1999년도 대한전기학회 하계학술대회 논문집 1999. 7.19-21

# 이중구조의 결함내성형 마이크로콘트롤러

백승수, 이인환
(한양대학교 전자전기공학부)

# A Duplexed Fault-Tolerant Microcontroller

Seung-Soo Baek, Inhwan Lee
(Division of Electrical and Computer Engineering, Hanyang University)

**요약** - 본 논문은 이중구조의 고장허용형 마이크로콘트롤러 구조를 제시한다. 제시한 마이크로콘트롤러는 두 프로세서 모듈간의 동작을 비교하여 하나의 프로세서 모듈에 고장이 발생할 경우 콘트롤러의 외부 인터페이스를 차단한다. 이후 각 프로세서 모듈은 자체 점검을 통해 상태를 판단하고, 이상이 발견되지 않을 경우, 마이크로콘트롤러는 외부 인터페이스 기능을 수행할 주 모듈을 선정하고 다시 동작을 시작한다. 따라서 제시한 마이크로콘트롤러는 고장정지 특성 및 일시적인 고장으로부터의 회복 기능을 갖는다. 본 논문에서는 제시한 구조를 모토롤라 MC68360 프로세서와 범용 로직 IC를 이용하여 구현하고 고장주입을 통해 제시한 구조의 고장허용 특성을 확인한다.

## 1. 서 론

오늘날 산업용기기, 의료기기, 자동차제어 등의 다양한 분야에서 마이크로콘트롤러가 널리 사용되고 있다. 그리고 이러한 시스템에 대한 우리의 의존도가 높아짐에 따라 이들에 대한 신뢰도 요구조건 또한 증가하고 있다. 또한 산업용 제어시스템 등의 대형 분산 시스템에서는 흔히 기본적인 고장허용 기능을 갖는 마이크로콘트롤러 노드를 사용하여 시스템 레벨의 고장허용 기능을 구현한다. 본 논문은 이러한 응용 환경을 위한 이중 구조의 결함내성형 마이크로콘트롤러 구조를 제시한다.

시스템의 신뢰성 향상을 위한 고장허용의 기본적인 개념과 방법은 오래 전부터 연구되어 왔다[1-3]. 그리고 이러한 방법은 1960년대부터 고신뢰도를 필요로 하는 우주개발, 전화교환기, 온라인 데이터베이스 등의 분야에 활용되었다[4-6]. 고장허용에 대한 지난 40여년의 연구 결과는 고장허용의 개념은 비교적 간단하지만 이를 이용하여 실제 고장허용 시스템을 구성하는 것은 쉽지 않다는 것을 보여준다.

고신뢰도 마이크로콘트롤러는 기본적으로 외부에 틀린 결과를 제공하지 않는 기능 즉 고장이 발생할 경우 즉시 스스로의 동작을 정지시키는 고장정지 기능을 가져야 한다. 또한 대부분의 동작 중의 고장은 일시적인 결함에 의하여 발생하는 것으로 추정되므로, 고신뢰도 마이크로콘트롤러를 실현하기 위해서는 콘트롤러가 일시적인 고장으로부터 회복할 수 있도록 설계하여야 한다.

제시한 이중구조의 마이크로콘트롤러는 두 개의 프로세서 모듈의 출력을 비교함으로써 하나의 프로세서 모듈에서 발생한 고장을 발견한다. 고장이 발견되면 콘트롤러는 즉시 외부 출력을 차단하여 잘못된 결과가 외부로 전달되는 것을 방지한다. 이후 콘트롤러내의 두 프로세서 모듈은 자체진단을 통해 각각의 동작상태를 판단하고, 이상을 발견하지 못할 경우, 외부 인터페이스 기능을 수행할 주 모듈을 선정한 다음 정상동작으로 복귀함으로써 일시적인 고장으로부터의 회복을 수행한다. 이와 같은 고장 발견과 재동작이 짧은 시간 내에 반복되면 자체적으로 진단이 되지 않는 영구적인 고장이 발생한 것으로 간주하고 두 프로세서 모듈을 모두 정지시킨다. 따라서 제안한 마이크로콘트롤러 구조는 정상적으로 동작하든지 아니면 동작을 중지하는 고장정지 특성과 일시적인 고장으로부터의 회복 기능을 제공한다. 본 논문에서는 제시한 이중구조를 모토롤라사의 MC68360 프로세서와 범용 로직 IC를 이용하여 구현하고 고장주입을 통해 그 기본적인 고장허용 기능을 확인한다.

## 2. 이중구조 설계

본 논문에서 제시한 이중구조는 그림 1과 같이 마이크로콘트롤러의 기본 기능을 수행하는 프로세서 모듈, 네트워크 인터페이스, 두 프로세서 모듈의 동작을 비교하는 비교기, 외부 네트워크 인터페이스 기능을 수행할 주 모듈을 결정하는 스위치 회로, 두 센서 모듈의 동작을 동기화는 동기화 회로, 그리고 전체 회로의 동작에 필요한 클럭으로 구성된다.

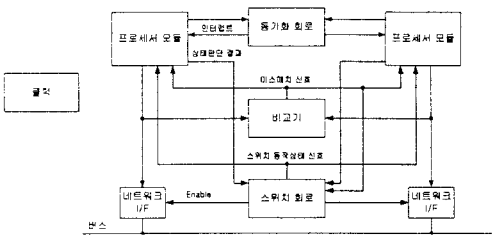


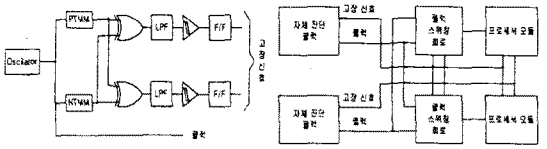

그림 1. 이중구조의 마이크로콘트롤러

비교기는 두 모듈의 외부 출력을 비교하여 프로세서 모듈의 고장을 발견한다. 만약 프로세서 모듈의 고장으로 인해 비교기에서 미스매치 신호가 발생하면, 스위치 회로는 즉시 외부 출력을 차단하고, 두 프로세서 모듈은 자체진단을 수행한다. 자체진단 수행 후 두 프로세서 모듈은 동기화 회로를 이용하여 동작시간을 일치시키고, 자체진단 결과를 스위치 회로에 전송한다. 스위치 회로는 두 프로세서 모듈의 자체진단 결과에 따라 주 모듈을 선정하고, 두 프로세서 모듈은 재시작을 통해 정상 동작으로 복귀한다. 이러한 과정을 통해 일시적인 고장으로부터의 회복을 수행하고, 영구적인 고장인 경우에는 두 프로세서 모듈이 정지함으로써 고장정지 특성을 갖는다.

고장허용 기능을 갖는 이중구조의 마이크로콘트롤러의 설계에 있어서 클럭의 설계방법, 고장의 발견 방법, 프로세서 모듈의 고장 유무를 판단하기 위한 자체진단 방법, 프로세서 모듈의 동작을 동기화 하는 방법, 정상 프로세서 모듈만이 외부 출력 기능을 수행하도록 하는 방법이 우선 고려되어야 한다. 또한 이러한 기능들을 수행하기 위해 추가되는 회로 자체의 고장에도 대비할 수 있어야 한다. 따라서 이들 회로들 또한 이중으로 구성하여 하나의 회로에 고장이 발생한 경우에도 고장정지 특성을 유지할 수 있도록 하여야 한다.

## 2.1 클럭

클럭은 마이크로콘트롤러와 같은 디지털 시스템의 기본 요소로서, 클럭에 고장이 발생하게 되면 전체 시스템이 정상적인 동작을 수행할 수 없다. 따라서 클럭 자체에 고장허용 기능이 있어야 한다. 고장허용 클럭은 크게 두 가지 방법으로 구현할 수 있다. 첫째는 각 프로세서 모듈마다 별도의 클럭을 사용하는 것으로, 이 경우 클럭 자체의 구현은 쉽지만, 클럭 간의 주파수 차이로 인해, 모듈간의 상태 비교시 비교기에서 이를 동기화 시키는 방법이 필요하고, 이로 인해 비교기의 설계가 복잡해진다. 두 번째는 자체적인 고장허용 기능을 갖는 하나의 클럭으로 두 프로세서 모듈을 동작시키는 방법이다. 이 경우 클럭 자체는 복잡해지지만 두 프로세서 모듈의 초기 동작 시점만 일치시켜주면 그 후의 동작이 모두 동기화 되므로, 비교기 자체의 설계가 간단해진다. 설계와 구현의 용이성을 고려하여 본 논문에서 제시한 이중구조는 자체적인 고장허용 기능을 갖는 클럭을 사용하는 방법으로 클럭을 구현한다.




a) 자체진단형 클럭　　b) 전체 클럭의 구조

그림 2. 고장허용형 클럭

고장허용 기능을 갖는 클럭은 그림 2a와 같은 자체진단 기능을 갖는 클럭을 여러 개 사용함으로써 구현한다[7]. 자체진단 기능을 갖는 클럭은 오실레이터가 정상적으로 동작할 때에는 EX-OR 게이트의 출력이 "1"이 되지만, 오실레이터에 고장이 발생할 경우 EX-OR 게이트의 출력이 "0"이 되어 클럭의 고장을 발견할 수 있다. 이러한 자체진단형 클럭을 실제 구현시에는, 두 개의 단안정 멀티바이브레이터(PTMM, NTMM)의 출력 펄스간의 미세한 시간 오차로 인하여 순간적인 고장신호, 즉 EX-OR 게이트의 출력이 순간적으로 "0"이 되는 경우가 발생한다. 이와 같은 잘못된 고장신호를 제거하기 위해 저역 통과 필터와 쉬미트 트리거 버퍼를 사용한다. 또한 짧은 시간동안 발생하는 고장신호를 유지시켜 프로세서 모듈에 전달하기 위하여 최종 출력단에 F/F을 사용하고, 초기 F/F의 값을 정상 값으로 설정하기 위해 초기 리셋회로를 사용한다.

자체진단형 클럭을 단일로 사용할 경우 클럭 자체의 고장시 전체 시스템에 오동작이 발생하게 되므로, 이를 해결하기 위해 자체진단형 클럭을 이중으로 사용한다(그림 2b). 구체적으로 만약 하나의 클럭에서 고장이 발생하면, 고장신호가 클럭 스위칭 회로와 프로세서 모듈에 전달되고, 클럭 스위칭 회로는 고장이 발생한 클럭을 차단하고 다른 정상 클럭을 프로세서 모듈에 전달한다. 또한 프로세서 모듈은 고장신호를 받아들이게 되면 즉시 동작을 정지하고 재초기화 과정을 거친 후 정상 클럭을 이용하여 다시 동작을 수행한다.

## 2.2 고장의 발견

이중구조에서는 두 프로세서 모듈간의 동작 상태 비교를 통하여, 하나의 프로세서 모듈에서 고장이 발생한 것을 발견한다. 동작상태 비교는 이중구조 내부에서 발생하는 고장을 모두 발견하여, 외부로 틀린 데이터가 전달되는 것을 막을 수 있어야 한다. 따라서 이중구조에서는 내부에서 처리된 데이터가 최종적으로 외부로 출력되기 전에 항상 비교를 수행하기 위해, 두 프로세서 모듈의 외부 출력을 비교한다. 이때 외부로 실제 데이터가 출력되는 경우에만 비교를 하기 위해, 실제 프로세서 모듈이 외부로 데이터를 출력할 때 발생하는 "output enable" 신호를 이용하여 외부 데이터 출력이 발생할 때만 비교를 수행한다.

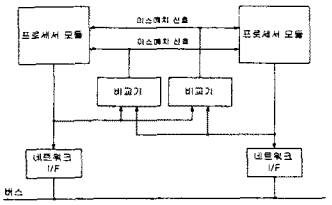



그림 3. 고장 발견을 위한 비교

동작상태 비교에 있어서 비교기를 단일로 구성할 경우, 비교기에서 고장이 발생하게 되면 고장정지 특성을 보장할 수 없다. 따라서 비교기 또한 그림 3에서와 같이 이중으로 구성함으로써, 하나의 비교기에 고장이 발생하더라도 다른 정상 비교기를 이용하여 고장정지 특성을 보장하도록 한다. 이때 비교기에서는 다음과 같은 고장이 발생할 수 있다. 1) 하나의 비교기의 출력 값이 항상 정상 상태로 고정되는 경우, 2) 하나의 비교기의 출력 값이 항상 미스매치 상태로 고정되는 경우, 3) 하나의 비교기의 출력 값이 계속하여 랜덤한 상태를 나타내는 경우, 4) 하나의 비교기에서 일시적으로 고장신호가 발생하는 경우이다.

이들 각각의 경우에 대해 이중구조의 동작을 살펴보면 다음과 같다. 1)번의 경우 미스매치 발생시 두 비교기의 동작 상태가 틀리게 되고, 이러한 현상이 미스매치가 발생할 때마다 계속 반복되게 된다. 이 경우 두 프로세서 모듈은 비교기의 영구적인 고장으로 판단하여 정지한다. 2)번의 경우 비교기에서 고장이 발생하는 순간 미스매치 신호가 발생하여 두 프로세서 모듈이 자체진단을 수행하게 된다. 자체진단 과정에서 두 비교기의 상태가 계속 틀리게 되고, 두 프로세서 모듈은 비교기의 영구적인 고장으로 판단, 동작을 정지한다. 3)번의 경우는 1)번의 경우와 동일하다. 마지막으로 4)번의 경우 두 프로세서 모듈은 자체진단을 수행한 후 재시작 과정을 거쳐 다시 정상동작으로 복귀한다. 이와 같은 과정을 통해 하나의 비교기에서 일시적인 고장이 발생한 경우는 고장으로부터의 회복을 수행하고, 영구적인 고장이 발생한 경우는 두 프로세서 모듈이 정지함으로써 고장정지 특성을 만족한다.

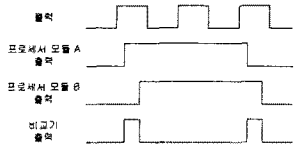



그림 4. 프로세서 모듈의 출력 타이밍

두 프로세서 모듈이 실제 정상적으로 동작하더라도 두 프로세서 모듈의 출력 값이 변화하는 시간에 차이가 발생할 수 있고, 이로 인해 비교기에서 미스매치 신호가 발생할 수 있다(그림 4). 이와 같은 문제를 해결하기 위해 그림 5와 같이 비교기의 출력에 F/F을 사용함으로써 순간적인 미스매치 신호를 무시한다.


그림 5. 비교기 출력의 순간적인 미스매치 제거

### 2.3 자체진단 및 상태판단

비교기에서 미스매치 신호가 발생하면 각 프로세서 모듈은 자신의 고장 발생 여부를 판단하기 위해 자체진단을 수행하게 된다. 구체적으로 자체진단에서는 초기에 설정된 내부 제어 레지스터의 값을 확인하고, 정확한 값으로 재설정을 한다. 또한 미리 저장된 테스트 패턴을 사용하여 각 메모리와 외부 하드웨어를 점검한다. 그리고 비교기, 동기화 회로, 스위치 회로의 출력 상태를 점검하여 이들 회로에 영구적인 고장이 있는지를 점검한다. 현재 자체점검은 보다 완벽한 고장발견을 위해 개선중이다.

자체진단 과정에서 고장이라고 판단한 모듈은 스스로 정지하여 다른 정상 프로세서 모듈이 단일모드로 동작할 수 있도록 한다. 그리고 자체진단 결과 정상이라고 판단한 모듈은 자체진단 결과를 스위치에 전달한다.

자체진단에 있어서 문제가 되는 것은 고장이 있음에도 이를 발견하지 못하는 경우이다. 특히 마이크로콘트롤러에 사용되는 프로세서와 같은 VLSI는 동작 환경에서 완벽한 테스트를 수행하는 것이 거의 불가능하고, 이로 인해 자체점검을 통해 모든 고장을 발견할 수 없다. 이와 같은 경우가 발생하면 두 프로세서 모듈은 동기화를 거친 후 다시 이중모드로 동작을 하게되고, 정상동작 중에 또 다시 미스매치가 발생하게 된다. 이러한 현상이 반복될 경우 추가적인 고장으로 인해 오동작이 발생할 수 있다. 따라서 자체점검 결과 두 프로세서 모듈이 모두 정상임에도 불구하고 짧은 기간 동안 계속하여 미스매치가 발생하는 경우 콘트롤러는 두 프로세서 모듈을 모두 정지시킨다.

### 2.4 재시작을 위한 동기화

두 프로세서 모듈이 자체진단 과정을 수행하는 동안 두 프로세서 모듈의 동작시간이 틀려질 수 있고, 이로 인해 정상동작 중에 비교시점이 일치하지 않게 된다. 따라서 두 프로세서 모듈이 다시 정상동작을 수행하기 이전에 동작시작 시점을 일치시켜주어야 하는데, 이를 위해서 동기화를 위한 외부 하드웨어를 사용한다.

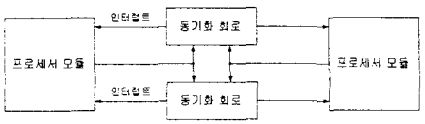



그림 6. 동기화 회로의 구성

전체 동기화 회로는 그림 6과 같다. 두 프로세서 모듈은 정지상태(Intel 계열 마이크로콘트롤러의 IDLE 모드[8], Motorola 계열 마이크로콘트롤러의 STOP 모드[9])로 들어가기 바로 전에 동기화 회로에 신호를 전달하고, 동기화 회로는 일정 시간 후에 두 프로세서 모듈에 동시에 인터럽트를 발생시킨다. 동기화 회로로부터 인터럽트가 발생하면 두 프로세서는 재시작 루틴을 수행하고 동작을 재개한다.

동기화 회로를 단일로 구성할 경우 동기화 회로에 고장이 발생하게 되면 정상 동작을 수행할 수 없다. 따라서 동기화 회로 또한 고장에 대비하여 이중으로 구성한다. 이때 동기화 회로에서도 2.2절의 비교기에서와 같이, 동기화 회로의 영구적인 고장으로 동기화 회로의 출력이 고정되거나, 랜덤하게 변하는 경우, 그리고 동기화 회로의 일시적인 고장으로 순간적인 인터럽트 신호가 발생하는 경우와 같은 고장이 발생할 수 있다. 각각의 고장에 대한 이중구조의 동작 또한 비교기에서 고장이 발생한 경우와 유사하다. 첫째 영구적인 고장인 경우 재시작 루틴과 자체진단 과정에서 두 동기화 회로의 동작이 다른 것을 발견하고 다시 자체진단을 수행한다. 이러한 동작을 계속하여 반복하면 동기화 회로의 영구적인 고장으로 판단하여 두 프로세서 모듈이 정지한다. 둘째 일시적인 고장인 경우는 재시작 루틴에서 두 동기화 회로의 동작이 다른 것을 발견하여, 다시 자체진단과 재시작 과정을 거쳐 정상동작으로 복귀한다.

### 2.5 동작 모드 결정

비교기에서 미스매치가 발생하면 스위치 회로는 즉시 외부 출력을 차단하여, 외부로 잘못된 결과가 전달되는 것을 방지한다. 그리고 스위치 회로는 각 프로세서 모듈이 자체진단 수행 후 스위치 회로에 전달한 상태판단 결과를 바탕으로 외부 네트워크 액세스를 수행할 모듈을 결정한다.

2.3에서 설명했듯이 자체진단 중에 고장이라고 판단한 모듈은 자체진단 과정에서 정지하게 되고, 스위치로 자체진단 결과를 전송하지 않게 된다. 또한 프로세서 모듈에 고장이 발생하여 프로그램을 정상적으로 수행할 수 없는 경우 또는 프로세서 모듈이 고장으로 인해 정지한 경우에도 스위치로 자체진단 결과를 전달하지 않게 된다. 이러한 경우에 프로세서 모듈의 고장을 판단하기 위해 스위치 회로는 두 프로세서 모듈이 정상이라고 판단하여 동작하는 중에 비교기에서 미스매치가 발생하면 즉시 두 프로세서 모듈의 네트워크 인터페이스를 차단하고, 내부 회로를 두 프로세서 모듈이 모두 고장인 상태로 설정한다. 그리고 미스매치 발생이후에 정상적으로 자체진단을 마치고 자체진단 결과를 스위치 회로에 전달한 프로세서 모듈만을 정상이라고 판단하며, 고장으로 인해 자체진단 결과를 정확히 전달하지 못한 프로세서 모듈은 고장이라고 판단한다. 스위치 회로에 전달된 자체진단 결과에 따른 구체적인 동작은 표 1과 같다.

표 1. 스위치 회로의 동작

| 전체 점검 결과 | 스위치 회로의 동작 상태 |
|---|---|
| 두 모듈 모두 정상 | 하드웨어적으로 미리 고정된 프로세서 모듈의 네트워크 I/F를 enable |
| 하나의 모듈만 정상 | 정상이라는 자체점검 결과를 보내온 프로세서 모듈의 네트워크 I/F를 enable |
| 두 모듈 모두 고장 | 두 프로세서모듈 모두 차단 |

자체진단 결과 두 프로세서 모듈이 모두 정상인 경우에는 두 모듈 중 어느 모듈이 외부 네트워크 인터페이스 기능을 수행해도 문제가 없기 때문에, 설계시 미리 하드웨어적으로 고정된 모듈이 네트워크 인터페이스 기능을 수행한다. 자체진단 결과 하나의 프로세서 모듈에 고장이 있는 경우에는 다른 정상 프로세서 모듈의 네트워크 인터페이스를 enable시킴으로써, 정상 모듈이 네트워크 인터페이스 기능을 수행하도록 한다. 마지막으로 두 프로세서 모듈이 모두 고장이라고 판단한 경우에는 두 프로세서 모듈의 네트워크 인터페이스를 모두 차단하여 틀린 데이터가 외부로 전달되는 것을 막는다.

표 2. 센서 모듈의 동작 모드 결정

| 자체진단 결과 | 전체 점검 결과 | 동작 모드 |
|---|---|---|
| 정상 | 두 모듈 모두 정상 | 이중 모드 |
| 정상 | 하나의 모듈만 정상 | 단일 모드 |
| 정상, 하지만 연속된 미스매치 | 두 모듈 모두 정상 | 정지 모드 |

스위치 회로에 자체진단 결과를 전송한 후 각 프로세서 모듈은 자신의 자체진단 결과와 스위치 회로에서 판단한 전체 점검 결과를 바탕으로 동작 모드를 결정하게 된다(표2). 자신이 정상이라고 판단한 모듈은, 스위치 회로가 두 모듈 모두 정상이라는 신호를 보내오는 경우 이중모드로, 하나의 모듈만 정상이라는 신호를 보내온

경우는 단일모드로 동작하게 된다. 이때 자신이 정상이라고 판단하고 전체 점검 결과 또한 두 모듈 모두 정상이지만, 연속된 미스매치가 발생한 경우에는 자체점검에서 발견되지 않는 고장이 있는 경우이다. 이러한 경우에는 두 모듈이 모두 정지하는 정지모드로 들어가게 된다. 자체진단 결과가 고장인 경우에는 자체진단 후 바로 동작을 정지하기 때문에 표에서 제외하였다.

하나의 프로세서 모듈에서 고장이 발생한 경우, 정상 프로세서 모듈이 외부 출력을 할 때마다 미스매치가 계속 발생하게 된다. 따라서 단일 모드로 동작할 경우 프로세서 모듈은 미스매치를 무시하고 동작을 계속 수행한다. 스위치 회로 또한 하나의 모듈만 정상이라고 판단한 경우에는 미스매치를 무시하여 정상 프로세서 모듈이 네트워크 인터페이스 기능을 계속 수행할 수 있도록 한다. 이와 같은 단일모드의 경우 정상 프로세서 모듈이 계속하여 동작을 수행하지만, 비교기를 통한 고장발견을 할 수 없게 된다. 따라서 단일모드에서 추가적인 고장이 발생할 경우 마이크로콘트롤러가 오동작을 할 가능성이 있다.

스위치 회로를 단일로 구성할 경우 스위치 회로에 고장이 발생하면, 네트워크 인터페이스를 차단하지 못하여 틀린 데이터가 외부로 전달될 수 있고, 따라서 고장정지 특성을 보장할 수 없다. 때문에 하나의 스위치 회로의 고장에 대비하여 스위치 회로 또한 이중으로 구성한다. 이때 스위치 회로에서도 앞서 2.2절의 비교기, 2.4절의 동기화 회로에서와 같이 영구적인 고장으로 스위치 회로의 동작상태가 고정되거나, 랜덤한 상태를 나타내는 경우, 또는 일시적인 고장으로 동작상태가 바뀌는 경우와 같은 고장이 발생할 수 있다. 각각의 경우에 있어서 이중구조의 동작 또한 앞서의 경우와 유사하다. 첫째 영구적인 고장인 경우에는 자체진단 과정에서 스위치 회로의 동작상태가 다른 것을 발견하여, 다시 자체진단을 수행한다. 이와 같은 과정을 반복하게 되면 스위치 회로의 영구적인 고장으로 판단하여 두 프로세서 모듈이 정지한다. 둘째 일시적인 고장인 경우에는 자체진단 후에 다시 올바른 상태가 다시 쓰여지게 되고, 이후 재시작 과정을 거쳐 정상동작으로 복귀한다.

## 3. 구현 및 검증

본 논문에서 제시한 이중구조의 마이크로콘트롤러는 모토롤라사의 MC68360 프로세서와 범용 로직 IC를 이용하여 구현하였다. 구현된 마이크로콘트롤러는 MC68360과 프로그램/데이터 메모리로 구성되는 프로세서 모듈, 동기화 회로, 스위치 회로, 비교기, 네트워크 인터페이스 기능을 하는 라인 버퍼로 구성된다(그림 7).

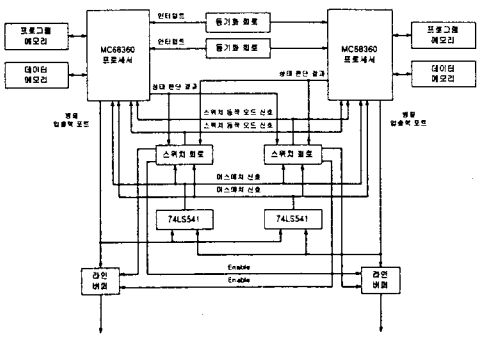



그림 7. 이중구조의 구현

구현된 마이크로콘트롤러의 기능 평가를 위해 동작 중에 여러 테스트 포인트에 각각 일시적인 고장과 영구적인 고장을 주입하였다. 그리고 일시적인 고장을 주입할 경우 콘트롤러가 자체진단과 재시작 과정을 통해 고장회복 기능을 수행하는 것을 확인하였고, 영구적인 고장을 주입할 경우에는 콘트롤러가 두 프로세서 모듈을 정지시킴으로써 고장정지 특성을 제공함을 확인하였다.

## 4. 결 론

본 논문에서는 고장정지 특성을 갖는 이중구조의 고장허용형 마이크로콘트롤러의 구조를 제시하였다. 제시한 마이크로콘트롤러는 두 개의 프로세서 모듈간의 동작 비교를 함으로써, 하나의 프로세서 모듈에서 고장이 발생하는 것을 발견한다. 고장이 발견되면 마이크로콘트롤러는 즉시 외부 출력을 차단함으로써 잘못된 결과가 외부로 전달되는 것을 방지하고, 두 프로세서 모듈은 자체진단을 통해 상태를 판단한다. 두 프로세서 모듈의 상태판단 결과를 바탕으로 마이크로콘트롤러는 외부 인터페이스 기능을 수행할 주 모듈을 선정한 후, 동기화와 재시작 과정을 거쳐 정상동작으로 복귀함으로써 일시적인 고장으로부터의 회복을 수행한다. 이러한 고장 발견과 회복의 과정이 짧은 시간 동안 반복적으로 일어날 경우, 마이크로콘트롤러는 자체적으로 진단이 되지 않는 영구적인 고장이 발생한 것으로 간주하고 두 프로세서 모듈을 모두 정지시킴으로써 고장정지 특성을 제공한다.

제시한 마이크로콘트롤러는 프로세서 모듈, 고장허용 클럭, 비교기, 동기화 회로, 스위치 회로로 구성된다. 구체적으로 고장허용 클럭은 자체진단 기능을 갖는 클럭을 이중으로 사용하여 구현하였다. 또한 비교기, 동기화 회로, 스위치 회로들도 각각의 고장에 대비하여 모두 이중으로 구성하여 어느 하나의 회로 블럭에 고장이 발생하더라도 콘트롤러가 고장정지 특성 및 일시적인 고장으로부터의 회복 기능을 제공하도록 하였다.

제시한 마이크로콘트롤러는 모토롤라의 MC68360 프로세서와 범용 로직 IC를 이용하여 구현하였다. 구현된 마이크로콘트롤러의 기능 평가를 위해 동작 중에 인공적으로 고장 현상을 주입하면서 콘트롤러의 고장허용 특성을 확인하였다.

### [참 고 문 헌]


[1] J.-C. Laprie, *Dependability: Basic Concepts and Terminology*, Springer-Verlag, 1991.

[2] T. A. Anderson and P. A. Lee, *Fault-Tolerance : Principle and Practice*, Springer-Verlag, 1990.

[3] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley Publishing Company, Inc., 1989.

[4] A. Avižienis, G. C. Gilley, F. P. Mathur, D. A. Rennels, J. A. Rohr, and D. K.Rubin, "The STAR(Self-Testing And Reparing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," *IEEE Transactions on Computer*, Vol. C-20, No. 11, pp. 1312-1321, Nov. 1971.

[5] W. N. Toy, "Fault-Tolerant Design of Local ESS Processors," *Proceedings of the IEEE*, Vol. 66, No. 10, pp. 1126-1145, Oct. 1978.

[6] J. A. Katzman, "System Architecture for Nonstop Computing," *Proceedings of the 14th Computer Society International Conference (Compcon)*, San Francisco, pp.77-80, Feb. 1977.

[7] A. M. Usas, "A Totally Self-Checking Checker Design for the Detection of Errors in Periodic Signals," *IEEE Transaction on Computers*, Vol. c-24, No. 5, pp. 483-488, May 1975.

[8] *8XC196Kx, 8XC196Jx, 87C196CA Microcontroller Family User's Manual*, Intel Corporation, June 1995.

[9] *MC68360 Quad Integrated Communications Controller User's Manual*, Motorola, 1993.