'93 한국자동제어학술회의논문집 (1993. 10. 20~22)

홍익 직접 구동 로보트를 위한 프로그래밍 언어 및 관리 통합 환경의 개발

° 김성훈, 이종수, 최경삼
홍익대학교 공과대학 전기공학과

A Study on the Development of Programming Language and Management Integrated Environment

° Sung-Hoon Kim, Jong-Soo Lee, and Kyung-Sam Choi
* Department of Electrical & Control Engineering, Hong-Ik University

Abstract

In this paper, we develop the basic robot commands on the level of VAL robot language and the integrated environment software of the robot management system to give users an easy way of programming and running the robot . The developed software is designed to support Korean language and to be run by the pop-up menus for programming commands and inputs. Geometrical and dynamical features can be viewed on a computer monitor by graphics and the taught works can be interfaced with a computer and controllers.

1. 서 론

1960년대 오프 라인 프로그래밍 언어의 개발 이후 로보트 프로그래밍 기술은 고수준(high level) 프로그래밍 언어로 변화했다. 로보트 응용 범위의 확대와 자동화 수준의 향상 그리고 로보트 주변 환경의 컴퓨터화로 오프 라인(off-line) 로보트 프로그래밍 언어가 널리 보급될 것으로 예상된다.

홍익 대학교에서는 직접 구동 로보트를 설계 제작하였으며, 구동부 제어기들의 연계적 운용과 사용자에게 편리성을 제공하는 인터페이스의 개발이 요구되었다. 따라서 본 연구를 통하여 로보트가 갖추어야 할 기본적인 명령어를 모두 포함하는 VAL 언어 수준의 로보트 언어를 개발하고, 이를 사용자가 쉽게 프로그램하고 실행시키는데 필요한 모든 편의를 제공할 수 있도록 로보트 운영 시스템의 성격을 갖는 통합 소프트웨어를 개발하였다. 개발한 소프트웨어는 우리 나라 사용자들의 편의를 위하여 한글 출력이 되도록 하였으며, 프로그램 명령어들과 입력 방식을 메뉴 방식에 의해 구동되도록 하여 사용자의 편의성이 최대로 보장되도록 하였다. 또한 로보트의 동적인 특성과 기하학적인 특성을 컴퓨터 모니터상에서 그래픽으로 관찰할 수 있다.

2. 전체 시스템 블럭도

본 운영 시스템은 사용자의 편의를 강조한 대화형의 통합환경으로서 프로그램을 작성하기 위한 편집 모드, 동작에 필요한 경로 방정식의 계수를 찾아내어 화일로 보관하는 컴파일 모드, 프로그램의 오류를 쉽게 발견할 수 있는 시뮬레이션 모드, 교시에 의해 위치와 명령어를 편집하는 교시 모드, 실제의 하드웨어와의 접속에 의해 로보트를 구동시키는 실행 모드로 구성되어 있다. 시스템 블럭도는 그림 1과 같다.

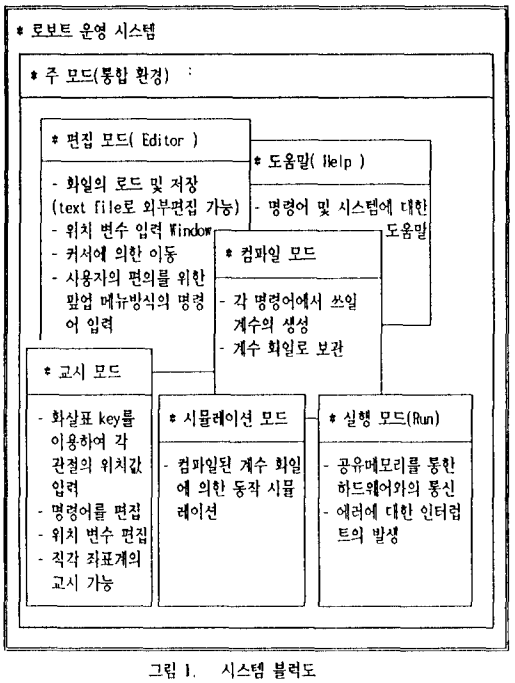

그림 1. 시스템 블럭도

3. 개발한 로보트 프로그래밍 언어

(1) Moveto <위치정보> [시간(sec)]
로보트를 현재 위치에서 지정된 위치로 이동시

킨다. 경로 계획은 관절 보간에 의한다. 위치 정보는 변수 선언부에서 참조할 수도 있고, 직접 데이타(Degree/Radian)를 입력 할 수도 있다.
(예) Moveto < P1 > [3.1] :
로보트를 변수 P1에 정의된 위치로 3.1초의 경과 시간을 가지고 이동시킨다.
Moveto <10, 10> [1.5] :
로보트를 관절값 <10, 10>의 위치로 1.5초내에 이동한다.

(2) Moverel <상대 위치정보> [시간(sec)]
로보트를 현재 위치에서 지정된 위치로 이동시킨다. 경로 계획은 관절 보간에 의한다. 상대 위치 정보는 변수 선언부에서 참조할 수도 있고 직접 데이타(Degree/Radian)를 입력 할 수도 있으며, 입력된 값에 의한 상대 위치로 이동한다.
(예) Moverel < P1 > [3.1] :
로보트를 현재 위치값에서 상대적으로 변수 P1의 위치로 3.1초의 경과 시간을 가지고 이동시킨다.
Moverel <10, 10> [1.5] :
로보트를 현재 위치값에서 관절값 10,10을 더한 위치로 1.5초내에 이동한다.

(3) Movesingle <관절번호> <위치정보> [시간(sec)]
한 개의 관절만을 이동시키는 명령어로서 선택된 관절을 입력된 관절값의 위치로 주어진 시간내에 이동시킨다. 위치 정보는 변수 선언부에서 참조할 수도 있고, 직접 데이타(Degree/Radian)를 입력 할 수도 있다.
(예) Movesingle <1> <20.4> [3.1] :
메뉴플레이터의 관절 1번을 20.4도의 각도로 3.1초 동안에 이동시킨다.
Movesingle <2> <P2> [2.5] :
메뉴플레이터의 관절2번을 변수 P2에 정의된 2번관절의 위치로 2.5초 동안에 이동시킨다.

(4) Movevia <위치정보1>, <위치정보2>, ·, <위치정보N> [시간(sec)]
여러개의 경유점을 지날 때 사용한다. 그 경유점들을 지나는 경로 계획 함수를 발생시키며, 지정된 시간을 알맞게 분할하여 함수가 생성되므로 일부러 사용자가 일일이 각 구간마다의 시간을 정의하여 줄 필요가 없다. 각 경유점들의 위치 정보는 직접 값이 대입될 수도 있고, 미리 정의된 변수에 맞추어서 선언될 수도 있다.
(예) Movevia <5> <10,10><P1><P0><P3><0,0> [6.5] :
로보트의 관절은 5개의 경유점을 지나게 되고 지정된 시간을 알맞게 나누어서 각 구간을 이동하게 된다.

(5) Relax [시간(sec)]
프로그램의 동작을 지정된 시간 동안 지연시킨다.
(예) Relax 10

(6) Goto <행번호>
지정된 행으로 무조건 분기한다.
(예) 2 Moveto <P10> [3.3]
3 Goto #9 : 9번 행으로 분기한다.
·
·
9 Relax 5
·
·

(7) GoSub <행번호>
지정된 행으로 무조건 분기한다. Return 문을 만나면 다음 행으로 되돌아 온다.
(예) 2 Moveto <P10> [3.3]
3 GoSub #9 : 9번 행으로 분기한다.
4 Moveto <P11> [3.3]
·
·
9 Relax 5
·
·
Return : return 문에 의해 GoSub 문 다음의 4번행인 Moveto <P11> [3.3] 를 수행한다.

(8) Home
로보트의 자세를 초기 위치로 돌린다. 초기 위치란 로보트 각각의 관절각이 0도인 절대 위치를 의미한다.
(예) Home

(9) Initial <위치정보>
로보트의 각 관절을 사용자가 원하는 시작 위치로 옮긴다. 소프트웨어 측에서 경로를 발생하지 않고 위치값 만을 하드웨어부에 전달하므로서 각 관절을 조절할 수 있다. 위치 정보는 변수 선언부에서 참조할 수도 있고, 직접 데이타(Degree/Radian)를 입력 할 수도 있다.
(예) Initial <0,0>

(10) End
프로그램을 종료하고 시스템 모드로 복귀한다.

(11) Circle <위치정보1><위치정보2> [시간(sec)]
현재의 위치와 주어지는 두 점을 통해 이루어지는 삼각형의 외접원을 구하여 로보트의 각 관절을 보간하여 원을 그리게 한다. 이 때 로보트가 이동할 수 없는 위치가 입력되면 에러 메세지를 출력한다. 위치 정보는 변수 선언부에서 참조할 수도 있고, 직접 데이타(Degree/Radian)를 입력 할 수도 있다.
(예) Circle <P1><P0> [6.5]

(12) Speed (백분율(%))
전체 최대 속도를 고려해서 주어지는 백분율에 필요한 속도에 해당하는 이동 시간을 구하게 된다. 편집시에는 이동 시간을 묻지 않고 명령어 입력을 받게되며, 실제 동작시 현 위치와 목적 위치와의 거리를 이용하여 이동시간을 구하게 된다. Speed 명령어를 해제 시키려면 Speed 이후에 0을 입력하면 된다.
(예) Speed (50)

4. 경로 계획 함수

로보트의 초기 위치에서 목적 위치까지 시간에 기초한 공간상의 점들의 집합을 순서대로 발생시키는 것이다. 여기서 로보트의 팔이 지나야 할 공간상의

점, 즉 이 경유점들은 관절 좌표나 직교 좌표로 표현된다. 공간상에 두 점이 주어진 경우, 메뉴플레이터가 두 점을 잇는 직선 경로를 따라서 움직이는 직선 경로 (Straight-line Trajectory), 시작점과 끝점에서 위치와 자세를 만족하면서 경로를 따라서 유연하게 움직이는 관절 경로(Joint-interpolated Trajectory)를 생각할 수 있다.

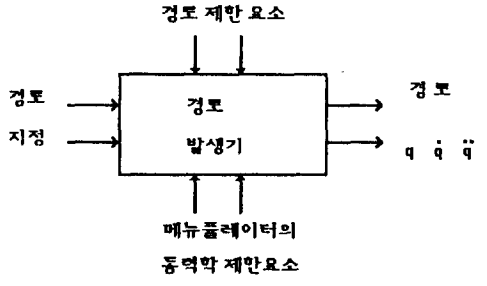

그림 2. 경로 발생기의 블럭 다이어그램

그림 2의 경로 발생기의 블럭 다이어그램을 보면, 경로 발생기는 경로의 제약을 나타내는 변수들을 받아들이고 시작점부터 끝점까지 로보트 손의 위치, 자세, 속도, 가속도를 연속적으로 출력한다. 이때 출력은 관절 또는 직교 좌표로 표현된다.

본 연구에서는 Moveto, Moverel, Movesingle등의 로보트 팔의 경로 계획을 위해 관절 보간법 중 4-3-4, 3-5-3, 큐빅 스플라인을 구현하였다. 또한 직선 보간법중 BDJP(Bounded Deviation Joint Path) 보간법을 이용해 경로 계획 알고리즘을 구현하였다. [3]

관절 보간을 하는 경우에 4-3-4나 3-5-3 경로는 시점과 종점, 가속점(Lift-off Point), 감속점(Set-down Point), 그리고 경유 시간이 필요로 하게 된다. 큐빅 스플라인의 경우에는 가상의 가속점과 감속점이 필요하다. 본 연구에서는 가속점과 감속점은 내부적으로 계산이 되며, 사용자는 시점과 종점, 경과 시간 혹은 속도만을 지정하게 된다.

Movevia 명령어에 의한 여러개의 경유점을 지나는 경우에는 지정된 시간을 분할하여 경유점간의 경로 계획 함수가 속도가 연속이 되도록 하면서 주어진 경유점들을 지나도록 경로점들을 발생 시킨다.

관절 보간법중 특히 큐빅 스플라인의 장점은 첫째, 속도와 가속도의 연속성을 허용하는 최저차 다항식이다. 둘째, 저차 다항식은 계산에 필요한 노력과 수치의 불안정을 줄인다.

BDJP 직선 경로는 메뉴플레이터가 반복적인 분할 방식을 이용해서 경로 계획을 하는 동안에 충분히 많은 중간점을 선택하여 직각 좌표계의 경로에서 미리 지정된 에러 범위내에만 머무르게 함으로써 직선 경로를 발생시킨다. 각 세그먼트간에는 일정 가속도에 의한 사다리꼴 형태의 속도 제어를 하게된다. 이 방식은 Circle 명령어에 의한 경로 발생시에도 이용된다.

5. 로보트 통합 관리 환경의 모드

(1) 통합 환경

로보트 운영 시스템의 목적은 사용자가 로보트의 경로 계획을 작성하고 실행하는데 필요한 모든 편의를 제공하는데 있다. 본 연구에 주 시스템으로 IBM-AT를 사용하고 있으므로 메모리의 주 모드의 운영에 필요한 특별한 스택이나 변수를 저장할 기억장소를 따로 제한할 이유가 없다. 로보트 프로그램의 저장은 IBM-AT내의 보조 기억장치를 이용한다.

하드웨어와의 공유 메모리를 유지하기 위한 시스템으로서 IBM-PC의 절대번지 Hex D00000번지를 사용하여 I/O를 사용하는 것에 비해 프로그램이 간편해지고 속도가 빨라지도록 하였다. [부록 1]

의외의 상황이나 긴급 상황이 발생했을 때의 처리를 위해서 인터럽트를 발생하는 방식으로 서로간에 대비할 수 있는 체제를 갖추고 있다. [부록 2]

그래픽 모드에서 모든 프로그램을 동작함으로써 한글의 직접 출력과 시뮬레이션이 가능 하도록 하였다. 또한 풀다운 메뉴 관리 시스템으로서 전체적인 동작을 관리 운영한다. 그리고 빠른 편집을 위한 Hot-key도 설정하여 메뉴바를 이동하지 않고도 프로그램을 운영할 수 있도록 개발하였다.

(2) 시스템 모드

① 새로운 화일 : 새로운 내용을 편집할 때 사용하며, 편집하던 내용이 있을 경우 새로운 화일을 편집하기 전에 화일의 저장여부를 묻는다.
② 불러오기 : 편집할 화일을 주 메모리에 불러 들인다.
③ 저장하기 : 편집중의 화일을 보조 기억장치에 저장한다.
④ 새이름으로 : 편집중인 화일의 이름을 변경하여 저장한다.
⑤ 도스로 외출 : 잠시 도스의 영역으로 빠져나올 경우에 사용하는 메뉴이다.
⑥ 끝내기 : 전체 관리 프로그램을 종료한다.

(3) 편집 모드

편집 모드란, 사용자가 화일을 작성하고 수정하도록 허용하는 프로그램을 의미한다. 홍익 로보트 언어에서는 팝업(Pop-Up) 메뉴방식에 의해 선택 사항을 선택하게 하는 대화식 에디터이다. 즉, 명령어의 종류, 경로 계획의 방식, 변수로서 포인트를 지정할 것 인지 아니면 직접 값으로 할 것인지를 사용자가 결정하게 하였다. 커서키에 의해 행 단위로 이동하게 되며 탭키에 의하여 윈도우 사이를 이동할 수 있다. 그림 3과 4는 명령어 편집 화면들이다.

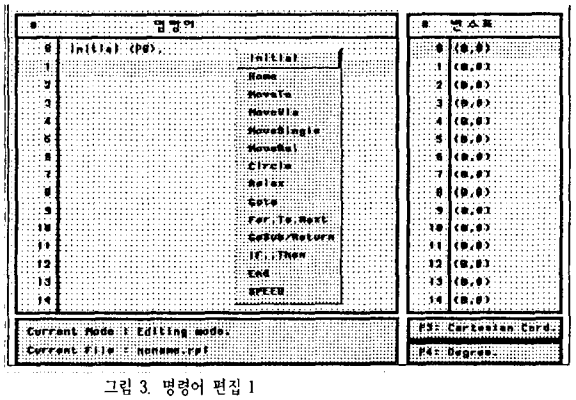

그림 3. 명령어 편집 1

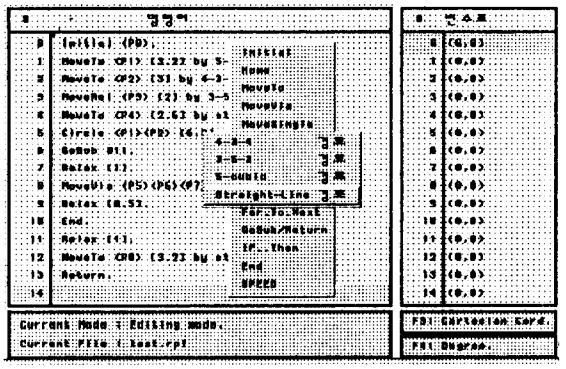

그림 4. 명령어 편집 2

변수 편집은 펑션키에 의해 관절 좌표계에서 Degree와 Radian 단위 모두 입력이 가능하며, 직각 좌표계로의 입력 또한 가능하다. 그림 5와 6은 변수 편집 화면들이다.

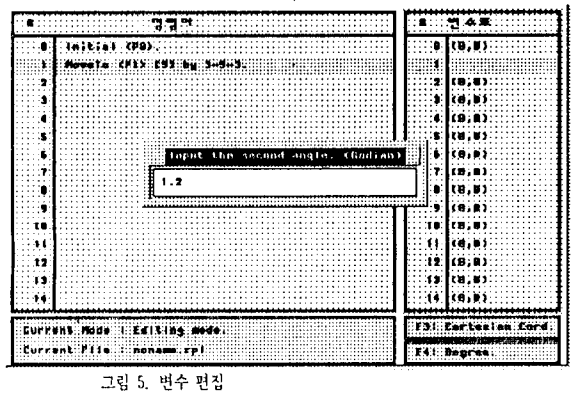

그림 5. 변수 편집

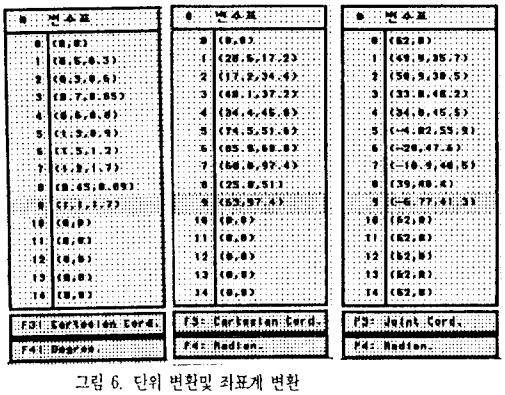

그림 6. 단위 변환및 좌표계 변환

(4) 컴파일 모드

소스 화일을 분석하여 그 명령어별로 미리 정의된 약속에 의하여 경로 계획 함수를 선택한후 계수를 생성해서 화일로 저장하는 역할을 한다. 그림 7은 경로 계획의 흐름도를 나타낸다.

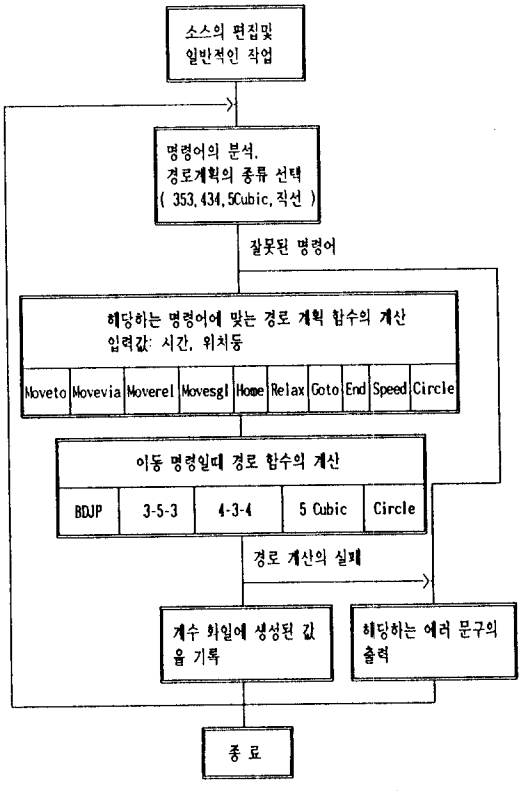

그림 7. 경로 계획 분석의 흐름도

(5) 실행 모드

① 시스템 초기화 : 일반적인 하드웨어와 소프트웨어 변수들을 초기화시키며 동작 준비를 한다.

② 순차적인 동작 : 생성된 계수 화일에 의해 한 스텝씩 실행을 하는 기능의 메뉴로 프로그램의 수정에 도움을 준다.

③ Simulation : 생성된 계수 화일에 의하여 그 동작을 보여주는 기능이다. 생성된 계수 화일에 의해 프로그램의 동작을 직접 하드웨어에 의존하지 않고 컴퓨터 상에서 볼 수 있게 하는 기능이다.

④ 전체적인 동작 : 생성된 계수 화일에 의하여 실제의 로보트 구동 모드로 들어가서 직접 하드웨어를 제어하는 기능이다. 실제의 시스템을 구동시키는 모드로서 하드웨어와의 통신이 매우 중요하게 된다.

컴퓨터와 시스템간의 정해진 공유 메모리 영역에 현재의 위치, 속도, 가속도를 관절별로 쇄신을에 따라 넣어준다. 또한 에러가 발생 할 경우에 대비한 상호간의 인터럽트가 정의되어 우발적인 사고를 방지하는 동작을 하게된다.

다음 그림 8은 실행 모드의 메뉴 화면을 보여주고 있으며, 그림 9는 전체 동작시의 화면이다.

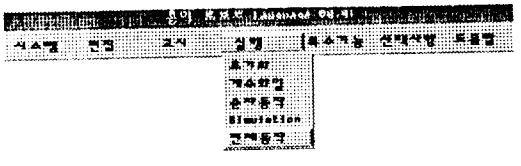

그림 8. 실행 모드 메뉴 화면

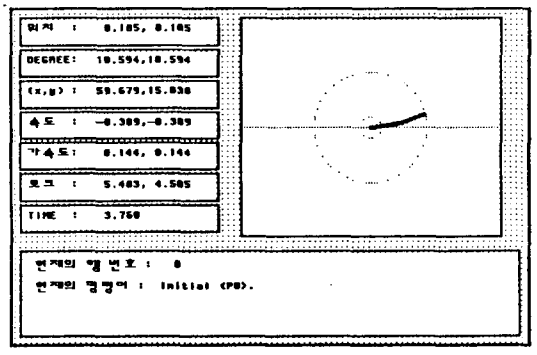

그림 9. 실행 모드 화면

(6) 교시 모드

다음 그림 10은 교시 모드의 메뉴 화면을 나타낸다.

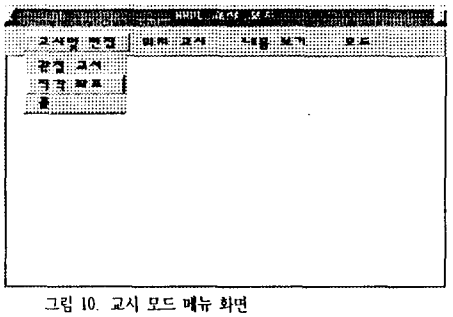

그림 10. 교시 모드 메뉴 화면

① 관절 좌표계 교시 : 각 관절의 위치를 화살표 키를 이용해서 교시한다. 이때, 관절의 증가분은 숫자키를 통해 조정 할 수 있다. 화면상에서 관절 위치 변화에 따른 로보트의 움직임을 볼 수 있다. 또한 각 관절의 관절값들과 각 관절들의 (x,y)좌표값을 직접 수치로 보여준다.

② 직각 좌표계 교시 : 숫자키를 이용한 교시시, 증감분이 선택된 x좌표, 또는 y좌표, 또는 동시에 x,y 좌표를 증감시키면서 교시한다. 교시후 위치 변수에 저장한다.

그림 11은 교시 모드 실행 화면이다.

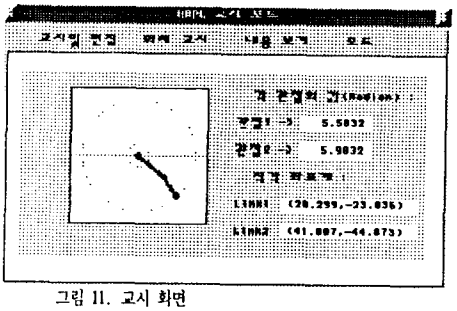

그림 11. 교시 화면

③ 교시후 편집 : 관절 교시나, 직각 좌표 교시후에 교시된 것을 위치 변수에 저장한 후 필요에 따라 직접 이동 명령어를 편집 할 수 있다. 그림 12는 교시후 편집하는 화면이다.

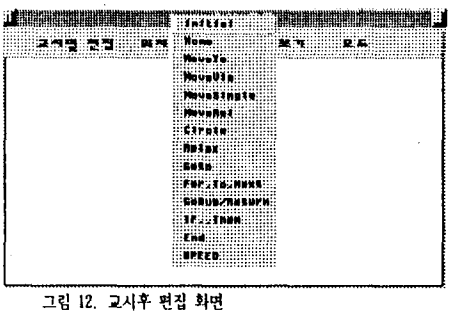

그림 12. 교시후 편집 화면

(7) 그외의 모드

① 위치 읽어오기 : 현재의 로보트 관절의 위치를 읽어와서 현재의 위치 변수에 저장한다.

② 위치로 이동 : 입력한 위치로 메뉴플레이터의 관절을 이동시킨다.

③ 원위치로 이동 : HOME의 위치로 이동한다.

④ 로보트 Setting : 기본적인 사항, Default 속도, 가속도, Home의 위치등을 바꿀 수 있는 기능을 한다.

⑤ 도움말 : 도움말은 편집 기능키에 대한 도움말과 프로그램 명령어에 대한 도움말을 사용자에 보인다. 각 모드에서 F1 키에 의해 필요한 도움말을 얻을 수 있다.

⑥ 외부 텍스트 화일 편집기를 이용해서 만든 텍스트 화일의 토큰 변환. : 외부의 텍스트 에디터에서 편집한 화일을 사용자 관리 시스템의 주모드에서 이용하는 이진 토큰 화일로 번환시켜 사용한다. 변환이 되면, 프로그램 토큰을 가진 filename.rpl과, 변수 선언부에 해당하는 filename.var 화일이 만들어진다. 변환 도중 에러를 발견하면 에러가 발견된 행을 표시하고 실행을 중단한다.

6. 결과및 검토

로보트가 갖추어야 할 기본적인 명령어를 모두 포함하는 로보트 언어를 개발하고, 이를 사용자가 쉽게 프로그램하고 실행시키는데 필요한 모든 편의를 제공할 수 있도록 통합 관리 로보트 운영 시스템의 성격을 갖는 통합 소프트웨어를 개발하여 홍익 직접 구동 로보트에 적용하였다.

본 소프트웨어는 한글 출력을 지원하며, 메뉴 방식에 의해 구동되며, 작업 화일과 각종 변수를 관장하는 편집 모드, 작업 수행시 규정 경로 발생 함수의 계수를 산출하고 이를 계수 화일로 보관하기 위한 컴파일 모드, 작업 변수를 교시하는 교시 모드, 작업 수행시 발생하는 규정 경로의 수치값과 이를 그래픽으로 나타내 주는 시뮬레이션 모드, 그리고 편집모드에서 작성한 화일을 수행하여 그 출력을 로보트 주제어기의 공유 메모리로 전송하고 주제어기와 약속된 인터럽트 규약을 통해 정보 전달을 하도록 하는 실행 모드로 구분하여 개발하였다.

본 연구의 결과로, 로보트의 동적 제어 알고리즘 개발및 이의 실현을 위한 하드웨어 개발에 필요한 소프트웨어 기술을 제공할 수 있으며, 나아가 로보트 언어 및 로보트 관리 통합 환경을 개발하는데에 소프트웨어 기술을 제공할 수 있을 것이다.

참 고 문 헌

1. William A. Gruver, Barry I. Soroka, PROGRAMMING, HIGH LEVEL LANGUAGES, ENCYCLOPEDIA OF ROBOTICS SYSTEM AND CONTROL, VOL.2, Industrial Training Corporation, 1987.
2. Andrew Kusiak, PROGRAMMING, OFF-LINE LANGUAG-ES, ENCYCLOPEDIA OF ROBOTICS SYSTEM AND CONTROL, VOL. 2, Industrial Training Corporation, 1987.
3. K.S. Fu, R.C. Gonzalez, C.S.G. Lee, ROBOTICS Control, Sensing, Vision, and Intelligence, McGraw-Hill, 1987.
4. 송희선, IBM PC/XT/AT용 TURBO EDITOR TOOLBOX, 기전연구사, 1987.
5. Ben Ezzel, Programming the IBM User Interface USING TURBO PASCAL, Addison-Wesley, 1989.
6. Alan Watt, THREE-DIMENSIONAL COMPUTER GRA-PHICS, Adisson-Wesley, 1989.
7. John J. Craig, INTRODUCTION TO ROBOTICS MECHA-NICS AND CONTROL, 2nd edition, Addison-Wesley, 1989.
8. TURBO PASCAL Reference Guide Version 5.0, Borland, 1991.
9. Michael Tischer, PC System Programming for de-velopers, Abacus, 1989.
10. System BIOS for IBM PC/XT/AT Computers and Compatibles, Phoenix, 1989.
11. RoboTalk User's Manual, Rhino Robots inc., 1991.
12. Herbert Schildt, C Power User's Guide, Osbor-ne McGraw-Hill, 1988.
13. 김영택, 컴파일러 구성론, 희중당, 1990.
14. 오세만, 컴파일러 입문, 정익사, 1988.

부록1. 하드웨어와 소프트웨어간의 공유메모리 규약

하드웨어와 소프트웨어 서로간의 통신을 위하여 정의된 메모리상의 위치 규약이다. 항목으로는 소프트웨어측에서 전달해 주는 위치, 속도, 가속도 정보와 하드웨어측에서 전달해주는 관절들의 현재 위치 정보, 그리고 인터럽트이 요구시에 포함되는 세부 정보이다. 자세한 내용은 표 1에 정리하였다.

소프트웨어측에서 기록하는 정보	D000:0000 D000:0004 D000:0008	위치값(4바이트의 실수): 관절 1 속도값(4바이트의 실수): 관절 1 가속도값(4바이트의 실수): 관절 1
	D000:0100 D000:0104 D000:0108	위치값(4바이트의 실수): 관절 2 속도값(4바이트의 실수): 관절 2 가속도값(4바이트의 실수): 관절 2
하드웨어측에서 기록하는 정보	D000:0200 D000:0204	관절 1 의 현재 위치값(4바이트) 관절 2 의 현재 위치값(4바이트)
인터럽트의 요청 영역	D000:0300 D000:0301	하드웨어에서 소프트웨어로의 인터럽트 소프트웨어에서 하드웨어로의 인터럽트

표 1. 공유 메모리의 규약

부록2. 하드웨어와 소프트웨어간 인터럽트 규약

1. 하드웨어에서 소프트웨어로 요구하는 인터럽트
   ① 시스템의 하드웨어적인 에러의 발생시
   ② 소프트웨어에 요청에대한 응답
   ③ 동작모드에서 동작개시와 종료에 대한 응답
   ④ 교시 모드에서의 위치값의 반환
2. 소프트웨어에서 하드웨어로 요구하는 인터럽트
   ① 사용자에 의한 긴급한 정지의 요구시
   ② 프로그램의 동작의 이상시의 정지
   ③ 교시 모드에서 현재 위치값의 요구
   ④ 동작 모드에서 동작개시의 요구

하드웨어에서 소프트웨어로의 인터럽트는 IBM-PC 시스템의 하드웨어 인터럽트 IRQ 5번, 소프트웨어 인터럽트 13번을 이용하며, 소프트웨어적인 인터럽트는 주제어기에 연결된 8255A를 통하여 전달되며, 그 상세한 요구사항은 공유 메모리를 통하여 전달된다. 자세한 내용은 표 2에 정리하였다.

```
〈 하드웨어에서 소프트웨어로의 인터럽트의 발생〉
(1) 공유메모리의 절대번지 (D000:0300)에 인터럽트의 종류를 기록
    D000:0300(1 바이트)
    bit0 = 하드웨어 시스템에 에러 발생했음
       1 = 관절의 위치값을 새로 기록했음
       2 = 시스템이 기동을 시작했음
       3 = 시스템의 동작이 종료되었음을
(2) IRQ 5 번의 발생

〈 소프트웨어에서 하드웨어로의 인터럽트의 발생〉
(1) 공유메모리의 절대번지 (D000:0301)에 인터럽트의 종류를 기록
    D000:0301(1 바이트)
    bit0 = 사용자에 의한 긴급한 정지 요구
       1 = 관절의 위치값의 요구
       2 = 시스템의 기동을 요구
       3 = 시스템의 동작의 종료 요구
(2) 주제어기에 연결된 8255A(PPI)를 통하여 주제어기 측에 인터럽트를
    발생
```

표 2. 인터럽트 규약의 형식