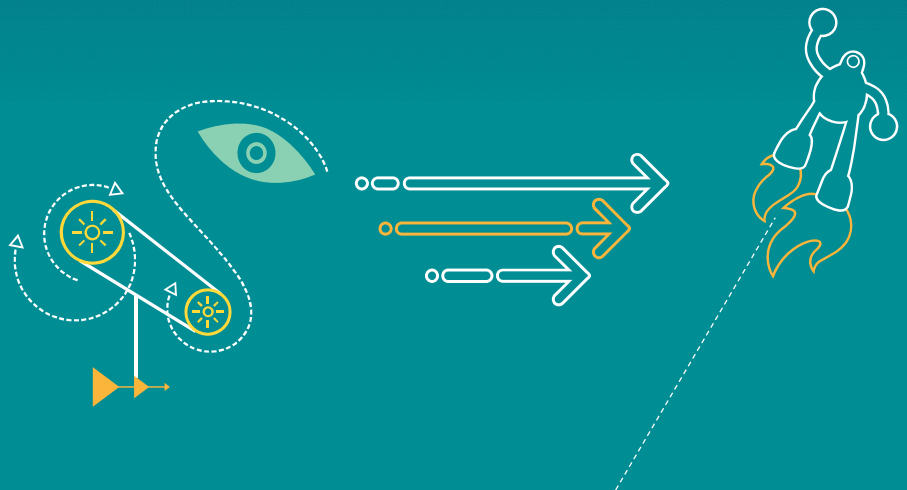
Watchdog and Abnormal Reset Mode Overview

Qualcomm Technologies, Inc.



80-NU268-1 A

Confidential and Proprietary – Qualcomm Technologies, Inc.

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

Revision History

Revision	Date	Description
A	Nov 2014	Initial release

Contents

Objectives

- At the end of this presentation, you will understand:
  - Watchdog reset debug
  - Abnormal reset debug
  - RAM dump analysis
    - Watchdog and abnormal reset

Problem Statements

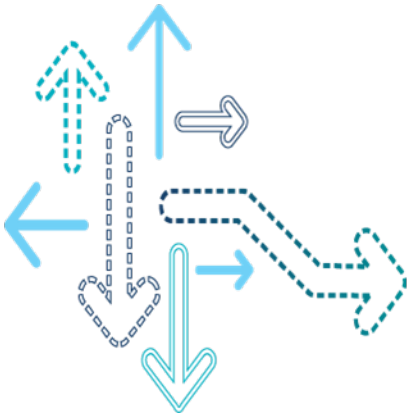
- Critical issues from OEMs
  - When the system is reset, e.g., watchdog reset on APQ8084, critical information for debug is lost.
  - When the system is stuck and the display is still on, it does not respond to any keypad inputs or peripherals, e.g., USB, and needs a way to reset to save critical information.
  - When PMIC or temperature sensor resets, critical information for debug is lost.

Watchdog reset debug fixes this problem on AQP8084

APQ8084 addresses this partially to keep KPSS power rail on, but cannot save current PC, and flush caches; MDM9607 and MSM8996 address this via abnormal reset debug

APQ8084 still has this problem; MDM9607 and MSM8996 address this via abnormal reset debug

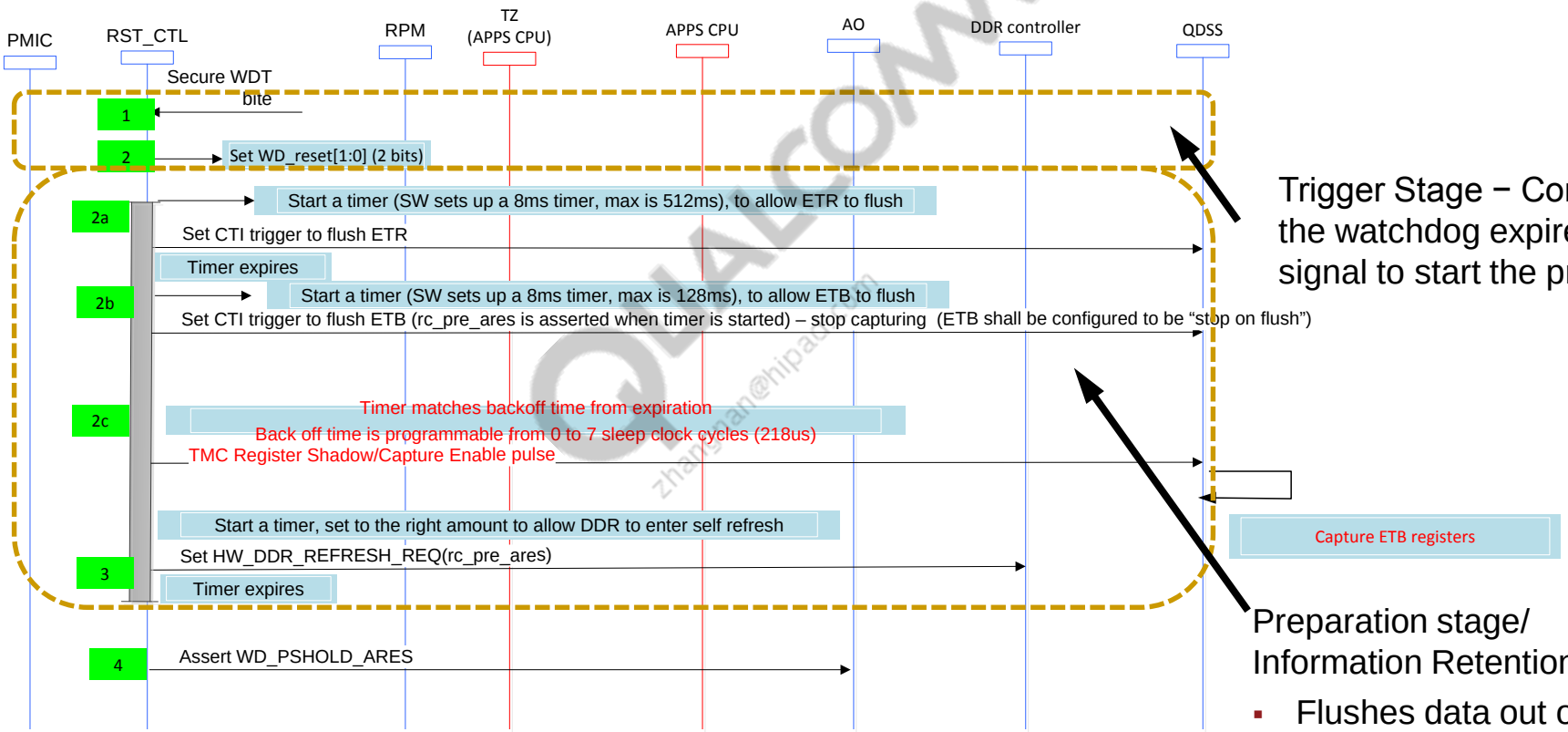
Secure Watchdog Reset Debug



Secure Watchdog Reset (System Crash) Debug

- Secure watchdog reset debug stages
  - Trigger stage
    - Captures the hardware signal or software error when a crash is about to happen, then feeds this signal to the custom reset sequence logic
  - Preparation stage/Information Retention stage
    - Drains the critical data out of the internal buffers; e.g., some inflight trace data is in the internal queues and will drain the data out to be collected
  - Information Retrieval stage
    - Provides a custom reset of the System-on-a-Chip (SoC) to go back to a minimal operation stage where information can be retrieved; e.g., the power shall be supplied to CPU to retain the cache, general purpose registers, and the DDR shall be refreshed, etc. A reset is necessary to get out of the Crash state.
    - Runs a minimal software image to retrieve data from various sources and dumps it to the DDR; e.g., invalidate caches—make it commit to memory
  - Information Download stage
    - After keeping all the information in the memory, generate a full reset to get back to the Normal Operational mode.
    - Go into a Special Download mode to dump data (RAM dump) out of the SoC for further analysis.

Secure Watchdog Reset Debug (Interaction Chart)

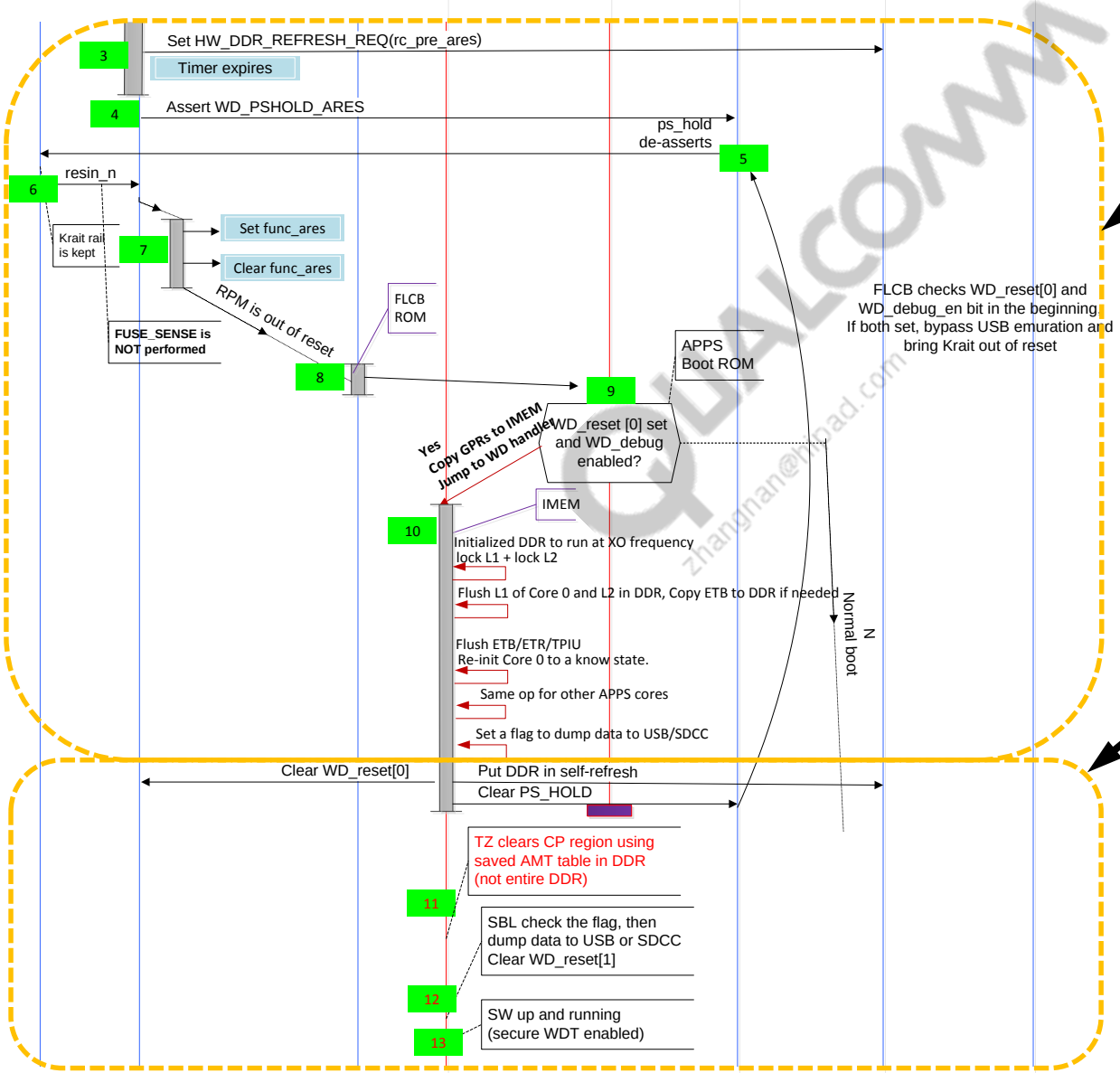


Trigger Stage – Connects the watchdog expired signal to start the process

Preparation stage/ Information Retention stage

- Flushes data out of internal queues, in particular, the system trace data
- Keeps DDR memory in Refresh mode

Secure Watchdog Reset Debug (Interaction Chart) (cont.)



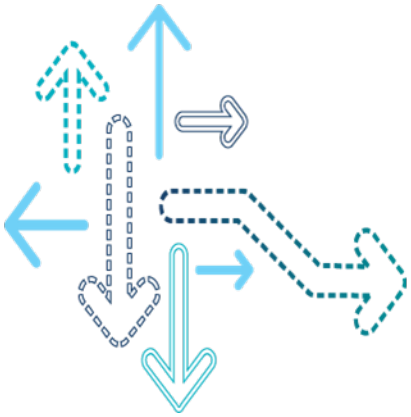
Information Retrieval stage

- Custom resets to keep power rails on CPUs
- Runs software to flush caches and reads critical data into memory

Information Download Stage

- Flushes data out of internal queues, in particular, the system trace data
- Keeps DDR memory in Refresh mode

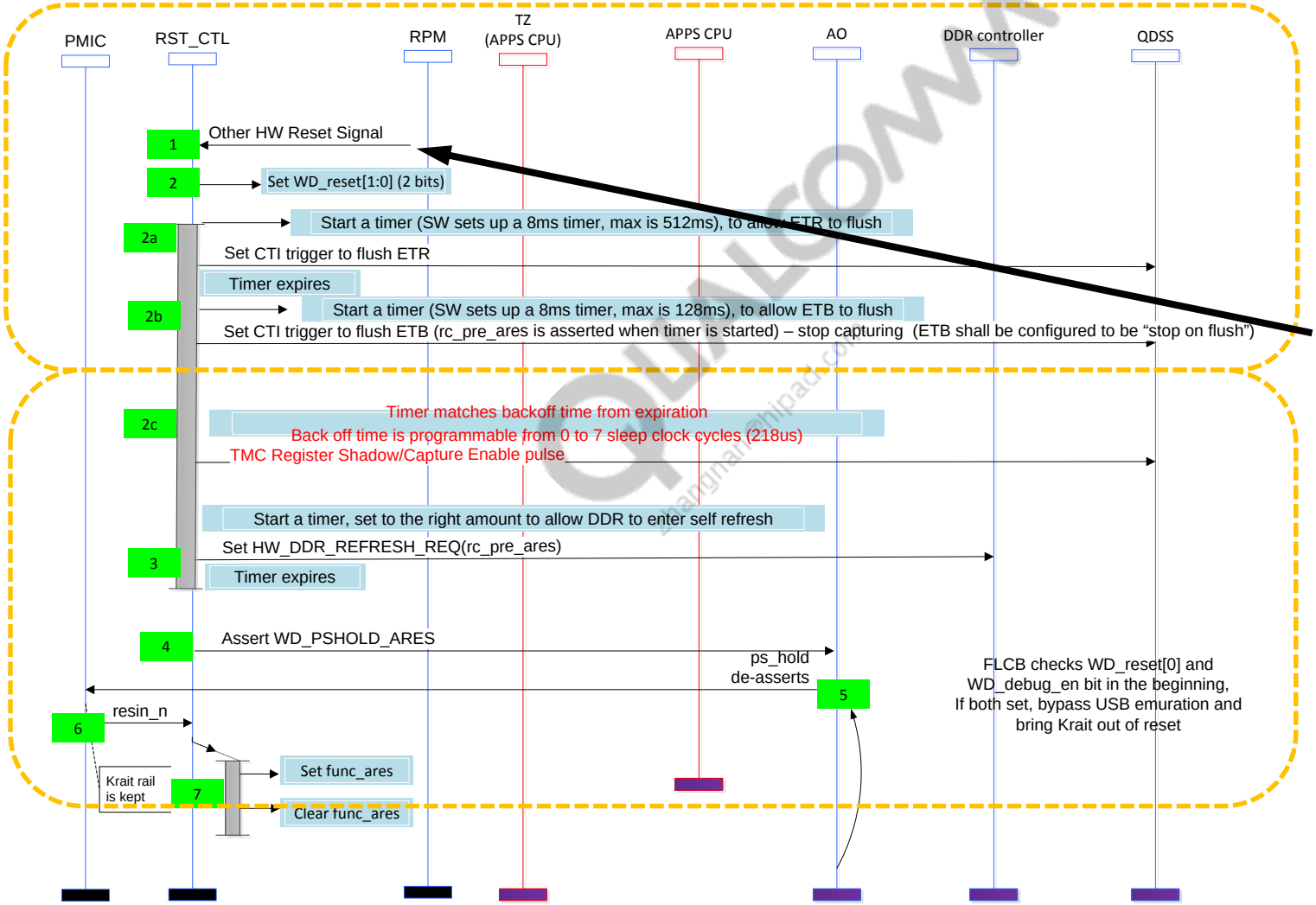
Abnormal Reset Debug



Abnormal Reset Debug

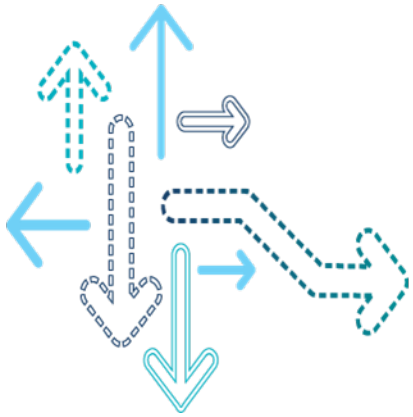
- Abnormal reset debug is an extension of watchdog reset debug in which the following problems are addressed:
  - When the system is stuck and the display is still on, but it does not respond to any keypad inputs or peripherals, e.g., USB, and needs a way to reset and save critical information
  - When the PMIC or temperature sensor resets and needs a way to save critical information
- How abnormal reset debug works
  - Connects the following abnormal resets (hardware reset, where software does not have a chance to save data) to watchdog at the trigger stage so that the critical information will be saved:
    - MSM Tsense
    - PMIC resets
      - PMIC Watchdog Expired
      - PMIC Tsense Level 1 and Level 2
      - PMIC Power+Volume Down reset (configurable)
  - Able to configure any one of the above resets to go through secure watchdog debug flow

Abnormal Reset Debug (cont.)

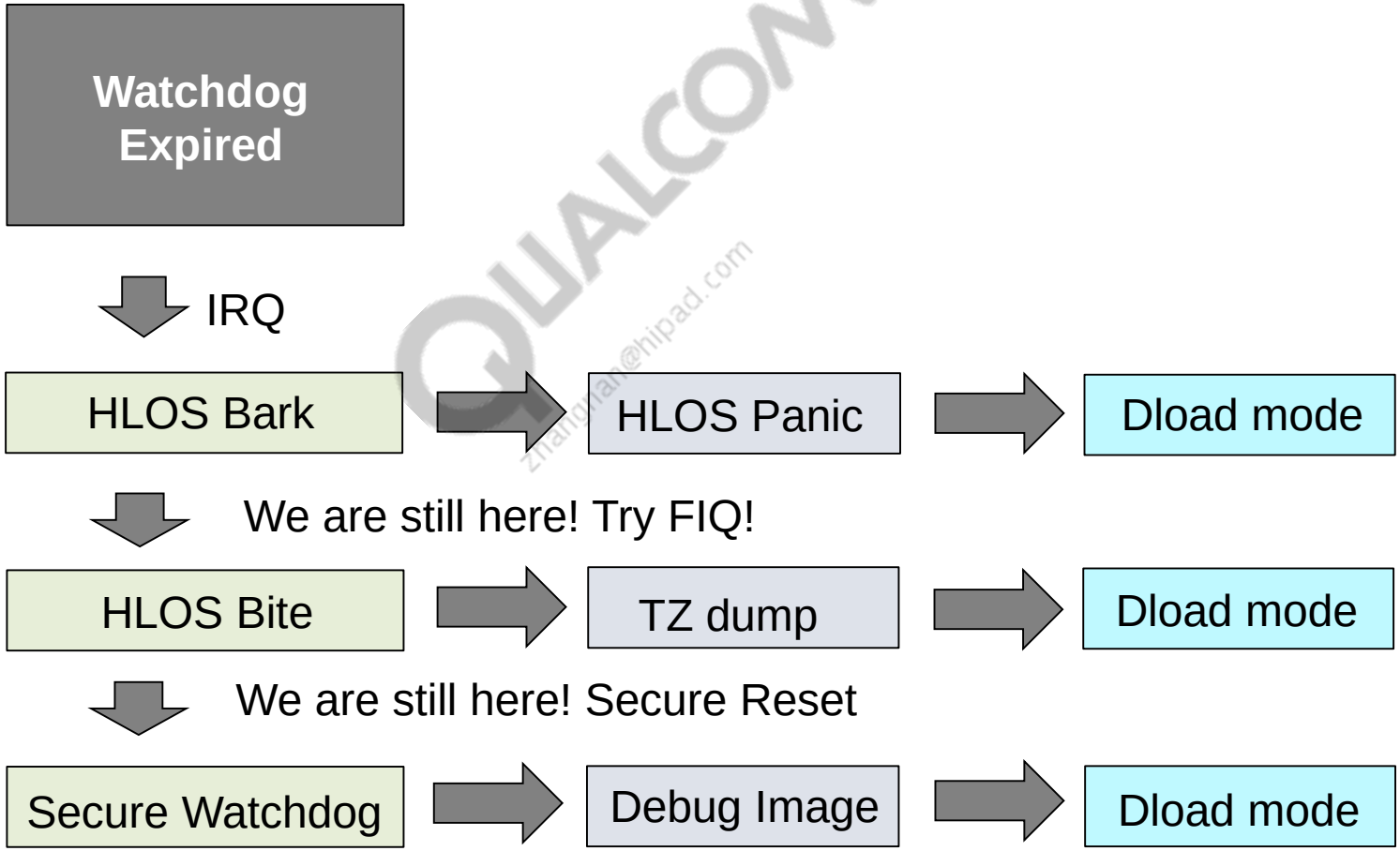


Trigger stage − Connects the other hardware reset signals such as MSM temperature sensor, PMIC reset signal, to start the same watchdog reset debug process

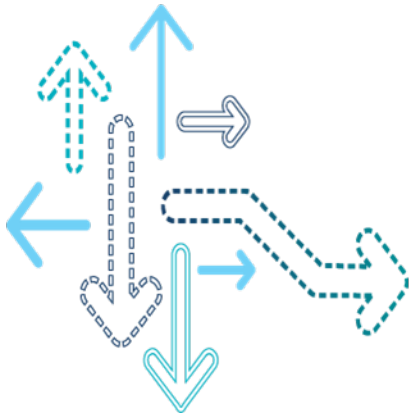
RAM Dump Debugging



Watchdog Flow

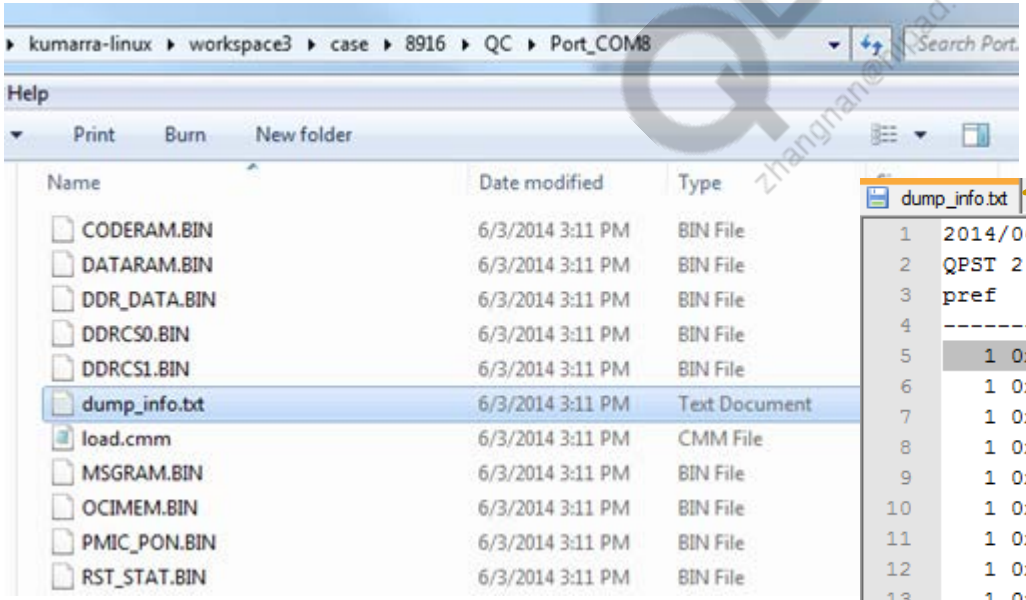


RAM Dump



RAM Dump Collection

- Watchdog (nonsecure/secure) triggered on various system-level crashes provides RAM dump which is collected using the QPST tool.
  - Kernel panic, apps WDOG bark, apps WDOG bite, fatal FIQs, secure WDOG bite
- The dump_info.txt provides details of various system region dumps.
  - Reset reason for watchdog/abnormal reset can be found in:
    - PMIC_PON.BIN, RST_STAT.BIN, and OCIMEN.BIN



OEMs dumps should ensure dump_info.txt is present.

dump_info.txt

```
2014/06/03 15:11:04.587
QPST 2.7.0.417
pref       base    length              region            file name
-----------------------------------------------------------------
   1 0x08600000     16384              OCIMEM           OCIMEM.BIN
   1 0x00200000    131072  RPM Code RAM region          CODERAM.BIN
   1 0x00290000     65536  RPM Data RAM region          DATARAM.BIN
   1 0x00060000     20480   RPM MSG RAM region           MSGRAM.BIN
   1 0x8671B360         8        Pmic PON stat         PMIC_PON.BIN
   1 0x8671B358         4  Reset Status Region         RST_STAT.BIN
   1 0x86771190      2048    DDR Training Data         DDR_DATA.BIN
   1 0x80000000 536870912       DDR CS0 Memory           DDRCS0.BIN
   1 0xA0000000 536870912      DDR CS1 Memory            DDRCS1.BIN
   2 0x86771990       756           CMM Script             load.cmm
2014/06/03 15:11:39.241
```

Watchdog Debug

- Linux RAM dump parser is available to parse logs for various WDOG and abnormal resets, see [R1]
  - Kernel panic – Unhandled page fault, BUG_ON, SLUB POISONs
  - Apps WDOG bark
  - Apps WDOG bite
  - TZ Fatal FIQ – Bus Timeout, NOC Error, xPU Error, etc.
  - Secure WDOG bite

Apps Watchdog Bark

- The watchdog work queue pet task is queued with a delay interval of pet time, which is ~10 sec, or as defined in the device tree file.
- As soon as a pet task is scheduled, it is expected to execute within 1 sec to avoid watchdog bark timeout.
- Otherwise, it will lead to watchdog bark, which generates fatal interrupts and prints the bark message shown below in kernel logs inside the IRQ handler.

```
----------begin TZRegDump----------
!!! Could not read from IMEM at address 8605658
------------end TZRegDump----------
```

```
[ 1228.584503] Watchdog bark! Now = 1228.584483
[ 1228.584542] Watchdog last pet at 1215.975101
[ 1228.584575] cpu alive mask from last pet 0
------------end Dmesg----------
```

```
----------begin DebugImage----------

Debug image version: 2.0 Number of table entries 1
--------
Debug image version: 2.0 Entry id: MSM_DUMP_TABLE_APPS Entry type: MSM_DUMP_TYP
--------
Parsing debug information for MSM_DUMP_DATA_CPU_CTX. Version: 3 Magic: 42445953
Parsing CPU0 context start ae01a000 end ae01a200
Core 0 PC: arch_counter_get_cntvct_cp15+4 <c0698aec>
Core 0 LR: arch_timer_read_counter_long+10 <c010b814>

[<c0698aec>] arch_counter_get_cntvct_cp15+0x4
[<c010b814>] arch_timer_read_counter_long+0x10
[<c0327b34>] read_current_timer+0x20
[<c0327b74>] __timer_delay+0x28
[<c03c97a0>] wdog_bark_handler+0x184
[<c015bbac>] handle_irq_event_percpu+0x84
[<c015bdd0>] handle_irq_event+0x3c
[<c015ea7c>] handle_fasteoi_irq+0xc0
[<c015b5b8>] generic_handle_irq+0x20
[<c0106a9c>] handle_IRQ+0x64
[<c0100520>] gic_handle_irq+0x70
[<c095be80>] __irq_svc+0x40
[<c0712cc4>] __sync_tlb+0xcc
[<c0712dc8>] __flush_iotlb+0xa0
[<c0712e1c>] msm_iommu_unmap_range+0x34
[<c07108a8>] iommu_unmap_range+0x28
[<c06b2ec8>] ion_iommu_map_release+0x68
[<c06b2fc8>] ion_unmap_iommu+0xd8
[<c036dae4>] mdss_mdp_put_img+0x128
[<c0383fa4>] mdss_mdp_overlay_free_buf+0x28
[<c0384024>] __mdss_mdp_overlay_free_list_purge+0x50
[<c03842dc>] mdss_mdp_overlay_cleanup+0x218
[<c0384b34>] mdss_mdp_overlay_kickoff+0x6f8
[<c039bd3c>] __mdss_fb_display_thread+0x128
[<c013aaa4>] kthread+0xa0
[<c0106258>] ret_from_fork+0x14
```

Apps Watchdog Bite

- The nonsecure watchdog bite time is configured as bark time +3 sec.
- If the watchdog pet task is unable to reset the watchdog counter and the watchdog bark interrupt is not executed in the next 3 sec, this will lead to a nonsecure watchdog bite timeout.
- This nonsecure watchdog bite timeout will generate an FIQ to TZ and is handled by TZ.

```
Reset Status
========================
CPU  |Reset Reason                                                  |Reset Count
0    |0x00000001   (TZBSP_ERR_FATAL_NON_SECURE_WDT                )|0x00000001
1    |0x00000000   (TZBSP_ERR_FATAL_NONE                          )|0x00000000
2    |0x00000000   (TZBSP_ERR_FATAL_NONE                          )|0x00000000
3    |0x00000000   (TZBSP_ERR_FATAL_NONE                          )|0x00000000


----------begin TZRegDump----------
!!! Could not read from IMEM at address 8605658
------------end TZRegDump----------


----------begin DebugImage----------

Debug image version: 2.0 Number of table entries 1
--------
Debug image version: 2.0 Entry id: MSM_DUMP_TABLE_APPS Entry type: MSM_DUMP_TYPE
--------
Parsing debug information for MSM_DUMP_DATA_CPU_CTX. Version: 3 Magic: 42445953
Parsing CPU0 context start ae01a000 end ae01a200
Core 0 PC: msm_jtag_mm_restore_state+4e4 <c03b36e8>
Core 0 LR: uncached_logk_pc+14 <c01a31fc>

[<c03b36e8>] msm_jtag_mm_restore_state+0x4e4
[<c094bf80>] msm_pm_spm_power_collapse+0x264
[<c06330bc>] msm_pm_power_collapse+0xf8
[<c063329c>] msm_cpu_pm_enter_sleep+0xc4
[<c06620f8>] lpm_enter_low_power.constprop.3+0x3d8
[<c0662560>] lpm_cpuidle_enter+0x2f8
[<c065f670>] cpuidle_enter_state+0x40
[<c065f864>] cpuidle_idle_call+0x144
[<c0106e4c>] arch_cpu_idle+0x8
[<c015b2d4>] cpu_startup_entry+0x134
[<c094a49c>] rest_init+0x8c
[<c0f00a7c>] start_kernel+0x30c
[<80008070>] (No symbol for address 80008070)+0x0
```

OCIMEM.BIN

0000087c	00	01	02	03
00000760	00000001	00000023	00000000	00000000
00000770	00000000	4fc6cccd	74404c02	9ba573f2

Nonsecure WDOG Bite and Fatal FIQ − TZ Diag Buffer

- The following shows RAM dump collected after a controlled reset (TZ fatal or WDOG trigger); the TZ diagnostic buffer provides more information on reset.
  - The TZ diag buffer struct pointer is saved at 0x720 offset on OCIMEM.BIN and can be loaded with tz.elf. v.v (struct tzbsp_diag_s *)(0x8600000+0x720).

```
B::v.v %hex %o (struct tzbsp_diag_s *)(*(unsigned long*)0x8600720)
⊟ (struct tzbsp_diag_s *)(*(unsigned long*)0x8600720) = 0x8665F000 → (
  · magic_num = 0x747A6461,
  · version = 0x00030000,
  · cpu_count = 0x4,
  · vmid_info_off = 0x24,
  · boot_info_off = 0xA4,
  · reset_info_off = 0x0104,
  · int_info_off = 0x0128,
  · ring_off = 0x082C,
  · ring_len = 0x07D4,
  ⊞ vmid = ((vmid = 0xFF, desc = (0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
  ⊟ boot_info = (
    ⊞ (warm_entry_cnt = 0x00035115, warm_exit_cnt = 0x00035115, term_entry
    ⊞ (warm_entry_cnt = 0x0480, warm_exit_cnt = 0x0480, term_entry_cnt = 0
    ⊞ (warm_entry_cnt = 0x025A, warm_exit_cnt = 0x025A, term_entry_cnt = 0
    ⊞ (warm_entry_cnt = 0x01E7, warm_exit_cnt = 0x01E7, term_entry_cnt = 0
  ⊟ reset_info = (
    ⊞ (reset_type = 0x0, reset_cnt = 0x0),
    ⊞ (reset_type = 0x1, reset_cnt = 0x1),
    ⊞ (reset_type = 0x0, reset_cnt = 0x0),
    ⊞ (reset_type = 0x0, reset_cnt = 0x0)),
  · num_interrupts = 0x20,
  ⊟ int_info = (
    ⊞ (int_info = 0x0, avail = 0x0, spare = 0x0, int_num = 0xE3, int_desc
    ⊞ (int_info = 0x0, avail = 0x0, spare = 0x0, int_num = 0xE4, int_desc
    ⊞ (int_info = 0x0, avail = 0x0, spare = 0x0, int_num = 0x24, int_desc
    ⊞ (int_info = 0x0, avail = 0x0, spare = 0x0, int_num = 0x0F, int_desc
```

TZ Fatal FIQ – NOC Error

- The Network-on-a-Chip (NoC) Error triggers TZ fatal FIQ. TZ diag logs show the specific NoC on which the error occurred.

```
PCNOC ERROR: ERRLOG0 = 0x80030000
SNOC ERROR: ERRLOG0 = 0x80030000
PCNOC ERROR: ERRLOG1 = 0x32b02030
SNOC ERROR: ERRLOG1 = 0x05404060
PCNOC ERROR: ERRLOG3 = 0x000e3000
SNOC ERROR: ERRLOG3 = 0x01de3000
PCNOC ERROR: ERRLOG4 = 0x00000000
SNOC ERROR: ERRLOG4 = 0x00000000
PCNOC ERROR: ERRLOG5 = 0x0000ba22
SNOC ERROR: ERRLOG5 = 0x0000ba11
Fatal Error: NOC_ERROR
Fatal Error: NOC_ERROR
```

```
Reset Status
=========================
CPU  |Reset Reason                                        |Reset Count
0    |0x00000000    (TZBSP_ERR_FATAL_NONE                )|0x00000000
1    |0x00000006    (TZBSP_ERR_FATAL_NOC_ERROR           )|0x00000001
2    |0x00000006    (TZBSP_ERR_FATAL_NOC_ERROR           )|0x00000001
3    |0x00000000    (TZBSP_ERR_FATAL_NONE                )|0x00000000
```

```
ENTER ERROR LOG VALUE :0x05404060
SNOC ROUTE ID INFORMATION :
InitFlow = 0x2 -> qxm_bimc/I/0
TargFlow = 0x0A -> qxs_pcnoc/T/0
MID = 0x3
BID = 0x2 -> BIMC
PID = 0x0 -> A53SS
TargSubRange = 0x0
SeqId = 0x0

ENTER ERROR LOG VALUE :0x32b02030
PCNOC ROUTE ID INFORMATION :
InitFlow = 0x0C -> qxm_snoc/I/0
TargFlow = 0x2B -> qhs4/venus0_cfg
MID = 0x3
BID = 0x2
PID = 0x0
TargSubRange = 0x0
SeqId = 0x0
```

```
----------begin TZRegDump----------
!!! Could not read from IMEM at address 8605658
----------end TZRegDump----------


----------begin DebugImage----------

Debug image version: 2.0 Number of table entries 1
--------
Debug image version: 2.0 Entry id: MSM_DUMP_TABLE_APPS Entry type: MSM_DUMP_TYPI
--------
Parsing debug information for MSM_DUMP_DATA_CPU_CTX. Version: 3 Magic: 42445953
Parsing CPU0 context start ae01a000 end ae01a200
Core 0 PC: msm_pm_swfi+8 <c0631e94>
Core 0 LR: msm_cpu_pm_enter_sleep+c4 <c063329c>

[<c0631e94>] msm_pm_swfi+0x8
[<c063329c>] msm_cpu_pm_enter_sleep+0xc4
[<c06620f8>] lpm_enter_low_power.constprop.3+0x3d8
[<c0662560>] lpm_cpuidle_enter+0x2f8
[<c065f670>] cpuidle_enter_state+0x40
```

Debugging SDI/SYSDBG Logs

- On the MSM8916, the SDI image will not be present.
  - Instead, the same functionality will be a part of TZ and boot image. The driver/feature set is labeled system debug or sysdbg.
  - Important sysdbg cookies get saved in IMEM.

```
* SHARED_IMEM_BASE = 0x08600000
* Dump Table Location = SHARED_IMEM_BASE+0x010
* SDI Pass 1 Successful Cookie = SHARED_IMEM_BASE+0xB14 with value 0xDEADD00D
* Value of GCC_RESET_STATUS when SDI Pass 1 executed = SHARED_IMEM_BASE+0x764[15:0]
 Some PMIC reset reasons are captured as well
 SHARED_IMEM_BASE + 0x764       - GCC_RESET_STATUS
 SHARED_IMEM_BASE + 0x768[15:8] - PMIC PON Reason 1
 SHARED_IMEM_BASE + 0x76C[15:8] - PMIC Warm Reset Reason 1
 SHARED_IMEM_BASE + 0x76C[7:0]  - PMIC Warm Reset Reason 2
 SHARED_IMEM_BASE + 0x770[15:8] - PMIC POFF Reason 1
 SHARED_IMEM_BASE + 0x770[7:0]  - PMIC POFF Reason 2
```

  - Only those sysdbg cookies shown above are updated, and they will not be updated if sysdbg did not run for any reason, such as kernel panic or secure fuse blown device.
    - In these cases, the SBL collects PMIC reset regions on PMIC_PON.BIN, RST_STAT.BIN.

Interpretation of GCC_RESET_STATUS

- GCC_RESET_STATUS

```
GCC_RESET_STATUS = 0x23 -> Secure Watchdog Bite
GCC_RESET_STATUS = 0x13 -> PMIC Abnormal Reset
GCC_RESET_STATUS = 0x1B -> TSENSE Reset (Temperature Sensor Triggered Reset)
GCC_RESET_STATUS = 0x4  -> Software Triggered Reset
GCC_RESET_STATUS = 0x0  -> Non-MSM triggered Reset
```

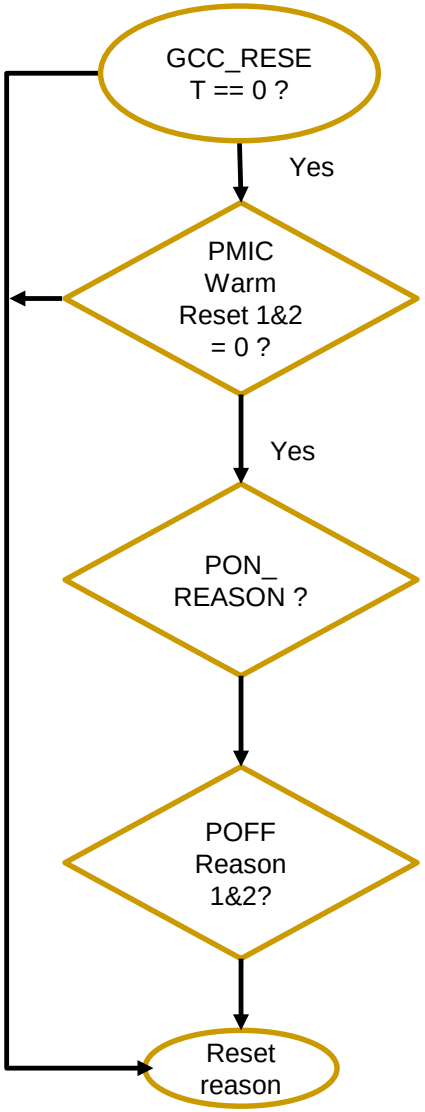
- TSENSE reset follows a reset signal generated to PMIC, which will be considered as a PMIC abnormal reset, hence in case of TSENSE reset both bits 3 and 4 of the GCC_RESET_STATUS register get updated.
- IN SRST reset case, bits 0 and 1 will not get updated unless the GCC_SW_SRST bit is enabled, hence the GCC_RESET_STATUS value will be 0x4 in the case of SRST reset.

Interpretation of GCC_RESET_STATUS (cont.)

- If GCC_RESET_STATUS = 0 x 0:
  - Check PMIC Warm Reset 1 and 2 -> If 0, go to (b).
    - If not, there was a PMIC warm reset. There can be multiple triggers for a warm reset at the same time, and all triggers are captured in these registers, i.e., you can have more than one bit set.
  - If Warm Reset reasons 1 and 2 are 0, then PON_REASON and POFF_REASON1/ POFF_REASON2 register.
    - These will be updated every time the PMIC powers on and off.
- The snapshot shows RESET reason as secure WDOG bite, if TZ does not receive nonsecure WDOG bite.

OCIMEM.BIN

0000087c	00	01	02	03
00000760	00000001	00000023	00000000	00000000
00000770	00000000	4fc6cccd	74404c02	9ba573f2



PON/POFF Reset Reason

- Available triggers for resets (see [Q2])
  - KPDPWR_N – External signal to PMIC
  - RESIN_N – External signal to PMIC
  - KPDPWR_N and RESIN_N – External signal to PMIC
  - GP2 – Routed internally from keypad combo
  - GP1 – Routed internally from keypad combo
  - PMIC Watchdog (PMIC_WD) – PMIC internal signal
  - PS_HOLD – External signal to PMIC, e.g., MSM™ watchdog
  - Software Reset – Registers write to PMIC to perform a reset
  - Overtemperature Stage 3 (OTST3) – PMIC internal signal (this is a trigger on OTST3)
  - Automatic Fault Protection (AFP) – PMIC internal signal
  - Undervoltage Lockout (UVLO) – Not really a trigger, just loss of power

References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*PMIC PON-Reset Software Drivers Overview*	80-NM620-1
Q3	*TrustZone (TZ.BF) Debug Manual*	80-NA157-209
Q4	*PM8916 Software Interface for OEMs*	80-NK808-2X
Q5	*MSM8916 Linux Android Software Debug overview*	80-NL239-7
Resources		
R1	*Code Aurora Forum*	https://www.codeaurora.org/cgit/quic/la/platform/vendor/qcom-opensource/tools/tree/linux-ramdump-parser-v2

Questions?

https://support.cdmatech.com