HiTechnic NXT Compass Sensor

Disclaimer: Much hardware registry information was copied from HiTechnic's manual. For more hardware specification, you should consult HitTechnic manuals.

Introduction

The NXT Compass Sensor contains a digital magnetic compass that measures the earth's magnetic field and calculates a heading angle. The Compass Sensor connects to an NXT sensor port using a standard NXT wire and uses the digital I2C communications protocol. The current heading is calculated to the nearest 1° and refreshed 100 times per second.

To test your new sensor:

- Compile the Compass code from the system:
  - File → Open Sample Program → NXT → "Try Me Program Source" → Compass.c
- Download to your nxt
- Go to "Try Me" Menu at the NXT, select "Compass".

Note: Hitechnic compass does not do tilt compensation. Therefore, your sensor must be stabilized on a horizontal platform.

To access the compass data

Valid data range: 0 (North) up to 359 (clockwise)

Sensor Type: sensorI2CHiTechnicCompass (for cooked mode) or sensorI2CCustom (for raw mode)

To initialize:

```
e.g.
SensorType[S1] = sensorI2CTechnicCompass;  // to initialize the port to be compass
sensor
SensorValue[S1]; //to obtain the sensor data
```

For HiTechnic Compass Sensor, RobotC provides driver to provide normalized compass value for you so that you do not need to do the I2C interface work. This is why you can just issue these simple APIs.

However, if you are interested in doing your own normalization, you will need to know the compass sensor register configuration. More information is at the following "WRITE YOUR OWN I2C LEVEL INTERFACE".

WRITE YOUR OWN I2C LEVEL INTERFACE

Sensor Register Configuration

However, if you do want to interface with another 3rd party compass sensor like this compass sensor, 2 things you must need to know:

1) The sensor MUST be a I2C sensor.
2) You need to know I2C register configuration. Consult the manufacturer's specification.

For your information, the following is the I2C register configuration from Hitechnic compass sensor

I2C register configuration for HiTechnic Compass Sensor			
Address	**Type**	**Contents**	
2H	Chars	I2C device address	
0H	Chars	Measurement Mode	Mode control field may be set 0x43 to start measurement or stop calibration mode.
41H	Byte	Mode control	
42H	Byte	Heading (two degree heading)	***
43H	Byte	Calibration Mode or Heading one degree adder	Mode control field may be set 0x43 to start calibrate mode. locn. 0x41 will be set to 2 if calibration fails.
44, 45H	Word	Heading (low byte, high byte)	

*** The heading is obtained by reading location 0x42 to obtain the two degree heading and 0x43 to obtain the one degree adder. The heading can then be computed as

*Heading = (two degree heading *2) + one degree adder.*
Or
Heading = (two degree heading <<1) | one degree adder

If the sign is a problem, then the conventional 16 bit value can be obtained from 0x44 (low byte) and 0x45 (high byte).

Protocol used for calibration

```
ubyte 2CMsg[5], reply[5];
SensorType[S1] = sensorI2CCustom;
memset(i2CMsg, 0, sizeof(i2CMsg));
memset(i2CMsg, 0, sizeof(reply));

i2CMsg[0] = 3;          // Number of bytes in I2C command
i2CMsg[1] = 0x02;       // I2C address of compass sensor
i2CMsg[2] = 0x41;       // control mode
i2CMsg[3] = 0x43;       // calibration mode
```

According to the 3rd party driver code suggestion, to calibrate, robot needs to spin >=540 clockwise and counter-clockwise with minimum 20 seconds duration for each direction.

Protocol used for stopping calibration and start measurement mode

```
ubyte 2CMsg[5], reply[5];
SensorType[S1] = sensorI2CCustom;
memset(i2CMsg, 0, sizeof(i2CMsg));
memset(i2CMsg, 0, sizeof(reply));

i2CMsg[0] = 3;          // Number of bytes in I2C command
i2CMsg[1] = 0x02;       // I2C address of compass sensor
i2CMsg[2] = 0x41;       // control mode
i2CMsg[3] = 0;          // stop calibration mode
```

EXERCISES

1. Download the calibrate.c from the system samples. Play with it to detect like a compass. Note that the zip file will include "calibrate.c", and a couple of other shared library files. You will need them too.
2. Download the csSimple.c from the system samples. Play with it to detect like a compass.
3. *csGoDir.c* : your controller only terminates when it faces west no matter which direction it faces at the startup time.
4. *csReturn.c*: your controller only terminates when it returns back to the direction where it faces at the startup time.
5. *calibrateCompass.c* : calibrate your compass

Now, you should make the modification yourselves:

1. ***csGoDir.c*** : add in motor navigation code to make your robot turn to west no matter which direction it faces at the startup time.
2. ***csReturn.c:*** add in motor navigation code to make your robot turn back to the direction where it faces at the startup time.