그래픽 시뮬레이터를 갖는 로보트 프로그래밍 시스템에 관한 연구

(A Study on Robot Programming System with Graphic Simulator)

呂 政 範*, 韓 浚 洙*, 崔 宗 秀**

(Jeong Beom Ryeo, Joon Soo Han and Jong Soo Choi)

要 約

본 로보트 프로그래밍 시스템은 컴퓨터로 수치제어되는 로보트 머니퓰레이터에 대한 off-line 프로그램의 작성을 위해서 설계되었다. 이 시스템은 인터프리터, 작업환경모델, 그래픽 시뮬레이터와 콘트롤러로 구성된다. 그리고 이 시스템은 디버깅 도구로써 그래픽 시뮬레이터를 가지며 이 시뮬레이터는 로보트의 동작을 터미날 상에서 그래픽으로 시뮬레이션하는 것이다. 그럼으로써 로보트의 가능한 동작을 관찰할 수 있는 것이다.

Abstract

This robot programming system is designed for off-line programming of numerical controlled robot manipulator. This system consists of manipulation interpreter, world model manager, graphic simulator and arm controller for simple robot programming language. The system has graphic simulation system as a debugging tool for task programming and it simulates the robot motion graphically on a CRT terminal, which makes the assessment of the possible robot motion.

Ⅰ. 서 론

오늘날 로보트는 산업의 각 분야에서 사용되고 있으며 그 응용성을 확장하기 위한 많은 연구가 실행되고 있다. 현재까지 로보트 프로그래밍에 주로 사용되어 온 방법(teach and repeat)은 뛰어난 효과에도 불구하고 다축 로보트의 폭넓은 사용에는 아직 많은 프로그래밍 시간을 필요로 한다. 이러한 산업용 로보트들의 잠재적인 사용의 한계성 때문에 작거나 중간 크기의 생산과 재 프로그래밍이 빈번한 자동생산 라인은 전체 시스템을 프로그래밍 하는데 필요한 시간으로 인해 비생산적인 기간이 증가한다. 이러한 문제의 해결 방법은 off-line 프로그래밍 시스템의 응용이다.[1,2] 그러나, 이러한 off-line 프로그래밍 시스템은 프로그램 작성시 발생할 수 있는 에러를 확인하는 강력한 디버깅도구(debugging tool)가 필요하게 되며 그래픽 시뮬레이터가 가장 효율적인 디버깅 도구라고 할 수 있다.[3]

본 논문에서는 이러한 특성을 고려하여 컴퓨터와 로보트의 하드웨어에 대한 전문적인 지식이 없이도 쉽게 작업(task) 프로그램을 작성할 수 있도록 off-line 프로그래밍 시스템을 설계하였다.

Ⅱ. 로보트 프로그래밍 시스템

1. 로보트 오퍼레이팅 시스템(robot operating system)

일반적으로 OS(operating system)란 컴퓨터 시스

*準會員, **正會員, 中央大學校 電子工學科
(Dept. of Elec. Eng., Chungang Univ.)
接受日字 : 1985年 12月 12日

템 내의 각 장치, 즉 중앙처리장치, 주 기억장치, 보조 기억장치, 입 출력장치 등을 보다 효율적으로 이용하고, 또한 이용자가 컴퓨터 시스템을 보다 편리하게 이용할 수 있게 하여주는 일종의 제어 프로그램이다.[4] 이용자는 OS를 통해서 하드웨어를 사용하므로 하드웨어에 관한 전문적인 지식이 없이도 시스템을 이용 할 수 있다.

일반적인 OS의 기능은 다음과 같다.

① 프로그램을 작성하거나 수정할 수 있는 기능
② 각종 언어를 기계어로 번역하는 기능
③ 화일의 관리기능
④ 사용자가 작성한 프로그램을 실행시키는 기능

본 논문에서는 이러한 OS의 특징을 갖는 특수 목적의 OS를 설계하여서 ROS라고 하였으며 프로그래머가 직접 단말기(terminal)를 사용하여서 작업을 마칠 때까지 대화형으로 작업을 지시할 수 있다. 이 ROS는 6개의 모드(mode)를 가지며 각 모드는 명령어(command)이거나 함수(function)이고 표 1에 각 모드와 각 모드에서 사용되는 명령어를 나타내었다. 각 모드는 모니터 모드에서 시작하여 다른 모드로 전환되며 사용자가 이러한 전환을 지시한다.

표 1. 로보트 프로그래밍 시스템의 모니터 명령어
Table 1. Robot programming system monitor command set.

MONITOR

WORLD	EDIT	LIST	SAVE	LOAD	RUN
NEW	MOVE	FILENAME	PROGRAM	PROGRAM	GRAPHIC
ADJUST	CLOSE				
READ	OPEN	PROGRAM	CONTROL	CONTROL	ROBOT
STORE	JUMP		DATA	DATA	
DEFINE	REPEAT				
	AGAIN				
	RESET				
	MODIFY				
	CLEAR				
	END				

1) World mode

작업환경 모델(world model)을 작성하는 모드로써 작업환경 모델이란 로보트가 동작할 작업환경 속에서 로보트나 여러가지 물체(object)의 위치 등의 배치구조를 정해진 좌표계에 대한 변환(transformation)으로 기술함으로써 작업환경을 수식적으로 표현하는 것이다. 이러한 작업환경 데이타는 소프트웨어에 의해서 구조적 데이타 베이스(structual data base)로 변환되며 물체의 모양이나 위치 그리고 물체 상호간 관계로써 정의된다.

이 모드에서 사용되는 명령어는 다음과 같이 정의하였으며 이 모드에 대한 플로우차트는 그림 1에 나타내었다.

① New－새로운 모델 작성시 사용
② Adjust－기존 모델의 수정
③ Read－보조 기억장치로 부터 주 기억장치로 모델을 이동
④ Store－주 기억장치에서 보조 기억장치로 모델을 저장
⑤ Define－기존 데이타 이용 새로운 데이타 정의

2) Edit mode

프로그램 에디터(editor)는 사용자에게 동작명령(motion command)을 사용하여 새로운 프로그램을 작성하고 존재하는 프로그램을 수정하는 기능을 제공한다. 프로그램은 한 행에 하나의 명령어가 사용되며 각 명령어는 각각 하나의 동작과 대응된다.

이 모드에서 사용되는 명령어는 다음과 같이 정의하였으며 이 모드에 대한 플로우차트는 그림 2에 나타내었다.

① Move－주어진 목표점으로 로보트를 이동시킴
② Close－로보트의 손을 닫음
③ Open－로보트의 손을 열음
④ Jump－프로그램내의 주어진 행으로 이동
⑤ Repeat－반복 루프의 시작 표시

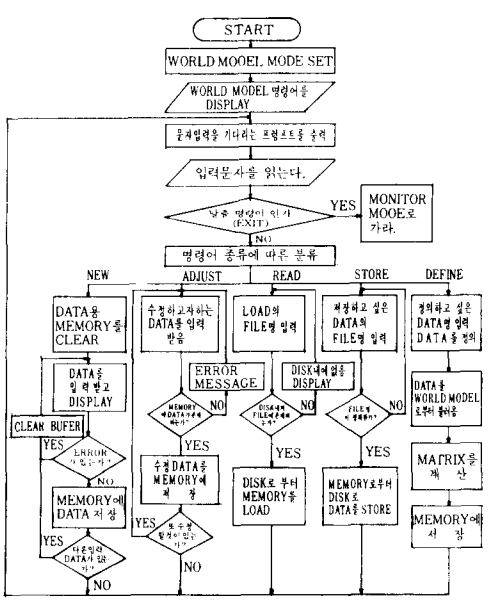

그림 1. 작업환경 모델 모드에 대한 플로우차트
Fig. 1. World model mode flowchart.

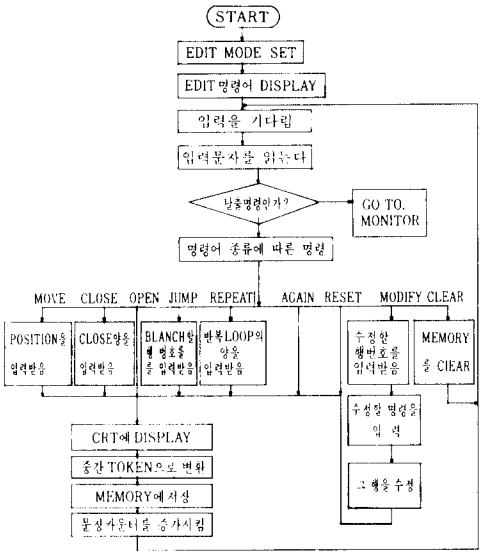

그림 2. 에디트 모드에 대한 플로우차트
Fig. 2. Edit mode flowchart.

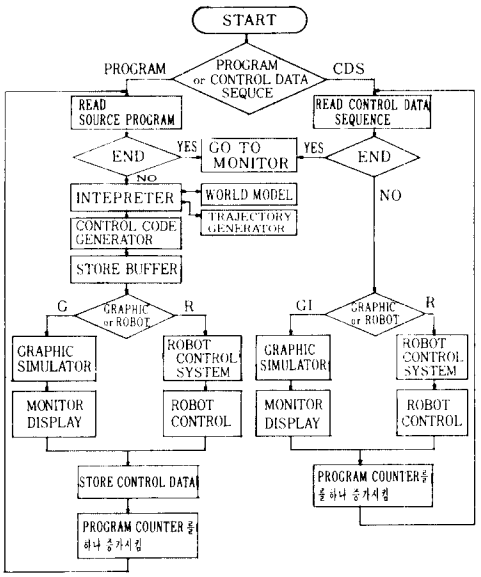

그림 3. 수행 모드에 대한 플로우차트
Fig. 3. RUN mode flowchart.

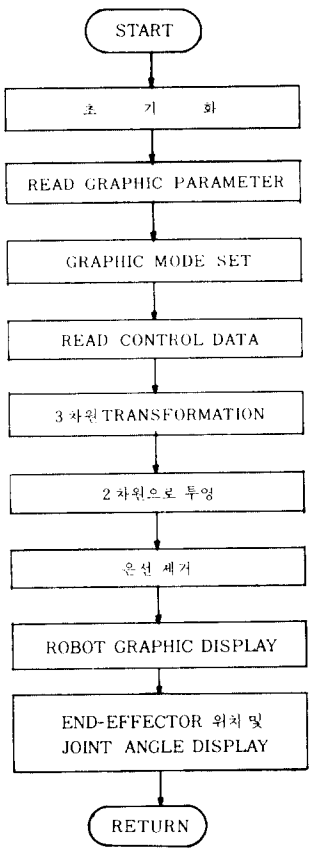

그림 4. 그래픽 시뮬레이터에 대한 플로우차트
Fig. 4. Graphic simulator flowchart.

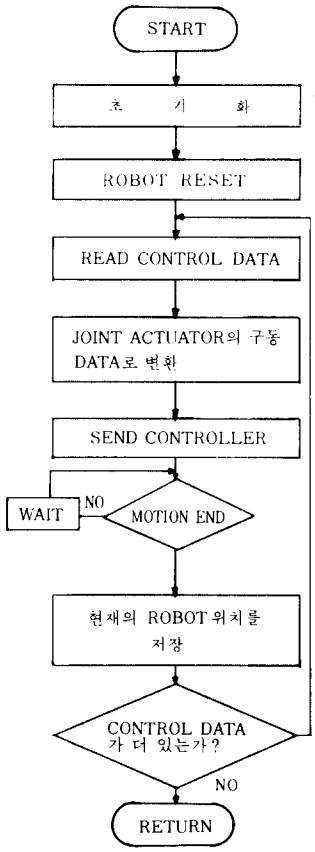

그림 5. 로보트 콘트롤러 구동 서브루틴에 대한 플로우차트
Fig. 5. Robot controller driving subrutine flowchart.

⑥ Again－반복 루프의 종료표시
⑦ Reset－로보트를 초기상태로 만듦
⑧ Modify－프로그램의 수정시 사용
⑨ Clear－메모리내의 프로그램을 소거
⑩ End－프로그램의 마침 표시

3) Run mode

이 모드는 프로그래머에 의해서 작성된 프로그램을 실행시킬 수 있으며 언제든지 필요할 때마다 다시 실행시킬 수 있다. 이 모드에서는 이미 작성된 프로그램을 사용하여 실제로 로보트를 구동하거나 디버깅용인 그래픽 시뮬레이터를 작동시킬 수 있으며 시스템에 의해서 콘트롤러 구동 데이타가 자동으로 생성되어진다. 이렇게 작성된 콘트롤러 구동 데이타는 실제로 로보트를 구동하는 데 사용할 수 있다.

이 모드에 대한 플로우차트는 그림 3 에서 그림 5 까지에 나타내었다.

4) List mode, save mode, load mode

리스트 모드는 주 기억장치나 보조 기억장치에 저장되어 있는 화일을 터미날(terminal) 상에 나타내주는 명령이 있다. 세이브 모드는 에디트 모드상에서 작성된 프로그램을 주 기억장치나 보조 기억장치에 저장하는 명령이 있다. 로드 모드는 세이브 모드의 반대 기능으로 보조 기억장치로 부터 주 기억장치로 읽어들이는 명령이 있다.

2. 로보트 언어 번역기(robot language interpreter)

컴퓨터가 수행하려는 프로그램은 기계어로 작성되어 있을 경우에만 곧 바로 실행시킬 수 있으며 기계어가 아닌 다른 언어로 작성되어 있을 때에는 이를 기계어로 변환한 후에 수행하여야 한다.[4] 이러한 변환작업을 언어번역이라고 부르며 본 논문에서는 앞에서 정의한 명령어를 인터프리터 기법을 사용하여서 번역한다. 인터프리터 기법은 소오스 프로그램의 각 문장이 한번에 하나씩 번역과 수행이 이루워지며 문장은 수행될 때마다 새로이 번역되게 된다. 따라서 프로그램의 수행시간은 길어지나 프로그램에서 사용된 변수들의 위치, 형, 값들을 알 수 있는 수행시에 번역이 행해지므로 배열의 크기를 동적으로 바꾸거나 변수형을 바꿀 수 있고 각 문장에 대해 보다 정확한 에러 메세지를 제공할 수 있다.

인터프리터의 기본적인 동작은 다음과 같다.

① 문장 카운터가 가리키는 문장을 소오스 프로그램으로부터 옮겨 온다.
② 문장 카운터를 증가시킨다.
③ 문장을 분석, 어떤 동작을 수행할지의 결정
④ 해당 서브루틴의 호출 문장을 실행

이와 같이 인터프리터는 위의 4 단계 작업을 반복적으로 수행하는 루우프 구조와 각 동작을 수행하는 서브루틴들로써 구성된다.

3. 로보트 시뮬레이션 시스템(robot simulation system)

로보트용 언어를 사용하는 프로그래밍 방법에는 많은 장점이 있으나 이 방법은 많은 에러를 발생할 수 있으므로 실질적으로 로보트가 어떤 동작을 할 것인지 예상하기란 어렵다. 그러므로 로보트 모션의 그래픽 시뮬레이션은 본질적으로 디버깅용이다. 그래픽 시뮬레이터는 작업 환경 내에서의 로보트를 디스플레이한다. 따라서 작업 환경에 대한 정보가 충분하다면 작업환경 내에서의 로보트의 모션을 디스플레이를 통해서 눈으로 확인할 수 있으므로 실제로 로보트를 구동하기 전에 프로그램의 에러를 확인하고 수정할 수 있게 하여준다.[3]

이러한 컴퓨터 그래픽을 이용한 로보트 시뮬레이션 시스템의 장점은 다음과 같다.[6,7]

① 모션 시뮬레이션을 통해 작업환경 내에서의 로보트의 동작을 분명히 구상화할 수 있다.
② 프로그래머가 쉽고 빠르게 모션을 시뮬레이션 할 수 있으므로 신속하게 프로그램을 작성할 수 있다.
③ 프로그램의 실행 가능성을 여러가지로 평가 고찰할 수 있다.

이상에서 보는 바와 같이 그래픽 시뮬레이터를 갖는 프로그래밍 시스템은 모션을 프로그래밍하는 데 훌륭한 도구를 갖는 것이며 이 시뮬레이터는 로보트의 성능(performance)을 평가하는 강력한 방법이다.

Ⅲ. 시스템 시뮬레이션 및 결과

본 시뮬레이션을 위한 로보트 프로그래밍 시스템의 모델 로보트로는 RHINO-XR2를 사용하였으며 프로그래밍 시스템은 모니터와 디스크 드라이버를 갖는 64K 메모리의 apple Ⅱ+컴퓨터에 입력시켰다.

그리고 로보트 프로그래밍 시스템은 다른 컴퓨터 시스템으로의 이동성을 높이고 수정을 용이하게 하기 위해서 basic으로 작성하였으며 기계어로 컴파일링(compiling) 하여서 사용하였다.

그림 6 에는 본 프로그래밍 시스템의 전체 블럭도를 나타내었다.

RHINO-XR2 로보트는 5 개의 축을 가지며 다음 그림 7 에 RHINO-XR2의 제원과 본 시뮬레이션에서 사용한 로보트의 기준위치를 나타내었다.

본 논문의 로보트 프로그래밍 시스템 중 world

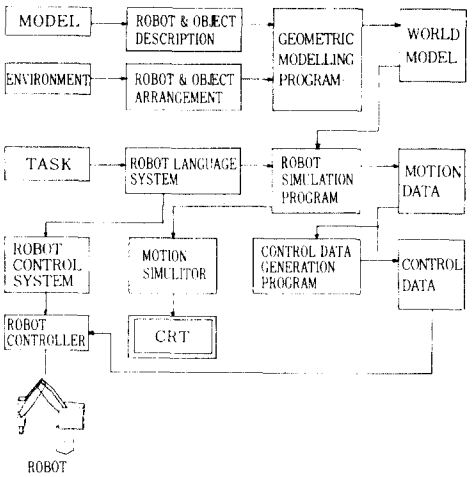

그림 6. 로보트 프로그래밍 시스템 전체 블럭도
Fig. 6. Block diagram of robot programming system.

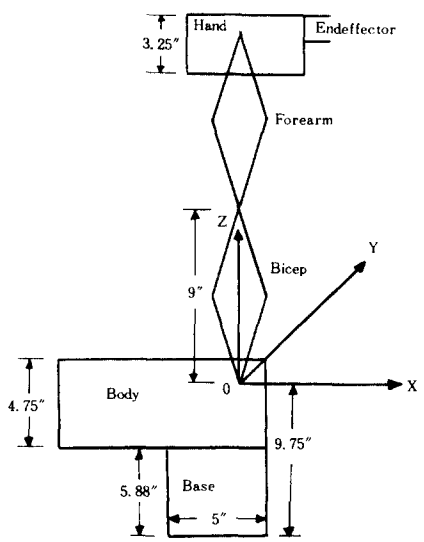

그림 7. RHINO-XR2의 체원과 기준위치
Fig. 7. Configuration and reference position of RHINO-XR2.

model mode에서는 작업환경 내에서의 object들을 modelling하는 것이며 이 mode에서는 사용자가 하나의 argument를 설정하고 그 argument에 대한 data값을 Denaviter와 Hartenberg에 의해서 발전된 A matrix라고 부르는 homogeneous transformation으로 기술되고 사용자가 object들에 대한 data를 world mode 명령어를 사용해서 쉽게 바꿀 수 있다.

$$T=\begin{bmatrix} Nx & Ox & Ax & Px \\ Ny & Oy & Ay & Py \\ Nz & Oz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

여기서 vector N, O, A는 로보트의 방향을 기술하는 단위 vector이고 P는 base frame의 좌표계에서의 로보트 위치를 기술하는 vector이다. 그리고 본 시뮬레이션에서는 argument를 20개까지 설정할 수 있도록 하였으나 메모리의 여유에 따라서 그 이상도 쉽게 확장할 수 있다.

Edit mode에서는 이 mode에서 사용되는 명령어를 이용한 line editing 방식을 사용해서 한 행씩 프로그램을 작성할 수 있도록 하였다.

로보트의 joint 위치를 계산하기 위한 inverse kinematic 방정식은 기하학적인 구조를 이용한 해석법을 사용하여서 풀이하였으며 방정식을 풀기 위한 조인트 변수는 그림 8 과 그림 9 에 나타내었다.

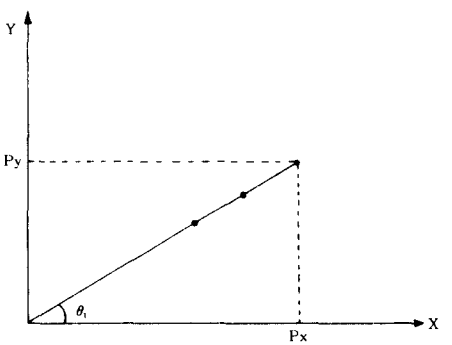

그림 8. X-Y 평면에서의 joint 변수
Fig. 8. Joint variable on X-Y plane.

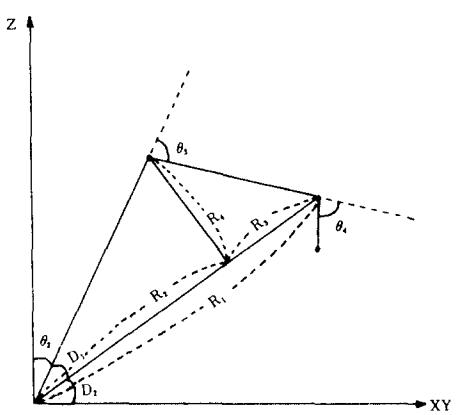

그림 9. 키네메틱 방정식을 기술하는 joint 변수
Fig. 9. Kinematic equation is described by joint variables.

$R1=(Px^2+Py^2+Pz^2)^{1/2}$ $R2=R3=R^2 1/2R1$

$R4=(Py^2+R^2 2)^{1/2}$ $D1=TAN^{-1}(R4/R2)$

$D2=TAN^{-1}(Pz/\sqrt{Px^2+Py^2})$

$\theta 1=TAN^{-1}(Py/Px)$

$\theta 2=\pi/2-R5-D2$

$\theta 3=2R5$

$\theta 4=\pi/2-\theta 2-\theta 3$

$$\theta 5=TAN^{-1}\frac{\sin\theta 1*Nx-\cos\theta 1*Ny}{\sin\theta 1*Ox-\cos\theta 1*Oy}$$

또한 RHINO-XR2 전용 로보트 콘트롤러는 serial contol만이 가능하므로 복잡한 trajectory control에는 제한이 따른다. 따라서 trajectory generation을 위한 방정식은 다음과 같은 간단한 조인트 보간법(interpolated method)를 사용하였다.

$$\vec{P}=\vec{A}t+\vec{C}$$

여기서 $\vec{P}$는 조인트의 최종 위치이고 t는 샘프링 시간, $\vec{C}$는 시작 위치이며 $\vec{A}$는 조인트의 속도이고 $\vec{P}$, $\vec{A}$, $\vec{C}$는 vector이다.

RHINO-XR2전용 콘트롤러는 RHINO 로보트를 구동시키는 데 필요한 data 형식이 지정되어 있으며 본 프로그래밍 시스템의 시뮬레이션에서는 이러한 data 형식에 맞는 콘트롤러 구동 data를 생성하여 RS-232 카드를 통해서 콘트롤러에 보냄으로써 실제로 로보트를 구동시켰다. 그리고 RHINO-XR2 로보트는 각 링크(link)의 동작범위가 제한되어 있으며 다음의 표 2에 링크와 모터의 동작제한 범위를 나타내었다. 따라서 로보트를 구동시킬 경우에는 모든 동작이 이 범위 안에서 이루어지도록 고려하여야 하며 본 시스템에서는 프로그래머가 작성한 프로그램을 그래픽 시뮬레이션을 통해 시뮬레이션하는 과정에서 모든 링크의 동작이 이 범위에서 벗어나는지의 여부를 검사하여 범위를 벗어나는 경우에는 그 링크에 대한 에러 메세지를 표시하도록 하였다.

본 시스템의 그래픽 시뮬레이터는 로보트의 모션을 키네메틱적으로 시뮬레이션하며 모션은 조인트 보간법에 의해서 디스크리트(discrete)하게 시뮬레이트 된다. 거기에다 로보트의 엔드 이펙터(end-effector)의 위치 및 조인트 각도를 표시함으로써 디버깅에 도움이 되도록 하였다.[10,11]

본 시스템을 이용한 실용 예로써 2개의 구멍을 갖는 블럭에 2개의 핀을 삽입하는 간단한 작업을 수행하기 위해 edit mode에서 작성한 프로그램과 이 프로그램을 시뮬레이터를 이용하여 모션을 시뮬레이션하는 과정을 보였다(그림10).

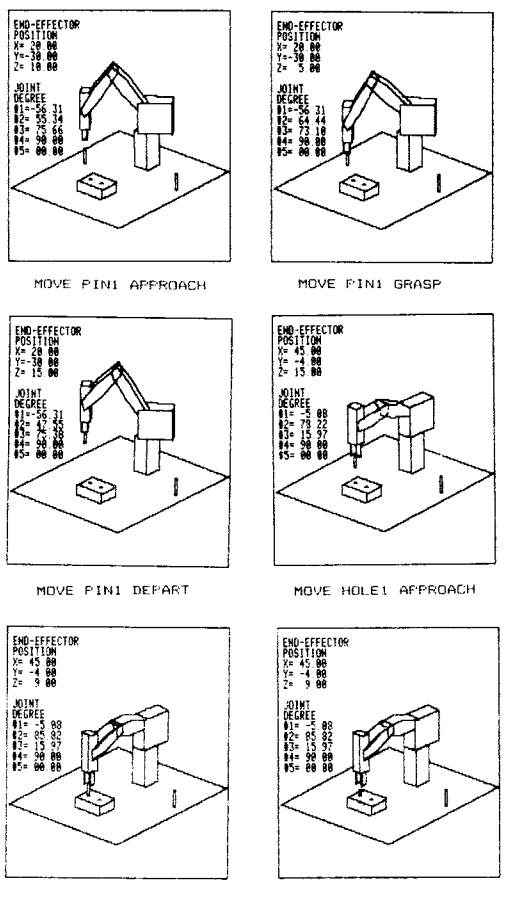

그림10. 그래픽 시뮬레이터의 모션 시뮬레이션 과정예
Fig. 10. An example of motion simulation process on the graphic simulater.

표 2. RHINO-XR2의 링크 및 모터의 동작범위
Table 2. Action range of the link of RHINO-XR2 and the motor.

Motor No.	Link의 動作 범 위	Motor의 動作 범 위	Motor의 動作 범 위
M_1	$-135°\le\theta_{a1}\le 135°$	$-135°\le\theta_{p1}\le 135°$	$-964\le p_1\le 964$
M_2	$-45°\le\theta_{a2}\le 135°$	$-45°\le\theta_{p2}\le 135°$	$-409\le p_2\le 1227$
M_3	$-135°\le\theta_{a3}\le 135°$	$-180°\le\theta_{p3}\le 270°$	$-1639\le p_3\le 2455$
M_4	$-135°\le\theta_{a4}\le 135°$	$-360°\le\theta_{p4}\le 540°$	$-4557\le p_4\le 6835$
M_5	$-180°\le\theta_{a5}\le 180°$	$-180°\le\theta_{p5}\le 180°$	限界없음

이 프로그램은 표3에 나타내었으며 다음과 같이 사용된다.

① Move pin1 approach : 월드 모드에서 첫번째의 핀 위치로 이동

② Move pin1 grasp : 첫번째 핀의 잡는 위치로 이동

③ Close 30 : 핀을 잡음
④ Move pin1 depart : 핀을 들어올림
⑤ Move hole1 approach : 핀을 구멍의 위치에 접근시킴

나머지도 마찬가지 방법으로 사용되며 로보트의 이동 경로는 프로그래머가 프로그램을 작성할 때에 임의로 지정해 줄 수 있다. 즉 4번행과 5번행 사이에 다음과 같은 명령어를 삽입하여 실행시킴으로써 이동 경로를 바꿀 수 있다.

Move pin1 mid hole1 : 핀과 구멍의 중간 위치로 이동

표 3. 작업기술 task 프로그램의 예
Table 3. Example of the task program.

ROBOT PROGRAMMING SYSTEM	
1 MOVE PIN1 APPROACH	
2 MOVE PIN1 GRASP	〈EDIT〉
3 CLOSE 30	
4 MOVE PIN1 DEPART	1 MOVE
5 MOVE HOLE1 APPROACH	2 CLOSE
6 MOVE HOLE1 INSERT	3 OPEN
7 OPEN	4 JUMP
8 MOVE HOLE1 DEPART	5 REPEAT
9 MOVE PIN2 APPROACH	6 AGAIN
10 MOVE PIN2 GRASP	7 RESET
11 CLOSE 30	8 MODIEY
12 MOVE PIN2 DEPART	9 CLEAR
13 MOVE HOLE2 APPROACH	A END
14 MOVE HOLE2 INSERT	B EXIT
15 OPEN	
16 MOVE HOLE2 DEPART	HIT KEY
17 END	

Ⅳ. 결 론

본 논문에서는 로보트의 모션을 기술하는 전용 명령어를 정의하였으며 컴퓨터의 종류에 무관하게 적용이 가능하도록 고급언어로 시스템을 설계하였다. 또한 이 시스템은 모듈적 구조를 갖도록 하였으므로 필요한 모듈을 교환함으로써 다른 다양한 응용에 확장 적용이 가능하도록 하였다. 그리고 모션을 기술한 프로그램의 디버깅용으로써 그래픽 시뮬레이터를 사용하여서 디버깅에 도움이 되도록 하였으며 시뮬레이션 하는 과정에서 콘트롤러 구동 데이타를 생성하도록 하였으므로 실행시 마다 프로그램을 번역할 필요가 없이 이 콘트롤러 구동 데이타를 이용함으로써 실행시간을 줄일 수 있다. 그러므로 본 시스템을 이용하여 프로그램을 작성하고 시뮬레이션을 거쳐 에러를 수정한 후 번역하여 콘트롤러 구동 데이타를 생성할 수 있으며 이 콘트롤러 구동 데이타는 전 프로그래밍 시스템이 없어도 실행이 가능하다. 따라서 이 콘트롤러 구동 데이타의 형식을 teaching 방식에서 사용하는 데이타 형식과 같이 하면 기존의 teaching 방식으로 프로그래밍하는 산업용 로보트들에 그대로 적용시킬 수 있으며 또한 여러 대의 로보트에 다양하게 적용시킴으로써 로보트의 유용성을 향상시킬 수 있을 것이다.

參 考 文 獻

[1] B.E. Shimano, "VAL-II: A new robot control system", *International conference on ROBOTICS. pp.* 278, 1984.

[2] M. Weck, "Requirements for robot off-line programming shown at the example ROBEX", *Advanced software in robot ics,* pp. 321, 1983.

[3] T. Arai, "A robot language system with a color graphic simulator", *Advanced software in robotics,* 1984.

[4] J. Donovan, "Systes programming," *McGraw-Hill,* 1972.

[5] R.P. Paul, robot manipulator mathematics, programming and control, *MIT Press,* 1982.

[6] T. Sata, "Robot simulation system as a task programming tool", *The 11th ISIR,* 1981.

[7] W.B. Hegibotham, "Robot application simulation", *The industrial robot,* June 1979.

[8] W.J. Crochetiere, "Locating the wrist of an elbow-type manipulator", *IEEE trans on system, man and cybernetic,* vol. SMC-14, no. 3, 1984.

[9] Hands on Introduction to Robotics, *The Manual-SANDHU,* 1983.

[10] 한준수, 성원기, 최종수, 하용수 "RHINO-XR2 로보트의 그래픽 시뮬레이션에 대한 연구", 대한전자공학회 합동학술발표회 논문집, vol 9, pp. 17-20, 1985.

[11] 여정범, 성원기, 최종수, 하용수, "마이크로 컴퓨터를 이용한 로보트 프로그래밍 시스템에 관한 연구," 대한한전기공학회 대한전자공학회 합동학술발표회대, pp. 13-16, 1986, 4 *