프레임버퍼를 이용한 디바이스 통합 제어 모듈의 설계 및 구현

송영호* 성재용 이철희 김대영 이호근 권택근**
(주)한백전자기술연구소, 충남대학교 컴퓨터공학과
{yhsong, young, chlee, dykim, hglee}@hanback.co.kr tgkwon@ce.cnu.ac.kr

The Design and Implementation of Device Control Module by Embedded System's Framebuffer

Youngho Song[0] Jae-Young Sung Cheul-Hee Lee Dae-Young Kim Ho-Geun Lee Taeck-Geun Kwon
Hanback Electronics Technical Research Institute, Dept. of Computer Engineering, Chungnam National Univ.

요 약

최근 IC 설계 및 제조 기술의 급격한 진보에 따라 PC 보급이 포화상태에 이르면서 가전제품으로서 첨단을 자랑하던 PC가 일반 가정에서 TV의 보급률과 비슷해지며 대중화에 되어가고 있다. 이에 따라 임베디드 시스템 기술이 PC뿐 아니라 정보 가전, 정보 단말, 통신 장비, 항공, 군사, 물류, 금융, 차량, 교통, 사무, 산업, 제어, 의료 등 전 분야에 걸쳐 다양하게 적용되고 있는 실정이다. 본 논문에서는 이러한 임베디드 시스템에서의 여러 디바이스를 효율적으로 관리하기 위한 한 가지 기법으로 프레임버퍼를 이용한 디바이스 통합 제어 모듈을 설계 및 구현하였다. 본 논문에서 제안하는 기법은 임베디드 시스템에서 사용자 인터페이스로만 부각되던 LCD 화면을 통해서 임베디드 시스템의 각종 디바이스를 진단하고 제어할 수 있게 하여 특별한 외부 장치 없이 곧바로 디버깅이 가능하도록 하였다.

1. 서 론

본 논문은 임베디드 시스템의 각 디바이스 모듈을 어떻게 제어 하고 효율적으로 관리 할 수 있는지에 대한 기법으로 프레임버퍼를 이용한 디바이스 통합관리 기법을 제안하고 있다. 임베디드 시스템 개발 과정은 하드웨어 설계 및 개발 과정, 부트로더 개발 과정, 커널 포팅 과정, 응용 프로그램 개발 과정으로 크게 나눌 수 있다.[1,2,3,7] 이때 각 디바이스에 대한 검증은 하드웨어 설계 와 부트로더 개발 과정에서 이루어지고 커널이 포팅 된 이후에는 각 디바이스가 하드웨어적으로 이상 없이 동작한다는 가정 하에 각 디바이스 드라이버를 작성하여 응용프로그램에서 해당 디바이스를 사용할 수 있도록 한다. 개발이 완료된 임베디드 시스템의 상태 정보는 콘솔(RS232)등을 통하여 확인하거나 이더넷(RJ45)을 통한 접근으로 이루어질 수 있다[2,7]. 하지만 전원 외에 어떠한 외부 접근이 허락하지 않는 경우(대부분의 임베디드 시스템의 환경) 임베디드 시스템의 CPU, Flash, SDRAM등의 디바이스 상태를 파악할 수 있는 어떠한 방법도 없게 된다[1,3]. 본 논문의 2장에서는 이러한 문제를 해결하기 위한 방법으로 기존 응용프로그램에서 사용자 인터페이스로 사용되는 그래픽 LCD를 통해서 특별한 응용프로그램을 구동시키지 않고 모든 디바이스에 대한 상태를 확인하거나 제어할 수 있는 방법으로 제시된 프레임 버퍼에 대해 알아본다[8]. 3장에서는 제시된 기법을 실험 하기위한 플랫폼을 설계 및 개발하고, 4장에서는 논의된 프레임버퍼를 통해서 디바이스를 제어하고 정보를 획득하는 모듈을 설계 및 구현 하도록 한다. 5장에서는 결론 및 향후연구 과제에 대한 논의로 끝을 맺는다.

2. 관련연구

프레임버퍼란 리눅스 시스템에서 그래픽을 표현할 수 있는 하드웨어를 말한다[8]. 임베디드 시스템에서 호스트로 불리는 사용자 컴퓨터 라면 그래픽 카드가 될 것이고, 타겟으로 불리는 임베디드 시스템 이라면 LCD 콘트롤러를 프레임버퍼 장치라고 말 할 수 있다.[1,2] 그리고 그 프레임 장치를 응용 프로그램에서 제어할 수 있도록 만들어진 디바이스 드라이버를 프레임버퍼 드라이버라고 한다. 이 프레임버퍼 디바이스 드라이버는 응용 프로그래머가 코딩을 할 수 있어야 하므로 어떠한 표준화된 인터페이스를 가지고 있다. 이러한 프레임버퍼 디바이스 드라이버를 사용하여 개발 보드의 그래픽 LCD에 그림을 나타낼 수 있으며 여기에 각 모듈을 X, Y 좌표로 메칭시켜 디바이스를 제어 할 수 있게 되는 것이다. 한 픽셀의 데이터는 Red, Green, Blue(RGB)의 값을 가지고 있어야 한다. 일반적으로 TRUE 칼라라고 하면 R, G, B 각각 한 바이트 씩 모두 3바이트를 가지고 있어서 16581375(255*255*255)개의 칼라 수를 표현 할 수 있다. 여기서 각각의 바이트가 0이면 즉 Red=0,Green=0, Blue=0인 경우는 검정 색이 되고 모두 maximum 값을 가지고 있으면 Red=255, Green=255, Green=255으로 하얀색 점이 된다. 본 논문에서 제안된 임베디드 시스템에서는 bits_per_pixel값이 16이므로 위와 같이 표현되지 않

고 다음과 같이 표현된다. 아래그림은 16bpp에서 한 픽셀과 RGB값을 표현하는 일반적인 방법을 보여 주고 있다.

RED(5bit)	Green(6bit)	Blue(5bit)

그림 2.1 픽셀의 RGB 데이터 표현

즉 MSB, LSB Red와 Blue는 각각 5비트씩 이고 G는 6비트 정보를 가지게 된다. 그리고 MSB(Most Significant Bit)에서 LSB(Least Significant Bit)쪽으로 R, G, B 순으로 저장된다. 그러므로 색상을 표현하는 R(0~255), G(0~255), B(0~255)를 직접 사용하지 못하고 변환 함수를 이용하여 R ,G, B의 3byte값을 2byte로 고쳐주어야 한다. 그래픽 LCD에 디바이스 제어 모듈을 위한 그림을 디스플레이하고 디스플레이되는 X, Y좌표에 해당하는 디바이스를 할당한다. 각 디바이스에 해당하는 제어 모듈에 사용자 입력이 있으면 Data/Control을 이용해 해당 Device에 대한 정보를 요청하거나 변경시킬 수 있게 된다.

3. 임베디드 시스템 설계 및 구현

타켓이 될 임베디드 시스템을 그림 3.1과 같은 모듈을 장착한 시스템으로 구성하였다.

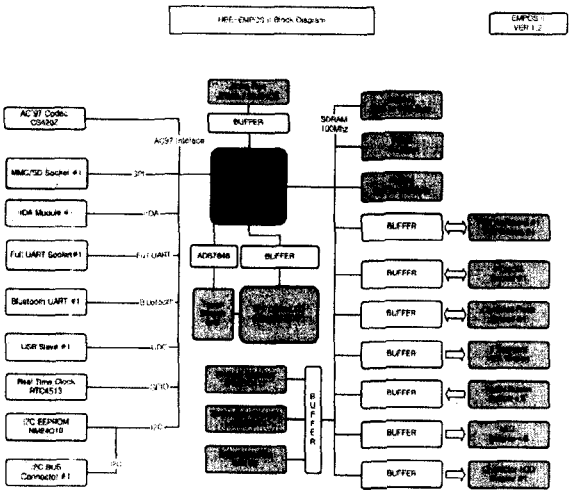

그림 3.1 전체 시스템 구성도

프로세서는 인텔의 Xscale(PXA255)을 사용하였고 터치 스크린 콘트롤러와 터치 스크린은 가각 Burr Brown ADS7846 터치 스크린 콘트롤러와 LG-Philips사의 LB064V02 640*480 TFT LCD 터치스크린을 장착하였다. Burr Brown ADS7846 터치스크린 콘트롤러는 프로세서의 SSPC (synchronous Serial Port Controller)를 통하여 통신하며, 모토롤라의 Serial Peripheral Interface (SPI) 프로토콜을 사용한다[2,4,5]. 아래 그림은 디바이스 통합 제어 모듈의 UI (User Interface)로 사용될 LCD Panel의 회로 구성도를 도식하한 그림이다.

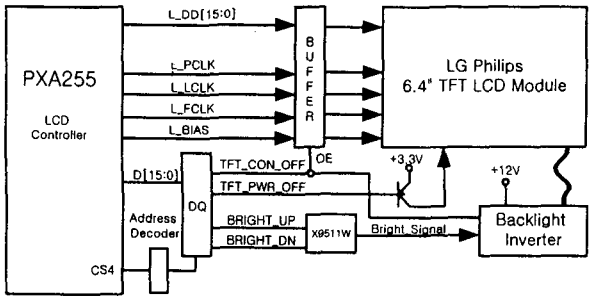

그림 3.2 LCD 모듈 회로 구성도

위와 같이 LCD 터치스크린을 가진 임베디드 시스템을 구성하고 프레임버퍼를 구동할 수 있도록 Linux 커널을 포팅하였다. Linux 커널은 2.4.19를 기반으로 하였고 arm패치, pxa255 패치 등을 가하여 타겟 시스템에서 구동되도록 포팅하였다. 부트로더는 blob을 기반으로 순수 로더의 기능 중 커널 부팅 기능을 최우선으로 하여 본 시스템의 구성에 맞게 재 구성하였다. 파일시스템은 16M RAMDISK를 구성하여 포팅하였으며 디바이스 통합 제어 모듈에 대한 이벤트나 변경사항을 지속적으로 관리할 수 있도록 MTD(Memory Technology Device)를 사용하여 Flash의 일정영역을 할당하여 Log를 기록할 수 있도록 하였으며 정기적으로 Log를 삭제하여 메모리을 효율성을 높였다.

4. 프레임버퍼 디바이스 제어 모듈 설계 및 구현

구성된 임베디드 시스템은 400Mhz Xscale core를 장착한 시스템으로 리눅스 2.4.19를 기반으로 구성되었다. 구성된 임베디드 시스템에 그림 4.1과 같이 프레임 버퍼를 이용한 디바이스 통합 제어 모듈을 설계 하였다.

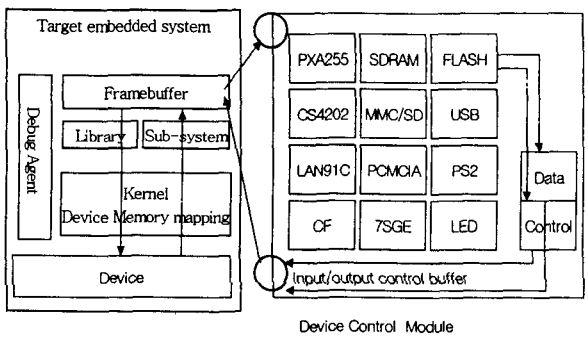

그림 4.1 Device Control Module의 Data/Control flow

전원 인가 시 PXA255 Process의 CS0(Chip Select 0)에 Flash 메모리가 장착되어 있으므로 곧바로 부트로더를

통해서 커널로 부팅할 수 있게 되고 커널의 INIT 프로세서를 통해서 device control module이 활성화 된다. 디바이스 통합 모듈은 LCD 터치스크린의 입력을 무한루프를 돌면서 기다리게 된다.

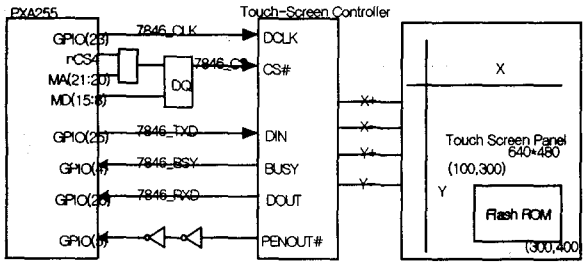

그림 4.2 Touch Screen Controllers

그림 4.2에서와 같이 터치 스크린에 사용자 입력이 있을 경우 X, Y축의 위치를 그림 4.3과 같이 판별하여 해당 디바이스를 선택하게 되면 제어 모듈에 의해서 모든 디바이스들은 제어권을 제어 모듈에 넘기고 DATA부와 Control부를 통해 해당 디바이스의 상태를 감시 하거나 변경할 수 있다.

```
if(300 < tsval.y && tsval.y < 400)
{
 if(100 < tsval.x && tsval.x < 300){
  if(!tsval.pressure){
      flash_get_info('1', data,);
      set_flash_info('1', va,contl);
      g_passwd_data[4 - g_pos] = '1';
      g_pos--;
  }
}.....
```

그림 4.3 디바이스 통합 제어 모듈에서 디바이스 선택

만약 X, Y 축의 Flash ROM이 선택된 경우 다음과 같은 정보들을 확인할 수 있다.

```
* Memory Map *
bootloader) ROM : 0x00000000, RAM : 0xA3D00000, LENGTH : 0x0000C000
kernel)     ROM : 0x000C0000, RAM : 0xA00C0000, LENGTH : 0x00100000
ramdisk)    ROM : 0x00200000, RAM : 0xA0600000, LENGTH : 0x00600000
usr)      ROM : 0x00900000, RAM : 0xA1600000, LENGTH : 0x01700000
* Registers Value *
MDCNFG  address : 0x48000000    value : 0x00001A
MDREFR  address : 0x48000004    value : 0x000BC018
MSC0    address : 0x48000008    value : 0x7FF42BF0
MECR    address : 0x48000014    value : 0x00000001
SXCNFG  address : 0x4800001C    value : 0x00040004
```

그림 4.4 Flash 메모리의 정보 디스플레이

최종 구현된 프레임버퍼 디바이스 통합 제어 모듈은 아래 그램과 같이 전원만 인가되면 사용자 UI를 통해서 LCD에 디스플레이 되며 어떠한 외부 장치(콘솔이나 이더넷)없이 바로 시스템을 확인 및 제어 할 수 있게 된다.

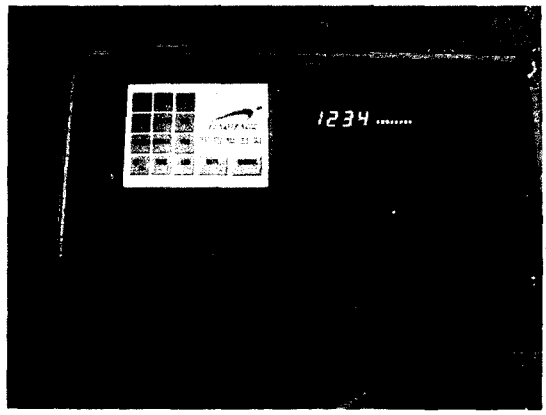

그림 4.5 프래임버퍼를 이용한 디바이스 통합 제어 모듈 임베드디 시스템

5. 결론 및 향후 연구 과제

지금까지 임베디드 시스템에서 프레임버퍼를 이용한 디바이스 통합 제어 모듈을 설계하고 구현하였다. 본 논문의 제안으로 사용자는 임베디드 시스템에서 특별한 외부 장치 없이 바로 시스템을 확인할 수 있게 되었으며 이러한 기법을 통해 LCD를 가진 모든 가전이나 특수 장치에서 디바이스를 관리하는 기법으로 활용될 수 있을 것이다. 특히 PDA나 핸드폰 같은 디바이스의 스크린 제어를 통해서 문제발생 시 사용자 암호만으로 시스템을 진단하거나 수리 할 수 있을 것이다. 앞으로 지금 까지 논의된 디바이스 통합제어 방식을 좀더 확장하는 연구가 진행되어 디바이스 모듈별 인증단계를 두고 디바이스의 중요 설정값 등을 변경 시 계층적 인증을 통해서 값의 변경이 이루어진다면 좀더 안전하고 편리한 시스템을 구현 할 수 있을 것이다.

6. 참고문헌

[1] Rajesh K, Gupta, "*Introduction to Embedded System*", ICS 212, 2002 Winter Workshop

[2] 한백전자기술연구소 *"임베디드 리눅스 시스템 HEB-EMPOSII"*, 2004.03 ver1.0

[3] Pillip J. Koopman, Jr "*Embedded System Design Issues(the Rest of the Story)*", Proc. of the Internationl Conf. on Computer Design(ICCD96)

[4] *Intel PXA255 Processor Developer's Manual*

[5] *ARM Architecture Reference Manual*

[6] 김선자, 김홍남, 김채규, " *임베디드 운영체제 표준화 동향*" , 2002 정보처리학회 제9권 제1호

[7] 류승범 " *임베디드 실시간 시스템 개발 교육 과정*" , 2002 정보처리학회 제9권 제1호

[8] *http://kelp.or.kr/korweblog/framebuffer*