SDM845 Extensive Power

Debug Guide

80-P9301-116 Rev. B

April 17, 2018

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

Revision history

Revision	Date	Description
A	September 2017	Initial release
B	April 2018	Added RPMh master stats and hansei RAM dump parser information

Contents

1 Introduction

1.1 Purpose

This document provides details on how to optimize specific power test cases and debug issues related to the resource power manager (RPM), application processor subsystem (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**.

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/ with the following information:

- Chipset – AMSS build ID and operating system
- Initial problem type – Software
- For multimedia use cases, select the following problem areas:
  - 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, select the following problem areas:
  - 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
  - Describe the use case if different from the QTI standard use case
  - List steps to reproduce the issue
  - Describe the debugging done so far
  - Include the following debugging logs:
    - Waveforms, rail-level breakdown, node power architecture (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

QTI recommends the following tools to ensure a complete and effective power debugging process.

2.1 Required software tools and setup

Specific debugging tools are required for use cases involving RPM, modem, and apps processor.

2.1.1 Debugging tools for RPM and modem use cases

- Trace32 (T32) software – This is essential for using JTAG for power debugging, especially to obtain on-target logs, load RAM dumps, and capture off-target logs using a T32 simulator.
- Licensed QPST and QXDM Professional™ tools – These tools are necessary to obtain logs and messages pertaining to modem power use case debugging.

2.1.2 Debugging tools for apps processor use cases

Tool	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	http://python-xy.github.io/ After installation, open a command prompt in C:\ and run easy_install pytimechart.	http://python-xy.github.io/
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, PMIC, 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 System sleep state and debugging

In RPM, system-level sleep is tightly coupled into two modes: XO shutdown and VDD-MIN. However, with RPMh, there are no sleep modes, only sleep states for a resource. SoC resource sleep states are independent. The following sections describe CX power collapse and AOSS sleep.

3.1 CX power collapse

With the introduction of RPMh, a new operating point (i.e., OFF) has been added to the VDD_CX domain where a dependent resource (e.g., clock) would have voted for RETENTION in RPM can now vote for OFF.

Picking up CX collapse is an RPMh decision, based on whether there is enough time, and if it is the most energy efficient mode

3.1.1 CX power collapse entry criteria

- Clients (DRVs) vote to CX for OFF
- Each subsystem PDC indicates CX collapse possible with power toggle
- AOP is interrupted when all PDCs indicate that CX collapse is possible
- AOP checks the entry criteria for CX collapse
- Criteria includes CX and MX at retention so that CPR is disabled and energy efficient to perform CX collapse (if these criteria are not met, then CX collapse will not be performed)
- AOP raises CX voltage to meet droop requirement.
- AOP also raises MX voltage to meet voltage relationship requirement
- AOP votes CX for OFF
- ARC transitions CX to OFF
- AOP is notified from VRM when CX is written to OFF
- AOP restores MX voltage to retention level

3.1.2 CX power collapse exit criteria

- In case of interrupt-based wakeup, specific subsystem PDC will indicate CX restore required with power toggle
- In case of timer-based wakeup, AOP timer will expire, which results in CX restore

- AOP raises MX voltage to meet voltage relationship requirement (system gets blocked in this step until MX voltage is raised)
- AOP votes CX to retention
- ARC transitions CX to retention
- ARC sends completion to AOP
- AOP restores MX voltage to retention level

3.1.3 CX power collapse common debug scenarios

CX collapse did not enter

- Check CX aggregation for DRVs with vote greater than 0 from ARC dump script
- If aggregation is greater than 0, debug corresponding DRVs subsystem is responsible for blocking CX collapse
- See ARC debugging

CX collapse did not exit

- Check each subsystem PDC to find the wakeup time for the expected subsystem
- Check PDC wakeup interrupt details for each subsystem
- If wakeup timer/interrupt is incorrect, check the corresponding subsystem
- See PDC debugging

CX collapse exits early

- Check whether some interrupt fired early
- Check whether a timer is programmed later than expected
- Check the operating level of CX in ARC dump script and the DRVs that voted for it
- Check PDC interrupt/timer details for further exit info from PDC dump script
  - See ARC/PDC debugging

RELATED INFORMATION

“ARC logs using T32” on page 61
“PDC logs using T32” on page 69

3.2 AOSS sleep

With the introduction of RPMh, AOSS sleep is the lowest system level sleep state supported.

3.2.1 AOSS sleep common debug scenarios

AOSS sleep did not enter

- RPMh blocks must be idle and subsystems must have sent their AOSS power toggle
- Check subsystems to see if any did not go to RPMh assisted sleep (sleep logs)
- Check RPMh block states (idle or not?) for VRM/BCM/ARC/PDC (dump scripts)

- If any of these are busy, it is preventing AOSS sleep
- SoC sleep mask for power toggle:
  - RPMH_PDC_SOC_SLEEP_LOCK_MASK_BANK0 | 0x0B2E0300
    - (Any SS bit that is 0 requires a power toggle to notify PDC that AOSS sleep can be entered)
  - RPMH_PDC_SOC_SLEEP_LOCK_STATUS_BANK0 | 0x0B2E0310
    - (For all SS with a MASK of 0, their LOCK_STATUS should be 1)

AOSS sleep did not exit

If this can be reproduced, break the system on the way into AOSS sleep in AOP and check for sanity.

- Check whether PDC wakeup interrupts are configured properly from PDC dump script
- Check that wakeup timers are properly configured

AOSS sleep early exit

- Varies depending on what stage of wakeup
- Consider the following items for early wakeup:
  - AOP state – Is AOP up and running? If so, what is the software state regarding AOSS sleep?
  - PMIC state – Did the PBS sequence execute properly?
  - MX\LPI MX state – Are rails on such that AOP should be capable of running?
  - PDC state – If AOP woke up but nothing else, check the corresponding PDC wakeups from PDC dump

RELATED INFORMATION

“PDC logs using T32” on page 69
“Debug a subsystem wakeup issue” on page 73

3.3 CX collapse and AOSS sleep count stats – adb shell commands

Type the following command to obtain the sleep statistics:

```
cat /sys/power/system_sleep/stats
```

Example command output

```
RPM Mode:aosd
    count:13923
time in last mode(msec):0
time since last mode(sec):16
actual last sleep(msec):291000

 RPM Mode:cxsd
     count:2101
 time in last mode(msec):0
 time since last mode(sec):16
 actual last sleep(msec):294000
```

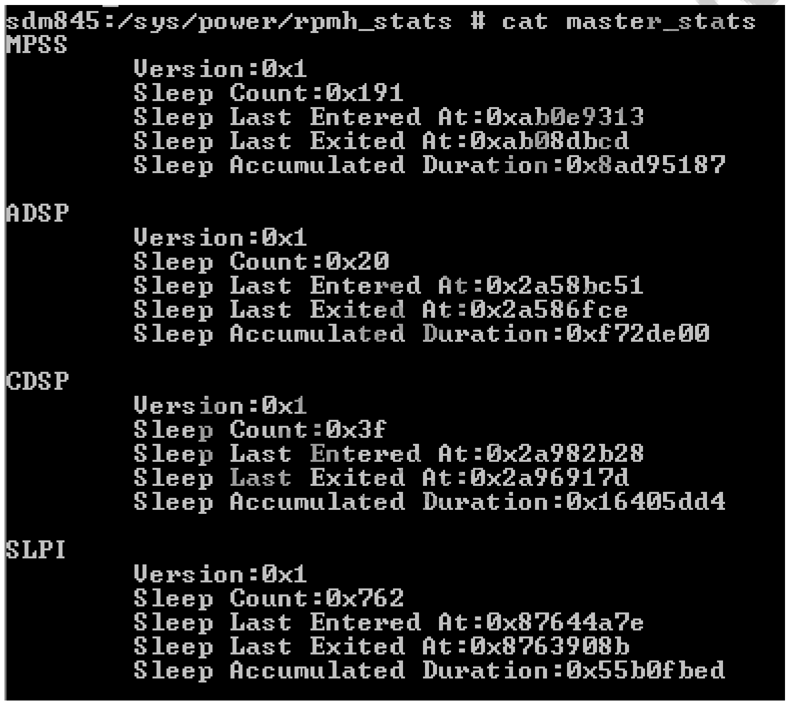
Alternatively, to verify if the system is entering CX collapse and AOSS sleep, use any of the following methods:

- T32 breakpoints on the AOP
- For CX collapse – `aop_cx_collapse_init()`
- For AOSS sleep – `aop_sleep_main()`

3.4 RPMh master stats (subsystem stats) – adb shell commands

Type the following command to obtain the master statistics:

```
cat /sys/power/rphm_sleep/master_stats
```



```
sdm845:/sys/power/rpmh_stats # cat master_stats
MPSS
        Version:0x1
        Sleep Count:0x191
        Sleep Last Entered At:0xab0e9313
        Sleep Last Exited At:0xab08dbcd
        Sleep Accumulated Duration:0x8ad95187

ADSP
        Version:0x1
        Sleep Count:0x20
        Sleep Last Entered At:0x2a58bc51
        Sleep Last Exited At:0x2a586fce
        Sleep Accumulated Duration:0xf72de00

CDSP
        Version:0x1
        Sleep Count:0x3f
        Sleep Last Entered At:0x2a982b28
        Sleep Last Exited At:0x2a96917d
        Sleep Accumulated Duration:0x16405dd4

SLPI
        Version:0x1
        Sleep Count:0x762
        Sleep Last Entered At:0x87644a7e
        Sleep Last Exited At:0x8763908b
        Sleep Accumulated Duration:0x55b0fbed
```

4 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.

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

RELATED INFORMATION

“Required software tools and setup” on page 8

“Required hardware tools” on page 9

4.1 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.	Technical assistance
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.	Analyze rockbottom waveform
4	For debugging a higher base current, verify that the device enters CX power collapse. ■ If the device does not enter CX power collapse, go to Step 5. ■ If the device enters CX power collapse, go to Step 6.	Verify CX power collapse and AOSS sleep

Step no.	Step	Reference
5	Check the subsystem status to determine which one is blocking the device from entering CX power collapse mode and debug further for that subsystem.	CX power collapse common debug scenarios
6	▪ File a case to obtain the following: ▫ A procedure to check CX voltage 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 switched-mode power supplies (SMPS) or low dropout regulators (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.	▪ Check leakage in sleep current ▪ 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, RPMh ARC/BCM, or APSS kernel logs.	Check unexpected wakeups
8	If power consumption is still higher than expected after following Steps 1 through 7, file a case with QTI for further debugging help.	Technical assistance

4.2 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.	Technical assistance
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 Rockbottom. ▪ For occurrences of unexpected wakeups, go to Step 7 in Rockbottom. ▪ If discontinuous reception (DRX) wakeup power consumption is higher, go to Step 4.	

Step no.	Step	Reference
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.	
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.	Analyze higher paging awake penalty
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.	Analyze higher paging awake penalty
7	If power consumption is still higher than expected after following Steps 1 through 6, file a case with QTI for further debugging help.	Technical assistance

4.3 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 CX power collapse. If the device does not enter CX power collapse, go to Step 5. ■ If the device enters CX power collapse, go to Step 15. No clocks are active if the device enters CX power collapse. ■ If using a dumb panel, go to Step 6.	Verify CX power collapse and AOSS sleep
5	Resolve the root cause for the device not entering CX power collapse and go to Step 15.	

Step no.	Step	Reference
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.	
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.	Debug SurfaceFlinger
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.	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.	Analyze high CPU usage process
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.	Analyze high CPU usage process
12	▪ If the DDR 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.	Analyze DDR clock
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.	Technical assistance
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.	Analyze interrupt
15	Ensure that the rockbottom use case is optimized or subtract the delta between the QTI rockbottom use case and customer device.	Rockbottom
16	Collect and compare the GPIO and PMIC LDO configurations with the QTI reference.	Check leakage in sleep current
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.	

4.4 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.	Technical assistance
2	Check Tunnel mode.	Tunnel mode and player type check
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.	Analyze high CPU usage process
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.	Analyze high CPU usage process
10	■ If the DDR 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.	■ Analyze DDR clock ■ Technical assistance

Step no.	Step	Reference
11	■ For frequent wakeups, check the period of the wakeup. ■ Capture PowerTop and ftrace and look for periodic wakeups.	Analyze interrupt
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.	Analyze interrupt
13	If dsp irq is observed frequently with a period less than 2 sec, check and resolve app wake lock issue.	Music application wake lock check and debugging
14	Ensure that the rockbottom use case is optimized, or subtract the delta between the QTI rockbottom use case and the customer device.	Rockbottom
15	Collect and compare the GPIO and PMIC LDO configurations with QTI reference.	Check leakage in sleep current
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.	Technical assistance

4.5 Video playback

Step no.	Step	Reference
1	Ensure that baseline rockbottom, static image, and MP3 use cases are optimized.	■ Rockbottom ■ MP3 playback
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 □ Sensors and other third-party components □ Different DDR size compared to the QTI reference □ Any additional postprocessing hardware ■ For this information, file a case.	Technical assistance
4	Check Tunnel mode and player type.	Tunnel mode and player type check
5	If using Nontunnel mode, which is not part of the QTI standard, ask QTI for the power impact for this variation.	
6	Obtain the power measurement on the device following the QTI standard power measurement test procedure.	80-N6837-1

Step no.	Step	Reference
	■ 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.	Debug SurfaceFlinger
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.	Analyze high CPU usage process
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.	Analyze high CPU usage process
14	■ If the DDR 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.	Analyze DDR clock
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.	Technical assistance
16	■ For frequent wakeups, check the period of the wakeup. ■ Capture PowerTop, ftrace and look for periodic wakeups.	Analyze interrupt
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.	Analyze interrupt

Step no.	Step	Reference
18	Collect and compare the GPIO, PMIC LDO configurations with the QTI reference.	Check leakage in sleep current
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.	

4.6 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.	Technical assistance
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 Check subsystem (APSS/MPSS/LPASS) power collapse from RPMh (PDC). ▪ 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 BCM dumps to determine if any clocks or shared resources are running at a higher level and which subsystems are voting for the same.	Inspect clocks and shared resources
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.	Check modem resource votings
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.	Check PA/RF power consumption
8	▪ If power consumption is still higher than expected after following Steps 1 through 7, file a case with QTI for further debugging help.	Technical assistance

4.7 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.	Technical assistance
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 BCM dumps to determine if any clocks or shared resources are running at a higher level and which subsystems are voting for the same.	Inspect clocks and shared resources
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.	Check modem resource votings
7	Check APSS-specific logs, such as PowerTop, Top, and ftrace logs, to determine what is causing the higher current consumption from the APSS.	Check APSS
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.	Check PA/RF power consumption
9	If power consumption is still higher than expected after following Steps 1 through 8, file a case with QTI for further debugging help.	Technical assistance

4.8 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. ▪ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	▪ Check the temperature before measuring the power ▪ 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.	Capture clock and regulator dumps
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.	Analyze high CPU usage process
6-1-1	Governor, scheduler, and CPU freq parameters: ▪ Compare the governor, scheduler, and CPU freq 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.	Check governor, scheduler, and CPU freq parameters
6-1-2	High CPU usage process.	Analyze high CPU usage process
6-1-3	Unexpected interrupt or interrupt count.	Analyze interrupt

Step no.	Step	Reference
6-2	DDR clock is high. ■ Compare the bandwidth voting under `/d/msm-bus-dbg/client-data/` with the MTP. ■ 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.	Analyze DDR clock
6-3	GPU clock is high.	Debug high GPU clock frequency
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.	Technical assistance
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.	Check leakage in sleep current
8	If power consumption is still higher than the QTI reference data, go to Step 6-4.	

4.9 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. ■ Obtain the power measurement on the device following the QTI standard power measurement test procedure.	■ 80-N6837-1 ■ Check the temperature before measuring the power
	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

Step no.	Step	Reference
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.	Verify Wi-Fi power
6	Compare the waveform with reference data.	Analyze waveforms
7	Capture all clocks and regulators to compare with QTI data.	Capture clock and regulator dumps
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, scheduler, and CPU freq parameters: ■ Compare the governor, scheduler, and CPU freq 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.	Check governor, scheduler, and CPU freq parameters
8-1-2	High CPU usage process	Analyze high CPU usage process
8-1-3	Unexpected interrupt or interrupt count.	Analyze interrupt
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 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.	Analyze DDR clock
8-3	GPU clock is high.	Debug high GPU clock frequency
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.	Technical assistance
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.	Check leakage in sleep current
10	If power consumption is still higher than the QTI reference data, go to Step 8-4.	

4.10 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 ■ Check the temperature before measuring the power
	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.	Verify Wi-Fi power
6	Check Tunnel mode and player type.	Tunnel mode and player type check
7	Capture all of the clocks and regulators to compare with QTI data.	Capture clock and regulator dumps
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, scheduler, and CPU freq parameters: ■ Compare the governor, scheduler, and CPU freq 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.	Check governor, scheduler, and CPU freq parameters

Step no.	Step	Reference
8-1-2	High CPU usage process.	Analyze high CPU usage process
8-1-3	Unexpected interrupt or interrupt count.	Analyze interrupt
8-2	DDR 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.	Analyze DDR clock
8-3	Other clock is high. File a case with the debugging information clock dump, ftrace, waveform, PowerTop, Top, systrace data, and power rail breakdown data.	Technical assistance
8-4	■ Must check FPS. ■ File a case for debug in detail if there is a different FPS.	■ Technical assistance ■ Debug high GPU clock frequency
9	If power consumption is still higher than the QTI reference data, go to Step 8-3.	

5 Dashboard use case debugging details

5.1 Rockbottom

The following sections describe methods to analyze and optimize rockbottom use cases.

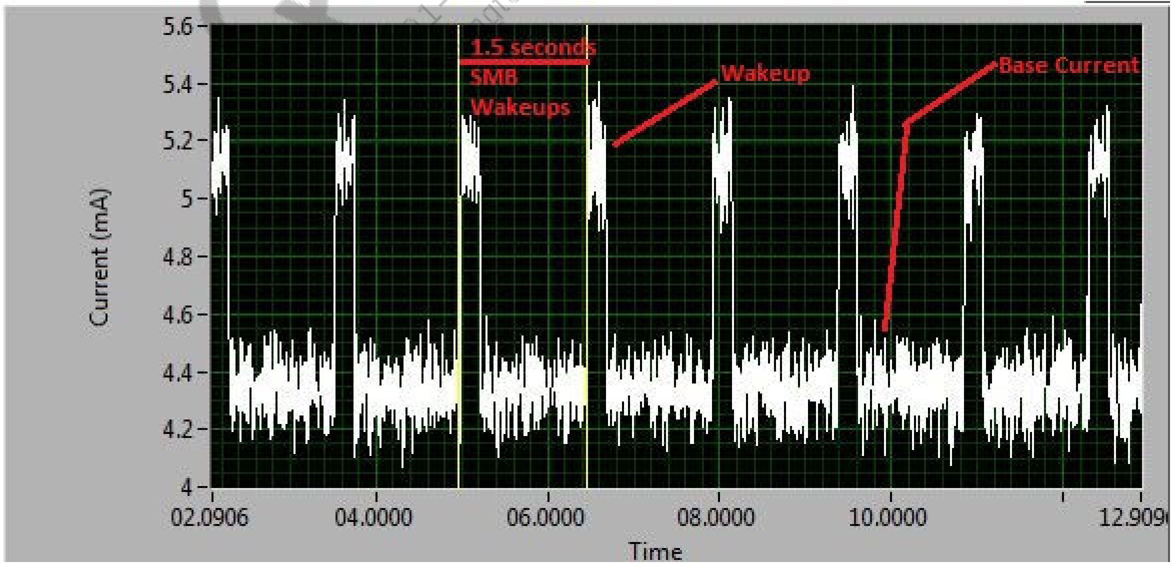
5.1.1 Analyze rockbottom waveform

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 is a snapshot of the rockbottom use case waveform taken on an MSM or SDM chipset:



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

5.1.2 Verify CX power collapse and AOSS sleep

To verify if the system is entering CX power collapse and AOSS sleep, use the following adb shell command to obtain RPM statistics:

```
cat /sys/power/system_sleep/stats
```

In the following example adb command output, the count represents the number of occurrences of CX collapse and AOSS sleep:

```
RPM Mode:aosd
    count:13923
time in last mode(msec):0
time since last mode(sec):16
actual last sleep(msec):291000

 RPM Mode:cxsd
     count:2101
 time in last mode(msec):0
 time since last mode(sec):16
 actual last sleep(msec):294000
```

Alternatively, to verify if the system is entering CX collapse, use ARC dump.

If the system enters CX power collapse, only the CXSD count is incremented. AOSS sleep is the lowest power state, and it includes CX power collapse. AOSS sleep count is only incremented when the system enters AOSS sleep mode.

RELATED INFORMATION

“ARC logs using T32” on page 61

5.1.3 Check subsystem (APSS/MPSS/LPASS) power collapse from RPMh (PDC)

See PDC logging for details.

RELATED INFORMATION

“PDC logs using T32” on page 69

5.1.4 Determine subsystem votes for major resources using RPM logs

The following major resources can be relinquished or voted for low power to attain AOSS sleep and CX power collapse:

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

Check the ARC block for votes by different subsystems/DRVs using the ARC dump script.

See ARC logging for details.

RELATED INFORMATION

“ARC logs using T32” on page 61

5.1.5 Check power collapse of individual subsystems

Individual subsystems prevent system power collapse for 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

5.1.5.1 Determine why the APSS is not voting for power collapse

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 &
   ```

   Note: The unit of time in wakeup_sources log is milliseconds.
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:
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.adb pull /data/wakelocks.txt <destination_folder>

5.1.5.2 Check clocks preventing CX power collapse/AOSS sleep

1. Type the following command to enable the clock debug suspend:

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

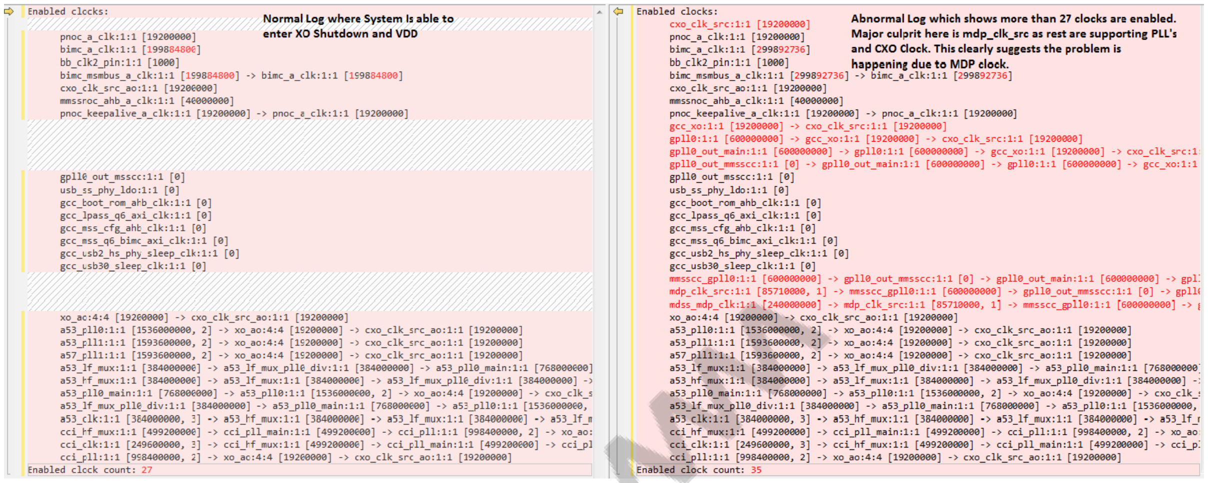
   After enabling this flag, the clocks that are enabled are shown when the system is going into Suspend mode in the dmesg logs.
2. Suspend and resume the system by pressing the power button three to four times, and type the following command to take a dmesg log:

   ```
   adb shell dmesg
   ```

Some clocks are always expected to be shown as enabled in this log. If any clocks other than usual major system clocks are shown as enabled, this 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.

The following figure shows a comparison of enabled clocks:



Normal Log where System Is able to enter XO Shutdown and VDD

```
Enabled clocks:
        pnoc_a_clk:1:1 [19200000]
        bimc_a_clk:1:1 [199884800]
        bb_clk2_pin:1:1 [1000]
        bimc_msmbus_a_clk:1:1 [199884800] -> bimc_a_clk:1:1 [199884800]
        cxo_clk_src_ao:1:1 [19200000]
        mmssnoc_ahb_a_clk:1:1 [40000000]
        pnoc_keepalive_a_clk:1:1 [19200000] -> pnoc_a_clk:1:1 [19200000]

        gpll0_out_msscc:1:1 [0]
        usb_ss_phy_ldo:1:1 [0]
        gcc_boot_rom_ahb_clk:1:1 [0]
        gcc_lpass_q6_axi_clk:1:1 [0]
        gcc_mss_cfg_ahb_clk:1:1 [0]
        gcc_mss_q6_bimc_axi_clk:1:1 [0]
        gcc_usb2_hs_phy_sleep_clk:1:1 [0]
        gcc_usb30_sleep_clk:1:1 [0]

        xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_pll0:1:1 [1536000000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_pll1:1:1 [1593600000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a57_pll1:1:1 [1593600000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_lf_mux:1:1 [384000000] -> a53_lf_mux_pll0_div:1:1 [384000000] -> a53_pll0_main:1:1 [768000000]
        a53_hf_mux:1:1 [384000000] -> a53_lf_mux:1:1 [384000000] -> a53_lf_mux_pll0_div:1:1 [384000000] ->
        a53_pll0_main:1:1 [768000000] -> a53_pll0:1:1 [1536000000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_s
        a53_lf_mux_pll0_div:1:1 [384000000] -> a53_pll0_main:1:1 [768000000] -> a53_pll0:1:1 [1536000000,
        a53_clk:1:1 [384000000, 3] -> a53_hf_mux:1:1 [384000000] -> a53_lf_mux:1:1 [384000000] -> a53_lf_m
        cci_hf_mux:1:1 [499200000] -> cci_pll_main:1:1 [499200000] -> cci_pll:1:1 [998400000, 2] -> xo_ao:
        cci_clk:1:1 [249600000, 3] -> cci_hf_mux:1:1 [499200000] -> cci_pll_main:1:1 [499200000] -> cci_pl
        cci_pll:1:1 [998400000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
Enabled clock count: 27
```

Abnormal Log which shows more than 27 clocks are enabled. Major culprit here is mdp_clk_src as rest are supporting PLL's and CXO Clock. This clearly suggests the problem is happening due to MDP clock.

```
Enabled clocks:
        cxo_clk_src:1:1 [19200000]
        pnoc_a_clk:1:1 [19200000]
        bimc_a_clk:1:1 [299892736]
        bb_clk2_pin:1:1 [1000]
        bimc_msmbus_a_clk:1:1 [299892736] -> bimc_a_clk:1:1 [299892736]
        cxo_clk_src_ao:1:1 [19200000]
        mmssnoc_ahb_a_clk:1:1 [40000000]
        pnoc_keepalive_a_clk:1:1 [19200000] -> pnoc_a_clk:1:1 [19200000]
        gcc_xo:1:1 [19200000] -> cxo_clk_src:1:1 [19200000]
        gpll0:1:1 [600000000] -> gcc_xo:1:1 [19200000] -> cxo_clk_src:1:1 [19200000]
        gpll0_out_main:1:1 [600000000] -> gpll0:1:1 [600000000] -> gcc_xo:1:1 [19200000] -> cxo_clk_src:1
        gpll0_out_mmsscc:1:1 [0] -> gpll0_out_main:1:1 [600000000] -> gpll0:1:1 [600000000] -> gcc_xo:1:1
        gpll0_out_msscc:1:1 [0]
        usb_ss_phy_ldo:1:1 [0]
        gcc_boot_rom_ahb_clk:1:1 [0]
        gcc_lpass_q6_axi_clk:1:1 [0]
        gcc_mss_cfg_ahb_clk:1:1 [0]
        gcc_mss_q6_bimc_axi_clk:1:1 [0]
        gcc_usb2_hs_phy_sleep_clk:1:1 [0]
        gcc_usb30_sleep_clk:1:1 [0]
        mmsscc_gpll0:1:1 [600000000] -> gpll0_out_mmsscc:1:1 [0] -> gpll0_out_main:1:1 [600000000] -> gpl
        mdp_clk_src:1:1 [85710000, 1] -> mmsscc_gpll0:1:1 [600000000] -> gpll0_out_mmsscc:1:1 [0] -> gpll
        mdss_mdp_clk:1:1 [240000000] -> mdp_clk_src:1:1 [85710000, 1] -> mmsscc_gpll0:1:1 [600000000] ->
        xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_pll0:1:1 [1536000000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_pll1:1:1 [1593600000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a57_pll1:1:1 [1593600000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
        a53_lf_mux:1:1 [384000000] -> a53_lf_mux_pll0_div:1:1 [384000000] -> a53_pll0_main:1:1 [768000000
        a53_hf_mux:1:1 [384000000] -> a53_lf_mux:1:1 [384000000] -> a53_lf_mux_pll0_div:1:1 [384000000] -
        a53_pll0_main:1:1 [768000000] -> a53_pll0:1:1 [1536000000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_
        a53_lf_mux_pll0_div:1:1 [384000000] -> a53_pll0_main:1:1 [768000000] -> a53_pll0:1:1 [1536000000,
        a53_clk:1:1 [384000000, 3] -> a53_hf_mux:1:1 [384000000] -> a53_lf_mux:1:1 [384000000] -> a53_lf_
        cci_hf_mux:1:1 [499200000] -> cci_pll_main:1:1 [499200000] -> cci_pll:1:1 [998400000, 2] -> xo_ao
        cci_clk:1:1 [249600000, 3] -> cci_hf_mux:1:1 [499200000] -> cci_pll_main:1:1 [499200000] -> cci_p
        cci_pll:1:1 [998400000, 2] -> xo_ao:4:4 [19200000] -> cxo_clk_src_ao:1:1 [19200000]
Enabled clock count: 35
```

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 sleep without any issues. These clocks/PLLs are turned off later by RPMh 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.

5.1.5.3 Check interrupts activity that can prevent CX power collapse/AOSS sleep

Higher frequency of interrupts can prevent the system from entering CX power collapse/AOSS sleep.

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.

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

Checking the active_state of resources can provide information about the state of the resource.

1. Check the MPSS NPA log for information about various NPA requests made by the MPSS.
2. In the NPA log, search for the term npa_dump to check which subsystem is holding sleep. This is the starting point for resource states for the system at the point of log collection

3. 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. In this 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.

```
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)
```

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

   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.

5.1.5.5 Determine why the LPASS is not voting for power collapse

Checking the active_state of resources can provide information about the state of the resource.

1. Check the LPASS NPA log for information about various NPA requests made by the MPSS.
2. In the NPA log, search for the term npa_dump to check which subsystem is holding sleep. This is the starting point for resource states for the system at the point of log collection.
3. 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.

5.1.6 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

One method to identify the wakeup source includes monitoring the subsystem and digital rails. Most chipsets have a dedicated power rail for each subsystem. All subsystems are expected to be in sleep during AOSS sleep, and the subsystem waking up can be easily identified.

Monitoring rails for external components and sensors can also be helpful in cases 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.

5.1.6.1 Check modem wakeups

The MPSS is not expected to wake up during this use case because the rockbottom use case is performed while the device is 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 this example 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 Professional logs can help determine what activity is occurring in the MPSS.

5.1.6.2 Check APSS wakeups

Kernel logs with the appropriate logging enabled can indicate if the APSS is waking up from its sleep state and, if so, for what reason.

1. Enable the following debug mask to log the interrupt information in the kernel logs:

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

2. 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
```

5.1.7 Check leakage in sleep current

If the base current is higher than expected, even after the device successfully enters AOSS sleep, there are one or more leakage sources on the device contributing to the total current consumption.

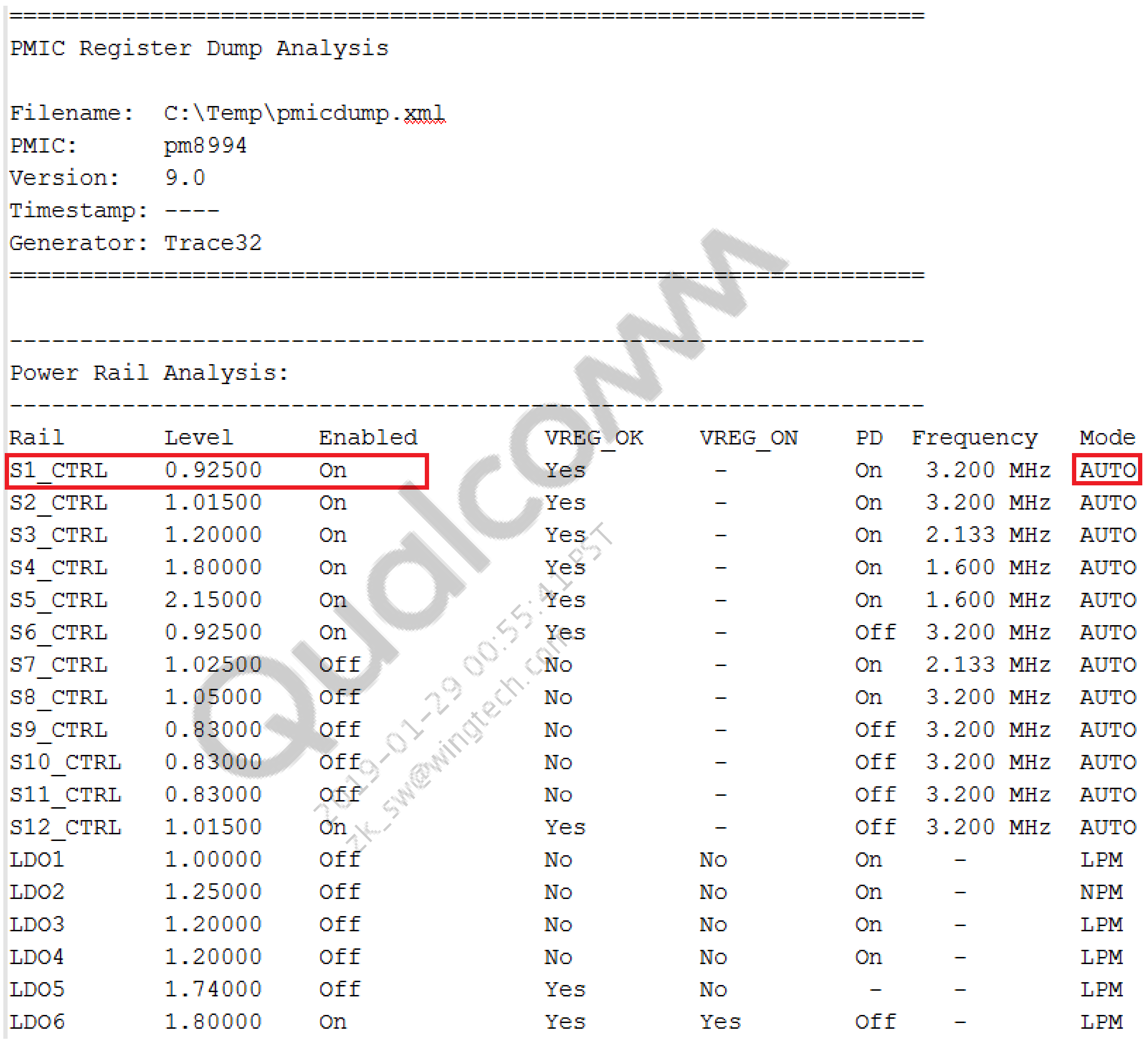
Leakage can occur from multiple sources, such as the following:

- 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

5.1.7.1 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 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.



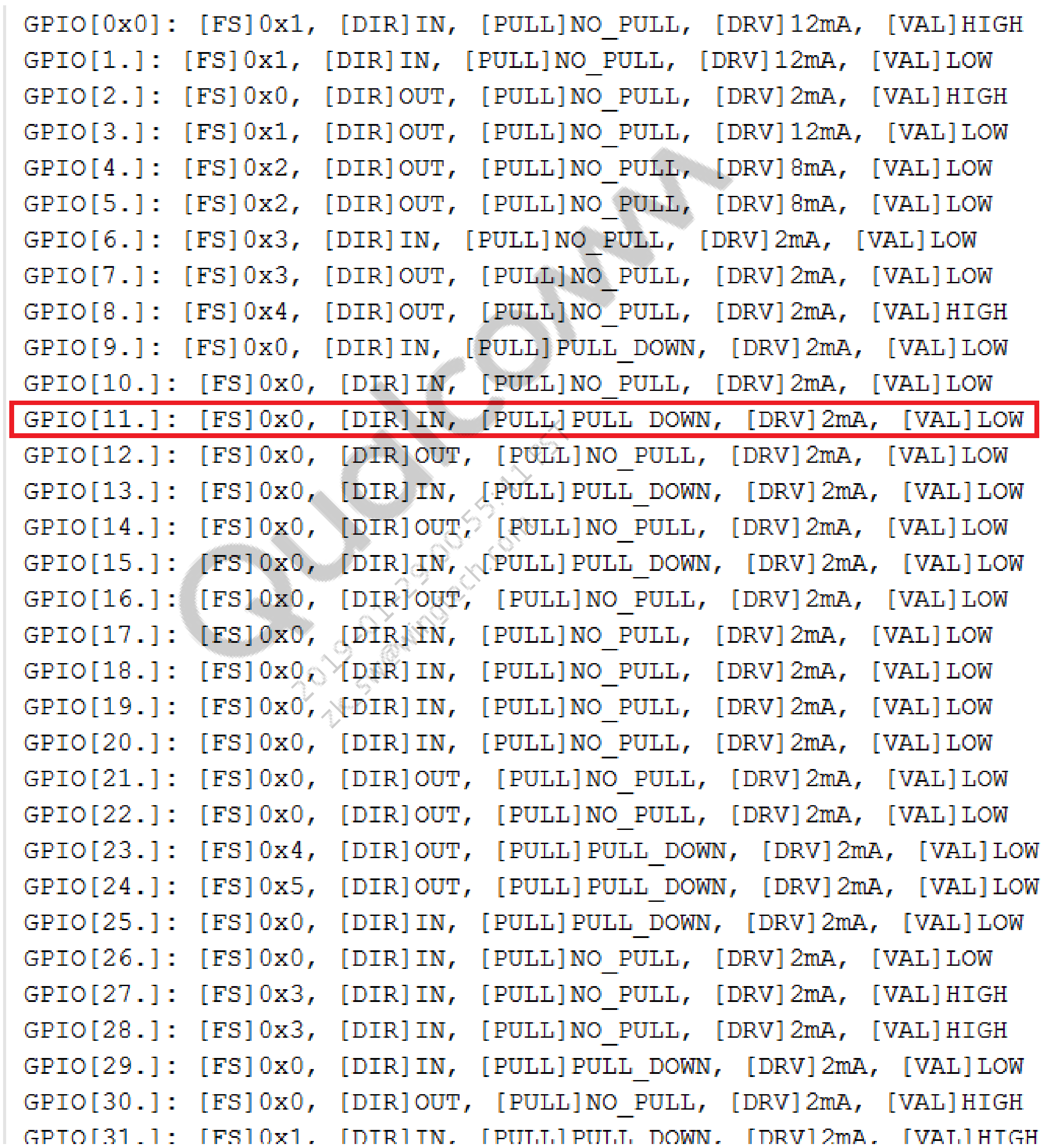
```
=================================================================
PMIC Register Dump Analysis

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
```

5.1.7.2 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 a 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.

5.1.7.3 Analyze rail level current consumption breakdown

Breakdown of current consumption for individual power rails can help narrow which component of the system is consuming higher current consumption.

1. Collect current consumption of all PMIC rails on PMIC input, for example, SMPS current, any LDOs that are not sourced by SMPS, etc.
2. Collect current consumption of components directly powered by VPH (components not powered by PMIC).
3. 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.

5.2 Standby

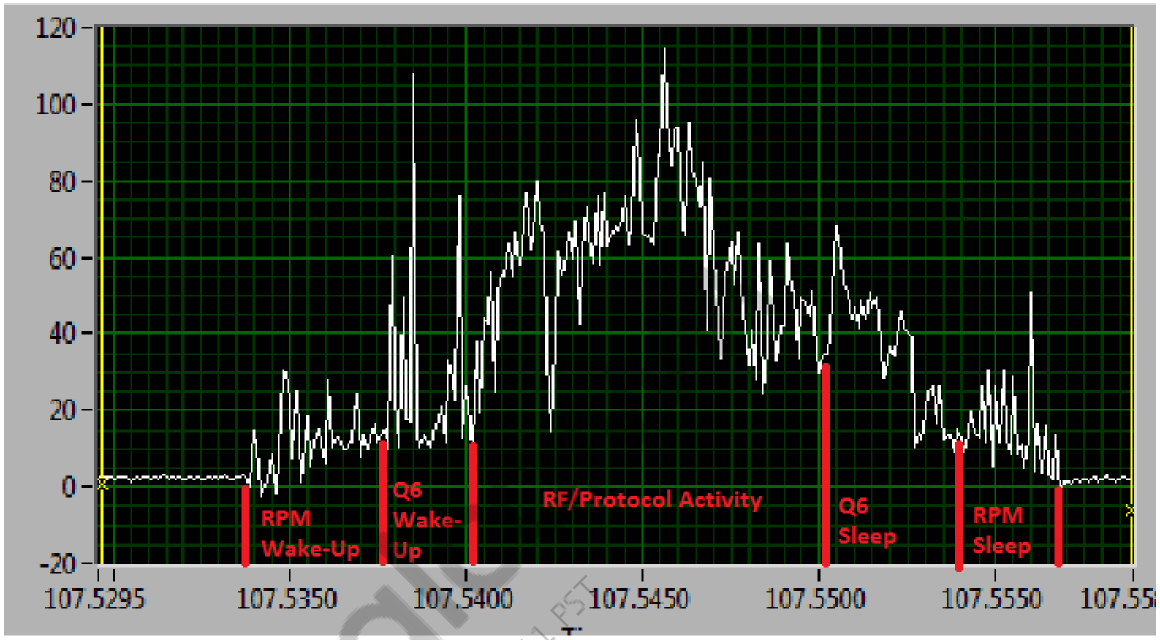
Total average standby current is highly dependent on the average rockbottom current consumption. If not already done, the first step for standby optimization is to optimize the rockbottom current consumption. Follow the steps in Rockbottom to optimize rockbottom current consumption.

5.2.1 Analyze higher paging awake penalty

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

1. 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:



2. 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
3. Collect F3 logs, which are helpful in debugging the awake period during paging, by setting the appropriate configuration for log messages. For example, if debugging WCDMA awake penalty, use the following method to collect logs:
   a. Open QXDM Professional 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 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 narrow down the issue.
4. 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.

5.3 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.
- 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.

RELATED INFORMATION

“Check APSS” on page 41

5.3.1 Inspect clocks and shared resources

Clocks running at higher frequency than expected can cause the system to consume more power. Take a clock dump and compare it with the QTI reference device clock dump to ensure that clocks are at expected state.

1. Break the system at any point through RPM and run the clock dump script.
2. Verify that no extra clocks are running other than the expected clocks.
3. Verify that all clocks 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)
4. 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. In this example, 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.

```
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)
```

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

5.3.2 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.

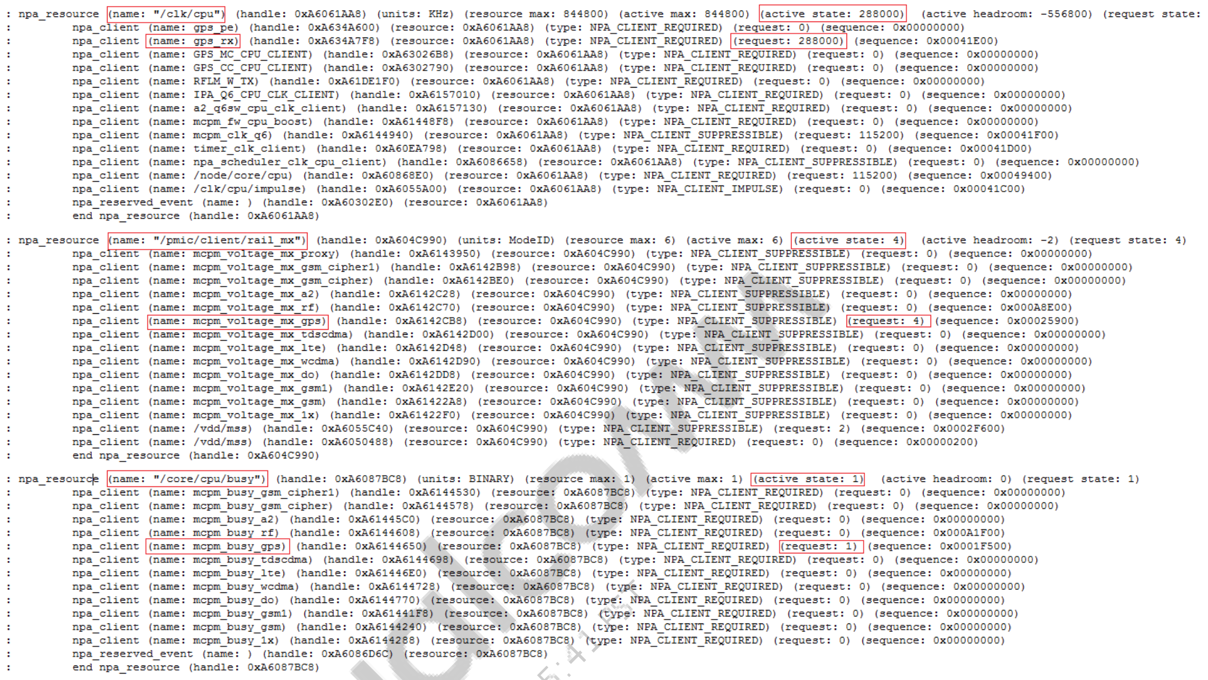
5.3.3 Check modem resource votings

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, and modem clock and power manager (MCPM) are some of the many resources whose voting can be seen from the modem NPA logs.

The following figure shows 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 clock/CPU and CPU/busy resources). In this example, the GPS client is voting for all the abovementioned 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)
```

5.4 Data

The data use case is complex. 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 helpful in debugging the data use case because it can be quickly identified if the MPSS or APSS is consuming current higher than expected. Subsystem-level debugging can then be done accordingly.

For RF/PA rails consuming higher current, see Check PA/RF power consumption.

If rail-level breakdowns are not readily available, resource voting by a different subsystem can be checked from the ARC/BCM/VRM logs; see Inspect clocks and shared resources.

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 Check modem resource votings.

5.4.1 Check APSS

The APSS can be one of the major reasons for 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.

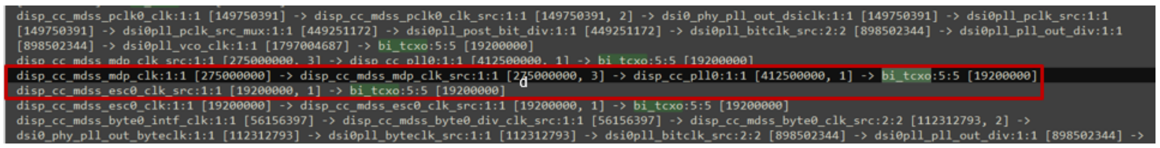
  - These tests provide a rough estimate of why the APSS consumes higher current.
  - These logs can be compared with the QTI reference device logs for a comparative analysis of which system components are supposed to be up.
- Analyze Top logs and compare them against the QTI reference device as follows:
  - 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.

5.5 Debug CX power collapse for static image with smart panel

1. Use the following debug mask to 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 tcxo. For example, in the following snippet, mdp clock is holding tcxo and blocking CX power collapse:



```
disp_cc_mdss_pclk0_clk:1:1 [149750391] -> disp_cc_mdss_pclk0_clk_src:1:1 [149750391, 2] -> dsi0_phy_pll_out_dsiclk:1:1 [149750391] -> dsi0pll_pclk_src:1:1
[149750391] -> dsi0pll_pclk_src_mux:1:1 [449251172] -> dsi0pll_post_bit_div:1:1 [449251172] -> dsi0pll_bitclk_src:2:2 [898502344] -> dsi0pll_pll_out_div:1:1
[898502344] -> dsi0pll_vco_clk:1:1 [1797004687] -> bi_tcxo:5:5 [19200000]
disp_cc_mdss_mdp_clk_src:1:1 [275000000, 3] -> disp_cc_pll0:1:1 [412500000, 1] -> bi_tcxo:5:5 [19200000]
disp_cc_mdss_mdp_clk:1:1 [275000000] -> disp_cc_mdss_mdp_clk_src:1:1 [275000000, 3] -> disp_cc_pll0:1:1 [412500000, 1] -> bi_tcxo:5:5 [19200000]
disp_cc_mdss_esc0_clk_src:1:1 [19200000, 1] -> bi_tcxo:5:5 [19200000]
disp_cc_mdss_esc0_clk:1:1 [19200000] -> disp_cc_mdss_esc0_clk_src:1:1 [19200000, 1] -> bi_tcxo:5:5 [19200000]
disp_cc_mdss_byte0_intf_clk:1:1 [56156397] -> disp_cc_mdss_byte0_div_clk_src:1:1 [56156397] -> disp_cc_mdss_byte0_clk_src:2:2 [112312793, 2] ->
dsi0_phy_pll_out_byteclk:1:1 [112312793] -> dsi0pll_byteclk_src:1:1 [112312793] -> dsi0pll_bitclk_src:2:2 [898502344] -> dsi0pll_pll_out_div:1:1 [898502344] ->
```

   There might be other clocks requesting bi_tcxo_ao. These are the “Active Only” requests by these clients and should be ignored. Only look for bi_txco.

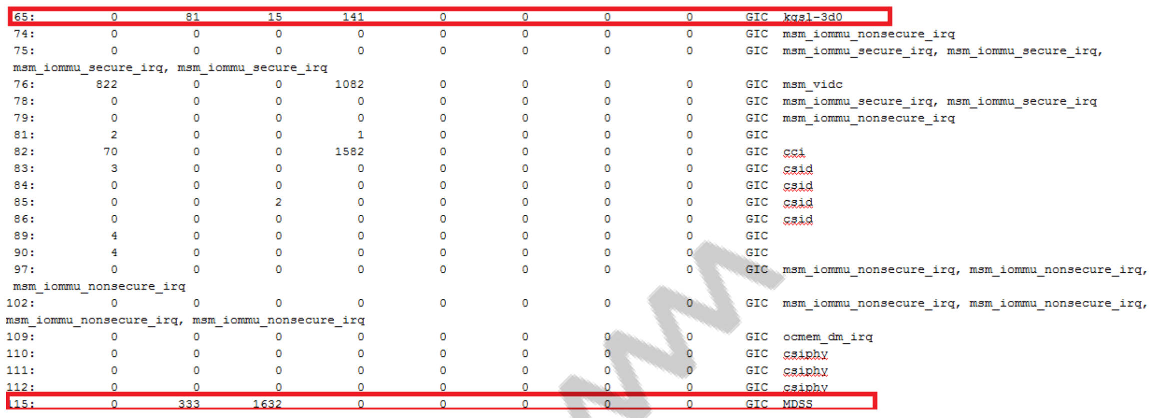
3. Look for any hwirqs that occur, as shown in the following example:

```
<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 SDM845, the .dtsi file is found in `/kernel/arch/arm64/boot/dts/qcom/sdm845-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) */
```

5.6 Debug SurfaceFlinger

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.

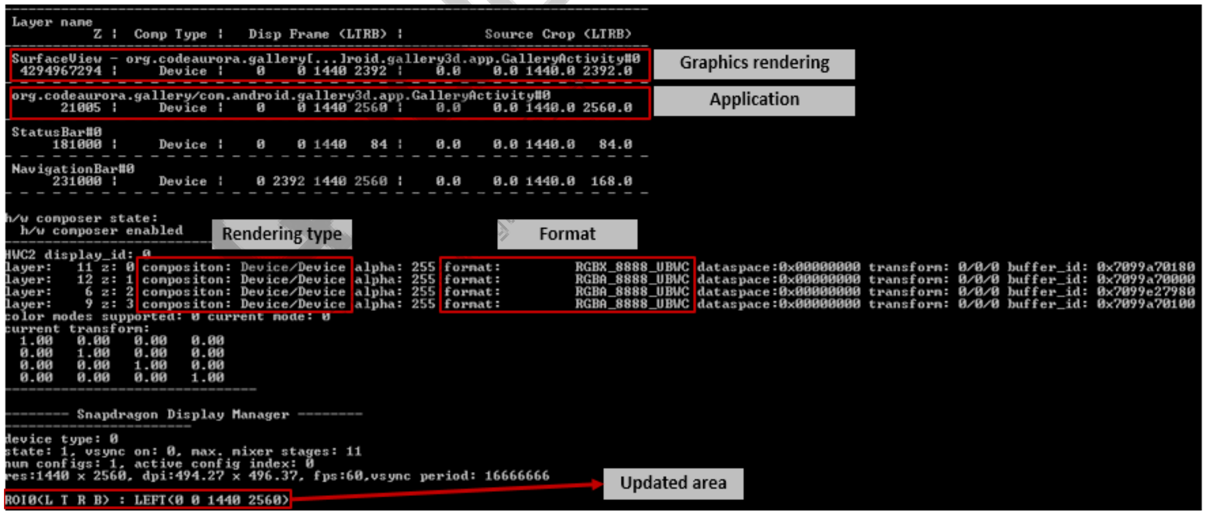
Common layers observed are as follows:

- Status bar
- Navigation bar
- Application

To collect SurfaceFlinger logs, type the following commands:

```
adb shell
dumpsys SurfaceFlinger
```

The following shows a static image SurfaceFlinger:



```
Layer name
           Z |  Comp Type |  Disp Frame (LTRB) |         Source Crop (LTRB)
SurfaceView - org.codeaurora.gallery[...]roid.gallery3d.app.GalleryActivity#0
 4294967294 |      Device |    0    0 1440 2392 |     0.0     0.0 1440.0 2392.0
org.codeaurora.gallery/com.android.gallery3d.app.GalleryActivity#0
      21005 |      Device |    0    0 1440 2560 |     0.0     0.0 1440.0 2560.0
StatusBar#0
     181000 |      Device |    0    0 1440   84 |     0.0     0.0 1440.0   84.0
NavigationBar#0
     231000 |      Device |    0 2392 1440 2560 |     0.0     0.0 1440.0  168.0

h/w composer state:
 h/w composer enabled
HWC2 display_id: 0
layer:    11 z: 0 compositon: Device/Device alpha: 255 format:        RGBX_8888_UBWC dataspace:0x00000000 transform: 0/0/0 buffer_id: 0x7099a70180
layer:    12 z: 1 compositon: Device/Device alpha: 255 format:        RGBA_8888_UBWC dataspace:0x00000000 transform: 0/0/0 buffer_id: 0x7099a70000
layer:     6 z: 2 compositon: Device/Device alpha: 255 format:        RGBA_8888_UBWC dataspace:0x00000000 transform: 0/0/0 buffer_id: 0x7099e27980
layer:     9 z: 3 compositon: Device/Device alpha: 255 format:        RGBA_8888_UBWC dataspace:0x00000000 transform: 0/0/0 buffer_id: 0x7099a70100
color modes supported: 0 current mode: 0
current transform:
  1.00   0.00   0.00   0.00
  0.00   1.00   0.00   0.00
  0.00   0.00   1.00   0.00
  0.00   0.00   0.00   1.00

-------- Snapdragon Display Manager --------

device type: 0
state: 1, vsync on: 0, max. mixer stages: 11
num configs: 1, active config index: 0
res:1440 x 2560, dpi:494.27 x 496.37, fps:60,vsync period: 16666666
ROI0(L T R B) : LEFT(0 0 1440 2560)
```

In the above examples, the following are shown:

- Application
- Rendering type
- Composition types:
  - MDP composition
  - GPU composition causes higher power
- Rendering format – RGB, YUV, etc.

5.7 Tunnel mode and player type check

The following describes methods to analyze Tunnel mode and player type.

5.7.1 Check Tunnel mode

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:

   ```
   audio_hw_primary: out_set_parameters: enter: usecase(3: compress-offload-
   playback) kvpairs: routing=0
   audio_hw_primary: out_set_parameters: enter: usecase(3: compress-offload-
   playback) kvpairs: closing=true
   audio_hw_primary: out_set_parameters: enter: usecase(3: compress-offload-
   playback) kvpairs: exiting=1
   ```

   Or

   ```
   audio_hw_primary: out_set_parameters: enter: usecase(4: compress-offload-
   playback2) kvpairs: routing=0
   audio_hw_primary: out_set_parameters: enter: usecase(4: compress-offload-
   playback2) kvpairs: closing=true
   audio_hw_primary: out_set_parameters: enter: usecase(4: compress-offload-
   playback2) kvpairs: exiting=1
   ```

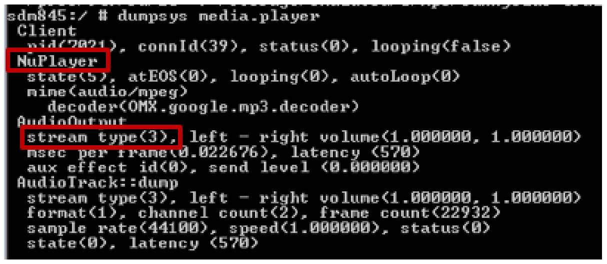
4. If the deep-buffer-playback message is found in the log, it indicates that nontunnel, i.e., normal, playback is ongoing.

5.7.2 Check player type

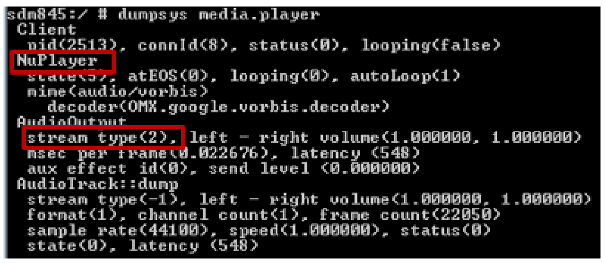
While running the use case, check the type of player and off-load playback by typing the following commands:

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

Examples



```
sdm845:/ # dumpsys media.player
 Client
  pid(7021), connId(39), status(0), looping(false)
 NuPlayer
  state(5), atEOS(0), looping(0), autoLoop(0)
  mime(audio/mpeg)
    decoder(OMX.google.mp3.decoder)
 AudioOutput
  stream type(3), left - right volume(1.000000, 1.000000)
  msec per frame(0.022676), latency (570)
  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(22932)
  sample rate(44100), speed(1.000000), status(0)
  state(0), latency (570)
```



```
sdm845:/ # dumpsys media.player
 Client
  pid(2513), connId(8), status(0), looping(false)
 NuPlayer
  state(5), atEOS(0), looping(0), autoLoop(1)
  mime(audio/vorbis)
    decoder(OMX.google.vorbis.decoder)
 AudioOutput
  stream type(2), left - right volume(1.000000, 1.000000)
  msec per frame(0.022676), latency (548)
  aux effect id(0), send level (0.000000)
 AudioTrack::dump
  stream type(-1), left - right volume(1.000000, 1.000000)
  format(1), channel count(1), frame count(22050)
  sample rate(44100), speed(1.000000), status(0)
  state(0), latency (548)
```

Stream type(3) means offload playback and stream type(2) means non-offload playback.

5.8 Music application wake lock check and debugging

The following describes how to check and debug music application wake lock.

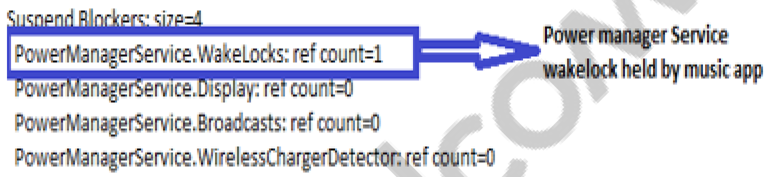
RELATED INFORMATION

“Collect wake locks” on page 82

“Determine why the APSS is not voting for power collapse” on page 30

5.8.1 Analyze wake lock

1. Ensure that the power manager service is holding a wake lock using the following check, which shows expected dumpsys power output:



```
Suspend Blockers: size=4
 PowerManagerService.WakeLocks: ref count=1
 PowerManagerService.Display: ref count=0
 PowerManagerService.Broadcasts: ref count=0
 PowerManagerService.WirelessChargerDetector: ref count=0
```

2. If the above wake lock is not observed, hold a test wake lock to determine if power consumption reduces. Type the following commands:

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

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

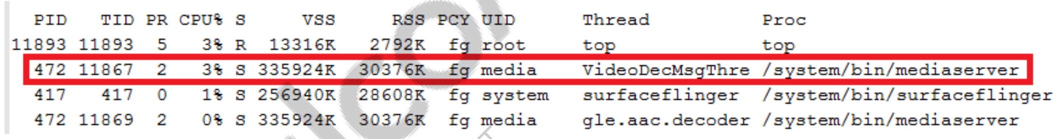
```
adb shell
echo test > sys/power/wake_unlock
```

3. If this check resolves the issue, modify the music app to hold a wake lock during MP3 playback with display turned off.

5.9 Analyze high CPU usage process

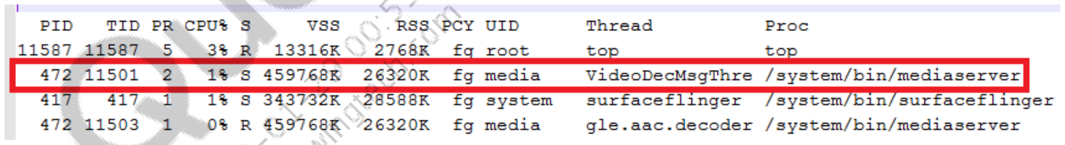
1. Capture the Top or ftrace data as described in Capture PowerTop and Top data.
2. Determine which process is consuming the most CPU and look for any processes that should not be running.
   a. If there is an unexpected process or thread, discuss it with the module engineer in charge of process or thread.
   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 (PID) 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
```

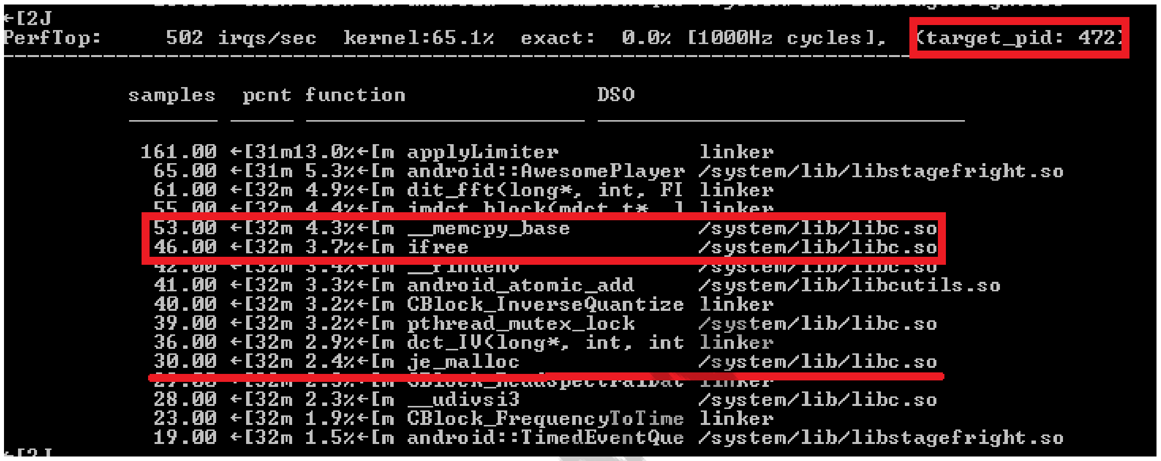
      ii. Debug in detail with the PerfTop tool (see Debugging tools for apps processor use cases).

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

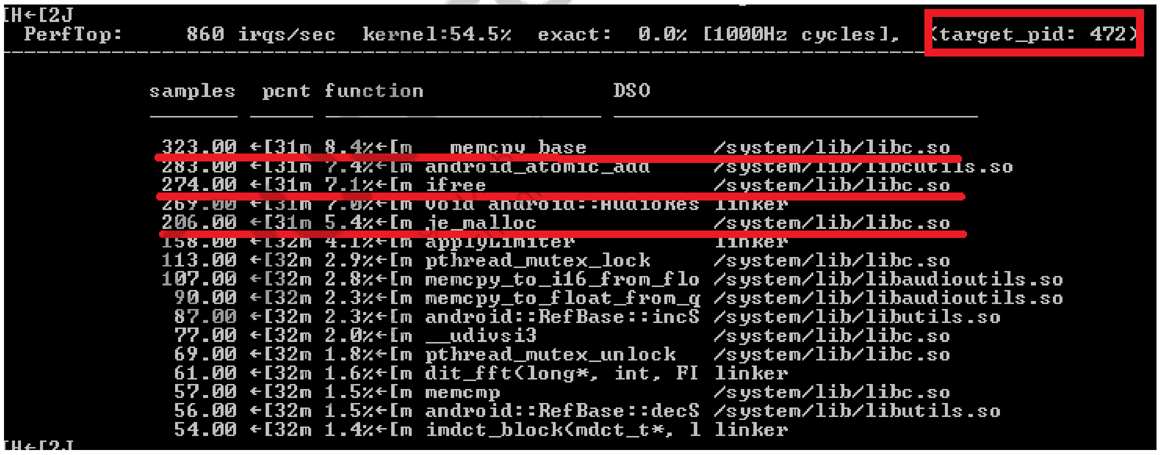


```
←[2J
PerfTop:      502 irqs/sec  kernel:65.1%  exact:  0.0% [1000Hz cycles],  (target_pid: 472)
------------------------------------------------------------------------------------------

          samples  pcnt function                   DSO
          _______ _____ ___________________________ _____________________________

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

PerfTop data B



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

           samples  pcnt function                   DSO
           _______ _____ ___________________________ _____________________________

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

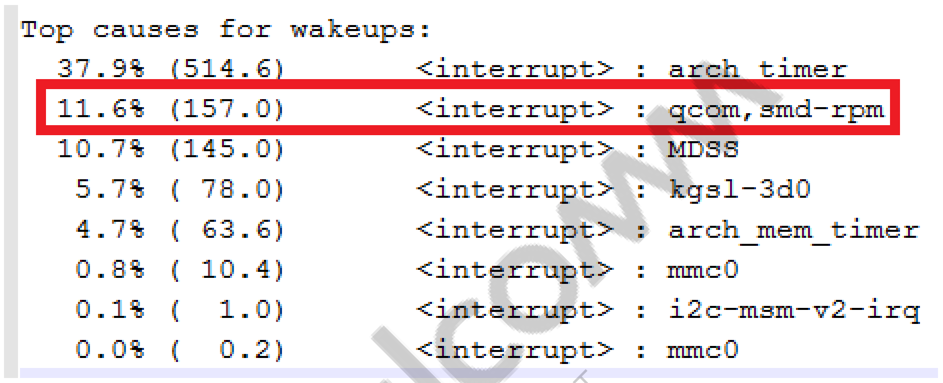
As the PerfTop data shows, the memory handling (memcpy, malloc, free) function in PID 472 is more frequently called in PerfTop data A.

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

5.10 Analyze interrupt

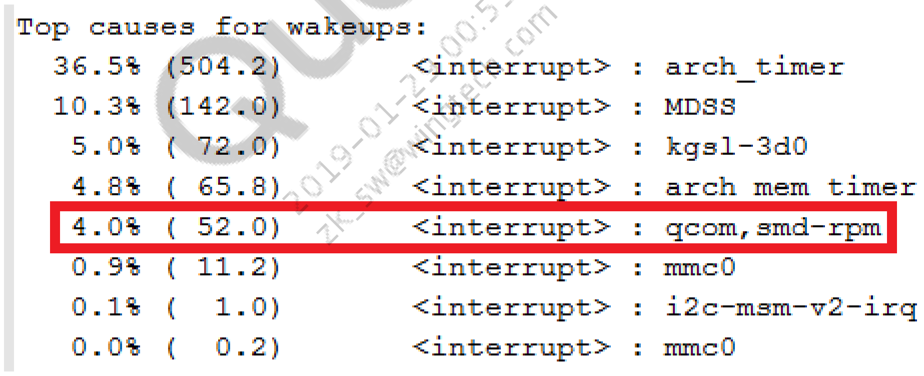
1. Capture the PowerTop data.
2. If the data is similar to the following examples, capture an ftrace log to debug in detail for qcom,smd-rpm.

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

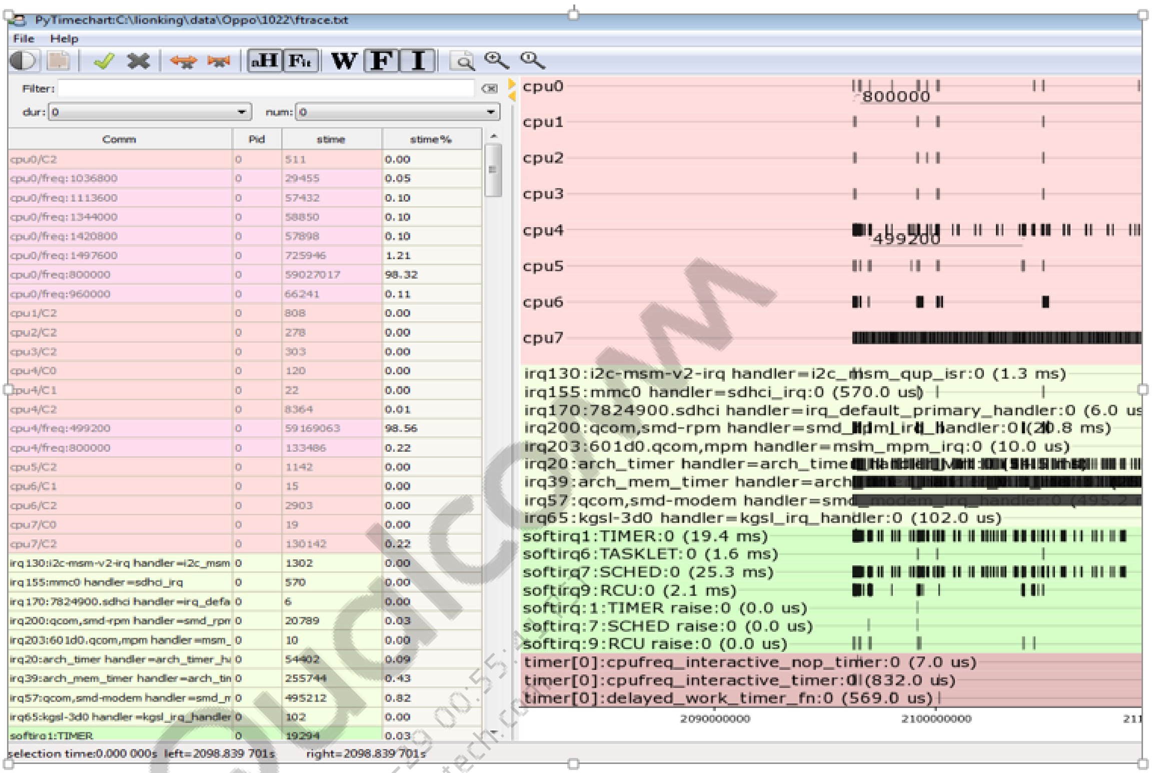


```
Top causes for wakeups:
  36.5% (504.2)        <interrupt> : arch_timer
  10.3% (142.0)        <interrupt> : MDSS
   5.0% ( 72.0)        <interrupt> : kgsl-3d0
   4.8% ( 65.8)        <interrupt> : arch_mem_timer
   4.0% ( 52.0)        <interrupt> : qcom,smd-rpm
   0.9% ( 11.2)        <interrupt>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

3. Launch and use the pytime chart to debug interrupt.

4. Open the ftrace log using the menu in the pytime chart.

   The following screen appears:

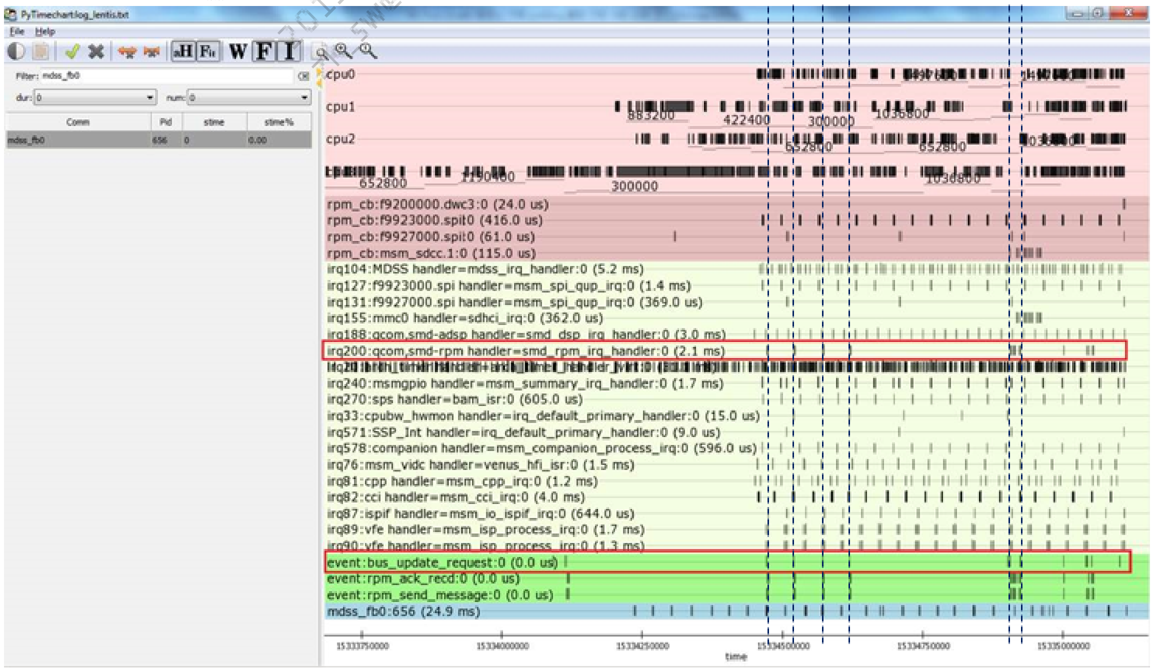


Comm	Pid	stime	stime%
cpu0/C2	0	511	0.00
cpu0/freq:1036800	0	29455	0.05
cpu0/freq:1113600	0	57432	0.10
cpu0/freq:1344000	0	58850	0.10
cpu0/freq:1420800	0	57898	0.10
cpu0/freq:1497600	0	725946	1.21
cpu0/freq:800000	0	59027017	98.32
cpu0/freq:960000	0	66241	0.11
cpu1/C2	0	808	0.00
cpu2/C2	0	278	0.00
cpu3/C2	0	303	0.00
cpu4/C0	0	120	0.00
cpu4/C1	0	22	0.00
cpu4/C2	0	8364	0.01
cpu4/freq:499200	0	59169063	98.56
cpu4/freq:800000	0	133486	0.22
cpu5/C2	0	1142	0.00
cpu6/C1	0	15	0.00
cpu6/C2	0	2903	0.00
cpu7/C0	0	19	0.00
cpu7/C2	0	130142	0.22
irq130:i2c-msm-v2-irq handler=i2c_msm	0	1302	0.00
irq155:mmc0 handler=sdhci_irq	0	570	0.00
irq170:7824900.sdhci handler=irq_defa	0	6	0.00
irq200:qcom,smd-rpm handler=smd_rpm	0	20789	0.03
irq203:601d0.qcom,mpm handler=msm_	0	10	0.00
irq20:arch_timer handler=arch_timer_h	0	54402	0.09
irq39:arch_mem_timer handler=arch_tin	0	255744	0.43
irq57:qcom,smd-modem handler=smd_r	0	495212	0.82
irq65:kgsl-3d0 handler=kgsl_irq_handler	0	102	0.00
softirq1:TIMER	0	19294	0.03

```
irq130:i2c-msm-v2-irq handler=i2c_msm_qup_isr:0 (1.3 ms)
irq155:mmc0 handler=sdhci_irq:0 (570.0 us)
irq170:7824900.sdhci handler=irq_default_primary_handler:0 (6.0 us
irq200:qcom,smd-rpm handler=smd_rpm_irq_handler:0 (20.8 ms)
irq203:601d0.qcom,mpm handler=msm_mpm_irq:0 (10.0 us)
irq20:arch_timer handler=arch_timer_handler_virt:0
irq39:arch_mem_timer handler=arch
irq57:qcom,smd-modem handler=smd_modem_irq_handler:0 (495.2
irq65:kgsl-3d0 handler=kgsl_irq_handler:0 (102.0 us)
softirq1:TIMER:0 (19.4 ms)
softirq6:TASKLET:0 (1.6 ms)
softirq7:SCHED:0 (25.3 ms)
softirq9:RCU:0 (2.1 ms)
softirq:1:TIMER raise:0 (0.0 us)
softirq:7:SCHED raise:0 (0.0 us)
softirq:9:RCU raise:0 (0.0 us)
timer[0]:cpufreq_interactive_nop_timer:0 (7.0 us)
timer[0]:cpufreq_interactive_timer:0 (832.0 us)
timer[0]:delayed_work_timer_fn:0 (569.0 us)
```

selection time:0.000 000s left=2098.839 701s right=2098.839 701s

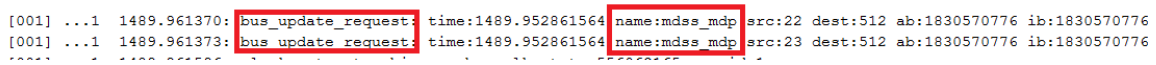
5. Determine which module caused the qcom,smd-rpm interrupt by alignment as shown.



Comm	Pid	stime	stime%
mdss_fb0	656	0	0.00

```
rpm_cb:f9200000.dwc3:0 (24.0 us)
rpm_cb:f9923000.spi:0 (416.0 us)
rpm_cb:f9927000.spi:0 (61.0 us)
rpm_cb:msm_sdcc.1:0 (115.0 us)
irq104:MDSS handler=mdss_irq_handler:0 (5.2 ms)
irq127:f9923000.spi handler=msm_spi_qup_irq:0 (1.4 ms)
irq131:f9927000.spi handler=msm_spi_qup_irq:0 (369.0 us)
irq155:mmc0 handler=sdhci_irq:0 (362.0 us)
irq188:qcom,smd-adsp handler=smd_dsp_irq_handler:0 (3.0 ms)
irq200:qcom,smd-rpm handler=smd_rpm_irq_handler:0 (2.1 ms)
irq20:arch_timer handler=arch_timer_handler_virt:0
irq240:msmgpio handler=msm_summary_irq_handler:0 (1.7 ms)
irq270:sps handler=bam_isr:0 (605.0 us)
irq33:cpubw_hwmon handler=irq_default_primary_handler:0 (15.0 us)
irq571:SSP_Int handler=irq_default_primary_handler:0 (9.0 us)
irq578:companion handler=msm_companion_process_irq:0 (596.0 us)
irq76:msm_vidc handler=venus_hfi_isr:0 (1.5 ms)
irq81:cpp handler=msm_cpp_irq:0 (1.2 ms)
irq82:cci handler=msm_cci_irq:0 (4.0 ms)
irq87:ispif handler=msm_io_ispif_irq:0 (644.0 us)
irq89:vfe handler=msm_isp_process_irq:0 (1.7 ms)
irq90:vfe handler=msm_isp_process_irq:0 (1.3 ms)
event:bus_update_request:0 (0.0 us)
event:rpm_ack_recd:0 (0.0 us)
event:rpm_send_message:0 (0.0 us)
mdss_fb0:656 (24.9 ms)
```

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.

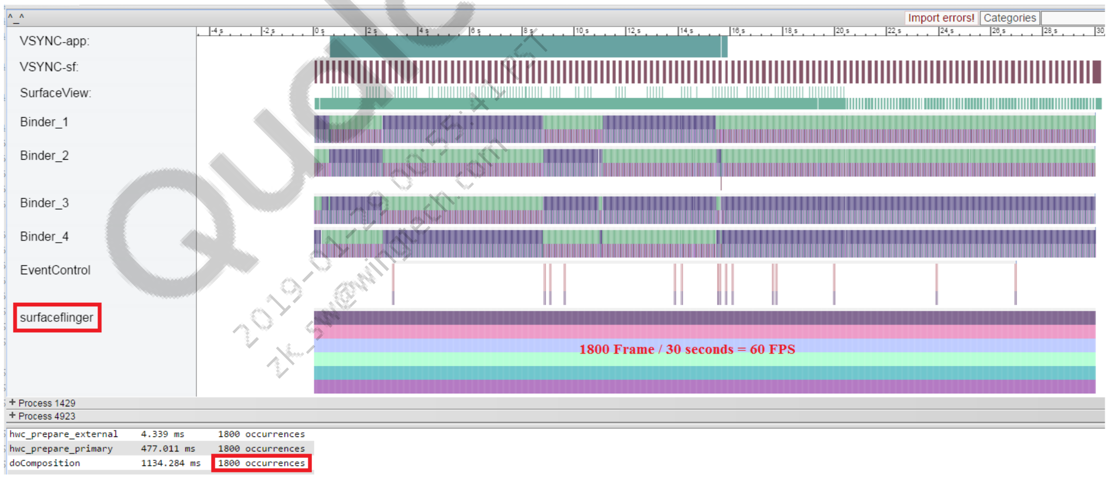
RELATED INFORMATION

"Debugging tools for apps processor use cases" on page 8

5.11 Debug high GPU clock frequency

For this procedure, a Chrome browser is required.

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



```
hwc_prepare_external    4.339 ms     1800 occurrences
hwc_prepare_primary     477.011 ms   1800 occurrences
doComposition           1134.284 ms  1800 occurrences
```

2. Verify 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.

5.11.1 Use 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
```

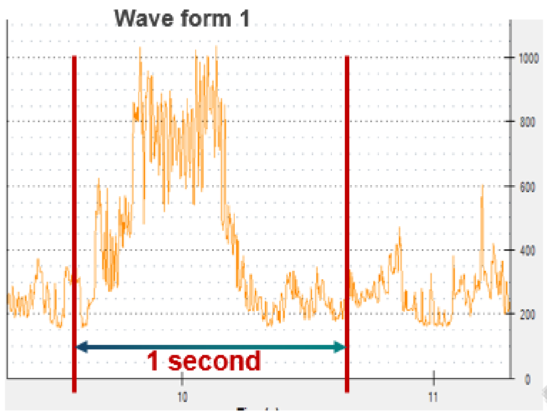
5.12 Analyze waveforms

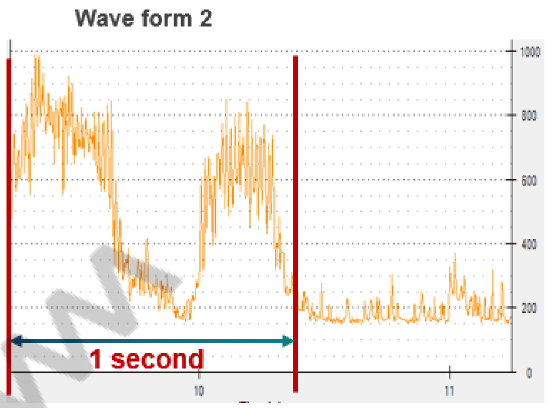
For a basic understanding of waveform analysis, read the following steps in conjunction with the images and systrace logs that follow.

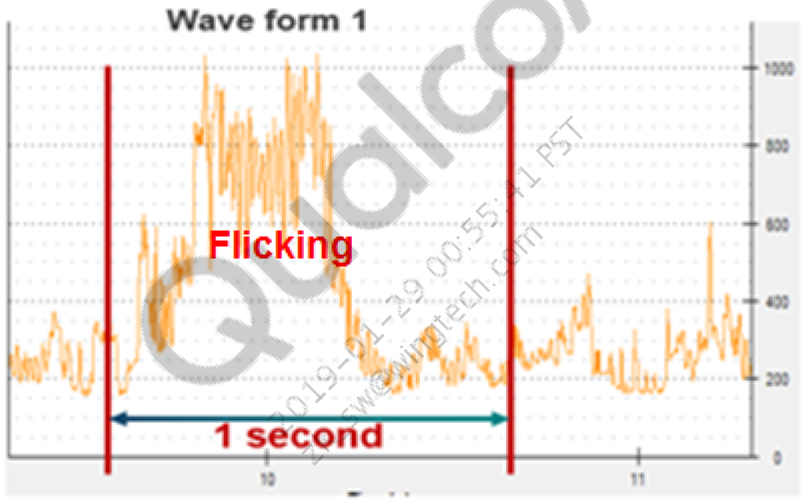
1. Observe that waveform 1 and waveform 2 have different behavior, i.e., there are two waves per one flicking wave from waveform 2 compared to waveform 1.
2. To determine the problem, use the systrace tool to debug in detail. A Chrome browser is required.
3. Capture a systrace log and open it with a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

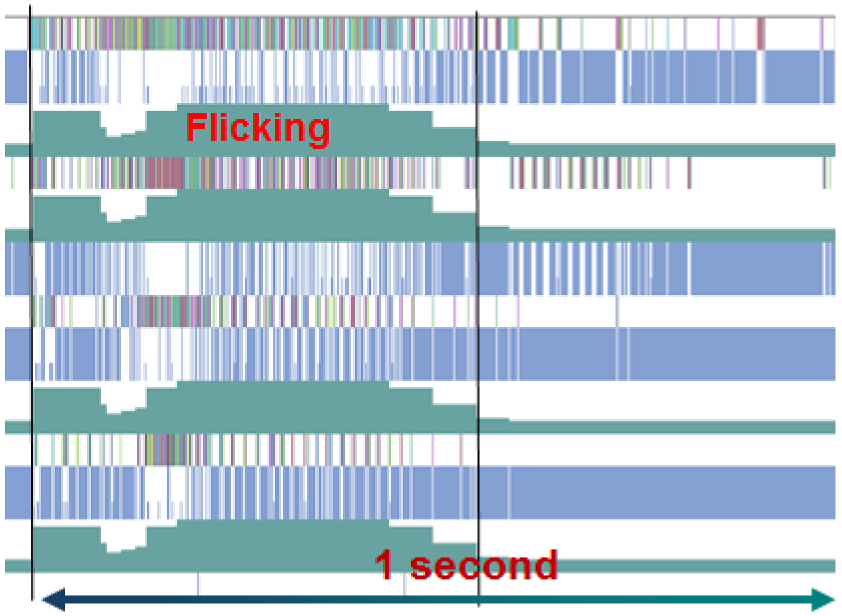
   As a result of the analysis, the second wave in waveform 2 is caused by render thread and composition thread.

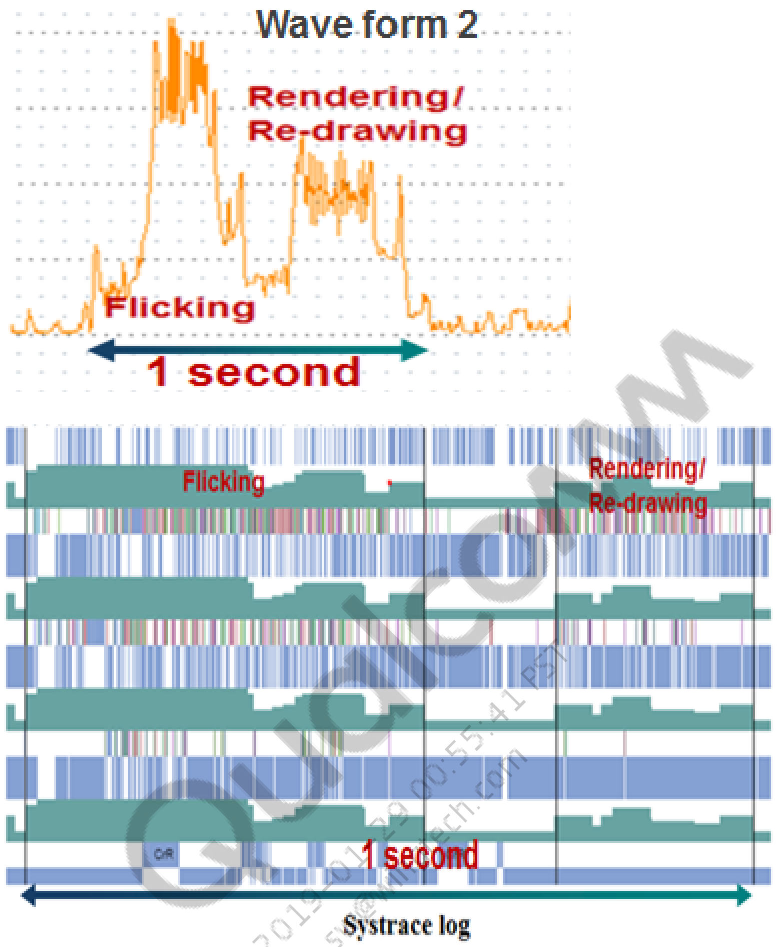
8. In this case, consult with the graphic engineer and display engineer as to whether this is the correct behavior.



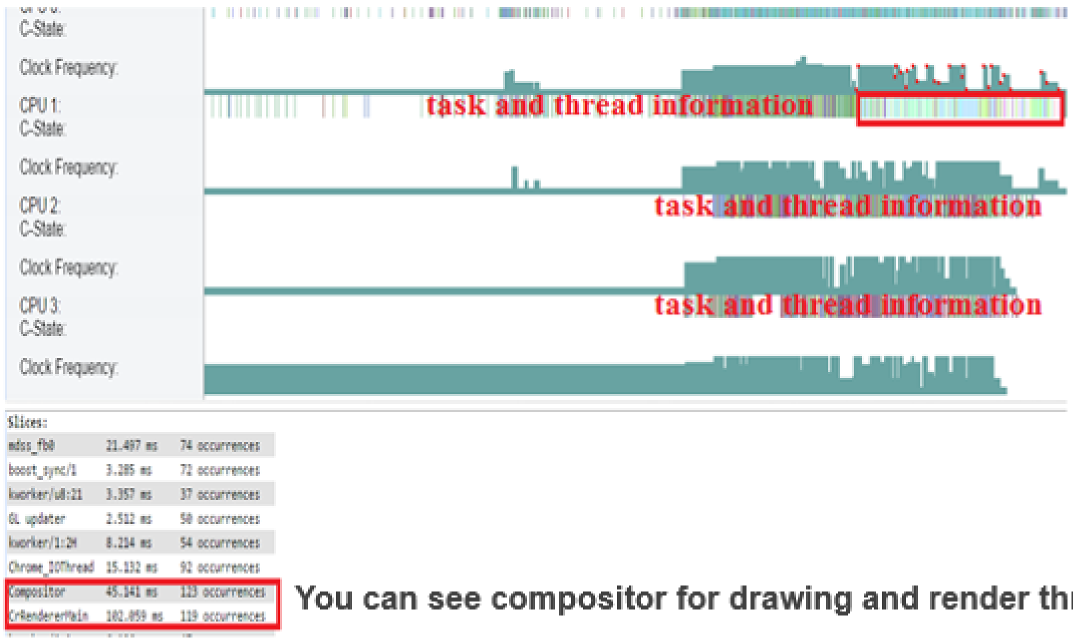








systrace log 2



```
Slices:
mdss_fb0          21.497 ms    74 occurrences
boost_sync/1      3.285 ms     72 occurrences
kworker/u8:21     3.357 ms     37 occurrences
GL updater        2.512 ms     50 occurrences
kworker/1:2H      8.214 ms     54 occurrences
Chrome_IOThread   15.132 ms    92 occurrences
Compositor        45.141 ms    123 occurrences
CrRendererMain    102.059 ms   119 occurrences
```

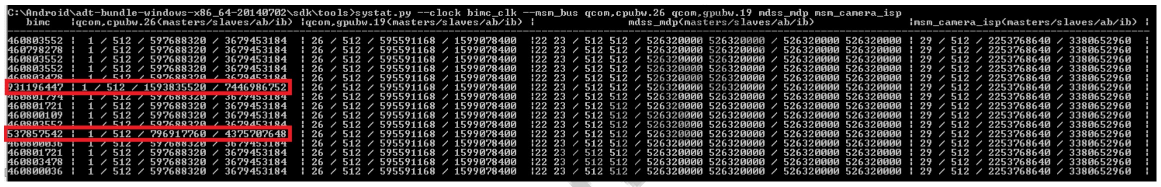
You can see compositor for drawing and render thread

5.13 Analyze DDR clock

1. Use the msmbusvoting tool to analyze the DDR clock.
2. Check the clock and bandwidth voting by each client in real time.

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

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

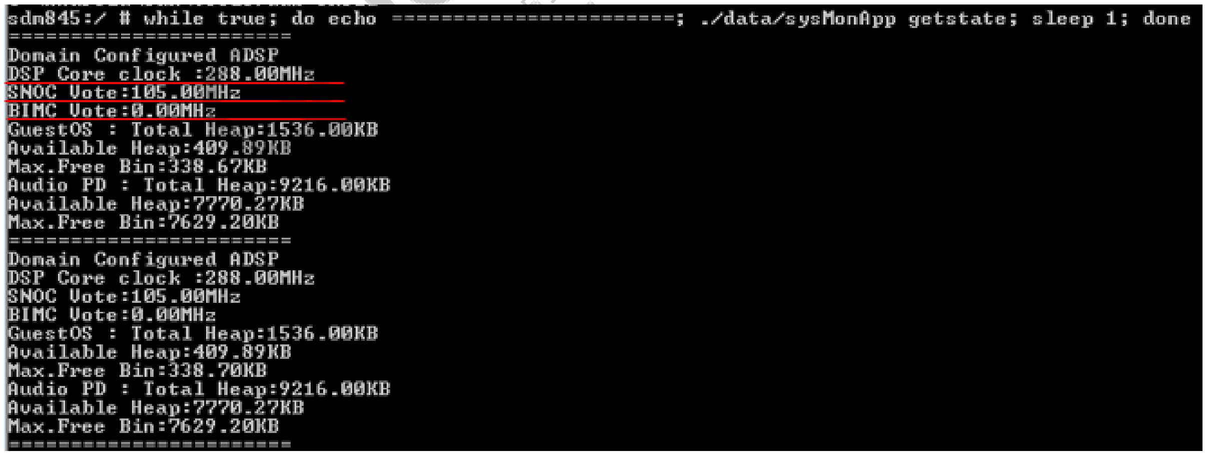


```
C:\Android\adt-bundle-windows-x86_64-20140702\sdk\tools>systat.py --clock bimc_clk --msm_bus qcom,cpubw.26 qcom,gpubw.19 mdss_mdp msm_camera_isp
   bimc    |qcom,cpubw.26(masters/slaves/ab/ib) |qcom,gpubw.19(masters/slaves/ab/ib) |               mdss_mdp(masters/slaves/ab/ib)                    |msm_camera_isp(masters/slaves/ab/ib) |
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
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  |
460801774 |  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  |
```

5.14 Check ADSP clock and voting

Use the sysmon tool to determine ADSP clock and voting:

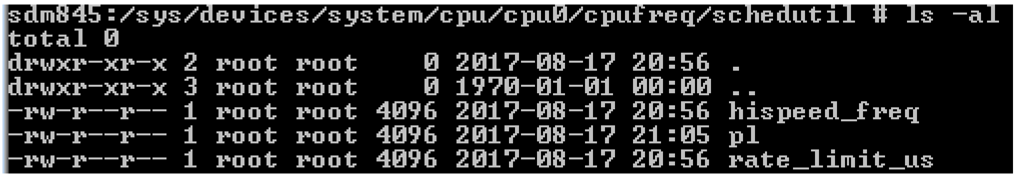
```
./SysMonApp  getstate
```



```
sdm845:/ # while true; do echo ======================; ./data/sysMonApp getstate; sleep 1; done
======================
Domain Configured ADSP
DSP Core clock :288.00MHz
SNOC Vote:105.00MHz
BIMC Vote:0.00MHz
GuestOS  : Total Heap:1536.00KB
Available Heap:409.89KB
Max.Free Bin:338.67KB
Audio PD  : Total Heap:9216.00KB
Available Heap:7770.27KB
Max.Free Bin:7629.20KB
======================
Domain Configured ADSP
DSP Core clock :288.00MHz
SNOC Vote:105.00MHz
BIMC Vote:0.00MHz
GuestOS  : Total Heap:1536.00KB
Available Heap:409.89KB
Max.Free Bin:338.70KB
Audio PD  : Total Heap:9216.00KB
Available Heap:7770.27KB
Max.Free Bin:7629.20KB
======================
```

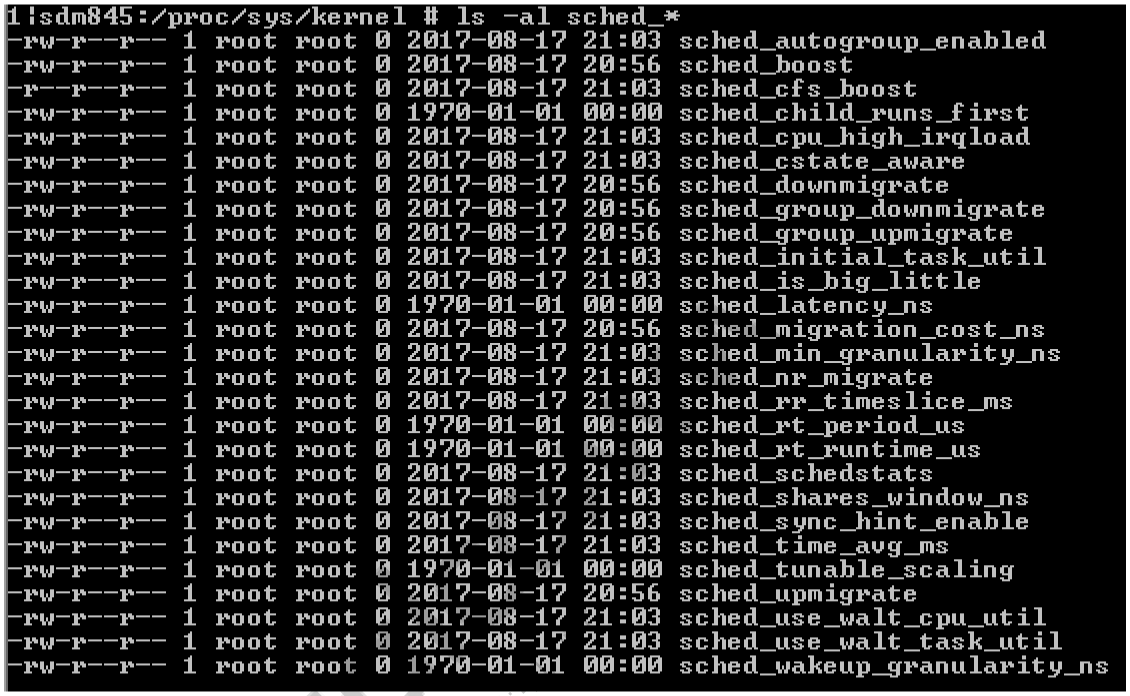
5.15 Check governor, scheduler, and CPU freq parameters

1. For governor parameters, check all nodes under the following paths:
   - Power (Silver cluster) – `/sys/devices/system/cpu/cpu0/cpufreq/schedutil/`
   - Perf (Gold cluster) – `/sys/devices/system/cpu/cpu4/cpufreq/schedutil/`



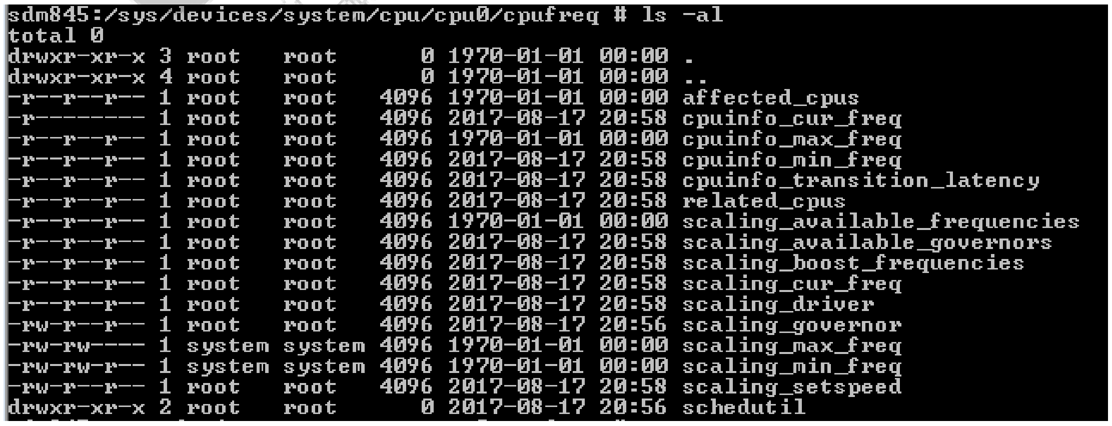
```
sdm845:/sys/devices/system/cpu/cpu0/cpufreq/schedutil # ls -al
total 0
drwxr-xr-x 2 root root    0 2017-08-17 20:56 .
drwxr-xr-x 3 root root    0 1970-01-01 00:00 ..
-rw-r--r-- 1 root root 4096 2017-08-17 20:56 hispeed_freq
-rw-r--r-- 1 root root 4096 2017-08-17 21:05 pl
-rw-r--r-- 1 root root 4096 2017-08-17 20:56 rate_limit_us
```

2. For scheduler parameters, check all nodes under the `/proc/sys/kernel/` path.



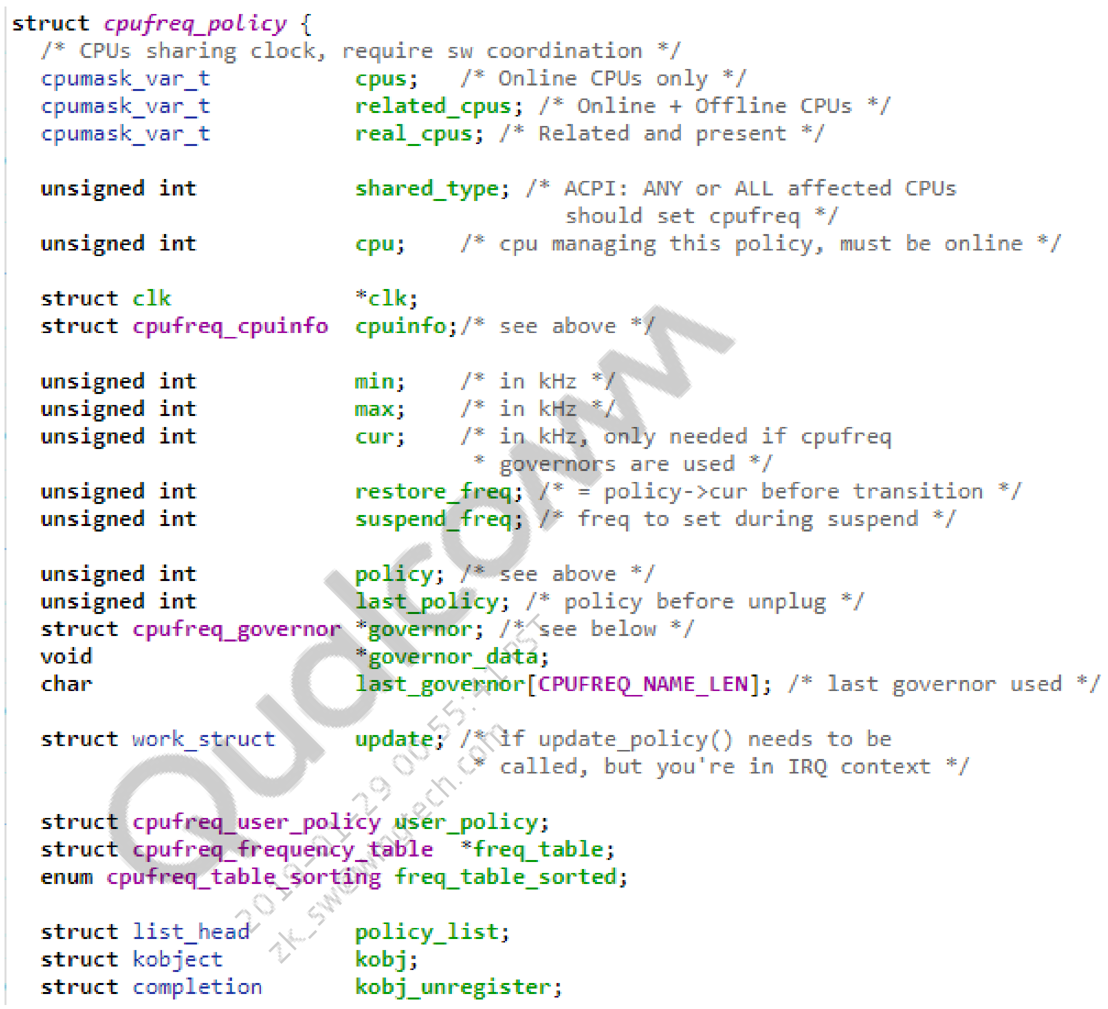
```
1|sdm845:/proc/sys/kernel # ls -al sched_*
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_autogroup_enabled
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_boost
-r--r--r-- 1 root root 0 2017-08-17 21:03 sched_cfs_boost
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_child_runs_first
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_cpu_high_irqload
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_cstate_aware
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_downmigrate
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_group_downmigrate
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_group_upmigrate
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_initial_task_util
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_is_big_little
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_latency_ns
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_migration_cost_ns
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_min_granularity_ns
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_nr_migrate
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_rr_timeslice_ms
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_rt_period_us
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_rt_runtime_us
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_schedstats
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_shares_window_ns
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_sync_hint_enable
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_time_avg_ms
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_tunable_scaling
-rw-r--r-- 1 root root 0 2017-08-17 20:56 sched_upmigrate
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_use_walt_cpu_util
-rw-r--r-- 1 root root 0 2017-08-17 21:03 sched_use_walt_task_util
-rw-r--r-- 1 root root 0 1970-01-01 00:00 sched_wakeup_granularity_ns
```

3. For CPU freq policy parameters, check all nodes under the following paths:
   - Power (Silver cluster) – `/sys/devices/system/cpu/cpu0/cpufreq`
   - Perf (Gold cluster) – `/sys/devices/system/cpu/cpu4/cpufreq`



```
sdm845:/sys/devices/system/cpu/cpu0/cpufreq # ls -al
total 0
drwxr-xr-x 3 root   root      0 1970-01-01 00:00 .
drwxr-xr-x 4 root   root      0 1970-01-01 00:00 ..
-r--r--r-- 1 root   root   4096 1970-01-01 00:00 affected_cpus
-r-------- 1 root   root   4096 2017-08-17 20:58 cpuinfo_cur_freq
-r--r--r-- 1 root   root   4096 1970-01-01 00:00 cpuinfo_max_freq
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 cpuinfo_min_freq
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 cpuinfo_transition_latency
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 related_cpus
-r--r--r-- 1 root   root   4096 1970-01-01 00:00 scaling_available_frequencies
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 scaling_available_governors
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 scaling_boost_frequencies
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 scaling_cur_freq
-r--r--r-- 1 root   root   4096 2017-08-17 20:58 scaling_driver
-rw-r--r-- 1 root   root   4096 2017-08-17 20:56 scaling_governor
-rw-rw---- 1 system system 4096 1970-01-01 00:00 scaling_max_freq
-rw-rw-r-- 1 system system 4096 1970-01-01 00:00 scaling_min_freq
-rw-r--r-- 1 root   root   4096 2017-08-17 20:58 scaling_setspeed
drwxr-xr-x 2 root   root      0 2017-08-17 20:56 schedutil
```

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 */
  cpumask_var_t           real_cpus; /* Related and present */

  unsigned int            shared_type; /* ACPI: ANY or ALL affected CPUs
                                          should set cpufreq */
  unsigned int            cpu;    /* cpu managing this policy, must be online */

  struct clk              *clk;
  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            restore_freq; /* = policy->cur before transition */
  unsigned int            suspend_freq; /* freq to set during suspend */

  unsigned int            policy; /* see above */
  unsigned int            last_policy; /* policy before unplug */
  struct cpufreq_governor *governor; /* see below */
  void                    *governor_data;
  char                    last_governor[CPUFREQ_NAME_LEN]; /* last governor used */

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

  struct cpufreq_user_policy user_policy;
  struct cpufreq_frequency_table  *freq_table;
  enum cpufreq_table_sorting freq_table_sorted;

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

5.16 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.

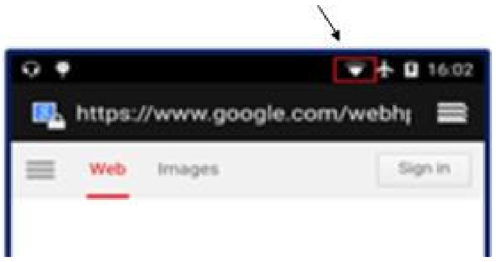
2. Check the temperature by typing the following commands:

   ```
   Cat /sys/class/thermal/thermal_zone1/temp
   ```

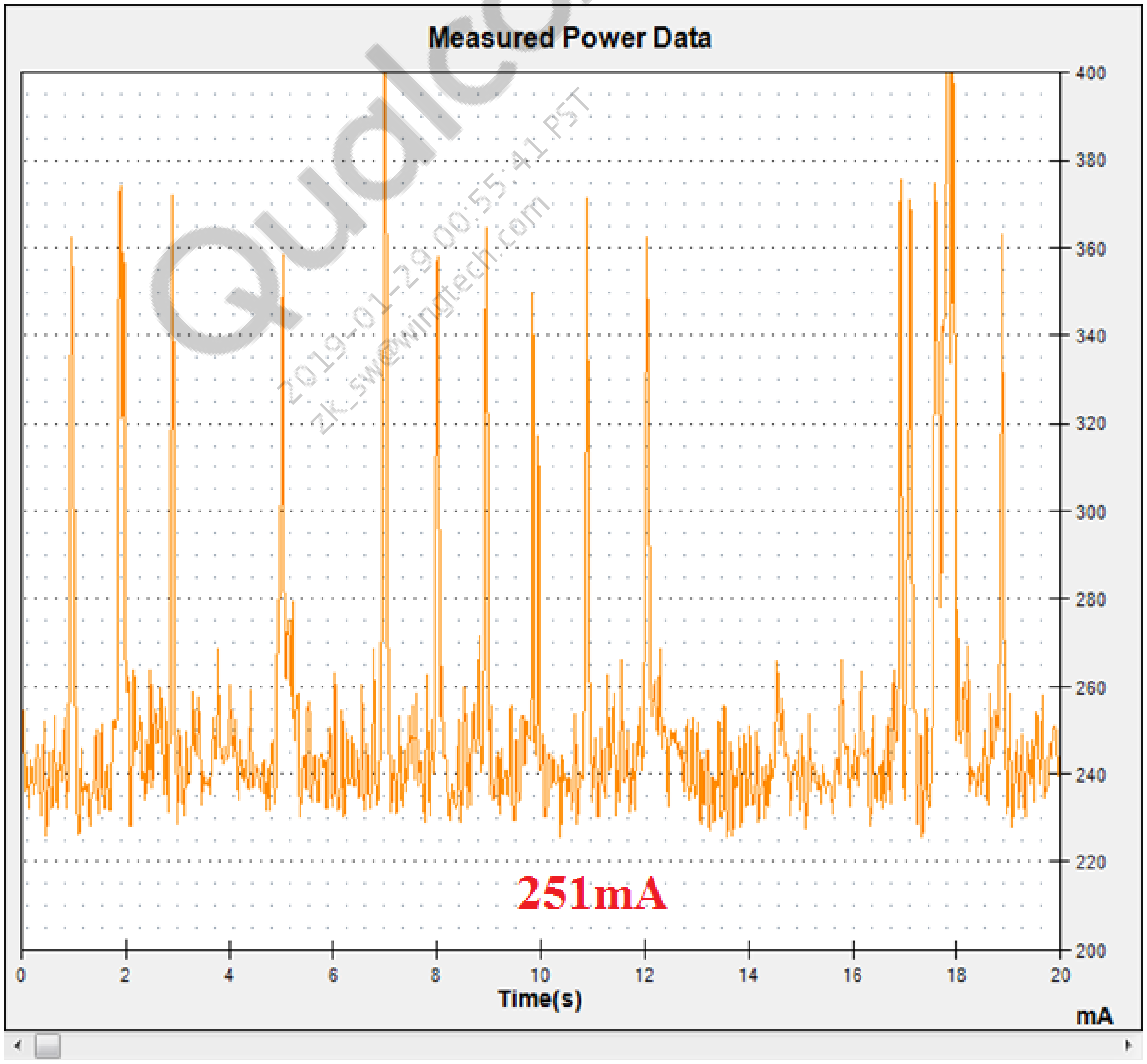
   Most dashboard use case data are captured with 25ºC excluding the graphics use case (75ºC).

5.17 Verify Wi-Fi power

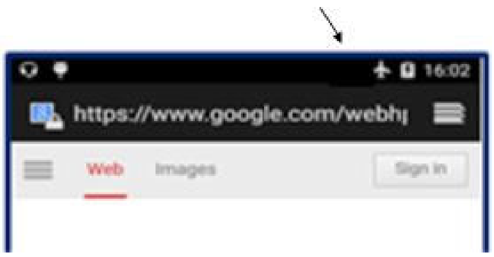
1. Use a dedicated apps processor to verify Wi-Fi power.
2. Open a browser with any page; see the following icon that shows Wi-Fi power is turned on:



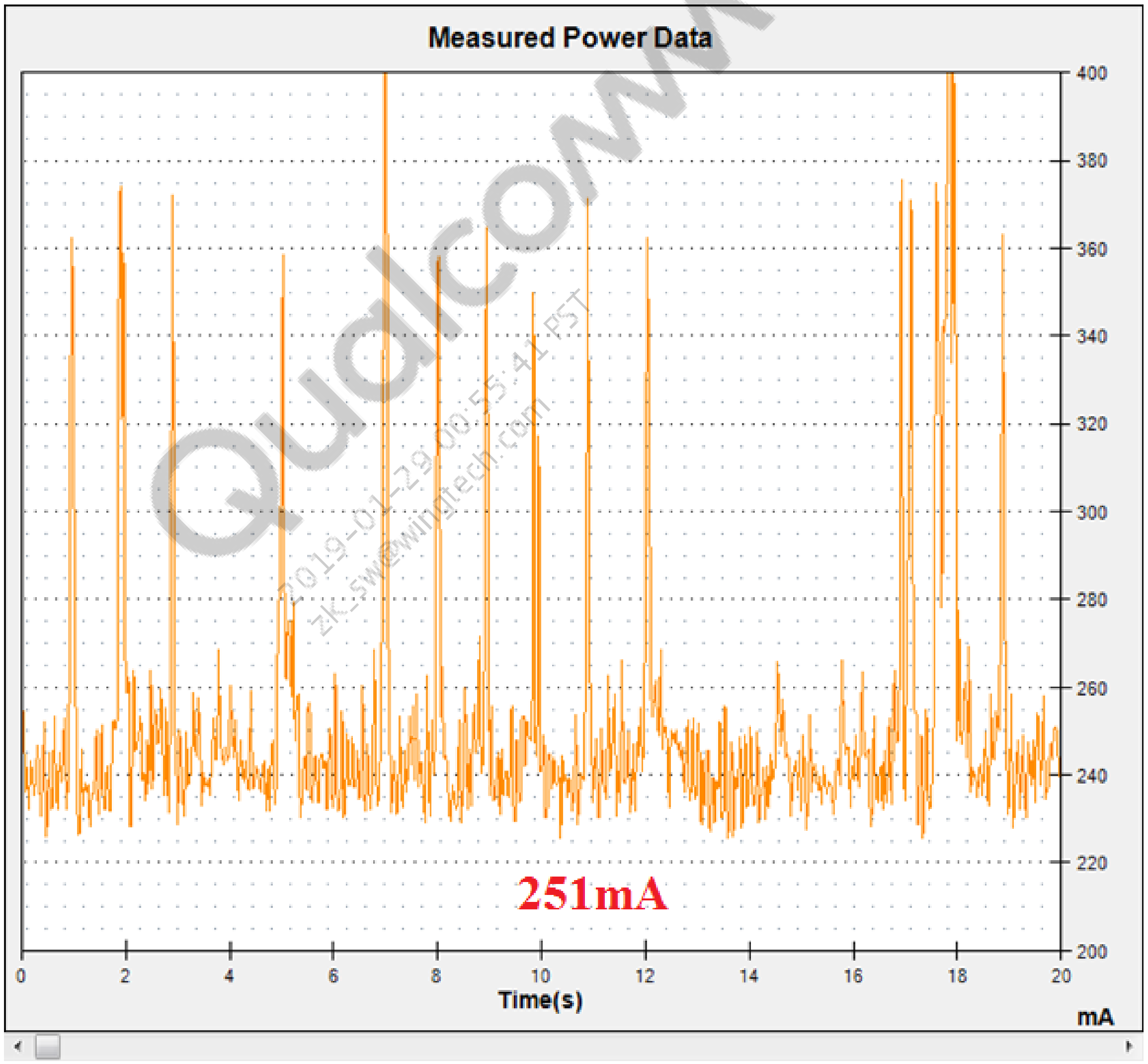
3. Measure power during 30 sec after the page loading is done. The following figure shows waveform with Wi-Fi on:



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



5. Measure power again during 30 sec. The following figure shows 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.

6 Log collection

6.1 AOSS logs

AOSS is a dedicated subsystem to manage the SoC's shared resources to obtain the lowest dynamic and static power profile possible. The subsystem incorporates hardened blocks (ARC, BCM, VRM, PDC, and DDR-AUX) and the always on processor (AOP/M3), which coordinate to handle power and resource requests. Each hardened block publishes a log about the resource states it handles and the AOP publishes a log into a limited area of spare message RAM.

The AOP log is designed as follows:

- In uLog format
- As a circular buffer, sized at about 4 kB
- As a raw log with a set of IDs and a variable number of parameters per message

6.1.1 Save AOP dumps

An AOP dump is also a part of the RAM dump collected after a system crash.

1. Open AOP T32.
2. Attach using the `sys.m.a` command.
3. Rereate the use case scenario where the AOP dumps are needed.
4. Break T32 at the desired point and run the following script:

```
do <aop build>\aop_proc\core\bsp\aop\scripts\aop_dump.cmm  <Location to
save dumps>
```

6.1.2 Load AOP dumps into T32 and extract logs

1. Open the T32 simulator.
2. Run the `do sys.up` command.
3. Run the following command:

```
do <aop build>\aop_proc\core\bsp\aop\scripts\aop_load_dump.cmm <dump path>
```

4. Load AOP.elf to start debug.

```
d.load.elf <RPM_Build>\aop_proc\core\bsp\aop\build \AOP_XXXXXXXXX.elf
   /nocode /noclear
```

6.1.3 Read AOP external logs (AOP uLog) using T32

Prerequisites:

To read the AOP external logs, one of the following conditions must be met:

- T32 is attached to the AOP and in the Break state
- AOP RAM dumps are loaded on the T32 simulator

1. Run the following command in T32:

   ```
   do <AOP_build>\aop_proc\core\power\ulog\scripts\aop_ulogdump.cmm
      <Location to save logs>
   ```

   This places the AOP external logs into the *`<location to save logs>`*. The AOP external logs require the use of a Python parsing tool to interpret its contents.

2. Run the `aop_log_hfam.py` Python script on the extracted logs using a command prompt.

   ```
   aop_log_hfam.py [-h] [-f] FILE [-tbl FILE] [-r] [-p]
   ```

Optional argument	Description
`-h, --help`	Displays the help message/options and exit
`-f FILE, --file FILE`	Path to the uLog file dumped by aop_ulogdump.cmm
`-p, --pretty`	Parses the timestamp
`-tbl FILE, --tblheader FILE`	Path to tracer_event_tbl.h
`-r, --raw`	Skips parsing the tracer_event_tbl.h and dumps the raw log file
`-d, --dump-file`	Dumps this script's result to a .txt file

Example

```
\\<aop_build>\aop_proc\core\bsp\aop\scripts\aop_log_hfam.py -f "<path>\AOP
External Log.ulog" -tbl<aop_build>\aop_proc\core\api\debugtrace
\tracer_event_tbl.h > AOP_ulog_parsed.txt
```

6.1.4 ARC logs using T32

6.1.4.1 ARC log information

The aggregated resource controller (ARC) controls the digital voltage domains of the platform and aggregates voted voltage operating points and sequences between operating points by coordinating with the core power reduction (CPR) and voltage regulator manager (VRM).

The ARC uses the following resources:

- CX
- MX
- EBI
- LCX
- LMX
- GFX
- MSS
- DDR_SS
- XO

The following table lists the ARC direct resource voters (DRVs):

ARC DRVs		
DRV0	tz	
DRV1	hyp	
DRV2	hlos	
DRV3	sec_proc	
DRV4	lpss	
DRV5	ssc	
DRV6	aop	
DRV7	debug	
DRV8	gpu	
DRV9	display	
DRV10	mss	
DRV11	mss_hw	
DRV12	cdsp	
DRV13	cprf	
DRV14	bcm_cd0	VCD0
DRV15	bcm_cd1	VCD1
DRV16	bcm_cd2	VCD2
DRV17	bcm_cd3	VCD3
DRV18	bcm_cd4	VCD4
DRV19	bcm_cd5	VCD5
DRV20	bcm_cd6	VCD6
DRV21	bcm_cd7	VCD7
DRV22	bcm_cd8	VCD8
DRV23	bcm_cd9	VCD9
DRV24	bcm_cd10	VCD10

ARC DRVs		
DRV25	bcm_cd11	VCD11
DRV26	bcm_cd12	VCD12
DRV27	bcm_cd13	VCD13
DRV28	bcm_cd14	VCD14
DRV29	bcm_cd15	VCD15

The following table lists ARC operating level index:

hlvl	MSS (mss.lvl)	CX (cx.lvl)	GFX (gfx.lvl)	MX (mx.lvl)	LPI_CX (lcx.lvl)	LPI_MX (lmx.lvl)	VDDA_EBI (ebi.lvl)
0	OFF	OFF	OFF	OFF	OFF	OFF	OFF
1	RET	RET	RET	RET	RET	RET	RET
2	MIN_SVS	MIN_SVS	MIN_SVS	SVS_L1	MIN_SVS	SVS	MIN_SVS
3	LOW_SVS	LOW_SVS	LOW_SVS	NOM	LOW_SVS	SVS_L1	LOW_SVS
4	SVS	SVS	SVS	TUR	SVS	NOM	SVS
5	SVS_L1	SVS_L1	SVS_L1	—	SVS_L1	TUR	SVS_L1
6	NOM	NOM	NOM	—	NOM	—	NOM
7	TUR	TUR	NOM_L1	—	TUR	—	TUR
8	—	—	TUR	—	—	—	—
9	—	—	TUR_L1	—	—	—	—

Note: Operation levels listed are examples and are for reference purposes only.

6.1.4.2 Dump ARC logs

1. Run the ARC dump script.

```
do \\<aop_build>\aop_proc\core\power\arc\scripts\arc_dump.cmm
[op=<output file path>]
```

   This command dumps all information for each ARC resource. If the output file is not specified, it dumps the output on the T32 area.

2. To filter ARC dump output, run the following scripts:

   - For specific DRV(s):

```
do \\<aop_build>\aop_proc\core\power\arc\scripts\arc_dump.cmm
drvs=mss,aop   [op=<output file path>]
```

   - For specific RM(s):

```
do \\<aop_build>\aop_proc\core\power\arc\scripts\arc_dump.cmm
rms=mx,mss     [op=<output file path>]
```

Example results (no filter)

```
RM0 : cx
    Status
        Enable: 1
    OLs
        Curr : 0x1
        Agg : 0x1
        Dest : 0x1
        Solved : 0x0
        Seq : 0x1
    Sequencer
        Busy : 0
        PC : 108
        Instr : 0x400F
    Votes
        DRV0 : 0x0
        DRV1 : 0x0
        DRV2 : 0x1*          Deciding direct resource voter (DRV)
        ...                  Other DRV votes
        DRV29 : 0x0

RM1 : mx                     Other RM details
...
ARC_RM Map
ARC_DRV Map
Note : The deciding DRV is marked by "*"
```

6.1.4.3 Debug a stuck ARC sequencer

The sequencer instruction listing can be found in the AOP build at `<aop_build>\aop_proc\core\power\arc\cfg\<target id>\arc_rm_seq.rinit.`

1. Run the ARC dump script.

```
do \\<aop_build>\aop_proc\core\power\arc\scripts\arc_dump.cmm
[op=<output file path>]
```

2. Check the ARC sequencer state for the RM in question.
3. Check the ARC sequencer PC for the RM in question.

Example results (stuck sequencer for the MX RM)

```
RM1 : mx
    Status
        Enable: 1
    OLs
        Curr : 0x4
        Agg : 0x4
        Dest : 0x4
        Solved : 0x0
        Seq : 0x4
    Sequencer
        Busy : 1                      Sequencer is busy
        PC : 36                       Offset in sequencer memory
        Instr : 0x5005                Instruction where the sequencer is
                                      stuck
    Votes
        DRV0 : 0x0
        DRV1 : 0x0
        DRV2 : 0x4
        ...
        DRV29 : 0x0
```

6.1.4.4 Debug an unexpected ARC aggregated state

1. Run the ARC dump script.

   ```
   do \\<aop_build>\aop_proc\core\power\arc\scripts\arc_dump.cmm
   [op=<output file path>]
   ```

2. Check the ARC vote table and aggregated state for the RM in question.

Example results (aggregated state for CX RM)

```
RM0 : cx
    Status
        Enable: 1
    OLs
        Curr : 0x6      Current aggregated OL
        Agg : 0x6
        Dest : 0x6
        Solved : 0x0
        Seq : 0x6
…
    Votes
        DRV0 : 0x0
        DRV1 : 0x0
        DRV2 : 0x1
        DRV3 : 0x0
        DRV4 : 0x6*         Decider DRV for OL
        ...
        DRV29
```

6.1.5 BCM logs using T32

6.1.5.1 BCM log information

The bus clock manager (BCM) handles shared buses, clock/power domains, memory controller, and IPs. Vote units are domain specific, i.e., bandwidth, latency, etc.

The BCM uses the following resources:

- System NoC
- Aggre NoC
- MemNoC
- Multimedia NoC
- Config NoC
- Peripheral NoC
- QUPv3
- IPA
- Crypto

The following table lists the virtual clock domains (VCDs):

Clock domain	Index	VCDs
MC	0	VCD0
SHUB	1	VCD1
SHRM	2	VCD2
SNoC	3	VCD3
MMNOC	4	VCD4
CNoC	5	VCD5
Crypto	6	VCD6
IPA	7	VCD7
QUP	8	VCD8
DDR_SS	10	VCD10
Active client vector	11	VCD11
Unused	9, 12-15	–

BCM logs contain information for the following areas:

- Front end
- Back end
- Vote table

Front end

- Contains information about VCD states
- AGG_BW – Aggregated bandwidth value
- FINAL_CP – Final selected CP; aggregation of SND CP and AGG_CP

- AGG_CP – Aggregated CP selection based on aggregated bandwidth
- BCM special node CP selections
  - SEL_CP – Selected CP based on latency vote and current voted load

Sample front end output

```
BCM Front End:
VCD0: AGG_BW: 0x1533 FINAL_CP: 0x8 AGG_CP 0x8
VCD1: AGG_BW: 0x1900 FINAL_CP: 0x5 AGG_CP 0x5
...
BCM SNDs:
SND0: SEL_CP: 0x0
SND1: SEL_CP: 0x0
...
```

Back end

- Back end per VCD status
- CLK_DEST_STATE – Next selected CP
- COMBINED_CP – Combined software and hardware selected CP
- SW_CP_SNAP – Software override CP selection (uncommon)
- WRITTEN_CP – Current CP written out of BCM
- CURR_CP – Current acknowledged CP
- Back end sequencer status
  - IDLE – 0x1; not busy
  - CURR_PC – 1 instruction past last run instruction

Sample back end output

```
BCM Back End:
VCD0: CLK_DEST_STATE: 0x8 COMBINED_CP: 0x8 SW_CP_SNAP: 0x0 WRITTEN_CP: 0x8
CURR_CP: 0x8
VCD1: CLK_DEST_STATE: 0x5 COMBINED_CP: 0x5 SW_CP_SNAP: 0x0 WRITTEN_CP: 0x5
CURR_CP: 0x5
...
BCM Back End Sequencers:
VCD0: IDLE: 0x1 CURR_PC: 0x3E
VCD1: IDLE: 0x1 CURR_PC: 0x3E
```

Vote table

- Per DRV, per BCM vote status
- VALID – Vote validity
- VOTE_X – Resource dependent, but usually average bandwidth (AB)
- VOTE_Y – Resource dependent, but usually instantaneous bandwith (IB)

AB is a value representing the bandwidth requirement. The AB value represents the average bandwidth over a sustained period. IB is a value representing the peak bandwidth requirement. The IB value represents the peak expected bandwidth in a given period.

Sample vote table output

```
Vote Table for DRV ID: 0
Vote Table for DRV ID: 1
Vote Table for DRV ID: 2
BCM0: VALID: 0x1 VOTE_X: 0x0 VOTE_Y: 0x1533
```

6.1.5.2 Dump BCM logs

Run the BCM dump script.

```
do \\<aop_build>\aop_proc\core\systemdrivers\icb\scripts\BCMDump.cmm
[op=<output file path>]
```

This command dumps all information for each VCD and BCM instance.

Example results

```
BCM Front End:
VCD0: AGG_BW: 0x1533 FINAL_CP: 0x8 AGG_CP 0x8
...
BCM SNDs:
SND0: SEL_CP: 0x0
...
BCM Back End:
VCD0: CLK_DEST_STATE: 0x8 COMBINED_CP: 0x8 SW_CP_SNAP: 0x0 WRITTEN_CP: 0x8
CURR_CP: 0x8
...
BCM Back End Sequencers:
VCD0: IDLE: 0x1 CURR_PC: 0x3E
...
Vote Table for DRV ID: 0
Vote Table for DRV ID: 1
Vote Table for DRV ID: 2
BCM0: VALID: 0x1 VOTE_X: 0x0 VOTE_Y: 0x1533
BCM3: VALID: 0x1 VOTE_X: 0x0 VOTE_Y: 0x8
```

6.1.5.3 Debug a stuck BCM sequencer

1. Run the BCM dump script.

   ```
   do \\<aop_build>\aop_proc\core\systemdrivers\icb\scripts\BCMDump.cmm
   [op=<output file path>]
   ```

2. Check the BCM sequencer state for the VCD in question.
3. Check the BCM sequencer PC for the VCD.

6.1.5.4 Debug an unexpected BCM aggregated state

1. Run the BCM dump script.

   ```
   do \\<aop_build>\aop_proc\core\systemdrivers\icb\scripts\BCMDump.cmm
   [op=<output file path>]
   ```

2. Check the BCM sequencer state for the VCD in question.
3. Check the BCM sequencer PC for the VCD in question.

Example results (unexpected aggregated state)

```
BCM Front End:
VCD0: AGG_BW: 0x1533 FINAL_CP: 0x8 AGG_CP 0x8     Final CP & AGG BW
VCD1: AGG_BW: 0x1900 FINAL_CP: 0x5 AGG_CP 0x5
...
Vote Table for DRV ID: 2
BCM0: VALID: 0x1 VOTE_X: 0x0 VOTE_Y: 0x1533     DRV2 voted for the above
chosen CP
BCM3: VALID: 0x1 VOTE_X: 0x0 VOTE_Y: 0x8
...
```

6.1.6 PDC logs using T32

6.1.6.1 PDC log information

The power domain controller (PDC) runs sleep/wake sequences to shut down/wakeup critical resources for a given subsystem. It handles wakeup interrupts when a subsystem is in RPMh-assisted power collapse.

PDC logs contain information for the following areas:

- Sequencer
- Trigger command set (TCS)
- Interrupt request (IRQ)
- Parameter

Sequencer

```
Sequencer
    Start : 0x0                     Start address when subsystem enters PC
    Busy : 0                        Sequencer state
    PC : 0x0                        Offset
    Instr : 0xE1                    Instruction
    Timer : 0x0                     PDC subsystem wake up timer
    Pwr Override : (0x0, 0x0)
    Wait Override : (0x0, 0x0)
    Br Override : (0x0, 0x0)
```

```
    Branches : (0x0, 0x0, 0x0, 0x0)
    Delays : (0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0)
```

TCS

TCS is a set of commands that can be programmed once by writing registers and triggered multiple times. These commands are typically votes to a set of critical resources that are required at a given time when a subsystem goes into and comes out of sleep.

```
TCS0
        AMC Mode : 0          Active mode command
        En Mask : 0x7         Control to enable/disable the no number of
                              commands in TCS
        Wait Mask : 0         Wait until current processing is completed
        Idle : 0              TCS controller state
        CMD0
            slave: 3, addr: 0x80, data: 0x1
            resp req: 0, triggerd: 0,                issued: 0, ackd: 0
         …
        CMDc
    …
    TCSn
```

IRQ

```
IRQs (SW)
  Num  : (ss_irq, gpio, en, status, cfg, owner)
      0  : (475, -, 0, 0, high, 0)
      1  : (476, -, 0, 0, falling, 0FFFFFFFF)
      2  : (477, -, 0, 0, falling, 0FFFFFFFF)
      3  : (478, -, 0, 0, falling, 0FFFFFFFF)
      4  : (479, -, 0, 0, falling, 0FFFFFFFF)
      5  : (480, -, 0, 0, falling, 0FFFFFFFF)
      6  : (481, -, 0, 0, falling, 0FFFFFFFF)
      …
      IRQn
```

Parameter

The parameter section of the logs provides the number of AOP interrupts, sequencer memory, DRV and TCS counts, version, etc.

```
Params
    Version
        Major : 1
        Minor : 0
    Resources
        DRVs : 2
        TCSs : 3
        cmds/tcs : 4
        ProfUnits : 5
        TimeStamps : 0
Sequencer
        CMDs : 96
        XPwrCtrls : 1
        XWaitInputs : 1
        AOP IRQs : 2
    IRQ/GPIOs
        IRQs : 15
        GPIOs : 96
        GP SELs : 30
```

6.1.6.2 Dump PDC logs

Run the PDC dump script.

```
do \\<modem_build>\modem_proc\core\power\pdc\scripts\pdc_dump.cmm
ss=<subsystem name> [op=<output file path>]
```

This script is located with the PDC driver at `//<image>/<proc>/core/power/pdc/script/pdc_dump.cmm`.

6.1.6.3 Dump the PDC uLog

In non-HLOS, run the uLogDump.cmm command.

```
ULogDump.cmm <output file path> PDC Log
```

Example results

```
Content-Type: text/pdc-driver-1.0; title=PDC Driver
         0x2F92FC02: Initializing target
TCS0
   AMC Mode : 0
   En Mask : 0x7
   Wait Mask : 0
   Idle : 0
   CMD0
   slave: 3, addr: 0x80, data: 0x1      Vote based on TCS configuration
   resp req: 0, triggerd: 0, issued: 0,
   ackd: 0
     …
      CMDc
    …
    TCSn
         0x2F93089E: Initializing main driver
         0x2F930B86:  Interrupt configuration (Number of Interrupts:
         15)
         0x2F930DA4:  GPIO configuration (Number of GPIOs: 96) (Number
         of MUXs: 20)
```

```
        0x2F9F7231: TCS Resource lookup: (Name: cx.lvl) (Base address:
        0x30000)
        0x2F9F77ED: TCS Resource lookup: (Name: mx.lvl) (Base address:
        0x30010)
        0x2F9F8287: TCS Resource lookup: (Name: xo.lvl) (Base address:
        0x30080)
        0x2F9F89DA: TCS write (Resource: xo.lvl) (TCS.Cmd: 0.0) (hlvl:
        1)
        0x2F9F8BC0: TCS write (Resource: cx.lvl) (TCS.Cmd: 0.1) (hlvl:
    0x2F9F969E: TCS write (Resource: mx.lvl) (TCS.Cmd: 1.2) (hlvl:
    0x2F9F9903: TCS write (Resource: mx.lvl) (TCS.Cmd: 2.0) (hlvl:
    0x2F9F9C0F: TCS write (Resource: cx.lvl) (TCS.Cmd: 2.1) (hlvl:
    0x2F9F9ED1: TCS write (Resource: xo.lvl) (TCS.Cmd: 2.2) (hlvl:
    0x2F9FA04A: TCS registers programmed successfully
    0x2FB5EF50: Config subsystem int 475 (PDC bit: 0) (trigger: 4)
    0x2FD0A43E: Successfully enabled 4 of 5 profiling units
```

6.1.6.4 Debug a stuck PDC sequencer

1. Run the PDC dump script.

```
do \\<modem_build>\modem_proc\core\power\pdc\scripts\pdc_dump.cmm
ss=<subsystem name> [op=<output file path>]
```

2. Check the PDC sequencer state for the DRV in question.
3. Check PDC sequencer PC for the DRV in question.
   - The PC points to the sequencer mem offset where the sequencer is stuck.
   - The instruction provides the instruction where the sequencer is stuck.
   - The sequencer instruction listing can be found in AOP build at `<aop_build>\aop_proc\core\power\pdc\seq\cfg\<target id>\aop\pdc_seq_cfg.c`.

Example results (for stuck sequencer)

```
Sequencer
    Start : 0x0
    Busy : 1            Sequencer is busy
    PC : 0x18           Offset
    Instr : 0xE1        Instruction where the sequencer is stuck
    Timer : 0x0
    Pwr Override : (0x0, 0x0)
    Wait Override : (0x0, 0x0)
    Br Override : (0x0, 0x0)
    Branches : (0x0, 0x0, 0x0, 0x0)
    Delays : (0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,   0x0,
0x0, 0x0, 0x0, 0x0)
```

6.1.6.5 Debug a subsystem wakeup issue

1. Run the PDC dump script.

   ```
   do \\<modem_build>\modem_proc\core\power\pdc\scripts\pdc_dump.cmm
   ss=<subsystem name> [op=<output file path>]
   ```

2. Check the PDC IRQ status.
3. Check the IRQ configuration for the expected interrupt.
4. Check the TIMER_MATCH_VALUE registers and compare them to the current time. If software logs are available, compare this information to data from software logs.

Example results (for incorrect IRQ configuration)

```
IRQs (SW)
    Num  : (ss_irq, gpio, en, status, cfg, owner)
      0  : (475, -, 0, 0, high, 0)
      1  : (476, -, 0, 0, falling, 0FFFFFFFF)
      2  : (477, -, 0, 0, falling, 0FFFFFFFF)
      3  : (478, -, 0, 0, falling, 0FFFFFFFF)   Incorrect configuration
      4  : (479, -, 0, 0, falling, 0FFFFFFFF)
      5  : (480, -, 0, 0, falling, 0FFFFFFFF)
      6  : (481, -, 0, 0, falling, 0FFFFFFFF)
      …
      IRQn
```

6.1.6.6 Debug a resource in an unexpected state during sleep

1. Run the PDC dump script.

   ```
   do \\<modem_build>\modem_proc\core\power\pdc\scripts\pdc_dump.cmm
   ss=<subsystem name> [op=<output file path>]
   ```

2. Check the PDS TCS log contents and compare to the expected for a given Sleep mode. Slaves are the internal blocks within AOP.
   - 3 – ARC
   - 4 – VRM
   - 5 – BCM

6.1.7 VRM logs using T32

6.1.7.1 VRM log information

The voltage regulator manager (VRM) along with the AOP manages the execution of votes on PMIC regulators and clock buffers. The VRM aggregates regulator votes, transitions the state of the regulator via the PMIC arbiter, and handles any settling time. The VRM has a current resource state (CRS) register which indicates the final VRM resource setting settled state.

The VRM supports voting on the following resource types:

- Regulators – Enable, mode, voltage, and headroom
  - PMIC_STS – Actual voltage set at the PMIC
  - VRM_CRS – Current state
  - VRM_WRS – Written resource state; last written state, mostly similar to VRM_CRS
  - VRM_DRS – Desired resource state; final aggregated state
  - AUTO_MODE – Dynamic switching between pulse frequency modulation (PFM) and pulse width modulation (PWM) based on load
  - NPM_MODE – PWM
  - LPM_MODE – PFM
- XOB – Enable
- XO – Enable

6.1.7.2 Get the VRM status

Run the following command, which dumps all information on each regulator and its votes:

```
do \\<glue_build>\\common\Core\tools\tools\systemdrivers\pmic\hoya\
    vrm.cmm [op=<output file path>]
```

Sample output

```
------------------------------ SMPS3A REG0 VOTES
---------------------------------
PMIC_STS - ON AUTO_6 1360mV
VRM_CRS  - ON AUTO_6 1360mV 0mV
VRM_WRS  - ON AUTO_6 1360mV 0mV
VRM_DRS  - ON AUTO_6 1360mV 0mV
TZ     - OFF UNUSED_0    0mV  0mV | HYP    - OFF UNUSED_0    0mV  0mV |
HLOS   - ON  UNUSED_0 1352mV  0mV | SECPRO - OFF UNUSED_0    0mV  0mV
AUDIO  - OFF UNUSED_0    0mV  0mV | SENSOR - OFF UNUSED_0    0mV  0mV |
AOP    - ON    AUTO_6 1360mV  0mV | DEBUG  - OFF UNUSED_0    0mV  0mV
GPU    - OFF UNUSED_0    0mV  0mV | DISP   - OFF UNUSED_0    0mV  0mV |
MDMSW  - OFF UNUSED_0    0mV  0mV | MDMHW  - OFF UNUSED_0    0mV  0mV
COMDSP - OFF UNUSED_0    0mV  0mV | ARCCPR - OFF UNUSED_0    0mV  0mV |
BCMCD0 - OFF UNUSED_0    0mV  0mV
```

6.1.7.3 Dump the VRM logs

Run the following command:

```
do \\<glue_build>\\common\Core\tools\tools\systemdrivers\pmic\hoya\
   vrm_dump.cmm <target> <register_filter> [op=<output file path>]
```

- <target> – Target on which the script is executed
- <register_filter> – Optional register filter name for debugging

Sample output

```
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG30_ADDR : 0x80000A90 = 2147486352
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG31_ADDR : 0xA90 = 2704
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG32_ADDR : 0xB30 = 2864
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG33_ADDR : 0x0 = 0
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG34_ADDR : 0x80000C10 = 2147486736
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG35_ADDR : 0x80000CF0 = 2147486960
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG36_ADDR : 0x0 = 0
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG37_ADDR : 0x80000708 = 2147485448
HWIO_RPMH_VRM_VOLTAGE_VOTE_DRV2_REG38_ADDR : 0x80000708 = 2147485448
```

6.1.7.4 Debug an unexpected state of regulator or resource

1. Get the VRM status logs.

   ```
   do \\<glue_build>\\common\Core\tools\tools\systemdrivers\pmic\hoya\
          vrm.cmm [op=<output file path>]
   ```

2. Check the current state of the regulator or resource in question.
3. Check the last written state of the regulator or resource in question.
4. Check the final aggregated state of the regulator or resource in question.
5. Check the final state votes.

The following example shows where to check the states and votes for an unexpected state issue:

```
------------------------------- SMPS7A REG1 VOTES
--------------------------------
PMIC_STS - ON NPM_7 2040mV       Actual voltage set at the PMIC
VRM_CRS  - ON NPM_7 2040mV 0mV    Current state
VRM_WRS  - ON NPM_7 2040mV 0mV    Last written state
VRM_DRS  - ON NPM_7 2040mV 0mV    Final aggregated state
TZ     - OFF UNUSED_0    0mV  0mV | HYP    - OFF UNUSED_0    0mV  0mV |
HLOS   - ON  UNUSED_0 2040mV  0mV HLOS votes for final state of SMPS7A
| SECPRO - OFF UNUSED_0    0mV  0mV
AUDIO  - OFF UNUSED_0    0mV  0mV | SENSOR - OFF UNUSED_0    0mV  0mV |
AOP    - ON    AUTO_6 1904mV  0mV | DEBUG  - OFF UNUSED_0    0mV  0mV
GPU    - OFF UNUSED_0    0mV  0mV | DISP   - OFF UNUSED_0    0mV  0mV |
MDMSW  - OFF    NPM_7    0mV  0mV | MDMHW  - OFF UNUSED_0    0mV  0mV
COMDSP - OFF UNUSED_0    0mV  0mV | ARCCPR - OFF UNUSED_0    0mV  0mV |
BCMCD0 - OFF UNUSED_0    0mV  0mV
```

6.2 Hansei RAM dump parser

1. Get the parser, which is located under the AOP build at `//<aop_crm_build>/aop_proc/core/bsp/aop/scripts/hansei`.
2. Run hansei.py from a command prompt:

```
python hansei.py [-h] [--elf aop.elf] [--output logs_path] [--verbose] [--
parser parse_path] [--parser_rel] dumpfile [dumpfile ...] -t
<target_number>
```

   Not all arguments are required in a typical hansei use case. Mandatory arguments are:

   - elf – Location of the aop.elf file
   - output – Location where the hansei output is to be created
   - dumpfile – Location of the dumps
   - target – Target number, e.g., 845

   Output

   - aop-log.txt – Postprocessed AOP log
   - ddr-log.txt – Postprocessed DDR log
   - ddr_stats.txt – DDR stats information, including the recent frequency switch history
   - npa-dump.txt – Standard NPA dump format without the timestamps
   - sleep_stats.txt – Sleep stats of the AOSS sleep and CX collapse
   - cmd_db.txt – Mapping of the named RPMh resources and addresses
   - binaries\RPMH_BINARY_RECOVERED.BIN – Recovered RPMh register dump (load into T32 for RPMh debugging)
   - reqs_by_master – Consolidated votes of each DRV to different RPMh resources
   - reqs_by_resources – Consolidated votes for each resource by different DRVs

6.3 Collect GPIO dumps

1. Run the following script from the AOP T32 window:

```
do <Modem_Build>\modem_proc\core\systemdrivers\tlmm\scripts\sdm845\
tlmm_gpio_hw.cmm
```

   All GPIOs in the hardware use this script to read the current configuration.
2. Select **Option 14: Read All GPIO Configurations**.

6.4 Collect PMIC dumps

1. Open any T32 window (AOP is recommended).
2. Type `sys.m.a` to attach T32 to the device under test.

3. Load the .elf file and set the breakpoint if 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 <GLUE_Build>\common\Core\tools\tools\systemdrivers\pmic\hoya
   \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=<GLUE_ BUILD>\common\Core\tools\tools
   \systemdrivers\pmic\hoya\pmi8998\v1 \CORE_ADDRESS_FILE_CUSTOMER.FLAT --
   file=pmicdump.xml > pmic_parsed.txt
   ```

6.5 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.

6.5.1 Collect clock dumps using JTAG

1. Open any T32 window (AOP is recommended).
2. Type `sys.m.a` to attach T32 to the device under test.
3. Load the .elf file and set breakpoint if 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.
   - Type the following command:

     ```
     do <GLUE_BUILD>\common\core\tools\tools\systemdrivers\clock\sdm845
     \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.

6.5.2 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 5 && while true;
do echo \=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=;
cat /proc/uptime;
cd /sys/kernel/debug/clk;
for i in *;
do if [ -d $i ];
then if [ -e $i/clk_measure ];
then echo $i \=\> measure:`cat $i/clk_measure`;
else echo $i \=\> rate:`cat $i/clk_rate`;
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 3;
done > /data/dumpclk.txt &
```

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

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

6.6 Collect modem logs

Collect modem uLogs and modem NPA logs.

6.6.1 Collect modem uLogs

Run the following script from Modem_Build in the modem T32 window:

```
do  <Modem_Build>\modem_proc\core\services\diag\diagbuffer\scripts
\UlogDump.cmm  <location to save logs>
```

This places the modem uLogs in the specified directory.

6.6.2 Collect modem NPA logs

Run 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.

6.7 Collect modem F3 messages/QXDM Professional logs

1. Connect the device to a PC via USB where the QXDM Professional 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; press **Alt+L** again to stop logging.
5. Obtain the saved logs from the location defined in **Options > Log View Config > Misc > Log File Path**.

6.8 Capture ftrace logs

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 150000 > 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 irq:* kgsl:kgsl_clk msm_bus:bus_update_request msm_low_power:*
clk:clk_disable clk:clk_enable clk:clk_set_rate power:cpu_frequency
```

```
power:cpu_idle sched:sched_switch sched:sched_task_util_energy_aware
sched:sched_task_util_energy_diff sched:sched_task_util_overutilzed
sched:sched_task_util_colocated sched:sched_task_util_bias_to_waker
sched:sched_wakeup sched:sched_wakeup_new power:bw_hwmon_meas
power:bw_hwmon_update thermal:* power:memlat_dev_update
power:memlat_dev_meas sched:sched_isolate sched:core_ctl_set_busy
sched:core_ctl_eval_need perf_trace_counters:sched_switch_with_ctrs >
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 the PID 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>
```

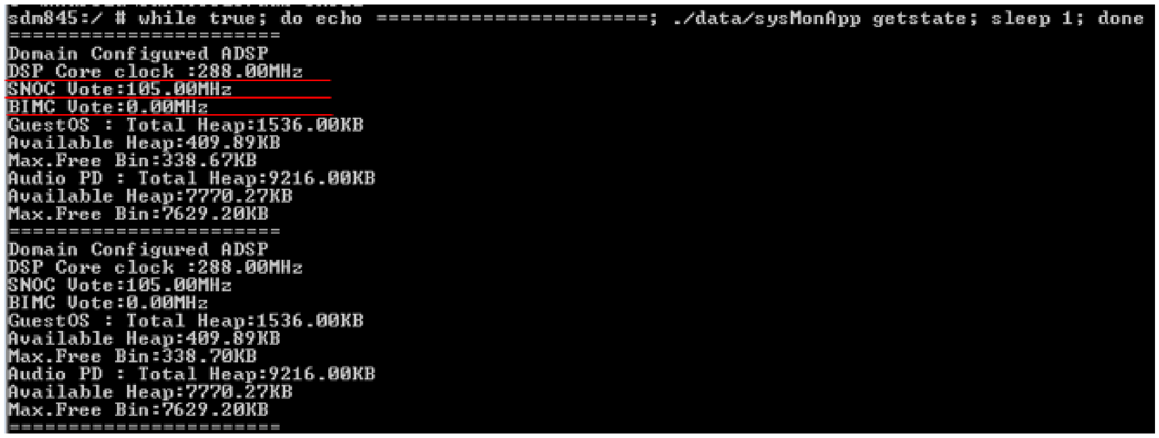
6.9 Capture ADSP clock and SNOC voting

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

```
adb root
adb remount
adb push sysMonApp /data/
adb shell chmod 777 /data/sysMonApp
adb shell
```

3. Run the use case and type the following command to capture ADSP clock and SNOC voting:

```
while true; do echo =======================; ./data/sysMonApp getstate;
sleep 1; done
```



```
sdm845:/ # while true; do echo =======================; ./data/sysMonApp getstate; sleep 1; done
=======================
Domain Configured ADSP
DSP Core clock :288.00MHz
SNOC Vote:105.00MHz
BIMC Vote:0.00MHz
GuestOS : Total Heap:1536.00KB
Available Heap:409.89KB
Max.Free Bin:338.67KB
Audio PD : Total Heap:9216.00KB
Available Heap:7770.27KB
Max.Free Bin:7629.20KB
=======================
Domain Configured ADSP
DSP Core clock :288.00MHz
SNOC Vote:105.00MHz
BIMC Vote:0.00MHz
GuestOS : Total Heap:1536.00KB
Available Heap:409.89KB
Max.Free Bin:338.70KB
Audio PD : Total Heap:9216.00KB
Available Heap:7770.27KB
Max.Free Bin:7629.20KB
=======================
```

6.10 Capture PowerTop and Top data

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

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

3. Type the following commands to start capturing PowerTop and Top data:

```
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 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>
```

6.11 Capture clock and regulator dumps

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

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

3. Start capturing data by typing the following command:

```
sleep 5 && while true;
do echo \=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=;
cat /proc/uptime;
cd /sys/kernel/debug/clk;
for i in *;
do if [ -d $i ];
then if [ -e $i/clk_measure ];
then echo $i \=\> measure:`cat $i/clk_measure`;
else echo $i \=\> rate:`cat $i/clk_rate`;
```

```
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 3;
done > /data/dumpclk.txt &
```

4. Disconnect the USB when the PID 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>
```

6.12 Collect wake locks

Type one of the following commands:

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

or

```
adb shelldumpsys power
```

6.13 Collect L3 clock and voting

Type the following commands:

```
adb shell
cat /sys/kernel/debug/clk/l3_clk/clk_measure
cat /sys/kernel/debug/clk/l3_cluster0_vote_clk/clk_rate
cat /sys/kernel/debug/clk/l3_cluster1_vote_clk/clk_rate
The following nodes show the mapping table b/w cpu freq and l3 freq.
cat /sys/class/devfreq/soc:qcom,l3-cpu0/mem_latency
cat /sys/class/devfreq/soc:qcom,l3-cpu4/mem_latency
```

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
System Power Monitor Version 4.1 Application Note	80-N6594-16
Power Consumption Measurement Procedure for MSM (Android-Based)/MDM Devices	80-N6837-1
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
AOSS	Always on subsystem
APSS	Applications processor subsystem
BIMC	Bus integrated memory controller
CAC	Chromatic aberration correction
CAF	Code aurora forum
CPP	Camera postprocessor
CPU	Central processing unit
DRX	Discontinuous reception
FPS	Frames per second
GPIO	General purpose input/output
GPU	Graphics processing unit
IQ	Image quality
ISP	Image signal processor
MCPM	Modem clock and power manager
MDP	Mobile display processor
LDO	Low dropout (voltage regulator)
LPASS	Low power audio subsystem
MPSS	Modem peripheral subsystem

Acronym or term	Definition
NPA	Node power architecture
PID	Process ID
PMIC	Power management integrated circuit
QPST	Qualcomm Product Support Tool
RAM	Random access memory
RF	Radio frequency
RPM	Resource power manager
SMPS	Switched-mode power supply
SPM	System power monitor
TNR	Temporal noise reduction
XO	Crystal oscillator
YUV	Luminance, bandwidth, chrominance; also known as YCbCr and YPbPr