

14 장 : 온도센서를 이용하여 디지털 온도계 만들기

ATmega128 마이크로컨트롤러를 이용한 임베디드시스템 구현



JCnet
제이씨넷

신상석

목차

1. 온도 센서
2. TWI(I2C) 통신
3. JKIT-128-1에서의 온도 센서 연결 설계
4. ATmega128의 TWI 인터페이스
5. ATS75 온도 센서의 TWI 인터페이스
6. 실습 TEMP-1 : TWI 인터페이스로 통신 하기
7. 실습 TEMP-2 : 온도센서로 디지털 온도계 만들기

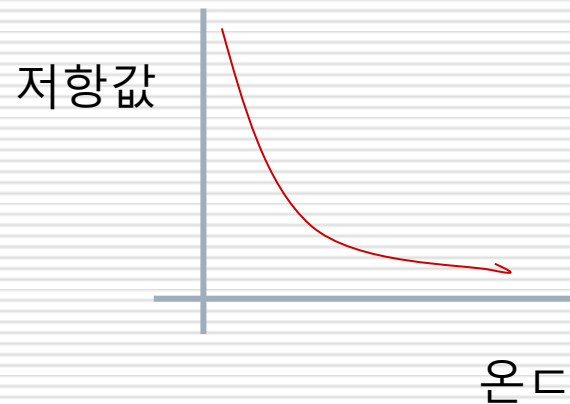
온도 센서

□ 온도 센서 (아날로그)

- 보통 Thermistor를 이용하여 온도 측정

- Thermistor

- 니켈, 코발트, 구리, 철 등의 화합물로 이루어진 물질
- 온도가 높아지면 저항값이 내려가고, 온도가 내려오면 저항값이 올라가는 특성을 가짐



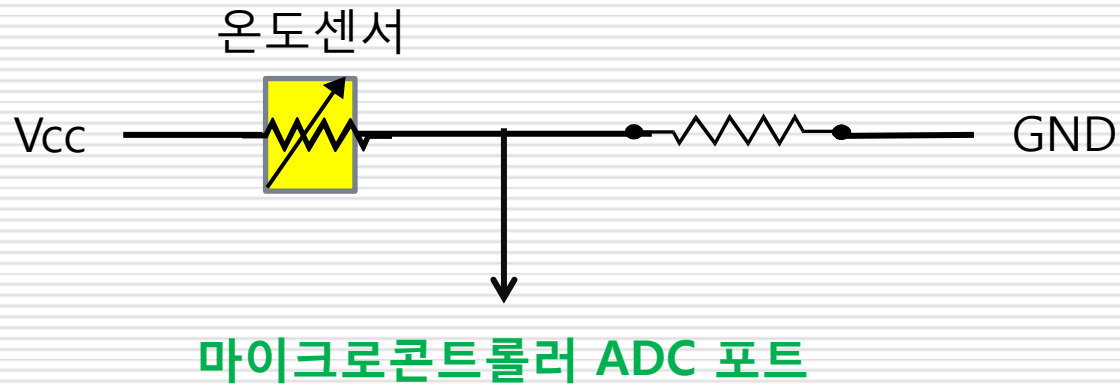
온도 센서

□ 온도 센서 (아날로그)

■ 온도센서 저항값의 예

- 100도 : 1 K Ω 이하
- 0 도 : 15 K Ω 정도
- -10도 : 25 K Ω 이상

■ 마이크로컨트롤러와의 연결



온도 센서

□ 온도 센서 (디지털)

- 아날로그 온도 센서와 이와 결합한 로직을 이용하여 디지털 인터페이스를 가지는 온도 센서
- 보통 모듈이나 IC 형태를 가짐
- 마이크로컨트롤러와의 연결



I2C(TWI) 통신

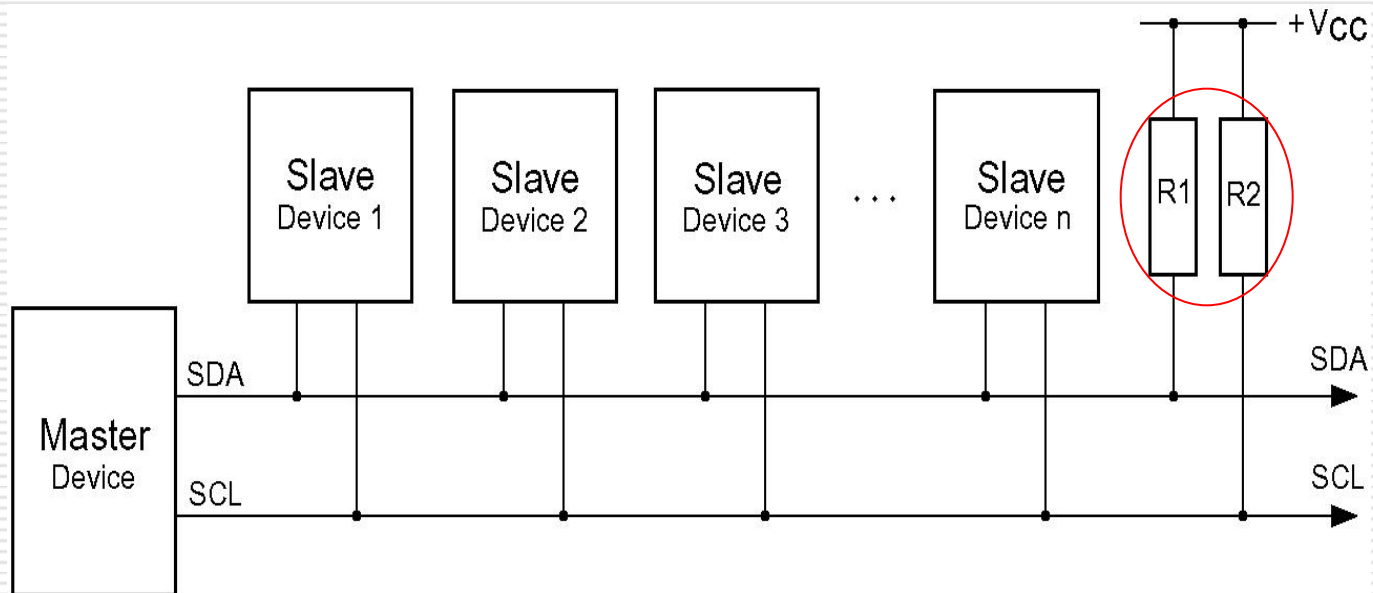
□ 기본 규격

- I2C : Inter Integrated Circuit (I square C)
- TWI : Two Wire Interface
- Philips 사가 제안한 근거리용 직렬 동기식 양방향 통신 규격
- 마이크로컨트롤러에 여러가지 메모리, I/O 소자를 연결할 때 사용
- SDA(Serial Data), SCK(Serial Clock)의 2개 신호만 사용
 - SDA(Serial Data)는 양방향으로 Master와 Slave가 함께 사용
 - SCK(Serial Clock)는 항상 Master가 제공
 - 100kbps, 400kbps, 3.4Mbps의 3가지 전송 속도 지원
 - Slave 지정은 7비트 어드레스를 이용하며, 전체 Slave를 호출하는 기능(broadcasting)도 지원
 - Multi Master 조정(arbitration) 기능 지원

I2C(TWI) 통신

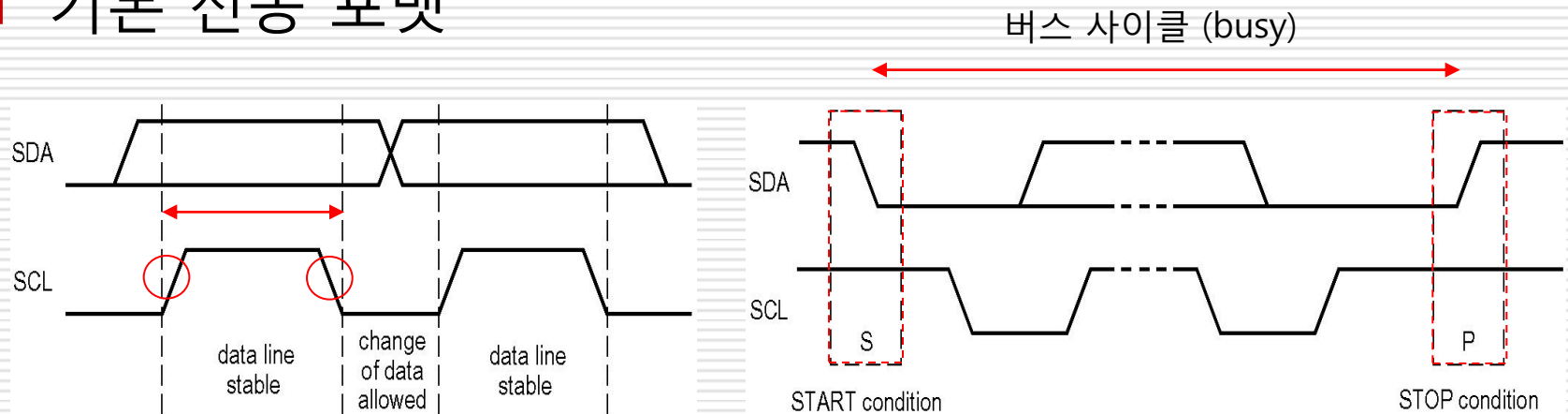
□ 시스템의 구성

* Open Drain 또는 Open Collector 방식



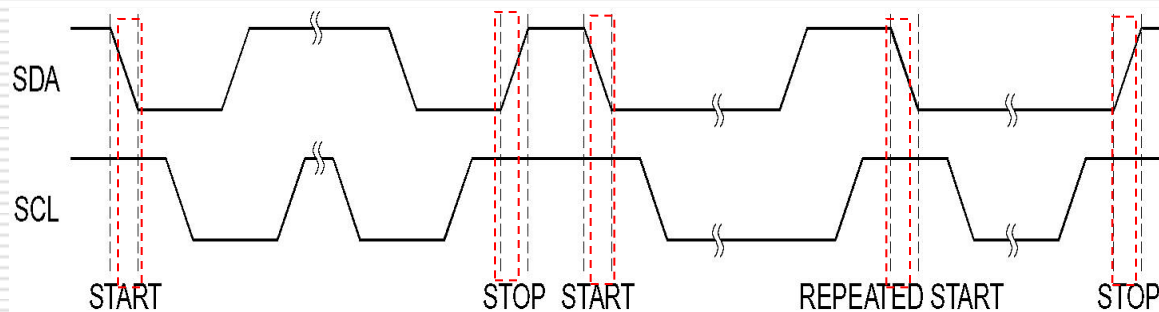
I2C(TWI) 통신

□ 기본 전송 포맷



(a) 1비트 데이터 전송

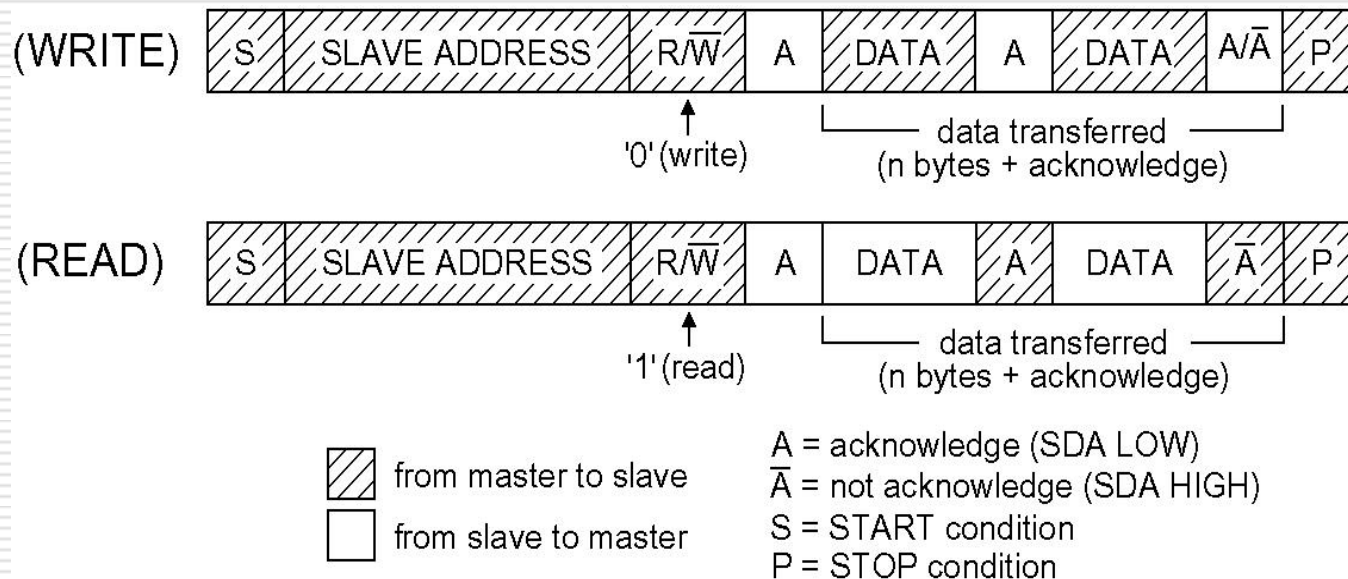
(b) START, STOP



(c) START, REPEATED START, STOP

I2C(TWI) 통신

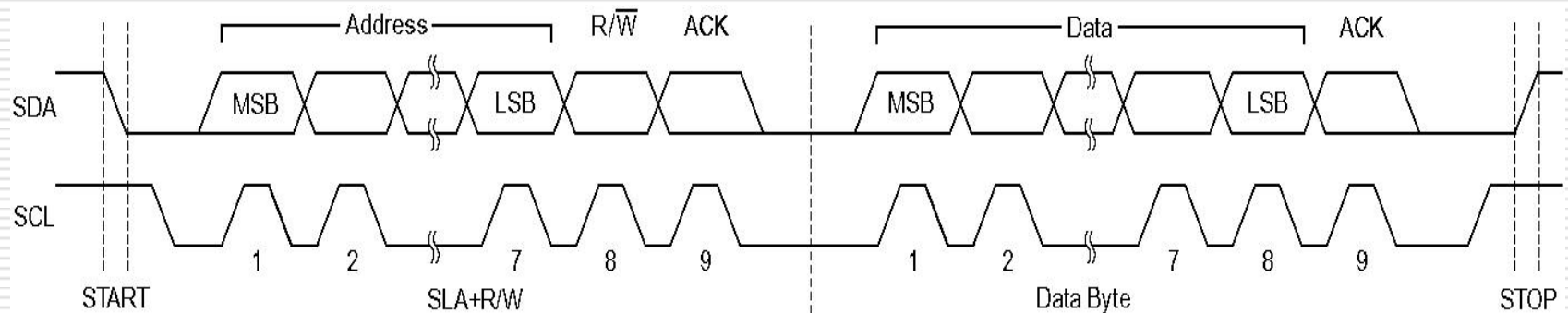
□ 데이터 송수신 포맷



I2C(TWI) 통신

□ 데이터 송수신 동작

- Address Cycle + Data Cycle
- Stop이 나타나기 전까지는 하나의 Master가 버스를 관장함



* Data Byte의 개수는 가변

JKIT-128에서의 온도센서 연결 설계

□ JKIT-128-1에서의 온도센서 연결 설계 개념

- 온도센서는 다양성을 위하여 디지털 온도센서 사용
- 가능한 I2C나 TWI 통신 인터페이스가 가능한 센서 채택
- 가격이 저렴하고 쉽게 구할 수 있는 센서 선택

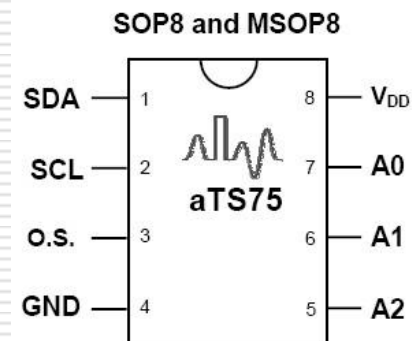
➔ <aTS75> 디지털 온도 센서 (I2C 인터페이스) 채택

- Slave Address 값 중 Low 3비트의 값은 "000"과 "100"의 2개의 값 중 선택이 가능하도록 처리
- SCL, SDA에 pullup 저항 연결 필요
 - (JKIT-128-1) MISSING !!! ➔ 설계 오류
 - 해결 방법 : SCL, SDA가 연결되어 있는 포트인 PD0, PD1에 연결된 pullup 저항을 활성화시켜서 해결(SW 해결법 사용)

JKIT-128에서의 온도센서 연결 설계

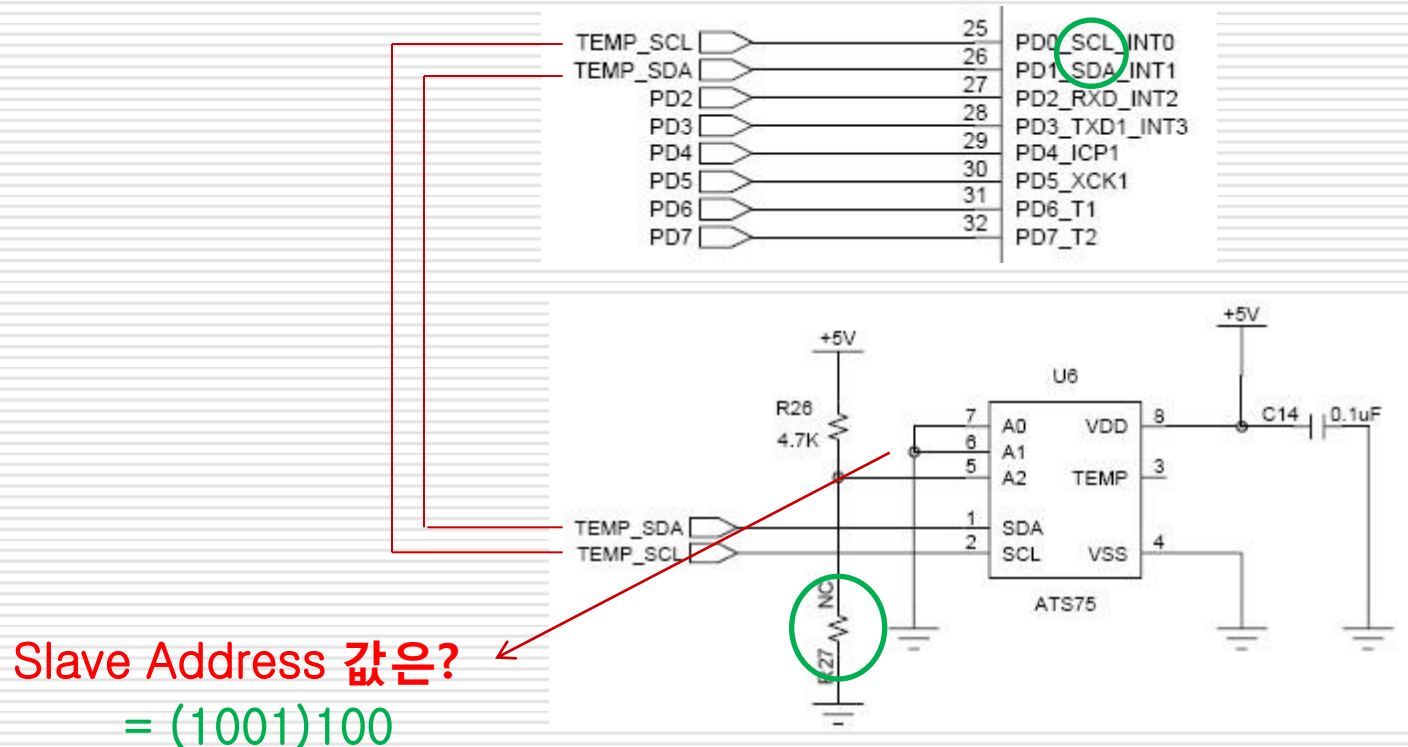
□ aTS75 온도 센서

- 디지털 온도 센서
- 온도 측정 범위 : 섭씨 -30도 ~ 125도
- 측정 오차 : 최대 +/- 2도(25도에서), +/-3도 (전 온도 구간)
- 분해능(resolution) : 9~12비트 (0.5도 ~ 0.0625도)
- 인터페이스 : I2C (TWI)
- 동작 전압 : 2.7V ~ 5.5V
- 패키지 타입 : SOIC-8, MSOP-8



JKIT-128에서의 온도센서 연결 설계

□ JKIT-128-1에서의 온도센서 연결 설계



JKIT-128에서의 온도센서 연결 설계

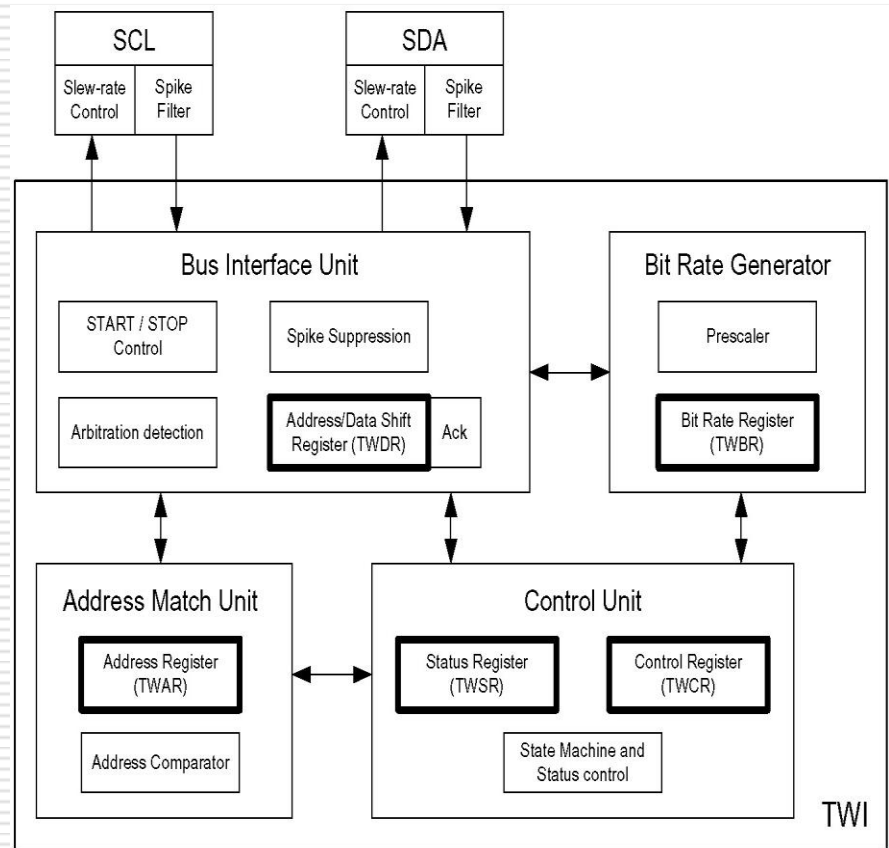
마이크로컨트롤러와 외부 IC(또는 모듈) 연결시 주의사항 !!!

1. 매뉴얼(datasheet, manual, 설명서 등)이 가장 중요 !!!
2. 하드웨어 연결
 - 핀 정의, 기능 및 연결 방법
 - 예제 연결도(Recommended Circuits)
 - 전기적 규격(Electrical Ratings : 정격 전압/전류 등)
 - 주변 RLC(Resistor, Inductor, Capacitor) 규격, 연결방법 등
3. 소프트웨어 인터페이스
 - 오퍼레이션(인터페이스) 방법/순서
 - 예제 프로그램(Example Source Code)

ATmega128의 TWI 인터페이스

□ ATmega128의 TWI 특징

- 7비트 어드레스 방식만 지원
- I2C 전송속도 중 표준모드(100kbps)와 고속모드(400kbps)만 지원
- 구성 블록도



ATmega128의 TWI 인터페이스

□ TWBR : TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
\$70	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
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	0	0	0	
	0	1	0	0	0	0	0	0	= 0x40

■ SCL 클록의 주파수를 맞추기 위한 분주비 설정

$$F_{scl} \text{ (SCL 클록 주파수)} = \frac{F_{cpu} \text{ (CPU 클록 주파수)}}{16 + 2 * TWBR * 4^{TWPS}} = \frac{16\text{Mhz}}{16 + 2 * 64 * 4^0}$$

(단, TWBR은 10 이상, TWPS는 TWSR 레지스터 설정값)

ATmega128의 TWI 인터페이스

□ TWDR : TWI Data Register

Bit	7	6	5	4	3	2	1	0	
\$73	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- 송신모드에서 다음에 송신(write) 할 바이트(슬레이브 어드레스 또는 데이터)를 저장
- 수신모드에서 수신(write)된 바이트(데이터)를 저장

ATmega128의 TWI 인터페이스

□ TWAR : TWI Control Register

Bit	7	6	5	4	3	2	1	0	
\$74	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- TWINT(TWI Interrupt Flag) : TWI가 현재의 동작을 완료했음을 알리는 플래그로 '1'은 동작완료를 나타냄, 자동으로 clear되지 않으며 이것이 '1'인 동안은 SCL이 '0' 상태를 유지하므로 반드시 clear시켜야 함, clear 하려면 이 비트에 '1'을 write함.
- TWEA(TWI Enable Acknowledge) : 1 바이트 데이터를 수신한 경우 ACK 신호를 '1'로 발생하도록 함
- TWSTA(TWI Start Condition Bit) : START 조건 출력, 다시 clear되어야 함

ATmega128의 TWI 인터페이스

□ TWAR : TWI Control Register

Bit	7	6	5	4	3	2	1	0	
\$74	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- TWSTO(TWI STOP Condition Bit) : STOP 조건 출력, 자동으로 clear됨
- TWEN(TWI Enable) : TWI 버스를 enable시킴
- TWIE(TWI Interrupt Enable) : TWI 인터럽트가 발생하도록 허용, '1'로 세트되어 있으면 TWINT가 생성될 때 인터럽트 발생

ATmega128의 TWI 인터페이스

□ TWSR : TWI Status Register

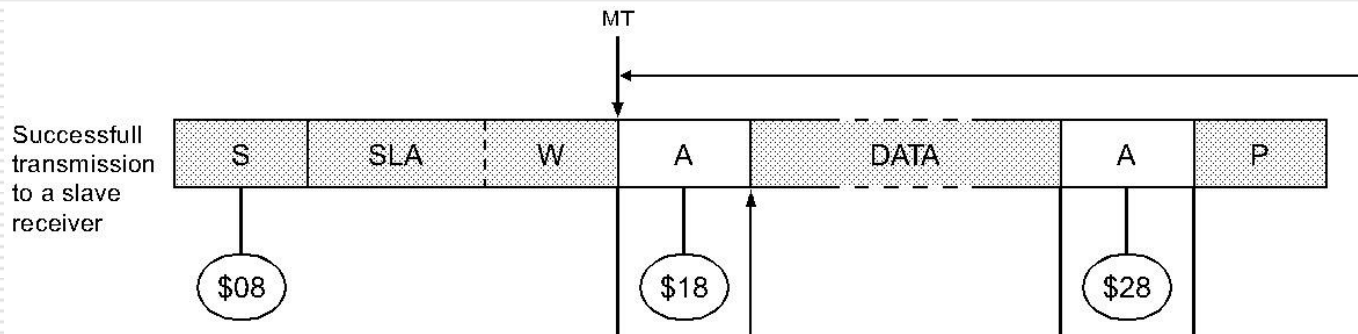
Bit	7	6	5	4	3	2	1	0	
\$71	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- TWS[7:3](TWI Status) : TWI 진행 상태 표시값 (단계별 동작 참조)
- TWPSI[1:0](TWI Prescaler Bits) : TWI 클록 계산에 사용되는 프리스케일러값 세팅
00 = 1분주, 01=4분주, 10=16분주, 11=64분주

ATmega128의 TWI 인터페이스

□ TWI 동작

■ Master 송신(Write)

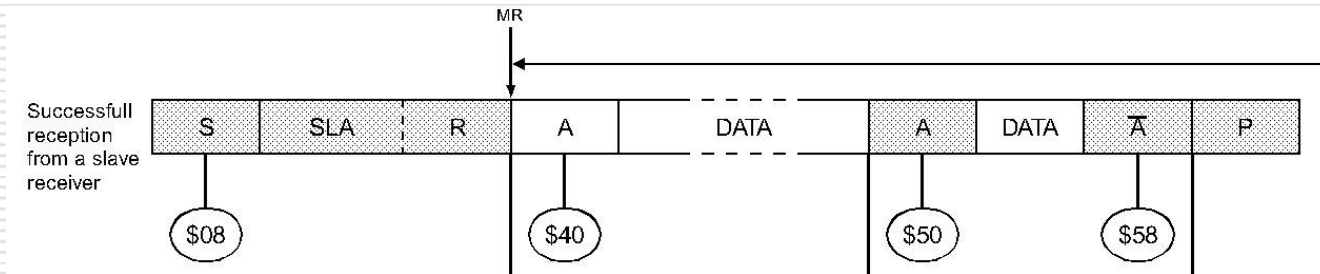


1. START 전송 : $TWCR \rightarrow TWINT=1, TWSTA=1, TWSTO=0, TWEN=1$
2. 상태 체크 : $TWSR \rightarrow TWSR \& 0xFC = 0x08$
3. TWDR에 SLA+W 세트 : $TWDR \rightarrow TWDR = SLA + W$
4. SLA+W 전송 : $TWCR \rightarrow TWINT=1, TWSTA=0, TWSTO=0, TWEN=1$
5. Ack 체크 : $TWSR \rightarrow TWSR \& 0xFC = 0x18$
6. TWDR에 DATA값 세트 : $TWDR \rightarrow TWDR = DATA$
7. Ack 체크 : $TWSR \rightarrow TWSR \& 0xFC = 0x28$
8. STOP 전송 : $TWCR \rightarrow TWINT=1, TWSTA=0, TWSTO=1, TWEN=1$

ATmega128의 TWI 인터페이스

□ TWI 동작

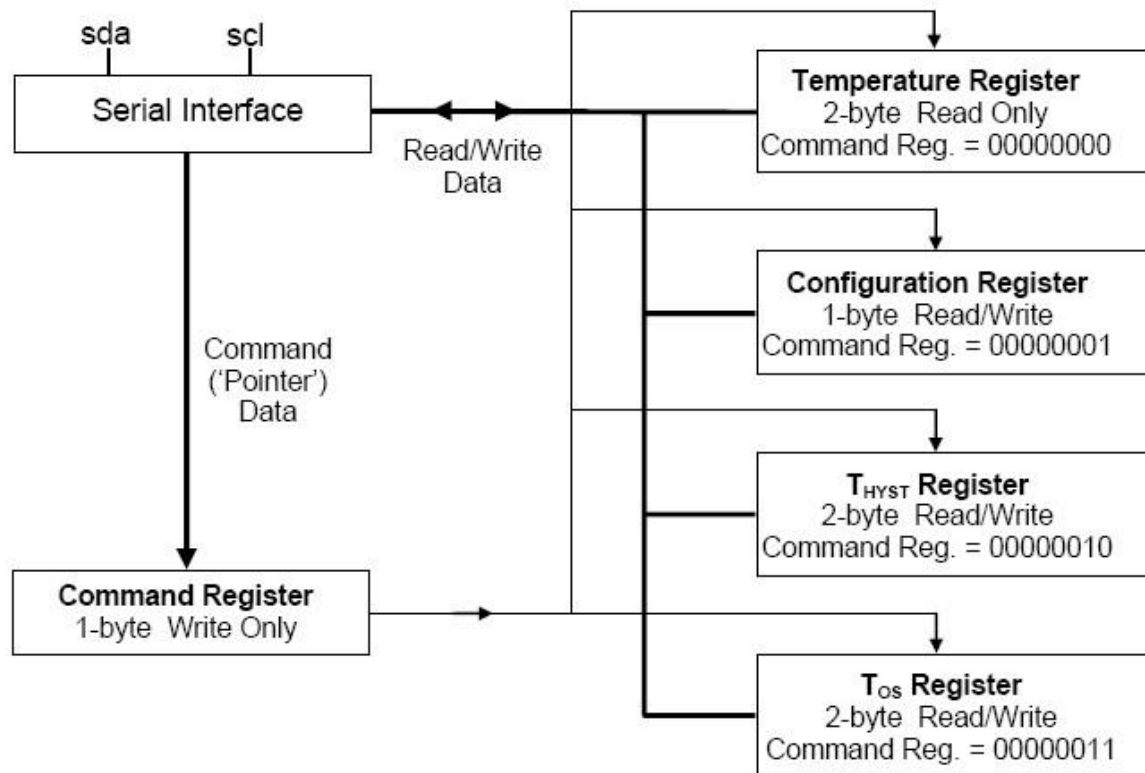
■ Master 수신(Read)



1. START 전송 : TWCR → TWINT=1, TWSTA=1, TWSTO=0, TWEN=1
2. 상태 체크 : TWSR → TWSR & 0xFC = 0x08
3. TWDR에 SLA+R 세트 : TWDR → TWDR = SLA + R
4. SLA+R 전송 : TWCR → TWINT=1, TWSTA=0, TWSTO=0, TWEN=1
5. Ack 체크 : TWSR → TWSR & 0xFC = 0x40
6. 데이터 수신 : TWCR → TWINT=1, TWEA=1, TWSTA=0, TWSTO=0, TWEN=1
7. 상태 체크 : TWSR → TWSR & 0xFC = 0x50 (다음 데이터의 경우 6, 7 반복)
8. STOP 전송 : TWCR → TWINT=1, TWSTA=0, TWSTO=1, TWEN=1

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : 레지스터



aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : Command Register

- 내부의 다른 레지스터를 지정하기 위한 레지스터
- 일종의 Indirect Address Register 역할
- 다른 레지스터에 접근하기 위하여는 먼저 Command Register에 접근하고자 하는 레지스터에 해당되는 값을 Write한 후에 원하는 레지스터를 Read/Write하여야 함

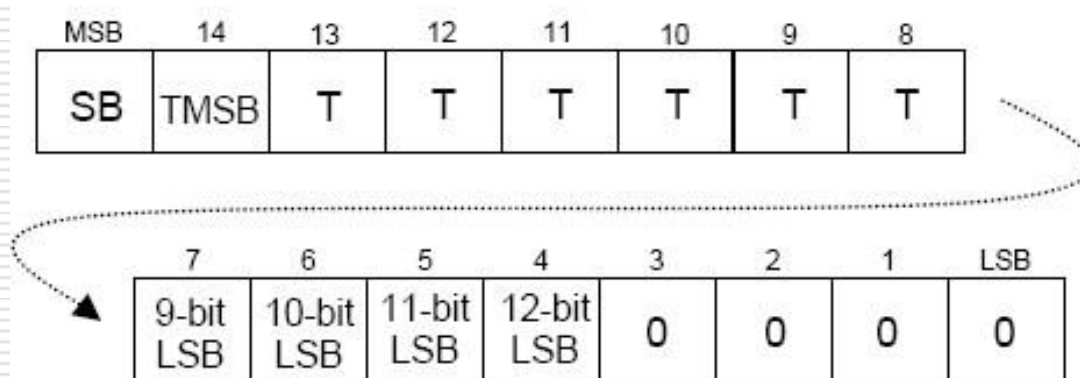
MSB						LSB	
0	0	0	0	0	0	P1	P0

Register	P1	P0
Temperature Register	0	0
Configuration Register	0	1
T _{HYST} Register	1	0
T _{OS} Register	1	1

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : Temperature Register

- 온도값 표시 : 표시 형식은 다음 테이블 참조
- 2 바이트 (16비트)



SB = Two's complement sign bit

TMSB = Temperature MSB

T = Temperature data

9-bit LSB = Temperature LSB for 9-bit conversions

10-bit LSB = Temperature LSB for 10-bit conversions

11-bit LSB = Temperature LSB for 11-bit conversions

12-bit LSB = Temperature LSB bit for 12-bit conversions

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : 온도와
디지털값과의
관계

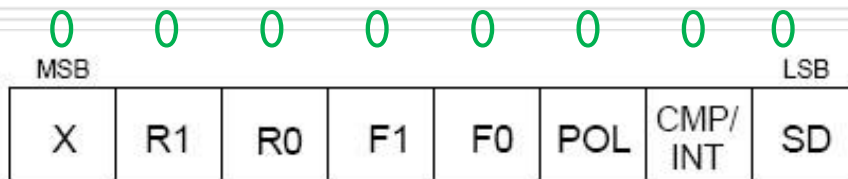
2's complement
표현 방식 사용

Temperature	Digital Output							
	Sign Bit	Number of bits used by conversion resolution		9-bit	10-bit	11-bit	12-bit	Always zero
All Temperatures	12-Bit Resolution							0000
	11-Bit Resolution					0	0000	
	10-Bit Resolution				0	0	0000	
	9-Bit Resolution			0	0	0	0000	
+125°C	0	111	1101	0	0	0	0	0000
+100.0625°C	0	110	0100	0	0	0	1	0000
+50.125°C	0	011	0010	0	0	1	0	0000
+12.25°C	0	000	1100	0	1	0	0	0000
0°C	0	000	0000	0	0	0	0	0000
-20.5°C	1	110	1011	1	0	0	0	0000
-33.25°C	1	101	1110	1	1	0	0	0000
-45.0625°C	1	101	0010	1	1	1	1	0000
-55°C	1	100	1001	0	0	0	0	0000

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : Configuration Register

- 온도값 유효 비트 설정 : 9, 10, 11, 12 비트
- 모드 설정 : **Comparator mode** vs Interrupt mode



R1 = Resolution bit 1. (See Table 3)

R0 = Resolution bit 0. (See Table 3)

F1 = Fault tolerance bit 1. (See Table 4)

F0 = Fault tolerance bit 0. (See Table 4)

POL = O.S. output polarity. 0 = active low, 1 = active high.

CMP/INT = Thermostat mode.

0 = Comparator Mode, 1 = Interrupt Mode.

SD = Shutdown. 0 = normal operation. 1 = Shutdown Mode

A-to-D Conversion Resolution	R1	R0
9 Bits	0	0
10 Bits	0	1
11 Bits	1	0
12 Bits	1	1

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : Slave Address 설정

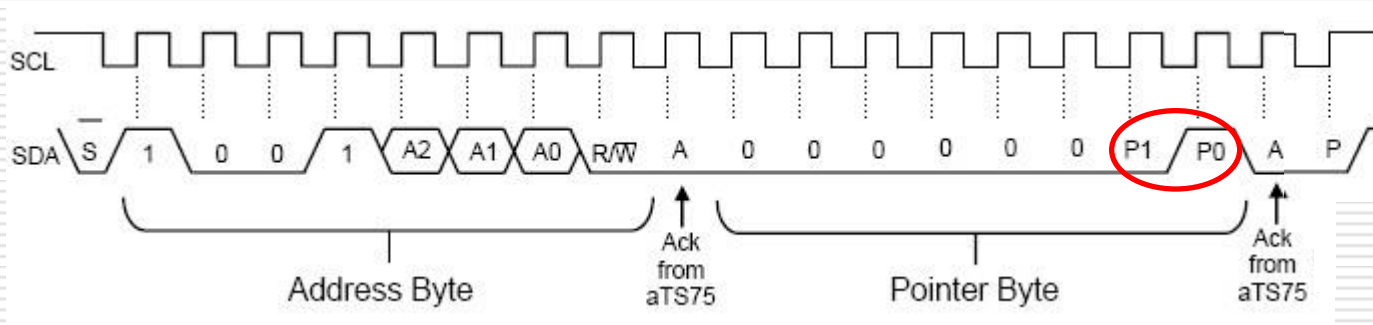
■ Slave Address 값 = 1001(A2)(A1)(A0) (7비트)

- 1001은 내부적으로 고정
- (A2)(A1)(A0)은 외부 하드웨어 신호 연결의 레벨에 따라 결정
 - ✓ Vcc 연결이면 '1'
 - ✓ GND 연결이면 '0'
 - ✓ 예 : JKIT-128-1에서는 '100' ➔ 1001100

aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : TWI Read/Write Operation

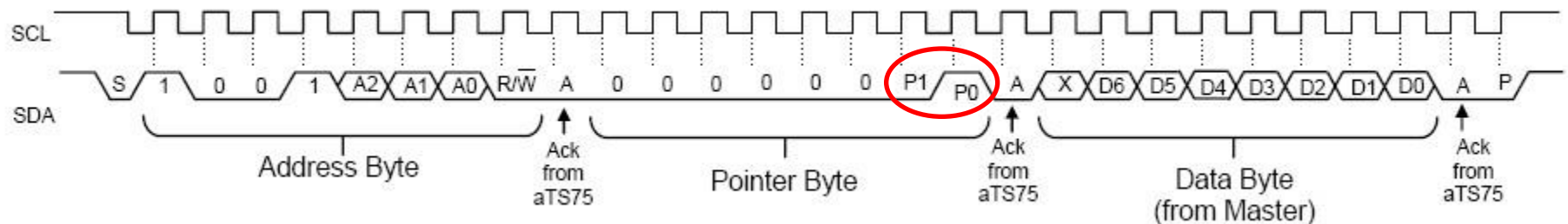
■ Pointer Set(Command Register)



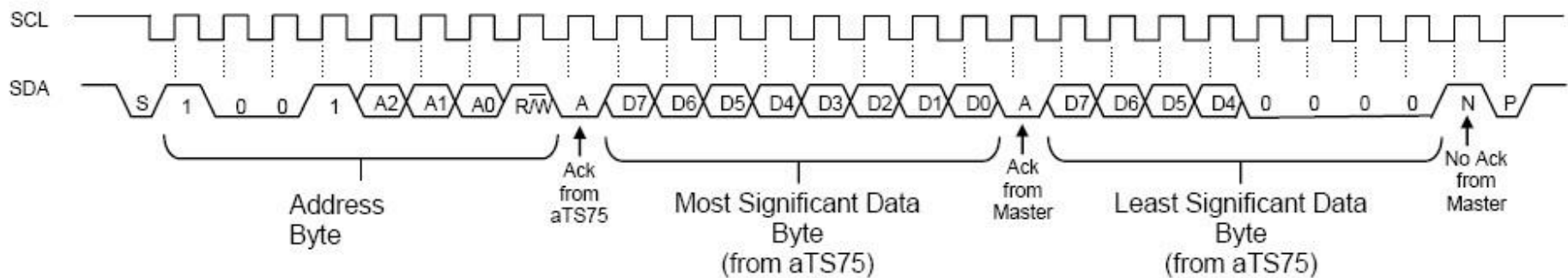
aTS75 온도 센서의 TWI 인터페이스

□ aTS75 : TWI Read/Write Operation

■ One Byte Write with Pointer Set(Command/Configuration R)



■ Two Bytes Read with Preset



실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 실습 내용

1. TWI 인터페이스로 통신하여 aTS75에 2 바이트 Write 하기
(Commnad Register / Configuration Register)
: aTS75를 9비트 Normal 모드로 동작하도록 세팅
 - 첫번째 데이터는 Command Register가 Configuration Register를 가르키도록 하는 내용
 - 두번째 데이터는 Configuration Register의 세팅 내용
2. TWI 인터페이스로 통신하여 aTS75에 1 바이트 Write 한 후,
(Commnad Register) 2 바이트 Read 하기 (Temperature Register) : aTS75의 온도값 읽기
 - 첫번째 Write의 데이터는 Command Register가 Temperature Register를 가르키도록 하는 내용
 - 두번째 Read의 2바이트 데이터는 Temperature Register의 값

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 설계 : TWI 2바이트 Write (temp_1_1.c)

■ TWI 초기화

- TWI 클럭 설정 : TWBR, TWSR = 40Kbps
- SCL, SDA pullup 처리 : SFIOR의 PUD 비트 세트

■ TWI (Start) 전송 및 Ack 검사

■ TWI (Slave Address + Write) 전송 및 Ack 검사

■ TWI (1st Data) 전송 및 Ack 검사

→ 1st Data는 Configuration Register를 가르키는 값

■ TWI (2nd Data) 전송 및 Ack 검사

→ 2nd Data는 Configuration Register에 Write할 값

■ TWI (Stop) 전송

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 설계 : TWI 2바이트 Write (temp_1_1.c)



실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 코딩 : TWI 2바이트 Write (temp_1_1.c)

```
#define F_CPU    16000000UL    // CPU 클럭 값 = 16 Mhz
#define F_SCK    40000UL      // SCK 클럭 값 = 40 Khz
#include <avr/io.h>
#include <util/delay.h>
#define UCHAR    unsigned char    // UCHAR 정의
#define AT575_ADDR    0x98    // 0b10011000, 7비트를 1비트 left shift
#define AT575_CONFIG_REG    1
#define AT575_TEMP_REG    0
void init_twi();
void write_twi_1byte_nopreset(UCHAR reg, UCHAR data);
void display_LED(UCHAR value);
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 코딩 : TWI 2바이트 Write (temp_1_1.c)

```
int main()
{
    init_twi();                // TWI 초기화
    write_twi_1byte_nopreset(ATS75_CONFIG_REG, 0x00); // 9비트, Normal
    display_LED((UCHAR)(0x0F)); // 성공시 0x0F 디스플레이
}

void init_twi()
{
    PORTD = 3;                // For Internal pull-up for SCL & SCK
    SFIOR &= ~(1<<PUD);       // PUD = Pull Up Disable
    TWBR = (F_CPU/F_SCK - 16) / 2; // 공식 참조, bit trans rate 설정
    TWSR = TWSR & 0xfc;        // Prescaler 값 = 00 (1배)
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 코딩 : TWI 2바이트 Write (temp_1_1.c)

```
void write_twi_1byte_nopreset(UCHAR reg, UCHAR data)
{
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);    // START 전송
    while((TWCR & (1 << TWINT)) == 0x00 || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));    // ACK 를 기 다 림
    TWDR = ATS75_ADDR | 0;    // SLA+W 준비, W=0
    TWCR = (1 << TWINT) | (1 << TWEN);    // SLA+W 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x18);
    TWDR = reg;    // aTS75 Reg 값 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // aTS75 Reg 값 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x28);
    TWDR = data;    // DATA 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // DATA 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x28);
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);    // STOP 전송
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 코딩 : TWI 2바이트 Write (temp_1_1.c)

```
void display_LED(UCHAR value)
{
    DDRA = 0xff;
    PORTA = value;
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)
 - TWI 초기화
 - TWI 클록 설정 : TWBR, TWSR = 40Kbps
 - SCL, SDA pullup 처리 : SFIOR의 PUD 비트 세트
 - TWI (Start) 전송 및 Ack 검사
 - TWI (Slave Address + Write) 전송 및 Ack 검사
 - TWI (1st Data) 전송 및 Ack 검사
 - 1st Data는 Configuration Register를 가르키는 값
 - TWI (Stop) 전송

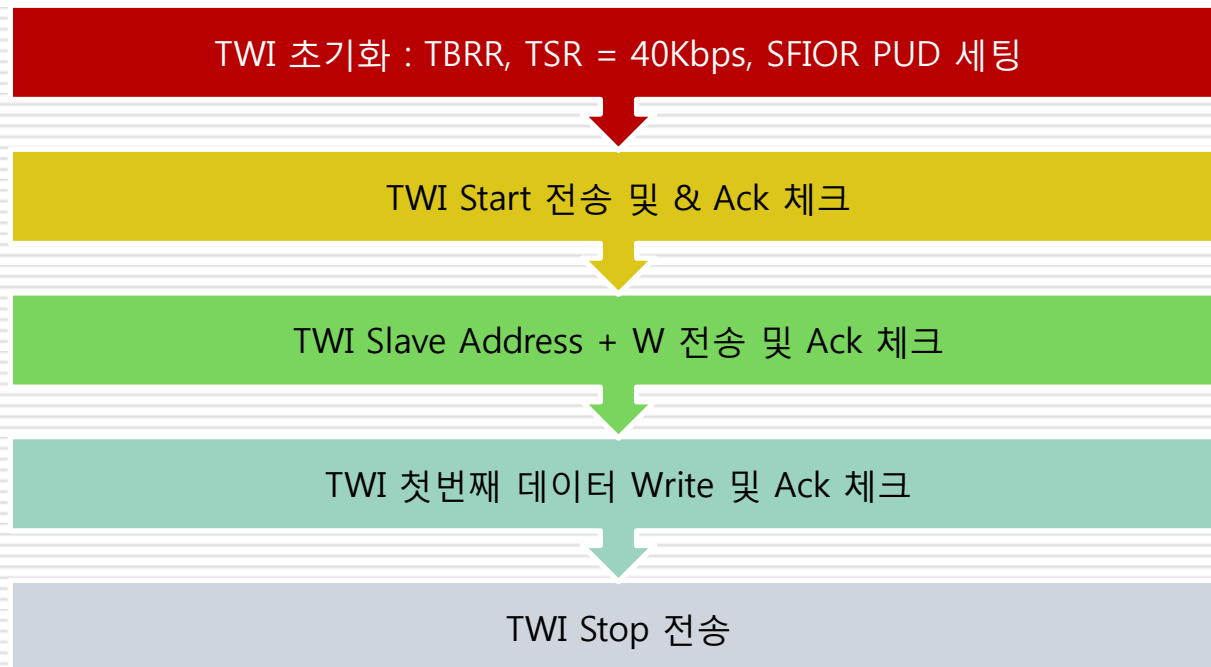
계속하여 ...

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)
 - TWI (Start) 전송 및 Ack 검사
 - TWI (Slave Address + Read) 전송 및 Ack 검사
 - TWI (1st Data) Read 및 Ack 송신
 - 1st Data는 Temperature Register High Byte 데이터 값
 - TWI (2nd Data) Read 및 Ack 송신
 - 2nd Data는 Temperature Register Low Byte 데이터 값
 - TWI (Stop) 전송

실습 TEMP-1 : TWI 인터페이스로 통신하기

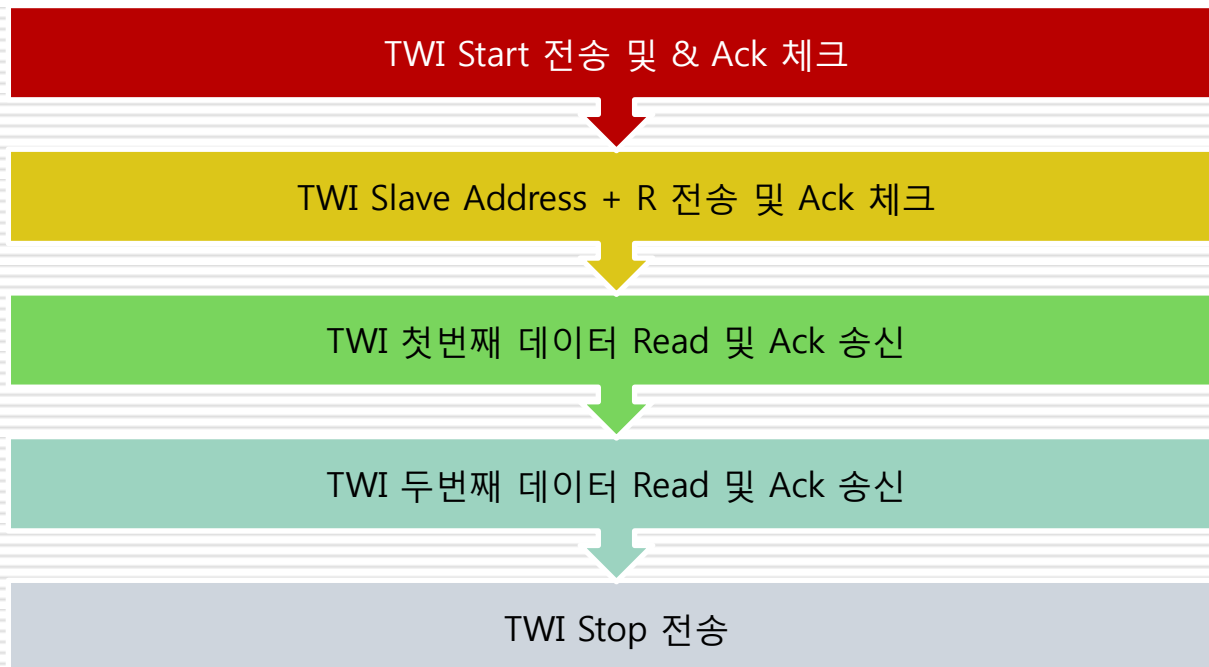
- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)



계속하여 ...

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)



실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)

```
#define F_CPU    16000000UL    // CPU 클럭 값 = 16 Mhz
#define F_SCK    40000UL      // SCK 클럭 값 = 40 Khz
#include <avr/io.h>
#include <util/delay.h>
#define UCHAR    unsigned char    // UCHAR 정의
#define USHORT   unsigned short    // USHORT 정의
#define AT575_ADDR    0x98    // 0b10011000, 7비트를 1비트 left shift
#define AT575_CONFIG_REG    1
#define AT575_TEMP_REG    0
void init_twi();
void write_twi_0byte_nopreset(UCHAR reg);
USHORT read_twi_2byte_preset();
void display_LED(UCHAR value);
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)

```
int main()
{
    USHORT temperature;
    init_twi();                // TWI 초기화
    write_twi_0byte_nopreset(ATS75_TEMP_REG);    // Temp Reg 포인팅
    temperature = read_twi_2byte_preset();        // 2바이트 온도값 읽기
    display_LED((UCHAR)(temperature>>8));        // 정수값 온도만 display
}

void init_twi()
{
    PORTD = 3;                // For Internal pull-up for SCL & SCK
    SFIOR &= ~(1<<PUD);       // PUD = Pull Up Disable
    TWBR = (F_CPU/F_SCK - 16) / 2;    // 공식 참조, bit trans rate 설정
    TWSR = TWSR & 0xfc;          // Prescaler 값 = 00 (1배)
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)

```
void write_twi_0byte_nopreset(UCHAR reg)
{
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);    // START 전송
    while((TWCR & (1 << TWINT)) == 0x00 || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));    // ACK를 기다림
    TWDR = AT75_ADDR | 0;    // SLA+W 준비, W=0
    TWCR = (1 << TWINT) | (1 << TWEN);    // SLA+W 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x18);
    TWDR = reg;    // aTS75 Reg 값 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // aTS75 Reg 값 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x28);
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);    // STOP 전송
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

□ 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)

```
USHORT read_twi_2byte_preset()
{
    UCHAR high_byte, low_byte;
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);          // START 전송
    while((TWCR & (1 << TWINT)) == 0x00 || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));                                // ACK를 기다림
    TWDR = AT575_ADDR | 1;                                       // SLA+R 준비, R=1
    TWCR = (1 << TWINT) | (1 << TWEN);                          // SLA+R 전송
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x40);
    TWCR = (1 << TWINT) | (1 << TWEN | 1 << TWEA); // 1st DATA 준비
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x50);
    high_byte = TWDR;                                             // 1st DATA 수신
    TWCR = (1 << TWINT) | (1 << TWEN | 1 << TWEA); // 2nd DATA 준비
    while((TWCR & (1 << TWINT)) == 0x00 || (TWSR & 0xf8) != 0x50);
    low_byte = TWDR;                                              // 2nd DATA 수신
    return((high_byte << 8) | low_byte);                          // 수신 DATA 리턴
}
```

실습 TEMP-1 : TWI 인터페이스로 통신하기

- 구동 프로그램 설계 : TWI 1 바이트 Write 및 2바이트 Read (temp_1_2.c)

```
void display_LED(UCHAR value)
{
    DDRA = 0xff;
    PORTA = value;
}
```

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 실습 내용

1. aTS75 온도센서를 이용하여 디지털 온도계 만들기

- 0.5도 단위까지 측정
- FND에 “(-)XY.Z” 형태로 표시

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 설계 : 디지털온도계 (temp_2_1.c)

- ATmega128의 TWI 관련 내용과 aTS75 TWI 관련 내용을 조합하여 적합한 방법과 순서로 TWI 통신 가동
- aTS75에서 온도값을 아래 순서로 Read
 1. aTS75의 Configuration Register를 적합한 값으로 Write
 - 9 bit 모드, Normal 모드 선택
 - Internal Address(0x98) & Register의 2바이트 데이터 전송
 2. aTS75의 Command Register가 Temperature Register를 포인팅하도록 세트
 3. aTS75의 Temperature Register로부터 2바이트 온도값 읽기

계속하여 ...

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

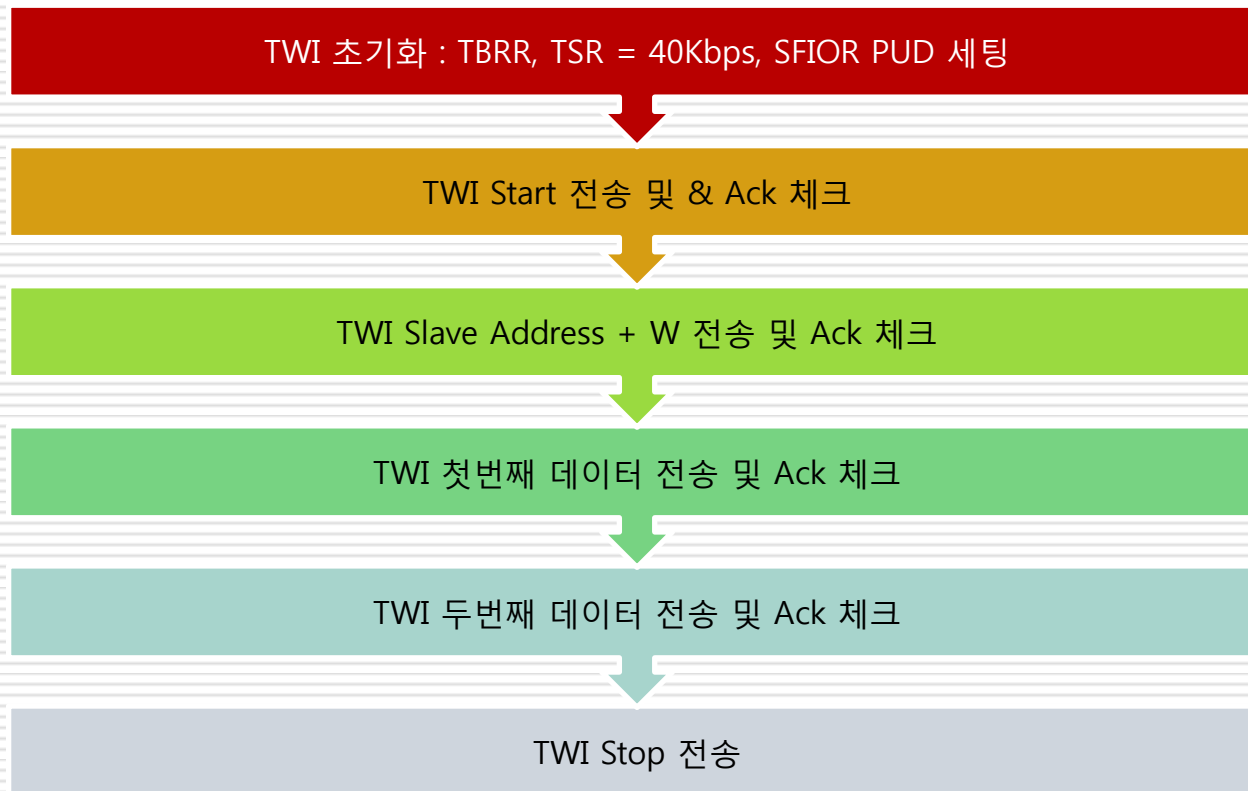
□ 구동 프로그램 설계 : 디지털온도계 (temp_2_1.c)

■ 읽은 온도값을 아래와 같이 디스플레이

1. 읽은 High Byte의 MSB 값을 확인하여 '0' 이면 FND[3]에 아무것도 디스플레이하지 않고 '1'이면 '-'를 디스플레이하고 나머지 부분을 비트 Exclusive OR 한 값으로 변경
2. High Byte 부분을 FND[2]-FND[1]에 디스플레이 하며, FND[1]에는 '.'를 함께 표시
3. FND[0]는 Low Byte의 MSB 값을 확인하여 '0'이면 '0'을 디스플레이하고, '1'이면, '5'를 디스플레이 함

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 설계 : 디지털온도계 (temp_2_1.c)



계속하여 ...

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 설계 : 디지털온도계 (temp_2_1.c)



실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
#define F_CPU    16000000UL    // CPU 클럭 값 = 16 Mhz
#define F_SCK    40000UL      // SCK 클럭 값 = 40 Khz
#include <avr/io.h>
#include <util/delay.h>
#define UCHAR    unsigned char    // UCHAR 정의
#define USHORT   unsigned short   // USHORT 정의
#define AT575_ADDR    0x98    // 0b10011000, 7비트를 1비트 left shift
#define AT575_CONFIG_REG    1
#define AT575_TEMP_REG    0
void init_twi_port();
void write_twi_1byte_nopreset(UCHAR reg, UCHAR data);
void write_twi_0byte_nopreset(UCHAR reg);
USHORT read_twi_2byte_preset();
void display_FND(USHORT value);
```

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
int main()
{
    unsigned short    temperature;
    init_twi_port();           // TWI 및 포트 초기화
    write_twi_1byte_nopreset(ATS75_CONFIG_REG, 0x00); // 9비트, Normal
    write_twi_0byte_nopreset(ATS75_TEMP_REG);        // Temp Reg 포인팅
    while (1) {temperature = read_twi_2byte_preset(); // 2바이트 온도값 읽기
                display_FND(temperature);           } // FND display
}

void init_twi_port()
{
    DDRC = 0xff; DDRG = 0xff; // FND 출력 세팅
    PORTD = 3;                // For Internal pull-up for SCL & SCK
    SFIOR &= ~(1<<PUD);       // PUD = no interrupt
    TWBR = (F_CPU/F_SCK - 16) / 2; // 공식 참조, bit trans rate 설정
    TWSR = TWSR & 0xfc;        // Prescaler 값 = 00 (1배)
}
// ATmega128 마이크로컨트롤러를 이용한 임베디드시스템 구현
```

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
void write_twi_1byte_nopreset(UCHAR reg, UCHAR data)
{
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);    // START 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));    // ACK를 기다림
    TWDR = AT75_ADDR | 0;    // SLA+W 준비, W=0
    TWCR = (1 << TWINT) | (1 << TWEN);    // SLA+W 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x18);
    TWDR = reg;    // aTS75 Reg 값 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // aTS75 Reg 값 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x28);
    TWDR = data;    // DATA 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // DATA 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x28);
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);    // STOP 전송
}
```

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
void write_twi_0byte_nopreset(UCHAR reg)
{
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);    // START 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));    // ACK를 기다림
    TWDR = AT575_ADDR | 0;    // SLA+W 준비, W=0
    TWCR = (1 << TWINT) | (1 << TWEN);    // SLA+W 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x18);
    TWDR = reg;    // aTS75 Reg 값 준비
    TWCR = (1 << TWINT) | (1 << TWEN);    // aTS75 Reg 값 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x28);
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);    // STOP 전송
}
```

실습 : 온도센서를 이용하여 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
USHORT read_twi_2byte_preset()
{
    UCHAR high_byte, low_byte;
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);    // START 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || ((TWSR & 0xf8) != 0x08 &&
        (TWSR & 0xf8) != 0x10));                          // ACK를 기다림
    TWDR = AT75C55_ADDR | 1;                               // SLA+R 준비, R=1
    TWCR = (1 << TWINT) | (1 << TWEN);                     // SLA+R 전송
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x40);
    TWCR = (1 << TWINT) | (1 << TWEN | 1 << TWEA); // 1st DATA 준비
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x50);
    high_byte = TWDR;                                       // 1 byte DATA 수신
    TWCR = (1 << TWINT) | (1 << TWEN | 1 << TWEA); // 2nd DATA 준비
    while(((TWCR & (1 << TWINT)) == 0x00) || (TWSR & 0xf8) != 0x50);
    low_byte = TWDR;                                        // 1 byte DATA 수신
    return((high_byte<<8) | low_byte);                      // 수신 DATA 리턴
}
```


실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
void display_FND(USHORT value)
{
    UCHAR digit[12] = {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x07, 0x7f,
0x67, 0x40, 0x00};
    UCHAR fnd_sel[4] = {0x08, 0x04, 0x02, 0x01};

    UCHAR value_int, value_deci, num[4];
    int i;
    if((value & 0x8000) != 0x8000)                // Sign 비트 체크
        num[3] = 10;
    else
    {
        num[3] = 11;
        value = (~value)-1;                        // 2's Compliment
    }
}
```

실습 TEMP-2 : 온도센서로 디지털온도계 만들기

□ 구동 프로그램 코딩 : 디지털온도계 (temp_2_1.c)

```
value_int = (UCHAR)((value & 0x7f00) >> 8);
value_deci = (UCHAR)(value & 0x00ff);
num[2] = (value_int / 10) % 10;
num[1] = value_int % 10;
num[0] = ((value_deci & 0x80) == 0x80) * 5;
for(i=0; i<4; i++)
{
    PORTC = digit[num[i]];
    PORTG = fnd_sel[i];
    if(i==1)
        PORTC |= 0x80;
    _delay_ms(2);
}
}
```

문고 답하기

Q & A

