Understanding Signals with the PropScope

Student Guide

VERSION 1.0

PARALLAX INC

WARRANTY

Parallax warrants its products against defects in materials and workmanship for a period of 90 days from receipt of product. If you discover a defect, Parallax will, at its option, repair or replace the merchandise, or refund the purchase price. Before returning the product to Parallax, call for a Return Merchandise Authorization (RMA) number. Write the RMA number on the outside of the box used to return the merchandise to Parallax. Please enclose the following along with the returned merchandise: your name, telephone number, shipping address, and a description of the problem. Parallax will return your product or its replacement using the same shipping method used to ship the product to Parallax.

14-DAY MONEY BACK GUARANTEE

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a full refund. Parallax will refund the purchase price of the product, excluding shipping/handling costs. This guarantee is void if the product has been altered or damaged. See the Warranty section above for instructions on returning a product to Parallax.

COPYRIGHTS AND TRADEMARKS

This documentation is Copyright © 2010-2011 by Parallax Inc. By downloading or obtaining a printed copy of this documentation or software you agree that it is to be used exclusively with Parallax products. Any other uses are not permitted and may represent a violation of Parallax copyrights, legally punishable according to Federal copyright or intellectual property laws. Any duplication of this documentation for commercial uses is expressly prohibited by Parallax Inc. Duplication for educational use, in whole or in part, is permitted subject to the following conditions: the material is to be used solely in conjunction with Parallax PropScope and BASIC Stamp products, and the user may recover from the student only the cost of duplication. Check with Parallax for approval prior to duplicating any of our documentation in part or whole for any other use.

BASIC Stamp, Board of Education, Boe-Bot, Stamps in Class, and SumoBot are registered trademarks of Parallax Inc. HomeWork Board, PING))), Parallax, the Parallax logo, Propeller, and Spin are trademarks of Parallax Inc. If you decide to use any of these words on your electronic or printed material, you must state that "(trademark) is a (registered) trademark of Parallax Inc." upon the first use of the trademark name. Other brand and product names herein are trademarks or registered trademarks of their respective holders.

ISBN 9781928982548

1.0.0-11.02.27-SCP

DISCLAIMER OF LIABILITY

Parallax Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, or any costs of recovering, reprogramming, or reproducing any data stored in or used with Parallax products. Parallax is also not responsible for any personal damage, including that to life and health, resulting from use of any of our products. You take full responsibility for your BASIC Stamp application, no matter how life-threatening it may be.

ERRATA

While great effort is made to assure the accuracy of our texts, errors may still exist. Occasionally an errata sheet with a list of known errors and corrections for a given text will be posted on the related product page at www.parallax.com. If you find an error, please send an email to editor@parallax.com.

Table of Contents

Preface

The Understanding Signals with the PropScope kit includes the PropScope to connect to your computer, project parts to build circuits on your BASIC Stamp development board, and a 350+ page book that guides you step-by-step from the basics through advanced electronic measurement techniques.

The PropScope has the features of many electronic test bench tools built into one small package that you connect to your computer:

- DC voltmeter, to measure voltage levels
- Oscilloscope, to measure and plot voltages that vary with time
- Logic analyzer, to measure and plots digital signal levels
- Spectrum analyzer, to measure and plot sine wave components in signals
- XY-Plotter, to plot one signal voltage against another
- Function generator, to synthesize signals for testing circuits and microcontroller projects

These tools are used to test circuits, microcontroller interactions with circuits, and microcontroller communication with other integrated circuits and computers. Topics include measuring:

- DC voltages and currents
- "Human-speed" signals that vary with time and can be observed
- Pulses for controlling devices and synthesizing signals
- Digital signaling between microcontrollers and integrated circuits (chips)
- Digital signaling between microcontrollers and other microcontrollers (or computers)
- Sine waves in signals, and how filters and amplifiers affect them
- RC circuit responses, sensor measurements, and simple filters
- Basic amplifier building blocks

Many oscilloscope lab manuals present the material in a format that assumes a teacher or mentor is on hand to provide background information. In contrast, the lessons in this book are designed so that a beginner can succeed on his or her own.

Instead of jumping straight from simple DC voltage and current measurements to high speed analog signals, *Understanding Signals with the PropScope* takes incremental steps through measuring human-speed signals, followed by measuring a variety of binary high/low signals first. By the time the reader gets to analog signal measurements, he or she has already honed many of the skills needed from the earlier work.

The "Human-speed Measurements" chapter is important because it provides a bridge between DC and high speed signal measurements. This chapter guides the reader through measuring signals that he/she creates by turning dials, pressing buttons, and blinking lights. Along the way, the reader gains hands-on experience with measuring time varying signals with the oscilloscope and binary high/low signals with both an oscilloscope and logic analyzer. The reader also gets hands-on experience generating signals with a function generator and with the BASIC Stamp.

Most chapters include test signals that are generated by the function generator along with similar test signals generated by the BASIC Stamp microcontroller and a circuit. Creating similar signals with a microcontroller and one or more circuits is important, especially for the reader who wants to invent, or will be faced with a design project at some point down the road.

All the circuits, circuit + microcontroller, and design technique examples measured in the book are standard ingredients in electronic product designs, and can be found in a myriad of products, projects and inventions. The measurement techniques are also widely used in industry, and at various levels in technical and engineering educational programs.

The measurement techniques in this book are introduced in a variety of courses and grade levels. This text leans toward qualitative introductions, and when math is required, it uses the simplest expressions available. This helps keep it compatible with various levels provided by the various disciplines, grade levels and their theory textbooks. For introductory and survey courses, at home students, and hobbyists, this book can be studied in step-by-step detail. More advanced courses can use this book as a primer, or sections of this book can be studied before lab work that requires a particular measurement technique.

AUDIENCE

This text is designed to be an entry point to technology literacy and as an easy learning curve for embedded programming and device design. The text is organized so that it can be used by the widest possible variety of students as well as by independent learners.

SUPPORT FORUMS

Parallax maintains free, moderated forums for our customers, covering a variety of subjects. Some that may be pertinent to the reader are:

- Propeller Chip: for all discussions related to the multicore Propeller microcontroller and development tools product line.
- BASIC Stamp: Project ideas, support, and related topics for all of the Parallax BASIC Stamp models.
- Education (Stamps in Class): Students, teachers, and customers discuss Parallax's education materials and school projects here.
- Robotics: For all Parallax robots and custom robots built with Parallax processors and sensors.
- Sensors: Discussion relating to Parallax's wide array of sensors, and interfacing sensors with Parallax microcontrollers.
- Wireless: Topics include XBee, WiFi, GSM/GPRS, telemetry, and data communication over amateur radio.
- PropScope: Discussion and Technical Assistance for the PropScope USB Oscilloscope.
- Projects: Post your in-process and completed projects here, made from Parallax products.

RESOURCES FOR EDUCATORS

We have a variety of resources for this text designed to support educators.

Stamps in Class "Mini Projects"

To supplement our texts, we provide a bank of projects for the classroom. Designed to engage students, each "Mini Project" contains full source code, "How it Works" explanations, schematics, and wiring diagrams or photos for a device a student might like to use. Many projects feature an introductory video, to promote self-study in those students most interested in electronics and programming. Just follow the Stamps in Class "Mini Projects" link at www.parallax.com/Education.

Educators Courses

These hands-on, intensive 1 or 2 day courses for instructors are taught by Parallax engineers or experienced teachers who are using Parallax educational materials in their classrooms. Visit www.parallax.com/Education → Educators Courses for details.

Parallax Educator's Forum

In this free, private forum, educators can ask questions and share their experiences with using Parallax products in their classrooms. Supplemental Education Materials are also posted here. To enroll, email education@parallax.com for instructions; proof of status as an educator will be required.

Supplemental Educational Materials

Select Parallax educational texts have an unpublished set of questions and solutions posted in our Parallax Educators Forum; we invite educators to copy and modify this material at will for the quick preparation of homework, quizzes, and tests. PowerPoint presentations and test materials prepared by other educators may be posted here as well.

Copyright Permissions for Educational Use

No site license is required for the download, duplication and installation of Parallax software for educational use with Parallax products on as many school or home computers as needed. Our Stamps in Class texts and BASIC Stamp Manual are all available as free PDF downloads, and may be duplicated as long as it is for educational use exclusively with Parallax BASIC Stamp products and the student is charged no more than the cost of duplication. The PDF files are not locked, enabling selection of texts and images to prepare handouts, transparencies, or PowerPoint presentations.

FOREIGN TRANSLATIONS

Many of our Stamps in Class texts have been translated into other languages; these texts are free downloads and subject to the same Copyright Permissions for Educational Use as our original versions. To see the full list, click on the Tutorials & Translations link at www.parallax.com/Education. These were prepared in coordination with the Parallax Volunteer Translator program. If you are interested in participating in our Volunteer Translator program, email translations@parallax.com.

ABOUT THE AUTHOR

Andy Lindsay joined Parallax Inc. in 1999, and has since authored a dozen books and numerous articles and product documents for the company. Much of the material Andy develops is based on observations and educator feedback that he collected while traveling the nation and abroad teaching Parallax Educator Courses and events. Andy studied Electrical and Electronic Engineering at California State University, Sacramento.

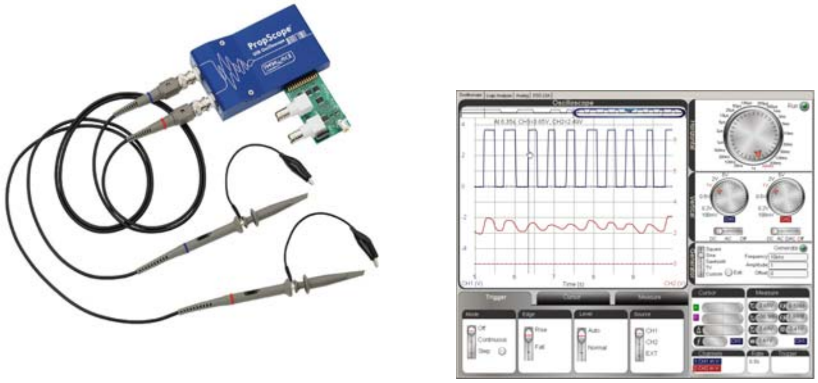
SPECIAL CONTRIBUTORS

The Parallax team assembled to prepare this textbook includes: Lesson design and technical writing by Andy Lindsay; cover art by Jen Jacobs; graphic illustrations by Rich Allred and Andy Lindsay; technical review by Joshua Donelson, editing and layout by Stephanie Lindsay. Thank you to: Ken Gracey, Stamps in Class program founder; Aristides Alvarez, Education Manager; Hanno Sander of Hannoware.com, PropScope Software Developer; Jeff Martin, BASIC Stamp Editor Software Developer; David Carrier, PropScope hardware developer. Thank you also to our customer reviewers for their questions, recommendations, and corrections: Paul Smith of Alverno College, Lisa Quackenbush, and Ingolf Sander.

Chapter 1: PropScope Introduction and Setup

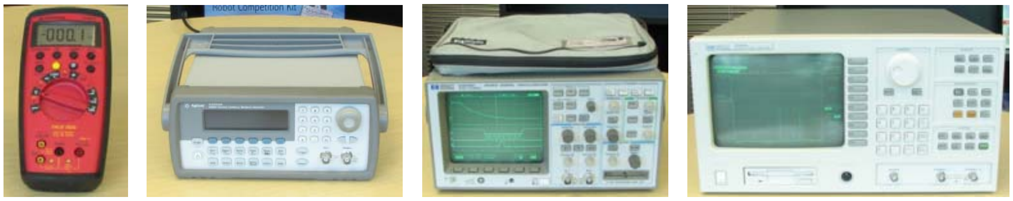
The PropScope USB hardware and software shown in Figure 1-1 make it possible to take a variety of electrical and electronic test and diagnostic measurements with your computer.

Figure 1-1: PropScope USB Hardware and Software



Many of the PropScope's capabilities used to require a test bench full of measurement equipment. Figure 1-2 shows examples (left to right) including the voltmeter, function generator, mixed signal oscilloscope (a combination oscilloscope and logic analyzer) and a spectrum analyzer. The PropScope provides the basic functionalities of these tools along with an affordable starting point for learning to take test measurements. Armed with the PropScope and the techniques this book introduces, you'll be able to test and troubleshoot many of the circuits and signals in your future electronics and/or robotics projects.

Figure 1-2: Test Equipment Examples



PROPSCOPE MEASUREMENT TOOLS IN A NUTSHELL

This section briefly explains each of the PropScope's measurement tools along with some of their most common uses. The PropScope is equipped to perform the basic functions of all these pieces of test equipment:

- Voltmeter
- Oscilloscope
- Function generator
- Logic analyzer
- Spectrum analyzer
- XY plotter

Voltmeter

The PropScope's voltmeter features are shown on the lower-left and lower-right of the Measure display in Figure 1-3. The lower-left value can be used to measure DC voltage, like across a battery's terminals or the regulated supply voltage for a microcontroller. You will take a variety of DC voltage measurements, starting in Chapter 2: DC Measurements.

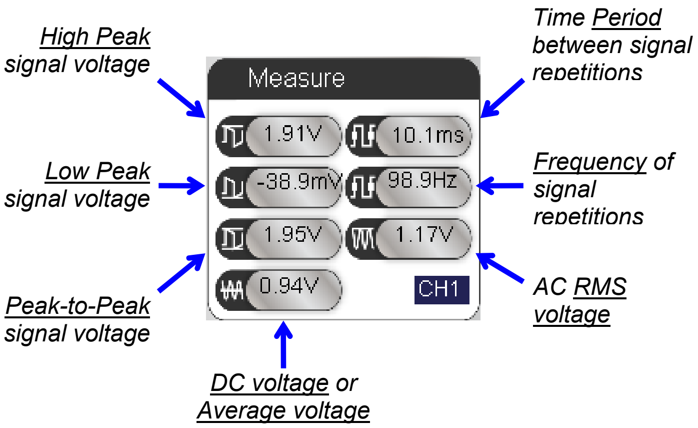


Figure 1-3
PropScope
Measure Display

Certain voltages that vary with time in a pattern that repeats periodically are called *periodic signals*. The varying voltages in such signals average out to some value. When the PropScope is measuring a periodic signal, it displays this average in this same field. If the voltage is DC, that lower left field displays the DC measurement. If it's a periodic signal, it displays the average voltage. A signal's average voltage is also called a *DC offset*, and it's introduced in Chapter 3: Human-speed Measurements.

The RMS voltage on the lower-right side of Figure 1-3 is a convenient way of measuring AC wall outlet voltages. RMS voltage is explained in the Understanding Signals Supplement, which is available as a free PDF online from the Downloads section of www.parallax.com/go/PropScope.

DO NOT TRY TO MEASURE WALL OUTLET VOLTAGE WITH THE PROPSCOPE.

Electrical shocks that can result from mistakes while attempting to measure wall outlet voltages can be fatal and improper use can also damage equipment and property.

If you want to learn how to take wall outlet and house/factory electrical measurements, seek qualified training in proper measurement procedures and safety precautions, and use only equipment that has been designed, rated, and certified for the measurements you take.

A stock PropScope cannot be used to measure wall outlet voltage. With the probes set to X10, the PropScope can measure up to 200 Vpp. (PropScope probes and probe settings will be introduced in Activity #2.) Since US wall outlet voltage is in the 340 Vpp neighborhood, it is well outside this range. X20, X50 or X100 probes can be purchased separately and used with the PropScope.

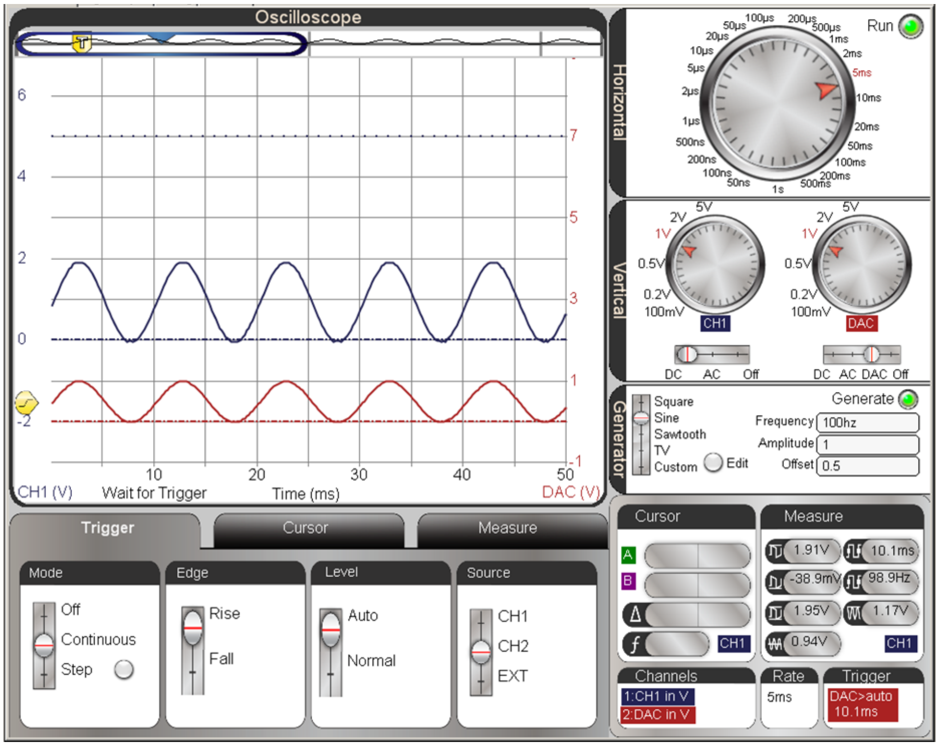
The other fields in Figure 1-3 summarize other attributes of periodic signals, and are components of oscilloscope measurements you will also take, starting in Chapter 3: Human-speed Measurements.

Oscilloscope

Many voltages that vary over time need to be monitored. An *oscilloscope*, which measures and plots voltage vs. time, is the perfect tool for the job. The Oscilloscope screen in Figure 1-4 is plotting two sine wave voltage signals. One sine wave (the shorter one on the bottom) is a signal going into an amplifier circuit. The taller sine wave above it is the amplified signal measured at the amplifier's output. (This amplifier will be introduced in Chapter 9: Op Amp Building Blocks.)

The Oscilloscope in Figure 1-4 has a Horizontal dial for adjusting the amount of time displayed in the x-axis—the large dial at the top. Below it are two Vertical dials for adjusting the amount of voltage displayed by two independent voltage y-axes.

Figure 1-4: PropScope Oscilloscope view



Frequency 100hz
Amplitude 1
Offset 0.5

1.91V
10.1ms
-38.9mV
98.9Hz
1.95V
1.17V
0.94V
CH1

Channels: 1:CH1 in V, 2:DAC in V
Rate: 5ms
Trigger: DAC>auto 10.1ms

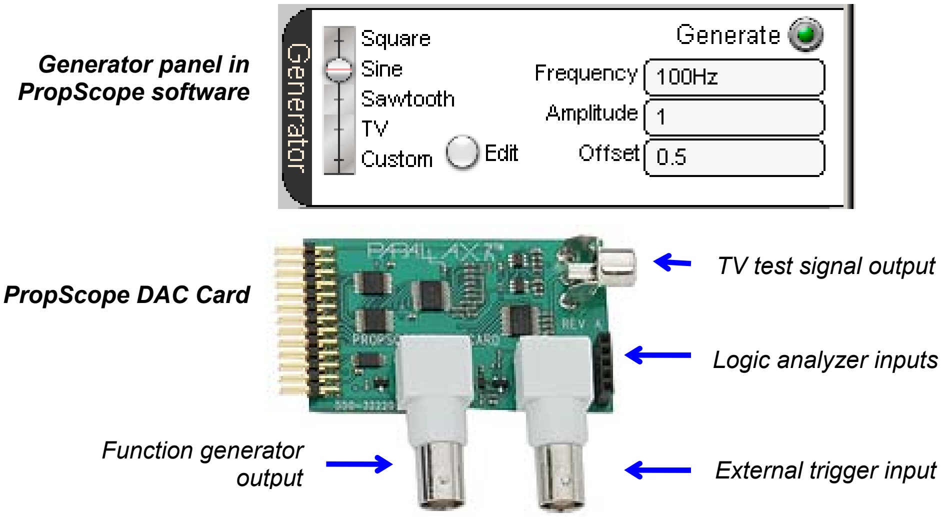
The channel 1 voltage scale is on the left side of the plot, and the channel 2 voltage scale is on the right. The voltage scales can be configured to convenient increments for displaying the signals measured by the PropScope hardware. Under the Vertical dials are the voltage coupling switches. When you set the CH2 coupling switch to DAC, it uses the CH2 trace to display output from the PropScope's function generator, which is introduced in the next section.

The Oscilloscope also has a Trigger tab for positioning the signal on the screen and a Cursor tab with tools for manually measuring voltage and time differences in the signal(s). The Measure tab displays measurements for both signals.

Function Generator

A *function generator* can synthesize a variety of voltage signals. These signals can be applied to a circuit input, and then the oscilloscope can measure results at a circuit output. This technique is useful for testing certain circuit properties and the effects they have on signals. It's also useful for verifying that a circuit is functioning properly as well as for tracking down certain malfunctions.

Figure 1-5: PropScope Function Generator



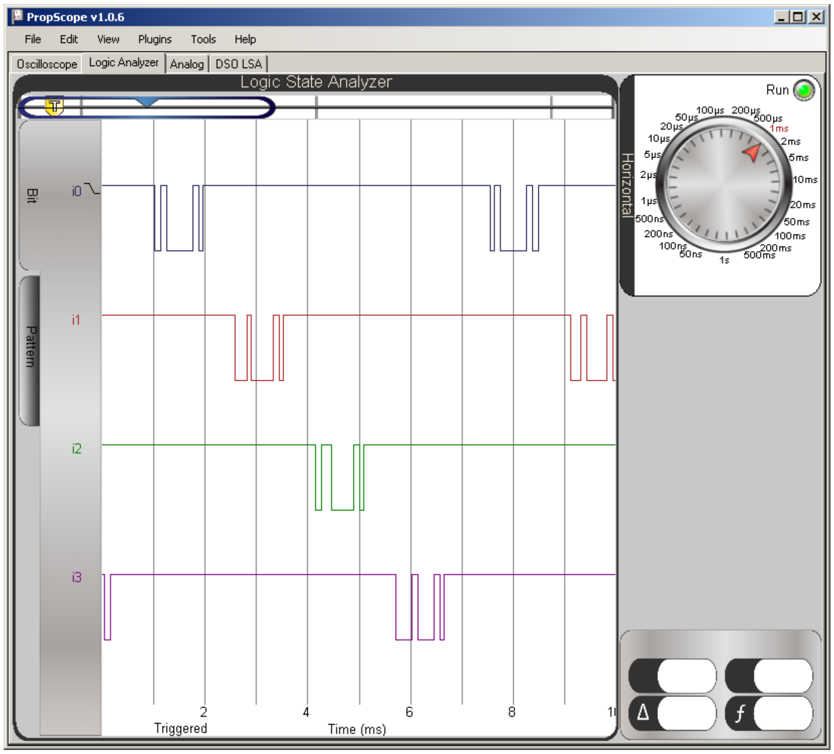
Frequency 100Hz
Amplitude 1
Offset 0.5

Common function generator signals include *square waves* (a series of high/low signals), *sine waves*, and a triangular pattern called "*sawtooth*." The PropScope Generator panel in Figure 1-5 also has an Edit feature that allows you to draw an arbitrary waveform that can be transmitted by sliding the waveform selector switch to the Custom setting. In addition to the signals listed, you can set DC voltages by setting the value in the Amplitude field to 0, and then adjusting the value in the Offset field for the desired DC voltage (from -1.5 to +4.7 V). We will set and measure PropScope function generator voltages starting in Chapter 2: DC Measurements.

Logic Analyzer

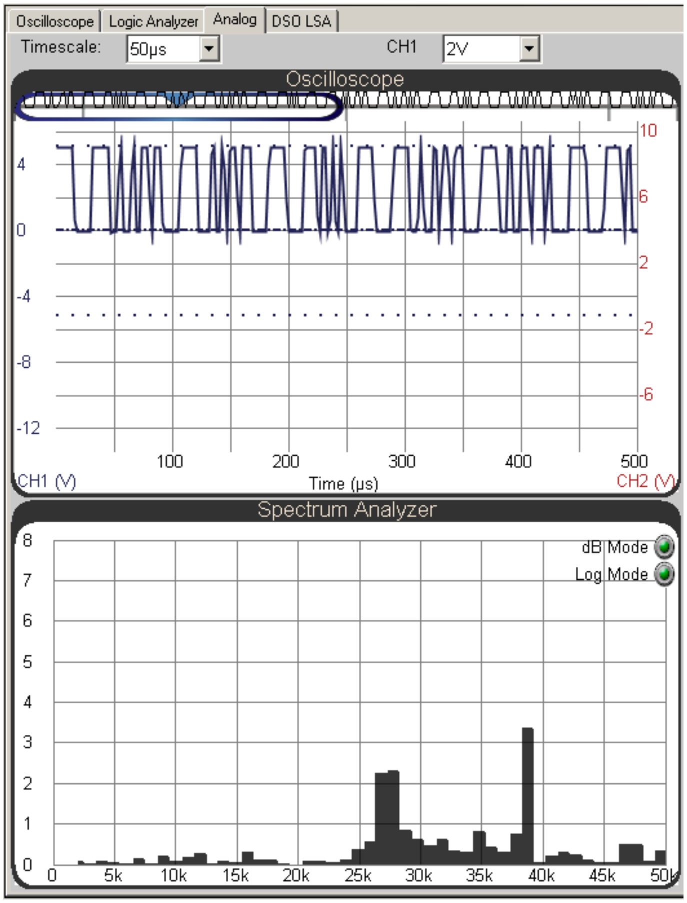
When digital devices communicate, there may be several signal lines exchanging binary high/low voltage signals. A *logic analyzer* plots the binary activity of multiple signal lines over time. Figure 1-6 shows an example in which the PropScope is monitoring four BASIC Stamp I/O pins, which are sending a sequence of four slightly different messages using a form of binary signaling called asynchronous serial communication.

Figure 1-6: Logic Analyzer



Spectrum Analyzer

A *spectrum analyzer* plots the amplitudes of sine wave components in a signal vs. their frequencies. Many signals have one or more sine wave components. Such signal components are commonly tested when monitoring engines for unhealthy vibration patterns, finding radio signal interference, and testing telephone dial tones.



Timescale: 50µs

CH1 2V

Figure 1-7
Oscilloscope above Spectrum Analyzer in the PropScope's Analog View

A fourth example of when a signal's sine wave components need to be tested is shown in Figure 1-7. The oscilloscope in the upper portion of the display is monitoring a rapidly switching binary signal that synthesizes two sine waves. It would be very difficult to tell what sine waves are being synthesized by looking at the upper Oscilloscope screen. The lower Spectrum Analyzer screen shows two prominent vertical bars in a graph, revealing two component sine waves, one in the 27 kHz neighborhood, and the other at about 38.5 kHz. Examining such signals will be introduced in Chapter 7: Basic Sine Wave Measurements.

XY Plot

This tool plots two signals' voltages against each other. One signal's voltages are treated as x-values and the other as y-values. The resulting plots are useful for determining certain relationships between two different signals. The line tipping at 45° in Figure 1-8 indicates that the two signals are "in phase," like the two sine waves in Figure 1-4 Their shape is the same, and the lower and upper sine wave's features are vertically aligned with no shift to the right or left. Certain circuits introduce delays between two signals at their inputs and two ones at their outputs. The result is an apparent left or right shift of one of the signals in the oscilloscope. In the XY Plot, this would produce an elliptical or even circular shape, or a line with a different angle, instead of the 45° slope shown in Figure 1-8. In addition to quickly determining phase relationships, the xy plot can also reveal relationships between signals at different frequencies.

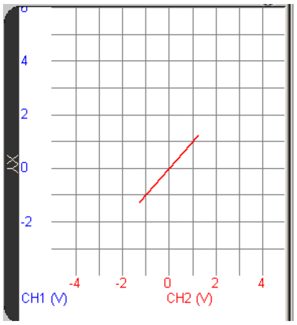


Figure 1-8
XY Signal Plot

TEST EQUIPMENT USES AND PROPSCOPE EXAMPLES

The voltmeter, oscilloscope, function generator, logic analyzer, and spectrum analyzer are arguably five of the most widely used pieces of test equipment in the electronics industry. Many engineers, technicians, repair specialists, students, and hobbyists use them to measure signals at various test points in circuits and systems. Specialized versions of many of these tools can also be found in the automotive and biomedical industries as well as in many science labs.

Most of the electronic devices we use on a daily basis were designed by groups of engineers, and tested by engineers and/or technicians. Each subsystem underwent multiple circuit and signal test measurements interleaved with troubleshooting to get to the finished product. Student and hobby projects often employ similar step-by-step approaches. Many devices also need to be repaired if they malfunction, which also involves test measurements.

There are many different types of signals and measurement techniques employed in the various fields of electronics and design. By following the activities in this book, you will get some initial, hands-on experience with many of the more common signals:

- DC supply and AC voltage
- Binary signal levels and timing for control and communication
- Digital signals that describe analog voltage measurements
- Sine waves for audio and analysis of filters and other circuits
- Sine wave components of audio, digital, and infrared signals
- Exponential decay for sensor measurements
- Amplifier signal conditioning

Along the way, you will also see a variety of circuit design and microcontroller programming techniques that can be found in common electronic products. If you completed *What's a Microcontroller, Robotics with the Boe-Bot*, or other Stamps in Class texts before starting here, you will also have the opportunity to use the PropScope's measurement tools to more closely examine the signaling involved in indicator and motor control, sensor monitoring, and communication with peripheral integrated circuits and other computers.

ACTIVITY #1: HARDWARE & SOFTWARE SETUP

The PropScope Quick Start guide is packaged with the PropScope and is also available from the PropScope product page. It lists computer system requirements and guides you through installing the PropScope software and connecting the hardware.

✓ Go to www.parallax.com/go/PropScope and find the Downloads section.
✓ Download the PropScope Quick Start Guide and follow the instructions.
✓ When you get to the step titled "Measure" continue from here.

Many of the activities in this book involve measurements of BASIC Stamp 2 microcontroller signaling as it interacts with circuits. These circuits are built from parts and components included in the Understanding Signals with the PropScope Kit. In these activities, you will use the BASIC Stamp Editor to load example programs into the microcontroller. The BASIC Stamp Editor software is available for free download from the Parallax web site. For best results, get the latest version before continuing.

- ✓ Go to http://www.parallax.com/basicstampsoftware and download and install the latest BASIC Stamp Editor.
- ✓ If you have never used your BASIC Stamp 2 and Board of Education or HomeWork Board before, click the BASIC Stamp Editor's Help menu, and select Getting Started with Stamps in Class. Follow the instructions for connecting your board and testing your BASIC Stamp programming connection.

If you do not already have a BASIC Stamp Microcontroller: Consult the Kit Choices section at www.parallax.com/go/WAM.

If you are new to microcontrollers: The Understanding Signals with the PropScope Kit has all the parts you'll need to complete chapters 1-5, 7, and 8 in the *What's a Microcontroller?* tutorial. If you have not already completed this text and are new to microcontrollers, it's highly recommended before continuing here.

- ✓ Locate the free PDF version of the *What's a Microcontroller?* text in the BASIC Stamp Editor software's Help menu (version 2.5 or higher).
- ✓ Complete Chapters 1-5, 8 and 9.
- ✓ Skip Chapter 2, Activity #5 and Chapter 3, Activity #5.

ACTIVITY #2: CONFIGURE AND ADJUST PROPSCOPE PROBES

The PropScope probes and software may need some small configurations and adjustments to make them compatible with the measurements and instructions in this book. This involves four simple steps:

1. Marking one probe with blue color bands and the other with red.
2. Setting the probe gain switch to X1.
3. Setting probe gain in the PropScope software to X1.
4. Connecting probes to BNC ports, blue to CH1 and red to CH2.

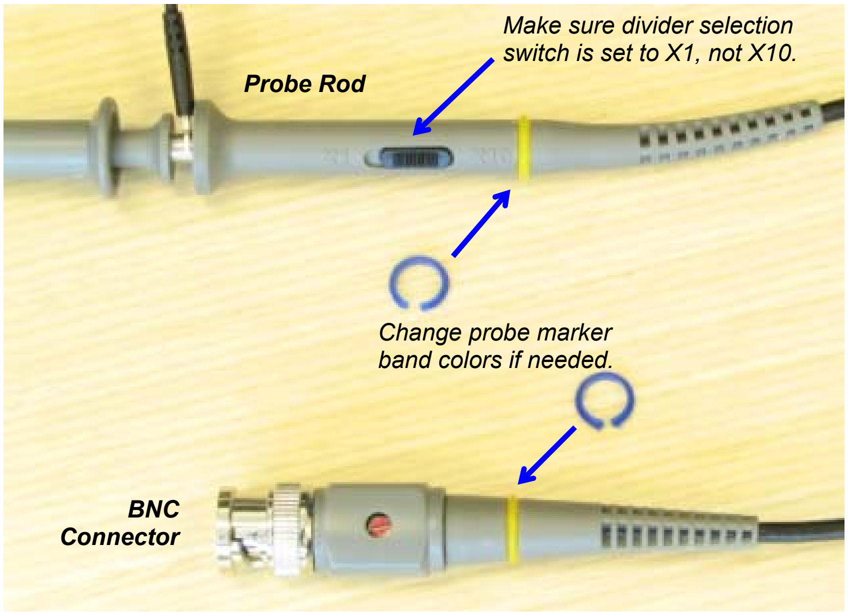
This activity will guide you through the four steps with some explanations along the way.

Step 1: Mark one probe with blue color bands and the other with red.

Since the information in the PropScope software is color coded blue for one channel and red for the other, one probe should be marked with blue marker bands, and the other with red. Each probe should have two marker bands of the same color, one on the probe rod and the other on the BNC connector as shown in Figure 1-9. With this setup, when you connect a probe to the circuit, all you need to do is check the marker band's color to know which color-coded information to check in the PropScope software.

- ✓ Make sure one probe has matching red color bands on its probe rod and BNC connector.
- ✓ Make sure the other probe has matching blue color bands on its probe rod and BNC connector.

Figure 1-9: Probe Rod and BNC Connector Adjustments


Make sure divider selection switch is set to X1, not X10.
Probe Rod
Change probe marker band colors if needed.
BNC Connector

BNC stands for Bayonet Neill-Concelman. Bayonet is name of the latching mechanism and Neill and Concelman are the names of the connector's inventors.

Step 2: Set the probe gain switch to X1.

The probe rod in Figure 1-9 also has a switch labeled “X1 X10.” These settings can be pronounced “times one” and “times ten.” Notations of 1x and 10x are also common, along with pronunciations of “one-X” and “ten-X.” These settings will be explained in more detail later in the book. For now, it’s important to make sure that both probe rod switches are set to X1.

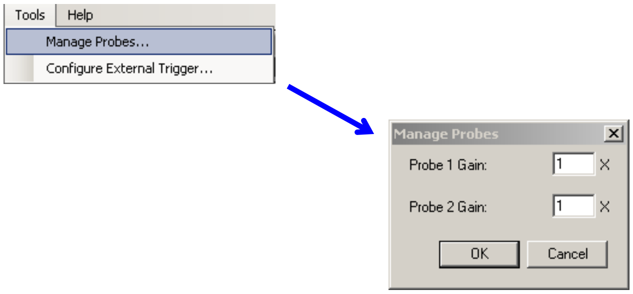
- ✓ Check each probe rod and make sure it is set to X1.

Step 3: Set probe gain in the PropScope software to X1.

A probe set to X10 reduces the signal voltage sent to the measuring device to 1/10 of the voltage applied to the probe. This voltage scaling is part of a circuit inside the probe that reduces the interactions with test circuits, and it also makes larger voltage measurements possible. In contrast, a probe set to X1 does not reduce the signal voltage and improves the resolution of measurements in the ranges we will study in this tutorial.

The PropScope software needs to know whether the probe is set to X1 or X10 since that setting would make it necessary to either multiply by 1 or 10 before reporting the measurement. Since our probes are set to X1, the software has to be configured accordingly.

- ✓ In the PropScope software, click Tools and select Manage Probes. Set the probe gain to 1 for both probes.



Probe 1 Gain: 1 X

Probe 2 Gain: 1 X

Figure 1-10
Configure Probe Gains in PropScope Software

You will use with the probe's X10 setting in Chapter 6, Activity #6. This will involve setting the probe tips and software settings to X10 and performing a one-time calibration with a small screwdriver; look for one packaged with your probes.

Step 4: Connect probes to BNC ports, blue to CH1 and red to CH2.

By default, the software color-codes voltage measurements from CH1 as blue, and from CH2 as red. So, the probes should be connected accordingly, like in Figure 1-11.

✓ Connect the probe with the blue marker bands to the PropScope's CH1 BNC connector, and connect the probe with the red bands to CH2.

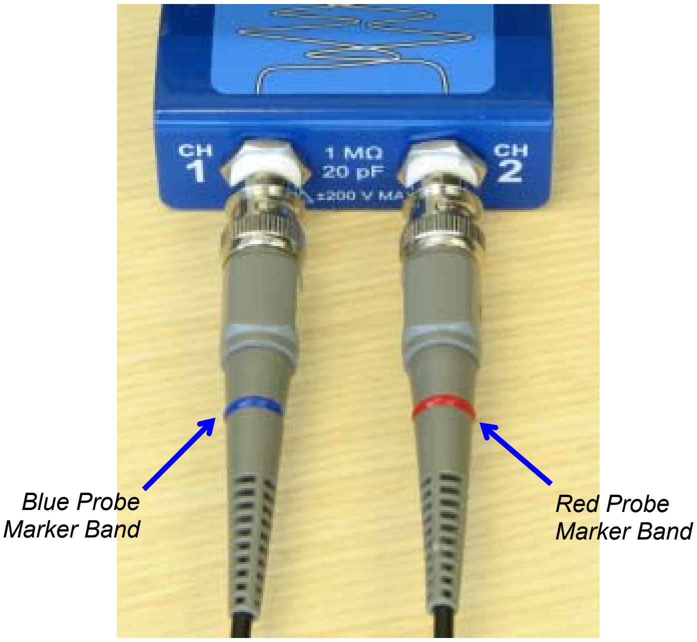


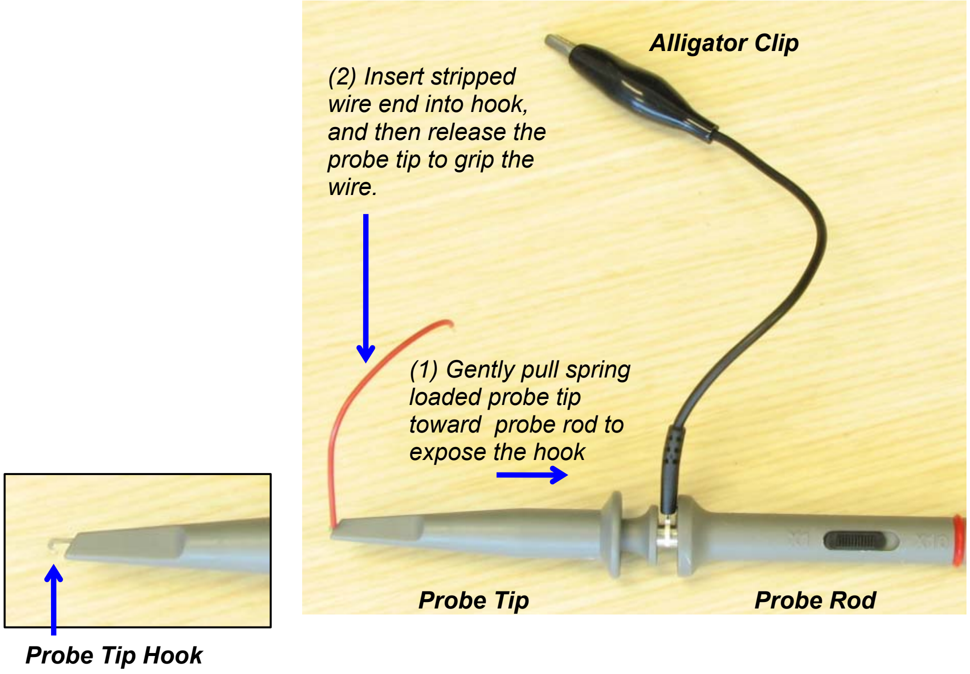
Figure 1-11
Channels and Marker Bands

Connect the probe with the blue marker bands to CH1 and the probe with the red marker bands to CH2.

Each PropScope probe has two clips shown in Figure 1-12: an alligator clip and a probe tip with a hook. Both are spring-loaded so that they can grip and hold wires and leads. A convenient setup for testing the voltages at breadboard sockets is to grab the stripped end of a jumper wire with a probe tip. The other end of the jumper wire can then be inserted into breadboard sockets to test voltages at various points.

- ✓ Grab a jumper wire to matches the color of each probe's marker bands.
- ✓ Retract the probe tip gently towards the probe rod to expose the hook, as shown in Figure 1-12.
- ✓ Insert a matching-colored jumper wire's stripped end into the probe tip hook.
- ✓ Release the probe tip to grip the jumper wire.

Figure 1-12: Grab a Stripped Wire End with the Probe Tip Hook


Alligator Clip
(2) Insert stripped wire end into hook, and then release the probe tip to grip the wire.
(1) Gently pull spring loaded probe tip toward probe rod to expose the hook
Probe Tip
Probe Rod
Probe Tip Hook

SUMMARY

The PropScope is a USB hardware peripheral and software tool that allows your computer to perform the functions of several different electronic test equipment components. The PropScope can stand in for the basic tasks of the voltmeter, oscilloscope, function generator, logic analyzer, spectrum analyzer, and XY-plotter. These devices are used by engineers, technicians, students, hobbyists and repair technicians for product development, projects, diagnostics, and repairs.

PropScope setup involves following the software installation and hardware connection instructions in the PropScope Quick Start Guide. Marker bands help make it clear which probe is connected to a given PropScope channel at a glance. They should be adjusted to match the colors of the channel information displayed by the PropScope: blue for CH1 and red for CH2.

Each probe tip has an alligator clip that gets attached to the ground connection of the device under test, and a spring-loaded, retractable probe tip that can grip wires and leads with a hook. Grabbing a jumper wire with the hook will simplify probing breadboard sockets. As with the color bands, use a blue wire with the CH1 probe and a red one with the CH2 probe.

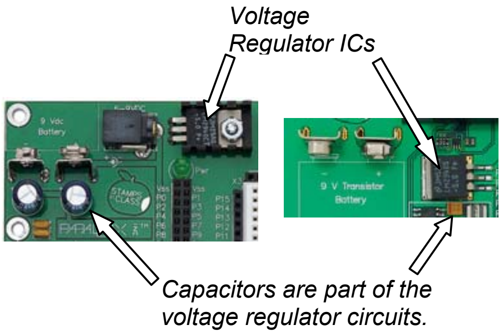
Chapter 2: DC Measurements

ABOUT SUPPLY AND OTHER DC VOLTAGES

Voltage is like a pressure that propels electrons through a circuit, and the resulting electron flow is called *electric current*. Electric current is often compared to water flow through a pipe or hose. If electric current is like the flow of water, then DC voltage is like the pressure that causes the water to flow through the pipe. The terms DC and AC describe current flow, and stand for direct current and alternating current. *DC current* flows "directly" through a circuit, propelled by a voltage that more or less remains constant. In contrast, *AC current* flows back and forth through the circuit, driven by an AC voltage source that typically alternates between positive and negative voltages many times per second.

DC voltage is a fairly easy concept to grasp because it's the measurement for typical supplies like batteries and solar panels. A common assumption about batteries is that they maintain a constant voltage. In contrast to an AC power source like a wall outlet, the voltage a battery supplies is constant; it's a "DC voltage" because it does not oscillate back and forth. However, the DC voltage is not necessarily a fixed value; it can vary. For example, if you measure the voltage across a battery with no current draw, it will be close to its rated voltage, like 9 V_{DC} (volts-DC) for an alkaline battery like the one you would use with your BASIC Stamp board. When a circuit that draws current is connected to the battery, its voltage may drop a little or a lot, depending on the electrical properties of the circuit and its current draw. The voltage of the battery also decays slowly as it loses its charge. A battery in continuous use might start at over 9 V_{DC}, but later, it will measure only 8.5 V_{DC}. The voltage will continue to decline as it loses its charge. At some point, it can no longer power the device, and we call it a "dead" battery.

Most digital circuits need a steady DC supply voltage to work properly. So, battery supplies get connected to circuit subsystems called *voltage regulators*. Figure 2-1 shows an example of a voltage regulator on the Board of Education and another on the BASIC Stamp Homework Board. The job of a voltage regulator is to maintain a certain voltage level for the circuits it supplies current to, regardless of whether they draw a little current or a lot. The voltage of the battery or other source that supplies power to the system is called the unregulated input. The voltage regulator's output, which stays at a fixed value, is called the regulated output. The Board of Education and BASIC Stamp HomeWork Board regulators have 5 V_{DC} regulated outputs, meaning they supply the circuits on the boards with a steady 5 V_{DC}, regardless of the current demand.



Voltage Regulator ICs

Capacitors are part of the voltage regulator circuits.

Figure 2-1
Voltage Regulators

Board of Education and BASIC Stamp HomeWork Board voltage regulators each have a regulator integrated circuit (IC) and a capacitor.

Since DC supply voltages may vary, they sometimes have to be measured before they get connected to certain types of loads—a *load* is a circuit that draws current when voltage is applied. Also, if a system has a short circuit in it, it can sometimes be detected as an unusually low supply voltage, likewise with faulty supply sources and dead batteries. Some systems even require supplies with certain characteristics, like output resistance, which can be determined by supply voltage and current tests.

Other devices rely on DC voltages to adjust how they operate. One example is a voltage-controlled oscillator. A voltage supplied by a device like the BASIC Stamp causes the oscillator to transmit a signal at a certain frequency. Another example: some amplifiers have to be fine tuned with DC voltages, which are called trim voltages. These voltages do not have to supply any power to the devices; they are just there to set some level. Because of this, they can be simpler circuits involving resistors or a digital to analog converter (D/A converter or DAC).

Both supply and other DC voltages are typically measured with a device called a *voltmeter*. Some voltmeters stand alone, and others are built into devices called *multimeters*. A multimeter can also measure resistance, current, and sometimes other electrical properties like capacitance and transistor gain.

In this chapter, you will use the PropScope in place of a voltmeter to measure your board's supply voltages. You will also build and test circuits that can generate DC voltages that could be used to adjust how other devices operate (trim voltages). Additionally, you will use voltage measurements at different points in circuits to determine the current passing through them, and also to determine whether a BASIC Stamp I/O pin can safely supply current to a given circuit.

ACTIVITY #1: GROUND TEST

The supply voltage sockets above your board's breadboard (Figure 2-2) are labeled Vdd, Vin, and Vss. Vdd is regulated 5 V_{DC}. Vin is the unregulated voltage of your board's power supply source, which should be in the 6 to 9 V_{DC} range. The sockets labeled Vss are considered 0 V. If you are using a battery, Vss is the voltage at the battery's negative terminal. Vss is commonly referred to as *ground* and also as a *ground reference* because the other voltages on your board are measured with reference to Vss. Vdd is 5 V_{DC} because it's 5 V above Vss. Likewise, Vin should be somewhere in the 6 to 9 V_{DC} range above Vss.

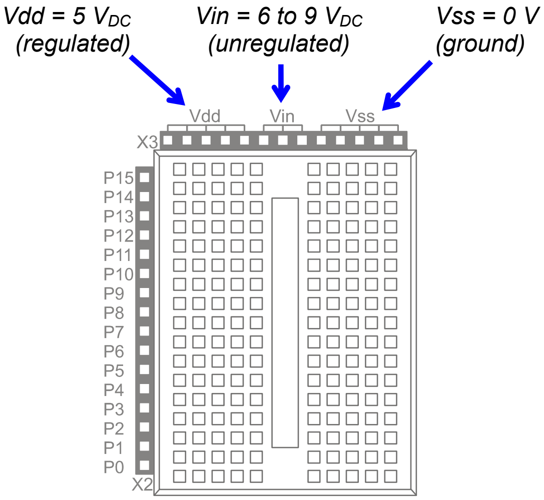


Figure 2-2
Your Board's Supply Voltages

What do the Vdd and Vss supply voltage labels stand for?

The labeling convention came from the names of the BASIC Stamp interpreter chip's power supply pins, which in turn came from a convention for names of voltage supplies to groups of a particular type of transistor. The transistors in the BASIC Stamp module's interpreter chip are called *field effect transistors* or FETs. A typical FET transistor has three terminals, named drain (d) gate (g) and source (s). When measuring a voltage at a particular transistor's drain, the voltage is typically labeled Vd. Likewise, the voltage measured at a transistor's source is Vs. A supply like 5 V_{DC} that goes to many FET transistor drains in a chip is labeled Vdd, and 0 V, which tends to be connected to many transistor sources is labeled Vss.

The PropScope can stand in for a voltmeter for measuring the supply voltages on your board. In this activity, you will first test the PropScope's probes to verify that they measure Vss as 0 V. In the next activity, you will use the PropScope to measure and verify that Vdd is 5 V_{DC}, and also to measure Vin. These measurements are useful for the sake of making sure your board's voltage supplies are in working order and properly connected. They are also useful for verifying that your PropScope is in working order and that you have correctly adjusted the controls for the measurements.

Steps before, between, and after measurements.

Before: This is your checklist for getting ready to work on activities in this book:

- ✓ If you have not done so, download and install the latest PropScope software from www.parallax.com/go/propscope
- ✓ If you have not done so, download and install the latest BASIC Stamp Editor software from www.parallax.com/basicstampsoftware.
- ✓ Connect your PropScope to your computer with a USB cable.
- ✓ Run the PropScope software and the BASIC Stamp Editor software.
- ✓ Connect your BASIC Stamp development board to your computer and power supply, and run a test program. (If you have never used your BASIC Stamp development board before, open the BASIC Stamp Editor Help, follow the Getting Started directions to connect your hardware to your computer and test your programming connection.)

Between: If instructed to modify a circuit on your board, or build a different circuit:

- ✓ Disconnect power from your board before modifying any circuit.
- ✓ Reconnect power after you have modified the circuit.

After: If you are done for the day, or leaving your circuit unattended for a while:

- ✓ If your board has a 3-position switch, move it to position 0.
- ✓ Disconnect the board's power source. (Even if the 3-position switch is off, it's still a good idea to disconnect the power source when the board is unattended.)

Ground Test Parts List

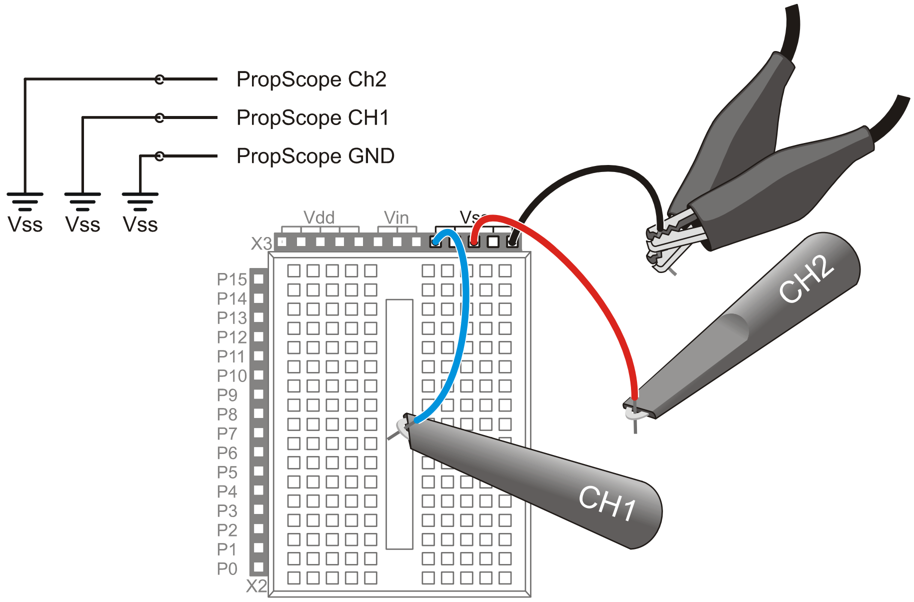
(3) Jumper wires

Ground Test Circuit

Figure 2-3 shows how to connect the PropScope's probes to verify that they measure 0 V at Vss.

- ✓ Disconnect power from your board.
- ✓ Connect the PropScope probes as shown in Figure 2-3.
- ✓ Reconnect power to your board.

Figure 2-3: PropScope Ground Test Schematic and Wiring Diagram



Ground Test Measurements and Settings

The next step for a voltage measurement is to configure the PropScope so that it can display voltage levels in its Oscilloscope screen. We will look more closely at what these configurations do in later activities.

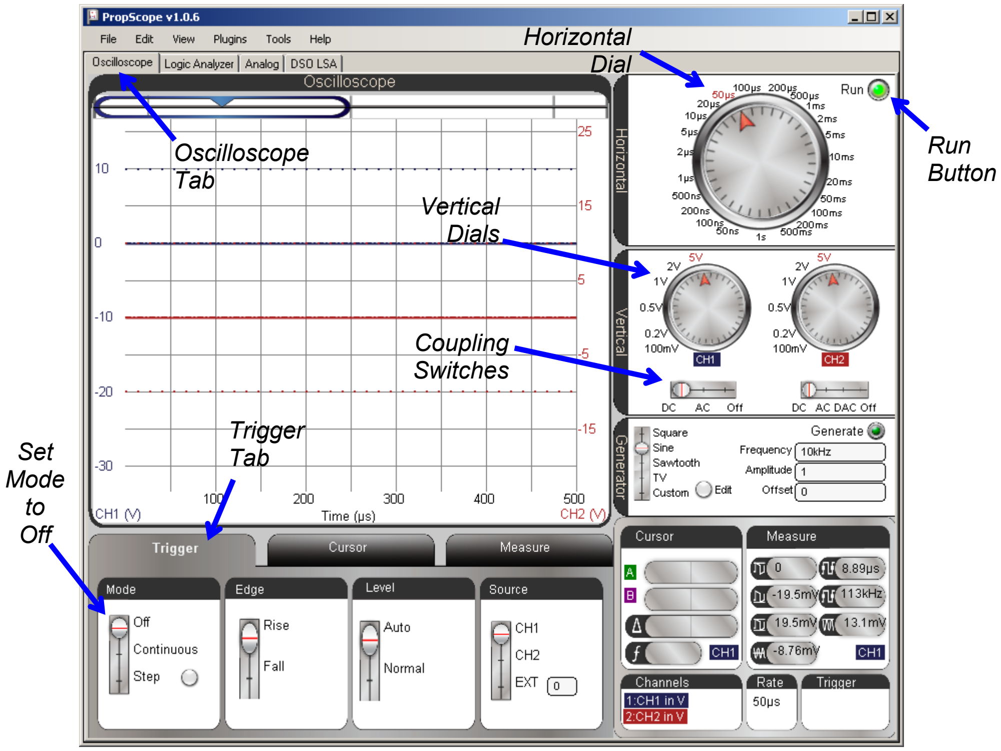
For now, just follow these steps using Figure 2-4 as a reference:

- ✓ Click the upper-left Oscilloscope tab to navigate to the PropScope's Oscilloscope view.

- ✓ Set both Vertical dials to 5 V.
- ✓ Set the Horizontal dial to 50 µs.
- ✓ Click the Trigger tab, and set the Mode slider switch to Off.
- ✓ Set both coupling switches (under the Vertical dials) to DC.
- ✓ Test the Run button by clicking it a few times to make it toggle between bright green (running) and dark green (not running).
- ✓ Make sure the Run button is bright green showing the oscilloscope is running.

Your display should now resemble Figure 2-4.

Figure 2-4: PropScope Settings for DC Voltage Measurements



Horizontal Dial

Run Button

Oscilloscope Tab

Vertical Dials

Coupling Switches

Trigger Tab

Set Mode to Off

Figure 2-5 shows a close-up of the oscilloscope display after some adjustments. Note that there are two bold colored lines, called *traces*, which cross the Oscilloscope screen.

You can grab these traces with your mouse pointer and drag them up and down so that their locations match Figure 2-5.

- ✓ In your PropScope software, use your mouse to point at, click, and hold the blue channel 1 trace.
- ✓ Drag the channel 1 trace either up or down to position it in your Oscilloscope display roughly matching the location in Figure 2-5.

As you move the channel 1 trace, the numbers in the voltage scale on the left should move with it. The 0 on the left, which indicates 0 V, should move to keep level with the trace as it moves.

- ✓ Make a similar adjustment to the red channel 2 trace so that it also matches the figure.

The scale on the right side should move with the channel 2 trace as you drag it up/down with your mouse, and the 0 V indicator on the right should also stay level with it as you make your adjustments.

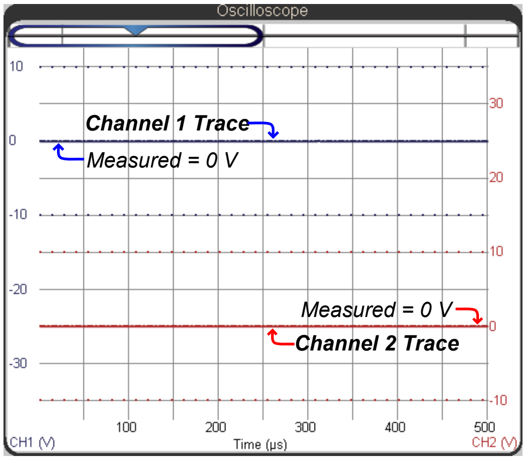


Figure 2-5
Voltages on the Oscilloscope

…with both Vertical scale dials set to 5 V.

Alright, so what are the traces in the Figure 2-5 Oscilloscope screen telling us? Since the probes are connected to Vss along with the ground clips, both traces should indicate 0 V measurements. The blue trace should be level with the blue 0 on the left side of the Oscilloscope screen, indicating that the voltage measured by the channel 1 (CH1) probe is zero volts. The voltage at the channel 2 (CH2) probe is also 0 V, so the red CH2 trace should be level with a red 0 on the right side of the Oscilloscope screen.

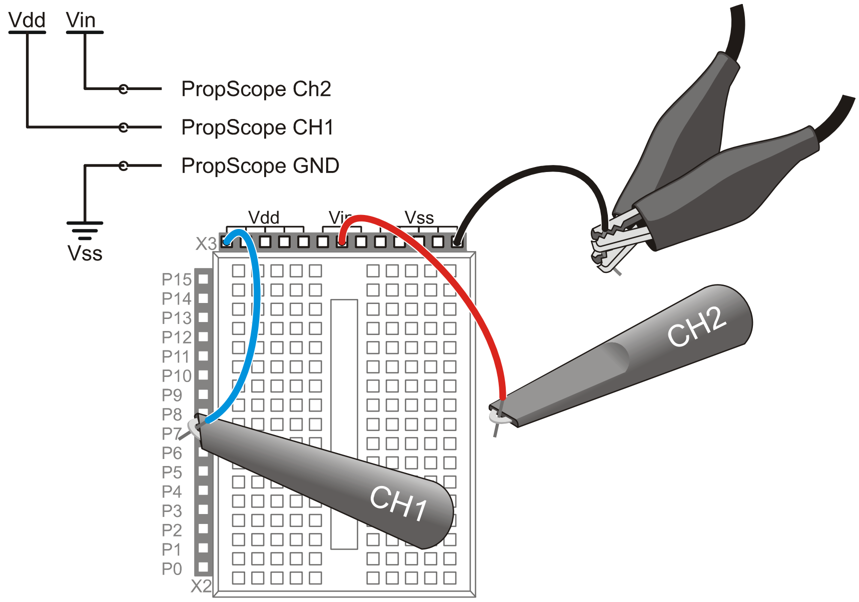
ACTIVITY #2: VDD AND VIN DC SUPPLY VOLTAGES

Next, let's verify that Vdd = 5 V_{DC} and measure your particular battery or DC supply voltage by probing Vin. We'll measure Vdd with channel 1, and Vin with channel 2. Figure 2-6 shows how to connect your probes. The ground clips don't move because they are tied to a reference voltage of Vss = 0 V.

Supply Voltage Test Circuit

- ✓ Disconnect power and plug the CH1 and CH2 probes into the Vdd and Vin sockets as shown in Figure 2-6.
- ✓ Reconnect power to your board.

Figure 2-6: Probe Supply Voltages with the PropScope



Supply Voltage Measurement Settings

When you reconnect your board's power, dashed lines will still remain at the 0 V levels for each channel to give an "at a glance" indication of ground voltage—the blue dashed line for channel 1, and the red dashed line for channel 2. The bold trace lines should move to new levels that indicate the voltages for each channel. The solid blue channel 1 trace should align with the 5 V division line, using the left-hand CH1 vertical scale. The solid red channel 2 trace should indicate the Vin voltage level, using the right-hand CH2 vertical scale. The example in Figure 2-7 shows that Vin is little more than half way between the 5 and 10 V divisions, so let's call it 8 V.

- ✓ Check where the voltage trace intersects with the CH1 voltage scale on the left. Is it 5 V?
- ✓ Repeat for Vin and the CH2 voltage scale on the right. It should be in the 6 V to 9 V range.

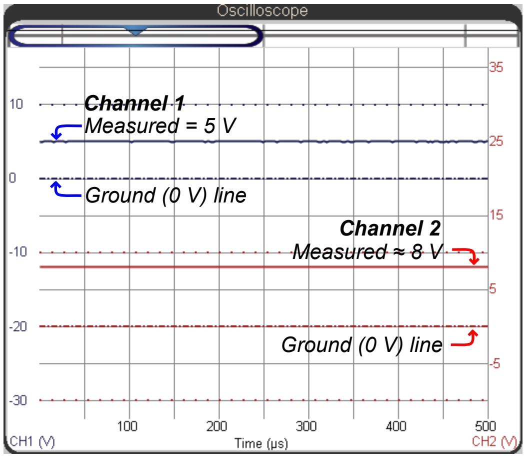
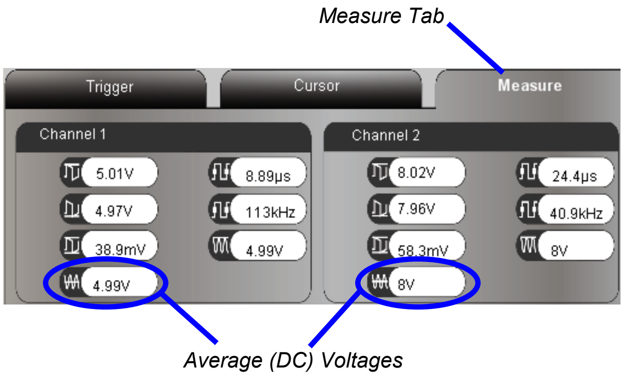


Figure 2-7
DC Signals on the Oscilloscope

…with both Vertical scales set to 5 V/div, CH1 connected to Vdd and CH2 connected to Vin.

The Oscilloscope view has a Measure tab that automates common signal measurements. While these measurements are designed mostly to quantify voltages that vary with time, the average voltage measurements circled in Figure 2-8 are essentially DC voltmeter measurements. As you will see later, the average voltage field can also give you information about DC components in periodic signals.

- ✓ Click the Oscilloscope view’s Measure tab.
- ✓ Examine the average voltage values.



Measure Tab

Average (DC) Voltages

Figure 2-8
Measure Tab Information

The Average voltage fields display DC values.

Your Turn: Battery Tests

The voltage across a 9 V battery’s terminals can give you an indication of how much charge is left. If the supply in Figure 2-8 is an alkaline battery, it has already supplied a significant portion of its charge, but still has some life left.

- ✓ Obtain a new 9 V alkaline battery, one that has been in service for a while but is still ‘good’, and a ‘dead’ one.
- ✓ Plug each battery into your Board, and make notes of the Vin voltage for each.

> **Testing battery voltage directly.** You can also connect a ground clip directly to a battery’s negative terminal and the probe hook to the positive terminal to measure its voltage.

ACTIVITY #3: OSCILLOSCOPE VOLTAGE SCALE ADJUSTMENTS

Now that you’ve tried a few simple DC voltage measurements with the oscilloscope, this activity takes a closer look at what each trace really represents. It also demonstrates how to adjust each channel’s voltage scale to accommodate the voltage measurement and how to check the voltage measurement limits for a given voltage scale adjustment.

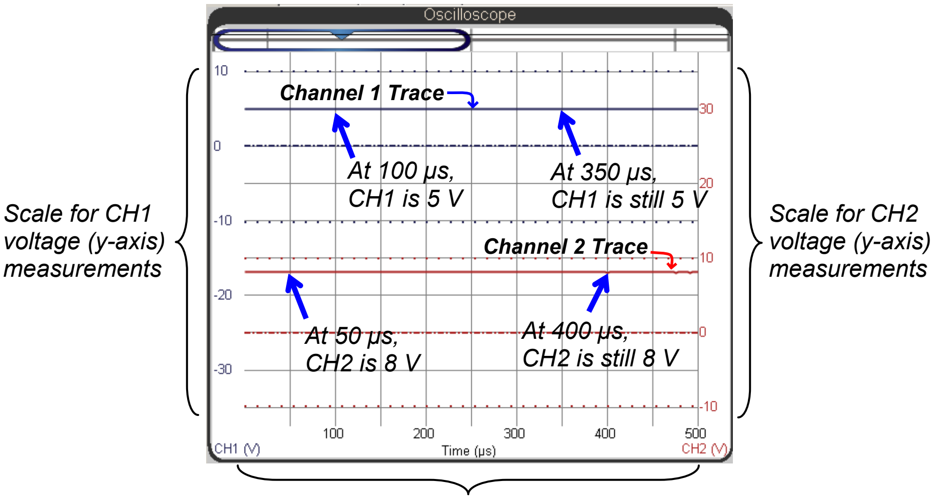
Points on the Trace (Voltage vs. Time)

A common math and science class activity is plotting x and y values with graph paper, a graphing calculator, or maybe a spreadsheet. In some graphs, the x values are equation input values that result in y output values. In other graphs, the x values are adjusted in lab tests, and the y values are the system's measured response to the x values.

The Oscilloscope screen is like two x-y plots on one sheet of graph paper. For the PropScope, each trace is made from 536 voltage measurements (y-axis values), plotted against the times the measurements were taken (x-axis values). Each channel has its own independent vertical scale for the voltage axis, and they both share a common horizontal time scale axis.

Figure 2-9 points out the scales for both channels along with some example points in each graph. To find out the voltage of a given point on a trace, just check what value it lines up with on its own voltage scale: CH1 on the left, CH2 on the right. The common time scale goes along the bottom of the plot, so to find the time of a point on either trace, just check what value the point is above on the time axis.

Figure 2-9: Scales and Sample Points on the Traces



Common time scale (x-axis) for both channel measurements

With the settings from Activity #2, the oscilloscope repeatedly displays 500 µs worth of voltage measurements. Figure 2-9 shows some examples of the voltage measurements and the times they were taken. The top-left example is a point on the CH1 trace that's half way between the 0 and 10 V scale values, so the voltage is 5 V, and the point is above 100 µs on the time scale. So the channel 1 voltage at 100 µs is 5 V. Later on, at 350 µs, the voltage is still 5 V. Of course, that's because the CH1 probe is connected to Vdd, which is regulated 5 V. So, every point on the CH1 trace will line up with the 5 V, and the result is a "flat line." In the next chapter, the voltage measurements will increase and decrease, resulting in traces that rise and fall as they plot the voltage fluctuations over time.

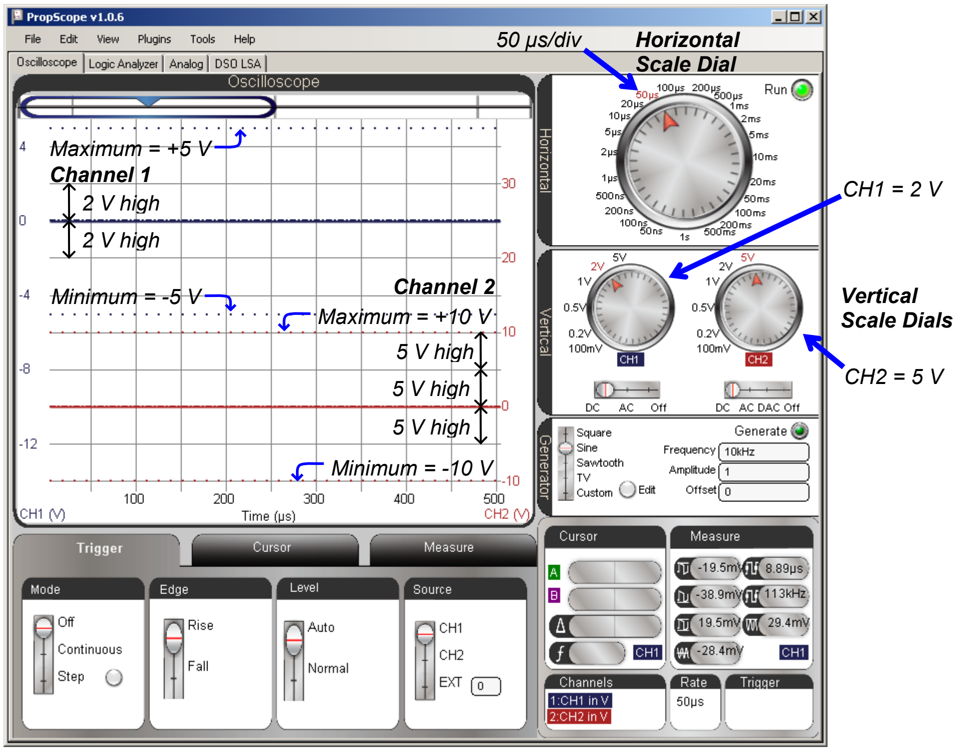
- ✓ Verify the two sample points on the CH2 trace. Remember that the voltage scale for CH2 is on the right side of the Oscilloscope screen. Also, keep in mind that your supply voltage depends on your particular supply, and should be in the 6 to 9 V range.

Adjust Vertical Scales

If you change the CH1 Vertical scale dial from 5 V to 2 V, each square height for CH1 on the Oscilloscope grid will represent 2 V. If you make this change, the blue values along the left side of the screen will also change so that they represent the smaller voltage increments. The vertical scales are independent, so if you only change the Channel 1 Vertical scale, the Channel 2 squares will still represent 5 V per square. Let's try it.

- ✓ Disconnect your PropScope's probes from Vdd and Vin, and reconnect them to Vss (like in Figure 2-3 on page 29).
- ✓ Try adjusting your CH1 vertical scale dial to 2 V. You may need to re-adjust the positions of the traces if you want it to closely resemble Figure 2-10.
- ✓ Look carefully at Figure 2-10. The CH1 (V) scale is now in units of 2 V, while the CH2 (V) scale is still in units of 5 V.

Figure 2-10: Channel 1 and 2 with Different Vertical Scales



Your Turn – Examine Measurement Limits and Clipping

Another important detail about changing the voltage scale is that certain settings affect the maximum and minimum measurements the channel can display. In Figure 2-10, the blue dotted line near the top of the display indicates the maximum and minimum input voltages for Channel 1, and now line up with 5 and –5 V on the CH1 voltage scale. The red dotted lines near the bottom indicate the maximum input voltages for channel 2, and are still at ± 10 V.

✓ Adjust the Vertical scales for each channel, and pay close attention to the maximum and minimum measurement limitations for each setting. You may need to adjust each trace up/down to see the dotted limit lines at certain settings.

If you try to measure a voltage that's outside the dotted line limits, it won't hurt the PropScope hardware, but you will get an incorrect voltage measurement. The PropScope will also display a red CLIPPED! warning, which means that the measured voltage has been "clipped" down to the maximum (or up to the minimum) voltage because it was outside the max or min for the voltage scale setting. That's your hint that you need to adjust the Vertical dial to get the correct measurement. Let's intentionally clip a voltage measurement so that you can see what happens.

- ✓ Restore the probes and oscilloscope settings as they were at the beginning of this activity with CH1 connected to Vdd, CH2 connected to Vin, and both the CH1 and CH2 Vertical dials set to 5 V.
- ✓ Verify that the CH1 trace indicates 5 V and the CH2 trace indicates your board's supply voltage.
- ✓ To cause the signal to clip, change the CH2 Vertical scale dial to 2 V. The voltage should incorrectly display as 5 V.
- ✓ Verify that a red CLIPPED! warning appears below the oscilloscope display. Again, remember that this is your warning that the measurement is incorrect because the voltage is outside the measurement set with the Vertical dials.

Nomenclature – Voltage and Time Divisions

Most graphs have grid lines for visually aligning a given point in the plot with its x and y-axis values. The Oscilloscope screen also has gridlines, but in oscilloscope terminology, they are called *division lines*. The grid lines that run across the screen are called *voltage division lines*, and the ones that run up/down are called *time division lines*.

Figure 2-11 points out examples of both voltage and time division lines. Voltage division lines go across the screen and separate it into ten vertical increments, which are called voltage divisions. The Vertical dials adjust the size of the voltage increment a voltage division represents. For example, in Figure 2-11, the CH1 Vertical dial is set to 2 V, meaning that each division represents a 2 V increment for Channel 1. The CH2 Vertical dial is set to 5 V, meaning that each division represents a 5 V increment for Channel 2. The Vertical dial settings are commonly referred to as *voltage settings*, and are described in terms of *volts per division*. So, you could say "The channel 1 Vertical dial is set to 2 volts per division." Likewise, the channel 2 dial is set to 5 volts per division. This would commonly be abbreviated to "CH1 = 2 V/div and CH2 = 5 V/div."

Figure 2-11: Division Lines and Divisions

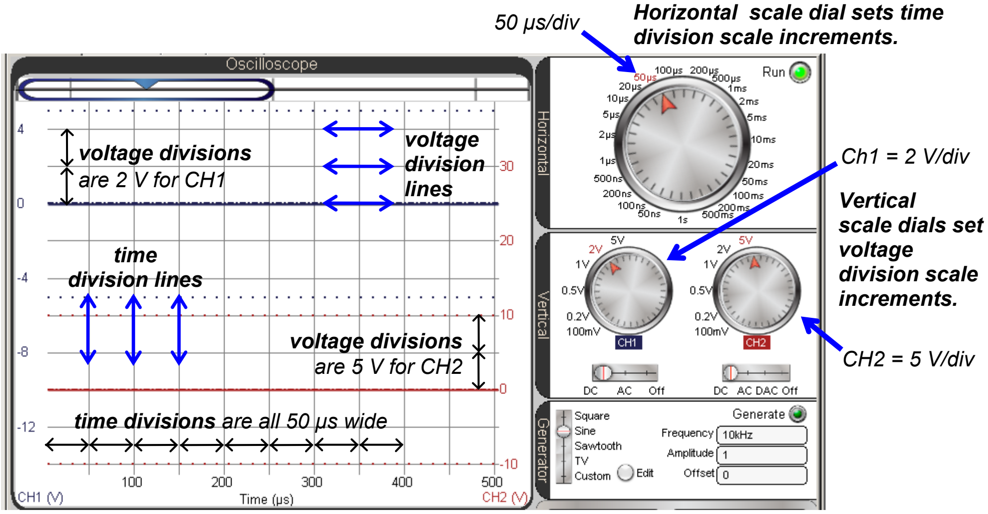


Figure 2-11 also points out the time division lines that travel up and down the screen, separating it into horizontal increments called *time divisions*. The Horizontal dial is set to 50 µs, which makes each time division represent a 50 µs increment. With the DC voltages in this chapter, it doesn't really matter what time division increment size you chose with the Horizontal dial. The DC measurements will still appear as flat lines that cross the Oscilloscope screen. In later chapters, the voltages will vary with time at certain rates, and you will have to adjust the Horizontal dial to get the best view of the signal as it rises and falls across the Oscilloscope screen.

ACTIVITY #4: DIGITAL TO ANALOG DC VOLTAGE

Digital to analog conversion is the process of taking a number (a digital value), and converting it to a corresponding output voltage (an analog value). Both the BASIC Stamp and PropScope have digital to analog conversion (abbreviated D/A or DAC) features that allow you to use numbers to generate DC voltages. The BASIC Stamp can generate voltages in the 0 to 4.98 V range, and the PropScope can generate voltages in the -1.5 to +4.7 V range.

Dual DAC Parts List

(2) Resistors – 1 kΩ (brown-black-red)
(2) Capacitors – 1 µF
(misc.) Jumper wires

Dual DAC Circuit

Figure 2-12 and Figure 2-13 show two BASIC Stamp resistor-capacitor D/A conversion (RC DAC) circuits along with the PropScope probes attached for measuring the DAC output voltages. These circuits can be useful for setting DC voltages in projects and prototypes. Some D/A converter application examples you will see in this book include BASIC Stamp controlled light sensor and amplifier adjustments.

CAUTION: This capacitor has a positive (+) and a negative (-) terminal. The negative terminal is the shorter lead that comes out of the metal canister closest to the stripe with a negative (–) sign. Always make sure to connect these terminals as shown in the circuit diagrams. Connecting one of these capacitors incorrectly can damage it. In some circuits, connecting this type of capacitor incorrectly and then connecting power can cause it to rupture or even explode.

CAUTION: Do not apply more voltage to an electrolytic capacitor than it is rated to handle. The voltage rating is printed on the side of the canister.

CAUTION: Safety goggles or safety glasses are recommended.

- ✓ Disconnect power from your board.
- ✓ Build the circuits shown in Figure 2-12 and Figure 2-13.
- ✓ Verify that the negative terminal for each capacitor is connected to Vss before reconnecting power to your board.

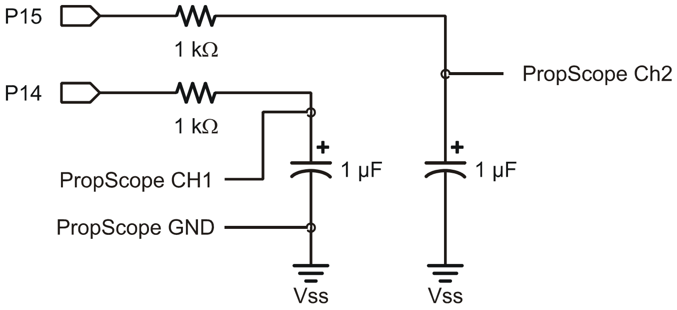


Figure 2-12
Dual DAC Circuit Schematic

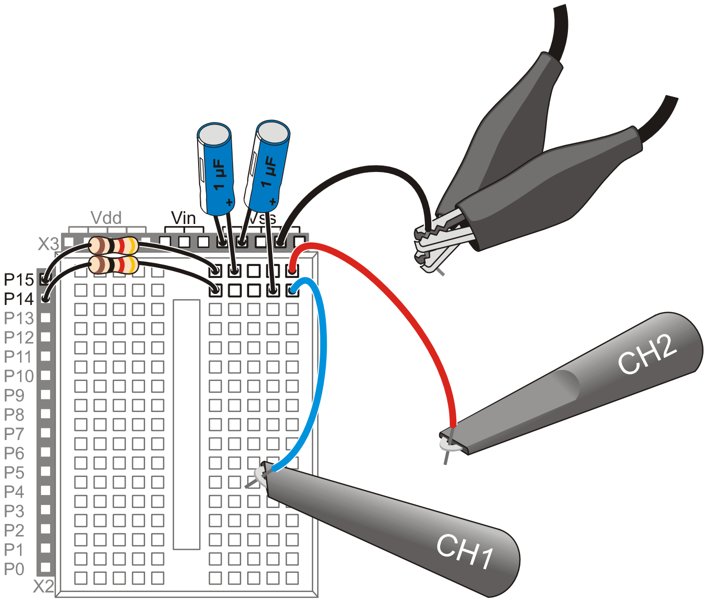


Figure 2-13
Wiring Diagram
Example of Figure 2-12

Make sure to connect the capacitors' negative leads (that come out of the metal can by the "–" stripe) to Vss.

Example Program: Test 2 Channel Dac.bs2

The PBASIC program Test 2 Channel Dac.bs2 uses the **PWM** command to repeatedly set voltages across the two capacitors in this circuit. In effect, the program uses the **PWM** command to make the BASIC Stamp perform D/A conversions, setting output voltages across the capacitors. The **PWM** command has three arguments, ***Pin***, ***Duty***, and ***Duration***. The ***Pin*** argument is for selecting which pin sends the PWM signal. ***Duty*** is the number of 256ths of 5 V the capacitor gets charged to, and ***Duration*** is the time the command has to deliver the signal.

In Test 2 Channel Dac.bs2, the command **PWM 15, 64, 5** sets the voltage across the capacitor in the circuit connected to P15 to 64/256ths of 5 V. That's 1.25 V. Likewise, **PWM 14, 192, 5** sets the voltage across the capacitor in the P14 circuit to 192/256ths of 5 V. That's 3.75 V. The ***Duration*** argument in both **PWM** commands is 5, which gives the command 5 ms to charge the capacitor. You will learn more about why it needs 5 ms in Chapter 8: RC Circuit Measurements.

- ✓ The **PWM** command should charge the capacitor in the P15 DAC circuit to 64/256^{ths} of 5 V, which is 1.25 V. Use a calculator to verify this voltage prediction.
- ✓ Repeat for the P14 **PWM** command and capacitor voltage to verify the 3.75 V prediction.
- ✓ Load Test 2 Channel Dac.bs2 into your BASIC Stamp.

Now that the program is running, the next step will be to test and verify the voltages across the capacitors with the PropScope.

```
' Test 2 Channel Dac.bs2
' Set P15 capacitor voltage to 1.25 V and P14 capacitor to 3.75 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 15, 64, 5                            ' 1.25 V to P15 capacitor
    PWM 14, 192, 5                           ' 3.75 V to P14 capacitor

LOOP                                         ' Repeat main loop
```

More about how PWM works and how to use it: Different facets of setting voltages with PWM and a resistor-capacitor (RC) DAC circuit will be explored later in this book:: Chapter 4: Pulse Width Modulation, Chapter 8: RC Circuit Measurements, Chapter 9: Op Amp Building Blocks.

Further Reading and Program Examples: Look up the command in the BASIC Stamp Editor's Help file. Also, see *Basic Analog and Digital* Chapter 7; the PDF tutorial is a free download from www.parallax.com/education.

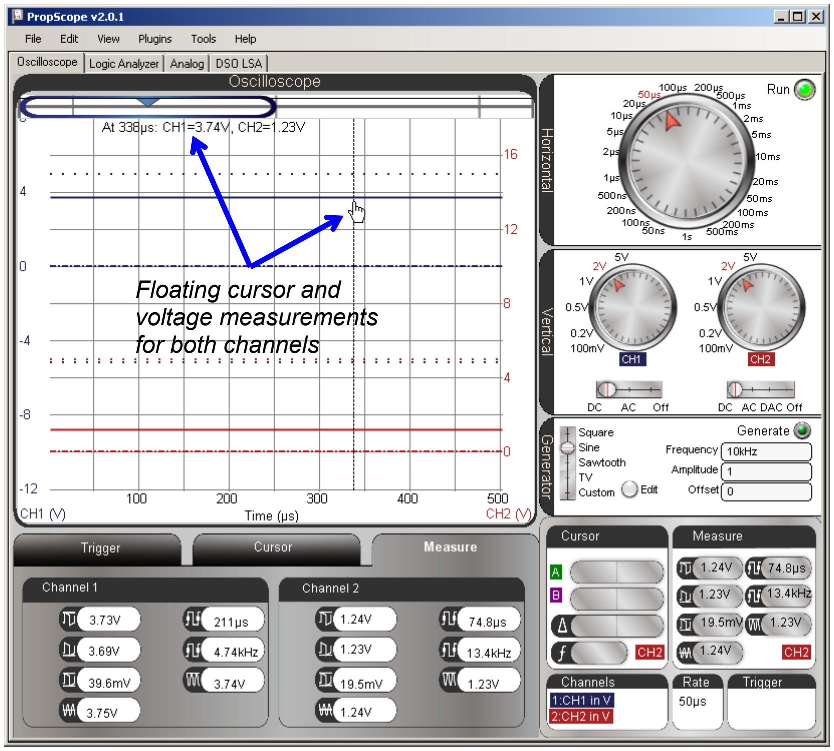
Measure 2 BASIC Stamp DC Outputs with the PropScope

Two good tools for testing DC voltages are the Oscilloscope view's floating cursor shown in Figure 2-14 and the Measure tab's Average voltage fields shown in Figure 2-15.

- ✓ Set both of the Vertical dials to 2 V/div.
- ✓ Adjust the positions of the traces to match Figure 2-14.

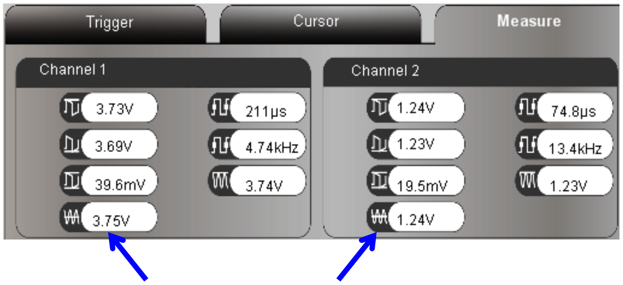
- ✓ Point at one of the traces with your mouse. It will make the floating cursor appear. The floating cursor will display the voltage measurements for both traces in the upper-left of the Oscilloscope screen.
- ✓ Make a note of the measured voltages. How close are they to your predictions?
- ✓ Click the Measure tab below the Oscilloscope screen.
- ✓ Check the average voltage measurements shown in the Average voltage fields.

Figure 2-14: Floating Cursor DC Voltage Measurements



At 338µs: CH1=3.74V, CH2=1.23V

Floating cursor and voltage measurements for both channels



Channel 1: 3.73V, 3.69V, 39.6mV, 3.75V, 211µs, 4.74kHz, 3.74V

Channel 2: 1.24V, 1.23V, 19.5mV, 1.24V, 74.8µs, 13.4kHz, 1.23V

Average voltage measurements

Figure 2-15
Measure Tab DC Voltage Measurements

> **Floating Cursor vs. Measure Tab**
>
> The floating cursor displays voltage at an instant in time corresponding to your mouse pointer's horizontal position on the Oscilloscope screen. The Measure tab's Average voltage displays the calculated average of the measurements over a period of time. Since even DC voltage fluctuates slightly, the average voltage is typically the more reliable means of obtaining DC measurements. However, if all you need is a quick and approximate DC voltage measurement, the floating cursor may save you a mouse click.

Your Turn: Calculate, Set and Test a Voltage with PWM

Let's say a device needs to have 3.5 V applied to one of its inputs. You may not be able to generate 'exactly' 3.5 V, but with PWM, you can probably find a digital value that will result in an analog voltage output that is close enough. Ideally, it should be within about 19.5 mV, which is 1/256th of 5 V. Knowing that the **`PWM`** command charges the DAC circuit's capacitor to voltages in terms of 256ths of 5 V, we can make a formula for calculating the **`PWM`** command's ***Duty*** argument for a given voltage in the 0 to 5 V range.

$$V_{PWM} = \text{Duty} \times 5V \div 256$$
$$\text{Duty} = V_{PWM} \div (5V \div 256)$$
$$\text{Duty} = V_{PWM} \div 0.01953125\ V$$

Next, solve for Duty with $V_{PWM} = 3.5$ V. You will have to round to the nearest integer since the **`PWM`** command's ***Duty*** argument accepts an integer value between 0 and 255.

$$\text{Duty} = 3.5\ V \div 0.01953125\ V = 179.2 \approx 179$$

So, the command **`PWM 14, 179, 5`** should set a voltage of just under 3.5 V at the P14 DAC circuit's output.

- ✓ Modify the **`PWM`** command to P14 in Test 2 Channel Dac.bs2 so that it transmits 3.5 V.
- ✓ Test with the PropScope.
- ✓ Repeat for 2.50 V.

Keep in mind that the **`PWM`** command's ***Duty*** value represents a certain member of 19.54 mV increments. The goal is to get within about 20 mV of the target voltage.

What does LSB mean? For digital to analog (D/A) and analog to digital (A/D) converters, an LSB describes the smallest voltage increment the device can work with. Under ideal conditions, 0.01953125 V is the voltage level across the DAC circuit's capacitor when ***Duty*** = 1 in the BS2 command **PWM *Pin*, *Duty*, *Duration***. The value 1 would be stored in the least significant bit (LSB) of the 8-bit ***Duty*** argument that controls the voltage output. Datasheets for D/A and A/D converters speak in terms of an LSB because their devices may be operating on a scale that's different from 0 to 5 V, so they keep it general. For example, if the scale is instead 0 to 3.3 V, the number of levels might still be 256 (8-bits), but the LSB would instead be 3.3 V ÷ 256 = 0.012890625 V. Same LSB, but a different voltage scale results in a different voltage increment.

Set DC Voltages with the PropScope's DAC Card

The PropScope can also set DC voltages, which is useful if a test circuit needs to be examined with a voltage applied. To generate test voltages with the PropScope, the CH2 Probe should be disconnected from the PropScope's CH2 input and connected to the DAC Card's function generator output.

- ✓ Disconnect the CH2 probe from the circuit in Figure 2-13 on page 41.
- ✓ Disconnect the probe from the PropScope's CH2 BNC connector, shown in Figure 2-16.
- ✓ Connect the BNC end of the free probe to the DAC Card's function generator output BNC connector, also shown in Figure 2-16.
- ✓ Connect the free probe end, which is now the DAC output, to the CH1 probe using Figure 2-17 and Figure 2-18 as a guide.
- ✓ If you just now plugged in the DAC Card into the PropScope, restart the PropScope software by clicking File and selecting Reset/Identify hardware.

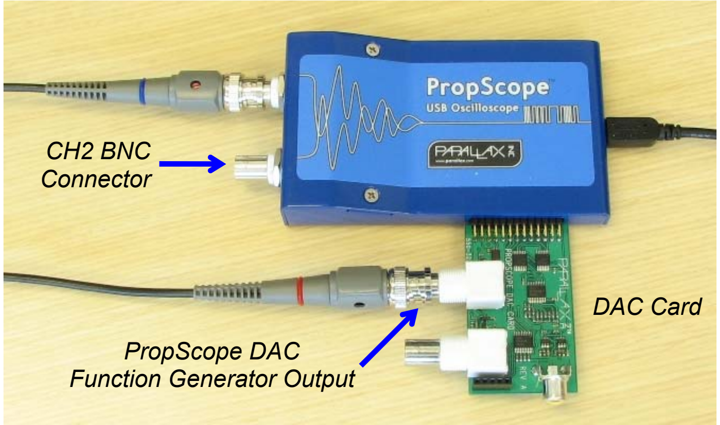

CH2 BNC Connector
PropScope USB Oscilloscope
PARALLAX
DAC Card
PropScope DAC Function Generator Output

Figure 2-16
DAC Card function generator Output

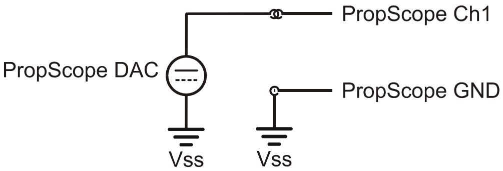


Figure 2-17
Probe DAC Output with Channel 1

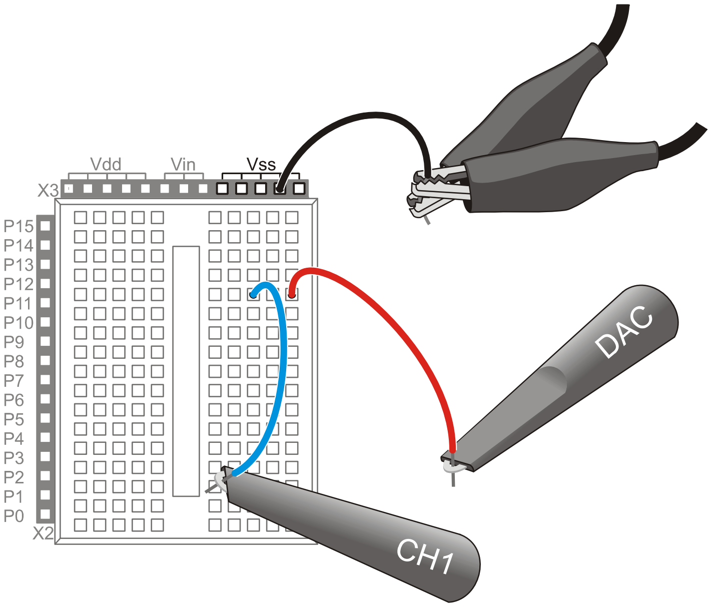


Figure 2-18
Wiring Diagram for Figure 2-17

Note that each probe tip is permanently connected to a ground clip.

Function generator Leads vs. Oscilloscope Probes

Oscilloscope input probes are not normally used as function generator test leads. For best results, get a set of BNC to alligator clip or BNC to Mini-Clip function generator test leads. If you have a variety of lengths to choose from, get the shortest one available.

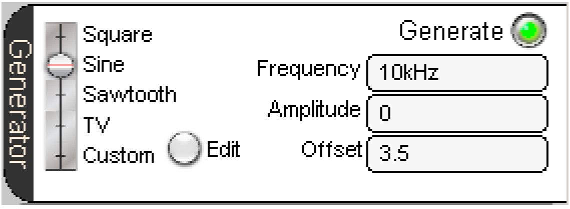
Even set to X1, the probes in your kit still have almost 100 Ω of series resistance between the function generator output and the probe tip contact. This is fine for transferring the measured signal to the oscilloscope inputs. However, if the function generator is driving a circuit that draws current, this resistance can cause a 1/10th of a volt per milliamp difference between the function generator's output and the voltage at the probe tip. Although it won't affect the measurements in this book, it could affect other PropScope measurements.

BNC to Mini-Clip Function Generator Test Leads

X1 for the function generator probe: The probe connected to the DAC Card's function generator output should always be set to X1. If you inadvertently set it to X10, it will divide down the voltage to 1/10th of what you would expect.

Set DC Signal with Function Generator, Probe with Channel 1

The Oscilloscope view's Generator panel is the PropScope's function generator control, and it will be used to generate a variety of signals. The first one will be a simple DC test voltage. Let's try a 3.5 V signal. Figure 2-19 shows the settings for a 3.5 V_{DC} signal.



Frequency 10kHz
Amplitude 0
Offset 3.5

Figure 2-19
DC 3.5 V with the Function Generator

- ✓ Frequency is set to 10 kHz by default, which is fine here.
- ✓ Set the wave type switch to Sine.
- ✓ Set the Amplitude to 0, and press Enter.
  (After every adjustment to a Generator panel Frequency, Amplitude or Offset value you have to press Enter.)
- ✓ Set the Offset to 3.5.
- ✓ Click the Generate button to start the digital to analog conversion.

DC Voltage with the Function Generator

A *function generator* is a device that makes a voltage vary over time according to certain mathematical functions. When plotted on graph paper, these functions look similar to the square, sine, sawtooth and other voltage variation patterns that you will see on the Oscilloscope screen when measuring a function generator's output. We will experiment with this, starting in the next chapter.

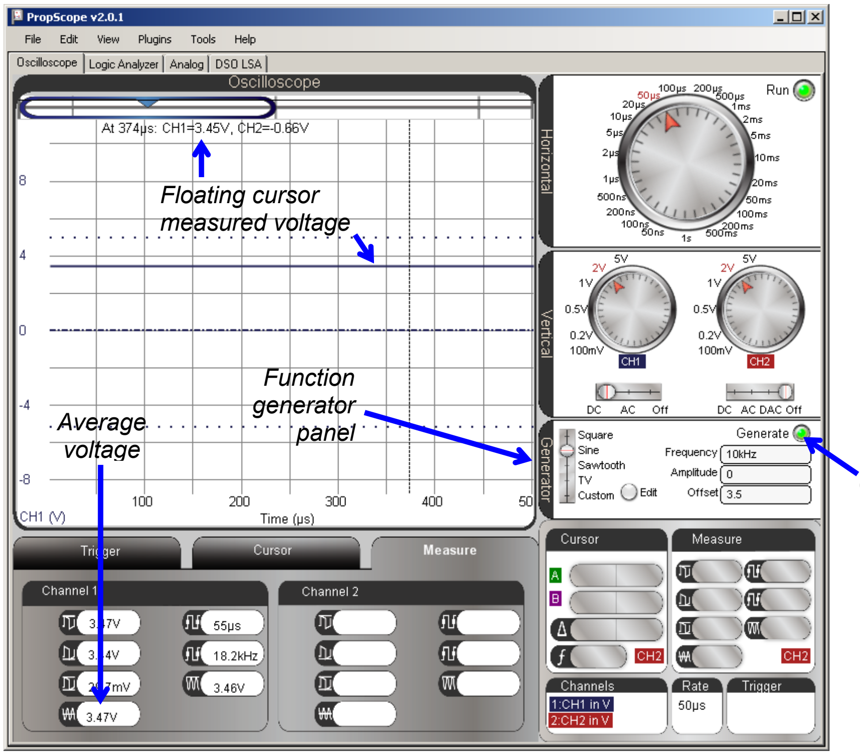
The function generator's Amplitude sets the total amount of voltage fluctuation, and the Offset sets the average voltage. The Frequency sets the rate of voltage fluctuations. So, if the function generator transmits a 1 V amplitude square wave with a 2 V offset, the square wave's high signals would be 2.5 V, and its low signals would be 1.5 V. That's a total voltage fluctuation of 1 V, with an average value of 2 V. If you change the amplitude to zero, the signal does not fluctuate any more, and you are left with a DC voltage at 2 V.

So long as the Amplitude is set to zero, the function generator will supply DC voltages. With Amplitude = 0, the frequency can be any value, and it doesn't matter whether the function is set to Sine, Square, or Sawtooth.

Figure 2-20 shows the measured voltage with the floating cursor and Measure tab.

- ✓ Set the CH1 Vertical dial to 1 V/div.
- ✓ Adjust the CH1 trace's position (point at the trace, click and drag up/down) so that it resembles Figure 2-20.
- ✓ Use your mouse to point at the Channel 1 trace to get the CH1 floating cursor measurements. Keep in mind that this is an instantaneous measurement that will fluctuate slightly with electrical noise and other factors.
- ✓ Check the value in the Measure tab's Channel 1 Average voltage field, and keep in mind that this measurement is an averaged value that is more similar to a DC voltmeter's measurement.

Figure 2-20: Channel 1 Probe of the Function Generator's 3.5 V Output



At 374µs: CH1=3.45V, CH2=-0.66V

Frequency 10kHz
Amplitude 0
Offset 3.5

Your Turn: Set and Measure Different Voltages

- ✓ Try setting the Offset to 3.75 V and repeat the measurements.

ACTIVITY #5: VOLTAGE DIVIDERS

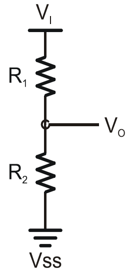
Many electronic products and prototypes have circuits that depend on a DC voltage that may not be available from the device's supply voltages. We just finished looking at one way to provide a set voltage for such a device, with a microcontroller and DAC circuit. However, a microcontroller D/A conversion can take I/O pins, processing time, and code. Provided the DC voltage you need is between two of the supply voltages like (Vdd = 5 V and Vss = 0 V) a quick solution for setting a voltage is to use two resistors in series.

Voltage Divider Parts List

(2) Resistors – 1 kΩ (brown-black-red)
(2) Resistors – 10 kΩ (brown-black-orange)
(1) Resistor – 2 kΩ (red-black-red)

Voltage Divider Circuit

Two resistors can be connected in series (end to end) with a voltage applied, like V_I and Vss in Figure 2-21. The voltage at the node where the two resistors meet will be "divided" according to the voltage divider equation in the figure.



$$V_O = V_I \times \frac{R_2}{R_1 + R_2}$$

Figure 2-21
Voltage Divider Circuit and Equation

> **The circuit in Figure 2-21 is designed to supply voltage, but not current.** The RC DAC you experimented with has a similar limitation. Many devices have voltage inputs that are called *high impedance inputs*—they do not draw current, and are compatible with voltage dividers and RC DACs.
>
> Op amp buffer circuits will be introduced in Chapter 9. *Op amp* is a shortened version of the component's real name—*operational amplifier*. A buffer's output voltage can be determined by a voltage divider or an RC DAC, and it can maintain its output voltage and still supply current to circuits. For example, the PropScope DAC Card's function generator output is buffered by an op amp, making it possible to drive DC current loads up to about 1 mA minimal without any measurable reduction in output voltage.

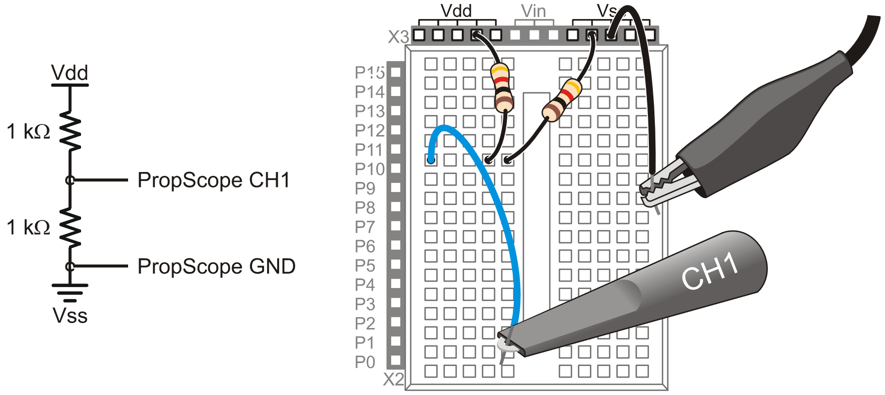
If you substitute V_I = Vdd = 5 V and $R_1 = R_2$ = 1 kΩ, the result of V_O should be 2.5 V.

- ✓ Try substituting the values into the voltage divider equation. Do your calculations agree? What happens if you use two 10 kΩ resistors instead? How about R_1 = 1 kΩ and R_2 = 2 kΩ? Also, try R_1 = 2 kΩ and R_2 = 1 kΩ.

Figure 2-22 shows a test circuit to verify the voltage divider output with $R_1 = R_2 = 1\ k\Omega$, and $V_I = Vdd = 5\ V$.

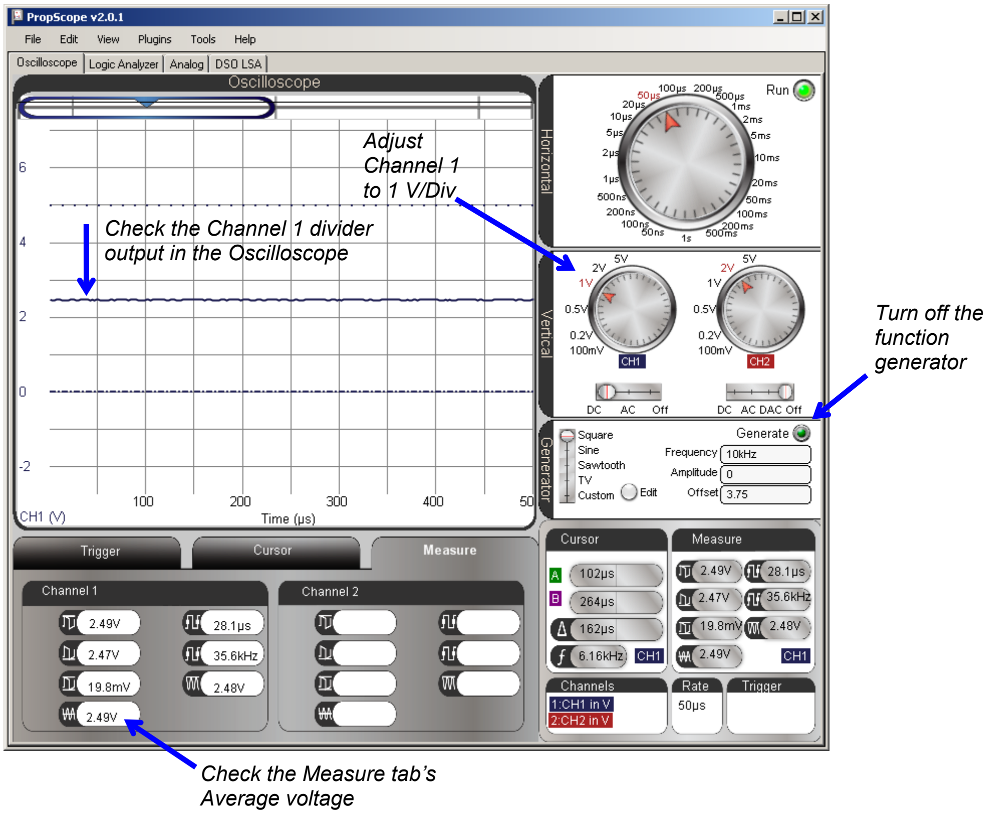
- ✓ Build the circuit shown in Figure 2-22.

Figure 2-22: Voltage Divider Measurement Schematic and Wiring Diagram



- ✓ Click the Generate button to turn off the function generator. The button's color should change from bright green to dark green.
- ✓ If you have not already done so, set the Channel 1 dial to 1 V/div and adjust the trace's position in the oscilloscope display so that it resembles Figure 2-23.
- ✓ The CH1 trace level should be about half way between the 2 and 3 V Channel 1 division lines, indicating 2.5 V.
- ✓ Check the CH1 Average voltage in the Measure tab to verify.

Figure 2-23: Test the Voltage Divider Output



Frequency 10kHz
Amplitude 0
Offset 3.75

Channel 1: 2.49V, 2.47V, 19.8mV, 2.49V, 28.1µs, 35.6kHz, 2.48V

Cursor: A 102µs, B 264µs, Δ 162µs, ƒ 6.16kHz CH1

Measure: 2.49V, 2.47V, 19.8mV, 2.49V, 28.1µs, 35.6kHz, 2.48V, CH1

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 50µs

Adjust Channel 1 to 1 V/Div

Check the Channel 1 divider output in the Oscilloscope

Turn off the function generator

Check the Measure tab's Average voltage

Not all voltage measurements will be as close as 2.49 V. Any measurement in the 2.3 to 2.7 V range indicates that the voltage divider is working. See the Voltage Divider Error Propagation section of the Understanding Signals Supplement, which is available as a free PDF online from the Downloads section of www.parallax.com/go/PropScope.

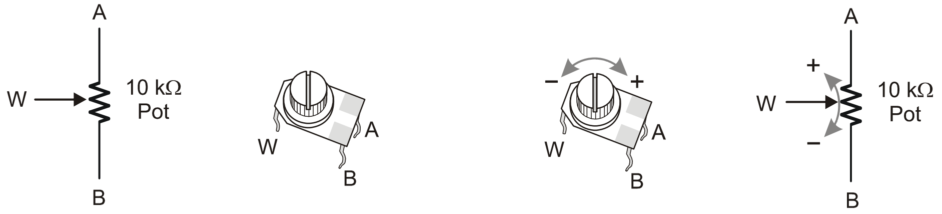
Your Turn: Other Resistor Combinations

✓ Repeat the voltage divider tests for the other resistor combinations discussed at the beginning of this activity.

The Potentiometer – an Adjustable Voltage Divider

In brief, a *potentiometer* has a resistive element that spans its A and B terminals (see Figure 2-24) and a knob on top for adjusting where the W terminal contacts that resistive element. As you turn the knob one direction, the resistance between W and B decreases while the resistance between W and A increases. Turn the knob in the other direction, and the resistance between W and B increases as the resistance between W and A increases. The sum of these two resistances will always add up to the resistance between A and B, which is 10 kΩ for the potentiometer in the Understanding Signals kit.

Figure 2-24: Potentiometer Illustrations from *What's a Microcontroller?*



If you connect the potentiometer's A terminal to Vdd and the B terminal to Vss, you will have a voltage divider output at the W terminal that you can adjust by turning the knob.

> **More Stamps in Class Experiments with Potentiometers.** The potentiometer (pot) is introduced as a variable resistor in *What's a Microcontroller?* Chapter 5, and as a variable voltage divider in *Basic Analog and Digital*, Chapters 1 and 3.

Potentiometer Voltage Parts List

(1) Potentiometer – 10 kΩ
(1) Resistor – 220 Ω (red-red-brown)
(misc.) Jumper wires

Potentiometer Voltage Test Schematic

Figure 2-25 shows a circuit with the pot's A terminal connected to Vdd, its B terminal connected to Vss, and its W terminal connected to the CH1 probe. The divider voltage can be varied simply by turning the potentiometer's knob. If you turn the knob counterclockwise, the pot's W terminal voltage decreases. If you turn it clockwise, the voltage increases. By twisting the knob back and forth over its 270° range of motion, you can set a DC voltage anywhere between 0 and 5 V. Since the CH1 probe is connected to the potentiometer's W terminal, you can verify the voltage with the Oscilloscope's trace cursor, and/or the average voltage in the Measure tab.

✓ Check the potentiometer's legs. If the legs have little angular kinks, you can squeeze them with pliers to straighten them out to improve the electrical contact the legs make when you plug the pot into the breadboard, if needed.

✓ Build the circuit in Figure 2-25.

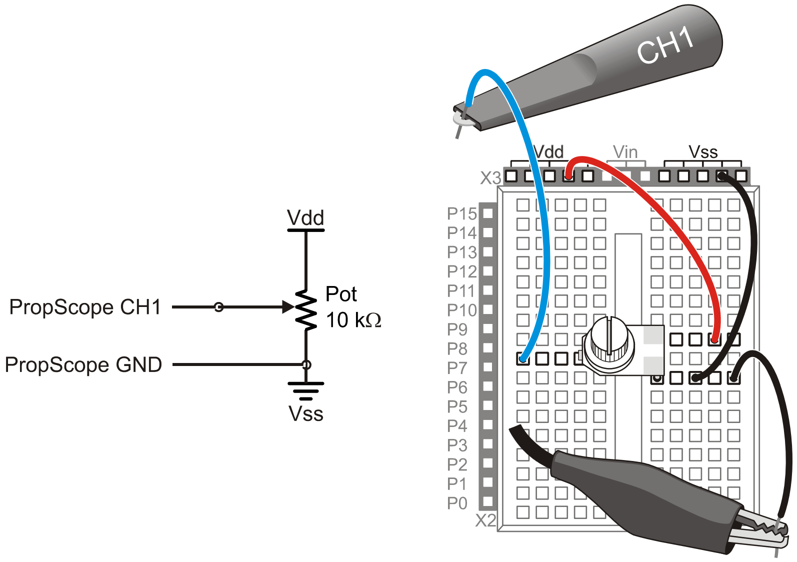


Figure 2-25
Potentiometer Voltage Divider Circuit

Was it necessary to move the ground clip connection? No, the ground clip is still connected to Vss through jumper wires, but it could just as easily be left connected to directly to one of the Vss sockets.

Another Option: Connect ground clips to the plated holes at the corners of your board. Like the Vss sockets, the metal in the plated holes at the each corner of your board are also connected to your board's negative power terminal, so they are also Vss connections.

Potentiometer Voltage Measurement with the PropScope

The PropScope's Vertical dial is still set to 50 µs. That means that the Oscilloscope screen's ten vertical division lines are each 50 millionths of a second apart. The trace line crossing the screen that indicates the voltage is actually a graph of voltage measurements over 500 µs. So, no matter how fast you turn the pot, the voltage trace on the screen will still display as a straight line crossing the screen. The only thing that appears to change as you adjust the pot is the flat trace line's level. In Chapter 3, we will use different time division settings to plot voltage variations over time, but for now, we are just interested in the pot's wiper terminal as a DC voltage level.

> **When you twist the potentiometer's knob,** make sure to press it downward onto the breadboard so that it keeps electrical contact.

- ✓ Rotate the pot's adjustment knob back and forth from one end of its 270° range of motion to the other. (If needed, push down on the pot as you adjust it to maintain its electrical contacts.)
- ✓ In the Oscilloscope screen, observe how the CH1 trace moves up and down.
- ✓ As you adjust the pot's knob slowly from one end of its range to the other, monitor the voltage with the floating cursor and/or the Measure tab's Average voltage measurement. How does the voltage relate to the pot's adjusting knob's position?

Determining I/O Pin Threshold Voltage

A BASIC Stamp I/O pin set to input interprets voltages above 1.4 V as binary-1 and below 1.4 V as binary-0. The 1.4 volt level is called the I/O pin's logic threshold. This threshold voltage level can vary slightly from one I/O pin to the next as well as from one BASIC Stamp to the next. These variations are typically small and are unlikely to exceed a tenth of a volt.

You can use the PropScope to determine your BASIC Stamp module's I/O pin threshold voltage. The circuit will need to be modified so that the pot's W terminal is connected to an I/O pin. Then, run a program to display the voltage state the I/O pin detects. As the pot is adjusted, you can monitor the I/O pin state in the Debug Terminal. When the pot reaches a position that causes the Debug Terminal to report a change in state, the W terminal voltage has reached the I/O pin threshold. At that point, the PropScope will display the threshold voltage.

- ✓ Add the 220 Ω resistor shown in Figure 2-26 to connect the pot's W terminal to BASIC Stamp I/O pin P7.

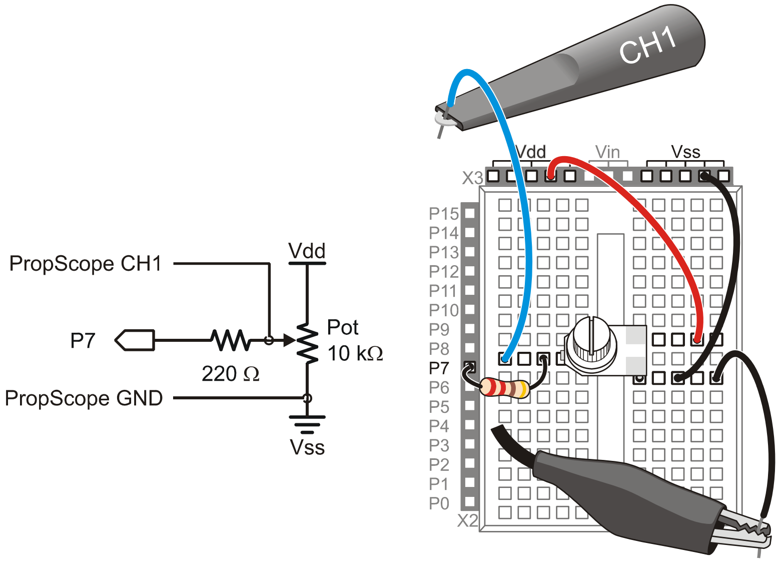


Figure 2-26
Potentiometer Voltage Divider Circuit Connected to a BASIC Stamp I/O Pin

If you are using a HomeWork Board, you can use a jumper wire instead of the resistor.

Why not just connect W to P7 with a wire? The 220 Ω resistor protects the I/O pin from certain prototyping mistakes. For example, imagine that the BASIC Stamp was still loaded with a previous program that sent a high signal to P7, so that pin ends up supplying 5 V to this circuit. If the pot's knob is turned all the way counterclockwise, shorting W to Vss (ground), P7 would try to supply enough current to pull the ground connection up to 5 V, and damage itself in the effort. The 220 ohm resistor limits the current from P7 to Vss in this situation, protecting the I/O pin.

If you have a BASIC Stamp HomeWork Board, there are 220 Ω resistors built-in to protect the I/O pins, and you actually could use a wire instead of the 220 Ω resistor.

The next step in testing the I/O pin's threshold voltage is to write a program that displays the binary state the I/O pin detects (above threshold = binary-1 or below threshold = binary-0). Test Threshold Voltage.bs2 does this with a **DEBUG** command that displays the value of the BASIC Stamp **IN7** register. **IN7** stores the state I/O pin P7 detects.

- ✓ Enter Test Threshold Voltage.bs2 into the BASIC Stamp Editor and load it into the BASIC Stamp.

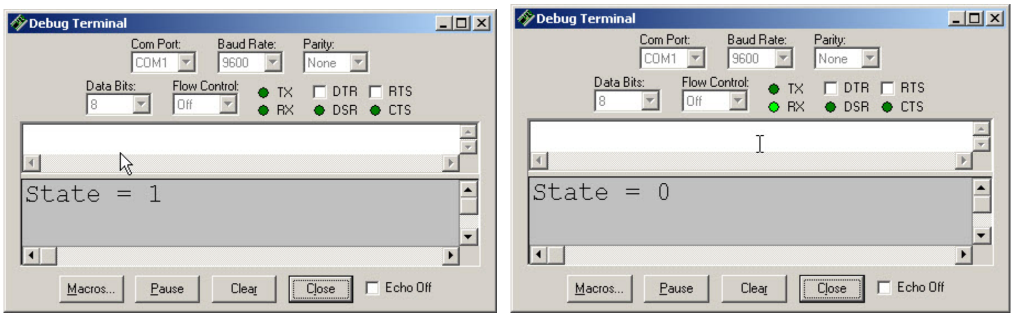
```
' Test Threshold Voltage.bs2
' Display P7 state for testing threshold voltage.

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

PAUSE 1000                                   ' Wait 1 s before values display

DO                                           ' Main loop

  DEBUG HOME, "State = ", BIN1 IN7           ' Display state of signal at P7
  PAUSE 100                                  ' Wait 100 ms

LOOP                                         ' Repeat main loop
```

For more information about this program, work through Chapter 3, Activities #1 and 2 of *What's a Microcontroller?* from the PDF in the BASIC Stamp Editor Help menu.

Figure 2-27 illustrates that when you run the program, it makes the BASIC Stamp send the Debug Terminal the "State = 1" message when the pot's W terminal voltage is above the threshold voltage (left) or the "State = 0" message when the voltage is below the threshold (right).

- ✓ Test this with your potentiometer.

Figure 2-27: Debug Terminal State Displays (above threshold, left; below threshold, right)



Com Port: COM1 Baud Rate: 9600 Parity: None
Data Bits: 8 Flow Control: Off

```
State = 1
```

Com Port: COM1 Baud Rate: 9600 Parity: None
Data Bits: 8 Flow Control: Off

```
State = 0
```

The trick to verifying the approximate threshold voltage is to adjust the pot until you find the point where only a slight adjustment results in a state change. Turn it the slightest amount possible to get a transition from 0 to 1 in the Debug Terminal, then check the voltage in the PropScope while the Debug Terminal displays State = 1. Turn the pot slightly back to transition from State = 1 to State = 0, and measure the voltage in the PropScope again. The approximate threshold voltage is the average of the two values.

- ✓ Try it.

ACTIVITY #6: ADVANCED TOPIC – I/O PIN VOLTAGE VS. CURRENT

A BASIC Stamp I/O pin is designed to send a high voltage of 5 V, or a low voltage of 0 V. The I/O pin can also supply small amounts of current to circuits like indicator lights. In this activity, you will examine the affect of the current a circuit draws on an I/O pin's high signal voltage level. You will also test voltages in the circuit and use them to determine the amount of current the I/O pin supplies to the circuit.

No-Load Test

Recall that a *load* is a circuit that draws current when a voltage is applied to it. Therefore, a *no-load test* would be measuring a voltage without any circuits connected that would draw current. First, let's test an I/O pin's high signal voltage with no load applied. We'll use I/O pin P14. In the previous activity, this I/O pin had a circuit attached to it that allowed it to use binary signals to control an analog voltage output. In this activity, we are looking at the voltages of the raw binary (high/low) signals the I/O pin transmits.

No-Load Parts

(misc.) Jumper wires

No-Load Circuit

Figure 2-28 shows a no-load circuit for testing the high signal voltage transmitted by BASIC Stamp via I/O pin P14.

- ✓ Build the test circuit in Figure 2-28.

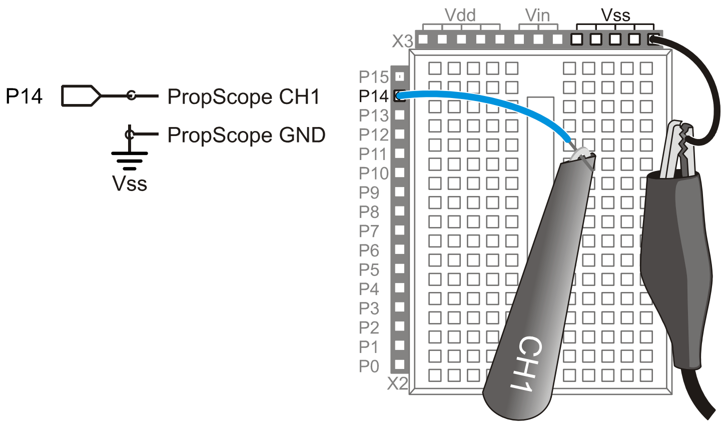


Figure 2-28
No-Load Test Circuit

No-Load Test Code: High P14.bs2

High P14.bs2 is a test program uses I/O pin P14 to transmit a high signal. After the **`HIGH`** command, **`STOP`** prevents the BASIC Stamp from automatically going into low power mode after it runs out of commands.

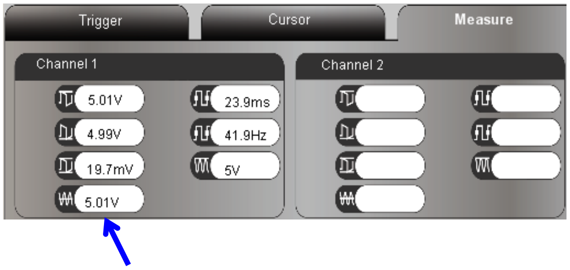
✓ Enter High P14.bs2 into the BASIC Stamp Editor, and then load the program into your BASIC Stamp.

```
' High P14.bs2
' Sends a high signal to P14.

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

DEBUG "Program Running!"                   ' Program running message

HIGH 14                                    ' P14 LED on

STOP                                       ' Halt execution but don't sleep
```

No-Load Test Measurements

Figure 2-29 shows an example of a high signal measurement with no load 5.01 V. Your measurement value may differ slightly.

✓ Use your PropScope to measure your I/O pin high signal with no load, and make a note of it.



5.01V
4.99V
19.7mV
5.01V
23.9ms
41.9Hz
5V

P14 I/O pin HIGH signal voltage

Figure 2-29
P14 High Signal Voltage Measurement – No Load

Signal Load Test Circuit Parts

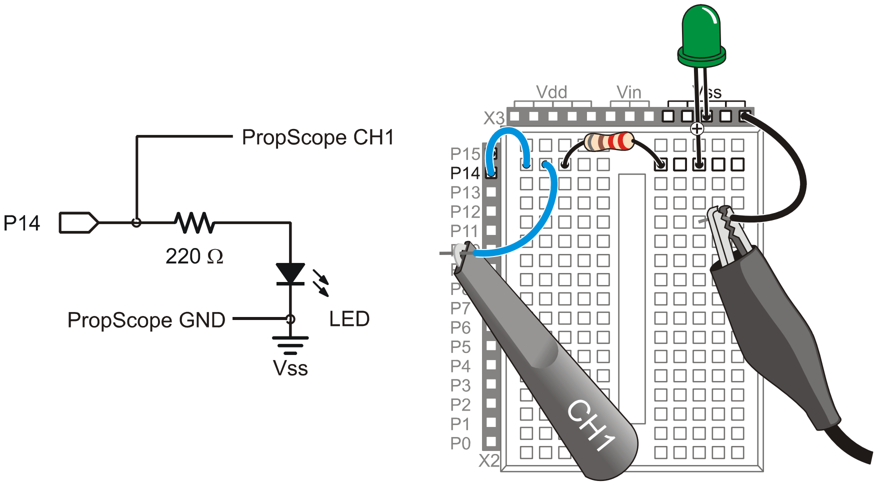
(1) LED – green
(1) Resistor – 220 Ω (red-red-brown)
(misc.) Jumper wires

LED Load Circuit

Figure 2-30 shows a light emitting diode circuit with a probe connected at the junction between P14 and the LED circuit's input. When the I/O pin sends a high signal to this circuit the light will turn on as a result of electric current flowing from the I/O pin, through the circuit, and into Vss.

✓ Build the circuit shown in Figure 2-30, and connect your PropScope probe as shown.
✓ Make sure to plug the LED's shorter pin into Vss, and its longer pin into the same row with the 220 Ω resistor.

Figure 2-30: PropScope Probe Monitoring I/O Pin Voltage with LED Circuit Load

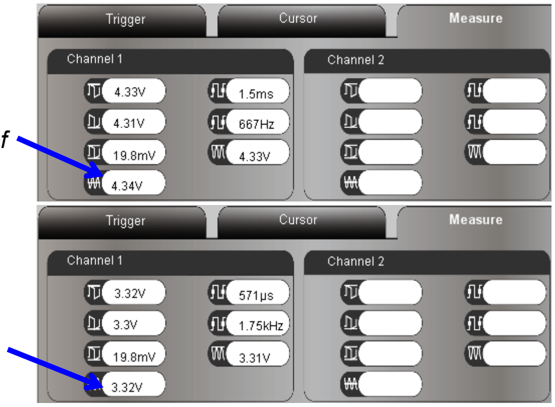


Voltage Measurements

The circuit load on the Board of Education reduced the P14 high signal to 4.34 V (Figure 2-31, top). The same circuit load on the HomeWork Board reduced the voltage to 3.32 V (Figure 2-31, bottom). Why? Keep reading.

✓ Make a note of the high signal voltage measurement with load for your board. Keep in mind that your measurements may be slightly different.

Notice that the same breadboard circuit resulted in different load measurements between the Board of Education (top) and HomeWork Board (bottom). This is due to the additional resistor that is surface-mounted on the HomeWork Board.



4.33V
4.31V
19.8mV
4.34V
1.5ms
667Hz
4.33V

3.32V
3.3V
19.8mV
3.32V
571µs
1.75kHz
3.31V

Figure 2-31
P14 High Signal Voltage Measurement with LED Circuit Load

Board of Education (top)

HomeWork Board (bottom)

Inside "With Load" Voltage Measurements on the Board of Education

The more current a circuit draws, the lower the I/O pin's high signal voltage. Figure 2-32 shows how the I/O pin driver circuit inside the BASIC Stamp Interpreter chip is supplied with 5 V. With the LED circuit's current draw, the I/O pin driver's output is only 4.34 V. If you replace 220 Ω with a larger resistance, like two 220 Ω resistors in series, the LED circuit would draw less current from the I/O Pin. So the I/O pin driver circuit would output a high signal voltage that's closer to 5 V. The CH1 measured voltage would probably be in the 4.6 V neighborhood. A 1 kΩ resistor would make the LED circuit draw even less current, and the CH1 voltage measurement would be even closer to 5 V. Of course, we saw earlier that with no load, the high signal voltage is very close to 5 V.

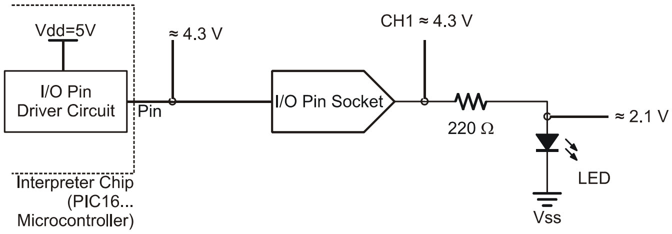


Figure 2-32
I/O Pin Driver's Response to Current Load

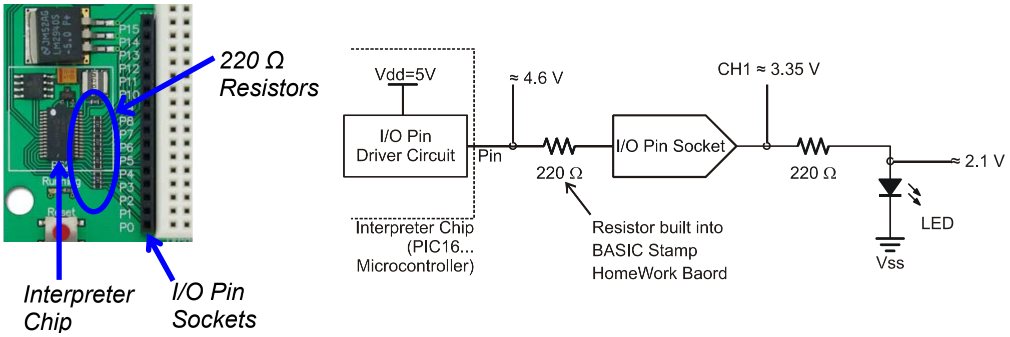
(Board of Education)

Inside "With Load" Voltage Measurements on the HomeWork Board

The BASIC Stamp HomeWork Board has 220 Ω resistors built into the board between the Interpreter Chip I/O pins and the I/O pin sockets you use to make connections to the breadboard. You can see those resistors on the left of Figure 2-33. As mentioned earlier, they help protect the I/O pins from certain prototyping mistakes.

The right side of Figure 2-33 shows the LED circuit on the BASIC Stamp HomeWork Board. Starting inside the Interpreter chip, the LED circuit loads the I/O pin driver, so it only outputs about 4.6 V instead of a full 5 V. The reason the voltage drops to about 3.35 V at the I/O pin socket is because the 220 Ω resistor built into the HomeWork Board is in series with the 220 Ω resistor on the breadboard. The two resistors form a voltage divider, and the output of that voltage divider is right at the I/O pin socket. Recall that the voltage divider output for two equal resistors is ½ the voltage applied to both ends of the circuit. Half the voltage between 4.6 V at the I/O pin driver's output and 2.1 V at the LED's anode is about 3.35 V.

Figure 2-33: I/O Pin Driver and Voltage Divider Responses to Current Load
(BASIC Stamp HomeWork Board)

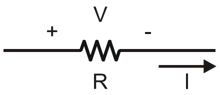


Your Turn: How Much Current Does a Single LED Draw?

In the BASIC Stamp Manual, the BASIC Stamp Model Comparison table states that the BASIC Stamp 2 can “source” up to 40 mA per 8 I/O pin bank, or “sink” up to 50 mA. This current could all be supplied by a single I/O pin or evenly distributed across several I/O pins within a given bank of eight. The I/O pin banks are P0..P7 and P8..P15. Two questions to ask before turning on both LEDs at once are: (1) how much current does the I/O pin supply one LED circuit, and (2) would it be safe to deliver twice that current to supply two LEDs at once?

Source and Sink:: When an I/O pin sends a high signal it acts as a current *source*, applying Vdd = 5 V to the circuit. When an I/O pin sends a low signal, it acts as a current *sink*, applying Vss = 0 V to the circuit.

If you can measure the voltage across a resistor, you can use the resistor’s value and Ohm’s Law to calculate the current passing through the resistor. Ohm’s Law states that the voltage measured across a resistor (V) is equal to the current (I) passing through the resistor multiplied by its resistance (R). Depending on which two terms you know (or have measured), you can rearrange the terms in Ohm’s Law to calculate the third.



$$V = I \times R$$
$$I = V \div R$$
$$R = V \div I$$

Figure 2-34
Ohm’s Law for a Resistor

We know the resistance R in the LED circuit, it is 220 Ω. So, the only other value we need to calculate the current through the resistor is the voltage V across it. One quick way to measure the voltage across the resistor is to measure the voltages at both ends of the resistor and then subtract. The difference between the two voltages is called the *voltage drop* across the resistor.

- ✓ Disconnect the BNC connector from the DAC Card's function generator port, and connect it to the CH2 port.
- ✓ Set the CH2 coupling switch to DC. It's the sliding switch right below the CH2 Vertical dial.
- ✓ Connect the CH2 probe as shown in Figure 2-35.

Figure 2-35: Measure Voltage at Both Resistor Terminals

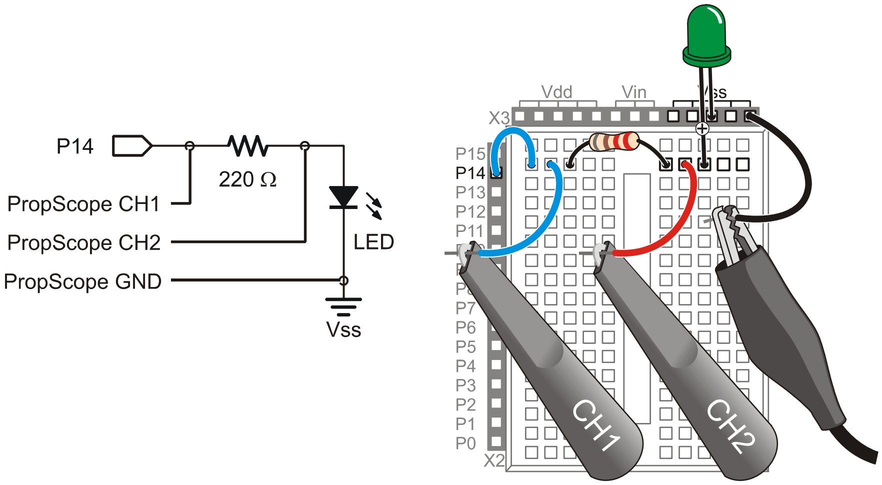
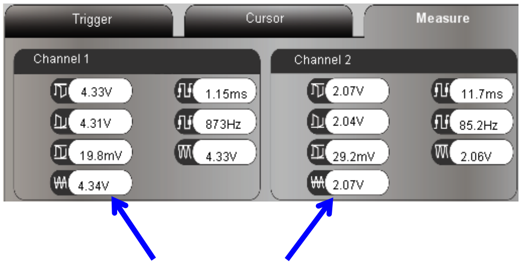


Figure 2-36 shows the resistor voltage measurements and current calculations for the Board of Education, and Figure 2-37 shows the same measurements and calculations for the HomeWork Board. The voltage V_R across the R = 220 Ω resistor in the LED circuit was the Measure tab's Channel 1 Average voltage minus the Channel 2 Average voltage measurements. Knowing both V_R and R, the calculations for $I_R = V_R \div R$ are shown at the right side of each figure.

- ✓ Measure the voltage at both resistor terminals while the LED circuit is connected to the I/O pin and emitting light.

- ✓ Calculate the voltage V_R across the 220 Ω resistor by taking the difference of the voltage at both terminals.
- ✓ Use V_R and Ohm's Law to calculate the current draw for your particular circuit and board.

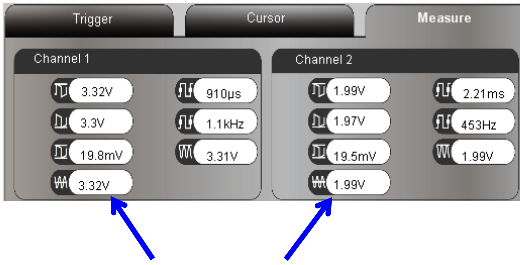
Figure 2-36: Resistor Voltage Measurements and Current Calculations *(Board of Education)*



Resistor Voltage
V_R = 4.34 V – 2.07 V = 2.27 V

$$\begin{aligned} I_R &= V_R \div R \\ &= 2.27\,V \div 220\,\Omega \\ &= 0.01031818...\,V/\Omega \\ &\approx 0.0103\,A \\ &= 10.3\,mA \end{aligned}$$

Figure 2-37: Resistor Voltage Measurements and Current Calculations *(HomeWork Board)*



Resistor Voltage
V_R = 3.32 V – 1.99 V = 1.33 V

$$\begin{aligned} I_R &= V_R \div R \\ &= 1.23\,V \div 220\,\Omega \\ &= 0.00604545...\,V/\Omega \\ &\approx 0.00605\,A \\ &= 6.05\,mA \end{aligned}$$

The first question was, "How much current does the I/O pin supply one LED circuit?" The answer is: about 10.3 mA on a Board of Education, or 6.05 mA on a BASIC Stamp HomeWork Board. The second question was, "Would it be safe to deliver twice that current to supply two LEDs at once?"

The solution is: The BASIC Stamp would have to deliver twice as much current as it does to one LED circuit load and still not exceed 40 mA. For the Board of Education, that would be I = 10.1 mA × 2 = 20.2 mA, which is still which is well below the BASIC

Stamp Manual's stated 40 mA limit. For the BASIC Stamp HomeWork Board, it's I = 5.59 mA × 2 = 11.18 mA, which is even less, and well within the 40 mA limit too. So, in both cases, the BASIC Stamp can safely deliver current to two LEDs at once.

Your Turn: More Current Measurements

- ✓ Repeat by replacing the 220 Ω resistor with a 1 kΩ resistor. Could your BASIC Stamp safely drive eight LEDs with a bank of 8 I/O pins, using 1 kΩ resistors?

SUMMARY

This chapter examined power supply voltages and surveyed a variety of circuits and measurement techniques for fixed DC voltages. DC voltages that were measured included Vdd, Vin, Vss, voltage dividers outputs, and DAC circuit outputs set by the BASIC Stamp, and the PropScope DAC Card's DAC output. The Oscilloscope's voltage divisions, floating cursor, and Measure tab were used to measure these voltages. The effects of circuit load on I/O pin high signal voltages were examined. Higher current draws resulted in lower I/O pin driver output voltages. Ohm's Law was used to determine the current through a circuit by measuring the voltage at each of the resistor's terminals. The voltage across the resistor and its resistance can be used in I = V ÷ R to calculate the current passing through the resistor.

Chapter 3: Human-speed Measurements

HUMAN-SPEED VS. ELECTRONIC SPEED

“Human-speed” and “electronic speed” are not really technical terms, but they can be useful for describing the differences between signals we can and cannot quantify without specialized equipment. Examples of human-speed signals include street lights, musical rhythm, Morse code, and pulse rate. These signals can be quantified either intuitively, with some training, or in the case of pulse rate, with little more than a clock that displays seconds. Some signals seem to fall in both categories, audio tones for example. Different tones played by an audio speaker sound like they have different pitches. So, using just our ears, we can detect fairly subtle variations in the rate at which the speaker vibrates. On the other hand, it would be difficult to actually count the number of speaker vibrations per second to determine the exact frequency of the tone. That would require some specialized equipment.

You may have already programmed the BASIC Stamp microcontroller to send and measure signals at electronic speeds. Examples from *What’s a Microcontroller?* and *Robotics with the Boe-Bot* include controlling servo motors, measuring light or a dial’s position, and communicating with a digital integrated circuit. The signals in those activities were quantified in terms of 2 microsecond (μs) increments. Typical human reaction times might be in tenths or even hundredths of seconds, but it takes a microcontroller or some other piece of “specialized equipment” to send or measure signals in terms of microseconds (millionths of seconds). If the electronic speed signals don’t work as expected, it can also be difficult to diagnose without the aid of a signal measuring device, and that’s where the PropScope proves itself to be exceedingly useful.

Before we move on to measuring electronic speed signals, this chapter provides a survey of many of the more commonly used PropScope measurement tools and techniques, applied to human-speed signals with a variety of circuits. Human-speed signal measurements provide a nice starting point with the PropScope. They make it possible to compare things we can manually initiate and monitor against the PropScope’s graphical display and measuring tools. Since many of the measurement techniques are the same at both human and electronic speeds, taking human-speed measurements first provides an opportunity to compare easily verifiable physical quantities against PropScope measurements. Having already measured similar signals at human-speeds will also make it easier to proceed with confidence with electronic speed measurements in later chapters.

ACTIVITY #1: A POTENTIOMETER'S VARIABLE VOLTAGE OUTPUT

In the previous chapter, a potentiometer was wired as a voltage divider, and its W terminal output was measured as a DC voltage. With the same wiring and a simple adjustment of the Oscilloscope's time scale, you can instead use the PropScope to plot the W terminal's voltage variations against time as you turn the pot's knob back and forth. So, instead of viewing the potentiometer's W terminal as an adjustable DC voltage signal, we will use the PropScope view it as a time-varying voltage signal.

Potentiometer Voltage Parts List

(1) Potentiometer – 10 kΩ
(misc.) Jumper wires

Potentiometer Voltage Test Schematic

Figure 3-1 is a repeat of Figure 2-25 from Chapter 2, Activity #5.

✓ Rebuild the circuit in Figure 3-1.

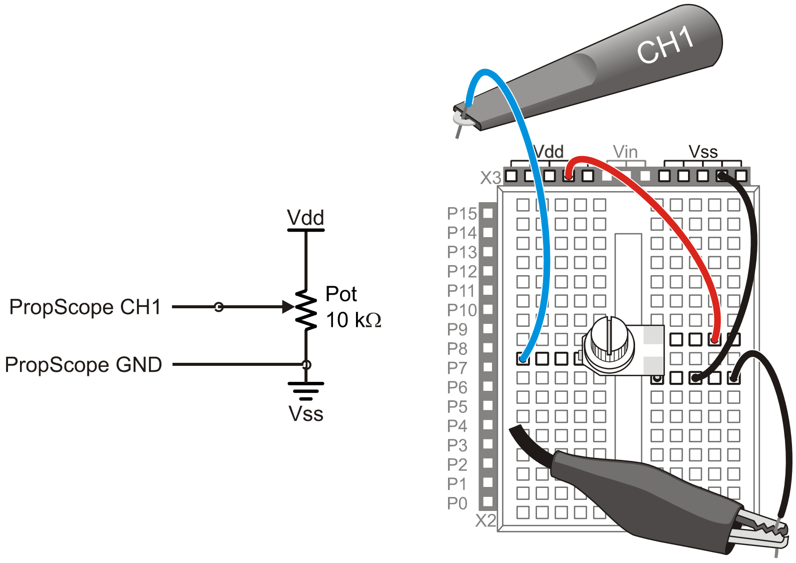


Figure 3-1
Potentiometer Voltage Divider Circuit

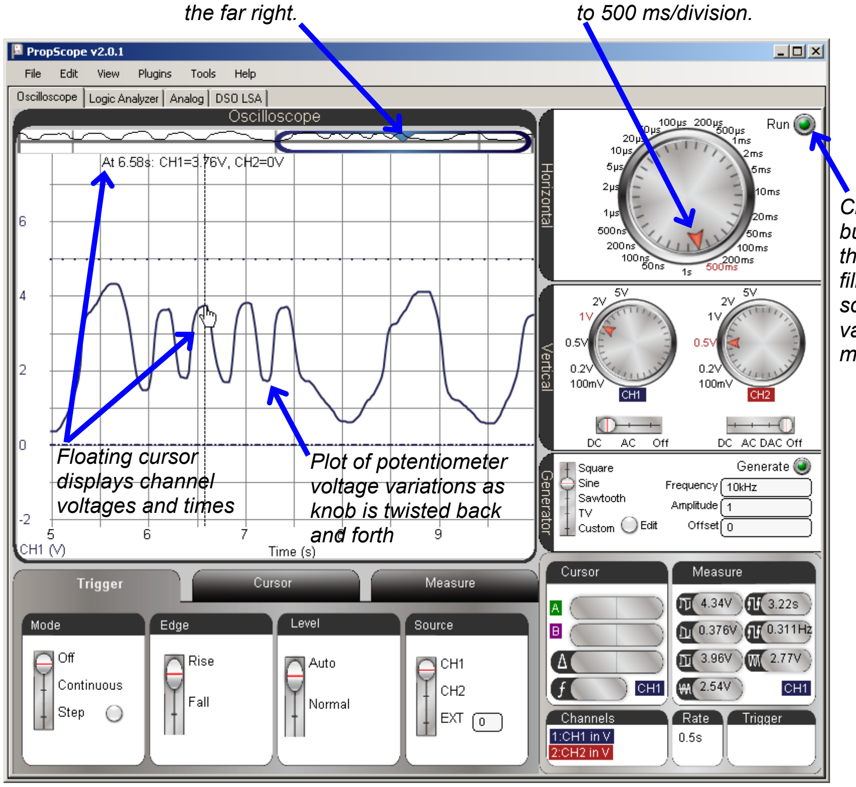
Potentiometer Voltage Test Procedure

Figure 3-2 shows an example of a running history of potentiometer W terminal voltages plotted by the PropScope. The wavy line is a plot of W terminal voltages over 5 seconds as the pot's knob is turned back and forth. Assuming your PropScope is still configured for Chapter 2 measurements, two PropScope settings have to be adjusted for this display. The first adjustment is to turn the oscilloscope's Horizontal dial to 500 ms. This sets the amount of time between two vertical lines in the oscilloscope display to 500 ms. So, the PropScope can display 500 ms worth of voltage measurements per time division. Since the Oscilloscope screen displays ten time divisions, it can display 5 seconds of plotted voltage activity (500 ms/div × 10 div = 5000 ms = 5 s).

Figure 3-2: Test the Voltage Divider Output

Slide the plot area bar to the far right.

Adjust the Horizontal scale to 500 ms/division.



Click the Run button to pause the display after filling the screen with varying voltage measurements.

Floating cursor displays channel voltages and times

Plot of potentiometer voltage variations as knob is twisted back and forth

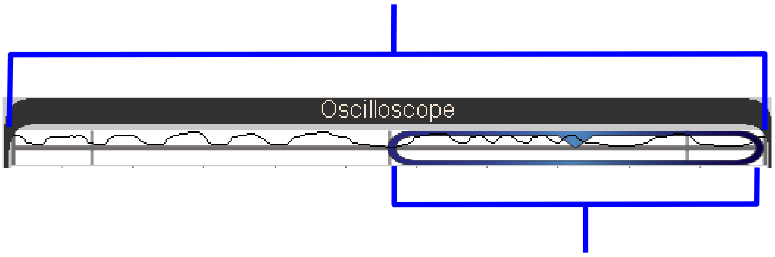
At 6.58s: CH1=3.76V, CH2=0V

When the Horizontal dial is set to 100, 200, or 500 ms, or 1 s/div, the PropScope displays in datalogging mode, which scrolls measurements from right to left. The most recent measurements appear on the right, and all the older measurements shift to the left. The further left on the plot you look, the older the measurements. This is similar to the way older polygraphs and seismographs draw plots, except that they feed paper under a pen that the device moves according to the property it measures. The most recent measurements are drawn by the pen, and as you look further along the paper that passed underneath it, you are looking at older and older measurements.

To view the most recent PropScope measurements in datalogging mode, the second adjustment you'll have to make to get the display in Figure 3-2 is to move the Plot Area bar to the far right of the Plot Preview so that it matches Figure 3-3. The PropScope plots two Oscilloscope screen's worth of measurements, and a "preview" of the entire plot is visible in the Plot Preview. See the squiggly line passing through the Plot Preview? That's the entire two Oscilloscope screens' worth of plotted potentiometer voltages. By positioning the Plot Area bar in the Plot Preview, you choose the portion of the plot the Oscilloscope screen shows you. When you slide the Plot Area bar to the far right, you will see the rightmost portion of the two screens' worth of plotted measurements. At this position, the variations in the plotted voltages will be visible at the right side of the Oscilloscope screen as soon as you make adjustments to the potentiometer.

Figure 3-3: Plot Preview and Plot Area Bar

***Plot Preview**: previews all voltage samples*



***Plot Area bar**:*
selects a portion of the preview bar
for Oscilloscope screen display.

- ✓ Make sure the CH1 coupling switch is positioned at DC.
- ✓ Turn channel 2 off by moving the CH2 coupling switch to Off. (The coupling switches are below the Vertical dials.)

- ✓ Set the CH1 Vertical dial to 1 V.
- ✓ Set the Horizontal dial to 500 ms.
- ✓ Click, hold, and drag the plot either up or down so that the dashed ground line and dotted maximum voltage line are positioned about as shown in Figure 3-2.
- ✓ Click, hold, and slide the plot area bar to the far right of the Plot Preview as shown in Figure 3-3.
- ✓ Check the Run button, and make sure it is bright green, indicating the oscilloscope is running.
- ✓ Start twisting the potentiometer's adjusting knob back and forth (while pressing downward if needed to maintain electrical contact).
- ✓ After the Oscilloscope screen has filled up, click the Run button again to stop the plotting and freeze the waveform.
- ✓ Point at various points on the waveform. The floating cursor should display the voltage and time of the measurement you are pointing at.

Your Turn: Adjust the Visible Portion of the Plot with the Plot Area Bar

As mentioned earlier, the PropScope stores two Oscilloscope screens' worth of voltage samples. It also previews them at the top of the Oscilloscope display in the Plot Preview shown in Figure 3-3. You can use the Plot Area bar to position the Oscilloscope screen over any portion of the Plot Preview, and then view the corresponding portion of the voltage plot in the Oscilloscope screen.

- ✓ Click the Run button to resume plotting voltages.
- ✓ Keep adjusting the potentiometer's knob back and forth until the measurements span the entire Plot Preview.
- ✓ Stop the display again by clicking the Run Button.
- ✓ Try positioning the Plot Area bar at different locations in the Plot Preview.
- ✓ Compare the small plot outlined by the Plot Area bar in the Plot Preview to the full size one in the Oscilloscope display. The portion of the plot outlined by the Plot Area bar should be a miniature of the one in the oscilloscope display.

What if you want to check the voltage of the pot's W terminal once every second? While the Run button is stopped, you can use the time division lines as a guide for where to point the floating cursor to get measurements that are one second apart. Since each time division is 500 ms, two time divisions add up to 1 second's worth of plotted voltage measurements. So, use the floating cursor to point at the plot at every other vertical time division line to check the voltage at one second intervals.

Now, what if you want to check voltages once every second for 10 seconds? You can treat 0 seconds as the leftmost edge of the Oscilloscope screen when the plot area bar all the way to the left, and treat 10 seconds as the rightmost edge when the plot area bar is all the way to the right. You will have to reposition the Plot Area bar at least once to check all ten measurements.

- ✓ The Run button should still be stopped so that you have a frozen plot of potentiometer measurements.
- ✓ Use the Floating Cursor to check the W terminal's voltages at seconds 0, 1, 2, 3, and up to second 10. Reposition the Plot Area bar as needed.

At the far right of the plot, the measurements may end at a value like 9.97 s; you can use that as your 10 second measurement.

ACTIVITY #2: HIGH/LOW SIGNAL VOLTAGES AND FREQUENCIES

Binary signals can be examined with an oscilloscope to get information about both timing and voltages. If only the timing needs to be examined, a device called a *logic analyzer* can be used instead. A logic analyzer doesn't plot actual voltage values, just high and low signal states. In this activity, you will use the Oscilloscope view to monitor both the voltages and the timing of two binary signals. In Activity #3, you will use the PropScope's DAC Card and the PropScope software's Logic Analyzer view to monitor the high/low patterns and timing of four binary signals at once.

Parts List for Probing I/O Pins

(misc.) Jumper wires

Circuit for Probing I/O Pins

The next example program will send high and low signals to the P14 and P15 I/O pins. Figure 3-4 shows how to connect the CH1 and CH2 probes directly to the I/O pin sockets so that you can monitor the signals with the PropScope.

- ✓ Connect the Probes as shown in Figure 3-4.

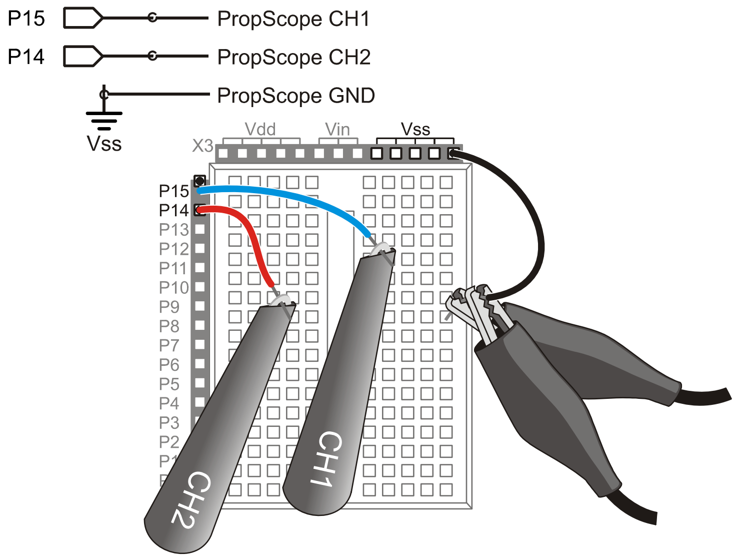


Figure 3-4:
PropScope Probes Connected to Two I/O Pins

Example Program: Alternate High Low Signals.bs2

The PBASIC program Alternate High Low Signals.bs2 contains a loop that starts by sending a high signal to P14 and a low signal to P15. After a 500 ms pause, it changes the P14 signal to low and the P15 signal to high. After another 500 ms pause, the program repeats the same signal sequence. Since the **HIGH**, **LOW**, and **PAUSE** commands are all in a **DO…LOOP**, the sequence repeats indefinitely.

✓ Enter and run Alternate High Low Signals.bs2.

```
' Alternate High Low Signals.bs2
' Alternate the high/low signals transmitted by P14 and P15 and repeat
' indefinitely.  Signal transitions separated by 500 ms pauses.

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

DEBUG "Program Running!"                   ' Program running message

DO                                         ' Main loop

  HIGH 14                                  ' P14 LED on
  LOW 15                                   ' P15 LED off
  PAUSE 500                                ' Wait 0.5 seconds
  LOW 14                                      ' P14 LED off
  HIGH 15                                     ' P15 LED on
  PAUSE 500                                   ' Wait 0.5 seconds

LOOP                                          ' Repeat main loop
```

For more information about this program, work through Chapter 2, Activities #1 and 2 of *What's a Microcontroller?* from the PDF in the BASIC Stamp Editor Help menu.

Oscilloscope Voltage Measurements

Let's examine the signals the BASIC Stamp transmits as it runs Alternate High Low Signals.bs2. Figure 3-5 shows the Oscilloscope view's representation of the high/low voltages transmitted by the BASIC Stamp as it executes the program. The channel 1 trace is monitoring the P15 signal, and the channel 2 trace is monitoring P14. When the channel 1 voltage is high, the voltage is 5 V. When it's low, its 0 V. The same applies to channel 2, except when the channel 2 signal is high, the channel 1 signal is low. The waveform this signal pattern generates is called a *square wave*.

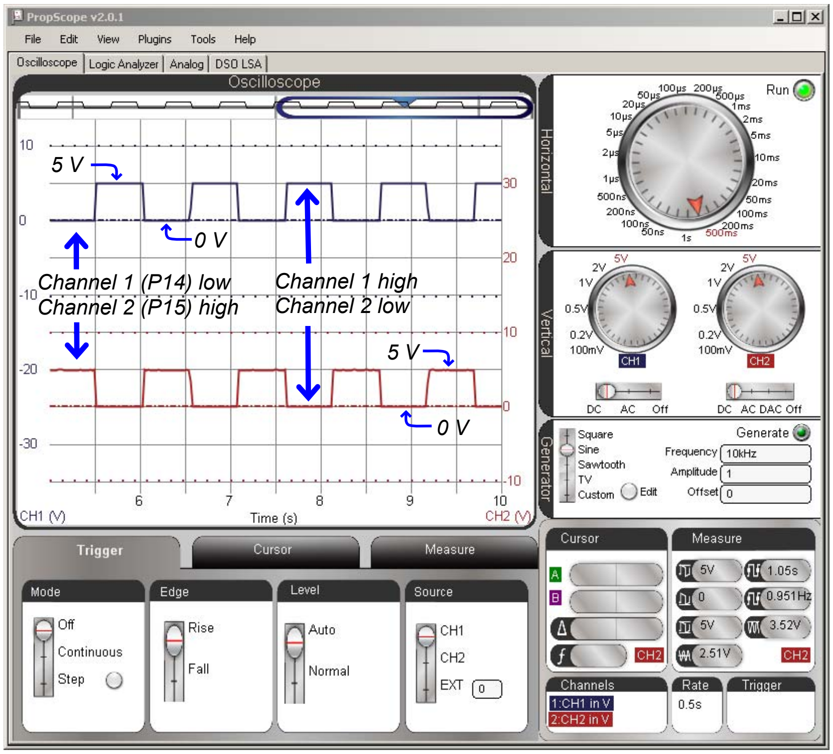
- ✓ If the plot is currently stopped, click the Run button to restart it.
- ✓ Restart the Channel 2 trace by moving the coupling switch below the CH2 Vertical dial from Off to DC.
- ✓ Verify that the Horizontal dial is still set to 500 ms.
- ✓ Set both Vertical dials to 5 V.
- ✓ Click and drag the traces up/down to arrange them as shown in Figure 3-5.
- ✓ Verify that the high signal for each trace is 5 V, and the low signal for each trace is 0 V. (Remember that the voltage scale for the channel 1 trace is on the left of the Oscilloscope screen and the scale for the channel 2 trace is on the right.)

Figure 3-6 shows the Oscilloscope view's Measure tab, which now has more useful voltage information. The highest voltage (Vmax) is 5.01 V, and the lowest voltage (Vmin) is 0 V. The peak-to-peak voltage (Vpp), is the difference between Vhigh and Vlow, and it's 5.01 V. The Average voltage no longer represents a DC voltage. Now it represents the average of the channel's voltage over time, which is 2.52 V in the figure. Since the signal is 5 V half the time, and 0 V, the other half of the time, it stands to reason that the 'average' voltage of half way between these two values would be about 2 ½ volts.

- ✓ Click the Measure tab.

- ✓ Check the voltages for each channel in the Measure tab, and make sure they agree with what you see on the Oscilloscope screen.

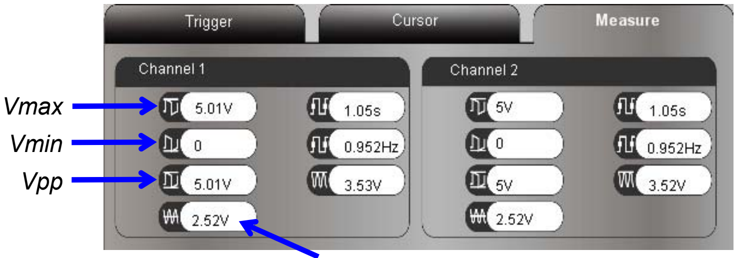
Figure 3-5: Binary Signals in the Oscilloscope view



Frequency 10kHz
Amplitude 1
Offset 0
EXT 0

Measure: 5V 1.05s; 0 0.951Hz; 5V 3.52V; 2.51V CH2

Channels: 1:CH1 in V; 2:CH2 in V
Rate: 0.5s



Channel 1		Channel 2	
5.01V	1.05s	5V	1.05s
0	0.952Hz	0	0.952Hz
5.01V	3.53V	5V	3.52V
2.52V		2.52V	

Average is now ~½ of Vmax – Vmin

Figure 3-6
Measure tab Voltage Info for Binary Signal

Your Turn: A Closer Look at Average Voltage

What happens to the Average voltage if the P14 signal is high *less* than half the time? Try making it high 1/5th of its cycle time—the *cycle time* is the amount of time a signal takes to repeat itself—and see what happens to the Average voltage measurement.

- ✓ Save a copy of Alternate High Low Signals.bs2 under a new name.
- ✓ In the copied program, change the first **`PAUSE`** command to **`PAUSE 100`**.
- ✓ Change the second **`PAUSE`** command to **`PAUSE 400`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Check the Measure tab. Is the Channel 2 Average voltage close to 1 V?
- ✓ How about the Channel 1 signal's Average voltage, is it now close to 4 V?
- ✓ Compare the amount of high time the Channel 1 and Channel 2 traces spend during their cycle times.
- ✓ Try the high time and cycle time values in this equation for average voltage:

$$\text{Average voltage} = 5\text{ V} \times (\text{high time} \div \text{cycle time})$$

- ✓ Reload the unmodified version of Alternate High Low Signals.bs2 into the BASIC Stamp before you continue. Both **`PAUSE`** commands should be **`PAUSE 500`**.

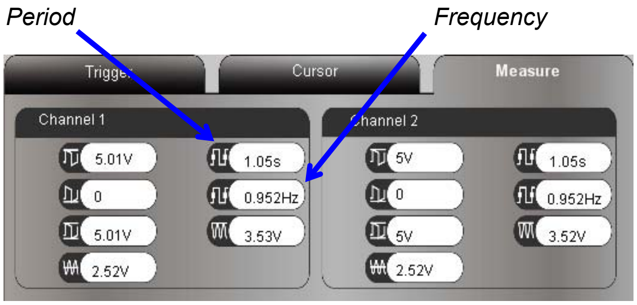
Oscilloscope Frequency Measurements

For a signal that repeats itself periodically (a periodic signal), its cycle time is also called its *period*. Again, that's the amount of time it takes a signal to repeat itself. A signal's *frequency* is the number of times it repeats in one second.

The Measure tab's Period and Frequency measurements for Channel 1 are pointed out in Figure 3-7. The PBASIC code in Alternate High Low Signals.bs2 sets each I/O pin output state, pauses for 500 ms (1/2 a second), then inverts the I/O pin output states and pauses another 500 ms before repeating. Since the program makes the signal repeat after two half-second intervals, its period should be close to 2 × ½ second = 1 second.

The Measure tab provides a quick verification that the program is in fact repeating its signals with a period of one second (1 s). Since the signal takes about one second to repeat itself, the frequency is also close to one repetition per second—one hertz (1 Hz).

Figure 3-7: Measure Tab Frequency Info for 1 Hz Binary Signal



Periodic Signal: A signal that repeats itself at regular time intervals, that is, periodically.

Cycle: A repetition of the signal.

Period: The time it takes for one cycle (repetition) of a periodic signal. You will see the terms *period* and *cycle time* used interchangeably in this book.

Frequency: They number of times in a second a signal repeats itself.

Hertz: A measurement of frequency in cycles per second.

Period (T) is the reciprocal of frequency (f), and vice-versa.

In other words, if you divide period into 1, you get frequency, and if you divide frequency into 1, you get period.

$$f = \frac{1}{T} \qquad \text{and} \qquad T = \frac{1}{f}$$

So, if you know the period is 100 ms = 0.1 second, you can use f = 1 ÷ T to calculate the frequency f = 1 ÷ 0.1 s = 10 Hz. Likewise, if you know the frequency is 10 Hz, divide it into 1, and you will get a period of 0.1 s, which is 100 ms.

Adjust the Signal Frequency, Adjust the Display

Next, let's examine what happens if both **PAUSE** commands in Alternate High Low Signals.bs2 get reduced to **PAUSE 50**.

- ✓ Save another copy of Alternate High Low Signals.bs2.
- ✓ In this version, change both **PAUSE** commands to **PAUSE 50**.
- ✓ Load the modified program into the BASIC Stamp.

Figure 3-8 shows how the higher frequency signals end up looking pretty cramped on the Oscilloscope screen. With 500 ms/division, the Oscilloscope is showing the signal's activity over 5 seconds. In that amount of time, a signal that repeats every tenth of a second ends up repeating 50 times.

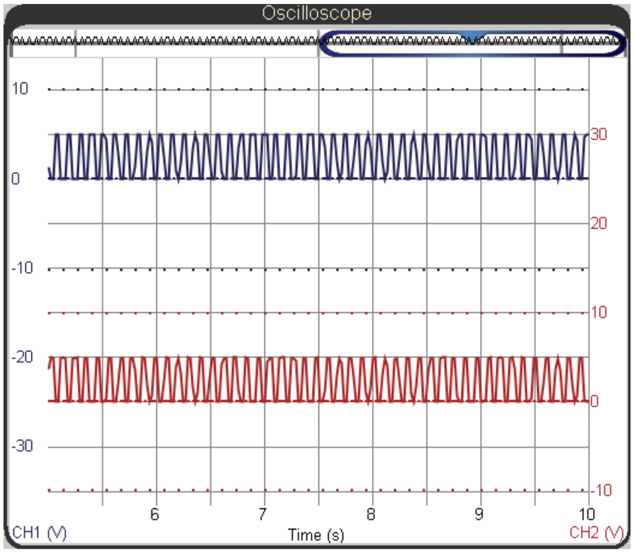


Figure 3-8
50 Cycles on the Oscilloscope screen

Looks a little cramped, doesn't it?

When a signal repeats itself too quickly to see clearly in the Oscilloscope screen, the horizontal dial should be adjusted to a smaller time increment. It's usually best to pick a time increment that accommodates two cycles. Since our square wave now has a period of 1/10th of a second (100 ms), the ideal setting would be one that makes the Oscilloscope screen 2/10ths of a second or 200 ms wide. Remember: the Horizontal dial selects the time per division, which is 1/10th of the Oscilloscope screen. So, to display two cycles, set the Horizontal dial to 1/10th of 200 ms, which is 20 ms.

✓ Stop here and about the main points in the previous paragraph:

Step 1: Figure out how much time a couple of signal cycles will take.

Step 2: Divide that value by 10 to get your Horizontal dial setting. Again, the Horizontal dial sets the time /division, which is 1/10th the width of the Oscilloscope display.

✓ Set the Horizontal dial to 20 ms.

- ✓ Check the time scale at the bottom of the Oscilloscope screen. Is it 200 ms wide?

> **The PropScope is now plotting in oscilloscope Mode, not datalogging mode.**
>
> The PropScope only plots in datalogging mode (from right to left) for the 200 ms/div, 500 ms/div, and 1 s/div.
>
> For all other Horizontal dial settings, the PropScope plots measurements in oscilloscope mode, from left to right. In this mode, the display does not actually scroll. It only updates the display when all the measurements in the plot have been acquired.

Oscilloscope Trigger and Settings

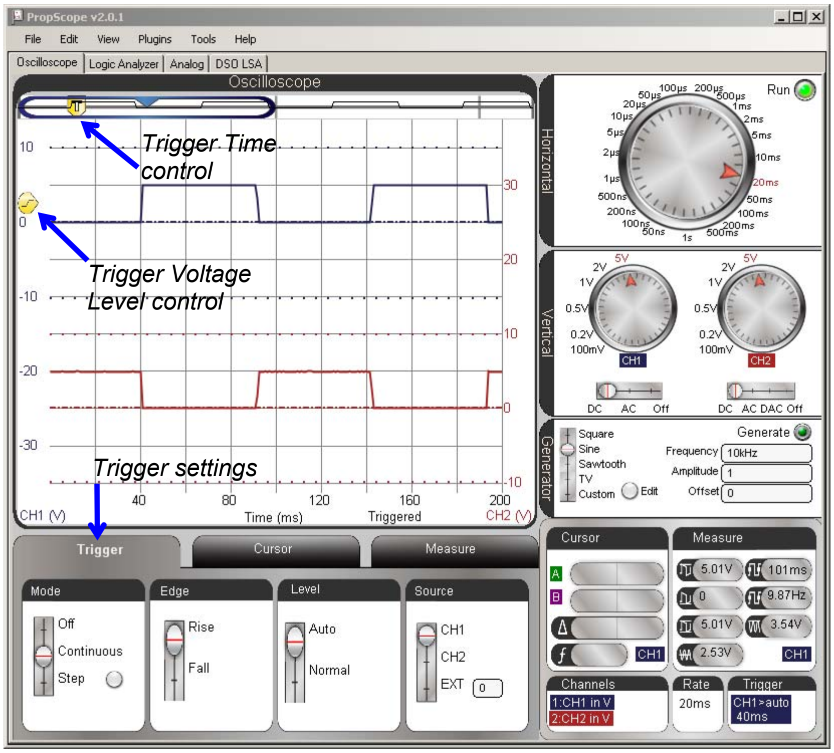
That may have solved cramped display problem, but now the signal isn't staying still in the Oscilloscope screen! Now that the oscilloscope is at a Horizontal setting below 200 ms/div, it is functioning in oscilloscope mode, refreshing the display only after it has acquired all the measurements, and displaying those measurements from left to right. So instead of evenly scrolling, it now skips each time the display updates.

Taking measurements on a waveform that jumps around like that could be nerve wracking. So, oscilloscopes have a built-in feature called a *trigger*. An oscilloscope trigger aligns the plot to an instant when the voltage either rises above or falls below a certain level. It also makes periodic signals like our square wave signal stay still in the Oscilloscope screen. Try this:

- ✓ Click the Trigger settings tab, and set the Mode to Continuous, the Edge to Rise, the Level to Auto, and the Source to CH1.

The display should now stay still in the Oscilloscope screen. Two new controls should have also appeared, the Trigger Voltage level control and the Trigger Time control. These controls set the time and voltage that the signal has to pass through to "trigger" a refresh of the display—a *trigger event*. The Trigger Voltage control should have appeared to the left of the channel 1 trace as shown in Figure 3-9, and the Trigger Time control should have appeared in the Plot Preview. The Plot Area bar may have repositioned to its default position, which is all the way to the left of the Plot Preview, and the Trigger Time control will probably be positioned most of the way to the left in the Plot Area bar.

Figure 3-9: Trigger Setting Adjustments



Trigger Time control

Trigger Voltage Level control

Trigger settings

PropScope v2.0.1
File Edit View Plugins Tools Help
Oscilloscope Logic Analyzer Analog DSO LSA
Oscilloscope
10
0
-10
-20
-30
30
20
10
0
-10
40 80 120 160 200
CH1 (V)
Time (ms)
Triggered
CH2 (V)
Horizontal
Run
50ns 100ns 200ns 500ns 1µs 2µs 5µs 10µs 20µs 50µs 100µs 200µs 500µs 1ms 2ms 5ms 10ms 20ms 50ms 100ms 200ms 500ms 1s
Vertical
5V 2V 1V 0.5V 0.2V 100mV CH1
5V 2V 1V 0.5V 0.2V 100mV CH2
DC AC Off
DC AC DAC Off
Generator
Square
Sine
Sawtooth
TV
Custom
Edit
Generate
Trigger
Cursor
Measure
A
B
CH1
CH1

Frequency 10kHz

Amplitude 1

Offset 0

Mode: Off / Continuous / Step

Edge: Rise / Fall

Level: Auto / Normal

Source: CH1 / CH2 / EXT 0

Measure: 5.01V, 101ms, 0, 9.87Hz, 5.01V, 3.54V, 2.53V

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 20ms

Trigger: CH1>auto 40ms

Let's talk about what each of those Trigger tab settings and the two trigger controls do. The Source is CH1, meaning that the PropScope monitors channel 1 for a trigger event. Since Edge has been set to Rise, that trigger event happens when the voltage rises above a certain voltage. What voltage? With the Level set to automatic, the oscilloscope uses the signal's average voltage to set that voltage. (Yes, it's the same average voltage you would see in the Measure tab's Average voltage field.) As proof, notice on the left that the Trigger Voltage control has been automatically positioned at the halfway point between the channel 1 high and low signals. The Trigger tab's Mode is Continuous, which means as soon it's done plotting measurements, it will start checking for another trigger event.

As soon as the PropScope is done with its current plot, it starts looking for another trigger event. Since the Trigger Voltage level is set to automatic, that'll be at about 2.5 V for the signal we are currently plotting. By adjusting the Trigger Time control, you'll be able to see where the trigger event occurs. Figure 3-10 shows an example of the crosshairs that appears as you point at the Trigger Time control. These crosshairs indicate the Trigger Time and Voltage level. Since the Trigger tab's Edge has been set to Rise with a Source of CH1, the channel 1 voltage passes through the crosshairs as it transitions from low to high.

Point at the Trigger Time control to see the trigger crosshairs. Click, hold and slide left/right to adjust the Trigger Time crosshair's position.

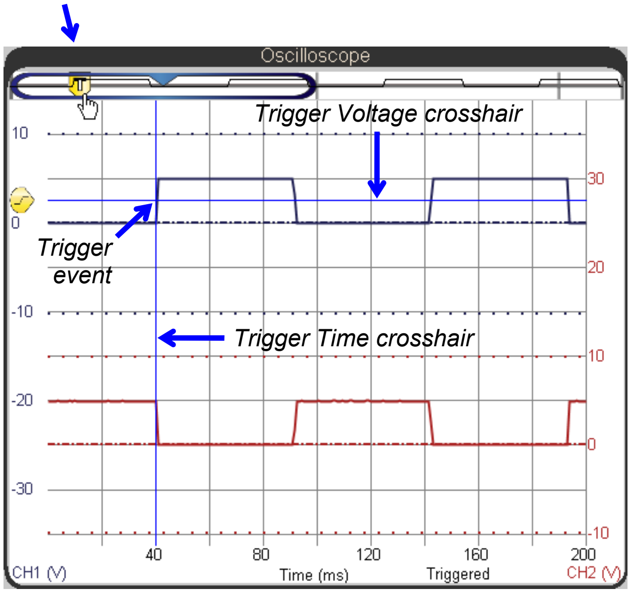


Figure 3-10
Adjusting the Trigger Time

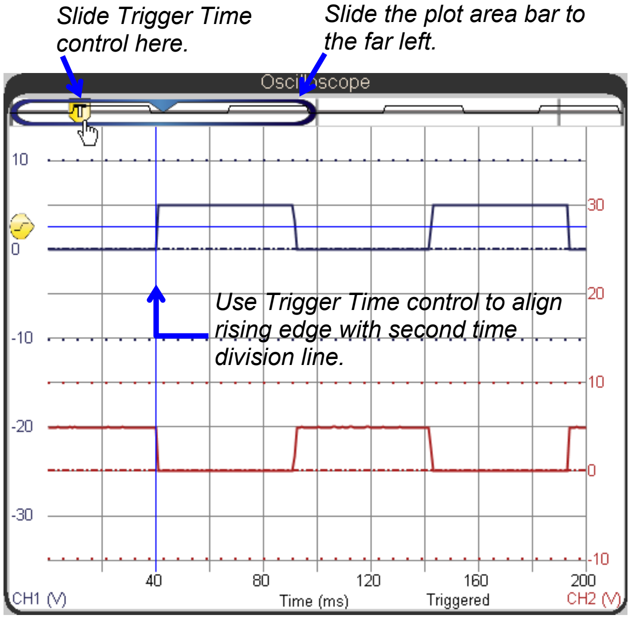
The vertical and horizontal crosshairs that indicate the trigger time and voltage move when you adjust the Trigger Time and the Trigger Voltage controls. Here we are only adjusting the Trigger Time control because the Trigger Level is set to Auto.

- ✓ If the Trigger Time control is not already in the Plot Area bar, click and hold the Trigger Time control, and drag it into the Plot Area bar.
- ✓ Point at the Trigger Time control with your mouse.
- ✓ A pair of crosshairs should appear in the Oscilloscope screen. These crosshairs should intersect at the location of the trigger event.
- ✓ Click, hold and drag the Trigger Time control left and right, but keep it inside the Plot Area bar.

- ✓ As you move the Trigger Time control back and forth, the vertical Trigger Time Crosshair should move with it. The low to high transition in the channel 1 trace should also move with it.

Since typical oscilloscope measurements examine what the signal does after the trigger event, it's usually a good idea to move the Plot Area bar to the far left of the Plot Preview, and to position the Trigger Time control near the left side of the Plot Area bar. This will make the signal that follows the trigger event visible on the Oscilloscope screen

- ✓ If it's not already in that position, slide the Plot Area bar all the way to the left in the Plot Preview.
- ✓ Slide the Trigger Time control to the position shown in Figure 3-11.
- ✓ As you slide the Trigger Time control into the Plot Area bar, the pair of crosshairs should that indicate the trigger voltage and time should reappear.
- ✓ Adjust the Trigger Time control's position slightly to make the vertical trigger time crosshair (and the Channel 1 signal's rising edge) align with the 2nd time division line.


Slide Trigger Time control here.

Slide the plot area bar to the far left.

Use Trigger Time control to align rising edge with second time division line.

Figure 3-11
Adjusting the Trigger Time control's Position

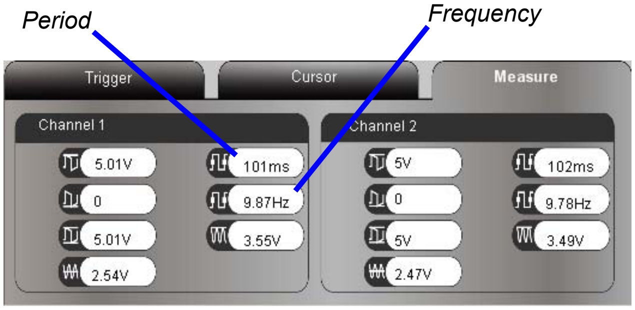
After you slide the Plot Area bar all the way to the left, slide the Trigger Time control to the approximate position shown in the Plot Area bar. Then, slowly adjust it until the vertical trigger time crosshair and the rising edge of the channel 1 trace aligns with the 2nd time division line.

If you ever want to check the current Trigger Time or Voltage settings, you can use your mouse to point at the Plot Preview or any of the scales (numbers around the plot). When you do that, it makes the Trigger Crosshairs appear. The vertical crosshair indicates the trigger time, and the horizontal crosshair indicated the trigger voltage level.

With the Trigger Level set to Auto, the PropScope software positions the trigger voltage level control icon at the signal's average voltage, which is currently half way between the channel 1 low and high voltage levels. Inside the trigger voltage level control icon is a line that transitions from low to high. This indicates that the Trigger Edge has been set to Rise, which in turn means that the PropScope software will wait until it detects a voltage passing through the voltage crosshair's level as it is increasing. The display then aligns that rising edge with the vertical time and horizontal voltage trigger crosshair intersection, and it positions the rest of the plot accordingly.

- ✓ Try adjusting the Trigger Edge to Fall. You can do this by setting the Edge switch in the Trigger tab. What happened to the Channel 1 trigger edge?
- ✓ Change the Trigger Edge back to Rise. See the difference?

Figure 3-12 shows the Measure tab with the modified program still running. The period it displays is 101 milliseconds (ms). Both **PAUSE** command ***Duration*** arguments in the program were reduced from 500 to 50, so the signal should repeat itself about every 100 ms. The time it takes the BASIC Stamp to process and execute all the commands in the **DO…LOOP** that switch the I/O pins on/off apparently add up to an additional millisecond. So, the 101 ms value displayed in the Measure tab's Period field is verifies that the BASIC Stamp program is doing what it's supposed to.



Channel 1: 5.01V, 0, 5.01V, 2.54V, 101ms, 9.87Hz, 3.55V

Channel 2: 5V, 0, 5V, 2.47V, 102ms, 9.78Hz, 3.49V

Figure 3-12
Measure tab Frequency and Period for Channel 1

… with High Low 100 ms Cycles.bs2 running.

A signal that lasts about 1/10th of a second repeats about 10 times in a second. If the signal takes a little longer than 1/10th of a second, the frequency is going to repeat slightly less than 10 times per second, so the 9.87 Hz value in the Measure tab's Frequency field also verifies that the program is working correctly.

Notice that the Channel 1 frequency displays as 9.87 Hz while the Channel 2 frequency, which should be the same, displays at 9.78 Hz. The Measure tab's automated measurements are for getting a rough idea what the signal is doing— a small discrepancy like this is not a big deal. Tools and techniques for more precise measurements will be introduced at various points throughout the book.

Look back at Figure 3-11 on page 82. Notice that each signal has a visible falling edge, a rising edge, and another falling edge. These edges are also called *positive* and *negative* edges. If the Trigger Time control is adjusted so that the second falling edge is not on the Oscilloscope screen, the Measure tab will not update the period and frequency information for the signal. This can also happen if you reduce the time divisions (Horizontal dial) too far.

- ✓ Count the number of falling edges visible for in the channel 1 trace. There should be 2.
- ✓ Slowly adjust the Trigger Time control to the right until the Measure tab's Channel 1 period and frequency measurements disappear.
- ✓ Count the number of consecutive falling edges visible for that signal in the Oscilloscope screen.
- ✓ Repeat for channel 2.
- ✓ When you are done, readjust the Trigger Time control so that it makes the vertical trigger crosshair line up with the second time division line.

Your Turn: Increase the Frequency Again

Setting the Horizontal dial improperly is a really easy mistake to make, and it can lead to incorrect or confusing measurements. So it's a good idea to practice the steps in this activity.

- ✓ Save another copy of High Low 100 ms Cycles.bs2.
- ✓ Repeat the process of increasing the frequency again, this time by reducing the **PAUSE** commands to **PAUSE 5**.
- ✓ Predict the signal period based on a signal that's low for 5 ms and high for 5 ms before it repeats.

- ✓ Multiply your predicted signal period by 2, because we want to start by displaying about 2 cycles in the Oscilloscope screen.
- ✓ Divide the amount of time the oscilloscope should display by 10 to get the per division value for the Horizontal dial setting.

The actual signal period will be slightly longer than predicted because of the time it takes for the BASIC Stamp to process all the commands: **`DO…LOOP`**, **`HIGH`**, **`LOW`**, **`PAUSE`**, etc. This will also make your measured frequency slightly lower than the one you calculated.

ACTIVITY #3: MULTIPLE HIGH/LOW SIGNALS

Many systems have microcontrollers that exchange information with integrated circuits and other microcontrollers using several I/O lines. The fact that the signals are 5 V when high and 0 V when low isn't in question, but the timing of these high and low signals might be causing problems with the data exchange. The tool for measuring high/low signals on multiple lines is called a *logic state analyzer*, which is abbreviated LSA and often referred to as just a "logic analyzer."

Logic analyzers are especially useful for examining binary communication between devices. This type of communication typically involves the exchange of binary values (high/low signals) over multiple signal lines. The upcoming chapters on synchronous and asynchronous serial communication will rely on the PropScope's logic analyzer features. The signals in those chapters occur at electronic speeds, which are much faster than you could observe with, say, an LED.

In this activity, we'll use the logic analyzer with a human-speed example that involves the pushbutton and LED circuits from *What's a Microcontroller?* Chapter 3. The PBASIC program for the BASIC Stamp will monitor the pushbutton circuits and blink one of two indicator lights when one of two pushbuttons is pressed and held. The logic analyzer will be used to monitor and display the signal activity (buttons pressed/not pressed and lights on/off) as the application runs.

For more information about this circuit and program, work through Chapter 3, Activities #1–4 of *What's a Microcontroller?* from the PDF in the BASIC Stamp Editor Help menu.

Pushbutton and LED Parts

(2) Pushbuttons – normally open
(2) Resistors – 10 kΩ (brown-black-orange)
(4) Resistors – 220 Ω (red-red-brown)
(2) LEDs – any color
(misc.) Jumper wires

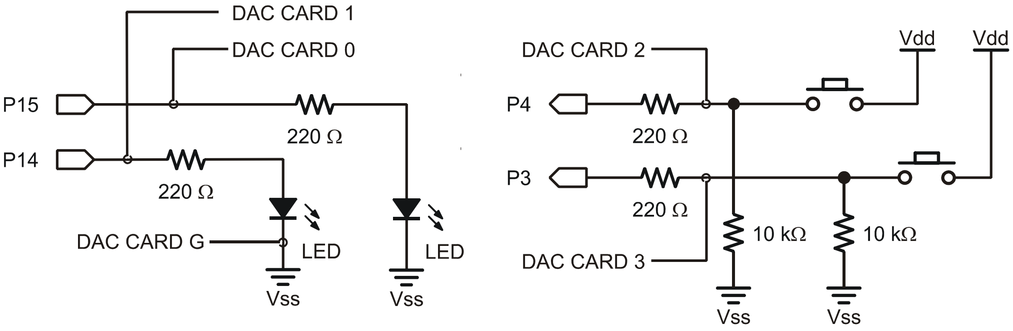
Pushbutton and LED Circuit

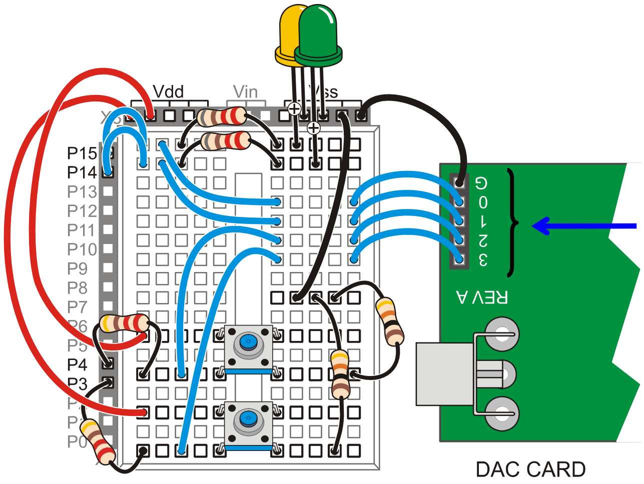
Figure 3-13 shows a schematic of four binary circuits, two LED indicator lights and two pushbuttons. It also shows connections between these circuits and the PropScope DAC Card's four logic analyzer channel inputs, which are the sockets labeled 1 through 4. There is also a mandatory common ground connection, which is the wire between Vss on your board and G on the DAC Card. Vss is your board's ground, and G is the DAC Card's Logic Analyzer ground connection. Figure 3-14 shows a wiring diagram example for the same circuit. These connections will make it possible to view plots of the binary signal exchange between the circuits and the BASIC Stamp I/O pins with the PropScope software's LSA (logic state analyzer) feature.

✓ Build the circuit and make the DAC Card connections shown in Figure 3-13 and Figure 3-14.

> **The DAC Card's 1 to 4 inputs** will be displayed as logic analyzer channels 1 through 4. The socket labeled G is the logic analyzer's ground connection, and it should be connected to your boards Vss ground. This gives the systems a "common ground" so that the logic analyzer has a 0 V reference to compare the voltage signals it measures.

Figure 3-13: Pushbutton and LED Circuit with Logic State Analyzer Connections





Logic State Analyzer Inputs 0..3 and Common Ground (G)

Figure 3-14
Wiring Example for the Figure 3-13 Schematic

The "DAC" in DAC Card stands for *Digital to Analog Conversion* and refers to the card's ability to set voltages and synthesize signals with the function generator. This card is really more of a multipurpose card with DAC being one of its features. Other features include: external trigger—an input that can be used to trigger oscilloscope measurements, a video generator signal output, and a 4-channel logic analyzer input.

When the DAC Card is in use, PropScope CH2 is disabled. You can still take measurements with CH1, just not CH2.

Example Program: PushbuttonControlOfTwoLeds.bs2

This example program from *What's a Microcontroller?* makes the P14 LED blink on/off at 10 Hz while the P3 LED is pressed, or it makes the P15 LED blink on/off at 10 Hz while the P4 button is pressed.

✓ Enter and run PushbuttonControlOfTwoLeds.bs2.
✓ Verify that the P15 LED blinks while the P4 pushbutton is pressed and held.
✓ Verify P14 LED control with the P3 pushbutton.

```
' What's a Microcontroller - PushbuttonControlOfTwoLeds.bs2
' Blink P14 LED if P3 pushbutton is pressed, or blink P15 LED if
' P4 pushbutton is pressed.

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

PAUSE 1000                                   ' Wait 1 second before DEBUG

DO                                           ' Main loop

  DEBUG HOME                                 ' Top-left position in terminal
  DEBUG ? IN4                                ' Display IN4 = value
  DEBUG ? IN3                                ' Display IN3 = value

  IF (IN3 = 1) THEN                          ' If P3 button pressed
    HIGH 14                                  ' P14 LED on
    PAUSE 50                                 ' Wait 1/20th of a second
  ELSEIF (IN4 = 1) THEN                      ' Else if P4 button pressed
    HIGH 15                                  ' P15 LED on
    PAUSE 50                                 ' Wait 1/20th of a second
  ELSE                                       ' Else no buttons pressed
    PAUSE 50                                 ' Just wait 1/20 seconds
  ENDIF                                      ' No more conditions in chain

  LOW 14                                     ' Turn P14 and P15 LEDs off
  LOW 15

  PAUSE 50                                   ' Wait another 1/20th second

LOOP                                         ' Repeat main loop
```

Logic Analyzer Measurements

Figure 3-15 shows a plot of the binary pushbutton and LED circuit activity in the PropScope software's Logic Analyzer view.

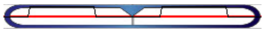
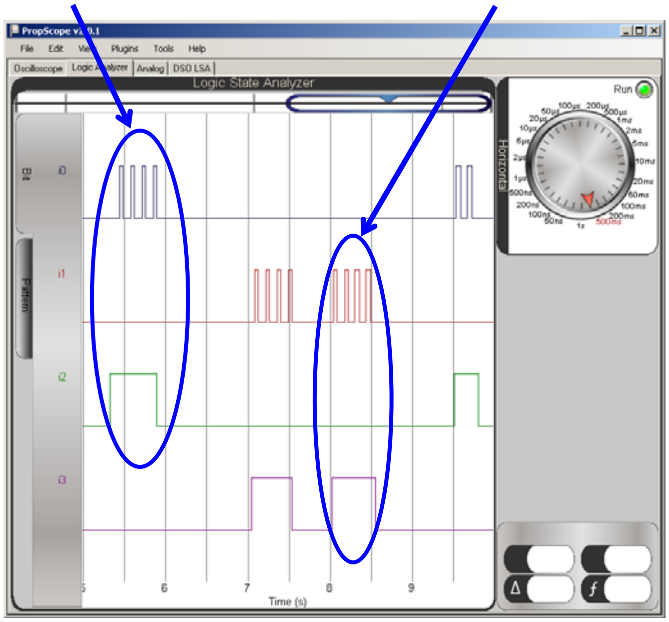
- ✓ Click the Logic Analyzer tab.
- ✓ Set the Horizontal dial to 500 ms.
- ✓ Slide the plot area bar  to the far-right of the Logic State Analyzer's left…right range
- ✓ Try briefly pressing and holding each button, and verify that the Logic Analyzer view correctly reports the binary activity you create on your board.
- ✓ Also watch the LED activity as you press and hold a pushbutton, and make sure to check the Debug Terminal display too.

Figure 3-15: Pushbutton and LED Activity in the Logic Analyzer View

P15 LED (i0) blinks while P4 button (i2) is pressed and held

P14 LED (i1) blinks while P3 button (i3) is pressed and held



Your Turn: Find and Fix the Bug!

PushbuttonControlOfTwoLeds.bs2 has a bug discussed and fixed in *What's a Microcontroller?* Chapter 3, Activity #4. The bug is that only one LED blinks when both buttons are pressed.

- ✓ Use your PropScope's Logic Analyzer to view this symptom as you press and hold both buttons. Only one of the signals monitoring LED circuits should toggle even though both i3 and i4 signals are high.
- ✓ Correct the problem in the PBASIC code, and load the modified code into the BASIC Stamp. Hint: Either break the **`IF…THEN…ELSE…ENDIF`** block into two separate **`IF…THEN…ELSE`** blocks, or look up the other solution near the end of Activity #4 in *What's a Microcontroller*? Chapter 3.
- ✓ Use your PropScope's Logic Analyzer view to demonstrate that the modified code corrected the problem.

ACTIVITY #4: D/A AND FUNCTION GENERATOR WAVEFORMS

The same digital to analog conversion features we used to set DC voltages in Chapter 2, Activity #4 can also be used to synthesize time-varying waveforms.

DAC Parts List

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

DAC Circuit

Figure 3-16 shows a schematic and wiring diagram of one BASIC Stamp D/A conversion (DAC) circuit along with a PropScope probe attached for measuring the DAC output voltages.

✓ Build the circuit shown in Figure 3-16.
✓ Verify that the negative terminal of the capacitor is connected to Vss before reconnecting power to your board.

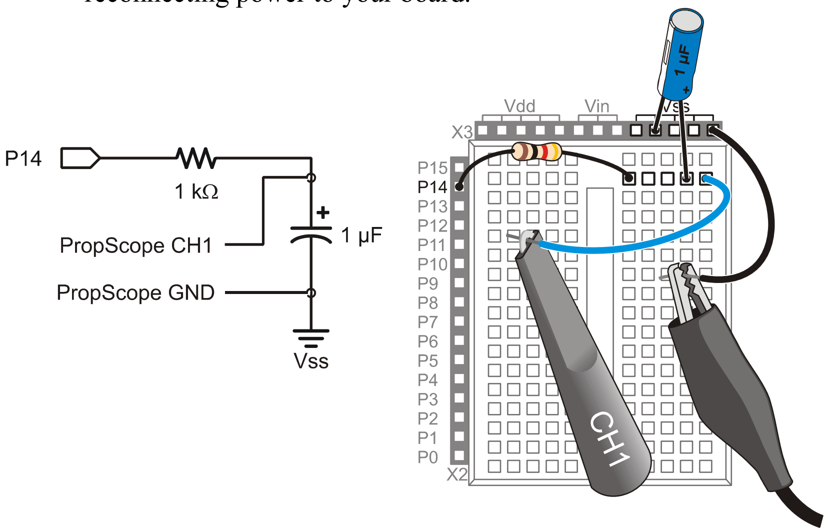


Figure 3-16
One DAC Circuit

Here is a program that repeatedly sweeps the **`PWM`** command's ***Duty*** value from 0 to 255. This in turn sweeps the voltage across the capacitor from 0 to 4.98 V, creating a voltage waveform that can be viewed with the Oscilloscope.

✓ Enter and run Test Saw Tooth.bs2

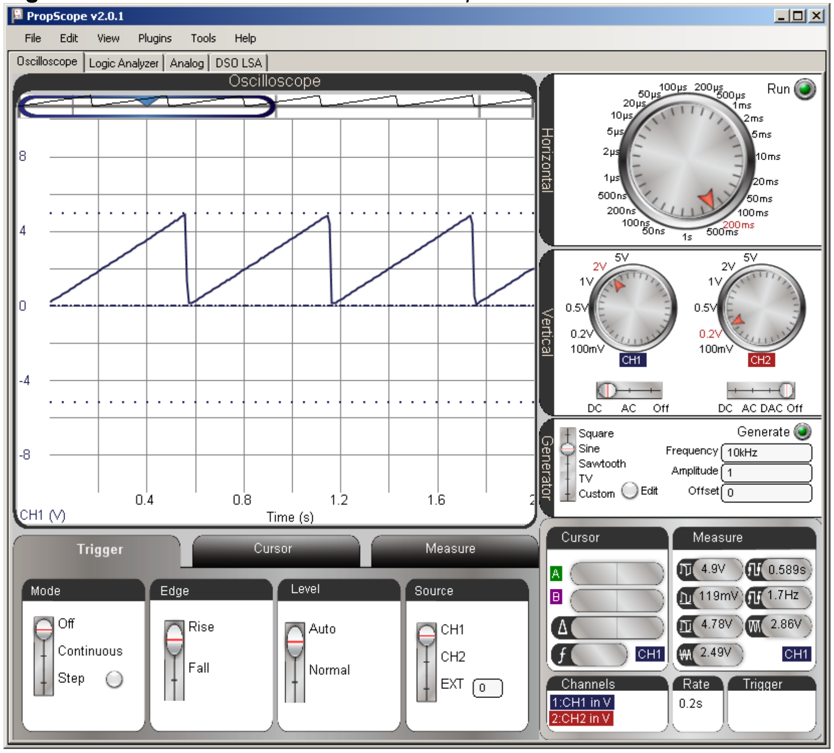
```
' Test Saw Tooth.bs2

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

dacV VAR Byte                               ' Byte variable declaration.

PAUSE 1000                                  ' 1 s before DEBUG
DEBUG "Program running..."                  ' Debug Terminal message

DO                                          ' Main loop
    ' dacV variable sweeps from 0 to 255, DAC output sweeps from
    ' 0/256 to 255/256 X 5 V.
    FOR dacV = 0 TO 255                     ' FOR...NEXT loop repeats 255x
      PWM 14, dacV, 1                       ' dacV sweeps from 0 to 255
    NEXT                                    ' Next repetition
LOOP                                        ' Repeat main loop
```

Isn't it supposed to be `PWM 14, dacV, 5`? According to the 5 × R × C guideline, yes. However, this program is repeatedly applying PWM signals in rapid succession, and one **`PWM`** command only varies slightly from the next. So, this program can get away with a shorter ***Duration*** argument in the **`PWM`** commands.

The Oscilloscope screen in Figure 3-17 shows a 2 second time window, which means that the oscilloscope's Plot Preview spans 4 seconds. This time, the Plot Area bar is on the left with the Horizontal dial set to 200 ms/div. So, it may take a couple of seconds before the waveform appears. The signal will first be visible in the Plot Preview as it makes its way toward the Oscilloscope screen. Your tasks are to:

✓ Change the Trigger tab's Mode setting from Continuous to Off.
✓ Slide the Plot Area bar to the far left of the Plot Preview.
✓ Adjust the Horizontal dial to 200 ms/division.
✓ Adjust the CH1 Vertical dial to 2 V/division.
✓ Set the CH2 Vertical coupling switch to Off.
✓ Wait a couple of seconds for the waveform to make its way from the right of the Plot Preview and into the Plot Area bar and your Oscilloscope screen.
✓ Click the Run button to freeze the display, and then examine the signal.

Figure 3-17: *P14 DAC Test Saw Tooth Output on Channel 1 Trace.*



Your Turn: Sawtooth vs. Triangle Wave

The saw tooth function in Figure 3-17 ramps slowly upward from 0 V to 4.98 V and then dives back to 0 V. In contrast, a triangle wave ramps slowly up, and then ramps slowly back down again. You can add a second **FOR…NEXT** loop after the first one to make the waveform ramp back down for a triangle wave.

- ✓ Save Test Saw Tooth.bs2 as Test Ramping.bs2.
- ✓ Insert this **FOR…NEXT** loop after the first one. This loop should occupy new lines between the **NEXT** and **LOOP** commands in the existing program.

```
FOR dacV = 254 TO 1          ' FOR...NEXT loop repeats 255x
  PWM 14, dacV, 1            ' dacV sweeps from 0 to 255
NEXT                         ' Next repetition
```

- ✓ Load the modified program into the BASIC Stamp, and remember to give the new waveform a couple of seconds to make its way across the Plot Preview and into the Oscilloscope screen.

> **Why does the second `FOR…NEXT` loop count from 254 down to 1?** On the way up, the first **`FOR…NEXT`** loop took the **`dacV`** variable from 0 to 255, so on the way down, the second loop only needs to step from 254 down to 1.

PropScope Function Generator Waveforms

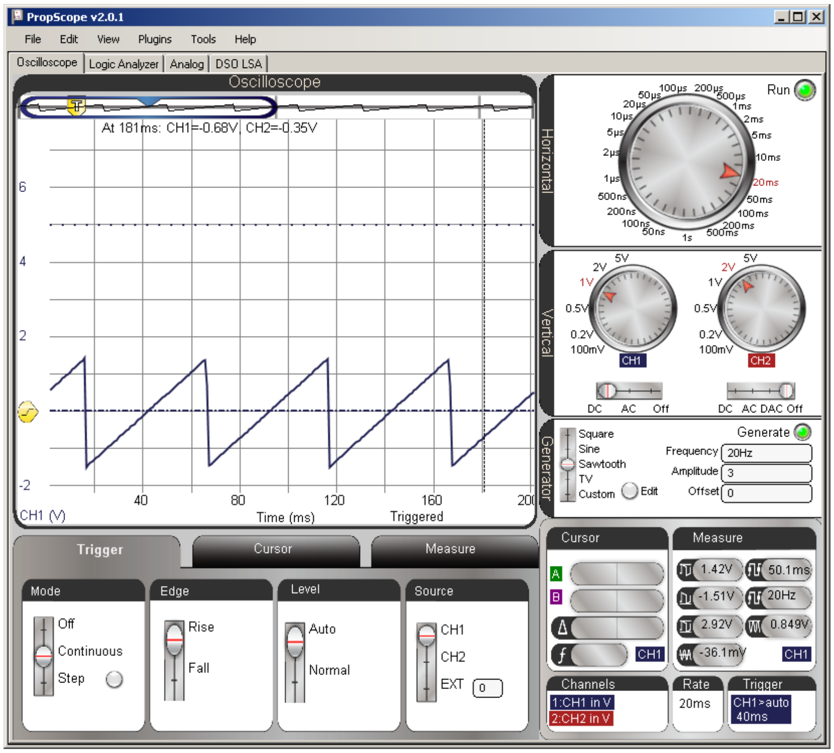
The PropScope's DAC Card and Generator panel can be used to generate square, sine, and saw tooth waves. The Generator panel also has a custom setting that you can use to draw your own waveform with the Edit feature. Many circuit tests involve applying a signal and examining the effect on the output. So, the PropScope's function generator feature is exceedingly useful for applying a signal to the circuit's input. Then, the circuit's output can be examined with the oscilloscope channel 1.

- ✓ Repeat the five checklist instructions in the Set DC Voltages with the PropScope's DAC Card section on page 45.
- ✓ If the Run button is off, click it to restart the oscilloscope.

Figure 3-18 shows a 20 Hz sawtooth waveform generated by the DAC Card's DAC output and monitored and displayed by Channel 1.

- ✓ In the Generator panel, set the slider to Sawtooth, the Frequency to 20 Hz, Amplitude to 3, Offset to 0, and then click Generate.
- ✓ Set the Horizontal dial to 20 ms, and the Vertical CH1 dial to 1 V.
- ✓ Click the Trigger tab and set the Mode to Continuous, Edge to Rise, Level to Auto, and Source to CH1.
- ✓ Adjust the Trigger Time so that the vertical trigger crosshair aligns with the second time division line.

Figure 3-18: Sawtooth Waveform



PropScope v2.0.1

At 181ms: CH1=-0.68V, CH2=-0.35V

Frequency 20Hz
Amplitude 3
Offset 0

1.42V 50.1ms
-1.51V 20Hz
2.92V 0.849V
-36.1mV CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 20ms
Trigger: CH1>auto 40ms

The sawtooth wave in Figure 3-18 has a peak-to-peak amplitude of 3 V. True to its name, the peak-to-peak amplitude is the voltage between the top and bottom voltage peaks. This waveform swings from about 1½ V above the 0 V ground line to about 1½ V below, so its amplitude is a total of 2 × 1½ V = 3 V.

- ✓ Try decreasing the amplitude to 2 V, and observe the result.
- ✓ Try decreasing the amplitude to 1 V, and observe the result again.

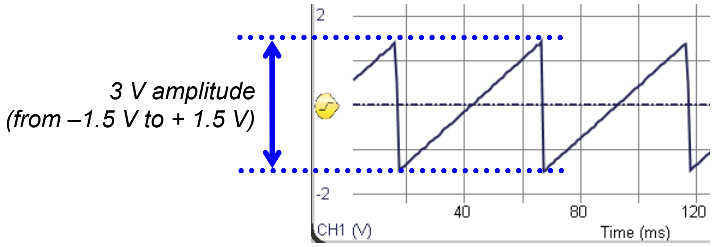


Figure 3-19
Sawtooth Wave
with 3 V Amplitude

The DAC Card can generate signals that fit into one of two ranges:

- –1.5 V to + 1.5 V
- 0 V to 4.7 V

If the amplitude or offset settings exceed the limits, the PropScope software will display an error message. For example, if you tried to enter a 5 V amplitude and a 3 V offset, it would display the error. A 5 V amplitude with a 3 V offset means that the signal should swing from 0.5 V to 5.5 V. That's the DC offset of 3 V + 2.5 V for the top peak and 3 V – 2.5 V for the bottom peak.

The sawtooth wave's Offset setting is currently 0 V. *DC offset* is a DC voltage component that can be added to the signal. Let's see what happens when you add a 2.5 V DC offset to the signal.

✓ Adjust the Generator panel's Offset to 2.5.
✓ You may need to wait a moment for the display to refresh.

The frequency adjustment is also simple. Let's try these values in the Generate panel's Frequency field: 10 Hz, 15 Hz, 20 Hz, 25 Hz, 30 Hz.

✓ Examine the Oscilloscope display as you enter each successive value into the Generator panel's Frequency field. Make sure to press the Enter key on your keyboard after typing each new value.
✓ How does the appearance of the waveform change with higher and lower frequencies?

Your Turn: Other Waveforms

You can make similar adjustments to the Sine, Square, and Custom waveforms. First set the Offset, Amplitude and Frequency values:

- ✓ Set Offset to 2.5 V, Amplitude to 4 V, and Frequency to 20 Hz.

Next, move the slider switch to the different waveforms and observe the change in the display:

- ✓ Set the Generator slider to Sine. Remember to wait a moment for the screen to refresh.
- ✓ Set the Generator slider to Square.
- ✓ Set the Generator to Custom, click Edit, and draw a waveform in the Waveform Editor with your mouse. Click Update to make the DAC Card start transmitting your custom waveform.

ACTIVITY #5: SIMULATED HEARTBEAT

What could be more “Human-speed” than a heartbeat signal? Modern heart monitors are oscilloscopes with special features and specialized probes to filter and amplify the very small electrical signals from contact points on the body. The BASIC Stamp can be programmed to emulate an amplified heartbeat signal with its **`PWM`** command and DAC circuit. Displaying and examining a heartbeat signal like the one in Figure 3-20 with the PropScope provides an example of why it’s important to understand basic manual measurement techniques and not depend too heavily on an oscilloscope’s automated measurements.

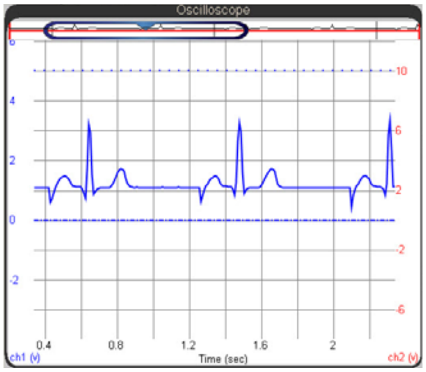


Figure 3-20
BASIC Stamp Generated Heartbeat Signal in the Oscilloscope

Probe Adjustments

This activity relies on the BASIC Stamp DAC circuit from the previous activity.

- ✓ If you have not already done so, reconnect the CH1 probe to the P14 DAC circuit output. (See Set DC Voltages with the PropScope's DAC Card, page 45.)

Example Program: Test Heartbeat.bs2

Test Heartbeat.bs2 emulates a heartbeat signal by fetching successive values from the heartbeat **DATA** directive near the bottom of the program. This **DATA** is stored in an unused portion of the BASIC Stamp module's EEPROM program memory and the **READ** command uses an address of **heartbeat + index** to retrieve each value. The **FOR…NEXT** loop increments the index variable from 0 to 99, so the **READ** command fetches the next value in the list each time through the **FOR…NEXT** loop. The **READ** command also places the value it fetched from EEPROM into the **dacV** variable each time through the **FOR…NEXT** loop. Then, the **PWM** command uses the **dacV** variable to modify the voltage across the DAC circuit's capacitor. Since the entire **FOR…NEXT** loop is nested in a **DO…LOOP**, the signal repeats itself indefinitely.

- ✓ Load Test Heartbeat.bs2 into the BASIC Stamp.

```
' Test Heartbeat.bs2

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

index VAR Byte                               ' Variable declarations
dacV  VAR Byte

PAUSE 1000                                   ' Delay before first message
DEBUG "Program running..."                   ' Debug Terminal message

DO                                           ' Main loop
  FOR index = 0 TO 99                        ' Repeat 99x, sweep index var
    READ heartbeat + index, dacV             ' Read index value from heartbeat
    PWM 14, dacV, 1                          ' Fetched READ value sets DAC
    PAUSE 5                                  ' Wait 5 ms before next iteration
  NEXT                                       ' Next iteration
LOOP                                         ' Repeat main loop

' Hand digitized values from an ECG printout.
heartbeat DATA 060, 063, 067, 069, 075, 077, 080, 082, 082, 080,
               075, 070, 063, 061, 061, 061, 061, 061, 061, 061,
               054, 044, 035, 130, 229, 183, 085, 013, 037, 060,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 062,
               069, 075, 081, 086, 090, 093, 095, 092, 086, 069,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 060,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 060,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 060,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 060,
               060, 060, 060, 060, 060, 060, 060, 060, 060, 060
```

The **`DATA`** table in Test Heartbeat.bs2 was developed by treating an electrocardiograph printout as a series of values plotted on a graph. The graph used a vertical scale of 0 to 255, so that it would fit nicely into the **`PWM`** command's ***Duty*** argument for generating voltages from 0 to 255 (0/255^{ths} to 255/255^{ths} of 5 V). Some head and foot room was incorporated into the vertical scale so that the DAC output did not reach all the way to 0 or 255. The lowest value is 13 and the highest is 229.

Measure the Simulated Heartbeat Signal

Figure 3-21 shows the PropScope settings to display the heartbeat. Remember that you can adjust the waveform's vertical position by sliding it up/down with your mouse. Just point at either the waveform or the dashed ground line for that channel, then click, hold, and drag.

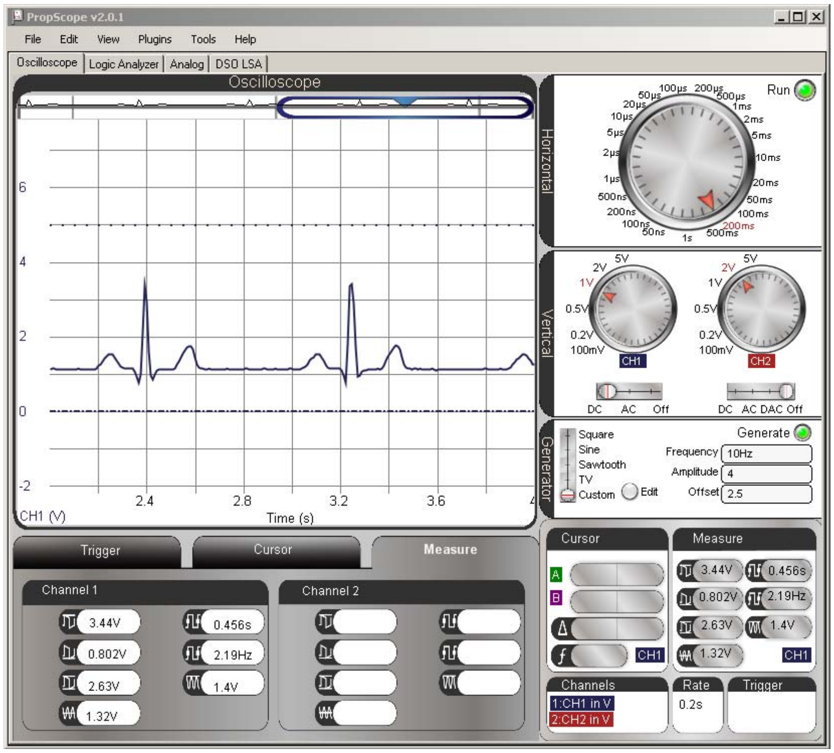
✓ Update your PropScope settings to match Figure 3-21. Pay close attention to the Horizontal and CH1 Vertical scales, the Plot Area bar's position, and the vertical position of the waveform.

✓ If the Run button is off, click it to restart the display.

✓ Locate the frequency for Channel 1 in the Measure tab. What's the patient's heart rate according to the automated Measure tab?

The Trigger tab's Mode setting automatically turns off when the Horizontal dial is set to 200 ms or larger. That's because the oscilloscope goes into datalogging mode with time scales of 200 ms/div or larger, and datalogging mode is not compatible with triggers. A trigger aligns the rest of the plot around some trigger condition in oscilloscope mode. In datalogging mode, the oscilloscope scrolls continuously, so it cannot be forced to align with a particular trigger event.

Figure 3-21: Pulse Rate in Measure Tab is Not Correct



Channel 1: 3.44V, 0.802V, 2.63V, 1.32V, 0.456s, 2.19Hz, 1.4V

Measure: 3.44V, 0.802V, 2.63V, 1.32V, 0.456s, 2.19Hz, 1.4V, CH1

Frequency 10Hz

Amplitude 4

Offset 2.5

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 0.2s

Beware of Automated Measurements

If we rely on the Measure tab for this patient's heart rate, it would be a frequency of 2.19 Hz. That's 2.19 beats per second × 60 seconds/minute = 125.4 beats per minute. That's a reasonable heart rate for exercise, but if this patient is in a hospital bed, it could indicate a problem. Fortunately for our patient, this measurement is not correct.

From the PropScope's standpoint, it is an understandable measurement because the PropScope correctly calculated the average voltage of this signal to be 1.32 V, and it is counting the number of times per second the voltage crosses this level. However, that's not the right measurement for a heart rate. This signal needs to be measured manually.

Time Division and Vertical Cursor Measurement Examples

If we instead measure the time between the highest points in the heart rate signal, we can get the correct pulse rate. Let's look at two different ways to take the manual measurements: by time division and with floating cursor measurements.

Count Time Divisions

The period of the pulse signal can be approximated by counting the number of 200 ms divisions between the signal peaks. It looks like they are just over four 200 ms time divisions apart; let's call it 4 ⅛. This value along with the fact that frequency is the reciprocal of period (f = 1/T) is enough information for the pulse rate's frequency in Hz. Then, to convert from Hz to the familiar beats per minute, just multiply by 60 seconds/minute.

Step 1: Figure out the signal period T based on the number of divisions between peaks.

$T = 4.125 \text{ divisions} \times 200 \text{ ms/division} = 825 \text{ ms}$

Step 2: Use the period to calculate the frequency in Hz.

$f = 1 \div T$
$= 1 \div 825 \text{ ms} = 1 \div (825/1000 \text{ s}) = 1000 \div 825 \text{ s} = 1.2121\ldots \text{ Hz}$

Step 3: Multiply the Hz measurement (beats per second) by 60 seconds/minute get the familiar beats per minute (bpm) heart rate measurement.

$f(\text{bpm}) = f(\text{Hz}) \times 60 \text{ seconds/minute}$
$= 1.2121\ldots\text{beats/s} \times 60 \text{ s/m} = 72.7272\ldots\text{beats/m}$
$\approx 72.7 \text{ bpm}$

Alright, our patient isn't so bad off after all.

Your Turn: Take a More Precise Measurement

This time division approximation of the heart rate might seem a little vague and imprecise.

✓ Try pointing at the top of adjacent peaks with your mouse, and record the times with the floating cursor.

✓ Subtract the smaller time from the larger time and use the result to recalculate the pulse rate.

You can also improve the precision by zooming in on each feature by adjusting the Horizontal dial to smaller values and then moving the screen around to find a more precise time of each event.

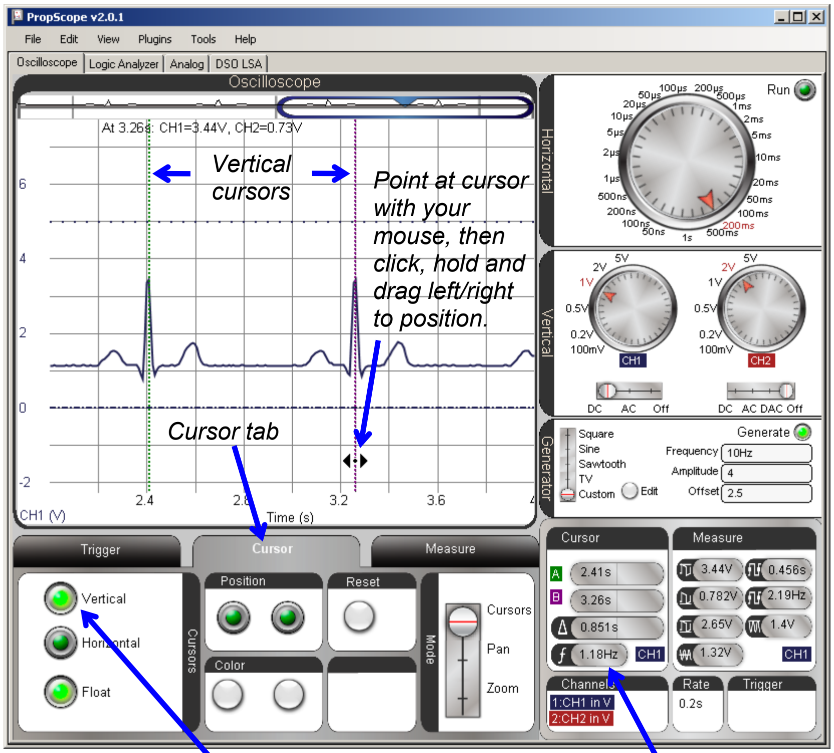
✓ Try it.

Use Vertical Cursors to Measure the Period

Oscilloscopes (including the PropScope) have tools called *cursors* to manually measure waveforms. So far, we have only used the PropScope's floating cursor feature, which gives you an instant readout of each channel's voltage value for a single point in time. In contrast to a single point, cursors are pairs of horizontal and vertical lines that can be positioned on the Oscilloscope screen over a waveform. Instead of simple time and voltage values, the Cursor display also gives information about lines relative to each other: voltage difference, time difference, and frequency calculations.

Figure 3-22 shows an example of the PropScope's vertical (time) cursors positioned over the heartbeat signal peaks. In the Cursor Display on the Oscilloscope view's lower-right, it shows both the time of each cursor's positions along with automated Δ and f measurements. The Δ measurement is pronounced "delta" and is shorthand for the time difference Δt (delta-t) between the vertical cursor lines. When applied to a signal that repeats itself, Δ is also the signal's period T. The f measurement is 1 ÷ Δ and is often used to automate the f = 1 ÷ T calculation. Note that the Cursor measurements panel indicates the frequency is 1.18 Hz. Multiplied by 60 seconds/minute, we get 70.8 bpm.

Figure 3-22: Cursors Placed over Pulse Peaks to Measure Pulse Rate



At 3.26s: CH1=3.44V, CH2=0.73V

Frequency 10Hz
Amplitude 4
Offset 2.5

Cursor: A 2.41s; B 3.26s; Δ 0.851s; ƒ 1.18Hz CH1

Measure: 3.44V; 0.456s; 0.782V; 2.19Hz; 2.65V; 1.4V; 1.32V; CH1

Channels: 1:CH1 in V; 2:CH2 in V

Rate: 0.2s

Vertical cursors

Point at cursor with your mouse, then click, hold and drag left/right to position.

Cursor tab

Click Vertical button in Cursors subpanel to display the vertical cursors.

Cursor Display: View Cursor measurements

Here is how to position cursors to take measurements like those shown in Figure 3-22:

- ✓ Click the Cursor tab below the Oscilloscope screen.
- ✓ In the Cursors tab, click the Vertical button to make the time cursors appear on the oscilloscope. They will appear as two vertical lines.
- ✓ Click either the waveform or the CH1 ground line to select CH1 as the active channel in the Cursor display.

- ✓ Point at one of the cursor lines with your mouse, and it will change to a double-arrow that points left and right. Click, hold, and move the cursor left or right to place it over one of the heartbeat signal peaks.
- ✓ Repeat with the other cursor and an adjacent heartbeat signal peak.
- ✓ Check the f measurement in the Cursor display near the Oscilloscope screen's lower-right corner.

Your Turn: Measure Voltage Differences with Horizontal Cursors

Figure 3-23 shows horizontal cursors used to measure the voltage difference between the tops of the heartbeat signal's less-prominent high points, the P and T wave peaks.

- ✓ Click the Horizontal button in the Cursor panel to activate the voltage cursors.
- ✓ Click the Vertical button deactivate the time cursors.
- ✓ Adjust the horizontal voltage cursors so that they measure the difference between the P and T peaks as shown in Figure 3-23.
- ✓ Check the voltage difference in the cursor display. It's near the lower-right corner of the Oscilloscope plot. It's the value that ends with V by the Δ symbol in the Cursor display.

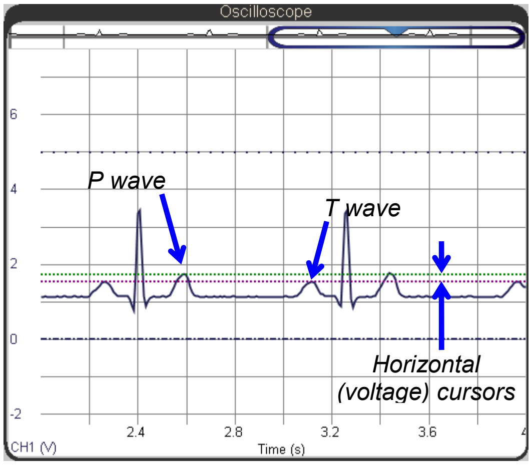


Figure 3-23
Voltage Cursors Measuring Difference between P and T waves

- ✓ Also, try using both horizontal and vertical cursors at the same time to measure both the voltage and time differences between the peaks of the P and T waves.

SUMMARY

This chapter surveyed a variety of circuit, microcontroller programming, and measurement techniques for time varying signals. Human-speed, time-varying signal measurements were applied to:

- Binary pushbutton and LED signals viewed with the logic analyzer
- Varying potentiometer wiper voltage displayed with the Oscilloscope
- BASIC Stamp D/A and PropScope function generator signals viewed with the Oscilloscope.

The PropScope's function generator was also used to generate signals with certain amplitudes, offsets, and frequencies.

The oscilloscope's Horizontal time and Vertical voltage display dials were used to adjust the time/division and volts/division settings. The Oscilloscope's divisions were used to measure voltage, time and frequency, and the measurements were compared to values obtained from the Measure tab and from cursor placement.

Chapter 4: Pulse Width Modulation

Modulation is the process of varying a signal's properties for communication or control. The property that *pulse width modulation* varies is the amount of time a signal spends at a certain active voltage before returning to a resting voltage. When viewed with an oscilloscope, a signal that holds a voltage for a certain amount of time has a certain width on the Oscilloscope screen.

Figure 4-1 shows a few examples of how these voltage pulses might look in an oscilloscope. On the left, two different positive pulse examples have 0 V resting states and a 5 V active states. If the signal spends a brief time at 5 V, the pulse looks narrower on the Oscilloscope screen. If the signal spends a longer time at 5 V, it looks wider. On the right, the negative version of those same pulses simply have opposite active and resting states.

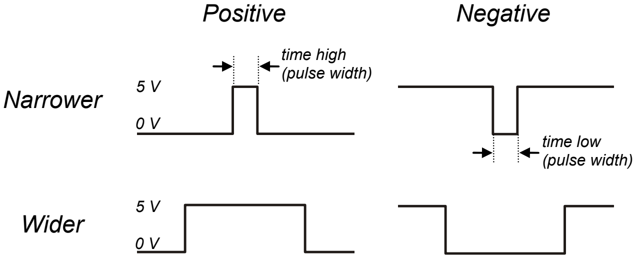


Figure 4-1
Voltage Pulse Examples

> **Active and resting voltages can vary from one system to the next.** For example, if the BASIC Stamp transmits pulses, it sends 5 V high signals and 0 V low signals, but another system might depend on different active and resting state voltages. Certain circuits called *translators* can translate the BASIC Stamp module's outputs to other voltages. Circuits called *drivers* also offer high output current capabilities for motor control, and circuits called *inverters* can change positive pulses to negative pulses.

PULSES FOR COMMUNICATION AND CONTROL

Pulse width modulation is widely used for both motor control and communication. This chapter's activities include PWM examples for each, and also examine how to use the PropScope's features to measure these signals.

ACTIVITY #1: PULSES FOR SERVO CONTROL

Hobby servos like the one in Figure 4-2 control the steering and throttle in radio controlled (RC) 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 (#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 an RC 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, an RC boat rudder has to push against the water flowing around it to make the boat turn in the water.

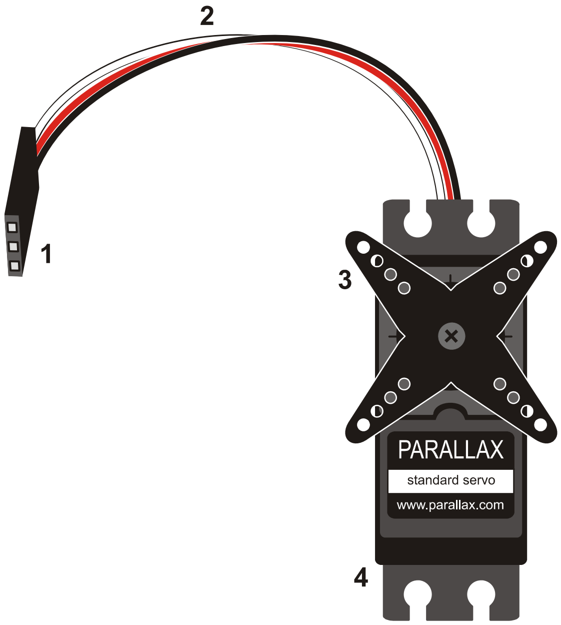


Figure 4-2
Servo

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

Driving an RC vehicle usually 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, a joystick moved left and right in the radio control transmitter unit might make the steering servo in an RC car rotate left and right. Also, if the joystick is held in a certain position, the servo holds its horn at the corresponding position and resists opposing forces.

The radio transmitters in these systems code 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 these 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.

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.

Servo Test Parts

(1) Parallax Standard Servo
(1) 3-pin header (HomeWork Board only)
(misc.) Jumper wires

Servo Test Circuit

Figure 4-3 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.

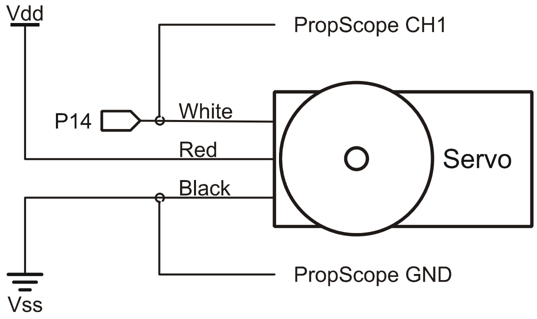


Figure 4-3
Servo Schematic

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 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 revisions). The instructions for this board start right after this checklist.
  - o BASIC Stamp HomeWork Board (Serial Rev C and newer, USB all revisions) starts on page 110.
  - 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, page 111.

Board of Education (Serial Rev C and newer, USB all revisions)

There's a jumper that connects two of three pins together between the X4 and X5 servo ports. In Figure 4-4, the jumper is shorting the upper two pins to set 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 4-4.

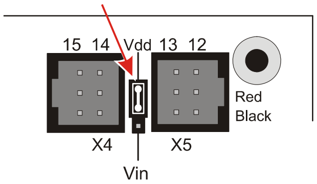


Figure 4-4
Power Jumper Between Servo Headers

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 4-5.

- ✓ Make sure that the color coded servo cable wires line up with the White, Red, Black labels shown in Figure 4-5.
- ✓ 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 111.

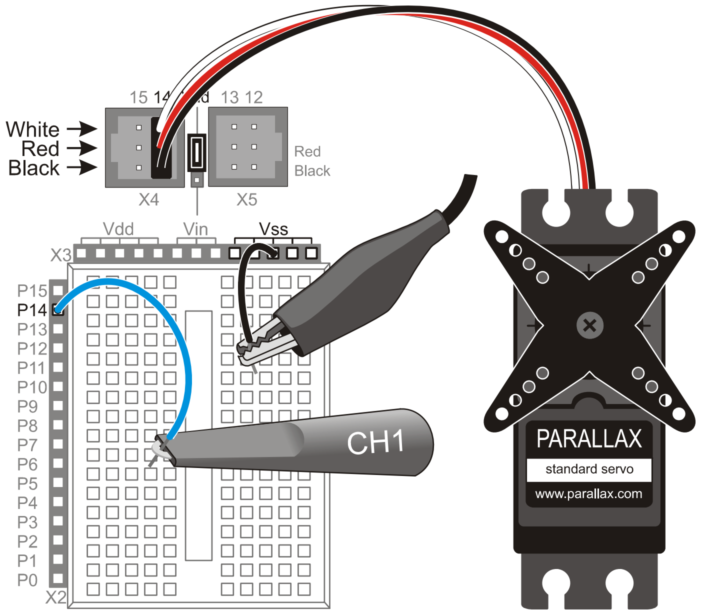


Figure 4-5
Wiring Diagram for Figure 4-3 using the Board of Education

BASIC Stamp HomeWork Board (Serial Rev C and newer, USB all revisions)

- ✓ Disconnect power from your board.
- ✓ Build the servo port as shown in Figure 4-6.

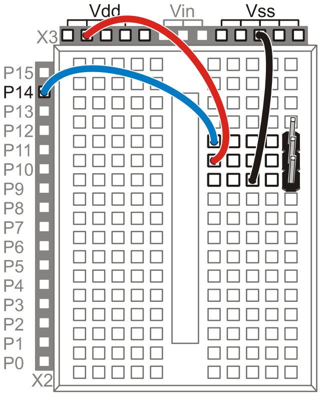


Figure 4-6
BASIC Stamp HomeWork Board Servo Port Wiring

- ✓ Connect the servo to the port as shown in Figure 4-7, making sure the color-coded servo cable wires line up with the labels in the figure.
- ✓ Plug the PropScope CH1 probe wire into the same row as the servo's signal (white) line, and the ground clip into the same row as the servo's ground (black) line.

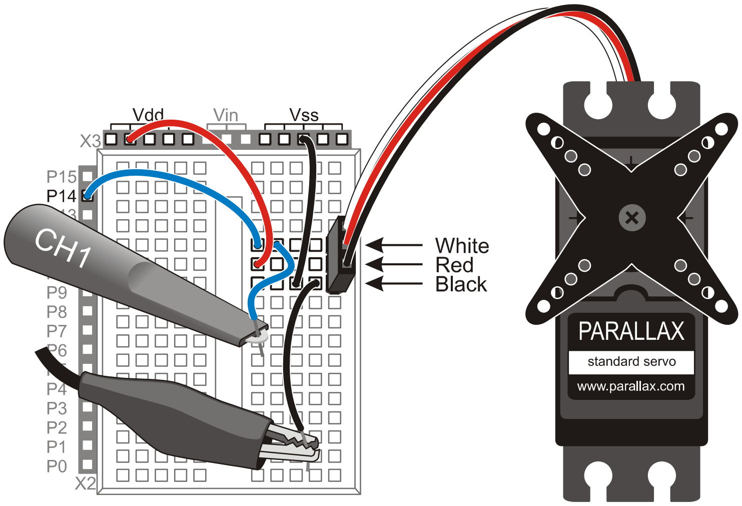


Figure 4-7
Wiring Diagram for Figure 4-3 using the BASIC Stamp HomeWork Board

PWM Signals for Servo Control

The Parallax Standard Servo's horn has a 180° range of motion. Figure 4-8 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.

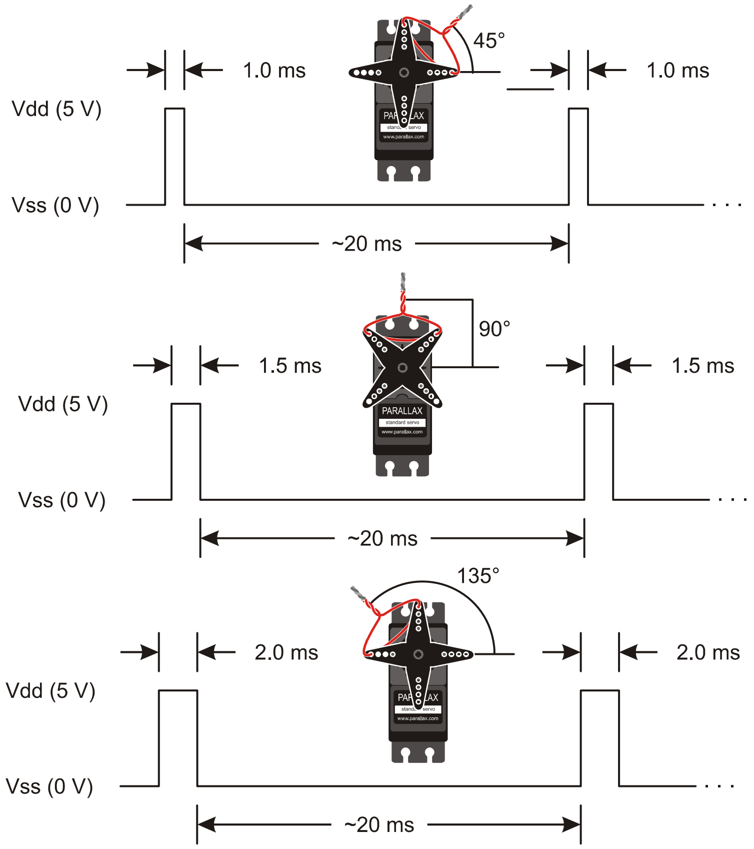


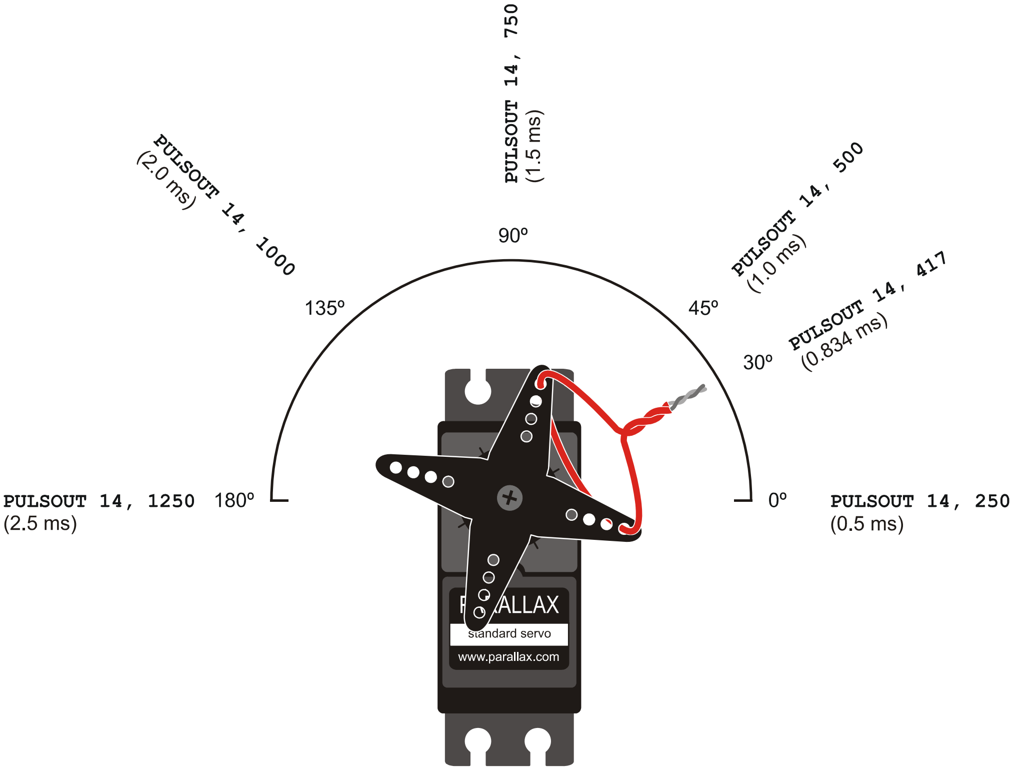
Figure 4-8
Pulse Width vs. Servo Horn Angle

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

Figure 4-8 shows just three servo horn positions in a range of motion that spans from 0° to 180°. Figure 4-9 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 4-9: Pulse Width vs. Servo Horn Angle *(Excerpt from What's a Microcontroller v3.0)*



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 4-9. 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 4-9. *What's a Microcontroller?* Chapter 4, Activity #2 explains the procedure in detail.

The angle vs. pulse time values in Figure 4-9 are approximate. More precise positioning may require some experimentation.

Keep the `PULSOUT` *Duration* arguments in the 350 to 1150 range. **`PULSOUT`** command ***Duration*** values 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.

Center Position Test Program

ServoCenter.bs2 positions the horn at about the 90-degree mark in Figure 4-9 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 $750 \times 2 \times 10^{-6}$ 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 horn holding different positions.

Positive vs. Negative Pulses: Servo pulses are considered *positive pulses* because they spend a certain amount of time as high signals before returning to the low resting state. 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. You will have a chance to measure negative pulses from a sensor in Activity #3 and Activity #4.

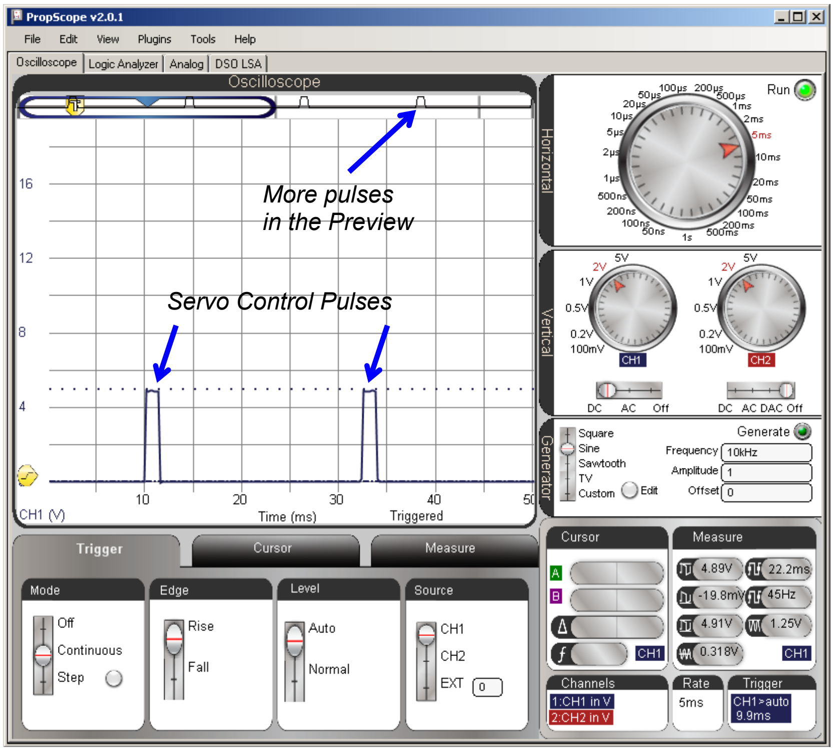
Examine the Servo Control Signals with the PropScope

Back to the "two cycles" guideline 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 4-10.

- ✓ With your BASIC Stamp running ServoCenter.bs2, adjust the Horizontal and Vertical dials, coupling switches and Trigger tab settings so that they match Figure 4-10.

- ✓ Use your Trigger Time control to position the vertical trigger crosshair at the 2nd time division line.
- ✓ Drag the trace down nice and close to the time scale for a better "at a glance" view of the pulse timing.
- ✓ Check the Oscilloscope's Plot Preview, there should be two more pulses off screen, and they continue to repeat to keep the servo holding its horn's position.

Figure 4-10: Servo Control Pulses



More pulses in the Preview

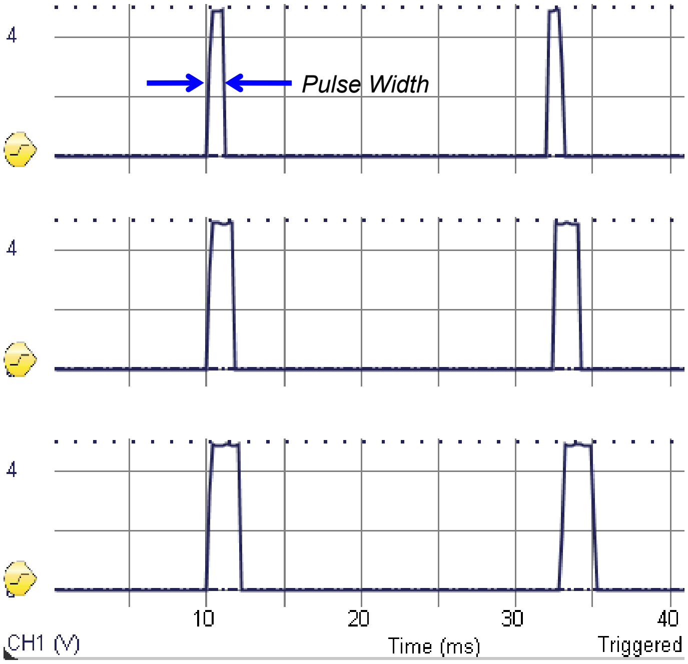
Servo Control Pulses

PropScope v2.0.1
File Edit View Plugins Tools Help
Oscilloscope | Logic Analyzer | Analog | DSO LSA
Oscilloscope
16
12
8
4
10 20 30 40 50
CH1 (V)
Time (ms)
Triggered
Horizontal
Run
50µs 100µs 200µs 500µs 1ms 2ms 5ms 10ms 20ms 50ms 100ms 200ms 500ms 1s 50ns 100ns 200ns 500ns 1µs 2µs 5µs 10µs 20µs
Vertical
5V 2V 1V 0.5V 0.2V 100mV CH1
5V 2V 1V 0.5V 0.2V 100mV CH2
DC AC Off
DC AC DAC Off
Generator
Square Sine Sawtooth TV Custom Edit
Generate
Frequency 10kHz
Amplitude 1
Offset 0
Trigger Cursor Measure
Mode: Off Continuous Step
Edge: Rise Fall
Level: Auto Normal
Source: CH1 CH2 EXT 0
Cursor A B Δ ƒ CH1
Measure 4.89V 22.2ms -19.8mV 45Hz 4.91V 1.25V 0.318V CH1
Channels 1:CH1 in V 2:CH2 in V
Rate 5ms
Trigger CH1>auto 9.9ms

CLIPPED! **What does this mean?** It's telling you that the signal voltage exceeded the voltage limits for the vertical setting. In this case it is not a big deal. All it indicates that the high signal might have exceeded 5 V maximum voltage by a few millivolts for a short period of time. (The author prefers to disregard this message and use the 2 V/div setting for a better visual representation of the pulses and ease of cursor placement.

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 4-11 shows traces for **PULSOUT 14, 500**, **PULSOUT 14, 750**, and **PULSOUT 14, 1000**. With the larger **PULSOUT** ***Duration*** values, 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*** values. 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.

Figure 4-11: Different Servo Control Pulses, Same 20 ms PAUSE



PULSOUT 14, 500
(Servo horn to 45°)

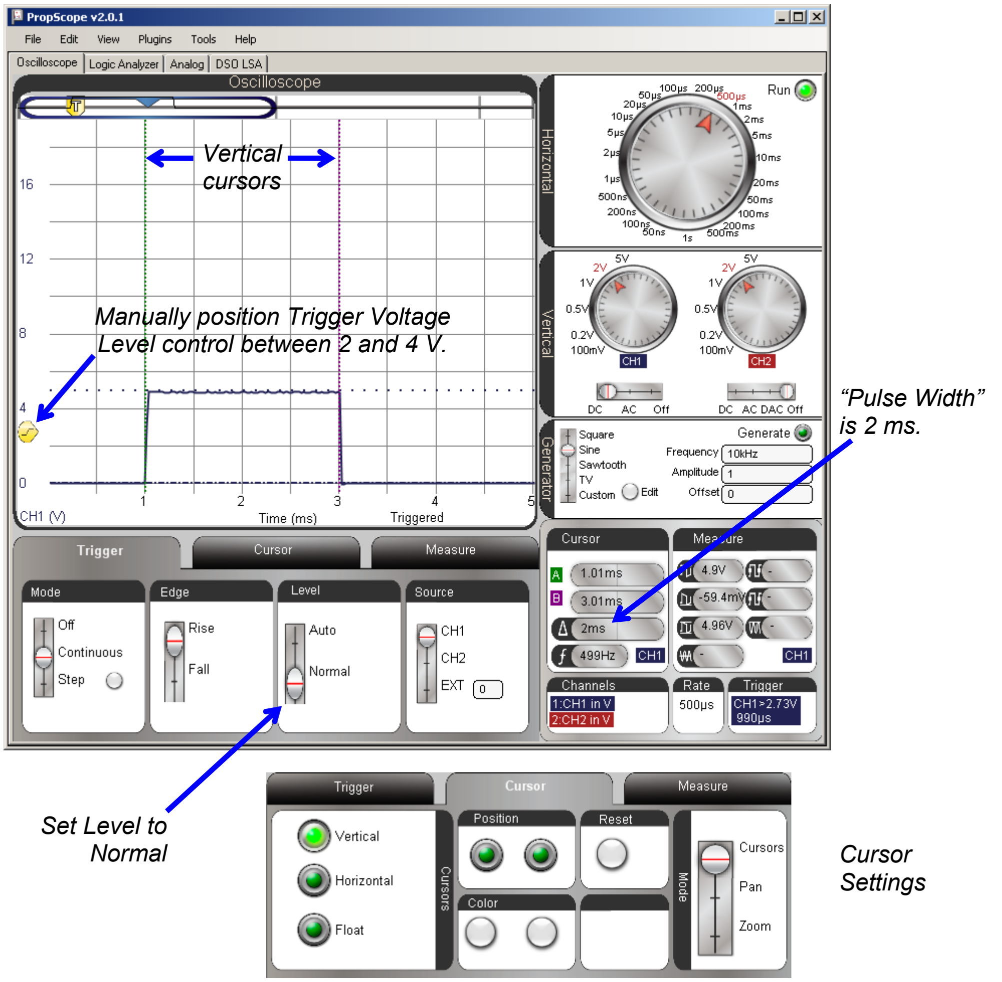
PULSOUT 14, 750
(Servo horn to 90°)

PULSOUT 14, 1000
(Servo horn to 135°)

✓ Modify ServoCenter.bs2 so that it reads **PULSOUT 14, 500**.
✓ Run the modified program, and note the width of the pulses.
✓ Repeat for **PULSOUT** ***Duration*** values of 750 and 1000, and verify that the pulse width increases each time.

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 4-12 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 was made with **`PULSOUT 14, 1000`**.

Figure 4-12: Cursors for Pulse Width Measurement



Vertical cursors

Manually position Trigger Voltage Level control between 2 and 4 V.

"Pulse Width" is 2 ms.

Set Level to Normal

Cursor Settings

The PropScope may have difficulty finding the trigger event when there's just one pulse. Usually it takes a couple of cycles' worth of data to calculate the average voltage used for automatically setting the trigger voltage level. So, we are going to manually set the trigger voltage for this measurement by setting the Trigger tab's Level switch to Normal and then adjusting the Trigger Voltage Level control.

- ✓ In the Trigger tab, set the Level switch to Normal.
- ✓ Drag the Trigger Voltage Level control up to between the 3 and 4 V division lines as shown in Figure 4-12.
- ✓ 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 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.

Your Turn: Modulated Pulse Width

Here is a program from *What's a Microcontroller?* 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 Oscilloscope 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
```

ACTIVITY #2: DUTY CYCLE IN PWM DAC

A more descriptive name for PBASIC's **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 256^{ths} 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.

DAC Test Parts

(1) Resistor – 220 Ω (red-red-brown)
(1) Resistor – 1 kΩ (brown-black-red)
(1) Capacitor – 1 µF
(misc.) Jumper wires

DAC Test Circuit

The circuit in Figure 4-13 is one of the two DAC circuits that were used to set DC voltages in Chapter 2, Activity #4. This circuit was also used to make the PropScope plot a simulated heart monitor display in Chapter 3, Activity #4. A resistor has been added between P14 and the CH2 probe to prevent possible I/O pin damage that could happen if the probe is inadvertently left connected to the DAC Card's function generator output.

- ✓ Before you start building this circuit, make sure the probe BNC connectors are connected to PropScope CH1 and CH2. There SHOULD NOT be any probes connected to the DAC Card.
- ✓ Build the circuit shown in Figure 4-13 and Figure 4-14.

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 #4. 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, 5                           ' 1.25 V to P14 capacitor

LOOP                                        ' Repeat main loop
```

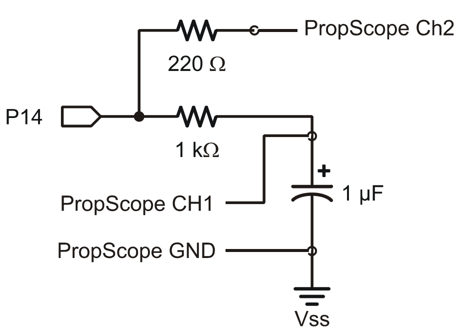


Figure 4-13
Probes for P14 PWM Signal and DAC Output

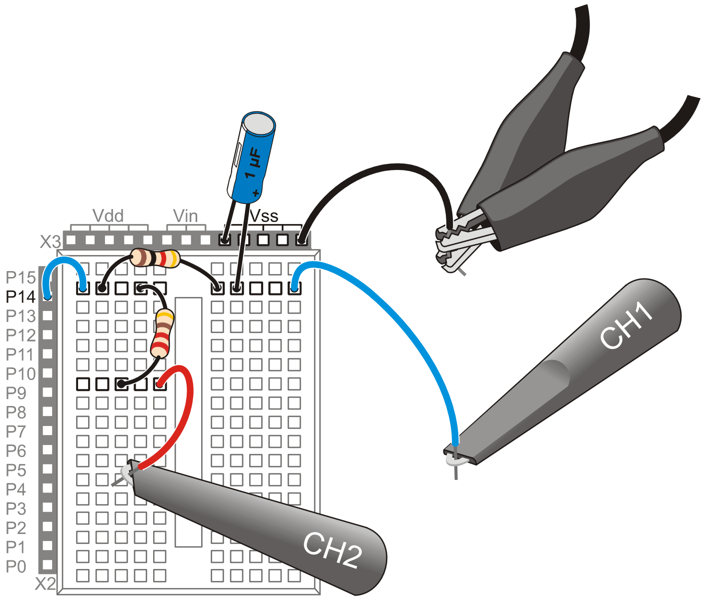


Figure 4-14
Example Wiring Diagram for Figure 4-13

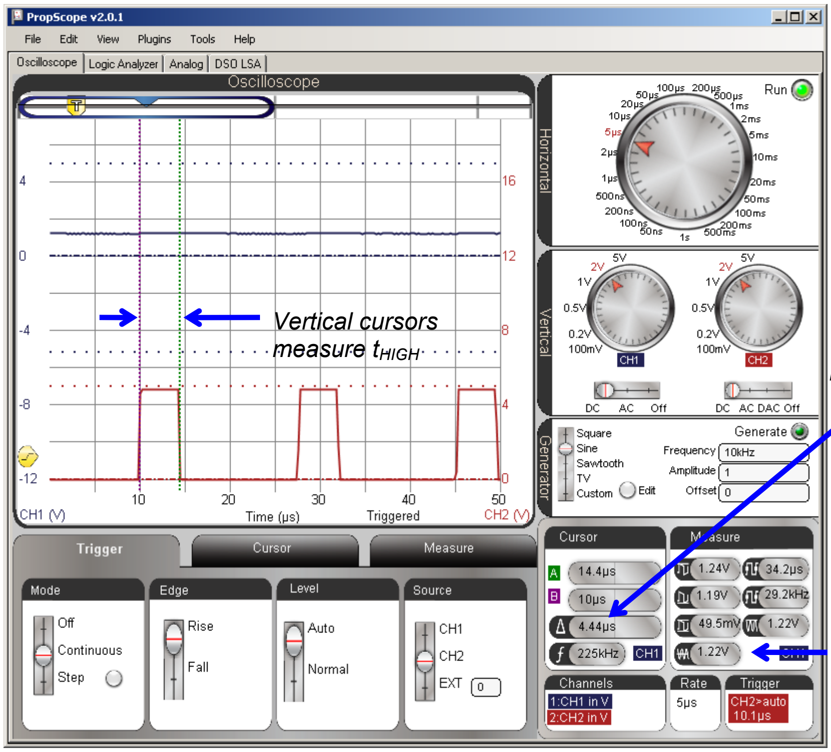
PWM Command Duty Cycle Measurements

In Figure 4-15, 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 to 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 4-15 shows the vertical time cursors placed for the t_{HIGH} measurement.

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


PropScope v2.0.1
File Edit View Plugins Tools Help
Oscilloscope | Logic Analyzer | Analog | DSO LSA
Oscilloscope
4 0 -4 -8 -12
16 12 8 4 0
10 20 30 40 50
CH1 (V) Time (µs) Triggered CH2 (V)
Horizontal
Run
50ns 100ns 200ns 500ns 1µs 2µs 5µs 10µs 20µs 50µs 100µs 200µs 500µs 1ms 2ms 5ms 10ms 20ms 50ms 100ms 200ms 500ms 1s
Vertical
5V 2V 1V 0.5V 0.2V 100mV CH1
5V 2V 1V 0.5V 0.2V 100mV CH2
DC AC Off
DC AC DAC Off
Generator
Square Sine Sawtooth TV Custom Edit
Generate
Frequency 10kHz
Amplitude 1
Offset 0
Trigger Cursor Measure
Mode: Off Continuous Step
Edge: Rise Fall
Level: Auto Normal
Source: CH1 CH2 EXT 0
Cursor
A 14.4µs
B 10µs
Δ 4.44µs
ƒ 225kHz CH1
Measure
1.24V 34.2µs
1.19V 29.2kHz
49.5mV 1.22V
1.22V CH1
Channels 1:CH1 in V 2:CH2 in V
Rate 5µs
Trigger CH2>auto 10.1µs

Vertical cursors measure t_{HIGH}

Cursor time measurement: t_{HIGH} = 4.44 µs

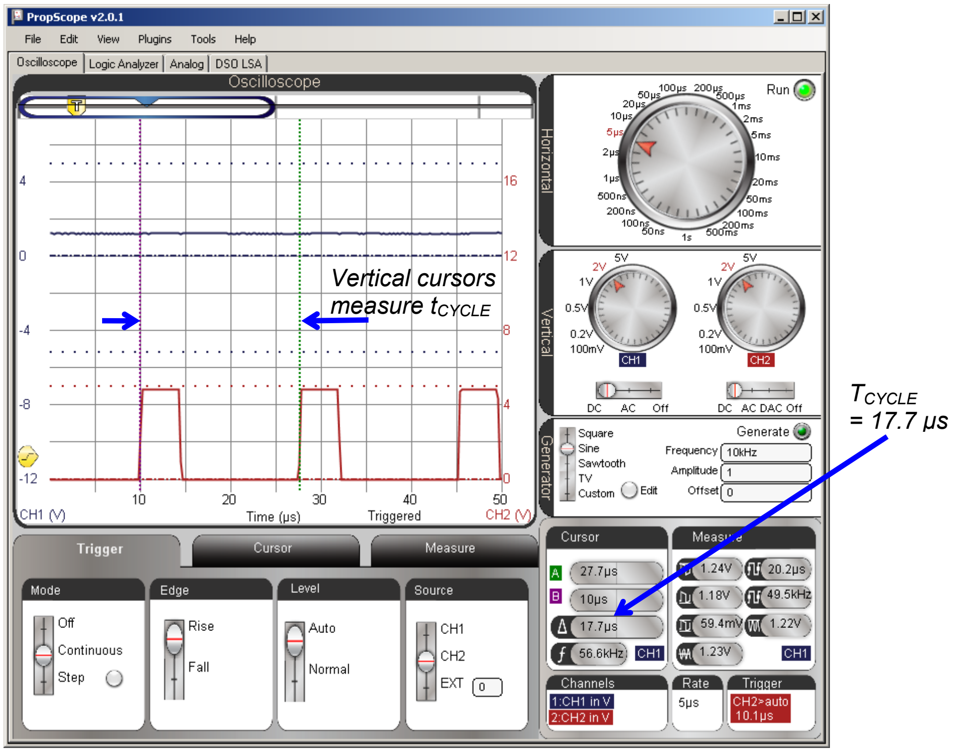
CH1 DC voltage measurement: 1.2 V

- ✓ Configure your PropScope's Horizontal dial, Vertical dials and coupling switches, and Trigger tab settings as shown in Figure 4-15. Make sure to set the Trigger Source to CH2.
- ✓ 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.

Cycle time (t_{CYCLE}) is the signal's period, and Figure 4-16 shows the time cursors aligned with successive rising signal edges for this measurement.

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

Figure 4-16: t_{CYCLE} Measurement with Cursors



Vertical cursors measure t_{CYCLE}

T_{CYCLE} = 17.7 µs

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

$$\begin{aligned} Duty\,Cycle &= \frac{t_{HIGH}}{t_{CYCLE}} \times 100\% \\ &= \frac{4.44\,\mu s}{17.7\,\mu s} \times 100\% \\ &\approx 25.1\% \end{aligned}$$

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

$$\begin{aligned} V_{DAC} &= Duty\,Cycle \times 5V \\ &\approx 25.1\% \times 5V \\ &= 1.255V \end{aligned}$$

The Channel 1 Average voltage measurement in Figure 4-15 and Figure 4-16 is oscillating between 1.22 V and 1.23 V. Since this BASIC Stamp feature is not typically used for setting precision voltages, a couple hundredths of a volt error is not bad at all. Now, if error turned out to be in terms of tenths of a volt, that might need a closer look.

Your Turn: Try a New Value

The command **`PWM 14, 64, 5`** 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 4-17 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.

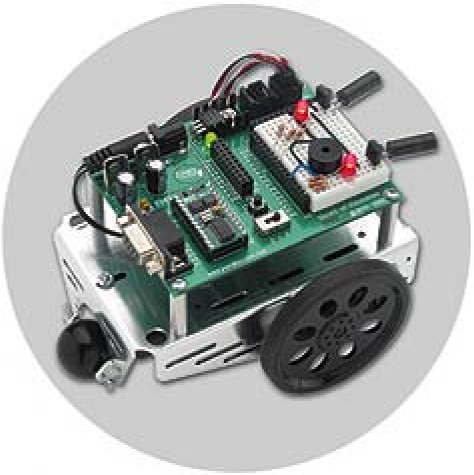


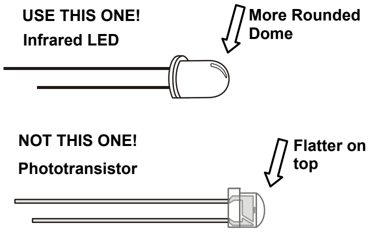
Figure 4-17
Boe-Bot® Robot

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

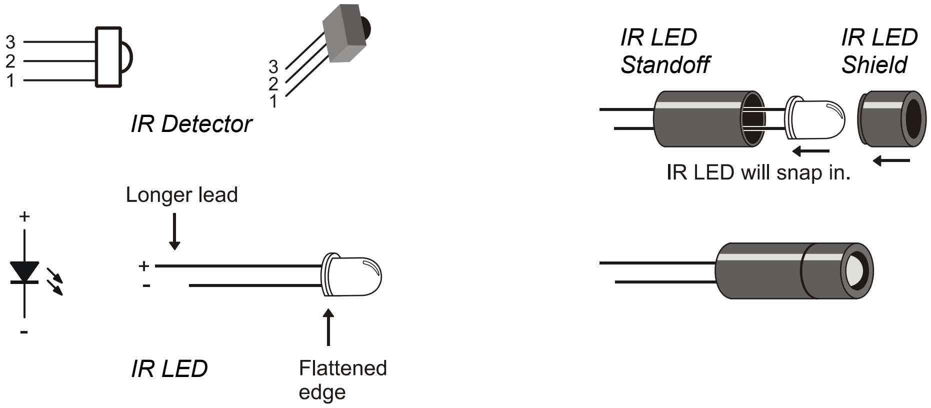
For reference, and to make it easier finding the parts in the kit,
Figure 4-18 shows drawings of the IR LED, standoff, shield, and detector. It also shows how to house the IR LED into the standoff and shield.

(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



✓ Snap the IR LED into the standoff and then place the IR shield on top using
✓ Figure 4-18 as a guide.

Figure 4-18: IR Detector, LED, Standoff and Shield



IR Object Detection Circuit

Figure 4-19 shows a schematic of the infrared detection circuit, and Figure 4-20 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. Unmodulated infrared from other sources such as a nearby window or indoor lighting normally have little to no effect.

- ✓ Build the circuit shown in Figure 4-19 and Figure 4-20.
- ✓ Check your work against Figure 4-20 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 4-19: Infrared Object Detection Schematic

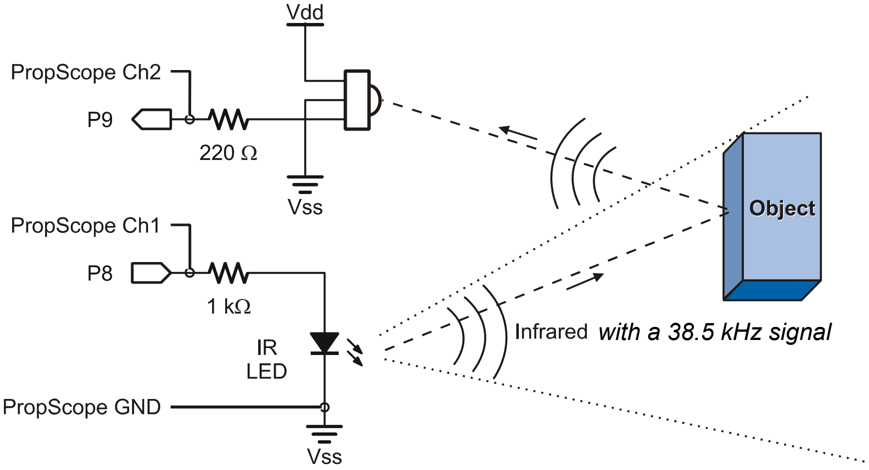
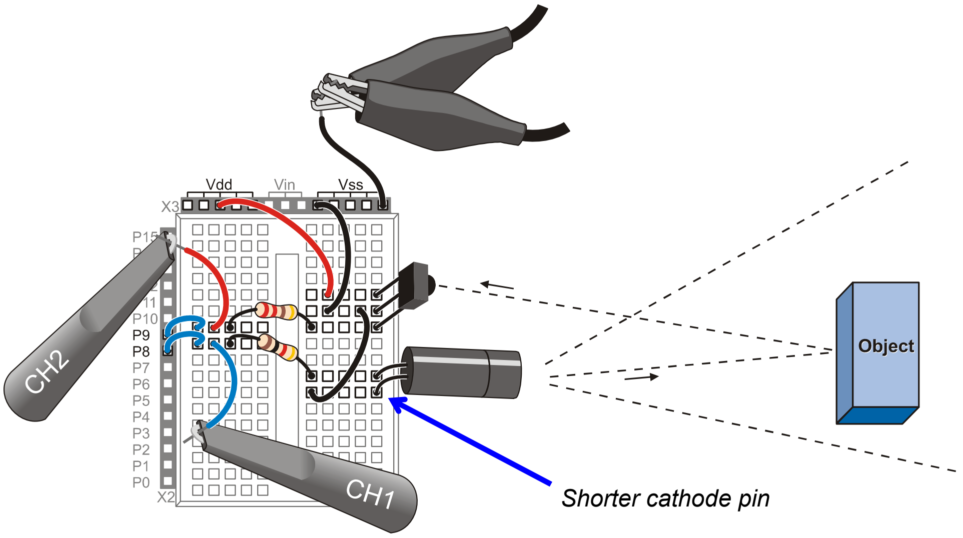


Figure 4-20: Wiring Diagram Example of Figure 4-19



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, making 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.

In Chapter 7: Basic Sine Wave Measurements, you will use the PropScope to examine sine wave signals the BASIC Stamp synthesizes with binary signals using the **`FREQOUT`** command.

✓ Enter and run TestLefIrPair.bs2.

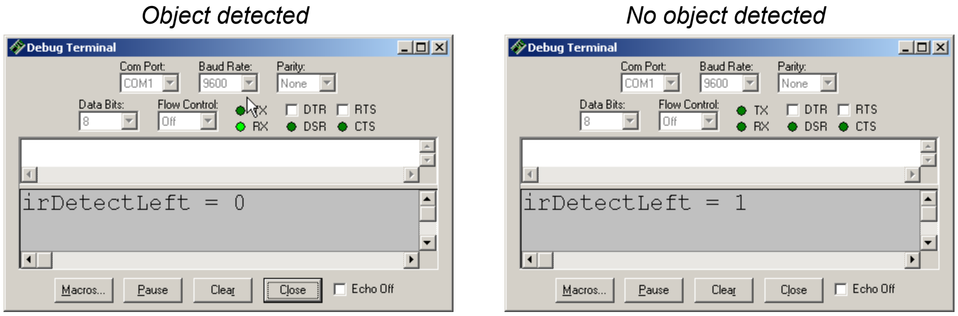
```
' Robotics with the Boe-Bot - TestLeftIrPair.bs2
' Test IR object detection circuits, IR LED connected to P8 and detector
' connected to P9.

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

irDetectLeft   VAR     Bit

DO

  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

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

LOOP
```

IR Object Detection Tests

Figure 4-21 shows the Debug Terminal's reports for object detected (0) and object not detected (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".
- ✓ With no objects in the IR LED's line of sight, verify that the Debug Terminal reports "irDetectLeft = 1."

Figure 4-21: Debug Terminal IR Detection Tests



Object detected

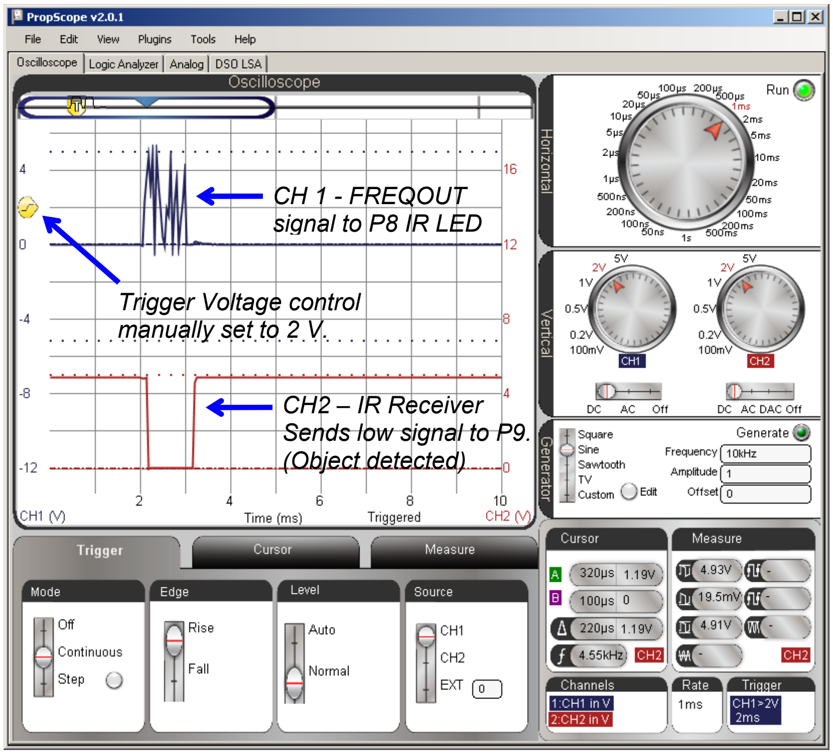
```
irDetectLeft = 0
```

No object detected

```
irDetectLeft = 1
```

Figure 4-22 shows the Oscilloscope view while an object is detected. The 1 ms **FREQOUT** signal is shown by the upper, blue CH1 trace. The lower, red CH2 trace sends a low signal during that time, which indicates that it detected reflected infrared with a 38.5 kHz component. The BASIC Stamp I/O pin interprets the 0 V applied to its P9 I/O pin as a binary 0, which results in the irDetectLeft = 0 message on the left side of Figure 4-21.

Figure 4-22: Object Detected



CH 1 - FREQOUT signal to P8 IR LED

Trigger Voltage control manually set to 2 V.

CH2 – IR Receiver Sends low signal to P9. (Object detected)

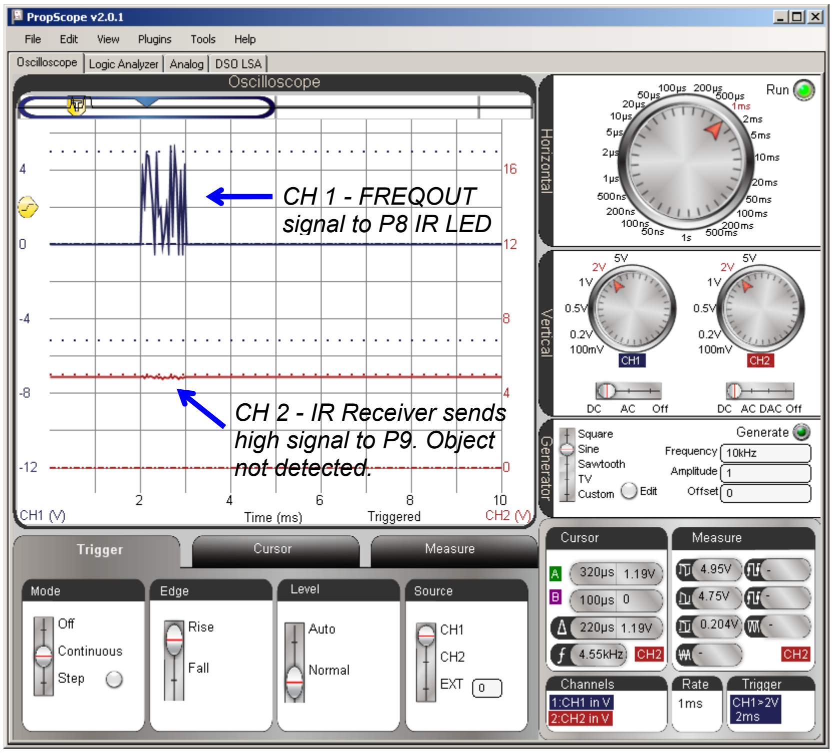
- ✓ Configure your PropScope's Horizontal dial, Vertical dials and coupling switches, and Trigger tab settings as shown in Figure 4-22.
- ✓ Manually adjust the Trigger Voltage Level control to about 2 V.
- ✓ Place an object that will (like your hand or a book) facing the infrared detector. Use Figure 4-20 on page 128 as a guide for object placement.
- ✓ Verify that it results in the irDetectLeft = 0 display in the Debug Terminal.

- ✓ Verify that it results in a brief 0 V signal on CH2 while the **FREQOUT** command is active in CH1.

Figure 4-23 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 lower 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 irDetectLeft = 1 message on the right side of 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 4-21, and verify that it results in an uninterrupted 5 V signal on CH2.

Figure 4-23: Object not Detected



CH 1 - FREQOUT signal to P8 IR LED

CH 2 - IR Receiver sends high signal to P9. Object not detected.

PropScope v2.0.1

Frequency 10kHz
Amplitude 1
Offset 0

Cursor:
A 320µs 1.19V
B 100µs 0
Δ 220µs 1.19V
ƒ 4.55kHz CH2

Measure:
4.95V
4.75V
0.204V

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 1ms
Trigger: CH1>2V 2ms

The 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 receiver sends.

- ✓ Set it up so that the IR detector has an object to detect and your PropScope display resembles Figure 4-22.
- ✓ 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.
- ✓ Measure the duration.
- ✓ Compare it to the duration of the **`FREQOUT`** signal on CH1.

Your Turn: Change the Pulse Duration

What does infrared object detection with infrared have to do with pulse width modulation? The answer is that you can control the duration of the low pulse the IR detector sends by adjusting the **`FREQOUT`** command's ***Duration***. In other words, in **`FREQOUT`** ***Pin***, ***Duration***, ***Freq1***, you can modulate the pulse width by changing the **`FREQOUT`** command's ***Duration*** argument. Try this:

- ✓ Change the **`FREQOUT`** command in TestLeftIrPair.bs2 to **`FREQOUT 8, 2, 38500`**.
- ✓ Place an object that will cause a detection in front of the IR LED and detector.
- ✓ Repeat the pulse width measurements, on the PropScope.
- ✓ See how changing the **`FREQOUT`** command's ***Duration*** argument results in a longer **`FREQOUT`** signal in CH1, which in turn causes the IR detector to send a 2 ms low pulse?

ACTIVITY #4: ADVANCED TOPIC: PULSES FOR TV REMOTE COMMUNICATION

Figure 4-24 shows how a TV remote sends bursts of signals in the 38 to 40 kHz neighborhood to an IR detector. The durations of a series of these bursts are coded to communicate which button on the remote is pressed. The IR detector on your board converts these brief IR signal durations to brief low pulses, which are also called negative pulses. The BASIC Stamp can decode these pulses to figure out which button on the remote is pressed in a manner similar to the microcontrollers inside TVs and other entertainment system components. In this activity, you will use the both the PropScope and BASIC Stamp to examine these signals.

Figure 4-24: IR Remote Signals to IR Detector

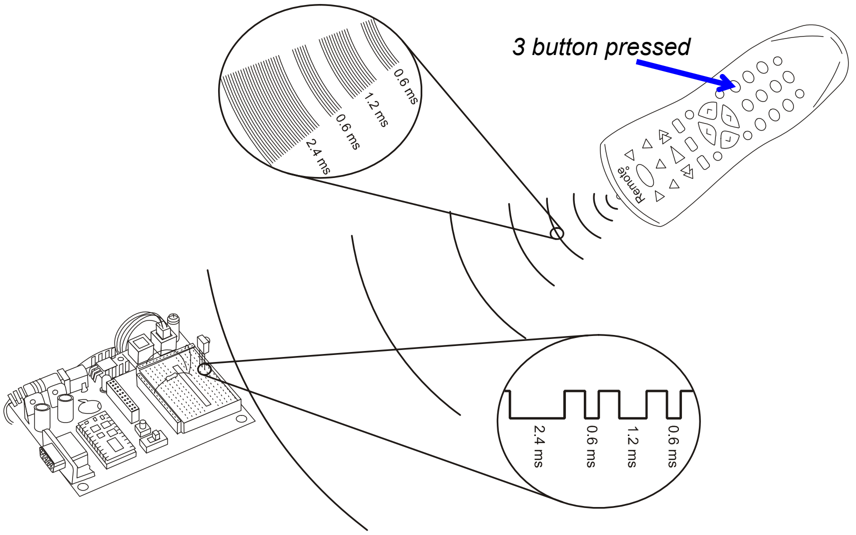
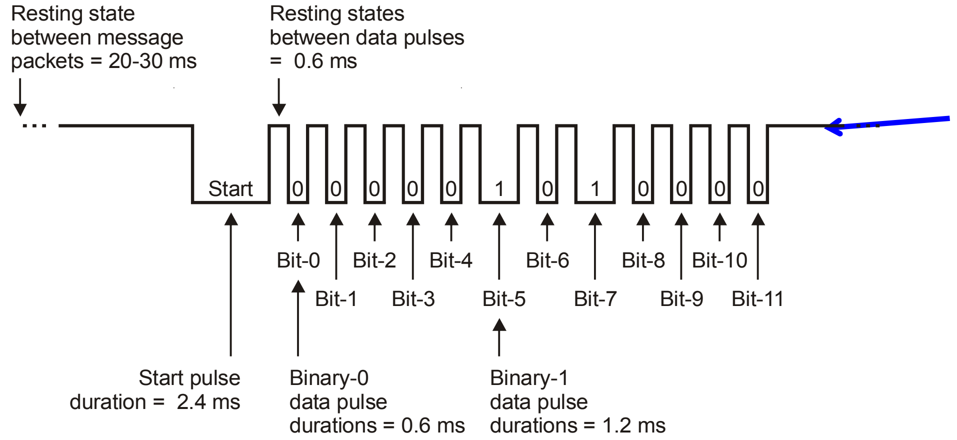


Figure 4-25 shows a timing diagram of a SONY TV remote protocol from the standpoint of the IR detector's output. After a 2.4 ms start pulse, a series of 12 low pulses follow. Each low pulse lasts for either 0.6 ms to indicate a binary-0, or 1.2 ms to indicate a binary-1. Just as an 8-bit binary number can contain a value in the 0 to 255 range, a 12-bit binary number can contain a value in the 0 to 4095 range. So, the pulse width modulated signal in Figure 4-25 can be used to transmit 4096 different values.

Figure 4-25: SONY TV Remote Protocol



Extra Parts, Equipment, and Setup

You will use the IR detector and CH2 probe shown in Figure 4-19 and Figure 4-20, on pages 127 and 128, so there is no need to modify your circuit or probe connections. The “extra equipment” you will need is a universal remote.

(1) Universal remote (Parallax #020-00001, for example)
(1) Instruction sheet/booklet for the universal remote
Compatible batteries

Although a universal TV remote is not included in the Understanding Signals kit, most households have one that can be configured to control a SONY TV. For those that don’t, inexpensive universal remotes can also be purchased from local department stores. Check the package before you purchase a particular remote to make sure it can be configured to control a SONY TV. If the packaging says something like "compatible with most/all major brands..." it should work with SONY TVs. These remotes are usually packaged with either a piece of paper or a booklet that explains how to configure it to be a SONY TV remote. The instructions typically involve:

- Looking up a number code for SONY TV in the remote’s sheet/booklet
- Pressing and holding a certain remote button until an indicator light comes on
- Entering the number code into the remote’s keypad

An IR Remote Parts Kit is available at www.parallax.com. Just type 29122 into Search field and click Go. The remote in this kit is an example of one that costs under $15 US.

The IR Remote for the Boe-Bot Parts + Text page (Item code 28139) has a link to a free .pdf tutorial and another to a zip file with lots of PBASIC code examples for BASIC Stamp IR Remote communication and control. These links are in the Downloads & Resources section near the bottom of the web page.

Here are example instructions for the TV remote in the IR Remote Parts Kit:

- ✓ Remove the battery compartment cover and determine how many and what kind of batteries to use (AA, AAA, etc).
- ✓ Load the battery compartment with new (or freshly charged rechargeable) batteries. Do not mix battery types.
- ✓ Find the TV setup codes section in the instruction sheet/booklet, such as "Setup Codes for TV", "Setup Codes for Television", "TV Code List".
- ✓ Find the code for SONY from the TV code list, and make a note of it.
- ✓ Find the section that explains how to manually program a TV code into your remote, such as "Programming Your Remote", "To Manually Program Your Remote Control", "Programming for TV".
- ✓ Follow the instructions in the manual programming section for entering the SONY TV code into your remote.

IR Remote Test Code

RecordAndDisplayPwm.bs2 measures the twelve pulses that follow the start pulse in Figure 4-25 and displays their durations in the Debug Terminal. The program uses a command called **`PULSIN`**, which is the inverse of **`PULSOUT`**. Instead of transmitting a pulse to another device, it measures a pulse transmitted by another device and stores the result in a variable. This result is a pulse measurement in terms of 2 µs units. The Debug Terminal displays these results, and you can multiply each one by 2 to convert it from a 2 µs unit measurement to a number of microseconds.

- ✓ Enter RecordAndDisplayPwm.bs2 and run it.

Remove Sources of IR Interference. Depending on the IR detector, sunlight from a nearby window could cause a lot of spurious signals. Certain fluorescent lights generate 38.5 kHz range inference too. For best results, close the blinds and turn off nearby fluorescent lights.

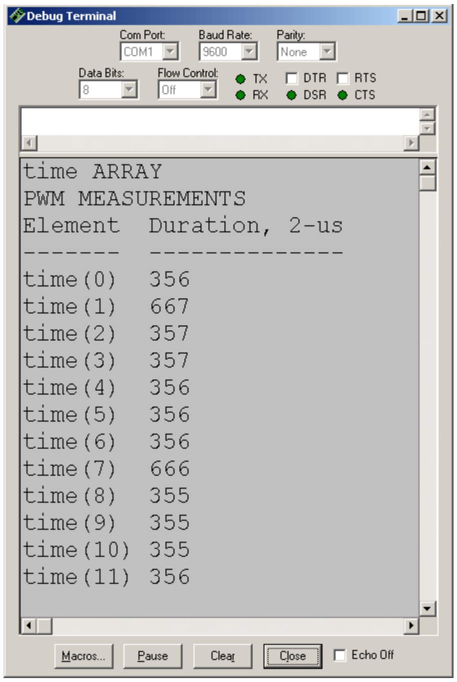
```
' IR Remote for the Boe-Bot - RecordAndDisplayPwm.bs2
' Measure all data pulses from a SONY IR remote set to control a TV.

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

time           VAR     Word(12)              ' SONY TV remote variables.
index          VAR     Nib
                                             ' Display heading.
DEBUG "time ARRAY", CR,
      "PWM MEASUREMENTS", CR,
      "Element  Duration, 2-us", CR,
     "-------  --------------"

DO                                           ' Beginning of main loop.

  DO                                         ' Wait for rest between messages.
    PULSIN 9, 1, time(0)
  LOOP UNTIL time(0) > 1000

  PULSIN 9, 0, time(0)                       ' Measure/store data pulses.
  PULSIN 9, 0, time(1)
  PULSIN 9, 0, time(2)
  PULSIN 9, 0, time(3)
  PULSIN 9, 0, time(4)
  PULSIN 9, 0, time(5)
  PULSIN 9, 0, time(6)
  PULSIN 9, 0, time(7)
  PULSIN 9, 0, time(8)
  PULSIN 9, 0, time(9)
  PULSIN 9, 0, time(10)
  PULSIN 9, 0, time(11)

  FOR index = 0 TO 11                        ' Display 12 pulse measurements.

    DEBUG CRSRXY, 0, 4 + index, "time(", DEC index, ")",
          CRSRXY, 9, 4 + index, DEC time(index), CLREOL

  NEXT

LOOP                                         ' Repeat main loop.
```

For more information on how the example programs in this activity work, download and consult the *IR Remote for the Boe-Bot pdf* tutorial from www.parallax.com/education.

IR Remote Test Measurements with the BASIC Stamp

Figure 4-26 shows the BASIC Stamp module's pulse measurements while the remote's 3 button is pressed.

✓ Point your remote at the IR detector and press and release the 3 button.



```
time ARRAY
PWM MEASUREMENTS
Element  Duration, 2-us
-------  --------------
time(0)  356
time(1)  667
time(2)  357
time(3)  357
time(4)  356
time(5)  356
time(6)  356
time(7)  666
time(8)  355
time(9)  355
time(10) 355
time(11) 356
```

Figure 4-26
BASIC Stamp Displays Pulse Measurements from an IR Detector

…that's receiving infrared signals from a SONY TV remote with the 3 button pressed.

Table 4-1 shows approximate measurements of the first four pulses as buttons 1 through 9 are pressed.

Table 4-1: Pulse Patterns for SONY® TV Remote Buttons

Pulse Variable	Button								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
time(0)	360	670	360	670	360	670	360	670	360
time(1)	360	360	670	670	360	360	670	670	360
time(2)	360	360	360	360	670	670	670	670	360
time(3)	360	360	360	360	360	360	360	360	670

For this remote, the 670 values are binary-1s and the 360 values are binary-0s. Time(3) through time(0) store pulse durations that represent the lowest four binary digits in the 12-digit binary number the IR remote transmits.

Table 4-2 shows a sequence of the first five remote buttons, their pulse duration patterns, the binary values they represent, and the decimal equivalent values. We'll call these values 'remote codes.'

- ✓ Test your remote buttons 6, 7, 8, 9, and 0, and fill in the rest of the table.
- ✓ Your values may differ, but there should be a pattern of values that are closer to 600 that are binary-1s and values that are closer to 300 that are binary-0s.

Table 4-2: Buttons, Pulse Patterns, Binary & Decimal Values

Button	time(3)	time(2)	time(1)	time(0)	Binary Value	Decimal value
1	360	360	360	360	0000	0
2	360	360	360	670	0001	1
3	360	360	670	360	0010	2
4	360	360	670	670	0011	3
5	360	670	360	360	0100	4
6						
7						
8						
9						
0						

Decimal Digits vs. Binary Digits

Four decimal digits, each of which could be a value from 0 to 9, represent the number of:

Thousands Hundreds Tens Ones

Four binary digits, each of which could be a value from 0 to 1, represent the number of:

Eights Fours Twos Ones

Knowing this, you can convert values from binary to decimal. For example you could convert binary-1010 do decimal knowing that there an 8 and a 2, which adds up to 10.

$(1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) = 10$

Variations in SONY TV Remote Communication Pulses

Servo control pulses have to be fairly precise because even small variations in pulse durations will be visible as different servo horn positions. In contrast, TV remote communication protocols tend to leave more room for variations in the pulse durations that represent binary-1s and binary-0s. The microcontroller inside the TV that receives and deciphers pulses from the IR detector can decide whether a given pulse falls into a range of durations that represents a binary-1, or a range of durations that represents a binary-0. Since universal remotes made by different manufacturers may vary somewhat in their IR signaling times (that get converted to pulses by the IR detector), leaving room in the protocol for variations in the pulse duration expectations is an important design feature.

With possible variations in mind, your remote's pulse duration values that represent binary 1s and 0s might differ from the values in Figure 4-26. For that matter, those values differ considerably from the values in the timing diagram in Figure 4-25 on page 134. That timing diagram indicates that binary zeros last 0.6 ms, and binary ones last 1.2 ms. However, in Figure 4-26, the binary-1s appear last for 670 × 2µs = 1340 µs. That's 1.34 ms, not 1.2 ms. Likewise, 360 × 2 µs = 720 µs, which is 0.72 ms, not 0.6 ms. How will you be able to tell the difference between your remote's binary-1 and binary-0 pulses? Look for longer duration, binary-1 pulses that are closer to 1.2 ms, and shorter duration binary-0 pulses that are closer to 0.6 ms.

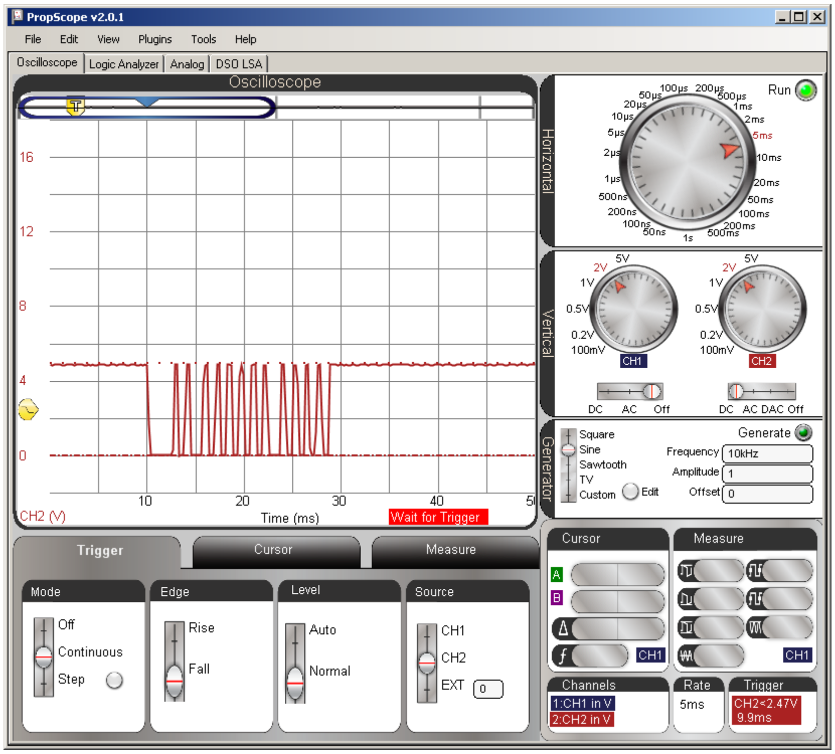
IR Remote Test Measurements with the PropScope

Figure 4-27 shows a first view of the IR detector's output when the 3 button on the remote is pressed. The Horizontal dial selection was chosen using the "two cycle" guideline of a signal in the Oscilloscope screen for a first look. In this case, we'll call a "cycle" one infrared message. Looking at the timing diagram in Figure 4-25 on page 134, a rough approximation of the time an IR message lasts would be:

$$
\begin{aligned}
&2.4\,ms = 1 \times 2.4\,ms\,(start\ pulse)\\
+&7.2\,ms = 12 \times 0.6\,ms\,(highs\,between\ low\ pulses)\\
+&6.0\,ms = 10 \times 0.6\,ms\,(binary - 0\ pulses)\\
+&2.4\,ms = 2 \times 1.2\,ms\,(binary - 1\ pulses)\\
=&18\,ms
\end{aligned}
$$

Assuming a message fits in a 20 ms window, multiply by 2 for enough room for two cycles, which is 40 ms. Then, divide by 10 for Horizontal dial setting in units with a result of 4 ms/div. The closest value is 5 ms/div.

Figure 4-27: SONY IR Message – Initial View



IMPORTANT: The timing diagram also showed that the Start pulse begins with a negative edge, so it will be important to set the Trigger tab's Edge switch to Fall. There are many negative (falling) edges in the signal, but this will increase the likelihood that the first negative edge triggers the display, like it did in Figure 4-27.

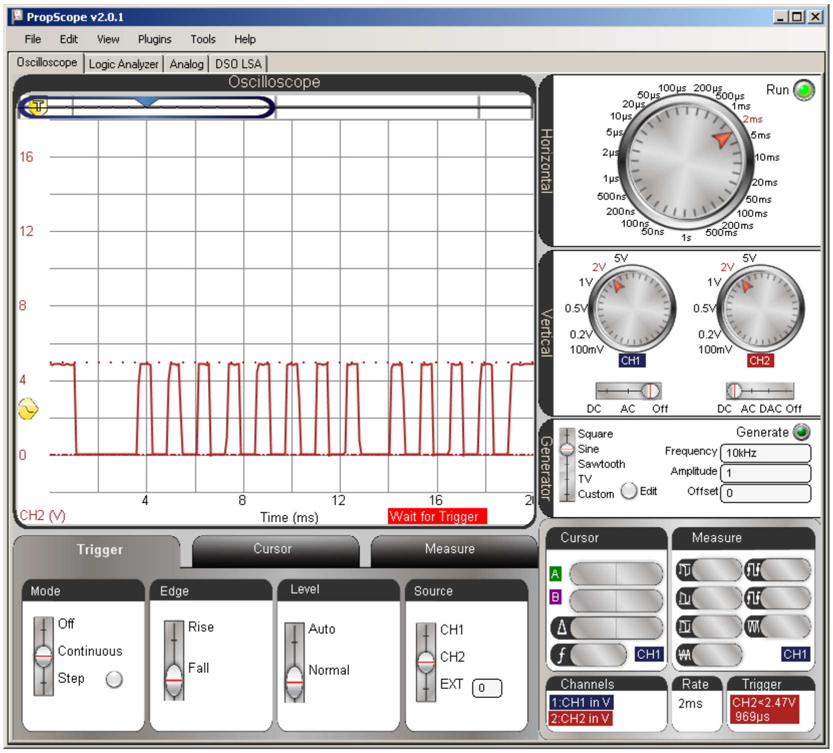
Also, the Trigger tab's Level switch should be set to Normal instead of Auto. Normal means you will manually adjust the trigger voltage level. Auto triggering uses the average voltage of the signal in the Oscilloscope screen, but there may be long periods of time between TV remote massages where the signal will stay at 5 V. So, the average voltage will be 5 V, as will the automatic trigger level. As the IR detector sends a series of negative pulses to the BASIC Stamp I/O pin, there will be lots of 0 V low signals. So, it would be better to manually set the trigger voltage level to 2.5 V instead of letting it hover around 5 V and hoping it catches the signal transitions that we want to trigger on.

- ✓ Configure your PropScope's Horizontal dial, Vertical dials and Coupling switches, and Trigger tab settings as shown in Figure 4-27.
- ✓ Make sure to set the Trigger Edge switch to Fall.
- ✓ Also, make sure to set the Trigger Level switch to Normal.
- ✓ Then, set the Trigger Voltage control along the left side of the Oscilloscope screen to about 2.5 V.
- ✓ Make sure the Trigger Time control is set half way between the Oscilloscope screen's left margin and the first time division line. (Careful here, in previous activities, we always used the second time division line.)
- ✓ Press/release a number button on your remote while pointing it at the IR detector.
- ✓ Verify that a signal appears that is similar to the one in the Figure 4-27.
- ✓ Verify that other number buttons result in similar signals.

Figure 4-28 shows a closer view of the timing with the remote's 1 number button pressed/released. You may have to press/release the button a couple of times to get a display that includes the wider start pulse at the far left.

- ✓ Adjust your PropScope's Horizontal dial to 2 ms/div, and press the 1 button a few times until you get a display similar to Figure 4-28.

Figure 4-28: IR Message (1 button) with Decreased Horizontal Scale

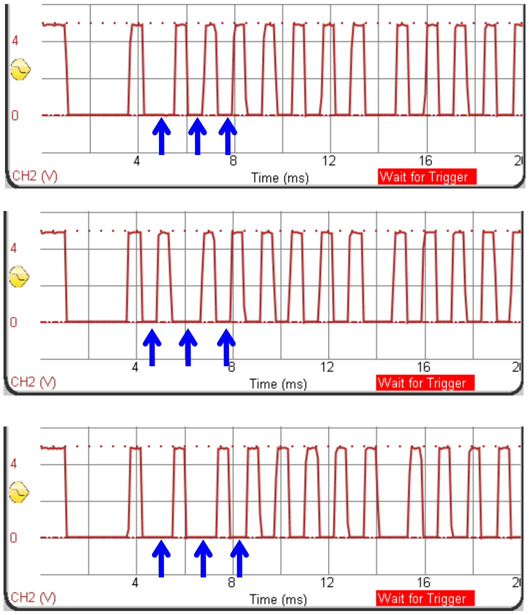


Frequency 10kHz
Amplitude 1
Offset 0
EXT 0
Channels: 1:CH1 in V, 2:CH2 in V
Rate: 2ms
Trigger: CH2<2.47V 969µs

With the Horizontal dial set to 2 ms, you’ll be able to see the low pulse durations change as you press/release different numbers on the remote. Figure 4-29 shows patterns that represent the 2, 3, and 4 buttons.

- ✓ Remember to point the remote at the IR detector before you press its buttons.
- ✓ Press/release the 2 button on your remote a few times, until the start pulse’s negative edge lines up with the 1st time division. The pattern of low pulses should resemble the signal labeled 2 Button in Figure 4-29.
- ✓ Repeat for the 3 and 4 buttons, and compare to Figure 4-29.
- ✓ Keep an eye on the three negative pulses that follow the start pulse. They are the ones that carry the binary values 001, 010, and 011, which are the binary remote codes for the 2, 3, and 4 buttons.

Figure 4-29: Pulse Patterns for the 2, 3, and 4 Buttons



2 Button

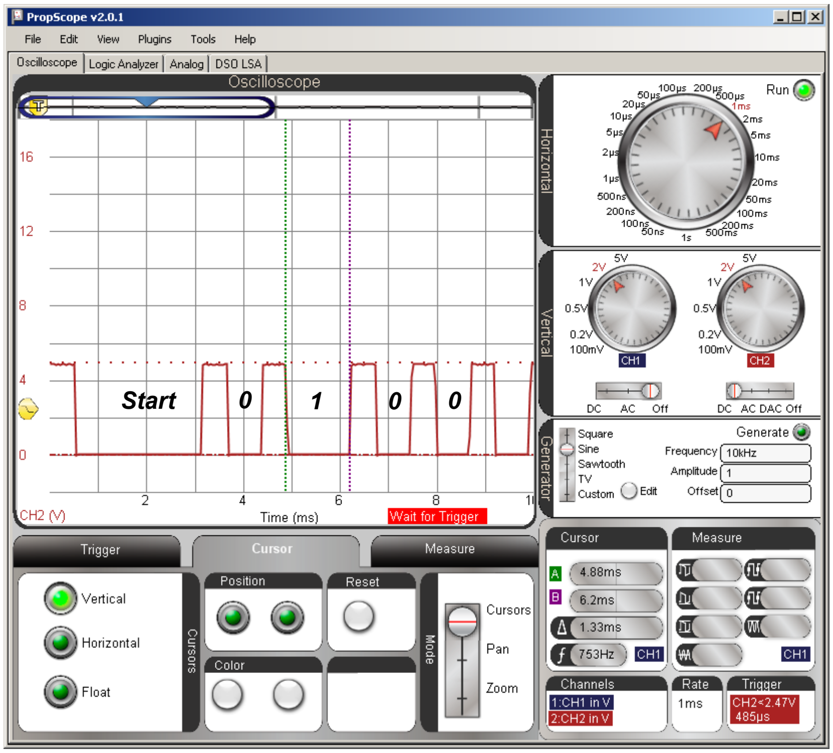
3 Button

4 Button

Figure 4-30 shows the Oscilloscope zoomed in another time division setting, with the Horizontal dial set to 1 ms/div. This is a good time setting for measuring low pulse times. The Vertical time cursors in the figure are measuring a binary-1 pulse.

- ✓ Adjust the Horizontal dial to 1 ms/div.
- ✓ Press the 3 button a few times until the Oscilloscope view resembles Figure 4-30.
- ✓ Use the Vertical time cursors to measure the Start, 0, and 1 pulse durations for your remote.
- ✓ How do they compare to your BASIC Stamp measurements?

Figure 4-30: Cursor Measurement of a Binary-1 Pulse



Frequency 10kHz

Amplitude 1

Offset 0

A 4.88ms

B 6.2ms

Δ 1.33ms

ƒ 753Hz CH1

CH1

1:CH1 in V

2:CH2 in V

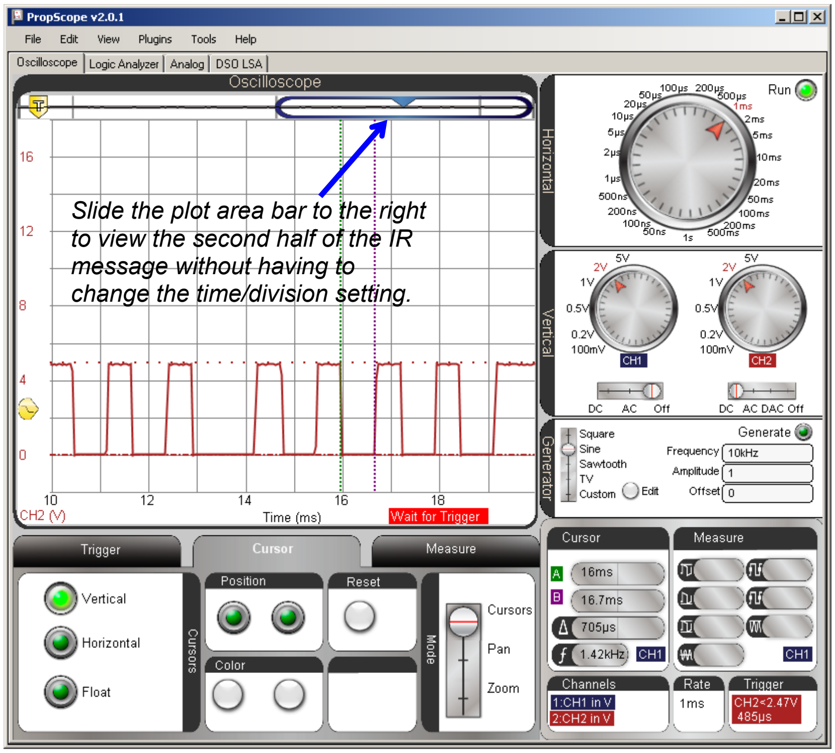
Rate 1ms

Trigger CH2<2.47V 485µs

With the 1 ms/div Horizontal setting, only half the pulses are visible in Figure 4-30. Figure 4-31 shows an example of the rest of the plot viewed by sliding the plot area bar to the right.

✓ Slide the Plot Area bar to the far right of the Plot Preview for a closer inspection of the second half of the IR remote's message.

Figure 4-31: Shift Oscilloscope screen right to See the Rest of the Message



Slide the plot area bar to the right to view the second half of the IR message without having to change the time/division setting.

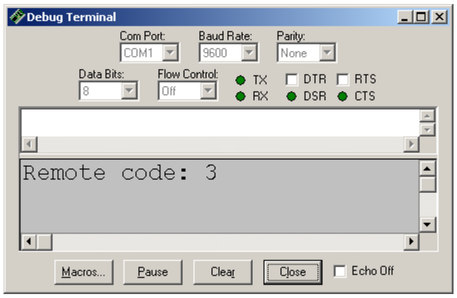
Frequency 10kHz
Amplitude 1
Offset 0

A 16ms
B 16.7ms
Δ 705µs
ƒ 1.42kHz

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 1ms
Trigger: CH2<2.47V 485µs

Your Turn: IR Remote Buttons Program

Figure 4-32 shows the Debug Terminal after the remote's 3 button was pressed with IrRemoteButtons.bs2 running. IrRemoteButtons.bs2 is an excerpt from *IR Remote for the Boe-Bot*.



Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off

```
Remote code: 3
```

Figure 4-32
Debug Terminal Remote Code Display

..with IrRemoteButtons.bs2 running.

- ✓ Download the IR Remote Book Source Code (.zip) from the IR Remote for the Boe-Bot Kit and Text page at www.parallax.com.
- ✓ Open IrRemoteButtons.bs2 with the BASIC Stamp Editor and load it into the BASIC Stamp.
- ✓ Study the codes for buttons like 0 through 9, Power, VOL+, VOL-, CH+, CH-, etc.

```
' -----[ Title ]-----------------------------------------------------------
' IR Remote for the Boe-Bot - IrRemoteButtons.bs2
' Capture and store button codes sent by a universal remote configured to
' control a SONY TV.

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

' -----[ I/O Definitions ]-------------------------------------------------

' SONY TV IR remote declaration - input received from IR detector

IrDet           PIN     9

' -----[ Constants ]-------------------------------------------------------

' SONY TV IR remote constants for non-keypad buttons

Enter           CON     11
ChUp            CON     16
ChDn            CON     17
```

```
VolUp          CON     18
VolDn          CON     19
Power          CON     21

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word
remoteCode     VAR     Byte

' -----[ Main Routine ]----------------------------------------------------

DO
  GOSUB Get_Ir_Remote_Code
  DEBUG CLS, "Remote code: ", DEC remoteCode
  PAUSE 100
LOOP


' -----[ Subroutine - Get_Ir_Remote_Code ]---------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0                             ' Clear all bits in remoteCode

  ' Wait for resting state between messages to end.

  DO
    RCTIME IrDet, 1, irPulse
  LOOP UNTIL irPulse > 1000

  ' Measure start pulse.  If out of range, then retry at Get_Ir_Remote_Code.

  RCTIME 9, 0, irPulse
  IF irPulse > 1125 OR irPulse < 675 THEN GOTO Get_Ir_Remote_Code

  ' Get data bit pulses.

  RCTIME IrDet, 0, irPulse                   ' Measure pulse
  IF irPulse > 300 THEN remoteCode.BIT0 = 1  ' Set (or leave clear) bit 0
  RCTIME IrDet, 0, irPulse                   ' Measure next pulse
  IF irPulse > 300 THEN remoteCode.BIT1 = 1  ' Set (or leave clear) bit 1
  RCTIME IrDet, 0, irPulse                   ' etc
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME IrDet, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME IrDet, 0, irPulse
```

```
IF irPulse > 300 THEN remoteCode.BIT4 = 1
RCTIME IrDet, 0, irPulse
IF irPulse > 300 THEN remoteCode.BIT5 = 1
RCTIME IrDet, 0, irPulse
IF irPulse > 300 THEN remoteCode.BIT6 = 1

' Adjust remoteCode so that keypad keys correspond to the value
' it stores.

IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
IF (remoteCode = 10) THEN remoteCode = 0

RETURN
```

SUMMARY

This chapter introduced pulse width modulation (PWM), and used the PropScope to study PWM signals in several example applications. The first application involved positive pulses sent to a servo's control input to set its output horn's position. Servo control pulse durations can range from 0.5 ms through 2.5 ms, which maps (approximately) to 0 through 180° servo horn positions. Servos are designed to receive pulses at a rate of about 50 Hz, but the frequency of the pulses does not have to be precise. In contrast, the positive pulse durations do have to be precise for good servo horn position control.

The PWM command for D/A conversation uses a duty cycle modulated signal to set voltage across the capacitor in an RC circuit. The key attribute of duty cycle modulation is a signal that stays high for a certain proportion of its cycle time. Infrared object detection and TV remote control demonstrates how PWM can be used for communication. An infrared detector converts brief periods of 40 kHz range infrared signals to brief low signals that contain binary values.

Chapter 5: Synchronous Serial Communication

SYNCHRONOUS SERIAL DEVICES

Two or more devices can exchange binary information with *synchronous serial communication*. The hallmark of this method is that the devices rely on a signal to synchronize the exchange of binary values. For example, one device might send a series of high/low signals, with each high signal indicating that it has made a new binary value available on a separate line for the other device to receive. Those high/low signals ‘synchronize’ the communication of the series of (serial) binary values.

Many electronic devices employ a microcontroller that communicates with other integrated circuits to perform tasks like memory storage, analog to digital conversion, and sensor measurements. Many of these devices are designed to exchange information with the microcontroller using one of a variety of synchronous serial communication protocols. A *protocol* is the set of rules both devices follow for the data exchange to work, and they have names like Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), 3-Wire, 4-Wire, and Microwire.

The BASIC Stamp Module’s EEPROM program memory indicated in Figure 5-1, and it is an example of a synchronous serial peripheral device. The BASIC Stamp module’s Interpreter chip is a microcontroller that uses the I^2C synchronous serial protocol to fetch the program tokens it executes from this chip, and also to store and fetch values when executing PBASIC **WRITE** and **READ** commands.

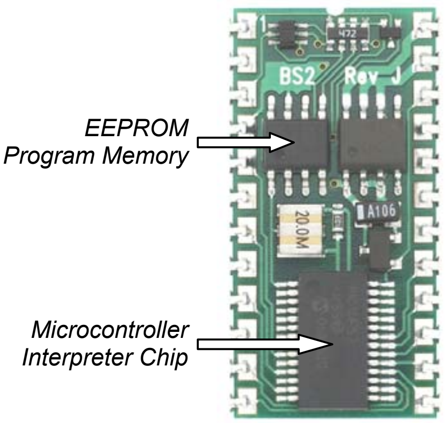


Figure 5-1
BASIC Stamp 2 Module’s Synchronous Serial EEPROM Program Memory

Another example of a synchronous serial device is the ADC0831 analog to digital converter in the Understanding Signals kit. This integrated circuit (IC) is an 8-bit analog to digital converter (A/D converter or ADC). An *analog to digital converter* measures an analog voltage and expresses it as a digital value. 8-bit refers to the number of binary bits in the digital value. An 8-bit number has 8 binary digits and can store any value in the 0 to 255 range. This ADC is a National Semiconductor (NSC) product, and is designed to communicate with the NSC Microwire synchronous serial protocol.

The activities in this chapter will chronicle the development and testing of PBASIC code that makes the BASIC Stamp obtain digitalized voltage measurements from the ADC0831 chip using synchronous serial communication.

ACTIVITY #1: ADC0831 ANALOG TO DIGITAL CONVERTER

Figure 5-2 shows a picture of the ADC0831 analog to digital converter chip along with its pin map. The pin map is similar to the one you will find in the ADC0831 datasheet's Connection Diagrams section. You can download this datasheet from www.national.com.

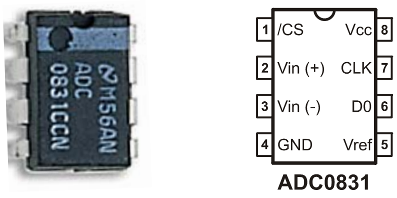


Figure 5-2
ADC0831 A/D Converter Picture and Pin Map

Here is a condensed explanation of the pin map with details that were drawn mainly from the datasheet's Connection Diagrams, Digital Interface and Analog Inputs sections:

- **Vcc** and **GND**: power supply and ground connections for the chip.
- **Vin(+)**: the analog voltage input. The ADC0831 measures the voltage applied to this pin.
- **Vin(-)** and **Vref**: set the limits of the measurement scale. For example, if 1 V is applied to Vin(-) and 4 V is applied to Vref, the Vin(+) voltage will have to fall in the 1 to 4 V range. All the ADC0831's result values would then describe voltages in this range.
- **/CS**: the active low chip select input. A transition from high to low starts an analog to digital conversion, and the signal applied to this pin has to stay low

during conversion and communication. A high signal to this pin “disables” the chip. While the chip is disabled, it will ignore communication, and its DO (data output) pin becomes an input so that it doesn’t interfere with other chips that might also use that line for communication with the microcontroller. The / character is shorthand for active-low, and an over bar that looks like this “$\overline{CS}$” is a common equivalent notation.

- **CLK**: the clock signal input. This is the input that will receive the signal for synchronizing the binary value exchange.
- **DO**: the data output that transmits the A/D converter measurement as a series of binary 1/0 (high/low) values. These values are updated with the negative edge of the (synchronizing) signal applied to the CLK input.

Getting information out of datasheets can be challenging, especially the first few tries. The ADC0831 datasheet is no exception to this rule; its target readers are electronics engineers, and the document covers an entire series of A/D converters, not just the ADC0831. One of the main things that make it easier to decipher datasheets is practice. So, it’s still a good idea to read it and try to understand as much as you can. Datasheets for parts you work with are also good to keep on hand for reference because they contain answers to questions like, “what’s the highest allowable supply voltage for this chip?”

Stamps in Class books, how-to articles, and microcontroller application notes often contain information for getting a given device “up and running.” They also tend to feature lots of useful techniques and helpful background information.

ADC0831 Test Parts

(1) ADC0831
(1) Potentiometer – 10 kΩ (103)
(3) Resistors – 220 Ω (red-red-brown)
(misc.) Jumper wires

ADC0831 Test Circuit

The schematic in Figure 5-3 and wiring diagram example in Figure 5-4 were set up based on the pin map and pin descriptions accompanying Figure 5-2. To make prototyping easier, the Vin(-) pin is tied to Vss = 0 V and the Vref pin is tied to Vdd = 5 V. This sets the bottom of the ADC0831’s input voltage scale to 0 V and the top to 5 V. The PropScope’s CH1 probe is connected to the potentiometer’s wiper so that the voltage it applies to the ADC0831’s Vin(+) input can be compared to the value the ADC0831 reports to the BASIC Stamp. Three of the PropScope DAC Card’s four Logic Analyzer inputs are also connected to the lines between the BASIC Stamp I/O pins and the

ADC0831 /CS, CLK and DO pins. They will be used to monitor the synchronous serial communication between the two devices.

✓ Build the schematic in Figure 5-3 using the wiring diagram example in Figure 5-4 as a guide.

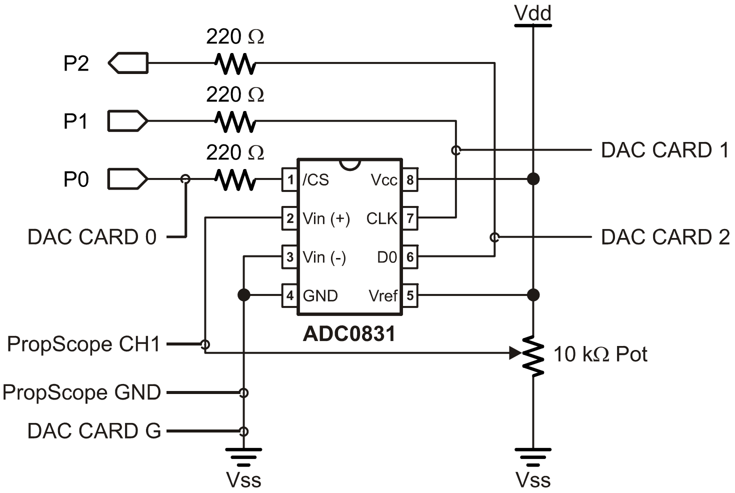


Figure 5-3
ADC0831 Test Schematic

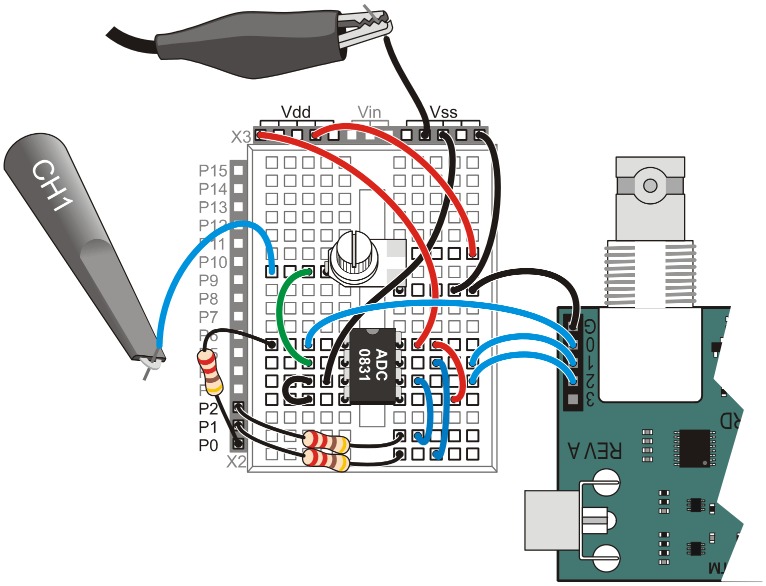
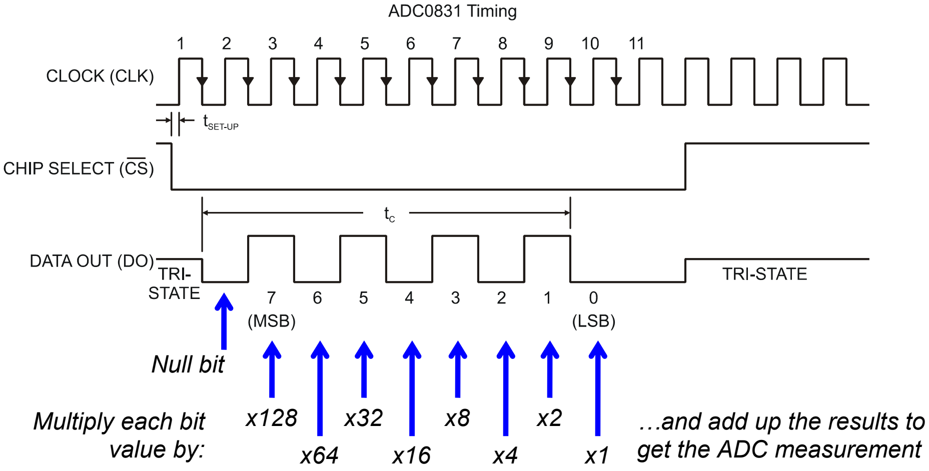


Figure 5-4
Wiring Diagram Example for Figure 5-3

ACTIVITY #2: WRITE CODE FROM A TIMING DIAGRAM

Figure 5-5 shows a timing diagram similar to the one in the ADC0831 datasheet. This diagram explains the signaling a microcontroller has to use to get measurements from the chip. The little inverted triangles on the CLK signal indicate that the DO line updates its output on the clock pulse's negative edge. So the BASIC Stamp should wait until it applies a falling clock edge before checking the next binary value from the DO line.

Figure 5-5: Timing Diagram from ADC0831 Datasheet



If you had to send a binary value like 10101010 as a series of high/low signals, would you send the leftmost digit first and work your way to the right, or would you send the rightmost digit first and work your way to the left? If you send the leftmost digit first and work your way to the right, you would be sending the values in the *most significant bit first order* (MSB first). If you instead work from right to left, you would be sending values in the *least significant bit first* (LSB first) order.

The ADC0831 transmits the binary values 'MSB first.' So the first binary value the DO line transmits is the number of 128s in the measurement. The second binary value is the number of 64s in the measurement, followed by the number of 32s and so on, down to the least significant bit, which is the number of 1s in the measurement. The example measurement in the timing diagram is:

$$(1 \times 128) + (0 \times 64) + (1 \times 32) + (0 \times 16) + (1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) = 170$$

> **The leftmost digit in a base-2 unsigned integer is "most significant"** because it is the digit that tells how many of the largest power of 2 the number contains. In an 8-bit binary number, that's how many 2^7 = 128 it contains. The rightmost digit is the least significant bit because it tells how many of the smallest power of 2 the number contains. In an 8-bit binary number, that's how many 2^0 = 1 it contains.

From the timing diagram in Figure 5-5, we can make a list of tasks for the BASIC Stamp:

Step 1: Set the /CS line low.
Step 2: Send the first clock pulse (a low-high-low signal) to the ADC0831's CLK pin. Don't worry about collecting any data yet. The ADC0831 just sends a low signal (the *null bit*) in reply to this first pulse.
Step 3: Send a second clock pulse, and record the binary value the DO pin transmits in bit 7 of the variable that stores the measurement.
Step 4: Apply a third clock pulse to CLK and store the DO value in bit 6 of the measurement variable.
Step 5: Repeat 6 more pulses to CLK, each time storing the value DO transmits in a successively lower bit in the result variable.
Step 6: Set /CS line high.

ADC0831Test1.bs2 is a program that performs the 6 steps from the datasheet, plus a seventh step: display the measurement in the Debug Terminal as an 8-digit binary value and as a 3-digit decimal value.

```
' ADC0831Test1.bs2
' Code from the ADC0831 timing diagram.

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

CS             PIN     0                     ' P0 -> ADC CS
CLK            PIN     1                     ' P1 -> ADC CLK
Dout           PIN     2                     ' P2 <- ADC DO

adcVal         VAR     Byte                  ' ADC result variable

PAUSE 1000                                   ' Delay 1 s before 1st message
DEBUG CLS                                    ' Clear display

HIGH CS                                      ' Start CS high
LOW CLK                                      ' Start CLK low
INPUT Dout                                   ' Set Dout to input
```

```
DO                                           ' Main Loop

  LOW CS                                     ' CS -> low start measurement

  PULSOUT CLK, 200                           ' First pulse, no data follows
  PAUSE 0                                    ' For display, time btwn pulses
  PULSOUT CLK, 200                           ' Second pulse
  adcVal.LOWBIT(7) = Dout                    ' Get bit 7
  PULSOUT CLK, 200                           ' Third pulse
  adcVal.LOWBIT(6) = Dout                    ' Get bit 6
  PULSOUT CLK, 200                           '...etc
  adcVal.LOWBIT(5) = Dout
  PULSOUT CLK, 200
  adcVal.LOWBIT(4) = Dout
  PULSOUT CLK, 200
  adcVal.LOWBIT(3) = Dout
  PULSOUT CLK, 200
  adcVal.LOWBIT(2) = Dout
  PULSOUT CLK, 200
  adcVal.LOWBIT(1) = Dout
  PULSOUT CLK, 200                           ' Ninth pulse
  adcVAl.LOWBIT(0) = Dout                    ' Get bit 0

  HIGH CS                                    ' CS -> High, disable ADC

  DEBUG HOME,                                ' Display measurement
        "Binary  adcVal = ", BIN8 adcVal, CR,'   in binary
        "Decimal adcVal = ", DEC3 adcVal     '   and decimal

  PAUSE 100                                  ' Screen update delay

LOOP                                         ' Repeat main loop
```

ADC0831Test1.bs2 might seem inefficient and complicated from the PBASIC programming standpoint, especially since PBASIC has a pair of commands named **SHIFTIN** and **SHIFTOUT** that automatically do most of this work for you in a single line of code. It's still a useful exercise because not all microcontroller programming languages have built-in commands that automate synchronous serial communication. ADC0831Test1.bs2 shows you how to use lower-level commands to implement a timing diagram.

ACTIVITY #3: VERIFY MICROCONTROLLER SIGNALING

The ADC0831 transmits a measurement as a number of least significant bit voltage increments (V_{LSB}). This voltage increment is just enough to make its measurement step from 0 to 1. The increment's size depends on the range of the voltage the device is configured to measure. The top of this range is set by the voltage applied to Vref, and the bottom of this range is set by the voltage applied to Vin(-). Since the ADC0831 splits whatever voltage range it measures into 256 levels, the value of V_{LSB} is:

$$V_{LSb} = \frac{Vref - Vin(-)}{256}$$

Since Vref is connected to Vdd = 5 V, and Vin(-) is connected Vss = 0 V, the ADC0831's measurements will be in terms of 256^{ths} of 5 V. So, that makes one LSB voltage increment:

$$\begin{aligned} V_{LSb} &= \frac{5V - 0V}{256} \\ &= 0.01953125V \\ &= 19.53125\,mV \end{aligned}$$

V_{LSB} vs. LSB. Analog to digital converter documentation often just refers to one V_{LSB} voltage increment as an LSB. LSB also refers to the rightmost binary digit in a binary number. The two are related because the voltage that causes an ADC measurement to step from 0 to 1 causes the binary number that describes the voltage measurement to store a 1 in its least significant bit. If you see LSB in this book, it means the rightmost digit in a binary number. If you see V_{LSB}, it means an analog to digital converter voltage increment.

Quantize and Quantization. If you see the term *quantize* in A/D converter documentation, it means that an analog to digital converter quantifies its voltage measurements (in terms of a number of V_{LSB} increments). The A/D converter "quantizes" voltage. *Quantization* is the process of making a conversation (from input voltage to a number of V_{LSB} increments). You might also see the term *quantization error* which refers to the difference between the actual voltage and the measured (quantized) value.

You can use V_{LSB} to calculate an analog to digital converter's measurement in terms of volts. Simply multiply the value the ADC reports by V_{LSB}. For example, if the ADC0831 reports 170, as it did in the Figure 5-5 timing diagram, the voltage applied to the Vin(+) terminal must be:

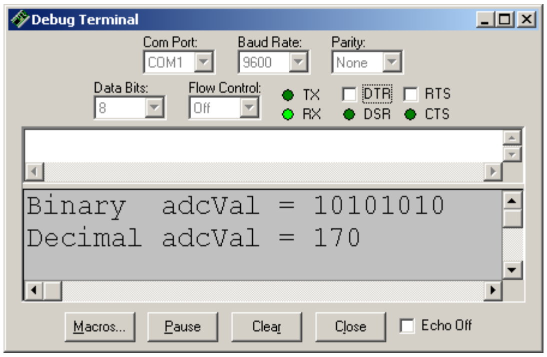
$$\begin{aligned} Vin(+) &= adcVal \times V_{LSB} \\ &= 170 \times 19.53125\,mV \\ &= 3.3203125V \\ &\approx 3.32V \end{aligned}$$

You can also calculate what the ADC will report for a certain voltage measurement by dividing it by V_{LSB}. For example, if you want to know what the ADC0831 will report if 3.32 V is applied to its Vin(+) terminal, divide the LSB voltage increment into it.

$$\begin{aligned} adcVal &= \frac{Vin(+)}{V_{LSB}} \\ &= \frac{3.32\ V}{19.53125\, mV} \\ &= 169.984 \\ &\approx 170 \end{aligned}$$

Let's verify the signaling by adjusting the potentiometer until the ADC0831 reports 170. With this value, the PropScope should measure a voltage of about 3.3 V.

- ✓ Enter and run ADC0831Test1.bs2.
- ✓ Adjust the potentiometer until the Debug Terminal shows that the ADC0831 is reporting a value of 170 like Figure 5-6.



```
Binary  adcVal = 10101010
Decimal adcVal = 170
```

Figure 5-6
ADC0831 Output with Pot set to 3.3 V

The PropScope DSO LSA view in Figure 5-7 is great for verifying the voltage measurement and the synchronous serial communication signaling because it shows both the Oscilloscope and Logic State Analyzer screens. Remember that the potentiometer's wiper terminal output voltage is connected to the ADC0831's Vin(+) pin as well as to the PropScope's CH1 probe. So, we can use the Oscilloscope to verify the voltage the potentiometer applies to the ADC0831's Vin(+) pin.

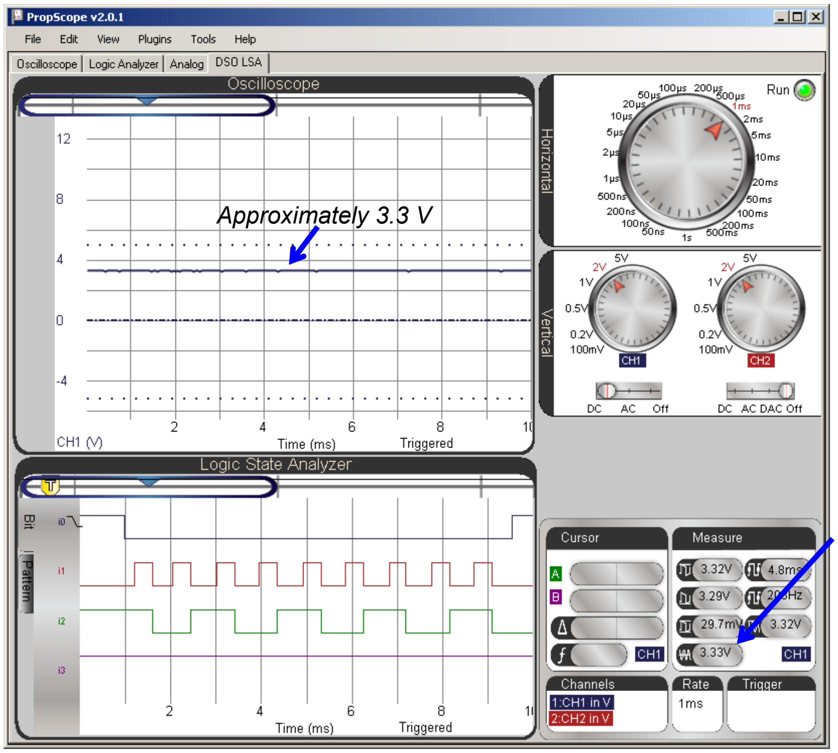
DSO: Digital Storage Oscilloscope.

LSA: Logic State Analyzer.

Mixed Signal: Refers to a combination of analog and digital signals. The DSO LSA view in Figure 5-7 is a mixed signal view.

- ✓ First, click the Oscilloscope view and set the Trigger tab’s Mode switch to Off.
- ✓ Click the DSO LSA view tab to see the Oscilloscope and Logic State Analyzer screens at the same time.
- ✓ Set the CH1 Vertical dial to 1 V/division, and the Vertical coupling switch to DC.
- ✓ Verify that the CH1 voltage measures 3.3 V.

Figure 5-7: Check Voltage and Communication with DSO LSA View



Approximately 3.3 V

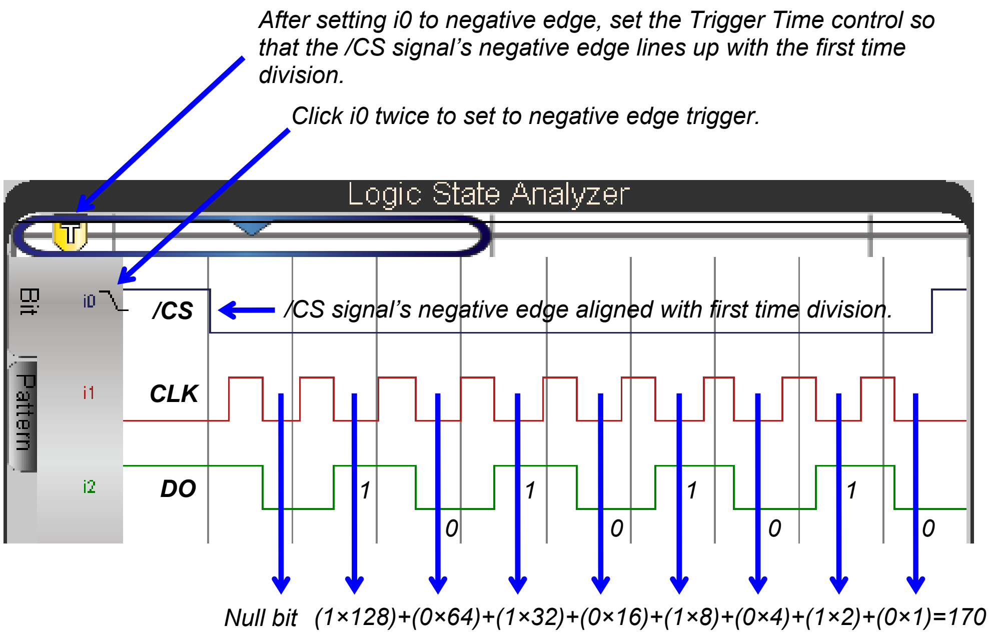
Confirmation of the 3.3 V measurement in the Oscilloscope screen

The Logic State Analyzer screen in Figure 5-8 is useful for synchronous serial signaling verification. The ADC0831Test1.bs2 code example from Activity #2 implements the timing diagram in Figure 5-5, and the Logic State Analyzer display in Figure 5-8 verifies that it's doing it right.

In Activity #1, you connected the PropScope DAC Card's Logic State Analyzer input 0 to the ADC0831's /CS line. So, the /CS signal will be displayed by the trace labeled i0. The first thing the program's main loop does is set the /CS line low to start a measurement and communication. It also sets /CS high again when it's done. Since the first communication event is the /CS line's negative edge transition (from high to low), the Logic State Analyzer can be configured to trigger on this event. You can do this by setting the Logic State Analyzer's i0 trace to negative edge trigger. After that, you will also have to adjust the horizontal scale so that the entire serial exchange is visible in the Oscilloscope screen. To match Figure 5-8, the Trigger Time control should also be adjusted to align the i0 trace's negative edge with the first time division line.

- ✓ Set the Horizontal dial to 1 ms/division.
- ✓ Repeatedly click the i0 signal label on the left side of the Logic State Analyzer screen until it displays a negative edge i0 symbol.
- ✓ Adjust the Logic State Analyzer's Trigger Time control T so that it aligns its vertical crosshair with the first time division line. This should in turn cause the i0 trace's negative edge to line up with the first time division line.
- ✓ Compare the signaling in Figure 5-8 to the timing diagram in Figure 5-5.
- ✓ Check i2 in the Logic State Analyzer to verify that the 1s and 0s transmitted by the ADC0831's DO line are the correct sequence for the binary number 10101010 = decimal 170. Figure 5-8 shows an example of how to do this.

Figure 5-8: Verify Value Transmitted by ADC0831 with Logic Analyzer


After setting i0 to negative edge, set the Trigger Time control so that the /CS signal's negative edge lines up with the first time division.

Click i0 twice to set to negative edge trigger.

/CS signal's negative edge aligned with first time division.

Null bit (1×128)+(0×64)+(1×32)+(0×16)+(1×8)+(0×4)+(1×2)+(0×1)=170

Your Turn: Try a Different Voltage

- ✓ Pick a target voltage in the 0 to 5 V range.
- ✓ Calculate the **`adcVal`** result.
- ✓ Adjust the potentiometer until the Debug Terminal displays your **`adcVal`** result.
- ✓ Verify the approximate voltage with the Measure display.
- ✓ Verify the signaling and make sure the DO values during the low clock cycles coincides with the value of **`adcVal`** displayed in the Debug Terminal. Again, Figure 5-8 shows an example of how to do this.

ACTIVITY #4: TROUBLESHOOTING EXAMPLES

Author's note: When I started designing the circuit and code for this chapter, I took advantage of the Oscilloscope and Logic Analyzer in the DSO LSA view illustrate common mistakes one might make along the way. Here are three errors it would be easy to make:

1) Not setting /CS high between measurements.
2) Not sending a clock pulse for the null bit before the eight pulses for data bits.
3) Swapping CLK and DO connections. As a result of this error, the first test program would send clock signals to the DO pin instead of the CLK pin. Since the DO pin also sends signals, this could possibly create a situation where the BASIC Stamp I/O pin sends a high signal while the DO pin sends a low. Without the 220 Ω resistor to limit current, either the ADC0831, or the BASIC Stamp, or both could be damaged.

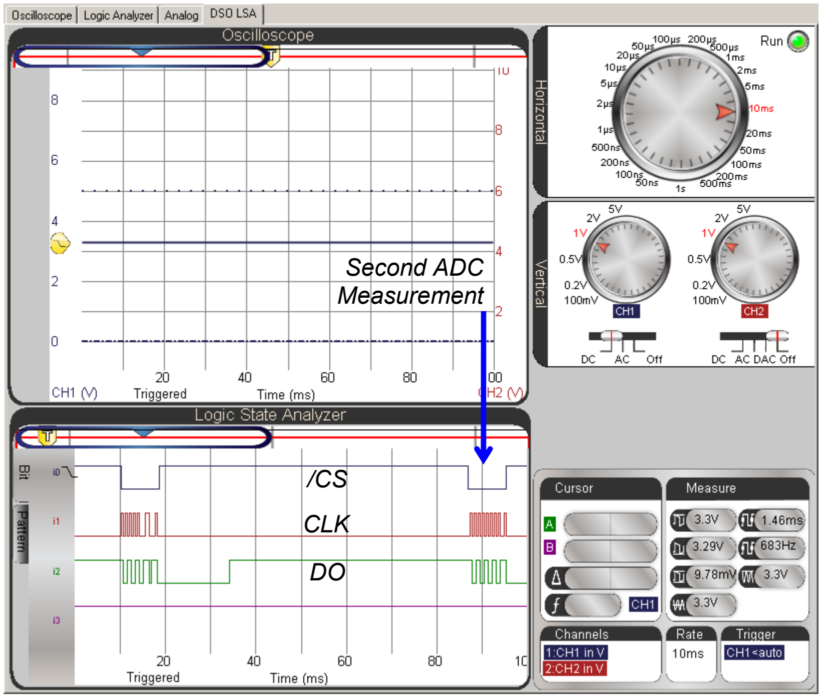
The Oscilloscope and Logic State Analyzer are useful diagnostic tools for finding and fixing errors. In this activity, we'll examine how the DSO LSA view's Oscilloscope and Logic Analyzer can be used to diagnose problems 1 and 2.

Recreate and Examine Error 1

Before we recreate error 1, it's important to zoom out a little to see multiple ADC communications. Figure 5-9 shows an example with the **`PAUSE 100`** command in the commented out of the example code and the Horizontal dial adjusted to 10 ms/division. In the Logic State Analyzer, the /CS line stays high between measurements, and there are two measurements visible.

- ✓ Remove the **`PAUSE 100`** from ADC0831Test1.bs2 by placing an apostrophe to its left so that it reads **`' PAUSE 100`**.
- ✓ Load the Modified program into the BASIC Stamp.
- ✓ Change the Horizontal dial setting in the PropScope's DSO LSA view to 10 ms/division.

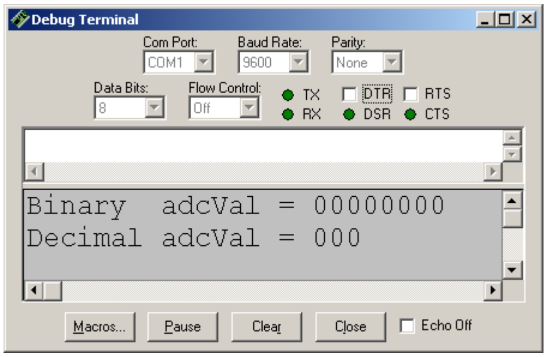
Figure 5-9: Two ADC0831 Measurements in Logic Analyzer



Now that we know what to expect when it's working right, let's try disabling the command that sets /CS high between measurements.

- ✓ In ADC0831Test1.bs2, find the command **`HIGH CS`** that's near the end of the **`DO…LOOP`**. (Not the one at the beginning of the program.)
- ✓ Comment the HIGH CS that's near the end of the program's **`DO…LOOP`** by placing an apostrophe to its left, like this: **`' HIGH CS`**.
- ✓ Load the modified program into the BASIC Stamp 2.

The potentiometer is still adjusted to 3.32 V, but Figure 5-10 shows that the change in the code caused the system to report all zeros instead of the correct values.



Com Port: COM1 Baud Rate: 9600 Parity: None
Data Bits: 8 Flow Control: Off

```
Binary  adcVal = 00000000
Decimal adcVal = 000
```

Figure 5-10
Debug Terminal with Zero Measurements

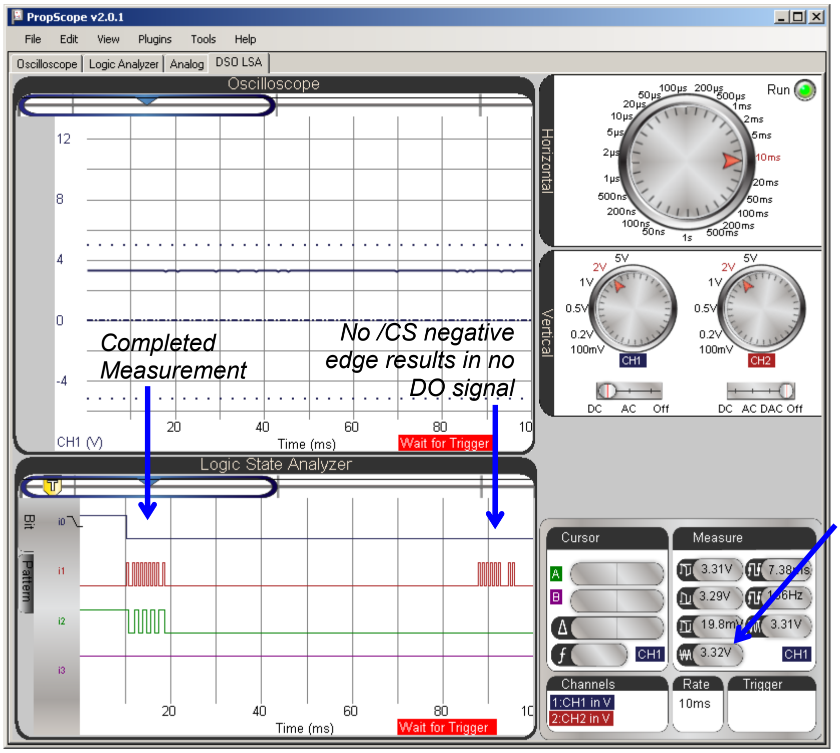
…because /CS was not set high between measurements.

If the Binary `adcVal` result in the Debug Terminal shows all ones instead of all zeros, it may mean that the first measurement the ADC0831 successfully completed when the program started was an odd number like 169 or 171 instead of 170. So the first and only successful measurement left the ADC0831's DO line high instead of low because odd numbers always have a 1 in the LSB, and the LSB is the last digit the DO line transmits.

For example, the odd decimal numbers 1, 3 and 5 have the binary equivalents: 001, 011, and 101. The rightmost digits are all ones. In contrast, even decimal numbers like 2, 4 and 6 end in zeros, and their binary equivalents are: 010, 100, and 110.

Figure 5-11 shows that the measured voltage on CH1 is still 3.3 V. It would have to be in the 0 to 19.5 mV range for a Debug Terminal to display of all zeros. Take a closer look at the BASIC Stamp module's second measurement attempt in the Logic State Analyzer. The /CS pin has to be set high and then low again to start a measurement. In the second measurement the BASIC Stamp sends the clock pulses to /CS, but the ADC0831's DO line doesn't reply because the /CS line was not set high and then low again to start the measurement.

Figure 5-11: CS is stays low, so DO does not respond to clock pulses after first measurement.



Completed Measurement

No /CS negative edge results in no DO signal

Voltage applied to Vin(+) is still 3.3 V.

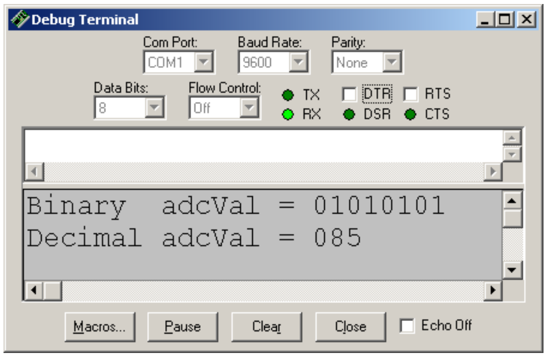
Before examining the next error, the example code should be returned to its original state and tested.

- ✓ Uncomment **`' HIGH CS`** by deleting the apostrophe.
- ✓ Uncomment **`' PAUSE 100`** by deleting the apostrophe.
- ✓ Make sure the potentiometer is still set to 3.3 V.
- ✓ Load the corrected program into the BASIC Stamp.
- ✓ Verity that the Debug Terminal still reports measurements of 170.
- ✓ Return the Horizontal dial to 1 ms/div.
- ✓ Verify that the synchronous serial communication still displays in the Logic State Analyzer screen like it did in Figure 5-8.

Recreate and Examine Error 2

Figure 5-12 shows what happens if the first **`PULSOUT CLK, 200`** is missing from the program; the adcVal variable indicates the measurement is only half of what it should be.

- ✓ Comment out the very first **`PULSOUT CLK, 200`** by placing an apostrophe to its left so that it looks like this: **`' PULSOUT CLK, 200`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Use the Debug Terminal to verify that your ADC measurement results have been cut in half.
- ✓ Compare the binary pattern in Figure 5-12 to the one in Figure 5-6 on page 157. Would you agree that the binary digits in Figure 5-12 shifted to the right by 1 digit?



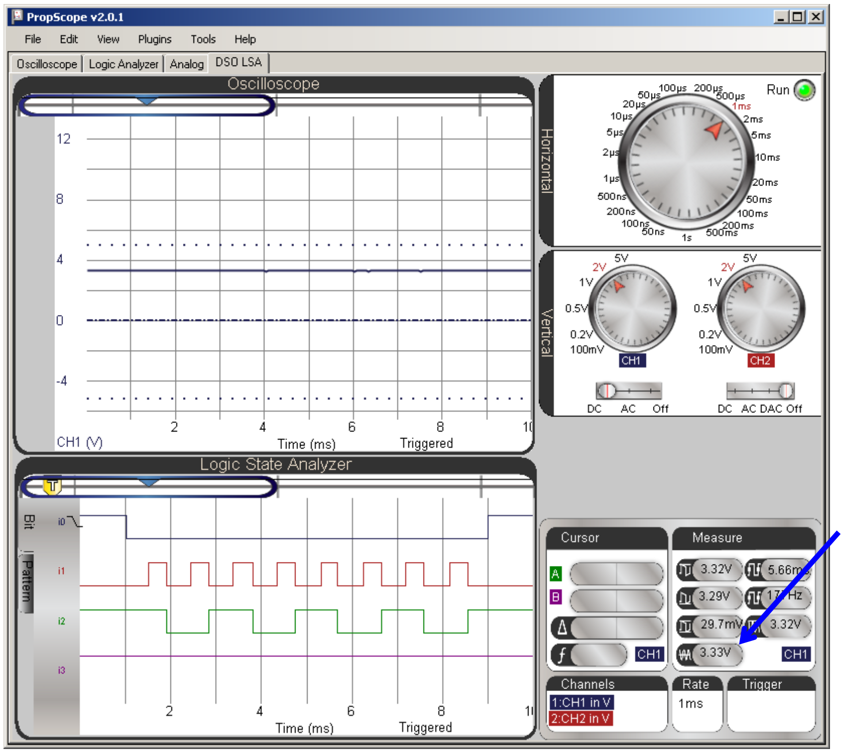
Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off

```
Binary  adcVal = 01010101
Decimal adcVal = 085
```

Figure 5-12
Missing Clock Pulse Causes ADC0831 to Return Half the Measured Value

Figure 5-13 shows that the voltage is still 3.3 V, and the signaling in the Logic Analyzer might “seem” in order, but if you examine it closely and compare it to the Figure 5-5 timing diagram on page 153, you’ll see that they are not the same.

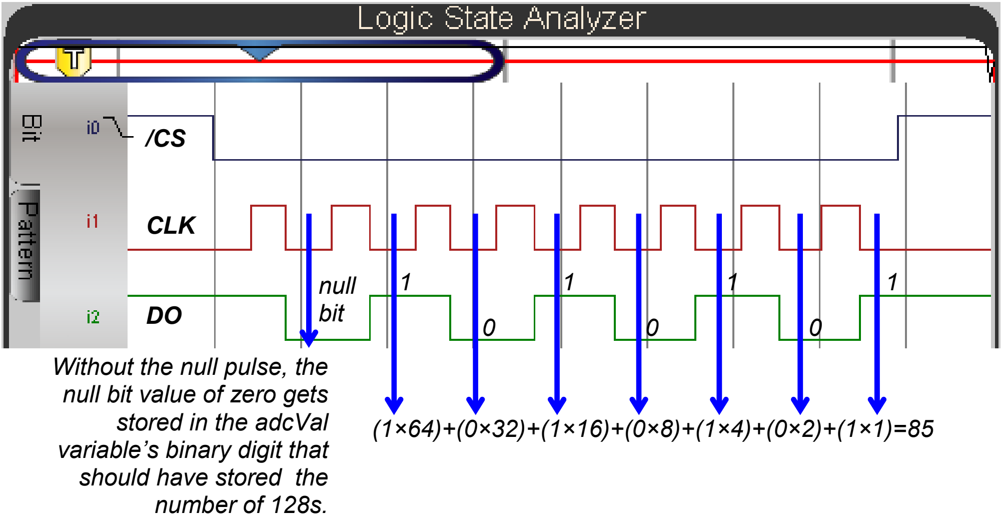
Figure 5-13: First Look at the Communication with Missing Clock Pulse



Voltage applied to Vin(+) is still 3.3 V

A closer look at the Logic State Analyzer display in Figure 5-14 shows the problem. Instead of 9 pulses, there are only 8. The ADC0831 uses the first pulse for its own processing regardless of how many clock pulses it receives, and the DO line sends a zero for this pulse, which the figure calls a *null bit*. Instead of not paying any attention to that first null bit, your modified PBASIC program stores it in **`adcVal`** bit 7. So a 0 gets stored in bit 7 of **`adcVal`** instead of a 1. Then, the value that should have represented the number of 128s in bit 7 gets stored in bit 6. So now, it represents the number of 64s instead. Likewise, the value that should have represented the number of 64s got stored in bit 6, so now it represents the number of 32s. The end result? All the binary digits got shifted to the right in the **`adcVal`** variable. In binary processing, this is a shortcut used to divide a number by 2. So the result of 85, which is ½ of 170 also provides a clue that there weren't enough clock pulses.

Figure 5-14: Closer Look at Communication with Missing Clock Pulse



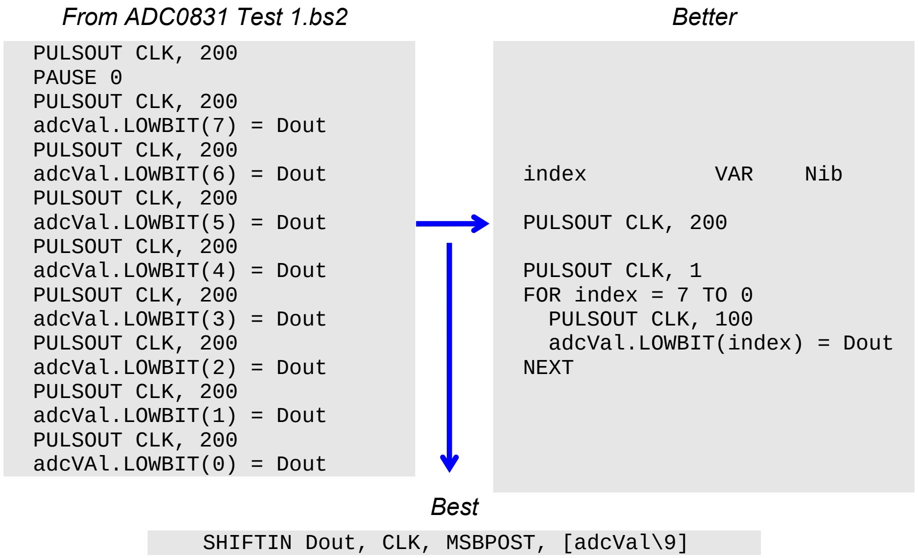
Without the null pulse, the null bit value of zero gets stored in the adcVal variable's binary digit that should have stored the number of 128s.

(1×64)+(0×32)+(1×16)+(0×8)+(1×4)+(0×2)+(1×1)=85

ACTIVITY #5: REFINE AND TEST THE CODE

As mentioned earlier, the code from ADC0831Test1.bs2 is not optimized. Figure 5-15 shows two ways to improve the code. The "better" version uses an **`index`** variable and **`FOR…NEXT`** loop to iteratively load each bit from the ADC0831 into the **`adcVal`** variable. The "best" version uses a PBASIC command called **`SHIFTIN`** to complete the operation, including a first null bit pulse, in a single command. Not only does this command take less code space, it is also much faster.

Figure 5-15: "Better" and "Best" Approaches to ADC0831 Communication



From ADC0831 Test 1.bs2

```
PULSOUT CLK, 200
PAUSE 0
PULSOUT CLK, 200
adcVal.LOWBIT(7) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(6) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(5) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(4) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(3) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(2) = Dout
PULSOUT CLK, 200
adcVal.LOWBIT(1) = Dout
PULSOUT CLK, 200
adcVAl.LOWBIT(0) = Dout
```

Better

```
index          VAR    Nib

PULSOUT CLK, 200

PULSOUT CLK, 1
FOR index = 7 TO 0
  PULSOUT CLK, 100
  adcVal.LOWBIT(index) = Dout
NEXT
```

Best

```
SHIFTIN Dout, CLK, MSBPOST, [adcVal\9]
```

The syntax for the **`SHIFTIN`** command is:

SHIFTIN *DataPin*, *ClockPin*, *Mode*, [*Variable* {\*bits*}{, *Variable* {\bits}…}

The ***DataPin*** and ***ClockPin*** are self explanatory. The ***Mode*** argument can be one of four values: **`MSBPRE`**, **`MSBPOST`**, **`LSBPRE`**, or **`LSBPOST`**. The ADC0831 transmits the most significant bit (MSB) first and updates its DO data output after the clock pulse (POST). So the correct value to use as the **`SHIFTIN`** command's ***Mode*** argument is **`MSBPOST`**. The **`\9`** after **`adcVal`** applies 9 pulses, and shifts in 9 bits. But, since the **`adcVal`** variable is a byte, it only retains the lowest 8 bits, and the null bit is lost, which is fine because it didn't hold any meaningful information anyhow.

✓ Enter and run ADC0831Test2.bs2.

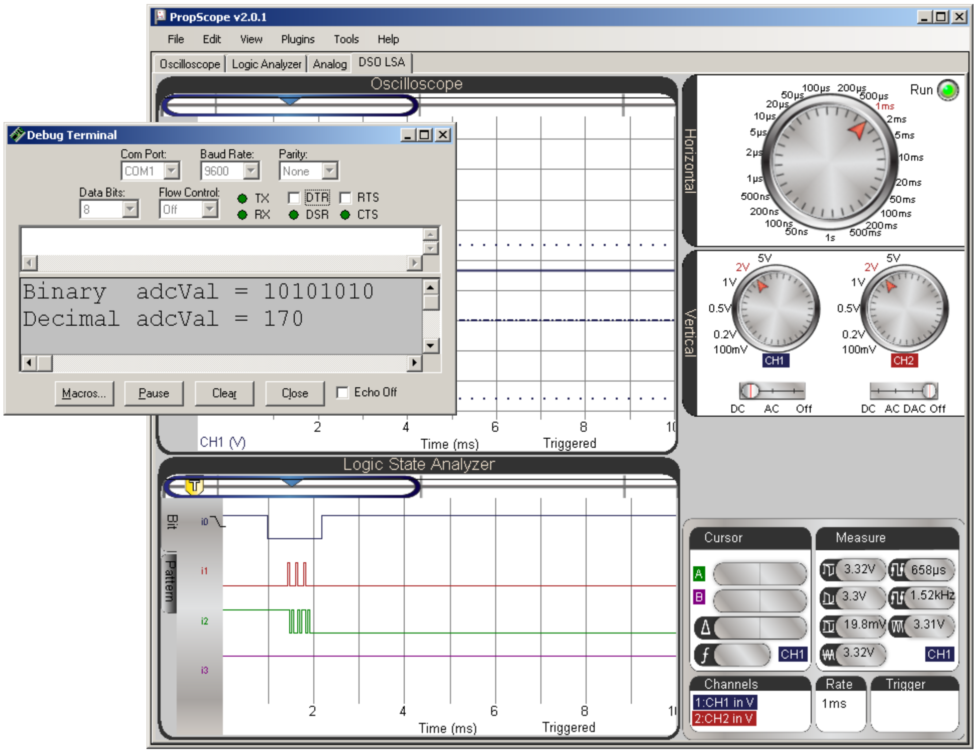
```
' ADC0831Test2.bs2
' Code from the ADC0831 timing diagram.

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

CS             PIN     0                     ' P0 -> ADC CS
CLK            PIN     1                     ' P1 -> ADC CLK
Dout           PIN     2                     ' P2 <- ADC DO

adcVal         VAR     Byte                  ' ADC result variable
index          VAR    Nib

PAUSE 1000                                   ' Delay 1 s before 1st message
DEBUG CLS                                    ' Clear display

HIGH CS                                      ' Start CS high
LOW CLK                                      ' Start CLK low
INPUT Dout                                   ' Set Dout to input

DO                                           ' Main Loop

  LOW CS                                     ' CS -> low start measurement
  SHIFTIN Dout, CLK, MSBPOST, [adcVal\9]     ' 9 CS pulses, keep 8 DO bits
  HIGH CS                                    ' CS -> High, disable ADC

  DEBUG HOME,                                ' Display measurement
        "Binary  adcVal = ", BIN8 adcVal, CR,'   in binary
        "Decimal adcVal = ", DEC3 adcVal     '   and decimal

  PAUSE 100                                  ' Screen update delay

LOOP                                         ' Repeat main loop
```

PBASIC has `SHIFTOUT` too. For sending synchronous serial messages to devices designed to receive them, use the **`SHIFTOUT`** command.

Figure 5-16 shows that the Debug Terminal is still displaying the correct information, but the **`SHIFTOUT`** command is finishing its job in a fraction of the time. In fact, it's running so quickly, that the current 1 ms/division Horizontal dial setting is too large to catch all the signal activity in the Logic State Analyzer.

✓ Verify that your Debug Terminal and Logic Analyzer resemble the example in Figure 5-16.

Figure 5-16: Debug Terminal Check, and SHIFTIN in Logic State Analyzer



Com Port: COM1 Baud Rate: 9600 Parity: None

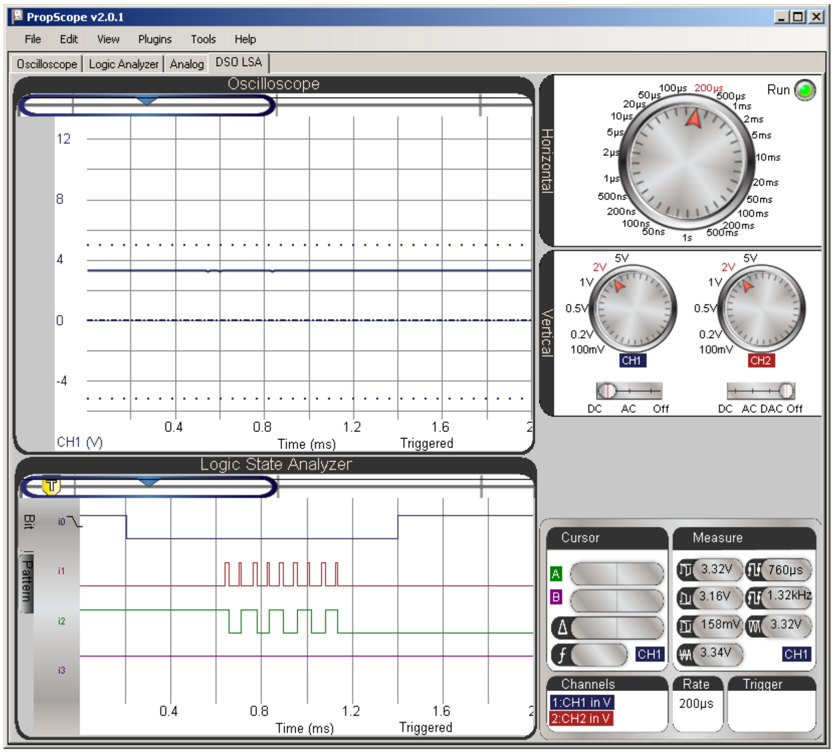
Data Bits: 8 Flow Control: Off

```
Binary  adcVal = 10101010
Decimal adcVal = 170
```

Figure 5-17 shows the Logic State Analyzer with the Horizontal dial set to 200 µs/division, which is small enough to see the pulses and data values, but large enough to still contain the entire /CS signal low time.

✓ Adjust the PropScope's Horizontal dial to 200 µs/division.

Figure 5-17: Closer Look at the Higher Speed SHIFTIN Command



Your Turn: Pattern Detection

When two devices communicate, troubleshooting may need to start at a very small time window when a list of certain binary conditions occur. The Pattern tab on the left side of the Logic Analyzer can be used to detect a list of conditions, like the ones shown in Figure 5-18. By clicking each signal label (i0, i1, i2, and/or i3) repeatedly, you can toggle through these five trigger conditions:

i0 → Rising (positive) Edge

i0 → Falling (negative) Edge

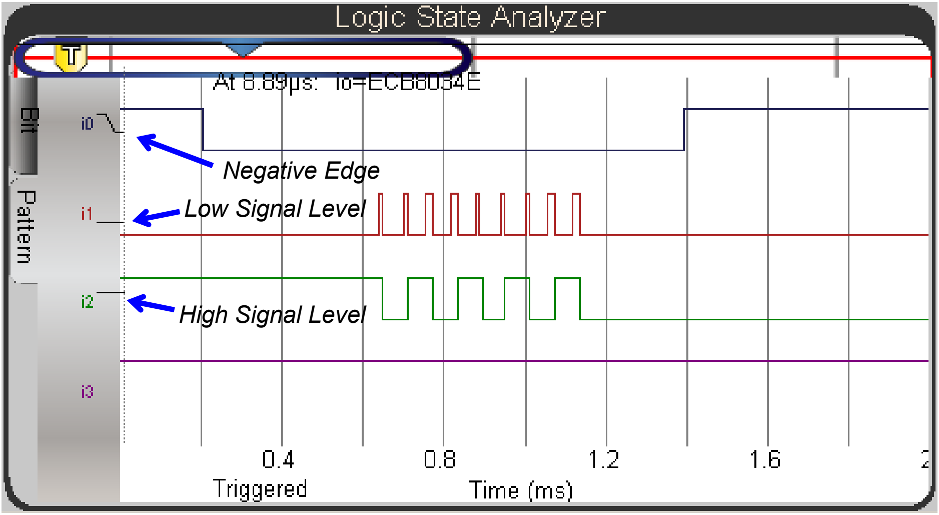
i0 → High

i0 → Low

i0 → No Trigger

- ✓ Set your PropScope's Horizontal dial to 100 µs/division.
- ✓ Click the Pattern tab on the side of the Logic State Analyzer.
- ✓ Set up the pattern of trigger conditions shown in Figure 5-18.
- ✓ Try creating a signal pattern that never happens. For example, set i0 to high and i1 to positive edge. What happens to the screen activity and what message appears? The Logic State Analyzer keeps waiting for a trigger event that doesn't occur.
- ✓ Try setting i0 to trigger on low and i1 to trigger on rising edge. The rising edge will line up with the fist division line because the i0 = Low condition was satisfied, but the display still has to wait for a rising edge on i1.
- ✓ Experiment with clicking through different options on different channels, and examine any changes in signal edge alignment with the first time division line.

Figure 5-18: Pattern Triggering



More ADC0831 Projects. *Basic Analog and Digital* and *Process Control* are two Stamps in Class textbooks/kits with activities and projects that utilize the ADC0831 A/D converter. Both textbooks are free .pdf downloads from www.parallax.com/education.

SUMMARY

A variety of inexpensive, special-purpose integrated circuits are available to perform tasks like memory storage, coprocessing and analog to digital conversion (to name a few), and many of them are designed to exchange information with a microcontroller using synchronous serial communication. Synchronous serial devices use a variety of protocols, with a common denominator of a clock signal that synchronizes the exchange of a series of binary values.

The ADC0831 8-bit A/D converter is an example of a synchronous serial device that transmits a number from 0 to 255 to represent measured voltages. This device's datasheet provides pin maps and timing diagrams that can be used to develop a test circuit and test code. Communication takes signals on three lines: chip select, clock, and data out. Chip select has to be held low for the duration of the communication, and pulses applied to the chip's clock pin result in successive binary values that represent the measurement value transmitted by its data out pin.

The PropScope's DSO LSA View has an Oscilloscope and Logic Analyzer. This arrangement makes it possible to verify both the voltage applied and the synchronous serially-communicated measurement. The oscilloscope can be used to view the voltage applied to the ADC0831's Vin(+) analog input while the Logic Analyzer monitors the communication lines to verify that the signaling is correct.

One incorrect microcontroller coding error can change the signaling in ways that prevent communication, or cause incorrect values. Two such errors were studied in this chapter, and the digital signal oscilloscope and logic state analyzer were used to diagnose each error.

PBASIC has **SHIFTIN** and **SHIFTOUT** commands for communicating with synchronous serial devices. Provided the command's arguments and features are used correctly, they can make short work of synchronous serial communication code, and they are also much faster than loops that store individual bits in a variable.

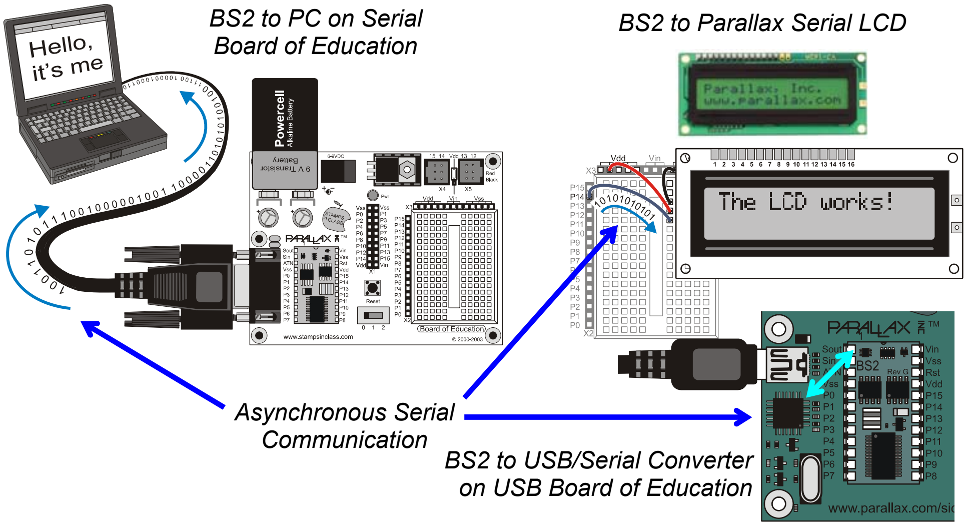
Chapter 6: Asynchronous Serial Communication

The previous chapter introduced synchronous serial communication, which relies on a separate clock signal to synchronize the exchange of a series of values. In contrast, *asynchronous serial communication* is the exchange of a series of values without the synchronizing clock signal.

ASYNCHRONOUS SERIAL DEVICES

Figure 6-1 shows some examples of devices that use asynchronous serial communication to exchange information. When a BASIC Stamp executes a **DEBUG** command, it uses asynchronous serial communication to send information to the PC, either through a serial cable, or to a chip that converts the serial message to the USB protocol.

Figure 6-1: Asynchronous Serial Device Examples



Another example: the BASIC Stamp sends the Parallax Serial LCD (Liquid Crystal Display) asynchronous serial messages to make it display text. Other useful peripherals that utilize asynchronous serial communication include geographic positioning systems (GPS), radio frequency identification (RFID), and radio frequency communication modules that make it possible for microcontrollers to communicate wirelessly. This chapter introduces the signaling these devices use to communicate and examines and decodes certain asynchronous serial messages with the PropScope.

ACTIVITY #1: ASCII CODES

ASCII stands for *American Standard Code for Information Exchange*, and it uses numeric codes to represent US alphabet characters. It also includes some special codes called *control characters* for keys on your keyboard like 'Esc' and 'Backspace.' When the BASIC Stamp sends messages to display as text on by a PC or serial LCD, it uses ASCII codes to send the characters that make up the message.

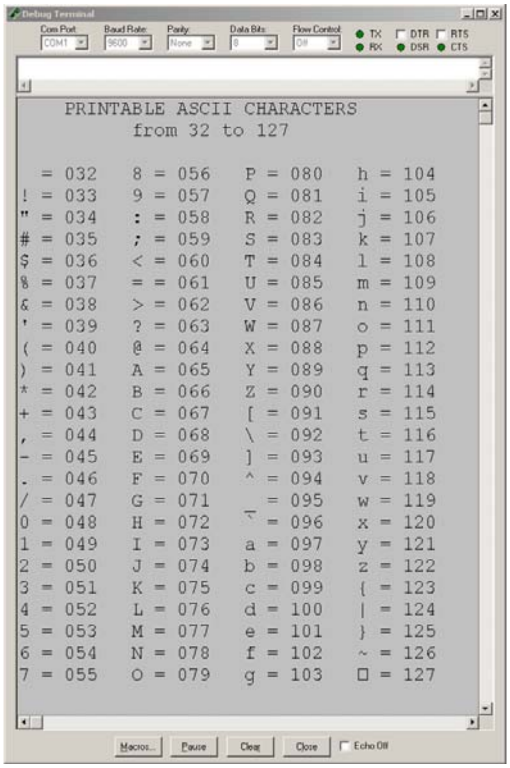
Printable ASCII Chart.bs2 displays characters and their corresponding ASCII codes for values of 32 through 127 in the Debug Terminal.

- ✓ Enter and run Printable ASCII Chart.bs2

```
' Printable ASCII Chart.bs2
' Display 32 through 127 along with the ASCII characters those
' values represent.

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

char           VAR     Word                  ' Will store ASCII codes

PAUSE 1000                                   ' 1 second delay before messages

DEBUG CLS, "    PRINTABLE ASCII CHARACTERS", ' Display chart heading
       CR, "          from 32 to 127"

FOR char = 32 TO 127                         ' Character=ASCII value loop
  DEBUG CRSRXY, char-32/24*10, char-32//24+3 ' Position cursor row, column
  DEBUG char, " = ", DEC3 char               ' Display Character=ASCII value
NEXT
```

The Debug Terminal display should resemble Figure 6-2.

- ✓ If the display gets scrambled because there's not enough room in the Debug Terminal, make the window larger and restart the program by pressing and releasing the Reset button on your board.



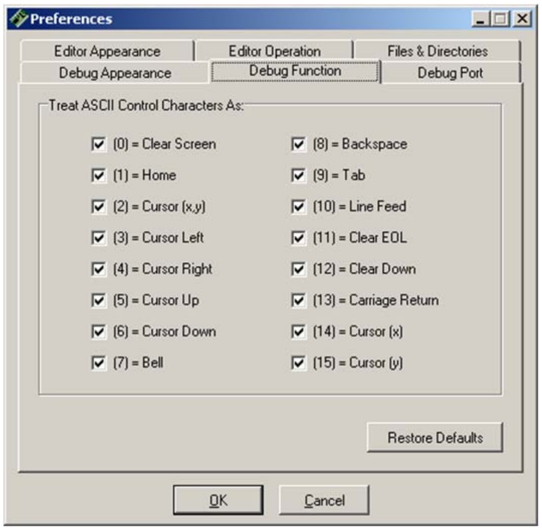
Com Port: COM1 Baud Rate: 9600 Parity: None Data Bits: 8 Flow Control: Off

```
    PRINTABLE ASCII CHARACTERS
          from 32 to 127

  = 032   8 = 056   P = 080   h = 104
! = 033   9 = 057   Q = 081   i = 105
" = 034   : = 058   R = 082   j = 106
# = 035   ; = 059   S = 083   k = 107
$ = 036   < = 060   T = 084   l = 108
% = 037   = = 061   U = 085   m = 109
& = 038   > = 062   V = 086   n = 110
' = 039   ? = 063   W = 087   o = 111
( = 040   @ = 064   X = 088   p = 112
) = 041   A = 065   Y = 089   q = 113
* = 042   B = 066   Z = 090   r = 114
+ = 043   C = 067   [ = 091   s = 115
, = 044   D = 068   \ = 092   t = 116
- = 045   E = 069   ] = 093   u = 117
. = 046   F = 070   ^ = 094   v = 118
/ = 047   G = 071   _ = 095   w = 119
0 = 048   H = 072   ` = 096   x = 120
1 = 049   I = 073   a = 097   y = 121
2 = 050   J = 074   b = 098   z = 122
3 = 051   K = 075   c = 099   { = 123
4 = 052   L = 076   d = 100   | = 124
5 = 053   M = 077   e = 101   } = 125
6 = 054   N = 078   f = 102   ~ = 126
7 = 055   O = 079   g = 103   □ = 127
```

Figure 6-2
Characters and Decimal Value ASCII Codes Displayed in Debug Terminal

Figure 6-3 shows ASCII values the Debug Terminal uses as control characters for operations like Clear Screen, Home, Backspace, Tab, Carriage Return and others.

- ✓ In the BASIC Stamp Editor, click Edit and select Preferences. Then, click the Debug Function tab.



Treat ASCII Control Characters As:

☑ (0) = Clear Screen
☑ (1) = Home
☑ (2) = Cursor (x,y)
☑ (3) = Cursor Left
☑ (4) = Cursor Right
☑ (5) = Cursor Up
☑ (6) = Cursor Down
☑ (7) = Bell
☑ (8) = Backspace
☑ (9) = Tab
☑ (10) = Line Feed
☑ (11) = Clear EOL
☑ (12) = Clear Down
☑ (13) = Carriage Return
☑ (14) = Cursor (x)
☑ (15) = Cursor (y)

Figure 6-3
Debug Terminal Control Character Values and Functions

All of the control characters listed in Figure 6-3 have symbol names in the PBASIC language. For example, instead of **`DEBUG 0`** for Clear Screen, you can use **`DEBUG CLS`**. In place of **`DEBUG 13`** for Carriage Return, you can use **`DEBUG CR`**. The **`DEBUG`** command documentation in the BASIC Stamp Editor Help's PBASIC Language Reference has a complete list of PBASIC control character symbol names, descriptions of their functions, and the values they represent.

✓ Close the Preferences window and open the BASIC Stamp Editor's Help. (Click Help and select BASIC Stamp Editor Help…)

✓ In the PBASIC Language Reference, follow the link to the **`DEBUG`** command documentation.

✓ Scroll down to the last table. It lists the names and ASCII values of the functions in Figure 6-3 along with PBASIC Symbol names you can use as **`DEBUG`** command arguments, like **`CLS`**, **`CR`**, **`HOME`**, etc.

Your Turn: Display “A” to “Z”, “a” to “z”, and 128 to 255

Printable ASCII Chart.bs2 uses the **char** variable in a **FOR…NEXT** loop to count from 32 TO 127. Each time through the loop, the **DEBUG** command transmits the **char** variable’s value to the PC using asynchronous serial communication. The Debug Terminal then displays the ASCII value’s corresponding character. The PBASIC language treats characters in quotes as their ASCII value equivalents, so you could actually create a **FOR…NEXT** loop that counts from "A" to "Z", "a" to "z", or even "!" to ".".

- ✓ In Printable ASCII Chart.bs2, try changing 32 TO 127 to "A" TO "Z".
- ✓ Load the modified program into the BASIC Stamp, and observe the results.
- ✓ Repeat for "a" TO "z" and "!" TO "." .

The Debug Terminal has more characters in the 128 to 255 range that you can check too. These can be useful for displays in certain languages as well as symbols like degrees °, ±, ¼ and other symbols. For example, the **DEBUG** command to display 180° would be **DEBUG "180", 176**. The code 176 makes the Debug Terminal display the ° symbol.

- ✓ Try this modified **FOR…NEXT** loop in Printable ASCII Chart.bs2.

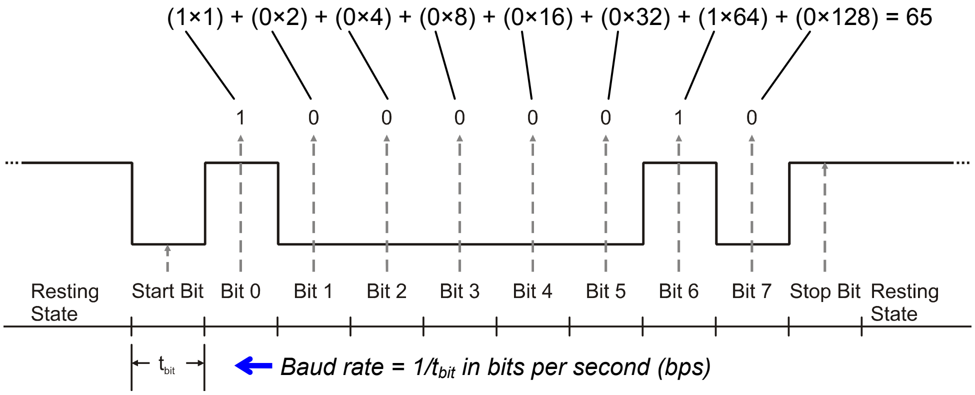
```
FOR char = 128 TO 255
  DEBUG CRSRXY, char-128/24*10, char-128//24+3
  DEBUG char, " = ", DEC3 char
NEXT
```

- ✓ Make a note of any character codes that might be useful for displays in future BASIC Stamp projects.

ACTIVITY #2: FIRST LOOK AT ASYNCHRONOUS SERIAL BYTES

Figure 6-4 shows an example timing diagram for the number 65— the letter "A"— transmitted with a widely used format of asynchronous serial signaling. The sending device that transmits this byte starts with a transition from high (Resting State) to low (Start Bit). Both of the devices use that as the starting point. The sending device uses it for updating the bit values it transmits at regular time intervals (t_{bit}), and the receiving device knows to check for new binary values at those intervals. The baud rate determines the time interval, which is t_{bit} = 1/baud rate. For example, if the baud rate is 9600 bits per second (bps), it means that each bit period has to be 1/9600th of a second. In other words, the bit time is $t_{bit} = 1 \div (9600 \text{ bits/second}) \approx 104.17\ \mu\text{s/bit}$. So, the transmitting device updates its binary values every 104.17 µs, and the receiving device checks in the middle of each of those time periods for the next binary value in the byte.

Figure 6-4: Number 65 Transmitted with 8-Bit, True, No Parity Asynchronous Serial Signaling



A device that transmits or receives the asynchronous serial signal in Figure 6-4 is using a format called “True signaling, 8 bits, no parity, and one stop bit.” The shorthand for this is 8N1, and it is usually preceded by a baud rate, like this: 9600 bps, 8N1, or just 9600 8N1. With *true* signaling, also called *non-inverted*, a high signal sends a binary-1, and a low signal sends a binary-0. *8 bits* means that the signal contains 8 binary values (bits). *No parity* indicates that this signal does not contain a *parity bit*, an option that the transmitter and receiver use to help detect communication errors. Parity bits will be examined in the Projects at the end of the chapter. One *stop bit* means that that the transmitter has to wait at least one bit period (t_{bit}) before sending another message. With 1 start bit, 8 data bits, and 1 stop bit, the total amount of time it takes to send/receive a single byte in this format is 10 bit periods.

Examining Figure 6-4 from left to right, the resting state of the signal is high. That signal could stay high for an indefinite amount of time if the device transmitting messages doesn’t have anything to send. When it does have something to send, it sends a low Start Bit signal for one bit period. Again, both transmitter and receiver use the negative edge of this signal for timing. The transmitter has to update its output between every bit period, and the receiver has to check for a value in the middle of each bit period. After the start bit, the transmitter sends the *least-significant bit* (Bit 0 or LSB), followed by Bit 1 during the next bit period, Bit 2 in the bit period after that, and so on, up through Bit

7 during the 8th bit period after the start bit. The receiver knows to treat Bit 0 as the number of 1s, in the value, Bit 1 as the number of 2s, Bit 2 as the number of 4s, and so on up through Bit 7, which is the number of 128s in the value. In Figure 6-4, Bit 0, which is the number of 1s is high, and so is Bit 6, which is the number of 64s. All the rest of the bits are low, so the value this signal transmits is (1 × 1) + (1 × 64) = 65.

In this activity, you will program the BASIC Stamp to use 9600 bps, 8N1 true asynchronous serial signaling to transmit the values in the previous activity's ASCII chart using an I/O pin. A new character will be transmitted once every second, and you will use the PropScope to monitor the sequence of ASCII values.

Asynchronous Serial Test Parts

(1) Resistor – 220 Ω (red-red-brown)
(misc.) Jumper wires

Asynchronous Serial Test Circuit

Figure 6-5 shows the test circuit and Figure 6-6 shows a wiring diagram example.

✓ Build the circuit in Figure 6-5 using Figure 6-6 as a guide.

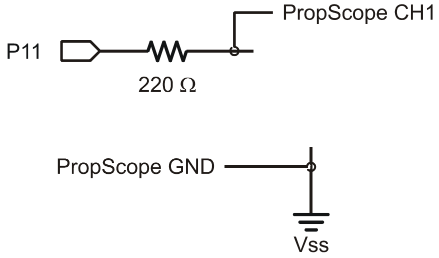


Figure 6-5
Test Circuit for Probing Asynchronous Serial Messages Transmitted by the BASIC Stamp I/O pin P11.

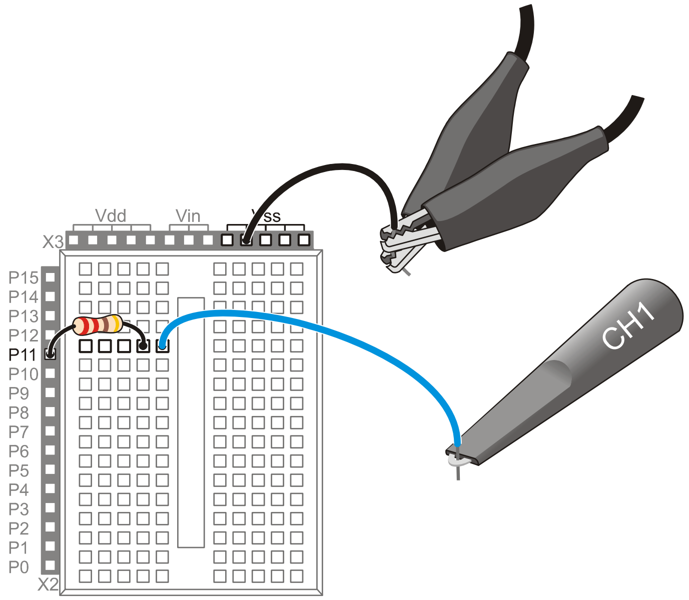


Figure 6-6
Wiring Diagram Example for Figure 6-5

Asynchronous Serial Test Code

Printable ASCII Chart to IO.bs2 sends a character from the ASCII chart to the Debug Terminal once every second. At about the same time, it sends a copy of that character to I/O pin P11 using 9600 8N1 true serial signaling.

✓ Open the BASIC Stamp Editor's Help, and look up the **`SEROUT`** command in the PBASIC Language Reference.
✓ Read the Syntax and Function sections.
✓ Find the Common Baud Rates and Corresponding Baud mode Values table for the BS2, and verify that 84 is the Baud mode in **`SEROUT 11, 84, [char]`** that will make it send its characters using true signaling at 9600, 8N1.
✓ Enter and run Printable ASCII Chart to IO.bs2.

```
' Printable ASCII Chart to IO.bs2
' Display another value in the ASCII character chart once every second.
' Transmit a copy of that value to P11 using 9600 bps 8N1 with true signaling.

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

```
char           VAR    Word                   ' For counting and storing ASCII

PAUSE 1000                                   ' 1 second delay before messages

DO                                           ' Main loop

  DEBUG CLS,
         "    PRINTABLE ASCII CHARACTERS",   ' Table heading
         CR, "          from 32 to 127"

  FOR char = 32 TO 127                         ' Count from 32 to 127
    DEBUG CRSRXY, char-32/24*10, char-32//24+3 ' Position cursor
    SEROUT 11, 84, [char]                    ' Send 9600 bps, 8N1 true byte
    DEBUG char, " = ", DEC3 char             ' Display byte in Debug Terminal
    PAUSE 1000                               ' 1 second delay
  NEXT                                       ' Repeat FOR...NEXT loop

LOOP                                         ' Repeat main loop
```

Asynchronous Serial Test Measurements

Figure 6-7 shows an example of the "A" character, which will display for one second, about 34 seconds into the program. (That's 33 seconds worth or characters and the 1 second **PAUSE** command at the very beginning.) Keep in mind that the signaling shown by the PropScope will change once every second as the BASIC Stamp transmits a new ASCII value. Also, notice that the Trigger in Figure 6-7 has been set to Fall since the message begins with a negative transition from high resting state to low start bit.

- ✓ Drag the CH1 trace downward so that 0 V is only slightly above the time scale. (In Figure 6-7, the 0 V ground line is 1 voltage division above the time scale.)
- ✓ Configure the PropScope's Horizontal, Vertical and Trigger controls according to Figure 6-7.
- ✓ Set the Trigger Voltage control to about 2.5 V, and the Trigger Time control to the second time division.
- ✓ Watch the counting pattern in the Oscilloscope, and use the Debug Terminal as a reference for seeing how the ASCII codes correspond to the signals in the Oscilloscope.
- ✓ Verify that the pattern displayed by the PropScope changes once every second.

Figure 6-7: ASCII 65, Capital "A"

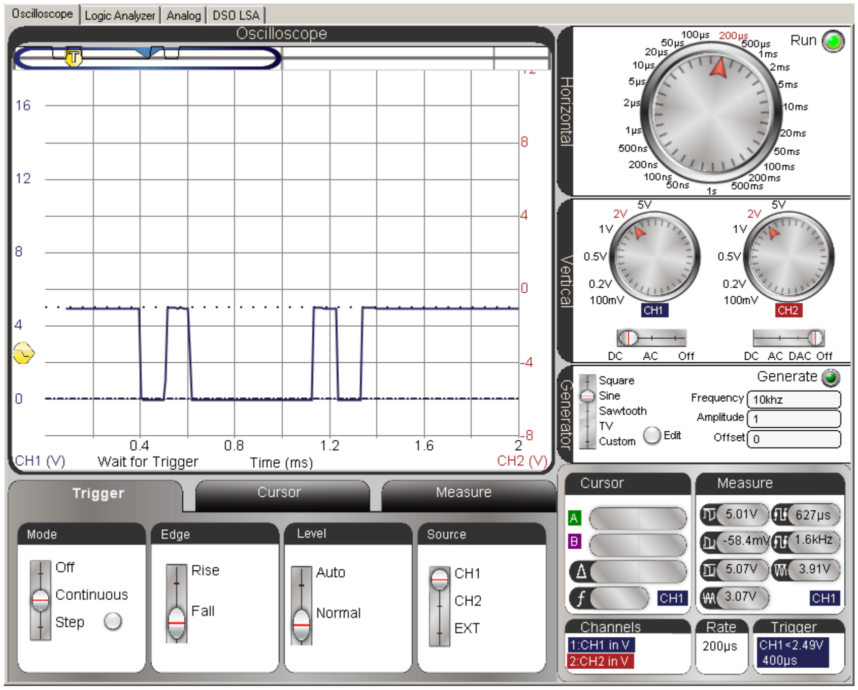


Figure 6-8 shows samples of the binary patterns for the characters "A" through "E" which correspond to ASCII codes 65 through 69. This is a portion of the sequence of asynchronous serial bytes the PropScope should display while Printable ASCII Chart to IO.bs2 is running.

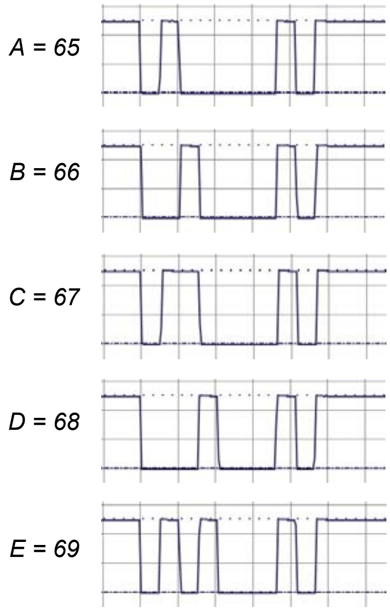


Figure 6-8
Byte Values 65 to 96

These are the ASCII codes for the characters "A" through "E" transmitted at 9600 bps with 8N1 true signaling.

In the next activity, you will examine these signals bit-by-bit to determine the ASCII values.

Your Turn: DEBUG vs. SEROUT

`DEBUG` is a special case of the **`SEROUT`** command. It's **`SEROUT 16, 84, [`*`arguments...`*`]`**. For example, in Printable ASCII Character Chart to IO.bs2, you can replace **`DEBUG char, " = ", DEC3 char`** with **`SEROUT 16, 84, [char, " = ", DEC3 char]`**, and the Debug Terminal will behave exactly the same.

- ✓ Try it.

ACTIVITY #3: A CLOSER LOOK AT A SERIAL BYTE

In this activity, you will program the BASIC Stamp to send the letter "A" (ASCII 65) using 9600 bps, 8N1, true asynchronous serial signaling and decode its value with the PropScope.

Letter A Test Code

Letter A to P11.bs2 transmits the "A" character via I/O pin P11 once every second, and it also sends it to the Debug Terminal to verify that characters are still getting sent once every second. The Debug Terminal verification can be useful for situations when you're not sure if the Oscilloscope display should be updating or not.

✓ Enter and run Letter A to P11.bs2.

```
' Letter A to P11.bs2
' Transmit "A" to P11 and Debug Terminal once every second.

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

PAUSE 1000                                 ' 1 second delay before messages

DO                                         ' Main loop

  SEROUT 11, 84, ["A"]                     ' "A" to P11
  DEBUG "A", " ", DEC3 "A", " ",           ' "A" to Debug Terminal
  BIN8 "A", CR
  PAUSE 1000                               ' 1 second delay

LOOP                                       ' Repeat main loop
```

Letter A Test Measurements

Figure 6-9 shows the letter "A" again. It's important to verify that your display shows all the transitions and states in the Oscilloscope screen before continuing.

✓ Verify your PropScope's Horizontal, Vertical, and Trigger settings against Figure 6-9.

✓ Verify that your Trigger Voltage control is at about 2.5 V and that the Trigger Time control is at the 2nd time division (0.4 ms)

✓ Make sure you've got a good view of this on your PropScope.

The time per division options on the Horizontal dial are not the only options, and this is a case where a custom time per division could come in really handy. Remember from Activity #2 that if the baud rate is 9600 bits per second (bps), then the bit time is $t_{bit} = 1 \div (9600 \text{ bits/second}) \approx 104.17\ \mu\text{s/bit}$. Wouldn't it be nice is if the Oscilloscope had a 104 µs units per division setting? Try this:

- ✓ Right click the Oscilloscope screen. The “Click on the Value you wish to change” window in Figure 6-10 should appear.
- ✓ Shade the value 200 in the (timescale, Value) cell and change it to 104, press the Enter key, and close the window. This will change the time scale value from the Horizontal dial’s 200 µs/division setting to 104 µs/division.

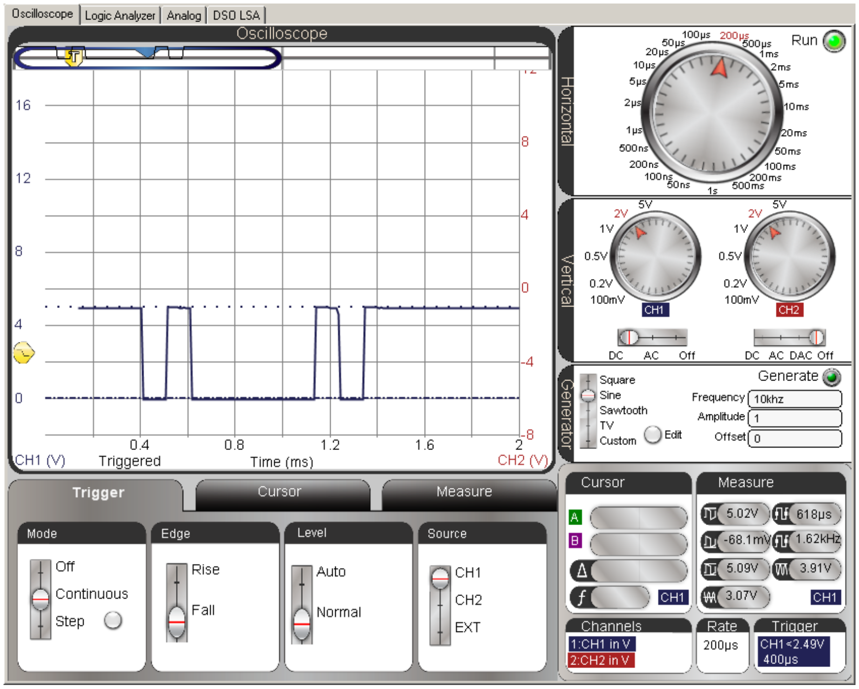
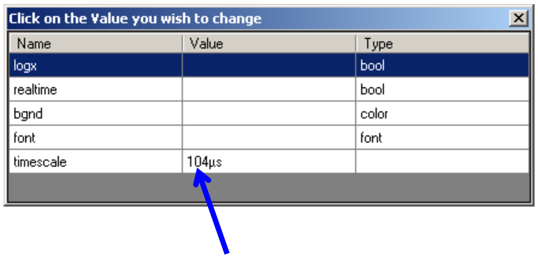


Figure 6-9
The Letter "A" Again

Time per division is 200 µs.


Click on the Value you wish to change

Name	Value	Type
logx		bool
realtime		bool
bgnd		color
font		font
timescale	104µs	

Shade the number 200 and change it to 104

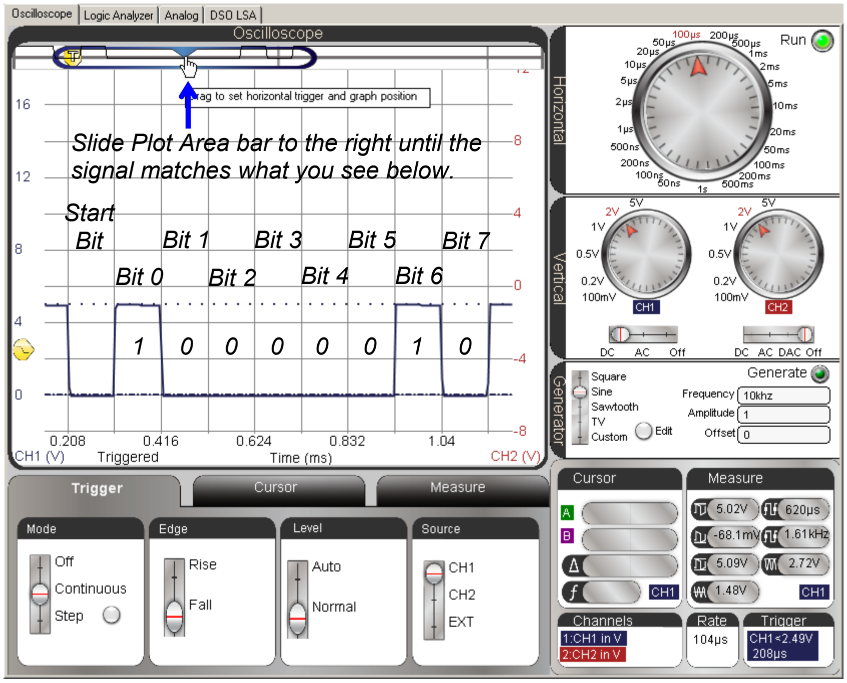
Figure 6-10
Configuring Custom Time per Division

Your Oscilloscope will need a few more adjustments before it resembles Figure 6-11.

- ✓ Adjust the Plot Area bar slightly to the right so that you can view all the bits of the asynchronous serial 65 byte as shown in Figure 6-11. The falling edge of the start bit should align with the first visible time division line.
- ✓ Compare your results to the timing diagram in Figure 6-4 on page 179.

The important feature here is that each bit in the asynchronous serial byte now occupies a single time division. This makes it much easier to translate an asynchronous serial byte displayed on the Oscilloscope screen into the value that's being transmitted. Remember that Bit 0 is the number of 1s in the number, Bit 1 is the number of 2s, Bit 3 is the number of 4s, and so on up through Bit 7, which is the number of 128^{ths}.

Figure 6-11: 9600 bps Byte Value = 65 Viewed with Timescale set to 104 µs/Division



Slide Plot Area bar to the right until the signal matches what you see below.

Bits in a Byte: The values in Figure 6-11 get stored in a byte so that it looks like this: %01000001. The % sign is a PBASIC formatter that tells the BASIC Stamp Editor that it's a binary number. In this binary number, Bit 0 gets stored in the rightmost position, Bit 1 in the next position to the left, and so on up through Bit 7, which is the leftmost digit. It's the opposite of the order from the binary digits get transmitted in an asynchronous serial byte.

Powers of 2 in a Byte: The value stored by Bit 0 determines the number of 1s in the variable, and 2^0 = 1. The value stored by Bit 1 determines the number of 2s in the variable, and 2^1= 2. The value in Bit 3 determines the number of 4s in the variable, and 2^2=4. More generally: Each bit determines whether a binary number has (1 or 0) × $2^{\text{bit position}}$.

Your Turn: Pick a Byte Value

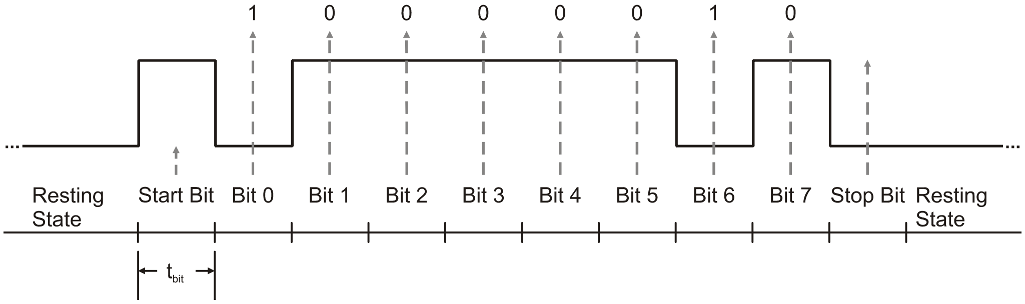
✓ Repeat the measurements in this activity with a character or byte value of your choosing.

ACTIVITY #4: TRUE VS. INVERTED

When the **DEBUG** command sends a message to the PC, the true signal is inverted by a circuit so that every high portion of the signal becomes low, and every low portion becomes high. You will take some measurements of the inverted signals going to and coming from the PC in Activity #6. Figure 6-12 shows an example of the same 65 value in an inverted signal. In this activity, you will program the BASIC Stamp to send the letter "A" with both true and inverted asynchronous serial signaling and compare the results with the PropScope.

✓ Compare the signaling in Figure 6-12 against Figure 6-4 on page 179.

Figure 6-12: Timing Diagram of 65 Transmitted with Inverted Signaling



True vs. Inverted Comparison Test Parts

(2) Resistors – 220 Ω (red-red-brown)
(misc.) Jumper wires

True vs. Inverted Comparison Test Circuit

The **SEROUT** command can be configured to send a true signal followed by an inverted signal, one after the other on the same line. However, that introduces some oddities to the signal that make it difficult for a device to receive and decipher. So, for a comparison of true and inverted signals, the BASIC Stamp will be programmed to send the signals on separate I/O pins. The PropScope will be used to probe each signal to display on its own channel.

✓ Connect the probes to the I/O socket through the 220 Ω resistors shown in Figure 6-13 using Figure 6-14 as a guide.

True vs. Inverted Comparison Test Code

True Inverted Comparison.bs2 sends the same "A" character with true signaling we've been measuring with CH1 on I/O pin P11. Then, it sends the "A" character with inverted signaling on I/O pin P10, and this will be measured with CH2.

✓ Enter and run True Inverted Comparison.bs2.

```
' True Inverted Comparison.bs2
' Transmit "A" on P11 with true signaling, and then on P10 with inverted.

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

char           VAR     Word                  ' Will store ASCII codes

PAUSE 1000                                   ' 1 second delay before messages

DO                                           ' Main loop

  SEROUT 11, 84, ["A"]                       ' Send 9600 bps, 8N1 true byte
  SEROUT 10, 16468, ["A"]                    ' 9600 bps, 8N1 inverted byte
  DEBUG "A"                                  ' "A" to Debug Terminal
  PAUSE 1000                                 ' 1 second delay

LOOP                                         ' Repeat main loop
```

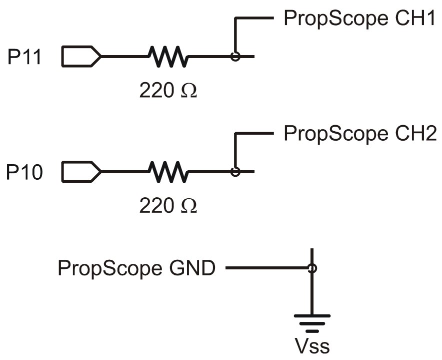


Figure 6-13
Both Channel Probes Connected to I/O Pin Sockets through Protection Resistors

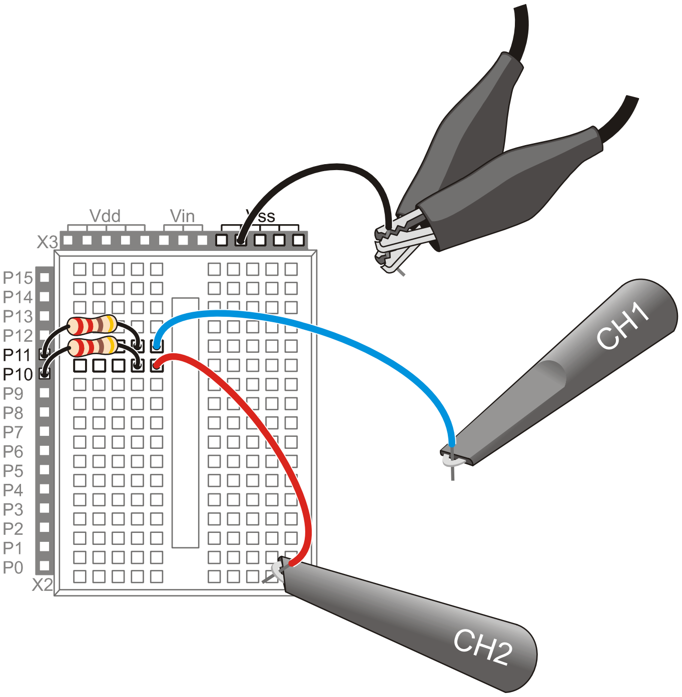


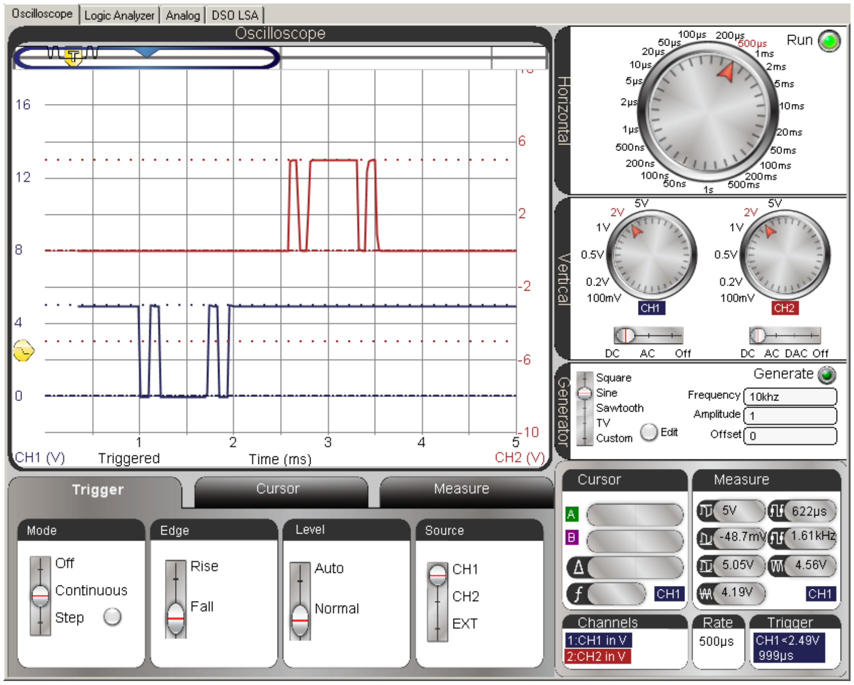
Figure 6-14
Wiring Diagram Example of Figure 6-13

True vs. Inverted Comparison Test Measurements

Figure 6-15 shows the true signal on the lower CH1 trace, and the inverted signal on the upper CH2 trace. Note that the inverted signal is simply the opposite level from the true signal for all bits and resting states.

- ✓ Slide the Vertical CH2 coupling switch from Off to DC.
- ✓ Click, hold, and drag the red CH2 trace and position it so that its 0 V ground line is slightly above the top of the CH1 trace. (In Figure 6-15, it's 1.5 voltage divisions above the top of the CH1 trace.)
- ✓ Adjust the Horizontal dial to 500 ms/division.
- ✓ Slide the Plot Area bar to the far left.
- ✓ If needed, adjust the Trigger Time control so that it aligns the CH1 signal's first negative edge with the second time division.

Figure 6-15: True Signal on CH1 (lower trace) Inverted on CH2 (upper trace)



Frequency 10khz
Amplitude 1
Offset 0

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 500µs
Trigger: CH1<2.49V 999µs

Your Turn: Trigger on CH2; Find the Missing CH1 Signal

Let’s say you want to take a closer look at the CH2 signal. One approach would be to set the Trigger Source to CH2 and the Trigger Edge to Rise. This will align the rising edge of the CH2 signal with the second time division, but the CH1 signal will mostly if not entirely disappear. So where did it go?

To answer this question, you can scroll the Plot Area bar to the center of the Plot Preview, and then move the Trigger Time control to adjust the signal positions. If you drag it to the approximate center of the Plot Area bar, the signals should start to resemble Figure 6-15 again. You can also set the units per division to 104 µs. With adjustments to some combination of the Trigger Time control, Plot Area bar, Trigger Edge, and Trigger Source, you can bring either signal into a close-up view to examine its bits. Remember that with the inverted signal, a low is a binary 1, and a high is a binary 0.

- ✓ Set your display up so that it triggers off the CH2 trace’s positive edge, and use the Plot Area bar and Trigger Time control to adjust the horizontal positioning of the signals in the Oscilloscope screen.
- ✓ Optional: Set the Horizontal timescale to 104 µs/div. See if you can use the Plot Area bar and Trigger Time control to navigate between the CH1 and CH2 bites.

ACTIVITY #5: HARDWARE FLOW CONTROL

Hardware flow control can be an exceedingly useful feature. For example, what if two BASIC Stamp modules are connected together and communicating with asynchronous serial communication? Robot applications sometimes use a pair of BASIC Stamp modules that exchange data this way. Without flow control, when one BASIC Stamp is busy with other tasks like servo control and sensor monitoring, it could miss serial bytes from the other BASIC Stamp. With hardware flow control, that BASIC Stamp can send a high signal on a separate line to let the other BASIC Stamp know that it’s busy. When it’s ready for serial messages, it can send a low signal. The other BASIC Stamp stops sending bytes when it receives a high (busy) signal, and resumes when it receives a low (ready) signal.

> **Systems can also use software flow control.** With software flow control, a receiver sends sequences of serial bytes to tell the transmitter when it is ready or busy. This approach is common in systems that automatically buffer (store in memory) certain numbers of bytes and examine portions of them between tasks. With BASIC Stamp 2 modules, software flow control might save a few I/O pins, but it tends to take more code and memory to implement.

In this activity, the PropScope will stand in for the BASIC Stamp that receives messages with flow control. It might be busy sometimes (sending a high signal) and ready to receive serial messages other times (sending a low signal). Your actual BASIC Stamp will be the one sending messages to the PropScope. It will be programmed to send serial bytes with hardware flow control enabled. The program will make your BASIC Stamp send serial bytes using an I/O pin connected to the PropScope's CH1 probe. Your BASIC Stamp module's program will also monitor another I/O pin for hardware flow control high/low signals that the PropScope's function generator will transmit. The function generator (which is emulating a BASIC Stamp receiving serial messages with flow control) will transmit a square wave. When the square wave's signal is high (busy), it will cause your BASIC Stamp to wait. When the signal is low (not busy) it will cause your BASIC Stamp to resume transmitting bytes.

The Oscilloscope will display the function generator's high/low signals with its CH2 trace, and its CH1 trace will indicate when serial messages are or are not transmitted. With this approach, you will be able to "see" how flow control works, with high signals on CH2 interrupting serial activity, and low signals allowing it to resume.

Flow Control Test Code

In Test Flow Control.bs2, the command **`SEROUT 11\10, 84, [REP "A"\254]`** uses P11 to transmit 254 "A" bytes in rapid succession and monitors P10 for flow control signals. The PropScope's function generator output sends a square wave flow control signal to P10. During the time the square wave signal is high, it interrupts the "flow" of serial bytes from the BASIC Stamp. When the signal is low, it allows the BASIC Stamp to transmit serial bytes.

Keep in mind that P10 was used as an output in the previous activity. Before connecting the PropScope's function generator output to it, you should make sure to run code that sets the I/O pin to input. Even though there is a 220 Ω resistor protecting the I/O pin from the function generator output and vice-versa, it's still a better practice to use code to prevent I/O conflicts along with the resistors that protect your hardware from possible coding mistakes.

Any PBASIC program starts all I/O pins as inputs, but certain commands can change them to output. Examples include **`HIGH`**, **`LOW`**, and **`PULSOUT`**. In Test Flow Control.bs2, the command **`SEROUT 11\10…`** sets P11 to output to make it transmit serial bytes, and the optional **`\10`** flow control pin argument sets P10 to input to make it monitor flow control signals. So, by loading Test Flow Control.bs2 into your BASIC Stamp *before* connecting the function generator output to P10, you are taking an additional step to protect the function generator output and BASIC Stamp I/O pin.

- ✓ Enter Test Flow Control.bs2 into the BASIC Stamp Editor and run it.

```
' Test Flow Control.bs2
' Main loop repeatedly sends 254 copies of "A" to P11, with P10 flow control.

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

PAUSE 1000                                   ' 1 second delay before messages

DO                                           ' Main loop

  SEROUT 11\10, 84, [REP "A"\254]            ' P11 transmits, P10 flow control
  DEBUG "254 bytes sent", CR                 ' Debug Terminal message

LOOP                                         ' Repeat main loop
```

Flow Control Test Circuit

To test flow control, the DAC Card's function generator output needs to be connected to the P10 input. Figure 6-16 shows the schematic, which is almost identical to the one from the previous activity, and Figure 6-17 shows a wiring diagram example. The only difference is that the probe has to be disconnected from the PropScope's CH2 BNC connector, and connected to DAC Card's function generator output BNC connector.

- ✓ Disconnect the CH2 probe from the PropScope's CH2 BNC connector and connect it to the DAC Card's function generator output. For a refresher on which one is the function generator output, see Figure 2-16 on page 46.

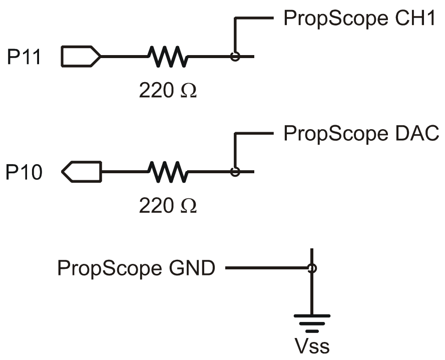


Figure 6-16
CH1 Probes P11 Serial Output, DAC Output Drives P10 Flow Control Input Line

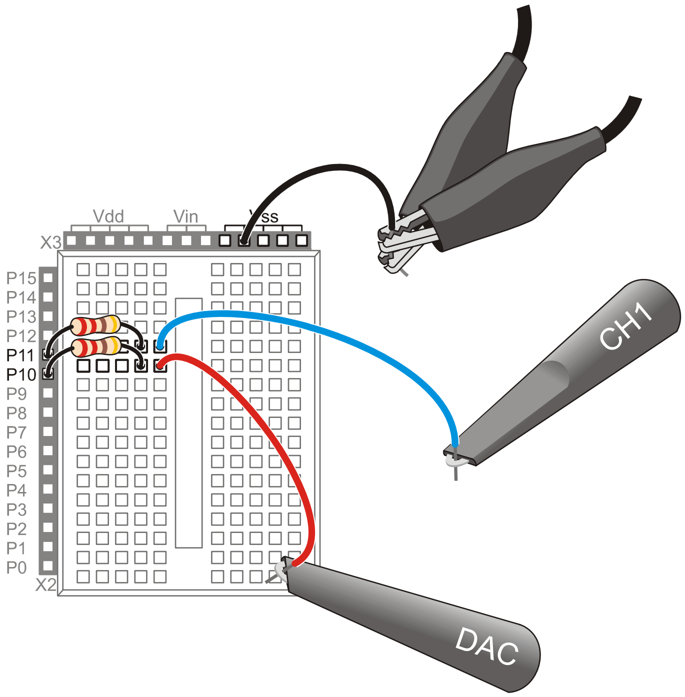


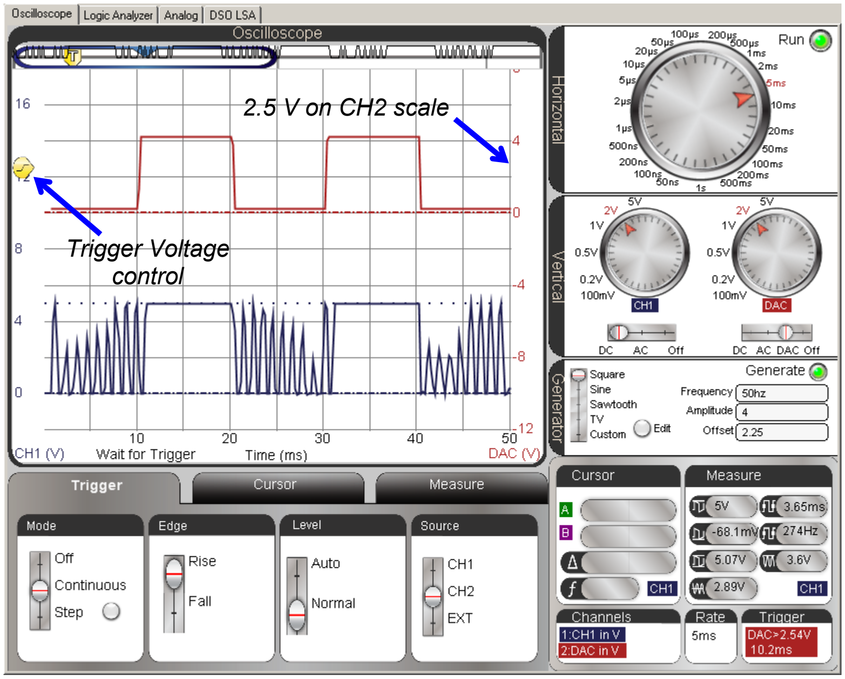
Figure 6-17
Wiring Diagram Example of Figure 6-16

Flow Control Test Measurements

The upper DAC trace in Figure 6-18 shows the PropScope DAC Card's function generator sending a 50 Hz square wave. Since the function generator's output is connected to BASIC Stamp P10 I/O pin for flow control, the serial activity on the lower channel 1 trace stops during the square wave's high signals and resumes during low signals.

- ✓ Move the Plot Area bar to the far left of the Plot Preview.
- ✓ Set the Horizontal dial to 5 ms/div and the Vertical dials to 2 V/div.
- ✓ Set the Generator panel's Function switch to Square.
- ✓ Set the Generator panel's fields to: Frequency = 50 Hz, Offset = 2.25 V, Amplitude = 4 V, then click the Generate button.
- ✓ Set the Vertical CH2 coupling switch to DAC. (When you set the CH2 coupling switch to DAC, it uses the CH2 trace to display the function generator output.)
- ✓ In the Trigger tab, set these switches: Mode = Continuous, Edge = Rise, Level = Normal, and Source = CH2.
- ✓ Set the Trigger Time control so that the rising edge of the DAC output lines up with the second time division line.
- ✓ Set the Trigger Voltage control to about 2.5 V. Make sure it's 2.5 V on the CH2 scale (on the right side of the screen). See notes in Figure 6-18.
- ✓ Check your PropScope settings and display against Figure 6-18.

Figure 6-18: PropScope DAC Card Flow Control of Serial Byte Stream



2.5 V on CH2 scale

Trigger Voltage control

Frequency: 50hz
Amplitude: 4
Offset: 2.25

Channels: 1:CH1 in V, 2:DAC in V
Rate: 5ms
Trigger: DAC>2.54V 10.2ms

Measure: 5V, 3.65ms, -68.1mV, 274Hz, 5.07V, 3.6V, 2.89V, CH1

In Figure 6-18, the jagged spikes in the lower CH1 trace occur during function generator’s low signals in the upper CH2 trace. This indicates that a lot of activity is happening on channel 1 when the flow control signal from the function generator is low. The actual bit-level activity is not visible, but it’s a good indicator that serial communication occurs during those times. For a closer look, you can pick a function generator frequency that lets fewer bytes through and then adjust the Horizontal timescale to a smaller value. For example, you could pick a function generator frequency that only lets two bytes through at a time.

For two bytes at a time, you’ll need to set the function generator in terms of some frequency. To figure out the frequency, the first step is to figure out how much low time is needed for two bytes. Next, multiply that time by 2 since a square wave has equal high and low times. The result is the cycle time or period. Then, take the reciprocal of the period for the frequency. That’s your frequency for the function generator.

Here is how to calculate the function generator frequency for letting two bytes through assuming the baud rate is 9600 bps. Since each byte has ten bit times, the time for transmitting one byte would be 10 × t_{bit} ≈ 10 × 104 µs = 1040 µs = 1.04 ms. For two bytes, that's 2.08 ms, so our square wave needs about 2 ms of low time. With a square wave, there will also be 2 ms of high time for a total cycle time of 4 ms. Since frequency is the reciprocal of period, f = 1/T = 1 ÷ 4 ms = 1 ÷ 0.004 s = 250 Hz. So, the function generator should be configured to send a 250 Hz square wave.

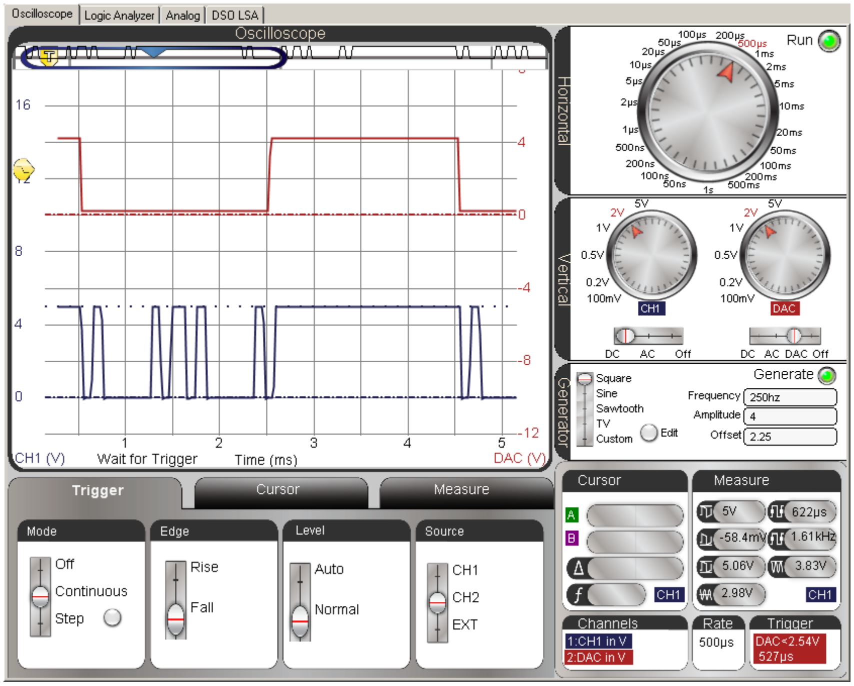
For a Horizontal timescale, we are really interested in the two bytes that get transmitted during the high time, not two cycles of the function generator signal. We also want to see that the bytes stop transmitting during the function generator's high signal. So, only one cycle of the function generator square wave needs to be visible in the Oscilloscope screen.

Since we only want to display one function generator cycle, that's a total of 0.004 s = 4 ms. Remember that to get the Horizontal dial setting for viewing, you have to divide the total amount of time you want to view in the plot area by 10 divisions to get the Horizontal dial's time per division setting. That's 4 ms ÷ 10 = 400 µs, which is close to the Horizontal dial's 500 µs setting, so we'll use that. Figure 6-19 shows the result. Two "A" bytes make their way through during each low cycle of the DAC Card's function generator's Square Wave signal.

✓ Adjust the Horizontal dial to 500 µs/division.

✓ Change the Trigger Edge to Fall, and adjust the Trigger Time control so that the DAC trace's negative edge lines up with the first time division line.

Figure 6-19: Two "A" Bytes Per Cycle



Frequency 250hz
Amplitude 4
Offset 2.25

Measure: 5V, 622µs, -58.4mV, 1.61kHz, 5.06V, 3.83V, 2.98V, CH1

Channels: 1:CH1 in V, 2:DAC in V

Rate: 500µs

Trigger: DAC<2.54V 527µs

Your Turn: Does Flow Control Stop at the Bit or the Byte?

When the flow control signal stops the BASIC Stamp from sending data, does the BASIC Stamp stop with the next bit of data to be sent, or does it finish sending the current byte before it stops? It is difficult to tell from Figure 6-19. If the transition from low to high happened 500 µs sooner, it would be more obvious because the flow control signal would transition to high in the middle of a byte. Then, you'd be able to tell if the flow control terminates the signal in the middle of a byte or not.

- ✓ Repeat the period to frequency calculations discussed earlier, but with high and low times that are 250 µs shorter.
- ✓ Adjust the function generator frequency.
- ✓ So, does flow control interrupt **SEROUT** transmit at the bit or the byte?

ACTIVITY #6: PROBE THE SERIAL PORT

In this activity, you will use one of the probes to test the signals a BASIC Stamp transmits to and receives from the PC via the serial port. It doesn't matter if your board is USB, the BASIC Stamp still uses serial signaling to exchange data with the USB board's serial/USB converter chip, and you can measure those signals.

PC serial ports typically send serial signals with peak-to-peak voltages that are outside the 0 to 5 V range that BASIC Stamp I/O pins transmit. These ports adhere to a standard called RS232, which allows for high signals in the +25 to + 3 V range, and low signals in the -3 to -25 V range. RS232 driver chips in computer serial ports typically get their signal voltages from the computer's power supply. So, a serial driver chip's high and low voltages will typically be the same as a pair of its power supply voltages, commonly called *power supply rails* or just *supply rails*. Supply rail values of ± 5 V and ± 12 V are common, but they do vary from one system to the next. For example, one system might have a pair of rails at ± 12 V and another at ± 5 V, while a different system might have a pair of rails at ± 5 V and another at ± 3.3 V.

Probe Setup

With probes set to 1x, the maximum voltage range the PropScope can measure is ± 10 V. Your computer's serial port voltages might be outside that range, so the best thing to do is adjust your probes and PropScope software to the 10x setting. Then, the PropScope will be able to measure up to ± 100 V instead of just ± 10 V. So, if your computer's serial port uses signal voltages outside ± 10 V, you'll still be able to measure the values.

One-Time 10x Probe Calibration

The PropScope Probe Kit came with instructions for a one-time probe calibration that should be done before taking measurements with the probes set to 10x. While monitoring a 1 kHz square wave trace with a probe set to 10x, a screwdriver that also came with the probe kit is used to adjust a potentiometer inside the BNC connector. The goal of the adjustment is to make the measured square wave in the CH1 trace as square as the wave it receives. This compensates for electrical interactions between the 10x probe circuit and the Oscilloscope input circuit. The source of the 1 kHz square wave will be the PropScope's function generator, and the trace will be measured on CH1.

✓ Set the switch in the CH1 probe rod to X10, and update the PropScope software's Tools →Manage Probes settings so that the Probe 1 Gain is 10X.

- ✓ Connect your PropScope probes so that the function generator sends a signal to the CH1 probe. For this particular task, you do not need to connect clips to your board—just connect the ground clips to each other and the DAC Card's function generator output probe to the CH1 input probe. Another option is to follow the connection instructions in the Set DC Voltages with the PropScope's DAC Card section on page 45. Either way will work fine.
- ✓ Configure the PropScope:
  - o Trigger tab to Mode = Continuous, Edge = Rise, Level = Auto, and Source = CH1.
  - o Horizontal = 200 µs/div, Vertical = 1V/div for both CH1 and DAC.
  - o Function generator to transmit a square wave with: Frequency = 1 kHz, Amplitude = 3 V peak-to-peak (Vpp), and Offset = 0 V.

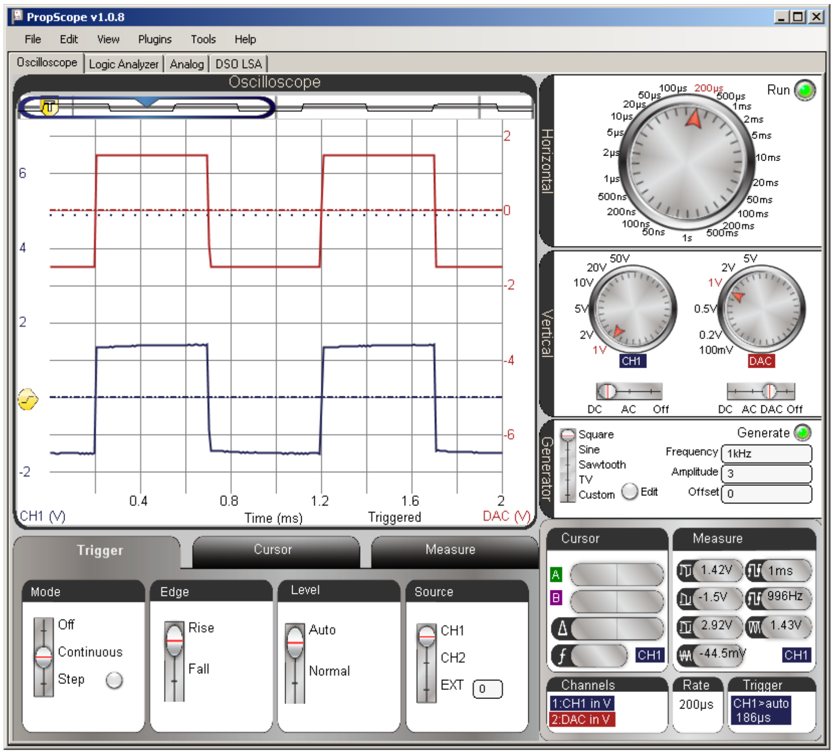


Figure 6-20
Measuring DAC Card function generator to CH1

With a probe set to 10x, a 1 kHz, 3 Vpp, 0 V offset square wave from the PropScope's function generator may result in a distorted square wave in the CH1 trace before adjustments.

Your lower CH1 trace may look misshapen compared to Figure 6-20. Figure 6-21 shows some examples of the waveform distortion you might see. If the tops and bottoms of the square wave in the CH1 trace are not flat, you'll need to adjust the trim potentiometer in the CH1 BNC connector.

✓ Using Figure 6-21 as a guide, adjust the trim potentiometer in the BNC connector of the probe connected to CH1 to correct the shape of the square wave displayed in the CH1 trace.

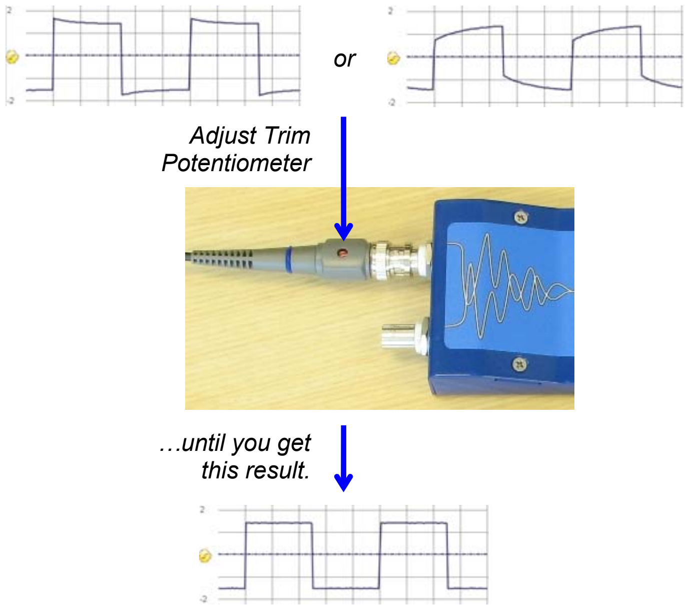


Figure 6-21
10X Probe Calibration

Adjust the trim pot in the CH1 BNC Connector to correct the shape of the waveform and the make the square wave square.

The CH1 probe adjustment procedure will have to be repeated for CH2. Since the PropScope cannot monitor CH2 while the function generator is running, you'll have to connect the probe you would normally connect to CH2 to CH1.

✓ Set the X1 X10 switch in the CH1 probe rod back to X1.
✓ Disconnect and swap the BNC connectors.

Now, the BNC connector with the red probe marker should be connected to CH1 and the one with the blue probe marker should be connected to the DAC Card's function generator output.

✓ Set the switch on the probe connected to the CH1 BNC connector to X10.
✓ Verify that the switch on the probe connected to the DAC Card's function generator BNC connector is set to X1.
✓ Perform the potentiometer adjustment shown in Figure 6-21 for the second probe.

Now that both probes have the proper trim settings, the probe with the blue markers should be reconnected to the PropScope's CH1 BNC port and the other one should be connected to the CH2 BNC port.

✓ Reconnect the blue-marker probe to the PropScope's CH1 BNC port.
✓ Reconnect the red-marker probe to the PropScope's CH2 BNC port.

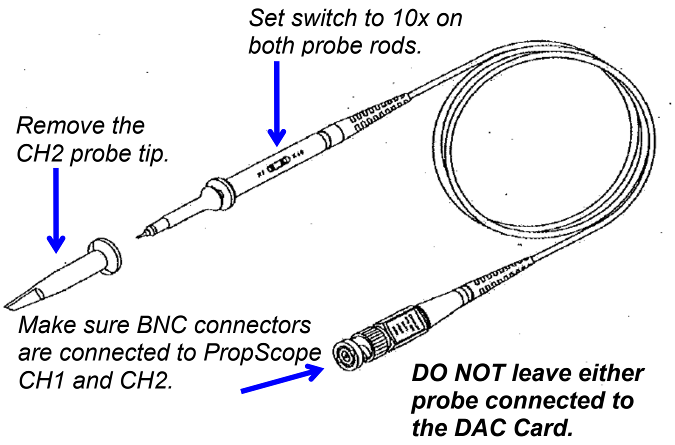
Configure Probes for RS232 Measurements

Remember that the RS232 voltages we'll be measuring might exceed the PropScope's 1X probe +/– 10 V measurement limits. Although it won't actually damage the probes or the PropScope CH1 or CH2 input ports, there won't be any way to tell if the PC is transmitting with +/– 12 V, for example. So the probe that will be used to measure the RS232 communication between the BASIC Stamp and PC should be set to 10x. In this section, both probes will be set to 10X for the sake of keeping the instructions simple and straightforward. It's also important to verify that both probes are connected to the PropScope's CH1 and CH2 inputs.

WARNING!

A probe that's inadvertently left connected to the DAC Card's function generator output but used to test a serial port could result in damaged equipment. Make sure both probes are connected only to CH1 and CH2 BNC connectors. DO NOT connect a probe to the DAC Card's BNC connectors.

✓ Make sure the Probes are connected to the CH1 and CH2 BNC connectors on the PropScope.
✓ **STOP and CHECK:**
→ No probes should be connected to any DAC Card BNC connector.
→ Probes should only be connected to the PropScope's CH1 and CH2 BNC connectors.
✓ Remove the CH2 probe's tip from the probe rod as shown in Figure 6-22.
✓ Set both probe rods' switches to X10.



Set switch to 10x on both probe rods.

Remove the CH2 probe tip.

Make sure BNC connectors are connected to PropScope CH1 and CH2.

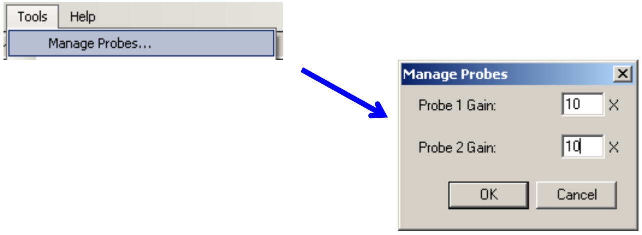
DO NOT leave either probe connected to the DAC Card.

Figure 6-22
X1 X10 Switch on the Probe Rod, Probe Tip, and BNC Connector

A 10x probe has a built-in voltage divider and a bypass capacitor that together minimize the probe circuit's electrical interactions with the test circuit. The signal that gets to the Oscilloscope's input circuit is 1/10th the voltage it would be if it was a 1x probe.

The PropScope software has to know that the signals it measures are coming from 10x probes. Reason being, these signals are 1/10th the value they would be if they were coming from a 1x probe. Figure 6-23 shows how to configure the PropScope software's probe settings to 10x.

- ✓ In the PropScope software, click Tools →Manage Probes. Set the probe gain to 10 for both probes.



Probe 1 Gain: 10 X

Probe 2 Gain: 10 X

Figure 6-23
Configure Probe Gain to 10x

Serial Port Transmit Test Code

PC Serial Transmit.bs2 sends "A" to P11 and another "A" to the BASIC Stamp SOUT pin with the **SEROUT** ***Pin*** argument of 16 in the second **SEROUT** command. This test code will make it convenient to compare the true signal transmitted by P11against the inverted signal transmitted by the SOUT pin.

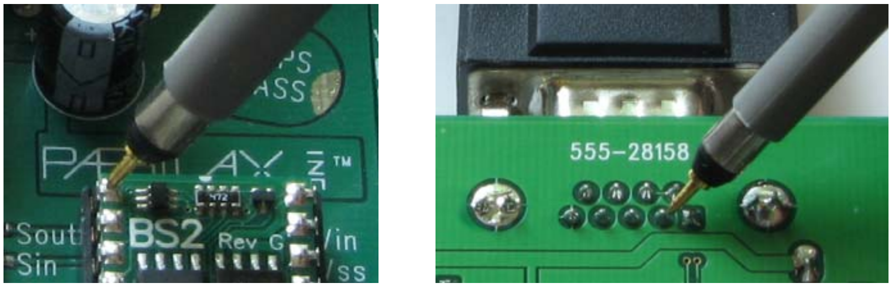
- ✓ Enter and run PC Serial Transmit.bs2.

```
' PC Serial Transmit.bs2
' Sends a true serial byte to P11 as well as to P16 (SOUT).
' The byte to SOUT gets inverted by a circuit before it is transmitted.
' {$STAMP BS2}                              ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                             ' Language = PBASIC 2.5

PAUSE 1000                                  ' 1 second delay before messages

DO                                          ' Main loop
  SEROUT 11, 84, ["A"]                      ' Send "A" to P11
  SEROUT 16, 84, ["A"]                      ' Send "A" to P16 (SOUT pin)
  PAUSE 1000                                ' 1 second delay
LOOP                                        ' Repeat main loop
```

Serial Port Transmit Test Measurements

Figure 6-24 shows how to probe the serial port with the CH2 probe while the program is running. The CH1 probe is still connected to P11, and the CH2 probe is connected to the signal that the BASIC Stamp sends, either to the PC's serial port or to a USB/serial converter chip.

Figure 6-24: CH2 Probe Position for SOUT Signal



Board of Education (All)

BASIC Stamp HomeWork Board Serial (Underneath the Serial Port Connecter)

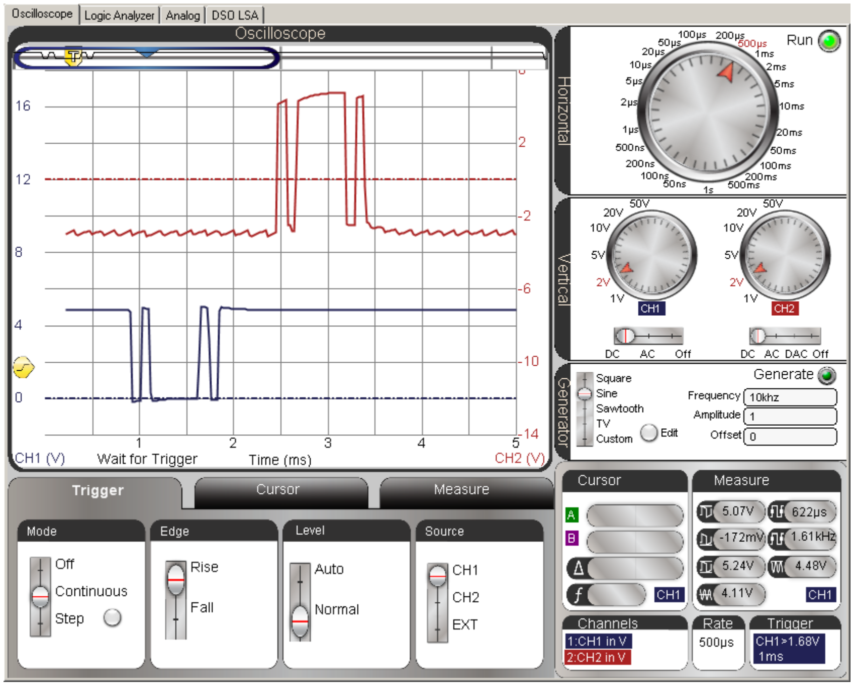
For other boards, get directions from the Supplement at www.parallax.com/go/propscope.

The lower CH1 trace in Figure 6-25 shows the P11 true signal, and the upper CH2 trace shows the signal the BASIC Stamp sends to the PC. The signal is inverted by a circuit built into the BASIC Stamp, and either the PC or USB/serial converter chip also has an inverter that converts this signal back to true for processing.

The inverted signals with higher voltage swings are part of standard computer serial port designs. Electromagnetic interference (EMI) generated by nearby motors and other electrical machines can cause voltage fluctuations in longer cable wires. So, the larger voltage swings in the signals are one way help ensure that the signals are still correct, even when a nearby electric motor is turned on or off.

- ✓ Set the PropScope controls as shown in Figure 6-25, and then use the CH2 probe to test the signal on the SOUT pin as shown in Figure 6-24.
- ✓ Make sure that the Trigger tab's Level switch is set to Normal and the Trigger Voltage control is set to about 2.5 V.

Figure 6-25: P11 Transmitted Byte on CH1 Followed by Inverted Byte on CH2



Generator: Square, Sine, Sawtooth, TV, Custom; Frequency 10khz; Amplitude 1; Offset 0

Trigger — Mode: Off, Continuous, Step; Edge: Rise, Fall; Level: Auto, Normal; Source: CH1, CH2, EXT

Measure: 5.07V, 622µs, -172mV, 1.61kHz, 5.24V, 4.48V, 4.11V, CH1, CH1

Channels: 1:CH1 in V, 2:CH2 in V; Rate: 500µs; Trigger: CH1>1.68V 1ms

Serial Port Receive Test Code

PC Serial Receive.bs2 receives a byte from the serial port and immediately sends a copy of whatever it receives on P11.

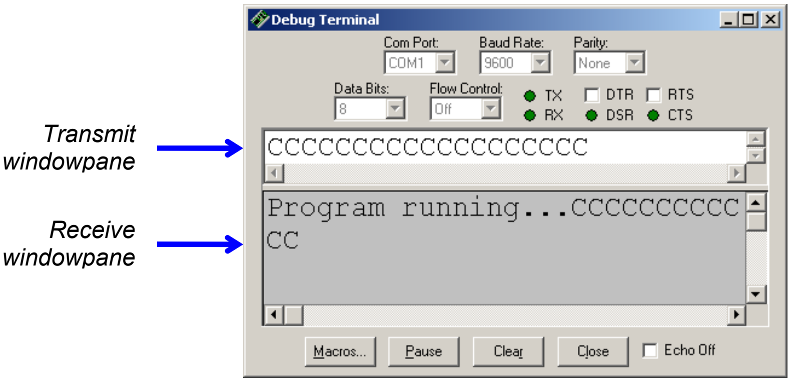
- ✓ Enter PC Serial Receive.bs2 into the BASIC Stamp Editor and then click Run to load it into the BASIC Stamp.

```
' PC Serial Receive.bs2
' Receives a byte from the PC's Debug Terminal and transmits it on P11.

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

char           VAR     Word                  ' For counting and storing ASCII

PAUSE 1000                                   ' 1 second delay before messages
DEBUG "Program running..."                   ' Program running message

DO                                           ' Main loop

  SERIN 16, 84, [char]                       ' Get character from PC
  SEROUT 11, 84, [char]                      ' Transmit a copy with P11
  DEBUG char                                 ' Echo in back to Debug Terminal

LOOP                                         ' Repeat main loop
```

Figure 6-26 shows the Debug Terminal's Transmit and Receive windowpanes. The Receive windowpane receives and displays messages that the BASIC Stamp sends as a result of **DEBUG** or **SEROUT 16,…** commands. The Transmit windowpane is for typing characters that you want to send to the BASIC Stamp. The program can receive these characters with either **DEBUGIN** or **SERIN 16,…** To send a message from the Debug Terminal to the BASIC Stamp, you have to first click the Transmit windowpane, and then you can start typing.

- ✓ Click the Debug Terminal's Transmit windowpane.
- ✓ Be prepared to repeatedly press a key on your keyboard.

Figure 6-26: Debug Terminal Transmit and Receive Windowpanes



Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off

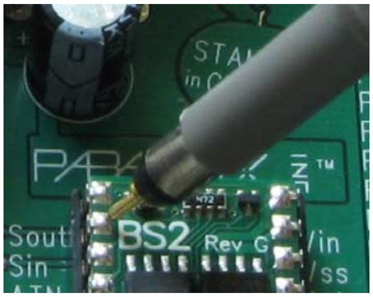
```
CCCCCCCCCCCCCCCCCCC
```

```
Program running...CCCCCCCCCC
CC
```

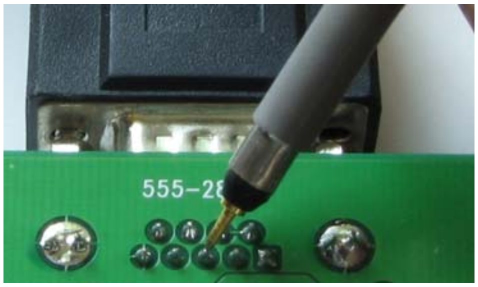
Figure 6-27 shows locations for testing the incoming signal from a PC on various boards. In both cases, the incoming signal test point is adjacent to the outgoing signal's test point.

- ✓ Press the CH2 probe end against the SIN probe point for your board.

Figure 6-27: CH2 Probe Position for SIN Signal



Board of Education (All)



BASIC Stamp HomeWork Board (Serial)
(Underneath the Serial Port Connecter)

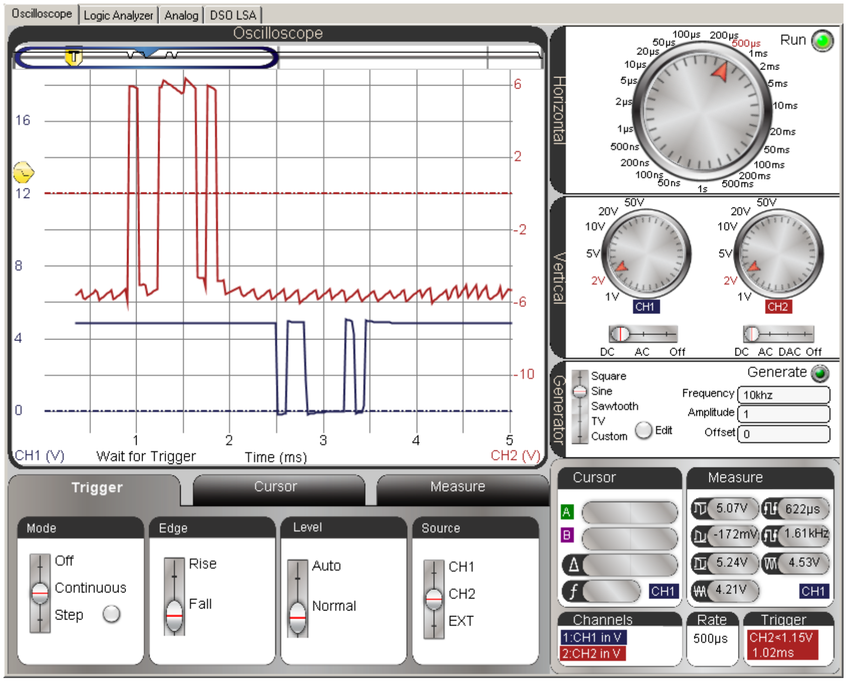
For other boards, get directions from the Supplement at www.parallax.com/go/propscope.

Figure 6-28 shows an incoming serial "C" from the keyboard. The inverted incoming signal is shown in the upper CH2 trace, and the true version transmitted by P11 is shown in the lower CH1 trace. Since the CH2 signal swings from +6 to –6 V, we can infer that the voltage rails that connect to the serial driver chip in this particular computer are ±6 V.

The voltage levels you observe may be different but should fall somewhere in the ±3 to ±25 V range.

- ✓ Adjust your PropScope for the Horizontal, Vertical and Trigger settings shown in Figure 6-28.
- ✓ Make sure the Trigger Voltage control is somewhere in the CH2 trace's ±3 V range and the Trigger Time control is aligned with the second time division. Keep in mind that even though the Trigger Voltage control is on the left margin, when you trigger from the CH2 trace, the voltage scale values are shown on the Oscilloscope screen's right margin.
- ✓ Click the Debug Terminal's Transmit windowpane again, and try tapping a few keys. Figure 6-28 shows an example of SHIFT + C.

Figure 6-28: Inverted "C" at SIN Pin (upper CH2 trace), True "C" at P11 (lower CH1 trace)



Frequency 10khz
Amplitude 1
Offset 0

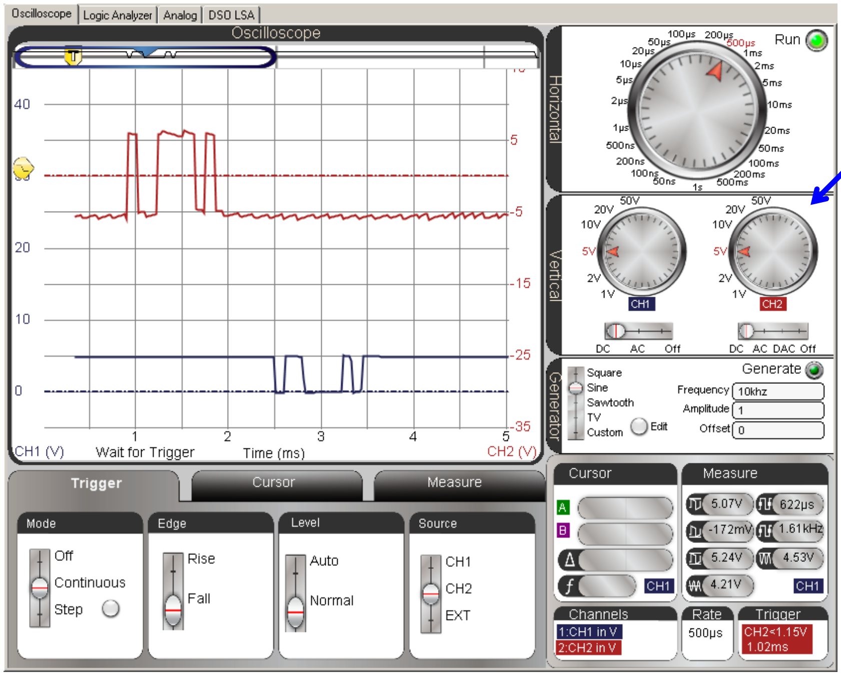
Measure: 5.07V | 622µs | -172mV | 1.61kHz | 5.24V | 4.53V | 4.21V

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 500µs
Trigger: CH2<1.15V 1.02ms

If your PC's voltage swings are more than ±6 V, there might not be enough room to comfortably display both signals. Figure 6-29 shows how increasing the voltage per division settings makes more room for larger voltage swings.

- ✓ Try changing the vertical dials from 2 V/division to 5 V/division.

Figure 6-29: Increasing Voltage Scales Reduces Trace Heights



Change Vertical dials to 5 V/div

Before You Continue

The activities in the next chapter are designed for 1x probes; none of them utilize the 10x setting.

- ✓ Set both probe X1 X10 switches back to X10.
- ✓ Navigate back to the PropScope software's Manage Probes window and set both probes back to multipliers of 1.
- ✓ Reattach the probe tip back to CH2 probe.

SUMMARY

Microcontrollers can communicate with a variety of devices using asynchronous serial communication. Like synchronous serial communication, a series of binary values are transmitted and received. Unlike synchronous serial communication, the devices exchanging data do not depend on a shared clock signal to synchronize when the binary values are transmitted/received.

ASCII codes represent printable characters and some control codes, and are often used in conjunction with asynchronous serial communication, allowing devices to exchange text like keyboard and display data.

Asynchronous serial signals can be true or inverted, have a baud rate, which is a number of bits per second, and have other characteristics such as a certain number of data bits, parity, and a certain number of stop bits. The baud rate determines the amount of time each bit lasts—the bit time is the reciprocal of the baud rate, in bits per second. A true signal has a high resting state, a low start bit, and then a certain number of data bits. Like all the other bits, each data bit is a signal that lasts one bit time. With true signaling, a high represents a binary-1 during a certain bit time and a low signal represents a binary-0. Asynchronous serial bytes are transmitted least significant bit first, followed by a stop bit, which is a resting state that lasts one or more bit times before the next message can be sent. Both devices rely on the negative edge of the start bit to determine when to send/receive the next binary value in the series.

Inverted serial communication is the voltage opposite of true serial communication, with all highs changed to lows and all lows changed to highs. RS232 serial communication is inverted and is the type of signaling between a PC and device(s) connected to its serial port(s). The high signals can be in the +3 to 25 V range and low signals in the -3 to -25 V range. High and low voltages will vary with computer hardware, but tend to be determined by their built-in power supply voltage rails.

The PropScope's time units per division can be customized by right clicking the Oscilloscope screen and filling in a number in the Timescale Value cell. This is useful for setting the units per division so that each one lasts a single bit time, which in turn is very useful for translating asynchronous serial messages into the values they represent. Setting the correct trigger edge is also important for aligning the start bit's edge with a time division. For trigger settings, keep in mind that a true signal initiates its start bit with a negative edge, and an inverted signal with a positive edge.

Chapter 7: Basic Sine Wave Measurements

SINE WAVE EXAMPLES

Figure 7-1 shows two cycles of a sine wave. Examples of this gradual up/down pattern can be found in waves in a pool or at sea, in air pressure variations from musical instruments playing notes, in radio and TV signals, and in the voltages supplied by AC electrical outlets.

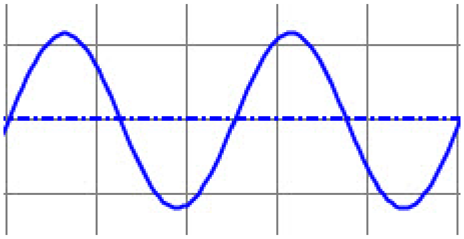


Figure 7-1
Sine Wave – Two Cycles

This chapter introduces basic sine wave measurements using signals that create tones and musical notes as examples. Many signals contain more than one sine wave component, like musical chords, phone tones, and even binary on/off signals. So, this chapter also introduces the *spectrum analyzer*, a diagnostic tool that can display a signal's sine wave components. Many circuit tests involve comparing a sine wave supplied to a circuit's input against the one at its output. So techniques for supplying a sine wave to a circuit and comparing the properties of the input and output signals are also introduced.

ACTIVITY #1: SINE WAVE AMPLITUDE AND FREQUENCY TESTS

When a musical note is played by an instrument like a guitar, the guitar string's vibrations cause air pressure variations. When a speaker plays a musical note, it converts electrical signals into vibrations that cause air pressure variations. Regardless of whether it's an actual instrument or a speaker playing the note, the eardrum senses the air pressure variations and forwards those signals to the brain via nerve connections. Then, our brain processes that information and we experience it as a musical note. Two properties of a note are volume and pitch. *Volume* is how loud the note sounds. *Pitch* is usually described in terms of high or low. For example, a bass guitar tends to play lower-pitched notes, while a normal guitar plays higher-pitched notes.

Musical notes are sine waves at specific frequencies, and you will experiment with them in the next activity. This activity will use more arbitrary frequencies, so instead of musical notes, they will just be “tones”. The PropScope’s function generator will be configured to apply sine waves to a piezospeaker, and its oscilloscope will be used to monitor them. With this setup, you’ll be able to “see” the sine waves with the oscilloscope at the same time that you “hear” the result played by the speaker.

The function generator has settings that allow you to control a sine wave’s amplitude and frequency. So in this activity, you will vary a sine wave’s amplitude and view its height change in the oscilloscope as you hear the speaker’s volume change. You will also vary the frequency and view the sine wave’s cycle width change in the oscilloscope as you hear the speaker tone’s pitch change.

Audio Tones Parts List

(1) Piezospeaker
(misc.) Jumper wires

Audio Tones Circuit

The PropScope’s function generator feature can be used to generate sine waves with certain amplitudes and frequencies, and those sine waves can be played by a speaker. Figure 7-2 shows a schematic and Figure 7-3 shows an example wiring diagram of the test circuit. The function generator sends sine wave voltages to the circuit, and the speaker makes it possible to “hear” to them as tones. The CH1 probe is for measuring the signals, and the Oscilloscope makes it possible to “see” them as sine waves.

- ✓ Disconnect the probe with the red marker band from the PropScope’s CH2 BNC port, and connect it to the DAC Card’s function generator output. (For detailed instructions, see the Set DC Voltages with the PropScope’s DAC Card section starting on page 45.)
- ✓ If the piezospeaker has a sticker covering it, remove that sticker.
- ✓ Build the circuit shown in Figure 7-2, using the wiring diagram in Figure 7-3 as a guide.

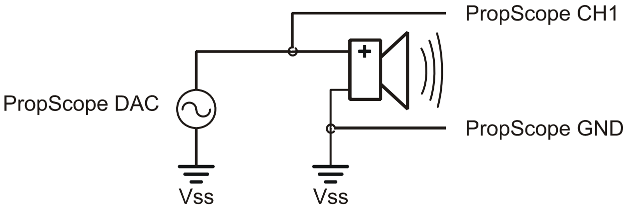


Figure 7-2: Audio Tones Schematic

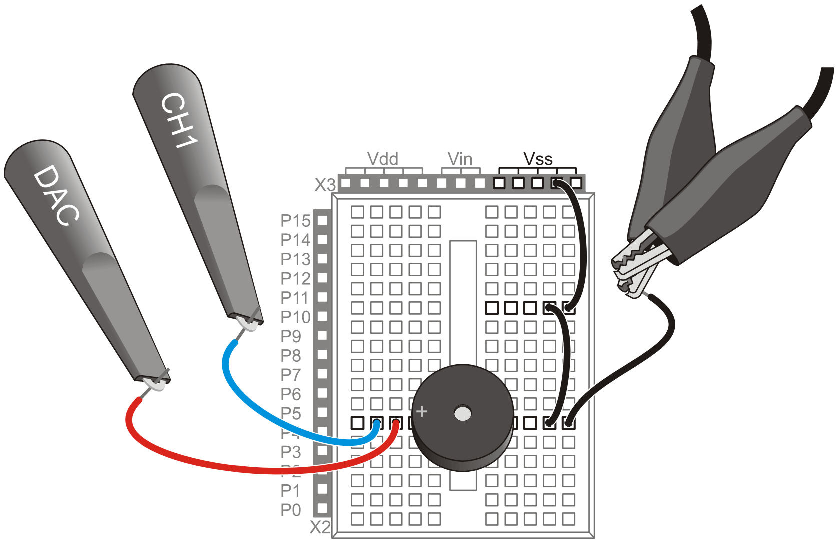


Figure 7-3: Wiring Diagram for Figure 7-2

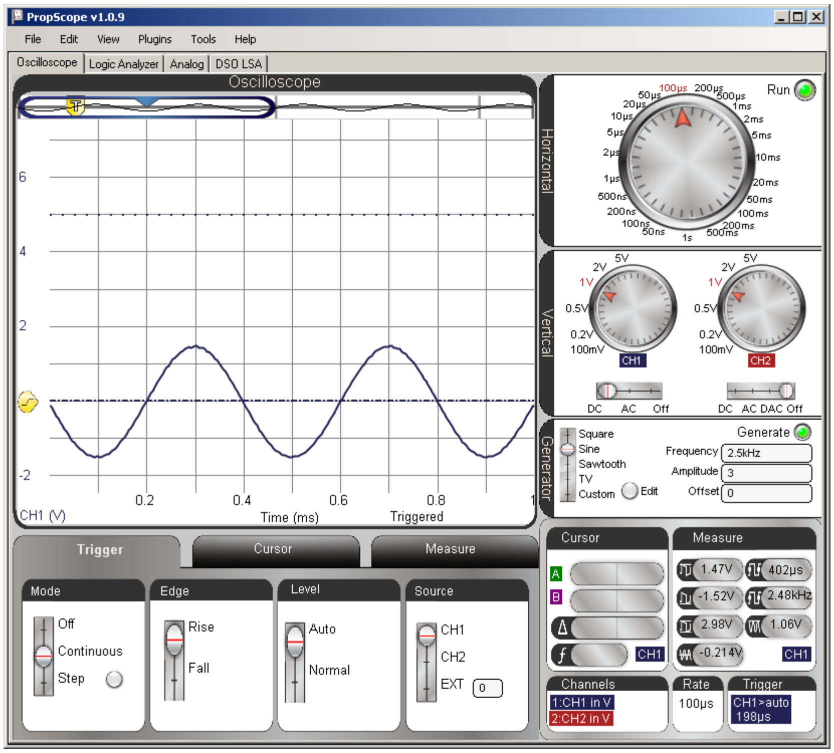
Audio Test Measurement Settings

Figure 7-4 shows the PropScope's Generator panel configured to make the function generator output a sine wave with a frequency of 2.5 kHz, a peak-to-peak voltage of 3 V, and a DC offset of 0 V. This signal should cause the piezospeaker to make an audible tone. The Oscilloscope screen makes the tone visible as well as audible. Remember that the Oscilloscope screen is a plot of voltage vs. time, so what you see in the display is a graph of the voltage applied to the speaker as it varies with time in a sine wave pattern.

- ✓ Set the dials: Horizontal = 100 µs/div, Vertical CH1 = 1 V/div, CH2 = 1 V/div.
- ✓ Vertical coupling switches: CH1 = DC, CH2 = Off.

- ✓ Optional: Drag the CH1 ground line up/down to position it about 2 ½ voltage divisions above the time scale so that it matches Figure 7-4.
- ✓ Trigger tab switches: Mode = Continuous, Edge = Rise, Level = Auto, and Source = CH1.
- ✓ Generator panel: Function switch = Sine, Frequency = 2500, Amplitude = 3, and Offset = 0.
- ✓ Click the Generator panel's Generate button make the DAC Card start transmitting the signal.
- ✓ Make sure you can hear the speaker making an audible tone, and that you can see a sine wave in your Oscilloscope screen similar to Figure 7-4.
- ✓ Try using the Run button (not the Generate button) to freeze the signal and turn the function generator signal (sound) off.
- ✓ Click the Run button again to restart the sound and the display.

Figure 7-4: Function Generator and Oscilloscope Settings



Run button vs. Generate button.

For this activity, the Run button is better than the Generate button for stopping the tone because it freezes the display before it stops the PropScope and DAC Card. It turns the sound off while preserving the last sine wave that was displayed in the Oscilloscope screen.

If you use the Generate button instead, it does toggle the function generator on/off, which will stop the tone too. However, the oscilloscope will keep running, and there's a chance that the signal activity as the function generator switches off will trigger the display. If that happens, your sine wave will be replaced by some miscellaneous switching activity.

Test Amplitude's Effect on Volume

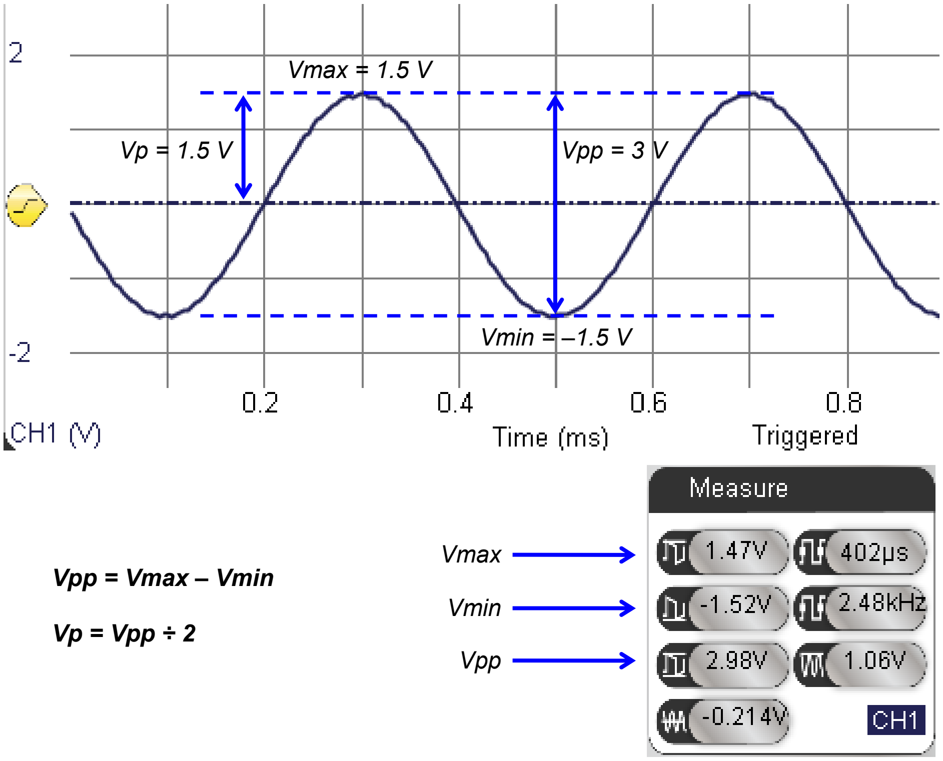
A sine wave's amplitude can be measured in several different ways. Figure 7-5 shows two examples: *peak-to-peak voltage* (Vpp) and *peak voltage* (Vp). With an oscilloscope, peak-to-peak amplitude is the voltage difference between the tops of a sine wave's peaks (Vmax) and the bottoms of its valleys (Vmin). The PropScope displays Vmax, Vmin, and Vpp for a given channel in its measure display.

Another common amplitude measurement is *peak voltage* (Vp), which is just the height of a sine wave's peaks measured from the halfway point. Peak amplitudes are more common in equations that describe sine waves. In contrast, peak-to-peak amplitudes are more common in oscilloscope measurements. If you ever need the peak amplitude measurement for an equation, just take the peak-to-peak measurement from your oscilloscope and divide by 2.

Larger sine wave amplitudes result in louder piezospeaker volumes; smaller amplitudes result in quieter speaker volumes. Figure 7-6 shows how a cycle of the sine wave looks at three different amplitudes. As the amplitude increases, the sine wave's height in the oscilloscope increases, and so does the volume. Try these adjustments to the signal's amplitude:

- ✓ If your tone and display are paused, click the On button again to resume viewing the live signal and playing the tone.
- ✓ Type 1 into the Generator panel's Amplitude field and press Enter.
- ✓ Listen to volume from the speaker and make a note of the sine wave's amplitude in the Oscilloscope screen.
- ✓ Repeat by entering values of 2 and then 3 into the Generator panel's Amplitude field. The tone should get a little bit louder with each adjustment. Listen to the volume and make a note of the Oscilloscope amplitude after each adjustment.

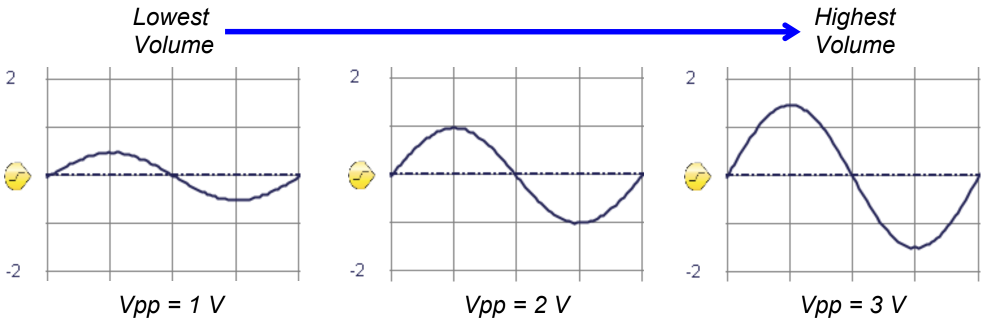
Figure 7-5: Peak-to-Peak vs. Peak Amplitude



Vpp = Vmax – Vmin

Vp = Vpp ÷ 2

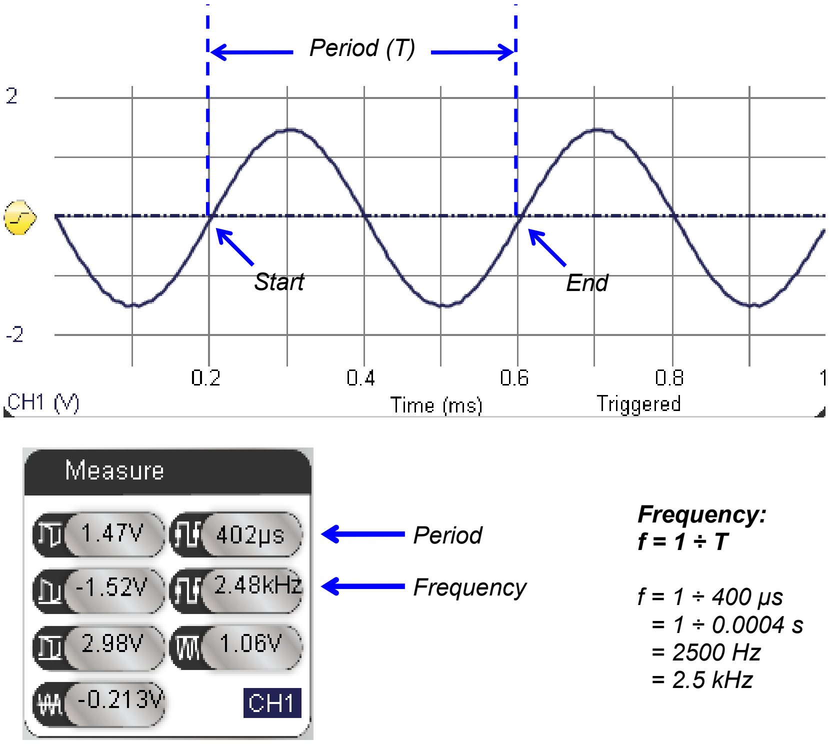
Figure 7-6: Signal Amplitude Compared to Speaker Volume



Test Frequency's Effect on Pitch

Chapter 3, Activity #3 introduced 'frequency' as the number of times a signal repeats itself per second and 'period' as the signal's cycle time. Figure 7-7 indicates the start and end of a single sine wave cycle. The start of a sine wave's cycle is the point where it passes upward through the half way point on its way from valley to peak. The cycle time can be measured to determine the sine wave's period (T). Then, its frequency (f) can be calculated with f = 1 ÷ T. The PropScope's Measure display also shows the sine wave's measured period and frequency values for a given channel. In this case, the period can also be measured by counting the number of 100 µs divisions in a cycle. If the sine wave is exactly 2.5 kHz, one cycle should span exactly four 100 µs time divisions.

Figure 7-7: Period of One Sine Wave Cycle



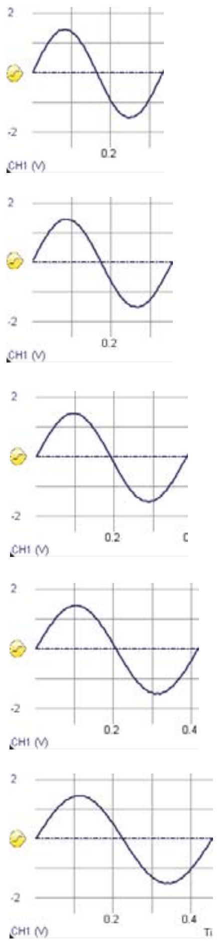
1.47V
402µs
-1.52V
2.48kHz
2.98V
1.06V
-0.213V
CH1

Frequency:
f = 1 ÷ T

f = 1 ÷ 400 µs
= 1 ÷ 0.0004 s
= 2500 Hz
= 2.5 kHz

Figure 7-8 shows how the frequency relates to the tone's pitch. A higher frequency results in a higher-pitched speaker tone. It also results in a more compressed waveform on the Oscilloscope screen. A lower frequency results in a lower-pitched tone, and it also results in a less compressed or more stretched out waveform on the Oscilloscope screen.

Figure 7-8: Pitch Compared to Frequency and Period



Pitch	*Frequency*	*Period*
Highest	*3.0 kHz*	*333.3 µs*
	2.8 kHz	*357.1 µs*
	2.6 kHz	*384.6 µs*
	2.4 kHz	*416.7 µs*
Lowest	*2.2 kHz*	*454.5 µs*

Look carefully at the width of each cycle. The widest cycle (longest period) is at the bottom, and they get progressively shorter as the frequencies increase.

As frequency increases, so does pitch.

Also, as frequency increases, period decreases because f = 1 ÷ T.

- ✓ Set the Generator panel's Amplitude to 3 Vpp.
- ✓ Type 2200 into the Frequency field and press Enter.

✓ Listen to the pitch of the tone the speaker makes.
✓ Make a note of the waveform's period as well as how wide it looks in the Oscilloscope.
✓ Repeat with frequency settings of 2400, 2600, 2800, and 3000.
✓ Try entering the five frequencies in a fairly rapid succession as you listen carefully to identify the increase in pitch.

In terms of f = 1 ÷ T, the highest frequency waveform has the shortest period (time between signal repetitions). That again is because period is the reciprocal of frequency: T = 1 ÷ f. Whenever f gets larger, T gets smaller, and vice-versa.

The PropScope's Frequency field accepts frequencies from 1 Hz to over 1 MHz in increments of 1/10th Hz.

If the sine wave gets too compressed or too stretched out to fit well in the Oscilloscope screen, just adjust the Horizontal dial's time/division setting. Decrease it for higher frequencies to uncompress the waveform's display, or increase it for lower frequencies to get a waveform to fit that would otherwise look stretched.

Your Turn: Test the Piezospeaker's Frequency Response

You may have noticed that the tone gets slightly louder as the frequency increases. That's because even though amplitude controls volume, certain speakers play certain frequencies louder than others. All speakers have natural filtering properties that depend on their electromechanical design. The woofer, midrange and tweeter names for speakers indicate that they play particular frequencies louder and clearer than the others. A *woofer* plays low frequencies, and it would be difficult to hear high notes played through it. A *tweeter* plays high frequencies, making it difficult to hear the low notes. The *midrange* excels with middle notes. You can hear the low and high notes through a midrange speaker, but they are quieter. With all three speakers working together, their signals add up to make a nice full sound.

When testing the frequency response of a speaker, keep the amplitude of the sine wave the same. If one tone is louder than the next, you know it's a property of the speaker since the amplitude did not change. If a tone sounds really loud, it may be because the speaker is designed to vibrate at that frequency—kind of like when you ring bells of different sizes and shapes. Each one has mechanical properties that resonate at a particular frequency. This natural frequency is called the *resonant frequency*. The piezospeaker in your Understanding Signals kit is designed with a resonant frequency for smoke alarms, so it sounds louder at higher frequencies.

- ✓ Set the Generator panel's Amplitude to 3 V, and make sure to press Enter with each adjustment to update the function generator's output.
- ✓ Try 1, 2, 3, 4, 5, 6, and 7 kHz.

Can you make it any louder? Hint: the resonant frequency should be half way between two of the frequencies you have already tested.

ACTIVITY #2: DC OFFSET TESTS

DC offset is useful for certain designs and detrimental to others. A useful example would be a signal that is given some DC offset so that every cycle crosses the BASIC Stamp 1.4 V logic threshold. A detrimental example would be giving a normal audio speaker a sine wave with DC offset. They are designed to receive signals with 0 V_{DC} offset.

In this part of the activity, we'll add some DC offset to the signal applied to the piezospeaker. The Understanding Signals Kit's piezospeaker is not the same as a normal audio speaker; it can receive signals with DC offset. From the audio standpoint, there will be little or no difference in the sound, but the signal will look different in the Oscilloscope screen.

> **WARNING: Only use a piezospeaker here. Do not** try adding DC offset to audio speaker signals. Audio speakers have wound wire coils built-in. These coils have very low resistance compared to piezospeakers. DC voltage could result in excessive current demands on the PropScope and its USB port power supply.

> **The element in a piezoelectric speaker** changes shape with applied voltage, and when the voltage oscillates rapidly, the piezo element's vibrations create the tones. This process takes only small amounts of current.
>
> **In an audio speaker**, an electromagnet acts on a magnet attached to a paper cone to make it vibrate. The oscillating voltage applied across the electromagnet's coil results in oscillating current through the coil. This oscillating current through the coil creates an oscillating magnetic field that acts on the magnet attached to the paper cone.

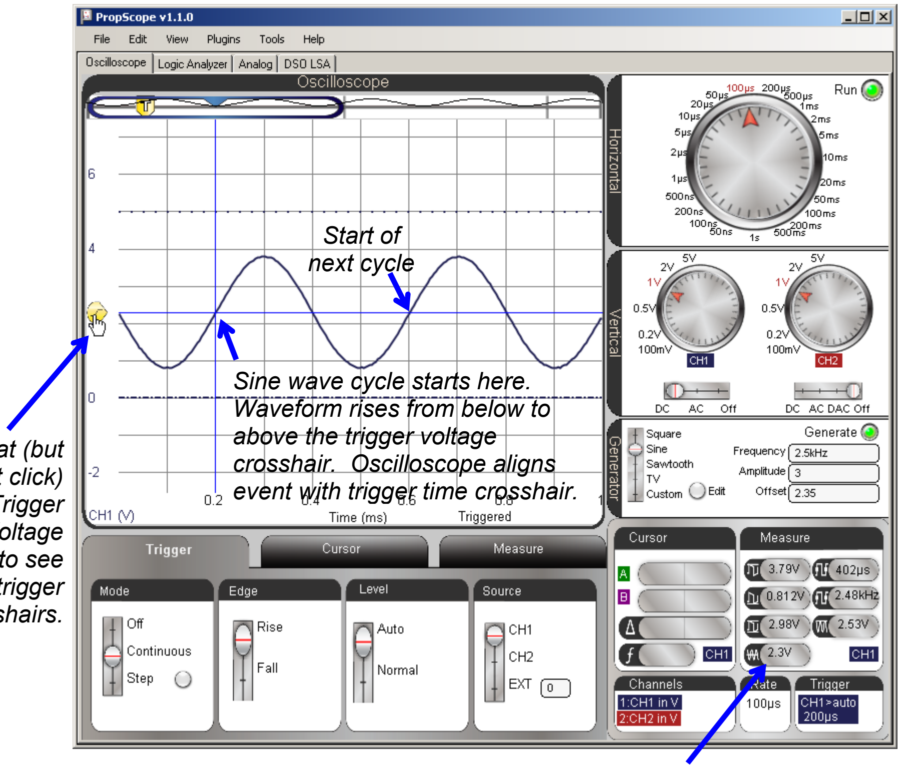
Auto Trigger Follows Average Voltage (DC Offset)

Since the Trigger tab's Level switch is set to Auto, the oscilloscope automatically adjusts the Trigger Voltage control to follow the waveform's average voltage. Remember that DC offset and average voltage are one and the same. So, when you set the Generator panel's Offset to a value, the DC offset = average voltage changes, and the

Trigger Voltage control follows it. Up to this point, the Generator panel's Offset has been set to 0 V, which is where the Oscilloscope has been automatically positioning the Trigger Voltage control.

- ✓ Verify that the Trigger Voltage control's level is 0 V. It should be automatically positioned to the left of the Ground line.
- ✓ Change the Generator panel's Offset from 0 to 2.35 V and press Enter.
- ✓ Verify that the waveform's average voltage changes to 2.35 V.
- ✓ Point at, but do not click the Trigger Voltage control. (See Figure 7-9.) Verify that the Trigger Voltage control has automatically repositioned to approximately 2.35 V.

Figure 7-9: Trigger Auto Level Adjusts with DC Offset



Frequency 2.5kHz

Amplitude 3

Offset 2.35

EXT 0

Measure: 3.79V, 402µs, 0.812V, 2.48kHz, 2.98V, 2.53V, 2.3V, CH1

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 100µs

Trigger: CH1>auto 200µs

Start of next cycle

Sine wave cycle starts here. Waveform rises from below to above the trigger voltage crosshair. Oscilloscope aligns event with trigger time crosshair.

Point at (but don't click) the Trigger Voltage control to see the trigger crosshairs.

Average voltage is ≈ 2.35 V

The Trigger tab's Edge switch in Figure 7-9 is set to Rise. So, the Oscilloscope aligns a point where the sine wave rises through the 2.35 V Trigger Voltage crosshair with the Trigger Time crosshair. For a sine wave, this is the start of its cycle. You can see part of a second cycle in the screen, starting four time division lines to the right of the trigger event.

- ✓ Try changing the Edge switch in the Trigger tab from Rise to Fall. The voltage should now fall through the trigger crosshair intersection instead of rising through it.
- ✓ Restore the Trigger tab's Edge switch to Rise.

If the Trigger tab's Level switch is set to Normal, it means that you will manually adjust the Trigger Voltage Level control. If the waveform's DC offset or amplitude changes enough so that the signal doesn't cross the Trigger Voltage level, the Oscilloscope screen will not refresh. There may still be signal activity, but it will not be displayed on the Oscilloscope screen. Try this.

- ✓ Set the Generator panel's Offset back to 0 V. The Trigger Voltage control should automatically reposition to 0 V.
- ✓ Move the Trigger tab's Level switch to the Normal setting.
- ✓ Change the Generator panel's Offset back to 2.35 V.

At this point, the sine wave should still appear to have a 0 V offset, and a red status warning should appear in the lower right of the Oscilloscope screen: Wait for Trigger. That's because the waveform never crosses the 0 V trigger voltage, so the display does not update.

- ✓ Now, manually drag the Trigger Voltage control up to 2.35 V.

The sine wave should reappear, displayed at the correct level.

- ✓ Before continuing, set the Trigger tab's Level setting back to Auto.

Offset and Amplitude Tests

Up to this point, your PropScope is displaying a 3 Vpp sine wave with a 2.35 V offset. The sine wave extends to 3.85 V peaks and 0.85 V valleys. Your signal is currently in the larger of two function generator voltage ranges: 0 to 4.7 V, and the waveform's DC offset is currently 2.35 V, which is in the middle of that range. So, you could increase the Generator panel's Amplitude setting all the way to 4.7 Vpp. This is a larger amplitude than any of the tests in the –1.5 to +1.5 V range, where the maximum amplitude was 3 Vpp.

- ✓ Try setting the Generator panel's Amplitude to 4.7 Vpp.
- ✓ Compare that to Amplitude = 3 Vpp. Can you hear the volume difference? Can you see the difference in the Oscilloscope screen as well?

With 4.7 Vpp of amplitude, there is no room to experiment with DC offset adjustments. With 3 Vpp of amplitude and 2.35 V_{DC} of offset, there is some room to experiment, but not much. 3 Vpp of amplitude makes it possible to adjust the DC offset 0.85 V up or down without exceeding the function generator's 0 to 4.8 V limits. By changing the Amplitude to 1 Vpp, you'll have more room to adjust the offset. In fact, 1 Vpp can fit into either the = –1.5 to +1.5 V or 0 to 4.8 V ranges. So, you'll be able to adjust the sine wave's offset to anywhere in the –1 V to +4.2 V range.

- ✓ Change the Generator panel's Amplitude to 1 V.
- ✓ Enter each of these DC offset values into the Generator panel's Offset field: –1, 0, 1, 2, 3, 4, and 4.2 V.

Keep in mind that the peak amplitude (which is ½ the peak-to-peak amplitude) is the total excursion above and below the sine wave's DC offset. So with a 1 Vpp amplitude, when you change the Generator panel's offset to –1 V, its voltage oscillates from 0.5 V above that level to 0.5 V below it. That's why the DC offset can be adjusted clear down to –1 V, because the lowest level the sine wave's valleys reach is the function generator's –1.5 V lower limit.

Amplitude and Offset for BASIC Stamp Frequency Counting

The BASIC Stamp can measure the frequencies of sine waves. The PropScope's function generator output needs to be connected to an I/O pin to test this. Then a program can make the BASIC Stamp count signal cycles over some period of time to measure the frequency. For the BASIC Stamp to take these measurements, the signal has to have some combination of amplitude and offset that makes it cross the BASIC Stamp I/O pin logic input threshold. I/O pin threshold voltage was tested back in the Determining I/O Pin Threshold Voltage section on page 55, and it's typically very close to 1.4 V.

Extra Test Parts

(1) Resistor – 220 Ω (red-red-brown)

Test Schematic

Figure 7-10 shows a modified version of the test circuit from Figure 7-2. The only difference is the extra 220 Ω resistor that connects the function generator's output to a BASIC Stamp I/O pin. So now, the function generator's output goes to both the piezospeaker and the BASIC Stamp I/O pin.

✓ Add the 220 Ω resistor connecting the DAC Card's function generator output to the I/O pin input. You can also leave it connected to the speaker's input.

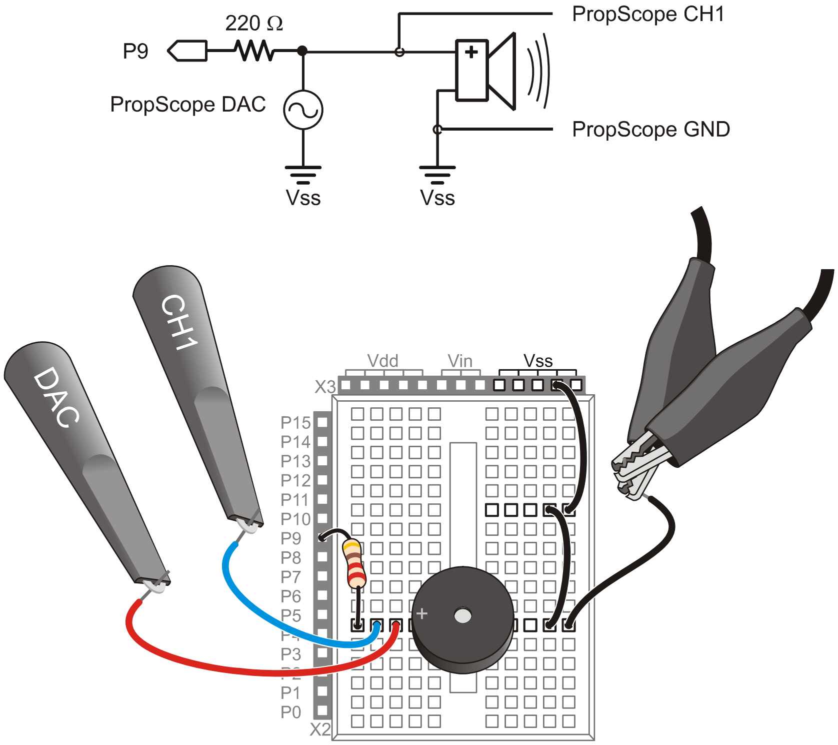


Figure 7-10
Audio Tones Test Circuit Modified for BASIC Stamp Monitored Frequency

Test Code

Display Measured Freq.bs2 measures the number of sine wave cycles the PropScope applies to P9 during 1 second intervals and displays the result in the Debug Terminal.

- ✓ Enter and run Display Measured Freq.bs2.

```
' Display Measured Freq.bs2
' Count signal cycles applied to P9 and display in Debug Terminal

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

cycles VAR Word                              ' Stores counted cycles

DO                                           ' Main loop

  COUNT 9, 1000, cycles                      ' Store 1 s counted in cycles var
  DEBUG HOME, "Input frequency = ",          ' Display result
        DEC5 cycles, " Hz"

LOOP                                         ' Repeat main loop
```

Test Procedure

The function generator should be adjusted to make the sine wave cross the BASIC Stamp module’s 1.4 V I/O pin input threshold.

- ✓ Set the function generator’s amplitude to 2.8 V, and the offset to 1.4 V. Remember to press the Enter key to make the PropScope update its function generator output.
- ✓ Verify that the frequency the BASIC Stamp reports to the Debug Terminal agrees with the one in the PropScope Measure display, as shown in Figure 7-11.

Your Turn: Amplitude/Offset Options to Maintain 1.4 V Crossings

Since the BASIC Stamp is just counting the number of times the signal crosses its 1.4 V threshold for a frequency measurement, the amplitude does not have to be 2.8 V. Since the offset is 1.4 V, the sine wave will still crosses the I/O pin threshold with each cycle, regardless of whether the amplitude is 2.8 V, 2 V, 1 V, or even 0.5 V.

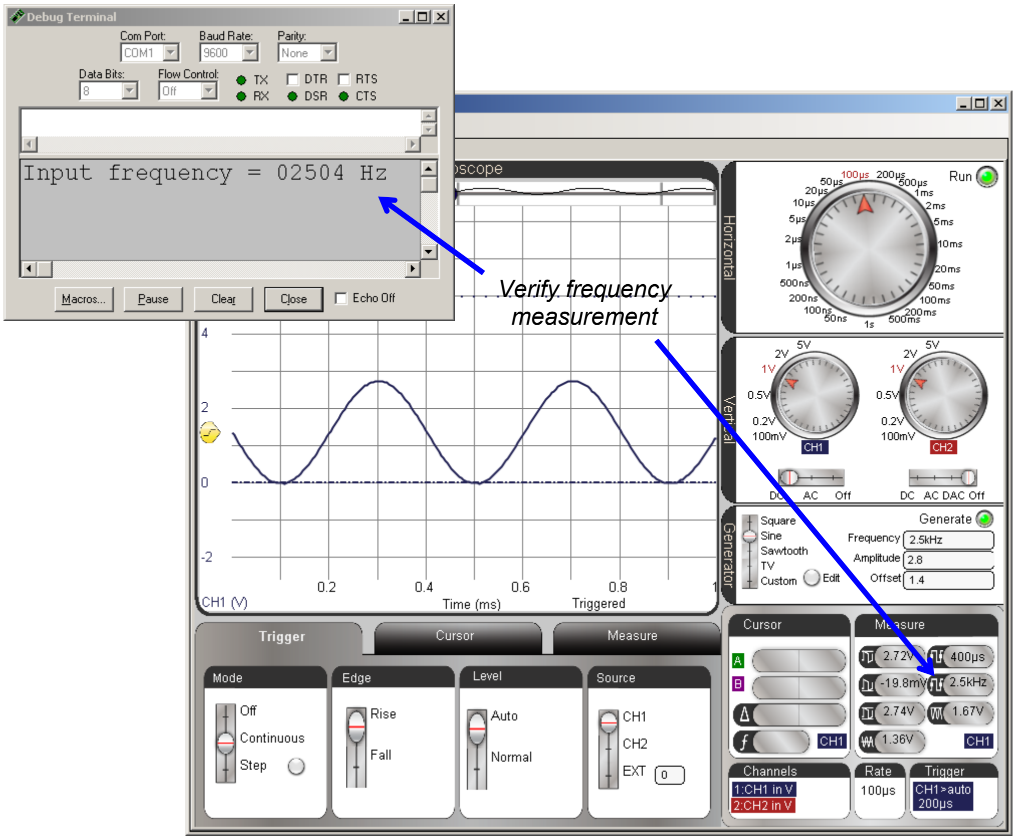
- ✓ Try a few different signal amplitudes with the 1.4 V offset, and verify that the BASIC Stamp continues to count signal cycles.

- ✓ Set the amplitude to 2 Vpp before continuing.

With a 2 Vpp sine wave, the offset can be increased to somewhere in the neighborhood of 2.4 V before the sine wave no longer passes the 1.4 V threshold.

- ✓ Try it.

Figure 7-11: BASIC Stamp Frequency Measurement Compared to PropScope



Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off

```
Input frequency = 02504 Hz
```

Frequency: 2.5kHz
Amplitude: 2.8
Offset: 1.4

Verify frequency measurement

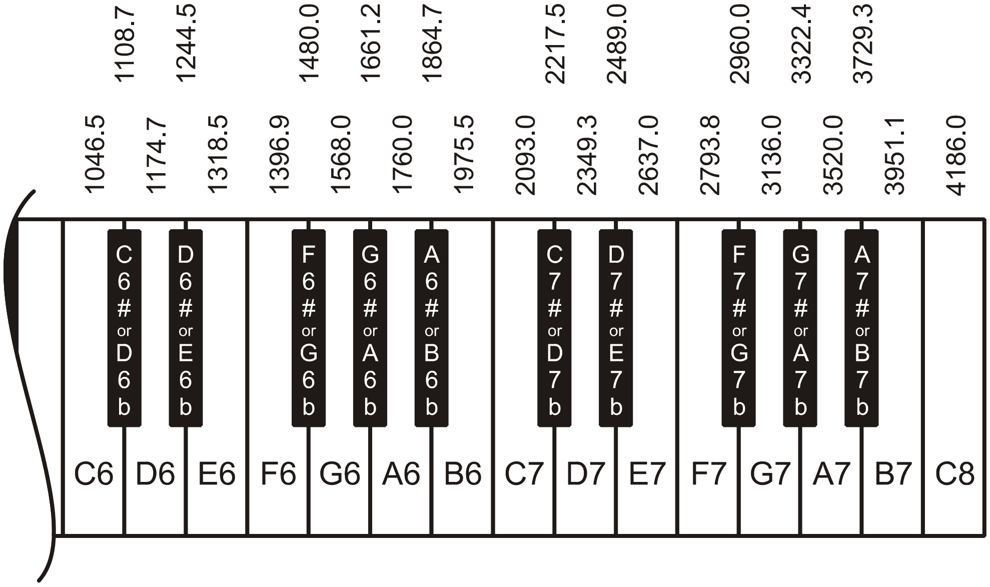
ACTIVITY #3: MEASURE BASIC STAMP MUSICAL NOTES

The BASIC Stamp can be programmed to transmit many sine wave frequencies, including integer versions of the musical notes shown in Figure 7-12. In this activity, you will program the BASIC Stamp to synthesize the D7# (pronounced D 7 sharp) and F7# notes. D7# is the black key below the 2489.0 Hz frequency in Figure 7-12, and F7# is the black key below 2960.0 Hz. You will then use the Oscilloscope view's cursor features to measure to examine each sine wave's amplitude, frequency, and offset.

Recommended: Music with Microcontrollers

What's a Microcontroller? Chapter 8 shows how to make sound effects and music with the BASIC Stamp 2 microcontroller using the same piezospeaker that's in the Understanding Signals kit. The book is a free PDF download from www.parallax.com/go/WAM.

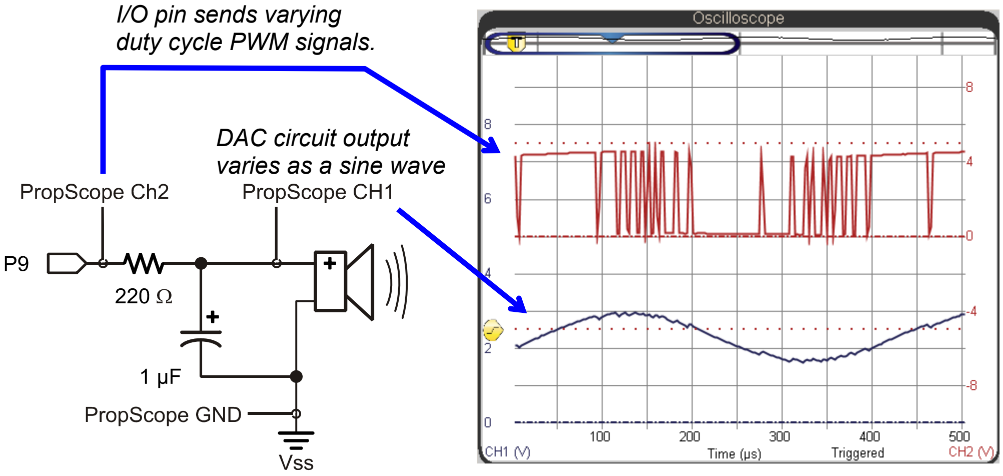
Figure 7-12: Rightmost Piano Keys and Their Frequencies



How the BASIC Stamp "Digitally Synthesizes" Sine Waves

This activity will use a PBASIC command called **`FREQOUT`** to make the BASIC Stamp "digitally synthesize" various sine waves. For example, **`FREQOUT 9, 60000, 2500`** uses I/O pin P9 to transmit a signal that makes an RC DAC circuit synthesize a 2.5 kHz sine wave for 60 seconds. Figure 7-13 shows a single cycle of that sine wave in the lower CH1 trace. The upper CH2 trace is the binary signaling the BASIC Stamp transmits with I/O pin P9 to make the RC DAC circuit's output voltage increase and decrease in the sine wave voltage pattern.

Figure 7-13: Varying PWM from I/O Pin Converted to Sine Wave with DAC Circuit



I/O pin sends varying duty cycle PWM signals.

DAC circuit output varies as a sine wave

Chapter 4, Activity #2 introduced how the **`PWM`** command used duty cycle to set DC voltages. The **`FREQOUT`** command varies the **`PWM`** duty cycle to create a sine wave. Notice how the **`PWM`** signal spends larger portions of time high as the sine wave's voltage increases, and more time low as the sine wave's voltage decreases. The duty cycle varies like this 2500 times per second to shape the 2.5 kHz sine wave.

The process the BASIC Stamp uses to set the varying duty cycles is similar to the process it used to synthesize the heartbeat signal in Chapter 3, Activity #5. Instead of a list of values obtained from an electrocardiograph, the **`FREQOUT`** command uses a list of values obtained from sine wave graph. The BASIC Stamp also has to cycle through the different D/A conversions at a much higher rate, 2500 cycles per second instead of 70.8 cycles per minute.

BASIC Stamp Tones Parts List

(1) Piezospeaker
(1) 1 µF Capacitor
(1) 220 Ω Resistor (If you have a BASIC Stamp HomeWork Board, use a wire instead.)
(misc.) Jumper wires

BASIC Stamp Tones Circuit

Figure 7-14 and Figure 7-15 show the schematic and wiring diagram of a circuit that will allow you to alternately play a note with the piezospeaker and then view its sine wave in the Oscilloscope screen. With the capacitor in place, the PropScope will display a clean sine wave, but it will be difficult to hear on the speaker. With the capacitor disconnected, the tone from the speaker will be easier to hear, but instead of a sine wave, the PropScope will display the switching activity the I/O pin uses to synthesize the sine wave.

Even with all that binary on/off switching the speaker tone still sounds true because the switching frequency is very high compared to what the piezospeaker can broadcast, and our ears cannot pick up frequencies that high either. So, the combination of speaker and eardrum mechanically perform filtering for our brains that is similar to what the RC circuit performs for the PropScope.

- ✓ Make sure the CH2/DAC lead is disconnected from the circuit.
- ✓ Build the circuit in Figure 7-14 and Figure 7-15.
- ✓ The capacitor's negative lead is the one that's closer to the stripe with the minus signs on the capacitor's metal canister. Make sure this lead is connected to Vss.
- ✓ Leave the 1 µF capacitor's positive lead disconnected for now.

CAUTION:

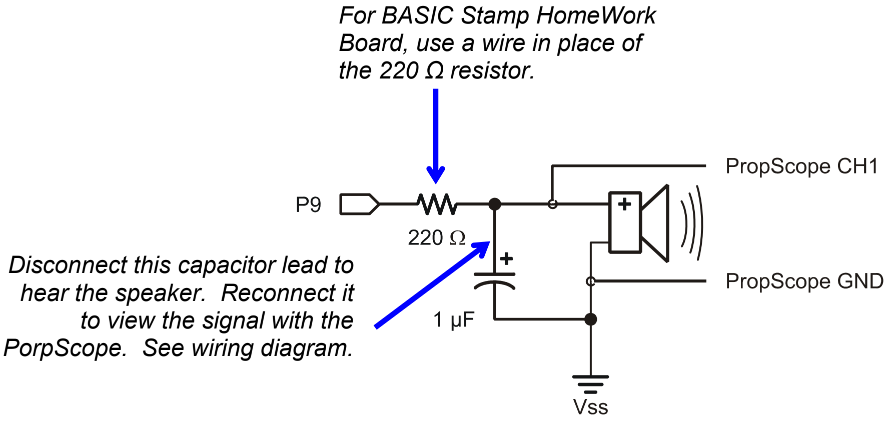
In some circuits, connecting this type of capacitor incorrectly and then connecting power can cause it to rupture or even explode. So, always make sure to carefully connect its terminals exactly as shown in the circuit diagrams.

This capacitor has a positive (+) and a negative (-) terminal. The negative terminal is the lead that comes out of the metal canister closest to the stripe with a negative (–) sign(s).

Do not apply more voltage to an electrolytic capacitor than it is rated to handle. The voltage rating is printed on the side of the canister.

Safety goggles or safety glasses are recommended.

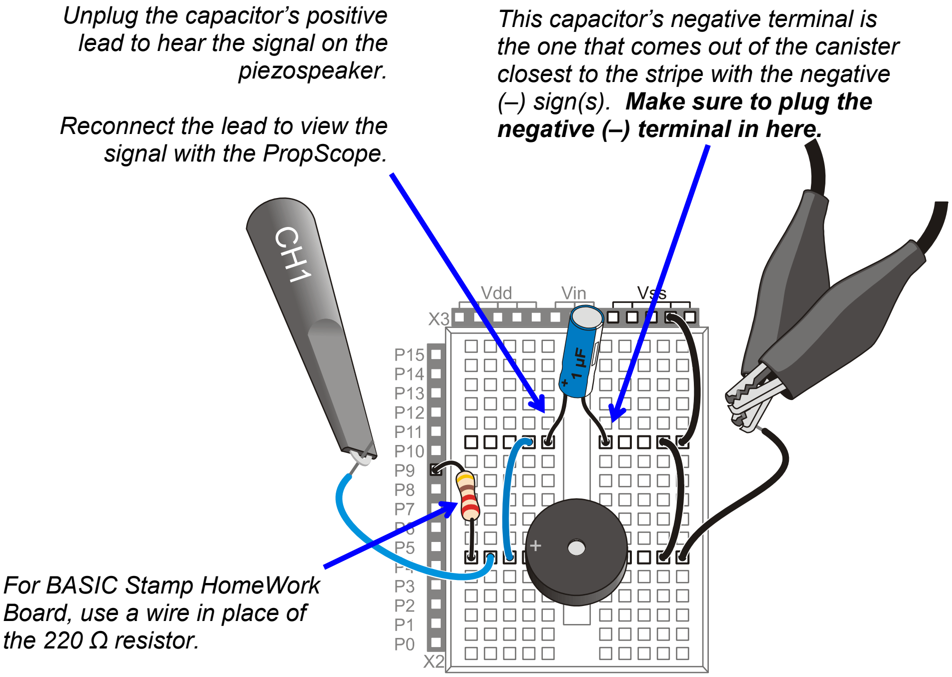
Figure 7-14: BASIC Stamp Controlled Piezospeaker with PropScope Probe and Capacitor for Signal Monitoring



For BASIC Stamp HomeWork Board, use a wire in place of the 220 Ω resistor.

Disconnect this capacitor lead to hear the speaker. Reconnect it to view the signal with the PorpScope. See wiring diagram.

Figure 7-15: Wiring Diagram for Figure 7-14



Unplug the capacitor's positive lead to hear the signal on the piezospeaker.

Reconnect the lead to view the signal with the PropScope.

This capacitor's negative terminal is the one that comes out of the canister closest to the stripe with the negative (–) sign(s). ***Make sure to plug the negative (–) terminal in here.***

For BASIC Stamp HomeWork Board, use a wire in place of the 220 Ω resistor.

Listen to the Notes First

Two Notes.bs2 plays the note D7# (2489 Hz), followed by F7# (2960 Hz). As mentioned earlier, it uses the **FREQOUT** command to digitally synthesize sine waves for these notes. The command **FREQOUT 9, 2000, 2489** makes I/O pin P9 synthesize a 2489 Hz sine wave for 2 seconds. Then, **FREQOUT 9, 2000, 2960** synthesizes a 2960 Hz sine wave for 2 seconds.

- ✓ Enter Two Notes.bs2 into the BASIC Stamp Editor and run it.
- ✓ Disconnect the 1 µF capacitor's positive lead from the circuit.
- ✓ Listen carefully to the two individual notes.
- ✓ Press and release the Reset button on your board a couple of times to re-run the program.

```
' Two Notes.bs2
' Play D7#, followed by F7#.

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

PAUSE 1000                                   ' Wait 1 sec. before 1st message

DEBUG "2489 Hz", CR                          ' Play/display D7#
FREQOUT 9, 2000, 2489

DEBUG "0 Hz", CR                             ' 1/2 s rest
PAUSE 500

DEBUG "2960 Hz", CR                          ' Play/display F7#
FREQOUT 9, 2000, 2960

DEBUG "0 Hz", CR                             ' 1/2 s rest
PAUSE 500

DEBUG "Done!", CR                            ' Display done

END                                          ' Enter low power mode
```

Play an Individual Note for Oscilloscope Viewing

One Note at a Time.bs2 makes the BASIC Stamp play the D# note indefinitely. You can instead play the F# note by commenting **FREQOUT 9, 60000, 2489** by placing an apostrophe **'** to the left of it. After that, remove the apostrophe to the left **of FREQOUT 9, 60000, 2960** and load the modified program into the BASIC Stamp.

✓ Enter and run One Note at a Time.bs2 as-is.

✓ After you are satisfied that it plays the note indefinitely, plug the capacitor's positive terminal back in using Figure 7-15 as a guide.

✓ The note should be much quieter, maybe even barely audible, but the sine wave will look much better on the Oscilloscope screen.

```
' One Note at a Time.bs2
' Select whether to play D7#, or F7# indefinitely.

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

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

DO                                           ' Main Loop

  FREQOUT 9, 60000, 2489                     ' Play 2489 Hz for 1 minute
  ' FREQOUT 9, 60000, 2960                   ' Commented, does not play

LOOP                                         ' Repeat main loop w/o delay
```

Examine the DC Offset, Amplitude and Frequency

Figure 7-16 shows the BASIC Stamp synthesized D7# note in the Oscilloscope view.

✓ Set the Oscilloscope dials to: Horizontal = 100 µs/div, Vertical CH1 = 1 V/div.

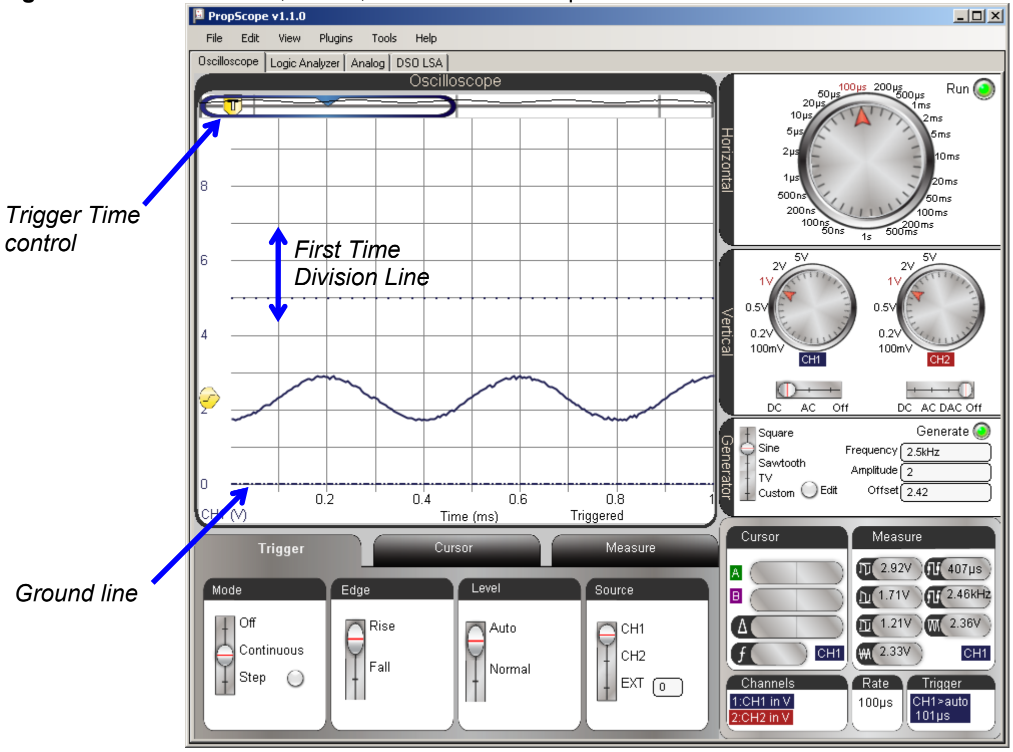
✓ Vertical Coupling switches: CH1 = DC, CH2 = Off.

✓ Trigger tab switches: Mode = Continuous, Edge = Rise, Level = Auto, and Source = CH1.

✓ Trigger Time control: align with first time division line. (See in Figure 7-16).

✓ Optional: Click, hold and drag the CH1 ground line downward until it is just above the time division scale values.

Figure 7-16: FREQOUT 9, 60000, 2489 in Oscilloscope



Frequency 2.5kHz
Amplitude 2
Offset 2.42
EXT 0

Measure: 2.92V, 407µs, 1.71V, 2.46kHz, 1.21V, 2.36V, 2.33V, CH1

Cursor: CH1

Channels: 1:CH1 in V, 2:CH2 in V

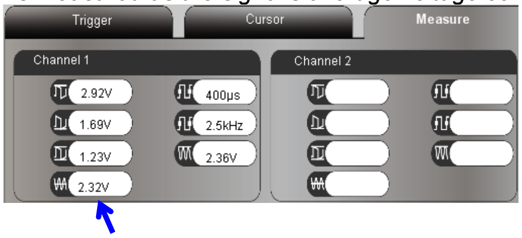
Rate: 100µs

Trigger: CH1>auto 101µs

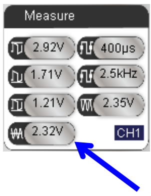
Note that the sine wave in Figure 7-16 has a DC offset like the ones that we added to the PropScope DAC sine waves in Activity #2. You can use the Measure display to get an automated DC offset measurement. This same measurement is also displayed in the Oscilloscope view's Measure tab. Both are shown in Figure 7-17.

- ✓ Click the Measure tab and check the CH1 Average voltage. It should resemble the example in Figure 7-17.
- ✓ Repeat using the Measure display.

Figure 7-17: Sine Wave DC Offset
…is measured as the signal's average voltage component



2.92V
1.69V
1.23V
2.32V
400µs
2.5kHz
2.36V



2.92V
400µs
1.71V
2.5kHz
1.21V
2.35V
2.32V

Average voltage
Is DC Offset

Average voltage
is also in the
Measure display

You can also use the PropScope software's horizontal and vertical cursors to measure amplitude and frequency. Figure 7-18 shows an example. The amplitude is the peak-to-peak-voltage, so place one horizontal cursor at the tops of the sine wave's peaks, and the other at the bottoms of its valleys. The average voltage is the half way point between the sine wave's peaks and valleys. Frequency can be determined from period, and the vertical cursors can be used to measure the amount of time it takes the signal to repeat itself. One way to do that is to align the vertical cursors with the centers of the sine wave's peaks.

✓ In the Cursor tab, click the Vertical and Horizontal buttons to make the vertical and horizontal cursors appear.

✓ Align the green horizontal cursor A with the tops of the sine wave's peaks.

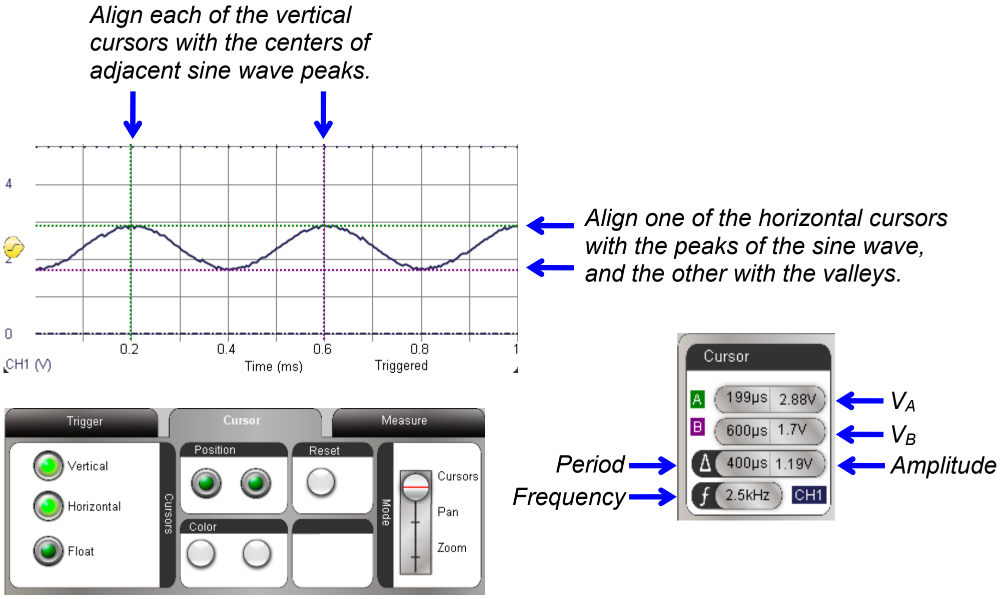
✓ Align the purple horizontal cursor B with the bottoms of the sine wave's valleys.

✓ Align the green vertical cursor A with the center of the top of the sine wave's first peak.

✓ Align the purple vertical cursor B with the center of the next second sine wave peak.

✓ Check the measurements in the Cursor display by the Oscilloscope screen's lower-right corner.

Figure 7-18: Cursor Measurements



Align each of the vertical cursors with the centers of adjacent sine wave peaks.

Align one of the horizontal cursors with the peaks of the sine wave, and the other with the valleys.

In Figure 7-17, the Measure tab and display reported amplitudes of 1.23 and 1.21 V. Since the cursor measurement from Figure 7-18 is 1.19 V it verifies that these measurements are in the right ballpark. Likewise with frequency measurements, both figures report 2.5 kHz. With the BASIC Stamp programmed to transmit 2489 Hz, these measurements indicate that the BASIC Stamp module's program and circuit are all working as intended. (This is the type of intermediate testing step you might see in a design that has an audio tones subsystem.)

The cursors can also provide enough information to verify the average voltage measurement the Measure tab and display reported. For a sine wave, the half way point between the sine wave's voltages at its peaks and valleys is the average of the two voltages. To calculate this average, just add the two cursor measurements, which are VA and VB in Figure 7-18, and divide by two:

$$\text{DC Offset} = \text{Average voltage} = (V_A + V_B) \div 2$$

For the cursor measurements in Figure 7-18, that's:

$$\text{DC Offset} = (2.88\text{ V} + 1.7\text{ V}) \div 2 = 2.29\text{ V}$$

The calculated DC offset value of 2.29 V from the cursor measurements is close enough to confirm the measured value of 2.32 V back in Figure 7-17.

Your Turn: Examine the F7# Sine Wave

Try taking a closer look at the F7# sine wave. Start by modifying One Note at a Time.bs2, and then examine the new signal. Because of the way the signal interacts with the resistor and capacitor, it may be of slightly lower amplitude. Since it's a higher frequency signal, you can also expect the period to be slightly shorter.

- ✓ In the BASIC Stamp Editor modify One Note at a Time.bs2 by placing an apostrophe to the left of **`FREQOUT 9, 60000, 2489`**. Then, delete the apostrophe to the left of **`FREQOUT 9, 60000, 2960`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Repeat this activity's offset, amplitude and frequency measurements.

7

> **If you want to replicate the measurement in Figure 7-13** after completing this activity, you can modify the **`FREQOUT`** command in One Note at a Time.bs2 so that it reads **`FREQOUT 9, 60000, 2500`**. You will also need to connect the CH2 probe to I/O pin P9 and adjust the PropScope's Vertical and Horizontal dials using what you have learned. If you have a BASIC Stamp HomeWork Board, accessing the actual I/O pin can be tricky because the 220 Ω resistor is built onto the board. See Figure 2-33 on page 63 for more info.

ACTIVITY #4: SUMMED SINE WAVES

A piano player presses several keys at a time to make sounds called *chords*. The notes seem to blend, and it's an example of a group of sine waves, one for each note, added together. Examples of pairs of sine waves added together can be found in touchtone phones. Each key press makes a unique tone that transmits that key's value. Since each touchtone is two tones added together, it's called dual tone frequency modulation (DTMF).

In addition to individual frequencies, the BASIC Stamp can be programmed to transmit two sine wave frequencies at once. This makes it possible to play part of a chord. In this activity, you will compare the D7# and F7# note frequencies from the previous activity against the two notes played together. As with the previous activity, you will listen to the results on the piezospeaker first, and then examine the signals with the PropScope's Oscilloscope view.

DTMF Tones: The BASIC Stamp can also play DTMF tones with the **`DTMFOUT`** command. Although they can be viewed with the PropScope, the tones' frequencies are typically too low to be audible through the piezospeaker in the Understanding Signals kit.

Listen to Each Note; then Play them Together

This example program starts off the same as the Two Notes.bs2 example from the previous activity— it plays the D7# and F7# notes individually. After that, it plays them simultaneously. **`FREQOUT 9, 2000, 2489, 2960`** uses the **`FREQOUT`** command's optional ***Freq2*** argument together with the ***Freq1*** argument to play the both sine waves together.

✓ Disconnect the capacitor's positive lead from the piezospeaker circuit so that you can hear the notes the piezospeaker plays.

✓ Enter Two Notes Together.bs2 into the BASIC Stamp Editor and load it into the BASIC Stamp.

✓ Listen carefully again to the two individual notes followed by the notes played together.

✓ Listen again and try to hear the differences in the three tones by pressing and releasing the Reset button on your board to re-run the program.

```
' Two Notes Together.bs2
' Play D7#, followed by F7#, followed by D7# + F7# together.

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

PAUSE 1000                                   ' Wait 1 s before first message

DEBUG "2489 Hz", CR                          ' Play/display D7#
FREQOUT 9, 2000, 2489

DEBUG "0 Hz", CR                             ' 1/2 s rest
PAUSE 500

DEBUG "2960 Hz", CR                          ' Play/display F7#
FREQOUT 9, 2000, 2960
```

```
DEBUG "0 Hz", CR                             ' 1/2 s rest
PAUSE 500

DEBUG "2489 Hz + 2960 Hz", CR                ' Play/display D7# + F7#
FREQOUT 9, 2000, 2489, 2960

DEBUG "Done!", CR                            ' Display done

END                                          ' Enter low power mode
```

Play an Individual Note for Oscilloscope Viewing (Again)

One or Two Notes at a Time.bs2 makes it possible to play either the D7# note, or the F7# note, or both at the same time. As with the previous activity, this program continuously transmits whichever note you leave un-commented. For now, we'll test one note; later we'll examine the other note as well as both notes played together.

- ✓ Enter and run One or Two Notes at a Time.bs2 as-is.
- ✓ After verifying that the note is clearly audible, reconnect the capacitor's positive lead to the circuit.

As before, the note's volume may drop considerably, but the signal displayed by the PropScope will be much cleaner.

```
' One or Two Notes at a Time.bs2
' Select whether to play D7#, or F7#, or D7# + F7# together.

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

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

DO                                           ' Main Loop

  FREQOUT 9, 60000, 2489                     ' Play 2489 Hz for 1 minute
  ' FREQOUT 9, 60000, 2960                   ' Commented, does not play
  ' FREQOUT 9, 60000, 2489, 2960             ' Commented, does not play

LOOP                                         ' Repeat main loop
```

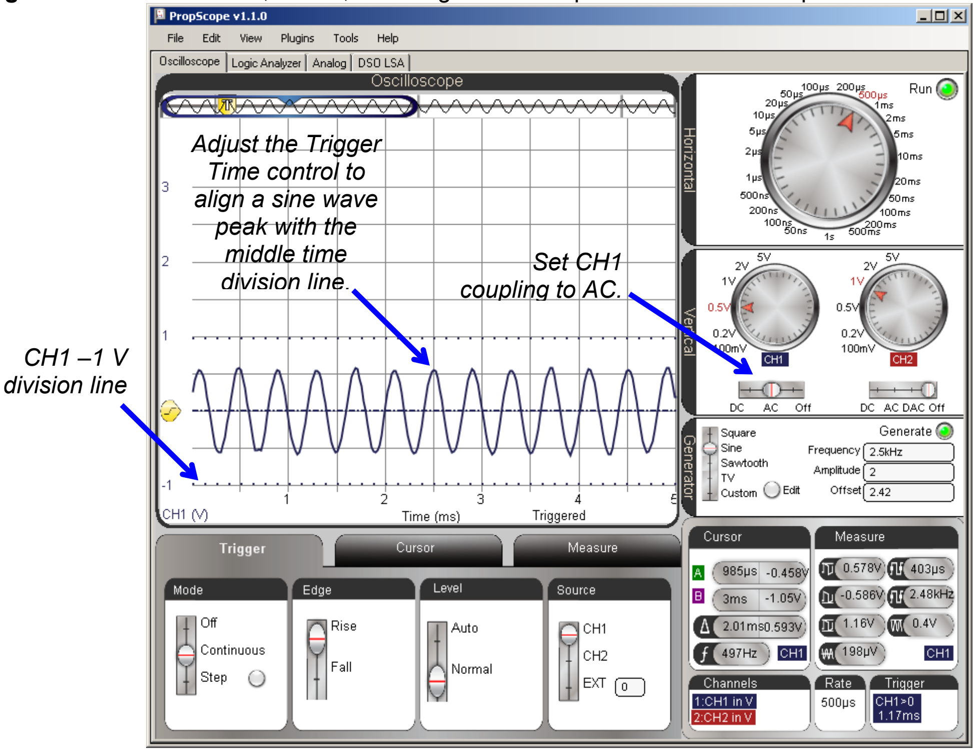
AC Couple the Signal

Each PropScope Vertical coupling switch has a setting that removes DC offset from a signal—it's called *AC coupling*. When you use this setting, it causes the sine wave voltages to swing above/below the ground line (0 V level), removing any offset you might see if you viewed it with DC coupling. In some measurements, DC offset doesn't matter, but other properties of the sine wave do. In those situations, AC coupling can simplify the measurements, especially if the two signals have different DC offsets.

Another important setting is the Trigger tab Level switch's Normal setting. This allows you to pick your own voltage level for triggering the display. This setting will be useful for aligning the trigger event to a high peak in the signal with two sine waves added together. Figure 7-19 shows the AC coupled D7# note from the BASIC Stamp with Normal (instead of Auto) trigger voltage level. It also shows the other settings we'll use to compare the individual frequencies against the frequencies added together.

- ✓ Click the Cursor tab's Horizontal and Vertical buttons to toggle them off.
- ✓ Oscilloscope dials: Horizontal = 500 µs/div, Vertical CH1 = 0.5 V/div
- ✓ Vertical Coupling Switches: CH1 = AC, CH2 = Off
- ✓ Click and drag the CH1 trace up/down and make the CH1 –1 V division line up with the bottom of the oscilloscope display.
- ✓ Trigger tab switches: Mode = Continuous, Edge = Rise, Level = Normal, Source = CH1.
- ✓ If needed, adjust the Trigger Voltage control to make the horizontal trigger crosshair line up with the CH1 ground (0 V) division line.
- ✓ Trigger Time control: Adjust to make one of the sine wave's peaks line up with the middle time division line. It doesn't matter where the vertical trigger time crosshair is, just make sure a peak in the sine wave is lined up with the middle time division line. See example in Figure 7-19.

Figure 7-19: FREQOUT 9, 60000, 2489 Signal AC Coupled in the Oscilloscope



Adjust the Trigger Time control to align a sine wave peak with the middle time division line.

Set CH1 coupling to AC.

CH1 –1 V division line

Frequency 2.5kHz
Amplitude 2
Offset 2.42

Cursor
A 985µs -0.458V
B 3ms -1.05V
Δ 2.01ms 0.593V
ƒ 497Hz CH1

Measure
0.578V 403µs
-0.586V 2.48kHz
1.16V 0.4V
198µV CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 500µs
Trigger: CH1>0 1.17ms

Compare a Dual Tone Signal to its Component Sine Waves

Two Notes Together.bs2 played each individual note, and then both notes together. What do you think the signal will look like when both notes are played together? Let’s use the PropScope to find out. First, let’s take pictures of the individual D7# and F7# notes and save them for comparison:

- ✓ Make a copy of the Oscilloscope screen for later reference by clicking the Edit Menu and selecting Copy Image.
- ✓ Paste it into a program called Paint (Start → All Programs → Accessories → Paint.)
- ✓ Save the file as 2489 Hz.bmp.

The next step is to modify One or Two Notes at a Time.bs2 so that it plays the F7# (2960 Hz), and then take a screen capture of that waveform.

- ✓ In the BASIC Stamp Editor, modify One or Two Notes at a Time.bs2 by placing an apostrophe to the left of **`FREQOUT 9, 60000, 2489`** to comment it out.
- ✓ Delete the apostrophe to the left of **`FREQOUT 9, 60000, 2960`** to uncomment it.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Use the Trigger Time control to line up one of the peaks of one sine wave with the middle time division line in the screen again.
- ✓ Take a screen capture and save it as 2960 Hz.bmp

Now that the individual notes have been examined and their screen captures stored, play both notes together. Then, examine the resulting signal and take a screen capture of it too.

- ✓ In the BASIC Stamp Editor, comment out **`FREQOUT 9, 60000, 2960`**.
- ✓ Uncomment **`FREQOUT 9, 60000, 2489, 2960`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Slide the Trigger Voltage control to make the horizontal crosshair line up near the top of the highest peak in the waveform, to keep it from scrolling.
- ✓ Use the Trigger Time control to line up the highest peak with the middle time division line. For an example, see the bottom waveform in Figure 7-20.
- ✓ Take a screen capture and save it as 2489 and 2960 Hz.bmp

Figure 7-20 conveys the basic idea of how adding two sine waves together works. (Actually, both sine waves are divided by 2 first and then added for the result at the bottom of the figure.) For example, a peak on the top sine wave added to a valley in the middle sine wave results in zero on the bottom waveform. The two values cancel each other out. Another example, two peaks in the component waveforms get added together to form the highest peak in the bottom waveform. Again, the BASIC Stamp adds ½ of the top waveform amplitude to ½ of the middle waveform's amplitude. That's why the bottom waveform's peak is not twice as high.

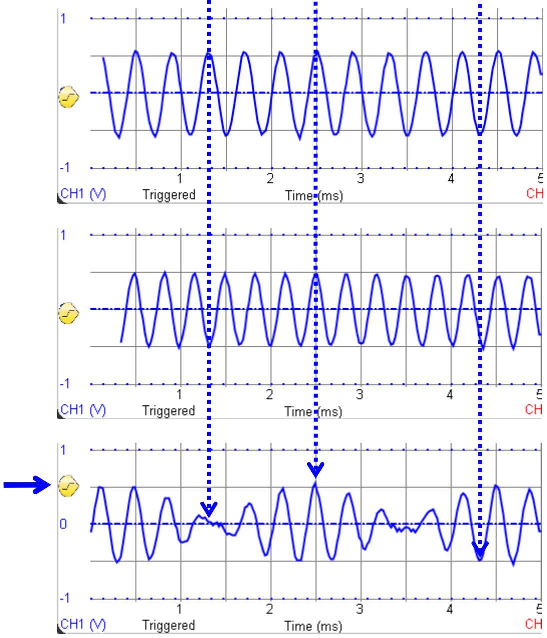
- ✓ Carefully examine Figure 7-20, and make sure you can see how ½ of each component sine wave is added together to make the resulting waveform that contains both musical notes.
- ✓ Try opening your screen captures, combining them together in a single document, and lining them to repeat the comparisons in Figure 7-20.

Figure 7-20: Two Sine Waves Added Together

A valley subtracts from a peak.

Peaks add together.

Valleys add together.



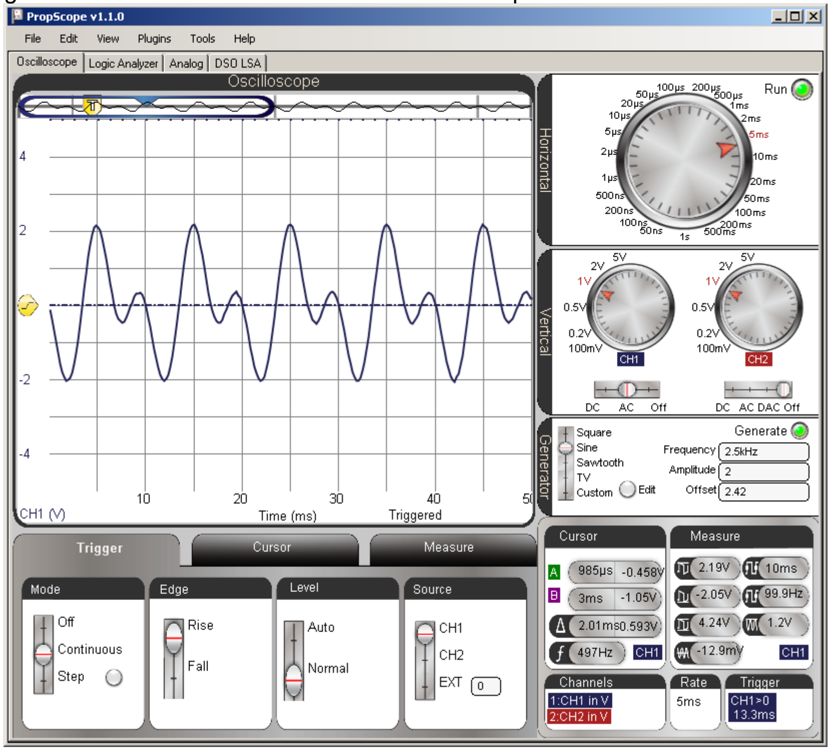
For two notes played together, adjust the Trigger Voltage control so that it's just below the top of the highest peak.

Your Turn: 100 and 200 Hz

Figure 7-21 shows an example of 100 and 200 Hz added together. These tones will not be audible with the piezospeaker, but you can still measure them with the PropScope.

- ✓ Modify One or Two Notes at a Time.bs2 for use with these two frequencies.
- ✓ Plot and screen capture the components.
- ✓ Plot and screen capture the sum.
- ✓ Examine the effect of the peaks and troughs adding together.

Figure 7-21: Waveform with 100 and 200 Hz Components



Frequency 2.5kHz
Amplitude 2
Offset 2.42

EXT 0

Cursor
A 985µs -0.458V
B 3ms -1.05V
Δ 2.01ms 0.593V
ƒ 497Hz CH1

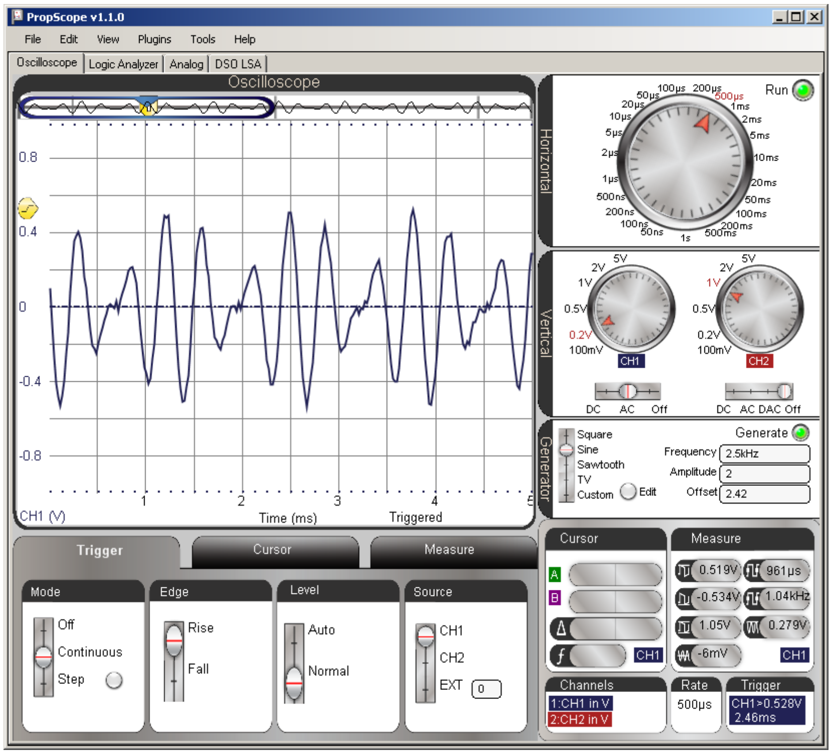
Measure
2.19V 10ms
-2.05V 99.9Hz
4.24V 1.2V
-12.9mV CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 5ms
Trigger: CH1>0 13.3ms

ACTIVITY #5: SINE COMPONENTS WITH A SPECTRUM ANALYZER

What are the frequencies of the sine wave components in the Figure 7-22 waveform? Here's a hint, they are not D7# and F7#. So, what are they? Could you use the cursors to figure it out? Maybe, but it might turn out to be difficult and time consuming. The answer to the question is that this is **`FREQOUT 9, 60000, 2349, 3136`**, which is the BASIC Stamp playing the D7 and G7 notes from Figure 7-12 on page 228.

Figure 7-22: Mystery Waveform



Frequency 2.5kHz
Amplitude 2
Offset 2.42
EXT 0

Measure: 0.519V, 961µs, -0.534V, 1.04kHz, 1.05V, 0.279V, -6mV, CH1
Channels: 1:CH1 in V, 2:CH2 in V
Rate: 500µs
Trigger: CH1>0.528V 2.46ms

A *spectrum analyzer* is a device that measures and displays the frequencies and relative amplitudes of a signal's sine wave components. The PropScope's analog tab has a built-in Spectrum Analyzer. Let's try it.

- ✓ Enter D and G Notes.bs2 into the BASIC Stamp Editor and load the program into the BASIC Stamp.

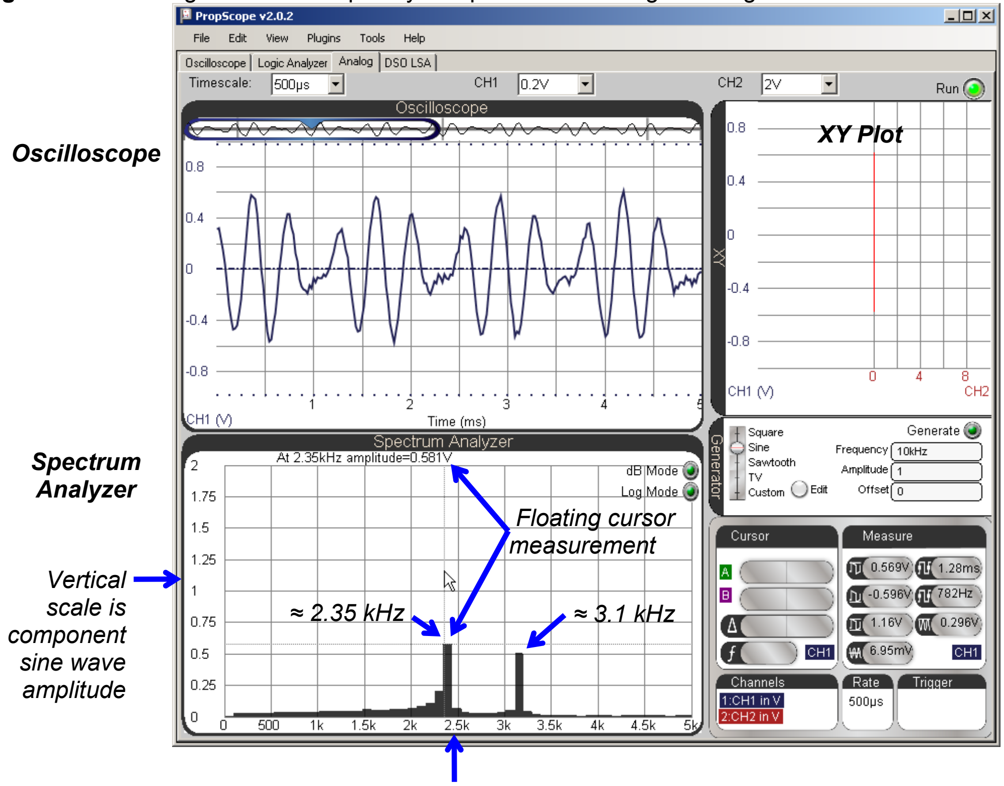
```
' D and G Notes.bs2
' Play D7 and G7 notes together.

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

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

DO                                          ' Main Loop

  FREQOUT 9, 60000, 2349, 3136              ' Play 2389 + 3136 Hz for 1 min.

LOOP                                        ' Repeat main loop
```

- ✓ View the waveform in your PropScope's Oscilloscope view using Figure 7-22 as a guide for settings.
- ✓ Click the Analog tab. It's in the same group with Logic Analyzer and Oscilloscope.

Figure 7-23 shows the Analog view, which includes three major tools, the now familiar oscilloscope, the Spectrum Analyzer, and XY Plot. The Spectrum Analyzer screen makes short work of determining the frequency components that might have looked somewhat confusing in the Oscilloscope screen.

In contrast to an oscilloscope, which plots voltage against time, a spectrum analyzer plots sine wave component amplitudes against frequency. The heights of the two vertical bars in the Spectrum Analyzer screen indicate the two sine wave component amplitudes, and their left-right positions place them over their frequencies. For example, the left bar in the spectrum analyzer has a height that lines up with about 0.46 V on the vertical scale, and is over the 2.35 kHz frequency on the horizontal scale. The bar to its right has a height that lines up with about 0.4 V on the vertical scale, and is over the 3.14 kHz frequency on the horizontal scale. This confirms what we know about D and G Notes.bs2. It transmits a signal that is the sum of two sine waves with frequencies of 2349 Hz and 3136 Hz. With the ability to indicate a waveform's sine wave component amplitudes and frequencies, the spectrum analyzer is quite a handy tool.

Figure 7-23: Analog View with frequency components of the signal in Figure 7-22



PropScope v2.0.2

Timescale: 500µs CH1 0.2V CH2 2V

At 2.35kHz amplitude=0.581V

Generator: Frequency 10kHz, Amplitude 1, Offset 0

Measure: 0.569V, 1.28ms, -0.596V, 782Hz, 1.16V, 0.296V, 6.95mV, CH1

Channels: 1:CH1 in V, 2:CH2 in V Rate: 500µs

Oscilloscope

Spectrum Analyzer

Vertical scale is component sine wave amplitude

XY Plot

Floating cursor measurement

≈ 2.35 kHz

≈ 3.1 kHz

Horizontal scale is frequency

Figure 7-23 also shows a floating cursor measurement. As you point your mouse at positions on the screen, the floating cursor measurement is displayed in the top-left of the Spectrum Analyzer screen. In the figure, the floating cursor's vertical crosshair is lined up with the left bar, and the measurement shows: At 2.35 kHz amplitude=0.581V.

✓ Use the floating cursor to check the amplitude and frequency of each of the bars in your spectrum analyzer.

Spectrum Analyzer Units: By default, the spectrum analyzer displays amplitude in terms of Vpp and frequency in terms of Hz. Both scales are linear. If you click the Log Mode button, the frequency units will increase logarithmically instead of linearly. If you click the dB Mode button, the amplitude will display as decibels instead of peak-to-peak volts.

Sine Waves in a Square Wave?

It turns out that some varieties of sine waves can be added together to create just about any signal you can display with your Oscilloscope. That also means that just about any signal you display in the Oscilloscope is composed of one or more sine waves, and the Spectrum Analyzer is your tool determining a signal's component sine waves. The PropScope's spectrum analyzer performs a Fast Fourier Transform (FFT) on the oscilloscope signal to determine its sine wave components.

A Fourier Series is a sum of sine waves that creates an arbitrary waveform. It is named after Joseph Fourier, the French mathematician and physicist who developed the technique of describing certain mathematical functions as sums of sine wave equations.

A Fourier Transform is the inverse of a Fourier Series. Given a mathematical function, a Fourier Transform resolves it into its Fourier Series of sine wave components.

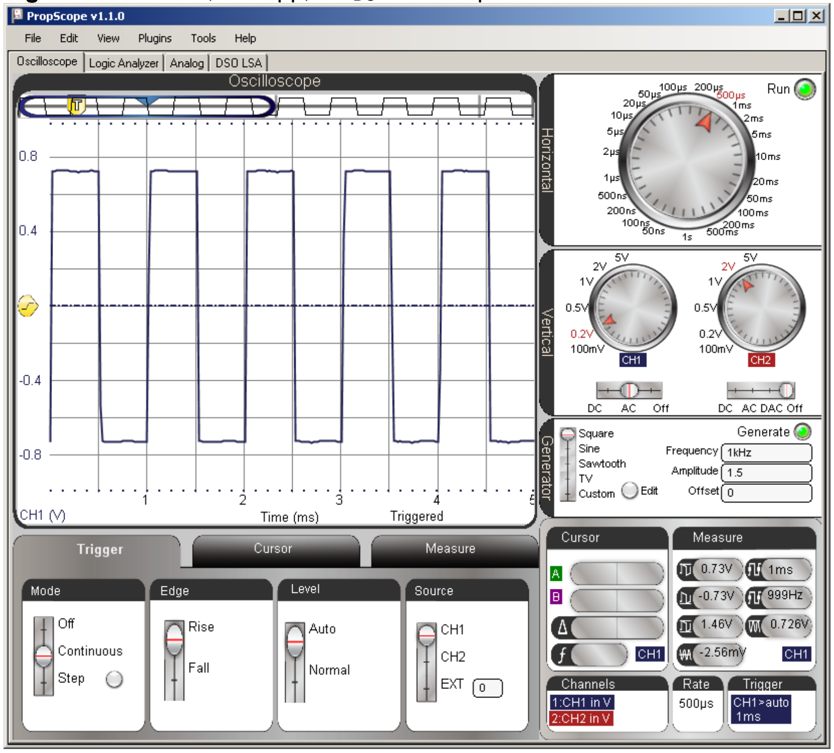
A Fast Fourier Transform is a widely used technique for determining a signal's sine wave components using fewer steps and less computing time.

Microcontrollers and other systems that exchange information with high/low signals have the potential to cause radio frequency (RF) interference. A radio transmitter applies a sine wave of a certain frequency to an antenna to broadcast. A radio receiver tuned for the transmitter's broadcasts detects them as sine wave voltage variations it gets from its own antenna. It turns out that binary signals like square waves contain many sine waves with many different frequencies. So, when a microcontroller applies a square wave to a wire (antenna), it has the potential to act like a radio transmitter that generates RF interference at many different frequencies. Many electronic devices have to be tested to ensure that they do not broadcast at certain frequencies with signals that are strong enough to be received as interference by nearby radio receivers.

To test this, the PropScope DAC Card's function generator can generate the square wave, and then the Spectrum Analyzer can display the frequency components.

- ✓ Set up your PropScope's hardware to transmit signals from the DAC Card's function generator to the CH1 probe. (Instructions and circuit for this were first introduced in the Set DC Voltages with the PropScope's DAC Card section starting on page 45.)
- ✓ Configure the Generator panel to make the DAC Card's function generator output transmit a 1 kHz, 1.5 Vpp, 0 V offset square wave. (You can use the Generator panel settings in Figure 7-24 as a guide.)

Figure 7-24: 1 kHz, 1.5 Vpp, 0 V_{DC} Offset Square Wave



Frequency 1kHz
Amplitude 1.5
Offset 0

Measure: 0.73V, 1ms, -0.73V, 999Hz, 1.46V, 0.726V, -2.56mV, CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 500µs
Trigger: CH1>auto 1ms

Next, even though the Analog view has variations on most of the controls you've been using in Oscilloscope view, it's better to set up the measurement in the Oscilloscope view and then switch over. Then, you can make minor adjustments in the Analog view to get the information you need.

For the clearest spectrum analysis, it's best to have four to eight cycles of the signal displayed in the Oscilloscope, and the voltage scale should make the waveform fill at least half of the total voltage amplitude the Oscilloscope can display. Along with the function generator settings, Figure 7-24 also shows an example of our square wave measured with CH1. It displays almost five full cycles and fills ¾ of the total +/– 1 V amplitude that the Oscilloscope can display with the CH1 vertical dial set to 0.2 V/div. Remember, these limits are indicated by the dotted lines, which are visible near the top and bottom of the Oscilloscope screen in Figure 7-24.

- ✓ Set the dials to: Horizontal = 500 µs/div, Vertical CH1 = 0.2 V/div.
- ✓ Set the Vertical coupling switches to: CH1 = AC, CH2 = Off.
- ✓ Drag the display up/down to position the ground line in the middle of the oscilloscope display.
- ✓ Set the Trigger tab's switches to: Mode = Continuous, Edge = Rise, Level = Auto, and Source = CH1.

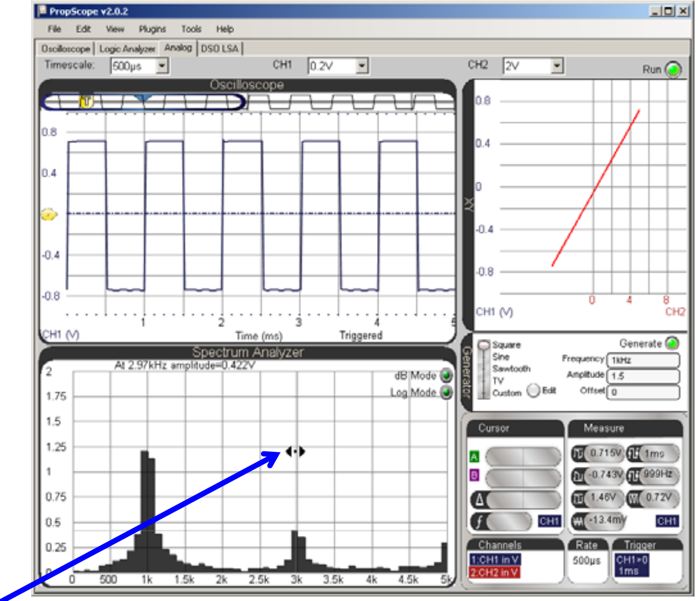
Figure 7-25 shows an example of how the Analog View should look when after you configure your Oscilloscope display and then click the Analog view tab. The Spectrum Analyzer shows several of the sine wave components that make up the square wave. These components are called *harmonics*. When a square wave is transmitted from one circuit to another along a wire, the wire can act like a small antenna, and broadcasts the harmonics. So, the 1 kHz square wave might be generating small amounts of radio interference at the 1 kHz, 3 kHz, 5 kHz, 7 kHz, 9 kHz and on up the frequency spectrum at odd multiples of the square wave's frequency. Note that as the frequency of the harmonic increases, its amplitude decreases.

- ✓ Click the Analog tab.

The Spectrum analyzer will probably display a horizontal frequency scale of 0 to 5 kHz. So only the first three sine wave harmonics will be visible. You can adjust the Spectrum Analyzer's frequency scale with your mouse. Just point at the Spectrum Analyzer screen, and then click, hold and drag either right or left. Dragging left increases the frequency scale; dragging right decreases it.

- ✓ To increase the frequency scale for viewing more harmonics, point at the Spectrum Analyzer screen with your mouse, then click, hold, and drag left.
- ✓ Adjust the Spectrum Analyzer plot so that it resembles the lower portion of Figure 7-25 with a frequency scale of 0 to 10 kHz.

Figure 7-25: Sine Wave Components in a Square Wave



Timescale: 500µs CH1 0.2V CH2 2V

At 2.97kHz amplitude=0.422V

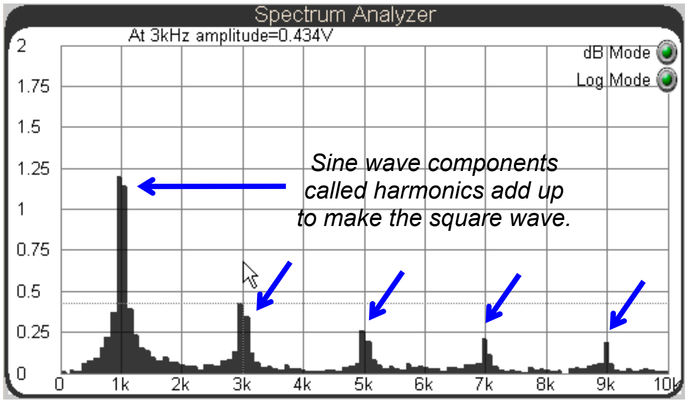
Frequency 1kHz Amplitude 1.5 Offset 0

0.715V 1ms -0.743V 999Hz 1.46V 0.72V -13.4mV

1:CH1 in V 2:CH2 in V 500µs CH1+0 1ms

Point at the Spectrum Analyzer screen with your mouse. Then, click, hold, and drag left until the Spectrum Analyzer's frequency scale displays from 0 to 10k.

The result should look like this.



At 3kHz amplitude=0.434V

Sine wave components called harmonics add up to make the square wave.

A square wave is made up of an infinite number of sine wave components, starting with a sine wave at the square wave's frequency called the *fundamental*. The next sine wave is smaller amplitude, and at three times the fundamental. It's called the *third harmonic*. The next higher frequency sine wave is called the *fifth harmonic*, and not surprisingly, it is five times the square wave's frequency, and has even smaller amplitude. This pattern continues for every odd multiple of the square wave's frequency.

Your Turn: Harmonics in a Sawtooth Wave

It takes a different set of sine waves to construct a sawtooth wave.

- ✓ Change the Generator panel's Function switch to Sawtooth.
- ✓ What sine wave component frequencies are present in a sawtooth wave that are not present in a square wave?

ACTIVITY #6: COMPARE TWO SINE WAVES

Circuits like filters and amplifiers that process sine waves can be analyzed by comparing a sine wave that enters the circuit's input against the one the circuit transmits at its output. The two sine wave attributes that tend to change at the circuit's output are amplitude and phase. Amplifiers might create a larger amplitude sine wave, and filters might make the sine wave a little or a lot smaller, depending on the frequency of the sine wave and certain properties of the filter. Filters also tend to introduce a delay in the output signal. There's no change in frequency, but the output sine wave's points all occur slightly later than they do on the input wave. This delay is called *phase shift*.

The RC circuit we have been using for the BASIC Stamp DAC voltages is a type of filter called a *low-pass filter*. Given an input sine wave, it will reduce the amplitude and shift the phase of the output signal. The amplitude change and phase shift depend on the combination of RC values and the frequency of the input signal. In this activity, you will measure amplitude and phase shift changes that an RC circuit introduces into a sine wave. Later, in Chapter 8, Activity #7, you will also use amplitude and phase comparisons to quantify properties of the filter.

The material in this activity uses certain concepts from Trigonometry.

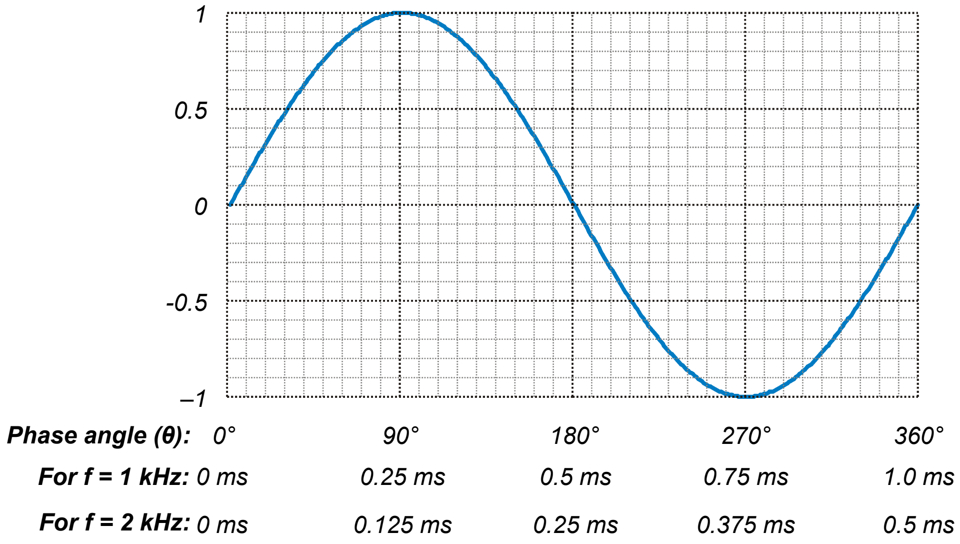
- ✓ If you have not yet studied trigonometry, read the sections in the list below. They are in the Sine Wave Math chapter of the Understanding Signals Supplement, which is available as a free PDF online from the Downloads section of www.parallax.com/go/PropScope.
  - o Degree Measurements
  - o Angles in Right Triangles and Simple Trigonometry Calculations
  - o Sine Calculations for Angles from 0° to 360°
  - o A Sine Wave – Sine Calculations from 0 to 360°
  - o Sine Wave Phase Angle and Phase Shift

Sine Wave Phase Angle and Phase Shift

The degree increments you are probably familiar with for measuring angles can also be used to measure how much of a sine wave cycle has elapsed. When applied to a sine wave, these degree measurements are called *phase angles*. The Greek letter theta θ is typically used to denote a phase angle, and phase angles are also typically expressed in degrees. Figure 7-26 shows some examples on a graph of sin(θ) vs. θ over 360°. Since a full sine wave cycle repeats every 360°, any fraction of a full cycle can be measured as a certain number of degrees.

Even though sine wave cycles take different amounts of time to repeat at different frequencies, phase angle measurements give us a way of describing how far into the cycle a certain time is for a sine wave at any given frequency. For example, 90° is ¼ of the way into the cycle. If the sine wave's frequency is 1 kHz, that happens at 0.25 ms into the sine wave. If the frequency is 2 kHz, that happens at 0.125 ms into the cycle. Regardless of the frequency, if the time is ¼ of the way into the sine wave cycle, the sine wave is at a phase of 90°.

Figure 7-26: Sine (θ) Vs. θ and Time for 1 and 2 kHz Signals



Phase angle measurements can be used to determine properties of circuits and/or the effects they have on signals. The next activity will use phase angle tests to determine the delay an RC circuit adds to a sine wave.

System Stability Phase angle measurements are also taken to ensure systems remain stable. An example of a stable system is public address (PA) system where somebody speaks into a microphone, and their voice is amplified and played by a loudspeaker for an audience. In a stable PA system, the amplifier is properly tuned and the speakers are oriented to minimize the sound energy that goes back into the microphone. So, the system amplifies the speaker's voice, and not what comes from the loudspeakers.

An example of an unstable system is when the person tries to say something into the PA system's microphone, and all the audience can hear is a loud, high-pitched whine. When the person tried to speak into the microphone, his/her voice was initially amplified by the amplifier and loudspeaker, but the microphone picked up the sound from the loudspeaker, and amplified it too. After that, the system gets stuck in an unstable feedback loop of amplifying what the microphone picks up from the loudspeaker.

The PA system is just one example of a system stability application. Ensuring system stability is part of designing many circuits and systems, including switching power supplies, automated ovens, and motor controllers. System stability has to be incorporated into many mechanical and electromechanical designs as well, including bridges and buildings, automobile cruise control, and aircraft autopilots.

You can convert a phase for any frequency to a degree measurement by keeping in mind that the ratio of θ to 360° is the same as the ratio of the time (t) to the cycle's period (T). From that, a phase angle equation is simple. Just multiply both sides of the equation by 360° to solve for θ.

$$\frac{t}{T} = \frac{\theta}{360^\circ} \quad \rightarrow \quad \theta = \frac{t}{T} \times 360^\circ$$

The equation $\theta = (t \div T) \times 360$ is saying that the phase angle θ is equal to the time t of a point on the sine wave, divided by its period T, and multiplied by 360°. For example, let's say we have a 2 kHz sine wave. That's a frequency, but we need to know its period T. Remember the period is the reciprocal of frequency, so you can use $T = 1 \div f$ to calculate the period.

$$T = \frac{1}{f} \quad \rightarrow \quad T = \frac{1}{2000\,Hz} = 0.0005\,s = 0.5\,ms$$

Let’s say we want to know the phase angle of a point that’s t = 0.375 ms into a cycle of the 2 kHz sine wave. Since we know its period is 0.5 ms, we have all the values we need to calculate the phase angle.

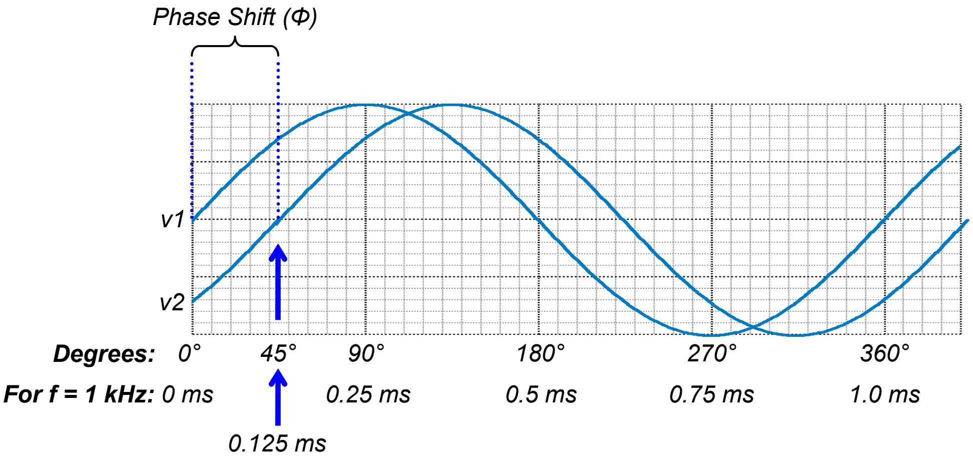
$$\theta = \frac{t}{T} \times 360^\circ = \frac{0.375\,ms}{0.5\,ms} \times 360^\circ = 270^\circ$$

> **TIP:** Since f = 1/T, you could replace t/T with t×f for this equation:
>
> $$\theta = t \times f \times 360^\circ$$
>
> This can save you the frequency to period conversion calculation. For example, If t = 0.375 ms and f = 2 kHz:
>
> $$\theta = 0.000375\,s \times 2000\,Hz \times 360^\circ = 270^\circ$$

- ✓ Take a look at Figure 7-26. Does the 0.375 ms match with 270° for a 2 kHz signal?
- ✓ Repeat this exercise for 0.125 ms in a 2 kHz signal and 0.25 ms in a 1 kHz signal. Make sure to check your results against Figure 7-26.
- ✓ Try 0.125 ms for a 1 kHz signal. The correct answer is 45°.

Some circuits introduce delays between input and output signals. For sine waves, this delay can be expressed as a phase angle measurement. For example, all the points in the 1 kHz signal labeled v2 in Figure 7-27 occur 0.125 ms later than the corresponding points in the v1 signal. Using phase angle math, we can calculate that v2 lags behind v1 by 45°, or that v1 leads v2 by 45°. In oscilloscope terminology, the expressions: “v1 leads v2” and “v2 lags v1” are common. The angle Φ is negative when measured from a reference sine wave to another that lags, or positive when measured to another sine wave that leads. So, the Φ from v1 to v2 is –45°, and from v2 to v1 is +45°.

Figure 7-27: Identical Sine Waves with Different Phases



Amplitude and Phase Angle Test Parts List

(1) Resistor – 10 kΩ (brown-black-orange)
(1) Capacitor – 0.01 μF (labeled 103)
(misc.) Jumper wires

Identifying Capacitors with Three-Digit Labels

While most larger electrolytic capacitors have their values clearly printed on their sides, smaller capacitors such as ceramic capacitors usually have just a three-digit code number representing their values. Here is how to interpret those code numbers:

(1) Take the first two digits as a two digit number.
(2) Multiply it by ten raised to the power of the third digit.
(3) Multiply that result by 1 pF, which is 1×10^{-12} F.
(4) Express in terms of one or more fractional multipliers (pico, nano, micro, etc…)

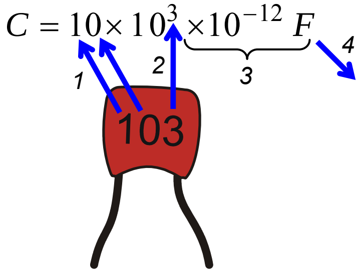
The top line in Figure 7-28 is an example of the three steps applied to the 0.01 μF capacitor in your kit.

Figure 7-28: 3-Digit Capacitor Label

Fractional Multipliers and Abbreviations

pico	(p)	$\times 10^{-12}$
nano	(n)	$\times 10^{-9}$
micro	(μ)	$\times 10^{-6}$
milli	(m)	$\times 10^{-3}$

$$C = 10 \times 10^{3} \underbrace{\times 10^{-12}\ F}_{3}$$



$$\begin{aligned} C &= 10 \times 10^{3} \times 10^{-12} \\ &= 10{,}000 \times 10^{-12} &&= 10{,}000\ pF \\ &= 10 \times 10^{-9} &&= 10\,nF \\ &= 0.01 \times 10^{-6} &&= 0.01\ \mu F \end{aligned}$$

Why step 4?

The reason you may need to express the result with “one or more” fractional multipliers is because you might go shopping for a 10 nF capacitor only to find that the electronics store doesn’t carry any, but they have a wide selection of 0.01 µF capacitors.

Amplitude and Phase Angle Test Circuit

We’ll use the circuit in Figure 7-29 and Figure 7-30 to examine how a circuit can change a sine wave’s amplitude and phase. The circuit is similar to the DAC circuit for BASIC Stamp D/A. The PropScope’s function generator will apply sine wave signals to the circuit’s input, and the PropScope’s Oscilloscope view will display the function generator signal as well as the output signal measured on CH1.

- ✓ Set up the CH2 probe hardware to be a function generator output. (See Set DC Voltages with the PropScope’s DAC Card on page 45 for step-by-step instructions.)
- ✓ Build the circuit in Figure 7-29, optionally using Figure 7-30 as a guide.

Figure 7-29: Schematic for Phase Angle Measurement Test

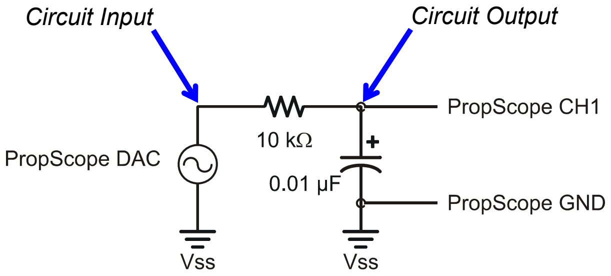
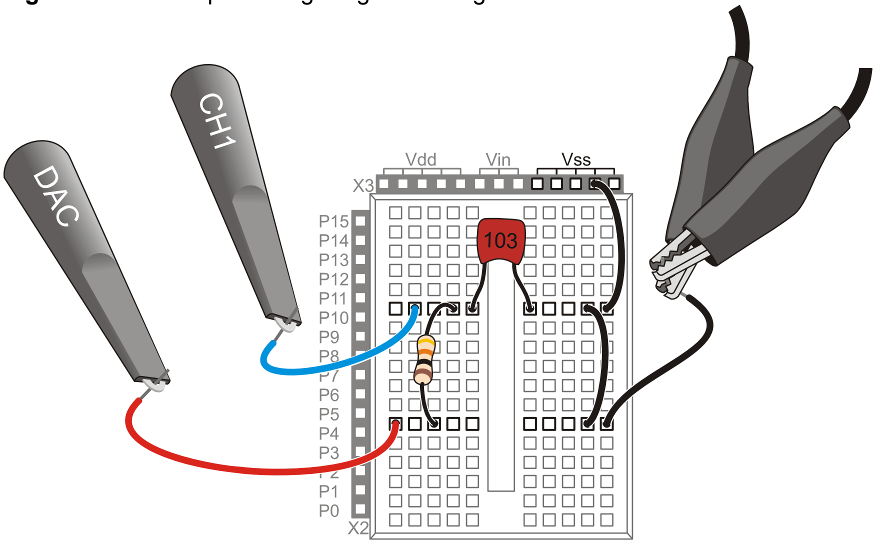


Figure 7-30: Example Wiring Diagram for Figure 7-29



The connection to the Vss socket is optional for this test.

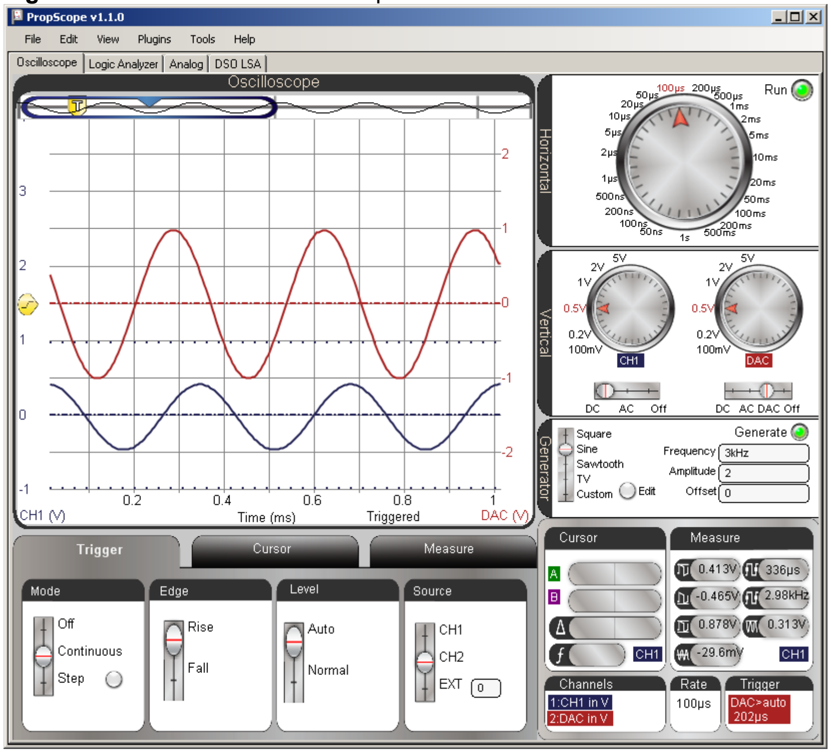
Since the BASIC Stamp is not interacting with this circuit, the jumper from the ground clips to the Vss socket is not necessary. The ground clips provide a connection to the USB port's ground connection, which is also your computer's ground connection.

If your measurements did involve interaction with the BASIC Stamp, connecting the ground clips to Vss would be necessary because the PropScope would need to share a common ground with your Board of Education/HomeWork board.

Amplitude and Phase Angle Test Measurements

Figure 7-31 shows a sine wave the PropScope DAC Card's function generator output applies to the RC circuit's input, along with the sine wave measured at the circuit's output. The function generator trace is shown as the red, upper CH2 trace, and the circuit's output is measured below it with the blue, lower CH1 trace. The two waves are positioned close together to highlight the differences in the output signal's amplitude and phase.

Figure 7-31: Visual Phase and Amplitude Check



Frequency 3kHz
Amplitude 2
Offset 0
EXT 0

0.413V 336µs
-0.465V 2.98kHz
0.878V 0.313V
-29.6mV

Channels: 1:CH1 in V, 2:DAC in V
Rate: 100µs
Trigger: DAC>auto 202µs

The voltage difference between the output sine wave's peaks and valleys are shorter than the input signal, so the amplitude is less. In electronics-speak, you could say, "the output signal is *attenuated*." Also, the output signal is the same frequency as the input signal, but the whole sine wave appears to be shifted slightly to the right in the Oscilloscope screen. This shift to the right shows that the circuit's voltage output is somewhat delayed, running behind the input by roughly 1/100th of a millisecond. Again in electronics-speak you could say, "The circuit introduced a phase delay into the output signal."

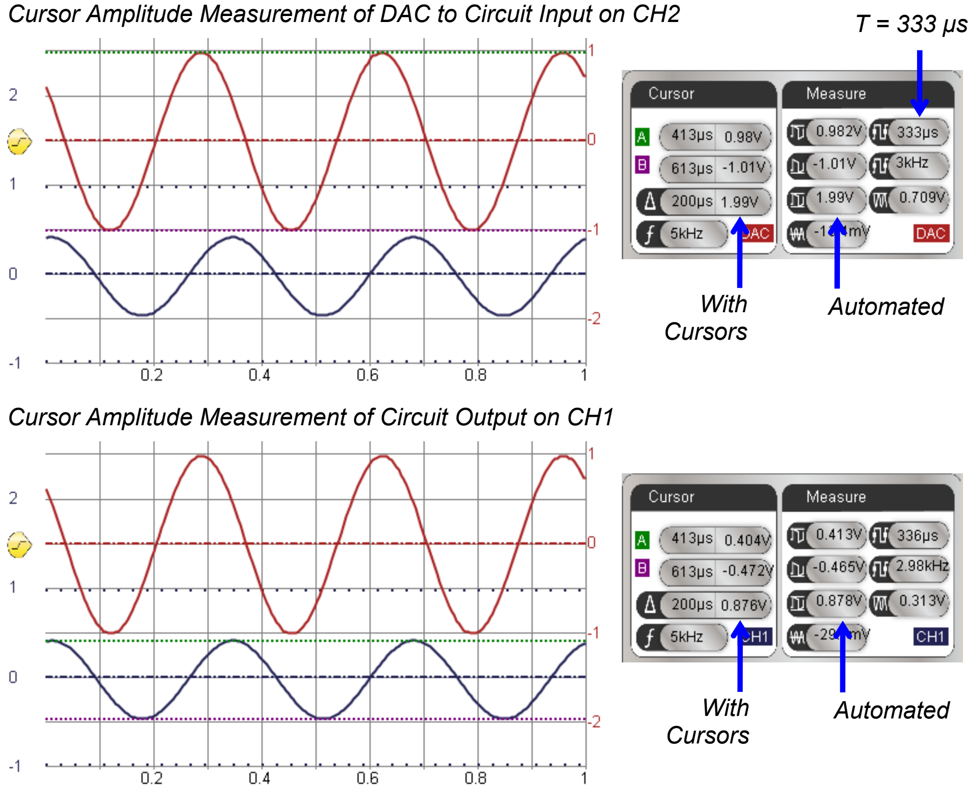
- ✓ Adjust the PropScope's dials to: Horizontal = 100μs/div, Vertical CH1 = 1 V/div, CH2 = 1 V/div.
- ✓ Coupling switches: CH1 = AC, CH2 = DAC.
- ✓ Trigger tab switches: Mode = Continuous, Edge = Rise, Level = Auto, Source = CH2.
- ✓ Trigger Time control: Align vertical crosshair with 2nd time division
- ✓ Generator panel: Function switch = Sine, Frequency = 3 kHz, Amplitude = 2 V, Offset = 0 V.
- ✓ Waveform Positions: Use Figure 7-31 as a guide. Position the red CH2/DAC trace slightly above the blue CH1 trace.

Figure 7-32 shows two different approaches to amplitude measurements, with cursors and with the Measure tab.

- ✓ Click the Cursor tab, and then click the Horizontal button to start the voltage cursors.
- ✓ Click the CH2 trace to set the lower-right Cursor and Measure displays to CH2 DAC output. (Or, click the CH1/CH2 label in the Cursor or Measure display to toggle between CH1 and CH2/DAC.)
- ✓ Position one voltage cursor at the tops of the CH2/DAC sine wave's peaks and the other at the bottoms of the valleys.
- ✓ Record the cursor Δ measurement, and compare it to the automated peak-to-peak value in the Measure display.
- ✓ Repeat for CH1.

Now we know that the circuit reduces the signal's amplitude by Figure 7-32 by almost ½: 0.867 ÷ 1.99 ≈ 0.436. Note that the cursor measurements closely match the automated measurements, so for sine wave amplitudes, automated measurements will speed up the process.

Figure 7-32: Amplitude Comparison



Cursor Amplitude Measurement of DAC to Circuit Input on CH2

T = 333 µs

With Cursors

Automated

Cursor Amplitude Measurement of Circuit Output on CH1

With Cursors

Automated

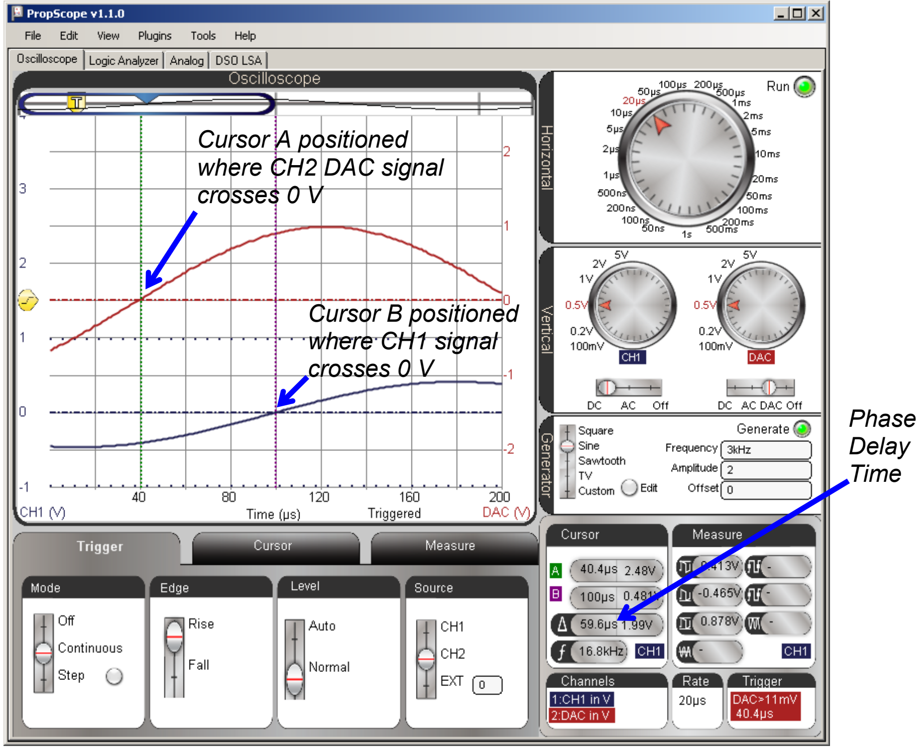
Next, let's examine the phase delay the circuit introduces by measuring the phase shift. First, we need to measure the time difference Δt between waveforms, and then express it as a number of degrees. The time difference can be measured by picking a point that's common to both signals. With the CH1 coupling set to AC, and CH2 with zero offset, a convenient reference point is where each wave crosses the zero volt line. Figure 7-33 shows the measurement. The Horizontal dial is adjusted to 20 µs for a closer look at the times that the waveforms cross their 0 V lines. Then, cursors are used to measure the time difference.

- ✓ Before adjusting the display, make a note of the signal's period T. (Example in top-right of Figure 7-32 is 333 µs.)
- ✓ Set the Horizontal dial to 20 µs/div.
- ✓ In the Trigger tab, change the Level switch from Auto to Normal.
- ✓ Adjust the Trigger Voltage control to align with the CH2 ground line.
- ✓ Adjust the Trigger Time control so that the CH2/DAC sine wave rises through the intersection of the ground line and the second time division.
- ✓ In the Cursor tab, click the Horizontal button to remove the voltage cursors, then click the Vertical button to start the time cursors.
- ✓ Adjust the green A time cursor so that it intersects with where the CH2/DAC sine wave crosses the CH2 ground (0 V) line.
- ✓ Adjust the purple B time cursor so that it intersects with where the CH1 sine wave crosses the CH1 ground (0 V) line.
- ✓ Get the phase delay time measurement from the Cursor display's Δ time difference calculation. The example in Figure 7-33 shows a Δt of about 59.6 µs.

With the measured Δt of 59.6 µs and a period T of 333 µs, the phase delay is:

$$\theta = \frac{\Delta t}{T} \times 360^\circ = \frac{59.6\,\mu s}{333\,\mu s} \times 360^\circ \approx 64.4^\circ$$

Figure 7-33: Determine Δt for Phase Comparison



Cursor A positioned where CH2 DAC signal crosses 0 V

Cursor B positioned where CH1 signal crosses 0 V

Phase Delay Time

Generator: Sine; Frequency 3kHz; Amplitude 2; Offset 0

Cursor: A 40.4µs 2.48V; B 100µs 0.481V; Δ 59.6µs 1.99V; ƒ 16.8kHz CH1

Measure: 0.413V; -0.465V; 0.878V; CH1

Channels: 1:CH1 in V; 2:DAC in V

Rate: 20µs

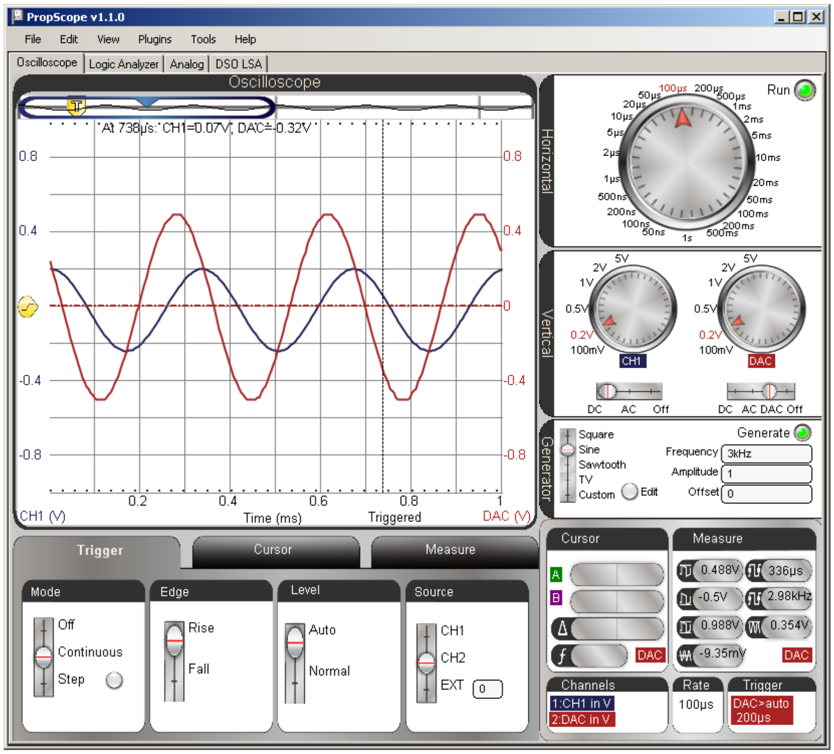
Trigger: DAC>11mV 40.4µs

In Figure 7-33, you could actually take the measurement by counting time divisions. The CH2/DAC sine wave crosses the ground line at the 40 µs time division line, and three time divisions later, the CH1 sine wave crosses the 100 µs division line. So the time delay from just counting time divisions is 60 µs, which is close enough to 59.6 µs for this type of measurement.

✓ Turn off the cursors and try counting time divisions to measure the phase delay time.

Figure 7-34 shows another common practice for comparing sine waves—superimposing one over another by positioning their ground lines at the same level in the Oscilloscope screen. You will use this setup for amplitude and phase measurements in Chapter 8, Activity #7.

Figure 7-34: Sine Waves Superimposed for Comparison



At 738µs: CH1=0.07V, DAC=-0.32V

Frequency 3kHz
Amplitude 1
Offset 0

0.488V 336µs
-0.5V 2.98kHz
0.988V 0.354V
-9.35mV DAC

1:CH1 in V
2:DAC in V
Rate 100µs
DAC>auto 200µs

Your Turn: What's Does a "Low-pass filter" Really Do?

A *low-pass filter* lets low frequencies through and takes away high frequencies by attenuating them (reducing their amplitudes). You will see more about this in the next chapter. For now, let's just examine the filter's behavior by trying a lower frequency and a higher frequency. The filter should allow the lower frequency to pass with less attenuation and phase delay. It should also remove more from the higher frequency signal and increase the phase delay.

- ✓ Repeat the amplitude and phase shift measurements in this activity for 1 kHz and 6 kHz. You may need to choose different Horizontal and Vertical settings to accommodate the different amplitudes and frequencies.
- ✓ Record the output signal amplitudes and phase delays for both frequencies.
- ✓ The pattern your measurements should fall into is: lower frequency → less output attenuation and phase delay, higher frequency → more attenuation and phase delay.

SUMMARY

In this chapter, sine waves were displayed with the PropScope and their basic properties of amplitude, frequency, DC offset and phase shift were studied. Practical applications of amplitude and frequency were demonstrated in a tone's volume and pitch. While DC offset doesn't necessarily affect the way a tone sounds in a piezospeaker, it can be problematic for audio speakers. DC offset can also be adjusted to position a sine wave so that it crosses a microcontroller's I/O pin threshold with each cycle, making it possible for the BASIC Stamp to count cycles and determine frequency.

Musical notes are sine waves of specific frequencies, and the PropScope was used to study the sine waves of individual notes as well as notes played together. In terms of sine waves, two notes played together is equivalent to adding two sine waves together, point-by-point. Given an arbitrary waveform, it can be difficult to determine its sine wave components. The spectrum analyzer is a tool that graphs each of a signal's component frequencies as a bar with a vertical height (amplitude) and horizontal position (frequency).

Two sine waves often have to be compared to each other for amplitude and phase differences. These comparisons are fundamental to numerous circuit analysis and system stability test procedures. As an example of amplitude and phase comparison, the PropScope's function generator was used to apply a signal to the input of an RC low-pass filter circuit. Then, the amplitude and phase of a sine wave at the circuit's output was compared to the function generated signal applied to its input. These tests were also performed at several frequencies to examine how the filter lets certain frequencies pass through while blocking others by reducing the signal amplitude at its output.

Chapter 8: RC Circuit Measurements

RESISTORS, CAPACITORS, AND RC CIRCUITS

A *resistor* “resists” the flow of current, and a *capacitor* has the “capacity” to store charge, kind of like a tiny, rechargeable battery. These two components form the building blocks for a variety of resistor-capacitor circuits, commonly called *RC circuits*.

The voltage waveform created by a capacitor charging or discharging through a resistor has a characteristic shape. Like a sine wave, this characteristic shape can be described by a mathematical equation. It’s called an *exponential decay* equation. In Activity #1, you will examine a graph of an exponential decay equation, plot a few sample points, and learn about key values called *time constants*. Then, in Activity #2, you will use the PropScope’s Oscilloscope view test a circuit’s time constants and use those values to make adjustments for better view of the voltage as the capacitor charges and discharges.

Certain sensors vary in either resistance or capacitance, which in turn affects how quickly the voltage in an RC circuit decays. In addition to using the PropScope to measure these decay times, you will also use the BASIC Stamp to automate these measurements, which could come in handy for remote sensing projects. In Activity #3, you will use RC decay to measure a potentiometer’s variable resistance, which indicates its knob’s position. In Activity #6, you will compare the linear decay traces from a light sensor that indicates brightness by the current it conducts to exponential RC decay traces.

Activity #4 and Activity #5 take a closer look at the signals in the RC circuits that were used for D/A conversion in earlier chapters. Activity #7 expands on the low-pass filter measurements that were introduced in Chapter 7 with a key value called *cutoff frequency*.

ACTIVITY #1: RC GROWTH AND DECAY MATH

Resistance (R) and capacitance (C) values can be incorporated into the exponential decay equation to predict and graph the rate at which voltage increases or decreases as a capacitor is charged or discharged through a resistor. For each RC decay graph, there is a point at which x = 1, and at that point the voltage is about 63.2% of the way to its final value. Viewed with an oscilloscope, this point designates 1 *time constant* on the oscilloscope’s time axis. Time constants have many uses. For example, they can be used to determine resistance and capacitance values in the circuit. They can also be used to calculate the amount of time it takes to get close enough to its final decay value for practical applications (even though it never reaches that point mathematically). Last, but not least, RC time constants are also used to predict how RC filters will reduce the amplitudes of sine waves at certain frequencies.

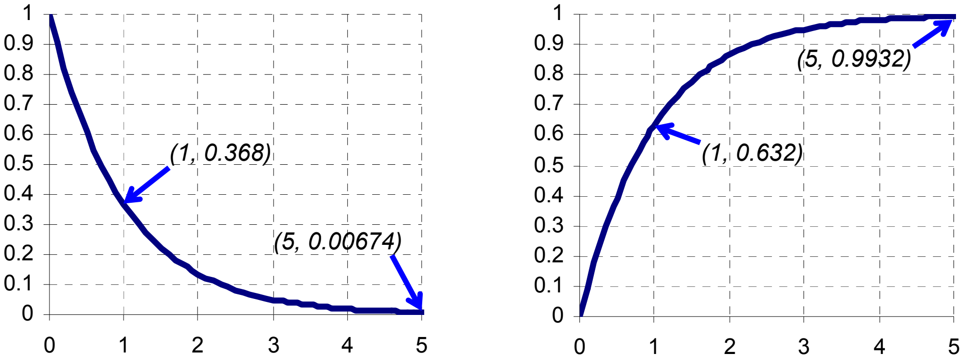
This activity introduces the exponential decay equation, and shows how the values of R and C are used determine an RC circuit's time constant.

RC Decay Math and Graphing Exercise

The equations in question are related to the *natural logarithmic constant*, *e*. The value of *e* is approximately 2.718. Variations on the *exponential growth equation* $y = e^{x}$ are best known for their use in modeling population growth. Variations on the *exponential decay equation* $y = e^{-x}$ also have many uses, and one of them is to describe how voltage changes as a capacitor loses or gains charge through a resistor. When applied to capacitor as it charges and/or discharges through a resistor, exponential decay is called *RC decay*.

The left side of Figure 8-1 shows a graph of $y = e^{-x}$. This graph shows how a capacitor's voltage behaves as it loses its charge through a resistor. When x = 1, $y = e^{-1} \approx 0.368$. You could also say that when x gets to 1, y has decayed to 36.8% of its starting value, or it's 63.2% of the way to its final value. By the time x reaches 5, the y value is only 0.00674, which is less than 1% of its starting value. The graph on the right is $y = 1-e^{-x}$, and it describes a capacitor accumulating charge through a resistor. The corresponding 63.2% level when x = 1 is also important, along with the y value when x = 5, which is within 1% of its final value of 1.

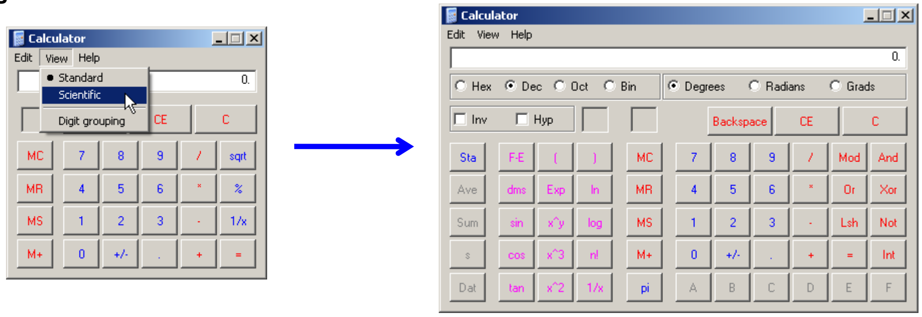
Figure 8-1: Graphs of $y = e^{-X}$ and $y = 1-e^{-X}$



Let’s use the Windows Calculator to verify a few values from the graphs in Figure 8-1.

- ✓ Run Windows Calculator by clicking Start → All Programs → Accessories → Calculator.
- ✓ If your calculator looks like the one on the left in Figure 8-2, click View and select Scientific.

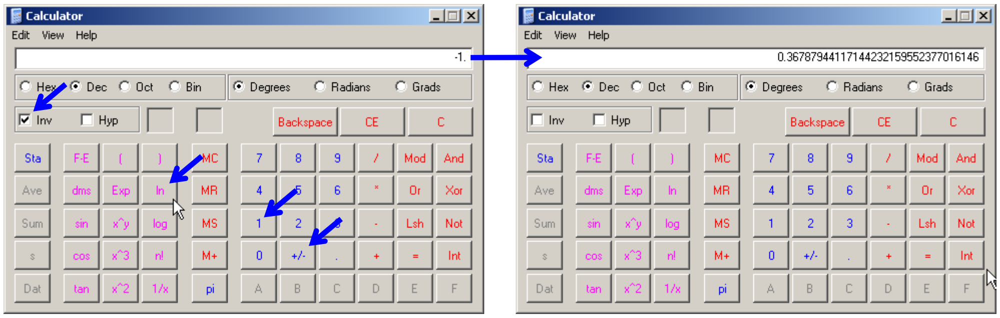
Figure 8-2: Windows Calculator



Most calculators have an e^x button you can use to calculate the points on $y = e^{-x}$ graphs, and it’s usually shared with the ln function, which tells you what x is in e^x for a given value. To save button space, the Windows Calculator has the ln key and an inverse (Inv) checkbox that you can click to make the ln key calculate e^x. Here is how to calculate e^{-1} with the Windows calculator’s ln button:

- ✓ Click the 1 button.
- ✓ Click +/– to make it –1.
- ✓ Click the Inv checkbox.
- ✓ Click the ln button.
- ✓ The calculator will display the result of e^{-1}.
- ✓ Calculate the value of e^{-5}.
- ✓ Try e^{-2}, e^{-3} and e^{-4}, and compare them to the points in the graph on the left hand side of Figure 8-1.

Figure 8-3: Calculate e^{-1} with the Windows Calculator


Calculator
Edit View Help
Backspace CE C
Sta F-E ( ) MC 7 8 9 / Mod And
Ave dms Exp ln MR 4 5 6 * Or Xor
Sum sin x^y log MS 1 2 3 - Lsh Not
s cos x^3 n! M+ 0 +/- . + = Int
Dat tan x^2 1/x pi A B C D E F

-1.

○ Hex ● Dec ○ Oct ○ Bin | ● Degrees ○ Radians ○ Grads

☑ Inv □ Hyp

0.36787944117144232159552377016146

○ Hex ● Dec ○ Oct ○ Bin | ● Degrees ○ Radians ○ Grads

□ Inv □ Hyp

You can create similar graphs to predict a voltage (v) as it decays from a certain starting value down to zero volts over time (t) with this version of the RC decay version of the exponential decay equation:

$$v = V_I \times e^{-t/RC}$$

V_I is the initial or starting voltage, and R and C are the values of the resistor and capacitor. The value RC in $v = V_I e^{-t/RC}$ is called the *RC time constant*, and is expressed as the Greek letter tau (τ)—rhymes with "saw"—but starting with a t. So, you may see the decay expressed as $v = V_I e^{-t/\tau}$. The value of $\tau = R \times C$ is in units of seconds.

How did you get seconds out of resistance and capacitance?

Recall that the ohm, Ω, is the unit of resistance, the R in RC time constant. Also, recall that the farad, F, is the unit of capacitance, C. Well, it turns out that F = s/Ω. If you want to know *why* this is so, keep studying physics and electronics—it's fascinating stuff! For now, just remember that τ = R×C is in units of seconds.

Another factoid to memorize is that the voltage decays to about 36.8% of its initial value in one τ time constant, and for a reminder, just calculate e^{-1} with your calculator.

Let's say that C = 1 µF and R = 1 kΩ. Then, R × C = 1 kΩ × 1 µF, which is 1,000 × 0.000001 = 0.001. If we assume the capacitor gets charged to 4 V before being allowed to decay through the resistor, which would make V_I = 4 V, the equation would be:

$$v = 4 \times e^{-t/0.001}$$

Figure 8-4 shows a graph of this equation. At 1 ms, the voltage has decayed to $4 \times e^{-0.001/0.001} = 4 \times e^{-1} = 4 \times 36.8\% = 1.47$ V. By the time the decay reaches the 5 ms mark, it is close enough to its final voltage to consider completely discharged.

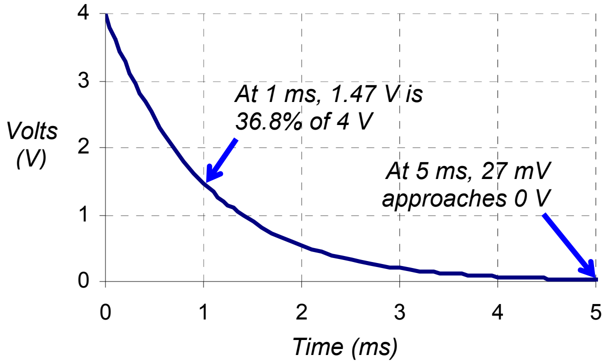


Figure 8-4
RC Voltage Decay Graph

…for R×C = 0.001.

ACTIVITY #2: RC GROWTH AND DECAY MEASUREMENTS

In this activity, you will build an RC circuit, and use a square wave from the PropScope's function generator to alternately charge and discharge the RC circuit's capacitor. To get good view of the RC decay curves as the capacitor's voltage increases and decreases, you will use the RC time constant from the previous activity to calculate an optimal square wave frequency and the oscilloscope's time division settings. You will also take measurements to verify the previous activity's RC time constant calculations.

RC Time Constant Test Circuit Parts

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

RC Time Constant Test Circuit

Figure 8-5 shows a schematic of the RC decay test circuit and Figure 8-6 shows an example wiring diagram.

- ✓ If your red-marked probe is not already connected to the DAC Card's function generator BNC connector, do that now (see Figure 2-16, page 46 for help).
- ✓ Connect the DAC probe to the circuit input, shown in Figure 8-5 and Figure 8-6.
- ✓ Connect the blue-marked CH1 probe to the RC circuit's output.

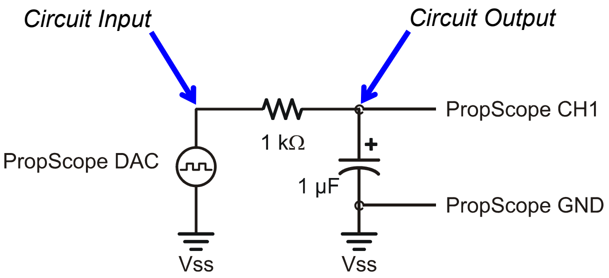
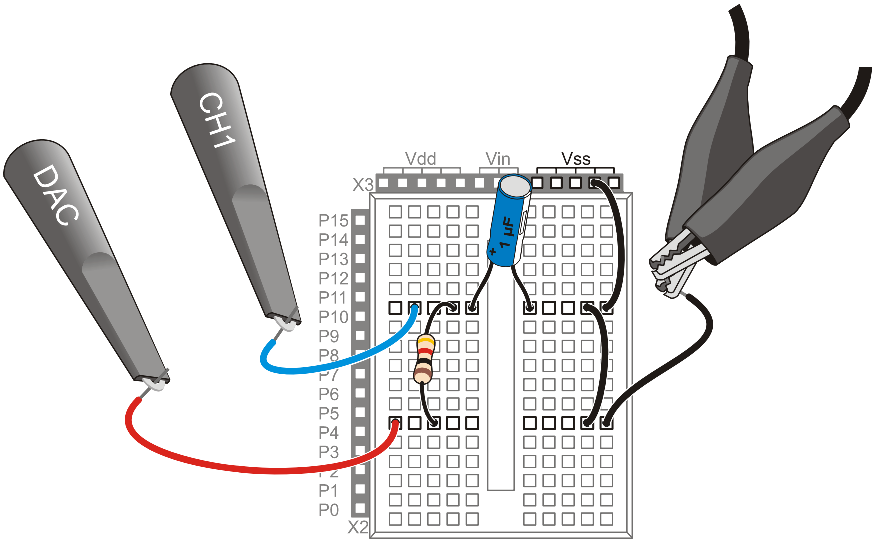


Figure 8-5
RC Decay Test Circuit

…with R×C = 0.001.

Figure 8-6: Wiring Diagram Example of Figure 8-5



Again, the connection to the Vss socket is optional for this test.

Since the BASIC Stamp is not interacting with this circuit, the jumper from the ground clips to the Vss socket is not necessary. The ground clips provide a connection to the USB port's ground connection, which is also your computer's ground connection.

If your measurements did involve interaction with the BASIC Stamp, connecting the ground clips to Vss would be necessary because the PropScope would need to share a common ground with your development board.

RC Time Constant Predictions

The decay time tests involve a square wave that will charge and discharge the RC circuit's capacitor. The first task is to figure out a good frequency for the square wave. Since R = 1 kΩ, and C = 1 μF, the RC time constant τ is:

$$\begin{aligned}\tau &= R \times C \\ &= 1\ \text{k}\Omega \times 1\ \mu\text{F} \\ &= 1{,}000 \times 0.000001\ \text{s} \\ &= 0.001\ \text{s} \\ &= 1\ \text{ms}\end{aligned}$$

The square wave's high signal has to last at least 5 time constants (5τ) for the capacitor to charge, and the low signal also needs to last 5τ for the capacitor to discharge. So, the minimum value of one signal cycle period (T_{min}) to the RC circuit's input has to last for a total of ten RC time constants:

$$T_{min} = 10\tau = 10 \times 1\ \text{ms} = 10\ \text{ms}$$

This means the input frequency has to be at most the reciprocal of T_{min}:

$$\begin{aligned}f_{max} &= 1 \div T_{min} \\ &= 100\ \text{Hz}\end{aligned}$$

For a first view, it's always good to see at least two full cycles, which means that the display has to be two cycles wide:

$$\text{Oscilloscope screen width}_{min} = 2 \times T_{min} = 2 \times 10\ \text{ms} = 20\ \text{ms}$$

Remember that the Horizontal dial specifies the width of a time division, which is 1/10th of the Oscilloscope screen's width, so:

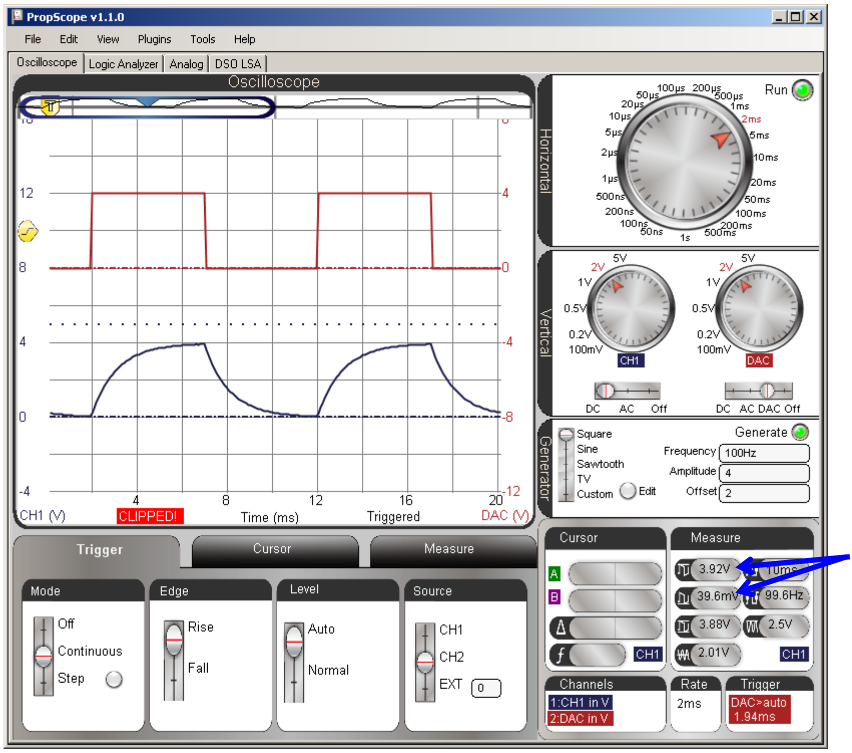
$$\text{Horizontal dial setting}_{min} = \text{Oscilloscope screen width}_{min} \div 10 = 20\ \text{ms} \div 10 = 2\ \text{ms}$$

Examine the Capacitor Voltage as it Charges and Discharges

Figure 8-7 shows two cycles of the square wave the function generator applies to the RC circuit along with the RC decays at the circuit's output.

- ✓ Generator panel: Function switch = Square, Frequency = 100 Hz, Amplitude = 4 V, Offset = 2 V.
- ✓ Dials: Horizontal = 2 ms/div, Vertical CH1 = 2 V/div, CH2 = 2 V/div.
- ✓ Vertical coupling switches: CH1 = DC, CH2 = DAC
- ✓ Trigger tab: Mode = Continuous, Edge = Rise, Level = Auto, Source = CH2.
- ✓ Trigger Time control: Set to 1st time division line
- ✓ Position the red CH2/DAC trace above the blue CH1 positive/negative RC decay trace.

Figure 8-7: 5 Time Constant Capacitor Charge and Discharge



High and low peak voltages

In Figure 8-7 the upper, red CH2/DAC trace shows a square wave that spends 5 ms at 4 V, then 5 ms at 0 V. This is the signal the function generator applies to the RC circuit's input. With each voltage level, current flows through the resistor and either charges or discharges the capacitor during each 5 ms time period. The lower, blue CH1 trace shows the capacitor voltage's response, which alternates directions of RC decay curves. Since the function generator voltage spends 5 ms, which is 5 ×τ, in each state, the capacitor should charge to almost 4 V and discharge to almost 0 V.

In the Oscilloscope screen, the CH1 voltage charges so close to 4 V and discharges so close to 0 V that it's difficult to tell it hasn't actually reached those values. Since the general guideline for practical applications is that 5 time constants is enough to consider the capacitor fully charged or discharged, the fact that the signal looks so close to its final voltages is a good sign. We can see from the high and low peak voltages in the Measure display that the capacitor charged to 3.92 V, then discharged to 39.6 mV.

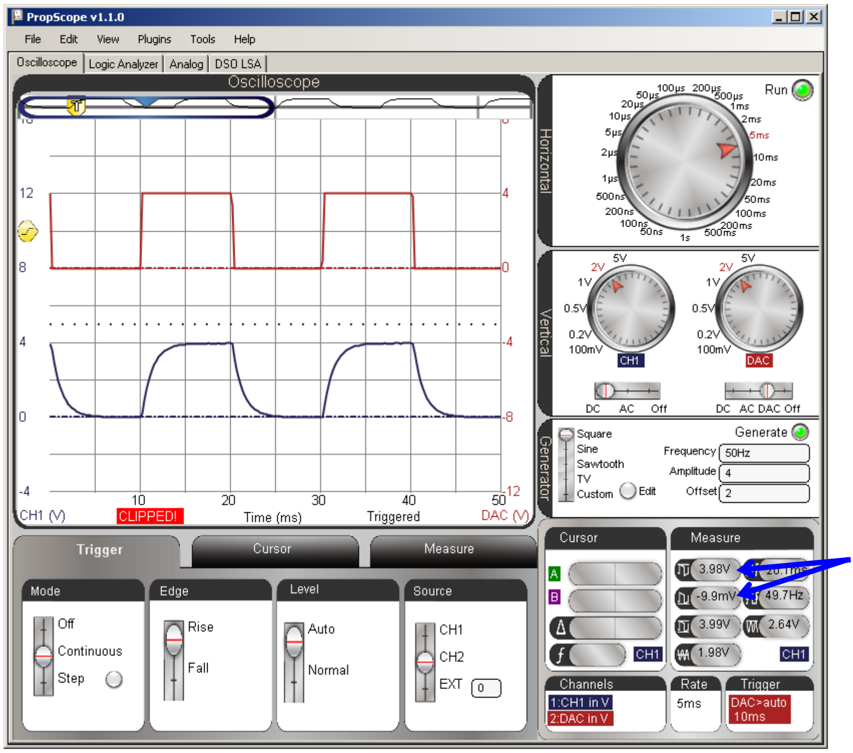
We also know from Activity #1 that the capacitor's voltage should charge to about 99.32% of the voltage applied after 5 time constants that it should decay to 0.674% of that voltage after 5 time constants. So, we'd expect the voltage to grow to 0.9932 × 4 V ≈ 3.97 V and decay to 0.00674 × 4 V ≈ 27.0 mV. These measurements are in the right ballpark to confirm that the circuit is working as expected.

- ✓ Click the CH2 trace to set the Measure panel to display its values. (Or click the CH1/CH2 label in the Measure display to toggle between measurements for the two traces.)
- ✓ Make a note of the CH2/DAC high and low peak voltages.
- ✓ Click the CH1 trace (or the CH2 label in the Measure display) to switch to displaying values for CH1, and make notes of the high and low peak voltages. Examples are shown in Figure 8-7.
- ✓ Compare them to the predicted 5τ values of 0.674% and 99.32%. For example, 0.674% of 4 V is 4 × 0.00674 = 0.02696 ≈ 27 mV.
- ✓ Optional: The CH2/DAC signal is an ideal value. You can probe the CH2 probe with the CH1 probe to find the actual measured voltage the function generator applies to the circuit's input, and then compare those to the measured peak voltages at the circuit's output.

Let's try adjusting to cycle high/low times that last 10τ (ten RC time constants) instead of 5 τ to see if anything interesting happens. For this, we'll want twice the signal period, which would be half the frequency. To maintain two cycles in the Oscilloscope display, the time per division will also have to be doubled, or in the case of the PropScope, adjusted from 2 ms/div to the next larger increment, which is 5 ms/div. Figure 8-8 shows the new view. In this particular example, the high peak in the Measure display is now about 3.98 V, which is 20 mV from 4 V, and the low peak is at almost –10 mV. As before, these values are still "in the right ballpark."

- ✓ Adjust the Horizontal dial to 5 ms/div.
- ✓ Adjust the Generator panel's Frequency field to 50 Hz.
- ✓ Adjust the Trigger Time control to the 2nd time division line.
- ✓ Repeat the input vs. output signal high and low peak measurement comparisons. How different are your results after ten RC time constants (10τ).

Figure 8-8: Square Wave Input and Increasing/Decreasing RC Decay Output



Frequency 50Hz
Amplitude 4
Offset 2

Measure: 3.98V, 20.1ms, -9.9mV, 49.7Hz, 3.99V, 2.64V, 1.98V, CH1

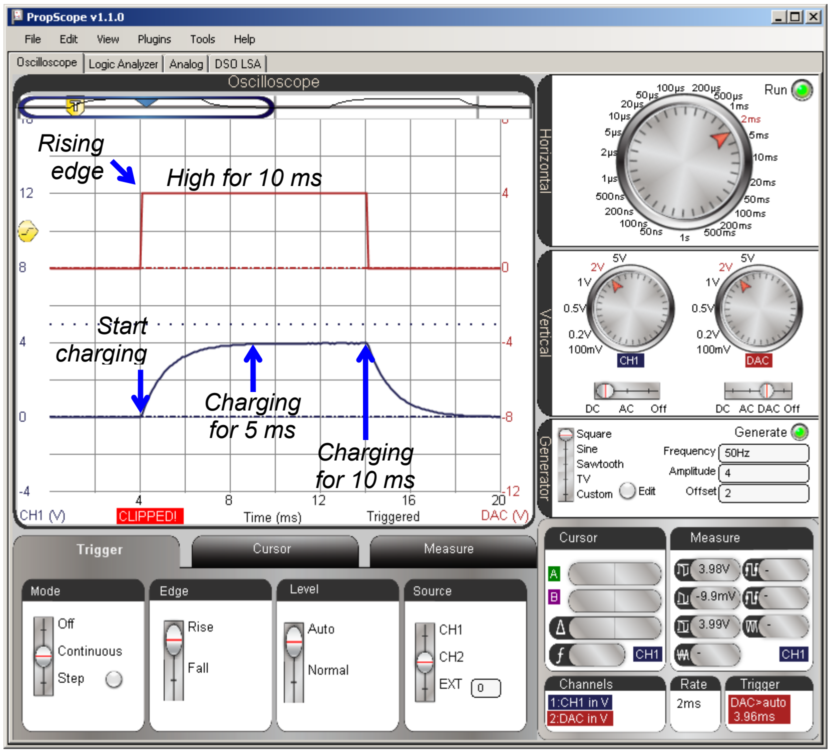
Channels: 1:CH1 in V, 2:DAC in V
Rate: 5ms
Trigger: DAC>auto 10ms

High and low peak voltages

Figure 8-9 zooms in to 2 ms/division with the horizontal scale for a closer look at voltage as the capacitor charges. Since the Trigger Time control is set to the 2nd time division line, the positive edge of the CH2 function generator signal lines up with that division line, and its negative edge lines up with the 7th time division line. Between 5 and 10 ms into the high signal, the CH1 RC circuit output voltage changes very little, if any. This makes sense since 5 ms is the 5 τ mark, and the capacitor should be charged to 99.32% of the applied voltage. At that point, there is very little room for it to increase its charge during the remaining 5 ms.

✓ Change the Horizontal dial to 2 ms/div for the view in Figure 8-9.

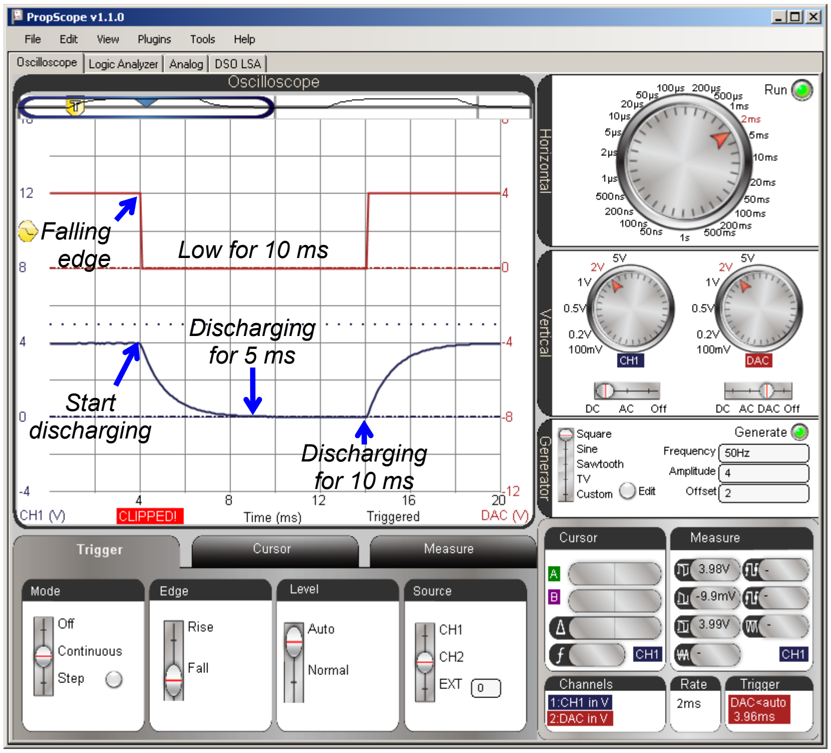
Figure 8-9: Charging the Capacitor for 10 ms



Here is a trick to remember. You don't have to move the screen back and forth to switch from the view of the capacitor's voltage response as it charges to its voltage response as it discharges. Instead, just change the Trigger Edge from Rise to Fall. Your display should then resemble Figure 8-10. With that change, it causes the function generator signal's negative edge to line up with the 2nd time division. The 10 ms of the function generator's low signal becomes visible in CH2, and the RC circuit's output voltage response as the capacitor discharges becomes visible in CH1.

- ✓ Change the Trigger Edge setting from Rise to Fall. Your view should now resemble Figure 8-10.

Figure 8-10: Toggle Trigger Edge from Rise to Fall to See the 10 ms Discharge



Falling edge

Low for 10 ms

Discharging for 5 ms

Start discharging

Discharging for 10 ms

Frequency 50Hz

Amplitude 4

Offset 2

3.98V

-9.9mV

3.99V

Channels 1:CH1 in V 2:DAC in V

Rate 2ms

Trigger DAC<auto 3.96ms

RC Time Constant Measurements

In an RC decay trace, if 1 time constant has elapsed, the voltage will have decayed from 100% to 36.8%. The converse is also true: if the voltage has decayed to the 36.8% level, one time constant has elapsed. With this in mind, the trick to measuring an RC circuit's time constant from its decay response is to set a horizontal voltage cursor at a level that's 36.8% above the final voltage. Then, use the time cursors to measure from the start of the decay to the point where the voltage cursor that's at the 36.8% level intersects with the trace.

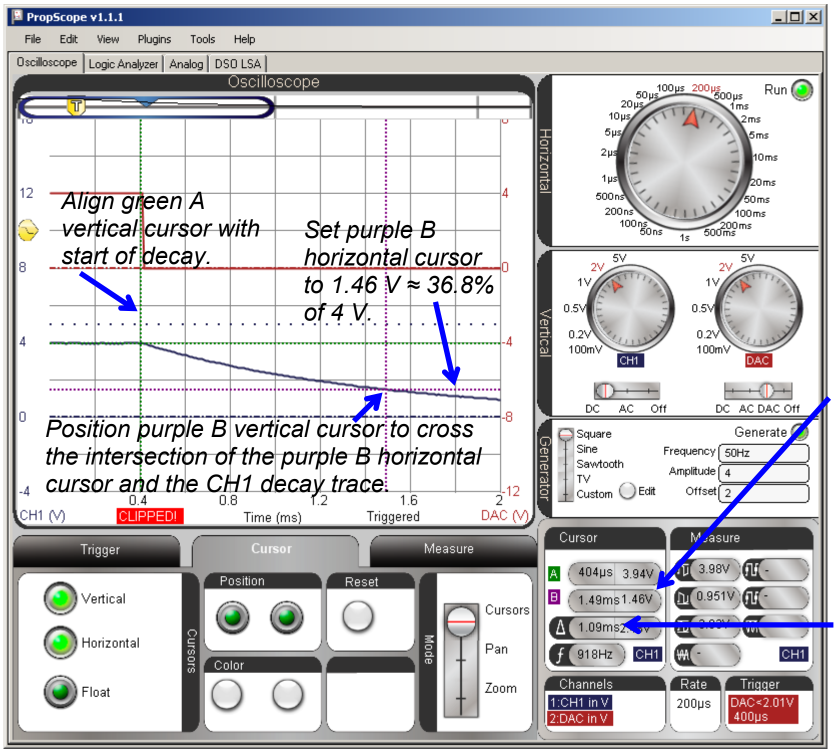
Figure 8-11 shows an example with the Horizontal dial adjusted to 200 µs/div for a zoomed-in view of that first one millisecond time constant's worth of decay. The vertical A time cursor is lined up with the start of the decay. Then, the horizontal B voltage cursor is set to as close as possible to 1.47 V, which is 36.8% of 4 V. You can use the B voltage level measurement in the Cursor display to know the cursor's voltage as you position it to 1.47 V. Next, the time from the start of the decay to the 36.8% level is measured by aligning the vertical B time cursor with the intersection of the decay trace and the horizontal B voltage cursor. The time measurement result will appear in the Measure Display's Δ field.

> **Horizontal Cursors** are positioned at voltage levels and measure voltage differences. They are commonly referred to as "voltage cursors."
>
> **Vertical Cursors** are positioned at times and measure time differences. They are commonly referred to as "time cursors."
>
> **Horizontal and Vertical Cursors** were introduced in **Chapter 3**, **Activity #5**.

- ✓ Set the trigger Level to Normal.
- ✓ Make sure the Trigger Voltage control is 2 V on the CH2 vertical scale (in the middle of the CH2/DAC square wave in Figure 8-11).
- ✓ We want to take a close look at that first millisecond of decay, so adjust the Horizontal dial to 200 µs/div.
- ✓ Multiply 4 V by 36.8%, the result should be about 1.47 V.
- ✓ Line the green A vertical cursor up with the start of the decay.
- ✓ Set the purple B horizontal cursor as close as possible to 1.47 V. Use the B cursor voltage measurement in the Cursor display as a guide.
- ✓ Position the purple B vertical cursor so that it crosses the intersection of the B horizontal cursor and the CH1 trace.

- ✓ The Cursor display's Δ field will show the time difference between the two vertical cursors, which is τ for your circuit.

Figure 8-11: Measuring the RC Time Constant



Align green A vertical cursor with start of decay.

Set purple B horizontal cursor to 1.46 V ≈ 36.8% of 4 V.

Position purple B vertical cursor to cross the intersection of the purple B horizontal cursor and the CH1 decay trace.

Watch this value to make sure your B horizontal cursor is set to 1.47 V, which is approximately 36.8% of 4 V.

Decay time is approximately 1 ms, which is the calculated RC time constant (τ).

Cursor

A 404µs 3.94V

B 1.49ms 1.46V

Δ 1.09ms 2.48V

ƒ 918Hz CH1

Generator: Square; Frequency 50Hz; Amplitude 4; Offset 2

Channels: 1:CH1 in V, 2:DAC in V

Rate: 200µs

Trigger: DAC<2.01V 400µs

ℹ️

Predicted vs. Measured Differences and Component Tolerances

In Figure 8-11, the measured RC Time constant is 1.09 ms, which is within 9% of the calculated τ value of 1 ms. Since the electrolytic capacitor's tolerance is ± 20%, a difference of up to 20% between the measured and predicted values would still be reasonable.

Your Turn: Repeat for RC Decay from 0 to 4 V

For RC decay from 0 to 4 V, all you have to do is toggle the Trigger Edge from Fall to Rise, and then use cursors to measure the time difference between when the capacitor starts charging and when it passes 63.2% of the way to its final voltage level.

- ✓ Try it.

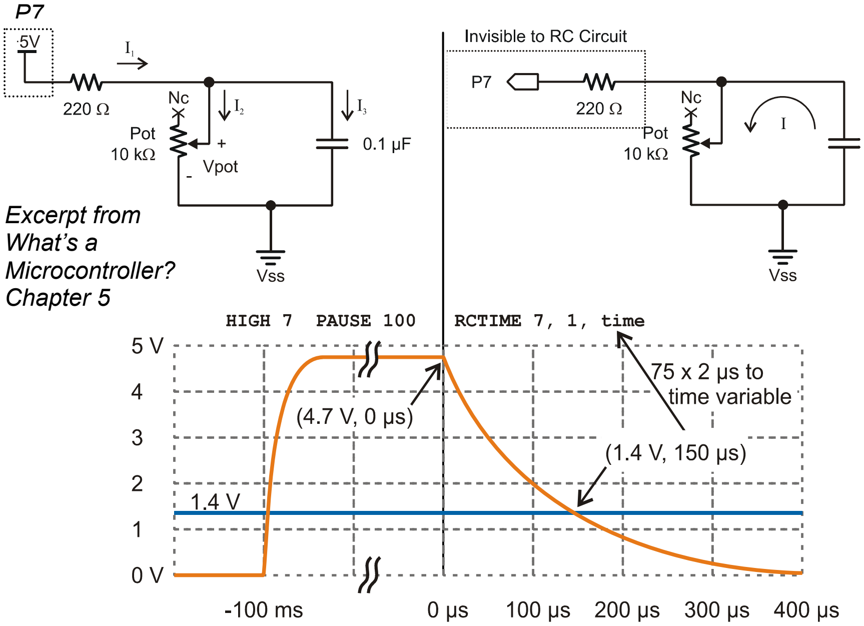
Your value should be almost identical to your decay time since it's a measurement of the RC time constant for the same circuit.

ACTIVITY #3: RC SENSOR MEASUREMENTS WITH A POTENTIOMETER

The BASIC Stamp can measure RC decay with a command called **`RCTIME`**. Since a wide variety of sensors either vary in resistance or capacitance, it's an extremely useful feature. It can also be used with sensors that vary in conductance, allowing more or less current to flow with changes in the physical property the sensor measures. Examples include: visible light, infrared light, humidity, potentiometer knob position, temperature, presence or absence of a flame, and the presence of certain gasses. These are just a few entries in a larger list; if the sensor varies in resistance or capacitance, it's a candidate for the techniques you will examine in this activity.

Figure 8-12 shows how the PBASIC **`RCTIME`** command works with a potentiometer. With its B and W terminals connected as shown, the pot functions as a variable resistor. The adjusting knob can be turned to vary the resistance from 0 Ω to 10 kΩ. On the left side of Figure 8-12, the microcontroller sets I/O pin P7 high. The resulting 5 V signal charges the capacitor through the 220 Ω resistor. After a brief pause (at least 5×τ to charge the capacitor), the **`RCTIME`** command sets P7 to input, and the circuit and voltage behave as shown on the right side of the figure. As an input, the I/O pin P7 is invisible to the circuit, and it's like the voltage source just disappeared. So, the capacitor starts losing its charge though the potentiometer. As it loses its charge, the capacitor's voltage decays. So, the PBASIC **`RCTIME`** command measures the time between when it changed its I/O pin P7 to input and when the capacitor's voltage decays to P7's 1.4 V logic threshold.

Figure 8-12: Voltage at P7 through HIGH, PAUSE, and RCTIME Commands



Different RC Time Constants for Charging and Discharging

Unlike the function generator, which applies a high signal to charge and a low signal to discharge, the BASIC Stamp applies a high signal to charge the capacitor, but then, it makes itself invisible as it lets the capacitor discharge through the potentiometer. Because of this, the capacitor charges more quickly than it discharges.

The RC time constant for the circuit on the left side of Figure 8-12 is 220 Ω × 0.1 µF = 22 µs. So only a tiny delay is needed while the capacitor charges. The RC time constant for the decay on the right is the potentiometer's resistance Rpot × 0.1 µF. Rpot could be as large as 10 kΩ. Assuming the 1.4 V threshold is less than 2 time constants into the decay, an estimate for the longest possible time measurement (with generous padding) would be:

$2 \times \tau = 2 \times 10{,}000 \times 0.1 \times 10^{-6} = 2$ ms

Decay time vs. charge time: With **`RCTIME 7, 1, time`**, the command's ***State*** argument is set to 1, which configures the command to measure voltage as it decays down to 1.4 V. Another way to think about it is that the ***State*** argument tells the **`RCTIME`** command that **`IN7`** will store a 1 when the measurement starts because the voltage applied will be above 1.4 V. Then, **`RCTIME`** measures the time it takes the voltage to decay to 1.4 V, at which point the value **`IN7`** stores transitions from 1 to 0.

Keep in mind that different potentiometer knob positions result in different resistances. In an RC circuit, different resistances result in different decay times that the BASIC Stamp can measure and use to infer the potentiometer knob's position. Since your microcontroller can use this to sense the knob's position, you can use this technique to incorporate an adjusting knob into your microcontroller designs.

Figure 8-13 shows examples of how two different potentiometer resistances result in two different decay times, which the PBASIC **`RCTIME`** command would record as two different time measurements. If the pot's knob has been adjusted to make the resistance larger, the RC time constant will also be larger, which means the capacitor will take longer to decay from its starting voltage to 1.4 V. If the pot's knob has been adjusted to make the resistance smaller, the RC time constant is smaller, and the voltage will take less time to decay.

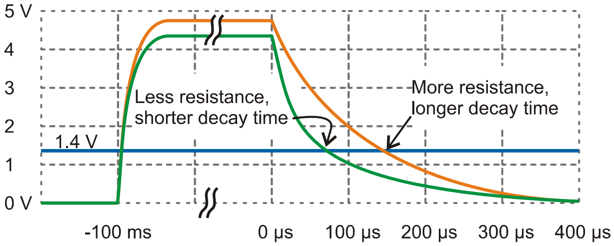
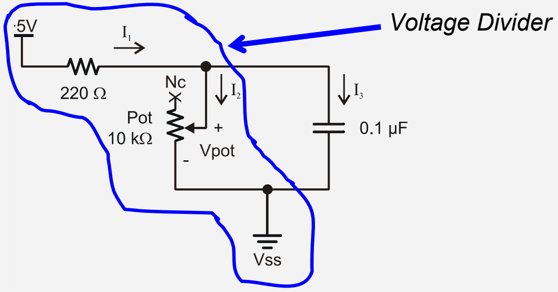


Figure 8-13
How Potentiometer Resistance Affects Decay Time

(Excerpt from What's a Microcontroller? Chapter 5)

The voltage the capacitor charges to also changes slightly in Figure 8-13. The 220 Ω resistor in Figure 8-12 protects the I/O pin from current surges as the capacitor starts charging, and it also protects the I/O pin if the potentiometer's knob is turned to the limit of its range of motion that gives it 0 Ω of resistance. Since the BASIC Stamp sends a high signal to the circuit to charge the capacitor, that high signal would be shorted to ground if the potentiometer is set to 0 Ω. So, the 220 Ω resistor protects the I/O pin from this condition. However, there is a price for the resistor protection. In the case of the RC decay measurement, the resistor forms a voltage divider with the potentiometer's resistance. That voltage divider limits the final voltage the potentiometer can charge up to. This shows in the graph as the capacitor charging up to a lower value when the potentiometer's resistance is less, which in turn reduces the decay time slightly more than the smaller RC time constant would on its own.

Voltage dividers were introduced in Chapter 2, Activity #5. The voltage divider shown below is the 220 Ω resistor in series with the potentiometer with 5 V applied.



Potentiometer Sensor Test Parts

(1) Potentiometer – 10 kΩ (103)
(1) Capacitor – 0.1 µF (104)
(1) Resistor – 220 Ω (red-red-brown)
If you have a BASIC Stamp HomeWork board, use a wire instead of the 220 Ω resistor.
(misc.) Jumper wires

Potentiometer Sensor Test Circuit

Figure 8-14 shows the potentiometer decay test circuit, and Figure 8-15 shows an example wiring diagram.

✓ Build the circuit shown in Figure 8-14 and Figure 8-15.

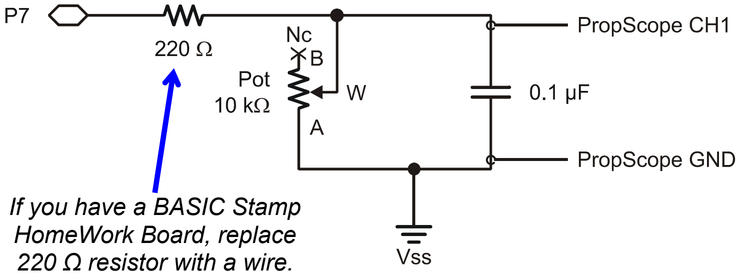

P7
220 Ω
Nc
B
Pot
10 kΩ
W
A
0.1 µF
PropScope CH1
PropScope GND
Vss
If you have a BASIC Stamp HomeWork Board, replace 220 Ω resistor with a wire.

Figure 8-14
Potentiometer Test Circuit Schematic

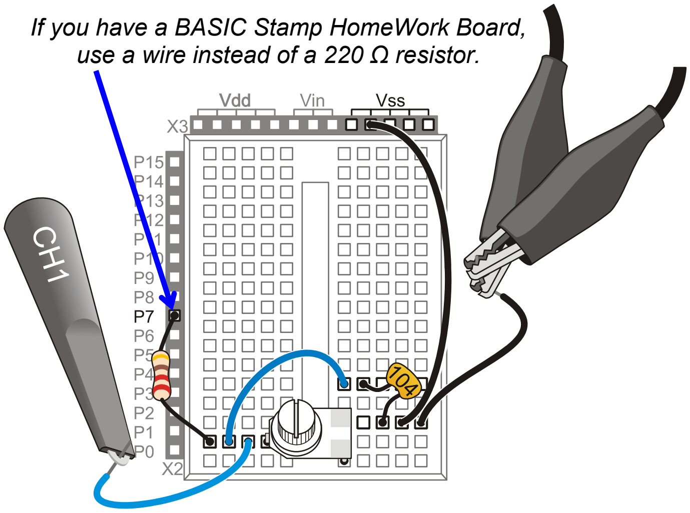

If you have a BASIC Stamp HomeWork Board, use a wire instead of a 220 Ω resistor.
Vdd
Vin
Vss
X3
P15
P14
P13
P12
P11
P10
P9
P8
P7
P6
P5
P4
P3
P2
P1
P0
X2
CH1
104

Figure 8-15
Example Wiring for Figure 8-14.

Potentiometer Sensor Test Code

ReadPotWithRcTime.bs2 is an example program from *What's a Microcontroller?* Its **DO…LOOP** has code that sets P7 high and then waits for 100 ms, which is way more time than necessary for charging capacitor. The main reason for the long **PAUSE** command is to slow down the massages the BASIC Stamp sends to the Debug Terminal to a rate of 10 Hz. After the **PAUSE**, the **RCTIME** command sets the I/O pin to input and then waits for the voltage at I/O pin P7 to decay to 1.4 V. Then, it stores the time measurement in terms of 2 µs units in a variable named **time**. Before it repeats the **DO…LOOP**, the **DEBUG** command displays the **time** variable's measurement result in the Debug Terminal.

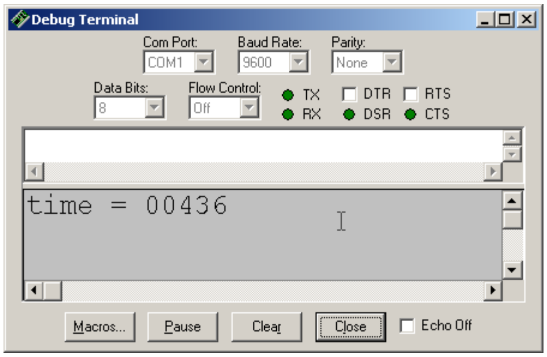
✓ Enter ReadPotWithRcTime.bs2 into the BASIC Stamp Editor and run it.

```
' What's a Microcontroller - ReadPotWithRcTime.bs2
' Read potentiometer in RC-time circuit using RCTIME command.

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

time           VAR     Word                 ' For storing decay times

PAUSE 1000                                  ' 1 s before sending messages
```

```
DO                                             ' Main Loop

  HIGH 7                                       ' Set P7 high
  PAUSE 100                                    ' Wait 0.1 seconds
  RCTIME 7, 1, time                            ' RC Decay time measurement
  DEBUG HOME, "time = ", DEC5 time             ' Display time in 2 us increments

LOOP                                           ' Repeat main loop
```

Figure 8-16 shows the decay measurement, as the number of 2 µs units it took the capacitor voltage to decay from its starting voltage down to 1.4 V. Since the time displayed in Figure 8-16 is 436, it means that the decay took 436 × 2 µs = 872 µs.

- ✓ Maintain downward pressure on the potentiometer knob as you adjust it to keep it in contact with the breadboard connections.
- ✓ Turn the potentiometer knob to a point about ¾ of the way counterclockwise in its range of motion, then make fine adjustments for a Debug Terminal value of 436.



```
time = 00436
```

Figure 8-16
Debug Terminal Pot Measurement

…in 2 µs units.

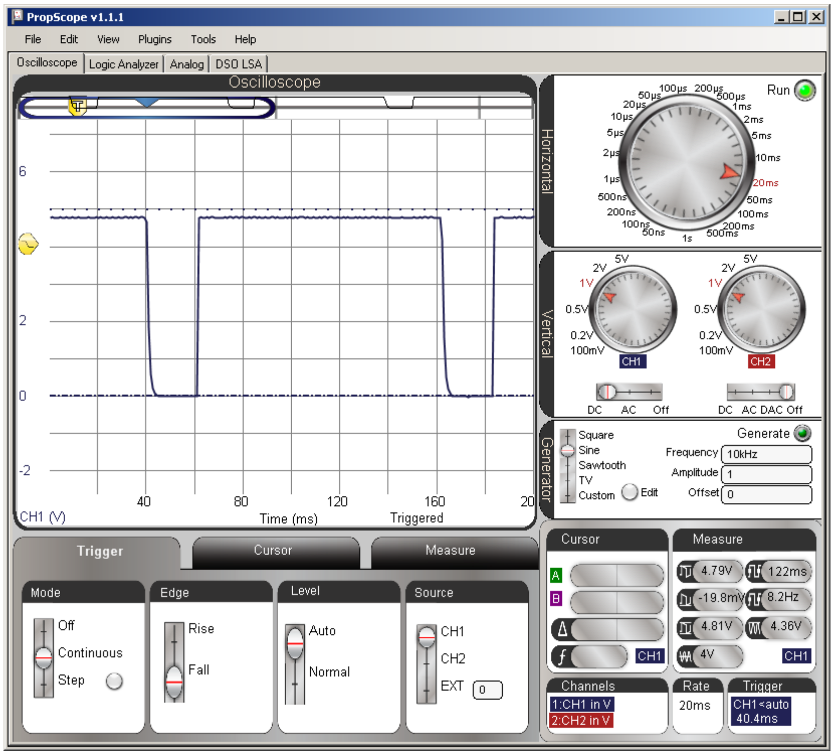
Potentiometer Sensor Test Measurements

Each repetition of ReadPotWithRcTime.bs2 takes at least 100 ms because of the **PAUSE 100** command. In addition, there's the time it takes for the RC decay measurement, the **DEBUG** command, and some time to process all the commands in the loop. All that said, 200 ms is a reasonable estimate for a first look at two cycles of the measurement, so the Horizontal dial is set to 20 ms/div in Figure 8-17.

- ✓ Dials: Horizontal = 20 ms/div, Vertical CH1 = 1 V/div.

- ✓ Vertical coupling switches: CH1 = DC, CH2 = Off.
- ✓ Trigger tab switches; Mode = Continuous, Edge = Fall, Level = Auto, Source = CH1.
- ✓ Trigger Time control: Adjust vertical crosshair to 2nd Time division line.

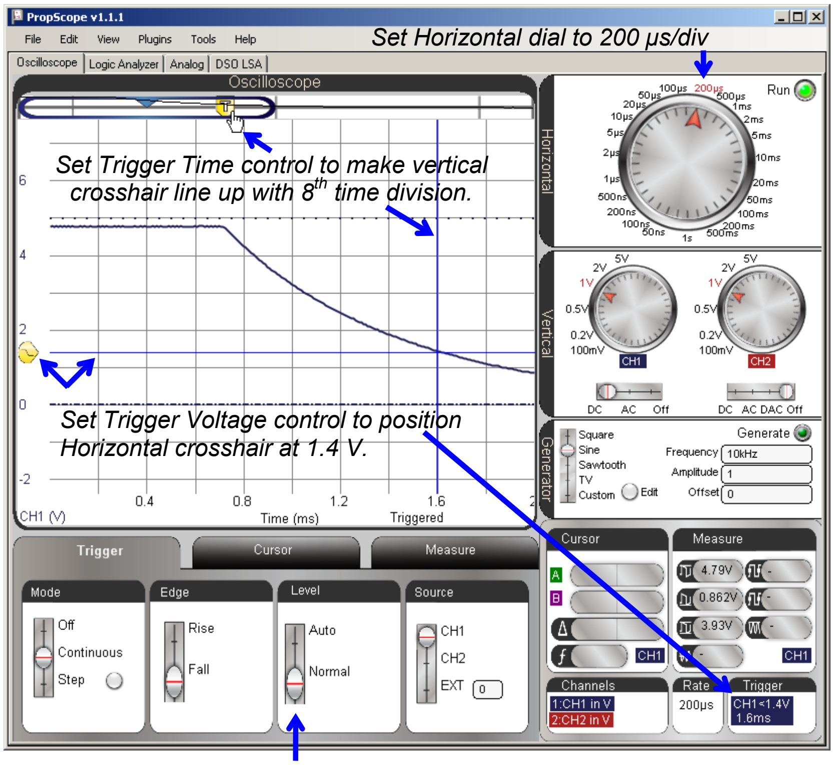
Figure 8-17: Two Decay Measurements Separated by a 100 ms Pause



In Figure 8-17, it's difficult to see any of the decay measurement because the **PAUSE** command is 100 ms, but the decay is only 872 µs. That's less than 1/100th of the **PAUSE** time! Figure 8-19 shows an example of a closer look at the decay portion of the signal. The fact that the Trigger Edge was set to Fall made it so that reducing the Horizontal dial to 500 µs/div zoomed right in on the decay measurement. With this view, the cursors can be used to measure the decay time and verify the BASIC Stamp module's measurement.

The PropScope uses a full cycle to calculate average voltage. When the Trigger tab's Level switch is set to Auto, it uses this level for positioning the Trigger Voltage control. When you zoom in on decay times to take cursor measurements, there's no full cycle, so it's hard for the PropScope to know what the automatic trigger level should be. Also, since we are measuring the time it takes the voltage to decay to the BASIC Stamp module's 1.4 V logic threshold, it's a good idea to set the set the Trigger tab's Level setting to Normal, and then manually adjust the Trigger Voltage control to 1.4 V. The trace will cross this at the end of the measurement, so it's also a good idea to adjust the Trigger Time control to somewhere near the right of the oscilloscope display, like maybe the 8th time division line.

Figure 8-18: Adjustments for Verifying BASIC Stamp RC Decay Measurements



Set Horizontal dial to 200 µs/div

Set Trigger Time control to make vertical crosshair line up with 8th time division.

Set Trigger Voltage control to position Horizontal crosshair at 1.4 V.

Change Trigger tab's Level switch to Normal.

Frequency 10kHz
Amplitude 1
Offset 0

Measure: 4.79V, 0.862V, 3.93V

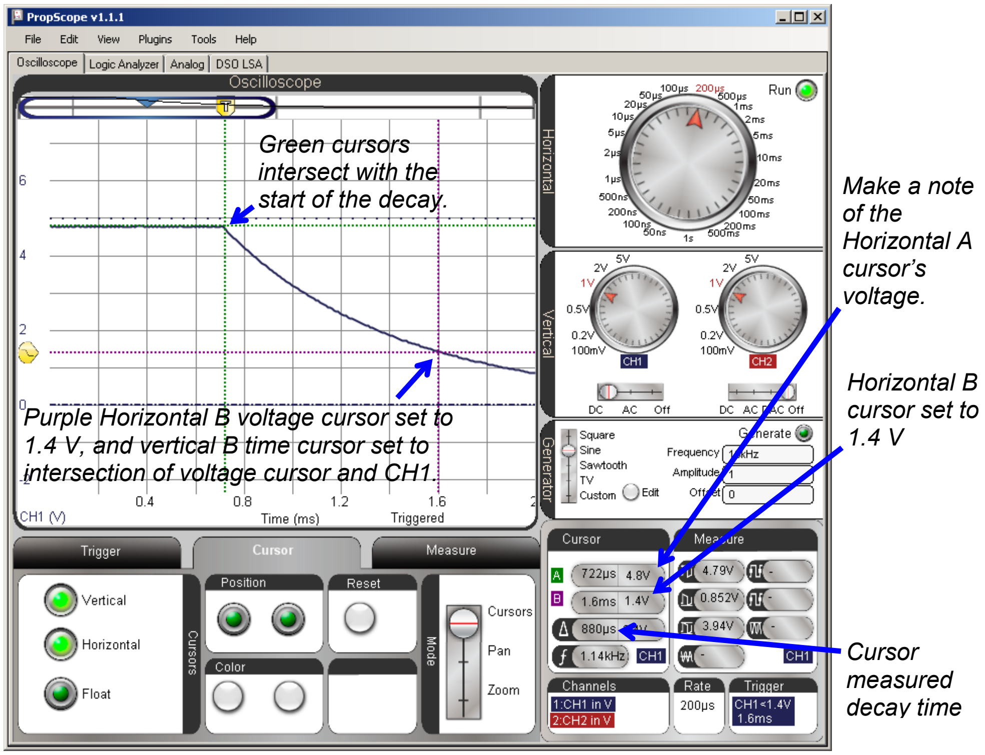
Channels: 1:CH1 in V, 2:CH2 in V
Rate: 200µs
Trigger: CH1<1.4V 1.6ms

- ✓ Change the Trigger tab's Level switch from Auto to Normal.
- ✓ Slide the Trigger Voltage control to 1.4 V. The Trigger display in the bottom-right corner of the window shows your manually adjusted trigger voltage value. Watch it as you adjust the Trigger Voltage control's horizontal trigger crosshair to get as close as possible to 1.4 V.
- ✓ Slide the Trigger Time control to align the vertical trigger crosshair with the 8th time division line.
- ✓ Adjust the Horizontal dial to 500 µs/div.
- ✓ Check your display against Figure 8-18.

The display is now ready for verifying the BASIC Stamp module's 872 µs decay time measurement with a cursor measurement. Figure 8-19 shows how the purple B cursors were adjusted to intersect with the CH1 decay trace at 1.4 V. (Keep in mind that 1.4 V is not 36.8%, it's the BASIC Stamp module's threshold voltage.) Then, the Green A cursors were adjusted to intersect at the start of the decay. The time measurement in the Cursor display's Δ field turns out to be 880 µs, which is close enough to the BASIC Stamp module's 436 × 2 µs = 872 µs time measurement for verification purposes.

- ✓ Set the purple horizontal B voltage cursor to 1.4 V using the Cursor display's B voltage measurement as a guide.
- ✓ Line the vertical B time cursor so that it crosses the intersection of the horizontal B voltage cursor and the CH1 trace.
- ✓ Position the green A cursors up to intersect with the start of the decay.
- ✓ The Cursor display's Δ field shows the difference between the two time cursors, which should be fairly close to twice the value displayed in by the BASIC Stamp in the Debug Terminal.
- ✓ Make a note of the Horizontal A cursor's voltage level. We'll compare the voltage the capacitor charged to in Figure 8-19 against that voltage with a lower potentiometer resistance.

Figure 8-19: Verify the BASIC Stamp module's Decay to 1.4 V Measurement



Green cursors intersect with the start of the decay.

Purple Horizontal B voltage cursor set to 1.4 V, and vertical B time cursor set to intersection of voltage cursor and CH1.

Make a note of the Horizontal A cursor's voltage.

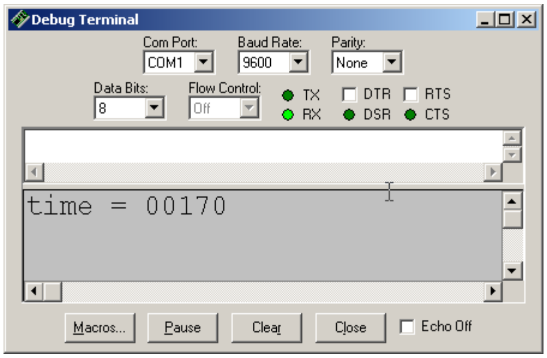
Horizontal B cursor set to 1.4 V

Cursor measured decay time

In Figure 8-19, the cursor A voltage measurement is close to 5 V (4.8 V), which indicates that the potentiometer's resistance is large compared to the 220 Ω resistor. For example, if you substitute 7000 Ω into the voltage divider equation from Chapter 2, the result would be: Vo = 5 V × 7000 Ω ÷ (7000 Ω + 220 Ω) ≈ 4.84 V

Figure 8-20 shows an example the BASIC Stamp module's measurement in the Debug Terminal when the pot is turned to about ¾ of the way clockwise in its range of motion. The decay time is 170 × 2 µs = 340 µs.

- ✓ Turn the potentiometer knob clockwise until the Debug Terminal reports a value in the neighborhood of 170.



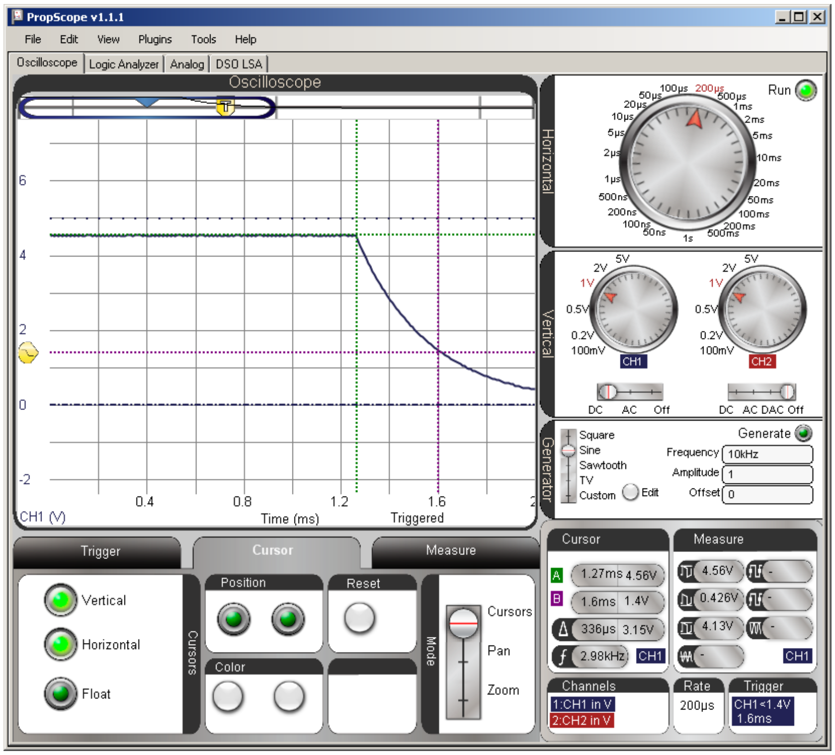
```
time = 00170
```

Figure 8-20
Shorter Decay Measurement

Figure 8-21 shows the corresponding Oscilloscope decay time measurement. The main change is that the potentiometer's resistance is smaller, so the decay time is shorter for a shorter **`RCTIME`** measurement. A secondary change is that the capacitor charges to a slightly lower voltage. That's because the 220 Ω protection resistor value didn't change, but the potentiometer is now a smaller value. As the BASIC Stamp charges the circuit, the slightly lower voltage divider level between the pot and 220 Ω resistor limits the capacitor to charging to a slightly lower level (4.56 V instead of 4.8 V).

- ✓ Repeat the decay cursor time measurement with the vertical cursors. It should be in the neighborhood of 340 µs.
- ✓ Also, repeat the voltage measurement before the decay starts with the green horizontal A cursor, and compare it to the level you measured with the other potentiometer setting.

Figure 8-21: Shorter Decay, Lower Starting Voltage



Frequency 10kHz
Amplitude 1
Offset 0

Cursor
A 1.27ms 4.56V
B 1.6ms 1.4V
Δ 336µs 3.15V
ƒ 2.98kHz CH1

Measure
4.56V -
0.426V -
4.13V -
- CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 200µs
Trigger: CH1<1.4V 1.6ms

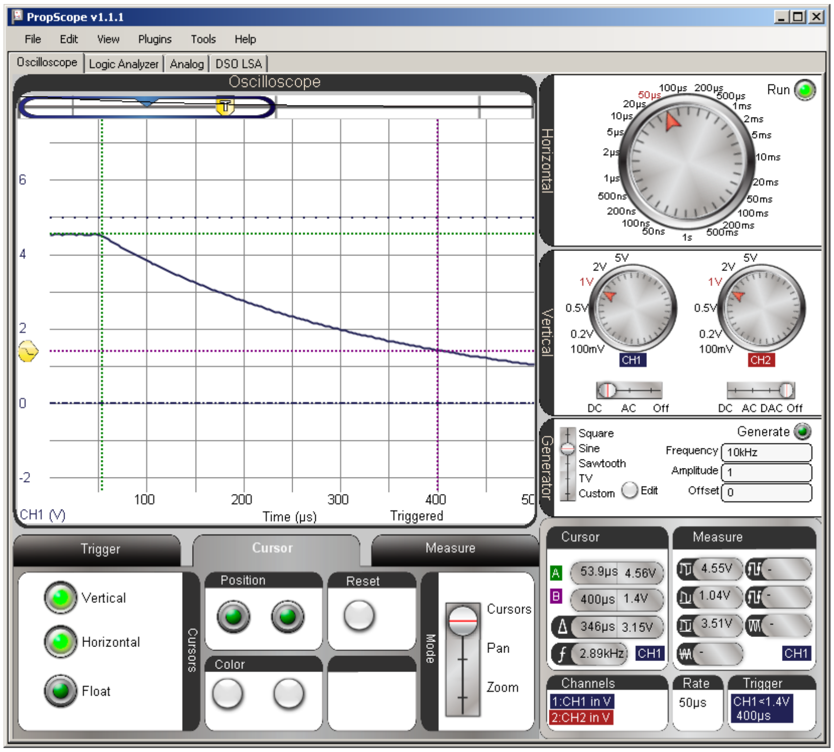
Zoom In on the Shorter Time Value

Figure 8-22 shows how reducing the Horizontal dial to a smaller time/division makes it possible to zoom in and get a more precise measurement. Take a look at the time Δ value in the Cursor panel. Ironically, it's 346 µs, is a little further off than the 336 µs from Figure 8-21. Still, the PropScope is now reporting a more precise measurement with the higher Horizontal time/division setting. Furthermore, a 6 µs difference is not cause for concern. It's only 1.76% off the BASIC Stamp measurement, which can be attributed to a combination of possible differences in the two devices' clock frequencies, slight alignment errors with the cursors, and slight deviations of the actual threshold voltage from the 1.4 V level.

- ✓ Carefully adjust the pot's knob to make the Debug Terminal report 170.

- ✓ Reduce your time division to 50 µs and repeat the cursor time and voltage measurements.

Figure 8-22: Adjust the Horizontal Dial to Zoom in on the Decay Time



Frequency 10kHz
Amplitude 1
Offset 0

Cursor
A 53.9µs 4.56V
B 400µs 1.4V
Δ 346µs 3.15V
ƒ 2.89kHz CH1

Measure
4.55V -
1.04V -
3.51V -
- CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 50µs
Trigger: CH1<1.4V 400µs

Your Turn: Pick a Decay Time and Test

- ✓ Pick a different potentiometer knob adjustment for a different Debug Terminal decay time and use the PropScope to verify the BASIC Stamp measurements.

ACTIVITY #4: RC CIRCUIT'S ROLE IN D/A CONVERSION

In Chapter 4, Activity #2, the relationship between duty cycle and DAC voltage was introduced. In this activity, you will take a closer look at this relationship. After repeating some of the duty cycle and average voltage measurements, you will compare the average voltage the PropScope reports against the average voltage at the RC DAC circuit's output. You will also calculate the average voltage of a signal.

DAC Test Parts

(1) Resistor – 220 Ω (red-red-brown)
(1) Resistor – 1 kΩ (brown-black-red)
(1) Capacitor – 1 µF
(misc.) Jumper wires

DAC Test Circuit

The circuit in Figure 8-23 and Figure 8-24 should be familiar from the duty cycle measurements in Chapter 4, Activity #2.

- ✓ **STOP:** If your CH2 probe's BNC end is still connected to the DAC Card's function generator output, disconnect it and reconnect it to the PropScope's CH2 BNC input before continuing.
- ✓ Build the circuit shown in Figure 8-23 and Figure 8-24.

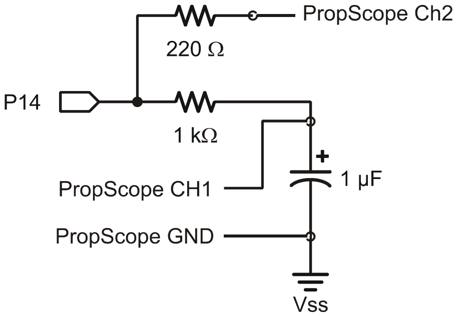


Figure 8-23
DAC Circuit with Probes Attached for P14 PWM Signal and DAC Output Monitoring

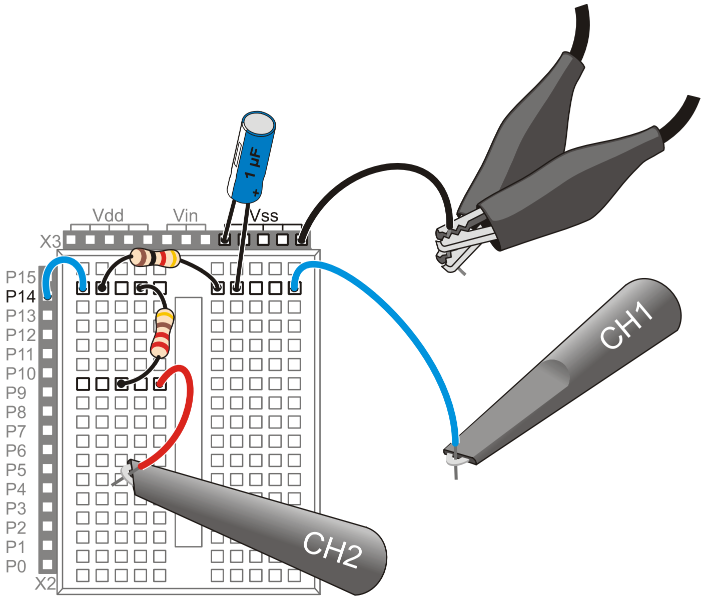


Figure 8-24
Example Wiring
Diagram for Figure 8-23

DAC Test Program

This program was introduced in Chapter 2, Activity #4. It repeatedly sets the voltage across the RC circuit's capacitor to 1.25 V.

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

```
' Test 1 Channel Dac.bs2
' Set P14 capacitor to 1.25 V.
' {$STAMP BS2}                               ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                              ' Language = PBASIC 2.5

PAUSE 1000                                   ' 1 ms before first DEBUG command
DEBUG "Program running..."                   ' Debug Terminal message

DO                                           ' Main Loop

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

LOOP                                         ' Repeat main loop
```

DAC Test Measurements

In Figure 8-25, the CH1 trace shows the DC voltage that the PWM signal establishes across the capacitor. The CH2 trace shows signal activity during six repetitions of the **DO…LOOP** in Test 1 Channel Dac.bs2. This Horizontal dial setting gives an overview of the program running with 1 ms periods of PWM activity followed by about ½ ms of delay between **DO…LOOP** repetitions. We would expect 5 V high PWM pulses; the sporadic pulse heights are a clue that there isn't enough room in the display to view all the signal activity. After this quick check, we'll use smaller settings to zoom in on the PWM signal activity.

- ✓ Dial adjustments: Horizontal = 1 ms/div, Vertical CH1 = 1 V/div, CH2 1 V/div.
- ✓ Vertical coupling switches: CH1 = DC, CH2 = DC.
- ✓ Trigger tab: Mode = Continuous, Edge = Rise, Level = Auto, Source = CH2.
- ✓ Trigger Time control: adjust vertical crosshair to second time division.

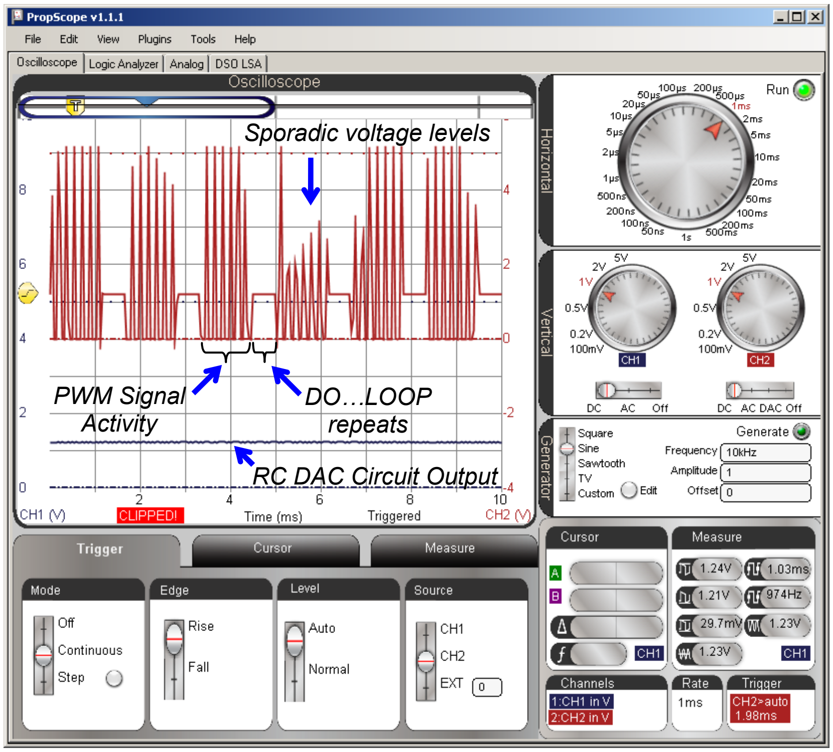


Figure 8-25
Overview of Signal Activity

Since we expect PWM to be a series of 5 V high/low signals, these sporadic voltage levels are a clue that the Horizontal time/division setting can be reduced for more signal detail.

Oversampling vs. Aliasing

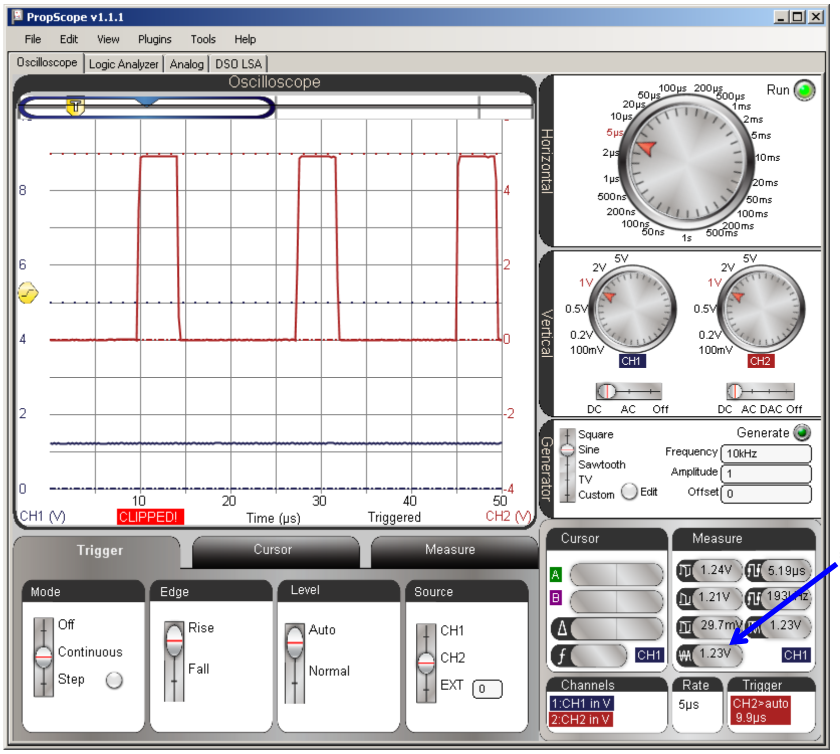
The PropScope takes a series of voltage measurements called *samples*, and sends them to the PropScope software to be displayed. When there are more than enough samples to represent the signal being measured, it is called *oversampling*. If there are not enough samples to properly display the waveform, it is *under-sampled*. If there are so few samples compared to the frequency that one voltage measurement is from one cycle of the signal and the next is from a later cycle, the result is often a signal that looks very different from the actual signal being measured. This is called *aliasing*, and the PropScope displays a signal that's standing in as an alias for the actual signal.

The theoretical minimum sampling rate needed to reconstruct a signal is two samples per cycle, and it's called the *Nyquist rate*. In practice, systems are usually designed to oversample, meaning they take many times the Nyquist rate. For example, it takes 536 voltage samples to construct all the activity in a trace as it crosses the PropScope's Oscilloscope screen.

The Horizontal dial in Figure 8-26 is adjusted to 5 µs/division for a detailed view of the PWM signal's binary pulses. Since the PWM signal is 5 V for ¼ of its cycle, it sets a voltage across the capacitor that's 1.25 V, which is ¼ of the 5 V high pulses. In review of Chapter 4, Activity #2 the duty cycle is the percent ratio of t_{high} to t_{cycle}. The PWM signal in Figure 8-26 is high for ¼ of its cycle, so the duty cycle is 25%. Applied to the RC circuit, it establishes a voltage that's 25% of the 5 V high signal (assuming the low signal is at ground = 0 V the other 75% of the time.)

- ✓ Adjust the Horizontal dial to 5 µs/div.
- ✓ Check the CH2 signal to verify that it is high for ¼ of the time.
- ✓ Check the CH1 Average voltage in the Measure display and verify that it is about 1.25 V. (You can either click the CH1 trace, or click the CH1/CH2 label in the Measure display's lower-right corner to toggle between displaying CH2 and CH1 values.)

Figure 8-26: RC Circuit Converts Signal High ¼ of the time to 1.25 V (¼ of 5 V)



CH2 DC voltage ≈ 1.25 V

PWM *Pin*, *Duty*, *Duration*

Pin is the I/O pin the BASIC Stamp sends the PWM signal to. ***Duty*** is the number of 256ths of a cycle the signal is high, and ***Duration*** is the time the BASIC Stamp transmits the PWM signal.

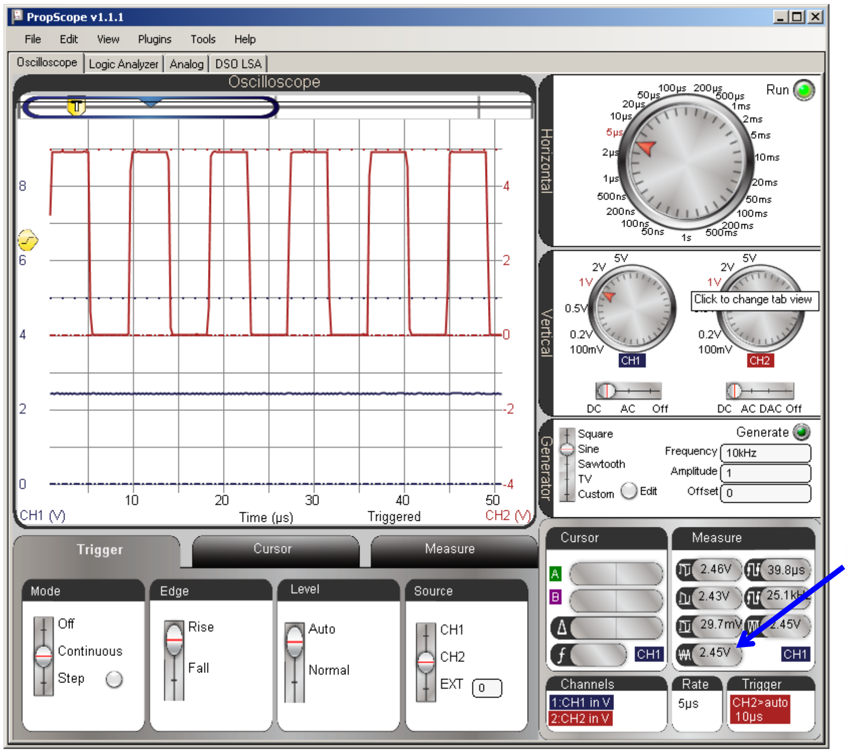
Example: **`PWM 14, 64, 1`** sends a PWM signal that's high 64/256ths of the time for 1 ms. 64/256ths is ¼, so that's why the signal is high ¼ of the time.

For now, all we want to do is be sure that there really is a relationship between the ratio of high time to cycle time (t_{high}/t_{cycle}) that controls the voltage across the capacitor. You can do this by varying the **`PWM`** command's ***Duty*** argument in Test 1 Channel Dac.bs2 and checking the duty cycle and CH1 voltage between each adjustment. For example, Figure

8-27 shows the binary signal for **`PWM 14, 128, 1`**. Now, the signal is high for half the time, and the voltage across the capacitor is 2.5 V, which is about half of 5 V.

- ✓ In Test 1 Channel Dac.bs2, change the **`PWM`** command to **`PWM 14, 128, 1`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Check the CH2 signal to verify that it is high for about half the time.
- ✓ Click the CH1 signal and check its Average voltage in the Measure display to verify that it is about 2.5 V.

Figure 8-27: RC Circuit Converts PWM Signal that's High ½ of the time to 2.5 V (½ of 5 V)

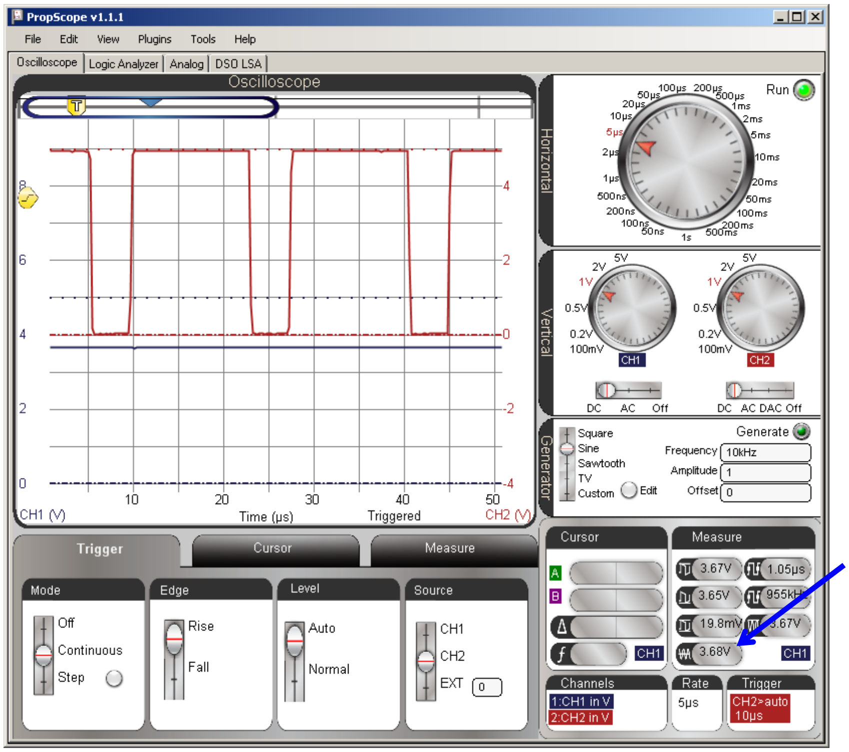


CH1 DC voltage ≈ 2.5 V

Figure 8-28 shows the Oscilloscope display with the command **`PWM 14, 192, 1`**. Now, the CH2 signal is high about ¾ of the time, and CH1 shows that the voltage across the capacitor is now about 3.75 V, which is ¾ of 5 V.

- ✓ In Test 1 Channel Dac.bs2, change the **`PWM`** command to **`PWM 14, 192, 1`**.
- ✓ Load the modified program into the BASIC Stamp.
- ✓ Check the CH2 signal to verify that it is high for about ¾ of the time.
- ✓ Check the Average voltage of the CH1 signal in the Measure display to verify that it is about 3.75 V.

Figure 8-28: RC Circuit Converts PWM Signal that's High ¾ of the time to 3.75 V (¾ of 5 V)



CH1 DC voltage ≈ 3.75 V

Verifying the RC Circuit's "Average" Voltage Output

When a duty cycle signal's period is much smaller than the RC time constant ($T << \tau$) the RC circuit's output transmits the average of the voltage input. With the PBASIC PWM signal, calculating the average is simple because the signal spends its low time at ground. So, the average voltage is 5 V × t_{high} ÷ t_{cycle}. More generally, the average voltage of a signal is the area between the trace and ground during a cycle, divided by the time of a cycle.

$$AverageVoltage = \frac{Area\,between\,signal\,and\,ground\,during\,a\,cycle}{Cycle\,Time}$$

For example, the average voltage in Figure 8-28 is:

$$AverageVoltage = \frac{5V \times 15\mu s}{20\mu s} = 3.75V$$

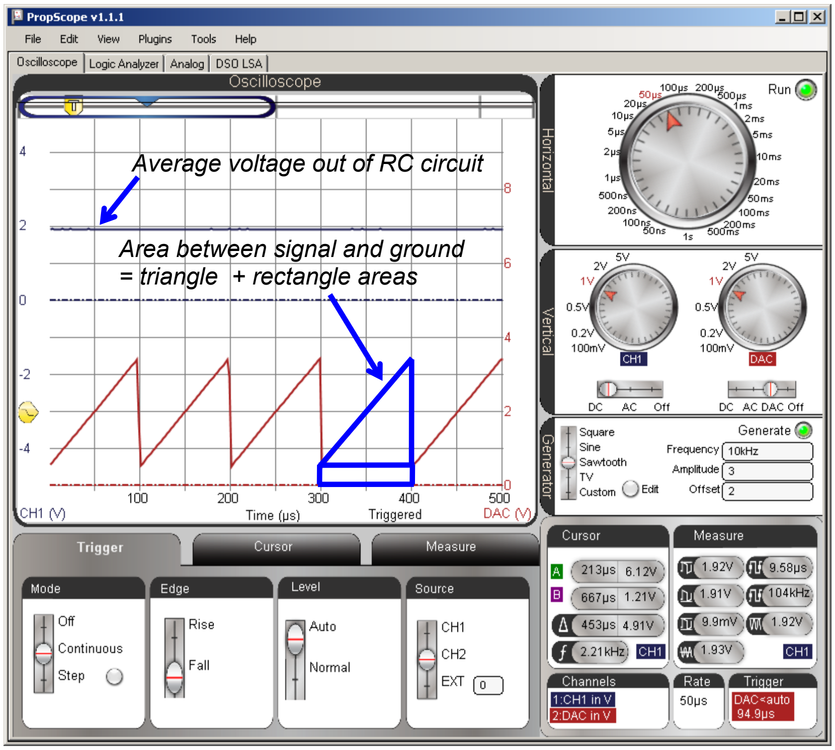
- ✓ Apply the average voltage calculation to Figure 8-27 and Figure 8-26.

Your Turn: Average Voltage for a Different Signal

The average voltage of the lower, red CH2 trace in Figure 8-29 is the area between one of the triangles in the sawtooth wave and the Ground line, divided by the time of one cycle. To calculate the area between the trace and the ground line for a cycle, calculate the area of the triangle (half a rectangle), and add the area of the rectangle below it.

- ✓ Calculate the average voltage by dividing the time of a cycle into the area for Figure 8-29.
- ✓ Set up an experiment that sends the signal in Figure 8-29 to the RC DAC circuit.
  - o Disconnect the I/O pin P14 from the circuit.
  - o Move the CH2 probe's BNC connector to the DAC Card's function generator output. (Repeat the five checklist instructions in the Set DC Voltages with the PropScope's DAC Card section on page 45.)
  - o Configure the PropScope's function generator to transmit a 10 kHz, 3 Vpp, 2 VDC offset Sawtooth wave.
  - o Swap the vertical positions of CH1 and CH2. In other words, position the Blue CH1 trace in upper region of Oscilloscope and ground line for red CH2 trace just above the horizontal time axis values.
  - o Compare your calculated average voltage to the Measured average voltage for both CH1 and CH2.

Figure 8-29: function generator Sawtooth into RC Circuit



Average voltage out of RC circuit

Area between signal and ground = triangle + rectangle areas

Frequency 10kHz
Amplitude 3
Offset 2

Cursor
A 213µs 6.12V
B 667µs 1.21V
Δ 453µs 4.91V
ƒ 2.21kHz CH1

Measure
1.92V 9.58µs
1.91V 104kHz
9.9mV 1.92V
1.93V CH1

Channels
1:CH1 in V
2:DAC in V

Rate
50µs

Trigger
DAC<auto
94.9µs

ACTIVITY #5: RC D/A CONVERTER DECAY FROM LOAD

We can use a PWM signal to generate an analog voltage (as we've been doing) but if we want to use that voltage to do anything, we're going to run into trouble. The load placed onto the PWM output pin by whatever device we attach will drain the capacitor's charge so that we don't actually get the 1.95 volts or whatever voltage we were expecting to get from our DAC circuit.

The DAC circuit needs an op amp buffer before a microcontroller application can use the voltage to drive a load. The op amp buffer circuit, which is also called a voltage follower, will be introduced in Chapter 9, Activity #2. In the meantime, this activity examines what placing a resistor load across the RC circuit does without the op amp buffer. The resistor takes the place of a device we might be trying to power using the output voltage of the DAC circuit.

Extra Parts for Load Test

(1) Resistor – 10 kΩ (brown-black-orange)
(1) Resistor – 2 kΩ (red-black-red)

Load Test Circuit

- ✓ **STOP:** Disconnect the CH2 probe's BNC plug from the DAC Card and reconnect it to the PropScope's CH2 BNC jack before continuing here.
- ✓ Modify the circuit from the previous activity to add a resistor load by placing the 10 kΩ resistor shown in Figure 8-30.

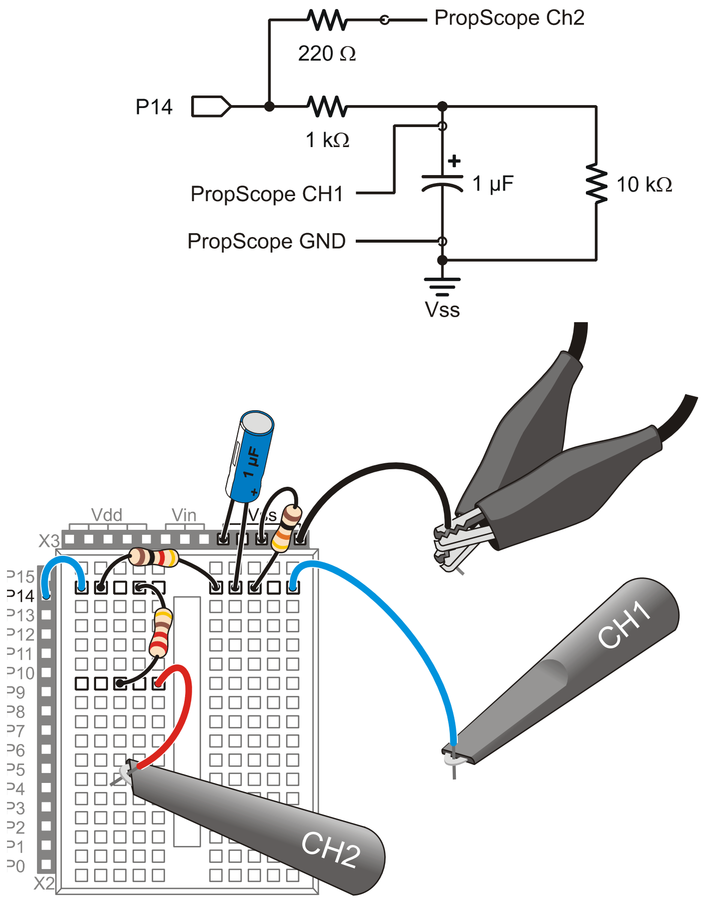


Figure 8-30
PWM DAC Test Circuit with 10 kΩ Resistor Load

Schematic (top)
Wiring diagram (bottom)

DAC Load Test Program

Instead of applying the PWM repeatedly at top speed like the example program from the previous activity, Test 1 Channel Dac with Delay.bs2 applies the PWM signal and then delays for 5 ms before repeating. This makes it convenient to use the PropScope to examine the what happens with the 10 kΩ resistor attached.

- ✓ Enter and run Test 1 Channel Dac with Delay.bs2.

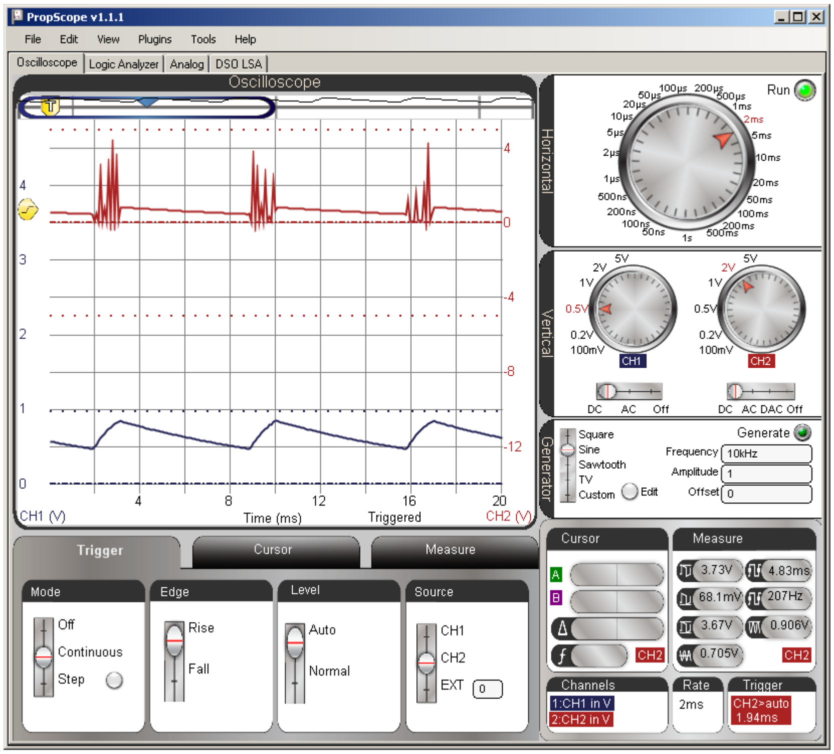
```
' Test 1 Channel Dac with Delay.bs2
' Set P14 capacitor to 1.25 V.
' {$STAMP BS2}                               ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                              ' Language = PBASIC 2.5

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

DO                                           ' Main Loop

  PWM 14, 32, 1                              ' 0.625 V to P14 capacitor
  PAUSE 5                                    ' Delay for 5 ms

LOOP                                         ' Repeat main loop
```

DAC Load Test Measurements

Figure 8-31 shows the decays that result as the capacitor loses its charge through the 10 kΩ resistor. Instead of a steady DC voltage, the capacitor gets charged to an initial voltage, but then it decays through the resistor load. Again, an op amp buffer (aka voltage follower) will solve this problem in Chapter 9, Activity #2.

- ✓ Dial adjustments: Horizontal = 2 ms/div, Vertical CH1 = 0.5 V/div, CH2 = 2 V/div.
- ✓ Vertical coupling switches: CH1 = DC, CH2 = DC.
- ✓ Trigger tab switches: Mode = Continuous, Edge = Rise, Level = Auto, Source = CH2.
- ✓ Trigger Time control: adjust vertical crosshair to first time division line.

Figure 8-31: Voltage Across Capacitor Decays through 10 kΩ Resistor between PWM Signals



Your Turn: Compare to Increased Load and No Load

A 2 kΩ resistor has less resistance, so it conducts more current than a 10 kΩ resistor. As a result, it drains the capacitor more quickly, and can be termed a "heavier" current load on the capacitor.

With the resistor removed, the capacitor should lose its charge very slowly. The rate the capacitor loses its charge depends on the insulator that keeps the two metal plates that hold charge apart. This insulator is called a *dielectric*, and the small current that flows through the dielectric is called *leakage current*.

- ✓ Replace the 10 kΩ resistor with a 2 kΩ resistor.
- ✓ Check the voltage output results with the PropScope.

- ✓ Disconnect the load resistor, the decay should be very small in comparison. With a low leakage current capacitor, it would be even smaller.

ACTIVITY #6: PHOTOTRANSISTOR LIGHT SENSOR EXAMPLE

The phototransistor shown in Figure 8-32 is a light sensor that is also compatible with `RCTIME` measurements. C, B, and E stand for the phototransistor's terminals, which are named collector, base, and emitter. Brighter light shining on the phototransistor's base causes it to conduct more current, dimmer light results in less current. By measuring how quickly the phototransistor allows the capacitor to discharge in an RC circuit, a microcontroller can detect light levels. In this activity, you will test BASIC Stamp light measurements with the BASIC Stamp, and verify the time measurements with the Oscilloscope.

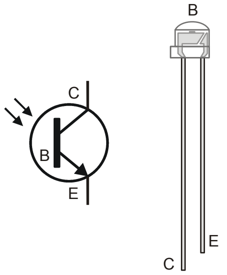
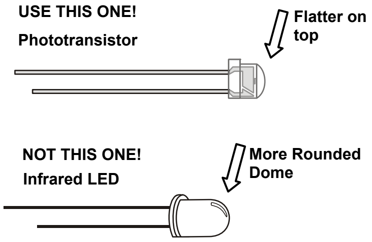


Figure 8-32
Phototransistor Schematic Symbol and Part Drawing

Phototransistor Light Sensor Parts

(1) Phototransistor
(1) Capacitor – 0.1 µF (104)
(1) Resistor 10 kΩ (brown-black-orange)
(misc.) Jumper wires



Phototransistor Light Sensor Circuit

Figure 8-33 shows a schematic and wiring diagram example of the phototransistor in an RC decay circuit. Although this is still an RC decay circuit, there are several differences. First, the pot has been replaced with a phototransistor. Second, the I/O pin assignment is different, P2 instead of P7. Third, the capacitor is ten times the value that was used with the previous circuit. Fourth and finally, this circuit has a 10 kΩ resistor instead of a 220 Ω resistor. This is not for increased I/O pin protection; it's for a light level compensation feature that will be examined after the basic measurements.

✓ Build the circuit shown in Figure 8-33.

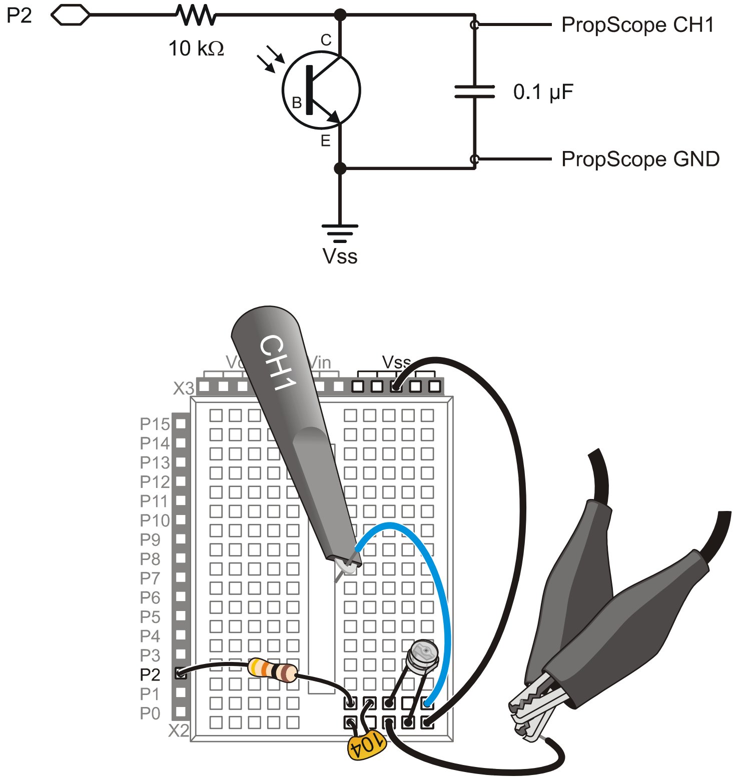


Figure 8-33
Phototransistor Circuit

Schematic (top)
Wiring diagram (bottom)

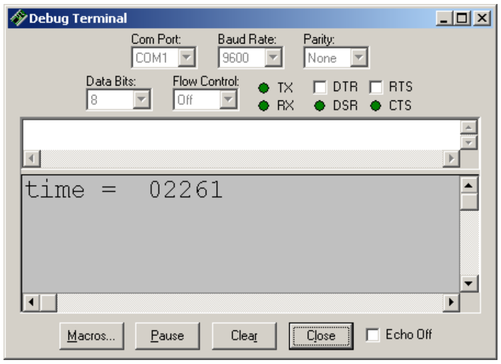
TestPhototransistor.bs2, tests the phototransistor in much the same fashion as Activity #3's example program tested the potentiometer.

- ✓ Enter TestPhototransistor.bs2 into the BASIC Stamp Editor and run it.
- ✓ Make sure the lighting conditions are low—no bright lights or sunlight from a nearby window.
- ✓ If the Debug Terminal measurement is 0, it indicates that he time measurement exceeded the **RCTIME** command's 65535 maximum value, which in turn indicates that it's either too dark, or there is a wiring mistake.

```
' What's a Microcontroller - TestPhototransistor.bs2
' Read phototransistor in RC-time circuit using RCTIME command.
' {$STAMP BS2}                                ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                               ' Language = PBASIC 2.5

time           VAR     Word                   ' For storing decay times
PAUSE 1000                                    ' 1 s before sending messages

DO                                            ' Main Loop
  HIGH 2                                      ' Set P2 high
  PAUSE 100                                   ' Wait 0.1 seconds
  RCTIME 2, 1, time                           ' RC Decay time measurement
  DEBUG HOME, "time = ", DEC5 time            ' Display time in 2 us increments
LOOP                                          ' Repeat main loop
```

Figure 8-34 shows the Debug Terminal display for the phototransistor test. This measurement should get larger with less light and smaller with more light. This circuit was designed for indoor lighting, so make sure to stay out of really bright light conditions, especially sunlight streaming in through a window.

- ✓ Test your phototransistor by casting a shadow over it with your hand; the measurement should increase.
- ✓ Remove the shadow, and the measurement should decrease again.

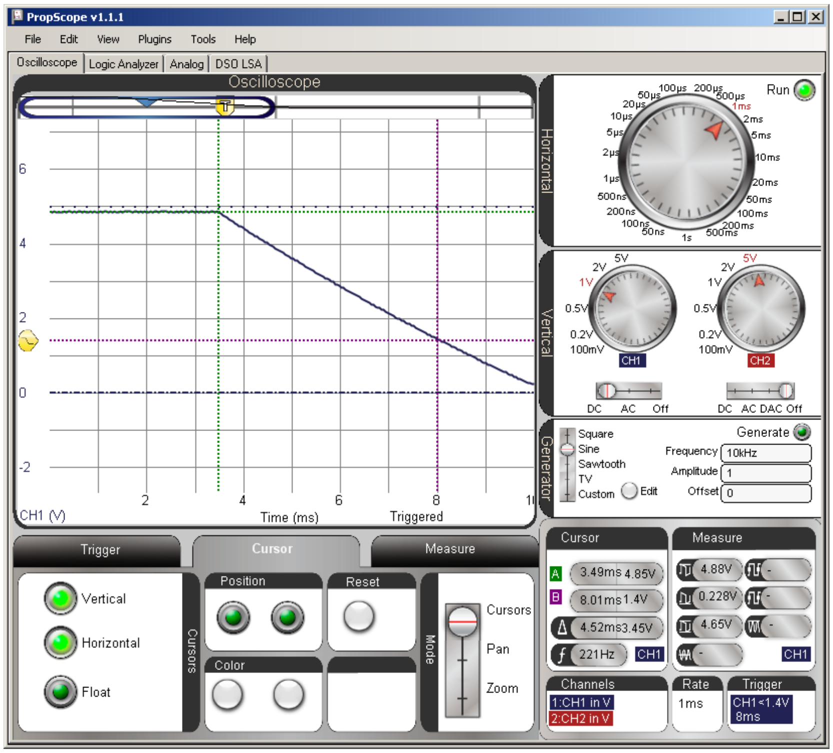


Com Port: COM1
Baud Rate: 9600
Parity: None
Data Bits: 8
Flow Control: Off

```
time =  02261
```

Figure 8-34
BASIC Stamp Phototransistor Decay Measurement

Figure 8-35 shows an example of the decay measurement. It utilizes the same procedure as the potentiometer decay measurement, except that the Horizontal dial is adjusted to a coarser time scale of 500 µs/div to accommodate the decay time. Your lighting conditions may be different, and you may end up having to choose a different Horizontal dial setting. In this example, the BASIC Stamp **`RCTIME`** measurement is 2261 × 2 µs = 4522 µs. The PropScope Δ measurement is 4.52 ms, which is 4520 µs. Especially since the light sensor measurement fluctuates, we are not expecting an exact match, just oscilloscope verification that the BASIC Stamp measurement is in the right neighborhood. So this might be somewhat of a closer match than a typical measurement.



Frequency 10kHz
Amplitude 1
Offset 0

Cursor
A 3.49ms 4.85V
B 8.01ms 1.4V
Δ 4.52ms 3.45V
ƒ 221Hz CH1

Measure
4.88V -
0.228V -
4.65V -
- CH1

Channels: 1:CH1 in V, 2:CH2 in V
Rate: 1ms
Trigger: CH1 <1.4V 8ms

Figure 8-35
BASIC Stamp Phototransistor Decay Measurement

- ✓ Use your Debug Terminal time measurement to figure out approximately what Horizontal dial setting will suit your lighting conditions/decay times. Remember to multiply by 2 to get the number of microseconds.

- ✓ Use the PropScope to verify the BASIC Stamp module's time measurements in your lighting conditions. Depending on your lighting conditions, your time scale may have to be significantly different from what's in the figure.

From RC Decay to Linear Decay

The shape of the trace no longer matches the RC decay curve. In fact, the decay looks almost linear, and that's because it **is** almost linear. The transistor is like a current valve, and the light level controls how much current it allows through, which in turn controls the rate the capacitor discharges. The light level is more or less constant during the measurement, so the capacitor loses its charge at a linear rate, which results in a linear voltage decay.

Automatic Lighting Condition Adjustments

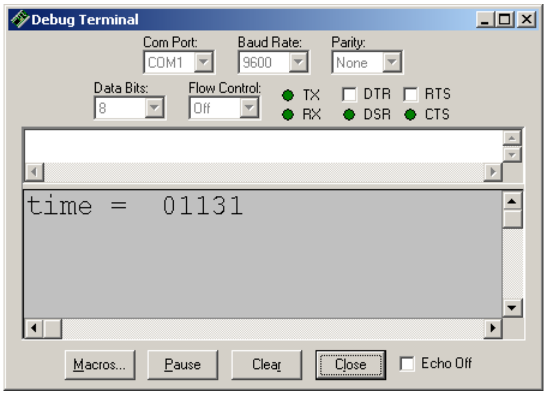
One important difference with the phototransistor is that it's a current conducting device that does not create a voltage divider. So the capacitor will charge up to the about the same value regardless of the light level. If you take the phototransistor out of the circuit, you have an RC DAC circuit for setting voltages. With the phototransistor in the circuit, you can actually use the **`PWM`** command to charge up the capacitor to a certain level, followed by the **`RCTIME`** command to measure the decay time. This makes it possible for the program to adjust to lighting conditions by charging the capacitor to higher or lower voltages before the measuring the decay time. For example, Test Phototransistor Adjustment.bs2 uses the **`PWM`** command to charge the capacitor up to about 3.34 V (instead of 5 V) before starting the **`RCTIME`** measurement. The result will be a lower light measurement value because the voltage before the decay is lower.

- ✓ Enter Test Phototransistor Adjustment.bs2 into the BASIC Stamp Editor and Run it.

```
' What's a Microcontroller – Test Phototransistor Adjustment.bs2
' Read phototransistor in RC-time circuit using RCTIME command.

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

time           VAR     Word                  ' For storing decay times

PAUSE 1000                                   ' 1 s before sending messages
```

```
DO
                                             ' Main Loop
  PAUSE 100                                  ' Wait 0.1 seconds
  PWM 2, 171, 10                             ' Charge to 171/256 of 5 V
  RCTIME 7, 1, time                          ' RC Decay time measurement
  DEBUG HOME, "time = ", DEC5 time           ' Display time in 2 us increments

LOOP                                         ' Repeat main loop
```

Figure 8-36 shows a light measurement in roughly the same lighting conditions as the previous measurement. However, the result is only half the value, with a decay time of 1131 × 2 µs = 2262 µs.



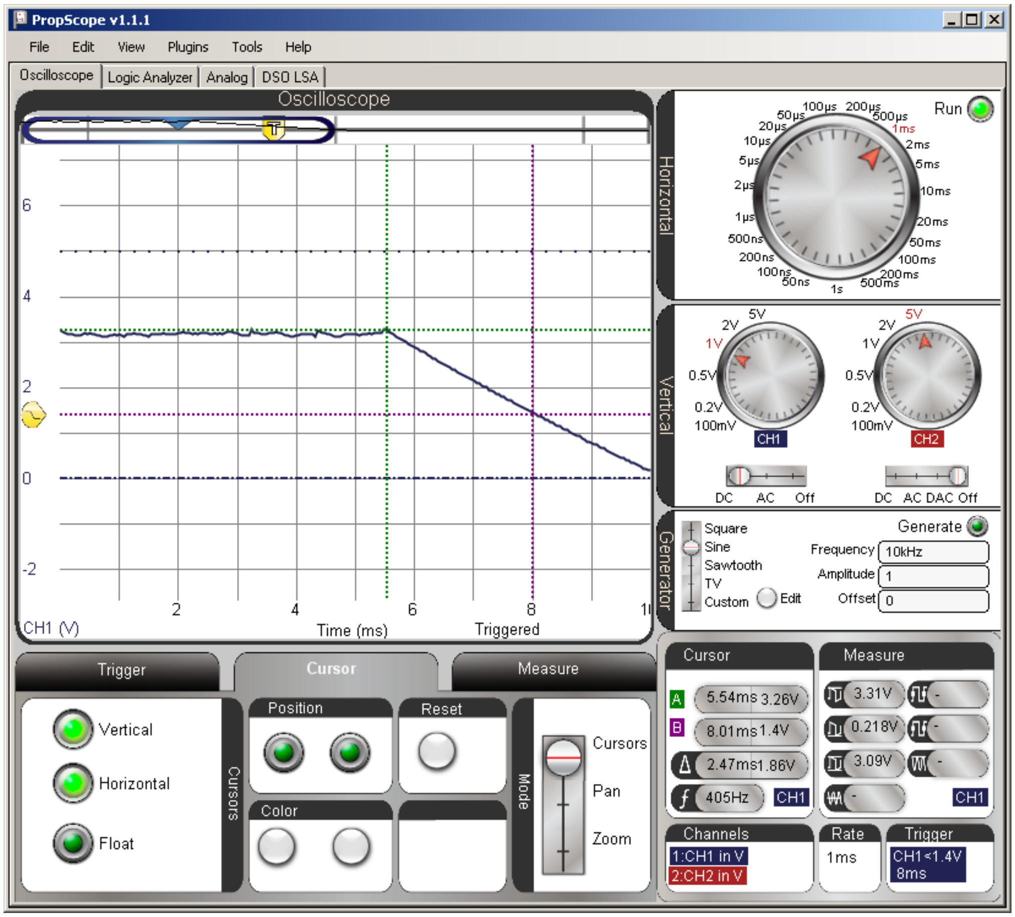
```
time =  01131
```

Figure 8-36
Same Light Level, PWM Reduced Sensitivity

Figure 8-37 shows an example of the lower starting voltage, which contributes to a smaller decay time. However, there appears to be something more than just lower starting voltage that's reducing the BASIC Stamp decay time measurement, which is 2262 µs. In contrast, the oscilloscope measurement is 2.47 ms = 2470 µs. That's more than 200 µs difference between the BASIC Stamp and PropScope measurement. Unlike previous measurement differences, which were small, this is a definite indicator that something is not performing as expected.

- ✓ Try comparing your BASIC Stamp module's measurements using Test Phototransistor Adjustment.bs2 against the PropScope decay measurements. See the difference between BASIC Stamp and PropScope measurement here?

Figure 8-37: How PWM Determines the Starting Voltage



Horizontal: 1ms

Vertical: CH1 1V, DC; CH2 5V, Off

Frequency 10kHz

Amplitude 1

Offset 0

Cursor

A	5.54ms	3.26V
B	8.01ms	1.4V
Δ	2.47ms	1.86V
ƒ	405Hz	CH1

Measure

3.31V	-
0.218V	-
3.09V	-
-	CH1

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 1ms

Trigger: CH1<1.4V 8ms

It turns out that the cause of the measurement difference is the fact that the **PWM** command changes the I/O pin direction to input when it's done. In D/A conversion with no circuit load, that change from output to input allows the capacitor to hold its charge (and voltage). However, our light sensor circuit has the phototransistor that starts draining the capacitor as soon as the **PWM** command is done. So, after the **PWM** command finishes, we can infer from the measurement that there's about 200 µs of capacitor voltage decay time, before the **RCTIME** command starts.

The **PWM**-before-**RCTIME** technique is still useful, because it does make it possible to adjust the system's light sensitivity. With additional testing, a better estimate of time between **PWM** and **RCTIME** can be determined. Then, the program can just add that value to the measurements. In other applications, this time difference doesn't even matter. For example, there's an application for the Boe-Bot robot that makes it compare two of these light sensor measurements to figure out which one detects brighter light. This works well for light source following activities. The Boe-Bot robot's program doesn't care about precise measurements, it just wants to know which measurement is larger. Since the extra time would be identical for both sensors, the comparison of the two measurements still supplies the information the BASIC Stamp needs for determining the direction of the light source.

A current load on a DAC circuit without an op amp has its uses; this is one of them.

The previous activity demonstrated how a resistive load prevented the capacitor in a DAC circuit from holding its charge, which in turn prevented it from maintaining an output voltage. If the goal is to maintain an output voltage under load, the load poses a problem that an op amp will be used to fix in the next chapter.

In this automatic light level adjustment activity, the goal was not to maintain a voltage output; it was to charge a capacitor with a light sensor load to a certain voltage before measuring the decay. With control over the voltage, it is possible to adjust a system to adapt to different light levels. So this is an application where placing a load a cross the DAC circuit without the op amp buffer is useful.

Think about how a property that's considered "bad" for one application can be "good" for another. Try to keep it in mind as you learn more about how circuits interact with each other. Applying an element that's considered "bad" for one circuit or application might just solve a thorny design problem for a different one.

ACTIVITY #7: LOW-PASS FILTER

Chapter 7, Activity #6 introduced a low-pass filter example, and tests demonstrated that higher frequency sine waves applied to the circuit's input resulted in lower amplitude sine waves with more phase shift at the output. In contrast, lower frequency sine waves were allowed to "pass" through the filter with less attenuation and phase shift.

The value of R×C that we've been using to characterize RC decay is also a key value for an RC low-pass filter. You can use it to get the value of the filter's *cutoff frequency*, which is the point at which the output sine wave amplitude is reduced to 70.7% of the input sine wave, and the phase is shifted by 45°. Another way to say this is that the output signal amplitude, divided by the input signal amplitude is 0.707. This value is the reciprocal of the square root of two.

$$\frac{v_{out}}{v_{in}} = 0.707 = \frac{1}{\sqrt{2}}$$

In this activity, you will calculate the cutoff frequency for an RC low-pass filter, and apply that frequency to the filter's input with the PropScope's function generator. You will then test the amplitude and phase of the circuit's output to verify the frequency calculation.

Corner frequency is another common expression for cutoff frequency. When the amplitude response of a filter is plotted over frequency, the amplitude turns a corner and starts declining at the cutoff frequency. These graphs are common in the analysis of filters and amplifiers, and they are called *Bode plots.* (The name Bode is pronounced "bo-dee.")

Low-pass Filter Test Parts List

(1) Resistor – 10 kΩ (brown-black-orange)
(1) Capacitor – 0.01 µF
(misc.) Jumper wires

Low-pass Filter Test Circuit

The circuit schematic in Figure 8-38 and wiring diagram example in Figure 8-39 are repeats of the circuits from Chapter 7, Activity #6.

- ✓ Unclip the CH2 probe tip from whatever it's connected to.
- ✓ Disconnect the CH2 probe from the PropScope at its BNC connector and connect it to the DAC Card's function generator output.
- ✓ Build the RC circuit shown in Figure 8-38 and Figure 8-39.
- ✓ Connect the function generator output to the RC low-pass filter input.
- ✓ Connect the CH1 probe to the filter's output.

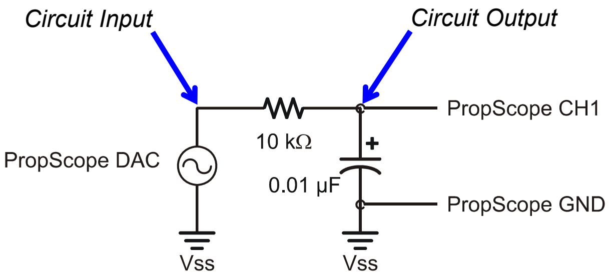
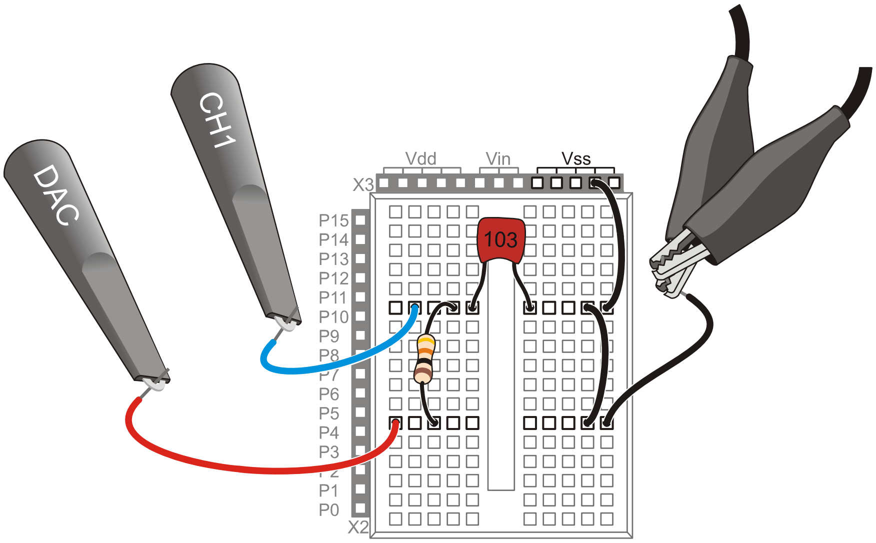


Figure 8-38
Schematic for RC Low-pass Filter Amplitude and Phase Angle Test Measurements

Figure 8-39: Example Wiring Diagram for Figure 8-38



Calculate the Cutoff Frequency and Phase Shift

The three values we need to know for verifying the filter's behavior are output amplitude, frequency, and phase shift in terms of time. This activity started by mentioning that the output sine wave amplitude will be 0.707 × the input sine wave amplitude and that the phase delay will be 45°. The cutoff frequency still needs to be calculated, and then that has to be used to calculate a 45° phase delay time.

The equation for cutoff frequency is:

$$f_C = \frac{1}{2\pi RC}$$

Substituting our R = 10 kΩ and C = 0.01 μF values, the cutoff frequency is:

$$f_C \approx \frac{1}{2 \times 3.1416 \times 10{,}000 \times 0.00000001} \approx 1{,}591.5\,Hz$$

Next, we need to figure out what the phase shift time is for 45° at 1591.5 Hz. To do this, all we need to do is rearrange the terms in one of the two phase angle equations from Chapter 7, Activity #6, and solve for Δt. Since we now know the frequency, we can use the phase angle equation with the frequency term and save the 1/T calculation:

$$\theta = \Delta t \times f \times 360^\circ \;\rightarrow\; \Delta t = \frac{\theta}{f \times 360^\circ}$$

8

Next, solve for Δt when θ = 45°.

$$\Delta t = \frac{45^\circ}{1591.5Hz \times 360^\circ} \approx 78.5\,\mu s$$

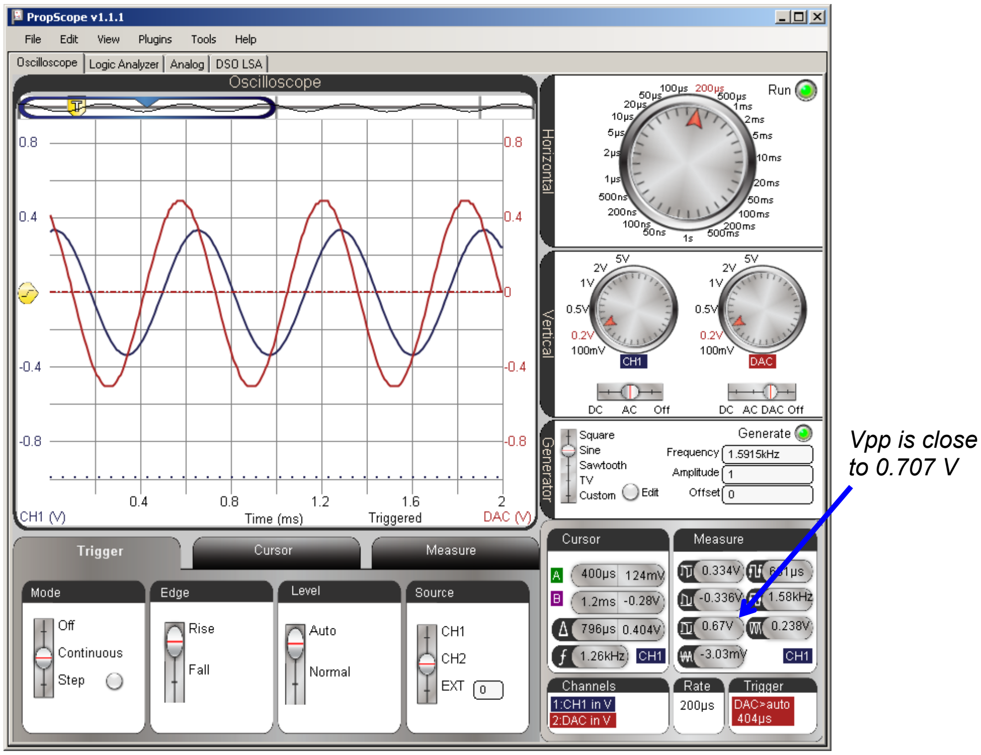
Now we know to that the cutoff frequency for our RC filter is 1591.5 Hz. If we supply the filter with a sine wave at this frequency, we'll expect the output to be 70.7% of the input signal amplitude, with a phase shift if 45°, which corresponds to a time shift of 78.5 μs.

Test the Amplitude and Cutoff Frequency

First let's check to find out if the filter's output signal is about 70.7% of the input signal from the function generator. To make this measurement easy, the PropScope's function generator can supply the RC circuit's input with a 1.5915 kHz sine wave that's 1 Vpp. Then, the output signal's amplitude should be about 0.707 Vpp. Figure 8-40 shows an example.

- ✓ Set the Generate switch to Sine, with a Frequency of 1591.5 Hz, an Amplitude of 1 V, an Offset of 0 V, and remember to click the Generate button.
- ✓ Adjust the dials, coupling switches and Trigger tab settings to match Figure 8-40.
- ✓ Adjust the trace positions as shown in the Oscilloscope display, with both the CH2/DAC trace and CH1 circuit output trace in the middle of the screen. Their ground lines should be right on top of each other.
- ✓ Set the Measure display to CH1. If its channel indicator is set to CH2, click it to toggle back to CH1.
- ✓ Use the Measure display to verify that the CH1 measurement of the output sine wave's amplitude is in the 0.707 V neighborhood.

Figure 8-40: Amplitude Output/Input Comparison



PropScope v1.1.1

Generator: Sine; Frequency 1.5915kHz; Amplitude 1; Offset 0

Cursor: A 400µs 124mV; B 1.2ms -0.28V; Δ 796µs 0.404V; ƒ 1.26kHz; CH1

Measure: 0.334V; 6[illegible]1µs; -0.336V; 1.58kHz; 0.67V; 0.238V; -3.03mV; CH1

Channels: 1:CH1 in V; 2:DAC in V

Rate: 200µs

Trigger: DAC>auto 404µs

Vpp is close to 0.707 V

In Chapter 7, the output sine wave from the circuit was positioned below the input sine wave, and cursors were used to make the comparisons. Since the Measure display's amplitude value always closely matched the horizontal voltage cursor measurements, we'll rely on it instead of the cursors for amplitude measurements. Chapter 7 also mentioned another common technique of displaying the sine waves with both the CH1 and CH2 ground lines set to the same level. It superimposes the sine waves for improved visual comparison. So, in this chapter we'll rely on that approach.

How close should the measurement be?

The tolerances of the resistor and capacitor are 5%, meaning that each one's actual value could be 5% smaller or larger than its nominal (named) value. Other sources of error might include the capacitance inside the breadboard clips and the PropScope's 2% voltage measurement tolerance. All tolled, ±7 % error is reasonable. The percent error equation is:

$$\%error = \frac{measured - predicted}{predicted} \times 100\%$$

With the measurement of 0.682 Vpp from Figure 8-40, the percent error is:

$$\%error = \frac{0.67\ Vpp - 0.707Vpp}{0.707Vpp} \times 100\% \approx -5.23\%$$

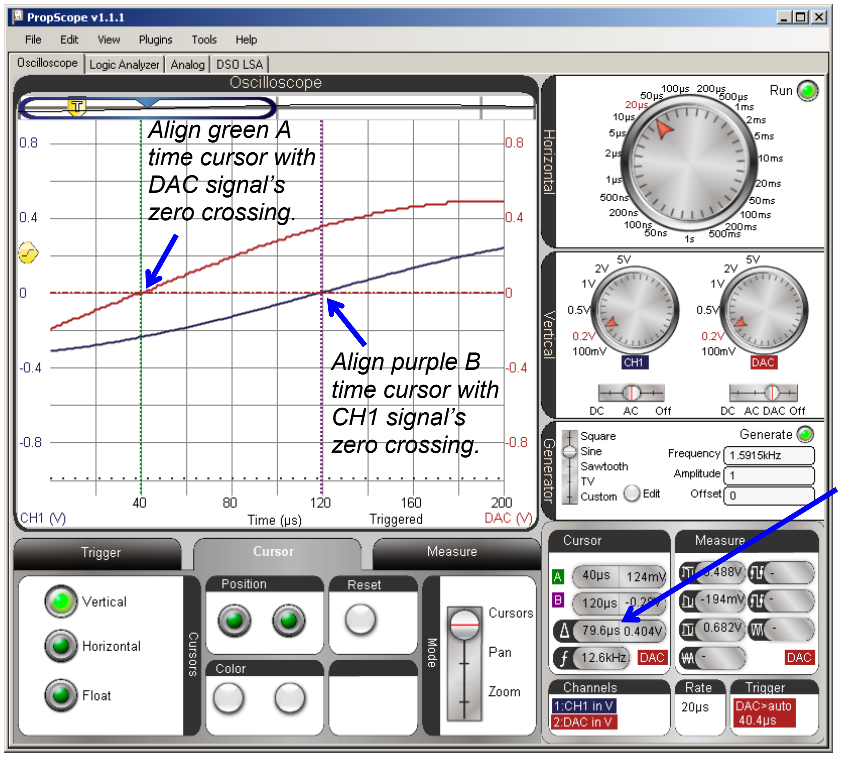
For more information, see the Filter Error Propagation Example in the Understanding Signals Supplement, which is available as a free PDF online from the Downloads section of www.parallax.com/go/PropScope.

Although the horizontal voltage cursors were not necessary for amplitude measurements, the vertical time cursors are still essential for the phase delay measurement. Figure 8-41 shows an example. The Horizontal time/division setting is reduced for a more precise time difference measurement. As in Chapter 7, each signal's 0 V crossing is a reference point for the beginning of each sine wave's cycle. To measure the time difference between the starting points of each sine wave's cycle, each vertical time cursors is positioned to intersect with a channel trace where it crosses the 0 V ground line. This provides a delay measurement that can be compared to our calculated Δt value of 77.8 µs.

✓ Adjust the Horizontal dial to 20 µs/div.
✓ Go to the Cursors tab and turn the Vertical cursors on and the other cursors off.
✓ Position the green vertical A time cursor so that it intersects with the CH2/DAC trace as it crosses the 0 V ground line.

- ✓ Position the purple vertical B time cursor so that it intersects with the CH1 trace as it crosses the CH1 0 V ground line.
- ✓ Check the Δ measurement in the Cursors display, and compare to the calculated 78.5 µs value.

Figure 8-41: Phase Output/Input Comparison



Align green A time cursor with DAC signal's zero crossing.

Align purple B time cursor with CH1 signal's zero crossing.

Get the resulting Δ time measurement from this field in Cursor display.

Generator: Sine; Frequency 1.5915kHz; Amplitude 1; Offset 0

Cursor:
A 40µs 124mV
B 120µs -0.29V
Δ 79.6µs 0.404V
ƒ 12.6kHz DAC

Measure: 0.488V; -194mV; 0.682V; DAC

Channels: 1:CH1 in V; 2:DAC in V

Rate: 20µs

Trigger: DAC>auto 40.4µs

Again, how close should the measurement be?

For the phase delay, ±6% is reasonable. Comparing the 79.6 µs measurement from Figure 8-41 against the predicted value of 78.5 µs, we have:

$$\begin{aligned}\%error &= \frac{measured - predicted}{predicted} \times 100\% \\ &= \frac{79.6\mu s - 78.5\mu s}{78.5\,\mu s} \times 100\% \\ &= 1.4\%\end{aligned}$$

Your Turn: Test Other Frequencies and Time Constants

Keep in mind that a “low-pass” filter lets low frequency sine waves through, and “filters out” higher frequency sine waves. If the DAC is set to 1/10th the cutoff frequency, the RC circuit’s output signal on CH1 should be almost the same amplitude as the input signal on the DAC/CH2. In contrast, if you make the DAC send a frequency that’s ten times the cutoff frequency, the RC circuit’s output signal on CH1 should be much lower in amplitude.

✓ Try setting the Generate panel’s Frequency field to 159 Hz, and compare the RC circuit’s output amplitude on CH1 to the DAC/CH2 signal amplitude. Adjust the oscilloscope display as needed.

✓ Repeat for 15.9 kHz; the signal’s amplitude should be much lower (attenuated).

You can increase the cutoff frequency by decreasing the value of RC. If you do this, the filter will not reduce the amplitude until the input frequency is higher. For example, decreasing the resistor by a factor of 10 increases cutoff frequency by a factor of 10. So, the value of the 10 kΩ resistor is changed to 1 kΩ in these cutoff frequency calculations:

$$\begin{aligned}f_C &= \frac{1}{2\pi RC} \\ &= \frac{1}{2 \times 3.1416 \times 1{,}000 \times 0.00000001} \\ &\approx 15{,}915\,Hz\end{aligned}$$

- ✓ Replace the 10 kΩ resistor in Figure 8-38 and Figure 8-39 with a 1 kΩ resistor and repeat the calculations and tests in this activity for the higher cutoff frequency.

You could also reduce the capacitor's value for the same effect, except that the Understanding Signals kit does not contain a 0.001 µF = 1 nF capacitor. You could however, leave the resistor alone and swap out the 0.01 µF capacitor for a 0.1 µF capacitor for a cutoff frequency that's ten time lower (instead of higher).

SUMMARY

This chapter introduced RC circuits and their relationship to the exponential decay equation. This relationship was examined with oscilloscope measurements, verifying the importance of the RC time constant.

This understanding of RC circuits is useful for measuring the output of devices that vary in resistance or capacitance. Using the PBASIC **`RCTIME`** command, the position of a potentiometer dial—a variable resistor—was measured. The same programming technique was also used to measure a phototransistor's output current, which resulted in a more linear decay.

We took a closer look at some circuits used in previous chapters. The RC DAC's role in D/A conversion was examined, along with its susceptibility to resistive loads. The low-pass filter was also revisited, this time, with attention to testing its cutoff frequency.

Chapter 9: Op Amp Building Blocks

OPERATIONAL AMPLIFIERS

Operational amplifiers (op amps) are versatile electronic building blocks that can be configured with circuits to perform a wide variety of operations on signals. Examples in this chapter include:

- Comparing two voltages
- Buffering a voltage signal that cannot drive a resistive load directly (like the RC DAC circuit from Chapter 8, Activity #5)
- Amplifying an input voltage
- Inverting an input voltage
- Attenuating and invert an input voltage
- Adding or removing offset from a signal

Figure 9-1 shows a picture of the Understanding Signals Kit's LM358 op amp along with a pin map similar to one you might find in the device's datasheet. This integrated circuit (IC) has two op amps, shown as triangles labeled A and B. The activities in this chapter will focus on using one op amp to perform operations on one or two signals. Each op amp can be configured separately, and they can both perform operations on signals in parallel. They can also be cascaded to perform a series of operations on one or more signals.

Figure 9-1: LM358 IC Photo and Pin Map

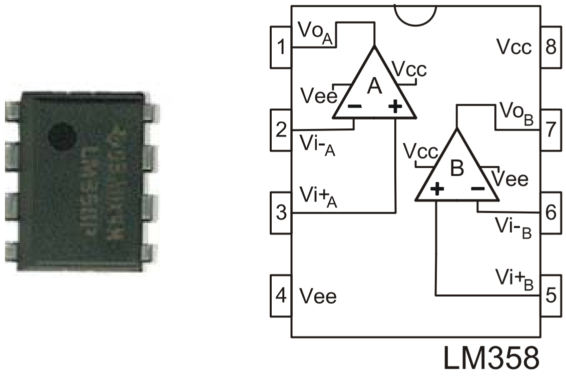


Figure 9-2 shows the op amp symbol with each of its connections labeled. The numbers by each terminal correspond to the pin map numbers for op amp A in Figure 9-1. The op amp has two supply voltage inputs, one positive and one negative. It also has two signal inputs. The one with a (+) label is called the *non-inverting* input, and the one with a (–) label is called the *inverting* input. The notations for some of the terminals in Figure 9-2 might vary from one datasheet to the next as well as from one manufacturer to the next, but all the LM358 manufacturers and their datasheets follow the same conventions for the terminals' names and their positions around the triangular op amp circuit symbol. The one exception to this is that the top/bottom positions of the non-inverting and inverting input terminals may be swapped as needed to simplify schematics.

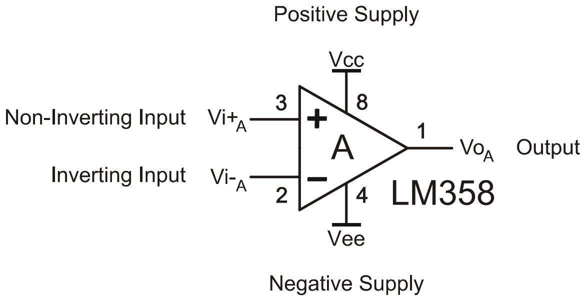


Figure 9-2
Op Amp Terminals

An operational amplifier's *output voltage* is the difference between the voltages at the non-inverting and inverting inputs multiplied by a very large value. For example, the Fairchild LM358 datasheet specifies a "large signal gain" of 100 V/mV. In other words, the output voltage will be 100,000 times the voltage difference between the two inputs. This gain is also called *open loop gain*, and will be utilized by a circuit called a comparator in Activity #1.

After the open loop comparator operation, other operations in the list (buffer, amplify, invert, etc.) require circuits that connect the op amp's output to its inverting input. The circuit configuration determines the operation, and the values of the components determine the relationship between input and output voltage levels and signal amplitudes. In these circuits, the output signal is "fed back" into the inverting input, either directly or through a circuit, so they are called *negative feedback circuits*. Activity #2 through Activity #4 utilize negative feedback circuits to buffer, amplify, invert and offset signals.

Op amp shopping tip:

Open loop gain is not just important for op amp comparator operations; it's also an ingredient in negative feedback applications. An op amp's open loop gain decreases with higher frequency signals, and as that gain decreases, so does its performance. Each op amp's datasheet has one or more graphs of its open loop gain's response to frequency. When shopping for op amps, make sure to check those graphs and verify that the open loop gain is still high in your application's frequency range.

ACTIVITY #1: COMPARATOR

An op amp *comparator* "compares" the voltage difference between an op amp's inverting and non-inverting inputs. If the non-inverting input voltage is greater than the inverting input voltage, the comparator sends a high signal. If the non-inverting input voltage is less than the inverting input voltage, it sends a low signal. With a high open loop gain, the comparator will detect even minute differences in voltage and send high or low signals as a result. This is useful for converting very small voltage differences into binary values.

The comparator's high and low signals are like the binary high/low signals we programmed BASIC Stamp I/O pins to transmit in earlier chapters. However, the voltages of the comparator's high and low output signals are not necessarily 5 and 0 V. Instead, they are determined by a combination of the op amp's supply voltages and its output characteristics.

An example of a comparator application is a clock signal. The small voltage fluctuations from oscillator circuits can be passed through comparators to create high/low clock signals for computing systems. Comparators can also be used with sensors. A threshold voltage can be set at one of the comparator's inputs, and if the sensor's voltage output rises above the level, the comparator sends a high signal, otherwise, it sends a low signal.

In this activity, an LM358 op amp will be configured to function as a comparator. A voltage divider will be used to set the voltage at its non-inverting terminal to 2.5 V. With no feedback, even a small difference above or below 2.5 V will result in the op amp trying to amplify the voltage difference by 100,000. The LM358's output voltage is limited by its supply voltage, so when the output reaches one of those limits, it can't go any further. The result will be that a voltage that's above the voltage applied to the inverting terminal will make the comparator send a high signal, and a voltage that's below it will result in a low signal.

Comparator Test Parts

(1) Potentiometer – 10 kΩ (103)
(2) Resistors – 1 kΩ (brown-black-red)
(1) Resistor – 220 Ω
(1) Op Amp – LM358
(misc.) Jumper wires

Comparator Test Circuit

Figure 9-3 shows a schematic of a 2.5 V threshold comparator and Figure 9-4 shows a wiring diagram example of the circuit. The fixed voltage divider (the two 1 kΩ resistors) applies approximately 2.5 V to the op amp's inverting input. The potentiometer is connected as a variable voltage divider, so turning the knob varies the voltage at its wiper terminal. If the potentiometer wiper's voltage is even slightly above the voltage at the comparator's non-inverting input, the op amp's output will send high signal that's about 3.6 V. If the potentiometer wiper's voltage is slightly below the 1 kΩ voltage divider output, the op amp's output will send a 0 V low signal.

- ✓ **STOP:** If your CH2 probe's BNC end is still connected to the DAC Card's function generator output, disconnect it and reconnect it to the PropScope's CH2 BNC input before continuing.
- ✓ Build the circuit shown in Figure 9-3 and Figure 9-4.

Supply Voltage Plus "Headroom"

When the LM358 sends a high signal, it's only 3.6 V instead of the Vdd = 5 V applied to its Vcc power input. Many op amp outputs require this type of headroom between their maximum output levels and their supply rails. Op amps with "rail to rail" outputs are available, but they are typically a little more expensive. The added cost is not uusally too steep for projects and prototypes, but for products with high sales volumes, an extra 50 cents per product could add up very quickly.

The 220 Ω resistor ensures that the DAC Card cannot be damaged if the CH2 probe's BNC end is inadvertently left connected to the function generator output. Since no current flows into or out of CH2, it will not have any other effects on the circuit or probe measurements.

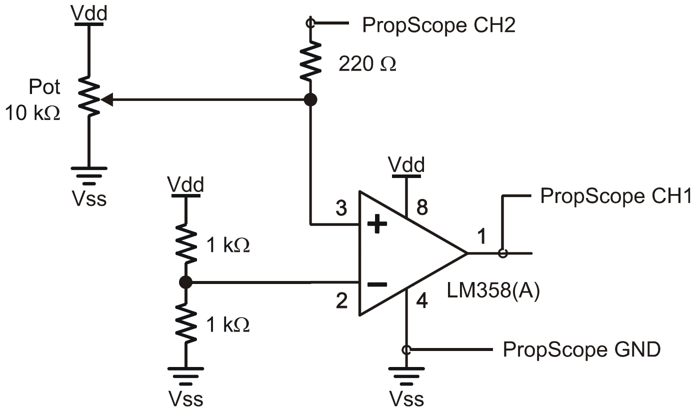
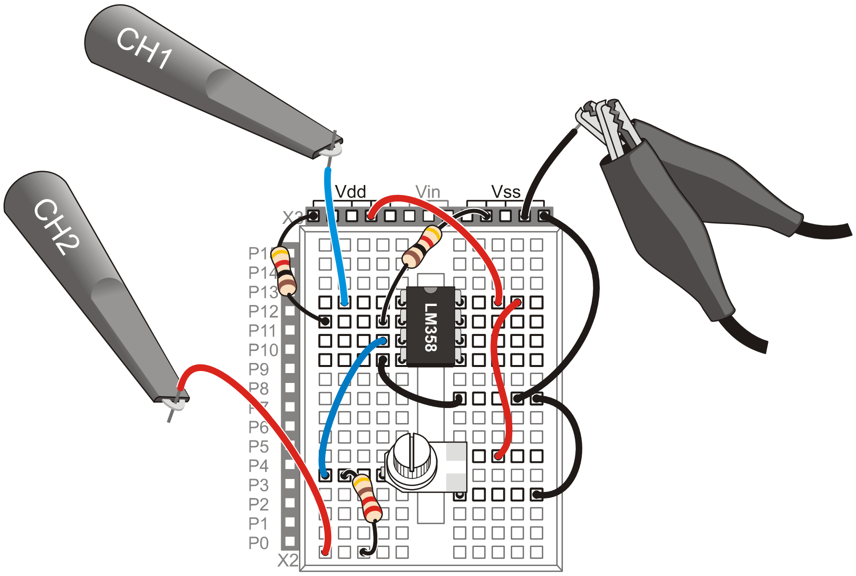


Figure 9-3
Comparator Test Schematic

Figure 9-4: Wiring Diagram Example of Figure 9-3



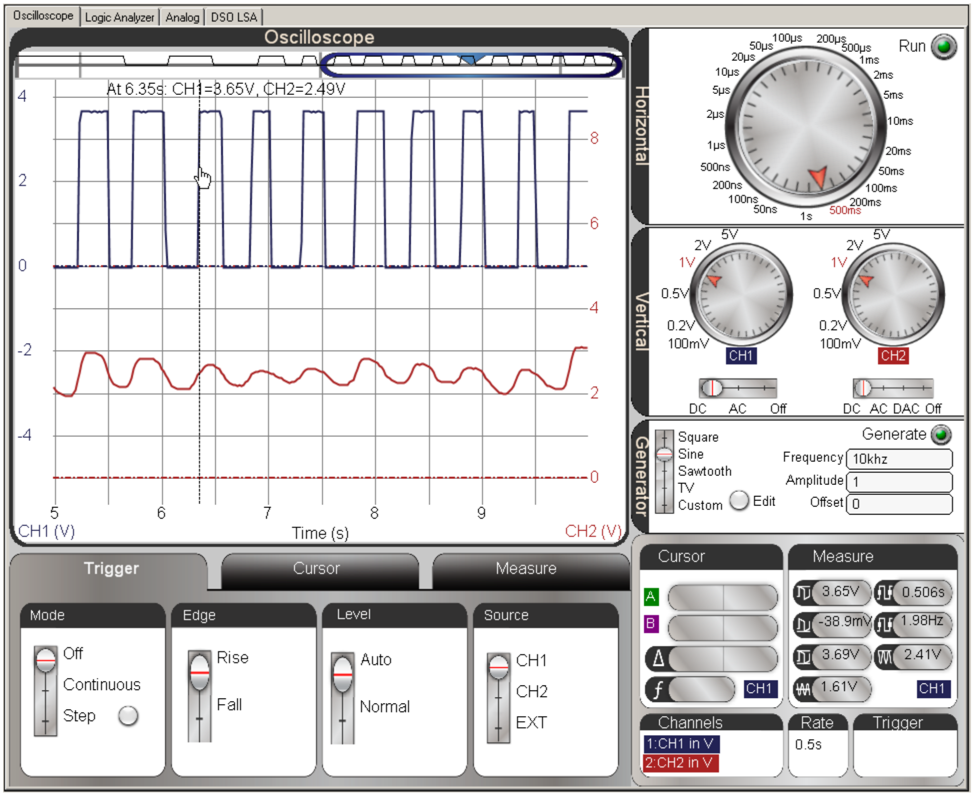
Comparator Test Measurements

Figure 9-5 shows an example of how slight fluctuations at the op amp comparator's non-inverting input result in high and low signals at the output. The lower, red CH2 trace shows the potentiometer wiper terminal's voltage as it is adjusted above and below the

2.5 V threshold, and the upper, blue CH1 trace shows the op amp comparator's output responding by switching high and low.

- ✓ Adjust the Horizontal, Vertical, and Trigger settings as shown in Figure 9-5.
- ✓ Slide the Plot Area bar to the far right of the Plot Preview so you can see immediate voltage changes as you twist the potentiometer's knob.
- ✓ Use the CH2 trace voltage scale on the right as a guide for adjusting the potentiometer's voltage into the 2 to 3 V range.
- ✓ Adjust the potentiometer back and forth in the 2 to 3 V range and verify that the comparator circuit's output responds with high/low signals.

Figure 9-5: 2.5 V Threshold Crossings Cause High/Low Signals at Comparator Output



At 6.35s: CH1=3.65V, CH2=2.49V

Generator: Frequency 10khz, Amplitude 1, Offset 0

Measure: 3.65V, 0.506s, -38.9mV, 1.98Hz, 3.69V, 2.41V, 1.61V, CH1, CH1

Channels: 1:CH1 in V, 2:CH2 in V

Rate: 0.5s

Your Turn: Verify the Threshold Voltage

The gold bars on the 1 kΩ resistors indicate a 5 % *tolerance*. That means their actual values may be up to 5% above or below their *nominal* (named) values. These variations from 1 kΩ typically cause the voltage divider's output to be close to, but not exactly 2.5 V. In Figure 9-5, the mouse is pointing at one of the transitions and the floating cursor info near the top of the Oscilloscope screen is reporting the CH2 voltage as 2.49 V. This voltage will vary from one transition to the next, but an average of these measurements can be taken to approximate the threshold. This value can be compared to a direct measurement of the threshold voltage.

- ✓ Use the CH2 probe to test the voltage at the LM358 pin 2. This is the resistor voltage divider output, and it's the actual threshold voltage.
- ✓ You may need to click the Run button to get the display to hold still.
- ✓ Try averaging the floating cursor values at the comparator output transitions to approximate the resistor divider's threshold voltage.
- ✓ Compare the averaged instantaneous threshold voltage measurement to the direct measurement.

ACTIVITY #2: VOLTAGE FOLLOWER AS AN OUTPUT BUFFER

This activity examines a closed loop op amp circuit called a *voltage follower*, so named because the output voltage "follows" the input voltage. In other words, the op amp's output voltage should be the same as the voltage applied to its non-inverting input. This circuit is also commonly called a *buffer*. That name comes from the fact that the op amp buffers or protects the circuit setting the non-inverting input voltage from any load that might be connected to the op amp's output.

Chapter 8, Activity #5 examined the effect of a resistor load on the RC DAC circuit voltage output. After PWM charged the capacitor, the load caused the voltage to decay, making it an unreliable DC voltage source. In this activity, you will use the same DAC circuit to create a voltage, along with an op amp voltage follower to prevent decay and maintain the voltage across an LED circuit load.

Buffer Test Parts

(1) Resistor – 1 kΩ (brown-black-red)
(1) Resistor – 470 Ω (yellow-violet-brown)
(1) Capacitor – 1 µF
(1) Op Amp – LM358
(1) LED – any color
(misc.) Jumper wires

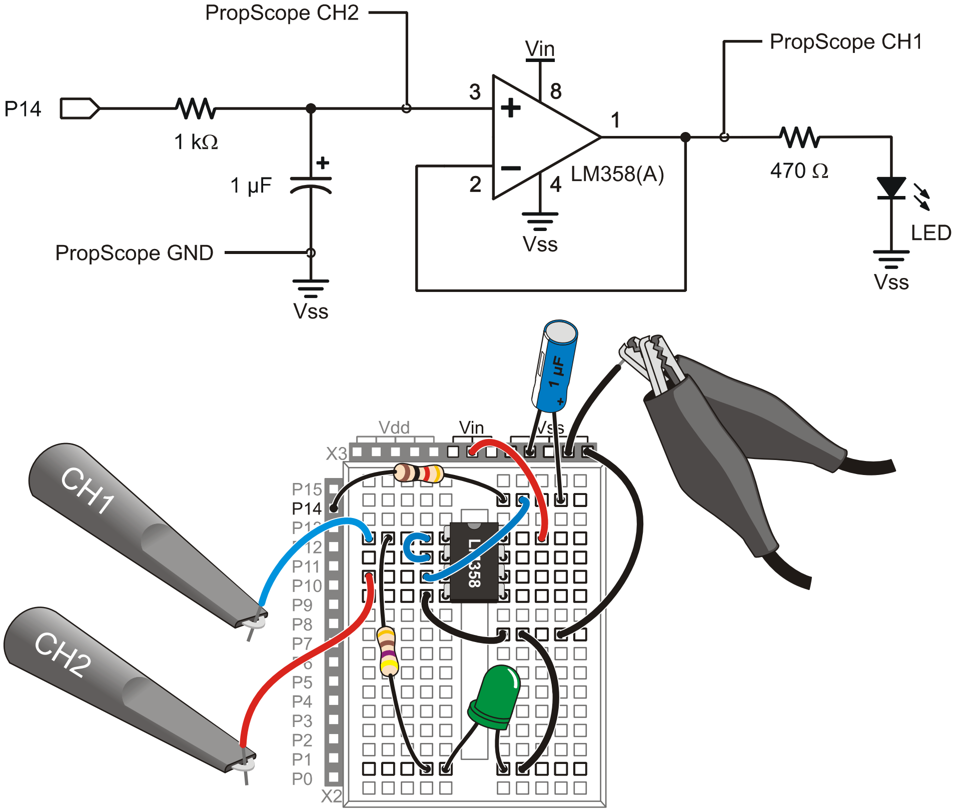
Voltage Follower Test Circuit

Figure 9-6 shows a schematic and wiring diagram example of the voltage follower test circuit with an LED circuit load. One subtle change to this circuit that's easy to miss is that the op amp's Vcc supply input is now connected to Vin. Provided your board's supply is at least 6.5 V, this change should make it possible for the op amp's output to follow the BASIC Stamp RC DAC's voltage through its entire 0 to 4.98 V range.

- ✓ Build the circuit shown in Figure 9-6.
- ✓ Make sure the op amp's pin 8 is connected to one of your board's Vin sockets, instead of a Vdd socket.
- ✓ A 9 V battery supply is recommended for this activity; your supply must be above 6.5 V to give the op amp supply enough headroom to output up to 4.98 V.

The circuit in Figure 9-6 is called a voltage follower because it forces the op amp's output to "follow" the voltage applied to its non-inverting input. The fact that we have made a circuit that matches the input voltage at its output might not seem like a big deal. Heck, a wire could do that, right? The thing a wire cannot do is prevent a load from affecting the circuit it draws current from, but an op amp voltage follower can. With this circuit, the BASIC Stamp can use its PWM command to set voltages across the RC DAC circuit to control the LED's brightness. The op amp's output supplies the necessary current to make its output voltage match the capacitor voltage applied to the non-inverting input, but there is no decay from the LED circuit load like there would be if the voltage follower circuit was replaced with a wire.

Figure 9-6: Voltage Follower Test Circuit



The op amp's inputs have very high input resistances—in the 200 MΩ range. This means that its non-inverting input will place almost no current load on the capacitor in Figure 9-6. So the capacitor will be able to hold its charge almost exactly the same way it did when no load was connected to it. At the voltage follower's output, it doesn't matter whether the load is a 10 kΩ resistor, a 2 kΩ resistor, or even an LED in series with a 470 Ω resistor. The op amp will supply whatever current it takes to keep the voltage at its output terminal the same as the voltage the capacitor applies to its non-inverting terminal. Of course, that's provided the load isn't beyond the amount of current the op amp's output is designed to supply.

How it Works

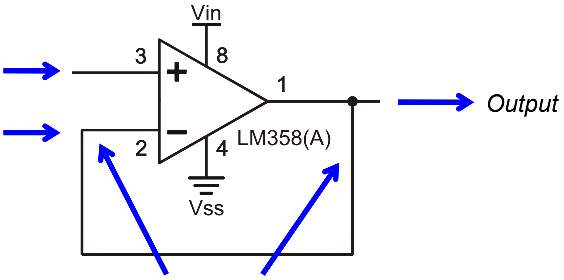
The rule for negative feedback op amps circuits is:

> *Op Amp Negative Feedback Rule: If an op amp's inverting input senses its output through a circuit, the op amp adjusts its output to make the voltage at the inverting input match the voltage at the non-inverting input.*

In Figure 9-7, the op amp's output is connected to its inverting input with a wire, so the output voltage is applied directly to the inverting input—they are the same voltage. To enforce the negative feedback rule, the op amp has to make its output voltage match the voltage applied to its non-inverting input.

Figure 9-7: Negative Feedback in a Voltage Follower Circuit

*To make sure the voltage at the **non-inverting input** is equal to the voltage at the **inverting input**, the op amp has to make its output voltage the same as the voltage applied to the non-inverting input.*



Feedback wire applies output voltage to inverting input, so these two voltages are the same.

Buffer Test Code

One Hz Sine Wave.bs2 sends a 1 Hz sine wave to the P14 DAC indefinitely, making it. convenient to compare the DAC's sine wave output to the voltage follower's output.

- ✓ Enter and run One Hz Sine Wave.bs2

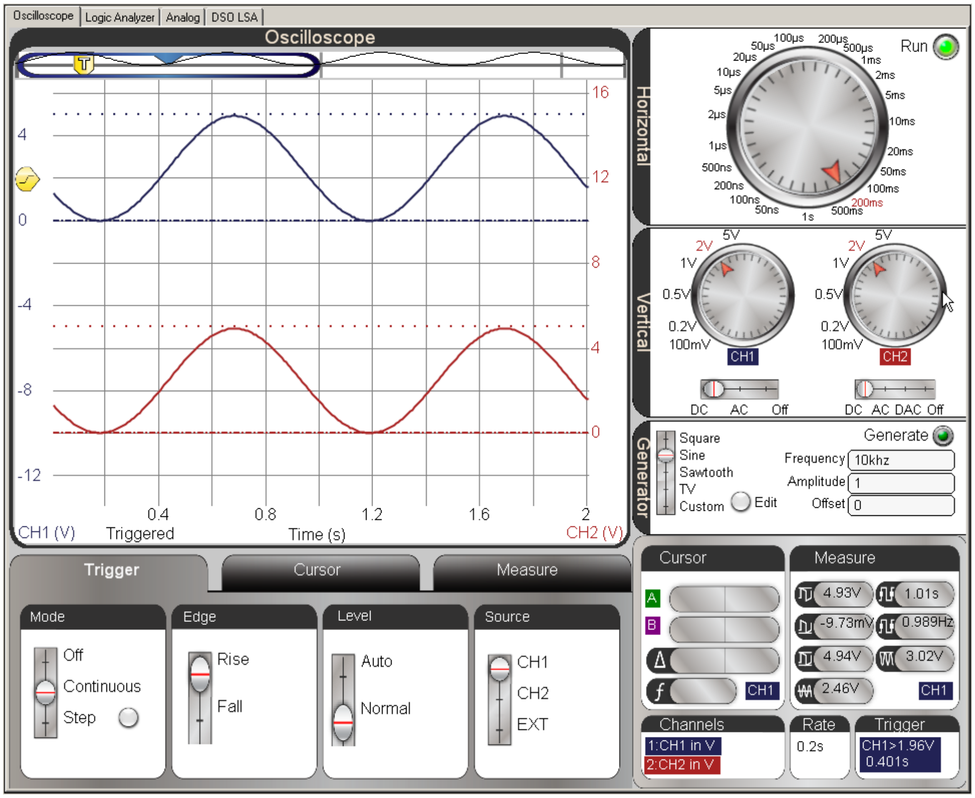
```
' One Hz Sine Wave.bs2
' Transmit a 1 Hz sine wave on P14 indefinitely.
' {$STAMP BS2}                               ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                              ' Language = PBASIC 2.5

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

DO                                           ' Main Loop
  FREQOUT 14, 60000, 1                       ' Play 1 Hz for 1 minute
LOOP                                         ' Repeat main loop
```

Buffer Test Measurements

Figure 9-8 shows the RC DAC output on the lower, red CH2 trace and the voltage follower's copy of that signal on the upper, blue CH1 trace. Note that the sine waves are the same amplitude and phase. This indicates that the op amp's output is "following" the DAC voltage applied to its non-inverting input, even with the LED current load.

- ✓ Adjust your Oscilloscope's Horizontal, Vertical and Trigger settings as to match Figure 9-8.
- ✓ Verify that the DAC's sine wave is identical to the buffer output's copy of it.
- ✓ Verify that the LED gets brighter and then dimmer at roughly once per second.

Figure 9-8: Buffer Output = DAC Output



Horizontal: 200ms

Vertical: CH1 2V, DC; CH2 2V, DC

Generator: Frequency 10khz; Amplitude 1; Offset 0

Trigger: Mode Continuous; Edge Rise; Level Normal; Source CH1

Measure: 4.93V; 1.01s; -9.73mV; 0.989Hz; 4.94V; 3.02V; 2.46V; CH1

Channels: 1:CH1 in V; 2:CH2 in V

Rate: 0.2s

Trigger: CH1>1.96V 0.401s

Your Turn: Other RC DAC Waveforms & Chapter 8, Activity #5 Comparison

The LED circuit transmitting the sine wave is interesting because it demonstrates the BASIC Stamp module's ability to control brightness with the op amp-buffered DAC circuit. However, the sine wave is just one example of many that you can create with the BASIC Stamp. Programs that cause variations in the `PWM` command's ***Duty*** argument can be used to create other signals. Back in Chapter 3, Activity #4, you used the `PWM` command and an RC DAC circuit to create triangle and sawtooth waves. At that point, we could only measure the varying voltages, because without the buffer, the RC DAC circuit could not drive an LED load. Now we have a buffer, so those programs can actually apply those voltage signals to the LED circuit at the op amp's output.

- ✓ Try Saw Tooth.bs2 from Chapter 3, Activity #4. (No circuit changes or oscilloscope setting changes are needed, just run the program.)
- ✓ Repeat the code modifications from that activity that make the sawtooth a triangle wave and examine that too.

ACTIVITY #3: NON-INVERTING AMPLIFIER

The previous activity mentioned a rule for op amp circuits with negative feedback that is very important.

> *Op Amp Negative Feedback Rule: If an op amp's inverting input senses its output through a circuit, the op amp adjusts its output to make the voltage at the inverting input match the voltage at the non-inverting input.*

The voltage follower circuit in the previous activity was the first example of this rule. The op amp's output was connected to its inverting input, which forced its output to always match the voltage applied to its non-inverting input.

The non-inverting amplifier is another application of the op amp negative feedback rule. Take a look at the voltage divider at the op amp's output in Figure 9-9. It sends some fraction of the op amp's output voltage back to the inverting input. So, to get the voltage at the inverting input to match the voltage at the non-inverting input, the op amp's output has to send a larger voltage. The net effect is that it amplifies the output signal. For example, if Rf and Ri are the same value, the voltage between Rf and Ri would be ½ of Vo. So, to make the voltage at the inverting (–) terminal match the voltage at the non-inverting (+) terminal, the op amp has to make Vo twice the value of Vi, the voltage applied to the non-inverting terminal.

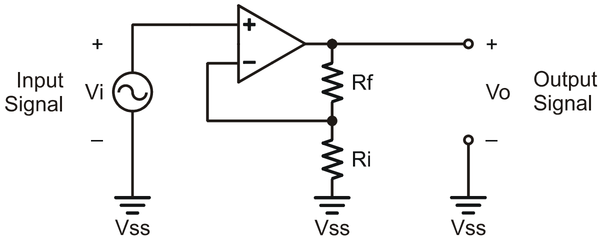
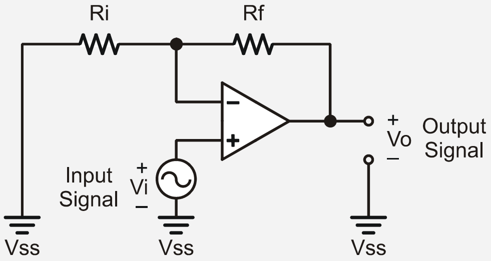


Figure 9-9
Non-Inverting Amplifier

Feedback and Input Resistors: Rf is called the feedback resistor and Ri is called the input resistor.

Common Textbook Version of the Same Circuit:. Compare the circuit below to the one in Figure 9-9. They are the same circuit, just drawn differently.



9

Understanding and Applying Gain

The amount an amplifier "amplifies" voltage is measured as *gain*, and it's the ratio of output voltage (Vo) to input voltage (Vi). You can also think about it as the ratio of output signal amplitude to input signal amplitude.

$$Gain = \frac{Vo}{Vi}$$

Gain is a convenient value for predicting output voltage based on input voltage. Simply multiply both sides of the Gain ratio by Vi, and you get an equation that predicts Vo. This equation is called a *transfer function*.

$$Vo = Gain \times Vi$$

Gain can also be calculated from the values of the feedback and input resistors. For example, a non-inverting amplifier's gain is:

$$Gain = 1 + \frac{Rf}{Ri}$$

We already know because of how voltage dividers work that the non-inverting amplifier's output will be twice its input if Rf and Ri are equal. In other words, we expect the amplifier circuit's gain to be 2. Let's try two 10 kΩ resistors and calculate the gain.

$$\begin{aligned} Gain &= 1 + \frac{Rf}{Ri} \\ &= 1 + \frac{10\,k\Omega}{10\,k\Omega} \\ &= 2 \end{aligned}$$

If the gain is 2, we can predict the output for a given input with the transfer function:

$$\begin{aligned} Vo &= Gain \times Vi \\ &= 2 \times Vi \end{aligned}$$

This activity will start with Rf = 10 kΩ and Ri = 10 kΩ, and also test with Rf = 20 kΩ and Ri = 10 kΩ.

✓ Repeat this calculations for Rf = 20 kΩ and Ri = 10 kΩ. What will the gain be with this different resistor combination?

Keep in mind that these voltage equations only apply if the output is within the limits set by the op amp's Vcc and Vee power supply rails. For the LM358, that's Vee $\leq$ Vo $\leq$ Vcc – 1.4 V. If Vo is outside that range, it will simply stop at the limit imposed by the supply rail.

The minimum gain for a non-inverting amplifier is 1. So it cannot attenuate a signal, only amplify. For a gain of 1, Rf has to be 0 Ω, which is a wire. So, a voltage follower is really a special case of non-inverting amplifier.

Non-Inverting Amplifier Test Parts

(2) Resistors – 10 kΩ (brown-black-orange)
(1) Resistor – 20 kΩ (red-black-orange)
(1) Op Amp – LM358
(misc.) Jumper wires

Non-Inverting Amplifier Test Circuit

Figure 9-10 shows a test circuit for our non-inverting amplifier, and Figure 9-11 shows an example of the wired circuit. The PropScope's function generator will send a test signal to the op amp's non-inverting input, and CH1 will monitor the resulting signal at the op amp circuit's output.

- ✓ **STOP:** Disconnect the CH2 probe from the breadboard first.
- ✓ Disconnect the probe from the PropScope's CH2 BNC connector in, and connect it to the DAC Card's function generator output. See Figure 2-16 on page 46.
- ✓ Build the circuit in Figure 9-10, using Figure 9-11 as a guide. (Set the 20 kΩ resistor aside for the moment.)

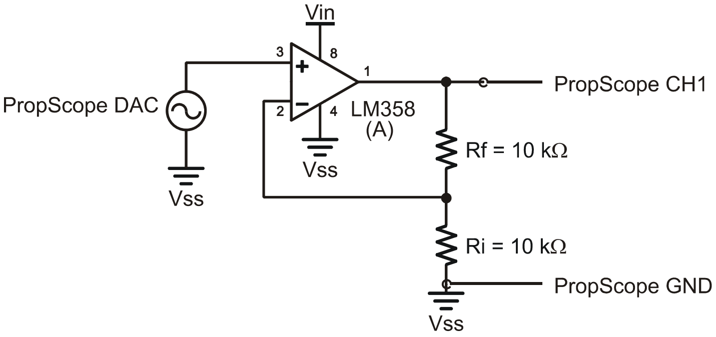


Figure 9-10
Non-Inverting Amplifier Test Circuit with a Gain of 2

Figure 9-11: Wiring Diagram Example of Figure 9-10

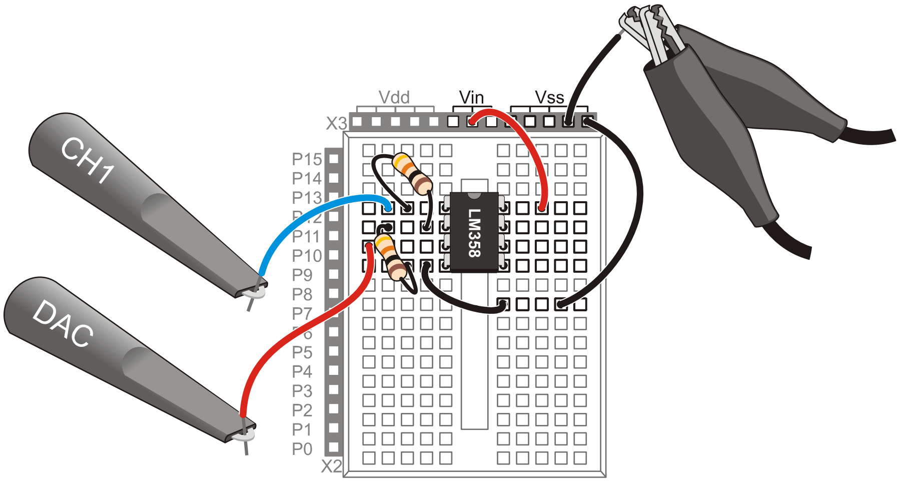


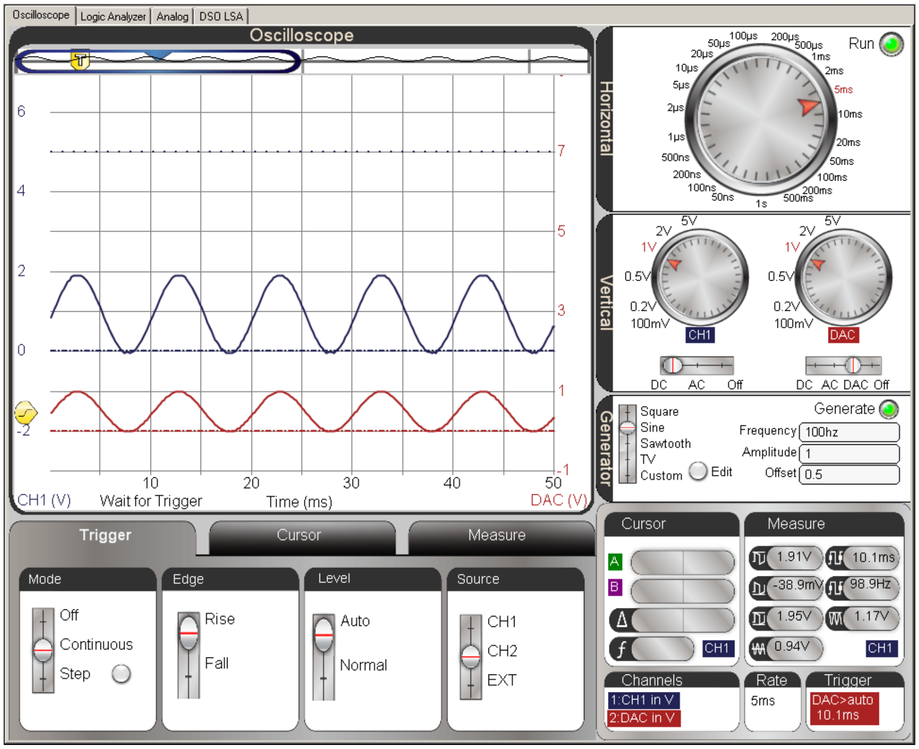
Figure 9-12 shows a quick amplitude check for a 100 Hz, 1 Vpp, 0.5 V offset sine wave generated by the DAC Card and displayed in the lower, red CH2 trace. The amplitude of the output in the upper, blue CH1 trace is approximately 2 Vpp, so it verifies the gain of 2 with Rf = Ri = 10 kΩ.

- ✓ Configure the PropScope's Horizontal, Vertical, Generator and Trigger settings according to Figure 9-12.
- ✓ Make sure to click the Generator button to start the DAC Card's signal.

Figure 9-13 shows what happens to the output signal when you replace the 10 kΩ feedback resistor (Rf) with a 20 kΩ one. The gain increased to 3.

- ✓ Replace the 10 kΩ Rf feedback resistor in Figure 9-10 on page 335 with one that's 20 kΩ. It's the top resistor in the Figure 9-11 wiring diagram.
- ✓ Verify that this resistor change results in gain change, from 2 to 3.
- ✓ Try increasing the input signal's offset to 0.75 and its amplitude to 1.5 in the Generator panel. How did the output respond?

Figure 9-12: Non-inverting Amplifier Gain Test with Rf = Ri = 10 kΩ



Frequency 100hz

Amplitude 1

Offset 0.5

Measure: 1.91V | 10.1ms | -38.9mV | 98.9Hz | 1.95V | 1.17V | 0.94V | CH1

Cursor: CH1

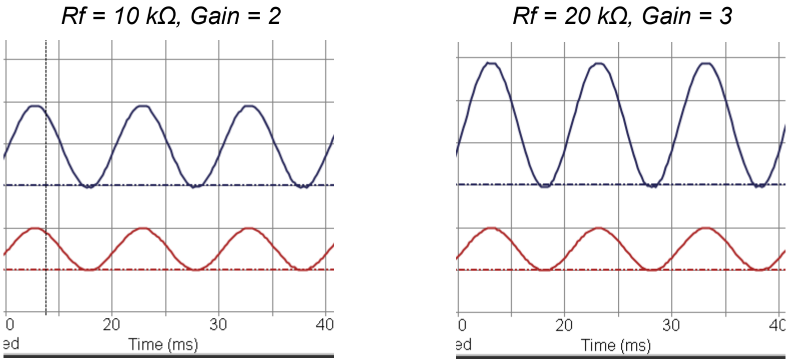
Channels: 1:CH1 in V, 2:DAC in V

Rate: 5ms

Trigger: DAC>auto 10.1ms

9

Figure 9-13: Gain with Two Different Feedback Resistor Values


Rf = 10 kΩ, Gain = 2

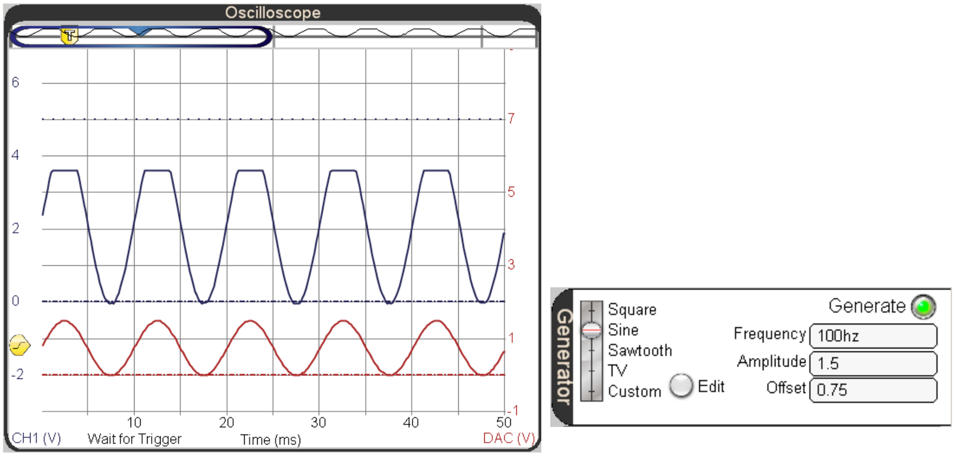
Rf = 20 kΩ, Gain = 3

Op Amp Supply Voltages and Clipping Signal Distortion

Remember that the upper limit for the op amp's output signal is 1.5 V less than Vcc, and the lower limit is Vee. Since Vcc is connected to the your board's Vin supply, that maximum voltage would be around 9 V – 1.5 V = 7.5 V, assuming you are using a 9 V battery. An output signal that swings from 0 to 4.5 V is well within that range, but what happens if you change the op amp's Vcc connection from Vin ≈ 9 V to Vdd = 5 V? Figure 9-14 shows an example. Notice that the output signal in the upper CH1 trace reaches the op amp's voltage limit and just levels off. It can't go any higher. This symptom is a form of signal distortion called *clipping* because the tops of the sine wave appeared to be "clipped" off. (Notice there is no red CLIPPED! warning; unlike Chapter 2, Activity #3, the signal is being clipped by the op-amp itself, and not by the PropScope software's voltage scale setting.)

- ✓ Make sure your Generator panel is set to Offset = 0.75 V and Amplitude = 1.5 V.
- ✓ Disconnect the end of the jumper wire plugged into Vin and plug it into Vdd instead.
- ✓ Verify that your trace resembles Figure 9-14.
- ✓ For clipping at the op amp output's lower power supply limit, try Amplitude = 1 V and Offset 0.25 V.
- ✓ Reconnect the op amp's Vee positive supply input to Vin before continuing.

Figure 9-14: Output Signal Clipping



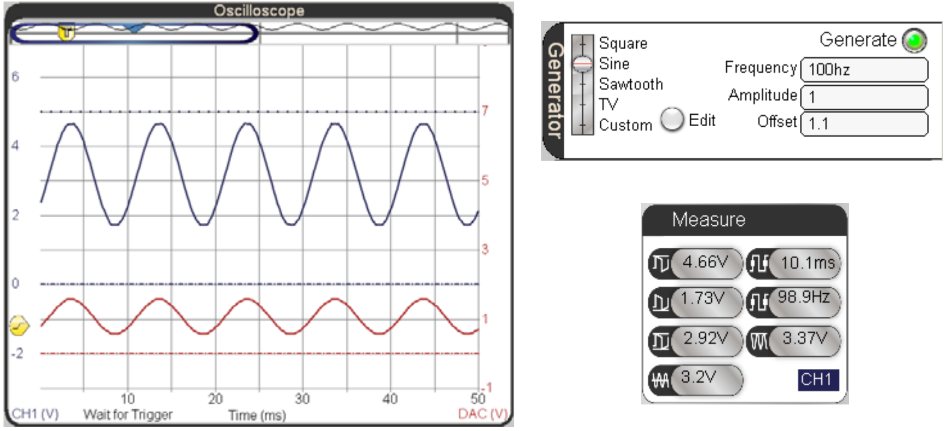
Frequency 100hz
Amplitude 1.5
Offset 0.75

Your Turn: How the Op Amp Translates Offset Voltage

Figure 9-15 shows an example of how a non-inverting op amp circuit amplifies both the offset and the amplitude. The offset of the input sine wave is 1.1 V. Since the gain of the circuit is 3, the amplitude of the sine wave is about 3 V peak to peak, and the offset is almost 1.1 × 3 = 3.3 V.

- ✓ Change the Generator settings back to Amplitude = 1 V and Offset = 0.5 V.
- ✓ Increase the Offset from 0.5 to 1.2 V in increments of 0.1 V, and observe the signal after each increase. The signal's amplitude should stay at 3 Vpp, but the offset should increase by 0.3 V for every 0.1 V increase in your input signal's offset.

Figure 9-15: 1.1 V Offset Amplified to almost 3.3 V



Frequency 100hz
Amplitude 1
Offset 1.1

4.66V 10.1ms
1.73V 98.9Hz
2.92V 3.37V
3.2V CH1

It's also useful to look at just a DC signal amplified.

- ✓ Set the Trigger Mode to Off.
- ✓ Change the Generator Amplitude to 0 V.
- ✓ Apply different DC voltages to the amplifier's input by entering values into the Offset field. Try 0.5 to 1.2 V in increments of 0.1 V again.

ACTIVITY #4: INVERTING AMPLIFIER

The input signals up to this point have been fed to the op amp's non-inverting input. In contrast, the inverting amplifier circuit feeds the input signal to the amplifier's inverting input. Figure 9-16 shows the inverting amplifier circuit. This is still a negative feedback amplifier because the inverting input is still connected to the output through Rf.

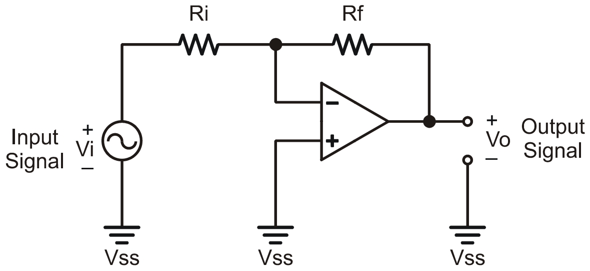


Figure 9-16
Inverting Amplifier

Remember the rule of negative feedback with op amps, the output will adjust to keep the voltage at the inverting input equal to the voltage at the non-inverting input. Let's say the resistors are equal, and Vi = +2 V. In that case, the output would have to transmit –2 V to keep the voltage at the inverting input equal to 0 V, which is the value at the non-inverting input. Another example with Rf = Ri, if Vi is –1 V, Vo has to be +1 V to keep the voltage at the inverting input at 0 V so that it is equal to the non-inverting input. These are two examples where the op amp's output inverts the signal. More generally, the gain for an inverting amplifier is:

$$Gain = -\frac{Rf}{Ri}$$

The negative sign in the gain comes into play when expressing the relationship of output to input signal:

$$Vo = -\frac{Rf}{Ri} \times Vi$$

If we use Rf = Ri = 1, then the output signal Vo is –1 multiplied by the input signal. Substituting Vi = +2 or Vi = –1 verifies the earlier predictions of Vo = –2 V and Vo = +1 V.

In contrast to the non-inverting amplifier, which had a minimum gain of 1, this amplifier can be configured for fractional gains that attenuate the signal by using a value of Ri that's larger than Rf. For example, if Ri = 10 kΩ and Rf = 1 kΩ, the output signal will be attenuated to –Rf/Ri = –1/10 the amplitude of the input signal.

Another difference with possible consequences is the fact that the input signal feeds into a relatively small resistor. The signals in the previous two example circuits were fed into the op amp's non-inverting input, which has resistance in the hundreds of mega ohms. The inverting amplifier in Figure 9-16 has input resistance of Ri, which might be a comparatively low value, like 10 kΩ. While that's fine for the PropScope's function generator, we have already seen how that can cause decay across a DAC circuit's capacitor. For a design that needs an inverted signal out of an RC DAC, two op amp circuits can be cascaded. The DAC's output can be fed into a buffer, and then the buffer's output could be fed into an inverting amplifier.

Inverting Op Amp Test Circuit Parts

(2) Resistors – 1 kΩ (brown-black-red)
(2) Resistors – 10 kΩ (brown-black-orange)
(1) Resistor – 20 kΩ (red-black-orange)
(1) Op amp – LM358
(misc.) Jumper wires

Inverting Op Amp Test Circuit

The op amp circuit in Figure 9-16 is typically implemented with a negative supply voltage connected to the op amp's Vee terminal. That makes it possible for the output voltage to swing above and below ground in an inverted version of the input signal (which also swings above/below ground). Neither the Board of Education nor the HomeWork Board has a negative supply, so our inverting amplifier test circuit in Figure 9-17 and Figure 9-18 will use 2.5 V as a reference voltage instead of 0 V ground. A resistor divider with two 1 kΩ resistors supplies the non-inverting input with 2.5 V. So long as the function generator output is configured with a 2.5 V offset, the op amp's output will be an inverted version of the input signal. The only difference will be that the input and output signals will swing above and below 2.5 V instead of ground.

✓ Build the circuit in Figure 9-17, optionally using the example wiring in Figure 9-18 as a guide.

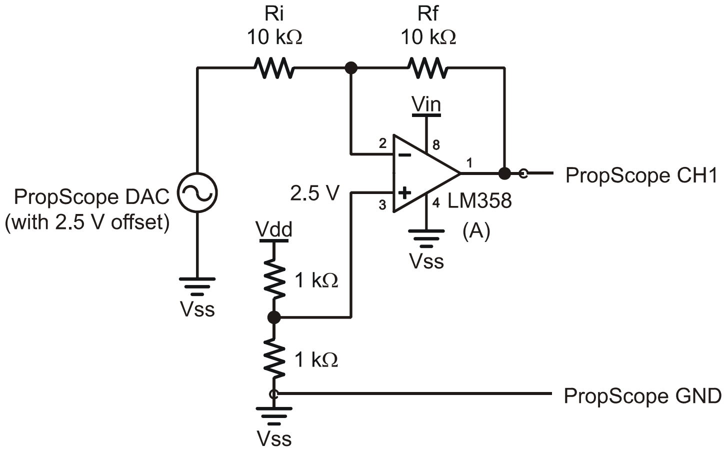


Figure 9-17
Inverting Amplifier
Test Circuit

Figure 9-18: Wiring Diagram Example of Figure 9-17

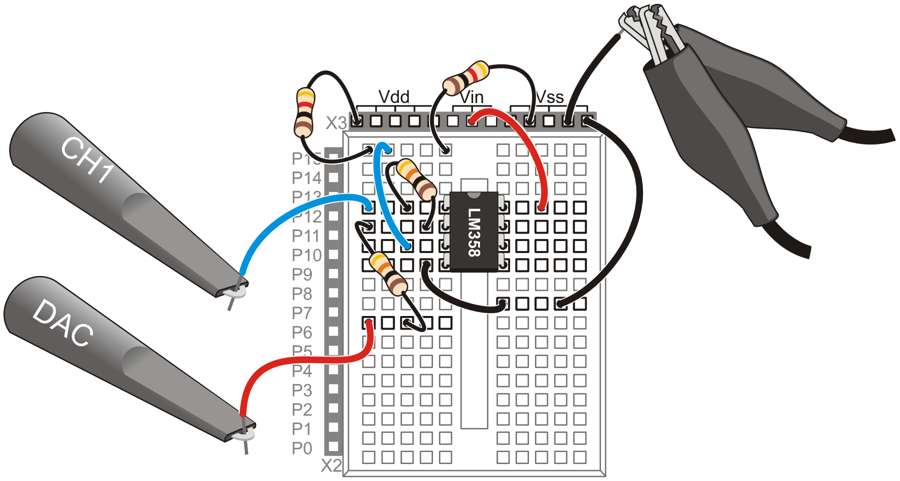
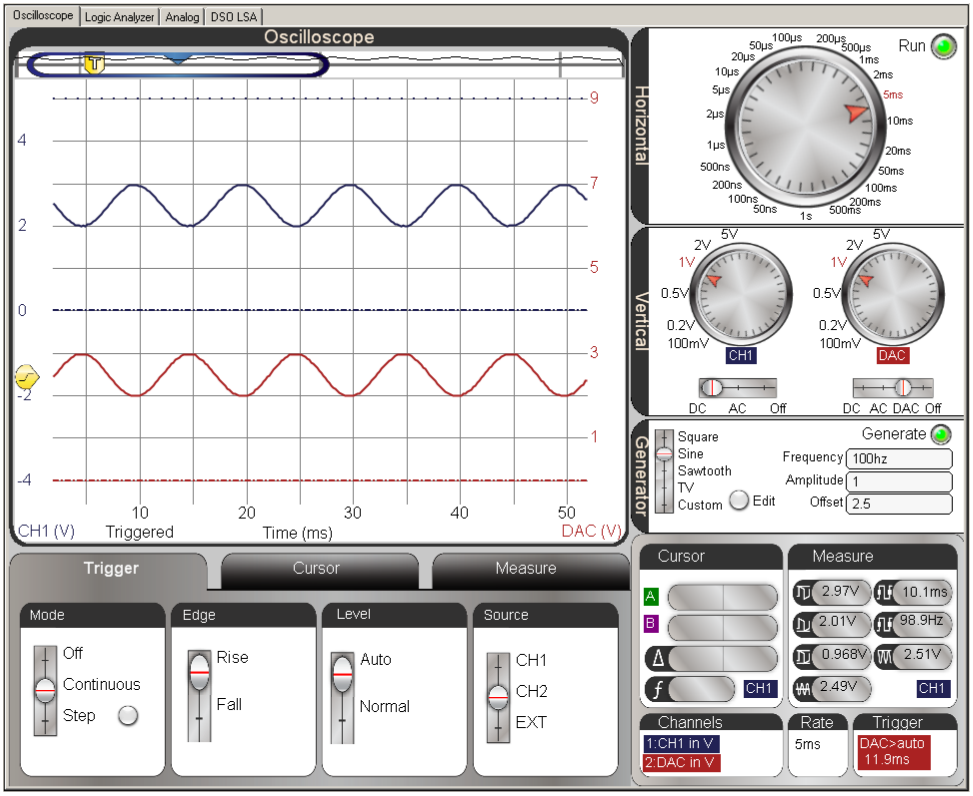


Figure 9-19 shows the lower DAC trace with an amplitude of 1 Vpp and offset of 2.5 V, which is applied to the amplifier's input. The upper CH1 trace is the amplifier's output. Remember that the gain for this amplifier is –Rf/Ri = –10 kΩ ÷ 10 kΩ = –1. That's why the amplifier output signal in the upper CH1 trace is the voltage opposite of the lower DAC trace. Look carefully: the output signal is an inverted version of the input signal with the top of every peak in the upper trace lined up with the bottom of a valley in the lower trace.

- ✓ Adjust the Horizontal, Vertical, Generator and Trigger settings according to Figure 9-20.
- ✓ Verify that CH1 shows an inverted version of the DAC output with the same amplitude.

Figure 9-19: Inverting Amplifier Output with a Gain of –1



Generator: Square, Sine, Sawtooth, TV, Custom
Frequency 100hz
Amplitude 1
Offset 2.5

Trigger Mode: Off, Continuous, Step
Edge: Rise, Fall
Level: Auto, Normal
Source: CH1, CH2, EXT

Cursor: CH1

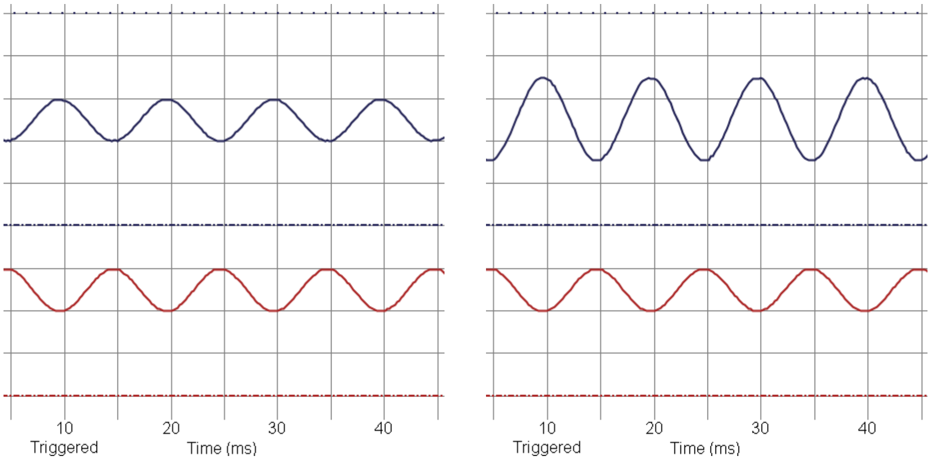
Measure: 2.97V, 10.1ms, 2.01V, 98.9Hz, 0.968V, 2.51V, 2.49V, CH1

Channels: 1:CH1 in V, 2:DAC in V
Rate: 5ms
Trigger: DAC>auto 11.9ms

Figure 9-20 compares the –1 gain amplifier circuit against one with Rf = 20 kΩ and Ri = 10 kΩ. Gain with the 20 kΩ feedback resistor is –Rf /Ri = –20 kΩ ÷ 10 kΩ = –2 and this is verified with the upper trace on the right, which is an inverted 2 Vpp version of the lower 1 Vpp DAC trace that goes to the amplifier's input.

- ✓ Replace the 10 kΩ Rf feedback resistor a 20 kΩ verify the –2 gain.

Figure 9-20: Gain = –1 (left) and -2 (right)



Your Turn: Voltage Offset Response and Other Gain Values

With a gain of 2, and if the input signal's offset is 3 V, (0.5 V above 2.5 V), the output signal should have an offset of 1.5 V. (That's 1 V below 2.5 V.)

- ✓ Test the offset response by adjusting the Offset in the PropScope's Generator panel. Is the prediction correct?

As mentioned earlier, the inverting op amp can also attenuate the signal with fractional gains.

- ✓ Adjust the Generator back to 1 Vpp, 2.5 V offset.
- ✓ Swap the Rf and Ri resistors. Calculate the gain and verify with the PropScope.

Of course, larger magnitude gains are also an option, try this:

- ✓ Rf = 10 kΩ and Ri = 2 kΩ.
- ✓ Calculate the gain.
- ✓ You may need to reduce the DAC signal's amplitude at the op amp circuit's input to prevent clipping at the output.
- ✓ Verify the gain with the Parallax PropScope.

SUMMARY

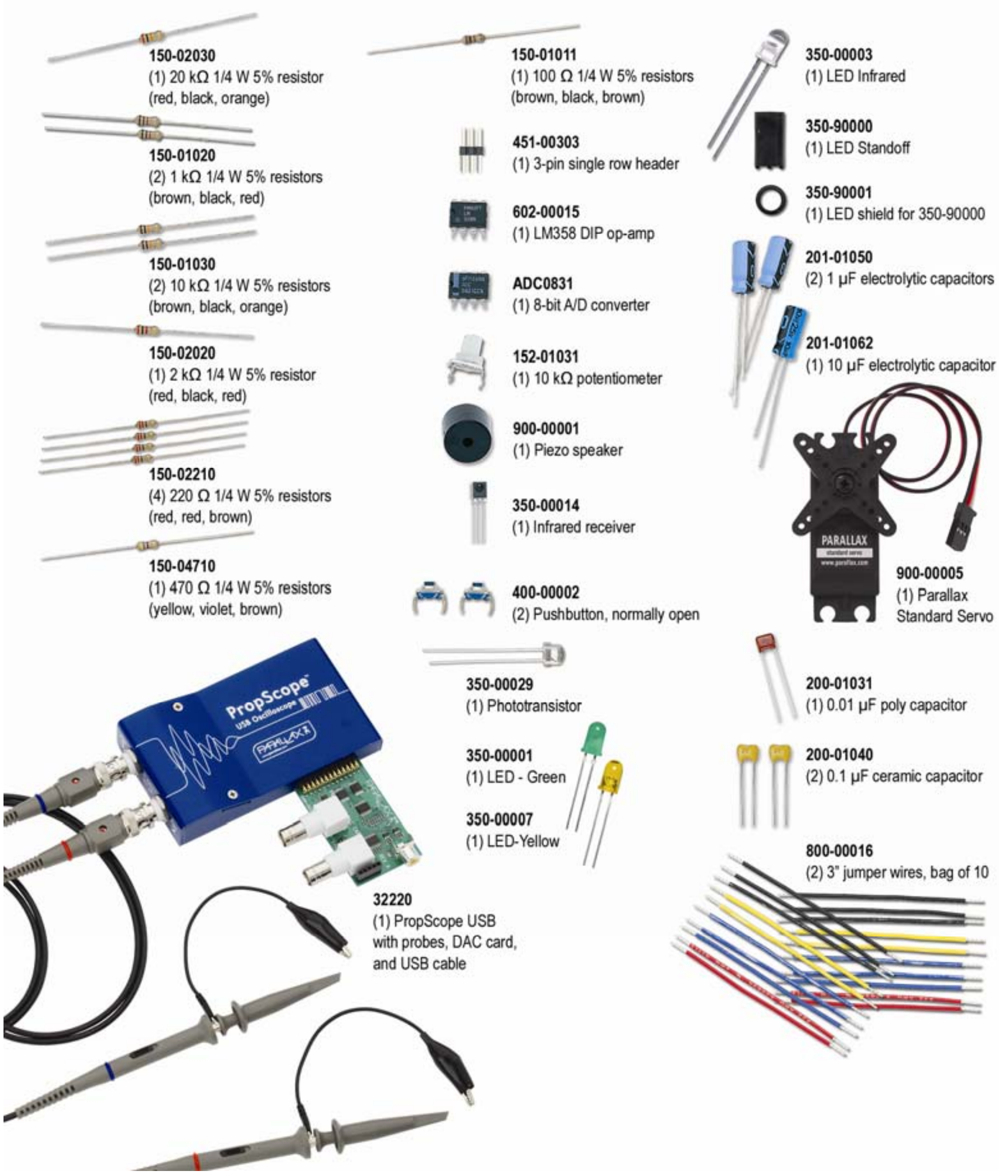
This chapter introduced the operational amplifier or op amp. Op amps can be used to perform a variety of operations on signals including: compare, buffer, amplify, attenuate, invert, and many more. The op amp has a large open loop gain. This chapter demonstrated how the open loop gain can be utilized in a comparator circuit to convert small signal differences into digital outputs.

Many signal operations rely on negative feedback circuits where the op amp's inverting input senses its output through a circuit. When an op amp's inverting input senses the output through a circuit, the op amp adjusts its output to make the voltage at the inverting input match the voltage at the non-inverting input. This behavior was used to demonstrate buffer, non-inverting amplifier, and inverting amplifier circuits.

Clipping is a form of signal distortion where the op amp's output tries to send a voltage that is outside the limitations of its supply voltages.

Gain is a measurement of the ratio of output to input signal amplitude. This value can be multiplied by the input signal to predict the output signal in an equation called a transfer function. For non-inverting and inverting amplifier circuits, the gain can be set by choosing a ratio of feedback and input resistors according to the gain equation for the circuit.

Parts and quantities are subject to change without notice. Parts may differ from what is shown in this picture. If you have any questions about your kit, please email education@parallax.com.



Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Parallax:

32225