'93 한국자동제어학술회의논문집 (1993. 10. 20~22)

실시간 다중처리 운영체제를 이용한 로보트 제어기의 설계

° 최성락, 정광조
한국기계연구원 자동제어실

Design of a Robot Controller using Realtime-Multitasking OS

° Choy Sung Lark, Jung Gwang Jo
Korea Institute of Machinery and Metals, Automatic Control Lab.

Abstract

In this paper, a robot controller that has a real time-multitasking OS (Operating System) is developed. It can do given jobs in realtime, so its effectiveness is increased. The controller has several CPU boards, and it is needed to communicate among these boards. For that reason, it is adopted VME bus system and VMEexec OS that can process multi-process in realtime. Multiprocess includes robot language edit process, vision process, low level motion control process, and teach process in higher layer. And dynamics, kinematics, and inverse kinematics that require realtime calculation are included in lower layer.

1. 서론

본 논문에서는 로보트의 각 축에 대한 motor, vision sensor에 대한 정보등을 총괄, 관리하고 foreground job, background job을 schedul 하면서, 사용자 인터페이스를 제공케하는 main controller에 대하여 설명하고자 한다.

main controller는 사용자가 interface를 통해서 모든 기능을 제어할 수 있어야 한다. 이를 위해서 사용자 인터페이스를 제공한다. 사용자 인터페이스의 구성은 초보적인 사용자 라도 쉽게 작동시킬 수 있도록 하여야 한다. 또한 사용자의 오입력(吳入力)을 최소화할 수 있는 구조로 설계하여야 한 다. 따라서 pull-down menu driven방식을 사용하였다. 자세한 설명은 3장에서 한다. 또한 main controller는 가능한 한 모든 계산을 실시간 내에 처리해서 로보트를 구동해야 한다. 로보 트를 움직이는데 필요한 계산에는 sine, cosine등 삼각함수 계산이 많이 사용되는데, 가능한 한 이를 줄이고, 또한 시스 템 자체에서 실시간 처리를 지원할 수 있도록 실시간 처리용 운영체제인 VMEexec을 사용하였다. 이 운영체제는 UNIX를 기초로 하고 있기 때문에 실시간 처리는 물론, 다중처리도 제 공하고 있어서 다양한 센서에서 나오는 정보를 실시간으로 다중처리하는 데에 적합한 운영체제이다. Hardware도 위의 S/W를 운용하고 다중처리를 원활하게 지원할 수 있는 VME bus system을 사용하였다. 각 sensor, 즉 motor에 대한 encoder, vision processing에 대한 camera, gripper에 대한 photo interruptor 등에서 오는 정보를 전용으로 처리할 수 있도록 전용 CPU board를 둘 수 가 있어서 실시간 처리를 가능케 하고 있다. CPU board는 총3개로, 사용자 인터페이스를 제공하고 target CPU board에서 오는 정보를 총괄 처리하는 main CPU board와, 6개의 motor를 control 하고 kinematics, inverse kinematics, dynamics 등을 계산하기 위해서 가장 빠른 속도를 필요로 하는 CPU board, vision sensor에서 오는 정보를 처리하는 CPU board등이다. 또한 motor의 속도제어를 위해서 2개의 motor마다 매우 빠른 계산처리 능력을 가진 DSP(Digital Signal Process)chip을 사용한 전용 motor contoller를 두었다.

2. Hardware의 구성

6축을 기반으로 하는 robot을 control하기 위해서 main controller는 VME bus system으로 구성하였다. 다음 그림은 전체 구성도를 나타낸 것이다.

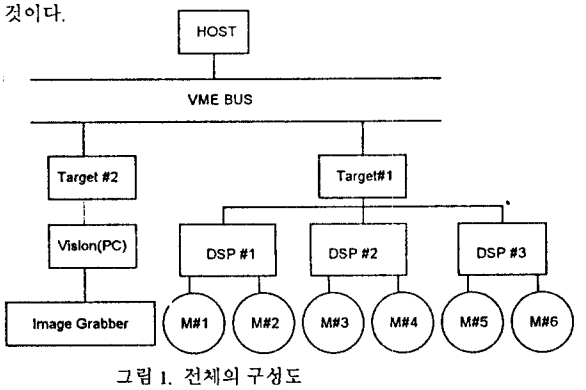

그림 1. 전체의 구성도

그림 1에서, 시스템의 사양은 다음과 같다.

표 1. 시스템의 사양

	Host	Target #1	Target #2
Module	MVME147SA-1	MVME162-20	MVME147S-1
CPU	MC68030 25MHz	MC68040 25MHz	MC68030 25MHz
FPU	MC68882 25MHz	자체 내장	MC68882 25MHz
RAM	8Mb	4Mb	4Mb
VME Address	0-0x7FFFFFFF	0x200000000-0x23FFFFFFF	0xC0000000-0xFFFFFFFF

Host는 program개발시에는 UNIX를 OS로 하며, 개발이 완료 된 후에는 UNIX와는 별도로 VMEexec에서 만들어진 kernel로 동작하게 된다. 이는 program개발시 file을 관리하고 개발하기 쉽게 하기 위해서 disk-based OS인 UNIX를 사용하기 때문이다. Host board는 UNIX 환경에서 동작할 수 있도록 메모리 용량을 충분하게 하였고, 각 보드마다 FPU(Floating-point Process Unit)를 장착하여 복잡한 수학식을 풀기 위한 CPU의 짐을 덜 수 있도록 했다. 표 1에서의 VME address는 addressable한 메모리 총 4Gb중에서 사용하기 쉽고 또한 각 보드의 local address끼리 충돌하는 일이 없도록 공간을 충분히 두었다.

Target #1과 DSP board는 모두 VME bus상에 같이 물려있으며 DSP board 한장당 2개의 모터를 제어할 수 있다. DSP board의 자세한 사양은 생략한다.

Target #2와 vision processor인 PC(Personal Computer)와는 비동기 통신인 RS232C를 사용하여 정보를 주고 받는다. Target #2 CPU board인 MVME147S-1은 RS-232C port를 4개 가지고 있다. 이중 port 0는 debugging용 terminal로 쓰이고 port 1을 통하여 PC와 통신을 한다. 이에 대한 설명은 3장에서 설명한다.

3. Software의 설계

3.1 사용자 인터페이스 (User Interface)

User가 robot를 control하고 원하는 작업을 하기 위해서는 user와 controller간의 S/W적인 연결이 있어야 한다. 이 역할을 하는 것이 사용자 인터페이스인데, 이의 구성은 다음과 같다.

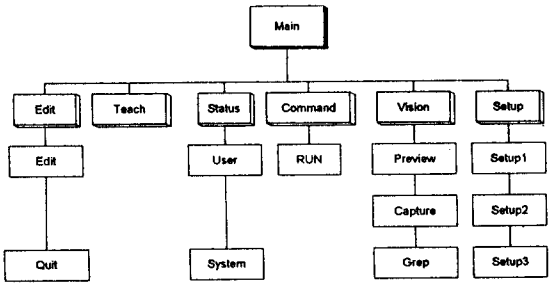

그림 2. 사용자 인터페이스의 구조

각 부분을 자세히 설명하면 다음과 같다.

1) Edit menu

① Edit : 사용자가 robot primitive를 사용하여 program을 작성할 수 있는 메뉴이다. Robot primitive란 program언어의 함수와 같은 개념으로써, 예를 들어 MOVJ(1,30)이란 primitive를 사용 했다면 이는 첫번째 joint를 30도 움직이라는 명령인 것이다.

② Quit : Program개발시에 필요한 메뉴로써, 개발이 끝나면 없어지는 메뉴이다. Program실행을 끝낸다.

2) Teach menu

로보트를 교시하는 메뉴이다. 이 메뉴에서 로보트가 가고자 하는 위치를 교시한다. 사용자는 화면에 나타나는 key를 사용, joint 좌표계, 또는 world좌표계등을 이용하여 원하는 위치로 end effector를 움직인다. 원하는 위치에 end-effector가 도달하면 그 점에 명칭을 부여하여 이를 program에 이용할 수 있다. 또한 화면상에는 end-effector의 현재 position, orcintation과 각 joint값이 display되어 교시를 더욱 쉽게 할 수 있도록 도와 준다. 다음은 교시할 때의 화면이다.

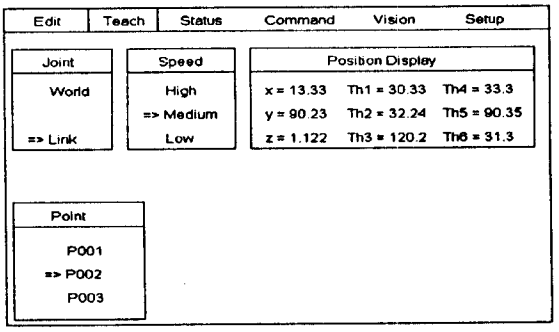

그림 3. Teach시의 화면 구성

위의 화면에서 position display부분은 로보트가 움직일때마다 자동으로 update된다. 이를 위해서는 task를 background job으로 처리하여 kinematics, inverse kinematics를 풀어 display해 주어야 한다. 우선 각 joint를 움직여서 교시를 할것인지, 아니면 end-effector를 중심으로 교시를 할 것인지 'j' key로 선택하고, 교시속도를 's' key로 선택, 교시를 시작한다. 원하는 위치에서 교시가 됐다면, 그 위치를 저장하기 위하여 'p' key로 위치를 저장한다. 이로써 교시를 마치게 되는데, teach pendant를 따로 제작하지 않았으므로, terminal의 ketboard를 이용한다. 이에 사용되는 'q',' 'w', 'e', 'r', 't', 'y' key들은 각 joint를 +방향으로 움직이게 하고, 'z', 'x', 'c', 'v', 'b', 'n' key들은 각 joint를 -방향으로 움직이게 한다.

3) Status menu

이 메뉴에서는 현 시스템의 상태를 볼 수 있는 메뉴이다. 로보트의 제어에는 직접적인 관계는 없으나 관리상의 필요 성으로 이 메뉴를 두었다.

① User : 사용자의 이름등이 display된다.

② System : 현재 사용가능한 메모리, 지정할 수 있는 point의 갯수등이 display된다.

4) Command menu

① Run : edit에서 작성한 program을 실행한다. 세부모드가 있어서 1 step씩 수행 할 수도 있고 전체 program을 한번에 수행할 수도 있다. Program을 수행하면서 수행되는 라인이 화면에 표시된다. 또한 현재 end-effector의 위치와 각 joint의 각도도 표시되어 debugging도 간편하게 할 수 있다.

5) Vision menu

Camera에서 잡은 vision image를 처리하는 명령이다. 물론 이 메뉴는 vision process가 하는일중 일부를 옮겨온 것이다.

6) Setup menu

기본적인 setting을 한다. 예를 들어 사용자 좌표계를 정의하 거나, world좌표계를 변환한다거나, end-effector를 transepose할 수 있는 menu이다. End-effector를 transepose 한다는 것은 end-effector에 tool을 장착하였을 때, end-effector가 가리키는 위치가 맞지 않게 된다. 따라서 새로운 tool의 끝점에 end-effector의 점을 맞추어 주어야

한다. 이를 위해서 end-effector를 transe -pose시켜주는 것이다.

3.2 Primitives

Robot language를 구성하기 위해서는 다른 computer program language와 마찬가지로 예약어(Reserved Word)와 함수(Function) 가 있어야 하는데, 이를 primitive라 한다.

1) 예약어

예약어란 program실행에는 직접적인 영향은 끼치지 않으면 서 program실행에 도움을 주는 명령들이다. 본 논문에서 정의한 예약어는 다음 표와 같다.

표 2 예약어(Reserved Word)의 종류

break	do	else	elseif
end	for	goto	if
switch	case	while	with

예를 들어 loop를 100번 돌린다면 함수를 백번 써주는 것이 아니라 위의 예약어를 사용, do, for등의 문장을 사용하여 loop 를 돌릴 수 있다. 각각의 사용법은 다음과 같다.

① do ... while

do가 사용되면 그 block의 끝에 반드시 while이 사용되어야 한다. Loop문을 이루는 것으로써 loop를 빠져나가는 조건은 block 맨끝의 while에서 check가 된다.

[예문]

```
i=0
do {
        movj(1,0.04)
        i = i + 1
} while(i!=100)
```

위의 예문은 movj를 100번 수행한다.

② for ... end

①과 비슷한 loop를 형성한다. 단지 차이점은 loop를 빠져나가는 조건은 block 맨앞에서 check가 된다는 것이다.

[예문]

```
for ( i=1 ; i<100 ; i++ )
        movj(1,0.04)
end
```

위의 예문은 movj를 99번 수행한다.

③ if ... end ... elseif ... end ... else ... end

조건문이다. 예문을 들어 설명한다.

[예문]

```
if i=0
        movp(P001)
end
elseif i>30
        movc(P002, P003)
end
else
        movj(3,30)
end
```

만일 i=0이면 movp(P001)이 수행되고 i가 30이하이면 movj(3,30) 이 수행된다. 그리고 그 외엔 movc(P002, P003)이 수행된다.

④ switch ... case ... end

③과 비슷하며 ③으로써도 이 기능을 대체할 수 있으나 훨씬 간단하게 해결된다. 예를 들면 다음과 같다.

[예문]

```
switch(i)
        case 0:
                movj(0,30)
                end
        case 1:
                movj(1,30)
                end
        case 2:
                movj(2,30)
                end
end
```

i가 0,1,2에 따라 그에 해당하는 joint를 움직이는 명령을 예문으로 보이고 있다.

⑤ goto

무조건적인 분기(branch)명령이다. 이를 사용하면 무한루프 의 반복문을 쉽게 만들수 있으나 권장사항은 아니다. goto다 음엔 어디로 분기할 것인지에 대한 label이 반드시 서술되야 한다. Label은 시작이 반드시 %로 시작되어야 한다.

[예문]

```
%LABEL#1
        movj(3,30)
        if i<30
                movp(P003)
        end
        else
                goto LABEL#1
        end
```

i가 30보다 작다면 movp를 수행하고, 그렇지 않다면 LABEL#1 으로 분기한다.

⑥ with

예약어중 실제 motion function과 같이 쓰이는 예약어이다. 혼자로는 사용할 수 없다. 즉, 속도 함수인 speed와 같이 사용하는데, 이에 대한 것은 '2) 함수'에서 설명한다.

[예문]

```
speed = 100
movj(3,30)
if i>400
        movj(3,30) with speed = 50
end
else
        movj(3,30)
end
```

위에서는 모든 movj가 스피드를 100으로 하여 움직이지만 i가 400보다 클때에만 50으로 움직인다.

2) 함수(Function)

함수란 실제 robot motion에 영향을 끼치는 것들로서, 다음의 표는 함수들을 나열한 것이다.

표 3. 함수(Function)의 종류

	첫번째 인자	두번째 인자	설 명
movj	Joint #	Angle	주어진 관절을 주어진 각도만큼 move
movp	point		현재위치에서 주어진 위치로 이동
movl	point		현재위치에서 주어진 위치로 직선 이동
movc	point#1	point#2	현재위치에서 point#1을지나 point#2로 직선운동
mova	point#1	point#2	3점을 잇는 arc운동

① movj (Move Joint)

movj(joint#, angle)

[Description]

첫번째 인자로 주어진 관절을 두번째 인자로 주어진 각도만큼 움직인다. 다음의 그림은 movj에 대한 block diagram이다.

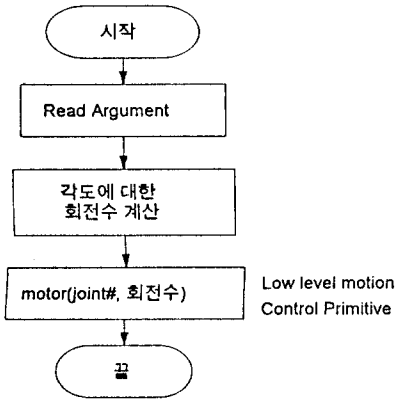

그림 4. MOVJ의 Block Diagram

위의 그림에서 low level motion control primitive란 말이 나오는 데, 이는 모든 motion에 적용되는 primitive로서, 최하위 단계의 primitive이다. 이는 직접 모터를 구동하는 routine을 가리킨다. Error없이 수행되면 0를 return하고 error가 있다면 미리 정의된 error값을 return한다.

② movp (Move Point)

[Format]

movp(loc1)

[Description]

PTP(Point To Point) motion이다. 각 joint의 motor는 주어진 속도(또는 default값)로 현재 위치에서 주어진 위치에 도달할 때까지 motor를 돌린다. MOVP의 순서도는 다음과 같다.

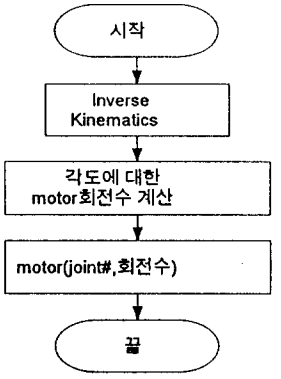

그림 5. MOVP의 Block Diagram

③ movl (Move Line)

[Format]

movl(loc1)

[Description]

②의 movp와 비슷하나 CP motion이란 점이 다르다. 즉, movp는 경로에는 상관없이 무조건적으로 주어진 위치에 도달하지만 movl은 직선이란 경로를 따라 움직이는 motion이다. 구현하는 방법은 주어진 경로를 여러개의 knot로 나눈후 그 knot마다의 각을 구한후 inverse kinematics를 풀어 움직이게 하는 것이다. 이의 block diagram은 다음과 같다.

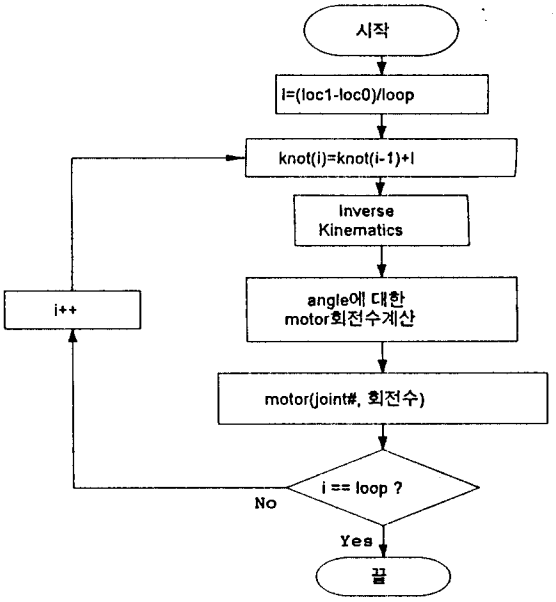

그림 6. MOVL의 Block Diagram

[주의점]

* Loop수는 real time execution이 될 수 있도록 실험을 통해 조절한다.
* 각 joint의 motor동작은 동시에 끝날 수 있도록 한다.

④ movc (Move Continue)

[Format]

movc(loc1, loc2)

[Description]

movc는 ③의 movl을 2개 붙여 놓은 형태와 같다. 즉 현재위치에서 loc1을 거쳐 loc2로 움직이는 motion이다. Block diagram은 다음과 같다.

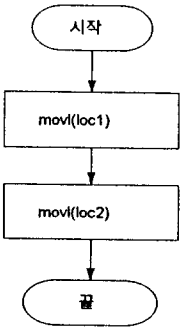

그림 7. MOVC의 Block Diagram

⑤ mova (Move Arc)

[Format]

mova(loc1, loc2)

[Description]

mova함수는 현재위치와 loc1, loc2 3점을 잇는 arc운동을 한다. 현재 위치에서 출발하여 loc1을 지나 loc2로 움직인다. 물론 CP motion이다. 구현하는 방법은 movl과 마찬가지로 원의 방정식을 이용 path를 구한 후 이를 여러 개의 knot로 나누어 그 경로에 해당하는 inverse kinematics를 구하여 motion을 하는 것이다. 복잡한 계산이 많으므로 실시간 제어를 위해서 routine을 간소화 할 필요가 있는 함수이다. 다음 그림은 mova의 block diagram이다.

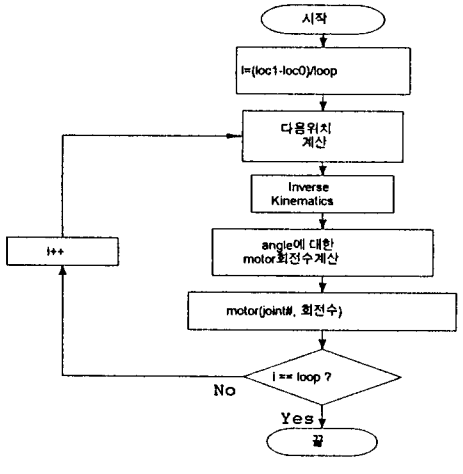

그림 8. MOVA의 Block Diagram

3.3 Lexical Analyzer

Lexical analyzer란, editor나 command mode에서 사용자가 robot primitice를 사용해서 명령을 입력했을때, 내부적으로 이 명령 을 해석하고, 정의된 각 module을 call하고 argument로서 각 data를 넘겨주는 역할을 하는 부분이다. 이를 interpreter라고도 하는데, lexical analyzer는 훨씬 좁은 의미에서 사용된다. 즉, 말그대로 문법만 해석하고 각 token을 넘겨주는 역할만 한다. 이 역할은 정의만 내려주면 그에 해당하는 분석기를 자동으로 만들어 주는 utility를 사용하여 구성하였다. UNIX의 utility에는 lex, yacc이 있는데, 이들이 바로 그런 역할을 담당한다. 분석기는 C source code로 출력을 내보내서 이를 control program에 연결할 수 있게 되어 있다. lex는 lexical analyzer로써, 사용자가 입력한 robot language가 문법에 맞는지를 check하고 error가 있다면 user에게 여부를 알려준다. yacc에서는 lex에서 보낸 token을 모아 필요한 action을 취하게 하는 C source code를 만들어 준다. 다음 그림은 lex와 yacc의 개괄적인 순서도 이다.

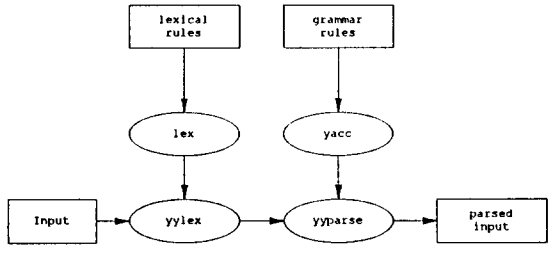

그림 9. lex와 yacc의 순서도

이 lex와 yacc을 사용하면 위의 그림에서와 같이 yylex()란 함수와 yyparse()란 함수를 자동으로 만들어 준다. 물론, 이 함수를 만들기 위한 문법형식과 token정의등은 lexical rule과 grammar rule에서 정의해 주어야 하고 각 token에 대한 action도 coding 해주어야 한다.

3.4 Serial Communication

Vision processor인 PC(Personal Computer)와 통신을 할 때는 PC가 VME bus를 지원하지 않으므로 서로가 공통적으로 가지고 있는 RS-232C port를 통해서 미리 정해진 protocol로서 통신을 한다. 우선 H/W적인 setting으로서, 다음의 표와 같다.

표 4. RS-232C의 H/W setting

이 름	Mode
통신형식	Serial
통신 포트	Port #2
통신 속성*	0
Data Size	8 bit
Stop bit수	1 stop bit
Parity	None Parity
Baud Rate	9600

* 통신 속성은 다음의 표를 참고한다.

표 5. 통신 속성

Name	Bit No.	설명
AS_FLOW	0	0:H/W flow control 1:XON/XOFF flow control
AS_ECHO	1	0:Echo received character 1:No echo
AS_TYPEA	2	0:Use type-ahead buffer 1:Disable type-ahead buffer
AS_DNULL	3	0:Discard received null characters 1:Don't discard received null characters
AS_DTR	4	0:DTR On 1:DTR Off
AS_RTS	5	0:RTS On 1:RTS Off
AS_RAW	6	0:Raw mode 1:device supports SYSTEM V/68 character processing

MVME147S-1은 VME상에서의 vision processor이다. 이 CPU board에서는 background job으로써 PC로의 송수신 routine이 돌고 있어서 request가 있을 때마다 job이 wake-up이 되어 PC와 통신을 한다. 먼저 VME상에서 PC에 position data를 요구하는 message를 보내면, PC는 미리 계산해 놓은 vision data를 format에 맞추어 보내준다.

4. 결 론

본 논문에서는 robot controller의 전체적인 H/W, S/W의 구성을 설계하였고, 사용자 인터페이스, robot language의 기본인 primitive와 예약어, 그리고 serial communication에 대하여 개발하였다. H/W는 CPU board의 추가, 삭제가 용이하고 multitasking의 구현이 일반화 되어 있는 VME bus를 사용하여, 구현하였고, 그에 맞추어 S/W도 OS를 UNIX로 하여 multitasking이 자연스럽게 이루어 질 수 있는 환경을 마련하였다. 여기에, 실시간 제어를 위해서 VMEexec를 사용하여 실시간 다중처리를 할 수 있는 환경을 제공케 하였다. 3장에서 serial communication에 대한 routine은 VMEexec에서 제공하는 함수를 이용하여 만든것으로서, 사용하기에 어려운 단점이 있지만 assembly수준의 정밀한 제어를 할 수 있다는 장점이 있다. 향후에는 실제 robot에 porting함과 동시에 그 실험결과를 통한 세세한 scheduling이 되어야 하며, 가장 최적의 실시간 제어를 위한 환경을 만들어 주어야 한다.

5. 참고 문헌

1. Commer, "Operating System Design, Volume I The XINU Approach", Prentice Hall
2. MVME147S MPU VMEmodule User's Manual, Motorolar Inc.
3. MVME162 Embedded Controller Programmer's Reference Guide, Motorolar Inc.
4. VMEexec User's Manual, Motorola Inc.
5. VMEexec Programmer's Refernce Manual, Motorola Inc.
6. Robot Motion Control 기술개발 연구보고서, 한국기계연구원, 1992
7. '92 수치제어(NC) 기술사업 보고서, 한국기계연구원, 1992
8. Schildt, "C Power User's Guide", Prentice Hall