

TI DSP, MCU, Xilinx Zynq FPGA

프로그래밍 전문가 과정

Control MPU9250 with I2C

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 안상재
sangjae2015@naver.com

1. CubeMX 설정

STM32CubeMX MPU9250_TEST.ioc: STM32F407VGTX

File Project Pinout Window Help

Keep Current Signals

Pinout Clock Configuration Configuration Power Consumption

ADC2

ADC3

CAN1

CAN2

CRC

DAC

DCMI

ETH

FSMC

I2C1

I2C2

I2C1 enable

UART4

UART5

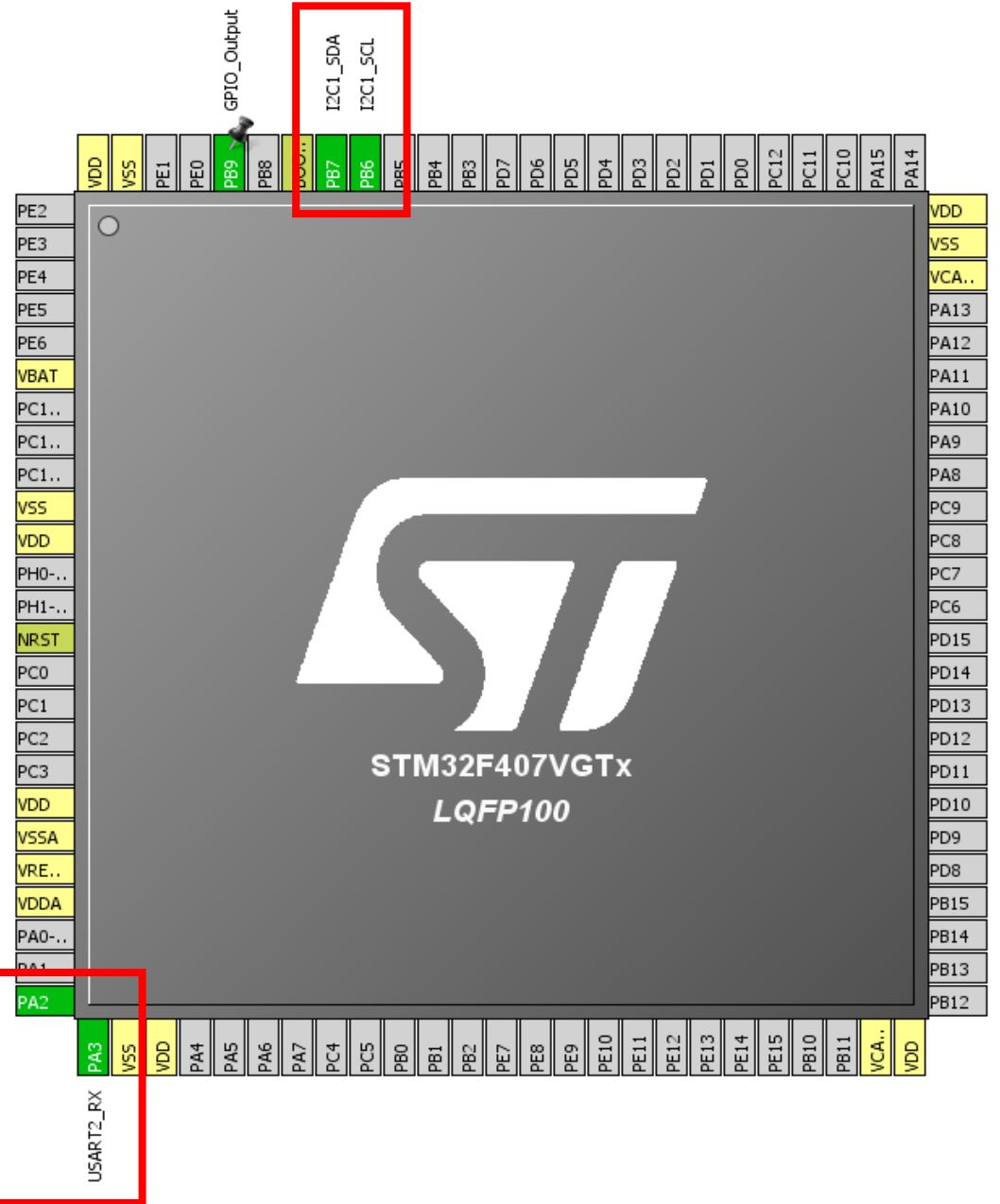
USART1

USART2

USART3

IISART6

USART2 enable
=> 센서 데이터 값 출력



MCU를 Master로 설정

I2C1 Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Configure the below parameters :

Search :

Master Features	
I2C Speed Mode	Fast Mode
I2C Clock Speed (Hz)	400000
Fast Mode Duty Cycle	Duty cycle Tlow/Thigh = 2
Slave Features	
Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

2. 소스 코드

WHO_AM_I 레지스터를 읽으면 0x71이 나와야 함!

```
whoami = readByte(MPU9250_ADDRESS, WHO_AM_I_MPU9250);  
sprintf(str, "str = 0x%x\r\n", whoami);  
HAL_UART_Transmit(&huart2, str, 30, 10);
```

WHO_AM_I에서 0x71이 나오면 정상

```
if(whoami == 0x71)  
{  
    resetMPU9250();  
    MPU9250의 모든 레지스터를 디폴트 값으로 셋팅  
    레지스터 calibrate  
    calibrateMPU9250(gyroBias, accelBias); // Calibrate gyro and accelerometers, load biases in bias registers  
  
    sprintf(str, "x gyro bias = %f\n\r", gyroBias[0]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
    sprintf(str, "y gyro bias = %f\n\r", gyroBias[1]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
    sprintf(str, "z gyro bias = %f\n\r", gyroBias[2]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
  
    HAL_UART_Transmit(&huart2, "\r\n", 2, 10);  
    sprintf(str, "x accel bias = %f\n\r", accelBias[0]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
    sprintf(str, "y accel bias = %f\n\r", accelBias[1]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
    sprintf(str, "z accel bias = %f\n\r", accelBias[2]);  
    HAL_UART_Transmit(&huart2, str, 50, 10);  
  
    HAL_UART_Transmit(&huart2, "\r\n\n", 5, 10);
```

MPU9250 레지스터 셋팅

```
initMPU9250();
```

```
initAK8963(magCalibration);
```

지자기 레지스터 셋팅

```
if(Mscale == 0)
{
    sprintf(str, "Magnetometer resolution = 14 bits\n\r");
    HAL_UART_Transmit(&huart2, str, 50, 10);
}
if(Mscale == 1)
{
    sprintf(str, "Magnetometer resolution = 16 bits\n\r");
    HAL_UART_Transmit(&huart2, str, 50, 10);
}
if(Mmode == 2)
{
    sprintf(str, "Magnetometer ODR = 8 Hz\n\r");
    HAL_UART_Transmit(&huart2, str, 50, 10);
}
if(Mmode == 6)
{
    sprintf(str, "Magnetometer ODR = 100 Hz\n\r");
    HAL_UART_Transmit(&huart2, str, 50, 10);
}
```

민감도 설정

```
getAres(); // get accelerometer sensitivity
getGres(); // get gyro sensitivity
getMres(); // get magnetometer sensitivity

sprintf(str, "Accelerometer sensitivity is %f LSB/g \r\n", 1.0f/aRes);
HAL_UART_Transmit(&huart2, str, 50, 10);

sprintf(str, "Gyroscope sensitivity is %f LSB/deg/s \n\r", 1.0f/gRes);
HAL_UART_Transmit(&huart2, str, 50, 10);

sprintf(str, "Magnetometer sensitivity is %f LSB/G \n\r", 1.0f/mRes);
HAL_UART_Transmit(&huart2, str, 50, 10);

magbias[0] = +470.; // User environmental x-axis correction in milliGauss, should be automatically calculated
magbias[1] = +120.; // User environmental x-axis correction in milliGauss
magbias[2] = +125.; // User environmental x-axis correction in milliGauss
```

지자기 값 보정

raw 데이터 read

```
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
    if(readByte(MPU9250_ADDRESS, INT_STATUS) & 0x01) // On interrupt, check if data ready interrupt
    {
        readAccelData(accelCount); // Read the x/y/z adc values
        // Now we'll calculate the acceleration value into actual g's
        ax = (float)accelCount[0]*aRes - accelBias[0]; // get actual g value, this depends on scale being set
        ay = (float)accelCount[1]*aRes - accelBias[1];
        az = (float)accelCount[2]*aRes - accelBias[2];

        readGyroData(gyroCount); // Read the x/y/z adc values
        // Calculate the gyro value into actual degrees per second
        gx = (float)gyroCount[0]*gRes - gyroBias[0]; // get actual gyro value, this depends on scale being set
        gy = (float)gyroCount[1]*gRes - gyroBias[1];
        gz = (float)gyroCount[2]*gRes - gyroBias[2];

        readMagData(magCount); // Read the x/y/z adc values
        // Calculate the magnetometer values in milliGauss
        // Include factory calibration per data sheet and user environmental corrections
        mx = (float)magCount[0]*mRes*magCalibration[0] - magbias[0]; // get actual magnetometer value, this depends on scale being set
        my = (float)magCount[1]*mRes*magCalibration[1] - magbias[1];
        mz = (float)magCount[2]*mRes*magCalibration[2] - magbias[2];
    }

    MadgwickQuaternionUpdate(ax, ay, az, gx*PI/180.0f, gy*PI/180.0f, gz*PI/180.0f, my, mx, mz);
}
```

raw 데이터 준비 완료

데이터 정규화 과정

온도 데이터 read

```
tempCount = readTempData(); // Read the adc values
temperature = ((float) tempCount) / 333.87f + 21.0f; // Temperature in degrees Centigrade
```

```
sprintf(str1, "ax = %f ay = %f az = %f mg", 1000*ax, 1000*ay, 1000*az);
HAL_UART_Transmit(&huart2, str1, sizeof(str1), 10);
HAL_UART_Transmit(&huart2, "\r\n", 2, 10);
```

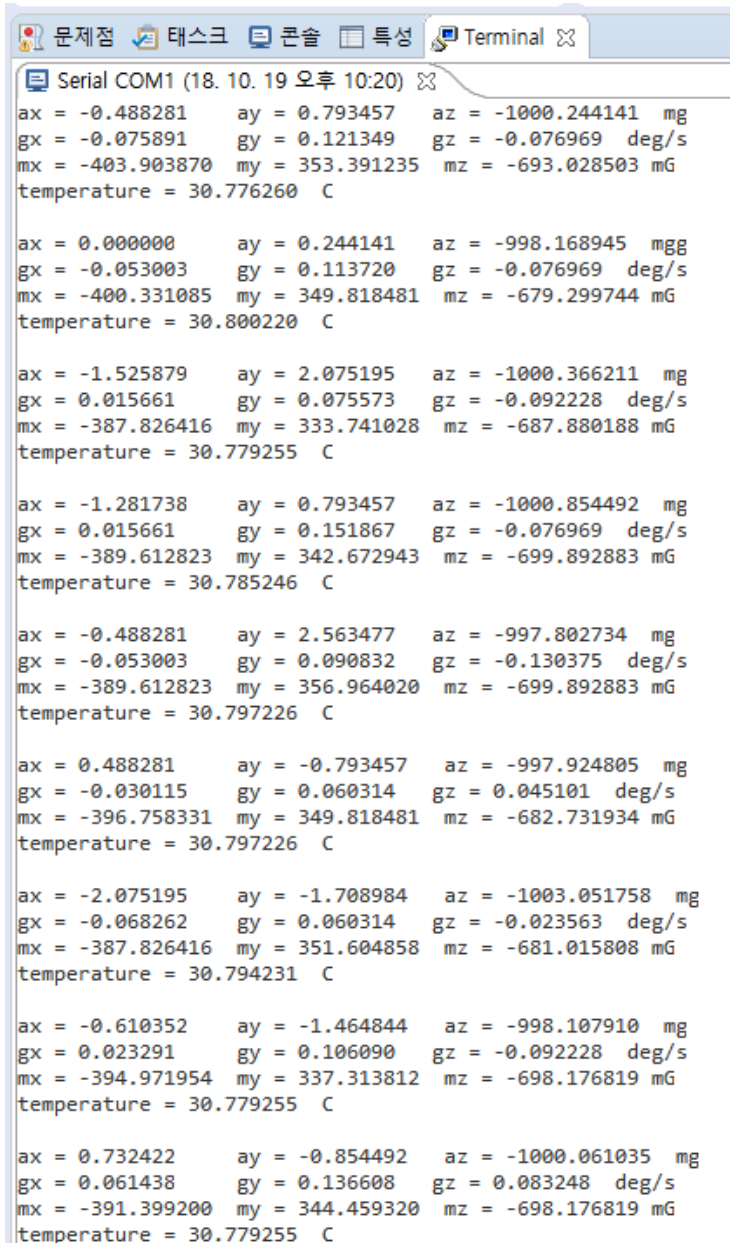
```
sprintf(str2, "gx = %f gy = %f gz = %f deg/s", gx, gy, gz);
HAL_UART_Transmit(&huart2, str2, sizeof(str2), 10);
HAL_UART_Transmit(&huart2, "\r\n", 2, 10);
```

```
sprintf(str3, "mx = %f my = %f mz = %f mG", mx, my, mz);
HAL_UART_Transmit(&huart2, str3, sizeof(str3), 10);
HAL_UART_Transmit(&huart2, "\r\n", 2, 10);
```

```
sprintf(str4, "temperature = %f C\r\n\r\n", temperature);
HAL_UART_Transmit(&huart2, str4, sizeof(str4), 10);
```

```
HAL_Delay(1000);
```

3. 결과 화면



The screenshot shows a terminal window with a title bar containing icons for '문제점' (Problem), '태스크' (Task), '콘솔' (Console), '특성' (Properties), and 'Terminal'. The terminal title is 'Serial COM1 (18. 10. 19 오후 10:20)'. The output displays sensor data in a tabular format, with each row representing a single data point. The data includes acceleration (ax, ay, az) in mg, gyration (gx, gy, gz) in deg/s, magnetic field (mx, my, mz) in mG, and temperature in degrees Celsius. The data is printed in a repeating cycle of four lines per data point.

```
ax = -0.488281    ay = 0.793457    az = -1000.244141 mg
gx = -0.075891    gy = 0.121349    gz = -0.076969 deg/s
mx = -403.903870  my = 353.391235  mz = -693.028503 mG
temperature = 30.776260 C

ax = 0.000000     ay = 0.244141    az = -998.168945 mgg
gx = -0.053003    gy = 0.113720    gz = -0.076969 deg/s
mx = -400.331085  my = 349.818481  mz = -679.299744 mG
temperature = 30.800220 C

ax = -1.525879    ay = 2.075195    az = -1000.366211 mg
gx = 0.015661     gy = 0.075573    gz = -0.092228 deg/s
mx = -387.826416  my = 333.741028  mz = -687.880188 mG
temperature = 30.779255 C

ax = -1.281738    ay = 0.793457    az = -1000.854492 mg
gx = 0.015661     gy = 0.151867    gz = -0.076969 deg/s
mx = -389.612823  my = 342.672943  mz = -699.892883 mG
temperature = 30.785246 C

ax = -0.488281    ay = 2.563477    az = -997.802734 mg
gx = -0.053003    gy = 0.090832    gz = -0.130375 deg/s
mx = -389.612823  my = 356.964020  mz = -699.892883 mG
temperature = 30.797226 C

ax = 0.488281     ay = -0.793457    az = -997.924805 mg
gx = -0.030115    gy = 0.060314    gz = 0.045101 deg/s
mx = -396.758331  my = 349.818481  mz = -682.731934 mG
temperature = 30.797226 C

ax = -2.075195    ay = -1.708984    az = -1003.051758 mg
gx = -0.068262    gy = 0.060314    gz = -0.023563 deg/s
mx = -387.826416  my = 351.604858  mz = -681.015808 mG
temperature = 30.794231 C

ax = -0.610352    ay = -1.464844    az = -998.107910 mg
gx = 0.023291     gy = 0.106090    gz = -0.092228 deg/s
mx = -394.971954  my = 337.313812  mz = -698.176819 mG
temperature = 30.779255 C

ax = 0.732422     ay = -0.854492    az = -1000.061035 mg
gx = 0.061438     gy = 0.136608    gz = 0.083248 deg/s
mx = -391.399200  my = 344.459320  mz = -698.176819 mG
temperature = 30.779255 C
```