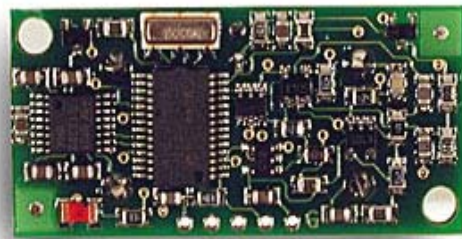


Devantech SRF08 UltraSonic Ranger



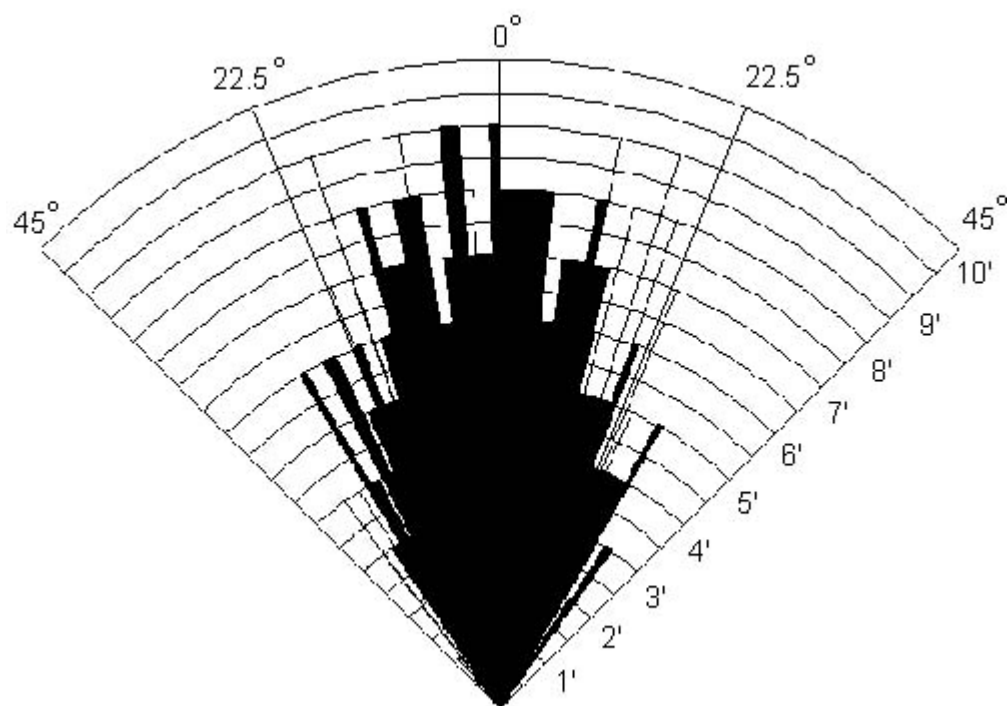
This Devantech high performance ultrasonic range finder is compact and measures an amazingly wide range from 3cm to 6m. The SRF08 interfaces to your microcontroller via the industry standard IIC bus. This ranger is perfect for your robot, or any other projects requiring accurate ranging information. There is even a built-in light sensor on the front of the module. You can also get a nifty Lynxmotion SRF08 Housing for [one ranger](#) or for [two rangers](#).

Specifications

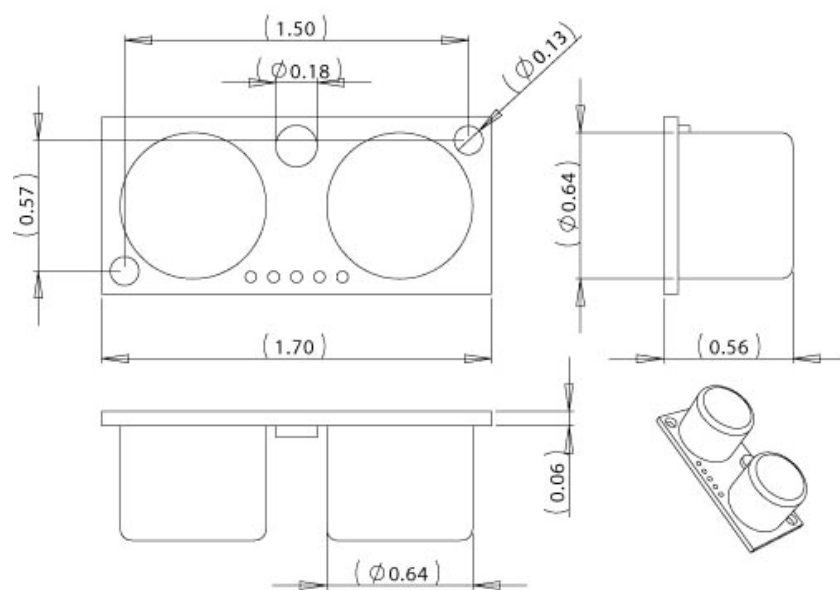
Beam Pattern	see graph
Voltage	5v
Current	15mA Typ. 3mA Standby
Frequency	40KHz
Maximum Range	6 m
Minimum Range	3 cm
Max Analogue Gain	Variable to 1025 in 32 steps
Connection	Standard IIC Bus
Light Sensor	Front facing light sensor
Timing	Fully timed echo, freeing host computer of task
Echo	Multiple echo - keeps looking after first echo
Units	Range reported n uS, mm or inches
Weight	0.4 oz.
Size	43mm w x 20mm d x 17mm h

Specifications subject to change without notice

Beam Pattern



Dimensions



Purchasing the SRF08

Acroname (www.acroname.com)

Price: \$52.00 each

Part Number: R145-SRF08

Technical Specifications for the SRF08

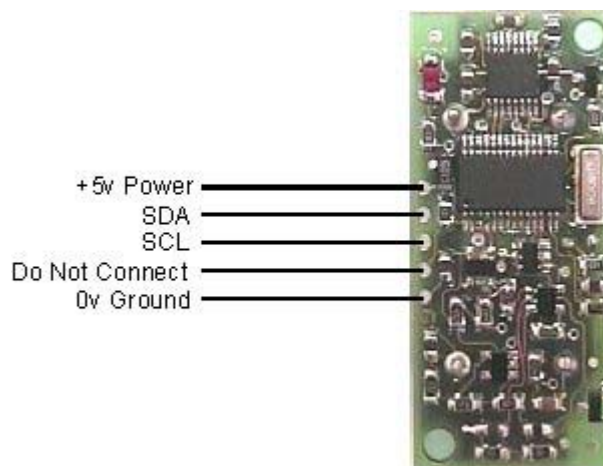
(Below is from <http://www.robot-electronics.co.uk/htm/srf08tech.shtml>), Gerry

gerry@robot-electronics.co.uk

Communication with the SRF08 ultrasonic rangefinder is via the I2C bus. This is available on popular controllers such as the OOPic and Stamp BS2p, as well as a wide variety of micro-controllers. To the programmer the SRF08 behaves in the same way as the ubiquitous 24xx series eeprom's, except that the I2C address is different. The default shipped address of the SRF08 is 0xE0. It can be changed by the user to any of 16 addresses E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC or FE, therefore up to 16 sonar's can be used. In addition to the above addresses, all sonar's on the I2C bus will respond to address 0 - the General Broadcast address. This means that writing a ranging command to I2C address 0 (0x00) will start all sonar's ranging at the same time. This should be useful in ANN Mode (See below). The results must be read individually from each sonar's real address. We have [examples](#) of using the SRF08 module with a wide range of popular controllers.

Connections

The "Do Not Connect" pin should be left unconnected. It is actually the CPU MCLR line and is used once only in our workshop to program the PIC16F872 on-board after assembly, and has an internal pull-up resistor. The SCL and SDA lines should each have a pull-up resistor to +5v somewhere on the I2C bus. You only need one pair of resistors, not a pair for every module. They are normally located with the bus master rather than the slaves. The SRF08 is always a slave - never a bus master. If you need them, I recommend 1.8k resistors. Some modules such as the OOPic already have pull-up resistors and you do not need to add any more.



Registers

The SRF08 appears as a set of 36 registers.

Location	Read	Write
0	Software Revision	Command Register
1	Light Sensor	Max Gain Register (default 31)
2	1st Echo High Byte	Range Register (default 255)
3	1st Echo Low Byte	N/A
~~~~	~~~~	~~~~
34	17th Echo High Byte	N/A
35	17th Echo Low Byte	N/A

Only locations 0, 1 and 2 can be written to. Location 0 is the command register and is used to start a ranging session. It cannot be read. Reading from location 0 returns the SRF08 software revision. By default, the ranging lasts for 65mS, but can be changed by writing to the range register at location 2. If you do so, then you will likely need to change the analogue gain by writing to location 1. See the **Changing Range** and **Analogue Gain** sections below.

Location 1 is the onboard light sensor. This data is updated every time a new ranging command has completed and can be read when range data is read. The next two locations, 2 and 3, are the 16bit unsigned result from the latest ranging - high byte first. The meaning of this value depends on the command used, and is either the range in inches, or the range in cm or the flight time in uS. A value of zero indicates that no objects were detected. There are up to a further 16 results indicating echo's from more distant objects.

## Commands

There are three commands to initiate a ranging (80 to 82), to return the result in inches, centimeters or microseconds. There is also an ANN mode (Artificial Neural Network) mode which is described later and a set of commands to change the I2C address.

Command		Action
Decimal	Hex	
80	0x50	Ranging Mode - Result in inches
81	0x51	Ranging Mode - Result in centimeters
82	0x52	Ranging Mode - Result in micro-seconds
83	0x53	ANN Mode - Result in inches
84	0x54	ANN Mode - Result in centimeters
85	0x55	ANN Mode - Result in micro-seconds

160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

## Ranging Mode

To initiate a ranging, write one of the above commands to the command register and wait the required amount of time for completion and read as many results as you wish. The echo buffer is cleared at the start of each ranging. The first echo range is placed in locations 2,3. the second in 4,5, etc. If a location (high and low bytes) is 0, then there will be no further reading in the rest of the registers. The default and recommended time for completion of ranging is 65mS, however you can shorten this by writing to the range register before issuing a ranging command. Light sensor data at location 1 will also have been updated after a ranging command.

## ANN Mode

ANN mode (Artificial Neural Network) is designed to provide the multi echo data in a way that is easier to input to a neural network, at least I hope it is - I've not actually done it yet. ANN mode provides a 32 byte buffer (locations 4 to 35 inclusive) where each byte represents the 65536uS maximum flight time divided into 32 chunks of 2048uS each - equivalent to about 352mm of range. If an echo is received within a bytes time slot then it will be set to non-zero, otherwise it will be zero. So if an echo is received from within the first 352mm, location 4 will be non-zero. If an object is detected 3m away the location 12 will be non-zero ( $3000/352 = 8$ ) ( $8+4=12$ ). Arranging the data like this should be better for a neural net than the other formats. The input to your network should be 0 if the byte is zero and 1 if its non-zero. I have a SOFM (Self Organizing Feature Map) in mind for the neural net, but will hopefully be useful for any type.

Location 4	Location 5	Location 6	Location 7	Locations 8 - 35
0 - 352mm	353 - 705mm	706 - 1057mm	1058 - 1410mm	and so on

Locations 2,3 contain the range of the nearest object converted to inches, cm or uS and is the same as for Ranging Mode.

## Checking for Completion of Ranging

You do not have to use a timer on your own controller to wait for ranging to finish. You can take advantage of the fact that the SRF08 will not respond to any I2C activity whilst ranging. Therefore, if you try to read from the SRF08 (we use the software revision number a location 0) then you will get 255 (0xFF) whilst ranging. This is because the I2C data line (SDA) is pulled high if nothing is driving it. As soon as the ranging is complete the SRF08 will again respond to the I2C bus, so just keep reading the register until its not 255 (0xFF) anymore. You can then read the sonar data. Your controller can take advantage of this to perform other tasks while the SRF08 is ranging.

## Changing the Range

The maximum range of the SRF08 is set by an internal timer. By default, this is 65mS or the equivalent of 11 metres of range. This is much further than the 6 metres the SRF08 is actually capable of. It is possible to reduce the time the SRF08 listens for an echo, and hence the range, by writing to the range register at location 2. The range can be set in steps of about 43mm (0.043m or 1.68 inches) up to 11 metres.

The range is  $((\text{Range Register} \times 43\text{mm}) + 43\text{mm})$  so setting the Range Register to 0 (0x00) gives a maximum range of 43mm. Setting the Range Register to 1 (0x01) gives a maximum range of 86mm. More usefully, 24 (0x18) gives a range of 1 metre and 140 (0x8C) is 6 metres. Setting 255 (0xFF) gives the original 11 metres ( $255 \times 43 + 43$  is 11008mm). There are two reasons you may wish to reduce the range.

1. To get at the range information quicker
2. To be able to fire the SRF08 at a faster rate.

If you only wish to get at the range information a bit sooner and will continue to fire the SRF08 at 65ms or slower, then all will be well. However if you wish to fire the SRF08 at a faster rate than 65mS, you will definitely need to reduce the gain - see next section.

The range is set to maximum every time the SRF08 is powered-up. If you need a different range, change it once as part of your system initialization code.

### Analogue Gain

The analogue gain register sets the *Maximum* gain of the analogue stages. To set the maximum gain, just write one of these values to the gain register at location 1. During a ranging, the analogue gain starts off at its minimum value of 94. This is increased at approx. 70uS intervals up to the maximum gain setting, set by register 1. Maximum possible gain is reached after about 390mm of range. The purpose of providing a limit to the maximum gain is to allow you to fire the sonar more rapidly than 65mS. Since the ranging can be very short, a new ranging can be initiated as soon as the previous range data has been read. A potential hazard with this is that the second ranging may pick up a distant echo returning from the previous "ping", give a false result of a close by object when there is none. To reduce this possibility, the maximum gain can be reduced to limit the modules sensitivity to the weaker distant echo, whilst still able to detect close by objects. The maximum gain setting is stored only in the CPU's RAM and is initialized to maximum on power-up, so if you only want to do a ranging every 65mS, or longer, you can ignore the Range and Gain Registers.

**Note** - Effective in Ranging Mode only, in ANN mode, gain is controlled automatically.

Gain Register		Maximum Analogue Gain
Decimal	Hex	
0	0x00	Set Maximum Analogue Gain to 94
1	0x01	Set Maximum Analogue Gain to 97
2	0x02	Set Maximum Analogue Gain to 100
3	0x03	Set Maximum Analogue Gain to 103
4	0x04	Set Maximum Analogue Gain to 107
5	0x05	Set Maximum Analogue Gain to 110
6	0x06	Set Maximum Analogue Gain to 114
7	0x07	Set Maximum Analogue Gain to 118
8	0x08	Set Maximum Analogue Gain to 123
9	0x09	Set Maximum Analogue Gain to 128
10	0x0A	Set Maximum Analogue Gain to 133
11	0x0B	Set Maximum Analogue Gain to 139
12	0x0C	Set Maximum Analogue Gain to 145
13	0x0D	Set Maximum Analogue Gain to 152
14	0x0E	Set Maximum Analogue Gain to 159
15	0x0F	Set Maximum Analogue Gain to 168
16	0x10	Set Maximum Analogue Gain to 177
17	0x11	Set Maximum Analogue Gain to 187
18	0x12	Set Maximum Analogue Gain to 199
19	0x13	Set Maximum Analogue Gain to 212
20	0x14	Set Maximum Analogue Gain to 227
21	0x15	Set Maximum Analogue Gain to 245
22	0x16	Set Maximum Analogue Gain to 265
23	0x17	Set Maximum Analogue Gain to 288
24	0x18	Set Maximum Analogue Gain to 317



25	0x18	Set Maximum Analogue Gain to 352
26	0x20	Set Maximum Analogue Gain to 395
27	0x21	Set Maximum Analogue Gain to 450
28	0x22	Set Maximum Analogue Gain to 524
29	0x23	Set Maximum Analogue Gain to 626
30	0x24	Set Maximum Analogue Gain to 777
31	0x25	Set Maximum Analogue Gain to 1025

Note that the relationship between the Gain Register setting and the actual gain is not a linear one. Also there is no magic formula to say "use this gain setting with that range setting". It depends on the size, shape and material of the object and what else is around in the room. Try playing with different settings until you get the result you want. If you appear to get false readings, it may be echo's from previous "pings", try going back to firing the SRF08 every 65mS or longer (slower). If you are in any doubt about the Range and Gain Registers, remember they are automatically set by the SRF08 to their default values when it is powered-up. You can ignore and forget about them and the SRF08 will work fine, detecting objects up to 6 metres away every 65mS or slower.

### Light Sensor

The SRF08 has a light sensor on-board. A reading of the light intensity is made by the SRF08 each time a ranging takes place in either Ranging or ANN Modes ( The A/D conversion is actually done just before the "ping" whilst the +/- 10v generator is stabilizing). The reading increases as the brightness increases, so you will get a maximum value in bright light and minimum value in darkness. It should get close to 2-3 in complete darkness and up to about 248 (0xF8) in bright light. The light intensity can be read from the Light Sensor Register at location 1 at the same time that you are reading the range data.

### LED

The red LED is used to flash out a code for the I2C address on power-up (see below). It also gives a brief flash during the "ping" whilst ranging.

### Changing the I2C Bus Address

To change the I2C address of the SRF08 you must have only one sonar on the bus. Write the 3 sequence commands in the correct order followed by the address. Example; to change the address of a sonar currently at 0xE0 (the default shipped address) to 0xF2, write the following to address 0xE0; (0xA0, 0xAA, 0xA5, 0xF2 ). These commands must be sent in the correct sequence to change the I2C address, additionally, No other command may be issued in the middle of the sequence. The sequence must be sent to the command register at location 0, which means 4 separate write transactions on the I2C bus. When done, you should label the sonar with its address, however if you do forget, just power it up without sending any commands. The SRF08 will flash its address out on the LED. One long flash followed by a number of shorter flashes indicating its address. The flashing is terminated immediately on sending a command the SRF08.

Address		Long Flash	Short flashes
Decimal	Hex		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6

238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10
246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

Take care not to set more than one sonar to the same address, there will be a bus collision and very unpredictable results.

### Current Consumption

Average current consumption measured on our prototype is around 12mA during ranging, and 3mA standby. The module will automatically go to standby mode after a ranging, whilst waiting for a new command on the I2C bus. The actual measured current profile is as follows;

Operation	Current	Duration
Ranging command received - Power on	275mA	3uS
+/- 10v generator Stabilization	25mA	600uS
8 cycles of 40kHz "ping"	40mA	200uS
Ranging	11mA	65mS max
Standby	3mA	indefinite

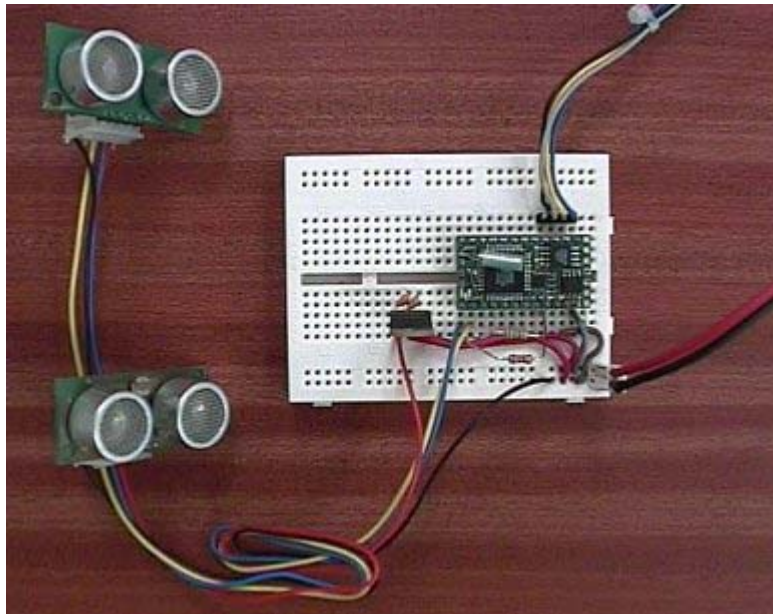
The above values are for guidance only, they are not tested on production units.

## Code

From <http://www.robot-electronics.co.uk/htm/srf08bx24.shtml>

## Connecting Multiple SRF08 Sonar Modules to the BX-24



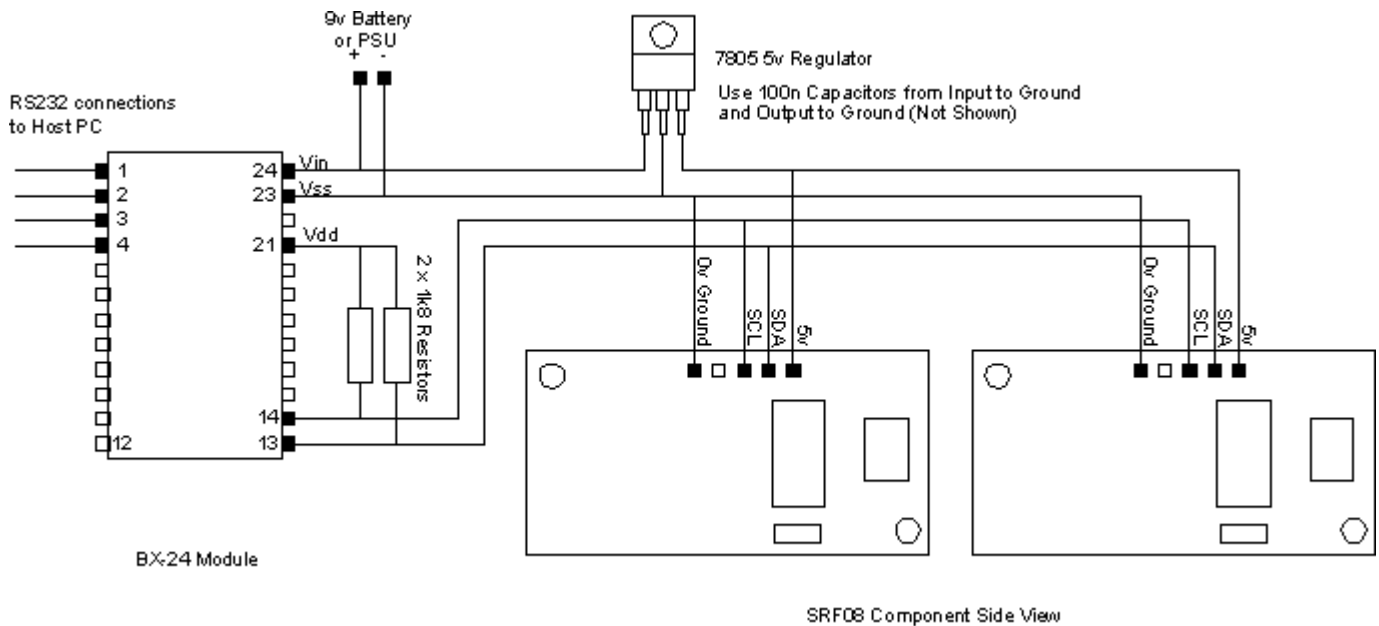


## Introduction

The SRF08 modules use the I2C bus for communication. This example shows how to connect two SRF08's to the BX-24, however it is expandable to up to 16 SRF08's on the I2C bus. The SDA (data) and SCL (clock) lines are connected to pins 13 and 14 on the BX-24. The BX-24 does not have I2C communication, so the example provided here uses a combination of bit bashing and the SHIFTIN and SHIFTOUT commands instead.

The BX-24 internal 5v regulator is not suitable for powering much external circuitry. I therefore recommend you use a separate 5v regulator as shown below..

## Circuit Schematic for connecting two SRF08 Sonar Modules to the BX-24



The schematic above shows 1k8 pull-up resistors on the SCL and SDA lines to Vdd. This is for good noise immunity, however any value up to 4k7 should be OK.

## Changing the SRF08 I2C Address

Before you can use the SRF08's you will need to re-program their I2C addresses from the default address of 0xE0 they are supplied with. The program below will do this. Make sure you only have one SRF08 connected when you do this. You only have to change the SRF08_NEW_ADDRESS constant in the program below to the address you want. For example if you want the SRF08 to be at hex address 0xF2, then change SRF08_NEW_ADDRESS to read;

```
Const SRF08_NEW_ADDRESS As Byte = $Hf2      ' Place new address for SRF08 here
```

Now download the program to the BX-24, you will see rapid brief flashes on the Red Led on the SRF08 indicating that the change of address was successful. If you set your Monitor port on the PC, you will see the LDR and first range displayed on screen. It is wise to make a note of the new address on the SRF08 itself. It is easy to forget which is which otherwise.

To use the example code described later on this page, set one SRF08 to address 0xE0 and the other to 0xE2. The following program can be downloaded [here](#).

```
*****
**
**      I2C Routines for the Basic BX-24      **
**      to change the I2C address of the SRF08  **
**
**      Copyright 2002 - Devantech Ltd          **
**      Commercial use of this software is prohibited  **
**      Private and educational use only is permitted  **
**
**      Written by Gerald Coe - February 2002      **
**
*****

Const SRF08_NEW_ADDRESS As Byte = &He0      ' Place new address for SRF08 here
'available addresses are: e0, e2, e4, e6, e8, ea, ec, ee
'      f0, f2, f4, f6, f8, fa, fc, fe

Const SCL As Byte = 14      ' I2C clock - choose any pins you wish for SCL and SDA
Const SDA As Byte = 13      ' I2C data

Const GB As Byte = 0      ' I2C General Broadcast address
Const CmdReg As Byte = 0      ' SRF08 command register
Const LdrReg As Byte = 1      ' Address of Light Sensor Register in SRF08
Const RangeReg As Byte = 2      ' Address of Range Register in SRF08
Const RangeCmd As Byte = 81      ' Ranging command - 80 for inches, 81 for cm, 82 for uS

Dim I2cAck As Boolean      ' Acknowledge flag

Sub Main()
Dim Ldr As Byte
Dim Range As New UnsignedInteger

    Call PutPin(SCL, bxOutputHigh)
```

```

Call PutPin(SDA, bxOutputHigh)

Call Delay(1.0) ' Delay just to be sure SRF08 is out of reset
Call I2cByteWrite(GB, CmdReg, &Ha0) ' 1st command in address change sequence
Call I2cByteWrite(GB, CmdReg, &Haa) ' 2nd command in address change sequence
Call I2cByteWrite(GB, CmdReg, &Ha5) ' 3rd command in address change sequence
Call I2cByteWrite(GB, CmdReg, SRF08_NEW_ADDRESS) ' The new I2C address

' That's the address changed, now perform SRF08 Ranging in an endless loop at the new address
Do
    Call I2cByteWrite(SRF08_NEW_ADDRESS, CmdReg, RangeCmd) ' Start Ranging in Cm
    Call Delay(0.07) ' 70mS wait for ranging to complete
    Ldr = I2cByteRead(SRF08_NEW_ADDRESS, LdrReg) ' Read light sensor
    Range = I2cWordRead(SRF08_NEW_ADDRESS, RangeReg) ' Read Range Register
    debug.Print "LDR = "; CStr(Ldr); ", Range = "; CStr(Range)
Loop
End Sub

'-----
' I2C subroutines follow
'-----

' writes I2cData to I2cReg at I2cAddr
Sub I2cByteWrite(ByVal I2cAddr As Byte, ByVal I2cReg As Byte, ByVal I2cData As Byte)
    Call I2cStart()
    Call I2cOutByte(I2cAddr) ' send device address
    Call I2cOutByte(I2cReg) ' send register address
    Call I2cOutByte(I2cData) ' send the data
    Call I2cStop()
End Sub

Function I2CByteRead(ByVal I2cAddr As Byte, ByVal I2cReg As Byte) As Byte
    Call I2cStart()
    Call I2cOutByte(I2cAddr) ' send device address
    Call I2cOutByte(I2cReg) ' send register address
    Call I2cStart() ' repeated start
    I2cAddr = I2cAddr+1
    Call I2cOutByte(I2cAddr) ' send device address with read set
    I2cAck = False ' setup to send Nak
    I2cByteRead = I2cInByte() ' get data byte with Nak
    Call I2cStop()
End Function

Function I2CWordRead(ByVal I2cAddr As Byte, ByVal I2cReg As Byte) As UnsignedInteger
    Set I2CWordRead = New UnsignedInteger
    Call I2cStart()
    Call I2cOutByte(I2cAddr) ' send device address
    Call I2cOutByte(I2cReg) ' send register address
    Call I2cStart() ' repeated start

```

```

I2cAddr = I2cAddr+1
Call I2cOutByte(I2cAddr)          ' send device address with read set
I2cAck = True                     ' setup to send Ack
I2cWordRead = CuInt(I2cInByte()*256)
I2cAck = False                    ' setup to send Nak
I2cWordRead = I2cWordRead + CuInt(I2cInByte())
Call I2cStop()
End Function

Sub I2cOutByte(I2cData As Byte)
    Call ShiftOut(SDA, SCL, 8, I2cData)    ' shift data out
    Call PutPin(SDA, bxInputTristate)      ' turn SDA around
    Call PutPin(SCL, bxOutputHigh)         ' and clock in the ack' bit
    Call PutPin(SCL, bxOutputLow)
End Sub

Function I2cInByte() As Byte
    I2cInByte = ShiftIn(SDA, SCL, 8)
    If I2cAck=True Then
        Call PutPin(SDA, bxOutputLow)
    Else
        Call PutPin(SDA, bxOutputHigh)
    End If
    Call PutPin(SCL, bxOutputHigh)         ' clock out the ack' bit
    Call PutPin(SCL, bxOutputLow)
End Function

Sub I2cStart()                    ' I2C start bit sequence
    Call PutPin(SDA, bxOutputHigh)
    Call PutPin(SCL, bxOutputHigh)
    Call PutPin(SDA, bxOutputLow)
    Call PutPin(SCL, bxOutputLow)
End Sub

Sub I2cStop()                    ' I2C stop bit sequence
    Call PutPin(SDA, bxOutputLow)
    Call PutPin(SCL, bxOutputHigh)
    Call PutPin(SDA, bxOutputHigh)
End Sub

```

### Displaying Light Sensor and Range readings in a PC Debug window

Now that you have your SRF08's re-programmed to their new I2C addresses (0xE0 and 0xE2) the following sample code will display the light sensor reading and the 1st range reading, for each SRF08 in the monitor port window on the PC .

The sample code below can be downloaded [here](#).

```

*****

```

```

***          **
***      I2C Routines for the BX-24          **
***      to demonstrate the use of multiple SRF08's      **
***          **
***      Copyright 2002 - Devantech Ltd          **
***      Commercial use of this software is prohibited      **
***      Private and educational use only is permitted      **
***          **
***      Written by Gerald Coe - February 2002          **
***          **
*****

Const SCL As Byte = 14          ' I2C clock - choose any pins you wish for SCL and SDA
Const SDA As Byte = 13          ' I2C data

Const CmdReg As Byte = 0          ' SRF08 command register
Const LdrReg As Byte = 1          ' Address of Light Sensor Register in SRF08
Const RangeReg As Byte = 2          ' Address of Range Register in SRF08
Const RangeCmd As Byte = 81          ' Ranging command - 80 for inches, 81 for cm, 82 for uS

' Note that SRF08's must have been previously set to these addresses
Const Sonar1 As Byte = &He0          ' 1st SRF08 at I2C address 0Xe0
Const Sonar2 As Byte = &He2          ' 2nd SRF08 at I2C address 0Xe2

Dim I2cAck As Boolean          ' Acknowledge flag

Sub Main()
Dim Ldr1 As Byte
Dim Range1 As New UnsignedInteger
Dim Ldr2 As Byte
Dim Range2 As New UnsignedInteger

    Call PutPin(SCL, bxOutputHigh)
    Call PutPin(SDA, bxOutputHigh)

Do
' 1st SRF08 Ranger
    Call I2cByteWrite(Sonar1, CmdReg, RangeCmd) ' Start Ranging in Cm
    Call Delay(0.07)          ' 70mS wait for ranging to complete
    Ldr1 = I2cByteRead(Sonar1, LdrReg)          ' Read light sensor
    Range1 = I2cWordRead(Sonar1, RangeReg)      ' Read Range Register

' 2nd SRF08 Ranger
    Call I2cByteWrite(Sonar2, CmdReg, RangeCmd) ' Start Ranging in Cm
    Call Delay(0.07)          ' 70mS wait for ranging to complete
    Ldr2 = I2cByteRead(Sonar2, LdrReg)          ' Read light sensor
    Range2 = I2cWordRead(Sonar2, RangeReg)      ' Read Range Register

    debug.Print "LDR1 = "; CStr(Ldr1); ", Range1 = "; CStr(Range1); _
        " LDR2 = "; CStr(Ldr2); ", Range2 = "; CStr(Range2)

Loop
End Sub

```

```

'-----
' I2C subroutines follow
'-----

' writes I2cData to I2cReg at I2cAddr
Sub I2cByteWrite(ByVal I2cAddr As Byte, ByVal I2cReg As Byte, ByVal I2cData As Byte)
    Call I2cStart()
    Call I2cOutByte(I2cAddr)          ' send device address
    Call I2cOutByte(I2cReg)           ' send register address
    Call I2cOutByte(I2cData)          ' send the data
    Call I2cStop()
End Sub

Function I2CByteRead(ByVal I2cAddr As Byte, ByVal I2cReg As Byte) As Byte
    Call I2cStart()
    Call I2cOutByte(I2cAddr)          ' send device address
    Call I2cOutByte(I2cReg)           ' send register address
    Call I2cStart()                   ' repeated start
    I2cAddr = I2cAddr+1
    Call I2cOutByte(I2cAddr)          ' send device address with read set
    I2cAck = False                    ' setup to send Nak
    I2cByteRead = I2cInByte()         ' get data byte with Nak
    Call I2cStop()
End Function

Function I2CWordRead(ByVal I2cAddr As Byte, ByVal I2cReg As Byte) As UnsignedInteger
    Set I2CWordRead = New UnsignedInteger
    Call I2cStart()
    Call I2cOutByte(I2cAddr)          ' send device address
    Call I2cOutByte(I2cReg)           ' send register address
    Call I2cStart()                   ' repeated start
    I2cAddr = I2cAddr+1
    Call I2cOutByte(I2cAddr)          ' send device address with read set
    I2cAck = True                     ' setup to send Ack
    I2cWordRead = CuInt(I2cInByte()*256)
    I2cAck = False                    ' setup to send Nak
    I2cWordRead = I2cWordRead + CuInt(I2cInByte())
    Call I2cStop()
End Function

Sub I2cOutByte(I2cData As Byte)
    Call ShiftOut(SDA, SCL, 8, I2cData) ' shift data out
    Call PutPin(SDA, bxInputTristate)    ' turn SDA around
    Call PutPin(SCL, bxOutputHigh)        ' and clock in the ack' bit
    Call PutPin(SCL, bxOutputLow)
End Sub

Function I2cInByte() As Byte
    I2cInByte = ShiftIn(SDA, SCL, 8)

```

```

If I2cAck=True Then
    Call PutPin(SDA, bxOutputLow)
Else
    Call PutPin(SDA, bxOutputHigh)
End If
Call PutPin(SCL, bxOutputHigh)      ' clock out the ack' bit
Call PutPin(SCL, bxOutputLow)
End Function

Sub I2cStart()                      ' I2C start bit sequence
    Call PutPin(SDA, bxOutputHigh)
    Call PutPin(SCL, bxOutputHigh)
    Call PutPin(SDA, bxOutputLow)
    Call PutPin(SCL, bxOutputLow)
End Sub

Sub I2cStop()                      ' I2C stop bit sequence
    Call PutPin(SDA, bxOutputLow)
    Call PutPin(SCL, bxOutputHigh)
    Call PutPin(SDA, bxOutputHigh)
End Sub

```

You can find more information on the SRF08 [here](#)

## SRF08 Ultra sonic range finder - a little History

The SRF08 is an evolutionary step from the SRF04, developed to improve on the following features of the SRF04. The key points are;

1. The maximum range of 3m can be limiting in some situations.
2. The 36mS timeout + 10mS recharge is rather long – equivalent to almost 8m on a 3m product.
3. The SRF04 requires 2 I/O pins per sonar. 32 I/O lines for a 16 sonar system.
4. The users host processor is required to time the returning echo
5. The 50mA maximum current is too high – 800mA for 16 sonar's.
6. Only a single returning echo is possible.
7. The SRF04 can't see the light (read on)

The 3m limit of the SRF04 is imposed by the need not to have a gain so high that the cross coupling between transmit and receive transducers causes the op-amps saturate at close range. If they did then the system could not tell the difference between the cross coupling and a legitimate returning echo. The SRF08 uses a digital pot to vary the gain as the range increases. This allows a higher overall gain to be set and consequently better range. The Typical range we are seeing on the prototype is 6m and we have had it up to 11m for a large object. This is too sensitive because it detects small close by anomalies in the floor that the robot really ought to ignore. The gain was therefore deliberately reduced to around 6m.

The 36mS timeout of the SRF04 was imposed because the PIC12C508 processor used only has a single timer, and this is used for tone detection of the returning echo. The watchdog timer is used to time out the ranging. This could only be set in increments of 18mS. Whilst 18mS is just enough –



about 3m range – it is a “typical” value only and not guaranteed, so the real range could be less depending on ambient temperature and chip tolerances. A further 10mS is specified in order to recharge the +/- 10v supplies for the op-amp and comparator. The max232 IC is switched off during echo timing to reduce noise in the op-amps. With the SRF08 the analog circuit has been changed to a single 5v supply, so the max232 (actually an ST232) does not need to charge up a 22uF capacitor, only the 100n’s. Recharge time now drops to just 600uS and is taken care of by the processor automatically when a new reading is requested. A change of processor from the PIC12C508 to the PIC16F872 means more timers are available and the SRF08 is not stuck with the 36mS watchdog timer. However one of the problems with terminating the ranging early is that the in-flight “ping” does not know this. It quite happily bounces off a far wall and returns. Now if it happens to return just after you have started a new ranging, the sonar will pick up this earlier “ping” and think there is an object much closer than there really is. The SRF08 allows the maximum gain to be limited to reduce this possibility.

The number of I/O lines required by multiple sonar’s has been an issue with some users. There is also a problem which has been identified with the basic stamp, which does not treat all I/O lines equally when timing. When using 16 sonar’s, 32 I/O lines are required. This can be reduced to 17 by gating the 16 echo pulse outputs together with 16 diodes. A further reduction to 6 I/O lines can be achieved by using a 4 to 16 line decoder such as the CD4514B. This involves the user building additional circuitry. The SRF08 uses the I2C interface so all 16 sonar’s can be controlled using just 2 I/O lines. The I2C bus interface is available on popular controllers such as the OOPic, and of course cheap processors such as many of the PIC family.

On the SRF04, the users host processor is required to time the returning echo. This has been an issue when using the Stamp, as it does not treat all I/O lines equally. This is an internal problem with the Stamp and was discovered by Jim Fry of Lynxmotion. The SRF08 does its own internal timing and sends you the result.

The 50mA max. current required by the SRF04, whilst already far better than the 150mA (2.5A peak) of the Polaroid units, has been further reduced to 15mA nominal and around 3mA in standby. The SRF08 automatically goes into standby when it has completed each ranging, and powers up again when it receives the next command.

Because of the way the SRF04 works, only a single echo can be received. After this the module powers up its +/- 10v generators again ready for the next trigger pulse. With the SRF08 multiple echo’s can be received. A buffer stores the first 16 echo’s received. The idea is to be able to see through open doorways where a standard sonar would just see the door frame.

Finally, to make the SRF08 even more useful, I included a light sensor. This is readable over the I2C bus just as the sonar data is.