PC에 기반을 둔 개방형 로봇제어시스템 : PC-ORC

A PC-Based Open Robot Control System : PC-ORC

김 점 구, 최 경 현, 홍 금 식
(Jeom-Goo Kim, Kyung-Hyun Choi, and Keum-Shik Hong)

Abstract : An open architecture manufacturing strategy intends to integrate manufacturing components on a single platform so that a particular component can be easily added and/or replaced. Therefore, the control scheme based upon the open architecture concept is hardware-independent. In this paper, a modular and object oriented approach for a PC-based open robot control system is investigated. A standard reference model for robot systems, which consists of three modules; hardware module, operating system module, and application software module, is first proposed. Then, a PC-based Open Robot Controller(PC-ORC), which can reconfigure robot control systems in various production environments, is developed. The PC-ORC is built upon the object-oriented method, and allows an easy implementation and modification of various modules. The PC-ORC consists of basic softwares, application objects, and additional hardware device on the PC platform. The application objects are: sequencer, computation unit, servo control, ancillary equipment, external sensor control, and so on. In order to demonstrate the applicability of the PC-ORC, the proposed PC-ORC configuration is applied to an industrial SCARA robot system.

Keywords : PC-based controller, object-oriented analysis, open architecture, robotics

I. 서론

현대의 생산시스템은 제품에 대한 소비자들의 다양한 요구를 만족시키고, 생산기술의 변화와 컴퓨터 관련 기술의 급격한 발달에 빠르게 적응하기 위해서 유연성(flexibility), 통합성(integration) 및 동시성(concurre-ncy)을 만족시키는 개방구조(open architecture)로의 전환이 요구되고 있다[1,2]. 개방형 생산시스템은 기존의 제조장비들의 하드웨어와 소프트웨어들을 하나의 표준환경으로 통합운영하여 서로 다른 장비들의 통합문제를 해결할 수 있는 생산체제이다. 생산시스템에서 중요한 역할을 담당하는 산업용 로봇, 수치제어 공작기계 등과 같은 자동화 장비들은 다양한 작업기능과 사용자 프로그래밍이 가능한 구조이다. 그렇지만, 제조회사마다 자사의 전용 제어시스템을 사용하는 폐쇄구조(closed architecture)로 되어있다. 폐쇄구조는 제조자 중심(vendor-oriented)의 구조이기 때문에 표준환경의 단일 제어시스템으로 통합하여 운영할 경우, 상호 호환성의 문제로 인한 유연성의 결여를 가져온다. 또한, 통합을 위해 중복된 주변장치로 인한 자원 낭비와 시스템 효율의 저하를 가져오게 된다.

서로 다른 제조회사들의 장비들을 통합하여 사용하지 못하는 문제를 해결하기 위한 한가지 방법으로 각 표준화 단체들은 자동화장비에 대한 제어시스템의 개발 시에 표준규격을 사용할 것을 제시하고 있다[3]. 그렇지만, 표준화된 방식을 통한 제어시스템의 통합은 제조회사의 개발경향을 제약할 수 있다. 그리고 기존의 제조자 중심에서 이미 개발된 하드웨어와 소프트웨어를 사용하지 못하게 되어, 자원의 낭비를 초래하기 때문에 제조회사들로부터 크게 호응을 얻지 못하고 있다.

또 다른 접근법으로 제조회사의 개발경향을 제약하지 않고, 기존의 자원을 재사용하며, 사용자 편의의 독자적인 기능을 구현할 수 있는 개방형 제어시스템에 대한 연구가 제조회사들에게 크게 호응을 얻고 있다. 이러한 접근법의 연구로써, GM등 몇 개의 자동차회사가 상호 협력하여 OMAC(Open Modular Architecture Controllers)이라 불리우는 개방형 모듈구조의 제어기 개발에 관한 연구를 수행하고 있으며[4], NC와 FA용 개방형 시스템을 위하여 OSEC(Open Systems Environm-ent for Controller)이 제안되어 있다[5]. 또한, 유럽에서는 FA용 제어장치의 개방화에 관한 OSACA (Open System Architecture for Controls within Automation Systems)를 제안하고 있다[6]. 이러한 개방형 모델에 대한 기존의 연구에 대하여 요약하여 서술하면 다음과 같다.

OSACA는 1992년부터 독일의 stuttgart 대학 연구소가 중심이 되어 유럽의 8개 기업이 참여하고 있는 FA용 제어장치의 개방화에 관련된 연구이다. 이러한 OSACA의 목적은 특정업체에 의존하지 않는 개방형 제어장치에 대한 구조를 제시하는 것이다. 여기서 제어장치는 CNC장치를 비롯해서 로봇 및 PLC, 셀 등의 각종 제어장치를 포함하고 있으며, 구조는 하드웨어에 의존하지 않는 소프트웨어 구조를 고려하고 있다. 특히, OSACA시스템의 특징은 모듈식 구조를 통하여 제어장치의 기능을 재편성하는 것이 항상 가능하다는 것이다. OMAC은 1994년부터 미국의 GM을 중심으로 통합이 용이한 NC장치의 구현에 대한 연구이다. OMAC이 최종적으로 실현하고자 하는 시스템은 PC를 플랫폼하여 OSACA와 동일한 모듈식 구조를 가진 개방형 시스템이다. OSEC은 1993년부터 다수의 일

접수일자 : 1999. 6. 10., 수정완료 : 2000. 1. 27.
김점구 : 부산대학교 대학원 지능기계공학과
최경현 : 제주대학교 기계에너지생산공학부
홍금식 : 부산대학교 기계공학부 및 기계기술연구소

본기업을 중심으로 차세대 생산기계 시스템의 개발에 대한 연구이다. OSEC은 NC공작기계 시스템에 대하여 크게 조작계, 기계계, CAD계 그리고 생산 네트워크계 등 4개의 관점에서 개방화를 추진하고 있다. 조작계에서는 기계의 표준모델화를 추진하고 있으며, 기계계에서는 제어 소프트웨어의 부품화와 인터페이스 표준화를 위해 OSEC API를 표준화했다. CAD계에서는 형상정의에 대응한 가공법의 서술과 가공 노하우를 소프트웨어로 구현하기위해 OSEL이라는 표준을 정했다. 그리고 최근에는 이러한 개방형 제어시스템에 대한 연구와는 별도로 가상시뮬레이션 환경과 제어기를 결합함으로써 생산 효율성을 높이고, 급변하는 생산환경에 적응하고자 하는 연구들이 진행되고 있다[7,8].

또한 최근 국내에서도 몇몇 산업체와 대학에서 개방형 제어시스템에 대한 연구들이 이루어지고 있으며, 이러한 연구들에 대하여 요약하여 서술하면 다음과 같다. 국내에서 개방형 제어시스템에 대한 초기 연구들은 CNC공작기계를 중심으로 이루어졌으며[9], 현재에는 일부 제품들이 상용화되어 시판되고 있다. 공작기계에서 시작된 개방형 제어기술들은 점차적으로 로봇 시스템[10,11,28] 및 MMI (Man Machine Interface) 시스템과 가공 셀 시스템에도 응용되기 시작하고 있으며[12,25], 개방형 제어시스템 구현을 위한 통신시스템에 대한 연구도 일부 이루어지고 있다[13]. 또한 최근 들어서는 PC를 기반으로 한 개방형 제어기들도 소개되고 있다[26,27]. 그러나 국내에서의 개방형 제어시스템에 대한 연구들은 하드웨어 및 통신시스템에 대한 연구에 국한되어 있으며, 개방형제어기와 가상시뮬레이션 환경을 결합하고자 하는 연구는 거의 미미한 상태이다.

따라서, 본 논문에서는 OSACA에 바탕을 둔 표준기준모델을 사용하여 관련된 모듈들을 객체지향형으로 모델링하여 로봇의 제어구조를 개방화하고, 이와 같이 개방화된 제어구조에 상용의 가상시뮬레이션 환경을 구현하는 소프트웨어를 탑재함으로서 시뮬레이션과 제어를 동시에 할 수 있는 새로운 형태의 PC기반 개방형 로봇제어기(PC-based Open Robot Controller, PC-ORC)를 제시하고자 한다. 또한 제안된 PC-ORC의 적합성을 검증하기 위해서 현재 생산현장에서 사용되고 있는 스카라 로봇에 PC-ORC를 적용하여 그 성능을 평가한다.

본 논문의 구성은 다음과 같다. II절에서는 PC-ORC를 위한 표준기준모델에 대하여 살펴보고, III절에서는 산업용 로봇시스템의 구조를 분석하고 분석된 로봇의 구성요소들을 하나의 독립된 객체로 정의한다. 이러한 객체들을 상용의 하드웨어와 소프트웨어로 구현하여 PC-ORC를 제시한다. IV절에서는 사용자와의 인터페이스를 위한 모션 프로그램밍 방법에 대해서 논의한다. 또한, V절에서는 산업현장의 스카라로봇에 제안된 PC-ORC를 적용하여 적합성 및 성능을 평가한다. 마지막으로 VI절에서는 연구결과를 요약정리 한다.

II. PC-ORC를 위한 표준기준모델

본 절에서는 기존의 OSACA를 바탕으로 해서 제안된 표준기준모델을 살펴본다. 이러한 모델은 로봇시스템이 객체지향형 개방구조를 가지도록 구성하기 위한 기준이 된다. 그림 1은 로봇제어시스템을 개방화하기 위해서 본 논문에서 사용하고 있는 표준기준모델을 보이고 있다. 이러한 모델은 하드웨어플랫폼, 운영시스템모듈, 응용소프트웨어모듈 등으로 구성되어 있다.

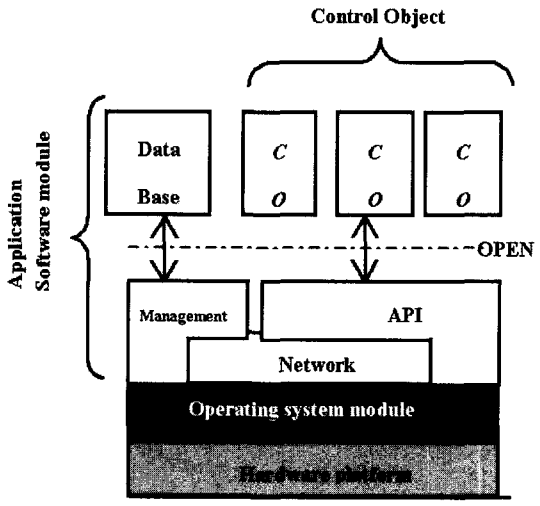


그림 1. 제안된 PC-ORC를 위한 표준기준모델.
Fig. 1 A proposed standard reference model for the PC-ORC.

1. 하드웨어 플랫폼

일반적으로 하드웨어 플랫폼은 계산을 담당하는 프로세서, 다른 외부장치로 입출력을 담당하는 I/O장치, 데이터 저장을 담당하는 메모리, 그리고 사용자 편의를 제공하는 주변장치들로 이루어져 있다. 이러한 하드웨어 플랫폼의 구성요소들을 개방화하기 위해서는 각 구성요소들이 표준화된 제원을 따르는 구조로 구성되어야 한다. 그러나, 오늘날 하드웨어가 빠른 속도로 발전하기 때문에 하드웨어 자체에 대한 제원을 표준화하기보다는 상황에 따라 가장 적합하고 가격이 저렴한 하드웨어를 어느 때고 쉽게 사용이 가능하게 하기 위해서는 하드웨어 드라이버가 개방화된 구조를 가져야 한다. 이러한 드라이버의 개방화를 위해서 최근 PC가 하드웨어플랫폼으로 도입되고 있다. 이와 같이 PC를 하드웨어플랫폼으로 도입하면 다음과 같은 장점을 가진다. PC는 하드웨어적인 측면에서 다양한 주변장치에 대한 적응성이 뛰어나기 때문에 최근에 개발된 하드웨어 기술들을 금방 도입할 수 있으며, 또한 하드웨어 드라이브 공급자 입장에서 쉽게 접할 수 있는 범용성을 갖추고 있다. 소프트웨어적인 측면에서는 구동프로그램을 모듈화하고 수정이나 확장성을 가지도록 설계하는 것이 용이하며, 빠르게 발전하는 소프트웨어 기술들을 쉽게 채용할 수 있다[14]. 또한 PC를 하드웨어 플랫폼으로 사용하면 제어시스템의 가격저하와 고장시 교체가 용이한 부수적인 효과가 기대된다.

2. 운영시스템 모듈

개방형 제어시스템에서 운영시스템 모듈은 하드웨어, 응

용소프트웨어, API(Application Programming Interface)등에 대한 유용성(availability)을 제공해 주어야 한다. 여기서 유용성은 제3자가 개발한 주변장치 및 소프트웨어에 대한 활용성을 나타낸다. 하나의 프로세서로 운영되는 PC를 개방형 제어시스템의 플랫폼으로 사용한다면 반드시 운영시스템 모듈이 실시간 운영시스템(Real Time Operating System: RTOS)으로 구성되어야 한다. 이러한 RTOS와 Windows/DOS의 중요한 차이는 CPU시간의 관리방법, 메모리의 할당방법, 인터럽트에 대한 응답과 기계장치를 구동하는 처리과정에서 자원공유방법 등에 달려 있다.

3. 응용소프트웨어 모듈

개방형 제어시스템에서 응용소프트웨어 모듈은 필요에 따라 기능의 추가 및 변경이 허용되고, 각 구성요소간에 정보교환이 가능한 객체지향적 특성을 만족해야 한다. 이와 같은 특성을 만족하기 위해서 응용소프트웨어 모듈은 관리, API, 네트워크, 제어객체 등으로 구성된다.

3.1 관리(management)

개방형 제어시스템의 가장 중요한 특징은 환경재설정(reconfiguration)의 기능이다. 관리는 이러한 개방형 제어기의 환경재설정의 역할을 수행한다. 일반적인 개방형 제어시스템의 환경 재설정과정은 다음과 같은 과정으로 이루어진다. 먼저 사용자는 off-line에서 실제 제어시스템의 기능 및 성능명세를 분석한다. 그리고 개방형 표준기준모델에 입력할 변수값을 환경설정편집기에 입력하여 환경설정주문(configuration order)을 작성한다. 그리고 환경설정주문에 포함된 변수값을 각종 제조장비의 구성요소에 대해 체계적으로 분류하여 구성한 데이터베이스로부터 가져와서 개방형 제어시스템의 환경을 재설정한다. 이러한 환경 재설정기능은 제조환경의 변화에 대응하기 위한 개방형 제어시스템의 필수요소가 된다.

3.2 API

제조환경은 생산제품이나 생산공정의 변화에 따라 제조장비의 재구성이나 추가적인 설치를 요구하게 된다. 이러한 제조환경의 변화에 따라 제어시스템의 구성요소도 재조정이 요구된다. 이와 같은 제어시스템의 유연성이나 확장성에 대한 문제는 개방형 제어시스템을 구성하는데 많은 어려움을 준다. 이러한 문제점에 대한 대처방안으로 하드웨어나 소프트웨어 측면에서 기존의 하드웨어나 소프트웨어와 공통된 인터페이스를 형성할 수 있는 범용의 API 모듈들을 시스템기반에 포함시킨다. 이러한 API모듈은 제어시스템의 모든 구성요소가 쉽게 재사용 또는 추가될 수 있도록 한다.

3.3 네트워크(network)

네트워크는 객체들의 현재상태의 값을 읽어 들이거나 필요한 동작에 대한 적절한 값을 보낼때 데이터버스의 역할을 한다. 네트워크는 개념적으로 두개의 부분으로 나뉘어 데이터를 교환한다. 즉, 기능 수행을 위한 명령을 하달하는 클라이언트(client)와 클라이언트의 명령에 따라 요구된 기능을 수행하는 서버(server)이다[1].

3.4 제어객체(control object)

제어객체는 제조장치를 실제로 제어하는데 사용되는 구성요소를 소프트웨어측면에서 캡슐화하여 객체로 나타낸 것이다. 이와 같은 제어객체는 사용자가 객체의 속성을 재설정함으로서 새로운 생산환경의 제어시스템 구축이 가능하고, 필요에 따라서 추가 및 삭제가 가능해야 한다. 또한, 제어객체는 개방형 제어시스템의 하드웨어에 독립적이어야 한다.

III. 산업용 로봇시스템의 구조분석과 객체지향형 모델링

본 절에서는 먼저 산업용 로봇시스템의 구조를 분석하고 분석된 구성요소들이 앞에서 제시된 표준기준모델과 같은 개방형 구조를 가질 수 있도록 새로운 객체를 정의하고, 정의된 객체들을 상용의 하드웨어와 소프트웨어로 구현한다. 이와 같이 구현된 객체들을 PC상에서 통합하고, 객체의 속성들을 재설정할 수 있는 소프트웨어를 개발함으로서 PC-ORC를 완성한다.

1. 산업용 로봇시스템의 구조분석

산업용 로봇시스템은 일반적으로 그림 2와 같이 머니퓰레이터, 메모리장치, 시퀀스장치, 외부센서장치, 서보제어기, 계산장치, 보조장비 제어장치, 작업 제어장치, 동력전환장치, 경로생성기 등으로 구성된다[15]. 이러한 로봇시스템의 구성요소들은 실시간 처리의 필요성에 따라 관리모듈과 서보모듈로 구별할 수 있다. 관리모듈은 기능적인 관점에서 로봇을 관리하면서 작업자로부터 받은 명령어를 처리하고 로봇이 동작하여야 할 궤적을 생성하는데 관여하기 때문에 꼭 실시간 처리가 필요하지는 않다. 서보모듈은 로봇관절의 위치와 속도를 제어하는데 관여하기 때문에 꼭 실시간 처리가 필요하다.

관리모듈은 시퀀스와 계산요소, 그리고 외부센서 인터페이스와 보조장비 연결요소, 작업제어장치, 경로생성기 등으로 구성된다. 시퀀스는 메모리에 저장된 로봇 모션과 관련된 정보를 해석하여 제어기 내의 다른 모듈들이 인식할 수 있도록 정보를 전환하여 전달한다. 그리고 계산요소는 로봇의 궤적생성, 역기구학 해석 등 로봇에 관련된 계산을 담당한다. 외부센스 인터페이스는 외부센서(비젼센서, 힘센서 등)를 통해 전달받은 정보를 시퀀스가 인식할 수 있도록 논리적 변환작용을 수행한다. 또한, 보조장비 연결요소는 외부장치(셀 제어기, 컨베이어, 수치제어 공작기계 등)와 통신하기 위해 입출력 신호의 송수신을

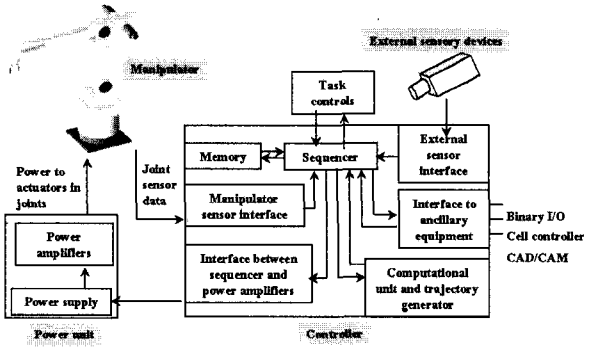


그림 2. 일반적인 산업용 로봇시스템의 구성요소.

Fig. 2. Various elements of a general industrial robot.

수행한다. 작업 제어장치는 로봇구동에 관련된 점, 경로, 프로그램을 운용자로부터 입력 받아서 로봇 모션작업을 제어하는 장치(로봇 모션프로그램 소프트웨어, 교시상자 등)들을 나타낸다.

서보모듈은 시퀀스로부터 로봇 동작에 대한 정보를 받아서 속도 및 위치를 제어하는 서보제어기와 각 관절의 위치 및 속도 정보를 획득하는 머니퓰레이터 센서 장치들로 이루어져 있다. 서보제어기는 일정한 주기로 호스트에서 전달받은 정보를 바탕으로 각 관절의 모터를 제어하며, 센서장치는 모터에 부착된 센서로부터 측정된 속도 및 위치 정보들을 호스트에 주기적으로 전달한다.

현재 이러한 산업용 로봇의 각 구성 요소들은 대부분 폐쇄형 구조로 이루어져 있기 때문에 다른 제어시스템과 통합하여 사용할 경우, 여러 가지 문제점들을 나타내고 있다. 또한, 각 구성 요소들을 재사용하거나 교체하는 것이 불가능하기 때문에 자원낭비가 심하다. 본 논문에서는 이러한 산업용 로봇시스템의 구조를 PC와 개방형의 운영체제 모듈 및 응용소프트웨어 모듈로 대체하여 새로운 개방형 로봇제어시스템을 구성한다.

2. 운영체제 모듈의 구성

본 논문에서는 PC-ORC의 구현을 위한 운영체제로서 PC기반에서 비교적 안정적인 Windows NT를 채용한다. 그러나 Windows NT를 실시간 운영체제로 사용하는데는 여러 가지 문제점이 있다. 이러한 문제점에 대해서 상술하면 다음과 같다. 먼저 서보의 디지털 제어를 위해서는 반듯이 주기적인 서보루프가 보장되어야 하는데 수 밀리의 서보 루프를 Windows NT에서 보장하기는 어렵다. 이러한 문제점을 해결하기 위한 대안으로 주기적인 빠른 시간의 응답을 요구하는 서보계는 전용 서보보드의 제어계를 이용하고, 로봇 동작계획 및 궤적생성 등과 같이 비교적 긴 시간의 응답이 요구되는 부분은 Windows NT가 담당하도록 구성한다. 그러나 이러한 구성만으로는 NT가 서보보드에서 발생되는 인터럽터를 실시간으로 처리하지는 못한다. 그러한 이유를 상술하면 다음과 같다. Windows NT는 스케쥴링 방식에 있어서 다중스레드를 생성하도록 지원하며, 스케줄링 방식이 선점형이다. 또한 스케줄링 과정에서 스레드 우선순위를 필요성에 따라 할당한다. 따라서 Windows NT는 스레드 우선순위 부여에 대한 관점에서는 실시간 운영 체제의 조건을 만족한다. 그러나, 일반 실시간 운영체제가 256개의 우선순위를 제공하는 반면, Windows NT는 최대치가 7개라는 단점이 있다. 또한 Windows NT는 ISR (Interrupt Service Routine)에 의해 짧은 시간 동안 서비스하고, DPC (Deferred Procedure Call)에 의해 나머지 비교적 긴 시간을 요하는 일을 처리한다. 그러나 DPC자체로는 비선점형 스케줄링 방식이라는 단점이 있다. 즉 DPC는 그것을 발생시키는 인터럽터의 우선 순위와 상관없이 모두 동일한 순위로 다루어지고 단지 ISR만이 선점형의 특성을 만족하기 때문에 응용프로그램에 따라서는 다른 디바이스 드라이버에 의한 영향을 받을 수 있으며, 특히 DPC처리 시간이 긴 하드디스크나 네트워크 드라이버에 의한 수 밀리초의 시간지연을 야기시킬 수도 있다[10,16]. 이러한 DPC의 비선점형으로 인하여 발생되는 시간지연을 최소화하기 위해서 서보보드의 메모리에 충분한 버퍼영역을 설정하여 전송된 로봇의 궤적데이터를 미리 버퍼에 저장하고, 버퍼의 공간이 일정량 비워지면 Windows NT상의 응용객체에 궤적의 추가전송을 요구하도록 구성하였다. 이러한 방법은 DPC의 시간지연에 대응하고 서보계의 인터럽터 요구를 최소화함으로서 실시간성을 보장하는 것을 가능하게 한다. 그러나 완전한 경성실시간성 (hard real time)의 보장을 위해서는 상용 RTOS (VxWorks, VRTX, QNX, Lynx, OS9)의 채용을 고려할 필요가 있다.

본 연구에서는 전용 서보계로서 상용의 PMAC 모션제어기를 사용하였으며, PMAC과 Windows NT사이에 dual-port RAM을 설치하여 PC에 할당된 특정메모리에서 고속으로 PMAC의 버퍼메모리 영역에 로봇의 궤적을 다운로드할 수 있다[17]. 또한 설정된 버퍼는 통상적으로 2000개정도의 관절값들을 저장할 수 있으며, 필요에 따라서 버퍼의 크기를 조절할 수 있다. 그리고, 각 관절의 제어 결과값과 서보의 상태가 NT로 전송되도록 구성한다.

3. 응용소프트웨어 모듈의 구성

본 절에서는 개방형 로봇제어시스템의 응용소프트웨어 모듈을 구현하기 위해서 모듈의 구성요소들을 각각 객체로서 정의하고 그 특성과 행위를 부여하는 객체지향적인 모델링을 수행한다[18]. 그림 3에서는 객체지향 접근법의 OMT (Object Modeling Technique)를 사용하여 각 클래스 및 클래스들 사이의 논리적 관계가 정의되어 있다[19, 24]. 이러한 객체지향적 모델링에서 모델링된 객체들에 대한 상호작용 및 역할과 구현방법 등을 상술하면 다음과 같다.

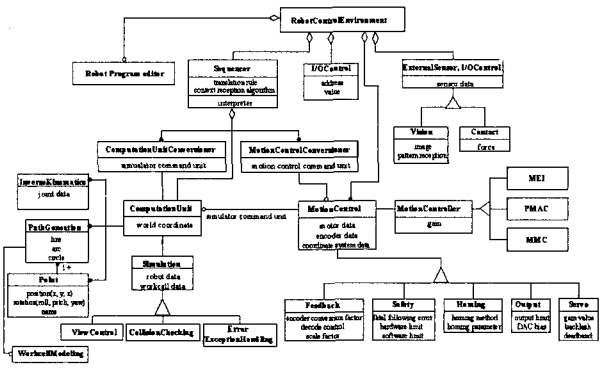


그림 3. 응용소프트웨어 모듈의 클래스 구조.

Fig. 3. Class structure of the application software module.

Sequencer객체는 OR Instruction Set 편집기에서 전달된 명령어를 다른 객체들이 인식할 수 있는 형태로 변환하기 위한 통역작업들을 처리하며, 이러한 변환을 위해서 ComputationUnitConversioner객체 및 MotionControl-Conversioner객체와 서로 상호작용한다. 여기서 ComputationUnitConversioner객체는 Sequencer객체로부터 전달된 정보를 바탕으로 ComputationUnit객체가 인식하고 실

행할 수 있는 형태로 변환한다. 또한 MotionControlConversioner객체는 Sequencer객체로부터 전달된 정보를 바탕으로 PMAC객체로 전달할 데이터파일을 생성한다. ComputationUnit객체는 Point객체, PathGeneration객체, InverseKinematics객체와 상호작용하여 Simulation객체로 전달할 시뮬레이션에 관한 정보를 생성하며, 실제 로봇 태스크 작업을 수행하기 위하여 Simulation객체로부터 CODE™의 3차원 그래픽상에서 수정 및 보완된 궤적데이타를 전달받는다[22]. 여기서 Point객체는 사용자 입력 윈도우를 통해 기준좌표계(world coordinate)에 대한 위치 및 방위정보를 운용자로부터 입력받아 저장하고, 가상 작업 셀에 나타나는 가상로봇의 교시작업을 수행한다. PathGeneration객체는 작업할 궤적이 미리 주어진 경우, CODE상의 가상로봇에 보간 작업을 하지 않고 직접 궤적을 전달하는 역할을 한다. 이 객체는 로봇이 미리 주어진 궤적을 추종하며 작업해야 하는 경우에 필요하다. InverseKinematics객체는 기준좌표계에서 공구의 위치가 주어질 때 각각의 관절들이 움직여야 하는 관절값들을 계산하는 객체이다. 이 객체는 PathGeneration객체를 통하여 사용자가 원하는 경로를 직접 가상로봇에 전달할 때 역기구학을 해석한다. 이러한 각 객체들의 특징 및 역할들은 객체지향형 언어인 C++을 사용하여 각각 클래스들로 작성된다. 이와 같이 작성된 클래스들은 데이터 멤버(data member)들과 멤버함수(member function)들로 구성되어 있다. 데이터 멤버는 각 객체들과 관련된 데이터를 저장하는 역할을 수행하고 멤버함수들은 클래스들이 처리해야할 다양한 연산들을 수행한다. 여기서 멤버함수들은 통상적으로 구현자 함수(implementor function), 보조함수(helping function), 접근함수(access function)들로 구성된다[20,21].

Simulation객체는 운용자로부터 입력된 로봇의 기구학적 정보와 구동기 정보, 작업셀 정보를 입력 받는다. 그리고 실제 로봇을 구동하기 전에 로봇의 제한조건에 근거하여 운용자가 의도한 작업이 맞게 구성되었는지를 검사할 수 있는 3차원 그래픽 환경의 가상 시뮬레이션 환경을 제공한다. Simulation객체는 로봇 및 작업 셀의 모델링기능, 수행할 로봇구동에 관한 충돌점검 작업, 발생된 오차에 대한 보상작업을 수행한다. 이러한 시뮬레이션 환경의 제안목적은 시뮬레이션을 수행하는 동안 원하는 작업이 아닌 경우에는 운용자가 다시 로봇 프로그램을 수정할 수 있는 기반을 제공하기 위함이다. 이러한 Simulation객체는 CODE의 CIMulation 부분과 C++로 구현된 인터페이스 다이얼로그 박스로 구성된다. 또한 인터페이스 다이얼로그박스와 CIMulation 사이의 정보교환은 CODE API를 사용하여 이루어진다[22]. 이러한 정보교환은 기본적으로 궤적을 보내는 경우와 포인터를 보내는 경우로 나누어진다. 궤적을 보내는 경우는 사용자가 로봇이 지나 가야할 경로를 미리 알고 있는 경우로서 로봇이 주어진 궤적을 추종해야 하는 작업을 할 때 필요하다. 포인터를 보내는 경우는 사용자가 로봇이 지나 가야할 몇 개의 점들만 알고 있는 경우로서 CODE상에서 각 점들의 보간이 이루어진다.

MotionControl객체는 로봇제어시스템에 설치된 각 구동기에 관한 환경설정작업을 수행하며, MotionControlConversioner객체에서 전달된 궤적데이터를 PMAC객체로 전송한다. 또한 구성된 하드웨어에 대하여 구동기를 시험하여 상태를 검증하는 작업을 수행한다. 이 객체는 PMAC, MEI, MMC 등 MotionController와 정보교환을 통하여 하드웨어 설정작업에 관련된 작업을 수행하는 하위객체들과 상호작용한다. 이러한 하위객체를 나열하면 다음과 같다. Feedback객체는 엔코드의 카운트당 보여지는 사용자단위를 설정한다. Safety객체는 전류오류가 발생할 때 취해야 할 동작의 모드를 설정하는 작업을 수행하고, 하드웨어/소프트웨어 측면의 근접스위치를 설정하는 작업을 수행하는 등 모터의 안정과 관련된 변수값을 설정한다. Homing객체는 모터의 귀환하는 방법을 규정한다. 그리고 모터의 출력 및 DAC(Digital to Analog Converter)에 관련된 다양한 제어변수를 설정하는 Output객체, 모터의 게인 값을 설정하는 Servo객체 등이 하드웨어 설정작업에 관여하는 하위객체들이다. 이러한 MotionControl객체 및 하위객체들은 PMAC에서 제공되는 OCX와 C++언어를 사용하여 독립된 다이얼로그박스를 작성함으로서 객체로 구현된다.

I/OControl객체는 입출력 인터페이스를 통해 보조장비(컨베이어, 수치제어 공작기계, 셀 제어기 등)들로부터 나온 신호를 입력포트를 통해 받고, 보조장비에 출력포트를 통해 신호를 보내는 작업을 수행한다. 이러한 작업은 인터페이스에 정해진 각 포트를 통해 정의되고, 하드웨어에 관련된 복잡한 상호작용의 세부관계는 캡슐화된다. 그러나, I/OControl객체가 NT상에서 각 입/출력 포터를 제어하기 위해서는 디바이스 드라이브가 구현되어야 한다.

ExternalSensorControl객체는 로봇제어시스템의 외부환경에 대한 정보를 외부 센서로부터 받아들여 필요한 필터연산을 수행하여 구체적인 정보를 생성하고, 이를 Sequencer객체가 인식할 수 있는 데이터형식으로 전환한다. 구체적인 예로서는 비전센서, 힘센서 등을 들 수 있다.

4. PC상에서 모듈의 통합과 객체 서로 간의 정보교환

PC-ORC는 PC를 기반으로 하는 하드웨어 플랫폼과 운영체제 모듈을 구축한 다음, 구현된 응용객체와 CODE 시스템, 기타 하드웨어장치를 PC에서 통합함으로서 구현된다. 또한 응용프로그램 모듈내에서 각 객체사이의 정보를 읽어들이거나, 전달하는 역할은 C++에서 제공되는 TCP/IP 프로토콜을 사용하여 windows NT 운영체제 내에서 내부통신의 형태로 구현된다[20, 21]. 이와 같이 각 객체간의 정보교환을 네트워크를 사용하여 구현하는 목적은 각 객체의 추가/변경시에 개방성 및 독립성을 보장하기 위해서이다. 특히, TCP/IP프로토콜을 사용하는 목적은 다음과 같다. 첫째, TCP/IP통신방식은 현재 산업용 네트워크 방식의 표준으로 소개된 MAP/MMS보다는 기술적으로 여러가지 부족한 점이 있지만 각 분야에서 광범위하게 사용되고 있어 제어기기 공급자도 손쉽게 선정할 수 있는 장점을 가지고 있으며, 또한 광범위한 분야에서 사용되는 관계로 기술의 보급과 발전이 빠르다. 둘째, 각 객

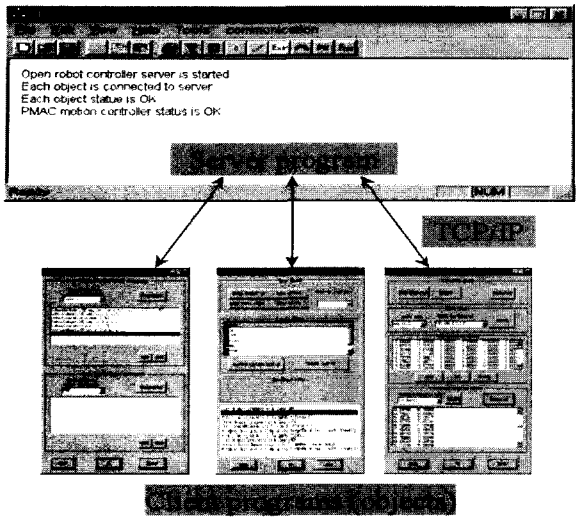


그림 4. 응용소프트웨어 모듈의 서버/클라이언트 구현.
Fig. 4. Client-server characterization of the application software module.

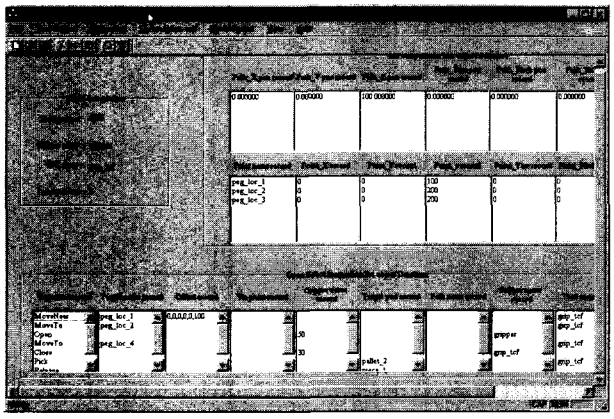


그림 5. PC-ORC의 환경 재설정 프로그램.
Fig. 5. A configuration editor of the PC-ORC.

체들이 서로 다른 컴퓨터에서 실행되는 분산처리 환경에서도 기존에 구축되어 있는 이더넷(ethernet)을 사용하여 객체간의 통신을 쉽게 구현할 수 있다. 그러나 이러한 객체들의 정보교환을 TCP/IP통신을 사용하여 구현하기 위해서는 각 객체들의 요구에 응답하고 서비스하는 서버프로그램이 필요하다. 그림 4는 이러한 서버프로그램과 다이얼로그 박스로 작성된 각 응용객체 (즉, 클라이언트)를 보여주고 있다. 또한 제안된 PC-ORC는 TCP/IP를 통하여 각 객체들의 속성을 재설정하는 것이 가능하도록 구성되었다. 그림 5는 이러한 환경설정편집기를 보이고 있다. 로봇 제어기의 분석과 객체 모델링을 통한 PC-ORC의 구조는 그림 6에 보이는 것과 같다. 이와 같이 구성된 각 객체들은 PC-ORC에서 정의된 프로세서 작업을 수행하고, 제어시스템의 구조를 변경 또는 추가할 때 유연성 및 확장성을 제공한다. 예를 들면, 기존의 로봇기구가 다른 형태의 로봇으로 대체될 경우 현재상태에서 환경재설정을 통하여 PMAC 과 CODE 시스템의 속성을 바꾸면 제안된 PC-ORC는 빠르게 새로운 로봇 시스템을 재구축하는 것이 가능하다.

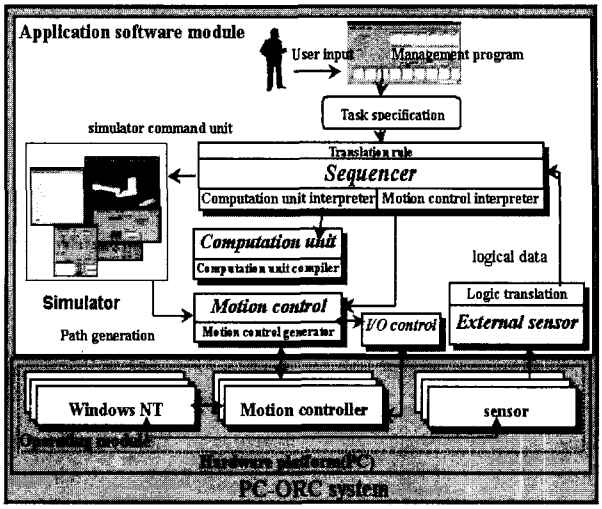


그림 6. PC-ORC의 구성요소
Fig. 6. Components of the PC-ORC.

IV. 개방형 로봇 프로그래밍

앞에서 모델링된 개방형 로봇제어시스템을 구동하기 위해서는 사용자로부터 입력된 모션정보가 필요하다. 일반적으로 사용자가 보내는 정보는 로봇의 모션프로그램의 형태로 전달된다. 본 절에서는 PC-ORC내에서 이러한 로봇 모션프로그램의 형태와 변환작업에 대해서 기술한다.

1. 개방형 명령어 (OR instruction set)

로봇모션을 프로그래밍하는 방법에는 온라인 방식의 교시상자를 이용한 "teach-and-playback" 방식과 오프라인 상태에서 로봇 전용언어를 이용한 "high level language programming" 방식으로 나눌 수가 있다[23]. 전자의 방식을 통한 로봇 모션 프로그래밍은 간단한 작업으로 프로그램을 작성할 수 있는 장점이 있지만, 운용자의 작업량이 많고 로봇모션을 프로그래밍하는 시간이 많이 걸리는 단점이 있다. 후자의 방식을 통한 로봇 모션 프로그램은 복잡한 작업을 원활하게 구성할 수 있는 장점이 있지만, 로봇시스템의 구조와 수행하고자하는 작업을 잘아는 숙련된 운용자가 필요하고, 다른 제조장비들과 상호작용이 어렵다는 단점이 있다. 이러한 단점들을 보완하기 위해서 본 논문에서는 새로운 개념의 OR Instruction Set(ORIS)를 제안한다. 이러한 ORIS는 PC-ORC내의 각 객체들에 명령을 하달하기 위한 모션 명령어의 집합이다. ORIS는 사용자와 대화식으로 사용할 수 있는 일상적 구조를 가진 자연스러운 언어체계로 이루어져 있으며, 다음과 같은 간단한 4개의 명령단위로 구성된다. 표 1은 ORIS의 대표적인 명령어 예들을 보인다.

① 모션제어 명령단위 : 모션제어 명령단위는 로봇구동에 관련된 구동명령어로 구성되어 있다. 이러한 구성명령어는 선형 보간운동(linear interpolated motion), 조인트 보간운동(joint interpolated motion), CP운동(continuous path motion), 말단효과장치 운동(gripper motion) 등으로 나누어진다.

② 환경제어 명령단위 : 환경제어 명령단위는 동적상태에서 작업조건에 따라 로봇환경을 규정하는 명령어들로 구성

표 1. OR Instruction set 명령어.

Table 1. OR Instruction set commands.

Command			Command unit
Move control command unit	Linear interpolated motion	MoveTo	MoveTo Tool(gripper_tcf) to TargetPoint(point_name)
		MoveNear	MoveNear Tool(gripper_tcf) relativeto TargetPoint(point_name) with Offset(x,y,x, roll, pitch, yaw)
		MovePath	MovePath Tool(gripper_tcf) according to Path(Path_name) with Offset(x, y, z, roll, pitch, yaw)
		MoveCircle	MoveCircle Tool(gripper_tcf) to TargetPoint(point_name)
	CP motion	Continous path genrtation	CP_Generation(Source, File_Name)
	Joint interpolated motion	MoveSingle Axis	MoveSingleAxis Joint(JointNum) with Const(value)
		MoveAllAxe x	MoveAllAxes Joints(Joint_1, Joint_2, Joint_3 ...) with Const(value_1, value_2, value_3 ....)
	Gripper motion	Open	Open Gripper(gripper_name) with Const(value)
		Close	Close Gripper(gripper_name) with Const(value)

Command		Command unit
Configuration control command unit	Set	Set Mode(joint/linear)
		Set JointSpeed(value)
		Set TipSpeed(value)
		Set ScrewSpeed(value)
		Set ScrewAccel(accel_value, decel_value)
		Set JointAccel(joint_num, accel_rise, accel_fall)
Logic device control command unit	Vision Operation Command	Board_Initalize(address, value)
		Capture_Image(value)
		Prefilter(value)
		Detect_boundary(value)
		Erosion_boundary(value)
		Save_Data(File_name)
	Send	Send Signal(signal_num, value)
	Get	Get Signal(signal_num)
	Wait	Wait Signal(signal_num, compare_value, timeout_value)
Program control command unit	OnState	OnState(condition) ~ stop/continue
	IfThen	IfThen(condition_compare) ~ (other command unit)
	While	While(condition) ~ exit

되어 있다. 이러한 로봇 환경명령어는 축의 이송속도, 최대/최소 속도 및 가속도, 로봇의 각종 모드등을 설정한다.

③ 논리장치제어 명령단위 : 논리장치제어 명령단위는 축의 모션과 동기적으로 실행되지 않는 센서나 액츄에이터의 신호를 입출력 인터페이스 카드의 I/O포트에 쓰거나 또는 입력되는 신호를 읽는 명령어로 구성되어 있다. 현재 본 논문에서는 이러한 명령단위는 주로 비젼시스템 구동을 위한 명령어들로 이루어져 있다.

④ 프로그램제어 명령단위 : 프로그램제어 명령단위는 로봇의 구동시 발생하는 돌발적인 상황이나 예외적 조건을 처리하기 위한 조건형식의 명령어로 이루어져 있다. 명령어 구조는 컴파일러를 가진 일반적인 언어 프로그램에 있는 조건부 함수와 비슷하다.

제안된 ORIS는 객체지향적 특성을 가지기 때문에 필요에 따라서 명령어의 추가/삭제가 가능하고, 새로운 형태의 명령어 단위로 대체될 수 있다. 또한 폐쇄적인 로봇 명령어와는 다르게 다른 하드웨어 장치들과 통합 작업을 할 경우 작업자가 논리적인 순서에 따라서 통합제어프로그램을 쉽게 구성할 수 있다. 그러나, 명령어를 다양화하고, 효율성을 개선하기 위해 추가적인 연구가 필요하다. 그림 7은 C++상에서 구현된 ORIS 편집기를 나타낸다. ORIS 편집기는 ORIS에서 제시한 4가지 명령단위의 명령어들을 상위구조에서부터 하위구조의 계층적 구조형식으로 나열한 일련의 리스트 박스와 모션프로그램을 구성할 때 작성된 결과를 운용자에게 보여주는 윈도우 창으로 구성되어 있다. 또한, 구성된 일련의 모션프로그램을 저장하거나 또는 기존에 저장된 모션프로그램을 여는 파일열기 및 저장버튼, 파일열기를 선택할 경우 이에 대한 경로를 나타내는 편집상자, 이미 ORIS의 문법에 익숙한 경우 단지 일련의 명령어를 입력하여 구성할 수 있는 라인 명령어 편집박스로 구성되어 있다. ORIS 편집기에서 모션프로그램 작업은 명령단위 내에 정의된 명령어에 의해 구성된 하부 윈도우를 통해 가능하다. 즉, 하부 윈도우를 통한 ORIS 편집기의 모션프로그램 구성은 먼저 사용자가 구성하고자 하는 명령어를 명령단위의 분류에 근거하여 선택한 다음 마우스로 더블 클릭하여 변수입력의 환경을 제공하는 하부윈도우를 생성시킨다. 그리고 생성된 하부윈도우의 변수값을 입력한 다음 확인버튼을 눌러 하부 윈도우의 객체가 소멸할 때 입력된 변수값에 근거하여 구성된 메커니즘에 의해 ORIS의 문법형식으로 전환되어 ORIS 편집기 내의 명령어 윈도우에 표시된다. 운용자는 이러한 GUI환경의 ORIS 편집기를 사용하여 쉽게 원하는 작업에 대한 계획과 동시에 통합작업 프로그램을 구성할 수 있어 운용자의 편의를 도모한다.

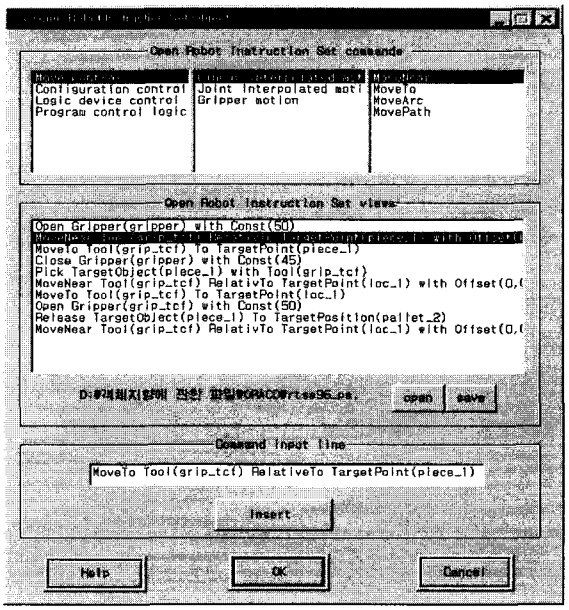


그림 7. 모션프로그램 편집기의 대화상자.

Fig. 7. A dialog box of the motion program editor.

2. ORIS의 변환작업

ORIS의 변환작업은 ORIS 문법으로 구성된 모션프로그램을 PC-ORC내에 설치된 각 응용 객체들이 식별할 수 있는 형태의 명령어로 변환하는 작업을 나타낸다. 변환작업의 순서는 다음과 같다. 먼저 운용자로부터 입력된 모션프로그램을 앞에서 정의된 명령단위로 분류하여 구축된 데이터베이스의 각 레코드 단위의 같은 필드 상에 저장된다. 이와 같이 저장된 ORIS를 사용자로부터 정해진

변환법칙에 근거하여 미리 구성된 라이브러리들, 또는 CODE API 및 PMAC 모션명령어로 변환한다. 변환된 명령어들은 PC-ORC 내의 각 객체들을 실제적으로 구동한다. 이와 같은 중립언어를 두는 목적은 PC-ORC 내의 각 객체들이 필요에 따라서 추가/삭제 되더라도 전체제어시스템의 개방성 및 독립성을 보장하기 위해서이다. 그림 11은 이러한 변환작업을 보여주고 있다. 이러한 모션 프로그램 변환작업은 ORIS편집기에 내장된 변환기 (Computa-tionUnitConversioner객체, MotionControlConversioner객체)를 통하여 이루어진다.

V. 산업용 스카라로봇에 PC-ORC 환경의 적용

본 논문에서는 제안된 PC-ORC의 적용성을 검토하기 위해서 현재 생산현장에서 사용되고 있는 스카라로봇을 PC에 기반을 둔 개방구조 로봇제어시스템으로 재구성하고자 한다. PC-ORC의 적용성은 크게 두 가지 측면에서 고려되어졌다. 첫번째 측면은 구현된 PC-ORC의 구성요소들을 임의로 변경 및 확장하는 것이 가능한지를 조사하기 위해서 기존의 PC-ORC 로봇제어시스템에 비젼시스템을 추가하여 PC-ORC가 쉽게 변경이 가능한지를 조사한다. 두번째 측면은 구현된 PC-ORC의 실시간제어 가능성 및 적합성을 조사하기 위해서 제어성능을 평가한다.

1. 하드웨어 구성

그림 8에서 점선으로 표시된 부분은 제안된 PC-ORC의 하드웨어 구성을 나타낸다. 이러한 PC-ORC는 하드웨어 플랫폼인 PC, 서보제어기 PMAC 모션제어기, 외부 센서제어기인 비젼보더등의 하드웨어로 이루어져 있다. 이와 같은 PC-ORC에 산업현장에서 사용되고 있는 스카라로봇을 부착함으로서 PC-ORC의 적용성 및 유연성을 검토한다. 그림 9는 구성된 실험장비들을 보이고 있다.

2. 소프트웨어 구성

2.1 환경 재설정 및 시스템 변경과 모션프로그래밍

PC-ORC의 개방성 및 객체지향성을 평가하기 위해서 환경설정 편집기를 사용하여 각 객체들의 속성들을 스카라 로봇시스템에 적합하도록 재설정하고, 비젼시스템의 하드웨어와 프로그램 및 구동명령어를 PC-ORC내에 새롭게 추가한다. 이러한 객체속성의 재설정 및 시스템의 변경 작업은 로봇시스템의 서보드라이브 및 기계장치들과 각 부속장치들의 하드웨어사양을 참고한다. 이러한 환경재설정 및 시스템 변경을 통하여 PC-ORC는 다양한 형태의 로봇시스템 및 생산환경에 적용될 수 있다. 또한 이와 같이 구축된 통합제어시스템의 성능을 평가하기 위해서 다음과 같은 작업들을 수행한다. 우선, PC는 PC-ORC에 새롭게 추가된 비젼시스템에서 입력된 화상정보를 신호처리를 거쳐 메모리로 읽어들이고, 입력된 알고리즘을 통하여 로봇의 궤적선을 추출한다. 또한, PC는 추출된 궤적선을 바탕으로 적절한 좌표변환을 거쳐 로봇의 궤

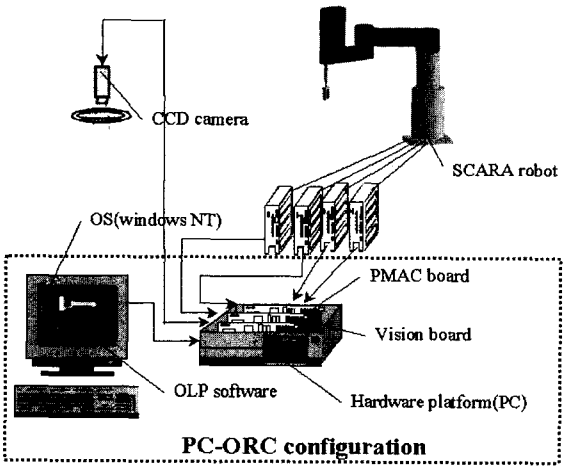


그림 8. 스카라로봇에 적용된 PC-ORC의 하드웨어구성.
Fig. 8. Hardware configuration of the PC-ORC applied to the SCARA robot.

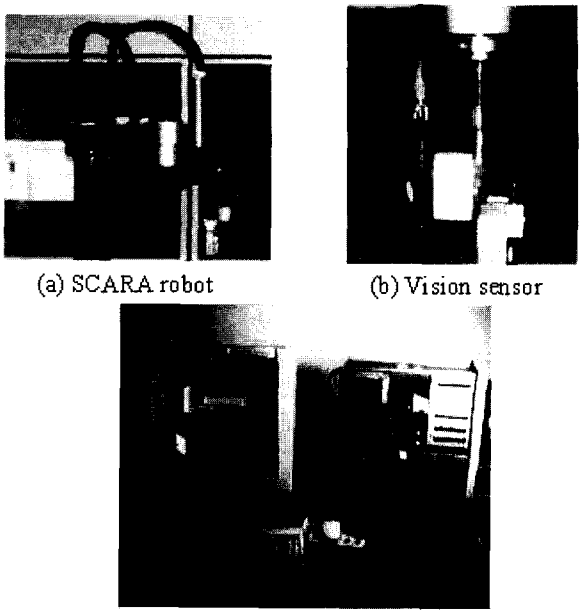


그림 9. 스카라로봇에 적용된 PC-ORC의 실험장비.
Fig. 9. An experimental setup of the PC-ORC applied to the SCARA robot.

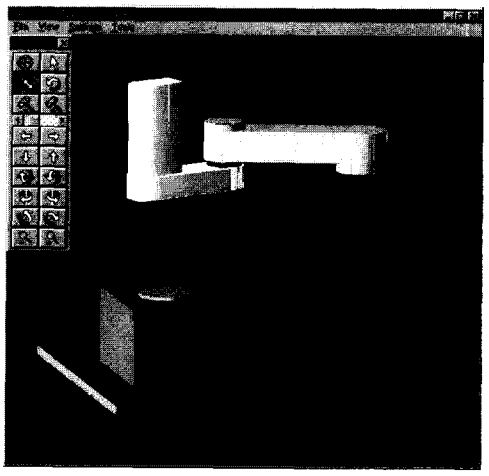


그림 10. CODE상에서 구현된 가상적인 스카라로봇.
Fig. 10. A virtual SCARA robot configured with CODE.

적을 생성한다. 이와 같이 획득된 궤적정보를 바탕으로 CODE의 가상 시뮬레이션 환경에서 로봇의 궤적을 수정/보완하여 최종적인 데이터파일을 작성한다. PC는 이와 같이 작성된 데이터파일을 설치된 Dual-port RAM을 사용하여 PMAC 메모리 버퍼로 다운로드한다. 버퍼에 저장된 궤적정보는 주기적인 서보루프에 따라 PMAC상에서 제어가 수행된다. 또한, PMAC은 수행된 결과값을 PC로 전달한다. 이와 동시에 Windows NT에서는 다음의 로봇 운동계획을 수립한다. 이러한 일련의 작업들을 수행하기 위해 ORIS를 사용하여 통합제어 프로그램을 작성하면 그림 11과 같다. 이와 같이 작성된 프로그램은 ORIS 편집기에 내장되어 있는 변환기를 통하여 각 객체들이 실행할 수 있는 형태의 언어로 변환된다.

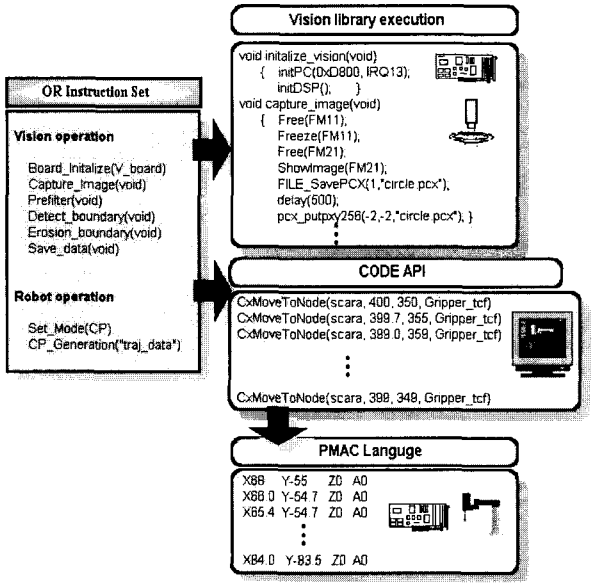


그림 11. OR Instruction Set을 사용한 모션프로그램.

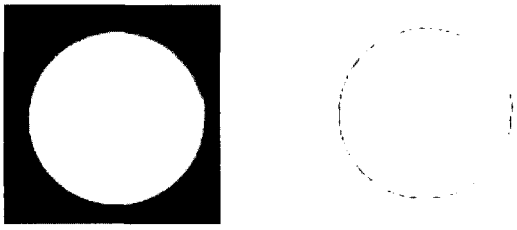
Fig. 11. An example of motion programming using the OR Instruction Set.

2.2 CODE를 사용한 가상 시뮬레이션

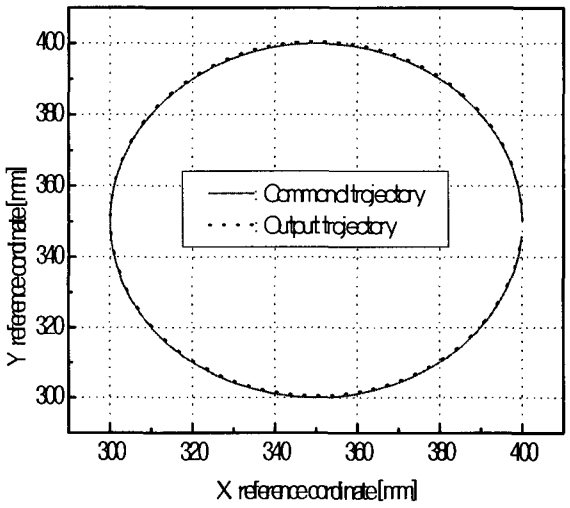
일반적으로 OLP소프트웨어는 컴퓨터에서 3차원 그래픽으로 가상적인 로봇과 작업환경을 구현하여 교시, 궤적계획, 시뮬레이션, 성능평가 등의 일련의 작업을 오프라인 상태에서 수행한다. 이러한 OLP기능은 제안된 PC-ORC에서는 상용의 CODE 시스템에서 제공된다. 먼저 CODE상에서 가상적인 로봇 및 작업공간을 모델링한다. 이러한 가상로봇을 통하여 사용자에 의해서 주어진 로봇 모션을 CODE의 3차원 그래픽상에서 가상적으로 시뮬레이션한다. 또한 이러한 시뮬레이션을 통하여 로봇의 궤적을 수정/보완하여 데이터파일을 작성하고 이것을 PMAC으로 전송한다. 그림 10은 CODE상에서 구현된 가상적인 스카라 로봇을 보이고 있다.

2.3 성능 평가

제안된 PC-ORC는 폐쇄형 구조로 되어있는 산업용 스카라로봇의 제어시스템을 개방형으로 대체할 수 있었으며, 새롭게 추가된 비젼시스템과 통합환경에서도 비교적 잘 작동하였다. 그림 12는 비젼시스템과 로봇제어시스템의 통합환경에서 PC-ORC의 성능을 나타낸다. (a)는 비젼센서로부터 획득된 화상정보를 나타내고, (b)는 화상정보에서 추출된 명령궤적을 나타낸다. 또한 (c)는 스카라로봇에서 실행되어 출력된 궤적을 보여주고 있다. 또한, 그림 13와 그림 14은 각각 스카라로봇의 1축과 2축의 궤적



(a) An image obtained from the CCD camera (b) The boundary extracted from the original image.



(c) The results executed on the PC-ORC.

그림 12. PC-ORC상에서 궤적의 추출과정 및 실행결과.

Fig. 12. Trajectory extraction process and the results executed with the PC-ORC.

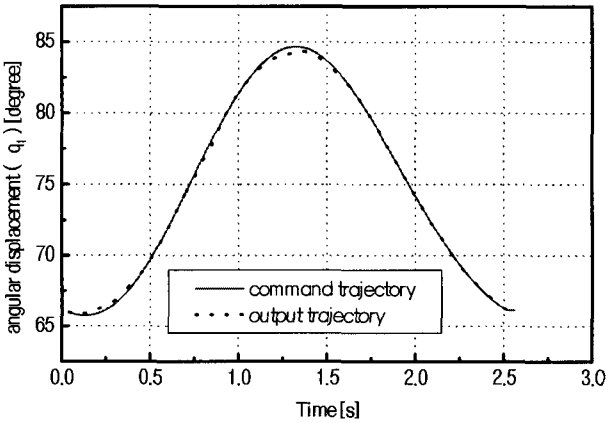


그림 13. 스카라로봇 제1축의 명령추종성능.

Fig. 13 The tracking performance of the first axis of the SCARA robot.

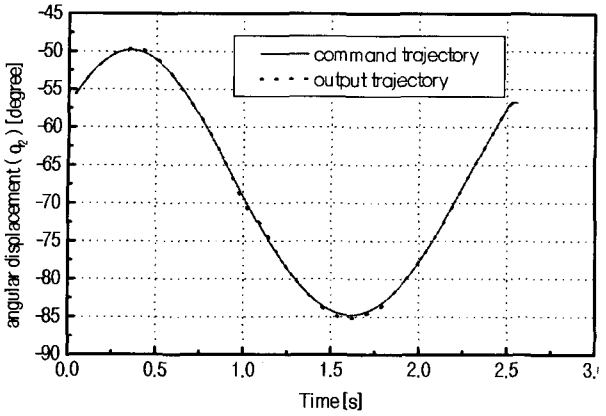


그림 14. 스카라로봇 제2축의 명령추종성능.

Fig. 14. The tracking performance of the second axis of the SCARA robot.

추종성능을 나타내고 있다. 이러한 결과는 PC-ORC가 부분적이지만 실시간 제어가 가능하고, 실제 산업용 로봇시스템의 제어에 적용될 수 있다는 것을 보여주고 있다. 또한, 컴퓨터상에서 구현된 가상시뮬레이션 환경을 활용함으로써 작업의 효율성을 높이고 새로운 생산환경에도 빠르게 적응할 수 있을 것으로 평가된다.

VI. 결론

본 논문에서는 기존의 산업용 로봇제어시스템의 제어기능들을 하나의 환경으로 통합하여 로봇 태스크작업을 수행할 수 있는 개방환경의 PC-ORC를 제시하였고, 그 적용성을 검토하였다.

PC-ORC는 표준 PC플랫폼에 상용의 소프트웨어와 객체로 구현된 응용소프트웨어를 탑재하는 구조로 이루어져 있다. 이러한 구조는 기존의 폐쇄형 로봇구조에 비하여 많은 장점들을 가진다. 이러한 PC-ORC의 장점은 기존의 로봇 제어시스템의 구성과는 달리 기본 하드웨어 플랫폼으로 통합성과 확장성을 가진 PC를 기반으로 하고, 소프트웨어적으로 객체지향방법에 근거하여 정의한 모듈과 객체로 구성되어 있기 때문에 로봇제어시스템의 재설계시 빠른 개발환경을 제공할 수 있다. 그리고 소프트웨어 및 하드웨어의 재사용 및 통합성을 제공한다. 그러나, 제안된 객체형의 개방구조 로봇시스템은 하드웨어인 모션 제어기 카드와 외부센서 제어기카드가 PC를 기반으로 하고 있어 아직까지는 폐쇄형 제어기에 비해서 안정되고 강인한 특성을 가지지 못하고 있다. 또한, 시스템의 효율성 측면에서는 제안된 PC-ORC는 여러가지 문제점들을 나타내고 있다. 향후 안정되고 실시간처리가 가능한 운영체제가 개발되고, 시스템의 효율성을 향상시킨다면 이러한 개방형 로봇제어시스템이 각 산업현장의 로봇제어시스템에 충분히 적용 가능할 것으로 기대된다.

참고문헌

[1] W. Sperling and P. Lulz, "Enabling open control systems: An introduction to the OSACA system platform," *ESPRIT III Project: Stuttgart: FISW GmbH*, 1995.

[2] P. K. Wright, "Principles of open-architecture manufacturing," *Journal of Manufacturing Systems*, vol. 14, no. 3, pp. 187-202, 1995.

[3] J. P. T. Mo, Y. Wang, and C.K. Tang, "The use of the virtual manufacturing device in the manufacturing message specification protocol for robot task control," *Computers in Industry*, vol. 28, pp. 123-136, 1996.

[4] OMAC API Work Group, *OMAC API SET-Working Document*, OMAC API Work Group, Ver. 0.18, 1998.

[5] OSEC, *Open Systems Environment for Controller*, http://www.sml.co.jp/OSEC.

[6] OSACA, *Open System Architecture for Controls within Automation Systems*, ESPRIT III Project 6379/9115, 1996.

[7] G. C. Burdea, "Invited review: The synergy between virtual reality and robotics," *IEEE Trans. Robotics and Automation*, vol. 15, no. 3, pp. 400-410, 1999.

[8] E. Freund and J. Rossmann, "Projective virtual reality: Bridging the gap between virtual reality and robotics," *IEEE Trans. Robotics and Automation*, vol. 15, no. 3, pp. 411-421, 1999.

[9] 이재영, 권욱현, 박재현, "머시닝 시스템의 실시간 제어를 위한 개방형 구조 제어기," *Proc. of the 11th KACC*, pp. 1324-1327, 1996.

[10] 이낙형, 서동수, 엄광식, 서일홍, 여인택, 김성락, "로봇시스템을 위한 PC기반 개방형 제어기 설계," *Proc. of the 13th KACC*, pp. 430-433, 1998.

[11] 김호철, 홍금식, 최경현, "객체구조에 근거한 개방구조 로봇시스템," *Proc. of the 13th KACC*, pp. 416-419, 1998.

[12] 최명수, 김동범, 이은정, 한종희, 홍용준, 황태인, 최정연, "산업용 시스템을 위한 개방형 시스템 구조 설계 및 적용," *Proc. of the 13th KACC*, pp. 434-437, 1998.

[13] 추성호, 박홍성, 강원준, "개방형 구조 제어 시스템을 위한 객체 에이전트 연구," *Proc. of the 14th KACC*, vol. A, pp. 21-24, 1999.

[14] M. Babb, "PCs: The foundation of open architecture control system," *Control Engineering*, pp. 75-82, January 1996.

[15] R. D. Klafter, T. A. Chmielewski, and M. Negin, *Robotic Engineering: An Integrated Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[16] T. Julian, *Beginning Windows NT Programming*, 정보문화사, 1999.

[17] PMAC, *PMAC User's Manual*, Delta Tau Inc., 1996.

[18] E. Coste-Maniere and D. Simon, "Reactive objects in a task level open controller," *IEEE Int. Conf. on Robotics and Automation*, pp. 2732-2737, Nice, France, 1992.

[19] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy,

and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[20] 이상엽, *Visual C++ Programming Bible Ver 5.x*, 영진출판사, 1997
[21] 곽준기, 김소영, *Inside Screat Visual C++ 6.0*, 삼각형출판사, 1999.
[22] CODE, *Getting started with CODE for Windows NT*, Cimetrix Inc., 1996.
[23] H. P. Hwang and C. S. Ho, "Knowledge task-level programming and execution environment for robots," *Robotics & Computer-Integrated Manufacturing*, vol. 12, no. 4, pp. 329-351, 1996.
[24] C. P. Jobling, P. W. Grant, H. A. Barker, and P. Townsend, "Object-oriented programming in control system design: a survey," *Automatica*, vol. 30, no. 8, pp. 1221-1261, 1994
[25] 김선호, 김동훈, 박경택, "생산장비 객체화와 개방형 가공 셀 구축 연구(I) -생산장비 객체화- ," 한국정밀공학회지, vol. 16, no. 5, pp. 91-97, 1999.
[26] 여인택, 김성락, 조용중, "PC 기반 로봇제어기 사례비교," 대한기계학회 동역학 및 제어 - 생산 및 설계공학 공동학술대회 논문집, 한국과학기술원(KAIST), pp. 151-158, 1999.
[27] 추영열, 김용수, "PC 기반 PLC 시스템의 제철공정 적용성 연구," 대한기계학회 동역학 및 제어-생산 및 설계공학 공동학술대회 논문집, 한국과학기술원(KAIST), pp. 164-169, 1999.
[28] 최명수, 최정연, 황태인, 홍용준, 한종희, 윤병운, 고원준, "개체기반 언어의 Robot Programming 적용," 동역학 및 제어-생산 및 설계공학 공동학술대회 논문집 한국과학기술원(KAIST), pp. 145-150, 1999.

김 점 구

1973년 11월 3일생. 1999년 부산대학교 기계공학부 (공학사). 1999년~현재 부산대학교 대학원 지능기계공학과 재학중. 관심분야는 개방형 로봇제어, 생산자동화 및 계측제어, 시스템 규명 및 견실제어.

홍 금 식

1957년 8월 25일생. 1975-1979 서울대학교 기계설계학과 (공학사). 1985-1986 Columbia University (New York) 기계공학과(공학석사). 1987-1991 University of Illinois at Urbana-Champaign (UIUC) 응용수학과(이학석사) 및 기계공학과(공학박사). 1991-1992 UIUC 기계공학과 Post-doctral fellow. 1979-1982 군복무(포병). 1982- 1985 대우중공업(인천) 기술연구소 연구원. 1993년-현재 부산대학교 공과대학 전임강사, 조교수, 부교수. 관심분야는 시스템이론, 적응제어, 비선형제어, 분포계수시스템의 제어.

최 경 현

1960년 4월 2일생. 1986년 부산대학교 공과대학 기계공학과 (공학사). 1988-1990 부산대학교(공학석사). 1990-1995 university of Ottawa(공학박사). 1986-1988 현대자동차 제품개발연구소 근무, 1996-1997 한국원자력연구소 선임연구원. 1997-1998 부산대학교 기계공학부 기금교수, 1999-현재 제주대학교 기계에너지생산공학부 조교수. 관심분야는 첨단생산자동화 및 시스템.