CAN protocol의 이해

자동차 기술적 요구에 부응하는 시리얼 통신 시스템의 필요성에 따라 개발된 CAN은 1986년 SAE에서 보쉬사가 제안하였고 1992년 벤츠자동차에 최초로 적용되어 출시되면서부터 현재는 사실상 표준에 해당할 정도 대부분의 자동차 제조사에서 채택하고 있다.

ID	RTR	SRR	IDE	R0	R1	DLC	Data bytes	CRC Sequence	CRC Delimiter	ACK Slot	ACK Delimiter	Error	Stuffed Bits	Baud Rate	Min Voltage	Max Voltage
329	1	-	1	1	-	8	80 5E 80 0E 12 32 00 10	4A3D	0	Yes	0	-	5	500kbps	2.456V	3.464V

그림 1) CAN 2.0A의 데이터 프레임

□ Protocol의 개요

CAN의 Protocol은 그림2와 같이 어떤 회의 진행과정을 보면 이해하기 쉽다. 각각의 분

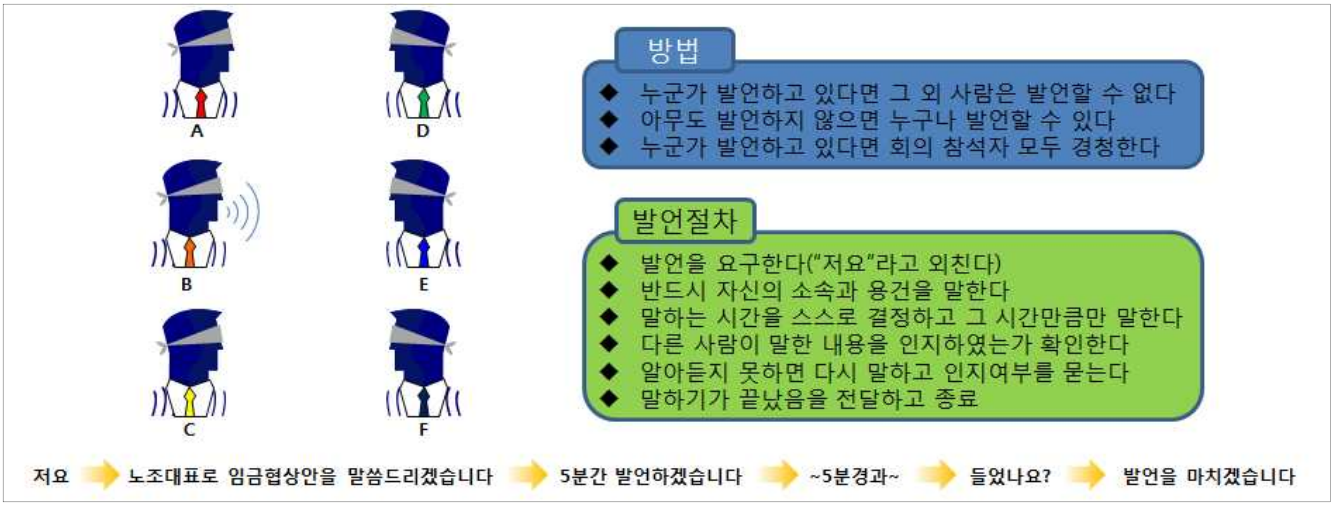

그림 2) 회의 방법과 절차

야를 담당하고 있는 6사람이 모여 협업을 위한 회의를 진행하고자 한다. 다만 회의 참석자들은 누가 참석하는지 모르고 있으며 회의 진행 시 상대방을 볼 수 없도록 안대로 눈을 가리고 있다. 눈이 보이지 않는 관계로 오로지 말하고 듣는 것만으로 회의를 진행해야만 한다. 여기서 말을 하는 입은 송신에 해당하고 듣고 있는 귀는 수신부, 공기를 매질로 음파로서 상대방에게 정보를 전달하기 때문에 공기는 전송선로에 해당한다 할 수 있다. 구강으로서 적당한 정보를 송신하고자 하는 경우 뇌에서는 자신이 한 말에 대하여 귀를 통해 정확하게 출력되었는지 피드백 제어할 것이며 질문이나 말하여야 할 순서 등을 정리해 두기 때문에 스스로의 잘못됨을 판단할 수 있다. 한편 듣고 있는 측에서는 상대편의 말을 경청하고 있기 때문에 듣는 순간 상대편의 말의 순서나 논리 등 잘못된 부분을 판단할 수 있으며 잘못됨이 발견된 즉시 잘못되었다고 말할 수 있다. 간단한 내용이지만 CAN의 프로토콜을 쉽게 이해하는데 큰 도움을 주는 내용이다. 지금까지의 얘기는 네트워크에 참여하는 각 노드(node)가 기본적으로 갖추고 있는 기능에 해당한다.

회의 방법은 누군가 발언할 경우 회의 참석자 전원은 모두 경청해야하는 브로드캐스트(broadcast) 방식이며 K-line이나 LIN(Local Interconnect Network)의 경우는 마스터가 하나이지만 CAN에서는 아무도 발언하지 않는 경우 누구나 발언할 수 있는 멀티 마스터(multi-master) 방식이다. 멀티 캐스트 네트워크는 모든 노드가 동일한 메시지를 수신하고 그 메시지에 대해 각각의 노드가 조치를 취하는 방식으로 노드 자신이 처리해야할 정보만을 필터링하여 조치하는 기능을 가진다. 한편 회의에서는 대부분 누군가의 발언 내용을 계기로 발언하는 경우가 많다. CAN에서도 마찬가지며 누군가의 요청에 응답하는 등의 이벤트 트리거(event trigger) 방식을 취하고 있다.

휴식 상태에서 누군가 발언을 하고자 하는 경우 그림2와 같이 우선 '저요 또는 제가 말하겠습니다'라고 외친 후 용건의 내용에 따라 발언권의 기회를 획득할 수 있기 때문에 자신의 소속과 용건을 말한다. 회의 참석자들간 발언권 경쟁이 없는 경우 발언 시간을 스스로 결정하고 참석자들에게 알린다. 스스로 정한 발언시간만큼 발언 후 발언 내용을 정확하게 인식하였는지에 대한 회신을 기다린다. 만약 인식에 대한 회신이 없는 경우 다시 말한다. 모두 알아들었다고 말하면 발언을 마친다고 말하고 발언을 종료한다. 이 때 발언은 표준어를 사용해야하며 말하는 속도 또한 일정해야만 한다.

마찬가지로 CAN에서도 일정한 규칙을 가지고 있으며 발언절차에서와 같이 '저요 또는 제가 말하겠습니다'라고 외치듯 어떤 노드가 송신을 개시하고자 하는 경우 CAN에서는 휴식상태에서 하나의 우성(dominant)신호인 SOF(Start Of Frame)를 송신함으로서 메시지 전송을 개시한다. 전송이 시작되면 소속과 말하고자하는 용건에 해당하는 ID와 발언 시간에 해당하는 DLC(Data Length Code)를 차례로 전송한 후 실제 Data를 전송한다. 한편 전송된 데이터가 정상적인 신호인지 여부를 판단하고 수신 노드가 메시지를 정상적으로 인식하였는지를

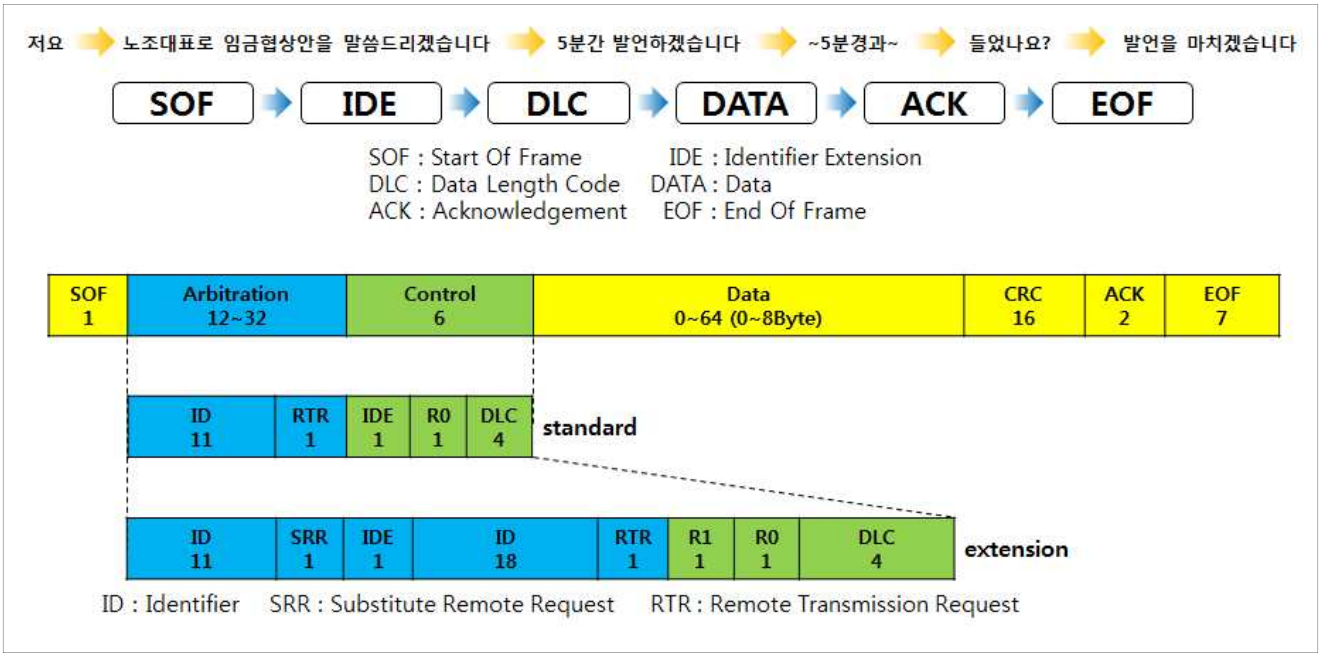

그림 3) Frame의 구조

판단할 수 있는 CRC(Cyclic Redundancy Check : 순환반복검사) 코드 추가로 전송함으로서 정상적인 송수신이 이루어졌는지를 묻는다. 수신 노드는 수신된 데이터를 CRC 코드로서 검증하게 되며 정상적인 신호이고 정확하게 인식하였다면 ACK(Acknowledgement : 인식) 신호를 버스에 추가한다. ACK를 수신한 송신 노드는 데이터 전송이 정상적으로 이루어진 것으로 판단하고 데이터 전송을 마친다는 의미의 EOF(End Of Frame)를 보냄으로서 데이터 전송을 마친다.

□ Frame의 구조

○ SOF

데이터를 전송하고자하는 노드는 우선 하나의 우성(dominant)신호인 SOF(Start Of Frame)를 송신함으로서 메시지 전송을 개시한다. SOF는 단일 bit로 구성되어져 있으며 이

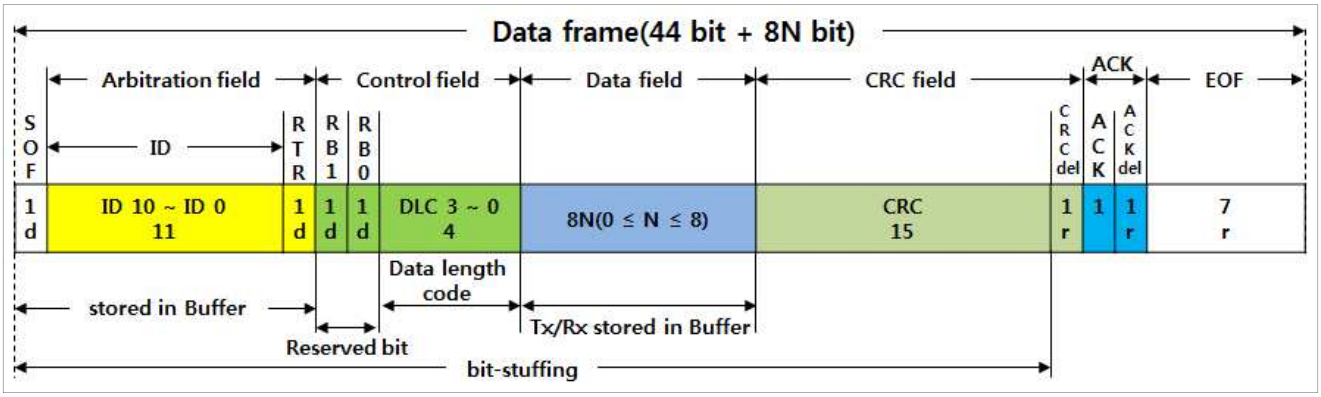

그림 4) CAN 2.0A의 데이터 프레임 구조

신호에 의해 모든 노드는 싱크 되어 수신 및 응답 준비가 되어야만 한다.

○ ID(Identifier) field

ID(식별자)는 대부분 수신 컨트롤러인 노드에 해당하지만 CAN에서의 ID는 메시지를 의미

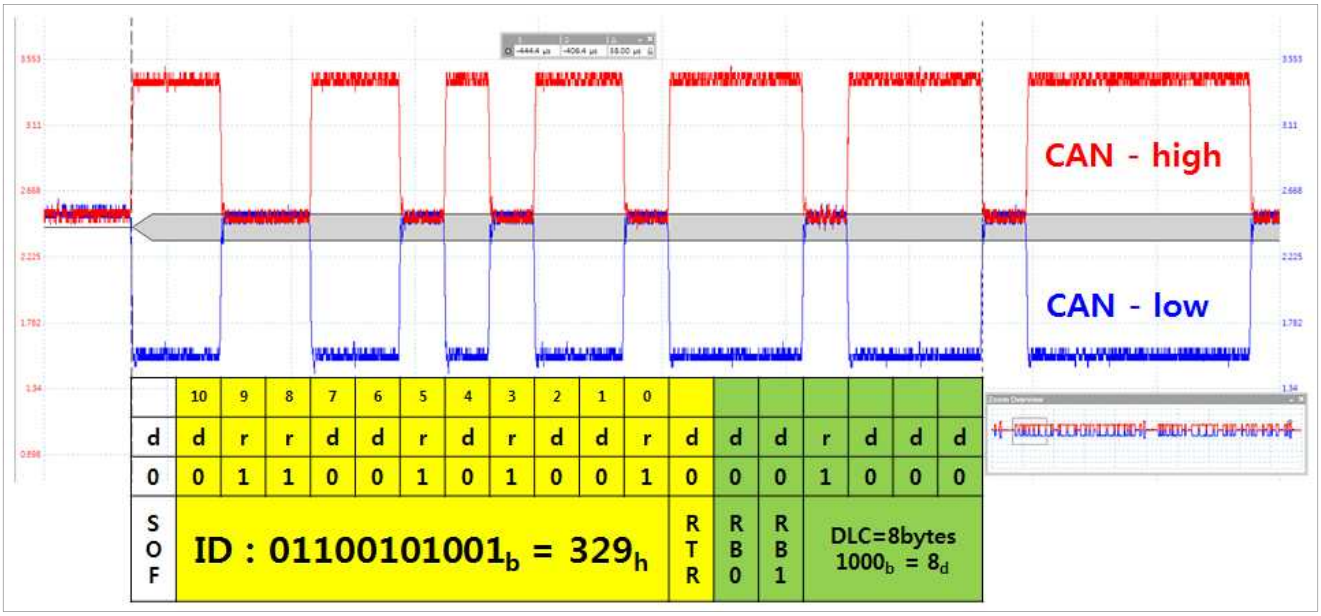

그림 5) ID field와 control field

한다. 즉 데이터 수신 제어기를 의미하는 것이 아니고 데이터의 성격을 의미한다.

CAN 2.0A에서 ID는 그림3에서와 같이 11bit로 구성되어져 있으나 2.0B에서는 29bit로 구성되어져 있다. 그림5에서 ID에 해당하는 2진화 부호 11bit 영역을 16진법으로 환산하면 329_h가 되어 표1에서 보이는바와 같이 엔진 컨트롤러가 TCU나 VDC 등에 보내는 신호에 해당된다. 엔진 ECU가 변속기나 제동장치의 제어기에 보내는 신호를 살펴보면 rpm이나 APS(또는 TPS), 엔진의 부하량 정도일 것이다. 참고로 그림5는 IG key on 상태에서 APS를 가감속할 때의 데이터 중 일부분을 확대한 것이다. 즉 ID는 데이터를 수신하는 제어기가 아닌 데이터 자체를 의미한다는 뜻이다.

massage	ID(hex)	Transmit units	Receiving units
EMS 1	0316	EMS	ABS/TCS, TCU
EMS 2	0329	EMS	ABS/TCS, TCU
EMS 4	0545	EMS	ABS/TCS, TCU
TCS 1	0153	ABS/TCS	EMS, TCU
TCS 2	01F0	ABS/TCS	EMS, TCU
TCU 1	043F	TCU	ABS/TCS, EMS

표 1) 대표 ID

그림2의 회의장에서 만약 B와 E가 동시에 발언을 하겠다고 외쳤다고 가정했을 때는 둘 중 한사람만 발언할 수 있도록 중재가 필요하다. E는 '점심식사' 거리에 대해서 말하려하고 B는 '임금협상안'에 대하여 말하려고 한다고 가정한다면 점심식사보다는 임금협상안이 더 중요 사안에 해당하므로 대부분 발언권은 B에게 주려할 것이다.

같은 방법으로 두 개 이상의 노드가 버스에 접근할 경우 충돌을 해결하기 위해 CAN에서는 ID에 따라 버스에 접근할 수 있는 우선순위를 두고 있다. 송신기는 송신된 비트의 레벨

과 버스를 모니터링 한 레벨을 비교할 수 있기 때문인데 ID의 각 bit에서 우성이 맨 앞에 나타난 것이 우선권을 갖는다. 우성은 2진 부호로 '0'이기 때문에 결과적으로 ID 숫자가 낮을수록 우선권이 있다는 얘기이다. 버스 레벨이 같은 경우 송신을 계속하지만 송신 레벨과 모니터링 레벨이 다르면 즉시 송신을 중지한다.

그림6과 같이 노드1과 2가 동시에 송신을 하겠다고 SOF를 송신한 경우 첫 번째에서부터 다섯 번째 비트까지는 서로 같은 레벨을 보이기 때문에 같이 전송하지만 그림에서 5번 bit

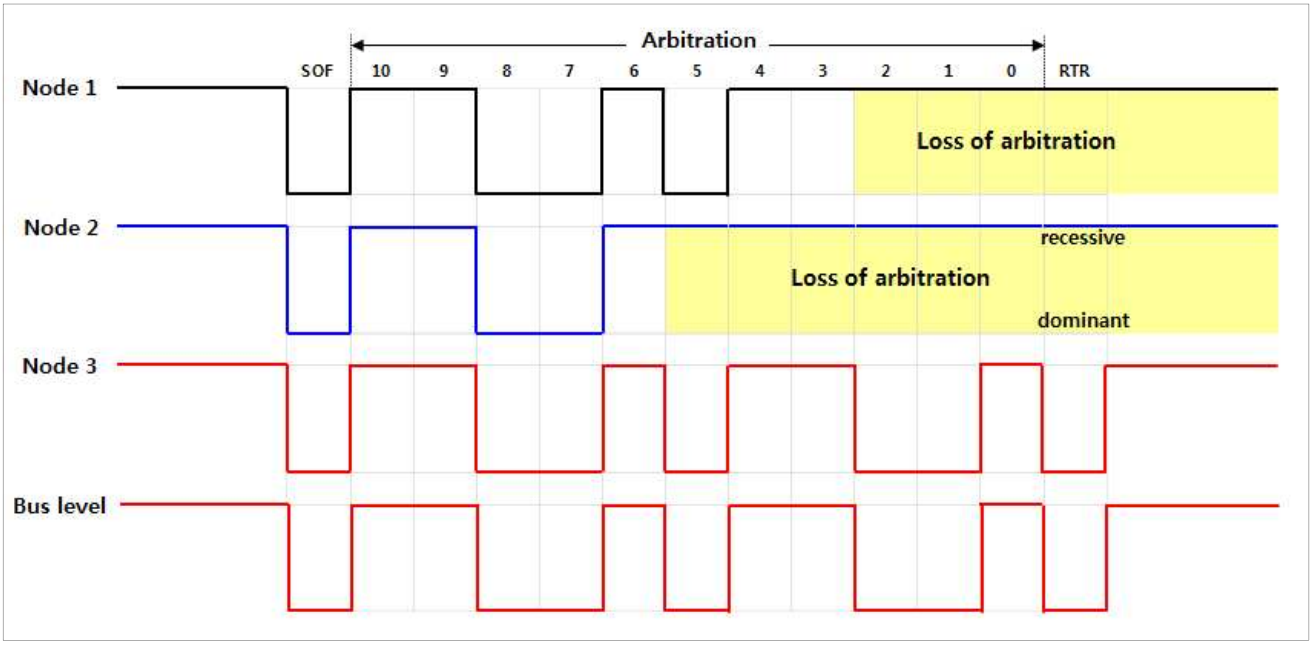

그림 6) 우선권의 조정 방식

인 여섯 번째 비트에서 노드1은 우성이지만 노드2는 열성이므로 노드2는 송신을 중단하고 수신만 한다. 한편 노드3도 동시에 송신을 개시했다고 가정하면 그림6의 2번 bit에서 노드3이 우성이므로 노드1은 송신을 중단한다. 결국 최종적으로 노드3만 CAN 버스에 신호를 계속하여 전송할 수 있게 된다. ID의 bit 단위 중재 원칙에 따라 송신하려고하는 레벨은 열성(recessive : 1)인데 상대측에서 우성(dominant : 0)인 경우 송신을 중단하는 방식으로 조정한다. SOF 이후 우성신호가 나타날 경우 2진법도 마찬가지이지만 16진법으로 숫자가 낮아지므로 ID 번호가 낮을수록 우선권을 갖는다. 때문에 시스템 운용에 필수적인 데이터는 ID 숫자가 낮다. 즉 표1에서와 같이 ECU가 전송하는 ID 329_h와 TCS가 전송하는 ID 153_h이 충돌할 경우 153_h이 우선한다는 뜻이다.

ID 필드 중 마지막 비트는 RTR(Remte Transmission Request)로 원격 요청신호이다. 이 비트가 열성(recessive : 1)인 경우 ID에 해당하는 신호를 요청하는 메시지라는 뜻이다.

○ Control field

컨트롤 필드는 6bit로 구성되어져 있으며 첫 번째와 두 번째 비트는 확장될 것을 위해 추

가로 준비해둔 것이다. 나머지 4개의 비트는 DLC(Data Length Code)로 이 코드 뒤에 올 데이터 바이트(bytes)의 수량을 표시한다.

Number of Data Bytes		Data Length Code			
		DLC 3	DLC 2	DLC 1	DLC 0
0	0 0 0 0	d	d	d	d
1	0 0 0 1	d	d	d	r
2	0 0 1 0	d	d	r	d
3	0 0 1 1	d	d	r	r
4	0 1 0 0	d	r	d	d
5	0 1 0 1	d	r	d	r
6	0 1 1 0	d	r	r	d
7	0 1 1 1	d	r	r	r
8	1 0 0 0	r	d	d	d

표 2) 컨트롤 필드의 DLC 코드

DLC 코드는 표2와 같은 2진 부호화 코드를 10진법 또는 16진법으로 변환한 값이 바이트 수에 해당하며 그림5에서 보이는 1000_b의 코드는 10진수로 8을 의미하므로 컨트롤 필드 이후 나타나는 데이터는 8bytes가 된다는 의미이다.

만약 컨트롤 필드 뒤에 오는 데이터 바이트가 16개가 넘는 경우 DLC 4bit로는 표현이 불가하여 확장비트를 사용하게 될 것이다. 데이터 바이트 수가 17개인 경우 2진 부호로 10001되어 DLC는 5bit가 요구된다.

○ Data field

데이터 필드는 8bit의 데이터가 전송되며 DLC에 나타난 길이만큼의 데이터 바이트가 전송된다.

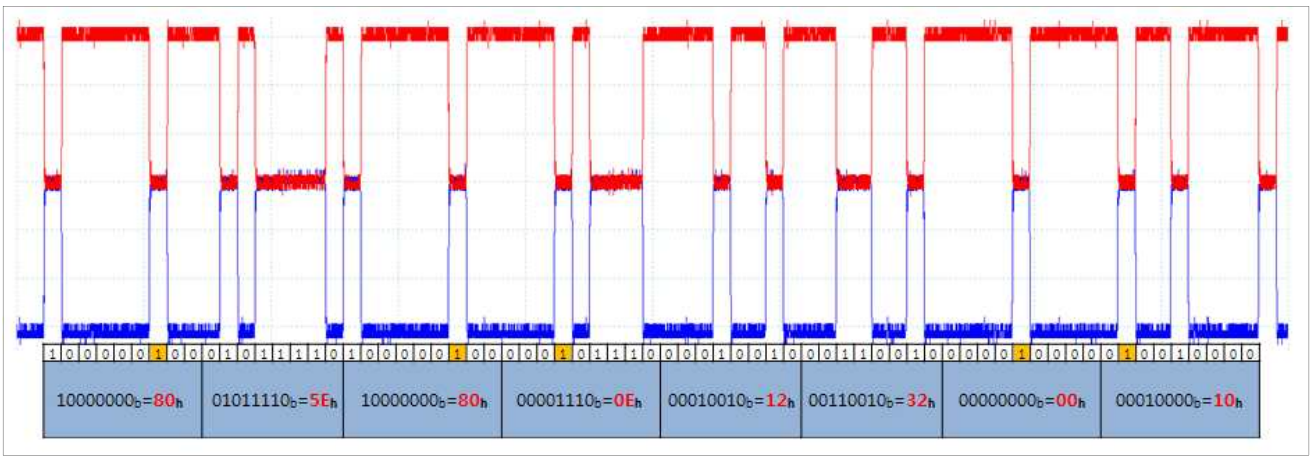

그림 7) Data field

CAN에 참여하는 노드들은 상대편의 송신이 잘못을 판단할 수 있을 뿐만 아니라 스스로를 진단하여 수신만 할지 아니면 버스에서 이탈할지를 결정한다. 이때 자신의 상태를 알리는 에러 프레임을 전송하며 에러 신호는 상태에 따라 연속한 6개 이상의 우성 또는 열성 신호를 송신한다. 결국 똑가은 값을 가진 6개 이상의 비트가 전송될 경우 수신 노드는 이상이 있다고 판단하는 오류가 발생한다. 이런 이유 때문에 데이터 프레임에서는 한 가지 극성만으로 6개 이상의 신호를 전송하지 않는 규칙을 가지고 있다. 즉 송신자가 전송되는 비트 흐름에 있어 우성 또는 열성 비트 신호가 5개 연속으로 같은 값을 가질 경우 자동적으로 전송중인 비트 흐름에 반대 극성의 비트를 채워 넣는다. 이것을 스터핑(Stuffing : 끼워넣기)

이라 한다.

스터핑의 방법은 똑같은 값의 5개 비트 이후 이 연속적인 같은 값의 리듬을 끊어주기 위해 인위적으로 반대 값을 가진 비트를 채워 넣는다. 전송해야할 데이터가 그림8의 맨 위 그림과 같이 3번 ~ 8번까지는 우성, 15번 ~ 20번까지 열성으로 같은 극성의 신호가 6개일

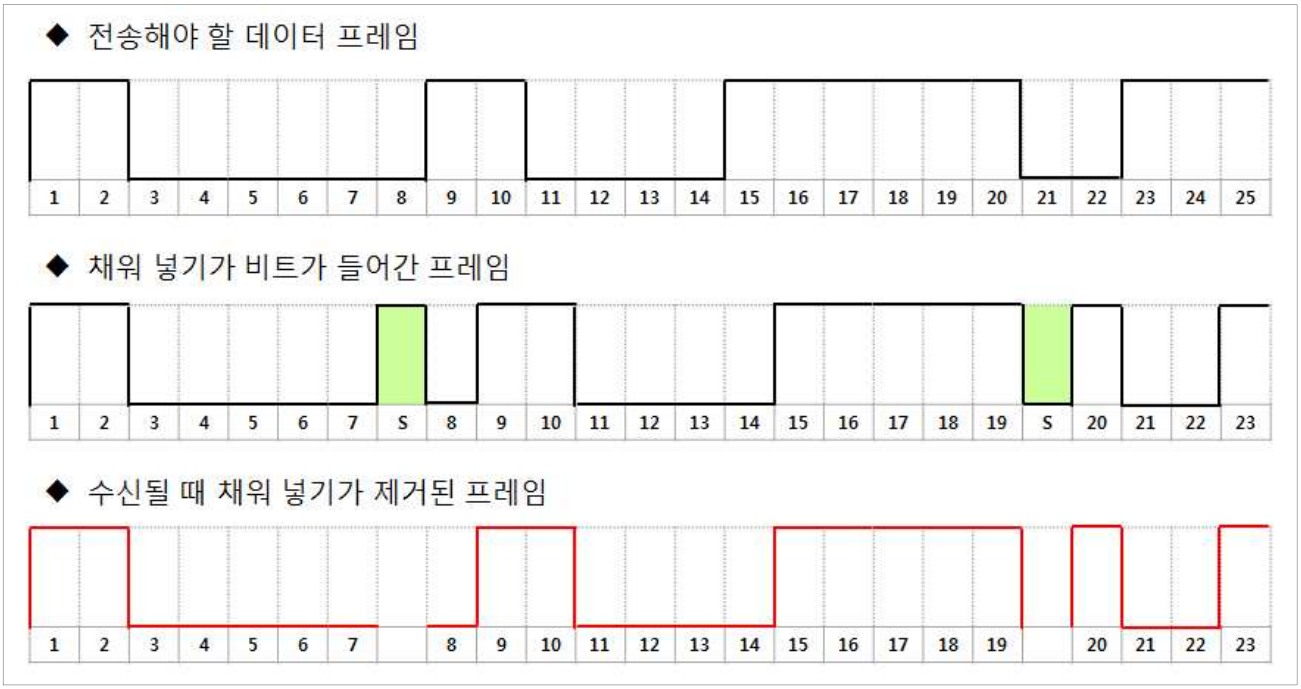

그림 8) stuffing 규칙

경우 자동으로 같은 극성 5개 이후 하나의 다른 극성을 채워 버스에 전송한다. 수신 노드측에서는 수신 데이터 중 같은 극성 5개 이후 나타나는 비트를 스터핑 비트로 판단하고 버린 후 데이터를 판단한다. 한편 수신된 신호가 스터핑 규칙이 잘못된 경우 수신 노드는 송신측의 스터핑 에러를 판단하고 에러 신호를 출력한다. 스터핑 규칙은 데이터 프레임과 원격 프레임 외 다른 프레임에는 비트들의 구조가 고정되어 있어 이 규칙을 따르지 않는다.

그림7에서 첫 번째 바이트 영역에서 원래신호 8비트와 스터핑 비트를 포함한 100000100_b의 9개 비트 신호가 전송되었음을 알 수 있고 수신노드는 해당 신호 중 스터핑 비트를 무시한 나머지 8비트 10000000_b의 신호를 진짜 신호로 판단한다. 이 신호는 다시 16진수로 변환하면 80_h가 된다. 이런 방법으로 8개 바이트 모두 계산하면 그림과 같이 각각 80_h, $5E_h$, 80_h, $0E_h$, 12_h, 32_h, 00_h, 10_h이 된다.

○ CRC field

메시지의 유효성 검증을 위한 방법으로 송신측에서 보내고자 하는 원래의 정보 이외에 별도의 잉여분 데이터를 추가하여 전송하면 수신측에서 이 추가된 데이터를 이용해 메시지를 검사함으로서 메시지에 대한 유효성을 검증한다. K-line, LIN 시스템에서는 Parity와

Checksum 등을 이용하고 있으나 CAN에서는 순환중복검사(CRC : Cyclic Redundancy Check)를 이용한다.

송신측에서 자신이 전송한 데이터가 정상적인 신호인지를 판단할 수 있게 CRC코드를 데이터 필드 뒤에 전송하며 이 영역은 CRC 시퀀스 영역과 CRC 구분자(delimiter)로 구성되어져 있다. 전송된 메시지에 대한 유효성의 판단은 수신 노드에 의해 이루어지고 있으며 여기서 특별한 다항식과 나눗셈의 나머지를 계산하는 수학적 연산 방식인 모듈로(modlulo)2를 적용하고 있다. 참고로 모듈로는 어떤 수를 n(divisor, modulus)으로 나누었을 때 그 나머지(residue)를 구하는 연산으로 이 때 몫(quotient)은 관심을 두지 않고 오로지 나머지에만 관심을 두는 방식이다.

그림9와 같이 노드1에서 송신하고 노드2가 수신측이라 했을 때 송신하는 노드1에서 CRC 생성 다항식을 이용하여 1234를 CRC 코드를 생성했다면 데이터 필드 뒤에 1234라는 시퀀

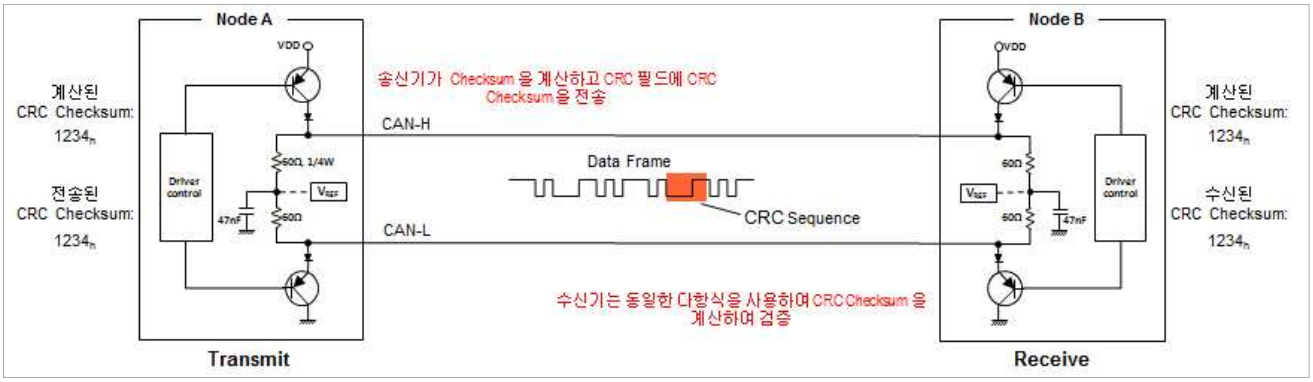

그림 9) CRC를 이용한 메시지의 검증

스 데이터를 추가하여 송신하고 수신측에서는 이 코드를 이용하여 SOF를 포함한 ID필드에서 데이터 필드까지의 메시지에 대한 유효성을 검증한다. 만약 CRC 코드로 검증했을 때 에러가 발생한 경우 전송된 메시지는 유효하지 않은 것으로 판단하고 메시지를 무시하게 된다.

CRC 코드 생성을 위한 생성식은

$$g(X) = X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + X^{0}$$

으로 64bit용 32bit용 등 많은 생성 다항식이 있으나 15bit를 사용하는 CAN에서는 앞에 표시된 다항식을 표준으로 사용하고 있다. 이 생성 다항식을 2진법으로 표시하면 g(X)=1100010110011001이 된다. 생성 다항식 g(X)를 통해 다항식 f(X)를 나눈 후 나머지가 15bit의 CRC가 되며 CRC 필드에 전송된다.

ID가 03_h인 데이터 프레임에서 컨트롤 필드는 01_h, 데이터는 DB_h인 경우 표3에서와 같이 데이터 이후 15개의 '0'을 추가한다. 이 후 그림10과 같이 생성다항식의 이진수로 나누어 최종 나머지를 CRC 코드로 사용한다. 그림에서 생성된 코드는 $F4C_h$로 이진수로는 000111101001100_d이 된다.

ID											RTR	Control						Data								이 지점부터 끝의 CRC까지 15개의 0을 추가															
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

표 3) CRC 코드의 생성과정

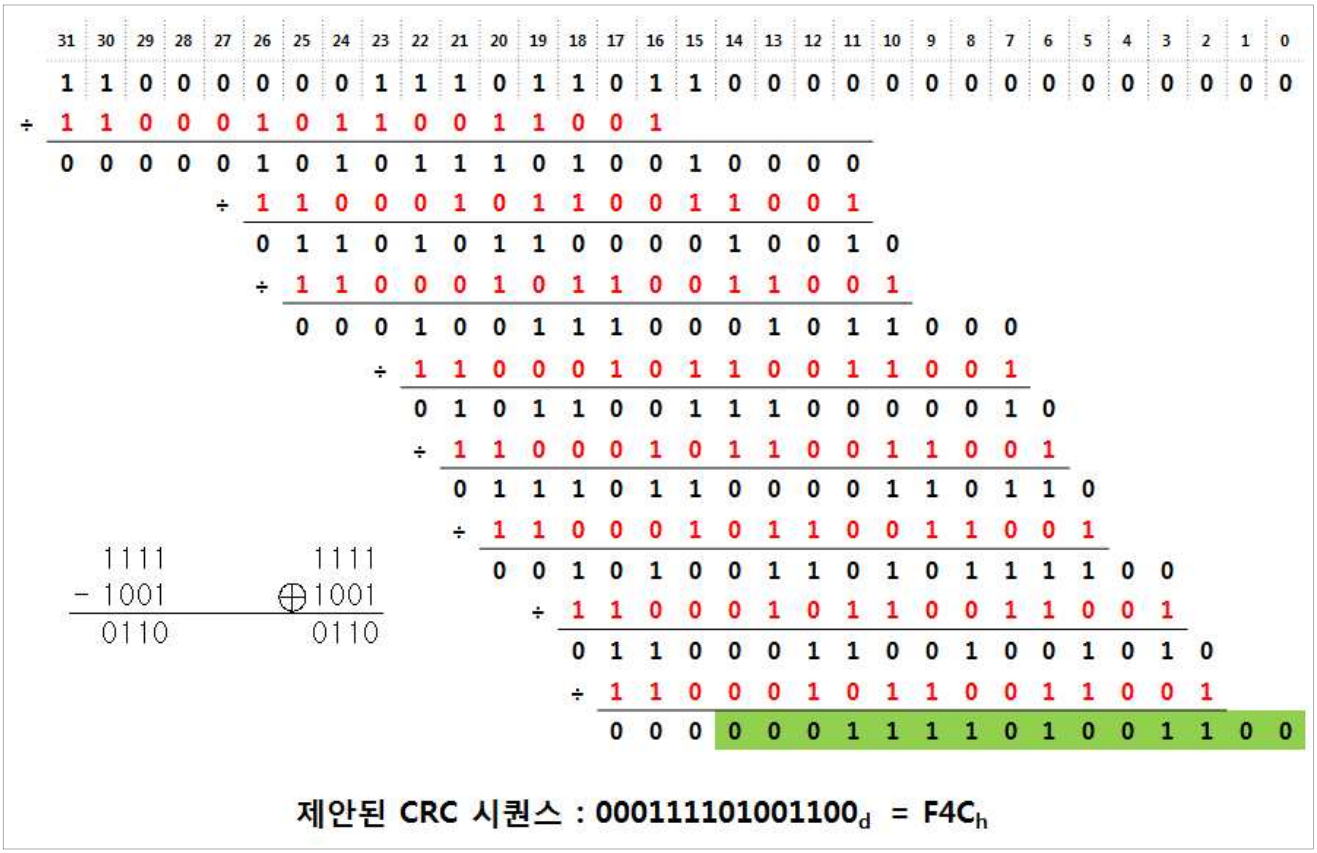

그림 10) 모듈로 2를 이용한 CRC 시퀀스 생성

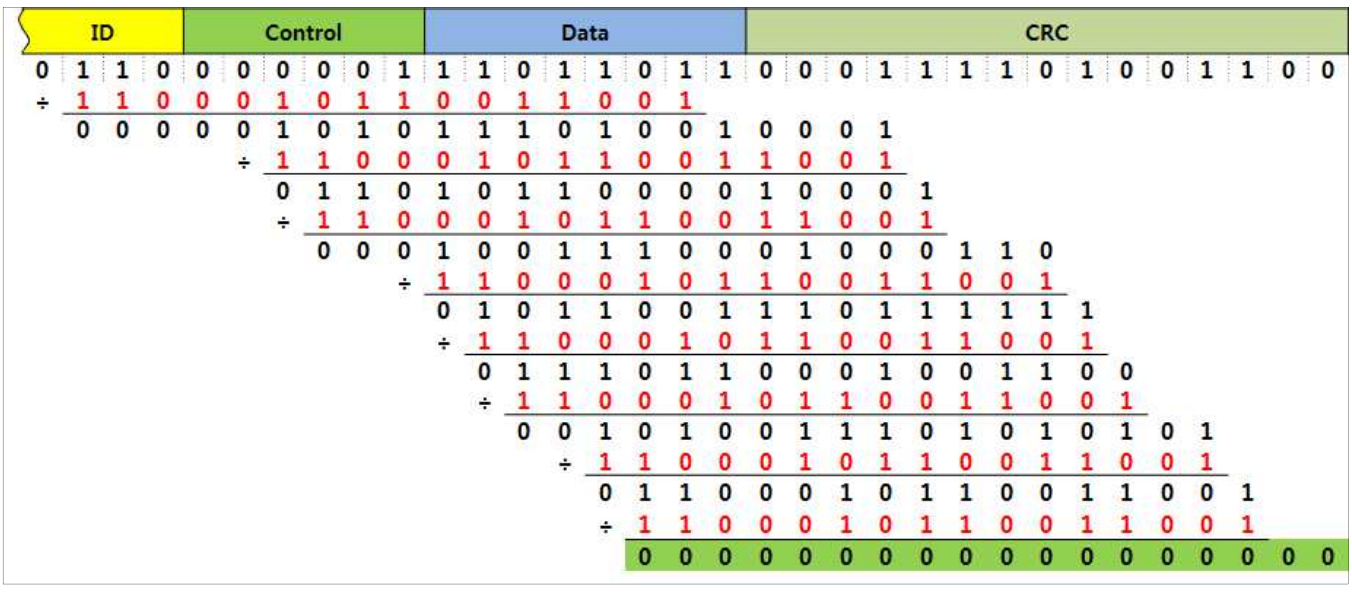

그림 11) 생성 다항식으로 나누었을 때 결과는 0이어야만 한다

생성된 CRC 코드를 데이터 필드 뒤에 적용하여 전송하면 수신측에서는 그림11에서와 같이 유효성을 검증하기 위해 생성 다항식으로 나누었을 때 최종적으로 '0'인 경우만 정상적인 메시지로 판단한다. 만약 '0'이 아닌 경우 전송된 메시지는 무효로 하고 재전송을 요청한다. 지면관계상 단순한 예를 들었지만 실제 데이터를 수작업으로 계산할 경우 최소 40여회 정도 연산해야만하는 번거로움이 있지만 이 방법은 그만큼 최적 조건에 가장 가까운 방법이라고 알려져 있어 CAN에서는 표준으로 사용하고 있다.

그림12는 실제 차량에서 전송되는 데이터를 포착한 것으로 그림1에서의 CRC 영역을 확대

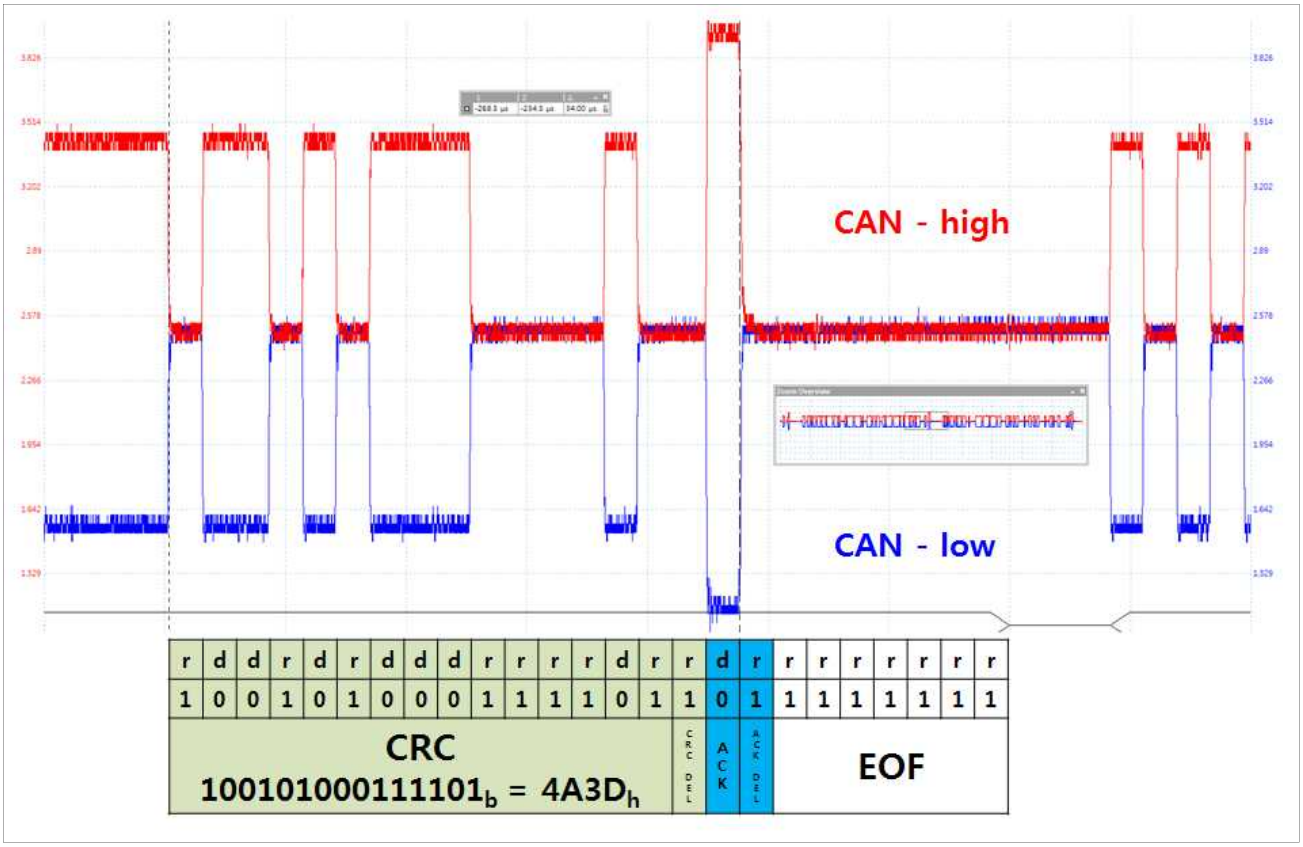

그림 12) CRC와 ACK

한 것이다. CRC 필드의 15개 bit는 CRC 시퀀스가 $4A3D_h$임을 보여주고 있으며 마지막 16번째는 CRC 구분자이다.

○ Acknowledgement field

ACK 필드는 그림12에서와 같이 ACK bit와 ACK 구분자로 구성되어진다. ACK은 메시지를 수신함으로서 확인 응답하는 것으로 송신노드가 아닌 수신노드가 응답해주는 것이다. 만약 어떤 노드가 메시지를 수신하지 못했다면 오류 프레임이 대신 생성된다. 이것은 송신측 CRC 시퀀스를 통해 '알아들었습니까?'라는 물음에 응답하는 형태라 할 수 있다. 한편 어떤 메시지를 여러 노드가 수신한다 했을 때 하나의 노드만 수신하지 못했다면 해당 노드의 에러일 확률이 크다. 반면 해당 메시지를 수신해야할 모든 노드가 메시지를 수신하지 못했다

면 송신측 노드가 불량이 된다.

ACK의 특징은 그림12에서 보이는 바와 같이 하나의 bit로 구성되어져 있으며 다른 bit에 비해 진폭이 큰 특징을 가지기 때문에 파형 분석 시 쉽게 구분할 수 있다.

수신노드는 수신한 메시지에 대하여 CRC를 통해 검증했을 때 이상이 없는 경우 ACK 필드에 1개의 우성(dominant : 0)비트를 CAN 버스의 데이터 프레임에 추가하고 그렇지 않은 경우는 열성(recessive : 1)을 추가한다. 즉 정상적으로 수신한 경우 ACK 필드의 신호는 구분자를 포함(ACK + ACK 구분자)하여 '01'이지만 수신이 불량한 경우 '11'이 된다.

○ EOF

EOF는 비트 채워 넣기 길이보다 2bit가 더 긴 7개의 열성 bit 시퀀스로 구성된 플래그(flag)에 의해 끝난다. 이 영역에서는 고정된 구조이므로 코딩(송신할 때)과 디코딩(수신할 때)을 위한 비트 채워 넣기 논리가 사용되지 않는다.

CAN 전송 속도의 판단은 한 프레임 중 최소 비트의 시간을 측정하여 판단할 수 있지만 정상신호의 프레임 중 판단하기 쉬운 ACK의 신호를 이용하면 쉽게 판단할 수 있다. ACK은 1bit이기 때문으로 ACK bit의 시간이 2μs인 경우 1초 ÷ 2μs = 500,000bit가 되어 통신 속도는 500kbps가 된다. 반대로 통신 속도를 알고 있고 한 개 bit의 시간을 구하고 싶다면 통신 속도의 역수가 1bit의 시간에 해당한다.

표준 데이터 프레임의 각 필드별 비트수는 표와 같으며 이중 우성신호는 SOF를 시작으로 마지막 우성신호인 ACK 필드의 첫 번째 비트까지는 총 100개 비트로 구성되어지므로 성능이 떨어지는 오실로스코프를 이용하더라도 첫 우성에서 마지막 우성까지의 시간을 측정하여 100비트로 나누면 1비트 당 시간을 알 수 있어 통신 주파수를 예측할 수 있다.

필드명	SOF	ID	Control	Data	CRC	ACK		EOF
우성(d)과 열성(r)	d	~				d	r	r
비트수	1	12	6	64	16	1	1	7

표 4) 표준형 데이터 프레임의 비트수