論文 88-25-8-7

다중 프로세서를 이용한 제어 시스템에서의 자체고장탐지

(Fault Detection of the Control System Based on Multiprocessors)

申 永 達*, 金 志 泓*, 金 炳 國**, 鄭 明 振*, 卞 增 男*

(Young Dal Shin, Ji Hong Kim, Byung Kook Kim, Myung Jin Chung and Zeung Nam Bien)

要 約

신뢰도의 향상은 컴퓨터를 이용하는 모든 시스템 특히 디지탈 제어시스템에 있어서는 매우 중요한 문제이다. 본 논문은 다중프로세서를 이용한 제어기에서 신뢰도를 개선하기 위한 연구이다. 제안된 방법은 다중프로세서 시스템에서 각 제어기가 스스로를 검사하는 내부검사와 다른 제어기를 검사하는 외부검사인 고장탐지, 고장의 전파를 막기 위한 고장분리, 고장시스템에 안전한 입력을 주기위한 동작 및 고장진단으로 구성된다. 이러한 방법은 VME버스에 여러개의 CPU보드, 인터페이스 보드, 그리고 그래픽 시스템으로 구성된 UNIX 시스템에서 실험하였으며, 이는 보일러 백업제어기에 적용하였다.

Abstract

The reliability enhancement is the critical issue in many computer applications, particulary in process control system. In this paper we describe how to achieve the reliability improvement in control system which is based on multiprocessors. The proposed method is accomplished by using the techniques of fault detection which composed by internal and external fault detections, fault isolation for removing the fault propagation, safety action for driving safe input, and fault diagnosis.

This approach is experimented and adopted in boiler backup control system constructed by VMEbus system, CPU boards, graphic system, and other interface boards with UNIX operating system.

I. 서 론

최근에 이르러 반도체 및 컴퓨터 분야의 발전으로 자동화 시스템이 복잡하고 대규모화 되어가고 있으며 이에 따른 안전성과 신뢰도의 증가는 필수적 요소로 등장하고 있다.[1,2,3] 또한 대규모 시스템을 효율적으로 제어하기 위하여 빠르고 신뢰도가 높은 다중프로세서를 이용한 제어기에 대한 연구가 많이 수행되고 있다.[5] 특히 디지탈 제어에 있어서 주기는 짧을수록 정밀한 제어를 할수있기 때문에 빠른 계산이 요구된다. 이를 위하여 다중프로세서 방식이 채택되어 계산량을 분산시키고 있으며, 아울러 신뢰도를 향상시키고 있다. 신뢰도를 높이기 위해서는 고장탐지 기능이 요구되며, 또한 고장발생시에 적절히 대응할 수 있는 장치가 있어야 하는데, 가장 좋

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Electrical Eng., KAIST)
**正會員, 韓國科學技術大學 電子電算學部
(School of Elec. Eng., & Comp. Science, Korea Institute of Techology)
接受日字 : 1988年 3月 22日

은 방법은 고장이 발생하더라도 계속해서 정상적인 출력을 낼 수 있는 FT(fault tolerant) 시스템일 것이다.[6,7,8] 지금까지는 원자력 발전소나 비행기의 제어기와 같이 고장으로 인한 큰 피해가 예상되는 제어대상에서 주로 적용되어 왔으나, 최근에는 많은 분야에서 신뢰도를 높이는 문제가 요구되어서 적용영역을 확장시키고 있다.[7,9]

시스템의 신뢰도는 고장의 발생빈도 및 정도에 의해서 결정되며, 신뢰도를 개선하기 위해서는 고장의 원인을 분석하고 규명하여 사전에 이를 예방하는 고장방지 및 고장 발생시 그 영향을 최소로 줄여 시스템이 정상적인 동작을 계속하도록하는 FT방법이 있다.[2,4] 고장 방지 방법은 경험 및 분석에 의하며 한계가 있어서 고장을 완전히 없앨 수는 없다. 이러한 이유에서 이보다 적극적인 방법으로 FT시스템을 구성하는 방법이 최근에는 많이 쓰이고 있다.[5,7]

FT시스템을 구성하기 위해서는 여분의 시스템을 지녀야 하며, 기본적인 과정은 첫째, 고장의 발생을 탐지하는 고장탐지과정 둘째, 고장이 어느 부위에서 발생 되었나를 판단하는 고장진단과정 셋째, 고장이 다른 영역에 확산되지 않도록 고장부분을 고립시키는 고장분리과정 네째, 고장이 발생한 부분을 분리시킨 후 여분의 하드웨어나 소프트웨어를 고장이 발생한 부분과 대치시키는 재 구성과정 다섯째, 재구성이 끝나고나서 고장의 영향을 제거시키고 고장이 발생하기 이전의 상태로 최대한 전환하는 고장회복과정 여섯째, 고장난 부분을 수리하여 다시 원래 상태의 시스템으로 환원 시키는 재 실장과정으로 되어 있다.[2,4]

위의 여섯 단계에 의해서 일반적인 컴퓨터의 FT 시스템이 구성되며, 이를 기존의 제어기에 적용하기에는 많은 변형이 요구되며, 또한 많은 비용이 요구되어 어려움이 따른다. 제어기에서는 특히 고장에 의해서 출력된 데이타는 회복할 수 없기때문에 고장회복 기능은 FT 시스템에서 고장이 발생한 후 재구성에 의하여 정상동작을 할 수 있을때 다음에 출력될 데이타만을 수정해서 내어보낼 수 있다. 그러나 비용 및 변형의 어려움 때문에 FT시스템을 구성할 수 없을때, 그리고 또한 고장시에 큰 영향을 주는 시스템에는 고장탐지 및 안전대처에 의하여 고장이 주는 영향을 줄여야한다. 이러한 관점에서 제어기의 고장에 의한 영향을 고려하여, 고장시에 최종적인 값을 유지할 것인지, 끊어서 제어기를 분리시켜 영향을 없앨 것인지를 미리 정하여 준대로 처리하여 과도한 제어를 막을 수 있다.

본 논문에서는 기존의 제어기에 간단히 첨가할 수 있도록 주로 하드웨어에 의한 고장을 다루게 되며, 소프트웨어적인 고장은 일반적으로 다루어질 수 있는 범위에서만 다룬다. 또한 소프트웨어 및 하드웨어의 고장을 모두 탐지하여 여분의 시스템으로 대치하여 FT 시스템을 구성해야 하지만 기존의 제어기가 대부분 여분의 하드웨어가 없다. 따라서 FT시스템을 구성할 수 없기 때문에 여기서는 고장탐지 및 고장진단 그리고, 고장에 인한 위급한 상황에서 처리하는 방법으로 제한하고 차후의 확장에 의하여 여분의 시스템을 둘 수 있을 때를 대비하여 고장이라는 신호를 준비하여 여분의 시스템이 이 신호에 의하여 대치할 수 있게한다.

고장탐지를 위해서 여러가지 테스트가 필요하며 따라서 테스트 시간이 요구된다. 필요한 시간을 할당하는 방법에는 컴퓨터의 FT 시스템에 주로 이용되는 일정한 시간마다 전체 시스템의 동작을 점검하는 방법이 있는데, 이러한 방법은 높은 우선순위의 인터럽트에 의하여 점검이 되기 때문에 시스템의 정확한 상태를 일정시간마다 점검할 수가 있어서 좋으나 실시간 제어에 있어서 일정하게 유지되어야 하는 주기가 변화할 우려가 있다. 그러나 이를 위하여 충분한 시간적 여유를 두어서 주기를 결정할 수도 있으나, 주기는 빠를수록 좋기 때문에 좋은 방법이라고는 볼 수 없다. 따라서 여기서 제안하는 방법은 그림 1과 같이 디지탈 제어의 특성을 이용하여 한 주기 내에서 제어기의 동작이 끝나고, 다음 주기를 기다리는 시간에 자체 고장 탐지 기능을 수행하게 한다. 이때 제어기의 동작은 그림 2와 같은 형태가 된다.

고장이 탐지되고난 후 그 고장이 어디에서 발생되었는 지를 판단하여 고장수리에 편이를 주는 고장진단기능 또한 중요하며 정확한 진단은 수리에 요구되

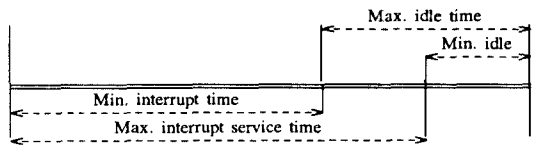


그림 1. 일반적인 디지탈 제어기의 타이밍
Fig. 1. The timing of a digital controller.

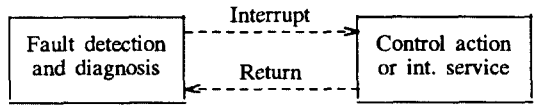


그림 2. 디지탈 제어의 흐름도
Fig. 2. The flow chart of a digital control.

는 시간을 줄이는데 매우 큰 역활을 한다.[2],[10]

본 연구는 보일러의 기존 제어기에 여분의 제어기를 첨가하여 콜드 스탠바이(cold standby) 시스템을 구성하는데,[11],[17] 있어서 요구되는 보일러 모델링 부분,[14] 모델과 실제의 입출력 신호로부터 고장을 판별하는 고장검출,[12],[16] 고장이 검출된 후 고장부분을 여분 시스템으로 대치시키는 여분 스위칭 부분으로[13] 이루어지는 기존의 여분제어기에서 이 제어기의 신뢰도가 매우 높아야 하기 때문에 이에 접합하도록 여분 시스템의 자체적인 고장검출 및 진단기능을 첨가하였다. 특히 여분의 시스템에 고장검출 및 보일러 모델 등의 고장을 검출할 수 있는 기초적인 데이타를 가지고 있을 뿐만 아니라 백업기능을 같이 가지고 있고 또한 여분 시스템의 고장은 전체 제어기에 심각한 영향을 줄 수 있기 때문에 신뢰도가 높아야 하며 또한 만약의 사태에 고장이 발생하였을때 대처할 수 있도록 여분 시스템 자체의 신뢰도를 높이는 방안 및 고장시의 고장정보 및 대처방안에 대해서 연구하였다.[15]

Ⅱ. 다중 프로세서에서의 고장탐지

1. 다중 프로세서의 기능별 분할

고장탐지 및 진단을 위해서는 시스템을 여러 부분으로 나눌 필요가 있다.[9] 또한 다중 프로세서이기 때문에 각각의 프로세서가 별개의 ROM 및 RAM등을 가진다고 할 때 그림 3 과 같으며 고장탐지를 위해서 각각의 프로세서 모듈들이 그의 주변장치들을 테스트하는 자체 테스트와 다른 모듈들의 주변장치들을 테스트한 결과로부터 고장여부를 판별하는 외부 테스트로 구성되어 있으며, 각각의 모듈들은 서로서로 CM (common memory) 을 거치지 않고는 통신할 수 없기 때문에 각 모듈들의 테스트한 결과를 CM에 모두 기록하고 그 결과로부터 외부 테스트 및 최종 진단을 할 수 있도록 한다.

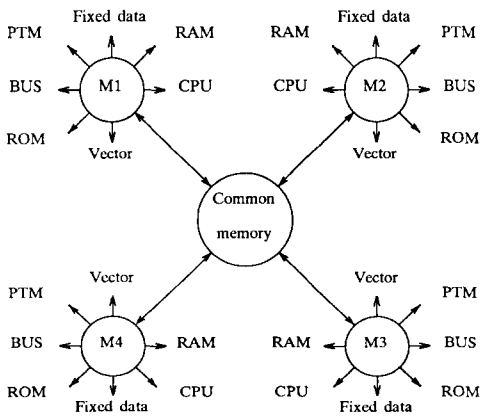


그림 3. 다중 프로세서의 테스팅 그래프
Fig. 3. Testing graph of multiprocessors.

고장탐지를 할 때 가장 주의가 요구되는 것은 제어기의 동작에 영향이 없어야 한다. 즉 테스트를 하면서 제어기에 필요한 데이타를 변화시키거나 주기에 영향을 주어서는 안된다. 대상으로 하는 제어기는 GB (global bus) 에 각각의 프로세서 모듈이 연결되어져 있으며, 또한 CM도 이 버스에 연결되어 있다.[17],[18]

2. 자체고장탐지

자체고장탐지는 각 모듈들이 그의 주변장치들을 테스트하는 것으로 앞의 모델에서와 같이 구분하여 각각을 테스트한다. 이때 테스트를 위하여 주변장치들을 다루게 되는데 특히 RAM과 같은 경우 제어기의 동작을 위한 자료가 들어 있을 수 있으므로 항상 먼저 다른 장소에 보관한 후 테스트를 하고, 테스트 도중 인터럽트에 의하여 제어기의 동작을 시작할 수 있기 때문에 데이타가 변화하기 직전에 인터럽트가 걸리지 못하게 한 후 테스트를 하고 데이타를 원상복귀 시킨 후 인터럽트가 동작할 수 있게 하여 항상 올바른 데이타로써 제어기가 동작하게 해야 한다. 그 각각의 테스트는 다음과 같이 수행된다.

CPU는 모듈 내에서 가장 중요한 부분으로서 모든 동작을 관장하고 있으며 주변장치들을 테스트하기 때문에 가장 면밀히 다루어져야 한다.[9] CPU의 구성은 그림 4 와 같이 크게 분류할 수 있으며 CPU 내부에서 데이타의 보존 및 전달기능을 테스트하는 레지스터 테스트, 그리고 산술 논리동작 테스트, 명령어 해석 동작 테스트, 예외 기능 테스트 등에 의하여 CPU의 동작을 점검한다. 레지스터는 모든 데이타를 다루는 것으로서 정밀한 점검이 요구되며, 모든 데이타 및 어드레스 레지스터를 "FFFF7FFF" 와 같이 오직 한개만이 "0"인 32가지의 데이타로써 그의 보수와 함께 전체를 돌고나서 두 내용을 더하여 "−1"이 되는가를 확인하고 다시 "10000000" 와 같이 오직 한개만이 "1"인 32가지의 데이타로써 그의 보수와 함께 전체를 돌고나서 두 내용을 더하여 "−1"이 되는가를 확인하여서 두가지 중에서 한번이라도 틀리면 CPU내의 레지스터 고장으로 판별되어서 테스트된 시간 및 모듈번호 그리고 고장번호를 CM에 기록하고 진단에 들어가게 된다. 산술 논리동작 테스트는 CPU가 수행할 수 있는 모든 산술동작에 특정 데이타를 인가하여 그 결과가 정확한 지

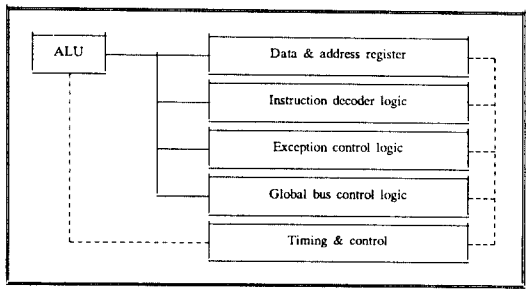


그림 4. CPU 내부의 검사를 위한 모델
Fig. 4. Internal block diagram of CPU.

를 검사한다. 산술동작은 MC68000 의 경우에는 AND, OR, NOT, BSET, BCLR, ASL, ASR, LSL, LSR, ROR, ROL, SWAP, ADD, SUB, MUL, DIV, TEST, CMP 등이 있다. 명령어 해석 동작 테스트는 앞의 두 가지 테스트에서 여러가지 명령어의 테스트가 동시에 이루어 졌으므로 여기서는 어드레싱 모드만을 테스트한다. 어드레싱 모드가 여러가지가 있기 때문에 같은 번지를 여러가지 방법으로 지정하여 그 내용이 일치하는 가를 점검한다. 어드레싱 모드에는 register addressing, immediate addressing, direct addressing, indirect addressing, index addressing, relative addressing 등이 있다. 그리고 예외 테스트는 MC68000 프로세서가 가지고 있는 여러가지 예외 동작을 이용하는 것으로서 하드웨어가 준비되지 않는 장소를 읽거나 또는 쓸 때 (bus trap exception), 잘못된 어드레스 즉 홀수 어드레스를 읽거나 쓸 때 (address trap exception), 정의되지 않은 명령어를 수행시킬 때 (illegal instruction trap exception), "0"으로 어떤 수를 나누었을 때 (zero divide error exception), 수행할 수 없는 명령어 즉 관리자 모드에서만 사용할 수 있는 명령어를 사용자 모드에서 사용하였을 때 (privilege violation exception) 등의 고장을 점검할 수 있는 지를 확인하고 또한 특수한 명령어 즉 CHECK Instruction Exception, Trap Overflow Exception, TRAP #0~TRAP #15 Instruction Exception 등의 수행을 점검하며, Trace 기능을 또한 확인한다. 이러한 예외 테스트가 잘 수행될 수 있는 지를 확인하기 위하여 그 원인이 되는 요소를 소프트웨어적으로 발생시킨 후 그 예외 동작이 잘 수행되었는지를 점검한다. 단 인터럽트는 PTM 테스트에서 다룬다.[9],[19],[22]

데이타 버스 테스트는 데이타 버스선 중에서 단락 또는 개방된 것이 있는지를 점검한다. 이는 특정한 번지에 레지스터 테스트에서와 같이 16가지의 데이타를 한 비트의 차이가 나게하여 각각 쓰고나서 다시 읽어 보았을 때 쓴 값과 읽은 값의 차이를 가지고 어느 비트가 고장인가를 판단할 수 있다. 또한 주변장치의 연결 부위에서 고장이 발생할 수도 있으므로 주변장치에 맞게 여러곳을 점검하여야 하며 여러가지로 쓰고 읽을 수 없는 주변장치는 별도로 그 주변장치에 맞는 테스트를 하여야 한다.

Flow of data bus test

a) A 0 ← Testing address
   A 1 ← A 0 + 2
   D 0 ← "FFFE"
   D 1 ← "0001"
b) D 5 ← (A 0)
   D 6 ← (A 1)
   Interrupt disable
   (A 0) ← D 0
   (A 1) ← D 1
c) D 2 ← (A 0) + (A 1)
   (A 0) ← D 5
   (A 1) ← D 6
   Interrupt enable
   if (D 2 ! = "FFFF") go to g)
d) shift left D 0
   D 1 ← complement D 0
e) if (D 0 ! = 0) go to b)
f) D 0 ← "0"
   go to h)
g) D 0 ← "−1"
h) return (D 0)

어드레스 버스 테스트는 위의 데이타 버스 테스트와 아주 유사하며 "FFFF" 번지를 기초로 하여 한 비트씩 "0" 으로 바꾼 번지 중 한쪽에 먼저 데이타를 쓰고 다른쪽에 다시 데이타를 쓴 후 먼저 쓴 데이타를 읽어보면 그 번지가 잘 지정되었는 지를 알수 있다. 즉 그 비트가 단락이나 개방이 되었다면 예를들어 "FEFF" 번지에 특정한 데이타를 쓰고 "FFFF" 번지에 다시 먼저 쓴 데이타와 다른 데이타를 쓴 후 다시 "FEFF" 번지를 읽으면 "FFFF" 번지에 쓴 데이타가 읽혀질 것이다. 따라서 이 선에 고장이 발생하였음을 알 수 있다. 이 또한 주변 장치에 따라 여러번 수행을 해야한다.

CSUM (check sum) 테스트는 기존의 데이타가 운전도중 잘못되어진 데이타는 없는 지를 알아보기 위해서 사용되며 테스트하고자 하는 모든 데이타를

워드 또는 바이트 단위로 모두 더하여 그 값이 "0"이 되게 하여서, 후에 다시 CSUM 하였을 때 "0"이 아니면 변화가 발생되었으며 가장 변화를 적게 탐지할 수 있는 MSB는 4번 이하의 고장을 찾을 수 있으며 LSB쪽으로 갈수록 2배씩의 에러까지도 검사할 수가 있다. 다른 방법으로는 XOR(exclusive or)를 써서 모든 데이타를 바이트 또는 워드 단위로 XOR 하여 "0"이 아닌가를 판단하면 되나 이 때는 같은 비트의 고장이 두번 일어나면 고장으로 탐지되지 않기 때문에 고장이 빈번한 경우에는 사용하기가 곤란하지만 테스트에 소용되는 시간이 적어서 많이 쓰인다, 그러나 여기서는 CSUM 방식을 쓰며, XOR 방식 보다 테스트하는데 소요되는 시간이 많은 단점이 있다. 주로 ROM 데이타 및 중요한 데이타 블럭에 데이타를 쓸때 두 개의 워드를 별도로 두어 CSUM 데이타가 "0"이 되도록 항상 만들어 주면 테스트할 수 있다.[19]

PTM (programmable timer)은 CPU에 인터럽트를 걸게하여 매 주기의 제어기 동작을 하도록 인터럽트를 발생시키는 장치로서 이 장치의 테스트를 위하여, 인터럽트 서비스 루틴을 테스트를 위한 것으로로 바꾸고 한 주기 정도의 충분한 시간을 기다린 후 인터럽트 루틴이 수행되어졌는 지를 검사한다. 이때 테스트를 위한 인터럽트 루틴은 제어에 미치는 영향을 줄이기 위하여 인터럽트 동작이 정상이라는 표시만 하고, 이 루틴이 비록 인터럽트 서비스 루틴이지만 수행이 끝났을때에도 계속하여 인터럽트가 기다리는 결과가 되도록 하고, 또한 정상적인 제어동작을 할 수 있도록 서비스 루틴을 원상태로 바꾸어 준 후 빠져나온다. 이는 PTM 테스트가 제어기 동작의 시간을 아주 적게 할애하여 인터럽트가 잘 동작하는가를 판별할 수 있다. 또한 초기 테스트를 별도로 두어 제어기가 처음 동작을 시작하기 전에 인터럽트에 관련된 모든 장치들을 세부적인 부분으로 나누어서 정밀히 테스트 한 후에 정상적인 동작으로 들어가게 한다.[23]

RAM은 비교적 크기 때문에 CPU의 레지스터 테스트와 같은 방법을 사용하기에는 많은 시간을 요구하기에 RAM 자체를 여러 부분으로 나누어서 테스트하며 시간 절약을 위해서 4바이트를 동시에 테스트한다. 먼저 "1010…10"의 형태로 쓴 후 검사하고 또한 "0101…01" 형태의 데이타로서 다시 테스트 한다. 이때 stuck-at-1과 stuck-at-0를 테스트 할 수 있으며 인접한 데이타와의 단락 개방 또한 테스트가 된다. 이때 테스트하는 동안에는 인터럽트가 동작하지 못하게 한 후 테스트하게 되는데 아무리 작은 불럭으로 나누었다고 해도 테스트하는 시간이 제어기에 영향을 줄 수 있기 때문에 4바이트의 테스트가 끝나면 인터럽트가 걸릴 수 있게 하여 외부에서 인터럽트가 걸린 상태였다면 바로 제어기의 동작을 할 수 있게 하고, 다시 인터럽트가 걸릴 수 없게 한 후 계속하여 다음 테스트를 수행한다.

CM은 모듈 내의 RAM과는 차이가 있으며, 여러 모듈이 동시에 읽고 쓰기 때문에 제어기가 사용하는 영역을 테스트하기 위해서는 인터럽트 및 상호 표시를 잘 이용해야 하며, 또한 복잡한 알고리듬이 요구되고 제어기의 동작에 상당한 영향을 주게 된다. 이러한 복잡한 알고리듬을 사용하였을 경우 기존 제어기에 적용할 때 많은 변화가 요구되어 여기서는 CM 부분을 제어 동작에서 하나의 모듈만 쓰고 읽는 영역을 결정하여 나눈 후 이 영역을 RAM 테스트와 같은 방법으로 테스트한다. 이때 제어동작에서 공용으로 사용되어지는 영역을 여러 부분으로 나눈 후 그 사이사이를 CM 테스트 하면 더 효과적인 고장탐지가 된다.

이와 더불어 MC68000에 내장되어 있는 고장탐지 기능인 예외 처리가 있으며, 이 기능에 의하여 소프트웨어 및 하드웨어의 고장중 많은 부분이 검출되어질 수 있다. 이의 동작은 앞에서 설명하였다.

3. 외부 테스트

외부 테스트는 각 모듈들의 동작 상태를 테스트하는 것으로, 각 모듈들은 모든 테스트 결과를 CM에 적어놓고 다른 모듈들의 테스트 결과를 가지고 그 모듈이 전체적으로 정상적인 동작을 하는지 테스트한다. 외부 테스트는 두 가지로 구성되며 시간초과 고장과 전갈형태 고장이 있다. 다중 프로세서 시스템에서 한 모듈이 고장으로 인하여 스스로가 고장이라는 정보를 다른 프로세서에게 알려줄 수 없을 때나 또는 스스로가 고장인지를 알 수 없을 때 다른 모듈이 그 모듈의 동작을 점검하여 고장여부를 판단한다.[10]

시간 초과는 각 모듈들이 내부에 다른 모듈들의 현지 시간과 그 시간으로부터의 경과를 데이타로 가지고 있으면서 다른 모듈의 동작하지 않은 시간을 검사하여 일정 시간을 초과하면 시간초과 고장으로 처리한다. 각 모듈은 자체 테스트를 수행하며 매 테스트가 끝나면 그 결과를 CM에 기록하는데, 이때 자신의 시간을 같이 기록하여 주게 된다. 따라서 한 모듈이 다른 모듈의 시간을 읽어서 그 값의 변 화가 어느 일정 시간 이상 없으면 고장이다. 이는 모듈이 전혀 동작을 하지못할 때도 발생하고, 또한 한 주기

내에서 남는 시간이 없을때 자체 테스트를 거의 수행할 수 없기 때문에 한가지를 테스트하는 시간이 너무 많이 소요되어서 시간초과가 된다.

전갈 형태고장은 CM에 기록된 테스트 결과가 주어진 형태에 맞게 구성되어져 있는지를 검사하게 된다. 즉 데이타 형태를 여분이 많게 특수한 몇가지만 존재하게 하여 그 나머지는 전갈 형태 고장으로 한다. 이 전갈 형태 고장은 재 테스트에 의하여 바로 고쳐지게 되며 재 테스트에 응답이 없으면 완전히 동작하지 않는 것으로 간주한다.

Ⅲ. 다중 프로세서에서의 고장 진단

CM에 기록된 테스트의 결과는 각 모듈별로 되어 있으며, 또한 각각의 테스트는 고유한 테스트 번호를 가지게 된다. 모든 테스트를 차례로 모두 테스트한 후 다음을 반복하는 것이 고장의 최대 잠복 시간이 짧아서 좋으나, 중요한 부분에서 고장의 잠복 시간이 길면 문제가 있으므로 중요한 부분은 매번 테스트하며 ROM 데이타와 같은 거의 고장이 나지 않는 것은 어는정도 시간이 경과 한 후 한번씩 테스트한다. 이러한 것을 고려하여 그림 5 와 같이 많은 시간을 요구하는 것은 여러부분으로 나누어서 테스트하였다.

그림 6 과 같이 각 상태 변화가 발생하여, 매 자체 테스트가 끝나면 그 결과를 CM에 기록하고 고장이 탐지되면 재 테스트를 하여 정상이면 외부 테스트를 수행한다. 재 테스트에서 다시 고장이면 CM에 고장이라고 쓰고 다시 외부에서 테스트 요구가 있는 지를 확인 한다. 이는 만약 고장이면 CM에 쓰는 결과가 정확하지 못할 경우도 있기 때문에 테스트 요구에 의해서 확인을 하게 된다. 또한 외부테스트에서

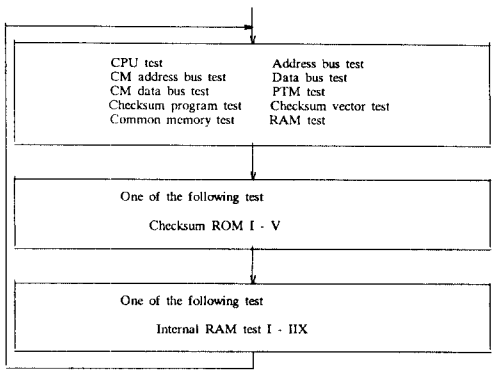


그림 5. 자체 테스트의 흐름도
Fig. 5. Flow chart of selft testing.

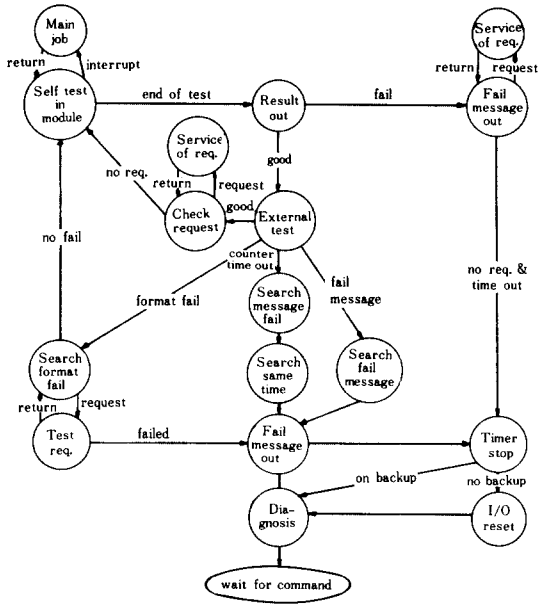


그림 6. 고장진단 상태도
Fig. 6. State diagram of fault diagnosis.

전갈 형태 및 시간 초과를 검사 한 후 다른 모듈의 고장이 있는지를 검사하여 분명한 고장이 있으면, 이 고장이 전체 동작에 크게 영향을 미치는 것인지를 판별하여 계속 운전할 수 없으면 제어기를 멈추게 하여야 하므로, 인터럽트가 동작하지 못하게 한 후 적절한 동작에 의해서 위급한 상태가 발생하지 않게 처리한 후 진단에 들어가게 된다. 이때는 시간적 여유가 충분하기 때문에 모든 데이타를 검색 할 수 있다. 진단을 위해서는 3 개 이상의 모듈이 있어야하며 모듈 수가 많을수록 진단은 정확하게 할 수 있다. 모든 모듈은 자기 모듈의 테스트 결과를 검색하여 2 가지 종류 이상의 고장이 찾아지면 진단을 포기하게 되며, 이는 자기 모듈이 고장이라고 다른 모듈이 이 모듈을 고장으로 처리하게 될 것이다.

한 모듈이 자체 테스트 또는 외부 테스트에서 고장이 검출되면 그 모듈 및 테스트 번호를 모든 모듈들에게 알리게 되는데 이 전갈에 의하여 다른 모든 모듈 또한 외부 테스트에서 고장이 검출되기에 한 모듈이 고장을 검출하면 모든 모듈이 그 고장을 알 수 있으며 각 모듈은 또다시 모든 테스트 결과를 검색해 본 후 고장으로 된 테스트를 찾아서 다시 모든 모듈에게 알리게 된다. 그리고 가장 신뢰도가 높은 모듈은 다른 모듈을 검사하여 고장을 알리지 않은 모듈이 있으면 동작을 하지 않은 모듈로서 별도로

처리하고, 그 나머지 모듈에서 서로 같은 정보를 지닌 것을 검사한다. 이때 같은 정보가 가장 많은 것의 모듈 및 테스트 번호로부터 어떤 모듈의 어떤 테스트가 고장인지를 안다. 만약 동수인 경우는 이들 중에서 미리 선정된 순서에 따라 신뢰도가 높은 순서에 의하여 신뢰도가 높은 모듈의 결과를 따른다.

이러한 방법은 고장이 탐지되면 바로 처리가 되어지기 때문에 2 개 이상의 고장이 발생할 가능성은 아주 희박하며 발생하였다면 주로 이 모듈에 고장이 발생하여 오동작에 의해서 다른 고장을 발생시킨 경우 이므로 이는 전체 테스트 결과에서 2 개 이상의 고장이면 고장을 검출한 모듈은 진단을 포기하게 하여 피한다.

Ⅳ. 다중 프로세서에서의 고장처리

위와 같은 동작 전체를 다루기 위하여 여러가지 모드가 필요된다. 다시 말하면 항상 모드를 확인하여 그 모드에 의하여 동작이 된다. 처음 운전을 시작하기 전에 테스트 모드에 의해서 세부적으로 테스트를 수행하며, 그 결과를 관리자에게 알려주게 하여 전체 시스템이 정상적인 동작을 할 수 있는 지를 판별하고, 고장이 있으면 정밀히 알아보기 위하여 모니터 모드에 의해서 각 모듈의 동작을 점검할 수 있는 여러가지면을 제공 받으며, 시스템 전체가 정상이면 운전 모드에 의해서 자체 테스트를 수행하면서 인터럽트에 의한 제어 동작이 동시에 수행되게 된다. 고장이 발견되면 출력을 그대로 유지할 것인지 아니면 초기화 시킬것인지를 미리 정의하여 준 것으로부터 판별하여 동작을 시킨 후 고장진단에 의하여 어느 고장인지를 알려준다. 그리고 다시 명령어 모드로 와서 사용자가 모드를 선택하여 다음 동작을 할 수 있게 한다. 또한 고장이 아닐때라도 명령어는 읽어들일 수 있도록 외부 테스트를 한 때마다 항상 명령어를 점검하여 다음 동작을 결정한다. 이때 다중 프로세서로 이루어진 시스템이므로 전체를 관장하는 것은 그 중에서 어느 하나가 하게 될 것이며 고장에 의한 모든 정보는 그곳으로 집중되어져야 한다. 따라서 모든 모듈은 관리자에게 주는 정보를 CM에 넣고서 그 표시를 특수한 메모리 영역에 기입하여 관리자 모듈이 읽어갈 수 있게 한다. 그러나 여러 모듈이 동시에 같은 곳에 정보를 쓸 수 있으므로 이의 제어를 위하여 특정 메모리에 표시를 하여 한 모듈이 쓰고 있을 때는 다른 모듈은 대기하게 하고, 끝났을 때 그 뒤에 이어서 쓰게 한다. 그리고 다 쓴 후에는 읽어가도 좋다는 표시를 하여 읽어가게 한다.

고장의 정보에 대한 최종 진단 결과는 오직 하나만 있을 수 있다. 따라서 여러 모듈이 최종 진단한 결과를 관리자에게 알려줄 때 스스로의 진단이 끝났으면 다른 모듈이 먼저 진단결과를 기록하였는 지를 검사하고, 먼저 기록하였다면 그의 우선순위를 확인하여 스스로의 우선순위가 더 높으면 이미 기록된 데이타를 무시하고 다시 기록하며, 더 낮으면 기록을 포기한다. 스스로의 최종진단결과를 기록하였다면 얼마의 시간이 경과하여도 그 결과를 바꾸는 모듈이 없으면 읽어가도 좋다는 표시를 한다.

Ⅴ. 실험결과

실험을 위해서 그림 7 과 같이 VME 시스템에 CPU 모듈로서 MVME 110－1 2 개와 MVME117 1개 그리고 CM과 시스템 제어기로서 MVME050을 사용하였다. 이 3 개의 모듈에서 각각이 고유한 제어동작을 수행하면서 자체 및 외부 테스트를 수행하게 하였다. 그리고 또한 SYS121이 전체를 관장하여 명령어를 내리게 하였다.

실험을 위하여 몇가지 고장을 발생시켜 보았는데 1 가지 씩의 고장을 발생시켰을 때는 정확히 찾아 내었다. 그러나 동시에 5 가지 이하의 전갈형태 고장을 발생시켰을 때는 모든 결과를 재 테스트에 의하여 회복시킬 수 있었다. 그러나 그 이상은 회복도중 시간초과 고장으로 되었다. 또한 여러개의 고장을 가하기 위하여 테스트의 결과를 고장으로 바꾸어서 여러개를 한꺼번에 가하였을 때는 각각의 모듈이 나타내는 고장이 모두 달라서 그 중 신뢰도가 가장 높은 모듈이 선택한 고장을 추측하여 나타내었다. 또한 CM의 IC 한개를 뽑았을때 CM고장, 또는 CM ABUS 고장, CM DBUS 고장의 3 가지 중에 한 가지로 나왔다. 그리고 하드웨어의 단락이나 개방에서는 BUS TRAP ERROR 또는 ADDRESS TRAP ERROR 또는 ILLEGAL instruction error등의 고장이 검출되었으며 관리자 모듈에게 잘 전달되는 경우도 있었으나, 아무런 동작을 할 수 없어서 완전한 고장으로 처리되는 경우가 더 많았다. 이때 완전한 고장은 다른 모듈이 이 모듈을 응답이 없는 고장으로 하여 진단할 것이다.

처음에 모든 테스트를 한꺼번에 수행시켰을 때 10〔sec〕 정도 소요되었다. 그러나 시간이 많이 소요되는 몇가지를 여러개로 나누어 한번 수행에 하나씩 테스트하였을 때 한번 수행하는데 0.1〔sec〕 정도 소요 되었다. 하지만 RAM 및 ROM 그리고 프로그램 CSUM테스트에서 역시 많은 시간이 소요되었다. 고장진단은 고장이 발생하면 검사에서 탐지되는 즉시 그래픽에서 고장이라고 나타나며 1〔sec〕정도 후에 어느 모듈의 어떠한 고장이라는 표시가 나타났다.

본 실험에서 안전제어를 위해서 이 여분의 다중프로세서 제어기를 고장이 발생하였을 때 여분 제어기가 백업동작을 하고 있으면 최종값을 유지하게 하였으며, 원래의 고장을 탐지만 하고있고 백업은 하고 있지 않을 때 이 여분 제어기가 고장을 탐지하여도 백업동작으로 들어가지 못하도록 분리시키게 하였으며, 이러한 동작은 GB가 고장이 아니면 언제나 잘 수행되었다.

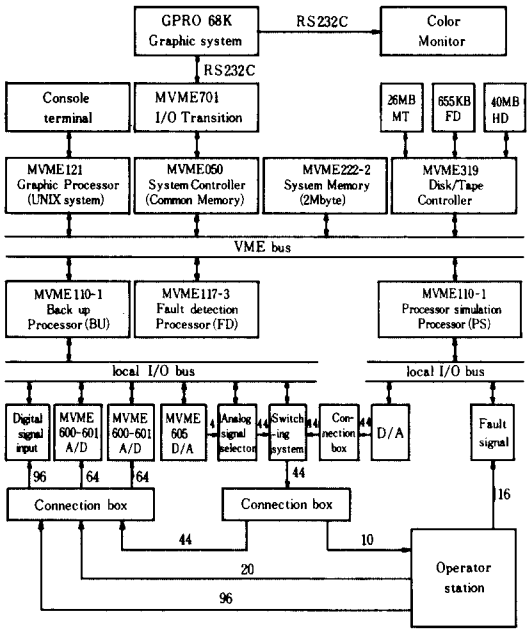


그림 7. 전체 시스템 구성도
Fig. 7. Overall system structure.

Ⅵ. 결 론

본 연구에서는 기존의 여러가지 FT 알고리듬에서 시스템에 의존하는 부분으로 언급하지 아니한 디지탈 제어기에서의 고장탐지에 대하여 연구하였다. 다중프로세서의 신뢰도를 개선하기 위하여 디지탈 제어의 특징을 이용, 주된 제어기의 동작을 모두 인터럽트 서비스 루틴으로 처리하고 그 밖에 모든 시간을 고장 탐지에 할애하였다. 또한 하드웨어 및 일부 소프트웨어의 고장을 탐지하게 하였으며, 진단에 의하여 고장 부위를 찾았다. 여분의 시스템이 있으면 고장 탐지에 의하여 재 구성을 하여 모든 고장에 대하여 운전을 계속할 수 있는 FT 시스템을 구성할 수 있을 것이다.

이러한 방법은 현장에서 쉽게 적용이 가능하며 기존의 제어기에서 변형을 가하지 않고 단지 소프트웨어 만을 첨가하여 고장 탐지 및 진단을 할 수 있다. 또한 첨가하는데 비용이 들지 않기 때문에 현장의 제어기에 첨가하기가 매우 용이하다. 이러한 방법으로 바꾸었을 때 고장 발생시에 오동작을 줄여서 안전한 시스템으로 할 수 있으며 차후의 FT 시스템 구성을 위하여 선행단계로도 볼 수 있다.

앞으로 더 연구가 필요한 부분으로서는 고장 탐지에서 시스템에 적합한 테스트를 더 첨가하고, 또한 여분의 CM 및 버스를 두어서 완전한 FT 시스템을 구성할 수 있을 것이며, 실 시간으로하여 고장발생 시간 또한 명확히 판별할 수 있을 것이다.

그러나 다중프로세서를 이용한 기존 제어기에 쉽게 적용하기 위하여 FT 시스템으로 하기는 어려우며, 단지 여러가지 고장 탐지 기능 및 각 시스템에 적합한 고장 대처 기능을 첨가하여 안정되고, 또한 고장시에 안전을 보장하는 시스템으로 만들 수 있을 것이다.

參 考 文 献

[1] David A. Rennels, "Fault-Tolerant Computing Concept and Examples," *IEEE Trans. on Computer* c-33, pp. 1116-1129, 1984.

[2] Barry W. Johnson, "Fault-Tolerant Microprocessor-based Systems," *IEEE Micro* December, pp. 6-21, 1984.

[3] Ormi Serlin, "Fault-Tolerant System in Commercial Application," *IEEE Computer* August, pp. 19-30, 1984.

[4] 이헌, "Fault-Tolerant Computing System," 전자교환기술, 제 1 권, 제 1 호, pp. 46-61, 1985.

[5] Jon G. Kuhl, S.M. Reddy, "Fault Tolerant Consideration in Large, Multiple-processor Systems," *IEEE Computer*, Mar. pp. 56-67, 1986.

[6] E. Schrodi, "Fault-Tolerant and Fail-Safe Microcomputer Systems," IFAC, pp. 183-188, 1984.

[7] L.Y. Lu, "Fault-Tolerant Programs in Process Control Systems," IECON, pp. 877-882, 1984.

[8] Patric P. Fasang, "A Fault Detection and Isolation Technique for Microcomputers," *IEEE Test Conference*, pp. 214-219, 1982.

[9] Dhanajay Brahme, Jacob A. Abraham,

"Functional Testing of Microprocessors," *IEEE Trans. on Computer*, c-33, pp. 475-485, 1984.

[10] S.H. Hosseini, Jon G. Kuhl Sudhakar M. Reddy, "A Diagnosis Algorithm for Distributed computing Systems with Dynamic Failure and Repair," *IEEE Trans. on Computers*, c-33, pp. 223-233, 1984.

[11] 조현용, "화력발전소를 대상으로 한 백업 콘트롤 시스템의 연구," KAIST, 석사학위 논문, 1986.

[12] 문봉채, "선형 동적시스템에서의 고장 진단 알고리듬, "KAIST, 석사학위 논문, 1986.

[13] 정광균, "화력발전소용 보일러 제어기를 위한 백업콘트롤 시스템에 관한 연구," KAIST, 석사학위 논문, 1987.

[14] 황동환, "보일러 제어 시스템의 시뮬레이터를 위한 모델구성에 관한 연구," KAIST, 석사학위 논문, 1987.

[15] 신영달, "Boiler Backup Control을 위한 Multiprocessor 방식에서의 신뢰도 개선에 관한 연구," KAIST, 석사학위 논문, 1987.

[16] 문봉채외 3인, "선형동적 시스템에서의 고장 진단 알고리듬," 대한전자공학회논문지, 제23권 6호, 1986.

[17] 김지홍외 3인, "발전소 보일러 제어기에 적용한 Fault Tolerant Control System의 연구," 대한전자공학회논문지, 제24권 1호, 1987.

[18] Motorola Inc., *"VME bus Spec. Manual,"* HD212/D, 1986.

[19] Motorola Inc., *"MVME117 Firmware Debug Monitor User's Manual,"* MVME117BUG/2, 1986.

[20] Motorola Inc., *"MVME117 VME Module Monoboard Microcomputer and MVME708-1 Transition Module User's Manual,"* MVME117/D1, 1986.

[21] Motorola Inc., *"MVME110 VME Module Monoboard Microcomputer User's Manual,"* MVME110/D2, 1986.

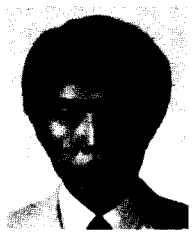
[22] Motorola Inc., *"MC68000 16/32-BIT Microprocessor Programmer's Reference Manual,"* M68000UM(AD4), fourth edition, 1986.

[23] Motorola Inc., *"MC6840 Programmable Timer Fundamentals and Applications,"* MC6840UM, 1986.

著 者 紹 介

申 永 達(正會員)

1961年 9月 26日生. 1984年 2月 경북대학교 전자공학과 공학사학위 취득. 1987年 한국과학기술원 전기 및 전자공학과 공학석사학위 취득. 현재 한국과학기술원 전기 및 전자공학과 박사과정. 주관심분야는 Flexible arm Control 등임.



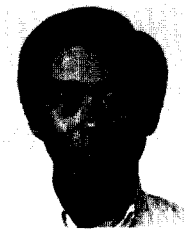
金 志 泓(正會員)

1959年 7月 9日生. 1982年 2月 서울대학교 전기공학과 공학사학위 취득, 1984年 한국과학기술원 전기 및 전자공학과 공학 석사학위 취득. 현재 한국과학기술원 전기 및 전자공학과 박사과정. 주관심분야는 고장진단 산업공정제어 및 자동화 등임.

金 炳 國(正會員)
1952年 10月 5日生. 1975年 2月 서울대학교 전자공학과 공학사학위 취득, 1977年 한국과학기술원 전기 및 전자공학과 공학석사학위 취득. 1981年 한국과학기술원 전기및 전자공학과 공학박사학위 취득. 1981年 우진계기 표준실장. 1983年 Michigan 대학 Post, Doc. 1986年 우진계기 연구실장 현재 한국과학기술대학 조교수. 주관심분야는 자동제어, 로보틱스 전산기 구조 등임.

●

鄭 明 振(正會員)
1950年 1月 31日生. 1973年 2月 서울대학교 전기공학과 공학사 학위 취득. 1976年 9月～1977年 12月 미국 미시간대학교 전기공학과 공학석사학위 취득. 1978年 1月～1983年 8月 미국 미시간 대학교 전기공학과 공학박사학위 취득. 현재 한국과학기술원 전기 및 전자공학과 조교수. 주관심분야는 로보틱스임.



卞 增 男(正會員)
1943年 10月 11日生. 1969年 2月 서울대학교 전자공학과 공학사 학위 취득. 1972年 Iowa대학 전기과 공학석사학위 취득. 1972年 Iowa 대학 수학과 공학석사학위 취득. 1975年 Iowa 대학 전기과 공학박사학위 취득. 1977年 Iowa대학 전기과 객원조교수. 1981年 한국과학기술원 전기 및 전자공학과 조교수. 1982年 Iowa대학 전기과 객원부교수. 현재 한국과학기술원 전기 및 전자공학과 교수. 주관심분야는 자동제어 이론, 로보틱스 및 인공지능, 공장 자동화 등임.