Linux Power Management Details

80-VR652-1 A

Qualcomm Confidential and Proprietary

Restricted Distribution. Not to be distributed to anyone who is not an employee of either Qualcomm or a subsidiary of Qualcomm without the express approval of Qualcomm's Configuration Management.

Not to be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm.

Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis.

This document contains Qualcomm confidential and proprietary information and must be shredded when discarded.

QUALCOMM is a registered trademark of QUALCOMM Incorporated in the United States and may be registered in other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners. CDMA2000 is a registered certification mark of the Telecommunications Industry Association, used under license. ARM is a registered trademark of ARM Limited. QDSP is a registered trademark of QUALCOMM Incorporated in the United States and other countries.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

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

Copyright © 2009 QUALCOMM Incorporated.
All rights reserved.

Revision History

Version	Date	Description
A	Jun 2009	Initial release

QUALCOMM 2009.06.30 at 01:19:26 PDT frankyang-arimacomm.com.tw

Note: The BSP information contained in these training materials and presented in connection with these training materials is to be used solely in connection with QUALCOMM® ASICs, related software and documentation.

Contents

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Introduction

QUALCOMM®
2009.06.30 at 19:26 PDT
frankyang-arimac.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Introduction

- This document focuses on how device drivers plug into the Linux power management frameworks in MSM7xxx/QSD chipsets.
- It discusses, in detail, the Linux power management frameworks used.
- It also describes the responsibilities of device drivers for maximum power saving.
- It shows the tight interaction between the modem and the Apps for power management.
- This document does not discuss debugging of power management features (refer to [Q2]).

QUALCOMM CDMA Technologies

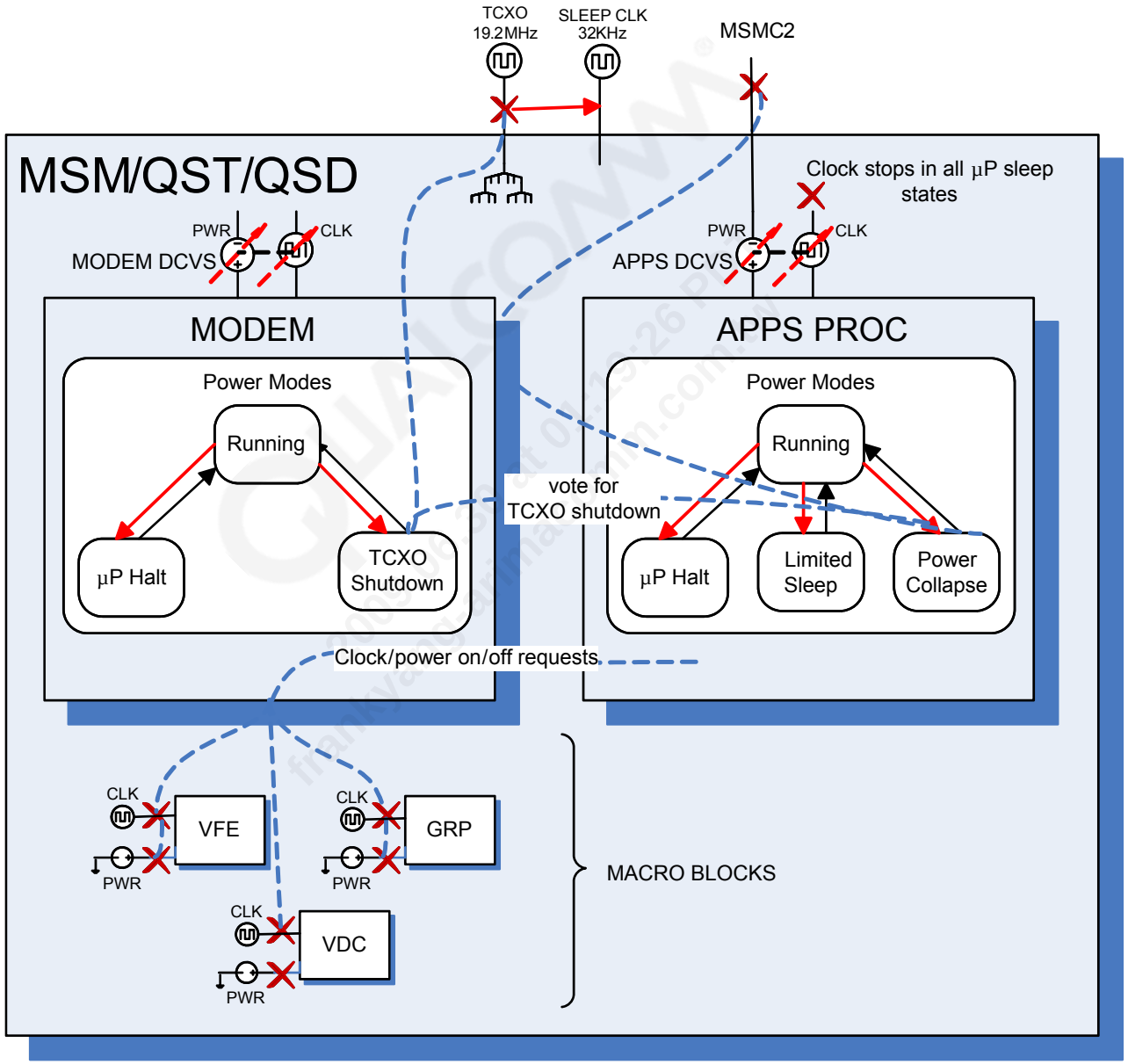
MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Linux Power Modes

2009.06.30 at 19:26 PDT
frankyang-arimacomm.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

QSD/QST/MSM7xxx PM Overview



MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Linux Power Modes – Overview

- Power modes
  - Running
  - Sleep
    - MSM Sleep = Apps power collapse + modem TCXO shutdown
    - Limited Sleep (Apps power collapse only) = Apps power collapse + Apps votes against modem TCXO shutdown
    - SWFI (only) = Apps executes SWFI instruction; no Apps power collapse or modem TCXO shutdown
    - Spins = Apps CPU spins; no Apps power collapse; no Modem TCXO shutdown
  - Suspend
    - Apps power collapse + modem TCXO shutdown + off state for all hardware devices (clocks will be off, see next bullet)
    - Drivers notified of pending suspend and will go into lower Power mode and disable clocks

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Linux Power Modes – Sleep

- Conditions for Sleep modes
  - Interrupt
    - Wake-up interrupt – Able to wake up Apps from power collapse
    - Nonwake-up interrupt – Unable to wake up Apps from power collapse
      - » This type of interrupt prevents AP from Idle power collapse
  - Latency
    - Time of entering and exiting a Low Power mode
    - Drivers' PM_QOS latency requirement (MIN) is checked against:
      - » Biggest Latency (msm_pm_data[MSM_PM_SLEEP_MODE_POWER_COLLAPSE].latency in arch/arm/mach-msm/board-xxxx.c)
        – Apps power collapse + Apps votes for modem TCXO shutdown
      - » Intermediate Latency (msm_pm_data[MSM_PM_SLEEP_MODE_POWER_COLLAPSE_NO_XO_SHUTDOWN].latency)
        – Apps power collapse + Apps votes against modem TCXO shutdown
      - » Small Latency (msm_pm_data[MSM_PM_SLEEP_MODE_WAIT_FOR_INTERRUPT].latency)
        – Apps clock ramps down and SWFI
      - » [smaller than Small Latency – Spin]

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Linux Power Modes – Sleep (cont.)

- Conditions for Sleep modes (cont.)
  - Residency
    - Time threshold (breakeven point) to save power next timer event is checked
  - Clocks are remotely monitored by clkrgm/Modem for TCXO shutdown
    - All clocks are disabled/enabled via proc_comm by clkgrm/Modem

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Linux Power Modes – Suspend

- Suspend
  - Entered from user application context by issuing suspend command through sysfs (/sys/power/state)
    - By key press from user (supported by Android UI)
    - After timeout of inactivity (supported by Android UI)
    - For details, refer to [Q2]
  - Drivers notified of suspend via callback
  - Drivers must enter their lowest Power mode
  - Apps CPU enters Power-Collapse state
  - Modem can enter TCXO shutdown
  - Since it is the longest sleep with power collapse and Low Power (off) mode for all hardware, this mode saves the most power

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Early Suspend and Late Resume

2009.06.30 at 19:26 PDT
frankyang-arimacomm.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Early Suspend and Late Resume

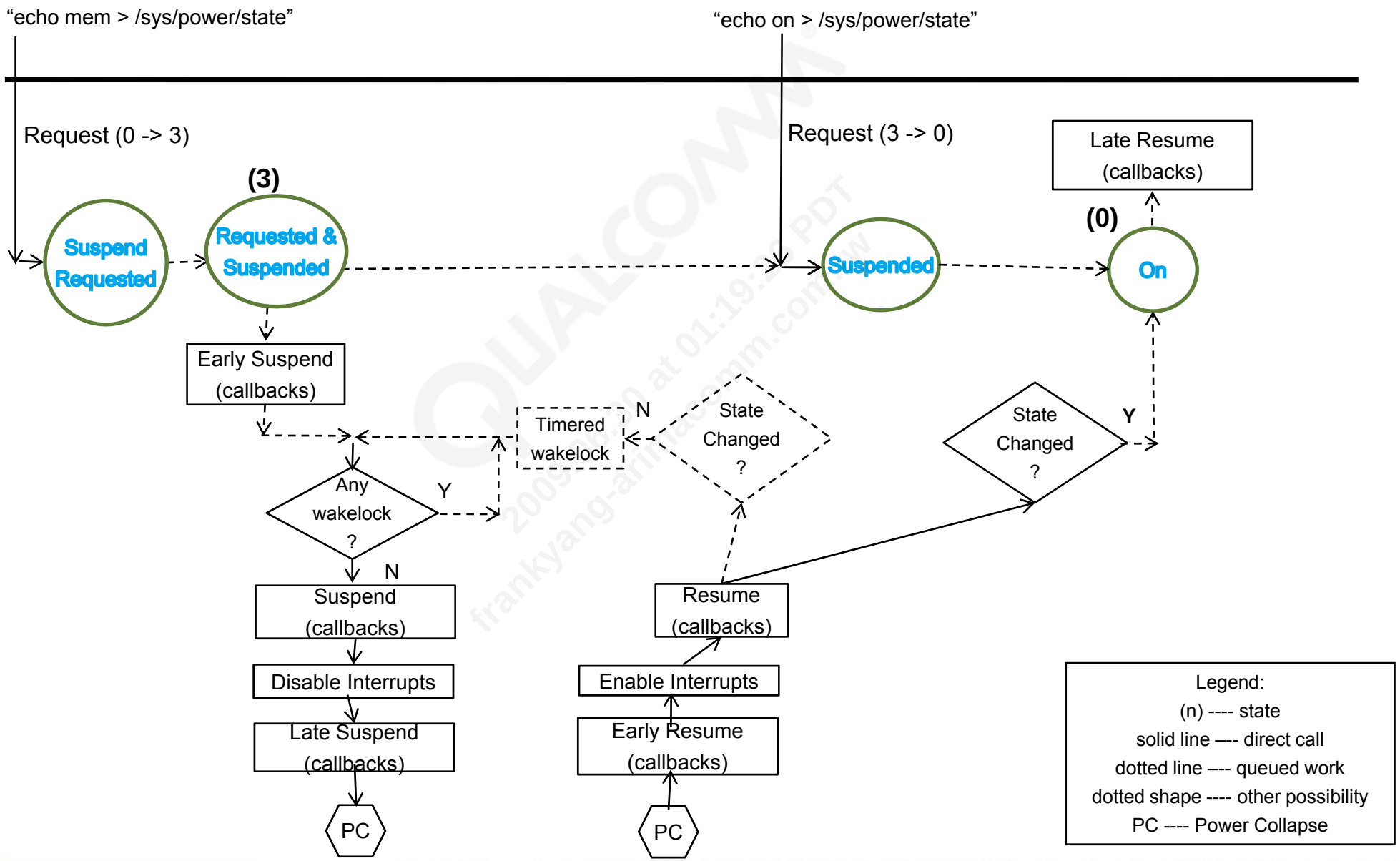
- Added in the kernel for Google/Android™
  - Early-suspend – Before normal suspend (refer to figure on next slide)
    - Drivers notified of early-suspend via callback
    - Drivers can put hardware into Low Power mode ahead of normal suspend callback
  - Late-resume – After regular resume (refer to figure on next slide)
- Users (Some drivers currently using early_suspend/late_resume)
  - Display driver
  - Keypad driver
  - Touchscreen driver
  - Audio driver
  - Battery driver

Note: Regular suspend can be held off by wakelocks but early_suspend is always called even when there are wakelocks (more details in next few slides).

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Early Suspend and Late Resume (cont.)



MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Early Suspend and Late Resume (cont.)

- Brief flow of the figure
  - "*echo mem > /sys/power/state*" causes suspend request
    - Makes the kernel platform call early_suspend () callbacks on all drivers registered for early_suspend()
    - A kernel thread is then spun, which will check for all wakelocks held; this thread uses a timer
      - » No wakelock → real Suspend (suspend() callbacks on all drivers called)
    - Interrupts are disabled
    - late_suspend() callbacks called, then Apps goes into power collapse

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Early Suspend and Late Resume (cont.)

- Brief flow of the figure (cont.)
  - When resuming, early_resume() callbacks called
    - Interrupts may be enabled as needed
    - resume() callbacks are called on drivers
    - From here, only UI events (for example, "*echo on > /sys/power/state*") will cause late_resume() callbacks to be called
      - This causes late_resume() on the display driver, which eventually turns on LCD
    - In some cases, display should not be turned on and kernel should go back into suspend (e.g., RPC call wakes up the Apps and the call is handled by the kernel, and then kernel can go back into suspend; hence, no need to wakeup/resume the LCD)

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Wakelock for Suspend

2009.06.30 at 19:26 PDT
frankyang-arimacomm.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Wakelock for Suspend

- Added in the kernel for Google/Android
- To give drivers/Apps a chance to finish transaction in the process of, but before suspend
- Users (some drivers currently using wakelocks)
  - USB
  - Keypad
  - Audio
- Suspend/wakelock contention
  - When suspend operation has already started while locking a wakelock, suspend operation will be aborted as long it has not already reached suspend_late stage
    - Locking a wakelock from an interrupt handler or a freezeable thread always works, but if wakelock is locked from a suspend_late handler, an error should be returned from that handler to abort suspend

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Wakelock for Suspend (cont.)

- Driver APIs and their usage:
  - Add a wakelock variable to driver state and call wake_lock_init.

```
struct state {
    struct wakelock wakelock;}
init() {
    wake_lock_init(&state->wakelock, WAKE_LOCK_SUSPEND,
       "wakelockname");
```

  - Before freeing the memory, wake_lock_destroy must be called:

```
uninit() {
    wake_lock_destroy(&state->wakelock); }
```

  - When the driver determines that it needs to run (usually in an interrupt handler) it calls wake_lock:

```
wake_lock(&state->wakelock);
```

  - When it no longer needs to run it calls wake_unlock:

```
wake_unlock(&state->wakelock);
```


MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Wakelock for Suspend (cont.)

- Driver APIs and their usage (cont.):
  - Calls wake_lock_timeout to release the wakelock after a delay:

```
wake_lock_timeout(&state->wakelock, HZ);
```

    - This works whether the wakelock is already held or not. It is useful if the driver woke up other parts of the system that do not use wakelocks, but still needs to run. This should be avoided whenever possible, since it will waste power if the timeout is long, or may fail to finish needed work if the timeout is short.

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Wakelock for Suspend (cont.)

- User space APIs and their usage:
  - Write "lockname" or "lockname timeout" to /sys/power/wake_lock lock and, if needed, create a wakelock (the timeout here is specified nanoseconds)
  - Write "lockname" to /sys/power/wake_unlock to unlock a user wakelock
  - Do not use randomly generated wakelock names, as there is no API to free a user space wakelock

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

QoS Provided by Power Management

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

QoS Provided by PM

- Try to fill the gap between performance and power saving
  - New API on kernel version 2.6.25
- Algorithms
  - Minimum
  - Maximum
- QoS parameters (or services) (like nodes in NPA)
  - PM_QOS_CPU_DMA_LATENCY
    - Use MIN algorithm in sleep decision
  - PM_QOS_SYSTEM_BUS_FREQ (added by QC)
    - Use MAX algorithm in clock driver

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

QoS Provided by PM (cont.)

- APIs
  - Register
    - » pm_qos_add_requirement(qos-parameter, driver_name, value/PM_QOS_DEFAULT_VALUE)
    - » User space – qos_fd = open (“/dev/[qos-parameter]”)
  - Request a QoS
    - » pm_qos_update_requirement(qos-parameter, driver_name, new-value)
    - » User space – write (qos_fd, new_value)
    - » Call this before entering QoS-sensitive code section (many times/different values)
  - Remove all previously requested QoS
    - » pm_qos_update_requirement (parameter, driver_name, PM_QOS_DEFAULT_VALUE)
    - » User space – write (qos_fd, -1)
    - » Call this after exiting QoS-sensitive code section
  - Unregister
    - » pm_qos_remove_requirement(parameter, driver_name)
    - » User space – close (qos_fd)

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

CPUFREQ

2009.06.30 at 19:26 PDT
frankyang-arimac.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

CPUFREQ – Linux Dynamic PM

- CPU frequency scaling (CPUFREQ) is PM method used in Running mode
- Switching of CPU frequency is governed by these governors in static or dynamic manner
  - Performance – CPU runs at static maximum frequency
  - Powersave – CPU runs at the static minimum frequency
  - User space – Determined by static user space program
  - Ondemand – On-demand dynamic governor sets target frequency based on CPU busy/idle statistics
  - Conservative – Conservative dynamic governor sets target frequency, based on CPU busy/idle statistics
- Uses MSM-dependent CPU driver underneath
  - AXI speed (vote) is tied to CPUFREQ table using perf-level
  - SVS (and AVS on 8 k) is tied to CPUFREQ table

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

CPUFREQ – Linux Dynamic PM (cont.)

- Sysfs interface – */sys/devices/system/cpu/cpu0/cpufreq/**
- Advantages
  - Users can change governors (refer to [Q2] for details)
  - Users can change speeds for user space governor
  - Users can change parameters for dynamic governors
    - Up threshold
    - Down threshold
    - Sampling rate
  - SVS and AVS can be tied with ACPU speed
- Disadvantages of dynamic governors
  - Reactive and timer based – though a deferred timer
  - Default up/down thresholds (80/50) do not fit for all
  - Default sampling rate (200 ms) does not fit for all
  - Does not know how to scale BUS clock

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Cpufreq – Ondemand Governor

- Kernel times – user; nice; system; softirq; irq; idle; iowait; steal
- Busy time – user + system + irq + softirq + steal [+ nice]
- Load % – Busy time/total time
- Parameters
  - sampling_rate: [200 ms – between 100 ms and 100 sec]
  - up_threshold: [80 %]
  - down_threshold: [70 % – changed to 50 for MM]
  - powersave_bias: [1 – between 0 and 1000]
  - ignore_nice_load: [0] (1: nice = idle; 0: nice = busy)
- Algorithm
  - Up = MAX * [(1000 – powersave_bias) / 1000]
  - Down = Cur * (load% / down_threshold) * [(1000 – powersave_bias) / 1000]

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Cpufreq – Conservative Governor

- Kernel times – Same as Ondemand
- Busy time – Same as Ondemand
- Load % – Same as Ondemand
- Parameters
  - sampling_rate: [500 ms – between 250 ms and 250 sec]
  - up_threshold: [80 %]
  - down_threshold: [20 %]
  - sampling_down_factor: [1 – between 1 and 10]
    - sampling_rate * sampling_down_factor -> down_sampling_rate
  - freq_step: [5 %]
  - ignore_nice_load: [0] (1: nice = idle; 0: nice = busy)
- Algorithm
  - Up = Cur + MAX * (freq_step / 100)
  - Down = Cur – MAX * (freq_step / 100)

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Power-Saving Guidelines

2009.06.30 at 19:26 PDT
frankyang-arimac.com.tw

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Guidelines for Power Savings

- All drivers
  - Must support suspend
  - Do not set periodic timers
    - Processors woken up on next timer expiration, so this kills power savings
  - Take advantage of deferred timers to alleviate requirement to wake up for your timer
  - Use inactivity timer to turn off
    - Clocks
    - Non-wakeup interrupts
  - Actively manage
    - VREG / MPP / GPIO

QUALCOMM CDMA Technologies

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Drivers Responsibilities

- Drivers responsibilities
  - Suspend
    - Register the device driver with Linux kernel and support .suspend and .resume callbacks
    - Put device in lowest Power mode in .suspend function
      - » device h/w and any VREGs/GPIOs/MPPs used by the device
    - Release wakelock
  - Idle power collapse
    - When not in use, driver should disable its interrupts, GPIOs, and clocks so that the Apps processor can enter power collapse and modem processor can enter TCXO shutdown
    - If possible, driver should use deferrable timer or range timer instead of regular timer

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Drivers Responsibilities (cont.)

- Drivers responsibilities (cont.)
  - CPUFREQ
    - Design and write power optimized drivers; debugging and optimizing power on later stages is an ordeal
    - Avoid burst CPU requirements from drivers, if possible
    - When using ondemand governor, CPU frequency will jump the highest when load is above the up_threshold; use judgement when executing loops, relinquish CPU as much as possible; this spreads out the load and provides better power numbers
    - Test for power and performance as new features are added
    - Test drivers setting ondemand as the governor and compare it against user space at a reasonable speed. If ondemand seems to be running at a higher frequency than it should, then
      - » Try tweaking the ondemand parameters (up_threshold, down_threshold, sampling_rate, scaling_min_freq etc)
      - » Review loops and spin locks in your code

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Drivers Responsibilities (cont.)

- Drivers Responsibilities (cont.)
  - How to handle resume from suspend:
    - Register device driver with Linux kernel and support .resume callback
    - In general, resume code should undo everything that was done in suspend code:
      - » If the driver saves hardware settings and powers down the hardware in suspend, it should power up the hardware and restore the hardware settings.
      - » If the driver disables services in suspend, it should re-enable the services.
    - The driver should delay enabling of interrupts, clocks, GPIOs, vreg, etc., for as long as possible, until needed
      - » For example, if the I^2C driver disables I^2C interrupt in suspend, the driver can probably skip the re-enabling of the I^2C interrupt in resume. Instead I^2C driver will re-enable the interrupt when an I^2C transaction is requested.
      - » Good drivers should disable resources (interrupt, clocks, timers, etc.) when not needed and re-enable them when they are needed.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Drivers Responsibilities (cont.)

- Drivers responsibilities (cont.)
  - Wakelocks
    - It is recommended that drivers should not use wakelock unless needed
      - When the kernel goes into regular suspend, i.e., after all wakelocks have been released, the kernel will call the suspend (and suspend_late) function of each registered driver (refer to diagram for early suspend).
      - In suspend(), the driver should put the respective hardware into a suspended state. If the driver cannot do this, e.g., if it is waiting for an outstanding transaction to finish, it should hold a wakelock so that the kernel will not enter regular suspend.
      - Whether a driver will need to use wakelock depends on whether the driver can finish (and finish quickly) all its work in the suspend function. If it can, it does not need wakelock; if it cannot, it will need wakelock.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Drivers Responsibilities (cont.)

- Drivers responsibilities (cont.)
  - Early_suspend/late_resume
    - Drivers should try to use the early-suspend/late-resume mechanism
      - » If a driver holds a wakelock, it is strongly recommended to register for early-suspend so that it gets notified via early-suspend callback when the suspend command is issued.
      - » Once the driver receives the early_suspend notification, it should stop accepting new transactions/requests, finish the outstanding one, and release the wakelock.
      - » If the driver does not hold a wakelock, it means the driver is ok with regular kernel suspend. Most likely, the driver should be able to perform its job in suspend instead of early-suspend.
      - » There may be some special cases where the driver needs to go into Low Power mode before the regular kernel suspend starts. In these cases, early-suspend is a solution.

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Device Low-Power Mode APIs

- Use the following APIs to allow sleep or avoid current leakage when
  - In driver's suspend function
  - Not in use
- Clock APIs
  - Include <linux/clk.h>
  - Use clk_disable () when clock is not needed, so suspend or sleep (TCXO shutdown) can proceed; power saved when clocks turned off
- Voltage Regulator (VREG) APIs
  - Include <asm/arch/vreg.h>
  - Use vreg_disable () to disable voltage regulator on PMIC
  - Use vreg_set_level () to set voltage regulator at lower voltage level

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Device Low-Power Mode APIs (cont.)

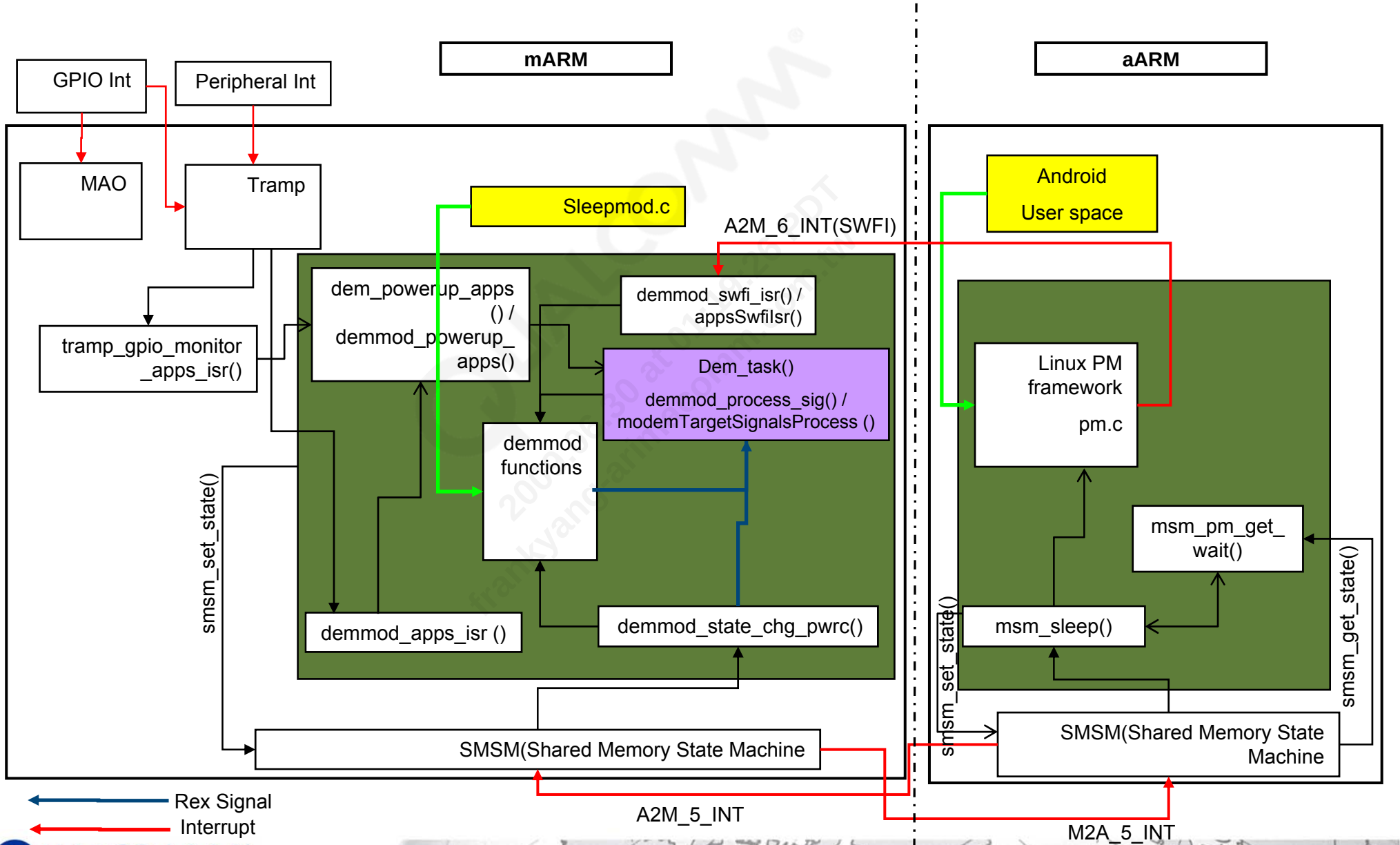
- Multipurpose Pin (MPP) APIs
  - Include <asm/arch/mpp.h>
  - Use mpp_config_digital_out() to set MPP pin on PMIC to correct state
    - Example
      - » Output logic level – MSME(1.8v), MSMP(2.6v), MMC(3v), VDD(3.6v)
      - » Output control – High, Low(0v), MPP Input, Inverted MPP Input
- GPIO Top-Level Mode Mux (TLMM) APIs
  - Include <asm/arch/gpio.h>
  - Use gpio_tlmm_config() to set driver GPIOs to correct state
    - Configurations
      - » Enable/disable
      - » Input/output
      - » Pull-up/pull-down/keeper/no pull
      - » Drive strength (current) – 2 mA to 16 mA (example)

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Apps and Modem Interaction

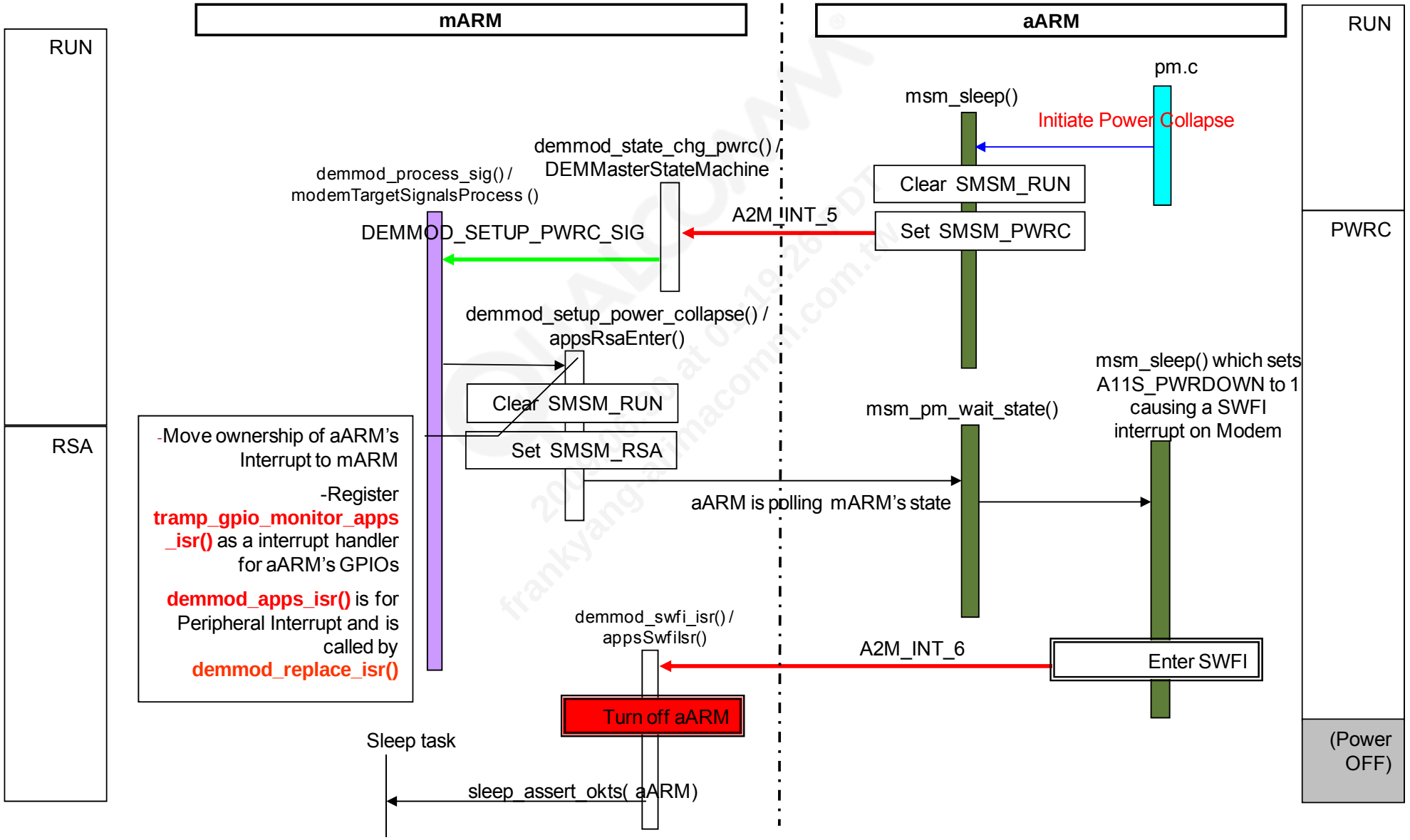
2009.06.30 at 19:26 PDT
frankyang-arimacomm.com.tw

Detailed Modem and Apps Interaction



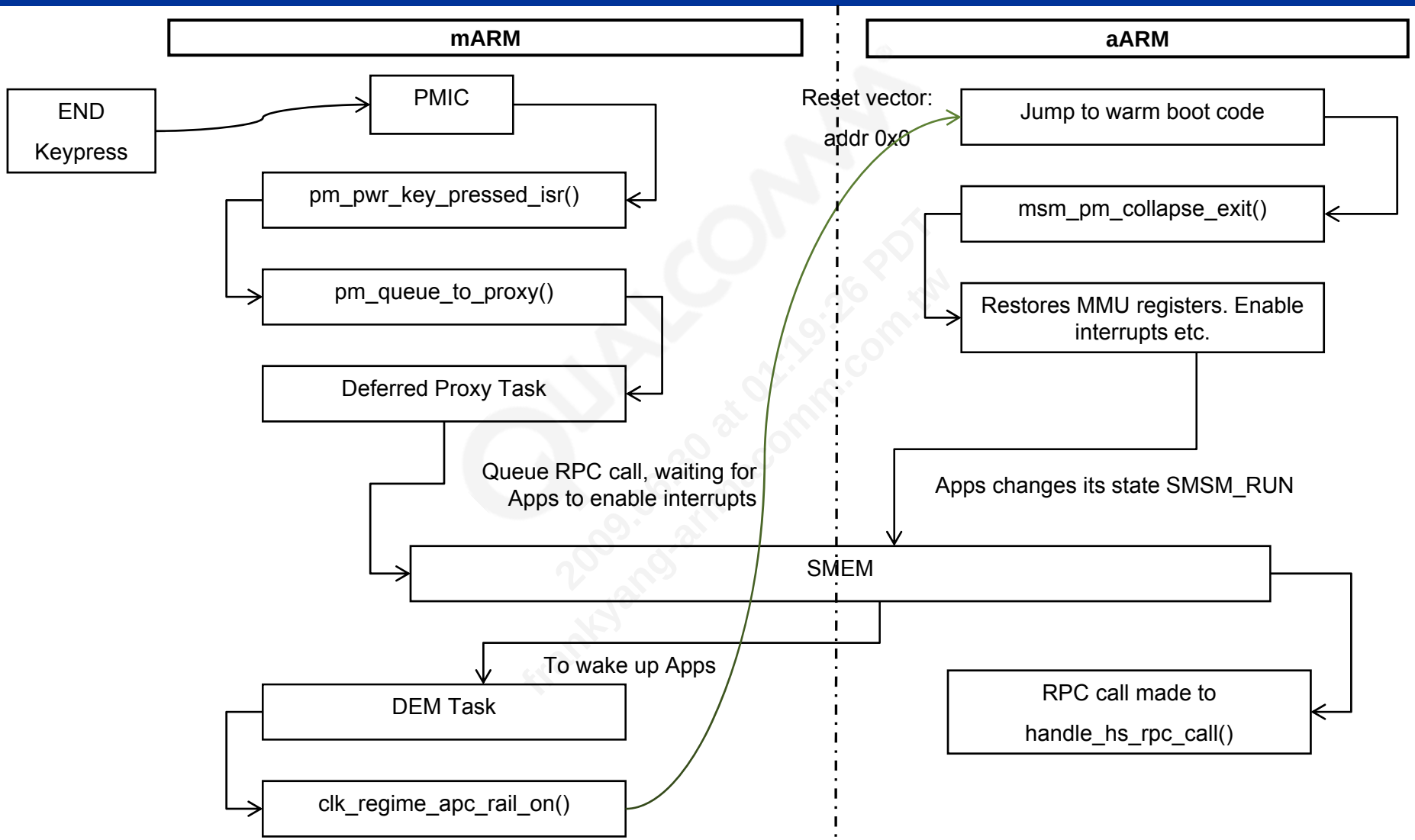
MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

Enter Power Collapse (Suspend/Idle)



MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

END Keypress to Wake up



MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION

References

Ref.	Document	
Qualcomm		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Linux Power Management Debugging Guide*	80-VR629-1 A

QUALCOMM
2009.06.30 at 01:19:26 PDT
frankyang-arimacomm.com.tw

Questions?

https://support.cdmatech.com

MAY CONTAIN U.S. AND INTERNATIONAL EXPORT CONTROLLED INFORMATION