Linux Power Management Debugging Guide

80-VR629-1 C

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Revision History

Version	Date	Description
A	Jun 2009	Initial release
B	Jun 2009	Engineer updates
C	Feb 2010	Engineer updates

Contents

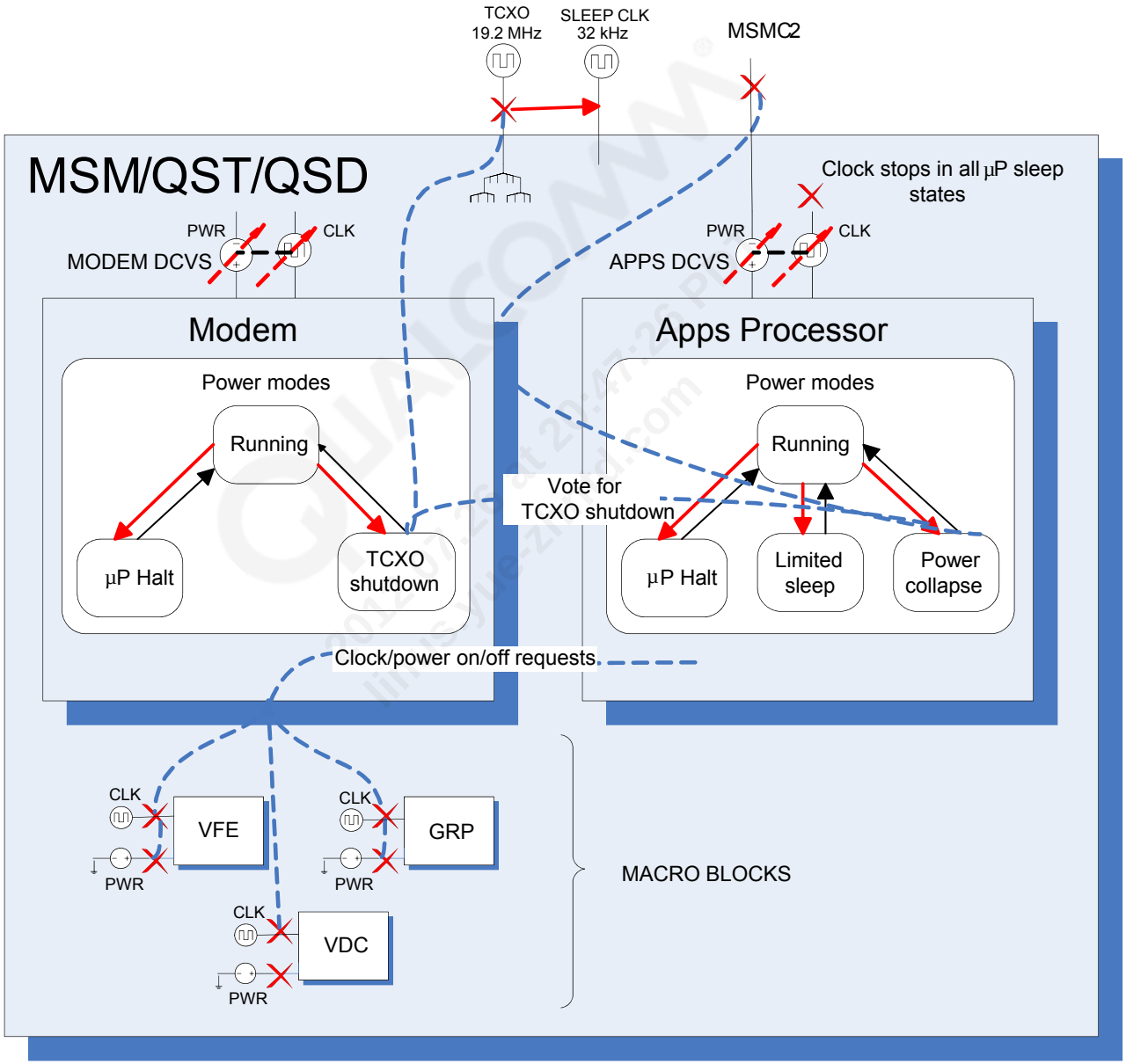
Introduction

Introduction

- Power Management (PM) for embedded devices is critical; battery life may be deciding factor for the end consumer
- MSM™/QSD chipsets have tight interaction between the modem and Apps Processor where the modem is the master
- Tight coupling leads to more complex PM implementation
- Basic power debugging and PM debugging issues
- Tools used for PM debugging
- Common conditions missed, making it seem like a power issue
- Troubleshooting related to PM
- Tips/tricks and steps useful for debugging PM issues, including taking logs
- For PM details and current optimization, see [Q2]

Overview

QSD/QST/MSM7xxx PM Overview



Linux Power Modes Supported

- Power modes
  - Suspend
    - Apps power collapse + modem TCXO shutdown + off state for all hardware devices (this is the maximum power-saving state and is where rock-bottom current is achieved)
  - Sleep (CPU idle)
    - MSM sleep – Apps power collapse + modem TCXO shutdown
    - Limited sleep (Apps power collapse only) – Apps power collapse + Apps votes against modem TCXO shutdown
    - SWFI (only) – Apps executes SWFI instruction; no Apps power collapse or modem TCXO shutdown
    - Spins – Apps spins

Note: Even after Apps power collapse, some drivers may not have not disabled their clocks, which will not let the modem go into TCXO shutdown, see Debugging Example 2 slide.

Tools Required

Tools Required

- Important tools required for PM debugging
  - SMEM logs capability
    - Should be able to take SMEM logs using TRACE32 or even ADB; see Tips for Power Debugging slide
  - TRACE32 capability
    - TRACE32 (for both modem and Apps) is required for debugging power issues
  - Console (preferably UART, ADB through USB should work, though there maybe dependency due to USB)
    - Console provides the flexibility of changing the SYSFS and makes for more efficient power debugging

Conditions to be Fulfilled/Prerequisites

Conditions to be Fulfilled/Prerequisites

- Some prerequisites required to be fulfilled before PM features are seen to be working
  - QCN file used for TCXO shutdown
    - Without a QCN file, the device can go into power collapse but will not go into TCXO shutdown.
    - The device must be Flashed with a QCN file for it go into TCXO shutdown.
  - USB should not be connected while doing suspend
    - If the USB is connected while trying to suspend the device, the Apps Processor will not go into power collapse since the USB driver holds a wakelock.
    - The USB is highly used since it is the ADB interface; it is used to check the sleep statistics, wakelocks, or even suspend the device.

Troubleshooting

How to Suspend the Device

- Suspending the phone
  - Enter the following command from user space to suspend the phone:
    - *echo mem > /sys/power/state*
  - Device can go into suspend based on a timeout
    - In Android™, there is a Screen Timeout option in the UI. Go to Settings → Sound and Display → Screen Timeout. The screen timeout can be set here to range from 15 seconds to never timeout.
  - Device can go into suspend using the END/Power keypress
    - See [Q2] for details.

How to Tell Power Collapse and TCXO Shutdown

- On SURF™:
  - ARM11™ power rail is tied to MSMC2 in MSM7xxx targets and Scorpion is powered from TI PMIC in QSD8x50 targets.
  - If the Apps is in suspend power collapse, the MSMC2 LED on the SURF turns off (this is only valid for MSM7xxx targets).
  - If the MSMC2 LED is Flashing, the Apps is going into idle power collapse.
  - To check Scorpion power collapse on QSD8x50 targets, voltage at TB3 pin 1, 2 can be measured.
  - If the SURF is in TCXO shutdown, the TCXO LED turns off (valid for all targets).

How to Tell Power Collapse and TCXO Shutdown (cont.)

- On customer device, there are four ways:
  - TRACE32 breakpoints
    - Apps power collapse
      - msm_pm_collapse() – Apps processor – This is the last function call on the Apps before going into power collapse, called from msm_pm_power_collapse() in pm2.c
      - clk_regime_apc_rail_off() – Modem – This function switches off the power rail to the Apps. After execution of this function, the Apps should be completely power collapsed.
    - Modem TCXO shutdown
      - clk_regime_tcxo_shutdown() – Modem – If this function is called, then the modem is ready to go into TCXO shutdown, i.e., all subsystems have voted for sleep.
  - ADB shell/console
    - "echo reset > /proc/msm_pm_stats" – Zeroes out the msm_pm_stats
    - "cat /proc/msm_pm_stats" – This prints out the all the sleep statistics which includes the number of times it went into suspend, idle power collapse, and SWFI (details on MSM_PM_STATS slides).

How to Tell Power Collapse and TCXO Shutdown (cont.)

- On customer device, there are four ways (cont.):
  - SMEM logs
    - Apps power collapse can be seen from SMEM logs, as illustrated below:

```
MODM: 378.738500  SMEM:  curr =                              TIN   PWRC
MODM: 378.740063  DEM: APPS SWFI  00000004  00010ca9  00000a29
MODM: 378.740375  DEM: PWRCLPS APPS  00000000  00000000  00000000
MODM: 378.740563  DEM: OKTS  00000002  00000000  00000000
```

    - TCXO shutdown can be seen in SMEM logs, as illustrated below:

```
MODM:  80.970313  DEM: OKTS  00000002  00000000  00000000
MODM:  80.970938  SLEEP:  ENTER TCXO  000c4d60  00000000  00000000
MODM: 157.401375  SLEEP:  TCXO END  00000000  00000000  00000000
```

    - Note that once the device goes into TCXO shutdown, TRACE32 connection is lost, but it is resumed when the device is woken up, which is when SMEM logs should be taken to see “SLEEP: TCXO END”.

How to Tell Power Collapse and TCXO Shutdown (cont.)

- On customer device, there are four ways (cont.):
  - Quick way through TRACE32
    - If the Apps goes into power collapse, the TRACE32 connection for the Apps goes from “running” to “running(timeout)”.
    - If the modem goes into TCXO shutdown, the TRACE32 connection for the modem goes from “running” to “running(timeout)”.
    - Running Daisy-chain mode prevents the modem TRACE32 debugging after Apps power collapse since the Apps TRACE32 loses connection.

Note: All of the above four methods are applicable to SURF/FFA.

MSM_PM_STATS

- idle-request
  - Count is the number of times the idle function is called.
  - Whenever the idle function is called, time to the earliest timer expiration is collected and sorted into the histogram. total_time is the sum of the earliest expiration time.
- idle-wfi
  - Count is the number of times SWFI without notification has occurred.
  - Time spent in SWFI without notification is collected and sorted into the histogram. total_time is the sum.

```
$ cat /proc/msm_pm_stats
cat /proc/msm_pm_stats
Clocks against last TCXO shutdown:
  i2c_clk (id=9)
Last power collapse voted for TCXO shutdown

idle-request:
  count:   50218
  total_time: 2651.754397730
   <      0.000062500:     135 (0-61666)
   <      0.000250000:     313 (63333-249999)
   <      0.001000000:    1168 (250000-999999)
   <      0.004000000:    4376 (1001666-3999999)
   <      0.016000000:   10764 (4000000-15998333)
   <      0.064000000:   16813 (16000000-63998333)
   <      0.256000000:   16647 (64000000-199621666)
   <      1.024000000:       0 (0-0)
   <      4.096000000:       2 (1738326666-1749549999)
  >=      4.096000000:       0 (0-0)
idle-spin:
  count:   39074
  total_time: 158.958848408
   <      0.000062500:   24080 (1666-61667)
   <      0.000250000:    2509 (63333-248334)
   <      0.001000000:    5175 (250000-998333)
   <      0.004000000:     714 (1000000-3996667)
   <      0.016000000:    2152 (4000000-15993334)
   <      0.064000000:    3786 (16000000-63996666)
   <      0.256000000:     658 (64141667-99800001)
   <      1.024000000:       0 (0-0)
   <      4.096000000:       0 (0-0)
  >=      4.096000000:       0 (0-0)
idle-wfi:
  count:   11003
  total_time: 11.218694987
   <      0.000062500:       0 (0-0)
   <      0.000250000:    7549 (96667-248334)
   <      0.001000000:    2716 (250000-986667)
   <      0.004000000:      94 (1011667-3990000)
   <      0.016000000:     313 (4015000-15903333)
   <      0.064000000:     331 (16113333-19873334)
   <      0.256000000:       0 (0-0)
   <      1.024000000:       0 (0-0)
   <      4.096000000:       0 (0-0)
  >=      4.096000000:       0 (0-0)
```

MSM_PM_STATS (cont.)

- idle-sleep
  - Count is the number of times power collapse has occurred.
  - Time spent in power collapse is collected and sorted into the histogram. total_time is the sum.
- idle-failed-sleep
  - Count is the number of times power collapse was attempted but aborted, e.g., incoming interrupts, SMSM timeout.
  - Time spent is collected and sorted into the histogram. total_time is the sum.

```
idle-sleep:
  count:         0
  total_time: 0.000000000
    <       0.000062500:         0 (0-0)
    <       0.000250000:         0 (0-0)
    <       0.001000000:         0 (0-0)
    <       0.004000000:         0 (0-0)
    <       0.016000000:         0 (0-0)
    <       0.064000000:         0 (0-0)
    <       0.256000000:         0 (0-0)
    <       1.024000000:         0 (0-0)
    <       4.096000000:         0 (0-0)
  >=        4.096000000:         0 (0-0)
idle-failed-sleep:
  count:         0
  total_time: 0.000000000
    <       0.000062500:         0 (0-0)
    <       0.000250000:         0 (0-0)
    <       0.001000000:         0 (0-0)
    <       0.004000000:         0 (0-0)
    <       0.016000000:         0 (0-0)
    <       0.064000000:         0 (0-0)
    <       0.256000000:         0 (0-0)
    <       1.024000000:         0 (0-0)
    <       4.096000000:         0 (0-0)
  >=        4.096000000:         0 (0-0)
```

How to Disable Power Collapse

- Disabling power collapse
  - Suspend power collapse
    - In Android, there is a “Screen Timeout” option in the UI. Go to Settings -> Sound and Display -> Screen Timeout. The screen timeout can be set here to range from 15 sec to never timeout.
    - Disable Suspend Power Collapse:
      - Do “*echo 3 > /sys/module/pm*/parameters/sleep_mode*” in the console/ADB shell.
    - Suspend can be disabled from the kernel configuration options:
      - System Type --->
        ...Suspend sleep mode (Power collapse suspend)
      - Suspend Sleep mode can be set to “Wait for interrupt”
        Set the Idle Sleep mode to the same value in menuconfig

How to Disable Power Collapse (cont.)

- Disabling power collapse (cont.)
  - Idle power collapse
    - Disable SWFI and idle power collapse:
      - Do "*echo 4 > /sys/module/pm*/parameters/idle_sleep_mode"* in the console/ADB shell.
    - Disable only idle power collapse, SWFI still possible:
      - Do "*echo 3 > /sys/module/pm*/parameters/idle_sleep_mode"* in the console/ADB shell.
    - Idle power collapse can be disabled from the kernel configuration options:
      - System type --->

        ...Idle Sleep mode (power collapse suspend)
      - Here, Idle Sleep mode can be set to "Wait for interrupt"

Power Debugging Tips

Tips for Power Debugging

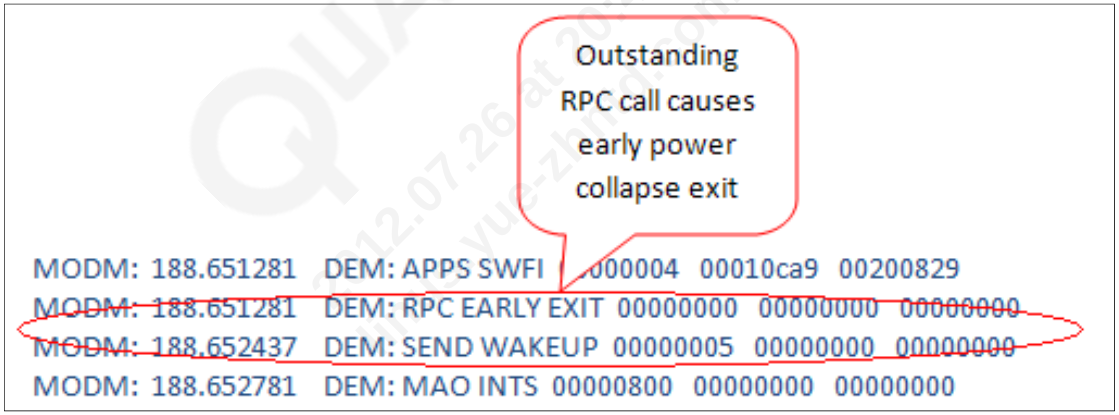
- Taking meaningful SMEM logs
  - Using TRACE32
    - SMEM logs maybe taken using the smemlog.cmm in AMSS/products/<asic>/tools/debug/. All debug symbols must be switched on for this.
      - » In AMSS/products/<asic>/build/ms/armtools.min, make sure "*DBG = -g --dwarf2*"

        ifeq ($(USES_NO_DEBUG),yes)

        DBG = -g --dwarf2
    - Example
      1. Perform a Suspend → Resume → Suspend cycle
      2. Take smem logs using TRACE32 on ARM9 by doing "do smemlog.cmm" in TRACE32
      3. Then on host computer shell do "perl smem_log.pl > smemlog.txt" to parse the smem logs into human readable text
  - Using ADB
    - *"cat /debug/smem_log/dump_sym"* (this requires DEBUG_FS and CONFIG_MSM_IDLE_STATS to be enabled in the kernel configuration)

Tips for Power Debugging (cont.)

- Apps power collapse debugging
  - Check if Apps is going into power collapse using one of the methods described in previous slides. If yes, proceed to the next step. If not, follow these steps for debugging:
    1. Take SMEM logs to see if there is any outstanding RPC call, as illustrated below.



    2. Check dmesgs to see if there is any error or a wakeup interrupt. Do the following to see detailed msgs:
       - *"echo 15 > /sys/module/irq/parameters/debug_mask"*
       - *"echo 127 > /sys/module/pm*/parameters/debug_mask"*
       - Compare these logs with the printk(s) in the code; see Debugging Example 1

Tips for Power Debugging (cont.)

- Apps power collapse debugging (cont.)
  - Check wakelocks by doing "*cat /proc/wakelocks*" – This prints all the wakelock information and shows which driver is holding the wakelocks.
    - The information below shows which driver is holding how many wakelocks including the time for which the wakelocks are held. For example, "alarm" is currently holding three wakelocks.

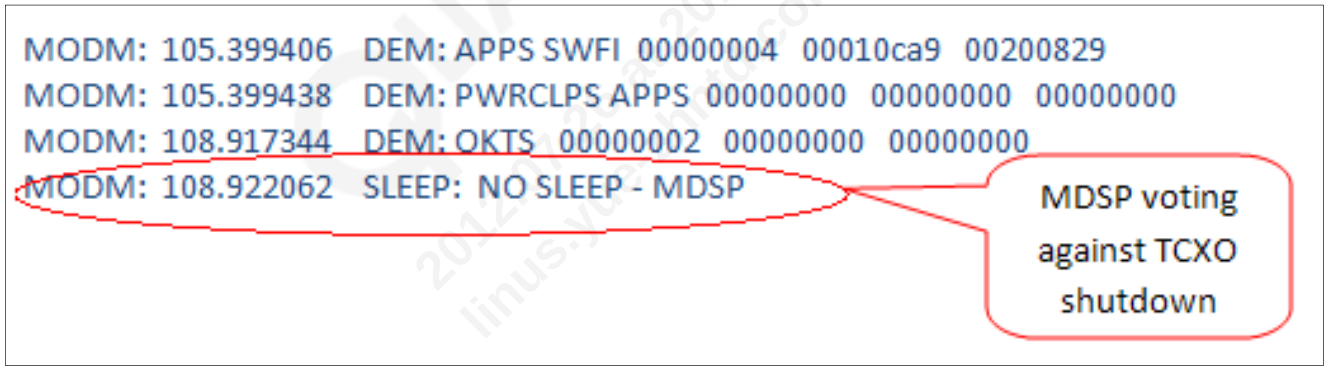
```
cat /proc/wakelocks
name    count   expire_count    wake_count      active_since    total_time      sleep_time      max_time        last_change
"SMD_RPCCALL"   244     0       0       0       306171660       5095001 62201667        346664261666
"rpc_server"    69      0       0       0       17385001        848333  4971667 346664184999
"usb_mass_storage"      0       0       0       0       0       0       0
"PowerManagerService"   13      0       0       0       9718311749      539501752       5552203335  299918808333
"KeyEvents"     15      0       0       0       518398334       9246666 282458333       299915261666
"evdev" 13      0       0       0       80243334        9598333 61620000        299914058333
"gpio_kp"       4       0       0       0       637379997       0       221428332       299913858333
"alarm" 3       0       0       0       67405001        36098336        31291666        297335469998
"audio_pcm_idle"        1       0       0       0       3745850001      0       3745850001      40572521666
"audio_pcm"     1       0       0       0       3745845001      0       3745845001      40572514999
"adsp"  1       0       0       0       3617913334      0       3617913334      40534888333
"DS"    0       0       0       0       0       0       0
"evdev" 0       0       0       0       0       0       0
"evdev" 0       0       0       0       0       0       0
"alarm_rtc"     0       0       0       0       0       0       0       0
"DATA7" 0       0       0       0       0       0       0
"DATA6" 0       0       0       0       0       0       0
"DATA5" 0       0       0       0       0       0       0
"msm_serial_hs_rx"      0       0       0       0       0       0       0       0
"qmi2"  0       0       0       0       0       0       0
"qmi1"  0       0       0       0       0       0       0
"qmi0"  0       0       0       0       0       0       0
"unknown_wakeups"       0       0       0       0       0       0       0       0
"deleted_wake_locks"    0       0       0       0       0       0       0       0
"mmc_delayed_work"      2       1       0       340995210000    341721198334    230658421667    340995210000    6336681666
"usb_bus_active"        3       2       0       42361390001     73966200001     0       42361390001 304970531665
"main"  2       0       0       49286135000     116423509999    0       67137374999     298045809999
"mass_storage_hold_idle"        2       0       0       41434386666     70978904999     0       41434386666     305897578333
```

Tips for Power Debugging (cont.)

- Apps power collapse debugging (cont.)
  - Breakpoint to set in the Apps processor
    - *msm_pm_collapse()* – This is the last function call on the Apps before going into power collapse, called from msm_pm_power_collapse() in pm2.c.
  - Breakpoints to set in the Modem
    - *demmod_swfi_isr()* on MSM7201A™ target, *AppsSwfiIsr()* on other MSM7xxx/QSD8x50 targets – This function is called when the Apps executes SWFI causing an A2M6 interrupt on the modem
    - *clk_regime_apc_rail_off()* – This function switches off the power rail to the Apps. After execution of this function, Apps should be completely power collapsed
  - Refer to [Q2] for detailed call flow diagram

Tips for Power Debugging (cont.)

- Modem TCXO shutdown debugging:
  - If Apps is going into power collapse, check if modem is going into TCXO shutdown using one of the methods described in previous slides. If not:
    - Take SMEM logs to see which modem subsystems are voting against sleep (In the SMEM logs, “OKTS” means ok to sleep and “NO SLEEP” means vote against sleep – as illustrated below).



    - See example in Debugging Example 2.

Tips for Power Debugging (cont.)

- Modem TCXO shutdown debugging (cont.):
  - Some breakpoints to set:
    - sleep_power_down_and_halt() – Modem: This function is called by sleep task to continuously check for sleep voters.
    - clk_regime_tcxo_shutdown() – Modem: If this function is called, then modem is ready to go into TCXO shutdown i.e. all susbystems have voted for sleep.
    - clk_regime_tcxo_shutdown_asm() – Modem: This is the last function called on the modem which actually executes the modem SWFI instruction.

Tips for Power Debugging (cont.)

- CPU frequency tips:
  - SYSFS interface – */sys/devices/system/cpu/cpu0/cpufreq/**
  - How to see current CPU frequency governor
    - *cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*
  - How to change governor
    - *echo <new_governor> > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*
    - Example
      - *echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*
      - Ondemand is a dynamic governor, the CPU frequency is adjusted based on CPU load; see [Q2]
      - Performance is a static governor, CPU runs maximum frequency; sometimes switching to this governor may mask cases where there is lack of CPU horsepower
  - How to change frequency in user space governor
    - *echo user space > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor*
    - *echo <new_freq> > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed*

Tips for Power Debugging (cont.)

- Output of testclock.cmm in the case of MSM7xxx targets (testclock2.cmm in the case of QSD targets); this script is located in AMSS\products\xxxx\build\ms
- Using this script, see all or required clock values and the PLL those clocks are using

```
---------------------------- MSM8600 Clock Test --------------------------------

*** Chip Clocks ***
   ad6_ahb          ad6_axi          ad6_dbg_ls       ad6_jtag         ad6_tcxo
   ad6_slp          adm              axi_arb          axi_ebi1         axi_imem
   axi_li_apps      axi_li_mss       axi_li_vg        axi_smi          btpcm
   cam_m            ce               codec_ssbi       dbg_clk_hs       dbg_clk_ls
   ebi1_async_2x    ebi1             ebi1_sync_2x     ebi2             ebi2_2x
   ecodec           ecodecif         emdh             fuse             gp
   grp              i2c              icodec_rx        icodec_rx_2x     icodec_tx
   imem_branch      imem_clk         io_cal           jtag_pc_dbg      lcdc_pclk
   lcdc_pclk_ext    mahb0            mahb1            marm             marm_etm
   mdc_io           mdc_vfe          mdp              mss_slp          pbus
   pcm              pmdh             pmic_ssbi        q6_core          qmembist
   ring_osc         sc_axi_src       scorpion         sdac             sdc1
   sdc2             sdc3             sdc4             sdc1_h           sdc2_h
   sdc3_h           sdc4_h           smi              smi_sync_2x      smi_async_2x
   spss_dbg_ls      tcxo4            tlmm             tsif             tsif_p
   tsif_ref         tv_dac           tv_enc           tx_sbi           tx_ssbi
   uart1            uart1dm          uart1dm_p        uart2            uart2dm_p
   uart3            usb_m            usb_s            usb_hs           vdc
   vfe

*** Modem Clocks ***
   bt                    bt_sbi                cc_ram                cdma_chipxn
   cdma_chipxn_div2      cdma_chipxn_div4      cdxo_at_cut           cdxo_y_cut
   edge                  edge_div2             f9                    fb_ant0
   fb_ant1               gacc                  gps_chipx8            gps_chipx16
   gps_chipx32           gsm                   mdm                   mdsp_hm
   mdsp_intf             modem_sbi             modem_web             offline
   pc_dac                rxf_samp              rxf_samp_ref          samp_fifo0
   samp_fifo1            sleep                 slpfast_cdma_chipxn_ref1
   slpfast_cdma_chipxn_ref2                    slpfast_fee           slpfast_gsm
   slpfast_wcdma_chipxn_ref                    ssrv_offline0         ssrv_offline1
   sym_buff              tcxo_pdm              tx_dac                tx_pdm
   tx_ram                wcdma_chipxn          wcdma_chipxn_div2     wcdma_chipxn_div4

*** Clock groups***                          *** Other options ***
   global      - all global sourced clocks       s       - status
   modem/_on   - all/on modem clocks             x       - exit
   all/_on     - all/on clocks                   ?       - help
   chip/_on    - all/on non-modem clocks         p       - write PLL test register
   <enter>     - repeat last command             t       - write SPSS_TESTCLK reg.


Clock (? for list):
```

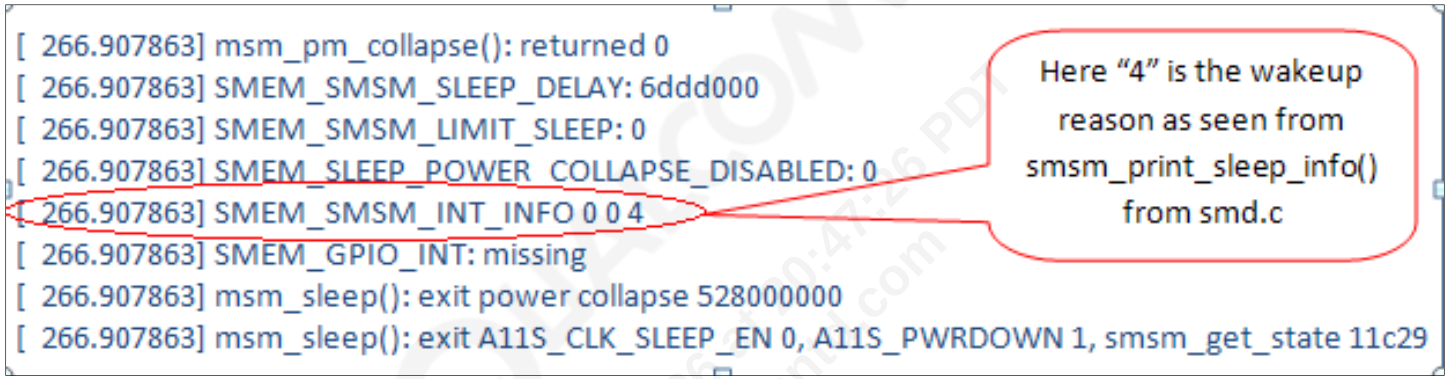
Debugging Examples

Debugging Example 1

- Example 1 – Apps does not go into power collapse
  - Test target – MSM7201A™
  - Test scenario – Trying to suspend the device
  - Test result – Sleep current is very high
  - Follow the debugging steps described in previous slides
    - Check if the Apps is going into power collapse by looking at the Apps TRACE32 status
      - If it still shows as “running”, then Apps is not going into power collapse
    - To get more detailed kernel logs:
      - *“echo 15 > /sys/module/irq/parameters/debug_mask”*
      - *“echo 127 > /sys/module/pm*/parameters/debug_mask”*

Debugging Example 1 (cont.)

- Example 1 – Apps does not go into power collapse (cont.)
  - Output shown below:



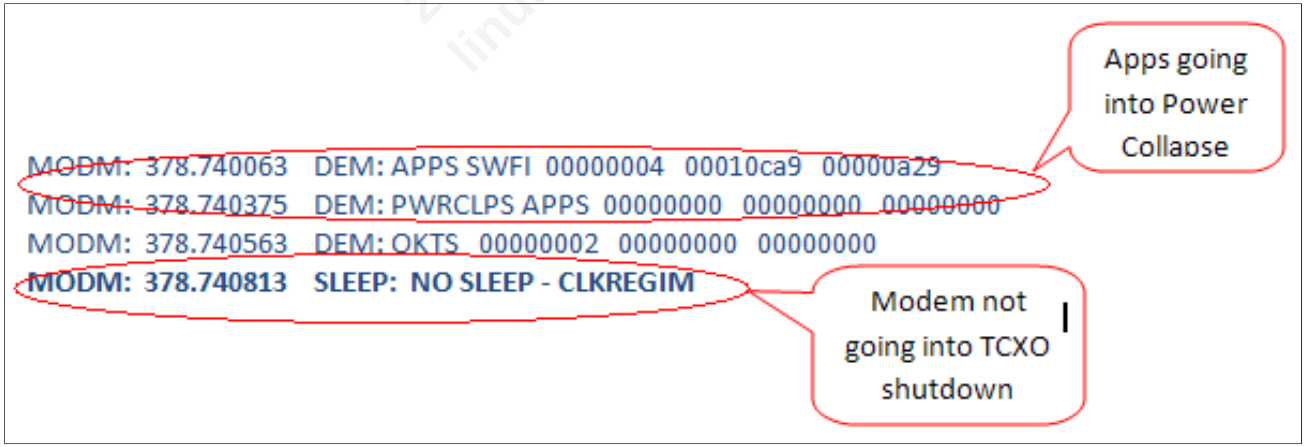
    - Look at smsm_print_sleep_info() in /kernel/arch/arm/mach-msm/smd.c
    - *printk(KERN_ERR "SMEM_SMSM_INT_INFO %x %x %x\n",*
      irq_mask, pending_irqs, wakeup_reason);
  - Wakeup reason "4" corresponds to GPIO interrupt wakeup from dem.h in AMSS/products/76xx/services/dem/
    - *#define DEM_WAKEUP_REASON_GPIO 0x00000004*

Debugging Example 1 (cont.)

- Example 1 – Apps does not go into power collapse (cont.)
  - Debugging in ARM9 T32, break at tramp_gpio_monitor_apps_isr() to see which GPIO interrupt fired to wakeup the Apps. For example, in this case the GPIO number is 40.
  - Since the GPIO responsible is now known, it can be debugged further to see the reason for that GPIO interrupt.
  - In this case, as Apps was trying to Power Collapse the GPIO interrupt would try to wakeup the Apps processor. Hence it could not complete its power collapse procedure leading to high sleep current.

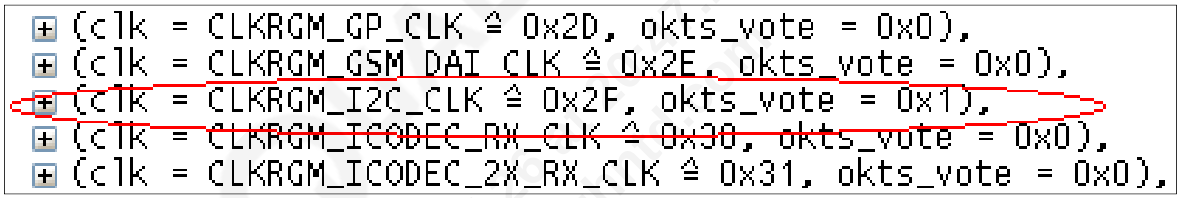
Debugging Example 2

- Example 2 – Modem does not enter TCXO shutdown
  - Test target – MSM7201A
  - Test scenario – Incoming call while the device is in suspend
  - Test result – Device wakes up as expected but sleep current after this MT call is higher than expected
  - Following the debugging steps described in previous slides:
    - Check if the Apps is going into power collapse through SMEM logs (illustrated below)
    - Check if Modem is going into TCXO shutdown (illustrated below)



Debugging Example 2 (cont.)

- Example 2 – Modem does not enter TCXO shutdown (cont.)
  - It is seen from the SMEM logs that Clock Regime is voting against TCXO shutdown
    - To find which clock(s) is voting against sleep, check clkrgm_linux.clk[] structure in ARM9 TRACE32 (illustrated below)

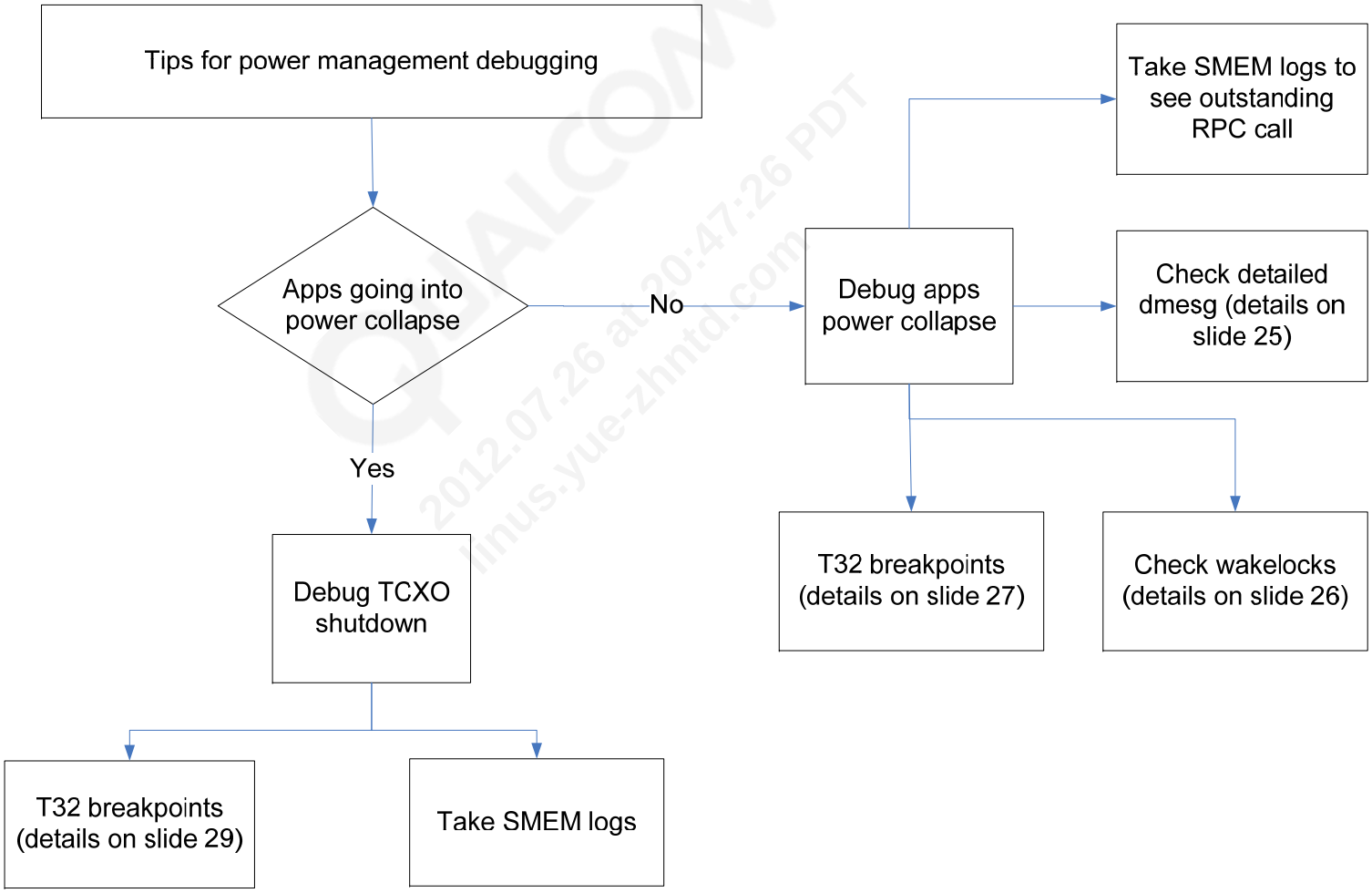


    - In this case, it is I2C clock which has not been disabled and it is some Linux driver which has not disabled its I2C clock. This narrows down the root cause to a Linux driver which should be using the APIs (see [Q2]) to disable its clock when not in use.

Note: This is a case when even though Apps is in power collapse, a kernel driver had not turned off its clock(s) in its suspend() which eventually caused Clock Regime to vote against TCXO shutdown. Hence, the root cause of modem not going into TCXO shutdown may not always on the modem side.

Summary

- Flow chart for power debugging



References

Ref.	Document	
Qualcomm		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Presentation: Linux Power Management Details*	80-VR652-1

Questions?

https://support.cdmatech.com