2001년도 대한전기학회 하계학술대회 논문집 2001.7.18-20

CAN(Controller Area Network)을 이용한 멀티플렉싱 기술개발

윤 상진*, 정 준홍*, 이 종성**, 조 용석***,민 덕기*, 박 기헌*
*성균관대학교, **부천대학, ***건양대학교

Development of a Multiplexing Method Using Controller Area Network(CAN Protocol)

Sangjin Yoon*, Joonhong Jung*, Jongsung Lee**,Yongseok Cho***, Deukgi Min*, Kiheon Park*
*Sung-Kyun-Kwan Univ, **Bucheon College, ***Konyang Univ

Abstract - 본 논문에서는 데이터 전송시 에러 검출빈도가 작아 신뢰성이 매우 높은 CAN 프로토콜을 이용하여 자동차 사이드 미러에 대한 멀티플렉싱 시스템을 구현하였다. 사이드 미러를 제어하기 위한 미러부 Slave Controller, 사용자의 상하좌우 및 기준위치 저장/로드입력을 처리하기 위한 입력부 Slave Controller, 그리고 이들 두 Slave Controller를 제어하는 Master CAN Controller로 시스템을 구성하고, 각각의 장치들을 중앙의 CAN 버스를 이용하여 데이터를 전송함으로써 이들을 제어할 수 있는 시스템을 구현하였다.

1. 서 론

최근 전자기술의 비약적인 발전으로 고성능의 마이크로프로세서가 소개되고 다양한 제어시스템 설계기법이 개발됨에 따라 필드에 설치된 여러 제어기기들을 중앙에서 제어할 수 있는 중앙집중형 제어시스템(Centralized Control System)이 주목받고 있다. 그러나 이러한 중앙집중형 제어시스템에서는 센서와 구동기와 같은 각각의 입·출력장치들과 중앙 제어장치 사이의 데이터 전송을 위해 RS232와 같은 직렬통신을 사용하였으며, 그 결과 중앙제어부와 필드에 분산되어 설치된 제어기기가 증가하거나 이들 사이의 거리가 멀어질수록 높은 설치비용과 함께 관리/보수의 어려움 등 시스템의 유연성이 떨어지는 단점을 피할 수 없었다. 이러한 문제점을 개선하기 위해 개발된 분산형 제어시스템(Decentralized Control System) 구조가 필드버스이다[1].

본 연구에서는 다양한 필드버스 프로토콜중에서 국제표준으로 선정되어 시장성이 뛰어나고 데이터 전송시 에러율이 매우 낮아 제어응용에 가장 적합한 CAN 프로토콜을 실시간 데이터 전송이 중요하고 많은 입출력 장치들을 동시에 제어해야 하는 차량내 전자장치 제어부 중의 하나인 사이드미러에 적용함으로써 CAN을 이용한 멀티플렉싱 시스템을 구현하고 이를 분석하였다.

2. 본 론

2.1 CAN(Controller Area Network)

분산형 제어시스템에서는 마이크로프로세서 사이의 통신기능이 필수적이며, 이러한 목적을 위하여 개발된 네트워크 시스템이 필드버스이다. CAN(Controller Area Network)은 1986년 독일의 Bosch GmbH에 의하여 개발되었으며, 최초의 CAN 칩은 Intel에 의하여 1987년에 만들어졌고, 이 후 Philips, NEC 등에서 CAN 칩이 제작되었다. CAN의 특징은 국제표준으로 선정되어 시장성이 뛰어나고(ISO11898, 1993), 높은 전송률과 안정성을 제공하고 있어서 다수의 ECU(Electronic Control Unit)를 상호연결하는 분산시스템의 실시간 제어를 효율적으로 지원할 수 있다[2],[3].

2.2 CAN의 구조

〈그림 1〉은 CAN의 계층구조를 나타내었다. CAN은 OSI 7계층 모델중 물리계층과 데이터링크 계층만으로 구성되며, CAN제어기와 CPU를 묶어 응용계층을 추가함으로써 Network 상에서의 제어를 가능하게 한다[4],[5].

	Application Layer	80C32
Data Link	Object Layer -Message Filtering -Message and Status Handling	PCA82C200 P82C150
	Transfer Layer -Fault Confinement -Error Detection and Signaling -Message Validation -Acknowledgement -Arbitration -Message Framing -Transfer Rate and Timing	
	Physical Layer -Signal Level Bit Representation -Transmission Medium	PCA82C250

〈그림 1〉 CAN의 계층구조

데이터링크 계층은 객체계층(Object Layer)과 전달계층(Transfer Layer)으로 나뉘어지는데, 객체계층은 어떤 메시지가 전송되는지를 찾고, 전달계층으로부터 받은 메시지를 필터링하는 기능을 수행한다(Message Filtering). 즉, 하드웨어와 관련된 응용계층에 인터페이스를 제공하는 역할을 수행하는 부분이 객체계층이며, 이에 반해 전달계층은 CAN프로토콜의 핵심부분으로서, 수신한 메시지를 객체계층에 보내고, 객체계층으로부터 송신해야 할 메시지를 받아들이는 과정을 수행한다.

2.3 CAN 프로토콜

2.3.1 메시지 종류(Message Type)

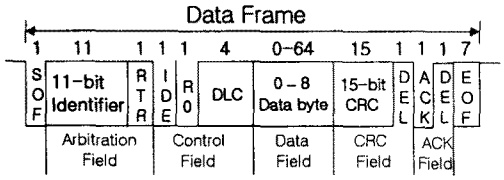
CAN 프로토콜은 4종류의 메시지 프레임을 사용하여 데이터를 전송한다[2].

Type	Function
Data Frame	Data transmission
Remote Frame	Request for data transmission
Error Frame	Global Signaling of errors detect locally
Overload Frame	Adjustment of gap between two consecutive data frames of remote frames

〈표 1〉 CAN의 메시지 프레임 종류

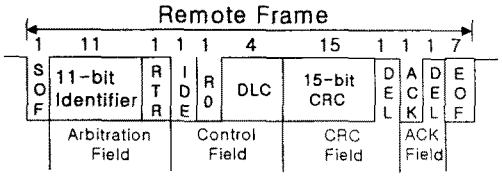
- 데이터 프레임(Data Frame): 데이터를 전송하는 프레임으로 〈그림 2〉와 같이 7개의 필드형식으로 구성되어 있으며, 최대 8바이트의 데이터를 전송할 수 있다. CAN에서는 11비트의 확인자(Identifier)를 통해서 데이터 전송을 위한 버스

액세스 우선순위(Bus access priority)를 조정하게 되는데 확인자값이 작은쪽이 우선순위가 높아서 버스를 액세스할 수 있는 CSMA/AB(Carrier Sense Multiple Access with Arbitrary Bitwise) 방법을 사용하고 있다.



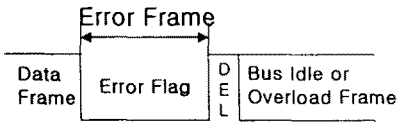
〈그림 2〉 Data Frame 구조

· 전송요청프레임(The Remote Frame): 한 노드에서 다른 노드로 데이터를 요구할 때 전송하는 프레임으로 〈그림 3〉과 같이 6개의 필드로 구성된다. 데이터 필드가 존재하지 않으며, RTR(Remote Transmission Request)비트가 recessive('r') 값을 가지며, 데이터프레임과 동일한 확인자를 갖고 있다.



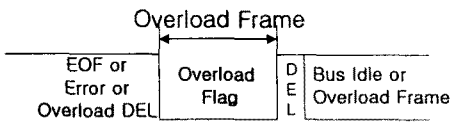
〈그림 3〉 Remote Frame 구조

· 에러프레임(Error Frame): 한 노드에서 수신한 메시지에서 에러를 발견할 때 전송하는 프레임으로 〈그림 4〉와 같이 두 개의 필드로 이루어진다. 여기서 에러플래그(Error Flag)에는 능동에러플래그(Active error flag)와 수동에러플래그(Passive error flag)가 있는데, 수동에러플래그는 6개의 연속적인 dominant('d')로, 수동에러플래그는 6개의 연속적인 recessive('r')비트로 구성된다.



〈그림 4〉 Error Frame 구조

· 오버로드프레임(The Overload Frame): 현재 수행중인 프레임의 다음에 오는 데이터 프레임과 리모트 프레임을 지연시키기 위해 사용된다. 〈그림 5〉에 오버로드프레임을 나타내었다.



〈그림 5〉 Overload Frame 구조

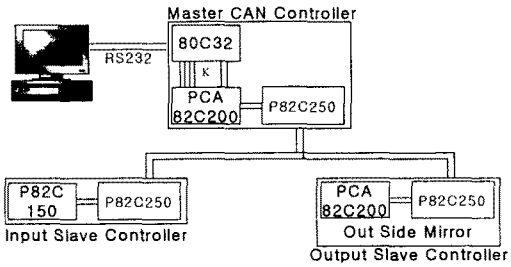
· 인터프레임스페이스(Interframe Space): 데이터프레임과 리모트프레임은 3비트의 인터프레임스페이스에 의해 프레임과 분리된다. 반면 오버로드 프레임과 에러 프레임 앞에는 인터프레임 스페이스가 존재하지 않으며 또, 여러개의 오버로드 프레임은 인터프레임스페이스에 의해 분리되지 않는다.

2.3.1 주소지정 방식과 메시지 우선순위

메시지 필터링(Message Filtering)에 의한 주소지정방식은 한 노드에서 메시지를 보내고자할 때 데이터와 확인자(Identifier)를 함께 보내는데, 수신대기상태에 있는 다른 노드들이 메시지를 수신하면, 확인자를 필터링하여 그 데이터가 자신과 관련성이 있는지 없는지를 판별-각 노드의 하드웨어적인 구성과 응용계층의 설정에 의해 판별한다-하여 관련이 있으면 데이터를 받아들여 저장하고, 없으면 무시하게 된다. 이와 같은 메시지 전달방식은 현재 구성된 CAN 시스템에 새로운 CAN node 등의 스테이션이 추가되더라도 소프트웨어나 하드웨어의 재구성이 필요없는 유연성있는 시스템구축을 가능하게 한다.[2]~[4]

2.4 전체시스템 구성

본 연구에서 적용한 아웃사이드 미러는 상하좌우를 동작시키는데 필요한 2개의 직류전동기와 미러의 위치를 저장하고 로드하기 위한 위치센서, 그리고 미러전체를 개·폐시키는데 필요한 전동기와 미러의 열선으로 구성되었다. 이를 제어하기 위해 〈그림 6〉에서 보는바와 같이 사용자 입력부/사이드미러 출력부 스테이션인 슬레이브컨트롤러(Slave Contro-ller) 2개와 마스터 CAN컨트롤러(Master CAN Controller)를 사용하여 중앙의 CAN Bus를 통해 멀티플렉싱 시스템을 구현하였다.

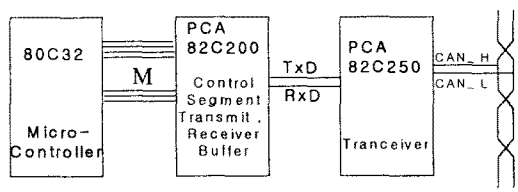


〈그림 6〉 구성된 전체 시스템의 블럭도

2.4.1 마스터 컨트롤러(Master Controller)

〈그림 7〉은 마스터 컨트롤러 구성의 블럭도이다. 본 논문에서는 80C32(Intel)와 PCA82C200(Philips) CAN Controller를 사용하여 마스터 CAN 제어부를 구성하고, P82C250을 CAN 프로토콜 전송기(Tranceiver)로 사용하였다.

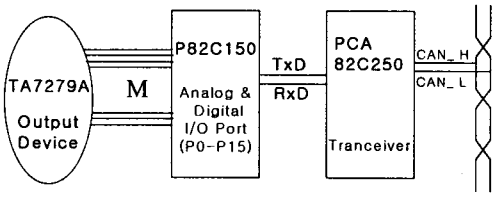
여기서 가장 핵심이 되는 부분이 CAN제어기이며, 이 CAN 제어기 (PCA82C200)는 물리계층(Physical Layer)과 응용계층(Application Layer)사이의 중계역할을 한다. 칩내부에 프로토콜이 내장되어 있어 사용자가 별도의 프로토콜 생성 알고리즘을 응용계층에서 만들 필요가 없으며, CAN버스를 통하여 PCA82C200의 수신버퍼에 들어오는 메시지를 8032의 수신버퍼에 저장하는 방식으로 메시지를 입력받고 이와 반대되는 과정으로 메시지를 출력하는 방식으로 데이터를 처리한다. 즉, CPU인 82C32는 CAN제어기(PCA82C200)의 수신버퍼에 저장되는 데이터를 할당된 번지에 받아들여 그 값을 기억하고, 필요한 시기마다 그 해당번지에서 데이터를 읽어오게 한다.



〈그림 7〉 Master Controller의 블럭도

2.4.2 슬레이브 컨트롤러(Slave Controller)

〈그림 8〉은 슬레이브 컨트롤러(Slave Controller)의 구성에 대한 블록도이다. 슬레이브컨트롤러를 구성하는데는 입출력 제어칩인 P82C150과 PCA82C250(CAN 전송기)를 사용하였다. P82C150은 16개의 아날로그/디지털 입출력 포트가 내장되어 있어 본 연구의 적용시스템인 자동차의 아웃사이드미러의 상하좌우,전동접이, 메모리기능을 수행하기 위해 입력슬레이브 컨트롤러(Input Slave Controller;SLIO1)에 10개의 입력 포트를 설정하였고, 출력 슬레이브컨트롤러(Output Slave Controller;SLIO2)에는 아웃사이드 미러의 구동과 전동접이 그리고, 열선기능을 구현하기 위해 7개의 포트를 출력포트로 이용하였으며, 기준위치저장/로드 및 ADC(Analog to Digital Conversion)를 위하여 4개의 포트를 사용하였다. 따라서 최초 전원이 인가되어 마스터와 슬레이브컨트롤러의 초기화 설정이 끝나면 마스터 컨트롤러인 PCA82C200의 수신 인터럽트(Receive Interrupt)를 활성화시켜 해당 인터럽트가 걸리면 프레임화된 데이터를 분석하여 정해진 기능을 수행하도록 하였다.



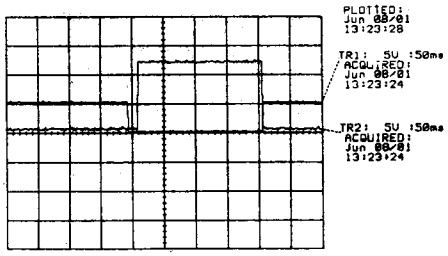
〈그림 8〉 Slave Controller의 블럭도

2.4.5 프로그램

본 실험에서는 C/C-51컴파일러를 사용하여 Slave Controller 1 (SLIO 1)에서 받아들인 사용자 입력에 따라 Slave Controller 2(SLIO 2)의 사이드미러를 제어할 수 있는 프로그램을 작성, CAN Master Controller의 Application Layer(ROM)에 에뮬레이터를 사용하여 다운로딩하는 방식을 사용하였다. 사용된 프로그램의 구조를 간단히 정리해보면,

-step1. 80C32/PCA82C200의초기화(Master Controller)
-step2. P82C150의 초기화
-step3. Mask 레지스터의 설정
-step4. 외부인터럽트 대기
-step5. 외부 인터럽트 실행
-step6. 메세지 오브젝트에서 메시지를 80C32 읽음
-step7. 메시지 전송
-step8. 버퍼에 입력 데이터 저장

2.6 실험결과



〈그림 9〉 입력에서 출력까지의 지연시간

〈그림 9〉는 SLIO1에서 발생한 사용자입력과 이러한 입력값을 Master 제어부에서 사이드미러를 제어하기 위해 SLIO3에 전달하는 메시지의 파형을 비교한 것으로, 사용자입력과 사이드미러를 제어하기 위한 메시지의 출력파형 사이에 지연시간이 발생함을 볼 수 있다. 이와같은 지연시간의 발생원인을 분석해보면, 사용자입력에 따라 발생한 메시지를 SLIO1의 버퍼까지 전달하는데 걸리는 발생지연시간, 메시지가 전송미디어(CAN버스)를 점유하는데 걸리는 대기 행렬시간(Queueing Delay), 메시지가 CAN 버스를 통해 목적노드까지 전송되는데 필요한 전송지연시간(Transmission Delay), 그리고 Master CAN 제어부의 응용계층에서 CAN 응용프로그램을 실행하여 그 결과값을 SLIO3 사이드미러에 전달하는 과정에서 발생하는 전달지연시간(Delivery Delay) 등으로 정의할 수 있다. 이러한 지연시간들은 데이터 필드(Data Field)의 크기와 노드의 증가/감소, 데이터 이동경로의 길이(버스길이) 등에 의해 달라질 수 있으며, 이는 각 시스템 설계시 필수적인 고려사항으로서 지연시간으로 인해 전체 시스템의 효율성이 떨어지지 않도록 해야 한다.

3. 결 론

본 연구에서는 자동차 아웃사이드 미러를 제어하기 위해 CAN을 이용한 멀티플렉싱 시스템을 구현하였으며, 이를 통하여 차량내 각종 ECU(Electronic Control Unit) 장비들을 제어하는데 있어서 배선수를 감소시켜 고장발생요인을 줄이는 한편, 협소한 차체 영역에서 효과적인 실장공간을 제공할 수 있다는 등의 장점을 얻을 수 있었다. 또한, 구현한 멀티플렉싱 시스템은 하드웨어/소프트웨어적인 변경필요없이 새로운 기능을 수행하는 다른 노드를 연결할 수 있는 우수한 확장성 및 유연성을 가지고 있다.

현재, 구성된 CAN을 이용한 사이드미러의 멀티플랙싱 시스템은 저장된 기준위치를 로드하는 과정에서 데이터 병목현상으로 인한 지연시간이 발생하는데, 향후 Master CAN 제어부의 응용계층 설계시 이와같은 시간지연을 고려함으로써 좀더 빠르게 기준위치로 복귀할 수 있는 시스템을 구현하고자 한다.

[참 고 문 헌]

[1] "CAN프로토콜의 개관", 전자기술, pp114-120, 1998년 2월

[2] BOSCH, CAN Specification, Part A,B 1991.

[3] Ken Tindell, Alan Burns, "Guaranteed Message Latencies for Distributed Safety-Critical Hard Real Time Control Network", report YCS229, Department of computer Science, University of York, May 1994.

[4] CiA, CAN Layer, www.can-cia.de, 1997.

[5] 오동진, "복수시스템 제어를 위한 멀티플랙싱 기법 개발", 성균관대학교 공학석사 학위논문, 1999.