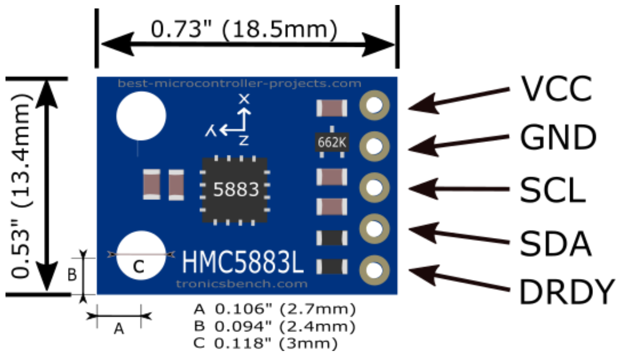
Изменено 09/03/2023 10:36:18 Создано 09/03/2023 10:36:18 https://www.best-microcontroller-projects.com/hmc5883l.html

# How the HMC5883L 3-axis digital magentomter works

The HMC5883L chip is a three axis magnetic sensor or digital compass, made by Honeywell. I bought a breakout board from ebay, and was a bit confused as the markings on the chip did not match this device. It's time for some investigation!

Warning:Three axis Magnetometers are sensitive to their orientation. When not level, the output bearing will be incorrect. The solution is to compensate for the orientation using anaccelerometer.



There are five major sections to this page:

HMC5883L and QST specification and capabilities.
How to use and calibrate a magnetometer.
How to use a Magnetometer as a compass, with example code.
The difference between breakout boards.
Differences between the HMC and QST chip versions.
In fact there are two chip versions: HMC (made by Honeywell) and the QMC (Made by QST Corporation).

Note: HMC and QMC versions use different I2C addresses.

Note: HMC and QMC versions use different register addresses.

Note: HMC and QMC chips output essentially the same data.

This page discusses the differences between the HMC and QMC versions and describes whether you can use the QMC version as an HMC5883L replacement. It also shows you how to calibrate the magnetometer for basic operation as a compass.

## But which chip have do you have?

It is important to find out which chip you have, as each chip has different internal registers. You will think the chip has failed if you use the wrong library!

Honeywell is stopping production of the HMC588L device and has licensed the design to QST. You can find out which one you have using the table below:

| Manufacturer | Chip Markings | Device Name |
|---|---|---|
| Honeywell | L883 | HMC5883L |
| QST Corporation | 5883 | QMC5883L |

If the marking on your chip is '5883' then you have the QMC5883L. if it is L883 then you have the HMC5883L. You can compare the features of these devices in detail further down the page here.

## Measuring magnetic fields

These magnetometer chips can measure field strengths from 2mGauss to 8Gauss. Since the average field strength of the Earth's magnetic field is:

> Between 30uT (0.3G) and 60uT (0.6G) [source]...

These chips easily measure the Earth's magnetic field strength, as well as magnetic fields from higher strength magnetic materials.

### QMC5883L and HMC5883L Specification

| Parameter | HMC5883L | QMC5883L |
|---|---|---|
| Voltage Supply (Vs) | 2V16 ~ 3V6 | 2V16 ~ 3V6 |
| Digital Supply (VDDIO) (max) | 1.71V ~ 3V7 | 1.65V ~ 3V6 |
| Abs. Max VDD/VDDIO | -0.3V ~ 4.8V | -0.3V ~ 5.4V |
| Interface | I2C | I2C |
| I2C Address (R,W) [RW] | 0x3D, 0x3C | 0x0D [R/W] |
| I2C rates (kHz) | 100, 400, 3400 | 100, 400 |
| Resolution (ADC) | 12 bits | 16 bits |
| Max Gauss (survival) | Not specified. | 50000G |
| Gauss Resolution | ±2mG ~ ±8G | ±2mG ~ ±8G |
| Acquisition time | 6ms | 6ms |
| Active current (7Hz,10Hz) | 100uA | 75uA |
| Active current | Not specified. | 75uA ~ 850uA[2] |
| Peak Active current | Not Specified[3] | 2.6mA |
| Standby mode (leakage) | 2uA | 3uA |
| Operating temperature | -30°C ~ 85°C | -40°C ~ 85°C |

[1] VDDIO is the digital interface I/O voltage level. The lower value (cf Vs) allows low voltage devices to operate (such as those using 1.8V).
[2] The QMC5883L has many more oversampling rates (OSR) and output rates (ODR) [higher OSR and ODR, cause the chip to consume more current].
[3] The QMC5883L uses automatic temperature compensation but will use the same degauss method as the HMC5883L. In the HMC5883L chip a 10mA driver is used for automatic degaussing using strap drivers (this self test process also takes 6ms which is the same acquisition time as the QMC5883L) so the peak current is probably the same as the QMC5883L.

## 5883L Datasheets

Download the QMC5883L Datasheet here. (it is more likely you have this chip).

Download the HMC5883L Datasheet here.

Using a magnetometer should be easy to use, and fundamentally it is. There are two problems with it: magnetic interference and orientation.

The problem is that you can get in a muddle converting the magnetometer x-y reading, to the screen x-y reading and comparing this to which way you are wanting to get a reading from the orientation of the device you are holding. Confusing - slightly.

The way to sort all this out is to recognise exactly what the magnetometer is reading.

Let's state the obvious:

A magnetometer measures the magnetic field strength
and has a maximum output along a specific axis when
the axis is aligned, and pointing to the Earth's North pole.

It is exactly the same as using a traditional compass where the needle points north. Once we have that sorted out we can write a program to show the output - however you must calibrate for offsets first (see below).

Note the angle returned from the atan function returns a zero reading when the North pole is aligned with the the x axis, so this influences the bearing you calculate. You can adjust the bearing by adding an offset value e.g. +90 Degrees rotates the bearing anti-clockwise (CCW- Counter ClockWise).

## Magnetic Distortion

The next thing to understand is that the reading from the magnetometer will probably not be usable, unless you are in a field with no other magnetic distortion around you (so called soft-iron and hard-iron interference).

## Hard Iron distortion

Hard-iron interference is:

A block of magnetic field producing material.
This distortion causes an offset in the magnetometer
output since it adds or subtracts a magnetic field.

For an example of hard iron interference look no further than a speaker or magnetised piece of iron, or even a battery (try moving a battery near by when looking at the output of the magnetometer). This distortion is by far the most important to be eliminated and must be removed before the magnetometer can be used as a compass.

## Soft Iron Distortion

Soft-iron interference is :

A distortion of the magnetic field caused by
ferro-magnetic material. This distortion warps the
Earth's magnetic field causing an ellipsoid magnetometer output.

For creating a compass this fact is not too important as a bearing will still be returned even for an ellipsoid output.

Soft iron distortion only warps the magnetic field - think of an non magnetised piece of iron. A compass application can cope with this and will return a bearing since soft iron distortion does not create a magnetic field just alters an existing one. However, it will cause the bearing to be changed slightly!

## QMC5883 and HMC5883L Arduino Calibration

Since the HMC and QMC versions essential provide the same output data the calibration process is the same in each case. All that is different is the library required to interface to the chip. Just select the relevant library and use the code below for calibration.

You usually perform a unit calibration to improve the accuracy of a device. However, for a magnetometer, there are many magnetic error sources from large blocks of metal, to batteries, and even the surrounding circuitry!

Calibration in this sense is to restore functionality - You simply cannot use a magnetometer without calibration!

Warning: You can't use a magnetometer without calibration.

Calibration should be done in an interference free area (or an area where conditions will always be the same). That means an area away from magnetic interference i.e. away from metallic structures and electric current.

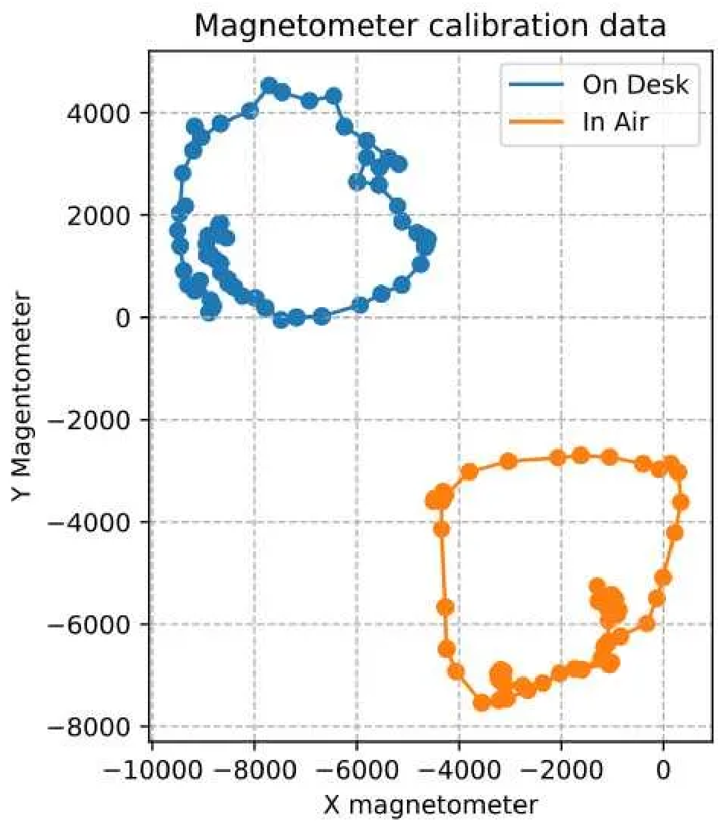
TIP: Perform calibration in a magnetic interference free area.

You should also perform a calibration when changing batteries as different batteries will have different magnetic field output. This calibration will calibrate out the errors caused by magnetic devices nearby e.g. batteries, motors and other ferro magnetic field generating parts within the circuit itself.

## Uncalibrated QMC5883 Outputs

The following diagram shows how a magnetometer is affected by surrounding magnetic fields, or iron - the only difference between the two plots was that the circuit was raised 12 inches off the desk!. The diagrams below should be centered around x,y = (0,0)! Unless the output is corrected you can't calculate a bearing at all.

Note: Calibration is essential to get meaningful data output.



These results were obtained while keeping the magnetometer level. Offset from the horizontal plane also significantly affects the results (which can be corrected by adding an accelerometer - and of course extra code).

Warning: Horizontal orientation is essential - keep it flat!

In normal use, you will use the magnetometer outside so the difference in offsets will not be as large as shown here.

## Hard Iron offset calibration

The goal of calibration is to identity the centre of the circle, and then apply these centre points as offsets to the signals from the magnetometer.

This is simply achieved by detecting maximum and minimum x and y coordinates to find the centre point (xmax-xmin)/2 and using this as the x offset in future calculations. Similarly a calculation is also done for y values.

In the code this operation is performed by the calc_offsets() function, which updates variables offx and offy. These values are then subtracted from the magnetometer reading to give a zero-centered magnetometer reading i.e. a calibrated reading.

## Magnetic Declination

You may need to adjust the magnetometer reading for declination offset. Declination offset is a value that you add to your magnetic bearing (so it applies to standard compasses as well). It defines true north since in some areas true north and magnetic north are not aligned. You can find more information and a link to a tool to find your magnetic declination here.

## Magnetic Inclination

Note: Magnetic Inclination = Vertical field angle from horizontal.

For a compass bearing the magnetic Inclination is irrelevant.

The first example uses the QMC5883L and the serial port to display data. This is useful as a simple demonstration but is inconvenient when using the device. It is far easier when using a dedicated display. So in the second example an SSD1306 displays the bearing.

## Example Serial output Compass

The following example code outputs a bearing to the serial terminal. When a button is pressed, a calibration routine executes (lasting 10 seconds) during which time you must rotate the chip through 360° while data is gathered from x and y axes.

Note the chip is sensitive to horizontal deviation, so to get good readings keep the chip level.

The program then stores the calibration data in the EEPROM.

## Hardware for Example Sketch 1

Arduino Nano (or Uno),
HMC5883L (or QST5883L) breakout board.
Push to make button (1 off).

## Hardware connections for Example Sketch1

Make connections as follows:

| Arduino Uno/ Nano pin | Label | Destination |
|---|---|---|
| A5 | SCL | 5883L SCL |
| A4 | SDA | 5883L SDA |
| 5V | VCC | 5883L VCC |
| GND | GND | 5883L GND |
| D2 | D2 | To Button |
| GND | GND | To Button |

The push button is a normally open one.

Copy Sketch

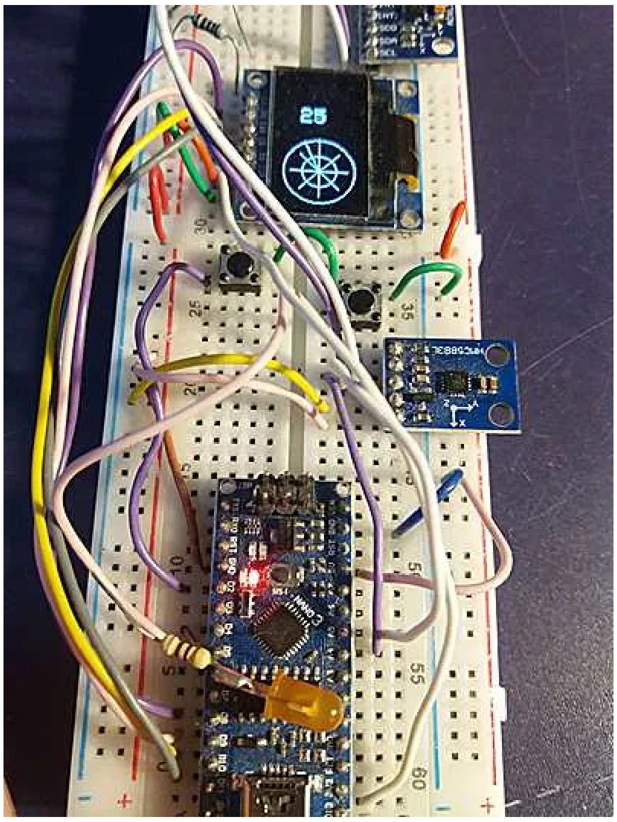
```
#include "Wire.h" #include <EEPROM.h>

#define BUTTON_CAL 2
#define LED  13
#define EEADDR 66 #define CALTIME 10000  static byte stat,ovfl,skipped;
static int minx,maxx,miny,maxy,offx=0,offy=0;

void I2C_write_AddrDev_AddrReg_Byte(byte i2cAddr, byte regaddr, byte d ) {
  Wire.beginTransmission(i2cAddr);
  Wire.write(regaddr);
  Wire.write(d);
  Wire.endTransmission();
}

void calc_offsets(void)  {
  offx = (maxx+minx)/2;
  offy = (maxy+miny)/2;
}

byte magnetometerReady(void) {
    Wire.beginTransmission(0x0d);    Wire.write(0x06);
  int num = Wire.requestFrom((byte)0x0d, (byte)1);
  stat = Wire.read();    Wire.endTransmission();

  ovfl    = stat & 0x02;
  skipped = stat & 0x04;

  return (stat && 0x01);  }

byte getMagnetometerRaw(int16_t *x, int16_t *y,int16_t *z) {

  if ( !magnetometerReady() ) return 0;

  Wire.beginTransmission(0x0d);
  Wire.write(0x00);        int err = Wire.endTransmission();

  if (!err) {
    Wire.requestFrom((byte)0x0d, (byte)6);       while(Wire.available()<6);       *x  = (int16_t)(Wire.read() | Wire.read() << 8);
    *y  = (int16_t)(Wire.read() | Wire.read() << 8);
    *z  = (int16_t)(Wire.read() | Wire.read() << 8);
  }
  return 1;
}

void getMagnetometer(int16_t *x, int16_t *y, int16_t *z) {

  if ( !getMagnetometerRaw(x, y, z) ) return;
  int tmp = *y;
  *y = -*x;        *x = tmp;     }

void readMagnetometer(int16_t *x, int16_t *y,int16_t *z) {
  while( !magnetometerReady() );
  getMagnetometer(x, y, z);  }

void setup() {

  pinMode(LED,OUTPUT);
  pinMode(BUTTON_CAL,INPUT_PULLUP);
  Wire.begin();    Wire.setClock(100000);    Serial.begin(115200);
  Serial.println("QMC5883L Digital compass: 3 axis");
  Serial.println("QMC5883L start initialise.");

    I2C_write_AddrDev_AddrReg_Byte(0x0d,0x0b,1);
  I2C_write_AddrDev_AddrReg_Byte(0x0d,0x09,B00000001);

  Serial.println(F("DONE initialise"));

    int EEAddr = EEADDR;
  EEPROM.get(EEAddr,minx); EEAddr +=sizeof(minx);
  EEPROM.get(EEAddr,maxx); EEAddr +=sizeof(maxx);
  EEPROM.get(EEAddr,miny); EEAddr +=sizeof(miny);
  EEPROM.get(EEAddr,maxy); EEAddr +=sizeof(maxy);
  calc_offsets();
}


void calibrate() {
  unsigned long calTimeWas = millis();
  byte cal = 1, startCal = 1;
  int x,y,z;
  float deg=0,deg2=0;

  readMagnetometer(&x, &y, &z);

  maxx = minx = x;    maxy = miny = y;

  while(cal) {

    if ( magnetometerReady() ) getMagnetometer(&x, &y, &z);
    if (x>maxx) maxx = x;
    if (x<minx) minx = x;
    if (y>maxy) maxy = y;
    if (y<miny) miny = y;

    Serial.print("CALIBRATE ");

    int secmillis  = millis()-calTimeWas;
    int secs = (int)((CALTIME-secmillis+1000)/1000);
    Serial.print("--> ");
    Serial.println((CALTIME-secmillis)/1000);

    if (secs==0) {          calc_offsets();
      cal = 0;

      int EEAddr = EEADDR;
      EEPROM.put(EEAddr,minx); EEAddr +=sizeof(minx);
      EEPROM.put(EEAddr,maxx); EEAddr +=sizeof(maxx);
      EEPROM.put(EEAddr,miny); EEAddr +=sizeof(miny);
      EEPROM.put(EEAddr,maxy); EEAddr +=sizeof(maxy);
      Serial.println("EEPROM Written");         }

    delay(10);
  } }

void loop() {
  static unsigned long BLTimeWas=millis();
  static int x,y,z;    static int bearing;
  if (digitalRead(BUTTON_CAL)==0) calibrate();

  getMagnetometer(&x, &y, &z);

  int atan2val = 180/M_PI * atan2((float)(x-offx),(float)(y-offy));
  bearing = (-atan2val + 360 ) % 360;

  if (millis()-BLTimeWas>400) {       BLTimeWas = millis();
    static byte togLED=0;
    togLED = !togLED;
    if(togLED) digitalWrite(LED,HIGH); else digitalWrite(LED,LOW);

    Serial.println(bearing);
  }

}
```

[ File: qmc5883l-compass.ino ]

## Example OLED Screen Compass

The following example operates in exactly the same way as the first example, but instead, the output display is an OLED SSD1306. - a graphical output display. Here it shows a bearing of 25° - this is the bearing of the y coordinate of the board i.e. the straight ahead bearing.

Magnetic North is shown by the line angled 25 Degrees to the left - the one that touches the outer circle - this is the one that moves round the compass rose as you move the board around.



As you can see in the picture above, the 5583 is aligned with x down and y to the right.

To get the system coordinates all calculations are made using the orientation of the OLED screen - otherwise your head blows up! So the magnetometer output is modified to return the following:

Screen x =  Magy (As Magnetometer y is aligned with the screen x axis).
Screen y = -Magx (As Magnetometer x is inverted and aligned to screen y axis).
These two calculations (inside getMagnetometer) convert the output from the magnetometer to the screen orientation.

The code also uses two functions:

drawLine()   - a wrapper function around display.drawLine(...)
drawCircle() - a wrapper function around display.drawCircle(...)
These draw lines and circles with the x,y reference point of (0,0) at bottom left. The graphics libraries use (0,0) as top left. They are labelled as normalized to indicate that the normal graph functions use y as up, and x to the right.

Doing this allows you to focus on the problem since now the screen matches the "normal" graph orientation.

The code calculates the arctangent of the magnitude in y over the magnitude in x - the usual mathematical operation of opposite over adjacent when looking at axes in standard graph operations.

The code use atan2 which takes into account the signs of opposite and adjacent. This gives a four quadrant angle unlike the atan function which returns a single quadrant (atan looses information while atan2 retains it).

Using the two functions drawLine and drawCircle makes it easier to create the draw bearing operatoin contained in the function:

drawBearing(float bearing, int midx, int midy, int radius)
This function still uses the CCW maths angle rotation but centres the bearing at xy coordingates midx and midy, and the bearing length will be length 'radius'. It also rotates the output angle so that zero degrees (x=0) is aligned vertically. You could say that this is not really the bearing but a vector with angle zero pointing vertically up.

The problem is that sin, tan and cos functions use a CCW rotating angle and a compass bearing uses a CW rotating angle. To convert the angle to a bearing the following calculation is made:

getMagnetometer(&x, &y, &z);
int atan2val = 180/M_PI * atan2((float)(x-offx),(float)(y-offy));
bearing = (-atan2val + 360 ) % 360;
The first line gives raw output from the magnetometer but adjusted for the orientation of x and y in this application (see earlier description above).

The values xoff and yoff are the calibration values so the results (x-xoff) and (y-yoff) represent the orientation of the magnetometer without any magnetic interference.

The last line constrains the output of the bearing value to 360 degrees and changes the rotational direction of the the bearing line.

## Hardware for Example Sketch 2

Arduino Nano (or Uno),
HMC5883L (or QST5883L) breakout board,
Push to make button (1 off),
SSD1306 OLED display.

## Hardware connections for Example Sketch2

Make connections as follows:

| Arduino Uno/ Nano pin | Label | Destination |
|---|---|---|
| A5 | SCL | 5883L SCL |
| A4 | SDA | 5883L SDA |
| 5V | VCC | 5883L VCC |
| GND | GND | 5883L GND |
| D2 | D2 | To Button |
| GND | GND | To Button |
| D11 | MOSI | OLED_MOSI |

| | | |
|---|---|---|
| D13 | CLK | OLED_CLK |
| D9 | DC | OLED_DC |
| D8 | CS | OLED_CS |
| D10 | RST | OLED_RESET |

The push button is a normally open one.

Copy Sketch

```
#include "Wire.h" ;# For 5883
#include <SPI.h>  ;# For SSD1306 board.
#include <EEPROM.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "I2Cdev.h"

#define OLED_MOSI  11
#define OLED_CLK   13       #define OLED_DC     9
#define OLED_CS     8
#define OLED_RESET 10

#define BUTTON_CAL  2
#define BUTTON_TEST 5

#define LED  4 #define EEADDR 66 #define CALTIME 10000  #define SCREEN_HEIGHT 128 #define SCREEN_WIDTH   64

static byte stat,ovfl,skipped;
static int minx,maxx,miny,maxy,offx=0,offy=0;

const int radius = 64/2;
const int midx = radius;
const int midy = radius;

Adafruit_SSD1306 display(OLED_DC, OLED_RESET, OLED_CS); void I2C_write_AddrDev_AddrReg_Byte(byte i2cAddr, byt
  Wire.beginTransmission(i2cAddr);
  Wire.write(regaddr);
  Wire.write(d);
  Wire.endTransmission();
}

void calc_offsets(void)  {
  offx = (maxx+minx)/2;
  offy = (maxy+miny)/2;
}

byte magnetometerReady(void) {
    Wire.beginTransmission(0x0d);    Wire.write(0x06);
  int num = Wire.requestFrom((byte)0x0d, (byte)1);
  stat = Wire.read();    Wire.endTransmission();

  ovfl    = stat & 0x02;
  skipped = stat & 0x04;

  return (stat && 0x01);  }

byte getMagnetometerRaw(int16_t *x, int16_t *y,int16_t *z) {

  if ( !magnetometerReady() ) return 0;

  Wire.beginTransmission(0x0d);
  Wire.write(0x00);        int err = Wire.endTransmission();
  if (!err) {
    Wire.requestFrom((byte)0x0d, (byte)6);      while(Wire.available()<6);      *x  = (int16_t)(Wire.read() | Wire.read() << 8);
    *y  = (int16_t)(Wire.read() | Wire.read() << 8);
    *z  = (int16_t)(Wire.read() | Wire.read() << 8);
  }
  return 1;
}

void getMagnetometer(int16_t *x, int16_t *y, int16_t *z) {

  if ( !getMagnetometerRaw(x, y, z) ) return;
  int tmp = *y;
  *y = -*x;       *x = tmp;     }

void readMagnetometer(int16_t *x, int16_t *y,int16_t *z) {
  while( !magnetometerReady() );
  getMagnetometer(x, y, z);  }

void setup() {

  pinMode(LED,OUTPUT);

  pinMode(BUTTON_CAL,INPUT_PULLUP);
  pinMode(BUTTON_TEST,INPUT_PULLUP);

  Wire.begin();    Wire.setClock(100000);    display.begin(SSD1306_SWITCHCAPVCC);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);

    I2C_write_AddrDev_AddrReg_Byte(0x0d,0x0b,1);
  I2C_write_AddrDev_AddrReg_Byte(0x0d,0x09,B00000001);

    int EEAddr = EEADDR;
  EEPROM.get(EEAddr,minx); EEAddr +=sizeof(minx);
  EEPROM.get(EEAddr,maxx); EEAddr +=sizeof(maxx);
  EEPROM.get(EEAddr,miny); EEAddr +=sizeof(miny);
  EEPROM.get(EEAddr,maxy); EEAddr +=sizeof(maxy);
  calc_offsets();

  display.setRotation(1);
}

void drawLine (int x1, int y1, int x2, int y2) {
   display.drawLine(x1, SCREEN_HEIGHT-y1, x2, SCREEN_HEIGHT-y2, 1);
}

void drawCircle (int x, int y, int radius) {
  display.drawCircle(x, SCREEN_HEIGHT-y, radius-1,1);
}

void drawBearing(float bearing, int midx, int midy, int radius) {
  bearing += 90;    int opp = sin(bearing*M_PI/180)*radius;
  int adj = cos(bearing*M_PI/180)*radius;
    drawLine(midx, midy, midx+adj, midy+opp);
}

void calibrate(boolean eeprom_write) {
  unsigned long calTimeWas = millis();
```

```
int x,y,z;
float deg=0,deg2=0;

readMagnetometer(&x, &y, &z);

maxx = minx = x;    maxy = miny = y;

delay(300);    while(1) {        if (digitalRead(BUTTON_CAL)==0 || digitalRead(BUTTON_TEST)==0) {
    delay(300);          return;       }

  if ( magnetometerReady() ) getMagnetometer(&x, &y, &z);
  if (x>maxx) maxx = x;
  if (x<minx) minx = x;
  if (y>maxy) maxy = y;
  if (y<miny) miny = y;

  display.clearDisplay();

  display.setTextSize(2);
  display.setCursor(0,0);

  if(eeprom_write) display.print("CALIBRATE ");
  else             display.print("TEST ");

  int secmillis  = millis()-calTimeWas;
  if (secmillis>CALTIME) break;                          int secs = (int)((CALTIME-secmillis+1000)/1000);
  display.setCursor(0,32);  display.print("--> "); display.print((int)((CALTIME-secmillis)/1000));

  drawBearing((int)deg2, midx, midy , radius);
  drawBearing((int)deg,  midx, midy , radius);

  deg = (360.0/CALTIME)*secmillis;       deg2 += deg;       deg = fmod(deg,360);

  for(int i=0;i<360;i+=45)           drawBearing(i + (45/secs)*10, midx, midy, radius-7);

  display.display();        delay(10);
}    calc_offsets();

if(eeprom_write) {
  int EEAddr = EEADDR;
  EEPROM.put(EEAddr,minx); EEAddr +=sizeof(minx);
  EEPROM.put(EEAddr,maxx); EEAddr +=sizeof(maxx);
  EEPROM.put(EEAddr,miny); EEAddr +=sizeof(miny);
  EEPROM.put(EEAddr,maxy); EEAddr +=sizeof(maxy);
}

unsigned long dispExitTimeWas = millis();

while(1) {

  display.clearDisplay();

  display.setTextSize(2);
  display.setCursor(0,0);

      if(eeprom_write) display.print("EEPROM Written");
  else             display.print("TEST DMY Write");
  if (millis()- dispExitTimeWas>2000) break;

  display.display();        delay(10);
}
```

```
}

void loop() {
static unsigned long BLTimeWas=millis();
int x,y,z; int bearing,i;

  if (digitalRead(BUTTON_CAL)==0) calibrate(1);
  if (digitalRead(BUTTON_TEST)==0) calibrate(0);

  getMagnetometer(&x, &y, &z);

  int atan2val = 180/M_PI * atan2((float)(x-offx),(float)(y-offy));
  bearing = (-atan2val + 360 ) % 360;

    display.clearDisplay();

    drawCircle(midx, midy, radius-1);       drawCircle(midx, midy, (radius-1)/2);

  for( i=0;i<360;i+=45)        drawBearing(i , midx, midy, radius-7);

  drawBearing(bearing, midx, midy, radius);
  drawBearing(0, midx, midy, radius);          display.setTextSize(2);
  int lineH = 16;
  int lineNum=2;

  display.setCursor(15,lineH*lineNum);  display.print(bearing);
  display.print((char)247);
  lineNum++;     if(x>maxy || x<minx || y>maxy || y<miny) {
    display.setCursor(0,lineH*lineNum++);  display.print("*CAL*");
  }

  display.display();     if (millis()-BLTimeWas>400) {       BLTimeWas = millis();
    static byte togLED=0;
    togLED = !togLED;
    if(togLED) digitalWrite(LED,HIGH); else digitalWrite(LED,LOW);
  }
}
```

[ File: qmc5883l-compass-ssd1306.ino ]

The program shows that you can make a quite useful compass using a magnetometer (HMC5883L or QST5883L). The problem with these devices is that you have to keep them horizontal to get an accurate reading.

One way of solving this problem is to add an accelerometer to your system. Using data from that chip you can compensate for tilt in the magnetometer to get a reading that does not vary with horizontal orientation.

## HMC5883L Breakout Board (three chips)

The HMC and QST versions of the chip have the same pinout so you could have either type of device. The only way to tell the difference is by the label printed on the chip (here shown as 5883 so this is a QST device).

The image below shows the layout of the breakout board (Connection names are also printed on the back):

GY 271 HMC5883L Breakout Board

This board is for 5V use as it has a 3V3 regulator to supply the QMC5883L. It also has a dual N-Channel Mosfet, used as two bi-directional logic level translators (for SDA and SCL). The level translators allow the connecting device to drive the signals at 5V while the 5883 can drive the signals back at 3V3.

Note: The DRDY signal is only sent from the board to the microcontroller, and therefore, does not require a level translator. It is of high enough voltage (Voh) to be seen by the microcontroller as a logic one.

## Chips on the three-chip Breakout Board

5883 - This is the QMC5883L chip.
If your's has the text 'L883' on it, then you have the HMC5883L chip.

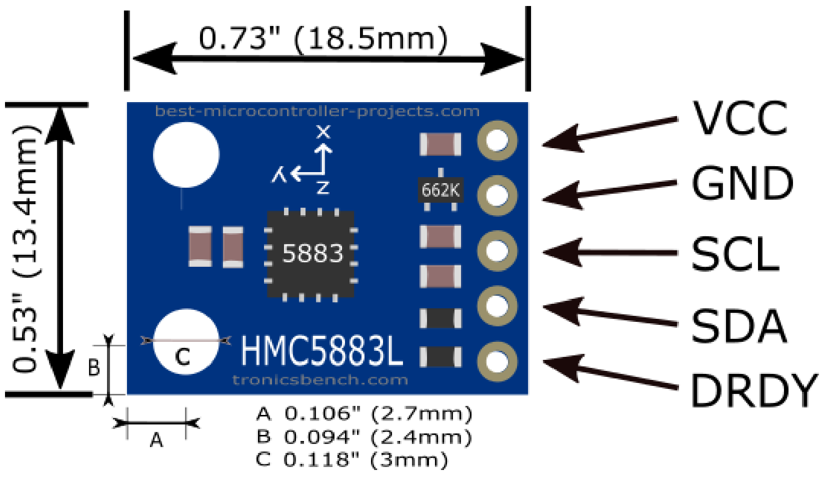
4A2D - Ultra low drop out regulator LD3985.
measured op=3V3 so part is:LD3985M33R.
The dropout is:
0.4mV ( typ) @ 1mA ~ 60mV( typ) @150mA.
2.0mV (max) @1mA ~ 100mV(max) @150mA.

702 - Dual N-Channel Fet (~L2N7002DW1T1G)
The RDS(ON) of the MOSFET is 7Î© - that's Ok depending on the application - here, its just logic level so the currents are low, but you can get many newer MOSFETS with RDSON in the range 70~100mÎ© (or lower) for higher current operation (if needed).

## Alternative breakout board (two chips)

Here is an image of another board I bought; Although HMC5883L is printed on the silkscreen, a QST device was fitted, so don't trust the silkscreen label!.

GY 273 HMC5883L Breakout Board



Notice how the board is labelled HMC5883L even though it does in fact have a QMC5883 soldered down! This is a little bit naughty, but understandable as the manufacturer probably has 1000's of pcbs and the QMC part does have the same pinout as the HMC version. However if an HMC version was fitted then the main volts from the microcontroller control outputs would have to be designed for a maximum output of 4.8V.

Notice also that the board connections are mirrored compared to the 3-chip board. This is more apparent if you keep x,y,z symbol in the same orientation (not done here as main HMC text would be upside down).

One big advantage that this board has (apart from being cheaper) is the mounting holes which allow robust fitting to a drone for instance.

## Chips on the two-chip Breakout Board

5883 - This is the QMC5883L chip.
If yours has the text 'L883' on it, then you have the HMC5883L chip.

662K - Low drop out regulator: XC6206P332MR
This is a three pin version of a 3V3 LDO.
The dropout is @3V3:
75mV ( typ) @ 30mA ~ 250mV( typ) @100mA.
350mV ( max) @ 30mA ~ 680mV( max) @100mA.

## About the Breakout Boards

The first board uses the "safe" way of designing a board - the way a production engineer would design it i.e. keep to the specified maximum 3V6 voltage levels using a 3V3 voltage regulator and level translator MOSFETs for each digital I/O used for the I2C interface.

In a production environment you keep to what the datasheet says because the manufacturer has tested many different devices to obtain the stated operation. You don't want to be the one who is blamed for a product recall involving thousands of units!

For the second board someone noticed that the QMC chip is stated to withstand >5V I/O and decided to use that fact and ignore the stated max operating condition of 3V3. Since the chip won't blow up for a 5V input why not use the Arduino (5V) output to drive the I2C signals directly and see what happens.

It works! but you should probably not use this method in a production environment.

The QMC version of the chip is an enhanced HMC chip. They both have the same pinout and the same fundamental internal operation but the QMC version is enhanced with automatic temperature compensation and offset cancellation.

All internal registers are different, and the ADC resolutions are different (QMC: 16bit, HMC 12bit).

In fact it turns out that the newer QMC5883L has some significant differences to the original design:

Higher resolution internal ADC (16bit).
Faster update rate (192Hz - 20% faster).
Automatic temperature compensated output.
Built in Self test removed (or at least hidden from the user).
I2C Addresses - different I2C addresses.
Internal Registers - different address locations.
Absolute maximum voltage - higher.
Oversampling / Averaging - more and better options.
Gauss Ranges - less ranges but easier to use.

## ADC resolution

Appears not to have any effect on the output resolution as both datasheets claim the lowest measured Gauss field is 2mG. However the increased ADC resolution (16 bit ADC - the HMC5883L has a 12 bit ADC) allows less range switching for the QMC chip version which means less software range switching making it is easier to use. Unlike the HMC it won't saturate on a range and require switching to a new range to get a result. See Gauss Range.

## Update rate

The increased update rate means that you can get a 20% increase in reading rate; The HMC5883L has a max output rate of 160Hz (the QMC is capable of a 192Hz output rate).

## Compensated output

There are several paragraphs in the HMC5883L datasheet describing how to obtain a corrected output by performing a self test. The self test also generates results that can be used to compensate for temperature drift. These paragraphs do not exist in the QMC5883L datasheet!

In the QMC5883L there is no self test and in the summary it states:

" it [ the QMC5883L ] offers the advantages of low noise, high accuracy, low power consumption, offset cancellation and temperature compensation "

For the HMC5883L you have to perform a built in self test to get the delta temperature (change since last reading) and then apply a scale factor to the outputs.

So the QMC5883L is easier to use because this feature has been automated for you.

## Built in self test

This has been removed in the QMC588L (now not accessible for user control) but it is probably still there and used as an automatic calibration mechanism.

## I2C addresses

QMC5883L has a single address for read and write (0D).
HMC5883L has a read address(3D) and a write address (3C).

## Absolute maximum voltage

QMC5883L abs. max is 5.4V.
HMC5883L abs. max is 4.8V.
The QMC5883L won't blow up if you use a normal 5V supply however the HMC5883L will be damaged (if the datasheet is to be believed!).

Note: The maximum operational Supply voltage is 3V6 in both cases.

However I have a board that has no level shifters and it does work at 5V (direct connection to arduino at 5V - the board has an internal 3V3 LDO regulator to supply the QMC.

This works since the maximum absolute input voltage is 5.4V (QMC only, for the HMC it is 4.8V). Should you do this? In a production environment - absolutely not (not tested by manufacturer and not guaranteed to work). In a hobby environment - well it works!

## Oversampling / Averaging

QMC5883L allows oversampling of data up to 512 times.
HMC5883L allows averaging of data up to 8 times.
Oversampling and averaging are the same thing - the data sheets just use different words. You can see that the output from the QMC chip is oversampled 64 more times than the HMC chip - and that means a cleaner output result (since it is averaged more).

## Gauss Range

QMC5883L has 2 gain ranges: 2G and 8G.
HMC5883L has 8 gain ranges:  from 0.88G to 8.1G.
At first this looks like a major blooper in the QMC5883L, however the chip has a higher resolution ADC (16bit whereas the HMC has a 12bit ADC). This means it has a finer Gauss measurement resolution compared to the HMC5883L.

With the higher resolution ADC you won't need to change ranges as much (there are only two available for the QMC), so it is easier to use.

Note: The default Gauss range for the HMC5883L is: 0 ~ ±1.3G . For a resolution of 1.49G/2047 = 0.73mG. For the QMC it is 0G ~  2.73G  @83uG (see below).

For the QMC5883:

minimum  gain: ±2G 1/12000 = ±83.3uG/LSB   : 0G ~  2.73G
maximum gain:  ±8G 1/3000  = ±0.244mG/LSB: 0G ~ 10.92G

There are only two ranges:
    The ADC Output numerical range is -32768 and 32767 for each setting.

For the HMC5883L

minimum range (gain setting = 1) ±0.73mG/LSB (1.0G/1370): 0G ~ 1.49G
...
...
maximum range (gain setting = 8) ±4.35mG/LSB (1.0G/230) : 0G ~ 8.91G

In all 8 cases (only the 1st and last are shown above):
    The ADC Output numerical range is -2048 ~ 2047 for each setting.
For the HMC5883L you have to be aware that the device range may be incorrect. You do this by monitoring the returned value. If it is set to -4096, then the input has saturated (ADC overflow indicating a too strong Gauss field), and you need to switch to the next higher Gauss range.

This could be a bit of a pain for data processing as it is using a data output register as an error flag i.e. You will have to read and suppress this value when using this chip.

Note: The QMC5883L has a much finer resolution in the lowest range 83uG compared to 0.73mG for the HMC version (~8.8 times the resolution). However in both datasheets the minimum claimed resolution is 2mG. This could be due to noise or non linearity in the ADC etc. then again, they may just be taking a conservative approach to the actual resolution capability.

## Similarities between the chips

Pinout: They both have the same pinout. So one is a drop in replacement for the other. However note that I2C addresses and internal registers are not the same.

Power Supply: They use the same max digital operating voltage (max survivable PSU voltage is different).

Fundamental Output Values: Similar, but the HMC5883L needs lots of range changes to retrieve Gauss values (see Gauss Range).

## HMC5883L

HMC5883L Datasheet.

## QMC5883L

QMC5883L Datasheet.

## Dual N-Channel MOSFET

This is used on the 1st breakout board as two level translators.

Dual N-Channel MOSFET L2N7002DW1T1G.

## Ultra LDO regulator (5 pin)

This is the 5 pin SOT23-5L regulator.

Ultra low dropout regulator LD3895.

## LDO regulator (3 pin)

This is the 3 pin SOT-23 regulator. This is not as good as the LD3895 (above).

low dropout regulator XC6206.

The HMC5338L and QST5883L perform the same operation but use different I2C addresses and different internal registers locations to do it, but they do have exactly the same pinout so you can use the QMC version as a replacement for the HMC version. You just need different software!

Since they generate the same output data (in Gauss) the calibration process is the same.

The single chip version of the breakout board has useful mounting holes not present on the 3 chip one. However, it ignores the specified max digital operating voltage of 3V6 (used out-of-spec) but it seems to work! However working with something used out of spec will probably cause readings to be off, or some other non obvious effect (or the manufacturer just did not test it at the higher voltage),

It is a case of suck-it-and-see, but if you are developing a system for a commercial interest then use the correct voltage level translation version of the circuit.

You might also be interested in:

How to tilt compensate this compass.
How to measure tilt (ADXL345).