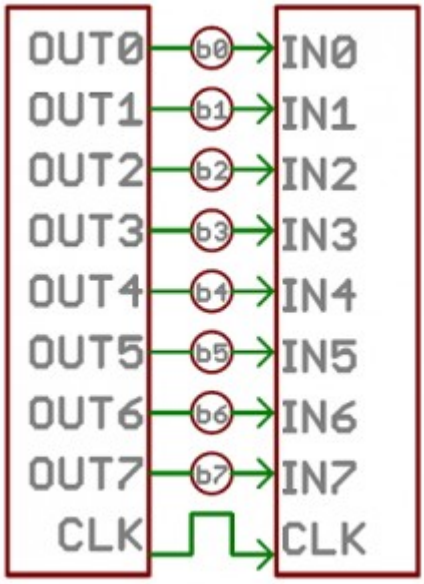
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 최대성
c3d4s19@naver.com

2018.05.14. – 053 일차 수업

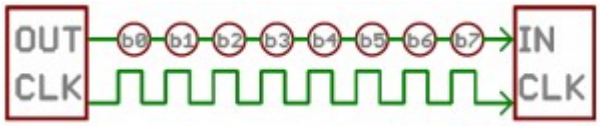
Serial 통신(직렬 통신)통신은 방식에따라 Serial 통신(직렬 통신)과 Parallel 통신(병렬 통신) 으로 나눌 수 있다.

- Parallel 통신 -



Parallel 통신은 다음과 같이 다수의 bit 를 타이밍(clock, CLK)에 맞춰 동시에 전송하는 방법이다.

- Serial 통신 -



Serial 통신은 데이터를 직렬화 시켜서 하나씩 순서에 맞춰 전송하는 방법이다.

속도만 고려하면 Parallel 통신이 더 좋아보이지만 Parallel 통신은 회로 노이즈에 취약하며 IO 핀 소모가 많아 실제로는 대부분 Serial 통신을 사용한다.

Serial 통신은 동기식(synchronous) 시리얼 통신과 비동기식(asynchronous) 시리얼 통신으로 나눌 수 있다.

- 동기식 시리얼 통신 -

동기식 시리얼 인터페이스는 데이터 라인의 동작을 클럭 시그널 라인의 동작과 동기화 시키고 시리얼 버스에 함께 연결된 장치들이 이 클럭을 공유한다.

대표적으로는 SPI, I2C 프로토콜이 이런 방식을 사용한다.

- 비동기식 시리얼 통신 -

비동기식은 데이터가 외부 클럭 시그널의 도움 없이 동작하는 것을 의미한다. 대신 데이터를 안정적으로 전송, 수신할 수 있도록 처리를 해야한다.

대표적으로는 UART 통신이 있다.

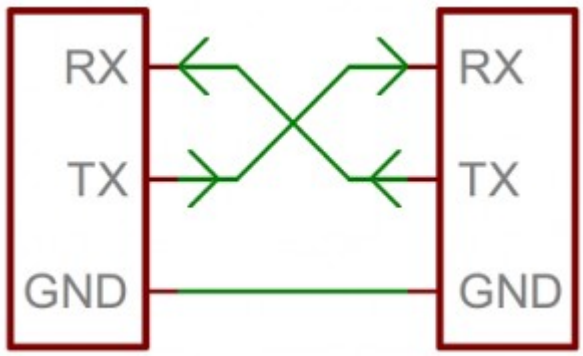
- UART 통신-

UART 통신을 하기 위해서 Baud Rate(통신 속도)를 먼저 설정해주어야 하는데 Baud Rate 는 bps(bit-per-second) 단위로 표시하며 통신 속도가 중요하지 않는 경우 일반적으로 9600 bps 를 사용한다. 다른 표준 Baud Rate 값으로는 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 을 사용할 수도 있고 값이 높을수록 전송/수신 속도가 빠르지만 Baud Rate 값이 너무 높으면 데이터 수신에 문제가 발생한다.

UART 통신 전송에 사용되는 데이터 패킷은 다음과 같이 구성되며 Start bit 를 제외하고 가변적인 크기를 가질 수 있다.

Start bit 와 Stop bit 는 통신의 시작과 종료를 의미하는 동기화 비트이고 Data bit 들은 실제 전송할 데이터가 들어있고 Parity bit 는 회로상 노이즈에 취약한 환경인 경우 데이터를 모두 더해서 홀수 짝수를 구분하는 등의 연산으로 잘못 전송된 Data 를 걸러내는 bit 이다. Parity bit 는 선택사항이며 사용시 문제가 발생하는 경우 데이터를 다시 받을 수 있도록 처리해야 한다.

- UART 통신 연결시 주의사항



UART 통신 연결시 TX(전송 핀) RX(수신 핀) 두 개를 서로 엇갈리게 연결해야 동작한다.

- 비동기식 시리얼 통신의 문제점

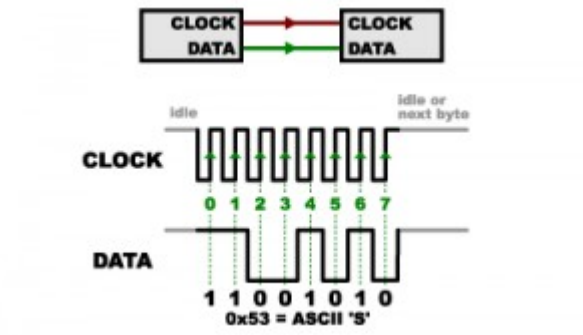
통신을 하는 두 장치 사이에 데이터를 보내는 중인지 알려주는 별도의 컨트롤러가 없어서 끊임없이 상대가 데이터를 보내는지 확인을 해야한다.

UART 통신을 사용하는 장치들은 1 초를 몇 개의 간격으로 나눠서 데이터를 보낼지 Baud Rate 를 미리 정해둬야 한다.

이 간격을 정확하게 지키기 위해서는 통신에 참여하는 장치가 모두 같은 클럭(clock)을 사용해야 하는데 만약 클럭이 미세하게 다르다면 문제가 발생할 수 있다.

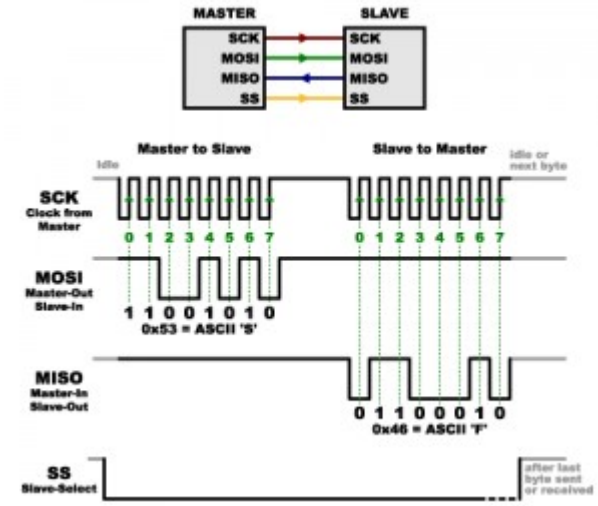
이 문제를 해결하기 위해 UART 통신은 매 1byte 송신을 할 때마다 추가로 start/stop bit 를 추가해야 한다.

- 동기식 시리얼 통신



동기식 시리얼 통신은 데이터를 보낼 때 데이터가 전달된다고 알려주는 별도의 클럭(clock)라인을 사용한다. 클럭 라인을 통신 참여자가 공유하여 언제 데이터가 오는지 알 수 있도록 해준다.

- SPI(Serial Peripheral Interface) 통신



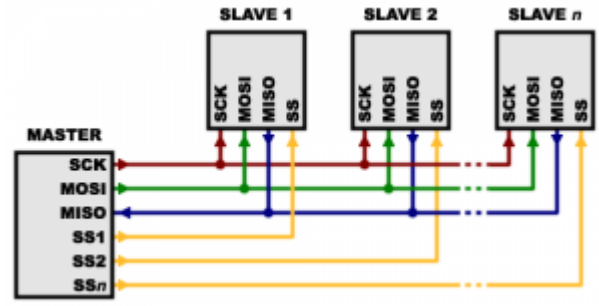
SPI 통신은 하나의 Master 장치 에서만 클럭(SCK) 라인에 신호를 생성하고 1:N 통신이 가능하여 Slave 장치는 여러 개가 될 수 있다.

데이터가 Master 에서 Slave 로 전달될 때는 MOSI(Master Out / Slave In)라인을 사용하며

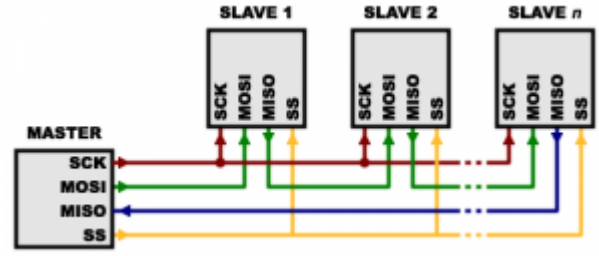
반대로 Slave 에서 Master 로 전달될 경우는 MISO(Master In / Slave Out)를 사용한다.

SPI 통신은 여러개의 Slave 장치와 통신하기 위해서 SS(Slave Select)라인을 사용하며 통신을 원하는 Slave 장치의 SS 라인을 High 상태에서 Low 상태로 바꾸는 방식으로 특정 Slave 장치와 통신한다.

- 다수의 Slave 장치 연결 방식

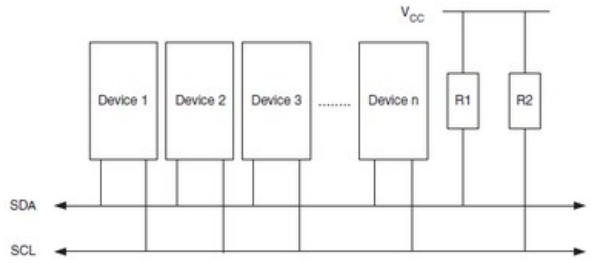


→ 장치마다 SS 선 할당

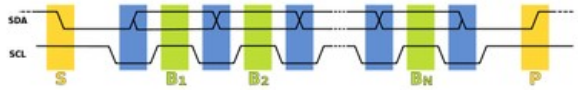


→ MOSI, MISO 라인을 체인처럼 연결하고 SS 공유

- I2C(=TWI) 통신



I2C 통신은 데이터를 주고 받기 위한 선(SDA) 하나와 송수신 타이밍 동기화를 위한 클럭 선(SCL)하나로 이루어지고 Master 와 Slave 간 N:N 통신이 가능하다.



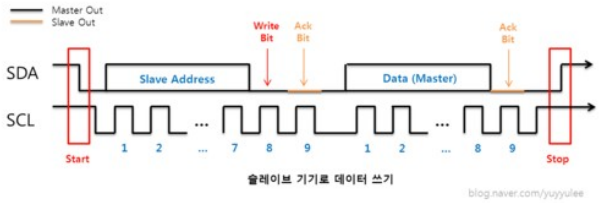
SDA 와 SCL 선의 신호는 풀업 저항에 의해 기본적으로 High 상태이며 여기서 SDA 신호만 Low 로 떨어지면 시작 신호라고 판단한다. 그 후에 SCL 선으로 클럭 신호가 만들어지는데 클럭신호가 Low 일 때가 SDA 신호를 비트 신호로 바꾸는 시간(파란색 부분), 클럭 신호가 High 일 때가 SDA 신호를 읽는 시간(녹색 부분)이다.

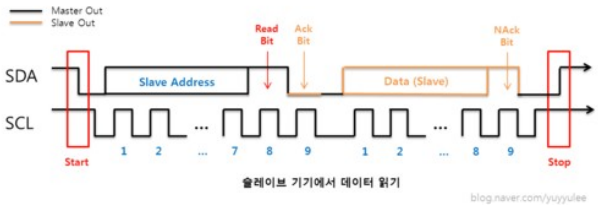
한 클럭에 한 비트씩 데이터 신호를 만들고 모든 비트의 전송이 끝난 후 SCL 신호가 High 가 되면 SDA 신호 역시 High 로 만들어 정지 신호를 만든다.

시작 신호 뒤에 나오는 첫 7 비트는 Slave 의 주소 값이고 8 번째 비트는 데이터를 읽는 신호인지 쓰는 신호인지를 나타내는 비트로 사용된다.

슬레이브의 주소 값과 읽기/쓰기 비트는 마스터에서 생성할 수 있으며 '쓰기'일 경우는 Master 에서 이후 데이터를 생성, '읽기'일 경우는 Slave 에서 데이터를 생성한다. 8 비트 데이터 전송 후에는 Slave 에서 응답신호(ACK)를 만들어 수신을 확인해준다.

응답신호는 기본적으로 LOW 이며 만일 Slave 가 데이터를 전송하는 상태에서 모든 데이터의 전송이 끝났을 경우 High 상태가 되는데 이를 NACK 라고 한다. 이 외의 경우에 NACK 가 발생하면 통신 에러나 데이터 에러이다.





검은색은 Master 에서 만드는 신호

주황색은 Slave 에서 만드는 신호

참고 자료

http://www.hardcopyworld.com/ngine/aduino/index.php/archives/2080

http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo_table=lecture_pract&wr_id=94

https://m.blog.naver.com/PostView.nhn?blogId=yuyyulee&logNo=220323559541&proxyReferer=https%3A%2F%2Fwww.google.com%2F