전자통신동향분석 제20권 제2호 2005년 4월

지능형 로봇 내부 실시간 제어 통신망 기술

Real-time Internal Control Network Technology for Intelligent Robot

지능형 로봇 특집

임을균 (E.G. Lim)	내장형H/W컴포넌트연구팀 연구원
조재일 (J.I. Cho)	내장형H/W컴포넌트연구팀 선임연구원
황대환 (D.H. Hwang)	내장형H/W컴포넌트연구팀 책임연구원, 팀장

목 차

지능형 로봇은 인간 생활 환경에서 작동하고 일반 소비자를 대상으로 한다는 점에서 기존의 산업용 로봇과는 다른 내부 구조를 가져야 한다. 또한 표준화된 개방 구조를 갖추어 로봇 전용의 부품 산업이 활성화 될 수 있도록 하여 결과적으로 로봇의 단가를 낮출 수 있어야 한다. 본 고에서는 공장자동화와 자동차 분야에서 먼저 도입되고 있는 버스 형태의 제어 통신망이 지능형 로봇 분야에서 어떤 이점을 제공하는지에 대해 살펴보고, 최근의 2족보행로봇에 성공적으로 적용되고 있는 대표적인 실시간 제어 통신망의 하나인 CAN에 대해 분석하고 CAN에 기반한 상위레벨 프로토콜을 소개한다.

Ⅰ. 서론

로봇 산업은 산업용 로봇의 보급 이후에 군사용이나 과학기술용도의 작은 규모의 특정 시장에 국한되어 개발이 이루어지는 정체양상을 보여 왔다. 그러나 1990년대 후반부터 계속해서 발표되는 세계 각국의 2족보행로봇 개발 성공사례와 최근 급속히 가격이 낮아지고 있는 청소로봇의 사례를 보면, 로봇은 머지 않아 인간의 생활 영역에서 작동하게 될 것으로 보인다.

인간 생활 영역에서 동작해야 하는 로봇은 환경에 잘 어울려야 하고 오작동 시 위협이 되지 않아야 하기 때문에 우선 그 크기에 제한을 받게 되며, 주위로부터 가능한 많은 정보를 얻어야 하기 때문에 기존의 산업용 로봇에 비해 여러 종류의 센서를 더 많이 갖추어야 한다. 이러한 센서는 각각의 용도에 따라 로봇 표면상의 특정 위치에 설치되어야 하기 때문에 한 곳에 집중될 수가 없다는 특징이 있다. 로봇의 이동이나 작업을 위한 구동기의 경우도 마찬가지로 각각의 기능에 적합한 위치에 분산되게 된다. 따라서 제한된 크기의 로봇 내부는 센서와 구동기, 그리고 주 제어기 사이의 전기적인 연결로 인해 매우 복잡해지며, 이는 제품의 설계, 대량 생산, 사후 유지 보수 측면에서 매우 불리한 요소로 작용하게 된다. 특히, 기존의 PC 메인보드와 확장슬롯을 이용하여 로봇을 구성하는 경우에는 거의 모든 전선이 로봇의 내부를 가로질러 한 곳으로 집중되는데, 유지 보수를 위해 와이어 정형이라도 하면 전선의 총 길이가 필요 이상으로 길어지게 되고 와이어 하니스(harness)의 한쪽 끝이 매우 굵어지게 되어 구조 설계의 유연성을 저해하게 될 것이다.

이 문제점은 공장 내부에 분포된 자동화설비에 대한 중앙집중식 제어에서도 동일하게 나타나는 문제점이며, ECU와 운전석 주위에 배선이 집중되는 자동차에서도 적극적으로 해결하고자 하는 문제점인데, 많은 부분에서 로봇의 경우와 유사하다. 연결된 각각의 기능 단말들이 실시간에 가깝게 정보를 주고 받으며 서로 유기적으로 동작해야 하기 때문에, 실시간 전송이 어느 정도 보장되는 제어용 네트워크를 도입하는 추세이다.

따라서 공장자동화 분야와 자동차 분야에서 도입하고 있는 실시간 제어 통신망을 분석함으로써 로봇 내부의 제어통신에 대한 해결책을 도출할 수 있을 것이다.

Ⅱ. 제어용 통신망의 종류

과거에는 대부분 제어대상 시스템의 규모가 크지 않고 단순하여 단일제어기로 제어가 가능하였으나 근래에 와서는 제어대상 시스템의 규모가 커지고 그 안에 투입되는 계기 및 구동기기 등이 정밀해지고 성능이 향상되고 있으며 그 수가 증가하는 추세에 있다. 이와 같이 제어 대상의 수가 증가하고 처리해야 하는 정보의 양도 늘어나게 되어 과거와 같이 단일 제어기를 이용한 방식에는 한계가 나타나고 있다. 이러한 한계를 극복하기 위해 제어기를 다수 분산 배치하여 각자 독자적으로 시스템의 정보를 수집하고 제어를 수행하며 동급의 제어기 및 상하위 제어기들 간에는 통신을 통해 정보를 교환하는 방식의 분산제어 시스템과 산업용 통신망이 등장하게 되었다[1].

산업용 자동화 시스템의 하위 레벨에 위치한 센서, 모터, 로봇, 논리연산제어장치 등 필드레벨의 기기를 통제하는 산업용 네트워크를 필드버스(fieldbus)라 하며 디지털 방식을 기반으로 하고 있어 다량의 신호를 원거리까지 전송할 수 있고 현장기기나 입출력 모듈의 엄청난 가닥 수의 케이블 대신 한 가닥의 네트워크 선을 사용함으로써 배선을 절약할 수 있고 설치를 간소화 할 수 있다[1].

소형 완구용 로봇과 기존의 각종 산업분야에서 제어나 계측 용도로 사용되는 제품에서 지원하는 통신망 인터페이스 위주로 산업용 제어 통신망을 몇 가지로 구분해 보면 다음과 같다.

1. 스타 토폴로지

대부분의 PC와 산업용 원보드 컨트롤러, 마이크

로프로세서에서 기본으로 제공하는 RS-232C가 이에 해당한다. 이 인터페이스를 지원하는 기능 단말로는 KEYENCE의 레이저 변위계 등이 있다. 호스트와 단말이 1:1로 연결되어야 하기 때문에 하나의 상위 컨트롤러에 연결할 수 있는 단말의 개수가 1개나 2개로 제한되는 단점이 있다.

그러나, 로봇의 메인프로세서 모듈로서 PCI 슬롯을 갖춘 PC 메인보드를 사용한다면, MOXA 등에서 판매하는 시리얼포트 확장 카드를 이용하여 하나의 PC 메인보드에 RS-232C 단말장치를 16개까지도 연결할 수 있다.

하지만, 서론에서 언급한 바와 같이 스타 형태의 제어망은 로봇에 적용하기 곤란한 측면이 있다. 예를 들어 로봇의 바깥 표면을 따라 여러 개가 배치되어야 하는 초음파 센서나 근접 센서에 RS-232C를 적용하면, 메인 프로세스 모듈로 집중되는 배선은 로봇 내부의 한 단면을 차지하게 되는데, 한정된 공간 내부에 높은 밀도로 부품들이 배치되기 마련인 로봇의 특성상 생산성과 유지 보수성이 매우 나빠지게 된다. 와이어 정형을 통해서 로봇의 내부 공간을 순서대로 지나가는 선의 형태로 배치한다고 하더라도 와이어 하니스의 굵기는 로봇 내부 설계에 하나의 제약이 된다. 여전히 다른 방식에 비해 구성 가능한 노드 수가 비교적 적은 것도 단점이다.

RS-232 단말을, 멀티드롭 구성이 가능한 RS-485 버스나 CAN 버스에 접속할 수 있도록 하는 컨버터가 많이 출시되고 있기 때문에 RS-232 단말을 사용한 로봇 구성이 전혀 불가능한 것은 아니지만 단말 개수만큼의 컨버터 비용과 공간이 추가로 필요하다는 점에서 로봇의 단가를 낮추는 효과도 크지 않을 것으로 예상된다.

2. 버스 토폴로지(멀티드롭, 데이지 체인)

일직선의 버스에 여러 개의 단말이 연결되는 형태는 로봇의 구성에 있어서 스타 토폴로지에 비해 유리하다. 앞서 논한 초음파 센서같이 로봇의 바깥 면을 따라 설치되는 센서들에 버스 형태의 제어망 인터페이스를 적용하면 로봇 내부로 배선이 가로지르는 경우를 현저히 줄일 수 있으며, 직렬로 연결되는 액추에이터에 데이지 체인 방식으로 적용하면 배선 집중으로 인한 와이어 하니스의 단면적이 증가하는 경우도 없게 된다.

가. UART

국내 완구형 소형 휴머노이드 로봇업체인 메가로보틱스의 AI MOTOR는 데이지 체인 방식으로 서로 연결되어 하나의 소형 로봇을 구성하는 소형 액추에이터 모듈이다. 연결하고자 하는 대상이 수 cm 이내로 매우 가깝고 총 연결길이가 수십 cm 이내인 점을 이용하여, 드라이버단을 생략하고 UART의 TTL 레벨 출력끼리 바로 연결함으로써 버스 방식으로 구성되도록 하였다. 매우 가깝고 이상적인 환경에서, 한정된 용도로만 사용이 가능하다.

나. RS - 422/485

UART의 드라이버 규격 중에서 버스를 구성할 수 있는 규격이다. 디퍼런셜 드라이브 방식이므로 전송 거리가 길고, 하나의 라인에 여러 개의 단말을 설치할 수 있는 멀티드롭(multi-drop) 방식의 구성이 가능하므로 여러 개의 단말로 구성되는 네트워크의 배선을 일렬로 간단하게 구성할 수 있다.

RS-422는 하나의 송신 단말과 32개까지의 수신단말로 이루어지는 1:N 구성이 가능하며, RS-485는 최대 32개의 단말로 N:N 구성이 가능하다.

프로토콜을 원하는 대로 구성할 수 있다는 장점이 있으나, 여러 단말간의 데이터 전송 중재를 위한 하드웨어적인 방안이 없기 때문에 소프트웨어적인 처리에 전적으로 의존하고, 표준화된 프로토콜이 거의 없다는 단점이 있다.

국내 업체 로보티즈에서 개발하여 판매하고 있는 완구형 소형 휴머노이드 로봇용 액추에이터 시리즈에 데이지 체인 방식으로 RS-485가 적용되었다.

다. CAN

RS-422/485와 마찬가지로 버스 방식이며 높은

신뢰성을 바탕으로 산업용 기계나 자동차용 전장 부품의 통신망 인터페이스로 점차 사용되기 시작하는 규격이다. 다양한 산업용 마이크로프로세서에 기본적으로 CAN 컨트롤러가 포함되는 추세이며, 모토롤라의 LITE5200처럼 CAN 인터페이스를 갖춘 산업용 원보드 컨트롤러도 출시되고 있다. CAN을 기반으로 몇 가지 종류의 상위 레이어 프로토콜이 정의되어 있기 때문에, 서로 다른 제작사 간의 호환이 어느 정도 가능한 장점이 있다. 이를 이용해서 타 산업 분야의 부품을 로봇에 적용할 가능성도 충분하다고 본다.

Ⅲ. CAN 개요

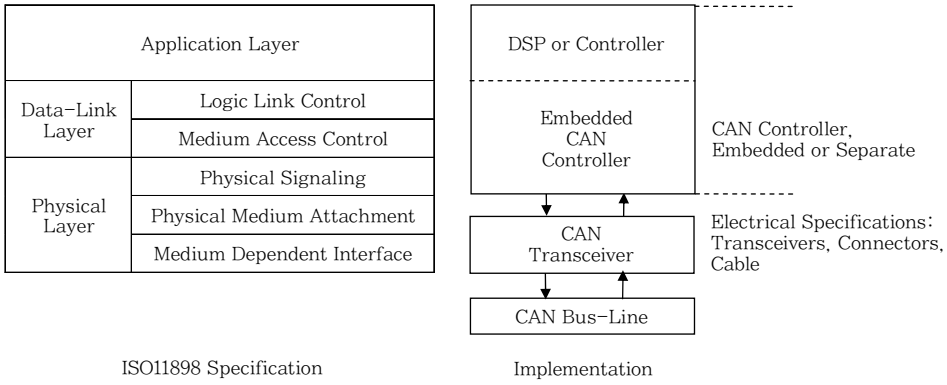
CAN은 Bosch가 차량제어를 위한 통신망으로 개발하였다. 차량용 제어 버스 시스템의 요구사항과 산업용 필드버스 시스템을 비교해보면 저가격, 열악한 전기환경에서의 작동성, 실시간 처리능력, 사용의 용이성과 같은 많은 유사점을 발견할 수 있다. 농기계와 선박기계의 제조업체도 CAN을 선택하고 있으며 특히 안전에 관해 특별한 요구가 있는 의공학 분야와 로봇 및 운송 시스템의 경우에도 CAN에 대한 인식이 좋아지고 있다. 신뢰성이 높은 전송임에도 불구하고 낮은 접속 가격을 제공하므로 가격이 중요한 제조 분야에서 CAN은 아주 중요한 역할을 할 수 있으리라 기대되고 있다.

CAN 규약은 ISO11898에 의해 표준화 되었으며 CiA 협회에서는 CAN을 기반으로 하여 ISO/OSI 계층모델에서 응용 계층에 해당하는 통신망 규약을 개발하였다.

CAN에 의해 데이터가 교환될 때 어떠한 노드도 주소화되지 않고 통신망에서 유일하게 존재하는 메시지 식별자(Identifier: ID)에 의해서 메시지의 내용(예를 들면 rpm 혹은 엔진 온도 등)과 같은 데이터가 교환된다. 즉, 데이터 혹은 메시지는 노드의 물리적 주소로 그 데이터의 수신지를 정하지 않고 메시지 ID를 이용하여 그 데이터의 사용처를 정한다. CAN의 주요 특징은 다음과 같다[2].

- 저비용
- 극대화된 견고성
- 빠른 데이터 전송 속도(최대 1Mb/s)
- 신뢰성. 탁월한 오류 처리와 오류 제한 기능들
- 결함 메시지들의 자동적인 재전송
- 물리적으로 결함이 추정되는 노드들의 자동적인 버스 연결절단
- 기능 위주의 어드레스 – 데이터 메시지들은 소스 혹은 목적지 주소들을 포함하지 않으며, 그들의 기능과 우선순위에 연관된 식별자들만을 포함

CAN 통신 프로토콜(ISO11898)은 (그림 1)과 같이 OSI 7 layer 중 하위 2개의 layer(data-link, physical)에 대해 정의한다[3].



(그림 1) ISO11898 Standard Architecture

〈표 1〉 CAN 버전

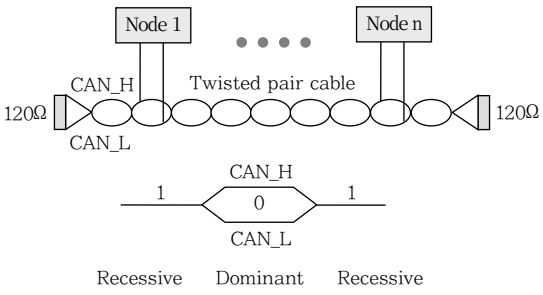
Nomenclature	Standard	Max. Signaling Rate	Identifier
low-speed CAN	ISO11519	125kbps	11bit
CAN 2.0A	ISO11898:1993	1Mbps	11bit
CAN 2.0B	ISO11898:1995	1Mbps	29bit

〈표 1〉은 CAN 표준을 버전별로 요약한 것이다. 초기 ISO11519(low speed CAN)의 경우 11비트의 Identifier를 갖고 125kb/s 속도로 동작되며 그 이후에 ISO11898(1993년)에서 전송속도를 1Mb/s까지 끌어올렸고 이를 CAN 2.0A 또는 Standard CAN이라 한다. Identifier가 11비트이므로 총 2048개의 다른 메시지 Identifier를 가질 수 있다. 그 이후에 Identifier를 29비트로 확장한 규격을 CAN 2.0B 또는 Extended CAN이라 한다[3].

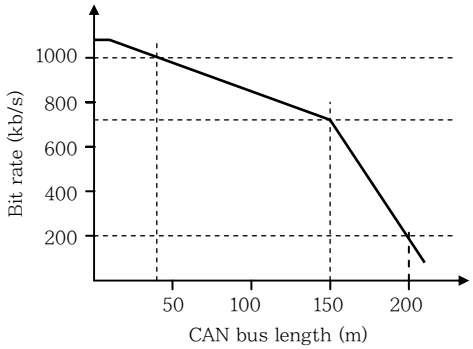
1. CAN Physical Layer

가. CAN Bus Topology

CAN의 전송매체로는 CAN의 bus arbitration에 사용되는 dominant(logical 0)와 recessive(logical 1)에 대한 요구사항을 만족하면 모두 사용할 수 있으므로 전기적 매체 또는 광케이블 또는 파워라인과 무선 전송도 가능하다. 가장 널리 쓰이는 전송매체로는 differential로 구동되는 twisted pair cable이다. (그림 2)는 twisted pair cable을 사용한 CAN bus connection을 나타낸다. CAN_H와 CAN_L는 differential로



(그림 2) CAN Bus Connection



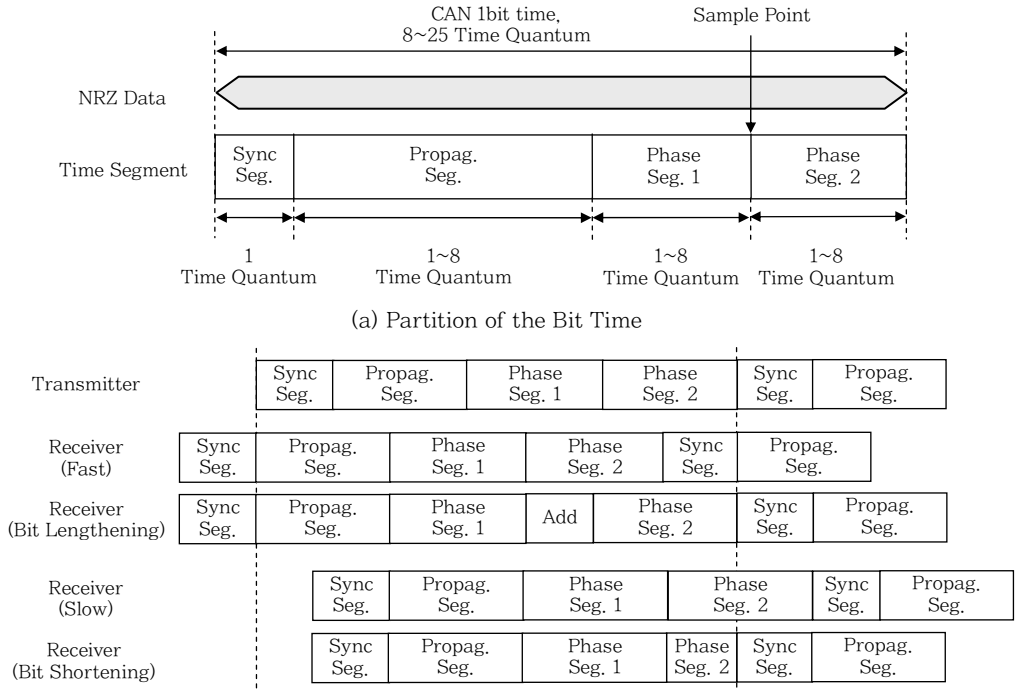
(그림 3) CAN Bus Line by Twisted Pair Cable

구동되기 때문에 외부 electrical interference에 높은 면역력을 가지고 있다[3].

CAN 버스의 길이를 늘릴 경우 신호전파, 라인 저항 등 전송매체 특성을 고려하여야 하며 (그림 3)은 twisted pair cable을 이용할 경우의 버스 라인별 최대 전송속도를 보여준다. 1Mb/s의 고속 전송을 할 경우 최대 CAN 버스 길이는 약 40m로 알려져 있다[4].

나. CAN Bit Timing

CAN 프로토콜은 NRZ 비트 코딩을 사용한다. NRZ 코드의 특성상 연속적인 같은 극성의 비트를 전송할 때 클럭 동기(synchronization)에 이용될 수 있는 비트 천이(edge)를 제공할 수 없다는 단점이 있다. 이의 해결 방안으로 bit stuffing 방식이 이용되는데 이 방식은 송신측에서 5개의 연속적인 같은 극성의 메시지를 전송한 후에는 반드시 그와 극성이 반대되는 비트 한 개를 추가로 삽입(stuffing)하고 수신측에서는 추가로 삽입된 비트를 제거(destuffing)하여 본래의 데이터를 복구하게 된다.



(그림 4) CAN Bus Bit Timing

CAN 버스의 모든 노드들은 CAN 프레임의 시작점인 SOF 비트의 falling edge에 동기(hard synchronization)되며 마지막 비트까지 정확하게 샘플링하기 위해 CAN 노드들은 메시지에 포함된 recessive와 dominant edge에 지속적으로 동기(re-synchronization)되어야 한다.

(그림 4) 는 CAN에서 이용되는 비트 time과 동기방식에 대한 그림이다. 먼저 (그림 4a)는 CAN의 1bit time을 각 노드에서 사용하는 oscillator 자체 클럭으로 작게 분할한 것을 나타낸다. 분할 단위는 TQ이라 하며 그림에서처럼 CAN 데이터 1비트는 1개의 TQ로 이루어지는 synchronization segment와 1~8개의 TQ로 이루어지는 propagation time segment, 재동기 과정에서 늘리거나 줄일 수 있는 phase buffer segment 1과 2로 나눌 수 있다. 샘플링 시 유효한 비트 값으로 예상되는 위치인 샘플 포인트는 phase buffer segment 1과 2 사이에 위치한다[4].

(그림 4b)는 송신측보다 수신측이 빠른 경우와 느린 경우의 재동기 과정을 나타낸다. 수신측이 송신측에서 전송된 데이터보다 빠른 시점에서 sync segment를 예상했으므로 이때에는 phase buffer segment 1의 TQ를 늘려 재동기를 하게 되며 반대로 수신측이 송신측에서 전송된 데이터 보다 느린 시점에서 sync segment를 예상했으므로 이때에는 phase buffer segment 2의 TQ를 줄여 재동기를 하게 된다.

다. CAN Frame Format[5]

(그림 5)에 CAN에 사용되는 프레임 포맷을 나타내었다.

프레임 포맷의 각 필드별 내용을 살펴보면 다음과 같다.

• SOF

1비트의 dominant(Logical 0)로서 메시지의

(a) Standard CAN Frame Format

(b) Extended CAN Frame Format

(그림 5) CAN Frame Format

시작점을 나타내며 모든 노드들은 전송을 시작하는 노드의 SOF에 의해 만들어지는 falling edge에서 동기화 된다.

• Identifier

메시지의 종류 및 우선순위를 나타내며 (낮은 바이너리 값이 높은 우선순위) 이를 통해 버스 액세스 경쟁이 이루어진다.

• RTR

Remote Transmission Request 비트이며 데이터 프레임인지 원격 프레임인지를 구별하게 한다. 데이터 프레임이면 dominant 값을 가지고 원격 프레임이면 recessive 값을 가진다.

• IDE

Identifier Extension 비트로서 standard 프레임인지 extended 프레임인지를 구분한다. Standard 프레임이면 dominant 값을 가지고 extended 프레임이면 recessive 값을 가진다.

• r0

Reserved bit를 나타내며 추후 사용을 위한 예비비트이다.

• DLC

Data Length Code(4비트)로서 전송될 데이터 바이트의 크기를 나타낸다. CAN에서는 최대 8바이트의 데이터를 보낼 수 있으므로 최대값은 "1000" 이다.

• Data

Payload 데이터 영역으로 보내려고 하는 정보를 최대 8바이트까지 담게 된다.

• CRC

Cyclic Redundancy Check(CRC 15비트+CRC Delimiter 1비트)로서 전송된 비트들의 checksum을 담고 있으며 전송에러를 검사하는데 사용한다.

• ACK

ACKnowledge(ACK 1비트+ACK Delimiter 1비트)로서 메시지를 올바르게 받은 수신 스테이션이 ACK 필드를 받는 바로 그 순간에 ACK 슬롯의 비트 값을 dominant 값으로 세팅한다. 만약 이 영역에서 recessive 비트가 검출되면 송신측에서는 데이터 전송이 올바르게 이루어지지 않았음을 알 수 있게 된다.

• EOF

End of Frame(7비트)으로서 7bits의 recessive로 구성되고, 메시지의 전송이 끝났음을 나타낸다.

• IFS

Inter Frame Space(7비트)로서 다음 사이클을 준비하기 위해 요구되는 시간이다.

• SRR

Substitute Remote Request로서 extended

프레임에서 사용되며 standard 프레임의 RTR 비트 자리에 위치하여 recessive 값을 가진다.

• r1

Reserved bit를 나타내며 추후 사용을 위한 예비비트이다.

2. CAN Data-link Layer

◎ CAN Message Type[5]

CAN 버스를 통해 서로 다른 4개의 프레임(메시지)이 송수신 된다. 이들은 각각 data frame, remote frame, error frame 그리고 over-load frame으로서 data frame의 경우 송신단에서 수신단으로 데이터를 보내는 데 사용되고 remote frame은 같은 identifier를 갖는 data frame의 송신을 요청하는 데 사용되며 error frame은 버스 상의 에러와 관련된 기능을 처리하기 위해 사용되며 overload frame은 이전 data/remote frame과 다음 data/remote frame 사이의 추가 delay를 제공하기 위해 사용된다.

가. Data Frame

Data frame은 가장 일반적인 형태의 메시지 형태이며 앞서 언급된 (그림 5)의 프레임 형태를 가진다. 앞 절의 프레임 비트의 상세 설명을 참조하기 바란다.

나. Remote Frame

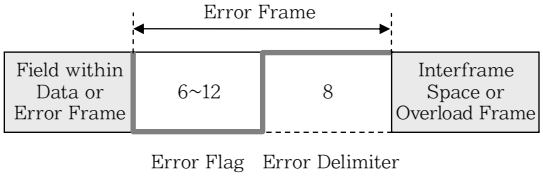
Remote frame은 다른 노드로부터 데이터를 전송해줄 것을 요청하는 데 사용되며 data frame 형태와 유사하나 RTR 비트 값이 recessive 값이 되며 데이터 필드가 없다는 차이가 있다. (그림 5)의 프레임 형태를 참조하기 바란다.

다. Error Frame

Error frame은 버스상의 에러를 검출한 임의의 노드에서 출력된다. Error frame은 (그림 6)과 같이 error flag(6~12비트)와 error delimiter(8비트 recessive)로 구성된다. Error flag는 에러를 검출한 노드의 상태(error status)에 따라 두 가지 형태로 나뉜다. Error status는 뒤에 언급되는 error handling 부분을 참고하기 바란다.

먼저 버스의 에러를 검출한 노드의 error status가 error-active인 경우에는 버스 에러를 검출하자 마자 bit stuffing 규칙을 깨는 active error flag를 발생시켜 현재 버스 상에 있는 메시지의 전송에 에러를 삽입시켜 현재 버스에 에러가 있음을 다른 노드들에게 알리게 된다. 이 active error flag는 연속적인 6개의 dominant 비트로 구성된다. CAN 버스 규격상 5개의 연속적인 값 후에는 그 반대되는 극성의 비트를 삽입(bit stuffing)시켜야 하는 규칙을 깨버리게 되어 버스에 물려있는 다른 노드들 역시 자체적으로 error frame을 발생시키게 된다. 여러 노드들에서 발생된 error frame에 의해 버스의 error flag는 6~12 비트의 dominant 비트로 만들어지게 되며 이후의 error delimiter로 error frame이 마무리된 후 버스는 다시 정상 상태로 돌아가게 되고 error flag에 의해 전송이 중지된 노드는 다시 재송신을 하게 된다.

다음 버스의 에러를 검출한 노드의 error status가 error-passive인 경우에는 버스 에러를 검출하자 마자 passive error flag(6개의 recessive bit)를 발생시키게 되고 전체 error frame은 error delimiter까지 포함하여 총 14비트의 recessive 비트를 출력하게 된다. 이 경우 실제 송신측에서 버스 에러를 검출하지 못하는 한 error-

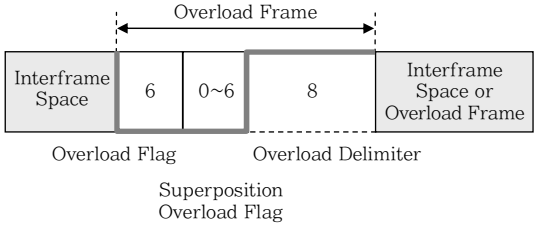


(그림 6) Error Frame Format

passive 상태의 노드에서 발생된 error frame은 무시되게 된다. 그러나 송신측 노드가 error passive flag를 발생하게 된다면 이는 stuffing 규칙을 어기게 되어 다른 노드들이 error frame을 발생하게 만들게 된다.

라. Overload Frame

Overload frame은 active error frame과 같은 포맷으로 이루어진다. 그러나 overload frame은 interframe 동안에만 만들어진다. Overload frame은 (그림 7)과 같이 overload flag(6~12비트)와 overload delimiter(8비트 recessive)로 구성된다. Overload flag는 다른 노드에서 만들어진 overload flag 이후에 6 비트의 dominant 비트로 구성된다. Overload frame은 노드가 내부 사정으로 다음 번 메시지를 받을 준비가 되어 있지 않을 경우 발생시키게 된다.



(그림 7) Overload Frame Format

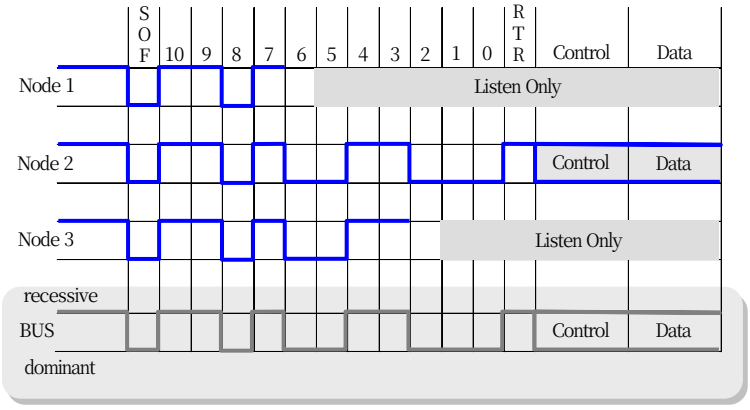
3. CAN Bus Arbitration

CAN 통신 프로토콜은 CSMA/CD+AMP 또는 CSMA/CD-NBA라는 메시지 전송 메커니즘을 가지고 있다. CD+ AMP 또는 NBA는 메시지의 identifier 필드에 미리 할당된 우선순위에 따라 bit-wise arbitration을 통해 충돌을 피하는 방식이다. 높은 우선순위의 identifier(높은 우선 순위의 메시지)가 버스를 액세스하는 경쟁에서 이기게 된다.

CAN 버스는 "1"로 표현되는 recessive 상태와 "0"으로 표현되는 dominant 상태 두 가지를 가지고 있다. CAN 버스에 연결되어 있는 노드들은 모두 한 전송선로를 사용하게 되므로 각 노드의 출력들은 wired AND 형태로 묶이게 된다. 이는 dominant(0)한 상태가 recessive(1)한 상태보다 우선하게 됨을 의미한다.

두 개 이상의 노드가 동시에 메시지를 전송하려고 할 때 각 메시지는 서로 식별자를 한 비트씩 비트 단위로 비교하여 제일 높은 우선 순위의 메시지(가장 낮은 식별자 값을 가진 메시지)는 전송되고 낮은 우선 순위의 메시지들은 전송을 중단하게 된다. 전송하는 노드는 한 비트를 보낼 때마다 버스를 모니터링 한다. 어떤 노드가 recessive 비트를 보냈는데 버스를 모니터링한 결과 dominant 비트가 검출되었다면 이는 버스 내의 다른 노드가 자기보다 높은 우선 순위의 메시지를 전송하고 있는 것이므로 해당 노드는 메시지의 전송을 즉시 중단하고 수신모드로 전환하게 된다. 전송을 중단한 노드는 버스의 상태를 계속 감지하여, 버스가 다시 비활성 상태가 되면 자동적으로 다시 메시지의 전송을 시작하게 된다.

(그림 8)은 CAN bus arbitration 방식을 그림으로 나타내었다. 먼저 임의의 노드에서 전송을 개시하기 위해 SOF를 dominant 상태로 만든다. 이에 맞추어 나머지 노드들도 동기화되어 전송을 시작한다. 각 노드들은 SOF 이후의 identifier 비트를 bit 10부터 차례로 버스에 출력한다. (그림 8)의 예에서는 node 1의 경우 Identifier[10:0]= 110111… 이고 node 2의 경우 Identifier[10:0] =11010011000, node 3의 경우 Identifier [10:0]=110100111…이다. Identifier의 bit[6]에서 node 1은 recessive를 출력했는데 버스의 상태는 dominant이므로 node 1은 자신보다 더 높은 우선순위의 메시지가 버스를 액세스 하고 있음을 판단하여 바로 수신모드로 전환한다. 이 후 bit[2]에서 역시 node 3은 recessive를 출력했는데 버스의 상태는 dominant이므로 node 3은 자신보다 더 높은 우선순위의 메시지가 버스를 액세스하고 있음을 판단하여 바로 수신모드로 전환한다.



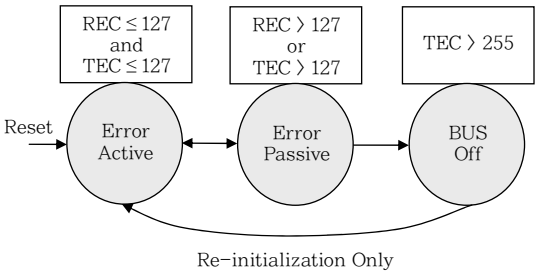
(그림 8) CAN Bus Arbitration

결국 버스는 node 2가 점유하게 되어 node 2의 송신을 마치게 된다.

4. CAN Error Handling

CAN 버스의 에러는 5가지 방식으로 검출된다. 이들은 각각 CRC error, ACK error, frame error, bit error 그리고 stuffing error이다.

먼저 CRC error는 프레임 포맷 속에 할당된 CRC 필드와 수신된 프레임으로부터 계산된 CRC 값을 비교하여 에러를 검출하게 된다. 수신단에서 수신된 CRC 필드와 계산된 CRC 값이 일치하지 않는 경우에는 수신된 메시지를 버리고 error frame을 송신하여 송신측에 재송신을 요청하게 된다. ACK error는 송신측에서 메시지의 acknowledge 필드를 검사하는 것으로 적어도 한 개의 노드에서 수신을 정확하게 하였을 경우 ACK 필드를 dominant한 값으로 overwrite하게 되어 전송이 정상적으로 이루어졌음을 나타내게 된다. Frame error는 송신측에서 CRC delimiter, acknowledge delimiter, end of frame, interframe space 영역의 값을 검사하게 된다. Bit error는 송신측에서 arbitration 필드 외의 비트에서 송신한 비트와 다른 값이 읽힐 경우 발생된다. Stuffing error는 SOF부터 CRC delimiter까지의 영역에서 같은 극성의 비트가 연속적으로 6개 읽힐 경우 발생된다. 이상과 같은 에러



(그림 9) Error Status(Fault Confinement)

가 검출될 경우 모든 노드에 error frame을 전송하여 에러가 섞여 있는 메시지를 버리고 새로 재전송을 수행한다.

각 CAN 노드는 내부의 error 카운터 값에 따라 (그림 9)와 같은 error-active, error-passive 그리고 bus off의 3가지 error status를 가진다. Reset 후 각 노드는 error-active 상태이며 이 상태에서는 어떤 제한 없이 active error frame (dominant bit)을 송신할 수 있게 된다. CAN 통신중에 발생되는 수신과 송신에러에 따라 내부 error 카운터(REC, TEC) 값이 증가하게 된다. REC 또는 TEC가 128을 넘어가면 해당 노드는 error passive 상태로 천이된다. 이 상태에서는 더 이상 버스의 에러에 적극적으로 대응하지 못하게 된다. 즉 passive error frame(recessive bit)만을 내놓을 수 있게 된다. 이 상태에서 REC 또는 TEC가 128 아래로 내려가면 다시 error-active 상태로 복귀된다. 그러나 TEC가 255 이상 올라가

면 bus-off 상태로 천이하며 해당 노드는 모든 버스 운용을 중단하게 된다. 이것은 동작에 문제 있는 노드를 CAN 버스에서 자동적으로 끌어내게 되는 효과를 나타내는 것이다. 이 상태에서 다시 error active 상태로 천이는 CAN node를 다시 initialization할 때만 가능하다[4].

Ⅳ. CAN 상위레벨 프로토콜

CAN 버스에 CAN 인터페이스를 가진 센서와 액추에이터를 노드로써 연결해 놓는다고 해서 곧바로 동작하는 것은 아니다. 단순히 각각의 노드와 CAN 프레임의 identifier 필드를 일대일로 대응시켜 사용한다면 각각의 노드는 단 한 종류의 메시지 밖에 전송하지 못할 것이며 낮은 우선순위의 identifier를 배정받은 노드는 메시지 전송이 거의 불가능할 수도 있다. 따라서 identifier 필드는 단순한 노드의 식별자 기능보다 훨씬 많은 기능을 포함해야 한다.

CAN 상위레벨 프로토콜은 현재의 하위 CAN 계층(물리 계층과 데이터 링크 계층) 위에서 실행되는 프로토콜로서 identifier 필드와 data 필드의 일부 바이트를 사용하여 CAN 본연의 실시간 데이터 전송부터 노드의 프로그램 교체를 위한 대용량 데이터 전송, 노드들의 동기 작업, 네트워크 관리 등 네트워크 전반에 걸쳐서 다양한 기능을 제공한다.

여러 단체들이 시스템 통합을 쉽게 하기 위해서 표준화된 개방형 애플리케이션 계층들을 개발하고 있다. 몇 가지 이용할 수 있는 CAN 상위 단계 프로토콜들은 다음과 같다[2].

- Open DeviceNet Vendors Association의 DeviceNet
- Honeywell의 Smart Distributed System(SDS)
- CiA의 CAN Application Layer(CAL)
- CiA의 CANOpen(subset of CAL)
- 스웨덴 Kvaser의 CANKingdom

특히 CANopen의 경우에는 CAN 프레임의 identifier와 장치 내부 변수의 대응 관계를 정의하는 object dictionary라는 것을 각각의 장치마다 유지하는데 이 중의 일부 영역을 표준 device profile로 정의하여 서로 다른 제조사 제품 사이의 호환성을 도모하고 있다. 이에는 센서와 서보모터에 대한 표준안도 포함되어 있으며 이를 지원하는 행동제어 모듈(motion controller)이 여러 회사에서 시판되고 있기도 하다. 이 표준을 따르는 센서와 서보모터를 로봇에 사용할 수 있도록 로봇의 표준 아키텍처를 정하든가, 로봇에 알맞은 표준 device profile을 제안하여 로봇 부품 시장을 활성화하는 것도 장기적으로는 로봇 산업에 도움이 될 것이다.

하나의 시스템에 CANopen을 적용한다는 것은 수많은 메시지의 identifier와 각각의 노드마다 내장해야 하는 object dictionary들을 관리해야 하는 것을 뜻하는데 이를 위한 종합관리 프로그램이 시중에 많이 나와 있다.

Ⅴ. 결론

지능형 로봇 내부의 센서와 액추에이터 등을 메인 프로세서와 연결할 때 좀 더 효과적인 배선 설계가 가능하고 설치를 간소화 할 수 있도록 하는 serial 제어 버스에 대해 검토하였으며 특히 그 내부 구성이 유사하리라 생각되는 차량내부 제어에 대해 널리 사용되고 있는 CAN에 대해 검토하였다.

CAN은 최대 8바이트로 제한된 한 CAN 프레임 내에 오버로드가 많아 전송대역의 손실이 있긴 하지만, 현재 개발되어 발표되고 있는 2족보행로봇에서 성공적으로 사용되고 있기도 하다.

그러나, 로봇 시장을 열어나갈 수 있는 현실적인 가격 구현과 사용자의 다양한 요구를 만족시킬 수 있는 다양한 로봇 부품의 개발과 적용을 위해, 유연한 확장성과 전문적인 제어망 관리 기능을 제공하는 상위 레벨 프로토콜을 적극적으로 도입할 필요가 있겠다.

최근에는 제어 유니트들 간의 데이터 전송 용량이 증가함에 따라 높은 데이터 전송 속도를 제공할 수 있는 TDMA 방식에 기초한 TTP가 도입되고 있으므로 이에 대한 검토 또한 지속적으로 이루어져야 할 것이다.

약어 정리

AMP	Arbitration on Message Priority
CAN	Controller Area Network
CD	Collision Detection
CiA	CAN in Automation
CSMA	Carrier-Sense Multiple-Access protocol
ECU	Electronic Control Unit
ISO	International Organization for Standardization
NBA	Non-Destructive Bit-wise Arbitration
NRZ	Non-Return-to-Zero
OSI	Open System Interconnection
REC	Receive Error Counter
TDMA	Time Division Multiple Access
TEC	Transmit Error Counter
TQ	Time Quantum
TTP	Time Triggered Protocol

참 고 문 헌

[1] 권욱현, 김형석, 김동성, "산업용 필드버스 통신망," 성안당, 2004. 10.

[2] Embedded Systems Korea, "CAN/LIN 기술정보," http://www.eskorea.net.

[3] Steve Corrigan, "Introduction to the Controller Area Network," Texas Instruments, 2002. 8.

[4] Gerhard Goller, "Atmel Wireless & Microcontrollers CAN Tutorial," Atmel, 2000. 10.

[5] BOSCH, "CAN Specification Ver2.0," 1991, http://www.can.bosch.com/docu/can2spec.pdf.