Sitara Linux Training: Power Management

Introduction

This lab is going to give you a hands on tutorial of Power Management (PM) using the AM335x Sitara Linux SDK. Each of the following sections below will walk you through a particular PM topic describing the actions that are about to be performed, the key points to take away, and the step-by-step instructions to complete the lab. If you have questions or feedback please e-mail the sdk_feedback@list.ti.com [1] mailing list.

Kernel Configuration

Description

This lab will familiarize the user will all Power Management related kernel configuration options.

Prerequisites

- Sitara Linux SDK installed

Key Points

- Gain familiarity with Linux menuconfig, specifically regarding PM configuration options.

Lab Steps

CPU idle Kernel Config

1. On host, open terminal and cd to /home/sitara/ti-sdk-am335x-evm-05.04.01.00/board-support/linux-3.2-psp04.06.00.07.sdk.
2. Start the *Linux Kernel Configuration* tool.

```
$ make menuconfig
```

1. Select *CPU Power Management* from the main menu.

```
   ...
```

```
...
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
  ...
```

1. Select *CPU idle PM support* to enable the cpuidle driver.

```
   ...
```

```
  ...
      CPU Frequency Scaling --->
  [*] CPU idle PM support
  ...
```

NOTE
cpuidle is enabled by default in the am335x_evm_defconfig.

CPU Freq Kernel Config

1. Select *CPU Power Management* from the main menu.

```
    ...
```

```
   ...
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
```

...

1. Select *CPU Frequency Scaling* as shown here:

```
    ...
```

```
   ...
      CPU Frequency Scaling --->
  [*] CPU idle PM support
```

...

1. Select *CPU Frequency scaling* and required governors as shown here. Not all governors may be enabled by default! If your application requires a specific governor, check here to ensure it has been enabled.

```
    [*] CPU Frequency scaling
```

```
   <*> CPU frequency translation statistics
  [ ] CPU frequency translation statistics details
      Default CPUFreq governor (userspace)  --->
  < >   'performance' governor
  < >   'powersave' governor
  -*-   'userspace' governor for userspace frequency scaling
  <*>   'ondemand' cpufreq policy governor
  < >   'conservative' cpufreq governor
  ...
```

NOTE
CPUFreq is enabled by default in the am335x_evm_defconfig.

PM Firmware Kernel Config (required for suspend/resume)

On AM335x, suspend-resume involves loading a binary to the Cortex-M3 core during the boot process. For this, the binary named am335x-pm-firmware.bin needs to be kept in the firmware/ folder on the kernel sources before the build process is started.
For more information on how to obtain the above mentioned binary, refer to the accompanying Release Notes for the PSP package.
The pre-built kernel image in the PSP package has the binary blob compiled into the kernel image.

To manually compile a kernel image with the PM firmware start the *Linux Kernel Configuration* tool.

1. Start the *Linux Kernel Configuration* tool.

```
$ make menuconfig
```

1. Select *Device Drivers* from the main menu.

```
 ...
```

```
...
Kernel Features  --->
Boot options  --->
CPU Power Management  --->
Floating point emulation  --->
Userspace binary formats  --->
Power management options  --->
[*] Networking support  --->
Device Drivers  --->
...
...
```

1. Select Generic Driver Options

```
 Generic Driver Options
```

```
CBUS support
...
...
```

1. Configure the name of the PM firmware and the location as shown below

```
 ...
```

```
-*- Userspace firmware loading support
[*] Include in-kernel firmware blobs in the kernel binary
(am335x-pm-firmware.bin) External firmware blobs to build into the kernel binary
(firmware) Firmware blobs root directory
```

Note: The above configuration assumes that the PM firmware named **am335x-pm-firmware.bin** has been placed under **firmware** folder of the kernel sources.

NOTE
The above changes are enabled by default in the am335x_evm_defconfig.

Enable/Disable suspend-resume support

1. To enable/ disable suspend-resume support start the *Linux Kernel Configuration* tool.

```
 $ make menuconfig
```

1. Select *Power management options* from the main menu.

```
 ...
```

```
...
Kernel Features  --->
Boot options  --->
CPU Power Management  --->
Floating point emulation  --->
Userspace binary formats  --->
Power management options  --->
[*] Networking support  --->
Device Drivers  --->
...
...
```

1. Select *Suspend to RAM and standby* to toggle the power management support.

```
 [*] Suspend to RAM and standby
```

```
-*- Run-time PM core functionality
...
< > Advanced Power Management Emulation
```

Dynamic Voltage and Frequency Scaling (cpufreq)

Description

Dynamic Voltage and Frequency Scaling, or DVFS, is realized via the cpufreq framework. Cpufreq provides support to change frequency of processor on the run. This helps to save processor power, when the load is less (processor power is proportional to frequency and square of the voltage).

The cpufreq governor decides frequency at which processor should run. Various governors like 'ondemand', 'userspace', 'performance', 'powersave', and 'conservative' exist in Linux Kernel.

- performance

  The Performance governor forces the CPU to use the highest possible clock frequency. This frequency will be statically set, and will not change. As such, this particular governor offers no power saving benefit. It is only suitable for hours of heavy workload, and even then only during times wherein the CPU is rarely (or never) idle.

- powersave

  By contrast, the Powersave governor forces the CPU to use the lowest possible clock frequency. This frequency will be statically set, and will not change. As such, this particular governor offers maximum power savings, but at the cost of the lowest CPU performance. However, it is possible for a slow CPU on full load to consume more power than a fast CPU that is not loaded. Therefore, 'powersave' is most useful in systems and environments where overheating can be a problem.

- ondemand

  The Ondemand governor is a dynamic governor that allows the CPU to achieve maximum clock frequency when system load is high, and also minimum clock frequency when the system is idle. The cost of this dynamism is the latency caused by frequency switching. As such, latency can offset any performance/power saving benefits offered by the Ondemand governor if the system switches between idle and heavy workloads too often. For most systems, the Ondemand governor can provide the best compromise between heat emission, power consumption, performance, and manageability. When the system is only busy at specific times of the day, the Ondemand governor will automatically switch between maximum and minimum frequency depending on the load without any further intervention.
- userspace (default)

  The Userspace governor allows userspace programs (or any process running as root) to set the frequency. Of all the governors, Userspace is the most customizable; and depending on how it is configured, it can offer the best balance between performance and consumption for your system.
- conservative

  Like the Ondemand governor, the Conservative governor also adjusts the clock frequency according to usage (like the Ondemand governor). However, while the Ondemand governor does so in a more aggressive manner (it only switches between maximum and minimum frequency), the Conservative governor switches between frequencies more gradually. It may use all available frequencies. The latency is greater than the Ondemand governor.

When frequency is changed, the voltage is also changed according to the selected frequency. Each voltage-frequency pair is known as an Operating Performance Point (or OPP). The TRM defines the OPP's for the given processor. AM335x OPP's are as follows:

- OPP50: 275MHz, 0.95V
- OPP100: 500MHz, 1.1V
- OPP120: 600MHz, 1.2V
- OPPTurbo: 720MHz, 1.26V

Prerequisites

- The Sitara Linux SDK should be installed and booted.

Key Points

- Learn how to select a cpufreq governor and the differences between them.
- Learn how to manually select frequency (when using userspace governor).

Lab Steps

Exploring CPUFreq Governors

1. View all available governors:

```
target# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
ondemand userspace
```

target#

1. View the current governor:

```
target# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
userspace
```

View and Change Frequency (OPP)

An OPP (Operating Performance Point) is a voltage-frequency pair that defines a specific power state that the SoC supports running at. Refer to SoC Technical Reference Manual [2] for more details.

1. First, set the cpufreq governor back to 'userspace', which is required in order to change the frequency manually.

```
target# echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

target#

1. View the current frequency (note that your frequency may differ depending on the previous steps of the lab):

```
target# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
720000
```

target#

1. View the current voltage:

```
target# cat /sys/class/regulator/regulator.3/microvolts
1262500
```

target#

1. View the supported OPP's (frequencies):

```
target# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
275000 500000 600000 720000
```

target#

1. Change the frequency to 500MHz with the command below. This can be done only for userspace governor. If ondemand governor is used, OPP change happens automatically based on the system load.

```
target# echo 500000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

Suspend / Resume

Description

The suspend operation results in the system transitioning to the lowest possible power state. On AM335x, this maps to DeepSleep0 state. For more info on this refer to the AM335x TRM available @ www.ti.com/am335x

The drivers implement the suspend() function defined in the LDM. When the suspend for the system is asserted, the suspend() function is called for all drivers. The drivers quiesce the peripherals and release the clocks to reach the desired low power state. The actual transition to suspend is implemented in the function am33xx_pm_suspend().

Wakeup from suspend is supported using the following interfaces:

- UART0 (console)
- GPIO0
- Touchscreen

Prerequisites

- Sitara Linux SDK installed

Key Points

- Learn how to manually suspend and resume the system.

Lab Steps

1. (optional, but more fun) From Matrix home screen, touch '3D', then touch 'Chameleon' to see 3D graphics in action.
2. The suspend for device can now be asserted as follows (note that it is recommended to issue a 'sync' first to avoid data loss):

```
$ sync
$ echo mem > /sys/power/state
```

1. To wakeup, send a character over the UART0 terminal, or touch the touchscreen.
2. The following output is expected for a successful suspend/resume cycle:

```
[root@arago /]# echo mem > /sys/power/state
[    8.774536] PM: Syncing filesystems ... done.
[    8.804199] Freezing user space processes ... (elapsed 0.01 seconds) done.
[    8.827575] Freezing remaining freezable tasks ... (elapsed 0.01 seconds) done.
[    8.847717] Suspending console(s) (use no_console_suspend to debug)
[    8.868530] PM: suspend of devices complete after 12.542 msecs <-- Wake event using console (UART0)
[    8.870147] PM: late suspend of devices complete after 1.556 msecs
[   10.840209] GFX domain entered low power state
[   11.044891] PM: early resume of devices complete after 204.184 msecs
[   11.284027] PM: resume of devices complete after 238.709 msecs
[   11.317382] Successfully transitioned all domains to low power state
[   11.325012] Restarting tasks ... done
[root@arago /]#
```

1. If you were running the chameleon man demo, you'll see that chameleon man begins running again.

CPU Idle

Description

The cpuidle framework consists of two key components:

- A governor that decides the target C-state of the system.
- A driver that causes the transition to the target C-state.

AM335x contains only two C-states. State C1 is described as 'MPU WFI', where WFI means 'wait for interrupt'. Essentially, the MPU is notified that there is no work to do, and it enters a low-power state. State C2 is deeper, and in addition to MPU WFI, it takes the DDR into self-refresh mode, saving more power.

Cpuidle governors available are 'menu' and 'ladder':

- ladder - steps down or up sleep states one at a time depending on the time spent in the last idle idle period. It works well with a regular timer tick, but not with

dynamic tick.

- menu - selects sleep state based on expected idle time. Works well with dynamic tick systems.

Prerequisites

- Sitara Linux SDK installed

Key Points

- Become familiar with AM335x C-states.
- Become familiar with cpuidle governors.

Lab Steps

1. Use sysfs interface to view detailed information about the C-states (state0 and state1). Sysfs contains useful entries which have detailed information about the C-states (state0 and state1). Most importantly, 'time' allows you to see how much time has been spent in a given state.

[root@arago /]# ls -l /sys/devices/system/cpu/cpu0/cpuidle/state0/

```
-r--r--r--    1 root     root         4096 Jan  1 00:02 desc
-r--r--r--    1 root     root         4096 Jan  1 00:02 latency
-r--r--r--    1 root     root         4096 Jan  1 00:02 name
-r--r--r--    1 root     root         4096 Jan  1 00:02 power
-r--r--r--    1 root     root         4096 Jan  1 00:02 time
-r--r--r--    1 root     root         4096 Jan  1 00:02 usage
```

```
[root@arago /]# ls -l /sys/devices/system/cpu/cpu0/cpuidle/state1/
-r--r--r--    1 root     root         4096 Jan  1 00:05 desc
-r--r--r--    1 root     root         4096 Jan  1 00:05 latency
-r--r--r--    1 root     root         4096 Jan  1 00:03 name
-r--r--r--    1 root     root         4096 Jan  1 00:05 power
-r--r--r--    1 root     root         4096 Jan  1 00:05 time
-r--r--r--    1 root     root         4096 Jan  1 00:02 usage
```

1. Print out the time spent in state0.

```
target# cat /sys/devices/system/cpu/cpu0/cpuidle/state0/time
```

target#

1. Run the previous command a few times to observe that the timestamp is increasing. This is because the MPU is actually able to re-enter the idle state between your commands!
2. Now, let's write a simple shell script to repeatedly dump the timestamp.

```
target# cd ~
```

target# vi test.sh **target#**

1. Press 'i' to enter insert mode on vi, and type:

```
#/bin/sh
```

while true do cat /sys/devices/system/cpu/cpu0/cpuidle/state0/time

done

1. In vi, hit 'esc' and ':wq!' to save.
2. Make your script executable:

```
target# chmod 777 test.sh
```

target#

1. Execute your script! You will notice that because we are keeping the CPU busy, the state0 timestamp does not advance.

```
target# ./test.sh
```

target#

1. Kill the script with Ctrl-Z at anytime.
2. You may notice on the AM335x EVM that state1 is never entered. This is due to the LCD module being enabled and driving constant interrupts. If the LCD module was disabled, CPUIdle state1 is possible and represents a lower power state of idle.

References

[1] mailto:sdk_feedback@list.ti.com
[2] http://www.ti.com/lit/pdf/sprs717

Article Sources and Contributors

Sitara Linux Training: Power Management *Source*: http://processors.wiki.ti.com/index.php?oldid=157979 *Contributors*: Fcooper, Gary, Gguyotte, Kevinsc

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha