1 Introduction

This KBA will show customers for basic debugging skills of RPM Issues, including RPM Dump Analysis and Typical RPM Issue Debugging Tips.

This is Applicable to new platforms later than B family, especially latest platforms such as 8937,8953,8976,8x96, 9x45,8x98.

2 How to load RPM dump

2.1 Prepare

Usually only 4 file are needed for a RPM Crash Dump analysis. They are

```
CODERAM.bin,

DATARAM.bin,

MSGRAM.bin

\rpm_proc\core\bsp\rpm\build\RPM_AAAAANAAR.elf
```


They have very small size and easy to load and upload to SF for QCOM further analysis

2.2 Steps

1 Launch a T32 simulator window

Create a config file, named "config_sim.t32" for example, based on in <YOUR_DIR>\T32\Config.t32 and change "USB" to "SIM"

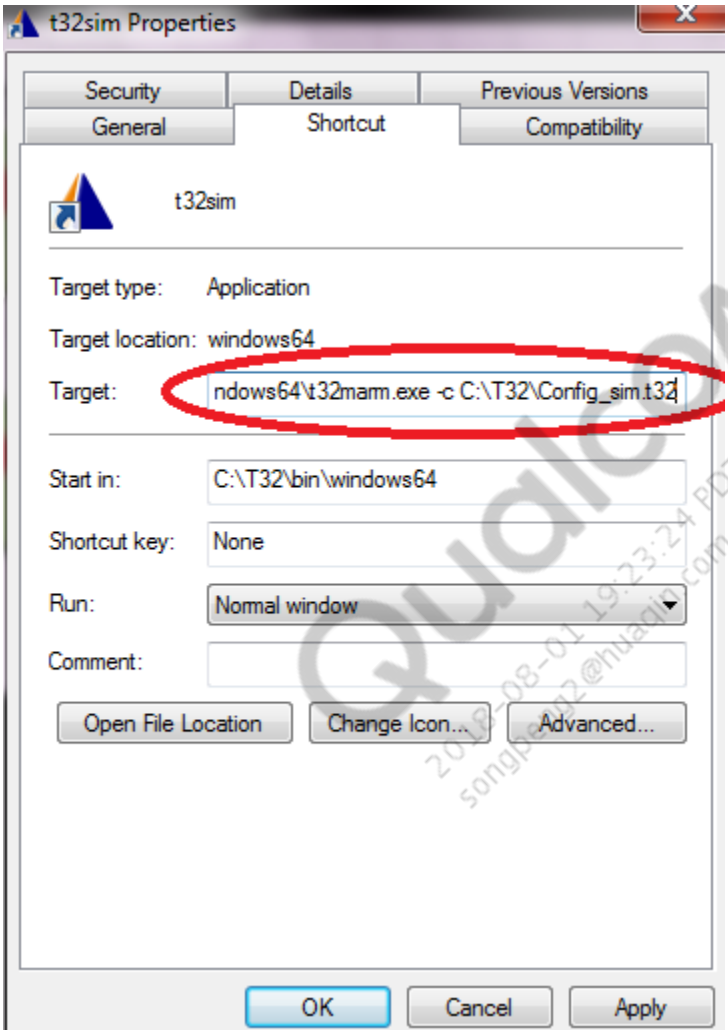
```
; Environment variables
OS=
ID=T32
TMP=C:\Users\tongliu\AppData\Local\Temp
SYS=C:\T32


PBI= SIM
```

```
;USB

; Printer settings
PRINTER=WINDOWS
```

then make a shortcut to launch T32 as simlator config file



2 Use the following cmd to load dumps to T32 simulator

you can make it in your own cmm. different platforms have different load address, you can find CODERAM/DATARAM/MSGRAM loading addresses in

rpm_proc\core\bsp\rpm\scripts\rpm_load_dump.cmm

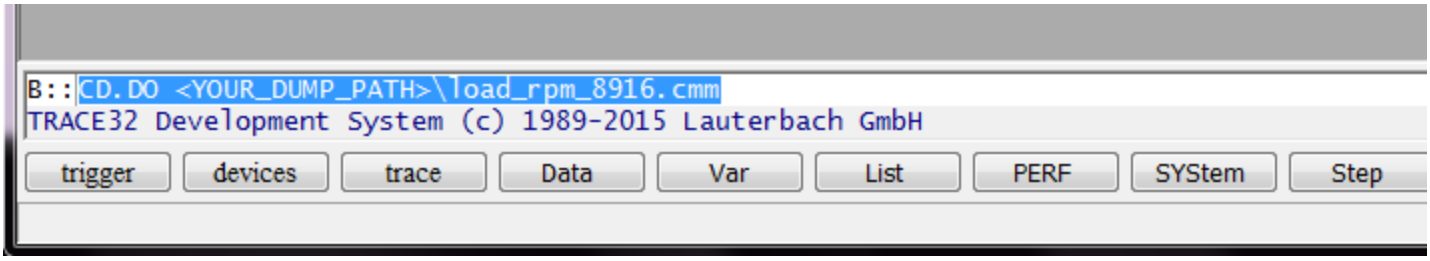
```
sys.cpu.CORTEXM3

sys.down

sys.up
```

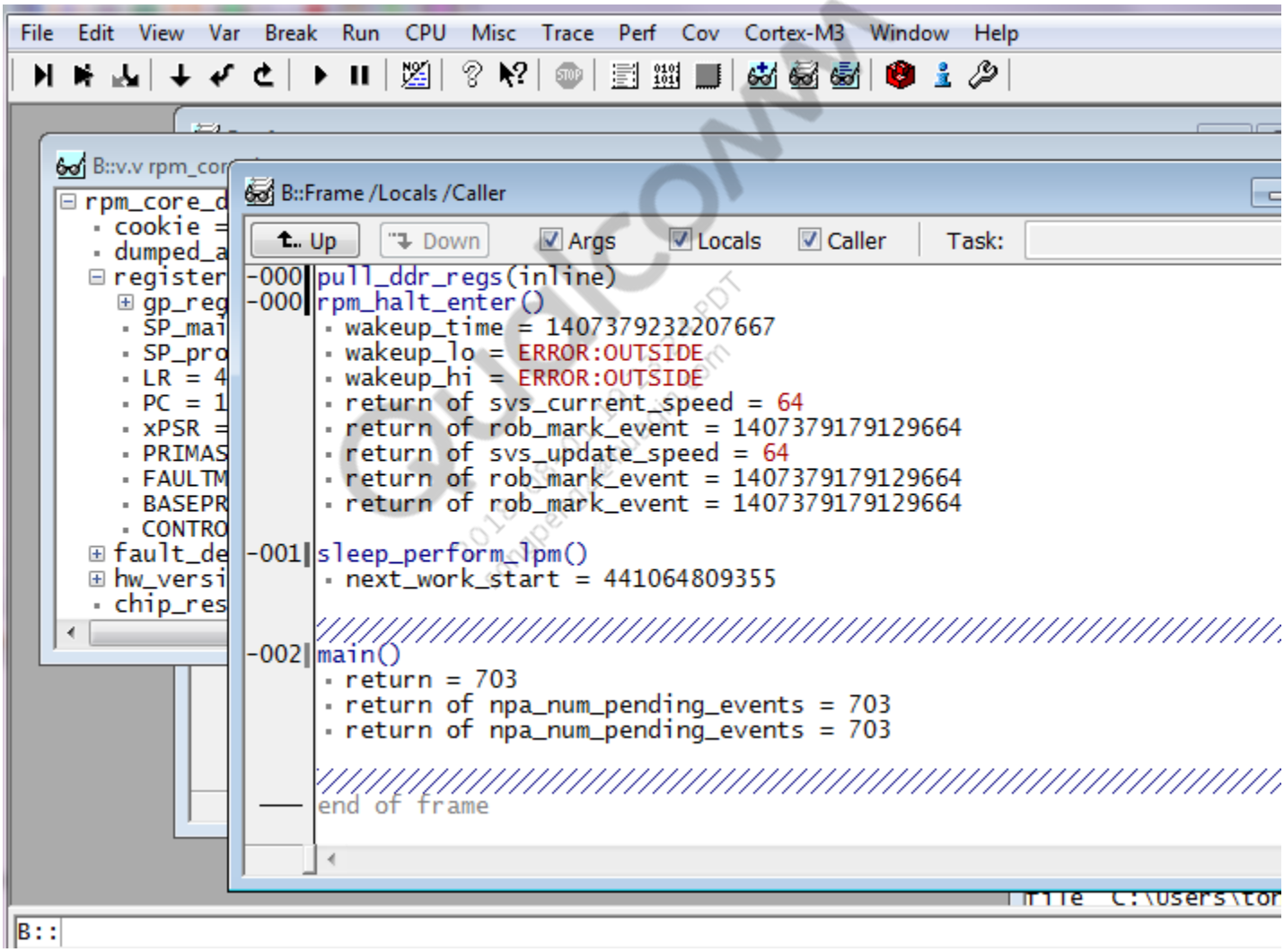
```
if OS.FILE(CODERAM.BIN)
(
d.load.binary CODERAM.BIN 0x0 /noclear
)
if OS.FILE(DATARAM.BIN)
(
d.load.binary DATARAM.BIN 0x90000 /noclear
)
if OS.FILE(MSGRAM.BIN)
(
d.load.binary MSGRAM.BIN 0x60060000 /noclear
)
d.load.elf *.elf /nocode /noclear
Do YOUR_RPM_DIR\rpm_proc\core\bsp\rpm\scripts\rpm_restore_from_core.cmm
Do YOUR_RPM_DIR\rpm_proc\core\bsp\rpm\scripts\rpm_m3_unstack.cmm
v.v %string gBuildDate gBuildTime
v.v rpm_core_dump
v.v event_data
v.v (EEData[6])(*rpm.ees)
v.v sleep_stats[0]
v.v sleep_stats[1]
v.f
```

Load your cmm with above lines to T32 simulator



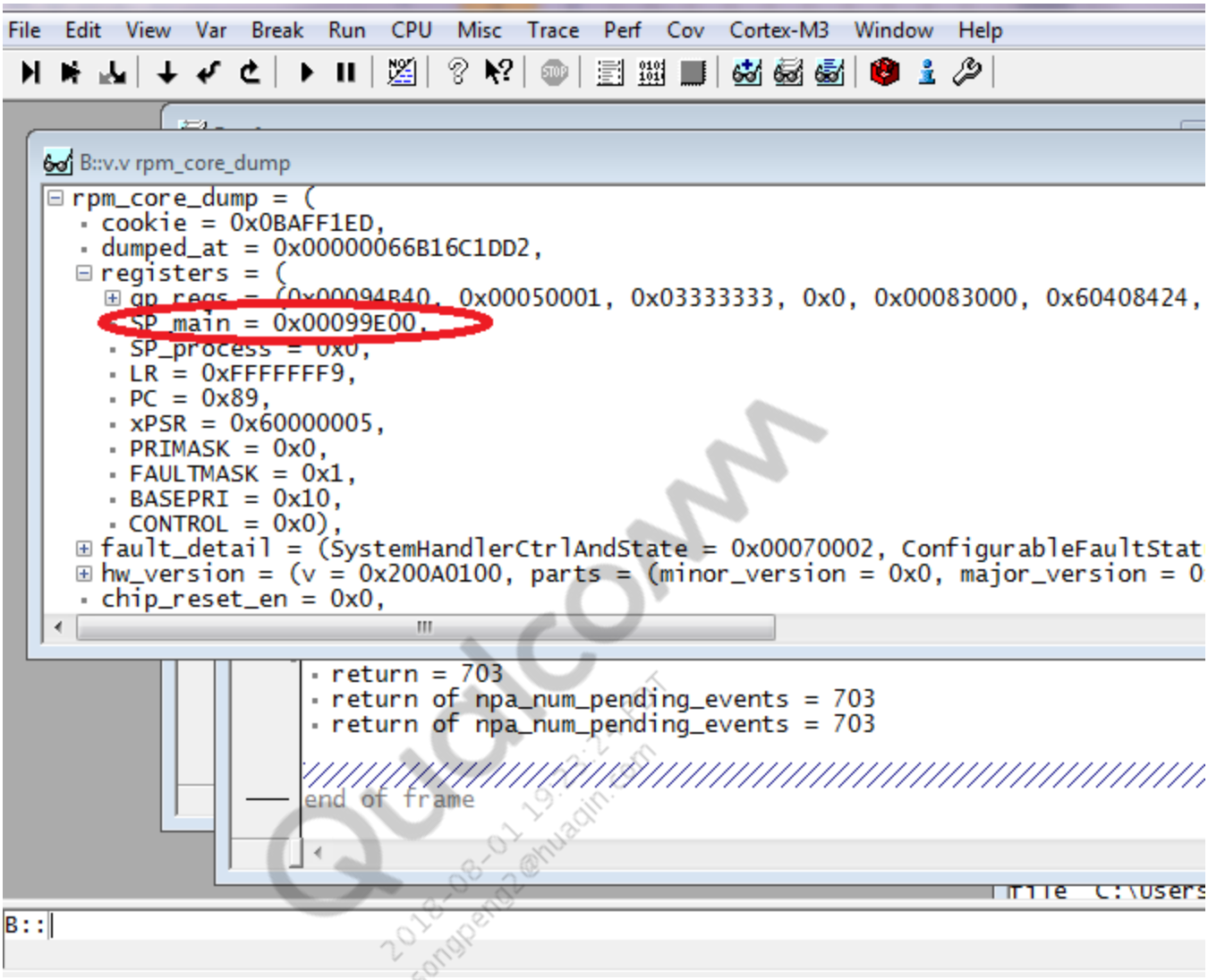

```
v.v (struct railway_rail_state_t[3])*(railway.rail_state)
```

Here you should already get callstack for a normal RPM err fatal, for example, like this



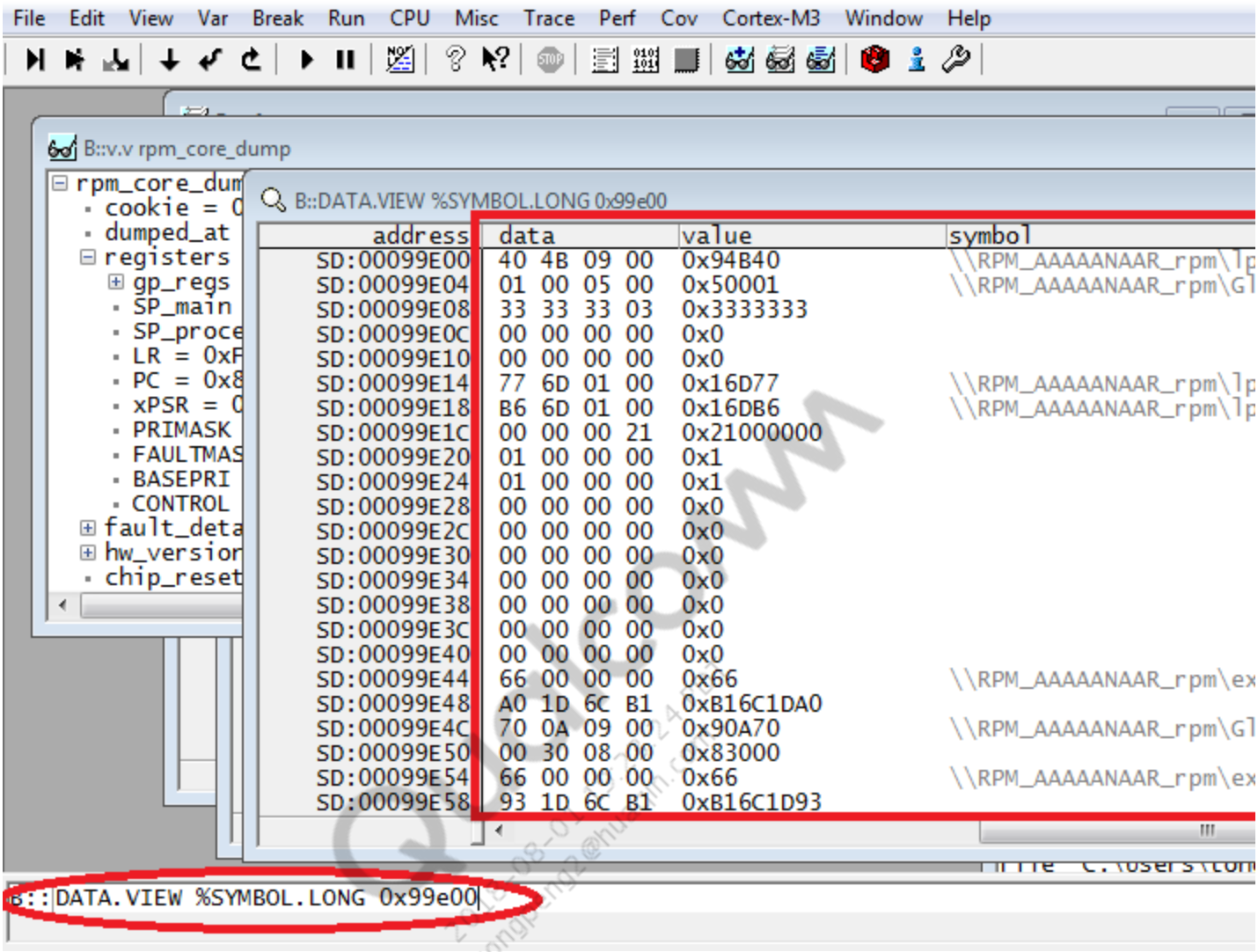
if you cannot get it, you may need to manually restore the callstack A simple way is to check variable by v.v

```
"rpm_core_dump"
```



and find the main SP(R13) value, then use the following cmd to help you get a rough callstack view

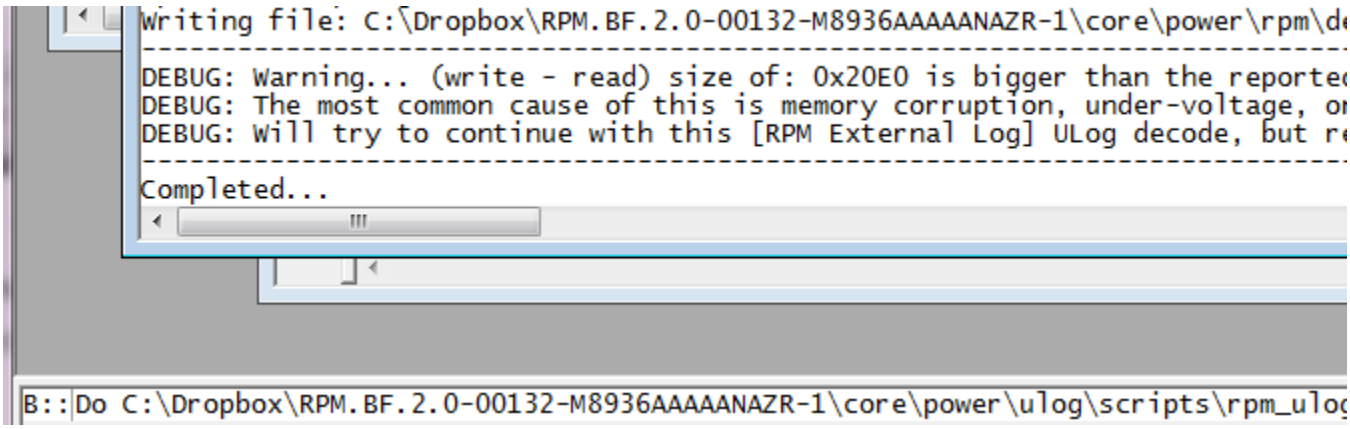
```
DATA.VIEW %SYMBOL.LONG 0xXXXXX(R13 value)
```



2.3 Extracting RPM logs from dump

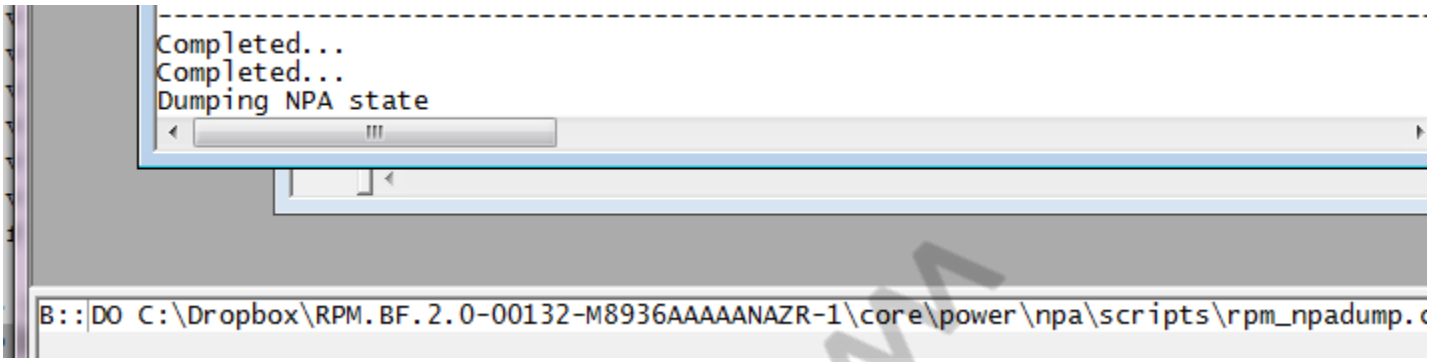
Extract an RPM external log.(T32 simulator)

```
do YOUR_RPM_DIR\rpm_proc\core\power\ulog\scripts\ULogDump.cmm YOUR_RPM_DIR\rpm_proc\
core\power\rpm\debug\scripts\
```



Extract an NPA log. .(T32 simulator)

```
do YOUR_RPM_DIR\rpm_proc\core\power\npa\scripts\NPADump.cmm YOUR_RPM_DIR\rpm_proc\
core\power\rpm\debug\scripts\
```

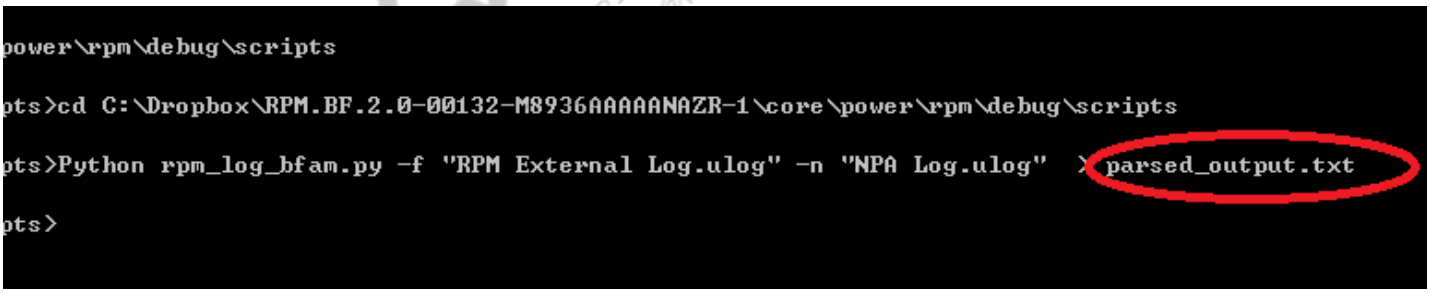


Execute the Python script to parse the RPM logs(DOS cmd window)

open a "cmd" window and

```
cd YOUR_RPM_DIR\rpm_proc\core\power\rpm\debug\scripts\

Python rpm_log_bfam.py -f "RPM External Log.ulog" -n "NPA Log.ulog" >
parsed_output.txt
```



2.4 How to check current VDDCX/VDDMX/VDD_EBI

Sometimes, we need to know key voltage info like VDDCX/VDDMX and VDD_EBI(if has), please Use cmd

```
v.v (struct railway_rail_state_t[3])*(railway.rail_state)
```

2.5 Extracting RPM logs by Hansei scripts

Tool for parsing debug information out of the RAM dump; generates RPM logs, NPA logs, master status, resource states, etc.

Installation

Install Python 2.7.x (not 2.6.x). Check version -python -V.

Install the pyelftoolslibrary that supports the ARM compiler; the mainline version does not work. Instead, use

```
https://bitbucket.org/pplesnar/pyelftools-pp.
```

Install command

```
python setup.py install
```

Hansei script

Location

```
rpm_proc\core\bsp\rpm\scripts\hansei\
```

Usage

```
hansei.py [-h] --elf rpm.elf[--output path] dumpfile[dumpfile...]
```

Example

```
python hansei.py -elf rpm.elf-o .
rpm_code_ram.binrpm_data_ram.binrpm_msg_ram.bin
```

Output explanation

rpm-summary.txt -Contains general information about the health of the RPM, including the core dump state and fault information

rpm-log.txt -PostprocessedRPM external log

npa-dump.txt -Standard NPA dump format, albeit without (inaccurate) timestamps

ee-status.txt -Contains information about which subsystems and their cores are active or sleeping

reqs_by_master/* -Folder containing a file for each execution environment, detailing current requests EE has in place with the RPM

reqs_by_resource/* -Folder structure containing a folder for each resource type registered with the RPM server, and under that folder, a file containing all of the requests to each resource of that type

3 RPM Issue Debug

3.1 CODERAM bitflip

Most normal RPM crash are caused by CODERAM bitflip (inside MSM, not DDR). This should be no SW fix unless bump up vddcx/mx, usually customer only need to follow the steps to confirm a bitflip, and then check with QCOM whether we need a RMA process or some test changes. Bitflip in CODERAM will cause ARM usage fault or bus fault due to wrong address/values.

Usually we can compare the CODERAM from dumps with the CODERAM from ELFs by beyondCompare

Especially the single bit flip around the address of R13. Please notice that the CODERAM will be re-used by XBL/PBL after reset, so there are some large area of delta compared to ELF. So we only focus on 1-2 bit flips and ignore the large area of delta.

3.1.1 Steps

1 Load rpm ELF to T32 simulator. (do not load .bin, only ELF)

```
sys.up

d.load.elf YOUR_DIR/ RPM_AAAAANAAR.elf
```

2 Save CODERAM area from ELF, for start address, please refer to rpm_proc\core\bsp\rpm\scripts\ rpm_load_dump.cmm

for 128K coderam

```
D.save.binary c:\CODERAM.bin 0x0++(0x20000-1)
```

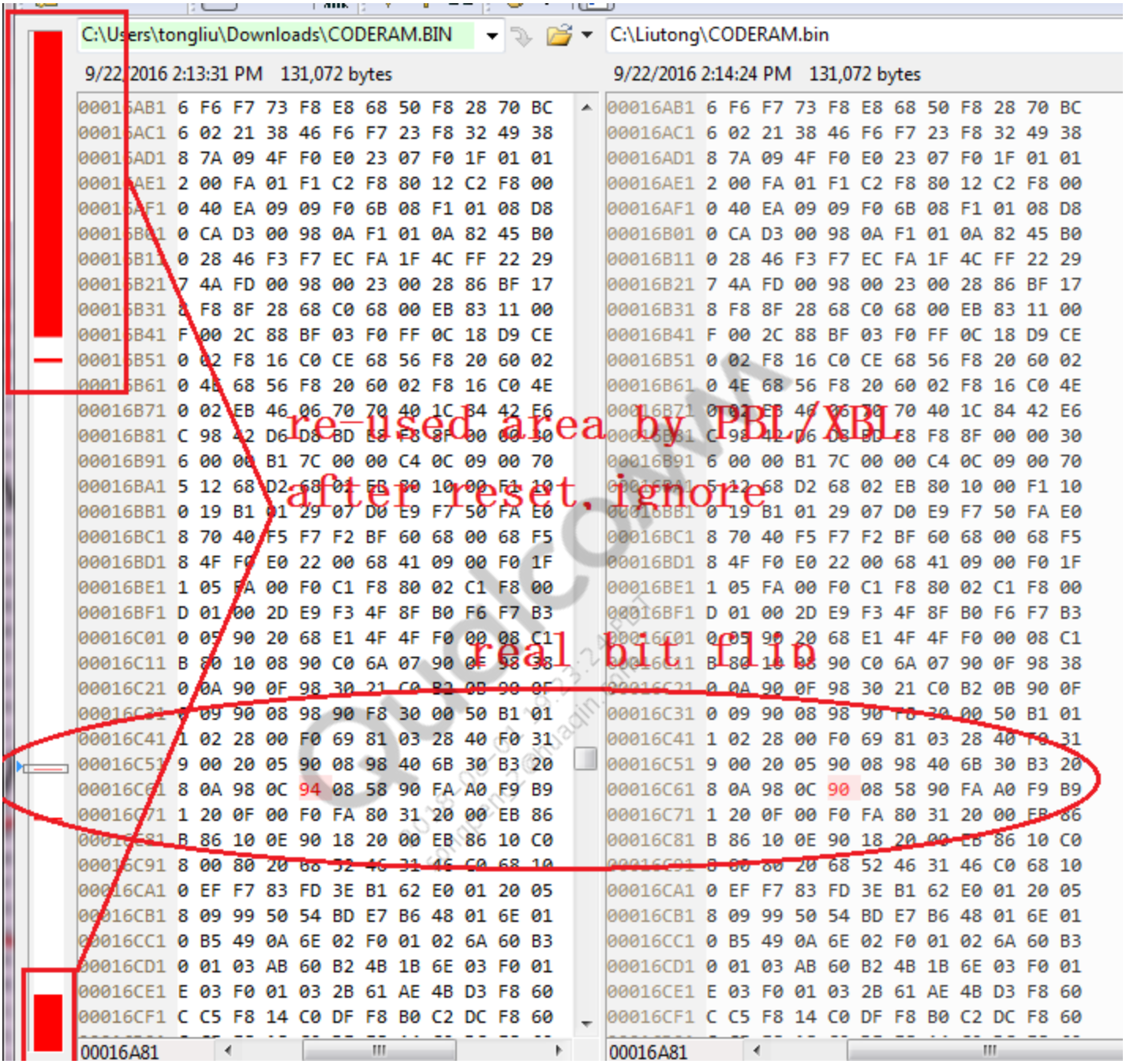
For 160K coderam

```
D.save.binary c:\CODERAM.bin 0x0++(0x28000-1)
```

3 Compare with CODERAM.BIN from dumps by beyond compare. ignore the large consecutive diff because some area are overwritten by PBL, only focus on separately single or double bit flip

If we found bit flips, check the address of the flip position ,and confirm whether this is around the R13(stack) address, you can refer to section 2.3 for how to get R13 and callstack.

If the bit flip address is shown around the R13 address, we can 100% confirm this bit flip cause the RPM abort during function call process.



3.1.2 Example

From CodeRAM

_____addr/line|code_____|label____|mnemonic_____|comment

ST:000161F2|1843 adds r3,r0,r1 ; r3,return of

From ELF

_____addr/line|code_____|label____|mnemonic_____|comment

ST:000161F2|1043 asr r3,r0,#0x1 ; r3,return of svs_current_

In address 0x161f2, there is a 1->0 bitflips, the code should be 0x1043 but become 1843
get call stack from R13,

_____address|_data_____|value_____|symbol

SD:0009D8F8| 55 61 F0 74 0x74F06155
SD:0009D8FC| 00 30 08 00 0x83000
SD:0009D900| 3A 00 48 D2 0xD248003A
SD:0009D904| 2F 00 00 00 0x2F \\RPM_AAAAANAAR(3)\Global__rt_entry+0x5
SD:0009D908| F0 50 09 00 0x950F0 \\RPM_AAAAANAAR(3)\lpr_definition_uber\sleep_mode
SD:0009D90C| 43 01 00 00 0x143 \\RPM_AAAAANAAR(3)\lpr_definition_uber__asm___21_lpr_definition_uber_c_ebf0609d__halt+0x1B
SD:0009D910| 00 00 00 00 0x0
SD:0009D914| FD 61 01 00 0x161FD \\RPM_AAAAANAAR(3)\lpr_definition_uber\rpm_halt_enter+0x1AD

we can see the function address 0x161FD just before the abort, so we can confirm this bitflip in 0x161F2 cause the crash.

3.2 PLL stuck during clock switch

This is usually linked to MSM chipset issue, please raise case to check further with QCOM

RPM logs

```
"rpm_err_fatal (lr: 0xffffffff9) (ipsr: 0x00000041)"
```

indicate this is a RPM WD bite Usually this is caused by PLL stuck

Callstack should be like this

-000|Hal_clk_DumpBIMCReg(?)
-001|HAL_clk_BIMCConfigMux(?, ?)

Or like this

-000|busywait()
-001|HAL_clk_GenericPLLEnableVote()

3.3 LDO timeout

Please Raise case to PMIC team for such kinds of failure, usually linked to PMIC or HW issue.

Callstack should be like this

```
-000|abort()
-001|pm_rpm_check_vreg_settle_status()
-002|pm_rpm_ldo_settle()
-003|pm_rpm_ldo_apply()
-004|resource_ee_apply(settling_time = 0xA4)
```

Get the failure LDO number from last RPM logs before err fatal

```
47364.130300: rpm_apply_request (resource type: ldoa) (resource id: 16)
47364.130314: rpm_apply_request (resource type: ldoa) (resource id: 16)
47364.130327: rpm_apply_request (resource type: ldoa) (resource id: 16)
47364.130340: rpm_apply_request (resource type: ldoa) (resource id: 16)
47364.132288: START Apply() VS0B
47364.132289: START Post-Dep() VS1039B
47364.132290: rpm_err_fatal (lr: 0x00013033) (ipsr: 0x00000000) - Normal Abort
```

3.4 Cannot Exit from Deep Sleep(VDD MIN)

Confirm device already enter VDD MIN from RPM log. sometimes from AP/Linux, we may think device is in sleep, but other subsystem such as Modem/LPASS may still active, so we need check RPM logs and search key words "VDD Minimization"

```
1016.141622: deep_sleep_enter: (mode: "VDD Minimization") (count: 8)
1016.141817: Clock PLL: GPLL0 Status = Disable
1016.141848: deep_sleep_enter_complete: (mode: "VDD Minimization"), (duration:
0x112aa644)
1016.142026: mpm_wakeup_time: (timetick: 0x000000049c0c80f9)
1031.143602: mpm_wakeup_ints (ints: 0x00000001 0x00000000)
```

Most likely this is a HW specific issue linked to retention voltage, please raise a HW case to follow up .

Meanwhile, we can do some test chanages from SW side to narrow down the root cause

3.4.1 Try mock VDD MIN mode

With the following changes, the device will only shut down XO but keep current vddcx/vddmx

```
[rpm_proc\core\power\sleep\src\sleep_perform.c]
-volatile boolean mock_vdd_min_only = FALSE;
+volatile boolean mock_vdd_min_only = TRUE;
```

If the issue cannot reproduced, we can confirm this is linked to VDD MIN (vddcx/vddmx), not linked to XO.

3.4.2 Bump up retention voltage of vddcx/vddmx

Take 8996 as examples(other latest platforms should be simliar)

```
rpm_proc\core\power\sleep\src\8996\sleep_target_config.c
static const uint32 vddcx_pvs_retention_data[8] =
{
/* 000 */ 600000+50000,
/* 001 */ 550000+50000,
/* 010 */ 500000+50000,
/* 011 */ 450000+50000,
/* 100 */ 400000+50000,
/* 101 */ 400000+50000,
/* 110 */ 400000+50000,
/* 111 */ 600000+50000
};

static const uint32 vddmx_pvs_retention_data[8] =
{
/* 000 */ 700000+50000,
/* 001 */ 650000+50000,
/* 010 */ 580000+50000,
/* 011 */ 550000+50000,
/* 100 */ 490000+50000,
/* 101 */ 490000+50000,
/* 110 */ 490000+50000,
/* 111 */ 490000+50000
};
```

50mv is just an example, you can try different bump up values and find out what is the voltage needed for that specific failure devices. If above Mock VDD MIN test is working, there must be some value of retention voltage can avoid the issues, because Mock VDD MIN is actually using the "current active voltage" as the retention voltage.

3.5 Cannot Enter Deep Sleep (VDD MIN or XO SHUTDOWN)

Usually we need to get a RPM dump and try to find out which client block the deep sleep

3.5.1 Cannot enter XO SHUTDOWN

If devices cannot even enter XO SHUTDOWN, we can check NPA log and find out which subystem block the XO shutdown mode

Find the Node "xo/cxo" and check all the request: 1 for "NPA_CLIENT_REQUIRED", in the following example, LPASS block the sleep

From RPM side, we can only find which subsystem block the sleep, after confirm the subsystem

Please raise a case to related subsystem area to check further which modules in that subsystem is the root cause. We cannot get details inside a subsystem from RPM point of view

```
: npa_resource (name: "/xo/cxo") (handle: 0x198848) (units: Enable) (resource max: 1)
(active max: 1) (active state 1) (active headroom: 0) (request state: 1)
: npa_client (name: WCSS) (handle: 0x195D28) (resource: 0x198848) (type:
NPA_CLIENT_LIMIT_MAX) (request: 1)
: npa_client (name: WCSS) (handle: 0x195CE8) (resource: 0x198848) (type:
NPA_CLIENT_REQUIRED) (request: 0)
: npa_client (name: MPSS) (handle: 0x195438) (resource: 0x198848) (type:
NPA_CLIENT_LIMIT_MAX) (request: 1)
: npa_client (name: MPSS) (handle: 0x1953F8) (resource: 0x198848) (type:
NPA_CLIENT_REQUIRED) (request: 0)
: npa_client (name: LPASS) (handle: 0x19CC30) (resource: 0x198848) (type:
NPA_CLIENT_LIMIT_MAX) (request: 1)
: npa_client (name: LPASS) (handle: 0x19CBF0) (resource: 0x198848) (type:
NPA_CLIENT_REQUIRED) (request: 1)
: npa_client (name: APSS) (handle: 0x198A60) (resource: 0x198848) (type:
NPA_CLIENT_REQUIRED) (request: 0)
: npa_change_event (name: sleep) (handle: 0x199760) (resource: 0x198848)
: end npa_resource (handle: 0x198848)
```

3.5.2 Cannot enter VDD MIN (can enter XO SHUTDOWN)

Usually we need a RPM dump to check further, since JTAG is not very easy to check deep sleep issue, we suggest customer to use a manually triggered dump. For example

Adding err fatal in vdd min enter to get a normal dump

1 Add the following line in vdd_min_enter()

```
ERR_FATAL( "err fatal ", 0, 0, 0 )
```

2 if vdd_min_enter() do not hit, add it in rpm_halt_enter (should add some delay because it will hit very early before the test scenario you expect)

```
+static uint64_t t1;

+if(! t1)

+ t1 = time_service_now()

if(sleep_deep_imminent())

{

vdd_min_enter(mock_vdd_min_only);

vdd_min_exit(mock_vdd_min_only);

}

else

{

+if (time_service_now() -t1 > 0x80000000) //0x80000000 is just an example, about 2 minutes, you can adjust the time

+ERR_FATAL( "err fatal ", 0, 0, 0 );

rpm_halt_enter();

rpm_halt_exit();

}
```


After you get a RPM dump, please refer to section2 for how to load RPM dump, and then check

```
railway.rail_state[0] //VDDMX

railway.rail_state[1] //VDDCX
```

Walk through all voter_link and find the voltage_corner which is higher than RAILWAY_RETENTION

and meanwhile the suppressible = FALSE (if it is true, we can ignore the vote)

if you find any vote like I said above, check the master_id of it .it will tell you which subsystem hold the vote for voltage. and then we can raise related case to related subsystem (from RPM point of view, we can only find the subsystem, details need subsystem analysis)

> If id < 100, it's the EE# (master ID): 0(APPS) / 1(MODEM) etc;
> If id >= 100, see definition of railway_voter_id

I give you an example, the following example show it is TZ which cause the voltage cannot drop to retention

finally we find it is a TZ related sensor problem of AP side

```
railway.rail_state[1] = (
• current_active = (mode = RAILWAY_SVS_SOC, microvolts = 840000),
• ...
• voter_list_head = 0x0019ED40 -> (
• ...
• voter_link = 0x0019DAF8 -> (
• voltage_corner = RAILWAY_SVS_SOC, <-- it is higher than RAILWAY_RETENTION !!!
• active_floor = RAILWAY_NO_REQUEST,
• explicit_voltage_in_uv = 0,
• suppressible = FALSE, <-- Required voter !!!
• id = 4, <-- ID 4 means it is TZ
```


3.6 Wake up subsystem late

RPM will check deadline offered by subsystem when it need to wakeup subsystem, if the deadline is exceeded, RPM will throw err fatal to avoid further issues.

Call stack should be like this

```
-000|abort()
-001|isWaking(inline)
-001|SetChanger::execute_until(
```

```
-002|time_service_now(inline)
-002|Task::execute(
-003|Sched::run()
-004|time_service_now(inline)
-004|main()
```

Mostly, we can find some clue in the RPM logs, This is usually caused by the following two reasons

3.6.1 Example - Delayed by LDO actions

Customer need raise a PMIC case for such issue

1 A wake up late of Modem system ,from call stack, we can see RPM assert when wake up late than expected

```
SetChanger::execute_until
-000|abort()
-001|isWaking(inline)
-001|SetChanger::execute_until(
| preempt = FALSE,
| stop_time = 0xFFFFFFFFFFFFFFFF)
| settling_time__16 = 0x0
| now = 0x000000013E232C08
| old_deadline = 0x000000013E21FA6D
| return of theEstimateCache = (cache_ = 0x0, cacheSize_ = 0x0, cacheSeq_ =
| return of resource_ee_has_transition_work = 0xE3
| return of Handler::processNAS = 0xE3
-002|time_service_now(inline)
-002|Task::execute(
| preempt = FALSE,
| stop_time = 0xFFFFFFFFFFFFFFFF)
-003|Sched::run()
| stopping_time = 0xFFFFFFFFFFFFFFFF
| schedule_changed = 0x0F
| nextState = 0x20
| __result$$2 = 0x10
| __result$$3 = 0x10
-004|time_service_now(inline)
-004|main()
```

2 RPM logs

```
277.983317: rpm_apply_request (resource type: ldoa) (resource id: 19)
277.983384: rpm_apply_request (resource type: clk0) (resource id: 0) (full name: CX0)
277.983402: rpm_apply_request (resource type: ldoa) (resource id: 12)
277.983452: rpm_apply_request (resource type: ldoa) (resource id: 32)
277.983464: rpm_apply_request (resource type: ldoa) (resource id: 4)
277.983482: rpm_resource_settling_complete (master: "MSS SW") (resource type: ldoa) (
resource id: 4)
277.983484: rpm_apply_request (resource type: ldoa) (resource id: 15)
277.983501: rpm_resource_settling_complete (master: "MSS SW") (resource type: ldoa) (
resource id: 15)
277.983504: rpm_apply_request (resource type: ldoa) (resource id: 16)
277.992596: START Apply() VS0B
277.992600: rpm_resource_settling_complete (master: "MSS SW") (resource type: ldoa) (
resource id: 16)
277.992602: rpm_apply_request (resource type: smpb) (resource id: 1)
277.992662: rpm_resource_settling_complete (master: "MSS SW") (resource type: smpb) (
resource id: 1)
277.992664: rpm_apply_request (resource type: ldoa) (resource id: 1)
277.992692: rpm_resource_settling_complete (master: "MSS SW") (resource type: ldoa) (
resource id: 1)
277.992695: rpm_master_set_transition_complete (master: "MSS SW") (deadline:
0x000000013e21fa6d) (exceeded: yes)
277.992710: rpm_bringup_req (master: "MSS SW") (core: 0)
277.992715: rpm_bringup_ack (master: "MSS SW") (core: 0)
277.992718: rpm_err_fatal (lr: 0x00108e41) (ipsr: 0x00000000) - Normal Abor
```

we can see this late is due to LDO16 apply operation
the dead line is 0x000000013e21fa6d (277.9886), all other actions had no delays and cost very little time
expect LDO16, if LDO16 complete as others (less than 1ms), it will not exceed the deadline 277.9886

```
277.983504: rpm_apply_request (resource type: ldoa) (resource id: 16)
277.992596: START Apply() VS0B
```

but actually , it costed about 9ms. and let the modem wakeup late

3.6.2 Example - subsystem send an obvious invalid deadline info

Usually this is caused by SW issue of related subsystems, customer need raise customer to related subsystem such as AP/Modem..

> 139.040161: rpm_master_set_transition_complete (master: "APSS") (deadline: 0x0000000000000000) (exceeded: no)
> 139.040180: rpm_transition_queued (master: "APSS") (scheduled: "yes") (deadline: 0x000000009f1c90d2)
> 139.040189: rpm_svs (mode: RPM_SVS_FAST) (reason: imminent processing) (current_speed: 400000)
> 139.040193: rpm_master_set_transition (master: "APSS") (leaving: "Sleep Set") (entering: " Active Set")
> 139.040227: rpm_master_set_transition_complete (master: "APSS") (deadline: 0x000000009f1c90d2) (exceeded: yes)
> 139.040230: rpm_err_fatal (lr: 0x00008813) (ipsr: 0x00000000) - Normal Abort

we can see the deadline is exceed, deadline is 0x000000009f1c90d2, about 139.0338 . so it is even much earlier than the time when AP start to wakeup.(139.040193).

In this example, we have known CRs in Linux side. need raise AP case to request the CR