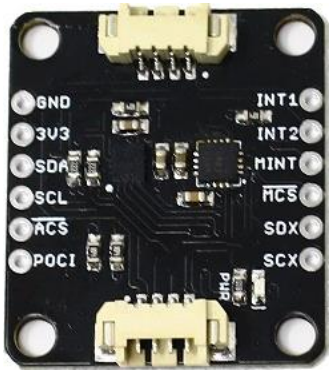
SmartElex 9DoF IMU Breakout - ISM330DHCX, MMC5983MA



The MMC5983MA combines the high-performance ISM330DHCX 3D digital accelerometer and gyroscope from STMicroelectronics with the highly sensitive triple-axis magnetometer by MEMSIC to give you an ultra powerful, easy to use breakout board.

With a full scale acceleration range of ±2/±4/±8/±16 g and a wide angular rate range of ±125/±250/±500/±1000/±2000/±4000 dps, as well as an unmatched set of embedded features (Machine Learning Core, programmable FSM, FIFO, sensor hub, event decoding and interrupts), the ISM330DHCX delivers high performance at very low power. Add to that the MMC5983MA, which can measure magnetic fields within the full scale range of ±8 Gauss (G), with 0.25mG/0.0625mG per LSB resolution at 16bits/18bits operation mode and 0.4 mG total RMS noise level and you've got 9 Degrees of Freedom on one tiny little breakout board.

Hardware Overview

ISM330DHCX 6DoF

The ISM330DHCX is a small, system-in-package from STMicroelectronics featuring a high-performance 3D digital accelerometer and 3D digital gyroscope capable of wide bandwidth, ultra-low noise and a selectable full-scale range of ±2/±4/±8/±16 g. The 3D

gyroscope has an angular rate range of ±125/±250/±500/±1000/±2000/±4000 dps and offers superior stability over temperature and time along with ultra-low noise.

The ISM330DHCX as implemented in the 9DoF can run in two different modes:

Mode 1

This is the default "peripheral only" mode. This mode allows you to use either I^2C or SPI. By default, I^2C is enabled with an address of 0x6B. By manipulating the associated jumper, you can change the I^2C address to 0x6A (cut the power side and close the ground side) or switch to SPI mode (both jumpers open).

Mode 2

This mode enables a secondary I^2C port that the 6DoF controls; up to 4 external sensors can be connected to the I^2C controller interface of the device. External sensors communicate via the SCX and SDX (PICOX) lines - the SCX jumper will need to be opened.

MMC5983MA

The MMC5983MA is an AEC-Q100 qualified, fully integrated 3-axis magnetic sensor with on-chip signal processing and integrated I^2C/SPI bus. It has superior dynamic range and accuracy with ±8G FSR, 18bit operation, 0.4mG total RMS noise, and can enable heading accuracy of 0.5º. More information can be found in the datasheet.

Warning! It should be noted that the Z axis for the ISM330DHCX and the MMC5983MA are calibrated from opposite directions. The ISM330DHCX Z axis functions as looking through from top to bottom. The MMC5983MA Z axis functions as looking from the bottom of the board through to the top. This is denoted by the "MAG -Z" labeled dot on the underside of the board.

Connectors

Ther connector on either side of the 9DoF board - ISM330DHCX, MMC5983MA breakout to provide power and I^2C connectivity simultaneously. The default I^2C address for the ISM330DHCX is **0x6B**, and the I^2C address for the MMC5983MA Magnetometer is **0x30**.

Power

Ideally, power will be supplied via the connectors, but we've also broken out plated through hole pins to supply 3.3V and Ground, should you prefer. Make sure to pay attention to logic levels - supply voltage to this board should range from **1.71 V to 3.6 V**.

I^2C

For flexibility, we've broken out the I^2C functionality as seen below. Primary I^2C pins are broken out to SDA and SCL. Secondary I^2C pins are broken out to SDX and SCX. These pins are used solely for Mode 2- Sensor Hub Mode - where the ISM330DHCX reads other sensors. You must cut the SDX and SCX jumpers on the back of the board in order to use this mode.

SPI

Like the I^2C functionality, we've also broken out the SPI functionality to PTH pins.

SPI Pins for the ISM330DHCX are broken out to SDA, POCI, and the ACS (Accelerometer Chip Select).SPI pins for the MMC5983MA are broken out to SDA, POCI, and the MCS (Magnetometer Chip Select).

I^2C Address/SPI

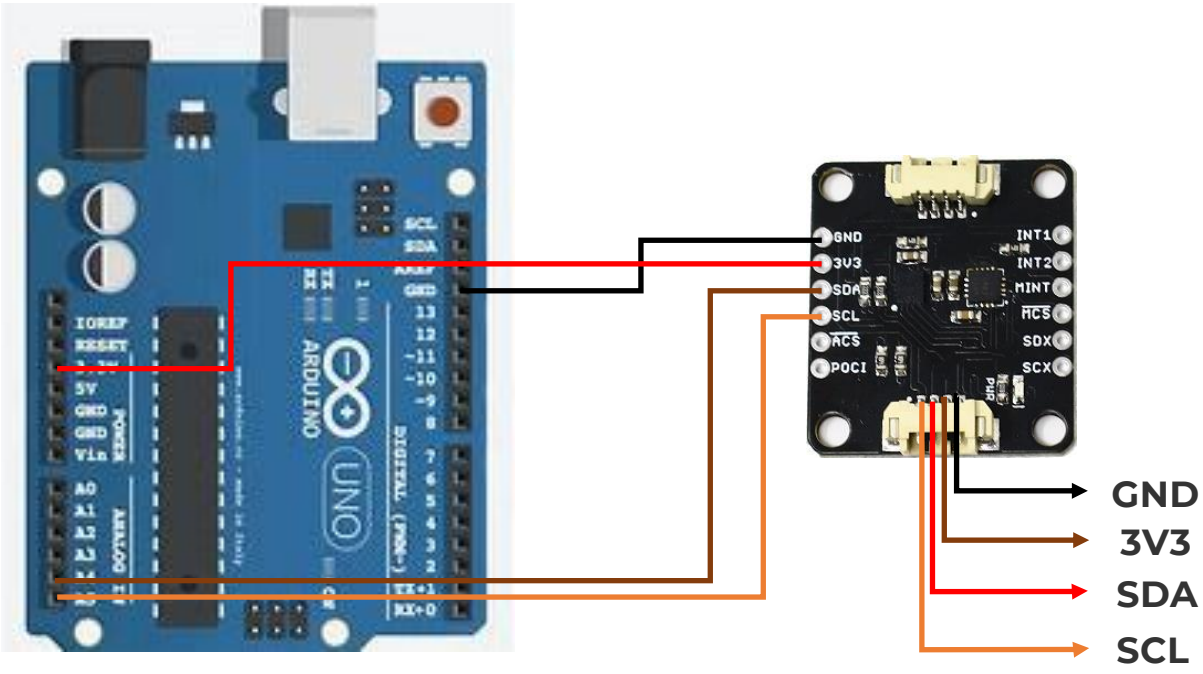
By default, I^2C is enabled with an address of 0x6B. By manipulating this jumper, you can change the I^2C address to 0x6A (cut the power side and close the ground side) or switch to SPI mode (both jumpers open).

SCX/SDX

The SCX and SDX pins are specific to Mode 2 of the ISM330DHCX and are used for peripheral communication. By default they are closed - to use Mode 2 you will need to cut both traces to open the jumper.

Wiring

Connecting the MMC5983MA to Arduino:



Arduino	MMC5983MA & ISM330DHCX
SCL(A5)	SCL
SDA(A4)	SDA
3.3v	3V3
GND	GND

Example Code

1.MMC5983MA-Magnetometer

```
#include <Wire.h>

#include <SparkFun_MMC5983MA_Arduino_Library.h> //Click here to get the library:
http://librarymanager/All#SparkFun_MMC5983MA

SFE_MMC5983MA myMag;
```

```
void setup()
{
  Serial.begin(115200);
  Serial.println("MMC5983MA Example");
  Wire.begin();
  if (myMag.begin() == false)
  {
    Serial.println("MMC5983MA did not respond - check your wiring. Freezing.");
    while (true) ;
  }
  myMag.softReset();
  Serial.println("MMC5983MA connected");
  int celsius = myMag.getTemperature();
  float fahrenheit = (celsius * 9.0f / 5.0f) + 32.0f;
  Serial.print("Die temperature: ");
  Serial.print(celsius);
  Serial.print("°C or ");
  Serial.print(fahrenheit, 0);
  Serial.println("°F.");
}
void loop()
{
  uint32_t currentX = 0;
  uint32_t currentY = 0;
  uint32_t currentZ = 0;
  double scaledX = 0;
```

```
double scaledY = 0;

double scaledZ = 0;

// This reads the X, Y and Z channels consecutively

// (Useful if you have one or more channels disabled)

currentX = myMag.getMeasurementX();

currentY = myMag.getMeasurementY();

currentZ = myMag.getMeasurementZ();

// Or, we could read all three simultaneously

//myMag.getMeasurementXYZ(&currentX, &currentY, &currentZ);

Serial.print("X axis raw value: ");

Serial.print(currentX);

Serial.print("\tY axis raw value: ");

Serial.print(currentY);

Serial.print("\tZ axis raw value: ");

Serial.println(currentZ);

// The magnetic field values are 18-bit unsigned. The _approximate_ zero (mid) point is 2^17 (131072).

// Here we scale each field to +/- 1.0 to make it easier to convert to Gauss.

//

// Please note: to properly correct and calibrate the X, Y and Z channels, you need to determine true

// offsets (zero points) and scale factors (gains) for all three channels. Futher details can be found at:

// https://thecavepearlproject.org/2015/05/22/calibrating-any-compass-or-accelerometer-for-arduino/

scaledX = (double)currentX - 131072.0;

scaledX /= 131072.0;

scaledY = (double)currentY - 131072.0;

scaledY /= 131072.0;

scaledZ = (double)currentZ - 131072.0;
```

```
  scaledZ /= 131072.0;
  // The magnetometer full scale is +/- 8 Gauss
  // Multiply the scaled values by 8 to convert to Gauss
  Serial.print("X axis field (Gauss): ");
  Serial.print(scaledX * 8, 5); // Print with 5 decimal places
  Serial.print("\tY axis field (Gauss): ");
  Serial.print(scaledY * 8, 5);
  Serial.print("\tZ axis field (Gauss): ");
  Serial.println(scaledZ * 8, 5);
  Serial.println();
  delay(100);
}
```

2.ISM330DHCX 6DoF IMU

```
#include <Wire.h>
#include "SparkFun_ISM330DHCX.h"
SparkFun_ISM330DHCX myISM;
// Structs for X,Y,Z data
sfe_ism_data_t accelData;
sfe_ism_data_t gyroData;
void setup(){
  Wire.begin();
  Serial.begin(115200);
  if( !myISM.begin() ){
    Serial.println("Did not begin.");
    while(1);  }
  // Reset the device to default settings. This if helpful is you're doing multiple
```

```
  // uploads testing different settings.
  myISM.deviceReset();
  // Wait for it to finish reseting
  while( !myISM.getDeviceReset() ){
    delay(1);
  }
  Serial.println("Reset.");
  Serial.println("Applying settings.");
  delay(100);
  myISM.setDeviceConfig();
  myISM.setBlockDataUpdate();
  // Set the output data rate and precision of the accelerometer
  myISM.setAccelDataRate(ISM_XL_ODR_104Hz);
  myISM.setAccelFullScale(ISM_4g);
  // Set the output data rate and precision of the gyroscope
  myISM.setGyroDataRate(ISM_GY_ODR_104Hz);
  myISM.setGyroFullScale(ISM_500dps);
  // Turn on the accelerometer's filter and apply settings.
  myISM.setAccelFilterLP2();
  myISM.setAccelSlopeFilter(ISM_LP_ODR_DIV_100);
  // Turn on the gyroscope's filter and apply settings.
  myISM.setGyroFilterLP1();
  myISM.setGyroLP1Bandwidth(ISM_MEDIUM);
}
void loop(){
  // Check if both gyroscope and accelerometer data is available.
```

```
 if( myISM.checkStatus() ){
   myISM.getAccel(&accelData);
   myISM.getGyro(&gyroData);
   Serial.print("Accelerometer: ");
   Serial.print("X: ");
   Serial.print(accelData.xData);
   Serial.print(" ");
   Serial.print("Y: ");
   Serial.print(accelData.yData);
   Serial.print(" ");
   Serial.print("Z: ");
   Serial.print(accelData.zData);
   Serial.println(" ");
   Serial.print("Gyroscope: ");
   Serial.print("X: ");
   Serial.print(gyroData.xData);
   Serial.print(" ");
   Serial.print("Y: ");
   Serial.print(gyroData.yData);
   Serial.print(" ");
   Serial.print("Z: ");
   Serial.print(gyroData.zData);
   Serial.println(" ");
  }
  delay(100);
}
```