



DM3730/AM3703 SOM Power Management with DM37x Linux BSP

Application Note 597

Logic PD // Products
Published: August 2014

This document contains valuable proprietary and confidential information and the attached file contains source code, ideas, and techniques that are owned by Logic PD, Inc. (collectively "Logic PD's Proprietary Information"). Logic PD's Proprietary Information may not be used by or disclosed to any third party except under written license from Logic PD, Inc.

Logic PD, Inc. makes no representation or warranties of any nature or kind regarding Logic PD's Proprietary Information or any products offered by Logic PD, Inc. Logic PD's Proprietary Information is disclosed herein pursuant and subject to the terms and conditions of a duly executed license or agreement to purchase or lease equipment. The only warranties made by Logic PD, Inc., if any, with respect to any products described in this document are set forth in such license or agreement. Logic PD, Inc. shall have no liability of any kind, express or implied, arising out of the use of the Information in this document, including direct, indirect, special or consequential damages.

Logic PD, Inc. may have patents, patent applications, trademarks, copyrights, trade secrets, or other intellectual property rights pertaining to Logic PD's Proprietary Information and products described in this document (collectively "Logic PD's Intellectual Property"). Except as expressly provided in any written license or agreement from Logic PD, Inc., this document and the information contained therein does not create any license to Logic PD's Intellectual Property.

The Information contained herein is subject to change without notice. Revisions may be issued regarding changes and/or additions.

© Copyright 2014, Logic PD, Inc. All Rights Reserved.

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	SWE	-Initial Release	AF, BSB	08/26/14

Table of Contents

1	Introduction	1
1.1	Nomenclature.....	1
2	Hardware Power Management Features	1
2.1	DM37x Processor Power Domains.....	1
2.1.1	Power Domain States.....	2
2.2	DM37x Clock Domains	3
3	Linux Power Management Features	4
3.1	Prerequisites	4
3.2	Active Power Management – <i>CPUfreq</i>	6
3.2.1	Tune <i>ondemand</i> Governor	7
3.2.2	Tune <i>conservative</i> Governor	8
3.2.3	Debug <i>CPUfreq</i> – Trace Clock Rate Transitions	9
3.3	Idle Power Management - <i>CPUidle</i> Framework	10
3.3.1	Prerequisites.....	10
3.3.2	C-states	15
3.3.3	Kernel Tick Effect on Idle Power Management	15
3.3.4	Governors	16
3.3.5	Access <i>CPUidle</i> Framework from Linux Userspace	17
3.4	Suspend	19
3.4.1	Debug Suspend from Linux Userspace	19
3.4.2	Low-Power Design Recommendation.....	27
	Appendix A: Linux Power Management Subsystem	28

1 Introduction

Linux has a rich set of power management features that are available to system developers. Many of the features are discussed in Linux documentation or online; this application note integrates these sources to form a complete picture of the embedded Linux power management features available in the Logic PD DM37x Linux Board Support Package (BSP). After completing this document you should be prepared to manage power on your Linux-based system.

1.1 Nomenclature

- This document covers the DM3730/AM3703 SOM-LV, DM3730/AM3703 Torpedo SOM, and DM3730/AM3703 Torpedo + Wireless SOM. Use of "DM3730/AM3703 SOM" suggests text that applies to all three platforms; information specific to one platform will call out the precise name.
- Use of "DM3730 Development Kit" suggests text that applies to both the DM3730 SOM-LV Development Kit and DM3730 Torpedo Development Kit; information specific to one development kit will call out the precise name.

2 Hardware Power Management Features

This section will discuss the DM3730/AM3703 SOM hardware power management features that are supported by the DM37x Linux BSP, including:

- Frequency scaling and Voltage domain voltage scaling
- Clock domain gating
- Power domain low-power state transitions

NOTE: Additional hardware features may be supported by the DM37x Linux BSP; discussion of those features will be added in future revisions of this document.

2.1 DM37x Processor Power Domains

The DM3730/AM3703 processor has eighteen power domains. The nine power domains below have four states that the kernel can manage based on system demand:

- **MPU:** Power domain of the microprocessor subsystem
- **CORE:** Power domain of the ARM® Cortex™-A8 processor
- **SGX:** Power domain of the 2D/3D graphics accelerator
- **DSS:** Power domain of the display subsystem
- **CAM:** Power domain of the camera controller subsystem
- **PER:** Power domain of the low-power uses peripherals, including: UART(3, 4), Watch Dog Timer 3, McBSP(2 – 4), GPIO(2 – 6), GPTIMER(2 – 9), L4-Per Interconnect
- **NEON:** Power domain of the NEON multimedia coprocessor
- **IVA2:** Power domain of the IVA2 audio video processor
- **USBHOST:** Power domain of the USB host

NOTE: See the *POWERSTATEST* field in the power domain register summaries in "Section 3.7.2" of the Texas Instruments (TI) [AM/DM37x Multimedia Device Technical Reference Manual \(TRM\)](http://www.ti.com/product/dm3730).¹

¹ <http://www.ti.com/product/dm3730>

Each power domain has logic and memory; the memory has a dedicated power rail. The logic of the CORE and PER power domains incorporates retention flip flops (RFFs) that allow them to reach a deeper retention state.

To see the above power domains managed by the kernel, enter the command below at the command line.

```
DM-37x# cat /debug/pm_debug/count | grep 'ON' | awk '{print $1}'
usbhost_pwrdom
sgx_pwrdom
per_pwrdom
dss_pwrdom
cam_pwrdom
core_pwrdom
neon_pwrdom
mpu_pwrdom
iva2_pwrdom
```

NOTE: In the command above, you can strip the *awk* command to see the power state transitions; this will be covered later.

The eight power domains below can be turned on or off by the PRCM, which resides in the WAKEUP domain. The WAKEUP power domain is the only domain that does not turn off.

- **EMU:** Power domain of the Emulation System
- **SMARTREFLEX:** Power domain of the SmartReflex interface to the PMIC
- **EFUSE:** Power domain of the EFUSE farm
- **DPLL1:** Power domain of the MPU DPLL
- **DPLL2:** Power domain of the IVA2 DPLL
- **DPLL3:** Power domain of the CORE DPLL
- **DPLL4:** Power domain of the Peripherals DPLL
- **DPLL5:** Power domain of the Peripherals DPLL2

2.1.1 Power Domain States

2.1.1.1 Off State – (0x0)

In the off state, Vdd to the domain is usually cut. The logic (DFF and RFF) is lost, except for what is stored in the scratchpad memory of the WKUP power domain, which is always on. Voltage to the power domain memory array may be on, lowered, or off.

2.1.1.2 Retention State – (0x1)

Retention is utilized for idle power management. It has lower entry and exit latencies, and will consume less power without any loss of context. In retention, logic is not operational and voltage to the domain can be lowered. There are two retention states: closed switch retention (CSWR) and open switch retention (OSWR).

CSWR is available in all power domains. The power domain retains all of its logic but the clocks are cut and the voltage is lowered to its retention state. The memory can be powered on or off and consequently retain or lose its data.

OSWR is only available to the Core and PER power domains. It allows the power domain to be powered off, but retain its context by supplying a lowered voltage to the RFF logic. This allows

the CORE and PER power domains to go into a deeper retention state without losing context. The memory can be powered on or off and consequently retain or lose its data.

2.1.1.3 Inactive State – (0x2)

The inactive state retains power to the domain's logic but the clocks are cut because the logic is inactive. The memory of the domain can be on, in retention, or off.

2.1.1.4 On State – (0x3)

In the on state, the domain is fully powered and operational. The memory of the domain can be on, in retention, or off.

2.2 DM37x Clock Domains

A clock domain is created by gating the clock source to a group of functional modules. Thus, when the modules are inactive, the dynamic power consumption of these modules can be reduced by cutting the clock sourced to them when logic is inactive.

3 Linux Power Management Features

The Linux operating system (OS) has many features that enable power management. The following is a subset of these features that are discussed in this application note:

- Tickless kernel
- CPUidle framework
- CPUfreq framework
- Suspend to RAM

In Table 3.1, the DM3730/AM3703 processor features are shown along with their respective implementations in the DM37x Linux BSP.

Table 3.1: DM3730/AM3703 Processor Feature Implementation

DM3730 / AM3703 Processor Feature	Linux BSP Feature
Dynamic Voltage and Frequency Scaling (DVFS)	CPUfreq
Power and Clock Domains	CPUidle
Power and Clock Domains	Suspend to RAM

3.1 Prerequisites

Before exploring each Linux power management feature documented herein, you will need to enable several options in the LTIB menu.

1. On your Linux build host PC, enter the commands below to enter the LTIB menu; be sure to replace `<LTIB_PATH>` with the path to your LTIB directory.

```
bash$ cd <LTIB_PATH>
bash$ ./ltib -c
```

You should see the following output:

```

my@my-desktop: ~/Linux_BSP_2.3-x/REL-ltlib-DM3730-2.3-2
File Edit View Terminal Help

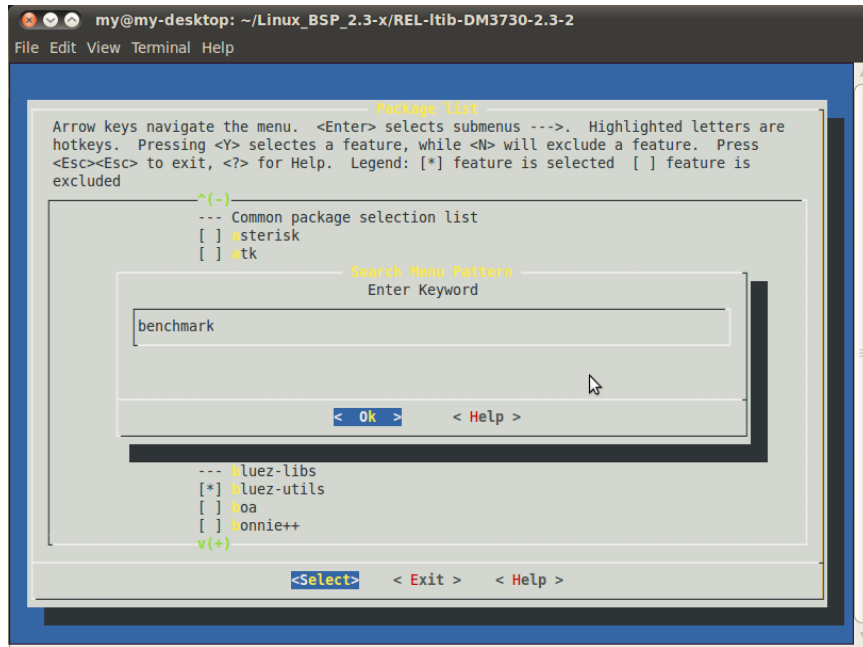
LTIB: Logic OMAP3530/3560/3730/03 reference boards
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted letters are
hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press
<Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [ ] feature is
excluded

[*] SCM global values --->
--- LTIB settings
System features --->
--- Choose the target C library type
Target C library type (glibc) --->
C library package (from toolchain only) --->
Toolchain component options --->
--- Toolchain selection.
Toolchain (CodeSourcery-2009q1-203 gcc-4.3.3 ARMv5te/glibc-2.5-nptl-3) --->
(-O2 -fsigned-char -mfloat-abi=softfp -mfpu=vfp) Enter any CFLAGS for gcc/g++
--- Choose your bootloader
u-boot (u-boot-2011.06 for Logic omap/dm37xx SOM-LV/Torpedo) --->
(omap3logic_config) U-boot configuraiton target
--- Choose your Kernel
kernel (Linux 3.0-omap-logic) --->
[ ] Always rebuild the kernel
[ ] Produce cscope index
(omap3logic_defconfig) kernel preconfig
v(+)

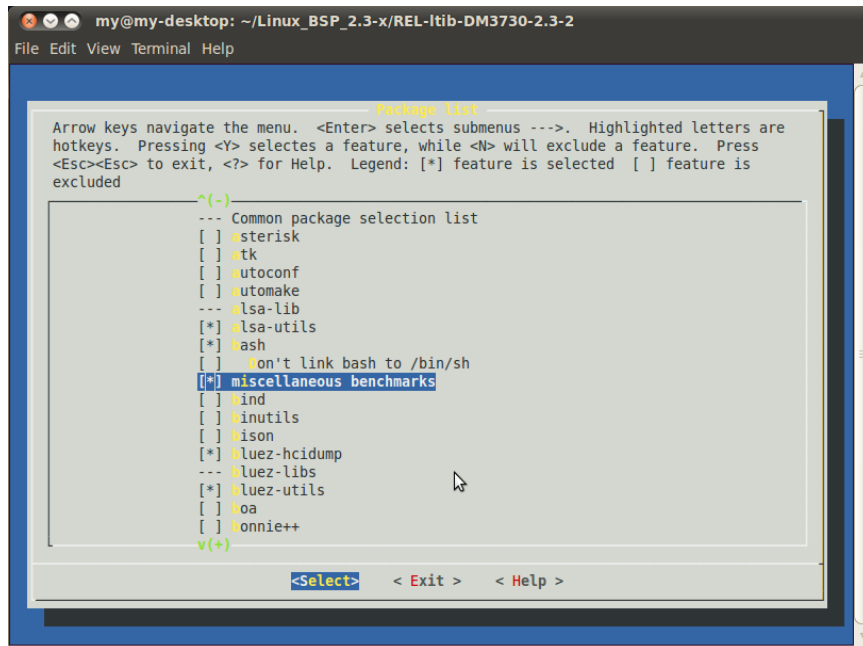
<Select> < Exit > < Help >

```

- Press the **S** key to bring up search in the LTIB menu.
- Enter *benchmark* in the search bar. With <Ok> highlighted, press **Enter**.



- Select the checkbox labeled "miscellaneous benchmarks" by highlighting it and pressing the **space bar**; this will include the miscellaneous benchmarks in the build.



- Exit and select "yes" when asked to save the configuration.
- After the kernel image builds, boot with it on the DM3730 Development Kit before continuing.

NOTE: If you are unsure how to boot the image that you just built, please refer to the [DM37x Linux BSP User Guide](#).²

3.2 Active Power Management – *CPUfreq*

The *CPUfreq* framework provides a mechanism to scale the CPU frequency. The framework has several writable files exported to sysfs that affect its policy. To explore these files, enter the command sequence below.

```
DM-37x# cd /sys/devices/system/cpu/cpu0/cpufreq
DM-37x# ls -l
-r--r--r-- 1 root root affected_cpus
-r----- 1 root root cpuinfo_cur_freq
-r--r--r-- 1 root root cpuinfo_max_freq
-r--r--r-- 1 root root cpuinfo_min_freq
-r--r--r-- 1 root root cpuinfo_transition_latency
-r--r--r-- 1 root root related_cpus
-r--r--r-- 1 root root scaling_available_frequencies
-r--r--r-- 1 root root scaling_available_governors
-r--r--r-- 1 root root scaling_cur_freq
-r--r--r-- 1 root root scaling_driver
-rw-r--r-- 1 root root scaling_governor
-rw-r--r-- 1 root root scaling_max_freq
-rw-r--r-- 1 root root scaling_min_freq
-rw-r--r-- 1 root root scaling_setspeed
drwxr-xr-x 2 root root stats
```

The framework uses governors to set the frequency scaling policy. The frequencies available to the governors are bounded by the values in *scaling_min_freq* and *scaling_max_freq*. The value in *scaling_governor* sets the current governor.

To see which governor is currently in use, enter the command below.

```
DM-37x# cat scaling_governor
ondemand
```

To see the available governors, enter the command below.

```
DM-37x# cat scaling_available_governors
conservative userspace ondemand performance
```

Each of these governors will set a unique frequency scaling policy; a description of each governor is provided below.

- *conservative*: Sets the CPU at the minimum-allowed clock speed
- *userspace*: Allows the CPU speed to be set by user space applications that modify the value in the *scaling_setspeed* file
- *ondemand*: Dynamically adjusts CPU speed based on a load calculation performed at a defined sampling period
- *performance*: Sets the CPU at the maximum-allowed clock speed

² <http://support.logipd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1392>

3.2.1 Tune *ondemand* Governor

The *ondemand* governor is a tunable governor. The tunable parameters allow it to achieve a better balance between performance and power. To see these tunable parameters, enter the commands below.

```
DM-37x# echo ondemand >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
DM-37x# cd /sys/devices/system/cpu/cpufreq/ondemand
DM-37x# ls
ignore_nice_load      sampling_down_factor  up_threshold
io_is_busy            sampling_rate
powersave_bias       sampling_rate_min
```

The value in each of these files affects the *ondemand* governor's frequency scaling policy.

3.2.1.1 *up_threshold*

The *up_threshold* value is used to determine when the frequency should be scaled up. For example, if the value is set at 80, the CPU load calculation must be above 80% before the frequency can be scaled up. To view the default value, enter the command below.

```
DM-37x# cat up_threshold
95
```

3.2.1.2 *sampling_rate*

The *sampling_rate* value is the number of microseconds between successive CPU load calculations. To view the current setting, enter the command below.

```
DM-37x# cat sampling_rate
300000
```

The current period from the output is 300000 μ s. Since each CPU load calculation has CPU overhead associated with it, increasing this value will decrease the CPU overhead associated with running the *ondemand* governor. This can improve performance if the CPU is running at full load. This value should always be based on the transition latency value in *cpuinfo_transition_latency*. Be aware that the transition latency value is in nanoseconds and the sampling rate is in microseconds.

For example, if you wanted to set the sampling rate at 875 times the transition latency, you would use the following calculation:

$$sampling_{rate} = \frac{transition_{latency} * 875}{1000}$$

3.2.1.3 *sampling_down_factor*

The *sampling_down_factor* value acts as a down sampling rate. For example, if *sampling_rate* = 300000 and *sampling_down_factor* = 2, the effective sampling rate will be 600000. This improves performance when the processor is at full load by avoiding the overhead related to load evaluation. To see the current value, enter the command below.

```
DM-37x# cat sampling_down_factor
1
```

3.2.1.4 *powersave_bias*

The *powersave_bias* value defines the amount of bias towards power saving when a frequency is selected by the *ondemand* governor. The value represents one thousandth of a percentage point reduction in the targeted frequency. To see the current value, enter the command below.

```
DM-37x# cat powersave_bias
0
```

If a reduction in frequency does not match one of the available frequencies, the governor will switch between two available frequencies in an attempt to reach an average frequency defined by the governor and the *powersave_bias* value. For example, if *powersave_bias* = 100 and the *ondemand* governor targets 300 MHz, the actual frequency run will be 270 MHz. If 270 MHz is not one of the available frequencies, some combination of the available frequencies will be used in time to achieve an average frequency of 270 MHz.

3.2.1.5 *ignore_nice_load*

The *ignore_nice_load* parameter takes a value of 0 or 1. When it is set, the processes run with a nice value are ignored in the load calculation. This is useful if you are running a process that you don't want to cause a CPU frequency increase. To see the current setting, enter the command below.

```
DM-37x# cat ignore_nice_load
0
```

3.2.2 Tune *conservative* Governor

Like the *ondemand* governor, the *conservative* governor is tunable. The *conservative* governor differs from the *ondemand* governor in that it scales the CPU speed more smoothly. The behavior of the *conservative* governor is better for battery-powered systems. To see the *conservative* governor's parameters, enter the commands below.

```
DM-37x# echo conservative >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
DM-37x# cd /sys/devices/system/cpu/cpufreq/conservative/
DM-37x# ls
down_threshold      sampling_down_factor  up_threshold
freq_step           sampling_rate
ignore_nice_load    sampling_rate_min
```

3.2.2.1 *down_threshold*

The *down_threshold* value has the same behavior as the *up_threshold* value discussed in Section 3.2.1.1 except that it is in the reverse direction. For example, if the *down_threshold* value is set to 20%, the CPU load calculation needs to be below 20% before the CPU frequency can be scaled down. To see the current value, enter the command below.

```
DM-37x# cat down_threshold
20
```

3.2.2.2 *freq_step*

The *freq_step* value affects the amount of percentage point change by which the governor adjusts the frequency. A lower value will cause smoother transitions. If this value is set to 100, the *conservative* governor behaves similar to the *ondemand* governor. To see the current value, enter the command below.

```
DM-37x# cat freq_step
5
```

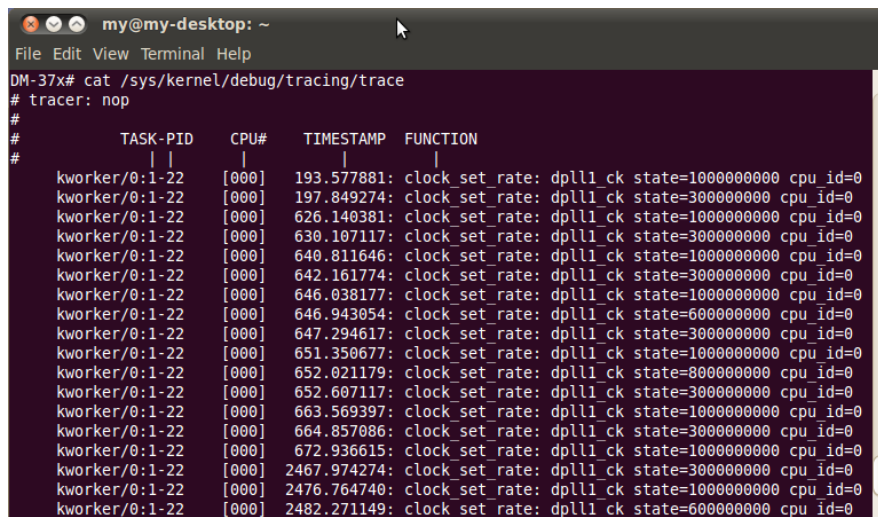
3.2.3 Debug *CPUfreq* – Trace Clock Rate Transitions

Now, let's examine a more interesting trace, *clock_set_rate*. At the shell prompt, enter the commands below.

```
DM-37x# echo 1
>/sys/kernel/debug/tracing/events/power/clock_set_rate/enable
DM-37x# /usr/bin/whetstones 10000

Loops: 10000, Iterations: 1, Duration: 5.58677 sec.
C Converted Double Precision Whetstones: 179.0 MIPS
DM-37x# cat /sys/kernel/debug/tracing/trace
```

You should see the output below, in which you will see several pieces of information. We are interested in *state*, which indicates the frequency of the processor.



```
my@my-desktop: ~
File Edit View Terminal Help
DM-37x# cat /sys/kernel/debug/tracing/trace
# tracer: nop
#
#          TASK-PID    CPU#    TIMESTAMP    FUNCTION
#          |   |   |   |   |
kworker/0:1-22 [000] 193.577881: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 197.849274: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 626.140381: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 630.107117: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 640.811646: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 642.161774: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 646.038177: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 646.943054: clock_set_rate: dp111_ck state=6000000000 cpu_id=0
kworker/0:1-22 [000] 647.294617: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 651.350677: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 652.021179: clock_set_rate: dp111_ck state=8000000000 cpu_id=0
kworker/0:1-22 [000] 652.607117: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 663.569397: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 664.857086: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 672.936615: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 2467.974274: clock_set_rate: dp111_ck state=3000000000 cpu_id=0
kworker/0:1-22 [000] 2476.764740: clock_set_rate: dp111_ck state=1000000000 cpu_id=0
kworker/0:1-22 [000] 2482.271149: clock_set_rate: dp111_ck state=6000000000 cpu_id=0
```

3.3 Idle Power Management - *CPUI* Framework

To minimize unnecessary power consumption, it is important for the OS to know what to do when there is no work to be done. When the system is idle, there is an opportunity for the OS to save power; the *CPUI* framework in the DM37x Linux BSP minimizes power consumption when these opportunities occur. The *CPUI* framework defines C-states that the system can be in while it is idle. Each C-state is associated with a certain level of power consumption and wake-up latency. Good management of both is critical to a system that is to be responsive to user demands while consuming a minimal amount of power when the system is idle. Choosing which C-state to enter is a critical task that is performed by a governor. The *CPUI* framework is equipped with two different governors, *ladder* and *menu*.

In this section we will explain each aspect of idle power management and how to implement different idle power management schemes with the DM37x Linux BSP. Figure 3.1 shows the state of each of the nine power domains while residing in different idle states. Power savings in the idle states primarily comes from power down unused power domains.

	SGX	PER	DSS	CAM	CORE	NEON	MPU	IVA2
C1	OFF	ON	ON	OFF	ON	ON	ON	OFF
C2	OFF	OFF / RET	ON	OFF	ON	INA	INA	OFF
C3	OFF	OFF / RET	ON	OFF	ON	RET	RET	OFF
C4	OFF	OFF / RET	ON	OFF	ON	OFF	OFF	OFF
C5	OFF	OFF	ON	OFF	ON	RET	RET	OFF
C6	OFF	OFF	ON	OFF	ON	OFF	OFF	OFF
C7	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF

Figure 3.1: Observed Power Domain States During C-State Residency

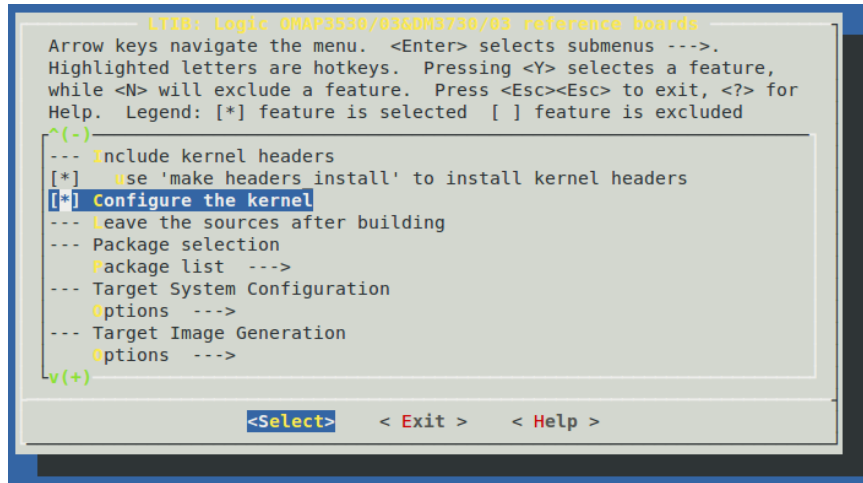
3.3.1 Prerequisites

Before beginning, we need to enable the *CPUI* framework and rebuild the images; this can be done in the kernel configuration menu.

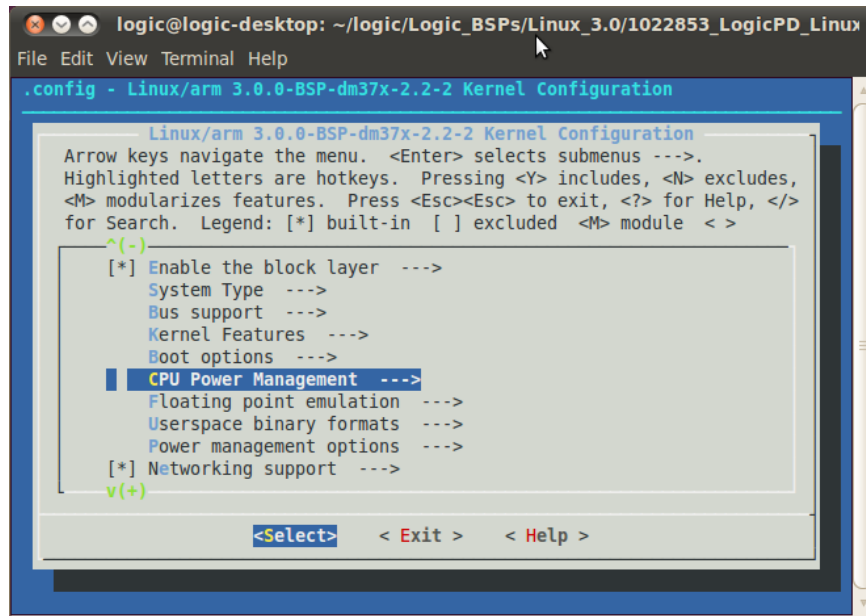
1. Open the LTIB configuration menu; be sure to replace `<LTIB_PATH>` with the path to your LTIB directory.

```
bash$ cd <LTIB_PATH>
bash$ ./ltib -c
```

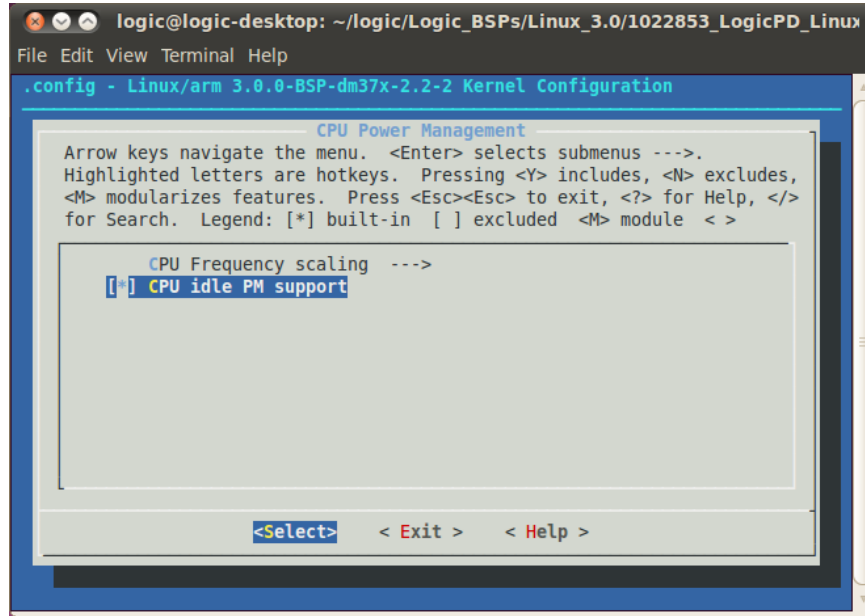
- To open the kernel configuration menu, select "Configure the kernel." Exit and save the LTIB configuration.



- From the root of the kernel configuration menu, select "CPU Power Management" and press **Enter**.

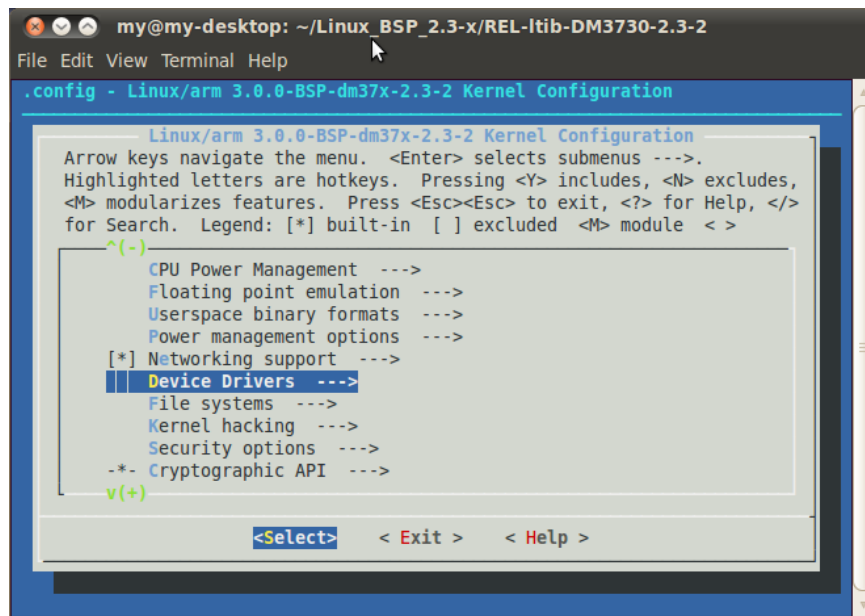


- Highlight "CPU idle PM support" and press the **space bar** to place an asterisk between the brackets.

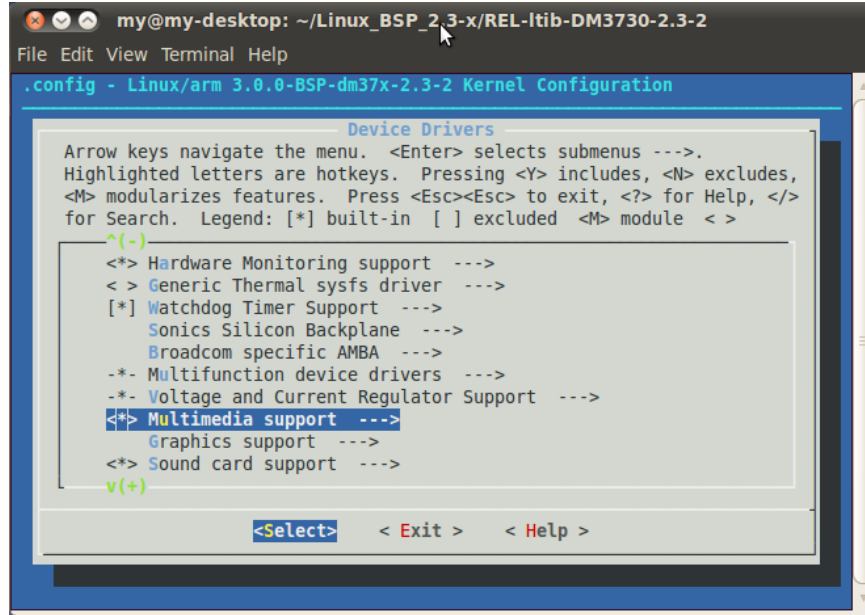


- Now we need to keep several drivers from automatically loading that prevent the idle power management from going into higher C-states. The idle power management looks at whether the CAM power domain is active before going into higher C-states because the CAM power domain does not have wake-up capability. The drivers that we will prevent from loading keep the CAM power domain active all the time, even when they are not in use. This serves as a good example of how drivers can affect the power management of an entire system.

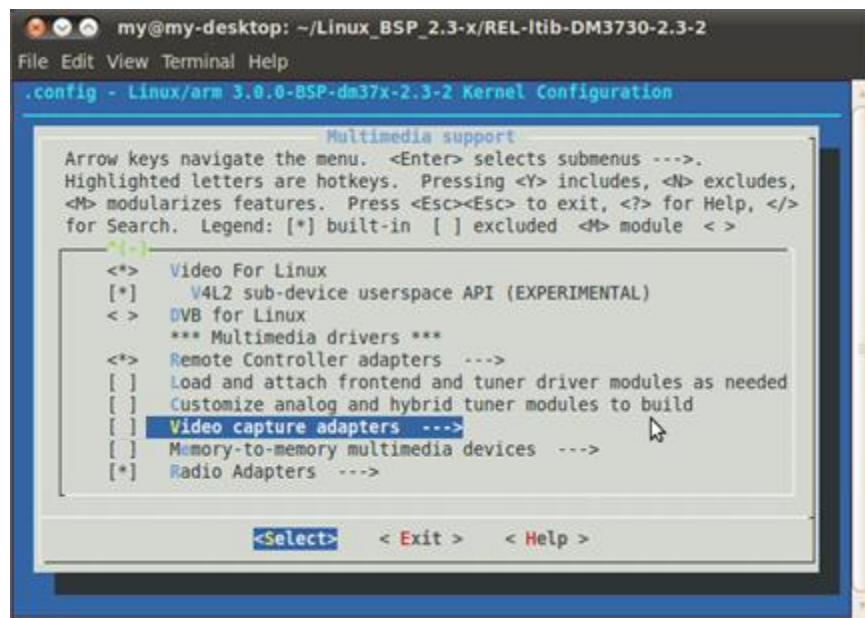
From the kernel configuration menu root, highlight "Device Drivers" and press **Enter**.



6. Highlight “Multimedia support” and press **Enter**.



7. Highlight “Video capture adapter --->” and use the space bar to remove the asterisk between the brackets.

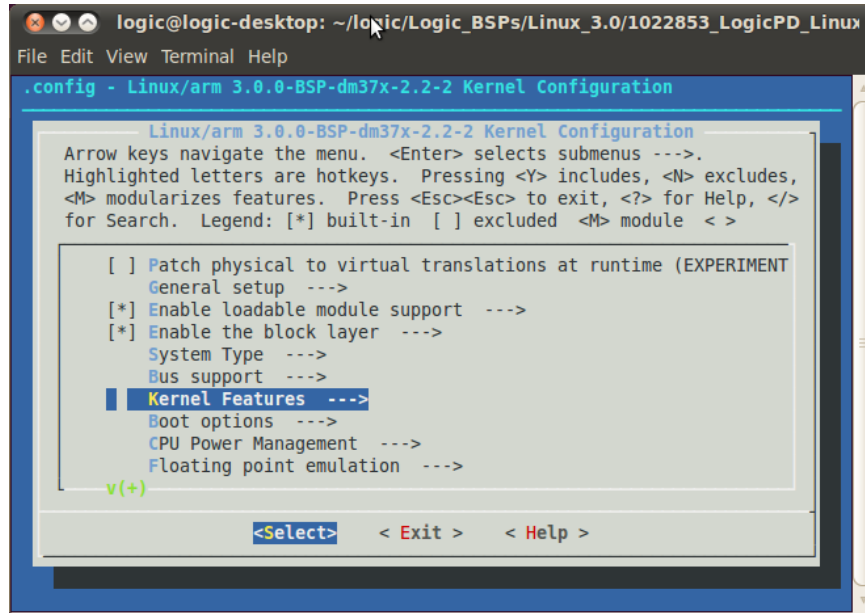


Continue to the next section to consider enabling or disabling the tickless kernel.

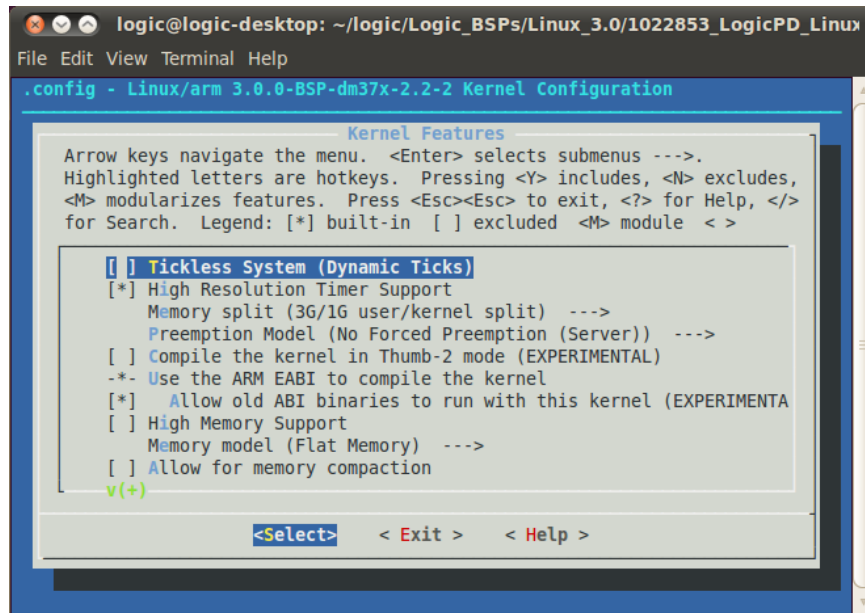
3.3.1.1 Enable or Disable Tickless Kernel

By enabling the *CPUIidle* PM support, the *ladder* and *menu* governor drivers will automatically be included in the build. By default the kernel is set up as tickless, which also dictates that the *menu* governor be used. To use the *ladder* governor, configure the kernel to use a tick by following the steps below.

1. From the root of the kernel configuration menu, highlight "Kernel Features" and press **Enter**.



2. Highlight "Tickless System (Dynamic Ticks)" and use the **space bar** to remove the asterisk between the brackets.



3. Exit the kernel configuration menu and save the configuration. Let the build complete and then boot the board with the newly built images.

NOTE: If you are unsure how to boot the image you just built, please refer to the [DM37x Linux BSP User Guide](#)

3.3.2 C-states

As mentioned before, C-states are states or modes that the system can be in while there is no useful work to be done. The C-states in the DM37x Linux BSP are defined in the `/arch/arm/mach-omap2/cpuidle34xx.c` file.

In this source file you will find parameters that you can adjust to achieve a better balance between system responsiveness and idle power consumption. Below is an example from `cpuidle34xx.c` where these parameters have been defined.

```
1. static struct cpuidle_params cpuidle_params_table[] = {
2. /* C1 */
3. {2 + 2, 5, 1},
4. /* C2 */
5. {10 + 10, 30, 1},
6. /* C3 */
7. {50 + 50, 300, 1},
8. /* C4 */
9. {1500 + 1800, 4000, 1},
10. /* C5 */
11. {2500 + 7500, 12000, 1},
12. /* C6 */
13. {3000 + 8500, 15000, 1},
14. /* C7 */
15. {10000 + 30000, 300000, 1},
16. };
```

The `struct cpuidle_params` has three members that are important to note. The first defines exit latency, which is the sum of the time it takes to go into the C-state and come out of the C-state. The second member is the target residency, which defines the time the system should remain in a C-state for it to be worth it considering the cost of the exit latency. The third tells the *idle* governor if the state can be used or not.

3.3.3 Kernel Tick Effect on Idle Power Management

OSs need some way of keeping time. This can be done using a periodic tick or a dynamic tick that indicates to the kernel that maintenance needs to be done, such as process accounting. Each have an effect on idle power management and should be carefully considered when developing a Linux-based system.

3.3.3.1 Periodic Tick and Dynamic Tick

A kernel that uses the periodic tick is applying the traditional approach of time keeping with periodic timers. This means that while idle, the system will be woken up at a regular frequency even if there are no jobs to work on. This results in a system that can only stay in a C-state for a fixed period.

Below are statistics, exported through sysfs, that show the number of times a C-state has been used and the total amount of time that has transpired while in that state.

```
DM-37x# cd /sys/devices/system/cpu/cpu0/cpuidle
DM-37x# cat state*/time
775382113
...
348549104
DM-37x# cat state*/usage
```

```
101702
...
46294
```

In the above output, you can see the periodic tick statistics for both C1 (high power/low latency) and C7 (low power/high latency). Below the assumption is made that in the low-power state, the system is idle often and the time in each instance of the state would be much greater than in the high-power state C-1. Now, let's make a comparison:

$$Avg(\Delta T) = (Total\ State\ Time) \div (Number\ of\ State\ Uses)$$

$$Avg(\Delta C1) = \frac{775382113}{101702} \approx 7624us$$

$$Avg(\Delta C7) = \frac{348549104}{46294} \approx 7529us$$

Here is the same calculation done for the dynamic tick.

```
DM-37x# cat state*/time
141459331
...
432642338
DM-37x# cat state*/usage
3876
...
544
```

$$Avg(\Delta C1) = \frac{141459331}{3876} \approx 36496us$$

$$Avg(\Delta C7) = \frac{432642338}{544} \approx 795298us$$

Based on the above calculations, we can see that two systems with similar activity utilized the available C-states by differing amounts. The periodic tick calculation shows that the system was continuously woken up by the periodic system tick.

3.3.4 Governors

Governors are a critical element to the Linux idle power management framework. They are the algorithms that decide which state should be entered based on current and previous idle performance. The type of governor used will be determined by whether the kernel has a tick or is tickless (dynamic).

3.3.4.1 Ladder

The *ladder* governor is used with periodic tick and is a simple governor that puts the system into the first C-state when the Linux kernel detects that the system is idle. The *ladder* governor will only progress to the next-highest C-state if the current C-state is occupied for a minimum time specified. This time is specified based on the exit latency time that the governor views as a cost. The *ladder* governor relies on the periodic kernel tick and is enabled when a periodic tick is enabled. If you would like to read more about the *ladder* governor and how it decides which C-state to enter, refer to the *drivers/cpuidle/governors/ladder.c* file.

3.3.4.2 Menu

The *menu* governor is used with dynamic tick and is more sophisticated than the *ladder* governor. It also works well in tickless kernels. The *menu* governor bases its decision to enter a C-state on the following three factors:

1. Energy breakeven point
2. Performance impact
3. Latency tolerance

If you would like to read more about the *menu* governor and how it decides which C-state to enter, refer to the *drivers/cpuidle/governors/menu.c* file; it contains a thorough description of the algorithm.

3.3.5 Access CPUIidle Framework from Linux Userspace

The following three directories expose the *CPUIidle* framework to the userspace through the sysfs file system:

```
/sys/devices/system/cpu/cpuidle
/sys/devices/system/cpu/cpu0/cpuidle
/sys/kernel/debug/tracing/events/power/cpu_idle
```

3.3.5.1 CPUIidle System-generic Information

Information that is generic to the system can be accessed in userspace through the */sys/devices/system/cpu/cpuidle* directory. To explore what information is contained in this directory, enter the commands below at the DM3730's Linux shell prompt.

```
DM-37x# cd /sys/devices/system/cpu/cpuidle
DM-37x# ls -l
-r--r--r-- 1 root root 4096 Feb 11 20:19 current_driver
-r--r--r-- 1 root root 4096 Feb 11 20:19
current_governor_ro
DM-37x# cat current_driver
omap3_idle
DM-37x# cat current_governor_ro
menu
DM-37x#
```

From this we have learned that the *CPUIidle* driver is *omap3_idle* and the current governor is the *menu* governor.

3.3.5.2 CPUIidle CPU-specific Information

The *CPUIidle* framework also exports information and interfaces to the userspace through the sysfs file system that details information specific to each CPU in a system. The */sys/devices/system/cpu/cpuX/cpuidle* directory contains CPU X's specific idle information.

In the case of the DM3730/AM3703 processor, X can be replaced with 0 (*/sys/devices/system/cpu/cpu0/cpuidle*) since there is only one CPU managed by the idle framework.

3.3.5.3 Debugging *CPUI* Framework

From userspace, it is possible to see how each power domain transitions into and out of its available states.

1. Before enabling idle power management, enter the command below to get an idea about the initial state of the system.

```
DM-37x# cat /debug/pm_debug/count | grep ON
usbhost_pwrdm (OFF), OFF:1, RET:1, INA:0, ON:1, RET-LOGIC-OFF:0, RET-
MEMBANK1-OFF:0
sgx_pwrdm (OFF), OFF:1, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
per_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
dss_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
cam_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
core_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0, RET-MEMBANK2-OFF:0
neon_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0
mpu_pwrdm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
iva2_pwrdm (OFF), OFF:1, RET:1, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0, RET-MEMBANK2-OFF:0, RET-MEMBANK3-OFF:0, RET-MEMBANK4-OFF:0
```

From the above output, you can see that all of the nine power domains except USBHOST and IVA2 have only been in the ON state.

2. Enter the commands below to enable idle power management.

```
DM-37x# echo 1 > /sys/kernel/debug/pm_debug/sleep_while_idle
DM-37x# echo 5 > /sys/devices/platform/omap/omap_uart.0/sleep_timeout
DM-37x# echo 5 > /sys/devices/platform/omap/omap_uart.1/sleep_timeout
DM-37x# echo 5 > /sys/devices/platform/omap/omap_uart.2/sleep_timeout
```

3. Now that the system is managing its power during idle times, we can view the usage of each idle state from sysfs by entering the commands below.

```
DM-37x# cd /sys/devices/system/cpu/cpu0/cpuidle
DM-37x# ls
state0 state1 state2 state3 state4 state5 state6
DM-37x# cat state*/usage
2042
0
4
21
0
19
0
```

3.4 Suspend

Suspend to RAM (STR) is a deeper sleep state than any of the C-states. When the system goes into suspend, all devices on the board are typically powered off except for those needed to refresh RAM and those needed to cause a wake-up transition.

3.4.1 Debug Suspend from Linux Userspace

The DM37x Linux BSP provides several useful tests in sysfs that will help you test the suspend and wake-up features of your device drivers.

3.4.1.1 *pm_test*

The Linux power management subsystem provides the facility *pm_test*, which allows the suspend core to run in a test mode. The following five tests are available:

1. Freezer
2. Devices
3. Platform
4. Processor
5. Core

Each test is progressively more extensive than the previous test. The freezer test checks the suspend of processes while the core test checks the entire chain of suspend activities. We will explore the use of each test.

1. *pm_test [freezer]*

The freezer option will freeze all processes, wait five seconds, and then thaw all processes.

```
DM-37x# echo freezer > /sys/power/pm_test
DM-37x# echo mem > /sys/power/state
[ 2026.082244] PM: Syncing filesystems ... done.
[ 2026.096496] PM: Preparing system for mem sleep
[ 2026.105163] mmc0: card 1234 removed
[ 2026.253448] Freezing user space processes ... (elapsed 0.02 seconds)
done.
[ 2026.283569] Freezing remaining freezable tasks ... (elapsed 0.02
seconds) done.
[ 2026.314788] suspend debug: Waiting for 5 seconds.
[ 2031.309204] PM: Finishing wakeup.
[ 2031.312744] Restarting tasks ... done.
[ 2031.786254] mmc0: host does not support reading read-only switch.
assuming write-enable.
[ 2031.796630] mmc0: new SD card at address 1234
[ 2031.816497] mmcblk0: mmc0:1234 SA02G 1.85 GiB
[ 2031.862945] mmcblk0: p1

DM-37x# DM-37x# echo none > /sys/power/pm_test
```

2. *pm_test [devices]*

The device option is useful when you want to test the suspend and resume capabilities of device drivers in the system. This test will freeze all processes and suspend all devices, then resume all devices and thaw all processes.

Below is a test, where the UART driver's suspend method was modified to always return an error code of -123. From the output, you can see what to expect if one of your devices fails to suspend.

```
DM-37x# echo devices > /sys/power/pm_test
DM-37x# echo mem > /sys/power/state
[ 103.914367] PM: Syncing filesystems ... done.
[ 103.932006] PM: Preparing system for mem sleep
[ 103.956176] mmc0: card 1234 removed
[ 104.049407] omap_device: omap_i2c.1: new worst case activate latency
0: 122070
[ 104.171691] Freezing user space processes ... (elapsed 0.01 seconds)
done.
[ 104.197601] Freezing remaining freezable tasks ... (elapsed 0.02
seconds) done.
[ 104.228820] PM: Entering mem sleep
[ 104.232635] omap_device: smartreflex.0: new worst case deactivate
latency 0: 122070
[ 104.259307] ISP1763 resuming
[ 104.277893] dm3730_torpedo_bl_set_intensity: level 0 (0% on)
[ 104.283996] dm3730_torpedo_bl_set_intensity: turn off GPIO_154 as
backlight!
[ 104.382843] ISP1763 suspended
[ 104.388763] pm_op(): platform_pm_suspend+0x0/0x64 returns -123
[ 104.395050] PM: Device omap_uart.2 failed to suspend: error -123
[ 104.401428] PM: Some devices failed to suspend
[ 104.409576] ISP1763 resuming
[ 104.420349] dm3730_torpedo_bl_set_intensity: level 178 (69% on)
[ 104.426696] dm3730_torpedo_bl_set_intensity: turn on GPIO_154 as
backlight!
[ 104.501556] PM: resume of devices complete after 95.367 msecs
[ 104.516021] PM: Finishing wakeup.
[ 104.519531] Restarting tasks ...
[ 104.524444] omap_device: musb-omap2430.-1: new worst case activate
latency 0: 91552
[ 104.555114] done.
[ 104.587402] hub 2-0:1.0: activate --> -22
[ 104.942443] mmc0: host does not support reading read-only switch.
assuming write-enable.
[ 104.960571] mmc0: new SD card at address 1234
[ 104.979095] mmcblk0: mmc0:1234 SA02G 1.85 GiB
[ 105.007263] mmcblk0: p1
[ 109.077697] ISP1763 suspended

DM-37x# echo none > /sys/power/pm_test
```

3. *pm_test [platform]*

In addition to testing the freezing of processes and the suspending of devices, the platform option will test the platform global control methods that are only available through the ACPI standard. This is not implemented in the DM37x Linux BSP.

4. *pm_test [processor]*

The processor option is an extension of the devices test. It will freeze all processes, suspend all devices, and test the disabling of all non-boot CPUs. Since there are no non-boot CPUs on the system, this test will provide no additional information.

5. *pm_test* [core]

In addition to freezing processes and suspending devices, the core option will test the suspending of platform/system devices.

```
DM-37x# echo core > /sys/power/pm_test
DM-37x# echo mem > /sys/power/state
[ 2583.617095] PM: Syncing filesystems ... done.
[ 2583.627990] PM: Preparing system for mem sleep
[ 2583.636474] mmc0: card 1234 removed
[ 2583.784667] Freezing user space processes ... (elapsed 0.02 seconds)
done.
[ 2583.814819] Freezing remaining freezable tasks ... (elapsed 0.02
seconds) done.
[ 2583.846069] PM: Entering mem sleep
[ 2583.865142] ISP1763 resuming
[ 2583.976196] ISP1763 suspended
[ 2583.982879] PM: suspend of devices complete after 120.081 msecs
[ 2583.994812] PM: late suspend of devices complete after 5.615 msecs
[ 2584.002044] suspend debug: Waiting for 5 seconds.
[ 2588.965332] PM: early resume of devices complete after 3.021 msecs
[ 2588.977966] ISP1763 resuming
[ 2589.071166] PM: resume of devices complete after 96.862 msecs
[ 2589.085388] PM: Finishing wakeup.
[ 2589.088867] Restarting tasks ... done.
[ 2589.140197] hub 2-0:1.0: activate --> -22
[ 2589.520233] mmc0: host does not support reading read-only switch.
assuming write-enable.
[ 2589.538452] mmc0: new SD card at address 1234
[ 2589.558166] mmcblk0: mmc0:1234 SA02G 1.85 GiB
[ 2589.597106] mmcblk0: p1
[ 2593.116333] ISP1763 suspended

DM-37x# echo none > /sys/power/pm_test
```

3.4.1.2 Examine *power_domain_target* During Suspend Cycle

1. Let's view *power_domain_target* after the system is put into suspend by entering the command below at the shell prompt.

```
DM-37x# echo 1 >
/sys/kernel/debug/tracing/events/power/power_domain_target/enable
DM-37x# echo mem > /sys/power/state
...
[ 644.972290] PM: late suspend of devices complete after 8.728 msecs
```

Make note of the last time stamp printed to the terminal before the system went into suspend. In this example, it is 644.972290.

2. Press **Enter** in the terminal to wake the system from suspend and enter the commands below.

```
DM37x# echo 0 >
/sys/kernel/debug/tracing/events/power/power_domain_target/enable
DM-37x# cat /sys/kernel/debug/tracing/trace > /trace.txt
DM-37x# less /trace.txt
```

3. Using less, use the search function to find the time stamp noted above. In this example, the target is "644.9".

```
state=0 sh-705 [000] 644.979370: power_domain_target: mpu_pwrldm
state=0 cpu
state=0 sh-705 [000] 644.979401: power_domain_target: neon_pwrldm
state=0 cpu
state=0 sh-705 [000] 644.979431: power_domain_target: core_pwrldm
state=0 cpu
state=0 sh-705 [000] 644.979523: power_domain_target: neon_pwrldm
state=0 cpu
state=2147 sh-705 [000] 717.334991: power_domain_target: dpll1_pwrldm
state=2147 sh-705 [000] 717.335022: power_domain_target: emu_pwrldm
state=2147 sh-705 [000] 717.335022: power_domain_target: per_pwrldm
state=2147 sh-705 [000] 717.335022: power_domain_target: dss_pwrldm
state=2147 sh-705 [000] 717.335052: power_domain_target: neon_pwrldm
state=2147 sh-705 [000] 717.335052: power_domain_target: mpu_pwrldm
state=2147 sh-705 [000] 717.335083: power_domain_target: mpu_pwrldm
state=3 cpu
state=3 sh-705 [000] 717.335114: power_domain_target: neon_pwrldm
state=3 cpu
state=3 sh-705 [000] 717.341492: power_domain_target: core_pwrldm
state=3 cpu
```

You should be able to find the trace output where the system hit suspend. Notice how the MPU, NEON, and CORE power domains hit the off state (0x0) and then eventually return to the on state (0x3).

4. The debug file system also holds the `/debug/pm_debug/count` file that we can use to see which power domains are hitting off when the system is suspended. With a rebooted system, enter the command below to get the initial state of the system.

```
DM-37x# cat /debug/pm_debug/count | grep ON
usbhost_pwrldm (OFF), OFF:1, RET:1, INA:0, ON:1, RET-LOGIC-OFF:0, RET-
MEMBANK1-OFF:0
sgx_pwrldm (OFF), OFF:1, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
per_pwrldm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
dss_pwrldm (ON), OFF:0, RET:0, INA:0, ON:1, RET-LOGIC-OFF:0, RET-MEMBANK1-
OFF:0
```

```
cam_pwrldm (OFF),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
core_pwrldm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0,RET-MEMBANK2-OFF:0
neon_pwrldm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0
mpu_pwrldm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
iva2_pwrldm (OFF),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0,RET-MEMBANK4-OFF:0
```

5. Cycle the system in and out of suspend and reread the count file for comparison by entering the command below.

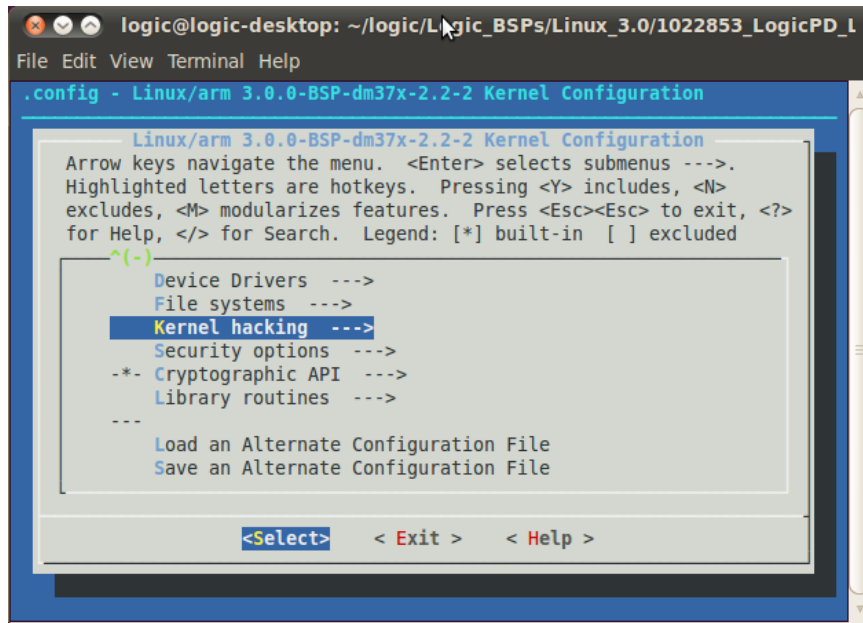
```
DM-37x# echo mem > /sys/power/state
...
DM-37x# cat /debug/pm_debug/count | grep ON
usbhost_pwrldm (OFF),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-
MEMBANK1-OFF:0
sgx_pwrldm (OFF),OFF:1,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
per_pwrldm (ON),OFF:1,RET:0,INA:0,ON:2,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
dss_pwrldm (ON),OFF:1,RET:0,INA:0,ON:2,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
cam_pwrldm (OFF),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
core_pwrldm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0,RET-MEMBANK2-OFF:0
neon_pwrldm (ON),OFF:1,RET:0,INA:0,ON:2,RET-LOGIC-OFF:0
mpu_pwrldm (ON),OFF:1,RET:0,INA:0,ON:2,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0
iva2_pwrldm (OFF),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-
OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0,RET-MEMBANK4-OFF:0
```

The output above shows that most of the nine power domains hit the off state while in suspend.

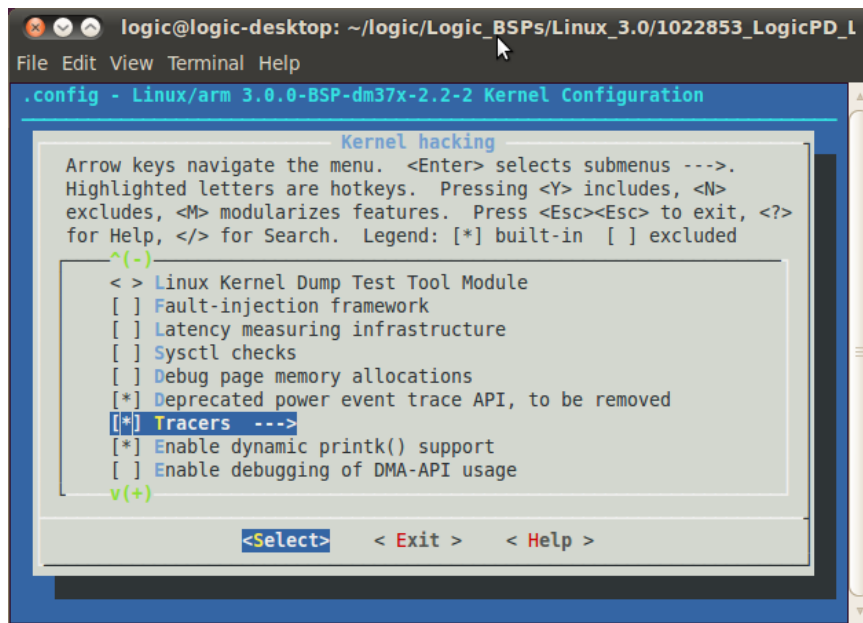
3.4.1.3 Use *ftrace* to Trace Functions

The *ftrace* tool is useful to trace function calls and see what is going on inside a running kernel. For example, for power management debugging, one could track calls to the function *omap_sram_idle* to see when the system is transitioning into idle states.

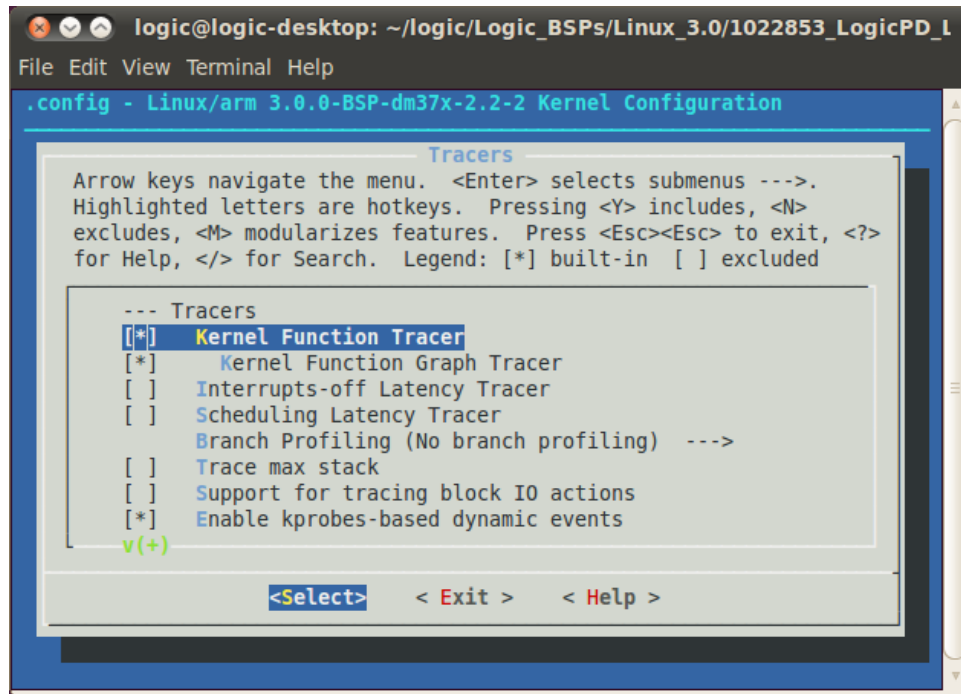
1. Before you can use *ftrace*, it must be added to the kernel build. To do this, from the kernel configuration menu root, highlight "Kernel hacking" and press **Enter**.



2. Highlight "Tracers" and press **Enter**.



- Highlight "Kernel Function Tracer" and use the **space bar** to place an asterisk between the brackets.



After building and booting the kernel, you will have the `/sys/kernel/debug/tracing` directory with `fttrace` as an available tracer.

- After booting, enter the commands below to see the available traces.

```
DM-37x# cd /sys/kernel/debug/tracing/
DM-37x# cat available_tracers
function_graph function nop
DM-37x#
```

- Enable the function tracer.

```
DM-37x# echo function > current_tracer
DM-37x#
```

- View a function trace.

```
DM-37x# cat trace
# tracer: function
#
# TASK-PID      CPU#    TIMESTAMP  FUNCTION
#   | |         |      |         |
<idle>-0       [000]    803.796906: pwrldm_for_each_clkdm <-
omap3_enter_idle
<idle>-0       [000]    803.796906: __cpuidle_allow_idle <-
pwrldm_for_each_clkdm
<idle>-0       [000]    803.796906: clkdm_allow_idle <-
__cpuidle_allow_idle
```

```

<idle>-0      [000]    803.796906: omap3_clkdm_allow_idle <-
clkdm_allow_idle
<idle>-0      [000]    803.796936: omap3xxx_cm_clkdm_enable_hwsup <-
omap3_clkdm_allow_idle
<idle>-0      [000]    803.796936: _write_clktrctrl <-
omap3xxx_cm_clkdm_enable_hwsup
<idle>-0      [000]    803.796936: pwrldm_clkdm_state_switch <-
clkdm_allow_idle

```

From the output you can see several function calls to *omap3_enter_idle*. This function is called from the *CPUidle* framework to program the device to the specified target state selected by the governor.

3.4.1.4 Use Event Tracing to Trace Suspend Events

One of the powerful debugging features built into Linux is the ability to trace events.

1. First, let's see what files are in the */debug/tracing* directory.

```
DM-37x# ls /debug/tracing
```

You should see the output below.

```

README          kprobe_events   set_event
tracing_cpumask  kprobe_profile  trace
available_events tracing_enabled
available_tracers options          trace_clock     tracing_on
buffer_size_kb   per_cpu         trace_marker    tracing_thresh
current_tracer   printk_formats  trace_options
events          saved_cmdlines  trace_pipe

```

2. View the events that can be traced.

```
DM-37x# cat /debug/tracing/available_events
```

A long list of event will be printed in the terminal window; the events that are of interest for power management debugging are:

```

power:cpu_idle
power:cpu_frequency
power:machine_suspend
power:power_start
power:power_frequency
power:power_end
power:clock_enable
power:clock_disable
power:clock_set_rate
power:power_domain_target

```

- Each event can be enabled/disabled with the commands below respectively; be sure to replace `<EVENT>` with one of the available events in Step 2 above.

```
DM-37x# echo 1 > /sys/kernel/debug/tracing/events/power/<EVENT>/enable
DM-37x# echo 0 > /sys/kernel/debug/tracing/events/power/<EVENT>/enable
```

NOTE: If you do not see the `/sys/kernel/debug` directory, you will need to enable *debugfs* in the kernel configuration menu. This option is found under the *Kernel Hacking* heading.

- Alternatively, all available events can be enabled/disabled with the commands below.

```
DM-37x# echo 1 > /sys/kernel/debug/tracing/events/power/enable
DM-37x# echo 0 > /sys/kernel/debug/tracing/events/power/enable
```

- Now, we will use the *machine_suspend* event to see what output is created when we enter suspend. Enable the event tracing for the *machine_suspend* event by entering the command below.

```
DM-37x# echo 1 > /sys/kernel/debug/tracing/events/power/machine_suspend/enable
```

- Put the system in suspend mode.

```
DM-37x# echo mem > /sys/power/state
```

- Press **Enter** after the system has suspended and then enter the command below to see the output of the trace.

```
DM-37x# cat trace
# tracer: nop
#
#          TASK-PID      CPU#      TIMESTAMP      FUNCTION
#          | |          |          |          |
#          sh-713      [000]      3311.554352: machine_suspend: state=3
#          sh-713      [000]      3312.225220: machine_suspend:
state=4294967295
```

From the output, you can see that the system started in state 3 (on state). It then transitioned into state 4294967295, which can be interpreted as state 0 (off state).

3.4.2 Low-Power Design Recommendation

Always implement *suspend()* and *resume()* functions in your driver. When the system enters the suspend state, the kernel will ask the drivers to suspend so they are in a state compatible with the target system state. If your drivers do not provide *suspend()* and *resume()*, the device will remain on, consuming power while the rest of the system is in a low-power state. This is an unnecessary waste of power.

Appendix A: Linux Power Management Subsystem

The Linux power management subsystem is a unified interface between the userspace and sysfs that is independent of the computer architecture running Linux.

To see the files in the Linux power management subsystem, enter the command below at the Linux shell prompt.

```
DM-37x# ls -l /sys/power
```

The following files are found in the *DM37x Linux Power Management Subsystem* directory:

```
-rw-r--r-- 1 root root 4096 Jan 1 00:32 notify_enable
-rw-r--r-- 1 root root 4096 Jan 1 00:32 notify_times
-rw-r--r-- 1 root root 4096 Jan 1 00:32 pm_async
-rw-r--r-- 1 root root 4096 Jan 1 00:32 pm_notify
-rw-r--r-- 1 root root 4096 Jan 1 00:32 pm_test
-rw-r--r-- 1 root root 4096 Jan 1 00:32 state
-rw-r--r-- 1 root root 4096 Jan 1 00:32 wakeup_count
```

pm_async

The *pm_async* file controls whether the suspend and resume of devices is carried out synchronously or asynchronously. This will have a direct impact on the latencies of suspend and resume.

state

The *state* file controls the state of the system. For example, writing *mem* to this file will cause the system to suspend to RAM.

```
DM-37x# echo mem > /sys/power/state
```

NOTE: To get out of suspend, press any key in the terminal.

pm_test

For information on the *pm_test* file, refer to Section 3.4.1.1

wakeup_count

The *wakeup_count* file is used to put a system into a sleep state while taking into account the concurrent arrival of wake-up events. Reading from it returns the current number of registered wake-up events and it blocks some wake-up events if they are being processed at the time the file is read from. Writing to it will only succeed if the current number of wake-up events is equal to the written value. If successful, writing to it will make the kernel abort a subsequent transition to a sleep state if any wake-up events are reported after the write has returned.