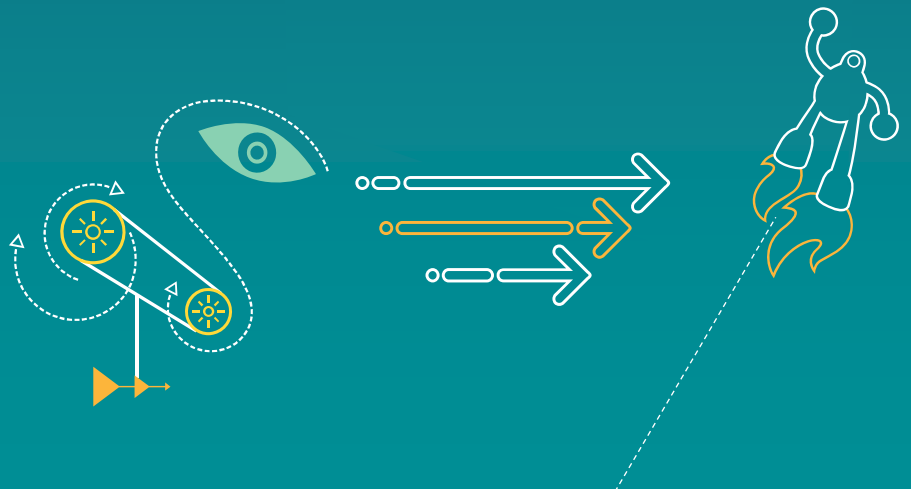
高通多媒体技术期刊　20140709

QUALCOMM®

Qualcomm Technologies, Inc.



Revision History

Revision	Date	Description
A	July 2014	Initial release

Contents

- Display
- Graphics
- Video

Display

Android Display Debug 调试综述

- 显示流程从时间上可以分为 LK display 和 Android display。
  - LK display 显示的是一张静态图片Logo.
  - Android display 显示的时间流程如下

    继续显示LK的Logo图片（连续显示）

    开机动画

    进入launcher 的桌面，Android开始正常运行。

  当看到手机屏幕显示不正常时，首先我们要明确什么是framework领域的display问题，什么是kernel领域的display问题。
- Framework领域的display问题。（参考Solution:00028564）
  - Android 的framework可以分为 Java层和Native层。display的相关问题更多的涉及到native层。Java层的WindowManagerService和其它的系统服务一般与底层硬件关系不大，更多属于UI定制相关。
  - SurfaceFlinger 是native层的显示核心服务进程。Display HAL 层被 SurfaceFlinger 调用。
  - 代码分布

    //frameworks/native/services/surfaceflinger/ 主目录

    //frameworks/native/services/surfaceflinger/DisplayHardware/ 与HAL层HWC的接口文件
- Kernel领域的display问题。（参考Solution:00028565）
  - display HW logic core MDSS 的寄存器操作，以及 各种clock和系统BW的vote和set
  - DSI/HDMI/eDP/LVDS/LCDC/MDDI/EBI2/QPIC/WFD接口相关的驱动，LCD panel相关设置和操作
  - 代码分布

    //kernel/drivers/video/msm/mdss/ MDSS 核心驱动

    //kernel/arch/arm/boot/dts/qcom/ LCD panel, mdss以及各种 board 设备树配置文件

如何调试LCD蓝屏(under-run)问题

- 在开发过程中，遇见的LCD闪蓝屏问题，一般属于MDSS under-run。具体视觉效果差异较大，可以全屏闪，也可以局部闪动。有时难以确定复现路径，属于概率性问题。 其颜色可以通过panel dtsi文件(@file \kernel\arch\arm\boot\dts\qcom\dsi-panel-nt35590-720p-video.dtsi). <qcom,mdss-dsi-underflow-color>设置，以方便调试和最后阶段预防性的规避问题（可以设置成黑色）。
- Under-run一般是因为display data flow 不能及时为MDSS 传送显示数据，MDSS HW自动填充数据以满足外部LCD panel的时序要求。
- 可以参考Solution:00028556 进行under-run debug调试。
- 1. 有两种方法可以辨别under-run发生与否。
  - 使用debugfs，如果under-run计数增加，那么就发生了under-run.
- adb shell
- cd d/mdp //d 是 debugfs 路径
- cat stat
- mdp:
- intf2: play: 00000000 vsync: 00006823 underrun: 00000000
  - 在kernel文件中增加log信息，方便与特定frame信息一起打印。
- mdss_mdp_video_underrun_intr_done(void *arg)” (@ file \kernel\drivers\video\msm\mdss\mdss_mdp_intf_video.c).
- 更改 pr_debug 为 pr_err
- pr_debug("display underrun detected for ctl=%d count=%d\n", ctl->num, ctl->underrun_cnt);
- 2. 有两种初步的调试方法
  - 增加MDP的clock来确定是否与MDP clock计算错误有关。
- static void mdss_mdp_ctl_perf_update(struct mdss_mdp_ctl *ctl, int params_changed)” (@ file \kernel\drivers\video\msm\mdss\mdss_mdp_intf_ctl.c)
- -- mdss_mdp_set_clk_rate(clk_rate);
- ++ mdss_mdp_set_clk_rate(320000000); // 320 MHZ 是 8916的最大值，每个芯片有所不同。

如何调试LCD蓝屏(under-run)问题

- 增加DDR clock 和AXI bus clock的投票值。
- “int mdss_mdp_bus_scale_set_quota(u64 ab_quota, u64 ib_quota)” (@ file \kernel\drivers\video\msm\mdss\mdss_mdp_intf_ctl.c)
- -- vect->ab = ab_quota;
- -- vect->ib = ib_quota;
- ++ vect->ab = ab_quota *ab_fudge_factor;
- ++ vect->ib = ib_quota *ib_fudge_factor;
- 这些 factor可以是1.25, 1.5, 1.75，或者 中间数值。
- 以上两种方法可以初步定位问题，是否与MDSS clock以及系统带宽相关。
- 3. 增加VBP
  - 如果VBP+VSYNC<6 ， 可以考虑增加VBP，VBP的具体数值需要符合LCD panel的要求，否则LCD panel可能花屏或者黑屏。
- 4. LK MDSS设置检查。如果under-run只发生在第一次suspend之前，那么需要检查LK中 MDSS的设置，重点检查MDSSL clock设置是否与Kernel一致。并观察kmsg是否有明确的mdss error信息。
- 5. 添加实时logs，进行调试。
  - https://www.codeaurora.org/cgit/quic/la//kernel/msm-3.10/commit/?id=50df44137b1ada21b4cb294fbb0c2fe07133d683
  和入这个patch之后，抓取logs 一直到问题复现为止，然后提供logs给高通技术支持团队。
- 6. 看问题是否与整个系统performance相关。设置CPU进入performance模式，并强制多核同时工作。
- 需要注意这个设备是否过热，如果过热，热保护模块会自动对CPU进行降频处理，以下操作可能无效。
- adb shell stop mpdecision
- adb shell stop thermal-engine
- adb shell "echo 1 > /sys/devices/system/cpu/cpuX/online" (cpuX : cpu0, cpu1 …)
- adb shell "echo performance > /sys/devices/system/cpu/cpuX/cpufreq/scaling_governor" (cpuX: cpu0, cpu1, ...)

如何调试LCD蓝屏(under-run)问题 — 续一

7. 看问题是否与整个系统performance相关。提高AXI 总线速度。

   和入下面的patch

   https://www.codeaurora.org/cgit/quic/la//kernel/msm-3.10/commit/?id=68152d72c2ab0a6c9e252ad323ec898389faebeb

   看问题是否解决，来判断是否与AXI bus速度相关。

   adb shell cat /d/mdp/perf/min_bus_vote 来看当前最新的总线投票。

   adb shell echo [required bw to be set] /d/mdp/perf/min_bus_vote by 1.2 , 1.3 or 1.5 ,etc来增加总线投票。

8. 请把以上调试情况反馈给高通技术支持团队，进行最后的分析确认，以上是调试手段，不能做为最终的解决方案。

如何获取BIMC/DDR 当前clock，进行Android kernel display调试

- 在调试过程当中， display相关的performance , power以及前面的under-run 问题，都与BIMC当前clock相关。BIMC是高通DDR控制器的名称，一般在软件及文件系统中，经常用BIMC来代指DDR。在display 调试中,BIMC clock可以理解为DDR内存当前频率。系统内存运行的速度，高度影响实际刷新帧率，系统功耗。内存运行速度越快，under-run发生的机会就越小，因为display数据搬运加速，数据流加快，系统的功耗也会随之增加。
- 在调试过程当中， display相关的performance , power以及前面的under-run 问题，都与BIMC当前clock相关。在 8916上面，可以用以下的方法来获取当前的BIMC clock。
  a) adb root
  b) adb shell
  c) mount -t debugfs nodev /sys/kernel/debug/
  d) cat /sys/kernel/debug/clk/bimc_clk/measure
  199999487 // use-case: 静态桌面
  e) cat /sys/kernel/debug/clk/bimc_clk/measure
  532996791 // use-case: 滑动桌面
- 参考 Solution:00028883。

8916 display 相关库文件，可执行文件以及apk说明。

- CABL是一种高通提供的免费的显示节能技术。/system/app/CABLService.apk ，这个apk提供对CABL的开关控制。一般情况下CABL是默认打开的。在Qualcomm setting apk里面，有一个CABL选项，如果点击一下，这个apk启动，会看见CABL的开关界面。

  ro.qualcomm.cabl 提供同样的控制。如果不使用CABL功能，在ROM中可以不打包此文件。

  并设置ro.qualcomm.cabl=0

  如果计划使用CABL功能，请与高通技术支持团队联系，进行专项技术支持。
- SVI是一种强光照下显示增强技术，适用于8916平台。/system/app/SVIService.apk，这个apk与SVI技术控制相关。如果不计划使用此功能， ROM中可以不打包此文件。如果计划评估此功能，请与高通市场及产品管理团队联系。
- /system/app/PPPreference.apk，提供对颜色，饱和度的简单调节。在Qualcomm setting apk里面，有一个HSIC选项，如果点击一下，这个apk启动，会看见HSIC的调节界面。

  此APK可以供显示效果简单调节使用。
- /system/bin/mm-pp-daemon，可执行文件，在系统启动时，开始运行。是显示子系统后处理功能的守护进程，核心服务程序。如果不使用任何高通显示后处理功能的情况，可以在init.rc中停止本项服务。可以和高通技术支持团队进行各种显示后处理功能的确认。
- /system/lib/liboverlay.so, /system/vendor/lib/hw/hwcomposer.msm8916.so HAL层composition相关的库文件
  /system/vendor/lib/hw/gralloc.msm8916.so 显示HAL层的库文件，内存分配相关
- /system/vendor/lib/libmm-abl-oem.so, /system/vendor/lib/libmm-abl.so, CABL相关的库文件

如何计算Panel的Timing寄存器值

1. 首先，需要从https://downloads.cdmatech.com/ 网站，下载计算Timing的Excel 表格，具体的文档标号为：
   80-NH713-1 DSI TIMING PARAMETERS USER INTERACTIVE SPREADSHEET.xlsm

2. 在80-NH713-1表格中，把LCD vendor推荐的V Porch, H porch 等相关的值 输入到 “DSI and MDP registers” 工作单，如右图所示，然后按“CTRL + L”。

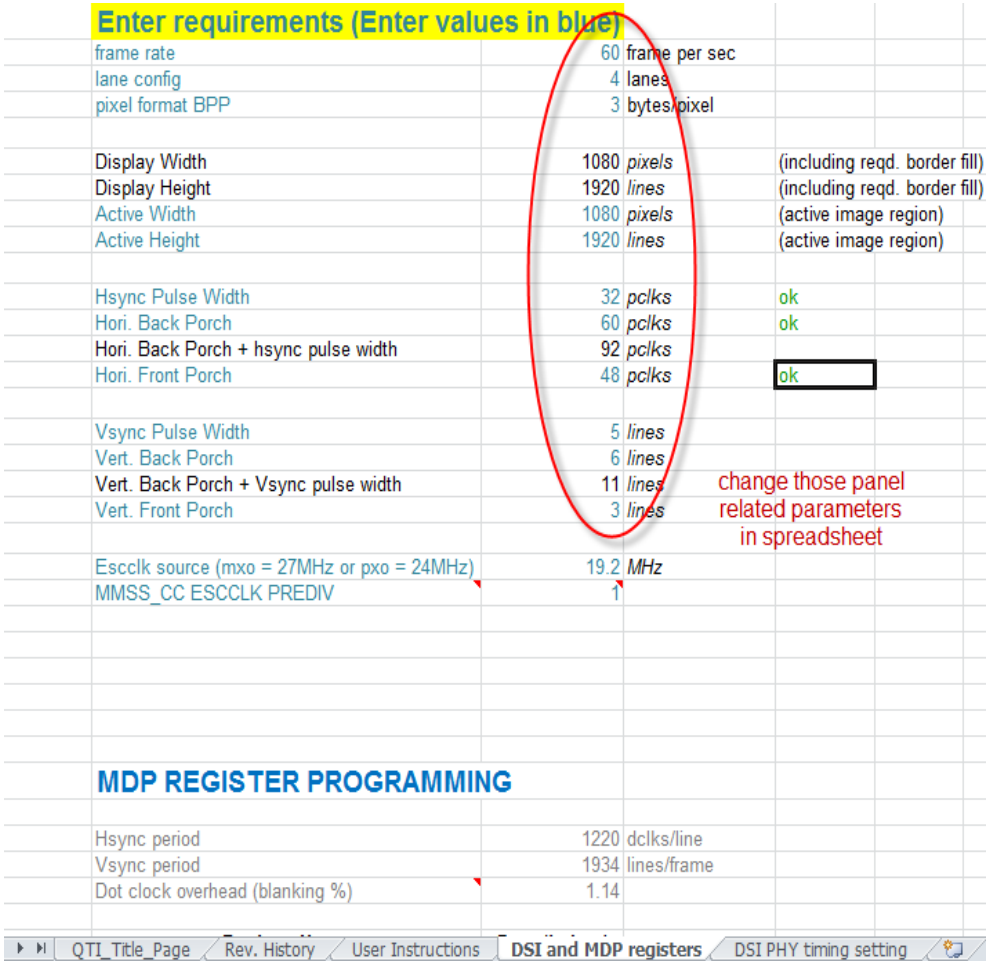
3. 切换到“DSI PHY timing setting” 工作单，在“Check for T_CLK_ZERO”选项，会显示“INVALID”。按“CTRL + J”，“CTRL + K”，最终得到有效的DSI Timing值。

注意：

a) 如果80-NH713-1 表格不能在 Windows XP 正常工作，请使用Win7 系统。

b) 最好使用Microsoft Excel 2010版本去运行此表格。

参考solution：00029225



Graphics

HW UI 问题解答(Solution: 00028654)

- **1.** 怎么检查一个apk是否应用了HW UI?
  Google 提供了dump HW UI 信息的方法，运行命令 dumpsys gfxinfo 就会打印出最近50 批commands中调用那些HWUI 接口函数。
  如果HW UI 没有应用，则dumpsys gfxinfo 就没有信息打印出来。
  下面是一个例子：
  adb shell
  >dumpsys gfxinfo [PID of the app, or name of app]
  ** Graphics info for pid 1095 [com.android.systemui] **
  Recent DisplayList operations
  DrawDisplayList
  Save
  ClipRect

- **2.** 怎样得到Application 设置的EGL configurations?
  apk可以应用Java层面的Holder 或者Window 操作来设置。我们可以检查是否是EGL configures引起的问题。
  具体命令如下：
  adb shell setprop debug.hwui.print_config choice
  adb shell stop
  adb shell start
  adb logcat

D/HardwareRenderer(8808): EGL configuration com.google.android.gles_jni.EGLConfigImpl@4202ad88:
D/HardwareRenderer(8808): RED_SIZE = 8
D/HardwareRenderer(8808): GREEN_SIZE = 8
D/HardwareRenderer(8808): BLUE_SIZE = 8
D/HardwareRenderer(8808): ALPHA_SIZE = 8
D/HardwareRenderer(8808): DEPTH_SIZE = 0
D/HardwareRenderer(8808): STENCIL_SIZE = 8
D/HardwareRenderer(8808): SAMPLE_BUFFERS = 0
D/HardwareRenderer(8808): SAMPLES = 0
D/HardwareRenderer(8808): SURFACE_TYPE = 0x5e5
D/HardwareRenderer(8808): CONFIG_CAVEAT = 0x3038

HW UI 问题解答 — 续一

- **3.** 怎么获得GL-API调用trace 信息?

  Canvas API 是基于HWUI之上工作的，通过调用OPENGLES API来实现。因此GL API call trace 是一个很重要的调适方法。用如下命令可以在logcat log中获得GL-API 调用trace信息

  adb shell setprop debug.egl.trace 1

  adb shell stop

  adb shell start

- **4.** 如果我怀疑一个issue可能是HWUI引起的，我怎么关掉HWUI?

  1.) 从系统层面彻底关掉HWUI，影响所有application。

  HWUI 是android framework的一部分，Canvas API 本来是基于2D SKIA软件图像库实现的，现在部分被替换到3D Graphics来实现，因为在高分辨率的情况下，3D的性能要比2D号很多。如果关掉HWUI, 那么所有的Canvas API都只工作在2D SKIA 软件图形库上面。

  修改如下code, 重新编译libhwui.so库文件并更新

  frameworks/base/core/jni/android_view_GLES20Canvas.cpp

  static jboolean **android_view_GLES20Canvas_isAvailable**(JNIEnv* **env**, jobject **clazz**) {

  + return JNI_FALSE;

  -

  }

  通过关掉HWUI, 我们可以判断是否application自己的问题。

  2.) 从具体应用apk层面关掉HWUI, 只影响特定的application。

  在apk的源文件目录，找到AndroidManifest.xml, 定位<application /> tag后做如下修改:

  android:hardwareAccelerated="false“

  然后重新build apk push到手机测试。

HW UI 问题解答 – 续二

- **5.** 怎么设置让HWUI全屏更新

  默认情况下，HWUI在render EGL Surface时设置了preserved bit, 当只有部分区域render不正确，需要检查一下是否是partial update的问题

  adb shell setprop debug.hwui.render_dirty_regions false

  adb shell stop

  adb shell start

  设置之后，HWUI render会不再保留EGL Surface里面的内容，每次都会render全部的内容。

- **6.** 怎么获取更多Texture的信息?

  HWUI 自己处理Texture相关的信息，意味着即使application 删除了一个Texture, 实际的数据可能还存在于HWUI的Cache里面，下面的编译选项可以输出更多Texture相关的debugging log

  [PLATFORM]/frameworks/base/libs/hwui/Debug.h

  - #define DEBUG_TEXTURES 0

  + #define DEBUG_TEXTURES 0

  然后rebuild hwui然后更新libhwui.so 手机测试。

  adb push libhwui.so /system/lib

- **7.** 怎么找到有问题的Call trace?
- Google 提供了GLTracer 用来Trace GLES/EGL的相关操作。具体的信息，可以参看Google Android 官方网站 http://developer.android.com/tools/help/gltracer.html

GPU performance问题解答(Solution: 00028664)

- **1.** 怎么监控设备FPS (刷新率) ?

  通过设置adreno_config.txt可以使能FPS log输出，参考如下command：

  adb shell rm /data/local/tmp/adreno_config.txt

  adb shell “echo log.fps=1 > /data/local/tmp/adreno_config.txt”

  adb reboot

  然后就可以在logcat log中搜索fps关键字

  adb logcat | grep i –fps

  监控在不同应用场景下fps的变化。

- **2.** 怎么检查设备上GPU, CPU 和DDR 支持的最大Clocks

  首先要mount debugfs

  adb shell mount -t debugfs none /sys/kernel/debug

  对GPU adb shell cat sys/class/kgsl/kgsl-3d0/max_gpuclk

  对CPU adb shell cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq

  对DDR adb cat /sys/kernel/debug/clk/bimc_clk/rate

- **3.** 怎么实时监控GPU, CPU, DDR的Clocks

  首先要mount debugfs

  adb shell mount -t debugfs none /sys/kernel/debug

  对GPU， adb shell cat /sys/kernel/debug/clk/oxili_gfx3d_clk/measure

  对GPU， adb shell cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq

  对DDR， adb shell cat /sys/kernel/debug/clk/bimc_clk/measure

GPU performance问题解答 — 续一

- **4.** 什么是GPU Performance mode 和GPU performance mode?

  GPU performance mode就是强制GPU一直运行在最高时钟频率上。如果设置GPU performance mode问题得到改进，那就需要继续从GPU/Kernel方面检查。

  CPU performance mode是所有的CPU cores运行在最高时钟频率，同时DDR也是最高时钟频率。如果设置CPU performance mode 问题解决了，那么需要System performance team帮助进一步找到root cause.

- **5.** 怎么设置GPU/CPU performance mode?
- GPU performance mode

  你可以检查device 是否支持devfreq? 检查在/sys/class/kgsl/kgsl-3d0 路径下面是否有devfreq目录，如果有就支持，没有就不支持。

  如果你的device 支持devfreq,

  adb shell "echo 1 > /sys/class/kgsl/kgsl-3d0/force_clk_on"
  adb shell "echo 10000000 > /sys/class/kgsl/kgsl-3d0/idle_timer"
  adb shell "echo performance > /sys/class/kgsl/kgsl-3d0/devfreq/governor"
  adb shell "echo <max GPU clock> > /sys/class/kgsl/kgsl-3d0/gpuclk"

  反之你可以用如下命令：

  adb shell "echo 0 > /sys/class/kgsl/kgsl-3d0/pwrnap"
  adb shell "echo 10000000 > /sys/class/kgsl/kgsl-3d0/idle_timer"
  adb shell "echo none > /sys/class/kgsl/kgsl-3d0/pwrscale/policy"
  adb shell "echo <max GPU clock> > /sys/class/kgsl/kgsl-3d0/gpuclk“

- CPU Performance mode

  可以运行如下命令：

  adb shell stop mpdecision
  adb shell stop thermal-engine
  sleep 1
  adb shell "echo 1 > /sys/devices/system/cpu/cpu1/online"
  adb shell "echo 1 > /sys/devices/system/cpu/cpu2/online"
  adb shell "echo 1 > /sys/devices/system/cpu/cpu3/online"
  sleep 1
  adb shell "echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor"
  adb shell "echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor"
  adb shell "echo performance > /sys/devices/system/cpu/cpu2/cpufreq/scaling_governor"
  adb shell "echo performance > /sys/devices/system/cpu/cpu3/cpufreq/scaling_governor“

GPU performance问题解答 – 续二

- 6. 什么工具用来调适Performance问题? 怎么获得?

  一般用Systrace 和 Snapdragon performance visualizer (原名QView)工具来调适Performance问题。Systrace是Android SDK 里面提供的一个功能，可以从Google开发者网站上下载。Qualcomm Snapdragon performance Visualizer 是高通自己的工具，在你开始一个Qualcomm的正式项目，你可以通过联系CE或者TAM，我们会把这个工具release给你。

- 7. 怎么使用Systrace?

  你可以按照如下步骤去抓取一个问题的Systrace：

  1. 下载Android SDK并安装(http://developer.android.com).
  2. 打开tools 路径下面的 monitor.bat
  3. 点击“Capture system wide trace using Android systrace” (http://developer.android.com/tools/help/systrace.html#gui)
  4. 在'Select tags to enable' 里面勾选所有的选项
  5. 现在开始重现出现问题的应用场景，可能会比较慢
  6. 点击 OK开始抓取 Systrace

  然后把抓到Systrace 上传到case attachment供QC case owner 去分析问题。

- 8. 怎么使用Snapdragon Performance Visualizer 工具?

  完整安装好这个工具以后，里面会有一个用户手册(User guide)，有很详细的使用方法说明。

Video

8916视频文档列表

- Video 概述:
  - 80-NL239-14 MSM8916 LINUX ANDROID VIDEO OVERVIEW
- WFD(Wi-Fi Display):
  - WFD源端概述 80-NL239-35 MSM8916 WI-FI DISPLAY MULTIMEDIA OVERVIEW
  - WFD目的端概述 80-NL239-50 MSM8916 WI-FI DISPLAY SINK OVERVIEW
  - WFD设置指南 80-NP152-1 Wi-Fi Display Setup Guide
  - WFD能力描述 80-NP350-1 APPLICATION NOTE: WI-FI DISPLAY CAPABILITY DESCRIPTION for Source
- Openmax集成文档(集成第三方多媒体framework时用):
  - 视频编码集成 80-N1933-3 OPENMAX INTEGRATION LAYER VIDEO ENCODER FOR LINUX ANDROID
  - 视频解码集成 80-VT322-2 OPENMAX INTEGRATION LAYER VIDEO DECODER FOR LINUX ANDROID
- Video特性:
  - 内容保护的视频播放 80-NA157-206 CONTENT PROTECTION FOR MEDIA PLAYBACK
  - 流媒体视频分辨率变化的支持 80-NK913-1 HANDLING RESOLUTION CHANGE ON ANDROID KITKAT
  - 高帧率视频播放支持 80-NL385-1 Smooth Playback for High Framerate Videos on Android Framework

视频CTS/GTS测试指南

- CTS/GTS测试一般性指南:
  - Solution:00028618 What are the steps to debug a GTS/CTS/Compliance tests issue?
  - 我们建议在碰到CTS/GTS测试用例失败时, 通过log及测试报告找到具体失败的测试用例, 然后单独测试这个失败的用例, 并重复测试, 看看能否通过测试, 因为有时问题在于CTS测试代码本身, 并不是手机软件的问题.
  - 记下失败的测试用例以及测试代码版本, 在case中告诉高通的工程师这些信息

- CTS/GTS测试碰到问题请求高通帮助时需要提供的log信息:
  - Solution:00028619 What is the minimal debug information to debug GTS/CTS/Compliance tests?
  - 针对失败原因的不同, 请在抓取log之前打开详细log信息,

    OMX logs:

    >adb shell setprop vidc.debug.level 7

    OMX input/output buffer logs:

    Output buffer log:

    adb shell setprop vidc.dec.log.out 1

    Input buffer log:

    adb shell setprop vidc.dec.log.in 1

MSM8x10平台GTS测试已知问题

- Solution:00028904 8x10 GTS Known Failures

  GTS Package Version: 1.5_r2

  1. com.google.android.xts.media.MediaCodecStressTest#testDecodeDecodeCompositeEncode720p

  Kernel log 指示失败原因:

  msm_vidc: 4: Opening video instance: e4b4e000, 0

  msm_vidc: 4: Opening video instance: e4b4f000, 1

  msm_vidc: 4: Opening video instance: d358d000, 1

  msm_vidc: 1: H/w is overloaded. needed: 324000 max: 244800

  msm_vidc: 1: Running instances:

  msm_vidc: 1: type| w| h| fps

  msm_vidc: 1: 0|1280| 720| 30

  msm_vidc: 1: 1|1280| 720| 30

  msm_vidc: 1: 1|1280| 720| 30 --> 2个720p解码和1个720p编码实例同时运行超过8x10视频支持范围

  msm_vidc: 1: Failed to open instance

  msm_vidc: 4: Closed video instance: d358d000

  msm_vidc: 4: Closed video instance: e4b4f000

  msm_vidc: 4: Closed video instance: e4b4e000

MSM8x10平台GTS测试已知问题 – 续一

2. com.google.android.xts.media.Vp8CodecTest#testSimulcastBitrate

失败原因是vp8视频文件超过MSM8x10能力范围. MSM8x10支持解码vp8的最大分辨率是FWVGA(864x680), 而本测试用例里面的文件是720p的.

Logcat指示失败原因:

VP8CodecTestBase: Creating decoder OMX.qcom.video.decoder.vp8. Color format: 0x13. 1280 x 720

VP8CodecTestBase: Format: {color-format=19, height=720, width=1280, mime=video/x-vnd.on2.vp8}

VP8CodecTestBase: In: /storage/emulated/0/football4.ivf. Out:null

OMXClient: Using client-side OMX mux.

ACodec : [OMX.qcom.video.decoder.vp8] configureCodec returning error -1010

MediaCodec: Codec reported an error. (omx error 0x80001001, internalError -1010)

TestRunner: failed: testSimulcastBitrate(com.google.android.xts.media.Vp8CodecTest)

MSM8x10平台GTS测试已知问题 – 续二

3. com.google.android.xts.media.Vp8CodecTest#testSimulcastEncoderQuality

失败原因是vp8视频文件超过MSM8x10能力范围. MSM8x10支持解码vp8的最大分辨率是FWVGA(864x680), 而本测试用例里面的文件是720p的.

Logcat指示失败原因:

VP8CodecTestBase: Creating decoder OMX.qcom.video.decoder.vp8. Color format: 0x13. 1280 x 720

VP8CodecTestBase: Format: {color-format=19, height=720, width=1280, mime=video/x-vnd.on2.vp8}

VP8CodecTestBase: In: /storage/emulated/0/football4.ivf. Out:null

OMXClient: Using client-side OMX mux.

ACodec : [OMX.qcom.video.decoder.vp8] configureCodec returning error -1010

MediaCodec: Codec reported an error. (omx error 0x80001001, internalError -1010)

TestRunner: failed: testSimulcastEncoderQuality(com.google.android.xts.media.Vp8CodecTest)

MSM8x26/8926/8x28/8928平台GTS测试已知问题

- Solution:00028903 GTS Known Failures on 8x26/8926/8x28/8928

  GTS Package Version: 1.5_r2

  1. com.google.android.xts.media.MediaCodecStressTest#testDecodeDecodeCompositeDisplay1080p

     Kernel log 指示失败原因:

     msm_vidc: 4: Opening video instance: c3cc7000, 1

     msm_vidc: 4: Opening video instance: df5cf000, 1

     msm_vidc: 1: H/w is overloaded. needed: 482400 max: 352800

     msm_vidc: 1: Running instances:

     msm_vidc: 1: type| w| h| fps

     msm_vidc: 1: 1|1920|1080| 30

     msm_vidc: 1: 1|1920|1080| 30 --> 2个1080p@30fps视频实例是不支持的. 最大可以支持1080p@30fps + 720p@30fps.

     msm_vidc: 1: Failed to open instance

     msm_vidc: 4: Closed video instance: df5cf000

     msm_vidc: 4: Closed video instance: c3cc7000

MSM8x26/8926/8x28/8928平台GTS测试已知问题 – 续一

2. com.google.android.xts.media.MediaCodecStressTest#testDecodeDecodeCompositeEncode1080p

Kernel log指示失败原因:

msm_vidc: 4: Opening video instance: c3cc7000, 1

msm_vidc: 4: Opening video instance: df5cf000, 1

msm_vidc: 1: H/w is overloaded. needed: 482400 max: 352800

msm_vidc: 1: Running instances:

msm_vidc: 1: type| w| h| fps

msm_vidc: 1: 1|1920|1080| 30

msm_vidc: 1: 1|1920|1080| 30 --> 2个1080p@30fps视频实例是不支持的. 最大可以支持1080p@30 fps + 720p@30fps.

msm_vidc: 1: Failed to open instance

msm_vidc: 4: Closed video instance: df5cf000

msm_vidc: 4: Closed video instance: c3cc7000

Questions?

You may also submit questions to:
https://support.cdmatech.com