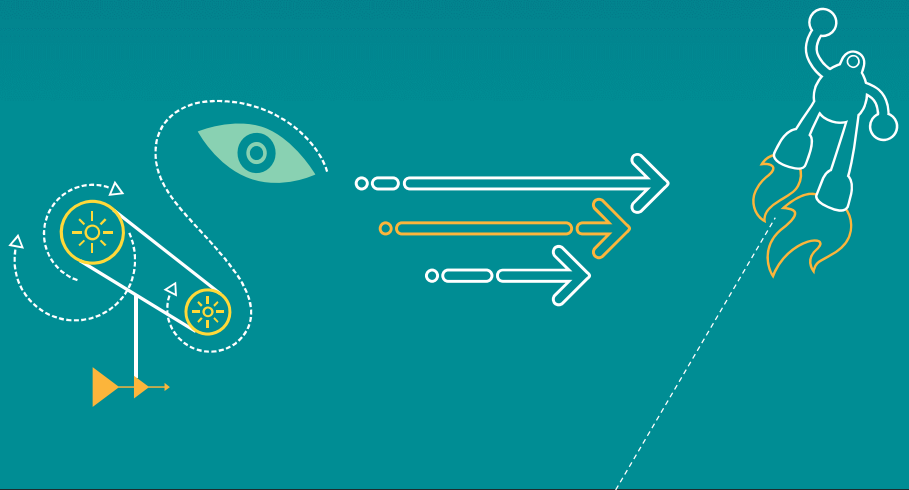
RPM Debug Overview

QUALCOMM®

Qualcomm Technologies, Inc.



80-NA157-9 E

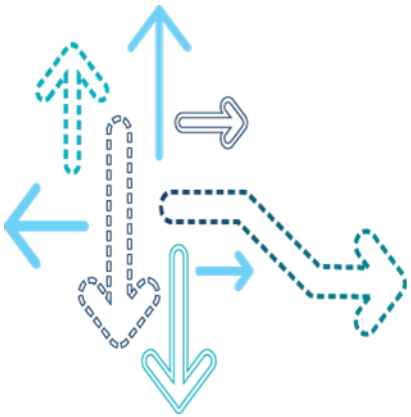
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	Jul 2012	Initial release
B	Sep 2012	Updated title to include MDM9x25, Execution Environment and RPM Memory Map slides; added [Q3] and [Q4]
C	Feb 2013	Added MSM8x26 to RPM Core and RPM Memory Map slides
D	May 2013	Updated to include MSM8x10, MPQ8092, and additional debug information
E	August 2015	Added APQ8074, APQ8084, MSM8x28, MSM8x12, MDM9x35, MSM8916, MSM8994, and railway section, and updated debug section

Contents

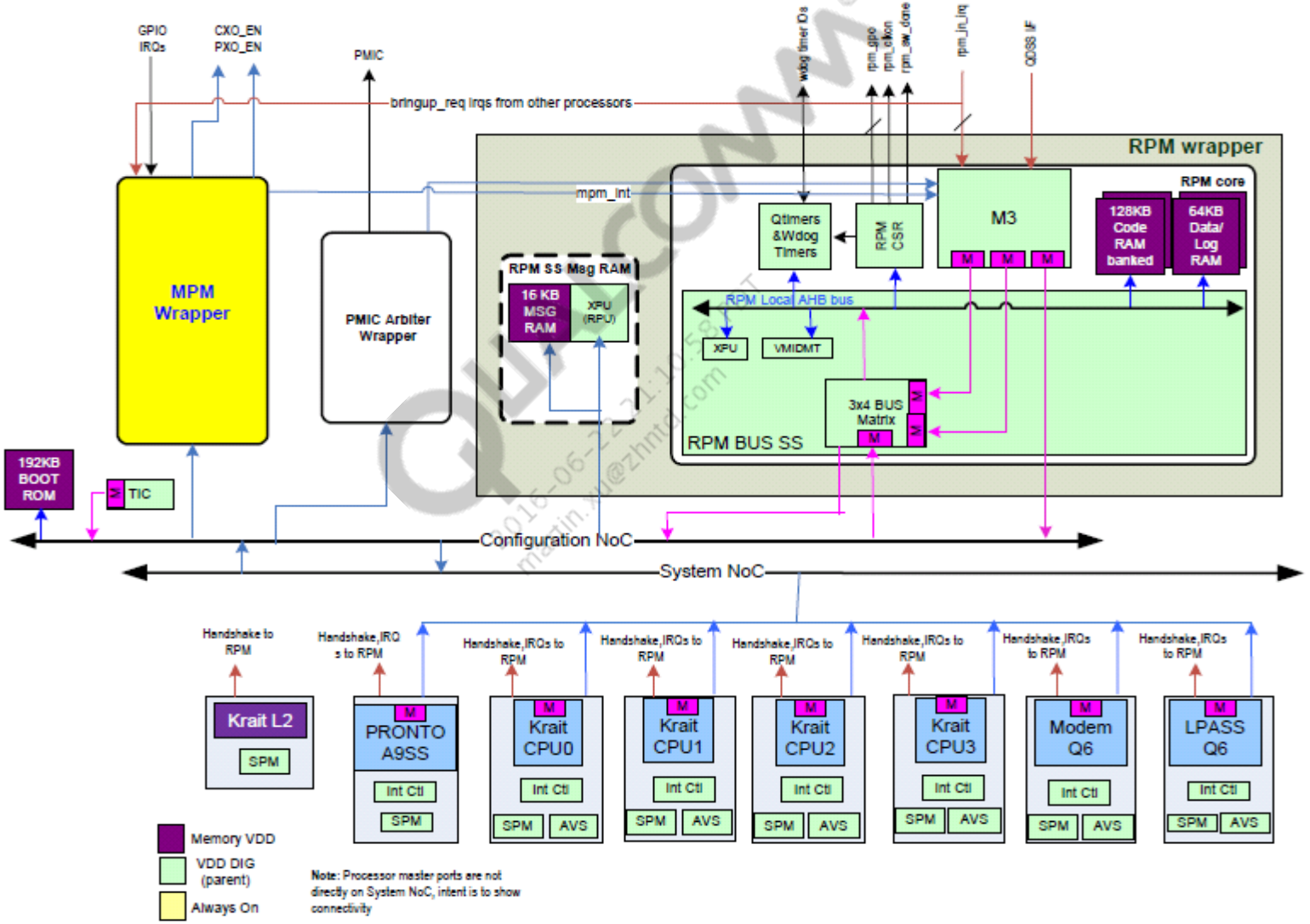
Overview



RPM Core

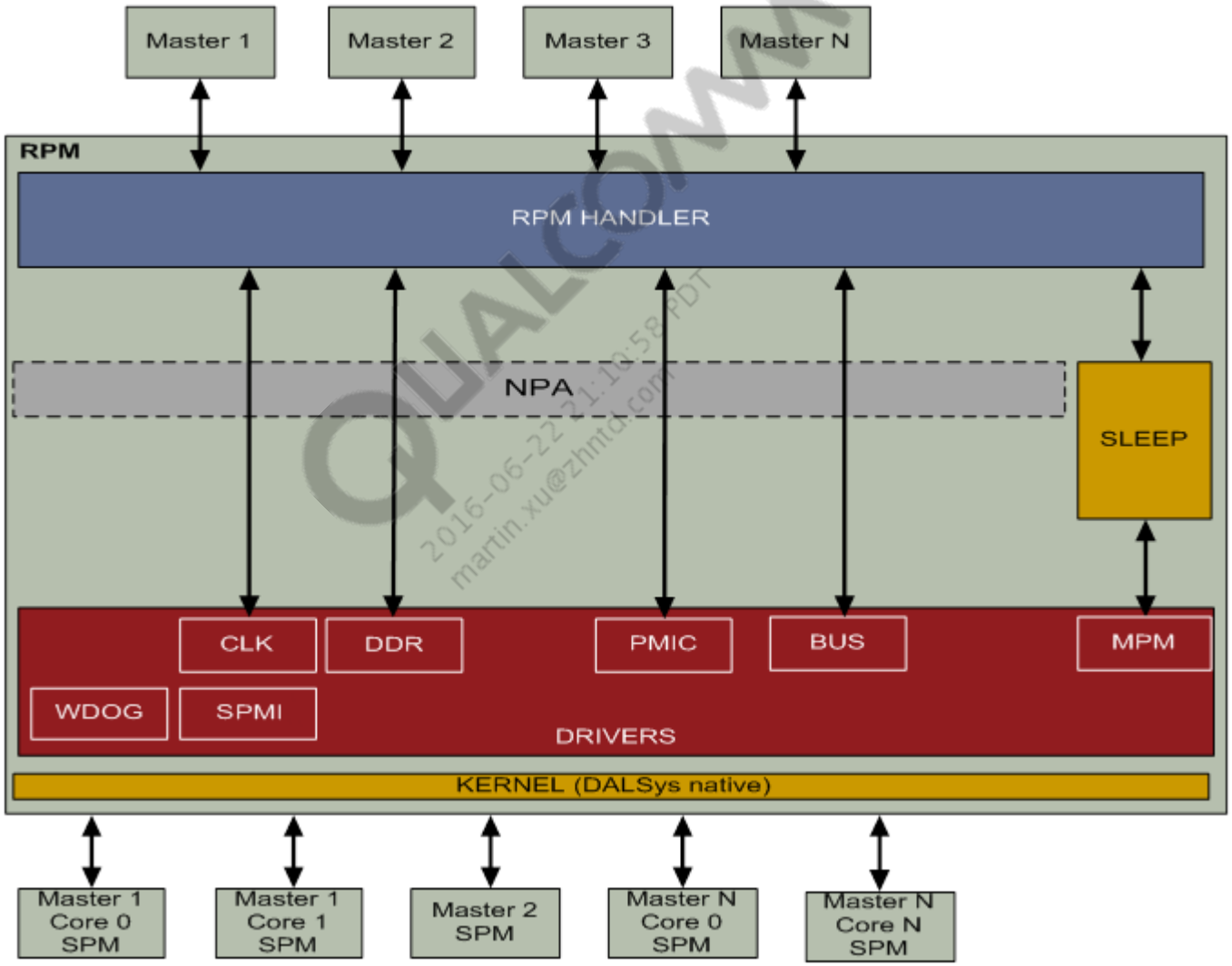
- Resource Power Manager (RPM)
  - Manages shared resources for the MSM
  - Manages chip Low Power modes
  - RPM.BF is the second generation RPM used by chipsets, including:
    - MSM8974 family – MSM8974, MSM8x26, MSM8x28, MSM8x10, MSM8x12, APQ8074, APQ8084, MSM8992, MSM8994, MDM9x25, MDM9x35, etc.
    - MSM8916 family – MSM8916, MSM8939, MSM8909, MSM8952, MSM8976, MDM9x45, MDM9x55, etc.
    - MSM8996 family – MSM8996, MSM8998, etc.
  - Uses an ARM® Cortex M3 processor; compared to ARM7 TDMI processor in RPM.AF:
    - Higher MIPS and low power
    - Harvard architecture with simultaneous instruction fetch with data load/store
    - Low interrupt latency
    - Thumb2, hardware multiply and divide

RPM Example Block Diagram



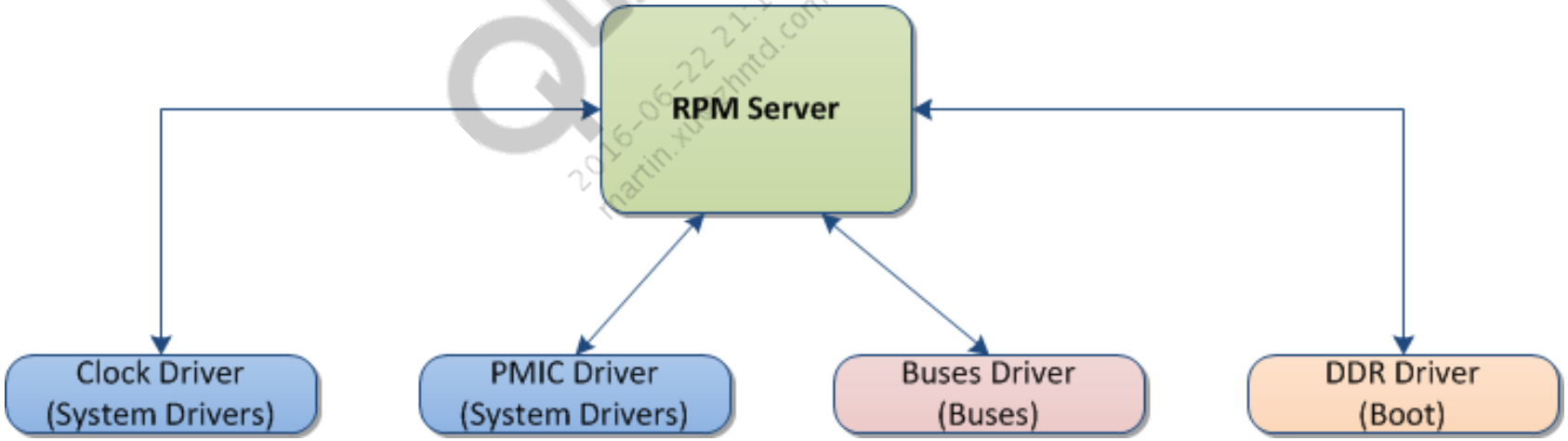
Note: No RPM BOOTROM in MSM8916 family and MSM8996 family, since it does not boot from the RPM core

High-Level Software Component Block Diagram – RPM



High-Level Software View

- RPM server – Intelligence in RPM
- Scheduler algorithms, estimation, prioritization, chip sleep programming
- Individual drivers for low-level hardware interaction



Power Features

- PMIC regulator management
  - All shared rails on the chip
- Clock management
  - All shared clocks/PLLs on the chip
- Bus fabric management
  - Bandwidth requests and interconnect paths
- Chip Sleep modes
  - Traditionally separate modes for XO shutdown, VDD minimization, and mock VDD minimization
  - Unified handling from MSM8994 onward; XO is gated in all modes
    - VDD minimization – CX and MX taken down to retention level
    - VDD low – CX and MX taken down to the lowest possible active level, based on votes
    - Mock VDD minimization – CX and MX remain at the existing level

Power Features (cont.)

- CPR
  - Open and closed-loop Core Power Reduction (CPR)
  - CX, MX, EBI rails
  - SSC_CX and SSC_MX open-loop support for SSC subsystem on MSM8996
- Scheduler
  - Schedules prioritized workload to ensure deadlines are met
  - Appropriate backoffs are applied
  - Scheduled and immediate tasks
  - Additional support of periodic task is implemented on MSM8996 for DDR temperature polling
- Set management
  - Active, Sleep, and Next Active sets
  - Opportunistic short circuit paths

System Features

- DDR controller software
  - Programming, periodic tasks, e.g., Z/Q cal, rail voting, software triggered self refresh
- ACC settings
  - Set timing for on-chip memories, i.e., code RAM, data RAM, IMEM, etc.
- Rail dependency management
  - CX, MX, EBI
  - Interaction with CPR, limits-related management, MX >= CX, MX – CX <= 300 uV, etc.
  - MSM8996 onward, constraint that MX >= CX does not exist
- Hardware workarounds
  - Place for hardware workarounds
  - Clocks, PMIC, voltage droops, etc.

System Features (cont.)

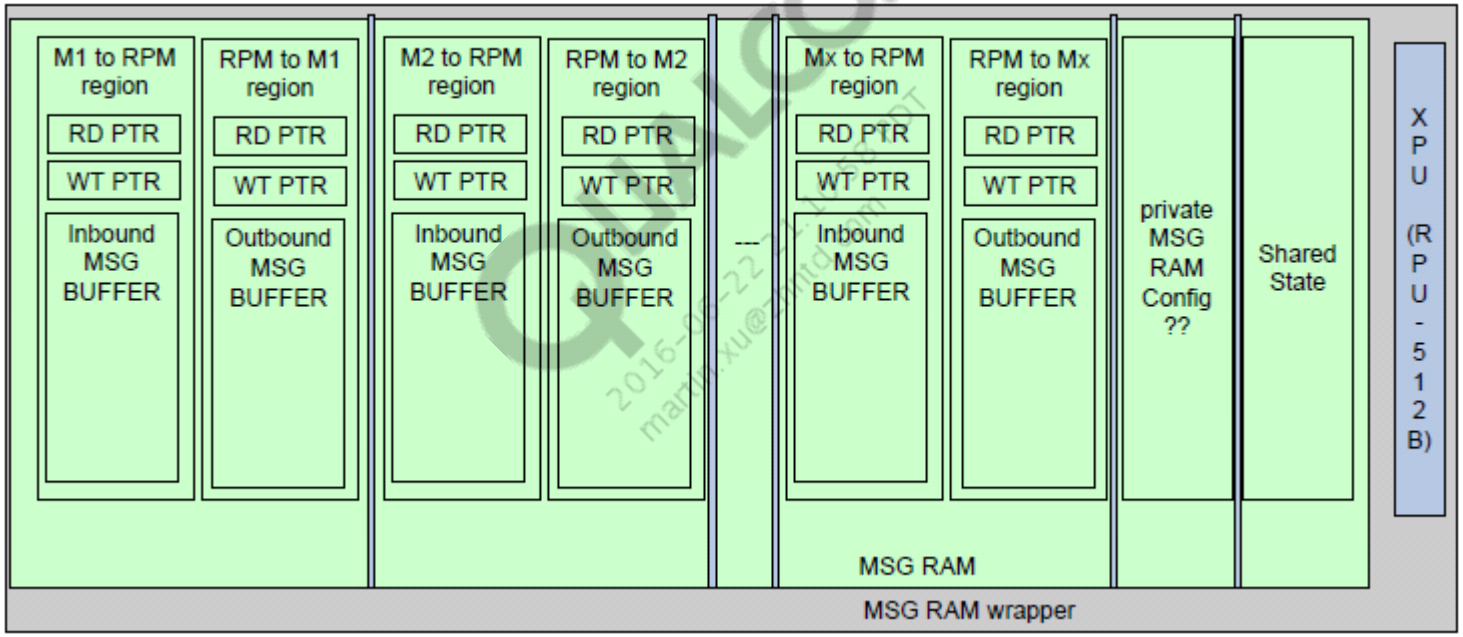
- Interrupts
  - NVIC programming, handling ISRs, priorities, etc.
- Timers
  - RPM Qtimer
- RPM watchdog

Debug Features

- Event/log captures
  - Dedicated data structures
- NPA and Ulogs
  - State of resources, software execution history
- QDSS
  - Masking/unmasking events

Message RAM Partition

- Memory partitioned and protected for each processor/execution environment



RPM Memory Map – MSM8974 Family

	MSM8960	MSM8974 family	MSM8992/MSM8994
RPM_CODE_START_ADDR	0x20000	▪ RPM view – 0x100000 ▪ SYS view – 0xFC100000	▪ RPM view – 0x100000 ▪ SYS view – 0xFC100000
RPM_CODE_SIZE	0x24000	0x20000	0x28000
RPM_DATA_START_ADDR	NA	▪ RPM view – 0x190000 ▪ SYS view – 0xFC190000	▪ RPM view – 0x190000 ▪ SYS view – 0xFC190000
RPM_DATA_SIZE	NA	0x10000	0x10000
RPM_MSG_RAM_ADDR	0x108000	0xFC428000	0xFC428000
RPM_MSG_RAM_SIZE	0x6000	0x4000	0x4000
RPM_SWVERSION_ADDR	0x108008	STR at 0x190040	STR at 0x190040

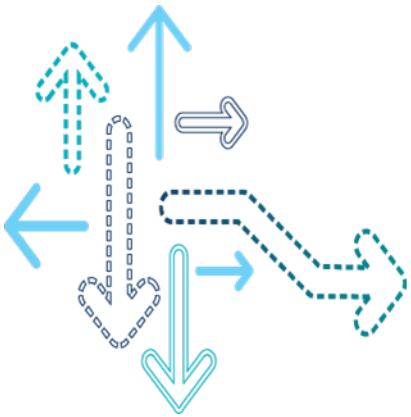
RPM Memory Map – MSM8916 Family

	MSM8916 family MSM8916/MSM8939/ MSM8909	MSM8952/MSM8976	MDM9x45/MDM9x55
RPM_CODE_START_ ADDR	▪ RPM view – 0x0 ▪ SYS view – 0x200000	▪ RPM view – 0x0 ▪ SYS view – 0x200000	▪ RPM view – 0x0 ▪ SYS view – 0x200000
RPM_CODE_SIZE	0x20000	0x24000	0x24000
RPM_DATA_START_ ADDR	▪ RPM view – 0x90000 ▪ SYS view – 0x290000	▪ RPM view – 0x90000 ▪ SYS view – 0x290000	▪ RPM view – 0x90000 ▪ SYS view – 0x290000
RPM_DATA_SIZE	0x10000	0x10000	0x10000
RPM_MSG_RAM_ADDR	▪ RPM view – 0x60060000 ▪ SYS view – 0x60000	▪ RPM view – 0x60060000 ▪ SYS view – 0x60000	▪ RPM view – 0x60060000 ▪ SYS view – 0x60000
RPM_MSG_RAM_SIZE	0x4000	0x5000	0x6000
RPM_SWVERSION_ ADDR	STR at 0x90040	STR at 0x90040	STR at 0x90040

RPM Memory Map – MSM8996 Family

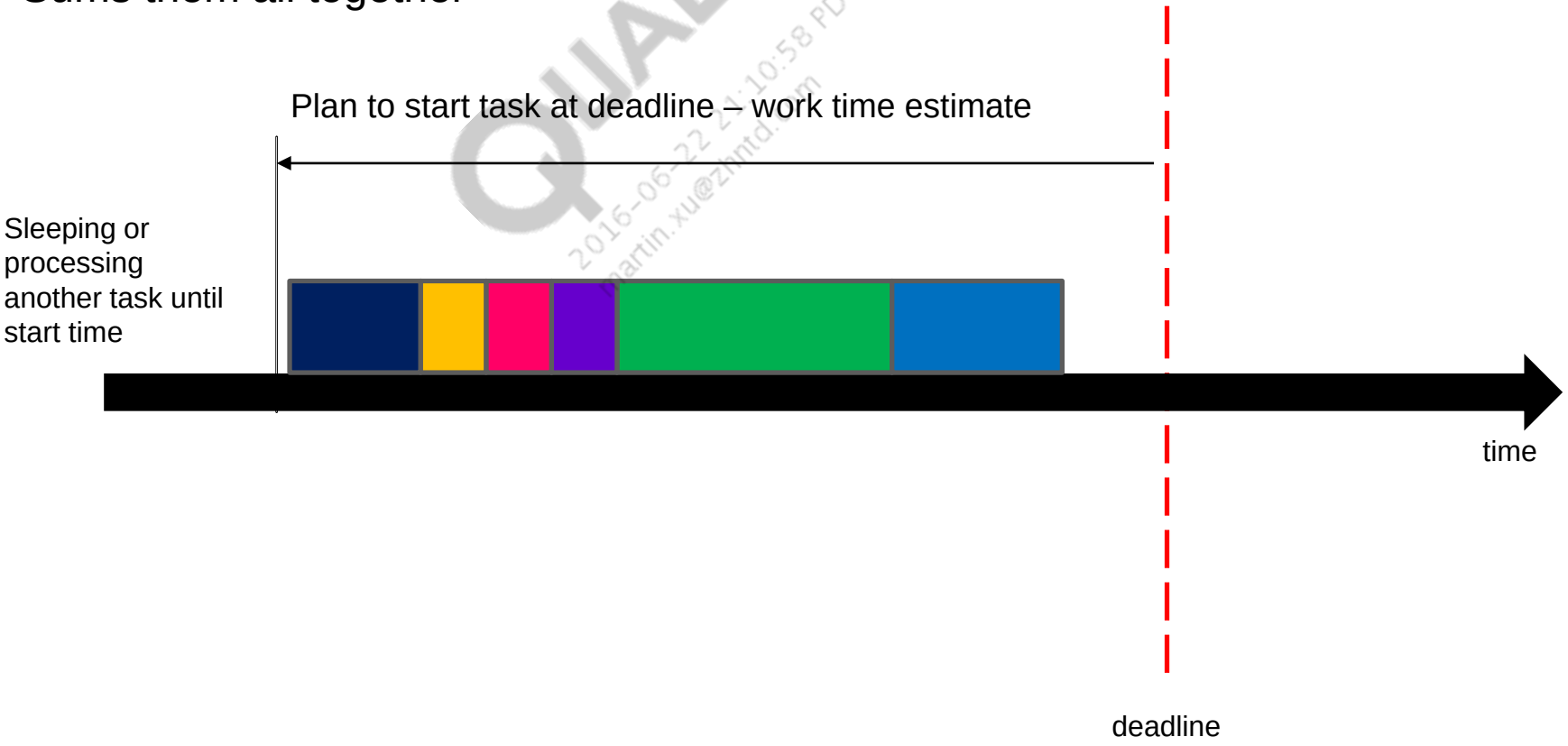
	MSM8996	MSM8998
RPM_CODE_START_ADDR	▪ RPM view – 0x0 ▪ SYS view – 0x200000	▪ RPM view – 0x0 ▪ SYS view – 0x200000
RPM_CODE_SIZE	0x28000	0x28000
RPM_DATA_START_ADDR	▪ RPM view – 0x90000 ▪ SYS view – 0x290000	▪ RPM view – 0x90000 ▪ SYS view – 0x290000
RPM_DATA_SIZE	0x14000	0x14000
RPM_MSG_RAM_ADDR	▪ RPM view – 0x60068000 ▪ SYS view – 0x68000	▪ RPM view – 0x60778000 ▪ SYS view – 0x778000
RPM_MSG_RAM_SIZE	0x6000	0x7000
RPM_SWVERSION_ADDR	STR at 0x90040	STR at 0x90040

Scheduling

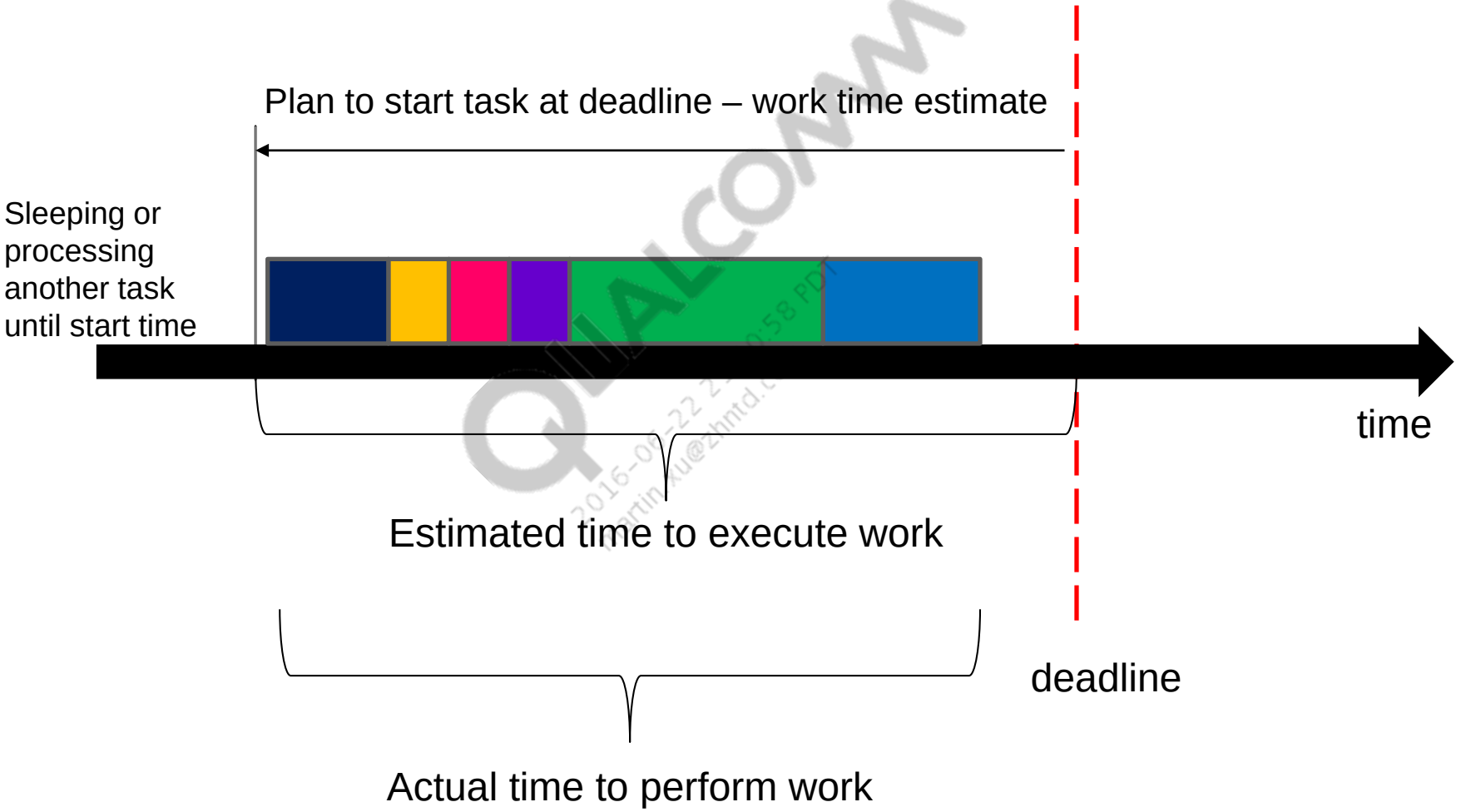


Scheduled Sleep Wakeup

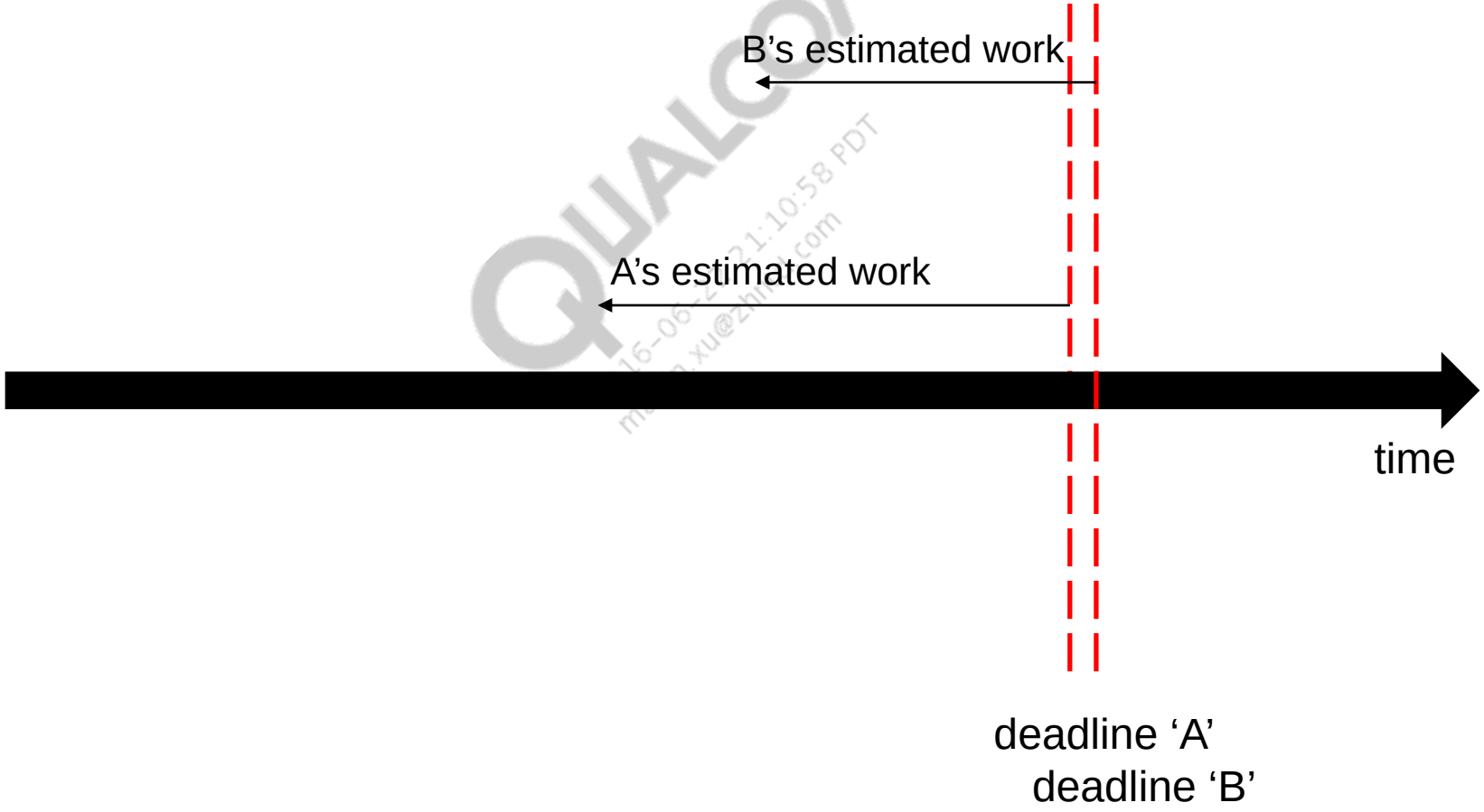
- RPM is single-thread time-based scheduler
- Current estimation algorithm
  - Determines which resources need to be updated
  - Looks up worst-case transition time for that resource based on historical data
  - Sums them all together



Scheduled Sleep Wakeup (cont.)

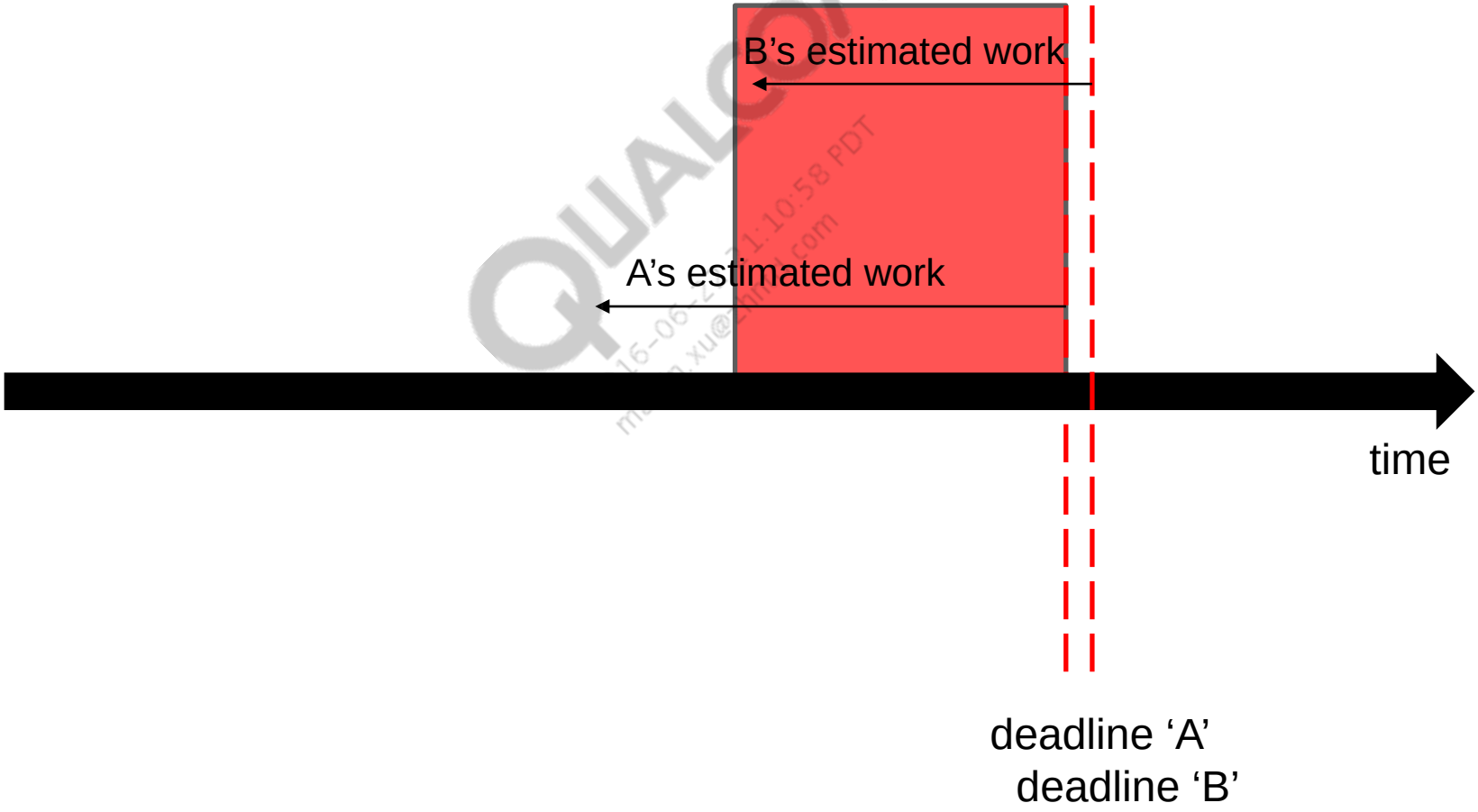


Schedule Collision – Two Scheduled



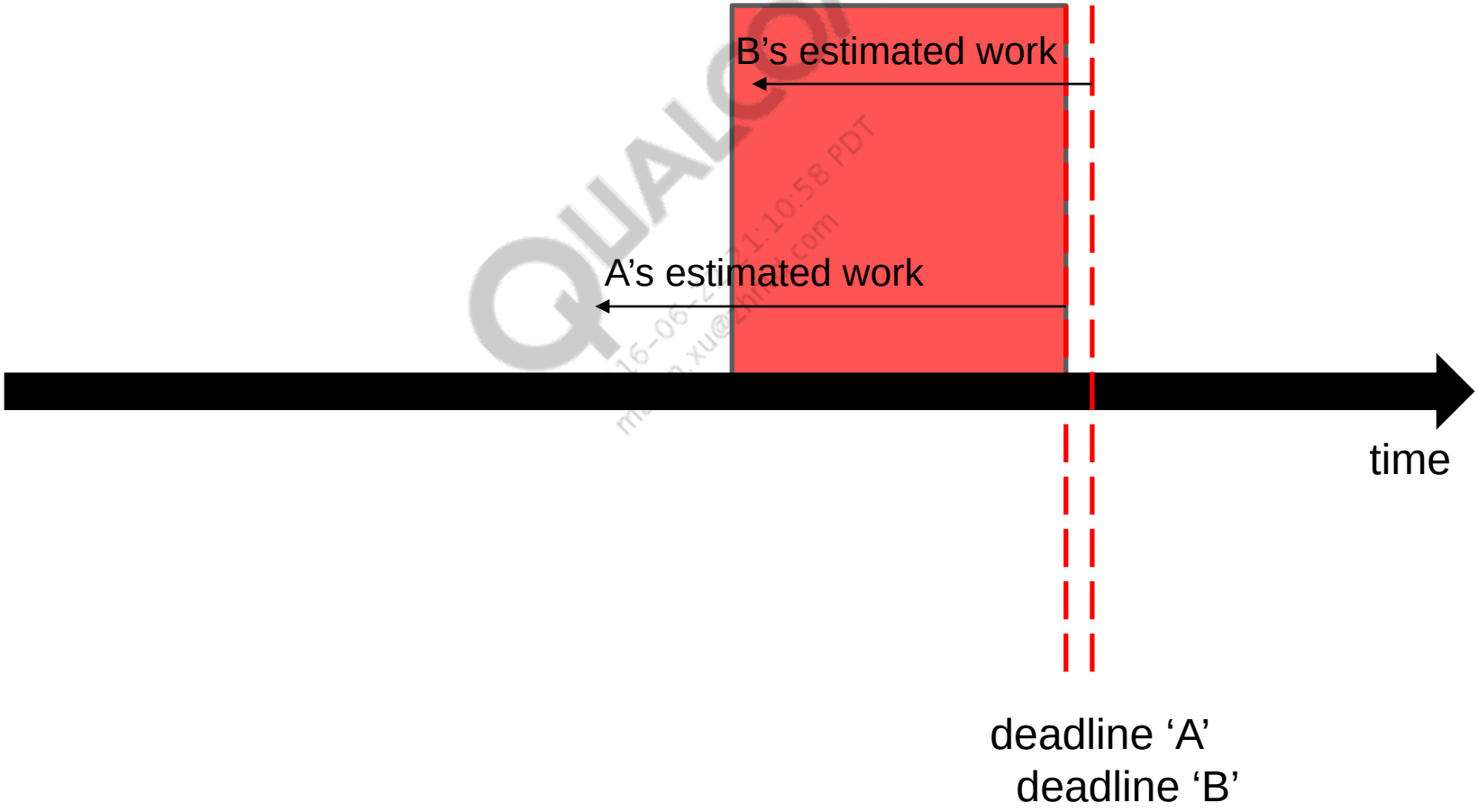
Schedule Collision – Two Scheduled (cont.)

- Conflict – Need to work on two things at once

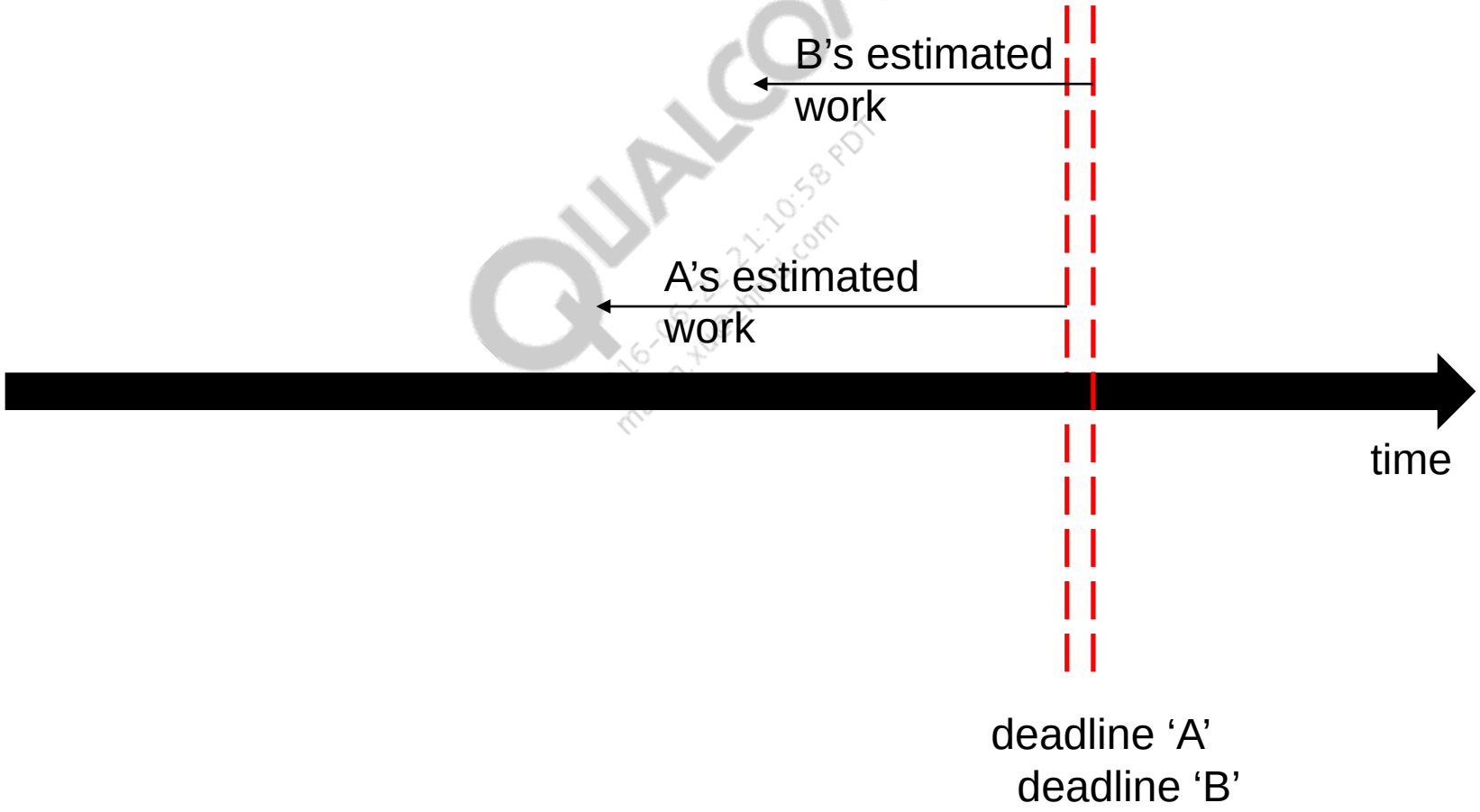


Schedule Collision – Two Scheduled (cont.)

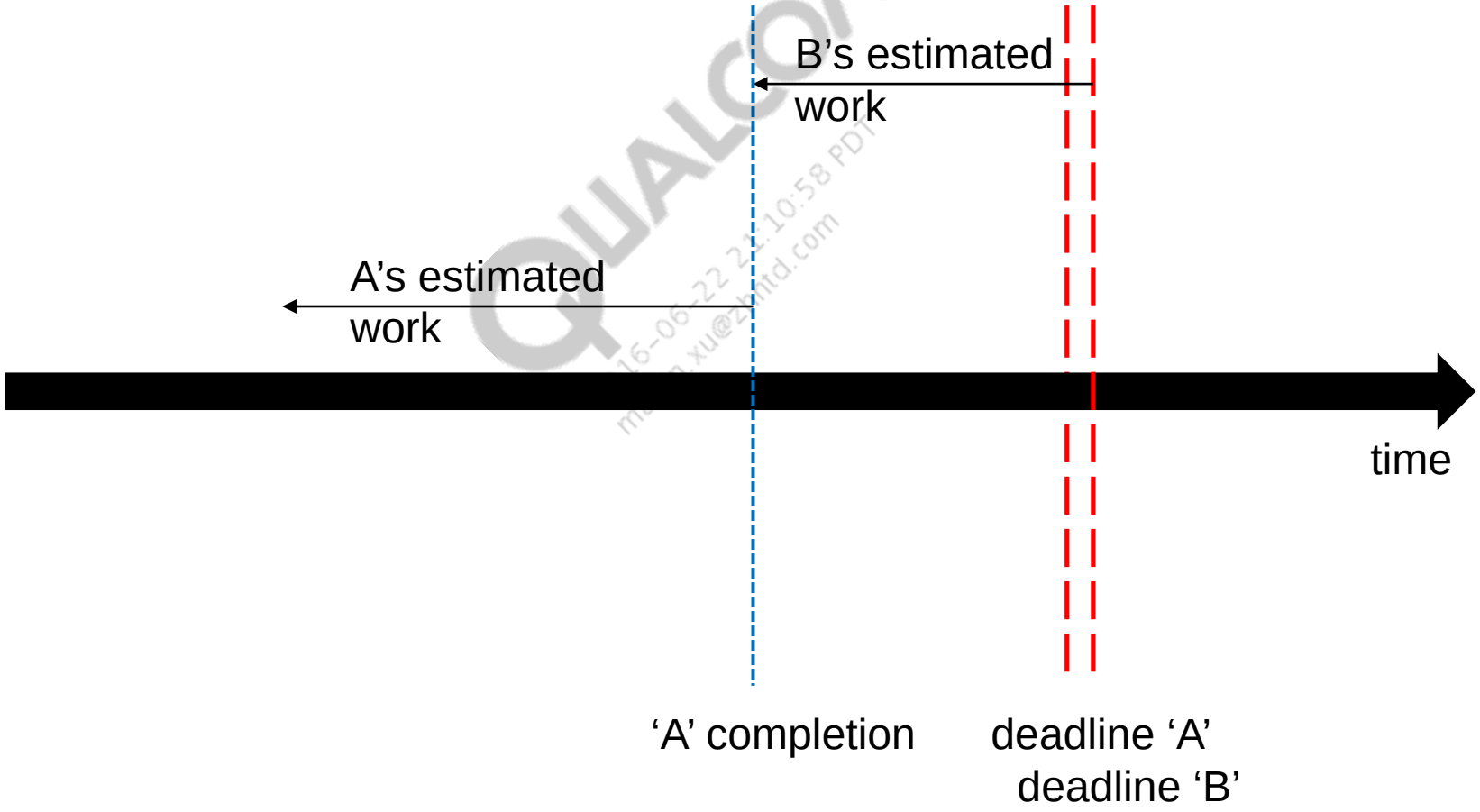
- Resolution?
- Inspired by OS scheduling approaches known as “earliest deadline first”



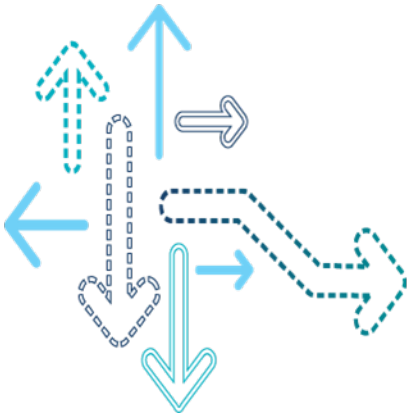
Schedule Collision – Two Scheduled (cont.)



Schedule Collision – Two Scheduled (cont.)



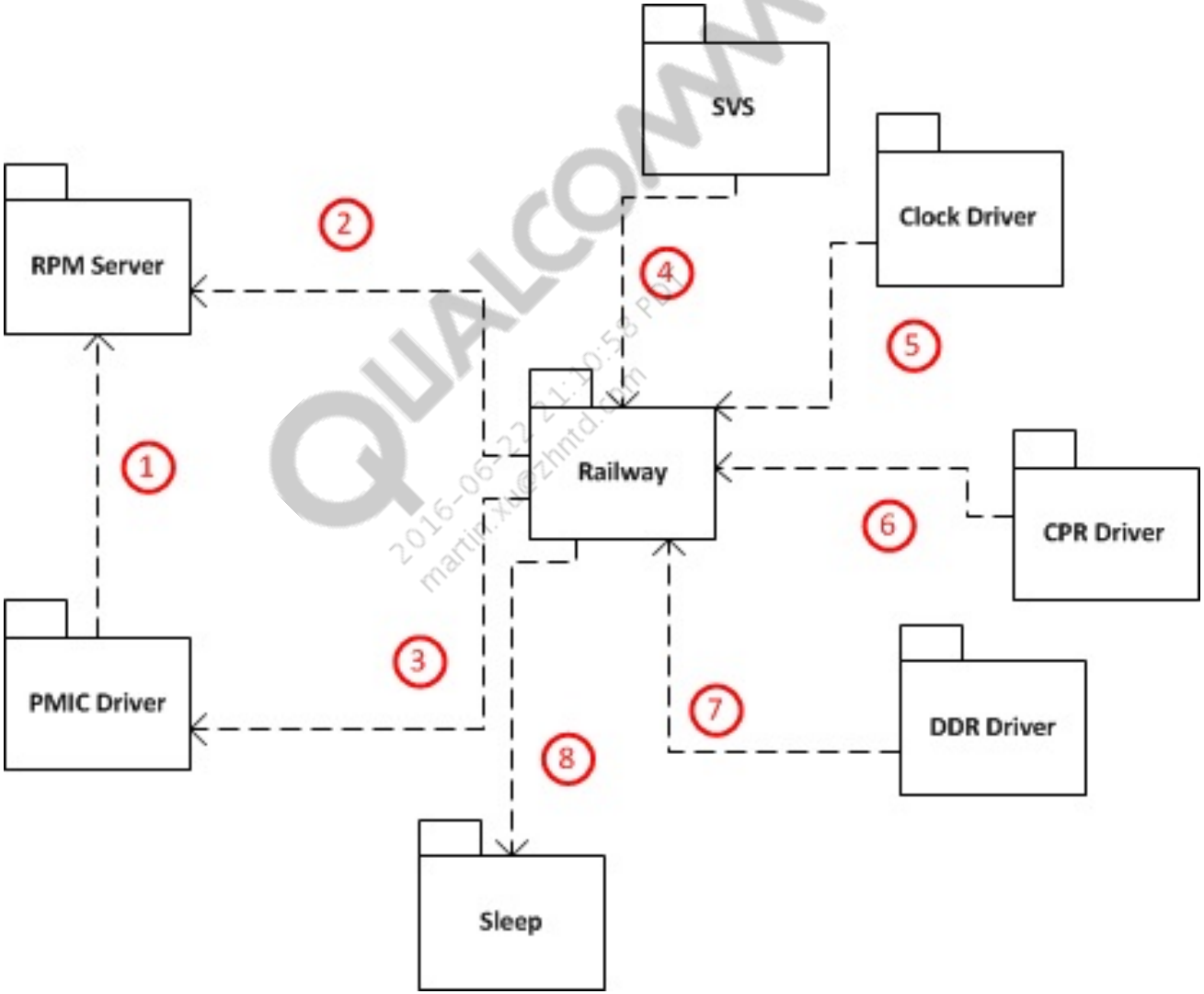
Railway Driver



Railway Overview

- RPM railway driver
  - Manages shared railways – VDDCX and VDDMX; VDDGFX and VDDEABI if they exist
  - Aggregates rail voltage votes
  - Notifies registered clients when a rail changes voltage
  - Notifies the sleep module if vdd_min is prohibited at the time
  - Makes voltage and sw_mode requests to PMIC driver

Railway Component Interface and Dependency



Tune Railway Voltage

- Code changes

```
rpm_proc\core\power\railway_v2\src\<target>\railway_config.c
static const railway_config_data_t temp_config_data = …
    .default_uvs = (const unsigned[])
    {
        0,          //RAILWAY_NO_REQUEST
        675000,     //RAILWAY_RETENTION
        950000,     //RAILWAY_SVS_KRAIT
        950000,     //RAILWAY_SVS_SOC
        950000,     //RAILWAY_SVS_HIGH
        950000,     //RAILWAY_NOMINAL
        1050000,    //RAILWAY_TURBO
        1050000,    //RAILWAY_TURBO_HIGH
        1050000,    //RAILWAY_SUPER_TURBO
        1050000,    //RAILWAY_SUPER_TURBO_NO_CPR
    },
```

- Consult PMIC and hardware engineers for voltage recommendations
- Entries in default_uvs are in strict ascending order
- Ensure VDDMX >= VDDCX and VDDMX >= VDDGFX

Decode Railway Voting

- Follow railway.rail_state[n].voter_list_head to walk through voters

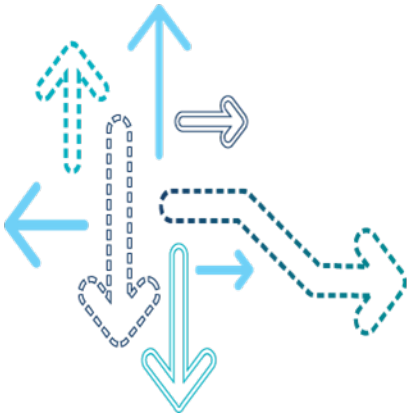
```
railway.rail_state[1] = (
(current_active = (mode = RAILWAY_SUPER_TURBO, microvolts = 1050000, off_corner = FALSE),
 ...
  voter_list_head = 0x0019C9E8 -> (
     ...
        voter_link = 0x0019A840 -> (
          voltage_corner = RAILWAY_TURBO,
          explicit_voltage_in_uv = 1045000,        <-- Set VDDCX explicitly
          suppressible = FALSE,                    <-- Required voter
          id = 0,                                  <-- ID 0 means it is APPS
          rail = 1,
          sw_enable = TRUE ,
```


  - If id < 100, it is EE# (master ID): 0(APPS)/1(MODEM), etc.
  - If id >= 100, see definition of railway_voter_id

Railway Target Specific Changes

- In MSM8996 onward, railway voltage limits, e.g., VDDMX >= VDDCX, are removed; each railway scales independently
- In MSM8909 merged Railway mode, VDDCX and VDDAPC are merged and controlled by RPM
- In MSM8996, VDDGFX is controlled by APPS
- MSM8996 supports vdd_ssc_mx and vdd_ssc_cx

RPM Inter-Subsystem Messaging



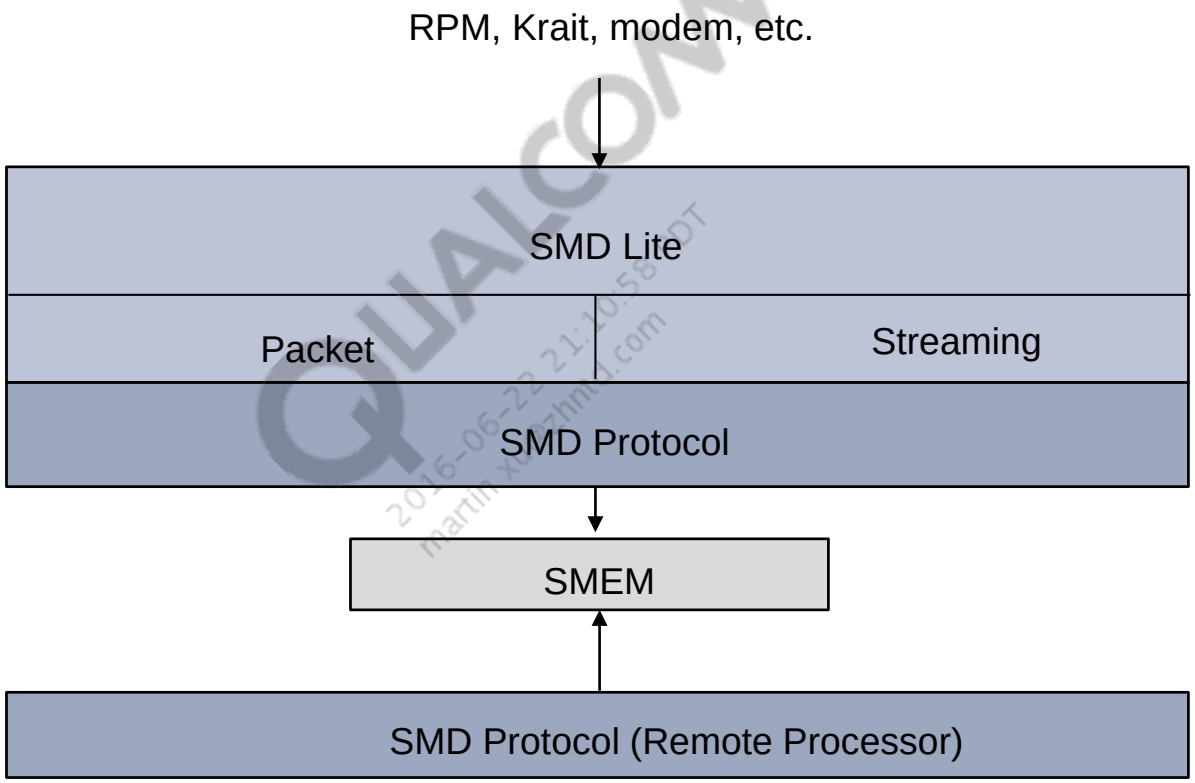
Transport – Register Emulation vs SMD Lite

- Limitation of register emulation
  - Model creates unnecessary inter-target work
  - Performance is undesirable
  - Resource usage is nonoptimal
- Shared Memory Driver (SMD) Lite is a message-oriented FIFO transport
  - Standard, well-understood Qualcomm Technologies, Inc. (QTI) technology
  - Allows symmetric full-duplex communication

SMDL

- SMD Lite (SMDL) is a lightweight implementation of SMD
  - Task-less, giving clients the control to read, write, and process data in their own context (priority)
  - Provides simple and intuitive APIs
    - smdl_read, smdl_write, etc.
    - Client pushes a nonblocking read or write, returns right away vs SMD pull buffer from clients
  - Lets clients manage their own buffers directly
  - Fully interoperable with SMD APIs
    - An SMD connection can have SMD on one end and SMDL on the other
- SMD is only a transport, a protocol must still be layered on top

SMD APIs and Layers



Message Structure – KVP

- SMD Lite provides messages and formats them as KVP
- KVP is 2+n words of data
  - One word for key – Generally used as a 4-byte string (uv\0\0, clk\0, etc.)
  - One word for length – Describes how many bytes follow
  - Blob of data
- Structuring is repeated recursively to build full messages

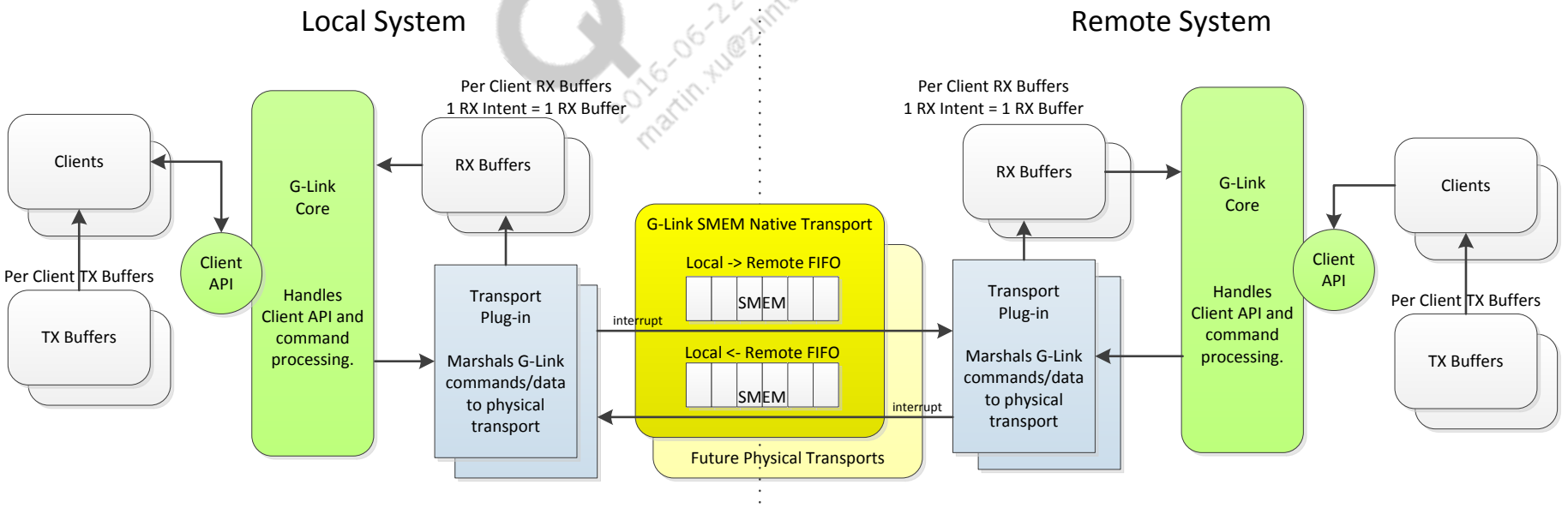
Message Structure – KVP (cont.)

- Example of a request to change LDO3 voltage and current

```
{
    “req\0” : {
        {“rsrc” : “ldo\0”}
        {“id”   : 3}
        {“set”  : 0}
        {“data” : {
              {“uv\0\0” : 1100000}
              {“mA\0\0” : 130}
          }
        }
    }
}
```

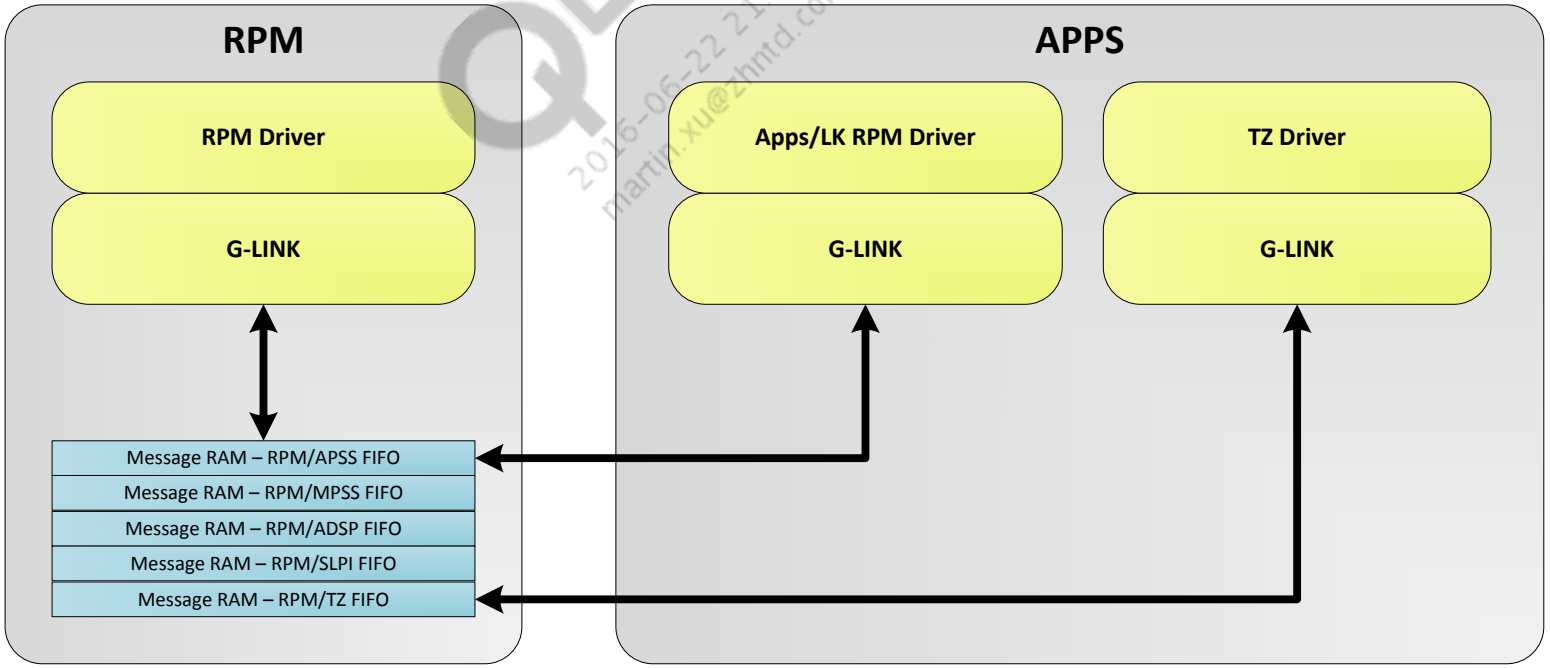

G-Link – Replacement of SMD

- New point-to-point link-layer transport
- Multiplexes logical channels over one or more physical transports
- Works with shared memory (replacing SMD over SMEM), copy-based (UART, etc.), and DMA-based physical transports without affecting client API
- Transport handler plug-in can be developed independently of existing transports
- Initial roll out is on MSM8996 (v2) and all SMD clients will be migrated to G-Link



G-Link – RPM System Overview

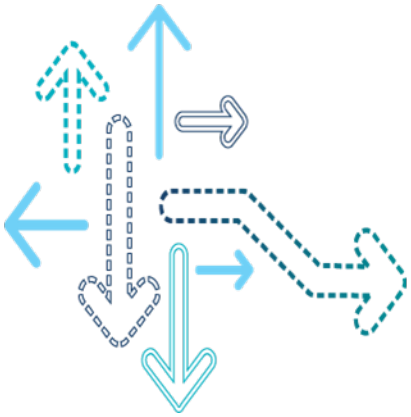
- G-LINK implementation for RPM uses message RAM
- RPM transport only supports intent-less mode, where:
  - Client reads directly from FIFO (no copy into client buffer) to reduce memory usage
  - No in-band control messages for performance
- RPM does not support channel migration, due to memory restrictions



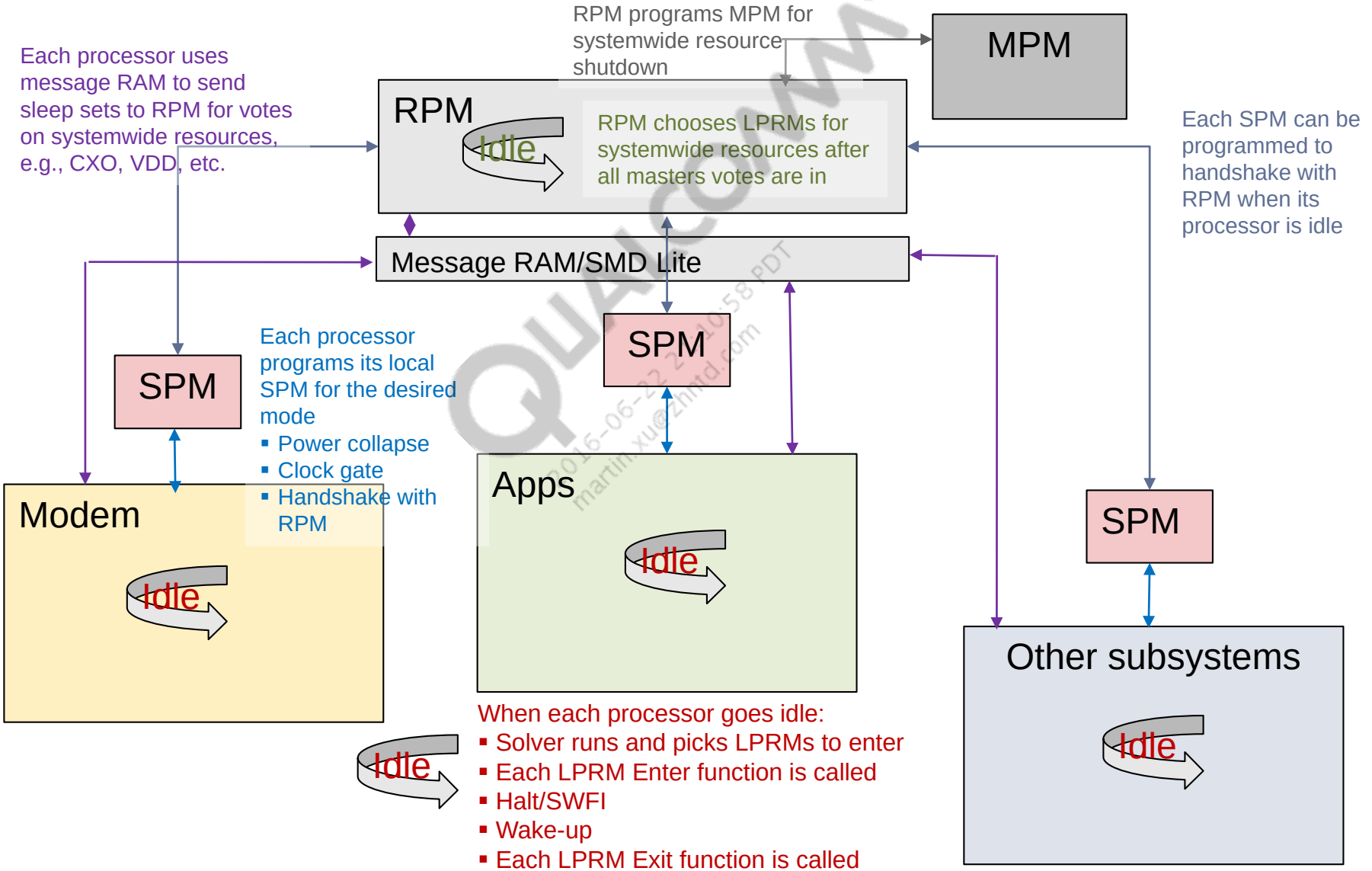
Debugging G-Link

- G-LINK transports and channels are obtained by loading corresponding ELF file and running glinklist.cmm T32 script under //source/qcom/qct/core/mproc/glink/main/latest/tools/cmm/glinklist.cmm

System Sleep



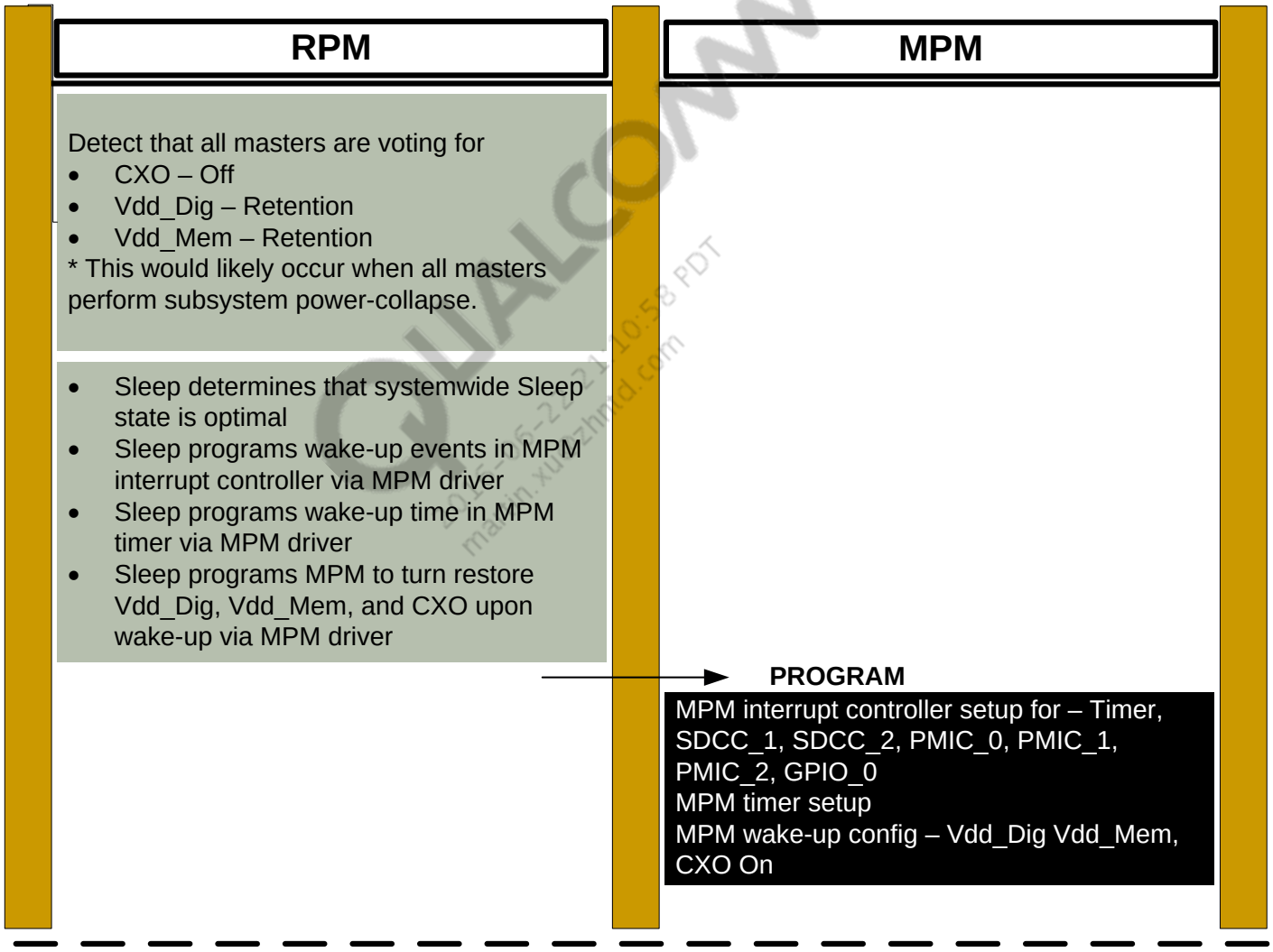
System Sleep Overview



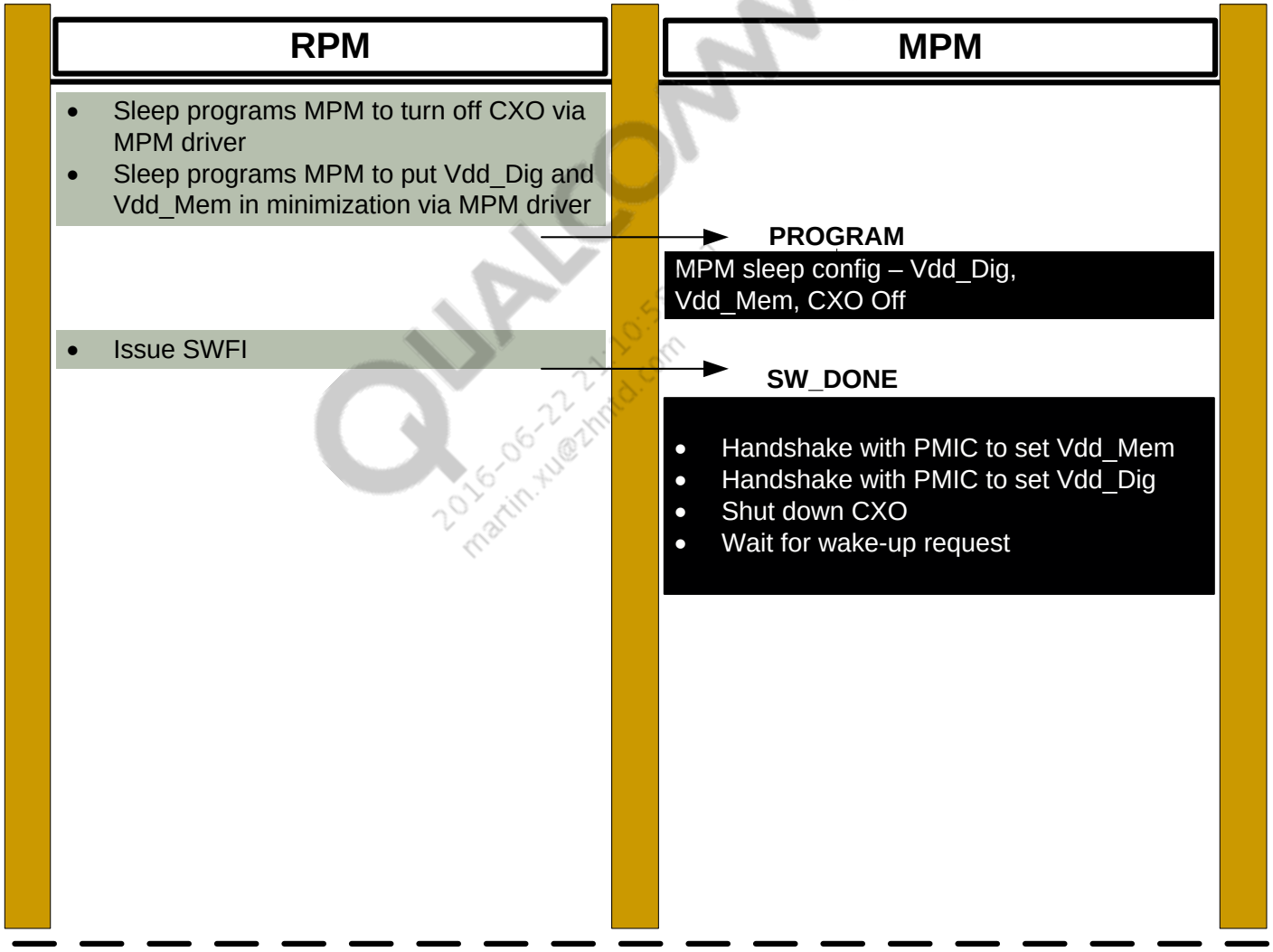
System Sleep Overview (cont.)

- XO shutdown – First form of system sleep gates off (shuts down) reference clocks in the system
- VDD minimization – Second form of system sleep puts system power rails in a retention state; this form can be used concurrently with the first form
- Since MSM8994, a Unified Sleep Model is implemented; XO shutdown is replaced with its functional equivalent, VDD-LOW
  - VDD-MIN – CX and MX taken down to retention level
  - VDD-LOW – CX and MX taken down to lowest possible active level, based on votes
  - Mock VDD-MIN – CX and MX remain at existing level

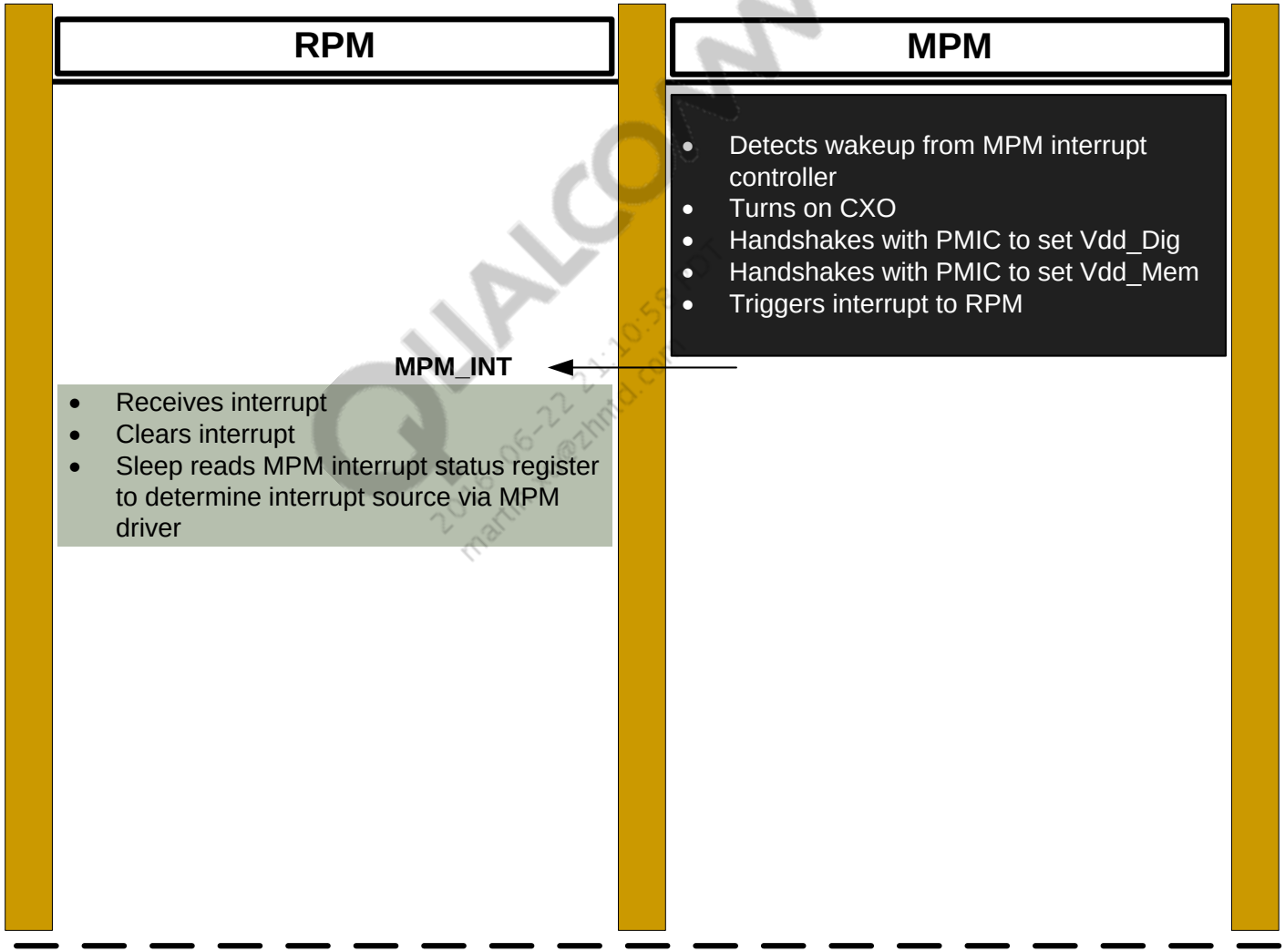
XO Shutdown/System Vdd_Min Flow



XO Shutdown/System Vdd_Min Flow (cont.)

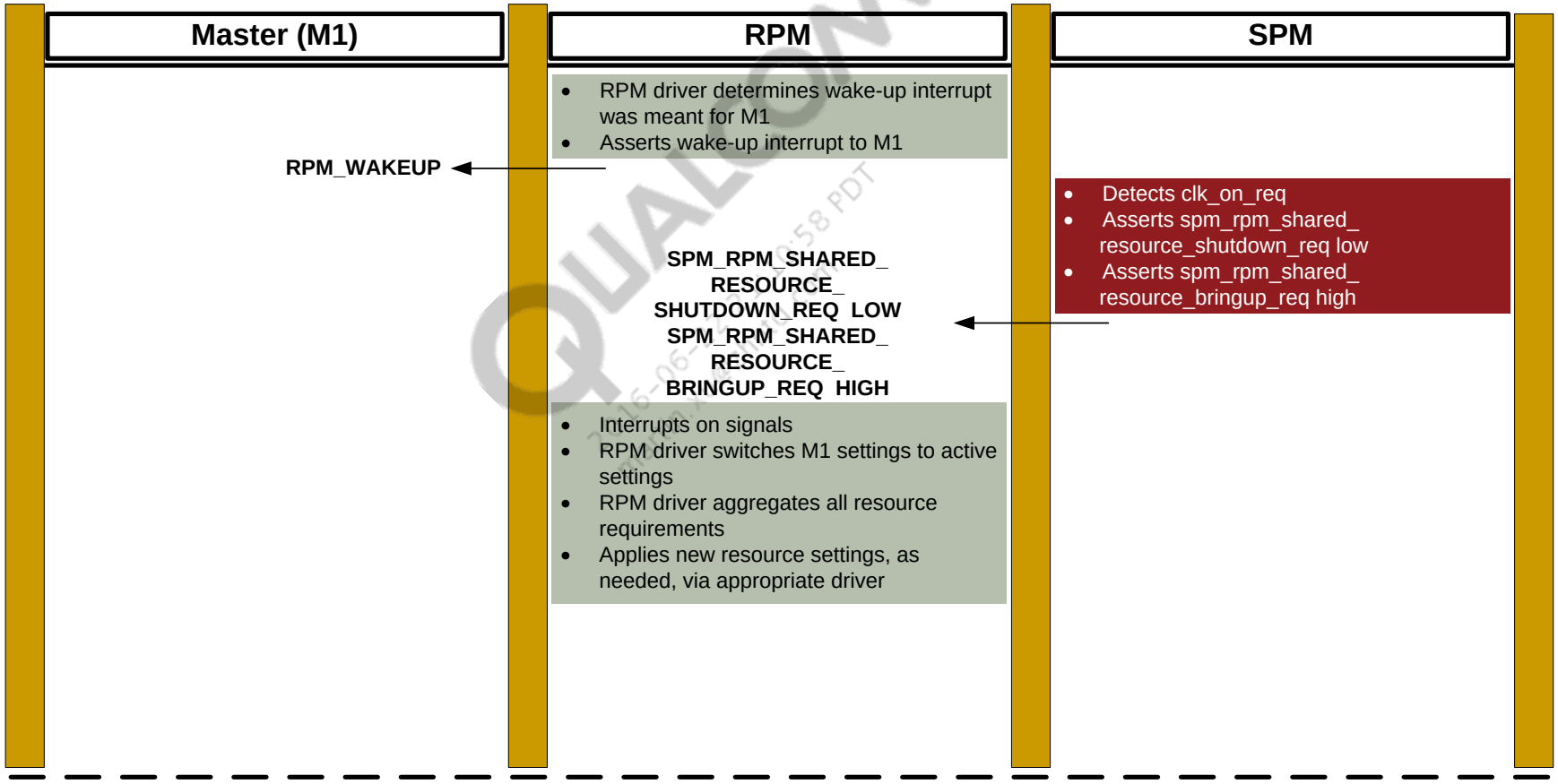


XO Restore/System VDD Restore Flow

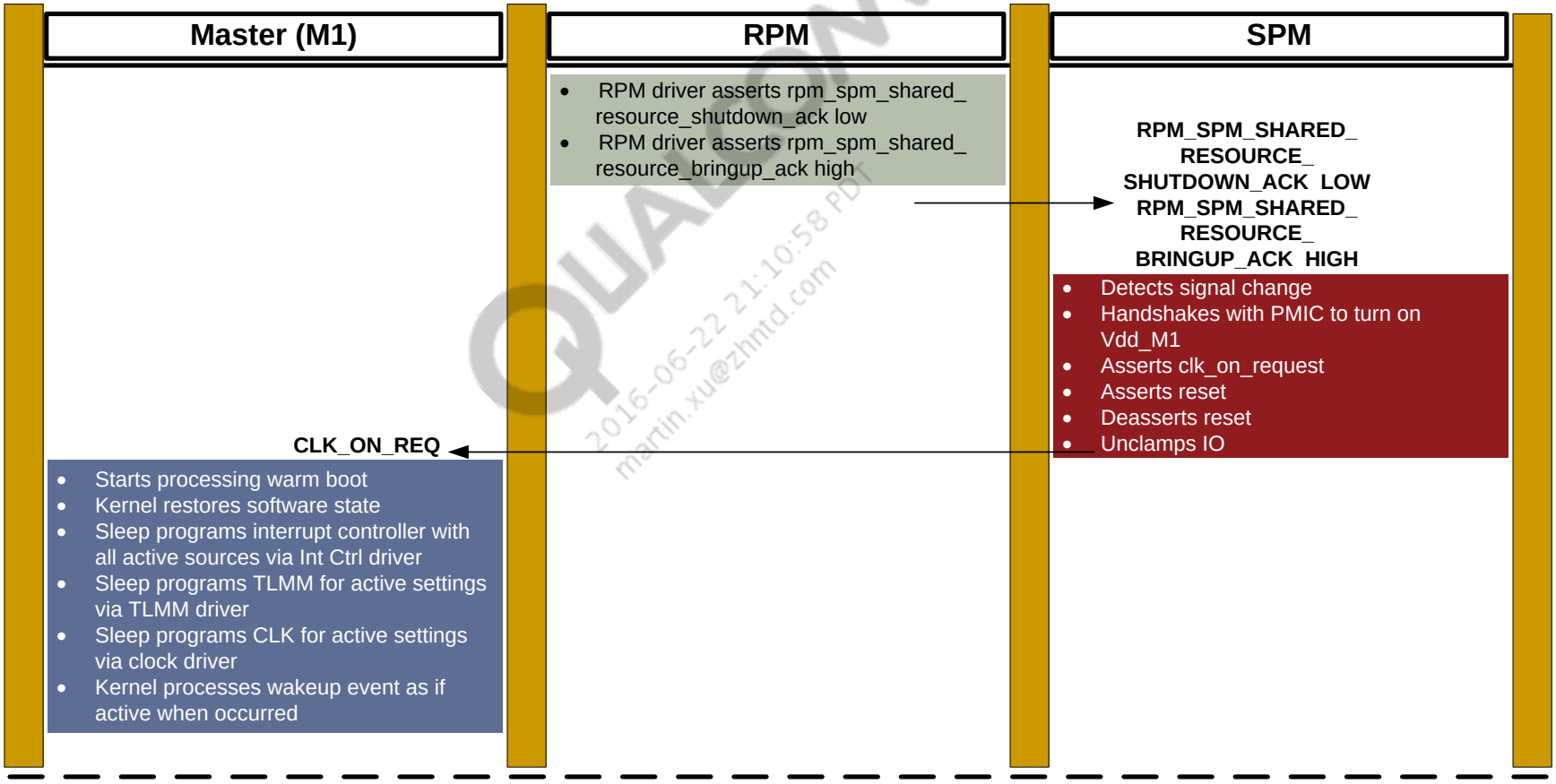


Subsystem Core Power Restore Flow

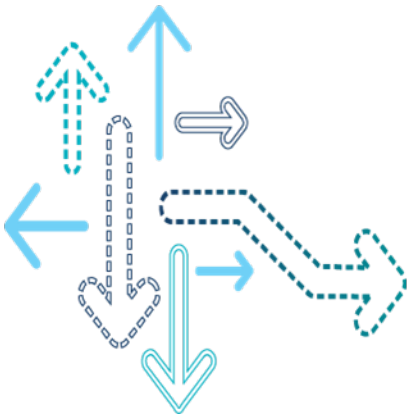
- Triggered from end of XO Restore/System Vdd_Min flow



Subsystem Core Power Restore Flow (cont.)



Rapid Bridge Core Power Reduction (RBCPR)

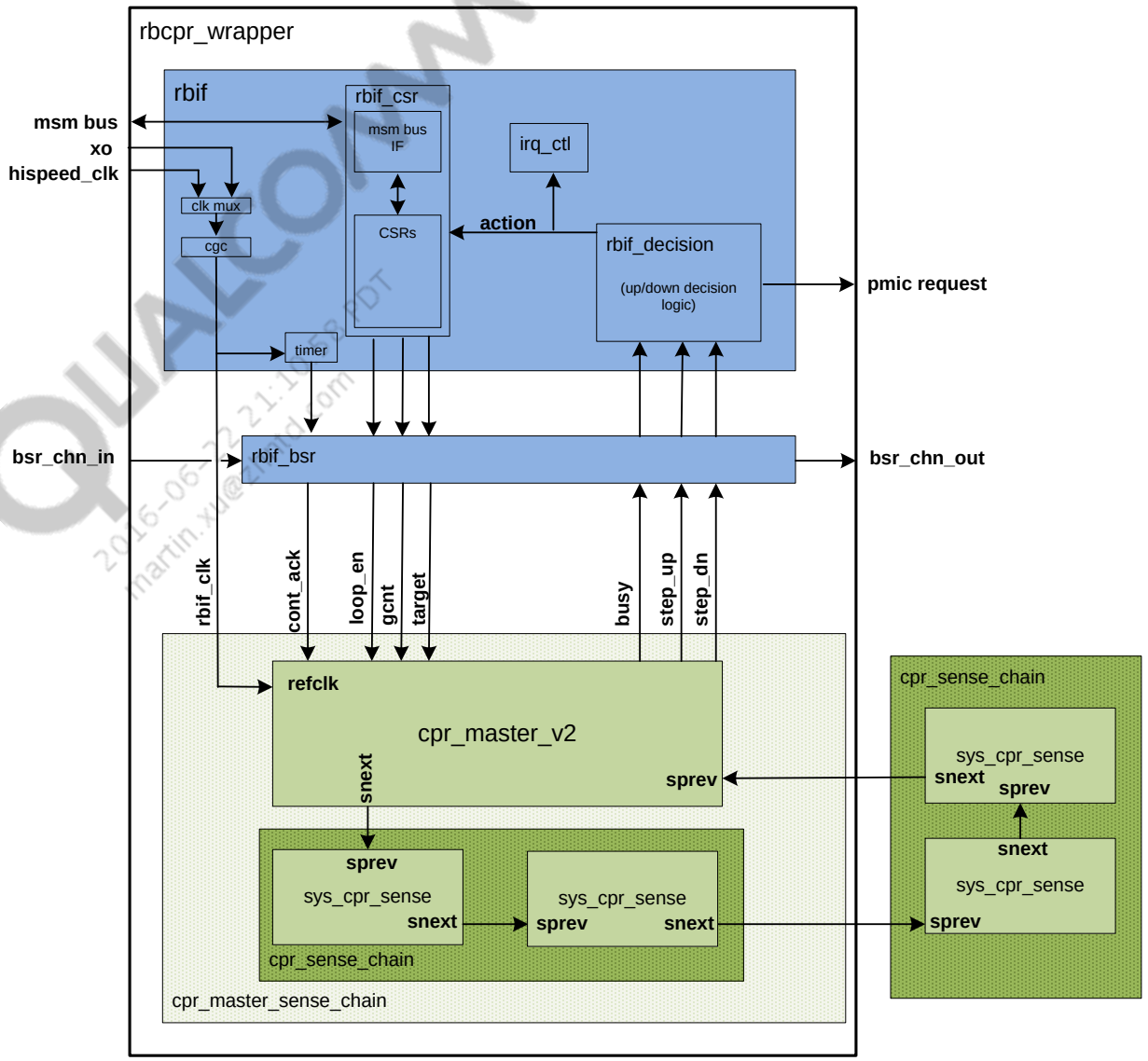


Overview

- RBCPR technology provides a feedback loop to optimize the voltage setting
- Two main use cases of RBCPR
  - Setup – When the voltage corner is changed and RBCPR configuration must be changed
  - Adjustment – When RBCPR hardware senses a voltage change is necessary and RBCRP software adjusts voltage
- RBCPR is expected to apply to the following domains and be controlled by the indicated execution environment
  - VDD Dig – Controlled by RPM
  - VDD GPU – Controlled by RPM
  - VDD EBI – Controlled by RPM

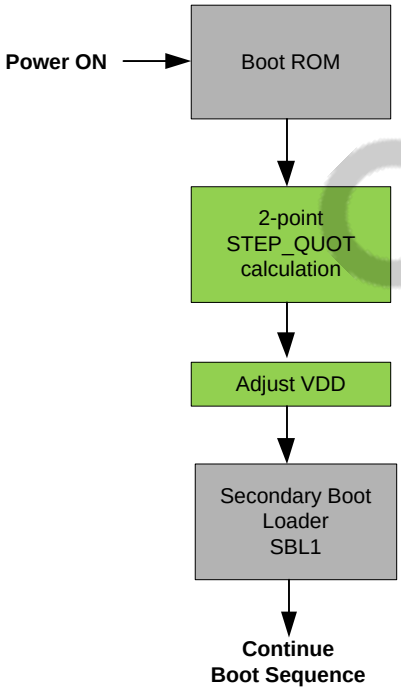
Block Diagram

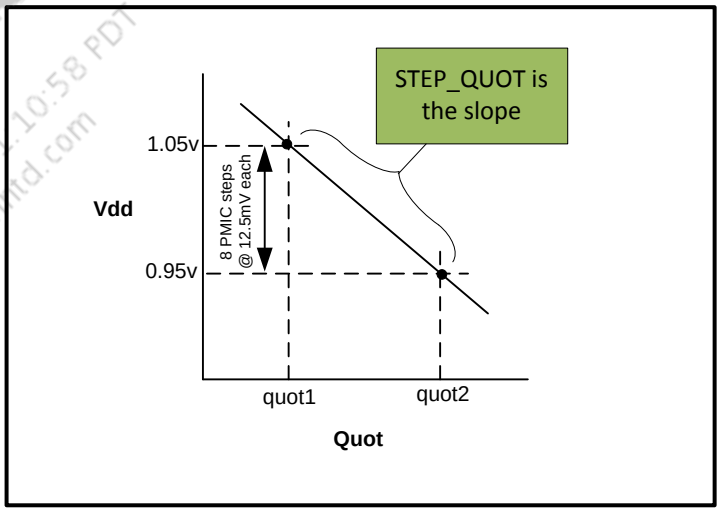
- The RBCPR core from Rapid Bridge consists of one master and a number of sensors (shown in green).
- There is an additional QTI wrapper logic called rbif (shown in blue).



Two-Point STEP_QUOT Calculation

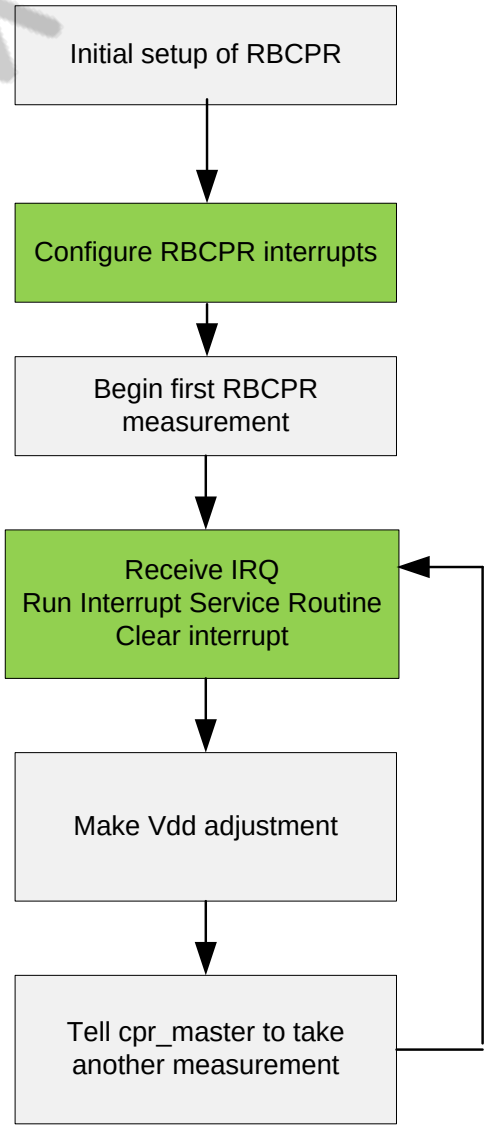
- STEP_QUOT is the number of QUOT units per PMIC step
- STEP_QUOT = (quot2 - quot1)/8
- 8 = (1.05 V - 0.95 V)/0.0125 V





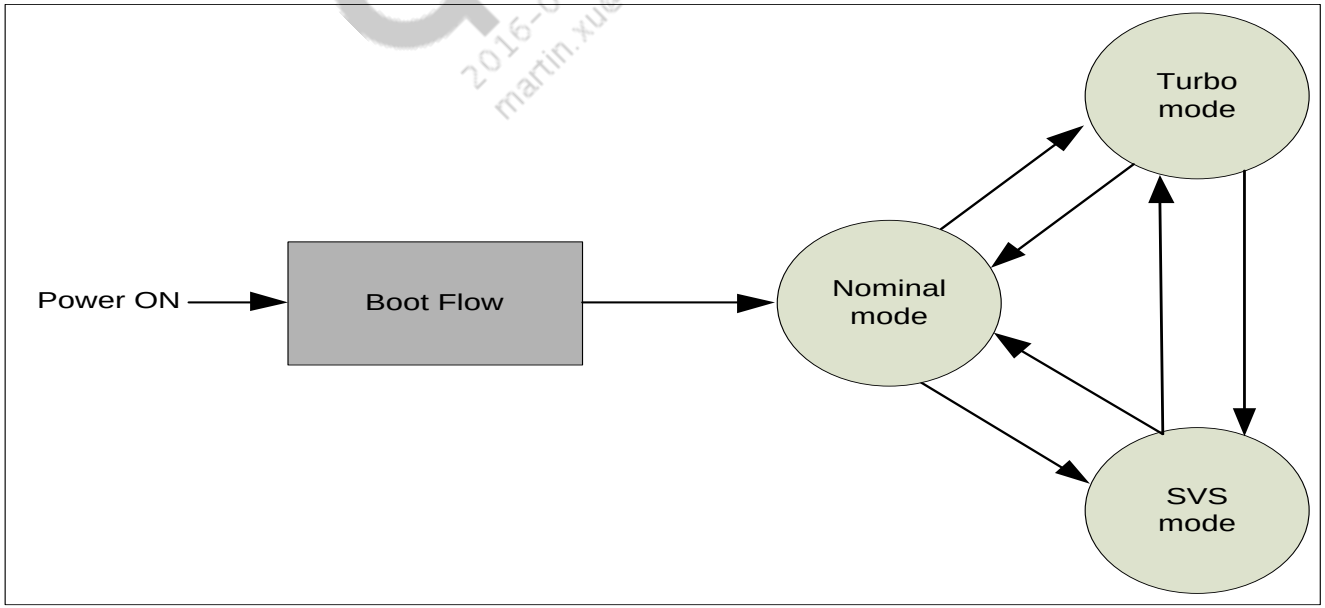
Adjustment

- Enable CPR interrupts (up_flag_en/down_flag_en/rbcpr_done_en)
- Receive CPR interrupt, look at RBCPR_STATUS, and make PMIC adjustment
  - If step_up is 1, increase PMIC VDD by one step
  - If step_down is 1, decrease PMIC VDD by one step
- Tell CPR to take another RBCPR measurement (write to RBCPR_CONT_ACK_CMD or RBCPR_CONT_NACK_CMD)

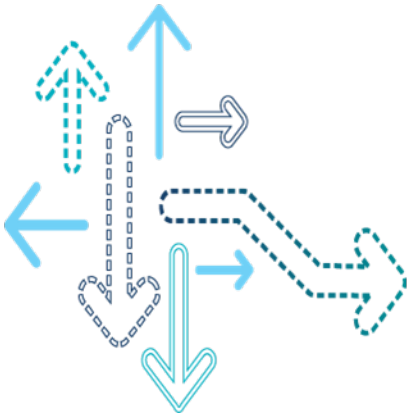


RPM Mode (Corner)

- RPM transitions from one mode to another
  - Nominal
  - Turbo
  - SVS
  - SVS2 (Introduced starting from 8994, slowest mode)
- CPR is notified of the mode transition and changes its configuration (gcnt/target pairs)



Debug



RPM External Log

- The RPM publishes a small log into a very limited area of DataRAM.
- The physical format of the log is the ULog format used for various other logs.
  - Circular buffer, currently sized at 16 KB
  - Raw log, using a set of IDs and a variable number of parameters per message

Saving RPM Dumps

- Customers can save the RPM dumps in the following manner and send for further analysis:
  1. Create the issue scenario where RPM dumps are needed.
  2. Open ARM7 Trace32 (T32) and attach (sys.m.a).
  3. Break T32 and do the following to save the RPM memory dump:
     - d.save.binary CODERAM.bin RPM_CODE_START_ADDR++(RPM_CODE_SIZE - 1)
     - d.save.binary MSGRAM.bin RPM_MSG_RAM_ADDR++(RPM_MSG_SIZE - 1)
     - d.save.binary DATARAM.bin RPM_DATA_START_ADDR++(RPM_DATA_SIZE - 1)

     or

     - do rpm_proc/core/bsp/rpm/scripts/rpm_dump.cmm \\location\to\put\logs

Loading RPM Dumps onto T32 and Extracting Logs

- To load RPM dumps received from the customer onto a T32 simulator:
  1. Open a T32 simulator and do sys.up.
  2. Jump to the dumps directory and load the same as shown below:
     - d.load.binary CODERAM.bin RPM_CODE_START_ADDR
     - d.load.binary MSGRAM.bin RPM_MSG_RAM_ADDR
     - d.load.binary DATARAM.bin RPM_DATA_START_ADDR

     or
     - d.load.elf \build\rpm\8974\build\RPM.elf /nocode
  3. Restore core registers after an err_fatal.
     - do rpm_proc/core/bsp/rpm/scripts/rpm_restore_core.cmm
  4. Recover the call stack from the exception handler or interrupt context.
     - do rpm_proc/core/bsp/rpm/scripts/rpm_m3_unstack.cmm

Extract RPM Logs with rpm_log_bfam.py

- Extracting logs
  1. Extract an RPM external log.
     - do rpm_proc\core\power\ulog\scripts\ULogDump.cmm <path to your directory>
  2. Extract an NPA log.
     - do rpm_proc\core\power\npa\scripts\NPADump.cmm <path to your directory>
  3. Execute the Python script for postprocessing.
     - python rpm_proc\core\power\rpm\debug\scripts\rpm_log_bfam.py -f “RPM External Log.ulog” -n “NPA Log.ulog” > rpm_parsed.txt

       Additional switches are –r, which print raw (hex sclk value) timestamps.

Hansei RAM Dump Parser

- Tool for parsing debug information out of the RAM dump; generates RPM logs, NPA logs, master status, resource states, etc.
- Installation
  1. Install Python 2.7.x (not 2.6.x).
     Check version – python –V.
  2. Install the pyelftools library 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 release
  1. Released since RPM 100
  2. Location – rpm_proc\core\bsp\rpm\scripts\hansei\

Hansei RAM Dump Parser (cont.)

- Usage

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

  python hansei.py –elf rpm.elf -o . rpm_code_ram.bin rpm_data_ram.bin rpm_msg_ram.bin
- Output
  - rpm-summary.txt – Contains general information about the health of the RPM, including the core dump state and fault information
  - rpm-log.txt – Postprocessed RPM 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

RPM External Log – Analysis

- Message request contents – Timestamp, operation, data

```
0x0000000009156ac1: rpm_message_received (master: "APSS") (message id: 151)

0x0000000009156bb5: rpm_svs (mode: RPM_SVS_FAST) (reason: imminent processing)

0x0000000009156ce8: rpm_process_request (master: "APSS") (resource type: clk1) (id: 1)
                    (full name: snoc)

0x0000000009156d23: rpm_xlate_request (resource type: clk1) (resource id: 1) (full name: snoc)

0x0000000009156d70: rpm_apply_request (resource type: clk1) (resource id: 1) (full name: snoc)

0x0000000009156df4: Clock: gcc_sys_noc_axi_clk           Frequency =  50MHz

0x0000000009156e73: rpm_send_message_response (master: "APSS")
```


RPM External Log – Analysis (cont.)

- Message request contents – Entering Low Power mode

```
0x00000037775e5068: rpm_shutdown_req (master: "MSS SW") (core: 0)
0x00000037775e50b7: rpm_shutdown_ack (master: "MSS SW") (core: 0)
0x00000037775e5b4a: rpm_transition_queued (master: "MSS SW") (scheduled: "no")
0x00000037775e5c4b: rpm_svs (mode: RPM_SVS_FAST) (reason: speedup) (old_duration: 0x0001f63e)
                    (new_duration: 0x000178ae) (switch_time: 0x000030e8)
...
0x00000037775e8462: rpm_master_set_transition (master: "MSS SW") (leaving: "Active Set")
                    (entering: "Sleep Set") (cache hit?: no)
...
0x00000037775f49f9: rpm_master_set_transition_complete (master: "MSS SW")
                    (deadline: 0x0000000000000000)                    --- Last EE power-collapse
0x00000037775f52be: rpm_transition_queued (master: "MSS SW") (scheduled: "yes")
                    (deadline: 0x0000003778158556)
0x00000037775f7313: deep_sleep_enter: (mode: "VDD Minimization") (count: 15918)
...
0x00000037775f84ad: deep_sleep_enter_complete: (mode: "VDD Minimization")  --- Enter VDD-MIN
0x000000377814afce: mpm_wakeup_ints (ints: 0x00000001 0x00000000)
0x000000377814b858: deep_sleep_exit: (mode: "VDD Minimization")
...
0x000000377814c4d7: deep_sleep_exit_complete: (mode: "VDD Minimization")   --- Exit VDD-MIN
0x000000377814c5eb: rpm_master_set_transition (master: "MSS SW") (leaving: "Sleep Set")
                    (entering: "Active Set") (cache hit?: yes)             --- Wakeup EE (Modem)
...
0x0000003778158d45: rpm_master_set_transition_complete (master: "MSS SW") (deadline: 0x0000003778158556)
0x0000003778158dab: rpm_bringup_req (master: "MSS SW") (core: 0)
0x0000003778158dec: rpm_bringup_ack (master: "MSS SW") (core: 0)
```

RPM NPA Log

- The RPM also contains an NPA log similar to those found on other processors.
- The RPM NPA log is considerably smaller than on other processors, so using NPADump.cmm to retrieve full NPA system state is suggested.
- Starting with RPM.00.00.71, this information is incorporated into the RPMLog, but NPADump must be run to get names.
  - ULOGDump.cmm – Gets logs
  - NPADump.cmm – Gets resource and client names
  - rpm_log.py – Parses the log to a readable format

NPA Log Deciphering

- Each of the following entry contains:
  - CLIENT – Master or RPM-based resource making the request
  - HANDLE – Unique identifier used to determine when request starts and completes
  - RESOURCE – Resource to which the client would like the request to be applied
  - REQUEST – Resource state being requested
- Example – LPASS votes against XO-SHUTDOWN

```
npa_client (name: APSS) (handle: 0x198ec0) (resource: 0x198c60) (type:
  NPA_CLIENT_REQUIRED) (request: 1)
```

- Example

```
npa_resource (name: "/xo/cxo") (handle: 0x198ca8) (units: Enable) (resource max: 1) (active max: 1)
            (active state: 1) (active headroom: 0) (request state: 1)
    npa_client (name: MPSS) (handle: 0x199fc8) (resource: 0x198ca8) (type: NPA_CLIENT_REQUIRED) (request: 0)
    npa_client (name: LPASS) (handle: 0x199e48) (resource: 0x198ca8) (type: NPA_CLIENT_REQUIRED) (request: 1)
    npa_client (name: APSS) (handle: 0x198ef8) (resource: 0x198ca8) (type: NPA_CLIENT_REQUIRED) (request: 0)
    npa_change_event (name: "sleep") (handle: 0x195e98) (resource: 0x198ca8)
    end npa_resource (handle: 0x198ca8)
```

VDD Minimum Issue Debug Steps

- System is not entering VDD minimization
  - With T32
    - Correlate RPM Log entry for xo_shutdown_enter with NPA log dump and examine the /node/sleep/uber requests that occurred directly before xo_shutdown_enter
  - Without T32
    - Dump RPM log from the apps
    - RPM log contains NPA log entries

Extending RPM External Log

- Interface to RPM external log provided in core\power\rpm\inc\rpm_log.h
- To add a log message:
  1. #define a new log ID to use in rpm_log.h
  2. Where the message should be logged, #include rpm_log.h and call the following macro:
     - RPM_LOG_EVENT(YOUR_LOG_ID, your_data1, your_data2);
     - Number of data elements can be variable, but each argument has a cost, so <4 arguments is recommended
     - Best practice is to have arguments in each position always mean the same thing for a given ID, i.e., first argument is always the master ID, second argument is always the resource, etc.
  3. rpm_log.py can then be extended to parse the new ID as required
     - core\power\rpm\dal\scripts\rpm_log.py

Key Debugging Structures and Variables

Question: Where is gpRPMFWMaster?

Answer: There is no gpRPMFWMaster; most information is reorganized in the RPM structure.

```
typedef struct
{
   unsigned num_ees;             // The number of EEData structures.
   unsigned supported_classes;   // The number of ResourceClassData structures.
   unsigned supported_resources; // The number of ResourceData structures.

   EEData *ees;
   ResourceClassData *classes;   // Stored in order of registration.
   ResourceData      *resources; // Indexed by perfect hash lookup.
} SystemData;

extern SystemData * const rpm;
```


Debugging System Sleep

- SPM state for each master
  - rpm.ees[master_id].subsystem_status
- Sleep counts
  - sleep_stats[0] / sleep_stats[1]
- How to disable deep sleep
  - Set sleep_allow_low_power_modes = FALSE

Debugging Railways

- Railway configuration
  - Railway_config.c
- Check rail votes
  - Railway.rail_state[x]; x= rail# (mx=0/cx=1/gfx=2)
  - Follow voter_list_head and voter_link to see all voters
  - Voter ID

    master number (0: APPS, 1: MODEM, 2: QDSP, 3: RIVA)

```
typedef enum
{
    RAILWAY_SVS_VOTER_ID = 100,
    RAILWAY_RPM_CX_VOTER_ID,
    RAILWAY_RPM_MX_VOTER_ID,
    RAILWAY_DDR_TRAINING_VOTER_ID,
    RAILWAY_RPM_BRINGUP_VOTER,
    RAILWAY_RPM_INIT_VOTER,
    RAILWAY_CLOCK_DRIVER_VOTER_ID,
    RAILWAY_CPR_SETTLING_VOTER,
} railway_voter_id;
```

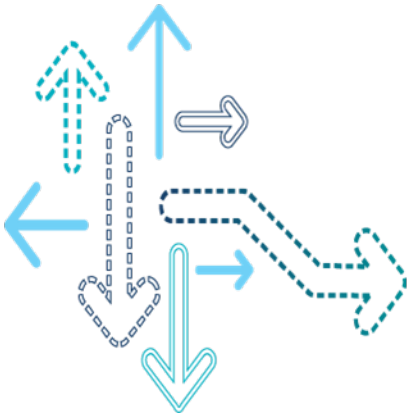

Debug RBCPR

- How to disable RBCPR
  - To disable CPR on an RPM build, edit the following file:
    rpm_proc\core\power\rbcpr\src\target\<target>\rbcpr_bsp.c
    and set all instances of .use_this_cpr_block in this file to False.
  - For MSM8994 and later targets, never disable CPR, use open-loop instead. In an SBL build, boot_images\core\power\rbcpr\src\target\<target>\rbcpr_bsp.c, set all instances of .rbcpr_enablement to RBCPR_ENABLED_OPEN_LOOP.
- RBCPR status (rbcpr_stats)
  - CPR stats collects information on voltage scaling recommendations from CPR hardware.
    - Fuse voltage (CPR starting point)
    - For each mode (SVS/Normial/Turbo):
      - # of interrupts in the mode
      - Latest recommendations with timestamps
      - Programmed voltage to railway
      - Exception events – Recommended voltage hitting Min or Max

Disable PMIC Watchdog

- PMIC WDOG recovers the device from lockup
  - PMIC WDOG barks in 17 sec
  - PMIC WDOG bites in 18 sec
  - RPM pets PMIC WDOG every 15 sec or less
- Disable PMIC WDOG
  - Set pmic_wdog_enable to 0 and rebuild RPM

References



References

Document	
Qualcomm Technologies, Inc.	
Presentation: Resource Power Manager (RPM) Overview and Debug	80-VP169-1
Resource Power Manager User Guide	80-N6955-1
Resource Power Manager (RPM.BF) User Guide	80-NA157-15

Questions?

https://createpoint.qti.qualcomm.com