Trans. KIEE. Vol 37. No. 11 NOV. 1988

論 文
37~11~6

SCARA형 로보트의 프로그래밍 언어개발 및 구성에 관한 연구

A Study on the Development and Construction of a Programming Language for SCARA Type Robots

高 明 三*·李 範 熙**·李 起 東§·金 大 元§
(Myoung-Sam Ko · Bum-Hee Lee · Ki-Dong Lee · Dae-Won Kim)

요 약

본 논문에서는 SCARA형 산업 로보트를 위한 전용언어의 설계방법과 설계기법 및 구조를 제시한다. 새로 제안된 언어는 C-언어 및 8086 어셈블러어로 작성되었으며, 모듈화 되어있고, 확장 가능하게 설계되어 있다. 이 언어는 전체의 흐름을 관장하는 모니터 모드, 프로그램을 생성시키고, 수정, 편집 기능을 담당하는 에디팅 모드, 작성된 프로그램을 수행시키는 수행모드, 외부와의 통신을 위해 마련된 I/O모드, 그리고 교시모드로 구성되어 있다. 개발된 언어는 자체 설계한 로보트제어기 상에서 그의 성능을 입증하였다.

Abstract - In this paper, the design method, design techniques and structure of a language for a SCARA type industrial robot, are presented. The proposed new language is modular and expandable using the C programming language and the 8086 assembly language. It is composed of monitor mode which controls the main flow of the programs, editing mode which generates, corrects and edits the programs, execution mode which executes the generated programs, I / O mode which interacts with the external devices, and teach mode. The developed language is implemented on the robot controller to verify its performance.

1. 서 론

로보트 프로그래밍 언어(robot programming language)는 로보트의 종류에 따라 서로 다르며 그언어들은 마이크로 컴퓨터 단계(micro-computer level)에서 인간지능 단계(human intelligence level)까지 여러 단계가 있다. 그림 1은 여러가지 로보트 언어의 분류를 나타낸다.[1-,11,12,13,14] 일반적으로 1960년대의 산업용 로보트는 교과서적인 프로그래밍 언어(textual programming language)를 사용하지 않고 단순한 교시(teach)와 반복(repeat)의 방법에 의한 도장(painting)이나 스폿트 용접(spot welding) 등에 사용되어 왔다.

그러나 1960년대 후반부터는 교과서적인 프로그래밍 언어를 사용하게 되었는데 그 이유는 팰레타이징(palletizing) 같은 경우는 교시(teach)하는 것보다 계산하는 것이 훨씬 편리하며, 정확하고, 오

*正 會 員 : 서울大 工大 制御計測工學科 教授 · 工博
**正 會 員 : 서울大 工大 制御計測工學科 助教授 · 工博
§ 正 會 員 : 서울大 大學院 制御計測工學科 博士課程
接受日字 : 1987年 9月 23日
1次修正 : 1988年 4月 4日

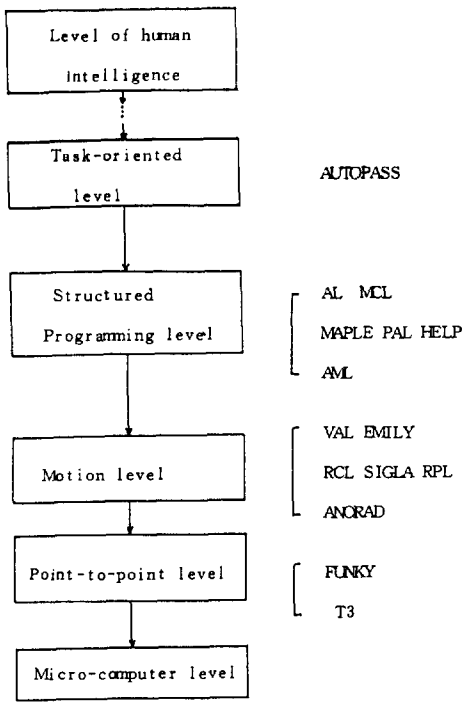

그림 1 로보트 언어의 분류

Fig. 1 Classification of robot language.

프라인 프로그래밍 언어(off-line programming language)의 필요성이 대두되었고, 센서(sensor) 정보를 이용할 수 있다는 장점이 있기 때문이다. 이것은 MIT의 Hand-eye프로젝트에서 최초로 시도하였다. 그 이후 1973년 미국의 Stanford 인공지능연구소에서 로보트의 작업 수행 능력 보다 로보트의 이론적인 한계사항을 발견하기 위해서 WAVE란 언어를 개발하였고, 1974년에는 ALGOL을 기초로 하여 여러개의 로보트를 평행으로 두고 공동 작업을 제어하기 위한 AL을 발표하였다. 이것은 조립작업용으로 만든 최초의 언어이다. 그이후 IBM의 T. J. Watson연구소에서 직각 좌표형 매니퓰레이터(Cartesian manipulator)를 제어하기 위해서 EMILY, ML, AUTOPASS, AML등의 언어를 발표하였다. 또한 1979년에는 Unimation사에서 최초로 상업화된 로보트 프로그래밍 언어인 VAL을 개발했는데 이것은 Stanford의 AL과흡사하며베이직(Basic)언어를 기초로하여 만든 것이다. 현재는 좀 더 확장된 VAL-II가 발표되었다.[16],[17] 그밖에 Cincinnati Milacron의 T_3, McDonnell Douglas의 MCL,[4] Automation사의RAIL, General Electric사의 HELP 등이 발표되었다.[15]

일반적으로 로보트 프로그래밍 언어를 로보트 몸체와 독립적으로 분리하여 생각할 수 없다. 이에 따라 각양 각색의 로보트언어가 개발되게 되었지만, 현존하는 로보트 프로그래밍 언어에 대한 구체적인 자료는 공개되고 있지 않다. 본 논문에서는 앞서 소개한 VAL언어시스템에서 파서(parser), 에디터(editor), 자료구조(data structure)등을 참조하였으며,[4],[9] 인터프리터(interpreter)구조를 채택하였고, SCARA형 로보트에 보다 효과적이라고 생각되는 명령어들을 정의하고, 그에 따른 실행 프로그램을 작성하여 자체 설계한 로보트 제어기와 함께 SCARA형 로보트의 제어에 이용하였다. 본문에서는 2장에서 로보트언어의 구성 및 개발, 3장에서 실험및 결과 검토, 그리고 4장에서 결론에 대하여 서술한다.

2. 로보트 언어의 구성 및 개발

2.1 로보트 언어의 일반적인 구성

일반적으로 로보트 언어는 모니터 모드(Monitor Mode), 에디팅 모드(Editing Mode), 수행모드(Execution Mode)등이 있다. 이 외에도 교시모드(Teaching Mode), 입출력 모드등이 있고, 공통적으로 사용하는 알고리즘(Algroithm)이 있다. 이것의 구성을 간단한 블록선도로 나타내면 그림 2와 같다.

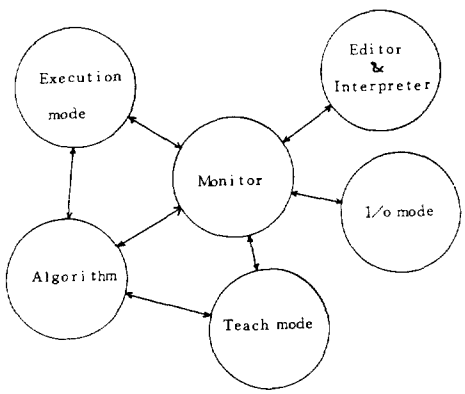

그림 2 로보트 언어의 구성도

Fig. 2 Block diagram of general robot language configuration.

모니터 모드는 전체 시스템의 주 흐름이 되며 제어기 운전의 시작과 완료과정을 관장한다. 또한 다른 모드를 관할한다. 에디팅 모드는 매니퓰레이션 프로그램을 새로이 만들 뿐만 아니라 이미 만들어

놓은 프로그램을 수정하여 편집하는 기능과 동시에 프로그램 생성 및 편집시에 작성된 프로그램이 명령어 형식 및 문법구조(syntax)에 맞는 것인가를 점검하는 파싱(parsing)도 수행하게 된다. 물론 이에디팅 모드에 인터프리터(interpreter)가 선택되었을 경우에는 중간 코드(intermediate code)까지 작성된다. 수행 모드는 사용자가 에디팅 모드에서 작성한 프로그램을 한스텝씩 실행시키는 디버깅 기능을 수행한다. 교시모드는 교시상자(teaching pendant)를 이용한 제어에 필요한 프로그램이며, 입출력 모드는 외부의 기계나 장치등을 연결해서 사용할 수 있게 한 부분이고, 알고리즘은 Kinematics, 역 Kinematics 및 경로계획(path planning)등을 총칭한다.

2.2 개발된 로보트 언어의 구성[9]

본 논문에서 제안된 로보트 언어는 앞에서 설명한 일반적인 로보트 언어의 구성과 같다. 여기서 에디팅 모드의 경우는 라인 에디터(line editor)방식과 인터프리터 방식을 채택하였다. 표 1은 작성된 프로그램의 기능과 그 내용을 나타내며, 프로그램은 C-언어와 8086어셈블리 언어로 작성되었다. 구성된 로보트 언어의 각 모드에 대한 계통도중 지면관계상 모니터 모드와 에디팅 모드에 관한 내용만을 첨가한다. 그림 3은 개발된 로보트 언어의 각 모드중 모니터모드의 계통도이며, 그림 4와 그림 5는 에디팅 모드의 계통도이다.

표 1 작성된 프로그램의 내용

Table 1 The contents of subroutines.

프로그램이름	프로그램내용
MAIN	모니터 모드 프로그램
EDITOR	에디터 모드 프로그램
INTER-COM	중간코드 작성 프로그램
EXEC	수행 모드 프로그램
RUN	명령어 수행 프로그램
SERV	명령어 서비스 프로그램
MLIB-ASM	입출력 프로그램(어셈블리 언어)
MLIB-C	알고리즘 및 산술적 계산 프로그램
MCOM-10	모니터 모드 명령어 수행 프로그램 I
MCOM-20	모니터 모드 명령어 수행 프로그램 II 및 데이타 초기화
MCOM-30	데모(Demo) 프로그램 및 경로계획 교시모드와 인터럽트(Interrupt) 프로그램

2.3 개발된 로보트 언어의 명령어

일반적으로 로보트 언어라 하면 크게 두 가지로

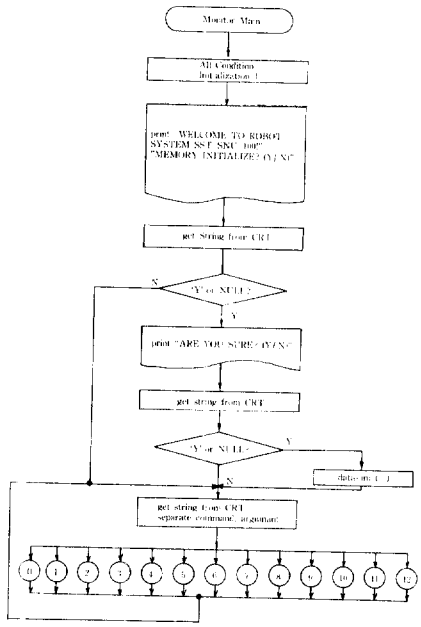

그림 3 모니터 모우드의 계통도 1

Fig. 3 Flow chart of main monitor mode.

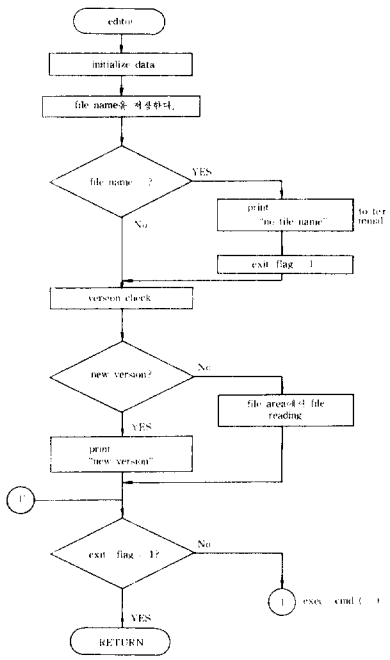

그림 4 에디팅 모우드의 계통도 1

Fig. 4 Flow chart 1 of editing mode.

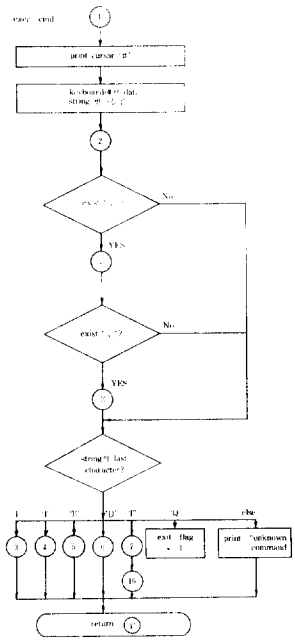

그림 5 에디팅 모우드 계통도 2
Fig. 5 Flow chart 2 of editing mode.

구분할 수 있다. 즉 언어 자체에서 이용하는 모드 명령어와 로보트의 작업을 규정하는데 이용하는 매니퓰레이션 명령어로 나눌 수 있다.

2.3.1 모드 명령어

모드 명령어는 다시 모니터 모드 명령어와 에디터 모두 명령어로 구분된다. 모니터 모드 명령어는 모두 14개이며 그 종류는 아래와 같다.

INITIAL, HERE, TEACH, LDIR, LSET, LLIMIT, ULIMIT, LDELETE, ZERORETURN, FDIR, FDEL, EDIT, RUN, SPEED.

다음 에디터 모드 명령어는 모두 7개이며 그 종류는 다음과 같다.

D(delete), L(list), I(insert), Q(quit), E(end), T(teach), TS(teach straight line).

2.3.2 매니퓰레이션 명령어[3), 4)]

실제 로보트 몸체의 운동을 제어하는 매니퓰레이션 명령어들은 로보트의 종류에 따라 서로 다르나 일반적으로 그 기능을 분류하면 다음과 같다.

i) 로보트의 손을 제어하는 명령어
ii) 로보트 팔의 운동을 제어하는 명령어
iii) 외부 출력을 제어하는 명령어
iv) 프로그램을 제어하는 명령어
v) 기타의 명령어

매니퓰레이션 명령어들은 에디터에서 입력될 때 그 문법구조를 조사하여 틀린점이 없을 경우 수행시의 편리를 위하여 중간코드로 바꾸어진다. 여기서 문법구조라는 것은 각각의 명령어들에 따라 미리 정해진 구조를 의미한다. 예를들면 FOR-TO-STEP의 명령어 같은 경우 그 문법구조는 다음과 같다.

$$\text{FOR}\langle\text{변수 1}\rangle = \left\{\begin{matrix}\langle\text{변수 2}\rangle\\ \langle\text{정 수}\rangle\end{matrix}\right\}\text{TO}\left\{\begin{matrix}\langle\text{변수 3}\rangle\\ \langle\text{정 수}\rangle\end{matrix}\right\}$$
$$\text{STEP}\left[\left\{\begin{matrix}\langle\text{변수 4}\rangle\\ \langle\text{정 수}\rangle\end{matrix}\right\}\right]$$

여기서 문법구조를 조사한다는 것은 위의 형태로 입력 데이타가 들어 왔는지의 여부를 조사하는 것을 의미하며 [ ]로 표시한 부분은 생략가능한 부분이다. 또 〈변수 2〉, 〈변수 3〉, 〈변수 4〉는 이 프로그램 라인이 수행되기 이전에 그 값이 선언되어야 한다. 그러나 〈변수 1〉은 그럴 필요가 없다. FOR명령어를 중간코드로 바꾸는 구조는 아래의 그림 6 과 같다.

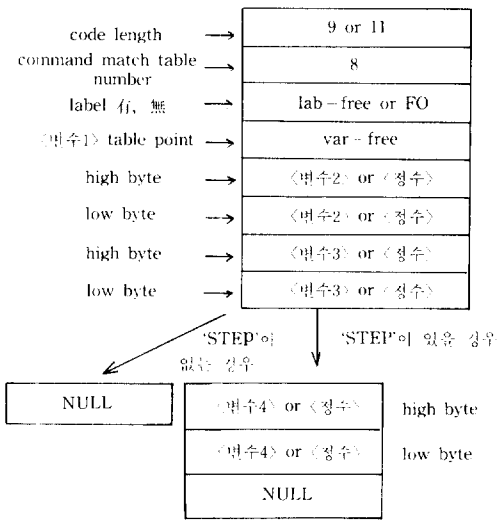

그림 6 FOR명령어의 중간코드 구조
Fig. 6 Record format for FOR command.

제일 윗단의 9 또는 11은 코드의 길이를 나타내며 여기서는 두 가지의 값을 가질 수 있다. 즉 'STEP'이 없을 경우에는 9이고 'STEP'이 있을 경우에는 11이 된다. 그 밑에는 command match table

number가 있는데 이것은 각 명령어에 따라 미리 정해진 토큰(token) 값을 의미한다. 그 토큰값은 표 2에 정리되어 있다. 여기서는 FOR 명령어이므로 8이 된다. 그 아래에는 라벨(label)의 유, 무를나타낸다. 즉 라벨이 없을 경우에는 FO(Hex)의 값을 저장하고 라벨이 있을 경우에는 그 라벨이 저장된 테이블의 위치를 나타내는 'lab-free'의 값을 기록해 놓는다. 이렇게 하면 프로그램 수행시에 라벨의 위치를 쉽게 찾을 수 있어 시간을 단축한다. 'lab-free'의 값은 한 프로그램이 가질 수 있는 라벨의 갯수만큼 가질 수 있는데 본 프로그램에서는 0에서 99까지 가질 수 있다. 그 다음은 〈변수1〉에 관한 정보를 기록하는데〈변수1〉은 미리 그 값이 정해진 변수일 필요가 없으므로 만일〈변수1〉이 변수 테이블에 등록되지 않은 변수일 경우에는 변수 테이블에 등록하고 난 뒤에〈변수1〉 테이블 포인터에는 'var-free' 값을 기록해 놓는다. 'var-free'의값은 'lab-free'와 마찬가지로 본 프로그램에서는 0에서 99까지 가질 수 있게 하였다. 그 다음은 〈변수2〉 또는 〈정수〉에 관한 정보가 입력되는데 〈변수2〉는 〈변수1〉과 같이 'var-free' 값이 0에서99까지 가질 수 있으나 〈정수〉값과 구별하기 위하여 65436에서 65535(Decimal) 값으로 변환시키고 〈정수〉 값은 0에서 65436(Decimal)까지만 유효한 것으로 정의한다.

데이타값이 16비트(bit)이므로 상위 바이트를 먼저 저장하고 그 다음 하위 바이트를 저장한다. 마찬가지로 〈변수3〉 또는 〈정수〉에 관한 정보도 상위, 하위 바이트로 나누어서 저장한다. 'FOR' 명령어에서 'STEP'은 생략가능하므로 'STEP'이없을 경우에는 중간코드 마지막에 NULL을 저장하고 'STEP'이 있을 경우에는 〈변수4〉와 〈정수〉에 관한 정보를 상위, 하위 바이트로 나누어서 저장한다. 그리고 마지막에는 역시 NULL을 저장한다. 여기서 본 프로그램상 주의할 사항은 변수는 숫자를 제외한 문자 5개까지이며 숫자와 문자를 같이 사용한 변수는 허용되지 않고, 정수는 0에서 65435(Decimal)까지만 유효하다는 것이다.

3. 실험 결과 및 검토

로보트 프로그래밍 언어를 연구하는데 사용된 장비는 그림 7에서 나타낸 바와 같이 HP64000LDS (Local Development System)와 IBM-PC, 8086, 8085Emulator등인데 실험의 개요는 64000LDS 에서 프로그램을 작성하여 목적 프로그램(object program)을 8086Emulator에 실어서 프로그램을 수행시켜보고 수정하는 과정을 반복하였다.

일반적으로 SCARA형 로보트는 1,2축과 그 밖의 축을 기계적으로 분리시키는 것이 가능하기 때문에 보다 정밀한 위치제어가 가능한 조립 작업등에 유용한데, 개발한 언어에서는 이러한 SCARA형 로보트의 특성을 포함한 'UP', 'DOWN', 'GAIN'(표 2 참조) 명령어를 비롯하여 TOSHIBA의 SACRA

표 2 명령어들의 토큰값 및 중간코드 발생프로그램과 수행프로그램 명칭

Table 2 Command match table number & code generation program name & execution name.

Command	Match Number	Code Generate	Execution Program
CLOSE	1	inter-cls ( )	cls-ser ( )
CLOSEI	2	inter-cli ( )	cli-ser ( )
DELAY	4	inter-dly ( )	dly-ser ( )
DOWN	6	inter-dwn( )	dwn-ser ( )
FOR	8	inter-for ( )	for-ser ( )
GAIN	10	inter-gan ( )	gan-ser ( )
GETF	11	inter-get ( )	get-ser ( )
GOSUB	12	inter-gos ( )	gos-ser ( )
GOTO	13	inter-got ( )	got-ser ( )
GOTO(A)	14	inter-goa ( )	goa-ser ( )
GOTO(I)	15	inter-goi ( )	goi-ser ( )
IF	17	inter-if ( )	if-ser ( )
IFSIG	18	inter-ifs ( )	ifs-ser ( )
MOVE	20	inter-mov( )	mov-ser ( )
MOVEF	21	inter-mvf ( )	mvf-ser ( )
MOVEP	22	inter-mvp ( )	mvp-ser ( )
MOVES	23	inter-mvs ( )	mvs-ser ( )
NEXT	25	inter-nxt ( )	nxt-ser ( )
OPEN		inter-opn ( )	opn-ser ( )
OPENI		inter-opi ( )	opi-ser ( )
PAUSE		inter-pau ( )	pau-ser ( )
READY		inter-rdy ( )	rdy-ser ( )
REMARK		inter-ren ( )	rem-ser ( )
RETURN		inter-rtu ( )	rtu-ser ( )
SET		inter-set ( )	set-ser ( )
SETI	37	inter-sei ( )	sei-ser ( )
SHIFT	38	inter-sht ( )	sht-ser ( )
SIGNAL	39	inter-sig ( )	sig-ser ( )
SPEED	40	inter-spd ( )	spd-ser ( )
STOP	41	inter-stp ( )	stp-ser ( )
UP	42	inter-up ( )	up-ser ( )
VPNT	43	inter-vp ( )	vp-ser ( )

표 3 5가지 로보트 언어의 실제적 특성 비교
Table 3 Comparison of the actual features on the 9 robot languages.

	VAL	AL	PAL	Autopass	New Language
Language Basis	Assembler	Algol	Transform Basis	PL/1	C and Assembler
Language Type	Interpreter	Compiler and Interpreter	Interpreter	Interpreter	Interpreter
Control Structure	Go to, if / then, ifsig / then	Begin / end While / Do if / then / else	For / to / do, begin / end	If / then / else, While / do, begin / end	Go to, Go to(i) If / then Goto(a)
Callable routines	Gosub return	Procedures functions macros(P)	None	Procedures	Gosub return
Data Type Available	Frame	Scalar, Vector rot, trans, frame	Matrix Array Scalar, stack	Integer, real array, World model	Integer real joint angle
Simple Motion	Move, Movet, Appro.	Move with constraint clauses	Mov	Move to move	Move, Movep
Stright Line Motion	Moves Appros	None	None	Push Slide	Moves
Splined through Several points Continuous Path motion	Yes	Yes "via" gives Intermediate Points	Yes	Implicit	None
Coordinate Transformation Command	Shif Inverse frame	Affix unfix	+(mult) -(invert and mult)	None	None
Interaction with external Devices	Ignore, Ifsig, react, signal	Signal Wait	None	Switch load unload	Ifsig, Goto(i)
Higher Level Commands	None	Compliant motion if force set to zero	None	Place / on Drive / on name / assembly	None

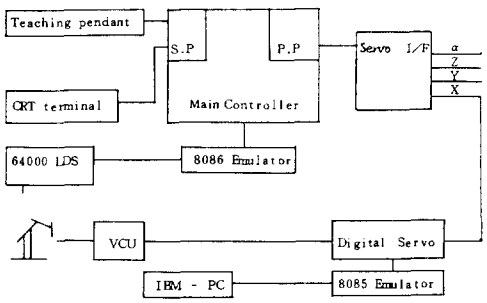

그림 7 로보트 제어 개발 시스템 블럭선도
Fig. 7 Block diagram of robot control development system.

형 로보트 전용언어인 SCOL-I이 지닌 각종 기능을 발휘하도록 설계하였다.

특히 SCOL-I에서는 찾아볼 수 없는 교시상자에 의한 프로그램 생성기능을 첨가하였고, 운동지점의 이름도 알파벳트와 숫자를 병행하여 자유롭게 사용할 수 있도록 하였다. 개발된 프로그램을 직접 확인하였다.

표 3에서는 언어의 기저, 언어의 형태, 제어구조, 서부루틴 기능, 이용가능한 데이타형태, 단순동작, 직선 동작, 여러 개의 경유점 통과 동작, 좌표축변환명령, 외부기기와의 상호작용, 높은 단계의 명령에 관하여 선택한 4개의 언어와 새로 구성한 언어를 비교하였다. 개발된 언어는 C-언어와 8086어셈블리어를 병행하여 사용하였으며, 제어 구조의 경우 다른 언어에 비하여 부족하지 않고, 데이타 구조, 운동 명령 및 여러 가지 기능을 갖추고 있으며, 특히 외부기기와의 상호작용을 도와주는 명령어들을 많이 갖추고 있다는 점이 특징이다.

4. 결 론

지금까지 설명한 로보트 언어는 운영체제(operating system)을 작성하기 편리한 C-언어로 설계되었으며 입력 출력 부분은 8086어셈블리(assembly)언어로 작성되었다.[5-10] 본 논문에서 설계한 로보트의 언어는 실제 SCARA형 로보트인 TOSHIBA 로보트 몸체에 자체로 설계한 로보트 제어부를연결하여 동작을 시키는 하드웨어 및 소프트웨어의 실제적 구현에 중점을 두었다. 따라서 다른 로보트 언어들에서 고려한 사항들을 다 수록하지 못한 점도 있다. 그러나 일반적으로 SCARA형 로보트가 지니는 특성은 모두 포함하고 있으며, 원시 프로그램을 직접 구성했기 때문에 명령어를 확장하거나 수정을 쉽게 할 수 있으며 약간의 수정을 거치면 다른 기종의 로보트에도 적용시킬 수 있다. 그 다음 본 프로그램은 대부분 C-언어를 사용해서 설계했기 때문에 쉽게 수정할 수 있으며 될 수 있는 한 모든 프로그램들을 세분화하여 계층 구조를 이루게 하였고, 수정 및 편집을 단위 프로그램별로 할 수 있게 하였다. 이러한 방식은 프로그램 기법의 현재 추세이다.

그러나 개발된 로보트 프로그램은 보충해야 할 부분과 확장해야 할 부분이 있다. 즉 각각의 모드에서 새로운 기능의 명령어를 첨가시키는 것이며 또한 자기 진단 기능을 강화해야 한다. 자기 진단 기능은 주위 환경 및 로보트의 상태, 사용자의 상태등 모든 점을 고려하여 발생할 수 있는 상황을 처리할 수 있어야하므로 그 프로그램이 단기간에 나올 수 있는 것이 아니다. 그러나 국내에서도 계속 로보트 언어를 설계 및 공개하고 있으므로 조만간 이에 대한 연구가 체계적으로 될 수 있으리라 생각한다. 또한 알고리즘에서도 좀 더 적합한 그리고 시간을 줄일 수 있는 형태로 개선하고, 새로운 알고리즘을 적용 실험해 보는 것이 필요하다.

본 논문에서 발표하는 로보트 언어는 로보트 언어의 발달 단계로 봤을 때는 운동 단계언어(motion level langage)로 볼 수 있으며 프로그램 분량은 약 9000줄에 62K바이트 정도된다. 앞으로의 과제는 로보트 개발의 추세에 따라 비젼(vision) 시스템을 적용할 수 있게 프로그램을 확장하고, 토오크 및 힘센서(torque/force sensor)를 사용하는데 편리한 구조로 확장하여 그 동작 오차를 줄여 나가야 할 것이다. 이러한 문제점이 있음에도 본 논문에서 제시한 프로그램은 로보트 언어 설계 및 개발에 많은 도움을 줄 수 있을 것이다.

"본 논문은 1987년도 과학재단 목적기초 연구비 지원에 의하여 연구되었음."

참 고 문 헌

1) Larry Heath, "Fundamentals of Robotics Theory and Applications, Reston Publishing Company, pp.145-165, 1985.

2) W.A.Gruver et al., "Evaluation of Commercially Available Robot Programming Languages," Conf. Proceedings of 13th ISIR, 1983.

3) John J. Donovan, "System Programming," McGraw-Hill, pp.279-314, 1982.

4) Unimation Inc., "User's Guide to VAL," Ver.12, June, 1980.

5) Kernighan, "The C Programming Language," Prentice- Hall, 1978.

6) B.O.Wood, "MCL, the Manufacturing Control Language," Conf. Preceedings of 13th ISIR, 1983.

7) S. Bonner et al., "A Comparative Study of Robot Language," IEEE, Computer, Dec., 1982.

8) 김대원, "디버깅 특성을 고려한 로보트 매니퓰레이터 언어에 관한 연구," 서울대학교 공학석사 학위 논문, 1985.

9) 김경환, "VAL-컴파일러를 이용한 로보트 프로그래밍 시스템의 설계," 서울대학교 공학석사 학위 논문, 1986.

10) 打越浩宰, "C プログラムブッグ, アスキ-出版局, pp82-112, 1984.

11) R.P. Paul, WAVE:A Model-Based Language for Manipulator Control," Technical Paper Mr. 76-615, SME, 1976.

12) R.Finkel et al., "An Overview of AL, A Programming System for Automation," Preceedings of the 4th International Joint Conference on AI, 1975.

13) L.I. Liebeman et al., "AUTOPASS:An Automatic Programming System for Computer Controlled Mechanical Assembly," IBM J. of Research and Development, vol.21, no.4, July, 1977.

14) R.H.Taylor et al., "AML:A Manufacturing Language," Int.J.of Robotics Research, vol.1, no.3,Fall, 1982.

15) William Thomas Park, "The SRI Robot Programming System," Proceedings of the 13th ISIR, 1983.

16) B.E.Shimano et al., "VAL-II:A Robot Programming Languages and Control System," Robotics Research, pp.917-940, MIT Press, 1984.

17) B.E.Shimano, "VAL:A Versatile Robot Programming and Control," Proceedings of COMPAC 79, 1979.

18) Daniel H.Marcellus, "Systems Programming for Small Computer," Prentice-Hall, pp.231-260, 1984.