ZBasic

Application Note

AN-215 Measuring Distance using Ultrasonic Echos and Timer1

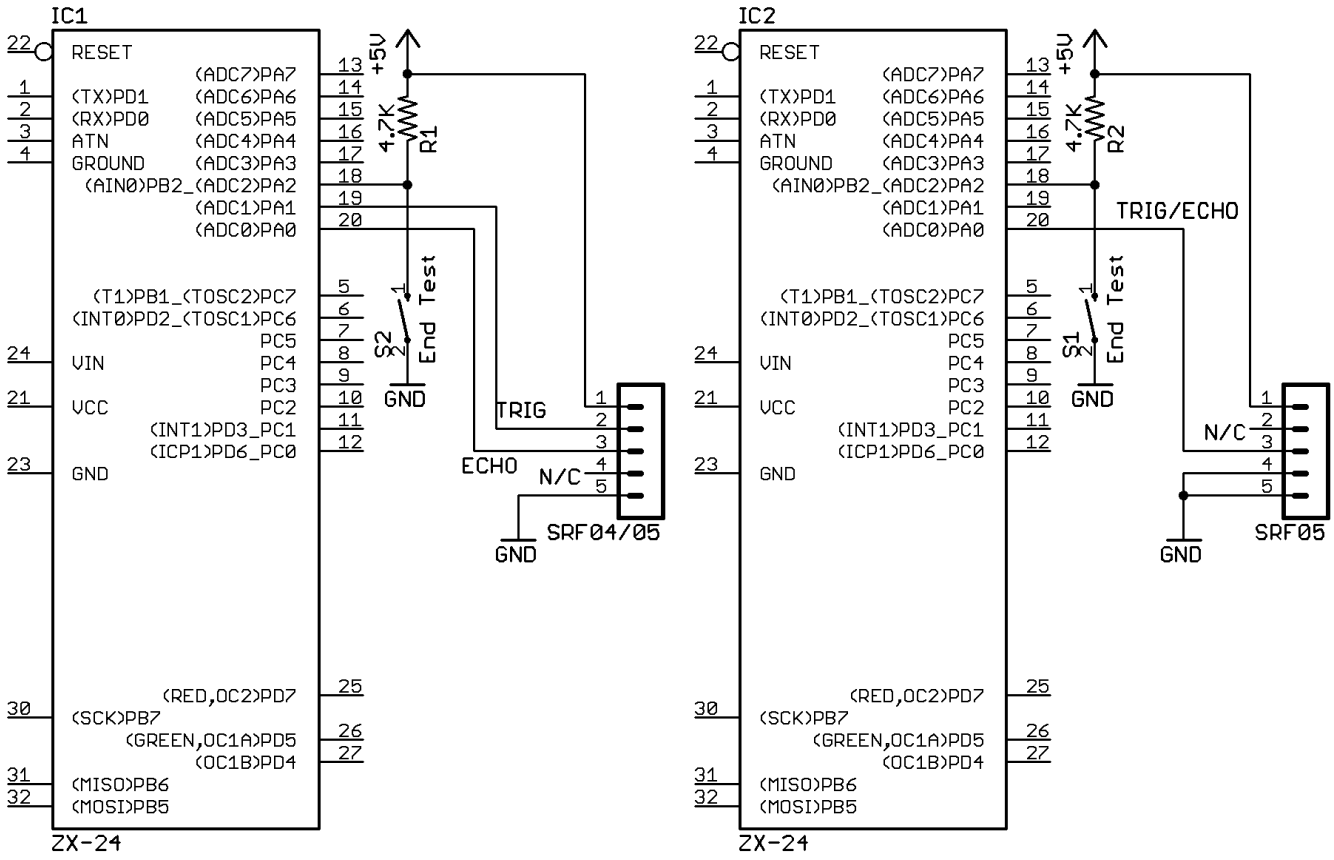
Introduction

This application note describes how to connect and use the Devantech SRF05 Ultrasonic Range Finder (http://www.robot-electronics.co.uk/htm/srf05tech.htm). Although not tested the software in the zip file with this application note should also work with the SRF04 Ultrasonic Range Finder as well.

Ultrasonic range finders such as the SRF05 are typically used in robots for collision avoidance or object detection. The transmitter transducer sends out a 40 KHz ultrasonic burst that is then reflected off any nearby objects and picked up by the receiver transducer. These range finders (SRF04/SRF05) require the microcontroller to calculate the distance to the object by timing how long it takes to receive an echo and dividing by the speed of sound in air. This makes them less expensive than the I2C-based SRF08 and SRF10 ultrasonic range finders described in Application Note AN-208 but slightly increases the burden on the hosting microcontroller.

Hardware Hookup

Connecting a ZX chip to the SRF04 and SRF05 Devantech Ultrasonic Range Finders is very simple as shown below for the ZX-24. The circuit on left which can be used by both the SRF04 and SRF05 shows separate I/O lines for triggering the ultrasonic pulse and receiving the echo. The circuit on the right which can only be used with the SRF05 reduces the number of I/O pins from two to one.



Software

This application note also comes with some ZBasic software. The file SRF05.bas is an interface module for the SRF04 and SRF05 and the file AN215.bas is a test program for the SRF05.bas module. The public interface implemented by the SRF05.bas module is very similar to the one provided in AN-208. This is a deliberate design

Published March 2006

decision as it is unlikely that any system will use both types of range finders. The public interface provided by SRF05.bas consists of some constants and public routines named `InitSRF()`, `TermSRF()`, and `GetSRFRange()`.

The `InitSRF()` subroutine shown below is used to gather the I/O pin information, the required measurement units, and start the `doRanging()` task (which is described later). Note that for the single I/O pin connection, simply make tPin the same as ePin as shown in AN215.bas example code.

```
Public Const SRF_UNITS_INCHES as Byte      = 80
Public Const SRF_UNITS_CM as Byte          = 81

Public Sub InitSRF(ByVal tPin as Byte, ByVal ePin as Byte, ByVal rangeUnits as Byte)
   trigPin = tPin
   echoPin = ePin
   units = rangeUnits
   range = 32767       ' set to maximum range
   CallTask "doRanging", rangeStack
End Sub
```

The `TermSRF()`subroutine is used to simply stop the `doRanging()` task started by `InitSRF()`. The source code can be found in the zip file associated with this application note.

Because ultrasonic ranging and waiting for any echos to completely dissipate takes a significant time (50 milliseconds), a fundamental design element of the SRF module is that the ranging is done in a separate task. This allows time-critical operations such as a robot control task to continue processing other sensor input and then request the distance from the SRF on demand. This is what the `GetSRFRange()` public function does and returns the distance in either inches or centimeters.

```
Public Function GetSRFRange() as UnsignedInteger
   ' convert to correct units
   If units = SRF_UNITS_CM Then
       GETSRFRange = CUInt(CSng(range) * CM_CONVERT_FACTOR)
   Else
       GETSRFRange = CUInt(CSng(range) * INCH_CONVERT_FACTOR)
   End If
End Function
```

The private variable named **range** holds the number of Timer1 ticks (1.085 us) for the last echo pulse received from the SRF04 or SRF05 device. The two factors for conversion are constants calculated from the speed of sound in air at sea level. You could adjust this speed for your altitude to get a more accurate reading.

```
Private Const TIMER1_RESOLUTION as Single = 1.085  ' 1.085 us
Private Const SPEED_SOUND as Single = 340.29       ' speed of sound in m/s at sea level

' conversion factors from Timer1 ticks to half distance in cm or inches
Private Const CM_CONVERT_FACTOR as Single = TIMER1_RESOLUTION * SPEED_SOUND / 20000.0
Private Const INCH_CONVERT_FACTOR as Single = CM_CONVERT_FACTOR / 2.54
```

The `doRanging()` subroutine initiates an ultrasonic pulse and times how long it takes to receive an echo. This process repeats every 50ms as shown in the code below

```
' Minimum length of trigger pulse must be 10us
Private Const TRIGGER_PULSE_LENGTH as Integer = CInt(10.0/TIMER1_RESOLUTION)
' Time to wait before starting next ranging
Private Const RANGING_WAIT_TIME as Single = 0.05

Private Sub doRanging()
   Do
       ' wait time between doing ranging
       Call Delay(RANGING_WAIT_TIME)
       ' lock task to do ranging
       Call LockTask()
       Call PulseOut(trigPin, TRIGGER_PULSE_LENGTH, 1)
```

```
        range = PulseIn(echoPin, 1)
        Call UnlockTask()
    Loop
End Sub
```

Note that the task is locked while waiting for the echo pulse to complete and Timer1 is utilized to measure the pulse width. Because the Timer1 is only needed for 30 ms for each ranging request, a possible enhancement is to share the use of Timer1 with other tasks by utilizing a semaphore.

Using the Range Finder

The example test program (AN215.bas) below shows how to invoke the interface to the SRF module described previously.

```
Private Const trigPin As Byte = A.0
Private Const echoPin As Byte = A.0 ' or A.1 for SRF04
Private Const stopPin As Byte = A.2

Sub Main()
    ' start test and initialize SRF
    Debug.Print "Start of SRF test"
    Call InitSRF(trigPin, echoPin, SRF_UNITS_INCHES)

    ' main test loop which gets the range in inches every second
    ' until the stop button is pressed
    Do While GetPin(stopPin) = 1
        Debug.print "Range ";CStr(GetSRFRange()); " inches"
        Call Sleep(1.0)
    Loop

    ' stop using the SRF
    Call TermSRF()
    Debug.Print "SRF test finished"
End Sub
```

Here is the console output from the above program showing how the range changes as a hand is moved over the SRF05 device until the stop test button is pressed. Although the range is only displayed every second, the LED on the SRF device is flashing approximately 10 times a second, once for each request from the `doRanging()` task.

```
ZBasic v1.1.5
Start of SRF test
Range 238 inches
Range 49 inches
Range 2 inches
Range 6 inches
Range 8 inches
SRF test finished
```

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.