Rockchip HDMI RX开发指南

文件标识：RK-KF-YF-321

发布版本：V1.1.7

日期：2024-04-28

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

本文档是基于RK3588 Android 12平台使用HDMI RX模块开发 HDMI IN功能的帮助文档。

概述

HDMI IN功能可以通过桥接芯片的方式实现，将HDMI信号转换成MIPI信号接收，RK3588芯片平台自带HDMI RX模块，可以直接接收HDMI信号。本文档主要介绍在RK3588 Android 12平台通过HDMI RX模块开发实现HDMI IN功能的方法。支持HDMI IN自适应分辨率：4K60、4K30、1080P60、720P60等，支持HDMI IN热拔插，支持录像功能，支持EDID可配置，支持HDCP1.4/HDCP2.3，支持CEC。

产品版本

芯片名称	**Kernel**版本	**Android**版本
RK3588	Linux 5.10	Android12

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	温定贤/郭旺强/王杭/陈顺庆/兰顺华	2022.06.29	初始版本
V1.1.0	陈顺庆	2022.08.09	修改HDCP和CEC说明
V1.1.1	王杭	2022.11.04	修改HDMIIN APK说明
V1.1.2	陈顺庆	2023.4.12	更新HDCP/CEC相关配置说明
V1.1.3	王杭	2023.11.17	添加HDMIIN APK的vtunnel、mipi配置说明
V1.1.4	郭旺强	2023.11.20	增加常见问题调试方法及日志说明
V1.1.5	王杭	2024.01.30	更新TIF预览时预留内存CMA的说明
V1.1.6	王杭	2024.03.04	更新RK3576 MIPI预览时节点属性配置
V1.1.7	郭旺强	2024.04.28	增加重启问题的典型日志及补丁说明

目录

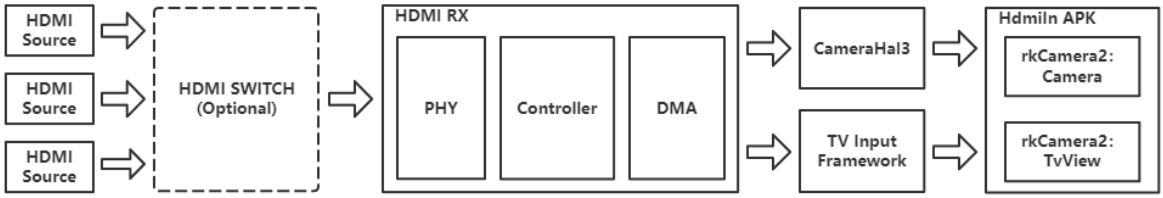
1. HDMI IN功能概述

1.1 HDMI RX模块特性简介

- HDMI 1.4b/2.0 RX：Up to 4K@60fps
- Support FMT：RGB888/YUV420/YUV422/YUV444 8bit
- Pixel clock：Up to 600MHz
- HDCP1.4/2.3
- CEC hardware engine
- E-EDID configuration
- S/PDIF 2channel output
- I2S 2/4/6/8channel output

注：RK3588S不含HDMI RX模块。

1.2 HDMI IN功能框图



根据应用场景需要，HDMI RX可适配TIF框架或是Camera框架，适配TIF框架图像传输延时更低，适配Camera框架可以使用标准Camera API，更方便录像、对接后端算法等应用功能开发。

2. HDMI RX驱动代码和dts配置

2.1 SDK版本要求

HDMI IN功能可能存在持续更新，建议将SDK版本升级到最新，SDK版本号的查询方法如下：

```
rk3588_s:/ # getprop | grep rksdk
[ro.rksdk.version]: [ANDROID12_RKR9]
```

2.2 驱动代码和Kernel配置

驱动代码：

```
drivers/media/platform/rockchip/hdmirx/
```

Kernel Config配置：

```
CONFIG_VIDEO_ROCKCHIP_HDMIRX=y
```

2.3 dts配置说明

参考SDK中RK3588 EVB1的dts配置：

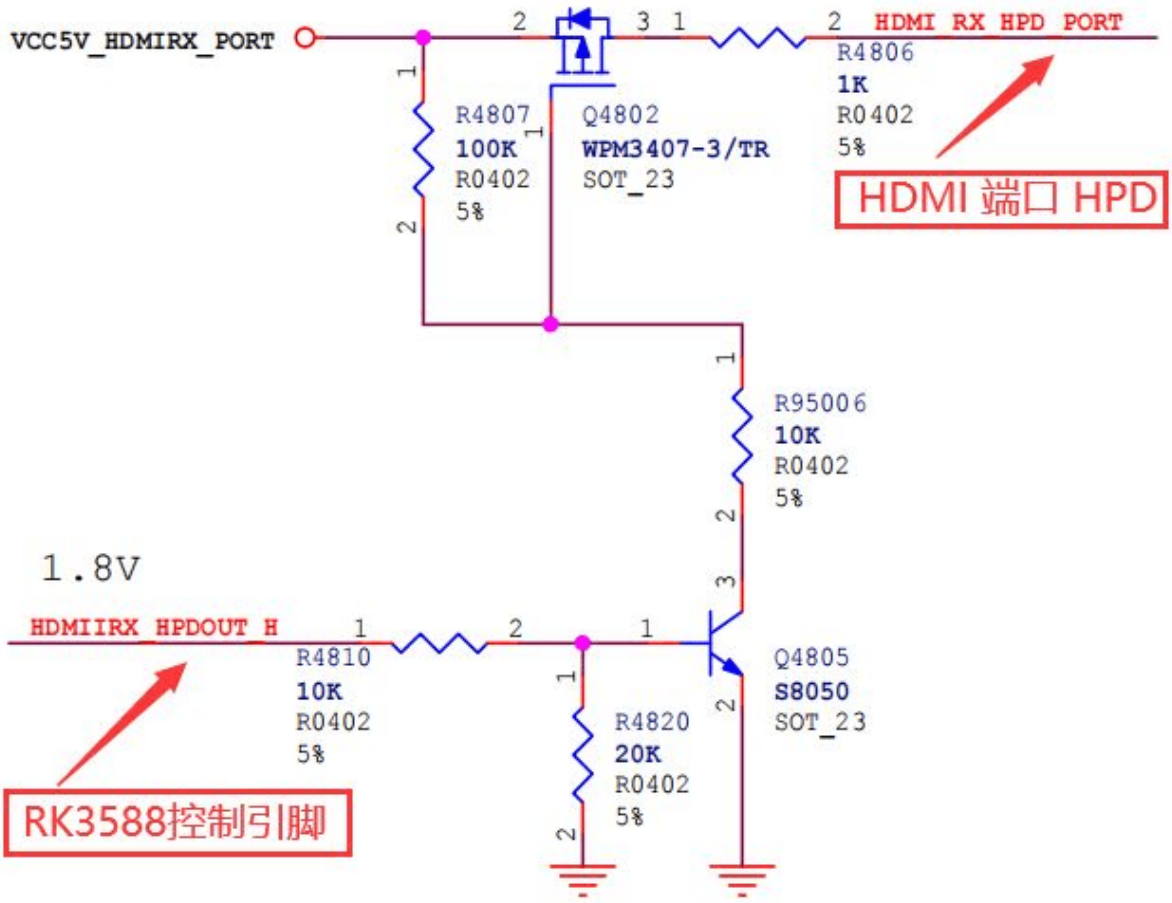
```
arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
```

2.3.1 HDMI RX控制器配置

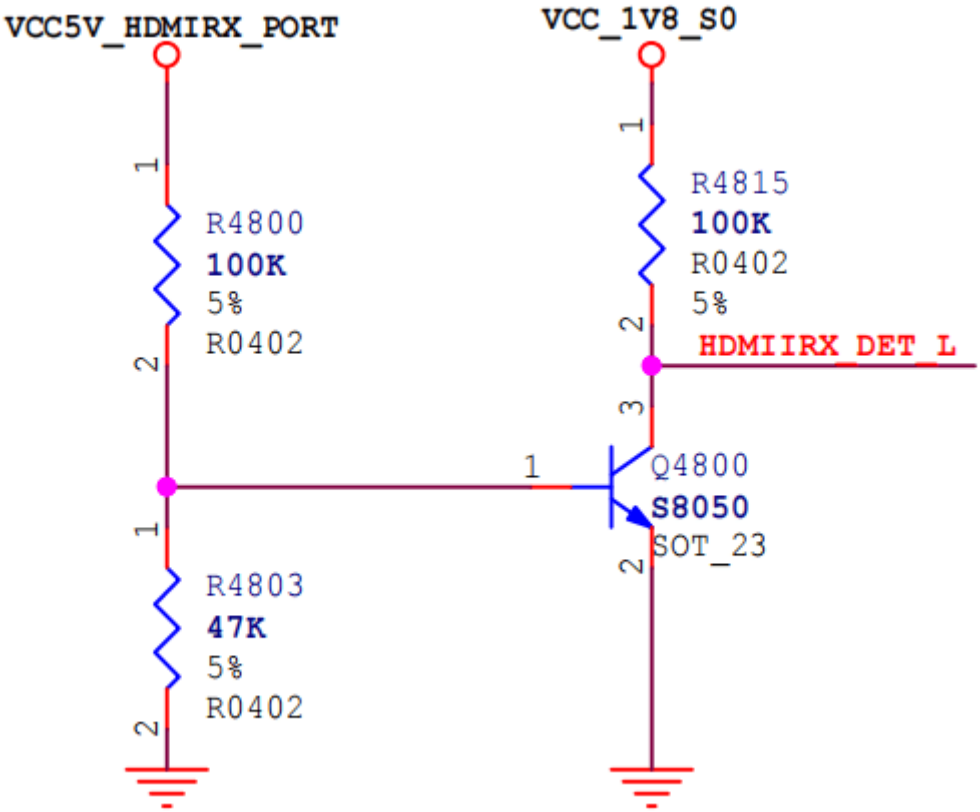
```
/* Should work with at least 128MB cma reserved above. */
&hdmirx_ctrler {
    status = "okay";

    /* Effective level used to trigger HPD: 0-low, 1-high */
    hpd-trigger-level = <1>;
    hdmirx-det-gpios = <&gpio2 RK_PB5 GPIO_ACTIVE_LOW>;
    pinctrl-names = "default";
    pinctrl-0 = <&hdmim1_rx &hdmirx_det>;
};

hdmi {
    hdmirx_det: hdmirx-det {
        rockchip,pins = <2 RK_PB5 RK_FUNC_GPIO &pcfg_pull_up>;
    };
};
```

板级配置需要与实际硬件电路连接对应：

- hpd-trigger-level：触发HPD的有效电平，<1>表示RK3588控制引脚和HDMI端口HPD电平状态相同，<0>则表示相反。



- hdmirx-det-gpios：HDMI插入检测引脚，需要根据实际硬件连接配置GPIO和有效电平，低电平有效时，需要配置pinctrl为内部上拉。



2.3.2 预留内存

RK3588 HDMI RX模块只能使用物理连续内存，需要预留至少128MB的CMA内存，如需增大，修改dts中预留内存长度即可：

注：按3840x2160分辨率，RGB888图像格式，4个轮转Buffer计算。

```
/* If hdmirx node is disabled, delete the reserved-memory node here. */
reserved-memory {
    #address-cells = <2>;
    #size-cells = <2>;
    ranges;

    /* Reserve 128MB memory for hdmirx-controller@fdee0000 */
    cma {
        compatible = "shared-dma-pool";
        reusable;
        /* Start address: 256 * 0x100000, Length: 128 * 0x100000 */
        reg = <0x0 (256 * 0x100000) 0x0 (128 * 0x100000)>;
        linux,cma-default;
    };
};
```

2.3.3 Audio配置

DTS添加声卡配置

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
index 4aa6a8ce9364..84c9c51d3b7f 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
@@ -47,6 +47,26 @@ play-pause-key {
                };
        };
+       hdmiin_dc: hdmiin-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 = <&i2s7_8ch>;
+               };
+               dailink0_master: simple-audio-card,codec {
+                       sound-dai = <&hdmiin_dc>;
+               };
+       };
```

```
+
        pcie20_avdd0v85: pcie20-avdd0v85 {
                compatible = "regulator-fixed";
                regulator-name = "pcie20_avdd0v85";
@@ -460,6 +480,10 @@ &i2s6_8ch {
        status = "okay";
 };
+&i2s7_8ch {
+       status = "okay";
+};
+
```

HAL填写HDMIIN声卡信息

```
diff --git a/tinyalsa_hal/audio_hw.c b/tinyalsa_hal/audio_hw.c
index ea97bee..509d140 100644
--- a/tinyalsa_hal/audio_hw.c
+++ b/tinyalsa_hal/audio_hw.c
@@ -362,6 +362,7 @@ struct dev_proc_info HDMI_IN_NAME[] =
 {
     {"realtekrt5651co", "tc358749x-audio"},
+    {"rockchiphdmiin", NULL},
     {NULL, NULL}, /* Note! Must end with NULL, else will cause crash */
 };
```

以上配置，默认SDK已经包含，如果客户自己修改声卡名称，则需要在HAL的HDMI_IN_NAME数组里面添加对应的声卡信息

2.3.4 HDCP配置

RK3588有两个HDCP2.3控制器(hdcp0 和 hdcp1)，每个HDCP2.3控制器都有3个port：

hdcp0：DPTX0和DPTX1

hdcp1：HDMIRX、HDMITX0和HDMITX1

若HDMIRX需要支持 HDCP2.3，则需要使能 hdcp1。

- 单独支持 HDCP1.4：

```
&hdmirx_ctrler {
    status = "okay";
    hdcp1x-enable;
};
```

- 支持 HDCP2.3，HDCP2.3 开启之后会默认兼容 HDCP1.4：

```
&hdcp1 {
    status = "okay";
};

&hdmirx_ctrler {
    status = "okay";
    hdcp2x-enable;
};
```

2.3.5 CEC配置

```
&hdmirx_ctrler {
    status = "okay";
    cec-enable;
};
```

2.4 EDID配置方法

RK3588支持EDID可配置，目前驱动代码中EDID支持的分辨率包括：

```
3840x2160P60、3840x2160P50、3840x2160P30、3840x2160P25、3840x2160P24、
1920x1080P60、1920x1080P50、1920x1080P30、1920x1080i60、1920x1080i50、
1600x900P60、1440x900P60、1280x800P60、
1280x720P60、1280x720P50、1024x768P60、
720x576P50、720x480P60、720x576i50、720x480i60、
800x600P60、640x480P60
```

支持输入的格式包括：

```
RGB888、YUV420、YUV422、YUV444
```

当前EDID分为两组，通过上层应用可选择配置：

- edid_init_data_340M：是pixel clk小于340M的分辨率，主要是HDMI1.4支持的分辨率，包含4K60 YUV420。
- edid_init_data_600M：是pixel clk为594M的分辨率，支持4K60 YUV422/YUV444/RGB888。

若有EDID配置需求，可直接在驱动代码中修改相应数组：

drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c

```
static u8 edid_init_data_340M[] = {
    0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
    0x49, 0x70, 0x88, 0x35, 0x01, 0x00, 0x00, 0x00,
    0x2D, 0x1F, 0x01, 0x03, 0x80, 0x78, 0x44, 0x78,
    ...
};

static u8 edid_init_data_600M[] = {
    0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00,
    0x49, 0x70, 0x88, 0x35, 0x01, 0x00, 0x00, 0x00,
    0x2D, 0x1F, 0x01, 0x03, 0x80, 0x78, 0x44, 0x78,
    ...
};
```

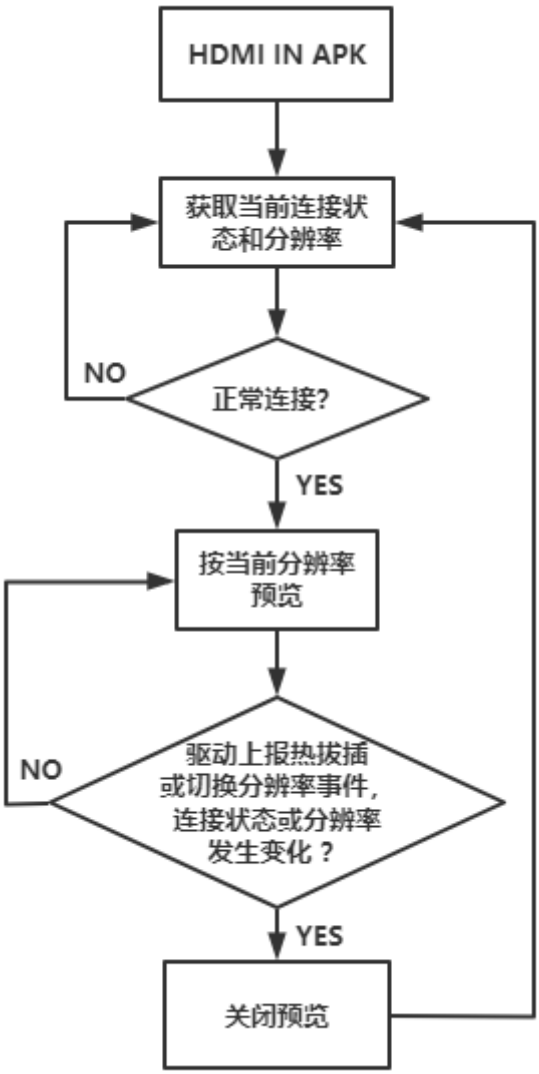
或是通过调用vidoe节点的ioctl接口来动态配置EDID：

```
static const struct v4l2_ioctl_ops hdmirx_v4l2_ioctl_ops = {
    ...
    .vidioc_g_edid = hdmirx_get_edid,
    .vidioc_s_edid = hdmirx_set_edid,
    ...
};
```

可通过工具编辑配置EDID，再替换到驱动代码中，EDID可视化编辑工具自行网上搜索下载。

3. HDMI IN Video架构

3.1 HDMI IN Video工作流程



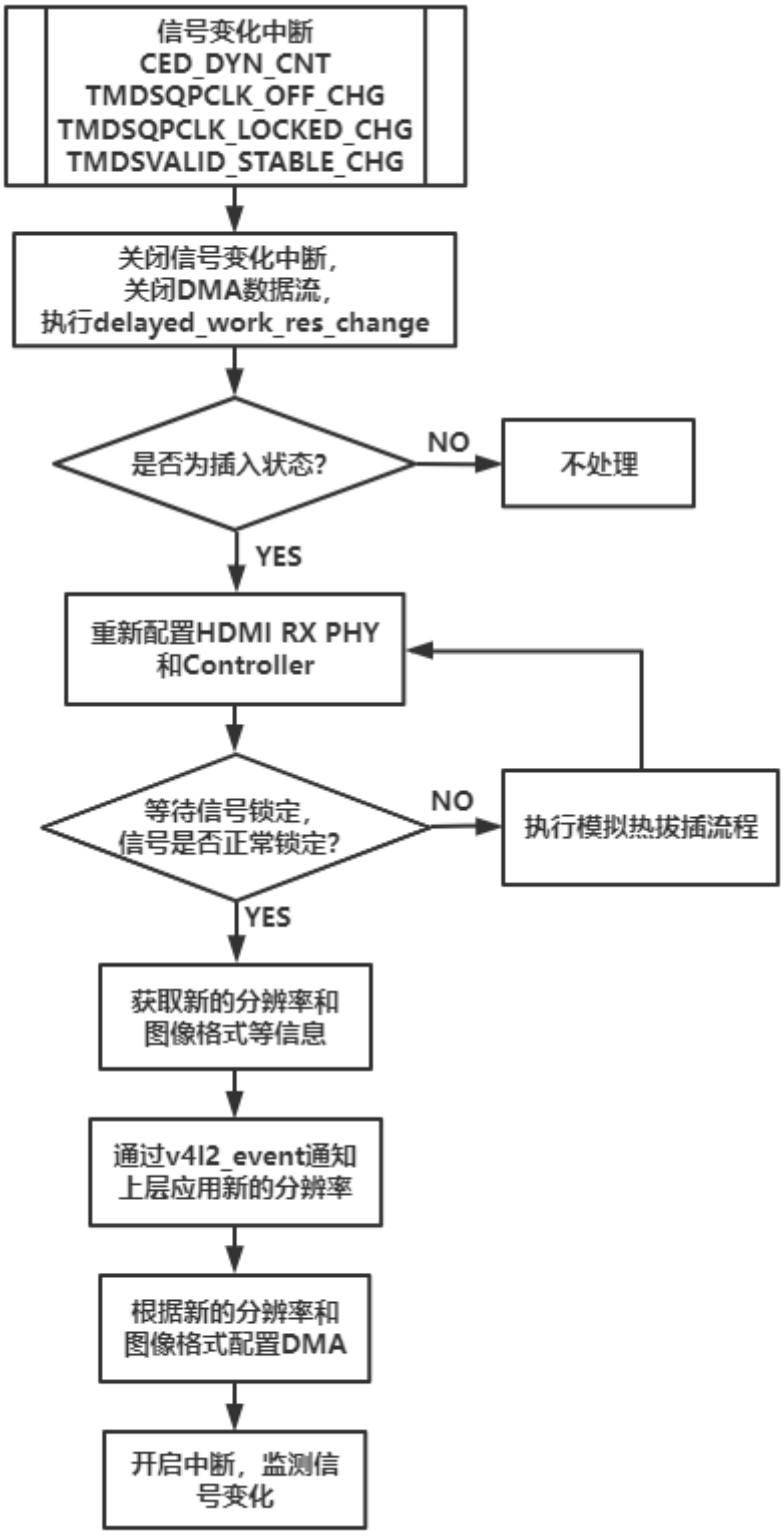
3.2 HDMI RX主要驱动架构

HDMI RX驱动架构中需要重点关注以下几个中断和delayed_work：

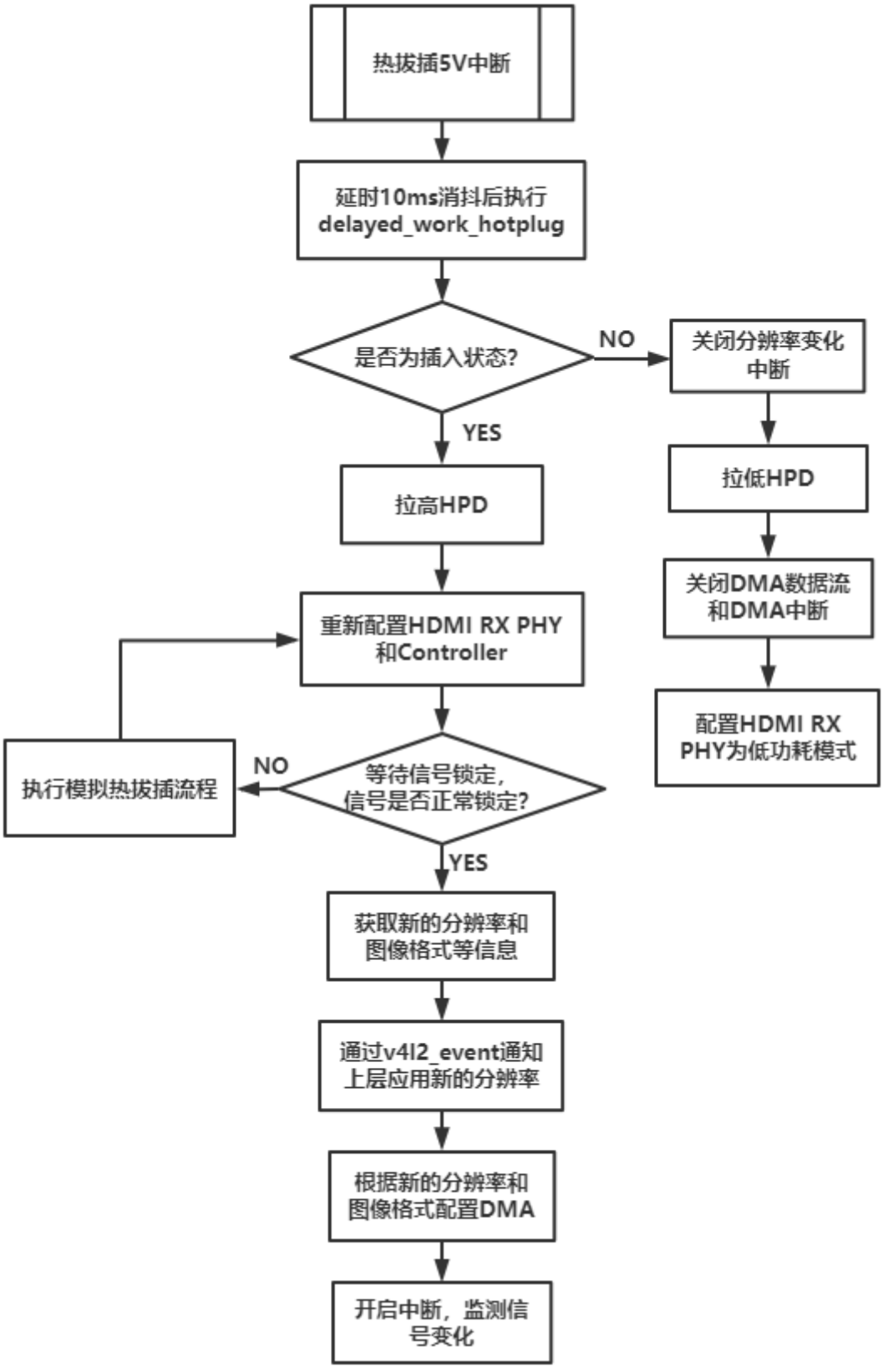
- hdmirx_5v_det_irq_handler：5V检测中断，由gpio引脚HDMIIRX_DET_L触发，用于检测HDMI接口的拔插动作。

- hdmirx_hdmi_irq_handler：HDMI RX控制器中断，主要用于初始化配置，以及监测HDMI信号变化时使用。
- hdmirx_dma_irq_handler：HDMI RX DMA中断，主要是在图像预览过程中Buffer转轮时使用。
- hdmirx_delayed_work_hotplug：5V检测中断对应的delayed_work，产生热拔插动作时，对HDMI RX模块进行相应的配置处理。
- hdmirx_delayed_work_res_change：HDMI RX控制器检测到信号变化中断时，对控制器进行重新配置，等待信号锁定。

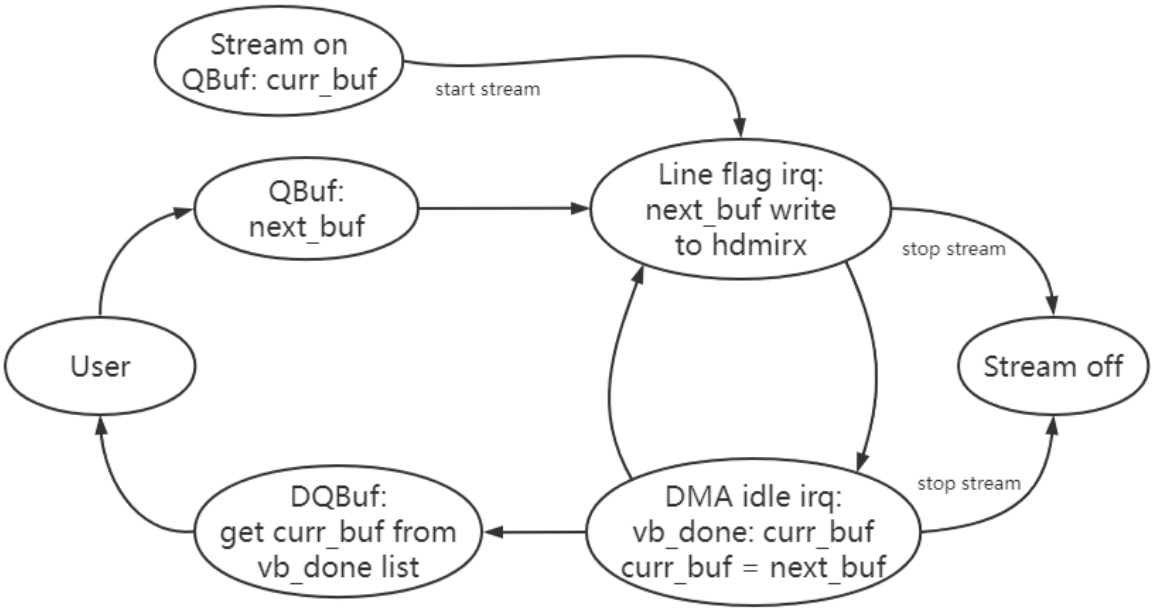
分辨率切换流程如下：



热拔插流程如下：



3.3 图像Buffer轮转机制



- QBuf/DQBuf：用户通过QBuf传入空闲buffer，通过DQBuf获取已经完成图像数据填充的buffer。
- Line flag irq：配置固定行数产生中断，当前是配置width/2行，即半帧时中断，在中断程序中更新buffer，buffer的物理地址写入HDMI RX控制器不会立即生效，在下一个vsync才会生效。因为vblank时间较短，可能会来不及更新buffer，导致下一帧内容覆盖到上一帧，所以需要提前更新buffer。
- DMA idle irq：DMA空闲中断，可以理解为frame end中断。一帧图像数据传输完成后，会产生此中断。中断后将buffer加入vb_done list，等待用户通过DQBuf来获取。
- Stream on：开流指令，初始buffer是使用curr_buf。
- Stream off：关流指令，用户发出关流指令后，驱动会等待中断后再停止数据流，之后归还所有buffer。

3.4 图像传输延时

- 对接TIF框架后加速显示，延时约为：20-30ms。
- 对接Camera框架，显示延时约为：100~120ms。

4. HDMI IN HDCP功能

HDCP1.4 或是 HDCP2.3 KEY 都需要客户自行到 HDCP 官网购买，RK3588 没有内置 HDCP KEY.

4.1 HDCP1.4

4.1.1 dts 配置

参考HDCP配置章节，配置为hdcp1x-enable;

4.1.2 Key 烧写

Key 拆分和转换工具从SDK获取：

```
RKTools/windows/Rockchip_HdcpKey_Writer_V1.0.1.7z 工具包下面有 KeyConvertor key 的拆分和转换工具，更新到 KeyConvertor-V2.0.5 可以拆分成一个bin文件只有一个key.
```

Key 烧写工具从SDK获取：

```
RKTools/windows/Rockchip_HdcpKey_Writer_V1.0.1.7z
```

- 先用 Key 拆分和转换工具，从原始的 Key 文件中拆出部分 Key， 然后把该 Key 转成 .skf 后缀的烧写Key.
- 如果不用RK提供的工具烧写，需要把Key拆成一个文件只包含一个Key，可以在提取原始Key的"每文件KEY个数"填入 1 ，点提取即可生成单独的Key文件，不需要转成 .skf 格式。
- 机器进入Loader模式，烧写工具勾选 "hdcp1.4Hdmirx" 和 "RK3588" 选项，RK3588 HDCP1.4 Key 增加了 AES 加密，所以 "Do AES" 为可选项，不选的话只有 SEED 加密，选上的话会在 SEED 加密的基础上再进行 AES 加密，更安全。然后导入上面的 .skf 文件，点击写入即可完成 Key 的烧写。

  说明：工具写完一个 Key 之后自动会跳到下一个 Key。
- HDCP1.4 Key 结构说明。

```
一个HDCP 1.4 KEY有308 Bytes，分别如下：
8 Bytes KSV: 5 Bytes KSV + 3 Bytes 0x0组成;
280 Bytes DPK;
20 Bytes SHA.
```

4.1.3 HDCP1.4 状态查看

```
cat /sys/class/misc/hdmirx_hdcp/status

HDCP Disable: hdcp1.4 没使能
HDCP1.4: Authenticated start: 认证过程中
HDCP1.4: Authenticated success: 认证成功
HDCP1.4: Authenticated failed: 认证失败
HDCP1.4: Unknown status: 未知状态
```

4.2 HDCP2.3

当前 HDCP2.3 Firmware 打包工具需要单独提供，所以需要先通过 Redmine 获取到 HDCP2.3 的工具包 。

4.2.1 dts 配置

参考HDCP配置章节，配置为hdcp2x-enable;

4.2.2 打包 firmware 和 启动服务

- WC_HDCP2_BASE_ESM_Firmware 解压到 Linux 环境下，然后把 hdcp_receivers.bin 拷贝到根目录和 tools/ 目录下，如果同时支持HDMITX的话，可以同时把 hdcp_transmitter.bin 也拷贝到相同的目录下。

```
DWC_HDCP2_BASE_ESM_Firmware$
cp  ../hdcp_receivers.bin .
cp  ../hdcp_receivers.bin tools/
```

- 打包 Firmware 和 RX KEY，执行 **build_rockchip_fw.sh** 脚本

```
DWC_HDCP2_BASE_ESM_Firmware$ ./build_rockchip_fw.sh
```

打包过程会有几个步骤会提示输入的：

1、选择打包固件类型，选择: 1 -build firmware for both HDMIRX and HDMITX，生成 **./firmware/hdpc2_hdmi.fw** 固件。

```
1 -build firmware for both HDMIRX and HDMITX
2 -build firmware for HDMITX only
3 -build firmware for DP
Choose the fw type: 1
```

2、需要创建 RX KEY 的数量，根据需求创建，比如10或是100等，生成单独的 KEY 存放到 **./rxkeys/hdcpkeys?-?/** 目录下的 **fw_hdcp_receivers_*.bin** 文件，如果客户用自己的工具烧写的话，可以直接烧写 **fw_hdcp_receivers_*.bin** 文件

```
Create Rx Key Number:
10
......
Create 10 keys to tools/rxkeys/hdcpkeys1-10/
```

3、如果需要用 RK 提供的工具烧写 KEY，需要打包成 .skf 文件，存放到**./rxkeys/hdcpkeys?-?/**目录下，如果不需要 RK 的工具烧写，这边可以选择 n。

```
Do want to pack these keys to .skf file? [y/n]:
y

10 file packed to ./rxkeys/hdcpkeys1-10/hdcprxkeys1-10.skf
```

- HDCP2.3 RX KEY 的烧写

1、把 hdcp2_hdmi.fw 放到 device/rockchip/rk3588/ 工程目录下，编译的时候会拷贝到 vendor/firmware/hdcp2_hdmi.fw。

2、用 RKDevInfoTool.exe 工具，选择 HDCP2X_HDMIRX 导入上面生成的 hdcprxkeys1-10.skf，进入 Loader 模式进行烧写。写完一个 Key，工具会自动跳到下一个 Key 等待烧写。

3、HDCP2.3 Key 不需要AES加密，所以这边只需要勾选 "RK3588" 和 "hdcp2x hdmirx"。

- hdcp2_rx_tx 服务

开机自动加载服务，如果出现认证异常，可以logcat | grep HDCP2 查看对应的log.

4.2.3 HDCP2.3 状态查看

```
cat /sys/class/misc/hdmirx_hdcp/status

HDCP Disable: HDCP没使能
HDCP2.3: Authenticated success: 认证成功
HDCP2.3: Authenticated failed: 认证失败
HDCP2.3: No dectypted: 源端没有开启HDCP2.x
```

4.3 HDCP KEY 烧写

1. 客户可以用 RK 提供的工具进行烧写，可以从 SDK 的 RKTools/windows/ 目录下获取到对应的工具。
2. 客户可以自己写应用来烧写 Key，这种情况下可以用 RK 提供的 libhdcp.so 库，然后应用直接调用库接口对 Key 进行加密并烧写到 vendor storage. SDK 默认没有包含该库，可以提 Redmine 来获取。

   libhdcp.so 包含以下接口：

```
enum HDCP_KEY_ID {
    HDCP1X_KEY_HDMITX_RK33 = 0,
    HDCP1X_KEY_HDMITX_RK3588,
    HDCP1X_KEY_HDMIRX_RK3588,
    HDCP1X_KEY_HDMIRX_RK628,
    HDCP1X_KEY_DP_RK3588,
    HDCP2X_KEY_HDMIRX_RK3588,
};

/*
 * Encrypt the key, but is not written to vendor
 *
 * id: HDCP KEY type
 * keyin: HDCP1.x 308 Byte raw Key, HDCP2.x 1000 Byte Key
 * keyin_size: HDCP1.x 308, HDCP2.x 1000
 * keyout: The encrypted key
 * keyout_size: HDCP1.x 314, HDCP2.x 1000
 */
int hdcp_key_process(enum HDCP_KEY_ID id, uint8_t *keyin, int keyin_size, uint8_t
*keyout, int keyout_size);

/*
 * Encrypt key and write it to vendor
 *
 * id: HDCP KEY type
 * keyin: HDCP1.x 308 Byte raw Key, HDCP2.x 1000 Byte Key
 * keyin_size: HDCP1.x 308, HDCP2.x 1000
 */
int hdcp_key_process_and_write(enum HDCP_KEY_ID id, uint8_t *keyin, int
keyin_size);

/*
 * Write key to vendor
 *
 * id: HDCP KEY type
```

```
 * key: HDCP1.x 308 Byte raw Key, HDCP2.x 1000 Byte Key
 * size: key size
 */
int hdcp_key_write(enum HDCP_KEY_ID id, uint8_t *key, int size);

/*
 * Read key from vendor
 *
 * id: HDCP KEY type
 * key: HDCP1.x 308 Byte raw Key, HDCP2.x 1000 Byte Key
 * len: key length
 */
int hdcp_key_read(enum HDCP_KEY_ID id, uint8_t *key, int *len);
```

5. HDMI IN CEC功能

device/rockchip/common 如下修改：

TX: ro.hdmi.device_type=4 (Playerback) ;

RX: ro.hdmi.device_type=0 (TV) ;

```
diff --git a/device.mk b/device.mk
index d600bf1..a5e9ae3 100644
--- a/device.mk
+++ b/device.mk
@@ -699,10 +699,10 @@ endif
 endif

 # hdmi cec
-ifneq ($(filter atv box, $(strip $(TARGET_BOARD_PLATFORM_PRODUCT))), )
+ifneq ($(filter atv box tablet, $(strip $(TARGET_BOARD_PLATFORM_PRODUCT))), )
 PRODUCT_COPY_FILES += \

frameworks/native/data/etc/android.hardware.hdmi.cec.xml:$(TARGET_COPY_OUT_VENDOR
)/etc/permissions/android.hardware.hdmi.cec.xml
-PRODUCT_PROPERTY_OVERRIDES += ro.hdmi.device_type=4
+PRODUCT_PROPERTY_OVERRIDES += ro.hdmi.device_type=0
 PRODUCT_PACKAGES += \
        hdmi_cec.$(TARGET_BOARD_PLATFORM)

+DEVICE_MANIFEST_FILE +=
device/rockchip/common/manifests/android.hardware.tv.cec@1.0-service.xml
 # HDMI CEC HAL
 PRODUCT_PACKAGES += \
diff --git a/manifests/android.hardware.tv.cec@1.0-service.xml
b/manifests/android.hardware.tv.cec@1.0-service.xml
new file mode 100644
index 00000000..5afa59d7
--- /dev/null
+++ b/manifests/android.hardware.tv.cec@1.0-service.xml
@@ -0,0 +1,11 @@
+<manifest version="1.0" type="device">
+    <hal format="hidl">
```

```
+        <name>android.hardware.tv.cec</name>
+        <transport>hwbinder</transport>
+        <version>1.0</version>
+        <interface>
+            <name>IHdmiCec</name>
+            <instance>default</instance>
+        </interface>
+    </hal>
+</manifest>
```

6. HDMI IN APK适配方法

6.1 APK源码路径

- packages/apps/TV/partner_support/samples ：提供TV源数据服务，通过framework与HAL层、预览APK进行交互，由于是开机运行的隐藏服务，该APK在桌面上是隐藏图标的。
- packages/apps/rkCamera2 ：预览apk，通过framework层与上述TV源数据服务进行交互，该APK在桌面上图标名称为 HdmiIn ，通常客户会二次开发替换为自己的预览APK。
- hardware/rockchip/tv_input ：HAL层代码，开关流、热拔插和分辨率切换事件等与驱动进行命令交互。
- SDK默认代码HDMI IN功能是关闭的，使能HDMI IN功能，需配置如下属性，开启后会编译含上述APK在内的相关模块：

```
vim device/rockchip/rk3588/BoardConfig.mk
BOARD_HDMI_IN_SUPPORT := true
```

开启BOARD_HDMI_IN_SUPPORT后会默认配置属性vendor.hwc.enable_sideband_stream_2_mode为l。机器开机后，先使用getprop检查该值是否设置成功。

6.2 HdmiIn预览APK说明

- MainActivity主界面，TIF(复用部分Android的Tv Input Framwork框架)预览方式，支持HDMI TO MIPI和HDMI RX通路数据预览，如需自己编写预览APK，需要使用标准的TvView控件。HDMI TO MIPI的驱动相关配置见文档《Android12+版本HDMIIN桥接芯片开发指南》。

```
String INPUT_ID =
"com.example.partnersupportsampletvinput/.SampleTvInputService/HW0";
Uri channelUri = TvContract.buildChannelUriForPassthroughInput(INPUT_ID);
//tvinput.hdmiin.type取值0为HDMI RX，1为HDMI TO MIPI。需在tune之前设置，不设置默认为
HDMI RX
//SystemProperties.set("tvinput.hdmiin.type", "1");
tvView.tune(INPUT_ID, channelUri);
```

注意：如果产品没有HDMI RX，只有HDMI TO MIPI，例如RK3576，以及RK3588 dts禁用HDMI RX的场景，需要把tvinput.hdmiin.type属性配置到系统中，否则TIF方案会黑屏无法预览HDMI TO MIPI，RK3576的举例配置如下：

```
device/rockchip/rk3576/device.mk
PRODUCT_PROPERTY_OVERRIDES += \
    tvinput.hdmiin.type=1
```

TIF预览方式，所使用的送显方式，需开启vtunnel配置以及使能vendor.hwc.enable_sideband_stream_2_mode属性，默认的3588 evb1开发板开启vtunnel的dts配置如下，不同机型在dts中做相应修改：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
index a10dad37f9cf..b3dd3a8bbb19 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
@@ -275,6 +275,10 @@ &dp1_in_vp2 {
        status = "okay";
 };

+&rkvtunnel {
+        status = "okay";
+};
+
 /*
  * mipi_dcphy0 needs to be enabled
  * when dsi0 is enabled
diff --git a/arch/arm64/boot/dts/rockchip/rk3588.dtsi
b/arch/arm64/boot/dts/rockchip/rk3588.dtsi
index fa4a7e35099e..e3acf575aefd 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588.dtsi
@@ -8,6 +8,10 @@
 #include "rk3588-vccio3-pinctrl.dtsi"

 / {
+        rkvtunnel: rkvtunnel {
+                compatible = "rockchip,video-tunnel";
+                status = "disabled";
+        };
        aliases {
                dp0 = &dp0;
                dp1 = &dp1;
diff --git a/arch/arm64/configs/rockchip_defconfig
b/arch/arm64/configs/rockchip_defconfig
index c742a4cc4fe5..039173d51bbe 100644
--- a/arch/arm64/configs/rockchip_defconfig
+++ b/arch/arm64/configs/rockchip_defconfig
@@ -684,6 +684,7 @@ CONFIG_ROCKCHIP_MPP_IEP2=y
 CONFIG_ROCKCHIP_MPP_JPGDEC=y
 CONFIG_ROCKCHIP_MPP_AV1DEC=y
 CONFIG_ROCKCHIP_MPP_VDPP=y
+CONFIG_ROCKCHIP_VIDEO_TUNNEL=y
 CONFIG_SOUND=y
 CONFIG_SND=y
 CONFIG_SND_DYNAMIC_MINORS=y
```

注意：TIF方式，需要5块buffer进行轮转，如果要支持3840x2160的BGR888、RGB888、NV24格式数据，理论CMA大小为`3840 * 2160 * 3 * 5 = 121500KB ≈ 119M`，加上系统本身可能会占用部分CMA资源以及碎片化的存在，建议分配多些，例如256M，如何调大请参考上文“2.3.2预留内存章节”。

像下方例子，logcat显示cma: Out of memory，且cat已申请的cma内存，看到少于5块，那就是CMA预留内存小了。

```
rk3588_u:/ # cat /d/dma_buf/bufinfo | grep cma
24883200       00000002        00080007        00000002        cma     00000416
     cma
24883200       00000002        00080007        00000002        cma     00000415
     cma
24883200       00000002        00080007        00000002        cma     00000414
     cma
24883200       00000002        00080007        00000002        cma     00000413
     cma
rk3588_u:/ #

W tv_input_Sideband: dequeueBuffer 259 do allocateBuffer
I android.hardware.graphics.allocator-V1-service: alloc_format: FMT:0x113,MOD:0
I android.hardware.graphics.allocator-V1-service: mali_gralloc_adjust_dimensions:
alloc_format=FMT:0x113,MOD:0 usage=0x11000000a0000833 alloc_width=3840,
alloc_height=2160
D cma     : __cma_alloc(cma 00000000a43238b2, count 6075, align 8)
E cma     : __cma_alloc: cma: alloc failed, req-size: 6075 pages, ret: -12
I cma     : number of available pages:
1@543+1@575+4@588+2@598+2@606+2@638+2@1694+2@1726+2@1758+2@1766+2@1774+2@1782+2@1
790+2@1798+2@1806+2@1814+2@1822+2@1830+2@1838+2@1846+2@1854+2@1862+2@1870+2@1878+
2@1886+2@1894+2@1902+2@1910+2@1918+2@1926+2@1934+2@1942+2@1950+2@1958+2@1966+2@19
74+2@1982+2@1990+2@1998+2@2006+2@2014+2@2022+2@2030+2@2038+2@2046+2@2054+2@2062+2
@2070+2@2078+2@2086+2@2094+2@2102+2@2110+2@2118+2@2126+2@2134+2@2142+2@2150+2@215
8+2@2166+2@2174+2@2182+114@2190+69@8379+69@14523+69@20667+5957@26811=> 6402 free
of 32768 total pages
D cma     : __cma_alloc(): returned 0000000000000000
E DMABUFHEAPS: Unable to allocate from DMA-BUF heap: cma: Out of memory
E DMABUFHEAPS: No ion heap of name cma exists
E DMABUFHEAPS: No ion heap of name cma exists
E android.hardware.graphics.allocator-V1-service: libdmabufheap allocation failed
for cma heap
E android.hardware.graphics.allocator-V1-service: buffer allocation failed: Out
of memory
E android.hardware.graphics.allocator-V1-service: buffer allocation failed: Out
of memory
E GraphicBufferAllocator: Failed to allocate (3840 x 2160) layerCount 1 format 40
usage 11000000a0000833: 1
```

- RockchipCamera2界面，Camera预览方式，支持HDMI TO MIPI与HDMI RX通路预览。默认情况下，点击app图标，启动的是MainActivity界面，如需启用camera通路预览，请先使能HDMI IN的camera功能，配置属性：

```
vim device/rockchip/rk3588/BoardConfig.mk
CAMERA_SUPPORT_HDMI := true
```

同时需要配置属性persist.sys.hdmiinmode值为2，此时点击HdmiIn应用，启动的是RockchipCamera2通过camera方式预览数据界面，也可以用系统自带camera相机进行预览。

6.3 TIF与Camera预览方式差异

	TIF	Camera
优点	延迟低	app端能够拿到预览数据进行后处理
缺点	不支持屏幕旋转、分屏、画中画、异显功能； app端拿不到预览buffer数据； 不支持screencap方式的截图命令	延迟高于TIF

1. TIF预览不支持画中画功能，要使用画中画，可以从TIF切换为camera方案。在TIF预览示例MainActivity中，点击屏幕任一位置，弹出框的“PIP”按钮，提供了从TIF预览切换到camera画中画预览的功能，要查看该效果，前提需确保CAMERA_SUPPORT_HDMI := true，即camera预览方案是使能状态。
2. TIF预览下，支持谷歌标准的MediaProjectionManager创建虚拟屏方式进行录像与截图，也支持screenrecord命令录屏。

   如果有上述录屏和截图需求，或者需要pc通过adb投屏进行投屏操作，需要配置属性：

   ```
   vim device/rockchip/rk3588/device.mk
   PRODUCT_PROPERTY_OVERRIDES += debug.sf.enable_hwc_vds=true
   ```

   如果有重载过 device 下的产品目录，需将其配置在对应产品的目录下，可在开机后通过 adb 执行 getprop debug.sf.enable_hwc_vds 查看打印值是否为 true确认是否改动有效。
3. TIF预览下，不支持adb的screencap命令截图，包括音量键+电源键这类快捷按钮截图，需要上述第2点提到的谷歌标准MediaProjectionManager虚拟屏截图。

7. 驱动调试方法

7.1 调试工具获取

调试需要使用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/
```

7.2 调试命令举例

一般在调试分析问题时，建议配置开启debug log，参考章节打开log开关。

7.2.1 查看HDMIRX的video节点

```
grep -H hdmirx /sys/class/video4linux/video*/name
```

7.2.2 查找rk_hdmirx设备

使用v4l2-ctl -d参数指定vidoe节点，-D命令查看节点信息，通过Driver name确认哪个是节点是rk_hdmirx设备：

```
rk3588_s:/ # v4l2-ctl -d /dev/video17 -D
Driver Info:
        Driver name      : rk_hdmirx
        Card type        : rk_hdmirx
        Bus info         : fdee0000.hdmirx-controller
        Driver version   : 5.10.66
        Capabilities     : 0x84201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps      : 0x04201000
                Video Capture Multiplanar
                Streaming
                Extended Pix Format
```

7.2.3 获取驱动tmings信息

获取设备在信号锁定时保存的timings信息：

```
rk3588_s:/ # v4l2-ctl -d /dev/video17 --get-dv-timings
DV timings:
        Active width: 3840
        Active height: 2160
        Total width: 4400
        Total height: 2250
        Frame format: progressive
        Polarities: -vsync -hsync
        Pixelclock: 594024000 Hz (60.00 frames per second)
        Horizontal frontporch: 172
        Horizontal sync: 92
        Horizontal backporch: 296
        Vertical frontporch: 8
        Vertical sync: 10
        Vertical backporch: 72
        Standards:
        Flags:
```

7.2.4 实时查询timings信息

实时从HDMI RX的寄存器读取timings信息:

```
rk3588_s:/ # v4l2-ctl -d /dev/video17 --query-dv-timings
        Active width: 3840
        Active height: 2160
        Total width: 4400
```

```
        Total height: 2250
        Frame format: progressive
        Polarities: -vsync -hsync
        Pixelclock: 594024000 Hz (60.00 frames per second)
        Horizontal frontporch: 172
        Horizontal sync: 92
        Horizontal backporch: 296
        Vertical frontporch: 8
        Vertical sync: 10
        Vertical backporch: 72
        Standards:
        Flags:
```

执行query-dv-timings时，若debug等级配置为2，通过dmesg查看kernel log，其中会打印详细的timings信息，以及pixel fmt，color depth，tmds clk等信息，参考如下：

```
[16750.029542][ T2247] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
YUV422
[16750.029581][ T2247] fdee0000.hdmirx-controller: hdmirx_get_colordepth:
color_depth: 24, reg_val:4
[16750.029592][ T2247] fdee0000.hdmirx-controller: get timings from dma
[16750.029602][ T2247] fdee0000.hdmirx-controller: act:3840x2160,
total:4400x2250, fps:60, pixclk:594024000
[16750.029610][ T2247] fdee0000.hdmirx-controller: hfp:172, hs:92, hbp:296,
vfp:8, vs:10, vbp:72
[16750.029618][ T2247] fdee0000.hdmirx-controller: tmds_clk:594024000
[16750.029626][ T2247] fdee0000.hdmirx-controller: interlace:0, fmt:1, vic:127,
color:24, mode:hdmi
[16750.029633][ T2247] fdee0000.hdmirx-controller: deframer_st:0x11
[16750.029643][ T2247] fdee0000.hdmirx-controller: query_dv_timings:
3840x2160p60.00 (4400x2250)
```

7.2.5 查询分辨率和图像格式

查询当前的分辨率和图像格式：

```
rk3588_s:/ # v4l2-ctl -d /dev/video17 --get-fmt-video
Format Video Capture Multiplanar:
        Width/Height      : 3840/2160
        Pixel Format      : 'NV16'
        Field             : None
        Number of planes  : 1
        Flags             : premultiplied-alpha, 000000fe
        Colorspace        : Unknown (1025fcdc)
        Transfer Function : Unknown (00000020)
        YCbCr Encoding    : Unknown (000000ff)
        Quantization      : Default
        Plane 0           :
           Bytes per Line : 3840
           Size Image     : 16588800
```

7.2.6 开启图像数据流

开启图像数据流，需要根据实际情况配置正确的video节点、分辨率、pixelformat：

```
v4l2-ctl --verbose -d /dev/video17 \
--set-fmt-video=width=3840,height=2160,pixelformat='NV16' \
--stream-mmap=4
```

注：RGB888->'RGB3'、YUV422->'NV16'、YUV420->'NV12'、YUV444->'NV24'

7.2.7 抓取图像文件

保存图像文件到设备，可adb pull到PC端，通过7yuv、YUView等工具软件查看：

```
v4l2-ctl --verbose -d /dev/video17 \
--set-fmt-video=width=3840,height=2160,pixelformat='NV16' \
--stream-mmap=4 --stream-skip=3 \
--stream-to=/data/4k60_nv16.yuv \
--stream-count=5 --stream-poll
```

7.2.8 正常取流log

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      : 3840/2160
        Pixel Format      : 'NV16'
        Field             : None
        Number of planes  : 1
        Flags             : premultiplied-alpha, 000000fe
        Colorspace        : Unknown (1025fcdc)
        Transfer Function : Unknown (00000020)
        YCbCr Encoding    : Unknown (000000ff)
        Quantization      : Default
        Plane 0           :
           Bytes per Line : 3840
           Size Image     : 16588800
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: 16588800 ts: 103.172405
```

```
idx: 1 seq:      1 bytesused: 16588800 ts: 103.189072 delta: 16.667 ms
idx: 2 seq:      2 bytesused: 16588800 ts: 103.205738 delta: 16.666 ms
idx: 3 seq:      3 bytesused: 16588800 ts: 103.222404 delta: 16.666 ms
idx: 0 seq:      4 bytesused: 16588800 ts: 103.239070 delta: 16.666 ms fps: 60.00
idx: 1 seq:      5 bytesused: 16588800 ts: 103.255736 delta: 16.666 ms fps: 60.00
idx: 2 seq:      6 bytesused: 16588800 ts: 103.272402 delta: 16.666 ms fps: 60.00
```

8. 常见问题调试方法

8.1 打开log开关

- 可参考如下命令配置HDMI RX驱动的debug log等级：0-3。

```
echo 2 > /sys/module/rockchip_hdmirx/parameters/debug
dmesg -n 8
```

- 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，参考如下补丁：

```
diff --git a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
index c763a9558169..bd7f3effb45a 100644
--- a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
+++ b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
@@ -34,7 +34,7 @@
 #include "rk_hdmirx_cec.h"

static struct class *hdmirx_class;
-static int debug;
+static int debug = 2;
 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)
```

注：一般情况不建议配置debug等级为3，因为会打印大量的log。

8.2 通过io命令读写寄存器

- 可通过io命令读写HDMI RX的寄存器，需要在kernel config中使能以下配置：

```
CONFIG_DEVMEM=y
```

- io命令查询寄存器举例：

```
// 通过以下命令可查看io使用帮忙：
io -h

rk3588_s:/ # io -4 -l 0xc 0xfdee0594
fdee0594:  0000000f 80008000 00008000
```

8.3 HDMI RX状态查询

- 查询HDMI RX当前状态，包括信号锁定情况、图像格式、Timings信息、Pixl Clk等：

```
rk3588_s:/ # cat /d/hdmirx/status
status: plugin
Clk-Ch:Lock     Ch0:Lock        Ch1:Lock        Ch2:Lock
Ch0-Err:0       Ch1-Err:0       Ch2-Err:0
Color Format: YUV422                    Store Format: YUV422 (8 bit)
Mode: 3840x2160p60 (4400x2250)          hfp:172  hs:92  hbp:296  vfp:8  vs:10
vbp:72
Pixel Clk: 594024000
```

- 查询HDMI RX控制器寄存器信息：

```
rk3588_s:/ # cat /d/hdmirx/ctrl

--------------------hdmirx ctrl--------------------
00000000: 48515258 30313130 6c753031 01000310
00000010: Reserved 31353138 30333039 32303231
00000020: WO...... 00211f01 198b7b25
00000040: WO...... 00001001 00000000
00000050: 00000001
...
-----------------------------------------------------
```

- 查询HDMI RX PHY寄存器信息：

```
rk3588_s:/ # cat /d/hdmirx/phy

--------------------hdmirx phy--------------------
0000004f: 01000000
0000100f: 01000000
0000110f: 01000000
0000120f: 01000000
0000130f: 01000000
0000104a: 01002600
...
-------------------------------------------------
```

- 查询图层信息：

```
rk3588_s:/ # cat /d/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: DISABLED
Video Port3: ACTIVE
    Connector: DSI-1
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[0] color_space[0], eotf:0
    Display mode: 1080x1920p60
        clk[132000] real_clk[132000] type[48] flag[a]
        H: 1080 1095 1099 1129
        V: 1920 1935 1937 1952
    Esmart3-win0: ACTIVE        //HDMI-IN预览图层
        win_id: 11
        format: RG24 little-endian (0x34324752) SDR[0] color_space[0]
glb_alpha[0xff]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 0
        src: pos[0, 0] rect[1920 x 1080]
        dst: pos[0, 0] rect[1080 x 1920]
        buf[0]: addr: 0x00000000f070a000 pitch: 5760 offset: 0
    Cluster3-win0: ACTIVE
        win_id: 6
        format: AB24 little-endian (0x34324241)[AFBC] SDR[0] color_space[0]
glb_alpha[0xff]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 1
        src: pos[0, 0] rect[1080 x 1920]
        dst: pos[0, 0] rect[1080 x 1920]
        buf[0]: addr: 0x00000000ef903000 pitch: 4352 offset: 0
```

8.4 HDMI IN信号不锁定问题

HDMI IN信号不锁异常log如下：

```
[  285.949990][  T191] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing signal not lock, tmds_clk_ratio:0
```

排查分析步骤如下：

- 测量插入检测引脚HDMIIRX_DET_L电平是否符合设计预期，拔插HDMI接口，是否会有电平跳变。
- 在HDMI插入时，在HDMI端口处测量HDMI_RX_HPD_PORT是否正常拉高，拔出时是否会拉低。
- 在HDMI插入时，用示波器实测TMDS信号是否正常输出。
- 根据log提示确认SCDC是否正常交互，HDMI协议规定，在pixel clk大于340M时tmds_clk_ratio需要配置为1。假设当前源端输出图像为4K60，pixel clk 594M，但log提示tmds_clk_ratio:0，则说明SCDC没有正常交互，需要检查HDMI的DDC通讯情况，或是拔插重试；
- 查询寄存器状态：

```
console:/ # io -4 0xfdee0050
fdee0150: 00000001 // bit0：1表示HPD拉高，0表示HPD拉低

// 0x0594：bit[3:0]表示scdc_ch2locked、scdc_ch1locked、scdc_ch0locked、
scdc_clockdetected；
// 0x0598：bit[31]:scdc_err_det1_valid，bit[15]:scdc_err_det0_valid，低bit为误码数量
统计；
// 0x059c：bit[31]:scdc_erdet_lane0_valid，bit[15]:scdc_err_det2_valid，低bit为误码
数量统计；
console:/ # io -4 -l 0xc 0xfdee0594
fdee0594: 0000000f 80008000 00008000 // 正常锁定时的值
```

- 部分设备拔插概率性lock，可尝试延长wait lock的等待时间，确认能否完成锁定：

```
diff --git a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
index 39e4e15a6e17..a612fe30bda4 100644
--- a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
+++ b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
@@ -1264,7 +1264,7 @@ static int hdmirx_wait_lock_and_get_timing(struct
rk_hdmirx_dev *hdmirx_dev)
        u32 mu_status, scdc_status, dma_st10, cmu_st;
        struct v4l2_device *v4l2_dev = &hdmirx_dev->v4l2_dev;

-       for (i = 0; i < 300; i++) {
+       for (i = 0; i < 600; i++) {
                mu_status = hdmirx_readl(hdmirx_dev, MAINUNIT_STATUS);
                scdc_status = hdmirx_readl(hdmirx_dev, SCDC_REGBANK_STATUS3);
                dma_st10 = hdmirx_readl(hdmirx_dev, DMA_STATUS10);
@@ -1283,7 +1283,7 @@ static int hdmirx_wait_lock_and_get_timing(struct
rk_hdmirx_dev *hdmirx_dev)
                hdmirx_tmds_clk_ratio_config(hdmirx_dev);
        }

-       if (i == 300) {
+       if (i == 600) {
                v4l2_err(v4l2_dev, "%s signal not lock, tmds_clk_ratio:%d\n",
                                __func__, hdmirx_dev->tmds_clk_ratio);
                v4l2_err(v4l2_dev, "%s mu_st:%#x, scdc_st:%#x, dma_st10:%#x\n",
```

- 部分设备在切分辨率以后容易出现锁定失败的情况，可尝试在拉搞HPD前增加一些延时，确认是否有改善：

```
diff --git a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
index 8183485a6e4c..27b900e90501 100644
--- a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
+++ b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
@@ -2768,6 +2768,7 @@ static void hdmirx_delayed_work_res_change(struct
work_struct *work)
                hdmirx_submodule_init(hdmirx_dev);
                hdmirx_update_bits(hdmirx_dev, SCDC_CONFIG, POWERPROVIDED,
                                   POWERPROVIDED);
+               msleep(300);
                hdmirx_hpd_ctrl(hdmirx_dev, true);
                hdmirx_phy_config(hdmirx_dev);
                hdmirx_audio_setup(hdmirx_dev);
```

- 部分信号发生器或者信号源可能存在信号质量较差的情况，接入hdmirx后会检测到部分channel误码较高，导致无法锁定。通过读取0xfdee0594-0xfdee059c可寄存器判断是否存在误码：

```
rk3588_s:/ # io -4 -l 0xc 0xfdee0594
fdee0594:  0000000f 80008000 00008000       //正常数值

rk3588_s:/ # io -4 -l 0xc 0xfdee0594
fdee0594:  0000000f 804c8001 00008aff       //检测到误码，分别对应ch1、ch0、ch2统计的
误码总数
```

某个通道存在误码导致不锁定，可手动关闭驱动内部的误码检测功能增强兼容性：

```
diff --git a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
index 606a08cd00505..6236196991c2a 100755
--- a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
+++ b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
@@ -1416,11 +1416,7 @@ static void hdmirx_controller_init(struct rk_hdmirx_dev
*hdmirx_dev)
                           CED_GBCHECKEN_QST |
                           CED_CTRLCHECKEN_QST |
                           CED_CHLOCKMAXER_QST_MASK,
-                          CED_VIDDATACHECKEN_QST|
-                          // CED_DATAISCHECKEN_QST |
-                          CED_GBCHECKEN_QST |
-                          CED_CTRLCHECKEN_QST |
-                          CED_CHLOCKMAXER_QST(0x10));
+                          0);
 }

 static void hdmirx_format_change(struct rk_hdmirx_dev *hdmirx_dev)
```

8.5 HDMI IN不出图、黑屏问题

HDMI IN不出图问题可分两种情况讨论，驱动部分异常或者上层及应用功能异常。简单排查思路如下：

驱动排查：

1. 判断kernel log是否锁定，分辨率读取是否正常。若有使用switch，还需要判断switch是否正常锁定输出；（参考日志：拔插日志）
2. 通过节点信息判断各通道是否锁定，分辨率是否同源端一致；（cat /d/hdmirx/status）
3. 通过日志或者寄存器状态判断是否正常开流，io -4 0xfdee4414 若bit1则正常开流；
4. log等级开到3看看是否有帧数据，每半帧会打印一句log；

上层排查：

1. 执行date命令,查看当前问题大概时间点；
2. cat /d/dri/0/summary 检查一下图层信息；
3. dumpsys SurfaceFlinger；
4. 保存logcat日志；
5. 确认有出流以及第2步有图层后，抓图确认。命令如下:

```
android12：
rm -rf data/system/dumpimage
mkdir data/system/dumpimage
chmod 777 data/system/dumpimage
setprop vendor.hdmiin.debug.dump 1
setprop vendor.hdmiin.debug.dumpnum 5
1s后立马
setprop vendor.hdmiin.debug.dumpnum 0
2s后立马
setprop vendor.hdmiin.debug.dump 0

android13：
rm -rf data/system/dumpimage
mkdir data/system/dumpimage
chmod 777 data/system/dumpimage
setprop vendor.tvinput.debug.dump 1
setprop vendor.tvinput.debug.dumpnum 10
半s后立马
setprop vendor.tvinput.debug.dumpnum 0
等个2s后
setprop vendor.tvinput.debug.dump 0
```

   看下data/system/dumpimage有没有文件，打包data/system/dumpimage并提供。
   如果没看到有文件，可能是selinux的关系，setenforce 0后重新敲上面的命令

6. 保留现场，不要做多余操作，提供上述信息。如果复现时，无法及时联系上我们，无法确保机器长时间保持现场，这种紧急情况，在上面信息已抓取的情况下，特别是要确认logcat已抓取的情况下，可以敲以下命令抓取10s信息。（由于会操作logcat -c ，清除之前的log，非必要不要执行这一步，最好先联系我们）

```
setprop vendor.tvinput.debug.level 3
logcat -c
logcat
抓10s
setprop vendor.tvinput.debug.level 0
```

特殊排查---出流和抓图正常，但画面为黑屏可能是avmute状态引起的。通过一下命令排查：

1. 检查avmute状态位：

```
io -4 0xfdee4458    //bit25 avmute statue
io -4 0xfdee1030    //bit 0 GCP Set_AVMUTE bit4 GCP Clear_AVMUTE
```

2. 修改avmute颜色确认：（默认黑色）

```
io -4 0xfdee0430 0xff00 //设置RGB格式avmute为红色,yuv格式为其他颜色，若变色说明是avmute导致。
```

   部分网页或者广告页面会进行hdcp加密，若sink端未使能hdcp功能可能会导致avmute。

8.6 信号不稳定失锁问题

测试过程中可能会发现部分信号源不稳定，预览过程中概率性闪黑屏，或者log一直在持续尝试锁定获取分辨率，导致开流失败。常见于一些信号发生器、电视盒子、DVD等设备。可能原因如下：（参考日志：信号锁定后失锁）

- source端有hdcp加密，但sink端没有使能hdcp功能导致不锁定；

  常见的hdcp加密设备有：苹果电脑、DVD碟机、电视盒子等。
- 线材质量较差、线材过长、座子松动等导致输入到soc端的hdmi信号不符合spec规范；

  更换线材测试是否稳定，判断是否仅个别线材异常。
- 前级switch的驱动强度、swing等配置过高，在pcb走线上产生了信号反射等不良；

  适当修改前级switch的驱动强度、swing、端接电阻等，具体可协调switch原厂配合。
- 部分中断设置灵敏度过高，检测到信号变化后断联重试；

  关闭预览后从log判断具体触发了哪个中断，注释后是否正常。（建议提交redmine联调）
- 源端初始一段时间输出信号不稳定；

  增加延时，等待信号稳定后再上报分辨率。

```
diff --git a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
index a69f8f6bd577..ede8d91f06dc 100644
--- a/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
+++ b/drivers/media/platform/rockchip/hdmirx/rk_hdmirx.c
@@ -1674,7 +1675,7 @@ static int hdmirx_wait_lock_and_get_timing(struct
rk_hdmirx_dev *hdmirx_dev)
        }

        hdmirx_reset_dma(hdmirx_dev);
-       usleep_range(200*1000, 200*1010);
+       usleep_range(500*1000, 500*1010);
        hdmirx_format_change(hdmirx_dev);

        return 0;
```

8.7 源端切分辨率失败

常见于苹果电脑，N卡显卡、部分国产操作系统的笔电等设备。驱动在切分辨率过程中默认会拉低再拉高HPD尝试重连，导致源端认为信号sink断开取消动作。可修改为HPD为GPIO控制，保持切分辨率过程HPD常高。

提交redmine获取相应补丁：rk3588_hdmirx_config_hpd_as_gpio_config_scdc_pwr_asap_230513.patch

8.8 驱动统计TMDSCLK错误

常见于4k@60 RGB格式的输入源，源端输入TMDSCLK为594M，统计为148.5M，引发帧率和音频采样率错误，参考日志：TMDSCLK统计错误。

提交redmine获取相应补丁：rk3588_a12_hdmirx_patch_fps_err_0714.patch

8.9 源端特殊场景闪黑屏

常见于打开部分网页（优酷、blibli）、或者放大缩小某些窗口，导致信号变化或者误码增多，间接触发中断或者失锁。

1. 触发中断导致信号重连：通过关闭预览测试可从log打印中观察触发了具体什么中断，关闭对应中断是否可以正常。(建议提交redmine)
2. 误码增多导致失锁：关闭误码检测功能，参考章节：HDMI IN信号不锁定问题。

9. 典型日志说明

一般需要配置debug等级为2，通过dmesg才会输出相关日志，或是直接修改代码，参考章节：打开log开关。

9.1 拔插日志

```
// 检测到拔出动作
[29830.165185][ T2655] fdee0000.hdmirx-controller: hdmirx_delayed_work_hotplug:
plugin:0
// 关闭中断
[29830.165203][ T2655] fdee0000.hdmirx-controller: hdmirx_interrupts_setup:
disable
// 拉低HPD
[29830.165211][ T2655] fdee0000.hdmirx-controller: hdmirx_hpd_ctrl: disable,
hpd_trigger_level:1
[29830.177109][  T598] fdee0000.hdmirx-controller: hdmirx_query_dv_timings port
has no link!
...
// 检测到插入动作
[29843.128833][ T2655] fdee0000.hdmirx-controller: hdmirx_delayed_work_hotplug:
plugin:1
// 拉高hpd
[29843.139653][ T2655] fdee0000.hdmirx-controller: hdmirx_hpd_ctrl: enable,
hpd_trigger_level:1
...
[29843.247796][ T2655] fdee0000.hdmirx-controller: wait_reg_bit_status:  i:0,
time: 10ms
[29843.286353][ T2655] fdee0000.hdmirx-controller: wait_reg_bit_status:  i:38,
time: 50ms
[29843.286373][ T2655] rk_hdmirx fdee0000.hdmirx-controller:
hdmirx_audio_interrupts_setup: 1
// HDMI信号锁定
[29843.703996][ T2655] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing signal lock ok, i:54!
...
// 图像格式
[30098.978794][ T2655] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
YUV422
[30098.978803][ T2655] fdee0000.hdmirx-controller: hdmirx_get_colordepth:
color_depth: 24, reg_val:4
// 详细分辨率timing
[30098.978813][ T2655] fdee0000.hdmirx-controller: get timings from ctrl
[30098.978819][ T2655] fdee0000.hdmirx-controller: act:1920x1080,
total:2200x1125, fps:60, pixclk:148500000
[30098.978825][ T2655] fdee0000.hdmirx-controller: hfp:84, hs:48, hbp:148, vfp:4,
vs:5, vbp:36
// TMDS CLK
[30098.978829][ T2655] fdee0000.hdmirx-controller: tmds_clk:148500000
[30098.978835][ T2655] fdee0000.hdmirx-controller: interlace:0, fmt:1, vic:127,
color:24, mode:hdmi
[30098.978840][ T2655] fdee0000.hdmirx-controller: deframer_st:0x11
...
// 上报分辨率变化事件
[30099.023034][ T2655] fdee0000.hdmirx-controller: hdmirx_format_change: New
format: 1920x1080p60.00 (2200x1125)
[30099.023039][ T2655] fdee0000.hdmirx-controller: hdmirx_format_change: queue
res_chg_event
```

9.2 切换分辨率日志

```
// 信号变化，检测到数据误码中断
[  312.662740][    C4] fdee0000.hdmirx-controller: avpunit_0_int_handler:
avp0_st:0x700000
[  312.662750][    C4] fdee0000.hdmirx-controller: mu2_st:0x2
// TMDS信号变化中断
[  312.662756][    C4] fdee0000.hdmirx-controller: mainunit_2_int_handler:
TMDSVALID_STABLE_CHG
[  312.662760][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  312.688916][  T196] fdee0000.hdmirx-controller: hdmirx_delayed_work_audio: no
supported fs(0), cur_state 0
[  312.688928][  T196] rk_hdmirx fdee0000.hdmirx-controller: audio off
// 进入切换分辨率流程，当前HDMI状态为插入
[  312.723309][  T196] fdee0000.hdmirx-controller:
hdmirx_delayed_work_res_change: plugin:1
[  312.723316][  T196] fdee0000.hdmirx-controller: hdmirx_interrupts_setup:
disable
[  312.724986][    C4] fdee0000.hdmirx-controller: mu0_st:0x4000000
[  312.724991][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
// 相关配置完成后拉高HPD
[  312.725000][  T196] fdee0000.hdmirx-controller: hdmirx_hpd_ctrl: enable,
hpd_trigger_level:1
...
// 信号锁定
fdee0000.hdmirx-controller: hdmirx_wait_lock_and_get_timing signal lock ok, i:2!
...
// 获取新的分辨率，图像格式等
[  312.849040][  T196] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
YUV420
[  312.849049][  T196] fdee0000.hdmirx-controller: hdmirx_get_colordepth:
color_depth: 24, reg_val:4
[  312.849056][  T196] fdee0000.hdmirx-controller: get timings from ctrl
[  312.849060][  T196] fdee0000.hdmirx-controller: act:3840x2160,
total:4400x2250, fps:60, pixclk:296996000
[  312.849064][  T196] fdee0000.hdmirx-controller: hfp:84, hs:48, hbp:148, vfp:8,
vs:10, vbp:72
[  312.849067][  T196] fdee0000.hdmirx-controller: tmds_clk:296996000
[  312.849070][  T196] fdee0000.hdmirx-controller: interlace:0, fmt:3, vic:127,
color:24, mode:hdmi
```

9.3 信号未锁定异常日志

```
// 信号未锁定
[  285.949990][  T191] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing signal not lock, tmds_clk_ratio:0
[  285.950011][  T191] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing mu_st:0x0, scdc_st:0x0, dma_st10:0x10
...
// 读取到错误的分辨率
[  257.222739][  T193] fdee0000.hdmirx-controller: get timings from dma
[  257.222744][  T193] fdee0000.hdmirx-controller: act:39024x0, total:17732x1,
fps:8375, pixclk:148500000
[  257.222749][  T193] fdee0000.hdmirx-controller: hfp:4294904560, hs:184,
hbp:41260, vfp:1, vs:0, vbp:0
[  257.222753][  T193] fdee0000.hdmirx-controller: tmds_clk:148500000
[  257.222758][  T193] fdee0000.hdmirx-controller: interlace:0, fmt:0, vic:127,
color:24, mode:dvi
// 连续读取到错误分辨率
[  257.254994][  T193] fdee0000.hdmirx-controller: hdmirx_try_to_get_timings: res
not stable!
```

9.4 驱动开关流日志

```
// 开流使能dma，获取到30帧数据打印
[ 2360.096789][  T473] fdee0000.hdmirx-controller: hdmirx_start_streaming:
start_stream cur_buf y_addr:0x10700000, uv_addr:0x10700000
[ 2360.096813][  T473] fdee0000.hdmirx-controller: hdmirx_start_streaming: enable
dma
[ 2360.689582][  T205] fdee0000.hdmirx-controller: rcv frames
...
// 关流
[ 2475.818301][  T473] fdee0000.hdmirx-controller: stream start stopping
[ 2475.818575][  T205] fdee0000.hdmirx-controller: hdmirx_dma_irq_handler: stop
stream!
[ 2475.818613][  T473] fdee0000.hdmirx-controller: stream stopping finished
```

9.5 HDMI IN数据流打印

```
// 查看数据流打印需要将log等级开到3，每隔半帧打印一句log，打印如下：
[ 2778.207227][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.215891][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.223889][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.232559][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.240557][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.249222][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.257226][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.265887][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.273887][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.282557][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
```

9.6 信号未锁定异常日志

```
// 信号未锁定
[  285.949990][  T191] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing signal not lock, tmds_clk_ratio:0
[  285.950011][  T191] fdee0000.hdmirx-controller:
hdmirx_wait_lock_and_get_timing mu_st:0x0, scdc_st:0x0, dma_st10:0x10
...
// 读取到错误的分辨率
[  257.222739][  T193] fdee0000.hdmirx-controller: get timings from dma
[  257.222744][  T193] fdee0000.hdmirx-controller: act:39024x0, total:17732x1,
fps:8375, pixclk:148500000
[  257.222749][  T193] fdee0000.hdmirx-controller: hfp:4294904560, hs:184,
hbp:41260, vfp:1, vs:0, vbp:0
[  257.222753][  T193] fdee0000.hdmirx-controller: tmds_clk:148500000
[  257.222758][  T193] fdee0000.hdmirx-controller: interlace:0, fmt:0, vic:127,
color:24, mode:dvi
// 连续读取到错误分辨率
[  257.254994][  T193] fdee0000.hdmirx-controller: hdmirx_try_to_get_timings: res
not stable!
```

9.7 信号锁定后失锁

```
// 驱动开流
[140.628993][ T1525] fdee0000.hdmirx-controller: hdmirx_start_streaming:
start_stream cur_buf y_addr:0x10b00000, uv_addr:0x10b00000
[140.629009][ T1525] fdee0000.hdmirx-controller: hdmirx_start_streaming: enable
dma
[140.632067][  T198] fdee0000.hdmirx-controller: line_flag_int_handler: last have
no dma_idle_irq
[141.223426][  T198] fdee0000.hdmirx-controller: rcv frames
// 预览3s后检测到信号变化
[144.955132][    C4] fdee0000.hdmirx-controller: mu0_st:0xc0000000
[144.955142][    C4] fdee0000.hdmirx-controller: mu2_st:0x2
[144.955173][    C4] fdee0000.hdmirx-controller: mainunit_2_int_handler:
TMDSVALID_STABLE_CHG
[144.955179][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler: en_fiq
[144.958942][ T1525] fdee0000.hdmirx-controller: stream start stopping
...
// 锁定后报TMDSVALID_STABLE_CHG、TMDSQPCLK_LOCKED_CHG、TMDSQPCLK_OFF_CHG、AVP1中断
[  176.865529][   T55] fdee0000.hdmirx-controller: hdmirx_format_change: New
format: 1920x1200p59.99 (2592x1242)
[  176.873290][   T55] fdee0000.hdmirx-controller: hdmirx_format_change: queue
res_chg_event
[  176.873311][   T55] fdee0000.hdmirx-controller: hdmirx_set_ddr_store_fmt:
pix_fmt: RGB888, DMA_CONFIG1:0x12000001
[  176.873317][   T55] fdee0000.hdmirx-controller: hdmirx_interrupts_setup:
enable
[  176.873355][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler: hdmi
irq not handled!
[  176.873358][    C4] fdee0000.hdmirx-controller: avp0:0x0, avp1:0x0, mu0:0x0,
mu2:0x0, pk0:0x800, pk2:0x0, scdc:0x0
```

```
[  177.389362][    C4] fdee0000.hdmirx-controller: avpunit_0_int_handler:
avp0_st:0x200000
[  177.389378][    C4] fdee0000.hdmirx-controller: mu2_st:0x2
[  177.389397][    C4] fdee0000.hdmirx-controller: mainunit_2_int_handler:
TMDSVALID_STABLE_CHG
[  177.389403][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  177.390164][    C4] fdee0000.hdmirx-controller: mu0_st:0x10
[  177.390185][    C4] fdee0000.hdmirx-controller: mainunit_0_int_handler:
TMDSQPCLK_LOCKED_CHG
[  177.390189][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  177.392165][    C4] fdee0000.hdmirx-controller: mu0_st:0x10
[  177.392185][    C4] fdee0000.hdmirx-controller: mainunit_0_int_handler:
TMDSQPCLK_OFF_CHG
[  177.392190][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  177.393391][    C4] fdee0000.hdmirx-controller: mu2_st:0x2
[  177.393410][    C4] fdee0000.hdmirx-controller: mainunit_2_int_handler:
TMDSVALID_STABLE_CHG
[  177.393415][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  177.397433][    C4] fdee0000.hdmirx-controller: avpunit_1_int_handler:
avp1_st:0x40000000
[  177.397445][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
[  177.397994][    C4] fdee0000.hdmirx-controller: avpunit_1_int_handler:
avp1_st:0x80000000
[  177.398002][    C4] fdee0000.hdmirx-controller: hdmirx_hdmi_irq_handler:
en_fiq
```

9.8 TMDSCLK统计错误

```
// 源端实际输入4k@60 实际tmdsclk为594M
[147.366197][  T192] fdee0000.hdmirx-controller: get timings from ctrl
[147.366201][  T192] fdee0000.hdmirx-controller: act:3840x2160, total:4400x2250,
fps:16, pixclk:153864000
[147.366206][  T192] fdee0000.hdmirx-controller: hfp:176, hs:88, hbp:296, vfp:8,
vs:10, vbp:72
[147.366210][  T192] fdee0000.hdmirx-controller: tmds_clk:153864000,
pix_clk:153864000
[147.366214][  T192] fdee0000.hdmirx-controller: interlace:0, fmt:0, vic:0,
color:24, mode:hdmi
...
// 由于tmdsclk统计错误导致音频采样率错误
[152.555134][  T187] rk_hdmirx fdee0000.hdmirx-controller:
hdmirx_delayed_work_audio: audio underflow and overflow 0x3000000, with fs
invalid 12400
[152.555191][  T187] rk_hdmirx fdee0000.hdmirx-controller: hdmirx_audio_fifo_init
[152.758428][  T192] rk_hdmirx fdee0000.hdmirx-controller:
hdmirx_delayed_work_audio: audio underflow and overflow 0x3000000, with fs
invalid 12400
[152.758458][  T192] rk_hdmirx fdee0000.hdmirx-controller: hdmirx_audio_fifo_init
```

```
[152.965070][  T192] rk_hdmirx fdee0000.hdmirx-controller:
hdmirx_delayed_work_audio: audio underflow and overflow 0x3000000, with fs
invalid 12400
[152.965093][  T192] rk_hdmirx fdee0000.hdmirx-controller: hdmirx_audio_fifo_init
```

9.9 热拔插或切分辨率重启

拔出后空指针报错重启

```
[  453.482730][  T819] fdee0000.hdmirx-controller: hdmirx_set_fmt: req(3840,
2160), out(3840, 2160), fmt:0x33524742
[  453.482745][  T819] fdee0000.hdmirx-controller: hdmirx_get_content_type:
Graphics
[  453.482752][  T819] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
RGB888
[  453.987489][  T819] fdee0000.hdmirx-controller: C-Plane 0 size: 24883200,
Total imagesize: 24883200
[  453.987524][  T819] fdee0000.hdmirx-controller: hdmirx_set_fmt: req(3840,
2160), out(3840, 2160), fmt:0x33524742
[  453.987539][  T819] fdee0000.hdmirx-controller: hdmirx_get_content_type:
Graphics
[  453.987545][  T819] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
RGB888
[  454.111292][    C3] hrtimer: interrupt took 15194002 ns
[  454.144120][ T3686] Unable to handle kernel paging request at virtual address
ffffffc0c684007d
DDR 73dffea49e typ 23/09/20-14:18:32,fwver: v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB

[ 4088.362084][T31719] Unable to handle kernel paging request at virtual address
0000000200000077
[ 4088.364060][  T268] dw-hdcp fde70000.hdcp: hdcp reset
[ 4088.366597][T31719] Mem abort info:
[ 4088.366599][T31719]   ESR = 0x96000005
[ 4088.366601][T31719]   EC = 0x25: DABT (current EL), IL = 32 bits
[ 4088.366602][T31719]   SET = 0, FnV = 0
[ 4088.366604][T31719]   EA = 0, S1PTW = 0
[ 4088.366604][T31719] Data abort info:
[ 4088.366606][T31719]   ISV = 0, ISS = 0x00000005
[ 4088.366607][T31719]   CM = 0, WnR = 0
[ 4088.366609][T31719] user pgtable: 4k pages, 39-bit VAs, pgdp=000000010318c000
[ 4088.366614][T31719] [0000000200000077] pgd=0000000000000000
[ 4088.367094][  T580] Unable to handle kernel write to read-only memory at
virtual address ffffffc008105928
[ 4088.367373][T31719] , p4d=0000000000000000
[ 4088.367375][    C1] ------------[ cut here ]------------
[ 4088.367386][    C1] WARNING: CPU: 1 PID: 0 at kernel/workqueue.c:1660
queue_delayed_work_on+0xb4/0x114
```

拔出后报**hdmirx**相关中断后重启

```
[  417.412262][  T827] fdee0000.hdmirx-controller: C-Plane 0 size: 24883200,
Total imagesize: 24883200
[  417.412293][  T827] fdee0000.hdmirx-controller: hdmirx_set_fmt: req(3840,
2160), out(3840, 2160), fmt:0x33524742
[  417.412308][  T827] fdee0000.hdmirx-controller: hdmirx_get_content_type:
Graphics
[  417.412314][  T827] fdee0000.hdmirx-controller: hdmirx_get_pix_fmt: pix_fmt:
RGB888
[  417.666427][    C4] fdee0000.hdmirx-controller: mu0_st:0xc0000000
[  417.666443][    C4] fdee0000.hdmirx-controller: hdmirx_interrupts_setup:
disable
[  417.666468][    C4] fdee0000.hdmirx-controller: mainunit_2_int_handler:
TMDSVALID_STABLE_CHG
[  417.666581][ T3437] ------------[ cut here ]------------
DDR 73dffea49e typ 23/09/20-14:18:32,fwver: v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
```

插入后开流重启

```
[   31.300604][  T556] fdee0000.hdmirx-controller: hdmirx_get_color_range:
color_range: limit
[   31.300662][  T556] fdee0000.hdmirx-controller: hdmirx_get_color_space: video
standard: xvYCC601
[   31.316905][  T556] fdee0000.hdmirx-controller: hdmirx_start_streaming:
start_stream cur_buf y_addr:0x10900000, uv_addr:0x10900000
[   31.316924][  T556] fdee0000.hdmirx-controller: hdmirx_start_streaming: enable
dma
[  417.666581][ T3437] ------------[ cut here ]------------
DDR 73dffea49e typ 23/09/20-14:18:32,fwver: v1.14
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
```

插入后画面黑（绿）屏，且拔插或更换信号源都无法出图

测试部分信号源存在HDMIRX异常，但未导致总线异常的情况。此时DMA功能异常无法搬运数据导致画面黑屏，可通过log等级开到3，判断DMA相关中断是否正常交替产生，若缺少dma_idle_irq中断则也是同类问题。

正常log：

```
[ 2778.207227][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.215891][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.223889][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.232559][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.240557][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x80, st13:1080
[ 2778.249222][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
```

异常log：

```
[ 2778.215891][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.232559][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.249222][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.265887][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
[ 2778.282557][  T205] fdee0000.hdmirx-controller: dma_irq st1:0x100, st13:540
```

10. 重启问题相关补丁说明

针对HDMIRX重启问题，更新SDK可降低一定概率。已提交代码版本如下：

SDK version	kernel
android-12.0-mid-rkr14	5.10
android-13.0-mid-rkr6	5.10
android-14.0-mid-rkr1	6.1
linux-5.10-gen-rkr5	5.10

注：NVR-sdk-v1.7 未包含最新重启问题补丁需要单独提供。

若未更新到以上版本，则需要另外打上补丁，参考: https://redmine.rock-chips.com/documents/130

kernel-5.10

- 方法一 直接保存自己的修改，git am打上对应的补丁，再手动加上自己的修改。
- 方法二 替换../hdmirx/路径下的所有驱动源码，再打上0009、0010、0022补丁，再加上你们自己的修改。

rkbin

需要rk3588_bl31_v1.41.elf 的版本至少保持在v1.39 之后，即（rk3588_bl31_v1.39.elf）。

版本确认参考：vim rkbin/RKTRUST/RK3588TRUST.ini

通常默认更新到v1.42版本，可解决其他ddr相关问题，参考补丁简报：https://redmine.rock-chips.com/issues/441328

更新bl31之后对应的ddr_bin版本也需要配套更新。

更新bl31烧录uboot后可通过开机log确认是否替换成功，例如**v1.42**对应的编译时间为：

```
INFO:    Preloader serial: 2
NOTICE:  BL31: v2.3():v2.3-639-g87bcc5dfe:derrick.huang
NOTICE:  BL31: Built : 18:06:16, Sep  9 2023
```

打上补丁之后需要检查几个重要函数的代码是否一致：

中断配置：

```
static void hdmirx_interrupts_setup(struct rk_hdmirx_dev *hdmirx_dev, bool en)
{
    struct v4l2_bt_timings *bt = &hdmirx_dev->timings.bt;
```

```
    v4l2_dbg(1, debug, &hdmirx_dev->v4l2_dev, "%s: %sable\n",
            __func__, en ? "en" : "dis");

    if (en && bt->pixelclock > 590000000 && hdmirx_dev->pix_fmt != HDMIRX_YUV420)
{
        hdmirx_update_bits(hdmirx_dev, VMON_CONTROL,
                   VMON_IRQ_THR_MASK, VMON_IRQ_THR_MASK);
        hdmirx_update_bits(hdmirx_dev, VMON_CONTROL2,
                   VMON_IRQ_VERTICAL_MASK |
                   VMON_IRQ_HORIZONAL_MASK,
                   VMON_IRQ_VERTICAL_SEL(0x1f) |
                   VMON_IRQ_HORIZONAL_SEL(0x1f));
        hdmirx_update_bits(hdmirx_dev, DEFRAMER_CONFIG0,
                   VS_REMAPFILTER_EN_QST |
                   VS_FILTER_ORDER_QST_MASK |
                   HS_FILTER_ORDER_QST_MASK,
                   VS_REMAPFILTER_EN_QST |
                   VS_FILTER_ORDER_QST(0x3) |
                   HS_FILTER_ORDER_QST(0x0));
    } else {
        hdmirx_update_bits(hdmirx_dev, VMON_CONTROL,
                   VMON_IRQ_THR_MASK, 0);
        hdmirx_update_bits(hdmirx_dev, VMON_CONTROL2,
                   VMON_IRQ_VERTICAL_MASK |
                   VMON_IRQ_HORIZONAL_MASK,
                   VMON_IRQ_VERTICAL_SEL(0x1f) |
                   VMON_IRQ_HORIZONAL_SEL(0x0));
        hdmirx_update_bits(hdmirx_dev, DEFRAMER_CONFIG0,
                   HS_FILTER_ORDER_QST_MASK,
                   HS_FILTER_ORDER_QST(0x3));
    }
```

开流：

```
static int hdmirx_start_streaming(struct vb2_queue *queue, unsigned int count)
{
    ...
    touch_flag = (hdmirx_dev->bound_cpu << 1) | 0x1;
    sip_hdmirx_config(HDMIRX_AUTO_TOUCH_EN, 0, touch_flag, 100);
    ...
    sip_hdmirx_config(HDMIRX_INFO_NOTIFY, 0, DMA_CONFIG6, HDMIRX_DMA_EN);
    ...
}
```

特殊修改

部分客户测试在某些场景，如经过switch，或者经过转接芯片（DP2HDMI）的通路重启概率会增大，需要转接芯片原厂介入分析。目前已知龙讯LT8641（switch）和LT8711 (DP2HDMI)关于RK3588热拔插场景有优化，需要咨询原厂提供对应的固件进行测试。