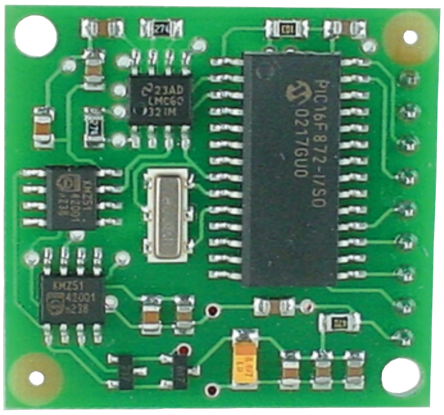
MODULE BOUSSOLE ÉLECTRONIQUE CMPS03

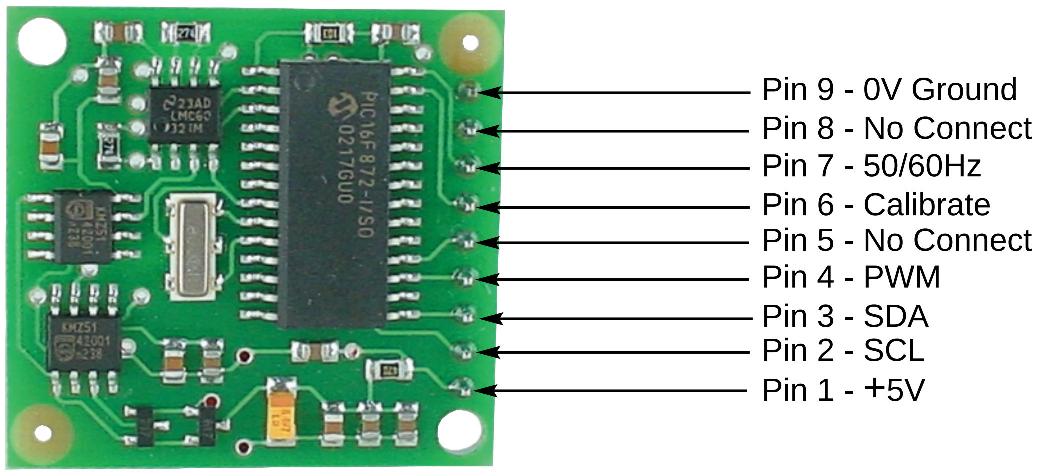
Référence : 0660-3



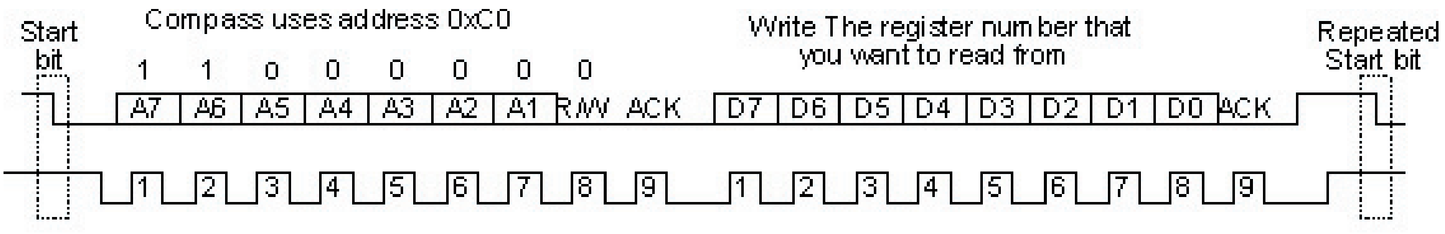
CMPS03 Magnetic Compass.

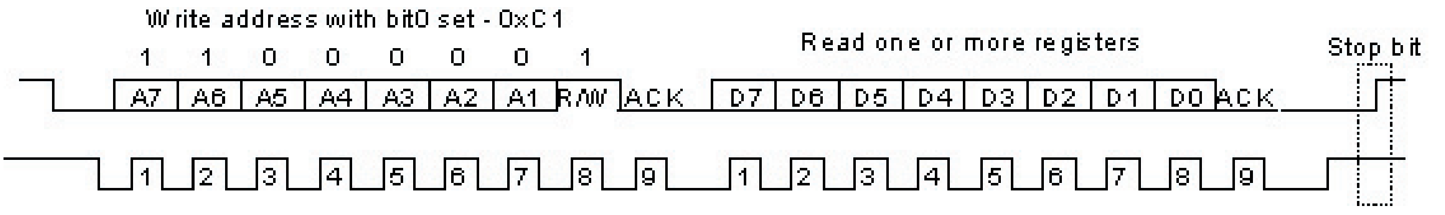
Voltage :	5v only required
Current :	20mA Typ.
Resolution :	0.1 Degree
Accuracy :	3-4 degrees approx. after calibration
Output 1 :	Timing Pulse 1mS to 37mS in 0.1mS increments
Output 2 :	I2C Interface, 0-255 and 0-3599
	SCL speed up to 1MHz
Small Size :	32mm x 35mm
Low Cost :	Best Price Compass Module Available

CMPS03 - Robot Compass Module
This compass module has been specifically designed for use in robots as an aid to navigation. The aim was to produce a unique number to represent the direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field. We have examples of using the Compass module with a wide range of popular controllers.
Connections to the compass module.



The compass module requires a 5v power supply at a nominal 15mA.
There are two ways of getting the bearing from the module. A PWM signal is available on pin 4, or an I2C interface is provided on pins 2,3.
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). Make sure you connect the I2C pins, SCL and SDA, to the 5v supply if you are using the PWM, as there are no pull-up resistors on these pins.
Pin 2,3 are an I2C interface and can be used to get a direct readout of the bearing. If the I2C interface is not used then these pins should be pulled high (to +5v) via a couple of resistors. Around 47k is ok, the values are not at all critical.


Start bit
Compass uses address 0xC0
1 1 0 0 0 0 0 0
A7 A6 A5 A4 A3 A2 A1 R/W ACK
Write The register number that you want to read from
D7 D6 D5 D4 D3 D2 D1 D0 ACK
Repeated Start bit
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9


Write address with bit0 set - 0xC1
1 1 0 0 0 0 0 1
A7 A6 A5 A4 A3 A2 A1 R/W ACK
Read one or more registers
D7 D6 D5 D4 D3 D2 D1 D0 ACK
Stop bit
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

I2C communication protocol with the compass module is the same as popular eeprom's such as the 24C04.. First send a start bit, the module address (0XC0) with the read/write bit low, then the register number you wish to read. This is followed by a repeated start and the module address again with the read/write bit high (0XC1). You now read one or two bytes for 8bit or 16bit registers respectively. 16bit registers are read high byte first. The compass has a 16 byte array of registers, some of which double up as 16 bit registers as follows;

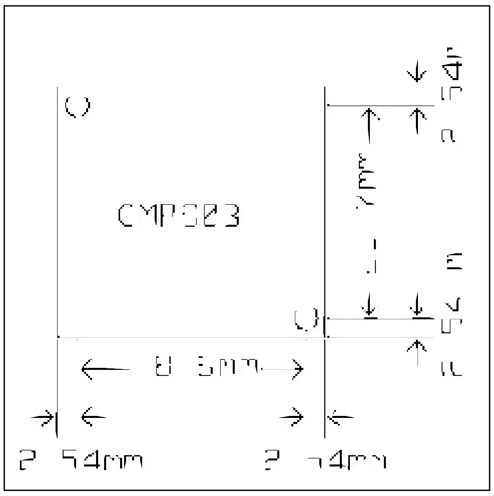
Register	Function
0	Software Revision Number
1	Compass Bearing as a byte, i.e. 0-255 for a full circle
2,3	Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-359.9 degrees.
4,5	Internal Test - Sensor1 difference signal - 16 bit signed word
6,7	Internal Test - Sensor2 difference signal - 16 bit signed word
8,9	Internal Test - Calibration value 1 - 16 bit signed word
10,11	Internal Test - Calibration value 2 - 16 bit signed word
12	Unused - Read as Zero
13	Unused - Read as Zero
14	Unused - Read as Undefined
15	Calibrate Command - Write 255 to perform calibration step. See text.

Register 0 is the Software revision number (8 at the time of writing). Register 1 is the bearing converted to a 0-255 value. This may be easier for some applications than 0-360 which requires two bytes. For those who require better resolution registers 2 and 3 (high byte first) are a 16 bit unsigned integer in the range 0-3599. This represents 0-359.9°. Registers 4 to 11 are internal test registers and 12,13 are unused. Register 14 is undefined. Don't read them if you don't want them - you'll just waste your I2C bandwidth. Register 15 is used to calibrate the compass.
The I2C interface does not have any pull-up resistors on the board, these should be provided elsewhere, most probably with the bus master. They are required on both the SCL and SDA lines, but only once for the whole bus, not on each module. I suggest a value of 1k8 if you are going to be working up to 400KHz and 1k2 or even 1k if you are going up to 1MHz. The compass is designed to work at up to the standard clock speed (SCL) of 100KHz, however the clock speed can be raised to 1MHZ providing the following precaution is taken;
At speeds above around 160KHz the CPU cannot respond fast enough to read the I2C data. Therefore a small delay of 50uS should be inserted either side of writing the register address. No delays are required anywhere else in the sequence. By doing this, I have tested the compass module up to 1.3MHz SCL clock speed. There is an example driver using the HITECH PICC compiler for the

PIC16F877. Note that the above is of no concern if you are using popular embedded language processors such as the. The compass module always operates as a slave, its never a bus master. Pin 7 is an input pin selecting either 50Hz (low) or 60Hz (high) operation. I added this option after noticing a jitter of around 1.5° in the output. The cause was the 50Hz mains field in my workshop. By converting in synchronism with the mains frequency this was reduced to around 0.2° . An internal conversion is done every 40mS (50Hz) or every 33.3mS (60Hz). The pin has an on-board pull-up can be left unconnected for 60Hz operation. There is no synchronism between the PWM or I2C outputs and the conversion. They both retrieve the most recent internal reading, which is continuously converted, whether it is used or not.
Pin 6 is used to calibrate the compass. The calibrate input (pin 6) has an on-board pull-up resistor and can be left unconnected after calibration. Calibration is identical to the CMPS01 Rev7 procedure.
Pins 5 and 8 are No Connect. Actually pin 8 is the processor reset line and has an on-board pull-up resistor. It is there so that we can program the processor chip after placement on the PCB.

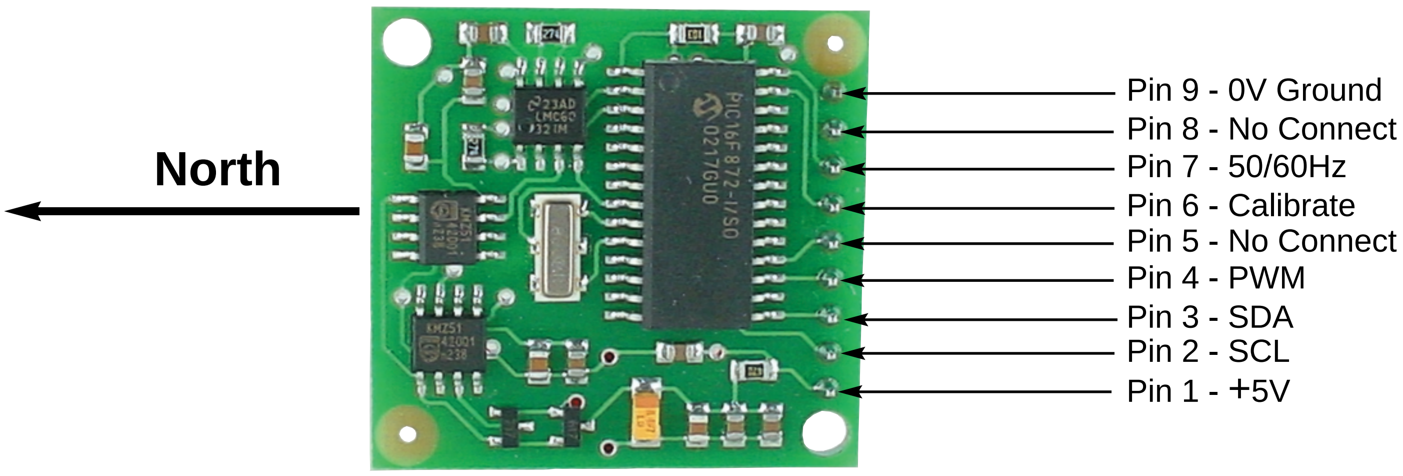
PCB Drilling Plan

The following diagram shows the CMPS03 PCB mounting hole positions.



Calibrating the CMPS01, CMPS03 Robot Compass Modules

CMPS03 - Calibration procedure is the same as CMPS01 Rev 7.
CMPS01 - As from Software revision 7, the calibration procedure has changed. Both methods are detailed below
Compass module orientation to produce 0 degrees reading.



Register	Function
0	**Software Revision Number**
1	**Compass Bearing as a byte, i.e. 0-255 for a full circle**
2,3	**Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-359.9 degrees.**
4,5	**Internal Test - Sensor1 difference signal - 16 bit signed word**
6,7	**Internal Test - Sensor2 difference signal - 16 bit signed word**
8,9	**Internal Test - Calibration value 1 - 16 bit signed word**
10,11	**Internal Test - Calibration value 2 - 16 bit signed word**
12	**Unused - Read as Zero**
13	**Unused - Read as Zero**
14	**Calibration Done Flag - Zero in calibrate mode when un-calibrated, 255 otherwise - unused in Rev 7 software & CMPS03**
15	**Calibrate Command - Write 255 to enter calibrate mode, write zero to exit. See text.**

15 Calibrate Command - Write 255 to enter calibrate mode, write zero to exit. See text.
Register 0 is the Software revision number (originally 3, now 7 with new calibration routines and 8 for the CMPS03).
Registers 14 & 15 are used to calibrate the compass. The procedure changed with Rev 7 software. Full calibration information is here

IMPORTANT - The compass module must be kept flat (horizontal and parallel to the earths surface) with the components on top and for the CMPS01 the sensors underneath. Keep the module away from metallic - especially magnetic - objects.

Calibrating Rev 3 Software - Recognized by lack of revision number on CPU chip, or read revision number from register 0

I2C Method
To calibrate the compass using the I2C bus, you only have to write 255 to register 15 and rotate the module very slowly through 360° . Writing zero to register 15 will store the calibration values in the processors internal EEPROM. Readings are taken by the processor at four compass points and these values are used to generate the calibration values. Register 14 reads 255 during normal operation. It reads zero when Calibrate mode is entered and 255 again when the four compass points have been measured. Register 14 will therefore indicate that the four points have been acquired and that zero can be written to register 15 to store the calibration and return to normal operation. It is necessary to rotate the compass very slowly during calibration to avoid missing the required compass points and to keep it horizontal to ensure the calibration figures are accurate.

Pin Method
Pins 5,6 are used to calibrate the compass. The calibrate input (pin 6) has an on-board pull-up resistor and can be left unconnected after calibration. To calibrate the compass you only have to take the calibrate pin low and rotate the module very slowly through 360° . Taking the calibrate pin high will store the calibration values in the processors internal EEPROM. Readings are taken by the processor at four compass points and these values are used to generate the calibration values. The CalDone output pin (pin 5) is high during normal operation. It goes low when the Calibrate pin is pulled low and high again when the four compass points have been measured. The CalDone pin will therefore indicate that the four points have been acquired and that the Calibrate pin can be raised

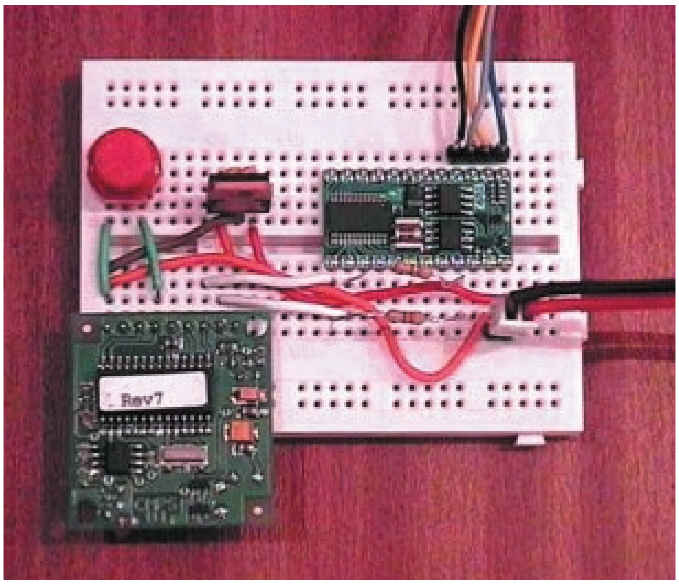
high again. It is necessary to rotate the compass slowly during calibration to avoid missing the required compass points and to keep it horizontal to ensure the calibration figures are accurate.
Calibrating Rev 7 Software - Recognized by revision number label on CMPS01 CPU chip, or read revision number from register 0.
Also applies to **Calibrating the CMPS03 Module.**

Note that pin 5 (CalDone) and register 14 (Calibration Done Flag) are not used with Rev 7 software or the CMPS03. Pin 5 should be left unconnected and register 14 ignored. When calibrating the compass, you must know exactly which direction is North, East, South and West. Don't guess at it. Get a magnetic needle compass and check it.

I2C Method
To calibrate using the I2C bus, you only have to write 255 (0xff) to register 15 for each of the four major compass points North, East, South and West. The 255 is cleared internally automatically after each point is calibrated. The compass points can be set in any order, but all four points must be calibrated. For example
1. Set the compass module flat, pointing North. Write 255 to register 15
2. Set the compass module flat, pointing East. Write 255 to register 15
3. Set the compass module flat, pointing South. Write 255 to register 15
4. Set the compass module flat, pointing West. Write 255 to register 15
That's it.

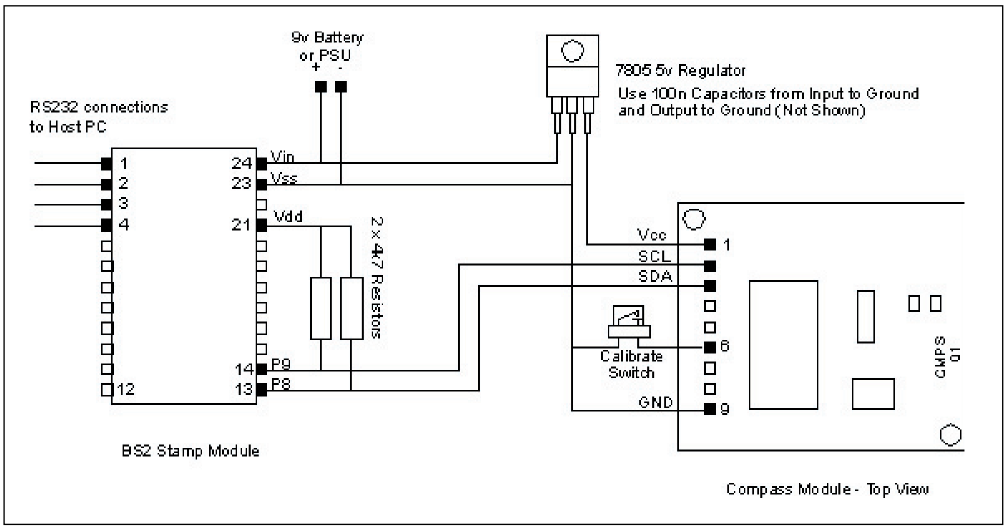
Pin Method
Pin 6 is used to calibrate the compass. The calibrate input (pin 6) has an on-board pull-up resistor and can be left unconnected after calibration. To calibrate the compass you only have to take the calibrate pin low and then high again for each of the four major compass points North, East, South and West. A simple push switch wired from pin6 to 0v (Ground) is OK for this. The compass points can be set in any order, but all four points must be calibrated. For example
1. Set the compass module flat, pointing North. Press and release the switch
2. Set the compass module flat, pointing East. Press and release the switch
3. Set the compass module flat, pointing South. Press and release the switch
4. Set the compass module flat, pointing West. Press and release the switch
That's it.

Connecting the CMPS01/03 to the BS2 using the PWM Signal



Introduction
The compass module uses either a PWM output or the I2C bus for communications. This example uses the PWM Signal. The example provided here uses PULSIN command. The PULSIN timing varies with the type of Stamp used. The BS2 and BS2e have 2uS resolution. The BS2sx has 0.8uS and the BS2p is 0.75uS. The example code below shows the calculation for each and is therefore suitable for use with all BS2 variants. It has been tested with the BS2 and BS2p.
The BS2 internal 5v regulator is not suitable for powering much external circuitry. I therefore recommend you use a separate 5v regulator.

Circuit Schematic for connecting the BS2 Stamp to the CMPS01/CMPS03 Compass Module using the PWM Signal



Note the switch connecting pin 6 to Ground. This is all that is needed for calibration of Rev7 Firmware.

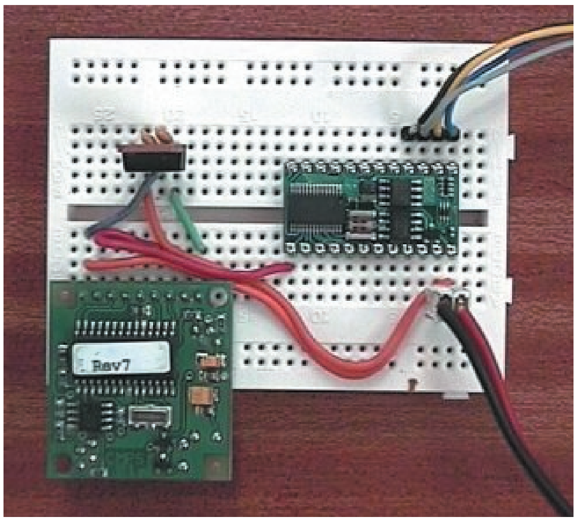
Software

The sample code below displays the compass bearing in Degrees (0-359) in a Debug window on the PC.

It can be downloaded

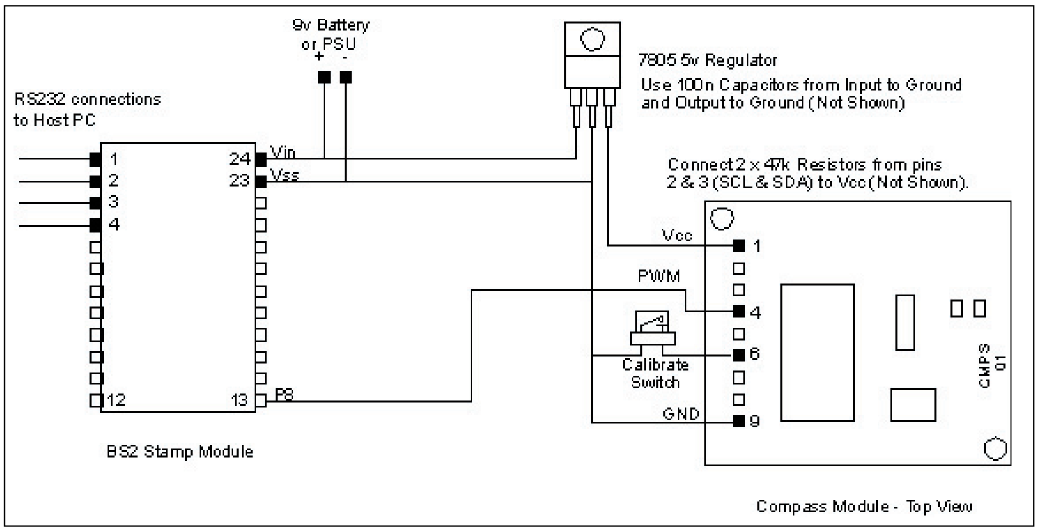
```
{$STAMP BS2}
'**********************************************************
'** **
'** PWM Routines for the Basic Stamp **
'** using the CMPS01/03 Compass Module **
'** **
'** Copyright 2002 - Devantech Ltd **
'** Commercial use of this software is prohibited **
'** Private and educational use only is permitted **
'** **
'** Written by Gerald Coe - January 2002 **
'** **
'** This Code has been Tested on BS2 and BS2p **
'** It should work equally well on the BS2e and BS2sx **
'** **
'**********************************************************
bearing var word
main:
pulsin 8, 1, bearing ' Get reading
bearing = (bearing-500)/50 ' BS2(e) - Calculate Bearing in degrees
' bearing = (bearing-1250)/125 ' BS2sx - Calculate Bearing in degrees
' bearing = (bearing-1333)/133 ' BS2p - Calculate Bearing in degrees
debug 2,1,1, "Compass Bearing ", dec3 bearing ' Display Compass Bearing
goto main ' around forever
```

Using the I2C Bus for Connecting the CMPS01/03 to the BS2



Introduction
The compass module uses either a PWM output or the I2C bus for communications. This example uses the I2C Bus. The SDA (data) and SCL (clock) lines are connected to pins P8 and P9 on the BS2. These are the pins used by the BS2p for the I2CIN and I2COUT commands. These commands currently cannot be use with the compass module.
This example uses a combination of bit bashing and the SHIFTIN and SHIFTOUT commands. It has been tested with the BS2 and BS2p.
The BS2 internal 5v regulator is not suitable for powering much external circuitry. I therefore recommend you use a separate 5v regulator.

Circuit Schematic for connecting the BS2 Stamp to the CMPS01/CMPS03 Compass Module using the I2C bus


9v Battery or PSU
+ -
7805 5v Regulator
Use 100n Capacitors from Input to Ground and Output to Ground (Not Shown)
RS232 connections to Host PC
Connect 2 x 47k Resistors from pins 2 & 3 (SCL & SDA) to Vcc (Not Shown).
1
2
3
4
24 Vin
23 Vss
Vcc
PWM
Calibrate Switch
GND
12
13 P8
BS2 Stamp Module
CMPS 01
Compass Module - Top View

Note the switch connecting pin 6 to Ground. This is all that is needed for calibration of Rev7 Firmware.

Software for BS2, BS2e, BS2sx and BS2p

The sample code below displays the compass bearing as both BRAD's (0-255) and Degrees (0-359) in a Debug window on the PC.
It can be downloaded

```
{$STAMP BS2}
'***********************************************************
'** **
'** I2C Routines for the Basic Stamp **
'** using the CMPS01/CMPS03 Compass Module **
'** **
'** Copyright 2002 - Devantech Ltd **
'** Commercial use of this software is prohibited **
'** Private and educational use only is permitted **
'** **
'** Written by Gerald Coe - January 2002 **
'** **
'** This Code has been Tested on BS2 and BS2p **
'** It should work equally well on the BS2e and BS2sx **
'** **
'**********************************************************

SDA con 8 ' I2C data
SCL con 9 ' I2C clock
SDAin var in8
SDAout var out8 ' To change the pins used, alter these 5 lines
SDAdir var dir8 ' The 4 SDA numbers must be the same, of course
loop var byte ' just a looping counter
I2cBuf var byte ' I2c read/write buffer
I2cAddr var byte ' Address of I2C device
I2cReg var byte ' Register number within I2C device
I2cData var word ' Data to read/write
I2cAck var bit ' Acknowledge bit

Main:
I2cAddr = $c0 ' CMPS01/03 Compass module address
I2cReg = 1 ' Bearing as 0-255 (BRAD)
gosub I2cByteRead
debug 2,0,0, "Compass Bearing (0-255 BRAD) ", dec3 I2cData
I2cReg = 2 ' Bearing as 0-359.9 degrees
gosub I2cWordRead
debug 2,0,1, "Compass Bearing (0-359 Degrees ", dec3 I2cData/10
goto main
'-------------------------------------------------------------------------------------------
' I2C subroutines follow
'-------------------------------------------------------------------------------------------
I2cByteWrite: ' writes I2cData.lowbyte to I2cReg at I2cAddr
gosub I2cStart
I2cBuf = I2cAddr
gosub I2cOutByte ' send device address
I2cBuf = I2cReg
gosub I2cOutByte ' send register number
I2cBuf = I2cData.lowbyte
gosub I2cOutByte ' send the data
gosub I2cStop
return
I2cWordWrite: ' writes I2cData to I2cReg at I2cAddr
gosub I2cStart
I2cBuf = I2cAddr
gosub I2cOutByte ' send device address
I2cBuf = I2cReg
gosub I2cOutByte ' send register number
I2cBuf = I2cData.highbyte
gosub I2cOutByte ' send the data - high byte
I2cBuf = I2cData.lowbyte
gosub I2cOutByte ' send the data - low byte
gosub I2cStop
return
I2CByteRead:
gosub I2cStart
I2cBuf = I2cAddr
gosub I2cOutByte ' send device address
```

```
I2cBuf = I2cReg
gosub I2cOutByte ' send register number
gosub I2cStart ' repeated start
I2cBuf = I2cAddr | 1
gosub I2cOutByte ' send device address (with read set)
I2cAck = 0 ' send Nak
gosub I2cInByte
I2cData.lowbyte = I2cBuf ' read the data
I2cData.highbyte = 0
gosub I2cStop
return
I2CWordRead:
gosub I2cStart
I2cBuf = I2cAddr
gosub I2cOutByte ' send device address
I2cBuf = I2cReg
gosub I2cOutByte ' send register number
gosub I2cStart ' repeated start
I2cBuf = I2cAddr | 1
I2cAck = 1 ' send Ack
gosub I2cOutByte ' send device address (with read set)
gosub I2cInByte
I2cData.highbyte = I2cBuf ' read the data
I2cAck = 0 ' send Nak
gosub I2cInByte
I2cData.lowbyte = I2cBuf
gosub I2cStop
return
I2cOutByte:
shiftout SDA, SCL, MSBFIRST, [I2cBuf]
input SDA
high SCL ' clock in the ack' bit
low SCL
return
I2cInByte:
shiftin SDA, SCL, MSBPRE, [I2cBuf]
SDAout = 0
SDAdir = I2cAck
high SCL ' clock out the ack' bit
low SCL
input SDA
return
I2cStart ' I2C start bit sequence
high SDA
high SCL
low SDA
low SCL
return
I2cStop: ' I2C stop bit sequence
low SDA
high SCL
high SDA
return
```

Notes

Notes

Produit importé et distribué par :

S e l e c t r o n i c

86 rue de CAMBRAI

59000 LILLE

TEL : 0 328 550 328 Fax : 0 328 550 329

SAV : 0 328 550 323 www.selectronic.fr

01/2003/ZM