2003년도 대한전기학회 하계학술대회 논문집 2003.7.21-23

네트워크를 이용한 분산형 제어시스템 개발

정태수*, 김준국*, 이종성**, 박기헌*
*성균관대학교, **부천대학

Development of networked distributed control system

Taesoo Jung*, Joonkook Kim*, Jongsung Lee**, Kiheon Park*
*Sung-Kyun-Kwan Univ., **Bu-cheon College

Abstract - In this paper, several network nodes that are essential elements of create a distributed system have designed and developed using CAN(Controller Area Network). The network nodes consist of RISC(Reduced Instruction Set Computer) CPU and CAN controller. By these two parts used, the network nodes have merits that are fast process speed, stable communication, cheapness and others. The most important quality that is the designed network node can be applied to various network control systems.

1. 서 론

분산형 제어시스템을 효과적으로 제어하기 위해서는 각각의 장치를 상호 연결하여 제어에 필요한 데이터를 실시간 전송할 수 있는 통신수단이 필수적으로 요구된다. 최근, 이와 같은 통신수단으로 네트워크를 적용하는 연구가 활발히 진행되고 있으며, 네트워크를 이용한 분산형 제어시스템에서는 센서, 구동기, 그리고 제어기 등의 역할을 수행하는 네트워크 노드를 개발하는 것이 필수적이다.

차량용 네트워크로 개발된 CAN(Controller Area Network)은 국제 표준 프로토콜로서 시장성이 우수하고 데이터 전송상의 안정성과 신뢰성이 우수하다는 등의 여러 가지 장점 때문에 최근에는 다양한 분야에 걸쳐 응용되고 있다. ATMEL사에서 개발한 AVR 마이크로 컨트롤러는 전력 소모가 적고 별도로 외부장치들을 연결하지 않더라도 기본적으로 필요한 기능들을 사용할 수 있으며, 가장 큰 장점은 RISC(Reduced Instruction Set Computer)방식으로, 처리 속도가 매우 빠르다는 점이다.[4]

본 논문에서는 CAN 프로토콜을 연구하고, 이를 일반적인 제어 대상에 적용할 수 있도록 AVR 시리즈인 ATmega128을 탑재한 RISC 기반의 네트워크 노드를 개발하여 독립된 네트워크 시스템을 설계하였으며 데이터 송수신 실험을 통하여 CAN 프로토콜의 안정성과 AVR-CAN 시스템의 빠른 처리속도를 증명하여 실시간 제어가 요구되는 시스템에 적용 가능함을 보였다.

2. 본 론

2.1 CAN의 특성

CAN은 자동차 내부의 전장품들을 상호 연결하기 위한 Network용으로 개발되었으며, 물리적으로 차량 내에 산재해 있는 다양한 전자장치들을 서로 연결해 주는 역할을 한다. CAN은 이러한 차량용 네트워크의 특성에 맞도록 견고성과 강인성, 그리고 배선의 간결함 등의 장점을 가지고 있고, ISO 11898(high-speed application)과 ISO 11519(lower-speed application) 국제 표준으로 공인 받은 프로토콜로서 데이터 전송 시 강력한 오류 처리 기능을 내장하고 있어 제조산업, 항공우주분야, 자동제어 분야 등 다양한 영역에서 그 활용범위를 넓혀 가고 있는 추세이다[1],[2],[3],[5].

2.2 CAN의 구조

CAN은 OSI(Open System Interconnection) model의 하위 두 계층 Physical Layer와 Data Link Layer로 구성된다. Physical Layer에서 CAN은 보편적으로 Twist-pair Cable을 이용하여 데이터를 전송한다. 이 때, 전기적으로 두 선간의 차 전압(balanced line signalling, [2])을 이용하기 때문에 데이터 이동 경로에서 노이즈의 영향에 강인한 특성을 가질 수 있다. Data Link Layer에서는 상위 계층인 Application Layer의 데이터를 전송하기 위한 기능적, 절차적 수단을 제공한다. 여기에는 Physical Layer에 전달할 메시지를 포맷하고 매체 접근방법(CSMA/BA, [2])을 제공하며 전송상의 에러를 검출하고 수정하는 역할이 포함되어있다[1],[2].

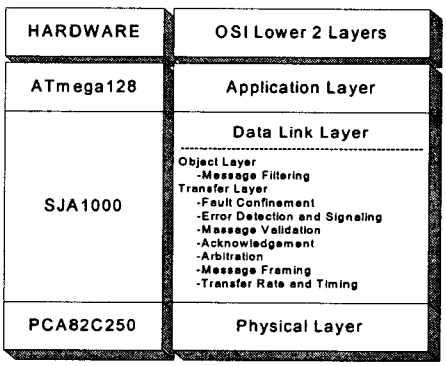

그림 1. CAN Layer Structure

2.3 CAN 프로토콜 분석

2.3.1 메시지

가. 데이터 프레임(Data Frame): 데이터를 전송하는 프레임으로 그림 2와 같이 7개의 필드형식으로 구성되어 있으며, 최대 8바이트의 데이터를 전송할 수 있다.

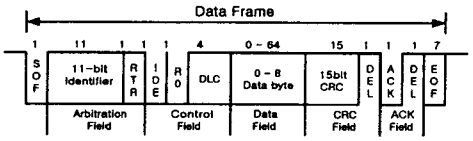

그림 2. Data Frame

나. 전송요청 프레임(Remote Frame): 한 노드에서 다른 노드로 데이터를 요구할 때 전송하는 프레임으로 데이터 필드가 존재하지 않으며, RTR (Remote Transmission Request)비트가 recessive ('r')값을 가지고, 데이터프레임과 동일한 확인자를 갖고 있다.

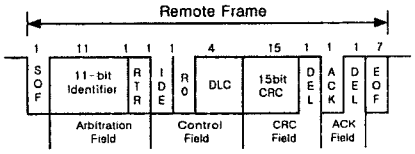

그림 3. Remote Frame

다. 에러 프레임(Error Frame): 한 노드에서 수신한 메시지에서 에러를 발견할 때 전송하는 프레임이다.

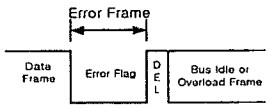

그림 4. Error Frame

라. 오버로드 프레임(Overload Frame): 현재 수행중인 프레임의 다음에 오는 데이터 프레임과 전송요청 프레임을 지연시키기 위해 사용된다.

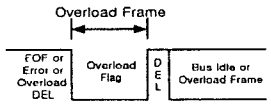

그림 5. Overload Frame

마. 인터프레임 스페이스(Interframe Space): 위의 4가지 프레임들을 서로 구획하는 방법으로 인터프레임 스페이스가 있는데 데이터 프레임과 전송요청 프레임은 3비트의 인터프레임 스페이스에 의해 프레임과 분리된다[1],[3].

2.3.2 메시지 필터링과 우선순위

CAN의 확인자(identifier)는 크게 두 가지의 역할을 수행한다. 첫째는, 자기 자신의 메시지를 확인할 수 있는 메시지 필터링 방법을 제공하는 것이며, 둘째는 이러한 확인자의 우선순위를 통해 버스에 접근하는 우선순위를 결정하는 방법이다. 전자는 CAN 버스를 통해 방송(Broadcasting)되는 메시지에 자기 자신의 확인자와 일치하는 데이터 프레임이나 전송요청 프레임이 존재할 경우, 메시지 필터링에 의해서 자신의 확인자에 해당하는 메시지를 수신하는 방법을 말한다. 후자는 데이터를 전송하고자 하는 노드가 공유버스(CAN 버스)에 접근하기 위한 방법으로, 확인자의 크기가 작은 노드에 높은 버스점유 우선권을 부여하고 반대인 경우 상대적으로 낮은 우선권을 부여하여 두 메시지 간의 충돌이 발생할 경우 우선권이 높은 메시지가 버스를 점유하도록 하는 CSMA/BA(Carrier Sense Multiple Access with Bitwise Arbitrary) 방식을 뜻한다.

2.3.3 에러 검출 및 처리

가. 비트 에러(Bit Error): 각각의 전송단은 전송된 신호의 레벨을 모니터링하고 있다가 전송된 신호와 모니터링 된 신호의 레벨이 다를 때 발생한다. 단, 중재기간 동안에는 Bit Error는 발생하지 않는다.

나. 스터프 에러(Stuff Error): CAN에서는 같은 레벨의 연속된 5비트의 반대되는 레벨을 추가하며 6개의 연속되는 같은 레벨을 검출했을 때 Stuff Error를 발생한다.

다. CRC 에러(CRC Error): 수신된 CRC Sequence의 계산결과가 일치하지 않을 때 CRC에러를 발생한다.

라. 폼 에러(Form Error): CAN 메시지 중 일부분은 고정형태비트 필드가 잘못된 비트를 담고 있을 때 발생한다.

마. ACK 에러(ACK Error): 메시지를 보낸 노드가 ACK 신호를 받지 못했을 경우 발생한다.

네트워크 노드에서 검출된 모든 에러들은 즉시 나머지 네트워크 노드들로 통보가 되며 에러메시지를 수신한 후에 모든 노드들은 수신된 비트들을 무시하게 되고 버스가 다시 Idle 상태가 될 때까지 지연된 후에 메시지를 재 전송하게 된다.

2.3.4 에러 제한

잘못된 연결, 케이블의 불량, 지속되는 외부 외란에 의해 발생되는 영구적인 오류는 특별한 알고리즘에 의해 제한된다. 하나의 CAN 버스에 연결된 각각의 노드는 에러 능동(Error Active), 에러 수동(Error Passive) 그리고 버스 오프(Bus off)의 세 가지 노드상태를 가진다.

가. 에러 능동(Error Active): 에러능동 노드들은 버스 통신에 정상적으로 참여할 수 있고 에러가 검출될 때 능동에러 플래그를 보낸다.

나. 에러 수동(Error Passive): 에러수동 노드들은 보통 버스 통신에 참여할 수 있고, 에러가 검출될 때 수동에러 플래그를 보낸다.

다. 버스 오프(Bus off): 에러카운터 값이 일정값을 초과할 경우 CAN 프로토콜은 해당 노드의 결함으로 판단하고 버스로부터 차단한다. 버스차단 상태의 노드는 물리적으로 CAN 버스와 완전히 분리된 상태와 같으며, 어떠한 메시지 프레임도 버스를 통해 전송할 수 없다.

그림 6은 CAN 노드의 에러상태 천이에 대한 블록도이다. CAN 컨트롤러는 자체적으로 수신에러 카운터(Receive Error Counter)와 전송에러 카운터(Transmit Error Counter)의 두 가지 카운터를 가지고 있다. 메시지 송신시 에러가 발생할 경우, 송신에러 카운터가 증가하고, 메시지 송신이 정상적으로 종료되면 이 카운터의 값은 감소한다. 마찬가지로, 메시지 수신시 에러가 발생할 경우 수신에러 카운터가 증가하고, 메시지 수신이 정상적으로 종료되면 이 카운터의 값은 감소하게 된다. 이러한 두 가지의 카운터 값에 따라 정상적인 CAN 노드들은 결함이 있는 노드에 의해 방해받지 않을 수 있다.

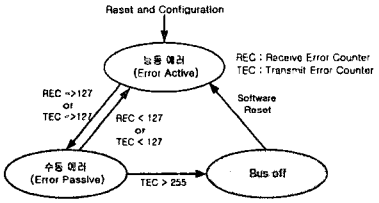

그림 6. 에러 상태 알고리즘

3. 실험장치 구성

3.1 ATmega128

ATmega128은 AVR 계열의 마이크로 컨트롤러로 Software 의존적인 RISC방식의 Instruction set을 사용, 기존의 상용 컨트롤러보다 빠른 속도의 데이터 처리속도를 갖는다. 또한 다수의 입출력 포트와 ISP 및 SPI, 그리고 128Kbyte in system flash는 사용하기 쉽고 저렴한 개발환경을 구축 가능케 한다.

그림 7은 ATmega128의 개발 환경을 구축하기 위해 본 논문에서 제작한 Flash memory writer이다. 여기서는 ATmega128의 내부 Flash memory에 프로그램을 탑재하는 작업과 입출력 포트의 검사, 인터럽트 및 AD변환을 수행할 수 있도록 제작하였다[4].

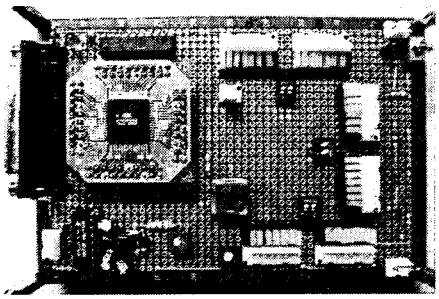

그림 7. Flash memory writer

3.2 네트워크 노드

본 논문에서 네트워크 시스템을 구성하기 위해 제작한 CAN 노드는 그림 8과 같다. 제작한 노드는 CAN 버스에 연결되어 있는 다른 노드와 데이터를 상호 전송할 수 있으며, 구성도는 그림 9와 같다.

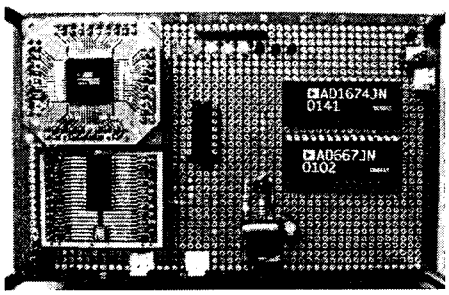

그림 8. 네트워크 노드

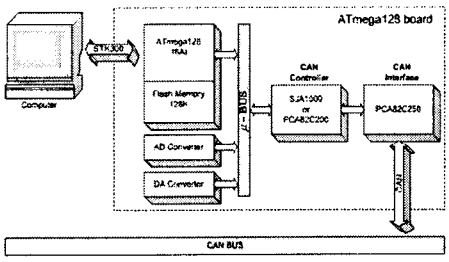

그림 9. 실험용 네트워크 노드 구성도

4. 실험 및 결과분석

3장에서 제작한 네트워크 노드를 이용하여 실제 CAN 네트워크 시스템을 구축하였다. 즉, 그림 8의 CAN 노드를 이용하여 Sensor 노드와 Actuator 노드를 구성한 후 두 노드간의 데이터 전송실험을 수행하였다. 먼저, Sensor 노드에는 입력된 아날로그 데이터를 AD 변환하여(Sampler), CAN 프로토콜로 변환한 후 버스를 통해 Actuator 노드로 전송하는 프로그램을 탑재했으며, Sensor 노드에서 전송하는 데이터를 수신하여 적절한 프로토콜 변환과정을 통해 DA 과정을 거쳐서 다시 아날로그 신호로 복원하여 출력할 수 있는 프로그램을 작성하였다. 그리고, 이러한 두 과정상의 전송지연시간을 측정하기 위해 Oscilloscope를 통하여 Sensor 노드의 아날로그 신호와 Actuator 노드에서 복원된 아날로그 신호를 상호 비교하였다. 그림 10은 이때의 실험결과를 나타낸다.

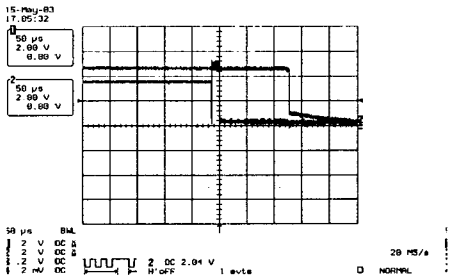

그림 10. 전송지연시간의 측정

그림 10에서 그래프의 중심선은 Sensor 노드가 데이터를 전송하기 위해서 전송명령을 내려주기 직전의 시점이고, 중심선 오른쪽에 보이는 수직선은 Actuator 노드가 CAN 버스를 통해 메시지를 받아서 이를 복원하여 출력하는 시점이며, 이 두 가지 신호사이에는 약 130 μs의 시간지연이 존재함을 확인할 수 있다. 이러한 시간지연은 데이터가 CAN 버스를 통해서 이동할 때 걸리는 전송지연시간, Sensor 노드의 메인 컨트롤러가 전송 지령을 내리는데 걸리는 지연시간, Actuator 노드가 데이터를 수신한 후 이를 다시 아날로그 신호로 복원하는 지연시간 등을 포함하는 것으로 분석할 수 있다.

5. 결 론

본 논문에서는 CAN 프로토콜을 연구하고, 이를 일반적인 제어 대상에 적용할 수 있도록 AVR 시리즈인 ATmega128을 탑재한 RISC 기반의 네트워크 노드를 개발하였다. 이후, 설계된 네트워크 노드를 이용하여 독립된 네트워크 시스템을 설계하였으며 데이터 송수신 실험을 통해 CAN 프로토콜의 데이터 통신 특성과 안정성을 검증하고, AVR-CAN 시스템의 빠른 처리속도를 증명하여 실시간 제어가 요구되는 시스템에 적용 가능함을 보였다. 향후, 실제 분산제어시스템을 구축하는데 있어 설계된 CAN 노드들을 이용한다면 저렴하고 성능이 뛰어난 제어 시스템의 구성이 가능할 수 있으리라 판단된다.

[참 고 문 헌]

[1] BOSCH, CAN Specification, Part A/B,1991

[2] SJA 1000 Stand-alone CAN Controller - Product Specification, Phillips Semiconductors, 4. 2000.

[3] 이원무 외, "CAN을 이용한 Smart Sensor/Actuator Node 개발", 대한 전기학회 하계학술대회 논문집 pp 2132~2134, 7. 2002.

[4] ATMEL, ATmega128(L) Data Sheet, Rev. 2467GS-AVR. 9. 2002.

[5] Micheal Barr, "Programming Embedded Systems in C and C++", O'reilly, 2000.