2020/04/10 Discussion

0616014 楊政道

April 17, 2020

1 Discussion 1

1.1 Based on the requirements, set correct value in the sample code

- 8 samples per measurement
- Data Output Rate = 15Hz
- Gain=1090(LSb/Gauss)
- Convert LSB to Gauss (by using self.scale)

```
self.scale = ?? # convert bit value(LSB) to gauss. DigitalResolution

# Configuration Register A
self.write_byte(HMC5883L_CRA, 0b????????)

# Configuration Register B
self.write_byte(HMC5883L_CRB, 0b????????)

# Mode Register
self.write_byte(HMC5883L_MR, 0b????????)
```

1.1.1 scale

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)

Gain = 1090(LSb/Gauss), so the value of the scale is 0.92.

1.1.2 Register A

CRA7	CRA6	CRA5	CRA4	CRA3	CRA2	CRA1	CRA0
(0)	MA1(0)	MA0(0)	DO2 (1)	DO1 (0)	DO0 (0)	MS1 (0)	MS0 (0)

Table 3: Configuration Register A

Location	Name	Description
CRA7	CRA7	Bit CRA7 is reserved for future function. Set to 0 when configuring CRA.
CRA6 to CRA5	MA1 to MA0	Select number of samples averaged (1 to 8) per measurement output. 00 = 1(Default); 01 = 2; 10 = 4; 11 = 8

8 samples

DO2	DO1	DO0	Typical Data Output Rate (Hz)
0	0	0	0.75
0	0	1	1.5
0	1	0	3
0	1	1	7.5
1	0	0	15 (Default)
1	0	1	30
1	1	0	75
1	1	1	Reserved

15 Hz

Table 5: Data Output Rates

8 samples per measurement, Data Output Rate 15Hz, so the value of Register A is 0b00010000.

1.1.3 Register B

CRB7	CRB6	CRB5	CRB4	CRB3	CRB2	CRB1	CRB0
GN2 (0)	GN1 (0)	GN0 (1)	(0)	(0)	(0)	(0)	(0)

Table 7: Configuration B Register

GN2	GN1	GN0	Recommended Sensor Field Range	Gain (LSb/ Gauss)	Digital Resolution (mG/LSb)	Output Range
0	0	0	± 0.88 Ga	1370	0.73	0xF800–0x07FF (-2048–2047)
0	0	1	± 1.3 Ga	1090 (default)	0.92	0xF800–0x07FF (-2048–2047)

Gain = 1090(LSb/Gauss), set the value of Register B to 0b00100000.

1.1.4 Mode Register

```
# Mode Register, write value: 0000 0000
self.write_byte(HMC5883L_MR, 0b00000000)
# MR1-MR0 = 00 (Mode Select Bits) -> Continuous-Measurement Mode.
```

MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
HS(0)	(0)	(0)	(0)	(0)	(0)	MD1 (0)	MD0 (1)

Table 10: Mode Register

MD1	MD0	Operating Mode
0	0	Continuous-Measurement Mode. In continuous-measurement mode, the device continuously performs measurements and places the result in the data register. RDY goes high when new data is placed in all three registers. After a power-on or a write to the mode or configuration register, the first measurement set is available from all three data output registers after a period of $2/f_{DO}$ and subsequent measurements are available at a frequency of f_{DO}, where f_{DO} is the frequency of data output.

Set to thhe continuous mode, so the value of the Mode Register is 0b00000000.

1.2 Continuously measurement(infinite loop)

In order to make the sensor measure conitnuously, we can put the getValue function of the sensor into a while loop with a time sleep function.

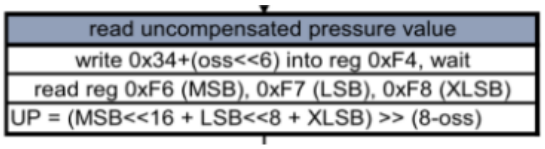
```
while True:
    value = getValue()
    print (value)
    time.sleep(0.1)
```

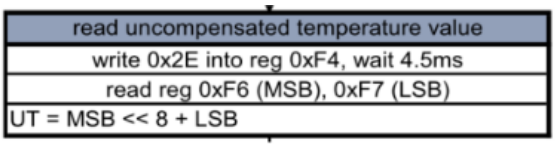
1.3 Calibrate your sensor (see next page)

In theory, the maximum and minimum value of three axises should be same in absolute value. Therefore, we can spin the sensor first and get the offset value to calibrate it.

2 Discussion 2

2.1 Based on the datasheet, set correct value in the sample code






```
def getPress(self) :
    # print ("Calculating temperature...")
    self.write_byte(0xF4, 0x??)
    time.sleep(0.005)
```

```
# read uncompensated temperature value
def getTempC(self) :
    # print ("Calculating temperature...")
    self.write_byte(0xF4, 0x??)
    time.sleep(0.005)
```

2.1.1 getPress

According to the datasheet, we need to set the value into 0x34 + (self.oversampling « 6)

2.1.2 getTempC

According to the datasheet, we need to set the value into 0x2E

2.2 Continuously measurement(infinite loop)

In order to make the sensor measure conitnuously, we can put the getValue function of the sensor into a while loop with a time sleep function.

```
while True:
    value = getValue()
    print (value)
    time.sleep(0.1)
```