TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

- BMP280 -

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

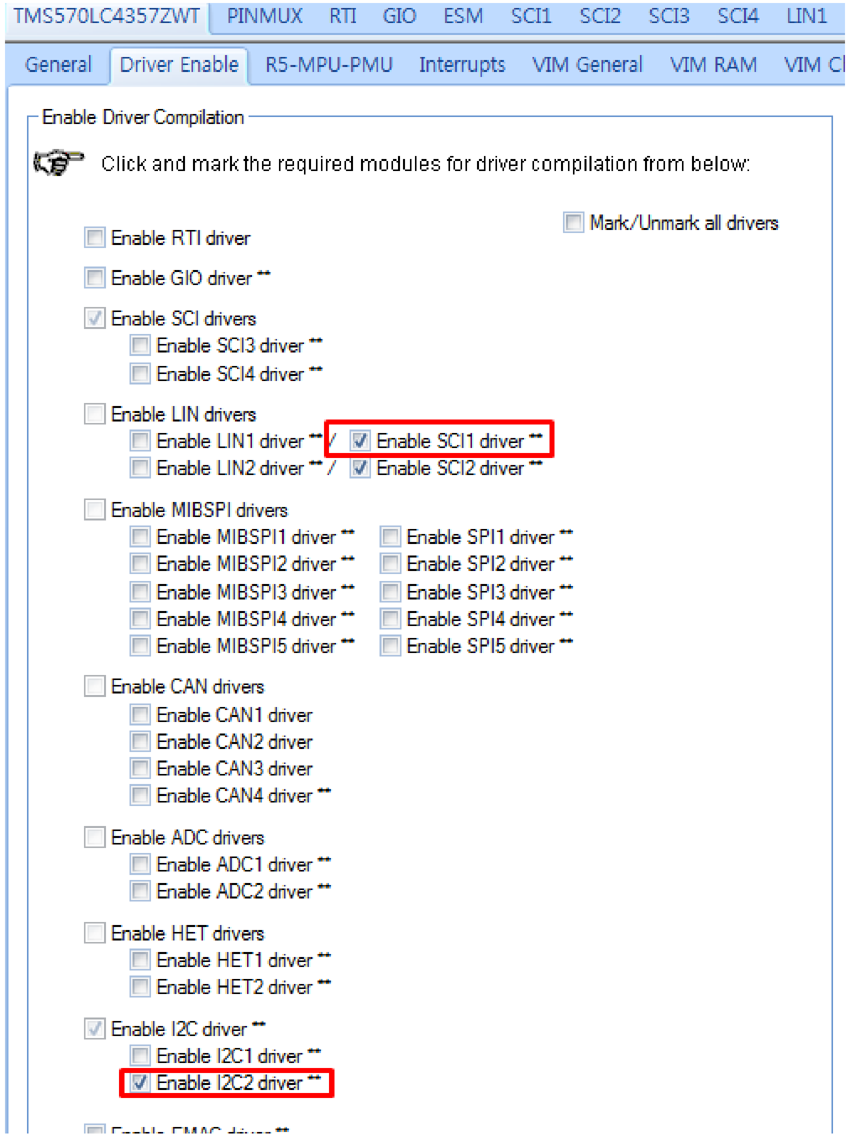
목차

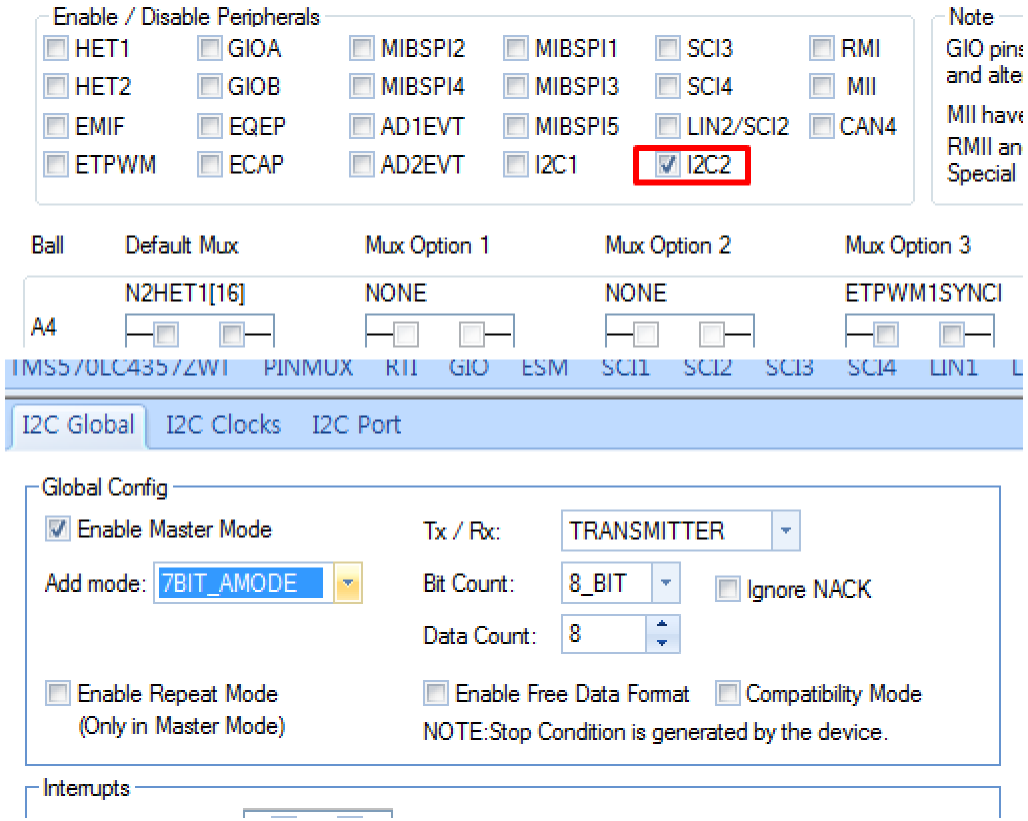
BMP280

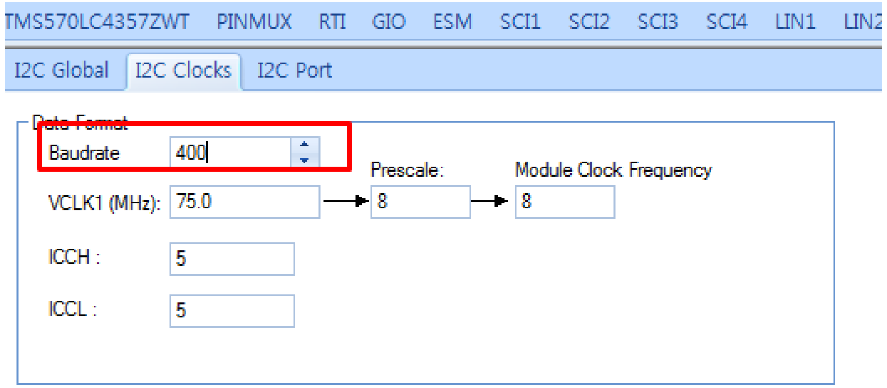
1. HALCOGEN 설정
2. Register
3. CODE (주석을 많이 참고 바람.)
4. SIMULATION

ps. MPU9250 datasheet, zip파일의 추가된 코드와 MPU9250.h를 참고하여 꼭 같이 보세요.

1. HALCOGEN 설정







2. Register

```
BMP280_REGISTER_CONTROL, 0x3F    0011 1111
```

ctrl_meas	0xF4	osrs_t[2:0]	osrs_p[2:0]	mode[1:0]	0x00

3.6 Power modes

The BMP280 offers three power modes: sleep mode, forced mode and normal mode. These can be selected using the mode[1:0] bits in control register 0xF4.

Table 10: *mode* settings

***mode*[1:0]**	**Mode**
00	Sleep mode
01 and 10	Forced mode
11	Normal mode

3.3.2 Temperature measurement

Temperature measurement can be enabled or skipped. Skipping the measurement could be useful to measure pressure extremely rapidly. When enabled, several oversampling options exist. Each oversampling step reduces noise and increases the output resolution by one bit, which is stored in the XLSB data register 0xFC. Enabling/disabling the temperature measurement and oversampling setting are selected through the osrs_t[2:0] bits in control register 0xF4.

Table 5: *osrs_t* settings

***osrs_t*[2:0]**	**Temperature oversampling**	**Typical temperature resolution**
000	Skipped (output set to 0x80000)	–
001	×1	16 bit / 0.0050 °C
010	×2	17 bit / 0.0025 °C
011	×4	18 bit / 0.0012 °C
100	×8	19 bit / 0.0006 °C
101, 110, 111	×16	20 bit / 0.0003 °C

4.3.4 Register 0xF4 "*ctrl_meas*"

The "*ctrl_meas*" register sets the data acquisition options of the device.

Table 20: Register 0xF4 "*ctrl_meas*"

Register 0xF4 "*ctrl_meas*"	**Name**	**Description**
Bit 7, 6, 5	osrs_t[2:0]	Controls oversampling of temperature data. See chapter 3.3.2 for details.
Bit 4, 3, 2	osrs_p[2:0]	Controls oversampling of pressure data. See chapter 3.3.1 for details.
Bit 1, 0	mode[1:0]	Controls the power mode of the device. See chapter 3.6 for details.

Table 21: register settings *osrs_p*

***osrs_p*[2:0]**	**Pressure oversampling**
000	Skipped (output set to 0x80000)
001	oversampling ×1
010	oversampling ×2
011	oversampling ×4
100	oversampling ×8
101, Others	oversampling ×16

3. CODE 1

```
void BMP280_Init(void){
    if (read8(BMP280_REGISTER_CHIPID) != 0x58){
        disp_set("Wrong chipId");
        while(1);
    }
    bmp280_readCoefficients();
    writeByte(BMP280_ADDRESS, BMP280_REGISTER_CONTROL, 0x3F);  // 01 1111
}
```

temp_lsb	0xFB	temp_lsb<7:0>	0x00
temp_msb	0xFA	temp_msb<7:0>	0x80

```
void bmp280_readCoefficients(void)
{

  _bmp280_calib.dig_T1 = read16_LE(BMP280_REGISTER_DIG_T1);
  _bmp280_calib.dig_T2 = readS16_LE(BMP280_REGISTER_DIG_T2);
  _bmp280_calib.dig_T3 = readS16_LE(BMP280_REGISTER_DIG_T3);

  _bmp280_calib.dig_P1 = read16_LE(BMP280_REGISTER_DIG_P1);
  _bmp280_calib.dig_P2 = readS16_LE(BMP280_REGISTER_DIG_P2);
  _bmp280_calib.dig_P3 = readS16_LE(BMP280_REGISTER_DIG_P3);
  _bmp280_calib.dig_P4 = readS16_LE(BMP280_REGISTER_DIG_P4);
  _bmp280_calib.dig_P5 = readS16_LE(BMP280_REGISTER_DIG_P5);
  _bmp280_calib.dig_P6 = readS16_LE(BMP280_REGISTER_DIG_P6);
  _bmp280_calib.dig_P7 = readS16_LE(BMP280_REGISTER_DIG_P7);
  _bmp280_calib.dig_P8 = readS16_LE(BMP280_REGISTER_DIG_P8);
  _bmp280_calib.dig_P9 = readS16_LE(BMP280_REGISTER_DIG_P9);

  _bmp280_calib.dig_H1 = read8(BMP280_REGISTER_DIG_H1);
  _bmp280_calib.dig_H2 = readS16_LE(BMP280_REGISTER_DIG_H2);
  _bmp280_calib.dig_H3 = read8(BMP280_REGISTER_DIG_H3);
  _bmp280_calib.dig_H4 = (read8(BMP280_REGISTER_DIG_H4) << 4) | (read8(BMP280_REGISTER_DIG_H4+1) & 0xF);
  _bmp280_calib.dig_H5 = (read8(BMP280_REGISTER_DIG_H5+1) << 4) | (read8(BMP280_REGISTER_DIG_H5) >> 4);
  _bmp280_calib.dig_H6 = (int8_t)read8(BMP280_REGISTER_DIG_H6);

}
```

Table 17: Compensation parameter storage, naming and data type

Register Address LSB / MSB	Register content	Data type
0x88 / 0x89	dig_T1	unsigned short
0x8A / 0x8B	dig_T2	signed short
0x8C / 0x8D	dig_T3	signed short
0x8E / 0x8F	dig_P1	unsigned short
0x90 / 0x91	dig_P2	signed short
0x92 / 0x93	dig_P3	signed short
0x94 / 0x95	dig_P4	signed short
0x96 / 0x97	dig_P5	signed short
0x98 / 0x99	dig_P6	signed short
0x9A / 0x9B	dig_P7	signed short
0x9C / 0x9D	dig_P8	signed short
0x9E / 0x9F	dig_P9	signed short
0xA0 / 0xA1	reserved	reserved

1. Read8함수로 0xD0를 읽어 리셋값이 0x58이 나와야 한다.
2. Bmp280_readCoefficients함수를 보면 little endian방식으로 2bytes 씩 읽어 온다.

3. 2bytes 씩 읽어오므로 왼쪽과 같이 레지스터를 맵핑한다.

3. CODE 2

```c
void BMP280_Init(void){
    if (read8(BMP280_REGISTER_CHIPID) != 0x58){
        disp_set("Wrong chipId");
        while(1);
    }
    bmp280_readCoefficients();
    writeByte(BMP280_ADDRESS, BMP280_REGISTER_CONTROL, 0x3F);  // 01 1111
}
```

1. wirteByte를 이용하여 컨트롤 레지스터에 0x3F를 입력한다. 데이터시트를 참고하면 0x3F가 무인지를 파악하고 본인이 사용하고 싶은 용도에 맞게 변형하여 값을 넣으면 된다.

3. CODE 3

```c
float bmp280_readTemperature(void)
{
  int32_t adc_T = read24(BMP280_REGISTER_TEMPDATA);

  int32_t var1, var2;

  adc_T >>= 4;

  var1  = ((((adc_T>>3) - ((int32_t)_bmp280_calib.dig_T1 <<1))) *
           ((int32_t)_bmp280_calib.dig_T2)) >> 11;

  var2  = (((((adc_T>>4) - ((int32_t)_bmp280_calib.dig_T1)) *
             ((adc_T>>4) - ((int32_t)_bmp280_calib.dig_T1))) >> 12) *
           ((int32_t)_bmp280_calib.dig_T3)) >> 14;

  t_fine = var1 + var2;

  float T  = (t_fine * 5 + 128) >> 8;

  return T/100;
}
```

1. 온도를 읽어오는 함수로 간단히 설명하자면 3bytes를 읽어온다.

2. 우리가 사용할 bit는 20bits이기 때문에 4bit는 옮겨버리고 var1과 var2에 bit를 나누어 셋팅한다.

3. t_fine 변수에 var1과 var2를 넣어 온도 변환하는 수식을 넣고 값을 출력한다.

2. CODE

```
float bmp280_readPressure(void)
{
    int64_t var1, var2, p;

    // Must be done first to get the t_fine variable set up
    bmp280_readTemperature();

    int32_t adc_P = read24(BMP280_REGISTER_PRESSUREDATA);
    adc_P >>= 4;

    var1 = ((int64_t)t_fine) - 128000;
    var2 = var1 * var1 * (int64_t)_bmp280_calib.dig_P6;
    var2 = var2 + ((var1*(int64_t)_bmp280_calib.dig_P5)<<17);
    var2 = var2 + (((int64_t)_bmp280_calib.dig_P4)<<35);
    var1 = ((var1 * var1 * (int64_t)_bmp280_calib.dig_P3)>>8) +
      ((var1 * (int64_t)_bmp280_calib.dig_P2)<<12);
    var1 = (((((int64_t)1)<<47)+var1))*((int64_t)_bmp280_calib.dig_P1)>>33;

    if (var1 == 0) {
      return 0;  // avoid exception caused by division by zero
    }
    p = 1048576 - adc_P;
    p = (((p<<31) - var2)*3125) / var1;
    var1 = (((int64_t)_bmp280_calib.dig_P9) * (p>>13) * (p>>13)) >> 25;
    var2 = (((int64_t)_bmp280_calib.dig_P8) * p) >> 19;

    p = ((p + var1 + var2) >> 8) + (((int64_t)_bmp280_calib.dig_P7)<<4);
    return (float)p/256;
}
```

```
float bmp280_readAltitude(float seaLevel)
{
  // Equation taken from BMP180 datasheet (page 16):
  //  http://www.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf

  // Note that using the equation from wikipedia can give bad results
  // at high altitude.  See this thread for more information:
  //  http://forums.adafruit.com/viewtopic.php?f=22&t=58064

  float atmospheric = bmp280_readPressure() / 100.0F;

  return 44330.0 * (1.0 - pow(atmospheric / seaLevel, 0.1903));
}
```

1. readPressure함수도 위의 온도함수와 마찬가지이다.
2. readAltitude 함수는 해수면의 높이를 파라미터로 받아와 압력 값과 해수면값을 이용하여 고도를 구한다.

3. SIMULATION

```
altitude = 25.297636      pressure = 101021.937500        temperature = 31.520000
altitude = 25.333424      pressure = 101021.468750        temperature = 31.500000
altitude = 25.333424      pressure = 101021.468750        temperature = 31.500000
altitude = 25.103075      pressure = 101024.226563        temperature = 31.500000
altitude = 25.208923      pressure = 101022.968750        temperature = 31.480000
altitude = 25.277473      pressure = 101022.171875        temperature = 31.469999
altitude = 25.344009      pressure = 101021.375000        temperature = 31.469999
altitude = 25.410543      pressure = 101020.578125        temperature = 31.459999
altitude = 25.246727      pressure = 101022.507813        temperature = 31.459999
altitude = 25.313766      pressure = 101021.710938        temperature = 31.450001
altitude = 25.285540      pressure = 101022.078125        temperature = 31.440001
altitude = 25.258320      pressure = 101022.406250        temperature = 31.420000
altitude = 25.258320      pressure = 101022.406250        temperature = 31.420000
altitude = 25.324350      pressure = 101021.609375        temperature = 31.420000
altitude = 25.687271      pressure = 101017.234375        temperature = 31.410000
altitude = 25.593515      pressure = 101018.390625        temperature = 31.400000
altitude = 25.132814      pressure = 101023.906250        temperature = 31.400000
altitude = 25.496233      pressure = 101019.523438        temperature = 31.389999
altitude = 25.401470      pressure = 101020.687500        temperature = 31.379999
altitude = 25.534540      pressure = 101019.085938        temperature = 31.370001
altitude = 25.534540      pressure = 101019.085938        temperature = 31.370001
altitude = 25.437761      pressure = 101020.218750        temperature = 31.360001
altitude = 25.706427      pressure = 101017.000000        temperature = 31.340000
altitude = 25.410040      pressure = 101020.585938        temperature = 31.340000
altitude = 25.772964      pressure = 101016.203125        temperature = 31.330000
altitude = 25.284531      pressure = 101022.078125        temperature = 31.320000
altitude = 25.351570      pressure = 101024.007813        temperature = 31.320000
altitude = 25.418608      pressure = 101020.453125        temperature = 31.309999
altitude = 25.390381      pressure = 101020.820313        temperature = 31.299999
altitude = 25.523451      pressure = 101019.195313        temperature = 31.290001
altitude = 25.226566      pressure = 101022.750000        temperature = 31.290001
altitude = 25.293100      pressure = 101021.953125        temperature = 31.290001
altitude = 25.265377      pressure = 101022.312500        temperature = 31.270000
altitude = 25.398951      pressure = 101020.687500        temperature = 31.260000
altitude = 25.168600      pressure = 101023.445313        temperature = 31.260000
altitude = 25.398951      pressure = 101020.687500        temperature = 31.260000
altitude = 25.140375      pressure = 101023.812500        temperature = 31.250000
altitude = 25.209932      pressure = 101022.945313        temperature = 31.240000
altitude = 25.209932      pressure = 101022.945313        temperature = 31.240000
altitude = 25.209932      pressure = 101022.945313        temperature = 31.240000
altitude = 25.346529      pressure = 101021.351563        temperature = 31.230000
```

1. 데이터시트를 보면 압력값은 300-1100hPa까지 측정이 가능하다.

2. 압력 값이 101021.375000으로 정상적으로 나온다. 하지만, 우리가 보기에는 너무 큰 숫자이므로, 100을 나누어 스케일링을 해서 출력을 하면 된다.

3. 고도와 온도는 정상적으로 나오는 것을 확인할 수 있다.