byteflight 입문

Safety-critical 애플리케이션을 위한 time-controlled 프로토콜

byteflight는 주로 모터 차량에서의 safety-critical 애플리케이션을 위해 Motorola, Elmos, Infineon, BMW가 함께 개발한 것입니다. 이 시스템은 주로 BMW 7 시리즈에서 차체의 데이터 전송과 섀시 전자 기술에 덧붙여, 특히 에어백 시스템의 time-critical 데이터의 전송을 위해 사용됩니다. byteflight는 10Mbps의 데이터 속도를 갖습니다. 플라스틱 광 섬유들이 전송 매체로 사용됩니다. 버스 가입자(subscriber)들은 지능적 star coupler로 star net 구성에 연결됩니다.

byteflight 기술은 특히 여기에 필요한 구성 요소들이 시중에서 널리 이용되기 때문에 단지 자동차 애플리케이션에만 관심을 두지 않습니다. 빠른 데이터 속도를 가진 매우 높은 실시간 필수 조건들을 필요로 하는 애플리케이션들 또는 전송 결함 없이 매우 어려운 전자기 환경에서 기능해야 하는 모든 애플리케이션들에서 사용 가능합니다.

CAN 처럼, byteflight 역시 메시지-지향 전송 처리에 바탕을 두고 있으며,메시지 포맷 또한 CAN과 매우 유사합니다. 데이터 필드의 최대 길이는 12 byte입니다. CAN과의 근본적 차이는 버스가 접속되는 방식입니다. 이것은 이른 바 "Time-Division-Multiple-Access(TDMA)"라 불리는 원리에 바탕을 둡니다. 즉, 한 개 특정 메시지가 전송될 수 있는 범위에서의 "time slot" 정의. 이 처리에서 요구되는 개별적인 버스 가입자들의 가장 정확한 시간 동기화는 한 특정 버스 가입자에 의한 소위 "동기 펄스 (synchronization pulse)"로 불리는 한 개의 순환 전송(cyclic transmission)에 의해 실행됩니다. 이와 같은 동기 펄스들간의 시간 간격은 "동기적(synchronous)" 으로 불리는 메시지들의 전송을 위해 다수의 시간 슬롯들로 분할됩니다. 다시 말하면, 각 사이클에서 보내지는 상위-우선 순위 메시지들과 단지 이따금씩 전송되는 낮은-우선순위 "비동기" 메시지들의 전송을 위한 잔여 시간 범위 (그림 1). 그러므로 비동기 메시지들은 그들을 위해 남겨진 time window를 공유

합니다. 여기서, 한 개의 비동기 메시지만이 만약 그 메시지가 실제적으로 전송된다면 full time 슬롯을 차지하게 됩니다. 이런 방식으로 단지 가끔씩 전송되는 다수의 하위-우선 순위 메시지들과 매우 빠른 시간 결정을 포함하는 제한된 수의 상위-우선 순위 메시지들의 전송을 위해 이용 가능한 대역폭을 매우 효과적으로 사용하는 것이 가능해집니다. 비동기 메시지들의 이러한 time window의 유연한 사용 덕분에, byteflight에 의해 사용되는 버스 할당 처리는 "Flexible Time Division Multiple Access(FTDMA)"로 언급되기도 합니다.

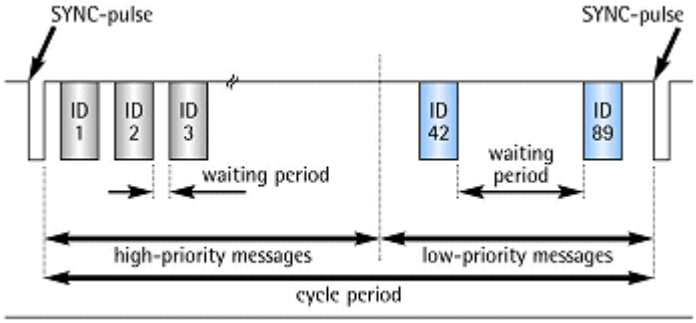

그림 1 : FTDMA process (principle)

byteflight의 현재 표준 데이터 속도 10 Mbps를 이용하면, information 업데이트 속도는 250 µs입니다. 사이클의 초기에서, 슬롯 카운터로 불리는 것들은 모든 가입자들에 대해 0 으로 설정됩니다. 모든 가입자들은 이제 동시에 그들의 슬롯 카운터의 증가와 함께 시작합니다. 한 가입자의 슬롯 카운터가 이 노드에 나타난 전송 요청 식별자에 일치하는 값에 도달하면, 이 메시지가 그 후에 전송됩니다. 여기서 확실히 할 것은 한 개의 식별자가 이 시스템에서 이중으로 점유되어 있지 않아야 합니다. 한 메시지의 전송 동안에, 모든 가입자들의 슬롯 카운터들은 정지됩니다.

byteflight의 메시지 포맷 (그림 2)은 CAN의 그것과 원칙적으로 일치합니다. 6-bit start sequence의 시작, 8-bit 메시지 식별자, 그리고 1 length byte, 최대 12 data byte가 그 다음 데이터 필드에서 전송 될 수 있습니다. Closing 2-byte CRC-sequence를 통해, 6의 hamming 간격이 있는 하나의 결함 검출이 이루어집니다. 비트 코딩 또한 CAN 프로토콜과 부합합니다. 그러나 비트 재동기화가 bit stuffing에 의한 것이 아니라, start와 stop 비트에 의해 각 바이트를 프레임함으로써 실행됩니다.

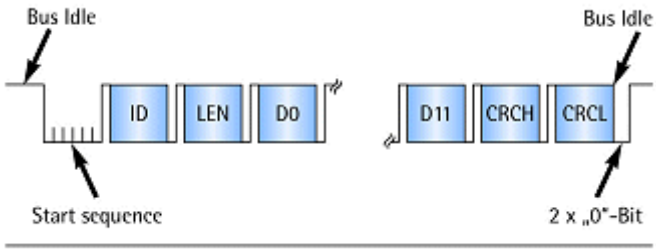

그림 2 : Message format byteflight