

---

# 高通多媒体技术期刊 20150325

---



Qualcomm Technologies, Inc.

Confidential and Proprietary – Qualcomm Technologies, Inc.

机密和专有信息——高通技术股份有限公司



# Confidential and Proprietary – Qualcomm Technologies, Inc.

---

## Confidential and Proprietary – Qualcomm Technologies, Inc.

**NO PUBLIC DISCLOSURE PERMITTED:** Please report postings of this document on public servers or web sites to: [DocCtrlAgent@qualcomm.com](mailto:DocCtrlAgent@qualcomm.com). **禁止公开：**如在公共服务器或网站上发现本文档，请报告至：[DocCtrlAgent@qualcomm.com](mailto:DocCtrlAgent@qualcomm.com).

**Restricted Distribution:** Not to be distributed to anyone who is not an employee of either Qualcomm or its affiliated without the express approval of Qualcomm's Configuration Management. **限制分发：**未经高通配置管理部门的明示批准，不得发布给任何非高通或高通附属及关联公司员工的人。 Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc. 未经高通技术股份有限公司明示的书面允许，不得使用、复印、复制、或修改全部或部分文档，不得以任何形式向他人透露其内容。

The user of this documentation acknowledges and agrees that any Chinese text and/or translation herein shall be for reference purposes only and that in the event of any conflict between the English text and/or version and the Chinese text and/or version, the English text and/or version shall be controlling. 本文档的用户知悉并同意中文文本和/或翻译仅供参考之目的，如英文文本和/或版本和中文文本和/或版本之间存在冲突，以英文文本和/或版本为准。

This document contains confidential and proprietary information and must be shredded when discarded. 未经高通明示的书面允许，不得使用、复印、复制全部或部分文档，不得以任何形式向他人透露其内容。本文档含有高通机密和专有信息，丢弃时必须粉碎销毁。

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis. 高通保留未经通知即修改本档中提及的产品或信息的权利。本公司对使用或应用本文档所产生的直接或间接损失概不负责。本文档中的信息为基于现状所提供，使用风险由用户自行承担。

Qualcomm is a trademark of QUALCOMM Incorporated, registered in the United States and other countries. All QUALCOMM Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners. Qualcomm是高通公司在美国及其它国家注册的商标。所有高通公司的商标皆获得使用许可。其它产品和品牌名称可能为其各自所有者的商标或注册商标。

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited. 本文档及所含技术资料可能受美国和国际出口、再出口或转移出口法律的 限制。严禁违反或偏离美国和国际的相关法律。

**Qualcomm Technologies, Inc. 5775 Morehouse Drive San Diego, CA 92121 U.S.A.**

高通技术股份有限公司，美国加利福尼亚州圣地亚哥市莫豪斯路 5775 号，邮编 92121

# Revision History

---

Revision	Date	Description
A	Mar 2015	Initial release

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

# Contents

---

- Display
  - CABL, SVI简介及文档
  - AD 简介及评估方法
  - CABL调试总结
  - SVI调试总结
  - AD调试总结
- Audio
  - Audio Common Issues



---

# Display

---

# 高通显示(Display)后处理技术概述

- 显示后处理技术, display post processing technique , 是高通提供的对显示图像数据进行处理的综合解决方案。
- 显示后处理技术大致可以分为Content based和Non-content based两大类。
  - QDCM 和 QDCM mobile (on-device tuning ) 以及更早期的Picture adjustments , 都属于Non-content based技术。这类技术, 更多的是针对 LCD panel和用户主观喜好做全局性的色彩调节和各种图像处理。之前的多媒体期刊display部分, 对此做了若干的详细介绍。
  - SVI , AD , CABL 则属于 Content based的显示后处理技术。
    - SVI - Sunlight Visibility Improvement 低端芯片上日光下显示增强解决方案
    - AD - Assertive Display 中高端芯片上日光下显示增强解决方案
    - CABL - Content Adaptive Backlight 基于内容的自动背光调整
    - 此类显示处理技术会逐帧去分析显示图像内容, 环境光 ( ALS ) 和当前背光设定也会做为处理技术单元的 输入。
    - 此类技术主要的目的, 在于对显示内容数据本身做增强, 同时也可以通过降低背光, 达到降低功耗的目的。
    - 此类技术的本质, 在于通过图像处理, 增加图像的视觉对比度, 从而可以在降低背光时, 不损失显示屏幕的主观视觉感受 ( 对比度 ) 。
    - 本期多媒体期刊Display部分, 主要对此类技术的开发调试, 做更为详细的介绍。
    - 此期刊介绍的技术, 和partial update技术冲突, 如果是command mode LCD panel , 请关闭 partial update , 需要删除LCD panel dtsi file中的“qcom,partial-update-enabled”。

# CABL, SVI简介及文档

---

- CABL属于免费技术，主要用于降低功耗。基本原理---在降低背光的同时，增加图像亮度/对比度，使用户感觉不到视觉效果不同，从而达到省电的效果。
- LCD panel侧的同类技术一般称之为CABC。
- SVI属于收费技术，如果对此项技术感兴趣，需要联系高通的marketing team，签订相应的license，然后开始技术评估，如果对效果满意，就可以开始商用的流程。
  - SVI的基本原理，通过判断当前的环境光，分析当前帧的图像数据，对图像数据做增强。
- CABL+ SVI可以同时工作。在增加显示效果的同时降低功耗（背光）。
  - 环境光很亮的情况下（室外,e.g., >5000lux），不建议打开CABL降低背光。具体的配置方案，需要综合Android auto-brightness等背光方案，统一考虑。
- SVI和CABL的技术文档如下。
  - [80-NP906-1 Sunlight Visibility Improvement Overview](#)
  - 80-NP906-2\_SVI\_OEM\_Design\_Guide (不能直接下载，OEM签订相关license后，CE帮助下载)
  - [80-NP952-1 Content Adaptive Backlight OEM Design Guide](#)

# AD 简介及评估方法

- AD属于收费技术，如果对此项技术感兴趣，需要联系第三方的Apical Inc，签订相应的license，然后开始技术评估，如果对效果满意，就可以开始商用的流程。AD的文档，请联系Apical Inc. 联系人 Haijun Guo [haijun@apical.co.uk](mailto:haijun@apical.co.uk)，Judd Heape [judd@apical.co.uk](mailto:judd@apical.co.uk)
  - AD的基本原理，通过判断当前的环境光，分析当前帧的图像数据，对图像数据做增强，也可以根据目前的背光值做调整降低功耗（在图像增强之后，达到同样的显示效果）。
  - 增加强环境光照（比如日光）下的显示效果
  - 通过一次性等比调整背光，来降低功耗。
  - AD和SVI的区别在于，SVI是软件解决方案，AD是硬件解决方案，高通在中高端平台支持AD, 而低端平台支持SVI
- CABL+ AD可以同时工作。
  - CABL和AD 都是通过降低背光来减少功耗。区别在于CABL根据内容逐帧调节背光等级，而 AD根据目前的背光值做一次性的等比降低。
  - CABL和AD对背光的处理可以同时进行，默认的软件会处理相应的叠加效应。
- CABC + AD可以同时工作。
- QDCM + AD + CABL(CABC)可以同时工作。一般建议先完成QDCM的tuning，之后CABL及AD分别tuning，最后集成到一起。



# CABL调试总结

- CABL在高通默认的baseline 当中是打开的。在进行LCD panel 自身的调试的时候，需要关闭CABL。  
/system/build.prop  
# System property for cabl  
ro.qualcomm.cabl=0-----关闭CABL。  
ro.qualcomm.cabl=1-----8974 ,8084, 8x26以及更老的平台的默认设置。  
ro.qualcomm.cabl=2-----8939,8994等新平台的默认设置。  
一般来说，不需要改变默认设置。
- mm-pp-daemon 对CABL 进行各种控制操作，如果是L release，需要考虑 sepolicy 的访问权限问题。  
adb shell start/stop ppd可以对 mm-pp-daemon做 重启操作。  
init.target.rc 中，启动mm-pp-daemon service，使用pps socket接受各种控制command  
service ppd /system/bin/mm-pp-daemon  
class late\_start  
disabled  
user system  
socket pps stream 0660 system system  
group system graphics

## CABL调试总结 - cont.

- 可以在build.prop中设置各种prop来，抓取更多的logs。注意，在测量电量时候要把log 都关掉。在抓取logs时，最好用手滑动桌面，或者播放视频。

debug.listener.logs=1 mm-pp-daemon command相关的log开关。

debug.aba.logs=1,2,3 ,越高logs越多，debug时，建议开到3.

debug.cabl.logs=1,2越高logs越多，debug时，建议开到2.

adb shell logcat -v threadtime

adb shell "echo -n 'file mdss\_fb.c +tp'> /d/dynamic\_debug/control"

adb shell "echo -n 'file mdss\_mdp\_pp.c +tp'> /d/dynamic\_debug/control"

adb shell cat /proc/kmsg

- CABL启动之后，按照以上方法得到的kmsg(播放视频或滑动桌面)，如果可以看见

mdss\_fb\_scale\_bl: input = 1275, scale = 820

mdss\_fb\_scale\_bl: input = 1275, scale = 828

- 其中的scale的数值动态变化，那么CABL就已经生效了。由于CABL以不改变视觉效果为目标，肉眼无法观察CABL是否在正确运行。通过看kmsg的方法是可靠的。
- 如果LCD panel的Gamma curve较为接近Gamma 2.2，实际上CABL基本不用进行效果的tuning，可以直接使用。由于CABL在高通的SW release里面是自动生效的，在您完成对LCD panel的tuning之后(达到标准2.2)，基本就可以直接使用了。
- 如果无法把LCD panel的Gamma 调整为2.2，或者有特殊的光学，色度学指标上面的要求，请联系高通显示技术支持团队进行支持。

## CABL调试总结 - cont.

- CABL的参数在xml file内设置（详见[80-NP952-1](#) ,CABL app note）。一般不用修改。如果不做修改的话，默认的参数也可以工作。  
下面表内的参数，可以根据具体的设备背光配置修改。

Parameter Name	Description	Valid Range	Default Value	Comments
CABLBackLightMaxValue	The maximum brightness level the OS supports.	[0,255]	255	可以根据LCD panel dtsi file里面的最大背光来设置。 qcom,mdss-dsi-bl-max-level。如果panel file没有设置此项，此处不需要更改
CABLBackLightMinValue	The min brightness level the OS supports.	[0, CABLBackLightMaxValue]	0	可以根据LCD panel dtsi file里面的最小背光来设置。 qcom,mdss-dsi-bl-min-level。如果panel file没有设置此项，此处不需要更改。
CABLBackLightThreshold	The Normalized threshold below which CABL does not reduce its backlight level further	[0, 1023]	124	配置CABL在低背光时，不降低背光等级。因为在低背光时，动态改变背光，非常容易引起闪烁。这个值，需要通过实验设定。

# SVI调试总结

---

- SVI 的评估，请先联系高通的marketing team，在签订了相应的license之后，可以进行评估。
- 取得相关license后，请下载80-NP906-1和80-NP906-2，仔细阅读，然后进行bring-up和调试。
- 如何抓log，注意，在测量电量的时候要把log 都关掉。
  - `debug.svi.logs = <0,1,2>`，越高越详细，建议开启2。
  - `debug.refresh.logs=1`
  - `debug.listener.logs=1`
  - `debug.als.logs=1`
  - `adb shell logcat -v threadtime.`
  - `adb shell "echo -n 'file mdss_fb.c +tp'> /d/dynamic_debug/control"`
  - `adb shell "echo -n 'file mdss_mdp_pp.c +tp'> /d/dynamic_debug/control"`
  - `adb shell cat /proc/kmsg`

## SVI调试总结 - cont.

### 关于SVI tuning的一般建议

1. 完成LCD panel本身的tuning。LCD panel vendor提供调试帮助，以及使用QDCM调试。使LCD panel的gamma接近标准的2.2
2. 完成ALS sensor本身的tuning。请保证ALS sensor 的数值是正确的。建议和sensor vendor同步工作。ALS sensor 的正确运行是SVI工作的前提。
3. 如果LCD panel与Gamma 2.2差距较大，可以参考80-NP906-2 3.4 Backlight response linearization，进行 LCD panel 背光的校正。
4. （可选）参考80-NP906-2 3.3 Panel reflectance-related calibration 使用高通提供的工具软件（请联系高通技术支持团队）确定相应的 Brightness Factor Range。

IndoorMaxLuxLevel	The lux value to differentiate indoor and outdoor boosting strength, which can be measured during ALS calibration. Calibration should be done in a bright indoor environment.	Range: [300 – OutdoorMaxLuxLevel].	Default is 2000.	一般范围在 2000 - 5000 lux。此数值可以根据设备的target market的典型环境光数值有所调整。比如，在热带地区可以考虑适当提高。
-------------------	---	------------------------------------	------------------	---

## SVI调试总结 - cont.

- ALS sensor 的SW API可以分为
  - Android default API (native API) , ro.qcom.svi.sensortype=2
  - Qualcomm solution API (default value of AD), ro.qcom.svi.sensortype=3 (default), 如果不设置此prop , 那么3就是default value。
  - Dummy sensor API (软件模拟ALS , 只用于debug) ro.qcom.svi.sensortype=1

需要和sensor工程师确认项目的sensor配置情况 , 进而确定上述prop数值。

当怀疑是由于ALS input不正确而引入的问题 , dummy sensor可以用来debug。

1) push file dummy\_als.txt to /data/

2) ro.qcom.svi.sensortype=1 in build.prop

3) ro.qcom.lightsensorsim.input=/data/dummy\_als.txt

4) reboot and adb shell setenforce 0

- dummy\_als.txt 示例。实际上此文件格式非常简单 , 每行就是一个单独的ALS模拟输入。在文件中 , 需要每个值都单独成一行。

10

110

210

XX

# AD调试总结

- AD 的评估，请先联系第三方的Apical Team，在签订了相应的license之后，可以进行评估。
- Apical Team 会提供tuning/calibration apk，以及相应文档。
  - 在Apical team 提供了APK之后，需要通过SF SR的形式，向高通CE team确认SW baseline是否缺少相关的patch
  - Apical Team开始tuning/calibration和power测量。
  - 如果遇到技术问题，需要通过SF SR的形式，和高通CE team一起debug相应的问题。
- 抓log方法：**注意，在测量电量的时候要把log 都关掉。**

debug.listener.logs=1 // socket通信相关的log。

debug.ad.logs=1 // AD运行相关的log。

debug.als.logs=1 // ALS sensor读值相关的log。

debug.refresh.logs=1 // AD frame刷新相关的log。

adb shell logcat -v threadtime.

adb shell "echo -n 'file mdss\_fb.c +tp'> /d/dynamic\_debug/control"

adb shell "echo -n 'file mdss\_mdp\_pp.c +tp'> /d/dynamic\_debug/control"

adb shell cat /proc/kmsg

## AD调试总结 - cont.

- 在Apical team提供AD enable的方法之后，按照以上方法，如果在kmsg中看见
  - pp\_ad\_calc\_worker: calc\_str = 20, calc\_itr 193
  - pp\_ad\_calc\_worker: calc\_str = 19, calc\_itr 192calc\_str = 19 随着环境光的变化而变化，我们就可以确定AD在运行了。
- 建议在AD 调试之前完成
  1. ALS sensor的调试。
  2. LCD panel的调试。
- ALS sensor 的SW API可以分为
  - Android default API (native API) , ro.qcom.svi.sensortype=2
  - Qualcomm solution API (default value of AD), ro.qcom.svi.sensortype=3 (default), 如果不设置此prop，那么3就是default value。
  - Dummy sensor API (软件模拟ALS，只用于debug) ro.qcom.svi.sensortype=1需要和sensor工程师确认项目的sensor配置情况，进而确定上述prop数值。

当怀疑是由于ALS input不正确而引入的问题，dummy sensor可以用来debug。

- 1) push file dummy\_als.txt to /data/
- 2) ro.qcom.ad.sensortype=1 in build.prop
- 3) ro.qcom.lightsensorsim.input=/data/dummy\_als.txt
- 4) reboot and adb shell setenforce 0



## AD调试总结 - cont.

---

- dummy\_als.txt 示例。实际上此文件格式非常简单，每行就是一个单独的ALS模拟输入。在文件中，需要每个值都单独成一行。

```
10
110
210
310
410
510
Xx
Xx
Xx
xx
```



---

# Audio

---

# Audio common issues ( 1 )

---

- 描述：Mp3 ID3信息显示乱码
- 复现步骤：进入音乐播放器，查看音乐文件列表，有些文件显示乱码，但是在**KK基线没有该问题**（不是所有的乱码问题都可以解决，因为片源ID3的编码格式差异）
- 基线：MSM8909.LA.1.0
- CR：794923
- 代码修改：
  - 修改代码在目录vendor/qcom-proprietary/ship/mm-parser，因为是vendor目录下的修改，需要在chipcode看是基线否包含改CR。如果有直接合入；如果没有可以申请SBA.

## Audio common issues ( 2 )

---

- 描述：音频播放参数不生效
- 复现步骤：
  - 1 Play music.
  - 2 Insert headset.
  - 3 Plug headset(music is pause) and click play button quickly.
  - 4 We can see headset parameters in QACT (It should be speaker parameters, you can make the codec gain as the comparing parameter.)
- 基线：MSM8909.LA.1.1/ AP: LA.BR.1.2.3
- CR：805996
- 代码修改：
  - <https://www.codeaurora.org/cgit/quic/la/platform/hardware/qcom/audio/commit/?id=ad7d7b5d4a00309782b94a6ec77d79f8ec173c87>

# Audio common issues ( 3 )

---

- 描述：铃音、消息通知音播放卡顿
- 复现步骤：播放铃音、消息通知音，概率性发生卡顿
- 基线：跟芯片平台无关
- CR：无
- 代码修改：
  - <https://android.googlesource.com/platform/frameworks/av/+620208dc0bbd7a0792702df3ab08800fdad60cec%5E!/>
  - 如果是在L基线上，基于以上修改还需要额外修改：
    - -int ret = \_\_futex\_syscall4(&mCblk->mFutex,
    - - FUTEX\_WAIT,
    - - old & ~CBLK\_FUTEX\_WAKE, &ts);
  
    - +#include <sys/syscall.h>
    - +int ret = syscall(\_\_NR\_futex, &cblk->mFutex,
    - + FUTEX\_WAIT,
    - + old & ~CBLK\_FUTEX\_WAKE, ts);

# Audio common issues ( 4 )

---

- 描述：概率性耳机不识别，因为纠错检测时寄存器0x159为0x10.
- 复现步骤和现象：
  - Insert headset then plug out repeatedly, headset was not detected with low probability.
  - 检测kernel log有如下打印
    - 11050 <7>[ 2213.888883] wcd\_correct\_swch\_plug: enter
    - 11075 <7>[ 2214.097806] **wcd\_correct\_swch\_plug: stop requested: 0**
    - 11091 <7>[ 2214.097946] wcd\_correct\_swch\_plug: leave
  - 0x159 register is 0x10
- 基线：MSM8916/39 KK或L基线
- CR：无
- 代码修改：
  - <https://www.codeaurora.org/cgit/quic/la/kernel/msm-3.10/commit/?id=e99d21039935fd760cad145f1de3be14832d01aa>

# Audio common issues ( 5.1 )

- 描述：There is a memory leak when play the music
- 复现步骤和现象：
  - 1 Start playing music from music player app
  - 2 Skip to next song
  - 3 Keep repeating #2
  - 4 Run the command "adb shell dumpsys meminfo mediaserver"
  - Heap memory keeps growing
- 基线：8994.LA.1.1 / LA.BF64.1.1
- CR：778868
- 代码修改：
  - 1). diff --git a/services/audiopolicy/AudioPolicyService.cpp b/services/audiopolicy/AudioPolicyService.cpp
  - index 8a9abc9..0955e10 100644 (file)
  - --- a/services/audiopolicy/AudioPolicyService.cpp
  - +++ b/services/audiopolicy/AudioPolicyService.cpp
  - @@ -907,8 +907,10 @@ void AudioPolicyService::AudioCommandThread::insertCommand\_I(sp<AudioCommand>& c
  - }
  - removedCommands.clear();
  - 
  - - // Disable wait for status if delay is not 0
  - - if (delayMs != 0) {
  - + // Disable wait for status if delay is not 0.
  - + // Except for create audio patch command because the returned patch handle
  - + // is needed by audio policy manager
  - + if (delayMs != 0 && command->mCommand != CREATE\_AUDIO\_PATCH) {
  - command->mWaitStatus = false;
  - }

# Audio common issues ( 5.2 )

## ■ 代码修改：

- 2). diff --git a/services/audioflinger/PatchPanel.cpp b/services/audioflinger/PatchPanel.cpp
- index ae7d8af..fa74e83 100644 (file)
- --- a/services/audioflinger/PatchPanel.cpp
- +++ b/services/audioflinger/PatchPanel.cpp
- @@ -166,7 +166,9 @@ status\_t AudioFlinger::PatchPanel::createAudioPatch(const struct audio\_patch \*pa
- if (\*handle == mPatches[index]->mHandle) {
- ALOGV("createAudioPatch() removing patch handle %d", \*handle);
- halHandle = mPatches[index]->mHalHandle;
- +         Patch \*removePatch = mPatches[index];
- mPatches.removeAt(index);
- +         delete removePatch;
- break;
- }
- }



---

## Questions?

<https://support.cdmatech.com>

