실시간 운영체제를 위한 그래픽 윈도우 시스템의 설계 및 구현

강희성[0], 이정원, 조문행, 이철훈
충남대학교 컴퓨터공학과,
{hskang[0], jwlee, mhcho, chlee}@ce.cnu.ac.kr

Design and Implementation of Graphic Windows System for Real-Time Operating Systems

Hui-Sung Kang[0], Jung-Won Lee, Moon-Haeng Cho, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National University*

요 약

그래픽 윈도우 시스템은 가장 널리 사용되는 GUI 종류중 하나이며, 그 사용 범위가 임베디드 시스템에 이르기까지 확대되고 있는 추세이다. 그래픽 윈도우 시스템은 임베디드 시스템의 요구조건을 만족시키기 위해 경량이고 자원의 소모를 작게 해야 하며 고성능이어야 한다. 또한 높은 신뢰성을 가져야 하며 목적에 따라서 재구성이 가능해야 한다. 본 논문에서는 실시간 운영체제를 위한 그래픽 윈도우 시스템에서 윈도우 관리, 기본적인 그래픽 호출, 텍스트 출력, 이미지 출력 부분을 설계 및 구현하였다.

1. 서 론

과거의 임베디드 시스템(Embedded System)은 사용자 인터페이스(User Interface)가 아주 간단한 텍스트 기반이었고 비용문제와 하드웨어적인 제약으로 인해서 흑백 LCD 정도만 제공하였다. 하지만 최근에는 기술발전에 힘입어 임베디드 시스템도 화려한 사용자 인터페이스를 제공하기 위해 GUI 를 제공하고 있고 상당히 큰 사이즈의 TFT-LCD 화면을 장착한 임베디드 시스템도 많이 보급되고 있다.

이러한 GUI 를 지원하기 위해서 가장 많이 사용되는 것이 그래픽 윈도우 시스템이다. 그래픽 윈도우 시스템(Graphic Window System)은 화면을 여러 사각형 영역으로 나누어 각각 다른 응용 프로그램이 제어함으로써 독립적인 입/출력 장치처럼 동작할 수 있게 하는 소프트웨어이다.

임베디드 시스템은 그 활용 목적이 미리 정해져 있고, 하드웨어의 제약이 존재한다. 따라서 임베디드 시스템에서 쓰이는 사용자 인터페이스로써 그래픽 윈도우 시스템은 경량화, 최적화, 실시간성 지원, 플랫폼 다양성 지원 등이 필수적이다. 특히 그래픽 윈도우 시스템은 기능들의 특성상 메모리를 많이 소모하고 수행시간이 길다. 그러므로 제한된 마이크로프로세서의 성능과 작은 용량의 메모리를 가진 임베디드 시스템에서 작고 가벼우면서 수행속도가 빠른 그래픽 윈도우 시스템을 지원해야 한다.

최근의 임베디드 시스템 분야에서 실시간 운영체제를 탑재하여 개발된 제품들이 점차 증가하고 있는 추세이고, 네트워크나 멀티미디어 장비와 같이 시스템에서 처리해 주어야 하는 일의 양이 점점 많아지는 상황에서 실시간 운영체제의 탑재는 너무나 당연한 것으로 인식 되고 있다.

본 논문은 임베디드 시스템에 적합한 경량 그래픽 윈도우 시스템을 설계하고 구현한 내용을 기술한다. 2 장에서는 관련연구에 대해 살펴보고, 3 장에서는 실시간 운영체제용 그래픽 윈도우 시스템에 대한 설계 및 구현을 다룬다. 4 장에서는 구현된 그래픽 윈도우 시스템의 동작 및 실험결과를, 마지막으로 5 장에서는 결론 및 향후 연구과제에 대해 기술한다.

2. 관련 연구

2.1 그래픽 윈도우 시스템의 개요

GUI(Graphic User Interface)는 그래픽을 통해 사용자와 컴퓨터간 인터페이스 역할을 한다. 윈도우는 GUI 의 한 부분으로, 여러 개의 투시창을 허용하는 시스템에서 화면 위에 나타난 별개의 투시 창을 의미한다. 각 윈도우 영역을 다른 응용 프로그램의 제어 하에 둠으로써 독립적인 입/출력 장치처럼 동작하게 할 수 있다. 또한 사용자가 여러 프로세스 결과를 한번에 볼 수 있도록 만드는 동시에, 마우스로 윈도우를 선택해 입력 받을 프로세스를 지정할 수도 있게 한다.

윈도우 시스템이 갖추어야 할 일반 기능으로는 다중 윈도우 환경, 클립보드, 사용자 인터페이스를 위한 다양한 컨트롤(Widget), 다양한 글꼴, 그래픽, 멀티미디어, 마우스, 객체모델 등이 있는데, 임베디드 시스템에서 윈도우 시스템을 적용할 때는 사용 목적에 따라 필요한 기능들만 모듈화해서 사용하고 있다.

2.2. 임베디드 시스템에서 GUI 기술 분류 및 종류

임베디드 시스템은 다양한 목적을 가지고 있기 때문

* 본 논문은 국방과학연구소의 실시간 운영체제 인터페이스용 미들웨어 연구과제의 수행결과임.

에 GUI 기술 또한 그 분류에 따라서 특징과 구현사항이 다르게 적용된다. GUI 관점에서 가장 중요한 분류는 해상도에 따른 분류로 아래와 같다.

[표 2-1] GUI 의 해상도에 따른 기술분류

구 분	해상도	특징
Full Screen 형 (휴대폰, TV 형)	176*192	간단한 아이콘, 저전력
	640*480	페이지단위, 큰아이콘, 폰트
세미 Full Screen 형 (PDA 형)	240*320	창크기조절, 펜입력, 윈도우 형태의 Full Screen
PC 윈도우형	1024*768	일반적인 윈도우 환경

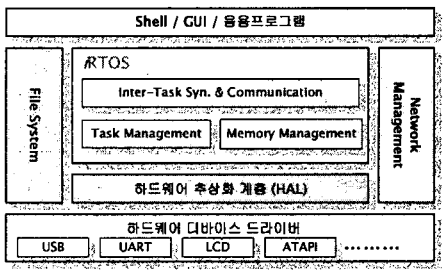
상용 실시간 운영체제와 임베디드 리눅스에서 사용되고 있는 GUI 들은 아래와 같다.

[표 2-2] 상용 / 임베디드 리눅스에서 사용되는 GUI 종류

상용 실시간 운영체제	임베디드 리눅스
G-Window and G-View	MicroWindows
MAUI	PicoTK and PicoGui
OpTIC	DirectFB
WinLight	Qt/Embedded

2.3. *i*RTOS™

본 논문에서 윈도우 시스템을 구현하기 위한 기반으로 사용한 *i*RTOS™ 는 우선순위 기반 선점형 멀티 쓰레드 실시간 운영체제이다. 즉 실시간 운영체제 커널과 응용 프로그램이 통합되어 하나의 큰 프로그램으로 동작하는 구조로써 공통의 메모리 영역을 자유롭게 접근할 수 있고 사이즈는 약 24Kbyte 정도이다.



[그림 2-1] *i*RTOS™ 의 기능 블록 다이어그램

[그림 2-1]은 *i*RTOS™ 의 기능을 블록 다이어그램으로 나타낸 것으로, 운영체제에서 제공되는 기능은 태스크 관리, 태스크간 동기화 / 통신, 동적 메모리 관리 등이 있다.

3. 설계 및 구현

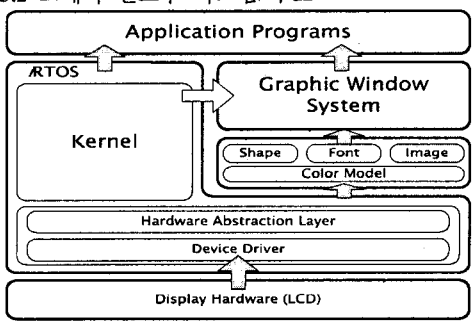
3.1 그래픽 윈도우 시스템 구현

본 논문에서 임베디드 시스템을 위한 그래픽 윈도우 시스템을 설계하면서 고려한 사항은 다음과 같다.

- 경량이고 자원의 소모를 작게 해야 한다.
- 고성능이어야 한다.
- 높은 신뢰성을 가져야 한다.
- 목적에 따라 재구성이 가능해야 한다.

그리고 파일 시스템이 없는 임베디드 시스템의 경우 폰트 이미지(Font Image), 그림 파일 등이 메모리상에 존재하여야 하기 때문에 메모리 공간이 부족하게 된다. 이러한 단점을 보완하기 위해 폰트 이미지에 대해서는 메모리를 절약할 수 있는 메커니즘[1]을 사용하였다. 또한 타겟 보드가 바뀌었을 경우 동일한 기능을 쉽게 이식시키기 위해서 Layered Architecture Design 방식을 사용한다. 즉 하드웨어와 관련 있는 기본기능은 맨 하위에서 하드웨어와 근접하여 존재하고, 하드웨어와 관련 없는 부분은 하드웨어와 관련 있는 부분에서 제공하는 API 를 이용하여 직접 하드웨어를 제어하지 않게 함으로써 하드웨어의 이식성을 높여주도록 설계하였다.

3.2 그래픽 윈도우 시스템 구조

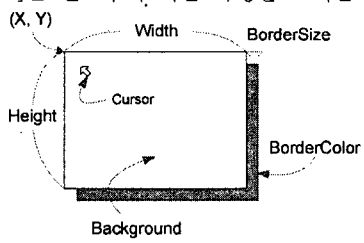


[그림 3-1] 구현한 실시간 운영체제의 그래픽 윈도우 시스템

[그림 3-1]은 본 논문에서 구현한 그래픽 윈도우 시스템으로 LCD 를 위한 디바이스 드라이버, 기본 그리기, 폰트, 이미지를 위한 그래픽 프리미티브와 이를 이용한 그래픽 윈도우 시스템을 Layered Architecture Design 방식으로 설계한 것이다.

3.3 윈도우 설계 및 구현

[그림 3-2]은 윈도우의 기본 속성을 보여준다



[그림 3-2] 윈도우 속성

아래는 윈도우를 관리하기 위한 함수들이다

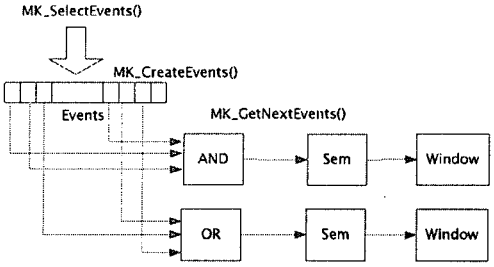
- MK_CreateNewWindow : 새로운 윈도우 생성
- MK_DestroyWindow : 윈도우와 자식 윈도우 삭제
- MK_MapWindow : 화면상에 보이게 함
- MK_UnmapWindow : 화면상에서 보이지 않게 함
- MK_RaiseWindow : 형제 윈도우중 가장 높은 수준으로
- MK_LowerWindow : 형제 윈도우중 가장 낮은 수준으로
- MK_MoveWindow : 윈도우를 이동
- MK_ResizeWindow : 윈도우 크기를 변경
- MK_ClearWindow : 윈도우를 새로 그림

3.4 이벤트(event)

이벤트는 마우스, 키보드 상태의 비동기적인 변화를 그래픽 윈도우 시스템에 전달하는 방법이다.

이벤트는 탑-다운 방식으로 구현되어 이벤트를 기다리다가 이벤트가 발생하면 이벤트 타입과 특정 윈도우를 찾아 전달하는 방식을 사용한다.

[그림 3-3]은 이벤트 처리 구조를 나타낸 것이다.



[그림 3-3] 이벤트 처리

이벤트 관리를 위한 함수는 아래와 같다.

- MK_SelectEvents : 특정 윈도우를 위한 이벤트 선택
- MK_GetNextEvents : 마스크된 이벤트 발생을 기다림

3.5 그래픽 문맥 (graphic context)

그래픽 함수 호출시 많은 인자는 프로세스의 성능저하를 초래하는 원인이 되는데, 본 논문에서는 다양한 부가 인자를 하나로 대체, 그래픽 문맥개념을 사용하여 구현하였다.

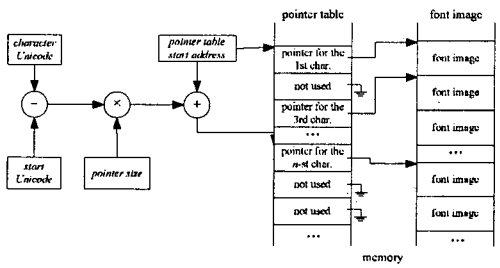
예:
```
DrawLine(window, x1,y1,x2,y2,width,style,operation)
DrawLine(window, gc, x1,y1,x2,y2)
```

3.6 그래픽 호출

그래픽 호출은 지정된 윈도우 영역에 그림이나 텍스트를 그려주는 함수들을 말한다. 각각 자신의 영역에만 그릴 수 있게 하기 위해서 MK_DRAWABLE 구조체를 사용하여 영역을 구분하고 기본적으로 점, 선, 사각형, 원, 타원, 다각형 그리기 함수를 구현하였다.

3.7 폰트 / 이미지

파일 시스템이 없는 임베디드 시스템에서 폰트와 이미지를 화면상에 그리기 위해서 폰트와 이미지를 C 파일에 배열 형태로 포함시켜 메모리에 위치시키는 방법을 사용하였다.



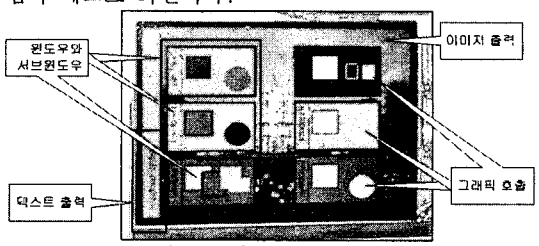
[그림 3-4] 메모리를 절약하는 폰트 출력 알고리즘

그림 [3-4]는 필요 없는 문자의 font image 를 메모리 상에서 제외시키고 중간에 pointer table 인 indir-ect map 을 두어 font image 의 크기를 절약하는 알고리즘이다.

4. 테스트 환경 및 결과

테스트는 SMDK2400 (CPU : ARM920T) 보드와 실시간 운영체제 *i*RTOS™ 기반으로 하였다.

[그림 4-1]는 본 논문에서 구현된 그래픽 윈도우 시스템의 테스트 화면이다.



[그림 4-1] 테스트 결과 화면

본 논문에서 구현한 것과 QNX/Photon, Microwindows, QT/Embedded 와의 실행이미지 크기비교는 아래와 같다

[표 4-2] 실행 이미지 크기 및 기능 비교

구분	구현그래픽 윈도우 시스템	QNX/Photon	Microwindow	QT/Embedded
API	Nano-X	X	Wind32	Nano-X,win32
이미지 크기	83K	70K	600K	1.5MB
플랫폼	ARM	ARM, PPC	X86,ARM,PPC	X86,ARM

또한 본 논문에서 구현한 그래픽 윈도우 시스템에서 내부적으로 사용하는 프로세서의 점유율을 아래와 같은 방식으로 측정한 결과 약 5%정도로 나타났다.

$$\text{CPU Useage} = 100 \times \left(1 - \frac{\text{MK\_IdleCtr}}{\text{MK\_MaxCtr}}\right)$$

[그림 4-2] 프로세서의 이용률 계산 방법

5. 결론 및 향후 과제

본 논문에서는 실시간 운영체제 *i*RTOS™ 에서 동작하는 그래픽 윈도우 시스템을 설계 및 구현하였다. 윈도우, 이벤트 처리, 그래픽 문맥, 그래픽 호출을 통한 기본 그리기 함수, 폰트와 이미지 파일의 출력 등의 기능을 제공하고 있으며, Nano-X 와 호환 가능한 API 를 제공함으로써 응용 프로그램 작성시 편리성을 제공하고 있다. 실행 이미지 크기는 83KB 정도로 다른 그래픽 윈도우 시스템보다 작은 크기를 가짐으로 제한된 메모리 크기를 갖는 임베디드 시스템에 적당하다. 향후 과제로는 미려한 외관을 위한 다양한 위젯의 구현과 파일 시스템이 존재하는 임베디드 시스템에서 폰트 로딩기법을 설계하는 것이다.

참고문헌

[1] 윤기현, 김용희, 박희상, 이철훈, " Design of Graphic User Interface for the Real Time Operating System" , 한국정보과학회, Vol.29,No2(I),pp400-402,2002

[2] Robert W.Scheifler & James Gettys, " X Window System 2nd Edition" , Digital press, 1990

[3] http://microwindows.org