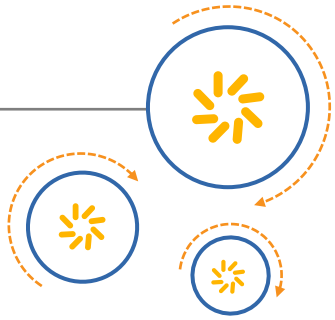
Qualcomm Technologies, Inc.



功耗调试通用指南

80-P0955-1SC 版本B

2015年8月5日

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

修订历史记录

版本	日期	说明
A	2015年6月	初始版本
B	2015年8月	更新了章节4.5和5.7

目录

图

1 简介

1.1 用途

本文档提供如何优化特定功耗测试用例以及调试与PRM、APSS、多媒体和Modem相关问题的详细信息。

1.2 约定

使用不同字体显示函数声明、函数名称、类型声明、属性和代码示例，例如：`#include`。

代码变量带有尖括号，例如：`<number>`。

使用不同字体显示要输入的命令，例如：**`copy a:*.* b:`**。

以粗体显示按钮和按键名称，例如：点击**Save**或按**Enter**键。

阴影表示内容在此文档版本中是新增的或已经过更改。

1.3 技术协助

如需协助或澄清本文档中的信息，可通过https://createpoint.qti.qualcomm.com/向Qualcomm Technologies, Inc.(QTI)提交用例。

若所有建议的调试方法均未解决问题，提交一个用例并提供以下信息：

- 正确的芯片组 – AMSS版本ID和操作系统(OS)
- 初始问题类型 – 软件
- 对于多媒体用例：
  - 问题区域1 – 多媒体
  - 问题区域2 – 功耗
  - 问题区域3 – 特定用例
    - 音频、视频、图形、浏览、传感器等
- 对于核心和Modem用例：
  - 问题区域1 – BSP/HLOS
  - 问题区域2 – 功耗/温升(BSP/HLOS)
  - 问题区域3 – 特定用例
    - 功耗-Modem、待机电量等

- 问题描述字段
  - 有关该问题的详细信息
    - 若与QTI标准用例不同，请提供用例详情
    - 再现问题的步骤
  - 有关在此之前进行的调试的所有信息
  - 所有建议的调试日志：波形、轨面细分、NPA转储、kmsg、Top、PowerTop、时钟转储、SurfaceFlinger、ftrace、systrace、logcat等。

如果无法访问CDMATech支持网站，则需进行注册，或发送电子邮件至 support.cdmatech@qti.qualcomm.com。

2 调试设置

在执行功耗调试之前，QTI建议先使用以下工具，从而确保功耗调试流程的完整性和有效性。

2.1 所需软件工具和设置

2.1.1 涉及RPM和Modem的用例调试工具设置

- TTrace32(T32)软件 – 该软件是利用JTAG进行功耗调试的关键，特别是使用T32模拟器获得已连目标日志、加载RAM转储和捕捉未连目标日志的关键。
- 授权的Qualcomm®产品支持工具(QPST)和QXDM Professional™ (QXDM Pro)工具 – 这两者是获得与Modem功耗用例调试相关的日志和消息的必要工具。

2.1.2 涉及应用处理器的用例调试工具设置

工具	先决条件	安装	下载地址
PowerTop	NA	adb root adb remount adb push <PowerTop location>\powertop /data/ adb shell chmod 777 /data/powertop	通过Salesforce提交用例
PerfTop	NA	adb root adb remount adb push <perf location>\perf /data/ adb shell chmod 777 /data/perf	通过Salesforce提交用例
Pytime chart	Pythonxy工具；验证已选择安装ETS和pythonxy	https://code.google.com/p/pythonxy/wiki/Downloads 安装完成后，在C:\盘中打开命令提示符，然后运行easy_install pytimechart。	https://code.google.com/p/pythonxy/wiki/Downloads
Systrace	SDK工具	http://developer.android.com/tools/sdk/tools-notes.html	Android SDK工具包
msmbusvoting	adb root adb remount	无需安装	通过Salesforce提交用例

2.2 所需硬件工具

- JTAG – 在功耗优化或调试功耗问题时获得已连目标日志（如时钟转储、PMIC转储以及GPIO转储）的必要工具。
- 功耗监控器 – 必要工具，最低采样率需为5kHz，以确保准确测量用例功耗并进行波形分析。

2.3 功耗细分仪表板

- 详细的细分测量是功耗调试和尽快找到功耗较高问题的关键。
- 构建能够使用系统功耗监控器(SPM)测量轨面电流和电压的细分仪表板；有关详情，参见*System Power Monitor Version 4 Application Note* (80-N6594-16)。

3 （功耗）仪表板用例调试流程

开始进行功耗调试前，出于调试和功耗优化目的，务必安装所有必要软件和硬件工具；有关详情，参阅第2章。

注意： 除非另外指定，否则下表中引用的步骤仅指该表中的步骤。

3.1 底电流

步骤序号	步骤	参考资料
1	▪ 为了与QTI参考数据进行适当的比较，硬件配置也必须可以进行比较。 ▪ 因此必须量化所有传感器或外部元器件的电流消耗量。 ▪ 量化以下终端的电流消耗量（必须计入已知增量）： ▫ 传感器和其他第三方元器件 ▫ 在终端中使用的不同于QTI参考数据的DDR大小 ▪ 关于此信息，可提交用例。	章节1.3
2	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 将获得的数据与QTI参考功耗数据相比较（需要将第1步中量化的已知增量考虑在内），判断功耗是否过高。 ▪ 若用户终端测量数据减去已知增量大于QTI参考数据，转到第3步。	发行说明
3	▪ 捕捉电池电量电流波形，检查导致功耗过高的原因。 ▫ 若是由于基底电流过高，转到第4步。 ▫ 若是由于异常唤醒的出现，转到第7步。	章节4.1.1
4	▪ 针对较高基底电流的调试，验证终端是否已进入VDD最小化模式。 ▫ 若终端未处于VDD最小化模式，转到第5步。 ▫ 若终端处于VDD最小化模式，转到第6步。	章节4.1.2
5	▪ 检查子系统状态以确认是哪一项阻止了终端进入VDD最小化模式，并进一步调试该子系统。	章节4.1.3 章节4.1.4

步骤序号	步骤	参考资料
6	▪ 提交用例以获取以下信息： ▫ 检查被测终端的VDD_CORE、VDD_MEM保持电压的程序 ▫ PMIC转储和GPIO参考日志（从电流消耗达到特定芯片组目标的参考终端中获取的调试日志）以用于比较 ▪ 捕捉被测终端的PMIC转储并将其与QTI参考PMIC日志相比较，以确定是否已打开了任何未使用的SMPS和LDO。如果是，关闭所有（未使用的）SMPS和LDO。 ▪ 捕捉GPIO转储并与QTI参考GPIO日志相比较，以确定GPIO配置是否与预期相同。对于使用的与QTI参考设计不同的GPIO，睡眠配置必须遵循定制设计。 ▪ 捕捉轨面电压和电流数据并与QTI参考细分数据相比较，以确定导轨消耗的电流是否过高并进行进一步调试。	章节4.1.7 功耗细分参考特定芯片组应用指南
7	▪ 对于异常唤醒的调试，通过监控子系统导轨并检查子系统特定日志（如Modem NPA日志、RPM NPA日志或应用处理器子系统(APSS)内核日志）确定导致这些唤醒的是哪个子系统。	章节4.1.6
8	▪ 第1步到第7步都执行完毕后，若功耗仍大于预期值，向QTI提交用例以获取进一步调试帮助。	章节1.3

3.2 待机

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ 传感器 ▫ 不同的DDR大小 – 通过用例请求估计值 ▫ 其他第三方元器件 ▪ 关于此信息，可提交用例。	章节1.3
2	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 与QTI参考功耗数据相比较（需要将第3步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明
3	▪ 捕捉电池电量电流波形，确认导致功耗过高的原因。 ▫ 若是由于基底电流过高，转到章节3.1的第6步。 ▫ 若是由于异常唤醒的出现，转到章节3.1的第7步。 ▫ 若DRX唤醒功耗过高，转到第4步。	
4	▪ 针对DRX唤醒周期的较高电流消耗，将其与参考波形进行比较并通过以下内容确定其中的原因： ▫ 若是由于唤醒周期过长，转到第5步。 ▫ 若是由于唤醒幅度过大，转到第6步。	
5	▪ 若DRX唤醒时间线过长，重新确认测试仪表的设置。例如，若启用了相邻基站，则终端将在RAT内或RAT间进行搜索，这将反过来导致唤醒时间线的增长。	章节4.2.1

步骤序号	步骤	参考资料
6	▪ 若唤醒时间线接近参考时间线且唤醒周期幅度较大，即峰值电流较高，则判断以下内容： ▫ 是否为特定RAT和频段配置进行了适当校准 ▫ 是否正确启动和校准了APT/ET ▪ 若使用的是第三方PA，电流消耗也同样必须进行量化。	章节4.2.1
7	第1步到第6步都执行完毕后，若功耗仍大于预期值，向QTI提交用例以获取进一步调试帮助。	章节1.3

3.3 静态显示

步骤序号	步骤	参考资料
1	▪ 若可以量化显示屏面板功耗，转到第2步。 ▪ 若不能量化显示屏面板功耗，转到第3步。	
2	▪ 将以下终端元器件与QTI参考设置相比较： ▫ 屏幕分辨率 ▫ 智能/非智能屏 ▫ 任何其他OEM特定元器件 ▪ 向QTI提交用例，说明以上任何与标准QTI参考设置不同的元器件的功耗影响。	
3-1	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。若第2步中的已知增量不能量化，转到第4步。	80-N6837-1
3-2	▪ 与QTI参考功耗数据相比较（需要将第2步中量化的已知增量考虑在内）并判断功耗是否过高。	发行说明
4	▪ 若使用的是智能屏，则检查终端是否进入了VDD最小化模式。若终端未进入VDD最小化模式，转到第5步。 ▪ 若终端进入了VDD最小化模式，转到第15步。若终端进入了VDD最小化模式，则不会有处于活动状态的时钟。 ▪ 若使用的是非智能屏，转到第6步。	章节4.1.2
5	▪ 解决终端未进入VDD最小化模式的根本原因并转到第15步。	章节4.5
6	▪ 观察功耗波形模式并识别基准电流是过高、频繁唤醒还是两者皆有。 ▫ 若基准电流过高，转到第7步。 ▫ 若观察到频繁唤醒，转到第14步。	
7	▪ 若终端具有细分能力，则捕捉完整的电源轨细分并将其与QTI参考数据相比较，以确定是否有电流消耗过高的导轨。 ▪ 捕捉SurfaceFlinger并将其与SurfaceFlinger参考日志相比较，以检查层级总数、复合类型以及更新的层级数。	章节4.6
8	▪ 识别特定芯片组功耗概览文档中特定导轨的子系统。根据第9步到第13步中的说明收集并调试这些子系统的时钟信息。	
9	▪ 捕捉时钟转储并将其与静态图像时钟计划相比较，以确定是否存在过高的时钟或是否启用了其他时钟。	章节5.4 特定芯片组（功耗）仪表板目标及用例时钟计划文档

步骤序号	步骤	参考资料
10	▪ 若CPU时钟过高，捕捉ftrace、PowerTop和Top以了解CPU停留时间和CPU负载信息。	章节4.9
11	▪ 确定哪个进程的CPU使用率过高并寻找是否存在参考日志之外的异常进程/中断。 ▪ 联系相关领域的工程师解决异常进程/线程。	章节4.9
12	▪ 若BIMC时钟过高，将/d/msm-bus-dbg/client-data/下的带宽支持与参考日志相比较。 ▪ 若需要参考日志，提交一个用例。 ▪ 确定特定客户端支持了较多带宽后，联系相关子系统工程师解决问题。	章节4.13
13	▪ 若其他时钟过高，提交用例并提供所有调试信息，如时钟转储、ftrace、波形、PowerTop、Top和电源轨细分。	章节1.3
14	▪ 针对频繁唤醒，检查PowerTop是否存在参考日志之外的异常中断。 ▪ 利用pytime图表分析ftrace，以确认中断的来源。 ▪ 联系相关领域的工程师解决中断问题。	章节4.10
15	▪ 确保已优化底电流用例或减去QTI底电流用例和用户终端之间的增量。	章节3.1
16	▪ 收集GPIO和PMIC LDO配置并将其与QTI参考数据相比较。	章节4.1.7
17	▪ GPIO和LDO的配置取决于硬件设计。 ▪ 已捕捉的GPIO和LDO数据必须经过QTI和OEM硬件团队的审核，以关闭不必要的元器件。 ▪ 若仍存在问题，转到第13步。	

3.4 MP3播放

步骤序号	步骤	参考资料
1	▪ 为了与QTI参考功耗数据进行适当的比较，硬件配置也必须可以进行比较。 ▪ 检查终端上的定制元器件。 ▫ 隧道模式与非隧道模式 ▫ 传感器和其他第三方元器件 ▫ 相对于QTI参考数据的DDR大小 ▫ 其他后处理硬件 ▪ 禁用以上所有元器件或将以上各个元器件的电流增量计入已知增量。 ▪ 关于此信息，可提交用例。	章节1.3
2	▪ 检查隧道模式。	章节4.7
3	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 若使用的是隧道模式音频播放，将所测数字与QTI参考数据相比较，以确定功耗是否过高（需要将第1步中量化的已知增量考虑在内）。 ▪ 若使用的是非隧道模式，向QTI提交用例以请求提供参考功耗数。	发行说明
4	▪ 观察功耗波形模式并识别基准电流是过高、频繁唤醒还是两者皆有。 ▫ 若基准电流过高，转到第5步。 ▫ 若观察到频繁唤醒，转到第11步。	

步骤序号	步骤	参考资料
5	▪ 若终端具有细分能力，则捕捉完整的电源轨细分并将其与QTI参考数据相比较，以确定是否有电流消耗过高的导轨。 ▪ 若不能捕捉细分，转到第7步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
6	▪ 识别哪个电源轨大于QTI功耗数据并用其确定子系统；仅在第7步到第16步中查找这些元器件。	特定芯片组功耗概览文档
7	▪ 捕捉时钟完全转储从而与QTI MP3参考时钟计划进行比较。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
8	▪ 若CPU时钟过高，捕捉ftrace、PowerTop和Top以了解CPU停留时间和CPU负载信息。	章节4.9
9	▪ 确定哪个进程的CPU使用率过高并寻找是否存在参考日志之外的异常进程/中断。 ▪ 联系相关领域的工程师解决异常进程/线程。	章节4.9
10	▪ 若BIMC时钟过高，将/d/msm-bus-dbg/client-data/下的带宽支持与QTI参考数据相比较。 ▪ 提交关于QTI参考数据的用例。 ▪ 确定特定客户端支持了较多带宽后，联系相关子系统工程师解决问题。	章节4.13 章节1.3
11	▪ 针对频繁唤醒，检查唤醒周期。 ▪ 捕捉PowerTop和ftrace并查找定期唤醒。	章节4.10
12	▪ 检查PowerTop是否存在参考日志之外的异常中断。 ▪ 利用pytime图表分析ftrace，以确认中断的来源。 ▪ 联系相关领域的工程师解决中断问题。	章节4.10
13	▪ 若频繁观察到dsp irq的周期小于2秒，检查并解决应用唤醒锁定问题。	章节4.8
14	▪ 确保已优化底电流用例或减去QTI底电流用例和用户终端之间的增量。	章节3.1
15	▪ 收集GPIO和PMIC LDO配置并将其与QTI参考数据相比较。	章节4.1.7
16	▪ GPIO和LDO的配置取决于硬件设计。 ▪ 已捕捉的GPIO和LDO数据必须经过QTI和OEM硬件团队的审核，以关闭不必要的元器件。 ▪ 若问题仍然存在，向QTI提交用例以获取进一步调试帮助；需提供所有调试信息，如时钟转储、ftrace、波形、PowerTop、Top和电源轨细分。 ▪ 关于此信息，可提交用例。	章节1.3

3.5 视频播放

步骤序号	步骤	参考资料
1	▪ 确保已优化基底底电流、静态图像和MP3用例。	章节3.1 章节3.4
2	▪ 若可以量化显示屏面板功耗，则向QTI索要可比较的面板功耗，然后再继续执行第3步。	

步骤序号	步骤	参考资料
3	▪ 为了与QTI参考数据进行适当的比较，硬件配置也必须可以进行比较。 ▪ 检查终端上的定制元器件；将各个元器件的电流增量计入已知增量： ▫ 隧道模式与非隧道模式下的音频播放 ▫ Awesome播放器或Nuplayer ▫ 传感器和其他第三方元器件 ▫ 相对于QTI参考数据的DDR大小 ▫ 其他后处理硬件 ▪ 关于此信息，可提交用例。	章节1.3
4	▪ 检查隧道模式和播放器类型。	章节4.7
5	▪ 若使用的是非隧道模式或Nuplayer（这两者不是QTI标准的一部分），向QTI索要该变体的功耗影响。	
6	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 若使用的是隧道模式音频播放，将所测数字与QTI参考数据相比较，以确定功耗是否过高（需要将第3步中量化的已知增量考虑在内）。 ▪ 若使用的是非隧道模式，通过用例请求提供参考功耗数。	发行说明
7	▪ 观察功耗波形模式并识别基准电流是过高、频繁唤醒还是两者皆有。 ▫ 若基准电流过高，转到第8步。 ▫ 若观察到频繁唤醒，转到第16步。	
8	▪ 若终端具有细分能力，则捕捉完整的电源轨细分并将其与QTI参考数据相比较，以确定是否有电流消耗过高的导轨。 ▪ 若不能捕捉细分，转到第6步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
9	▪ 识别哪个电源轨大于QTI功耗数据并用其确定子系统；仅在第10步到第14步中查找这些元器件。	特定芯片组功耗概览文档
10	▪ 捕捉SurfaceFlinger并将其与SurfaceFlinger参考日志相比较，以检查层级总数、复合类型以及更新的层级数。	章节4.6
11	▪ 捕捉时钟完全转储从而与QTI视频播放参考时钟计划进行比较。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
12	▪ 若CPU时钟过高，捕捉ftrace、PowerTop和Top以了解CPU停留时间和CPU负载信息。	章节4.9
13	▪ 确定哪个进程的CPU使用率过高并寻找是否存在参考日志之外的异常进程/中断。 ▪ 联系相关领域的工程师解决异常进程/线程。	章节4.9
14	▪ 若BIMC时钟过高，将/d/msm-bus-dbg/client-data/下的带宽支持与参考日志相比较。 ▪ 若需要参考日志，提交一个用例。 ▪ 确定特定客户端支持较多带宽后，联系相关的子系统工程师解决问题。	章节4.13
15	▪ 若问题仍然存在，向QTI提交用例以获取进一步调试帮助；需提供所有调试信息，如时钟转储、ftrace、波形、PowerTop、Top和电源轨细分。 ▪ 关于此信息，可提交用例。	章节1.3
16	▪ 针对频繁唤醒，检查唤醒周期。 ▪ 捕捉PowerTop和ftrace并查找定期唤醒。	章节4.10

步骤序号	步骤	参考资料
17	▪ 检查PowerTop是否存在参考日志之外的异常中断。 ▪ 利用pytime图表分析ftrace，以确认中断的来源。 ▪ 联系相关领域的工程师解决中断问题。	章节4.10
18	▪ 收集GPIO和PMIC LDO配置并将其与QTI参考数据相比较。	章节4.1.7
19	▪ GPIO和LDO的配置取决于硬件设计。 ▪ 已捕捉的GPIO和LDO数据必须经过QTI和OEM硬件团队的审核，以关闭不必要的元器件。 ▪ 若问题仍然存在，转到第15步。	

3.6 通话/语音呼叫

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ 传感器 ▫ 不同的DDR大小 ▫ 其他第三方元器件 ▪ 关于此信息，可提交用例。	章节1.3
2	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 与QTI参考功耗数据相比较（需要将第3步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明
3	▪ 捕捉轨面功耗细分数据并确认哪个子系统或模块消耗了高于预期的电流。	功耗细分参考特定芯片组应用指南
4	▪ 检查以下要点确定消耗较高电流的特定模块或导轨： ▫ APSS未处于电力休眠状态 – 转到章节4.1.3。 ▫ 共享时钟/资源在较高电流上运行 – 转到第5步。 ▫ Modem软件Qualcomm® Hexagon™处理器电流较高 – 转到第6步。 ▫ Modem RF或PA消耗较高电流 – 转到第7步。	
5	▪ 检查时钟转储和RPM NPA转储，以确定时钟或共享资源是否在较高电流上运行以及哪个子系统支持同一电流。	章节4.3.1
6	▪ 检查Modem NPA资源日志，以确定在较高电流上运行的Modem特定资源以及哪些客户端支持这些资源。	章节4.3.3
7	▪ 检查是否为特定RAT和频段配置进行了适当校准。 ▪ 检查是否正确启动和校准了APT/ET。 ▪ 若使用的是第三方PA，电流消耗也同样必须进行量化。	章节4.3.2
8	▪ 第1步到第7步都执行完毕后，若功耗仍大于预期值，向QTI提交用例以获取进一步调试帮助。	章节1.3

3.7 数据通话

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ 传感器 ▫ 不同的DDR大小 ▫ 其他第三方元器件 ▪ 关于此信息，可提交用例。	章节1.3
2	▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1
	▪ 与QTI参考功耗数据相比较（需要将第3步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明
3	▪ 捕捉轨面功耗细分数据并确认哪个子系统或模块消耗了高于预期的电流。	功耗细分参考特定芯片组应用指南
4	▪ 检查以下要点确定消耗较高电流的特定模块或导轨： ▫ 共享时钟/资源在较高电流上运行 – 转到第5步。 ▫ APSS消耗较高电流 – 转到第6步。 ▫ Modem软件Hexagon处理器电流较高 – 转到第7步。 ▫ Modem RF或PA消耗较高电流 – 转到第8步。	
5	▪ 检查时钟转储和RPM NPA转储，以确定时钟或共享资源是否在较高电流上运行以及哪个子系统支持同一电流。	章节4.3.1
6	▪ 检查Modem NPA资源日志，以确定在较高电流上运行的Modem特定资源以及哪些客户端支持这些资源。	章节4.3.3
7	▪ 检查APSS特定日志（如PowerTop、Top和ftrace日志），以确定引起APSS电流消耗较高的原因。	章节4.4.1
8	▪ 检查是否为特定RAT和频段配置进行了适当校准。 ▪ 检查是否正确启动和校准了APT/ET。 ▪ 若使用的是第三方PA，电流消耗也同样必须进行量化。	章节4.3.2
9	▪ 第1步到第8步都执行完毕后，若功耗仍大于预期值，向QTI提交用例以获取进一步调试帮助。	章节1.3

3.8 游戏

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ LCD分辨率 ▫ 智能/非智能屏 ▪ 若量化了面板功耗，转到第3步。 ▪ 若未量化面板功耗，则数据不能与QTI参考数据进行比较。将所有调试信息与MTP数据进行比较。转到第5步。	
2	▪ 确保结温为35°C。 ▪ 参阅章节4.15。 ▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	章节4.15 80-N6837-1
	▪ 将获得的数据与QTI参考功耗数据相比较（需要将第1步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明
3	▪ 捕捉完整细分并将其与3D游戏细分相比较。若不能捕捉电源轨细分，转到第6步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
4	▪ 识别哪个电源轨的数据大于QTI功耗数据，并确定哪个子系统正在使用它。	
5	▪ 捕捉所有时钟和调节器数据并与QTI数据进行比较。	章节5.4 章节5.9
6	▪ 将捕获的时钟数据与3D游戏时钟计划相比较，以确定哪个时钟数据较高。 ▪ 通常，CPU/GPU/BIMC时钟和中断会导致游戏功耗问题。遵循步骤6-1-2、6-2和6-3。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
6-1	▪ CPU时钟数据较高	章节4.9
6-1-1	▪ Governor、cpufreq和计划程序参数 ▫ 将governor、cpufreq和计划程序参数与QTI默认设置进行比较。 ▫ 若存在不同的参数，则： – 将参数更改为QTI默认设置后重新进行测量。 – 若没有改善，转到步骤6-1-1的下一步。 ▫ 若不存在不同的参数，通过ftrace、PowerTop和Top捕捉CPU相关信息。	章节4.14
6-1-2	▪ CPU使用率较高的进程；参见章节4.9。	章节4.9
6-1-3	▪ 异常中断或中断数量，参见章节4.10。	章节4.10
6-2	▪ BIMC时钟数据较高。 ▫ 将/d/msm-bus-dbg/client-data/下的带宽支持与MTP相比较。 ▫ 参阅章节4.13。 ▫ 若需要比较MTP数据，向QTI提交用例。 ▫ 若可以确定与MTP相比支持较多带宽的客户端，联系子系统工程师或提交用例。	章节4.13
6-3	▪ GPU时钟过高；参见章节4.11。	章节4.11
6-4	▪ 若其他时钟过高，提交用例并提供调试信息时钟转储、ftrace、波形、PowerTop、Top、systrace数据和电源轨细分数据。	章节1.3

步骤序号	步骤	参考资料
7	▪ GPIO和LDO的配置取决于硬件设计。 ▪ 已捕捉的GPIO和LDO数据必须经过QTI和OEM硬件团队的审核，以关闭不必要的元器件。	章节4.1.7
8	▪ 若功耗仍大于QTI参考数据，转到步骤6-4。	

3.9 浏览器

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ LCD分辨率 ▫ 智能/非智能屏 ▪ 若量化了面板功耗，转到第3步。 ▪ 若未量化面板功耗，则数据不能与QTI参考数据进行比较。仅将所有调试信息与MTP数据进行比较。转到第5步。	
2	▪ 确保结温为35°C。 ▪ 参阅章节4.15。 ▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1 章节4.15
	▪ 与QTI参考功耗数据相比较（需要将第1步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明
3	▪ 捕捉完整细分并将其与浏览器细分相比较。若不能捕捉电源轨细分，转到第5步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
4	▪ 识别哪个电源轨的数据大于QTI功耗数据，并确定哪个子系统正在使用它。	特定芯片组功耗概览文档
5	▪ 量化Wi-Fi消耗的电流。 ▪ 验证Wi-Fi功耗。 ▪ 与模块工程师或解决方案提供商核实，以确定Wi-Fi电流是否符合预期。	章节4.16
6	▪ 将波形与参考数据相比较。	章节4.12
7	▪ 捕捉所有时钟和调节器数据并与QTI数据进行比较。	章节5.4 章节5.9
8	▪ 将捕获的时钟数据与浏览器时钟计划相比较，以确定哪个时钟数据较高。 ▪ 通常，CPU/GPU/BIMC时钟、中断和Wi-Fi功耗会导致浏览器电流过高问题。遵循相关问题区域中的第8-1步到第10步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
8-1	▪ CPU时钟数据较高。	

步骤序号	步骤	参考资料
8-1-1	▪ Governor、cpufreq和计划程序参数 ▫ 将governor、cpufreq和计划程序参数与QTI默认设置进行比较。 ▫ 参阅章节4.14。 ▫ 若存在不同的参数，则： – 将参数更改为QTI默认设置后重新进行测量。 – 若没有改善，转到步骤8-1-2。 ▫ 若不存在不同的参数，通过ftrace、PowerTop和Top捕捉CPU相关信息。	章节4.14
8-1-2	▪ CPU使用率较高的进程；参见章节4.9。	章节4.9
8-1-3	▪ 异常中断或中断数量；参见章节4.10。	章节4.10
8-2	▪ BIMC时钟数据较高。 ▫ 将/d/msm-bus-dbg/client-data/下的带宽支持与MTP相比较。 ▫ 参阅章节4.13。 ▫ 若必须比较MTP数据，提交用例。 ▫ 若可以确定与MTP相比支持较多带宽的客户端，与模块工程师商议或提交用例。	章节4.13
8-3	▪ GPU时钟数据较高。 ▫ 参阅章节4.11。	章节4.11
8-4	▪ 其他时钟数据较高。 ▫ 提交用例并提供调试信息时钟转储、ftrace、波形、PowerTop、Top、systrace数据和电源轨细分数据。	章节1.3
9	▪ GPIO和LDO的配置取决于硬件设计。 ▪ 已捕捉的GPIO和LDO数据必须经过QTI和OEM硬件团队的审核，以关闭不必要的元器件。	章节4.1.7
10	▪ 若功耗仍大于QTI参考数据，转到步骤8-4。	

3.10 网页流

步骤序号	步骤	参考资料
1	▪ 量化终端所使用的元件的电流消耗量（必须计入已知增量）： ▫ LCD分辨率 ▫ 智能/非智能屏 ▪ 若量化了面板功耗，转到第3步。 ▪ 若未量化面板功耗，则数据不能与QTI参考数据进行比较。仅将所有调试信息与MTP数据进行比较。转到第5步。	
2	▪ 确保结温为35°C。 ▪ 检查温度然后再运行用例。 ▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1 章节4.15
	▪ 与QTI参考功耗数据相比较（需要将第1步中量化的已知增量考虑在内），判断功耗是否过高。	发行说明

步骤序号	步骤	参考资料
3	▪ 捕捉完整细分并将其与视频流细分相比较。若不能捕捉电源轨细分，转到第5步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
4	▪ 识别哪个电源轨的数据大于QTI功耗数据，并确定哪个子系统正在使用它。	特定芯片组功耗概览文档
5	▪ 量化Wi-Fi消耗的电流。 ▪ 与模块工程师或解决方案提供商核实Wi-Fi电流是否符合预期。	章节4.16
6	▪ 检查隧道模式和播放器类型。	章节4.7
7	▪ 捕捉所有时钟和调节器数据并与QTI数据进行比较。	章节5.4 章节5.9
8	▪ 将捕获的时钟数据与视频流时钟计划相比较，以确定哪个时钟数据较高。 ▪ 通常，CPU/GPU/BIMC时钟、中断、fps和Wi-Fi功耗会导致视频流电流过高问题。遵循相关问题区域中的第8-1步到第8-3步。	特定芯片组（功耗）仪表板目标及用例时钟计划文档
8-1	▪ CPU时钟数据较高。	
8-1-1	▪ Governor、cpufreq和计划程序参数： ▫ 将governor、cpufreq和计划程序参数与QTI默认设置进行比较。 ▫ 若存在不同的参数，则： – 将参数更改为QTI默认设置后重新进行测量。 – 若没有改善，转到步骤8-1-2。 ▫ 若不存在不同的参数，通过ftrace、powerTop和Top捕捉CPU相关信息。	章节4.14
8-1-2	▪ CPU使用率较高的进程	章节4.9
8-1-3	▪ 异常中断或中断数量。	章节4.10
8-2	▪ BIMC时钟数据较高。 ▫ 将/d/msm-bus-dbg/client-data/下的带宽支持与MTP相比较。 ▫ 若需要比较MTP数据，提交用例。 ▫ 若可以确定与MTP相比支持较多带宽的客户端，与模块工程师商议或提交用例。	章节4.13
8-3	▪ 其他时钟数据较高 ▫ 提交用例并提供调试信息时钟转储、ftrace、波形、PowerTop、Top、systrace数据和电源轨细分数据。	章节1.3
8-4	▪ 必须检查fps。 ▪ 若存在不同的fps，提交用例以获得调试的详细信息。	章节1.3 章节4.11
9	▪ 若功耗仍大于QTI参考数据，转到步骤8-3。	

3.11 摄像头预览

步骤序号	步骤	参考资料
1	▪ 按照功耗调试设置部分所述准备所需设置。 ▪ 确保结温为35°C。 ▪ 参阅章节4.15。 ▪ 根据QTI标准功耗测量测试程序获得终端的功耗测量数据。	80-N6837-1 章节4.15
2	▪ 若OEM没有定制的功耗数细分，转到第4步进行一般调试。 ▪ 若OEM拥有多个元器件的功耗数细分，则量化以下终端的电流消耗量（必须计入已知增量）： ▫ OIS ▫ 双摄像头 ▫ 基于硬件的人脸检测 ▫ 其他特殊硬件，如独立的IQ处理图像信号处理器(ISP) ▫ 摄像头管道中包含的除QTI IQ库之外的其他软件库，如降噪、视频稳定和场景探测 ▪ 电流增量通常是由启用和禁用功能并测量电流之差计算出来的。	
3	▪ 对功耗数进行比较时，为所有功耗测量使用主流Qualcomm® Snapdragon™摄像头应用。 ▪ 访问https://www.codeaurora.org/cgit/quic/la/platform/packages/apps/SnapdragonCamera/tree/中的Code Aurora论坛(CAF)资源库查看Snapdragon摄像头应用。	
	▪ 从发行说明中获取用例的参考功耗数。 ▪ 若用户终端测量数据减去第2步获得的已知增量大于QTI参考数据，转到第4步。	发行说明
	▪ 若参考功耗数不可用，向QTI提交用例，请求提供与QTI MTP具有相似配置的功耗数。参见章节1.3了解提交用例所需详情。 ▪ 若用户功耗数大于QTI功耗数，转到第4步。	章节1.3
4	▪ 摄像头传感器配置分析： ▫ 收集关于传感器的详细信息，如已配置的传感器分辨率或传感器fps配置。	章节4.17.3
	▫ 检查传感器是否已配置为最低分辨率且fps最大限度地符合OEM的要求；例如，若预览为1080p且快照大小为30fps 8 MP，则需要确认传感器是否已配置为输出分辨率且fps与这些设置是否较接近。 ▫ 若传感器配置正确，转到第5步。	
5	▪ 摄像头子系统分析： ▫ 收集时钟转储并检查摄像头子系统的时钟(VFE0、VFE1、CPP和BIMC)。	章节4.17.1
	▫ 若VFE时钟在TURBO上运行，查看章节4.18.1，确认是否能够启用双ISP硬件以处理传感器输出。 ▫ 若VFE和CPP时钟与时钟计划一致且摄像头硬件在正确的频率上运行，转到第6步。	参见特定芯片组（功耗）仪表板目标及用例时钟计划文档。 章节4.18.1

步骤序号	步骤	参考资料
6	▪ CPU使用率分析： ▫ 按照章节5.8和章节5.9中的步骤收集CPU使用日志（ftrace、systrace、PowerTop）。参见章节4.9识别使用CPU的Top进程。 ▫ 获取关于图像库（如用例中OEM摄像头管道中所使用的第三方3A、降噪算法）的详细信息。	章节5.8 章节5.9 章节4.9
6-1	▪ 若Top CPU使用率来自QTI 3A（线程名称 – AECAWB、AF、AFD、ASD）或其他QTI模块，提交用例并提供所有调试信息（时钟转储、ftrace转储和Top日志）。转到章节5.2了解其他子系统分析的内容。 ▪ 若Top使用率来自第三方算法，与第三方供应商协商进行CPU优化。	
6-2	▪ BIMC使用率： ▫ 遵循章节4.13中的步骤记录用例中的BIMC使用率。将BIMC使用率与QTI MTP相比较。 ▫ 若需要比较MTP数据，提交用例。 ▫ 若在摄像头用例中观察到较高的BIMC时钟和带宽支持，转到章节5.3获取关于降低总线带宽的建议。否则，转到第7步调试GPU使用率。	
6-3	▪ 若显示分辨率大于1080p，验证摄像头请求的预览大小是否能维持在1080p，并且能使显示硬件放大到物理分辨率。 ▪ 缩小预览大小将降低CPP和MDP请求的总线带宽。 ▪ 若显示分辨率不高，转到第7步调试GPU使用率。	章节4.18.3
7	▪ GPU使用率分析： ▫ 使用章节4.11.1中提到的程序监控用例中的GPU使用率。若GPU使用率较高，转到第7.1步。否则转到第8步。	章节4.11.1
7.1	▪ 检查摄像头应用是否使用GPU进行预览帧渲染；参见章节4.17.6。 ▫ 若使用了GPU，遵循章节4.18.2中的建议降低GPU使用率。 ▫ 若不再使用TextureView但GPU使用率仍旧较高，转到第7.2步。	章节4.17.6 章节4.18.2
7.2	▪ 检查摄像头用例是否从SurfaceFlinger中触发了GPU复合；参见章节4.17.7。 ▫ 若已触发GPU复合，参见章节4.17.7.1确认GPU复合的原因。 ▫ 若由于层级较多触发了GPU复合，联系应用开发者减少层级数量。 ▫ 若层级数量不足以触发混合模式复合，向QTI提交用例以对该问题进行进一步分析。 ▫ 转到第8步。	章节4.17.7
8	▪ 其他功耗优化 ▫ 若通过以上优化建议仍未获得改善，参见章节4.18.5、4.18.7和4.18.8： – 按照建议执行。 – 参见影响/副作用建议了解关于优化的可能的图像质量(IQ)和/或性能影响的信息。	章节4.18.5 章节4.18.7 章节4.18.8

3.12 视频录制

步骤序号	步骤	参考资料
1	▪ 由于视频录制所使用的管道与摄像头的相似，遵循章节3.11中的步骤优化摄像头预览电流。 ▪ 若功耗使用率仍较高，转到第2步进行视频录制特定优化。	章节3.11
2	▪ 摄像头传感器配置分析： ▫ 找到传感器操作模式和分辨率（即传感器硬件初始化和配置的模式）以及用例中的ISP配置。参见章节4.17.3和章节4.17.4了解收集信息步骤。 ▫ 检查传感器输出的配置是否接近OEM预览和实况分辨率。例如，若实况要求为8 MP，预览为1080p，传感器为16 MP，则传感器的配置应为仅输出8 MP。而不应将传感器配置为16 MP的全分辨率模式。 ▫ 若较小的传感器输出大小不可行或功耗改善不足，参见章节4.18.1启用双ISP配置（如有可能）。 ▫ 转到第3步。	章节4.17.3 章节4.17.4
3	▪ 视频编码器配置： ▫ 收集时钟转储并检查摄像头子系统的时钟(VFE0、VFE1、CPP和BIMC)和视频编码器的时钟(venus0_vcodec0_clk) ▫ 若为超高清录制且视频编码器时钟较高，考虑为视频编码器启用较低功耗模式。有关详细信息，可参见第4.19.1节。 ▫ 转到第4步。	章节4.19.1
4	▪ CPU使用率分析： ▫ 检查录制过程中摄像头管道中是否使用了其他IQ处理库。 – 量化由于使用了其他IQ处理库而产生的增量。检查是否能够仅为视频录制或4K用例减少额外图像处理以降低功耗。	
5	▪ GPU使用率分析： ▫ 收集GPU使用日志并应用摄像头预览的GPU使用率分析部分中提及的所有优化建议。 ▫ 若已完全优化GPU使用率，转到第6步。	章节3.11中的第7步
6	▪ 其他功耗优化： ▫ 若通过以上优化建议仍未获得改善，按照以下建议执行。参见影响/副作用建议了解关于优化的可能的IQ和/或性能影响的信息。 – 时域降噪是在录制期间触发的IQ功能。遵循章节4.18.4中的步骤考虑禁用时域降噪功能。	

4 （功耗）仪表板用例调试详情

4.1 底电流

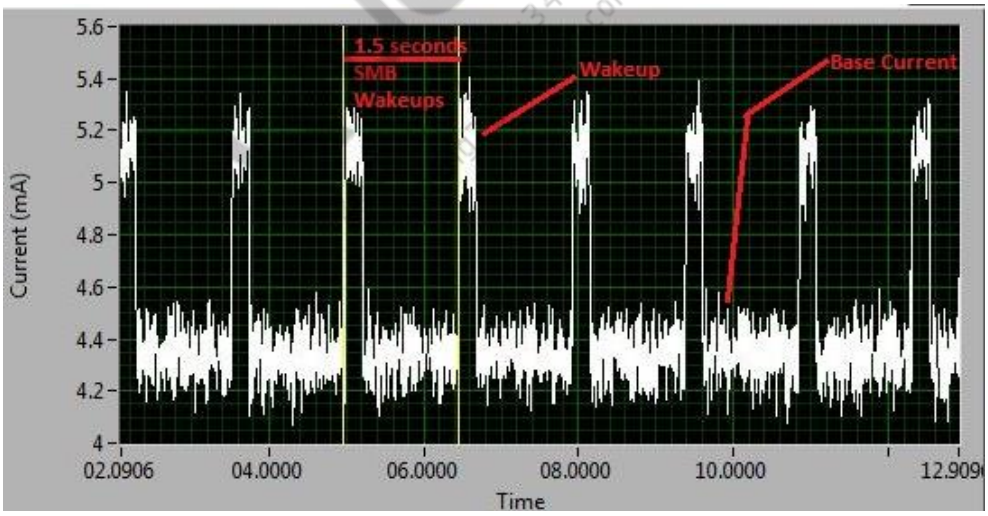
4.1.1 分析底电流波形

波形分析对功耗调试至关重要，能够提供问题的本质和正确的调试方向的信息。

底电流波形可以分为以下几个部分：

- 基底电流
- 唤醒，例如电量计唤醒和PMIC看门狗唤醒

下图是捕捉自MSM8994芯片组的底电流用例波形的快照。



从在被测端上捕捉的底电流波形可以与QTI参考波形相比较，从而确定基底电流是否较高并且/或者是否出现异常唤醒。

4.1.2 验证XO关闭和VDD最小化

使用以下任一方法验证系统是否进入了XO关闭和VDD最小化状态：

- RPM上的T32断点
  - 针对XO关闭 – xo_shutdown_enter()然后mpm_sw_done()
  - 针对VDD最小化 – vdd_min_enter()然后mpm_sw_done()

- XO关闭和VDD最小化计数的adb shell命令
  - 输入以下命令获取RPM数据：

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

以下内容是以上命令的输出；计数代表XO关闭和VDD最小化出现的数量。

```
RPM Mode:xosd
     count:0
time in last mode(msec):0
time since last mode(sec):791
actual last sleep(msec):0
client votes: 0x00020001
RPM Mode:Vdd Min
      count:28
time in last mode(msec):8000
time since last mode(sec):475
actual last sleep(msec):233000
client votes: 0x00000000
```

  - 或者检查以下RPM RAM转储的变量，以确定XO关闭和VDD最小化的计数。
    - sleep_stats[0] – XO关闭计数（系统进入XO关闭的次数）
    - sleep_stats[1] – VDD最小化计数

注意： 若系统进入VDD最小化状态，则只有VDD最小化计数会增加。VDD最小化是最低的功耗状态，包含XO关闭。仅当系统未成功进入VDD最小化而进入了XO关闭时XO关闭计数才会增加。

4.1.3 检查RPM的子系统(APSS/MPSS/LPASS)电力休眠

1. 检查RPM外部日志。
   - 针对APSS电力休眠，在RPM外部日志中查找以下消息：

```
34.521729 - rpm_shutdown_req (master: "APSS") (core: 0)
34.521759 - rpm_shutdown_ack (master: "APSS") (core: 0)
34.522064 - rpm_master_set_transition (master: "APSS") (leaving:
"Active Set") (entering: "Sleep Set")
```

   - 针对MPSS电力休眠，在RPM外部日志中查找以下消息：

```
34.513367 - rpm_shutdown_req (master: "MSS") (core: 0)
34.513397 - rpm_shutdown_ack (master: "MSS") (core: 0)
```

```
34.514038 - rpm_master_set_transition (master: "MSS") (leaving:
"Active Set") (entering: "Sleep Set")
```

- 针对LPASS电力休眠，在RPM外部日志中查找以下消息：

```
34.513367 - rpm_shutdown_req (master: "LPASS")
34.513397 - rpm_shutdown_ack (master: "LPASS")
34.514038 - rpm_master_set_transition (master: "LPASS") (leaving:
"Active Set") (entering: "Sleep Set")
```

如果没有任务限制低功耗模式，APSS/MSS/LPASS会进入电力休眠状态，以上消息可能会被覆盖且不会在RPM日志中显示。为了更好的验证，检查RPM中可用的数据结构。

2. 检查RPM数据结构。
   - 可以通过检查RPM的RPM数据结构确认子系统的电力休眠状态。显示被关闭且具有充足的时间进入睡眠后，随机断开RPM中的T32执行以监控以下变量：
     - 检查rpm.ees[0].subsystem_status, 0→APSS信息
     - 检查rpm.ees[1].subsystem_status, 1→Modem信息
     - 检查rpm.ees[2].subsystem_status, 2→LPASS信息

     或者利用系统重置获取崩溃转储，利用T32模拟器分析RPM转储。
     - 若状态为SPM_SLEEPING，则可以认为子系统在最低功耗状态中。
     - 若状态为SPM_AWAKE，则子系统处在唤醒状态而非最低功耗状态。
   - 通过提取子系统特定日志继续进行调试，以确定子系统不支持睡眠的原因。
3. 确保所有主服务器已电力休眠。
   a. 从RPM转储中获取Hansei解析器工具生成的ee-status.txt文件的主服务器(EE)状态。

```
ee-status.txt contains information about which subsystems (and their
cores) are active or sleeping.
*** APPS ***
         status:           SPM_AWAKE
         num_active_cores: 2
         pending_bringups: 0x00000000
*** MSS ***
         status:           SPM_SLEEPING
         num_active_cores: 0
         pending_bringups: 0x00000000
*** WCSS ***
         status:           SPM_SLEEPING
         num_active_cores: 0
         pending_bringups: 0x00000000
```

4.1.4 利用RPM日志确定主要资源的子系统支持

以下是可以放弃或支持低功耗以实现XO关闭和VDD最小化的主要资源：

- VDD CX – 数字电源轨
- VDD MX – 内存电源轨
- CXO – 系统时钟(XO)

1. 利用RPM NPA日志检查不同子系统的CXO支持。

   没有CXO客户端支持时，可以关闭XO以降低功耗。系统可以将数据(CX)和内存(MX)轨道维持在保持电压上从而进入VDD最小化以减少功耗。

2. 查看RPM NPA日志中的以下消息以确定哪个子系统需要CXO并且抑制了XO关闭和VDD最小化。

   - 检查RPM NPA日志中/xo/cxo节点的活动状态；例如，如以下NPA日志代码段所示，MPSS和APSS是请求CXO资源的客户端：

```
npa_resource (name: "/xo/cxo") (handle: 0x196DE8) (units: Enable)
(resource max: 1) (active max: 1) (active state: 1)  (active
headroom: 0) (request state: 1)
npa_client (name: MPSS) (handle: 0x19C780) (resource: 0x196DE8) type:
NPA_CLIENT_LIMIT_MAX) (request: 1)
npa_client (name: MPSS) (handle: 0x19C740) (resource: 0x196DE8)
(type: NPA_CLIENT_REQUIRED) (request: 1)
npa_client (name: WCSS) (handle: 0x19C620) (resource:   0x196DE8)
(type: NPA_CLIENT_LIMIT_MAX) (request: 1)
npa_client (name: LPASS) (handle: 0x19C260) (resource: 0x196DE8)
(type: NPA_CLIENT_LIMIT_MAX) (request: 1)
npa_client (name: LPASS) (handle: 0x19C220) (resource: 0x196DE8)
(type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: APSS) (handle: 0x196FF0) (resource: 0x196DE8)
(type: NPA_CLIENT_REQUIRED) (request: 1)
npa_change_event (name: sleep) (handle: 0x198C10) (resource:
0x196DE8)
end npa_resource (handle: 0x196DE8
```

   这表明MPSS和APSS抑制了终端睡眠。继续对子系统层进行调试；参见章节4.1.5中的详细信息。

4.1.5 检查单个子系统的电力休眠

由于以下原因之一，单个子系统抑制了系统电力休眠：

- 子系统中需要运行系统的主动应用或进程

- 应用或进程未正常释放资源以实现成功挂起；以下是被系统抑制的资源的示例：
  - 时钟
  - 电压轨
- 抑制系统电力休眠的频繁中断

4.1.5.1 确定APSS为何不支持电力休眠

唤醒锁

唤醒锁是APSS不支持电力休眠的主要原因之一。要检查哪个唤醒锁抑制了电力休眠：

1. 将USB连接到系统。
2. 在adb shell中输入以下命令以获取唤醒锁日志：

```
sleep 60 && cat /d/wakeup_sources > /data/wakelocks.txt &
```

3. 移除USB并使用电源键立即挂起系统。
4. 60秒后，输入以下命令重新连接USB并提取wakelocks.txt。

```
adb pull /data/wakelocks.txt <destination_folder>
```

5. 打开wakelocks.txt并检查active_since字段。若任一唤醒锁处于活动状态超过60秒，则表明唤醒锁抑制了电力休眠。

注意： wakeup_sources日志中的时间单位为毫秒。

检查抑制XO关闭和VDD最小化的时钟

1. 要检查抑制XO关闭/VDD最小化的时钟，输入以下命令以启用时钟调试挂起：

```
echo 1 > /d/clk/debug_suspend
```

   启用该标记后，系统进入挂起模式后将在dmesg日志中显示已启用的时钟。

2. 按下电源按钮三到四次以挂起并恢复系统，然后输入以下命令获取dmesg日志：

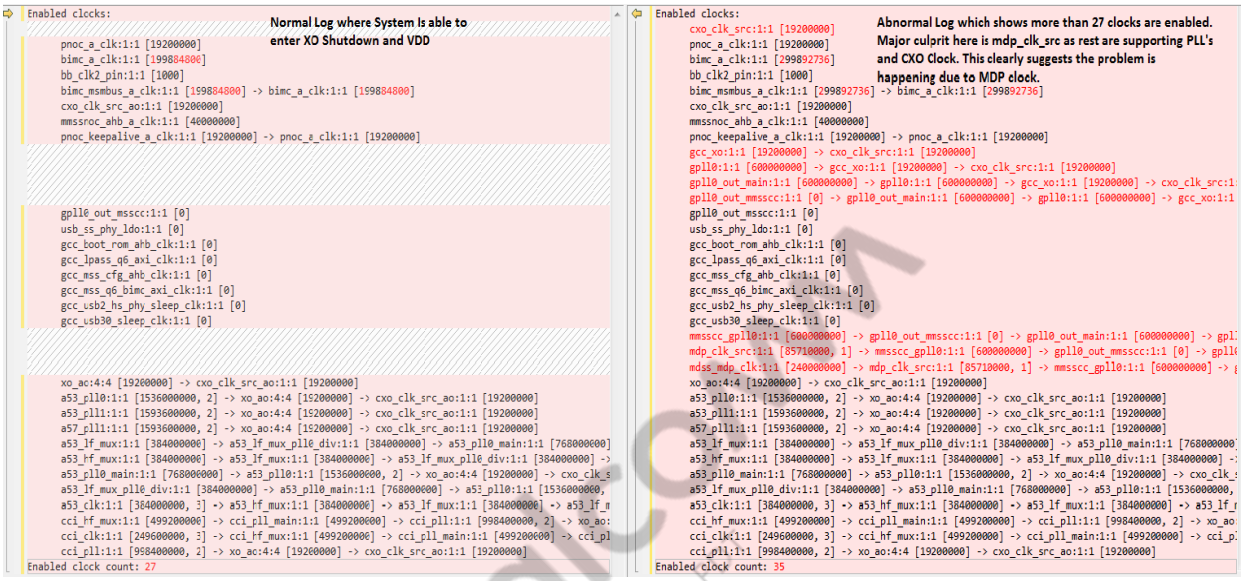
```
adb shell dmesg
```

   在该日志中，某些（常规）时钟应始终显示为已启用，若常规主要系统时钟之外的时钟显示为已启用，则可能就是抑制电力休眠的原因。以下时钟是不应在该日志中显示的时钟示例：

   - 外设时钟
   - 与显示相关的时钟

- 与多媒体子系统相关的时钟等

已启用时钟的比较 – 参见下图：



- 可以观察到，如图左侧的时钟日志所示，有27个时钟显示为已启用，但系统却在没有任何问题的情况下进入了XO关闭/VDD最小化。RPM稍后会关闭这些时钟/PLL，这些时钟是BIMC、系统总线和子系统PLL等主要共享资源的一部分。
- 如图右侧的时钟日志所示，有35个时钟已启用，但少数时钟具有区别性特征。
  - APSS请求CXO时钟源。
  - MDP（显示子系统）时钟已启用，表明驱动程序抑制了CXO进入电力休眠。

检查可能抑制XO关闭和VDD最小化的中断活动

中断频率较高会抑制系统进入XO关闭/VDD最小化。

1. 在adb shell中输入以下命令以检查频繁出现的中断：

```
sleep 20 && cat /proc/interrupts > /data/interrupt1.txt && sleep 30 &&
cat /proc/interrupts > /data/interrupt2.txt &
```

2. 立即移除USB并按下电源按钮将系统挂起。
3. 重新连接USB并提取interrupt1.txt和interrut2.txt。
4. 比较两个文件中的中断计数。
5. 若任何中断与interrupt1.txt和interrupt2.txt之间存在较大差异，联系相关子系统工程师。

4.1.5.2 确定Modem子系统为何不支持电力休眠

要确定Modem子系统为何不支持电力休眠，检查MPSS NPA日志。NPA日志中包含MPSS提出的各种NPA请求的信息。要检查哪个子系统抑制睡眠，在NPA日志中搜索npa_dump一词。这是系统将要进行日志收集时资源状态的起点。

查看资源的active_state可以获取关于资源状态的信息。检查以下主要资源：

- CXO
- VDD_CX
- VDD_MX
- CPU_VDD（Modem子系统导轨）

若这些资源的客户端占有资源，查看该客户端；例如：

```
npa_resource (name: "/core/cpu/vdd") (handle: 0xD38ECA84) (units:
active/off) (resource max: 1) (active max: 0) (active state: 1)  (active
headroom: 1) (request state: 0)
npa_client (name: gps_pe) (handle: 0xD3B6E450) (resource: 0xD38ECA84)
(type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: gps_rx) (handle: 0xD3B6E660) (resource: 0xD38ECA84)
(type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: GPS_MC_CPU_VDD_CLIENT) (handle: 0xD3B12850) (resource:
0xD38ECA84) (type: NPA_CLIENT_REQUIRED) (request: 1)
npa_client (name: RFCA_NPA_CLIENT) (handle: 0xD3B12A60) (resource:
0xD38ECA84) (type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: a2_latency_client) (handle: 0xD394F778) (resource:
0xD38ECA84) (type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: mcpm_wakeup_priority_cpu_vdd_client) (handle: 0xD394D068)
(resource: 0xD38ECA84) (type: NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: mcpm_cpu_vdd_client) (handle: 0xD394D0C0) (resource:
0xD38ECA84) (type: NPA_CLIENT_REQUIRED) (request: 0)
end npa_resource (handle: 0xD38ECA84)
```

在以上代码段中，GPS客户端请求了/core/cpu/vdd资源。这意味着GPS抑制了CPU_VDD（Modem子系统导轨）进入电力休眠状态。

要找到GPS未进入电力休眠状态的根本原因，继续从GPS方面进行分析。

警告： 由于资源状态只提供即时请求状态，需要认真分析NPA日志。NPA日志也包含请求历史记录，可以在子系统提出特别请求时查看。

4.1.5.3 确定LPASS为何不支持电力休眠

要确定LPASS为何不支持电力休眠，检查LPASS NPA日志。NPA日志中包含MPSS提出的各种NPA请求的信息。要检查哪个子系统抑制睡眠，在NPA日志中搜索npa_dump一词。这是系统将要进行日志收集时资源状态的起点。

查看资源的active_state可以获取关于资源状态的信息。检查以下主要资源：

- CXO
- VDD_CX
- VDD_MX
- CPU_VDD（Modem子系统导轨）

若这些资源的客户端占有资源，查看该客户端。

4.1.6 检查异常唤醒

唤醒是使得底电流升高的原因之一。以下原因可能引起唤醒：

- 由于传感器处理请求导致子系统唤醒
- 计划唤醒（计时器逾时）
- 由于其他子系统的中断（例如从MPSS到APSS的smd中断）导致的子系统唤醒。

要确认唤醒源，其中一个方法是监控子系统和数字轨道。大多数芯片组的各个子系统都有专用的电源轨。VDD最小化期间，所有子系统都应当处于睡眠状态，因此可以轻松识别子系统唤醒。

监控外部元器件和传感器的轨道对于识别芯片组之外的唤醒也十分有帮助。

检查子系统特定日志以确定该特定子系统是否从电力休眠中唤醒以执行某些计划活动或由于某些中断才从电力休眠中唤醒。

4.1.6.1 检查Modem唤醒

在该用例中MPSS不应该唤醒，因为执行了底电流用例使得终端处于飞行模式。捕捉Modem用户日志查看Modem是否从电力休眠中唤醒。MPSS用户日志包含多个与睡眠相关的日志，这些日志有助于确定Modem子系统是否唤醒以及为何唤醒。

以下是睡眠信息日志的代码段，是MPSS用户日志的一部分。查看Modem退出之前模式进入新模式时的时间戳。

```
0x00000000FE597E4E:   Exiting modes
0x00000000FE59DBA3:   Master wakeup stats (reason: Timer) (int pending: 33)
(Actual: 0xfe597803) (Expected: 0xfe5ae383) (Err: -93056)
0x00000000FE5A584F:   Sleep CPU frequency set (576000 Khz)
0x00000000FE5A58B2:   Solver entry (cpu frequency: 576000) (hard duration:
0x11879) (soft duration: 0x249fffffd1f) (latency budGet: 0xffffffff)
0x00000000FE5A58F3:   Solver table (mLUT: 1) (Duration: 17930)
0x00000000FE5A592D:   Mode chosen: ("CLM.disable + l2.ret + tcm.ret +
cpu_vdd.pc_l2_tcm_ret")
0x00000000FE5A5955:   Solver exit
0x00000000FE5A5AD6:   Entering modes (hard deadline: 0xfe5b7101) (backoff
deadline: 0xfe5b6985) (backoff: 0x77c) (sleep duration: 0x10ed8)
0x00000000FE5A5D40:    Program QTMR (match tick: 0xfe5b6985)
```

```
0x00000000FE5B6FB5:   Exiting modes
0x00000000FE5B763D:   Master wakeup stats (reason: Timer) (int pending:
242) (Actual: 0xfe5b6985) (Expected: 0xfe5b6985) (Err: 0)
```

以上代码段中唤醒的原因是计时器，表明是计划唤醒。Modem完成所需处理后，在下一次计划唤醒之前（如果有），睡眠解算器将根据可用延迟选择Modem要进入的适当模式。还可以看到解算器选择的模式以及下一次唤醒的硬截止日期。

根据唤醒原因，检查代码中是否设置了计时器或进行了与计时器相关的修改。

若唤醒原因为“rude”，即非计划唤醒，则QXDM Pro日志有助于确定在MPSS中执行了哪项活动。

4.1.6.2 检查APSS唤醒

启用适当记录功能的Kernel日志能够指示APSS是否从其睡眠状态中唤醒以及（如果唤醒时的）唤醒原因。可以启用以下调试掩码，以在kernel日志中记录中断信息。

```
echo 1 > /sys/module/msm_show_resume_irq/parameters/debug_mask
```

检查kernel日志中的输出数以确定哪个中断唤醒了APSS。以下代码段显示，是电源键按下中断qpnp_kpdpwr_status唤醒了APSS。

```
<6>[0414 06:51:27.872744]@0 @0 __qpnpint_handle_irq: 288 triggered [0x0,
0x08,0x0] qpnp_kpdpwr_status
<6>[0414 06:51:27.872751]@0 @0 gic_show_resume_irq: 200 triggered qcom,smd-
rpm
<6>[0414 06:51:27.872758]@0 @0 gic_show_resume_irq: 203 triggered
601d0.qcom,mpm
<6>[0414 06:51:27.872765]@0 @0 gic_show_resume_irq: 222 triggered
200f000.qcom,spmi
```

4.1.7 检查休眠电流的泄漏

若终端成功进入VDD最小化后，基底电流仍大于预期值，则可以得出结论，终端中存在一个或多个泄漏源计入总电流消耗中。

泄漏可能有多处来源，例如：

- 睡眠期间未使用但启用了的SMPS和LDO的泄漏；PMIC转储有助于检查SMPS和LDO泄漏。
- 某个或多个GPIO的最低泄漏设置不正确配置导致的GPIO管脚泄漏；GPIO转储有助于确定MSM™ GPIO的睡眠配置。
- 外围和外部元器件未禁用或进行低功耗模式配置所导致的泄漏

4.1.7.1 分析PMIC转储

PMIC转储提供关于所有LDO和SMPS的状态以及由PMIC驱动的不同模块的PMIC寄存器设置的信息。

以下是在MSM8994终端将要进入睡眠状态时提取的已解析的PMIC转储的代码段。里面包含有关SMPS和LDO的状态、电压电平和工作模式的信息。使用该信息确定在睡眠模式中是否存在无需使用但启用了的SMPS或LDO，将其关闭以减少泄漏。还可以将睡眠中需要使用的SMPS和LDO置于LPM模式而不是NPM模式以达到减少泄漏的目的。

```
================================================================
PMIC Register Dump Analysis

Filename:  C:\Temp\pmicdump.xml
PMIC:      pm8994
Version:   9.0
Timestamp: ----
Generator: Trace32
================================================================


----------------------------------------------------------------
Power Rail Analysis:
----------------------------------------------------------------
Rail        Level      Enabled      VREG_OK    VREG_ON    PD   Frequency    Mode
S1_CTRL     0.92500    On           Yes          -        On    3.200 MHz   AUTO
S2_CTRL     1.01500    On           Yes          -        On    3.200 MHz   AUTO
S3_CTRL     1.20000    On           Yes          -        On    2.133 MHz   AUTO
S4_CTRL     1.80000    On           Yes          -        On    1.600 MHz   AUTO
S5_CTRL     2.15000    On           Yes          -        On    1.600 MHz   AUTO
S6_CTRL     0.92500    On           Yes          -        Off   3.200 MHz   AUTO
S7_CTRL     1.02500    Off          No           -        On    2.133 MHz   AUTO
S8_CTRL     1.05000    Off          No           -        On    3.200 MHz   AUTO
S9_CTRL     0.83000    Off          No           -        Off   3.200 MHz   AUTO
S10_CTRL    0.83000    Off          No           -        Off   3.200 MHz   AUTO
S11_CTRL    0.83000    Off          No           -        Off   3.200 MHz   AUTO
S12_CTRL    1.01500    On           Yes          -        Off   3.200 MHz   AUTO
LDO1        1.00000    Off          No          No        On      -         LPM
LDO2        1.25000    Off          No          No        On      -         NPM
LDO3        1.20000    Off          No          No        On      -         LPM
LDO4        1.20000    Off          No          No        On      -         LPM
LDO5        1.74000    Off          Yes         No         -      -         LPM
LDO6        1.80000    On           Yes         Yes       Off     -         LPM
```

4.1.7.2 分析GPIO转储

GPIO转储提供转储收集时有关所有MSM GPIO配置的信息。以下是在MSM8994终端将要进入睡眠状态时提取的GPIO转储的代码段。

```
GPIO[0x0]: [FS]0x1, [DIR]IN, [PULL]NO_PULL, [DRV]12mA, [VAL]HIGH
GPIO[1.]: [FS]0x1, [DIR]IN, [PULL]NO_PULL, [DRV]12mA, [VAL]LOW
GPIO[2.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]HIGH
GPIO[3.]: [FS]0x1, [DIR]OUT, [PULL]NO_PULL, [DRV]12mA, [VAL]LOW
GPIO[4.]: [FS]0x2, [DIR]OUT, [PULL]NO_PULL, [DRV]8mA, [VAL]LOW
GPIO[5.]: [FS]0x2, [DIR]OUT, [PULL]NO_PULL, [DRV]8mA, [VAL]LOW
GPIO[6.]: [FS]0x3, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[7.]: [FS]0x3, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[8.]: [FS]0x4, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]HIGH
GPIO[9.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[10.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[11.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[12.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[13.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[14.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[15.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[16.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[17.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[18.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[19.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[20.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[21.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[22.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[23.]: [FS]0x4, [DIR]OUT, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[24.]: [FS]0x5, [DIR]OUT, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[25.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[26.]: [FS]0x0, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]LOW
GPIO[27.]: [FS]0x3, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]HIGH
GPIO[28.]: [FS]0x3, [DIR]IN, [PULL]NO_PULL, [DRV]2mA, [VAL]HIGH
GPIO[29.]: [FS]0x0, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]LOW
GPIO[30.]: [FS]0x0, [DIR]OUT, [PULL]NO_PULL, [DRV]2mA, [VAL]HIGH
GPIO[31.]: [FS]0x1, [DIR]IN, [PULL]PULL_DOWN, [DRV]2mA, [VAL]HIGH
```

红色框可以解读为：GPIO 11被配置为输入 – 下拉，2 mA驱动，值为“Low”。正确的GPIO睡眠配置取决于GPIO在终端设计中的使用方式以及在睡眠中是否需要GPIO。对于与QTI参考设计使用方式相同的GPIO，可以直接与QTI参考终端中提取的GPIO转储进行比较，以确定是否存在需要更改和修复的错误配置。

对于与QTI参考设计使用方式不同的GPIO，需要根据这些GPIO的使用率来确定正确的睡眠配置。

4.1.7.3 轨面电流消耗细分

单个电源轨的电流消耗细分非常有助于缩小范围并识别哪个系统元器件消耗了较高的电流。

- 首先，可以收集PMIC输出上所有PMIC轨道的电流消耗，例如SMPS电流、不是来源于SMPS的LOD等。收集由VPH（不是由PMIC驱动的元器件）直接驱动的元器件的电流消耗也是进行细分的第一步。
- 确认哪个SMPS或元器件消耗较高电流后，收集连接到这个特定SMPS的元器件的进一步细分以确定哪个元器件消耗了较高电流。

电流消耗细分会在各个（功耗）仪表板用例的特定芯片组的功耗应用指南中公布。

4.2 待机

总的平均待机电流主要依赖于平均底电流的电流消耗。若尚未优化底电流，则待机优化的第一步是优化底电流的电流消耗。按照章节4.1中的步骤对底电流的电流消耗进行优化。

4.2.1 较高的寻呼唤醒损失

电流消耗捕捉工具提供的波形非常有助于分析寻呼唤醒产生的唤醒损失。

- 对比寻呼唤醒周期的波形以确定唤醒的哪一部分消耗了较多时间或电流。
  - 系统唤醒时间
  - RF唤醒
  - RF/协议处理
  - RF睡眠
  - 系统睡眠

下图展示了DRX寻呼周期中所涉及的不同系统模块的活动阶段的寻呼唤醒波形的关联部分。

- 检查以下公共区确定寻呼时间范围是否超出预期：
  - 测试仪表设置错误，如启用了邻区
  - 对RPM进行了代码修改或启用了某些Modem日志
- 通过为日志消息设置适当的配置收集F3日志，这非常有助于在寻呼中对唤醒周期进行调试。例如调试WCDMA唤醒损失，则使用以下方法收集日志：

  a. 打开QXDM Pro并按下**F3**打开“Message View”窗口。

  b. 右键点击“Message View”窗口并从菜单中选择**Config**。将打开“Message View Configuration”窗口。

  c. 在“Message Packets”选项卡下选择**Known Message (By Subsystem)**并展开视图。

  d. 选择**Legacy**、**Mpower**、**UMTS**和**Radio Frequency**。

  e. 在“Log Packets”选项卡下选择**WCDMA**日志包。

  f. 运行WCDMA待机用例并收集F3日志。

  这些日志是详细的协议/功耗/RF日志并具有准确的时间戳，可以帮助确定占有较多时间的系统部分并帮助缩小问题的范围。

- 检查以下公共区确定寻呼唤醒振幅/峰值是否较高：
  - 如果适用，为该RAT、频段和APT/ET启用进行适当的RF校准。
  - 若在寻呼唤醒周期中峰值电流较高，轨面细分会非常有助于缩小消耗峰值电流的轨道范围。可以将问题锁定在这一区域。

4.3 通话

通话用例电流消耗较高可能是由于以下原因：

- 处理器时钟或系统时钟在较高频率上运行。
- APSS未处于电力休眠状态（参见章节4.4.1了解调试信息）。
- PA/RF前端消耗较高电流。
- 软件和硬件设计增量，如使用OEM专有的语音算法、音频输出路径中的功率放大器IC以及具有特殊功能的麦克风硬件等。

4.3.1 检查时钟和共享资源

时钟在系统中以预期以上的较高频率运行会导致系统功耗较大。可以提取时钟转储并将其与QTI参考终端时钟转储相比较以确保时钟处于预期的状态。比较时钟转储时应考虑以下几点：

- 可以通过RPM在任何时间点中断系统并运行时钟转储脚本以提取时钟转储。
- 验证除预期时钟外没有其他时钟正在运行。
- 验证所有时钟都在与QTI参考终端近似的频率上运行。进行完整的比较会更准确、更有帮助，以下是需要比较的重要的时钟：
  - BIMC(DDR)控制器时钟
  - Modem时钟
  - LPASS时钟
  - 总线（系统NOC、外围NOC、配置NOC）
- 若共享资源或时钟以预期以上的较高频率运行，检查ROM NPA日志。

  以下代码段显示了共享资源支持的不同子系统：

```
npa_resource (name: "/clk/pnoc") (handle: 0x198418) (units: KHz)
(resource max: 100000) (active max: 100000) (active state: 50000)
(active headroom: -50000) (request state: 50000)
npa_client (name: MPSS) (handle: 0x19ed58) (resource: 0x198418) (type:
NPA_CLIENT_LIMIT_MAX) (request: 4294967295)
npa_client (name: MPSS) (handle: 0x19ed18) (resource: 0x198418) (type:
NPA_CLIENT_REQUIRED) (request: 0)
npa_client (name: LPASS) (handle: 0x19e2d0) (resource: 0x198418) (type:
NPA_CLIENT_LIMIT_MAX) (request: 4294967295)
```

```
npa_client (name: LPASS) (handle: 0x19e290) (resource: 0x198418) (type:
NPA_CLIENT_REQUIRED) (request: 1)
npa_client (name: APSS) (handle: 0x11ea78) (resource: 0x198418) (type:
NPA_CLIENT_REQUIRED) (request: 50000)
npa_client (name: ICB Driver) (handle: 0x1988e8) (resource: 0x198418)
(type: NPA_CLIENT_REQUIRED) (request: 25000)
 end npa_resource (handle: 0x198418)
```

在以上代码段中，可以观察到共享资源pcnoc时钟的客户端“MPSS、LPASS、APSS和ICB驱动程序”的支持。检查这个特定资源的实际支持的NPA_CLIENT_REQUIRED请求。

可以对这个支持大于预期的特定子系统进行进一步调试。例如，若发现MPSS支持更高层次的一个或多个共享资源，可以检查MPSS NPA资源日志的MPSS特定客户端和资源支持。

4.3.2 检查PA/RF功耗

确认正确的时钟支持、导轨电压和共享资源后，若电流消耗仍然很高，需要通过测量相关轨道检查RF硬件和PA的电流消耗。

若与QTI参考数据相比，RF和PA消耗了较高的电流，需要检查RF是否适当校准。不正确的校准会导致PA在较高的增益级上工作（即使输出功率为0 dBm）。还需要进行APT/ET启用和适当校准。

PA电流消耗随第三方PA的使用而有所变化（同样也必须计入增量）。

4.3.3 检查Modem资源支持

与RPM NPA资源日志相似，Modem NPA资源日志提供关于支持Modem资源和共享资源的Modem特定客户端的信息。

Modem Hexagon时钟、Modem轨道(VDD MSS)、VDD_Core和VDD_Mem、BIMC时钟、mcpm等资源的支持便可以在Modem NPA日志中查看。

图4-1是在导航用例中收集的，支持系统级资源（如rail_MX）和Modem内部资源（如clk/cpu和cpu/busy资源）的Modem NPA资源日志的代码段。在该示例中，可以看到GPS客户端支持所有上述资源。

检查该支持是否与预期相同。例如，通过查看相同示例的QTI参考日志验证288 MHz cpu时钟的gps_rx支持。

```
: npa_resource (name: "/clk/cpu") (handle: 0xA6061AA8) (units: KHz) (resource max: 844800) (active max: 844800) (active state: 288000)  (active headroom: -556800) (request state:
:         npa_client (name: gps_pe) (handle: 0xA634A600) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: gps_rx) (handle: 0xA634A7F8) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 288000) (sequence: 0x00041E00)
:         npa_client (name: GPS_MC_CPU_CLIENT) (handle: 0xA63026B8) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: GPS_CC_CPU_CLIENT) (handle: 0xA6302790) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: RFLM_W_TX) (handle: 0xA61DE1F0) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: IPA_Q6_CPU_CLK_CLIENT) (handle: 0xA6157010) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: a2_q6sw_cpu_clk_client) (handle: 0xA6157130) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_fw_cpu_boost) (handle: 0xA61448F8) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_clk_q6) (handle: 0xA6144940) (resource: 0xA6061AA8) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 115200) (sequence: 0x00041F00)
:         npa_client (name: timer_clk_client) (handle: 0xA60EA798) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00041D00)
:         npa_client (name: npa_scheduler_clk_cpu_client) (handle: 0xA6086658) (resource: 0xA6061AA8) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: /node/core/cpu) (handle: 0xA60868E0) (resource: 0xA6061AA8) (type: NPA_CLIENT_REQUIRED) (request: 115200) (sequence: 0x00049400)
:         npa_client (name: /clk/cpu/impulse) (handle: 0xA6055A00) (resource: 0xA6061AA8) (type: NPA_CLIENT_IMPULSE) (request: 0) (sequence: 0x00041C00)
:         npa_reserved_event (name: ) (handle: 0xA60302E0) (resource: 0xA6061AA8)
:         end npa_resource (handle: 0xA6061AA8)

: npa_resource (name: "/pmic/client/rail_mx") (handle: 0xA604C990) (units: ModeID) (resource max: 6) (active max: 6) (active state: 4)  (active headroom: -2) (request state: 4)
:         npa_client (name: mcpm_voltage_mx_proxy) (handle: 0xA6143950) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_gsm_cipher1) (handle: 0xA6142B98) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_gsm_cipher) (handle: 0xA6142BE0) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_a2) (handle: 0xA6142C28) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_rf) (handle: 0xA6142C70) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x000A8E00)
:         npa_client (name: mcpm_voltage_mx_gps) (handle: 0xA6142CB8) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 4) (sequence: 0x00025900)
:         npa_client (name: mcpm_voltage_mx_tdscdma) (handle: 0xA6142D00) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_lte) (handle: 0xA6142D48) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_wcdma) (handle: 0xA6142D90) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_do) (handle: 0xA6142DD8) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_gsm1) (handle: 0xA6142E20) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_gsm) (handle: 0xA61422A8) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_voltage_mx_1x) (handle: 0xA61422F0) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 0) (sequence: 0x00000000)
:         npa_client (name: /vdd/mss) (handle: 0xA6055C40) (resource: 0xA604C990) (type: NPA_CLIENT_SUPPRESSIBLE) (request: 2) (sequence: 0x0002F600)
:         npa_client (name: /vdd/mss) (handle: 0xA6050488) (resource: 0xA604C990) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000200)
:         end npa_resource (handle: 0xA604C990)

: npa_resource (name: "/core/cpu/busy") (handle: 0xA6087BC8) (units: BINARY) (resource max: 1) (active max: 1) (active state: 1)  (active headroom: 0) (request state: 1)
:         npa_client (name: mcpm_busy_gsm_cipher1) (handle: 0xA6144530) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_gsm_cipher) (handle: 0xA6144578) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_a2) (handle: 0xA61445C0) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_rf) (handle: 0xA6144608) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x000A1F00)
:         npa_client (name: mcpm_busy_gps) (handle: 0xA6144650) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 1) (sequence: 0x0001F500)
:         npa_client (name: mcpm_busy_tdscdma) (handle: 0xA6144698) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_lte) (handle: 0xA61446E0) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_wcdma) (handle: 0xA6144728) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_do) (handle: 0xA6144770) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_gsm1) (handle: 0xA61441F8) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_gsm) (handle: 0xA6144240) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_client (name: mcpm_busy_1x) (handle: 0xA6144288) (resource: 0xA6087BC8) (type: NPA_CLIENT_REQUIRED) (request: 0) (sequence: 0x00000000)
:         npa_reserved_event (name: ) (handle: 0xA6086D6C) (resource: 0xA6087BC8)
:         end npa_resource (handle: 0xA6087BC8)
```

图4-1 Modem NPA资源日志代码段

4.4 数据

数据用例是更复杂的用例之一。它涉及获取分组数据，将这些分组数据传递给APSS进行处理，然后供一个或多个用户应用使用的Modem。

轨面细分对于调试数据用例非常有帮助，因为若MPSS或APSS消耗了高于预期的电流，则会很容易识别。然后便可以相应地进行子系统层次的调试。

若RF/PA轨道消耗较高电流，参见章节4.3.2。

若轨面细分不易获取，可以从RPM NPA日志检查不同子系统的资源支持；参见章节4.3.1。

若Modem轨道电流消耗较高或MPSS是支持较高系统资源的子系统，则检查Modem NPA资源日志缩小范围并确定引起该支持的Modem。参见章节4.3.3了解检查Modem NPA资源日志的信息。

4.4.1 检查APSS

APSS可能是数据用例中电流消耗较高的主要原因之一，因为附加功能会影响功耗。使用以下方法或日志确定APSS是否消耗较高电流：

- 分析PowerTop日志并将其与QTI参考终端相比较，步骤如下：
  - 观察PowerTop日志中每个C状态（低功耗状态）的时间。
  - 检查PowerTop日志中每个频率和中断活动的时间。若PowerTop中显示了异常中断，检查请求该中断的驱动程序。
  - 这些测试能够对APSS消耗较高电流的原因进行粗略的估计。
  - 可以将这些日志与QTI参考终端日志相比较，比较分析哪个系统元器件消耗较高电流。
- 分析Top日志并将其与QTI参考终端相比较，步骤如下：
  - 停止或取消Top日志中消耗CPU时间的异常进程。
  - 利用Top数据检查额外运行的进程。
- 分析ftrace日志，步骤如下：
  - 分析kernel中运行的消耗额外CPU时间的进程或API。
  - 分析特定频率中每个核的时间量。
  - 若kworker线程活动较高，检查哪个进程独占了CPU。
  - 将ftrace日志与QTI参考终端相比较，以确定哪个进程在APSS上更活跃以及是否能够停止。

4.5 对智能面板进行静态图像VDD最小化调试

1. 使用以下调试掩码收集kmsg：

```
adb root
adb remount
adb shell
mount -t debugfs none /sys/kernel/debug
echo 1 > /sys/kernel/debug/clk/debug_suspend
echo 32 > /sys/module/msm_pm/parameters/debug_mask
echo 8 > /sys/module/mpm_of/parameters/debug_mask
```

2. 在kmsg中查找保持CXO的时钟。例如在以下代码段中，UART保持CXO并抑制了XO关闭和VDD最小化。

```
<6>[ 1163.416398]   gpll0_out_main:1:1 [600000000] -> gpll0:1:1 [600000000] -> gcc_xo:1:1 [19200000] -> cxo_clk_src:2:2
[19200000]
<6>[ 1163.416441]   blsp1_uart2_apps_clk_src:1:1 [7372800, 1] -> gpll0_out_main:1:1 [600000000] -> gpll0:1:1 [600000000]
-> gcc_xo:1:1 [19200000] -> cxo_clk_src:2:2 [19200000]
<6>[ 1163.416497]   gpll0_out_msscc:1:1 [0]
<6>[ 1163.416515]   usb_ss_phy_ldo:1:1 [0]
<6>[ 1163.416532]   gcc_blsp1_ahb_clk:1:1 [0]
<6>[ 1163.416551]   gcc_blsp1_uart2_apps_clk:1:1 [7372800] -> blsp1_uart2_apps_clk_src:1:1 [7372800, 1] ->
gpll0_out_main:1:1 [600000000] -> gpll0:1:1 [600000000] -> gcc_xo:1:1 [19200000] -> cxo_clk_src:2:2 [19200000]
<6>[ 1163.416610]   gcc_boot_rom_ahb_clk:1:1 [0]
<6>[ 1163.416628]   gcc_mss_q6_bimc_axi_clk:1:1 [0]
<6>[ 1163.416648]   gcc_usb2_hs_phy_sleep_clk:1:1 [0]
```

注意： 可能有其他需要cxo_clk_src_ao的时钟。这些是这些客户端提出的“Active Only”请求，应忽略。仅查找cxo_clk_src。

3. 查找出现的hwirqs，如以下示例所示：

```
<6>[  376.522299] msm_mpm_interrupts_detectable(): gic preventing system sleep modes during idle
<6>[  376.522312]   hwirq: 65
<6>[  376.522316]   hwirq: 115
<6>[  376.589984] msm_mpm_interrupts_detectable(): gic preventing system sleep modes during idle
<6>[  376.589997]   hwirq: 115
```

4. 要检查irq对应哪些内容，输入以下命令在终端的中断列表中查找特定的irq：

```
adb shell
cat /proc/interrupts
```

5. 与相应的团队商议以了解为何会触发该irq。例如，针对下图，与显示和图形团队商议。若特定irq不应存在或可以忽略，将其添加到芯片组功耗管理.dtsi文件中的跳过列表中。

```
 65:          0         81         15        141          0          0          0          0      GIC  kgsl-3d0
 74:          0          0          0          0          0          0          0          0      GIC  msm_iommu_nonsecure_irq
 75:          0          0          0          0          0          0          0          0      GIC  msm_iommu_secure_irq, msm_iommu_secure_irq,
msm_iommu_secure_irq, msm_iommu_secure_irq
 76:        822          0          0       1082          0          0          0          0      GIC  msm_vidc
 78:          0          0          0          0          0          0          0          0      GIC  msm_iommu_secure_irq, msm_iommu_secure_irq
 79:          0          0          0          0          0          0          0          0      GIC  msm_iommu_nonsecure_irq
 81:          2          0          0          1          0          0          0          0      GIC
 82:         70          0          0       1582          0          0          0          0      GIC  cci
 83:          3          0          0          0          0          0          0          0      GIC  csid
 84:          0          0          0          0          0          0          0          0      GIC  csid
 85:          0          0          2          0          0          0          0          0      GIC  csid
 86:          0          0          0          0          0          0          0          0      GIC  csid
 89:          4          0          0          0          0          0          0          0      GIC
 90:          4          0          0          0          0          0          0          0      GIC
 97:          0          0          0          0          0          0          0          0      GIC  msm_iommu_nonsecure_irq, msm_iommu_nonsecure_irq,
 msm_iommu_nonsecure_irq
102:          0          0          0          0          0          0          0          0      GIC  msm_iommu_nonsecure_irq, msm_iommu_nonsecure_irq,
msm_iommu_nonsecure_irq, msm_iommu_nonsecure_irq
109:          0          0          0          0          0          0          0          0      GIC  ocmem_dm_irq
110:          0          0          0          0          0          0          0          0      GIC  csiphy
111:          0          0          0          0          0          0          0          0      GIC  csiphy
112:          0          0          0          0          0          0          0          0      GIC  csiphy
115:          0        333       1632          0          0          0          0          0      GIC  MDSS
```

6. 在.dtsi文件中添加带有0xff的irq，以将该irq屏蔽，将其忽略。对于MSM8994，可以在/kernel/arch/arm64/boot/dts/qcom/msm8994-v2-pm.dtsi.dtsi中找到.dtsi文件。

   例如，要屏蔽irq 66，将<0xff 66>添加到.dtsi文件中列表结尾处的mpm中断映射中，如下所示：

```
qcom,mpm@fc4281d0 {
        compatible = "qcom,mpm-v2";
        reg = <0xfc4281d0 0x1000>, /* MSM_RPM_MPM_BASE 4K */
            <0xf900f008 0x4>;   /* MSM_APCS_GCC_BASE 4K */
        reg-names = "vmpm", "ipc";
        interrupts = <0 171 1>;
        clocks = <&clock_rpm clk_cxo_lpm_clk>;
        clock-names = "xo";

        qcom,ipc-bit-offset = <1>;

        qcom,gic-parent = <&intc>;
        qcom,gic-map = <2 216>, /* tsens_upper_lower_int */
                <47 165>, /* usb30_hs_phy_irq */
                <52 212>, /* lfps_rxterm_irq for pwr_event_irq */
                <55 172>, /* usb1_hs_async_wakeup_irq */
                <62 222>, /* ee0_krait_hlos_spmi_periph_irq */
                <0xff 20>,  /* arch_timer */
                <0xff 23>,  /* ARM64 Single-Bit Error PMU IRQ */
                <0xff 33>,  /* APCC_qgicL2PerfMonIrptReq */
                <0xff 34>,  /* APCC_qgicL2ErrorIrptReq */
                <0xff 35>,  /* WDT_barkInt */
                <0xff 40>,  /* qtimer_phy_irq */
                <0xff 48>,  /* cpr */
                <0xff 51>,  /* cpr */
                <0xff 54>,  /* CCI error IRQ */
                <0xff 56>,  /* modem_watchdog */
                <0xff 57>,  /* mss_to_apps_irq(0) */
                <0xff 58>,  /* mss_to_apps_irq(1) */
                <0xff 59>,  /* mss_to_apps_irq(2) */
                <0xff 60>,  /* mss_to_apps_irq(3) */
```

4.6 SurfaceFlinger调试

SurfaceFlinger是基于渲染在屏幕上的多来源的用于组成显示帧的服务。

SurfaceFlinger为每个源创建一个层和一个缓冲区，这些层被渲染在屏幕上。

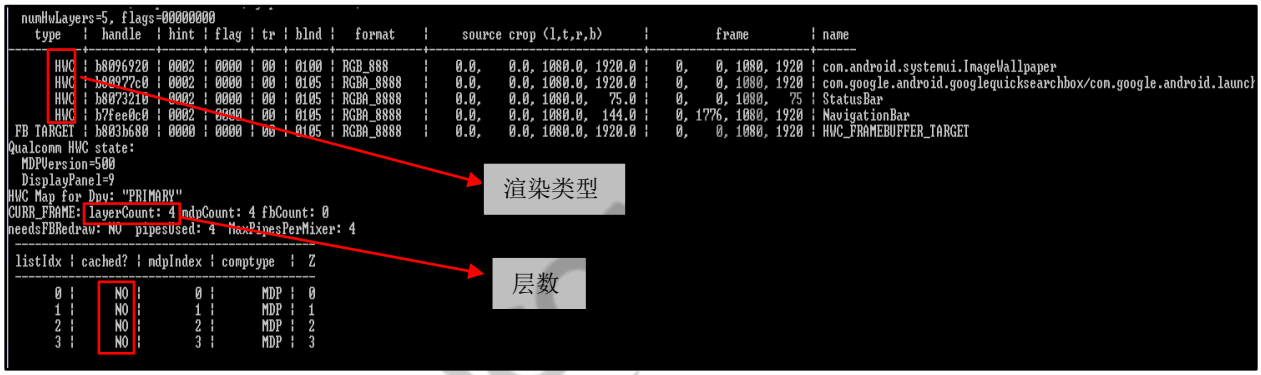
经常观察到的通用层如下：

- 状态栏
- 导航栏
- 应用

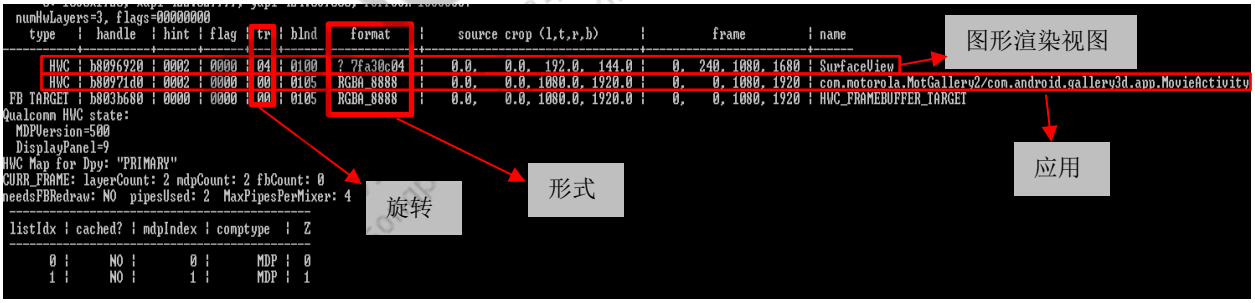
要收集SurfaceFlinger日志，输入以下命令：

```
adb shell
dumpsys SurfaceFlinger
```

下图为静态图像SurfaceFlinger：



下图为全屏视频播放SurfaceFlinger：



在上例中显示了以下内容：

- 应用
- 渲染类型
- 复合类型：
  - MDP复合
  - GPU复合意味着高功耗
- 总层数 – 更新层数较多意味着电流也较高
- 渲染形式 – RGB、YUV等
- 屏幕旋转
- 缓存和未缓存的层（缓存的层尚未更新）

4.7 隧道模式和播放器类型检查

4.7.1 隧道模式检查

1. 输入以下adb命令捕捉user-space(logcat)日志。

```
adb logcat > c:\temp\logcat.txt
```

2. 播放MP3然后捕捉MP3持续播放约30秒的日志。
3. 在logcat日志中，查找以下确保隧道模式播放持续进行的消息：

```
music_offload_avg_bit_rate=128000;music_offload_sample_rate=44100
```

4. 若日志中显示deep-buffer-playback消息，则表明正在进行非隧道模式（即标准模式）播放。

4.7.2 播放器类型检查

1. 运行用例时，输入以下命令检查播放器的类型（Nuplayer或Awesome播放器）。

```
adb shell
Dumpsys media.player
```

例如：

```
shell@victara:/ $ dumpsys media.player
dumpsys media.player
 Client
  pid(21330), connId(35), status(0), looping(false)
 AwesomePlayer
  Track 1
   MIME(audio/mp4a-latm), decoder(OMX.google.aac.decoder)
  Track 2
   MIME(video/avc), decoder(OMX.qcom.video.decoder.avc)
videoDimensions(192 x 144)
Total Video Frames Decoded(43)
Total Video Frames Rendered(41)
Total Playback Duration(2783 ms)
numVideoFramesDropped(1)
Average Frames Per Second(14.7301)
First Frame Latency (75 ms)
Number of times AV Sync Lost(1)
Max Video Ahead Time Delta(136)
Max Video Behind Time Delta(43)
Max Time Sync Loss(40055)
EOS(0)
PLAYING(1)
 AudioOutput
  stream type(3), left - right volume(1.000000, 1.000000)
  msec per frame(0.022676), latency (261)
  aux effect id(0), send level (0.000000)
 AudioTrack::dump
  stream type(3), left - right volume(1.000000, 1.000000)
  format(1), channel count(2), frame count(7680)
  sample rate(44100), status(0)
  state(0), latency (261)
```

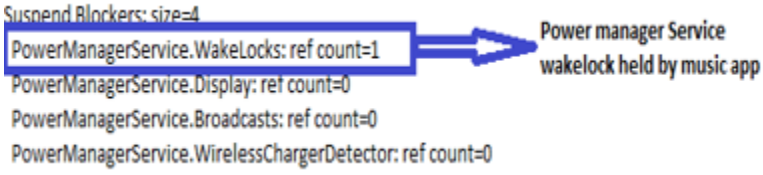
```
shell@victara:/ $ dumpsys media.player
dumpsys media.player
 Client
  pid(21330), connId(36), status(0), looping(false)
 NuPlayer
  numFramesTotal(127), numFramesDropped(0), percentageDropped(0.00)
 AudioOutput
  stream type(3), left - right volume(1.000000, 1.000000)
  msec per frame(0.022676), latency (435)
  aux effect id(0), send level (0.000000)
 AudioTrack::dump
  stream type(3), left - right volume(1.000000, 1.000000)
  format(1), channel count(2), frame count(15360)
  sample rate(44100), status(0)
  state(0), latency (435)
```

4.8 音乐应用唤醒锁检查和调试

4.8.1 唤醒锁分析

1. 关于唤醒锁信息的采集，参见章节5.10；关于唤醒源的分析，参见章节4.1.5.1。
2. 通过以下检查确保电源管理器服务正在占用唤醒锁。

   以下代码段显示预期的dumpsys功耗输出：



3. 若未观察到以上唤醒锁，占用测试唤醒锁并确认功耗是否有所减少。

4.8.1.1 占用测试唤醒锁

要占用试验的唤醒锁，输入以下命令：

```
adb shell
echo test > sys/power/wake_lock
```

4.8.1.2 移除测试唤醒锁

1. 要移除测试唤醒锁，输入以下命令：

   ```
   adb shell
   echo test > sys/power/wake_unlock
   ```

2. 若以上实验解决了问题，则在MP3播放时将音乐应用修改为占用唤醒锁并关闭显示。

4.9 分析CPU使用率较高的进程

1. 按照章节5.8所述捕捉Top或ftrace数据。
2. 确认哪个进程的CPU使用率最高，查找是否存在不应该运行的进程。
   a. 若存在异常进程或线程，与模块工程师商议该进程或线程。
   b. 若不存在异常进程或线程，执行以下步骤：
      i 确定CPU使用率较高的进程或线程。以下代码段显示进程ID为472的CPU使用率与用户终端和MTP终端存在差异。

Top数据A

```
  PID   TID PR CPU% S     VSS     RSS PCY UID      Thread          Proc
11893 11893  5   3% R  13316K   2792K  fg root     top             top
  472 11867  2   3% S 335924K  30376K  fg media    VideoDecMsgThre /system/bin/mediaserver
  417   417  0   1% S 256940K  28608K  fg system   surfaceflinger  /system/bin/surfaceflinger
  472 11869  2   0% S 335924K  30376K  fg media    gle.aac.decoder /system/bin/mediaserver
```

Top数据B

```
  PID   TID PR CPU% S     VSS     RSS PCY UID      Thread          Proc
11587 11587  5   3% R  13316K   2768K  fg root     top             top
  472 11501  2   1% S 459768K  26320K  fg media    VideoDecMsgThre /system/bin/mediaserver
  417   417  1   1% S 343732K  28588K  fg system   surfaceflinger  /system/bin/surfaceflinger
  472 11503  1   0% R 459768K  26320K  fg media    gle.aac.decoder /system/bin/mediaserver
```

      ii 利用PerfTop工具详细地进行调试（参见章节2.1.2）。

3. 连接USB。
4. 输入以下命令以捕捉PerfTop数据。

```
adb root
adb remount
adb shell /data/perf top -p 472
```

以下代码段显示PerfTop数据A和B。

PerfTop数据A

```
←[2J
PerfTop:     502 irqs/sec  kernel:65.1%  exact:  0.0% [1000Hz cycles],  (target_pid: 472)
----------------------------------------------------------------------------

          samples  pcnt function               DSO
          _______ _____ ______________________ ____________________________

         161.00 ←[31m13.0%←[m applyLimiter           linker
          65.00 ←[31m 5.3%←[m android::AwesomePlayer /system/lib/libstagefright.so
          61.00 ←[32m 4.9%←[m dit_fft(long*, int, FI linker
          55.00 ←[32m 4.4%←[m imdct_block(mdct_t*, l linker
          53.00 ←[32m 4.3%←[m __memcpy_base          /system/lib/libc.so
          46.00 ←[32m 3.7%←[m ifree                  /system/lib/libc.so
          42.00 ←[32m 3.4%←[m __rindex               /system/lib/libc.so
          41.00 ←[32m 3.3%←[m android_atomic_add     /system/lib/libcutils.so
          40.00 ←[32m 3.2%←[m CBlock_InverseQuantize linker
          39.00 ←[32m 3.2%←[m pthread_mutex_lock     /system/lib/libc.so
          36.00 ←[32m 2.9%←[m dct_IV(long*, int, int linker
          30.00 ←[32m 2.4%←[m je_malloc              /system/lib/libc.so
          29.00 ←[32m 2.3%←[m CBlock_ReadSpectralDat linker
          28.00 ←[32m 2.3%←[m __udivsi3              /system/lib/libc.so
          23.00 ←[32m 1.9%←[m CBlock_FrequencyToTime linker
          19.00 ←[32m 1.5%←[m android::TimedEventQue /system/lib/libstagefright.so
←[2J
```

PerfTop数据B

```
←[H←[2J
  PerfTop:     860 irqs/sec  kernel:54.5%  exact:  0.0% [1000Hz cycles],  (target_pid: 472)
----------------------------------------------------------------------------

            samples  pcnt function               DSO
            _______ _____ ______________________ ____________________________

           323.00 ←[31m 8.4%←[m __memcpy_base          /system/lib/libc.so
           283.00 ←[31m 7.4%←[m android_atomic_add     /system/lib/libcutils.so
           274.00 ←[31m 7.1%←[m ifree                 /system/lib/libc.so
           269.00 ←[31m 7.0%←[m void android::HwAudioRes linker
           206.00 ←[31m 5.4%←[m je_malloc             /system/lib/libc.so
           158.00 ←[32m 4.1%←[m applyLimiter          linker
           113.00 ←[32m 2.9%←[m pthread_mutex_lock    /system/lib/libc.so
           107.00 ←[32m 2.8%←[m memcpy_to_i16_from_flo /system/lib/libaudioutils.so
            90.00 ←[32m 2.3%←[m memcpy_to_float_from_q /system/lib/libaudioutils.so
            87.00 ←[32m 2.3%←[m android::RefBase::incS /system/lib/libutils.so
            77.00 ←[32m 2.0%←[m __udivsi3              /system/lib/libc.so
            69.00 ←[32m 1.8%←[m pthread_mutex_unlock   /system/lib/libc.so
            61.00 ←[32m 1.6%←[m dit_fft(long*, int, FI linker
            57.00 ←[32m 1.5%←[m memcmp                 /system/lib/libc.so
            56.00 ←[32m 1.5%←[m android::RefBase::decS /system/lib/libutils.so
            54.00 ←[32m 1.4%←[m imdct_block(mdct_t*, l linker
←[H←[2J
```

如PerfTop数据所示，进程ID为472中的内存处理（memcpy、malloc、free）功能在PerfTop数据A中被呼叫的频率更高。

分析该数据后，与负责媒体服务器的模块工程师商议或提交用例。

4.10 中断分析

1. 捕捉PowerTop数据。
2. 若数据与以下示例相似，捕捉ftrace日志以对qcom,smd-rpm进行详细调试。

数据A

```
Top causes for wakeups:
  37.9% (514.6)       <interrupt> : arch_timer
  11.6% (157.0)       <interrupt> : qcom,smd-rpm
  10.7% (145.0)       <interrupt> : MDSS
   5.7% ( 78.0)       <interrupt> : kgsl-3d0
   4.7% ( 63.6)       <interrupt> : arch_mem_timer
   0.8% ( 10.4)       <interrupt> : mmc0
   0.1% (  1.0)       <interrupt> : i2c-msm-v2-irq
   0.0% (  0.2)       <interrupt> : mmc0
```

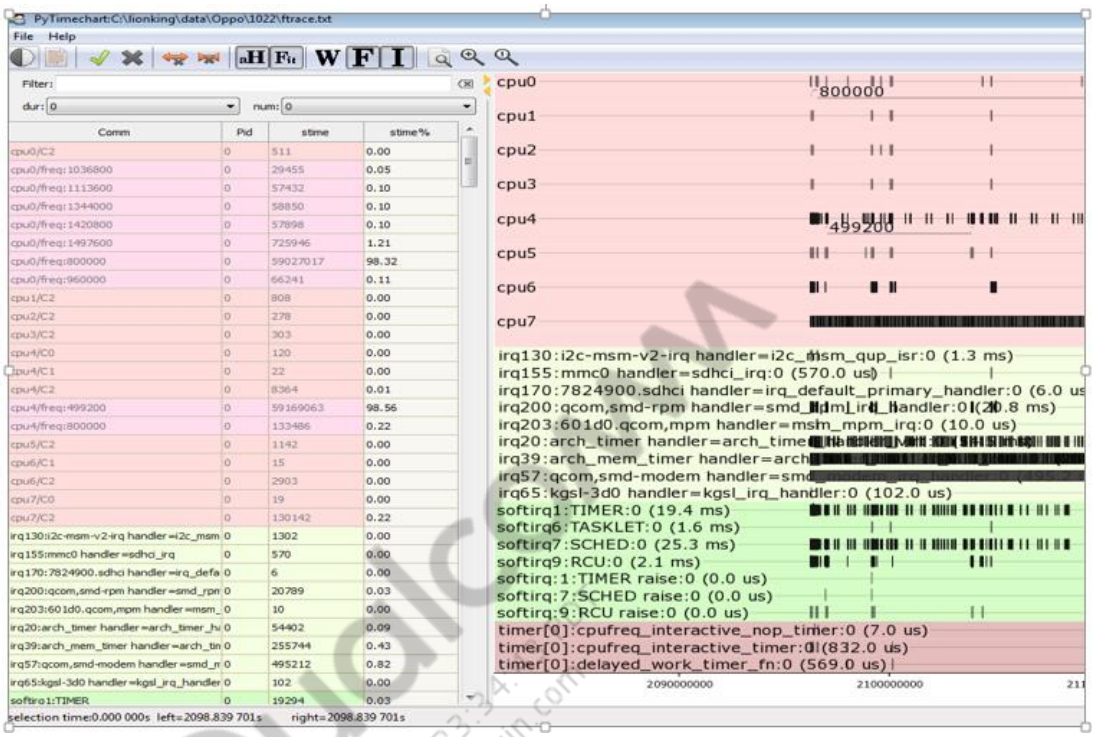
数据B

```
Top causes for wakeups:
  36.5% (504.2)       <interrupt> : arch_timer
  10.3% (142.0)       <interrupt> : MDSS
   5.0% ( 72.0)       <interrupt> : kgsl-3d0
   4.8% ( 65.8)       <interrupt> : arch_mem_timer
   4.0% ( 52.0)       <interrupt> : qcom,smd-rpm
   0.9% ( 11.2)       <interrupt> : mmc0
   0.1% (  1.0)       <interrupt> : i2c-msm-v2-irq
   0.0% (  0.2)       <interrupt> : mmc0
```

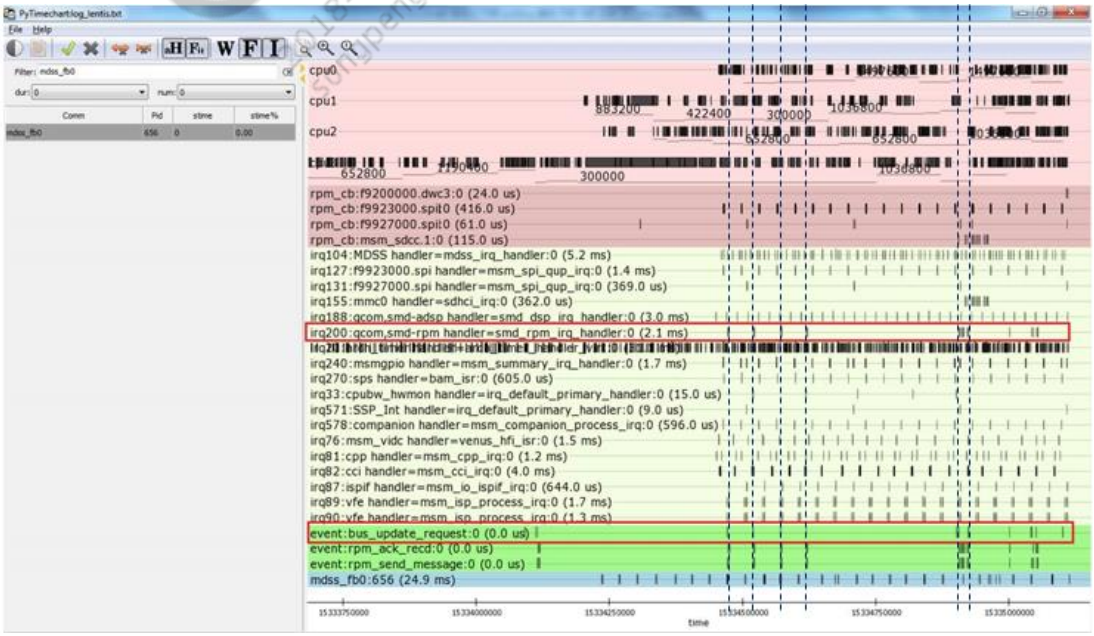
利用pytime图表对中断进行调试。关于pytime图表的安装，参见第2章。

3. 启动pytime图表。
4. 通过pytime图表中的菜单打开ftrace日志。

以下屏幕随即显示。



5. 按照以下对准确定引起qcom,smd-rpm中断的是哪个模块。



6. 若观察到与上述ftrace(bus_update_request)中相似的行为，检查ftrace日志文件中bus_update_request的客户端，如下所示：

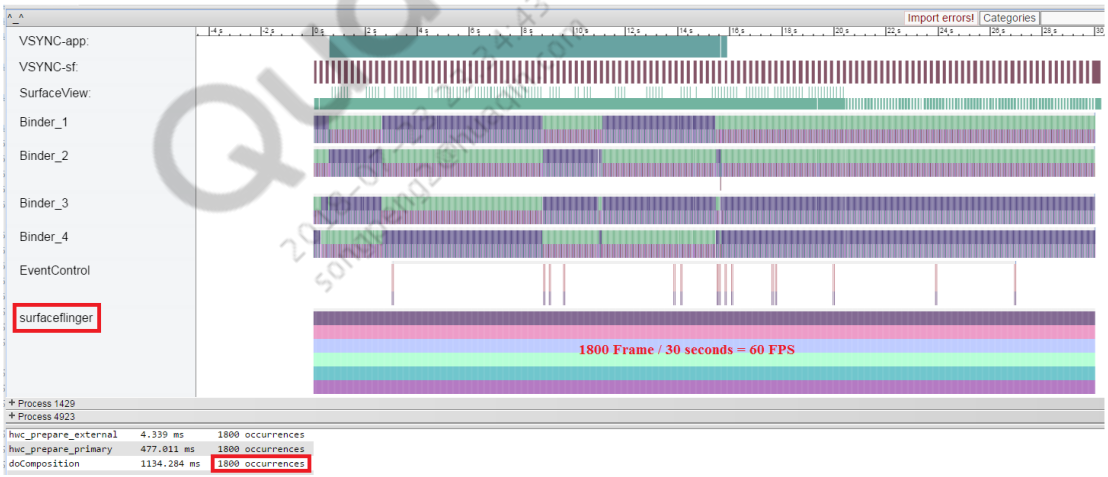
```
[001] ...1  1489.961370: bus_update_request: time:1489.952861564 name:mdss_mdp src:22 dest:512 ab:1830570776 ib:1830570776
[001] ...1  1489.961373: bus_update_request: time:1489.952861564 name:mdss_mdp src:23 dest:512 ab:1830570776 ib:1830570776
```

根据分析结果所示，是MDSS_MDP导致了qcom,smd-rpm中断。与显示工程师商议并解决该问题或提交用例。

4.11 调试较高的GPU时钟频率

注意： 执行该步骤需要用到Chrome浏览器。

1. 首先通过systrace验证fps：
   a. 捕捉systrace日志并用Chrome浏览器打开。
   b. 利用分解量和持续时间计算fps。



2. 验证GPU占有率。
   - 键入以下命令并将其与MTP相比较以检查GPU占有率。

```
Cat /sys/class/kgsl/kgsl-3d0/gpubusy
355818 1013309
```

   - GPU占有率=（每帧的活跃时间/每帧的总时间）* 100
   - 总时间表示每帧的活跃时间+每帧的休眠时间
   - 若GPU进入睡眠状态，则gpubusy的值将重置。
   - 例如，(355818/1013309) * 100 = 35.1%每帧
   - 若与MTP相比存在不同的GPU占有率，则GPU在另一环境中使用。

3. 提交用例，提供获取适当的调试信息。有关详细信息，可参见*Presentation: Graphics Power and Performance Overview* (80-NP885-1)了解更多与调试相关的信息。

4.11.1 监控GPU使用率的脚本

1. 将以下脚本复制粘贴到文件中并命名为gpu_busy.pl：

```
>>>
#!/usr/bin/perl -w
while(1)
{
   &busy;
   print "\n";
   sleep 1 ;
}
sub busy
{
   $gpu3d = `adb  shell cat /sys/class/kgsl/kgsl-3d0/gpubusy`;
   $pct = 0.0;
   if( $gpu3d =~ m/\s*(\d+)\s+(\d+)/)
   {
        if( $1 > 0  && $2 > 0 )
        {
            $pct = $1 / $2 * 100;
        }
       printf("3D GPU Busy: %5.2f\n", $pct);
   }
}
>>>
```

2. 在已安装Perl的Linux或Windows机器中输入以下命令：

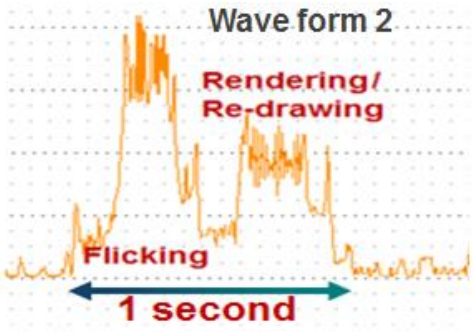
```
$./perl gpu_busy.pl
```

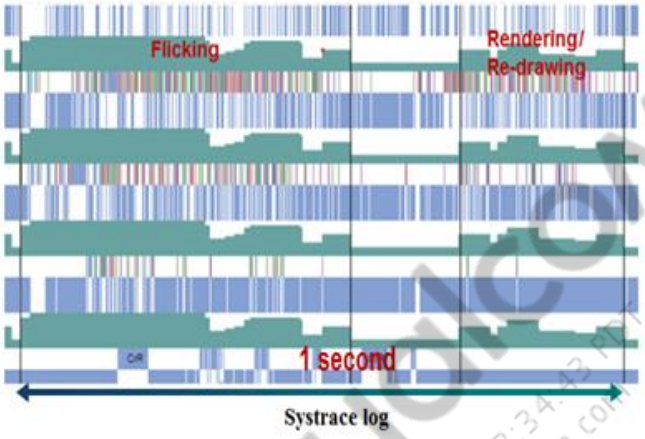
4.12 分析波形

要对波形分析做基本了解，阅读以下说明并与之后的图片和systrace日志相结合。

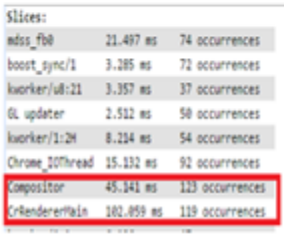
- 在以下图片中可以观察到，波形1和波形2具有不同的行为，与波形1相比，波形2的每个跳动波包含两个波。
- 要确定问题所在，使用systrace工具进行详细调试。需要用到Chrome浏览器。
- 捕捉systrace日志并用Chrome浏览器打开。
- 按照时间范围对齐每个波形和每个systrace日志。
- 可以观察到波形1和波形2的systrace日志具有不同的行为。
- 确保波形2的systrace日志的第二个区域是systrace日志2。

- 在systrace日志中查看渲染线程和复合线程。
- 根据分析结果，渲染线程和复合线程是波形2中的第二个波出现的原因。
- 在这种情况下，与图形工程师和显示工程师商议该行为是否正确。





systrace日志2



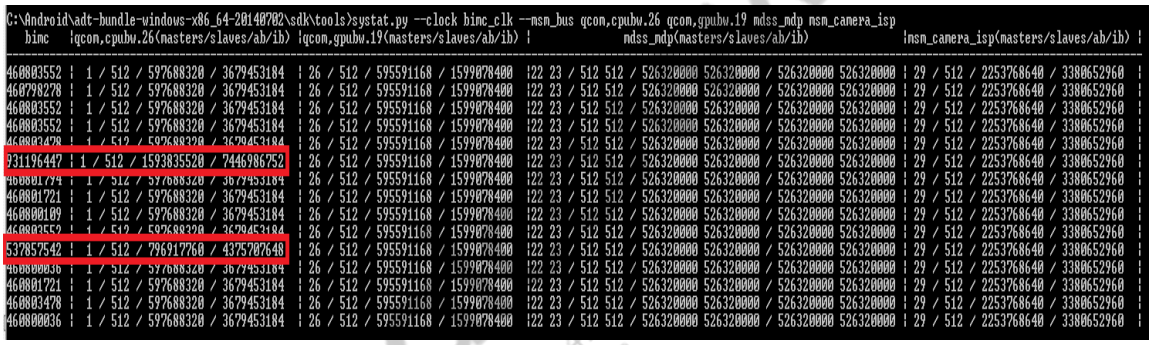
You can see compositor for drawing and render thread

4.13 分析BIMC时钟

- 使用msmbusvoting工具分析BIMC时钟。
- 实时检查每个客户端的时钟和带宽支持。

```
Msmbusvoting.exe --clock bimc_clk -msm_bus-list <a client name seen in
the following example>
```

- 确定哪个客户端实时支持BIMC时钟，如以下示例所示：



4.14 检查调节器、计划程序和CPU频率参数

- 调节器参数
  - 检查以下路径下的所有节点：
    - Cortex-A53（小群集）– /sys/devices/system/cpu/cpu0/cpufreq/interactive/
    - Cortex-A57（大群集）– /sys/devices/system/cpu/cpu4/cpufreq/interactive/

```
root@msm8992:/sys/devices/system/cpu/cpu0/cpufreq/interactive # ls
ls
above_hispeed_delay
align_windows
boost
boostpulse
boostpulse_duration
go_hispeed_load
hispeed_freq
io_is_busy
max_freq_hysteresis
min_sample_time
target_loads
timer_rate
timer_slack
use_migration_notif
use_sched_load
```

- 计划程序参数
  - 检查/proc/sys/kernel/路径下的所有节点。

```
root@msm8992:/proc/sys/kernel # ls sched_*
ls sched_*
sched_account_wait_time
sched_boost
sched_cfs_bandwidth_slice_us
sched_child_runs_first
sched_cpu_high_irqload
sched_downmigrate
sched_enable_power_aware
sched_freq_account_wait_time
sched_freq_dec_notify
sched_freq_inc_notify
sched_heavy_task
sched_init_task_load
sched_latency_ns
sched_migration_cost_ns
sched_migration_fixup
sched_min_granularity_ns
sched_min_runtime
sched_nr_migrate
sched_power_band_limit
sched_ravg_hist_size
sched_rr_timeslice_ms
sched_rt_period_us
sched_rt_runtime_us
sched_shares_window_ns
sched_small_task
sched_spill_load
sched_spill_nr_run
sched_time_avg_ms
sched_tunable_scaling
sched_upmigrate
sched_upmigrate_min_nice
sched_wakeup_granularity_ns
sched_wakeup_load_threshold
sched_window_stats_policy
```

- CPU频率策略参数
  - Cortex-A53（小群集）– /sys/devices/system/cpu/cpu0/cpufreq/
  - Cortex-A57（大群集）– /sys/devices/system/cpu/cpu4/cpufreq/

```
root@msm8992:/sys/devices/system/cpu/cpu0/cpufreq # ls
ls
affected_cpus
cpuinfo_cur_freq
cpuinfo_max_freq
cpuinfo_min_freq
cpuinfo_transition_latency
interactive
related_cpus
scaling_available_frequencies
scaling_available_governors
scaling_cur_freq
scaling_driver
scaling_governor
scaling_max_freq
scaling_min_freq
scaling_setspeed
stats
```

参见cpufreq.h了解更多信息– Cpufreq.h (/kernel/include/linux)

```
struct cpufreq_policy {
    /* CPUs sharing clock, require sw coordination */
    cpumask_var_t           cpus;   /* Online CPUs only */
    cpumask_var_t           related_cpus; /* Online + Offline CPUs */

    unsigned int            shared_type; /* ACPI: ANY or ALL affected CPUs
                                            should set cpufreq */
    unsigned int            cpu;    /* cpu nr of CPU managing this policy */
    unsigned int            last_cpu; /* cpu nr of previous CPU that managed
                                       * this policy */
    struct cpufreq_cpuinfo  cpuinfo;/* see above */

    unsigned int            min;    /* in kHz */
    unsigned int            max;    /* in kHz */
    unsigned int            cur;    /* in kHz, only needed if cpufreq
                                     * governors are used */
    unsigned int            policy; /* see above */
    struct cpufreq_governor *governor; /* see below */
    void                    *governor_data;
    bool                    governor_enabled; /* governor start/stop flag */

    struct work_struct      update; /* if update_policy() needs to be
                                     * called, but you're in IRQ context */

    struct cpufreq_real_policy      user_policy;

    struct list_head        policy_list;
    struct kobject          kobj;
    struct completion       kobj_unregister;
```

4.14.1 CPU频率策略参数的示例

- 用户报告了相比之前版本浏览器用例15mA的回归。
- 当前版本为v1.3，之前版本为v1.2。
- v1.3中Cortex-A53的Scaling_min_freq为864 MHz，v1.2中Cortex-A53的scaling_min_freq为384 MHz（QTI的默认值）。
- 864 MHz请求Turbo，384 MHz通过时钟计划请求LowSVS
  - v1.2和RCM

```
root@msm8992:/sys/devices/system/cpu/cpu0/cpufreq # cat scaling_min_freq
cat scaling_min_freq
384000
```

  - v1.3

```
root@msm8992:/sys/devices/system/cpu/cpu0/cpufreq # cat scaling_min_freq
cat scaling_min_freq
864000
```

– Cortex-A53时钟计划（小群集）

Performance level	Frequency (MHz)	Source	VDD APC0 requirement
0	300.00	GPLL0	LowSVS
1	384.00	A53PLL	LowSVS
2	460.80	A53PLL	SVS
3	600.00	GPLL0	SVS
4	672.00	A53PLL	Nominal
5	787.20	A53PLL	Nominal
6	864.00	A53PLL	Turbo
7	960.00	A53PLL	Turbo
8	1248.00	A53PLL	SuperTurbo
9 *	1440.00	A53PLL	SuperTurbo

– Cortex-A57时钟计划（大群集）

Performance level	Frequency (MHz)	Source	VDD APC1 requirement
0	300.00	GPLL	SVS
1	384.00	A57PLL	SVS
2	480.00	A57PLL	SVS
3	633.60	A57PLL	SVS
4	768.00	A57PLL	Nominal
5	864.00	A57PLL	Nominal
6	960.00	A57PLL	Nominal
7	1248.00	A57PLL	Turbo
8	1344.00	A57PLL	Turbo
9	1440.00	A57PLL	Turbo
10	1536.00	A57PLL	Turbo
11	1632.00	A57PLL	Turbo
12	1689.60	A57PLL	Turbo
13	1824.00	A57PLL	SuperTurbo

4.15 在测量功耗之前检查温度

1. 确保在同一温度下进行比较测量。

   温度高，电流消耗就会高。

2. 输入以下命令检查温度：

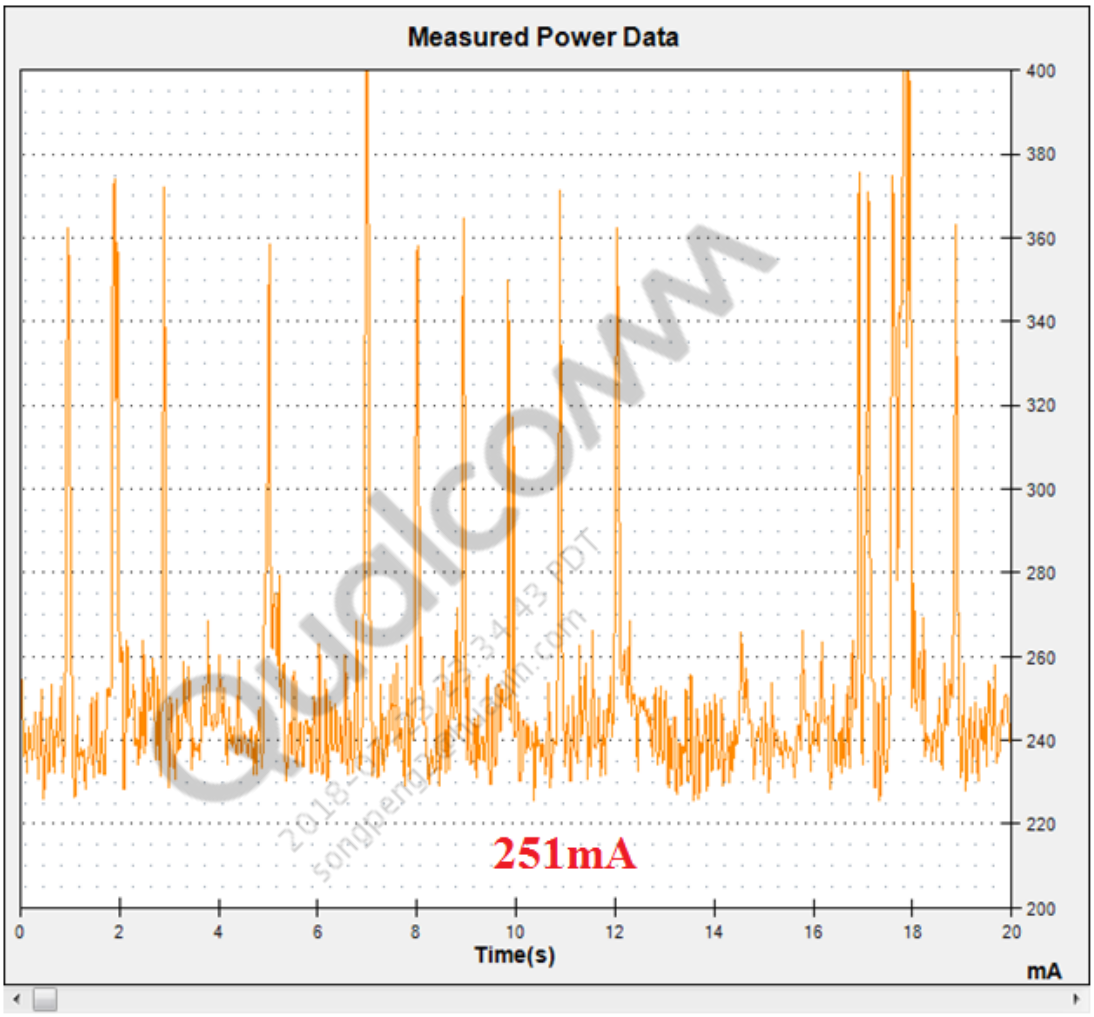
```
Cat /sys/devices/virtual/thermal/thermal_zone8/temp (temperature for
Core 0)
Cat /sys/devices/virtual/thermal/thermal_zone20/temp (temperature for
XO)
```

除图形用例(75 °C)外，大多数（功耗）仪表板用例数据是在25 °C下捕捉的。

4.16 验证Wi-Fi功耗

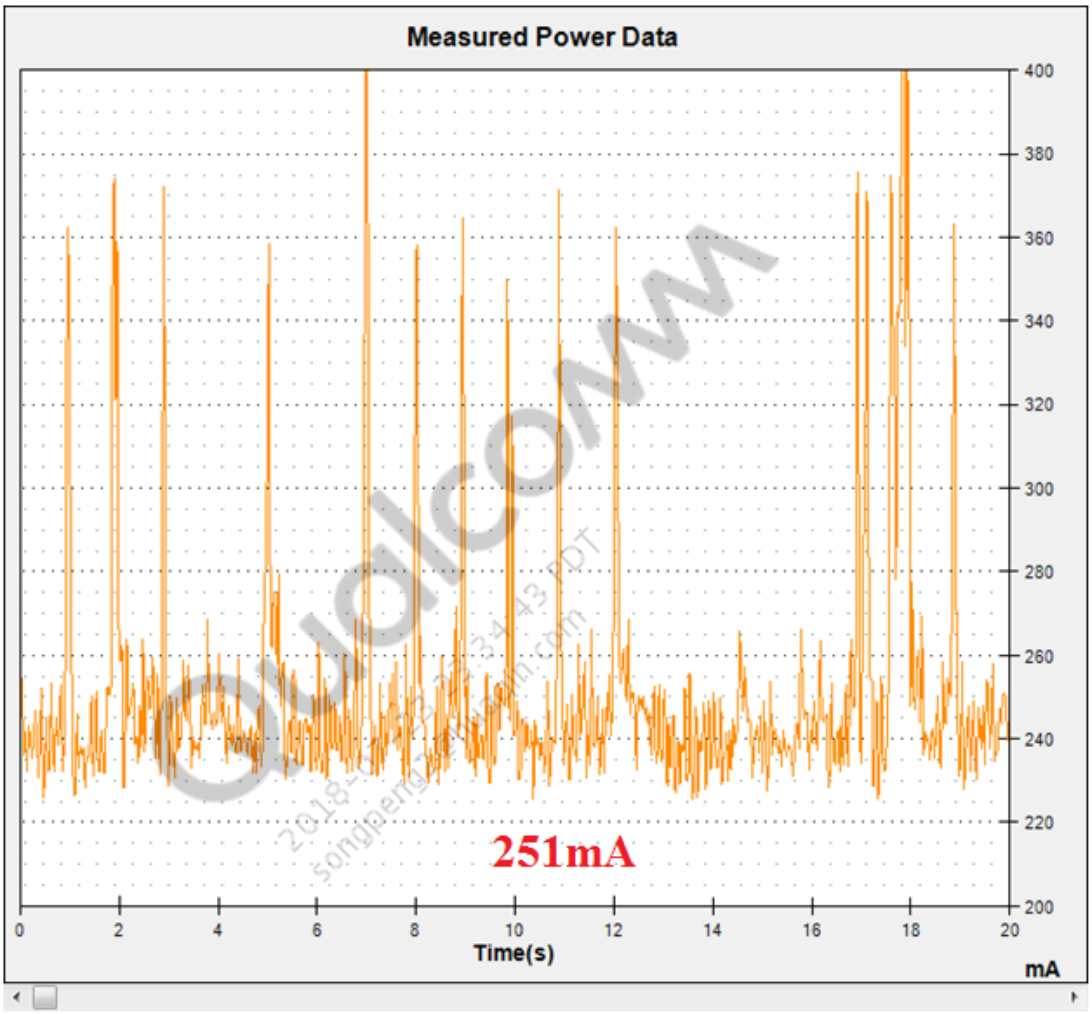
1. 要验证Wi-Fi功耗，使用专用的应用处理器。
2. 用浏览器打开任意页面；查看是否有以下显示Wi-Fi已打开的图标：

3. 页面加载完成后，测量20秒内的功耗。
   - Wi-Fi为开启状态时的波形



4. 关闭Wi-Fi；如下所示没有Wi-Fi图标，表示Wi-Fi已关闭：

5. 再次测量20秒内的功耗。
   - Wi-Fi为关闭状态时的波形



6. 比较Wi-Fi开启和关闭时的功耗数。

   若存在功耗差，表明Wi-Fi功耗为总功耗。

7. 若Wi-Fi功耗高，与Wi-Fi工程师或解决方案提供商商议该问题。

4.17 摄像头预览调试

4.17.1 摄像头子系统时钟

1. 遵循章节5.4中的步骤获取时钟转储。
2. 检查以下时钟：

时钟名称	说明
camss_vfe_vfe0_clk	VFE0时钟
camss_vfe_vfe1_clk	VFE1时钟（如适用）
camss_vfe_cpp_clk	CPP时钟
fd_core_clk	硬件FD时钟（如适用）
bimc_a_clk	BIMC时钟

4.17.2 测量摄像头预览fps

1. 输入以下命令：

```
adb shell setprop persist.camera.hal.debug 1
adb shell setprop persist.camera.sf.showfps 1
```

2. 重启摄像头；在预览中输入以下命令：

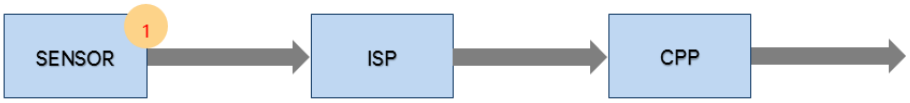
```
adb logcat | grep sensor_pick_resolution
```

以下代码段是为用例选择的分辨率和模式：

```
mm-camera-sensor: sensor_pick_resolution:636 res_idx: 0
mm-camera-sensor: sensor_pick_resolution:636 res_idx: 0
```

以上日志表明传感器驱动已使用读出模式0 (res idx 0)打开了传感器。这通常是传感器的全分辨率模式。

4.17.3 测量用例中的传感器分辨率



1. 输入以下命令：

```
adb shell setprop persist.camera.sensor.debug 1
```

2. 重启摄像头；在预览中输入以下命令：

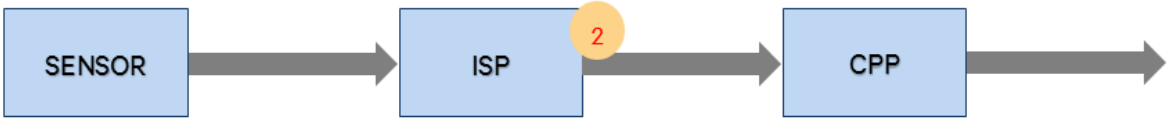
```
adb logcat | grep sensor_get_output_info
```

预览fps日志与以下内容相似：

```
I/mm-camera-sensor(  454): sensor_get_output_info:3422requested dim 0 0
stream mask 0
I/mm-camera-sensor(  454): sensor_get_output_info:3448pick res 2 dim
2104X1560 op clk 319200000
```

摄像头传感器的分辨率输出为2104 x 1560。传感器时钟的分辨率为319200000。

4.17.4 在摄像头用例汇总测量ISP配置



- 输入以下命令：

```
adb logcat | grep  port_sensor_caps_reserve
```

以下代码段显示了已记录的ISP配置：

```
I/mm-camera-sensor(  470): port_sensor_caps_reserve:151ide 10001 stream
type 3 w*h 1920*1080
Stream 3 is the snapshot stream . The above log indicates that ISP is
configured to output a (snapshot) liveshot resolution of 1920*1080.
I/mm-camera-sensor(  470): port_sensor_caps_reserve:151ide 10002 stream
type 4 w*h 3840*2160
Stream 4 is the video stream. The above log indicates that ISP is
configured to output a video stream of size : 1920*1080.
```

```
I/mm-camera-sensor(  470): port_sensor_caps_reserve:151ide 10004 stream
type 1 w*h 1920*1080
Stream 1 is the video stream. The above log indicates that ISP is
configured to output a preview stream of size : 1920*1080.
E mm-camera-isp2: isp_util_decide_stream_mapping:5145 INFO: type 3
resolution 4160x3120 hw_stream 2 need_native_buff 0 controllable_output
0 shared_output 0
```

StreamType 3是摄像头快照流。该流的分辨率为4160 x 3120。

```
E mm-camera-isp2: isp_util_decide_stream_mapping:5145 INFO: type 1
resolution 2048x1536 hw_stream 1 need_native_buff 0 controllable_output
0 shared_output 0
```

StreamType 1 is the camera preview stream. The resolution of this stream is 2048 x 1536.

```
E mm-camera-isp2: isp_util_decide_stream_mapping:5145 INFO: type 11
resolution 640x480 hw_stream 0 need_native_buff 0 controllable_output 0
shared_output 0
```

StreamType 11是FaceDetect流。该流的分辨率为640 x 480。

4.17.5 修改摄像头预览分辨率

要缩小摄像头预览尺寸：

1. 输入以下命令：

```
adb root
adb shell setprop persist.camera.preview.size <value>
```

2. 使用QTI Snapdragon摄像头应用中可用的不同尺寸选项：
   - 值：0 – 根据快照高宽比的默认尺寸
   - 值：1 – 640 x 480
   - 值：2 – 720 x 480
   - 值：3 – 1280 x 720
   - 值：4 – 1920 x 1080

   例如：

```
adb shell setprop persist.camera.preview.size 1 // this will set the
preview to 640x480
```

3. 要重置默认分辨率：

```
adb shell setprop persist.camera.preview.size 0
```

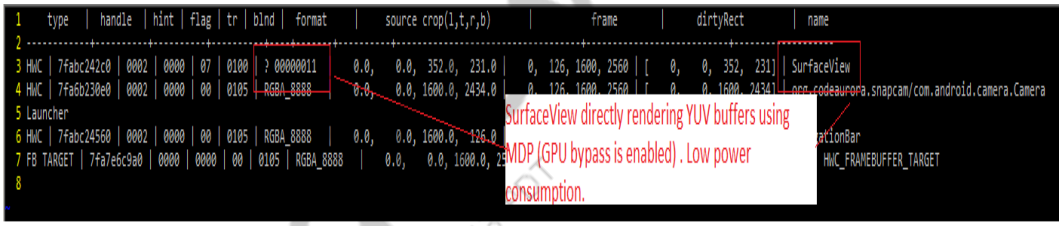
4.17.6 检查利用GPU进行预览缓冲渲染的摄像头

1. 在终端上运行以下命令

```
$adb shell dumpsys SurfaceFlinger
```

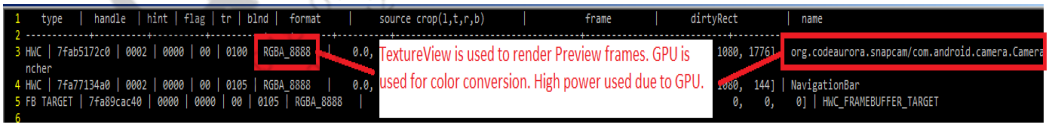
2. 检查颜色格式以及图层中的SurfaceView层。

   - SurfaceView示例



以上Dumpsys表明SurfaceFlinger要渲染YUV缓冲并且通过SurfaceView发送。在该颜色转换示例中未涉及到GPU。

   - TextureView示例



以上Dumpsys表明SurfaceFlinger要渲染RGB缓冲并且通过TextureView发送。其中涉及到了GPU进行颜色转换。在该示例中GPU使用率更高。

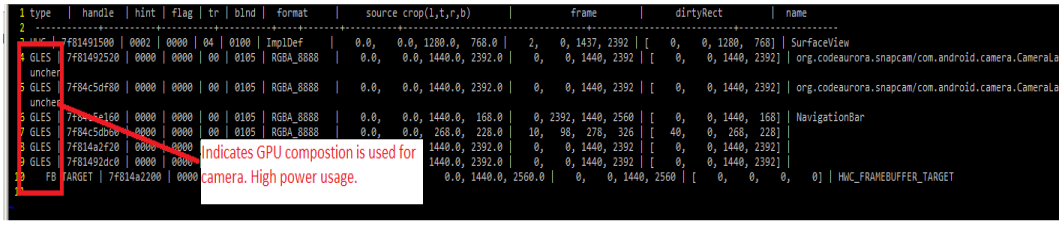
参见章节4.18.2了解使用SurfaceView代替TextureView进行用户界面预览的步骤。

4.17.7 确定摄像头利用GPU进行复合而非MDP/叠加

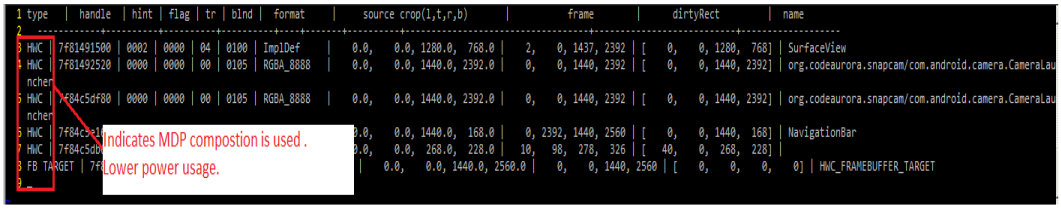
- 在终端上运行以下命令：

```
$Adb shell dumpsys Surfaceflinger
```

   - GPU复合中的SurfaceFlinger dumpsys – GPU用于复合。



- MDP/叠加复合中的SurfaceFlinger dumpsys – GPU未用于复合。



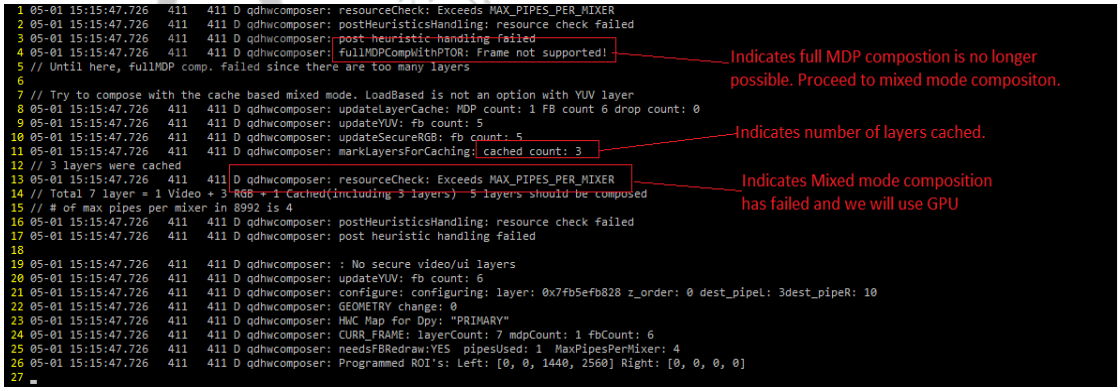
4.17.7.1 确认GPU复合的原因

当要复合的层数大于基于MDP复合可用的管道数/叠加数时将启用GPU复合。当应用创建了多个层并且发送以进行硬件复合时才会出现这种情况。

1. 在显示模块中启用以下日志：

```
adb shell service call display.qservice 15 i32 1 i32 1
adb shell "echo 'file mdss_mdp_ctl.c +p' > /d/dynamic_debug/control"
adb shell "echo 'file mdss_mdp_overlay.c +p' > /d/dynamic_debug/control"
adb shell "echo 'file mdss_mdp_pipe.c +p' > /d/dynamic_debug/control"
```

2. 重新运行摄像头用例并捕捉logcat日志。
3. 在logcat中检查以下日志模块。



4.18 摄像头功耗优化技术

4.18.1 双ISP配置

Snapdragon最近推出的一些芯片组支持两个图像处理内核，可以将Bayer Input加工成YUV。传感器分辨率较高时，摄像头软件会自动将传感器输出分别拆分并传送给两个ISP。

使用分辨率较高的传感器时，QTI建议OEM启用双ISP功能，这样ISP处理的数据便会平均拆分给两个而不是一个ISP硬件块。

- 启用双ISP – adb shell setprop persist.camera.isp.dualisp 1

- 禁用双ISP – adb shell setprop persist.camera.isp.dualisp 0

为分辨率较低的传感器启用双ISP并无益处，有时还可能造成高功耗。QTI建议按照下表所示指南进行选择。

遵循以上指南了解哪些传感器分辨率必须启用双ISP（假定30 fps ZSL预览）。

MSM	单ISP (MP)	双ISP (MP)
MSM8996	≤ 8	> 8
MSM8994	≤ 13	> 13
MSM8992	≤ 13	> 13
APQ8084	≤ 13	> 13
MSM8952	≤ 8	> 8
MSM8956	≤ 8	> 8

4.18.2 使用SurfaceView代替TextureView

摄像头应用可以使用TextureView或SurfaceView渲染摄像头预览缓冲。若使用TextureView，则摄像头预览区域是Android视图层次的一部分。但是，这会使得预览帧通过GPU为YUV进行ARGB转换。用户界面层和预览帧与GPU混合相比使用SurfaceView功耗更高。

若使用SurfaceView，显示硬件会将预览内容直接复合为重叠，不涉及GPU。

有关详细信息，访问https://source.android.com/devices/graphics/architecture.html#surface-or-texture。

Snapdragon摄像头应用默认使用SurfaceView。参见以下提交将Snapdragon摄像头应用所使用的TextureView更新为SurfaceView：

https://www.codeaurora.org/cgit/quic/la/platform/packages/apps/SnapdragonCamera/commit/?h=caf/LA.BF64.1.1_rb1.18&id=6ee7e415bf6f73d423df8da68f72fa6fdabde714

要将摄像头应用从使用SurfaceView转换为使用TextureView进行摄像头预览：

1. 将TextureView替换为SurfaceView。
2. 将SurfaceTexture替换为SurfaceHolder。
3. 将TextureView.SurfaceTextureListener修改为SurfaceHolder.Callback。
4. 将TextureView.setSurfaceTextureListener(TextureView.SurfaceTextureListener修改为surfaceHolder = surfaceView.GetHolder();
surfaceHolder.addCallback(SurfaceHolder.Callback)。
5. 将摄像机的setPreviewTexture(SurfaceTexture)替换为setPreviewDisplay(SurfaceHolder)。
6. 为了达到适宜缩放，在布局类型中设置SurfaceView，SurfaceView不应是根<merge>标记的一部分。

7. 当表面进行如下变化时，若预览尺寸与屏幕不完全匹配，在SurfaceView路径中手动将预览重新缩放至正确的高宽比。

```
mSurfaceView.GetLayoutParams().width = (int) correctWidth;
mSurfaceView.GetLayoutParams().height = (int) correctHeight;
mSurfaceView.requestLayout();
```

不会对图像质量或性能有不良影响。若应用在摄像头预览数据中使用了复合变换和动画效果，则无法使用SurfaceView。

4.18.3 缩小摄像头预览尺寸

若传感器模式设置的分辨率大于显示屏尺寸或MDP硬件支持放大，考虑为摄像头请求较小的预览尺寸。

将预览尺寸缩小到最佳尺寸可以降低从ISP到DDR总线以及从DDR总线到MDP的流量。这可以降低摄像头预览用例的功耗。

不会对图像质量或性能有不良影响。

4.18.4 摄录像机用例中的CPP镜像

可以同颜色格式一起对预览和视频尺寸进行不同的配置。若预览或视频尺寸之间或颜色格式存在差异，则CPP会采用双道处理路径，这会加大功耗。

OEM可以将预览和视频配置为相同的格式和分辨率从而避免CPP进行双道处理。

若以下日志在adb中输出，则软件可以触发用例的CPP输出复制功能：

```
cpp_module_set_output_duplication_flag:643 linked streams formats match:
output duplication enabled
```

预览和视频流应用了相同的图像质量设置。除此之外，该更改不会带来负面影响。

4.18.5 禁用时域降噪(TNR)

TNR功能适用于选定芯片组。该算法的复杂程度取决于图像帧的分辨率。禁用TNR可以在分辨率较高的用例（如4K录制和预览帧）中降低DDR带宽和总功耗。

- 要禁用TNR，输入以下命令：

```
adb shell setprop persist.camera.tnr.preview off
adb shell setprop persist.camera.tnr.video off
```

在完成更改后重新运行IQ和性能测试。该更改会影响图像质量。

4.18.6 将传感器模式分辨率设置为视频分辨率

1. 在摄录像机录制用例中将传感器模式分辨率设置为与视频录制分辨率（而非全分辨率）相同以降低功耗。例如，将1080p视频摄录像机录制的13 MP传感器模式设置为1080p。
2. 同样，在非ZSL预览用例中将传感器模式设置为显示尺寸或较低尺寸以减低功耗。

QTI建议，若终端功能集允许或可以对各自的分辨率尺寸进行调试，则在摄录像机和摄像头预览用例中将传感器模式设置为较低分辨率。用户可以在各自的传感器驱动中进行更改。

预计不会产生影响。预览帧以较低的分辨率输出。

4.18.7 禁用无色

- 要禁用无色以降低功耗，输入以下命令：

```
adb setprop persist.camera.tintless=disable
```

在完成更改后重新运行IQ和性能测试。该更改会影响图像质量。

4.18.8 禁用色差校正(CAC)

CAC区块是摄像头ISP的特殊硬件块。可以在某些高端芯片组上看到以提供精细的噪音消除性能。在预览或预览和快照时禁用该功能可以实现节能。

在Chromatix™颜色优化工具(Chromatix)文件中禁用CAC。提交用例以获取禁用CAC功能的步骤。

从摄像头预览中禁用CAC操作后检查是否对图像质量有影响

4.19 视频录制功耗优化技术

4.19.1 编码器节能模式

编码器节能模式是单会话4K H.264摄录像机支持的一种可选模式功耗优化功能。

- 要启用编码器节能模式，输入以下命令：

```
adb shell setprop vidc.debug.perf.mode 2
```

该功能在OMX_QcomIndexConfigVideoVencPerfMode OMX扩展中公开。

参见*Video Power Encoder Save Mode and Encoder DCVS* (80-NV307-1)了解关于该功能的详细信息。

该选项仅适用于UHD(4K)编码。还需要在Nominal时钟而非Turbo上运行VFE。

该模式可能会对编码的质量有一定的影响。

5 日志收集

5.1 RPM日志

RPM将小型日志发布到备份消息RAM的有限区域内。日志的实体格式为uLog格式，供各种其他日志使用。它是一个环形缓冲区，一般大小为4 kB。它是一种原始日志，每条消息具有一组ID和可变数目的参数。

5.1.1 保存RPM RAM转储

在MSM8994中，保持JTAG菊花链的连接没有必要禁用RPM终止。

要保存RPM RAM转储：

1. 打开RPM T32并用sys.m.a命令连接。
2. 创建RPM转储需要的问题场景。
3. 在所需位置断开T32并运行以下脚本以保存RPM日志：

```
do <RPM Build location>\rpm_proc\core\bsp\rpm\scripts\rpm_dump.cmm
<Location to save dumps>
```

5.1.2 将RPM转储加载到T32上并提取日志

将RPM转储加载到T32模拟器上以进行进一步分析：

1. 打开T32模拟器并运行sys.up。
2. 运行以下脚本：

```
do <RPM Build location>\rpm_proc\core\bsp\rpm\scripts\rpm_load_dump.cmm
   <Location of logs>
```

3. 加载RPM.elf开始调试。

```
d.load.elf <RPM_Build>\rpm_proc\core\bsp\rpm\build\RPM_XXXXXXXXX.elf
   /nocode /noclear
```

5.1.2.1 使用T32的RPM外部日志（RPM uLog）

1. 连接到RPM、在断开状态中或RPM RAM转储加载到T32模拟器时，在T32中运行以下命令：

```
do <RPM_build>\rpm_proc\core\power\ulog\scripts\rpm_ulogdump.cmm
   <Location to save logs>
```

这会将RPM外部日志置于所需日志目录中。RPM外部日志需要使用Python解析工具中断其内容。

2. 利用命令提示符使用以下提取的日志的Python脚本：

```
Python \\<RPM_build_location>\rpm_proc\core\power\rpm\debug\scripts\
rpm_log_bfam.py -f "RPM External Log.ulog"
```

5.1.2.2 使用T32的RPM NPA日志

连接到RPM、在断开状态中或RPM RAM转储加载到T32模拟器时，在T32中运行以下命令：

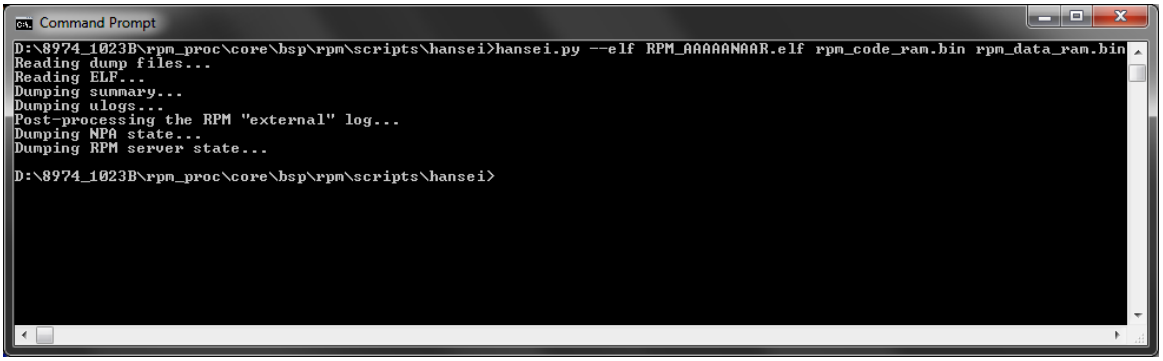
```
do <RPM_build>\rpm_proc\core\power\npa\scripts\rpm_npadump.cmm
<Location to save logs>
```

5.1.2.3 使用Hansei脚本的RPM日志

Hansei – 一种RPM RAM转储解析器工具，可以在rpm_proc\core\bsp\rpm\scripts\hansei文件夹下找到（需要Python 2.7.2版本）。

```
Usage - hansei.py [-h] --elf rpm.elf [--output path] dumpfile
[dumpfile ...]

Example - hansei.py -elf rpm.elf -o . rpm_code_ram.bin rpm_data_
ram.bin rpm_msg_ram.bin
```



- 输出文件摘要：
  - rpm-summary.txt – 包含关于RPM健康状态的一般信息，包含核心转储状态和各种故障信息
  - rpm-log.txt – 后处理RPM外部日志
  - rpm-rawts.txt – 后处理RPM外部日志和原始时间戳
  - npa-dump.txt – 标准NPA转储格式
  - ee-status.txt – 包含关于活跃或睡眠的子系统（及其内核）的信息
  - reqs_by_master/* – 包含每个执行环境文件的文件夹，详细描述了EE为RPM准备的所有电源请求
  - reqs_by_resource/* – 包含每个注册RPM服务器的资源类型文件夹的文件夹结构，该文件夹下是包含所有该类型的每个资源请求的文件

5.2 收集GPIO转储

硬件中的所有GPIO都使用以下脚本（可以从RPM T32窗口读出）读取硬件中所有GPIO的电流配置。

```
do <Modem_Build>\modem_proc\core\systemdrivers\tlmm\scripts\
tlmm_gpio_hw.cmm
```

1. 选择**Option 14: Read All GPIO Configurations**。
2. 对于软件中存储的默认睡眠配置，使用以下脚本（从TLMM.xml读出）读取：

   ```
   do <Modem_Build>\modem_proc\core\systemdrivers\tlmm\scripts\
   tlmm_sleep_configs.cmm
   ```

5.3 收集PMIC转储

1. 打开RPM T32窗口。
2. 输入sys.m.a将T32连接到在试终端。
3. 从<RPM Build>/rpm_proc/core/bsp/rpm/build/*.elf加载.elf文件，并在必要时设置断点。
4. 重新创建用例场景。
5. 通过以下方式之一断开T32：
   - 点击T32用户界面中的**Pause**。
   - 按下**F8**键。
   - 使用用户定义的断点。

6. 到达断点后，输入以下命令：

```
CD.DO <RPM_Build>/rpm_proc\core\systemdrivers\pmic\scripts\PMICDump.cmm
```

默认情况下，原始PMIC转储文件保存在c:\temp\pmicdump.xml中。

7. 输入以下命令解析pmicdump.xml文件：

```
python PMICDumpParser.py --flat=<RPM
BUILD>\rpm_proc\core\systemdrivers\pmic\scripts\pm8994\v1_1\CORE_ADDRESS
_FILE_CUSTOMER.FLAT --file=pmicdump.xml > pmic_parsed.txt
```

5.4 收集时钟转储

若终端具有JTAG性能，使用JTAG方法。若终端不具有JTAG，使用ADB方法获取时钟转储。

5.4.1 利用JTAG收集时钟转储

1. 打开RPM T32窗口。
2. 输入sys.m.a将T32连接到在试终端。
3. 从<RPM Build>/rpm_proc/core/bsp/rpm/build/*.elf加载.elf文件，并在必要时设置断点。
4. 重新创建用例场景。
5. 通过以下方式之一断开T32：
   - 点击T32用户界面中的**Pause**。
   - 按下**F8**键。
   - 使用用户定义的断点。
   - 输入以下命令：

```
do <Build_Location>\rpm_proc\core\systemdrivers\clock\scripts\
msm8994\testclock.cmm
```

6. 要获取所有时钟的转储，在弹出窗口中输入**all**。

   时钟转储在T32的消息区中打印。

5.4.2 使用adb shell收集时钟转储

1. 连接USB。
2. 运行以下命令：

```
adb shell
mount -t debugfs none /sys/kernel/debug
sleep 10 && while true; do echo
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=
\=\=\=\=\=\=\=\=\=\=\=\=; cat /proc/uptime; cd /sys/kernel/debug/clk;
for i in *; do if [ -d $i ]; then if [ "$(cat $i/enable)" == "1" ]; then
if [ -e $i/measure ]; then echo $i \=\> enable:`cat $i/enable`
measure:`cat $i/measure`; else echo $i \=\> enable:`cat $i/enable`
rate:`cat $i/rate`; fi; fi; fi; done; echo \-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-; cd
/sys/class/regulator; for i in *; do if [ -d $i ]; then if [ -e
$i/state ]; then if [ "$(cat $i/state)" == "enabled" ]; then if [ -e
$i/microvolts ]; then echo $i \=\> name:`cat $i/name` state:`cat
$i/state` microvolt:`cat $i/microvolts`; else echo $i \=\> name:`cat
$i/name` state:`cat $i/state` microvolt: N\/A; fi; fi; fi; fi; done;
sleep 2; done > /data/dumpclk.txt &
```

3. 出现PID时拔出USB。
4. 运行测试场景<执行测试场景至少2分钟>。
5. 连接USB并输入以下命令取消时钟检查进程ID。

```
adb shell ps | grep [PID]
adb shell "kill [PID]"
adb pull /data/dumpclk.txt
```

5.5 收集Modem日志

5.5.1 收集Modem用户日志

- 在Modem T32窗口中执行以下源于Modem_Build的脚本：

```
do  <Modem_Build>\modem_proc\core\power\ulog\scripts\UlogDump.cmm
<location to save logs>
```

这会将Modem用户日志置于指定目录中。

5.5.2 收集Modem NPA日志

- 在Modem T32窗口中执行以下源于Modem_Build的脚本：

```
do  <Modem_Build>\modem_proc\core\power\npa\scripts\NPADump.cmm
<Location to save logs>
```

这会将Modem NPA日志置于指定目录中。

5.6 收集Modem F3消息/QXDM Pro日志

1. 通过USB将终端连接到安装有QXDM Pro的PC。
2. 通过“File > Load”配置加载日志掩码（dmc文件）。
3. 按下**F3**键或从“View”选项卡中选择**Messages View**。
4. 按下**Alt+L**开始记录，再次按下**Alt+L**停止记录。
5. 从“Options > Log View Config > Misc > Log File Path”定义的位置获取保存的日志。

5.7 捕获ftrace日志

注意： 本节进行了大量更改。

1. 连接USB。
2. 输入以下命令：

```
adb root
adb remount
adb shell
cd /sys/kernel/debug/tracing
echo 0 > tracing_on;
echo 100000 > buffer_size_kb;
```

3. 输入以下命令检查buffer_size_kb：

```
Cat buffer_size_kb
```

4. 要收集ftrace日志，输入以下命令：

```
echo "" > set_event
echo "" > trace
sync
echo power:cpu_idle power:cpu_frequency power:cpu_frequency_switch_start
msm_low_power:* sched:sched_cpu_hotplug sched:sched_switch
```

```
sched:sched_wakeup sched:sched_wakeup_new sched:sched_enq_deq_task >>
set_event
echo power:clock_set_rate power:clock_enable power:clock_disable
msm_bus:bus_update_request >> set_event
echo irq:* >> set_event
echo mdss:mdp_mixer_update mdss:mdp_sspp_change mdss:mdp_commit >>
set_event
echo kgsl:kgsl_pwrlevel kgsl:kgsl_buslevel kgsl:kgsl_pwr_set_state >>
set_event
sleep 10 && echo 0 > tracing_on && echo "" > trace && echo 1 >
tracing_on && sleep 10 && echo 0 > tracing_on && cat trace >
/data/local/trace.txt &
```

5. 出现PID（进程ID）时拔出USB。
6. 运行用例。
7. 完成用例后，连接USB并输入以下命令将终端中的追踪文件置于您的PC中：

```
adb pull /data/local/trace.txt C:\<location>
```

5.8 捕捉PowerTop和Top数据

1. 连接USB。
2. 输入以下命令捕捉PowerTop和Top数据：

```
adb root
adb remount
adb shell
```

3. 输入以下命令开始捕捉PowerTop和Top数据：

```
sleep 3 && while true;
do echo
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=
\=\=\=\=\=\=\=\=\=;
cat /proc/uptime;
top -m 25 -d 1 -n 1 -t;
echo \-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-;
/data/powertop -r -d -t 5
done > data/dumptop.txt &
```

4. 出现PID（进程ID）时断开USB。
5. 运行用例。

6. 完成用例后，连接USB并输入以下命令取消进程。

```
adb shell "kill [PID]"
adb pull /data/ dumptop.txt  C:\<location>
```

5.9 捕捉时钟和调节器转储

捕捉Top和PowerTop数据：

1. 连接USB。
2. 输入以下命令：

```
adb root
adb remount
adb shell
```

3. 输入以下命令开始捕捉PowerTop和Top数据：

```
sleep 3 && while true;
do echo
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=;
cat /proc/uptime;
cd /sys/kernel/debug/clk;
for i in *;
do if [ -d $i ];
then if [ "$(cat $i/enable)" == "1" ];
then if [ -e $i/measure ];
then echo $i \=\> enable:`cat $i/enable` measure:`cat $i/measure`;
else echo $i \=\> enable:`cat $i/enable` rate:`cat $i/rate`;
fi; fi; fi; done;
echo \-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-;
cd /sys/class/regulator; for i in *;
do if [ -d $i ];
then if [ -e $i/state ];
then if [ "$(cat $i/state)" == "enabled" ];
then if [ -e $i/microvolts ];
then echo $i \=\> name:`cat $i/name` state:`cat $i/state` microvolt:`cat
$i/microvolts`;
else echo $i \=\> name:`cat $i/name` state:`cat $i/state` microvolt:
N\/A; fi; fi; fi; fi; done;
sleep 2;
done > /data/dumpclk.txt &
```

4. 出现PID（进程ID）时断开USB。
5. 运行用例。

6. 完成用例后，连接USB并输入以下命令取消进程。

```
adb shell "kill [PID]"
adb pull /data/ dumpclk.txt  C:\<location>
```

5.10 采集唤醒锁信息

- 输入以下命令之一：

```
adb shell
cat /sys/kernel/debug/wakeup_sources
```

或

```
adb shelldumpsys power
```

A 参考

A.1 相关文档

标题	编号
Qualcomm Technologies, Inc.	
System Power Monitor Version 4 Application Note	80-N6594-16
Power Consumption Measurement Procedure for MSM (Android-Based)/MDM Devices	80-N6837-1
Presentation: Graphics Power and Performance Overview	80-NP885-1
Video Power Encoder Save Mode and Encoder DCVS	80-NV307-1

A.2 缩写和术语

缩写或术语	定义
APSS	应用处理器子系统(Applications Processor Subsystem)
CAC	色差校正
CAF	Code Aurora论坛
DRX	非连续性接收
LPASS	低功耗音频子系统(Low Power Audio Subsystem)
MPSS	Modem外围子系统
参考日志	从电流消耗达到特定芯片组目标的参考终端中获取的调试日志
spm	系统功耗监控器
YUV	亮度、带宽、色度；又称为YCbCr和YPbPr