AVR - HW8

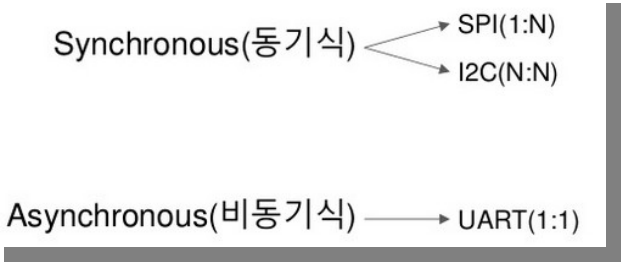
임베디드스쿨1기

Lv1과정

2020. 11 . 06

박하늘

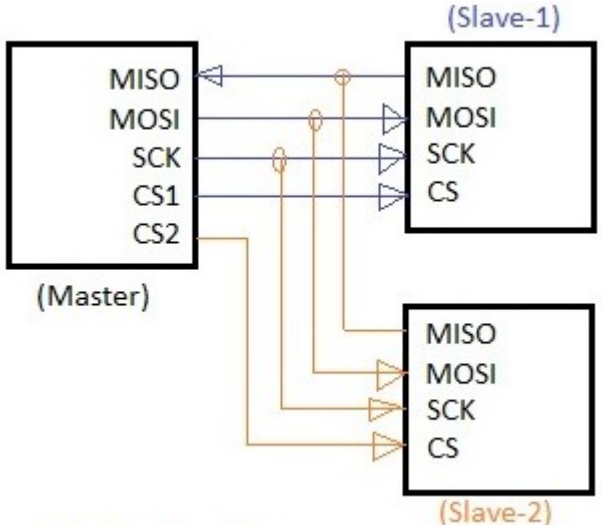
1. [Review] SPI



동기, 비동기는 데이터가 클럭신호에 동기가 되는지 그렇지 않은지로 구별된다. 데이터라인만 있고 클럭 신호가 없는것을 비동기 통신임

1) SPI(Serial Peripheral Interface)는?

- 동기식 직렬 통신 방식(같은 clock을 사용하여 통신을 한다)
- 3가닥의 선(MOSI, MISO, SCLK)으로 Full-duplex 통신이 가능
- 싱글 MASTER 구조를 가지고 있으므로 1:N으로 다수의 외부 주변 회로와 통신이 가능함
- 이때 주변회로를 선택하기 위한 선 (/SS)이 추가로 필요하다 (이때, SS를 여러개 사용하려면 MUX를 활용한다)



1. [Review] SPI

Code 동작: master에 'A'라는 대문자를 data register(SPDR)에 가지고 있고, slave에 하나씩 보냄
Slave는 'A'를 받아 'a' 바꾸어 data register (SPDR)에 가지고 있는다. 그러면 또 'A'라는 데이터가 밀려오며, 한 BIT씩 밀려서 Master에 보냄

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

#define sbi(PORTX, BitX) (PORTX |= (1 << BitX))
#define cbi(PORTX, BitX) (PORTX &= ~(1 << BitX))

void UART_INIT(void){
    sbi(UCSR0A, U2X0);  //U2X0 = 1 --> Baudrate 9600 = 207

    UBRR0H = 0X00;
    UBRR0L = 207;       //Baudrate 9600

    UCSR0C |= 0X06;     //1 stop bit, 8 bit data

    sbi(UCSR0B, RXEN0); //enable receiver and transmitter
    sbi(UCSR0B, TXEN0);
}
unsigned char UART_receive(void)
{

    while(!(UCSR0A & ( 1 << RXC0 )));   //wait for data to be recieved
    return UDR0;    //get and return received data from buffer
}
unsigned char UART_tranmit(unsigned char data)
{

    while(!(UCSR0A & ( 1 << UDRE0 )));  //wait for data to be trasmitted
    UDR0 = data;    //put data into buffer, sends the data
}
void SPI_Master_Init(void)
{

    sbi(DDRB, 5);       //OUTPUT SCK
    sbi(DDRB, 3);       //OUTPUT MOSI
    sbi(DDRB, 2);       //OUTPUT SS
    cbi(DDRB, 4);       //OUTPUT MISO

    PORTB = 0xff;       //PULL UP
    sbi(SPCR, SPE);     //SPE, Mast
    sbi(SPCR, MSTR);    //master se

}
```

1. 통신 속도 설정

	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
Baud Rate (bps)	UBRRn	Error	UBRRn	Error
2400	416	–0.1%	832	0.0%
4800	207	0.2%	416	–0.1%
9600	103	0.2%	207	0.2%

2. 데이터 설정

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

1) UCSR0C = 0x06 // 0000 0110 data 8bit 설정
2) UCSR0A & (1 << RXC0) // 수신버퍼가 비어 있다면 0으로 클리어, 1이 되면 데이터 수신 받았다는 것
3) UCSR0A & (1<<UDRE0) // 전송버퍼가 비어 있다면 1 값을 가지고 새로운 데이터를 받아서 전송할 준비 완료된 것

3. spi masster init(void)
- SCK, MOSI, SS : SET
- MISO : RESET

4. PortB Pull Up을 하는 이유

If SS is configured as an input, it must be held high to ensure master SPI operation. If the SS pin is driven low by peripheral circuitry when the SPI is configured as a master with the SS pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. [Review] SPI

Code 동작: master에 'A'라는 대문자를 data register(SPDR)에 가지고 있고, slave에 하나씩 보냄
Slave는 'A'를 받아 'a' 바꾸어 data register (SPDR)에 가지고 있는다. 그러면 또 'A'라는 데이터가 밀려오며, 한 BIT씩 밀려서 Master에 보냄

```
void SPI_Slave_Init(void)
{
    cbi(DDRB, 5);
    cbi(DDRB, 3);
    cbi(DDRB, 2);
    sbi(DDRB, 4);

    PORTB = 0xff;
    sbi(SPCR, SPE);
}
```

master

```
unsigned char SPI_TxRx(unsigned char Data) //Master Root
{

    SPDR = Data;// SPDR 을 값을 쓰자 마자 바로 슬레이브에 보냄.
    while(!(SPSR & (1<<SPIF)));//while문에서 상태 확인한다. SPSR에 값이 1이 되면 값을 받겠다.
                               //Slave로 부터 문자를 받으면 1이 set된다. 데이터를 보내는 순간 0으로 set된다.
    return SPDR;
}
```

```
int main(void)
{
    unsigned char data;
    UART_INIT();
    //SPI_Master_Init();
    SPI_Slave_Init();
```

master

```
/*
    cbi(PORTB, 2); //Start Communication , SS를 클리어 되면 통신이 시작된다.
    while(1)    //master
    {
        data = SPI_TxRx('A');
        UART_tranmit(data);
        _delay_ms(1000);
    }
*/
```

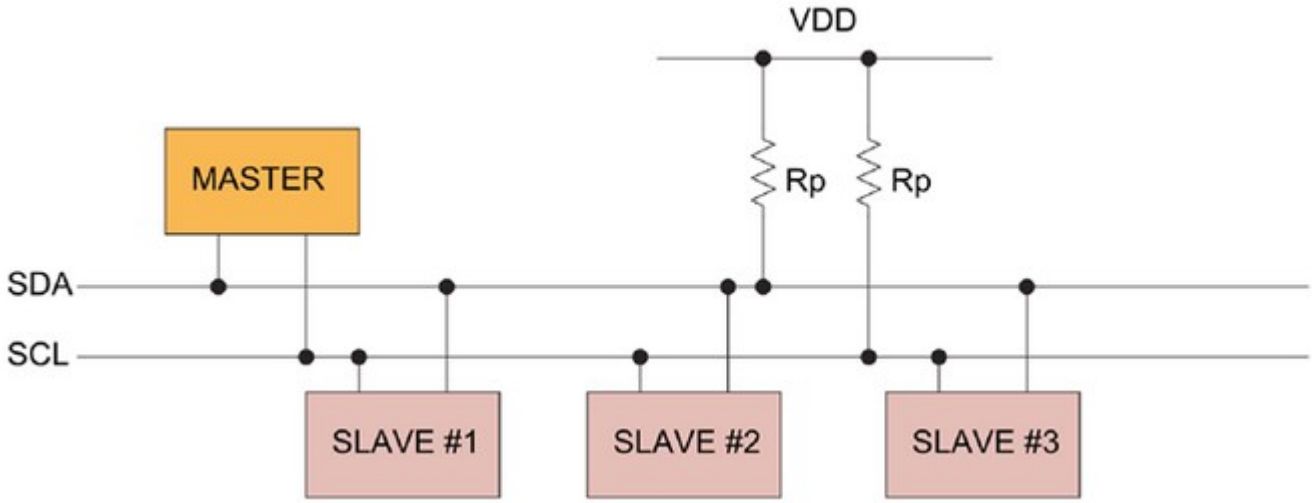
slave

```
    while(1)  //Slave
    {
        while(!(SPSR & ( 1 << SPIF)));//while문에서 상태 확인한다. SPSR에 값이 1이 되면 값을 받겠다.
                                      //Master로 부터 문자를 받으면 1이 set된다. 데이터를 보내는 순간 다시 0으로 set이 된다.
        SPDR = SPDR + 0X20; //Master에서 받은 문자(대문자)를 +0x20을 받아 소문자로 바꿔줌
    }
    return 0;
}
```

2. [Preview] I2C

1) I2C (Inter Integrated Circuit)

- 2가닥의 선 (SDA, SCL)로 Half-duplux 통신이 가능
- Multi Mater 모드를 지원하기 때문에 N:N으로 다수의 외부 주변회로와 통신이 가능
- SCL: 동기를 위한 Clock용 선, SDA: 데이터용 선
- SCL, SDA선 모두 오픈 드레인으로 풀업 저항 연결



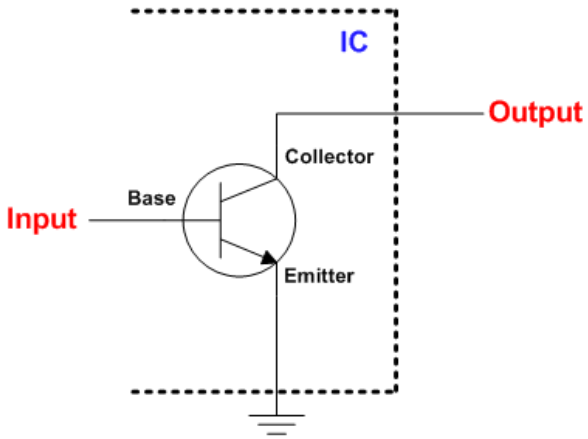
2) 동작구조

: 12C는 Multi master와 Multi Slave 모두 같은 통신 라인에 연결 되어 있다.

2-1) 라인에 연결된 마스터중 하나가 start신호 보내면 bus를 점유
2-2)bus를 점유한 master가 slave address와 read/write 정보를 bus에 전송
2-3)master가 보낸 address와 일치하는 slave가 ACK신호를 전송
2-4)ACK신호를 받은 마스터가 Slave에 data를 전송, 전송 후 bus 점유를 해제

3) 오픈 드레인?

:Open-collector(오픈콜렉터)/open-drain(오픈드레인)은 여러개의 장치(Device)를 하나의 연결선으로 양방향(bi-directionally) 통신할 수 있도록 하는 Circuit 테크닉



Open-Collector의 경우 입력이 Low 일때 Pull-Up 저항때문에 High 가 되며 Transistor에 의해 연결이 끊어진 상태가 됨. 따라서 여러개의 Device가 연결되어도 신호의 충돌이 생기지 않는다.
Open-Collector를 사용하지 않을경우 여러개의 Device에서 Low또는 High 신호를 보내주면 단락이 발생 할 수 있음

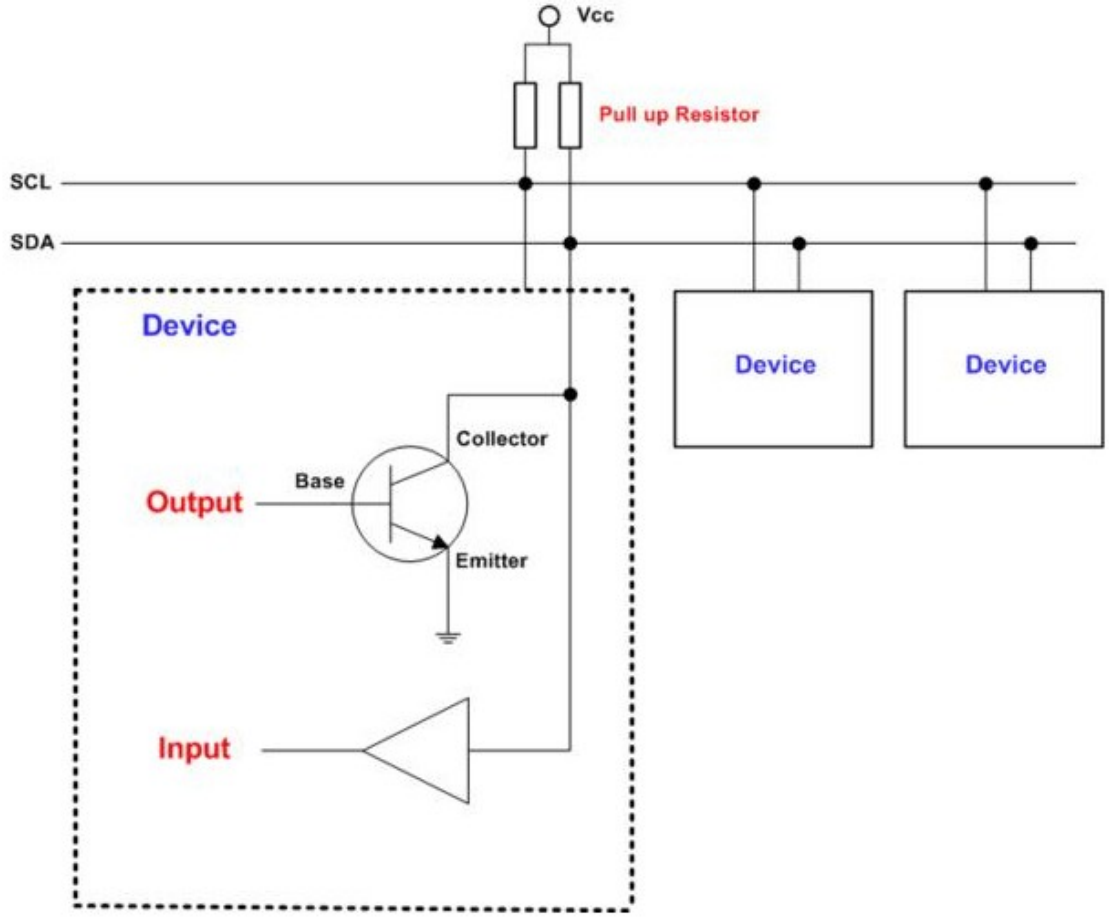
Open Collector의 특징

1. 여러개의 Device를 연결하여 양방향(Bi-directionally)으로 Data를 전송가능
2. Level이 다른 Device에 Data 전송이 가능 (예로 3v를 사용하는 Master 가 5v를 사용하는 Slave와 통신이 가능)

2. [Preview] I2C

4) 12C에서 오픈드레인 ?

:I2C도 이와 같은 개념을 이용.
I2C에서 제일 처음 하는 동작이 SDA로 Slave Address를 전송한다. 전송후 ACK를 기다린다.
이때 여러개의 Device 중에 Slave Address에 해당하는 Device가 SDA Line을 Low로 만들면 Master는 해당 Device가 존재한다고 판단하게 됨.



감사합니다.