ARM 9 에서의 전력관리 구현

손동환○ 이형석 김선자
한국전자통신연구원 컴퓨터·소프트웨어연구소
{dhson○, hyslee, sunjakim}@etri.re.kr

Power Management Implementation on ARM 9 based Platform

Donghwan Son○ , Hyung-Seok Lee, Sunja Kim
Computer & Software Laboratory, ETRI

요 약

전력관리는 시스템의 성능 저하를 최소화 하면서 전력 소모를 최소화 하기 위한 기술로써 하드웨어 별로 구현이 되어야 한다. 본 논문은 ARM 9 기반의 플랫폼 상에서 디바이스별 또는 시스템 전체 레벨에서의 전력관리를 수행할 수 있도록 하기 위한 기반 작업과 시스템의 상태를 주기적으로 모니터링 하는 데몬 및 사용자 인터페이스를 구현한 내용을 기술하였다.

1. 서 론

시스템에서 운영체제의 두 가지 중요한 역할은 추상화의 제공과 자원의 관리이다. 운영체제는 시스템 하드웨어의 기능을 추상화하여 응용프로그램이 하드웨어에 대한 정보 없이 시스템 콜을 통하여 제어를 할 수 있도록 한다. 또한 운영체제는 프로세서, 디스크, 메모리 등의 자원을 관리한다.

전력은 시스템의 하나의 자원이지만 전원이 연결되어 있는 시스템에서는 거의 무한한 자원이라고 볼 수 있기 때문에 최근까지 운영체제에서의 전력관리에 대한 고려가 없었으나 이동 기기의 증가와 이에 비하여 배터리의 느린 발전속도로 인하여 중요한 자원으로 인식되어지고 있으며 운영체제의 성능을 가름하는 주요 요인으로 자리를 잡게 되었다. 전력관리는 기본적으로 시스템의 성능의 감소를 최소화하면서 시스템이나 기기가 사용되지 않을 때 전력 상태를 저전력 상태로 천이시킴으로써 불필요한 전력소모의 낭비를 막는 것과 시스템이나 기기의 작업의 변화를 통해 저전력 상태로 존재하는 시간을 최대화하는 것을 목적으로 한다.

전력관리는 하드웨어에서 지원하는 전력 관리 기능을 효율적으로 이용하기 위한 것이므로 하드웨어에 따라 구현이 되어야 한다. 인텔 계열의 하드웨어들은 ACPI 규격에 의해 설계가 되어있으며 이 규격에 따른 OS에서의 전력관리 구현이 이루어져야 한다.[1] 비 인텔 계열의 경우에는 이러한 규격이 아직 없으며 따라서 플랫폼 별로 OS에서의 전력관리 구현이 이루어져야 한다. 단지 인터페이스의 경우는 APM 인터페이스를 많이 쓰고 있다.

본 논문에서의 전력관리의 구현은 ARM 9 계열의 삼성 S3C2400 reference 보드 상에서 구현되어 있다.[2,3] S3C2400은 ARM 9 core외에 여러 디바이스가 하나의 칩으로 구성되어있고 이들의 전력상태를 제어하는 전력모듈이 내장되어있다. 이 전력모듈을 통해 전체 시스템을 sleep모드로 변환시킬 수 있을 뿐 아니라 각 디바이스별로 sleep 모드로 천이시킬 수 있다. 따라서 이 전력모듈에 대한 일종의 디바이스 드라이버를 구현함으로써 하위레벨에서의 전력 상태 제어를 할 수가 있다. 이러한 디바이스 드라이버를 이용하여 시스템의 모니터링하여 사용자의 설정한 값과 비교하여 시스템이 idle 상태인지를 결정하고 idle 상태일 경우 자동으로 시스템의 상태를 STOP 모드로 천이시키는 daemon 및 사용자 설정을 도와주는 GUI를 구현하였다.

2. 전력관리 구조

Qplus-P에서의 전력관리는 아래와 같은 구조를 가지고 있다.

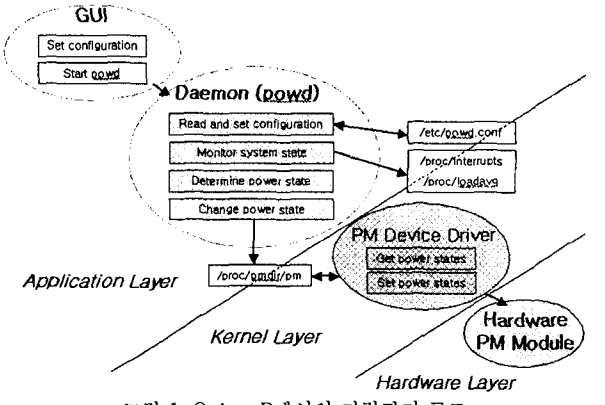

그림 1. Qplus-P에서의 전력관리 구조

PM Device Driver를 통하여 하드웨어의 전력상태를 읽거나 설정할 수 있으며 Daemon(powd)은 이러한 기능을 이용하여 시스템을 모니터링하고 사용자가 설정한 조건에 의해 시스템의 전력 상태를 천이시킨다. GUI는 사용자가 이러한 조건을 설정하기 위한 도구이다.

2.1 하드웨어 전력관리 모듈

S3C2400은 ARM9 계열의 코어 외에 SPI, ADC, USBD 등의 디바이스들과 이들의 전력상태를 제어하는 전력관리 모듈로 구성되어있다. 전력관리 모듈은 관련 레지스터를 통하여 각 디바이스 및 시스템 전체의 전력상태를 제어할 수 있다. S3C2400는 NORMAL, SLOW, IDLE 및 STOP모드로 구성되는 총4개의 시스템 전력모드를 가지고 있다.

NORMAL모드는 S3C2400 내의 CPU 및 주변기기에 클럭을 제공하는 상태이고 이 경우에 최대의 전력소모가 일어난다. 사용자는 S/W에 의해 주변기기의 동작을 제어할 수 있다. 예를 들어 타이머가 필요하지 않은 경우에는 전력소모를 줄이기 위해 타이머를 위한 클럭을 끊을 수 있다.

SLOW모드는 non-PLL모드로써 PLL 없이 외부 클럭을 직접 FCLK로 사용함으로써 PLL에 의한 전력소모를 줄이는 모드이다. 하지만 SLOW모드는 STN모드의 LCD에서만 동작하고 TFT모드에서는 동작하지 않기 때문에 TFT LCD 모드로 동작하는 현재의 구현 상에는 지원되지 않는다.

IDLE모드는 CPU에 FCLK를 끊음으로써 CPU에 의한 전력소모를 줄이는 모드이다. CPU에 대한 인터럽트로 wake-up 시킬 수 있다.

STOP모드는 PLL을 disable 시킴으로써 CPU와 모든 주변기기의 동작을 중단시키는 모드이다. wake-up은 외부 인터럽트나 RTC 알람에 의해 일어날 수 있다.

전력관리 모듈은 관련 레지스터를 통하여 각 디바이스를 on/off시키고 프로세서를 sleep모드로 천이시킬 수 있으며 또한 메모리를 제외한 시스템 전체의 sleep모드로의 전환을 시킬 수 있다. 시스템 전체를 sleep모드로 전환시킬 때에는 시스템의 정보를 유지하기 위해 메모리를 self-refresh 상태로 설정해야 하며 이 때는 메모리의 접근이 불가능하기 때문에 sleep 관련 코드는 롬상에 존재해야 하므로 boot loader에 시스템 sleep 관련 루틴을 구현하였다.

2.2 전력모듈 디바이스 드라이버

전력모듈의 제어는 커널 상의 디바이스 드라이버 모듈로 구현하였다.[4] 인터페이스는 /proc 인터페이스를 사용하였고 /proc/pmdir/pm 파일을 통해 전력모듈을 제어하거나 전력상태 정보를 read할 수 있다. 관련 루틴은 아래와 같다.

- pm_idle : 프로세서의 전력상태를 sleep 상태로 전환하는 루틴
- pm_stop : 램을 제외한 시스템 전체의 전력상태를 sleep 상태로 전환하는 루틴
- pm_spi, pm_iis, pm_iic, pm_adc, pm_rtc, pm_gpio, pm_uart1, pm_uart0, pm_mmc, pm_pwmt, pm_usbd, pm_usbh, pm_lcdc : 각각 SPI, IIS, IIC, ADC, RTC, GPIO, UART1, UART0, MMC, PWMT, USBD, USBH, LCDC의 전력상태를 sleep 상태로 전환하는 루틴

전력 상태의 제어 및 전력 상태 정보의 획득은 아래와 같은 방법으로 이루어진다.

/proc 인터페이스를 통한 전력상태의 제어 예

```
# echo " ADC 0"  >/proc/pmdir/pm  -> ADC를 sleep 모드로 전환
# echo " ADC 1"  >/proc/pmdir/pm  -> ADC를 wakeup
# echo " SPI 0"  >/proc/pmdir/pm  -> SPI를 sleep 모드로 전환
# cat /proc/pmdir/pm              -> 각 기기의 전력상태 출력
  SPI    OFF
  IIS    ON
  IIC    ON
  ADC    ON
  RTC    ON
  GPIO   ON
  UART1          ON
  UART2          ON
  MMC    ON
  PWMT   ON
  USBD   ON
  USBH   ON
  LCDC   ON
  IDLE   ON
  SIDLE  ON
  STOP   ON
  SLOW   ON
  ....
# echo " SLEEP"  > /proc/pmdir/pm          -> 프로세서를 sleep 모드로 전환
  S3C2400 will wake up by keyboard interrupt ->Sleep 모드
                        <- keyboard interrupt
  Return to normal mode            -> wakeup
# echo " STOP"  > /proc/pmdir/pm  -> 시스템 전체를 sleep 모드로 전환
  S3C2400 will wake up by keyboard interrupt ->Sleep 모드
```

실제로 USB는 하드웨어 전력관리 모듈의 불완전한 동작으로 인하여 sleep은 수행이 되나 wakeup 이후에는 정상적인 동작을 안하므로 따로 USB 초기화를 수행시켜야 한다. USB 초기화 루틴은 USB 드라이버의 수정을 통해 /proc/usb파일에 임의의 값을 writing하는 것으로 수행이 되도록 하였다.

2.3 powd daemon

powd는 시스템의 인터럽트 및 workload를 모니터링 하여 사용자의 설정한 값과 비교하여 시스템이 idle 상태인지를 결정하고 idle 상태일 경우 자동으로 시스템의 상태를 STOP모드로 천이시키는 기능을 수행하도록 사용자 daemon의 형태로 구현되었다.
사용자는 /etc/powd.conf파일을 통해 inactivity 및 workload 값을 설정할 수 있다. 시스템이 사용되지 않은 시간이 inactivity 보다 크고, 그 때의 시스템의 작업량이 workload 값보다 작을 경우에 powd는 시스템이 사용되지 않는다고 판단하고 시스템을 STOP 모드로 천이시키게 된다. powd는 이러한 판단 작업을 1초에 한번씩 하도록 구현되었다. 여기에서 시스템의 사용 여부 및 시스템의 작업량에 대한 정보는 각각 /proc/interrupts와 /proc/loadavg를 통해 구한다.
powd는 아래와 같은 루틴들로 구성되어 있다.

- parse_command_line : 커맨드라인 상에서 사용자가 입력한 argument를 읽거나 입력한 argument가 없을 때에는 /etc/powd.conf파일로부터 설정값을 읽는다. 이 읽은 값들에 의해 powd가 수행된다.
- check_cpu : CPU의 평균 workload값을 /proc/loadavg로부터 읽어와서 사용자가 설정한 값보다 크면 1을 리턴한다.
- probe_IRQs : /proc/interrupts를 통해 인터럽트 종류와 발생 회수를 읽어서 이전의 인터럽트 발생 회수와 비교해 차이가 있으면 1을 리턴한다. 그러나 USB는 사용이 없어도 주기적으로 인터럽트를 발생시키므로(매 초당 약 2회) 이를 고려해서 사용이 있었는지를 판단한다.
- do_sleep : /proc/pmdir/pm을 통해 sleep 루틴을 호출한다.
- 무한루틴 : chech_cpu와 probe_IRQs를 통해 시스템의 activity가 없다고 판단되면 total_unused값을 1 증가시키고 1초간 sleep을 한다. total_unused 값이 사용자가 설정한 시간보다 커지면 시스템을 sleep모드로 전환한다. check_cpu와 probe_IRQs를 통해 activity가 있었다고 판단되면, 즉 CPU의 평균 workload값이 사용자가 설정한 값보다 크거나 또는 인터럽트 발생이 있었다면 total_unused 값을 0으로 하고 1초간 sleep을 한다.

2.4 powd GUI

GUI는 GTK 1.2를 기반으로 구현되었으며 inactivity time과 평균 CPU 사용 threshold를 spin button을 통하여 입력하고 이 값을 argument로 Start button을 통해 powd를 수행시킬 수 있고 Stop 버튼을 통해 수행중인 powd를 중지시킬 수 있다.[5]

그림 2는 reference 보드 상에서 powd GUI가 수행되는 장면이며 3분 동안 인터럽트가 없고 그 때의 CPU 사용이 0.13보다 작다면 sleep 모드로 전환하도록 설정하고 있다.

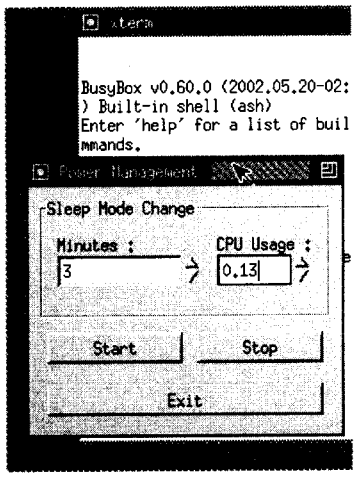

그림 2. powd GUI

3. 결론

본 논문은 하드웨어 전력관리 모듈을 제어할 수 있는 전력관리 디바이스 드라이버,시스템의 인터럽트 및 workload를 모니터링 하여 사용자의 설정한 값과 비교하여 시스템이 idle 상태인지를 결정하고 idle 상태일 경우 자동으로 시스템의 상태를 STOP 모드로 천이시키는 daemon 및 사용자 설정을 도와주는 GUI의 구현을 기술하였다. 이러한 기술을 통해 사용자의 입력이 없는 IDLE 한 상태에서의 전력소모를 NORMAL 상태의 약 3.3% 미만으로 줄일 수 있고 결과적으로 대기시간을 최대 300배까지 늘릴 수 있다.

참고문헌

[1] http://developer.intel.com/technology/iapc/acpi/
[2] Samsung S3C2400X User' s Manual.
[3] http://www.samsungelectronics.com/semiconductors/System_LSI/32_bit_ARM_based_RISC_MPU/PDA/S3C2400X/s3c2400x.htm.
[4] Ori Pomerantz, " Linux Kernel Module Programming Guide," iUniverse.com, Inc. 1999.
[5] www.gtk.org