

시리얼 인터페이스

- TWI(Two Wire Serial Interface)
- SPI(Serial Peripheral Interface)
- TWI(I2C)로 온습도 센서 제어하기
- 타이머를 이용하여 측정한 온습도를 PC에 출력하기
- SPI로 EEPROM 붙이기
- EEPROM에 저장된 온습도 정보를 PC에 출력하기



엣지아이랩

시리얼 인터페이스

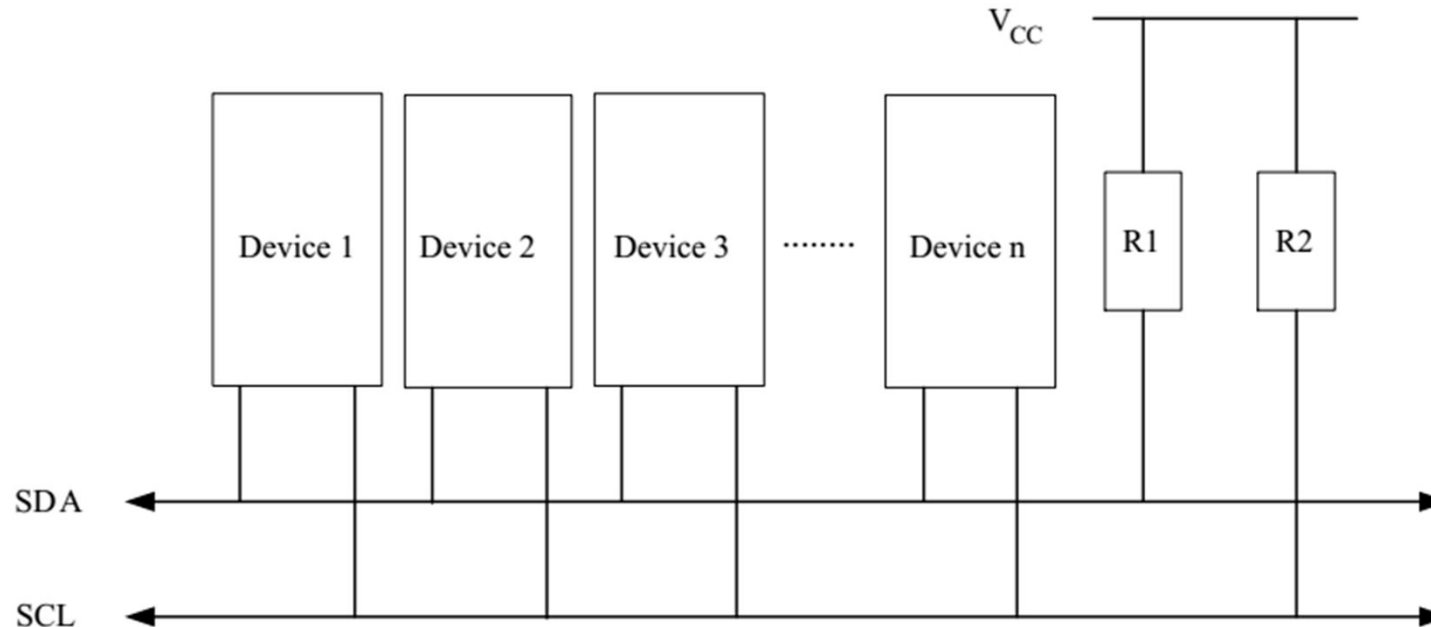
- 패러렐 인터페이스와 시리얼 인터페이스
 - 패러렐 인터페이스(Parallel Interface)
 - 데이터 및 어드레스가 병렬로 동시에 처리
 - 데이터의 처리 속도 빠름
 - SRAM등의 외부 메모리 및 고속의 주변 장치들을 연결하는데 주로 사용
 - 다수의 어드레스 신호와 데이터 신호를 사용 칩의 크기를 소형화 어려움
 - ATmega128의 외부 메모리 인터페이스
 - 시리얼 인터페이스(Serial Interface)
 - 고속의 제어가 필요없는 장치들을 위해 소수의 신호를 사용
 - 어드레스와 데이터를 순차적으로 처리
 - 데이터의 처리속도가 비교적 느림
 - 필요한 핀수를 최소화하여 칩의 소형화에 유리
 - 고속제어가 필요없는 소형 칩들에 주로 사용
 - TWI(Two Wire Serial Interface), SPI(Serial Peripheral Interface)

TWI(Two-wire Serial Interface)

- TWI(Two Wire Interface)
 - 단순하면서도 강력한 시리얼 통신 인터페이스
 - 필립스사에서 제창한 I2C(Inter IC Bus)와 같은 방식
 - 2선을 이용해 시스템 내부에서 여러 장치들과 통신
 - TWI 프로토콜은 클록(SCL)과 데이터(SDA)만으로 양방향 버스라인 사용
 - 마스터와 슬레이브 동작을 지원하며, 다중 마스터도 가능
 - TWI의 7비트 어드레스는 128개의 다른 슬레이브 어드레스까지 허용
 - 버스에 연결된 모든 디바이스는 독립적인 주소를 가짐
 - 디바이스 어드레스(Device Address) or 디바이스 아이디(Device ID)라 하며, 칩의 구분을 위해 각 칩마다 고유의 디바이스 ID를 가짐
 - TWI는 400kHz까지의 데이터 전송 속도를 가짐

TWI(Two-wire Serial Interface)

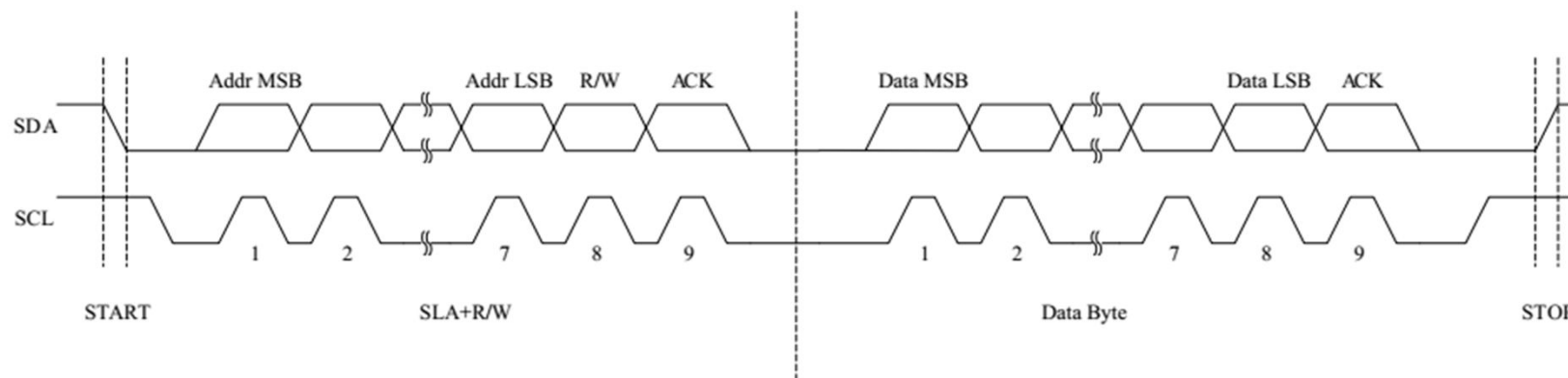
- TWI(Two Wire Interface)



- 버스 라인(SCL, SDA)은 풀업 저항을 통하여 + Vcc 전압으로 연결
- 각 디바이스들은 평상시 Tri-State 상태를 유지
- 버스를 사용할 때에는 레벨 하이('1')는 Tri-State로, 레벨 Low('0')는 '0'로 출력

TWI(Two-wire Serial Interface)

- TWI 데이터 전송형식
 - 마스터가 버스에 START 조건을 출력할 때, 전송은 시작되고 STOP 조건에서 완료
 - START 조건, 어드레스 패킷(SLA + R/W)와 하나 또는 많은 데이터 패킷 그리고 STOP으로 구성



TWI(Two-wire Serial Interface)

- ATmega128A TWI 레지스터
 - TWBR(TWI Bit Rate Register)
 - TWI 비트율 레지스터
 - TWCR(TWI Control Register)
 - TWI 제어 레지스터
 - TWSR(TWI Status Register)
 - TWI 상태 레지스터
 - TWDR(TWI Data Register)
 - TWI 데이터 레지스터
 - TWAR[TWI (Slave) Address Register]
 - TWI 슬레이브 어드레스 레지스터

TWI(Two-wire Serial Interface)

- TWBR(TWI Bit Rate Register)
 - TWI 비트율 레지스터
 - 비트율 발생기에 대한 분주 요소를 선택하기 위한 레지스터
 - 비트 7~0(TWI Bit Rate Register)
 - 비트율 발생기에 대한 분주 요소를 선택
 - 비트율 발진기는 마스터 모드에서 SCL 클록 주파수를 발생하는 주파수
 - SCL의 클럭 주파수

$$SCL \text{ 주파수} = \frac{CPU \text{ 클록 주파수}}{16 + 2(TWBR) \cdot 4^{TWPS}}$$

7	6	5	4	3	2	1	0
TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

TWI(Two-wire Serial Interface)

- TWCR(TWI Control Register)
 - TWI 제어 레지스터(TWI 동작 제어)
 - 비트 7 : TWINT(TWI Interrupt Flag)
 - TWI 인터럽트 플래그
 - TWI가 현재 작업을 완료하고 응용 소프트웨어 응답을 기다릴 때, 하드웨어에 의해 세트됨
 - SREG의 I 비트와 TWCR의 TWIE 비트가 세트되면 TWI 인터럽트 벡터로 점프

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

TWI(Two-wire Serial Interface)

- TWCR(TWI Control Register)
 - 비트 6 : TWEA(TWI Enable Acknowledge Bit)
 - TWI Enable 응답 비트(ACK 펄스의 생성을 제어)
 - 이 비트에 1을 써넣은 후, 다음 조건을 만나면 ACK 펄스가 발생
 - 디바이스에 자기 슬레이브 어드레스가 수신될 경우
 - TWAR의 TWGCE 비트가 세트인 동안 일반적인 호출이 수신될 경우
 - 데이터 바이트가 마스터 수신기 또는 슬레이브 수신기 모드로 수신될 경우
 - TWEA 비트에 0를 써넣으면 디바이스는 TWI로부터 일시적으로 끊어짐
 - 비트 5 : TWSTA(TWI START Condition Bit)
 - TWI START 조건 비트
 - TWI 버스에서 마스터가 되고자 할때 이 비트에 1을 써넣는다.
 - START 조건이 전송되었을 때 이 비트는 소프트웨어에 의해 클리어해야 함

TWI(Two-wire Serial Interface)

- TWCR(TWI Control Register)
 - 비트 4 : TWSTO(TWI STOP Condition Bit)
 - TWI STOP 조건 비트
 - 마스터 모드에서 이 비트에 1을 써넣으면, TWI 버스로 STOP 조건을 발생시킴
 - STOP 조건이 버스에서 실행될 때 이 비트는 자동적으로 클리어됨
 - 비트 3 : TWWC(TWI Write Collision Flag)
 - TWI 쓰기 충돌 플래그
 - 이 플래그 비트는 TWINT가 Low일 때 TWDR(TWI Data Register)로 써넣기를 시도하면 1로 세트되고, TWINT가 High일 때 TWDR 레지스터에 써넣으면 클리어 됨

TWI(Two-wire Serial Interface)

- TWCR(TWI Control Register)
 - 비트 2 : TWEN(TWI Enable Bit)
 - TWI Enable 비트
 - 이 비트는 TWI 동작을 Enable시키고 TWI 인터페이스를 활성화
 - 이 비트에 1을 써넣으면 TWI가 Enable되어 SCL과 SDA 핀을 제어하게 되고, 0을 써넣으면 TWI는 오프 되고 모든 TWI 전송은 어떤 행위에 관계없이 종료
 - 비트 0 : TWIE(TWI Interrupt Enable)
 - TWI 인터럽트 Enable 비트
 - 이 비트에 1이고, SREG의 I 비트가 1로 세트 되면 TWINT가 High인 동안 TWI 인터럽트가 활성화

TWI(Two-wire Serial Interface)

- TWSR(TWI Status Register)
 - TWI 상태 레지스터
 - TWI의 상태와 프리스캐일러 값을 보여주는 레지스터
 - 비트 7~3 : TWS(TWI Status) : TWI 상태
 - TWI의 상태를 알려주는 비트

7	6	5	4	3	2	1	0
TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0

TWI(Two-wire Serial Interface)

- TWSR(TWI Status Register)
 - Master Transmitter Mode에서의 Status Code

Status Code(Hex)	설명
08	Start 코드 전송 완료
10	Repeated START 코드 전송 완료
18	SLA+W 전송완료 및 ACK 신호 수신 완료
20	SLA+W 전송 완료 및 ACK 신호 수신 불가
28	데이터 바이트 전송완료 및 ACK 신호 수신 완료
30	데이터 바이트 전송 완료 및 ACK 신호 수신 불가
38	SLA+W 나 데이터 바이트 전송시 중재 불가(오류)

TWI(Two-wire Serial Interface)

- TWSR(TWI Status Register)
 - Master Receiver Mode 에서의 Status Code

Status Code(Hex)	설명
08	Start 코드 전송 완료
10	Repeated START 코드 전송 완료
38	SLA+R 나 NOT ACK 비트 중재 불가(오류)
40	SLA+R 전송완료 및 ACK 신호 수신 완료
48	SLA+R 전송 완료 및 NOT ACK 신호 수신
50	데이터 바이트 수신완료 및 ACK 신호 반송 완료
58	데이터 바이트 수신완료 및 NOT ACK 신호 반송

TWI(Two-wire Serial Interface)

- TWSR(TWI Status Register)
 - 비트 1~0 : TWPS(TWI Prescaler Bit) :
 - TWI 프리스케일러 비트
 - 비트율 프리스케일러를 제어하기 위한 비트
 - TWI Prescaler 설정표

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWI(Two-wire Serial Interface)

- TWDR(TWI Data Register) :
 - TWI 데이터 레지스터
 - 전송 모드에서 TWDR은 전송될 다음 바이트를 포함
 - 수신 모드에서는 수신된 마지막 바이트를 포함
 - 비트 7~0 : TWD(TWI Data)
 - TWI 데이터
 - 전송되는 다음 데이터 바이트 또는 2줄 직렬 버스에서 수신된 마지막 데이터 바이트를 포함

7	6	5	4	3	2	1	0
TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0

TWI(Two-wire Serial Interface)

- TWAR[TWI (Slave) Address Register]
 - TWI 슬레이브 어드레스 레지스터
 - 슬레이브 모드로 사용될 때 슬레이브 어드레스를 저장하는 레지스터
- 비트 7~1 : TWA[TWI(Slave) Address]
 - TWI 슬레이브 어드레스
 - 이 일곱 비트는 TWI 유닛의 슬레이브 어드레스를 나타냄
- 비트 0 : TWGCE(TWI General Call Recognition Enable Bit)
 - TWI 일반호출 인식 Enable비트
 - 이 비트가 세트 되면 TWI 버스에 주어진 일반 호출의 인식을 Enable함

7	6	5	4	3	2	1	0
TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE

TWI(Two-wire Serial Interface)

- TWI 동작
 - AVR TWI는 바이트 단위로 동작이 이루어지고, 인터럽트를 기본으로 함
 - 인터럽트는 바이트의 수신이나 START 조건의 전송처럼 모든 버스의 이벤트 뒤에 발생
 - SREG의 I 비트와 함께 TWCR의 TWIE(TWI Interrupt Enable) 비트가 1로 세트 되면 TWINT 플래그가 '1'이 될 때 인터럽트가 발생
 - TWIE 비트가 클리어 되면, 응용 소프트웨어에서 TWI 버스에서의 동작 상태를 알기 위해 TWINT 플래그를 정기적으로 조사(poll)해야 함

TWI(Two-wire Serial Interface)

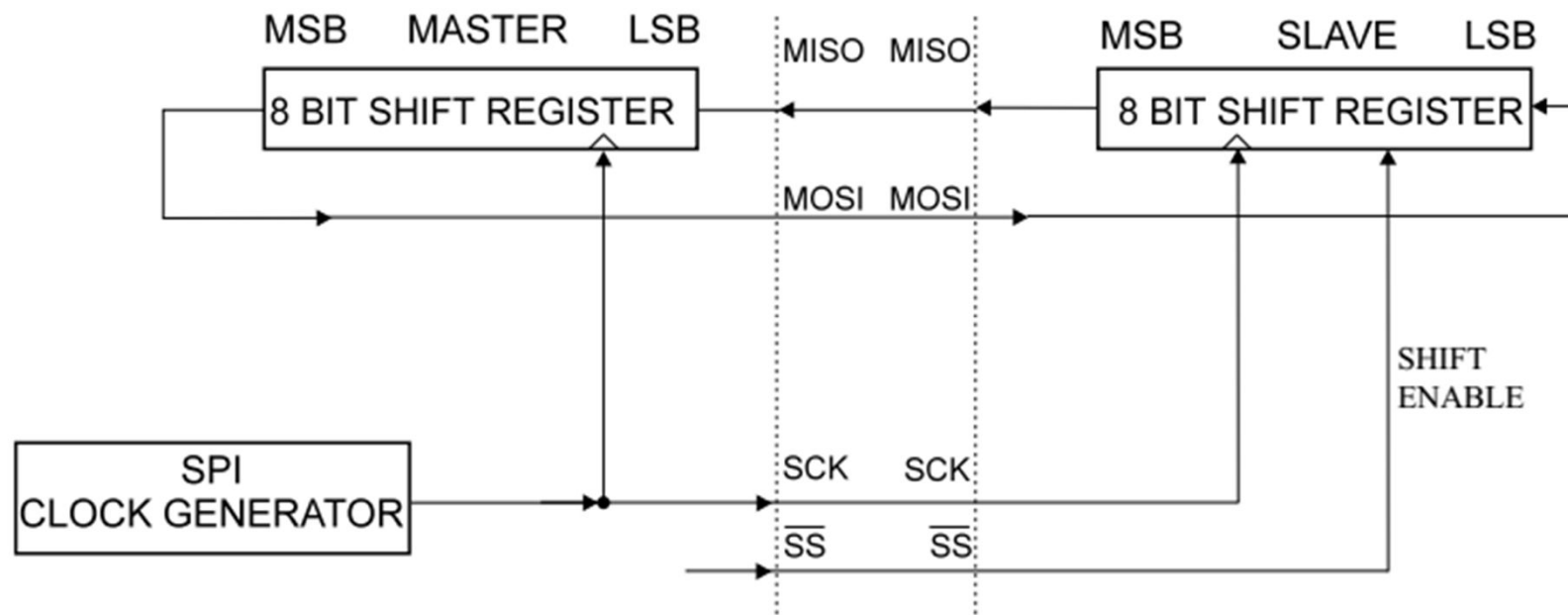
- TWI를 동작시키는 방법
 1. TWCR의 TWINT, TWSTA, TWEN 비트를 세팅하여 START Condition을 내보냄
 2. TWCR의 START Condition이 정상적으로 출력되어 TWINT 플래그가 세팅 되길 기다림
 3. TWSR을 체크하여 START 상태인지 확인, 아니면 오류
 4. SLA+W 를 TWDR 레지스터에 넣고, TWCR의 TWINT 플래그를 클리어 시킴
 5. SLA+W가 전송되고 ACK비트가 정상적으로 도착하여 TWCR의 TWINT가 세팅 되길 기다림
 6. TWSR을 체크하여 MT_SLA_ACK 상태인지 확인, 아니면 오류
 7. Data를 TWDR 레지스터에 넣고, TWCR의 TWINT 플래그를 클리어 시킴
 8. Data가 전송되고 ACK비트가 정상적으로 도착하여 TWCR의 TWINT가 세팅 되길 기다림
 9. TWSR을 체크하여 MT_DATA_ACK 상태인지 확인, 아니면 오류
 10. TWCR TWINT, TWSTO, TWEN 비트를 세팅하여 STOP Condition을 전송

SPI(Serial Peripheral Interface)

- SPI(Serial Peripheral Interface) 동작
 - SPI는 반드시 1개의 마스터와 1개의 슬레이브 사이에서만 동작
 - 마스터가 슬레이브에게 데이터를 보낼 때
 - 여러 슬레이브에서 원하는 슬레이브에게 SS(Slave Select) 신호를 0레벨로 출력하여 선택
 - 클록 신호를 발생하여 SCK(Serial Clock)을 통해 출력
 - 송신할 데이터를 시프트 레지스터에 데이터를 준비하여 MOSI(Master Output Slave Input) 단자로 출력
 - 동시에 MISO(Master Input Slave Output) 단자를 통해서는 더미 데이터가 입력

SPI(Serial Peripheral Interface)

- SPI(Serial Peripheral Interface) 동작
 - SPI Master와 Slave간 상호 연결



SPI(Serial Peripheral Interface)

- SPI(Serial Peripheral Interface) 레지스터
 - SPCR(SPI Control Register)
 - SPI 제어 레지스터
 - SPSR(SPI Status Register)
 - SPI 상태 레지스터
 - SPDR(SPI Data Register)
 - SPI 데이터 레지스터

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
 - SPI 제어 레지스터 (SPI 동작을 제어하기 위한 레지스터)
 - 인터럽트 Enable, 동작 모드 설정, Clock 모드 설정들에 사용
 - 비트 7(SPIE:SPi Interrupt Enable)
 - 1로 세트 하면 SPI 전송 완료 인터럽트 개별 Enable
 - 비트 6(SPE: SPi Enable)
 - 1로 세트 하면 SPI 직렬 통신을 허용

7	6	5	4	3	2	1	0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
 - 비트 5(DORD:Data ORDer)
 - 1로 설정하면 LSB부터 전송하고 0으로 하면 MSB부터 전송
 - 비트 4(MSTR:Master/Slave Select)
 - 1로 설정하면 마스터로 동작하고 0으로 하면 슬레이브로 동작
 - 비트 3(CPOL:Clock POLarity)
 - 데이터 샘플링 동작이 수행되는 SCK 클록의 극성을 설정
 - 디폴트값인 0이면 Leading Edge의 경우는 상승 에지로, Trailing Edge의 경우에는 하강 에지로 선정
 - CPOL=1이면 Leading Edge의 경우는 하강 에지로, Trailing Edge의 경우에는 상승 에지로 선정

SPI(Serial Peripheral Interface)

- SPCR(SPI Control Register)
 - 비트 2(CPHA:Clock PHAse)
 - 데이터 샘플링 동작이 수행되는 SCK 클록의 위상을 설정
 - 디폴트값인 0이면 Leading Edge의 경우는 샘플링이 되고, Trailing Edge의 경우에는 셋업됨
 - CPHA=1이면 반대
 - 비트 1~0 (SPR1~0:SPi clock Rate select 1~0)
 - SPSR의 비트0(SPI2X비트)과 함께 SCK 클럭 신호의 주파수 분주비를 설정
 - SPSR의 비트0이 0인 상태에서 00이면 시스템 클록의 4분주, 01이면 16분주, 10이면 64분주, 11이면 128분주로 설정됨
 - SPSR의 비트0이 1인 상태에서는 주파수를 두배로 하여 분주비를 반감시키는데 00이면 시스템 클록의 2분주, 01이면 8분주, 10이면 32분주, 11이면 64분주로 설정됨

SPI(Serial Peripheral Interface)

- SPSR(SPI Status Register)
 - SPI 상태 레지스터(SPI의 동작 상태를 나타내는 레지스터)
 - 인터럽트 플래그와 쓰기 충돌 플래그 등이 있고, SCK 클럭 주파수를 2배로 설정하는 용도에도 사용

7	6	5	4	3	2	1	0
SPIF	WCOL	-	-	-	-	-	SPI2X

SPI(Serial Peripheral Interface)

- SPSR(SPI Status Register)
 - 비트 7(SPIF:SPI Interrupt Flag)
 - 전송이 완료되면 1로 세트 되면서 인터럽트가 발생
 - 마스터로 설정하고 SS 핀이 입력으로 설정되어 0레벨이 입력되면 SPCR의 비트4(MSTR)는 자동으로 클리어 되며 슬레이브 모드로 되고 SPIF가 세트 되면서 인터럽트가 발생
 - 비트 6(WCOL:Write COLision flag)
 - SPI를 통해 데이터를 전송하고 있는 동안에 SPDR레지스터를 기록하려고 하면 '1'로 세트됨
 - SPSR을 읽고 SPDR에 접근하는 경우에 SPIF와 함께 클리어 됨
 - 비트 0(SPI2X:SPI Double speed)
 - 마스터로 동작할 때 SCK클록 신호의 주파수를 2배로 설정

SPI(Serial Peripheral Interface)

- SPDR(SPI Data Register)
 - SPI 데이터 레지스터
 - SPI의 데이터 전송에 사용되는 레지스터

7	6	5	4	3	2	1	0
SPDR7	SPDR6	SPDR5	SPDR4	SPDR3	SPDR2	SPDR1	SPDR0

SPI(Serial Peripheral Interface)

- SPI 제어 프로그램
 - SPI 마스터 모드로 SPI를 초기화하고, 데이터를 전송하는 프로그램

```
void SPI_MasterInit(void)
{
    // MOSI와 SCK 핀을 출력으로, 다른 핀들을 입력으로 선언
    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
    // SPI Enable, Master 모드, clock rate fck/16
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
    // 전송 시작
    SPDR = cData;
    // 전송완료까지 대기
    while(!(SPSR & (1<<SPIF))) ;
}
```

SPI(Serial Peripheral Interface)

- SPI 제어 프로그램
 - SPI 슬레이브 모드로 SPI를 초기화하고, 데이터를 수신하는 프로그램

```
void SPI_SlaveInit(void)
{
    // MISO핀을 출력으로, 나머지는 모두 입력으로 선언
    DDR_SPI = (1<<DD_MISO);
    // SPI Enable
    SPCR = (1<<SPE);
}

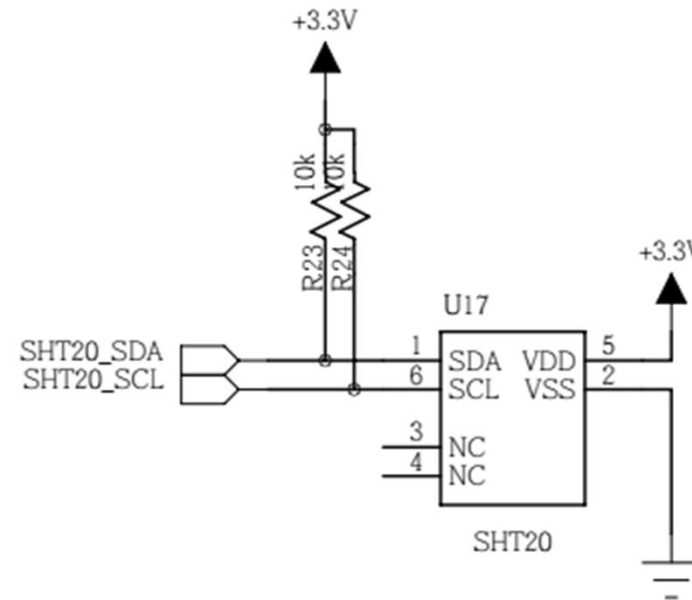
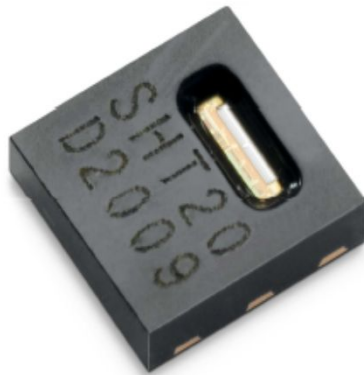
char SPI_SlaveReceive(void)
{
    // 수신 완료시까지 대기
    while(!(SPSR & (1<<SPIF)));
    // Data Register 값을 리턴
    return SPDR;
}
```

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 실습 개요
 - I2C로 제어되는 온습도 센서인 SHT20을 ATmega128A의 TWI 포트에 연결하고, 온도와 습도 정보를 읽어 들여 TEXT LCD에 표시
 - 제조사에서 제공하는 SHT20 칩의 TWI 인터페이스 관련 라이브러리 함수를 사용
- 실습 목표
 - TWI(I2C)의 동작 원리 이해 (레지스터 설정)
 - SHT20 온도센서 제어 방법 습득

실습 1 : TWI(I²C)로 온습도 센서 제어하기

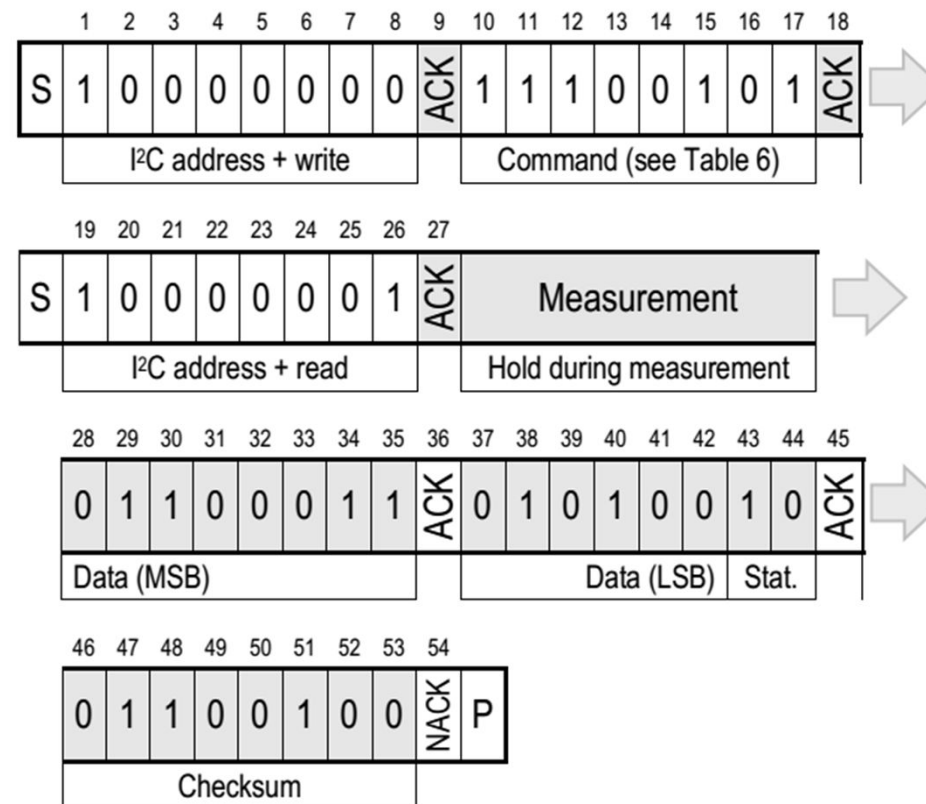
- 사용 모듈
 - 센서 모듈 중 온습도 센서 부분의 회로
 - SHT20
 - 상대습도와 상대 온도를 측정하는 칩, 이슬점(Dew Point) 측정이 가능
 - TWI(I2C) 인터페이스를 사용



실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 사용 모듈
 - SHT20 온습도 센서

SHT20의 (I2C)데이터 형식



실습 1 : TWI(I²C)로 온습도 센서 제어하기

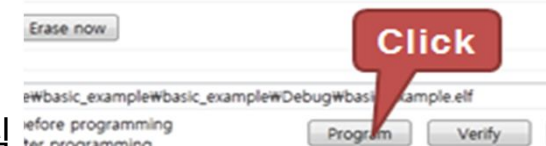
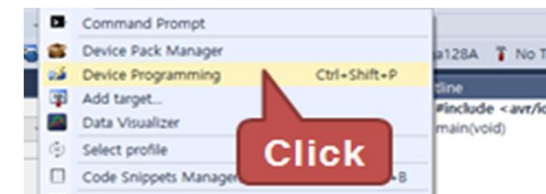
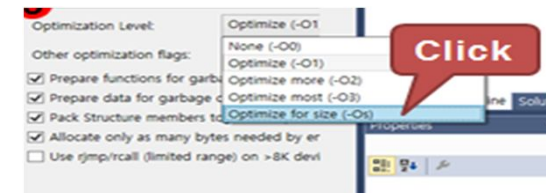
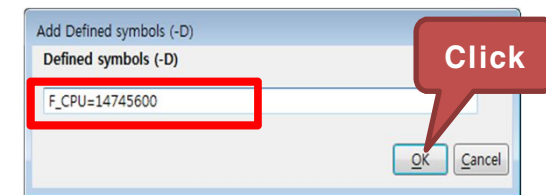
- 구동 프로그램 : 사전 지식
 - SHT2x.c 에서 제공하는 라이브러리 함수
 - int8_t SHT2x_CheckCrc(uint8_t *data, uint8_t nbrOfBytes, uint8_t checksum)
 - SHT20으로부터 수신된 데이터의 체크섬을 계산
 - int8_t SHT2x_ReadUserRegister(uint8_t *pRegisterValue)
 - SHT20의 유저 레지스터를 읽어옴
 - void SHT2x_WriteUserRegister(uint8_t *pRegisterValue)
 - SHT20의 유저 레지스터에 데이터를 씀
 - int8_t SHT2x_MeasureHM(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasuran d)
 - SHT20의 측정 데이터를 Hold Master 방식으로 읽어옴
(한번의 읽기시도)
 - int8_t SHT2x_MeasurePoll(etSHT2xMeasureType eSHT2xMeasureType, nt16 *pMeasuran d)
 - SHT20의 측정 데이터를 Hold Master 방식으로 읽어옴
(10ms 마다 20번의 읽기시도)

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램 : 사전 지식
 - SHT2x.c 에서 제공하는 라이브러리 함수
 - void SHT2x_SoftReset()
 - SHT20을 Software 리셋
 - float SHT2x_CalcRH(uint16_t u16sRH)
 - SHT20으로부터 읽어온 측정데이터를 실제 습도값으로 변환
 - float SHT2x_CalcTemperatureC(uint16_t u16sT)
 - SHT20으로부터 읽어온 측정데이터를 실제 온도값으로 변환
 - void SHT2x_GetSerialNumber(uint8_t *u8SerialNumber)
 - SHT20의 시리얼 넘버를 읽어옴
 - int8_t SHT2x_Init(void)
 - SHT20을 초기화

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 10_I2C_Example, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램
 - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h>    // delay 함수사용을 위한 헤더파일
#include "TWI_driver.h"   // TWI 사용을 위한 헤더 파일
#include "SHT2x.h"        // SHT2x 라이브러리 사용을 위한 헤더 파일
#include "lcd.h"          // Text LCD를 사용하기 위한 헤더 파일

void printf_2dot1(uint8_t sense,uint16_t sense_temp);
// 온도 및 습도를 LCD에 출력하는 함수
uint16_t temperatureC,humidityRH;
// 온도, 습도 값 측정에 사용되는 변수
```

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {
    uint8_t error = 0;    // 에러를 저장하는 변수
    nt16 sRH;             // 습도의 raw 데이터를 저장하는 변수
    nt16 sT;              // 온도의 raw 데이터를 저장하는 변수

    Init_TWI();           // TWI를 초기화
    lcdInit();            // Text LCD를 초기화
    SHT2x_Init();         // SHT 센서를 초기화
    while(1) {
        error |= SHT2x_MeasureHM(HUMIDITY, &sRH);    // 습도를 측정
        error |= SHT2x_MeasureHM(TEMP, &sT);         // 온도를 측정
        // 온도 습도를 계산, 소숫점 첫째자리까지 출력하기 위해 10을 곱함
        temperatureC = SHT2x_CalcTemperatureC(sT.u16)*10; // 온도를 계산
        humidityRH    = SHT2x_CalcRH(sRH.u16)*10;      // 습도를 계산
    }
}
```

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램
 - main.c 코드 작성

```
//...
if(error == SUCCESS) { // 에러없이 정상 측정일 경우
    lcdGotoXY(0,0); // 커서위치를 첫번째 줄, 첫번째 칸으로 이동
    printf_2dot1(TEMP,temperatureC); // 온도를 출력
    lcdGotoXY(0,1); // 커서위치를 두번째 줄, 첫번째 칸으로 이동
    printf_2dot1(HUMIDITY,humidityRH); // 습도를 출력
}
else { // 에러가 있을 경우
    lcdGotoXY(0,0); // 커서위치를 첫번째 줄, 첫번째 칸으로 이동
    lcdPrintData(" Temp: ---C",12); // 온도를 ---로 출력
    lcdGotoXY(0,1); // 커서위치를 두번째 줄, 첫번째 칸으로 이동
    lcdPrintData(" Humi: ---%",12); // 습도를 ---로 출력
}
_delay_ms(300); // 다음 측정을 위한 시간 지연(300ms)
}
```

실습 1 : TWI(I²C)로 온습도 센서 제어하기

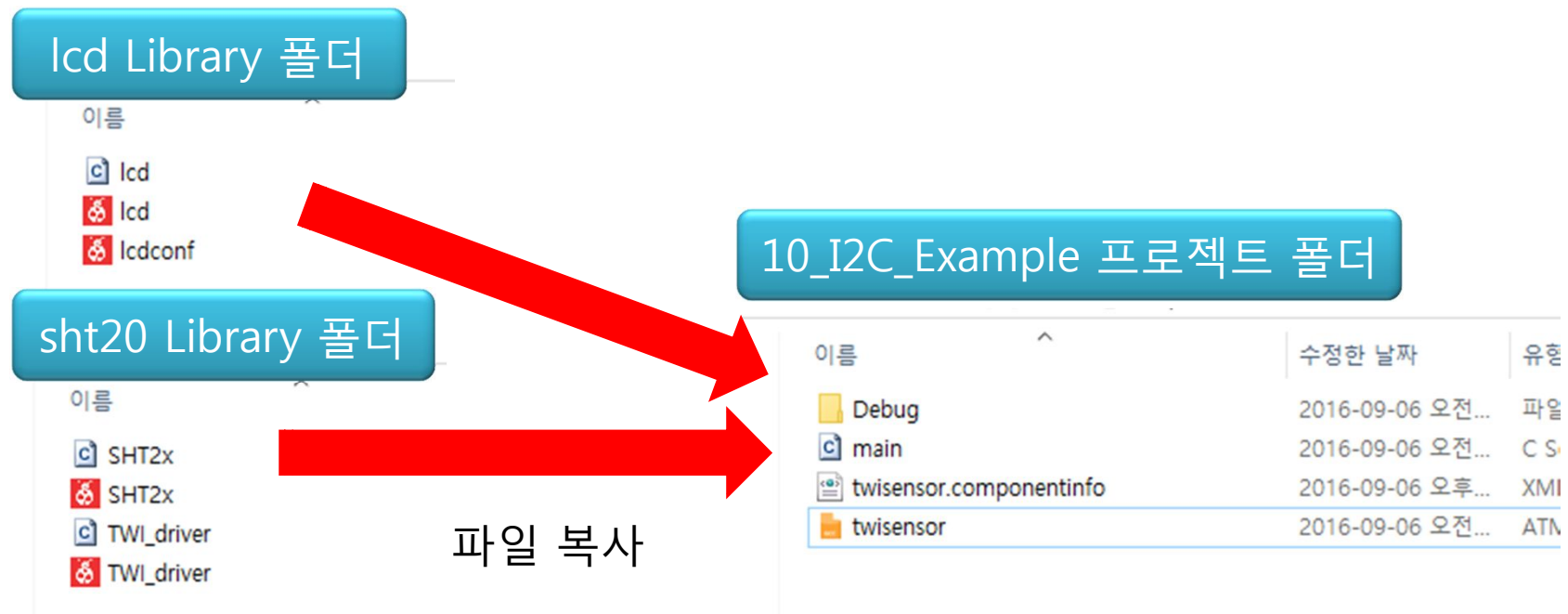
- 구동 프로그램
 - main.c 코드 작성

```
void printf_2dot1(uint8_t sense,uint16_t sense_temp) {
    uint8_t s100,s10;
    if(sense == TEMP) lcdPrintData(" Temp: ",7); // 온도 출력
    else if(sense == HUMIDITY) lcdPrintData(" Humi: ",7); // 습도 출력
    s100 = sense_temp/100;           // 100의 자리 추출
    if(s100> 0) lcdDataWrite(s100+'0'); // 100의 자리 값이 있으면 출력
    else lcdPrintData(" ",1);         // 100의 자리 값이 없으면 빈칸 출력
    s10 = sense_temp%100;             // 100의 자리를 제외한 나머지 추출
    lcdDataWrite((s10/10)+'0');       // 10의 자리 추출하여 출력
    lcdPrintData(".",1);              // 소숫점 출력
    lcdDataWrite((s10%10)+'0');       // 1의 자리 추출하여 출력

    if(sense == TEMP) lcdDataWrite('C');           // 온도 단위 출력
    else if(sense == HUMIDITY) lcdDataWrite('%');  // 습도 단위 출력
}
```

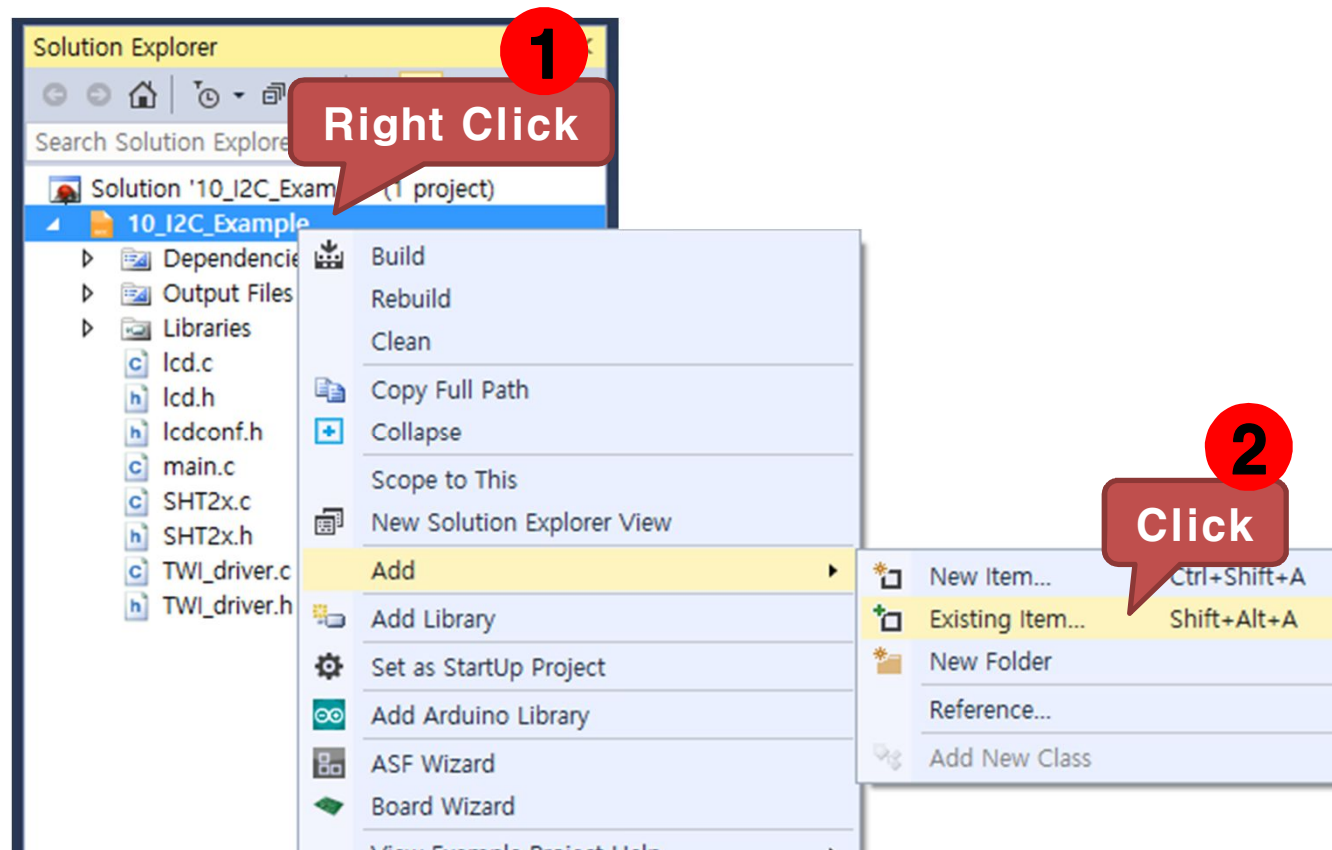

실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - SHT2x.c, SHT2x.h, TWI_driver.c, TWI_driver.h 파일은 "AVR_Example\library\sht20" 폴더에 존재
 - lcd.c, lcd.h, lcdconf.h 파일은 "AVR_Example\library\lcd" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



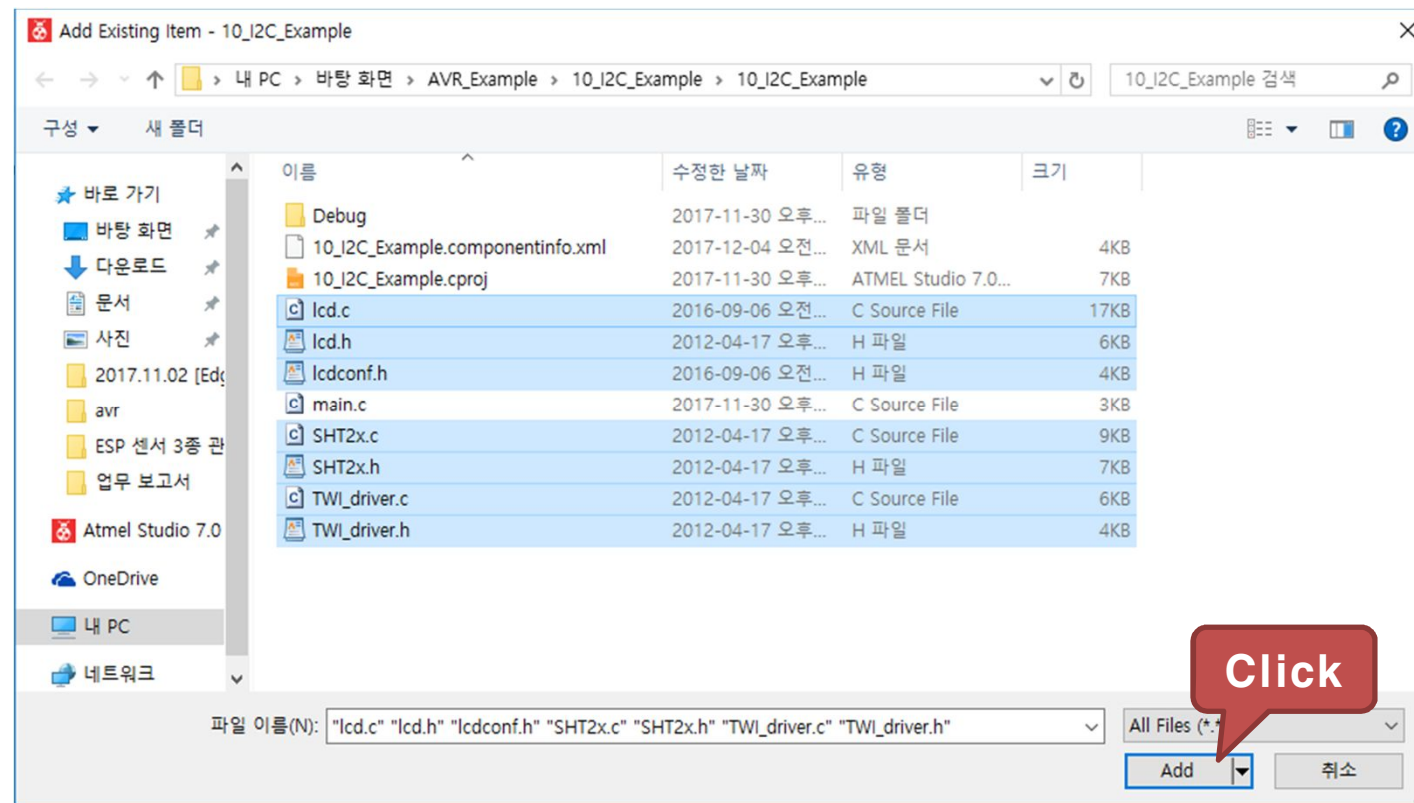
실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



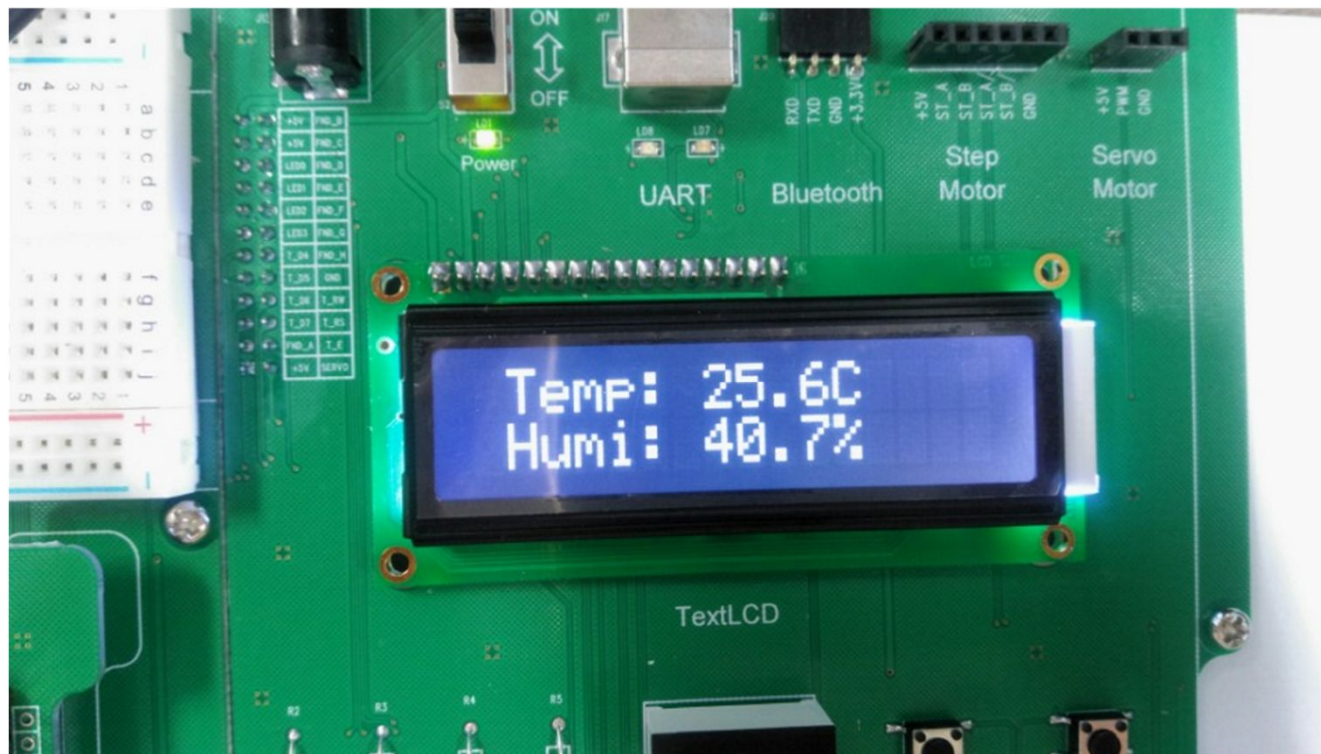
실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 일곱개의 파일을 추가
 - Ctrl 키를 누르고 파일 일곱개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



실습 1 : TWI(I²C)로 온습도 센서 제어하기

- 실행 결과
 - 온습도 센서로부터 온도 값 센서 값을 TextLCD에 출력
 - 온습도 센서에 입김을 불거나, 손을 갖다대가면서 온도와 습도값이 변화하는지 확인

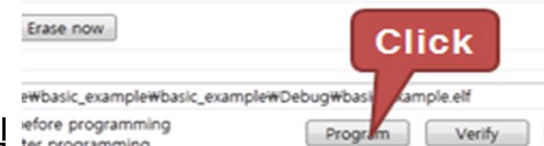
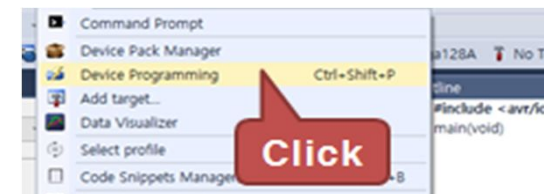
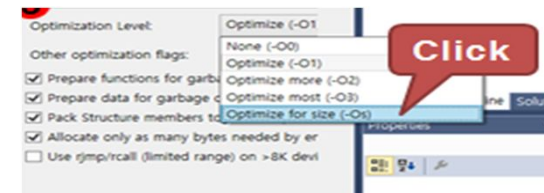
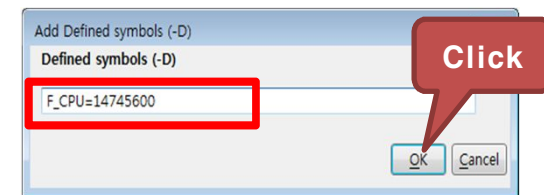


응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 실습 개요
 - I2C로 제어되는 온습도 센서인 SHT20을 ATmega128A의 TWI 포트에 연결하고, 온도와 습도 정보를 읽어 들여 UART로 전송
 - 제조사에서 제공하는 SHT20 칩의 TWI 인터페이스 관련 라이브러리 함수를 사용
- 실습 목표
 - TWI(I2C)의 동작 원리 이해 (레지스터 설정)
 - SHT20 온도센서 제어 방법 습득

응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 10_I2C_Application, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
#include <avr/io.h>           //AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>        //delay 함수사용을 위한 헤더파일
#include "TWI_driver.h"       //TWI 사용을 위한 헤더 파일
#include "SHT2x.h"            //SHT2x 라이브러리 사용을 위한 헤더 파일

unsigned int temperatureC, humidityRH;    //온도, 습도 값 측정에 사용
되는 변수
unsigned int timer0Cnt = 0;
volatile unsigned char Read_flag = 1;

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0); // 전송준비가 될때까지 대기
    UDR0 = data;                 // 데이터를 UDR0에 쓰면 전송된다
    UCSR0A |= 0x20;
}
```


응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
void putch_Str(char *str)          // 문자열을 출력하는 함수
{
    unsigned char i=0;
    while(str[i]!='\0')
        putch(str[i++]);          //문자열을 출력
}

int main(void)
{
    unsigned char error = 0;        //에러를 저장하는 변수

    nt16 sRH;                       //습도의 raw 데이터를 저장하는 변수
    nt16 sT;                        //온도의 raw 데이터를 저장하는 변수

    Init_TWI();                     //TWI를 초기화 한다
    SHT2x_Init();                  //SHT 센서를 초기화 한다
```


응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
//...
TCCR0 = 0x07;
TCNT0 = 112; // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
TIMSK = 0x01;
TIFR |= 1 << TOV0;

DDRE = 0x02; // Rx(입력 0), Tx(출력, 1)
UCSR0A = 0x00;
UCSR0B = 0x18; // Rx, Tx enable
UCSR0C = 0x06; // 비동기 방식, No Parity bit, 1 Stop bit

UBRR0H = 0x00;
UBRR0L = 0x07; // 115200 bps

sei();
```

응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
while(1)
{
    if(Read_flag == 1) // 5초마다 1번씩 온습도 값을 읽어 PC로 전송
    {
        error |= SHT2x_MeasureHM(HUMIDITY, &sRH); //습도를 측정한다.
        error |= SHT2x_MeasureHM(TEMP, &sT);      //온도를 측정한다.
        //온도 습도를 계산, 소숫점 첫째자리까지 출력하기 위해 10을 곱한다.
        temperatureC = SHT2x_CalcTemperatureC(sT.u16)*10;
        //온도를 계산한다.
        humidityRH    = SHT2x_CalcRH(sRH.u16)*10;
        //습도를 계산한다.

        if(error == SUCCESS) //에러없이 정상 측정 되었으면
        {
            putch_Str("\n\r Temp: ");
            // 온도 출력
            // 100의 자리 값 확인
        }
    }
}
```

응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
if(temperatureC/100) // 100의 자리가 0이 아닐 경우
{
    // 100의 자리 출력
    putchar(temperatureC/100 + '0');
}
putchar((temperatureC/10)%10 + '0');
putchar('.');
putchar(temperatureC%10 + '0');
putchar_Str("[C], Humi : ");
// 습도 출력
// 100의 자리 값 확인
if(humidityRH/100) // 100의 자리가 0이 아닐 경우
{
    // 100의 자리 출력
    putchar(humidityRH/100 + '0');
}
```

응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
        putchar((humidityRH/10)%10 + '0');
        putchar('.');
        putchar(humidityRH%10 + '0');
        putchar_Str("[%]");
    }
    else //에러가 있을 경우
    {
        //온도 ---로, 습도를 ---로 출력한다.
        putchar_Str("\n\r Temp: ---C, Humi: ---%");
    }
    Read_flag = 0;
}
}
```

응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
SIGNAL(TIMER0_OVF_vect){
    cli();
    TCNT0 = 112; // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
    timer0Cnt++; // timer0Cnt 변수를 1 증가 시킨다.

    // 0.01s * 500 = 5s //5초를 얻기 위한 카운트 횟수
    if(timer0Cnt == 500){
        Read_flag = 1; // 온도와 습도 측정 및 PC로 전송하도록 1로 설정
        timer0Cnt = 0;
    }
    sei();
}
```

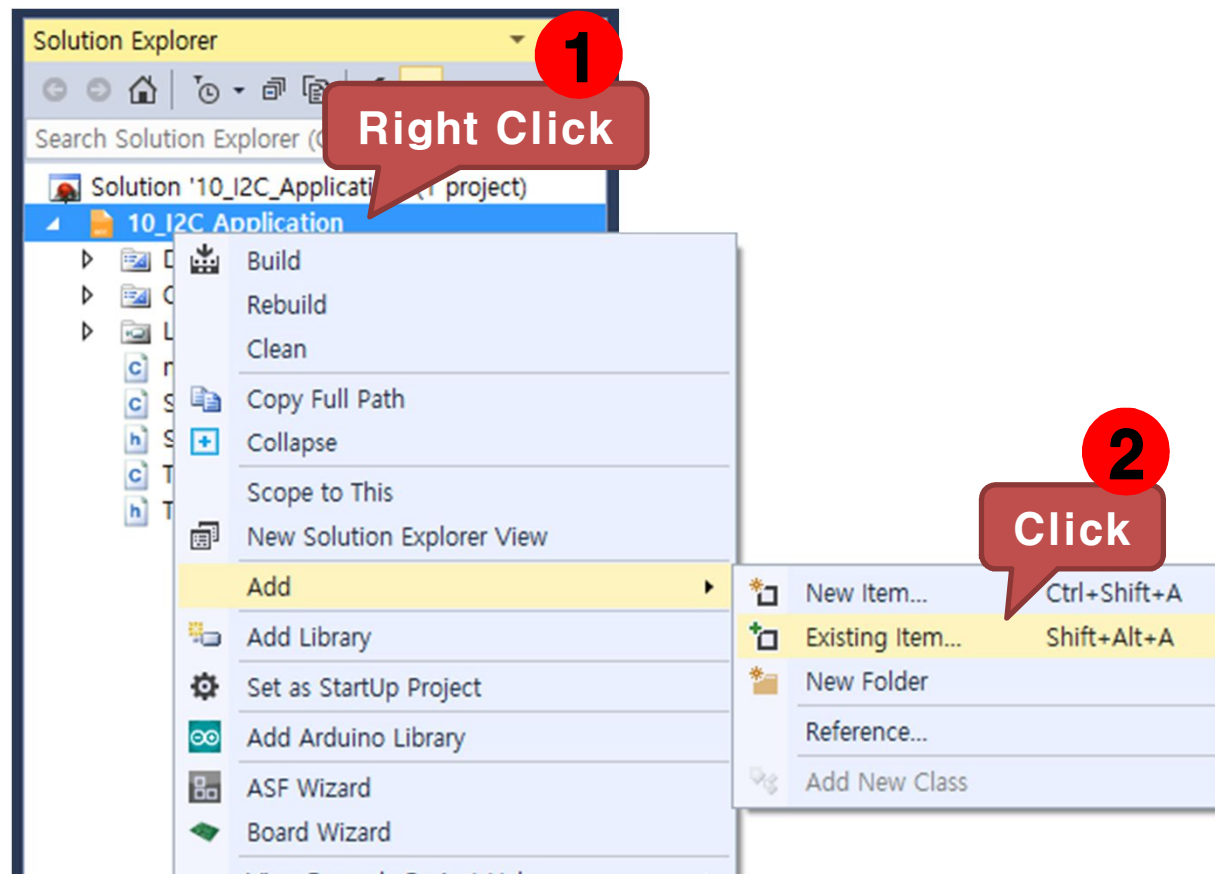
응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - SHT2x.c, SHT2x.h, TWI_driver.c, TWI_driver.h 파일은 "AVR_Example\library\sht20" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



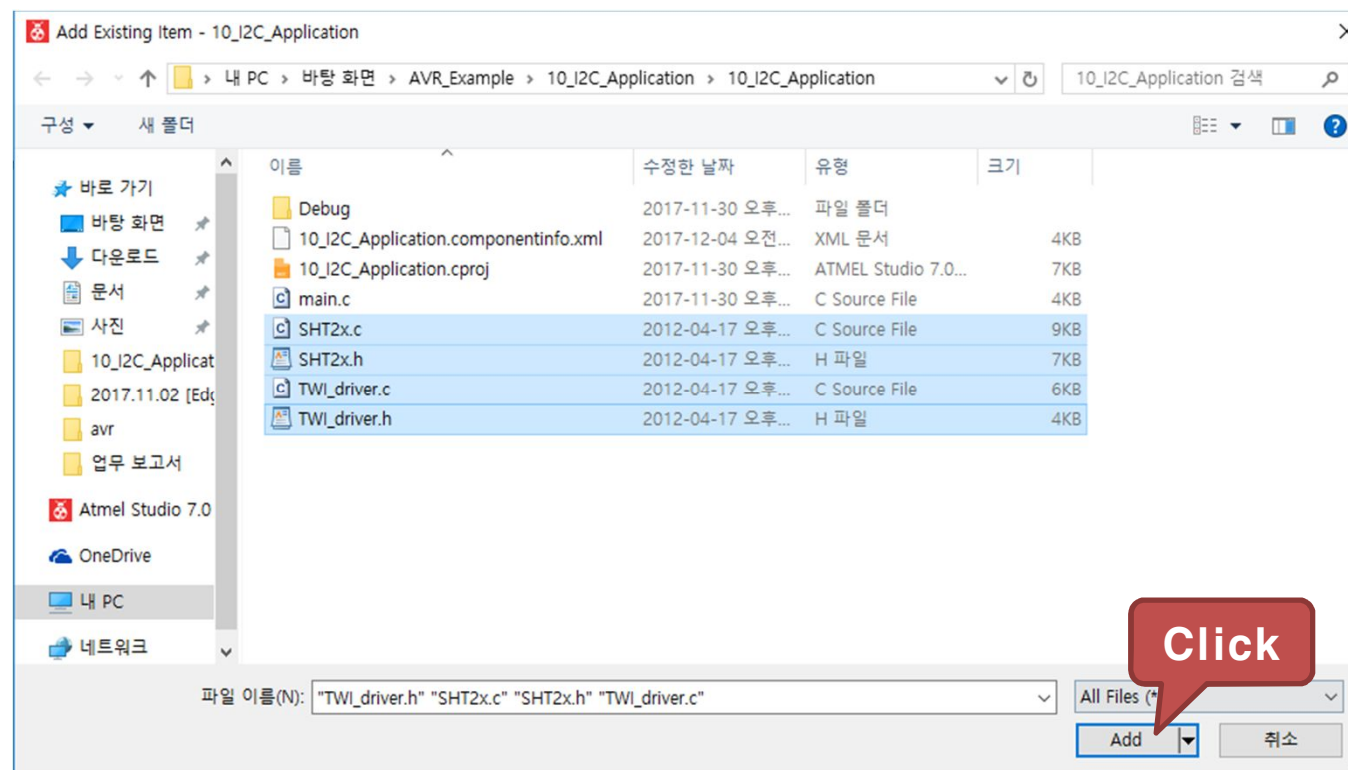
응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



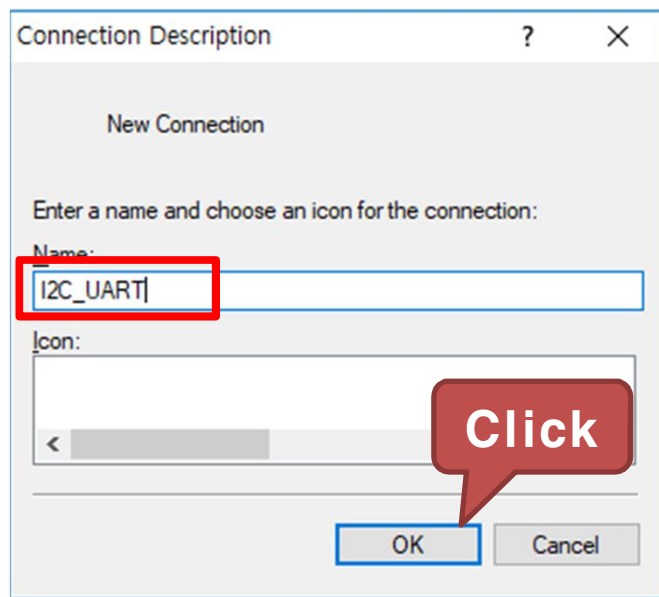
응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 일곱개의 파일을 추가
 - Ctrl 키를 누르고 파일 일곱개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 하이퍼터미널 실행
 - 연결 설명 창의 이름 란에 "I2C_UART" 라고 확인을 클릭
 - 연결 대상 창의 연결에 사용할 모뎀 번호를 설정하고 확인을 클릭



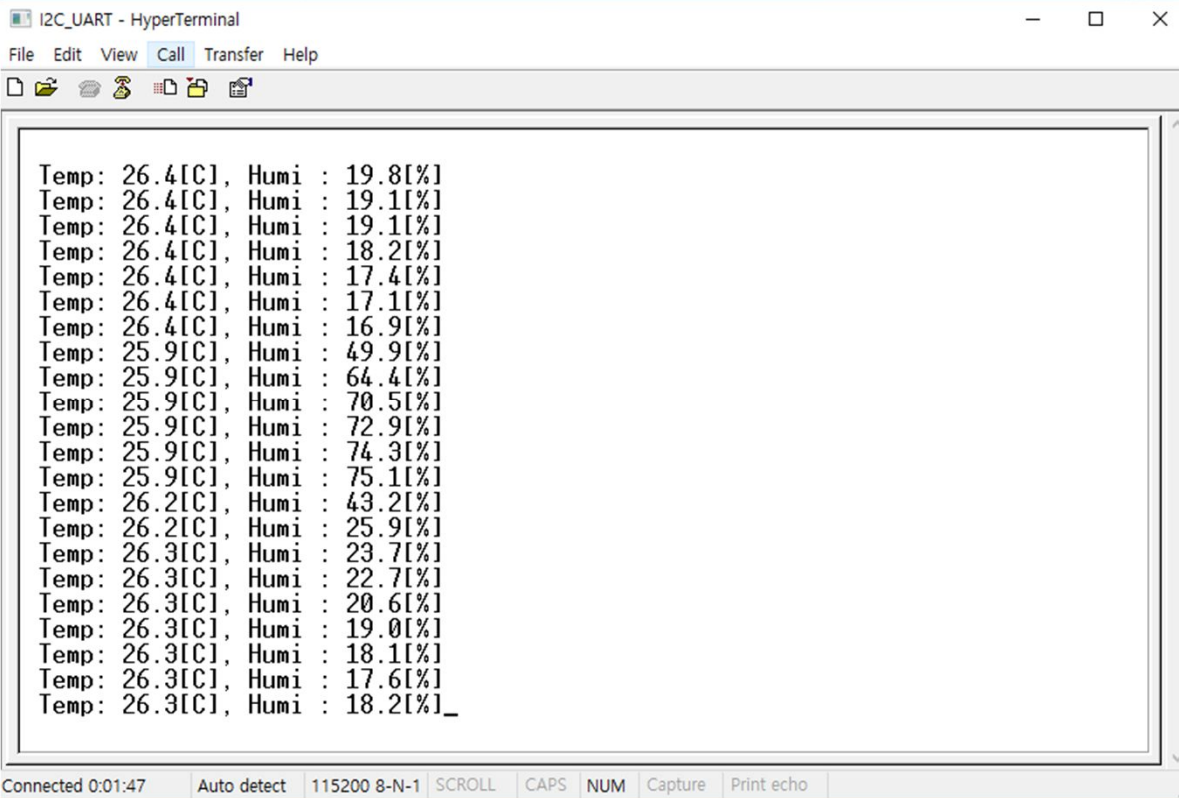
응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 하이퍼터미널 실행
 - 통신 설정을 그림과 같이하고, 확인을 클릭



응용 1: 타이머를 이용하여 측정한 온습도를 PC에 출력하기

- 실행 결과
 - 온습도 센서로부터 온도 값 센서 값을 하이퍼터미널 창에 출력
 - 온습도 센서에 입김을 불거나, 손을 갖다대가면서 온도와 습도값이 변화하는지 확인



```
I2C_UART - HyperTerminal
File Edit View Call Transfer Help
Temp: 26.4[C], Humi : 19.8[%]
Temp: 26.4[C], Humi : 19.1[%]
Temp: 26.4[C], Humi : 19.1[%]
Temp: 26.4[C], Humi : 18.2[%]
Temp: 26.4[C], Humi : 17.4[%]
Temp: 26.4[C], Humi : 17.1[%]
Temp: 26.4[C], Humi : 16.9[%]
Temp: 25.9[C], Humi : 49.9[%]
Temp: 25.9[C], Humi : 64.4[%]
Temp: 25.9[C], Humi : 70.5[%]
Temp: 25.9[C], Humi : 72.9[%]
Temp: 25.9[C], Humi : 74.3[%]
Temp: 25.9[C], Humi : 75.1[%]
Temp: 26.2[C], Humi : 43.2[%]
Temp: 26.2[C], Humi : 25.9[%]
Temp: 26.3[C], Humi : 23.7[%]
Temp: 26.3[C], Humi : 22.7[%]
Temp: 26.3[C], Humi : 20.6[%]
Temp: 26.3[C], Humi : 19.0[%]
Temp: 26.3[C], Humi : 18.1[%]
Temp: 26.3[C], Humi : 17.6[%]
Temp: 26.3[C], Humi : 18.2[%]_
Connected 0:01:47 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

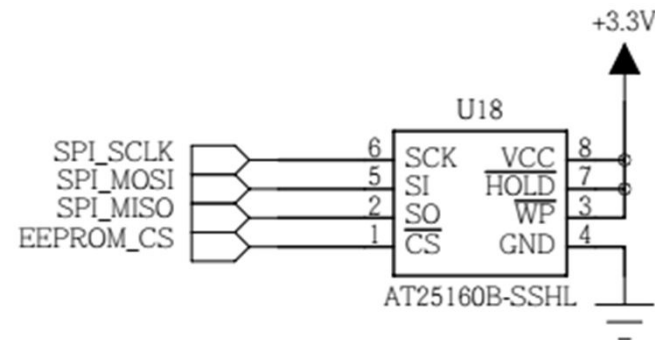
실습 2 : SPI로 EEPROM 붙이기

- 실습 개요
 - 정해진 문자열을 SPI 인터페이스를 이용하여 Serial EEPROM Memory에 저장했다가, 이를 다시 꺼내어 TEXT LCD에 표시
 - SPI로 제어되는 EEPROM Memory인 AT25160B칩을 메모리로 사용
 - AT25160B 칩의 SPI 인터페이스 관련 라이브러리 함수는 at25160.c에 포함되어 있음

- 실습 목표
 - SPI 인터페이스 동작 원리 이해 (레지스터 설정)
 - ATmega128A의 SPI 포트에 대한 프로그램 방법 습득
 - Serial EEPROM Memory(AT25160B) 동작 원리 이해

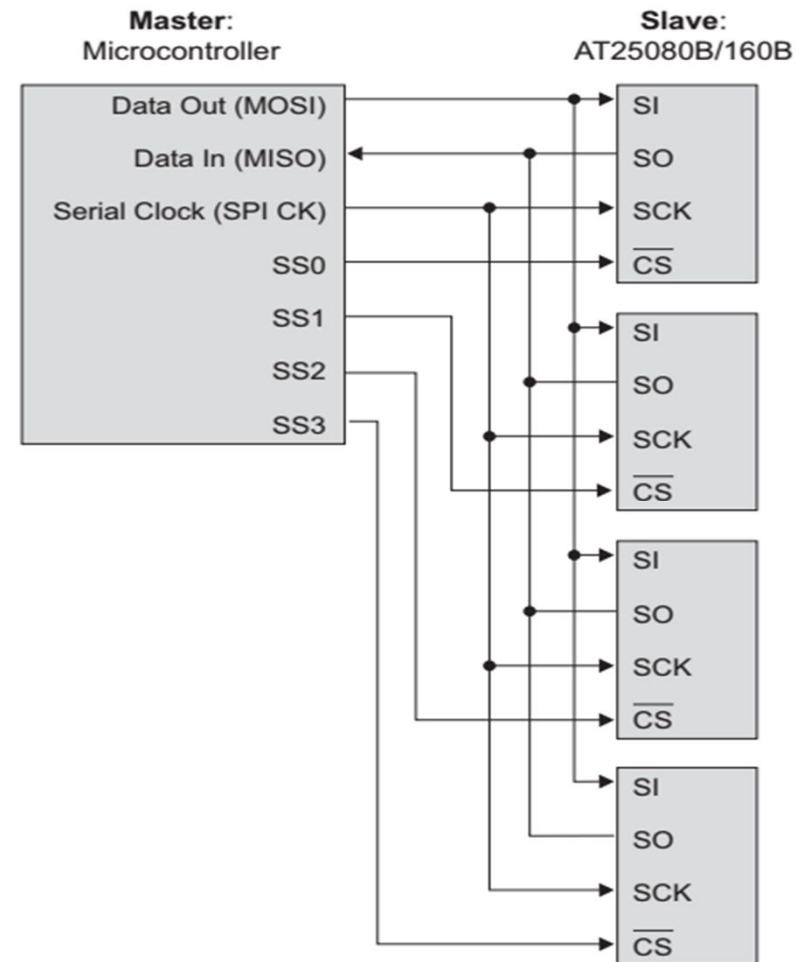
실습 2 : SPI로 EEPROM 붙이기

- 사용 모듈
 - 메모리 모듈의 SPI EEPROM 메모리 부분 회로
 - AT25160B
 - SPI를 지원하는 16K(2,048 x 8) 비트 용량의 Serial EEPROM memory
 - 32-Byte크기의 페이지를 가지고 있으며, Write-Protect 기능을 가지고 있음



실습 2 : SPI로 EEPROM 붙이기

- 사용 모듈
 - AT25160B를 ATmega128A에 연결하는 방법



실습 2 : SPI로 EEPROM 붙이기

- 사용 모듈
 - AT25160B 핀 설명

Pin Name	Function
\overline{CS}	Chip Select
GND	Ground
\overline{HOLD}	Suspends Serial Input
SCK	Serial Data Clock
SO	Serial Data Output
SI	Serial Data Input
\overline{WP}	Write Protect
Vcc	Power Supply

실습 2 : SPI로 EEPROM 붙이기

- 사용 모듈
 - AT25160B 8비트 명령어 세트

Instruction Name	Instruction Format	Operation
WREN	0000 X 110	Ser Write Enable Latch
WRDI	0000 X 100	Reset Write Enable Latch
RDSR	0000 X 101	Read Status Register
WRSR	0000 X 001	Write Status Register
Read	0000 X 011	Read Data from Memory Array
Write	0000 X 010	Write Data to Memoy Array

실습 2 : SPI로 EEPROM 붙이기

- AT25160B Serial EEPROM Memory 제어
 - AT25160B 칩의 임의의 메모리 주소로부터 데이터 읽어 오기
- 1. CS를 low로 만듦(Chip-Select active)
- 2. SPDR에 READ(0x03)명령을 넣어 AT25160B로 전송한다. SPSR레지스터의 SPIF(SPi Interrupt Flag) 비트를 관찰하면서 1로 세트될 때까지 대기
- 3. SPDR에 0x00을 넣고(최상위 어드레스), SPSR의 SPIF가 1로 세트될때까지 대기
- 4. SPDR에 상위 8비트 어드레스를 넣고, SPSR의 SPIF가 1로 세트될때까지 대기
- 5. SPDR에 하위 8비트 어드레스를 넣고, SPSR의 SPIF가 1로 세트될때까지 대기
- 6. SPDR에 0을 넣고(데이터 수신 대기), SPSR의 SPIF가 1로 세트될때까지 대기
- 7. SPDR에서 데이터를 읽어옴
- 8. CS를 High로 만듦(Chip-Select inactive)

실습 2 : SPI로 EEPROM 붙이기

- AT25160B Serial EEPROM Memory 제어
 - AT25160B 칩의 임의의 메모리 주소에 데이터를 써넣기(Program)
 1. CS를 low로 만듦(Chip-Select active)
 2. SPDR에 WREN(0x06)명령을 넣음으로써, AT25160B로 전송, SPSR레지스터의 SPIF(SPi Interrupt Flag) 비트를 관찰하면서 1로 세트될 때까지 대기
 3. SPDR에 PROGRAM(0x00)을 넣고(쓰기선언), SPSR의 SPIF가 1로 세트될때까지 대기
 4. SPDR에 0x00을 넣고(최상위 어드레스), SPSR의 SPIF가 1로 세트될때까지 대기
 5. SPDR에 상위 8비트 어드레스를 넣고 SPSR의 SPIF가 1로 세트될때까지 대기
 6. SPDR에 하위 8비트 어드레스를 넣고 SPSR의 SPIF가 1로 세트될때까지 대기
 7. SPDR에 데이터를 넣고(데이터 송신), SPSR의 SPIF가 1로 세트될 때 까지 대기
 8. SPDR에 RDSR(0x05)을 넣고(Status Register 읽기 선언), SPSR의 SPIF가 1로 세트될 때까지 대기
 9. SPDR에 0을 넣고(레지스터 내용 수신대기), SPSR의 SPIF가 1로 세트될 때까지 대기
 10. SPDR에서 데이터를 읽어옴(Status Register 내용)
 11. Status 레지스터의 RDY(0번 비트)가 1로 세트 될때까지 ⑧~⑩ 반복
 12. CS를 High로 만듦(Chip-Select inactive)

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : 사전 지식(SPI 인터페이스 제어)
 - SPI 초기화
 - SPI 신호로 사용할 핀들을 설정하고, 입출력을 결정
 - ATmega128A가 마스터로 사용될 경우, MISO는 입력으로, 나머지 MOSI, SCK, /SS는 출력으로 설정
 - 동작모드와 클럭 모드를 결정
 - SPCR레지스터의 MSTR비트를 세팅하여, Master/ Slave모드를 결정
 - 여기서는 Master모드를 사용
 - DORD 비트를 세팅하여, 데이터 전송순서를 결정
 - 여기서는 AT25160B에 적용하기 위해, MSB 먼저 보내도록 설정
 - CPOL과 CPHA 비트를 세팅, 클럭의 극성(Polarity)와 위상(Phase)를 결정
 - 여기서는 그냥 디폴트('0')로 설정
 - 클럭의 주파수를 결정
 - SPCR의 SPR(1:0)과 SPSP의 SPI2X를 세팅
 - 여기서는 SPI2X='0', SPR(1:0)="00"로 하여, 4분주($14.7456\text{MHz}/4 = 3.6864\text{MHz}$)로 설정
 - SPCR의 SPE 비트를 '1'로 세팅하여 SPI를 Enable

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : 사전 지식
 - AT25160B칩을 위한 라이브러리 함수
 - SPI_Init()
 - SPI를 초기화
 - at25160_Write_Byte(unsigned int addr, unsigned char data)
 - 임의의 주소에 1바이트 데이터를 써넣고, 인수로 주소와 써넣을 1바이트 데이터를 받음
 - unsigned char at25160_Read_Byte(unsigned int addr)
 - 인수로 주어진 주소의 1 바이트 데이터를 읽어냄
 - void at25160_Write_Array(unsigned int addr, unsigned char* BPdata, unsigned char size)
 - 인수로 주어진 주소로부터 size만큼 BPData에 존재하는 데이터를 써넣음
 - void at25160_Read_Array(unsigned int addr, unsigned char* BPbuf, unsigned char size)
 - 인수로 주어진 주소로부터 size만큼의 데이터를 읽어 BPbuff에 넘겨줌

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : AT25160B칩을 위한 라이브러리 함수
 - SPI 초기화

```
void SPI_Init()
{
    DDRB |= 0x07;           // 포트를 설정, miso 입력설정
    PORTB |= 0x01;          // ~cs = 1

    SPCR = 0x50;
    SPSR = 0x00;
}
```

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : AT25160B칩을 위한 라이브러리 함수
 - SPI EEPROM 칩 쓰기 활성화

```
#define AT25160_CS_LOW          (PORTB  &= 0xFE)
#define AT25160_CS_HIGH        (PORTB  |= 0x01)

void at25fxx_WREN()
{
    AT25160_CS_LOW;           // 칩선택
    SPDR = WREN;               // wren명령
    while((SPSR & 0x80)!=0x00); // 데이터 전송완료?
    AT25160_CS_HIGH;          // cs high가 되어야 쓰기 실행
}
```

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : AT25160B칩을 위한 라이브러리 함수
 - SPI EEPROM 칩 명령 대기

```
void at25160_Ready()  
{  
    unsigned char data;  
    do{  
        AT25160_CS_LOW;  
        SPDR = RDSR;  
        while((SPSR & 0x80)==0x00);  
        SPDR=0;  
        while((SPSR & 0x80)==0x00);  
        data = SPDR;  
        SPDR = 0;  
        while((SPSR & 0x80)==0x00);  
        AT25160_CS_HIGH;  
    } while((data & 1<<(RDY)));  
    //레지스터를 읽어와 준비상태가 되어있을까지 루프  
}
```

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : AT25160B칩을 위한 라이브러리 함수
 - SPI EEPROM 칩 1바이트 읽기

```
unsigned char at25160_Read_Byte(unsigned int addr)
{
    unsigned char data=0;
    AT25160_CS_LOW;
    SPDR = READ;                // 읽기선언
    while((SPSR & 0x80)==0x00);
    SPDR = ((addr>>8) & 0xff);   // 주소
    while((SPSR & 0x80)==0x00);
    SPDR = ((addr) & 0xff);      // 주소
    while((SPSR & 0x80)==0x00);
    SPDR = 0;
    while((SPSR & 0x80)==0x00);
    data = SPDR;                // 데이터
    AT25160_CS_HIGH;
    return data;
}
```


실습 2 : SPI로 EEPROM 붙이기

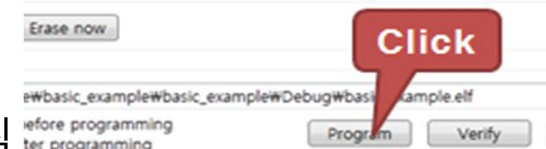
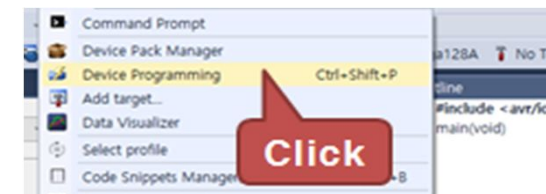
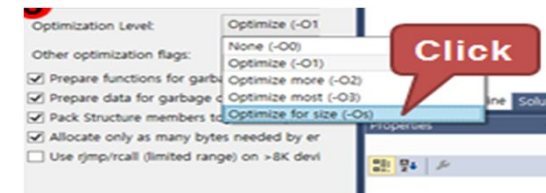
- 구동 프로그램 : AT25160B칩을 위한 라이브러리 함수
 - SPI EEPROM 칩 1바이트 쓰기

```
void at25160_Write_Byte(unsigned int addr, unsigned char data)
{
    at25160_WREN();
    AT25160_CS_LOW;
    SPDR = PROGRAM;          // 쓰기 선언
    while((SPSR & 0x80)==0x00);
    SPDR = ((addr>>8) & 0xff); // 주소
    while((SPSR & 0x80)==0x00);
    SPDR = ((addr) & 0xff);    // 주소
    while((SPSR & 0x80)==0x00);
    SPDR = data;              //데이터
    while((SPSR & 0x80)==0x00);

    AT25160_CS_HIGH;         // cs high가 되어야 입력한 내용이 적용
    at25160_Ready();
}
```

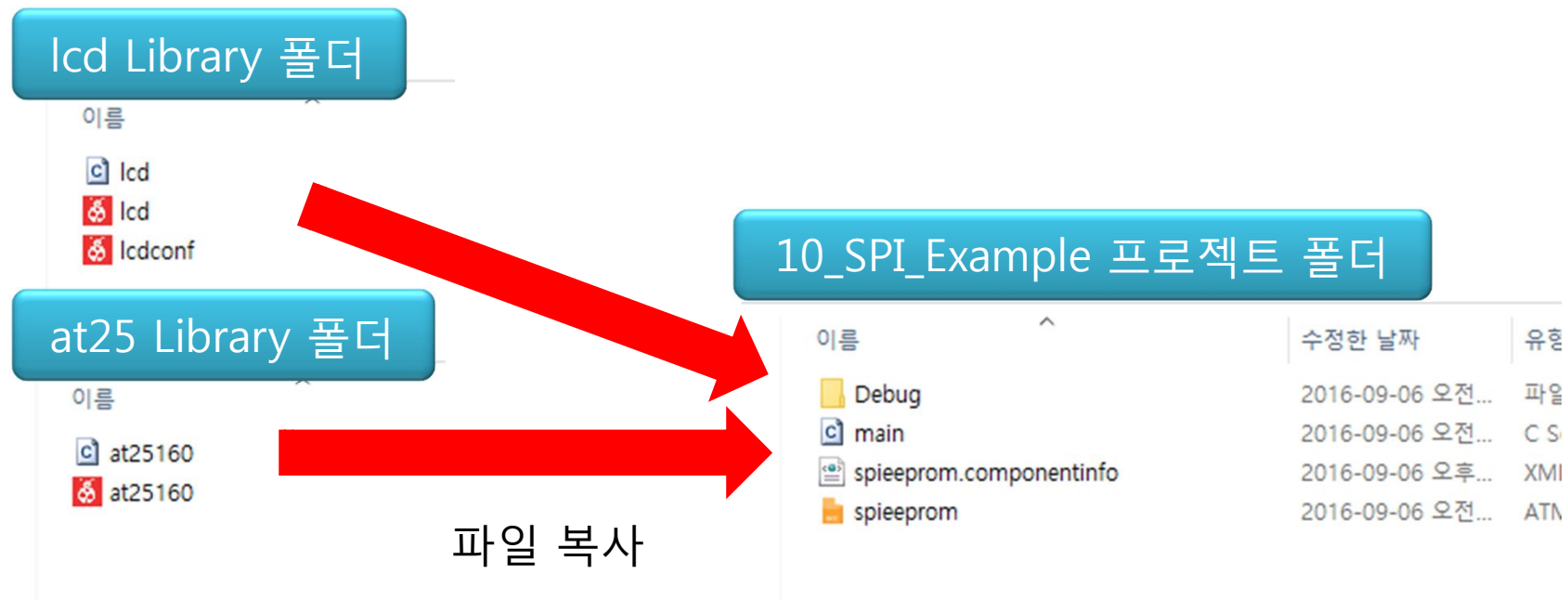
실습 2 : SPI로 EEPROM 붙이기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 10_SPI_Example, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



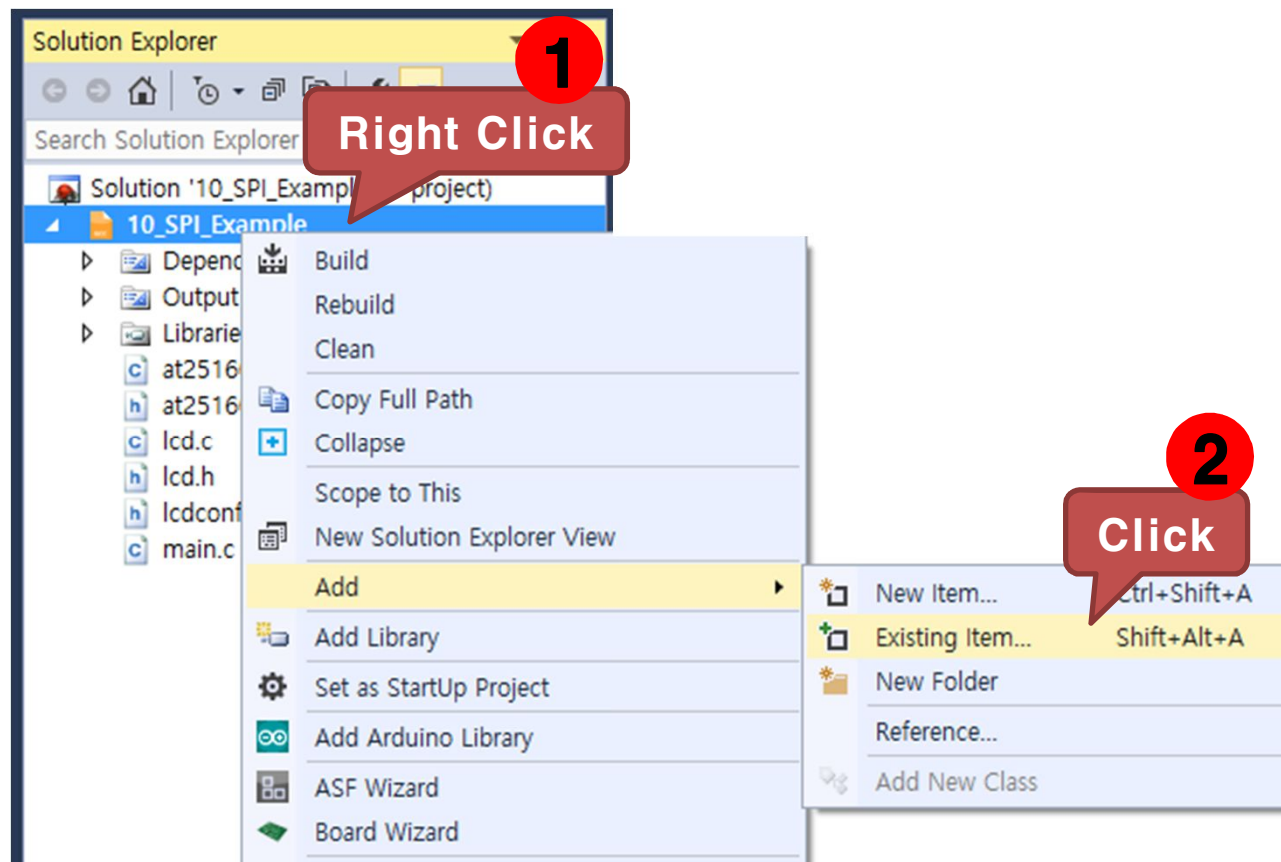
실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - at25160.c, at25160.h 파일은 "AVR_Example\library\wat25" 폴더에 존재
 - lcd.c, lcd.h, lcdconf.h 파일은 "AVR_Example\library\lcd" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



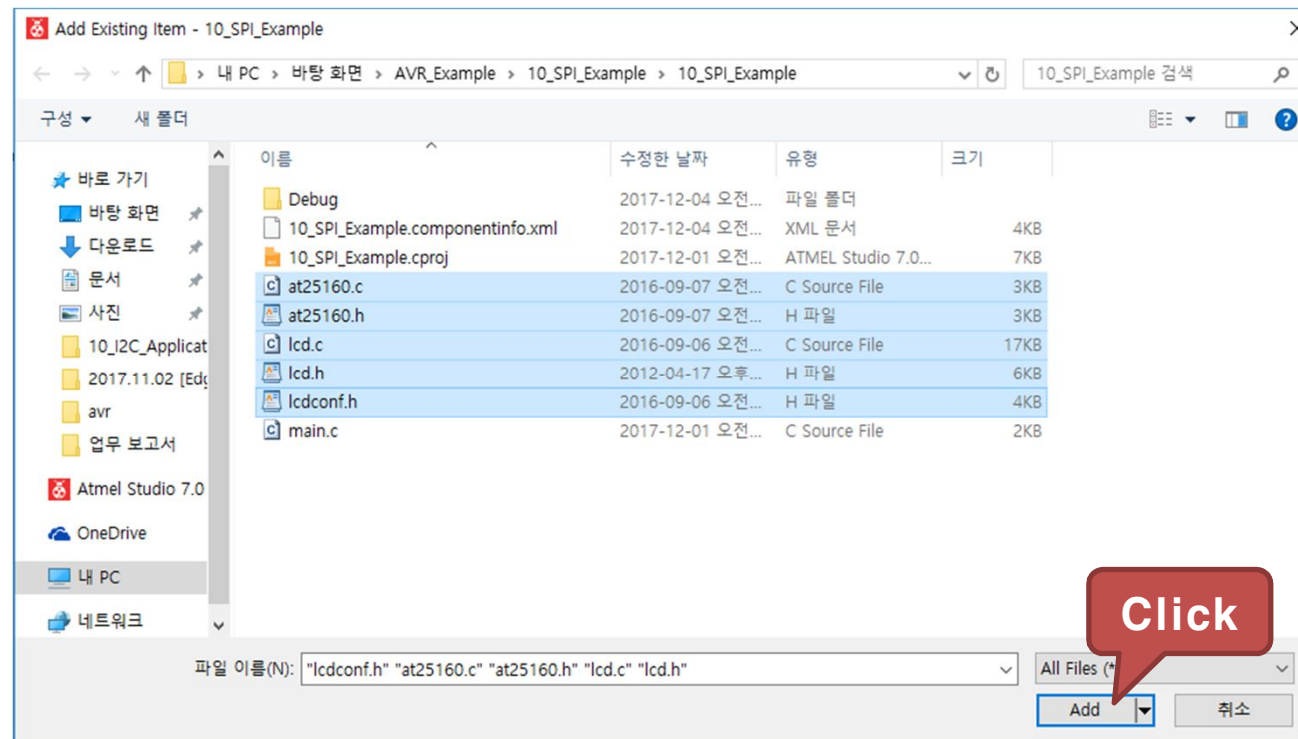
실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 다섯개의 파일을 추가
 - Ctrl 키를 누르고 파일 다섯개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램
 - main.c 코드 작성

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h>  // delay 함수사용을 위한 헤더파일
#include "at25160.h"     // AT25160B 라이브러리 사용을 위한 헤더 파일
#include "lcd.h"         // Text LCD를 사용하기 위한 헤더 파일

#define ARRAY_SIZE(array) (sizeof(array)/sizeof(array[0]))

unsigned char msg1[]="Welcome!!";
unsigned char msg2[]="edgeiLAB-World!!";
unsigned char msg3[]="SPI-Flash Exam";

int main(void) {
    unsigned char i=0;
    unsigned char buf1[20]={0};
    unsigned char buf2[20]={0};
    unsigned char buf3[20]={0};
```

실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {
    //...
    SPI_Init();
    lcdInit();
    at25160_Write_Arry(0x0100, msg1, ARRAY_SIZE(msg1));
    at25160_Write_Arry(0x0200, msg2, ARRAY_SIZE(msg2));
    at25160_Write_Arry(0x0300, msg3, ARRAY_SIZE(msg3));
    // “Welcome”, “edgeiLAB-World!!”, “SPI-Flash Exam” 각각 저장
    at25160_Read_Arry(0x0100, buf1, ARRAY_SIZE(msg1));
    at25160_Read_Arry(0x0200, buf2, ARRAY_SIZE(msg2));
    at25160_Read_Arry(0x0300, buf3, ARRAY_SIZE(msg3));
    // 플래쉬에 저장한 문자열을 읽어와서 각 배열에 저장
    for(lcdGotoXY(0, 0); i < ARRAY_SIZE(msg1) - 1; i++){
        lcdDataWrite(buf1[i]);          // 읽어들이는 문자열 출력
        _delay_ms(100);
    }
}
```

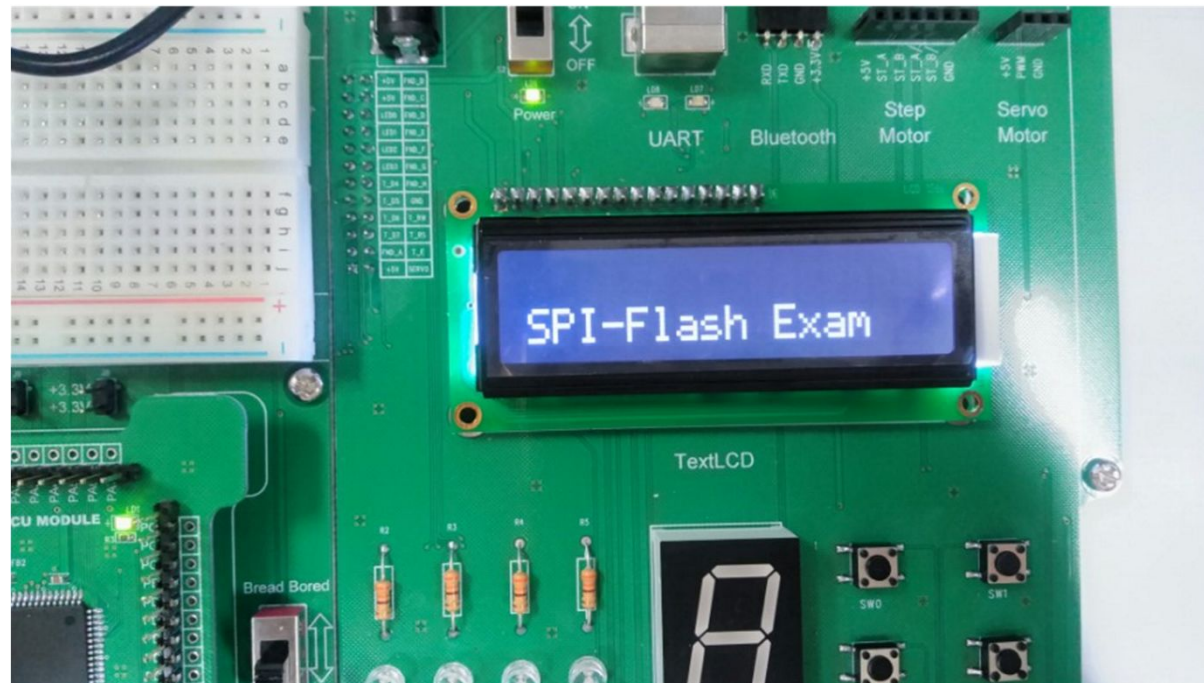
실습 2 : SPI로 EEPROM 붙이기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {  
    //...  
    for(i=0, lcdGotoXY(0, 1); i<ARRAY_SIZE(msg2)-1; i++){  
        lcdDataWrite(buf2[i]);  
        _delay_ms(100);  
    }  
    for(i=0; i<ARRAY_SIZE(msg3)-1; i++){  
        lcdDataWrite(buf3[i]);  
        _delay_ms(100);  
    }  
    for(i=0; i<16; i++){          // LCD 화면을 왼쪽으로 시프트  
        lcdControlWrite(1<<LCD_MOVE | 1<<LCD_MOVE_DISP );  
        _delay_ms(100);  
    }  
    while(1) { }  
}
```


실습 2 : SPI로 EEPROM 붙이기

- 실행 결과
 - "Welcome! edgeiLAB-World!! SPI-FLASH Exam" 문자열을 LCD에 출력

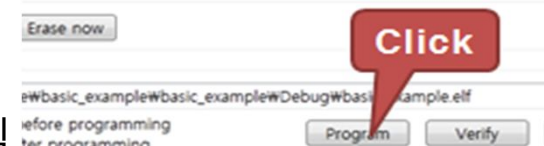
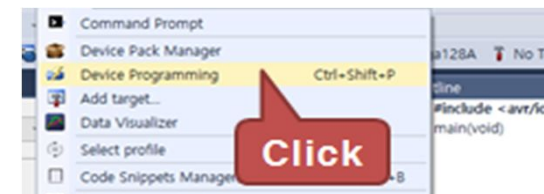
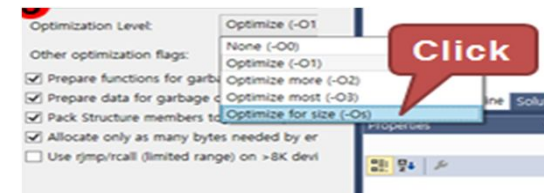
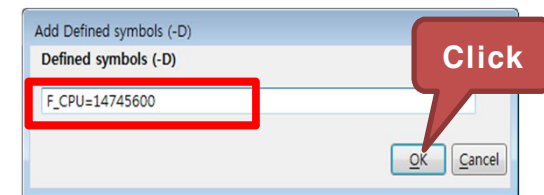


응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 실습 개요
 - I2C를 이용한 SHT20의 온습도 정보를 SPI 인터페이스를 이용하여 Serial EEPROM Memory에 저장했다가, 이를 다시 꺼내어 UART로 전송
 - SPI로 제어되는 EEPROM Memory인 AT25160B칩을 메모리로 사용
 - AT25160B 칩의 SPI 인터페이스 관련 라이브러리 함수는 at25160.c에 포함되어 있음
- 실습 목표
 - SPI 인터페이스 동작 원리 이해 (레지스터 설정)
 - TWI(I2C) 의 동작 원리 이해 (레지스터 설정)
 - ATmega128A의 SPI 포트에 대한 프로그램 방법 습득
 - Serial EEPROM Memory(AT25160B) 동작 원리 이해

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 10_SPI_Application, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭







응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - at25160.c, at25160.h 파일은 "AVR_Example\library\wat25" 폴더에 존재
 - SHT2x.c, SHT2x.h, TWI_driver.c, TWI_driver.h 파일은 "AVR_Example\library\wsht20" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



sht20 Library 폴더

이름





 SHT2x
 SHT2x
 TWI_driver
 TWI_driver

at25 Library 폴더

이름

 at25160
 at25160

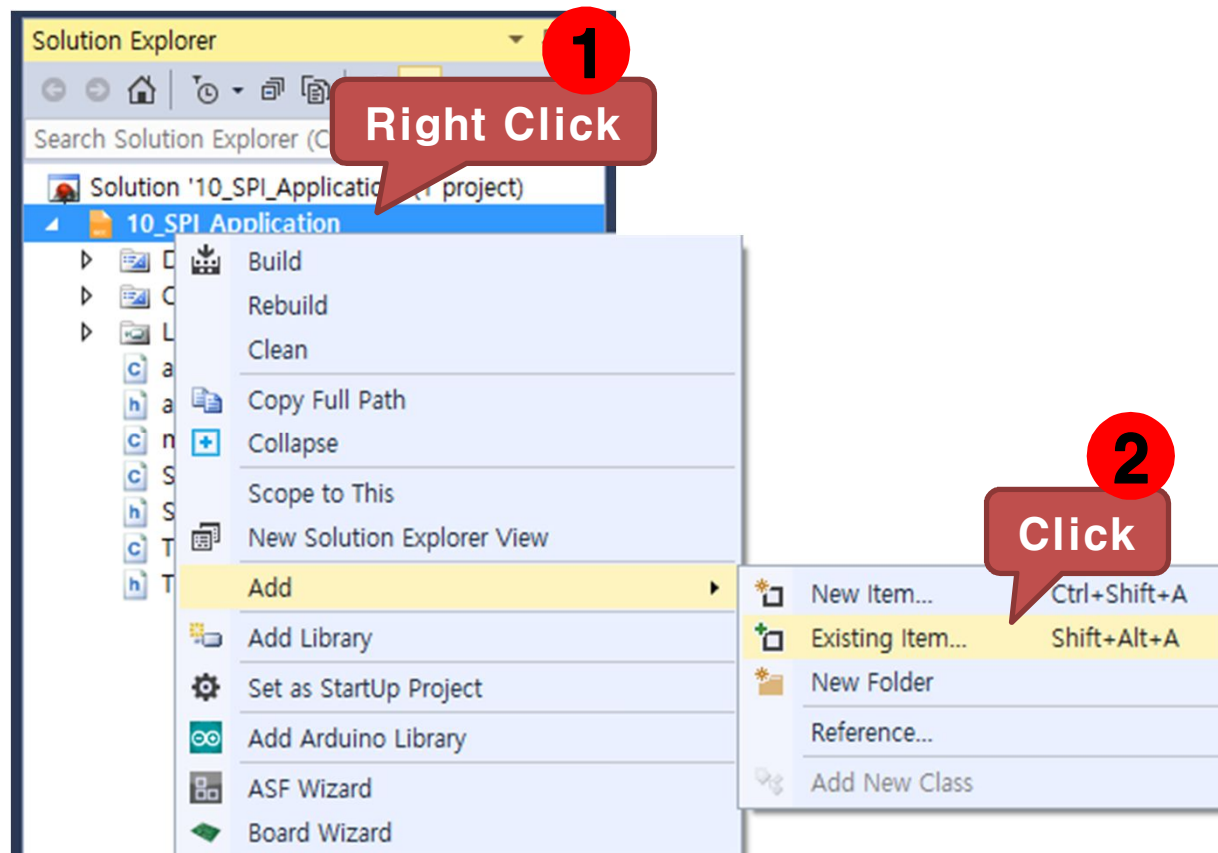
10_SPI_Application 프로젝트 폴더

이름	수정한 날짜	유형
 Debug	2016-09-06 오전...	파일
 main	2016-09-06 오전...	C S
 spieeprom.componentinfo	2016-09-06 오후...	XMI
 spieeprom	2016-09-06 오전...	ATM

파일 복사

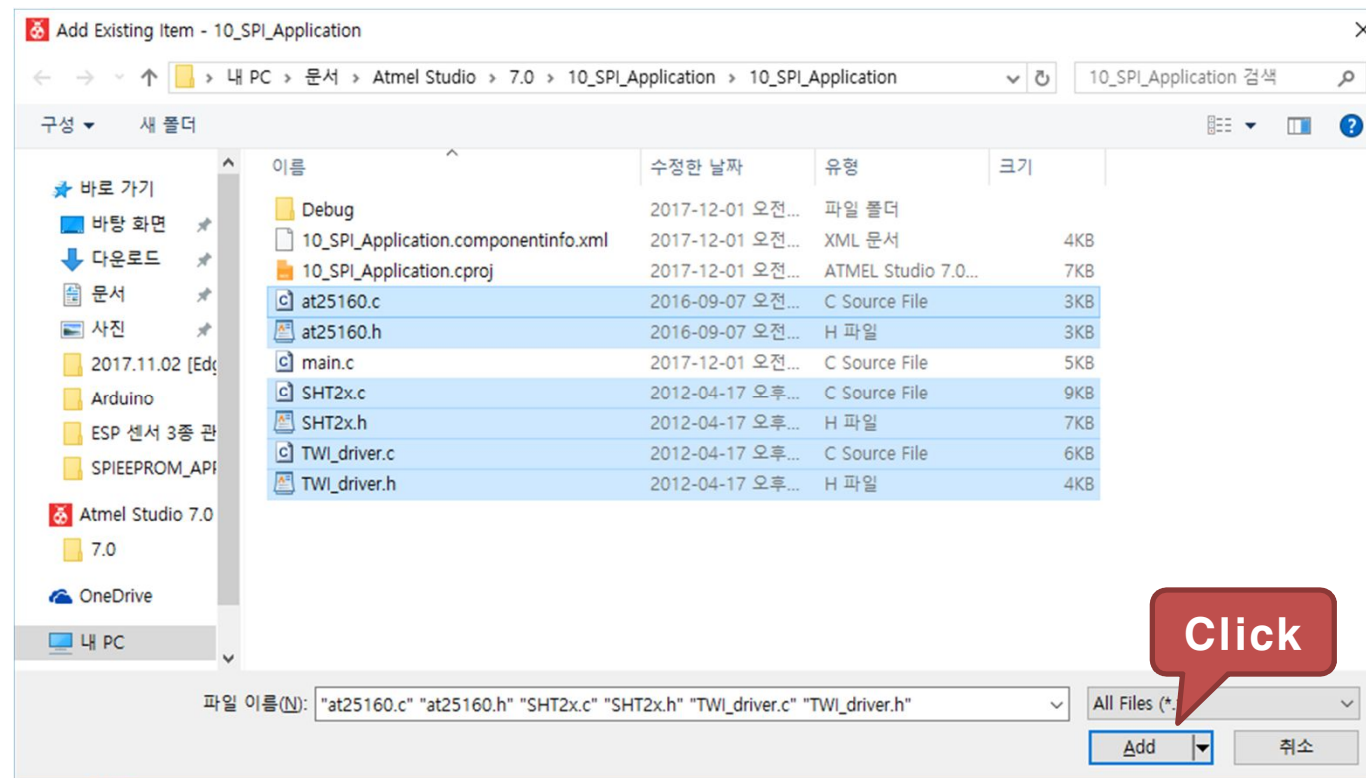
응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 여섯개의 파일을 추가
 - Ctrl 키를 누르고 파일 여섯개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
#include <avr/io.h>           //AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>        //delay 함수사용을 위한 헤더파일
#include "at25160.h"          //AT25160B 라이브러리 사용을 위한 헤더 파일
#include "TWI_driver.h"       //TWI 사용을 위한 헤더 파일
#include "SHT2x.h"            //SHT2x 라이브러리 사용을 위한 헤더 파일

volatile unsigned char Read_flag = 1, TX_flag = 0;
unsigned int timer0Cnt=0;
unsigned int temperatureC, humidityRH; //온도, 습도 값 측정에 사용
되는 변수

// EEPROM에 읽고 쓰기 위한 데이터
typedef union {
    unsigned int data16;
    unsigned char data8[2];
}DataDB;
```


응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
DataDB Read_Data;          // EEPROM 읽기 버퍼
DataDB Write_Data;         // EEPROM 쓰기 버퍼

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0); // 전송준비가 될때까지 대기
    UDR0 = data;                  // 데이터를 UDR0에 쓰면 전송된다
    UCSR0A |= 0x20;
}

// 문자열을 출력하는 함수
void putch_Str(char *str)
{
    unsigned char i=0;
    while(str[i]!='\0')
        putch(str[i++]);        //문자열을 출력
}
```


응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void)
{
    unsigned char error = 0;    //에러를 저장하는 변수

    int16 sRH;    //습도의 raw 데이터를 저장하는 변수
    int16 sT;    //온도의 raw 데이터를 저장하는 변수

    SPI_Init();

    Init_TWI();    //TWI를 초기화 한다
    SHT2x_Init(); //SHT 센서를 초기화 한다

    DDRE = 0x02;    // Rx(입력 0), Tx(출력, 1), SW0입력

    UCSRA = 0x00;
    UCSRB = 0x18; // Rx, Tx enable
    UCSR0C = 0x06; // 비동기 방식, No Parity bit, 1 Stop bit
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {  
    //...  
    UBRR0H = 0x00;  
    UBRR0L = 0x07; // 115200 bps  
  
    EICRB = 0x03; // 인터럽트 4을 상승에지에서 동작하도록 설정한다.  
    EIMSK = 0x10; // 인터럽트 4을 허용  
    EIFR = 0x10; // 인터럽트 4 플래그를 클리어  
  
    TCCR0 = 0x07;  
    TCNT0 = 112; // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생  
    TIMSK = 0x01;  
    TIFR |= 1 << TOV0;  
  
    sei(); // 전체 인터럽트를 허용
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {
    //...
    while(1)
    {
        if(Read_flag)
        {
            error |= SHT2x_MeasureHM(HUMIDITY, &sRH);
            //습도를 측정한다.
            error |= SHT2x_MeasureHM(TEMP, &sT);
            //온도를 측정한다.
            //온도 습도를 계산, 소숫점 첫째자리까지 출력하기 위해 10을
            곱한다.

            temperatureC = SHT2x_CalcTemperatureC(sT.u16)*10;
            //온도를 계산한다.
            humidityRH    = SHT2x_CalcRH(sRH.u16)*10;
            //습도를 계산한다.
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {  
    //...  
  
    if(error == SUCCESS) //에러없이 정상 측정 되었으면  
    {  
        Write_Data.data16 = temperatureC;  
        at25160_Write_Arry(0x0100, Write_Data.data8,2);  
        // EEPROM에 온도 저장  
        Write_Data.data16 = humidityRH;  
        at25160_Write_Arry(0x0200, Write_Data.data8,2);  
        // EEPROM에 습도 저장  
    }  
    Read_flag = 0;  
}
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {  
    //...  
  
    if(TX_flag)  
    {  
        at25160_Read_Arry(0x0100, Read_Data.data8,2);  
        // EEPROM에서 온도 읽음  
        temperatureC = Read_Data.data16;  
        at25160_Read_Arry(0x0200, Read_Data.data8,2);  
        // EEPROM에서 습도 읽음  
        humidityRH = Read_Data.data16;  
        putch_Str("\n\r Temp: ");  
        // 온도 출력
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {
    //...

    // 100의 자리 값 확인
    if(temperatureC/100) // 100의 자리가 0이 아닐 경우
    {
        // 100의 자리 출력
        putchar(temperatureC/100 + '0');
    }
    putchar((temperatureC/10)%10 + '0');
    putchar('.');
    putchar(temperatureC%10 + '0');
    putchar_Str("[C], Humi : ");
    // 습도 출력
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {
    //...

    // 100의 자리 값 확인
    if(humidityRH/100) // 100의 자리가 0이 아닐 경우
    {
        // 100의 자리 출력
        putchar(humidityRH/100 + '0');
    }
    putchar((humidityRH/10)%10 + '0');
    putchar('.');
    putchar(humidityRH%10 + '0');
    putchar_Str("[%]");
    TX_flag = 0;
}
}
```

응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

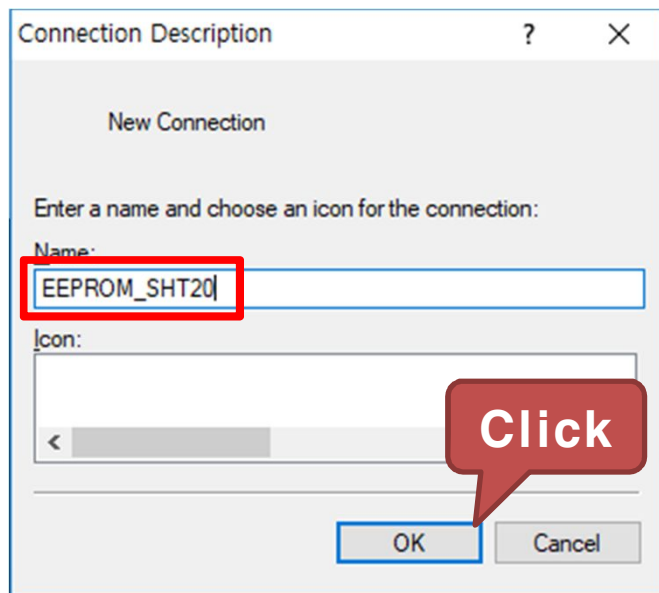
- 구동 프로그램
 - main.c 코드 작성

```
SIGNAL(INT4_vect) { // 인터럽트 서비스 루틴
    cli();          // 전체 인터럽트를 금지
    TX_flag = 1;    // 온도 및 습도를 PC로 전송하도록 설정
    sei();          // 전체 인터럽트를 허용
}

SIGNAL(TIMER0_OVF_vect){
    cli();
    TCNT0 = 112;    // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
    timer0Cnt++;    // timer0Cnt 변수를 1 증가 시킨다.
    if(timer0Cnt == 100){
        // 0.01s * 100 = 1s //1초를 얻기 위한 카운트 횟수
        Read_flag = 1; // 온도와 습도 측정 및 PC로 전송하도록 1로 설정
        timer0Cnt = 0;
    }
    sei();
}
```


응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 하이퍼터미널 실행
 - 연결 설명 창의 이름 란에 "EEPROM_SHT20" 라고 확인을 클릭
 - 연결 대상 창의 연결에 사용할 모뎀 번호를 설정하고 확인을 클릭



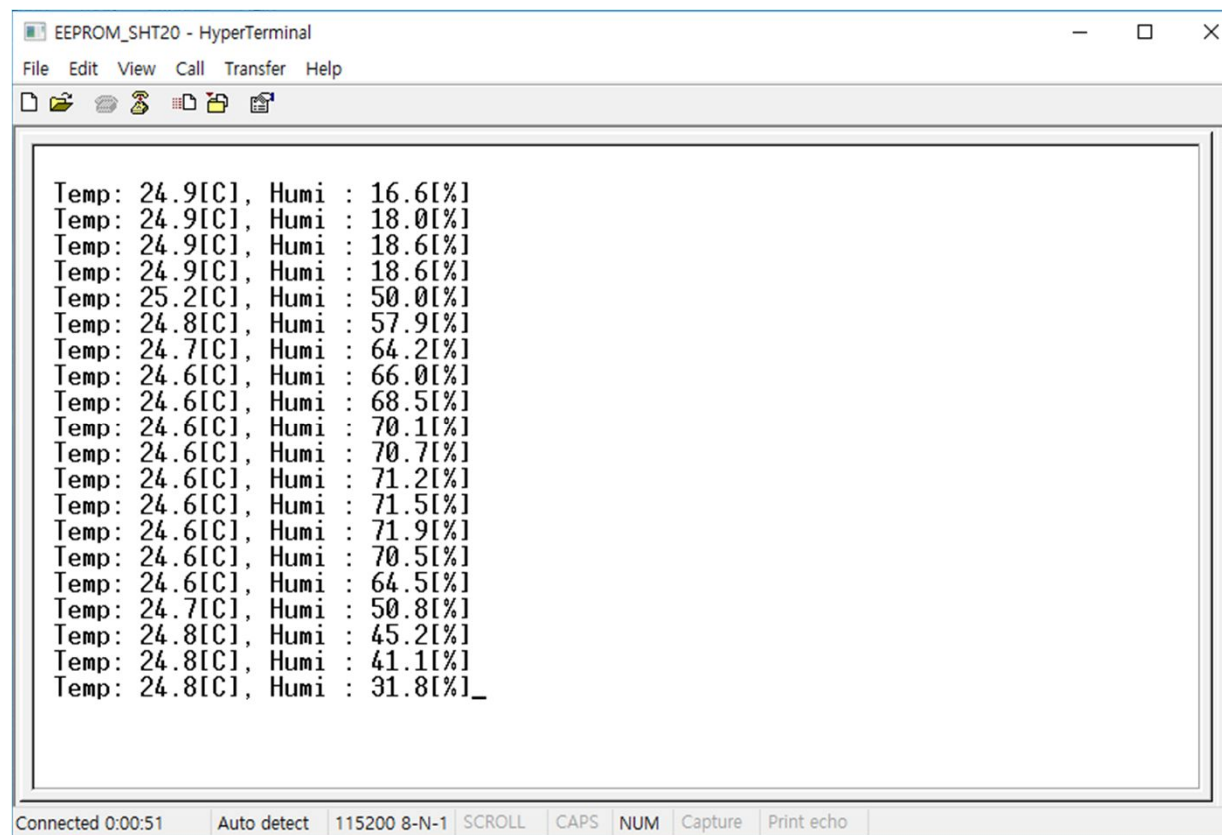
응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 하이퍼터미널 실행
 - 통신 설정을 그림과 같이하고, 확인을 클릭



응용 2 : EEPROM에 저장된 온습도 정보를 PC에 출력하기

- 실행 결과
 - 온습도 결과를 EEPROM에 저장하고, 스위치를 누르면 PC로 전송됨을 확인



```
EEPROM_SHT20 - HyperTerminal
File Edit View Call Transfer Help

Temp: 24.9[C], Humi : 16.6[%]
Temp: 24.9[C], Humi : 18.0[%]
Temp: 24.9[C], Humi : 18.6[%]
Temp: 24.9[C], Humi : 18.6[%]
Temp: 25.2[C], Humi : 50.0[%]
Temp: 24.8[C], Humi : 57.9[%]
Temp: 24.7[C], Humi : 64.2[%]
Temp: 24.6[C], Humi : 66.0[%]
Temp: 24.6[C], Humi : 68.5[%]
Temp: 24.6[C], Humi : 70.1[%]
Temp: 24.6[C], Humi : 70.7[%]
Temp: 24.6[C], Humi : 71.2[%]
Temp: 24.6[C], Humi : 71.5[%]
Temp: 24.6[C], Humi : 71.9[%]
Temp: 24.6[C], Humi : 70.5[%]
Temp: 24.6[C], Humi : 64.5[%]
Temp: 24.7[C], Humi : 50.8[%]
Temp: 24.8[C], Humi : 45.2[%]
Temp: 24.8[C], Humi : 41.1[%]
Temp: 24.8[C], Humi : 31.8[%]_

Connected 0:00:51  Auto detect  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```