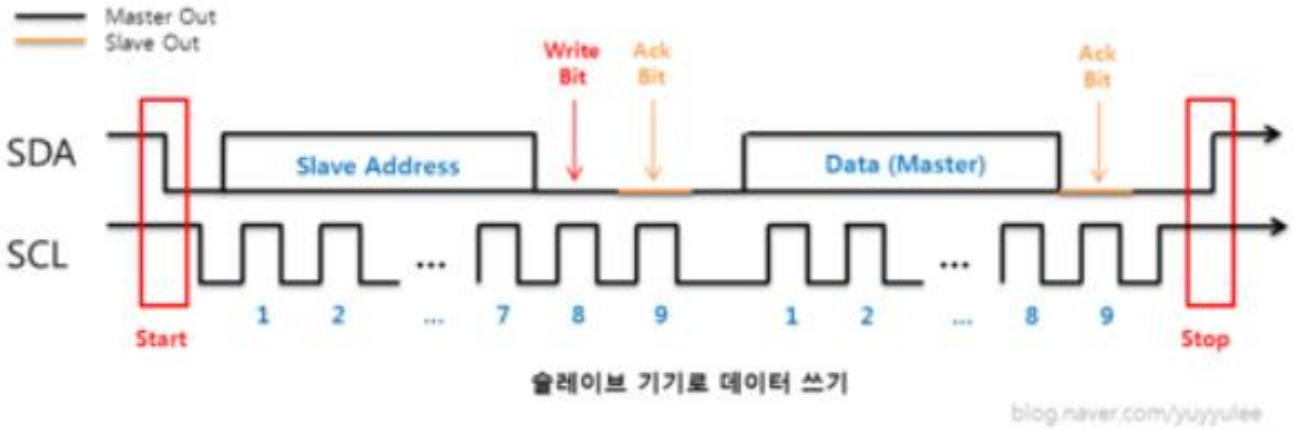
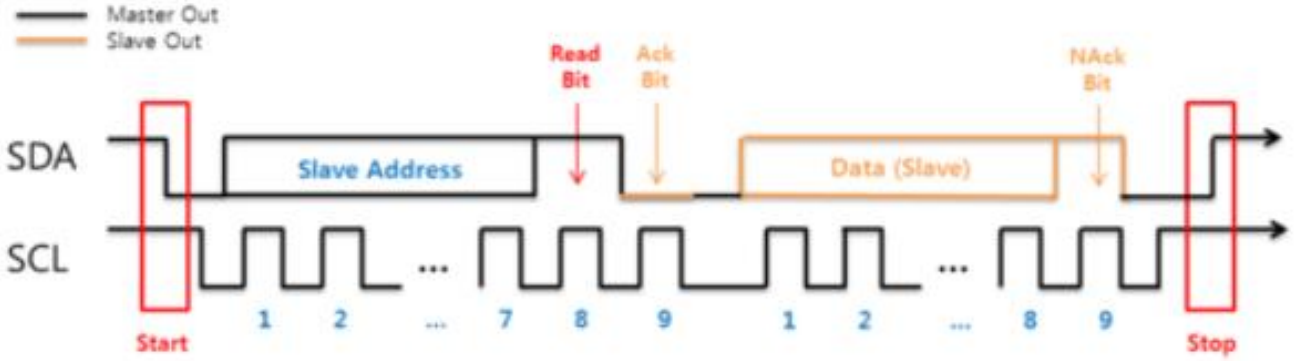
AVR - HW9

임베디드스쿨1기
Lv1과정
2020. 11. 13
박성환

1. Review-TWI



슬레이브 기기로 데이터 쓰기



슬레이브 기기에서 데이터 읽기

응답 신호는 기본적으로 ACK(=LOW)이나 슬레이브가 데이터 전송하는데 모든 데이터 전송이 끝난 경우 NACK로 하는점이 특이함 (가끔 I2C 보면 있기도 없기도 함)

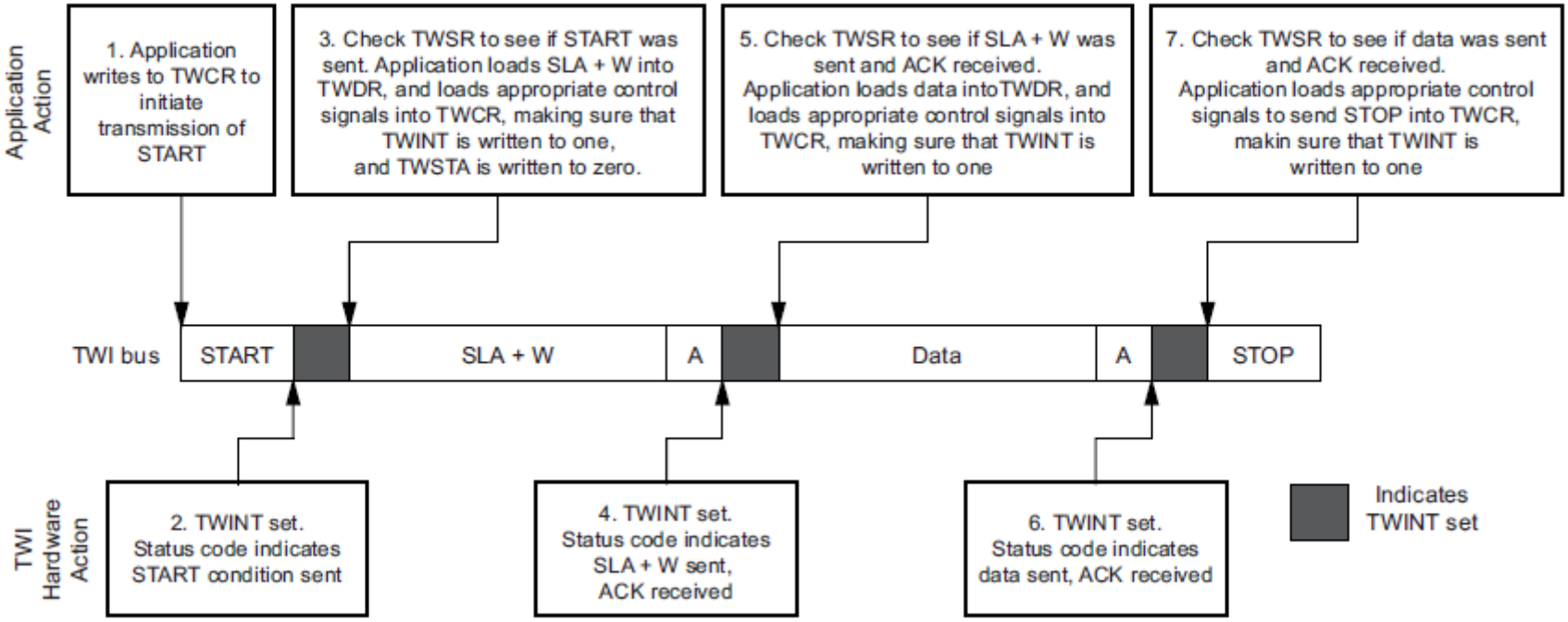
7bit address인데 보통 0번지는 잘안쓰고 127대 정도까지 가능

bus가 충돌나는 경우에 대한 대비책이 있는지 모르겠음(Can은 버스 충돌 중지 기능이 있다고 들었는데...)

I2C는 누가 받는 상관없이 막 그냥 쏘는거 잘받으니 일종의 Broad Casting 방식이라고도 할 수 있지 않을까 ACK를 받으니 아니라고 해야하나?

1. Review-TWI

Figure 21-10. Interfacing the Application to the TWI in a Typical Transmission



중요한 포인트는 TWINT를 통해 제대로 보내줬다는 것을 확인 가능하다는 점!

2. TWI – MS5611

- **기압센서(Barometric Pressure Sensor)**
- 온도, 기압, 고도 계산 가능
- SPI, I2C 통신 둘다 가능함

SPECIFICATIONS

- High resolution module, 10 cm
- Fast conversion down to 1 ms
- Low power, 1 µA (standby < 0.15 µA)
- QFN package 5.0 x 3.0 x 1.0 mm³
- Supply voltage 1.8 to 3.6 V
- Integrated digital pressure sensor (24 bit ΔΣ ADC)
- Operating range: 10 to 1200 mbar, -40 to +85 °C
- I²C and SPI interface up to 20 MHz
- No external components (Internal oscillator)
- Excellent long-term stability

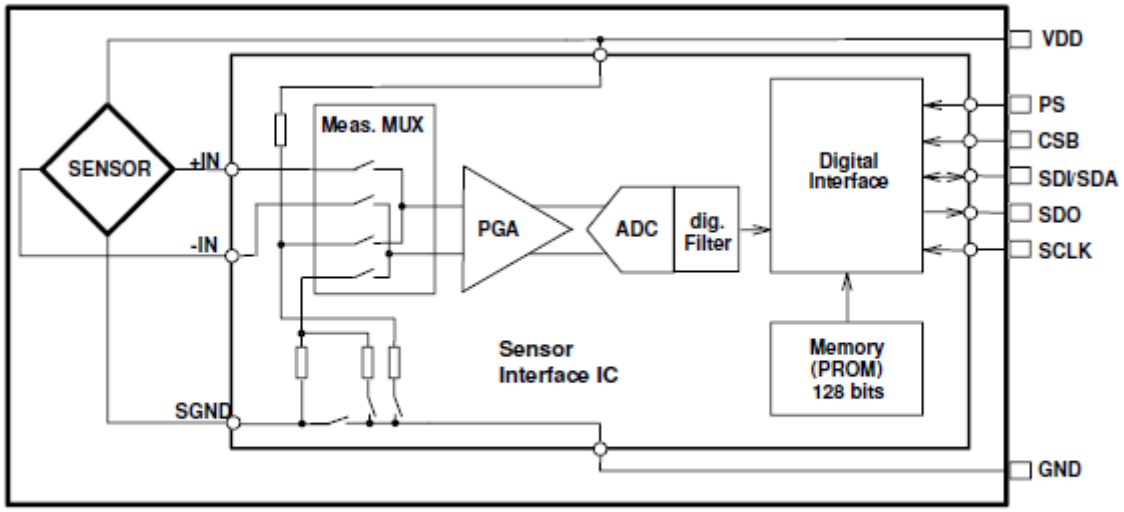
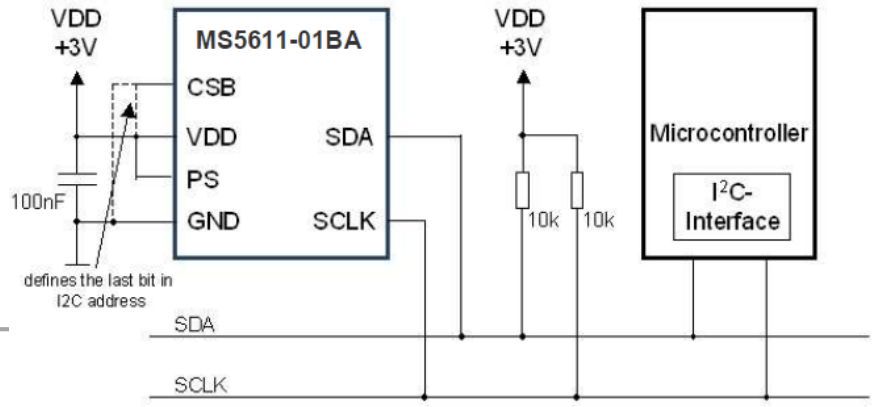


Figure 1: Block diagram of MS5611-01BA

TECHNICAL DATA

Sensor Performances (V_{DD} = 3 V)				
Pressure	**Min**	**Typ**	**Max**	**Unit**
Range	10		1200	mbar
ADC		24		bit
Resolution (1)		0.065 / 0.042 / 0.027 / 0.018 / 0.012		mbar
Accuracy 25°C, 750 mbar	-1.5		+1.5	mbar
Error band, -20°C to +85°C, 450 to 1100 mbar (2)	-2.5		+2.5	mbar
Response time (1)		0.5 / 1.1 / 2.1 / 4.1 / 8.22		ms
Long term stability		±1		mbar/yr
Temperature	**Min**	**Typ**	**Max**	**Unit**
Range	-40		+85	°C
Resolution		<0.01		°C
Accuracy	-0.8		+0.8	°C
Notes: (1) Oversampling Ratio: 256 / 512 / 1024 / 2048 / 4096 (2) With autozero at one pressure point				

I²C protocol communication



2. TWI – MS5611

계산 순서는 옆과 같이 진행

```
)ms5611_cal.tref << 8);
t*((int64_t)ms5611_cal.tsens)) >> 23);
l.off << 16) + (((int64_t)dt*(int64_t)ms5611_cal.tco) >> 7);
al.sens << 15) + ((int64_t)ms5611_cal.tcs*dt >> 8);
_t)press_raw*sens) >> 21) - off) >> 15;
```

계산 식 확인 가능

PRESSURE AND TEMPERATURE CALCULATION

Start
Maximum values for calculation results:
P_{MIN} = 10mbar P_{MAX} = 1200mbar
T_{MIN} = -40°C T_{MAX} = 85°C T_{REF} = 20°C

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

Read digital pressure and temperature data

D1	Digital pressure value	unsigned int 32	24	0	16777216	9085466
D2	Digital temperature value	unsigned int 32	24	0	16777216	8569150

Calculate temperature

dT	Difference between actual and reference temperature [2] $dT = D2 - T_{REF} = D2 - C5 * 2^8$	signed int 32	25	-16776960	16777216	2366
TEMP	Actual temperature (-40…85°C with 0.01°C resolution) $TEMP = 20°C + dT * TEMPSENS = 2000 + dT * C6 / 2^{23}$	signed int 32	41	-4000	8500	2007 = 20.07 °C

Calculate temperature compensated pressure

OFF	Offset at actual temperature [3] $OFF = OFF_{T1} + TCO * dT = C2 * 2^{16} + (C4 * dT) / 2^7$	signed int 64	41	-8589672450	12884705280	2420281617
SENS	Sensitivity at actual temperature [4] $SENS = SENS_{T1} + TCS * dT = C1 * 2^{15} + (C3 * dT) / 2^8$	signed int 64	41	-4294836225	6442352640	1315097036
P	Temperature compensated pressure (10…1200mbar with 0.01mbar resolution) $P = D1 * SENS - OFF = (D1 * SENS / 2^{21} - OFF) / 2^{15}$	signed int 32	58	1000	120000	100009 = 1000.09 mbar

Display pressure and temperature value

Notes
[1] Maximal size of intermediate result during evaluation of variable
[2] min and max have to be defined
[3] min and max have to be defined
[4] min and max have to be defined

Figure 2: Flow chart for pressure and temperature reading and software compensation.

2. TWI – MS5611

- 5가지 기본 명령어가 있음

COMMANDS

The MS5611-01BA has only five basic commands:

1. Reset
2. Read PROM (128 bit of calibration words)
3. D1 conversion
4. D2 conversion
5. Read ADC result (24 bit pressure / temperature)

	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

Figure 4: Command structure

각 command 명령값 확인

2. TWI – MS5611

RESET SEQUENCE

The reset can be sent at any time. In the event that there is not a successful power on reset this may be caused by the SDA being blocked by the module in the acknowledge state. The only way to get the MS5611-01BA to function is to send several SCLKs followed by a reset sequence or to repeat power on reset.

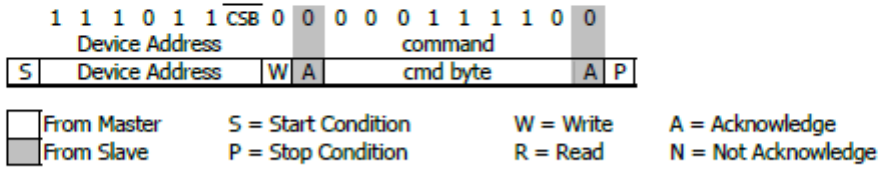


Figure 10: I²C Reset Command

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

RESET 명령값은 앞페이지 참조
CSB VCC로 설정해서 1이기 때문에 /CSS는 0임
쓰기모드로 설정

2. TWI – MS5611

PROM READ SEQUENCE

The PROM Read command consists of two parts. First command sets up the system into PROM read mode. The second part gets the data from the system.

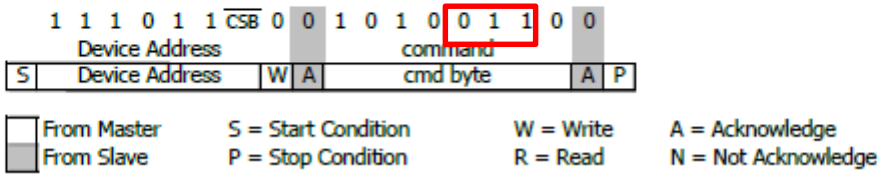


Figure 11: I²C Command to read memory address= 011 (Coefficient 3)

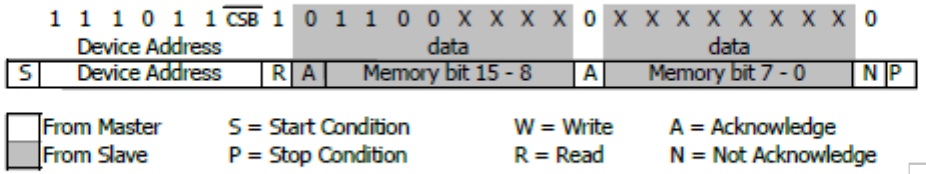


Figure 12: I²C answer from MS5611-01BA

Read 신호 맨 마지막 N(=NAK) 확인

```
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);
```

	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

Figure 4: Command structure

PROM이기 때문에 공장에서 한번 writing 한게 끝이기 때문에 쓰지는 못하고 읽기만 가능(시대가 어느때인데 아직도 PROM을 쓰는 이유가 있을까? EEPROM 놔두고?
=> 특성상 EEPROM에 값을 저장할 필요는 없을것 같지만....)

2. TWI – MS5611

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

   uint16_t data;

   /* First command sets up the system into PROM read mode */
   i2c_start( (MS5611_ADDR << 1) | I2C_WRITE );
   i2c_write(MS5611_CMD_PROM(reg)); //이어가기 때문에 stop 하지 않고 다음에 rep_start 진행함

   /* Second part gets the data from the system */
   i2c_rep_start(MS5611_ADDR <<1 | I2C_READ);
   PROM_dat1 = i2c_readAck(); //slave로 부터 데이터를 입력받을 때까지 기다린 후 Ack 신호
   PROM_dat2 = i2c_readNak(); //slave로 부터 데이터를 입력받을 때까지 기다린 후 Nack 신호(Datasheet에 명시됨)
   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;

   return data;
}
```

주석 확인

16bit 데이터 처리

2. TWI – MS5611

CONVERSION SEQUENCE

A conversion can be started by sending the command to MS5611-01BA. When command is sent to the system it stays busy until conversion is done. When conversion is finished the data can be accessed by sending a Read command, when an acknowledge appears from the MS5611-01BA, 24 SCLK cycles may be sent to receive all result bits. Every 8 bit the system waits for an acknowledge signal.

	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

Figure 4: Command structure

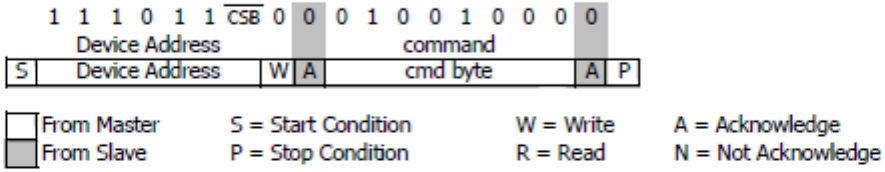


Figure 13: I²C Command to initiate a pressure conversion (OSR=4096, typ=D1)

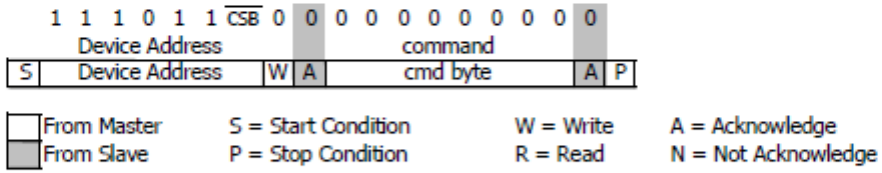


Figure 14: I²C ADC read sequence

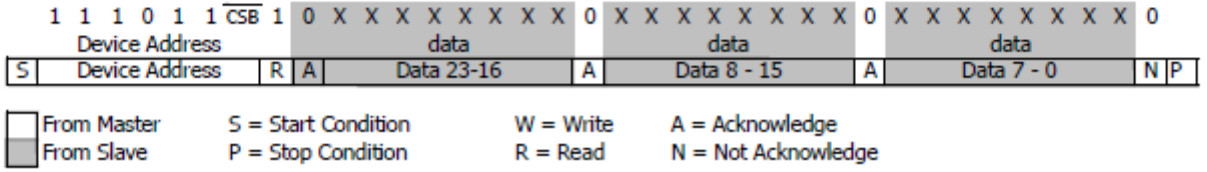


Figure 15: I²C answer from MS5611-01BA

OSR은 샘플링을 의미하며 앞페이지 명령어 값 내용 확인하기

2. TWI – MS5611

```
uint32_t ms5611_conv_read_adc(uint8_t command)
{

   uint8_t rv1;
   uint8_t rv2;
   uint8_t rv3;

   uint32_t adc_data;

   i2c_start( (MS5611_ADDR << 1) | I2C_WRITE );
   i2c_write(command);
   i2c_stop();
   _delay_ms(10); //conversion Time delay


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

   rv1 = i2c_readAck();
   rv2 = i2c_readAck();
   rv3 = i2c_readNak();
   i2c_stop();

   adc_data = ((uint32_t)rv1 << 16) + ((uint32_t)rv2 << 8) + (uint32_t)rv3;
   /*
   printf("rv1 : %d\n", rv1);
   printf("rv2 : %d\n", rv2);
   printf("rv3 : %d\n", rv3);
   */

   /*
   rv1 : 135
   rv2 : 28
   rv3 : 72
   */

   return adc_data;
}
```

3byte 데이터 수신 from Slave, 마지막 Nak

24bit ADC이므로 해당 처리

2. TWI – MS5611

```
double ms5611_getAltitude(void)
{
    double alt;
    alt = (1 - pow(_ms5611_pres/(double)101325, 0.1903)) / 0.0000225577;
    return alt;
}
```

기압 -> 고도 변환 공식 적용(아래 참조)

Air is assumed to be dry and at 20°C. The site elevation is used to calculate standard atmospheric pressure using the equation for 'standard atmospheric' pressure on p 6.1 of the ASHRAE 1997 HOF.
Density = P / (R * T)
Where:
R is the Gas constant = 287.05 J/kg-K
T is the temperature in K. Air is assumed to be at 20°C so, T = (20 + 273.15)
P is standard pressure:
P = 101325 x (1.0 – Z x 0.0000225577)5.2559
where:
Z = Elevation above sea level (m)

위의 코드 공식과 동일

From the user, a station pressure (P_{sta}) is given. To calculate the pressure altitude, the station pressure must be in units of millibars (mb). To see how to convert station pressure to millibars see the link below:

Pressure Conversion

Then, the pressure altitude (h_{alt}) can be calculated using the equation below:

$$h_{alt} = \left(1 - \left(\frac{P_{sta}}{1013.25}\right)^{0.190284}\right) \times 145366.45$$

이 공식은 자세히 보니 Psta (=Station Pressure)로
Barometric Pressure(=기압) 과 약간 다른 개념임(사용x)

The answer will be in units of feet. To convert the answer to units of meters, see the equation below:

$$h_m = 0.3048 \times h_{alt}$$

감사합니다.