Physics 335
Lab 7 - PIC Lab 3: Timing put to work—A sonic ranger device

This week, we'll work some more with our PIC and the on-chip counters. This time, we'll have you do a bit more of the coding itself and give you only a basic outline of what you need to do. The rest is up to you. Along the way, you'll use the pic's on-board counters and basic arithmetic functions.

Hardware

We'll use a simple sonic ranger, a device that sends out a short burst of ultrasonic pulses and then waits for the time for an echo of the pulse to be received. If you know the speed of sound, you can determine the distance between the sonic ranger and the object from which the reflected from. You may have used a similar device in introductory physics to measure the position of a small cart on a track. There are lots of uses for sonic rangers beyond the physics lab. For example, one could use one to measure distances between a car and other object (like concrete walls or other cars) to make a "backup safety measurement device" for a car.

If the time between the outgoing pulse and the received echo is Δt, and the speed of sound is v_s, the distance d would be given by

$$d = \frac{v_s \Delta t}{2} \ . \tag{1}$$

The factor of 2 comes in because the sound has to travel to and then back from the object being measured.

Question: The speed of sound in dry air at 20°C is 343 m/s. If the farthest distance that the sonic ranger can detect is about 400 cm, what would be the longest measurable echo time $\Delta t_{\max}$?

The device we will use is pretty trivial and works well with our PicDemo boards. The device has only 4 pins—and two of those are for power. A picture is shown below.

A complete data sheet is included with this write up at the end. Briefly, to use the device, a measurement is initiated by a positive going pulse on the trigger line. A short time later, the Echo line goes high and gives a pulse proportional to the distance between the sonic ranger and the nearest echo-able object.

7-1 Setting up the Sonic Ranger

We'll develop our project in steps. The first job is to simply get the sonic ranger wired up and pulsing.

The sonic ranger should be plugged into the white breadboard along the edge closest to the edge of the PicDem board, with the small cylinders facing outward, away from the board. If you hold the board so that you can read `PICDEM LAB`, the best place to put the ranger is in

the lower right corner of the white breadboard. (You will need the rest of the breadboard for the digital displays.)

Wire up the sonic ranger as follows:

- First, don't forget *power*. Pull `Vdd2` to the `Vdd` bus on the black rails and `Vss` to the `Vss` bus. Then wire `Vcc` sonic ranger power to the `Vdd` bus and `Gnd` to the `Vss` bus.
- Use `RA0` (pin 17) as an output on the PIC. This should be connected to the `Trig` pin on the sonic ranger.
- Use `RA1` (pin 18) as an input on the PIC. This should be connected to the `Echo` pin on the sonic ranger.

Make your wiring as clean and neat as possible. Use color coding, e.g., black and red for power, other colors for signals. Use the pliers to insert wires cleanly and without damage.

7-2 **Trigger the sonic ranger**

Download the template file `SonicRanger_template.asm` from the course website. This file contains a skeleton of code, along with the usual configuration settings. Also grab the `p16f88.inc` file that you use in all of your projects. Save these files somewhere convenient.

Then create a new project with the usual settings: a standalone project, using the PIC16F88, powered from the PICkit3 programmer. Attach the template file and the "include" file to the usual spots on the project tree.

Now, open up the template file and have a look. Scroll down to where you see `; Microcontroller initialization`. You will see a sequence of labels (`SetOSC`, `SetIO`, `SetTimer`, etc.). There are also many areas with instructions that are commented out—these will be used later.

Your first task will be to code the following sections:

`SetOSC` Use the oscillator control registers to select a 4MHz internal oscillator to run the PIC.

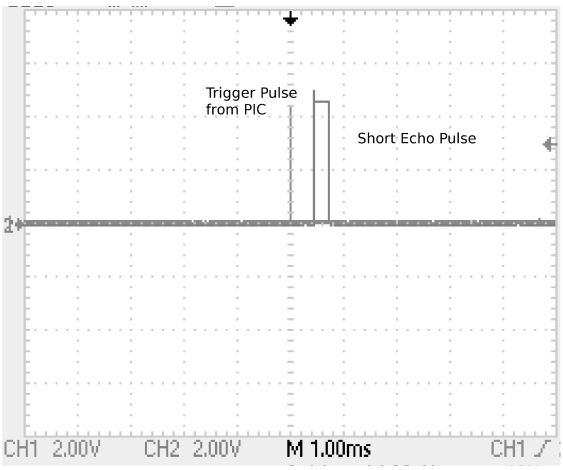
`SetIO` Use the `PORTA` and `TRISA` registers to set up `PORTA<0>` as an output and `PORTA<1>` as an input. Also, since you will need them later, set up all bits of `PORTB` as outputs.

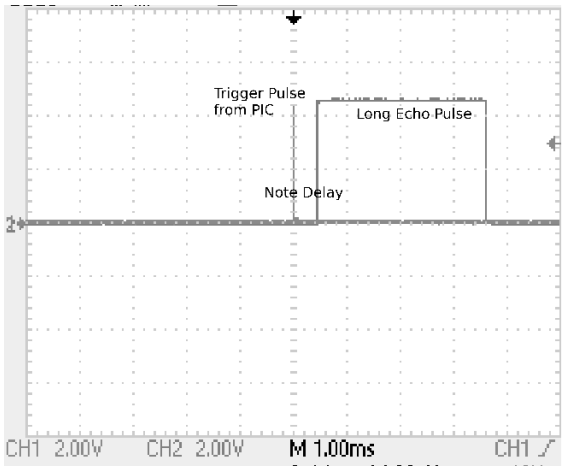
`Pulse` Write code here that sets `PORTA<0>` HI, waits 10–15 microseconds, and then sets `PORTA<0>` LO.

`Delay` Write code here that will waste time for about 60 milliseconds. Note where this code is called from.

After you code this up, test it, and confirm that you see a short pulse appear on the `RA0` pin every 60 ms or so.

Then hook up another scope probe and look at the `Echo` line. You should see a much longer pulse whose length will vary when you move your hand toward and away from the front of the sonic ranger. You should see something like the following:





Remember, it's a *sonic* pulse, so it will not be perfectly collimated coming out of the ranger. It will work better for hard "echoey" objects with flat surfaces (a book or cardboard or the flat of your hand) than soft, roundish sound-absorbing objects (a wad of Kleenex or a pillow for your pet hamster). Make sure there is nothing between the ranger and what you want it to "see."

Verify that you have the sonic ranger creating an appropriate echo pulse before proceeding, or you'll get nowhere from here on.

7-3 Using a counter to time the pulse

Next, you will need to configure your counter. You will have have to configure it in such a way that it counts sufficiently accurately for a two decimal place display by the end of the lab. This is just a range of 0 to 99, so an 8 bit counter is plenty (it goes to 255), but you'll still need to configure the counter so that it uses those bits well. You don't want to count too slowly, which will give poor resolution, or to quickly, lest you overflow. The two parameters that control the counting rate are the oscillator frequency and the timer prescaler. We recommend using the Timer0 module because it has a wide range of prescaler settings: you can prescale by any factor of 2 between 1:1 and 1:256.

The counter rate will determine the length units that your ranger will use. If you want your readout to measure in centimeters, for example, the time between successive counts should be the time it takes sound to travel *two* centimeters (i.e., there and back). Work this out, and then select the prescaler setting which will give the frequency closest to the inverse of this time interval.

After you determine your counter and prescaler values, you will be ready to code these sections; some may have only two or three lines:

`SetTimer` If using the Timer0 module, you will need to set the appropriate bits in `OPTION_REG`.

`EchoWait_and_Clear` Make a tight loop that simply waits until `PORTA<1>` goes HI, at which point the timer register (e.g., `TMR0`) is cleared.

`EndEcho_and_Read` Make another tight loop that simply waits until `PORTA<1>` goes LOW, at which point the timer register is copied into `W`.

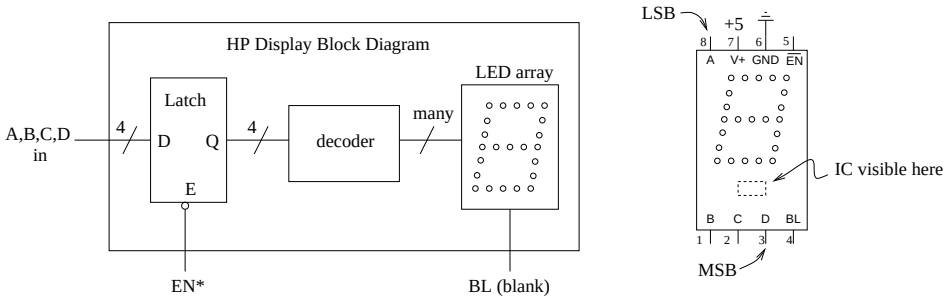
To see if this code is working, use the debugger to examine the contents of the variables `TimerCounts` or `bin` at the end of each iteration of the `MainLoop`. (Use a "breakpoint.") You

should see that the number is proportional to the distance between the ranger and whatever is in front of it.

7-4 **Add Displays**

Locate the pair of HP Hex digit displays that are on the special socket, and plug them into the white breadboard (you may have to push hard). Connect up the power to pins 7 (+5V) and 6 (Gnd), and ground both the ENABLE* and BLANKING pins (4 and 5). Connect pins `RB0` through `RB3` to the data lines on one display and `RB4` through `RB7` to the data pins on the other display. As you did with the sonic ranger, make your wring neat so that you can easily see the displays and have access to the PIC, etc., for troubleshooting.

A diagram of the HP display is given below.



In the assembly language file, uncomment the sections marked `CALL _bin2bcd` and `UpdateDisplay`. Also scroll down and uncomment all of the code under the labels `UpdateDisplay` and `Overflow`.

Then download the file `bin2bcd.inc`. This contains a routine that converts binary to binary-coded-decimal (BCD). The code that does this conversion is shown on the next page. It is am implementation of the so-called "double-dabble" algorithm.

To access this code in your program, put the `bin2bcd.inc` file into your project under "Source Files" and uncomment the line that says `#include bin2bcd.inc`

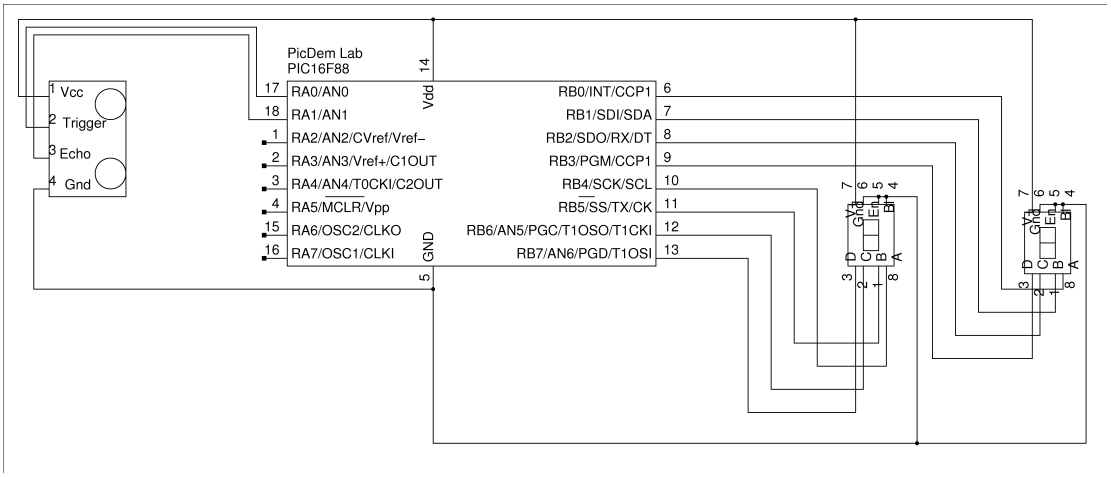
```
; 8-bit binary to BCD conversion
; pete griffiths 2007
; http://picprojects.org.uk/projects/pictips.htm
;
; bin contains the binary value to convert.
; Conversion process destroys contents
; Result is in bcdH, bcdL on return.
; Call _bin2bcd to perform conversion.

_bin2bcd    movlw     d'5'
            movwf     counter
            clrf      bcdL
            clrf      bcdH
; we can save some execution time by not
; doing the 'test and add +3'code for the
; first two shifts
            rlf       bin,F
            rlf       bcdL,F
            rlf       bin,F
            rlf       bcdL,F
            rlf       bin,F
            rlf       bcdL,F

_repeat     movfw     bcdL
            addlw     0x33
            movwf     temp
            movfw     bcdL
            btfsc     temp,3
            addlw     0x03
            btfsc     temp,7
            addlw     0x30
            movwf     bcdL
; we only need to do the test and add +3 for
; the low bcd variable since the
; largest binary value is 0xFF which is 255
; decimal so the high bcd byte variable will
; never be greater than 2.
            rlf       bin,F
            rlf       bcdL,F
            rlf       bcdH,F
            decfsz    counter,F
            goto      _repeat
            return
```

7-5 **Wrap it up**

The full circuit diagram for the sonic ranger is shown below



Test it out. You should see the numbers indicate roughly the centimeter distance between an object and the sonic ranger. When the object gets too far away, it should show `OF`, indicating an overflow.

Include the following in your report:

- Your full assembly program.
- a calibration check: use a meter stick to compare the sonic ranger's reported distance and the actual distance to some object. You should try it at a couple of distances. If you need to convert from "sonic ranger units" to centimeters, calculate a conversion constant.
- A description of how the code for `UpdateDisplay` works. This can be included in the comments inside your program. In particular, explain how the overflow checking and display of `OF` works.

HC-SR04 Ultrasonic Range Finder

Manual

Features

- Distance measurement range: 2cm - 400cm
- Accuracy: 0.3cm
- Detect angle: 15 degree
- Single +5V DC operation
- Current comsuption: 15mA

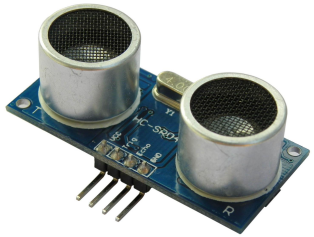


Fig. 1

How It Works

HC-SR04 consists of ultrasonic transmitter, receiver, and control circuits. When trigged it sends out a series of 40KHz ultrasonic pulses and receives echo from an object. The distance between the unit and the object is calculated by measuring the traveling time of sound and output it as the width of a TTL pulse.

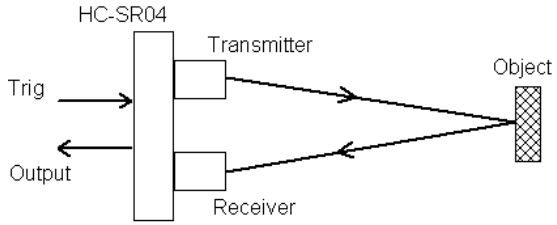


Fig. 2

How To Use It

To measure distance you need to generate a trig signal and drive it to the Trig Input pin. The trig signal leve must meet TTL level requirements (i.e. High level > 2.4V, low level < 0.8V) and its width must be greater than 10us. At the same time you need to monitor the Output pin by measuring the pulse width of output signal. The detected distance can be calculated by the formula below.

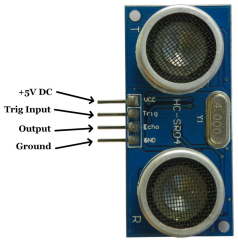


Fig.3

$$\text{Distance} = \frac{\text{Pulse Width} * \text{Sound Speed}}{2}$$

where the pulse width is in unit of second and sound speed is in unit of meter/second. Normally sound speed is 340m/s under room temperature.

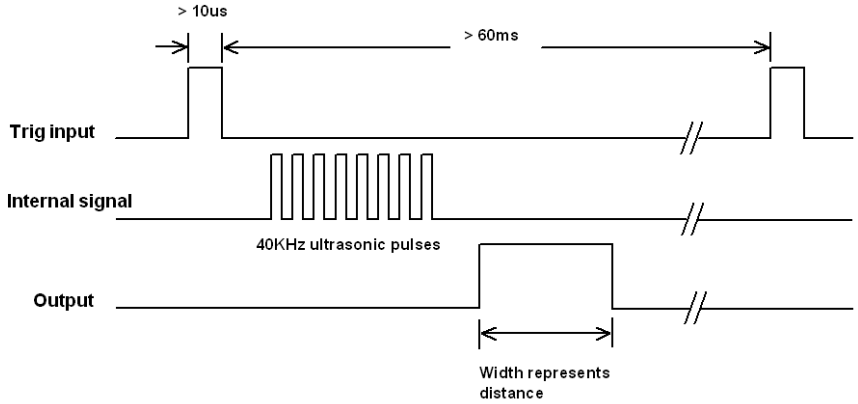


Fig. 4

Notes:
1. The width of trig signal must be greater than 10us
2. The repeat interval of trig signal should be greater than 60ms to avoid interference between connective measurements.

Specifications

Parameters	Specification
Operating Voltage	+5V DC
Operating Current	15mA
Operating Frequency	40KHz
Maximum Distance	400cm
Minimum Distance	2cm
Detect Angle	15 degree
Resolution	0.3cm
Input Trig Signal	>10us TTL pulse
Output Signal	TTL pulse with width representing distance
Weight	
Dimension	45 x 20 x 15 mm