
# AVR – HW9

임베디드스쿨1기

Lv1과정

2020. 11. 13

강경수

# 1. I2C 통신 예제

▣ i2c 복습

2020.11.10 강경수

```
void i2c_init(void)
{
    /*initallize TWI clock : 100kHz clock, TWPS = 0 -> PRESCALER = 1 */
    TWSR = 0x00;
    TWBR = 12;
}
```

**TWSR – TWI Status Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xB9) | 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 | |

- **Bits 7..3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire serial bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \times (\text{PrescalerValue})}$$

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

# 1. I2C 통신 예제

Master Transmitter Mode 에서의 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) |

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 |

```
unsigned char i2c_start(unsigned char address)
{
    uint8_t   twst;

    // send START condition
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

    // wait until transmission completed
    while(!(TWCR & (1<<TWINT)));

    // check value of TWI Status Register. Mask prescaler bits.
    twst = TWSR & 0xF8;
    if ( (twst != TWI_START) && (twst != TWI_RESTART)) return 1;

    // send device address
    TWDR = address;
    TWCR = (1<<TWINT) | (1<<TWEN);

    // wail until transmission completed and ACK/NACK has been received
    while(!(TWCR & (1<<TWINT)));

    // check value of TWI Status Register. Mask prescaler bits.
    twst = TWSR & 0xF8;
    if ( (twst != TWI_MT_SLA_ACK) && (twst != TWI_MR_SLA_NACK) ) return 1;

    return 0;
}
```

**TWCR – TWI Control Register** SOFTWARE 1시 0으로 CLEAR됨!!

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xBC) | 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 | | | | | | | | | |

'The TWINT flag must be cleared by software by writing a logic one to it.

TWINT 가 1로 SET될때까지 기다린다

<- 하위 3BIT MASK 용도

START도 RESTART도 아니면 1반환

TWINT 가 1로 SET될때까지 기다린다

# 1. I2C 통신 예제

- TWINT를 0으로 클리어 하고 TWSTA를 1로 하여 START 조건 발생
- TWINT 1로 SET될때까지 대기 하며TWSR 상위 5BIT에 START완료 됐는지 확인
- TWDR에 ADDRESS 지정후 TWINT 다시 0으로 CLEAR하고 TWI통신 ENABLE
- TWINT 1로 SET될때까지 대기 하며TWSR 상위 5BIT에 ACK, NACK완료 됐는지 확인

```
unsigned char i2c_write( unsigned char data)
{
    uint8_t twst;

    TWDR = data;
    TWCR = (1<<TWINT) | (1<<TWEN);

    while(!(TWCR & (1<<TWINT)));

    twst = TWSR & 0xF8;
    if(twst != TWI_MT_DATA_ACK) return 1;
    return 0;
}
unsigned char i2c_readAck(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);

    while(!(TWCR & (1<<TWINT)));

    return TWDR;
}
```

WRITE한 후 ACK반환되면 0
ACK반환 완되면 1

ACK 혹은 NACK PULSE를 발생시키고
SLAVE로 부터 READ 한 데이터를 반환
하기 위한 함수.

Q. readNak를 하면
기존에 저장돼 있는 TWDR DATA
가 WRITE되는것은 아닌지 궁금해요
(i2c_readNak 와 i2c_write함수 유사함)

# 1. I2C 통신 예제

```
unsigned char i2c_readNak(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));

    return TWDR;
}
```

| | Command byte | | | | | | | | hex value |
|---|---|---|---|---|---|---|---|---|---|
| Bit number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| Bit name | PRM | COV | - | Typ | Ad2/Os2 | Ad1/Os1 | Ad0/Os0 | Stop | |
| Command | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0x1E |
| Convert D1 (OSR=256) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40 |
| Convert D1 (OSR=512) | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0x42 |
| Convert D1 (OSR=1024) | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0x44 |
| Convert D1 (OSR=2048) | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0x46 |
| Convert D1 (OSR=4096) | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0x48 |
| Convert D2 (OSR=256) | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0x50 |
| Convert D2 (OSR=512) | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0x52 |
| Convert D2 (OSR=1024) | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0x54 |
| Convert D2 (OSR=2048) | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0x56 |
| Convert D2 (OSR=4096) | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0x58 |
| ADC Read | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x00 |
| PROM Read | 1 | 0 | 1 | 0 | Ad2 | Ad1 | Ad0 | 0 | 0xA0 to 0xAE |

# 2. USING MS5611

▣ MS5611분석

2020.11.10 강경수

```
void ms5611_reset(void)
{
    i2c_start( (MS5611_ADDR << 1) | I2C_WRITE );
    i2c_write( RESET );
    i2c_stop();
    _delay_ms(10);
}
```

Q. 그냥 0xEC 를 ADDRESS로 하면 안되나요?

추가) 아마 I2C_READ사용할때 코드 일관성을 위해서 인듯

```
#define MS5611_ADDR 0x76
```

111011Cx,

0x76 => 01110110 << 1

-> 11101100

이때 CSB 핀 GND 면 0

VCC 면 1

write함수 명령어

| | Command byte | | | | | | | | hex value |
|---|---|---|---|---|---|---|---|---|---|
| Bit number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| Bit name | PR M | COV | - | Typ | Ad2/ Os2 | Ad1/ Os1 | Ad0/ Os0 | Stop | |
| Command | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0x1E |
| Convert D1 (OSR=256) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40 |
| Convert D1 (OSR=512) | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0x42 |
| Convert D1 (OSR=1024) | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0x44 |
| Convert D1 (OSR=2048) | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0x46 |
| Convert D1 (OSR=4096) | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0x48 |
| Convert D2 (OSR=256) | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0x50 |
| Convert D2 (OSR=512) | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0x52 |
| Convert D2 (OSR=1024) | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0x54 |
| Convert D2 (OSR=2048) | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0x56 |
| Convert D2 (OSR=4096) | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0x58 |
| ADC Read | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x00 |
| PROM Read | 1 | 0 | 1 | 0 | Ad2 | Ad1 | Ad0 | 0 | 0xA0 to 0xAE |

# 2. USING MS5611

**RESET SEQUENCE**

The Reset sequence shall be sent once after power-on to make sure that the calibration PROM gets loaded into the internal register. It can be also used to reset the device ROM from an unknown condition

Data sheet에 명시 된 것처럼 power on 이후 reset 한번 해줘야함

```
struct _ms5611_cal {
    uint16_t sens, off, tcs, tco, tref, tsens;
} ms5611_cal;
```

구조체 선언과 동시에 ms5611_cal 라는 구조체 변수 선언.

```
void ms5611_init(void)
{

    ms5611_reset();
    UART_string_transmit("ms5611 reset ok\n");

    ms5611_cal.sens  = ms5611_read_cal_reg(1);
    ms5611_cal.off   = ms5611_read_cal_reg(2);
    ms5611_cal.tcs   = ms5611_read_cal_reg(3);
    ms5611_cal.tco   = ms5611_read_cal_reg(4);
    ms5611_cal.tref  = ms5611_read_cal_reg(5);
    ms5611_cal.tsens = ms5611_read_cal_reg(6);

    _delay_ms(1000);
}
```

공장 calibrate 값 불러오기

이 calibtrate값 수정해서

더 정확한 data수령 가능

산업용 계측기 옆에 두고

cal값 맞추면 좋을듯

# 2. USING MS5611

```
uint32_t ms5611_read_cal_reg(uint8_t reg)
{
   uint8_t PROM_dat1;
   uint8_t PROM_dat2;

   uint16_t data;

   i2c_start( (MS5611_ADDR << 1) | I2C_WRITE );
   i2c_write(MS5611_CMD_PROM(reg));
   i2c_rep_start(MS5611_ADDR <<1 | I2C_READ);

   PROM_dat1 = i2c_readAck();
   PROM_dat2 = i2c_readNak();
   i2c_stop();

   printf("PROM_dat1:%d, %d\n", PROM_dat1, PROM_dat2);
   /*
      PROM_dat1:180, 246
      PROM_dat1:188, 144
      PROM_dat1:111, 211
      PROM_dat1:101, 87
      PROM_dat1:126, 66
      PROM_dat1:108, 68
    */
    data = ( PROM_dat1 << 8 ) + (uint16_t)PROM_dat2;
```

```
#define MS5611_CMD_PROM(reg)    (0xA0 + ((reg) << 1))
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| PROM Read | 1 | 0 | 1 | 0 | Ad2 | Ad1 | Ad0 | 0 | 0xA0 to 0xAE |

Ack 신호를 보내주면 센서가 data를 뱉는다.

Q. Nak 신호 줘도 유의미한 data 반환 안하는거 같은데 어떤의도에서 dat2 에 저장하는지 모르겠습니다.

# 2. USING MS5611

**Read calibration data (factory calibrated) from PROM**

| Variable | Description \| Equation | Recommended variable type | Size [1] [bit] | Value min | Value max | Example / Typical |
|---|---|---|---|---|---|---|
| *C1* | Pressure sensitivity \| $SENS_{T1}$ | unsigned int 16 | 16 | 0 | 65535 | 40127 |
| *C2* | Pressure offset \| $OFF_{T1}$ | unsigned int 16 | 16 | 0 | 65535 | 36924 |
| *C3* | Temperature coefficient of pressure sensitivity \| TCS | unsigned int 16 | 16 | 0 | 65535 | 23317 |
| *C4* | Temperature coefficient of pressure offset \| TCO | unsigned int 16 | 16 | 0 | 65535 | 23282 |
| *C5* | Reference temperature \| $T_{REF}$ | unsigned int 16 | 16 | 0 | 65535 | 33464 |
| *C6* | Temperature coefficient of the temperature \| TEMPSENS | unsigned int 16 | 16 | 0 | 65535 | 28312 |

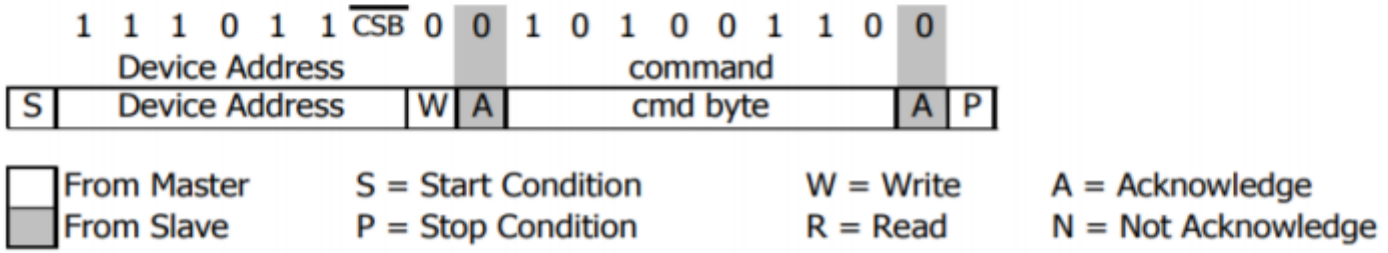


Figure 11: I²C Command to read memory address= 011 (Coefficient 3)

??

NAK 뒤에 SLAVE측에서 뱉는 DATA?

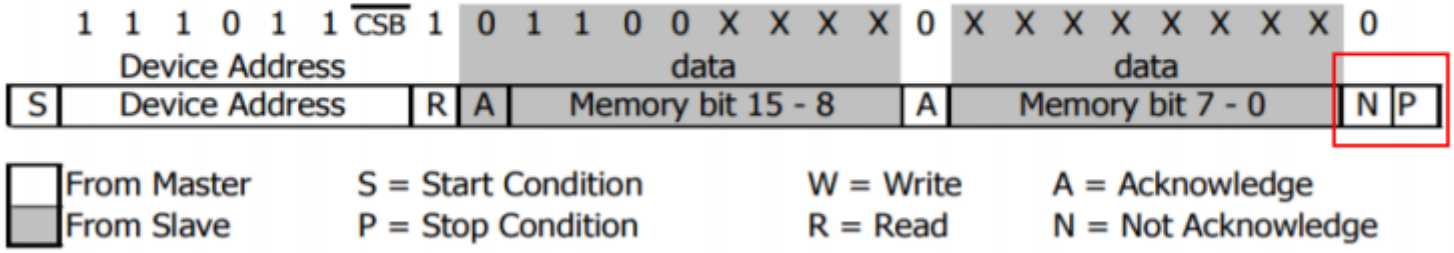


Figure 12: I²C answer from MS5611-01BA

# 2. USING MS5611

```
void ms5611_measure(void)
{
    int32_t temp_raw, press_raw, dt;
    int64_t sens, off;

    temp_raw = ms5611_conv_read_adc(CONV_D2_4096);
    press_raw = ms5611_conv_read_adc(CONV_D1_4096);

    dt = temp_raw - ((int32_t)ms5611_cal.tref << 8);
    _ms5611_temp = 2000 + ((dt*((int64_t)ms5611_cal.tsens)) >> 23);
    off = ((int64_t)ms5611_cal.off << 16) +
          (((int64_t)dt*(int64_t)ms5611_cal.tco) >> 7);
    sens = ((int64_t)ms5611_cal.sens << 15) +
           ((int64_t)ms5611_cal.tcs*dt >> 8);
    _ms5611_pres = ((((uint64_t)press_raw*sens) >> 21) - off) >> 15
}
```

0x48 0x58 ADC RESOLUTION 1024

더 정밀한 값을 측정할 수 있지만 시간 오래걸림

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Convert D1 (OSR=4096) | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0x48 |
| Convert D2 (OSR=4096) | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0x58 |

값 보정 및 계산에 대해서는 내일 수업전까지 더 공부해서 정리해서 올게요