ZBasic Application Note

AN-211 Navigating the CMPS03 Magnetic Compass

Introduction

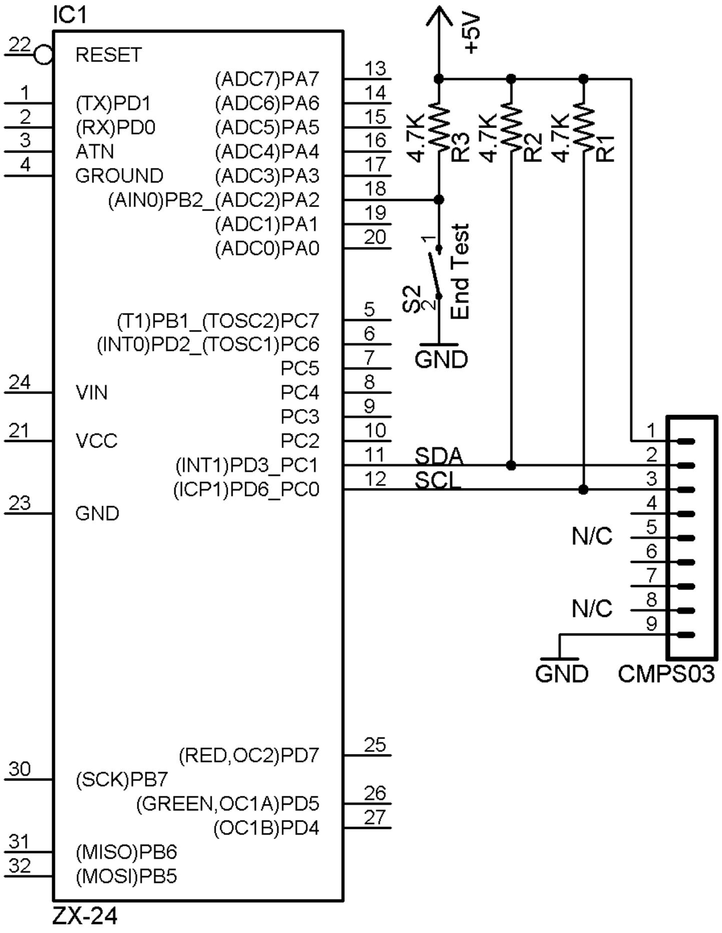
This application note describes how to connect and use the Robot Electronics CMPS03 Magnetic Compass (see http://www.robot-electronics.co.uk/shop/Compass_CMPS032004.htm). This device detects the local magnetic field (in the XY plane) and reports a compass heading from 0 to 360 degrees. On the CMPS03 circuit board are two Philips KMZ51 magnetic field sensors (http://www.semiconductors.philips.com/pip/KMZ51.html), a LMC6032 op-amp and a PIC microprocessor than handles all the calculations and provides an I2C or PWM interface.

Compass modules are typically used in robot applications. The compass heading returned by the module is related to Magnetic North rather than True North. Note also that the Earth's magnetic field is heavily influenced by buildings and nearby magnetic objects so the results from this sensor should always be correlated with other sensor input.

Hardware Hookup

For the purposes of this application note, the ZX chip is connected to the CMPS03 Magnetic Compass using the I2C serial protocol. The I2C SDA and SCL pins are connected to the ZX hardware-based I2C on the ZX-24 pins 11 and 12 respectively. For a ZX-40 these are pins 23 and 22, and for a ZX-44 the corresponding pins are 20 and 19. Don't forget the pullup resistors on the SDA and SCL pins – any value between 1.8K and 6.8K should be sufficient.

Note that with a ZX-24, the standard SDA/SCL pins overlap with interrupt 1 and input capture. The other alternative of using the software-based I2C "uses up" the Timer1 resource. This resource issue is much less of a problem with ZX-40 and ZX-44.



Published January 2006

Software

This application note also comes with some ZBasic software. The file CMPS03.bas is an interface module for the CMPS03 Magnetic Compass and the file AN211.bas is a test program for the CMPS03.bas module. The public interface implemented by the CMPS03.bas module consists of some public routines named `InitCMPS03()`, `TermCMPS03()`, `GetCMPS03Bearing()`, `GetCMPS03IntegerBearing()`, `GetCMPS03Version()`, and `CalibrateCMPS03()`.

When driven at the highest ZBasic speed for I2C (410 KHz), the device can return up to 3700 compass readings per second. This is more than sufficient for any application and does not significantly impact the timing for any kind of robot control loop. The function `GetCMPS03Bearing()` returns the compass heading as a floating point number and is implemented as shown below. There is also a version named `GetCMPS03IntegerBearing()` that returns the bearing as a fixed point integer where the least significant digit is tenths of degrees.

```
Public Function GetCMPS03Bearing() as Single
   GetCMPS03Bearing = CSng(GetCMPS03IntValue(REG_BEARING))/10.0
End Function
```

The internal function `GetCMP03IntValue()` does all of the work and is coded below. This internal function is also used by `CalibrateCMPS03()` routine. The I2C address for the CMPS03 device is defined using a constant as it is hardcoded on the CMPS03 device.

```
Private Const ADDRESS As Byte = &HC0    ' default I2C Address for CMPS03

Private Function GetCMPS03IntValue(ByVal reg as Byte) as Integer
   Dim cmd(1 to 3) as Byte
   Dim rc as Integer

   rc = I2CCmd(channel, ADDRESS, 1, reg, 2, cmd)
   If rc = 2 Then
       ' the MS byte is returned in the first byte, LS byte in the second
       GetCMPS03IntValue = CInt(MakeWord(cmd(2), cmd(1)))
   Else
       Debug.Print "GetRange i2cmd returned ";CStr(rc)
       GetCMPS03IntValue = -1
   End If
End Function
```

The `InitCMPS03()` subroutine is used to initialize the I2C channel as shown in the source code below. The I2C channel is hardcoded to be channel 0 which uses the underlying hardware-based I2C support. This was a deliberate design decision so that the Timer1 resource is available for other uses. The CMPS03 device works with the highest ZBasic supported I2C bit rate of 410K Hz. Note that if multiple modules call OpenI2C using the same I2C channel number, the communication bitrate is determined by the last caller.

```
Private Const BITRATE as Byte = 66      ' default 100KHz speed

Public Sub InitCMPS03()
   ' open the I2C channel
   Call OpenI2C(channel, sdaPin, sclPin, BITRATE)
   'Call OpenI2C(channel, sdaPin, sclPin, 10) ' this one overrides previous open
End Sub
```

Use of the I2C channel to the CMPS03 compass can be terminated using the `TermCMPS03()` subroutine. See the source zip file attached to this application note for the code to `TermCMPS03()`.

Using the CMPS03 Compass

The example test program (AN211.bas) below shows how to invoke the interface to the CMPS03 device as described previously. The `GetCMPS03Version()` function is used to get the software version for the CMPS03 device and source can be found in the associated zip file.

```
Private Const stopPin as Byte = A.2 ' just used for testing purposes

Sub Main()
   ' start test
   Debug.Print "Start of CMPS03 test"

   ' initialize CMPS03 and get software version number
   Call InitCMPS03()
   Call Sleep(10)  ' wait for compass to initialize after power up
   Debug.Print "CMPS03 Software Version is ";CStr(GetCMPS03Version())

   ' main test loop which gets the compass bearing
   ' until the stop button is pressed
   Do While GetPin(stopPin) = 1
       Debug.Print "Bearing ";CStr(Fmt(GetCMPS03Bearing(),1)); " degrees"
       Call Sleep(0.5)
   Loop

   ' stop using the Compass
   Call TermCMPS03()
   Debug.Print "CMPS03 test finished"
End Sub
```

Here is the corresponding console output from the above program:

```
Start of CMPS03 test
CMPS03 Software Version is 10
Bearing 61.2 degrees
Bearing 62.4 degrees
Bearing 60.2 degrees
…
Bearing 88.7 degrees
Bearing 72.8 degrees
Bearing 68.9 degrees
CMPS03 test finished
```

Calibrating the CMPS03 Compass

An important part of using the CMPS03 Magnetic Compass is that it needs to be calibrated at the target location using a traditional magnetic compass. This only needs to be done once. The `CalibrateCMPS03()` routine can be used to perform this calibration and it can be found in the associated ZIP file. The Robot Electronics calibration webpage (http://www.robot-electronics.co.uk/htm/cmps_cal.shtml) describes how to orient the device during calibration and an alternative method using a push button and pin 6 on the CMPS03 device.

Author

Mike Perks is a professional software engineer who became interested in microcontrollers a few years ago. Mike has written a number of articles, projects and application notes related to ZBasic, BasicX and AVR microcontrollers. Mike is also the owner of Oak Micros which specializes in AVR-based devices including his own ZX-based products. You may contact Mike at mikep@oakmicros.com or visit his website http://oakmicros.com.

Disclaimer: Elba Corp. makes no warranty regarding the accuracy of or the fitness for any particular purpose of the information in this document or the techniques described herein. The reader assumes the entire responsibility for the evaluation of and use of the information presented. The Company reserves the right to change the information described herein at any time without notice and does not make any commitment to update the information contained herein. No license to use proprietary information belonging to the Company or other parties is expressed or implied.