VMEbus를 통한 이중화 네트워크 프로토콜 구현

박정원* · 박성진

아주대학교 전자공학과

Implementation of a redundant network protocol based on VMEbus

Jeong-Weon Park* · Seong-Jin Park

Ajou University

E-mail : *ephome@ajou.ac.kr

요 약

본 논문에서는 VMEbus를 적용한 시스템에서의 안정성과 생존성을 증대시킬 수 있는 이중화 네트워크 프로토콜을 구현한다. 일반적으로 시스템의 생존성을 증대시키기 위한 방법으로써 적용하는 이중화 기법은 두 개의 프로세스 상호 간에 두 개의 네트워크망을 구성하여 이중화를 구현하는 것이다. 그러나 프로세스의 고장이나 물리적 네트워크망이 손실되었을 경우 기능을 제대로 수행하지 못할 수 있다. 이러한 문제점을 보완하고 안정성을 확보하기 위한 방법으로 프로세스 간에 VMEbus 통신을 통한 이중화 네트워크망 프로토콜을 제시한다. 본 논문에서는 이 프로토콜을 직접 구현하고 실험을 통하여 이 방안의 타당성을 확인하고자 한다.

ABSTRACT

In this paper, we apply the system in VMEbus to increase the stability and viability to implement a redundant network protocol. Typically, methods for increasing the viability of the system applied by the redundancy scheme between the two processes by configuring the network in between the two is to implement redundancy. However, the physical network in the process of the failure or when the loss function may not perform properly. These issues are complementary processes as a way to ensure stability between the VMEbus communication protocol is presented through a redundant network. In this paper, a direct implementation of the protocol and experimental results confirm the validity of these measures is to.

키워드

이중화, Network Protocol, VMEbus

Ⅰ. 서 론

국내 기술의 발달과 동시에 최신의 군용 장비들이 국내 연구기관과 기업들에게서 개발되고 있으며, 그 성능을 구현하고 입증하는 단계에서 많은 성과를 보이고 있다. 더불어 사용자인 군의 요구에 의해서 장비 성능에 대한 안정성과 긴박한 시간에 그 성능을 유지할 수 있는 생존성을 증대시키기 위한 방법이 대두되고 있으며, 그 방법 중의 하나로 시스템에서의 이중화 설계에 대한 이슈가 늘어나고 있는 추세이다. 또한 많은 군용 시스템에서 그 방법을 적용하기 위하여 노력하고 있다.

이에 VMEbus 환경에서 구현한 이중화 하드웨어를 사용하여 이중화 네트워크 프로토콜의 성능을 확인한다. VME는 Versa Module Eurocard의 약자로 보드규격인 Versa 보드를 유럽 규격에 맞춘 것이다[1]. 군용 장비들이 VMEbus 방식의 시스템을 사용하는 이유는 그 성능에 대한 확장성이 편리하고, 개발 시 하드웨어에 구애받지 않고 시스템을 설계할 수 있는 편의성 때문이다. 또한 대형 시스템의 설계 시 독자적인 형태의 통신체계나 OS(Operating System), 그리고 Bus 체계 보다는 공업표준에 의거한 제품이 사용자에게 신뢰

감을 줄 수 있고, 만일의 경우 이것을 생산한 회사가 없어지더라도 성능 확장 및 유지보수가 가능한 것도 하나의 이유가 된다.

이에 두 개의 일반적인 LAN 망에 의존하는 네트워크망에서 VMEbus를 통한 두 개의 프로세서가 상호 데이터를 공유하면서 실시간 네트워크망을 이룰 수 있도록 설계하면, 장비의 안정성과 생존성을 증대시킬 수 있다.

II. 기존 대비 개선점

국내에서 기 개발하여 운용중인 군용 장비에 적용한 기본적인 이중화의 기법은 두 장비 간 각각의 장비 내부에서 주 업무를 수행하는 프로세스인 SBC(Single Board Computer)의 네트워크망을 이중화하여 적용하고 있다[4]. 이 방법은 하나의 네트워크망이 손실되었을 경우 다른 네트워크망을 통하여 통신을 지속적으로 수행하는 것이다. 그러나 SBC 자체가 비정상 동작을 하거나 기능을 상실하게 되면 더 이상의 업무를 수행할 수가 없게 된다. 이러한 경우 군용 장비에서는 치명적인 운용적 손실을 가져올 수 있다.

이에 좀 더 생존성을 증대시키기 위한 방법으로 본 논문에서 제시하는 이중화 설계는 기존의 이중화 설계 방법에서 두 SBC 간의 VMEbus를 통한 이중화 네트워크 프로토콜을 제시한다. 이 방법은 두 개의 SBC에서 각각 두 개의 네트워크망을 사용하며, 서로 간에 VMEbus로 통신을 위하여 Master 역할의 SBC1과 Slave 역할의 SBC2로써 공유 메모리 영역을 설정하여 사용하고, 인터럽트 방식을 적용하며, 이중화를 담당하는 전용 Task와 통신 이상 시 이를 처리할 이벤트를 발생시키는 방법을 적용한다. 이 방법을 통하여 실시간으로 데이터를 공유하고 있는 두 개의 SBC 중 하나의 SBC가 손실되었을 경우 또는 네트워크망이 손실 될 경우 좀 더 지속적으로 통신을 수행해 나아갈 수 있다.

III. 프로토콜 설계

VMEbus를 통한 이중화 네트워크 프로토콜을 수행하기 위한 구성은 그림 1과 같다. 두 개의 SBC는 Master 역할과 Slave 역할로써 상호 간에 VMEbus 통신을 통하여 데이터를 주고받는다. 그리고 각각의 SBC는 두 개의 Ethernet 네트워크망이 두 개의 HUB로 연결되며, 각각의 채널 1은 HUB1, 그리고 채널 2는 HUB2로 연결된다. 외부 장비와의 네트워크망은 각각의 HUB에서 하나의 채널이 나와 최종적으로 두 개의 망으로 연결된다. 이로써 내부에서는 두 개의 SBC를 사용하여 네트워크망을 구현하였지만, VMEbus를 통하여 상호 데이터를 주고받고 있기 때문에 외부에서 보기에는 하나의 프로세서와 통신을 하는 것과 동일하게 인식하며, 전체 4개의 채널을 통하여 통신을 수행하여 생존성을 증가시킬 수 있다.

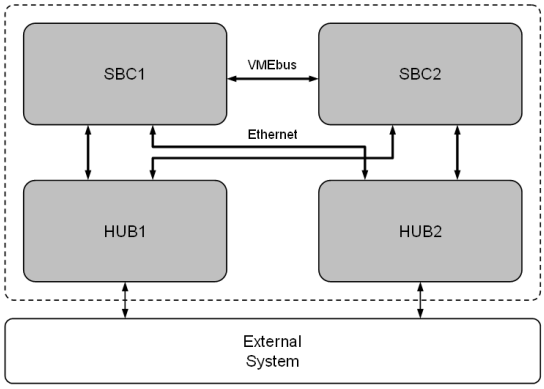

그림 1. 이중화 하드웨어 구성 블록도

Master 역할의 SBC1과 Slave 역할의 SBC2는 각각 우선 한 채널씩 외부와 실시간 통신을 수행하며, VMEbus를 통하여 상호 Ethernet 네트워크망과 SBC 자체 상태를 주고받게 된다.

소프트웨어는 RTOS(Real Time Operating System)인 VxWorks를 통하여 구현하였다[2][3].

이중화 네트워크 프로토콜은 VMEbus Master 역할의 SBC1과 Slave 역할의 SBC2 간의 공유 메모리 영역과 인터럽트 방식을 이용하여 구현한다. 그러기 위하여 이중화를 담당하는 전용 Task인 tManagerTask가 존재한다. tManagerTask의 기능은 그림 2와 같다.

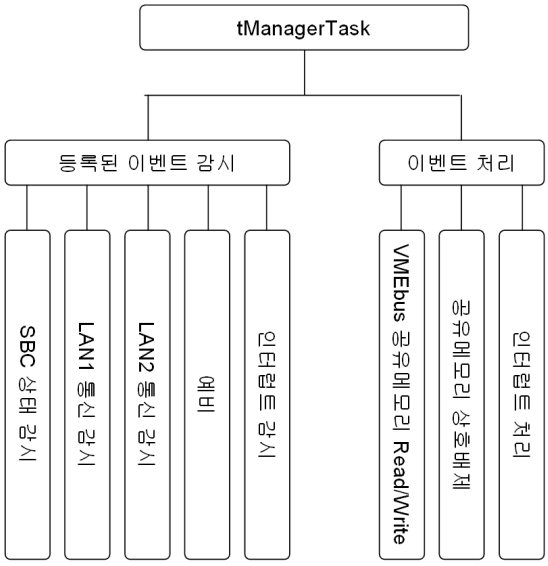

그림 2. tManagerTask의 기능

tManagerTask는 '등록된 이벤트 감시'와 '이벤트 처리'를 수행하는 두 가지 기능으로 나눌 수 있다. '등록된 이벤트 감시' 기능은 먼저 'SBC 상

태 감시' 기능을 통하여 주기적으로 SBC의 상태를 감시한다. 이 기능을 통하여 SBC의 부분적인 통신 기능의 손상뿐만 아니라 완전 손상에 대비하여 다른 SBC가 모든 기능을 상속받을 수 있게 한다. 두 번째로 'LAN1 통신 감시' 기능을 통하여 Ethernet LAN1 통신 기능에 이상이 발생한 경우에 이벤트를 발생시킨다. 세 번째로 'LAN1 통신 감시' 기능을 통하여 Ethernet LAN2 통신 기능에 이상이 발생한 경우에 이벤트를 발생시킨다. 그 밖의 통신 기능에 대비하여 '예비' 기능으로 포함하였고, 마지막으로 '인터럽트 감시' 기능을 통하여 SBC의 상태, Ethernet LAN1 통신 상태, 그리고 Ethernet LAN2 통신 상태를 알리기 위하여 VMEbus 공유 메모리에 Write하고 발생한 인터럽트를 이벤트로 등록하여 처리하게 된다.

tManagerTask의 두 번째 기능인 '이벤트 처리'에 대한 첫 번째 'VMEbus 공유 메모리 Read/Write' 기능은 두 개의 SBC 간에 정보를 공유하게 되고, 두 번째 '공유메모리 상호배제' 기능은 VMEbus의 Read/Write 수행 시 정보가 손상되는 것을 막아준다. 마지막으로 '인터럽트 처리' 기능은 VMEbus의 공유 메모리 영역에 Write한 후 인터럽트를 발생시켜 다른 SBC에 알려주는 기능을 갖는다.

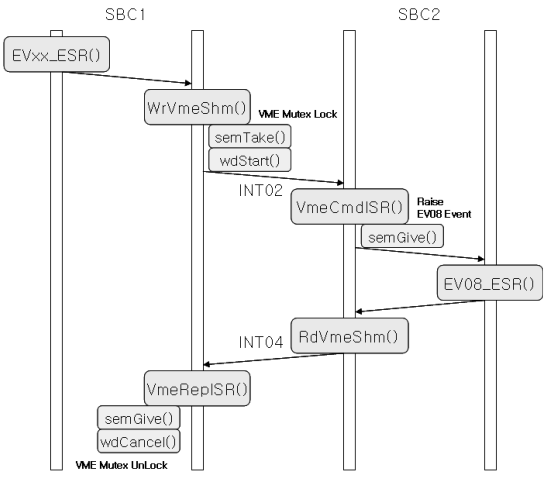

그림 3. 통신 이상 시 이중화 시퀀스 다이어그램

외부 장비와의 통신을 담당하는 Task에서 통신상의 이상이 발생하면 tManagerTask에게 이벤트를 발생시켜 이상이 발생하였음을 알려준다. 그리고 이벤트 종류에 따라 VMEbus의 Master와 Slave 간의 공유 메모리 영역에 데이터를 Write한다. 그리고 Master 역할의 SBC1은 인터럽트를 발생시켜 통신 이상이 발생하였음을 Slave 역할의 SBC2에게 알려준다. Slave 역할의 SBC는 인터럽트가 수신된 것을 확인한 후 VME Master와 Slave 간의 공유 메모리 영역으로부터 데이터를 Read한다. 그리고 Read한 데이터의 내용에 따라 해당 통신 채널을 오픈한다.

VMEbus 이중화 네트워크 프로토콜의 정상동작 시퀀스 다이어그램은 그림 3과 같다.

프로토콜을 위해 구현한 함수로는 이벤트 처리 서비스 루틴 함수와 VMEbus Read/Write 함수, 그리고 인터럽트 처리 함수로 나눌 수 있다. 이벤트 처리 서비스 루틴 함수는 EVxx_ESR() 함수로서 각 이벤트를 처리하는 함수이다. VMEbus Read/Write 함수로는 공유 메모리에 Write를 수행하는 기능의 WrVmeShm() 함수와, Read를 수행하는 기능의 RdVmeShm() 함수가 있다. 마지막으로 인터럽트 처리 함수로는 VmeCmdISR() 함수와 VmeRepISR() 함수가 있다.

기존에 VxWorks 라이브러리로 제공되는 함수로는 WatchDog 제어 함수와 세마포어 제어 함수를 사용하였다. WatchDog 제어 함수에는 WatchDog 기능을 시작하는 wdStart() 함수와 중지하는 기능의 wdCancel() 함수를 사용하였고, 공유 메모리 상호배제 기능을 위하여 세마포어 제어 함수를 사용하였으며, semTake() 함수와 semGive() 함수가 있다.

Ⅳ. 구 현

Ethernet 통신 프로토콜을 운용하기 위하여 본 시스템은 이중화를 담당하는 전용 Task인 tManagerTask 외에 여러 Task를 필요로 하며 각각의 기능은 표 1과 같다. tLan1UdpRcvTask와 tLan2UdpRcvTask에서 UDP 통신을 통하여 외부 장비로부터 메시지를 수신하면 tMsgFilterTask에서 중복 메시지 여부를 판별하여 중복 메시지인 경우 메시지를 버린다.

표 1. 이중화 Task 기능

Task명	기능
tLan1UdpRcvTask	LAN1 UDP 메시지 수신
tLan2UdpRcvTask	LAN2 UDP 메시지 수신
tMsgFilterTask	중복 메시지 분류
tDataProcTask	수신 메시지 처리
tUdpSendTask	UDP 메시지 송신

중복 메시지가 아닌 경우에는 tDataProcTask에서 수신 메시지를 처리한 후 메시지 처리 결과를 tUdpSendTask를 통하여 외부 장비에 송신한다.

외부 장비와 정상적으로 통신을 수행하던 중 통신이 두절되게 되면, 연결 상태를 확인하는 Communication Status 명령을 수신하지 못한다. 프로토콜에서 정의한 시간 내에 Communication Status 명령을 수신하지 못하면 WatchDog 타이머가 동작하게 된다.

WatchDog 타이머가 실행되면 서비스 루틴을 실행시키며, 서비스 루틴에서는 tManagerTask에

이벤트를 발생시킨다. 그리고 tManagerTask가 이벤트를 수신하면 해당 이벤트의 종류에 따라 처리한다.

```
/* ESR = Event Service Routine */
static void EV01_ESR(  int nVmeId, int nVmeWrAckIrq ) {
        char wrbuff[255] = { 0 };

        ASSERT( (nVmeId >= 1) && (nVmeId <= 5) );
        ASSERT( (nVmeWrAckIrq >= 1) && (nVmeWrAckIrq <= 4) );

        /* Delete Current LAN1 UDP Receive Task */                    ①
        _UDPRCV_TASK_delete_task( LAN1 );

        wrbuff[0] = 01;                                               ②
        wrbuff[1] = nVmeWrAckIrq;

        /* Write a message to the Vme Shared Memory */
        write_vme_shm( pcVmeSlaveAddr, wrbuff );

        if(nVmeId == 1) {                                             ③
                ifAddrSet("mgi0", "192.168.10.111");
        } else {
                ifAddrSet("mgi0", "192.168.10.222");
        }

        sysBusIntGen( nVmeWrAckIrq, VME_CMD_VECTOR );                 ④
}
```

그림 4. 이벤트01 처리 서비스 루틴

만약 LAN1에 이상이 발생하여 이벤트가 발생하였을 경우, 그림 506와 같이 이벤트01 처리 절차를 수행한다.

①과 같이 현재 동작중인 tLan1UdpRcvTask를 먼저 종료한다. Task를 종료한 후에 ②와 같이 공유 메모리 영역에 이상 정보를 Write 한다. ③의 과정을 통하여 SBC2와의 IP 충돌을 방지하게 된다. 모든 과정이 끝나면 ④의 과정을 통하여 인터럽트를 발생시켜 SBC2에 알린다.

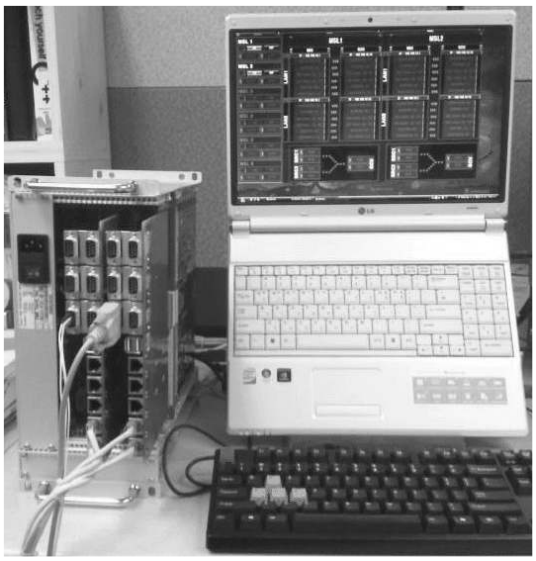

그림 5. 하드웨어 구성

VMEbus를 통한 이중화 네트워크 프로토콜을 구현하여 확인한 화면과 하드웨어 구성은 그림 5와 같다.

이로써 두 SBC 각각의 비정상 동작을 모의하고, Ethernet LAN1과 Ethernet LAN2의 통신 기능에 이상이 발생한 경우를 모의하여 다른 여유 채널을 통한 외부 장비와의 실시간 통신이 유지되는 것을 확인함으로써 프로토콜에 대한 안정성과 생존에 대한 확장성을 확인하였다.

Ⅴ. 결 론

본 논문을 통해서 향상된 이중화 설계 기법을 제시했으며 기존의 설계 방식보다 시스템의 안정성과 생존성을 높일 수 있음을 확인했다. 향후 개발하는 장비에 적용하여 좀 더 좋은 성능의 군용 장비를 생산하였으면 한다.

참고문헌

[1] The VMEbus64 Handbook 4th Edition, Wade D. Peterson
[2] VxWorks reference manual 5.3.1
[3] RTOS를 이용한 실시간 임베디드 시스템 디자인, Qing Li, Caroline Yao
[4] VR9 Hardware User's Manual Edition 1.6, SBS Technologies