# 실시간 운영체제를 위한 저전력 GUI 설계 및 구현

## Design and Implementation of Low-Power GUI for Real-Time Operating System

정재엽, 이철훈*
충남대학교 컴퓨터공학과

Jae-Yeop Jeong, Cheol-Hoon Lee*
Dept. of Computer Engineering,
Chungnam National University

**요약**

제한된 배터리를 가지고 있는 내장형 시스템에서 전력을 최소화하여 제한된 전력을 오랫동안 사용하는 기술은 중요한 문제로 인식되어 왔다. 최근 내장형 시스템은 대형 TFT-LCD와 터치스크린을 장착하면서 그에 따른 전력 소비가 더 증가하고 있다. 본 논문에서는 GUI에서 전력 소모를 감소할 수 있는 방안인 프레임버퍼 모니터링을 연구하였다. 프레임버퍼 모니터링 기법은 화면의 퀄리티(Quality)를 보장하는 범위 내에서 Refresh-rate과 Backlight을 조절하여 LCD에서 전력 소모를 줄이는 방법이다.

**Abstract**

A technique which uses the energy for a long time have been recognized as a important problem in embedded system with restricted battery. Recently the energy consumption is increased by using a large size of TFT-LCD and touch screen in embedded system. In this paper, we studied the frame buffer monitoring which can be reduced an energy consumption in GUI. The frame buffer monitoring technique is the energy degrade plan which adjusts Refresh-rate and Backlight. The technique must guarantee the quality of screen.

## I. 서 론

내장형 시스템(Embedded System)의 하드웨어 성능이 향상되고 사용자의 실생활과 밀접한 디바이스들이 개발됨에 따라, GUI(Graphic User Interface)는 사용자와 시스템간의 대화수단으로 그 중요성이 증가하고 있다. 최근 내장형 시스템에 고화질 대형 TFT-LCD와 터치스크린이 장착되면서, 전체 시스템의 전력소모 중 LCD의 소모전력 비중이 커지고 있다[1].

제한된 배터리를 가지고 있는 내장형 시스템에서 소모 전력을 최소화하여 제한된 전력을 오랫동안 사용하는 기술은 중요한 문제로 인식되어 왔다. CPU에 공급되는 전압을 조절하는 DVS나 각각의 디바이스의 상태에 따라 On/Off를 조절하는 DPM 기법은 많은 연구가 진행되었지만, 이 외에 추가적인 전력감소 방안이 필요하다. 이에 본 논문에서는 프레임버퍼 모니터링을 통해 화면의 퀄리티(Quality)를 보장하는 범위 내에서 Refresh-rate과 Backlight을 조절하여 전력 소모를 감소할 수 있는 방안을 제시한다.

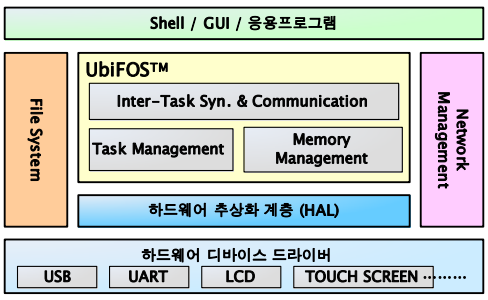
본 논문에서는 프레임버퍼 모니터링을 통한 Refresh-rate과 Backlight 조절을 이용하여 실시간 운영체제를 위한 저전력 GUI를 설계하고 구현한 내용을 기술한다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 프레임버퍼 모니터링에 기반한 저전력 GUI의 설계 및 구현을 다룬다. 4장에서는 실험환경 및 실험결과를, 마지막으로 5장에서는 결론 및 향후 연구과제에 대해 기술한다.

본 논문은 국방과학연구소의 임무통제장비 다중화 관리기법 연구과제의 지원으로 수행된 것임.

## II. 관련연구

### 1. 실시간 운영체제 UbiFOS™

실시간 운영체제인 UbiFOS™은 우선순위 기반의 선점형 스케줄러를 제공하며, 태스크의 우선순위를 0부터 255까지 256단계로 구분하고 있다. 그리고 실행 준비된 태스크들 중에서 정해진 시간 내에 가장 높은 우선순위의 태스크를 찾기 위해 별도의 테이블과 리스트를 관리하며, 같은 우선순위의 태스크 사이에서는 선입선출(FIFO) 방법과 라운드-로빈(Round-Robin) 방법을 지원한다[2].




▶▶ 그림 1. 실시간 운영체제 UbiFOS™의 전체 구성도

그림 1은 UbiFOS™ 의 전체 구조를 도시한 것으로, 운영체제에서 제공되는 기능은 태스크 관리, 태스크간 통신, 동기화 및 동적 메모리 관리 등이 있다.

## 2. 저전력 기법

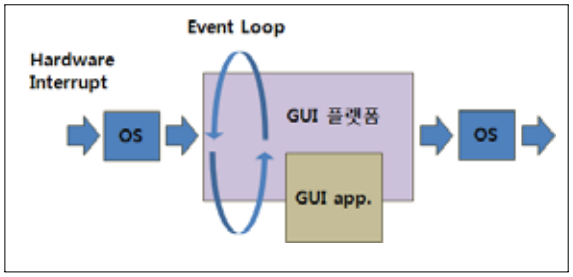
소프트웨어적인 저전력 기법으로는 컴파일러 레벨과 운영체제 레벨에서 연구되어 왔다. 컴파일러 레벨에서는 최적화된 컴파일 기법으로 메모리 사용을 최소화할 수 있고, 운영체제 레벨에서는 타겟 시스템의 목적에 따라 필요 없는 모듈을 제거함으로써 전력 소비를 줄일 수 있다. 대표적인 저전력 기법으로는 DVS[3]와 DPM[4]이 있다.

DVS는 실시간 시스템에서 데드라인을 만족하면서 공급전압을 동적으로 조절하는 것을 말한다. 일반적으로 CMOS 회로의 전력 소모는 공급전압에 대해 제곱의 관계($E \propto V^2$)를 가지므로 공급 전압의 감소는 에너지 소모를 줄일 수 있는 매우 효과적인 방법이다. 그러나 전압이 낮아질수록 회로의 최대 동작 주파수가 선형적으로 느려지기 때문에 프로세서의 처리량도 줄어들게 된다. 이러한 프로세서의 처리속도와 에너지 소모에 대한 관계를 이용하여 주어진 작업이 요구하는 서비스의 질을 낮추지 않는 범위 내에서 전압 및 클럭 속도를 조절하는 기법을 DVS라 한다.

DPM은 시스템의 구성 요소별 사용 패턴을 기반으로 해당 장치가 비활성화 상태에 있을 경우 해당 장치가 제공하는 구분된 파워 상태 중 주어진 수행의 요구조건에 맞추어 가장 적절한 파워 상태로 전이함으로써 전력의 소모를 줄이는 기법이다. 시스템을 구성하는 다양한 장치들의 특성에 따라 구체적인 구현의 내용은 달라지나 기본적으로 해당 장치가 사용되지 않는 사용 패턴에 대한 정확한 예측이 필요하며, 상태전이에 따른 시간 및 에너지 부대비용을 고려해야 한다.

# Ⅲ. 프레임버퍼 모니터링을 이용한 저전력 GUI의 설계 및 구현

## 1. 저전력 GUI 구현을 위한 고려사항




▶▶ 그림 2. GUI 애플리케이션 처리 순서

그림 2는 실시간 운영체제에서 GUI 애플리케이션의 처리 순서이며, 처리과정은 다음과 같다.

① 사용자로부터 발생된 하드웨어 인터럽트를 처리.
② GUI 애플리케이션에 이벤트 전달.
③ GUI 플랫폼이 화면을 업데이트.
④ 운영체제의 다른 서비스를 수행.

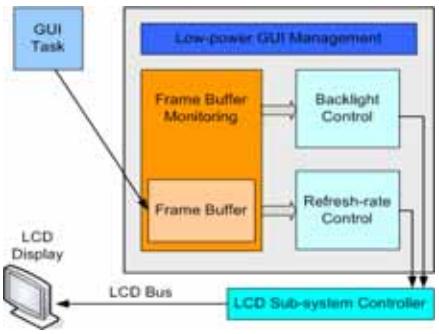
실시간 운영체제를 위한 저전력 GUI를 설계하기 위해서는 타겟 디바이스에 탑재되는 운영체제와 수행할 애플리케이션의 특성을 고려해야 한다. 저전력 GUI 설계 시 고려사항으로 GUI 자체의 크기, 성능, 이식성, 안전성 등이 있다. 이러한 조건을 만족하면서 실시간 운영체제의 성능에 영향을 주지 않도록 설계하려면 구현에 제약이 많이 따르게 된다.

본 논문에서 구현한 저전력 GUI는 배터리 용량에 민감한 내장형 시스템을 기반으로 하였으며, 사용될 실시간 운영체제는 시스템이 예상하지 못한 특정 이벤트가 발생하는 악조건에서도 태스크 수행의 데드라인을 초과하지 않도록 시간 결정성을 보장해야 한다.

## 2. 프레임버퍼 모니터링을 이용한 저전력 GUI 설계 및 구현

### 1. 프레임버퍼 모니터링

프레임버퍼 모니터링을 통해 화면의 퀄리티(Quality)를 보장하는 범위 내에서 Backlight과 Refresh-rate 조절을 통해 전력관리가 가능하다. Backlight과 Refresh-rate을 낮출 경우 디스플레이의 시그널을 늦춰 디스플레이 서브시스템의 모듈에서의 전력 감소가 일어난다. 또한 LCD 컨트롤러의 DMA가 사용하는 시스템 버스 타임을 줄임으로써 전력소비를 줄일 수 있다.

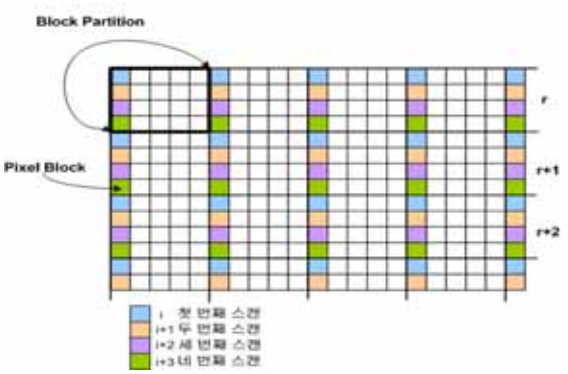



▶▶ 그림 3. 저전력 GUI 매니지먼트

Backlight과 Refresh-rate은 화면의 깜빡임이 나타나지 않는 범위 내에서 줄여야만 화면의 퀄리티를 보장할 수 있다. 그러나 LCD의 특성상 화면의 정보에 따라 Backlight과 Refresh-rate의 최소 요구치가 다르다. 픽셀 구성이 단순한 텍스트 위주의 화면은 Backlight과 Refresh-rate을 낮게 줘

도 깜빡임이 나타나지 않아 만족할만한 퀄리티를 보여주지만, 픽셀 구성이 복잡한 고화질 이미지에서는 낮은 Backlight과 Refresh-rate이 화면의 퀄리티를 저하시킨다.

본 논문에서 제안하는 프레임버퍼 모니터링 기법은 화면에 나타나는 전체 화면에 대한 이미지 영역을 수집한다. 수집된 정보를 이용하여 고화질 이미지와 텍스트를 구분함으로써 Backlight과 Refresh-rate 조절에 이용한다.




▶▶ 그림 4. Pixel Block 과 Block Partition

프레임버퍼를 모니터링하는 방법은 그림 4와 같다. 전체 화면을 k * l 픽셀 크기의 격자 형태로 분할하고, 분할된 각 영역을 Pixel Block이라 한다. 여러 개의 Pixel Block들을 합쳐서 화면을 Block Partition 단위로 나눈다. Block Partition에 들어있는 Pixel Block의 수를 m이라 한다. 위 그림에서 i ~ i+3은 프레임버퍼를 스캔한 순서를 나타낸다. 화면 해상도를 x * y 픽셀이라 했을 때 Block Partition의 크기는 (x * y)/m 픽셀이 된다.

분할된 프레임버퍼에서 Pixel Block Partition에서만 값을 읽어 들이고, 그 위치는 그림 4와 같다. r행의 Block Partition에서 첫 번째 Pixel Block 값을 읽어 온 후에 r+1, r+2 행에서도 같은 위치의 값을 읽는다. 즉 일정한 간격의 픽셀을 읽게 된다. 전체 프레임버퍼를 스캔한 후, 두 번째 스캔인 i+1 스캔에서도 (x,y)의 위치가 (x,y+1)의 위치로 바뀌고 각각의 Block Partition 내부의 간격은 일정한 위치를 스캔한다. x,y의 방향의 좌표가 범위를 벗어날 경우 로테이션(rotation)하도록 한다.

Pixel Block은 일정한 위치에 해당하는 픽셀만을 비교한다. 이미지 영역의 경우 일정한 경계 값(t) 이상의 색상이 사용되면 이미지로 판단 가능하므로 k와 l 그리고 t의 값을 조절하여 대각 위치의 픽셀만으로 이미지와 텍스트의 정확한 구분이 가능하다. 즉, t값을 넘어서는 경우를 이미지로 간주하게 된다. 프레임버퍼 모니터링에 의한 정보를 이용하여 이미지로 판단되면 Backlight과 Refresh-rate을 조절하여 전력 감소를 할 수 있다.

### 2. 프레임버퍼 모니터링을 이용한 Backlight 조절

LCD에서의 전력 소비는 Backlight에서 가장 많이 발생한다. Backlight 조절은 화면에 대한 이미지의 비율에 따라 조절하며, 사용자 환경에 따른 퀄리티의 정도가 다르므로 하드웨어의 종류에 따라 변화할 수 있다.

[표 1] Backlight 비율

| 이미지 비율 | Backlight(Candela) |
|---|---|
| 0 ~ 10 % | 50 cd |
| 10 ~ 30 % | 100 cd |
| 30 ~ 70 % | 150 cd |
| 70 ~ 100 % | 255 cd |

표 1은 Backlight과 이미지 영역 비율에 대한 일람표이다. 프레임버퍼를 모니터링한 후 이미지의 영역 비율이 10% 이하일 경우에는 Backlight를 50cd(Candela)로, 10~30%일 경우에는 100cd, 30~70%는 150cd, 70% 이상 일 때는 255cd로 조절한다.

Backlight 조절은 프레임버퍼 모니터링에서 전체 화면에 대한 이미지의 비율을 넘겨받아 표1과 같이 조절한다. 표 1의 각 상태에서는 퀄리티를 보장하는 범위에서 Backlight을 조절하므로 전력 소모를 감소 할 수 있다.

### 3. 프레임버퍼 모니터링을 이용한 Refresh-rate 조절

Refresh-rate 조절은 전체 화면에 대한 이미지 영역 비율에 따라 동작한다. 사용자의 환경에 따른 퀄리티 보장과 동시에 전력 감소를 위해서는 이미지 영역의 비율에 따라 깜빡임이 나타나지 않는 최저 Refresh-rate에 대한 정보가 필요하다. 깜빡임이 나타나는 기준은 하드웨어 의존적인 부분으로써 LCD 종류와 화면의 크기, 컨트롤러 등에 의해서도 변할 수 있다. 따라서 각 장치마다 이를 명시한 하드웨어 일람표의 구성이 필요하다. 이는 해당 기기의 각 Refresh-rate단계마다 이미지의 영역을 늘려가면서 깜빡임이 발생하는 이미지 비율의 범위를 토대로 작성 가능하다.

[표 2] Refresh-rate 비율

| 이미지 비율 | Refresh Rate |
|---|---|
| 0 ~ 25 % | 20 Hz |
| 25 ~ 50 % | 40 Hz |
| 50 ~ 70 % | 50 Hz |
| 70 ~ 100 % | 60 Hz |

표 2는 Refresh-rate과 이미지 영역 비율에 대한 일람표이다. 이미지의 영역이 25% 이하일 경우는 20Hz 모드로, 25~

50%일 때는 40Hz, 50~70%일 때는 50Hz, 70%이상 일때는 60Hz로 조정한다.

Refresh-rate 조절은 프레임버퍼 모니터링에서 전체화면에 대한 이미지의 영역의 비율을 넘겨받아 표 2에 나타나있는 상태로 조절한다. 표 2의 각 상태 공간에는 깜빡임이 나타나지 않는 이미지 영역 비율의 범위가 명시되어 있어 퀄리티를 보장하면서 전력 소모를 줄일 수 있다. 특히, 텍스트 위주의 작업인 경우 전력 소모가 가장 적은 Refresh-rate(20Hz)을 사용함으로써 전력을 감소할 수 있다.

### 4. 저전력 GUI 수행 과정

본 연구에서는 구현한 저전력 GUI를 실시간 운영체제인 UbiFOS™ 실에 탑재하였다. 그림 5는 UbiFOS™에서 저전력 GUI가 실행되는 과정이다. 애플리케이션에서 GUI 태스크를 생성할 때, MK_LPGUIControl() 함수를 이용하여 저전력 모드로 실행하게 된다.

[표 3] 저전력 GUI API

| API | 역 할 |
|---|---|
| MK_LPGUIControl() | 저전력 모드와 일반모드 구분 |
| MK_MonFrameBuffer() | 프레임버퍼 모니터링 |
| MK_Backlight() | Backlight cd(Candela) 조절 |
| MK_RefreshControl() | Refresh-rate Hz 조절 |

```
MK_Status_t
MK_MonFramebuffer(MK_TASK *pTask)
{
  ......
  Lcd_Init(MODE_TFT_8BIT_240320);
  Glib_Init(MODE_TFT_8BIT_240320);

  Select_Tcon();
  Lcd_PowerEnable(0, 1);
  rLCDCON5|=(1<<11);

  /* 프레임버퍼의 내용을 읽어와 image에 저장 */
  for(j=0;j<320;j++)
    for(i=0;i<240;i++)
      PutPixel(i,j,((int)DEMO256[k++]));

  rGPGDAT=rGPGUP&(~(1<<4))|(1<<4);
  rLCDCON1=rGPGDAT&(~(1<<4))|(1<<4);
  rLCDCON2=rGPGCON&(~(3<<8))|(1<<8);
  image = rLCDCON1 & rLCDCON2

  if (image<=10) {
        MK_BacklightControl(50);
  } elseif(10<image<=30) {
        MK_BacklightControl(100);
  } elseif(30<image<=70) {
        MK_BacklightControl(155);
  } else {
        MK_BacklightControl(255);
  }

  if(image<=25) {
        MK_RefreshControl(20);
  } elseif(25<image<=50) {
        MK_RefreshControl(40);
  } elseif(50<image<=70) {
        MK_RefreshControl(50);
  } else {
        MK_RefreshControl(60);
  }
  ......
}
```

프레임버퍼 모니터링

```
MK_Status_t
MK_BacklightControl(MK_INT_t Argc)
{
  ......
  image = Argc;
  Lcd_Start(image);
  rLCDCON5 &= 0x1ffffd;
  Lcd_EnvidOnOff(0);
  Lcd_Port_Return();
  rLCDCON5 &= 0x1ffffd;
  ......
}
```

Backlight 조절

```
MK_Status_t
MK_RefreshControl(MK_INT_t Argc)
{
  ......
  image = Argc;
  rLCDCON5 = 0x1ffffd & image;
  rLCDCON5 &= 0x1ffffd;
  rGPGUP = rGPGUP&(~(1<<4))|(1<<4);
  rGPGCON=rGPGCON&(~(3<<8))|(1<<8);
  pISR_LCD=(unsigned)Lcd_Int_Frame;
  ......
}
```

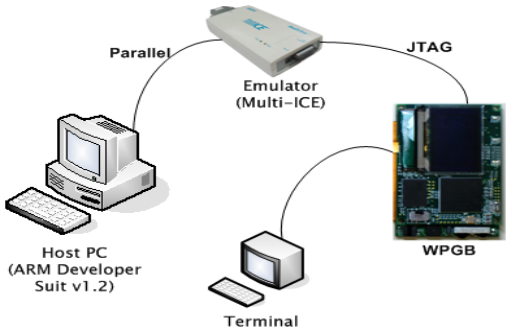
Refresh-rate 조절

▶▶ 그림 5. 저전력 GUI 수행 과정

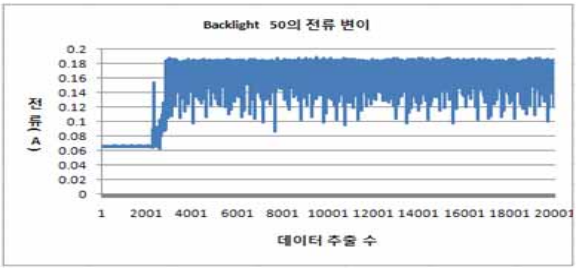
MK_LPGUIControl() 함수에서 MK_MonFrameBuffer() 함수를 호출하여 프레임버퍼의 정보를 읽는다. 이 정보를 바탕으로 조정할 Backlight과 Refresh-rate를 분기시켜 적절한 값으로 조절하게 한다. 표 3은 저전력 GUI에서 구현된 API이다.

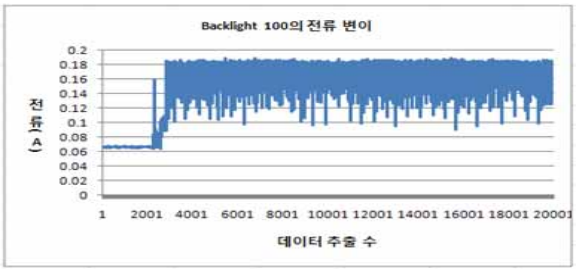
## Ⅳ. 실험 환경 및 결과

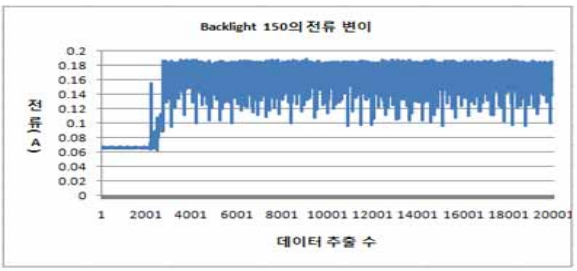
본 논문에서 구현한 저전력 GUI는 ARM 926EJ 기반의 i.MX21 Processor가 탑재된 WPGB(Wearable Personal Gateway Board)에서 실험하였다.

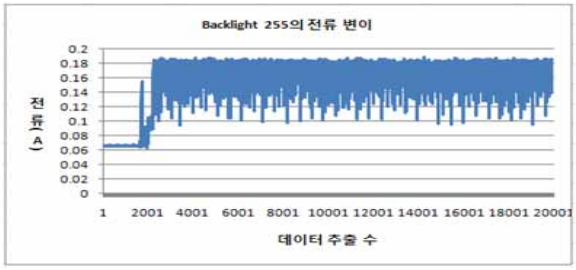



▶▶ 그림 6. 실험 환경






▶▶ 그림 7. Backlight변화에 따른 전류변이

그림 7은 프레임버퍼 모니터링을 이용한 저전력 GUI를 적용하여 bmp형식 이미지와 txt형식을 LCD창에 디스플레이를 수행한 결과로, 시스템 전체 구동 시 소모되는 전류를 멀티미터로 측정하였다. 표 4는 표 1에서와 같이 Backlight에 따른 전체 시스템 전류 소비의 평균값으로, 약 2%의 전력 감소 효과가 있으며, 소모전력 감소는 구동시간에 비례할 수 있다. Refresh-rate의 변화에 따른 전력소모는 기기의 결함으로 측정하지 못하였지만, 효율적인 전력감소 효과가 있을 것으로 기대된다.

[표 4] Backlight에 따른 전체 시스템 전류 소모

| | 50cd | 100cd | 150cd | 255cd |
|---|---|---|---|---|
| 전류 평균값 | 0.150027A | 0.150311A | 0.150781A | 0.153017A |

## V. 결론 및 향후 연구과제

본 논문에서는 최근 내장형 시스템에 고화질의 대형 TFT-LCD와 터치스크린이 사용됨에 따라, 시스템 전체전력의 많은 부분을 차지하고 있는 GUI의 소모 전력을 감소하기 위해 프레임버퍼 모니터링을 이용하여 저전력 GUI를 설계 및 구현하였다.

향후 연구과제로는 프레임버퍼 모니터링을 기반으로 Refresh-rate의 변화에 따른 전력 소모를 측정하고, color depth의 부분에서도 화면의 퀄리티를 보장하면서 전력을 감소할 수 있도록 연구가 진행되어야 한다.

▌참 고 문 헌▐

[1] Hyoseung Kim, Hojung Cha, Rhan Ha, "Dynamic refresh-rate scaling via frame buffer monitoring for power-aware LCD management", Wiley InterScience, 12, September, 2006

[2] Aijisystems, "Embedded hardware & software solution, UbiFOS(Ubiquitous Flexible Real-Time Operating Systems) & MCU platforms," http://www.aijisystem.com.

[3] Woonseok Kim, Dongkun Shin, Han-Saem Yun, Jihong Kim, Sang Lyul Min, "Performance Comparision of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems", IEEE, Real-Time and Embedded Technology and Applications Symp., 2002

[4] IBM and MontaVista Software, "Dynamic Power Management for Embedded Systems", November, 2002