Arduino Magnetometer Driver

Generated by Doxygen 1.8.9.1

Thu Feb 4 2016 20:21:55

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 MagnetometerHMC5883L::CRAbits Union Reference

```
#include <MagnetometerHMC5883L.h>
```

Public Attributes

- struct {
  unsigned char MS0:1
  unsigned char MS1:1
  unsigned char DO0:1
  unsigned char DO1:1
  unsigned char DO2:1
  unsigned char MA0:1
  unsigned char MA1:1
  unsigned char:1
  };

- struct {
  unsigned char MS:2
  unsigned char DO:3
  unsigned char MA:2
  unsigned char:1
  };

- unsigned char value

4.1.1 Detailed Description

Configuration Register A.

CRA6 to CRA5 (MA1 to MA0): Select number of samples averaged (1 to 8) per measurement output.

MA1 MA0 ->Number of samples. 00 -> 1 (Default); 01 -> 2; 10 -> 4; 11 -> 8;

CRA4 to CRA2 (DO2 to DO0): Data Output Rate Bits. These bits set the rate at which data is written to all three data output registers.

DO2 DO1 DO0 -> Typical Data Output Rate (Hz) 000 -> 0.75 001 -> 1.5 010 -> 3 011 -> 7.5 100 -> 15 (Default) 101 -> 30 110 -> 75 111 -> Reserved

CRA1 to CRA0 (MS1 to MS0): Measurement Configuration Bits. These bits define the measurement flow of the device, specifically whether or not to incorporate an applied bias into the measurement.

MS1 MS0 -> Measurement Mode 00 -> Normal measurement configuration (Default). In normal measurement configuration the device follows normal measurement flow. The positive and negative pins of the resistive load are left floating and high impedance. 01 -> Positive bias configuration for X, Y, and Z axes. In this configuration, a positive current is forced across the resistive load for all three axes. 10 -> Negative bias configuration for X, Y and Z axes. In this configuration, a negative current is forced across the resistive load for all three axes. 11 -> This configuration is reserved.

CRA default is 0x10

Definition at line 82 of file MagnetometerHMC5883L.h.

4.1.2 Member Data Documentation

4.1.2.1 struct { ... }

4.1.2.2 struct { ... }

4.1.2.3 unsigned MagnetometerHMC5883L::CRAbits::char

Definition at line 92 of file MagnetometerHMC5883L.h.

4.1.2.4 unsigned char MagnetometerHMC5883L::CRAbits::DO

Definition at line 96 of file MagnetometerHMC5883L.h.

4.1.2.5 unsigned char MagnetometerHMC5883L::CRAbits::DO0

Definition at line 87 of file MagnetometerHMC5883L.h.

4.1.2.6 unsigned char MagnetometerHMC5883L::CRAbits::DO1

Definition at line 88 of file MagnetometerHMC5883L.h.

4.1.2.7 unsigned char MagnetometerHMC5883L::CRAbits::DO2

Definition at line 89 of file MagnetometerHMC5883L.h.

4.1.2.8 unsigned char MagnetometerHMC5883L::CRAbits::MA

Definition at line 97 of file MagnetometerHMC5883L.h.

4.1.2.9 unsigned char MagnetometerHMC5883L::CRAbits::MA0

Definition at line 90 of file MagnetometerHMC5883L.h.

4.1.2.10 unsigned char MagnetometerHMC5883L::CRAbits::MA1

Definition at line 91 of file MagnetometerHMC5883L.h.

4.1.2.11 unsigned char MagnetometerHMC5883L::CRAbits::MS

Definition at line 95 of file MagnetometerHMC5883L.h.

4.1.2.12 unsigned char MagnetometerHMC5883L::CRAbits::MS0

Definition at line 85 of file MagnetometerHMC5883L.h.

4.1.2.13 unsigned char MagnetometerHMC5883L::CRAbits::MS1

Definition at line 86 of file MagnetometerHMC5883L.h.

4.1.2.14 unsigned char MagnetometerHMC5883L::CRAbits::value

Definition at line 100 of file MagnetometerHMC5883L.h.

The documentation for this union was generated from the following file:

- MagnetometerHMC5883L.h

4.2 MagnetometerHMC5883L::CRBbits Union Reference

```
#include <MagnetometerHMC5883L.h>
```

Public Attributes

- struct {
  unsigned char:5
  unsigned char GN0:1
  unsigned char GN1:1
  unsigned char GN2:1
  };

- struct {
  unsigned char:5
  unsigned char GN:3
  };

- unsigned char value

4.2.1 Detailed Description

Configuration Register B.

The configuration register B for setting the device gain. CRB0 through CRB7 indicate bit locations, with CR←B denoting the bits that are in the configuration register. CRB7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

CRB7 to CRB5 (GN2 to GN0): Gain Configuration Bits. These bits configure the gain for the device. The gain configuration is common for all channels.

GN2 GN1 GN0 -> {Recommended Sensor Field Range} {Gain (LSb/Gauss)} {Digital Resolution (mG/LSb)} {Output Range} 000 -> ±0.88Ga 1370 0.73 {0xF800–0x07FF (-2048–2047)} 001 -> ±1.3Ga {1090 (default)} 0.92 {0x←F800–0x07FF (-2048–2047)} 010 -> ±1.9Ga 820 1.22 {0xF800–0x07FF (-2048–2047)} 011 -> ±2.5Ga 660 1.52 {0xF800–0x07FF (-2048–2047)} 100 -> ±4.0Ga 440 2.27 {0xF800–0x07FF (-2048–2047)} 101 -> ±4.7Ga 390 2.56

{0xF800–0x07FF (-2048–2047)} 110 -> ±5.6Ga 330 3.03 {0xF800–0x07FF (-2048–2047)} 111 -> ±8.1Ga 230 4.35 {0xF800–0x07FF (-2048–2047)}

CRB4 to CRB0 0: These bits must be cleared for correct operation.

CRB default is 0x20.

Definition at line 129 of file MagnetometerHMC5883L.h.

4.2.2 Member Data Documentation

4.2.2.1 struct { ... }

4.2.2.2 struct { ... }

4.2.2.3 unsigned MagnetometerHMC5883L::CRBbits::char

Definition at line 132 of file MagnetometerHMC5883L.h.

4.2.2.4 unsigned char MagnetometerHMC5883L::CRBbits::GN

Definition at line 139 of file MagnetometerHMC5883L.h.

4.2.2.5 unsigned char MagnetometerHMC5883L::CRBbits::GN0

Definition at line 133 of file MagnetometerHMC5883L.h.

4.2.2.6 unsigned char MagnetometerHMC5883L::CRBbits::GN1

Definition at line 134 of file MagnetometerHMC5883L.h.

4.2.2.7 unsigned char MagnetometerHMC5883L::CRBbits::GN2

Definition at line 135 of file MagnetometerHMC5883L.h.

4.2.2.8 unsigned char MagnetometerHMC5883L::CRBbits::value

Definition at line 141 of file MagnetometerHMC5883L.h.

The documentation for this union was generated from the following file:

- MagnetometerHMC5883L.h

4.3 MagnetometerHMC5983::EMRbits Union Reference

```
#include <MagnetometerHMC5983.h>
```

Public Attributes

- struct {
  unsigned char MD0:1
  unsigned char MD1:1
  unsigned char SIM:1
  unsigned char:2
  unsigned char LP:1
  unsigned char HS:1
  };

- struct {
  unsigned char MD:2

```
        unsigned char:6
    };
```

- unsigned char value

4.3.1 Detailed Description

(Extended) Mode Register

The mode register is an 8-bit register from which data can be read or to which data can be written. This register is used to select the operating mode of the device. MR0 through MR7 indicate bit locations, with MR denoting the bits that are in the mode register. MR7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. Mode register default is 0x01.

MR7 to MR2 0: Bit MR7 is set to 1 internally after each single-measurement operation. Set to 0 when configuring mode register.

MR1 to MR0 (MD1 to MD0): Mode Select Bits. These bits select the operation mode of this device.

MD1 MD0 -> Operating Mode 00 -> 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. 01 -> Single-Measurement Mode (Default). When single-measurement mode is selected, device performs a single measurement, sets RDY high and returned to idle mode. Mode register returns to idle mode bit values. The measurement remains in the data output register and RDY remains high until the data output register is read or another measurement is performed. 10 -> Idle Mode. Device is placed in idle mode. 11 -> Idle Mode. Device is placed in idle mode.

Definition at line 73 of file MagnetometerHMC5983.h.

4.3.2 Member Data Documentation

4.3.2.1 struct { ... }

4.3.2.2 struct { ... }

4.3.2.3 unsigned MagnetometerHMC5983::EMRbits::char

Definition at line 79 of file MagnetometerHMC5983.h.

4.3.2.4 unsigned char MagnetometerHMC5983::EMRbits::HS

Definition at line 82 of file MagnetometerHMC5983.h.

4.3.2.5 unsigned char MagnetometerHMC5983::EMRbits::LP

Definition at line 80 of file MagnetometerHMC5983.h.

4.3.2.6 unsigned char MagnetometerHMC5983::EMRbits::MD

Definition at line 85 of file MagnetometerHMC5983.h.

4.3.2.7 unsigned char MagnetometerHMC5983::EMRbits::MD0

Definition at line 76 of file MagnetometerHMC5983.h.

4.3.2.8 unsigned char MagnetometerHMC5983::EMRbits::MD1

Definition at line 77 of file MagnetometerHMC5983.h.

4.3.2.9 unsigned char MagnetometerHMC5983::EMRbits::SIM

Definition at line 78 of file MagnetometerHMC5983.h.

4.3.2.10 unsigned char MagnetometerHMC5983::EMRbits::value

Definition at line 88 of file MagnetometerHMC5983.h.

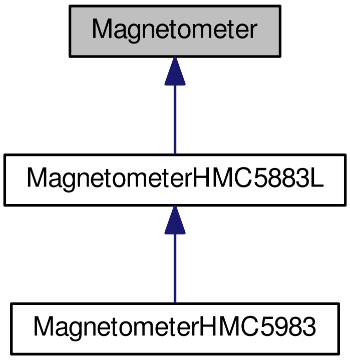
The documentation for this union was generated from the following file:

- MagnetometerHMC5983.h

4.4 Magnetometer Class Reference

```
#include <Magnetometer.h>
```

Inheritance diagram for Magnetometer:



Public Member Functions

- virtual ~Magnetometer ()
- virtual double getHeading ()=0
- double radiansToDegrees (double radians)
- double computeVectorAngle (int16_t x, int16_t y)

4.4.1 Detailed Description

Arduino - Magnetometer driver.

Interface for all Magnetometer (compass) implementations.

Author

Dalmir da Silva `dalmirdasilva@gmail.com`

Definition at line 15 of file Magnetometer.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Magnetometer::∼Magnetometer ( ) [virtual]

Definition at line 3 of file Magnetometer.cpp.

4.4.3 Member Function Documentation

4.4.3.1 double Magnetometer::computeVectorAngle (int16_t *x,* int16_t *y*)

To convert the micro-Tesla readings into a 0-360 degree compass heading, we can use the atan2() function to compute the angle of the vector defined by the Y and X axis readings.

The result will be in radians, so we multiply by 180 degrees and divide by Pi to convert that to degrees.

Parameters

x	X read in micro-tesla
y	Y read in micro-tesla

Returns

The heading in degrees.

Definition at line 10 of file Magnetometer.cpp.

4.4.3.2 virtual double Magnetometer::getHeading ( ) [pure virtual]

Gets the heading in degree.

Implemented in MagnetometerHMC5883L.

4.4.3.3 double Magnetometer::radiansToDegrees (double *radians*) [inline]

Radians to degrees.

Parameters

radians	Radians.

Returns

Degrees.

Definition at line 6 of file Magnetometer.cpp.

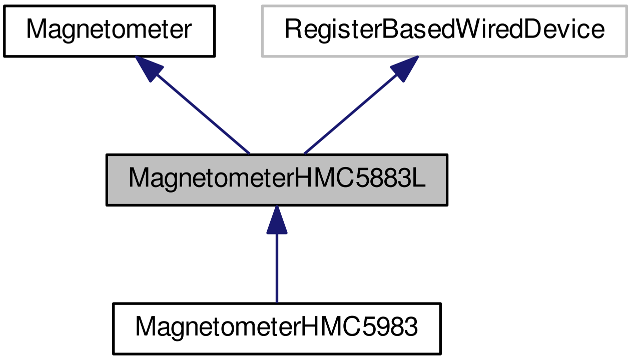
The documentation for this class was generated from the following files:

- Magnetometer.h
- Magnetometer.cpp

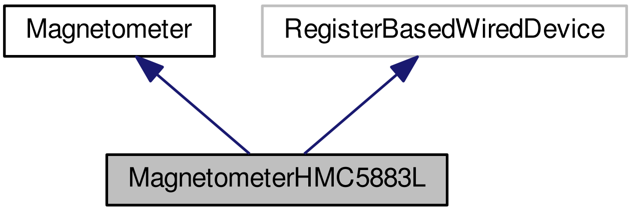
4.5 MagnetometerHMC5883L Class Reference

```
#include <MagnetometerHMC5883L.h>
```

Inheritance diagram for MagnetometerHMC5883L:



Collaboration diagram for MagnetometerHMC5883L:



Classes

- union CRAbits
- union CRBbits
- union MRbits
- union SRbits

Public Types

- enum Register {
  CRA = 0x00, CRB = 0x01, MR = 0x02, DXRA = 0x03,
  DXRB = 0x04, DYRA = 0x05, DYRB = 0x06, DZRA = 0x07,
  DZRB = 0x08, SR = 0x09, IDA = 0x0a, IDB = 0x0b,
  IDC = 0x0c }
- enum OperatingMode { IDLE_MODE = 0x00, CONTINUOUS_MEASUREMENT_MODE = 0x01, SINGLE_MEASUREMENT_MODE = 0x02 }
- enum SamplesAveraged { SA_1 = 0x00, SA_2 = 0x01, SA_4 = 0x02, SA_8 = 0x03 }

- enum DataOutputRate {
  DAR_0_75 = 0x00, DAR_1_5 = 0x01, DAR_3 = 0x02, DAR_7_5 = 0x03,
  DAR_15 = 0x04, DAR_30 = 0x05, DAR_75 = 0x06 }
- enum MeasurementMode { NORMAL_MEASUREMENT = 0x00, POSITIVE_BIAS = 0x01, NEGATIVE_BIAS = 0x02 }
- enum Gain {
  GAIN_0_88_GA = 0x00, GAIN_1_3_GA = 0x01, GAIN_1_9_GA = 0x02, GAIN_2_5_GA = 0x03,
  GAIN_4_0_GA = 0x04, GAIN_4_7_GA = 0x05, GAIN_5_6_GA = 0x06, GAIN_8_1_GA = 0x07 }

Public Member Functions

- MagnetometerHMC5883L ()
- void setOperatingMode (unsigned char operatingMode)
- virtual ∼MagnetometerHMC5883L ()
- void setSamplesAveraged (unsigned char samplesAveraged)
- void setDataOutputRate (unsigned char dataOutputRate)
- void setMeasurementMode (unsigned char measurementMode)
- void setGain (unsigned char gain)
- SRbits getStatusRegister ()
- void readSample (unsigned char buf[6])
- double getHeading ()

4.5.1 Detailed Description

The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry.

The HMC5883L includes our state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I2C serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9mm surface mount 16-pin leadless chip carrier (LCC). Applications for the HMC5883L include Mobile Phones, Netbooks, Consumer Electronics, Auto Navigation Systems, and Personal Navigation Devices.

Definition at line 35 of file MagnetometerHMC5883L.h.

4.5.2 Member Enumeration Documentation

4.5.2.1 enum MagnetometerHMC5883L::DataOutputRate

These bits set the rate at which data is written to all three data output registers.

Enumerator

DAR_0_75

DAR_1_5

DAR_3

DAR_7_5

DAR_15

DAR_30

DAR_75

Definition at line 388 of file MagnetometerHMC5883L.h.

4.5.2.2 enum MagnetometerHMC5883L::Gain

These bits configure the gain for the device.

The gain configuration is common for all channels.

Enumerator

GAIN_0_88_GA

GAIN_1_3_GA

GAIN_1_9_GA

GAIN_2_5_GA

GAIN_4_0_GA

GAIN_4_7_GA

GAIN_5_6_GA

GAIN_8_1_GA

Definition at line 412 of file MagnetometerHMC5883L.h.

4.5.2.3 enum MagnetometerHMC5883L::MeasurementMode

These bits define the measurement flow of the device, specifically whether or not to incorporate an applied bias into the measurement.

Enumerator

NORMAL_MEASUREMENT

POSITIVE_BIAS

NEGATIVE_BIAS

Definition at line 402 of file MagnetometerHMC5883L.h.

4.5.2.4 enum MagnetometerHMC5883L::OperatingMode

Modes Of Operation.

Continuous-Measurement Mode

During continuous-measurement mode, the device continuously makes measurements, at user selectable rate, and places measured data in data output registers. Data can be re-read from the data output registers if necessary; however, if the master does not ensure that the data register is accessed before the completion of the next measurement, the data output registers are updated with the new measurement. To conserve current between measurements, the device is placed in a state similar to idle mode, but the Mode Register is not changed to Idle Mode. That is, MD[n] bits are unchanged. Settings in the Configuration Register A affect the data output rate (bits DO[n]), the measurement configuration (bits MS[n]), when in continuous-measurement mode. All registers maintain values while in continuous-measurement mode. The I2C bus is enabled for use by other devices on the network in while continuous-measurement mode.

Single-Measurement Mode

This is the default power-up mode. During single-measurement mode, the device makes a single measurement and places the measured data in data output registers. After the measurement is complete and output data registers are updated, the device is placed in idle mode, and the Mode Register is changed to idle mode by setting MD[n] bits. Settings in the configuration register affect the measurement configuration (bits MS[n])when in single-measurement mode. All registers maintain values while in single-measurement mode. The I2C bus is enabled for use by other devices on the network while in single-measurement mode.

Idle Mode

During this mode the device is accessible through the I2C bus, but major sources of power consumption are disabled, such as, but not limited to, the ADC, the amplifier, and the sensor bias current. All registers maintain values while in idle mode. The I2C bus is enabled for use by other devices on the network while in idle mode.

Enumerator

- ***IDLE_MODE***
- ***CONTINUOUS_MEASUREMENT_MODE***
- ***SINGLE_MEASUREMENT_MODE***

Definition at line 369 of file MagnetometerHMC5883L.h.

4.5.2.5 enum MagnetometerHMC5883L::Register

Identification Register A.

The identification register A is used to identify the device. IRA0 through IRA7 indicate bit locations, with IR↩ A denoting the bits that are in the identification register A. IRA7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. The identification value for this device is stored in this register. This is a read-only register. Register values.

ASCII value H Identification Register B

The identification register B is used to identify the device. IRB0 through IRB7 indicate bit locations, with IRB denoting the bits that are in the identification register A. IRB7 denotes the first bit of the data stream. Register values.

ASCII value 4 Identification Register C

The identification register C is used to identify the device. IRC0 through IRC7 indicate bit locations, with IR↩ C denoting the bits that are in the identification register A. IRC7 denotes the first bit of the data stream. Register values.

ASCII value 3

Enumerator

- ***CRA***
- ***CRB***
- ***MR***
- ***DXRA***
- ***DXRB***
- ***DYRA***
- ***DYRB***
- ***DZRA***
- ***DZRB***
- ***SR***
- ***IDA***
- ***IDB***
- ***IDC***

Definition at line 323 of file MagnetometerHMC5883L.h.

4.5.2.6 enum MagnetometerHMC5883L::SamplesAveraged

Number of samples averaged (1 to 8) per measurement output.

Enumerator

- ***SA_1***
- ***SA_2***
- ***SA_4***
- ***SA_8***

Definition at line 378 of file MagnetometerHMC5883L.h.

4.5.3 Constructor & Destructor Documentation

4.5.3.1 MagnetometerHMC5883L::MagnetometerHMC5883L ( )

Public constructor.

The HMC5883L has a fairly quick stabilization time from no voltage to stable and ready for data retrieval. The nominal 56 milli-seconds with the factory default single measurement mode means that the six bytes of magnetic data registers (DXRA, DXRB, DZRA, DZRB, DYRA, and DYRB) are filled with a valid first measurement.

Definition at line 4 of file MagnetometerHMC5883L.cpp.

4.5.3.2 MagnetometerHMC5883L::∼MagnetometerHMC5883L ( ) [virtual]

Virtual destructor.

Definition at line 8 of file MagnetometerHMC5883L.cpp.

4.5.4 Member Function Documentation

4.5.4.1 double MagnetometerHMC5883L::getHeading ( ) [virtual]

Gets the heading in degree.

Implements Magnetometer.

Definition at line 11 of file MagnetometerHMC5883L.cpp.

4.5.4.2 MagnetometerHMC5883L::SRbits MagnetometerHMC5883L::getStatusRegister ( )

Gets the status register.

The status register is an 8-bit read-only register. This register is used to indicate device status. SR0 through SR7 indicate bit locations, with SR denoting the bits that are in the status register. SR7 denotes the first bit of the data stream.

Definition at line 42 of file MagnetometerHMC5883L.cpp.

4.5.4.3 void MagnetometerHMC5883L::readSample (unsigned char *buf[6]*)

Reads the sample.

Read all 6 bytes. If gain is changed then this data set is using previous gain.

Definition at line 48 of file MagnetometerHMC5883L.cpp.

4.5.4.4 void MagnetometerHMC5883L::setDataOutputRate (unsigned char *dataOutputRate*)

Sets data output rate.

These bits set the rate at which data is written to all three data output registers.

Parameters

dataOutputRate	Rate.

Definition at line 28 of file MagnetometerHMC5883L.cpp.

4.5.4.5 void MagnetometerHMC5883L::setGain (unsigned char *gain*)

Sets gain.

These bits configure the gain for the device. The gain configuration is common for all channels.

NOTE: Choose a lower gain value (higher GN#) when total field strength causes overflow in one of the data output registers (saturation). Note that the very first measurement after a gain change maintains the same gain as the previous setting.

Register: CRB

Parameters

gain	Gain.

Definition at line 36 of file MagnetometerHMC5883L.cpp.

4.5.4.6 void MagnetometerHMC5883L::setMeasurementMode (unsigned char *measurementMode*)

Set measurement mode.

These bits define the measurement flow of the device, specifically whether or not to incorporate an applied bias into the measurement.

Parameters

measurement↵ Mode	Measurement mode.

Definition at line 32 of file MagnetometerHMC5883L.cpp.

4.5.4.7 void MagnetometerHMC5883L::setOperatingMode (unsigned char *operatingMode*)

Configure operating mode.

The mode register is an 8-bit register from which data can be read or to which data can be written. This register is used to select the operating mode of the device. MR0 through MR7 indicate bit locations, with MR denoting the bits that are in the mode register. MR7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. Mode register default is 0x01.

Parameters

operatingMode	Operating Mode

Definition at line 20 of file MagnetometerHMC5883L.cpp.

4.5.4.8 void MagnetometerHMC5883L::setSamplesAveraged (unsigned char *samplesAveraged*)

Sets samples averaged.

Select number of samples averaged (1 to 8) per measurement output.

Parameters

samples↵ Averaged	Number of samples averaged (1 to 8) per measurement output.

Definition at line 24 of file MagnetometerHMC5883L.cpp.

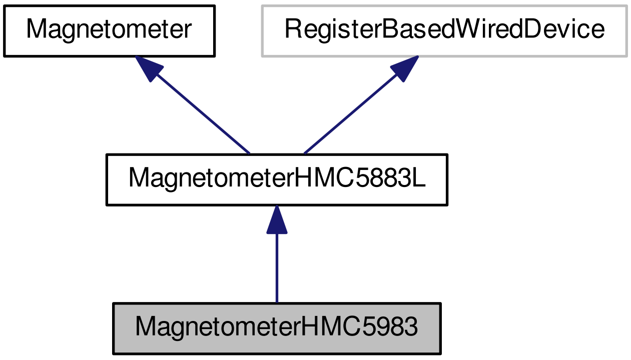
The documentation for this class was generated from the following files:

- MagnetometerHMC5883L.h
- MagnetometerHMC5883L.cpp

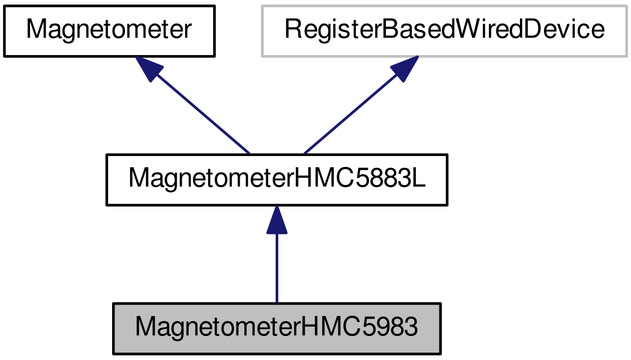
4.6 MagnetometerHMC5983 Class Reference

```
#include <MagnetometerHMC5983.h>
```

Inheritance diagram for MagnetometerHMC5983:



Collaboration diagram for MagnetometerHMC5983:



Classes

- union EMRbits
- union SRbits

Public Types

- enum ExtendedRegister { TEMPH = 0x31, TEMPL = 0x32 }
- enum SpeedMode { NORMAL_MODE = 0x00, HIGH_SPEED_MODE = 0x01 }
- enum TemperatureSensor { DISABLE_TEMPERATURE_SENSOR = 0x00, ENABLE_TEMPERATURE_SENSOR = 0x01 }
- enum LowestPowerMode { DISABLE_LOWEST_POWER_MODE = 0X00, ENABLE_LOWEST_POWER_MODE = 0X01 }
- enum SerialInterfaceMode { FOUR_WIRE = 0X00, THREE_WIRE = 0X01 }

Public Member Functions

- MagnetometerHMC5983 ()
- void setTemperatureSensor (unsigned char temperatureSensor)
- void setHighSpeedMode (unsigned char speedMode)
- void setLowestPowerMode (unsigned char lowestPowerMode)
- void setSerialInterfaceMode (unsigned char serialInterfaceMode)
- double getTemperature ()

4.6.1 Detailed Description

The same as MagnetometerHMC5883L but with temperature sensor.

Temperature Compensation

Temperature compensation of the measured magnetic data is enabled by default at the factory. Temperature measured by the built-in temperature sensor will be used to compensate the sensor's sensitivity change due to temperature based on the sensor's typical sensitivity temperature coefficient. The compensated data will be placed in the Data Output Registers automatically. Temperature sensor must be enabled (set CRA7 =1) for compensation to work.

Temperature Output

HMC5983 has a built-in temperature sensor that its output can be enabled by setting bit 7 of Configuration Register A (CRA7). This bit is disabled at power-on by default. When this feature is enabled, a temperature measurement will be taken at each magnetic measurement and the output is placed in Temperature Output Registers (0x31 and 0x32).

Definition at line 37 of file MagnetometerHMC5983.h.

4.6.2 Member Enumeration Documentation

4.6.2.1 enum **MagnetometerHMC5983::ExtendedRegister**

Enumerator

TEMPH

TEMPL

Definition at line 41 of file MagnetometerHMC5983.h.

4.6.2.2 enum **MagnetometerHMC5983::LowestPowerMode**

Lowest power mode.

Enumerator

DISABLE_LOWEST_POWER_MODE

ENABLE_LOWEST_POWER_MODE

Definition at line 156 of file MagnetometerHMC5983.h.

4.6.2.3 enum **MagnetometerHMC5983::SerialInterfaceMode**

SPI serial interface mode selection.

Enumerator

FOUR_WIRE

THREE_WIRE

Definition at line 164 of file MagnetometerHMC5983.h.

4.6.2.4 enum MagnetometerHMC5983::SpeedMode

Speed mode.

Enumerator

NORMAL_MODE

HIGH_SPEED_MODE

Definition at line 140 of file MagnetometerHMC5983.h.

4.6.2.5 enum MagnetometerHMC5983::TemperatureSensor

Temperature sensor.

Enumerator

DISABLE_TEMPERATURE_SENSOR

ENABLE_TEMPERATURE_SENSOR

Definition at line 148 of file MagnetometerHMC5983.h.

4.6.3 Constructor & Destructor Documentation

4.6.3.1 MagnetometerHMC5983::MagnetometerHMC5983 ( )

Definition at line 3 of file MagnetometerHMC5983.cpp.

4.6.4 Member Function Documentation

4.6.4.1 double MagnetometerHMC5983::getTemperature ( )

Gets the temperature measurement.

Temperature output in C is related to the temperature output register values as follows. Temperature = (MSB ∗ 2^8 + LSB) / (2^4 ∗ 8) + 25 in C

Returns

temperature in Celsius degrees.

Definition at line 22 of file MagnetometerHMC5983.cpp.

4.6.4.2 void MagnetometerHMC5983::setHighSpeedMode (unsigned char *speedMode*)

Set speed mode.

MR7 HS Set this pin to enable I²C High Speed mode, 3400 kHz.

Parameters

speedMode	SpeedMode option.

Definition at line 10 of file MagnetometerHMC5983.cpp.

4.6.4.3 void MagnetometerHMC5983::setLowestPowerMode (unsigned char *lowestPowerMode*)

Set Lowest power mode.

MR5 LP Lowest power mode. When set, ODR=0.75 Hz, and Averaging = 1.

Parameters

lowestPower↩ Mode	LowestPowerMode option.

Definition at line 14 of file MagnetometerHMC5983.cpp.

4.6.4.4 void MagnetometerHMC5983::setSerialInterfaceMode (unsigned char *serialInterfaceMode*)

Set serial interface mode selection.

SPI serial interface mode selection: 0 -> 4-wire SPI interface 1 -> 3-wire SPI interface

Parameters

serialInterface↩ Mode	SerialInterfaceMode option.

Definition at line 18 of file MagnetometerHMC5983.cpp.

4.6.4.5 void MagnetometerHMC5983::setTemperatureSensor (unsigned char *temperatureSensor*)

Sets temperature sensor.

CRA7 TS Set this bit to enable temperature sensor. Temperature sensor will be measured at each magnetic measurement. Enable Temperature sensor for automatic compensation of Sensitivity over temperature.

Parameters

temperature↩ Sensor	TemperatureSensor option.

Definition at line 6 of file MagnetometerHMC5983.cpp.

The documentation for this class was generated from the following files:

- MagnetometerHMC5983.h
- MagnetometerHMC5983.cpp

4.7 MagnetometerHMC5883L::MRbits Union Reference

```
#include <MagnetometerHMC5883L.h>
```

Public Attributes

- struct {
  unsigned char MD0:1
  unsigned char MD1:1
  unsigned char:6
  };

- struct {
  unsigned char MD:2
  unsigned char:6
  };

- unsigned char value

4.7.1 Detailed Description

Mode Register.

The mode register is an 8-bit register from which data can be read or to which data can be written. This register is used to select the operating mode of the device. MR0 through MR7 indicate bit locations, with MR denoting the bits that are in the mode register. MR7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. Mode register default is 0x01.

MR7 to MR2 0: Bit MR7 is set to 1 internally after each single-measurement operation. Set to 0 when configuring mode register.

MR1 to MR0 (MD1 to MD0): Mode Select Bits. These bits select the operation mode of this device.

MD1 MD0 -> Operating Mode 00 -> 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. 01 -> Single-Measurement Mode (Default). When single-measurement mode is selected, device performs a single measurement, sets RDY high and returned to idle mode. Mode register returns to idle mode bit values. The measurement remains in the data output register and RDY remains high until the data output register is read or another measurement is performed. 10 -> Idle Mode. Device is placed in idle mode. 11 -> Idle Mode. Device is placed in idle mode.

Definition at line 171 of file MagnetometerHMC5883L.h.

4.7.2 Member Data Documentation

4.7.2.1 struct { ... }

4.7.2.2 struct { ... }

4.7.2.3 unsigned MagnetometerHMC5883L::MRbits::char

Definition at line 176 of file MagnetometerHMC5883L.h.

4.7.2.4 unsigned char MagnetometerHMC5883L::MRbits::MD

Definition at line 179 of file MagnetometerHMC5883L.h.

4.7.2.5 unsigned char MagnetometerHMC5883L::MRbits::MD0

Definition at line 174 of file MagnetometerHMC5883L.h.

4.7.2.6 unsigned char MagnetometerHMC5883L::MRbits::MD1

Definition at line 175 of file MagnetometerHMC5883L.h.

4.7.2.7 unsigned char MagnetometerHMC5883L::MRbits::value

Definition at line 182 of file MagnetometerHMC5883L.h.

The documentation for this union was generated from the following file:

- MagnetometerHMC5883L.h

4.8 MagnetometerHMC5883L::SRbits Union Reference

```
#include <MagnetometerHMC5883L.h>
```

Public Attributes

- struct {
  unsigned char RDY:1

```
unsigned char LOCK:1
unsigned char:6
  };
```

- unsigned char value

4.8.1 Detailed Description

Data Output X Registers A and B.

The data output X registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel X. Data output X register A contains the MSB from the measurement result, and data output X register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DXRA0 through DXRA7 and DXRB0 through DXRB7 indicate bit locations, with DXRA and DXRB denoting the bits that are in the data output X registers. DXRA7 and DXRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit. In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made. Data Output Y Registers A and B

The data output Y registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel Y. Data output Y register A contains the MSB from the measurement result, and data output Y register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DYRA0 through DYRA7 and DYRB0 through DYRB7 indicate bit locations, with DYRA and DYRB denoting the bits that are in the data output Y registers. DYRA7 and DYRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made. Data Output Z Registers A and B

The data output Z registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel Z. Data output Z register A contains the MSB from the measurement result, and data output Z register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DZRA0 through DZRA7 and DZRB0 through DZRB7 indicate bit locations, with DZRA and DZRB denoting the bits that are in the data output Z registers. DZRA7 and DZRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made. Data Output Register Operation

When one or more of the output registers are read, new data cannot be placed in any of the output data registers until all six data output registers are read. This requirement also impacts DRDY and RDY, which cannot be cleared until new data is placed in all the output registers. Status Register

The status register is an 8-bit read-only register. This register is used to indicate device status. SR0 through SR7 indicate bit locations, with SR denoting the bits that are in the status register. SR7 denotes the first bit of the data stream.

SR7 to SR2 0: These bits are reserved.

SR1 (LOCK): Data output register lock. This bit is set when:

1. Some but not all for of the six data output registers have been read,
2. Mode register has been read. When this bit is set, the six data output registers are locked and any new data

will not be placed in these register until one of these conditions are met:

1. All six bytes have been read, 2. the mode register is changed,
2. The measurement configuration (CRA) is changed,
3. Power is reset.

SR0 (RDY): Ready Bit. Set when data is written to all six data registers. Cleared when device initiates a write to the data output registers and after one or more of the data output registers are written to. When RDY bit is clear it shall remain cleared for a 250 s. DRDY pin can be used as an alternative to the status register for monitoring the device for measurement data.

Definition at line 266 of file MagnetometerHMC5883L.h.

4.8.2 Member Data Documentation

4.8.2.1 struct { ... }

4.8.2.2 unsigned MagnetometerHMC5883L::SRbits::char

Definition at line 271 of file MagnetometerHMC5883L.h.

4.8.2.3 unsigned char MagnetometerHMC5883L::SRbits::LOCK

Definition at line 270 of file MagnetometerHMC5883L.h.

4.8.2.4 unsigned char MagnetometerHMC5883L::SRbits::RDY

Definition at line 269 of file MagnetometerHMC5883L.h.

4.8.2.5 unsigned char MagnetometerHMC5883L::SRbits::value

Definition at line 273 of file MagnetometerHMC5883L.h.

The documentation for this union was generated from the following file:

- MagnetometerHMC5883L.h

4.9 MagnetometerHMC5983::SRbits Union Reference

```
#include <MagnetometerHMC5983.h>
```

Public Attributes

- struct {
  unsigned char RDY:1
  unsigned char LOCK:1
  unsigned char:2
  unsigned char DOW:1
  };
- unsigned char value

4.9.1 Detailed Description

(Extended) Status Register

The status register is an 8-bit read-only register. This register is used to indicate device status. SR0 through SR7 indicate bit locations, with SR denoting the bits that are in the status register. SR7 denotes the first bit of the data stream.

SR7 to SR5 0: These bits are reserved.

SR4 (DOW): Data Over Written. Set when the measurement data are not read before the subsequent data measurements are posted to the output registers. This happens when master device skips reading one or more data samples. Bit is cleared at the beginning of a data read.

SR3 to SR2 0: These bits are reserved.

SR1 (LOCK): Data output register lock. This bit is set when:

1. Some but not all for of the six data output registers have been read,
2. Mode register has been read. When this bit is set, the six data output registers are locked and any new data will not be placed in these register until one of these conditions are met:

1. All six bytes have been read, 2. the mode register is changed,
2. The measurement configuration (CRA) is changed,
3. Power is reset.

SR0 (RDY): Ready Bit. Set when data is written to all six data registers. Cleared when device initiates a write to the data output registers and after one or more of the data output registers are written to. When RDY bit is clear it shall remain cleared for a 250 s. DRDY pin can be used as an alternative to the status register for monitoring the device for measurement data.

Definition at line 125 of file MagnetometerHMC5983.h.

4.9.2 Member Data Documentation

4.9.2.1 struct { ... }

4.9.2.2 unsigned MagnetometerHMC5983::SRbits::char

Definition at line 130 of file MagnetometerHMC5983.h.

4.9.2.3 unsigned char MagnetometerHMC5983::SRbits::DOW

Definition at line 131 of file MagnetometerHMC5983.h.

4.9.2.4 unsigned char MagnetometerHMC5983::SRbits::LOCK

Definition at line 129 of file MagnetometerHMC5983.h.

4.9.2.5 unsigned char MagnetometerHMC5983::SRbits::RDY

Definition at line 128 of file MagnetometerHMC5983.h.

4.9.2.6 unsigned char MagnetometerHMC5983::SRbits::value

Definition at line 134 of file MagnetometerHMC5983.h.

The documentation for this union was generated from the following file:

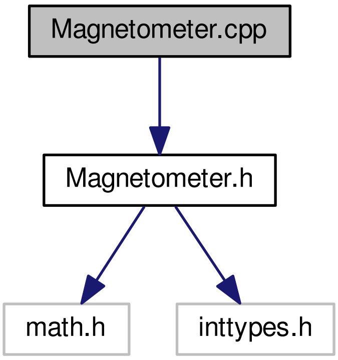
- MagnetometerHMC5983.h

5 File Documentation

5.1 Magnetometer.cpp File Reference

`#include "Magnetometer.h"`

Include dependency graph for Magnetometer.cpp:



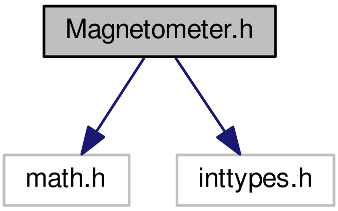
5.2 Magnetometer.cpp

```
#include "Magnetometer.h"

Magnetometer::~Magnetometer() {
}

double Magnetometer::radiansToDegrees(double radians) {
    return radians * 180.0 / M_PI;
}

double Magnetometer::computeVectorAngle(int16_t x, int16_t y) {
    double degrees = radiansToDegrees(-atan2(y, x));
    if (degrees < 0) {
        degrees += 360.0;
    }
    return degrees;
}
```

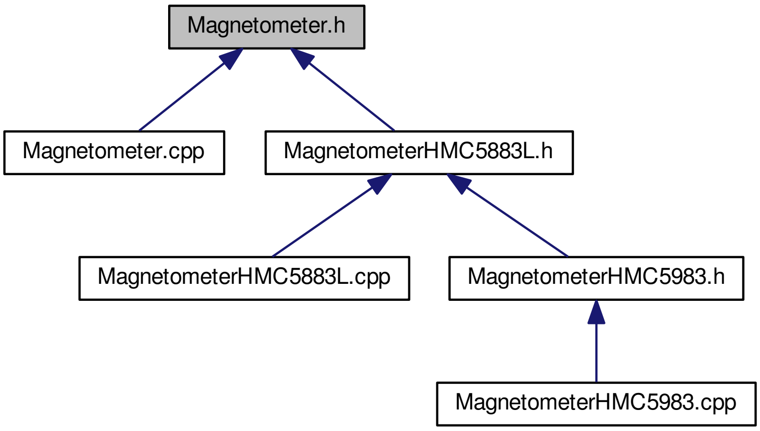
5.3 Magnetometer.h File Reference

```
#include <math.h>
#include <inttypes.h>
```

Include dependency graph for Magnetometer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Magnetometer

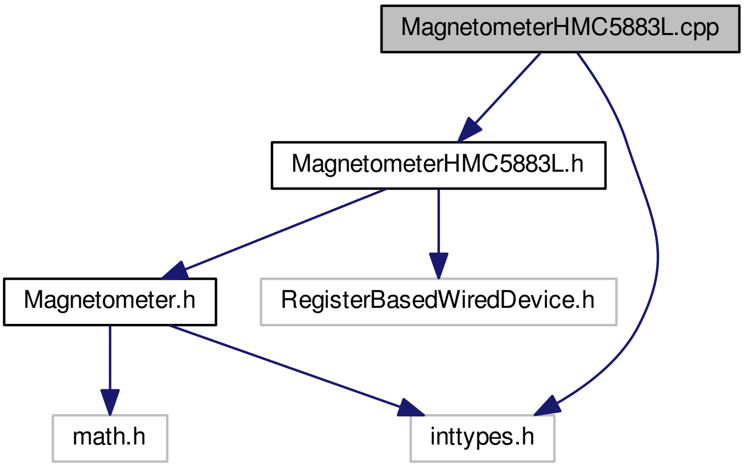
5.4 Magnetometer.h

```

#ifndef __ARDUINO_DRIVER_MAGNETOMETER_H__
#define __ARDUINO_DRIVER_MAGNETOMETER_H__ 1

#include <math.h>
#include <inttypes.h>

class Magnetometer {

public:

    virtual ~Magnetometer();

    virtual double getHeading() = 0;

    double inline radiansToDegrees(double radians);

    double computeVectorAngle(int16_t x, int16_t y);
};

#endif // __ARDUINO_DRIVER_MAGNETOMETER_H__
```

5.5 MagnetometerHMC5883L.cpp File Reference

```
#include "MagnetometerHMC5883L.h"
#include <inttypes.h>
```

Include dependency graph for MagnetometerHMC5883L.cpp:



5.6 MagnetometerHMC5883L.cpp

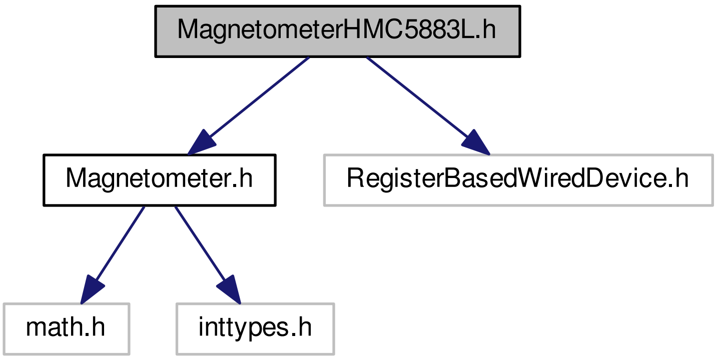
```
#include "MagnetometerHMC5883L.h"
#include <inttypes.h>

MagnetometerHMC5883L::MagnetometerHMC5883L()
        : RegisterBasedWiredDevice(MAGNETOMETER_HMC5883L_DEVICE_ADDRESS) {
}

MagnetometerHMC5883L::~MagnetometerHMC5883L() {
}

double MagnetometerHMC5883L::getHeading() {
    unsigned char buf[6];
    int16_t x = 0, y = 0;
    readSample(buf);
    x = (buf[0] << 8) | buf[1];
    y = (buf[2] << 8) | buf[3];
    return computeVectorAngle(x, y);
}

void MagnetometerHMC5883L::setOperatingMode(unsigned char operatingMode) {
    writeRegister(MagnetometerHMC5883L::MR, operatingMode & MAGNETOMETER_HMC5883L_MR_MASK);
}

void MagnetometerHMC5883L::setSamplesAveraged(unsigned char samplesAveraged) {
    configureRegisterBits(CRA, MAGNETOMETER_HMC5883L_CRA_MS_MASK, samplesAveraged << 5);
}

```

```
void MagnetometerHMC5883L::setDataOutputRate(unsigned char dataOutputRate) {
    configureRegisterBits(CRA, MAGNETOMETER_HMC5883L_CRA_DO_MASK, dataOutputRate << 2);
}

void MagnetometerHMC5883L::setMeasurementMode(unsigned char measurementMode) {
    configureRegisterBits(CRA, MAGNETOMETER_HMC5883L_CRA_MA_MASK, measurementMode);
}

void MagnetometerHMC5883L::setGain(unsigned char gain) {
    MagnetometerHMC5883L::CRBbits crb = {0};
    crb.GN = gain;
    writeRegister(MagnetometerHMC5883L::CRB, crb.value);
}

MagnetometerHMC5883L::SRbits MagnetometerHMC5883L::getStatusRegister() {
    MagnetometerHMC5883L::SRbits sr = {0};
    sr.value = readRegister(SR);
    return sr;
}

void MagnetometerHMC5883L::readSample(unsigned char buf[6]) {
    readRegisterBlock(DXRA, buf, 0x06);
}

```

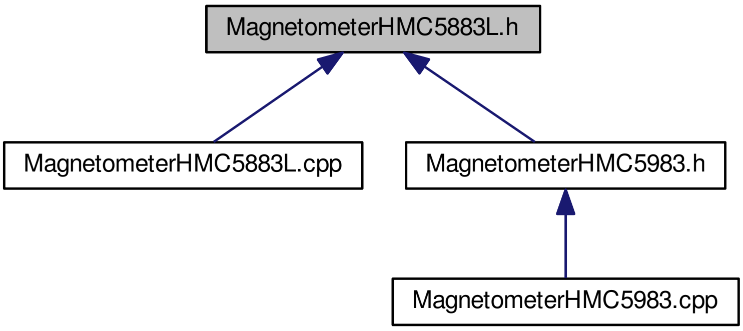
5.7 MagnetometerHMC5883L.h File Reference

```
#include <Magnetometer.h>
#include <RegisterBasedWiredDevice.h>
```

Include dependency graph for MagnetometerHMC5883L.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MagnetometerHMC5883L
- union MagnetometerHMC5883L::CRAbits
- union MagnetometerHMC5883L::CRBbits
- union MagnetometerHMC5883L::MRbits
- union MagnetometerHMC5883L::SRbits

Macros

- #define MAGNETOMETER_HMC5883L_DEVICE_ADDRESS 0x1e
- #define MAGNETOMETER_HMC5883L_CRA_MS_MASK 0x60
- #define MAGNETOMETER_HMC5883L_CRA_DO_MASK 0x1c
- #define MAGNETOMETER_HMC5883L_CRA_MA_MASK 0x03
- #define MAGNETOMETER_HMC5883L_MR_MASK 0x03

5.7.1 Macro Definition Documentation

5.7.1.1 #define MAGNETOMETER_HMC5883L_CRA_DO_MASK 0x1c

Definition at line 18 of file MagnetometerHMC5883L.h.

5.7.1.2 #define MAGNETOMETER_HMC5883L_CRA_MA_MASK 0x03

Definition at line 19 of file MagnetometerHMC5883L.h.

5.7.1.3 #define MAGNETOMETER_HMC5883L_CRA_MS_MASK 0x60

Definition at line 17 of file MagnetometerHMC5883L.h.

5.7.1.4 #define MAGNETOMETER_HMC5883L_DEVICE_ADDRESS 0x1e

Arduino - MagnetometerHMC5883L driver.

Concrete implementation of HMC5883L magnetometer.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 15 of file MagnetometerHMC5883L.h.

5.7.1.5 #define MAGNETOMETER_HMC5883L_MR_MASK 0x03

Definition at line 21 of file MagnetometerHMC5883L.h.

5.8 MagnetometerHMC5883L.h

```
00001
00009 #ifndef __ARDUINO_DRIVER_MAGNETOMETER_HMC5883L_H__
00010 #define __ARDUINO_DRIVER_MAGNETOMETER_HMC5883L_H__ 1
00011
00012 #include <Magnetometer.h>
00013 #include <RegisterBasedWiredDevice.h>
00014
00015 #define MAGNETOMETER_HMC5883L_DEVICE_ADDRESS    0x1e
00016
00017 #define MAGNETOMETER_HMC5883L_CRA_MS_MASK       0x60
00018 #define MAGNETOMETER_HMC5883L_CRA_DO_MASK       0x1c
00019 #define MAGNETOMETER_HMC5883L_CRA_MA_MASK       0x03
00020
00021 #define MAGNETOMETER_HMC5883L_MR_MASK           0x03
00022
00035 class MagnetometerHMC5883L: public Magnetometer, public
      RegisterBasedWiredDevice {
00036
00037 public:
00038
00082     union CRAbits {
00083
00084         struct {
00085             unsigned char MS0 :1;
00086             unsigned char MS1 :1;
00087             unsigned char DO0 :1;
00088             unsigned char DO1 :1;
00089             unsigned char DO2 :1;
00090             unsigned char MA0 :1;
00091             unsigned char MA1 :1;
00092             unsigned char :1;
00093         };
00094         struct {
00095             unsigned char MS :2;
00096             unsigned char DO :3;
00097             unsigned char MA :2;
00098             unsigned char :1;
00099         };
00100         unsigned char value;
00101     };
00102
00129     union CRBbits {
00130
00131         struct {
00132             unsigned char :5;
00133             unsigned char GN0 :1;
00134             unsigned char GN1 :1;
00135             unsigned char GN2 :1;
00136         };
00137         struct {
00138             unsigned char :5;
00139             unsigned char GN :3;
00140         };
00141         unsigned char value;
00142     };
00143
00171     union MRbits {
00172
00173         struct {
00174             unsigned char MD0 :1;
00175             unsigned char MD1 :1;
00176             unsigned char :6;
00177         };
00178         struct {
00179             unsigned char MD :2;
00180             unsigned char :6;
00181         };
00182         unsigned char value;
00183     };
00184
```

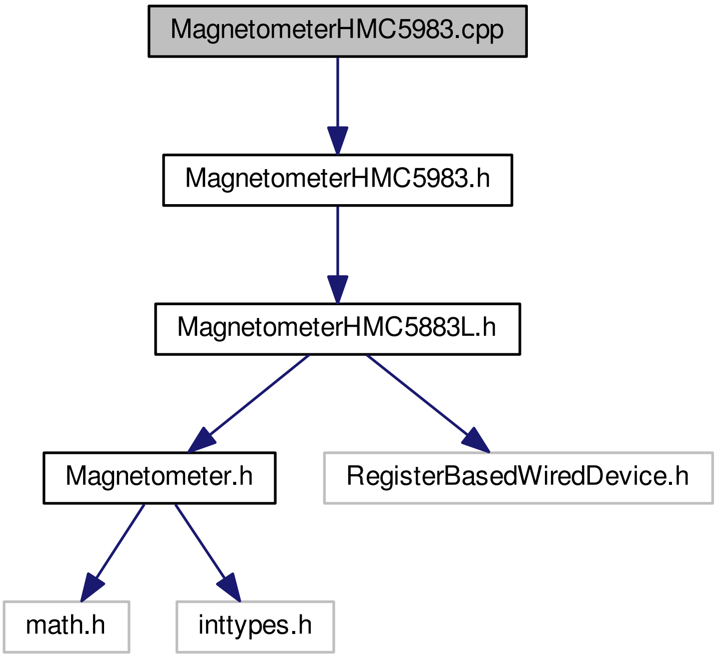
```
union SRbits {

    struct {
        unsigned char RDY :1;
        unsigned char LOCK :1;
        unsigned char :6;
    };
    unsigned char value;
};

/*
 * 00 Configuration Register A Read/Write
 * 01 Configuration Register B Read/Write
 * 02 Mode Register Read/Write
 * 03 Data Output X MSB Register Read
 * 04 Data Output X LSB Register Read
 * 05 Data Output Z MSB Register Read
 * 06 Data Output Z LSB Register Read
 * 07 Data Output Y MSB Register Read
 * 08 Data Output Y LSB Register Read
 * 09 Status Register Read
 * 10 Identification Register A Read
 * 11 Identification Register B Read
 * 12 Identification Register C Read
 */
enum Register {
    CRA = 0x00,
    CRB = 0x01,
    MR = 0x02,
    DXRA = 0x03,
    DXRB = 0x04,
    DYRA = 0x05,
    DYRB = 0x06,
    DZRA = 0x07,
    DZRB = 0x08,
    SR = 0x09,
    IDA = 0x0a,
    IDB = 0x0b,
    IDC = 0x0c
};

enum OperatingMode {
    IDLE_MODE = 0x00,
    CONTINUOUS_MEASUREMENT_MODE = 0x01,
    SINGLE_MEASUREMENT_MODE = 0x02
};

enum SamplesAveraged {
    SA_1 = 0x00,
    SA_2 = 0x01,
    SA_4 = 0x02,
    SA_8 = 0x03,
};

enum DataOutputRate {
    DAR_0_75 = 0x00,
    DAR_1_5 = 0x01,
    DAR_3 = 0x02,
    DAR_7_5 = 0x03,
    DAR_15 = 0x04,
    DAR_30 = 0x05,
    DAR_75 = 0x06
};

enum MeasurementMode {
    NORMAL_MEASUREMENT = 0x00,
    POSITIVE_BIAS = 0x01,
    NEGATIVE_BIAS = 0x02
};

enum Gain {
    GAIN_0_88_GA = 0x00,
    GAIN_1_3_GA = 0x01,
    GAIN_1_9_GA = 0x02,
    GAIN_2_5_GA = 0x03,
    GAIN_4_0_GA = 0x04,
    GAIN_4_7_GA = 0x05,
    GAIN_5_6_GA = 0x06,
    GAIN_8_1_GA = 0x07
};

MagnetometerHMC5883L();

void setOperatingMode(unsigned char operatingMode);

virtual ~MagnetometerHMC5883L();

```

```
    void setSamplesAveraged(unsigned char samplesAveraged);

    void setDataOutputRate(unsigned char dataOutputRate);

    void setMeasurementMode(unsigned char measurementMode);

    void setGain(unsigned char gain);

    SRbits getStatusRegister();

    void readSample(unsigned char buf[6]);

    double getHeading();
};

#endif // __ARDUINO_DRIVER_MAGNETOMETER_HMC5883L_H__
```

5.9 MagnetometerHMC5983.cpp File Reference

`#include "MagnetometerHMC5983.h"`

Include dependency graph for MagnetometerHMC5983.cpp:



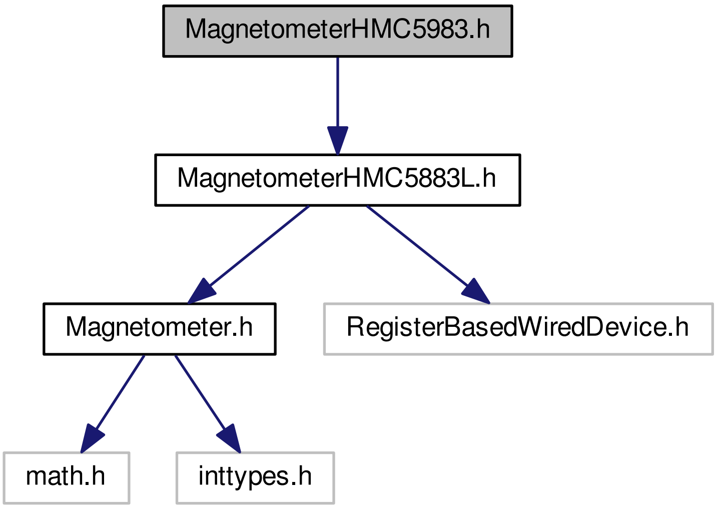
5.10 MagnetometerHMC5983.cpp

```
#include "MagnetometerHMC5983.h"

MagnetometerHMC5983::MagnetometerHMC5983() {
}

void MagnetometerHMC5983::setTemperatureSensor(unsigned char temperatureSensor) {
    configureRegisterBits(CRA, 0x80, temperatureSensor << 7);
}

void MagnetometerHMC5983::setHighSpeedMode(unsigned char speedMode) {
    configureRegisterBits(MR, MAGNETOMETER_HMC5983_MR_HS_MASK, speedMode << 7);
}

void MagnetometerHMC5983::setLowestPowerMode(unsigned char
lowestPowerMode) {
    configureRegisterBits(MR, MAGNETOMETER_HMC5983_MR_LP_MASK,
lowestPowerMode << 5);
}

void MagnetometerHMC5983::setSerialInterfaceMode(unsigned char
serialInterfaceMode) {
    configureRegisterBits(MR, MAGNETOMETER_HMC5983_MR_SIM_MASK,
serialInterfaceMode << 3);
}

double MagnetometerHMC5983::getTemperature() {
    uint16_t raw = 0;
    readRegisterBlock(TEMPH, (unsigned char *) &raw, 2);
    return (raw / (double) 0x80) + 0x19;
}
```

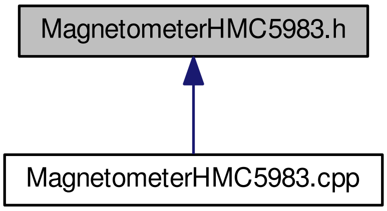
5.11 MagnetometerHMC5983.h File Reference

```
#include <MagnetometerHMC5883L.h>
```

Include dependency graph for MagnetometerHMC5983.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MagnetometerHMC5983
- union MagnetometerHMC5983::EMRbits
- union MagnetometerHMC5983::SRbits

Macros

- #define MAGNETOMETER_HMC5983_MR_HS_MASK 0x80
- #define MAGNETOMETER_HMC5983_MR_LP_MASK 0x20
- #define MAGNETOMETER_HMC5983_MR_SIM_MASK 0x04
- #define MAGNETOMETER_HMC5983_SR_DOW_MASK 0x10

5.11.1 Macro Definition Documentation

5.11.1.1 #define MAGNETOMETER_HMC5983_MR_HS_MASK 0x80

Arduino - MagnetometerHMC5983 driver.

Concrete implementation of HMC5983 magnetometer witch replaces HMC5883L one.

Author

Dalmir da Silva `dalmirdasilva@gmail.com`

Definition at line 14 of file MagnetometerHMC5983.h.

5.11.1.2 #define MAGNETOMETER_HMC5983_MR_LP_MASK 0x20

Definition at line 15 of file MagnetometerHMC5983.h.

5.11.1.3 #define MAGNETOMETER_HMC5983_MR_SIM_MASK 0x04

Definition at line 16 of file MagnetometerHMC5983.h.

5.11.1.4 #define MAGNETOMETER_HMC5983_SR_DOW_MASK 0x10

Definition at line 17 of file MagnetometerHMC5983.h.

5.12 MagnetometerHMC5983.h

```

#ifndef __ARDUINO_DRIVER_MAGNETOMETER_HMC5983_H__
#define __ARDUINO_DRIVER_MAGNETOMETER_HMC5983_H__ 1

#include <MagnetometerHMC5883L.h>

#define MAGNETOMETER_HMC5983_MR_HS_MASK         0x80
#define MAGNETOMETER_HMC5983_MR_LP_MASK         0x20
#define MAGNETOMETER_HMC5983_MR_SIM_MASK        0x04
#define MAGNETOMETER_HMC5983_SR_DOW_MASK        0x10

class MagnetometerHMC5983: public MagnetometerHMC5883L {

public:

    enum ExtendedRegister {
        TEMPH = 0x31,
        TEMPL = 0x32
    };

    union EMRbits {

        struct {
            unsigned char MD0 :1;
            unsigned char MD1 :1;
            unsigned char SIM :1;
            unsigned char :2;
            unsigned char LP :1;
            unsigned char :1;
            unsigned char HS :1;
        };
        struct {
            unsigned char MD :2;
            unsigned char :6;
        };
        unsigned char value;
    };

    union SRbits {

        struct {
            unsigned char RDY :1;
            unsigned char LOCK :1;
            unsigned char :2;
            unsigned char DOW :1;
            unsigned char :3;
        };
        unsigned char value;
    };

    enum SpeedMode {
        NORMAL_MODE = 0x00,
        HIGH_SPEED_MODE = 0x01
    };

    enum TemperatureSensor {
        DISABLE_TEMPERATURE_SENSOR = 0x00,
        ENABLE_TEMPERATURE_SENSOR = 0x01
    };

    enum LowestPowerMode {
        DISABLE_LOWEST_POWER_MODE = 0X00,
        ENABLE_LOWEST_POWER_MODE = 0X01
    };

    enum SerialInterfaceMode {
        FOUR_WIRE = 0X00,
        THREE_WIRE = 0X01
    };

    MagnetometerHMC5983();

    void setTemperatureSensor(unsigned char temperatureSensor);

    void setHighSpeedMode(unsigned char speedMode);

    void setLowestPowerMode(unsigned char lowestPowerMode);

    void setSerialInterfaceMode(unsigned char serialInterfaceMode);

    double getTemperature();
};

#endif // __ARDUINO_DRIVER_MAGNETOMETER_HMC5983_H__
```