다중 로보트 제어 언어 개발

°김 태원*, 서 일홍*, 오 영석**
* 한양대학교 전자공학과, ** 금오공대 제어계측과

Development of a Multi-Robot Control Language

Tae Won Kim*, Il Hong Suh*, Young Suk Oh**
* Hanyang Univ., ** Kumoh Inst. of Tech.

ABSTRACT

A Multi-Robot Control Language (MRCL) is proposed to effectively control the multi-robot. MRCL has not only single-robot command, but multi-robot command with multi-task OS, XINU. Concurrent motion, coordinate motion, and simple collision avoidance motion are implemented. This language is expected to act as a intelligent supporting tool for multi-robot system. To verify the effectiveness of the MRCL, a simple puzzle matching example is illustrated.

1. 서론

산업용로보트를 이용하는 공장자동화에 대한 계속되는 노력은 생산성 향상을 통한 국제 경쟁력 강화 및 노동자의 처우개선등을 위하여 더욱 증가하고 있다. 자동화 공장에서의 산업용 로보트의 응용은 단순한 물건을 집어 옮기거나 간단한 작업물을 분류하는 것을 넘어서, 연속된 경로를 오차없이 추적하며 작업을 하는 아크 용접작업이나 고도의 정밀도를 요하는 조립작업에까지 확대되고 있다[1][2]. 그러나 이들 대부분은 한대의 로보트를 이용하여 작업을 하고있어 그 응용이 제한되어 있다. 예를들어 한대의 로보트가 옮기기에는 부적합한 특수한 모양의 작업물을 이동시키거나, 작업물을 잡고 정밀 조립작업을 하는 일등에는 인간과 같이 두대 또는 그 이상의 로보트가 협조적인 동작을 취해야만 한다. 또한 생산라인에 여러대의 로보트를 배열해 놓았을때 이들 로보트들의 작업공간이 서로 겹치게 되므로 효과적인 작업을 위해서는 로보트 간에 협조제어를 해주어야 한다. 따라서 작업장 내에 있는 여러 로보트뿐만 아니라 주변장치들을 동시에 제어하며 로보트를 이용하여 복잡하거나 정밀한 조립작업을 하기 위해서는 힘센서(force sensor)나 비전센서(vision sensor), 근접센서(proximity sensor)등 여러 센서 신호를 사용하여야 하는 바, 현재의 대부분의 산업용 로보트 제어장치와는 다른 보다 발전된 형태의 다중 로보트 및 제어 언어가 필요하다[3].

다중 로보트 제어 언어에 관한 연구는 1974년 Stanford 대학의 Artificial Intelligence Laboratory에서 AL에 대한 연구를 시작으로, 1981년 McDonnell Douglas에서 NC 기계를 프로그래밍하기 위한 언어인 ATP를 확장시켜 개발한 MCL, 1982년 GE에서 개발한 Help등이 있으나[4] 국내에서는 아직 초보적인 연구 단계에 머무르고 있다.

본 제어 언어는 사용자가 프로그램하기 편하도록 자연어에 가깝게 하고, 기존의 한대용 로보트 언어 및 다중 로보트용 언어를 구비하고, 다양한 센서 입출력 (Sensor I/O) 기능을 갖도록 한다.

개발된 로보트 제어 언어를 다중 로보트 시스템에 이식하여 다양한 실험을 통해 그 효용성을 보이고자 한다.

2. 로보트 제어 언어의 구조와 특징

2.1. 개요

기존의 자동화에 비해 산업용 로보트를 이용한 자동화의 주요 잇점은 융통성(flexibility)과 programmability에 있다. 이의 실현은 로보트의 제어에 적합한 로보트 제어 언어 (Robot Control Language)를 설계 · 이용함으로써 가능하다[5].

본 연구에서는 기존의 한대용 로보트 제어 언어의 기능을 거의 포함할 뿐 아니라 두대의 로보트를 대상으로 하는 명령어를 추가하여 효율적인 프로그래밍 환경을 제공하고자 한다.

제안한 다중 로보트 제어 언어 시스템은 다음의 사항에 중점을 두었다.

첫째, 언어의 구조는 이해하기 쉽게 자연어의 형태를 취한다. 상품화된 기존의 언어는 대다수가 이해하기 힘든 약어로 되어 있기 때문에 언어 자체만을 가지고 작업내용을 알기는 힘들다. 따라서 High-level Language인 C나 BASIC과 유사한 형태로 기술하여 이해 및 수정이 쉽게 한다.

둘째, 기존의 로보트 제어 언어 중 필요한 기능은 모두 구현한다. 기존의 제어 언어를 비교 · 분석하여 필요없거나 유사한 기능을 제외한 거의 대다수 기능을 구현하여 로보트 콘트롤러(Robot Controller)에 한대의 로보트를 접속하여도 동작에 전혀 이상이 없게 한다.

셋째, Concurrent OS를 이용하여 다중 로보트 전용 명령어를 구현한다. 다중 로보트의 장점인 동시작업을 위하여 cobegin, coend 명령을 추가하였으며, collision avoidance 기능을 간이로 구현하였다.

넷째, 다양한 센서(sensor)의 접속이 가능하게 한다. 디지틀 입출력(Digital I/O), 아나로그 입출력(Analog I/O) 뿐 아니라 비전 센서(Vison Sensor)의 접속이 가능하게 한다. 특히 이들 센서를 로보트 언어에서 직접 다룰 수 있어 센서를 이용한 지능제어(Sensor-based Intelligent Control)가 가능하게 된다.

2.2. 시스템 OS (XINU)

시스템에 사용된 concurrnet OS인 XINU는 다중 프로세싱(Multi-processing)이 가능하고, 소스 코드(source code)가 공개되어 있어 이식 및 수정이 용이하다는 장점이 있다[6]. 이외에도 XINU는 일반적인 OS를 구성하는 모든 요소 - 메모리 관리, 프로세스(process) 관리, 프로세스 코오디네이션(process coordination) 및 동기화(synchronization), 프로세스간 통신, 리얼 타임 클락(real time clock) 관리, 디바이스 드라이버(device driver), 머신(machine) 관리 통신망, 파일 시스템(file system) 등 - 를 갖추고 있다. XINU는 한 프로세서(processor)가 독립적인 여러 개의 프로그램을 수행해야 하므로 시분할(time sharing) 프로그래밍 방법을 적용하여 일정 시간만큼 각각의 프로세스(process)를 수행시키는 방법을 이용한다.

2.3. 프로그래밍 시스템의 구성

4대의 68000 CPU board를 이용한 다중 로보트 제어 시스템의 전체 구조는 그림 3, 4와 같다[7]. 본 로보트 제어 언어는 관리제어 시스템(Supervisory System)인 CPU #1에 porting되어 다른 CPU board에 명령을 내리게 된다. 관리제어 시스템은 시스템 자기 진단을 하는 Diagnosis mode, 시스템 초기화 및 파라미터를 설정하는 Setup mode, 비전(vison) 시스템을 관리하는 Look mode, 각 로보트의 움직임을 교시(teach)하는 Teach mode, 교시한 대로 움직임을 수행케하는 Run mode, 그리고 다중 로보트용 언어를 이용하여 원하는 작업을 수행하도록 프로그램하는 Edit mode 등이 있다. 이중에서 로보트 제어 언어는 Edit mode와 Run mode에서 쓰이게 된다. Edit mode에서는 실행하고자 하는 명령을 키보드(Keyboard)를 이용하여 입력하는데, 이때 각 문장이 미리 정의된 문법에 맞는지 점검한다. Run mode에서는 전달된 프로그램이 문법에 맞는지 점검한 후 해당 명령을 수행한다.

2.4. 다중 로보트 제어 언어의 특징

본 로보트 언어는 앞에서 언급한 4가지 중점 사항인 자연어 형태, 기존 언어 구현, 다중 로보트 전용 명령어 구현, 다양한 센서 접속 외에도 조건부 동작이 있다.

1) 다중 로보트 전용 명령어

① 동시 동작 (cobegin, coend)

로보트 두대가 동시에 작업을 할 경우, cobegin, coend 사이에 해당 명령어를 넣으면 동시에 움직이게 된다. 예를 들어, 그림 5에서 보듯이 cobegin, coend에 의해서 robot1과 robot2는 joint motion으로 각각 loc1과 loc2 위치로 동시에 움직이게 된다.

cobegin, coend 명령은 XINU의 concurrent processing을 이용하여 구현된다. 그 과정을 자세히 살펴 보면 다음과 같다.

i) 문장을 해석하는 process의 priority를 현재 실행되는 process의 priority보다 높게 한다.
ii) cobegin, coend 사이의 문장을 해석하여 각각의 process를 생성한다. 이때 생성된 process의 priority는 문장을 해석하는 process보다 낮다.
iii) 문장을 해석하는 process의 priority를 생성된 process의 priority보다 낮게 만든다.
iv) 생성된 process가 동시에 수행된다.

단순히 process를 생성하지 않고 위와 같이 복잡한 과정을 거친 이유는, 해석 process와 수행 process의 priority가 같은 경우, 수행중에 다음 문장이 해석되어 순차적인 동작이 이루어지지 않기 때문이다. 즉, cobegin, coend 사이의 문장뿐아니라 다른 모든 문장이 한꺼번에 생성되어 수행되는 것을 방지하기 위해서이다. 다른 일반적인 문장의 경우에 있어서도 수행 process의 priority가 해석 process의 priority보다 높아야 한다는 규칙은 반드시 지켜져야 한다.

② 협조 동작 (master & slave motion ; ms_mov)

마스터(master)로 지정된 로보트에 대해 슬레이브(slave) 로보트가 일정 거리와 각도를 유지하며 동시에 움직이게 된다. 이 움직임은 로보트 두 대가 협동작업 - 특히, 막대나 상자 등을 들고 움직일 때 - 을 할 때, 각각에 대해 교시 및 명령을 내릴 필요가 없이 마스터 로보트 만을 교시하면 되도록 해준다.

2) 센서 입출력

다양화된 자동화 공정에서 로보트가 자신의 동작만으로 주어진 작업을 성공적으로 끝내기는 거의 불가능하다. 즉, 다른 보조적인 장치의 도움없이는 일을 할 수 없다. 이를 위해서 효과적인 센서 입출력이 반드시 필요하다. 본 연구에서는 디지털 및 아나로그 입출력 외에 로보트의 고기능 작업을 위한 비전 센서를 접속할 수 있게 하였다. 범용 비전 시스템을 이용할 수 있게 종의 비전 전용 명령어를 만들었다. 관리제어 시스템의 Look mode에서 학습시킨 데이터를 이용하여 움직임의 목적지를 비전 센서 입력으로 지정할 수 있다. 예를 들어, Look mode에서 어떤 물체 A를 학습시켰을 경우, 'movj robot1 to loc1 …' 대신 물체의 이름 A를 이용하여 'movj robot1 to A …'로 할 수 있다. 이와같이 비전 센서를 이용하게 되면 고정된 교시점(teaching point) 대신 물체가 현재 놓여진 위치를 이용하여 로보트를 제어할 수 있다.

3) 조건부 동작

공동 작업 영역 안에서 두 대의 로보트가 움직이는 경우 충돌의 가능성이 항상 있게 마련이다. 또한 센서의 입력에 따라 움직임을 조정해야 될 경우도 있다. 이를 위해서 두로보트 사이의 거리나 센서 입력 등에 따라 움직임을 조정해주는 것이 조건부 동작이다. 이는 일반적인 움직임 명령 뒤에 'on *condition* do *action*'의 문장이 덧붙는다. 그림 6에서 보듯이 조건에 따라 필요한 함수나 동작이 수행된다.

이와같이 동작 중에 조건에 따라 움직임이 변하는 방식외에도 조건이 맞을 때까지 기다리다 수행하는 방식도 있다. 이 방식의 표현은 위의 방법과 거의 같으나 수행 문장과 조건의 순서가 반대로 된다. (그림 7 참조)

3. 실험

제안된 다중 로보트 제어 언어는 IBM-PC 386에서 LEX, YACC을 이용하여 C 소스(source)로 만든 후, MCC68K로 크로스 컴파일(cross compile)하여 다중 로보트 제어 시스템에 이식하였다[8][9]. 이 시스템을 이용한 '글자 맞추기' 실험을 통해 제안된 다중 로보트 언어의 효용성을 살펴 보았다. 먼저, 관리 제어 시스템의 Look mode에서 해당 물체를 학습시킨 후, Teach mode에서 각각이 놓여질 위치를 학습시킨다. 물체를 임의로 흩어 놓고 로보트가 글자의 정해진 순서대로 맞추게 한 후, 맞춰진 조각이 든 판(plate)을 두 대의 로보트가 협조제어하며 들어 다른 위치로 옮기게 한다. 이와같은 실험을 통해 각 명령의 개별 동작, cobegin - coend 동작, ms_mov 동작, 비전 기능, I/O 기능 등의 효용성을 입증하였다. 실험에 사용된 프로그램의 중요 부분은 다음과 같다.

```
.....
40   movl robot1 to loc5_h with sp = 40
60   movl robot1 to loc5_l with sp = 20
70   release robot1 with width = 1
80   movl robot1 to loc5_h with sp = 60
90   cobegin
100      movl robot1 to safe_loc with sp = 70
110      movl robot2 to loc1_h with sp = 70
120  coend
140  movl robot2 to loc1_l with sp = 20
150  release robot2 with width = 1
160  movl robot2 to loc1_h with sp = 70
170  cobegin
180      movl robot1 to loc1_h with sp = 70
190      movl robot2 to safe_loc with sp = 80
200  coend
.....
1200 ms_mov master to pick_l with sp = 20
1210 release robot1 with width = 1
1220 release robot2 with width = 1
1230 ms_mov master to pick_h with sp = 50
1240 ms_mov master to place_h with sp = 25
1250 ms_mov master to place_l with sp = 20
1260 grasp robot1 with width = 1
1270 grasp robot2 with width = 1
1280 ms_mov master to place_h with sp = 70
```

4. 결과

본 연구에서는 다중 로보트 제어 시스템을 위한 제어 언어의 개략적인 구조를 제시하고, 이를 위해 사용된 multi-task OS XINU와 전체 제어 시스템에 대하여 간략히 설명하였다. 또한 기존의 제어 언어에 비해 다중 로보트 제어 언어의 다른점에 대해 서술하였다. 제어 언어의 효용을 확인하기 위하여 '글자 맞추기' 실험을 통하여 XINU가 제공하는 concurrent processing이 동시 동작과 협조 동작에 유효한지 검토하였다.

참고 문헌

[1] Yoram Koren, *Robotics for Engineers*, McGraw-Hill, 1985

[2] M.P.Groover et al., *Industrial Robotics Technology. Programming, and applications*, McGraw-Hill, 1986

[3] 한양대학교, 삼성항공, "다중 로보트 제어 시스템 개발" 최종 보고서, 1990. 10.

[4] William A. Gruver et al., "Industrial robot programming languages:a comparative evaluation," IEEE Trans. on Sys., Man, and Cyber., Vol.SMC-14, No.4, 1984, pp.565-570

[5] 어 희주, 다중 로보트 시스템을 위한 로보트 제어 언어의 개발, 한양대학교 석사학위 논문, 1990.2.

[6] D. Comer & T.V. Fossum, *Operating System Design. Vol.1: The XINU Approach*, Prentice-Hall, 1988

[7] 현 웅근, 서 일홍 외 10명, "다중 로보트 제어 시스템의 개발," 한국 자동제어 학술회의, 1990.10.

[8] *System V/68 Release 3 Programmer's Guide*, Ch.5,6, Motorolar Inc., 1987

[9] *PARAGON MCC68K C Cross Compiler Reference Manual*, Microtec Research, 1986

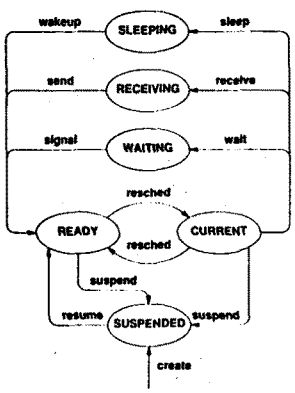


그림 1. XINU의 프로세스 상태도

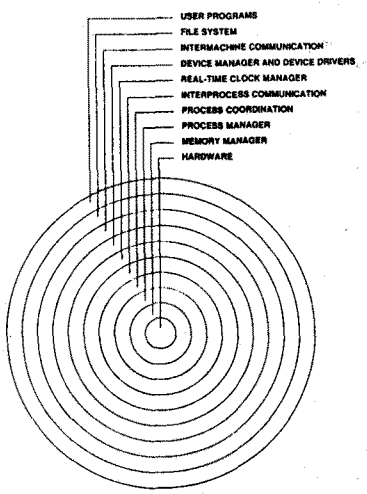


그림 2. XINU의 계층적 구조

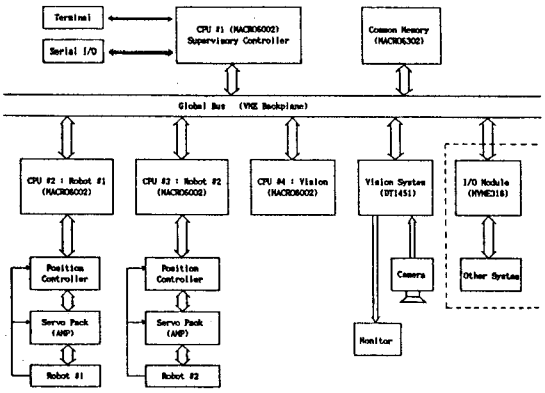


그림 3. 다중 로보트 제어 시스템의 전체 H/W 구성도

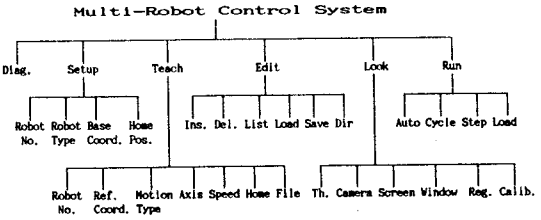


그림 4. 다중 로보트 제어 시스템의 전체 S/W 구성

```
cobegin
    movj robot1 to loc1 with sp = 40
    movj robot2 to loc2 with sp = 60
coend
```

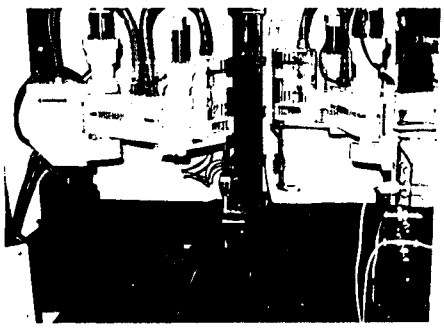
그림 5. 동시 작업의 예

```
movj robot1 to loc1 with sp = 40 on dist < 3
        do stop
movl robot2 to loc2 with sp = 80 on dist < 3
        do col_avoid
movj robot1 to loc3 with sp = 70 on time > 30
        do call time_out
```

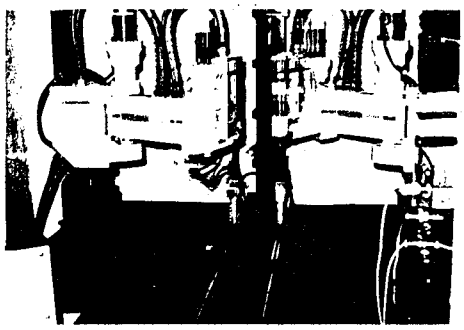
그림 6. 조건부 동작의 예 (1)

```
on dist > 3 do movj robot1 to loc1 with sp = 40
on dist > 3 do call move_motion
on time > 20 do call time_out
```

그림 7. 조건부 동작의 예 (2)



(a) cobegin, coend 동작



(b) MS_MOV 동작

그림 8. '글자 맞추기' 실험 모습