RT-CMPS03 Compass Module

General Description

RT-CMPS03 module is a 2-Axis magnetic sensing module that identifies the angle of rotation in relation to the Earth's magnetic field. This module is designed specifically for navigation in robots and it allows you to know the direction of your robot into the North according to the calibration you did. RT compass module consists of a 2-Axi magnetometer sensor which can measure magnetic field with a full range of ±2 gausses and a sensitivity of 512 counts/gauss at 3.0 V and 25 °C. Signal processing and I2C interface in these sensors allowing them to be connected directly to a microprocessor eliminating the need for A/D converter. In addition to the I2C interface, the amount of module rotation is also obtained through Pulse width modulation.

Features

- 2-Axis magnetometer sensor
- Low voltage operation
- Low power consumption
- Small size – 32*34 mm
- Supporting I2C interface
- Supporting PWM
- Resolution – 0.1 degree
- Accuracy – 1% (is dependent on system design , calibration and compensation algorithm)

Absolute Maximum Ratings

Parameter	Value	Unit
Field Range	+2	gauss
Exposed field	10000	gauss
Supply voltage	+5	V
Supply Current	25	mA
SCL clock frequency	400	KHz
Operating temperature	+85	℃

Specifications

Parameter	Conditions	Min	Typ	Max	Units
Field Range	Total applied field	-2		+2	gauss
Supply voltage		3.3	5.0	5.5	V
Supply current	50 measurements/second		10	25	mA
Operating temperature		-40		+85	°C
Accuracy			1.0		%
Resolution			0.1		deg
Dimension			32*34		mm
Weight				5	gr

Pin Description

Pin number	Pin name	Description
1	VCC	Supply voltage – 3.3 to 5 v
2	SCL	I2C clock
3	SDA	I2C data
4	PWM	PWM signal representing the compass bearing.
5	Calibrating	Representing the status of calibration.
6	Calibrate	Used in calibration process.
7	No connect	Currently not used.
8	No connect	Currently not used.
9	GND	Ground – 0 v

Pin 1: The Compass module requires a +5v power supply at a nominal 25mA.

Pin 2,3: There are two ways of getting the bearing from the module. One of them is an I2C interface that is provided on pin 2,3. These pins can be used to get a direct readout of the bearing. If the I2C interface is not used, then these pins should pull up via a couple of 4.7K resistors.

Pin 4: Another way of getting the bearing from the module is a PWM signal which is available on this pin. The PWM signal is a pulse width modulated signal with the positive width of the pulse representing the angle. The pulse width varies from 1mS (0°) to 36.99mS (359.9°) – in other words 100uS/° with a +1mS offset. The signal goes low for 65mS between pulses, so the cycle time is 65mS + the pulse width - ie. 66ms-102ms. The pulse is generated by a 16-bit timer in the processor giving a 1uS resolution, however I would not recommend measuring this to anything better than 0.1° (10uS).

Pin 5: This pin is used to indicate calibration is in progress. An LED is used that will be turn on during the calibration process.

Pin 6: One of the ways to calibrate the compass module will be done by this pin that is explained in the following.

Pin 7,8: These pins are currently unused.

Pin 9: This is the 0v power supply.

Registers List

Register number	Description
0	Software Revision Number
1	Compass Bearing as a byte, 0-255 for a full circle
2,3	Compass Bearing as a word, 0-3599 for a full circle, representing 0-359.9 degrees
4,5	X-Axis sensor processed data, 16 bit signed word
6,7	Y-Axis sensor processed data, 16 bit signed word
8,9	X-Axis sensor raw data, 16 bit signed word
10,11	Y-Axis sensor raw data, 16 bit signed word
12	Unlock code1 – changing I2C address or restoring factory calibration
13	Unlock code2 – changing I2C address or restoring factory calibration
14	Unlock code3 – changing I2C address or restoring factory calibration
15	Calibration Register

Reg 0: The software revision number is shown with this register that is 19 at the time of writing.

Reg 1: The value of compass bearing as a byte is shown by this register that is converted to a 0-255 value. This may be easier for some applications than 0-360 which requires two bytes.

Reg 2,3: These registers use for those who require better resolution that are a 16-bit unsigned integer in the range 0-3599 and represents 0-359.9°.

Reg 4,5: Each axis consists of two bytes, which are 16 bits of data. The raw data of the sensor at the X-Axis direction stores by these registers.

Reg 6,7: The raw data of the sensor at the Y-axis direction stores by these registers.

Reg 8,9,10,11: These registers contain the raw data of the sensor. These are the signals coming directly from the sensor and are starting point for all the internal calculation which produces the compass bearing. The raw data of the sensor at the X-axis direction stores by registers 8 and 9. Also, the raw data of the sensor at the Y-axis direction stores by registers 10 and 11.

Reg 12,13,14: These command registers are for writing the unlock codes for changing the I2C address or restoring factory calibration.

Reg 15: The last register uses for calibration.

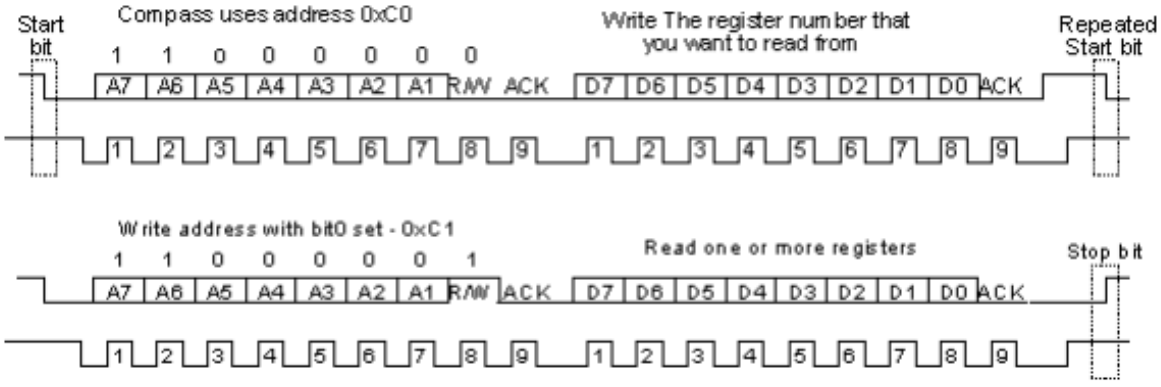
Setup

1- Data Transfer

I2C communication protocol with the RT-CMPS03 module is the same as popular eeprom's such as the 24C04. RT-CMPS03 module supports I2C communication protocol, which is designed to work at up to the standard clock speed (SCL) of 400MHz.

A data transfer is 8 bits long and each byte has to be followed by an “Acknowledge” bit. A data transfer at the I2C communication, started with a “START” condition and ended with a “STOP” condition. A “START” condition is defined by sending a start bit, after that the module address (0XC0) with the read/write bit low. The slave device being called should respond by an acknowledge signal, which is pulling SDA line LOW. Then the register number you wish to read will be send. This is followed by a repeated start and the module address again with the read/write bit high (0XC1) and an acknowledge bit. Now you can read one or more registers.

The RT-CMPS03 module has a 16-byte array of registers, which they are described before. Finally, “STOP” condition is defined by sending a stop bit.



2- Communication with a 2-Axis magnetometer IC using I2C protocol

Note: This part is for getting data directly from magnetometer IC on the board and not the compass module . So it is not recommended to do this for reading compass module .for reading compass module read previus part(Data Transfer).

- START followed by calling a specific slave device (0x30) to write through master device. When an acknowledge signal is received by master device, it sends two bytes to slave device. First of all, master device sends "[00000000]" as the target address to write in to. At the end, 2-Axis magnetometer device should acknowledge.
- After that, in order to read sensor signal, master device should write to internal magnetometer device memory the code "[00000001]" that this action also serves as a "wake-up" call. At the end of write operation, a stop bit will be send.
- Now at least 5ms wait should be given to magnetometer device to finish receiving data and return a valid output.
- Again, master device sends a START command that followed by slave device address (0x30) to write through master device. At the end an acknowledge should be send by slave device.
- In order to read from internal memory, master device writes a [00000000] as the starting address. Since [00000000] is the address of internal control register, reading from this address can serve as a verification operation to confirm the write command has been successful. Notice that the starting address can be any of 5 addresses. For example, user can start read from address [00000001], which is X channel MSB.
- Now, it's time to read 5 bytes from slave device. So master device calls slave device address with a read and then first addressed memory data appears on SDA line. Master device should send acknowledge at the end.
- The last step continues and next byte of internal memory appears on SDA line (MSB of X channel). Automatically the internal memory address pointer moves to the next byte and at the end, acknowledge will be send by master device.
- Similarly, LSB of X channel, MSB of Y channel and LSB of Y channel will be received.
- Communication will be ended by not sending acknowledge and sending a STOP command.

3- Communication with the RT-CMPS03 module using I2C protocol

- To communicate with the module through I2C protocol, first the transmission rate between master and slave should be specified in the range of 10-400khz. For example with "i2c_master_init (clock_value)" function of i2c library

Example:

```
i2c_master_init (400);        //set the i2c master clock to 400 khz
```

- Now to connect the module and transmit data with it, you need to know the address of module (0x60). So, for example “i2c_master_trans(slave_address, data_snd_pt ,send_num,data_rcv_pt,rcv_num)” function can be used to send data and recive specified byte of response data . this fuction can be defined by 5 arguments. Arguments of this function are the address of module, the pointer to data send array, the number of bytes to send to slave, the pointer to array for receive data from the slave and the number of bytes of the data which is receive from the slave, respectively.

Exmple :

```
// below sodo code is a sample to read register 1 and 2 of RT-CMPS03 module using I2C protocol

Uint8_t snd_data[5],rcv_data[5],snd_num,rcv_num;

i2c_master_init (400);        //set the i2c master clock to 400 khz

snd_data[0]=1;

i2c_master_trans(0x60, snd_data,1, rcv_data,2);

// now rcv_data[0] contain register1 of compass value and rcv_data[1] contain register2 of compass value
```

Calibration

RT-CMPS03 module calibration process is easy and it can be done by two methods:

Note: Do not attempt this until you have your compass working ! Especially if you are using the I2C interface - get that fully working first. Calibration only needs to be done once - the calibration data is stored in EEPROM on the Mega8 chip. You do not need to re-calibrate every time the module is powered up.

Pin method

1- First of all, place the module horizontally on a flat surface.
2- Then, press the push button key on the module for a long time (at least 3s). There is an LED on the module that it turns on, indicating the start of the calibration process.

3- At this step, it is enough to rotate the module several times slowly in about 15 secends. Note that after turning on the LED, you have 15 secends time to do this step.
4- After that , the LED will turn off and the calibration process will end.
5- The Campus module is now calibrated and ready to use.

I2C Method

To calibrate using the I2C bus, you can write 0xFF to register 15.

-At this time the LED on the board will be turned on indicat device going to calibration mode .

-At this step, it is enough to rotate the module several times slowly in about 15 secends. Note that after turning on the LED, you have 15 secends time to do this step.

-After that , the LED will turn off and the calibration process will end.

-The Campus module is now calibrated and ready to use.

Sodo code Example

```
// below sodo code is a sample to calibrating compass module using I2c
Uint8_t snd_data[5],rcv_data[5],snd_num,rcv_num;
i2c_master_init (400);        //set the i2c master clock to 400 khz
snd_data[0]=15;
snd_data[1]=255;
i2c_master_trans(0x60, snd_data,2, 0,0);
// now device is ready to calibrated
```

Restoring factory calibration

It is possible to restore the factory calibration settings. It will be done by writing unlock codes to registers 12, 13, 14 and 15. So 0x55, 0x5A, 0xA5 and 0xF2 unlock codes should be written in order to registers 12, 13, 14 and 15.

Sodo code example

```
// below sodo code is a sample for restoring factory calibration
Uint8_t snd_data[5],rcv_data[5],snd_num,rcv_num;
i2c_master_init (400);        //set the i2c master clock to 400 khz
snd_data[0]=12;
```

```
snd_data[1]=0x55;
snd_data[2]=0x5a;
snd_data[3]=0xa5;
snd_data[4]=0xf2;
i2c_master_trans(0x60, snd_data,5, 0,0);
// now device restored to factory calibration setting
```

Changing the I2C address from factory default of 0xC0

With the Rev 19, it is possible to change the I2C address to any of 8 addresses 0xC0, 0xC2, 0xC4, 0xC6, 0xC8, 0xCA, 0xCC or 0xCE. What only you should do is that writing unlock codes to registers 12, 13 and 14 and the new address to register 15.

For example, changing the I2C address to 0xC2 will be done by writing 0xA0, 0xAA, 0xA5 and 0xC2 unlock codes registers 12, 13, 14 and 15 respectively.

Sodo code Example

```
// below sodo code is a sample to changing i2c address of compass module to 0xc2
Uint8_t snd_data[5],rcv_data[5],snd_num,rcv_num;
i2c_master_init (400);        //set the i2c master clock to 400 khz
snd_data[0]=12;
snd_data[1]=0xa0;
snd_data[2]=0xaa;
snd_data[3]=0xa5;
snd_data[4]=0xc2;
i2c_master_trans(0x60, snd_data,5, 0,0);
// now device address changed to 0xc2 (0x62)
```

Mechanical Dimensions