PC 를 이용한 PUMA 로보트의 제어시스템 구성

김 대원 °이 원식 경 계현 이 상무 고 명삼 이 범희
서울 대학교 제어계측공학과 로보틱스 및 지능 시스템 연구실

A Design of the PUMA Robot Control System Using a PC

Dae Won Kim, °Won Sik Lee, Kye Hyun Kyung, Sang Moo Lee,
Myoung Sam Ko, Bum Hee Lee
Robotics and Intelligent Systems Lab.
Dept. of Control & Instrumentation Eng., Seoul Nat'l Univ.

ABSTRACT

In this paper, a control system of the PUMA 560 robot manipulator using a PC (Personal Computer) is presented. The hardware of the designed control system is composed of IBM-PC/AT, interface board, selection board, interrupt generating circuit, and the servo control unit of the PUMA controller. A robot control library is developed using assembly and C language, and is composed of several low-level functions and arm interface routines, world model routines, arm kinematics routines, and motion command service routines. Using the designed library, joint interpolated motion and Cartesian interpolated motion of the PUMA robot manipulator are realized. In the future, our system is expected to be a very helpful basis and a useful tool for developing various control algorithms of robot manipulator using sensory information.

1. 서론

오늘날 산업의 규모가 확대되고, 복잡해 짐에 따라, 다양한 생산의 요구와 더불어 많은 분야에서 자동화에 대한 관심이 증대되어 가고 있다. 이와함께 산업 자동화 및 성력화의 한 요소로서 사용되는 로보트 매니퓰레이터에 대한 관심은 더욱 높아지고 있는데, 이는 자동화에 있어서 로보트 매니퓰레이터가 차지하는 비중이 얼마나 큰가를 의미한다고 볼 수 있다. 그러나 로보트 매니퓰레이터는 인간이 설계한 복합적 기능을 소유하는 도구의 하나이므로 인간의 지능을 로보트 매니퓰레이터에 부여하고자 하는 노력이 계속적으로 이루어 지고있다. 한편, 로보트 매니퓰레이터에 있어서 감각기관이라 할 수 있는 센서류와, 공장 자동화의 작업 단위에 포함될 수 있는 컨베이어 시스템 등을 로보트 매니퓰레이터와 함께 총괄 제어하고자 하는 시도 또한 다각적으로 진행되고 있다.

현재 상업적으로 제작된 산업용 로보트의 경우, 대개 확장성이 없으며, 내부 소프트웨어에 관한 내용이 공개되고 있지 않은 실정을 감안하여, 본 연구에서는 그동안의 자체 연구 결과 [1,2,5,15,16,17]를 바탕으로 하여 널리 사용되고 있는 PC 를 이용하여 다관절형 산업용 로보트인 PUMA 를 제어 하며, 동시에 외부 센서로부터의 정보를 수용하여 사용자 자신이 작성한 제어 알고리즘을 실현시켜 보고자 한다.

센서 정보를 이용한 로보트의 제어 알고리즘을 실현시켜 보기 위하여 IBM-PC/AT 를 사용하였고, 대상 로보트인 PUMA 560 매니퓰레이터의 써어보 제어부는 그대로 이용하였으며, 이들의 인터페이스를 위하여 인터페이스 보오드를 설계 , 제작 하였다. 또한 PUMA 로보트 매니퓰레이터의 동작을 위하여 초기화를 비롯한 기본적인 소프트웨어를 설계하였다. 센서 정보를 이용한 로보트 매니퓰레이터의 각종 제어 알고리즘 실행은 IBM-PC/AT를 통하여 이루어졌다.

본문에서는 2 장에서 설계된 전체 시스템의 구성을 알아 보고 , 3 장에서는 PC 에 의한 로보트 매니퓰레이터의 제어를 위한 PC 와 써어보 장치의 인터페이스에 대하여 서술하며, 4 장에서는 C-언어에 의한 제어 프로그래밍 시스템의 설계에 대하여, 마지막 장에서 결론 및 앞으로의 연구 방향에 대하여 서술한다.

2. 설계된 시스템의 구성

2.1 PUMA 로보트 제어 시스템의 구조

PUMA 로보트 매니퓰레이터의 써어보 제어구조는 그림 1 과 같다. LSI-11/02 와 써어보 제어부는 DRV-11

을 통해서 연결 되는데 LSI-11/02 는 28ms 마다 위치에 대한 정보를 내어주고 써어보 제어부의 6503 마이크로 프로세서는 이것을 32등분 하여 PUMA ARM 을 제어 하게 된다.

그런데 기존의 PUMA 로보트 매니퓰레이터를 제어해 주는 제어기는 확장성이 없고, 또한 센서 정보를 처리 하는 기능이 없기 때문에 센서정보를 이용하고, 다른 주변기기들을 제어하기 위해서는 전용 언어인 VAL을 포함하는 디지탈 제어부를 컴퓨터로 교체 해야 한다. 이를 위하여 소형 컴퓨터로는 IBM-PC/AT를 이용 하고, 인터페이스 보오드를 설계, 제작하여 써어보 제어부와의 연결을 꾀하였다.

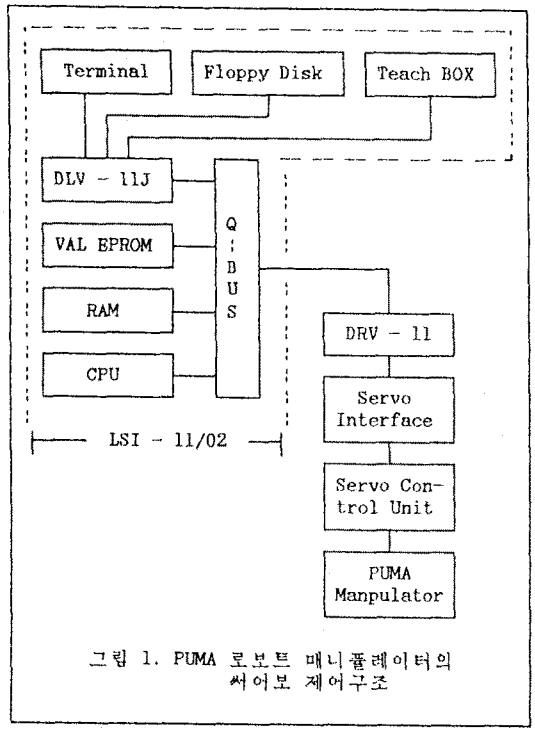

그림 1. PUMA 로보트 매니퓰레이터의 써어보 제어구조

2.2 설계된 시스템의 구성

설계된 제어 시스템은 IBM-PC/AT 본체와 IBM-PC/AT 와 PUMA 로보트 매니퓰레이터 제어기의 써어보 인터페이스 보오드 (servo interface board) 사이의 연결을 위한 인터페이스 보오드, 그리고 PC 본체와 PUMA 로보트의 전용언어인 VAL 을 포함하는 디지탈 제어부를 선택할 수 있게 해주는 선택 보오드 (selection board), 마지막으로 기존의 PUMA 제어기중 써어보 제어부로 구성된다. 그림 2 는 PUMA 로보트 매니퓰레이터와 IBM-PC/AT 의 인터페이스에 관한 블럭선도이다.

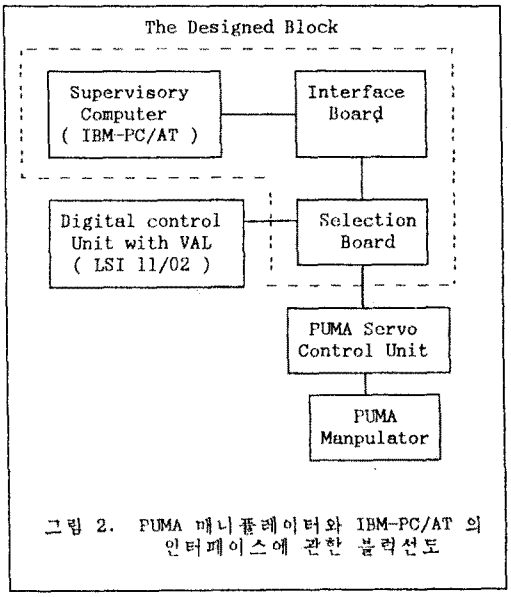

그림 2. PUMA 매니퓰레이터와 IBM-PC/AT 의 인터페이스에 관한 블럭선도

3. PC 와 PUMA 써어보 제어부와의 인터페이스

3.1 인터페이스 보오드의 설계

인터페이스 보오드는 기존의 PUMA 제어기에서 디지탈 제어부와 써어보 제어부를 연결하는 DRV-11의 역할을 대신하는데, PIO, Buffer, Decoder, Timer 등으로 구성되며, 이에대한 블럭선도는 그림 3 과 같다. PUMA 매니퓰레이터의 경우 써어보 인터페이스 보오드로 28 ms 주기마다 새로운 위치 명령을 주어야 한다. 이를 위해 PC 내부 CLOCK을 이용하려고 하였으나, 사용자가 사용할 수 있는 것은 10 ms 와 55 ms 단위여서 사용이 불가능하므로 LM-322를 이용하여 Timer 회로를 구성하여 이용하였다.

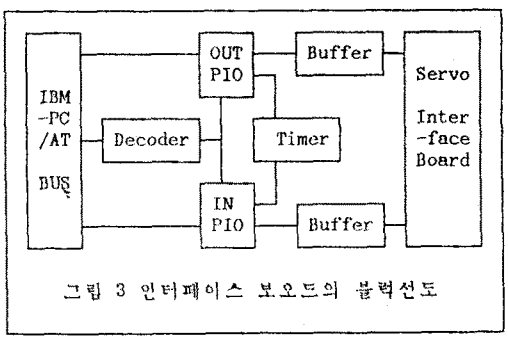

그림 3 인터페이스 보오드의 블럭선도

3.2 선택 보오드의 설계

기존의 PUMA 제어기 내에서 써어보 제어부 만을 이용하기 위해서는, LSI-11/02 보오드와 전용언어 VAL에 의하여 구성된 디지탈 제어부와의 통신 프로토콜과 VAL의 내용을 파악하고 있어야 한다. 그러나 초기화 루틴과 보정루틴 (calibration routine) 에 대한 해독이 불완전 하여 기존의 디지탈 제어부에서 써어보 인터페이스 보오드로 내보내는 초기화 및 보정 명령을 써어보 인터페이스 보오드로 부터 또다시 IBM-PC/AT 로 받아서, 참고자료의 내용과 함께 초기화 및 보정루틴을 수립하기 위해 선택 보오드를 설계하였다. 이에 대한 블럭선도는 그림 4 와 같다.

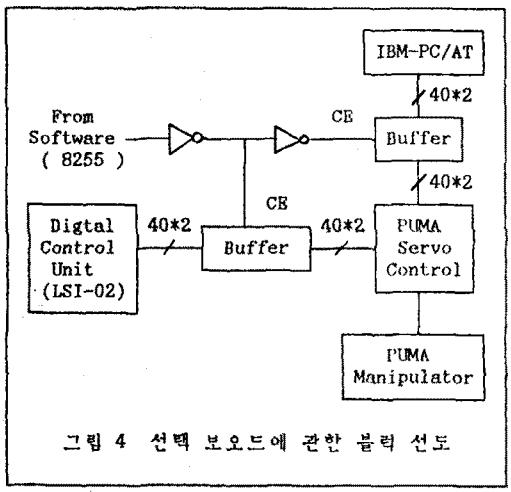

그림 4 선택 보오드에 관한 블럭 선도

3.3 인터럽트 처리를 위한 설계

IBM - PC/AT 를 이용하여 시스템 전체를 총괄 제어하기 위해서는 몇개의 주기적인 인터럽트가 필요 하게 되는데, 이는 PUMA 로보트 매니퓰레이터가 써어보 인터페이스 보오드로 28ms 주기 마다 새로운 위치 명령을 보내어 주어야하고, 힘센서를 이용할 경우에도 주기적으로 힘센서의 정보를 새로운 위치 명령에 반영해 주어야 한다. 그러나, 현재 IBM-PC/AT는 MS-DOS 상에서 사용 하기 때문에 동시에 2가지 이상의 작업을 실행 시킬수 있는 시분할 방식이 아니므로 외부에서 주기적으로 인터럽트를 발생시켜 줄 수 있는 회로를 만들어 주어야 한다. 설계된 인터럽트 발생회로에서는 정해진 일정 주기마다 인터럽트를 발생시켜서 새로운 위치 명령을 내주거나, 힘센서의 정보를 새로운 위치 명령에 반영하게 하였다. 이를 실현하기 위한 하드웨어의 구성은 아래 그림 5 와 같다.

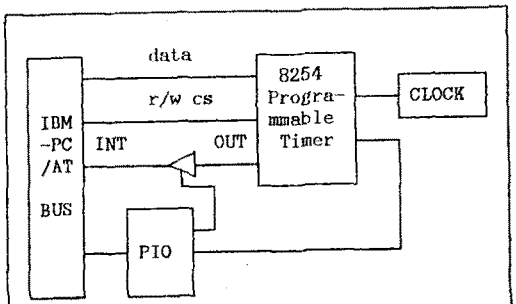

그림 5 인터럽트 발생을 위한 하드웨어의 구성

4. C-언어에 의한 제어 프로그래밍 시스템의 설계

4.1 제어 프로그래밍 시스템의 구성

기존의 PUMA 매니퓰레이터를 제어하기 위해서는 전용 언어인 VAL 에서의 통신 방식과, 초기화 및 여러가지 구조를 파악하고, 필요한 소프트 웨어를 설계하여야 한다 [1,2,6]. PUMA 를 제어하기 위한 제어 프로그래밍 시스템은 각 축에 위치 명령을 내리고, 현재의 위치를 확인하며, 각 축을 한계범위 내에서 동작하게 하고, 비상정지 등을 처리 하는 Low-Level Function 및 Arm Interface 부분, PUMA 로보트 매니퓰레이터의 작업환경을 Modelling 하는 World Model 부분, Cartesian 좌표와 각축 좌표 사이의 변환을 위해서 Kinematics 및 Inverse Kinematics 를 푸는 Arm Kinematics 부분, 그리고 관절 보간 운동과 직선 보간 운동을 실행하는 운동 명령어 처리 부분등으로 구분할 수 있다. 이의 구성을 블럭으로 표현하면 그림 6 과 같다.

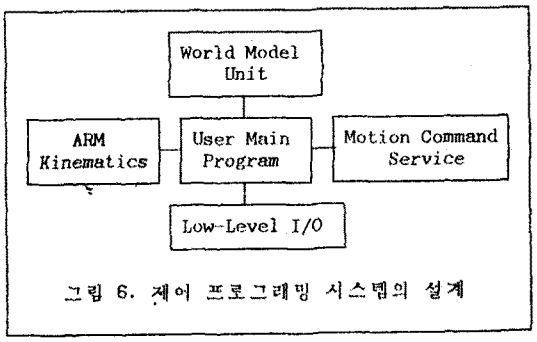

그림 6. 제어 프로그래밍 시스템의 설계

4.2 인터페이스 보오드를 위한 소프트웨어의 설계

앞서 설명한 인터페이스 보오드의 경우에는 INTEL 8255 를 사용하여 설계하였으므로 한번에 8 비트를 제어 할 수 있다. 따라서, 16 비트씩 제어되는 DRV-11의 경우와 달리 제어명령을 순차적으로 실행시켜야 한다.

인터페이스 보오드를 제어하기 위한 부프로그램들은 다음과 같다.

```
.SETUP()        : 인터페이스 보오드를 초기화 시킴
.READSINGLE()   : 써어보 보오드 내용을 읽어 들임
.WRITESINGLE()  : 써어보 보오드에 명령을 내림
.READ7()        : 써어보 보오드 I/O 의 입력을 직
                  접 읽어 들임
.WRITE7()       : 써어보 보오드 I/O 의 출력으로
                  직접 내보냄
.ARMON()        : ARM Power Enable
.ARMOFF()       : ARM Power Disable
.HANDOPEN()     : Hand Open
.HANDCLOSE()    : Hand Close
.planner()      : 경로계획을 수행하는 경로계획기
.main_int()     : main interrupt 를 처리함
```

4.3 World Modelling을 위한 소프트웨어의 설계[4, 7,13,14]

본 시스템의 World Model은 연결된 kinematic 고리에 의해 표현되는 기하적 상황의 집합으로 coordinate frame들의 상대적 위치를 나타낸다. 이때 각 상황은 homogeneous transform 식으로 표현되며, transform 들의 값은 프로그램상에서 특정 값으로 지정 될 수도 있고, 또한 작업 수행 중에 센서 정보나 계산 또는 다른 여타의 정보에 의해 개선될 수 있다. Homogeneous transform 은 한 좌표계에 대한 또다른 좌표계의 위치 (position) 와 방향 (orientation) 을 나타내는 4 X 4 의 행렬이다. 이같은 World Model 은 Cartesian 좌표계에서 로보트의 작업을 프로그램 하는 것을 가능하게 하며, 위치식 (position equation) 은 작업의 공간적 구조로 표현된다. 예를 들어 테이블 위에 놓여있는 핀을 잡아서 테이블 위의 조립대의 구멍에 넣는 작업은 식 (1) 과 (2) 의 위치식으로 나타낼 수 있다.

$$B\ T6\ E = T\ P\ PG \qquad (1)$$

$$B\ T6\ E = T\ AS\ H\ PG \qquad (2)$$

이 식들에서 각 transform은 다음과 같다.

B : 기준 좌표계에 대한 로보트 base의 위치

T6 : 로보트 base에 대한 로보트 마지막링크의 위치

E : 로보트의 마지막 링크에 대한 툴(tool)의 위치

T : 기준 좌표계에 대한 테이블의 위치

P : 테이블에 대한 핀의 위치

PG : 핀에 대한 잡는 위치(grasping position)

AS : 테이블에 대한 조립대의 위치

H : 조립대에 대한 구멍의 위치

여기서 직선 보간 운동을 의한 궤적은 위와 같은 위치식에 drive funtion 이라는 transform 을 삽입함에 의해 계산될 수 있다. 이 transform D(r) 은 로보트를 한 상태에서 다음 상태로 이동시키는 직선 병진과 축에 대한 회전을 나타낸다. 위의 예에서 로보트가 식(1)의 위치에서 식(2)의 위치로 이동할 때 로보트의 궤적은 (3)식과 같이 계산 된다.

$$B\ T6 = T\ (T^{-1}\ T\ P\ PG\ E^{-1}\ E)\ D(r)\ E^{-1}$$

$$= T\ P\ PG\ D(r)\ E^{-1},\quad 0 <= r < 1 \qquad (3)$$

이 때 (3)식에 r=0 와 r=1 을 대입하면 각각 (1)과 (2) 식의 위치가 된다.

World Modeling에 관한 부프로그램은 다음과 같다.

```
.makeposition() : 위치구조(position structureion)
                  를 형성
.joint_new()    : 축 구조(joint structure) 의 메
                  모리할당 및 초기화
.position_new() : 위치 구조의 메모리 할당 및 초
                  기화
.term_new()     : 연결 구조(linkterm structure)
                  의 메모리 할당 및 초기화
.trans_new()    : transform structure 의 메모리
                  할당 및 초기화
.oat_new()      : OAT 구조의 메모리 할당 및 초기
                  화
.trsf_tr()      : 전이 벡터와 한 벡터에 대한 회
                  전이 주어졌을 때 transform 을
                  생성
.trsf_t()       : 전이 벡터가 주어졌을 때 trans-
                  form 을 생성
.trsf_euler()   : 전이 벡터와 Euler 각으로 표시된
                  회전이 주엊겼을 때 transform 을
                  생성
.trsf_pao()     : p,a,o 벡터가 주어졌을 때 trans-
                  form 을 생성
.trsf_rpy()     : 전이 벡터와 rpy 각으로 표시된
                  회전이 주어졌을 때 transform
                  을 생성
.transl()       : 주어진 전이 벡터를 transform의
                  p 벡터에 할당
.rotation()     : transform 의 회전 부분을 벡터
                  k 주위로 theta만큼 회전시킴
.euler()        : transform 의 회전 부분을 Euler
                  각으로 회전 시킴
.rpy()          : transform 의 회전 부분을 rpy
                  각으로 회전시킴
.pao()          : transform 의 회전 부분을 a, o
                  벡터를 이용하여 완성시킴
.assign_tr()    : rhs transform 을 lhs transform
                  에 할당함
```

.take_rot()	: rhs transform 의 회전 부분을 lhs transform 의 회전 부분에 할당함
.tr_mult_inv()	: r = r * m-1 , m-1 : m의 역행열
.invert()	: i = m-1
.assignvector()	: 벡터의 사본(copy)을 만듬
.cross()	: 두 벡터의 cross product를 만듬
.unit()	: 벡터를 unit vector 로 만듬
.dot()	: 두 벡터의 dot product 를 만듬
.p_oat()	: transform 을 사용하여 PUMA arm 의 위치를 pOAT로 표시함
.drive()	: scale 요소 r 에따라 DRIVE 함수를 구함
.move()	: DRIVE 함수를 구하는데 필요한 x, y, z, theta, phi, psi 를 구하고 주어진 arm 의 속도 곡선에 따라 scale 요소 r 을 구함

4.4 운동 명령을 위한 소프트웨어의 설계

PUMA 의 운동명령은 관절 보간 운동 (Joint interpolated motion)과 직선 보간 운동(Cartesian interpolated motion) 으로 나눌수 있다.

1) 관절 보간 운동(Joint interpolated motion)

관절 보간 운동은 경로계획기에서 계획된 경로계획에 따라 시간축상의 위치에 따른 매 샘플링 시각에서의 각 축의 위치를 구하여, 각 축으로 내보내어, 각 축을 원하는 위치로 이동 시키는 것이다. 관절 보간 운동을 위하여 경로를 계획하는 경로계획기를 작성하였고, 디지탈 써어보로 내주어야 할 각 축의 값을 계산하고, 써어보로 부터 각 축의 위치와 상태를 확인하는 Main Interrupt Routine 을 작성 하였으며, 목표 위치를 읽어 들이는 루틴을 작성하여 실제로 관절 보간 운동을 PUMA 로보트 매니퓰레이터에 적용할 수 있었다.

관절 보간 운동을 위한 부프로그램은 다음과 같다.

.initial()	: PUMA 를 초기화 시키고 현재의 위치를 화면에 표시함
.stop()	: PUMA 를 Stop 모드에 두고 현재의 상태를 화면에 표시함
.read_position()	: 목표위치를 읽어들임
.set_position()	: 목표위치를 궤적계획기에 적합하게 변환함
.print_planner()	: 궤적계획기에서 계획된 경로계획을 화면에 표시함
.move_position()	: 목표위치로 움직임
.read_adc()	: A/D 변환기값을 읽어들임
.adc_degree()	: A/D 변환기값을 Degree 값으로 변환함
.degree_encoder()	: Degree 값을 엔코더 (Encoder) 값으로 변환함
.encoder_degree()	: 엔코더 값을 Degree 값으로 변환함
.inverse()	: Inverse Kinematics 의 해를 Geometric 정보를 이용해 구함
.forward()	: Kinematics의 해를 구함

2) 직선 보간 운동(Cartesian Interpolated Motion)

직선 보간 운동은 경로계획기에서 계산해낸 Drive Function D(r) 에 의해 목표 지점으로 향하는 직선상의 check point 들의 Cartesian 좌표를 구하고, 이로부터 inverse kinematics 해에 의해 각 축의 위치를 구하여 각 축을 원하는 위치로 이동 시키는 것이다. 직선 보간 운동을 위하여 IBM-PC/AT 의 키보오드를 이용해 교시도구를 개발 하였고, 현재의 위치를 [JT1, ... JT6] 로 표현하게 하였으며, 또한 [ㄱㄴㄷㄹ] 로도 표현하고, [X Y Z O A T]로 표현할 수 있게 하였다. 또한 Drive Function D(r) 에서의 변수 r (t)를 생성하는 루틴을 작성하였고, 각 시간축 상에서 각 축이 동작 한계범위를 초과 하는가를 확인하여 동작 범위를 초과하는 경우는 동작 시간을 늘여 r(t) 를 새로이 생성하게 하였다. 교시 도구를 이용하여 목표 위치를 읽어 들여 실제로 직선 보간 운동을 PUMA 로보트 매니퓰레이터로 실현 할 수 있었다.

직선 보간 운동을 위한 부프로그램은 아래와 같다.

.head()	: 교시상자 (Teach Box) 의 메뉴를 표시함
.teach()	: 교시상자로 동작하게함
.car_move()	: 직선 보간 운동을 위한 부프로그램
.move_pos()	: 관절 보간 운동을 위한 부프로그램
.drive()	: 직선 보간 운동을 위한 Drive Function 생성
.gener()	: Drive Function D(r) 을 발생시키기 위한 r 값을 생성함
.check_a()	: 각축의 위치 명령이 정해진 범위 내에서 동작하는가를 확인함
.check_b()	: 샘플링 마다 각축의 위치명령 증가분이 한계를 벗어나는가를 확인함
.car_int()	: 직선 보간 운동에 대한 interrupt service 를 행함

5. 결론 및 앞으로의 연구 방향

산업용 로보트 PUMA 에 보다 다양하고, 확장성 있는 기능을 부여 하기 위하여 PC 와 PUMA 로보트 매니퓰레이터와의 인터페이스를 실현 하였고, 기존의 PUMA 로보트 매니퓰레이터가 갖고 있던 최소한의 기능을 발휘하도록 C-언어를 이용하여 Library 형식의 프로그램을 작성하였다. 사용자는 주 프로그램상에서 Library 형식으로 짜여진 각종 부프로그램을 이용하여 자신의 제어 알고리즘을 점검할 수 있다.

PUMA 로보트 매니퓰레이터와 외부 센서류, 컨베이어 시스템등을 총괄 제어해 주는 중앙 컴퓨터인 IBM-

PC/AT 와 PUMA 로보트 각 관절을 구동하기 위한 써어보 제어부와의 연결을 위하여 인터 페이스 보오드와 선택 보오드를 설계, 제작하였고, 주기적인 위치 명령과 센서 정보처리를 위하여 인터럽트 발생회로를 설계, 제작하였다. 현재 C-언어및 어셈블리어로 작성된 프로그램 분량은 110KB, 약 5000 줄에 달하며, 그 기능으로는 관절 보간 운동과 직선 보간 운동, 각종 프로그램 제어 기능과 교시 기능등이 있다.

앞으로 PUMA 로보트의 손목에 힘/토오크 센서 JR3 를 부착하여 이미 설계된 Library 와 현재 연구 진행중인 힘 제어 알고리즘을 통하여 컴프라이언트 동작도 실현할 수 있으리라 사료되며, 로보트 매니퓰레이터를 포함하는 FMS (Flexible Manufacturing Systems)의 한 작업 단위 전체를 관장하여 제어하는 중앙 컴퓨터로서의 기능도 확인할 수 있을 것으로 생각된다.

참고 문헌

[1] 김 경환, "VAL - 컴파일러를 이용한 로보트 프로그래밍 시스템의 설계", 서울 대학교 공학 석사 학위 논문, 1986.

[2] 김 점근, "PDP - 11/44 컴퓨터와 PUMA 서보와의 인터 페이싱" 서울 대학교 제어계측공학과 실험 프로젝트 보고서, 1984.

[3] 최 명환, "A study of two dimensional object Contour Tracking with a sensor equipped manipulator", 서울 대학교 공학 석사학위 논문, 1987.

[4] Vincent Hayward, R.P. Paul,"Robot Manipulator Control under Unix RCCL : A Robot Control "C" Library",The International Journal of Robotics Research, pp.94 - 110, 1986.

[5] Sohari S., "Multiple Manipulators and Robotic Workcell coordination ", Proceedings IEEE International conference on Robotics and Automation, pp.1236 - 1242, 1986.

[6] Unimation, "Controlling the PUMA Series 500 Robot Arm Without using VAL"

[7] R.P. Paul, Robot Manipulators : Mathematics, Programming, and Control, The MIT Press, 1982.

[8] C. S. George Lee, "Robot Arm Kinematics, Dynamics, and Control", IEEE Computer, VOL 15, No 12, 1982.

[9] B. Shimano, " VAL : A Versatile Robot Programming and Control System", Proceeding of COMPAC 79, 1979.

[10] Unimation Inc., User's Guide to VAL : A Robot Programming and Control System, Unimation Inc., Danbury, CT, Ver. 12, June 1980.

[11] B. W. Kernighan, D. M. Ritchie, The C Programming Language, Prentice-Hall, 1978.

[12] A. Kelley ans I. Pohl, A Book on C, The Benjamin/Cummings Publishing Company Inc., 1984.

[13] Vincent Hayward, Introduction to RCCL : A Robot Control "C" Library, Technical Reports TR-EE 83-43, Purdue Univ., Oct. 1983.

[14] Vincent Hayward , Robot Real Time Control User's Manual, Thecnical Reports TR-EE 83-42, Purdue Univ., Oct 1983.

[15] 김 대원, "디버깅 특성을 고려한 로보트 매니퓰레이터의 언어에 관한 연구", 서울 대학교 공학 석사 학위 논문, 1985.

[16] 송 진원, "로보트 매니퓰레이터 제어를 위한 광학식 근접 센서의 설계와 응용", 서울 대학교 공학 석사 학위 논문, 1986.

[17] 경 계현, "Visual Servoing Set Up", 서울 대학교 제어계측공학과 실험프로젝트 보고서,1987.

[18] William H. Murray III and Chris H. Pappas, 80386/80286 Assembly Language Programming, Osborne McGraw-Hill Berkeley, Carifornia, 1986.