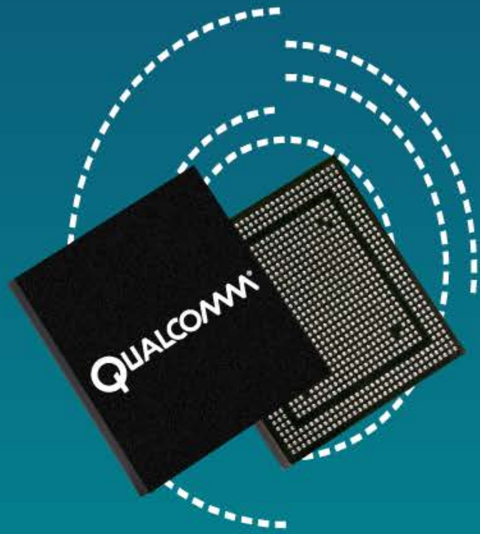
Qualcomm
TECHNOLOGIES, INC



Multimedia Power Debugging Case Studies Overview

80-NT616-1 C

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

Revision History

Revision	Date	Description
A	Oct 2014	Initial release
B	Oct 2014	Formatting updates, no new information
C	Dec 2014	Updated slide 45 and slide 48

Contents

Objectives

- This presentation provides debugging steps for multimedia power-related cases that have been resolved by QTI engineers.
- At the end of this presentation, you will understand:
  - Steps and debugging techniques used to resolve multimedia power-related issues
  - How to file a case/SR if the problem persists after using all the techniques

Debugging/Optimizing/Tuning

Steps for Debugging

Compare your power rail data with QTI reference data

VDD_APC / VDD_CX

- Systrace / Clock residency / Clock dump → CPU, GPU, BIMC
- Top / Perf top / Powertop → CPU
- MSM bus voting → GPU, BIMC
- MSM bus requests / NPA dump → BIMC, MDP

VDD_MX

- Memory
- L2 cache
- OCMEM

VDD_MSS

- Q6V5F core
- MEMPOOL
- TDEC
- DemBack

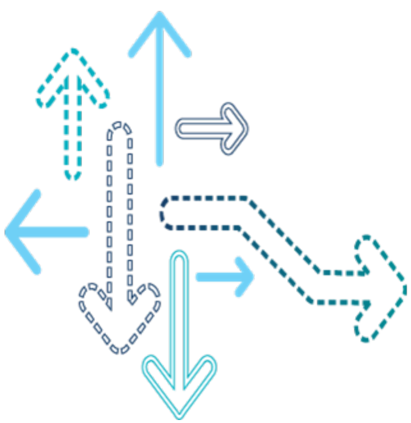
Determining Tuning Parameters

Compare your power rail data with QTI reference data

VDD_APC / VDD_CX		VDD_MX	VDD_MSS
CPU	Interactive governor parameters Thread migration parameters *above_hispeed_delay, go_hispeed_load, target_loads, timer_rate, hispeed_freq, scaling_min_freq, min_sample_time, sampling_down_factor, sync_threashold, io_percent*	Memory *L2 Cache* *OCMEM*	Q6V5F core MEMPOOL TDEC
GPU	GPU init clock Idle timer		
MDP	Fudge factor Mdpcomp.idletime		
BIMC	BIMC min lock Static mapping with GPU clock and BIMC clock		

Note: Memory (L2 cache, OCMEM) depends on BIMC, so it needs to be tuned.

Video Playback Power Debugging



Debugging the Video Playback Use Case

Step	Action	Reference
1	If the customer use case is different from the standard QTI use case, verify the QTI standard first.	
2	Compare the power of the customer device with QTI reference power data in the power app note after isolating the chipset.	▪ *MSM8994 Linux Android Current Consumption Data* (80-NJ051-7) ▪ *MSM8936/MSM8939 Linux Android Current Consumption Data* (80-NM683-7)
3	Capture the full breakdown on the customer device and compare it with the published QTI breakdown.	
4	Identify which power rail is higher than the QTI power data and determine the subsystems on it.	
5	Capture a clock dump to analyze the subsystems.	Trace 32 Adb command
6	Compare the captured data with the published clock plan for the video playback use case to determine which clock is high.	
7	If the CPU clock is high, capture CPU-related information using PowerTop and Top.	
7.1	Determine which process is consuming the most CPU and look for unexpected processes/ interrupts. If there are unexpected processes or threads, contact QTI. If there are no unexpected processes or threads, examine the high CPU usage process or thread using PerfTop. If PerfTop shows any module with high usage, contact QTI. If there is an unexpected interrupt, check the pytime chart to determine the source of interrupt. If there is no source listed, tune the CPU clock by scaling_min_freq or the parameters of the interactive governor.	
7.2	If the MDP clock is high, capture SurfaceFlinger information using dumpsys and compare it with the MTP. Compare the clock voting in d/msm-bus-dbg/client-data/mdss_mdp with the MTP.	Adb shell dumpsys SurfaceFlinger Adb shell cat /d/msm-bus-dbg/client-data/mdss_mdp
8	If any other clock is high, collect FTrace, NPA dump, wave form, and Systrace logs.	

Problem Definition

- Customer is reporting 720p video playback power is high for an APQ8084-based product
- Determine customer use case power consumption
  - Use case mentioned is a QTI standard use case
  - Customer measured power consumption using QTC77
  - Customer device power consumption is high using the QTI standard dashboard use case
  - Root cause of power difference must be determined

Customer device current (mA)	QTI reference current (mA)
580	425

Comparing Customer Device Power with the QTI Reference

- 580 mA measurement at the battery with display attached
- Display power (touch, LCD, backlight current) was isolated and is ~320 mA
- Chipset power alone on the customer device is 580 - 320 = 260 mA
- Reference current for APQ8084 during the 720p video playback use case using a WQXGA panel is 100 to 110 mA

Capturing the Breakdown

- Customers are requested to get a breakdown board with channels mapped to the channel numbers/test point numbers mentioned on the power grid for the chipset.
- The breakdown board, in combination with a power measurement system like Kratos or Xerxes, can be used to capture the current/voltage on all rails of the system.
- The table to the right provides a sample output of a breakdown.

Rail	Voltage (V)	Current (mA)
SMPS1A_input	3.681216	9.784111977
SMPS2A_input	3.680652	40.91268158
SMPS12A_input	3.681581	3.099509954
SMPS3A_input	3.680847	21.16083908
SMPS4A_input	3.680409	12.72926044
SMPS5A_input	3.679405	43.97269821
SMPS6A_input	3.681721	1.057733059
SMPS7A_input	3.681741	-0.075547047
SMPS8A_input	3.681626	21.77234077
SMPS9A_input	3.681826	0.960167885
SMPS10A_input	3.681819	0.122583598
SMPS11A_input	3.681805	0.203078002
LDO8A_input	3.680927	0.157750294
DBBYP1_input	3.68164	22.48349953
DBO_HDMI_input	3.681911	0.196008801
DBB2_input	3.680746	0.180944398
SMB_VPH_FLSH_PVDD		
DISP_FLASH_LED_DRV		
WCD9320_CDC_VDD_BAT	3.681199	0.242282704
DISP_DRV	3.654627	79.82125092
QCA6164_74_VPH	3.679191	0.40868181
QCA1990_VBAT	3.678732	0.205164298
LDO16A_LDO25A		
WLED_SMPS	3.629873	236.1233063

Breakdown Comparison

- This breakdown comparison shows that the excess current is coming from higher S2 and S4:

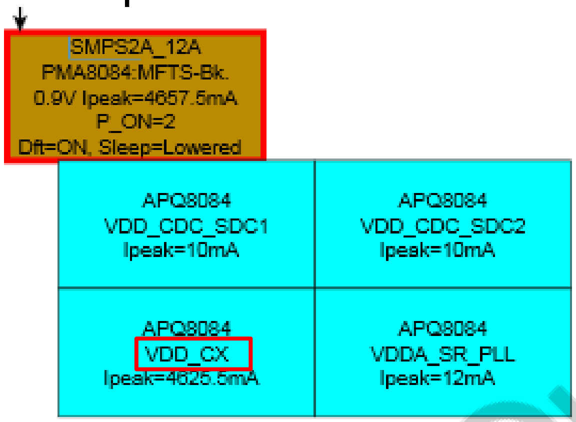
Customer device		
VIN_S1	3.7	14.83931203
VIN_S2_S12	**3.7**	**79.09674873**
VIN_S3	3.7	33.10164088
VIN_S4	**3.7**	**63.94561014**
VIN_S5	3.7	8.224414208
VIN_S6_S7	3.7	1.53402721
VIN_S8_TO_S11	3.7	20.5990193

Reference device		
SMPS1A_input	3.681216	9.784111977
SMPS2A_input	**3.680652**	**40.91268158**
SMPS12A_input	3.681581	3.099509954
SMPS3A_input	3.680847	21.16083908
SMPS4A_input	**3.680409**	**12.72926044**
SMPS5A_input	3.679405	43.97269821
SMPS6A_input	3.681721	1.057733059
SMPS7A_input	3.681741	-0.075547047
SMPS8A_input	3.681626	21.77234077
SMPS9A_input	3.681826	0.960167885
SMPS10A_input	3.681819	0.122583598
SMPS11A_input	3.681805	0.203078002

- Rail information can be obtained from the power tree published in the power app note for a chipset
  - A major power difference has been found on the S2 and S4 rails
  - Need to determine the components on these rails causing the power increase
  - Collect clock dumps to see the clock frequencies of all components

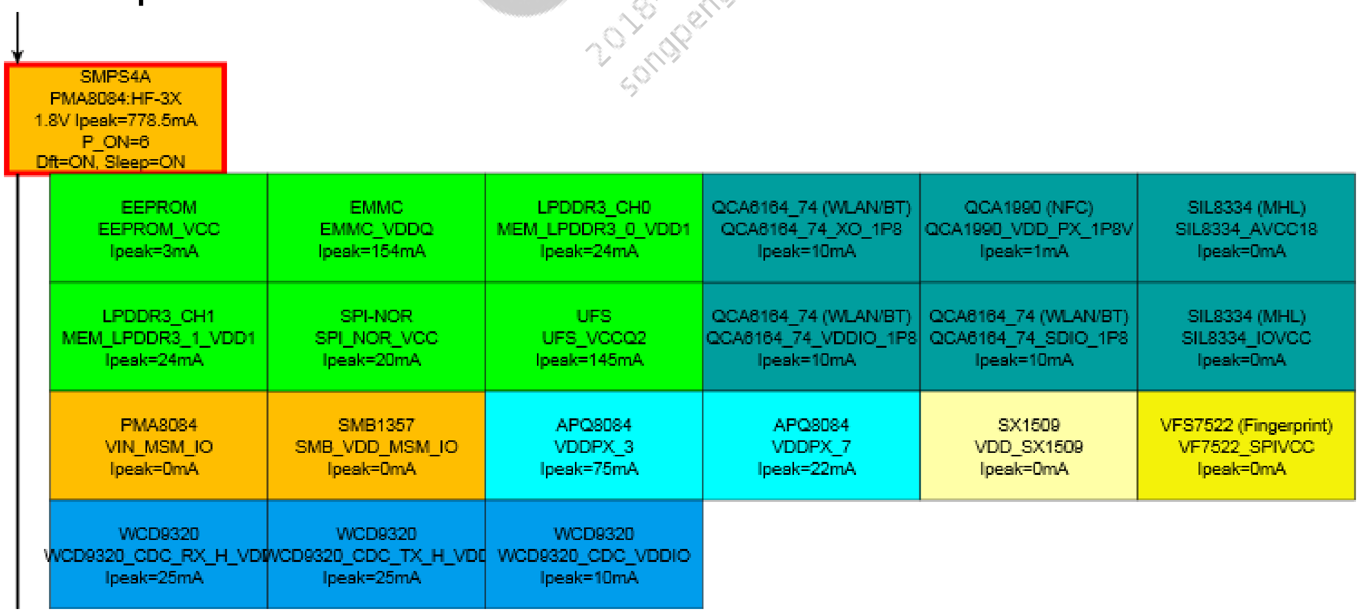
Identifying Power Rails Using a Power Tree

- SMPS2A powers CX rail



- SMPS2 is VDD_CORE (CX) rail source
- SMPS4 powers the LPDDR, EMMC, etc.
  - BIMC manages the clock for DDR, hence look for BIMC power consumption to know DDR consumption

- SMPS4A powers audio codec and DDR



Identifying Power Rails Using a Power Tree (cont.)

Component on CX	Clock name	ADB command for component	JTAG command

MDP	mdss_mdp_clk	`Adb shell`
		`Do cat /d/clk/<clock name>`
		`Sleep .1`
		`Done`
		
		Example:
		`Adb shell`
		`Do cat/d/clk/bimc_clk`
		`Sleep .1`
		`done`
			Provides active clock values for all system components
			
			From RPM window, run `testclock.cmm`
			
			From `\core\systemdrivers\clock\scripts`
			
			See *MSM8974 Linux Android Power Debugging and Optimization Guide* (80-NA157-246) for details

Venus	venus0_vcodec0_clk		
CAMSS	camss_vfe_cpp_clk		
MMSS NOC	mmss_mmssnoc_axi_clk		
SNOC	snoc_clk		
PNOC	pnoc_clk		
BIMC	bimc_clk		
LPASS		Cannot be collected using adb; an NPA dump is needed for voting and active frequency	

Capturing the Clock Dump

- Collect enabled clocks from the devices available using the following script in the adb shell interface:

```
Adb shell
sleep 5 && while true; do echo
\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=\=
\=\=\=;cat /proc/uptime; cd /sys/kernel/debug/clk; for i in *; do if [ -d $i ]; then if [
"$(cat $i/enable)" == "1" ]; then if [ -e $i/measure ]; then echo $i \=\> enable:`cat
$i/enable` measure:`cat $i/measure`; else echo $i \=\> enable:`cat $i/enable` rate:`cat
$i/rate`; fi; fi; fi; done;echo \-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-
\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-; cd /sys/class/regulator; for i in *; do if [ -d $i
]; then if [ -e $i/state ]; then if [ "$(cat $i/state)" == "enabled" ]; then if [ -e
$i/microvolts ]; then echo $i \=\> name:`cat $i/name` state:`cat $i/state` microvolt:`cat
$i/microvolts`; else echo $i \=\> name:`cat $i/name` state:`cat $i/state` microvolt: N\/A;
fi; fi; fi; fi; done; sleep 2; done > /data/dumpclk.txt &
```

- The method above provides the enabled clocks in the system.

Capturing the Clock Dump (cont.)

- The BIMC clock was higher than the internal reference value from the use case document.

```
================================================
932.69 1730.78
acpu_aux_clk => enable:1 rate:300000000
axi_clk_src => enable:1 rate:19200000
bimc_a_clk => enable:1 rate:99942400
bimc_clk => enable:1 measure:393603039
bimc_msmbus_a_clk => enable:1 rate:99942400
bimc_msmbus_clk => enable:1 rate:18937500
cxo_lpm_clk => enable:1 rate:1000
div_clk1 => enable:1 rate:1000
```

bimc_a_clk is not high, so the BIMC high vote is not from apps

Reference value for BIMC is 50 MHz compared to the 393 MHz observed

Clocks	Clock speed (MHz)
Krait	300
BIMC	50
MDP	150
Venus HD codec	133
MMSS NoC	100
System NoC	100
Peripheral NoC	19.2
Config NoC	19.2
OCMEM	37.5
OCMEM NoC	37.5
AUDIO_ss	19.2
AUDIO_ss_Q6	163.84

- Bimc_a_clk is used when the CPU is not power collapsed and bimc_clk is used when the CPU is power collapsed.
- The BMIC clock is not high, so the high vote is not from apps.
- To find non-apps-related components influencing the CX current and causing the higher BIMC clock, use NPA dumps.

Analyzing BMIC Clocks

- To analyze bus client votes, use bus client votes and NPA dumps

Usage:
Bus client votes

```
adb shell
cd /d/msm-bus-dbg/client-data
cat bimc_clk
```

Sample:

```
curr    : 2
masters: 22  23
slaves : 512  512
ab      : 1418342400  1418342400
ib      : 1418342400  1418342400
```

- The above information shows the last vote for BIMC; see *MSM8974 Linux Android Power Debugging and Optimization Guide* (80-NA157-246).
- This only provides information about the apps processor.
- NPA dumps
  - Using JTAG, we can collect the bus bandwidth requests for all subsystems like Apps, LPASS, Modem, etc., on the system
  - Sample:

```
npa_resource (name: "/clk/bimc") (handle: 0x19b020) (units: KHz) (resource max: 796800) (active max:
796800) (active state: 393600) (active headroom: -422008) (request state: 374792)
            npa_client (name: LPASS) (handle: 0x11e7b0) (resource: 0x19b020) (type:
            NPA_CLIENT_LIMIT_MAX)   (request: 4294967295)
            npa_client (name: LPASS) (handle: 0x11e770) (resource: 0x19b020) (type:
            NPA_CLIENT_REQUIRED) (request: 374792)
            npa_client (name: APSS) (handle: 0x19cc58) (resource: 0x19b020) (type:
            NPA_CLIENT_REQUIRED) (request: 122414)
            npa_client (name: ICB Driver) (handle: 0x1988b0) (resource: 0x19b020) (type:
            NPA_CLIENT_REQUIRED) (request: 185524)
            end npa_resource (handle: 0x19b020)
```

NPA Dumps

- Collecting an NPA dump
  - The NPA log is part of the RPM dump.
  - Use the JTAG T32 window to execute the following command:

```
rpm_proc\core\bsp\rpm\scripts\rpm_dump.cmm
```

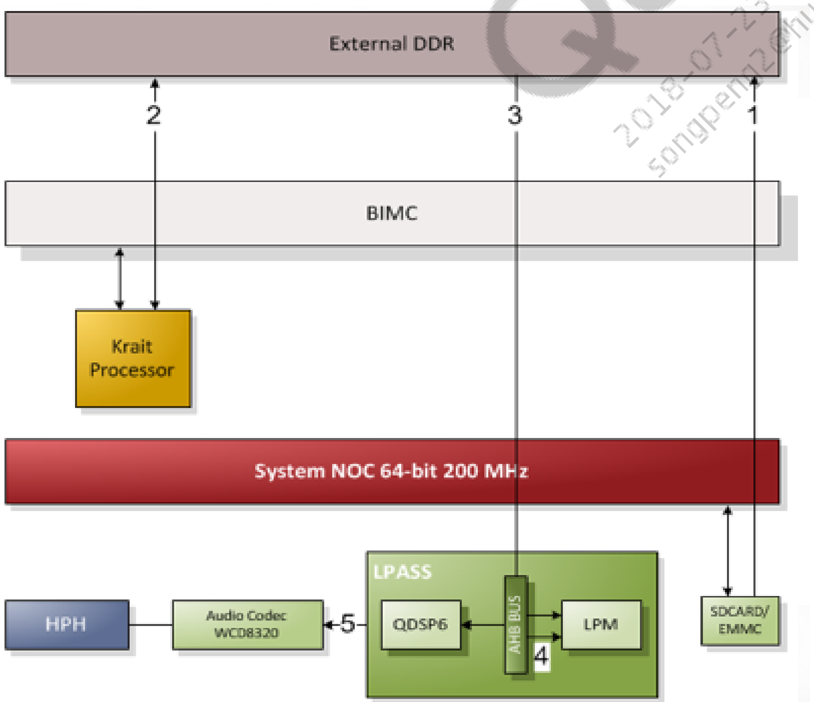
- View the collected NPA logs to see the subsystem voting for the high BMIC clock.
- From the NPA logs, it is found that LPASS is voting for higher bandwidth:

```
npa_resource (name: "/clk/bimc") (handle: 0x19b020) (units: KHz) (resource max:
796800)   (active max: 796800) (active state: 393600) (active headroom: -422008)
(request state: 374792)
   npa_client (name: LPASS) (handle: 0x11e7b0) (resource: 0x19b020)
   (type: NPA_CLIENT_LIMIT_MAX) (request: 4294967295)
   npa_client (name: LPASS) (handle: 0x11e770) (resource: 0x19b020)
   (type: NPA_CLIENT_REQUIRED) (request: 374792)
   npa_client (name: APSS) (handle: 0x19cc58)(resource: 0x19b020)
   (type: NPA_CLIENT_REQUIRED) (request: 122414)
   npa_client (name: ICB Driver) (handle: 0x1988b0) (resource: 0x19b020)
             (type: NPA_CLIENT_REQUIRED) (request: 185524)
   end npa_resource (handle: 0x19b020)
```

- LPASS is the root cause of high BMIC frequency.

Why LPASS in Video Playback?

- The diagram shows that LPASS is involved in audio processing during video playback.
- LPASS requests DDR bandwidth and can influence BIMC frequency.
- The customer issue was isolated to audio processing and does not exist when video clips without audio are used.
- LPASS/audio bandwidth voting code must be checked to see LPASS bandwidth voting.



1. CPU reads 1 MB of data from the SDCARD/EMMC to the DDR.
2. CPU parses the data and puts it in the DDR.
3. LPASS sets up DM-Lite to transfer 32 KB of bit stream from the DDR to LPM.
4. LPASS performs decoding and postprocessing and puts a PCM sample into LPM.
5. LPASS sets up the DMA to render the PCM sample to the audio codec.

Debugging High Current Due to Audio

- High current consumption for audio can be caused by postprocessing methodologies employed by customers.
- In this case, the customer had Dolby enabled. Disable Dolby using the QACT and QPST tools and note the power difference. For details on these tools, see:
  - QACT – *Installer, Qualcomm Audio Calibration Tool 5.0 Release 00000* (72-NT688-1)
  - QPST – *Installer, QPST 2.7 Version 00424* (72-V1400-160)

Disabling Dolby Topology

1. Run QPST and make sure the device is connected.
2. Run QACT and connect to the phone.
3. Click **File→Save** to save existing ACDB files in the phone.
4. Disconnect from QACT and restart it.
5. Click **Open File** and open the workspace file (.qwsp).
6. Click **Tools→Device Designer** and select the following devices:
   - BT_A2DP_SPKR
   - HEADSET_SPKR_STEREO
   - HEADSET_SPKR_ANC_STEREO_FF
   - HEADSET_SPKR_ANC_STEREO_FB
   - VIRTUAL_IN_CALL_MUSIC_DELIVER_RX
   - VIRTUAL_A2DP_RX
   - USB_RX
   - SPKR_PHONE_SPKR_STEREO

Disabling Dolby Topology (cont.)

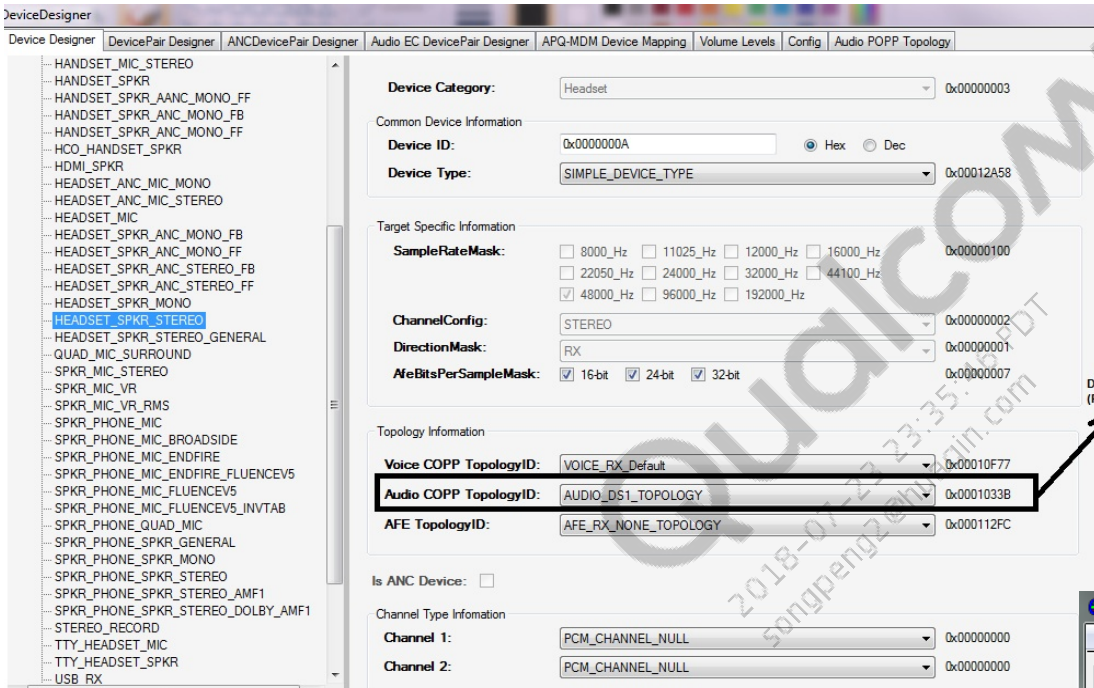
7. Check if the audio COPP topology ID is AUDIO_DS1_TOPOLOGY. If it is, change it to AUDIO_RX_STEREO.
8. Click **File→Save** to save all files corresponding to these devices.
9. Push these ACDB files to the device using the following commands:

```
Adb push <directory where modified ACDB files are present> /etc/
Adb shell sync
Adb reboot
```

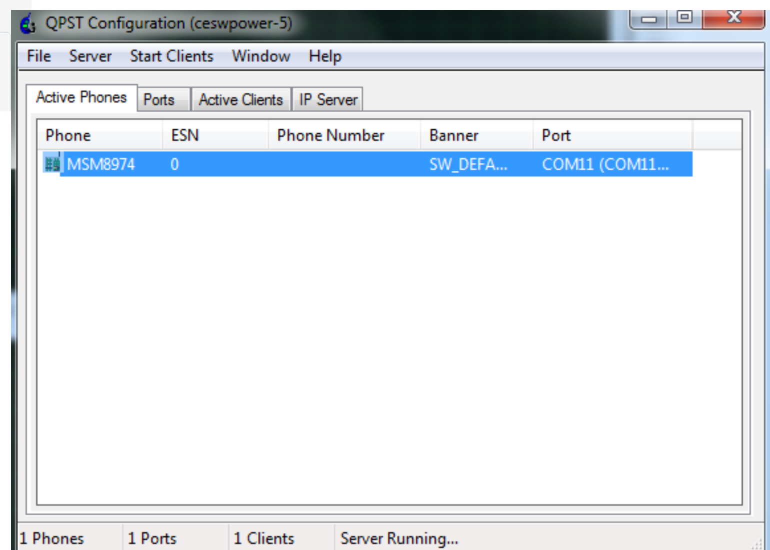
10. If QPST does not detect the device, run the following commands:

```
Adb shell
Setprop persist.sys.usb.config mtp.diag.adb
```


Using QACT and QPST to Change ACDB Files



DS1 Topology enable (Post-Processing)



Dolby Power Impact

- Now that Dolby is disabled on the customer device, a power reduction is observed.

Software configuration of device	BIMC frequency
Customer ADSP image + Dolby topology enabled	393 MHz
Customer ADSP image + Dolby topology disabled	150 MHz

Setup	Customer build current (mA)	QTI ADSP + Dolby topology disabled current (mA)
Customer device qtc77 playback	580	495

- Though disabling Dolby reduced the current, there is still some difference between the target and the observed current, 495 – 425 = 70 mA.
- Further debugging is needed.

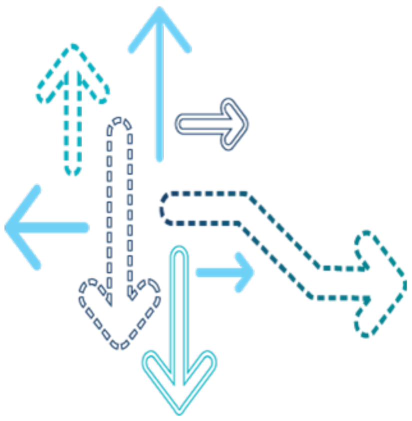
Debugging ADSP Bandwidth Voting

- RAM dumps and QXDM Professional™ logs are used by the audio debugging team to further analyze the difference in current.
  - Contact the ADSP team for specific configuration requirements for collecting these dumps to analyze audio issues.
- After the audio team analyzed the RAM dumps and QXDM Pro logs from the customer device, a power management bandwidth vote patch that was released by QTI was found to be missing from the customer device.

Setup	Dolby enabled (mA at battery)	Dolby disabled (mA at battery)	Fixed ADSP image + Dolby disabled (mA at battery)
Qtc77 video (customer device + customer build)	590	500	425

- Resolution – The customer was given the image with the ADSP bandwidth voting changes to decrease the power consumption.

MP3 Playback Debugging



Debugging the MP3 Playback Use Case

Step	Action	Reference
1	Compare the power of your device with the QTI reference power data.	▪ *MSM8994 Linux Android Current Consumption Data* (80-NJ051-7) ▪ *MSM8936/MSM8939 Linux Android Current Consumption Data* (80-NA157-246)

2	Tunnel mode is recommended. If Tunnel mode playback is not working, it will result in high current consumption. To identify if Tunnel mode playback is enabled, run the command:
	`adb logcat > c:\<your directory>\logcat.txt`
	If you can see the log below, Tunnel mode is not enabled:
	`D/AudioPolicyManager( 227): copl: offload disabled by audio.offload.disable=1`
	In this case, enable Tunnel mode by running the following commands:
	`adb pull /system/build.prop  <your directory name on PC>`
	`Add audio.offload.disable=0 to the build.prop file`
	`adb push <yourdirectory name on PC>\build.prop /system`	

3	Check the waveform and look for frequent wake-ups or a high baseline current.	
3.1	In the case of frequent wake-ups, frequent interrupts can prevent a device from entering idle power collapse. To debug this further, PowerTop, Top, and FTrace logs and interrupts are necessary.	▪ *Presentation: Android Debugging Overview (80-NT615-1)*
3.2	In the case of high baseline current, capture the full breakdown and compare it with the mp3 breakdown. Determine if a major power rail, e.g., CX, MX, DDR, or CPU, is consuming more current than the reference mp3 breakdown. Check MX and CX voltage levels. If DDR/MX is consuming more current, read/write or extra logging may be occurring. Using the clock dump, determine if the level of clocks is too high. For example, if the MSS Hexagon clock is running higher than expected, bump up the digital core voltage (VDDCX) to the next voltage level. This will result in high current consumption. Determine if extra audio postprocessing is enabled. This results in an increase in CX voltage and hence an increase in baseline current. Extra audio postprocessing also results in a higher Hexagon core clock frequency. Disable audio postprocessing and rerun the test.	▪ *MSM8994 Linux Android Current Consumption Data* (80-NJ051-7) ▪ *MSM8936/MSM8939 Linux Android Current Consumption Data* (80-NM683-7)
4	If mp3 power issues are still unresolved, file a case with the wave form, Powertop, Top, FTtrace, captured power rail breakdown, and clock dump information included.	

Problem Definition

- The customer is reporting high current consumption for music playback in Offload mode using a third-party file manager application.
- Ask the customer about the player, clip, and procedure they are using. Recommend that the customer follow QTI's default procedure in *MSM8936/MSM8939 Linux Android Current Consumption Data* (80-NM683-7) and measure current consumption.
- Determine if the customer use case is different from the standard QTI use case.
  - For example, the customer is using a third-party app called File Manager to launch the music player. This is not part of the standard procedure.
  - In this case, the power of the customer device cannot be compared to the QTI reference data.

Customer use case current (mA)	QTI reference current (mA)
46	21

Check Offload (Tunnel Mode)/Non-Offload (Non-Tunnel Mode)

- Use of Tunnel mode is recommended. If Tunnel mode playback is not working, it will result in high current consumption.
- To check the mode of playback, capture userspace logs (logcat) using the following command:

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

- Look for the following messages:
  - Non-offload

```
08-29 09:55:12.683   239   825 D audio_hw_primary: out_set_parameters: enter: usecase(0: deep-buffer-playback) kvpairs: routing=2
08-29 09:55:12.683   239   825 D audio_hw_extn: audio_extn_set_anc_parameters: anc_enabled:0
08-29 09:55:12.683   239   825 V listen_hw: listen_hw_set_parameters: Enter, kvpairs = routing=2
```

Non-offload mode is enabled

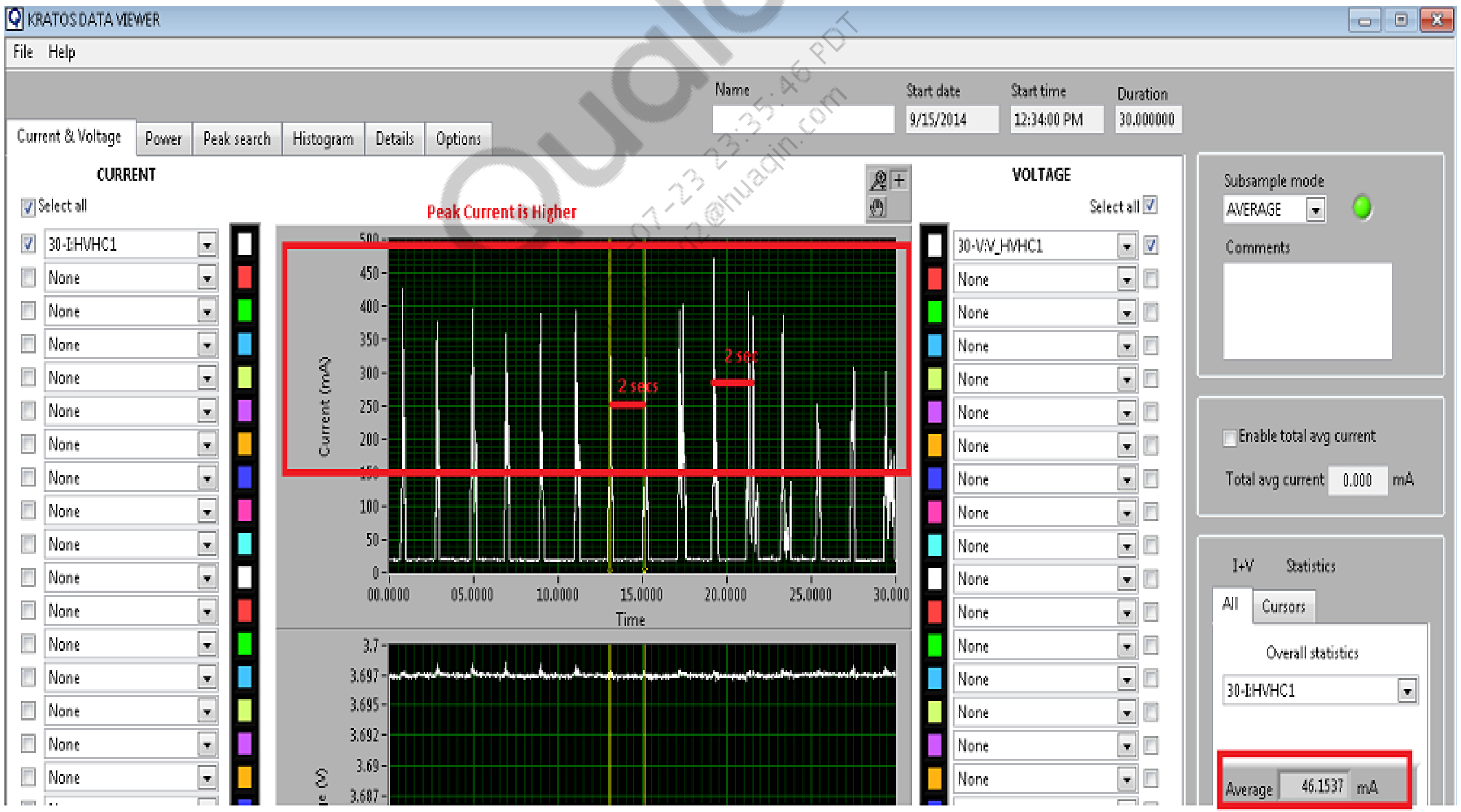
  - Offload

```
01-02 00:55:15.798 V/SRS_ProcWS( 208): SRS_Processing - SourceOutAdd - No Available Slot for 0xb7d06790
01-02 00:55:15.808 D/audio_hw_primary( 208): out_set_parameters: enter: usecase(3: compress-offload-playback) kvpairs: music_offload_avg_bit_rate=128000;music_offload_sample_rate=44100
01-02 00:55:15.818 D/audio_hw_primary( 208): out_set_parameters: enter: usecase(3: compress-offload-playback) kvpairs: routing=4
01-02 00:55:15.818 I/MediaFocusControl( 1042): AudioFocus requestAudioFocus() from android.media.AudioManager@41bcebc8com.android.music.AudioPreview$2@41b6e168
```

offload mode is enabled

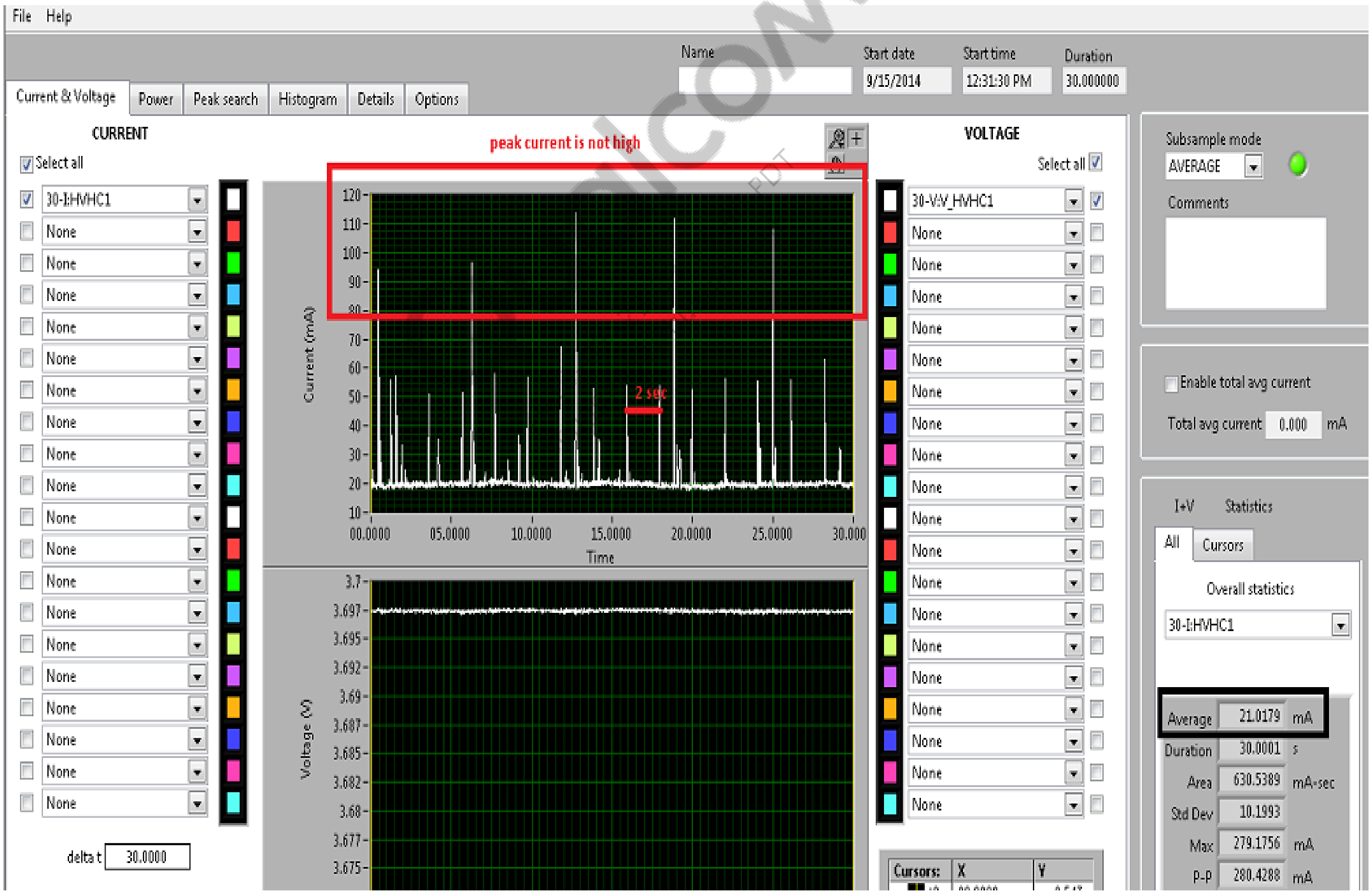
Wave Form Comparison

- The current wave form for the use case showed higher peak current with the customer setup (mp3 Offload mode playback using file manager).
- Wake-up is occurring every 2 sec (apps to DSP), confirming the mp3 is being played in Offload mode.



Wave Form Comparison (cont.)

- As shown in the graph below, peak current is not high for mp3 playback in Offload mode using the stock Android music app (Music.apk).



Apps Side Debugging to Determine the Root Cause of High Peak

- The following logs must be analyzed:
  - PowerTop
  - FTrace
  - Wakelock

PowerTop Analysis

- To capture PowerTop logs, run the following command:

```
adb shell sleep 10 –d powertop > /data/powertop.txt &
```

- As shown in the data below, wake-ups from idle and smd_dev interrupts are occurring more frequently in the right side image than the left.
- To find the root cause of why more smd_dev interrupts are occurring, analysis of FTrace logs using a pytime chart is necessary.

MP3 Offload mode playback using stock android music app

```
Wakeups-from-idle per second : 43.4      interval: 15.0s

no ACPI power usage estimate available

Top causes for wakeups:
  35.8% ( 15.5)        <interrupt> : arch timer
  26.9% ( 11.7)        <interrupt> : smd dev
```

MP3 Offload mode playback using File Manager

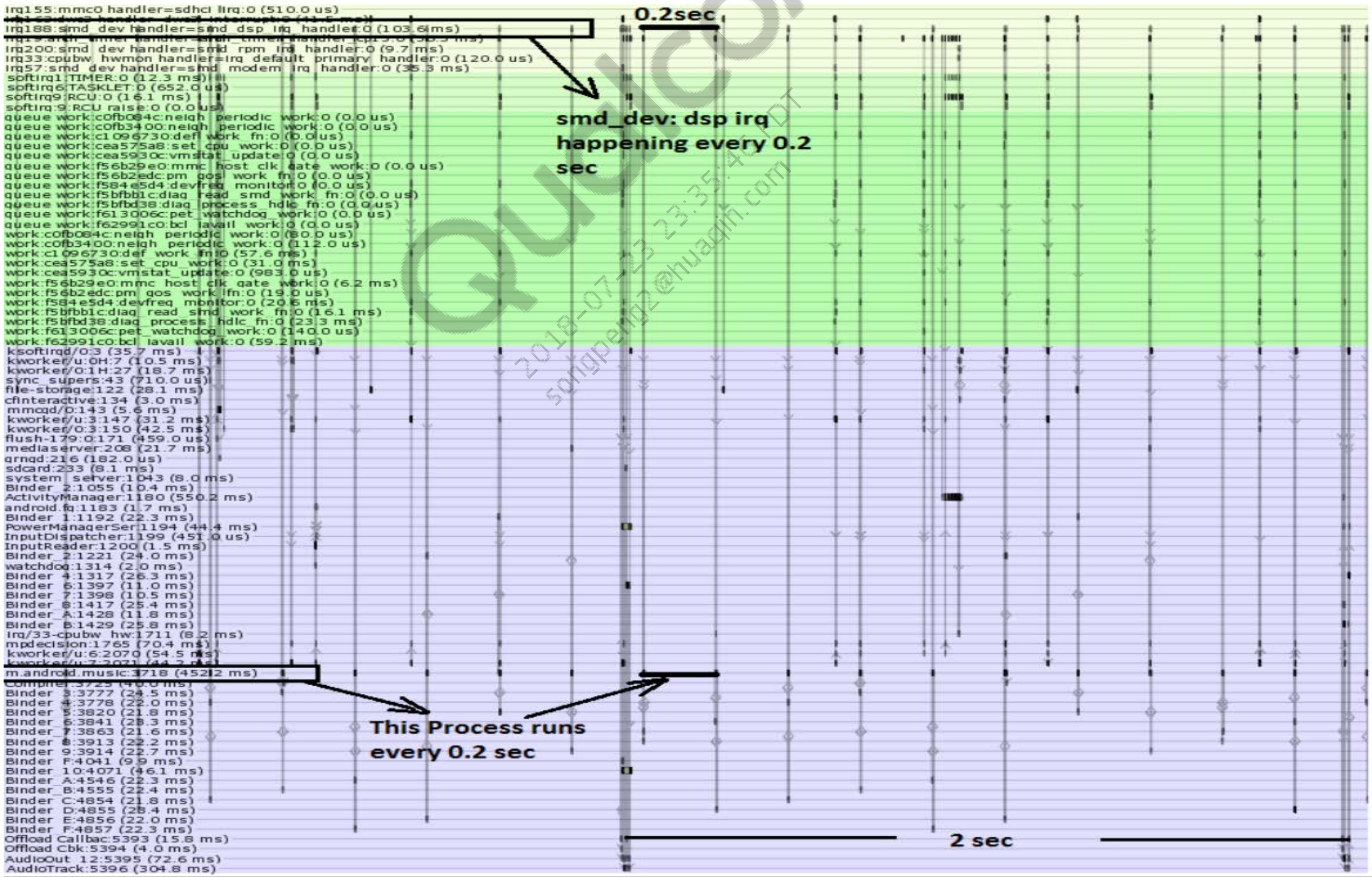
```
Wakeups-from-idle per second : 156.9     interval: 15.0s

no ACPI power usage estimate available

Top causes for wakeups:
  37.6% ( 59.0)        <interrupt> : arch timer
  35.6% ( 55.9)        <interrupt> : smd dev
```

FTrace Analysis

- During Offload mode mp3 playback using the default Android music app, smd_dsp_irq is occurring every 2 sec, which is the expected correct behavior. This occurs when apps is giving data to the DSP.

FTrace Analysis (cont.)

- During Offload mode mp3 playback using File Manager, smd_dsp_irq is occurring every 0.2 sec, which is not correct behavior. This IRQ is triggered because of the m.android.music process.



FTrace Analysis (cont.)

- Offload mode mp3 when played using File Manager apk:
  - m.android.music process runs every 0.2 sec
  - This causes 1rq188 smd_dev (dsp irq handler) interrupts to be fired every 0.2 sec.
- The above pattern was not observed while playing an mp3 in Offload mode using the default music player in stock Android.

Player	mp3 (Offload mode playback)		
	Batt (mA) at 3.7 V	CPU wakeup from idle	smd_dev interrupts
Music.apk (default)	21	43 times/sec	26.9 % (11.7)
Launch music.apk from File Manager	46	156 times/sec	35.6% (55.0)

- This behavior occurs because the app fails to recognize that the screen is off and continues to query the timestamp for wakelock.
- Wakelocks must be checked to see if this is the root cause.

Wakelock Analysis

- To collect wakelocks, run the following commands:

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

or

```
adb shell
dumpsys power
```

- The message “PowerManagerService.WakeLocks: ref count=1” indicates that the Power Manager Service wakelock is being held.
- The same behavior should be occurring using the File Manager app, which was not holding any wakelock.

Wakelock Analysis (cont.)

- Wakeup_sources and dumpsys power output

wakeup_sources output for mp3 plyback in offload mode using stock android music app

name	active_count	event_count	wakeup_count	expire_count	active_since	total_time	max_time	last_change	prevent_suspend_time
PowerManagerService.WakeLocks	554	554	72	0	88027	663769	145813	1328699	514651
msm_dwc3	7	7	3	0	496748	1276825	496748	919980	866649
power-supply	48	130	1	0	0	1444	186	920540	1085
diag_nrt_wcnss_read	0	0	0	0	0	0	0	2312	0

To check wakelock always look for active_since

Power Manager Service wakelock held by music app

msm_dwc3 wakelock held because of USB connected

dumpsys Power output for mp3 playback in offload mode using stock android music app

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

Power manager Service wakelock held by music app

Resolution

- After holding the test wakelock, current consumption for mp3 playback in Offload mode using File Manager was reduced and aligned with the Android stock music app.
- To test held wakelock:

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

- To remove test wakelock:

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

- Conclusion − The customer must modify their music player app to hold a wakelock when an mp3 is played and the display is off. In this case, the customer app is File Manager.

Filing a Case/SR

Information Requested from Customers in SRs

- Cases should only be filed if the suggested debugging methodologies/case studies do not resolve customer issues.
- When an SR is filed, the following information must be included:
  - Correct chipset – AMSS build ID and Operating System (OS)
  - Initial problem type – Software
  - Problem area 1 – Multimedia
  - Problem area 2 – Power
  - Problem area 3 – Case specific
    - Select audio, video, graphics, browsing, or sensors
  - Problem description – Provide detailed information about the problem
    - Details of the use case if different from QTI standard use case
    - Steps to reproduce the issue
    - Information about debugging performed by the customer
  - Logs
    - Top, PowerTop, rail level breakdown, clock dump, SurfaceFlinger, FTrace, systrace, wave forms, NPA dumps, and other logs captured

Backup

FTrace

- To collect FTrace for all providers:

```
echo 16384 > /d/tracing/buffer_size_kb && echo "" > /d/tracing/set_event && echo "" > /d/tracing/trace && echo "irq:* sched:* power:* msm_low_power:*" > /d/tracing/set_event && sleep 60 && cat /d/tracing/trace > /data/local/trace.txt &
```

- FTrace logs showed bmic_a_clk was scaling according to activity.
- The bimc_a_clk votes from FTrace show that it varies and is not high all the time:

```
irq/33-cpubw_hw-1545  [000] ...1   184.031270: clock_set_rate: bimc_a_clk state=149946368 cpu_id=0
kworker/u8:2-34    [002] ...1   184.184806: clock_set_rate: bimc_a_clk      state=99942400 cpu_id=2
kworker/u8:3-180   [000] ...1   184.358104: clock_set_rate: bimc_a_clk      state=74973184 cpu_id=0
irq/33-cpubw_hw-1545  [000] ...1   185.068369: clock_set_rate: bimc_a_clk   state=99942400 cpu_id=0
kworker/u8:3-180   [000] ...1   185.215232: clock_set_rate: bimc_a_clk      state=74973184 cpu_id=0
irq/33-cpubw_hw-1545  [001] ...1   186.590917: clock_set_rate: bimc_a_clk   state=99942400 cpu_id=1
```

- This indicates the BIMC clock is high due to some other subsystem on the CX rail and not the apps processor.
- The NPA log gives the votes of all the subsystems for all resources.
- Resolution – The above information and changes were provided to the customer. The power numbers met targets after the changes were applied.

Top

- Capturing Top

Tools	Output	When to use	How to use
Top	Cmd line text output shows CPU load for each process or thread	To identify the CPU load for each process or thread	adb shell top adb shell top -t

- Top showed no abnormal thread activity for video playback compared to the reference device.

```
User 8%, System 10%, IOW 0%, IRQ 2%
User 53 + Nice 0 + Sys 62 + Idle 478 + IOW 0 + IRQ 0 + SIRQ 13 = 606

  PID   TID PR CPU% S     VSS     RSS PCY UID      Thread          Proc
  207  3973  0   4% S 182652K  11412K  fg media    gle.aac.decoder /system/bin/mediaserver
  204   204  3   3% S 139500K  10476K  fg system   surfaceflinger  /system/bin/surfaceflinger
 4012  4012  0   2% R   1640K    756K     root     top             top
  207  3965  3   1% S 182652K  11412K  fg media    TimedEventQueue /system/bin/mediaserver
  207   818  3   1% S 182652K  11412K  fg media    AudioOut_2      /system/bin/mediaserver
  102   102  3   0% D      0K      0K     root     mdss_fb0
  207  3971  3   0% S 182652K  11412K  fg media    VideoDecMsgThre /system/bin/mediaserver
 3064  3064  3   0% S      0K      0K     root     kworker/3:1
  207  3972  3   0% S 182652K  11412K  fg media    OMXCallbackDisp /system/bin/mediaserver
  954  1093  0   0% S 611568K  47748K  fg system   ActivityManager system_server
```


PowerTop

- Installing PowerTop:

Tools	Precondition	Install
PowerTop	adb root adb shell mount –t debugfs none /sys/kernel/debug	adb shell su –c setenforce 0 adb push <powertop location>\powertop /data/ adb shell chmod 777 /data/powertop

- Capturing PowerTop:

Tools	Output	When to use	How to use
PowerTop	Cmd line output shows CPU residency information for each frequency and interrupt information	To verify CPU residency for each frequency and interrupt	adb shell /data/powertop –t (time)

- PowerTop showed no abnormal wake-ups or interrupts compared to reference device data.

```
Top causes for wakeups:
  66.3% (688.0)       <interrupt> : arch_timer
  17.6% (182.9)       <interrupt> : msm_vidc
   8.9% ( 92.0)       <interrupt> : smd_dev
   6.0% ( 62.2)       <interrupt> : MDSS
   0.6% (  6.5)       <interrupt> : mmc0
   0.5% (  5.5)       <interrupt> : cpubw_hwmon

Timer breakdown (dg_timer or gp_timer):
  19.3% ( 68.9)   TimedEventQueue : hrtimer_start_range_ns (hrtimer_wakeup)
  18.1% ( 64.6)         swapper/0 : hrtimer_start_range_ns (tick_sched_timer)
  16.6% ( 58.9)          DispSync : hrtimer_start_range_ns (hrtimer_wakeup)
  15.5% ( 55.3)         swapper/3 : hrtimer_start_range_ns (tick_sched_timer)
   8.4% ( 30.1)    mdss_fb_splash : hrtimer_start (event_hrtimer_cb)
   7.2% ( 25.5)         swapper/0 : hrtimer_start (lpm_hrtimer_cb)
   5.2% ( 18.6)        AudioTrack : hrtimer_start_range_ns (hrtimer_wakeup)
   2.8% ( 10.0)    ptt_socket_app : hrtimer_start_range_ns (hrtimer_wakeup)
   2.2% (  7.7)      boost_sync/3 : boost_mig_sync_thread (delayed_work_timer_fn)
   2.1% (  7.5)      boost_sync/0 : boost_mig_sync_thread (delayed_work_timer_fn)
   0.7% (  2.5)          mmcqd/0 : hrtimer_start (row_idle_hrtimer_fn)
```

SurfaceFlinger

- Capturing SurfaceFlinger

ADB command	Output	When to use	How to use
SurfaceFlinger	Text output shows layer-related information, e.g., size, rectangle, used pipe, allocated buffer, layer count, display panel type, etc.	To identify layer count, used pipe, updated rectangle per each layer	ADB shell dumpsys SurfaceFlinger

- SurfaceFlinger shows only one layer for full-screen video rendering and no additional activity compared to internal data.

```
numHwLayers=2, flags=00000000
   type    |  handle  |   hints  |   flags  | tr | blend |  format  |          source crop              |           frame           name
-----------+----------+----------+----------+----+-------+----------+-----------------------------------+-------------------------------
       HWC | b8c798d8 | 00000002 | 00000000 | 04 | 00100 | 7fa30c04 | [    0.0,    0.0, 1280.0,  720.0] | [    0,    0, 1080, 1920] SurfaceView
 FB TARGET | b8c75e78 | 00000000 | 00000000 | 00 | 00105 | 00000001 | [    0.0,    0.0, 1080.0, 1920.0] | [    0,    0, 1080, 1920] HWC_FRAMEBUFFER_TARGET
```

```
Qualcomm HWC state:
  MDPVersion=500
  DisplayPanel=8
HWC Map for Dpy: "PRIMARY"
CURR_FRAME: layerCount: 1 mdpCount: 1 fbCount: 0
needsFBRedraw: NO  pipesUsed: 1  MaxPipesPerMixer: 4
 ---------------------------------------------
```

References

Ref.	Document
Qualcomm Technologies, Inc.	
Presentation: Android Debugging Overview	80-NT615-1
MSM8916 Multimedia Power Solutions	80-NL239-62
MSM8994 Linux Android Current Consumption Data	80-NJ051-7
MSM8936/MSM8939 Linux Android Current Consumption Data	80-NM683-7
MSM8974 Linux Android Power Debugging and Optimization Guide	80-NA157-246
Installer, Qualcomm Audio Calibration Tool 5.0 Release 00000	72-NT688-1
Installer, QPST 2.7 Version 00424	72-V1400-160

Questions?

https://support.cdmatech.com