

Rockchip HDMI RX开发指南

文件标识: RK-KF-YF-321

发布版本: V1.1.0

日期: 2022-08-09

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有 © 2022 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

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

目录

Rockchip HDMI RX开发指南

- HDMI IN功能概述
 - HDMI RX模块特性简介
 - HDMI IN功能框图
- HDMI RX驱动代码和dts配置
 - SDK版本要求
 - 驱动代码和Kernel配置
 - dts配置说明
 - HDMI RX控制器配置
 - 预留内存
 - Audio配置
 - HDCP配置
 - CEC配置
 - EDID配置方法
- HDMI IN Video架构
 - HDMI IN Video工作流程
 - HDMI RX主要驱动架构
 - 图像Buffer轮转机制
 - 图像传输延时
- HDMI IN HDCP功能
 - HDCP1.4
 - dts 配置
 - Key 烧写
 - HDCP1.4 状态查看
 - HDCP2.3
 - dts 配置
 - 打包 firmware 和 启动服务
 - HDCP2.3 状态查看
- HDMI IN CEC功能
- HDMI IN APK适配方法
 - APK源码路径
 - HdmiIn预览APK说明
 - TIF与Camera预览方式差异
- 驱动调试方法
 - 调试工具获取

调试命令举例

查看所有video节点
查找rk_hdmirx设备
获取驱动timings信息
实时查询timings信息
查询分辨率和图像格式
开启图像数据流
抓取图像文件
正常取流log

常见问题调试方法

打开log开关
通过io命令读写寄存器
HDMI RX状态查询
HDMI IN信号不锁定问题

典型日志说明

拔插日志
切换分辨率日志
信号未锁定异常日志

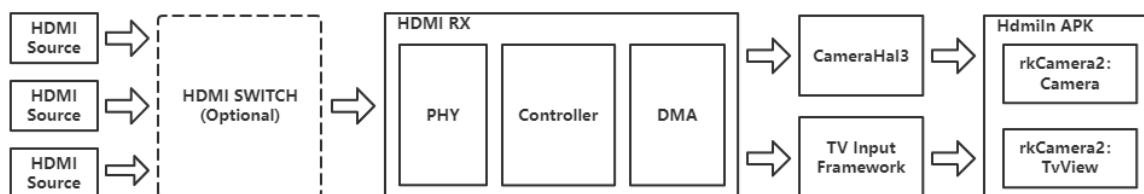
HDMI IN功能概述

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/DPDIF 2channel output
- I2S 2/4/6/8channel output

注：RK3588S不含HDMI RX模块。

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。

HDMI RX驱动代码和dts配置

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]
```

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
```

dtb配置说明

参考SDK中RK3588 EVB1的dtb配置：

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

HDMI RX控制器配置

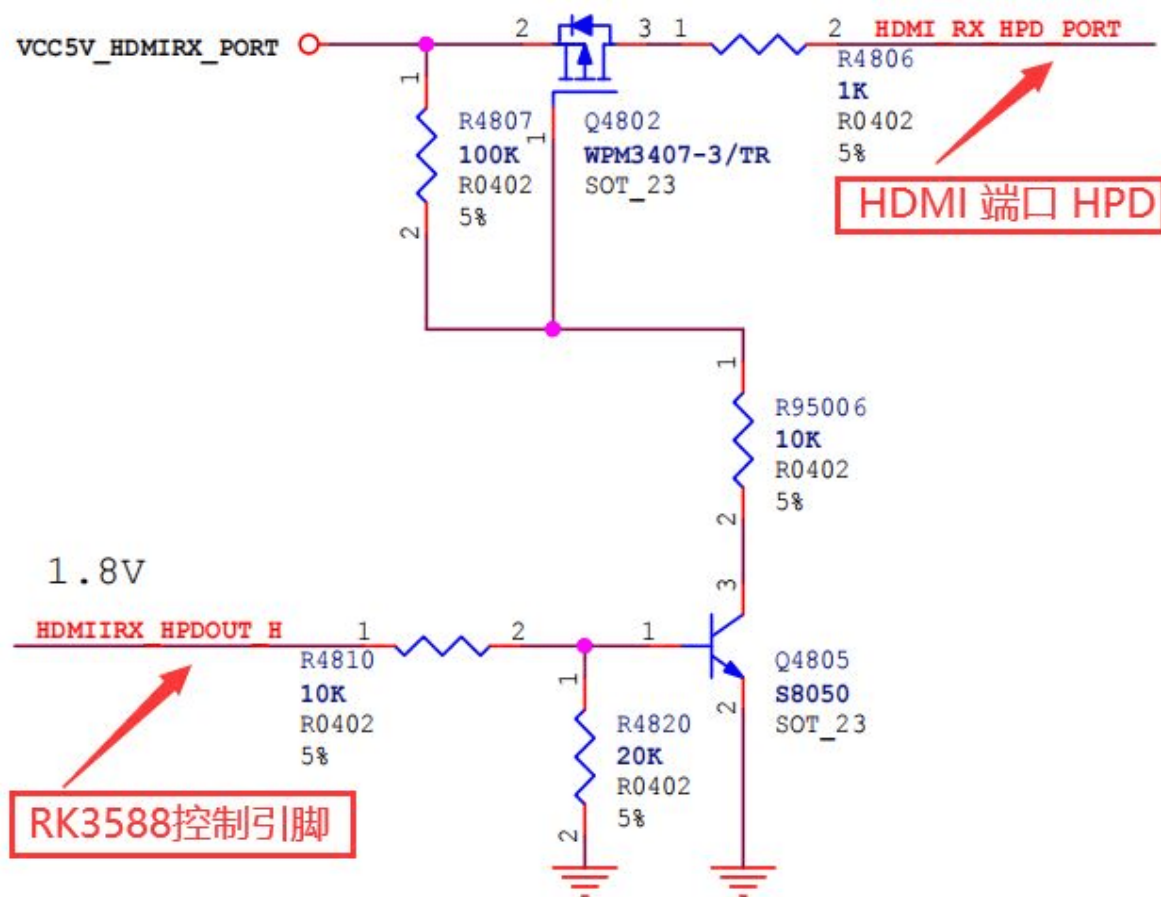
```
/* Should work with at least 128MB cma reserved above. */
&hdmirx_ctrler {
    status = "okay";

    /* 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>;
};

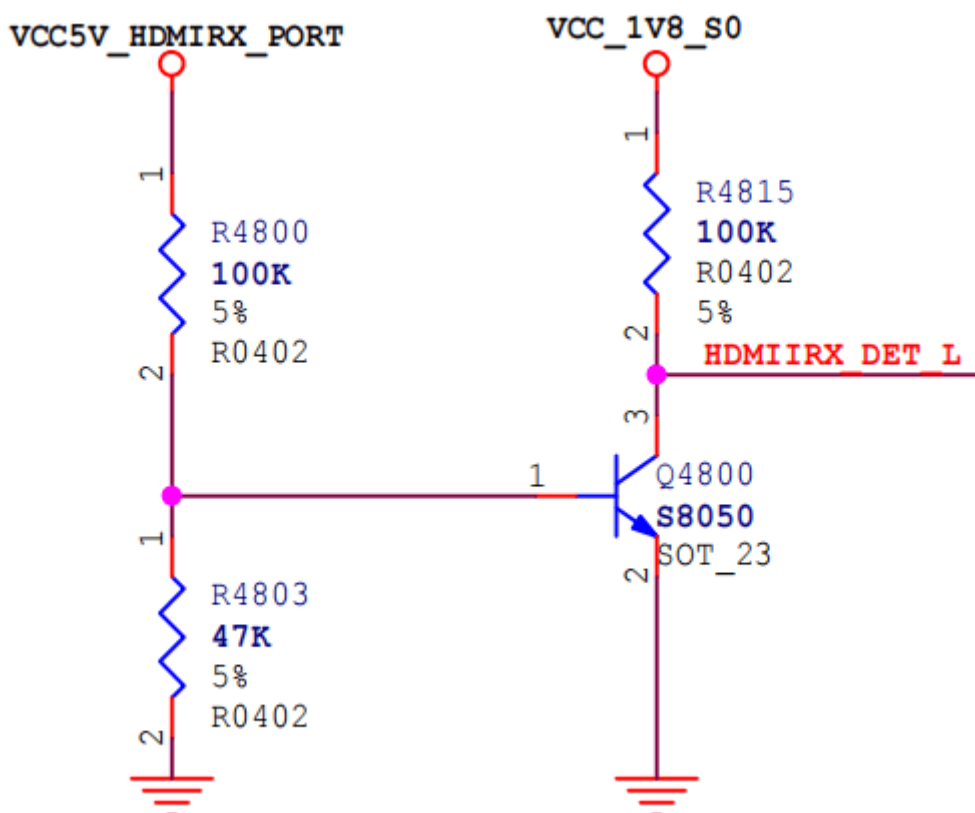
hdmirx {
    hdmirx_det: hdmirx-det {
        rockchip,pins = <2 RK_PB5 RK_FUNC_GPIO &pcfg_pull_up>;
    };
};
```

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。



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：

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;
        reg = <0x0 (256 * 0x100000) 0x0 (128 * 0x100000)>;
        linux,cma-default;
    };
};

```

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

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;
};
```

- 单独支持 HDCP2.3:

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

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

- 同时支持 HDCP1.4 和 HDCP2.3，开机默认打开 HDCP2.3 功能:

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

&hdmirx_ctrler {
    status = "okay";
    hdcp1x-enable;
    hdcp2x-enable;
};
```

- 同时支持 HDCP1.4 和 HDCP2.3, 开机默认打开 HDCP1.4 功能:

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

&hdmirx_ctrler {
    status = "okay";
    hdcplx-default-enable;
    hdcpl2x-enable;
};
```

CEC配置

CEC 功能内核驱动默认开启, 不需要单独配置。

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:

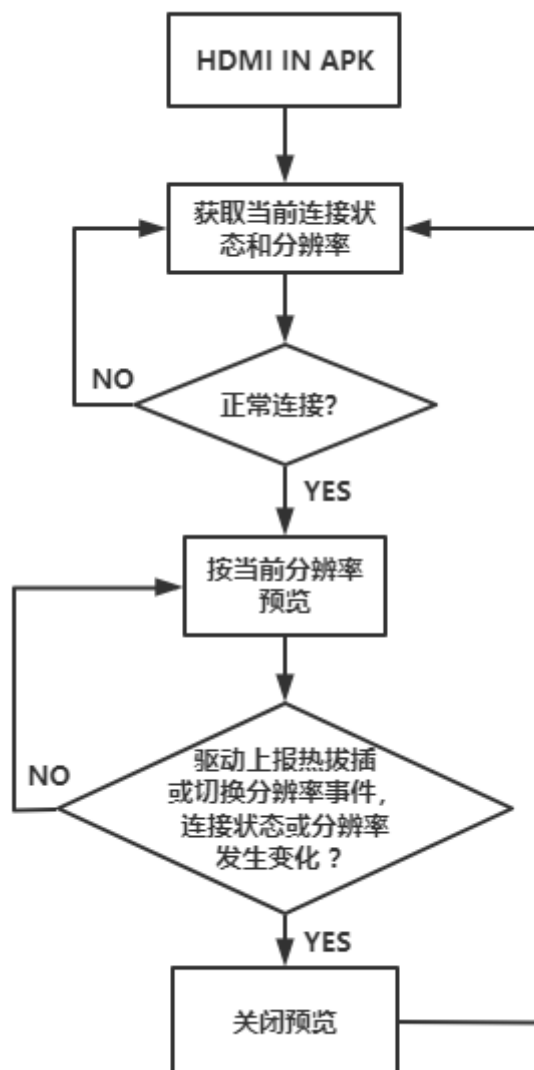

```
static const struct v4l2_ioctl_ops hdmirx_v4l2_ioctl_ops = {
    ...
    .vidioc_g_edid = hdmirx_get_edid,
    .vidioc_s_edid = hdmirx_set_edid,
    ...
};
```

可通过工具编辑配置EDID，再替换到驱动代码中，推荐EDID可视化编辑工具如下：

https://www.quantumdata.com/support/downloads/980/release_5_05/R_980mgr_5.05_Win32.msi

HDMI IN Video架构

HDMI IN Video工作流程



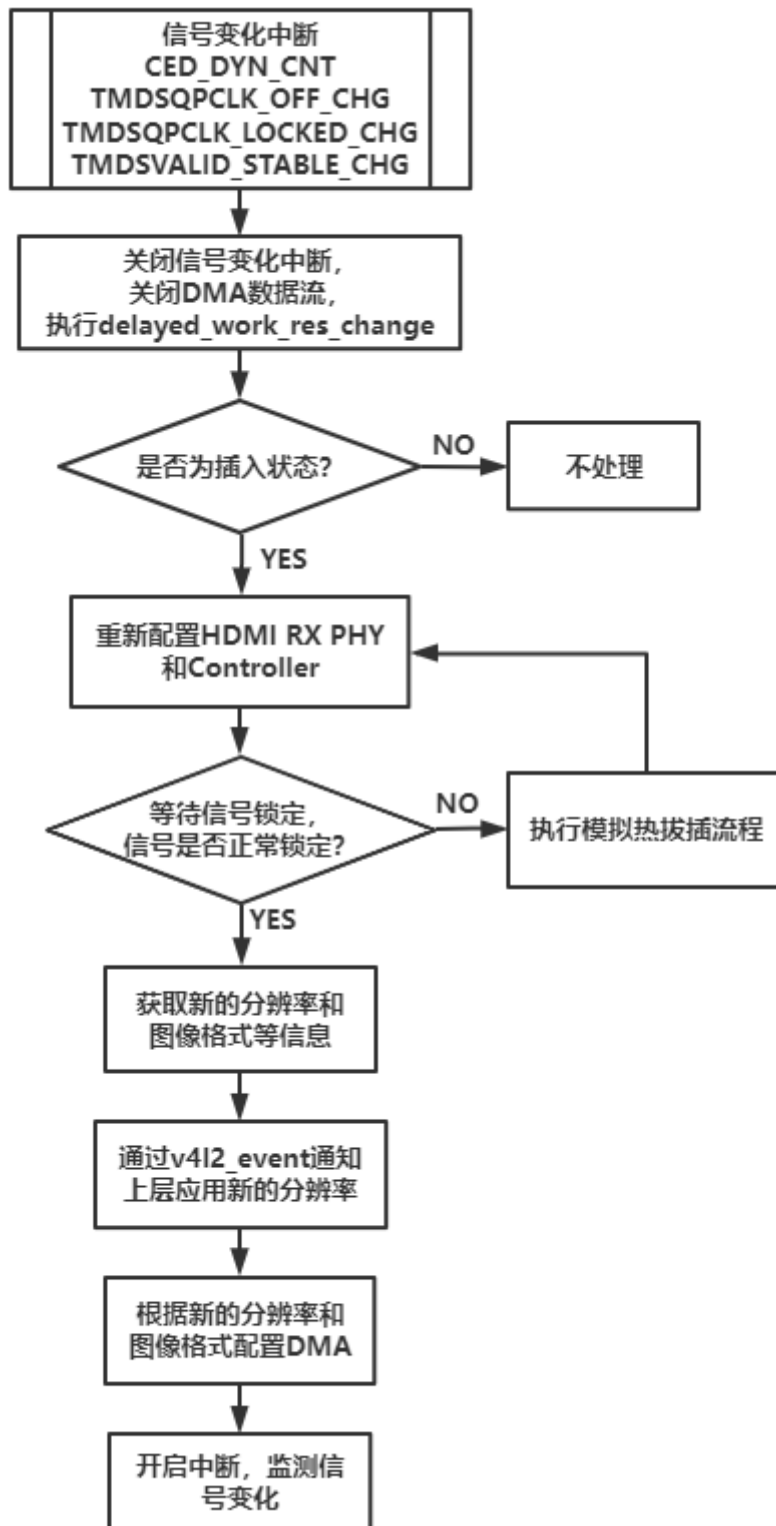
HDMI RX主要驱动架构

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

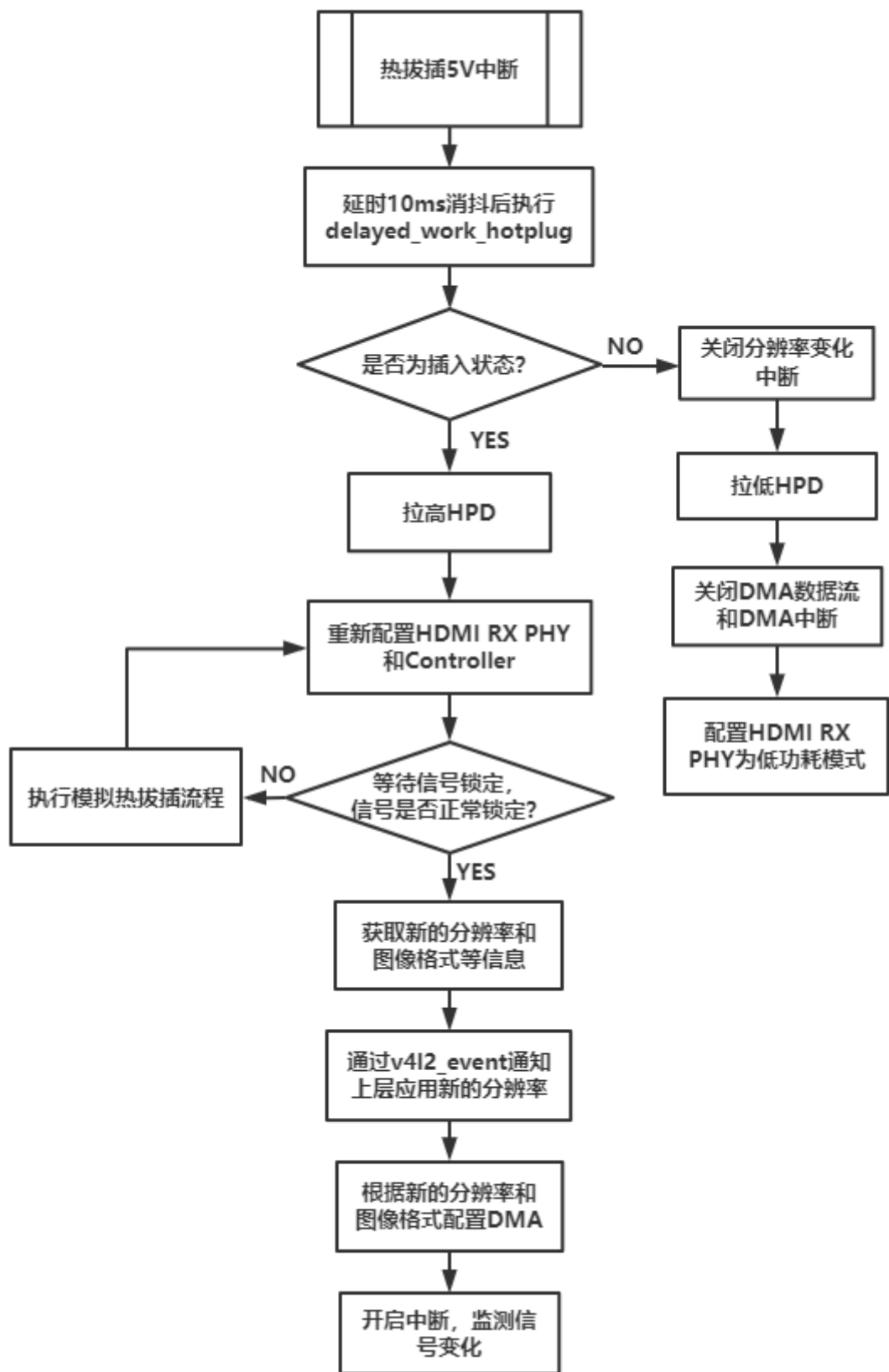
- hdmirx_5v_det_irq_handler: 5V检测中断，由gpio引脚HDMIIRX_DET_L触发，用于检测HDMI接口的拔插动作。
- hdmirx_hdmi_irq_handler: HDMI RX控制器中断，主要用于初始化配置，以及监测HDMI信号变化时使用。

- 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。

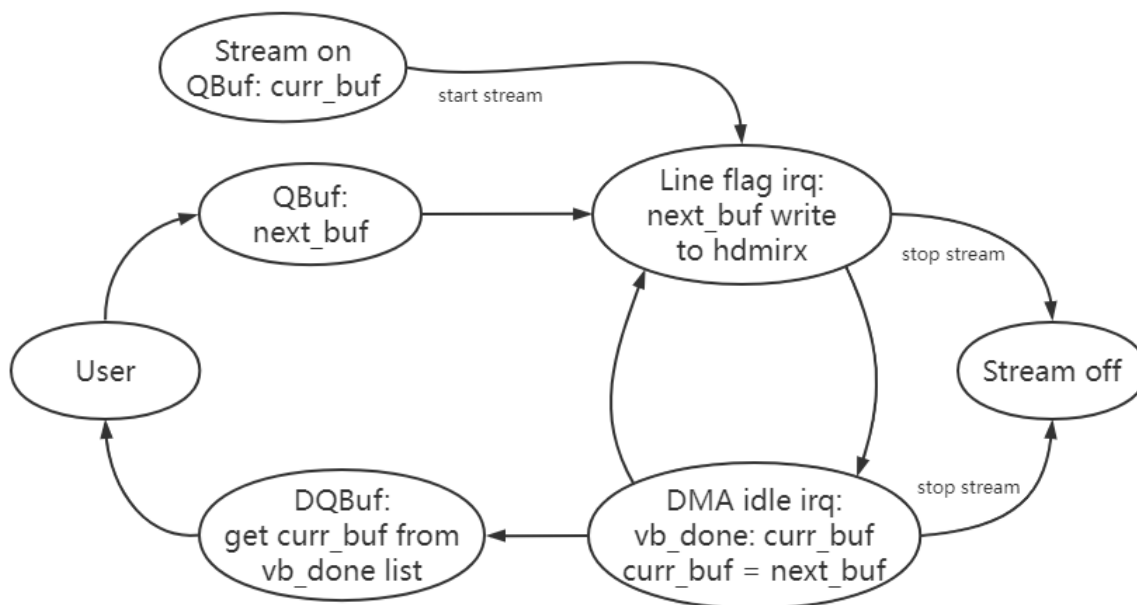
分辨率切换流程如下:



热拔插流程如下:



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。

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。

HDMI IN HDCP功能

HDCP1.4

dts 配置

参考[HDCP配置](#)章节，配置为hdcp1x-enable;

Key 烧写

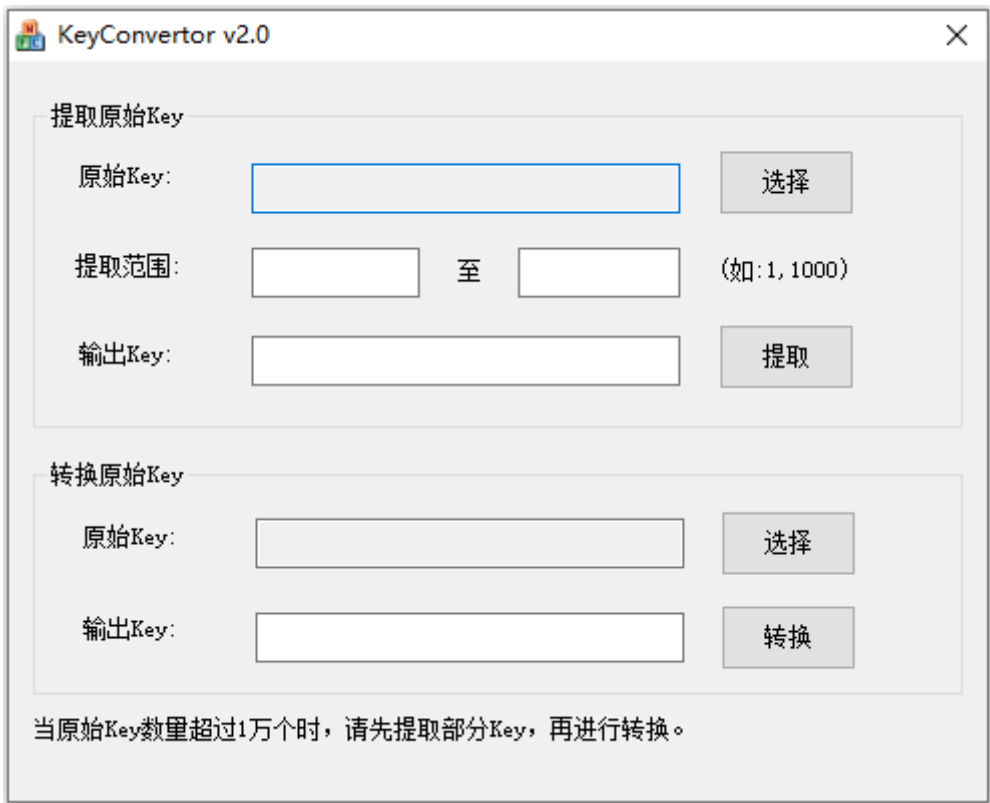
Key 拆分和转换工具从SDK获取：

RKTools/windows/HDCP Key工具Vx.x.7z

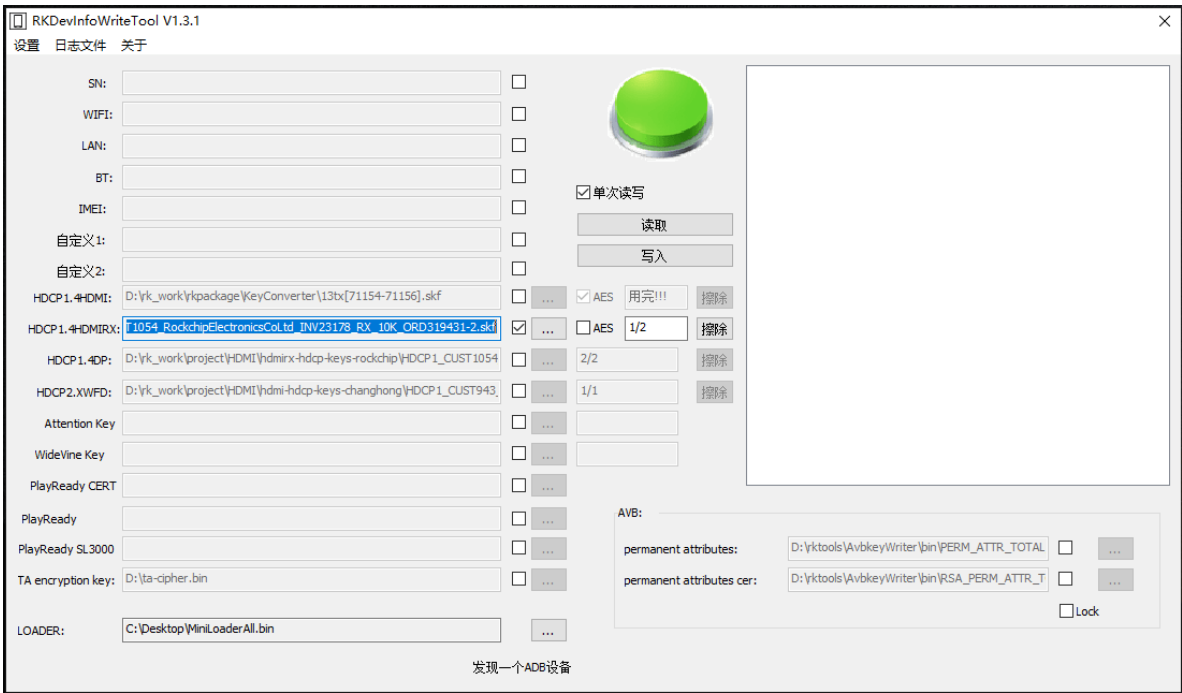
Key 烧写工具从SDK获取：

RKTools/windows/RKDevInfowriteTool-x.x.x.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。



- 机器进入Loader模式，烧写工具勾选HDCP1.4HDMIRX，选择上面的 .skf文件，点击写入即可完成 Key 的烧写。说明：写完一个 Key 之后会自动会跳到下一个 KEY。



- HDCP1.4 Key 结构说明。

一个 HDCP 1.4 有 308 Byte，分别如下：
8 Byte，前 5 个 Byte 是 KSV，后面 3 个 Byte 为 0x0；
280 Byte DPK；
20 Byte SHA。

HDCP1.4 状态查看

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

```
hdcp disable: hdcp1.4 没使能
hdcp_auth_start: hdcp1.4 认证过程中
hdcp_auth_success: hdcp1.4 认证成功
hdcp_auth_fail: hdcp1.4 认证失败
unknown status: 未知状态
```

HDCP2.3

当前 HDCP2.3 有些修改还没推送，而且 Firmware 打包工具和 hdcp2_tx_rx bin 文件需要单独提供，所以需要先获取到 HDCP2.3 的补丁包“HDCP2.3-RX-PATCH.zip”。

dts 配置

参考[HDCP配置](#)章节，配置为hdcp2x-enable;

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。

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

- 修改 PKF 和 DUK

```
--- a/firmware/firmware.aic.nouart
+++ b/firmware/firmware.aic.nouart
@@ -1,5 +1,5 @@
-PKf      = 0x00000000000000000000000000000000
-DUK      = 0x00000000000000000000000000000000
+PKf      = 0x00112233445566778899aabbccddeeff
+DUK      = 0xffeeddccbbaa99887766554433221100
IK        = 0xffeeddccbbaa99887766554433221100
IVC       = 0xdeadbeeffedcba9876543210
BBRcode   = 6144
```

- 打包 firmware.le，包含了所有的Production Key

```
DWC_HDCP2_BASE_ESM_Firmware$
./build_prod_image.sh HDCP_RX_2TX_HDMI
```

打包到 DWC_HDCP2_BASE_ESM_Firmware/firmware/firmware.le，然后把 firmware.le 重命名成 hdcp2_hdmi.fw

- 生成单独的RX Key

```
DWC_HDCP2_BASE_ESM_Firmware$
./build_rxkeys.sh HDCP_RX_2TX_HDMI NOUART BA

ls tools/rxkeys/hdcpkeys/
fw_hdcp_receivers_0737C22DEC.bin  fw_hdcp_receivers_4D64BA99B4.bin
fw_hdcp_receivers_62CCAA07DB.bin  fw_hdcp_receivers_AA113C1AFE.bin
fw_hdcp_receivers_CD23A407F3.bin  fw_hdcp_receivers_9D3355F824.bin
fw_hdcp_receivers_32DC5BF80C.bin  fw_hdcp_receivers_55EEC3E501.bin
fw_hdcp_receivers_B29B45664B.bin  fw_hdcp_receivers_F8C83DD213.bin  fw_lc128.bin
```

生成的 RX Key 在 DWC_HDCP2_BASE_ESM_Firmware/tools/rxkeys/hdcpkeys/ 目录下。

默认情况下会把 hdcp_receivers.bin 的所有 Key 都拆分出来，如果只需要拆分其中一部分，可以如下修改：

```
可以修改build_rxkeys.sh，指定需要从第几个 key 开始拆分，拆分几个 key
./troot_config_hdcpkey_images -c $CERT -l $LIC -t ./hdcp_transmitter.temp.bin -r
./hdcp_receivers.temp.bin -o ./DWC_RXKEY_DIR -k ./firmware.aic -a ./aictool -b
3 -n 5 -x 12345678901234567890123456789012
```

-b: 从第几个key开始， -n: 生成n个key，0表示生成所有key，如上的修改是从第3个 key 开始，拆分出5个 key，所以 next-index 保存的应该是8。
在tools/rxkeys/next-index.txt里面会记录下个key的index，下一次-b从next-index开始。

- RX Key 的烧写

- 1、把 hdcp2_hdmi.fw 放到 vendor/firmware/ 目录下，可以参考 HDCP2.3 补丁包，编译到固件中。
- 2、把上述生成的 fw_hdcp_receivers_*.bin 拷贝到 windows 目录下，然后 win+r，命令行执行工具目录下的 HdcKeyPack.exe 把单独的 Key 打包到同一个 .skf 文件中，方便于量产工具 RKDevInfoTool.exe 烧写，如：

```
D:\rktools\002-windows\003-hdcp\RKDevInfoTool\bin>HdcKeyPack.exe --pack --in
.\hdcpkeys --out .\hdcp.skf
Current dir is D:\rktools\002-windows\003-hdcp\RKDevInfoTool\bin\
output filename is .\hdcp.skf
dest file exist, will be overwrite
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_0737C22DEC.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_32DC5BF80C.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_4D64BA99B4.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_55EEC3E501.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_62CCAA07DB.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_9D3355F824.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_AA113C1AFE.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_B29B45664B.bin size is 1000
file D:\rktools\002-windows\003-
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_CD23A407F3.bin size is 1000
file D:\rktools\002-windows\003-
```

```
hdcp\RKDevInfoTool\bin\.\hdcpkeys\fw_hdcp_receivers_F8C83DD213.bin size is 1000
10 file packed to .\hdcp.skf[CRC: 0xc75F3091]
D:\rktools\002-windows\003-hdcp\RKDevInfoTool\bin>
```

3、用 RKDevInfoTool.exe 工具，选择 HDCP2X_HDMIRX 导入上面生成的 hdcp.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。



- 执行 hdcp2_rx_tx 服务

hdcp2_tx_rx 是 RK 提供的二进制执行文件，把 hdcp2_tx_rx 放到 vendor/bin/ 目录下，执行 ./vendor/bin/hdcp2_tx_rx 1

这边需要带参数 1 (HDMITX 和 HDMIRX)。

目前可以参考提供的补丁，做成开机加载的服务，不需要手动执行。

HDCP2.3 状态查看

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

```
HDCP2:
Decrypted / No decrypted: 解密 / 没解密
Capable: hdcp2.3 使能
Not capable: hdcp2.3 没使能
Authenticated success: 认证成功
Authenticated failed: 认证失败
```

HDMI IN CEC功能

device/rockchip/common 如下修改：

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

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>

```

HDMI IN APK适配方法

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。
- 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
```

HdmiIn预览APK说明

- MainActivity主界面，TIF预览方式，支持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。

```
String INPUT_ID =
"com.example.partnersupportsampletvinput/.SampleTvInputService/HW0";
Uri channelUri = TvContract.buildChannelUriForPassthroughInput(INPUT_ID);
tvView.tune(INPUT_ID, channelUri);
```

在MainActivity的预览界面点击任一位置，会出现EDID的切换UI，显示当前所设置的EDID组。点击EDID按钮，会切回到另一组EDID，APK对以下节点进行写值，HDMI RX驱动会实现EDID分组切换：

```
sys/class/hdmi-rx/hdmi-rx/edid
```

为实现掉电记忆EDID分组功能，APK在选择EDID分组时，会同步将节点值存储到以下属性。在下次机器开机时，SystemService系统服务会根据该属性值，修改edid节点：

```
persist.sys.hdmi-rx.edid
```

其中340M对应的属性和节点值为1，600M对应的属性和节点值为2，未配置时默认是340M。EDID详细配置说明请参考章节[EDID配置方法](#)。



- 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。

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screencap命令截图。MainActivity.java中也提供了另一种截图方式，见当中的startScreenShot函数。也可以使用上述第2点提到的谷歌标准MediaProjectionManager虚拟屏截图。

驱动调试方法

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/
```

调试命令举例

一般在调试分析问题，建议配置开启debug log，参考章节[打开log开关](#)。

查看所有video节点

```
ls /dev/video*
```

查找rk_hdmirx设备

使用v4l2-ctl -d参数指定video节点，-D命令查看节点信息，通过Driver name确认哪个是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
```

获取驱动timings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:
```

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:hdm
[16750.029633][ T2247] fdee0000.hdmirx-controller: deframer_st:0x11
[16750.029643][ T2247] fdee0000.hdmirx-controller: query_dv_timings:
3840x2160p60.00 (4400x2250)
```

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
  Colospace          : Unknown (1025fcdc)
  Transfer Function  : Unknown (00000020)
  YCbCr Encoding     : Unknown (000000ff)
  Quantization       : Default
  Plane 0            :
    Bytes per Line   : 3840
    Size Image        : 16588800
```

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
```

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
```

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
```

常见问题调试方法

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。然后通过dmesg命令打印kernel log：

```
echo 2 > /sys/module/rockchip_hdmirx/parameters/debug
dmesg
```

- 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。

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
```

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
...
-----
```

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 -1 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/hdmi/rk_hdmi.c
b/drivers/media/platform/rockchip/hdmi/rk_hdmi.c
index 8183485a6e4c..27b900e90501 100644
--- a/drivers/media/platform/rockchip/hdmi/rk_hdmi.c
+++ b/drivers/media/platform/rockchip/hdmi/rk_hdmi.c
@@ -2768,6 +2768,7 @@ static void hdmi_delayed_work_res_change(struct
work_struct *work)
    hdmi_submodule_init(hdmi_dev);
    hdmi_update_bits(hdmi_dev, SCDC_CONFIG, POWERPROVIDED,
                     POWERPROVIDED);
+    msleep(300);
    hdmi_hpd_ctrl(hdmi_dev, true);
    hdmi_phy_config(hdmi_dev);
    hdmi_audio_setup(hdmi_dev);
```

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](#)。

拔插日志

```
// 检测到拔出动作
[29830.165185][ T2655] fdee0000.hdmi-controller: hdmi_delayed_work_hotplug:
plugin:0
// 关闭中断
[29830.165203][ T2655] fdee0000.hdmi-controller: hdmi_interrupts_setup:
disable
// 拉低HPD
[29830.165211][ T2655] fdee0000.hdmi-controller: hdmi_hpd_ctrl: disable,
hpd_trigger_level:1
[29830.177109][ T598] fdee0000.hdmi-controller: hdmi_query_dv_timings port
has no link!
...
// 检测到插入动作
[29843.128833][ T2655] fdee0000.hdmi-controller: hdmi_delayed_work_hotplug:
plugin:1
// 拉高hpd
[29843.139653][ T2655] fdee0000.hdmi-controller: hdmi_hpd_ctrl: enable,
hpd_trigger_level:1
...
[29843.247796][ T2655] fdee0000.hdmi-controller: wait_reg_bit_status: i:0,
time: 10ms
[29843.286353][ T2655] fdee0000.hdmi-controller: wait_reg_bit_status: i:38,
time: 50ms
[29843.286373][ T2655] rk_hdmi fdee0000.hdmi-controller:
hdmi_audio_interrupts_setup: 1
// HDMI信号锁定
[29843.703996][ T2655] fdee0000.hdmi-controller:
hdmi_wait_lock_and_get_timing signal lock ok, i:54!
...
// 图像格式
[30098.978794][ T2655] fdee0000.hdmi-controller: hdmi_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:hdm
[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

```

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:hdm
```

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!
```