Journal of the Industrial Technology Research Institute,
Korea Maritime University, Vol. 11, 1993.

탱크레벨제어를 위한 멀티타스크 통합환경시스템의 개발에 관한 연구

정 재 학* · 류 길 수**

A Study on the Development of Multi-task Integrated Environment System for Controlling Tank Level

Chung Jae - Hak, Rhyu Keel - Soo

Abstract

During the real-time control performance, in general, the system must be executed by multi-task environment in order not to change the control characteristics while performing the supporting function simultaneously. Also the event driven multi-task method must be used to minimize the task switching time.

In this paper, a REal-time Level control Integrated environment System(RELIS) is introduced in controlling tank levels. The system uses both text mode and graphic mode of video adapter for the efficient performance of monitor display. Also in the system, the above mentioned multi-task method is implemented and a 8253 chip is contolled by the interrupt routine to adjust the sampling time at delicate interval.

With the developed system, modifications of the task and insertion of the new tasks can be done easily since tasks are constructed without considering effective use of CPU.

* 한국해양대학교 해사대학 서무과
** 한국해양대학교 이공대학 제어계측공학과

1. 서 론

지금까지 제어알고리즘을 실현시키기 위해 개발되어 있는 시스템들의 대부분은 통합환경을 고려하지 않고 특수목적의 제어대상에만 적합하도록 개발되어 있거나, 실시간 제어만을 수행하도록 개발되어 있다[1)~3)]. 이런 까닭에 운전중 특성이 크게 바뀌는 대상시스템에서는 제어특성을 변화시키지 않도록 수행하는 것이 극히 곤란하며, 운전중 대상시스템의 파라미터 변경등을 자동 또는 수동으로 수행할 수 있는 기능이 요구된다. 이러한 기능을 용이하게 수행하기 위해서는 시스템을 멀티타스크 환경으로 구축하는 것이 필수적이라 할 수 있다.

UNIX와 같은 운영체제에서는 이러한 멀티타스크 환경이 구축되어 있으며 타임슬라이스 기법에 의한 서비스의 공평성과 타스크간의 절환 시간의 최소화에 가장 큰 비중을 두고 있다. PC의 경우 종래에는 하드웨어적인 제약점 때문에 멀티타스크 환경을 구축한다는 것이 극히 어려울 뿐만 아니라 이용 효율도 나쁘기 때문에 도외시 되어 왔지만 최근에 이르러서는 하드웨어의 비약적인 발전으로 MS-WINDOWS와 같은 멀티타스크기능을 가진 운영체제가 등장하기 시작하고 있다.

한편 실시간 제어를 수행해야 하는 탱크레벨제어와 같은 시스템에서는 멀티타스크 환경을 구축하는 경우 제어 알고리즘의 특성을 변화시켜서는 안된다. 이와 같은 시스템에서 가장 고려해야 할 사항은 꼭 필요한 타스크만이 기동되도록 하는 이벤트 구동형 멀티타스크 기법을 이용함과 동시에 타스크 절환시간이 최소가 되도록 해야 하는 점이다.

본 논문에서는 이상과 같은 점을 고려하여 멀티타스크 기능을 가진 실시간 레벨제어 통합환경 시스템(REal-time Level control Integrated environment System : RELIS)을 구축하기로 한다. 즉 실시간 제어를 수행하면서 제어의 일시중지 타스크, 샘플링 타임의 변경 타스크, 목표치의 변경 타스크, 각종 피드백값의 변경 타스크들로 절환시키는 부분을 멀티타스크 환경으로 구현하며, 타스크 절환시간을 최소화 할 수 있도록 이벤트 구동형 멀티타스크 기법을 도입한다. 실제 시스템은 PC 상에서 A-D/D-A 변환기를 통하여 탱크레벨을 제어하도록 C언어로 구현되어 있으며, 풀-다운메뉴 방식으로 통합환경이 구축되어 있고, 탱크레벨을 제어하기 위해 적응제어 알고리즘[4)]을 이용하고 있다.

2. 탱크레벨 제어 시스템의 개발 환경

2.1 탱크레벨 제어 시스템의 개요

범용성을 고려하여 탱크레벨제어 시스템의 알고리즘을 컴퓨터상에 구현시키기 위해서는 시스템에 다음과 같은 기능이 요구된다.

① 시뮬레이션 및 실시간 제어 기능

② 실시간 데이타의 입출력 기능

③ 운전중 파라미터의 자동변경에 의한 적응 기능

dCollection

④ 운전중 샘플링 타임의 변경에 의한 제어성능의 조절기능

⑤ 각종 데이타의 정밀 표시 기능

⑥ 통합환경 기능

이런 기능을 가지는 시스템을 구현하는데 있어서 고려해야 할 점은 다음과 같다.

첫째, 시스템의 실행효율을 높이기 위해서 메뉴의 구현과 데이타의 디스플레이등은 텍스트 모드에서, 정밀한 결과값이 요구되는 제어결과 및 추이등의 출력은 그래픽 모드에서 동작되도록 필요에 따라 모드변경을 자유롭게 할 필요가 있다.

둘째, 통합환경 시스템을 구현하는데 있어서는 대화식 환경을 제공해야하므로 풀-다운 메뉴방식을 이용한 스크린 제어 기법으로 시스템을 구성하는 것이 바람직하다.

셋째, 멀티타스크를 구현할 경우 각 타스크간의 문맥교환 시간의 최소화, 타스크 절환의 공평성 등 고려해야 될 사항이 많이 있지만 실시간 제어 알고리즘의 경우에 이론상의 제어특성을 변경시키지 않고 수행하기 위해서는 문맥교환 시간의 최소화를 가장 중요시한 멀티타스크 환경을 구축해야 한다. 이를 위해서는 필요시에만 타스크를 기동시키는 이벤트 구동형 멀티타스크 기법을 이용하는 것이 바람직하다.

2.2 멀티타스크 환경

멀티타스크를 구현하는데 있어서는 타임 슬라이스(time slice)방식에 의한 멀티타스크와 이벤트 구동형(event driven)의 멀티타스크가 있다.

타임슬라이스 방식에 의한 멀티타스크는 그림 2.1과 같이 CPU의 점유시간을 일정시간으로 나누어 타스크를 실행시키는 방식으로 타이머를 사용한 인터럽트기법을 이용하여 구현할 수 있다. 즉 어떤 타스크가 절환될 때 마다 그 타스크가 절환되기 전의 레지스터 및 각 변수들의 값을 보존해 두고 다음의 타스크가 필요한 환경을 설정해서 실행해 나가는 방법이다. 이 방식에서는 외부에서 어떤 이벤트가 특별히 실행요구를 하지 않더라도 우선도 및 스케쥴링 기법등에 의해 타스크 절환이 이루어 진다. 따라서 경우에 따라서는 필요없는 절환이 발생하기도 하고, MS-DOS 레벨보다 더욱 하위부분에 시스템을 부가하지 않으면 안된다. 더욱이 이 방법에서는 어떤 타스크를 실행하고 있는 도중이라 할지라도 그 타스크의 절환이 언제 발생 할 것인가를 알 수 없으므로 프로그램 개발시에 메모리 관리와 그외의 서비스 호출이 특히 어려운 문제로 대두된다. 또한 MS-DOS는 멀티타스크환경이 아니기 때문에 function-call(int21)등의 시스템 호출은 재입가능(reenterent)이 아니며, 따라서 INT 21H를 실행중에 타스크 절환이 일어나 다시 INT 21H를 실행하는 것과 같은 일은 허용되지 않는다.

이벤트 구동형 멀티타스크는 그림 2.2와 같이 상태감시 부분은 항상 상태변화를 감시하고 있고 변화가 일어나면 그 변화에 따라 타스크를 기동한다. 이 방식에서는 상태변화를 감시하는 부분이 있어, 이 부분은 타스크 절환을 임의로 행하기 때문에 타임 슬라이스형일 때 처럼 프로그램상에서 예기치 못한 function-call은 발생되지 않으며, MS-DOS를 직접 수정하지 않고 멀티타스크환경을 실현할 수 있고, 타스크 절환 시간을 짧게 할 수 있는 장점을 가지고 있으나, 외부의 이벤트에 의해 타스크 절환이 이루어지므로 우선순위를 정하거나 스케쥴링 기법을 실현하는데는 많은 제약이 따른다.

Collection

본 논문에서와 같이 측정 데이타를 실시간에 처리하여 제어를 행해야 하는 시스템 환경에서는 그림 2.3과 같이 타스크 절환시간을 최소로 하기 위하여 타스크 절환이 필요한 경우에만 이루어지도록 외부 이벤트에 의한 타스크 절환 기법이 바람직하다. 따라서 전체 시스템을 그림 2.4에서와 같이 제어를 수행하는 타스크와 부가적인 기능을 수행하는 타스크로 분류하여 이 두가지 그룹의 타스크가 병행하여 실행될 수 있도록 타임 슬라이스 방식에 의한 멀티타스크 환경으로 구축하고, 부가적인 기능을 수행하는 타스크들은 이벤트 구동형 방식으로 기동되도록 함으로써 타스크의 절환시간을 최소화할 수 있다.

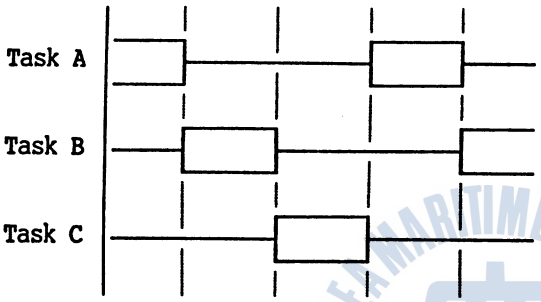


Fig. 2.1 Time slice multi-task method.

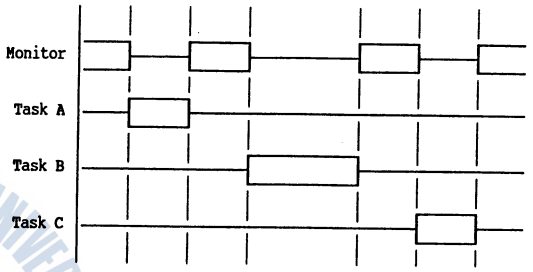


Fig. 2.2 Event driven multi-task method.

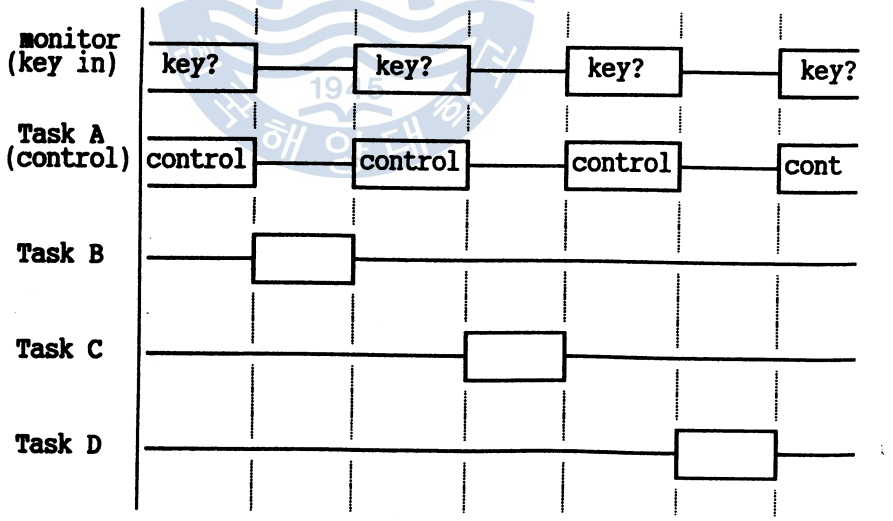


Fig. 2.3 Event driven multi-task method for general control system.

또한 멀티타스크 시스템을 실현하기 위해서는 절환시 마다 기동될 타스크의 기동 위치를 확인할 수 있도록 타스크별로 별도의 메모리를 확보하고 있어야 하며, 이는 스택 영역을 세분화 함으로써 가능해 진다. 즉 일반적인 프로그램에서는 그림 2.5(a)와 같이 스택영역을 SS=DS로 하여 메모리 배열을 사용하고 있지만 멀티타스크 시스템에서는 그림 2.5(b)와 같이 SS=DS인 상태에서 SS≠DS의 상태로 스택 위치를 변경할 수 있도록 프로그램밍 해야 한다. 이와 같은 메모리 관리를 위해서는 다음과 같은 C언어의 구

dCollection

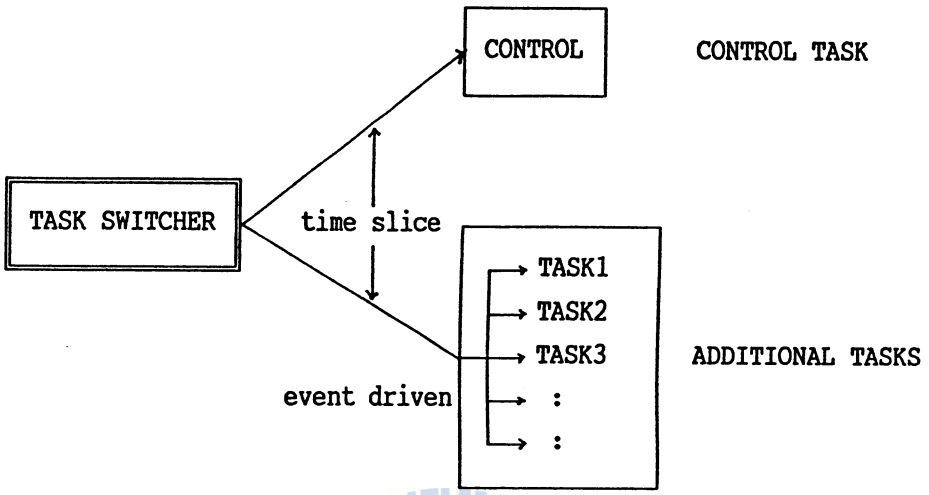
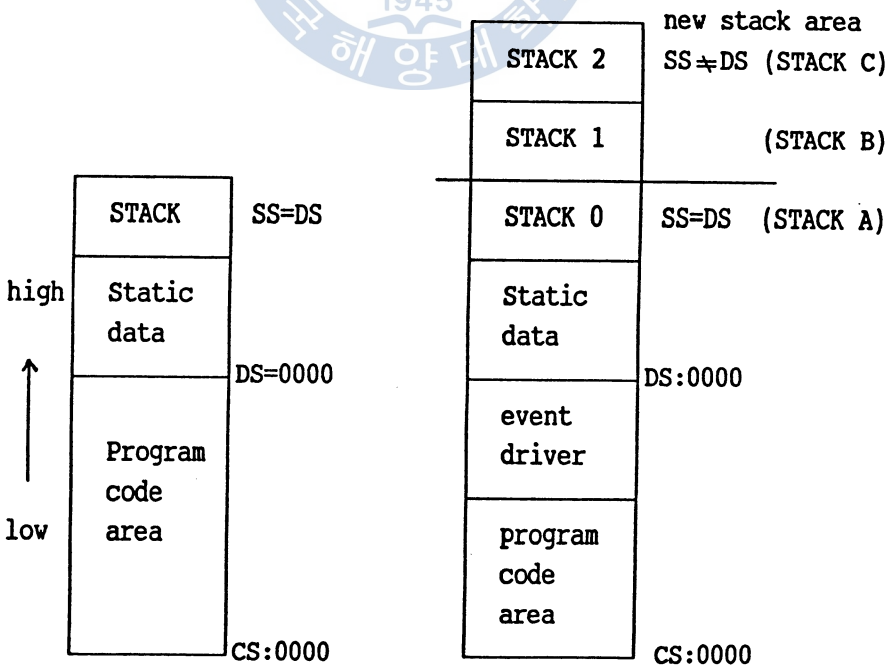


Fig. 2.4 Time slice and event driven multi-task method for controlling tank level.

조체변수를 이용하여 각 타스크마다 이용해야 할 스택영역을 관리하면 된다. 즉 타스크 마다 task stack size, task bottom address, stack top address, stack limit를 가르키는 변수를 가진 task-data라는 타스크 제어변수를 정의해 두고 각 타스크가 호출될 때에는 먼저 이들 변수에 있는 주소로 스택 및 타스크의 실행위치를 옮겨놓고 타스크를 실행하도록 한다.



(a) Normal memory arrange (b) Multi task memory arrange

Fig. 2.5 Comparison of normal memory with multi-task memory.

Collection

```
#define TASK_DATA struct task_data_tag
TASK_DATA {
        unsigned int task_size ;
        char            *task_pointer ;
        char            *stack_pointer ;
        unsigned int STKHQQ ;
        } ;
static TASK_DATA task_data[TASK_MAX] ;
```

3. RELIS 시스템의 구현

3.1 시스템 환경

일반적으로 마이크로컴퓨터와 센서/계측기를 결합하는 경우에 있어서 프로세스 제어용 신호로 4~20mA를 쓰는 이유는 전송선 저항이 오차의 원인이 되지 않으며, 낮은 임피던스로 수신하기 때문에 잡음에 강하고, 전송선의 단선유무를 알 수 있기 때문이다[5]. 센서로 부터 입력을 받고 밸브로출력하기 위한 수단으로서는 A-D/D-A 변환기가 사용되고 있으며, A-D/D-A 변환기의 입출력으로는 DC전압이 이용되고 있는 반면, 센서 및 제어밸브로의 입출력단에서는 4~20mA가 이용되고 있으므로 적절한 변환작업이 필요하다. 또한 본 시스템에서는 IBM-PC내의 8253칩을 인터럽트처리 루틴으로 조작하여 사용함으로써 운전중에 샘플링 타임 변경이 가능함과 동시에 프로그램을 설계하는데도 용이하게 하였다.

1) 시스템의 입출력 신호

식 (3.1)과 같이 표현되는 신호전송에 있어서는 R_o를 R_i에 비하여 충분히 크게 하면 $I=I_o$가 되어 전송선 저항에 관계없이 R_i에 의하여 전류/전압 변환을 할 수 있다.

$$I = \frac{R_o}{r_1 + r_2 + R_i) + R_o} \times I_o \qquad (3.1)$$

단 I_o : 전송기 출력 전류

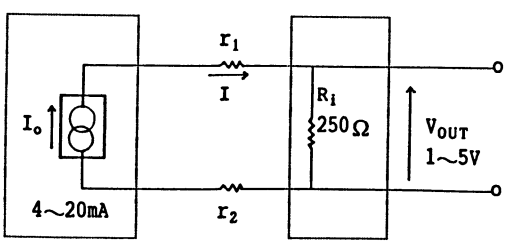
R_o : 전송기 출력 저항

r_1, r_2 : 전송선 저항

R_i : 전류/전압 변환 저항

이다.

그림 3.1은 이러한 전류/전압 변환 회로를 보여주고 있으며, R_i=250Ω을 사용함으로써 입력전류 I_o=4~20mA일 때 출력전압 V_{OUT}=1~5V가 된



〈 Sensor 〉 〈 I/V Conversion resistor 〉

Fig. 3.1 Conversion 4~20mA to 1~5V.

다. 또한 이 회로에서 전송선이 단선된 경우에 출력전압이 0V(정상시에는 1V 이상)로 되기 때문에 A-D 변환값을 읽음으로써 단선검출이 가능해 진다.

2) 8253 타이머

IBM-PC 호환기종에서 제공되는 시스템 클럭(14.3178MHz)에서 얻어지는 4,772,727Hz로부터 1,193,180Hz가 8253의 클럭 입력에 제공된다[6),7)]. 이 8253에 나오는 주파수를 1~65,536으로 나눔에 따라 최고 838.096(nsec)에서 최저 55(msec)의 인터럽트 주파수를 발생시킬 수 있다.

본 시스템에서는 8253에서 나오는 클럭 1,193,180Hz를 1,193으로 나누어줌으로써 1(msec)마다 클럭이 발생되게끔 인터럽트 루틴을 설계하였으며, 이에 대한 알고리즘은 다음과 같다.

step1 : Save OCW1 value(interrupt mask bits) of 8259 interrupt
controller and set interrupt mask register to 0x01.
old_mask=inp(0x21).
outp(0x21 , old_msk | 0x01)

step2 : Output 1,193 to channel 0 of 8253.
outp(0x40,1193)

step3 : Call DOS interrupt function and save current interrupt
vector value.
inregs.h.ah= 0x35
inregs.h.al= 0x08
int86x(0x21, &inregs, &outregs, &segregs)
old_vec=segregs.es≪16 | outregs.x.bx)

step4 : Call DOS interrupt function and set new interrupt vector
value.
inregs.h.ah=0x25
inregs.h.al=0x08
segregs.ds=FP_SEG(new_vec)
inregs.x.ds=FP_OFF(new_vec)
int86x(0x21, &inregs, &outregs, &segregs)

step5 : If interrupt execution is finished, restore to old vector
value saved in step3, otherwise go to step4.

실제 인터럽트 함수내에서는 클럭이 발생할 때 마다 변수를 1씩 증가시키도록 작성하여 두고 이 값이 사용자가 정의한 값에 도달되면 한번씩 샘플링 하도록 하여 둠으로써 샘플링 타임을 1(msec) 간격의 임의 시간으로 조절할 수 있도록 하였다.

Collection

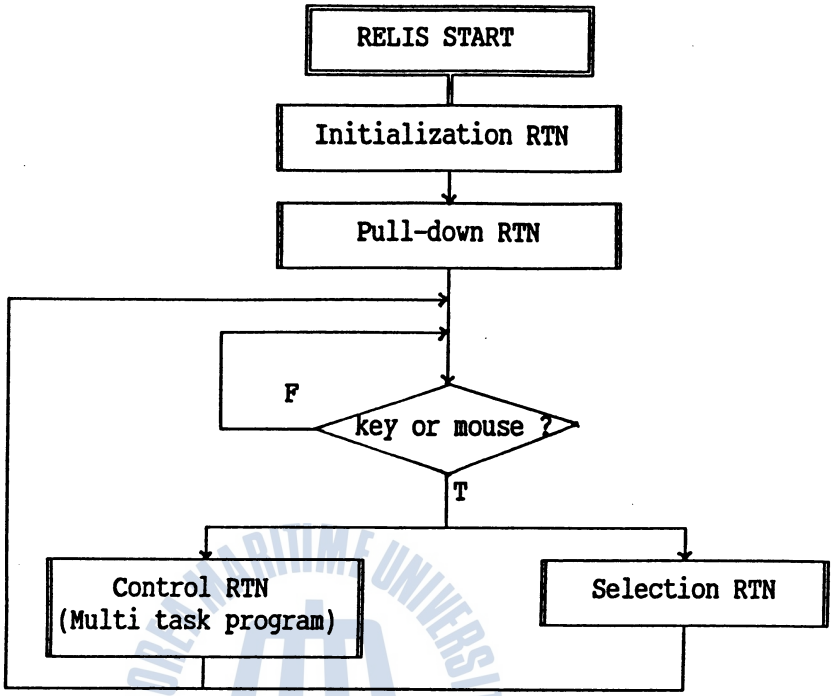


Fig. 3.2 Overall flowchart of RELIS.

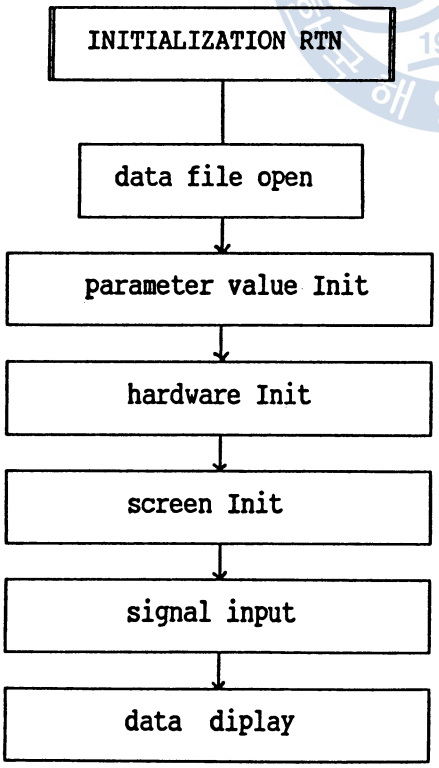


Fig. 3.3 Flow chart of Initialization routine.

3.2 시스템의 구조

시스템은 그림 3.2와 같이 Initialization 루틴, 풀-다운 메뉴 방식을 기본으로 메뉴를 선택할 수 있는 Pull-down menu 루틴, 시뮬레이션과 실시간 제어를 하기 위한 Control 루틴, 하드웨어 및 사용환경조건을 선택하는 Selection 루틴이 있다.

이중 Selection 루틴에는 신호들을 화면에 다양하게 디스플레이해 주는 기능, A-D변환기를 통해 입력되는 신호들의 범위를 설정해 주는 기능, 탱크레벨신호, 탱크레벨의 변화율신호 및 제어입력신호등 많은 입출력 채널중에서 제한된 화면에 표시하고자 하는 채널을 설정하기 위한 기능이 있다.

또한 Pull-down menu 루틴을 구현하기 위해 스크린 리스토어 기법, 마우스 제어 기법, 비디오 어댑터 제어 기법 등을 이용하였다.

Collection

1) Initialization 루틴

Initialization 루틴은 그림 3.3과 같이 구성되어 있으며, 이들에 대한 기능은 다음과 같다.

① 제어의 동작점에 따라 적응성 있는 피이드백이득값을 읽어 들이기 위해 미리 작 성된 테이블을 오픈한다.

② 이 오픈된 테이블에서 값을 읽어 들여 피이드백이득값을 초기화하고, 제어 루틴에서 사용하는 각종 파라미터값을 초기화한다.

③ A-D/D-A 변환기의 기종에 따라 서로 다른 베이스 어드레스 및 각종 환경을 설정한다.

④ 현재 화면에서 사용 가능한 키에 관련된 도움기능 및 윈도우 화면을 초기화 한다.

⑤ 실시간 데이타를 입력받아 텍스트 모드에서 실수형값을 출력한다.

2) Control 루틴

Main menu에서 키값 또는 마우스에 의해 선택되어지는 Control 루틴으로서 다음과 같다.

① Para_iden 루틴 : 제어대상의 파라미터값에 따른 적정 샘플링 타임을 도출한다.

② Control_start 루틴 : 멀티타스크 환경하에서 제어대상을 실제로 제어하는 부분 으로서 데이타의 입출력, 레벨제어 시스템의 일시 정지, 운전중 목표치와 샘플링 타임의 변경, 제어시스템 운전의 정지등을 할 수 있다.

③ Open_sim 루틴 : 제어대상의 수학적 모델과 파라미터값으로 부터 계단상 목표치 변경명령에 대한 여러 신호값을 수치계산 후 그 응답그래프를 개회로 특성으로 제공해 준다.

④ Close_sim 루틴 : 탱크레벨제어시스템의 수학적 모델로 부터 계단상 목표치 변경명령에 대한 여러 신호값을 수치계산 후 그 응답그래프를 폐회로 특성으로 제공해 준다.

여기서 Open_sim 루틴과 Close_sim 루틴은 시뮬레이션을 위한 루틴으로서 Control_ start 루틴에서 사용되는 기법의 대부분이 동일하게 이용되고 있으므로 Para_iden 루틴과 Control_start 루틴에 대해서만 설명한다.

① Para_iden 루틴

그림 3.4와 같이 샘플링 타임을 제어하는 인터럽트 루틴을 제외시킨 상태에서 단지 CPU 클럭 속도에 의한 입출력만을 행한다. 제어대상의 시정수를 구할 수 있게 해 주며, PC기종에 따라 다르게 나타나는 최소 샘플링 타임을 도출할 수 있게 해 준다.

그림에서 ⓐ의 bitmap1, bitmap2는 그래픽 모드에서 화면을 저장하기 위한 변수이며, ⓑ에서 한 화면을 전부 그린 경우에는 화면을 초기화시키기 위해 ⓐ의 변수를 다음과 같이 이용한다.

```
setgraphmode(mode)
farsize=imagesize(0,0,Tmax_dot,Ymax_dot)
bitmap1=_graphgetmem(farsize)
bitmap2=_graphgetmem(farsize)
getimage(pos_X,pos_Y1-Tmax_dot,pos X+Tmax dot,pos Y1,bitmap1)
getimage(pos_X,pos_Y2-Tmax_dot,pos X+Tmax dot,pos Y2,bitmap2)
```

Collection

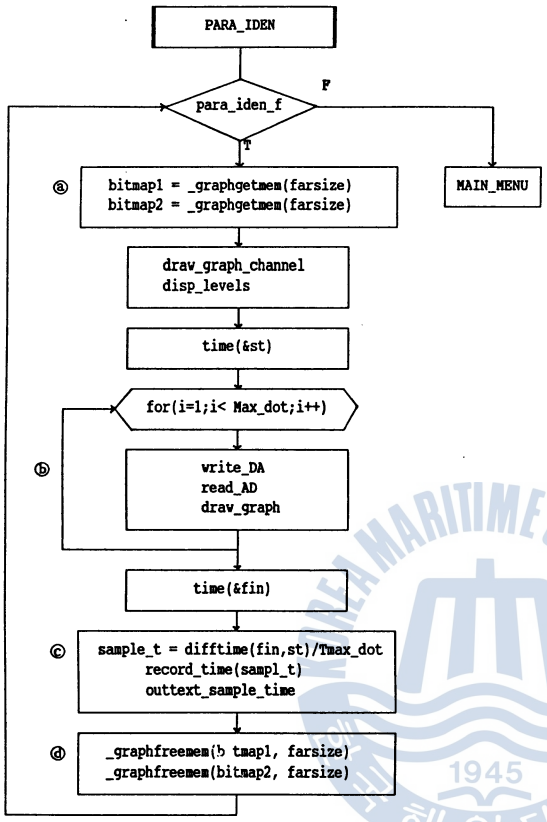


Fig. 3.4 Flow chart of Para_iden routine.

sample_t는 계산된 샘플링 타임이며 이것을 ⓒ에서 화면에 출력하고 있다. ⓓ는 para_iden 루틴에서 Main menu로 되돌아 오기전에 bitmap1, bitmap2를 메모리로부터 해방시키는 작업을 나타내고 있다.

② Control_start 루틴

그림 3.5(a)와 같이 Main menu에서 Control 루틴이 선택되면 그래픽 모드로의 전환이 이루어지며, Current level에 대응하는 피이드백이득값을 읽어 들인다. 여기에서는 멀티타스크 환경하에서 이벤트가 발생되면 타스크 절환이 행해져 시스템 운전의 일시정지, 운전중에 샘플링타임과 목표치를 변경시킬 수 있으며 우선적으로 control_start 메뉴가 선택되어 그림 3.5(b)와 같이 Control 루틴을 실행하게 된다.

즉 ⓑ에서는 샘플링 타임을 설정 및 변경시키기 위한 인터럽트처리 루틴을 나타내고, ⓒ는 Control 루틴이 동작되고 있는 도중에 초기 피이드백 이득값을 변경시킬 필요가 있을 때 동작점에 부합되는 값으로 변경시키는 루틴이다. 이때 피이드백 이득값을 변경시키는 시점은 다음과 같은 조건에 따라 결정된다.

Goal level 〉 Initial level인 경우 :

current level ⇨ Initial level + (Goal level - Initial level) × 0.9인 최초의 시점

Goal level 〈 Initial level인 경우 :

current level ⇦ Initial level + (Initial level - Goal level) × 0.9인 최초의 시점

ⓓ는 탱크레벨시스템의 제어알고리즘 과정을 나타내며, ⓔ는 입출력 결과를 그래픽으로 표시하는 루틴이다.

한편 샘플링 타임을 조정하기 위하여 8253 클럭을 변경시키는 경우에는 PC내의 모든 타임이 8253 클럭에 의해 제어가 이루어지고 있으므로 제어 루틴을 중지함과 동시에, ⓐ에서와 같이 타임제어 인터럽트 해제 루틴을 실행시키므로써 원상복귀시켜야 한다. 그렇지 않을 경우에는 모든 시스템의 타임이 CPU내의 클럭에 의해 제어되는 본래의 타임과는 달리 8253에 의해 제어가 되므로 주변장치의 실행과 타임제어가 오동작을 일으킨다.

dCollection

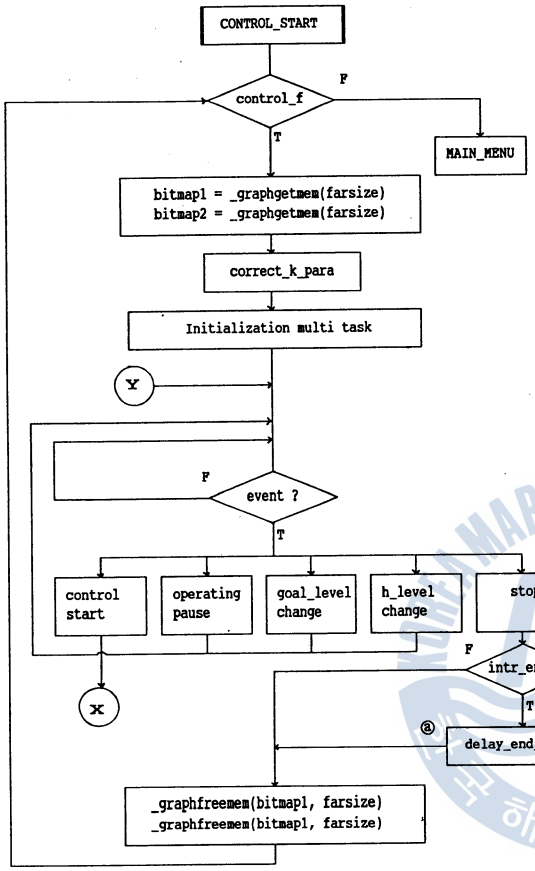


Fig. 3.5 Flow chart of Control_start routine (a).

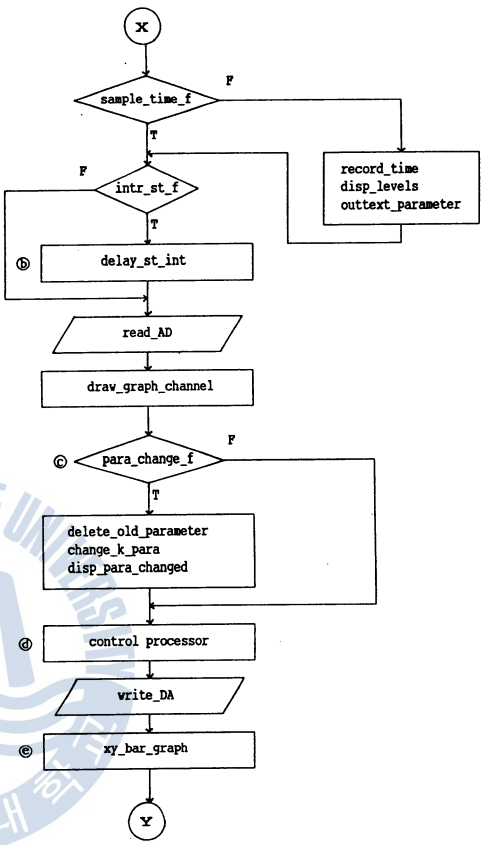


Fig. 3.5 Flow chart of Control_start routine (b).

4. 실 험

본 연구에서 개발한 시스템의 타당성을 입증하기 위하여 통합환경 시스템에 탱크레벨시스템을 제어할 수 있는 그림 4.1과 같은 적응제어기를 삽입하여 실험 하였으며, 실험장치는 그림 4.2와 같이 구성하였다. 여기서 (a)PC(cpu i80386), (b)레이저 프린터, (c)X-Y 레코더, (d)A-D/D-A 변환기, (e)전압/전류 변환기, (f)디지탈 멀티메타, (g)DC 전압 공급장치 (h)Main box(power 스위치, auto/manual 선택스위치, 인터페이스용 터미널, 전류/전압 변환기등), (i)압력 센서, (j)스위치 박스(Remote/Local, 펌프, 솔레노이드밸브 스위치), (k)저장 탱크, (l)유량계, (m)압력 스위치, (n)제어 밸브, (o)탱크로 구성되어 있다.

그림 4.3은 레벨제어 장치의 전체 블록선도를 나타낸 것으로 탱크의 레벨 신호가 센서로 부터 4~20mA로 검출되고, 이 신호가 전류/전압 변환회로를 거쳐 A-D/D-A 변환기로 입력된다. RELIS 시스템은 이 입력신호를 받아 실시간제어를 행한 후 다시 A-D/ D-A 변환기와 전압/전류 변환회로를 통하여 제

dCollection

어밸브를 조작한다.

그림 4.4는 RELIS 시스템의 주메뉴에서 Control 루틴을 선택하여 실시간 제어를 수행하는 과정에서의 입출력 그래프이다. A/D Channel No.1은 탱크레벨 입력값의 추이를 나타내며, D/A Channel은 제어밸브로 출력되는 값의 추이를 나타낸다. 여기서 약 35초가 경과되는 시점인 ⓐ에서 목표치를 변경하고자 이벤트를 가하여 수정한 후 ⓑ지점에서 엔터키를 쳤을 때의 실험결과이다. 즉 제어를 수행하는 타스크를 실행함과 동시에 목표치를 변경시키는 타스크를 병행처리함으로써 실제 탱크레벨의 목표치가 달라지고 있는 것을 나타내며, 그림에서는 목표치 변경 타스크가 실행중에도 제어가 잘 수행됨을 알 수 있다.

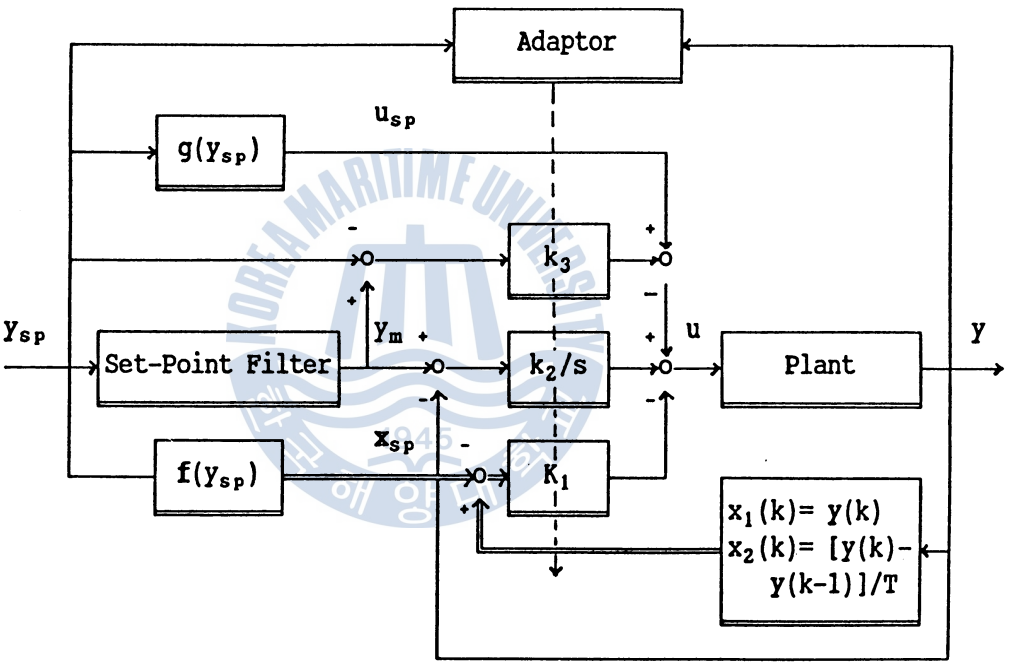


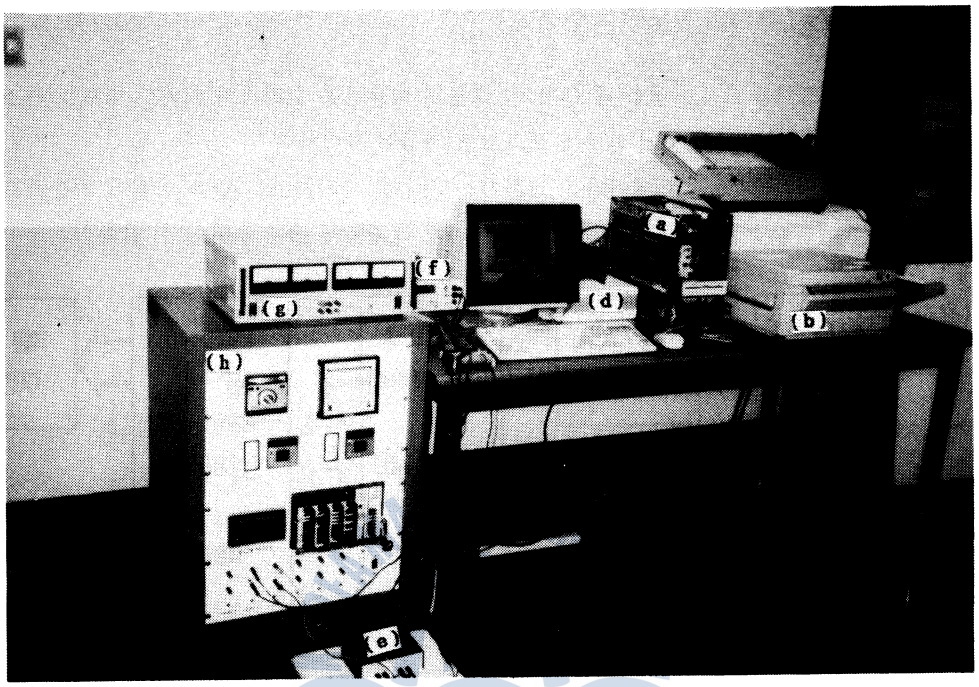
Fig. 4.1 Block diagram for tank level control system.

5. 결 론

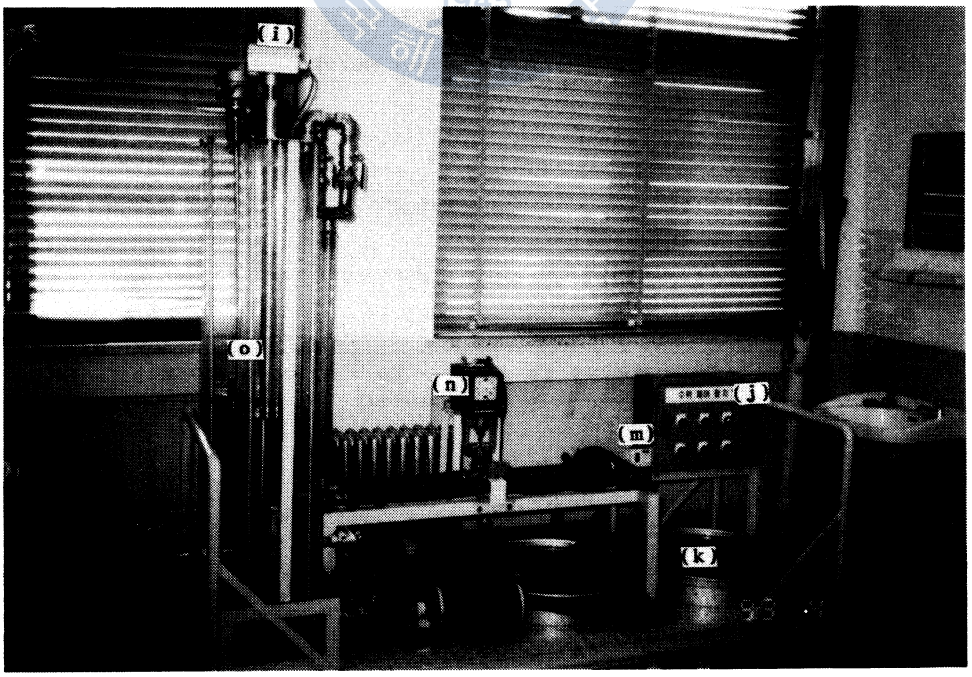
본 논문에서는 탱크 레벨제어를 위한 실시간 통합환경 시스템에 대하여 논하였다. 특히 실시간 레벨제어가 행해지는 가운데 다른 기능을 수행하고자 하는 부분에 대해서는 멀티타스크 환경하에서 운전의 일시중지, 샘플링타임의 변경 및 목표치 변경등의 타스크로의 절환을 가능하게 하여 데이타를 실시간으로 처리하는 부분의 시간지연 요소를 최소화 하였다. 또한 텍스트 모드와 그래픽 모드의 절환기법을 도출하고, IBM-PC내의 8253을 이용하여 운전중에 임의로 샘플링타임을 조정할 수 있게 하는 인터럽트 처리 기법을 도출하였다.

한편 본 시스템에 이미 설계된 제어기를 삽입하여 실험해 본 결과 다음과 같은 장점이 있음을 알 수 있었다.

dCollection



(1) Controller



(2) Controlled System

Fig. 4.2 Structure of Tank Level Control System.

dCollection

1) 시스템의 타스크를 기능별 독립한 형태로 구성할 수 있다. 따라서 기능별 프로그램 수정 및 삽입이 용이하다.

2) 각 부분 타스크들을 프로그램할 때 CPU의 효율적 이용등을 고려하지 않아도 되므로 타스크에 의존하는 부분만을 프로그래밍하면 되기 때문에 프로그램 작성이 용이하다.

3) 각 타스크는 다른 타스크와 데이타를 교환하는 경우를 제외하고는 단독 실행이 가능하므로 우선도 부여, 자원공유, 메일에 의한 기동 등의 기능을 통합환경 레벨에서 쉽게 부여할 수 있다.

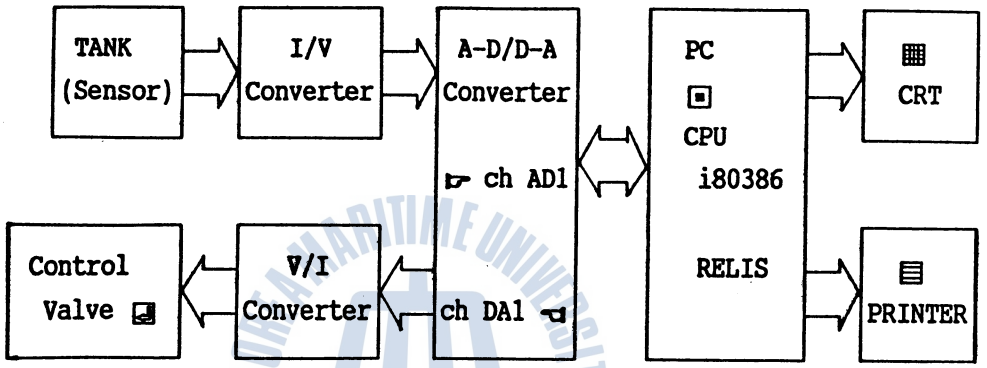


Fig. 4.3 Functional Block Diagram of Tank Level Control System.

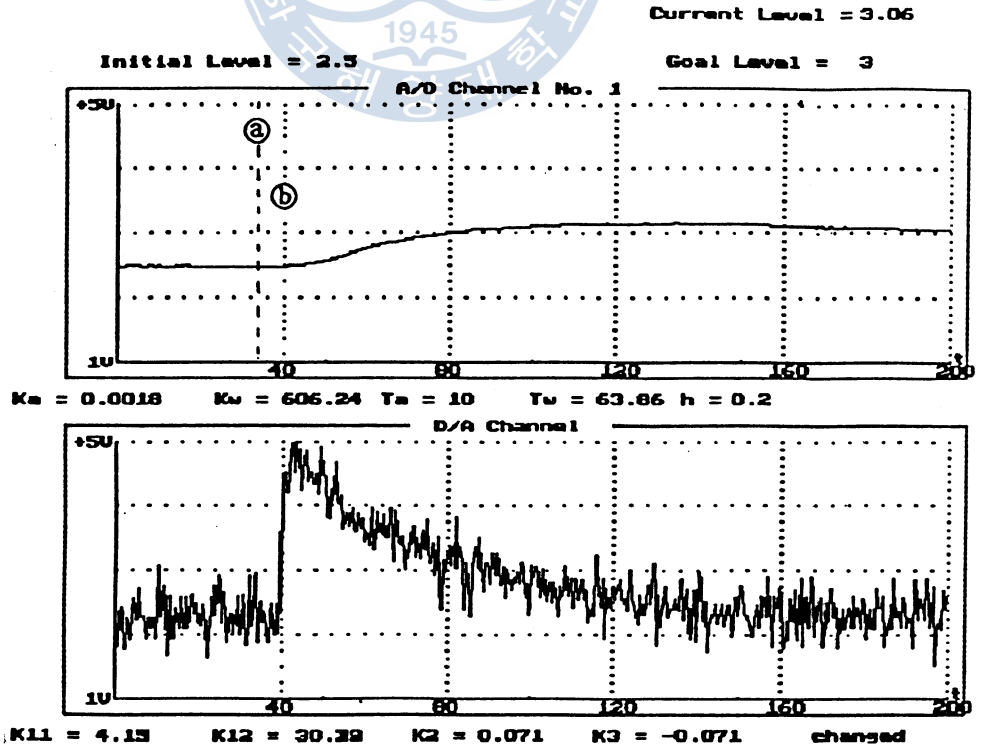


Fig. 4.4 step response for the set-point change, Goal level=+0.5(V).

dCollection

참고문헌

1) Benjamin C. Kuo, Automatic Control System, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1991.
2) 386-Matlab User's Guide, Math Works, Inc., 1989.
3) Program CC version 4 Reference Manual, Peter M. Thompson & Sysems Technology, Inc., 1988.
4) 정재학 "지능적 레벨 제어와 실시간 통합환경 시스템 구현에 관한 연구"한국해양대학교 대학원 석사학위논문, 1993.
5) 이문기, 마이크로컴퓨터 응용 계측 회로, 기술연구사, 서울, pp.15-21, 1986.
6) Avtar Singh & Walter A. Triebel, The 8086 and 80286 Microprocessors, Prentice-Hall, Inc., N. j., pp.275-287, 1990.
7) 권영준, "IBM PC 포트 분석 (1)", 마이크로 소프트웨어, Vol. 12, pp.188-194, 1988.
8) PCL-812 Enhanced Multi-Lab Card User's Manual, Advantech Co., Ltd. Taiwan, 1990.
9) B. Lorigerne, Analog-Digital and Digital-Analog Conversion, Heyden & Son Ltd., London, pp. 131-158, 1982.
10) 金井隆, 清水伸一, C/その實踐と應用.I, Shuwa System Trading Co., Ltd., Tokyo, Japan, 1985.

dCollection

Collection