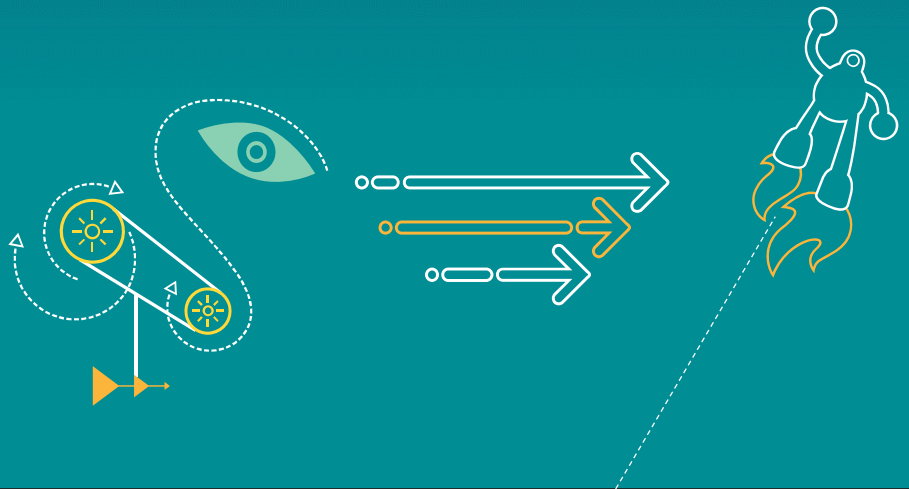
高通多媒体技术期刊　Display合辑 20141210

Qualcomm Technologies, Inc.



Confidential and Proprietary – Qualcomm Technologies, Inc.

Revision History

Revision	Date	Description
A	Dec 2014	Initial release

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

Contents

Key Update

M8916 Display工作流程介绍

- 为更好的支持客户项目显示屏 Bring up, bug fix and tuning, 高通Display团队制定了严格的Display工作流程来帮助客户，80-NP115-1_M8916_Linux_Display_Workflow 详细介绍了项目的开发步骤，以及在各个关键节点Display团队给客户的支持情况，请你尽早阅读
- Solution:00028724 Linux Display Enablement Plan, 高通对客户在开发阶段的支持在这里进行了详细描述
- Solution:00028805 Display Documentation Tree 包含所有M8916 Display相关的文档
- Solution:00028826 Online Display debugging Catalog 包含所有M8916 Display相关问题的解决建议，调试方法等等，请重点阅读
- 80-NN766-1_Linux_Android_Display_Driver_Porting_Guide 详细介绍了显示屏Bring up的步骤

显示屏驱动开发的一般流程

- 80-NN766-1：Linux Android Display Driver Porting Guide详细描述了显示屏的一般流程，这些关键步骤包括，
  - 从GCDB网站 https://createpoint.qti.qualcomm.com 下载高通已经验证过的显示屏驱动
  - 如果显示模组与高通验证的模组完全一样，只需直接把下载的显示驱动集成到你的软件中
  - 否则，请下载具有相同chipset和display IC 的显示驱动， 然后根据显示屏的参数修改输入的XML文件
  - 用QCDB 脚本自动生成 dtsi 文件和 .h 文件
  - 将 dtsi 文件和 .h 文件集成到对应的目录和文件
  - 重新编译kernel和LK images

- 先调试kernel显示驱动，然后再调试LK显示和连续显示
- 调试kernel显示前，请关闭LK显示和连续显示cont_splash

8916显示能力概述

- 8916 提供一路MIPI DSI接口和一路WFD显示
- DSI接口支持Video mode和Command mode.
- 对所有通常应用场景，8916支持 up to 60fps@1280x720分辨率
- 8916能够支持1080x1920分辨率，但某些复杂场景下，帧速率可能不能达到60fps.
- 8916 WFD只支持1280x720@30fps
- 高通推荐
  - 32bit同时分辨率≤ qHD，可以选择512MB RAM
  - 64bit或者分辨率> qHD，必须选择1GB RAM
  - 分辨率FHD，可以选择1GB RAM，同时为了更好的性能，高通推荐2GB RAM
- 请阅读80-NL239-15：MSM8916 LA Display Overview了解8916更多显示能力

Solution#:00029333 M8939 System level solution

- Solution#:00029333 列举了在各个基于M8939/8936芯片的手机开发阶段，高通能够给OEM厂家提供的各种帮助和建议，包括Features definition，各种workflow，以及手机认证等等关键信息，希望客户认真阅读。具体到多媒体领域，
- M8939 Android Project Multimedia Design Review Workflow 介绍了在项目开始阶段，高通多媒体支持团队如何通过multimedia review 会议，了解项目多媒体features要求，并制订相应计划来帮助完成
- MSM8939 Android Camera Project Workflow介绍了在项目各个阶段，camera团队如何给客户提供支持
- MSM8939 Android Audio Project Workflow介绍了在项目各个阶段，Audio团队如何给客户提供支持
- MSM8939 Android Display Project Workflow介绍了在项目各个阶段，Display团队如何给客户提供支持
- MSM8939 Android Video Project Workflow介绍了在项目各个阶段，Video团队如何给客户提供支持
- MSM8939 Android Graphics Project Workflow介绍了在项目各个阶段，Graphics团队如何给客户提供支持

Solution#:00029717 Display相关视频下载

- 高通在今年八月分别在深圳和北京举行了大型的Graphics and display 深入培训，受到了广泛的好评。为了更广泛的帮助到更多的客户，我们陆续发布了相关的视频文件，Solution#:00029717 包含了现在已经发布的Display相关的视频，希望你尽早阅读

- VD80-NP925-1SC - Video: Android Display Debug Guide Training - Simplified Chinese https://downloads.cdmatech.com/qdc/drl/objectId/0901001482a1319e：详细的解释了常见的Display相关的调试技巧和方法，这是最重要的视频，希望客户多多阅读

- VD80-NM328-17SC - Video: MSM8994.LA Linux Android Display Overview Training - Simplified Chinese https://downloads.cdmatech.com/qdc/drl/objectId/0901001482a29df2：介绍了Display相关的硬件模块，系统结构，软件结构

8909 HD/720p LCD panel DSI lanes 硬件设计建议

- 高通强烈建议在 8909 平台上，720p LCD panel采用 4 个 data lanes的设计。没有极为特殊的理由，不要违背这一设计原则。
- 由于8909芯片内部的电源管理机制，采用少于4个data lanes的设计，DSI bit clock的增高会使整个平台的功耗有较大的升高。
- 另外一般来说，在手机结构设计允许的情况下，尽量使用4个data lanes，可以降低DSI bitclock。降低DSI bitclock可以减少与RF以及系统其它模块的电磁干扰，进而提高系统的稳定性。

- Video模式以及command 模式的LCD panel都要遵循这一设计原则。

显示领域 推荐的参考文档

- 80-NP925-1_A_Android_Display_Debug_Guide
  display kernel driver 调试手册，建议仔细阅读，可操作性强。
- 80-NL472-1_B_LINUX_ANDROID_DISPLAY_COMPOSITION_STRATEGY_OVERVIEW
- Android layer composition HAL 原理性讲解文档，读后豁然开朗。建议对照源代码仔细阅读，可
- 以加入打印调试log，帮助理解。可以参考SurfaceFlinger dump。
- 80-NN766-1_A_Linux_Android_Display_Driver_Porting_Guide
- LCD panel bring-up 入门宝典。
- 80-NA157-174_D_DSI_Programing_Guide_B-Family_Android_Devices
- DSI接口调试手册
- 80-NJ164-1_B_Display_Postprocessing_Features_Tech_Notes
- 显示后处理软件核心文档，建议阅读。
- 80-NJ074-1_A_Vsync_Config_Smart_Panels
- Command 模式TE机制深度讲解，建议阅读。
- Solution : 00028826
- Salesforce上面的显示调试 总结性solution，建议阅读。
- https://qualcomm-cdmatech-support.my.salesforce.com/50130000000Vdqs
-

Android Display Overview

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

使用GDCB脚本加速显示屏驱动开发

- GCDB脚本是高通提供给OEM加速显示驱动开发的工具
- GCDB 脚本位于@//~/device/qcom/common/display/tools
- 典型GCDB 命令格式：#perl parser.pl <.xml> panel
- XML 文件是用户指定的输入文件，包括显示屏所有参数配置，XML的参数描述请参考 80-BA103-1：Display GCDB XML Entries
- 强烈建议可以从QCDB网站下载类似的XML，然后根据自己的显示屏做进一步修改，而不是从头创建新的XML

Solution#00029225: 如何计算Panel的Timing寄存器值

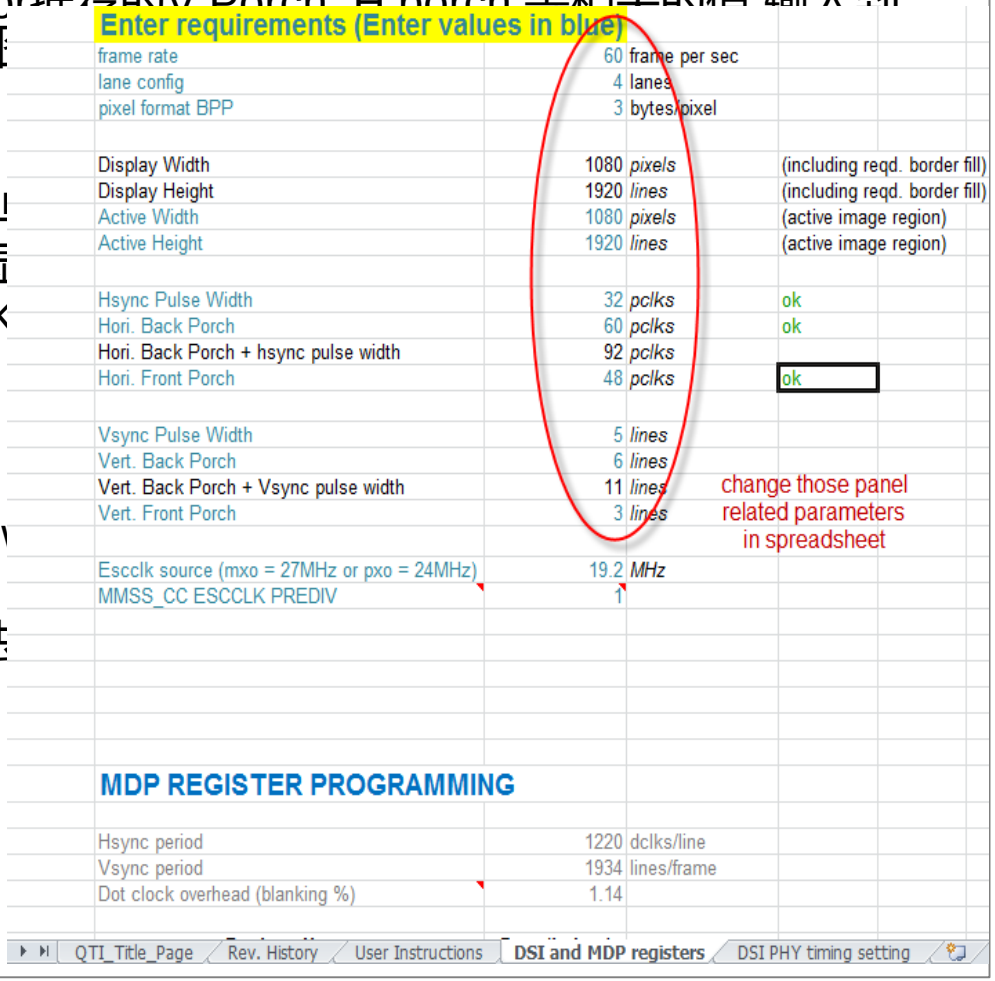
- 1. 首先，需要从https://downloads.cdmatech.com/ 网站，下载计算Timing的Excel 表格，具体的文档标号为： **80-NH713-1** DSI TIMING PARAMETERS USER INTERACTIVE SPREADSHEET.xlsm

- 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
- “CTRL + L”。

- 3. 切换到“DSI PHY timing setting” 工作
- 在“Check for T_CLK_ZERO”选项，会显
- “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 Windo
- 正常工作，请使用Win7 系统。
- b) 最好使用Microsoft Excel 2010版本去
- 此表格。

- 参考solution：00029225



显示屏驱动开发的调试技巧和检测点

通常情况下，如果kernel运行正常，如果参数设置正确，显示屏很容易点亮。但是你还是需要了解一些简单的调试技巧和检测点

1. 运行脚本：adb shell 检查android系统是否正常启动
2. 运行脚本：adb shell ls /dev/graphics/fb0, 检查系统是否成功创建fb0设备节点
3. 检查kmsg和logcat log，检查是否有明显的显示相关的错误
4. 运行脚本： adb shell cat /sys/kernel/debug/clk/mdss_byte0_clk/measure， 检查DSI 时钟是否正确. bitclock=measure*8 ,比如 720p的话，bitclock大概在450M左右。
5. 运行adb 命令： screencap –h /data/screencap.png，检查Surfaceflinger是否正常工作
6. 运行脚本：adb shell cat /sys/class/leds/lcd-backlight/brightness，检查显示屏背光是否正确设置。可以在暗室里面看背光打开是否正确，背光控制方法较多，需要认真检查。
7. 测量DSI相关的LDO电压， VDD/VDDIO/VDDA ，确保供电正常
8. 测量DSI 信号，确保波形正确
9. 阅读 SP80-NL239-5：MSM8916 SW Debug Manual和80-NA157-174：DSI MSM8974/MSM8x26 Android Programing Guide，了解更多DSI相关调试技巧

Android Display Kernel

如何动态debug 显示 Kernel 驱动

- 对于MDSS，主要包括下面几大模块 (参考文档：80-NL239-15)
  - Source Surface Processor(ViG pipe, RGB pipe, DMA pipe - SSPP)
  - Layer Mixer(LM)
  - Destination Surface Processor(DSPP)
  - Write-Back/Rotation(WB)
  - Display Interface

  具体的驱动文件在 //kernel/drivers/video/msm/mdss 目录下。

- 动态调试显示驱动，执行下面命令：
  - adb root
  - adb remount
  - adb shell
  - mount -t debugfs none /sys/kernel/debug/
  - root@android:/sys/kernel/debug # cd dynamic_debug
  - root@android:/sys/kernel/debug/ dynamic_debug # echo “file mdss_mdp_overlay.c +p” > control
  - root@android:/sys/kernel/debug/ dynamic_debug # echo “file mdss_mdp_ctl.c +p” > control
  - root@android:/sys/kernel/debug/ dynamic_debug # echo “file mdss_mdp_pipe.c +p” > control
  - root@android:/sys/kernel/debug/ dynamic_debug # echo “file mdss_mdp_util.c +p” > control
  - root@android:/sys/kernel/debug/ dynamic_debug # echo “file mdss_mdp_intf_video.c +p” > control

  另外，可以通过echo更多的文件进行调试。这样，通过 “cat /proc/kmsg”命令， 可以获取每个函数的调用流程， 以及参数的值，便于分析问题的关键所在。

- 参考Solution：00028591
  - 可以阅读 //kernel/Documentation/dynamic-debug-howto.txt

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 - 续一

- 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 - 续二

- 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。

如何调试LCD蓝屏(under-run)问题 - 续三

- 9. Enable ftrace 去debug
  - Enable Display Driver traces:
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_perf_update_bus/enable"
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_commit/enable"
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_sspp_set/enable"
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_sspp_change/enable"
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_video_underrun_done/enable"
    adb shell "echo 1 > /d/tracing/events/mdss/mdp_mixer_update/enable"

  - Enable Bus and Clock traces:
    adb shell "echo 1 > /d/tracing/events/power/clock_set_rate/enable"
    adb shell "echo 1 > /d/tracing/events/msm_bus/bus_update_request/enable"

  - 使能traces后，然后重现问题，并获取dump ftrace 的log
    adb shell cat /d/tracing/trace_pipe

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。

如何使用XLOG去调试 MDSS crash 问题

- 在开发过程中，会遇到MDSS crash的问题，可能是由于MDSS Ping-Pong timeout 引起的 system crash 或者 MDP IOMMOU page fault 导致system crash。
- 对于这类型的crash问题：
- 1. 首先，需要提供下面的logs
  - 完整的ramdump logs，对应的vmlinux 文件，以及所使用的具体基线的ID。把所有上面信息通过case提供给Qualcomm，Qualcomm内部分析后，会尽快您一个反馈。
- 2. 另外一种比较好的方法是 使能 XLOG，可以实时 dump所需要的信息。
  - Add debugfs entry for xlog to enable or disable logging and register dumping

    adb root，adb remount，adb shell

    mount -t debugfs none /d
  - Enable XLOG:

    echo 1 > /d/mdp/xlog/enable
  - Enable register dump:

    echo 1 > /d/mdp/xlog/reg_dump
  - Enable panic on error:

    echo 1 > /d/mdp/xlog/panic
  - 当crash产生后，然后执行下面命令，得到xlogs 和 registers dump，提供给Qualcomm分析。

    cat /d/mdp/xlog/dump > xlog_and_mdss_register.txt
- 具体XLOG的patch如下：

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=dcac801bd7b3b5a26627db1391c673c988770f4c

ESD solution问题

- 对于ESD，目前有很多规避方法，如下：
  - 让每一行数据都进入LP11 --- 对于质量比较好的panel，一般可以不需要Recovery机制
  - BTA check --- 出现BTA error时，调用blank/unblank 去recovery LCD
  - 读LCD的状态寄存器 --- 出现寄存器返回值错误时，调用blank/unblank 去recovery LCD

  注意：blank/unblank 是在HWC HAL的reset_panel 函数里进行recovery操作。

- 1. 如何让每一行都进行LP11，在LCD panel驱动的dtsi文件中 增加下面flags
  - qcom,mdss-dsi-hfp-power-mode
  - qcom,mdss-dsi-hbp-power-mode
  - qcom,mdss-dsi-hsa-power-mode

  这样，对于比较好的屏，不需要Recovery机制，但绝大多数情况，仍然需要Recovery机制。

- 2. Check BTA
  - 默认，每隔5秒，查看一次BTA

    #**define** STATUS_CHECK_INTERVAL_MS 5000
  - 下面是一个例子：

    --- a/arch/arm/boot/dts/qcom/dsi-panel-nt35590-720p-video.dtsi
    +++ b/arch/arm/boot/dts/qcom/dsi-panel-nt35590-720p-video.dtsi
    qcom,mdss-dsi-on-command-state = "dsi_lp_mode";
    qcom,mdss-dsi-off-command-state = "dsi_hs_mode";
    + **qcom,mdss-dsi-panel-status-check-mode = “bta_check”;** // 添加这行代码
    qcom,mdss-dsi-h-sync-pulse = <1>;
    qcom,mdss-dsi-traffic-mode = "burst_mode";
    qcom,mdss-dsi-bllp-eof-power-mode;

ESD solution问题 - 续一

- 3. 读LCD的状态寄存器
  - 在进行ESD测试时，通过读取LCD的状态寄存器的正确与否，来决定是否需要reset。
  - 下面是一个例子：

```
--- a/arch/arm/boot/dts/qcom/dsi-panel-innolux-720p-video.dtsi
+++ b/arch/arm/boot/dts/qcom/dsi-panel-innolux-720p-video.dtsi
qcom,mdss-dsi-on-command-state = "dsi_lp_mode";
qcom,mdss-dsi-off-command-state = "dsi_hs_mode";
+ qcom,mdss-dsi-panel-status-command = [06 01 00 01 05 00 02 0A 08]; // 发送的命令
+ qcom,mdss-dsi-panel-status-command-state = “dsi_lp_mode”; // 使用 LP mode去读取
+ qcom,mdss-dsi-panel-status-check-mode = "reg_read";
+ qcom,mdss-dsi-panel-status-value = <0x9c>;  // 寄存器的返回值
qcom,mdss-dsi-h-sync-pulse = <1>;
qcom,mdss-dsi-traffic-mode = "burst_mode";
```

- 4. 具体CAF链接如下(一般默认代码里面都有这个links)：
  - msm: mdss: Add support to enable ESD check through dtsi entry

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=6827717f961b10081e1ff849d56ae29377224e0a

  - ARM: dts: msm: enable ESD check for 8916 CDP with 720p and qrd-skuh

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=92e275c4aaf15ed1f1e421126ea44312d23076d4

  - msm: mdss: Add support for checking panel status through register read

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=5c687a4ca66f3b3728cadb871288ca25a73495b5

  - ARM: dts: msm: configure panel esd status check method for panels

https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=db3b47cb5a7f90acbd1c281de8327490adf9342a

ESD solution问题 - 续二

- 5. 如何设置EOT packet
  当读取寄存器时，不同的panel需求不一致：
  - 有些panel，不需要设置EOT packet，就可以成功地读取LCD的寄存器
  - 有些panel，必须需要设置EOT packet后，才能有效地读取LCD寄存器
- 5.1 在kernel display 去设置EOT packet
  - 方法一
    添加 qcom,mdss-dsi-tx-eot-append 到LCD panel 驱动的dtsi文件中
  - 方法二
    在mdss_dsi_host_init 函数中，以MSM8916为例：

```
data = 0;
if (pinfo->rx_eot_ignore)
        data |= BIT(4);
+++ pinfo->tx_eot_append = 1; // 添加这行代码
if (pinfo->tx_eot_append)
        data |= BIT(0);
MIPI_OUTP((ctrl_pdata->ctrl_base) + 0x00cc, data); /* DSI_EOT_PACKET_CTRL */
```

- 5.2 在LK display 去设置EOT packet
  - 在mdss_dsi_cmd_mode_config 函数中，以MSM8916举例如下：

```
writel(0x14000000, ctl_base + COMMAND_MODE_DMA_CTRL);
writel(0x10000000, ctl_base + MISR_CMD_CTRL);
+++ writel(0x1, ctl_base + EOT_PACKET_CTRL);   // 添加这个代码
```

- 6. 针对一些panel，即使LCD的状态寄存器返回值为 正确，但有时候也会出现下面一些logs：

```
197.805697:   <6> mdss_dsi_isr: ndx=0 isr=3310003
197.808853:   <6> mdss_dsi_fifo_status: status=44441000
197.813539:   <6> mdss_dsi_ack_err_status: status=1008000
197.818397:   <6> mdss_dsi_timeout_status: status=1
```

上面这些logs表明，100800 是来自LCD panel反馈，没有什么坏处。如果想继续研究，请联系LCD vendor FAE去check。

设置MIPI DSI Clock 和Data lanes的LP11在LCD HW Reset 之前

- 对于有些特殊的LCD panel，在LCD 硬件Reset之前，需要使能DSI CLK和Data lanes(LP11)。
- 具体实现：
- 在LCD panel dtsi 文件中，添加如下参数
  - qcom,mdss-dsi-lp11-init
  - qcom,mdss-dsi-init-delay-us = <50000>; // 根据不同的panel，可以修改

- 参考的CAF links代码 (以MSM8916为例)：
- 在kernel display，
  - msm: mdss: Enable hardware reset line based on panel setting
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=699406258812f7fa25dfd2a2ac60e86e731ef012
  - ARM: dts: msm: enable LP11 configuration for ssd2080m
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=67ea2af11374f4f395d86076cd9a365313445fda

- 在LK display，
  - platform: msm_shared: Support LP11 configuration
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=e2e6b71f1916fa3dd0a851a39b6fdf1c55a88fd8
  - dev: gcdb: Support post power API to handle LP11 panel config
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=c9611a9621f480c7096e7c03dc511fe8327586f2
  - platform: msm_shared: Support post power API in display
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=680f04f8fca25fcf9ba4b2d634b90b9359cc7947
  - platform: msm_shared: support panel pre-initialize function
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=4c7e37f0b3d797b613f87de15b2a6b8a4ee819d2
  - dev: gcdb: implement pre-initialize function
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=eb462f61fa747f773169fab8c3a27780cc0adbd3
  - dev: gcdb: enable LP11 configuration for ssd2080m
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=2f9e5228b02dddab8826c811b983e846a9e7dbb0
  - dev: gcdb: Remove lp11 init duplicate definition
  - https://www.codeaurora.org/cgit/quic/la/kernel/lk/commit/?h=LNX.LA.3.7&id=729aa97087a45aea3ed5876b50168454f21e5131

设置MIPI DSI Clock一直在HS mode

- 对于有些特殊的LCD panel，需要DSI CLK一直工作在HS模式。
- 下面两部分，分别介绍了在LK 和kernel 如何设置 DSI CLK一直为HS mode：
- 1. 在kernel display，mdss_dsi.c 文件中， mdss_dsi_on 函数：

```
+++ mipi->force_clk_lane_hs = 1;  // 强制HS mode
if (mipi->force_clk_lane_hs)
{
  u32 tmp;
  tmp = MIPI_INP((ctrl_pdata->ctrl_base) + 0xac);
  tmp |= (1<<28);
  MIPI_OUTP((ctrl_pdata->ctrl_base) + 0xac, tmp);
  wmb();
}
```

- 2. 在 LK display，mipi_dsi.c 文件中，mdss_dsi_host_init 函数：

```
writel(broadcast << 31 | EMBED_MODE1 << 28 | POWER_MODE2 << 26
        | PACK_TYPE1 << 24 | VC1 << 22 | DT1 << 16 | WC1,
         MIPI_DSI0_BASE + COMMAND_MODE_DMA_CTRL);

+++ writel(1 << 28, MIPI_DSI0_BASE + 0xac); // 强制HS mode，对应的寄存器，可参看8916的SWI手册
writel(lane_swap, MIPI_DSI0_BASE + LANE_SWAP_CTL);
```

调整MIPI DSI 驱动能力的问题

- 对于MIPI DSI 来说，有两种模式：
  - LP mode --- 从软件层面，可以调节
  - HS mode --- 从软件层面，不可以调节

- 从软件角度，LP mode可以调节，举例如下：
  - 对于MSM8916，在文件msm8916-mdss.dtsi中

    qcom,platform-strength-ctrl = [ff 06]; // [DSIPHY_STR_LP_N, DSIPHY_STR_LP_P]

    对应的寄存器为 0x01A98684 MDSS_DSI_0_PHY_DSIPHY_STRENGTH_CTRL_0

  - 芯片软件接口手册(Software Interface Manual)，具体文档号：80-NK807-2X，下载文档的网址为https://downloads.cdmatech.com。

- 从软件角度，HS mode不可以调节，DSI PHY HW自动地进行调整。如果，满足不了需求，请联系HW team去查看PCB的阻抗匹配等。

MIPI DSI PHY的 LDO mode 和 DC-DC mode

- 对于MIPI DSI PHY，有两种提供电源的方式：
  - LDO 模式
  - DC-DC 模式

- DC-DC模式：
  - 能提供很好的电源功效，但需要额外的电感元件，从而增加了HW PCB成本。

- LDO模式：
  - 与DC-DC模式相比，电源功效差些，但不需要额外的电感元件，从而节省了成本。

- 具体代码，参考CAF links (MSM8916为例)：
  - msm: mdss: enable LDO mode for DSI PHY regulator

  https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=18901f18865b10ff836cc85f546cff6ffc7852dc
  - msm: mdss: support LDO mode for DSI PHY regulator

  https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7&id=c91dc70dfa5b1477ce85c565b60844f44c117945

Frame Buffer 分配管理

- 目前，在 Android 系统上，Frame Buffer Target (FBT) 被应用，这也意味着 利用GPU 合成的layer 被绘制到 FBT 上，故 预留的物理连续 Framebuffer 内存可以被删除，这样对系统的memory优化有帮助。
  - 在正常的Android 应用场景，Frame Buffer Target(FBT) 被直接使用，且 FBT 所需内存 动态地从系统的ION memory heap中进行分配。
    当 apk 的layer 使用GPU 合成，那么 合成后的内容 是在FBT 上。
    当 apk 的layer 使用MDP 合成，那么 会使用overlay API，直接把显示的内容输出到LCD 屏幕上。
  - 在recovery mode 下，Qualcomm默认使用 overlay api。但在其它某些场景，如果 有的客户 需要 使用 FrameBuffer的 pan display API，那么，此时mmap 函数会被调用，从ION memory heap中动态分配内存， 直到这块 内存不再被使用，系统会自动释放。
- 比如，8974，8916，8939，8994，之前预留的8M的静态物理连续内存被删除，因为当前的显示驱动 overlay API 不再 使用 这块物理连续的内存。
- 为了对遗留平台的兼容支持，如果 用户 仍然要使用 FrameBuffer (pan display API)，那么这个块 memory 会通过 mmap 从系统的ION memory 中分配得到， 一直到 FB 的参考计数的count到0，这块内存被释放。
- 参考下面代码：
  - 在 dtsi 文件中，例如，8974 平台，参考下面的 CAF link。

    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/tree/arch/arm/boot/dts/qcom/msm8974-mdss.dtsi?h=msm-3.10

    mdss_fb0: qcom,mdss_fb_primary {
    cell-index = <0>;
    compatible = "qcom,mdss-fb";
    - qcom,memory-reservation-type = "EBI1";
    - qcom,memory-reservation-size = <0x800000>; // remove this 8M memory
    qcom,memblock-reserve = <0x03200000 0x01E00000>;
    };
- 对于8916，可以参看下面link：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/tree/arch/arm/boot/dts/qcom/msm8916-mdss.dtsi?h=msm-3.10

Frame Buffer 分配管理 - 续一

- 对于 连续显示功能，即continuous splash screen, 此时 显示内容 延续 LK display content，以 8916 为例，从下面的link，我们能看到：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/tree/arch/arm/boot/dts/qcom/msm8916-mdss.dtsi?h=msm-3.10

    mdss_fb0: qcom,mdss_fb_primary {
    cell-index = <0>;
    compatible = “qcom,mdss-fb”;
    qcom,memblock-reserve = <0x83200000 0xfa0000>;
- 针对 “qcom,memblock-reserve”，这块特殊的memory 主要目的为 连续显示使用， 在开机过程中，从 LK 过渡到 kernel 时，需要设置 qcom,cont-splash-enabled，一旦出现 Android Boot animation，此block 内存就会被释放。
- 具体CAF links
  - Remove FB memory allocation (***)
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=f1d88ed5d465d5ad3fc3fffe4b7bdbaf95cb84f2
  - Allocate FB memory during mmap call (***)
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=cd3b75f365dafcb2e3977b80890e0754a882069b
  - Add backward compatibility to FB memory (***)
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=24589b39babf0f0d9967c244f78047c76a01f2da
- 相关的 Fix changes：
  - mdss: mdp: update fix screen variable in mmap call
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=d5d7a9694bced89703b1340a238f0180bbdc63ba
  - mdss: fb: ensure that shared memory is available during probe
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=1e45a92c6dcbb79f5f56395489afb187535aa5ce
  - mdss: fb: update smem_len when user updates parameters
    https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=msm-3.10&id=b90090ee8dad738aae19fc4e60ccc01d7e2fdb78

参考Solution：00029644

Android Display HAL

UI Flickering 问题的调试

- 在Android系统中，会遇到 闪屏的问题，主要分为下面几大类：
  - 与 MDP 合成相关的 flickering
  - 与 GPU 相关的flickering
  - 与 framework/apk 本身相关的flickering

- 下面对其调试方法分别进行介绍

- 1. Disable HW overlay on settings apk, 此时，让所有的layers 使用GPU 进行合成。
  - 如果闪屏依然存在，需要查看GPU，framework, apk 相关领域。反之，需要查看MDP合成相关内容。

- 2. 让所有的layers 使用MDP 合成，请参看下面代码
  - 在 hardware/qcom/display/libhwcomposer/hwc_mdpcomp.cpp , MDPComp::tryFullFrame 函数

```
/* 注释掉下面代码
if(sIdleFallBack && !ctx->listStats[mDpy].secureUI) {
ALOGD_IF(isDebug(), "%s: Idle fallback dpy %d",__FUNCTION__, mDpy);
return false;
}
*/
```

- 3. 关闭混合 合成模式，即 MDP + GPU
  - 方法一：

```
adb shell setprop debug.mdpcomp.mixedmode.disable 1
adb shell stop
adb shell start
```

  - 方法二：

在 system/build.prop 文件中修改对应的flag，然后push到手机上

UI Flickering 问题的调试 - 续一

- 4. Disable PTOR(Peripheral Tiny Overlap Removal) 功能
  - 去查看flickering 问题 是否 与ptor 相关

    adb shell setprop persist.hwc.ptor.enable false
    adb shell stop
    adb shell start

- 5. Disable dirty-rect optimization for GPU
  - 去查看flickering 问题 是否 与 GPU 的dirty region 的优化 相关

    adb shell setprop debug.sf.gpu_comp_tiling 0
    adb shell stop
    adb shell start

- 6. Disable CABL 功能
  - 去查看flickering 问题 是否 与CABL 相关

    adb shell setprop ro.qualcomm.cabl 0
    adb shell stop
    adb shell start

- 7. 对于command mode panel来说，disable partial update function
  - 在panel driver dtsi 文件中， 删除下面flag，去查看 是否 与此功能相关

    qcom,partial-update-enabled

UI Flickering 问题的调试 - 续二

- 8. Dump apk 的layers，分类如下：
  - 分为 primary LCD display内容 和 external display内容
  - 分为 png 格式 和 raw 格式

- 对于LCD 主屏显示的layers dump 命令如下：

  adb shell setprop debug.sf.dump.enable true
  adb shell setprop debug.sf.dump.primary true
  adb shell stop
  adb shell start
  adb shell setprop debug.sf.dump 0 // Raw 格式，可以把video layer dump 下来，如果是png格式 (debug.sf.dump.png)，只能dump graphics layer.
  adb shell setprop debug.sf.dump 20 // 这里的是需要dump的帧数，根据需求，自行调整 // 去复现问题
  adb shell setprop debug.sf.dump 0 // 停止dump
  adb pull /data/sf*

  然后把dump layers 的数据pull到电脑上去查看。

- 对于 external display layers dump 命令如下：

  需要把上面命令中 setprop debug.sf.dump.primary 改为 debug.sf.dump.external 即可。

- 参考solution：00028590

Android Display Post Processing

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

QDCM 调试总结，适用于8916，8939，8994

- QDCM是Qualcomm_Display_Color_Mgmt的缩写。是一种用于LCD panel 校准，屏幕颜色调节的PC tool。通过QDCM可以使用MDSS 里面的复杂图像处理HW logic，对LCD屏幕上面的屏幕效果做复杂的调节和处理。

- QDCM目前还不可以任意下载，必须通过case ，高通显示技术支持团队帮助申请。在申请数日之后，邮箱里面会收到对应的下载链接。**80-NJ550-1** ， **80-NJ550-2** 两个用户手册，以及**72-NK466-8** QDCM安装文件。QDCM目前的最新版本为 3.3，可以支持 8916，8939，8994 。
  请认真阅读这两个用户手册，进行安装，调试，使用。

- QDCM 的使用，需要device SW的配合。Device上面的QIAG port必须打开，相关的系统文件和服务需要运行。高通默认的release build 上面一般包含了QDCM 使用所需要的文件，但是客户的ROM一般会有所删节。
  需要客户工程师保证以下文件的完整以及正常运行。
  /system/app/colorservice.apk
  /system/vendor/lib/libsd_sdk_display.so
  /system/etc/permissions/com.qti.snapdragon.sdk.display.xml
  /system/framework/com.qti.snapdragon.sdk.display.jar
  /system/bin/mm-pp-daemon
  /system/app/DisplaySDKSample.apk
  以上文件是QDCM save-on-target的相关文件。其中mm-pp-daemon开机会自动运行，在init.target.rc中启动
  这个服务进程。

QDCM 调试总结，适用于8916，8939，8994---cont

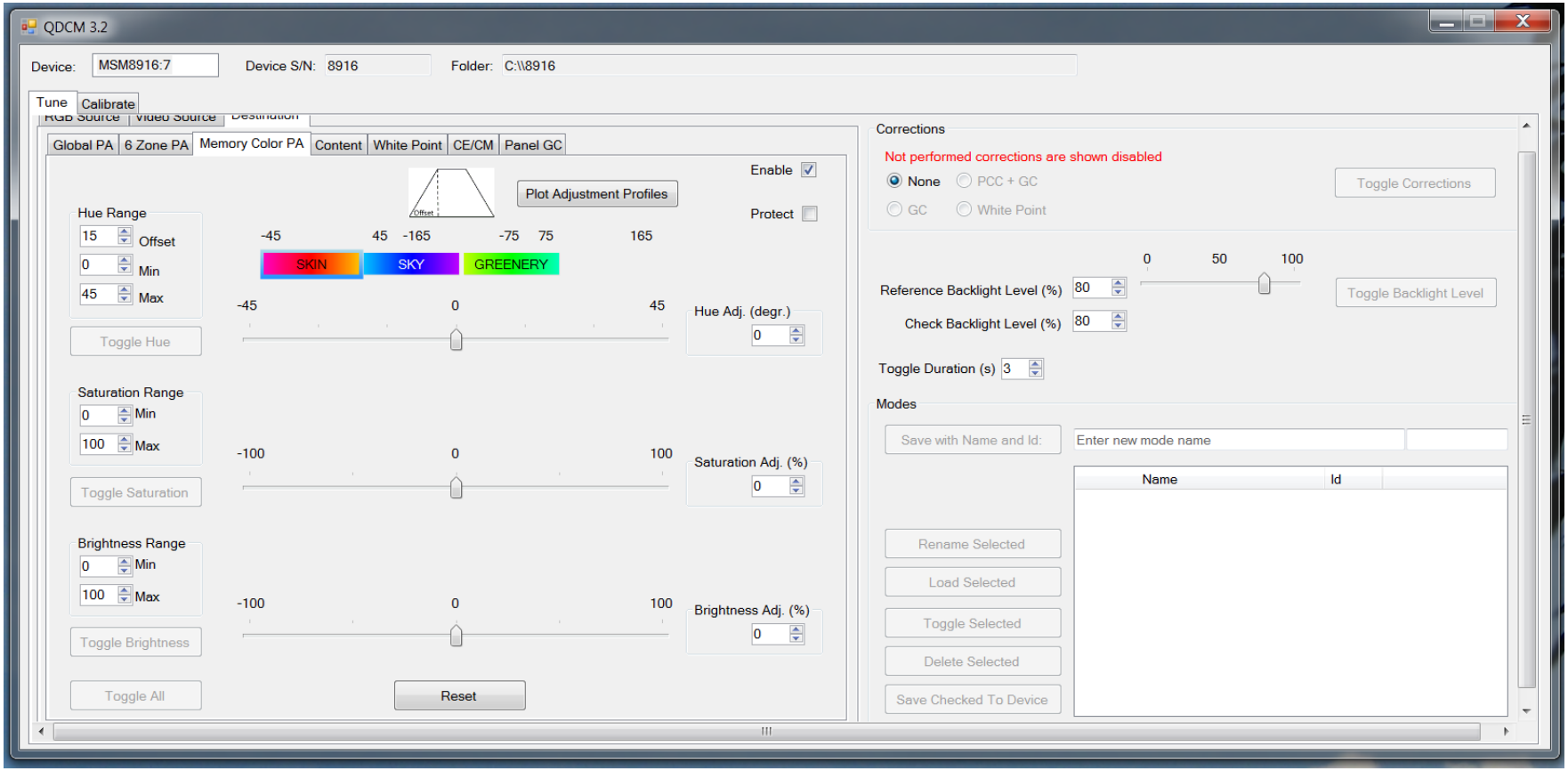
- 如果上述文件正常打包进入device ROM，那么在应用程序列表中可以看到DisplaySDKSample。点击启动该程序，其中可以选择简单色温调节和模式选择两个功能。
- DisplaySDKSample.apk是基于snapdragon SDK开发的，其源代码可以通过case的形式开放，其中提供了Java 层的调用API。
- 目前在 8916和8939平台，最高QDCM可以支持到单DSI FHD video模式以及command模式。

  目前在 8994平台，QDCM已经验证过了 双DSI，1440p video模式以及command模式。
- QDCM是一种较为复杂的颜色管理工具，建议客户在使用前，对色度学的基础知识进行温习。高通的显示技术支持团队，会努力为客户提供合理的建议。但是由于色彩管理，调节是较为主观的开发工作。最后的色彩调节程度必须由客户把握。
- 不同的平台，MDSS的硬件结构不尽相同，QDCM会自动对平台进行检测，开放相应的调试接口。

- QDCM属于高通显示后处理技术中的色彩管理方案，更多的显示后处理方案可以参考相应平台的显示培训文档。如 MSM89XX Linux Display Overview 等。
- 80-NJ164-1_B_Display_Postprocessing_Features_Tech_Notes是显示后处理相关平台软件的应用手册，可以作为开发参考。

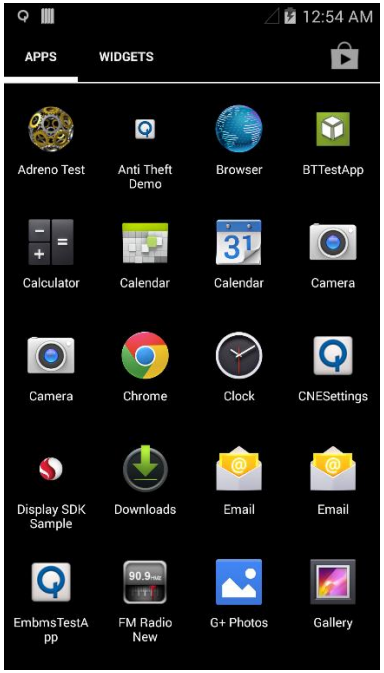
QDCM 调试总结，适用于8916，8939，8994---cont

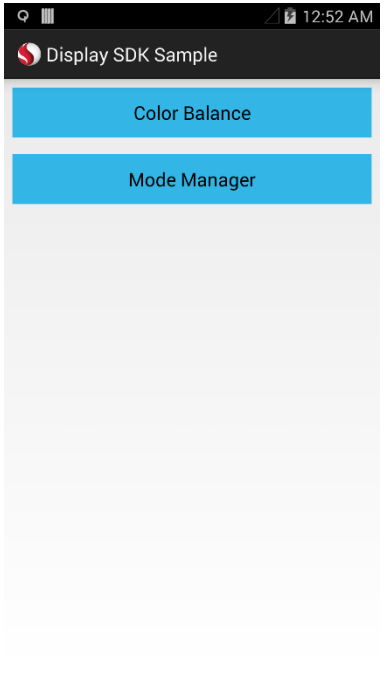
- QDCM 界面

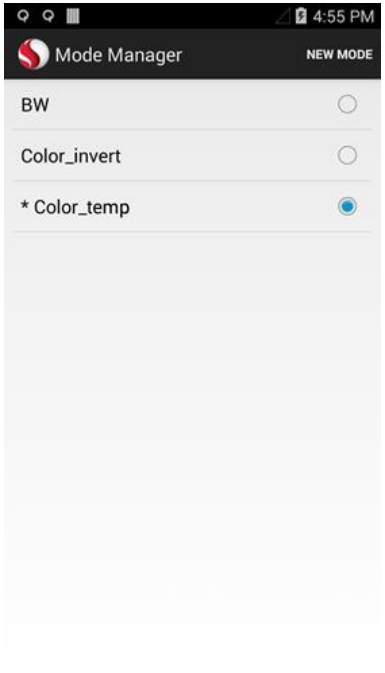


QDCM 调试总结，适用于8916，8939，8994---cont

- On-device display SDK sample 界面







QDCM Tool and Training Video

- QDCM tools and user manual guide:

1	**Qualcomm Display Color Management (QDCM) Tuning Tool, Ver. G**	**80-NJ550-1**
2	Qualcomm Display Color Management (QDCM) Calibration Tool, Ver. H	80-NJ550-2
3	Installer, Qualcomm Display Color Management (QDCM) Release 3.3	72-NK466-8

- Basic Display Tuning Training Video

No	Video Clip Title	DCN No
1	INSTALL QDCM TOOLS - VIDEO SIMPLIFIED CHINESE	VD80-NJ550-10SC
2	RUN DEPENDENCY CHECKER TOOL - VIDEO SIMPLIFIED CHINESE	VD80-NJ550-11SC
3	ADJUST WHITE COLOR TEMPERATURE - VIDEO SIMPLIFIED CHINESE	VD80-NJ550-12SC
4	ADJUST MEMORY COLORS - VIDEO SIMPLIFIED CHINESE	VD80-NJ550-13SC
5	USE PREDEFINED MODES - VIDEO SIMPLIFIED CHINESE	VD80-NJ550-14SC

- 上面的工具，文档，视频，需要提SR 到 Qualcomm 去申请。

Display SDK Sample 源代码

- 高通提供了Display SDK Sample apk的源代码，位于 vendor\qcom\proprietary\snapdragon-sdk\display\displaysdk_sample.

- 同时，提供了如何使用API的文档，如下：80-NR528-1_A_Display_Color_Managing_Target_Tools_Android_Overview.pdf
-
- 另外，你可以通过下面的网站下载 Display SDK Sample 源代码：https://developer.qualcomm.com/download/snapdragon-sdk-alpha-release.zip

如何编译xml文件到images中 – 参考solution: 00029913

- 使用QDCM tools 生成的pp_calib_data%s.xml 文件，需要使用下面的patch，把xml编译到image中，重新烧写images后，xml文件会出现在/persist 目录下面。

- vendor/qcom/proprietary/mm-core/display/qdcm/

```
diff --git a/display/qdcm/Android.mk b/display/qdcm/Android.mk
new file mode 100644
index 0000000..06b4c4e
--- /dev/null
+++ b/display/qdcm/Android.mk
+this_folder := $(call my-dir)
+LOCAL_PATH := $(this_folder)

+include $(CLEAR_VARS)
+LOCAL_MODULE := pp_calib_data%s.xml
+LOCAL_MODULE_CLASS := DATA
+LOCAL_SRC_FILES := $(LOCAL_MODULE)
+LOCAL_MODULE_TAGS := optional
+LOCAL_MODULE_PATH := $(PRODUCT_OUT)/persist
+LOCAL_MODULE_OWNER := qcom
+include $(BUILD_PREBUILT)
```

Android Display Critical Issues

8939 LA 1.0 1.0.35 release概述

- 高通11月21日发布了最新的8939 LA 1.0 CS5 - 1.0.35 release，该版本中包含了许多重要的补丁，同时该版本对系统性能和功耗有进一步优化，我们强烈建议客户升级到这次最新版本
- 为方便客户理解这次版本中的重要多媒体相关的改动，高通中国支持团队对这次基线中的一些重要和关键的多媒体改动进行了总结，方便客户理解。
- 如果客户想知道更完整的CR list， 请参考每个软件模块根目录下面的fixed_crs.xls
- 下面的篇幅对多媒体子模块的关键改动进行逐一介绍

[CR#708324] 连接WFD 和 开启 ESD check，系统会Crash

- 描述：
  - 如果开启了DSI ESD recovery thread，再连接 WFD，系统容易进入crash.

- 补丁：
  - https://git.quicinc.com/?p=kernel/msm-3.10.git;a=patch;h=df74e7e394af54c05436fcc4192464c4b3d980e6

- 补充说明：
  - 主要原因是MDSS 驱动内部的软件同步错误，导致指针无效.

[CR#752548] MDSS _sync_tlb Timeout Issue

- 描述：
  - 对于command mode panel, 在进行reboot压力测试时，MDP fence timeout 造成系统crash。

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=57c4488793a769db05bfc15ffc86bc7be0a3c432
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=d146a109602b19a0e42b30a320ea4f5dd4443b51
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=346c5ccfd95b37f29378e1498d85b981f087cfb5

- 补充说明：
  - 对于此问题，一般会出现下面的log

```
[   24.491291] __mdss_fb_sync_buf_done_callback: mdp-fence: frame timeout
[   24.555397] mdss_mdp_cmd_pingpong_done: too many kickoffs=1!
[   25.151791] mdss_dsi_cmd_mdp_busy: timeout error
[   29.683529] msm_iommu_v1: Timed out waiting for TLB SYNC to complete for apps_iommu
[   29.690284] msm_iommu_v1: Value of SMMU_IMPLDEF_MICRO_MMU_CTRL = 0x0
[   29.696589] Unhandled fault: external abort on non-linefetch (0x008) at 0xdf804200
[   29.704110] Internal error: : 8 [#1] PREEMPT SMP ARM
```

[CR#746099] Tearing Effect Issue on Command Mode Panel

- 描述：
  - 使用command mode panel 时，在kernel 阶段，无法检测到TE 信号的中断，从而照成系统死机

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=0e5fb6ab0acbc39260dba8b68cf1cf36636782b4

- 补充说明：
  - 此问题根源是 在continuous splash screen 阶段，没有配置HW TE引起的。
  - 如果disable continuous splash screen 后，在kernel 启动时，会配置HW TE，此问题不复存在。
  - 可以通过在mdss_mdp_cmd_readptr_done 函数中添加log来查看 中断是否有效。

[CR#749413] 使能Partial Update后，播放视频会出现Crash

- 描述：
  - 在command mode panel上，使能Partial Update后，播放视频时会照成系统死机。

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=52479d426d81ff641a8c8ec35eaf22a0ff3c8eae

- 补充说明：
  - 如何使能Partial Update，需要 添加下面的flags在LCD panel driver的 dtsi 文件中

    qcom,partial-update-enabled; //Enable Partial Update

    qcom,panel-roi-alignment = <4 4 4 4 4 4>;

    // 这些值来源于Panel vendor，需要联系LCD FAE进行确认。
  - 具体flags的解释说明，请参考 kernel/Documentation/devicetree/bindings/fb/mdss-dsi-panel.txt

[CR#729918] BUG_ON() Check Getting Triggered for No Source Buffer

- 描述：
  - MDP的空指针引起 Kernel Panic.

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=5583bdbd06b97a65ed2d6474b7018f029b0f5501
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=1c2712cdbbaa23d910e89ca36b5f08cdd2444426

- 补充说明：
  - 无

[CR#722989] QDCM Per Panel Save-on-Target SW support

- 描述：
  - 对于 每个项目 可能会用到 多款LCD panel，当连接 QDCM tools 时，通过 Panel Name 来自动检测。

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7.3_rb1.3&id=5550b9d8ecc8daf31642b580af0f2f129cfdb432
  - 私有的lib库，如libmm-qdcm.so，是以bin文件发布给客户，在发布的版本中默认被包括。

- 补充说明：
  - 对于如何使用QDCM Per Panel Save to Target, 请参考 solution: 00029913

[CR#717774] ULPS mode on Video Mode Panel during Suspend

- 描述：
  - 根据不同LCD panel 厂商的需求，在使用video mode panel时，在suspend时，需要进入ULPS mode，这样可以解决Panel 漏电流。

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7_master&id=6b7428b2b3d435293803e119c3475eedc278e68e
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7_master&id=9e0bdaee5ad29a1998212068230 6ed453a354ac8
  - https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?h=LNX.LA.3.7_master&id=317b9679eebfddbe9e6e1f870ecc491ae0a50eed

- 补充说明：
  - 如果想使能 ULPS mode，需要在LCD panel driver dtsi 文件中 添加

    + qcom,suspend-ulps-enabled;

[CR#728009] Blending Output Difference between GPU and MDP

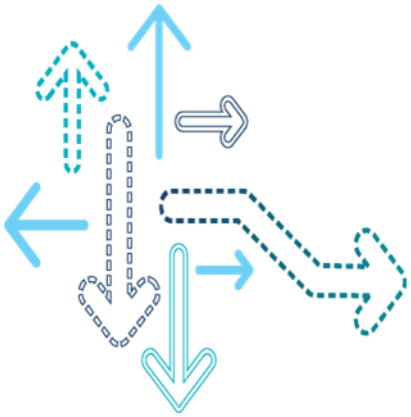
- 描述：
  - 由于MDP 和 GPU 合成方式输出的区别，在某些场景会出现闪烁的现象。

- 补丁：
  - https://www.codeaurora.org/cgit/quic/la/platform/frameworks/native/commit/?h=LNX.LA.3.7_master&id=29114a07283cf5edf026b53d598e3dd241bf10d5
  - https://www.codeaurora.org/cgit/quic/la/platform/hardware/qcom/display/commit/?h=LNX.LA.3.7_master&id=308cb9766c5db41b4eea2060e77e97aca14a8883
  - https://www.codeaurora.org/cgit/quic/la/platform/hardware/qcom/display/commit/?h=LNX.LA.3.7_master&id=a86edc1452c43252db548d0c7a428276c9594ac2

- 补充说明：
  - 对于flickering 问题的调试， 首先 disable HW overlay 去查看是否存在。
    Settings App → Developer Options → Disable HW overlay

References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*MSM8916 LA Display Overview*	80-NL239-15
Q3	*Linux Android Display Driver Porting Guide*	80-NN766-1
Q4	*Display GCDB XML Entries*	80-BA103-1
Q5	*MSM8916 SW Debug Manual*	SP80-NL239-5
Q6	*DSI MSM8974/MSM8x26 Android Programing Guide*	80-NA157-174
Q7	*MSM8916 LA Graphics Overview*	*80-NL239-16*
Standards		
S1		
Resources		
R1		



Questions?

https://support.cdmatech.com