論文 한국정밀공학회지 제10권 제4호 1993.12
Journal of the Korean Society of Precision Engineering Vol. 10. No.4 Dec. 1993.

다중처리기를 갖는 고성능 범용제어기의 개발과 여유자유도 로봇 제어에의 응용

박 주 이*, 장 평 훈*

Development of High Performance Universal Controller Based on Multiprocessor

J. Y. Park*, P. H. Chang*

ABSTRACT

In this paper, the development of a high performance flexible controller is described. The hardware of the controller, based on VME-bus, consists of four M68020 single-board computers (32-bit) with M68881 numerical coprocessors, two M68040 single board computers, I/O devices (such as A/D and D/A converters, parallel I/O, encoder counters), and bus-to-bus adaptor. This software, written in C and based on X-window environment with Unix operating system, includes : text editor, compiler, downloader, and plotter running in a host computer for developing control program; device drivers, scheduler, and mathematical routines for the real time control purpose; message passing, file server, source level debugger virtural terminal, etc. The hardware and software are structured so that the controller might have both flexibility and extensibility. In parallel to the controller, a three degrees of freedom kinematically redundant robot has been developed at the same time. The development of the robot was undertaken in order to provide, on the one hand, a computationally intensive plant to which to apply the controller, and on the other hand a research tool in the field of kinematically redundant manipulator, which is, as such, an important area. By using the controller, dynamic control of the redundant manipulator was successfully experimented, showing the effectiveness and flexibility of the controller.

Key Words : Controller(제어기), Redundant Robot(여유자유도 로봇), VMEbus(VME버스), Multiprocessor(다중처리기)

1. 서 론

과거에는 로봇등 어떤 제어대상이 주어지면, 그 대상에 맞는 특수한 제어기를 구성하는 것이 하나의 관행이었다. 그러나 로봇등 제어대상의 종류가 다양한데 반해, 제어기는 거의 동일한 구조를 가지고 있다는 인식

* 한국과학기술원 (정밀공학과)

이 확산되면서 특수한 제어기 대신 범용제어기를 개발하는 방향으로 추세가 바뀌고 있다. 이와 함께, 제어기는 지속적인 발전을 위한 확장성에 대한 요구와, 복잡한 구조의 로봇 및 복잡한 제어알고리즘을 적용하기 위한 고속의 처리속도에 대한 요구를 만족시키도록 개발되고 있다.

이러한 추세로 많은 제어기가 연구되고 발표되었는데, 그 중에 하나로 MIT 대학에서 개발된 Condor 시스템이 있다[1]. 시스템은 Narasimhan 등에 의하여 개발된 것으로서 범용제어기로 개발되어 실제로 많은 로봇을 제어하는데 사용되었다. 이 시스템은 또한 고속의 계산을 위해 다중처리기를 사용하였고, 표준버스인 VME 버스를 사용하여 확장성을 갖도록 하였다. 또, X윈도우의 그래픽 환경안에서 다양한 개발환경과 사용환경을 제공한다. 이와 유사한 시스템으로 Carnegie Mellon 대학의 Stewart 등에 의해 개발된 Chimera 시스템이 있다[2]. 이 시스템도 역시 VME 버스를 기반으로 범용성과 확장성 그리고 계산성능을 고려하여 개발되었는데, 특히 하드웨어와 소프트웨어의 계층적 구조를 구현하였다. 국내에서도 포항공대에서 실시간 운영체제를 이용한 범용 로봇제어 언어의 개발에 대해서 논문을 발표했다[3]. 한편 PC 버스와 개인용 컴퓨터(personal computer)를 사용한 범용제어기의 개발에 관한 연구도 진행되고 있는데[4], PC를 사용하는 제어기는 계산속도 면에서 손색이 없지만, PC 버스의 자료 전송속도에 한계가 있기 때문에 다중처리 구조로 구성하고자 할 때는 많은 제약이 따른다. 이 밖에도 실시간 운영체제로서 상용으로 시판되는 VxWork, Vrtx, VMEexec 등은 VME 기반의 제어시스템을 구현하는데 유용한 개발환경을 제공한다.

본 연구에서는 MIT의 Condor와 같은 구조의 시스템을 구현하였는데, 하드웨어의 기본적인 구조는 Condor와 같지만, 다음의 하드웨어를 추가 혹은 변경함으로써 더 발전된 하드웨어로 개선하였다. 우선 호스트는 M68020 기반의 SUN3/260 대신 M60830 기반의 SUN3/470을 사용하였고, 메모리 보드는 1M byte 대신 4M byte 용량의 보드를 사용하였으며, Condor에 없던 엔코더 카운터를 추가하였다. 특히 실시간 제어를 담당하는 단일기판 컴퓨터를 M68020 컴퓨터외에 M68040 컴퓨터를 추가하였다. 따라서, 계산속도를 향상했을 뿐만 아니라 680X0 계열의 여러 처리기(processor)들을 수용할 수 있는 기반을 확립하였다. 또한 Condor 소프트웨어를 새로운 하드웨어에 맞도록 수정하였는데, 실시간 제어기와 호스트의 메세지 전달을 위해 호스트의 커널을 수정하였고, 엔코더 카운터를 위한 장치구동기(device driver)를 작성하였다. 또한, M68040을 위하여 메세지 전달 루틴 및 인터럽트 처리 루틴, 벡터 테이블, 수학함수 루틴을 작성하였다. 구현된 제어기의 특징은 VME 버스를 기반으로 구성되어 있으므로 확장성을 갖고 있고, 68040 및 68020 처리기들이 병렬로 실행되어 고속의 계산이 가능하다. 입출력 장치로는 A/D 변환기, D/A 변환기, 병렬 입출력장치, 엔코더 카운터가 장착되어 로봇과의 입출력을 행한다. 제어 프로그램의 개발 및 실행을 위한 각종 프로그램이 Unix와 X윈도우 환경에서 제공된다.

한편, 구현한 범용제어기의 성능평가 및 적용예를 보이기 위하여 이 제어기를 사용하여 기구학적 여유자유도 로봇의 위치제어를 하였다. 여기에서 기구학적 여유자유도 로봇으로서는 평면 3자유도 로봇을 이용하였다. 이미 알려진 바와 같이 여유자유도 로봇은 여유자유도를 이용해 장애물 회피, 특이점 회피등을 행함으로써 여유자유도가 없는 경우보다 더 복잡하고 정교한 일을 능수능란하게 해 낼 수 있다. 따라서 여유자유도 로봇에 대한 많은 연구가 진행되어 왔으나, 대부분이 이론과 시뮬레이션 단계이고 실험을 행한 예가 적다. 실험의 예가 적은 이유는, 첫째는 기존의 로봇중에서 여유자유도 로봇으로 분류될 만한 로봇이 적기 때문이고, 둘째는 여유자유도 로봇 이론이 요구하는 복잡한 계산을 할 만한 제어기가 적기 때문이다. 그러므로 본 연구에서는 평면 3자유도 로봇을 제작하고, 범용제어기를 사용하여 여유자유도 로봇의 이론을 적용한 위치제어를 실험하였다.

논문의 구성은 다음과 같다. 2장에서 고성능 범용제어기의 하드웨어와 소프트웨어를 설명하고 3장에서는 제작한 여유자유도 로봇의 모양 및 여유자유도 로봇의 역기구학에 대해서 소개를 한다. 4장에서 간단한 실험예를 소개하고 5장에서 결론을 맺는다.

2. 범용제어기

2.1 하드웨어

범용제어기의 하드웨어는 Fig. 1과 같은 구조로 되어 있다. 모든 하드웨어는 VME 버스상에 설치되어 있고, 실제 제어를 담당하는 부분은 VME 버스상에 있는 6개

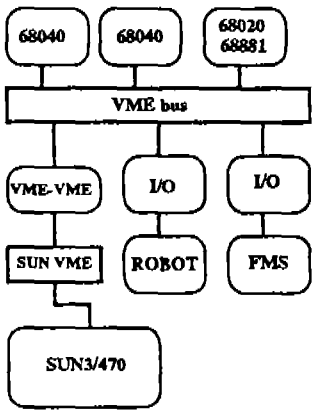

Fig. 1 Hardware architecture

의 단일기판 컴퓨터들이다. 호스트 컴퓨터는 실시간 제어에 직접적인 관련을 하지 않고, 제어 프로그램의 작성, 컴파일, 다운로딩 그리고 플로팅 등을 위해서만 사용된다. 사용자의 상황에 따라 필요한 하드웨어는 제어기의 VME 버스상에 첨가될 수 있다.

상기한 바와 같이 하드웨어의 기본적인 구조는 MIT Condor 시스템과 같지만, 성능의 향상을 위해 일부의 하드웨어는 다른 기종을 사용하였다. 우선 호스트는 M68020 기반의 SUN3/260 대신 M68030 기반의 SUN3/470을 사용하였다. 메모리 보드는 1M byte 용량의 보드대신 4M byte의 보드를 사용하여 더 많은 메모리를 사용할 수 있게 하였고, 엔코더 카운터를 추가하여 엔코더를 위치센서로 사용하는 시스템과 연결이 가능하도록 하였다. 또한, 실시간 제어를 수행하는 단일기판 컴퓨터를 Condor에 있는 M68020 기반의 IV3201외에 M68040 기반의 MVME162를 추가하였다. 따라서, 제어기의 계산속도를 향상시켰고, 다양한 680×0 계열의 단일기판 컴퓨터들을 수용할 기반을 구축하였다.

다음은 제어기의 중심이 되는 VME 버스로부터 제어를 담당하는 단일기판 컴퓨터, 제어 프로그램의 개발을 위한 호스트 컴퓨터 그리고 입출력 기기에 대하여 모듈별로 기술한다.

2.1.1 VME 버스(VME bus)

제어기의 모든 하드웨어는 VME 버스후면(back plane)에 장착된다. VME 버스는 Motorola에서 개발한 버스로서 산업체와 연구소 등에 폭넓게 사용되고 있기 때문에 필요한 기능의 하드웨어를 손쉽게 구할 수 있고 그만큼 선택의 폭도 넓다. 또한 버스의 자료 전송속도가 매우 빠르기 때문에 버스의 속도에 의한 제약이 적다. 많은 다중처리기 시스템이 처리기 자체의 계산속도 보다는 버스의 자료 전송속도에서 제한을 받는데, VME 버스는 자료의 전송속도가 타버스에 비해 빠르므로 다중처리 시스템에 유리하다. 참고로 표 1은 VME 버스와 그외 버스의 자료 전송속도를 비교한 것이다.

Table 1. The speed of various bus

Bus	Data Transfer Rate (Mbyte/sec)
PC bus	5
Multi bus I	10
VME bus	40

2.1.2 버스 제어기(bus controller)

버스 제어기는 VME 버스상의 각종 하드웨어가 요구하는 버스 사용요구신호, 인터럽트(Interrupt) 신호등이 서로 충돌없이 실행될 수 있도록 이 신호들을 관리한다.

2.1.3 단일기판 컴퓨터(single board computer)

제어시에 연산을 담당하는 부분으로, 현재 두 종류의 단일기판 컴퓨터가 장착되어 있다. 한 종류는 Motorola 68020을 중앙연산장치로 하고 실수연산장치(floating point unit)는 Motorola 68881을 사용하는 컴퓨터이고, 또 한 종류는 Motorola 68040을 CPU로 하는 컴퓨터이다. 각각의 단일기판 컴퓨터는 주 기억장치로 1메가 바이트의 램을 갖고 있다. 그런데, 이 램은 이중포트 메모리(dual ported memory)이어서 VME 버스상의 어느 기기든지 읽고 쓸 수 있는 공유메모리가 된다. 두 컴퓨터의 연산속도는 각각 M68020이 2.7 MIPS이고 M68040이 25 MIPS이다. 단일기판 컴퓨터들은 목적에 따라서 자유로이 제거와 추가가 가능하다.

2.1.4 개발 호스트(development host)

호스트로 SUN3/470 워크스테이션을 사용하는데, 이것의 역할은 제어 프로그램의 개발 및 실행에 필요한 환경을 제공하는 것이다. 사용자는 호스트 컴퓨터상의 Unix와 X윈도우 환경안에서 프로그램을 작성하고 컴파

일한 후 단일기판 컴퓨터에 다운로드 한다. 제어 프로그램이 단일기판 컴퓨터에서 실행되고 있는 동안, 사용자는 호스트의 화면을 통해서 각 단일기판 컴퓨터의 현재상황을 관찰하고, 호스트의 자판을 통하여 필요한 입력을 준다. 제어 프로그램의 실행결과로 저장된 결과치는 호스트의 X윈도우 그래픽 화면에서 그래프로 그려볼 수 있고, 호스트의 하드디스크에 저장할 수도 있다. SUN3/470은 CPU가 68030이기 때문에 단일기판 컴퓨터의 CPU와 호환이 되어서, 교차컴파일러를 사용할 필요가 없다.

2.1.5 버스간 연결기(bus to bus adaptor)

이것은 호스트인 SUN의 버스와 제어기의 버스를 연결해 주는 하드웨어이다. 이를 통해서 SUN은 제어기 VME 버스의 메모리를 SUN 자신의 램처럼 직접 읽고 쓸 수 있게 되고, 제어기와 SUN 사이에 인터럽트가 전달될 수 있게 된다. 따라서, SUN과 제어기 사이의 전송이 메모리에 읽고 쓰는 동작만으로 가능하여 고속의 자료전송이 가능하게 된다. 버스간 연결기를 통해서 선택된 인터럽트가 전달될 수 있는데, 이러한 인터럽트를 사용함으로써 뒤에 설명할 메세지 전달이 효율적으로 이루어지게 된다.

2.1.6 입출력 장치(I/O device)

입출력 장치는 필요에 따라서 첨가, 제거할 수 있다. 현재는 A/D 변환기, D/A 변환기, 병렬 입출력기 그리고 엔코더 카운터가 장착되어 있다.

2.1.7 메모리기판(memory board)

단일기판 컴퓨터의 메모리외에 메세지 전달(message passing)이나 자료의 저장등을 위해서 메모리 기판을 사용한다. 이것은 4M byte의 용량을 갖는데, 호스트와 모든 단일기판 컴퓨터로부터 읽고 쓸 수 있는 공유메모리가 된다

2.2 소프트웨어

본 연구에서는 Condor 시스템이 소프트웨어를 바탕으로, 추가 변경된 하드웨어에 맞도록 수정하였는데, Condor의 소프트웨어는 다음과 같은 개념으로 설계되었다.

- 제어 프로그램의 개발 및 실행과 평가에 편리한 환경을 제공한다.

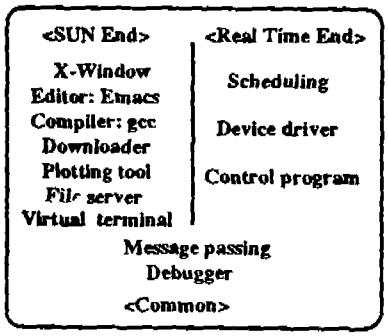

Fig. 2 Software architecture

- 유용하고 다양한 제어용 라이브러리를 제공한다.
- 고수준 언어인 C 언어만을 사용하여 제어 프로그램의 작성이 가능하도록 한다.

또한, Condor의 소프트웨어는 기능에 따라 세종류로 나눌 수 있다(Fig. 2).

- 첫째는, 개발 호스트에서 작동하는 소프트웨어로서, Unix 운영체제와 X윈도우 환경에서 제어 프로그램의 개발 및 실행 그리고 제어후의 결과치를 볼 수 있게 하는 부분이다. 편집기(editor), 컴파일러, 다운로더, 플로팅 도구(plotting tool)가 여기에 속한다.
- 둘째는 제어 프로그램 작성시에 사용되는 라이브러리로서, 제어기에서 실행되는 부분이다. 입출력장치 구동기(device driver), 시간 계획기(scheduler), 수학함수 루틴등이 있다.
- 세번째 종류의 프로그램은 SUN과 제어기에서 동시에 실행되는 것으로, 메세지 전달(message passing) 루틴, 가상단말기(virtual terminal), 화일지원기(file server), 오류수정기(debugger) 등이 여기에 속한다.

새로운 하드웨어에 맞도록 수정한 내용으로는, 호스트가 변경되었으므로 실시간 제어기와 호스트의 메세지 전달을 위해 호스트의 커널을 수정하였고, 엔코더 카운터를 사용하기 위한 장치구동기(device driver)를 작성하였다. M68040을 위하여 메세지 전달 루틴 및 인터럽트 처리 루틴, 벡터 테이블, 68881과 호환되지 않는 함수 sin( ), cos( ), sqrt( ), exp( ), log( )를 작성하였다. 또한, Condor에서는 Unix C compiler를 사용하나, 속도 및 실행코드의 크기를 최적화 해 주는 GNU C compiler를 사용하도록 개발 환경을 구성하였

다.

다음은 기능별로 소프트웨어를 설명한다.

2.2.1 다운로더(downloader)

다운로딩은 SUN의 컴파일러를 통해 컴파일된 실행화일을 제어기의 메모리에 복사함으로써 이루어진다. SUN의 하드디스크에 있는 것을 버스간 연결기를 통해서 제어기의 메모리에 직접 쓰기 때문에, 다운로딩 시간이 하드디스크에서 램에 복사하는 시간 밖에 소요되지 않는다.

2.2.2 플로팅 도구(plotting tool)

플로팅 도구는 제어시에 변하는 변수들의 값을 시각적으로 볼 수 있게 해 주는 프로그램이다. 이것을 사용하기 위해서는 제어 프로그램을 작성할 때, 원하는 변수의 값을 단일기판 컴퓨터의 메모리나 혹은 메모리기판에 저장하도록 작성한다. 제어를 수행한 후에, X윈도우 환경안에서 작동하는 플로팅 프로그램을 이용하여 실험결과치를 화면에서 그래프의 형태로 볼 수 있다.

2.2.3 입출력장치 구동기(device driver)

입출력을 위한 장치가 추가되면, 제어 프로그램에서는 그것을 사용하기 위한 구동기(driver)가 필요하다. 현재 Condor에는 D/A 변환기, A/D 변환기, 병렬입출력 장치, 엔코더 카운터를 위한 구동기가 작성되어 있다.

Condor에서 제공하는 구동기는 각 입출력 장치를 화일처럼 취급할 수 있게 작성되어 있다. 따라서, 사용자는 화일의 입출력과 같은 방법으로 이 장치들과의 입출력을 할 수 있다. 장치의 초기화, 제어 레지스터(control register)의 구조, 자료 레지스터(data register)의 주소등 하드웨어의 자세한 사항을 모르고도 사용이 가능하다.

2.2.4 시간 계획기(scheduler)

시간 계획기는 제어루프가 지정된 시간간격으로 시작되도록 해 준다. 예컨대, 하나의 처리기에 제어 루프(loop) A와 배경 루프 B가 실행되고 있을 때, A는 지정된 시간 간격대로 일정한 시간마다 한번씩 실행되고, B는 A가 실행되지 않을 때 실행되면서 사용자와의 통신과 제어기의 현재상태 출력등을 담당한다(Fig. 3).

2.2.5 메세지 전달 루틴(message passing routine)

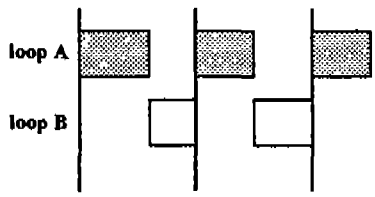

Fig. 3 Scheduling
loop A : Control loop
loop B : Background loop

다중처리기를 갖는 제어기는 복수의 처리기가 동시에 일을 하면서 서로 통신을 하는 것이 매우 중요하므로, 메세지 전달 루틴이 이를 뒷받침하기 위해 작성되었다. 이것은 공유메모리와 인터럽트를 사용하여 구현되었는데 Fig. 4에서 그 구조를 설명한다. Fig. 4는 처리기 A가 처리기 B에게 메세지를 전달하는 과정을 보여준다. 우선 A는 공유메모리에 전달할 메세지를 쓴다. 그후 메세지의 번호, 수신할 처리기의 번호(여기에서는 B) 그리고 메세지가 있는 위치등의 정보와 함께 인터럽트 신호를 버스에 보낸다. B에서는 이 인터럽트에 의해 메세지 처리기(handler)가 시작되어 A가 보내준 메세지를 읽고, A가 원한다면 응답(reply) 신호를 보내준다.

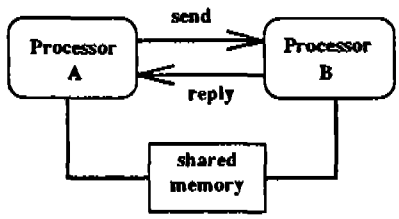

Fig. 4 Message passing

2.2.6 가상 단말기(virtual terminal)

가상 단말기는 SUN의 윈도우 하나가 각 처리기의 단말기처럼 동작할 수 있게 해 주는 프로그램이다(Fig. 5). 가상 단말기를 통해 사용자는 각 처리기의 상태를 감시할 수 있고, 처리기와의 입출력을 행할 수 있다.

2.2.7 오류 수정기(debugger)

제어 프로그램의 실행시 오류(run time error)의 수정을 돕기 위해 오류 수정기가 제공된다. 이것은 Unix

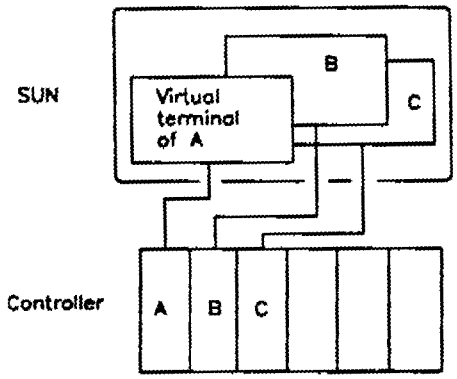

Fig. 5 Virtual terminal

와 X윈도우에서 제공하는 기호 오류 수정기(symbolic debugger)와 같은 기능을 갖는다. 사용자는 SUN에서 실행되고 있는 오류 수정기를 사용하여 제어기에서 실행되는 프로그램을 단계적으로 실행시키거나 중단점(break point)의 설정으로 프로그램의 흐름을 관찰할 수 있고, 변수들의 값 및 스택(stack)의 내용을 볼 수 있다.

2.2.8 화일 지원기(file server)

제어를 하다 보면 실험결과를 하드디스크에 저장하거나 하드디스크로부터 자료를 읽어 올 필요가 있을 것이다. 화일 지원기가 이를 위해 준비되어 있어서, 제어기가 SUN의 방대한 하드디스크를 사용할 수 있게 해준다. 제어기의 화일 개방명령은 메세지 전달의 방법으로 SUN에 전달되고, SUN의 화일지원기는 표준 Unix 화일 조작을 행한다. 따라서, SUN에서 실행되는 프로그램을 개발할 때 사용하던 화일의 입출력방식을 실시간 제어프로그램을 개발할 때에도 그대로 사용할 수 있다.

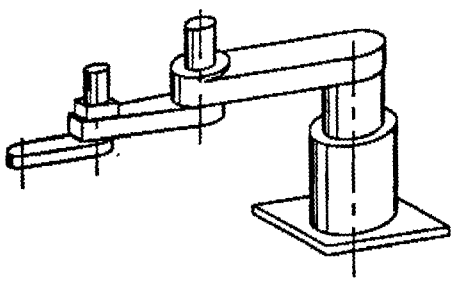

Fig. 6 Redundant robot

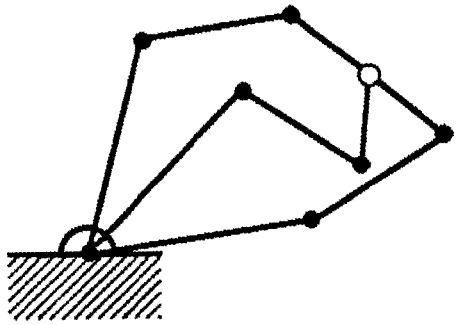

Fig. 7 3-DOF Planar robot

3. 여유자유도 로봇

3.1 여유자유도 로봇의 외형

본 연구에서 제작한 여유자유도 로봇의 외형은 Fig. 6과 같다. 보통의 2자유도 스카라(SCARA)형 로봇의 두번째 팔에 팔 하나를 추가하여 평면 3자유도 로봇으로 만들었다. 이 로봇은 여유자유도 로봇으로서 말단의 위치가 주어질 때 말단의 위치를 만족시키면서도 무한히 많은 팔의 모양을 갖을 수 있게 된다(Fig. 7).

각 link의 길이 및 질량을 포함한 기계적인 사양은 표2와 같다.

Table 2. Mechanical Demension

	Link 1	Link 2	Link 3
length (mm)	350	200	150
mass (kg)	11.2	6.8	2
center of mass from joint (mm)	285	174	75
moment of inertia (kg m^2)	1.03	0.224	0.015

3.2 여유자유도 로봇의 역기구학

여유자유도 로봇의 운동은 끝점이 원하는 경로로 움직이도록 하기 위한 작업운동(operational motion)과 끝점의 위치는 변화시키지 않으면서 움직이는 영운동(null motion)으로 나뉜다[5, 6]. 이것을 운동분해법(resolved motion rated control)의 식으로 표현하면 다음과 같다[5].

$$\dot{\theta} = J^{+}\dot{X} + \alpha(I - J^{+}J)\nabla H \qquad (1)$$

여기에서, 각 변수의 차원은 로봇의 자유도를 n, 작업의 차원을 m이라 할 때, $\theta \in R^m$, $X \in R^m$, $J \in R^{m \times n}$ 이다. 본 논문의 경우 n은 3이고, 말단의 위치(x, y)만 기술하므로 m은 2이다. (1)식의 의미를 상술하면, $J^+\dot{x}$는 θ의 작업운동을 일으키는 속도성분이고, $\alpha(I-J^+J)\nabla H$는 영 운동을 일으키는 속도성분이다. 이때, $J^+ = J^T(JJ^T)^{-1}$는 자코비안 행렬의 의사 역행렬이고, H는 영 운동에 의하여 증가시키거나 감소시키고자 하는 성능지수 이다. H로 사용되는 성능지수의 예로는 특이점회피, 장애물회피 그리고 관절한계 회피등을 위한 지수들이 있다. ∇H는 H의 θ에 대한 그라디언트(gradient)를 나타낸다. 즉,

$$\nabla H = \begin{pmatrix} \frac{\partial H}{\partial \theta_1} \\ \vdots \\ \frac{\partial H}{\partial \theta_n} \end{pmatrix} \tag{2}$$

이다. α는 영 운동에 가중치를 주는 상수로서, 양의 값이면 H가 증가하는 방향으로 영 운동이 일어나고 음의 값이면 H가 감소하는 방향으로 영 운동이 일어난다.

4. 실 험

4.1 실험내용

실험내용은 로봇의 끝점이 주어진 경로를 따라가게 하고, 동시에 로봇의 조작 성능지수(manipulability)[7]가 증가하는 방향으로 영 운동이 일어나도록 하는 것이다.

4.1.1 끝점의 경로

끝점의 경로는 직선으로 주는데, x좌표값은 x=350mm를 유지하면서 y좌표값은 350mm에서 출발하여 7초 동안에 -350mm가 되도록 한다. y는 시간에 대한 5차 정식으로 주어 속도, 가속도가 모두 연속이 되도록 계획한다.

$$\begin{aligned} y(0) &= 350mm \\ y(7) &= -350mm \\ \dot{y}(0) = \dot{y}(7) &= 0mm/sec \\ \ddot{y}(0) = \ddot{y}(7) &= 0mm/sec \end{aligned}$$

이므로, 구하는 정식과 그 미분치인 속도는 다음과 같이 결정된다.

$$y(t) = 350 - 20.4t^3 + 4.37t^4 - 0.250t^5 \tag{3}$$
$$\dot{y}(t) = -61.2t^2 + 17.5t^3 - 1.25t^4 \tag{4}$$

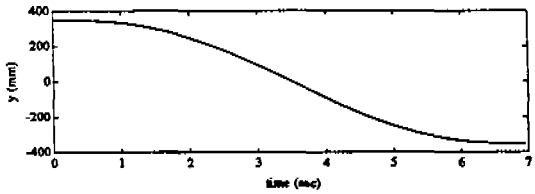

Fig. 8 Desired trajectory of y

4.1.2 역 기구학

식(1)의 H로서 로봇의 조작 성능지수[7]를 사용하였다. 조작 성능지수는 특이점 회피를 위한 성능지수중의 하나인데, 기존의 연구에서 많이 다루어 온 지수이므로 본 실험에서 도입하였다. 성능지수는 로봇의 자코비안을 J라 할 때, $\det(JJ^T)$로 정의된다. 계산의 편의를 위해 H를 첫번째 링크의 길이 l_1의 제곱으로 나누면, H는

$$H = det\left(\frac{JJ^T}{l_1^2}\right) \tag{5}$$

이다. 이때 J는 다음과 같다.

$$J = \begin{pmatrix} l_3c_{123} + l_2c_{12} + l_1c_1 & l_3c_{123} + l_2c_{12} \\ -l_3s_{123} - l_2s_{12} - l_1s_1 & -l_3s_{123} - l_2s_{12} \end{pmatrix}$$
$$\begin{pmatrix} l_3c_{123} \\ -l_3s_{123} \end{pmatrix} \tag{6}$$

여기에서, l_i는 i번째 링크의 길이로서, 표 2에 명시되어 있고,

$$\begin{aligned} u &= \frac{l_2}{l_1} \\ v &= \frac{l_3}{l_1} \\ s_{ij\cdots} &= sin(\theta_i + \theta_j + \cdots) \\ c_{ij\cdots} &= cos(\theta_i + \theta_j + \cdots) \end{aligned}$$

이다. 따라서,

$$\nabla H = \begin{pmatrix} 0 \\ 2(c_{23}s_3u + s_{2233})v^2 + 2s_{223}uv + s_{22}u^2 \\ 2(s_{33}u^2 + s_{233}u + s_{2233})v^2 + 2c_{23}s_2uv \end{pmatrix} \quad (7)$$

가 된다.

이 ∇H를 (1)식에 대입하면 θ를 구할 수 있다. 이러한 방법으로 θ를 구하면, θ가 수치적분되면서 위치의 오차가 누적되는 단점은 있지만, 여유자유도 로보트의 역기구학중 가장 보편적으로 사용되는 방법이므로 이 방법을 활용하였다.

4.1.3 관절수준의 제어

θ, $\dot{\theta}$가 계산되면, 관절수준의 제어는 경로 추종성능이 우수한 시간지연제어(time delay control) 기법을 사용한다[8].

여기에서 L은 추출시간(sampling time)을 나타내고 $\overline{M}$는 팔의 관성 모멘트의 추정치를 나타낸다.

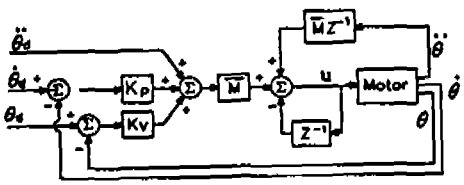

Fig. 9 Block diagram of TDC

4.1.4 처리기의 업무 분담

제어를 실행할 때는 제어기의 처리기 3개를 사용한다. 각 처리기의 업무는 Fig. 10과 같다. 처리기 0은 식(4)에 따라 끝점의 직교좌표계상의 경로를 생성한다. 처리기 1은 운동분해법을 사용하여 직교좌표상의 경로로부터 관절의 속도와 위치를 계산해 낸다. 처리기 2는 계산된 θ, $\dot{\theta}$를 로봇이 추종하도록 시간지연 제어기법에 따라 관절수준의 제어를 행한다.

각 처리기 사이의 통신은 공유 메모리를 통해 이루어진다. 즉, 변수들의 기억장소상의 위치(address)만 서로 알려주면, 각각의 처리기는 그 위치를 읽고 쓰는 것만으로 서로간에 변수의 내용을 주고 받을 수 있게 된다.

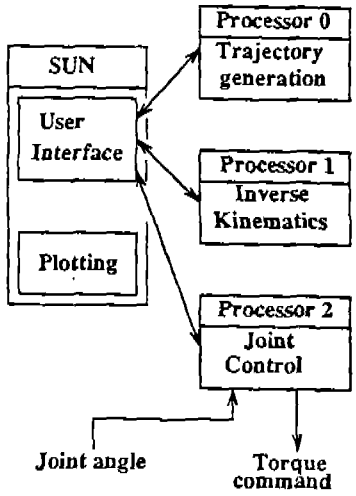

Fig. 10 Task of each processor

4.2 실험결과

앞에서 설명한 바와 같이 각 처리기가 업무를 분담하였을 때, 각각의 업무를 한번 실행하는 시간은 표 3과 같았다.

Table 3. Execution time

Task	Execution time (msec)	
	68020	68040
Trajectory generation	0.193	0.037
Inverse Kinematics	6.40	2.44
Joint Control	1.14	0.286

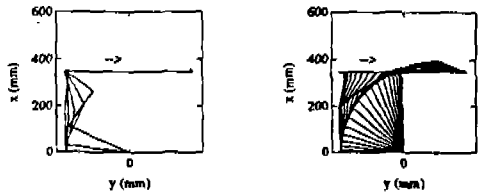

Fig. 11 Change of configuration

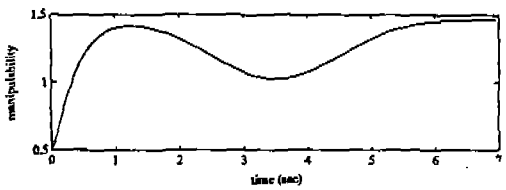

Fig. 12 Change of manipulability

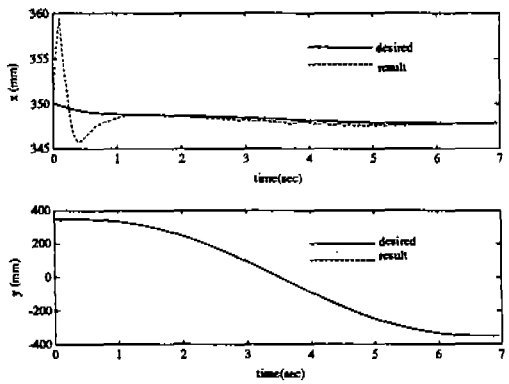

Fig. 13 Position of endpoint

실험한 결과는 Fig. 11, 12, 13에 나타나 있다. 시간에 따라서 끝점은 원하는 경로를 따라갔고, 동시에 영운동에 의해서 조작 성능지수가 초기치 보다 커졌다. Fig. 12에서 중간에 조작 성능지수가 작아지는 이유는 작업운동에 의하여 조작 성능지수의 값이 변하기 때문이다. Fig. 13에서는 x, y의 오차가 있음을 볼 수 있으나, 제어 알고리즘과 제어시의 오차등에 관한 논의는 본 논문의 주제에서 벗어나는 것으로 사료되므로 그에 대한 논의는 생략한다.

5. 결 론

지금까지 다중처리기를 이용한 고성능 범용제어기와 여유자유도 로봇의 개발에 대해서 설명하였고, 이를 이용하여 수행한 실험결과를 보였다. 본 연구를 통하여 다음 세가지의 의의를 찾을 수 있다. 첫째, 고성능 범용제어기에 대해서 그 하드웨어 및 소프트웨어를 수정하고 확장함으로써 성능의 개선을 이루었다. 둘째, 상용의 실시간 운영체제와는 달리 자체적으로 제어기 소프트웨어의 원시코드를 보유하게 됨으로써, 필요에 따라 발전, 확장시킬 수 있는 토대를 마련하였다. 셋째, 이미 개발된 이론중에서 실험이 어려웠던 기존의 여유자유도 로봇 제어 알고리즘의 성능을 실제 실험을 통해 확인해 볼 수 있었다.

이 시스템이 갖는 범용성, 유연성 및 계산능력을 적절히 활용하면 다음과 같은 분야에 유용하게 적용될 수 있을 것이다. 첫째, 여러가지 로봇의 제어이론을 손쉽게 실험하고 평가할 수 있으므로, 다양한 형태의 로봇에 대해서 실용적인 제어이론을 개발하는데 기여할 것으로 기대할 수 있을 것이다. 둘째, 로봇뿐만 아니라 공작기계, 유압기계 등을 위해서도 신속하고 편리하게 제어 시스템을 구축하는 것이 가능할 것이다.

참고문헌

1. Sundar Narasimhan, David M. Siegel and John M. Hollerbach, "CONDOR : An Architecture for Controlling the Utah-MIT Dexterous Hand", IEEE Trans. Robotics and automation, Vol. 5, No. 5, pp. 616~624, 1989.
2. David B. Stewart, Donald E. Schmitz and Pradeep K. Khosla, "Implementing Real-Time Robotic Systems Using CHIMERAII", Proc. IEEE Conf. Robotics and Automation, pp. 598~603, 1990.
3. 이덕만, 오종한, 이진수, "실시간 운영체제를 이용한 범용로봇 제어 언어의 개발", 한국 자동제어 학술회의 논문집, Vol. 1, pp. 18~23, 1991.
4. 구영재, 이준서, 이인범, 장근수, "PC를 이용한 자동제어 시스템 개발, 한국 자동제어 학술회의 논문집, Vol. 1, pp. 322~326, 1991.
5. C. Klein and C. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators", IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 2, pp. 245~250, 1983.
6. Yoshihiko Nakamura, "Advanced Robotics : Redundancy and Optimization", Addison-Wesley Publishing Company, 1991.
7. T. Yoshikawa, "Analysis and control of robot manipulators with redundancy", in Proc. Robotics Research : 1st Int. Symp., M. Brady and R. Paul, Eds. Cambrige, MA : MIR Press, pp. 735~748, 1984.
8. K. Youcef-Toumi and I. Osamu, "A Time Delay Controller for Systems with Unknown Dynamics", ASME J. Dynamic Sys. Meas. Contr., Vol. 112, pp. 133~183, 1990.