量产显示问题

快速排查指南

文档版本 F01
文档类型 FAQ 文档
发布日期 2018-12-05

文档履历

版本号	变更说明	作者	日期
V1.0	初始版本	刘丽	2018/12/05

目录

版权

商标声明



Allwinnertech、Allwinner、全志和其他全志商标均为珠海全志科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受全志公司合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，全志对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

本文档中所提供的数据在不同的测试环境下可能存在一定的偏差，因其使用而导致的设计、开发错误，或可能遭致的意外、疏忽、侵权及其造成的损失（包括但不限于直接的、间接的、后果性的或附带性的损失），全志不做任何明示或默示的声明或保证。请您谨慎决定是否使用。

本文档中提到的第三方技术，除全志明确表示已获得合法授权且转授权予您使用外，您需要自行向权利人获取合法授权，同样，如您需在全志交付的软件或产品上集成第三方技术（如 Sony、DTS、杜比、AVS、MPEGLA 等)时，也请您自行向权利人获取合法授权。全志不为您的未授权行为或前述集成行为支付任何许可费用或提供任何技术支持，也不承担任何保证责任。请您自行为其后续使用行为负责

前言

概述

- 本文档提供 DISP、HDMI、LCD、DE-Interlace、G2D、Android HWC 等显示基础模块常规问题快速排查方案
- 本文档提供的快速排查方案模型如下：
  常规问题场景 + 基础硬件排查 + 基础软件配置排查 + 软件快速调试排查
- 本文档仅用于帮助开发人员快速排查和解决初级问题问题，如果按照文档中的指导仍无法解决问题，请联系全志原厂 FAE 进一步协助解决，并提供说明是否按照文档中的排查方案进行过必要的排查，提供排查过程的必要记录和数据

读者对象

本文档主要适用于以下工程师:

- 不具备显示技术基础的 RD 研发人员
- 不具备显示技术基础的 FAE 工程人员

作者信息

类别	章节名称	作者
DISP		AW0962、AW0723
HDMI		AW1202
LCD		AW1221
DE-Interlace		AW0962
G2D		AW0962
Android HWC		AW0962、AW0723
displayd		AW1479
CVBS_IN		AW1221
CVBS_OUT		AW1221

修订记录

版本	修改人	时间	备注
V1.0	AW0962	2018.12.05	文档初始版本
V2.0	AW0962	2018.12.19	修改排版

1. 显示通路排查

1.1. 显示通路日志调试

全志平台 Android 方案的显示通路从上到下可以简单分成如下层次：Android 应用层（图形生产者，如多媒体中间件视频解码数据、应用通过 OpenGL 绘制数据）、SurfaceFlinger、 HWC 硬件抽象层、Soc DE 显示引擎、TCON 显示通道、显示输出接口、显示设备。

本章节主要介绍 Android 方案 SurfaceFlinger 层以下的显示通路的日志信息调试方法，从而帮助工程师快速分析和排查问题，在无法定位和解决问题的情况下，向全志原厂 FAE 和 RD 人员反馈准确、必要的的调试信息。

1.1.1. Surfaceflinger 层日志调试

1.1.1.1. 操作命令

adb shell dumpsys SurfaceFlinger

1.1.1.2. 信息解析

SurfaceFlinger 信息较多，需要拆分成三段来分析

- Visible layers 信息

```
Visible layers (count = 6)
+ Layer 0x7e8f141000 (com.android.systemui.ImageWallpaper#0)
  Region transparentRegion (this=0x7e8f141380, count=1)
    [  0,   0,   0,   0]
  Region visibleRegion (this=0x7e8f141010, count=1)
    [  0,   0, 800, 1280]
  Region surfaceDamageRegion (this=0x7e8f141088, count=1)
    [  0,   0,   0,   0]
    | layerStack=   0, z=    11000, pos=(0,0), size=( 800,1280), crop=(   0,   0,  -1,  -1), finalCrop=(   0,   0,  -1,  -1), isOpaque=1
      invalidate=0, dataspace=Default (0), pixelformat=RGBx_8888 alpha=1.000, flags=0x00000002, tr=[1.00, 0.00][0.00, 1.00]
      client=0x7e8f0c86c0
      format= 2, activeBuffer=[ 800x1280: 800,  2], queued-frames=0, mRefreshPending=0
          mTexName=5 mCurrentTexture=1
          mCurrentCrop=[0,0,0,0] mCurrentTransform=0
          mAbandoned=0
          - BufferQueue mMaxAcquiredBufferCount=1 mMaxDequeuedBufferCount=2
            mDequeueBufferCannotBlock=0 mAsyncMode=0
            default-size=[800x1280] default-format=2 transform-hint=00 frame-counter=8
          FIFO(0):
          Slots:
            >[01:0x7e8f040aa0] state=ACQUIRED 0x7e8f0b7ba0 frame=8 [ 800x1280: 800,  2]
             [00:0x7e8f03ef80] state=FREE     0x7e8f0b6010 frame=7 [1280x 800:1280,  2]
```

- **Visible layers (count = 6)**
  可见图层信息，count 代表 layer 的总数
- **Layer** 0x7f61d32000 (com.android.systemui.ImageWallpaper#0)
  图层指针及对应应用的包名
- **Region** transparentRegion
  可见区域范围，可以检查是否合理，若矩阵对应区域为空即此图层不会显示，可定位部分应用无法显示问题
- **layerStack/ z / pos / size / crop / finalCrop**

layerStack 表示在哪个显示器显示，Z 表示 z 序，越大越靠前，pos 表示左上角座标，size 表示宽高，crop 表示剪裁区域

- **Slots**

BufferQueue 使用情况，其中，state表示bufferqueue的状态，有free、dequeued、queued、acquireced，分别表示空闲、已从surfaceflinger 申请到buffer、 已把buffer 送到sf、sf或者GPU正在使用buffer；一般图层有2-3 个buffer，视频图层有16 个，如果buffer 都是free 状态，说明没有应用操作图层，该图层无法显示；如果没有一个是ACQUIRED 状态，说明sf 没有送显，sf 内部排查。

➢ Displays 信息

```
Displays (1 entries)
+ DisplayDevice: Built-in Screen
   type=0, hwcId=0, layerStack=0, ( 800x1280), ANativeWindow=0x7e8f09e010 (8:8:8:8), orient= 0 (type=00000000), flips=582141, isSecure=1, powerMode=2,
   activeConfig=0, numLayers=4
   v:[0,0,800,1280], f:[0,0,800,1280], s:[0,0,800,1280],transform:[[1.000,0.000,-0.000][0.000,1.000,-0.000][0.000,0.000,1.000]]
mAbandoned=0
- BufferQueue mMaxAcquiredBufferCount=2 mMaxDequeuedBufferCount=1
  mDequeueBufferCannotBlock=0 mAsyncMode=0
  default-size=[800x1280] default-format=1 transform-hint=00 frame-counter=1341
FIFO(0):
Slots:
 >[00:0x7e8f03da80] state=ACQUIRED 0x7e8f021e60 frame=1341 [ 800x1280: 800,  1]
  [02:0x7e8f03dd20] state=FREE     0x7e8f021f40 frame=1339 [ 800x1280: 800,  1]
  [01:0x7e8f03db60] state=FREE     0x7e8f021ed0 frame=1340 [ 800x1280: 800,  1]
```

- **Displays**

显示器列表及其信息

- **Slots**

显示器对应 framebufferTarget 的 buffer 状态

➢ EGL&OpenGL 参数

```
SurfaceFlinger global state:
EGL implementation : 1.4 Linux-r8p1-00rel0
EGL_KHR_image EGL_KHR_image_base EGL_KHR_image_pixmap EGL_KHR_gl_texture_2D_image EGL_KHR_gl_texture_cubemap_image EGL_KHR_gl_renderbuffer_image
EGL_KHR_fence_sync EGL_KHR_wait_sync EGL_KHR_swap_buffers_with_damage EGL_EXT_swap_buffers_with_damage EGL_ANDROID_image_native_buffer
EGL_ANDROID_recordable EGL_EXT_yuv_surface EGL_ANDROID_native_fence_sync EGL_ANDROID_framebuffer_target EGL_EXT_create_context_robustness
EGL_ANDROID_blob_cache EGL_KHR_create_context EGL_KHR_partial_update EGL_KHR_create_context_no_error
GLES: ARM, Mali-400 MP, OpenGL ES 2.0 Linux-r8p1-00rel0
GL_EXT_debug_marker GL_OES_texture_npot GL_OES_vertex_array_object GL_OES_compressed_ETC1_RGB8_texture GL_EXT_compressed_ETC1_RGB8_sub_texture
GL_OES_standard_derivatives GL_OES_EGL_image GL_OES_depth24 GL_ARM_rgba8 GL_ARM_mali_shader_binary GL_OES_depth_texture GL_OES_packed_depth_stencil
GL_EXT_texture_format_BGRA8888 GL_OES_vertex_half_float GL_EXT_blend_minmax GL_OES_EGL_image_external GL_OES_EGL_sync GL_OES_rgb8_rgba8
GL_EXT_multisampled_render_to_texture GL_EXT_discard_framebuffer GL_OES_get_program_binary GL_ARM_mali_program_binary GL_EXT_shader_texture_lod
GL_EXT_robustness GL_OES_depth_texture_cube_map GL_KHR_debug GL_ARM_shader_framebuffer_fetch GL_ARM_shader_framebuffer_fetch_depth_stencil
GL_OES_mapbuffer GL_KHR_no_error
```

➢ HWC 信息

```
Display 0 HWC layers:
------------------------------------------------------------------------------
 Layer name
     |    |   Z |  Comp Type |   Disp Frame (LTRB) |          Source Crop (LTRB)
------------------------------------------------------------------------------
 SurfaceView - com.softwinner.firepla[...].fireplayer.ui.VideoPlayerActivity#0
  4294967294 |     Device |    0  415  800  865 |    0.0    0.0 1920.0 1080.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 com.softwinner.fireplayer/com.softwinner.fireplayer.ui.VideoPlayerActivity#0
     |  21020 |     Device |    0    0  800 1280 |    0.0    0.0  800.0 1280.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 NavigationBar#0
     | 231000 |     Device |    0 1216  800 1280 |    0.0    0.0  800.0   64.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

h/w composer state:
  h/w composer enabled
    |layer      |  handle        |  format  |bl|space|TR|pAlp|       crop or color        |       frame         |zOrder|hz|ch|id|duto
-------------------------------------------------------------------------------------------------------------------------------------------
    0x78a2a3e354|    0x78a2a20180|        17| 1|    0| 0|1.00|[   0.0,   0.0,1920.0,1080.0]|[   0, 415, 800, 865]|     0| 0| 0| 0|HWC_LAYER
    0x78a2a3da94|    0x78a2a201f0|         1| 2|    0| 0|1.00|[   0.0,   0.0, 800.0,1280.0]|[   0,   0, 800,1280]|     1| 1| 1| 0|HWC_LAYER
    0x78a2a3e514|    0x78a2a20030|         1| 2|    0| 0|1.00|[   0.0,   0.0, 800.0,  64.0]|[   0,1216, 800,1280]|     2| 2| 2| 0|HWC_LAYER
disp:0 cur:692781-692780 ker:692781
-------------------------------------------------------------------------------------------------------------------------------------------
GL:29030400 GCL:29030400 GC:7411200 GR:3110400 DR:3110400 RM(C):3110400(0) RP(C):2073600(0)
memmalloc:10072
Allocated buffers:
0x7e8f021e60: 4000.00 KiB |  800 ( 800) x 1280 |    1 |        1 | 0x1e02 | FramebufferSurface
0x7e8f021ed0: 4000.00 KiB |  800 ( 800) x 1280 |    1 |        1 | 0x1e02 | FramebufferSurface
0x7e8f021f40: 4000.00 KiB |  800 ( 800) x 1280 |    1 |        1 | 0x1e02 | FramebufferSurface
0x7e8f0b5d70:  100.00 KiB |  800 ( 800) x   32 |    1 |        1 | 0xf02 | StatusBar#0
0x7e8f0b5e50:  200.00 KiB |  800 ( 800) x   64 |    1 |        1 | 0xf02 | NavigationBar#0
0x7e8f0b5fa0:  100.00 KiB |  800 ( 800) x   32 |    1 |        1 | 0xf02 | StatusBar#0
0x7e8f0b6010: 4000.00 KiB | 1280 (1280) x  800 |    1 |        2 | 0x933 | com.android.systemui.ImageWallpaper#0
0x7e8f0b61d0:  100.00 KiB |  800 ( 800) x   32 |    1 |        1 | 0xf02 | StatusBar#0
0x7e8f0b6240: 4000.00 KiB |  800 ( 800) x 1280 |    1 |        5 | 0x900 | Trapezoidal_Correction#0
0x7e8f0b6710: 4000.00 KiB |  800 ( 800) x 1280 |    1 |        5 | 0x900 | Trapezoidal_Correction#0
```

- **Display 0 HWC layers**
  送合成的图层信息，关注 crop 和 frame 大小，作为参考。
- **h/w composer state**
  图层合成策略状态，可以对照上面的信息确定对应图层是走 DE 还是走 GPU 合成，HWC_LAYER 是 DE，如果 zorder 是 65535 则为 FB_target
  下面是图层参数对应信息
  layer| handle|format|bl|space|TR|pAlp|crop or color|frame|zOrder|hz|ch|id|dutoX
  图层指针 | buffer 的地址 | buffer 格式 | 混合格式 | 旋转 | alpha | 剪裁区域| 帧大小 | Z 序|硬件 Z 序|通道|图层 | 合成策略
- **Allocated buffers**
  通过 surfaceflinger 分配的内存信息
  0x7f61c21e60: 4000.00 KiB | 800 (800) x 1280 | 1 | 1 | 0x1e02 | FramebufferSurface
  指针内存大小 | 分辨率 | | | buffer 标志位，特殊格式重要参数 | 名称

1.1.2. HWC 层日志调试

1.1.2.1. 操作命令

- 开启图层信息显示：
  adb shell setprop debug.hwc.showfps 2
  如为hwc1,x:
  logcat -s hwcomposer
  若为hwc2.0:
  logcat -s sunxihwc
- 关闭图层信息显示：
  setprop debug.hwc.showfps 0

1.1.2.2. 信息解析

Hwc1.X 显示结果如下：

```
Disp info:
 Fram Num|DP|Vsy|STP|CA(U)|Ad|IST|FPS (20,40,60)|MAX_LMTD|Cur_LMTD|TR_LMTD |All_Used|DSP_Used|CHNNEL_0|CHNNEL_1|CHNNEL_2|CHNNEL_3|timestamp
---------+--+---+---+-----+--+---+-------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+----------+
     4322| 0|Yes| No|1 (1)| 2|Yes|58.9,59.1,59.3|37324800|37324800| 2280000| 6021120| 6021120| 2170880| 3850240|       0|       0|0
---------+--+---+---+-----+--+---+-------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+----------+

Type|CH|V|SC H|SC W|PL|S|       Handle     |      Phyaddr     |  Usage  |  Flags  |Tr|Bld| Format  |        Source Crop      |         Frame Crop      | Reason
----+--+-+----+----+--+-+------------------+------------------+---------+---------+--+---+---------+-------------------------+-------------------------+-------------
 HWC| 0|N|1.00|1.00|ff|0| 0000000081457e00| 0000000072300000| 00000f02| 00000000|00|100| 00000004|[    0,    0,  752, 1280]|[   48,    0,  800, 1280]|I_OVERLAY
 HWC| 1|N|1.00|1.00|ff|0| 0000000081457aa0| 0000000073a00000| 00000f02| 00000000|00|105| 00000001|[    0,    0,  752, 1280]|[   48,    0,  800, 1280]|I_OVERLAY
 HWC| 0|N|1.00|1.00|ff|0| 0000000081dce500| 00000000700c0000| 00000f02| 00000000|00|105| 00000001|[    0,    0,   48, 1280]|[    0,    0,   48, 1280]|I_OVERLAY
  FB|-2|N|0.00|0.00|ff|0| 0000000081c4bfa0| 0000000070c00000| 00001e02| 00000000|00|105| 00000005|[    0,    0,  800, 1280]|[    0,    0,  800, 1280]|NOT_ASSIGNED
```

Hwc2.0 显示结果如下：

```
sunxihwc: dispy0(hwid0) cur:3733-3732-3733 mem:GL:49152000 GCL:49152000 GC:8294400 GR:0 DR:0 RM(C):8947848(0) RP(C):4147200(0)
sunxihwc:
sunxihwc:      layer      |  handle        |  format  | color  |bl|space|TR|pAlp|        crop                |        frame        |zOrder|hz|ch|id|duto
sunxihwc: ------------------------------------------------------------------------------------------------------------------------------------------------------
sunxihwc:       0xf2ccea0c|      0xf2783b60|       266|00000000| 1|    0| 0|1.00|[   0.0,   0.0,3840.0,2160.0]|[ 130, 320, 505, 530]|     0|-1|-1|-1|SCALE_OUT
sunxihwc:       0xf2cce8cc|      0xf2783460|         1|00000000| 2|  512| 0|1.00|[   0.0,   0.0,1920.0,1080.0]|[   0,   0,1920,1080]|     1|-1|-1|-1|CROSS_FB
sunxihwc:       0xf2cce28c|      0xf2cc8340|         1|00000000| 2|    0| 0|1.00|[   0.0,   0.0,1920.0,1080.0]|[   0,   0,1920,1080]| 65536| 0| 1| 0|HWC_LAYER
sunxihwc: memmalloc:9108
```

- **Fram Num**
  帧号，从 0 开始，一直往上增长
- **DP**
  真实的display 屏幕，0/1/..
- **Vsy**
  hw vsync 是否开启
- **STP**
  是否stop hwc，如果stop，所有的surface 都会走GPU 合成通道
- **CA(U)**
  为存储图层信息而分配的链表结点数目，（U）表示已经使用的个数，剩下的为空闲的个数，
- **Ad**
  链表结点中由于空闲个数过多，将要舍弃释放的结点个数，不包含在（U）之中。
- **IST**
  是否是在稳定期，在唤醒后的前60 帧以及从静止转为动态的前60 帧为非稳定期，该值为0。主要是休眠或者静态画面后ddr 会动态调整至较低的频率，如果一下子开放所有的DE 图层，可能会导致瞬间带宽的不足，所以在非稳定期，会限制一部分的DE 图层资源。
- FPS (20,40,60)
  每20 帧计算的帧率，该值用于计算当前的分配策略下的功耗以及全部使用GPU 合成的功耗数据，取功耗较小的合成方式，以节省功耗。
- **MAX_LMTD**
  DE 可用的带宽限制（包含所有的显示屏幕），如果送给DE 的图层消耗带宽超出该限制，可能会出现显示异常情况
- **Cur_LMTD**
  当前屏幕下的带宽限制，如果送给DE 该屏幕的图层消耗带宽超出该限制，可能会出现显示异常情况
- **TR_LMTD**
  rotate 硬件的带宽限制
- **All_Used**
  当前场景下DE 总共消耗的带宽（包含所有的显示屏幕）
- **DSP_Used**

当前屏幕下DE 消耗的带宽

- **CHNNEL_0~3**
  channel0 到 channe3 消耗的带宽
- **Timestamp**
  当前最新的hw vysnc 触发的时间戳图层信息
- **Type**
  "GLES"：GPU 合成, "HWC"：DE 合成, "BKGD"：背景, "FB"：framebuffer target, "SIDE"：（暂未使用）， "CURS"：硬件鼠标；
- **CH**
  使用的blending channel 索引号
- **V**
  是否是video 图像，以格式（yuv）判断
- **SC H**
  height 的放大倍数
- **SC W**
  width 的放大倍数
- **PL**
  plane alpha，面alpha
- **S**
  是否需要sync memory，是否需要刷cpu 的cache
- **Handle**
  buffer 的handle，与sf 传递下来的Handle 一致
- **Phyaddr**
  该buffer 的物理地址
- **Usage**
  handle 的usage
- **Flags**
  layer 的flag，与sf 传递下来的flag 一致
- **Tr**
  是否带旋转，旋转的角度是多少，0 表示不旋转，1：水平翻转，2：垂直翻转，3：180 度旋转，4：90 度旋转， 7：270 度旋转
- **Bld/bl**
  alpha blending ， 与sf 传递下来的blend 一致， 0x100 ： HWC_BLENDING_NONE ， 0x105: HWC_BLENDING_PREMULT ， 0x405:
  HWC_BLENDING_COVERAGE
- **Color**
  颜色值
- **pAlp**
  全局alpha 值0-1，0 表示全透，1 表示不透明；
- **zOrder**
  安卓层传下来的Z 序，越大越靠前，-1 表示走GPU 合成
- **Hz**
  硬件 Z 序，-1 表示走 GPU 合成
- **Ch**

图层分配到的DE 通道

- **Id**
  图层分配到的DE 通道中的图层id
- **Duto**
  同下面的reason
- **Format**
  图像的颜色格式
- **Source Crop**
  图像源的裁减区，只有裁减区内的内容会显示在屏幕上，<x_left, y_top, x_right, y_bottom>
- **Frame Crop**
  图像源的裁减区显示在屏幕上的矩形窗口，x_left, y_top, x_right, y_bottom
- **Reason**
  不能使用 DE 合成的原因。
  I_OVERLAY(HWC 合成) 、
  D_NULL_BUF（handle 是null）、
  D_CONTIG_MEM（非连续内存）、
  D_VIDEO_PD（视频保护）、
  D_CANNT_SCALE（由于不支持的图像格式）、
  D_SKIP_LAYER（surfaceflinger 不让DE 合成）、
  D_NO_FORMAT（DE 不支持的格式）
  D_BACKGROUND（由于是背景）、
  D_TR_N_0（没有旋转硬件模块）、
  D_ALPHA（不支持的alpha blending）、
  D_X_FB（跟FB 相交）、
  D_SW_OFTEN（由于CPU 频繁读写）、
  D_SCALE_OUT（scale 能力不够导致）
  D_STOP_HWC（通过系统属性设置停止使用HWC）、
  D_NO_PIPE（没有blending channel 导致）、
  D_NO_MEM（DE 运算能力不足）、
  D_MEM_CTRL（GPU 和 DE 合成的能耗控制导致，如果 GPU 合成功耗较低，则走 GPU 合成）

1.1.3. Display 驱动层日志调试

1.1.3.1. 操作命令

 # cat /sys/class/disp/disp/attr/sys

1.1.3.2. 信息解析

```
screen 0:
de_rate 696000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[370] force_sync[440] unblank direct_show[false]
dmabuf: cache[14] cache max[36] umap skip[7238] overflow[4708]
hdmi output mode(10) fps:60.6 1920x1080
err:8 skip:3510 irq:620319 vsync:619481 vsync_skip:1
BUF enable ch[0] lyr[0] z[0] prem[N] a[globl 255] fmt[ 1] fb[1920,1080; 960, 540; 960, 540] crop[ 0, 0,1920,1080] frame[ 5, 0,1915,1080]
addr[fb000000,fb1fa400,fb1fa400] flags[0x 0] trd[0,0]
BUF enable ch[1] lyr[0] z[1] prem[Y] a[globl 255] fmt[ 1] fb[1920,1080; 960, 540; 960, 540] crop[ 0, 0,1920,1080] frame[ 5, 0,1915,1080]
addr[f5800000,f59fa400,f59fa400] flags[0x 0] trd[0,0]
disp[0]all:5, sub:5, cur:5, free:3, skip:0
```

- screen
  显示通道
- de_rate
  de 的时钟频率
- ref_fps
  输出设备的参考刷新率
- lcd output backlight(97) fps:61.4 1280x 800
  屏输出 | 背光值（97）|屏刷新率:61.4hz|分辨率为：800x1280
- err
  de 缺数的次数
- skip
  de 跳帧的次数
- irq
  中断的次数
- Vsync
  已发送的 vsync 消息个数
- hdmi output mode(10) fps:60.6 1920x1080
  HDMI 输出 | 模式为（10：1080P@60HZ） | 刷新率为：60.6hz | 分辨率为：1920x1080
- Hdmi 参数
  mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[1] force_sync[1] unblank direct_show[false]
  mgr0: DE0 | 分辨率 | tcon 输出像素格式（YUV/RGB） | 颜色空间（BT709）| 颜色范围 | 光电强度 | 色 深 | err 更新寄存器是否在安全范围 | 强制刷新次数 | mgr 是否清掉数据 | 直传
- dmabuf 的参数
  dmabuf: cache[7] cache max[39] umap skip[66] overflow[166]
  dmabuf list 缓存编号 | 最大缓存编号 | 跳过 unmap 的次数 | overflow 之后会强制 unmap
- Composer 信息
  disp[0]all:29952, sub:29952, cur:29952, free:29948, skip:58
  总帧号 | 提交帧数 | 当前帧号 | 上一次释放帧号 | 跳过帧号数
- 图层信息说明：

BUF	图层类型，BUF/COLOR，一般为 BUF，即图层是带 BUFFER 的。COLOR 意思是显示一个纯色的画面，不带 BUFFER（dim layer）。
enable	显示处于 enable 状态
ch[0]	该图层处于 blending 通道 0
lyr[0]	该图层处于当前 blending 通道中的图层 0
z[0]	图层 z 序，越小越在底部，可能会被 z 序大的图层覆盖住

prem[Y]	是否预乘格式，Y 是，N 否
a	alpha 参数， globl/pixel/; alpha 值
fmt	图层格式，值 64 以下为 RGB 格式；以上为 YUV 格式，常见的 72 为 YV12, 76 为 NV12
fb	图层 buffer 的 size，width,height，三个分量
crop	图像 buffer 中的裁减区域， [x,y,w,h]
frame	图层在屏幕上的显示区域，[x,y,w,h]
addr	三个分量的地址，如果用 iommu 方式，那就是映射的设备虚拟地址，否则是物理地址
flags	一般为 0， 3D SS 时 0x4, 3D TB 时为 0x1, 3D FP 时为 0x2
trd	是否 3D 输出，3D 输出的类型（HDMI FP 输出时为 1）
err	de 缺数的次数，de 缺数可能会出现屏幕抖动，花屏的问题。de 缺数一般为带宽不足引起
skip	表示 de 跳帧的次数，跳帧会出现卡顿问题。跳帧是指本次中断响应较慢，de 模块判断在本次中断已经接近或者超过了消隐区，将放弃本次更新图像的机会，选择继续显示原有的图像
irq	表示该通路上垂直消隐区中断执行的次数，一直增长表示该通道上的 timing controller 正在运行当中
vsync	表示显示模块往用户空间中发送的 vsync 消息的数目，一直增长表示正在不断地发送中
fbd	Afbc 功能是否使能 Y 是 N 否

1.2. 显示通路通用调试方法

1.2.1. HWC 图层调试方法

1.2.1.1. 抓取静态图形显示

截主显命令：

```
screencap -p screen.png
```

截副显命令：

```
screencap -d 1 -p screen.png
```

1.2.1.2. 抓取动态图形显示

```
screenrecord screen.mp4
```

提示：无法录屏可以用手机录，使用可以帧回放的播放器逐帧回看问题现象细节。

1.2.1.3. 停用 HWC

Android setting 设置—〉开发者选项—〉停用硬件叠加层（Disable HW overlays）—〉打开

1.2.1.4. 查看图像帧率

开启帧率显示

```
setprop debug.hwc.showfps   1
logcat -s hwcomposer
```

```
logcat -s sunxihwc(hwc2.0)
```

关闭帧率显示

```
setprop debug.hwc.showfps   0
```

显示结果如下

D/hwcomposer(1551): >>>fps:: 40
D/hwcomposer(1551): >>>fps:: 57
D/hwcomposer(1551): >>>fps:: 21

1.2.1.5. 查看 HWC 层图层数据

开启 HWC 图层数据调试功能（d0 代表 display0，z0 代表 zorder0）

```
setenforce 0
setprop debug.hwc.showfps on.dump.d0z0
```

关闭 HWC 图层数据调试功能（d0 代表 display0，z0 代表 zorder0）

```
setprop debug.hwc.showfps off.dump.d0z0
```

把 data 目录下 bin 文件拷出（raw 数据），使用读图工具打开。

1.2.2. 显示驱动调试方法

1.2.2.1. 查看显示驱动回写数据

dump 回写数据

```
echo /data/test.bmp   >   /sys/class/disp/disp/attr/capture_dump
```

Adb pull 导出/data/test.bmp 文件后使用 PC 端读图工具进行查看

1.2.2.2. debugfs 调试接口

挂载 debugfs：

```
mount -t debugfs none /sys/kernel/debug
cd /sys/kernel/debug/dispdbg
```

有下面几个节点

```
command  info  name  param  start
```

- **切换指定分辨率指定显示接口**

步骤 1：使能功能

```
echo switch > command
```

步骤 2：选择显示接口(单显场景)

```
echo disp0 > name
```

步骤 3：选择显示接口(双显场景)

针对主显调试

```
echo disp0 > name
```

针对副显调试

```
echo disp1 > name
```

提示：如果不清楚当前的显示接口和分辨率等信息，可以通过 1.1.3 章节进行查询

步骤 4：输入切换参数

```
echo “显示接口代号  显示分辨率代号” > param
```

显示接口代号参数如下：

1: LCD 显示接口 2：tv 显示接口 4：HDMI 显示接口 8: VGA 显示接口 16: vdpo 接口 32: edp 显示接口；

显示分辨率代号可以查看源码：include/video/sunxi_display2.h, 其中 lcd 的分辨率是固定，所以当为 lcd 显示接口的时候，显示分辨率代号随便填，无影响;

步骤 5：启动切换

```
echo 1 > start
```

完整操作范例：

```
echo switch > command
echo disp0 > name
echo 2 14 > param;
echo 1 > start
```

- **开关显示（DE）**

```
echo blank > command
echo 0 > param          #关闭显示
```

或者

```
echo 1 > param          #开启显示
echo disp0 > name
echo 1 > start
```

- **显示的休眠和唤醒**

  显示模块可以独立于其它模块进行休眠唤醒测试。

  休眠：

  echo suspend > command
  echo disp0 > name
  echo 1 > start

  唤醒：

  echo resume > command
  echo disp0 > name
  echo 1 > start

- **HDMI 接口调试**

  查看某个分辨率是否支持:

  echo hdmi > name
  echo is_support > command
  echo 显示分辨率代号 > param
  echo 1 > start
  cat info

- **LCD 接口调试**

  获取当前背光值:

  echo getbl > command
  echo lcd0 > name
  echo 1 > start
  cat info

  设置当前背光值为 50:

  echo setbl> command
  echo lcd0 > name
  echo 50 > param
  echo 1 > start

  关闭 lcd 显示:

  echo disable > command
  echo lcd0 > name
  echo 1 > start

  打开 lcd:

  echo Enable > command
  echo lcd0 > name
  echo 1 > start

- **获取显示详细的信息**

下面和 cat /sys/class/disp/disp/attr/sys 效果一样
echo getinfo > command
echo disp0 > name
echo 1 > start
cat info

- **获取显示分辨率**
  echo getxres > command 或者 echo getyres > command
  echo disp0 > name
  echo 1 > start
  cat info

- **获取显示帧率 fps**
  echo getfps > command
  echo disp0 > name
  echo 1 > start
  cat info

- **获取显示 enhance 模块信息**
  echo getinfo > command
  echo enhance > name
  echo 1 > start
  cat info

- **获取显示 smbl 详细信息**
  echo getinfo > command
  echo smbl > name
  echo 1 > start
  cat info

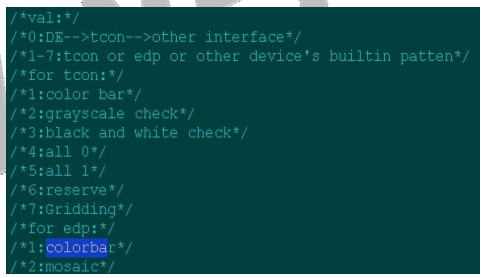
- **获取显示图层信息**
  echo layer > name
  echo getinfo > command
  echo 1 > start
  cat info

- **使能 VSYNC 事件**
  使能或者禁止 vsync 事件
  echo disp0 > name
  echo vsync_enable > command
  echo 1 > param
  echo 1 > start

1.2.2.3. sysfs 调试接口

cd /sys/class/disp/disp/attr

下面有如下节点：

capture_dump	回写 debug	echo /data/xx.bmp > capture_dump
disp	显示设备号	echo 0 / 1 > disp
xres	水平屏幕分辨率	cat xres
yres	垂直屏幕分辨率	cat yres
color_temperature	色温	
sys	显示相关信息	
enhance_mode	后处理开关 1 enable 2 demo	
enhance_edge	后处理-边沿处理（val： 0~10）	
enhance_saturation	后处理-饱和度（val： 0~10）	
enhance_denoise	后处理-降噪（val： 0~10）	
enhance_contrast	后处理-对比度（val： 0~10）	
enhance_bright	后处理-明亮度（val： 0~10）	
enhance_detail	后处理-锐化（val： 0~10）	
colorbar	DE/TCON 自刷新 colorbar 来 debug	
runtime_enable	使能 runtime 休眠唤醒的使用权限	

1.2.3. 显示性能调试方法

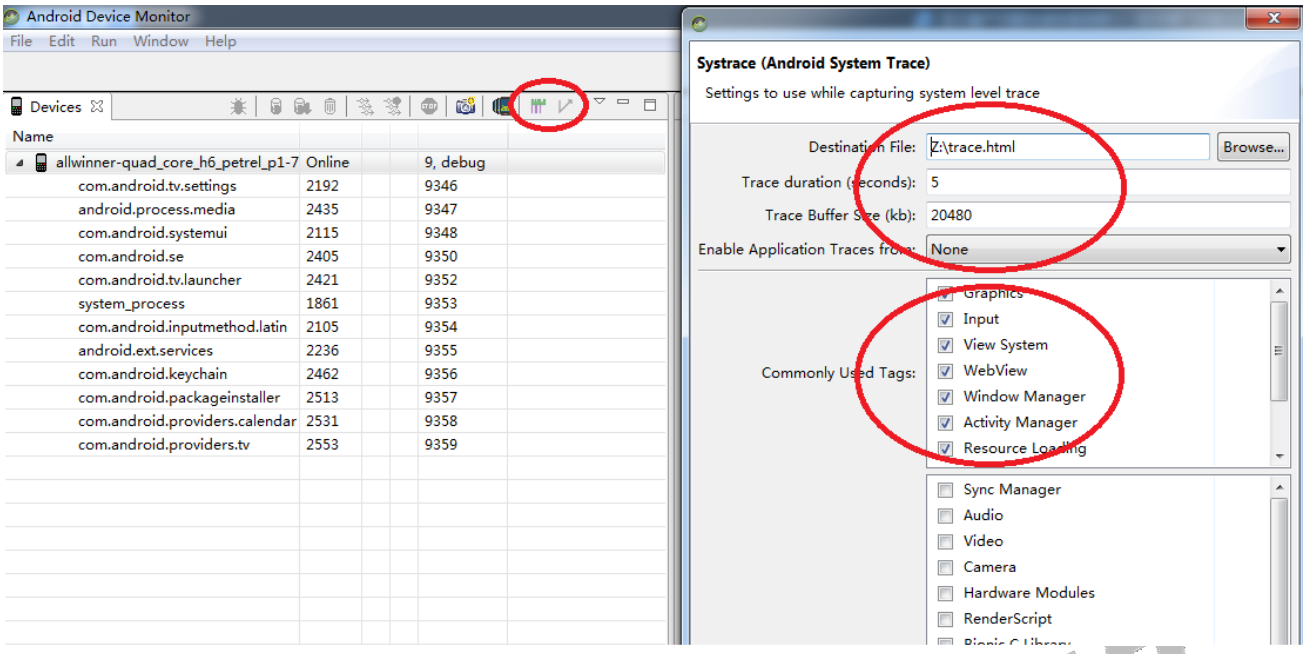
1.2.3.1. 显示平滑性 Systrace 调试

步骤 1：

安装 DDMS，即安装 eclipse 或 AndroidStudio、JAVA 环境，具体可参考网上教程；

步骤 2：

按照下图步骤点选 systrace 按钮、选择截取时长、截取数据大小、截取数据种类，保存到 trace.html；

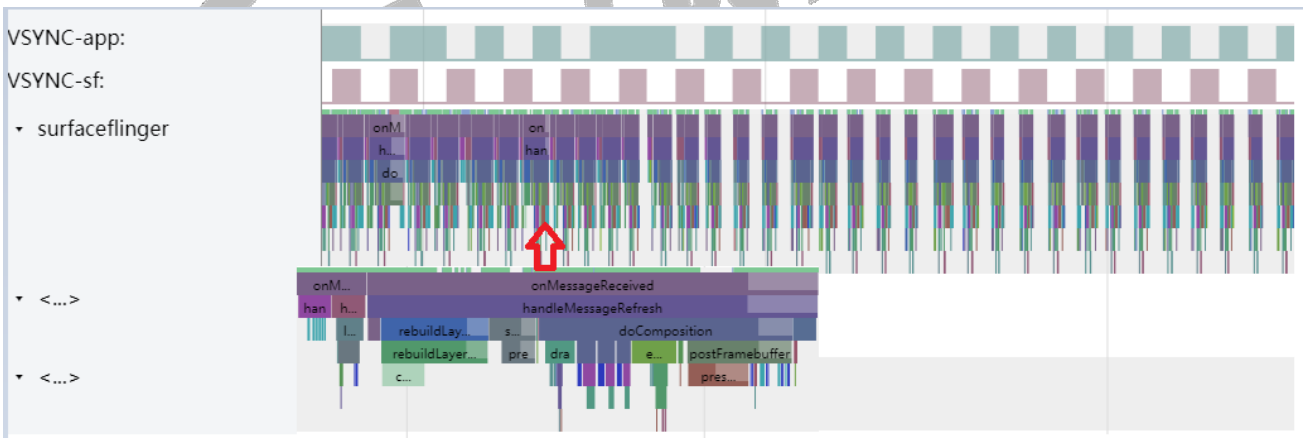


步骤 3：

用较新版本 chrome 浏览器打开。

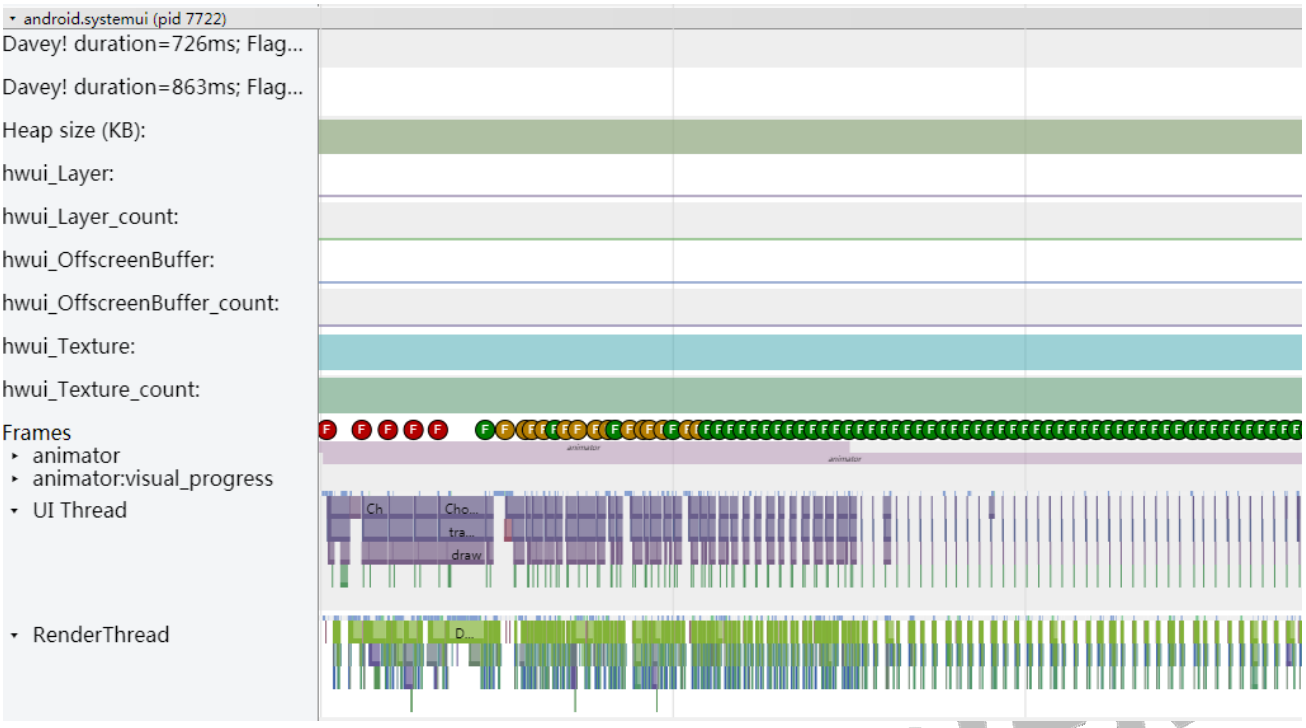
步骤 4：

查看 HWC_VSYNC_0、VSYNC-app、VSYNC-sf 或 surfaceflinger 是否均匀，对应判断硬件 vsync 是否中断不及时、应用是否绘图不及时、SurfaceFlinger 送帧不及时，如下图所示，即是 SurfaceFlinger 送帧不及时，放大可以看到具体的耗时函数。



步骤 5：

查看 frames 一栏有黄色甚至红色，判断应用是否绘制耗时过久。如下图，android.systemui 前半段一帧画太多数据了，请排查应用自身问题；



1.2.3.2. 内存占用

adb shell dumpsys meminfo

```
Applications Memory Usage (in Kilobytes):
Uptime: 11132312 Realtime: 11132312
```

各应用占用超过 100M 的，重点关注：

```
Total PSS by process:
    101,349K: system (pid 2075)
     43,927K: com.android.systemui (pid 15546)
     41,133K: com.softwinner.TvdFileManager (pid 5071)
     29,019K: zygote (pid 1693)
     28,211K: com.android.tv.launcher (pid 4392 / activities)
```

总内存使用和剩余情况，剩余少于 100M 的，内存不足造成系统卡顿时，显示效果会收到影响，造成用户体验卡顿。

```
Total RAM: 3,057,424K (status moderate)
 Free RAM: 2,244,954K (   142,790K cached pss + 1,124,412K cached kernel +     977,752K free)
 Used RAM:     521,864K (   406,868K used pss +     114,996K kernel)
 Lost RAM:     290,602K
       ZRAM:              4K physical used for              0K in swap (   262,140K total swap)
   Tuning: 192 (large 256), oom       81,920K, restore limit       27,306K (high-end-gfx)
```

1.2.3.3. surfaceflinger 进程堆栈调试

步骤 1：查询 surfaceflinger 进程 PID

```
ps -A | grep surfaceflinger
system        1707     1   155292  26856 SyS_epoll_wait      0 S surfaceflinger
```

步骤 2：调试 surfaceflinger 进程堆栈

```
debuggerd -b 1707
```

/*主线程堆栈信息头部*/

```
----- pid 1707 at 2018-12-13 08:39:30 -----
Cmd line: /system/bin/surfaceflinger
ABI: 'arm'
```

/*主线程堆栈*/

```
"surfaceflinger" sysTid=1707
    #00 pc 00053be4  /system/lib/libc.so (__epoll_pwait+20)
    #01 pc 00026079  /system/lib/libc.so (epoll_wait+16)
    #02 pc 0000f0d1  /system/lib/libutils.so (android::Looper::pollInner(int)+116)
    #03 pc 0000efdb  /system/lib/libutils.so (android::Looper::pollOnce(int, int*, int*, void**)+26)
    #04 pc 00066e7b  /system/lib/libsurfaceflinger.so (android::impl::MessageQueue::waitMessage()+62)
    #05 pc 00071931  /system/lib/libsurfaceflinger.so (android::SurfaceFlinger::run()+8)
    #06 pc 00002815  /system/bin/surfaceflinger (main+644)
    #07 pc 0008bc15  /system/lib/libc.so (__libc_init+48)
    #08 pc 0000254f  /system/bin/surfaceflinger (_start_main+46)
    #09 pc 00000306  <anonymous:ead49000>
```

/*软 vsync 产生线程*/

```
"DispSync" sysTid=1754
    #00 pc 00019d74  /system/lib/libc.so (syscall+28)
    #01 pc 0001d145  /system/lib/libc.so (__futex_wait_ex(void volatile*, bool, int, bool, timespec const*)+88)
    #02 pc 00063153  /system/lib/libc.so (pthread_cond_wait+32)
    #03 pc 00059447  /system/lib/libsurfaceflinger.so (android::DispSyncThread::threadLoop()+650)
    #04 pc 0000c0c7  /system/lib/libutils.so (android::Thread::_threadLoop(void*)+166)
    #05 pc 00063a25  /system/lib/libc.so (__pthread_start(void*)+22)
    #06 pc 0001df95  /system/lib/libc.so (__start_thread+22)
```

/*驱动 app 绘图的 vsync 线程*/

```
"appEventThread" sysTid=1756
    #00 pc 00019d74  /system/lib/libc.so (syscall+28)
    #01 pc 0001d145  /system/lib/libc.so (__futex_wait_ex(void volatile*, bool, int, bool, timespec const*)+88)
    #02 pc 00063153  /system/lib/libc.so (pthread_cond_wait+32)
    #03 pc 00049e59  /system/lib/libc++.so (std::__1::condition_variable::wait(std::__1::unique_lock<std::__1::mutex>&)+8)
    #04 pc 0005a725  /system/lib/libsurfaceflinger.so (android::impl::EventThread::threadMain()+756)
    #05 pc 0005b48d
```

```
    /system/lib/libsurfaceflinger.so
(_ZNSt3__114__thread_proxyINS_5tupleIJNS_10unique_ptrINS_15__thread_structENS_14default_deleteIS3_EEEEMN7android4impl
      #06 pc 00063a25  /system/lib/libc.so (__pthread_start(void*)+22)
      #07 pc 0001df95  /system/lib/libc.so (__start_thread+22)
```

/*驱动 surfaceflinger 送帧的 vsync 线程*/

```
    "sfEventThread" sysTid=1757
      #00 pc 00019d74  /system/lib/libc.so (syscall+28)
      #01 pc 0001d145  /system/lib/libc.so (__futex_wait_ex(void volatile*, bool, int, bool, timespec const*)+88)
      #02 pc 00063153  /system/lib/libc.so (pthread_cond_wait+32)
      #03 pc 00049e59  /system/lib/libc++.so (std::__1::condition_variable::wait(std::__1::unique_lock<std::__1::mutex>&)+8)
      #04 pc 0005a725  /system/lib/libsurfaceflinger.so (android::impl::EventThread::threadMain()+756)
      #05 pc 0005b48d /system/lib/libsurfaceflinger.so
(_ZNSt3__114__thread_proxyINS_5tupleIJNS_10unique_ptrINS_15__thread_structENS_14default_deleteIS3_EEEEMN7android4impl
      #06 pc 00063a25  /system/lib/libc.so (__pthread_start(void*)+22)
      #07 pc 0001df95  /system/lib/libc.so (__start_thread+22)
```

其他调试信息省略

1.2.3.4. 查看系统 Dram 带宽资源

mtop -m

命令调试信息如下：

```
    total: 17942, num: 10, Max: 1805, Average: 1794
    totddr   cpuddr   gpuddr   de_ddr   ve_ddr   csiddr   othddr
    1793       42        0      1743        0        0        8
    100.00    2.34     0.00    97.21     0.00     0.00     0.45
```

de_ddr：表示显示硬件引擎消耗 dram 带宽数量(1743)和占比(97.21)；

1.3. 显示通路问题排查思路

1.3.1.1. 工程排查思路介绍

步骤 1：

熟悉显示通路基础调试方法：

如果是 Android 系统跳第二步；如果是纯 linux 系统跳第四步；

步骤 2：

静止画面问题即参考 1.2.1 截图截屏命令，动态画面问题即录屏，如果出现问题，跳第三步；如果没有问题，跳第五步；

步骤 3：

参考 1.1 里面 Surfaceflinger 信息 dumpsys SurfaceFlinger 查看上层应用送显信息。

如果是进入某个应用出问题，查看该应用 buffer 状态，如果都是 free 就是应用没有送帧，从应用排查；查看该应用的图层合成信息，排查是否有特殊格式，如 afbc，是否帧大小超过屏大小；是否 handle 为空等；

步骤 4：

查看 1.1 显示驱动调试信息驱动显示信息，排查显示器类型是否正确，颜色空间是否正确，缺少页数是否增加，送图层的格式、混合模式、剪裁窗口大小、帧窗口大小等是否正确；

找到对应错误，如果是 hwc 传下来的，进行第五步；如果是纯 linux 系统排查调用 ioctl 设置参数的应用传参是否正确。

如果是驱动本身异常，请根据显示器类型到 HDMI、LCD 等接口章节查找问题。

步骤 5：

连续查看 hwc 图层信息参考命令 1.1 hwc 层图层日志信息，排查第四步发现的 HWC 对应参数错误引起原因，参照 HWC 代码进行修改。

Hwc1.X 主要修改源码在 hwc_sunxi.cpp 的 hwc_try_assign_layer 函数中；

Hwc2.0 主要修改源码在 DisplayOpr.cpp 的 TryToAssignLayer 函数中；

1.4. 常见问题排查方法

1.4.1. 显示黑屏/绿屏/卡住常见问题排查

- 问题现象：

屏幕出现黑屏、蓝屏、卡住等无法显示类问题

- 排查方法：

步骤 1：

排查硬件连接是否正确，是否连接紧。

步骤 2：

确认串口或者 adb 是否还能连接，cpu 有没有挂掉，如果正常使用，则进行第三步，如果已经无法输入，要靠 DS5 或者 gdb 等其他调试方式，去看死前停在哪里，在代码里继续排查。

步骤 3：

查看 1.1 显示驱动调试信息，确认显示器切换到对应显示接口进行显示。

例如：插入 hdmi，正常情况下应该显示 hdmi output，如果是其他设备，或者压根就没有图层信息，那么就要排查整条显示通路流程是否异常，查看当前 sysconfig 显示配置是否正确。如确认没有问题，进入步骤 4。

步骤 4：

使用 1.2.1 截图截图方法，查看抓取回来的数据是否正常。:正常情况下，应该是一张颜色大小正常的图片，跟解码出来的效果一致。如果不正常或无法截图，进行步骤 5。如果正常，进行步骤 7。

步骤 5：

根据 1.1 操作查看 Surfaceflinger 信息 surfaceflinger 调试信息。如果无法打出信息，说明 surfaceflinger 不工作了，跳步骤 6；如果发现图层信息为空，为框架层问题，需要 Android 系统人员帮忙协助进一步分析应用层是否存在异常；

步骤 6：

按照 1.2.3 章节打印进程堆栈，打印 surfaceflinger 进程堆栈，如果无法打印，说明 surfaceflinger 进程存在异常，用 logcat 查看崩溃堆栈。如果多次打印发现主线程堆栈没有变化且调用比较长，说明进程被卡住了。如果卡在锁的位置， 很可能死锁；如果卡在某个调用，请排查对应模块；

步骤 7：

根据 1.1 章节显示驱动调试信息查看驱动显示信息，排查显示器类型是否正确，颜色空间是否正确，: 缺少页数是否增加，送图层的格式、混合模式、剪裁窗口大小、帧窗口大小等是否正确；

找到对应错误后，如果是 hwc 传下来的，进行按照 1.1hwc 层图层日志信息查看 hwc 图层信息；如果是纯 linux 系统排查调用 ioctl 设置参数的应用传参是否正确。

如果是驱动本身异常，请根据显示器类型到 HDMI、LCD 等接口章节查找问题。

1.4.2. 显示花屏/闪屏等问题

- 问题现象：

  偶尔花屏、闪屏等能显示但有局部小异常的问题

- 排查方法：

步骤 1：

按照 1.2.1 章节停用 HWC 停用 HWC，如果画面正常，进行步骤 3；如果异常，进行步骤 2；

步骤 2：

根据 1.1 节 Surfaceflinger 信息 dumpsys SurfaceFlinger 查看应用传递图层信息。

检查该应用的图层合成信息，排查是否有特殊格式，如 afbc，是否帧大小超过屏大小；

检查 handle 是否为空；

检查预乘图层是否送到 DE0；

检查是否缩小比小于 1/16；

检查是否放大比超过 32 倍等；

如果以上检查都正确，请 Android 系统人员协助排查应用是否存在异常或者 GPU 异常。

步骤 3：

根据 1.1 节显示驱动调试信息查看驱动显示信息，做以下检查：

检查排查送图层的格式、混合模式、剪裁窗口大小、帧窗口大小，对齐方式等是否正确；

如果通过检查找到对应错误，需要进一步确认是否为 HWC 层异常导致，进行步骤 4；

如果是 linux 系统则需要排查应用层通过 ioctl 调用 DISP 驱动设置参数的应用传参是否正确；

如果是 DISP 驱动层存在异常，需要根据显示接口类型到以下章节中对应 HDMI、LCD 查找问题。

步骤 4：

连续查看图层信息，排查第三步发现的 HWC 对应参数错误引起原因，让显示组对应负责人。

1.4.3. 系统卡顿导致的显示异常问题

- 问题现象：

  卡顿、跳帧、高帧率高分辨率时花屏、快速切换界面时花屏

- 排查方法：

步骤 1：

根据 1.2.3 章节抓 Systrace，如果是应用绘制过多，请应用层减少每帧绘图工作量；

如果播放视频场景，发现送帧不及时，请找多媒体中间件同事处理；

如果 surfaceflinger 送帧不及时，排查耗时的函数，找出原因；如果耗时出现在显示驱动的 ioctl，请执行步骤 5；

如果检查 GPU 绘图过慢，可尝试增加 GPU 工作频率，或者增加 bufferqueue 的 buffer 数（通过 layer.cpp 的 onFisrtRef 函数中 mProducer->setMaxDequeuedBufferCount(5)）；

步骤 2：

根据 1.2.3 章节 Cpu 占用查看 CPU 占用，发现 CPU 负载紧张，请排查系统 CPU 负载消耗问题；

步骤 3：

根据 1.2.3 章节查看内存占用内存占用，发现剩余内存不足，请排查系统内存消耗问题；

步骤 4：

根据 1.2.3 章节查看带宽查看带宽，如果发现带宽不足，请排查对应模块资源占用情况；

步骤 5：

根据 1.1 章节显示驱动调试信息查看驱动显示信息，如果有缺页异常，排查 hwc 送帧时序和驱动中断是否正常；

如果 ref_fps 帧率过低，请找显示组负责人调整 sys_config.fex 中的参数。

1.4.4. 显示驱动打印 invalid address 问题

- 问题现象：

  串口狂打印”DE invalid address: 0x0, data:0x0, id:0x1”

- 排查步骤：

步骤 1：

根据 1.1 章节显示驱动调试信息打印显示驱动信息，查看访问地址和 size 是否正常

如果 DE 有访问报错地址，那就要看是否内存申请小了，或者对齐没做好，再或者是访问了一片已经释放的内存，进行步骤 2；

如果以上检查不出问题，请提供完整显示驱动调试信息、Android logcat 日志信息、Kernel dmesg 日志信息给到全志原厂 FAE 协助分析；

步骤 2：

根据 1.1 章节 hwc 层图层日志信息查看 hwc 图层信息，确定问题原因。

1.4.5. 显示驱动打印 dmabuf get fd failed 问题

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  显示 buffer 句柄 fd 提前释放或不存在，非驱动问题，排查显示应用 buffer 调用流程。

1.4.6. 显示驱动打印 dma_map_sg failed 问题

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  ION 申请 buffer 用错 flag，要检查
  Cma 内存：ION_HEAP_TYPE_DMA
  Iommu：ION_HEAP_TYPE_SYSTEM

1.4.7. 显示界面泛白

- 问题现象：
  显示界面发白发亮。
- 排查方法

步骤 1：

通常此类问题多属于预乘问题导致，根据 1.1 章节 hwc 层图层日志信息查看 hwc 传递图层信息，是否把带有预乘的图层放到了 video 图层，如果有，则修改 hwc 代码，如果没有，进入步骤 2；

步骤 2：

检查应用层，是否将非预乘的图层，当作预乘图层来传递；如果存在此类问题，应用端无法修改的情况下，可尝试通过 HWC 或者 GPU 去解决。例如只有一层图层存在此问题，可选择规避方法发到最底层 z 序；如果多图层都有这个问题，只能 apk 走 GPU 的方法规避。

1.4.8. 显示的主显和辅显设备输出内容混乱

- 问题现象：
  主屏意外出现副屏内容，或者副屏出现主屏内容
- 排查方法：
  最新代码已经解决问题，确认当前分支有没有此补丁。

```
disp2: fix the bug: multi thread call in case of disp 0 & 1 work together
```

1.4.9. 平板亮度调至最低画面黑屏

- 现象：
  平板亮度调到最低，什么都看不到了，黑屏。
- 第一步：
  根据 1.1 章节 Surfaceflinger 信息 dumpsys display 查看安卓上层亮度设置的值，正常情况下应该是一个非 0 值，若此时为 0，那就是最低值设置过低，进入第二步，如果非零，进入第三步
- 第二步：
  修改安卓系统配置文件 config.xml 的最低亮度。
- 第三步：
  根据 1.1 节显示驱动调试信息查看显示驱动信息中的 backlight 是否同第一步，相同，则找屏驱动同事去看，不同需要查看 hal 层 light 代码

2. HDMI1.4 调试手段

2.1. HDMI1.4 日志信息调试

2.1.1. 使能日志调试

1). 调整内核体质打印等级：

```
echo 8 > /proc/sysrq-trigger
```

2). 打开 HDMI1.4 日志调试开关：

```
echo 1 > /sys/class/hdmi/hdmi/attr/debug
```

3). 连接 HDMI，从串口获取到内核日志信息，分段说明如下：

【调试信息头部】

```
[HDMI] plugin
[HDMI] HDMI_State_EDID_Parse
[HDMI] ParseEDID
[HDMI] DDC_Read
```

【EDID BANK block0 数据】

如果出现 EDID block 0 checksum error 打印说明 DDC 通信失败；

如果出现 EDID block0 header error 的打印说明 DDC 通信失败；

```
[HDMI] Sink : EDID bank 0:
[HDMI]  0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
[HDMI]
================================================================================
==================
[HDMI]  00  ff  ff  ff  ff  ff  ff  00  22  45  9b  06  01  00  00  00
[HDMI]  0f  17  01  03  80  ba  69  78  0a  ee  91  a3  54  4c  99  26
[HDMI]  0f  50  54  bd  c8  00  81  00  81  40  81  80  95  00  a9  40
[HDMI]  01  01  01  01  01  01  04  74  00  30  f2  70  5a  80  b0  58
[HDMI]  8a  00  44  17  74  00  00  1e  02  3a  80  18  71  38  2d  40
[HDMI]  58  2c  45  00  a0  5a  00  00  00  1e  00  00  00  fd  00  18
[HDMI]  4b  1a  51  1e  00  0a  20  20  20  20  20  20  00  00  00  fc
[HDMI]  00  48  41  49  45  52  20  34  4b  32  4b  55  48  44  01  ea
[HDMI]
================================================================================
==================
[HDMI] EDID version: 1.3
[HDMI] PCLK=297000000    Xsize=3840     Ysize=2160     Frame_rate=30
[HDMI] PCLK=148500000    Xsize=1920     Ysize=1080     Frame_rate=60
[HDMI] DDC_Read
```

【EDID BANK block1 数据】

如果出现 EDID block 1 checksum error 打印信息，说明 DDC 通信失败；

如果出现 EDID block1 header error 打印信息，说明 DDC 通信失败；

```
[HDMI] Sink : EDID bank 1:
[HDMI]  0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
[HDMI]
================================================================================
==================
[HDMI]  02  03  27  f1  4b  90  9f  04  13  05  14  03  12  20  21  22
[HDMI]  23  09  07  07  83  01  00  00  6e  03  0c  00  30  00  b8  3c
[HDMI]  20  80  80  01  02  03  04  01  1d  00  bc  52  d0  1e  20  b8
[HDMI]  28  55  40  c4  8e  21  00  00  1e  02  3a  80  d0  72  38  2d
[HDMI]  40  10  2c  45  80  a0  5a  00  00  00  1f  01  1d  80  d0  72
[HDMI]  1c  16  20  10  2c  25  80  a0  5a  00  00  00  9f  00  00  00
[HDMI]  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
[HDMI]  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  8f
[HDMI]
================================================================================
==================
```

【HDMI sink 数据】

显示 sink 是否支持 yuv 格式，以下为 sink 支持哪些分辨率，分辨率的 VIC 定义于 CEA-861 协议]

```
[HDMI] device support YCbCr444 output
[HDMI] Parse_VideoData_Block: VIC 16(native) support
[HDMI] Parse_VideoData_Block: VIC 31(native) support
[HDMI] Parse_VideoData_Block: VIC 4 support
[HDMI] Parse_VideoData_Block: VIC 19 support
[HDMI] Parse_VideoData_Block: VIC 5 support
[HDMI] Parse_VideoData_Block: VIC 20 support
[HDMI] Parse_VideoData_Block: VIC 3 support
[HDMI] Parse_VideoData_Block: VIC 18 support
[HDMI] Parse_VideoData_Block: VIC 32 support
[HDMI] Parse_VideoData_Block: VIC 33 support
[HDMI] Parse_VideoData_Block: VIC 34 support
[HDMI] Parse_AudioData_Block: max channel=2
[HDMI] Parse_AudioData_Block: SampleRate code=7
[HDMI] Parse_AudioData_Block: WordLen code=7
[HDMI] Find HDMI Vendor Specific DataBlock
```

【HDID 显示输出模式】

```
[HDMI] 3D_present              [ 支持 3D 模式 ]
[HDMI] Parse_HDMI_VSDB: VIC 1 support  [ 支持 4K 显示, 3840x2160@30hz ]
[HDMI] Parse_HDMI_VSDB: VIC 2 support  [ 支持 4K 显示, 3840x2160@25hz ]
```

```
[HDMI] Parse_HDMI_VSDB: VIC 3 support  [ 支持 4K 显示, 3840x2160@24hz ]
[HDMI] Parse_HDMI_VSDB: VIC 4 support  [ 支持 4K 显示, 4096x2160@24hz ]
[HDMI] PCLK=74250000    Xsize=1280     Ysize=720      Frame_rate=50
[HDMI] PCLK=148500000   Xsize=1920     Ysize=1080     Frame_rate=50
[HDMI] PCLK=74250000    Xsize=1920     Ysize=540      Frame_rate=50
```

【HDID 其他信息】

```
[HDMI] switch_set_state 1
```

往 switch 结点向用户空间汇报插入消息（1）；如果是拔出则汇报拔出消息（0）

```
[HDMI] set_video_enable = 1!
[HDMI] video_on @ set_video_enable = 0!
[HDMI] video_config,
vic:19,cts_enable:0,isHDMI:1,YCbCr444_Support:1,hdcp_enable:0
[HDMI] hdmi video + audio
[HDMI] video_on @ video_config = 0!
```

- cts_enable：
  表示 cts_compatible 配置选项的值； isHDMI 为 1 表明当前连接的显示器是 HDMI 设备，0 则为 DVI 设备；
- YCbCr444_Support：
  为 1 表示当前连接的显示器支持 ycbcr 输入;
- hdcp_enable：
  表示 hdmi_hdcp_enable 配置选项的值，1 表示需要开启 hdcp 功能, 0 表示不需要开启 hdcp 功能

```
[HDMI] set_video_enable, vic:19,is_hdmi:1,is_yuv:1,is_hcts:0
```

- is_hdmi:
  1 表示 hdmi 控制器输出 hdmi 模式，0 表示输出 dvi 模式；is_yuv：1 表示 hdmi 控制器输出 yuv 格式，0 则为 rgb 格式；
- is_hcts：
  1 表示 hdmi 控制器开启 hdcp 功能，0 表示关闭 hdcp 功能

2.2. HDMI1.4 基础调试手段

2.2.1. 查看热插拔状态

linux-3.10 内核版本使用以下命令:
cat /sys/class/switch/hdmi/state
linux-4.4 及以上版本使用以下命令
cat /sys/class/extcon/hdmi/state

2.2.2. 强制 HDMI 插入/拔出

【强制 HPD 插入】
echo 0x11 > /sys/class/hdmi/hdmi/attr/hpd_mask

【强制 HPD 拔出】
echo 0x10 > /sys/class/hdmi/hdmi/attr/hpd_mask

2.2.3. 屏蔽 HDMI 热插拔检测

echo 0x100 > /sys/class/hdmi/hdmi/attr/hpd_mask

2.2.4. 运用 debugfs 切换分辨率

【手动切换 video hdmi 分辨率】
mount -t debugfs none /mnt
cd /mnt/dispdbg

echo [显示通道] > name
echo switch1 > command
echo [输出接口] [tv_mode] > param
echo 1 > start
tv_mode 的定义：

```
DISP_TV_MOD_480I                = 0,
DISP_TV_MOD_576I                = 1,
DISP_TV_MOD_480P                = 2,
DISP_TV_MOD_576P                = 3,
DISP_TV_MOD_720P_50HZ           = 4,
DISP_TV_MOD_720P_60HZ           = 5,
DISP_TV_MOD_1080I_50HZ          = 6,
DISP_TV_MOD_1080I_60HZ          = 7,
DISP_TV_MOD_1080P_24HZ          = 8,
DISP_TV_MOD_1080P_50HZ          = 9,
DISP_TV_MOD_1080P_60HZ          = 0xa,
DISP_TV_MOD_1080P_24HZ_3D_FP    = 0x17,
DISP_TV_MOD_720P_50HZ_3D_FP     = 0x18,
DISP_TV_MOD_720P_60HZ_3D_FP     = 0x19,
DISP_TV_MOD_1080P_25HZ          = 0x1a,
DISP_TV_MOD_1080P_30HZ          = 0x1b,
DISP_TV_MOD_PAL                 = 0xb,
DISP_TV_MOD_PAL_SVIDEO          = 0xc,
DISP_TV_MOD_NTSC                = 0xe,
DISP_TV_MOD_NTSC_SVIDEO         = 0xf,
DISP_TV_MOD_PAL_M               = 0x11,
DISP_TV_MOD_PAL_M_SVIDEO        = 0x12,
DISP_TV_MOD_PAL_NC              = 0x14,
DISP_TV_MOD_PAL_NC_SVIDEO       = 0x15,
DISP_TV_MOD_3840_2160P_30HZ     = 0x1c,
DISP_TV_MOD_3840_2160P_25HZ     = 0x1d,
DISP_TV_MOD_3840_2160P_24HZ     = 0x1e,
DISP_TV_MOD_4096_2160P_24HZ     = 0x1f,
DISP_TV_MOD_4096_2160P_25HZ     = 0x20,
DISP_TV_MOD_4096_2160P_30HZ     = 0x21,
DISP_TV_MOD_3840_2160P_60HZ     = 0x22,
DISP_TV_MOD_4096_2160P_60HZ     = 0x23,
```

范例：

```
echo switch > command;
echo 4 10 > param;
echo 1 > start
```

如上面的命令的参数 4 就代表 HDMI 输出，10 代表 1080p@60 ；

2.2.5. HDMI 寄存器 dump 调试

步骤 1：

查看/sys/class/hdmi/hdmi/attr/目录下是否有 dump 节点；

步骤 2：

如果没有 dump 节点，需要打开 HDMI 驱动代码的宏：HDMI_ENABLE_DUMP_WRITE，然后再重新编译内核和固件包，重新烧录固件后，上电启动机器，检查 dump 节点是否生成；

步骤 3：

输入命令 echo value > /sys/class/hdmi/hdmi/attr/dump

其中 value 的后 16 位为寄存器地址，前 16 位是寄存器的个数。

2.2.6. HDMI 寄存器 write 调试

步骤 1

查看/sys/class/hdmi/hdmi/attr/目录下是否有 write 节点；

步骤 2：

如果没有 write 节点，需要打开 HDMI 驱动代码的宏：HDMI_ENABLE_DUMP_WRITE，然后再重新编译内核和固件包，重新烧录固件后，上电启动机器，检查 write 节点是否生成；

步骤 3：

输入命令 echo value > /sys/class/hdmi/hdmi/attr/write，其中 value 的后 16 位为寄存器地址，前 16 位是寄存器的个数。

2.3. HDMI1.4 基础配置检查

2.3.1. 内核 menuconfig 检查

【linux-3.10】
（1）CONFIG_SWITCH=y
（2）CONFIG_DISP2_SUNXI=y
（3）CONFIG_HDMI_DISP2_SUNXI=y
（4）CONFIG_DISP2_SUNXI_DEBUG=y

【linux-4.4 及以上版本】
（1）CONFIG_EXTCON=y
（2）CONFIG_DISP2_SUNXI=y
（3）CONFIG_HDMI_DISP2_SUNXI=y
（4）CONFIG_DISP2_SUNXI_DEBUG=y

2.3.2. Dts 配置检查

【基本配置】

```
hdmi: hdmi@01ee0000 {
        compatible = "allwinner,sunxi-hdmi";
        reg = <0x0 0x01ee0000 0x0 0x20000>;
        clocks = <&clk_hdmi>,<&clk_hdmi_slow>;
};
```

需要结合 IC 的 spec，检查上面的寄存器首地址

【时钟配置】

```
clk_hdmi: hdmi {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        assigned-clock-parents = <&clk_pll_video1>;
        clock-output-names = "hdmi";
};
clk_hdmi_slow: hdmi_slow {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        clock-output-names = "hdmi_slow";
};
```

2.3.3. Sys_config.fex 配置检查

【hdmi 节点】

参数	说明
hdmi_used	1：使用 hdmi　0：不使用 hdmi
hdmi_hdcp_enable	1:使用 hdcp　0:不使用 hdcp
hdmi_cec_support	1:支持 cec　0:不支持 cec
hdmi_cts_compatibility	1：驱动兼容 hdmi cts 测试，符合 HDMI CTS 测试的标准； 0：驱动部分配置根据实际情况进行配置，部分配置不采用 HDMI CTS 测试标准

2.4. HDMI1.4 常见问题排查

2.4.1. 显示问题---插入 HDMI 无显示

步骤 1：

查看是否加载了 HDMI 模块：cd /sys/class/hdmi/hdmi/attr，查看是否有这个目录。如果没有这个目录，说明 HDMI 模块没有被加载；

步骤 2：

cat /sys/class/disp/disp/attr/sys，从显示框架上查看显示是否是切换到 hdmi 输出,检查相关参数是否正确

```
screen 0:
de_rate 432000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank dir
ect_show[false]
dmabuf: cache[0] cache_max[0] umap skip[0] overflow[0]
       hdmi output mode(10)     fps:60.6        1920x1080
        err:2    skip:4342      irq:495972      vsync:0 vsync_skip:0
   BUF    enable ch[2] lyr[0] z[0] prem[N] a[globl 255] fmt[  0] fb[ 240, 320; 240, 320; 240, 320] crop[
  0,    0, 240, 320] frame[    0,    0, 240, 320] addr[70080000,       0,       0] flags[0x       0] trd[0,0
]
```

步骤 3：

然后插拔一下，通过“HDMI 插入，log 信息调试”这一章节，判断当前 HDMI 输出的分辨率，sink 端是否支持。

步骤 4：

观察一下电视，看电视有没有识别 HDMI 信号，如果有 HDMI 信号，则可以通过 sunxi_dump 设置寄存器的方式，设置 tcon 内部输出图像到 HDMI（需要结合 IC 的 spec 查看 tcon_tv（tcon1）的寄存器，找到 TCON_CTL_REG 寄存器，找到 SRC_SEL 相关位，然后设置输出源），在看看输出的图像是否正常。如果图像正常，则说明 HDMI 端配置没有问题，应该检查 DE 端的设置。否则，则应该怀疑 tcon 和 hdmi 的问题。

步骤 5：

如果以上步骤查不出问题，应该联系负责 HDMI 驱动的工程师分析，将以上步骤的现象反映给驱动工程师。

2.4.2. 显示问题---间歇性黑屏或出现线状干扰线

步骤 1：

需要联系原厂专业的 HDMI 驱动或硬件工程师，用协议分析仪分析 HDMI 输出信号；

步骤 2：

如果步骤一中发现 HDMI 输出信号确实有问题，但是 TCON 和 HDMI 的配置没有问题，应该怀疑 PLL 的稳定性问题；

2.4.3. Audio 问题---audio 卡顿

步骤 1：

需要 Audio 驱动工程师，先查看确认 Audio 模块是否有声音输出；

步骤 2：

联系 HDMI 驱动工程师，联系 HDMI 驱动工程师用协议分析仪查看显示的 timing 参数，看显示是否影响到了 HDMI audio，特别需要重点检查 vsync、hsync 参数；

步骤 3：

如果以上步骤查不出问题，需要联系原厂专业的 HDMI 驱动的工程师协助分析。

2.4.4. Audio 问题---audio 无声音

步骤 1：

需要 Audio 驱动工程师，先查看确认 Audio 模块是否有声音输出；

步骤 2：

通过“HDMI 插入，log 信息调试”这一章节，查看 HDMI 是不是工作到了 DVI 模式；

步骤 3：

如果以上步骤查不出问题，应该联系负责 HDMI 驱动的工程师分析。

3. HDMI2.0 调试手段

3.1. HDMI2.0 日志信息调试

3.1.1. HDMI 日志打开方式

【设置系统打印等级】
echo 8 > /proc/sysrq-trigger

【设置 hdmi 内部打印等级】
echo [log_level] > /sys/class/hdmi/hdmi/attr/debug

log_level 的意义:
1---只打印 video 的 log
2---只打印 edid 的 log;
3---只打印 audio 的 log;
4---只打印 video + edid + audio 的 log;
5---只打印 cec 的 log;
6---只打印 hdcp 的 log;
7---以上的都打印;
8---以上的都打，log trace 也打印;

3.1.2. Video log 信息

【video log 信息】
echo 1 > /sys/class/hdmi/hdmi/attr/debug
插拔 HDMI 可以看到以下信息：

```
[ 4113.543420] HDMI cable is connected
[ 4113.613984] IRQ write unmute: irq[7] mask[0]
[ 4115.913351] Sink Support cea vic mode:16
[ 4115.918740] [HDMI2.0]CEA VIC=16
[ 4115.922251] 1920x1080p@60 fps
[ 4115.925634] 16:9, 8-bpp
[ 4115.928399] YCbCr-444
[ 4115.930933] BT709
[ 4115.933076] eotf:SDR_LUMINANCE_RANGE
[ 4115.937205] interpolation:0  decimation:0  color_depth:4
[ 4115.953288] PHY interface reconfiguration, set to I2CPHY PLL locked
```

3.1.3. Edid log 信息

【edid log 信息】

echo 2 > /sys/class/hdmi/hdmi/attr/debug

插拔 HDMI 可以看到以下信息：

```
[ 4270.453326] HDMI cable is connected
[ 4270.490444]
[ 4270.490444]
[ 4270.490444] EDID Block0 detailed discriptor:
[ 4270.498525]
[ 4270.498525] pixel_clock:148500000
[ 4270.503920] hactive * vactive: 1920 * 1080
[ 4270.508492]
[ 4270.508492] pixel_clock:74250000
[ 4270.513757] hactive * vactive: 1920 * 540
[ 4270.518226] Monitor name: HDMI Analyzer
[ 4270.522512] EDID Extension 1
[ 4270.558922] EDID: Video datablock parsing
[ 4270.563425] EDID: Video datablock parsing
[ 4270.567897] EDID: Video datablock parsing
[ 4270.572371] EDID: Audio datablock parsing
[ 4270.576855] SAD block parsing
[ 4270.580163]
EDID: VSDB HDMI and HDMI-F
 EDID HDMI VSDB parsed
[ 4270.586904]
[ 4270.588571] EDID CEA Extended field 0x07
[ 4270.592942] Video Capability Data Block
[ 4270.597250] EDID CEA Extended field 0x07
[ 4270.601626] Colorimetry Data Block
[ 4270.605441] EDID CEA Extended field 0x07
[ 4270.609814] HDR Static Metadata Data Block
```

3.1.4. Audio log 信息

【audio log 信息】

echo 3 > /sys/class/hdmi/hdmi/attr/debug

插拔 HDMI 可以看到以下信息：

```
[ 4395.433329] HDMI cable is connected
[ 4397.774477] Audio interface type = I2S
[ 4397.778673] Audio coding = PCM
[ 4397.782078] Audio frequency = 44100Hz
[ 4397.786190] Audio sample size = 16
[ 4397.789989] Audio FS factor = 64
[ 4397.793603] Audio ChannelAllocationr = 0
[ 4397.797975] Audio mChannelNum = 2
[ 4397.801743] channal 0 is enable
[ 4397.805301] channal 1 is enable
[ 4397.808806] channal 2 is enable
[ 4397.812307] channal 3 is enable
[ 4397.815830] channal 7 is enable
[ 4397.819360] DEBUG:Audio N value = 6272
[ 4397.823568] DEBUG:Audio channel count = 1
[ 4397.828037] DEBUG:Audio channel allocation = 0
[ 4397.832991] DEBUG:Audio level shift = 0
```

3.1.5. Cec log 信息

【cec log 信息----不使用标准 cec】
echo 5 > /sys/class/hdmi/hdmi/attr/debug
插拔 HDMI 可以看到以下信息：

```
[ 4539.933319] Playback_1--->All
[ 4539.936630] request active source
[ 4539.940337] CEC data:0x4f 0x85

[ 4540.893420] Playback_1--->TV
[ 4540.896650] image view on
[ 4540.899573] CEC data:0x40 0x04

[ 4541.053420] Playback_1--->TV
[ 4541.056654] image view on
[ 4541.059574] CEC data:0x40 0x04

[ 4541.213355] Playback_1--->TV
[ 4541.216568] image view on
[ 4541.219484] CEC data:0x40 0x04

[ 4541.373420] Playback_1--->TV
[ 4541.376651] image view on
[ 4541.379572] CEC data:0x40 0x04

[ 4541.533319] Playback_1--->TV
[ 4541.536533] image view on
[ 4541.539459] CEC data:0x40 0x04

[ 4541.783319] Playback_1--->All
[ 4541.786627] active source
[ 4541.789551] CEC data:0x4f 0x82 0x10 0x00

[ 4542.063340] Playback_1--->All
[ 4542.066663] active source
[ 4542.069585] CEC data:0x4f 0x82 0x10 0x00
```

安卓用户层送下来的 log 类似这样
CEC start transmitting, para: len:2 timeout:1000 sequence:0 tx_status:0 rx_status:0
msg[0]:0xbf msg[1]:0x36

3.1.6. Hdcp log 信息

echo 6 > /sys/class/hdmi/hdmi/attr/debug
插拔可以看到以下信息：

【hdcp1.4 log 信息】

失败信息：HDCP_ENGAGED/HDCP Failed

【hdcp2.2 log 信息】

成功信息：exception flag:109

失败信息：exception flag:99

3.1.7. HDMI Source 当前状态信息

【查看当前 HDMI 状态】

cat /sys/class/hdmi/hdmi/attr/hdmi_source

```
HPD:  1

RxSense:  1

PhyLock:  1

PhyPower:  1

TmdsMode:  HDMI

Scramble:  0

AvMute:  0

PixelRepetion:  0

BitDepth:  8

PixelFormat:  YUV444

Colorimetry:  ITU709

VideoFormat:  1080P60

AudioLayout:  0

AudioChannelCnt:  1

AudioSamplingFreq:  44100

AudioSampleSize:  16

AudioNvalue:  6272
```

【参数说明】

	参数	说明
	HPD	当前热插拔状态
	RxSense	Rxsense 用于判断 HDMI 是否属于连接状态
	PhyLock	HDMI phy 是否处于锁定状态
	PhyPower	Phy 是否上电

接口状态	TmdsMode	0: DVI　　1:HDMI
	Scramble	1:scremble enable，当 tmds clock > 340MHz 时，需要使能这个功能； 0:scremble disable，当 tmds clock <= 340MHz 时，需要关闭这个功能；
	AvMute	1:mute audio and video，向 hdmi sink 发 avmute，通知 hdmi sink 不要解析 hdmi 发来的数据； 0: clear avmute，结束向 hdmi sink 发 avmute
Video 参数	PixelRepetion	像素重复，当 pixel clock 小于 27MHz 时，需要进行像素重复
	BitDepth	色深，8bits/10bits/12bits
	PixelFormat	像素格式，RGB/YUV444/YUV422/YUV420
	Colorimetry	ITU601/ITU709/BT2020
	VideoFormat	480P60、576P60、720P50 等
audio 参数	AudioLayout	audio packet layout 0: 当 audio channel <= 2 时； 1: 当 audio channel > 2 时；
	AudioChannelCnt	audio 通道数
	AudioSamplingFreq	audio 采样率
	AudioSampleSize	audio 采样宽度
	AudioNvalue	audio 传输 N 值，N 与 CTS 值的意义与关系查看 HDMI Spec

3.1.8. HDMI Sink 的能力信息

【查看 HDMI Sink 的能力】

```
cat /sys/class/hdmi/hdmi/attr/hdmi_sink
```

```
Video Mode:  1080P60  1080I60  1080P24  1080P30  720P60  720x480P  720x480P  480I  480I  2160P24  2160PP25  2160P30  4096x
2160P24  4096x2160P302160P302160P252160P244096x2160P24

Only Support YUV420:

Also Support YUV420:  2160P50  2160P60  4096x2160P50  4096x2160P60

Pixel Format: RGB  YUV444  YUV422

Deep Color:  RGB444_30bit  YUV444_30bit  RGB444_36bit  YUV444_36bit  RGB444_48bit  YUV444_48bit  YUV420_30bit  YUV420_36bi
t  YUV420_48bit

3D Mode:  1080P60_FP  1080P60_SBS  1080P60_TAB  1080I60_FP  1080I60_SBS  1080I60_TAB  1080P24_FP  1080P24_SBS  1080P24_TAB
  720P60_FP  720P60_SBS  720P60_TAB  720x480P_SBS  720x480P_TAB

MaxTmdsCharRate:  120

Basic Audio Support:  YES

Audio Code:  LPCM
```

【参数说明】

Video Mode	HDMI Sink 支持的 video 模式
Only Support YUV420	仅支持 YUV420 格式的 Video 模式
Also Support YUV420	支持 YUV420 格式的 video 模式
Pixel Format	HDMI Sink 所支持像素格式
Deep Color	HDMI Sink 所支持 deep color 格式
3D Mode	HDMI Sink 所支持的 3D 模式
MaxTmdsCharRate	HDMI sink 最大支持的 Tmds char Rate，单位 MHz
Basic Audio Support	是否支持基本的 audio 格式
Audio Code	audio 编码格式

3.1.9. hdcp 状态信息

【查看 hdcp 状态】

cat /sys/class/hdmi/hdmi/attr/hdcp_dump

```
Tx do NOT use hdcp
Tx do NOT use hdcp 2.2
Disable HDCP

Lowlevel Part:
Tx do NOT use hdcp
Tx do NOT use hdcp 2.2
Disable HDCP
HDCP  hardware has NOT been Enable
```

【参数说明】

HDMI core 层	Tx do NOT use hdcp/Tx use hdcp	hdcp core 层是否支持 hdcp
	Tx do NOT use hdcp 2.2/Tx use hdcp 2.2	hdcp core 层是否支持 hdcp2.2
	Disable HDCP/Enable HDCP	hdcp core 层是否使能 HDCP
	HDMI MODE	HDMI/DVI 模式
	esm firmware addr: size:	esm firmware 物理地址与大小
	esm data addr: size:	esm data 物理地址与大小
HDMI 底层	Tx do NOT use hdcp/Tx use hdcp	hdcp 底层是否支持 hdcp
	Tx do NOT use hdcp 2.2/Tx use hdcp 2.2	hdcp 底层是否支持 hdcp2.2
	Disable HDCP/Enable HDCP	hdcp 底层使能
	HDCP hardware has been Enable/HDCP hardware has NOT been Enable	hdcp 硬件是否使能

	HDMI MODE	HDMI/DVI 模式
	hdcp 1.4 enable/hdcp 2.2 enable	hdcp1.4 使能或 hdcp2.2 使能

3.1.10. hdmi sink 端支持的 hdcp 类型信息

【命令】
Linux 方案命令
hexdump /sys/class/hdmi/hdmi/attr/hdcp_type
Android 方案命令
busybox hexdump /sys/class/hdmi/hdmi/attr/hdcp_type
返回值说明

返回值	说明
-1	DDC 读取 HDCP 类型失败，sink 不支持 HDCP
0	支持 HDCP1.4
1	支持 HDCP2.2

3.1.11. HDCP 身份认证结果信息

【命令】
Linux 方案命令
hexdump /sys/class/hdmi/hdmi/attr/hdcp_status
Android 方案命令
busybox hexdump /sys/class/hdmi/hdmi/attr/hdcp_status
返回值说明：

返回值	说明
0	HDCP 没有被使能
1	HDCP 正在认证
2	认证失败
3	HDCP 认证成功

3.1.12. Cec 状态信息

【查看 Cec 状态】
cat /sys/class/hdmi/hdmi/attr/cec_dump

```
cec thread is running
cec is active
cec is NOT in local standby model
Tx Current cec physical address:0x1000
Tx Current logical address:Playback_1
Rx Current logical address:TV
```

【参数说明】

cec thread is running/cec thread has been stopped	cec 内核线程运行情况
cec is active/cec is NOT active	cec 是否已经使能
cec is in local standby mode/cec is NOT in local standby mode	cec 是否使用 cec local standby，所谓的 local standby 是休眠时不发<Standby>信息。注意标准 CEC 不使用这个功能
Tx Current cec physical address	cec 主机当前物理地址
Tx Current cec logical address	cec 主机当前逻辑地址
Rx Current cec logical address	cec 从机当前逻辑地址

3.2. HDMI2.0 基础调试手段

3.2.1. 查看 EDID 原始数据

获取 EDID Raw Data

【查看 EDID 原始数据】

cat /sys/class/hdmi/hdmi/attr/edid > /mnt/edid.bin

将 edid.bin 用 edid 软件解析，可以看到 edid 情况，或者使用 cat hdm_sink，可以看到 hdmi edid 的简单的解析结果。

3.2.2. HDMI 寄存器 read 调试

【读 HDMI 寄存器】

echo [0x(address offset), 0x(count)] > /sys/class/hdmi/hdmi/attr/read

举例：从 0x100 地址开始读取 10 个寄存器：

echo 0x100, 0x10 > /sys/class/hdmi/hdmi/attr/read

```
start_reg=0x100  read_count=16
hdmi_addr_offset: 0x100 = 0x82
hdmi_addr_offset: 0x101 = 0x1b
hdmi_addr_offset: 0x102 = 0x0
hdmi_addr_offset: 0x103 = 0x0
hdmi_addr_offset: 0x104 = 0x3d
hdmi_addr_offset: 0x105 = 0x0
hdmi_addr_offset: 0x106 = 0x0
hdmi_addr_offset: 0x107 = 0x74
hdmi_addr_offset: 0x108 = 0x0
hdmi_addr_offset: 0x109 = 0x0
hdmi_addr_offset: 0x10a = 0x0
hdmi_addr_offset: 0x10b = 0x0
hdmi_addr_offset: 0x10c = 0x0
hdmi_addr_offset: 0x10d = 0x0
hdmi_addr_offset: 0x10e = 0x0
hdmi_addr_offset: 0x10f = 0x0
```

3.2.3. HDMI 寄存器 write 调试

【写寄存器】

echo [0x(address offset), 0x(value)] > /sys/class/hdmi/hdmi/attr/write

举例：

将地址为 0x100 的寄存器设置为 0x2：

echo 0x100,0x2 > /sys/class/hdmi/hdmi/attr/write

3.2.4. HDMI phy 寄存器 read 调试

【读 phy 寄存器】

cat /sys/class/hdmi/hdmi/attr/phy_read

可以看到 phy_read 的使用说明

```
[ 1247.783233] OPMODE_PLLCFG-0x16
[ 1247.786816] CKSYMTXCTRL-0x09
[ 1247.790088] PLLCURRCTRL-0x10
[ 1247.793425] VLEVCTRL-0x0E
[ 1247.796350] PLLGMPCTRL-0x15
[ 1247.799463] TXTERM-0x19
echo [0x(address offset), 0x(count)] > phy_read
```

3.2.5. HDMI phy 寄存器 write 调试

【写 phy 寄存器】
cat /sys/class/hdmi/hdmi/attr/phy_write
可以看到 phy_write 的使用说明

```
[ 1265.087827] OPMODE_PLLCFG-0x16
[ 1265.091262] CKSYMTXCTRL-0x09
[ 1265.094649] PLLCURRCTRL-0x10
[ 1265.097899] VLEVCTRL-0x0E
[ 1265.100830] PLLGMPCTRL-0x15
[ 1265.104055] TXTERM-0x19
echo [0x(address offset), 0x(value)] > phy_write
```

3.2.6. HDMI scdc 寄存器 read 调试

【读 scdc 寄存器】
cat /sys/class/hdmi/hdmi/attr/scdc_read
可以看到 scdc_read 的使用说明

```
echo [0x(address offset), 0x(count)] > scdc_read
```

3.2.7. HDMI scdc 寄存器 write 调试

【写 scdc 寄存器】
cat /sys/class/hdmi/hdmi/attr/scdc_write
可以看到 scdc_write 的使用说明

```
echo [0x(address offset), 0x(value)] > scdc_write
```

3.2.8. HDMI esm 寄存器 read 调试

【读 esm 寄存器】
cat /sys/class/hdmi/hdmi/attr/phi_read
可以看到 hpi_read 的使用说明

```
echo [0x(address offset), 0x(count)] > hpi_read
```

3.2.9. HDMI esm 寄存器 write 调试

【写 esm 寄存器】
cat /sys/class/hdmi/hdmi/attr/phi_write
可以看到 hpi_write 的使用说明

```
echo [0x(address offset), 0x(value)] > hpi_write
```

3.2.10. 发 av_mute

【发 av_mute】
echo 1 > /sys/class/hdmi/hdmi/attr/av_mute

3.2.11. 清 av_mute

【清 av_mute】
echo 0 > /sys/class/hdmi/hdmi/attr/av_mute

3.2.12. 设置 dvi/hdmi mode

【设置 dvi 模式】
echo 1 > /sys/class/hdmi/hdmi/attr/dvi_mode

【设置 hdmi 模式】
echo 0 > /sys/class/hdmi/hdmi/attr/dvi_mode

3.2.13. 打开/关闭 HDCP

【打开 HDCP】
echo 1 > /sys/class/hdmi/hdmi/attr/hdcp_enable

【关闭 HDCP】
echo 0 > /sys/class/hdmi/hdmi/attr/hdcp_enable

3.2.14. 打开/关闭 CEC

【打开 CEC】
echo 1 > /sys/class/hdmi/hdmi/attr/cec_enable

【关闭 CEC】
echo 0 > /sys/class/hdmi/hdmi/attr/cec_enable

3.2.15. 打开/关闭 phy

【打开 phy】
echo 1 > /sys/class/hdmi/hdmi/attr/phy_power

【关闭 phy】
echo 0 > /sys/class/hdmi/hdmi/attr/phy_power

3.2.16. 强制 HDMI 插入/拔出

【强制 HPD 插入】
echo 0x11 > /sys/class/hdmi/hdmi/attr/hpd_mask

【强制 HPD 拔出】
echo 0x10 > /sys/class/hdmi/hdmi/attr/hpd_mask

3.2.17. 屏蔽热插拔检测

echo 0x1000 > /sys/class/hdmi/hdmi/attr/hpd_mask

3.2.18. 查看热插拔状态

cat /sys/class/switch/hdmi/state （linux-3.4、linux-3.10）
0：表示拔出状态；
1：表示插入状态；

或者
cat /sys/class/extcon/hdmi/state （linux-4.4、linux-4.9）
HDMI=0：表示拔出状态；

HDMI=1：表示插入状态；

3.2.19. 运用 debugfs 切换分辨率，设置 hdmi 显示模式

【手动切换 video hdmi 分辨率】
mount -t debugfs none /mnt
cd /mnt/dispdbg

echo disp0 > name;echo switch1 > command;echo 4 10 1 1 0x4 0x104 0 0 0 8 > param;echo 1 > start
解释：如上面的命令的参数 4 就代表 HDMI 输出，10 代表 1080p@60 ，1 代表 YUV444， 1 代表 10bits，其余等参考以下解释：

命令解释：
即 echo [显示通道] > name;echo switch1 > command;echo [输出接口] [tv_mode] [format] [bits] [eotf] [色域空间标准] [dvi/hdmi 模式] [color range] [overscan/under scan] [aspect ratio]> param;echo 1 > start;

tv_mode 的定义：

```
DISP_TV_MOD_480I                = 0,
DISP_TV_MOD_576I                = 1,
DISP_TV_MOD_480P                = 2,
DISP_TV_MOD_576P                = 3,
DISP_TV_MOD_720P_50HZ           = 4,
DISP_TV_MOD_720P_60HZ           = 5,
DISP_TV_MOD_1080I_50HZ          = 6,
DISP_TV_MOD_1080I_60HZ          = 7,
DISP_TV_MOD_1080P_24HZ          = 8,
DISP_TV_MOD_1080P_50HZ          = 9,
DISP_TV_MOD_1080P_60HZ          = 0xa,
DISP_TV_MOD_1080P_24HZ_3D_FP    = 0x17,
DISP_TV_MOD_720P_50HZ_3D_FP     = 0x18,
DISP_TV_MOD_720P_60HZ_3D_FP     = 0x19,
DISP_TV_MOD_1080P_25HZ          = 0x1a,
DISP_TV_MOD_1080P_30HZ          = 0x1b,
DISP_TV_MOD_PAL                 = 0xb,
DISP_TV_MOD_PAL_SVIDEO          = 0xc,
DISP_TV_MOD_NTSC                = 0xe,
DISP_TV_MOD_NTSC_SVIDEO         = 0xf,
DISP_TV_MOD_PAL_M               = 0x11,
DISP_TV_MOD_PAL_M_SVIDEO        = 0x12,
DISP_TV_MOD_PAL_NC              = 0x14,
DISP_TV_MOD_PAL_NC_SVIDEO       = 0x15,
DISP_TV_MOD_3840_2160P_30HZ     = 0x1c,
DISP_TV_MOD_3840_2160P_25HZ     = 0x1d,
DISP_TV_MOD_3840_2160P_24HZ     = 0x1e,
DISP_TV_MOD_4096_2160P_24HZ     = 0x1f,
DISP_TV_MOD_4096_2160P_25HZ     = 0x20,
DISP_TV_MOD_4096_2160P_30HZ     = 0x21,
DISP_TV_MOD_3840_2160P_60HZ     = 0x22,
DISP_TV_MOD_4096_2160P_60HZ     = 0x23,
```

format 的定义:
RGB:0 YUV444: 1 YUV422: 2 YUV420:3

bits 的定义:
8bits: 0　　10bits: 1　　12bits: 3　　16bits: 4

eotf 的定义：

```
enum disp_eotf {
        DISP_EOTF_RESERVED = 0x000,
        DISP_EOTF_BT709 = 0x001,
        DISP_EOTF_UNDEF = 0x002,
        DISP_EOTF_GAMMA22 = 0x004, /* SDR */
        DISP_EOTF_GAMMA28 = 0x005,
        DISP_EOTF_BT601 = 0x006,
        DISP_EOTF_SMPTE240M = 0x007,
        DISP_EOTF_LINEAR = 0x008,
        DISP_EOTF_LOG100 = 0x009,
        DISP_EOTF_LOG100S10 = 0x00a,
        DISP_EOTF_IEC61966_2_4 = 0x00b,
        DISP_EOTF_BT1361 = 0x00c,
        DISP_EOTF_IEC61966_2_1 = 0X00d,
        DISP_EOTF_BT2020_0 = 0x00e,
        DISP_EOTF_BT2020_1 = 0x00f,
        DISP_EOTF_SMPTE2084 = 0x010, /* HDR10 */
        DISP_EOTF_SMPTE428_1 = 0x011,
        DISP_EOTF_ARIB_STD_B67 = 0x012, /* HLG */
```

色域空间标准定义:
BT601: 0x104　　BT709: 0x101　　BT2020: 0x107

3.3. HDMI2.0 基础配置检查

【linux-3.10】
（5）CONFIG_SWITCH=y
（6）CONFIG_DISP2_SUNXI=y
（7）CONFIG_HDMI2_DISP2_SUNXI=y
（8）CONFIG_DISP2_SUNXI_DEBUG=y

【linux-4.4 及以上版本】
（1）CONFIG_EXTCON=y
（2）CONFIG_DISP2_SUNXI=y
（3）CONFIG_HDMI2_DISP2_SUNXI=y
（4）CONFIG_DISP2_SUNXI_DEBUG=y

3.3.1. Dts 配置检查

【基本配置】

```
hdmi: hdmi@06000000 {
        compatible = "allwinner,sunxi-hdmi";
        reg = <0x0 0x06000000 0x0 0x100000>;
        interrupts = <GIC_SPI 64 IRQ_TYPE_NONE>;
        clocks = <&clk_hdmi>,<&clk_hdmi_slow>,<&clk_hdmi_hdcp>,<&clk_hdmi_cec>;
        pinctrl-names = "ddc_active","ddc_sleep","cec_active", "cec_sleep";
        pinctrl-0 = <&hdmi_ddc_pin_a>;
        pinctrl-1 = <&hdmi_ddc_pin_b>;
        pinctrl-2 = <&hdmi_cec_pin_a>;
        pinctrl-3 = <&hdmi_cec_pin_b>;
        status = "okay";
};
```

需要结合 IC 的 spec，检查上面的寄存器首地址是否对，时钟是否全，是否 ddc/cec 引脚配置

【时钟配置】

```
clk_hdmi: hdmi {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        assigned-clock-parents = <&clk_pll_video1>;
        clock-output-names = "hdmi";
};
clk_hdmi_slow: hdmi_slow {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        clock-output-names = "hdmi_slow";
};
clk_hdmi_cec: hdmi_cec {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        clock-output-names = "hdmi_cec";
};
```

```
clk_hdmi_hdcp: hdmi_hdcp {
        #clock-cells = <0>;
        compatible = "allwinner,sunxi-periph-clock";
        assigned-clock-parents = <&clk_pll_periph1>;
        clock-output-names = "hdmi_hdcp";
};
```

3.3.2. Sys_config.fex 配置检查

【hdmi 节点】

参数	说明
hdmi_used	1：使用 hdmi　0：不使用 hdmi
hdmi_hdcp_enable	1:使用 hdcp　　0:不使用 hdcp
hdmi_hdcp22_enable	1:使用 hdcp2.2　　0:不使用 hdcp2.2
hdmi_cec_support	1:支持 cec　　0:不支持 cec
hdmi_cts_compatibility	1：驱动兼容 hdmi cts 测试，符合 HDMI CTS 测试的标准； 0：驱动部分配置根据实际情况进行配置，部分配置不采用 HDMI CTS

	测试标准

3.4. HDMI2.0 常见问题排查

3.4.1. 显示问题---插入 HDMI 无显示

步骤 1：

查看是否加载了 HDMI 模块：cd /sys/class/hdmi/hdmi/attr，查看是否有这个目录。如果没有这个目录，说明 HDMI 模块没有被加载；

步骤 1：

cat /sys/class/disp/disp/attr/sys，从显示框架上查看显示是否是切换到 hdmi 输出

```
screen 0:
de_rate 432000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank dir
ect_show[false]
dmabuf: cache[0] cache_max[0] umap skip[0] overflow[0]
       hdmi output mode(10)     fps:60.6        1920x1080
        err:2    skip:4342       irq:495972      vsync:0 vsync_skip:0
   BUF    enable ch[2] lyr[0] z[0] prem[N] a[globl 255] fmt[  0] fb[ 240, 320; 240, 320; 240, 320] crop[
  0,    0, 240, 320] frame[   0,    0, 240, 320] addr[70080000,       0,       0] flags[0x       0] trd[0,0
]
```

步骤 3：

cat /sys/class/hdmi/hdmi/attr/hdmi_sink 通过此判断当前 HDMI 输出的分辨率，sink 端是否支持

步骤 4：

观察一下电视，看电视有没有识别 HDMI 信号，如果有 HDMI 信号，则可以通过 sunxi_dump 设置寄存器的方式，设置 tcon 内部输出图像到 HDMI，在看看输出的图像是否正常。如果图像正常，则说明 HDMI 端配置没有问题，应该检查 DE 端的设置。否则，则应该怀疑 tcon 和 hdmi 的问题。

步骤 5：

假如确认加载了 HDMI 模块，cat /sys/class/hdmi/hdmi/attr/hdmi_source
查看 hdmi 状态，参考“查看 HDMI sorce 状态”章节。

步骤 6：

如果以上步骤查不出问题，应该联系负责 HDMI 驱动的工程师分析，提供以上步骤的信息

3.4.2. 显示问题---HDMI 闪屏

步骤 1：

mount -t debugfs none /mnt;

cat /mnt/clk/clk_summary

查看 PLL_VIDEO、TCON_TV、HDMI 的时钟是否与分辨率相配

步骤 2：

如果步骤 1 查不出问题，应该联系负责 HDMI 驱动的工程师分析 HDMI 的 timing

步骤 3：

cat /sys/class/hdmi/hdmi/attr/sys

查看 fps 信息是否准确

步骤 4：

如果以上步骤查不出问题，应该联系负责 HDMI 驱动的工程师分析，提供以上步骤的信息

3.4.3. 显示问题---HDMI 显示颜色不对

步骤 1：

cat /sys/class/disp/disp/attr/sys

```
screen 0:
de_rate 432000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank dir
ect_show[false]
dmabuf: cache[0] cache max[0] umap skip[0] overflow[0]
        hdmi output mode(10)    fps:60.6        1920x1080
        err:2   skip:4342       irq:495972      vsync:0 vsync_skip:0
   BUF    enable ch[2] lyr[0] z[0] prem[N] a[globl 255] fmt[  0] fb[ 240, 320; 240, 320; 240, 320] crop[
  0,   0, 240, 320] frame[   0,   0, 240, 320] addr[70080000,       0,       0] flags[0x       0] trd[0,0
]
```

查看 cs 参数是否正确。cs，即 color space，这里被定义为颜色标准，其值的定义如下：

GBR： 0x100

BT709： 0x101

BTFCC:： 0x102

BT470BG： 0x103

BT601： 0x104

SMPTE240M： 0x105

YCGCO：0x106

BT2020NC： 0x107

BT2020C： 0x108

步骤 2：

如果步骤 1 查不出问题，请联系全志原厂 HDMI 驱动的工程师分析。

3.4.4. Audio 问题---audio 没有声音

步骤 1：

请 Audio 驱动工程师，首先查看确认 Audio 模块是否有声音输出；

步骤 2：

cat /sys/class/hdmi/hdmi/attr/hdmi_source 查看 hdmi 模块的 audio 状态；

或者 echo 3 > /sys/class/hdmi/hdmi/attr/debug，然后插拔一下 HDMI 线，观察 HDMI audio log。

看 hdmi 模块的 audio 的配置有没有问题。

步骤 3：

cat /sys/class/hdmi/hdmi/attr/hdmi_source 查看 HDMI 是不是工作到了 DVI 模式；

步骤 4：

如果以上步骤查不出问题，应该联系负责 HDMI 驱动的工程师分析。

3.4.5. Audio 问题---audio 卡顿

步骤 1：

请 Audio 驱动工程师，首先查看确认 Audio 模块是否声音输出卡顿；

步骤 2：

cat /sys/class/hdmi/hdmi/attr/hdmi_source 查看 hdmi 模块的 audio 状态；

或者 echo 3 > /sys/class/hdmi/hdmi/attr/debug，然后插拔一下 HDMI 线，观察 HDMI audio log。

看 hdmi 模块的 audio 的配置有没有问题。

步骤 3：

联系 HDMI 驱动工程师用协议分析仪查看显示的 timing 参数，看显示是否影响到了 HDMI audio，特别是 vsync、hsync 参数；

步骤 4：

如果以上步骤查不出问题，请联系全志原厂 HDMI 驱动的工程师分析，提供以上调试信息。

3.4.6. CEC 问题---CEC log 报错

```
[12158.953296] HDMI cable is connected
[12159.473387] CEC Send error: IH Status: 0x11
[12159.623287] CEC Send error: IH Status: 0x11
[12161.203447] CEC Send error: IH Status: 0x11
```

这个发送失败的 log 会经常出现，因为 cec 在发送的同时，可能也会收到从机发来的消息，因此并不是每个消息每次发送都一定会成功。但每条消息一旦发送失败之后，都会重发 5 次，如果这 5 次都没有成功，才是真正的发送失败，会出现以下 log：

```
CEC send error, error code:-1
```

3.4.7. CEC 问题---CEC direct address 消息无法发送

步骤 1：

cat /sys/class/hdmi/hdmi/attr/cec_ump 查看 cec 的工作状态，确认 cec 是否已经处于正常工作状态；

步骤 2：

echo 0x7d05,2 > /sys/class/hdmi/hdmi/attr/read

有可能已经发送出消息了，但是发出去的消息只是 head 部分，data 部分没有发送出去。出现这种情况下的原因为：

有可能是 CEC 主机的从机地址被设置成了从机的逻辑地址，注意 CEC 主机的逻辑地址可以设置成多个。可以通过读寄存器来确认：echo 0x7d05,2 > /sys/class/hdmi/hdmi/attr/read；

步骤 3：

检查发送的 CEC 消息的 head 消息是否正确。

步骤 4：

如果以上步骤找不到原因，应联系 HDMI 驱动工程师分析。

3.4.8. CEC 问题---无法接收到 CEC 直接地址信息

步骤 1：

cat /sys/class/hdmi/hdmi/attr/cec_ump 查看 cec 的工作状态，确认 cec 是否已经处于正常工作状态；

步骤 2：

echo 0x7d05,2 > /sys/class/hdmi/hdmi/attr/read，确认 CEC 主机逻辑地址有没有设置正确；

步骤 3：

查看 CEC 主机是否发送了<Report Physical Address>和<Active Source>将逻辑地址与物理地址绑定；

步骤 4：

如果以上步骤找不到原因，请联系全志原厂 HDMI 驱动工程师分析。

3.4.9. CEC 问题---cec config err, can NOT send cec

cec 的信息包发送不下去，出现这个问题，首先看 cec 的 clk 设置有没有被关闭，时钟被关闭出现这个问题概率很大。

3.4.10. DDC 问题---插拔 HDMI 时 edid 报错

步骤 1：

检查电视是否已经打开，如果只是将 HDMI 线连接到电视，但电视没有上电打开的话，这种情况下 EDID 肯定是读取失败的；

步骤 2：

根据 spec，检查 ddc 的时钟、引脚配置是否正确；

步骤 3：

cat /sys/class/hdmi/hdmi/attr/hdmi_sink

查看是否真正读到 edid，如果是真正读到了 edid，则说明有可能是 ddc 频率问题，在读 edid 过程中有读取失败的情况，但却在重读中成功了。这时候应该联系 HDMI 驱动工程师分析；

步骤 4：

如果以上步骤找不到原因，联系全志原厂 HDMI 驱动工程师分析。

3.4.11. HDCP 问题---HDCP1.4 log 报错

报错的 log：HDCP failed

步骤 1：

尝试多个电视，因为有可能报错的电视可能不支持 HDCP1.4；

步骤 2：

cat /sys/class/hdmi/hdmi/attr/hdcp_dump，查看 Tx use hdcp 项，看驱动是否支持 HDCP；

步骤 3：

有可能是 HDCP 的 key sram 中的 key 不对，这时应该联系 uboot 相关的工程师确认 HDCP key 问题；

步骤 4

如果以上步骤找不到原因，联系全志原厂 HDMI 驱动工程师分析。

3.4.12. HDCP 问题---使能 HDCP 后屏幕无显示

3.4.13. HDCP 问题---HDCP2.2 log 报错

报错 log：exception flag:99

步骤 1：

cat /sys/class/hdmi/hdmi/attr/hdcp_dump，查看 esm 的 firmware、data 区域的物理地址和内存大小是否有问题，查看 Tx use hdcp2.2 参数，查看驱动是否支持 hdcp 2.2；

步骤 2：

打开 hdcp 的 log：echo 6 > /sys/class/hdmi/hdmi/attr/debug，然后插拔一下，观察打印的 HDCP log，然后进行以下步骤；

步骤 3：

查看 esm 是否启动成功：

如果有 log：[esm-error]:esm boots fail!，则说明 esm 启动失败，说明 esm firmware 有问题，应坚持板子烧录的 PKF 是否正确，坚持 esm firmware 是否正确；

如果有 log：esm boots successfully，说明 esm 启动成功，但却认证失败了，应确认 esm firmware 的生成过程有没有问题，连续 HDMI 驱动工程师确认 esm firmware 的配置，正确配置 esm firmware 后，再重新生成新的 esm firmware。

步骤 4：

如果以上步骤找不到原因，联系全志原厂 HDMI 驱动工程师分析。

4. LCD 调试说明

4.1. LCD 接口调试方法

以下大部分信息都需要 debugfs，如果没有挂载可以通过下面命令行挂载：

```
mount -t debugfs none /sys/kernel/debug
```

4.1.1. LCD 显示状态调试

cat /sys/class/disp/disp/attr/sys

screen 0:
de_rate 297000000 hz, **ref_fps:60**
mgr0: 1280x800 fmt[rgb] cs[0x204] range[full] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank direct_show[false]
lcd output backlight(50)**fps:60.9** 1280x 800
err:0skip:31 irq:1942 vsync:0 vsync_skip:0
BUF enable ch[1] lyr[0] z[0] prem[N] a[globl 255] fmt[8] fb[1280, 800;1280, 800;1280, 800] crop[0, 0,1280, 800] frame[0, 0,1280, 800] addr[0, 0, 0] flags[0x 0] trd[0,0]

- **lcd output**
  表示当前显示接口是 LCD 输出。如果不是，要么应用层要主动切换，要么你要在 sys_config.fex 设置好 screenX_output_type 属性。
- **1280x 800**
  表示当前 LCD 的的分辨率，需要和 sys_config.fex 中 lcd0 的设置一样，
- **ref_fps:60**
  表示当前 LCD 的显示帧率理论值，该值是根据 sys_config.fex 的 lcd0 填的时序算出来的**理论值；**
- **fps:60.9**
  表示当前 LCD 的显示帧率实时统计值，正常来说应该是在 60(期望的 fps)附近，如果差太多则不正常，重新检查屏时序，和在屏驱动的初始化序列是否有被调用到。
- irq:1942
  表示 LCD 的显示中断的次数，正常来说是一秒 60（期望的 fps）次，重复 cat sys，如果无变化，则异常。
- BUF
  表示图层信息，一行 BUF 表示一个图层，如果一个 BUF 都没有出现，那么将是黑屏，不过和屏驱动本身关系就不大了，应该查看应用层&框架层。

4.1.2. LCD 电源状态调试

对于 axp 方案，如果需要检查 LCD 的供电状态，可以通过以下命令检查 LCD 电源是否有 enable，同并结合实际万用表量取电压值进行比较，以帮助排查初级硬件问题。

cat /sys/class/regulator/dump

```
pmu1736_ldoio2 : disabled    0    700000       supply_name:
pmu1736_ldoio1 : disabled    0    700000       supply_name:
pmu1736_dc1sw : enabled    1    3300000       supply_name:    vcc-lcd
pmu1736_cpus : enabled    0    900000       supply_name:
pmu1736_cldo4 : disabled    0    700000       supply_name:
pmu1736_cldo3 : disabled    0    700000       supply_name:
pmu1736_cldo2 : enabled    1    3300000       supply_name:    vcc-pf
pmu1736_cldo1 : disabled    0    700000       supply_name:
pmu1736_bldo5 : enabled    2    1800000       supply_name:    vcc-cpvin    vcc-pc
pmu1736_bldo4 : disabled    0    700000       supply_name:
pmu1736_bldo3 : disabled    0    700000       supply_name:
pmu1736_bldo2 : disabled    0    700000       supply_name:
pmu1736_bldo1 : disabled    0    700000       supply_name:
pmu1736_aldo5 : enabled    0    2500000       supply_name:
pmu1736_aldo4 : enabled    0    3300000       supply_name:
pmu1736_aldo3 : enabled    1    1800000       supply_name:    avcc
pmu1736_aldo2 : enabled    0    1800000       supply_name:
pmu1736_aldo1 : disabled    0    700000       supply_name:
pmu1736_rtc : enabled    0    1800000       supply_name:
pmu1736_dcdc6 : disabled    0    500000       supply_name:
pmu1736_dcdc5 : enabled    0    1480000       supply_name:
pmu1736_dcdc4 : enabled    1    900000       supply_name:    vdd-sys
pmu1736_dcdc3 : enabled    0    900000       supply_name:
pmu1736_dcdc2 : enabled    0    1160000       supply_name:
pmu1736_dcdc1 : enabled    4    3300000       supply_name:    vcc-emmc    vcc-io    vcc-io    vcc-io
```

4.1.3. LCD PWM 状态调试

LCD 的 PWM 接口用于提供背光电源，调试方法如下

cat /sys/kernel/debug/pwm

打印如下信息：

```
platform/7020c00.s_pwm, 1 PWM device
 pwm-0     ((null)                    ): period: 0 ns duty: 0 ns polarity: normal

platform/300a000.pwm, 2 PWM devices
 pwm-0     (lcd                       ): requested enabled period: 20000 ns duty: 3984 ns polarity: normal
 pwm-1     ((null)                    ): period: 0 ns duty: 0 ns polarity: normal
```

上面输出的信息表示：通道 0 的 pwm 被请求和使能了，使能者是”LCD”，后面还有 pwm 的周期频率信息。

注意：方案配置中存在两种类型的 pwm，一种为是 CPUS(supper standby)的 pwm 对应上面的 **platform/7020c00.s_pwm**，这种 pwm 在软件上通常对应通道 8 的 pwm 上。一种是 CPUX(主控)的 pwm，对应上面的 **platform/300a000.pwm**，软件上一般是 pwm0 到 pwm7。

4.1.4. LCD PIN 引脚状态调试

LCD 屏一般需要配置部分管脚的复用功能，一般来说如果是 RGB 或者 LVDS 接口需要配置对应的数据脚，时钟脚和同步信号脚。

此外，有些屏会用到一些管脚用于复位、电源使能、背光使能、甚至是用 gpio 模拟 SPI，模拟 I2C 等。

这个时候想知道自己设置的管脚有没有设置成功可以参考下面的打印。

```
cat /sys/kernel/debug/pinctrl/pio/pinmux-pins
```

打印以下信息：

```
pin 227 (PH3): twi1 (GPIO UNCLAIMED) function io_disabled group PH3
pin 228 (PH4): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 229 (PH5): (MUX UNCLAIMED) pio:229
pin 230 (PH6): (MUX UNCLAIMED) pio:230
pin 231 (PH7): (MUX UNCLAIMED) pio:231
pin 106 (PD10): lcd0 (GPIO UNCLAIMED) function lvds1 group PD10
```

上面的信息表示：

如果一行里面同时出现(MUX UNCLAIMED) (GPIO UNCLAIMED)，说明该管脚处于初始状态没有声明也没有复用为某个功能。

如果(MUX UNCLAIMED) 被换成其它字符，比如 twi1 或者 lvds 说明该管脚被声明为改字符串的功能。
如果(GPIO UNCLAIMED) 被换成 pio:xxx，说明该管脚被复用为普通 GPIO，而且处于使能状态。

PH5，PH6 这些 IO 被申请为普通 GPIO 功能，而 PD10 被声明而且复用为 lvds 管脚功能。

PH3 虽然被声明为 twi1，但是后面 function io_disabled，表示当前为 io disable 状态。

4.1.5. LCD CLK 时钟配置调试

抓取与显示 Tcon 相关的 CLK 时钟信息

cat /sys/kernel/debug/clk/clk_summary | grep tcon

抓取与显示 pll_video 相关的 CLK 时钟信息
cat /sys/kernel/debug/clk/clk_summary | grep pll_video

抓取与显示 mipi 抓取与显示
cat /sys/kernel/debug/clk/clk_summary | grep mipi

4.1.6. LCD 接口 colorbar 调试

显示是一整条链路，中间任何一个环节出错，最终的表现都是显示异常，图像显示异常几个可能原因，

一、图像本身异常。

二、图像经过 DE（Display Engine）后异常。

三、图像经过接口模块后异常。这是我们关注的点。

有一个简单的方法可以初步判断，接口模块（tcon 和 dsi 等）可以自己输出内置的一些 patten，比如说彩条，灰阶图，棋盘图等。当接口输出这些内置 patten 的时候，如果这时候显示就异常，这说明了：

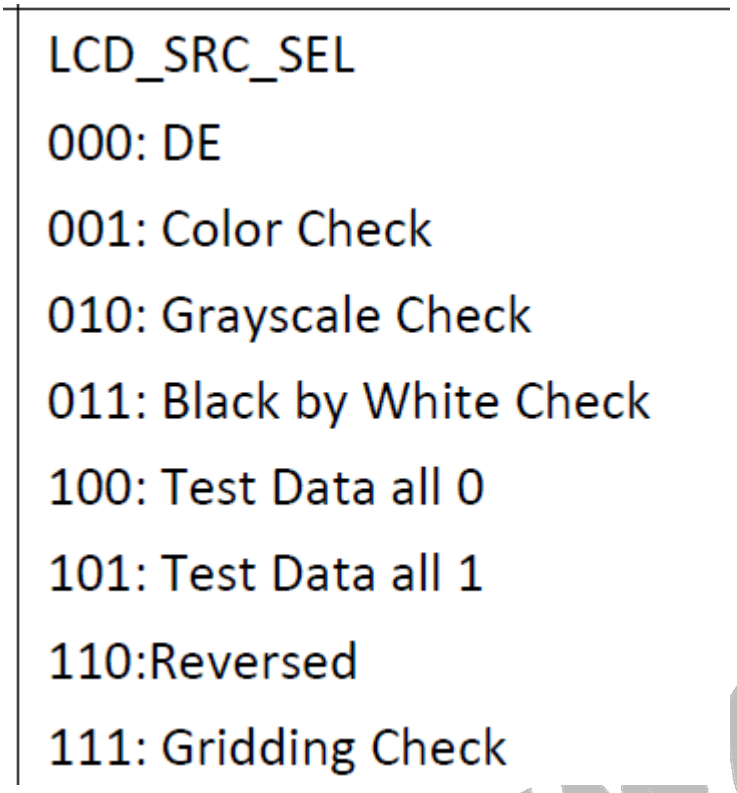
1. LCD 的驱动或者配置有问题
2. LCD 屏由于外部环境导致显示异常

显示自带 patten 的方式：

在 linux-4.9 上，disp 的 sysfs 中有一个 attr 可以直接操作显示:

echo X > /sys/class/disp/disp/attr/colorbar

上面的操作是显示 colorbar，其中的 **X** 可以是 0 到 8，0 到 7 对应的含义如下图所示：



8 表示让 DE 绘制 colorbar。

如果有多个显示，想让第二个显示显示 colorbar 的话，那么先
echo 1 > /sys/class/disp/disp/attr/disp
然后再执行上上面操作。

如果没有这个 attr 的话，可以直接操作寄存器，也就是操作 tcon 寄存器的 040 偏移的最低 3 位。
在 linux 下，cd /sys/class/sunxi_dump 然后
echo 0x06511040 > dump;cat dump
这样会打印当前 tcon 的 040 偏移寄存器的值，然后在上面值的基础上修改最低 3 位为上图的值即可，修改方式，示例：
echo 0x06511040 0x800001f1 > write
注意 tcon 的基地址不一定是 0x06511000，不同平台不一样，请参考 SOC 文档获取 tcon 的基地址。

4.1.7. DE 截屏

显示出现异常的时候，有可能是下面三个原因：
1. SOC 端屏接口模块+LCD 屏出现了问题。
2. 图像经过 SOC 端图像合成模块(DE）处理后出现了问题。
3. 图像源本身就有问题。

本节介绍，确认图像源本身没问题的前提下，如何进一步确认是否经过 DE 处理之后图像是否有问题。

语法：
echo 屏幕索引 > /sys/class/disp/disp/attr/disp
echo 路径/bmp 文件名 > /sys/class/disp/disp/attr/capture_dump
第一个 echo 的作用是确定捕捉哪个显示的，“屏幕索引”可选取值是 0 或者 1
第二个 echo 作用是生成截屏 bmp 文件，确保“路径”是可写的，有剩余空间的。

比如：
echo 0 > /sys/class/disp/disp/attr/disp
echo /data/xx.bmp > /sys/class/disp/disp/attr/capture_dump

这样就会在/data 目录下生成 screen 0 的 bmp 截图，文件名是 xx.bmp。

注意：这个功能只有 linux-4.9 以及后续的内核才支持这个功能。

4.2. LCD 常见问题

4.2.1. 黑屏（无背光）

背光一般是由 pwm 来控制的，没有背光的话有多种原因，请按下下面步骤依次检查：

（一）屏驱动没有加载成功。屏驱动编写有问题，没有加载，自然不会执行屏背光开启的操作，你可以通过查看显示信息一小节，确定屏驱动是否加载成功了。

（二）PWM 通道没有使能。检查原理图，确定所用 pwm 通道编号，查看 pwm 信息一节确认对应 pwm 有没有使能。软件确定 sys_config.fex 中的[lcd0]中 lcd_pwm_开头的属性配置正确。

（三）背光使能脚没有使能。不少设计中都用一根 gpio 来控制背光的使能，当这根脚处于无效状态时，背光自然无法亮，根据屏手册得知背光使能脚有效状态（高低电平，以及电压值），然后用万用表确认下。软件上确认 sys_config.fex 中的[lcd0]的 lcd_bl_en 的定义是准确的。

（四）背光值设置太低。不少屏，但 pwm 的占空比为接近 0%的时候就会完全黑屏。

（五）原理图或者 PCB 有错导致背光电压不够。屏手册里面会规定背光所需要的电压范围，直接用万用表确认下是否符合屏的要求。

4.2.2. 黑屏（有背光）

黑屏但是有背光，可能有多种原因导致，请依次按以下步骤检查：

（一）没送图层。如果应用没有送任何图层那么表现的现象就是黑屏，通过查看显示信息一小节可以确定有没有送图层。如果确定没有图层，可以通过查看接口自带 colorbar，确认屏能否正常显示。

（二）SOC 端的显示接口模块没有供电。SOC 端模块没有供电自然无法传输视频信号到屏上。一般 SOC 端模块供电的 axp 名字叫做 vcc-lcd，vcc-dsi，vcc33-lcd，vcc18-dsi 等。

（三）复位脚没有复位。如果有复位脚，请确保硬件连接正确，确保复位脚的复位操作有放到屏驱动中。

（四）sys_config.fex 中[lcd0]有严重错误。第一个是 lcd 的 timing 太离谱，请严格按照屏手册中的提示来写！手册没写就找屏厂，这个时序只有屏厂清楚。第二个就是，接口类型搞错，比如接的 DSI 屏，配置却写成 LVDS 的。

（五）屏的初始化命令不对。包括各个步骤先后顺序，延时等，这个时候请找屏厂确认初始化命令。

4.2.3. 闪屏

[问题表现]
1. 整个屏幕闪，忽明忽暗的感觉，一般为背光或智能背光的问题。
2. 屏的部分区域闪，有的时候在开屏后闪，一段时间后消失

[分析/试验]

1. 检查智能背光是否开着，如果开着，关掉是否还会出现，澄清智能背光的问题

2. 如果不是智能背光问题，用万用表测量下 vled+/-的电压是否稳定，如果稳定说明不是背光原因；如果不稳定，跳下一步

3. 查一下 pwm pin 的电压，看是否稳定，如果稳定，则要查一下背光电路或背光 IC 是否存在问题，或者是否有其他电路会影响到 vled+/-

如果不稳定，先检查下软件原因，pwm 的配置是否一直变化:
如果有变化则为软件原因;
如果没变化，再查硬件原因，检查 pwm 受到哪些因素的干扰。

3. 如果是智能背光原因，则如果检查智能背光调节问题。智能背光会对背光及显示内容进行调节，可以分开检查，对背光的调节是否正确合理/对显示内容调节是否正确合理。

4. 针对表现 2，很大的可能是屏的上电顺序不正确 或者 初始化未正确引起。

4.2.4. 白屏

不少屏初始化失败的，但是电源有上（至少背光有上），那么呈现的现象是白屏。

（一）检查所有屏相关相关电源。通过查看电源信息一小节确认需要开 axp 电源有开启。
（二）检测复位脚电压。如果有的话。
（三）检查初始化代码以及调整 LCD 的 timing（lcd_dclk_freq，lcd_ht，lcd_vt，lcd_hbp，lcd_vbp，lcd_hspw，和 lcd_vspw），咨询屏厂。

4.2.5. 雪花屏

一部分屏再初始化成功之后，由于 SOC 端没有发送数据，会呈现雪花屏的状态，一般是那种需要 te 脚（tear effect）同步的屏。

这种情况需要，

（一）检查 te 脚有没有连到 SOC 端的 lcd_vsync 脚

（二）lcd_vsync 所对应的脚有没有被设置成 vsync 功能

（三）检查 te 脚的电压是否够大，是否足以触发中断

（四）雪花屏的另外一个原因，就是图像数据被串改，这个时候和屏本身就没关系了。可以通过截屏一小节来确认图像本身是否正常，通过查看接口自带 colorbar 一小节来确认接口信号是否正常。

4.2.6. 图像抖动

图像抖动可能是屏的问题也可能是图像数据本身的问题。

（一）这时候可以通过查看接口自带 colorbar，确认图像本身是否正常

（二）通过查看接口自带 colorbar 一小节来确认接口信号是否正常。

（三）如果是屏的问题，那么可能原因就是时钟管脚以及其它数据信号脚的驱动能力不足。这个时候需要修改 sys_config.fex 中[lcd0]的管脚部分的设置，详情查看《sunxi display2 LCD 调试指南》。

4.2.7. 条形波纹

有些 LCD 屏的像素格式是 18bit 色深（RGB666）或 16bit 色深（RGB565），建议打开 FRM 功能，通过 dither 的方式弥补色深，使显示达到 24bit 色深（RGB888）的效果。如下图所示，上图是色深为 RGB66 的 LCD 屏显示，下图是打开 dither 后的显示，打开 dither 后色彩渐变的地方过度平滑。

设置[lcd0]的 lcd_frm 属性可以改善这种现象。

4.2.8. 背光太亮或者太暗

背光是模拟信号，有些电路无法保证线性，所以出现某些背光值过亮或者过暗的情况。
设置[lcd0]的 Lcd_bl_n_percent 属性可以改善这种现象。

5. CVBS OUT 调试说明

5.1. CVBS OUT 调试信息

5.1.1. 查看热插拔信息

cat /sys/class/extcon/cvbs/state

CVBS=0 表示拔出
CVBS=1 表示插入

如果内核版本是 4.4 之前的
cat /sys/class/switch/cvbs/state

5.1.2. 读写 AC200 寄存器

cd /sys/class/i2c-adapter/i2c-3/3-0010/acx00_debug

用法提示，执行下面两个获取用法提示。
cat dump
cat write

通过读写至少可以说明两个问题：
1. I2C 能否正常通信
2. AC200 是否运行中（不管配置是否正确）

5.2. CVBS OUT 常见问题

5.2.1. 电视没有检测没有信号

依次检查下面各项：
1. 电视是否有切换到 tv 模式
2. 连线是否有错误。一般来说有三根线，黄色那根是视频，电视那端插孔是可能是黄色也可能是绿色（请根据背后印刷提示选择）
3. 内核有没有使能 tv out 模块。
4. 查看/sys/class/disp/disp/attr/sys 信息。如果 tv out 显示使能，将会有下面类似的信息，如果没有则是软件没有开，与驱动无关。

screen 1:

de_rate 696000000 hz, ref_fps:100

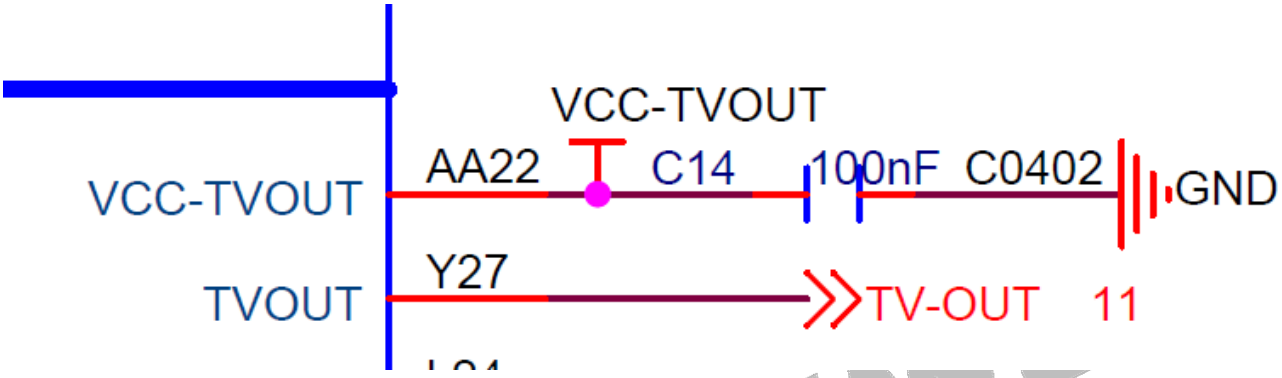
mgr1: 720x576 fmt[yuv444] cs[0x204] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank direct_show[false]

dmabuf: cache[0] cache max[0] umap skip[0] overflow[0]

tv output mode(11) fps:0.0 720x 576

err:2skip:2 irq:89 vsync:0 vsync_skip:0

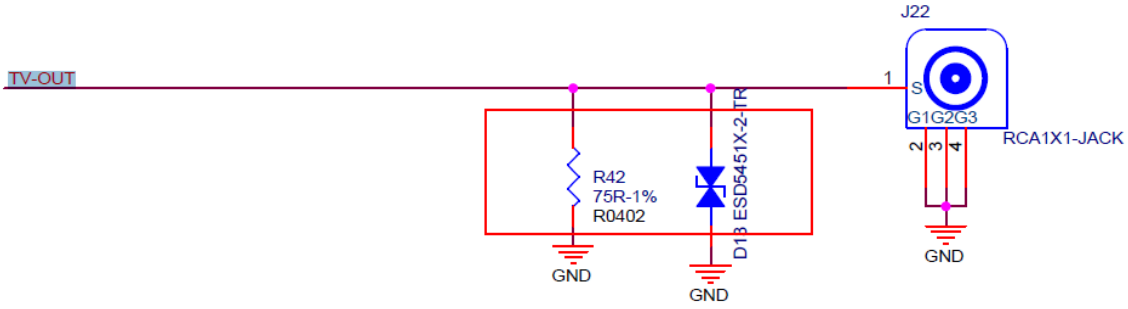
4. 确认板子的 tvout 电源有开启。Soc 端通常有一路电源供给 tvout 模块，原理图上一般就在 TV-OUT 管脚旁边。



5.2.2. 软件没有检测到热插拔信号或者不稳定

依次检查下面各项：

1. 检查上面 1.6.1 小节各项。值得注意的是，即使没有使能 tv out 显示，只要有加载 tv 模块也是可以检测到热插拔信号。
2. 查看/sys/class/extcon/（linux-4.4 内核及其以上）或者/sys/class/switch 下面是否存在 cvbs 节点。如果不存在这个节点，有几种可能，第一，内核没有使能 extcon 或者 switch；第二加载 tv 驱动的时候失败了。
3. Tve 模块需要进行 DAC 校准，这个 DAC 校准如果不准确可能会影响到热插拔检查的准确性。一般来说这个 DAC 校准是芯片出厂后就做好的。
4. TV-OUT 管脚硬件问题。CVBS out 的热插拔检测是根据 TV-OUT 管脚电压的变化来检测的。如下图红框所示的原件，它的大小直接影响热插拔检测。



5.2.3. AC200 模块无法检测到信号

全志芯片有两个模块支持 CVBS out。第一个叫做 TVE，第二个叫做 AC200，前者是内嵌 IP，后者是独立芯片。

AC200 问题排查如下：

1. I2C 能否正常通信。AC200 是通过 I2C 进行配置的，I2C 无法工作，AC200 自然无法工作。如果在 log 里面见到 I2C 相关错误请找 I2C 的负责的人查看。I2C 的通道，从地址一般不会错误，要特别注意下 I2C 的供电。
2. PWM 能否正常工作。PWM 这里作为 AC200 的晶振，PWM 无法工作，AC200 自然无法工作。PWM 通道和占空比配置一般不会错误，要特别注意下是否有使能 PWM，PWM 管脚的供电（比如 vcc-pd）。
3. AC200 本身也有一个供电，一般叫做 vcc-tvout 或 vcc-tv。

6. CVBS IN 调试说明

6.1. CVBS IN 调试信息

6.1.1. 查看驱动加载情况

ls /dev/video*

一般来说会有 5 个以上，其中 video4，video5，video6 和 video7 是四路 tvd 各自的 v4l2 节点，video8 是一个用以上四路拼接成一路的特殊设备。其它的可能是 CSI 的。

如果没有节点，那么检查：
1. 内核配置是否有使能。
2. Sys_config.fex 中是否有配置 tvd

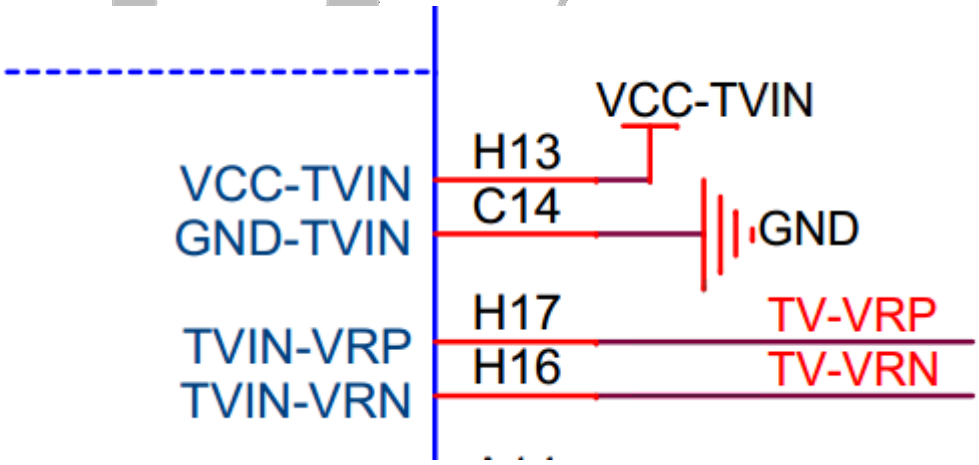
6.2. CVBS OUT 常见问题

6.2.1. 无法采集到信号

依次检查下面各项：

（一）摄像头的电源，摄像头没电没输出数据自然采集不到视频。

（二）CVBS IN 模块的电源，如下图的 VCC-TVIN。



6.2.2. 采集的信号撕裂画面乱跳

采集信号过程中有一个信号稳定的过程。有的应用程序没有等信号锁住就立刻采集，那么这种情况很可能造成信号异常或者采集不到信号，这个时候应用层应该在确保调用 v4l2 接口的 VIDIOC_G_FMT 和 VIDIOC_S_FMT 返回成功之后再进行接下去的操作。

6.2.3. 提高视频显示效果

1. 整条视频通路按照 nv16 或者 nv61 的格式来进行处理。包括 tvin，DI，G2D，GPU，DE 模块。
2. 开启 3D 滤波。也就是 sys_config.fex 中[tvd]的 fliter_used 属性。

7. De-Interlace 调试说明

7.1. 设备节点

dev/deinterlace 设备节点

7.2. 传递参数信息

1.echo 0xff > /sys/module/deinterlace/parameters/debug_mask 打开调试信息
2.echo 0x0 > /sys/module/deinterlace/parameters/debug_mask 关闭调试信息
如果打印太多，可以先调高打印等级 echo 0 > /proc/sys/kernel/printk,然后 dmesg 查看信息

sunxi_di_ioctl: enter!!
di_process: input_fb.fd = 29 ----------input buffer fd
di_process: pre_fb.fd = 25 ----------pre buffer fd
di_process: next_fb.fd = 32 ----------next buffer fd
di_process: output_fb.fd = 35 ----------output buffer fd
di_process: input_fb.addr[0] = 0x63d53000 ----------输入映射地址
di_process: input_fb.addr[1] = 0x63db3000
di_process: input_fb.addr[2] = 0xab85c3d8f70a91b8
di_process: input_fb.size.width = 720 ----------输入 buf 的宽度
di_process: input_fb.size.height = 480 ----------输入 buf 的高度
di_process: input_fb.format = 1 ----------输入 buf 的 format
di_process: pre_fb.addr[0] = 0x63de3000 ----------同上
di_process: pre_fb.addr[1] = 0x63e43000
di_process: pre_fb.addr[2] = 0xab5c948cab783810
di_process: pre_fb.size.width = 720
di_process: pre_fb.size.height = 480
di_process: pre_fb.format = 1
di_process: next_fb.addr[0] = 0x62f2a000
di_process: next_fb.addr[1] = 0x62f8a000
di_process: next_fb.addr[2] = 0xab85c3c000000002
di_process: next_fb.size.width = 720
di_process: next_fb.size.height = 480
di_process: next_fb.format = 1
di_process: source_regn.width = 720
di_process: source_regn.height = 480
di_process: output_fb.addr[0] = 0x640b3000
di_process: output_fb.addr[1] = 0x64113000
di_process: output_fb.addr[1] = 0x4790a70
di_process: output_fb.size.width = 720
di_process: output_fb.size.height = 480
di_process: output_fb.format = 1
di_process: out_regn.width = 720 ----------这个参数是指视频图像有效尺寸

视频制作出来就固定的

```
di_process: out_regn.height = 480
di_process: field = 1                    ----------表示现在是顶场还是底场
di_process: top_field_first = 1          ----------顶场先还是底场先
di_process: field = 1
di_process: in_flag_phy = 0x61f7f000     ----------动态监测参数的地址
di_process: out_flag_phy = 0x61ef5000
di_irq_service: enter                    ----------中断进入标志，timeout 则异常可能是带宽或者参数配错导致

sunxi_di_ioctl: out!!
```

7.3. De-Interlace 问题指南

7.3.1. 播放 I 源视频卡顿

- 现象：
  播放 I 源视屏卡顿
- 第一步：
  播放相同帧数的 P 源视频，如果也卡顿，那就不是驱动的问题，按照性能类导致显示异常问题显示卡顿问题去排查，如果不卡进入第二步。
- 如果播放 P 源视频不卡顿，那么查看带宽查看当前系统带宽情况，看给 deinterlace 的带宽够不够

7.3.2. timeout 问题

- 现象：
  当 di 驱动报“di_timer_handle: timeout”
- 第一步：
  先确认是否偶尔出现一次，如果小概率出现是可以接受的，可能当前系统负载过大，di 的优先级又很低，分配到的资源少，处理不过来，导致超时，否则进入第二步。
- 第二步：
  如果频报 timeout 就要看是不是输入参数传递错误，详见传递参数信息

7.3.3. invalid address

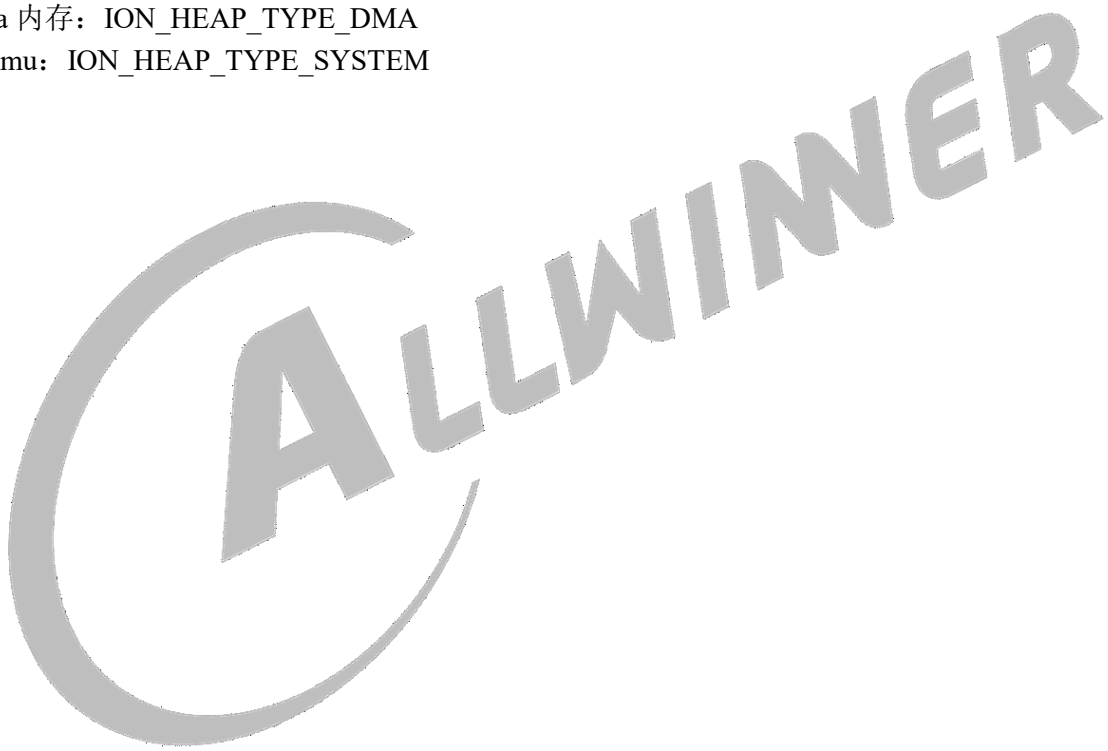
- 问题现象：
  串口狂打印”DE invalid address: 0x0, data:0x0, id:0x1”
- 排查步骤：
  访问了一片没有分配到内存的区域，导致越界报错。
  排查出现问题的地址是否是 deinterlace 访问的那片区域，如果是，那就要看是否内存申请小了，或者对齐没做好；如果不是，那就找 iommu 同事联调。

7.3.4. dmabuf get fd failed

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  显示 buffer 句柄 fd 提前释放或不存在，非驱动问题，排查显示应用 buffer 调用流程。

7.3.5. dma_map_sg failed

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  ION 申请 buffer 用错 flag，要检查
  Cma 内存：ION_HEAP_TYPE_DMA
  Iommu：ION_HEAP_TYPE_SYSTEM



8. G2D 调试说明

8.1. 设备节点

dev/g2d

8.2. 传递参数信息

1. echo 1 > /sys/class/g2d/g2d/attr/debug 打开调试信息
2. echo 0 > /sys/class/g2d/g2d/attr/debug 关闭调试信息
   如果打印太多，可以先调高打印等级 echo 0 > /proc/sys/kernel/printk,然后 dmesg 查看信息

```
[G2D-g2d_bsp_bitblt] line:2036: BITBLT_flag:    0x100 ----------设置旋转角度
                                                       0x0      不旋转
                                                       0x400    0 度
                                                       0x100    90 度
                                                       0x200    180 度
                                                       0x300    270 度
[G2D-g2d_bsp_bitblt] line:2163: ROT input info: ----------
[G2D-g2d_bsp_bitblt] line:2164: Src_fd:    21                源图片 buf fd
[G2D-g2d_bsp_bitblt] line:2165: Format:    0x6 ----------输入 format
[G2D-g2d_bsp_bitblt] line:2166: Flag:    0x100
[G2D-g2d_bsp_bitblt] line:2167: inClipRectX:    0 ----------crop x
[G2D-g2d_bsp_bitblt] line:2168: inClipRectY: 0 ----------crop y
[G2D-g2d_bsp_bitblt] line:2169: inClipRectW:    1280 ----------crop w
[G2D-g2d_bsp_bitblt] line:2170: inClipRectH: 800 ----------crop h
[G2D-g2d_bsp_bitblt] line:2179: ROT_IFMT: 0x6
[G2D-g2d_bsp_bitblt] line:2180: ROT_ISIZE: 0x31f04ff ----------w h 写入寄存器的值
[G2D-g2d_bsp_bitblt] line:2182: SRC_align: 64, 0, 0 ----------源对齐方式
[G2D-g2d_bsp_bitblt] line:2184: DST_align: 64, 0, 0 ----------输出图片对齐方式
[G2D-g2d_bsp_bitblt] line:2195: ROT_InPITCH: 5120, 0, 0 ----------输入 pitch
[G2D-g2d_bsp_bitblt] line:2237: ROT output info: ----------
[G2D-g2d_bsp_bitblt] line:2238: Dst_fd:    40 ----------输出图片 buf fd
[G2D-g2d_bsp_bitblt] line:2239: Format:    0x6 ----------输出 format
[G2D-g2d_bsp_bitblt] line:2249: ROT_OutPITCH: 3200, 0, 0 ----------输出 pitch
[G2D-g2d_bsp_bitblt] line:2250: outClipRectX:    0 ----------crop x
[G2D-g2d_bsp_bitblt] line:2251: outClipRectY: 0 ----------crop y
[G2D-g2d_bsp_bitblt] line:2252: outClipRectW:    800 ----------crop w
[G2D-g2d_bsp_bitblt] line:2253: outClipRectH: 1280 ----------crop h
[G2D-g2d_bsp_bitblt] line:2270: DST_ADDR0: 0x2800000 ----------输出映射后的地址
[G2D-g2d_bsp_bitblt] line:2271: DST_ADDR1: 0x2904000 ---------- 同上
[G2D-g2d_bsp_bitblt] line:2272: DST_ADDR2: 0x29fe000 ---------- 同上
[G2D-g2d_bsp_bitblt] line:2278: init_module: 0x80000011 ---------- 写到寄存器控制器值
```

[G2D-g2d_bsp_bitblt] line:2036: BITBLT_flag: 0x100

8.3. G2D 问题指南

8.3.1. invalid address

- 问题现象：
  串口狂打印”DE invalid address: 0x0, data:0x0, id:0x1”
- 排查步骤：
  一般这种情况是访问了一片没有分配到内存的区域，导致越界报错
  要排查出现问题的地址是否是 g2d 访问的那片区域，如果是，那就要看是否内存申请小了，或者对齐没做好；如果不是，那就找 iommu 同事联调。

8.3.2. dmabuf get fd failed

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  显示 buffer 句柄 fd 提前释放或不存在，非驱动问题，排查显示应用 buffer 调用流程。

8.3.3. dma_map_sg failed

- 问题现象：
  串口打印 get fd failed。
- 排查方法：
  ION 申请 buffer 用错 flag，要检查
  Cma 内存：ION_HEAP_TYPE_DMA
  Iommu：ION_HEAP_TYPE_SYSTEM

8.3.4. 旋转画面撕裂

一般出现这个原因，多半是对齐没做好，尤其是 YV12 这种，旋转之后 Y、U、V 三个分量也要对齐

8.3.5. 旋转系统挂死

一般出现这种情况，可能是上电时序出了问题，这个时候检查几个总线 gating 是否开启
对应 spec 寄存器 G2D_SCLK_GATE、G2D_HCLK_GATE、G2D_AHB_RESET。

9. displayd 调试说明

9.1. 调试信息说明

9.1.1. 获取常规 Log 的方法

```
/*
    logcat -s displayd
*/
```

Logcat 是调试 Displayd 模块的主要方法

9.1.2. 获取底层驱动状态的方法

```
/*
    cat /sys/class/disp/disp/attr/sys
*/
```

通过驱动节点获取 DE 驱动当前的输出信息

9.2. 调试方法介绍

9.2.1. 系统启动时的 Log 分析

```
/*
    01-02 13:37:29.976   1830   1830 D displayd: device: type=4, hotplugsuport=1, revertplug=0
    01-02 13:37:29.976   1830   1830 D displayd: device: type=2, hotplugsuport=1, revertplug=0
    01-02 13:37:29.976   1830   1830 D displayd: has no propName=persist.sys.disp_dev2
    01-02 13:37:29.976   1830   1830 D displayd: MAIN_DISPLAY: priority=0, type=4
    01-02 13:37:29.976   1830   1830 D displayd: AUX_DISPLAY: priority=1, type=2
    01-02 13:37:29.976   1830   1830 D displayd: currentDispNum=2, mainPriority=0
    01-02 13:37:29.976   1830   1830 E displayd: tiger a hdmi mode check!
    01-02 13:37:29.976   1830   1830 D displayd: Force: display 0 type 4 mode 255
    01-02 13:37:30.003   1830   1830 D displayd: display.0 type 4, plug in
    01-02 13:37:30.003   1830   1830 D displayd: display.1 type 2, plug out
    01-02 13:37:30.003   1830   1830 D displayd: Force: display 1 type 0 mode 0
*/
```

#type 表示显示设备的类型

```
DISP_OUTPUT_TYPE_NONE   = 0,
DISP_OUTPUT_TYPE_LCD    = 1,
DISP_OUTPUT_TYPE_TV     = 2,
DISP_OUTPUT_TYPE_HDMI   = 4,
DISP_OUTPUT_TYPE_VGA    = 8,
```

4 表示 HDMI，2 表示 CVBS，其他设备类型定义如上

#mode 表示显示模式

```
DISP_TV_MOD_480I                = 0,
DISP_TV_MOD_576I                = 1,
DISP_TV_MOD_480P                = 2,
DISP_TV_MOD_576P                = 3,
DISP_TV_MOD_720P_50HZ           = 4,
DISP_TV_MOD_720P_60HZ           = 5,
DISP_TV_MOD_1080I_50HZ          = 6,
DISP_TV_MOD_1080I_60HZ          = 7,
DISP_TV_MOD_1080P_24HZ          = 8,
DISP_TV_MOD_1080P_50HZ          = 9,
DISP_TV_MOD_1080P_60HZ          = 0xa,
DISP_TV_MOD_1080P_24HZ_3D_FP    = 0x17,
DISP_TV_MOD_720P_50HZ_3D_FP     = 0x18,
DISP_TV_MOD_720P_60HZ_3D_FP     = 0x19,
DISP_TV_MOD_1080P_25HZ          = 0x1a,
DISP_TV_MOD_1080P_30HZ          = 0x1b,
DISP_TV_MOD_PAL                 = 0xb,
DISP_TV_MOD_PAL_SVIDEO          = 0xc,
DISP_TV_MOD_NTSC                = 0xe,
DISP_TV_MOD_NTSC_SVIDEO         = 0xf,
DISP_TV_MOD_PAL_M               = 0x11,
DISP_TV_MOD_PAL_M_SVIDEO        = 0x12,
DISP_TV_MOD_PAL_NC              = 0x14,
DISP_TV_MOD_PAL_NC_SVIDEO       = 0x15,
DISP_TV_MOD_3840_2160P_30HZ     = 0x1c,
DISP_TV_MOD_3840_2160P_25HZ     = 0x1d,
DISP_TV_MOD_3840_2160P_24HZ     = 0x1e,
DISP_TV_MOD_4096_2160P_24HZ     = 0x1f,
DISP_TV_MOD_4096_2160P_25HZ     = 0x20,
DISP_TV_MOD_4096_2160P_30HZ     = 0x21,
DISP_TV_MOD_3840_2160P_60HZ     = 0x22,
DISP_TV_MOD_4096_2160P_60HZ     = 0x23,
DISP_TV_MOD_3840_2160P_50HZ     = 0x24,
DISP_TV_MOD_4096_2160P_50HZ     = 0x25,
```

以上除了 DISP_TV_MOD_PAL 和 DISP_TV_MOD_NTSC 是针对 CVBS 接口的分辨率外，剩下的分辨率都是针对 HDMI 定义的

Mode 为 255 表示提示 HDMI 自适应输出，具体由电视的 EDID 信息决定

#MAIN_DISPLAY 表示主显设备的类型

从上面的 Log 看主显是 HDMI

#AUX_DISPLAY 表示辅显设备类型

从上面的 Log 看辅显是 CVBS

#小结

综上所述，通过以上 Log 可以获取到：当前主显设备和辅显设备各是什么类型，输出模式是怎样的，当前热插拔状态如何。如果需要导入 RDM，就把对应 Log 和问题现象给到相应工程师

9.2.2. 热插拔时的 Log 分析

```
/*
    [   537.050101] HDMI cable is disconnected
    01-02 13:46:16.415   1830   2118 D displayd: hotplug change: name=hdmi, state=0
    01-02 13:46:16.416   1830   2118 D displayd: hotplugListener: hdmi disconnect
    [   537.217934] CPU1: Booted secondary processor
    [   537.217934] CPU1: update cpu_power 1024
    [   538.811784] CPU1: shutdown
    [   538.814808] psci: CPU1 killed.
    [   539.490084] HDMI cable is connected
    01-02 13:46:20.462   1830   2118 D displayd: hotplug change: name=hdmi, state=1
    01-02 13:46:20.462   1830   2118 D displayd: hotplugListener: hdmi connect
    01-02 13:46:20.462   1830   2118 D displayd: Force: display 0 type 4 mode 255
*/
```

#"hotplug change: name=%s, state=%s "用来提示当前检测到的热插拔状态

hotplug change: name=hdmi, state=1 表示当前 HDMI 插入，displayd 在检测到插入后会去打开 HDMI 设备

如果检测到 HDMI 插入的情况下没有输出一般是 HWC 或者 DE 驱动的问题，如果反复插拔 HDMI 线没有相应 Log 一般是热插拔检测存在问题

#cvbs 接口分析过程同上

9.2.3. 分析 DE 的实际状态

```
console:/ # cat /sys/class/disp/disp/attr/sys
screen 0:
de_rate 696000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank direct_show[false]
dmabuf: cache[7] cache max[8] umap skip[0] overflow[2]
        hdmi output mode(10)    fps:60.6        1920x1080
        err:0   skip:11 irq:734 vsync:165       vsync_skip:0
   BUF    enable ch[0] lyr[0] z[0] prem[N] a[globl 255] fmt[  5] fb[1920,1080; 960, 540; 960, 540] crop[   0,   0,1920,1080] frame[
 0,   0,1920,1080] addr[fd000000,fd1fa400,fd1fa400] flags[0x       0] trd[0,0]
disp[0]all:55, sub:55, cur:55, free:53, skip:0
disp[1]all:0, sub:0, cur:0, free:0, skip:0
console:/ #
```

通过/sys/class/disp/disp/attr/sys 获取当前 DE 驱动的输出信息

#cs 表示 color space 颜色空间

```
DISP_UNDEF = 0x00,
DISP_UNDEF_F = 0x01,
DISP_GBR = 0x100,
DISP_BT709 = 0x101,
DISP_FCC = 0x102,
DISP_BT470BG = 0x103,
DISP_BT601 = 0x104,
DISP_SMPTE240M = 0x105,
DISP_YCGCO = 0x106,
DISP_BT2020NC = 0x107,
DISP_BT2020C = 0x108,
```

一般低分辨率为 0x104，高分辨率为 0x101，这里重点关注

#eotf

主要是内部使用

#fps

表示当前硬件设备的输出帧率，和设置的显示模式有关

#mode 表示显示模式

具体定义同上

#小结

一般有输出但是显示异常的问题都和 DE 驱动状态异常有关，导入相关问题时至少应该将对应 Log 给到相应工程师

9.3. 常见问题总结

9.3.1. 典型案例一：切换分辨率后有图像无声音

HDMI 分为两种工作模式：HDMI 和 DVI，DVI 模式是没有声音输出的，如果想要声音而没有，检查下是否错误切换到了 DVI 模式

如下图所示，表明当前 HDMI 工作在 HDMI 模式

```
console:/ # cat /sys/class/hdmi/hdmi/attr/hdmi_source

HPD:  1

RxSense:  1

PhyLock:  1

PhyPower:  1

TmdsMode:  HDMI
```

9.3.2. 典型案例二：切换分辨率后黑屏问题

切换高分辨率异常，尤其是 4K@60 4K@50 等典型分辨率，注意电视是否仅支持 YUV420 格式的高分辨率规格

如下图所示，表明电视端仅支持 YUV420 格式的高分辨率有哪些

```
console:/ # cat /sys/class/hdmi/hdmi/attr/hdmi_sink


Video Mode:  2160P24  1080P60  1080P50  720P60  720P50  1080I60  1080I50  720x480P  720x480P  576P  1080P24  1080P25  1080P30  576I
640x480P  2160PP25  2160P30  4096x2160P24  4096x2160P25  4096x2160P302160P302160P252160P244096x2160P24

Only Support YUV420:  2160P50  4096x2160P50  2160P60  4096x2160P60
```

确认当前是否切换到了 YUV420 输出格式

```
console:/ # cat /sys/class/disp/disp/attr/sys
screen 0:
de_rate 696000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank direct_show[false]
dmabuf: cache[14] cache max[25] umap skip[2] overflow[14]
        hdmi output mode(10)     fps:60.6        1920x1080
        err:0    skip:35 irq:75186       vsync:986       vsync_skip:0
   BUF    enable ch[0] lyr[0] z[0] prem[N] a[globl 255] fmt[  1] fb[1920,1080; 960, 540; 960, 540] crop[   0,   0,1920,1080] frame[
 0,   0,1920,1080] addr[ff000000,ff1fa400,ff1fa400] flags[0x       0] trd[0,0]
   BUF    enable ch[1] lyr[0] z[1] prem[Y] a[globl 255] fmt[  1] fb[1920,1080; 960, 540; 960, 540] crop[   0,   0,1920,1080] frame[
 0,   0,1920,1080] addr[fb800000,fb9fa400,fb9fa400] flags[0x       0] trd[0,0]
disp[0]all:977, sub:977, cur:977, free:974, skip:2
disp[1]all:0, sub:0, cur:0, free:0, skip:0
```

9.3.3. 典型案例三：插入 HDMI 或者 CVBS 后无图像

首先检测下 Displayd 是否检测到了热插拔事件

/*

logcat -s displayd

*/

如果未检测到插入事件排查下是否驱动未上报导致的异常：

a.获取 HDMI 的热插拔状态

/*

cat /sys/class/switch/hdmi/attr/state

*/

b.获取 CVBS 的热插拔状态

/*

cat /sys/class/switch/cvbs/attr/state

*/

其中 1 表示插入状态，0 表示拔出状态

如果热插拔状态正常而无图像把问题反馈给相应工程师，否则应该确认硬件连接是否存在问题，导致的热插拔上报异常

9.3.4. 典型案例四：设置或者切换分辨率后有图像但是颜色异常

一般有图像输出但是颜色异常的问题是颜色空间设置异常导致的，重点检查设置的颜色空间是否正确

```
console:/ # cat /sys/class/disp/disp/attr/sys
screen 0:
de_rate 696000000 hz, ref_fps:60
mgr0: 1920x1080 fmt[yuv444] cs[0x101] range[limit] eotf[0x4] bits[8bits] err[5] force_sync[6] unblank direct_show[false]
dmabuf: cache[7] cache max[44] umap skip[175] overflow[217]
        hdmi output mode(10)    fps:60.5        1920x1080
        err:2   skip:112        irq:79546       vsync:75210     vsync_skip:0
   BUF    enable ch[1] lyr[0] z[0] prem[Y] a[globl 255] fmt[  1] fb[1920,1080; 960, 540; 960, 540] crop[   0,   0,1920,1080] frame[
 0,   0,1920,1080] addr[f1000000,f11fa400,f11fa400] flags[0x       0] trd[0,0]
disp[0]all:35551, sub:35551, cur:35550, free:35547, skip:211
disp[1]all:0, sub:0, cur:0, free:0, skip:0
```

对于 HDMI 来说，分辨率小于 720P@50 应该是 0x104，大于则为 0x101

对于 CVBS 来说，因为分辨率不超过 720P，所以应该为 0x104