The default I²C address of the BNO055 device is 0101001b (0x29). NO…

* **ADR**: Set this pin high to change the default I2C address for the BNO055 if you need to connect two ICs on the same I2C bus. The default address is 0x28. If this pin is connected to 3V, the address will be 0x29 (FROM ADAFRUIT WEBSITE)

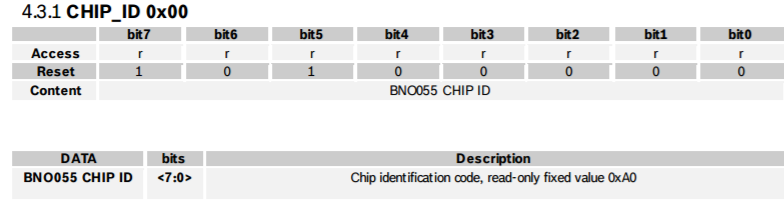
The I²C write phase addresses the slave and **sends the register address to be read**. After slave acknowledges the transmission, the master generates again a start condition and sends the slave address together with a read bit (RW = 1). Then the master releases the bus and waits for the data bytes to be read out from slave. After each data byte the master has to generate an acknowledge bit (ACK = 0) to enable further data transfer. A NACKM (ACK = 1) from the master stops the data being transferred from the slave. The slave releases the bus so that the master can generate a STOP condition and terminate the transmission. The register address is automatically incremented and, therefore, more than one byte can be sequentially read out. Once a new data read transmission starts, the start address will be set to the register address specified in the latest I²C write command.



Adafruit steps all write commands have delays of like 20 ms between

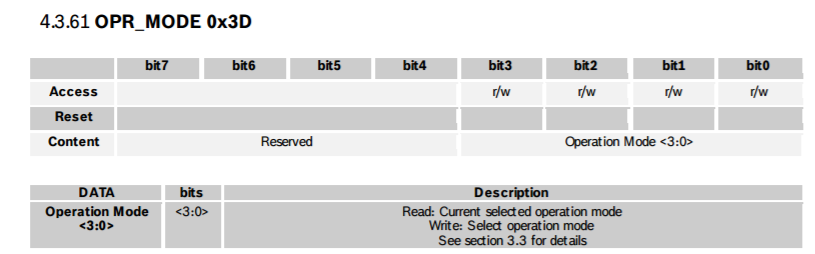
Setup

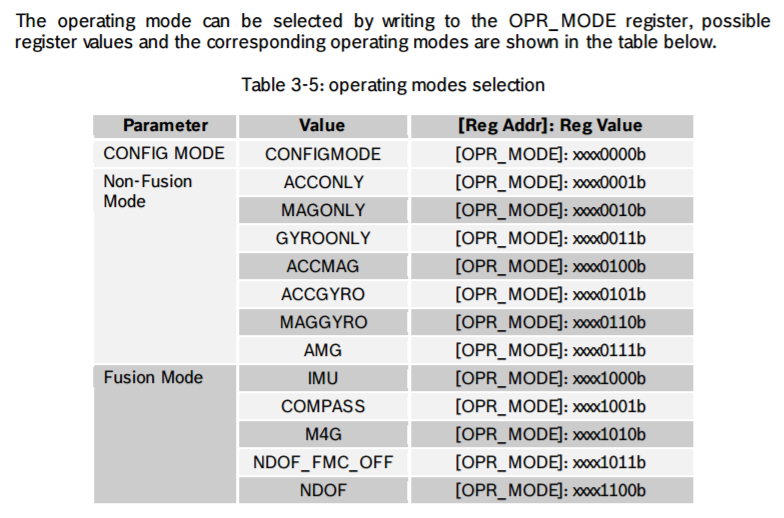
* Begin wire
* Read chip address (0x00)



* If it’s not 0xA0, wait 1second, then try again, and then bail
* setMode🡪 write (0X3D, 0X00) [ CONFIG MODE ]

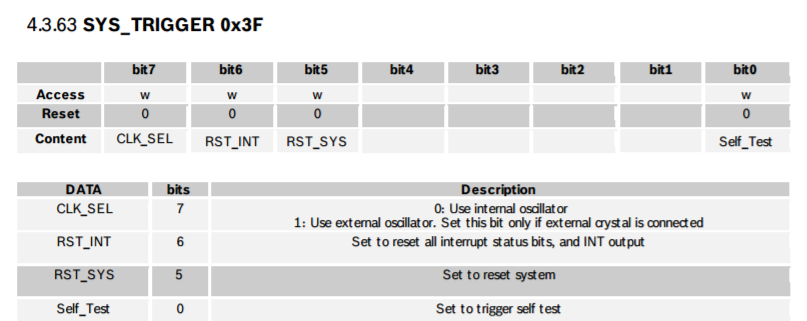






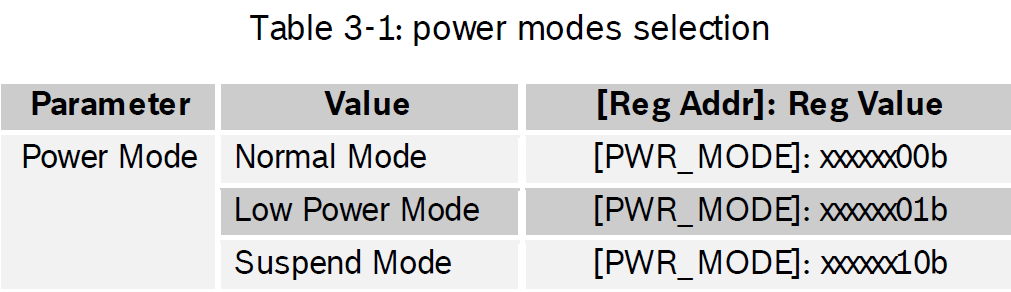
* reset… write(0X3F, 0x20), 00100000





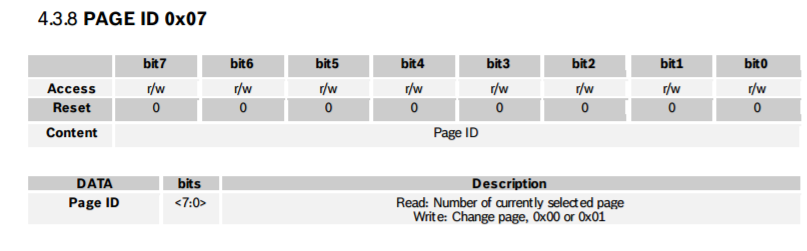
* wait until we read the chip address and it’s 0xA0 again
* set to normal power mode: write(0X3E, 0X00)





* write (0X07, 0)





* write( 0X3F, 0)
* setmode -> write (0X3D, 0X0C) [ FUSION MODE, NDOF, 1100b ]
* return true!

bno.setExtCrystalUse(true)

* set mode configure write (0X3D, 0X00) [ CONFIG MODE ]
* write8(0X07, 0) (Page ID register set to zero)
* write8(0X3F, 0x80) (An External clock can be selected by setting bit CLK\_SEL in the SYSTEM\_TRIGGER register.) 0x80 = 10000000
* set mode write (0X3D, 0X0C) [ FUSION MODE, NDOF, 1100b ]

loop;

* uint8\_t buffer[6];
* memset (buffer, 0, 6);
* int16\_t x, y, z;
* x = y = z = 0;

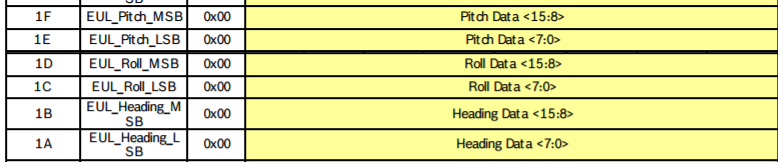
buffer, 6

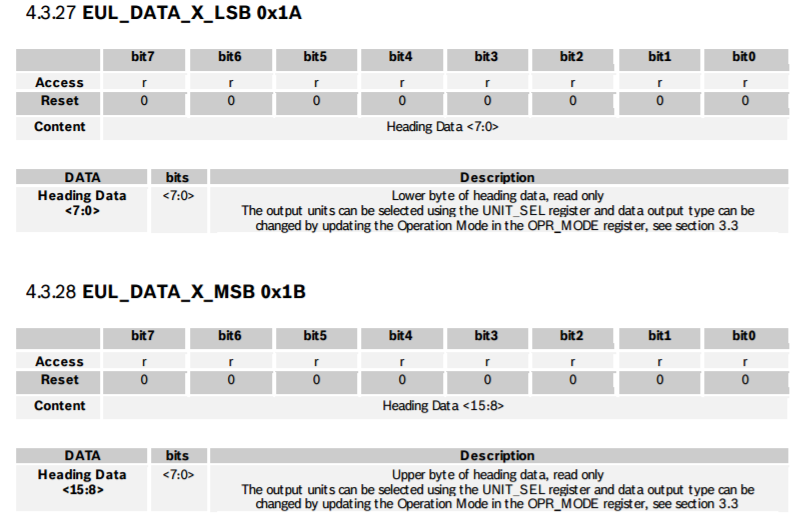
readLen(adafruit\_bno055\_reg\_t reg, byte \* buffer, uint8\_t len)

Wire.beginTransmission(\_address);

#if ARDUINO >= 100

Wire.write((uint8\_t)reg); 🡪 I THINK 0X1A CAUSE THIS IS FIRST ADDRESS OF EULER DATA





BASICALLY SAME FOR OTHER 4 BYTES

#else

Wire.send(reg);

#endif

Wire.endTransmission();

Wire.requestFrom(\_address, (byte)6);

for (uint8\_t i = 0; i < 6; i++)n

{

#if ARDUINO >= 100

buffer[i] = Wire.read();

#else

buffer[i] = Wire.receive();

#endif

}

/\* ToDo: Check for errors! \*/

return true;

x = ((int16\_t)buffer[0]) | (((int16\_t)buffer[1]) << 8);

y = ((int16\_t)buffer[2]) | (((int16\_t)buffer[3]) << 8);

z = ((int16\_t)buffer[4]) | (((int16\_t)buffer[5]) << 8);

Testing:

1. Chip ID works… have to pass it a pointer address &value as the location for the info to be saved.
2. Page ID write works… (that’s a r/w memory register)

NACKACK = i2cWrite(BN0055\_ADDRESS, PAGE\_ID, 1); // Go to Page 1 of memory registers

NACKACK = i2cRead(BN0055\_ADDRESS, PAGE\_ID, 1, &value);

Serial.print("Page ID: ");

Serial.println(value);

NACKACK = i2cWrite(BN0055\_ADDRESS, PAGE\_ID, 0); // Go to Page 0 of memory registers

NACKACK = i2cRead(BN0055\_ADDRESS, PAGE\_ID, 1, &value);

Serial.print("Page ID: ");

Serial.println(value);

This returns Page ID: 1 then Page ID: 0

1. Checked OPR\_MODE…

value = -100;

NACKACK = i2cWrite(BN0055\_ADDRESS, OPR\_MODE, 0); // Set operation mode to configure

NACKACK = i2cRead(BN0055\_ADDRESS, OPR\_MODE, 1, &value);

Serial.print("OPR\_MODE: ");

Serial.println(value);

value = -100;

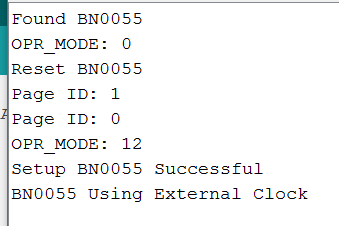
NACKACK = i2cWrite(BN0055\_ADDRESS, POWER\_MODE, 0); // Set power mode to normal

NACKACK = i2cWrite(BN0055\_ADDRESS, PAGE\_ID, 1); // Go to Page 1 of memory registers

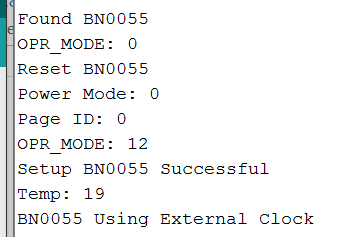
NACKACK = i2cRead(BN0055\_ADDRESS, PAGE\_ID, 1, &value);

Serial.print("Page ID: ");

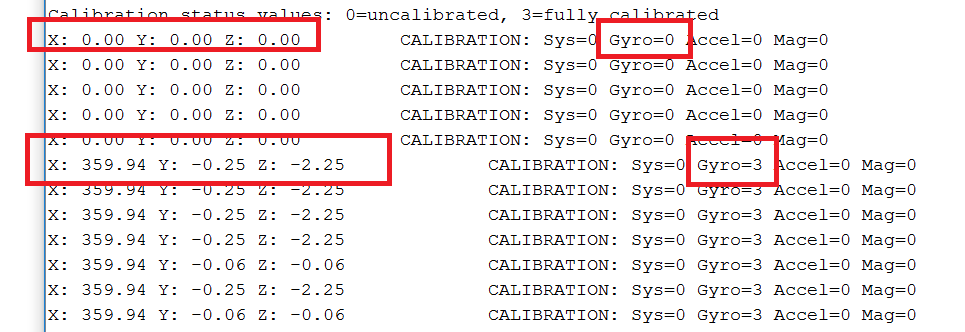
Serial.println(value);



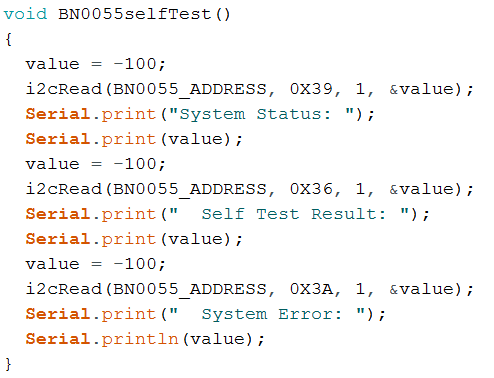
1. Checked power mode, works
2. Checked Temperature, works



Noticed that gyro needs to be calibrated? Before we get values from it.



1. Self Test gives good results:





/\* System Status (see section 4.3.58)

---------------------------------

0 = Idle

/\* Self Test Results (see section )

--------------------------------

1 = test passed, 0 = test failed

Bit 0 = Accelerometer self test

Bit 1 = Magnetometer self test

Bit 2 = Gyroscope self test

Bit 3 = MCU self test

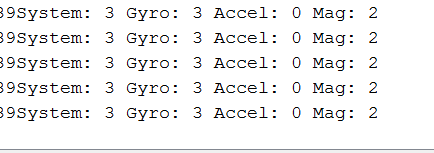
0x0F = all good! \*/ 🡪 F = dec(15), see above means all good.

/\* System Error (see section 4.3.59)

---------------------------------

0 = No error

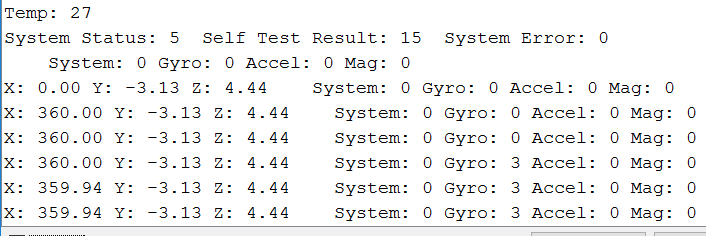
1. When I turn off the use of the external clock, then I get calibration of the gyro (3) and accelerometer (1), and then I start to get some values for sensor.



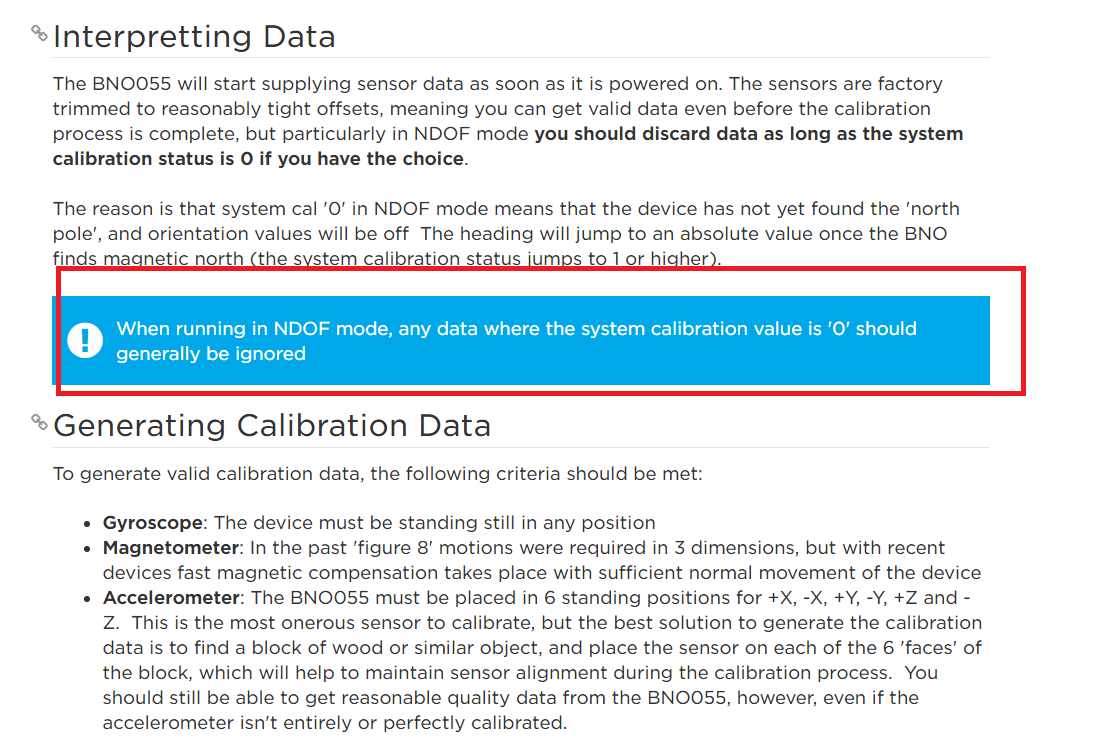
Getting values but only seems like for first part of register… check other registers manually, one at a time

Also, need to divide by 16.0… but type is imu (for vector) and may require some math? Look in util/imumaths.h

OK, got it working. I was only collecting 1 byte when I read Euler angles which is why it wasn’t working. Now I am getting values that make sense. Note still not using external crystal (does the board have an external crystal?) Also note that now the system status shows 5, which means Sensor Fusion Algorithm is running (that’s good!).



NOTENOTENOTE!!!!



This isn’t too big a problem. If you move it around a bit, soon the system cal goes to 3. So could just move around at beginning until that criteria is met. Have LED that goes green or something?