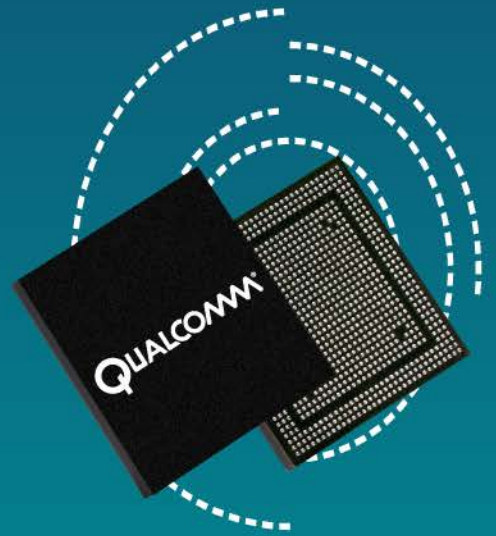
Qualcomm
TECHNOLOGIES, INC



MSM8974 Dynamic Sleep Overview

80-NA157-48 B

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

Revision History

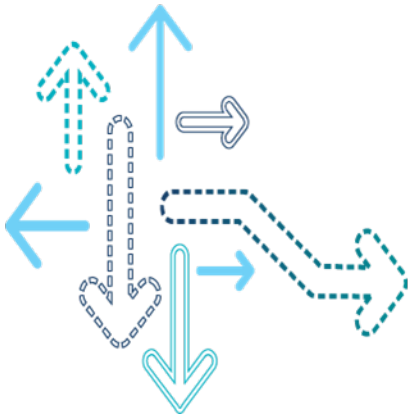
Revision	Date	Description
A	Jul 2012	Initial release
B	Aug 2013	Added subsystem and RPM sleep flow

Contents

Acronyms

- RPM – Resource Power Manager, a processor dedicated to managing shared resource and power
- NPA – Node Power Architecture, framework for managing system resources
- MPM – Multiprocessor Power Manager, hardware block that remains on during XO down and VDD minimization
- LPR – Low Power Resource, a resource cannot be shut off until CPU is halted
- LPRM – Low Power Mode, the low mode an LPR can enter
- DEM – Dynamic Environment Manager, a module that manages apps sleep in MSM7xxx, MDM9xxx; no longer used in MSM8974

Introduction



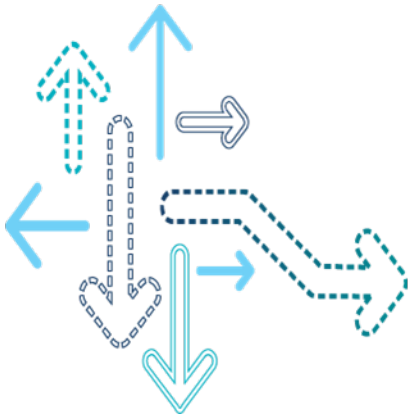
Sleep Mechanism

- Sleep mechanism in MSM™
  - Conserves power by shutting off hardware/software components and drawing minimum current when device is idle
  - Involves various hardware/software modules, such as clock regime, interrupt, PMIC, MPM, RPM (DEM in MSM7xxx and MDM9xxx), etc.
  - Requires other software modules to indicate whether it is OK to sleep and how long it can sleep
- Sleep in MSM8974 compared to MSM7x30 or MDM9xxx
  - With RPM, a processor dedicated to shared resource and power management, sleep processes in modem, apps, and LPASS become independent of one another in MSM8974
  - The dynamic sleep scheme in MSM8974 relies on LPR states to decide on the optimal low power mode to enter; this offers more flexibility comparing to the plain “OK to sleep” voting method

Dynamic Sleep

- Benefits of Dynamic Sleep scheme
  - Puts ownership of LPRs under resource owners' control; this removes intimate knowledge of LPRs and their properties from the idle thread.
  - Idle thread becomes a common algorithm, and LPRs can be dynamically registered based on the target.
  - It adds flexibility for determining which low power modes to enter based on latency and power savings, and leaves room for optimized power savings by improving the algorithm that chooses the low power modes.
  - It adds the capability for granular unit testing of individual and combinations of low power modes.
  - It allows clients to indicate their requirements on exactly which resources they depend.
    - Modes are automatically chosen to meet client needs.
    - It removes intimate knowledge of low power modes from clients.
  - Each target can be fully optimized independently of other targets.
    - New resource data is measured for every target.

Low Power Resources



Low Power Resource

- LPR
  - A resource that must wait until the CPU is idle before entering its low power mode, e.g., XO, VDD, etc.
  - LPRs are registered with idle thread (/sleep/lpr node) at runtime
- Low Power Resource Modes (LPRMs)
  - Low power modes that are associated with a certain LPR
  - Dynamic sleep chooses at most one LPRM for every LPR, based on latency and power savings properties
  - Each LPRM must provide:
    - Time to enter/exit the mode
    - Functions to enter/exit the mode
    - Power saved by entering the mode

An LPR Example – vdd_dig

- Definition of SleepLPR_vdd_dig in MSM8974 Hexagon™ modem software subsystem
  - static sleep_lprm SleepLPR_vdd_dig_modes[] =

```
{
  {
    "min",
    "cxo.shutdown",
    "BEFORE DEP vdd_dig.min",
    SLEEP_POW_FUNC_POLY_DURATION(&vdd_dig_min_Savings_fn_coeffs),
    SLEEP_LAT_FUNC_CONST(42000),
    SLEEP_LAT_FUNC_CONST(4200),
    vdd_dig_min_enter,
    vdd_dig_min_exit,
    SLEEP_BACKOFF_CONST(4200)
  },
};
```

  - sleep_lpr SleepLPR_vdd_dig =

```
{
  "vdd_dig",
  CORE_ARRAY(SleepLPR_vdd_dig_modes ),
  SLEEP_ALL_CORES
};
```


An LPR Example – vdd_dig (cont.)

- The following properties of vdd_dig LPR can be deduced from the definition:
  - There is one LPRM for vdd_dig, min
  - It has dependency on other LPR cox.shutdown
  - Its power saving can be calculated based on a polynomial power function
  - Its enter/exit latencies
  - Its enter/exit functions

Register to and Request for an LPR Resource

- For LPRs that are defined as NPA nodes, NPA APIs can be used to register to and request for the resource.
- An example based on PXO resource is:
  - Initialize the client by registering with the LPR

```
reqClient = npa_create_sync_client(
            "/xo/pxo",          // Connect to PXO resource
            "Client Name",      // Name describing the client
            NPA_CLIENT_REQUIRED, // Your client type
            NULL);              // User Data
```

  - Request for the resource

```
npa_issue_required_request(reqClient, 1);
// 1 indicates the resource is needed
```

  - Release the resource

```
npa_complete_request(reqClient);
```


Some LPRs in MSM8974

- Some LPRs and their LPRMs in MSM8974 are:
  - CXO LPR – SleepLPR_cxo
    - shutdown (via RPM)
  - Vdd Digital/Mem LPR – SleepLPR_vdd_dig
    - min (via RPM)
  - CPU Vdd LPR – SleepLPR_cpu_vdd
    - pc_l2_noret
    - pc_l2_ret
  - RPM LPR – SleepLPR_rpm
    - sync

Sleep LPR Node

- Sleep LPR is an NPA node.
  - It is the dependency node for NPA resources that are LPRs.
  - It allows clients (NPA resources) to enable/disable their LPRMs dynamically at runtime.
  - Idle algorithm queries this node to get enabled LPRMs.

```
static npa_node_definition sleep_lpr_node =
{
  "/node/sleep/lpr", /* name */
  sleep_lpr_driver,  /* driver_fcn */
  NPA_NODE_DEFAULT,  /* attributes */
  NULL,              /* data */
  NPA_EMPTY_ARRAY,
  NPA_ARRAY(sleep_lpr_resource)
};
```

- All LPR state information is saved in the following internal low power resource registry that is queried through the sleep LPR node.

```
static CLprRegistry *gLprRegistry;
```

- The synthesized sleep modes are stored in SleepLPR_synth, and used for computing optimal low power modes to enter.

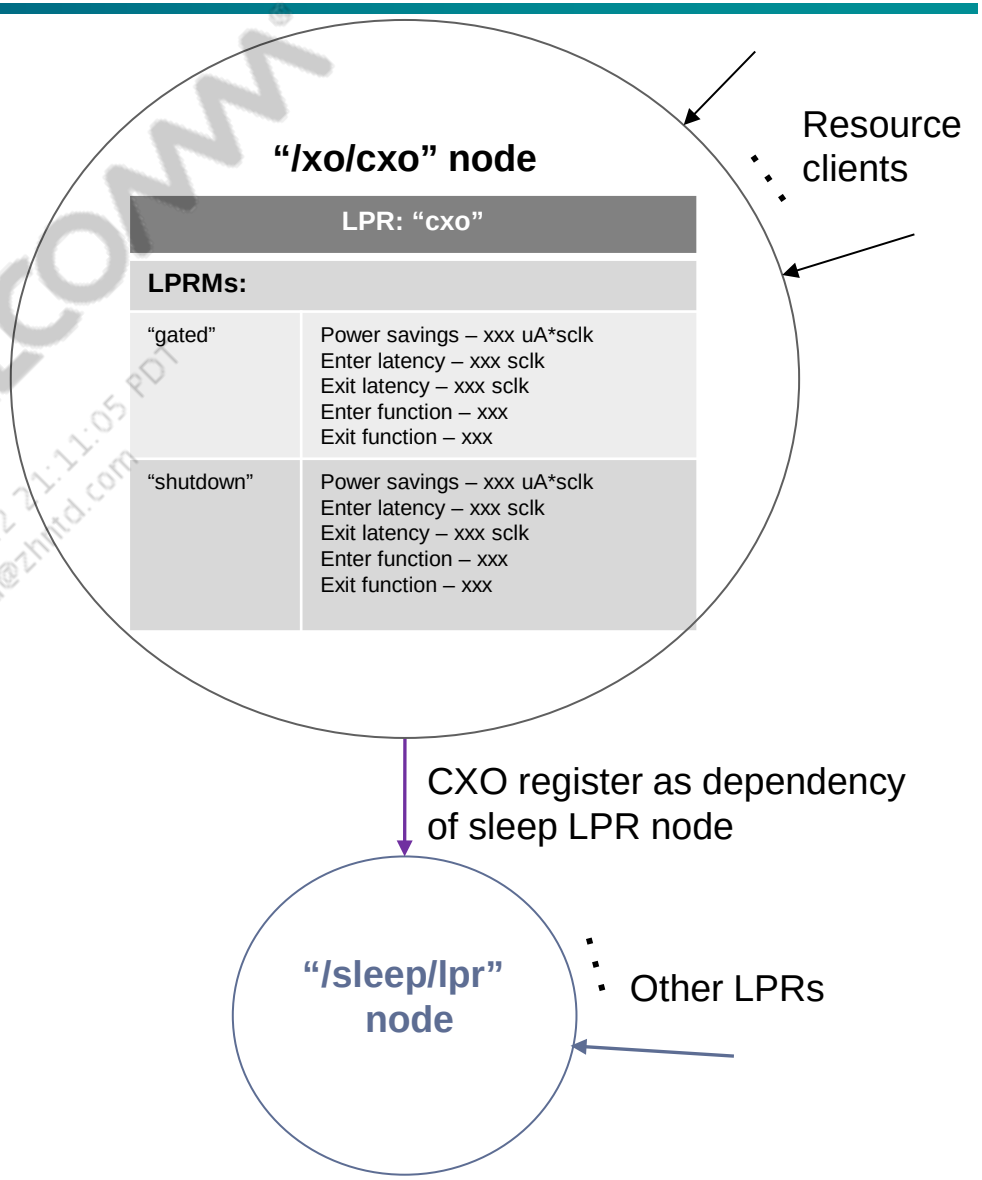
```
sleep_lpr_synth SleepLPR_synth;
```

Latency and Wake-Up Nodes

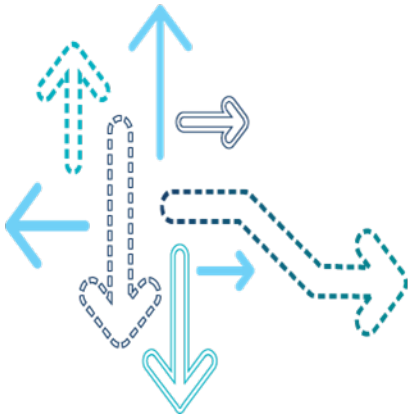
- Latency node (sleep_latency_node)
  - Allows a client to specify a desired interrupt latency, i.e., specify a certain amount of time from the time the interrupt fires to the time the ISR runs
  - When choosing LPRMs during idle, this latency will be honored
  - Client must cancel the latency request when it is no longer valid
- Wake-up node (sleep_wakeup_node)
  - Hard wake-up
    - Time until the next scheduled wake-up (Sleep Controller, timer, etc.)
    - When choosing LPRMs, the hard wake-up deadline must be met
  - Soft wake-up
    - Allows a client to indicate that a wake-up interrupt is expected in a certain amount of time
    - Prevents wasting time by entering modes just to exit them soon
    - Client hints are used when determining which modes to enter
    - Use of this node is an optimization, not a requirement

LPR/LPRM Example

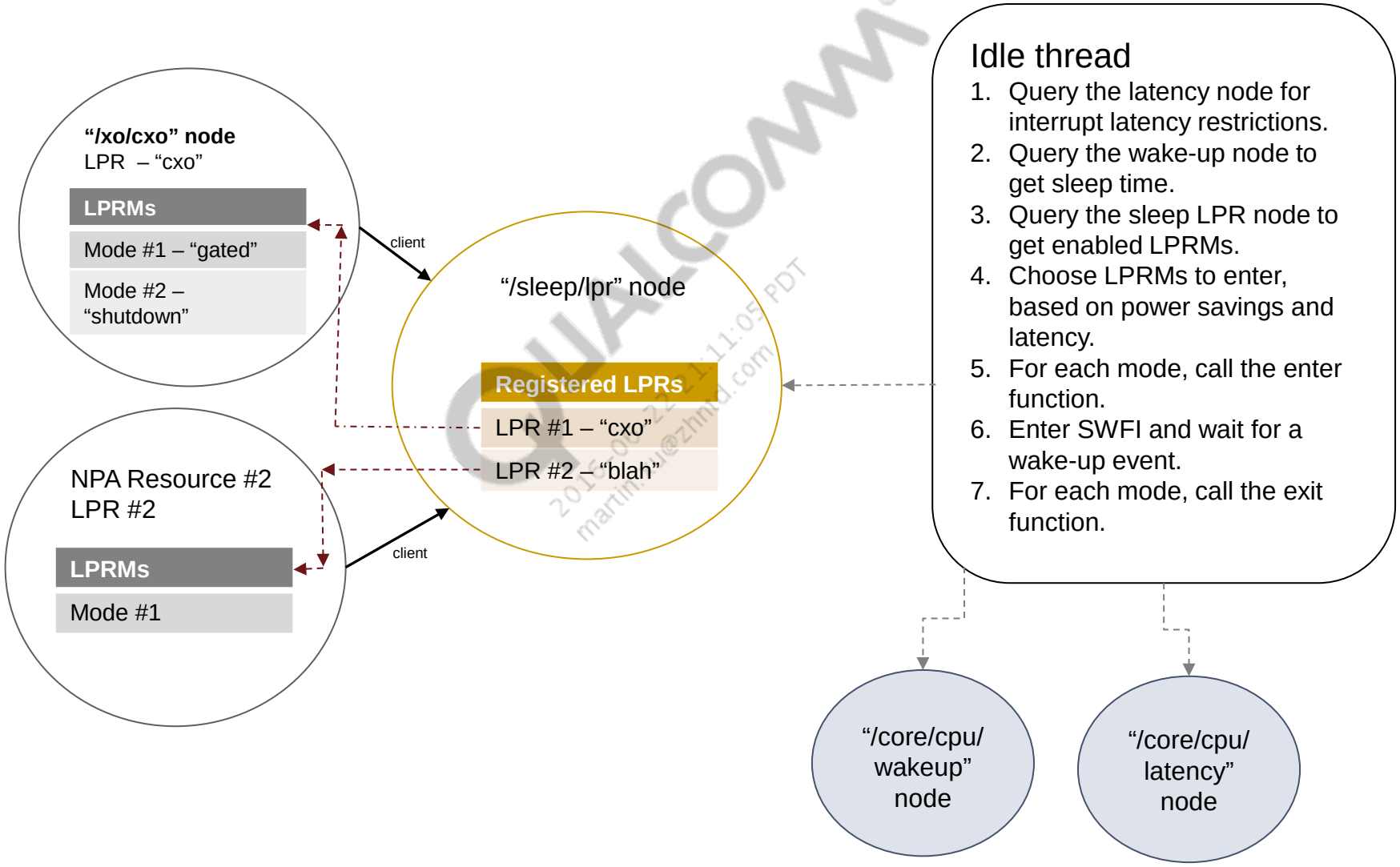
- LPR nodes
  - The LPR node should enable its LPRMs when there is no request disallowing sleep from entering the low power mode.
  - The LPR node should disable its LPRMs when there are requests that require the resource to remain on, even during sleep.
- Sleep LPR node
  - NPA LPRs register as dependency of the /sleep/lpr node.
  - Non-NPA LPRs register as clients of the sleep/lpr node.
  - The sleep/lpr node is used to query the current state of all LPRs, i.e., what LPRMs are enabled.



Dynamic Sleep Process



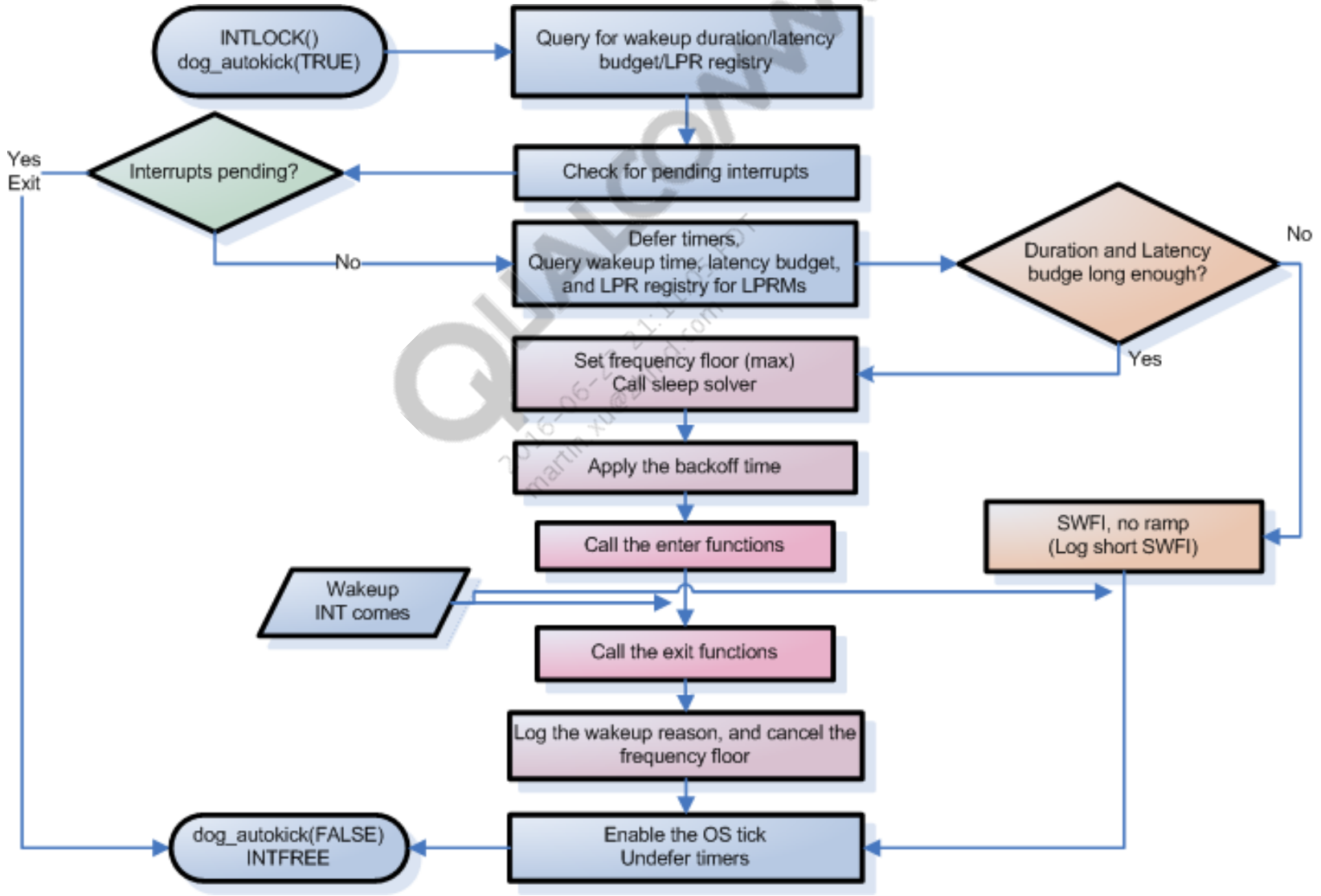
LPRs and Sleep Process



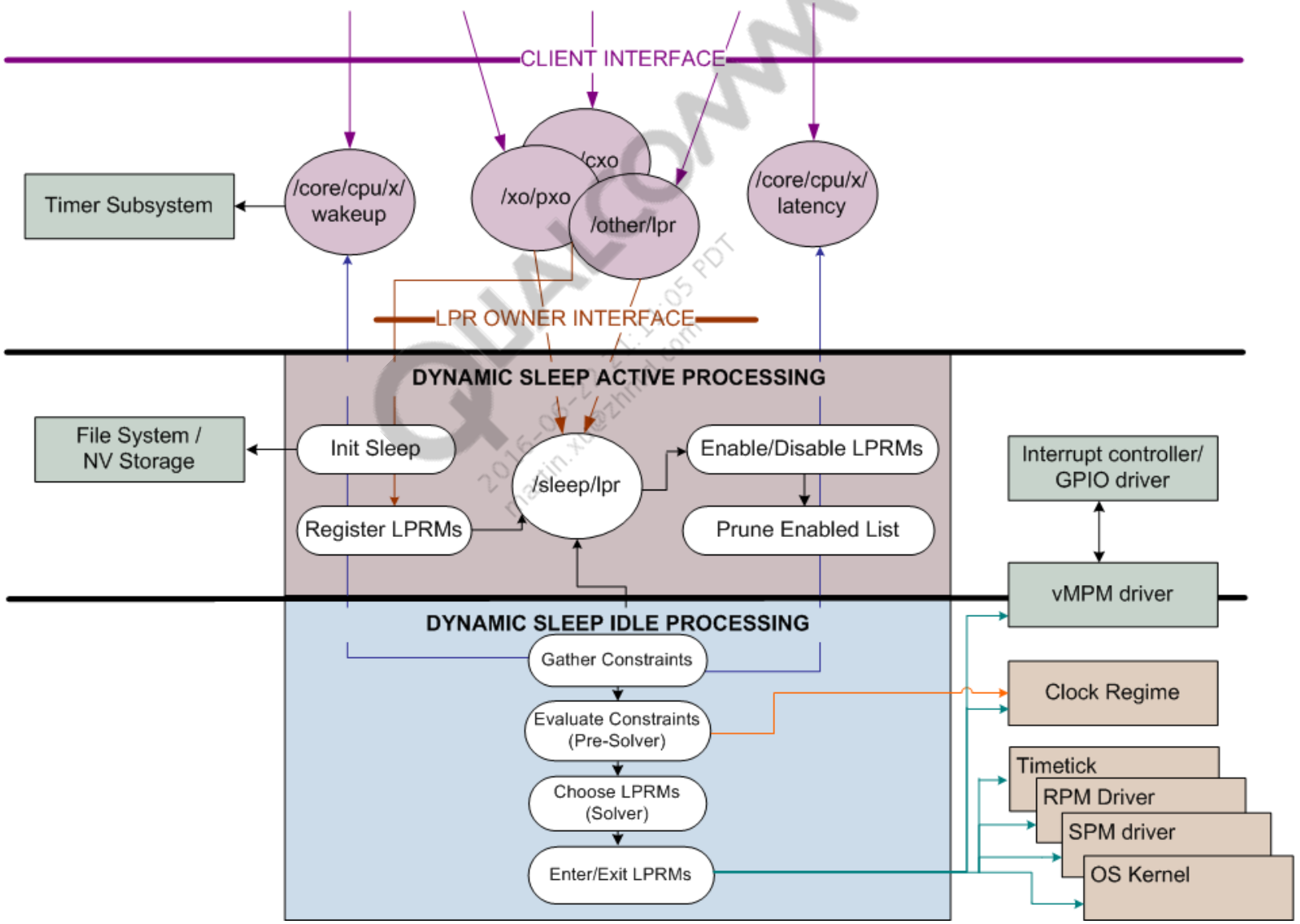
Sleep Flow in MSM8974 Hexagon Modem Software

- Sleep process in sleep_perform_lpm goes through these main steps:
  - Checks whether we are allowed to sleep based on some global switches
  - Puts dog on auto-kick and disables deferrable timers
  - Queries wake-up node for hard/soft durations and latency node for delay budget
  - Queries sleep LPR node for enabled LPRMs
  - If there is sufficient time, runs the sleep solver to compute the optimal LPRMs based on current LPR states, sleep duration, and latency budget
  - Runs the enter functions of picked LPRMs to enter sleep
  - Wake-up interrupt comes
  - Runs the exit functions of picked LRPMs to exit sleep
  - Logs wake-up reason
  - Enables deferred timers and disables dog auto-kick

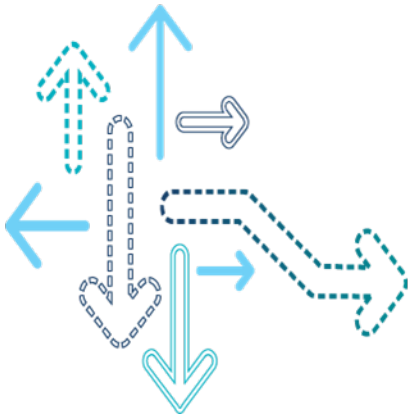
Dynamic Sleep Flow Block Diagram



Another View of Dynamic Sleep Process



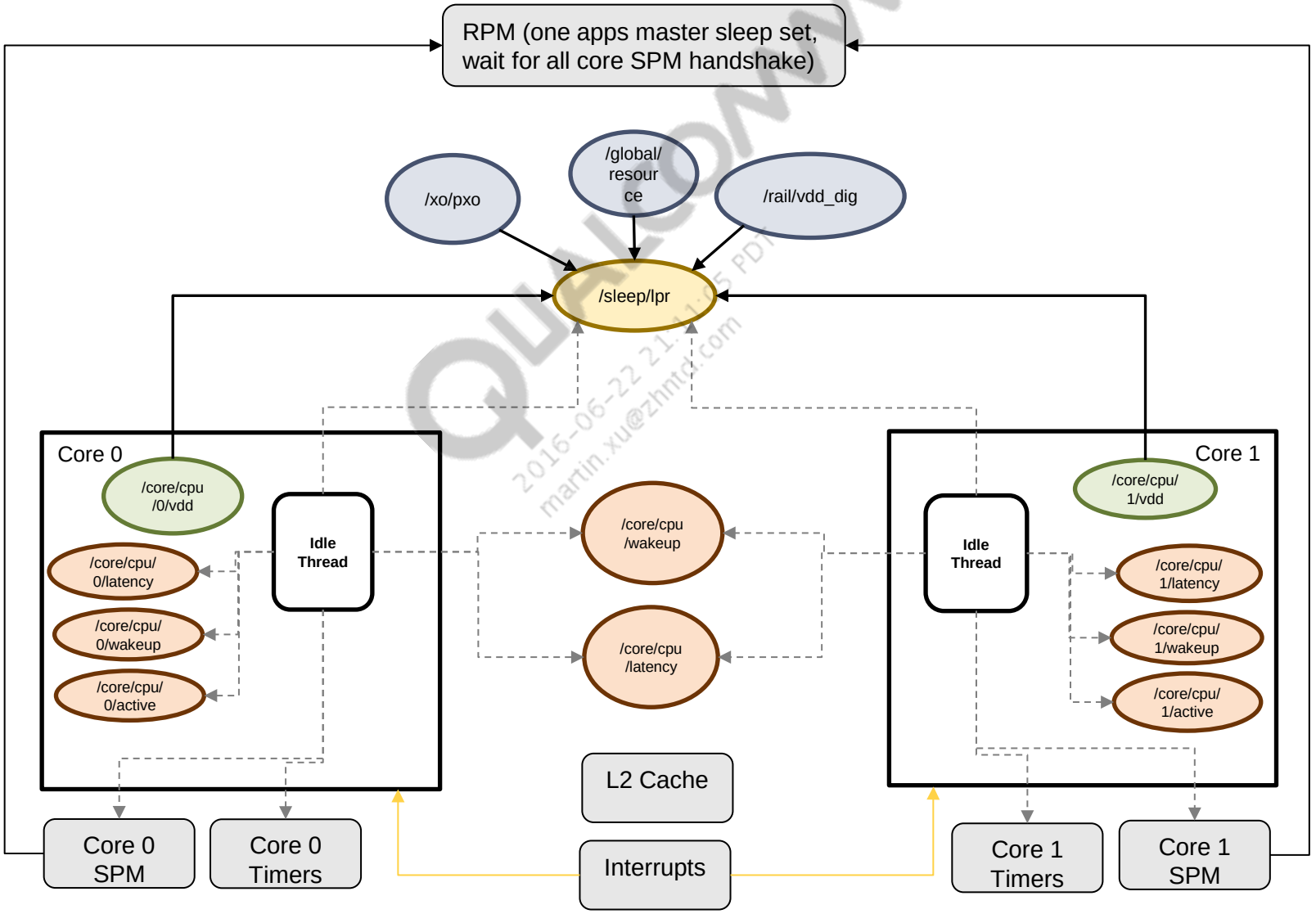
Multicore and Multiprocessor Sleep



Multicore Sleep

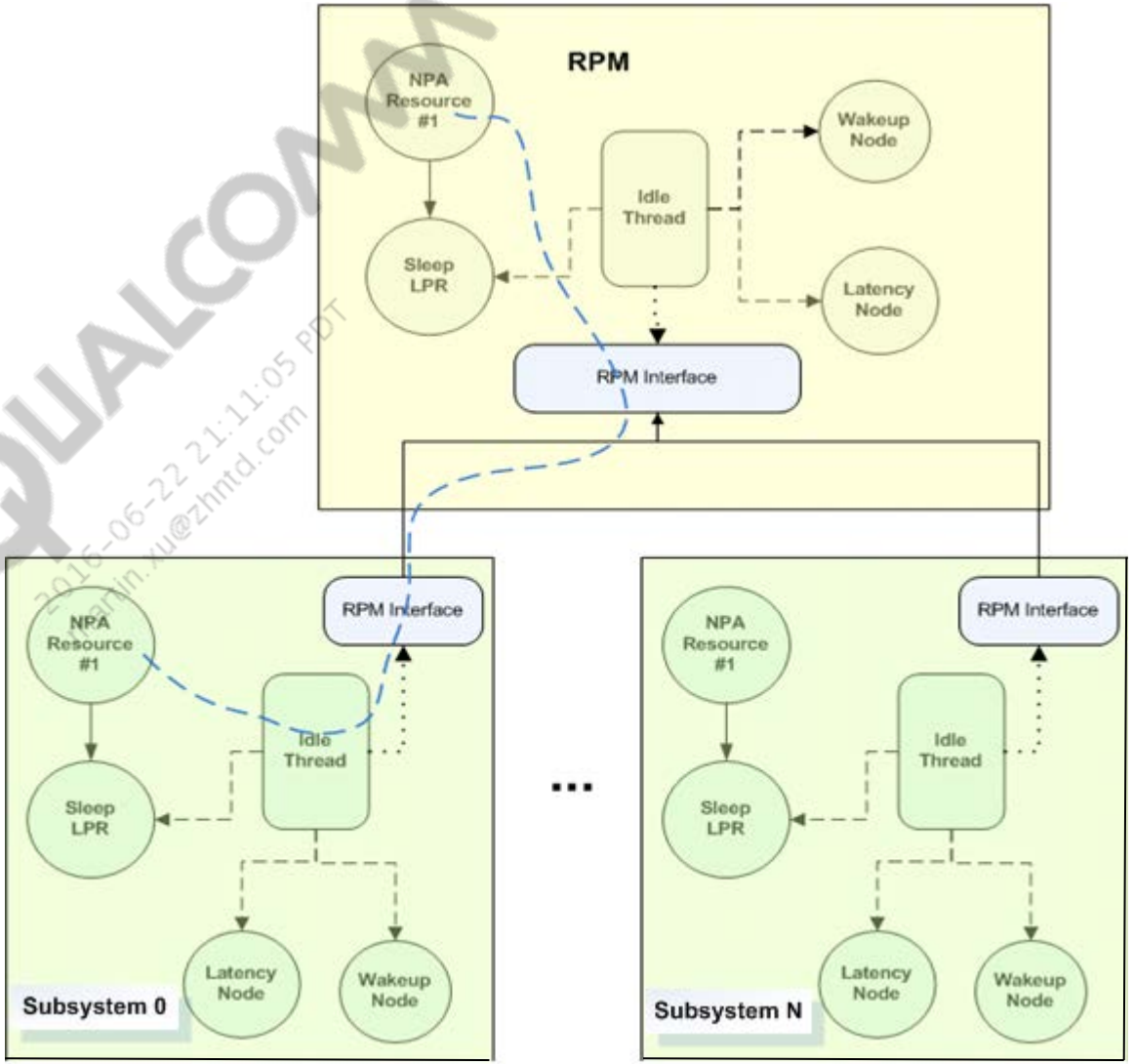
- An example of multicore is the application subsystem in the MSM8974, which contains two cores, 0 and 1.
- The idle thread runs independently on each core and chooses low power modes for the subsystem.
- Each core goes in and out of power-collapse/halt independently.
- The last core to go down communicates global low power modes to the RPM. Any core can be the last to go down or the first to come up.
- Any interrupt will wake up the core. The sleep code will ensure that a low power mode is not entered in which your interrupt will not be received.
- There is a local latency node on each core (/core/cpu/x/latency), as well as a global latency node (/core/cpu/latency).

Multicore Sleep View

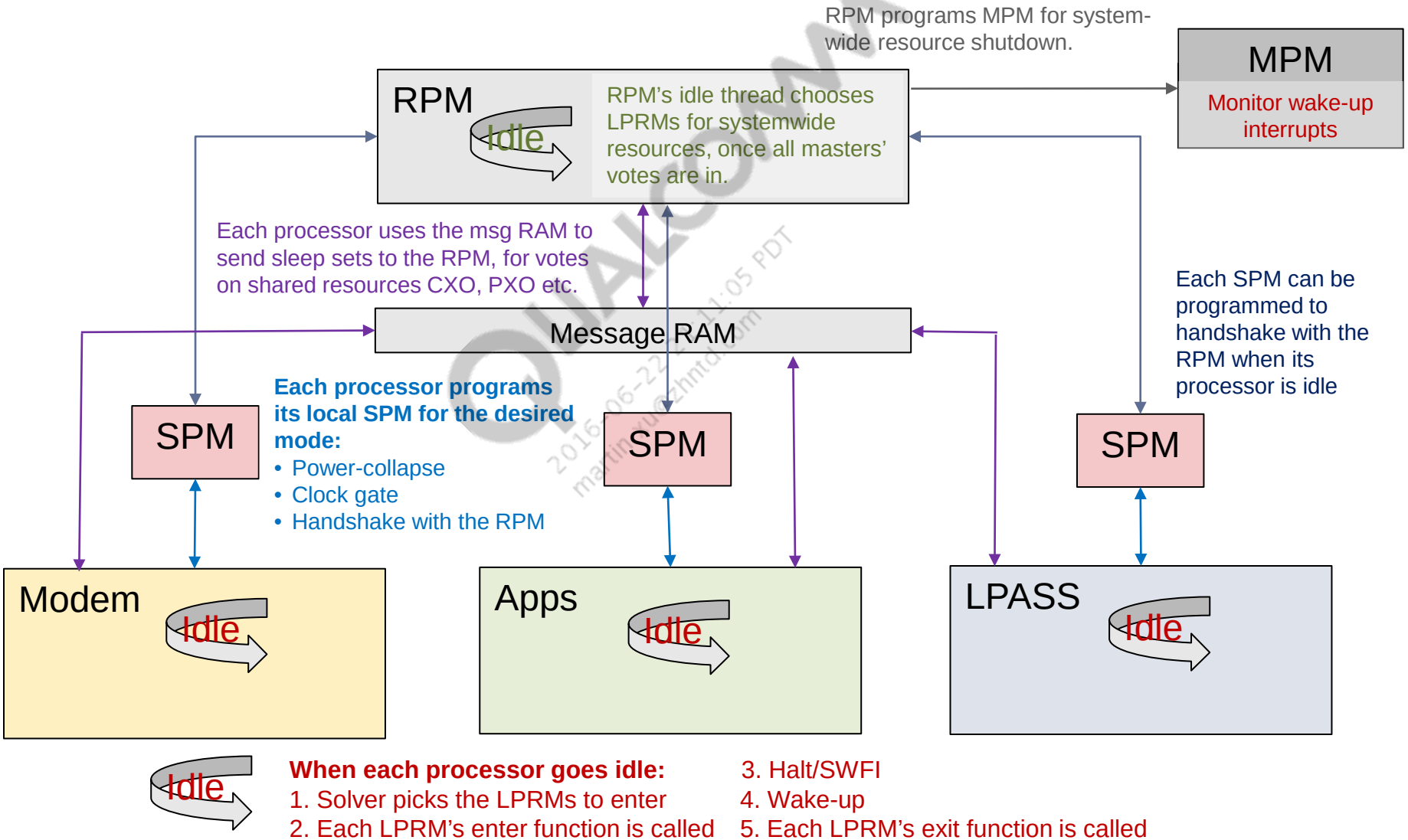


Multiprocessor Sleep View

- An example of a multiprocessor system is the modem, LPASS, and apps in the MSM8974.
- Each subsystem has its own idle thread and LPRs.
- Each subsystem sleeps independently of one another.
- Shared resources such as CXO and VDD are handled by RPM and requested by these systems through the RPM interface.



MSM8660 Multiprocessor Sleep System



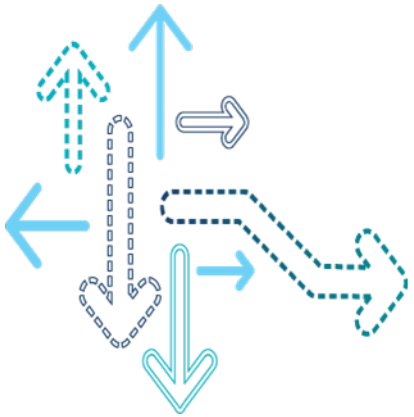
Some Guidelines on Using Dynamic Sleep

- Sleep voting no longer exists. In general, drivers must request the resources that they require. LPRMs are enabled/disabled based on resource requests.
- Latency
  - If a driver has a latency requirement from the time their interrupt fires to the time the ISR runs, it must be registered with sleep.
  - Latency requests must be canceled when they no longer apply.
  - If an interrupt has an affinity to a specific core, request latency on that core's latency node. Otherwise, request latency to the global latency node.
- Timers
  - Do not use polling timers.
  - Use deferrable timers where possible.

Some Guidelines on Using Dynamic Sleep (cont.)

- Clocks/bus
  - If a driver needs a certain clock to be on, request that through clock regime or the NPA interface. Just request the specific clock, not the XO.
  - When a clock is no longer in use, turn it off.
  - The LPRMs will be enabled/disabled based on these requests.
  - If a clock is on and should not prevent sleep, it should be made a suppressible request. This will tell the node not to consider this request during sleep. (Note: The resource driver must support the suppressible request for this to work.)
- Interrupts
  - Each core will exit low power modes for any interrupt that occurs.
  - All interrupts are automatically wake-up interrupts.
  - The sleep code will ensure that a low power mode is not entered in which your interrupt will not be received.

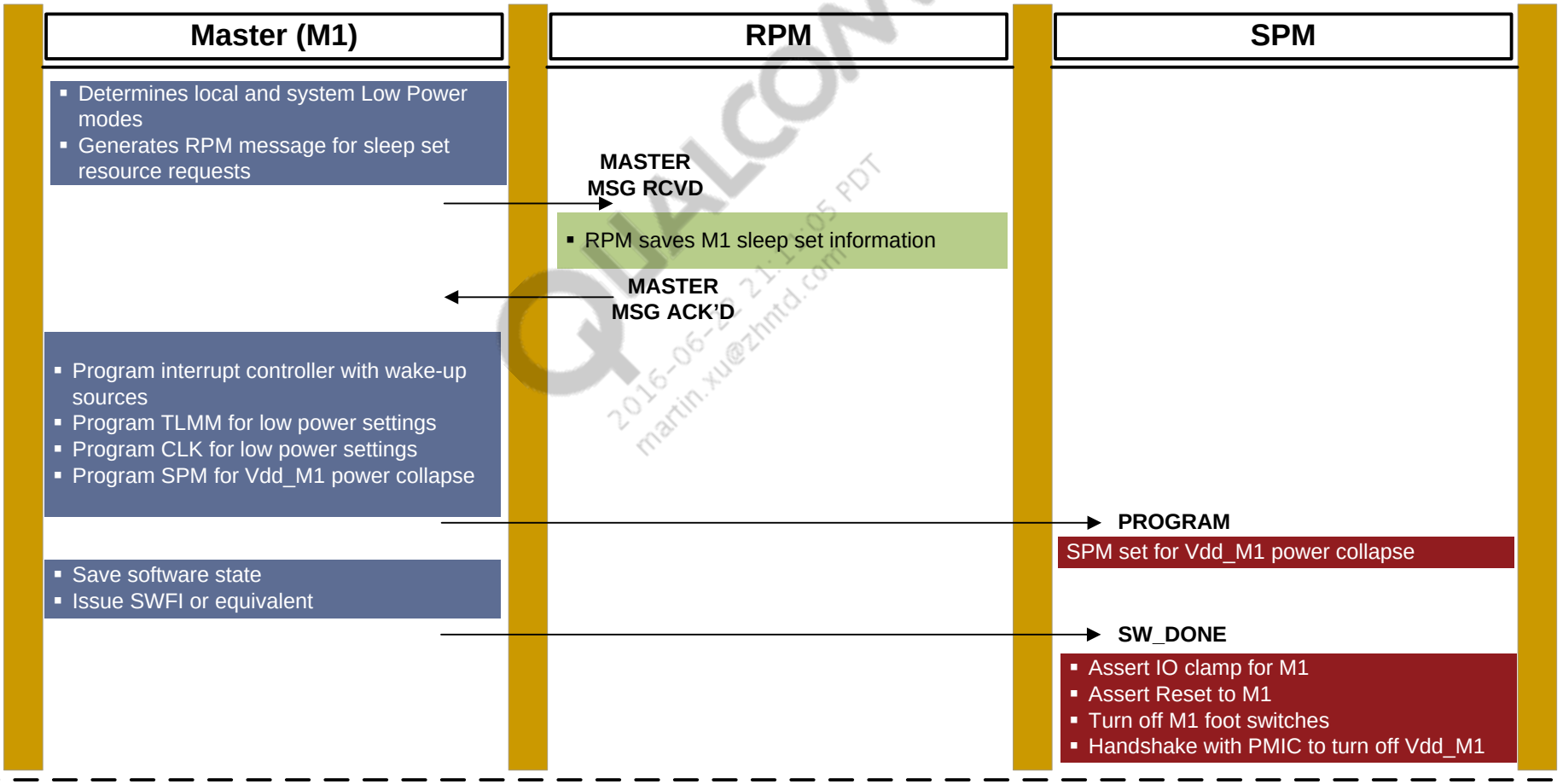
Subsystem and RPM Sleep Flow



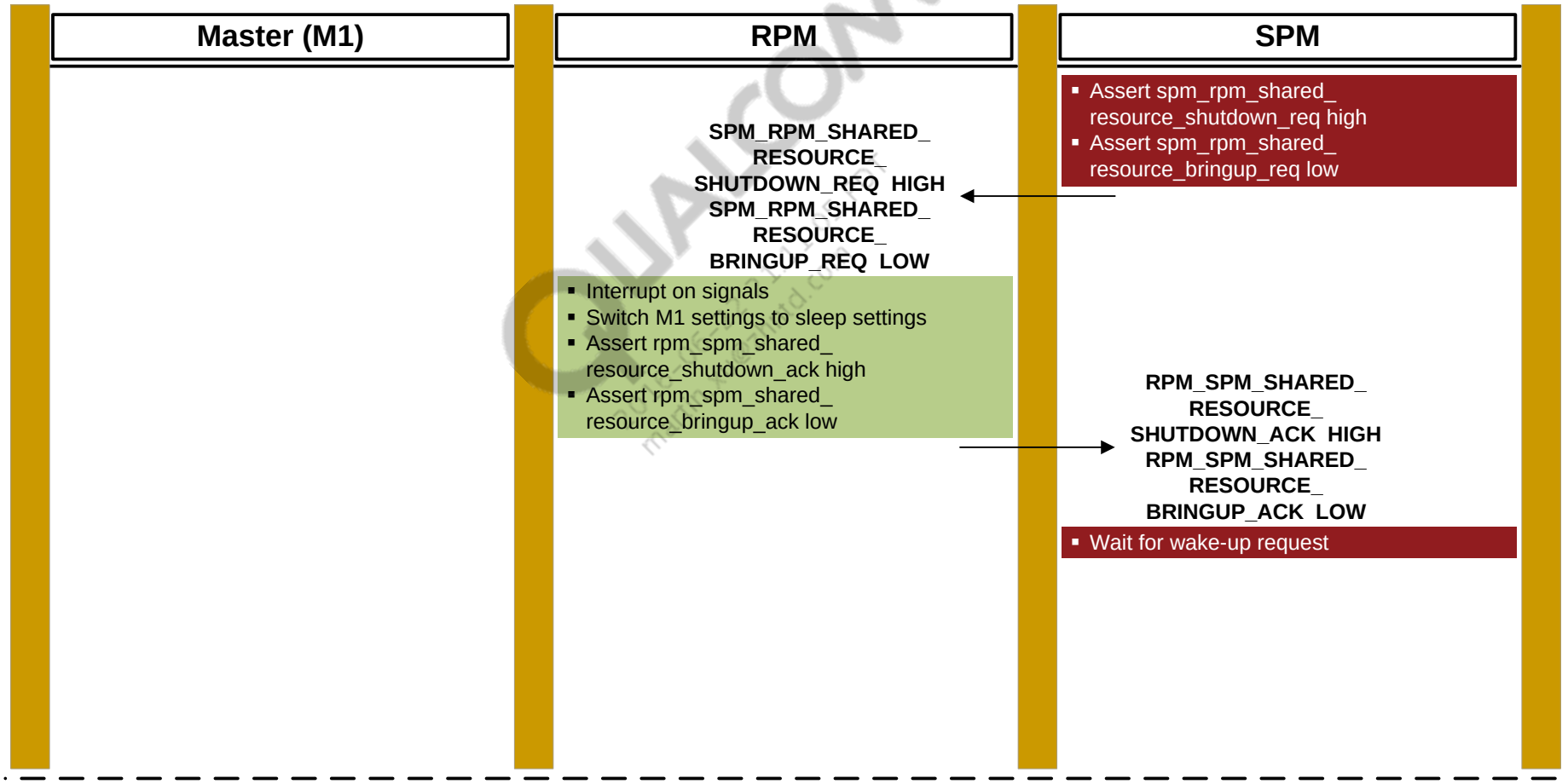
Subsystem and RPM Sleep Flow

- The interaction among the subsystems (SPM), RPM, and MPM during the following processes is described in this section.
  - Subsystem power collapse
  - Deep sleep process – CXO shutdown and VDD MIN

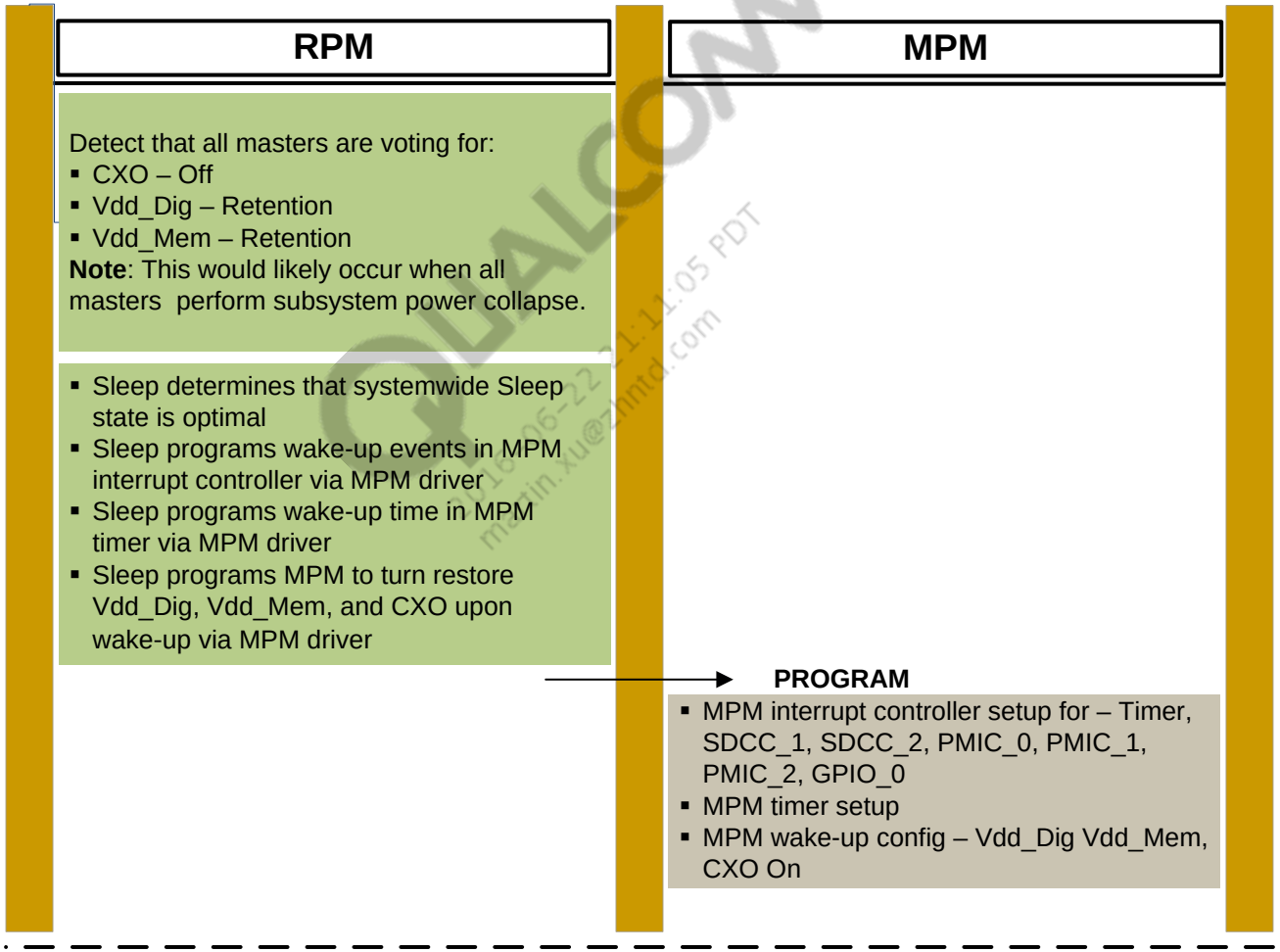
Subsystem Core Power-Collapse Flow



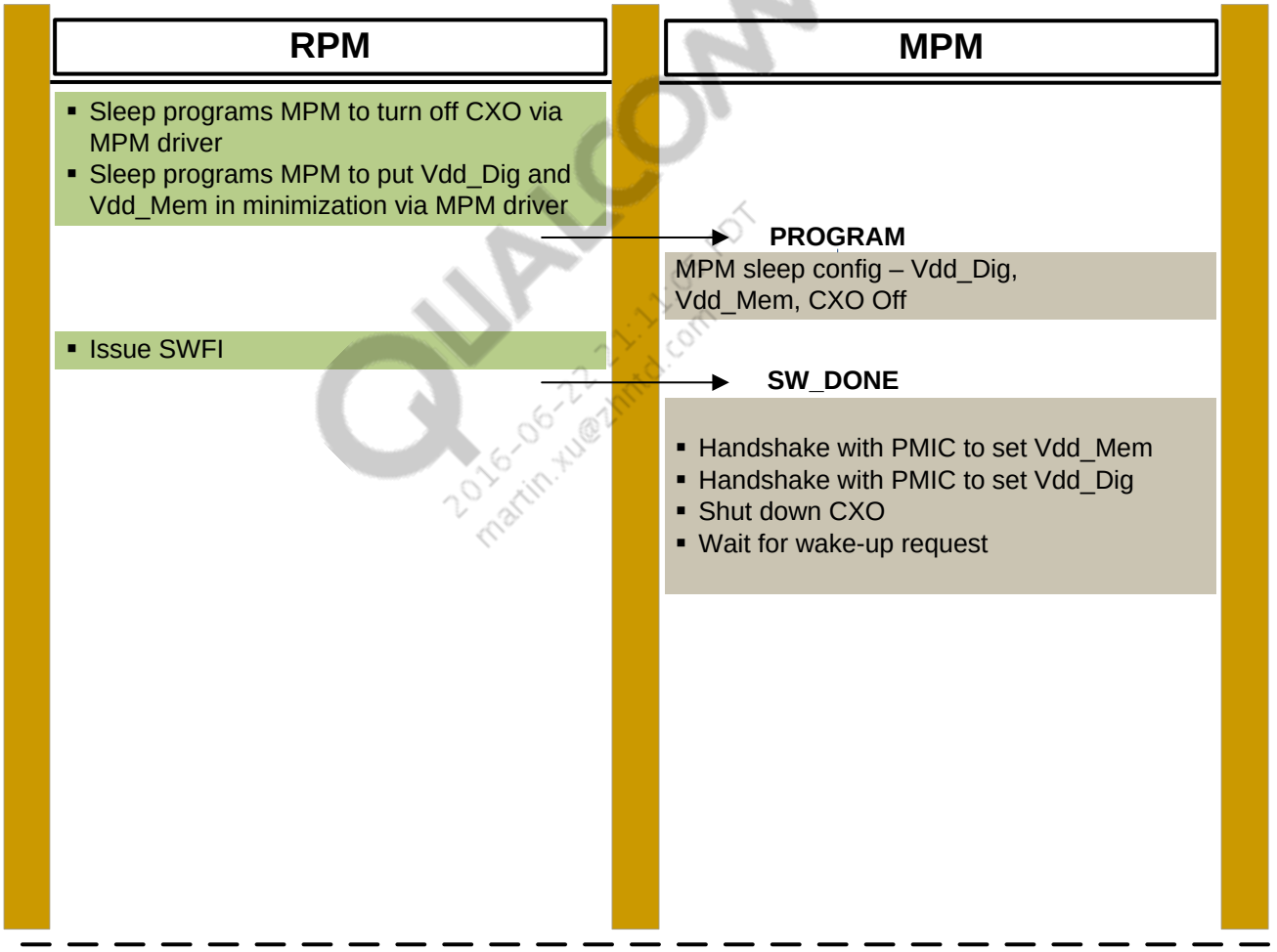
Subsystem Core Power-Collapse Flow (cont.)



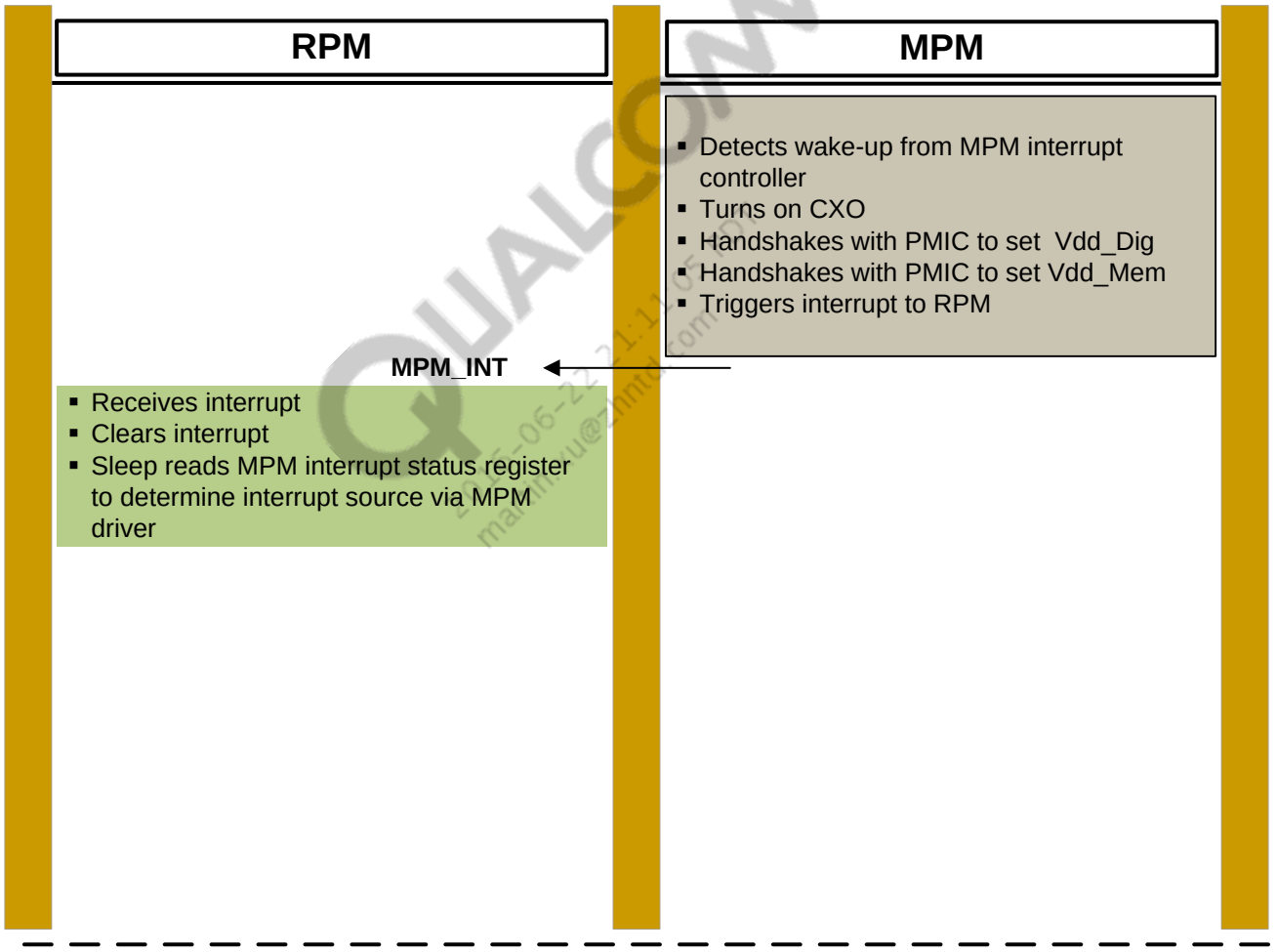
XO Shutdown/System Vdd_Min Flow



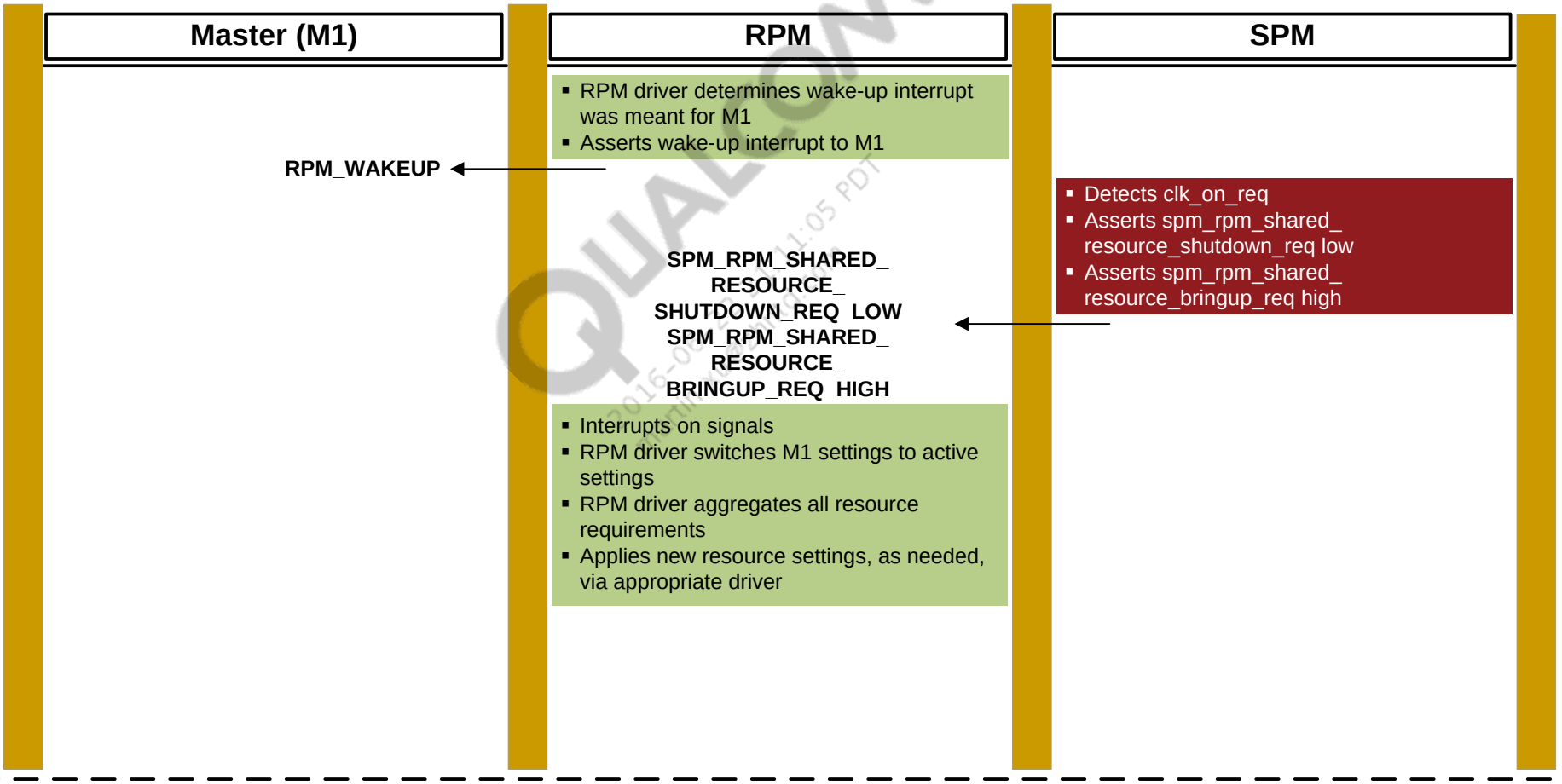
XO Shutdown/System Vdd_Min Flow (cont.)



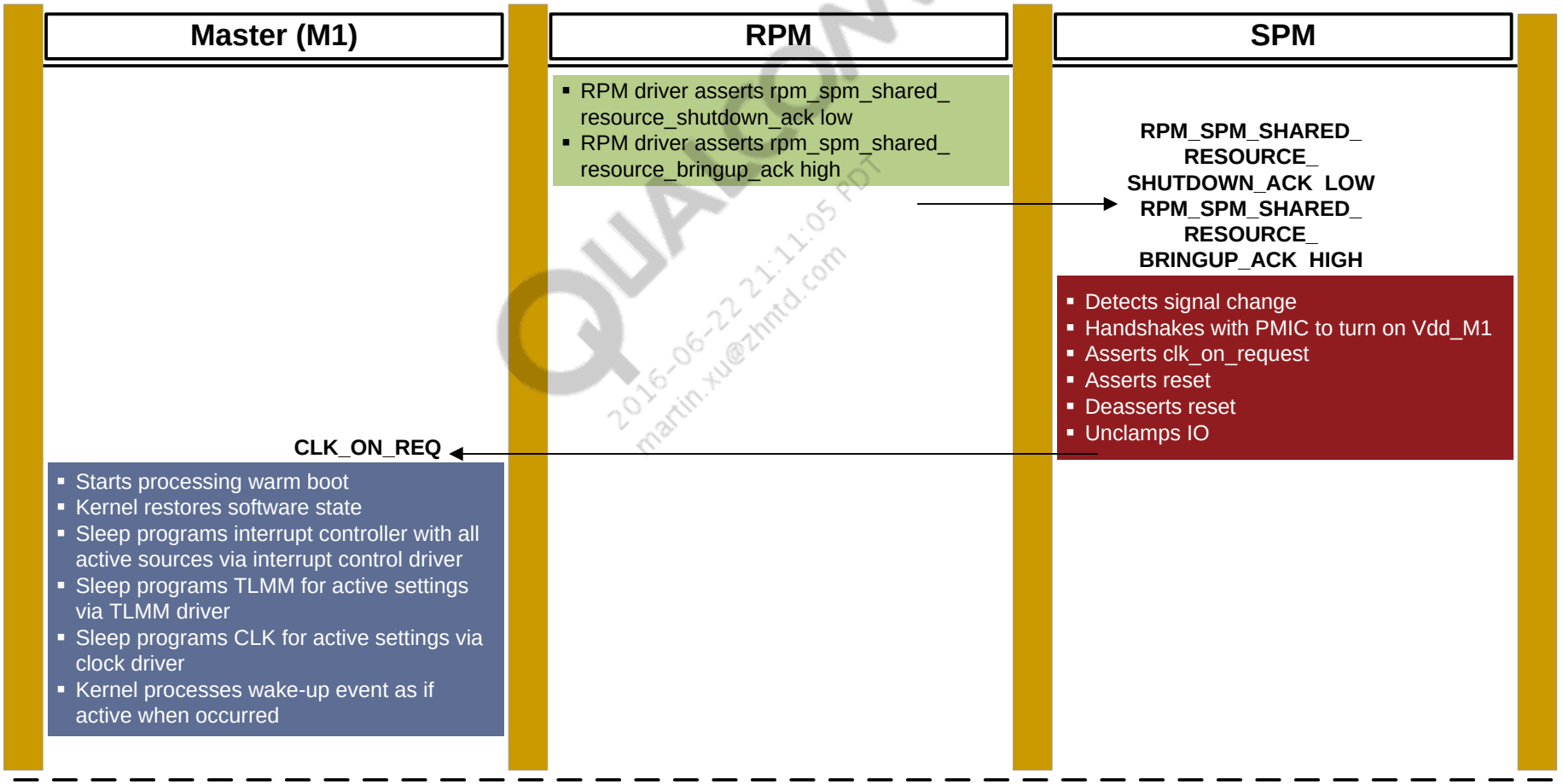
XO Restore/System VDD Restore Flow



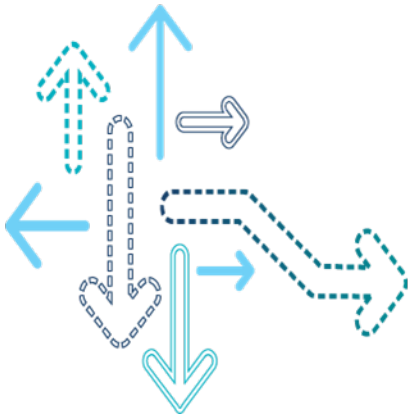
Subsystem Core Power Restore Flow



Subsystem Core Power Restore Flow (cont.)



Sleep Debug



Collecting the Sleep Log

- Examples are taken from the MSM8660 modem subsystem.
- The sleep logs use the ulog facility.
- To collect the sleep logs, run the ULogDump.cmm in a T32 debugger. Specify the output location as the command parameter.

```
do ..\..\core\power\ulog\scripts\ulogdump c:\temp
```

- From c:\temp, you should be able to find the following sleep log files:

```
Sleep Debug.ulog
Sleep Info.ulog
Sleep Profiling.ulog
Sleep Statistics.ulog
Sleep WarningError.ulog
```


Debug Log

- This shows the hard wake-up durations from the timer and sleep controller.

```
0x407A9E: Hard wakeup durations (timer duration: 0x000002D6) (target
duration: 0xFFFFFFFF)
```

- Debug logs the early wake-up when there is pending interrupt or sleep duration is set to 0.

```
0x40F9EF: Hard wakeup durations (timer duration: 0x00000000) (target
duration: 0xFFFFFFFF)
0x40F9F0: Early wakeup (min_duration: 0x00000000) (ints pending:
true)
```

- Debug log also records the short SWFI if there is not sufficient time for the sleep solver to run.

```
0x4084AE: Short SWFI (hard duration: 0x0000001E) (soft duration:
0xFFFFFFFF) (latency budget: 0xFFFFFFFF)
```

Info Log

- Sleep info log contains mainly information related to low power modes and sleep solver operation, e.g.:

```
0xF9AB62A3: Solver constraints (hard duration: 0x0000003C) (soft
duration: 0xFFFFFFFF) (latency budget: 0x00000021)
0xF9AB62A4: Mode skipped (name: "swfi.clk_gate_with_ramp") (reason:
"Insufficient Time") (enter latency: 0x0000001E) (exit latency:
0x0000000D) (latency budget: 0x00000021) (duration: 0x0000003C)
0xF9AB62A4: No mode chosen
```

- In this sequence, the sleep solver was entered with the following constraints:
  - Hard duration – 0x3C
  - Soft duration – 0xFFFFFFFF
  - Latency budget – 0x21
- We can see that there was only one enabled low power mode, swfi.clk_gate_with_ramp, which was skipped because of the insufficient time on latency budget.

Profiling Log

- The profiling log records the entry and exit points of sleep_perform_lpm, sleep solver, and sleep mode, e.g.:

```
0x40C128: Sleep entry
0x40C131: Solver Entry (name: "Legacy Solver")
0x40C132: Solver exit
0x40C132: Mode entering (lpr: "swfi") (lprm: "clk_gate_with_ramp")
0x40C22F: Mode exiting (lpr: "swfi") (lprm: "clk_gate_with_ramp")
0x40C238: Sleep exit
```


Statistics Log

- The statistics log records the count and total time that the modem spends in each low power mode, e.g.:

```
0x13FDD8: (Mode: vdd_dig.min_level_0) (Total time in mode:
0x001AEBB4 sclk) (Mode enter count: 1943)
0x13FDF5: (Mode: swfi.clk_gate_with_ramp) (Total time in mode:
0x00264132 sclk) (Mode enter count: 6149)
0x13FE1A: (Mode: mpm_warmup.slow_warmup) (Total time in mode:
0x001BBD3C sclk) (Mode enter count: 2080)
0x13FE30: (Mode: mpm_rampup.slow_rampup) (Total time in mode:
0x001AEBB4 sclk) (Mode enter count: 1943)
0x13FE40: (Mode: rpm_batch.send_handshake) (Total time in mode:
0x001BBD3C sclk) (Mode enter count: 2080)
0x13FE51: (Mode: rpm.handshake) (Total time in mode: 0x001BBD3C
sclk) (Mode enter count: 2080)
0x13FE60: (Mode: pxo.shutdown) (Total time in mode: 0x001BBD3C sclk)
(Mode enter count: 2080)
0x13FE71: (Mode: cxo.shutdown) (Total time in mode: 0x001BBD3C sclk)
(Mode enter count: 2080)
```

Sleep Request

- In recent builds, the requests on LPRs are filtered out to the sleep request log, e.g.:

```
0x4A237E: Mode disabled (lpr: "pxo") (lprm: "shutdown")
0x4A6443: Mode enabled (lpr: "pxo") (lprm: "shutdown")
0x4A6444: Synthesized component modes are not enabled(CLpr:
0x42F3BDDC) (Core: 0x00000001)
```

- Here, PXO's LPRM is disabled at 0x4A237E and enabled at 0x4A6443. However, the synthesized modes associated with PXO are not enabled, probably because cxo.shutdown mode is disabled.

Sleep Dump

- The SleepDump.cmm can be used to dump the state of sleep LPRs.

```
/core/power/sleep/scripts/SleepDump.cmm <path to write the logs>
```

- The output contains:
  - All LPRs' LPRMs and their enter/exit latency and power savings
  - List of synthesized LPRMs and their current states (enabled or not)
- An example snippet of sleep dump is:

```
0x0: Core 0
0x0:   Registered LPRs:
0x0:     (Resource Name: cpu_vdd0)  (Mode Count: 1)
0x0:       (Mode: cpu_vdd0.power_collapse)  (status: Enabled)  (enter latency: 1)
(exit latency: 1)  (backoff: 1)  (Attribute: 0x3)
0x0:     (Resource Name: swfi_core0)  (Mode Count: 3)
0x0:       (Mode: swfi_core0.clk_gate_with_ramp)  (status: Enabled)  (enter latency:
30)  (exit latency: 13)  (backoff: 13)  (Attribute: 0x3)
0x0:       (Mode: swfi_core0.swfi_power_collapse)  (status: Enabled)  (enter
latency: 96)  (exit latency: 160)  (backoff: 59)  (Attribute: 0x3)
0x0:       (Mode: swfi_core0.swfi_l2cache_power_collapse)  (status: Enabled)  (enter
latency: 110)  (exit latency: 180)  (backoff: 59)  (Attribute: 0x0)
0x0:   Component pre-synthesized modes:
0x0:    (Mode: cpu_vdd0.power_collapse + swfi_core0.swfi_power_collapse)
0x0:      (Status: Enabled)
0x0:    (Mode: swfi_core0.clk_gate_with_ramp)
0x0:      (Status: Enabled)
```

NPA Log

- NPA also logs into ulog. From the \build\ms directory, you can run the following command in a T32 debugger to save the NPA logs:

```
do ..\..\core\power\npa\scripts\NPADump.cmm c:\temp
```

- The NPA log has an npa_dump section that contains the current state of all NPA resources. Information on wake-up and latency nodes is:

```
: npa_resource (name: "/core/cpu/latency") (handle: 0x90103260) (units: sclk)
(resource max: 4294967295) (active max: 4294967295) (active state 1)  (active
headroom: 2) (request state: 1)
: npa_client (name: GPS_CPU_LATENCY_CLIENT) (handle: 0x9018B7F4) (resource:
0x90103260) (type: NPA_CLIENT_REQUIRED  0x40) (request: 0)
: npa_client (name: GSM_LATENCY) (handle: 0x90172DE4) (resource: 0x90103260) (type:
NPA_CLIENT_REQUIRED  0x40) (request: 0)
: npa_client (name: HKADC) (handle: 0x9013FF24) (resource: 0x90103260) (type:
NPA_CLIENT_REQUIRED  0x40) (request: 1)
: end npa_resource (handle: 0x90103260)
: npa_resource (name: "/core/cpu/wakeup") (handle: 0x901032AC) (units: sclk)
(resource max: 4294967295) (active max: 4294967295) (active state 0)  (active
headroom: 0) (request state: 0)
: end npa_resource (handle: 0x901032AC)
```

- We can see that HKADC requested for 1 tick latency.

Sleep Config File

- To disable low power modes via the file system on the modem, a sleep_config.ini file can be placed at this location (using EFS Explorer):

```
/nv/item_files/sleep/core0/sleep_config.ini
```

- The .ini file must be in this format:

```
[section]
disable=1
```

- In this example, “section” is the LPR or LPR.LPRM name to be configured.
- Section names must be in brackets, e.g., [vdd_dig].
- Any LPR or LPRM that is registered with the sleep subsystem may be configured in this way.
- If an LPR/LPRM is in the file more than once, the first occurrence will be honored.

Sleep Config Examples

- To disable all low power modes, your file would contain:

```
[all_lprms]
disable=1
```

- To disable all VDD DIG LPRMs, your file would contain:

```
[vdd_dig]
disable=1
```

- To disable a single VDD DIG LPRM, your file would contain:

```
[vdd_dig.min_level_0]
disable=1
```

- To disable RPM halt, your file would contain:

```
[rpm.halt]
disable=1
```

- To avoid timeout issues while debugging with JTAG, use the above procedure to disable RPM halt from the modem side.

References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1

Questions?

https://support.cdmatech.com