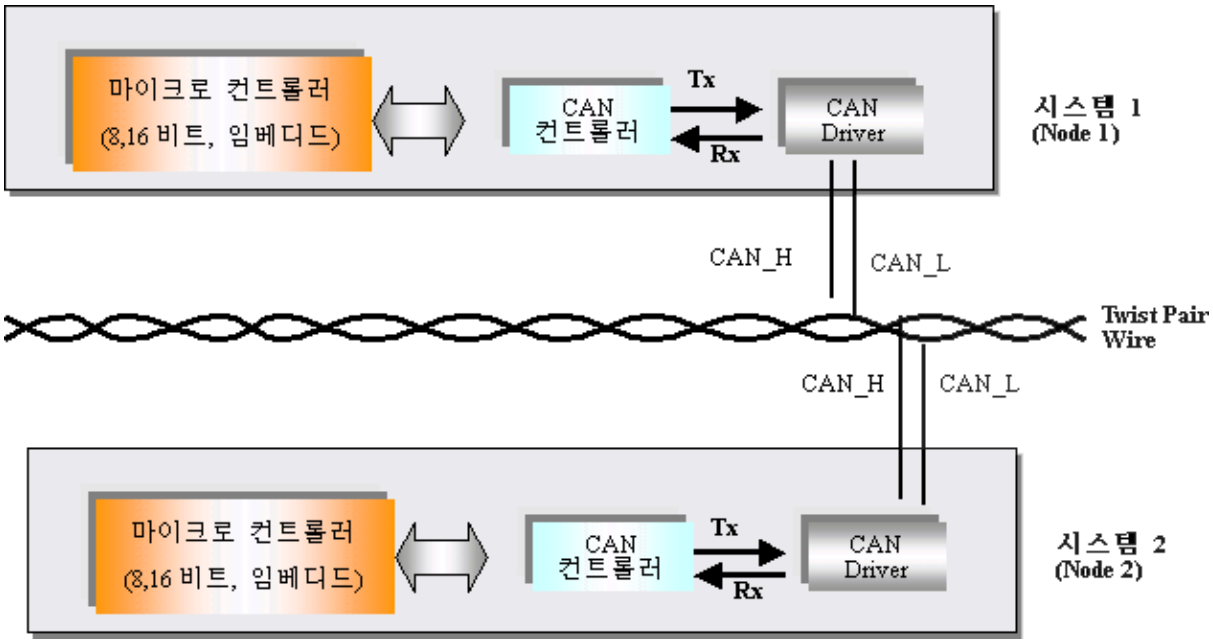
CAN 개요

정의

CAN 은 처음에 자동차 산업 분야에 적용하기 위해 고안된 시리얼 네트웍 통신방식입니다. 근래에는 자동차 분야뿐만 아니라 산업 전 분야에 폭 넓게 적용되고 있으며 기본적인 시스템 구성은 아래와 같습니다.



특징

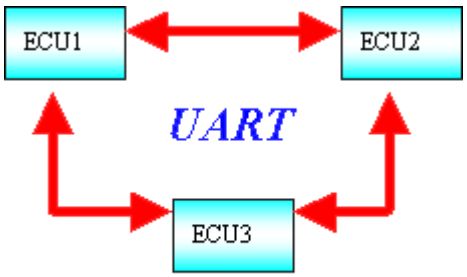
임베디드 시스템 (또는 마이크로 컨트롤러) 에서 일반적으로 사용되는 CAN 버스는 마이크로 컨트롤러 (마이컴) 사이에서 통신망을 형성하며, 2 가닥의 꼬임선 (Twist Pair Wire) 으로 연결되어 반이중 통신 (Half Duplex) 방식으로 짧은 메시지를 사용하는 고속 응용 시스템에 적당합니다. 더불어 외부의 요인 (노이즈 등) 등에 강인성을 가져 통신 에러율을 최소화 하여 높은 신뢰성을 가지고 있습니다. 이론적으로는 2032 개의 서로 다른 디바이스 (임베디드 컨트롤러) 를 하나의 네트웍 상에 연결하여 통신을 수행할 수 있으나 CAN 트랜시버 (송신기) 의 한계로 인하여 110 개까지의 Node (통신 주체) 를 연결하여 사용할 수 있습니다. (필립스 트랜시버 82C250 의 경우)

통신 속도는 실시간 제어가 가능한 1Mbps (ISO 11898 규격) 의 고속 통신을 제공하며 더불어 자동차 환경 (자동차 엔진룸의 경우 다양하고 심각한 전기적인 노이즈 상존) 과 같은 심각한 노이즈 환경에 적합하도록 에러 검출 및 에러 보정의 기능이 있습니다.

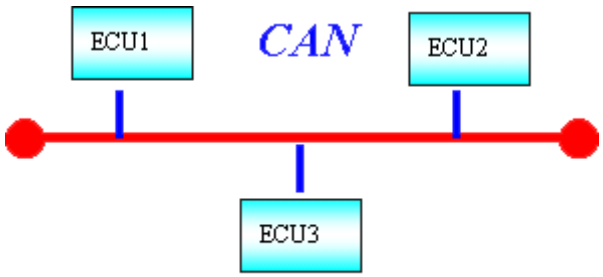
연혁

1986 년 자동차 내의 서로 다른 세 개의 전자장치 (ECU-Electronic Control Unit) 간의 통신을 위한 통신 장치 개발을 자동차 업체인 벤츠의 요구에 의하여 자동차 부품 업체인 독일의 보쉬에 의해 최초로 개발되었습니다.

초기에는 UART 방식을 고려하였으나 일대일 (Point To Point) 통신 방식이기에 서로 다른 세 개의 ECU 간의 통신 방식으로는 적합하지 않아 다중통신 (Multi Master Communication) 방식이 필요하게 되어 CAN 이 탄생하게 되었습니다. 그 후 최초의 집적화된 CAN 부품은 1987 년 인텔에 의해 생산 되었습니다.



(Pont To Point 방식)



(Multi Master 방식)

규격

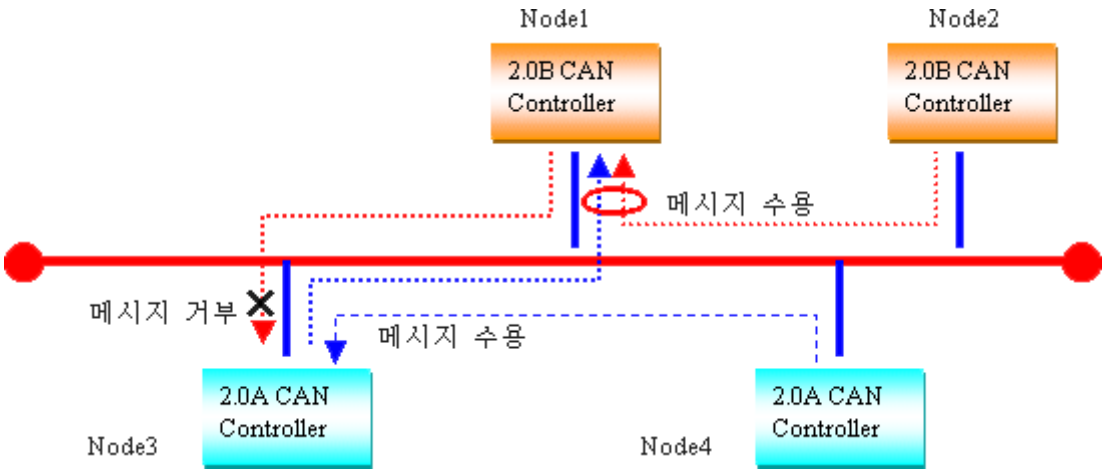
CAN 메시지에 있는 식별자 (ID) 의 길이에 따라 두 가지 모드로 구분됩니다.

- 표준 CAN (버전 2.0A) : 11 비트 식별자
- 확장 CAN (버전 2.0B) : 29 비트 식별자

ISO 규격에 따라 두 가지로 구분되며 이 경우에는 물리계층에서 차이가 있습니다.

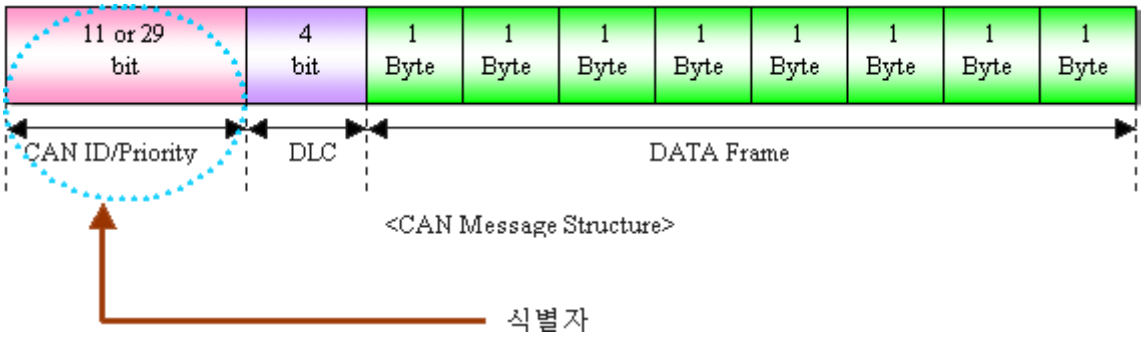
- ISO 11898 : 1Mbps 이상의 고속 통신 가능
- ISO 11519 : 125Kbps 까지의 통신 가능

대부분의 CAN 2.0A Controller 는 오직 표준 CAN 포맷 방식의 메시지만 전송 및 수신이 가능하며 확장 CAN 포맷 방식 (CAN 2.0B)의 메시지를 수신 하더라도 그 데이터를 무시해 버립니다. 즉, CAN 2.0A Controller 에서 보내온 메시지 데이터만 유효합니다. 그러나 CAN 2.0B Controller 는 양쪽 모두의 메시지 포맷을 송수신 가능합니다.



동작원리

- CAN 은 다중통신망 (Multi Master Network) 이며 CSMA/CD+AMP (Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority) 방식을 이용합니다.
- 먼저 CAN Node 에 메시지를 보내기 전에 CAN 버스라인이 사용 중 인지를 파악합니다.
- 또한 메시지간 충돌 검출을 수행합니다. 이러한 방식은 이더넷 통신 방식과 유사합니다.
- 어떠한 Node (시스템)로부터 보내어진 데이터 메시지는 송신측이나 수신측의 주소를 포함하지 않습니다.
- 대신에 각 노드의 데이터 메시지 항목에 CAN 네트웍상에서 각각의 노드 (시스템)를 식별할 수 있도록 각 노드 (시스템) 마다 유일한 식별자 (ID-11bit 또는 29bit)를 가지고 있습니다.



- 네트웍상에 연결된 모든 노드 (CAN Controller 시스템)는 네트웍상에 있는 메시지를 수신한 후 자신에게 필요한 메시지인지를 식별자를 통하여 평가한 후 자신이 필요로 하는 식별자의 메시지인 경우만 취하고 그렇지 않은 경우의 메시지는 무시합니다.
- 네트웍상 (CAN 통신 라인)에 흘러 다니는 여러 노드의 데이터들이 동시에 사용자가 필요로 하는 노드로 유입되는 경우에 식별자의 숫자를 비교하여 먼저 취할 메시지의 우선순위를 정합니다. 식별자의 숫자가 낮은 경우가 우선순위가 가장 높습니다. (식별자가 1 인 경우가 10 인 경우보다 우선순위가 높음)

- 우선순위가 높은 메시지가 CAN 버스의 사용 권한을 보장 받으며 이때 낮은 순위의 메시지는 자동적으로 다음 버스 사이클에 재전송을 수행합니다. 이때 까지도 높은 우선순위를 가진 메시지가 완료되지 않은 상태이면 전송을 완료할 때까지 대기하고 있습니다.
- 각 CAN 메시지는 11 비트의 식별자 (CAN 2.0A), 또는 29 비트의 식별자 (CAN 2.0B)를 가지면 CAN 메시지의 맨 처음 시작부분에 위치합니다.
- 더불어 식별자는 메시지의 형태를 식별시켜 주는 역할과 메시지의 우선 순위를 부여하는 역할을 합니다.

메시지 구조

개요

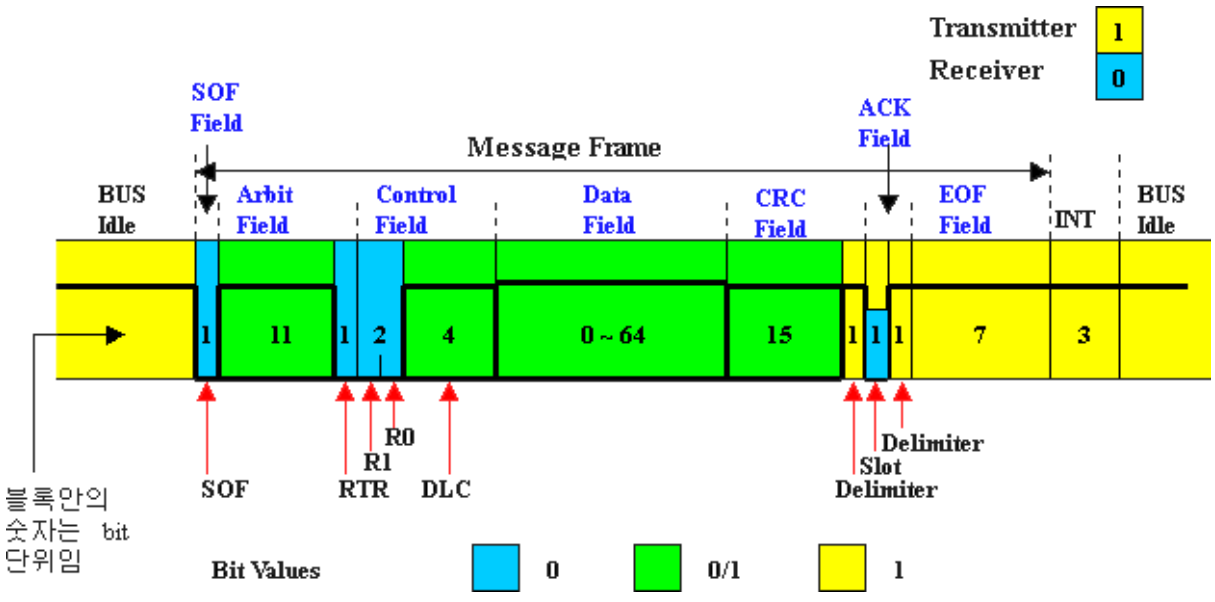
CAN 시스템 (통신)에서 데이터는 메시지 프레임을 사용하여 송수신이 이루어집니다. 메시지 프레임은 하나 또는 그 이상의 송신 노드로 부터 데이터를 수신노드로 운반합니다. CAN Protocol (통신 규약)은 다음과 같은 두 가지 형태의 메시지 프레임을 지원합니다.

- 표준 CAN (버전 2.0A)
- 확장 CAN (버전 2.0B)

대부분의 2.0A CAN Controller 는 표준 CAN 방식을 사용하나 2.0B CAN Controller 는 표준 또는 확장 방식 모두를 사용하여 데이터의 송수신을 행할 수 있습니다.

표준 CAN 메시지 구조 (2.0A)

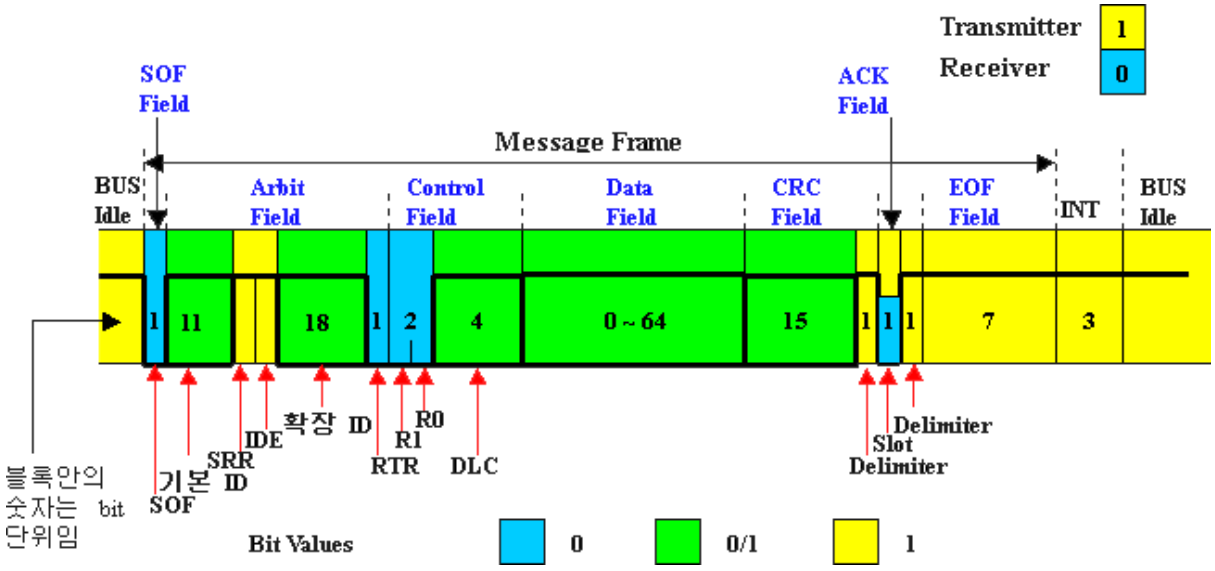
- 7 개의 서로 다른 필드로 구성됩니다. (그림 참조)



- 프레임의 시작 (SOF : Start Of Frame) 필드 ; 메시지 프레임의 시작을 표시하며 메시지 프레임의 최우선에 위치하며 디폴트 "0" 값을 가집니다.
- 중재 필드 (Arbitration Field) ; 11 비트의 식별자와 원격 전송 요구 (RTR) 비트를 가지며 디폴트 "0"을 가지는 RTR 비트는 비트값이 "0" 일 때 CAN 메시지가 데이터 프레임이라는 것을 가리킵니다. 역으로 RTR 비트 값이 "1"이면 CAN 메시지가 원격전송요청 (RTR : Remote Transmission Request)을 의미합니다. 다시 말해 CAN 메시지가 데이터 프레임이 아닌 원격프레임 (Remote Frame) 상태임을 나타냅니다. 원격 프레임은 데이터 버스상의 어떤 한 노드로부터 다른 노드로 데이터를 전송 요청할 때 사용 됩니다. 데이터를 보내기 전에 사용되는 메시지 프레임이기에 데이터 필드를 포함하지 않습니다.
- 제어 필드 (Control Field) ; 6 비트로 구성되며 향후에 사용되기 위해 예약된 두 개의 "0"의 값을 가지는 R0, R1 과 데이터 필드의 바이트 수를 가리키는 4 비트의 데이터 길이 코드 (DLC : Data Length Code)로 구성됩니다.
- 데이터 필드 (Data Field) ; 한 노드로부터 다른 노드로 전하고자 하는 데이터를 포함하며 0~8 바이트로 구성됩니다.
- CRC 필드 (CRC : Cyclic Redundancy Check) ; 17 비트의 주기적 중복확인 (CRC) 코드를 가지며 데이터필드의 끝을 알리는 "1"의 값을 가지는 비트가 이어집니다.
- ACK 필드 (ACKnowledge Field) ; 2 비트로 구성되며 첫 번째 비트는 "0" 값을 가지는 Slot 비트입니다. 그러나 메시지를 성공적으로 수신한 다른 노드로부터 전송된 "1"의 값으로 기록될 수 있습니다. 두 번째 비트는 "1"의 값을 가집니다.
- 프레임 종료 필드 (EOF : End Of Frame Filed) ; 7 비트로 구성되며 모두 "1"의 값을 가집니다. EFO 뒤이어 모두 "1"의 값을 가진 3 비트의 프레임 중단 필드 (INTermission Field)가 이어집니다. 3 비트의 INT 주기 이후에 CAN 버스라인은 자유상태로 인식됩니다. 이후 버스 Idle Time 은 "0"을 포함한 임의의 길이입니다.

확장 CAN 메시지 구조 (2.0B)

- 2.0A 와 구분되기 위해 29 비트의 메시지 프레임 식별자를 가집니다.
- 기존에 사용중인 2.0A 와 호환되는 2.0B 로 발전되었습니다.
- 2.0A 와 차이점은 중재 필드가 두 개의 CAN 메시지 식별자로 구분되어 포함되며 첫 번째 (기본 ID)는 11 비트 길이로 2.0A 와 호환되게 하며 두 번째 필드 (확장 ID)는 18 비트 길이로 ID 는 총 29 비트로 구성됩니다. 두 개의 ID 필드 사이에 ID 확장자 (IDE : IDentifier Extension)가 있어 두 개의 ID 필드를 구분합니다. (그림 참조)
- SRR (Substitute Remote Request) 비트는 중재 필드에 속해 있으며 표준 데이터 프레임과 확장 데이터 프레임을 중재해야 하는 경우에 대비하기 위해 항상 "1"의 값을 전송합니다. 만약 표준 데이터 프레임과 확장 데이터 프레임이 같은 기본 ID (11 비트)를 가지고 있으면 표준 데이터 프레임이 우선 순위를 가집니다. 그리고 2.0B 메시지 프레임에 있는 다른 필드들은 표준 메시지 포맷으로 식별됩니다.



CAN 2.0A 와 2.0B 의 호환성

- 2.0B Controller 입장에서는 2.0A 와 송수신에 있어 완벽하게 호환이 됩니다.
- 2.0A Controller 는 두 가지 경우가 있습니다. 1) 첫째는 2.0A Format 의 메시지만 송수신이 가능한 경우이며 2.0B 의 메시지는 에러를 발생시킵니다. 2) 두 번째는 2.0B Passive 로 알려져 있으며 2.0A 의 메시지는 송수신 가능하고 더불어 2.0B 의 메시지는 식별을 하여 무시해버립니다. 그 결과 2.0A 와 2.0B 는 하나의 CAN 네트웍상에서 함께 사용할 수 있게 됩니다.
- 사용자가 사용할 수 있는 CAN ID 는 2.0A 는 2,032 개이고 2.0B 는 5 억 개를 초과 사용할 수 있습니다.