클럭 동기 구조를 handshake 로 사용하기

I2C 프로토콜은 또한 동기화 구조를 포함하고 있습니다. 이것은 느리고 빠른 디바이스들 간에 그리고 다중-마스터 세션에 있는 마스터들 간에 핸드쉐이크 구조로 사용될 수 있습니다.

Slow 슬래이브가 (내부 실행 관점에서의 slow) 버스에 배속될 때 그러면 문제가 발생할 수 있습니다. 직렬 EEPROM 을 가정해 봅시다. EEPROM 안에서의 실제 writing 처리는 약간의 시간이 걸릴 수 있습니다. 이제 그와 같은 디바이스에 다중 바이트를 보낸다면, 이것이 write 사이클을 완료하기 전에 새 데이터를 보내는 위험이 존재합니다. 이것은 데이터를 파괴하거나 데이터 손실을 유발할 것입니다.

슬래이브는 마스터에게 바쁘다고 말할 수 있는 몇 가지 수단을 갖고 있어야 합니다. 이것은 물론 ACK 사이클에 간단하게 응답할 수 없을 것입니다. 이는 마스터가 stop 조건을 전송하게 하고 재 시행하도록 합니다. (이것이 EEPROM 의 하드웨어에서 이루어지는 방식입니다.)

다른 경우들도 그다지 간단하지 않을 수 있습니다. A/D 컨버터에 대해 생각해봅시다. 변환이 완성되기까지는 어느 정도의 시간이 걸릴 것입니다. 마스터가 그저 계속 진행한다면 이것은 새롭게 얻은 데이터 대신 이전 변환의 결과를 읽게 될 것입니다.

이제 동기화 구조가 도움이 될 수 있습니다. 이 구조는 SCL 라인에서만 동작합니다. 마스터가 대기하기를 원하는 슬래이브는 단지 필요한 만큼 SCL 을 LOW 로 끌어 당깁니다. 이것은 “wait states”를 I2C 버스 사이클에 추가하는 것과 같습니다.

그러면 마스터는 SCL 라인을 high 로 가게 할 수 없기 때문에 ACK 클럭 펄스를 만들 수가 없습니다. 물론 마스터 소프트웨어는 반드시 이 상황을 확인하고 그에 따라 행동해야 합니다. 이 경우, 마스터는 SCL 라인을 HIGH 로 가게 할 수 있을 때까지 그저 대기한 다음 이것이 무엇을 하고 있었든 그에 맞춰 진행합니다.

이것을 구현할 때 포함되는 많은 사소한 장애들이 있습니다. SCL 이 회로의 전기적 실패로 인해 꼼짝 못하면, 마스터는 교착 상태로 들어갈 수 있습니다.

물론 이것은 타임 아웃 카운터에 의해 처리될 수 있습니다. 또한 버스가 이처럼 막히면, 통신은 어떻든 되지 않습니다.

또다른 장애는 속도입니다. 버스는 현재 잠겨있습니다. 사용자가 다소 오래 지연을 하면 (위의 예에서 긴 변환 시간), 이것은 전체 버스 속도를 상당히 불리하게 합니다. 그때는 다른 마스터들이 버스를 어느 쪽도 사용할 수 없습니다.

이 기법은 low SCL 라인이 버스를 요구하려고 할 수 있는 다른 디바이스들에서 back-off 상황을 이끌기 때문에 앞서 소개된 arbitration 구조를 방해하지 않습니다. 그러므로 이 기법에서 속도와 대역폭 그리고 마스터에서의 약간의 소프트웨어 오버헤드를 제외하고는 실질적인 장애는 없습니다.

사용자는 다중-마스터 환경에서의 마스터들 간에 이 구조를 사용할 수 있습니다. 이것은 다른 마스터가 버스를 데려가는 것을 막을 수 있습니다. 2-마스터 시스템에서 이것은 도움이 되지 않습니다. 그러나 세 개 또는 그 이상의 마스터들이 있다면 이것은 매우 도움이 됩니다. 세 번째 마스터는 이 방식에서 마스터 1 과 2 간의 전송을 방해할 수 없습니다. 어떤 중요한 임무를 수행하는 상황에서 이것은 매우 우수한 기능이 될 수 있습니다.

사용자는 SCL 라인뿐 아니라, SDA 라인을 low 로 끌어서 이 기법을 엄격하게 만들 수 있습니다. 그러면 서로 대화하는 두 개 마스터들 외의 모든 마스터는 즉시 back off 될 것입니다. 사용자는 계속하기 전에, stop 조건을 표시하여, 먼저 SDA 를, 그리고 SCL 을 high 로 돌아가게 합니다. 한편 통신을 시도했던 모든 마스터는 back-off 상황을 탐색했을 것이며 STOP 이 나타나기를 기다리고 있었을 것입니다.