# Chapter 4: Pulse Width Modulation

PULSES FOR COMMUNICATION AND CONTROL

Pulse width modulation is abbreviated PWM, and it refers to a technique of varying the amount of time high and/or low times in a signal last. Pulse width modulation is widely used for both motor control and communication. This chapter's activities include PWM examples for both motor control and communication, and it also examines how to utilize the PropScope's features to measure these signals.

4

ACTIVITY #1: PULSES FOR SERVO CONTROL

Hobby servos like the one in Figure 4-1 control the steering and throttle in radio controlled cars, boats and airplanes. They are also useful for a variety of robotic tasks, including moving arms, legs and grippers. The servo can rotate its 4-point star shaped horn (indicated by 3 in the figure) to various positions. More importantly, the servo can hold its horn in a given position and resist external forces that might try to displace it. For example, in a radio controlled car, the servo has to make the wheels turn left and right for steering. The servo has to hold the wheels in a certain position, or they would just straighten out and the car would go forward again. Likewise with a boat rudder, it has to push against the water flowing around it to make a radio controlled boat turn in the water.
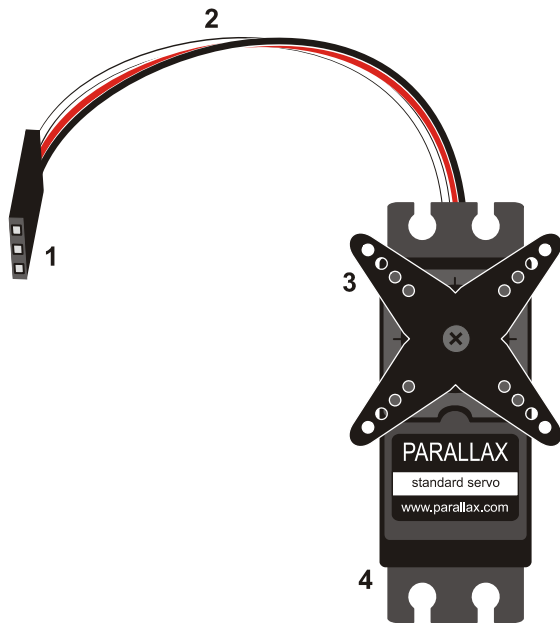
Figure 4-1
Servo

1. Plug
2. Cable
3. Horn
4. Mounting Flange

Driving a radio controlled vehicle involves manipulating joysticks on a radio transmitter. As the joysticks are moved to certain positions, the servos in the vehicle rotate their horns to positions that mirror the joystick positions. For example, one joystick moved left and right in the radio control unit, it might make the servo that controls the steering in a radio controlled car rotate left and right. Also, if the joystick is held in a certain position, the servo holds its horn at the position the joystick makes it hold.

The radio transmitters in these systems codes the joystick's position into bursts of radio activity. The durations of the bursts indicate the joystick positions. On the RC vehicle, a radio receiver converts theses bursts of radio activity into binary high/low signals. The signal is high while the radio bursts last, and low when there is no radio activity. The brief high signals are called pulses, and the radio receiver sends these binary pulses to the various servos in the vehicle to control them.

The process of adjusting pulse durations that get sent to a servo to control its horn position is an example of pulse width modulation. In this activity, you will program the BASIC Stamp to transmit PWM control signals to a servo to dictate its horn's position.

You will then measure and examine the PWM signal's pulse durations with the PropScope.

> **i** **Servo Tutorial**. There's lots more information about servos and activities that introduce a variety of servo control techniques in What's a Microcontroller, Chapter 4.

4

Servo Test Parts

(1) Parallax Standard Servo
(1) 3-pin header (HomeWork Board only)
(Misc) Wires

Servo Test Circuit

Figure 4-2 shows a schematic of the servo and PropScope probe connections. The servo's black and red wires provide power to the small DC motor and electronics inside the servo. The white wire conveys the PWM signal to control system electronics inside the servo. That white wire is connected to BASIC Stamp I/O pin P14, and the BASIC Stamp will be responsible for using that pin to send PWM control signals to the servo.

Vdd

PropScope CH1

White

P14

Red   Servo

Black

PropScope GND

Vss
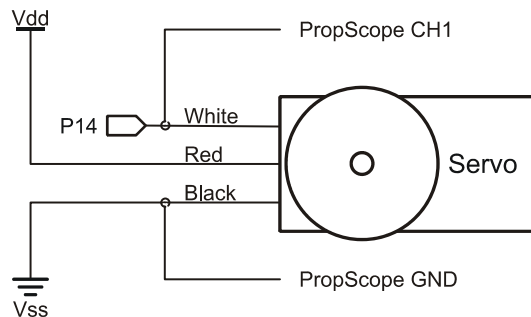
Figure 4-2
Servo Schematic

The instructions for connecting the circuit depend on your board and revision.

- ✓ If you do not already know which board and revision you have:
  - o Open the BASIC Stamp Editor Help (v2.5 or newer).
  - o Click the Getting Started with Stamps in Class link on the Help's home page.
  - o Follow the directions to determine which board you have.

✓ Skip to the section in this book that covers your board:
  o Board of Education (Serial Rev C and Newer, USB all Revs). The instructions for this board start right after this checklist.
  o BASIC Stamp HomeWork Board (Serial Rev C and newer) starts on page 131.
  o All other boards: go to www.parallax.com/Go/WAM → Servo Circuit Connections to find servo circuit instructions for your board.
✓ When you are done with the servo circuit instructions for your board, go to PWM Signals for Servo Control section; it starts on page 133.

Board of Education (Serial Rev C and Newer, USB all Revs)

There's a jumper that shorts two of three pins together between the X4 and X5 servo ports. In Figure 4-3, the jumper is shorting the upper two pins to connect the servo supply to Vdd, regulated 5 V. We will use that jumper setting in this activity.

✓ Move your board's 3-position switch left to position-0.
✓ Verify the jumper is set to Vdd. It should cover the two pins closer to the Vdd label, leaving the lowest of the three pins just above the Vin label exposed.
✓ If the jumper is instead in the lower position and the Vdd pin is exposed, lift the jumper upward to pull it off the pins. Then place it as shown in Figure 4-3.
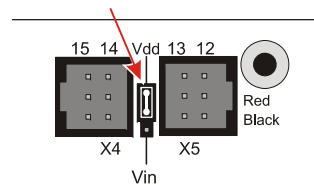
Figure 4-3
Power Jumper Between Servo Headers

Servo port 14 in the X4 header connects P14 to the servo's white signal ~~line. So~~, you can use the P14 socket next to the breadboard to probe the control signals.

✓ Plug the servo into servo port 14 as shown in Figure 4-4.
✓ Make sure that the color coded servo cable wires line up with the White, Red, Black labels shown in Figure 4-4.
✓ Connect the PropScope CH1 probe to P14 and the probe's ground clip to Vss.
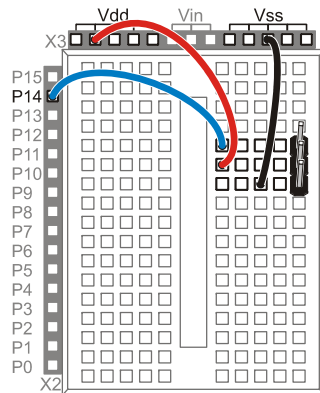✓ Skip to the PWM Signals for Servo Control section; it starts on page 133
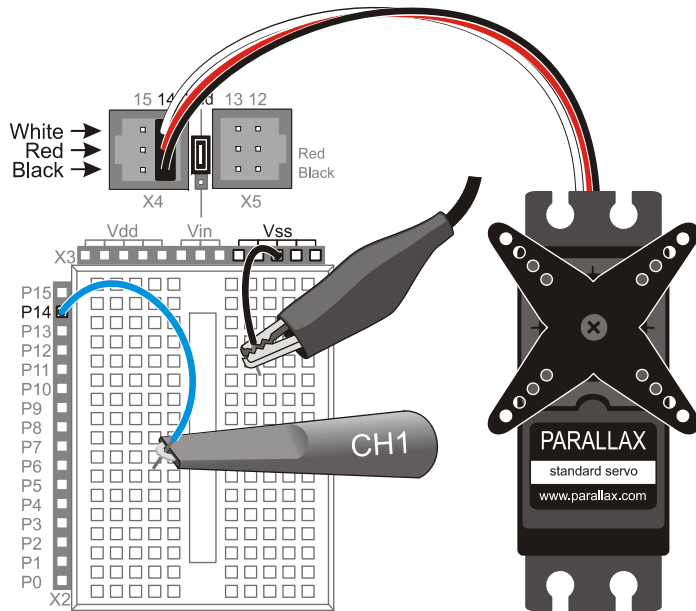
Figure 4-4
Wiring Diagram for
Figure 4-2

*with the Board of
Education.*

BASIC Stamp HomeWork Board (Serial Rev C and newer)

✓ Disconnect power from your board.
✓ Build the servo port Figure 4-5.

Figure 4-5
BASIC Stamp
HomeWork Board Servo
Port Wiring

✓ Connect the servo to the port as shown in Figure 4-6.

✓ Make sure the color coded servo cable wires line up with the labels in Figure 4-6.
✓ Connect the PropScope CH1 probe to the breadboard row that the servo cable's white wire plugs into.
✓ Connect the ground clip to the row that the servo cable's black wire plugs into.

Figure 4-6: Wiring Diagram for Figure 4-2 *with the BASIC Stamp HomeWork Board*
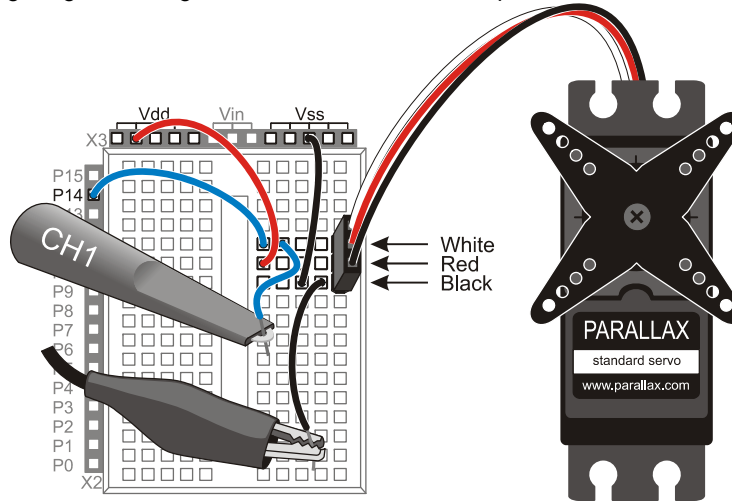
PWM Signals for Servo Control

The Parallax Standard Servo's horn has a 180° range of motion. Figure 4-7 shows timing diagrams for example PWM signals that make the servo hold its horn at 45°, 90° and 135° positions. The pulse high times control the servo's position, and they have to be precise. PBASIC has a command called PULSOUT for making the BASIC Stamp deliver precisely timed pulses. The pulses also have to be sent repeatedly to make the servo maintain a certain position. Repeating the pulses at 50 Hz is ideal, but the rate the pulses are delivered does not have to be precise –a few Hz faster or slower is fine too. Most PBASIC examples use a simple PAUSE 20 inside a loop with a PULSOUT to make the BASIC Stamp send control pulses that repeat at a rate in the 44 Hz neighborhood.
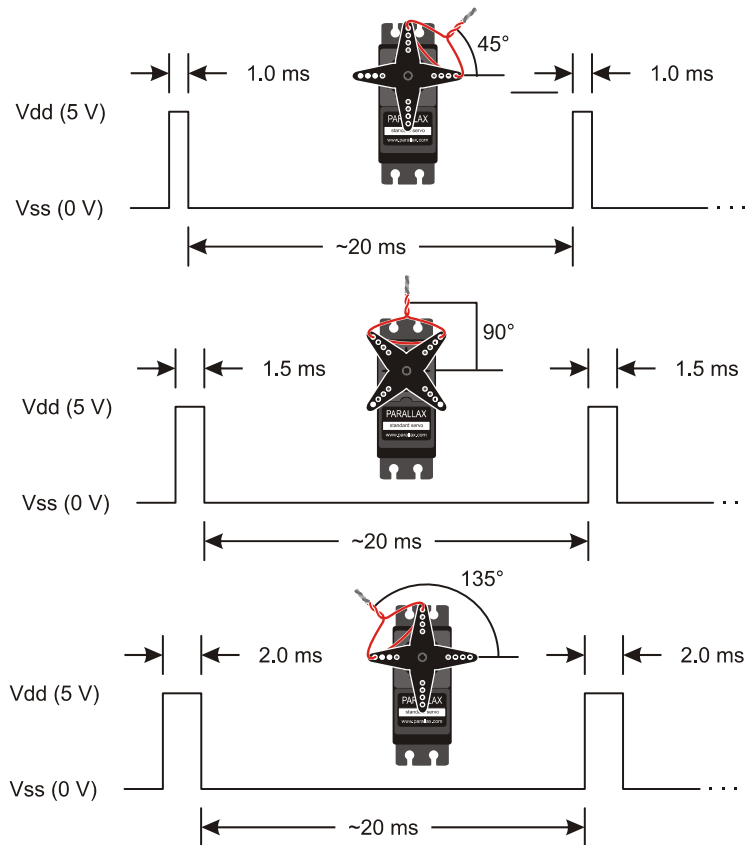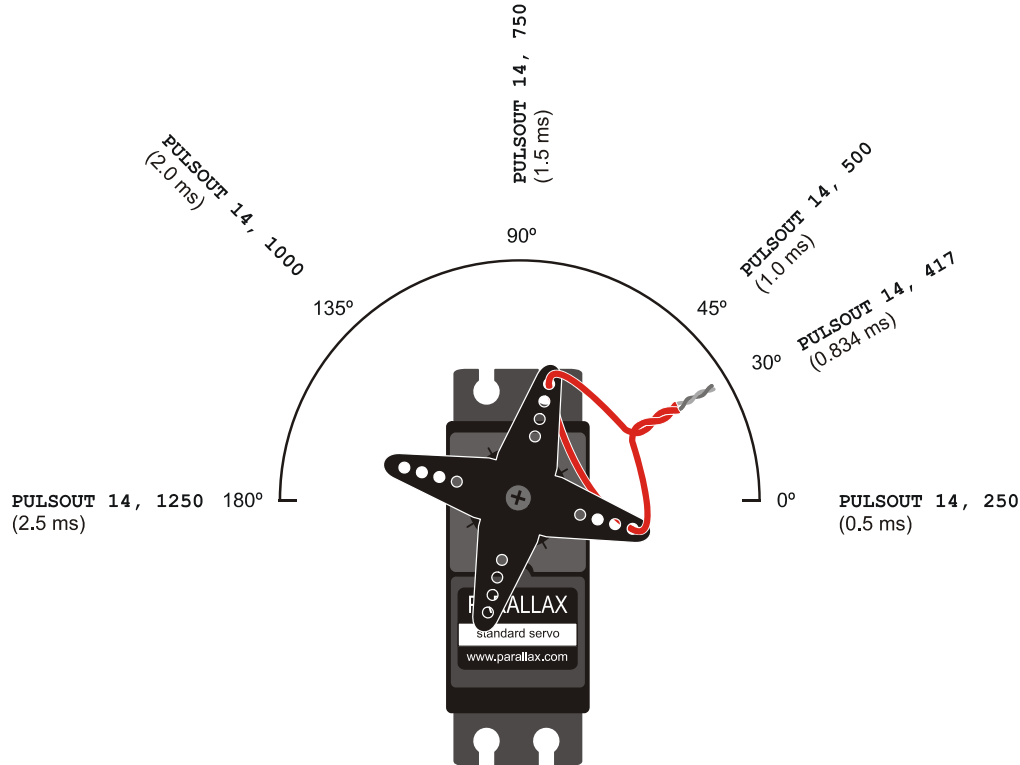

Figure 4-7
Pulse Width vs.
Servo Horn Angle

(Excerpt from What's a Microcontroller v3.0)

Figure 4-7 shows just three servo horn positions in a range of motion that spans from 0° to 180°. Figure 4-8 shows a map of the servo's range of horn positions with some example pulse durations that will make the servo point in certain directions. You can use this map to make the servo point to an angle of your choosing. For example, you could choose to make the servo point to the 30° position by sending 0.834 ms pulses. That pulse duration was calculated using the fact that 30° is 30/45ths between the pulse durations for 0° and 45°. The corresponding PULSOUT Pin Duration command for that angle is PULSOUT 14, 417. That command tells the BASIC Stamp to transmit a (417 × 2 μs ≈ 0.834 ms) pulse using I/O pin P14. For other PULSOUT duration arguments, just divide 2 μs into the millisecond pulse duration, round to the nearest integer, and use that result for the Duration argument.

Figure 4-8: Pulse Width vs. Servo Horn Angle *(Excerpt from What's a Microcontroller v3.0)*

> (i) **The twist tied wire needs to be adjusted.** A factory mounted servo horn does not necessarily make the wire point in the exact directions shown in Figure 4-8. The horn typically has to be removed, rotated slightly and replaced before it'll point a twist tied wire just like it does in Figure 4-8. What's a Microcontroller, Chapter 4, Activity #2 explains the procedure in detail.
>
> **The angle vs. pulse time values in Figure 4-8 are approximate.** More precise positioning may require some experimentation.
>
> **Keep the PULSOUT Duration arguments in the 350 to 1150 range.** PULSOUT command Durations near the 250 or 1250 limits could cause the servo to push against its built-in mechanical limits. There are also tips on finding the PULSOUT Duration values just inside these limits in What's a Microcontroller, Chapter 4.

4

Center Position Test Program

ServoCenter.bs2 positions the horn at the 90-degree mark in Figure 4-8 and holds it there. Your servo horn will probably not point the twist tied jumper wire straight up in the 90° direction. That's okay, whatever position the loop points while the program is running; you can consider that 90°. Another option is to remove the screw that attaches the servo horn, pull it off the output shaft, rotate it slightly, then push it back on to correct the error. Make sure to push the horn back onto the output shaft while the program is running. That way, it'll be as close as possible to the 90° mark.

- ✓ Reconnect power to your board.
- ✓ If you have a Board of Education, make sure to move its power switch to position-2, so that it supplies power to the servo ports.
- ✓ Enter ServoCenter.bs2 into your BASIC Stamp Editor and run it.

```
' What's a Microcontroller - ServoCenter.bs2
' Hold the servo in its 90 degree center position.

' {$STAMP BS2}
' {$PBASIC 2.5}

PAUSE 1000
DEBUG "Program Running!", CR

DO
  PULSOUT 14, 750
  PAUSE 20
LOOP
```

While the program is running, the servo should resist light twisting force applied to the horn to maintain its position.

- ✓ Apply light twisting force to the servo horn with only your fingertips. DO NOT FORCE IT!
- ✓ The servo should prevent the horn from turning, and the motor inside should hum a little bit as it resists and holds the correct position.

In the ServoCenter.bs2 program, the DO…LOOP repeats the PULSOUT and PAUSE commands indefinitely. As mentioned earlier, the Duration argument in PULSOUT Pin, Duration sets the pulse's duration in terms of 2 μs increments. So, the PULSOUT command sends a pulse (brief high signal) to P14 that lasts 750 × 2μs units. That's 750 × 2×10^{-6} s = 1500 × 10^{-6} s = 1.5 × 10^{-3} s = 1.5 ms. After the pulse is done, the PAUSE 20 command delays for 20 ms. Then, the DO…LOOP repeats, beginning with another pulse, then another pause, then another pulse, then another pause, and so on…

Keep in mind that different PULSOUT Duration arguments will result in the servo holding its horn in different positions.

> **(i)** **Positive vs. Negative Pulses:** Servo pulses are considered positive pulses because they spend a certain amount of time as high signals before returning to low. A negative pulse would start high, and spend a certain amount of time low. For negative pulses, simply set the I/O pin to output-high (like with the HIGH command) before the PULSOUT command, and it will send a low pulse instead of a high pulse.

Examine the Servo Control Signals with the PropScope

Back to the two cycles rule of thumb for a first look at any given signal in the oscilloscope. We know from the code and timing diagrams that each repetition of the DO…LOOP in ServoCenter.bs2 takes 20 ms + 1.5 ms + maybe 1 ms of code overhead ≈ 22.5 ms. Then multiply by 2 with a result of about 45 ms for two servo pulse signal cycles. Next, remember that the Horizontal dial selects the time for one division and that the Oscilloscope screen has 10 divisions. So, divide 45 ms by 10, and the result is the 4.5 ms, which is close to 5 ms. So select the 5 ms/div Horizontal setting in Figure 4-9.

- ✓ With your BASIC Stamp running ServoCenter.bs2, adjust the Horizontal and Vertical dials, coupling switches and Trigger tab settings so that they match Figure 4-9.
- ✓ Use your Trigger Time Control position the vertical trigger crosshair at the 2nd time division line.

✓ ~~Check the Oscilloscope preview bar, there are three more pulses off screen, and they continue to repeat to keep the servo holding its horn's position.~~

Figure 4-9: Servo Control Pulses
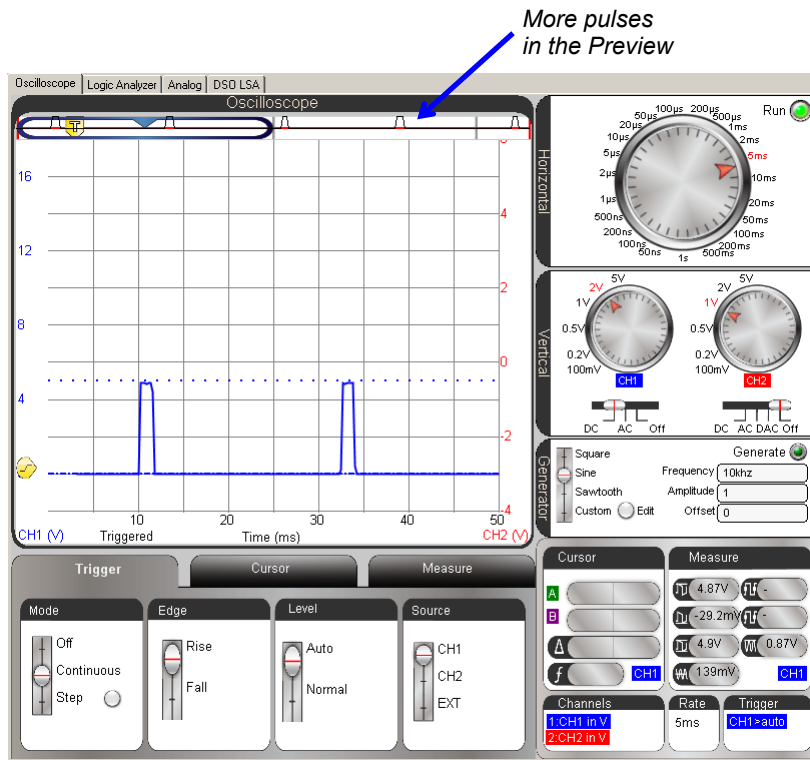
More pulses in the Preview

With this initial view of the servo pulses, let's check to see if the PWM signal does anything interesting with different PULSOUT Duration values that make the servo hold different positions. Figure 4-10 shows traces for PULSOUT 14, 500, PULSOUT 14, 750, and PULSOUT 14, 1000. With the larger PULSOUT Duration arguments, the period of the signal gets a little longer because the PAUSE in the program does not get any shorter to compensate for the longer pulse durations. So the frequency of the pulses drops slightly with larger PULSOUT Duration arguments. As mentioned earlier, the pulse durations have to be precise, but the frequency of the pulses just has to be in the neighborhood of 50 Hz. So small differences, like the ones caused by different pulse durations, do not affect the servo's performance.

✓ Change the PULSOUT command in ServoCenter.bs2 so that it reads PULSOUT 14, 500.

✓ Load the modified program into the BASIC Stamp.

✓ Note the width of the pulses.

✓ OPTIONAL: Use the cursors to measure the signal's period and frequency at each pulse width. Remember, a frequency of 50 Hz is ideal, but most servos are fairly forgiving, and frequencies in the low 40 Hz range will work fine too.

✓ Repeat for PULSOUT Duration arguments of 750 and 1000.

✓ Each time you load a modified program with an increased PULSOUT Duration, verify that the pulse width (the amount of time the pulse stays high) increases.

Figure 4-10: Different Servo Control Pulses, Same 20 ms PAUSE

PULSOUT 14, 500
(Servo horn to 45°)

PULSOUT 14, 750
(Servo horn to 90°)

PULSOUT 14, 1000
(Servo horn to 135°)
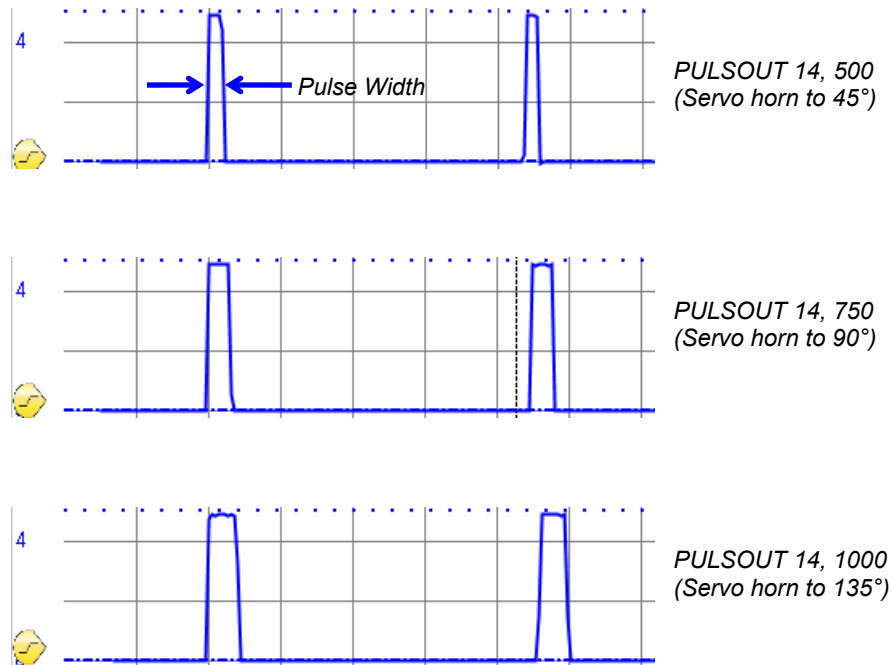
For more precise pulse time measurements, the Horizontal dial can be adjusted to a lower value that makes the pulse fill more of the Oscilloscope screen. Figure 4-11 shows an example with the Horizontal dial set to 500 µs/division. The Vertical Cursors are placed

at the pulse's rising and falling edges. (They are also called positive and negative edges.)
This particular pulse width measurement is with PULSOUT 14, 1000.

- ✓ Zoom in on the time scale by setting the Horizontal dial to 500 μs/div.
- ✓ Verify your 2 ms measurement with PULSOUT 14, 1000.
- ✓ Change the PULSOUT command to PULSOUT 14, 1050 and load the modified program.
- ✓ Watch the horn carefully as you load the modified program into the BASIC Stamp. It should take a new position slightly counterclockwise of the position it held with PULSOUT 14, 1000. If you didn't notice the change, try running the PULSOUT 14, 1000 version again, then switch back to the PULSOUT 14, 1050 version.
- ✓ Check the new pulse duration in the Oscilloscope.
- ✓ Calculate the pulse width for PULSOUT 14, 1050, and compare it to your measurement.

Figure 4-11: Cursors for Pulse Width Measurement
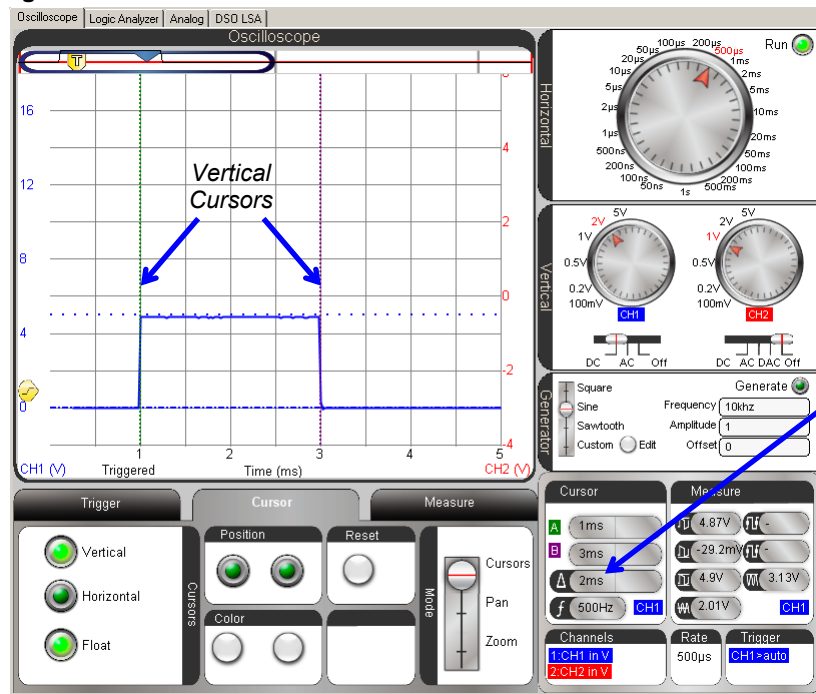
Your Turn: Modulated Pulse Width

Here is a program that sweeps the servo's horn back and forth by gradually increasing and then decreasing the pulse width over time.

- ✓ Examine the program and try to predict what you would expect to see in the Figure 4-11 display.
- ✓ Run the program and test your predictions.
- ✓ If the display's behavior does not match your predictions, study what it does, then study the program and try to figure out what's happening.

```
' What's a Microcontroller - ServoVelocities.bs2
' Rotate the servo counterclockwise slowly, then clockwise rapidly.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter          VAR     Word

PAUSE 1000

DO
  DEBUG "Pulse width increment by 8", CR

  FOR counter = 500 TO 1000 STEP 8
    PULSOUT 14, counter
    PAUSE 7
    DEBUG DEC5 counter, CR, CRSRUP
  NEXT

  DEBUG CR, "Pulse width decrement by 20", CR

  FOR counter = 1000 TO 500 STEP 20
    PULSOUT 14, counter
    PAUSE 7
    DEBUG DEC5 counter, CR, CRSRUP
  NEXT

  DEBUG CR, "Repeat", CR
LOOP
```

> ℹ **For more information about this program,** see What's a Microcontroller v3.0, Chapter 4, Activity #5..

ACTIVITY #2: DUTY CYCLE IN PWM DAC

A more descriptive name for the PWM command would be DCM for duty cycle modulation. Duty cycle is a measurement of the ratio of a binary signal's high time to its cycle time. When you set the PWM command's Duty argument to a value, you are specifying the number of 256ths of its cycle time that the signal stays high. In this activity, you will test to verify this by measuring and calculating the PWM signal's duty cycle for different PWM command Duty arguments.

4

DAC Test Parts

(1) Resistor – 1 kΩ (brown-black-red)
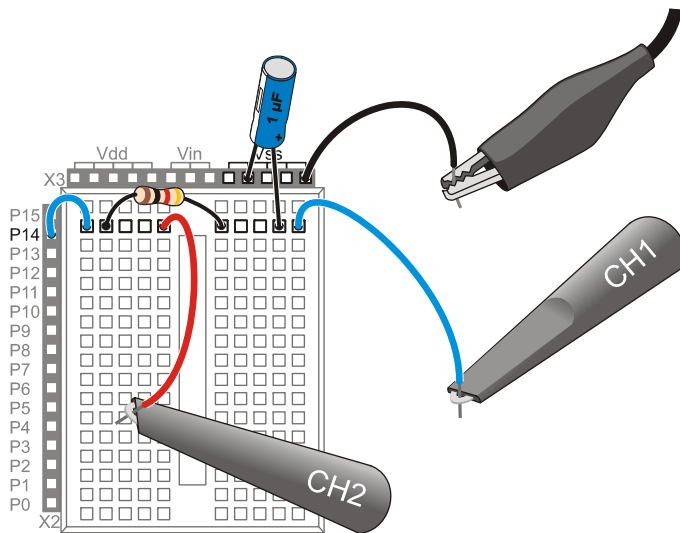(1) Capacitor – 1 µF
(misc.) Jumper wires

DAC Test Circuit

The circuit in Figure 4-12 is one of the two DAC circuits that were used to set DC voltages in Chapter 2, Activity #3. This circuit was also used to make the PropScope plot a simulated heart monitor display in Chapter 3, Activity #4.

  ✓   Build the circuit shown in Figure 4-12 and Figure 4-13.
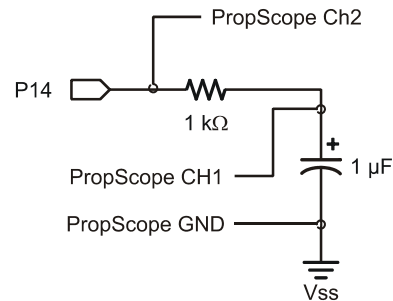
Figure 4-12
Probes for P14 PWM
Signal and DAC Output

Figure 4-13

Example Wiring
Diagram for Figure 4-12

DAC Test Program

This program uses the same PWM 14, 64, 1 to set the voltage across the capacitor to 1.25 V, like it did in Chapter 2, Activity #3. Instead of focusing on the voltage the PWM command sets across the capacitor, we'll look at the binary signal the BASIC Stamp transmits that causes the capacitor to charge to that voltage.

✓ Enter Test 1 Channel Dac.bs2 into the BASIC Stamp Editor and run it.

```
' Test 1 Channel Dac.bs2
' Set capacitor voltage in P14 RC DAC circuit to 1.25 V.

' {$STAMP BS2}                                ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                               ' Language = PBASIC 2.5

DEBUG "Program running..."                    ' Debug Terminal message

DO                                            ' Main Loop

    PWM 14, 64, 1                             ' 1.25 V to P14 capacitor

LOOP                                          ' Repeat main loop
```

4

PWM Command Duty Cycle Measurements

In Figure 4-14, channel 1 displays the DC voltage across the capacitor. That can be considered the DAC circuit's output signal. Channel 2 displays the PWM signal the BASIC Stamp applies the DAC circuit's input. This PWM signal has a duty cycle that determines the capacitor's voltage. Duty cycle is the ratio of a signal's high time to its cycle time, and it is commonly expressed as a percent measurement:

$$Duty\,Cycle = \frac{t_{HIGH}}{t_{CYCLE}} \times 100\%$$

Figure 4-14 shows the vertical time cursors placed for the t_{HIGH} measurement.

✓ Configure your PropScope's Horizontal dial, Vertical dials and Coupling switches, and Trigger tab settings as shown in Figure 4-14.

✓ Click the Cursor tab, and turn on the vertical time cursors by clicking the Vertical button.

✓ Align the vertical time cursors with the pulse's positive and negative edges.

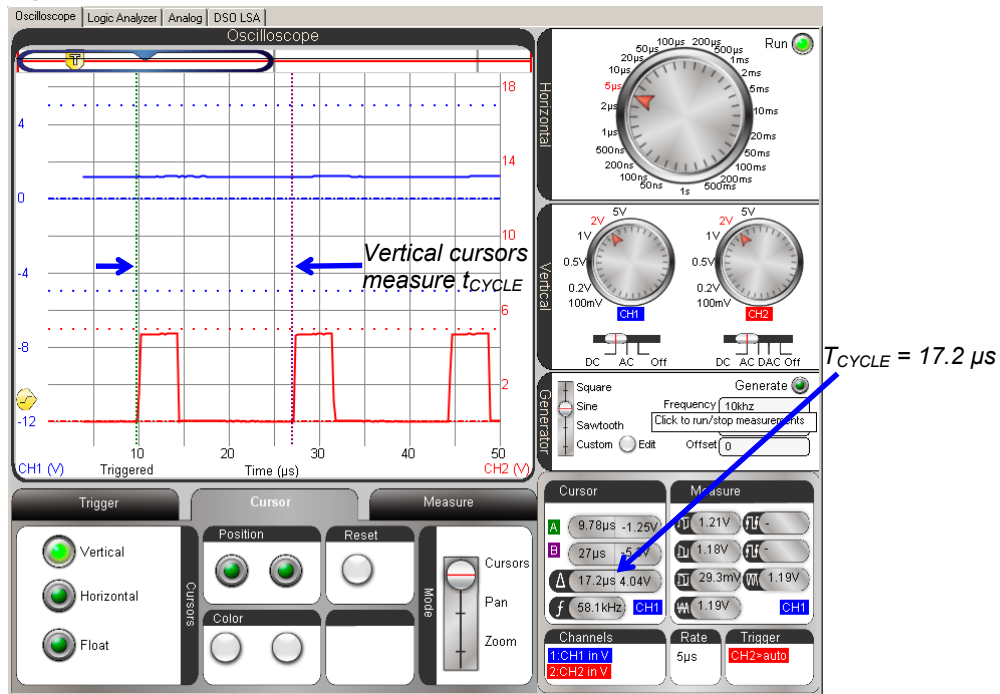✓ Make a note of your t_{HIGH} measurement.

Figure 4-14: t_{HIGH} Measurement with Cursors
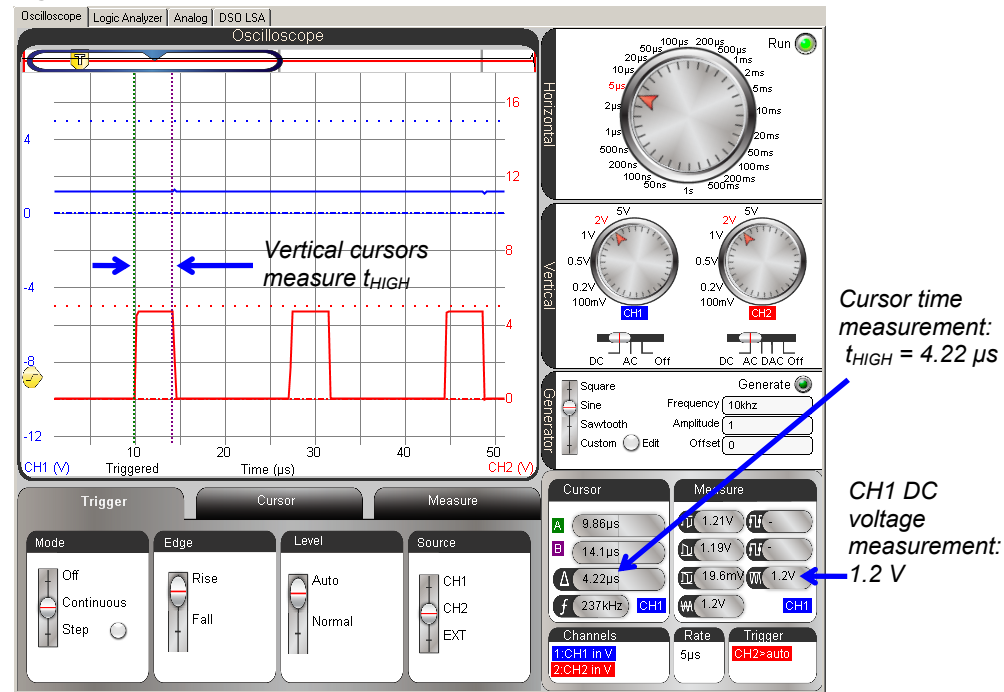
Cycle time (t_{CYCLE}) is the signal's period, and Figure 4-15 shows the time cursors aligned with successive rising signal edges for this measurement.

✓ Use your cursors to get your PWM signal's cycle time - t_{CYCLE}.

Figure 4-15: t_{CYCLE} Measurement with Cursors

With t_{HIGH} and t_{CYCLE}, the duty cycle calculation for this signal is:

$$Duty\ Cycle = \frac{t_{HIGH}}{t_{CYCLE}} \times 100\%$$

$$= \frac{4.22\ \mu s}{17.2\ \mu s} \times 100\%$$

$$\approx 24.5\%$$

Duty cycle simplifies calculations for the DAC output. Just multiply the percentage by the high signal level for a prediction of the DAC output.

$$V_{DAC} = Duty\,Cycle \times 5V$$
$$\approx 24.5\% \times 5V$$
$$= 1.23\,V$$

The average voltage measurements in Figure 4-14 and Figure 4-15 is oscillating between 1.19 V and 1.20 V, so 1.23 V is a reasonably good prediction.

4

Comment [AL54]:

To-do:

Update to match final screen captures. Final screen captures should use final software revision.

Also, update Duty Cycle Calculations with time measurements from the new screen captures.

Your Turn: Try a New Value

The command PWM 14, 64, 1 is probably getting a little old.

- ✓ Pick a target voltage in the 0 to 4.98 V range.
- ✓ Calculate the number of 256ths of 5 V that voltage represents for the PWM command's Duty argument.
- ✓ Modify Test 1 Channel Dac.bs2 accordingly and load the modified program into the BASIC Stamp.
- ✓ Check the CH1 average voltage in the Measure display.
- ✓ Measure and calculate the duty cycle, multiply it by 5 V, and compare to the CH1 DAC output voltage.

ACTIVITY #3: INFRARED OBJECT DETECTION

The Parallax Boe-Bot® Robot in Figure 4-16 uses infrared light emitting diodes (IR LEDs) as small flashlights and infrared detectors as eyes to check for objects in its path. This detection scheme involves just a few inexpensive electronic parts found in common appliances.

Figure 4-16
Boe-Bot® Robot

The IR LED in the Understanding Signals kit is the kind you might find on the end of a TV remote that you point at the TV to change the channel, adjust the volume, etc. The light it emits is not in the visible spectrum, but it's the light that sends signals to the TV. Likewise, the IR detector is one you might find in a TV that receives and demodulates the remote's infrared messages for the microcontroller in the TV. The IR detector is designed to send a low signal if it detects infrared flashing on/off at 38.5 kHz. Otherwise, it sends a high signal. In this activity, you will program the BASIC stamp to send a brief 38.5 kHz signal through an IR LED and observe the resulting low pulse at the IR detector's output.
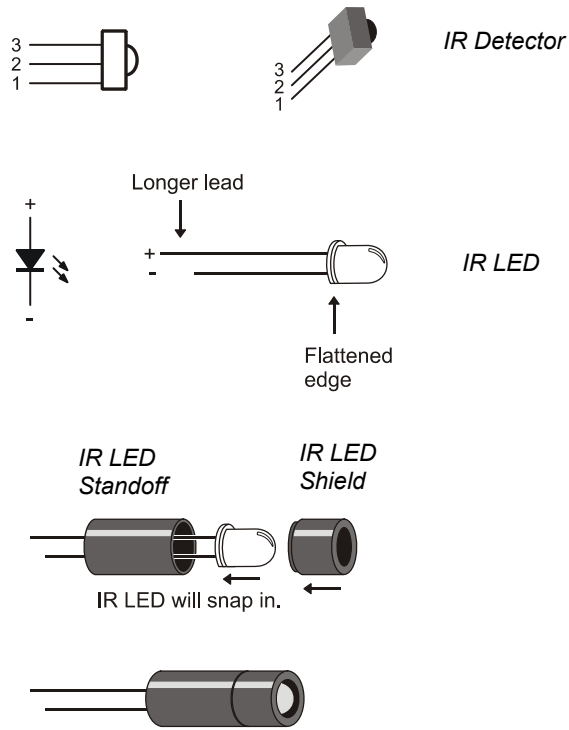
IR Object Detection Parts List

For reference, and to make it easier finding the parts in the kit, Figure 4-17 shows drawings of the IR LED, standoff, shield, and detector.

(1) Resistor – 220 Ω (red-red-brown)
(1) Resistor – 1 kΩ (brown-black-red)
(1) IR LED
(1) IR LED Shield
(1) IR LED Standoff
(1) IR detector

Figure 4-17 also shows how to house the IR led in the standoff and shield.

✓ Snap the IR LED into the standoff and then place the IR shield on top using Figure 4-17 as a guide.

4

IR Detector

Longer lead

IR LED

Flattened edge

Figure 4-17
IR Detector, LED,
Standoff and Shield

IR LED
Standoff

IR LED
Shield

IR LED will snap in.

IR Object Detection Circuit

Figure 4-18 shows a schematic of the infrared detection circuit, and Figure 4-19 shows an example of the circuit in a wiring diagram. With the IR LED in the standoff and shield housing, it directs the IR light more like a flashlight. Without the housing, it's more like a beacon.

The IR detector only reports that it "sees" IR light that has a frequency component in the 38.5 kHz neighborhood. Infrared from other sources such as a nearby window or indoor lighting normally have little to no effect.

✓ Build the circuit shown in Figure 4-18 and Figure 4-19.

✓ Check your work against Figure 4-19 to make sure you connected the IR LED's shorter cathode pin to the right row in the breadboard. The cathode pin should be connected to a row that is connected to Vss.

✓ If your board is sitting on a table, point the IR LED upward at an angle of about 30° so that the system does not report the table's reflection as a detected object.

✓ Make sure other objects are out of the IR LED's beam path so that they won't cause false detections. The maximum detection range with a 1 kΩ series resistor is usually in the 30 to 40 cm neighborhood.

✓ Make sure to get the probes out of the way of the IR LED's beam path so that they don't end up reflecting IR and become "objects" that are inadvertently detected.
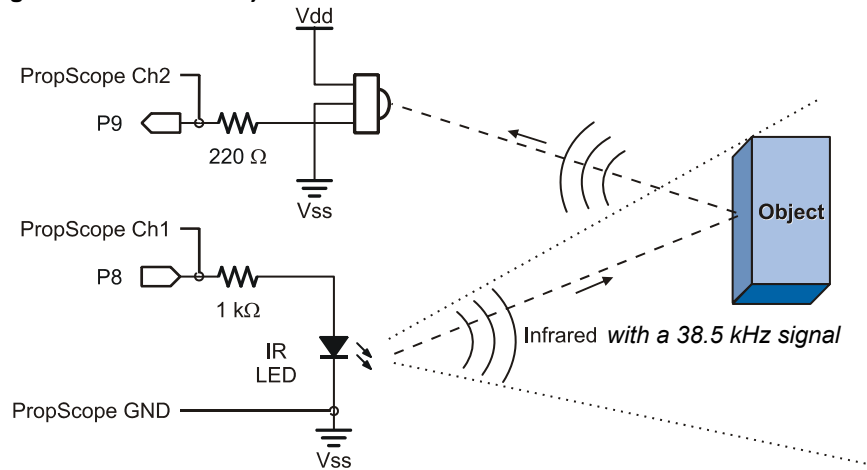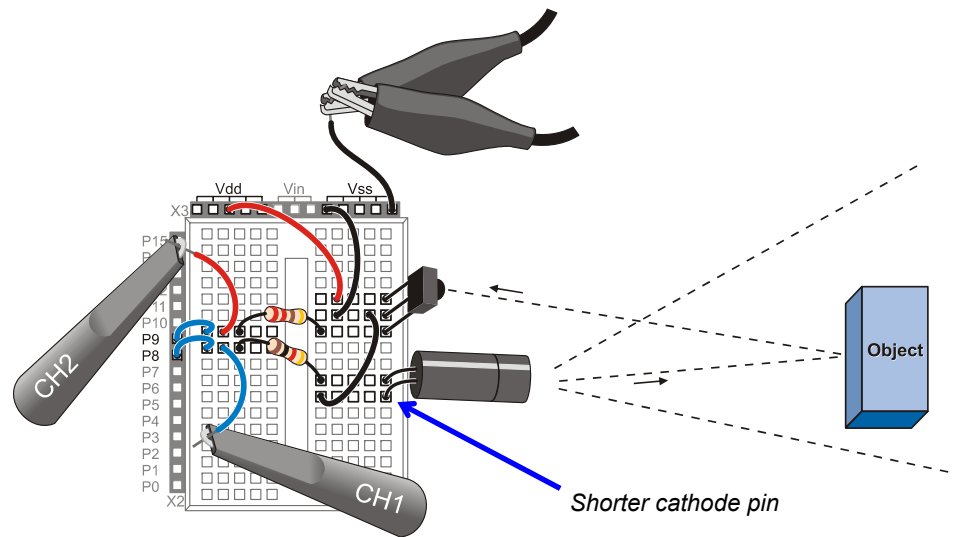
Figure 4-18: Infrared Object Detection Schematic

Figure 4-19: Wiring Diagram Example of Figure 4-18

Shorter cathode pin

IR Object Detection Test Code

~~TestLeftIrPair.bs2 is an excerpt from Robotics with the Boe-Bot that displays whether or not an object is detected in the Debug Terminal.~~ The command FREQOUT 8, 1, 38500 sends a 38.5 kHz signal to the IR LED that makes it a very rapidly blinking flashlight. If the IR detector detects this rapidly blinking light reflected off an object, it sends a low (0 V) signal. Otherwise, it sends a high (5 V) signal. The command irDetectLeft = IN9 stores the IR detector's output in a bit variable named irDetectLeft. This command has to immediately follow the FREQOUT command because there is only a brief time window between when the FREQOUT command stops and IR detector's output rebounds to high because it's not seeing the IR reflection any more.

✓ Enter and run TestLefIrPair.bs2.

```
' Robotics with the Boe-Bot - TestLeftIrPair.bs2
' Test IR object detection circuits, IR LED connected to P8 and detector
' connected to P9.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

```
irDetectLeft    VAR     Bit

DO

  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

  DEBUG HOME, "irDetectLeft = ", BIN1 irDetectLeft
  PAUSE 100

LOOP
```

> **i**   **In Chapter 7: Basic Sine Wave Measurements,** you will use the PropScope to examine the sine wave signals the FREQOUT command makes the BASIC Stamp synthesize with binary signaling.

IR Object Detection Tests

Figure 4-20 shows the Debug Terminal's reports for object not detected (1) and object detected (0).

- ✓ With no objects in the IR LED's line of sight, verify that the Debug Terminal reports irDetectLeft = 1.
- ✓ With an object (like a book or your hand) about 10 cm in front of where the IR LED is pointing, verify that the Debug Terminal displays irDetectLeft = 0.

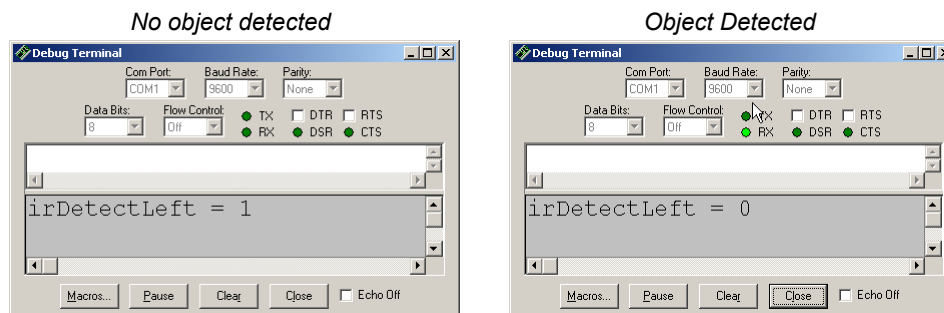**Figure 4-20:** Debug Terminal IR Detection Tests

Figure 4-21 shows the PropScope's Oscilloscope view while no object is detected. The FREQOUT signal transmits for 1 ms on the upper CH1 trace, but the CH2 trace remains

at 5 V. The BASIC Stamp I/O pin interprets the 5 V applied to its P9 I/O pin as a binary 1, which results in the left side of Figure 4-20.

4

- ✓ Configure your PropScope's Horizontal dial, Vertical dials and Coupling switches, and Trigger tab settings as shown in Figure 4-21.
- ✓ Make sure the trigger time control is set to the 2nd time division.
- ✓ Repeat the conditions that resulted in the irDetectLeft = 1 display from Figure 4-20, and verify that it results in an uninterrupted 5 V signal on CH2.
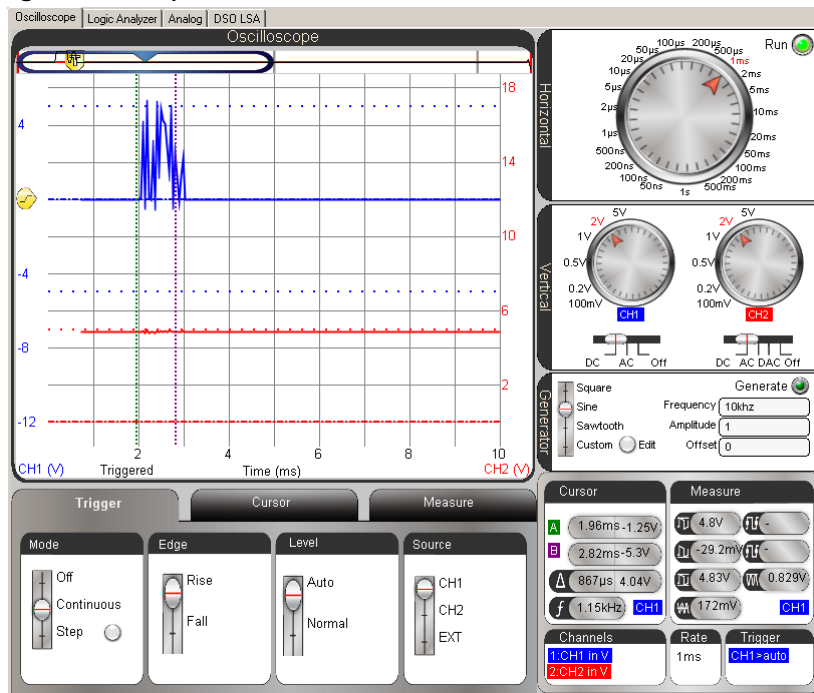
Figure 4-21: Object not Detected

Figure 4-22 shows the Oscilloscope view while an object is detected. The 1 ms FREQOUT signal is still shown by the upper CH1 trace, but now, the CH2 trace sends a low signal during that time, which indicates that it detected reflected infrared with a 38.5 kHz component.

✓ Place an object that will (like your hand or a book) facing the infrared detector. Use Figure 4-19 on page 149 as a guide for object placement.
✓ Verify that it results in the irDetectLeft = 0 display on the right side of Figure 4-20
✓ Verify that it results in a brief 0 V signal on CH2 while the FREQOUT command is active in CH1.

Figure 4-22: Object Detected
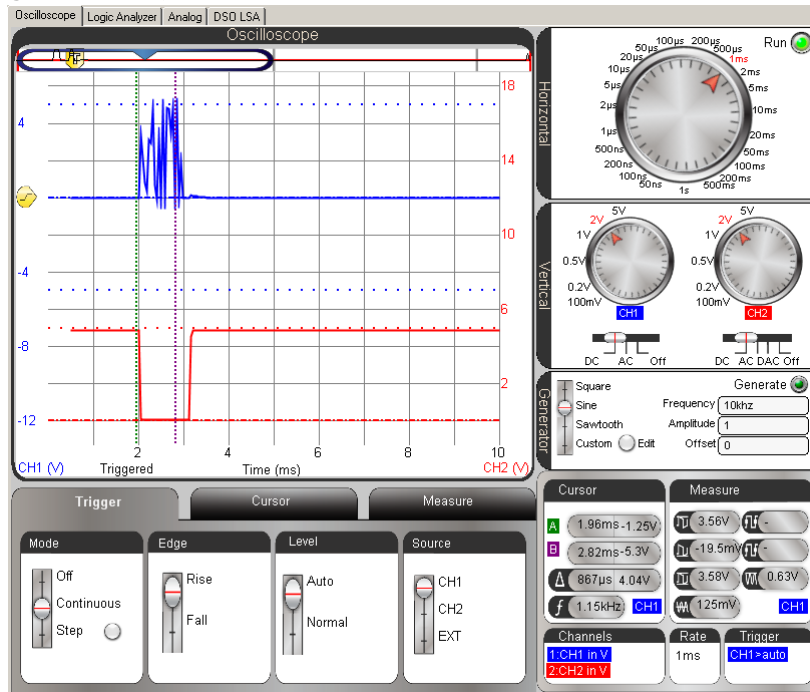
The BASIC Stamp's PBASIC program cannot measure this pulse duration while it is sending the FREQOUT command. However, you could use a second BASIC Stamp to send IR, and then use the first BASIC Stamp to measure the pulse duration with a command called PULSIN. You can also use the PropScope to measure the duration of the pulse the IR detector sends.

✓ Click the Cursor tab and turn the Vertical time cursors on.
✓ Line the cursors up along the negative and positive edges of the CH2 low pulse.