한국공작기계학회 '99년도 추계학술대회 논문집 pp.585~590

Windows 기반의 PC-NC 구현을 위한 실시간 시스템 구성

조인호* 편영식**

The implementation of Real time system for PCNC based on Windows
In Ho Cho*, Young Shik Pyoun**

*선문대학교 대학원 **선문대학교 기계 및 제어공학부

1. 서론

공작기계를 정밀하게 제어하는 장치인 수치제어장치(NC)는 처음 미 국방성 과제로부터 시작하여 지금까지 많은 변천을 거듭하며 발전해 왔다. 컴퓨터 하드웨어 기술과 소프트웨어 기술의 발전에 따라 보다 편리하고 고기능 고정밀성 고성능을 가진 수치제어 장치(NC)가 생산되어 보급되고 있다.

현재 수치제어 장치가 이용되는 생산현장에서는 변화하는 생산환경에 대응하는 생산시스템의 구축을 위하여 보다 용이하고 빠른 시스템 구축 기술이 필요하게 되었다. 이러한 경향에 따라 상용 PC와 호환이 되거나 PC 그 자체를 이용한 제어기기의 구축을 통하여 각 제어 하드웨어와 소프트웨어 양면에서 Open화가 이루어져 왔고, 이러한 산업 제어기기의 동향에 따라 PC를 기반으로한 수치제어장치가 연구 개발되어 수치제어장치에서 Open화를 구현하려고 시도하고 있다.

현재 보급되어 있는 수치제이장치는 전용화되어져 있어서 필요한 지원 프로그램이나 하드웨어를 사용자가 원하는 데로 추가하는 것이 어렵고 반드시 생산자의 지원을 받아야만 하였다. 그러나 PC를 기반으로 하는 수치제어장치는 하드웨어 및 소프트웨어의 추가 및 제거가 보다 용이하며, 생산시스템 구축시 다른 용도로 제어장치의 변환이 용이하므로 추가적인 비용과 노력을 최소화할 수 있는 장점이 있다.

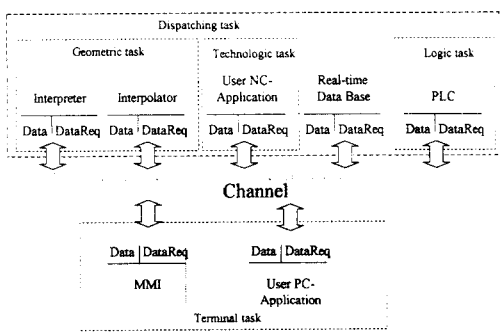
현재 PC를 기반으로 하는 수치제어장치 개발의 가장 특징적인 점은 기존보다 뛰어난 MMI(Man Machine Interface)를 가지고 있다는 점이다. 이러한 장점 때문에 사용자는 보다 이용하기 쉽게 되었다. 이러한 뛰어난 MMI를 구현하기 위하여 상용화되어 널리 이용되고 있는 OS(Operating System)를 이용한 시스템의 구축이 필요하다. 이러한 OS를 이용함으로써 얻어지는 이점은 이 기종간의 데이터의 호환성이 좋아질 뿐만 아니라, 보다 풍부한 환경을 지원하기 때문에 사용자가 이용하기 쉬운 장점이 있다.

그러나 이러한 상용 OS를 채택하게 될 때 중요한 점은 공구 경로 제어를 위한 실시간 처리 부분이다. 이 부분이 시간에 대한 제한을 가지고 있기 때문이다.

실시간의 처리 지원문제는 하나의 CPU를 가진 PC-NC를 구현할 때 가장 중요한 문제로 나타난다. 실제 상용 OS중 Windows 95/98또는 NT를 이용하여 PC-NC를 구현한다면 이러한 OS가 실시간을 지원해주지 못한다는 단점이 있어서 실시간 처리 문제를 따로 보장하는 방법을 써야하는데 아직까지 이러한 문제에 대한 연구는 발표되어지지 않았다. 본 논문에서는 Windows 95/95 , NT를 이용하여 하나의 CPU를 가진 PC-NC를 구현할 때 PC-NC 각 모듈의 시간 제한을 조사하고 적정한 응답성을 가진 보다 경제적인 시스템 구성방안에 대하여 논의하고자 한다.

2. PC-NC를 위한 실시간 시스템

2.1 PC-NC의 모듈 구조



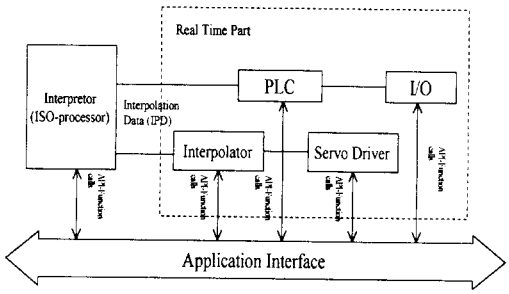
<그림 1> PC-NC의 모듈 구조

<그림 1>은 PC-NC의 모듈 구조를 나타낸다. PC-NC는 Interpreter, Interpolator, PLC, MMI, User Application등의 모듈로 구성되어져 있다. 위의 구조는 수치제어장치를 구성하는 기본 기능

들을 모듈 구조로 구현하여 Open화를 실현하였다. 위와 같은 구조의 특징은 각각의 기본 모듈의 추가 및 제거가 쉬워 사용자가 원하는 시스템의 구축이 빠르고 쉽다는 것이다. 현재 PC-NC의 구조는 이와 같은 모듈 구조를 지향하고 있으며, 사용자가 필요한 모듈의 첨가가 자유롭게 되어있어, PC에서 다른 시스템의 구성이 용이하게 구현된다.

2.2 PC-NC의 실시간 처리 필요 모듈

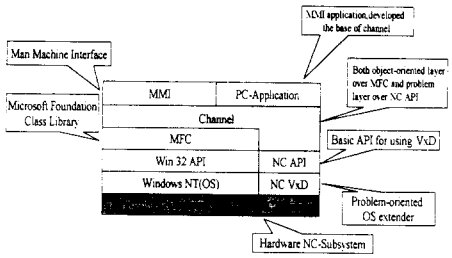
앞절 2.1의 모듈 구조중에 항상 실시간으로 처리하여야 하는 것은 아니다. 실시간 처리 부분은 외부센서들과 Actuator들을 제어하는 PLC와 Servo Driver를 제어하여 최종적으로는 공구경로를 제어하는 부분이다. 또한 이와 연관되어 Servo Driver에 공구 경로 data를 생성한 후 보내주는 Interpolator부분도 실시간으로 처리되어야 한다. 다음 <그림 2>는 실시간 처리부의 간략한 다이어그램이다.



<그림 2> PC-NC의 Real Time Part

2.3 PC-NC의 수직구조

앞절 2.1에서 설명한 모듈 구조를 가지기 위하여 Windows 기반 PC-NC는 <그림 3>과 같은 수직구조를 가진다.



<그림 3> PC-NC의 수직구조

실시간 처리부는 위의 수직구조에서 NC 하드웨어 바로 위 계층인 NC VxD 또는 NC Device Driver 부분이 된다. 이 계층위에 실시간으로 실행되어야 하는 모듈이 동작하게 된다.

2.4 PC-NC를 위한 실시간 시스템

하나의 CPU를 가진 PC-NC를 구현하기 위하여서는 같은 환경에서 MMI와 실시간 처리 모듈이 같이 움직여야 한다. 이러한 환경을 구성하기 위한 방법에는 크게 두가지 방법이 있다.

첫 번째는 전체 시스템을 실시간 시스템으로 구현하는 것이다. 이 방법은 실시간 처리가 필요없는 부분까지 지원해 주어서 시간제약이 너무 크고 이로 인하여 발생하는 Interrupt가 많아 지기 때문에 시스템의 효율이 떨어지는 단점이 있다. 상용 실시간 운영체제를 이용하기 때문에 도입가격이 비싸게 된다.

두 번째는 전체 시스템을 실시간 지원이 필요한 부분과 그렇지 않은 부분으로 나누고 각각 따른 시간 처리 제약을 두는 방법이다. 이 방법은 MMI와 시간에 대한 제약이 엄격하게 적용되지 않는 부분은 일반 사용OS로 이용하고 시간제약이 엄격하게 적용되어야 하는 부분은 부가적인 지원을 통하여 실시간의 시간제약을 가질 수 있도록 하는 것이다. 그러나 이러한 시스템의 단점은 진정한 실시간 시스템이 될 수 없다는 것이다. 그 이유는 그 근본 부분이 실시간 시스템이 아니기 때문이다. 따라서 이 부분은 소프트웨어적인 지원이 많이 필요하게 된다.

2.5 PC-NC를 위한 실시간 처리의 특징

PC-NC를 위한 실시간 처리의 특징은 다음과 같다.

1) 작은 시간 지연성

Servo Drvier 또는 PLC제어를 위한 모듈들은 실제 아주 엄격한 시간제한성을 가진다. 특히 공구 경로를 제어하는 부분은 가공 후 제품의
에 큰 영향을 미친다. 주어진 시간내에 주어진 일이 처리되어지지 않으면 공구 경로를 제어가 정밀하게 되지 못하여 특히 원호 보간 및 각종 복잡한 형상 가공시에는 많은 오차를 발생할 수 있다. 따라서 이러한 부분에서는 엄격한 시간제한성을 가진다. 따라서 task가 전환되어 실행되기까지의 시간지연이 작아야 한다.

2) 실시간 다중 처리

Servo Driver를 제어하는 부분에서 수치제어 장치는 3축이상을 동시에 제어해야 하므로 동시에 서로 다른 일을 수행할 수 있어야 한다. 따라서 동시에 여러 작업을 할 수 있는 다중 실시간

처리를 할 수 있는 능력이 있어야 한다.

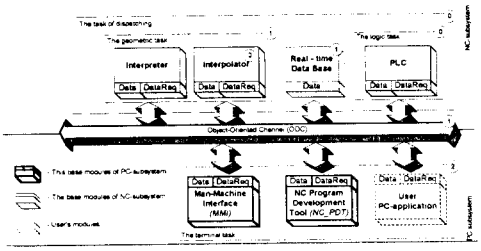
3) 하드웨어 제어의 용이성

실시간 처리가 필요한 부분은 앞에서 언급한 바와 같이 공구 경로의 제어나 PLC제어를 위한 I/O 부분이다. 이 부분은 하드웨어를 직접적으로 제어하여야 하므로 실제 하드웨어 제어가 용이하도록 실시간 라이브러리를 지원하여야 한다.

4) 보수의 용이성

실시간 제어모듈에 error가 발생했을 경우 error를 표시하고 그 부분을 찾을 수 있는 Debug기능을 제공하여서 프로그램 보수를 용이하게 할 수 있도록 지원하여야 한다.

PC-NC를 위한 실시간 시스템의 시간 제약은 <그림 4>에 나타난다. 각 프로그램 모듈마다 번호가 매겨져 있다. 이 번호는 실시간 처리에서 시간제약을 나타내고 있다.



<그림 4> PC-NC의 각 모듈의 시간 제약

레벨 2 : 시간지연이 수십 ms의 정도 되는 시간제약을 가지며 Win32 API process를 이용한 화면의 update의 예를 들 수 있다.

레벨 1 : 시간지연이 수백 μs정도의 시간제약을 가지는 soft real time인 경우이다.

레벨 0 : 시간지연이 수 μs정도의 시간 제약을 가지는 hard real time인 경우이다.

PC-NC은 이와 같은 시간제약을 가지고 각각의 프로그램 모듈을 구성해야만 한다.

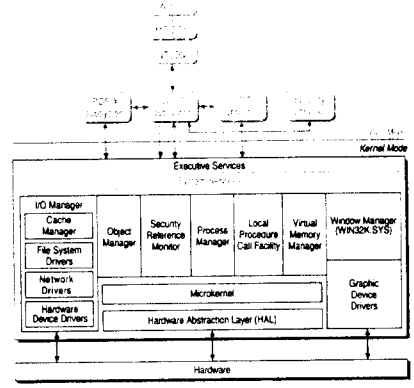
3. Windows 기반에서 시간 응답성

실제 Windows는 운영체제 자체에서 real time processing을 지원하고 있지 못하다. 이러한 단점 때문에 3가지 방법으로 Windows 시스템의 응답성을 검사하였다.

3.1 Windows 95와 NT의 구조

Windows 운영체제는 95/98과 NT가 있다. 95/98과 NT는 Workstation이라는 측면에서 큰차이가 있다.

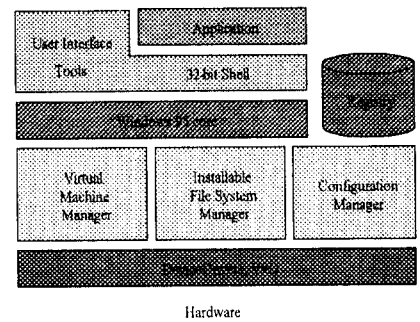
<그림 5>는 Windows NT의 구조를 보여주고 있다. Windows NT의 가장 큰 특징중에 하나는 보안성과 isolation이다. 보안성은 Windows 95나 98과는 달리 저장되어있는 자료의 보안성이 뛰어나다. 그리고 isolation은 <그림 5>에서 보는 것과 같이 Kernel부와 API 부가 분리되어 있다는 것이다.



<그림 5> Windows NT의 구조

이러한 구조로 되어져 있는 특성 때문에 프로그램에서 하드웨어를 직접 제어하기 힘들다. Kernel의 최하위층인 HAL(Hardware Abstraction Layer)이 실시간 타이머를 지원하고 있지 않기 때문에 실시간 처리를 지원할 수 없다.

<그림 6>은 Windows 95의 구조를 간략하게 그린 것이다. Windows 95는 NT보다 하드웨어와 소프트웨어의 고립성과 보안성이 떨어진다. 그러나 실제 응용프로그램에서 하드웨어를 직접 제어할 수 있는 특징을 가지고 있다.



<그림 6> Windows 95의 구조

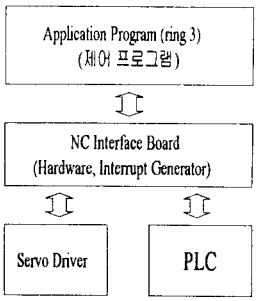
3.2 Windows 95에서 Interrupt에 대한 응답성

Windows 95에는 실제 응용프로그램이 하드웨어를 제어할 수 있기 때문에 두가지 방법으로 Interrupt에 대한 응답성을 살펴보았다. 첫 번째 방법은 실제 응용 프로그램에서 직접 하드웨어를 제어하는 방법이고, 두 번째 방법은 시스템 프로그래밍을 통하여 시스템에 하드웨어 제어모듈을 포함시키는 방법이다.

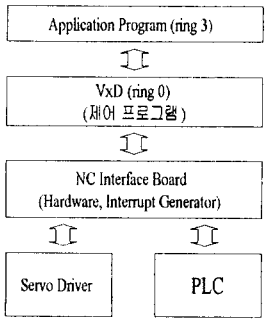
첫 번째 방법을 이용하여 직접 응용프로그램에서 하드웨어의 I/O 번지에 읽고 쓰기를 일정한 Interrupt 2ms 주기마다 반복한 경우 Pentium 133MHz에서 최대 4ms의 task전환시간을 측정하였다.

두 번째 방법을 이용하여 가상장치 드라이버를 만들어 시스템에 삽입한 후 하드웨어 I/O 번지에 읽고 쓰기를 일정 Interrupt 2ms 주기마다 반복한 경우 첫 번째와 같은 기종에서 500μs로 task전환시간이 측정되었다. 두 실험 모두 같은 환경에서 현재 주어진 작업 외에 다른 process나 thread가 실행되지 않았을 경우에 얻어진 결과이다. Windows 95에서 다른 process나 thread가 발생한다면 현저하게 그 응답성이 떨어지는 것을 확인할 수 있었다.

<그림 7>과 <그림 8>은 첫 번째 와 두 번째 방법의 제어구조를 보여준다.



<그림 3-6> VxD를 이용하지 않은 제어구조



<그림 3-7> VxD를 이용한 제어구조

3.3 Windows NT의 multimedia timer기반 프로그램에서의 시간 응답성

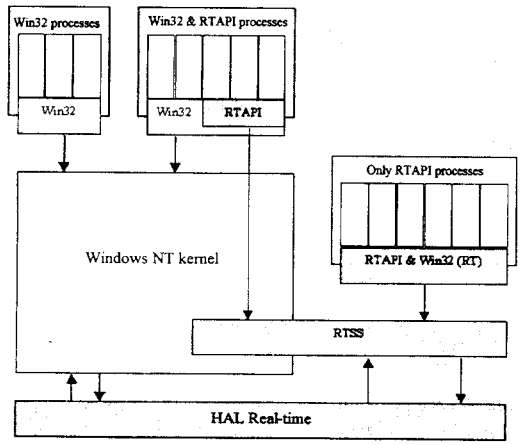
Windows NT의 Multimedia timer는 Sound의 Sampling이나 동영상 화면을 보여줄 때 많이 이용되는 timer로서 시간과 데이터가 동기되어야 하는 특징을 가지고 있다. 이러한 특성을 이용하여 필요한 데이터를 외부 하드웨어에 일정한 시간에 내보낼 수 있다. 하드웨어의 제어는 이미 알려진 I/O 번지에 제어하고자 하는 값을 쓰는 것이기 때문에 충분히 하드웨어의 제어에 multimedia timer가 적용될 수 있다.

시간 응답성에 대한 실험은 multimedia timer를 기반으로 Interrupt를 2ms마다 발생하여 하드웨어의 I/O 번지 읽고 쓰는 일을 수행하게 하였다. 이 실험은 수십 μs ~ 500μs의 결과를 얻을 수 있었다. 실험 환경은 Windows 95의 실험 방법과 동일하며, 다른 process나 thread의 실행도 방지하였다.

3.4 Real time extension을 가진 Windows NT 기반 프로그램에서의 시간 응답성

Real time extension을 가지고 있는 Windows NT의 시간 응답성을 확인하기 위하여 현재 상용화 되어져 있는 제품을 이용하였다. 미국 Venturcom에서 만든 Real time extension(RTX)은 Widnows NT를 기반으로 하여 HAL을 바꾸어서 실시간을 지원하게 하였다. <그림 9>는 Extension의 구조를 보여주고 있다.

RTX는 RTSS로 알려져 있는 실시간 subsystem을 Windows NT에 추가하도록 구성되어져 있다. RTSS는 다른 Windows NT와 같은 subsystem과 개념적으로 유사하며, 자신의 실행환경을 제공하고 있다. 그리고 API.RTSS는 Windows NT의 스케듈러를 사용하지 않고 RTSS가 수행하는 자체 실시간 thread 스케듈링을 사용하고 있다. 또한 RTSS가 수행하는 스케듈링은 모든 Windows NT 스케듈링에 앞서서 발생하게 된다. 물론 Windows NT가 관리하는 인터럽트나 DPCs(Differed Procedure Calls)도 포함된다.



<그림 9> RTX의 구조

또한 RTSS는 실시간 IPC(Inter-Process Communication) Object를 지원하여 RTSS와 Win32 process간의 통신도 가능하게 하였다.

실제 Windows NT에서 하드웨어를 제어하기 위해서는 Devic Driver가 필요하게 된다. 앞의 3.1절에서 언급한데로 Windows NT의 구조상 Kernel과 API가 분리되어져 있기 때문에 Device Driver를 이용하여서만이 하드웨어를 직접 제어할 수 있다.

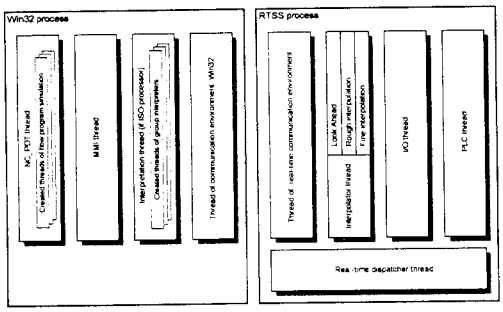
그러나 RTX에서는 특별히 실시간 제어 기능들을 라이브러리로 제공하고 있어서 하드웨어를 직접 작성할 수 있다.

RTX 응용프로그램에서 일정 Interrupt 주기 2ms마다 I/O의 읽고 쓰기의 시간의 응답성은 10 μs 이내로 측정되었다.

4. Windows NT extension을 이용한 PC-NC 실시간 시스템의 효율적인 구현

4.1 PC-NC의 process 와 thread

PC-NC의 process와 thread는 두 개의 그룹으로 나뉜다. <그림 10>은 Win32 와 RTSS 두 개의 그룹의 process를 나타내고있다.



<그림 10> Win32와 RTSS process와 thread

PC-NC의 실시간 시스템을 구축하기 위하여 2.4절의 실시간 처리의 각 모듈별 시간 제약에 따라에 process와 thread를 구성하였다.

Win32 process는 PC application thread, MMI main thread, ISO-processor thread, RTSS communication thread등으로 Soft real time의 시간제약을 가지고 있다.

RTSS process는 Win32 process와의 통신 thread, PLC thread, 실시간 관리자 thread, interpolator thread 등으로 Hard real time의 시간 제약을 가지고 있다.

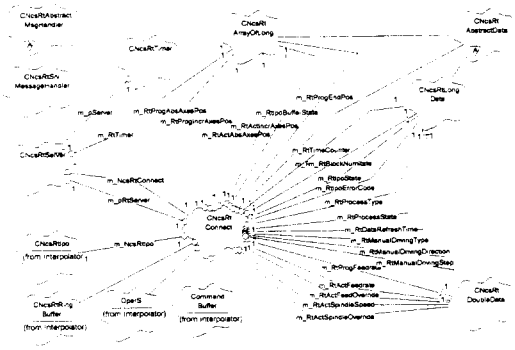
4.2 실시간 관리자

실시간 시스템에서 각 부분의 실시간 처리를 관리해 주는 부분이 있는데 이것이 실시간 관리자라고 한다. 시간에 따라 각각의 작업을 각 실시간 모듈에 할당하는 역할을 하고 있다. <그림 11>는 실시간 관리자의 기능구조를 보여주고 있다.

실시간 관리자의 가장 중심된 기능은 실시간 타이머의 기능을 가지고 있고 가장 높은 우선순위를 가지고 있다. 타이머는 실시간처리가 필요한 시간 주기를 가지고 있고 가장 높은 우선순위를 가지고 있기 때문에 일정한 시간마다 필요한 기능들을 수행할 수 있다.

4.3 실시간 Subsystem

<그림 12>은 실시간 subsystem의 구현을 위한 rose diagram을 보여주며 아래는 실시간 Timer를 구현한 부분이다.



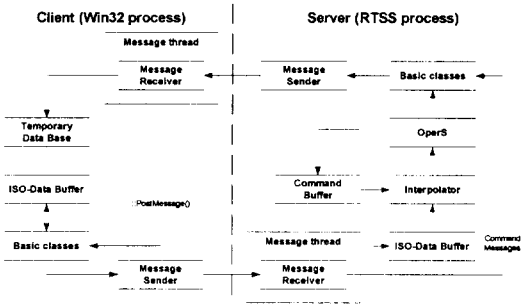
<그림 12> 실시간 Subsystem의 rose diagram

```
CNcsRtTimer m_RtTimer;
m_RtTimer.SetServer(this);
// Setting the pointer to the server.
m_RtTimer.SetPhasNum(20);
// Setting the number of timer phases.
BOOL bStartTimer =
            m_RtTimer.SetSampleTime(1*1000);

// Setting the timer period (1mS)
if (!bStartTimer)
{
// The timer has not been created: handling
errors.
}
```

4.4 RTSS process와 Win32 process의 상호 통신

실제 PC-NC가 움직일 때 서로다른 시간 제약을 가진 두 개의 시스템이 서로 단독적으로 움직이는 것이 아니라 서로 영향을 주고 받으면서 움직인다. <그림 13>는 Win32 process와 RTSS process가 어떻게 상호작용을 하면서 움직이는 지를 보여주고 있다.



<그림 13> 실시간 시스템과 다른 PC-NC 모듈간의 상호작용

앞의 4.1에서 언급한 바와 같이 각각의 process는 다른 process와의 통신을 담당하는 thread가 각각 존재한다. 이 두 thread는 다른 시스템으로부터오는 메시지를 받거나 다른 process로 가는 메시지를 전달하는 역할을 하게 된다. 따라서 Win32 process와 RTSS process에서는 통신을 담당하는 thread를 class로서 포함하여 사용하여 상호통신을 한다.

4.5 시스템 성능 실험

위와같이 실시간 시스템을 구성하여 두가지의 성능 테스트를 하였다. 첫번째는 시스템이 효율적으로 동작하는 지를 실험하였다. 실험은 실시간 운전인 interpolation을 수행하고 모든 MMI가 정상적으로 움직이는 상태에서 CPU의 활용도를 측정하였다. 측정한 결과는 Pentium 100 processor에서 약 40%의 활용도가 측정되는 결과를 얻을 수 있었다. 아직 PLC가 구현이 안된 상태이기 때문에 CPU의 활용도가 낮게 측정된 것으로 생각되지만 현재 측정에 쓰인 하드웨어보다 수 배의 성능을 가진 하드웨어가 생산되기 때문에 시스템의 효율이 높다는 것을 알 수 있다.

첫 번째 실험 결과를 만족하면서 interrupt 처리의 응답시간을 측정하였다. 이 방법은 구현된 PC-NC의 Interpretation과 Interpolation을 수행하게 하면서 하드웨어를 제어하기 위하여 I/O address에 임의의 값을 쓰는 실험이다. 임의의 I/O address에 0과 1을 번갈아 씀으로써 그 신호 주기를 오실로스코프로 측정하였다.

측정결과는 2ms의 interrupt 주기에 따라 실제 I/O address에 4ms주기의 구형파가 나오는 것을 측정하였고 최소 1ms 주기의 interrupt를 부가하여 2ms의 구형파가 나오는 것을 측정하였다.

5. 결론

현재 하나의 CPU를 가진 PC-NC의 개발에서 가장 중요한 부분의 하나가 실시간 시스템의 구축이며 PC-NC를 구성하는 모듈의 실행시 필요한 시간 제약에 따라 시간 응답성을 최적화함으로서 보다 효율적인 구성이 가능하다.

사용하기 쉬운 MMI, 배우기 쉬운 수치제어 장치를 구현하기 위하여 Windows NT를 기반으로 하는 수치제어장치를 구축하였다. 시스템의 시간 응답성의 효율을 위하여 시스템을 구성하는 각 모듈마다 다른 시간제약을 갖도록 설계하였다. 상용 Real time extension을 이용하여 시스템 각 모듈의 시간제약에 따라 서로 다른 처리 시간을 적용하여 전체 효율을 높일 수 있었다. 전체 시스템의 동작시 약 40%의 CPU활용도가 측정되었다.

앞으로 PLC 기능을 추가하여 완전한 NC의 기능을 구현하여야 하며, 보다 시스템의 효율과 경제성을 높이기 위하여 전체 구성을 최적화해야 할 것이다.

참고 문헌

[1] 무토 카즈오, "NC의 Open화(1)",일본 M&E 11월호,pp102-113, 1996

[2] 青木一信, "퍼스컴 NC", 일본 M&E 11월호, pp114-119, 1996

[3] Walter Oney, "System Programing for Windows 95", Microsoft Press, pp11-102, 377-424, 1996

[4] Microsoft, "Windows NT resource kit", microsoft press, 1996, pp90 -118

[5] Venturcom, "RTX user's guide and reference", 1997, p3-6

[6] 조인호,"PCNC Interface board를 위한 Windows 95용 가상장치 드라이버의 개발",1999, pp3-6

[7] "Open Architecture Controller/ Manufacturing Operating System (OAC, MOS), Version 1.2, LLNL, LANL, ICON Ind. Controls.Co, 1997