Qualcomm Technologies, Inc.

Extensive Power Debug Guide

80-P0955-1 B

August 5, 2015

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

Revision history

Revision	Date	Description
A	June 2015	Initial release
B	August 2015	Updated Sections 4.5 and 5.7

Contents

Figures

1 Introduction

1.1 Purpose

This document provides details on how to optimize specific power test cases and debug issues related to the RPM, APSS, multimedia, and modem.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

Shading indicates content that has been added or changed in this revision of the document.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If all the suggested debugging methodologies do not resolve the issues, file a case with the following information:

- Correct chipset – AMSS build ID and Operating System (OS)
- Initial problem type – Software
- For Multimedia use cases:
  - Problem Area 1 – Multimedia
  - Problem Area 2 – Power
  - Problem Area 3 – Use-case specific
    - Audio, video, graphics, browsing, sensors, etc.
- For core and modem use cases:
  - Problem Area 1 – BSP/HLOS
  - Problem Area 2 – Power/Thermal (BSP/HLOS)
  - Problem Area 3 – Use-case specific
    - Power-Modem, Power-Idle Power, etc.

- Problem description field
  - Detailed information about the problem
    - Details of the use case if different from the QTI standard use case
    - Steps to reproduce the issue
  - All information about the debugging done up to that point
  - All logs suggested in debugging: Waveforms, rail-level breakdown, NPA dumps, kmsg, Top, PowerTop, clock dump, SurfaceFlinger, ftrace, systrace, logcat, etc.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Debug setup

Before engaging in power debugging, QTI recommends using the following tools to ensure a complete and effective power debugging process.

2.1 Required software tools and setup

2.1.1 Debugging tool setup for use cases involving RPM and modem

- Trace32 (T32) software – This is essential for using JTAG for power debugging, especially for obtaining on-target logs, loading RAM dumps, and capturing off-target logs using a T32 simulator.
- Licensed Qualcomm Product Support Tool (QPST) and QXDM Professional™ (QXDM Pro) tool – These tools are necessary to obtain logs and messages pertaining to modem power use case debugging.

2.1.2 Debugging tool setup for use cases involving the apps processor

Tools	Precondition	Install	Where to find
PowerTop	NA	adb root adb remount adb push <PowerTop location>\powertop /data/ adb shell chmod 777 /data/powertop	Through a Salesforce case
PerfTop	NA	adb root adb remount adb push <perf location>\perf /data/ adb shell chmod 777 /data/perf	Through a Salesforce case
Pytime chart	Pythonxy tool; verify that ETS and pythonxy are selected for installation	https://code.google.com/p/pythonxy/wiki/Downloads After installation, open a command prompt in C:\ and run easy_install pytimechart.	https://code.google.com/p/pythonxy/wiki/Downloads
Systrace	SDK tool	http://developer.android.com/tools/sdk/tools-notes.html	Android SDK toolkit
msmbusvoting	adb root adb remount	No installation required	Through a Salesforce case

2.2 Required hardware tools

- JTAG – This is required to obtain on-target logs, such as clock dumps, PMIC dump, and GPIO dumps when doing power optimization or debugging a power issue.
- Power monitor – This is required with a minimum of 5 kHz sampling rate to accurately measure use case power consumption and for waveform analysis.

2.3 Power breakdown board

- Detailed breakdown measurements are essential for power debugging and help to quickly narrow down higher power consumption issues.
- Build a board that has the capability of measuring rail level current and voltage by using the System Power Monitor (SPM); see *System Power Monitor Version 4 Application Note* (80-N6594-16).

3 Dashboard use case debugging flow

Before starting with power debugging, it is necessary to have all the software and hardware tools required for debugging and power optimization purposes; see Chapter 2.

NOTE: References to steps in the following tables refer only to the steps within that table, unless otherwise specified.

3.1 Rockbottom

Step no.	Step	Reference

1	▪ To make a proper comparison with the QTI reference data, the hardware configuration also must be comparable.
	▪ Current consumption for any sensors or external components must be quantified for that purpose.
	▪ Quantify the current consumed by the following; this must be accounted for as the known delta:
	  ▫ Sensors and other third-party components
	  ▫ Different DDR size compared to the QTI reference used in the device
	▪ For this information, file a case.
		Section 1.3

2	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ Compare it with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 1. ▪ If the customer device measurement data minus the known delta is greater than the QTI reference data, go to Step 3.	Release Notes

3	▪ Capture the battery level current waveform and check what is causing the higher power consumption.
	  ▫ For a higher base current, go to Step 4.
	  ▫ For occurrences of unexpected wakeups, go to Step 7.
		Section 4.1.1

4	▪ For debugging a higher base current, verify that the device enters VDD minimization mode.
	  ▫ If the device is not in VDD minimization mode, go to Step 5.
	  ▫ If the device is in VDD minimization mode, go to Step 6.
		Section 4.1.2

5	▪ Check the subsystem status to determine which one is blocking the device from entering VDD minimization mode and debug further for that subsystem.	Section 4.1.3 Section 4.1.4

Step no.	Step	Reference
6	▪ File a case to obtain the following: ▫ A procedure to check VDD_CORE, VDD_MEM retention voltages of the device under test ▫ For comparison, PMIC dumps and GPIO reference logs (debug logs obtained from a reference device where the current consumption meets the goals for the particular chipset) ▪ Capture PMIC dumps on the device under test and compare them with the QTI reference PMIC logs to determine if any unused SMPS and LDOs are left on. Turn off any SMPS and LDOs. ▪ Capture GPIO dumps and compare with the QTI reference GPIO logs to determine if the GPIO configuration is as expected. For GPIOs used differently compared to the QTI reference design, sleep configuration must follow the custom design. ▪ Capture rail level voltage and current data and compare with the QTI reference breakdown data to determine the rails consuming higher current and debug further.	Section 4.1.7 Specific chipset application note for power breakdown reference
7	▪ For debugging unexpected wakeups, check which subsystem is causing those wakeups by monitoring subsystem rails and checking subsystem specific logs, for example, modem NPA logs, RPM NPA logs, or Application Processor Subsystem (APSS) kernel logs.	Section 4.1.6
8	▪ If power consumption is still higher than expected after following Steps 1 through 7, file a case with QTI for further debugging help.	Section 1.3

3.2 Standby

Step no.	Step	Reference
1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta: ▫ Sensors ▫ Different DDR size – Request estimated value through a case ▫ Other third-party components ▪ For this information, file a case.	Section 1.3
2	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 3.	Release Notes
3	▪ Capture the battery level current waveform and determine what is causing the higher power consumption. ▫ For a higher base current, go to Step 6 of Section 3.1. ▫ For occurrences of unexpected wakeups, go to Step 7 in Section 3.1. ▫ If DRX wakeup power consumption is higher, go to Step 4.	
4	▪ For debugging higher current consumption in the DRX wakeup period, compare with the reference waveforms and determine the cause from the following: ▫ For a longer awake period, go to Step 5. ▫ For a higher awake amplitude, go to Step 6.	

Step no.	Step	Reference
5	▪ If the DRX wakeup timeline is longer, reconfirm the call box settings, for example, the device does intra- or inter-RAT searching if the neighbor cells are enabled, which in turn causes the wakeup timeline to increase.	Section 4.2.1
6	▪ If the wakeup timeline is close to the reference timeline and the awake period amplitude is higher, i.e., higher peak current, determine the following: ▫ If the proper calibration is done for that particular RAT and band configuration ▫ If APT/ET enablement and calibration is done correctly ▪ If using a third-party PA, the current consumption must be quantified for the same.	Section 4.2.1
7	If power consumption is still higher than expected after following Steps 1 through 6, file a case with QTI for further debugging help.	Section 1.3

3.3 Static display

Step no.	Step	Reference
1	▪ If display panel power can be quantified, go to Step 2. ▪ If display panel power cannot be quantified, go to Step 3.	
2	▪ Compare the following device components to the QTI reference setup: ▫ Panel resolution ▫ Smart/dumb panel ▫ Any other additional OEM-specific components ▪ File a case with QTI for power impact of any of the above components that vary from the standard QTI reference setup.	
3-1	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure. If the known delta from Step 2 cannot be quantified, go to Step 4.	80-N6837-1
3-2	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta quantified in Step 2.	Release Notes
4	▪ If using a smart panel, check if the device enters VDD minimization mode. If the device does not enter VDD minimization, go to Step 5. ▪ If the device enters VDD minimization, go to Step 15. No clocks are active if device enters VDD minimization. ▪ If using a dumb panel, go to Step 6.	Section 4.1.2
5	▪ Resolve the root cause for the device not entering VDD minimization and go to Step 15.	Section 4.5
6	▪ Observe the power waveform pattern and identify if the baseline current is high, there are frequent wakeups, or both are observed. ▫ If the baseline current is high, go to Step 7. ▫ If frequent wakeups are observed, go to Step 14.	

Step no.	Step	Reference
7	▪ If a device has breakdown capability, capture a full power rail breakdown and compare it with the QTI reference data to identify if any rail is consuming higher current. ▪ Capture SurfaceFlinger and compare it with SurfaceFlinger reference logs to check for total number of layers, composition type, and number of layers being updated.	Section 4.6
8	▪ Identify the subsystems on that particular rail found in the specific chipset power overview document. Collect and debug clock information for those subsystems following the instructions in Steps 9 through 13.	
9	▪ Capture and compare clock dumps with the static image clock plan to determine if any clocks are high or any additional clocks are enabled.	Section 5.4 Specific chipset dashboard goals and use case clock plan document
10	▪ If the CPU clock is high, capture ftrace, PowerTop, and Top to understand CPU residency and CPU load information.	Section 4.9
11	▪ Determine which process has high CPU usage and look for any unexpected processes/interrupts compared to the reference logs. ▪ Resolve the abnormal process/thread by contacting the respective area engineer.	Section 4.9
12	▪ If BIMC clock is high, compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the reference logs. ▪ If reference logs are needed, file a case. ▪ After a particular client that voted for more bandwidth is identified, contact the respective subsystem engineer to resolve this.	Section 4.13
13	▪ If another clock is high, file a case with all the debug information, for example, clock dump, ftrace, waveform, PowerTop, Top, and power rail breakdown.	Section 1.3
14	▪ For frequent wakeups, check PowerTop for any unexpected interrupts compared to the reference logs. ▪ Use pytime chart to analyze ftrace to determine the source of the interrupt. ▪ Resolve interrupts by contacting the respective engineer for the domain.	Section 4.10
15	▪ Ensure that the rockbottom use case is optimized, or subtract the delta between the QTI rockbottom use case and customer device.	Section 3.1
16	▪ Collect and compare the GPIO and PMIC LDO configurations with the QTI reference.	Section 4.1.7
17	▪ GPIO and LDO configurations are dependent on the hardware design. ▪ Captured GPIO and LDO data must be reviewed by QTI and OEM hardware teams to turn off unnecessary components. ▪ If there are still issues, go to Step 13.	

3.4 MP3 playback

Step no.	Step	Reference
1	▪ To make a proper comparison with the QTI reference power data, the hardware configuration also must be comparable. ▪ Check for any custom components on the device. ▫ Tunnel mode vs Nontunnel mode ▫ Sensors and other third-party components ▫ Different DDR size compared to the QTI reference ▫ Any additional postprocessing hardware ▪ Disable all of the above or account for the current delta from each of these components as the known delta. ▪ For this information, file a case.	Section 1.3
2	▪ Check Tunnel mode.	Section 4.7
3	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ If using Tunnel mode audio playback, compare the measured number with the QTI reference data to determine if power consumption is higher, considering the known delta as quantified in Step 1. ▪ If using Nontunnel mode, request the reference power number by filing case with QTI.	Release Notes
4	▪ Observe the power waveform pattern and identify if the baseline current is high, there are frequent wakeups, or both are observed. ▫ If the baseline current is high, go to Step 5. ▫ If frequent wakeups are observed, go to Step 11.	
5	▪ If a device has breakdown capability, capture a full power rail breakdown and compare it with the QTI reference to identify if any rail is consuming higher current. ▪ If a breakdown cannot be captured, go to Step 7.	Specific chipset dashboard goals and use case clock plan document
6	▪ Identify which power rail is higher than the QTI power data and determine the subsystems using it; look only for these components in Steps 7 through 16.	Specific chipset power overview document
7	▪ Capture full clock dump to compare with the QTI reference clock plan for MP3.	Specific chipset dashboard goals and use case clock plan document
8	▪ If the CPU clock is high, capture ftrace, PowerTop, and Top to understand CPU residency and CPU load information.	Section 4.9
9	▪ Determine which process has high CPU usage and look for any unexpected processes/interrupts compared to the reference logs. ▪ Resolve the abnormal process/thread by contacting the respective area engineer.	Section 4.9
10	▪ If the BIMC clock is high, compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the QTI reference data. ▪ File a case for the QTI reference data. ▪ After a particular client that voted for more bandwidth is identified, contact the respective subsystem engineer to resolve this.	Section 4.13 Section 1.3
11	▪ For frequent wakeups, check the period of the wakeup. ▪ Capture PowerTop and ftrace and look for periodic wakeups.	Section 4.10

Step no.	Step	Reference
12	▪ Check PowerTop for any unexpected interrupts compared to the reference log. ▪ Use pytime chart to analyze ftrace to determine the source of the interrupt. ▪ Resolve the interrupts by contacting the respective engineer for the domain.	Section 4.10
13	▪ If dsp irq is observed frequently with a period less than 2 sec, check and resolve app wake-lock issue.	Section 4.8
14	▪ Ensure that the rockbottom use case is optimized, or subtract the delta between the QTI rockbottom use case and the customer device.	Section 3.1
15	▪ Collect and compare the GPIO and PMIC LDO configurations with QTI reference.	Section 4.1.7
16	▪ GPIO and LDO configuration are dependent on the hardware design. ▪ Captured GPIO and LDO data must be reviewed by QTI and OEM hardware teams to turn off unnecessary components. ▪ If the issue still exists, file a case with QTI for further debugging help; include all the debug information, for example, a clock dump, ftrace, waveform, PowerTop, Top, and power rail breakdown. ▪ For this information, file a case.	Section 1.3

3.5 Video playback

Step no.	Step	Reference
1	▪ Ensure that baseline rockbottom, static image, and MP3 use cases are optimized.	Section 3.1 Section 3.4
2	▪ If display panel power can be quantified, ask QTI for comparable panel power before proceeding to Step 3.	
3	▪ To make a proper comparison with the QTI reference data, the hardware configuration also must be comparable. ▪ Check for any custom components on the device; account for the current delta from each component as the known delta: ▫ Audio playback in Tunnel mode vs Nontunnel mode ▫ Awesome player or Nuplayer ▫ Sensors and other third-party components ▫ Different DDR size compared to the QTI reference ▫ Any additional postprocessing hardware ▪ For this information, file a case.	Section 1.3
4	▪ Checking Tunnel mode and player type.	Section 4.7
5	▪ If using Nontunnel mode or Nuplayer, which are not part of the QTI standard, ask QTI for the power impact for this variation.	

Step no.	Step	Reference
6	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ If using Tunnel mode audio playback, compare it with the measured number with the QTI reference data to determine if power consumption is higher, taking into consideration the known delta as quantified in Step 3. ▪ If using Nontunnel mode, request the reference power number through a case.	Release Notes
7	▪ Observe the power waveform pattern and identify if the baseline current is high, there are frequent wakeups, or both are observed. ▫ If the baseline current is high, go to Step 8. ▫ If frequent wakeups are observed, go to Step 16.	
8	▪ If a device has breakdown capability, capture a full power rail breakdown and compare it with the QTI reference to identify if any rail is consuming higher current. ▪ If the breakdown cannot be captured, go to Step 6.	Specific chipset dashboard goals and use case clock plan document
9	▪ Identify which power rail is higher than the QTI power data and determine the subsystems using it; look only for these components in Steps 10 through 14.	Specific chipset power overview document
10	▪ Capture SurfaceFlinger and compare it with the SurfaceFlinger reference logs to check for total number of layers, composition type, and number of layers being updated.	Section 4.6
11	▪ Capture a full clock dump to compare it with the QTI reference clock plan for video playback.	Specific chipset dashboard goals and use case clock plan document
12	▪ If the CPU clock is high, capture ftrace, PowerTop and Top to understand CPU residency and CPU load information.	Section 4.9
13	▪ Determine which process has high CPU usage and look for any unexpected processes/interrupts compared to the reference logs. ▪ Resolve the abnormal process/thread by contacting the respective area engineer.	Section 4.9
14	▪ If the BIMC clock is high, compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the reference log. ▪ If reference logs are needed, file a case. ▪ After a particular client which voted for more bandwidth is identified, contact the respective subsystem engineer to resolve this.	Section 4.13
15	▪ If the issue still exists, file a case with QTI for further debugging help; include all the debug information, for example, clock dump, ftrace, waveform, PowerTop, Top, and power rail breakdown. ▪ For this information, file a case.	Section 1.3
16	▪ For frequent wakeups, check the period of the wakeup. ▪ Capture PowerTop, ftrace and look for periodic wakeups	Section 4.10
17	▪ Check PowerTop for any unexpected interrupts compared to a reference log. ▪ Use pytime chart to analyze ftrace to determine the source of the interrupt. ▪ Resolve the interrupts by contacting the respective engineer for the domain.	Section 4.10

Step no.	Step	Reference
18	▪ Collect and compare the GPIO, PMIC LDO configurations with the QTI reference.	Section 4.1.7
19	▪ GPIO and LDO configuration are dependent on the hardware design. ▪ Captured GPIO and LDO data must be reviewed by QTI and OEM hardware teams to turn off unnecessary components. ▪ If issues still exist, go to Step 15.	

3.6 Talk/Voice call

Step no.	Step	Reference

1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta:
	  ▫ Sensors
	  ▫ Different DDR size
	  ▫ Other third-party components
	▪ For this information, file a case.
		Section 1.3

2	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 3.	Release Notes
3	▪ Capture rail level power breakdown data and determine which subsystem or module is consuming higher current than expected.	Specific chipset application note for power breakdown reference

4	▪ Check the following points for higher current consumption by a particular module or rail:
	  ▫ APSS not in power collapse – Go to Section 4.1.3.
	  ▫ Shared clocks/resources running at a higher level – Go to Step 5.
	  ▫ Modem software Hexagon™ processor taking higher current – Go to Step 6.
	  ▫ Modem RF or PA consuming higher current – Go to Step 7.	

5	▪ Check clock dumps and RPM NPA dumps to determine if any clocks or shared resources are running at a higher level and which subsystems are voting for the same.	Section 4.3.1
6	▪ Check modem NPA resource logs to determine the modem-specific resources running at a higher level and which clients are voting for those resources.	Section 4.3.3
7	▪ Check if proper calibration is done for that particular RAT and band configuration. ▪ Check if APT/ET enablement and calibration is done correctly. ▪ If using a third-party PA, the current consumption must be quantified for the same.	Section 4.3.2
8	▪ If power consumption is still higher than expected after following Steps 1 through 7, file a case with QTI for further debugging help.	Section 1.3

3.7 Data call

Step no.	Step	Reference
1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta: ▫ Sensors ▫ Different DDR size ▫ Other third-party components ▪ For this information, file a case.	Section 1.3
2	▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1
	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 3.	Release Notes
3	▪ Capture rail level power breakdown data and determine which subsystem or module is consuming higher current than expected.	Specific chipset application note for power breakdown reference
4	▪ Check the following points for higher current consumption by a particular module or rail: ▫ Shared clocks/resources running at a higher level – Go to Step 5. ▫ APSS consuming higher current – Go to Step 6. ▫ Modem software Hexagon processor taking higher current – Go to Step 7. ▫ Modem RF or PA consuming higher current – Go to Step 8.	
5	▪ Check clock dumps and RPM NPA dumps to determine if any clocks or shared resources are running at a higher level and which subsystems are voting for the same.	Section 4.3.1
6	▪ Check modem NPA resource logs to determine the modem-specific resources running at a higher level and which clients are voting for those resources.	Section 4.3.3
7	▪ Check APSS-specific logs, such as PowerTop, Top and ftrace logs, to determine what is causing the higher current consumption from the APSS.	Section 4.4.1
8	▪ Check if proper calibration is done for that particular RAT and band configuration. ▪ Check if APT/ET enablement and calibration is done correctly. ▪ If using a third-party PA, the current consumption must be quantified for the same.	Section 4.3.2
9	▪ If power consumption is still higher than expected after following Steps 1 through 8, file a case with QTI for further debugging help.	Section 1.3

3.8 Game

Step no.	Step	Reference

1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta:
	  ▫ LCD resolution
	  ▫ Smart/dumb panel
	▪ If panel power is quantified, go to Step 3.
	▪ If panel power is not quantified, the data cannot be compared with the QTI reference data. Compare all of the debug information with MTP data. Go to Step 5.	

2	▪ Ensure that the junction temperature is 35ºC. ▪ See Section 4.15. ▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	Section 4.15 80-N6837-1
	▪ Compare it with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 1.	Release Notes
3	▪ Capture the full breakdown and compare it with the 3D gaming breakdown. If power rail breakdown cannot be captured, go to Step 6.	Specific chipset dashboard goals and use case clock plan document
4	▪ Identify which power rail is higher than the QTI power data and determine what subsystems are using it.	
5	▪ Capture all of clocks and regulators to compare them with QTI data.	Section 5.4 Section 5.9
6	▪ Compare the captured clock data with the 3D gaming clock plan to determine which clock is high. ▪ Typically, the game power issues are from the CPU/GPU/BIMC clock, interrupts. Follow Steps 6-1-2, 6-2, and 6-3.	Specific chipset dashboard goals and use case clock plan document
6-1	▪ CPU clock is high	Section 4.9

6-1-1	▪ Governor, cpufreq, and scheduler parameters
	  ▫ Compare the governor, cpufreq, and scheduler parameters with QTI default settings.
	  ▫ If there are different parameters:
	    – Remeasure after changing the parameters to the QTI default.
	    – If there is no improvement, go to the next step in this Step 6-1-1.
	  ▫ If there are no different parameters, capture CPU-related information by ftrace, PowerTop, and Top.
		Section 4.14

6-1-2	▪ High CPU usage process; see Section 4.9.	Section 4.9
6-1-3	▪ Unexpected interrupt or interrupt count, see Section 4.10.	Section 4.10

6-2	▪ BIMC clock is high.
	  ▫ Compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the MTP.
	  ▫ See Section 4.13.
	  ▫ File a case with QTI if the MTP data needs to be compared.
	  ▫ If the client that voted more bandwidth compared to MTP can be determined, contact the subsystem engineer or file a case.
		Section 4.13

Step no.	Step	Reference
6-3	▪ GPU clock is high; see Section 4.11.	Section 4.11
6-4	▪ If other clocks are high, file a case with the debugging information clock dump, ftrace, waveform, PowerTop, Top, systrace data, and power rail breakdown data.	Section 1.3
7	▪ GPIO and LDO configuration are dependent on the hardware design. ▪ Captured GPIO and LDO must be reviewed by data by QTI and OEM hardware teams to turn off unnecessary components.	Section 4.1.7
8	▪ If power consumption is still higher than the QTI reference data, go to Step 6-4.	

3.9 Browser

Step no.	Step	Reference
1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta: ▫ LCD resolution ▫ Smart/dumb panel ▪ If panel power is quantified, go to Step 3. ▪ If panel power is not quantified, the data cannot be compared with the QTI reference data. Only compare all of debug information with MTP data. Go to Step 5.	
2	▪ Verify that the junction temperature is 35ºC. ▪ See Section 4.15. ▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1 Section 4.15
	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 1.	Release Notes
3	▪ Capture the full breakdown and compare it with the browser breakdown. If power rail breakdown cannot be captured, go to Step 5.	Specific chipset dashboard goals and use case clock plan document.
4	▪ Identify which power rail is higher than the QTI power data and determine what subsystems are using it.	Specific chipset power overview document
5	▪ Quantify the current consumed by Wi-Fi. ▪ Verify Wi-Fi power. ▪ Check with module engineer or solution provider to determine if Wi-Fi current is expected.	Section 4.16
6	▪ Compare the waveform with reference data.	Section 4.12
7	▪ Capture all clocks and regulators to compare with QTI data.	Section 5.4 Section 5.9

Step no.	Step	Reference
8	▪ Compare the captured clock data with the browser clock plan to determine which clock is high. ▪ Typically, the browser current issues are high from CPU/GPU/BIMC clock, interrupts, and Wi-Fi power. Follow Steps 8-1 through 10 in the respective problem area.	Specific chipset dashboard goals and use case clock plan document
8-1	▪ CPU clock is high.	

8-1-1	▪ Governor, cpufreq, and scheduler parameters
	  ▫ Compare the governor, cpufreq, and scheduler parameters with QTI default settings.
	  ▫ See Section 4.14.
	  ▫ If there are different parameters:
	    – Remeasure after changing the parameters as QTI default.
	    – If there is no improvement, go to Step 8-1-2.
	  ▫ If there are no different parameters, capture CPU-related information by ftrace, PowerTop, and Top.
		Section 4.14

8-1-2	▪ High CPU usage process; see Section 4.9.	Section 4.9
8-1-3	▪ Unexpected interrupt or interrupt count; see Section 4.10.	Section 4.10

8-2	▪ BIMC clock is high.
	  ▫ Compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the MTP.
	  ▫ See Section 4.13.
	  ▫ File a case if the MTP data must be compared.
	  ▫ If the client that voted for more bandwidth compared to the MTP can be determined, discuss it with the module engineer or file a case.
		Section 4.13

8-3	▪ GPU clock is high. ▫ See Section 4.11.	Section 4.11
8-4	▪ Other clock is high. ▫ File a case with the debugging information clock dump, ftrace, waveform, PowerTop, Top, systrace data, and power rail breakdown data.	Section 1.3
9	▪ GPIO and LDO configuration are dependent on the hardware design. ▪ Captured GPIO and LDO data must be reviewed by QTI and OEM hardware teams to turn off unnecessary components.	Section 4.1.7
10	▪ If power consumption is still higher than the QTI reference data, go to Step 8-4.	

3.10 Web streaming

Step no.	Step	Reference
1	▪ Quantify the current consumed by the following used in the device; this must be accounted for as the known delta: ▫ LCD resolution ▫ Smart/dumb panel ▪ If panel power is quantified, go to Step 3. ▪ If panel power is not quantified, the data cannot be compared with the QTI reference data. Only compare all debug information with MTP data. Go to Step 5.	
2	▪ Verify that the junction temperature is 35ºC. ▪ Check the temperature before running the use case. ▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1 Section 4.15
	▪ Compare with the QTI reference power data and determine if the power consumption is higher, taking into consideration the known delta as quantified in Step 1.	Release Notes
3	▪ Capture the full breakdown and compare it with the video streaming breakdown. If power rail breakdown cannot be captured, go to Step 5.	Specific chipset dashboard goals and use case clock plan document
4	▪ Identify which power rail is higher than the QTI power data and determine what subsystems are using it.	Specific chipset power overview document
5	▪ Quantify the current consumed by Wi-Fi. ▪ Check with the module engineer or solution provider if Wi-Fi current is expected.	Section 4.16
6	▪ Check Tunnel mode and player type.	Section 4.7
7	▪ Capture all of the clocks and regulators to compare with QTI data.	Section 5.4 Section 5.9
8	▪ Compare the captured clock data with the video streaming clock plan to determine which clock is high. ▪ Typically, the video streaming current issues are high from CPU/GPU/BIMC clock, interrupts, fps, and Wi-Fi power. Follow Steps 8-1 to 8-3 in the respective problem area.	Specific chipset dashboard goals and use case clock plan document
8-1	▪ CPU clock is high.	
8-1-1	▪ Governor, cpufreq, and scheduler parameters: ▫ Compare the governor, cpufreq, and scheduler parameters with QTI default settings. ▫ If there are different parameters: – Remeasure after changing the parameters as QTI default. – If there is no improvement, go to Step 8-1-2. ▫ If there are no different parameters, capture CPU-related information by ftrace, powerTop, and Top.	Section 4.14
8-1-2	▪ High CPU usage process	Section 4.9
8-1-3	▪ Unexpected interrupt or interrupt count.	Section 4.10

Step no.	Step	Reference
8-2	▪ BIMC clock is high. ▫ Compare the bandwidth voting under /d/msm-bus-dbg/client-data/ with the MTP. ▫ File a case if the MTP data needs to be compared. ▫ If the client that voted for more bandwidth compared to the MTP can be determined, discuss it with the module engineer or file a case.	Section 4.13
8-3	▪ Other clock is high ▫ File a case with the debugging information clock dump, ftrace, waveform, PowerTop, Top, systrace data, power rail breakdown data.	Section 1.3
8-4	▪ Must check fps. ▪ File a case for debug in detail if there is a different fps.	Section 1.3 Section 4.11
9	▪ If power consumption is still higher than the QTI reference data, go to Step 8-3.	

3.11 Camera preview

Step no.	Step	Reference
1	▪ Prepare the required setup and tools as mentioned in the power debug setup section. ▪ Verify that the junction temperature is 35ºC. ▪ See Section 4.15. ▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1 Section 4.15
2	▪ If the OEM does not have a breakdown of power numbers from the customizations, go to Step 4 for general debugging. ▪ If the OEM has a breakdown of power numbers due to various components, quantify the current consumed by the following; this must be accounted for as the known delta: ▫ OIS ▫ Dual camera ▫ Hardware-based face detection ▫ Any other special hardware, such as a separate Image Signal Processor (ISP) for IQ processing ▫ Extra software library included in the camera pipeline, such as noise reduction, video stabilization, and scene detect, other than QTI IQ libraries ▪ Current delta is usually calculated by enabling and disabling the feature and measuring the difference in current.	

Step no.	Step	Reference
3	▪ When taking power numbers for comparison, use the mainline Snapdragon™ camera application for all power measurements. ▪ Find the Snapdragon camera application in the code aurora forum (CAF) repository at https://www.codeaurora.org/cgit/quic/la/platform/packages/apps/SnapdragonCamera/tree/.	
	▪ Obtain the reference power numbers for the use case from the release notes. ▪ If the customer device measurement data minus the known delta from Step 2 is greater than the QTI reference data, go to Step 4.	Release Notes
	▪ If the reference power numbers are not available, file a case with QTI requesting the power numbers for a similar configuration on the QTI MTP. See Section 1.3 for details required to file a case. ▪ If customer numbers are higher than QTI numbers, go to Step 4.	Section 1.3
4	▪ Camera sensor configuration analysis: ▫ Collect details about sensor, such as configured sensor resolution or sensor fps configuration.	Sec 4.17.3
	▫ Check if the sensor is configured to the minimal resolution and fps as possible to meet OEM requirements; for example, if preview is 1080p and snapshot size is 8 MP for 30 fps, confirm that the sensor is configured to output resolution and fps closer to these settings. ▫ After the sensor configuration is correct, go to Step 5.	
5	▪ Camera subsystem analysis: ▫ Collect the clock dump and check clocks of the camera subsystem (VFE0, VFE1, CPP, and BIMC).	Section 4.17.1
	▫ If the VFE clock is running at TURBO, check Section 4.18.1 to determine if you can enable dual ISP hardware to process the sensor output. ▫ If the VFE and CPP clocks are aligned with the clock plan, the camera hardware is running at the correct frequency; go to Step 6.	See the specific chipset dashboard goals and use case clock plan document. Section 4.18.1
6	▪ CPU usage analysis: ▫ Collect CPU usage logs (ftrace, systrace, PowerTop) by following the steps in Section 5.8 and Section 5.9. See Section 4.9 to identify the Top process using the CPU. ▫ Obtain details about imaging libraries, such as third-party 3A, any noise reduction algorithms used in OEM camera pipeline for the use case.	Section 5.8 Section 5.9 Section 4.9
6-1	▪ If the Top CPU usage is from QTI 3A (thread names – AECAWB, AF, AFD, ASD) or other QTI modules, file a case with all debugging information (clockdump, ftrace dump, and Top logs). Go to Section 5.2 for other subsystem analysis. ▪ If Top usage is from the third-party algorithms, discuss with the third-party vendor for CPU optimization.	
6-2	▪ BIMC usage: ▫ Follow the steps in Section 4.13 to log BIMC usage during the use case. Compare the BIMC usage with the QTI MTP. ▫ File a case if the MTP data needs to be compared. ▫ If high BIMC clock and bandwidth vote is observed during the camera use case, go to Section 5.3 for suggestions to reduce bus bandwidth. Otherwise, go to Step 7 to debug GPU usage.	

Step no.	Step	Reference
6-3	▪ If display resolution is higher than 1080p, verify that the preview size requested by the camera can be maintained at 1080p and allow display hardware to upscale to physical display resolution. ▪ Reducing preview size reduces the bus bandwidth requested by CPP and MDP. ▪ If display resolution is not high, go to Step 7 to debug GPU usage.	Section 4.18.3
7	▪ GPU usage analysis: ▫ Monitor GPU usage during the use case using the procedure mentioned in Section 4.11.1. If GPU usage is high, go to Step 7.1. Otherwise, go to Step 8.	Section 4.11.1

7.1	▪ Check if the camera application is using GPU to render preview frames; see Section 4.17.6.
	  ▫ If GPU is used, follow the suggestions in Section 4.18.2 to reduce GPU usage.
	  ▫ If TextureView is no longer used and GPU usage is still high, go to Step 7.2.
		Section 4.17.6
		Section 4.18.2

7.2	▪ Check if the camera use case triggers GPU composition from SurfaceFlinger; see Section 4.17.7.
	  ▫ If GPU composition is triggered, see Section 4.17.7.1 to identify the reason for GPU composition.
	  ▫ If GPU composition occurs because of more layers, contact the application developer to reduce the number of layers.
	  ▫ If the number of layers are not high to trigger Mixed mode composition, file a case with QTI for further analysis of the problem.
	  ▫ Go to Step 8.
		Section 4.17.7

8	▪ Other power optimizations
	  ▫ If the above optimization suggestions do not yield improvements, see Sections 4.18.5, 4.18.7, and 4.18.8:
	    – Follow the suggestions.
	    – See the impact/side effect suggestions for information about possible Image Quality (IQ) and/or performance impact with the optimization.
		Section 4.18.5
		Section 4.18.7
		Section 4.18.8

3.12 Video recording

Step no.	Step	Reference
1	▪ As video recording uses a similar pipeline as camera, optimize the camera preview current by following the steps in Section 3.11. ▪ If power usage is still higher, go to Step 2 for optimizations specific to video recording.	Section 3.11

2	▪ Camera sensor configuration analysis:
	  ▫ Find the sensor Operating mode and resolution, i.e., mode at which the sensor hardware is initialized and configured, and ISP configuration during the use case. See Sections 4.17.3 and 4.17.4 for the procedure to collect this information.
	  ▫ Check if the sensor output is configured to be close to the OEM preview and live shot resolution. For example, if the live shot requirement is 8 MP, preview is 1080p, and sensor is 16 MP, the sensor should be configured to output only 8 MP. The sensor should not be configured to full resolution mode of 16 MP.
	  ▫ If smaller sensor output size is not feasible, or if power improvement is not enough, see Section 4.18.1 to enable dual ISP configuration, if possible.
	  ▫ Go to Step 3.
		Section 4.17.3
		Section 4.17.4

3	▪ Video encoder configuration:
	  ▫ Collect the clock dump and check clocks of the camera subsystem (VFE0, VFE1, CPP, and BIMC) and video encoder (venus0_vcodec0_clk)
	  ▫ If this is UHD recording and the video encoder clocks are high, consider enabling the Lower Power mode of the video encoder. See Section 4.19.1 for details.
	  ▫ Go to Step 4.
		Section 4.19.1

4	▪ CPU usage analysis:
	  ▫ Check if there is any additional IQ processing library used in camera pipeline during recording.
	    – Quantify delta due to additional IQ processing library. Check if extra image processing can be reduced only for video recording or 4K cases to save power.	

5	▪ GPU usage analysis:
	  ▫ Collect GPU usage logs and apply all optimization suggestions mentioned in the GPU usage analysis section of Camera Preview.
	  ▫ If GPU usage is optimized fully, go to Step 6.
		Step 7 in Section 3.11

6	▪ Other power optimizations:
	  ▫ If the above optimization suggestions did not yield improvements, follow the suggestion below. See also the Impact/Side Effect suggestions to know about possible IQ and/or performance impact with the optimization.
	    – Temporal noise reduction is an IQ feature triggered during recording. Consider disabling temporal noise reduction by following the steps in Section 4.18.4.	

4 Dashboard use case debugging details

4.1 Rockbottom

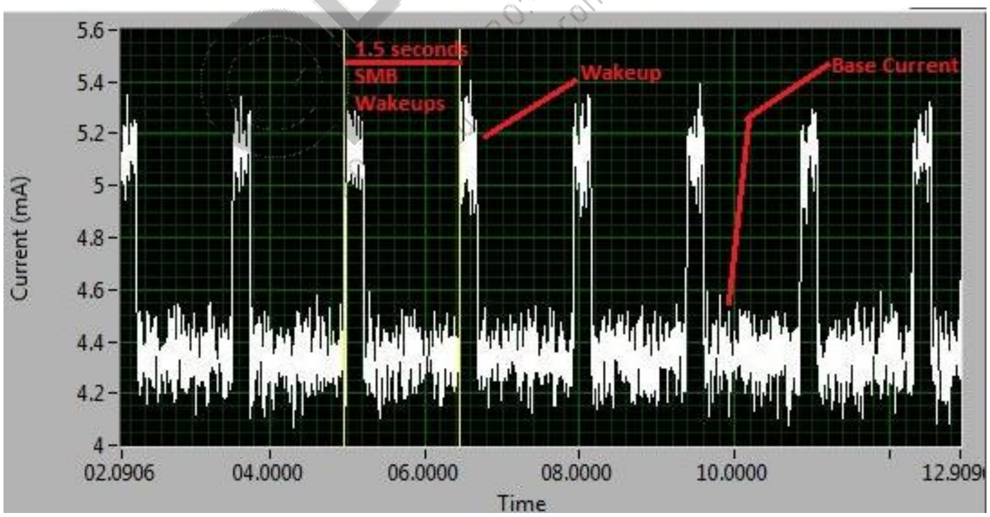
4.1.1 Analyze rockbottom waveform

Waveform analysis is critical for power debugging and gives information on the nature of the issue and correct direction of debug.

Rockbottom waveform can be divided into the following parts:

- Base current
- Wakeups, for example, fuel gauge wakeups and PMIC watchdog wakeups

The following shows a snapshot of the rockbottom use case waveform taken on an MSM8994 chipset.



Rockbottom waveform captured on the device under test can be compared with the QTI reference waveform to determine whether the base current is higher and/or there are occurrences of unexpected wakeups.

4.1.2 Verify XO shutdown and VDD minimization

To verify if the system is entering XO shutdown and VDD minimization, use any of the following methods:

- T32 breakpoints on the RPM
  - For XO shutdown − xo_shutdown_enter() and then mpm_sw_done()
  - For VDD minimization − vdd_min_enter() and then mpm_sw_done()
- adb shell commands for XO shutdown and VDD minimization count
  - Type the following commands to obtain the RPM statistics:

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

    The following is the output of the above commands; the count represents the number of occurrences of XO shutdown and VDD minimization.

```
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
```

  - Alternatively, check the following variables from RPM RAM dumps to determine count of XO shutdown and VDD minimization.
    - sleep_stats[0] – XO shutdown count (number of times the system went to XO shutdown)
    - sleep_stats[1] – VDD minimization count

NOTE: If the system enters the VDD minimization state, only the VDD minimization count is incremented. VDD minimization is the lowest power state, and it includes XO shutdown. XO shutdown count is only incremented when the system fails to enter VDD minimization and enters XO shutdown.

4.1.3 Check subsystem (APSS/MPSS/LPASS) power collapse from RPM

1. Check the RPM external logs.
   - For APSS power collapse, look for the following messages in the RPM external log:

```
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")
```

   - For MPSS power collapse, look for the following messages in the RPM external log:

```
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")
```

   - For LPASS power collapse, look for following messages in the RPM external log:

```
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")
```

   As APSS/MSS/LPASS are expected to power collapse as soon as there are no tasks to block Low Power modes, the above messages might be overwritten and are not visible in the RPM logs. For improved verification, check the data structures available in RPM.

2. Check the RPM data structure.
   - Power collapse status of subsystems can be confirmed by checking the RPM data structure on RPM. After the display is off and given enough time to go to sleep, break the T32 execution in RPM randomly to monitor the following variables:
     - Check rpm.ees[0].subsystem_status, 0→APSS information
     - Check rpm.ees[1].subsystem_status, 1→Modem information
     - Check rpm.ees[2].subsystem_status, 2→LPASS information

     Alternately, obtain a crash dump using a system reset and analyze the RPM dumps using the T32 simulator.

     - If status is SPM_SLEEPING, the subsystem can be considered in the lowest power state.
     - If status is SPM_AWAKE, the subsystem is awake and not in lowest power state.
   - Continue debugging by taking subsystem specific logs to determine why the subsystem is not voting for sleep.

3. Ensure all masters are power collapsed.
   a. Obtain the master (EE) status from the ee-status.txt file generated by the Hansei parser tool from the RPM dump.

```
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 Determine subsystem votes for major resources using RPM logs

The following are the major resources that can be relinquished or voted for low power to attain XO shutdown and VDD minimization:

- VDD CX − Digital power rail
- VDD MX − Memory power rail
- CXO − System clock (XO)

1. Check CXO votes by different subsystems using the RPM NPA log.

   When there are no CXO client votes, the XO can be shutdown to save power. The system can choose to enter VDD minimization by keeping the digital (CX) and memory (MX) rails at the retention voltage to save more power.

2. Check the RPM NPA log for the following messages to determine which subsystem requires CXO and is preventing XO shutdown and VDD minimization:
   - Check the active state of the /xo/cxo node from the RPM NPA logs; for example, from the following NPA log snippet, MPSS and APSS are shown as the clients requesting the CXO resource:

```
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
```

This indicates that the device sleep is blocked by the MPSS and APSS. Continue debugging on the subsystem level; see the details in Section 4.1.5.

4.1.5 Check power collapse of individual subsystems

Individual subsystems prevent system power collapse due to one of the following reasons:

- Active applications or processes on the subsystem requiring a system to be running
- Applications or processes not releasing resources gracefully for a successful suspend; the following are examples for resources held by a system:
  - Clocks
  - Voltage rails
- Frequent interrupts preventing system power collapse

4.1.5.1 Determine why the APSS is not voting for power collapse

Wake locks

Wake locks are one of the major reasons that the APSS does not vote for power collapse. To check which wake lock is holding power collapse:

1. Connect the USB to the system.
2. In the adb shell, type the following command to obtain the wake-lock logs:

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

3. Remove the USB and use the power key to suspend the system immediately.
4. After 60 sec, type the following command to reconnect the USB and pull wakelocks.txt:

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

5. Open wakelocks.txt and check the active_since field. If any of the wake locks were active for more than 60 sec, this suggests that wake lock is holding power collapse.

NOTE: The unit of time in wakeup_sources log is milliseconds.

Check clocks preventing XO shutdown and VDD minimization

1. To check the clocks that are preventing XO shutdown/VDD minimization, type the following command to enable the clock debug suspend:

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

   After enabling this flag, the clocks that are enabled are shown when the system is going into suspend mode in the dmesg logs.

2. Suspend and resume the system by pressing the power button three to four times, and type the following command to take a dmesg log:

```
adb shell dmesg
```

   Some of the clocks are always expected to be shown as enabled in this log, but if any clocks other than usual major system clocks are shown as enabled, it can be a reason for preventing power collapse. The following are examples of clocks that should not be seen in this log:

   - Any peripheral clocks
   - Display-related clocks (MDSS)
   - Any multimedia subsystem-related clocks, etc.

   Comparison of enabled clocks – See the following figure:

- Observe that in the clock log on the left side of the figure, there are 27 clocks shown as enabled, but the system goes to XO shutdown/VDD minimization without any issues. These clocks/PLLs are turned off later by RPM and are part of major shared resources like BIMC, system buses, and subsystem PLLs.
- In the snippet on the right, there are 35 clocks enabled but a few have distinctive features.
  - CXO clock source is requested by the APSS.
  - MDP (display subsystem) clocks are enabled, suggesting that the driver is holding CXO from being power collapsed.

Check interrupts activity that can prevent XO Shutdown and VDD minimization

Higher frequency of interrupts can prevent the system from entering XO shutdown/VDD minimization.

1. Type the following command in an adb shell to check the interrupts that are firing more frequently:

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

2. Remove the USB immediately and suspend the system by pressing the power button.
3. Reconnect the USB and pull interrupt1.txt and interrut2.txt.
4. Compare the count of interrupts in both files.
5. If any of the interrupts has a substantially high difference between counts in interrupt1.txt and interrupt2.txt, contact the respective subsystem engineer.

4.1.5.2 Determine why the modem subsystem is not voting for power collapse

To determine why the modem subsystem is not voting for power collapse, check the MPSS NPA log. The NPA log has information about various NPA requests made by the MPSS. To check which subsystem is holding sleep, in the NPA logs search for the term npa_dump. This is the starting point for resource states for the system at the point of log collection.

Checking the active_state of resources can provide information about the state of the resource. Check the following main resources:

- CXO
- VDD_CX
- VDD_MX
- CPU_VDD (modem subsystem rail)

If any of the clients for these resources are holding the resource, review them; for example:

```
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)
```

In the above snippet, the resource /core/cpu/vdd is requested by the GPS client. This means GPS is holding CPU_VDD (modem subsystem rail) from going into power collapse.

To find the root cause of why GPS is not going to power collapse, continue analyzing from the GPS side.

CAUTION: Analyze NPA logs carefully as the resource states only give the instantaneous request state. NPA logs also contain the history of requests and can be checked when a particular request was made by a subsystem.

4.1.5.3 Determine why the LPASS is not voting for power collapse

To determine why the LPASS is not going to power collapse, check the LPASS NPA log. The NPA log has information about various NPA requests made by the MPSS. To check which subsystem is holding sleep, in the NPA logs search for the term npa_dump. This is the starting point for resource states for the system at the point of log collection.

Checking the active_state of resources can provide information about the state of the resource. Check the following main resources:

- CXO
- VDD_CX
- VDD_MX
- CPU_VDD (modem subsystem rail)

If any of the clients for these resources are holding the resource, review them.

4.1.6 Check unexpected wakeups

Wakeups are one reason that can cause the rockbottom to be higher. The following reasons can also cause wakeups:

- Subsystem wakeups due to sensor processing requests
- Scheduled wakeups (timer expiry)

- Subsystem wakeups due to interrupts from another subsystem, for example, smd interrupts from MPSS to APSS.

To identify the wakeup source, one of the methods includes monitoring the subsystem and digital rails. Most of the chipsets have a dedicated power rail for each subsystem. All subsystems are expected to be in sleep during VDD minimization, and the subsystem waking up can be easily identified.

Monitoring rails for external components and sensors can also be helpful in the case where the wakeups are external to the chipset.

Check subsystem-specific logs to determine if that particular subsystem is waking up from power collapse to do some scheduled activity or due to some interrupts.

4.1.6.1 Check modem wakeups

It is not expected that the MPSS wakes up during this use case because the rockbottom use case is performed keeping the device in Airplane mode. Capture modem uLogs to see if the modem is waking up from power collapse. MPSS uLogs contain several sleep-related logs that can be useful in determining if the modem subsystem is waking up and, if so, for what reason.

The following is a snippet of the sleep info log, part of the MPSS uLogs. See the timestamp when the modem exited the previous mode and entered the new mode.

```
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)
```

The reason for wakeup, which in the above snippet is Timer, means it was a scheduled wakeup. After the required processing is done by the modem, the sleep solver selects the appropriate mode for modem to enter depending on the latency available until the next scheduled wakeup, if any. Also see the mode chosen by the solver and the hard deadline for the next wakeup.

According to the wakeup reason, check if any timers are set or modification relating to the timer has been made in the code.

If the wakeup reason is “rude,” which is the nonscheduled wakeup, QXDM Pro logs can be helpful in determining what activity is occurring in the MPSS.

4.1.6.2 Check APSS wakeups

Kernel logs with the appropriate logging enabled can give an indication if the APSS is waking up from its sleep state and, if so, for what reason. The following debug mask can be enabled, which logs the interrupt information in the kernel logs.

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

Check prints in the kernel log to identify which interrupt is causing the APSS to wake up. The following snippet shows that the APSS is awakened by qpnp_kpdpwr_status, which is the power key press interrupt.

```
<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 Check leakage in sleep current

If the base current is higher than expected, even after the device successfully enters VDD minimization, it can be concluded that there are one or more leakage sources on the device contributing to the total current consumption.

Leakage can occur from multiple sources, such as the following:

- Leakage from SMPS and LDOs not used during sleep but are kept enabled; a PMIC dump is helpful in checking possible leakage from SMPS and LDOs.
- Leakage from GPIO pads if one or more GPIOs are not configured correctly in their lowest leakage settings; a GPIO dump can be helpful in determining the sleep configuration of the MSM™ GPIOs.
- Leakage from peripherals and external components not disabled or put in Low Power mode configuration

4.1.7.1 Analyze PMIC dumps

PMIC dumps provide information about the status of all LDOs, and SMPS and PMIC register settings for different modules powered by the PMIC.

The following is a snippet of a parsed PMIC dump taken on a MSM8994 device just before entering sleep. There is information about SMPS and LDO status, the voltage level, and Operating mode. Use this information to determine if any SMPS or LDOs not required during Sleep mode are still enabled but can be turned off to save leakage. Leakage can also be reduced for those SMPS and LDOs that are required during sleep by putting them in LPM mode instead of NPM mode.

```
=================================================================
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 Analyze GPIO dumps

GPIO dumps provide information about the configuration of all MSM GPIOs at the time of dump collection. The following is a snippet of a GPIO dump taken on an MSM8994 device just before entering sleep.

```
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
```

The red box is read as GPIO 11 is configured as Input - Pull Down with 2 mA drive and the value is Low. The correct GPIO sleep configuration depends on how the GPIO is used in the device design and if it is required during sleep. For the GPIOs that are used the same as the QTI reference design, a direct comparison can be made with GPIO dumps taken on a QTI reference device to determine any misconfiguration that needs to be changed to fix it.

For the GPIOs that are used differently as compared to the QTI reference design, the correct sleep configuration needs to be determined according to the usage of those GPIOs.

4.1.7.3 Rail level current consumption breakdown

Breakdown of current consumption for individual power rails can be very helpful in narrowing down which component of the system is consuming higher current consumption.

- As a first step, current consumption of all PMIC rails on PMIC input can be collected, for example, SMPS current, any LDOs that are not sourced by SMPS, etc. The current consumption of components directly powered by VPH (components not powered by PMIC) is also a first step of the breakdown.
- After determining which SMPS or component is consuming higher current consumption, collect further breakdowns of components connected to that particular SMPS to identify which component is consuming higher current.

Current consumption breakdowns are published in power application notes for that particular chipset for each of the dashboard use cases.

4.2 Standby

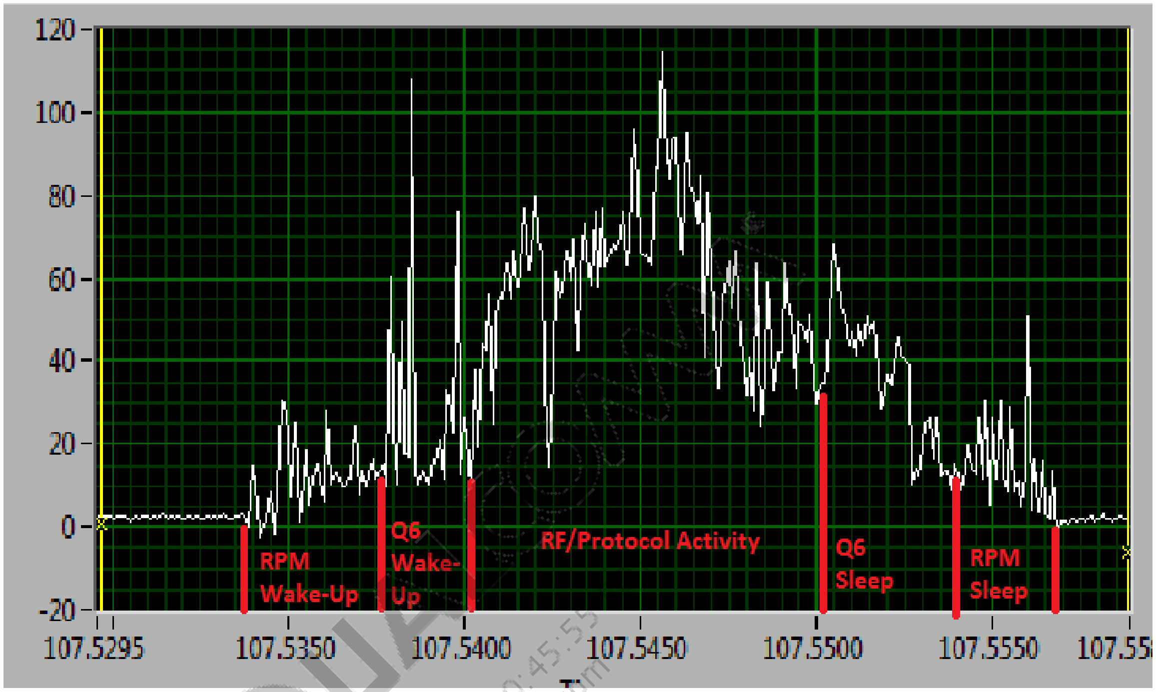
Total average standby current is highly dependent on the average rockbottom current consumption. If not already done, the first step for standby optimization is to optimize the rockbottom current consumption. Follow the steps in Section 4.1 to optimize rockbottom current consumption.

4.2.1 Higher paging awake penalty

Waveforms from current consumption capturing tools can be very helpful in analyzing the awake penalty incurred by paging wakeups.

- Compare waveforms of the paging awake cycle to determine which part of wakeup consumes more time or more current consumption.
  - System wakeup time
  - RF wakeup
  - RF/protocol processing
  - RF sleep
  - System sleep

The following figure shows correlating sections of the paging awake waveform to the stages of activities of different modules of the system involved during a DRX paging period.



- Check the following common areas if the paging timeline is higher than expected:
  - Callbox settings error, such as enablement of neighbor cells
  - Code modification in RPM or enablement of some of the modem logging
- Collect F3 logs, which are very helpful in debugging the awake period during paging, by setting appropriate configuration for log messages. For example, if debugging WCDMA awake penalty, use the following method to collect logs:

  a. Open QXDM Pro and press **F3** to open the Message View window.

  b. Right-click in the Message View window, and select **Config** from the menu. The Message View Configuration window opens.

  c. Under the Message Packets tab, select **Known Message (By Subsystem)** and expand its view.

  d. Select **Legacy**, **Mpower**, **UMTS**, and **Radio Frequency**.

  e. Under the Log Packets tab, select **WCDMA** log packets.

  f. Run the WCDMA standby use case and collect F3 logs.

  These logs are detailed logs for protocol/power/RF and have accurate timestamps that help determine the part of the system that is taking higher time and helps narrow down the issue.

- Check the following common areas if the paging awake amplitude/peak is higher:
  - Proper RF calibration for that RAT and band and APT/ET enablement, if applicable.
  - In case of high peak currents during paging awake cycle, a rail level breakdown is helpful to narrow down the rail causing the peak current consumption. The problem could be focused in that area.

4.3 Talk

Talk use case current consumption can be higher due to the following reasons:

- Processor clocks or system clocks are running at a higher frequency.
- APSS is not in power collapse (see Section 4.4.1 for debug information).
- PA/RF front end is consuming higher current consumption.
- Software and hardware design deltas, for example, using OEM proprietary voice algorithms, power amplifier ICs in the audio output path, special featured microphone hardware, etc.

4.3.1 Inspect clocks and shared resources

Clocks running at higher frequency than expected in the system can cause the system to consume more power. A clock dump can be taken and compared with the QTI reference device clock dump to ensure that clocks are at expected state. Consider the following when comparing clock dumps:

- Clock dumps can be taken by breaking the system at any point through RPM and running the clock dump script.
- Verify that no extra clocks are running other than the expected clocks.
- Verify that all cocks are running at similar frequencies as the QTI reference device. A complete comparison is more accurate and helpful, but the following are important clocks to compare:
  - BIMC (DDR) controller clocks
  - Modem clock
  - LPASS clock
  - Buses (system NOC, peripheral NOC, config NOC)
- Check the RPM NPA logs if the shared resources or clocks are running at a higher level than expected.

  The following snippet shows the voting of different subsystems for the shared resources:

```
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)
```

In the above snippet, votings by clients "MPSS, LPASS, APSS and ICB Driver" for the shared resource pcnoc clock is observed. Check the NPA_CLIENT_REQUIRED request for the actual votings for that particular resource.

Further debugging can be done for that particular subsystem whose votings are found higher than expected. For example, MPSS NPA resource logs can be checked for MPSS-specific clients and resource votings, if MPSS is found to be voting higher level of one or more shared resources.

4.3.2 Check PA/RF power consumption

If current consumption is still high, even after confirming correct votings for clocks, rail voltages and shared resources, check current consumption from the RF hardware and PA by measuring the respective rails.

If RF and PA is consuming higher current compared to the QTI reference data, proper RF calibration is required to be checked. Improper calibration can lead to the PA operating at higher gain stage even at 0 dBm Tx power. APT/ET enablement and proper calibration for the same is also required.

The PA current consumption varies with the usage of third-party PAs, and the delta for the same must be accounted for.

4.3.3 Check modem resource votings

Modem NPA resource logs, similar to the RPM NPA resource logs, provide information about modem-specific client voting for modem resources and shared resources.

Modem Hexagon clock, modem rail (VDD MSS), VDD_Core and VDD_Mem, BIMC clock, mcpm, are some of the many resources whose voting can be seen from the modem NPA logs.

Figure 4-1 is a snippet of modem NPA resource logs collected during a navigation use case, voting for system level resource (such as rail_MX) and modem internal resources (such as clk/cpu and cpu/busy resources). In this example, the GPS client can be seen voting for all the above-mentioned resources.

Check the voting to see if it is as expected. For example, verify gps_rx voting for the 288 MHz cpu clock by reviewing the QTI reference logs for the same use case.

```
: 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)
```

Figure 4-1 Modem NPA resource log snippet

4.4 Data

The data use case is one of the more complex use cases. It involves the modem obtaining the packet data and transferring those packets to the APSS for processing, and then being used by one or more user applications.

The rail level breakdown is very helpful in debugging the data use case because it can be quickly identified if the MPSS or APSS is consuming current higher than expected. Then subsystem level debugging can be done accordingly.

For RF/PA rails consuming higher current, see Section 4.3.2.

If rail level breakdowns are not readily available, resource voting by a different subsystem can be checked from the RPM NPA logs; see Section 4.3.1.

If modem rail current consumption is higher or it is found that MPSS is the subsystem that is voting for higher system resources, check the modem NPA resource logs to narrow down the modem client causing that vote. See Section 4.3.3 on checking modem NPA resource logs.

4.4.1 Check APSS

The APSS can be one of the major reasons of higher current consumption during data use cases because added features can impact power consumption. Use the following methods or logs to determine if the APSS is consuming higher current consumption:

- Analyze PowerTop logs and compare them against the QTI reference device as follows:
  - Observe the time in each C-State (low power states) in PowerTop logs.
  - Check time in each frequency and interrupt activity in PowerTop logs. If any unexpected interrupts are seen in the PowerTop logs, check the driver requesting the interrupts.
  - These tests provide a rough estimate of why the APSS consumes higher current.
  - These logs can be compared with the QTI reference device logs for a comparative analysis of which system components are supposed to be up.
- Analyze Top logs and compare them against the QTI reference device as follows:
  - Stop or kill any unexpected processes in the Top logs that are consuming CPU time.
  - Check for any extra running processes using Top data.
- Analyze ftrace logs as follows:
  - Analyze any process or API running in kernel resulting in extra CPU time.
  - Analyze the amount of time each core is in a particular frequency.
  - Check which process monopolizes the CPU if kworker thread activity is high.
  - Compare ftrace logs with the QTI reference device to determine which processes are more active on the APSS and if they can be stopped.

4.5 VDD minimization debugging for static image with smart panel

1. Use the following debug mask and collect a 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. In the kmsg, look for any clock holding CXO. For example, in the following snippet, UART is holding CXO and blocking XO shutdown and VDD minimization:

```
<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]
```

NOTE: There might be other clocks requesting cxo_clk_src_ao. These are the "Active Only" requests by these clients and should be ignored. Only look for cxo_clk_src.

3. Look for any hwirqs that occur, as shown in the following example:

```
<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. To check to what the irq corresponds, look for the specific irq in the interrupt list for the device by typing the following commands:

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

5. Check with corresponding teams to understand why this irq is triggered. For example, in the following figure, check with the display and graphics teams. If the particular irq is not expected and can safely be ignored, add it to the bypass list in the chipset power management .dtsi file.

```
 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. Add the irq with a 0xff to mask the irq to the .dtsi file, and ignore it. For MSM8994, the .dtsi file is found in /kernel/arch/arm64/boot/dts/qcom/msm8994-v2-pm.dtsi.

   For example, for masking irq 66, add <0xff 66> to the mpm interrupt map at the end of the list in the .dtsi file, as shown in the following:

```
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 debugging

SurfaceFlinger is the service used to compose the display frame based on multiple sources for rendering on the screen.

SurfaceFlinger creates a layer and a buffer for each source, and these layers are rendered on the screen.

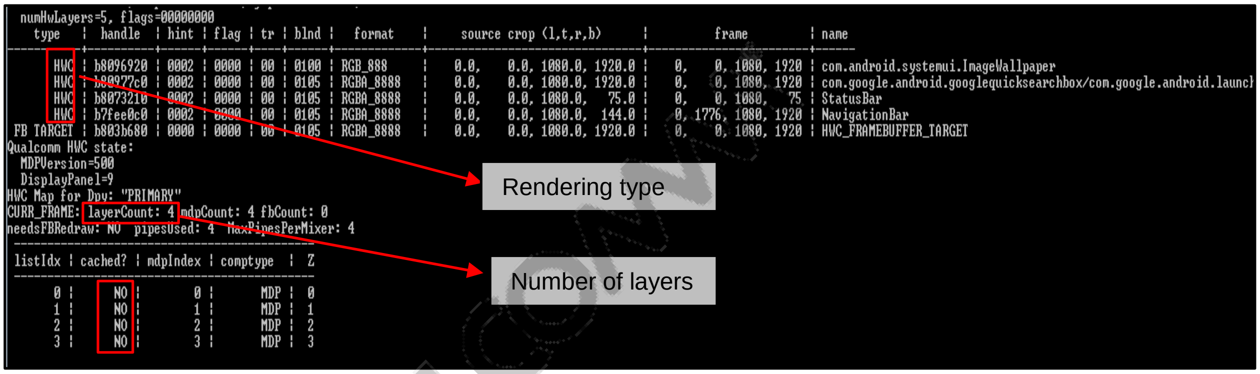
Common layers observed are as follows:

- Status bar
- Navigation bar
- Application

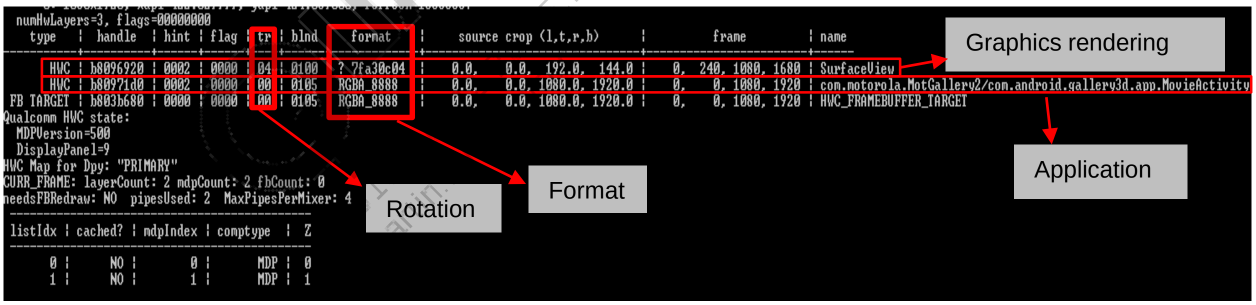
To collect SurfaceFlinger logs, type the following commands:

```
adb shell
dumpsys SurfaceFlinger
```

The following shows a static image SurfaceFlinger:



The following shows a full-screen video playback SurfaceFlinger:



In the above examples, the following are shown:

- Application
- Rendering type
- Composition types:
  - MDP composition
  - GPU composition causes higher power
- Total number of layers – Higher number of layers being updated implies higher current
- Rendering format – RGB, YUV, etc.
- Screen rotation
- Layers cached and noncached (a cached layer is not updated)

4.7 Tunnel mode and player type check

4.7.1 Tunnel mode check

1. Capture user-space (logcat) logs by typing the following adb command:

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

2. Play MP3 and capture logs for approximately 30 sec of MP3 playback duration.
3. In the logcat logs, look for the following message that ensures Tunnel mode playback is ongoing:

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

4. If the deep-buffer-playback message is found in the log, it indicates that nontunnel, i.e., normal, playback is ongoing.

4.7.2 Player type check

1. While running the use case, check the type of player (Nuplayer or Awesome Player) by typing the following commands:

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

For example:

```
shell@victara:/ $ dumpsys media.player
dumpsys media.player
 Client
  pid(21330), connId(35), status(0), looping(false)
 AwesomePlayer
  fd(22), flags(0x00006011)
  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)
```

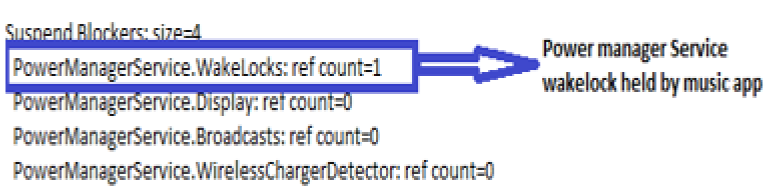
```
shell@victara:/ $ dumpsys media.player
dumpsys media.player
 Client
  pid(21330), connId(36), status(0), looping(false)
 NuPlayer
  numFramesTotal(127), numFramesDropped(0), percentageDropped(0.00)
 AudioOutput
  stream type(3), left - right volume(1.000000, 1.000000)
  msec per frame(0.022676), 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 Music application wake lock check and debugging

4.8.1 Wake lock analysis

1. See Section 5.10 for collecting wake locks and Section 4.1.5.1 for analyzing wakeup sources.
2. Ensure that the Power Manager Service is holding a wake lock using the following check:

   The following shows expected dumpsys power output:



3. If the above wake lock is not observed, hold a test wake lock and determine if power consumption reduces.

4.8.1.1 Hold test wake lock

To hold a test wake lock, type the following commands:

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

4.8.1.2 Remove test wake lock

1. To remove a test wake lock, type the following commands:

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

2. If the above experiment resolves the issue, modify the music app to hold a wake lock during MP3 playback with display turned off.

4.9 Analyze high CPU usage process

1. Capture the Top or ftrace data as mentioned in Section 5.8.
2. Determine which process is consuming the most CPU usage and look for any processes that should not be running.
   a. If there is an unexpected process or thread, discuss it with the module engineer for the process or thread.
   b. If there is no unexpected process or thread, do the following:
      i Determine the high CPU usage process or thread. The following shows that the CPU usage of process ID 472 is different between the customer device and the MTP device.

Top data 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 data 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
```

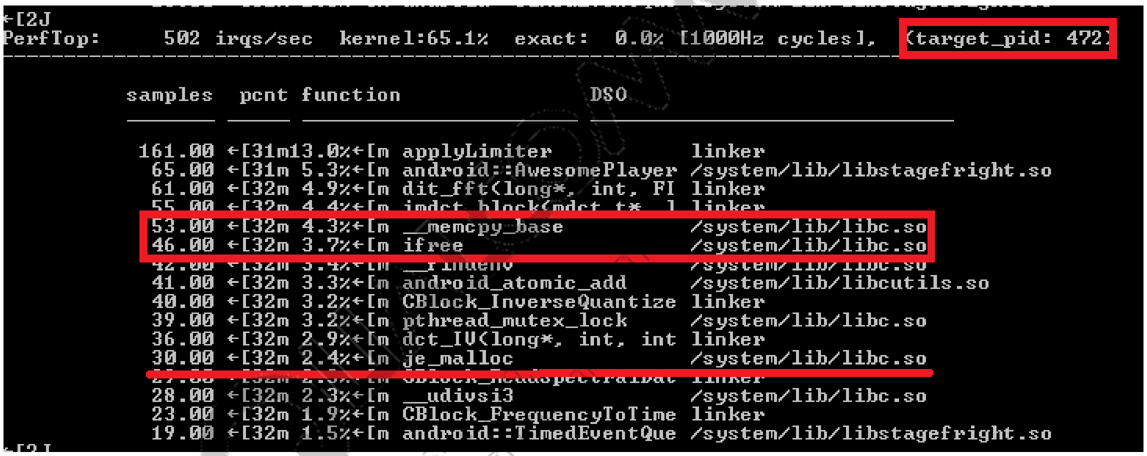
      ii Debug in detail with the PerfTop tool (see Section 2.1.2).

3. Connect the USB.
4. Type the following commands to capture PerfTop data.

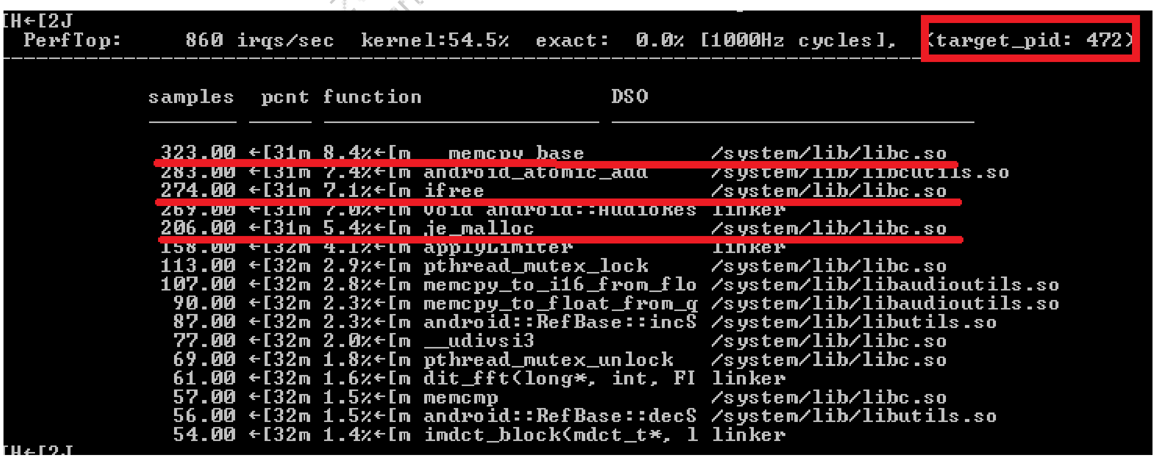
```
adb root
adb remount
adb shell /data/perf top -p 472
```

The following shows the PerfTop data A and B.

PerfTop data A



PerfTop data B



As the PerfTop data shows, the memory handling (memcpy, malloc, free) function in process ID 472 is more frequently called in PerfTop data A.

After analyzing this data, discuss it with the module engineer in charge of the media server or submit a case.

4.10 Interrupt analysis

1. Capture the PowerTop data.
2. If the data is similar to the following examples, capture an ftrace log to debug in detail for qcom,smd-rpm.

Data 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
```

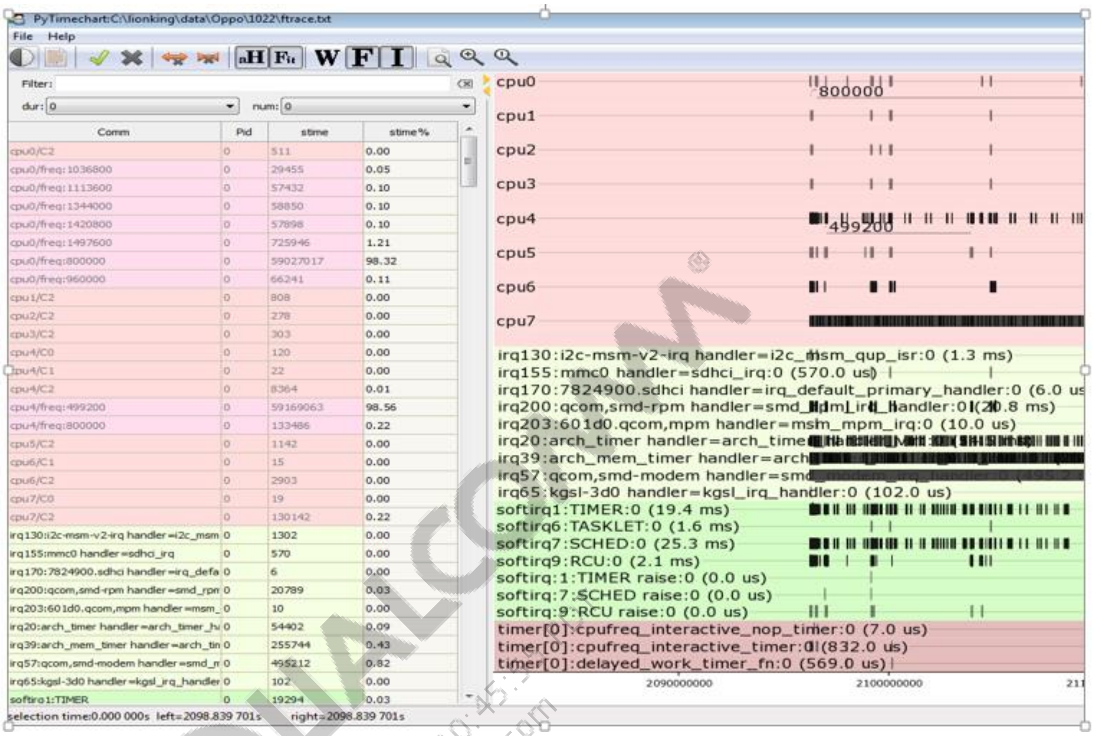
Data B

```
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
```

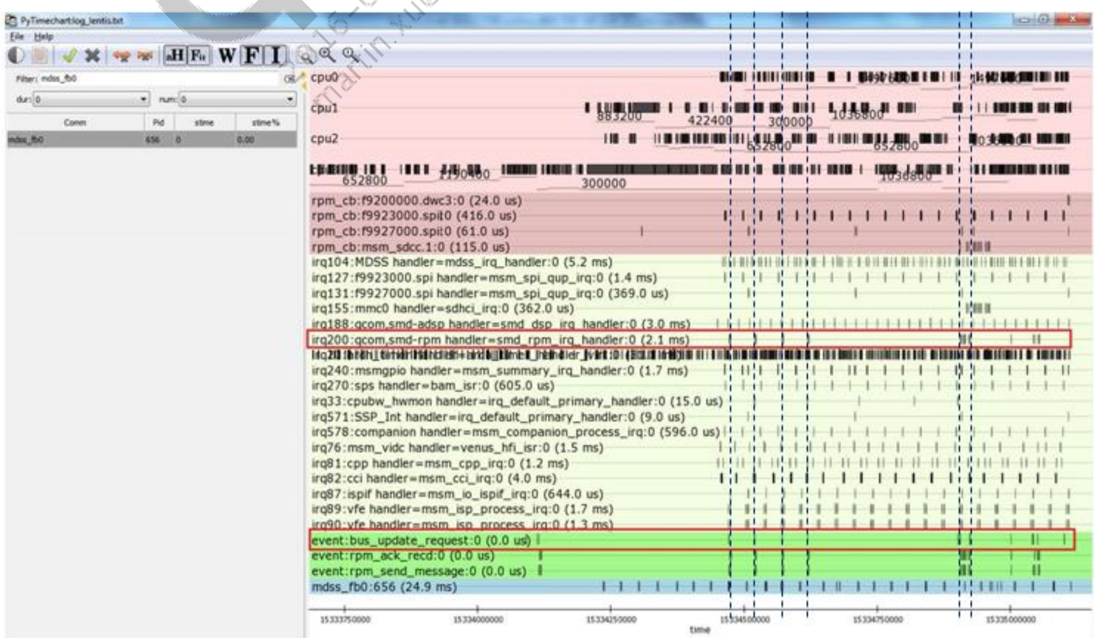
Use pytime chart to debug interrupt. See Chapter 2 for pytime chart installation.

3. Launch pytime chart.
4. Open the ftrace log by the menu in pytime chart.

The following screen appears.



5. Determine which module caused the qcom,smd-rpm interrupt by alignment as follows.



6. If you observe similar behavior as in ftrace above (bus_update_request), check the client for bus_update_request in the ftrace log file as follows:

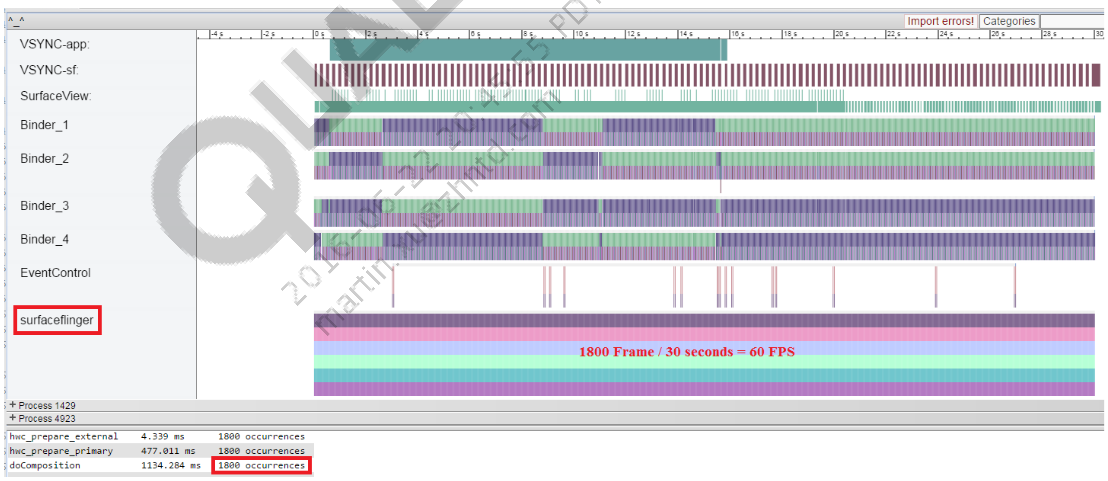
```
[001] ...1  1489.961370: bus_update_request: time:1489.952861564 name:mdss_mdp src:22 dest:512 ab:1830570776 ib:1830570776
[001] ...1  1489.961373: bus_update_request: time:1489.952861564 name:mdss_mdp src:23 dest:512 ab:1830570776 ib:1830570776
```

As a result of the analysis, MDSS_MDP caused qcom,smd-rpm interrupts. Discuss this with the display engineer to resolve this issue or submit a case.

4.11 Debug high GPU clock frequency

NOTE: For this procedure, a Chrome browser is required.

1. Verify the fps first by systrace:
   a. Capture the systrace log and open the systrace log with a Chrome browser.
   b. Calculate the fps with an amount of decomposition and duration.



2. Verify the GPU busy rate.
   - Check the GPU busy rate by typing the following command and comparing it to MTP.

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

   – GPU busy rate = (active time per frame/total time per frame) * 100
   – Total time means active time per frame + nap time per frame
   – The value of gpubusy is reset if the GPU goes to slumber.
   – For example, (355818/1013309) * 100 = 35.1% per frame
   – If there is a different GPU busy rate compared to the MTP, the GPU is used by another context.

3. File a case with the proper debug information. See *Presentation: Graphics Power and Performance Overview* (80-NP885-1) for more debug-related information.

4.11.1 Script to monitor GPU usage

1. Copy and paste the following script into a file named 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. From a Linux or Windows machine with Perl, type the following command:

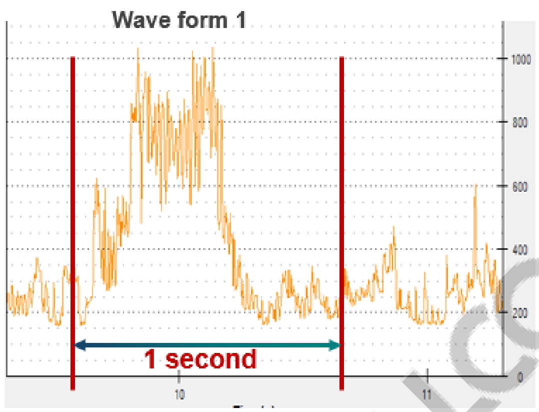
```
$./perl gpu_busy.pl
```

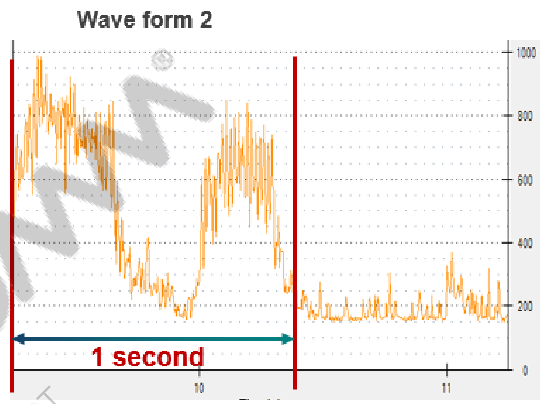
4.12 Analyze waveforms

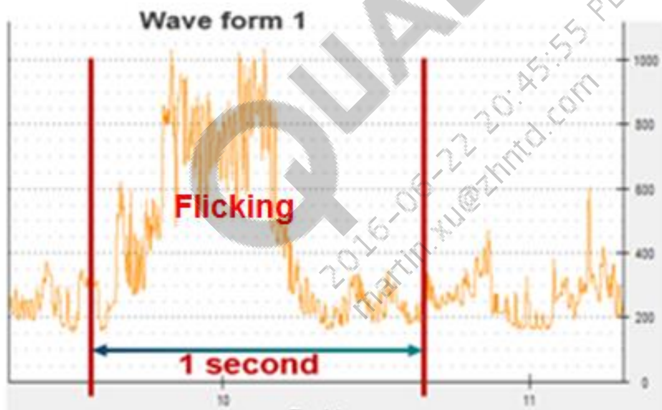
For a basic understanding of waveform analysis, read the following statements in conjunction with the images and systrace logs that follow.

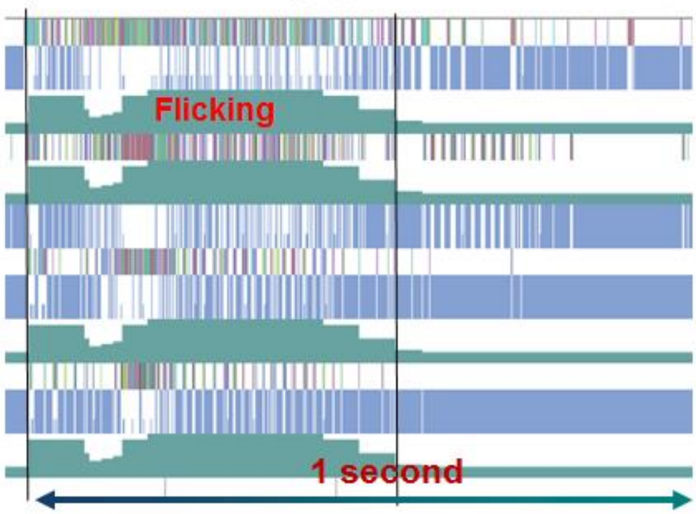
- In the following images, observe that waveform 1 and waveform 2 have different behavior, i.e., there are two waves per one flicking wave from waveform 2 compared to waveform 1.
- To determine the problem, use the systrace tool to debug in detail. A Chrome browser is required.
- Capture a systrace log and open it with a Chrome browser.
- Align each waveform and each systrace log by timeframe.
- Observe that the systrace log for waveform 1 and 2 have different behaviors.
- Ensure that the second area of the systrace log for waveform 2 is systrace log 2.

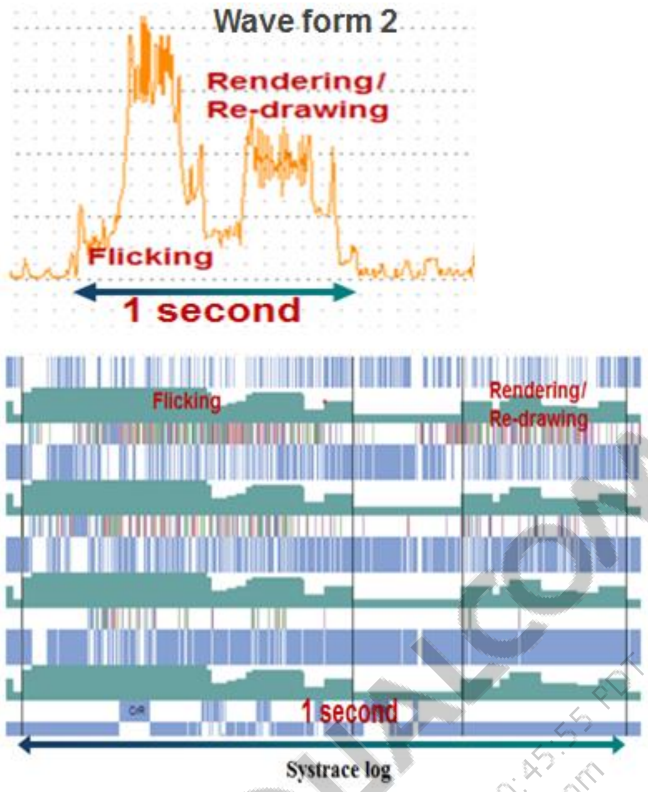
- In the systrace log, see the render thread and composition thread.
- As a result of the analysis, the second wave in waveform 2 is caused by render thread and composition thread.
- In this case, discuss whether this is the correct behavior with the graphic engineer and display engineer.



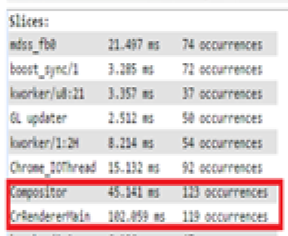








systrace log 2



You can see compositor for drawing and render thread

4.13 Analyze BIMC clock

- Use the msmbusvoting tool to analyze the BIMC clock.
- Check the clock and bandwidth voting by each client in real time.

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

- Determine which client voted BIMC clock in real time, as shown in the following example:

```
C:\Android\adt-bundle-windows-x86_64-20140702\sdk\tools>systat.py --clock bimc_clk --msm_bus qcom,cpubw.26 qcom,gpubw.19 mdss_mdp msm_camera_isp
   bimc   ¦qcom,cpubw.26(masters/slaves/ab/ib) ¦qcom,gpubw.19(masters/slaves/ab/ib) ¦              mdss_mdp(masters/slaves/ab/ib)              ¦msm_camera_isp(masters/slaves/ab/ib) ¦
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

460803552 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460798278 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460803552 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460803552 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460803478 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
931196447 ¦ 1 / 512 / 1593835520 / 7446986752 ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460801794 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460801721 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460800109 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460803552 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
537857542 ¦  1 / 512 / 796917760 / 4375707648  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460800036 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460801721 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460803478 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
460800036 ¦  1 / 512 / 597688320 / 3679453184  ¦ 26 / 512 / 595591168 / 1599078400  ¦22 23 / 512 512 / 526320000 526320000 / 526320000 526320000 ¦ 29 / 512 / 2253768640 / 3380652960  ¦
```

4.14 Check governor, scheduler, and CPU freq parameters

- Governor parameters
  - Check all nodes under the following paths:
    - Cortex-A53 (little cluster) − /sys/devices/system/cpu/cpu0/cpufreq/interactive/
    - Cortex-A57 (big cluster) − /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
```

- Scheduler parameters
  - Check all nodes under the /proc/sys/kernel/ path.

```
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 freq policy parameters
  - Cortex-A53 (little cluster) − /sys/devices/system/cpu/cpu0/cpufreq/
  - Cortex-A57 (big cluster) − /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
```

See cpufreq.h for more information – 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 Example of CPU freq policy parameters

- Customer reported 15 mA regression in browser use case compared to the previous build.
- Current version is v1.3 and previous version was v1.2.
- Scaling_min_freq on Cortex-A53 in v1.3 was 864 MHz, and scaling_min_freq on Cortex-A53 in v1.2 was 384 MHz, which is QTI default value.
- 864 MHz is requesting Turbo and 384 MHz is requesting LowSVS by the clock plan
  - v1.2 and 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
```

– Clock plan for Cortex-A53 (little cluster)

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

– Clock plan for Cortex-A57 (big cluster)

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 Check the temperature before measuring the power

1. Ensure that the comparative measurements are done at the same temperature.

   High temperature causes high current consumption.

2. Check the temperature by typing the following commands:

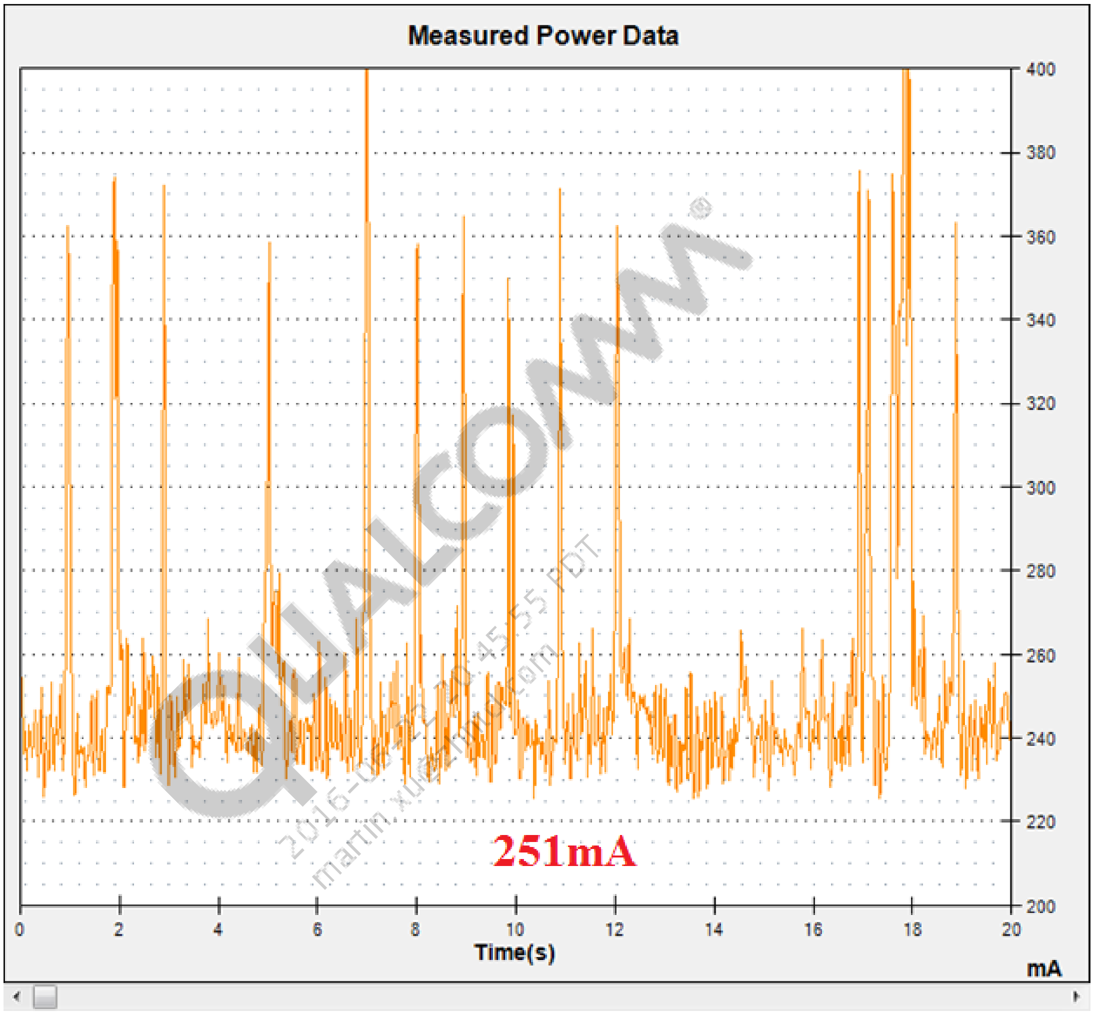
```
Cat /sys/devices/virtual/thermal/thermal_zone8/temp (temperature for
Core 0)
Cat /sys/devices/virtual/thermal/thermal_zone20/temp (temperature for
XO)
```

Most dashboard use case data is captured with 25ºC excluding the graphics use case (75ºC).

4.16 Verify Wi-Fi power

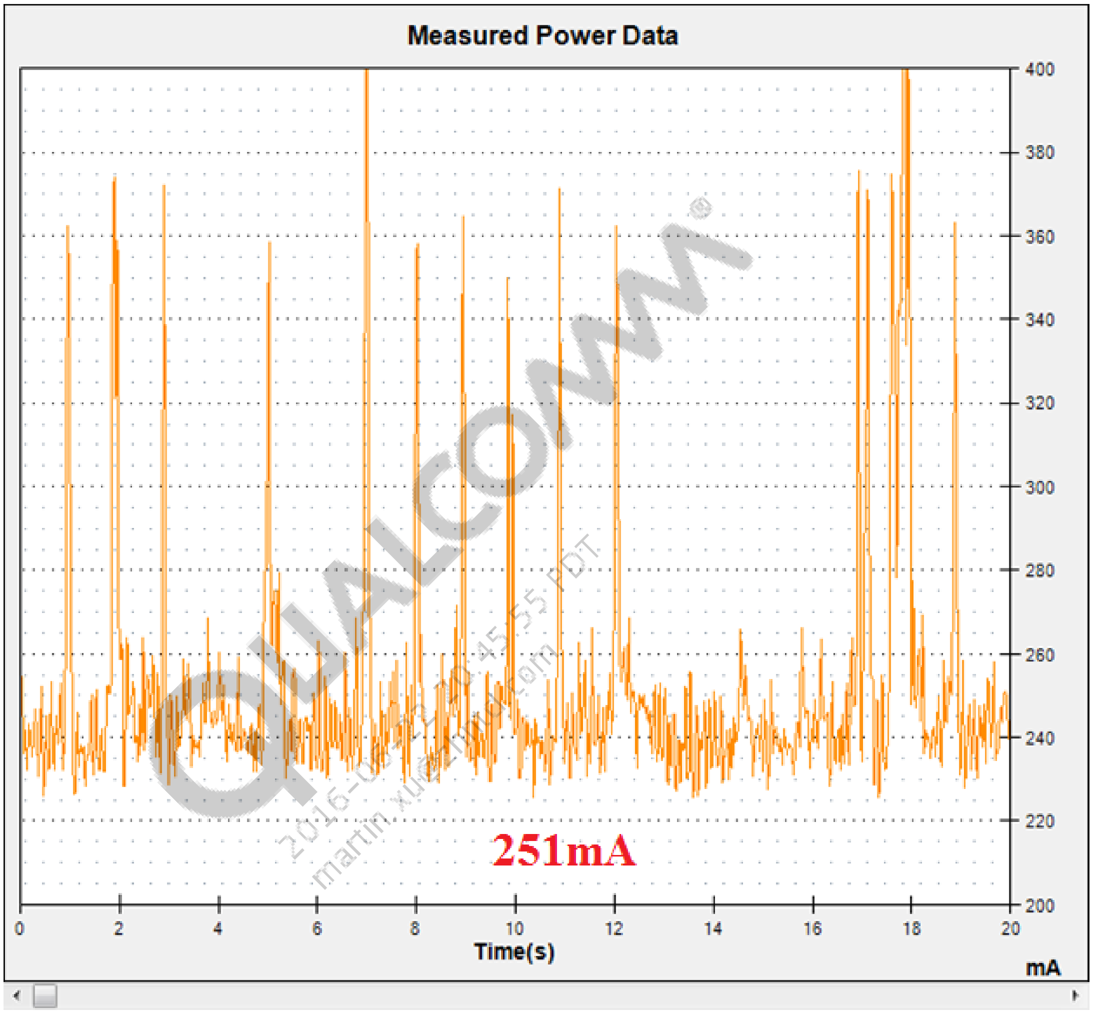
1. To verify the Wi-Fi power, use a dedicated apps processor.
2. Open a browser with any page; see the following icon that shows Wi-Fi power is turned on:

3. Measure power during 20 sec after the page loading is done.
   - Waveform with Wi-Fi on



4. Turn off Wi-Fi; see the following without the icon that shows Wi-Fi power is turned off:

5. Measure power again during 20 sec.
   - Waveform with Wi-Fi off



6. Compare the power number between Wi-Fi on and off.

   If there is a power difference that means the Wi-Fi power portion is in total power.

7. If high power is consumed by Wi-Fi, discuss it with the Wi-Fi engineer or solution provider.

4.17 Camera preview debugging

4.17.1 Camera subsystem clocks

1. Follow the steps in Section 5.4 to obtain clock dumps.
2. Check for the following clocks:

Clock name	Description
camss_vfe_vfe0_clk	VFE0 clock
camss_vfe_vfe1_clk	VFE1 clock (if applicable)
camss_vfe_cpp_clk	CPP clock
fd_core_clk	Hardware FD clock (if applicable)
bimc_a_clk	BIMC clock

4.17.2 Measure camera preview fps

1. Type the following commands:

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

2. Restart the camera; and in preview, type the following command:

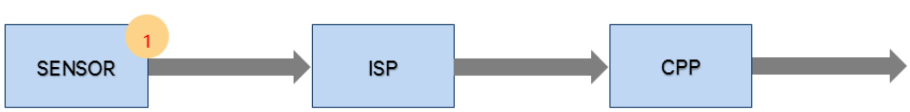
```
adb logcat | grep sensor_pick_resolution
```

The following is the resolution and mode selected for the use case:

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

The above log indicates that the sensor driver has opened the sensor using readout mode 0 (res idx 0). This is typically the full resolution mode of the sensor.

4.17.3 Measure sensor resolution during use case



1. Type the following command:

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

2. Restart the camera; and in preview, type the following command:

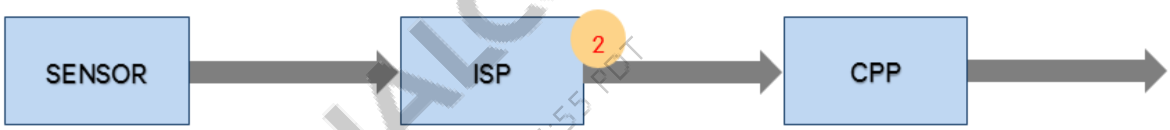
```
adb logcat | grep sensor_get_output_info
```

The preview fps logs are similar to the following:

```
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
```

The resolution output by the camera sensor is 2104 x 1560. The resolution of the sensor clock is 319200000.

4.17.4 Measure ISP configuration during camera use case



- Type the following commands:

```
adb logcat | grep  port_sensor_caps_reserve
```

The following shows the ISP configuration logged:

```
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 is the camera snapshot stream. The resolution of this stream is 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 is the FaceDetect stream. This resolution of this stream is 640 x 480.

4.17.5 Modify camera preview resolution

To reduce the camera preview size:

1. Type the following commands:

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

2. Use the different size options available in the QTI Snapdragon Camera application:
   - Value: 0 – Default size per snapshot aspect ratio
   - Value: 1 – 640 x 480
   - Value: 2 – 720 x 480
   - Value: 3 – 1280 x 720
   - Value: 4 – 1920 x 1080

   For example:

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

3. To reset to the default resolution:

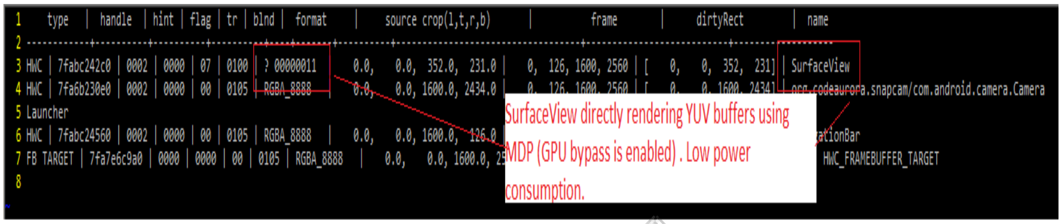
```
adb shell setprop persist.camera.preview.size 0
```

4.17.6 Check camera using GPU for preview buffer render

1. Run the following command on the device

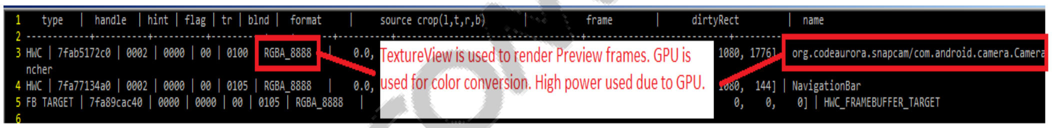
```
$adb shell dumpsys SurfaceFlinger
```

2. Check the color format and for SurfaceView layer in the layers.
   - SurfaceView example



The Dumpsys above indicates SurfaceFlinger got a YUV buffer to be rendered and it was sent via SurfaceView. The GPU is not involved in this case of color conversion.

   - TextureView example



The Dumpsys above indicates SurfaceFlinger got an RGB buffer to be rendered, and it was sent via TextureView. The GPU is involved and it does color conversion. There is higher GPU usage with this case.

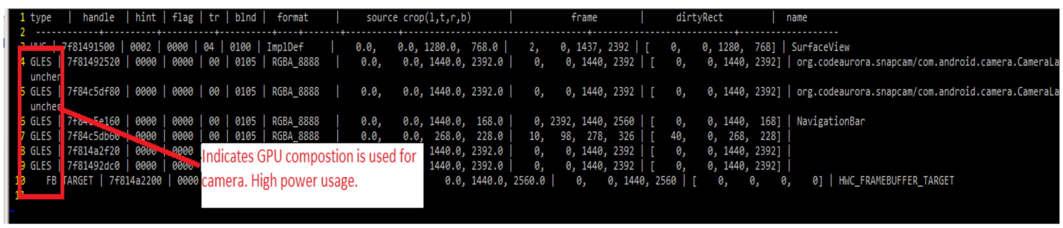
See Section 4.18.2 for a procedure to use SurfaceView instead of TextureView for preview UI.

4.17.7 Determine if camera uses GPU for composition instead of MDP/overlay

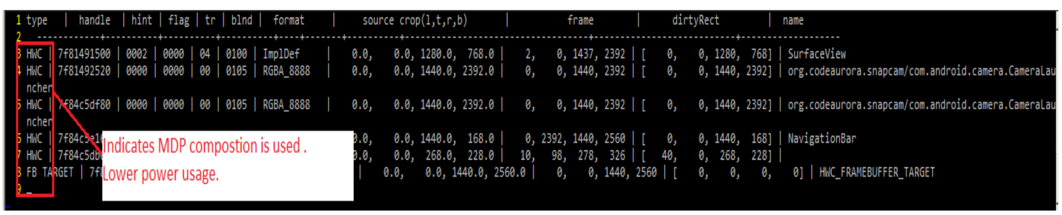
- Run the following command on the device:

```
$Adb shell dumpsys Surfaceflinger
```

   - SurfaceFlinger dumpsys during GPU composition – GPU is used for composition.



   - SurfaceFlinger dumpsys during MDP/overlay composition – GPU is not used for composition.



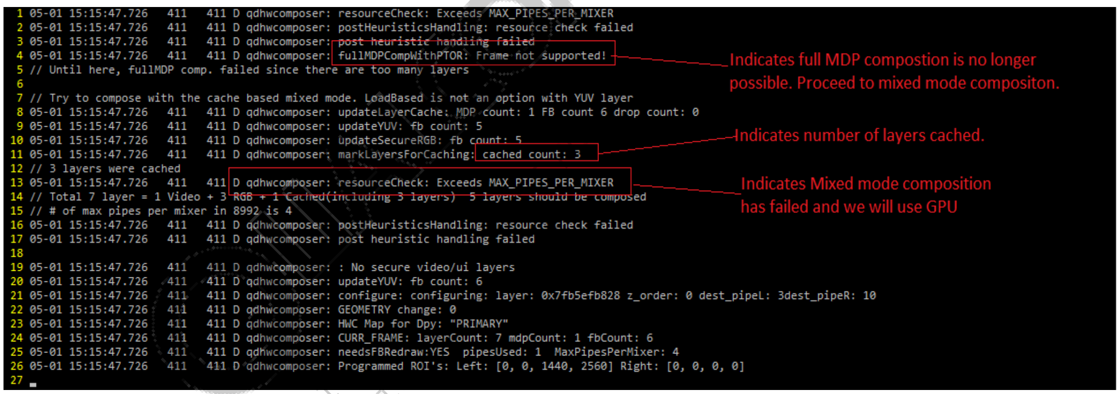
4.17.7.1 Identify reason for GPU composition

GPU composition is enabled when the number of layers to be composed is greater than the number of pipes/overlays available for MDP-based composition. This usually happens when there are several layers created by applications and sent for composition to hardware composition.

1. Enable the following logs in the display module:

```
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. Rerun the camera use case, and capture the logcat logs.
3. Check for following log pattern in the logcat:



```
05-01 15:15:47.726   411   411 D qdhwcomposer: resourceCheck: Exceeds MAX_PIPES_PER_MIXER
05-01 15:15:47.726   411   411 D qdhwcomposer: postHeuristicsHandling: resource check failed
05-01 15:15:47.726   411   411 D qdhwcomposer: post heuristic handling failed
05-01 15:15:47.726   411   411 D qdhwcomposer: fullMDPCompWithPTOR: Frame not supported!
// Until here, fullMDP comp. failed since there are too many layers

// Try to compose with the cache based mixed mode. LoadBased is not an option with YUV layer
05-01 15:15:47.726   411   411 D qdhwcomposer: updateLayerCache: MDP count: 1 FB count 6 drop count: 0
05-01 15:15:47.726   411   411 D qdhwcomposer: updateYUV: fb count: 5
05-01 15:15:47.726   411   411 D qdhwcomposer: updateSecureRGB: fb count: 5
05-01 15:15:47.726   411   411 D qdhwcomposer: markLayersForCaching: cached count: 3
// 3 layers were cached
05-01 15:15:47.726   411   411 D qdhwcomposer: resourceCheck: Exceeds MAX_PIPES_PER_MIXER
// Total 7 layer = 1 Video + 3 RGB + 1 Cached(including 3 layers)  5 layers should be composed
// # of max pipes per mixer in 8992 is 4
05-01 15:15:47.726   411   411 D qdhwcomposer: postHeuristicsHandling: resource check failed
05-01 15:15:47.726   411   411 D qdhwcomposer: post heuristic handling failed

05-01 15:15:47.726   411   411 D qdhwcomposer: : No secure video/ui layers
05-01 15:15:47.726   411   411 D qdhwcomposer: updateYUV: fb count: 6
05-01 15:15:47.726   411   411 D qdhwcomposer: configure: configuring: layer: 0x7fb5efb828 z_order: 0 dest_pipeL: 3dest_pipeR: 10
05-01 15:15:47.726   411   411 D qdhwcomposer: GEOMETRY change: 0
05-01 15:15:47.726   411   411 D qdhwcomposer: HWC Map for Dpy: "PRIMARY"
05-01 15:15:47.726   411   411 D qdhwcomposer: CURR_FRAME: layerCount: 7 mdpCount: 1 fbCount: 6
05-01 15:15:47.726   411   411 D qdhwcomposer: needsFBRedraw:YES  pipesUsed: 1  MaxPipesPerMixer: 4
05-01 15:15:47.726   411   411 D qdhwcomposer: Programmed ROI's: Left: [0, 0, 1440, 2560] Right: [0, 0, 0, 0]
```


4.18 Camera power optimization techniques

4.18.1 Dual ISP configuration

Some of the recent Snapdragon chipsets support two image processing cores that could process the Bayer Input to YUV. When sensor resolution is high, the camera software automatically splits the sensor output separately and passes it to both ISPs.

When using higher resolution sensors, QTI recommends that OEMs enable the Dual ISP feature so that the data processed by the ISPs is split evenly across two ISP hardware blocks instead of a single ISP.

- Enable Dual ISP – adb shell setprop persist.camera.isp.dualisp 1
- Disable Dual ISP – adb shell setprop persist.camera.isp.dualisp 0

Enabling Dual ISP for lower resolutions is not beneficial and sometimes can consume more power. QTI recommends following the guidelines shown in the table.

Follow the guideline sensor resolutions above as to which Dual ISP must be enabled (assumption – 30 fps ZSL preview).

MSM	Single ISP (MP)	Dual ISP (MP)
MSM8996	≤ 8	> 8
MSM8994	≤ 13	> 13
MSM8992	≤ 13	> 13
APQ8084	≤ 13	> 13
MSM8952	≤ 8	> 8
MSM8956	≤ 8	> 8

4.18.2 Use SurfaceView instead of TextureView

Camera applications can either use the TextureView or SurfaceView to render the camera preview buffers. With TextureView, the camera preview area is part of the Android view hierarchy. However, this causes the preview frames to go through the GPU for YUV to ARGB conversion. The UI layer and preview frames blending along with the GPU consume more power compared to using SurfaceView.

With SurfaceView, the preview content is directly composited as an overlay by the display hardware and there is no GPU involvement.

For more details, go to https://source.android.com/devices/graphics/architecture.html#surface-or-texture.

The Snapdragon camera application uses SurfaceView by default. See the following commit that updates the Snapdragon camera application to use SurfaceView instead of TextureView:

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

To convert camera applications to use SurfaceView instead of TextureView for Camera preview:

1. Replace TextureView with SurfaceView.
2. Replace SurfaceTexture with SurfaceHolder.
3. Modify TextureView.SurfaceTextureListener to SurfaceHolder.Callback.
4. Modify TextureView.setSurfaceTextureListener(TextureView.SurfaceTextureListener) to surfaceHolder = surfaceView.GetHolder();
   surfaceHolder.addCallback(SurfaceHolder.Callback).
5. Replace the setPreviewTexture(SurfaceTexture) of the camera object with setPreviewDisplay(SurfaceHolder).
6. For proper scaling to take effect, set the SurfaceView within a type of layout, which should not be a part of the root <merge> tag.
7. If the preview dimension does not exactly match the screen, in the SurfaceView path, manually rescale the preview with the correct aspect ratio when the surface changes as follows:

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

No side effect with IQ or performance is expected. If the application uses complex transformations and animation effects on camera preview data, it cannot be done with SurfaceView.

4.18.3 Reduce camera preview size

If the Sensor mode is set to a higher resolution than the display size or if the MDP hardware supports upscaling, consider requesting a smaller preview size from the camera.

Reducing the preview size to an optimal size reduces the traffic from the ISP to the DDR bus and from the DDR bus to the MDP. This reduces the power consumption for camera preview use cases.

No side effect with IQ or performance is expected.

4.18.4 CPP mirroring during camcorder use case

The preview and video size can be configured differently along with the color formats. If there is a difference in the preview or video size or in the color format, CPP takes two pass processing paths that result in more power consumption.

OEMs can avoid two pass processing in CPP by configuring the preview and video to be of same format and resolution.

If the following log is output in adb, the software can trigger the CPP output duplication feature for the use case:

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

Preview and video streams have the same IQ settings applied. Apart from that, there are no adverse effects with this change.

4.18.5 Disable Temporal Noise Reduction (TNR)

The TNR feature is available on select chipsets. The complexity of this algorithm depends on the resolution of the image frame. Disabling it reduces DDR bandwidth and overall power consumption during high resolution use cases, such as 4K recording and preview frames.

- To disable TNR, type the following commands:

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

Rerun IQ and performance testing after this change. This change can impact IQ.

4.18.6 Set the sensor mode resolution to video resolution

1. Set the sensor mode resolution to the same as video recording resolution instead of full resolution to reduce the power consumption in the camcorder recording use case. For example, set a 13 MP xensor mode to 1080p for 1080p video camcorder recording.

2. Similarly, set the sensor mode to display size or lower size to reduce the power consumption in a non-ZSL preview use case.

QTI recommends that the Sensor mode is set to a lower resolution in camcorder and camera preview use cases, if the device feature set permits it, or if tuning is available for the respective resolution size. Users can make changes in their respective sensor driver.

No impact is anticipated. Preview frames are output at a lower resolution.

4.18.7 Disable tintless

- To disable tintless to reduce power consumption, type the following command:

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

Rerun IQ and performance testing after this change. This change can impact IQ.

4.18.8 Disable Chromatic Aberration Correction (CAC)

The CAC block is a special hardware block with the camera ISP. It is present on some of the high end chipsets to provide finer noise cleaning capability. Disable this feature for preview or for preview and snapshot to get power savings.

CAC is disabled in the Chromatix™ color optimization tool (Chromatix) file. Submit a case for steps to disable the CAC feature.

Check for any IQ impact when the CAC operation is disabled from the camera preview.

4.19 Video recording power optimization techniques

4.19.1 Encoder power save mode

The encoder power save mode is an optional mode power optimization feature supported for the single session of a 4K H.264 camcorder.

- To enable the encoder power save mode, type the following command:

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

The feature is exposed through the OMX_QcomIndexConfigVideoVencPerfMode OMX extension.

See *Video Power Encoder Save Mode and Encoder DCVS* (80-NV307-1) for more details about this feature.

This option is applicable only for UHD (4K) encoding. It also requires the VFE to be running at Nominal clocks instead for Turbo.

There can be a slight impact with the quality of encoding in this mode.

5 Log collection

5.1 RPM logs

RPM publishes a small log into a limited area of spare message RAM. The physical format of the log is the uLog format used for various other logs. It is a circular buffer, currently sized at about 4 kB. It is a raw log with a set of IDs and a variable number of parameters per message.

5.1.1 Save RPM RAM dumps

In MSM8994, disabling the RPM halt is not necessary for keeping the JTAG daisy-chain connection.

To save the RPM RAM dumps:

1. Open RPM T32 and attach using the sys.m.a command.
2. Create the issue scenario where the RPM dumps are needed.
3. Break T32 at the desired point and run the following script to save the RPM logs:

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

5.1.2 Load RPM dumps onto T32 and extract logs

To load the RPM dumps onto T32 simulator for further analysis:

1. Open the T32 simulator and do sys.up.
2. Run the following script:

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

3. Load RPM.elf to start debugging.

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

5.1.2.1 RPM external logs (RPM uLog) using T32

1. While attached to the RPM and in the Break state or if the RPM RAM dumps are loaded on the T32 simulator, run the following command in T32:

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

   This places the RPM external logs into the desired log directory. The RPM external log requires the use of a Python parsing tool to interpret its contents.

2. Use the following Python script on extracted logs using a command prompt:

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

5.1.2.2 RPM NPA logs using T32

While attached to the RPM and in the Break state or if the RPM RAM dumps are loaded on the T32 simulator, run the following command in T32:

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

5.1.2.3 RPM logs using Hansei script

Hansei – An RPM RAM dump parser tool found in the rpm_proc\core\bsp\rpm\scripts\hansei folder (needs Python Ver 2.7.2)

```
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
```

- Summary of the output file:
  - rpm-summary.txt – Contains general information about the health of the RPM, including the core dump state and various fault information
  - rpm-log.txt – Postprocessed RPM external log
  - rpm-rawts.txt – Postprocessed RPM external log with the raw timestamp
  - npa-dump.txt – Standard NPA dump format
  - ee-status.txt – Contains information about the subsystems (and their cores) that are active or sleeping
  - reqs_by_master/* – Folder containing a file for each execution environment, detailing all current requests that EE has in place with the RPM
  - reqs_by_resource/* – Folder structure containing a folder for each of the resource types registered with the RPM server and, under that folder, a file containing all of the requests to each resource of that type

5.2 Collect GPIO dumps

All GPIOs in the hardware use the following script, which can be run from the RPM T32 window, to read the current configuration of all GPIOs in the hardware.

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

1. Select **Option 14: Read All GPIO Configurations**.
2. For default sleep configuration stored in software, use the following script to read, which is read out from TLMM.xml:

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

5.3 Collect PMIC dumps

1. Open an RPM T32 window.
2. Type sys.m.a to attach T32 to the device under test.
3. Load the .elf file from <RPM Build>/rpm_proc/core/bsp/rpm/build/*.elf and set the breakpoint if that is needed.
4. Recreate the use case scenario.
5. Break T32 using one of the following methods:
   - Click **Pause** in the T32 user interface.
   - Press **F8**.
   - User-defined breakpoint.

6. After the breakpoint is hit, type the following:

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

   By default, the raw PMIC dump file is saved in c:\temp\pmicdump.xml.

7. Type the following command to parse the pmicdump.xml file:

```
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 Collect clock dumps

Use the JTAG method if the device has JTAG capability. If the device does not have JTAG, use the ADB method for clock dumps.

5.4.1 Collect clock dumps using JTAG

1. Open an RPM T32 window.
2. Type sys.m.a to attach T32 to the device under test.
3. Load the .elf file from <RPM Build>/rpm_proc/core/bsp/rpm/build/*.elf and set breakpoint if needed.
4. Recreate the use case scenario.
5. Break the T32 by one of the following ways:
   - Click **Pause** in the T32 user interface.
   - Press **F8**.
   - User-defined breakpoint.
   - Type the following command:

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

6. To obtain a dump of all clocks, type **all** in the pop-up window.

   The clock dump is printed in the message area of T32.

5.4.2 Collect clock dumps using the adb shell

1. Connect the USB.
2. Run the following command:

```
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. Unplug the USB when PID appears.
4. Run the test scenario <execute test scenario for at least 2 min>.
5. Connect the USB and kill the clock checking process ID by typing the following commands:

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

5.5 Collect modem logs

5.5.1 Collect modem uLogs

- Execute the following script from Modem_Build in the modem T32 window:

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

This places the modem uLogs in the specified directory.

5.5.2 Collect modem NPA logs

- Execute the following script from Modem_Build in the modem T32 window:

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

This places the modem NPA logs in the specified directory.

5.6 Collect modem F3 messages/QXDM Pro logs

1. Connect the device to a PC via USB where the QXDM Pro is installed.
2. Load the logs mask (dmc file) through File > Load configuration.
3. Press **F3** or select **Messages View** from the View tab.
4. Press **Alt+L** to start logging, and press **Alt+L** again to stop the logging.
5. Obtain the saved logs from the location defined in Options > Log View Config > Misc > Log File Path.

5.7 Capture ftrace logs

NOTE: Numerous changes were made in this section.

1. Connect the USB.
2. Type the following commands:

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

3. To check the buffer_size_kb, type the following command:

```
Cat buffer_size_kb
```

4. To capture ftrace logs, type the following command:

```
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. Unplug the USB when PID (process ID) appears.
6. Run the use case.
7. After completing the use case, connect the USB and place the trace file from the device to your PC by typing the following command:

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

5.8 Capture PowerTop and Top data

1. Connect the USB.
2. Capture the PowerTop and Top data by typing the following commands:

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

3. Start capturing PowerTop and Top data by typing the following commands:

```
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. Disconnect the USB when the PID (process ID) appears.
5. Run the use case.

6. Connect the USB and kill the process after completing the use case by typing the following commands:

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

5.9 Capture clock and regulator dumps

Capture the Top and PowerTop data:

1. Connect the USB.
2. Type the following commands:

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

3. Start capturing PowerTop and Top data by typing the following command:

```
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. Disconnect the USB when the PID (process ID) appears.
5. Run the use case.

6. Connect the USB and kill the process after completing the use case by typing the following commands:

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

5.10 Collect wake locks

- Type one of the following commands:

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

or

```
adb shelldumpsys power
```

A References

A.1 Related documents

Title	Number
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 Acronyms and terms

Acronym or term	Definition
APSS	Applications Processor Subsystem
CAC	Chromatic Aberration Correction
CAF	code aurora forum
DRX	discontinuous reception
LPASS	Low Power Audio Subsystem
MPSS	Modem Peripheral Subsystem
Reference Log	debug log obtained from a reference device where the current consumption meets the goals for the particular chipset
spm	System Power Monitor
YUV	luminance, bandwidth, chrominance; also known as YCbCr and YPbPr