



Column #99 July 2003 by Jon Williams:

## You've Got Robot Eyes

*How about something simple this month, yet useful and even fun. Good – I've had a crazy couple of weeks leading up to writing this, especially the last few days. Today I told my best friend Farah, that the only thing stopping me from throwing myself from the balcony is the thought that falling just three stories would only add painful injury to the string of insults [uncooperative projects] I've been dealing with for the past several days.... Zoiks.*

Now, things haven't been all bad. The other day I received a package from my friends in Holland (I conducted some BASIC and Javelin Stamp training sessions there in February). Amongst the goodies were some great books ... and books are just about my favorite thing in the whole wide world next to BASIC Stamps. The book that I've really been enjoying while on my reading breaks is "The I2C Bus" (ISBN 0-905705-47-5) that is published by Elektor Electronics magazine. If you've not seen Elektor, it is to Europe what Nuts & Volts is to the United States: a high-quality electronics magazine that targets serious hobbyists like you and me.

Thumbing through the various I2C projects in the book encouraged me to spruce up my BASIC Stamp I2C library (for non-BS2p/pe Stamps) to take advantage of PBASIC 2.5, to put aside my various uncooperative projects, and to try the Devantech SRF08 ultrasonic range finder that's been begging me for some attention.

The SRF08 is physically identical to the SRF04 we experimented with last year, but its two-wire connection is an I2C bus instead of a trigger input and pulse output. The advantage for us is that we can connect up to 16 SRF08s on the same two wires. This is perfect for an array of robot "eyes."

Let's get right to it. Figure 99.1 shows the connections that will work with our BS2/BS2e/BS2sx I2C routines, as well as I2COUT an I2CIN on the BS2p and BS2pe.

Figure 99.1: SRF08 to BASIC Stamp Connection

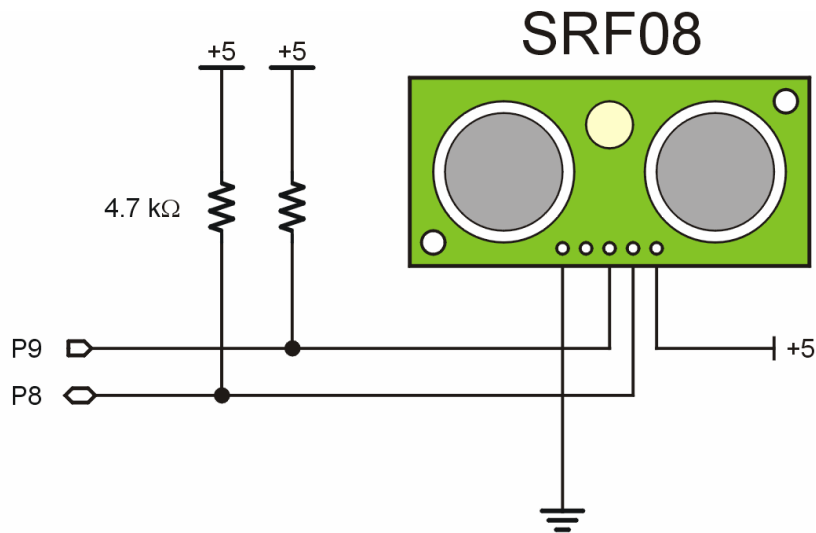
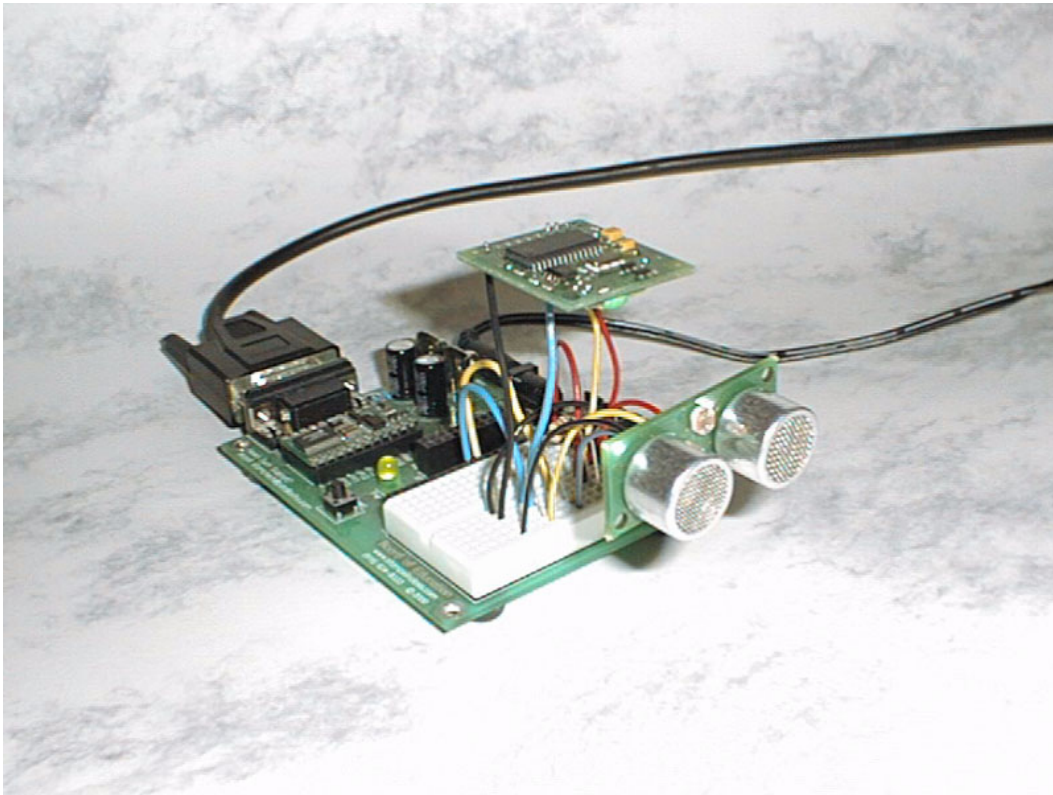


Figure 99.2: SRF08 to BASIC Stamp



### What's Your Address?

Since we discussed I2C bus details last year, I won't go into that except to remind you that I2C devices have a slave address (device code) and internal register addresses. Most I2C devices have external address pins that allow us to set the device address (a few devices have fixed addresses – like the CMPS01 from Devantech). With the SRF08 we must set its address electronically; through the I2C bus.

If we have only SRF08 in a project, we can use the default address, \$E0, and deal with the range finder much like an I2C EEPROM. To use more than one on the same bus, of course, necessitates an address change in one or more units. This is actually very simple to do. What is required is the

## Column #99: You've Got Robot Eyes

transmission of a specific byte sequence (\$A0, \$AA, \$A5) to the command register (\$00), followed by the new address (there are 16 valid choices). Here's the code:

```
FOR idx = 0 TO 3
  i2cSlave = $00
  i2cReg = RegCmd
  LOOKUP idx, [$A0, $AA, $A5, NewSRF08], i2cData
  GOSUB Write Byte
NEXT
```

The slave address used by this routine is the "broadcast" address (\$00) that will act on any SRF08 (more on this later). Once the address is set, it's a good idea to give the use a method of verification. Luckily, the SRF08 has an onboard LED that blinks when its slave address is received. We can cause this LED to blink by putting the code into a loop that writes a valid command to the SRF08 command register.

```
DO
  i2cSlave = NewSRF08
  i2cReg = RegCmd
  i2cData = $50
  GOSUB Write Byte
  PAUSE 65
LOOP
```

When power is cycled on the SRF08, the LED will display a code of long and short blinks that indicates the slave address setting. It is a good idea, however, to mark the device after we've changed the address – this will make things easier later.

### How Far to the Next Obstacle?

With the device address set, reading the distance from the sensor to an obstacle is a simple process:

- Send a ranging command to the desired device
- Wait 65 milliseconds for the measurement
- Read the range

The first step is to send a ranging command. In addition to the ability to control multiple devices on the same bus, I really like this feature: we can specify range in inches, centimeters, or in microseconds. This saves us conversion math in PBASIC. To start the process, the ranging command (\$50 for inches, \$51 for centimeters, \$52 for microseconds) is written to the command register at [internal] address \$00. After 65 milliseconds, the ranging value is ready and we read the two-byte range value from registers \$02 and \$03.

If you look closely at the SRF08 you'll see a CdS photocell nestled neatly between the ultrasonic elements. This is another nice feature of the SRF08: each time a range measurement is made, the light falling on the photocell is measured and placed in register \$01. This will be an eight-bit value that will get larger as more light falls on the photocell.

Since the program is fairly short, we can print most of it [sans I2C routines] here. This setup code takes care of formatting our DEBUG terminal screen and displaying the revision code of the SRF08. The revision code is read from register \$00 (same address where we write the command value).

```
Setup:
  DEBUG CLS
  DEBUG CrsrXY, 0, 0, "Devantech SRF08 Range Finder Demo"
  DEBUG CrsrXY, 0, 1, "-----"

  i2cSlave = SRF08
  i2cReg = RegCmd
  GOSUB Read Byte
  DEBUG CrsrXY, 0, 3, "Rev Num... "
  DEBUG DEC i2cData, ClrEOL

  DEBUG CrsrXY, 0, 5, "Light.... "
  DEBUG CrsrXY, 0, 6, "Range.... "
```

The main code sends the ranging command (inches), gives the SRF08 the time it needs to make the measurement, then reads back and displays the light and measurement values. The guys at Devantech have made the SRF08 a breeze to use.

```
Main:
  i2cSlave = SRF08
  i2cReg = RegCmd
  i2cData = RngIn
  GOSUB Write Byte
  PAUSE 65
  i2cReg = RegLight
  GOSUB Read Byte
  DEBUG CrsrXY, 11, 5, DEC i2cData, ClrEOL
  i2cReg = RegRange
  GOSUB Read Word
  DEBUG CrsrXY, 11, 6, DEC i2cData, " in.", ClrEOL
  GOTO Main
```

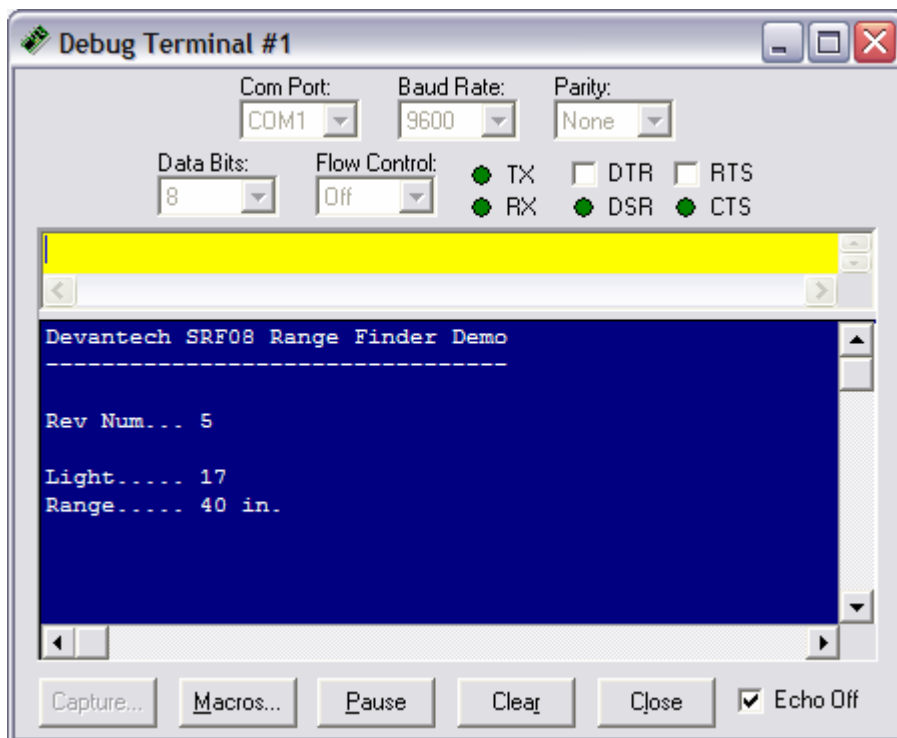
Now, what if we want to talk to more than one SRF08? Do we need to address them and wait for each one? No. Remember I mentioned the "broadcast" command earlier? All we have to do is

## Column #99: You've Got Robot Eyes

send a ranging command to the broadcast address (\$00), wait 65 milliseconds, then loop through each of our sensors and read its range value.

```
Read_Sensors:
  i2cSlave = $00
  i2cReg = RegCmd
  i2cData = RngIn
  GOSUB Write Byte
  PAUSE 65
  FOR idx = 0 TO 2
    LOOKUP idx, [$E0, $E2, $E4], i2cSlave
    i2cReg = RegRange
    GOSUB Read Word
    sonar(idx) = i2cData
  NEXT
```

Figure 99.3: Example DEBUG Output



### Remote Robot Start With a Laser

Many of the contests conducted by the Dallas Personal Robotics Group give a few extra points to robots that can be started without touching them. At a contest some months back, DPRG member David Anderson (a great robot builder and programmer) asked me if I had a laser pointer with me. As it happened, I did have one in my toolbox. He smiled, pointed it at his robot, and away it went – extra points scored.

Remember a couple months ago when I talked about learning through imitating? Well, that's what I did with David's laser trick. Since the SRF08 is a great robot part, and it has a CdS photocell, I wrote a short piece of code that lets us handle "non-contact" starts.

The concept is simple: read the ambient light on start-up, then wait until there is a significant rise in the light level. This rise, of course, will be caused by the laser pointer hitting the CdS. In my code, I decided a 25% change is significant. You may need to adjust for the ambient lighting in your situation. A smaller percentage change will make the trigger more sensitive. Be careful though, as too small a change could make the remote start subject to false triggering.

Here's the code that reads the ambient light and establishes the start threshold of 125%:

```
Setup:
  i2cSlave = SRF08
  i2cReg = RegCmd
  i2cData = RngIn
  GOSUB Write_Byte
  PAUSE 65
  i2cReg = RegLight
  GOSUB Read_Byte
  thresh = i2cData.LowByte */ $0140 MAX 255
```

The next step, then, is to monitor the light level until the threshold is exceeded.

Wait\_For\_Start:

```
DO
  i2cSlave = SRF08
  i2cReg = RegCmd
  i2cData = RngIn
  GOSUB Write_Byte
  PAUSE 65
  i2cReg = RegLight
  GOSUB Read_Byte
LOOP UNTIL (i2cData >= thresh)
```

## **Column #99: You've Got Robot Eyes**

See? I told you this was going to be easy. It's really a credit to Devantech and the work they put into the SRF08. And bear in mind that the SRF08 is capable of more than we've covered here – but these are the features that are most useful. The documentation and Devantech web site cover advanced features like multiple echoes.

For fun I've included a photo of my test BOE with an SRF08 and CMPS01 – just to make sure they "play nice." Of course they do. Now I just need to strap it onto a BOE-Bot chassis and let it tear around my apartment. After I do that I think I'll attach a big laser and target some of those uncooperative projects that have been troubling me for the past several days. I'll let you know how that goes.

Until next time, Happy Stamping.

```

' =====
'
' File..... SRF08_Demo.BS2
' Purpose... Daventech SRF08 Ultrasonic Range Finder Demonstration
' Author.... Jon Williams
' E-mail.... jwilliams@parallax.com
' Started...
' Updated... 20 MAY 2003
'
' {$STAMP BS2}
' {$PBASIC 2.5}
' =====
'
' ----[ Program Description ]-----
'
' ----[ Revision History ]-----
'
' ----[ I/O Definitions ]-----
SDA          PIN      8          ' I2C serial data line
SCL          PIN      9          ' I2C serial clock line
'
' ----[ Constants ]-----
SRF08        CON      $E2        ' address of SRF08
RegCmd       CON      0          ' command register
RegLight     CON      1          ' light value
RegRange     CON      2          ' ranger reading
RngIn        CON      $50        ' range in inches
RngCm        CON      $51        ' range in centimeters
RngUs        CON      $52        ' range in microseconds
Ack          CON      0          ' acknowledge bit
Nak          CON      1          ' no ack bit
'
' ----[ Variables ]-----
i2cSlave     VAR      Byte       ' I2C device address
i2cReg       VAR      Byte       ' register address
i2cData      VAR      Word       ' data to/from device
i2cWork      VAR      Byte       ' work byte for I2C io
i2cAck       VAR      Bit        ' Ack bit from device

```

## Column #99: You've Got Robot Eyes

```
' -----[ EEPROM Data ]-----  
  
' -----[ Initialization ]-----  
  
Setup:  
  DEBUG CLS  
  DEBUG CRSRXY, 0, 0, "Devantech SRF08 Range Finder Demo"  
  DEBUG CRSRXY, 0, 1, "-----"  
  
  i2cSlave = SRF08                ' set device ID  
  i2cReg = RegCmd                 ' rev number in cmd reg  
  GOSUB Read Byte  
  DEBUG CRSRXY, 0, 3, "Rev Num... "  
  DEBUG DEC i2cData, CLREOL  
  
  DEBUG CRSRXY, 0, 5, "Light.... "  
  DEBUG CRSRXY, 0, 6, "Range.... "  
  
' -----[ Program Code ]-----  
  
Main:  
  i2cSlave = SRF08  
  i2cReg = RegCmd  
  i2cData = RngIn                 ' get range in inches  
  GOSUB Write Byte  
  PAUSE 65                        ' allow measurement  
  i2cReg = RegLight  
  GOSUB Read Byte  
  DEBUG CRSRXY, 11, 5, DEC i2cData, CLREOL  
  i2cReg = RegRange  
  GOSUB Read Word  
  DEBUG CRSRXY, 11, 6, DEC i2cData, " in.", CLREOL  
  GOTO Main  
  
  END  
  
' -----[ Subroutines ]-----  
  
' -----  
' High Level I2C Subroutines  
' -----  
  
' Writes low byte of i2cData to i2cReg  
  
Write_Byte:
```

```

GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte           ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte           ' send register number
i2cWork = i2cData.LowByte
GOSUB I2C_TX_Byte           ' send the data
GOSUB I2C_Stop
RETURN

' Writes i2cData to i2cReg

Write Word:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte           ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte           ' send register number
i2cWork = i2cData.HighByte
GOSUB I2C_TX_Byte           ' send the data - high byte
i2cWork = i2cData.LowByte
GOSUB I2C_TX_Byte           ' send the data - low byte
GOSUB I2C_Stop
RETURN

' Read i2cData (8 bits) from i2cReg

Read_Byte:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte           ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte           ' send register number
GOSUB I2C_Start           ' start (sets register)
i2cWork = (i2cSlave | 1)
GOSUB I2C_TX_Byte           ' send read command
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData = i2cWork           ' return the data
RETURN

' Read i2cData (16 bits) from i2cReg

Read Word:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte           ' send device address
i2cWork = i2cReg

```

## Column #99: You've Got Robot Eyes

```
GOSUB I2C_TX_Byte           ' send register number
GOSUB I2C_Start            ' start (sets register)
i2cWork = (i2cSlave | 1)
GOSUB I2C_TX_Byte         ' send read command
GOSUB I2C_RX_Byte
i2cData.HighByte = i2cWork ' read high byte of data
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData.LowByte = i2cWork  ' read low byte of data
RETURN

' -----
' Low Level I2C Subroutines
' -----

' --- Start ---

I2C_Start:                ' I2C start bit sequence
  INPUT SDA
  INPUT SCL
  LOW SDA                 ' SDA --> low while SCL high

Clock_Hold:
  DO : LOOP WHILE (SCL = 0) ' wait for device ready
  RETURN

' --- Transmit ---

I2C_TX_Byte:
  SHIFTOUT SDA, SCL, MSBFIRST, [i2cWork\8] ' send byte to device
  SHIF TIN SDA, SCL, MSBP RE, [i2cAck\1]    ' get acknowledge bit
  RETURN

' --- Receive ---

I2C_RX_Byte_Nak:
  i2cAck = Nak           ' no Ack = high
  GOTO I2C_RX

I2C_RX_Byte:
  i2cAck = Ack           ' Ack = low

I2C_RX:
  SHIF TIN SDA, SCL, MSBP RE, [i2cWork\8] ' get byte from device
  SHIF TOU T SDA, SCL, LSBFIR ST, [i2cAck\1] ' send ack or nak
  RETURN
```

```
' --- Stop ---  
  
I2C Stop:                                ' I2C stop bit sequence  
  LOW SDA  
  INPUT SCL  
  INPUT SDA                                ' SDA --> high while SCL high  
  RETURN
```

## Column #99: You've Got Robot Eyes

```
' =====  
'  
' File..... SRF08 Laser Start.BS2  
' Purpose... Using the SRF08 Light Sensor for Robot Remote Start  
' Author.... Jon Williams  
' E-mail.... jwilliams@parallax.com  
' Started...  
' Updated... 20 MAY 2003  
'  
' {$STAMP BS2}  
' {$PBASIC 2.5}  
' =====  
'  
' -----[ Program Description ]-----  
'  
' Monitors the light sensor for a distinct change -- can be used as a  
' "non-contact" start for a robot.  
'  
' -----[ Revision History ]-----  
'  
' -----[ I/O Definitions ]-----  
'  
SDA          PIN      8          ' I2C serial data line  
SCL          PIN      9          ' I2C serial clock line  
'  
' -----[ Constants ]-----  
'  
SRF08        CON      $E2          ' address of SRF08  
  
RegCmd       CON      0          ' command register  
RegLight     CON      1          ' light value  
RegRange     CON      2          ' ranger reading  
  
RngIn        CON      $50          ' range in inches  
RngCm        CON      $51          ' range in centimeters  
RngUs        CON      $52          ' range in microseconds  
  
Ack          CON      0          ' acknowledge bit  
Nak          CON      1          ' no ack bit  
'  
' -----[ Variables ]-----  
'  
i2cSlave     VAR      Byte        ' I2C device address  
i2cReg       VAR      Byte        ' register address  
i2cData      VAR      Word        ' data to/from device
```

```

i2cWork      VAR      Byte      ' work byte for I2C io
i2cAck       VAR      Bit       ' Ack bit from device

thresh       VAR      Byte

' -----[ EEPROM Data ]-----

' -----[ Initialization ]-----

Setup:
  i2cSlave = SRF08
  i2cReg = RegCmd
  i2cData = RngIn          ' send range command
  GOSUB Write_Byte
  PAUSE 65                 ' allow measurement
  i2cReg = RegLight
  GOSUB Read_Byte
  thresh = i2cData.LOWBYTE */ $0140 MAX 255 ' thresh = ambient * 1.25

  DEBUG CLS
  DEBUG "Threshold set.  Waiting for start."

' -----[ Program Code ]-----

Wait For Start:
  DO
    i2cSlave = SRF08
    i2cReg = RegCmd
    i2cData = RngIn          ' send range command
    GOSUB Write_Byte
    PAUSE 65                 ' allow measurement
    i2cReg = RegLight
    GOSUB Read_Byte
  LOOP UNTIL (i2cData >= thresh)

Main:
  DEBUG CR
  DEBUG "Start detected."

  END

' -----[ Subroutines ]-----

' -----
' High Level I2C Subroutines
' -----

```

## Column #99: You've Got Robot Eyes

```
' Writes low byte of i2cData to i2cReg

Write Byte:
  GOSUB I2C_Start
  i2cWork = i2cSlave
  GOSUB I2C_TX_Byte                    ' send device address
  i2cWork = i2cReg
  GOSUB I2C_TX_Byte                    ' send register number
  i2cWork = i2cData.LOWBYTE
  GOSUB I2C_TX_Byte                    ' send the data
  GOSUB I2C_Stop
  RETURN

' Writes i2cData to i2cReg

Write Word:
  GOSUB I2C_Start
  i2cWork = i2cSlave
  GOSUB I2C_TX_Byte                    ' send device address
  i2cWork = i2cReg
  GOSUB I2C_TX_Byte                    ' send register number
  i2cWork = i2cData.HIGHBYTE
  GOSUB I2C_TX_Byte                    ' send the data - high byte
  i2cWork = i2cData.LOWBYTE
  GOSUB I2C_TX_Byte                    ' send the data - low byte
  GOSUB I2C_Stop
  RETURN

' Read i2cData (8 bits) from i2cReg

Read Byte:
  GOSUB I2C_Start
  i2cWork = i2cSlave
  GOSUB I2C_TX_Byte                    ' send device address
  i2cWork = i2cReg
  GOSUB I2C_TX_Byte                    ' send register number
  GOSUB I2C_Start                      ' start (sets register)
  i2cWork = (i2cSlave | 1)
  GOSUB I2C_TX_Byte                    ' send read command
  GOSUB I2C_RX_Byte_Nak
  GOSUB I2C_Stop
  i2cData = i2cWork                    ' return the data
  RETURN

' Read i2cData (16 bits) from i2cReg

Read Word:
  GOSUB I2C_Start
```

```

i2cWork = i2cSlave
GOSUB I2C_TX_Byte                    ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte                    ' send register number
GOSUB I2C_Start                      ' start (sets register)
i2cWork = (i2cSlave | 1)
GOSUB I2C_TX_Byte                    ' send read command
GOSUB I2C_RX_Byte
i2cData.HighByte = i2cWork          ' read high byte of data
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData.LowByte = i2cWork           ' read low byte of data
RETURN

' -----
' Low Level I2C Subroutines
' -----

' --- Start ---

I2C_Start:                          ' I2C start bit sequence
  INPUT SDA
  INPUT SCL
  LOW SDA                            ' SDA --> low while SCL high

Clock_Hold:
  DO : LOOP WHILE (SCL = 0)          ' wait for device ready
  RETURN

' --- Transmit ---

I2C_TX_Byte:
  SHIFTOUT SDA, SCL, MSBFIRST, [i2cWork\8] ' send byte to device
  SHIFTOUT SDA, SCL, MSBPRES, [i2cAck\1]   ' get acknowledge bit
  RETURN

' --- Receive ---

I2C_RX_Byte_Nak:
  i2cAck = Nak                       ' no Ack = high
  GOTO I2C_RX

I2C_RX_Byte:
  i2cAck = Ack                        ' Ack = low

I2C_RX:
  SHIFTOUT SDA, SCL, MSBPRES, [i2cWork\8] ' get byte from device
  SHIFTOUT SDA, SCL, LSBFIRST, [i2cAck\1] ' send ack or nak

```

**Column #99: You've Got Robot Eyes**

```
RETURN

' --- Stop ---

I2C_Stop:                                ' I2C stop bit sequence
LOW SDA
INPUT SCL                                ' SDA --> high while SCL high
INPUT SDA
RETURN
```

```

' =====
'
' File..... SRF08 Addr Set.BS2
' Purpose... Daventech SRF08 Ultrasonic Range Finder Address Change
' Author.... Jon Williams
' E-mail.... jwilliams@parallax.com
' Started...
' Updated... 20 MAY 2003
'
' {$STAMP BS2}
' {$PBASIC 2.5}
' =====
'
' -----[ Program Description ]-----
'
' Use this program to set new address in SRF08. To simplify the process,
' this routine uses the broadcast address, so make sure that only one
' device is connected.
'
' Valid Addresses:
'   $E0, $E2, $E4, $E6, $E8, $EA, $EC, $EE
'   $F0, $F2, $F4, $F6, $F8, $FA, $FC, $FE
'
' -----[ Revision History ]-----
'
' -----[ I/O Definitions ]-----
SDA          PIN      8          ' I2C serial data line
SCL          PIN      9          ' I2C serial clock line
'
' -----[ Constants ]-----
NewSRF08     CON      $E2          ' new address for SRF08
RegCmd       CON      0          ' command register
RegLight     CON      1          ' light value
RegRange     CON      2          ' ranger reading
Ack          CON      0          ' acknowledge bit
Nak          CON      1          ' no ack bit
'
' -----[ Variables ]-----
i2cSlave     VAR      Byte        ' I2C device address
i2cReg       VAR      Byte        ' register address

```

## Column #99: You've Got Robot Eyes

```
i2cData      VAR      Word      ' data to/from device
i2cWork      VAR      Byte      ' work byte for I2C io
i2cAck       VAR      Bit       ' Ack bit from device

idx          VAR      Nib       ' loop counter

' -----[ EEPROM Data ]-----
' -----[ Initialization ]-----
' -----[ Program Code ]-----

Main:
  DEBUG CLS
  DEBUG "Changing SRF08 address to: ", IHEX2 NewSRF08, CR
  DEBUG "Blinking LED confirms address change."

  FOR idx = 0 TO 3                ' send byte sequence
    i2cSlave = $00
    i2cReg = RegCmd
    LOOKUP idx, [$A0, $AA, $A5, NewSRF08], i2cData
    GOSUB Write_Byte
  NEXT

  DO                              ' blink LED to confirm
    i2cSlave = NewSRF08           ' address
    i2cReg = RegCmd               ' command register
    i2cData = $50                 ' range command
    GOSUB Write_Byte             ' send it
    PAUSE 65
  LOOP

' -----[ Subroutines ]-----
' -----
' High Level I2C Subroutines
' -----

' Writes low byte of i2cData to i2cReg

Write Byte:
  GOSUB I2C Start
  i2cWork = i2cSlave
  GOSUB I2C_TX_Byte               ' send device address
  i2cWork = i2cReg
  GOSUB I2C TX Byte               ' send register number
  i2cWork = i2cData.LowByte
```

```

GOSUB I2C_TX_Byte          ' send the data
GOSUB I2C_Stop
RETURN

' Writes i2cData to i2cReg

Write Word:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte          ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte          ' send register number
i2cWork = i2cData.HighByte
GOSUB I2C_TX_Byte          ' send the data - high byte
i2cWork = i2cData.LowByte
GOSUB I2C_TX_Byte          ' send the data - low byte
GOSUB I2C_Stop
RETURN

' Read i2cData (8 bits) from i2cReg

Read_Byte:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte          ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte          ' send register number
GOSUB I2C_Start            ' start (sets register)
i2cWork = (i2cSlave | 1)
GOSUB I2C_TX_Byte          ' send read command
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData = i2cWork          ' return the data
RETURN

' Read i2cData (16 bits) from i2cReg

Read_Word:
GOSUB I2C_Start
i2cWork = i2cSlave
GOSUB I2C_TX_Byte          ' send device address
i2cWork = i2cReg
GOSUB I2C_TX_Byte          ' send register number
GOSUB I2C_Start            ' start (sets register)
i2cWork = (i2cSlave | 1)
GOSUB I2C_TX_Byte          ' send read command
GOSUB I2C_RX_Byte
i2cData.HighByte = i2cWork ' read high byte of data

```

## Column #99: You've Got Robot Eyes

```
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData.LowByte = i2cWork          ' read low byte of data
RETURN

' -----
' Low Level I2C Subroutines
' -----

' --- Start ---

I2C_Start:                          ' I2C start bit sequence
  INPUT SDA
  INPUT SCL
  LOW SDA                            ' SDA -> low while SCL high

Clock_Hold:                          ' wait for device ready
  DO : LOOP WHILE (SCL = 0)
  RETURN

' --- Transmit ---

I2C_TX_Byte:
  SHIFTOUT SDA, SCL, MSBFIRST, [i2cWork\8] ' send byte to device
  SHIF TIN SDA, SCL, MSBP RE, [i2cAck\1]    ' get acknowledge bit
  RETURN

' --- Receive ---

I2C_RX_Byte_Nak:
  i2cAck = Nak                        ' no Ack = high
  GOTO I2C_RX

I2C_RX_Byte:
  i2cAck = Ack                        ' Ack = low

I2C_RX:
  SHIF TIN SDA, SCL, MSBP RE, [i2cWork\8]  ' get byte from device
  SHIF TOU T SDA, SCL, LSBFIR ST, [i2cAck\1] ' send ack or nak
  RETURN

' --- Stop ---

I2C_Stop:                            ' I2C stop bit sequence
  LOW SDA
  INPUT SCL
  INPUT SDA                            ' SDA --> high while SCL high
  RETURN
```