int numBufs)
{
    // set yuv fullrange mode for preview
    #if (defined(TARGET_RK3399) || defined(TARGET_RK3288) ||
defined(TARGET_RK3326))
-        gralloc_usage |= GRALLOC_USAGE_PRIVATE_1;
+        gralloc_usage |= GRALLOC_USAGE_PRIVATE_0;
    #elif (defined(TARGET_RK3368))
        gralloc_usage |= HAL_DATASPACE_BT601_525;
    #endif    // Set gralloc usage bits for window.
    err = mANativeWindow->set_usage(mANativeWindow, gralloc_usage);
```

声卡注册失败

如下，系统启动后，找不到hdmiin声卡

```
# cat /proc/asound/cards
 1 [rockchiphdmiin ]: rockchip_hdmiin - rockchip,hdmiin
                      rockchip,hdmiin
```

那么首先要检查的是，声卡依赖的一个soc dai、codec dai设备是否有加载成功：

```
rk3399:/ # cat /d/asoc/components
...
dummy-codec
ff8a0000.i2s
ff8a0000.i2s
ff880000.i2s
ff880000.i2s
...
rk3399:/ # cat /d/asoc/dais
...
ff8a0000.i2s
ff880000.i2s
dummy-codec
....
```

以上节点加载失败的话，可以检查defconfig、dts

在soc dai codec dai 都注册成功的情况下，出现声卡注册不上，有可能是dma资源不够，尤其是rk3399 情况下出现比较多，在4.19 以上内核，可以在i2s节点添加下面属性来解决：

```
--- a/arch/arm64/boot/dts/rockchip/rk3399.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399.dtsi
@@ -1788,6 +1788,7 @@
                clocks = <&cru SCLK_I2S2_8CH>, <&cru HCLK_I2S2_8CH>;
                resets = <&cru SRST_I2S2_8CH>, <&cru SRST_H_I2S2_8CH>;
                reset-names = "reset-m", "reset-h";
+               rockchip,capture-only;
                power-domains = <&power RK3399_PD_SDIOAUDIO>;
                status = "disabled";
```

常见需求处理

RK628 24M晶振来自其他SOC的配置方式

下面分别以几个ROCKCHIP的主控来提供配置方法，其中DTS的配置CLK都默认有在代码中使能，如果未使能开机后24M CLK会被关闭，从而导致I2C通信异常。

RK3399 添加24M 支持

1. 在rk628 dts中引用以下配置

```
pinctrl-names = "default";
pinctrl-0 = <&rk628_rst>, <&clk_testout2>;
assigned-clocks = <&cru SCLK_TESTCLKOUT2>;
assigned-clock-rates = <24000000>;
clocks = <&cru SCLK_TESTCLKOUT2>;
clock-names = "soc_24M";
```

2. 根据实际硬件接法添加IO脚的PINCTRL配置

```
&pinctrl {
        test {
                clk_testout2: clk_testout2 {
                rockchip,pins = <0 8 RK_FUNC_3 &pcfg_pull_none>;
        };
};
```

RK3288 添加24M 支持

1. 在rk628 dts中引用以下配置

```
pinctrl-names = "default";
pinctrl-0 = <&test_clkout>;
assigned-clocks = <&cru SCLK_TESTOUT_SRC>;
assigned-clock-parents = <&xin24m>;
clocks = <&cru SCLK_TESTOUT>;
clock-names = "soc_24M";
```

2. 根据实际硬件接法添加IO脚的PINCTRL配置

```
&pinctrl {
        test {
                test_clkout: test-clkout {
                        rockchip,pins = <0 17 RK_FUNC_1 &pcfg_pull_none>;
                };
        };
};
```

RK356X 添加24M 支持

RK356X能输出24MHZ的引脚比较多，例如REF_CLKOUT（clk_wifi/gpio0_a0）、CAM_CLKOUT1（clk_cam1_out/gpio4_b0）、ETH_REFCLK_25M_M0（clk_mac1_out/gpio3_b0）,现以第一个为例介绍。

1. 在rk628 dts中引用以下配置

```
pinctrl-names = "default";
pinctrl-0 = <&refclk_pins>;
assigned-clocks = <&pmucru CLK_WIFI>;
assigned-clock-rates = <24000000>;
clocks = <&pmucru CLK_WIFI>;
clock-names = "soc_24M";
```

2. 以下配置在rk3568-pinctrl.dtsi中自带

```
refclk {
         /omit-if-no-ref/
         refclk_pins: refclk-pins {
         rockchip,pins =
         /* refclk_ou */
         <0 RK_PA0 1 &pcfg_pull_none>;
       };
};
```

其他未写的RK芯片，可以参考文档“Rockchip_Develop_Guide_Gpio_Output_Clocks_CN.pdf”进行24M的处理。

双RK628支持

HDMI2CSI+HDMI2CSI支持

注意事项

1. 两路RK628实现类似camera双摄的做法，两路RK628能够同时工作，需要主控支持两个isp控制器，同时处理两个输入数据，例如RK3399等。
2. 在硬件设计上，两个RK628需要挂在不同的I2C下或者两路RK628挂在同一个I2C下但设计成不同的I2C地址。
3. 注意dts的配置需要跟硬件设计一致，具体的rk628硬件接到mipi_phy_rx0或者mipi_phy_tx1rx1，在dts也需要对应的配置。
4. 使用两路rk628实现HDMI2CSI，两路rk628的RESET、INT控制io不能用一样的，否则会异常。

kernel dts 配置问题

现在以RK3399配置两路RK628为例介绍。

1. rk628->mipi_dphy_rx0->rkisp1_0，第一路index为0，facing配置为back

```
&i2c1 {
    status = "okay";

    rk628_csi_v4l2: rk628_csi_v4l2@50 {
        compatible = "rockchip,rk628-csi-v4l2";
        reg = <0x50>;
        //clocks = <&ext_cam_clk>;
        //clock-names = "xvclk";

        interrupt-parent = <&gpio4>;
        interrupts = <16 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&rk628_irq>;
        //power-gpios = <&gpio0 RK_PD5 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio4 RK_PD2 GPIO_ACTIVE_LOW>;
        plugin-det-gpios = <&gpio0 RK_PD6 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "RK628-CSI";
        rockchip,camera-module-lens-name = "NC";
        port {
            rk628_out: endpoint {
                remote-endpoint = <&hdmi2mipi_in>;
                data-lanes = <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>;

            hdmi2mipi_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&rk628_out>;
                data-lanes = <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>;
            };
        };
    };
};

&rkisp1_0 {
    status = "okay";

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

        isp0_mipi_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy_rx0_out>;
        };
    };
};
```

2. rk628->mipi_dphy_tx1rx1->rkisp1_1，第二路index为1，facing配置为front

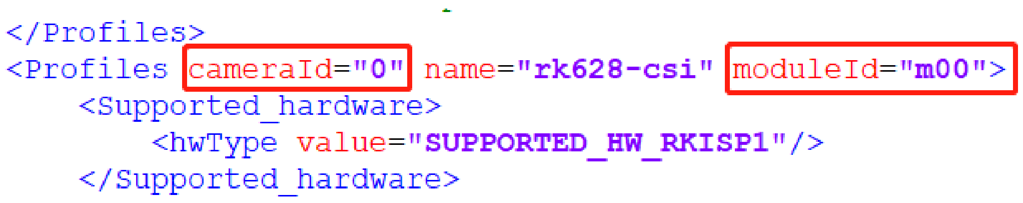
```
&i2c4 {
    status = "okay";

    rk628_csi_v4l2_1: rk628_csi_v4l2_1@50 {
        compatible = "rockchip,rk628-csi-v4l2";
        reg = <0x50>;
        //clocks = <&ext_cam_clk>;
        //clock-names = "xvclk";

        interrupt-parent = <&gpio3>;
        interrupts = <12 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&rk628_irq1>;
        //power-gpios = <&gpio0 RK_PD5 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio4 RK_PD3 GPIO_ACTIVE_LOW>;
        plugin-det-gpios = <&gpio0 RK_PD7 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
        rockchip,camera-module-name = "RK628-CSI";
        rockchip,camera-module-lens-name = "NC";
        port {
            rk628_out1: endpoint {
                remote-endpoint = <&hdmi2mipi_in1>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};

&mipi_dphy_tx1rx1 {
    status = "okay";

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

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

            hdmi2mipi_in1: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&rk628_out1>;
                data-lanes = <1 2 3 4>;
            };
        };

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

            dphy_tx1rx1_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&isp1_mipi_in>;
            };
        };
    };
};

&rkisp1_1 {
    status = "okay";

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

        isp1_mipi_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy_tx1rx1_out>;
        };
    };
};
```

android 配置问题

1. camera3_profiles.xml文件配置注意事项：

moduleId：需要与dts中的index一致

第一路rk628：



```
</Profiles>
<Profiles cameraId="0" name="rk628-csi" moduleId="m00">
    <Supported_hardware>
        <hwType value="SUPPORTED_HW_RKISP1"/>
    </Supported_hardware>
```

第二路rk628：

```
    </Profiles>
    <Profiles cameraId="1" name="rk628-csi" moduleId="m01">
        <Supported_hardware>
            <hwType value="SUPPORTED_HW_RKISP1"/>
        </Supported_hardware>
```

2. 两路rk628如果需要同时使用，cameraHAL需要添加支持，在SDK/hardware/rockchip/camera$目录下添加修改：

```
diff --git a/common/platformdata/PlatformData.cpp
b/common/platformdata/PlatformData.cpp
index 36d2ac9..d3fcb4b 100755
--- a/common/platformdata/PlatformData.cpp
+++ b/common/platformdata/PlatformData.cpp
@@ -1047,7 +1047,7 @@ CameraHWInfo::CameraHWInfo() :
     mProductName = "<not_set>";
     mManufacturerName = "<not set>";
     mCameraDeviceAPIVersion = CAMERA_DEVICE_API_VERSION_3_3;
-    mSupportDualVideo = false;
+    mSupportDualVideo = true;
     mSupportExtendedMakernote = false;
     mSupportFullColorRange = true;
     mSupportIPUAcceleration = false;
```

HDMI2CSI+HDMI2DSI支持

kernel dts 配置

HDMI2CSI+HDMI2DSI组合：一路RK628走camera框架，接到主控端，一路RK628接到屏幕。这里以RK3399为例

1. HDMI2CSI配置

rk628->mipi_phy_rx0->rkisp1_0 或者rk628->mipi_dphy_tx1rx1->rkisp1_1，以接rkisp1_0为例

```
&i2c1 {
    status = "okay";

    rk628_csi_v4l2: rk628_csi_v4l2@50 {
        compatible = "rockchip,rk628-csi-v4l2";
        reg = <0x50>;
        //clocks = <&ext_cam_clk>;
        //clock-names = "xvclk";

        interrupt-parent = <&gpio4>;
        interrupts = <16 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&rk628_irq>;
        //power-gpios = <&gpio0 RK_PD5 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio4 RK_PD2 GPIO_ACTIVE_LOW>;
        plugin-det-gpios = <&gpio0 RK_PD6 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "RK628-CSI";
        rockchip,camera-module-lens-name = "NC";
        port {
            rk628_out: endpoint {
                remote-endpoint = <&hdmi2mipi_in>;
                data-lanes = <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>;

            hdmi2mipi_in: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&rk628_out>;
                data-lanes = <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>;
            };
        };
    };
};

&rkisp1_0 {
    status = "okay";

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

        isp0_mipi_in: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&dphy_rx0_out>;
        };
    };
};
```

2. HDMI2DSI配置

直接参考2.4.5 章节的HDMI2DSI转换即可。

android配置

1. HDMI2CSI：参考一路rk628 HDMI2CSI即可，正确配置camera3_profiles.xml即可。
2. HDMI2DSI: android部分不需要做配置