CAT를 적용한 다중처리지원 계측 시스템 인터페이스 설계 및 구현에 관한 연구

A Study on the Design and Implementation of Multitasking Measurement System Interface with CAM

전 동 근*, 문 대 철**

(Dong Geun Jeon*, Dai Tchul Moon**)

*본 연구 논문은 "한국과학재단 목적기초연구" 지원에 의해서 연구되어진 연구과제중 일부임 (연구과제번호 : 90-01-00-14)

ABSTRACT

In this paper, interface of measurement systems supporting multitask for CAT(computer aided test) is realized. Thread-based method is used for processing, and round robin method for scheduleing. The interface of measurement systems supporting multitask for CAT is implemented using HP8590A spectrum analizer and HP473B meter as system operating instruments. Implemented softwares consist of a total of 9 modules and each module is mutually shared. We designed the multitasking system which is able to construct added software modules for purposive instrument, if more instruments needed additively. When a number of devices are connected, some problems may occur. One of them get into a deadlock and the other get loss of data during transmission. They are get into mainly due to the slight descrepancies among the communication protocals. To identify the variations of the standard and the cause of the problems, Protocol Analyzer is designed and implemented.

Experimental results show that it is able to analyze various protocols. User can save measurement time and even without expert knowledge.

요 약

본 논문에서는 CAT를 적용하여 다중 처리를 지원하는 계측 시스템의 인터페이스를 설계하고 구현하였다. 다중 처리는 프로세스 기법으로 트레드 방식을 사용하였고, 스케줄링으로는 라운드 로빈 방식을 사용하였다. 구현된 다중처리 시스템은 HP8590A 스펙트럼 분석기와 HP473B 전력계를 시스템 운영 계측기로 이용하였다. 이밖에도 15대 까지의 계측기를 첨가하여 시스템을 운용할 수 있도록 하였다. 개발된 소프트웨어는 총 9개 모듈로 구성되어 있고 각 모듈들은 상호 공유되도록 구현하였다. 더 많은 계측기를 부가할 경우에는 목적 계측기에 알맞는 소프트웨어 모듈을 합하여 구성할 수 있도록 하였다. 또한, 계측기를 여러대 접속할때 발생할 수 있는 문제점에 대한 해결책을 제시하였다. 문제점은 정보의 전송도중 버스가 데드록이 되거나 데이타를 손실하는 경우이다. 문제 발생의 원인은 각 계측기가 갖고있는 정보 전송 프로토콜의 차이점이 있기 때문인데, 이를 알아내고자 프로토콜 분석기를 설계하여 컴퓨터에 접속할 수 있도록 구현하였다. 실험한 결과 두 대의 서로 다른 계측기가 갖고 있는 공통적인 프로토콜 패턴을 찾을 수 있었다. 이 시스템을 이용할 경우 사용자는 전문지식 없이도 측정 시간과 오차를 줄일 수가 있다.

* 한서대학교 전자공학과
** 호서대학교 정보통신공학과
접수일자 : 1993년 4월 3일

Ⅰ. 서 론

최근 반도체 기술의 발달로 인한 컴퓨터 성능의 고도화로 컴퓨터는 사용목적이 일반적인 데이타 처리에서 정보화 사회를 지향하는 종합 정보 처리 형태로 변화하고 있으며 이러한 과정의 하나로 자동화 작업이 추진되어 점차로 컴퓨터 사용의 개념을 확대시켜 가고 있다. 처음에는 사무 자동화로 시작하여 공장 자동화로 이루어 지면서 제품에 대한 계측 자동화가 요구되고 있지만 이와 관련된 연구가 미비한 실정이다. 특히 공장 자동화와 계측·검사 자동화 분야에서 정확성과 신속성을 얻기위해 수작업으로 행해오던 계측 산업을 자동화시킴으로써 실험보다 계측 작업과 데이타 처리에 소요되는 검사 비용과 시간을 단축시키고 실험하는 시간에 투자하게 되었다. 이에 따라 이전에는 생각하지도 못했던 실험 및 측정을 다양하게 할 수 있게 되었고 이의 분석도 다각도로 할 수 있게 되는 등 여러가지 잇점이 나타나게 되었으며 결과적으로 좀 더 많은 부분의 계측 자동화가 절실히 요구되고 있다. 이와같이 계측 자동화의 대표적인 시스템이 CAT(computer aided test)이다.

CAT 시스템은 VXI 버스와 GP-IB를 컴퓨터에 연결시켜 구성한 시스템으로서 각종 장비를 연결, 통합 계측을 하고 있으며 컴퓨터가 갖는 고속 처리 능력을 이용하여 생산성의 향상에 크게 기여하고 있다.

본 논문에서는 자동 계측 시스템을 구성함때의 심각한 문제점이 각 접속 장치간의 프로토콜 차이를 알아내고 이를 해석하여 문제점을 해결할 수 있도록 하기 위해서 프로토콜 분석기를 설계하였으며, 다양한 계측 장비와의 인터페이스를 구성하기 위한 인터페이스용 소프트웨어를 개발하였다. 또한 라운드 로빈 방식을 스케쥴링과 트레드 베이스 방식의 프로세스 기법을 이용하여 동시에 여러대의 계측기를 측정할 수 있는 다중처리 프로그램을 구현하였다.

CAT는 계측 자동화를 위한 것으로 실험과 검사에 소요되는 시간과 비용을 절감하여 실험의 일관성과 품질의 통계적 정보를 즉각적으로 확보할 수 있는 잇점을 제공한다. 또한 개발 제품의 검사나 혹은 어떤 실험의 측정을 컴퓨터로 자동화시키는 것이며, 검사나 실험은 원하는 결과를 얻기 위해 실험 대상에 대한 다양한 정보를 추출하는 과정이다.

테스트 시스템은 계측기와 컴퓨터를 결합시켜 테스트 절차를 자동화할 수 있다. 계측기와 컴퓨터를 연결시키는 가장 전형적인 방법으로써 GP-IB와 VXI버스를 연결하여 인터페이스를 구성하는 시스템이다. 그림1은 일반적인 CAT 시스템 구조를 나타낸다.

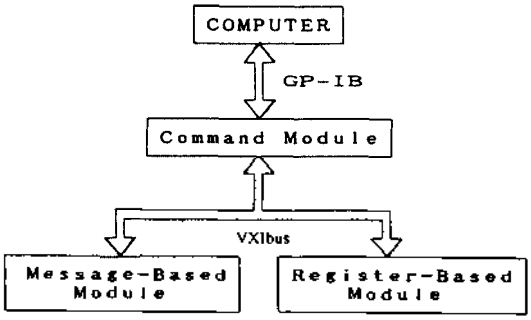


그림 1. CAT 시스템
Fig 1. CAT system

Ⅱ. 시스템 개발 환경

다중처리에서 프로세스를 사용하는 방법에는 프로세스 베이스 다중처리 방법과 트레스 베이스 다중처리 방법등이 있는데 프로세스 베이스 다중처리 방법은 동시에 둘 이상의 프로세스를 처리하는데 쓰이며 트레드 베이스 다중처리 방법은 동시에 단일 프로그램의 여러 부분을 실행하는데 쓰인다. 본 논문에서는 트레드 방법을 사용하여 단일 프로그램의 여러 부분을 실행하도록 구성하였다.

2.1 스케줄링

다중처리 시스템은 임의의 작업에서 다른 작업으로 수행을 바꾸는 루틴을 가지고 있는 장점이 있다. 이 루틴을 스케줄러라 부르며 스케줄링 기법은 단계별이나 방법별로 구분할 수 있다. 그림2는 4단계로 이루어진 중앙 처리 장치의 스케줄링 흐름도를 보여주고 있다.

2.2 프로세스 스케줄링 알고리즘

본 논문에서 프로세스 스케줄링 알고리즘으로는 라운드 로빈 스케줄링 방법을 사용하였다. 라운드 로빈 스케줄링은 시분할 시스템을 위하여 고안된 선점 스케줄링 방식의 하나이다. 선점 스케줄링은 높은 우선 순위를 가진 프로세스들이 빠른 처리를 요구하는

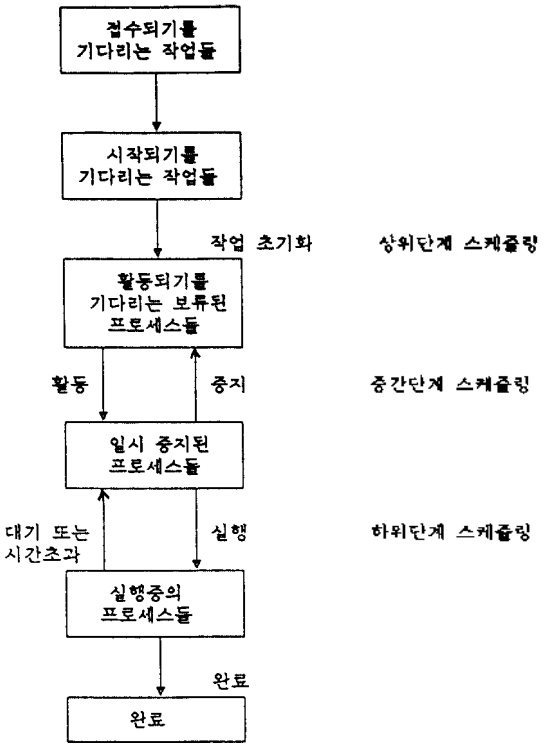


그림 2. 중앙 처리 장치의 스케줄링 흐름도
Fig 2. Scheduling flowchart of central processing unit

시스템에서 유용하다. 선점을 효과적으로 하기 위해서는 많은 프로세스들이 주기억 장치내에 있어야 하고 중앙 처리 장치가 사용가능해질 때마다 준비 완료 상태에 프로세스가 있어야 한다.

이 방법은 그림3과 같이 FCFS(First Come First Serve)에 의하여 프로세스들이 내보내지며 각 프로세스는 같은 크기의 중앙 처리 장치를 할당받는다.

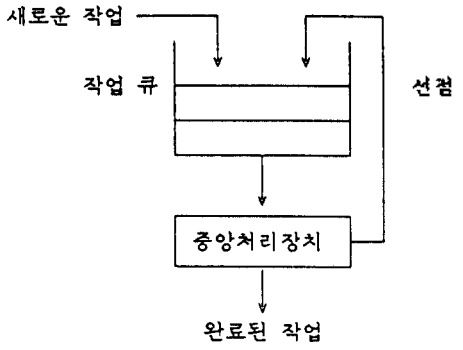


그림 3. 라운드 로빈 스케줄링
Fig 3. Round-robin scheduling

만약 프로세스가 중앙처리장치 시간이 만료될 때까지 처리를 완료하지 못하면 중앙 처리 장치는 대기중인 다음 프로세스로 넘어가며 수행못한 프로세스는 준비 완료 리스트의 가장 뒤로 보내어진다.

이 스케줄링 방법은 대화식 사용자들에게 알맞은 응답 시간을 보장해주는 시분할 방식의 시스템에 유리하다. 이때 효율적인 문맥 교환을 기법을 쓰고 프로세스들이 동시에 주기억 장치내에 유지될 수 있도록 충분한 기억 용량을 준비한 다면 선점으로 인한 오버헤드를 최대한 줄일 수 있고 시스템 성능도 높일 수 있다.

Ⅲ. 시스템 및 프로그램 구현

3.1 다중처리 프로그램 구현

데이타와 명령어를 처리하는데 있어서 인터페이스 처리는 컴퓨터와 계측기간에 그림4와 같이 메시지를 송수신한다.

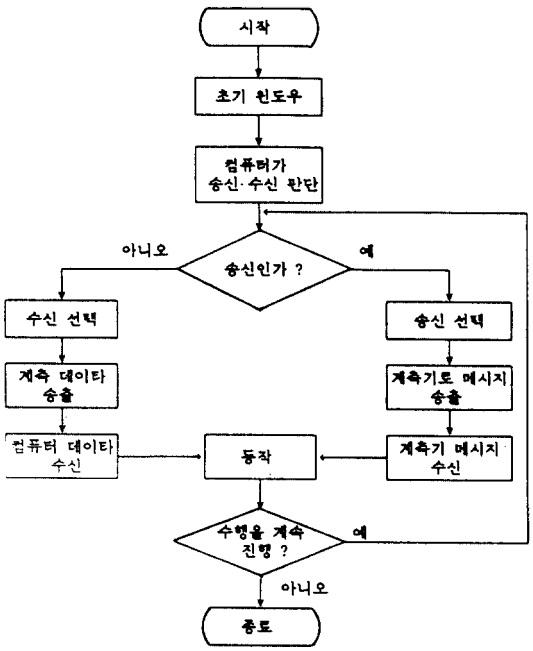


그림 4. 인터페이스 처리부의 송수신 흐름도
Fig 4. Flow chart of processing unit for transmitting and receiving messages

그림4에서 보는 바와 같이 컴퓨터가 송신자가 될 것인지 수신자가 될 것인지를 선택한 후 송신 상태에서는 송신 메시지를 송출하여 계측기가 송신명령에

따라 동작하도록 하며, 수신 상태에서는 컴퓨터가 데이타를 수신하여 데이타 값을 표시하도록 구현하였다.

논문에서 사용한 Turbo C는 인터럽트 서비스 루틴을 위해 제공되는 인터럽트 함수형 수정자가 있어서 인터럽트로 함수를 선언하면 컴파일러에서는 SP (stack pointer)와 SS(stack segent)를 제외한 모든 레지스터들을 자동적으로 보존시키고 인터럽트 복귀 명령으로 함수를 종료한다. 그림5는 하나의 작업에서 다른 작업으로 교환하기 위한 작업 교환과 다중처리를 위한 전체 작업 구조를 나타내고 있다.

또한, 인터럽트를 일으킬 때마다 작업을 바꾸기 위한 스케줄러가 요구되는데 그림6은 이러한 스케줄러의 흐름도를 나타내고 있다.

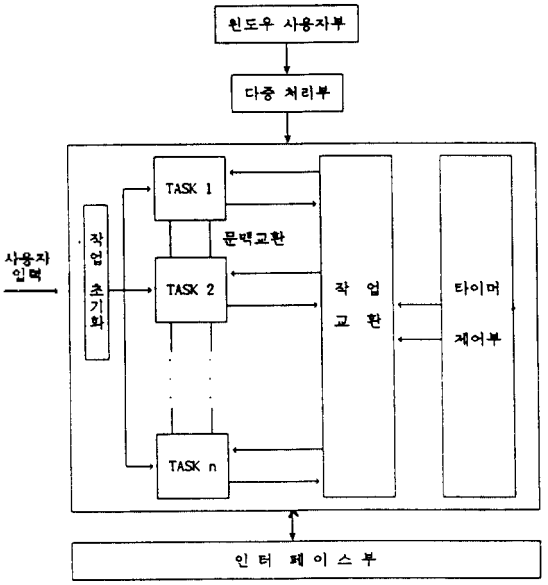


그림 5. 다중처리 전체 작업 구조
Fig 5. Entire task structure for multitask

3.2 프로토콜 분석기 하드웨어 설계

본 절에서는 GP-IB프로토콜이 다양하게 합성될 수 있기 때문에 발생하는 문제점이 버스의 데드록 상태를 제거하기 위한 정보를 얻기 위해 GP-IB 프로토콜 분석기를 설계하는 과정을 기술한다.

GP-IB 프로토콜은 그림7에 보인 것처럼 다양한 신호들의 조합으로 나타나고 있다. 여기에는 명령 모드 경우만을 보였지만 디바이스 메시지 프로토콜에서는

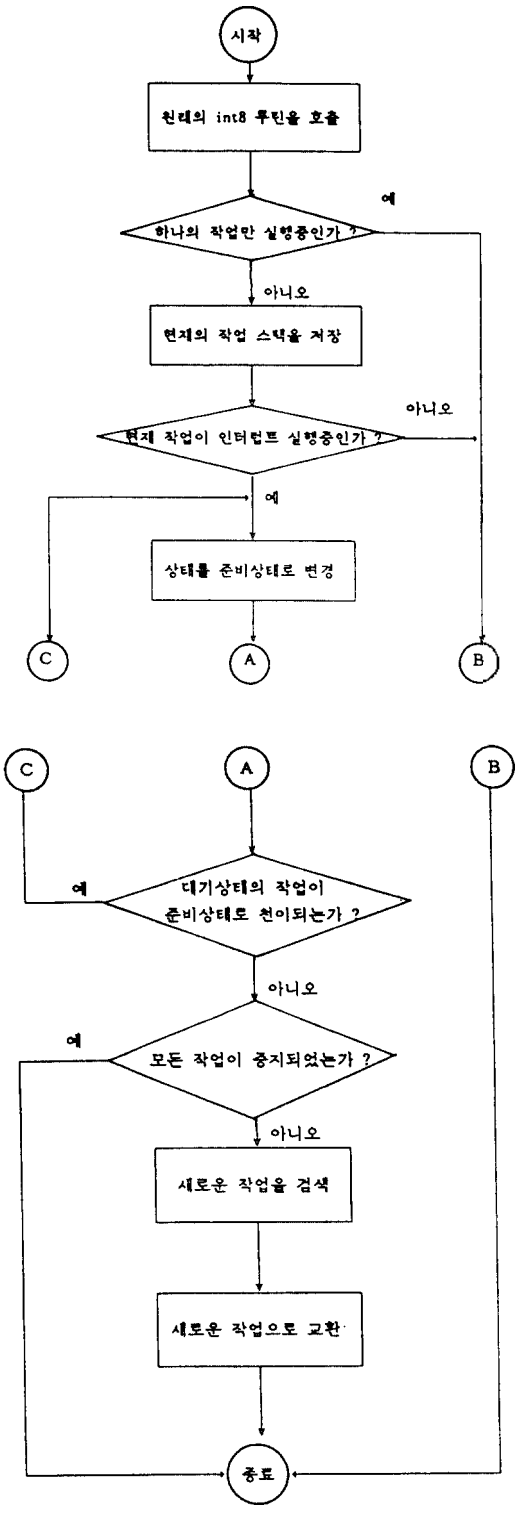
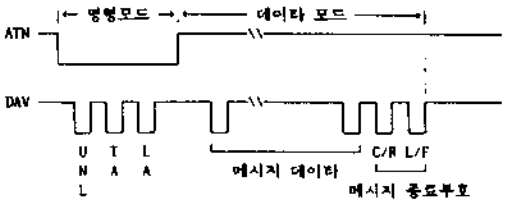
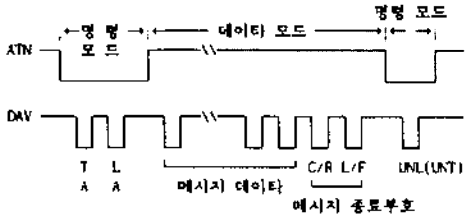


그림 6. 스케줄러 흐름도
Fig 6. Flow chart of schedular

더욱더 다양하게 나타나고 있다. 예를 들면 메시지 구분에서 사용하는 구분 기호를 코마를 사용한 것도 있고, C/R을 사용한 것도 있는데 C/R은 L/F와 함께 메시지 종료 부호로 사용되므로 송신측에서 C/R 다음에 전송한 정보를 수신측이 제대로 받지 못하고 잃게 되며 심각한 경우는 수신측에서 메시지 구분에 사용된 C/R을 종료 부호로 착각하여 그 다음 코드가 L/F가 들어 오길 기다리는 상태가 되어 버스가 데드록 상태에 놓이게 된다. 또한 종료 메시지를 나타내는 방법에 C/R, L/F 코드 시퀀스를 사용하는 것과 L/F 코드와 함께 EOI 신호선에 펄스를 사용하는 것이 있는데 기기마다 서로 다른 방법을 채택하고 있어서 이것 또한 버스의 데드록 요인이 되고 있다. 원인은 GP IB 프로토콜의 가능한 모든 조합을 고려하지 않고 어느 정도의 가정하에 프로그램을 작성한데 있다고 볼 수 있다. 최근에 생산되는 계측기기의 GP-IB는 일반적으로 많이 사용되는 프로토콜로 운용 프로그램을 작성하기 때문에 별 문제가 발생하지 않는다. 문제는 과거에 생산된 계측기기의 GP-IB 운용 프로그램이 가지고 있는 프로토콜이다.



가. 처음에 언리슨을 보내는 방식



나. 마지막에 언리슨(언토커)를 보내는 방식

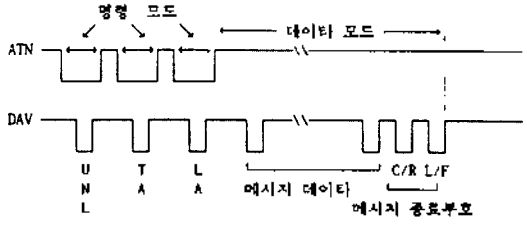


그림 7. 명령 모드의 송수신 프로토콜
Fig 7. Tranceiver protocol of command mode

이러한 경우에는 예측치 못했던 버스의 데드록이 발생하거나 정보를 잃어버리는 현상이 발생하는데 원인을 파악하기가 대단히 힘들다. 왜냐하면 기기의 메뉴얼에도 어떠한 프로토콜을 사용하였는지 세밀하게 기술되어 있지 않으면 실제로 기기의 프로그램이 가지고 있는 프로토콜이 어떻게 되어있는지 알 수 없기 때문이다. 이런 이유로 GP-IB 버스 분석기를 사용하는데 장비가 고가인 관계로 더욱이 국내에서 사용되어진 적이 없다. 이러한 문제점이 국내에서 계측 자동화 시스템을 구축하는데 주요한 장애 요인이 되고 있다.

본 논문에서 제안한 GPIB 프로토콜 분석기는 이런 점에 초점을 맞추어 장애요인인 버스의 데드록 현상을 일으키는 원인을 찾을 수 있는 정보를 얻을 수 있도록 다음과 같이 설계하도록 한다.

첫째, GP IB 버스상에 실리는 모든 정보를 모니터링 할 수 있어야 한다. 이것은 버스가 어느 시점에서 데드록이 되는가 알아내기 위해서이다. 이것을 하기 위해서는 GP-IB 버스 라인에 규격외의 다른 전기적 영향을 주지 않도록 접속하여 전기적 신호의 상태를 프로그램에서 읽게 해준다.

둘째, 각 기기에 장착된 GP-IB 접속 장치가 가지고 있는 운용 프로그램의 프로토콜을 분석할 수 있어야 한다. 이것은 프로토콜 차이에서 발생하는 버스의 데드록 현상이나 전송 도중 정보를 잃어버리는 원인을 찾기 위해서이다. 이것을 하기 위해서는 GP-IB 버스 라인에 규격외의 다른 전기적인 영향을 주지 않도록 접속하여 프로그램으로 시험 신호(test signal)를 버스 라인에 실을 수 있게 한다.

셋째, 설계된 하드웨어의 구현을 할 수 있어야 한다. 이것은 실험이 가능해야 하므로 실제로 설계가 제대로 되었는지 여부를 확인하기 위함이다. 따라서 실제로 제작을 하려고 할 경우 국내에서 회로에 사용되는 모든 칩을 구할 수 있어야만 한다.

넷째, 접속 장치로도 사용이 가능해야 한다. 이상의 4가지의 필요 사항에 따라 하드웨어를 설계하였다. 그림8은 3선 핸드쉐이크의 하드웨어 구성도이다.

하드웨어 설계 조건 세번째와 네번째는 모두 범용 TTL IC로 구성하였고, CPU는 MC3448A를 사용하

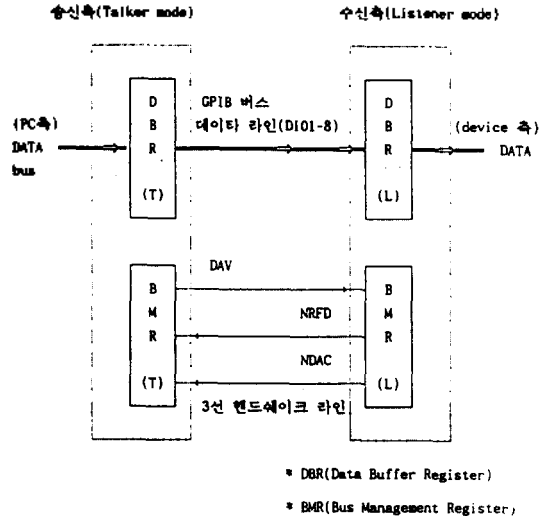


그림 8. 3선 핸드쉐이크 하드웨어의 블럭도
Fig 8. Block diagram of 3 line handshake hardware

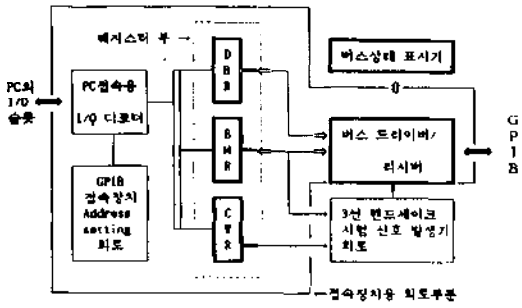


그림 9. 프로토콜 분석기의 블럭도
Fig 9. Block diagram of protocol analyzer

였다. 설계된 회로의 블럭도를 그림9에 나타내었다.

제작된 하드웨어의 전체 구성은 그림10과 같다.

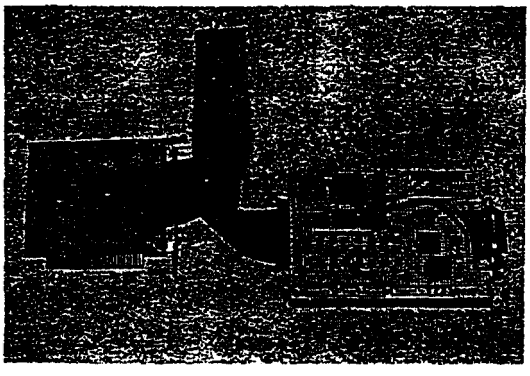


그림 10. 설계된 프로토콜 분석기
Fig 10. Designed protocol analyzer

3.3 프로토콜 분석기 소프트웨어 구현

본 절에서는 프로토콜 분석기의 운용 소프트웨어에 대해서 기술하는데 3.2절에서 설명한 프로토콜 분석기의 설계 조건을 고려하여 필요한 프로토콜 분석 알고리즘과 접속 장치로 사용하기 위한 알고리즘을 설계하고 의사 코드(pseudo code)를 사용하여 기술한다.

프로토콜 분석 알고리즘은 버스상의 정보에 대한 것을 상세히 알아야 분석이 가능하기 때문에 먼저 버스의 상태를 모니터할 수 있는 버스 모니터 알고리즘이 필요하다. 그림11은 이와 같이 설계될 버스 상태 모니터 알고리즘을 나타낸다.

```
// GPIB BUS status monitor Algorithm //
// limit time : 데드록 상태를 체크하는 제한시간 값 //
// status_data[ ] : 버스의 상태 데이타 //

loop (condition)
    read DBR ; // 버스의 상태 데이타를 읽는다.//
    if (전에 기록한 데이타 = 읽은 데이타)
       for i = 0 to limit  do
          count_time ;
          read DBR ;
          if (전에 기록한 데이타 〈〉 읽은 데이타)
             record status data ;
             exit for ;
          end_if
       end_for
          if (count_time = limit_time)
       print ("버스에 데드록 상태 발생 !!!") ;
       exit dead_verify ; // 버스의 데드록 상태 확인
                          루틴으로 분기한다.//
          end_if
    else
          record status_data ;
end loop
```

그림 11. 버스 상태 모니터 알고리즘
Fig 11. Monitor algorithm of bus state

여기에서는 버스의 상태를 그대로 읽어서 기록하고 화면에 표시하는데 표시하는 단위는 3선 핸드쉐이크를 행할때 각 상태마다 표시된다. 만약 버스의 상

태가 데드록이 되면 계속해서 3선 핸드쉐이크가 이루어지지 않으므로 더 이상 버스의 상태가 변화하지 않는다. 이 점을 이용하여 일정 시간이상 버스의 상태 변이가 없으면 버스에 데드록이 발생한 것으로 간주하고 데드록 확인을 하게 되는데 한번 더 일정시간 동안을 지켜 보다가 더 이상 아무 변화가 없게 되면 버스가 데드록이 되었음을 표시한다. 그리고 버스에 데드록이 일어나지 않았을 경우에는 계속해서 정해진 조건만큼 버스의 상태를 기록하고 수행을 완료한다.

정보 송수신 모드의 프로토콜 분석 알고리즘은 버스의 모니터 알고리즘에 의해서 버스의 데드록이 발생하였다는 보고를 받았을 때 사용하는 것인데 버스상에 접속된 각 기기들을 버스에서 분리하고 각각의 기기들이 갖고있는 정보의 송수신 프로토콜을 알아내는 것이다. 이때 프로토콜 분석기는 콘트롤러 모드, 리스너 모드를 모두 가지고 있으며 자유로이 그 상태를 바꿀 수 있고, 리모트와 로컬을 합해 놓은듯한 방식으로 운용된다. 그리고 이에 1 : 1로 물려 있는 기기는 각각 지정된 상태에서 동작이 되기 때문에 널리 사용되는 프로토콜을 사용하지 않았을 경우에는 예상되는 테스트 패턴을 가지고 기기를 시험하여 그 기기가 가지고 있는 정보 송수신용 프로토콜을 찾아낸다. 정보 송수신용 프로토콜에는 커맨드 송수신 모드에서의 프로토콜과 데이타 송수신 모드에서의 프로토콜로 나누어지는데 각각에서의 예상되는 테스트 프로토콜 패턴은 사용자에 의하여 입력된다. 이에 대한 알고리즘은 그림12와 그림13에 나타내었다. 찾아진 프로토콜은 마지막 기기에 대한 프로토콜을 자동적으로 시험하기 위한 테스트 프로토콜 패턴을 기록되며 이를 사용자에게 보여준다.

```
// 명령 송수신 모드 프로토콜 //
// test_com_trans_pattern[ ] : 테스트 커맨드 모드
   프로토콜 패턴 //
// input_pattern : 입력되는 커맨드 모드 프로토콜 예
   상 패턴 //
 {
  for i = 1 to # of test_com_trans_pattern
       test_check(test_com_trans_pattern) ;
  end_for

  test_check(input_pattern) ;
 }
 test_check(test_pattern)
  {
   loop ( # of each pattern sequence)
   BMR ← test_pattern[i] ; // 테스트 패턴을 버스상에
                              출력 시킨다.//
       do 3_line handshake ;
           if (detect_bus_dead_lock)
              dead_lock = true
              exit loop ;
           end_if
   end_loop
       if (dead_lock = false)
      print("Find the test_com_trans_pattern.") ;
      exit end ;
       end_if
   }
```

그림 12. 명령 송수신 모드 알고리즘
Fig 12. Mode algorithm of command transceiver

```
// 데이타 송수신 모드 프로토콜 //
// test_com_trans_pattern[ ] : 테스트용 데이타 모
   드 프로토콜 패턴 //
// input_pattern : 입력되는 데이타 모드 프로토콜 예
   상 패턴 //
 {
  for i = 1 to # of test_data_trans_pattern
       test_check(test_data_trans_pattern) ;
  end_for

  test_check(input_pattern) ;
 }
  test_check(test_pattern)
   {
    loop( # of each pattern sequence)
         DBR ← test_pattern. DBR[i] ; // 테스트 패
         턴을 버스상에
         DMR ← test_pattern. DMR[i] ; // 출력 시
         킨다.//
       do 3_line handshake ;
            if(detect_bus_dead_lock)
               dead_lock = true
               exit loop ;
            end_if
```

```
end_loop
    if(dead_lock = false)
  print("Find the test data_trans pattern.") ;
  exit end ;
    end_if
)
```

그림 13. 데이타 송수신 모드 알고리즘
Fig 13. Mode algorithm of data transceiver

Ⅳ. 실험 및 검토

현재의 CAT 시스템은 여러대의 계측기를 단일 프로그램으로 구동시키도록 구성되었다. 이러한 문제로 인하여 하나의 실험 모드를 테스트하기 위해서는 측정할 실험 모드에 알맞은 계측기를 구동하기 위해 컴퓨터는 하나의 프로그램만을 실행시키도록 되어 있기 때문에 동시에 두 대의 계측기를 구동하지 못한다. 예를들어, 하나의 실험 모드에 주파수와 전력을 측정할 경우 현존의 시스템은 주파수만을 측정하고 전력을 측정할 경우 주파수 정보를 없애고 전력을 측정할 수 있도록 구성되어 있다. 그래서 본 논문에서는 이러한 점을 보안하고 계측기간의 상호 데이타 교환을 위하여 다중처리로 프로그램을 수행할 수 있도록 하였다. 다중 프로그램을 수행 시킴으로써 사용자는 메시지 손실없이 다른 계측 상태를 관측할 수 있다는 장점이 있고, 또한 UNIX 시스템이 아닌 DOS 시스템 상태에서 멀티 태스크를 실현함으로서 사용자가 쉽게 접하기 쉬운 PC를 가지고 동시에 여러대의 계측기를 실험할 수 있도록 구현하였다.

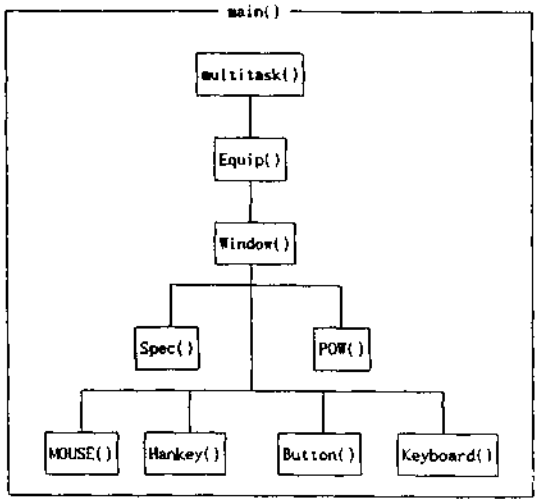


그림 14. 계측기 제어 프로그램 구조
Fig 14. Structure of control program for instruments

이러한 다중 프로그램을 실행하기 위한 소프트웨어 구조는 그림14에 나타내었다. 그리고 다중 처리를 수행하기 위한 전체 흐름도는 그림15와 같다.

Spec( ) 부터은 스펙트럼 분석기를 초기화 시키고 구동시키는 역할을 하며 제어 대상이 되는 주파수, 대역폭, 이득, 감쇄에 대한 제어를 담당한다. Pow( ) 부터은 파워미터를 구동시키는 역할을 하며 제어 대상이 되는 주파수, 전력, 조정(cal.)을 담당한다. 사용자는 마우스만으로 모든 제어를 행할 수 있다.

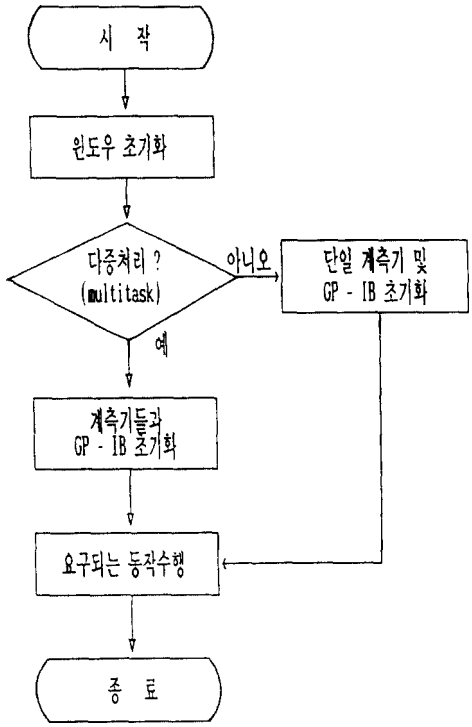


그림 15. 멀티 태스크 수행을 위한 전체 흐름도
Fig 15. Total flowchart for multitask

상기 구조와 같이 다중 프로그램을 작성하고 수행시켰을 경우에 각각 디바이스 명령을 전송한 결과 아무 이상없이 데이타를 전송함을 알 수 있다. 물론, 각 디바이스에 따라 전송하는 속도는 차이가 날 수 있으나 계기를 측정하는데는 별 다른 지장을 주지 않는다. 또한 본 논문에서 측정한 계측기는 두대로 제한

했으나 더 많은 계측기를 연결하여 다중처리를 할 때의 처리 속도가 늦어질 뿐 아무 상관이 없다. 표1은 작업수에 따른 작업 처리 속도를 나타내며 그림16은 구현된 CAT 시스템의 블럭도이다.

표 1. 작업 수에 따른 작업 처리 속도.
Table 1. Processing speed according task number

동시에 처리할 작업의 수	처리속도
2 대	500 kbps
4 대	250 kbps
8 대	175 Kbps
10 대	50 Kbps
15 대	28 Kbps

또한, 이미 알려진 자료를 기초로 하여 작성된 정보 전송용 예상 프로토콜을 가지고 각 기기에 대해서 실험한 결과 그림18과 같이 공통된 정보 전송 프로토콜을 찾아내었다.

따라서 제작회사에 서로 다르게 만들어진 기기라도 프로토콜만 제대로 찾아서 프로그램을 제대로 해준다면 이상없이 원하는 동작을 시킬 수 있음을 알 수 있었고 또한, 디바이스 명령이 복잡하거나 기기가 복잡한 수행을 하게되면 상대적으로 정보에 대한 반응이 느려짐을 알 수 있었다. 그림19은 다중 프로그래밍을 수행시켰을 때의 동작하는 화면 상태를 보여주고 있다.

V. 결 론

본 논문에서는 컴퓨터를 부가한 계측 시스템을 구성하여 각종 장비를 연결, 통합 계측이 가능하게 하고 컴퓨터가 갖는 고속 처리 능력을 이용하여 검사시간을 감소시킨 만큼 생산성의 향상에 크게 기여할 수 있도록 CAT 시스템을 구축하였고 시스템을 운용하기 위한 소프트웨어를 개발하였다. 또한, 다중처리를 위한 프로세스 기법은 트레드 베이스 방식을 사용하

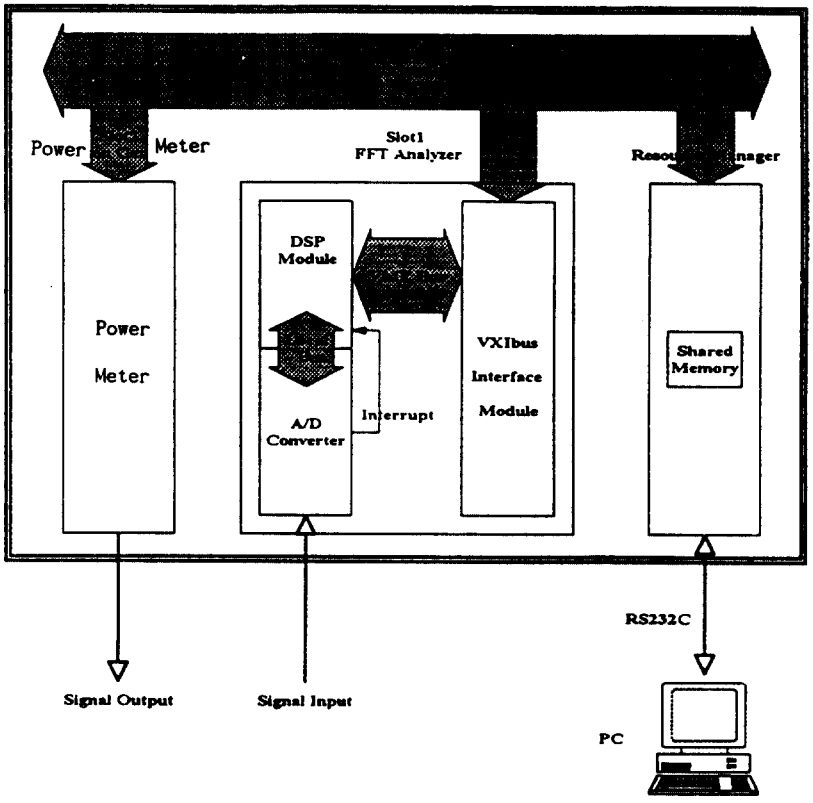
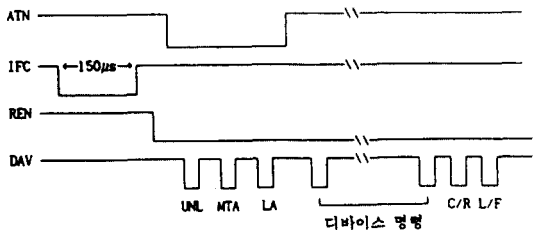
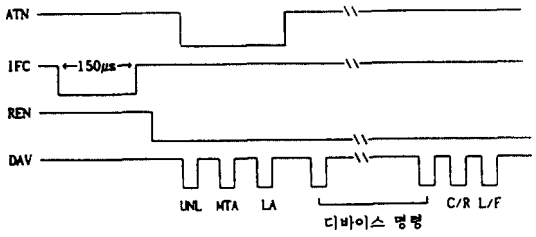


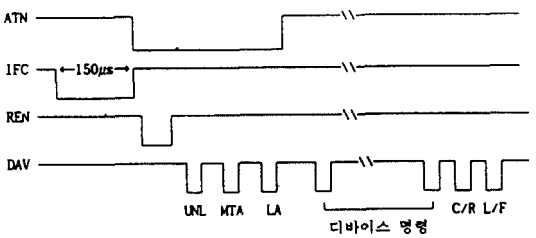
그림 16. 구현된 CAT 시스템의 블럭도
Fig 16. Configuration of CAT system



예상 프로토콜 ①



예상 프로토콜 ②



예상 프로토콜 ③

그림 17. 정보 전송용 예상 프로토콜 패턴
Fig 17. Predicted patten of protocol using information transmission

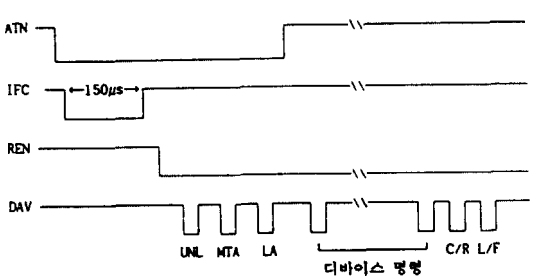


그림 18. 찾아낸 정보 전송 프로토콜 패턴
Fig 18. Patten of protocol for reguired information transmission

그림 19. 다중 프로그램을 수행시킨 후의 화면상태
Fig 19. CRT display state after excution of multitask program

였고, 스케줄러는 라운드 로빈 방식으로 각 작업에 유효 시간을 할당할 수 있도록 하였으며 동시에 여러 대의 계측기를 사용하여 테스트할 수 있도록 다중처리 프로그램을 구현하였다. 구현된 다중처리 시스템은 HP8590A 스펙트럼 분석기와 HP473B 전력계를 시스템 운용 계측기로 이용하였다. 이 외에도 15대까지의 계측기를 첨가하여 시스템을 운용할 수 있도록 하였다. 개발된 소프트웨어는 총 9개 모듈로 구성되어 있고 각 모듈들은 상호 공유되도록 구현하였다. 더 많은 계측기를 부가하려면 목적 계측기에 알맞는 소프트웨어 모듈을 합하여 구성할 수 있도록 하였다. 또한, 계측기를 제어하여 정보 전송 프로토콜의 패턴을 찾을 수 있었으며 각 기기의 디바이스 명령을 전송한 결과 에러없이 정보를 전송함을 알 수 있었다.

본 논문을 통해 개발된 시스템은 시간 절약과 사용자의 잘못으로 인한 오류를 방지할 수 있으며 테스트 검증에 걸리는 시간의 절감 효과를 가져올 수 있고 원하는 개발 시스템을 보다 더 빠르게 구현할 수 있다. 또한 사용자는 전문가적인 지식없이도 쉽게 계측기를 제어할 수 있다.

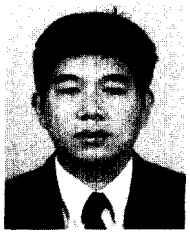
그리고 이와 같은 연구를 토대로 모듈화된 디바이스를 연결하는 VIX 버스와의 연계 시스템을 고려하면 더욱 향상된 시스템을 구현할 수 있다.

참 고 문 헌

1. IEEE Standard digital Interface for Programmable Instrumentation, IEEE Std New York, 1978.
2. Narbert laengrich, "Assuming Measuement Accuracy in IEEE 488 Based ATE System," Test & Measurement World, pp 102-135, May 1985.

3. VXIbus Consortium, VXIbus System Specification Revision 1.3, Jul. 1989.
4. 김영빈, 사용자 인터페이스의 새 물결 GUI, PC 어드벤스, 교학사, October 1990.
5. David A Howorth, An Architecture for Modular Instruments, Textronix, 1988.
6. P. Anderson, GPIB Cntrolled electrical engineering Lab., Frontiers in Education conference, 1989.
7. 심용득, IBM-PC용 GPIB 접속장치에 관한연구, 한국전자통신연구소 연구보고서, 1982. 2.
8. 편집부편, IEEE 488 표준 디지털 버스-계측 제어용 표준버스의 기초에서 설계까지-, 가남사, 1989. 4.
9. トランジスタ 技術, GPIB Interface, pp. 349-359, Jan. 1984.
10. ASIC Lab. Designed for HP-IB systems, Hewlett Packard Corp., 1990.
11. The General Purpose Interface Bus, IEEE MICRO Electronics, Feb. 1982.
12. Peter Coad and Edward Yourdon, Object-Oriented Analysis, Prentice Hall, pp 82-91, 1990.
13. Danforth, Scott and Tomlinson, Chris, "Type Theories and Object Programming," ACM Computing Surveys, pp 15 21, March 1988.
14. Peter Coad and Edward Yourdon, Object-Oriented Design, Prentice Hall, 1991.
15. Edmund W.Faison, Jr., Graphical User Interface with Turbo C++, SAMS, 1991.
16. David A Howorth, "An Architecture for Modular Instruments," Textronix, 1988.
17. John Novellina, "New Bus Arbitration Extended VXIbus," Electronic Design, pp 87-100, Apr., 1989.
18. Michael Tisher, PC System Programming for Developers, Abacus, 1989.

▲전 동 근



1986년 2월 : 고려대학교 전자공학과 졸업(공학사)
1988년 8월 : 고려대학교 대학원 전자공학과(공학석사)
1992년 8월 : 고려대학교 대학원 전자공학과(공학박사)
1993년 9월 ~ 현재 : 한서대학교 전자공학과 조교수

▲문 대 철
현재 : 호서대학교 정보통신공학과 교수
(12권 4호 참조)