전자부품 및 반도체 조립을 위한 PC 기반 개방형 제어기 플랫폼의 설계

장성진, 김홍록, 서일홍
한양대학교 전자공학과

A Design of PC-Based Open Architectural Controller Platform for Assembling Electronics Devices and Semiconductor Devices

Sung Jin Jang, Hong Rok Kim, Il Hong Suh
Dept. of Electronics Engineering, Hanyang University

Abstract: 산업화가 가속화됨에 따라 시스템은 더욱 복잡하고 다양화 되어가고 있으며, 이들의 제어기 또한 지속적으로 빠르게 발전되어오고 있다. 하지만, 대부분의 제어방식이 전용제어기를 통한 시스템 구조를 가지며 독점적인 기술에 의존하고 있어, 제어기의 통합 및 확장을 통한 생산설비의 구축은 많은 어려움을 지니고 있다. 따라서, 개방형 플랫폼 설계를 위해 제어기의 구성요소를 단독구동가능의 여러 부분으로 나누어 계획하고, 각 부분은 공통된 통신채널을 통해서 서로 데이터를 주고받을 수 있는 형태를 제안하고자 한다. 이 방식을 모듈화 설계라 부르기로 한다. 이를 이용하면, 제어기의 구성은 모듈간의 조합으로 간단히 이루어 질 수 있고, 새로운 제어기의 개발은 기존 모듈의 조합 및 확장 그리고 새로운 모듈의 추가로 쉽게 개발되어 질 수 있다. 이에, 모듈화 방식의 개방형 제어기 플랫폼 설계를 제안하고 실제 반도체 조립장비에 적용방안을 살펴봄으로써 제어기 설계의 개방성을 추구하고자 한다.

주요어: 모듈화 설계, 개방형 제어기, 플랫폼, 반도체 조립장비

1 서론

전자부품 및 반도체 조립을 위한 장비는 산업기술의 발전이 가속화됨에 따라 점점 더 고정밀도의 높은 기술력을 요구하게 되었다. 장비를 위한 전용제어기를 개발하여 시스템을 구성하면 높은 성능의 원하는 결과를 얻을 수는 있지만, 변화되는 장비들마다 새로운 제어 시스템을 구축하는 과정은 많은 시간과 비용을 요구하는 어려움이 있다. 또한 전용제어기는 개발상 위주의 높은 의존성을 지니고 있어, 서로 다른 제어기를 이용한 통합 및 확장에는 많은 문제점을 지니고 있다. 따라서, 이러한 단점을 극복하고 생산성 향상을 추구하기 위해서는 표준적인 제어기 모델을 통한 개방형 플랫폼 설계가 필요하다. 이에, 동일한 연구투자로 활용도를 극대화 할 수 있는 방안인 PC바탕의 개방형 제어기 플랫폼을 제안하고자 하는 것이다. PC제어기는 이미 NC분야를 선두로 그 활용도가 점차 확산되고 있는 추세에 있으며, 풍부한 범용 기능과 프로그램의 개발 및 조작의 용이성

을 가지고 있어 제어기 개발에 많은 이점을 지니고 있다. 즉, 본 논문에서 제시하고자 하는 것은 PC바탕의 모듈화 방식으로 제어기를 설계하고자 하는 것이다. 여기서 모듈의 의미는 재사용성, 확장성 및 이식성을 고려한 의미의 단독구동 가능한 형태의 하드웨어 및 구동 소프트웨어의 조합을 말한다. 각 모듈의 관리 및 통신을 위해서 표준적인 입출력 함수와 통신 프로토콜을 정의하여 서로 다른 목적의 모듈간에도 데이터 전송의 원활함을 이루게 한다. 또한, 이미 표준적인 방식으로 구성된 모듈은 언제든지 다른 시스템에서 재사용 되어질 수 있으며, 새로운 제어기 개발은 모듈간의 조합 및 필요한 새로운 모듈의 추가를 통해서 개발되어 질 수 있다. 따라서, 실제 반도체 조립 장비인 chip mounter장비를 분석하여 보고, 위 시스템 설계방식을 모듈화 방식으로 적용했을 때의 모습을 통해 개발의 이점을 살펴보기로 한다.

2 본론

2.1 장비의 분석

살펴보고자 하는 장비는 반도체 조립시스템 중의 하나인 chip mounter이다. 이는 장비들 중에서도 복잡도가 높으며 고정밀 제어를 요구하고 windows환경의 사용자 인터페이스를 제공한다. 이 장비의 기구부 구성도를 간략히 살펴보면 그림 1과 같다. 현재 이 장비의 제어부는 전용방식이며

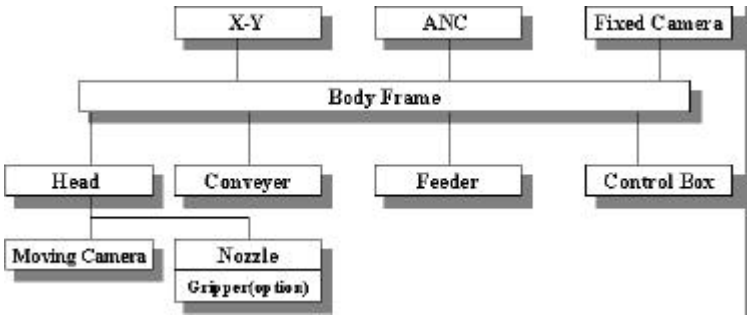


그림 1. 기구부 구성도

Fig. 1. A structure of machine

크게 모터 구동부, 이미지 처리부, 데이터 통신부로 나눌 수 있다. 이것들은 다시 body frame을 중심으로 vision head, x-y conveyer, ANC, feeder, nozzle, fixed camera, control box부 등으로 이루어져 있다. 또한, 각각의 구성 블록들은 데이터전송을 위해 다양한 방식으로 복잡하게 연결되어 있으며, 사용자 인터페이스를 통한 변수 설정 및 작업 화일을 작성하여 전체 시스템을 구동하게 된다. 이러한 방식은 전용제어기로서 원하는 목적의 시스템을 높은 성능의 장비로 구현하기에는 적합한 방식일지 모르나, 추후에 새로운 모듈의 추가 혹은 확장 시에는 제어부의 수정과 더불어 구동 프로그램의 전반적인 수정이 불가피한 방식이다. 따라서, 이것을 모듈화 방식의 제어기 플랫폼 설계를 통해서 개방형 제어기로의 전환을 고려해 볼 필요가 있다.

2.2 MMI(Man Machine Interface)

System menu구성은 MMI부분에서 가장 많이 접근하는 부분이며, 이것의 접근 용이성과 새로운

사용자 메뉴의 구성 및 확장은 매우 중요하다. 따라서, 기본 메뉴를 설정할 때 전체 시스템의 특성을 고려하여 특징에 맞는 분류기준을 선정할 필요가 있다. 여기서는 일반적인 제어기가 가지는 메뉴를 포괄 할 수 있는 Setup, Diagnostics, Teach, Run, Monitor, Comm, Option, Servo, Help의 9가지 기준 메뉴를 정하였고, 현재 반도체 조립장비의 구성메뉴를 이것에 적용하여 재분류하면

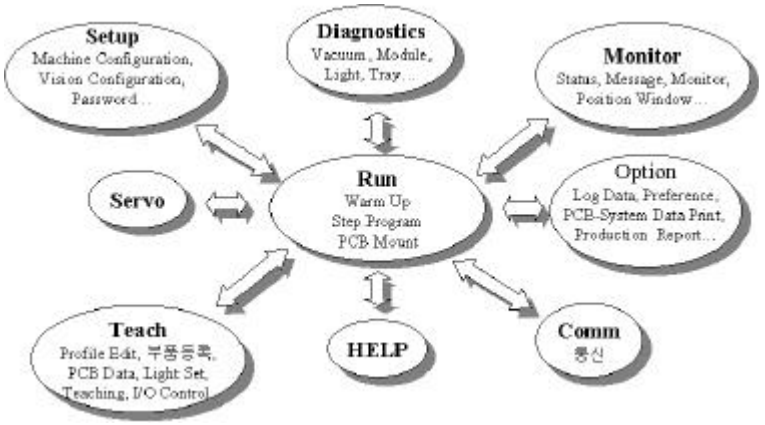


그림 2. 메뉴의 구성도

Fig. 2. A structure of menu

그림 2와 같은 모습을 가지게 된다. 이 기본분류 방식은 대부분의 제어시스템에 적용가능하며, 9

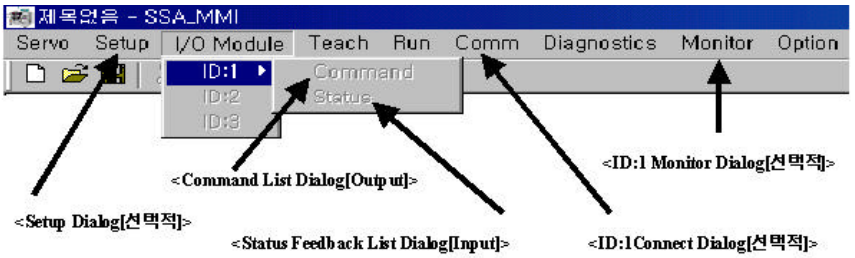


그림 3. 메뉴에 모듈추가하기

Fig. 3. Insert module to menu

가지의 상위 메뉴에 속하는 서브메뉴는 반도체 조립장비의 구성 부분에 맞게 추가할 수 있다. 유연성과 확장성의 의미로 서브 메뉴가 상위 메뉴로 plug-in방식으로 필요로 할 때 언제든지, 쉽게 추가 및 확장할 수 있어야 하는 것이다. 즉, 기본 메뉴의 틀은 변동 없이 시스템 구성의 변동에 따라 MMI에서 간단히 구동 프로그램을 연결할 수 있게 하는 방식을 적용하는 것이다. Plug-in 방식의 메뉴구성을 위해서는 MMI에 새로운 프로그램 추가를 통한 메뉴의 확장 시에 구분하여 인식할 수 있도록 추가된 구동메뉴에 구분 ID 번호를 부여한다. 그림 3에서 보는 바와 같이 부여된 ID 번호는 MMI에서 관리하는 구분자가 되며, 다음에서 제안하고자 하는 Module화 방식의 제어기 구성에서 Module의 연결번호가 된다. 이 방식을 실제로 구현하는데는 여러 가지 tool을 이용할 수 있으나, 여기서는 Visual C++프로그램의 COM(Component Object Model)방식을 제안한다. COM방식으로 구성한 프로그램을 메뉴에 추가하는 것은 windows registry에 각 하위 구동 프로그램의 구분 아이디를 등록하고, 필요로 하는 때에 호출하며, COM 통신채널을 통해 함수를 사용할 수 있게 하는 방식이다. 기존의 일반적인 DLL의 구성방법은 새로운 menu나 library추가 시에 전체적인 프로그램의 compile과정이 필요하였으나, 이 방식은 시스템 등록 후의 호출 방식이라 메

뉴연결만으로 더욱 간단하게 구동프로그램의 확장 및 추가 변경을 용이하게 한다.

2.3 Module화 구성

일반적인 제어 시스템을 살펴보면, 다음의 그림 4와 같이 상위에서 하위로의 명령전달과 하위에서 상위로의 상태전달 방식을 이루고 있음을 알 수 있다. 장비에 따라 이 구성의 레벨 차이는 있지만, 이는 하나의 시스템을 여러 독립된 모듈로 세분화하여 구성할 수 있다는 것을 의미하며, 또한, 이 개념은 모듈화 방식의 개방형 제어기 플랫폼 설계의 동기가 된다.

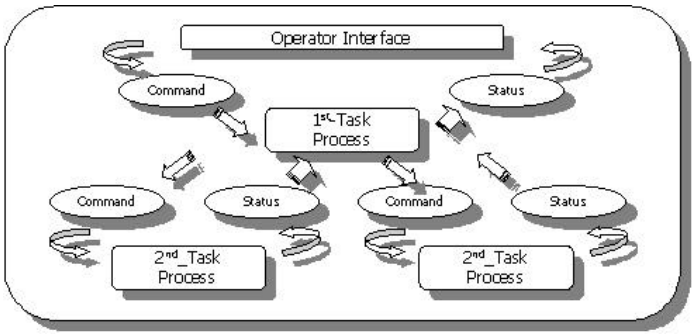


그림 4. 모듈형식의 제어기 구성

Fig. 4. the composition of Modular type controller

모듈의 기본 모델은 우선 단일단계의 형태로 구성한다고 가정한다면, 하드웨어 구동블록, 구동

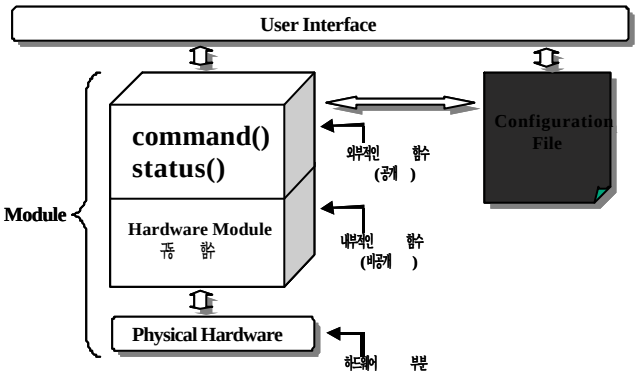


그림 5. 모듈

Fig. 5. Module

프로그램, 그리고 모듈의 setting을 위한 configuration file을 기본으로 이루어진다. 구동 블록은 실제 하드웨어와 구동함수부분이고, 구동 프로그램은 외부 개방 함수로 Command, Status부분을 통한 구동블록함수를 접근하는 통신채널을 갖는다. 마지막으로, configuration file은 MMI혹은 구동 프로그램을 통한 접근을 통해 관리한다. 이러한 방식의 구성은 모듈개발자에게 있어서 기술적인 노하우의 보존과 단독 구동프로그램의 전문적인 개발을 유도 할 수 있으며, 사용자는 모듈 구동함수의 깊은 이해 없이도 쉽게 plug-in방식으로 블록을 추가하여 메시지 전달방식으로 사용할 수 있는 것이다. 이에 사용자가 이해하여야 하는 기본 함수의 형태를 살펴보면 그림 6과 같다. 이 2개의 외부 연결 모듈 함수는 입력 string값의 변화만으로 모듈 내부의 구동 함수를 완벽히 구동

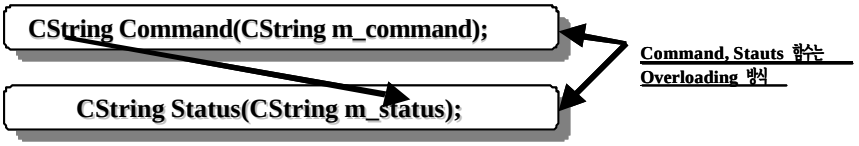


그림 6. 명령함수와 상태함수

Fig. 6. Command & Status function

할 수 있으며, 모든 통신 채널이 이 방식으로 표준화되기 때문에 연결에 있어서도 단일적인 방법이 될 수 있다.

그리고, 이 표준 입출력 함수의 protocol의 형태를 살펴보면, 하나의 간단한 형태로만 이루어져

COMMAND[STATUS] START	송신 ID	수신 ID	COMMAND[STATUS]	CHECK SUM	STOP

CString형태(COMMAND 혹은 STATUS는 Module의 정의에 따름)

그림 7. 표준 통신프로토콜

Fig. 7. Standard communication protocol

있어 다른 모듈과의 데이터 전송을 위한 연결 시에도 별 다른 수정 없이 사용할 수 있다. 실제로, 이를 사용한 시리얼 통신, TCP/IP, Can과의 연결테스터에서 이러한 공통된 표준적인 연결은 서로 다른 이종간의 통신 채널을 통한 메시지 전송을 간편히 이룰 수 있음을 볼 수 있었다. 앞에서 제시한 방식으로 구성 할 때의 예를 간단히 살펴보면 다음과 같다. 여기서, ","는 string에서 command와 parameter들의 구분자 역할을 한다.

```
Command("sender ID, receiver ID, command, parameters . . . ");
        ex) Command("001, 003, MOVJ, tpl, 70, . . . ");
Status(Command(". . ."));
        ex) Status(Command("001, 003, MOVJ, tpl, 70, . . . "));
```

PC에서는 수없이 많은 이종간의 통신이 사용되어 질 수 있다. 하지만, 우리가 필요로 하는 것은 이 이종간의 통신방식을 편리하게 쓸 수 있는 정보전송의 중간 매개체적인 부분이 필요하다. 여기서는 이를 "Common Bus Protocol"으로 칭하고, 이 부분을 소프트웨어적으로 구성을 한다. 즉, 가상적인 공통의 protocol을 사용하는 계층을 만들어 프로그래머로 하여금 하위의 다양한 하드웨어적인 통신방식에 관계없이 동일한 방식으로 모듈통신을 할 수 있게 하는 것이다. 그림 8을 보면 하위의 모듈들은 MMI를 통해서 Command, Status관계로 서로 연결이 되고, 모듈간의 통신은 Common Bus Protocol의 소프트웨어 부분을 통해 이루어짐으로서 하나의 전체적인 제어 시스템을 구성할 수 있음을 보여 주고 있다.

2.4 작업화일 구성

제어기를 모듈화로 구성한 후에 필요한 것이, 전체 구동 프로그램인 작업화일을 작성하는 것이다. 이전의 일반적인 방식은 다음 그림 9-(a)와 같이 단순 텍스트 방식인데, 이는 작업화일의 구성 시 작업자의 실수 발생율도 크고, 다른 작업자에 의해 구성된 화일의 이해도 어려움이 있다. 하지

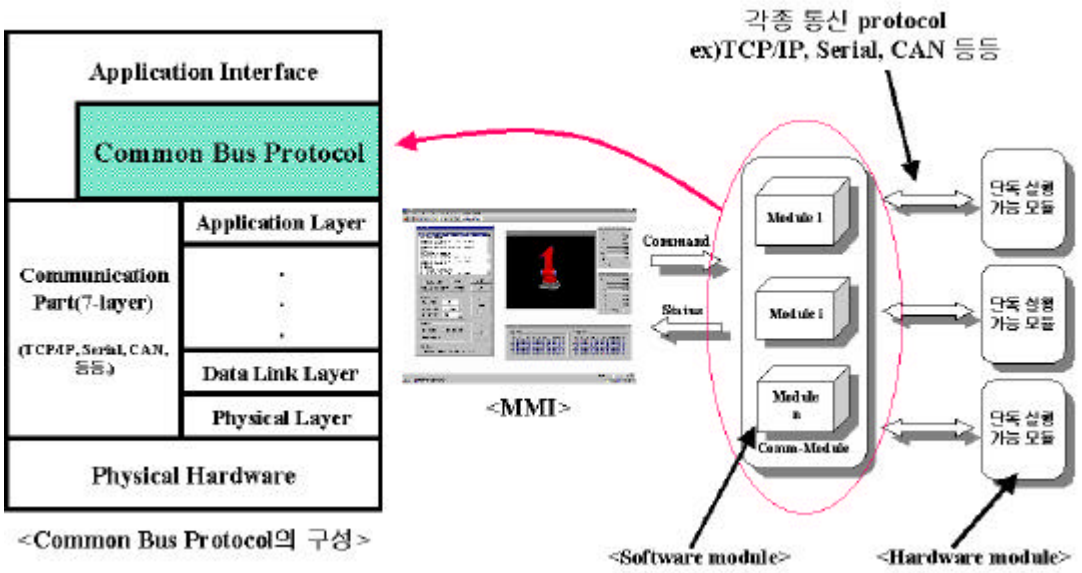
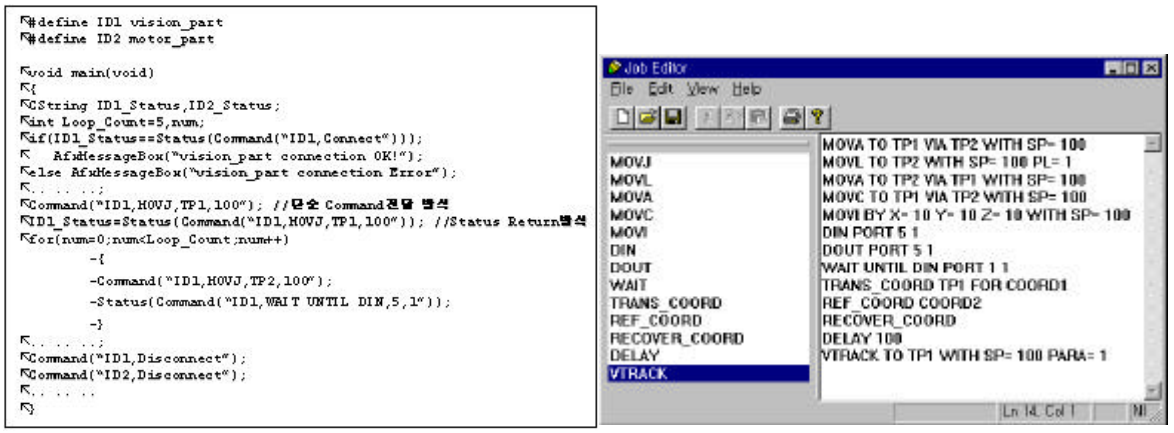


그림 8. Common Bus 프로토콜층

Fig. 8. Common Bus Protocol layer

만, 9-(b)와 같이 작성한다면, 작업화일 작성도 쉽고, 빠른 이해를 가져다 줄 수 있을 뿐만 아니라, 다양한 장비에 이식 및 적용할 수도 있다.

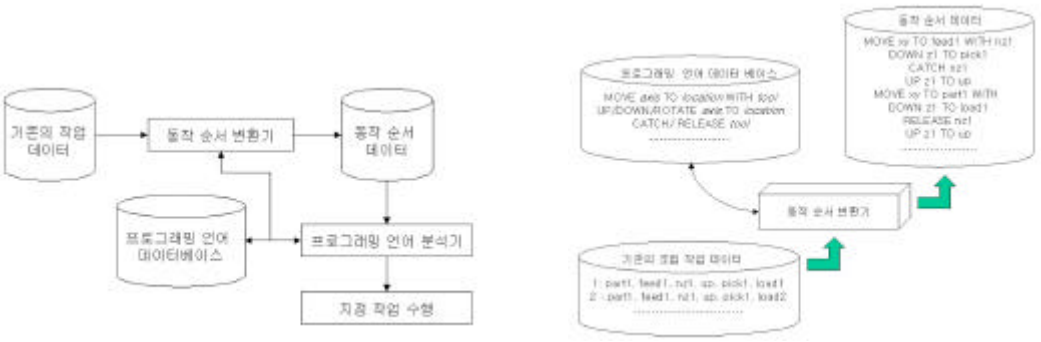


(a) 텍스트 형식 (b) 사용자 정의 GUI형식

그림 9. 작업화일

Fig. 9. Job file

그림 10은 개방형 제어기를 위한 프로그래밍 언어의 개발을 위한 언어분석기 적용을 나타낸 것이다. 그림 10.(a)는 기존의 전용장비에 대한 프로그래밍 언어 기법을 적용하는 방법을 나타낸 것이다. 이는 기존작업 순서를 규정하고 동작 순서 변환기를 통해서 최종의 사용자 정의 작업을 수행하는 방법을 나타낸 것이다. 그림 10.(b)는 전자부품 조립작업에서 사용되는 기존 전용장비의 경우에 적용할 수 있는 동작 순서 변환기의 원리를 나타낸 것이다. 여기서, 변환과정을 담당하는 프로그램 언어 분석기의 경우는 사용자에 의해 프로그래밍 언어의 추가 등록이 가능한 구조를 가지



(a) 프로그래밍 언어분석기의 적용방법 (b) 동작순서 변환기의 동작 원리

그림 10. 언어분석기에 의한 명령 수행

Fig. 10. The execution of command by language interpreter

고 있어, 사용자 정의의 명령어 사용을 그림 9.(b)에서와 같이 가능하게 해준다. 그림 11은 실제 명령어 사용의 예를 통해서 언어분석기가 어떻게 이용되는지를 보여주고 있다. "MOVJ TO LOC1 WITH SP=10"의 명령을 보면, MOVJ는 command가 되고 TO와 WITH는 구분자이며 나머지는 command에 대한 변수 값이 되어 하나의 명령라인을 이루게 되는 것이다.

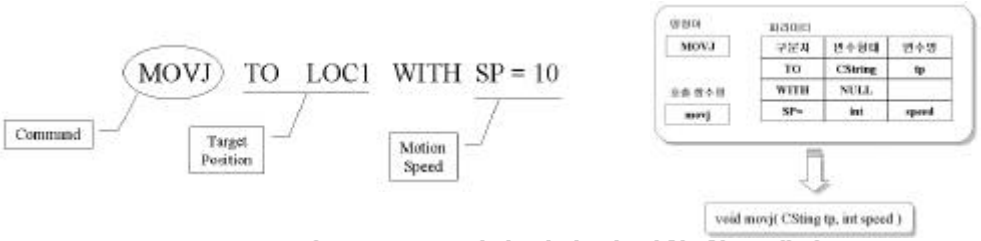


그림 11. 프로그래밍 언어 및 원형 함수 생성

Fig. 11. Programming language and generation of orignal function

그림 12. 프로그래밍 언어 분석기에 의한 명령 수행 과정

Fig. 12. The process of command by programming language interpreter

즉, 언어를 구성하여 이로부터 프로그래밍 언어 등록에 필요한 정보를 분리하여 함수의 원형을 생성하는 것이다. 결국 프로그래밍 언어 분석기는 그림 12에서와 같이 입력구문을 읽어 들여 내부

적인 처리를 거쳐 최종의 해당함수 호출까지의 과정을 이루게 되는 것이다. 따라서 사용자는 새로운 명령어를 데이터 베이스에 추가 등록할 수 있으며, 언어등록 프로그램은 이의 기본 함수 틀을 제공하여 줌으로써 편리하게 내부함수를 사용 할 수 있게 되는 것이다. 마지막으로 살펴보고자 하는 것은 모듈간의 scheduling문제이다. 모듈의 개수가 많지 않고 연결 단계가 복잡하지 않다면, 모듈간의 작업을 관리하는 scheduling은 그리 문제가 되지 않는다. 하지만, 대부분의 많은 제어 시스템을 구성하고자 할 때는 그림 13에서 보는 바와 같이 여러 단계의 모듈 연결이 수반 되게 된다. 따라서, 모듈의 연결 시에는 최상위 단에 모든 모듈관계를 관리할 수 있는 Super Module Manager가 있게 하고, 하위 단에선 그룹화를 통한 모듈의 구동을 관리하는 Sub Module Manager로 구성한다. Sub Module Manager는 이에 연결되어 있는 Module에 대해서 자체적인 관리 능력을 통해 빠른 반응과

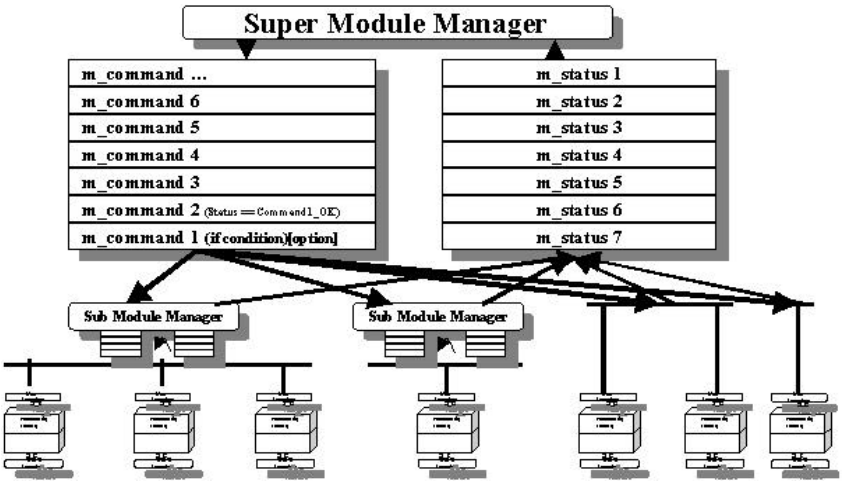


그림 13. 모듈들간의 스케줄링

Fig. 13. A scheduling of modules

구동을 위한 명령 전달 능력을 지니게 된다. 그리고 필요에 따라서 상위의 Super Module Manager로 상태의 전달이나 필요한 값의 전달을 요구할 수도 있으며, 여러 단계로 구성이 될 수도 있다. 명령의 전달과 상태값의 반환을 위한 부분은 이중의 송수신 데이터 버퍼를 통해서 값을 주고받으며, 작업의 스케줄링은 이벤트 호출방식과 태스크 수행시간을 고려한 타임 스케줄링 방식의 혼합을 통해서 이루어지게 한다. 결국, scheduling문제는 모듈간의 작업분담을 통한 기능분담을 통해서 해결하는 것이다.

2.5 Chip mounter에서의 모듈화 적용

실제 반도체 제조장비인 chip mounter에 대한 적용 시의 모듈은 크게 3가지로 구성되며, 그림 14, 15, 16에서 보는 순서대로 구동부, 이미지 처리부, Data 통신부로 나누어진다. 이 3부분은 하나의 모듈의 모습을 가지고 있지만, 내부적으로 보면 각각의 모듈은 다시 내부적인 모듈들의 조합으로 구성이 된다. 따라서 최상위단계에서는 3개의 모듈만을 관리하고, 내부적으로는 중간단계의 관리자를 두어 하위의 모듈을 관리하게 된다. 모듈간의 통신은 앞에서 제시한 Common Bus Protocol을 사용한다. 현재 모듈화의 적용으로 전체 제어기 구성 진행중이며, 앞으로 계속적인 개발을 통해 더욱 발전적인 성과가 기대되고 있다.

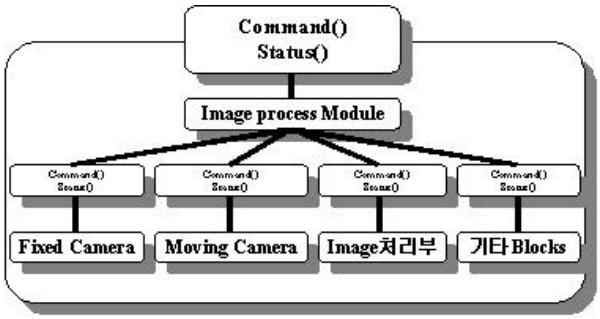


그림 14. 구동 모듈들

Fig. 14. Modules of driving

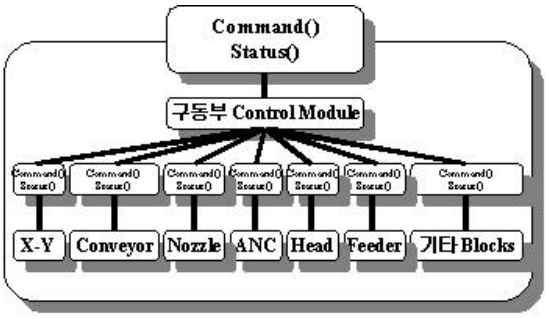


그림 15. 영상처리 모듈들

Fig. 15. Modules of image processing

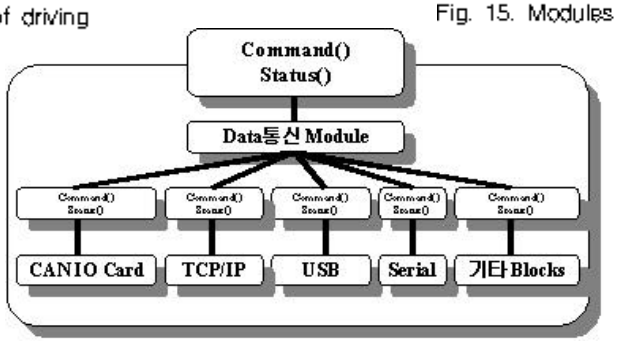


그림 16. 통신 모듈들

Fig. 16. Modules of communication

3 결론

복잡한 제어기를 모듈화 방식으로 구성하여 실제 제어시스템을 대상으로 적용하는 것은 많은 이점을 지니고 있다. 초기의 개발과정에서는 추후에 있을 문제에 대해서 많은 고려가 필요하지만, 일단 개발된 모듈은 표준적인 통신 프로토콜을 통해서 언제든지 재사용 되어질 수 있으며, 새로운 제어시스템으로의 적용 시에도 기존 모듈의 조합과 필요모듈의 추가로 효율적 개발을 할 수 있는 것이다. 실제로 복잡한 반도체 장비인 chip mounter에 대한 적용은 모듈화 구성의 이점을 보여주고 있으며, 앞으로 더욱 발전적인 연구로 모든 개방형 제어기에 공통적으로 적용될 수 있는 기술 발전의 토대를 이룰 것으로 기대되고 있다.

참 고 문 헌

[1] I.H.Suh, H.J.Yeo, J.S.Ryoo, S.R.Oh, C.W.Lee and B.H.Lee, *"Design of a Supervisory Control System for Multiple Robotic Systems"*, Proc. IROS 96

[2] Fan-Tien Cheng, Meng-Tsang Lin, Rong-Sheen Lee, *"Developing a Web-enabled Equipment Driver for Semiconductor Equipment Communications"*, Proc. 2000 IEEE

[3] Mathew L. Moore, Veysel Gazi, Kevin M. Passino, *"Complex Control System Design and Implementation Using the NIST-RCS Software Library"*, 1999 IEEE Control Systems

[4] 이낙형, 서동수, 엄광식, 서일홍, 여인택, 김성락, *"A Design Experience on PC-Based Open Architectured Controller for Robot Systems"*, Proc. 1998 KACC

[5] 추성호, 박풍성, 강원준, *"A Study of Object Agent for Open Architecture Control system"*, Proc. 1999 KACC

[6] http://www.osaca.org

[7] http://www.arcweb.com/omac