論 文
39~8~12

다중프로세서 방식의 자동조립시스템을 위한 관리제어

Supervisory Control for Multi-Processor-Based Automatic Assembly System

李 在 爀*·劉 凡 材*·吳 尙 錄**·卞 增 男***
(Jae-Hyeok Lee · Bum-Jae You · Sang-Rok Oh · Zeungnam Bien)

요 약

본 논문에서는 여러개의 프로세서를 사용하는 자동조립 시스템을 위한 새로운 구조의 관리제어를 제시 하였다. 새로 제안된 관리제어는 C언어를 사용하여 작성되었으며 구조화되어 있고 쉽게 기능의 확장이 가능하며 또 자기진단기능을 포함하여 많은 진단기능을 구현하여 신뢰성을 높였다. 이 새로운 관리제어를 비젼기능을 갖춘 고속 자동조립 시스템에 적용하여 그 유용성을 입증하였다.

Abstract- In this paper, a multi-processor-based supervisory control for automatic assembly system is presented. The proposed supervisory control is organized in terms of C-language and with structured and easily expandable characteristics. Also the controller is designed to possess diagnostic capability including self-diagnosis of processor module. The developed supervisory control has been shown to be very useful via a high speed automatic assembly system with vision capability.

1. 서 론

최근 산업계에서 생산성을 높이기 위하여 공장 자동화에 관심을 가지면서 그에 대한 연구가 활발히 추진중에 있다. 그중에서 특히 부품을 조립하여 완성품을 만들어내는 조립 공정의 자동화는 일찍 부터 주목을 받고 있으며 가장 그 필요성을 느끼고 있는 분야이지만 현재는 PLC(programmable logic controller)를 이용하여 주어진 순서대로 작업을 진행하는 단순한 조립 작업이 주종을 이루고있다. 하지만 좀더 복잡하고 정밀한 조립작업을 위해선 조립대상을 직접 보고 위치, 방향등을 알 수 있는 비젼 센서등 고도화된 센서 기능이 요구되고 생산성을 높이기 위해선 고속으로 작업이 이루어져야 한다. 이런 비젼 장치를 갖춘 고속 정밀 자동조립 시스템의 경우 비젼 정보의 양이 많아지므로 빠른 데이타 처리 능력이 요구되고 여러개의 액츄에이터(actuator)를 동시에 그리고 정확하게 제어해야 하므로 여러개의 프로세서를 써서 시스

*正 會 員 : 韓國科學技術院 電氣및電子工學科 博士課程
**正 會 員 : 韓國科學技術研究院 制御시스템研究室 先任研究員·工博
***正 會 員 : 韓國科學技術院 電氣및 電子工學科 教授·工博
接受日字 : 1990年 3月 30日
1次修正 : 1990年 7月 13日

템을 구성하는 것이 불가피하다. 하지만 이러한 다중 프로세서 시스템의 경우 일반적으로 처리속도의 증가등 여러가지 장점[1]이 있으나 전체 하드웨어가 크게 증가하여 시스템 운영을 위한 소프트웨어가 복잡해지고 시스템 성능이 소프트웨어에 크게 의존하게 된다. 따라서 이런 복잡한 시스템의 경우 효율적인 관리제어 기술이 매우 중요시 되고 있다.

이러한 자동조립 시스템의 예로는 반도체 조립공정에 많이 쓰이는 다이본더[2], 와이어본더[3] 등을 들 수 있고 이외에도 산업체에선 이와 비슷한 비젼 기능을 자동조립 시스템이 많이 사용되고 있다.

한편 관리제어(supervisory control)란 여러개의 프로세서를 사용하는 시스템의 원활한 운영을 위해 각 부 시스템의 작업을 총괄 조정하며 작업의 지시및 확인 기능을 하고 사용자와 기계간의 정보전달을 수행하는 운영 소프트웨어를 말한다.

이상과 같은 다양한 기능을 수행하는 관리제어를 다중 프로세서 방식의 자동조립 시스템에 구현하는 것은 매우 전문성을 요구하는 어려운 일이므로 체계적인 접근과 구조적인 설계없이 효율성있게 구현하기가 어렵다. 특히 최근의 다품종 소량생산 방식에 적합하게 하기위해선 관리제어의 기능을 쉽게 확장할 수 있는 확장성(expandability)과 여러가지 부품으로 다양한 제품을 만들수 있는 유연성(flexibility)이 중요시 되고 있다.

또한 최근 자동차의 전자엔진에서 엔진의 자기진단이 중요시 되고 있듯이 시스템이 스스로 자신을 진단하여 그 상태를 사용자에게 알려 줄 수 있는 지능형 시스템이어야 사용자가 조작하기 쉽고 또 고가의 장비를 안전하게 사용할 수 있다. 따라서 현대적 관리제어에서 이 기능이 충분히 고려되어야 한다.

하지만 기존의 자동조립 시스템용 관리제어는 주로 어셈블리어를 사용하거나[2, 3] 고유의 매크로언어(macro language)를 사용하기 때문에[4]프로그래밍 자체가 어렵고 구조화되기 힘들며 확장성이 나쁘고 디버깅(debugging)이 어려워 시스템의 규모가 커지고 사용자가 요구하는 기능이 많아질수록 구현이 점점 어려워지고 또 진단 기능이 거의 없는 상태이다.

따라서 본 논문에서는 최근 시스템 프로그래밍 언어로 각광을 받고있다 C언어를 이용, 소프트웨어 공학 측면에서 구조적으로 구현하여 확장성과 유연성을 증가시키고, 특히, 진단기능을 강화한 새로운 다중 프로세서 방식 자동조립 시스템의 관리제어를 제시한다.

본문의 2장에서는 본 연구의 대상이되고 있는 다중 프로세서 방식의 자동조립 시스템의 구조를 간단히 설명하고 3장에서 본 관리제어의 구성을 설명하며 4장에서 진단기능을 설명하고 5장에서 실험 결과를 설명하고 6장에서 결론을 맺는다.

2. 자종조립 시스템의 구성

2.1 기능적 구조

정밀한 부품을 비젼장치를 이용하여 원하는 조립작업을 수행하는 자동조립시스템을 기본적인 기능에 의해 부 시스템으로 아래와 같이 나누었다. 먼저 조립 대상 부품을 검사하고 위치및 경사도를 계산하는 비젼장치, 대상 부품을 운반해오고 조립된 부품을 다음 공정으로 이송하는 이송장치, 대상 부품을 직접 조립하는 매니퓰레이터(manipulator)장치, 메뉴, 정보등을 보여주는 디스플레이장치, 마지막으로 위의 4가지 장치를 총괄 관리하는 소프트웨어인 관리제어이다.

2.2 하드웨어 구조

다중프로세서 시스템을 구성하는 방식에는 여러가지가 있으나[5] 그중에서 자동조립 시스템에 적용하기에 가장 적합한 방식은 구성하기 쉽고 확장이 쉬우며 경제적인 시분할 공유버스(time shared/common bus)구조로 판단되었다.[6] 이 구조의 단점은 다른 방식에 비하여 데이타의 전송율이 낮은 것이나[5] 실제 자동조립 시스템에서는 프로세서간의 데이타 전송량이 이구조로 처리 못 할만큼 많지 않으므로 이 방식을 채택하였고 또 실제 산업체의 대부분 자동조립 시스템은 이방식을 채택하고 있다.[2, 3]

2.3 자동조립 시스템의 예

이와 같은 시스템의 예로 반도체 소자용 자동다이 본더를들 수 있다. 다이본더란 가공이 끝난 웨이퍼상의 작은 칩(die)을 집어서 리드프레임(lead-frame)위에 에폭시(epoxy)를 이용하여 붙이는 기계로 비젼 기능과 고속 정밀의 모터 서보기술이 요구되는 첨단 장비이다. 그림 1에 본 연구에서 개발한 다이본더의 외관도를 보였고 그림 2에 이의 전자 제어부 구조를 보였다. 이에 대한 좀 더 자세한 설명은 5절에서 하기로 한다.

3. 관리제어의 구성및 수행 방식

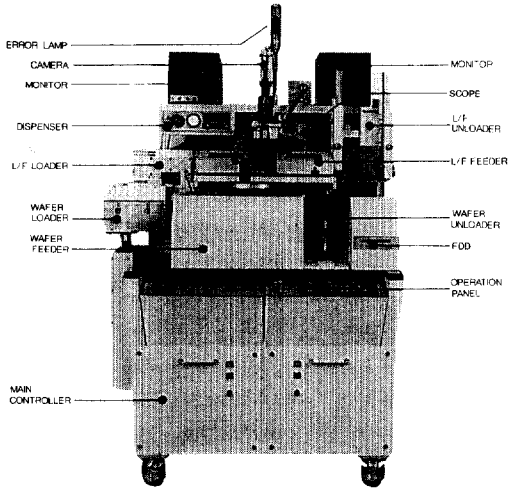


그림 1 다이 본더 외관
Fig. 1 Die Bonder

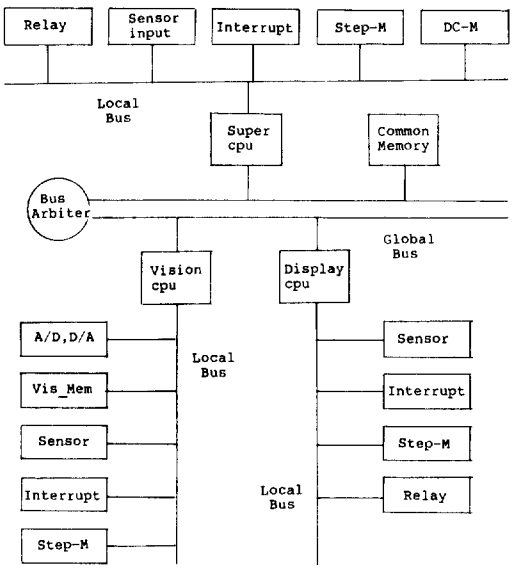


그림 2 다이 본더의 전자제어부
Fig. 2 electronic controller of Die Bonder

3.1 관리제어의 구성

개발된 관리제어를 자동조립 시스템이 가져야하는 기능적인 면에서 보았을 때의 구성과 사용자가 운전하는 면에서 보았을 때의 구성, 마지막으로 각 소프트웨어를 구성할 때 개념적으로 어떠한 구조로 구성하였는가를 설명한다.

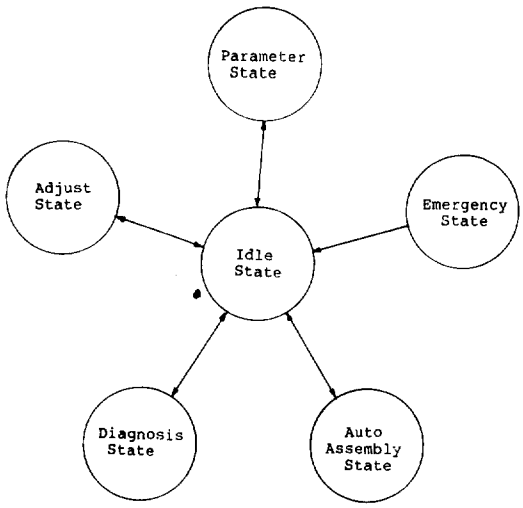


그림 3 관리제어의 6가지 스테이트
Fig. 3 6 state of supervisory control

3.1.1 기능적 구성

자동조립 시스템에서 가장 중요한 기능은 부품의 조립 기능이지만 다품종 소량 생산의 경향에 대처하기 위해 충분한 유연성을 가져야하고 생산성의 향상을 위하여 고정시간(down-time)을 줄이기 위해 시스템 자체의 안전성을 크게 할 필요가 있다. 이와 같은 점을 고려할 때 자동조립 시스템은 첫째 자동조립을 연속적으로 수행하는 기능을 가져야 하고, 둘째 조립 작업 대상에 따라 시스템 및 부품의 각 파라미터를 바꿀 수 있어야 한다. 세째로 위에서 바꾼 파라미터도 시스템 설치후 환경에 따라 좀 더 정밀하게 기계부위를 조정하거나 비젼용 데이타를 조정하는 즉 튜닝 작업이 반드시 필요하므로 이를 쉽게 할 수 있는 기능이 있어야 한다. 네째로 시스템을 안전하게 사용하기 위해 미리 시스템을 진단하거나 고장 시 수리를 용이하게 해주는 진단 기능이 있어야 한다. 마지막으로는 시스템이 동작 중에 위급한 상황이 발생한 경우 이를 처리할 수 있는 비상 상태 처리 기능이 요구된다.

따라서 본 관리제어에서는 각 기능들을 스테이트(state)라 이름하고 각각 자동조립(auto-assembly) 스테이트, 파라미터(parameter) 스테이트, 진단(diagnosis) 스테이트, 조정(adjust)스테이트, 비상(emergency) 스테이트라하고 그림 3과 같이 6가지 스테이트로 구현하였다. 여기서 아이들

(idle) 스테이트는 처음 전원을 인가한 후 시스템을 초기화 시키고 주 메뉴를 보여주고 사용자의 명령을 기다리는 스테이트이다.

3.1.2 운영상의 구성

생산라인의 사용자가 자동조립 시스템을 쉽고 안전하게 사용하게 하기 위해선 메뉴 선택 방식을 사용하는 것이 좋으므로 이를 위해서 그림 4와 같은 나무 구조로 관리제어를 구현하였다. 단 비상 스테이트는 리미트 센서의 인터럽트에 의해서만 들어갈 수 있으므로 나무 구조아래 들어있지 않다. 메뉴를 계속 선택하여 나무 구조에서 깊이 내려온 경우 뿌리(root)인 주 메뉴로 가려면 "0"키를 여러번 눌러야 하는 불편한 점이 있는데 이를 막기 위해서는 "EXIT"키를 만들어 어느 스테이트의 어느 가지에서든 주 메뉴로 쉽게 갈 수 있게 구성하였다. 이 방식은 구조적 프로그래밍[7]에 위배되는 방법이지만 사용자가 쉽게 기계를 운영할 수 있는 장점이 있다.

3.1.3 개념적 구성

자동 조립 시스템의 관리제어를 구현하는데 있어 가장 중요시 되어야 하는 것은 앞에서 설명한 바와 같이 관리제어가 확장성과 유연성이 좋아야 한다는 것이다. 그 이유는 다품중 소량 생산의 현대적 생산 개념에선 한 조립 라인에 들어오는 부품의 종류가 다양하므로 자동조립 시스템은 여러 가지 부품을 다룰 수 있어야 하며(유연성), 필요에 따라서는 새로운 기능을 추가하기 위해 프로그램을 추가할 경우도 있는데 이 경우 관리 제어 자체를 다시 프로그래밍하는 것은 비효율적이므로 관리제어는 쉽게 기능을 확장 가능한 구조여야한다(확장성). 이에 적합한 구조는 많은 컴퓨터의 운영체제(operating system)에서 채택하고 있는 계층구조(layer structure)이다. 따라서 본 관리제어도 계층구조의 장점을 고려하여 운영 체제처럼 그림 5와 같이 계층구조로 구상하였다.[8]

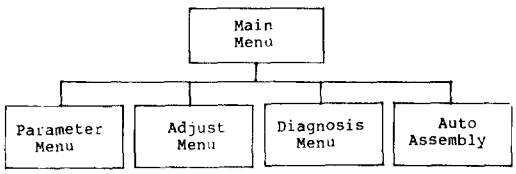


그림 4 관리제어의 나무구조
Fig. 4 tree structure of supervisory control

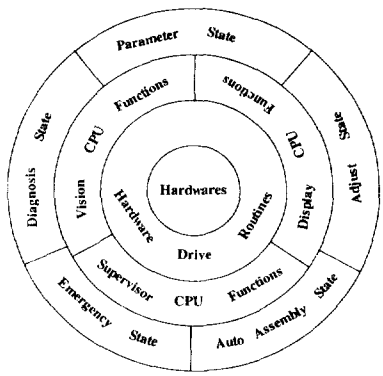


그림 5 관리제어의 계층구조
Fig. 5 layer structure of supervisory control

이를 보면, 가장 내부에 시스템 하드웨어 계층(hardware layer)이 있는데 하드웨어로는 교류 모터, 직류 모터, 스테핑 모터등 모타류의 여러 센서류, 공압 솔레노이드, A/D D/A보드, 램프등이 있다. 이 계층의 밖에는 이 하드웨어를 구동시키는 하드웨어 구동 계층(hardware drive layer)이 있는데 각 하드웨어에 따라 모듈별로 구현되어 있다. 다음 이 계층 밖에는 각 프로세서들 마다 필요한 기본 동작(primitive action)을 만들어 놓은 계층이 있다. 이를 기능 계층(function layer)라 했는데 자동조립 시스템의 경우 어떤 작업을 위해 필요한 동작의 수는 일반적인 로보트 시스템과는 달리 한정되어 있으므로 필요한 기본 동작들을 미리 데이타 베이스로 만들어 놓고 이를 이용하는 것이 효과적이므로 이와 같은 계층을 두었다. 현재 주(maser)프로세서용 기본 동작수는 약 70개, 나머지 각 부(slave)프로세서마다 약 40개가 마련되어 있다. 다음 가장 상위 계층에는 이 기능 계층의 기본 동작들을 이용하여 원하는 여러 가지 작업을 수행하는 각 테스트들이 구현 되어 있다. 이를 스테이트 계층(state layer)이라 하였다.

만일 새로운 기능이 요구되는 경우 하드웨어를 추가하고 그의 구동 소프트웨어를 작성하여 해당 계층에 추가하고 이를 이용하여 필요한 기본 동작을 만들어 기능 계층에 추가한 후 해당 스테이트에서 이를 엮어서 원하는기능을 쉽게 만들 수 있다.

3.2 관리제어 수행 방식

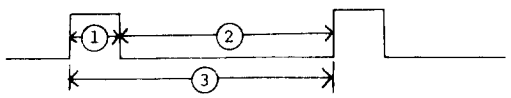
3.2.1 작업 수행 방식

관리제어는 작업 수행 도중에도 사용자의 요구에 응답할 수 있어야 하므로 타이머의 주기적인 인터럽트에 의한 폴링(polling)방식을 사용하여 키가 눌리었는가 체크한다. 즉 전체 작업을 2가지로 나누어 수행하는데 조립작업을 수행하는 것을 포그라운드(foreground) 작업이라 하고 주기적으로 키보드 체크를 하는 작업을 백 그라운드(background)작업이라 하였다. 두 작업을 그림 6과 같이 교대로 수행한다.

백 그라운드 작업은 타이머에 의한 인터럽트의 처리 루틴으로 인터럽트가 걸리면 먼저 부 프로세서들이 각각 담당하여 제어하는 시스템에 에러가 발생했는지 확인한다. 다음에는 키가 눌리었나 확인하여 눌리었으면 그 키에 해당하는 명령을 명령큐(command queue)에 넣고 마지막으로 현재 부 프로세서들의 작업 상황이 어떤가를 확인한다.

포 그라운드 작업은 각 스테이트에서 관리제어가 수행하는 작업을 말하며 중요도(priority)순서대로 명령 큐에서 하나씩 꺼내어 수행한다. 이 작업들은 그 중요도에 따라 3가지 그룹으로 나누었다.

먼저 중요도-2(priority 2)작업은 가장 높은 중요도를 갖고 있어 이 명령이 들어오면 현재하던 일을 멈추고 즉시 이 명령을 수행한다. 예를 들어 리세트(reset)과 긴급 정지(emergency stop)가 있다. 다음 중요도-1은 중요도-0를 수행하고 있을 경우 이 중요도-1을 먼저 처리하고 다시 중요도-0를 계속 수행할 수 있는 작업이다. 예를 들어 온라인 조립 작업중에 지금까지 작업 상황을 사용자가 알고 싶은 경우 이의 응답을 보여 주고 다시 계속하여 작업을 하는 경우이다. 나머지 대부분 작업들은 중요도-0로 명령 큐속에서 하나씩 꺼내어 수행한다.



No.	내 용
1	Background Job 수행 시간
2	Foreground Job 수행 시간
3	Interrupt 주기

그림 6 관리제어의 작업 수행 방식
Fig. 6 job execution scheme of supervisory control

3.2.2. 부 프로세서와의 통신 수행 방식

시분할/공유버스 구조위의 다중 프로세서 시스템에서 가장 간단하고 빠르게 통신하는 방법은 공유버스위에 공동 메모리를 사용하여 통신하는 것이다. 본 관리제어도 주 프로세서는 다른 부 프로세서 들과 공유 메모리를 사용하여 통신을 수행한다. 이때 최악의 경우 두 프로세서가 동시에 같은 메모리 영역을 억세스하여 한 프로세서가 넣어 둔 데이타를 다른 프로세서가 지워버려 시스템이 정지(dead lodk)될 수가 있다. 이러한 문제는 다중 프로세서 시스템에서는 자주 일어나는 문제로 동기문제(synchronization problem)라 한다.[9]

이 문제를 그림 7과 같이 TAS(test and set)기능을 이용하여 통신 방식을 구성하여 해결 하였다.[10] 위 방식을 좀더 자세히 설명하면·먼저 공유 메모리에 데이타를 쓰고자 할 때는 그 메모리에 대한 TAS 바이트를 먼저 살펴보아 '0이면 그 메모리는 다른 프로세서가 사용하지않고 있으므로 데이타를 쓰고 TAS바이트에 표시를 한후 나오고 0이 아니면 그냥 나오게 된다. 다음 그 메모리를 다른 프로세서가 읽을 때는 역시 TAS 바이트를 살펴보아 누가 데이타를 썼는가 살펴본 후 썼으면

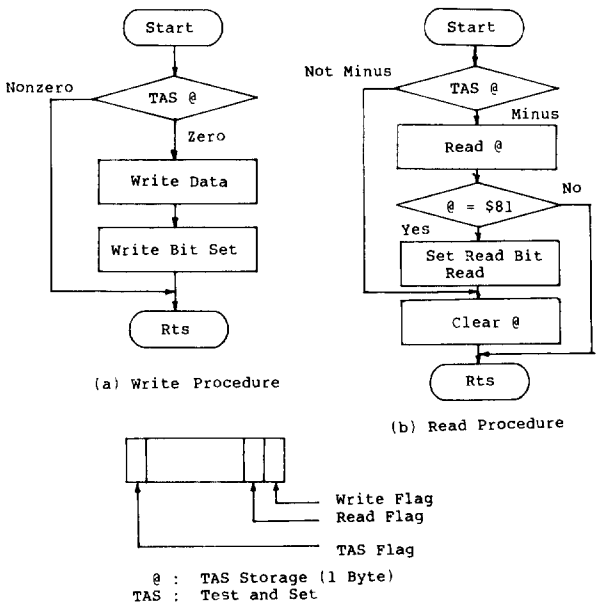


그림 7 상호 배제를 위한 통신 방법
Fig. 7 communication protocol for mutual exclusion

TAS바이트에 읽는 중임을 표시하고 데이타를 읽고나서 다시 그 TAS 바이트를 지워 순간적으로 다른 프로세서가 그 메모리를 읽고 지워 버리는 것을 못하게 하였다.

4. 관리제어의 진단 기능

생산성을 높이기 위해 1싸이클의 조립시간을 단축시키는 것도 중요하나 기계의 고장 가능성을 적게하여 고장시간을 줄이는 것이 더욱 좋은 방법일 수있으므로 본 관리제어는 자기진단 기능[11, 12]을 포함하여 전체적으로 진단기능을 강화하였다.

4.1 CPU부 자기 진단

이 부분은 프로세서에 따라 테스트 방법이 다를 수 있는데 현재 모토롤라사의 MC68000 16비트 프로세서를 사용하였다.

레지스터 및 CPU 내부의 데이타 버스 진단은 MC68000의 경우 레지스터와 CPU 내부의 데이타 버스는 32비트로 구성되어 있는데 이를 테스트하기 위해 32비트 중 단지 한 비트만 1로 하고 나머지 비트들은 0으로 한후 한 레지스터에서 버스를 거쳐 다른 레지스터로 옮긴 후 제대로 옮겨졌는가 본다. 이때 어드레스 레지스터와 데이타 레지스터를 구별하지않고 모두 테스트한다. 즉 A0레지스터에 16진수로 "FFFFFFFE"를 넣고 A1 레지스터에 A0의 보수 "00000001"를 넣고 A0->A2->A4->A6->D1->D3->D5->D7로 데이타를 한바퀴 돌리고 마찬가지로 A1->A3->A5->D0->D2->D4->D6로 데이타를 옮긴 다음 D6과 D7을 더하여 그 결과가 "FFFFFFFF"인가를 비교하여 레지스터들이 정상인가 확인한 후 A0를 하나 왼쪽으로 옮겨가면서 앞의 과정을 모든 비트에 대해 반복 수행한다. 다음 반대로 A0에 "00000001"을 넣은후 또 다시 32번의 테스트를 반복한다.

ALU(arithmetic logic unit)의 진단은 ALU는 논리 연산을 하나씩 수행하면서 그 결과를 예측한 값과 비교하여 테스트한다. 예로는

AND OF NOT BSET BCLR ASL ASR
LSL LSR ROR ROL SWAP ADD SUB
MUL DIV TEST등이 있다.

명령어 해석 기능(instruction decoder logic)의 진단은 모든 명령어를 하나씩 테스트해야 하나 지금까지의 테스트로 거의 대부분이 테스트되었다고 할 수 있으므로 여기서는 어드레싱 모드(addressing mode)만을 테스트한다. 어드레싱 모드로는

Register Addressing Immediate Addressing
Direct Addressing Indirect Addressing
Index Addressing Relative Addressing

등이 있는데 각 모드의 테스트는 이미 알고 있는 어드레스의 데이타를 각각의 모드로서 그 데이타를 읽어서 확인한다.

예외(exception)처리 기능의 진단은 MC68000자체에 내장되어 있는 예외처리 기능으로 많은 소프트웨어및 하드웨어 고장이 검출 될 수 있는 데 이에는 버스 트랩(bus trap), 어드레스 트랩(address trap), 틀린 명령(lillegal instruction), 영으로 나눔(zero devide)등이 있다. 이 예외처리 기능을 자기진단하여 시스템 사용 도중에 위와 같은 예외가 발생할 경우 관리제어에서 수행하는 예외처리 수행 루틴이 자동적으로 수행되어 시스템을 안전하게 만들 수 있는가를 테스트한다.

4.2 메모리부 자기진단

메모리에는 롬(ROM)과 (RAM)이 있는데 CPU 보드의 롬은 체크섬(check sum)으로 하며 램을 테스트하는 방법은 먼저 테스트하고자 하는 곳의 데이타를 미리 다른 곳으로 옮긴 후 그곳에 16진수 "55555555"를 쓴다. 이때 "5"는 2진수로 "0101"이므로 "0"과 "1"이 교대로 있게된다. 다음 그 어드레스를 읽어 "55555555"인가를 확인한 후 다시 그 어드레스에 "AAAAAAAA"를 써 넣는다.("A"는 "1010"이다) 마찬가지로 그 어드레스를 다시 읽어 "AAAAAAAA"인가 비교하면 항상 "1"이 되는곳과 "0"이 되는 곳이 있는지 확인할 수 있고 인접한 데이타 선 끼리의 단락 여부도 알아 볼 수 있다.

4.3 버스부 자기 진단

데이타 비스는 16비트로 구성되어 있으며 개방이나 단락을 테스트한다. 먼저 특정 어드레스를 선택하여 그곳에 16진수로 "FFFE"를 쓴 후 그 옆 어드레스에 "0001"을 쓴 후 다시 둘다 읽고 더하여 "FFFF"가 되는 지 확인한다. 다음 두 데이타를 회던(rotate)시킨 후 반복하여 테스트한다.

어드레스 버스 진단은 어드레스를 아래 비트로부터 "1"이 가장 많은 어드레스를 선택한다. 이를 베이스 어드레스라 하며 그곳에 "AAAA"를 써넣는다. 다음 그 베이스 어드레스의 한 비트씩 "0"으로 바꾼후 "5555"를 쓰고 다시 읽어 "5555"인가 비교한다.

4.4 타이머 부 자기진단

앞에서 설명한 바와 같이 타이머는 10msec마다 인터럽트를 걸어 주프로세서와 부 프로세서와의 통신과 키보드의 키가 눌리었는지 여부를 확인하는 일을 하므로 타이머의 고장은 통신 두절을 일으켜 시스템의 오동작을 가져온다. 따라서 시스템을 동작시키기 전에 반드시 진단해 보아야 하는데 타이머의 테스트방법은 인터럽트가 걸리면 메세지를 내보내게 하여 인터럽트가 걸리는 지를 확인한다.

4.5 입출력및 기타 모듈의 진단 기능

이 모듈들은 자기 진단 기능이 아니고 각 모듈의 동작 상태를 사용자가 직접 확인 하여 해당 모듈의 정상 여부를 확인한다.

입력 모듈에는 키보드와 센서가 있다. 키보드 입력은 키보드 모양을 모니터에 가상으로 그 배열을 나타낸후 그 위치에 "H" 를 쓰고 사용자가 키를 누르면 눌린 키는 "L"로 바뀌게 하여 키보드 동작을 테스트한다. 센서 입력은 각 센서의 현재 상태를 쉽게 점검할 수 있게 하기 위해 각 기계장치별로 그곳에 설치되어 있는 센서의 상태를 모니터에 나타내고 사용자가 직접 센서들을 하나씩 동작시킨 후 그 센서의 현재 상태가 모니터에서 변하는 가를 보아 쉽게 고장 유무를 알 수 있다.

출력 모듈은 온-오프 제어용 릴레이 구동부가 있다. 릴레이 구동부의 고장 여부는 각 구동부를 사용자가 모니터에 나타난 메뉴를 보고 하나씩 구동 시켜 그 동작을 직접 확인함으로 진단이 가능하다.

4.6 비젼 처리 기능 진단

각 비젼 처리 기능 역시 각각의 기능을 독립적으로수행시켜 각 기능들을 진단한다. 비젼 처리 기능은 주위 환경(조명)에 가장 영향을 많이 받는 모듈이므로 시스템을 처음 설치할때 각 기능 마다 모두 철저히 확인하고 필요에 따라서는 파라미터들을 바꿔 주어야 한다.

4.7 각 기계 부위의 진단

자동조립 시스템의 대부분의 구동 부위들이 모터를 사용하므로 각 기계 부위의 진단은 모터 제어 뮤듈의 진단과 함께 수행 되는 셈이다. 진단 방법은 각 구동 부위를 구동 부위별로 정해진 동작을 시키면서 사용자가 직접 확인한다. 예를 들어 웨이퍼 이송장치의 진단을 수행 시키면 이송장치는 정해진 동작을 수행하고 공압 솔레노이드를 진단하면 실린더가 작동된다.

4.8 시스템의 에러 진단

에러(error)란 하드웨어나 소프트웨어의 이상상태를 의미하며 일반적으로 소자의 고장, 환경의 물리적인 영향, 사용자의 실수, 부적절한 설계등의 원인으로 발생하게 된다. 본 관리제어에선 앞에서 기술한 바와 같이 진단 기능을 강화하여 시스템이 동작중에 에러가 발생할 가능성을 줄였으나 이런 에러가 발생시에는 다음과 같은 체계적인 에러 처리 과정으로 모든 에러를 처리한다. [13]

1) 에러 발견(detectionn)과정 : 에러의 발생을 알아내는 과정으로 자기진단 이나 센서 상태의 이상등으로 알아낸다.
2) 에러 국소화(isolation)과정 : 한번 발생한 에러가 다른 영역으로 확산되지 않게 하는 것으로 자동조립 시스템의 경우 에러가 난 영역과 관련된 모든 기계 장치를 세워야 한다.
3) 에러 회복(recovery)과정 : 에러로 인한 영향을 제거하고 에러가 발생하기 이전의 상태로 전환시킨다.

한편 자동조립 시스템의 에러를 분석하여 에러의 심각성에 따라 2가지 방식으로 나누어 처리한다. 즉 정상적으로 일어날 수 있거나 일어나도 기계가 스스로 처리할 수 있어 시스템을 멈추지 않아도 되는 에러와 긴급한 상황으로 시스템 자체의 안전이 문제가 되는 에러로 나누어 각각 온라인 에러, 긴급 에러로 이름하고 다음과 같이 처리한다.

온라인 에러란 정상적인 조립 작업중 사용자의 조치가 필요하게 되는 사건을 말하는데 예를 들어 부품을 제대로 잡지 못해 작업을 잘못할 수 있을 경우나 부품이 다 떨어져서 더 이상 작업을 못하는 경우등이다. 여기서 부품이 떨어진 경우는 엄밀한 의미의 에러는 아니지만 시스템의 계속적인 정상 작업을 막고 있는 원인이 되어 사용자가 처리를 해주어야 하므로 에러로 처리한다. 이 에러는 사용자가 그 조치를 취해주면 바로 계속 정상 작업으로 들어갈 수 있게 구성하였다.

긴급 에러는 시스템의 예기치못한 오동작으로 인해 각 기계 부위마다 허용동작 범위 밖에 설치되어 있는 리미트 센서에 의해 강제적으로 인터럽트가 걸려 이를 비상 스테이트에서 처리하여 주는 에러이다. 이 에러에 대한 처리는 각 액츄에이터를 세우고 그 액츄에이터 이름을 모니터에 보여주어 사용자가 알 수 있게 하고 수동 또는 자동으로

선택적으로 복구가 가능하다.

5. 실험 결과및 검토

지금까지 설명한 새로운 구조의 관리제어를 한국과학기술원과 삼성항공(주)이 공동으로 개발한 다이 본더에 적용하여 보았다.

그림 1은 다이본더의 외관이다. 그 동작 순서를 간단히 설명하면 다음과 같다. 먼저 가공이 끝난 웨이퍼가 웨이퍼 로더(loader)에 올려지면 이를 이송장치를 통하여 이송한 후 링 홀더가 이를 본딩헤드가 작업하기 쉬운 위치로 가져온다. 이때 다이를 붙일 리드 프레임은 정확한 위치에 미리 이송하여 두고 본딩헤드는 다이 하나를 진공으로 집어서 에폭시를 이용하여 리드프레임에 붙인다. 작업이 끝난 리드프레임은 다시 이송되어 통 속에 쌓이게 된다. 이때 비젼 장치는 앞 공정에서 테스트후 불량 다이로 판단되어 검은 점이 찍힌 다이를 구별하여 이런 다이는 조립에서 제외시키게 한다. 또 양호한 다이는 비젼 기능으로 그 위치를 매번 교정하여 정밀한 본딩 작업이 되게 정보를 제공한다.

그림 2는 각 프로세서와 그에 딸린 하드웨어 구조이다. 중앙의 버스가 VME버스이고 이곳에 3대의 프로세서가 있으며 각각 수퍼(super), 비젼(vision), 디스플레이(display)라 이름하였다. 그리고 버스를 시분할해주는 버스 아비터(arbiter)와 프로세서간의 정보 전달용 공유 메모리(common memory)가 있다. 또한 프로세서마다 릴레이 제어장치, 센서 입력기, 인터럽트 처리기, 모터 구동장치등이 로컬(local)버스를 통해 연결되어 있다. 그림 8는 모니터에 자기진단후 결과를 보여준 것이다. 그림 9는 각 센서들의 현재 상태를 보여주

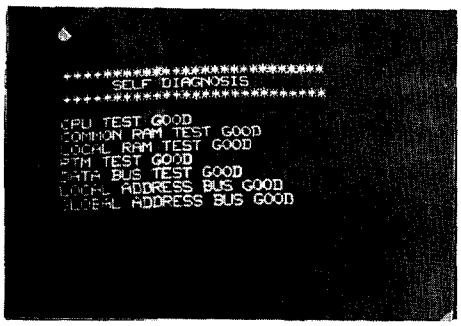


그림 8 자기진단후의 결과
Fig. 8 self-diagnosis results

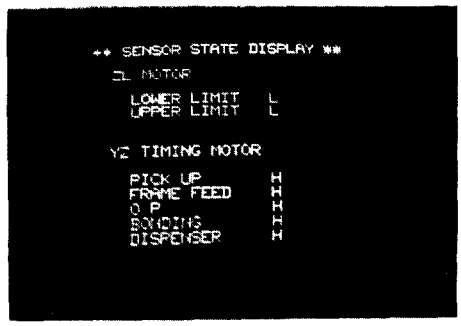


그림 9 각 센서들의 상태
Fig. 9 sensor status display

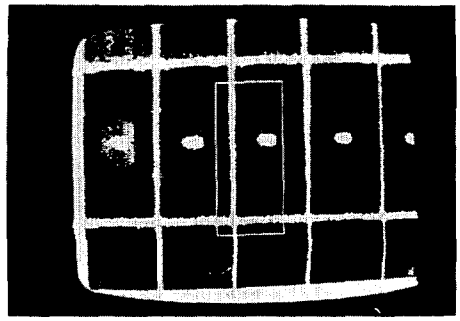


그림 10 비젼 처리후 결과
Fig. 10 result after vison processing

어 사용자가 진단하게 한 것이고 그림 10은 비젼처리 후의 결과를 보여준 것이다.

본 관리제어는 같은 다이본더에 기존에 어셈블리어로 매크로 방식으로 간이형 언어를 만들어 이 언어만으로 작업을 기술하는 방식에 비해 확장성과 디버깅성, 유연성이 뛰어나고 진단기능이 추가되어 신뢰도가 높아졌음을 적용 실험 결과 확인할 수 있었다. 이 매크로 방식은[4] 작업이 복잡한 경우 제한된 명령어 세트와 논리 판단 방식, 반복 루프등에서 크게 제한을 받고 인터프리트 방식으로 동작하므로 속도가 느려지는 단점이 있다. 특히 이 방식은 관리제어 CPU자체는 관리제어만을 수행하여 하나 본 방식은 관리제어 CPU도 직접 작업을 하면서 부 시스템을 총괄 관리할 수 있으므로 하드웨어 면에서도 최적화를 이룰 수 있다.

본 구조적인 관리제어는 시스템을 처음 개발할 때부터 계속적으로 기능이 보완되고 추가될 때 매우 효율적임을 확인하였고 특히 계층구조는 하드웨어와 소프트웨어를 여러 사람이 공동으로 개발할 경우 전체를 하나로 합치는 작업을 수행할 때

아주 효과적임을 확인하였다. 현재 나와있는 대부분의 시각 장치를 갖춘 자동 조립 시스템은[2, 3, 14, 15, 16, 17]어셈블리 언어로 되어 있어 이 분야에서는 본 논문에서 처음으로 C언어가 시도되었다. 하지만 C언어의 보급으로 머지않아 이 분야에도 고급 언어 적용 사례가 늘 것으로 생각된다.

현재 자기진단의 경우 프로세서와 관련된 모듈만 가능하고 입출력및 기타 기계부는 자기진단이 불가능하다. 따라서 좀 더 많은 부위의 자기진단 기능을 추가할 필요가 있으며 현재 키보드를 폴링 방식으로 처리하고 있으나 이를 인터럽트에 의해 수행할 경우 좀 더 조립시간이 단축되고 소프트웨어도 간단해 질 것이다.

현재 본 관리제어는 계속적으로 필드 시험을 하고 있으며 여기서 나타난 여러가지 문제점을 보완하여 좀 더 좋은 관리제어가 되게 연구를 계속하고 있다.

6. 결 론

본 논문에서는 여러개의 프로세서를 사용하는 자동조립 시스템의 관리제어를 C언어를 이용하여 모듈화 되고, 계층구조를 갖게 구조적으로 구현하였다. 또 실제로 현재 개발 중에 있는 다이 본더에 적용하여 본 논문에서 제시한 구조의 관리 제어가 작성하기도 쉽고, 유연성과 디버깅성(Debuggability)에서 어셈블리어로 구성하는 기존의 관리제어 보다 뛰어남을 확인하였으며 특히 계층구조는 하드웨어가 큰 비중을 차지하는 시스템의 개발에 있어 여러 사람과 공동 작업을 할때 유리하고 새로운 기능의 추가나 삭제등이 쉽게 되어 점점 기능이 확장되어 가는 자동조립 시스템 분야의 관리제어로서 적합함을 확인하였다.

또 이러한 구조성을 바탕으로 최근 산업체에서 강조되고 있는 진단기능을 강화하여 시스템을 쉽고 안전하게 쓸 수 있는 관리 제어를 구현하였다.

이와같은 관리제어는 본문에서 제시한 구조의 자동조립 시스템에는 모두 적용 가능하여 공장 자동화의 일원으로 산업계에서 추진중에 있는 자동조립 시스템 분야의 발전에 도움을 줄 수 있다고 생각한다.

추후로는 프로세서 모듈외에 나머지 모듈들의 자기진단 기능을 확장하는 등 신뢰도를 더욱 증가시키는 방법을 연구하여야 하고 특히 공장 자동화의 일원으로 여러 자동조립 시스템의 네트웍화가 요구되므로 지금의 구조 위에 공장 자동화용 네트웍으로 가장 널리 알려져 있는 MAP(Manufacturing Auto mation Protocol)등에 부응할 수 있는 네트웍 기능의 구현이 미래의 관리 제어에 필요하다고 할 수 있다.

끝으로 본 연구를 위해 많은 자료를 제공하여 주신 삼성 항공 산업(주)의 여러분께 깊은 감사를 드립니다.

참 고 문 헌

[1] Elit. Fathi and Moshe Krieger, "Multiple Microprocessor Systems: What, Why and When", IEEE Computer, pp. 23~32, Mar., 1983.

[2] Shinkawa Corp., "Die Bonder Manual", 1987.

[3] K&S Corp., "Wire Bonder Manual", 1985.

[4] 과학기술처, "시각장치와 로보트 매니퓰레이터를 이용한 자동조립 시스템에 관한 연구", 국책 연구 과제 보고서, 1986.

[5] P.H. Enslow JR., "Multiprocessor Organization-A Survey", Computing Surveys, Vol 9, No. 1, Mar., 1977.

[6] 김덕진, 김영천, 박석천, "밀 결합 멀티 프로세서 시스템의 구현및 성능 평가", 대한전자 공학회 논문지, 24권, 5호, 9호, 1987.

[7] Randal W. Jensen, "Structured Programming", IEEE, Computer, Mar, 1981, pp. 31~48.

[8] Comer, Operating System Design: The XINU Approach, Prentice-Hall, 1984, pp. 1~184.

[9] James L. Peterson, Abraham Silberschatz, Operating System Concept, Addison Wesley, 1985, pp. 337~339.

[10] Motorolar Corp., MC 68000 16/32-Bit Microprocessor, Prentice Hall, 1983, p. 159.

[11] Dhanajay Brahme, Jocob A. Abraham, "Functional Testing of Microprocessors", IEEE Trans. on Computer, pp. 475~485, Jun, 1984.

[12] 신 영 달, Boiler Backup Control 을 위 한 Multiprocessor방식에서의 신뢰도 개선에 관한 연구, 한국과학기술원, 석사논문, 1987.

[13] 이 헌, "Fault tolerant Computing System", 전자 교환 기술, 1권, 1호, 1985, pp. 46~61.

[14] Shinkawa, "Automatic Wire Bonder", Soft-

ware manual, 1986.
[15] Kulicke & Soffa, "Model 1412 Automatic Ball Bonder", 1986.
[16] Hughes Aircraft Company, "Model 2500 Die Bonder", Operator Manual, 1985.
[17] Foton Company, "Video-auto Die Bonder", 1982.