Proceedings of the 13th
KACC, October 1998

로봇 시스템을 위한 PC 기반 개방형 제어기 설계

A Design Experience on PC-Based Open Architectured Controller for Robot Systems

°이낙형*, 서동수*, 엄광식*, 서일홍*, 여인택**, 김성락**

*한양대학교 전자공학과 (Tel : +82-345-408-5802; Fax : +82-345-408-803;E-mail : nhlee@scorpio.hanyang.ac.kr)

**현대 중공업 (Tel : +82-331-289-5173; Fax : +82-345-408-5803; E-mail : ksr@hhi.co.kr)

Abstracts : This paper describes the design and implementation of a PC-Based open architectured robot control systems to be flexible and adaptable to various software and hardware requirements. To design a flexible and adaptable software structure, various modular API sets are implemented. To use commercial Windows NT as operating system of robot controller, characteristics of RTOS(Real Time Operating System) are examined for Windows NT. Several necessary tasks for robot control system are designed and then, the performance of designed system is analyzed by using Timed-Petri Nets.

Keywords : Open Architecture, API sets, Windows NT, RTOS, Timed-Petri Nets

1. 서론

산업화가 가속화 되어 자동화 및 고정밀도의 로봇에 대한 수요가 급증함에 따라 로봇 제어기에 대한 활발한 연구가 진행되었다. 이에 따라 우수한 성능을 지닌 많은 제어기들이 개발되었다. 하지만 이러한 제어기의 가장 큰 문제점은 독점적인 제어 기술이라는 것이다. 비록 이러한 독점적인 제어기술이 신뢰할 수가 있고 응용 시에 요구 사항을 충족시킬 수는 있지만 제어기가 판매상 위주로 구조화 되어서 높은 의존성을 지니게 되는 문제점을 야기시켰다. 이러한 문제점을 해결하기 위해서 본 논문에서는 동일한 설비 투자에 대해서 연구 결과를 극대화할 수가 있는 개방 구조형 제어기를 제안한다. 이는 기존의 제어기가 통합 및 확장 시에 많은 비용과 노력의 재 투자가 필요하였지만 제어기의 구조를 개방함으로써 사용자의 적은 노력에도 다양한 목적으로 제어기의 변경이 용이하도록 설계한다. 이에 대한 연구는 1988년 Greenfeld에 의해 개방 구조형 제조(Open Architecture Manufacturing)[7]에 대한 개념이 소개 된 이후에 많은 연구가 진행되었으며 대표적인 예로는 NGC-SOSAS 와 개방 구조 제어기의 표준화에 관한 연구를 한 유럽연합의 OSACA[6], 일본의 OSEC 등이 있고, 관련 제조 업체로는 Cimetrix[4]와 Kuka 등이 있다. 이와 같은 추세에 따라 개방 구조형 제어기의 하드웨어와 소프트웨어의 구조적 모델을 제시한다. 하드웨어적인 측면에서 다양한 하드웨어에 대한 적응성이 뛰어나고 값이 싸며 범용적으로 사용되고 있는 PC를 이용하여서 PC가 가진 장점을 이용하고 소프트웨어적인 측면에서는 프로그램을 모듈화하고 수정이나 확장성을 가지게 설계하였다.

로봇 시스템은 경성 실시간 시스템(Hard Real-Time System)이다[1,2]. 즉, 일정한 샘플링 시간 내에 계획된 궤적 값을 제어기로 내주어야 하며 시스템의 고장진단과 사용자 인터페이스 등의 지원이 필요하다. 본 논문에서는 로봇 제어 시스템의 운영체제로 일반적으로 사용되는 실시간 운영체제(RTOS)를 사용하지 않고 친숙한 사용자 인터페이스를 제공하고 차세대 PC 운영체제로 널리 사용되고있는 Windows NT를 이용하여 구현하고, 이에 대한 타당성과 적합성을 고찰한다. 제안한 로봇 제어 소프트웨어가 보다 지능적으로 외부 상황에 대처하도록 하기 위해서 로봇 제어에 필요한 기본적인 태스크를 정의하고 스케쥴링(Scheduling)을 통하여 제어 시스템의 안전성을 보인다.

2. 개방 구조형 로봇 제어기를 위한 OS

2.1 실시간 운영 체제의 개요

실시간 로봇제어 시스템을 분석하기 위해서 먼저 포괄적인 실시간 시스템에 대한 기본 개념을 정리해 본다. 실시간 시스템이란 어떤 작업을 수행할 때, 요구되어지는 시간(Deadline) 안에 정확하게 결과를 만들어내고 동시에 불규칙적인 외부 입력에 대하여 시간적으로 예견되는(Consistent time) 시스템을 말한다. 실시간 시스템에는 두 가지 유형이 있는데, 첫째는, 연성 실시간 시스템(Soft real-time system)으로서 작업 결과가 되도록 빠른 시간 안에 수행될 수 있는 시스템 이지만 여기에는 특정한 시간 이라는 의미는 없다. 둘째로, 경성 실시간 시스템(Hard real-time system)은 앞에서 정의한 바와 같이 특정한 시간 안에 작업 결과가 유출될 수 있는 시스템을 일컫는다. 실시간 시스템이 정해진 시간 안에 정확한 결과를 내기 위해서는 충분한 연산능력과 빠른 I/O, 시스템의 신뢰성(Reliability)과 강인성(Robustness)이 필요하다. 그리고 실시간 운영 체계를 위한 필요 조건으로 다음과 같은 사항들을 고려 해야 한다.

- 실시간 운영 체제는 멀티스레드(Multithread)를 지원 하고 선점형(Preemptive)이어야 한다.
- 스레드의 우선 순위를 할당할 수 있어야 한다.
- 예측 가능한 스레드 동기화 체계를 제공해야 한다.
- 우선 순위에 대한 상속 체계가 존재해야 한다.

2.2 실시간 운영 체제로서의 windows NT

실시간 운영 체제로서의 windows NT의 특징과 적합성을 알아 보기위해 우선 windows NT의 실시간 운영 체제로서의 특징

들을 살펴본다.

windows NT 는 멀티 스레드와 스케쥴링 방식에 있어서 다중 스레드를 생성하도록 지원하며, 스케줄링 방식이 선점형이다. 그리고 실시간 운영 체제에서는 모든 태스크가 동일한 우선 순위를 가져서는 안되고, 우선 순위를 필요에 따라 할당해 주어야 한다. Windows NT 는 스레드 우선순위 부여에 관한 실시간 운영 체제의 조건을 만족한다. 그러나, 일반 실시간 운영 체제가 256 개의 우선순위를 제공하는 반면 Windows NT 는 최대치가 7 개라는 단점이 있다. Windows NT 상에서 프로그램이 하드웨어에 접근하기 위한 유일한 방법은 커널 디바이스 드라이버를 통해서 이다. 외부의 인터럽트에 적절히 반응하기 위해서 개발자는 커널 디바이스 드라이버를 작성해 주어야 한다. 디바이스 드라이버가 하드웨어로부터 생성된 인터럽트를 핸들링하기 위해서, Windows NT 는 ISR(Interrupt Service Routine)에 의해 짧은 시간 동안 서비스하고 DPC(Deferred Procedure Call)에 의해 나머지 비교적 긴 시간을 요하는 일을 처리한다. 그러나 DPC 자체로는 비선점형 스케줄링 방식이라는 단점이 있다. 즉, DPC 는 그것을 발생시키는 인터럽트의 우선 순위와 상관없이 모두 동일한 순위로 다루어지고 단지 ISR 만이 선점형 특성을 지니기 때문에 응용 프로그램에 따라서는 다른 디바이스 드라이버에 의한 영향을 받을 수 있으며, 특히 DPC 처리 시간이 긴 하드디스크나 네트워크 드라이버에 의한 수 밀리 초의 지연을 야기시킬 수도 있다.

2.3 로봇 제어기 운영 체제로서 Windows NT 응용

Windows NT 의 여러 가지 문제점으로 인해 그 자체를 로봇 제어용 실시간 운영체제로 사용하기는 어려움이 있다. 예를 들면 서보의 디지털 제어를 위해서는 반듯이 주기적인 서보 루프가 보장되어야 하는데 수 밀리의 서보 루프를 Windows NT 에서 보장하기는 어렵다. 이에 대한 대안으로 주기적인 빠른 시간의 응답을 요구하는 서보계는 전용 서보 보드의 제어계를 이용하고, 로봇 동작 계획등과 같이 비교적 긴 시간의 응답이 요구되는 부분을 Windows NT 가 담당하도록 설계하였다. 즉, 서보계의 전류, 위치 제어 루프는 전용 DSP 보드가 그 기능을 담당하고 로봇의 동작 계획과 사용자 인터페이스 등과 같이 Windows NT 의 장점을 최대한 이용할 수 있는 부분은 PC 가 담당한다. 이에 따라 PC 에서 계획된 궤적정보와 제어기로부터의 엔코더 데이터나 센서 데이터를 원하는 주기로 정확하게 주고 받을 수 있어야 한다. 이를 위하여 본 논문에서는 타이머 인터럽트를 사용하였다. Windows NT 에서 타이머 인터럽트를 발생시키는 방법은 사용자 타이머, 멀티미디어 타이머, Waitable 타이머, 외부 인터럽트를 이용하는 방법이 있다. 이중 앞선 3 가지 방법은 Windows NT 의 특성상 정확한 시간을 보장하기 어려운 면이 있다. 그러나 Windows NT 에서 CPU 가 수행되는 임의의 스레드나 DPC 루틴을 언제라도 선점할 수 있는 방법은 외부 인터럽트를 발생시키고, 발생된 인터럽트에 분기하는 것이며, 이는 시간의 중요도가 높은 명령 전달의 기능을 구현하기에는 적합하다. 로봇 명령 전달 부분은 디바이스 드라이버의 ISR 에서 구현한다.

3. 개방 구조형 로봇 제어기의 하드웨어 구성

본 논문에서 제안한 로봇 제어기의 하드웨어 구조를 그림 1 에 나타내었다. 일반 PC 를 플랫폼으로 하고, 표준 OS(Windows NT)에서 동작하기 때문에 가격과 성능면에서 많은 이점이 있고 조작하기가 쉽다. 또한 상용 PC 를 이용한 표준 인터페이스 방식이기 때문에 제어기와 관련된 주변 장치들을 응용작업에 적합하게 재구성 할 수 있는 장점이 있다.

제안한 개방 구조형 로봇 제어기 시스템 사양은 다음과 같다.

Processor: Intel Pentium 200Mhz
Operating System: Windows NT 4.0
Bus Structure: Standard ISA(9)/PCI(4) Bus
Standard RAM: 32MB(128MB)
Standard Mass Storage: 1GB Hard Disk SCSI
Motion Board(2): DSP Board(3 축 당 1 개)
External Connection: Ethernet, 2 RS-232 Serial Ports, Mouse & Keyboard, External Floppy
Force/Vision Control: Vision Board, Force I/F Card

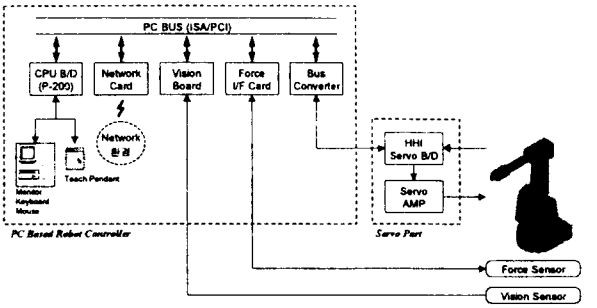

그림 1. 개방 구조형 로봇 제어기 H/W Configuration

Fig 1. H/W Configuration of Open Architecture Robot Controller

제어기의 상위 계층은 표준 PC 환경에 다양한 센서 인터페이스 보드를 장착할 수 있도록 되어있으며, 사용자 인터페이스(GUI)와 로봇 제어 소프트웨어가 상주하고, 20ms 마다 계산된 궤적 값을 Bus Converter 를 통해 제어기로 전달한다. 하위 계층은 전용 서보 보드로 구성되어 지령치를 받아서 1ms 마다 로봇을 제어한다.

4. 개방 구조형 로봇 제어기의 소프트웨어 구성

본 논문에서 제안할 개방 구조형 로봇 제어기의 소프트웨어 구성을 그림 2 에 나타내었다. 전체 제어기를 구성하는 소프트웨어를 독립된 기능별로 나누고 하드웨어와의 연관성을 고려하여 개방 구조형 로봇 제어기에 필요한 다양한 요소들을 모듈화 하여 설계 함으로서 각각의 프로그램들이 독립적으로 수행될 수 있고 사용자의 의도에 맞게 추가와 수정이 용이하도록 구성하였다[6]. 이를 위한 방법으로 각각의 API 들을 DLL(Dynamic link library)로 구현하였고, Robot Motion API 모듈은 로봇동작이나 연산에 관계된 다양한 함수들을 라이브러리(Library)화 하였다.

각 API 모듈의 기능은 다음과 같다.

- Real Time Kernel: 제어기 S/W 시스템에서 가장 하부구조로서 태스크 및 스레드 관리, 프로세스간의 통신 및 동기화, 메모리 관리, 시스템 운영 중 발생하는 각종 인터럽트 처리를 담당.
- Real Time Data Base: 로봇 제어기의 정보기반이 시스템의 실시간 요구 사항에 대처하도록 설계되고, 정보기반 인터페이스가 데이터의 공유와 다른 시스템 요소로부터 갱신되도록 설계.
- User Interface API: 사용자가 쉽게 응용프로그램에 대한 GUI 를 구성할 수 있도록 표준 인터페이스를 제공.
- GUI/Windows API: 사용자가 제어기와 인터페이스 하기위한 부분으로 Visual C++ 5.0 을 이용하여 구성.
- Language API: 새로운 로봇 언어를 정의하고 등록된 명령어에 대해 해당 함수를 호출.
- Robot Type API: 새로운 로봇을 등록하고 이에 대한 정보를 이용하여 대상 로봇에 관한 기구학과 역기구학을 생성.
- Robot Motion API: 사용자가 요구하는 동작을 로봇이 수행하도록 하기 위해서 로봇 동작에 관련된 일련의 함수를 library 형식으로 제공.
- Sensor System API: Vision 과 Force Control 등과 같이 Sensing

Interface 을 이용한 제어 시스템의 구축을 위한 부분으로 대상 센서의 구조와 초기화 그리고 Calibration 에 대한 기능을 제공.

- Network API: 네트워킹에 관련된 일련의 작업을 처리.
- Digital I/O: 범용의 다양한 I/O 시스템과 인터페이스하는 기능을 제공하며, 제어기의 다른 부분과 실시간으로 데이터를 주고받고 GUI 환경에서 Editing 되고 Display 됨.

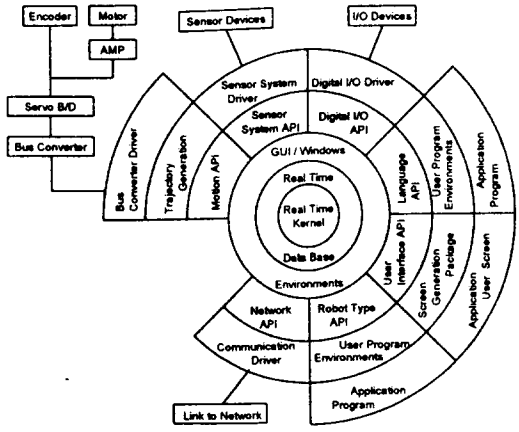

그림 2. 개방 구조형 로봇 제어기의 S/W Configuration

Fig 2. S/W Configuration of Open Architecture Robot Controller

이상과 같이 로봇 제어 시스템을 기능별로 모듈화 하여 구성함으로서, 구조적 측면에서 개방성과 효율성을 부여하였다. 그리고 Language API 에서 사용되어지는 언어는 ISO 표준 언어를 근간으로 한다. 그림 3 에 사용자 인터페이스를 나타내었다.

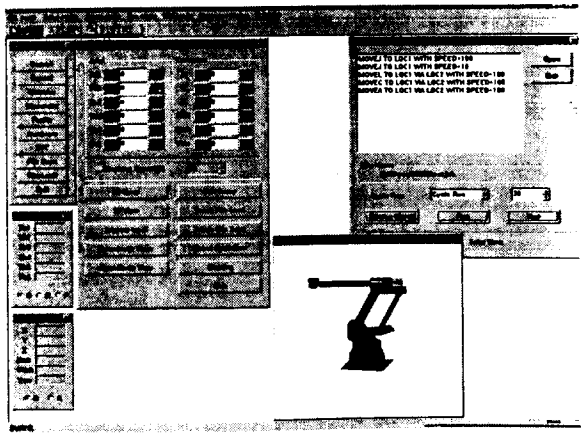

그림 3. 개방 구조형 로봇 제어기의 사용자 인터페이스

Fig 3. GUI of Open Architecture Robot Controller

5. 성능 분석

로봇 제어 시스템은 경성 실시간 시스템이다. 따라서, 제어 시스템의 올바른 동작을 보장하기 위해서 로봇 제어 프로그램은 그 산출결과의 논리적 정확성 뿐만 아니라 결과가 산출되는 시간이 중요하게 고려되어져야 한다. 즉, 모든 Time Critical Tasks 들은 결과 산출 값이 유효한 시간(Deadline) 안에 완료되어야 한다. 예를 들면, 센서 데이터 획득(Sensory Acquisition)과 같은 태스크는 센서 타입과 매 주기마다 처리되어야 할 정보의 양에 따른 특정한 샘플링 주기에 수행되어야 할 Time Critical Task 이다[3,5]. 이에 반하여, 그래픽 표현이나 상태 모니터링과 같은 반듯이 지정된 시간에 완료되어지지 않아도 되는 태스크들은 Deadline 안에 완료되어야 할 태스크들의 수행이 종료될 때 까지 지연(Delay)될 수 있는 태스크로 분류될 수 있다. 로봇 제어에 사용된 태스크들은 다음과 같다.

사용자 인터페이스 태스크: 사용자와 제어기 사이의 인터페이스를 담당하는 태스크.

Failure Check : 주기적으로 제어기의 이상상태를 점검하여 적절한 조치를 취함.

Servo: 생성된 궤적 명령을 주기적으로 제어기에 전달.

Status Read: 로봇 엔코더 값이나 I/O 값을 주기적으로 읽음.

Status Display: Status Read 태스크에서 읽은 값을 주기적으로 화면에 출력.

On-line Command: 주기적으로 외부 센서로부터 데이터를 읽어 새로운 로봇 동작 명령을 생성.

Emulation: 가상의 작업공간에서 대상 로봇을 3 차원으로 시뮬레이션을 수행.

본 로봇 제어 시스템의 경우 On-line Command 태스크는 온라인 궤적 생성 계획을 수행하며, 센서에 의한 로봇 동작이 아닐 경우에는 궤적계획과 기구학/역기구학 등의 실행은 오프라인으로 미리 수행하게 된다.

로봇 제어기와 같은 이산 현상 시스템(Discrete Event Dynamic System: DEDS)의 성능 분석 모델링 기법으로 많이 사용하는 페트리 넷을 이용하여 앞에서 제시한 로봇 태스크의 구조를 모델링하고 성능을 검증한다. 본 논문에서는 이를 위해서 시간 페트리넷 툴(Timed-Petri Net Tool)인 Visual Simnet1.37 을 이용한다[8]. Visual Simnet 은 MS-DOS 하에서 운영되며 각 트랜지션의 서비스 거리, 각 플레이스의 용량 등을 측정할 수 있다. 실제 시스템을 정확히 모사하기 위해서 Visual C++에서 제공하는 시간 측정함수를 사용하여 수행 시간을 측정하여 모의실험 데이터로 사용한다. 제어기의 태스크를 표 1 과 같이 정의하였다.

표 1. 로봇 태스크의 정의(0: 최상위 우선순위)

Table 1. Definition of robot tasks (0: highest priority)

태스크	특성	주기	실행시간	우선순위
Failure Check	주기적	100 ms	1 ms	4
Uncertain Task (DPC)	비주기적	0 ~ 40 ms	5 ms	1
Servo	주기적	20 ms	0.2 ms	0
Status Display	주기적	200 ms	5 ms	5
Status Read	주기적	20 ms	0.2 ms	3
On-line Command	주기적	20 ms	10 ms	2
Emulation	주기적	400 ms	30 ms	5

표 1 에서 Servo 태스크는 외부 하드웨어 최상위 인터럽트를 이용하여 구현되었으므로 DPC 보다 높은 우선순위를 가진다. Windows NT 시스템에서 DPC 를 정확히 모델링 하기란 불가능하다. 따라서 시뮬레이션에서는 불확실성 요소인 DPC 를 일정 구간 안에서 변화하는 주기를 갖고 실행되는 비주기 태스크로 가정하였다. 본 모의 실험의 목적은 제시한 로봇 제어 시스템에서 불확실성 요소인 DPC 가 로봇 태스크에 미치는 영향을 분석하는데 있다. 서비스 거리란 각 태스크가 실행되는 사이에 소요되는 시간을 말한다. 시뮬레이션에서 사용된 스케쥴링 클럭은 500 μs 를 사용하였다. 실험 결과를 표 2 에 나타내었다.

우선 DPC 를 고려하지 않은 Sim1 에서는 모든 태스크 들이 로봇 동작에 영향을 미치지 않음을 알 수 있다. 그리고 DPC 를 고려한 Sim2 에서는 On-line Command 등의 태스크가 서비스를 정확한 시간에 받지 못할 확률이 높음을 알 수 있다.

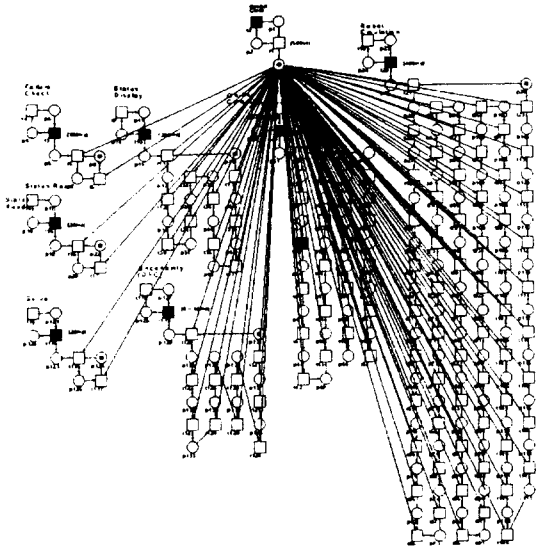

그림 4. 로봇 태스크의 페트리넷 모델링

Fig 4. Petri net modeling of robot task

Sim3 에서는 On-line Command 의 주기를 20ms 에서 60ms 로 증가 시켜 시뮬레이션을 수행한 결과 Time Critical Task 인 On-line Command 태스크는 약간의 오차는 있으나 로봇 동작에 영향을 미칠 정도는 아니다. Status Read 태스크의 편차가 비교적 큰 것은 주기가 작고 우선 순위가 Servo 에 상대적으로 낮은데 기인하지만 로봇 동작에 크게 영향을 끼치지는 않는다. Status Display 와 Emulation 태스크가 우선 순위가 낮음에도 불구하고 계속 서비스를 받는다. 이것은 우선 순위가 낮지만 실행 주기가 Servo 에 상대적으로 길기 때문이다. 이러한 페트리넷을 이용한 성능 분석을 통하여 제안된 로봇 제어용 소프트웨어의 성능을 미리 예측하고 그 자료를 이용하여 최적의 시스템 설계가 가능하다.

표 2. 태스크의 서비스 거리 (단위: ms)

Table 2. Service distance of tasks (unit: ms)

상 태	태스크	Mean	Deviation
Sim1 : DPC 가 없을 때	Servo	20	0
	On-line Command	20	0
	Status Read	20	0.099
	Failure Check	100	0.099
	Status Display	200	0.099
	Emulation	400	0
Sim2 : On-line Command (20ms)	Servo	20	0
	On-line Command	20.2	0.975
	Status Read	21.85	6.6
	Failure Check	101.5	4.88
	Status Display	200	0.099
	Emulation	400	0.099
Sim3 : On-line Command (60ms)	Servo	20	0
	On-line Command	60	0.099
	Status Read	20	0.201
	Failure Check	100	0.099
	Status Display	200	0.099
	Emulation	400	0.099

6. 결 론

최근 여러 가지 센서를 쉽게 인터페이스하고 사용자 정의의 다양한 기능을 쉽게 적용이 가능한 고기능 개방 구조형 로봇 제어기에 대한 관심이 증가하고 있고, PC 가 가지는 여러 가지 장점 때문에 이를 응용한 기술의 필요성이 높아졌음에도 불구하고 국내에서는 PC 를 이용한 로봇 제어기 설계 기술 및 개방 구조형 제어기에 대한 기술이 미비한 상태에 있다. 따라서, 본 논문에서는 PC 응용의 개방 구조형 로봇 제어기를 설계하기 위해 개방 구조형 제어기의 특성을 알아보았고, 이를 토대로 개방 구조형 로봇 제어기의 하드웨어와 소프트웨어 구조를 제시 하였다. 소프트웨어의 개방 구조를 위하여 로봇 언어, 로봇 동작 등 사용자 인터페이스를 중심으로 여러 개의 모듈로 나누고 이를 위한 API(Application Program Interface)를 정의 하였으며, 이를 위한 소프트웨어 구조를 정의 하였다. PC 에서 안정되고, 다양하고 강력한 개발 환경을 제공하며, GUI(Graphics User Interface)환경이 우수한 Windows NT 를 제어기 운영체제로 사용하기 위한 문제점을 분석하고, 이를 극복하기위한 방법을 강구하여, 로봇 제어기 실시간 운영 체제로서의 적합성을 보였다. 여러 가지 소프트웨어의 개발을 위한 모듈별 정의를 내리고 구현을 위한 구조적 특성을 함께 나타내었다. 그리고 소프트웨어의 안정성과 실시간성을 검증하기 위해서 로봇 제어시스템의 태스크를 정의하고 시간 페트리넷(Timed-Petri Nets)을 이용하여 시뮬레이션을 수행하여 결과를 분석하였다.

참고문헌

[1] D. Schmitz, R.Hoffman, P K. Khosla, and T. Kanade, "CHIMERA: A Real-time Programming Environment For Manipulator Control", IEEE International Conference on Robotics and Automation, pp846-825, 1989

[2] G. C. Buttazzo, and M. D. Natale, "HARTIK: A Hard Real-Time Kernel for Programming Robot Task with Explicit Time Constraints and Guaranteed Execution", IEEE International Conference on Robotics and Automation, pp. 404-409, 1993

[3] D. Simon, P. Freedman, and E. Castillo, "Analyzing the Temporal Behavior of Real-time Closed-loop Robotic Task", IEEE International Conference on Robotics and Automation, pp.866-873, 1994

[4] "Getting Start with CODE", Cimetrix Inc.,M-CV 3.4 1 E 7.95

[5] J. Park, S. Birla, and K. G.. Shin, "An Open Architecture Testbed for Real-Time Monitoring and Control of Machining Process', American Control Conference in Seattle, June 1995, WA08-2

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

[7] P. Wright, and S. Schofield, "Open Architecture Control for Machine Tools", canada96

[8] S. W. Son, and K. D. Lee, "An Implementation and Performance Analysis of Real_Time Robot Control Software", Trans. KIEE. Pp.1264-1272, Vol.46, No.8.Aug..1997