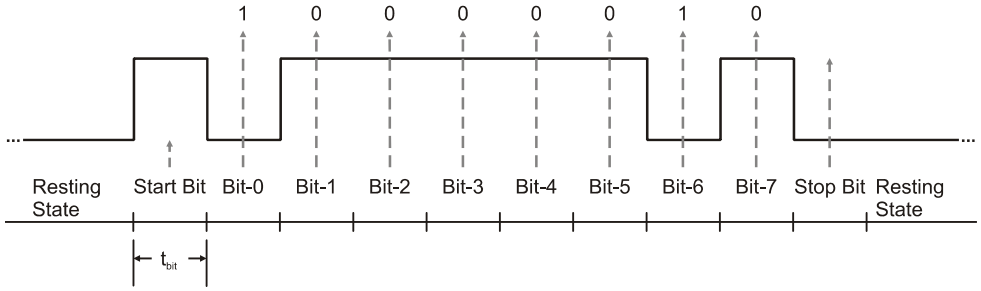
ACTIVITY #4: TRUE VS. INVERTED

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 220.

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 both signals with both oscilloscope probes as shown in the Figure 6-13 schematic and Figure 6-14 wiring diagram example.

- ✓ 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.

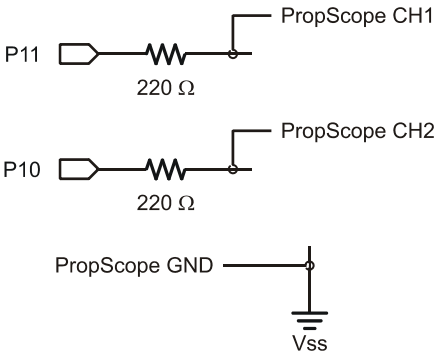


Figure 6-13
Both Channel Probes Connected to I/O Pin Sockets through Protection Resistors

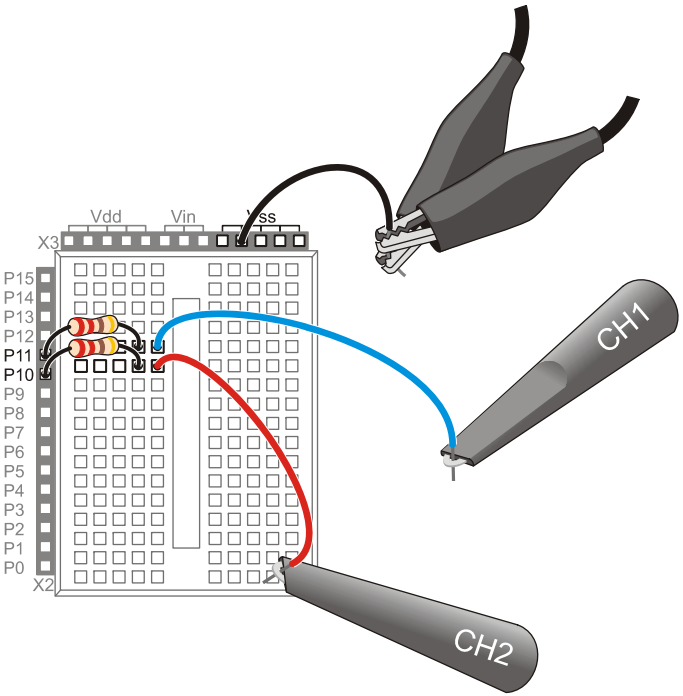


Figure 6-14
Wiring Diagram Example of Figure 6-13

True vs. Inverted Comparison Test 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.

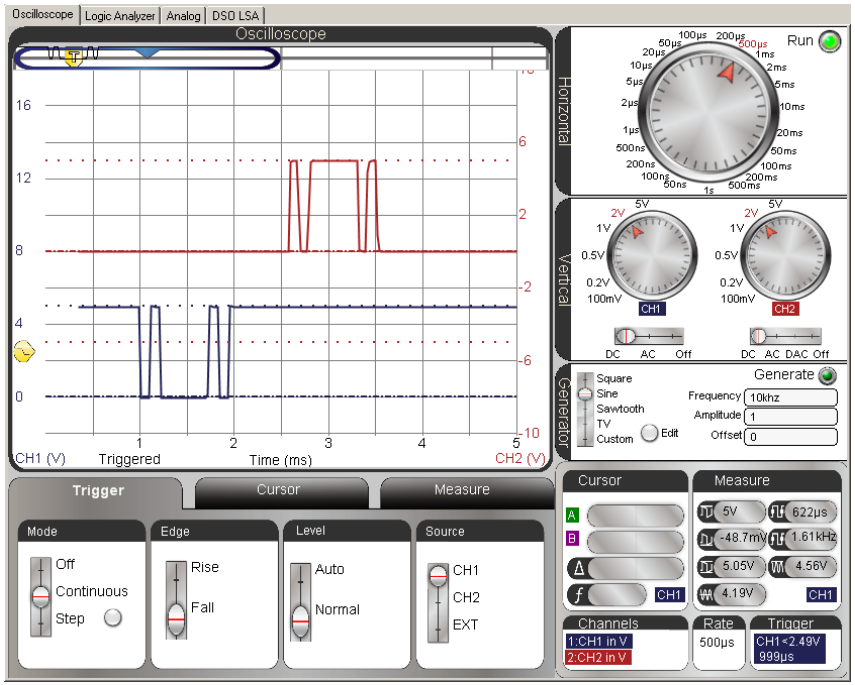
```
' True Inverted Comparison.bs2
' Transmit "A" on P11 with true signaling, and then on P10 with inverted.

' {$STAMP BS2}                                ' Target module = BASIC Stamp 2
' {$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
```

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, either up or down, 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)



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 Preview Bar, and then move the Trigger Time Control to adjust the signal positions. If you drag it to the approximately the 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 and 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 Stamps are connected together and communicating with asynchronous serial communication? Robot applications sometimes use a pair of BASIC Stamps 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, one 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 is busy sometimes and ready to receive serial messages other times. Your BASIC Stamp will be programmed to send serial bytes with hardware flow control enabled. The program will send serial bytes using an I/O pin that is connected to the PropScope's CH1 probe. The program will monitor another I/O pin for hardware flow control high/low signals that the PropScope's function generator transmits. The function generator will transmit a square wave, which will cause the BASIC Stamp to wait whenever the function generator sends a high, busy signal, or transmit bytes whenever it sends a low, ready signal.

The PropScope's 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.

6

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.

Comment [AL64]: Change any earlier "DAC output" references to "function generator output."

✓ 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 2-14 on page 44.

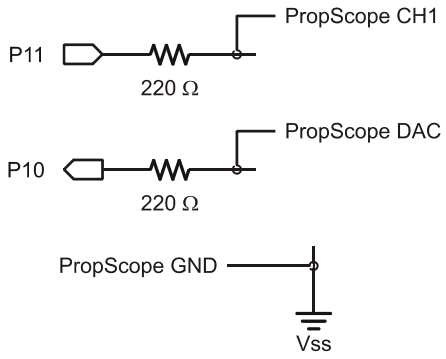


Figure 6-16
CH1 Probes P11 Serial Output, DAC Output Drives P10 Flow Control Input Line

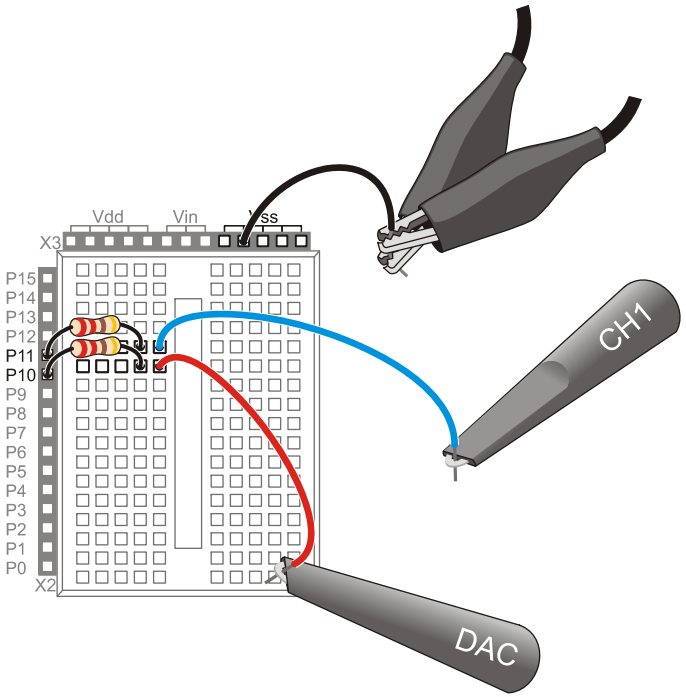


Figure 6-17
Wiring Diagram Example of Figure 6-16

6

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 the BASIC Stamp's 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.

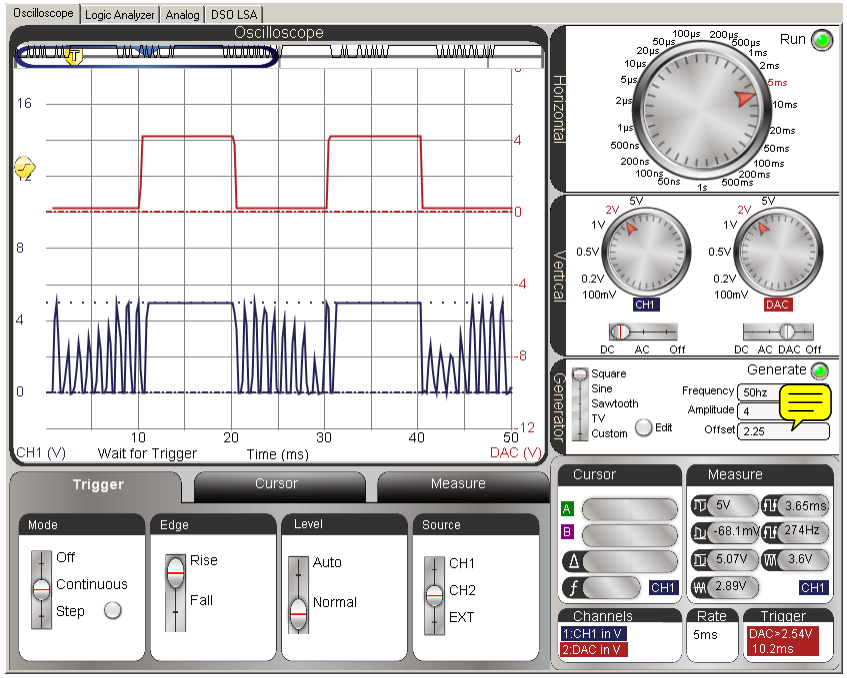
- ✓ Move the Plot Area Bar to the far left of the Plot Preview.
- ✓ Set the Horizontal Dial to 5 ms/div and the Vertical dials to 2 V/div.
- ✓ Set the Generator to Frequency = 50 Hz, ~~Offset = 2.25 V, Amplitude = 4.5 V,~~ then click the Generate button.
- ✓ Set the Vertical CH2 coupling switch to DAC. (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.

Comment [AL65]: Would it be better if the software Trigger Source option changed from CH2 to DAC when coupling switch is set to DAC?

i-box maybe?

- ✓ 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.
- ✓ Check your PropScope settings and display against Figure 6-18.

Figure 6-18: PropScope DAC Card Flow Control of Serial Byte Stream



In Figure 6-18, the lower CH1 trace's jagged spikes during the upper CH2 trace's function generator's low signals indicate 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 $10 \times t_{bit} \approx 10 \times 104\ \mu s = 1040\ \mu s = 1.04$ 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, $f = 1/T = 1 \div 4$ ms $= 1 \div 0.004$ 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.

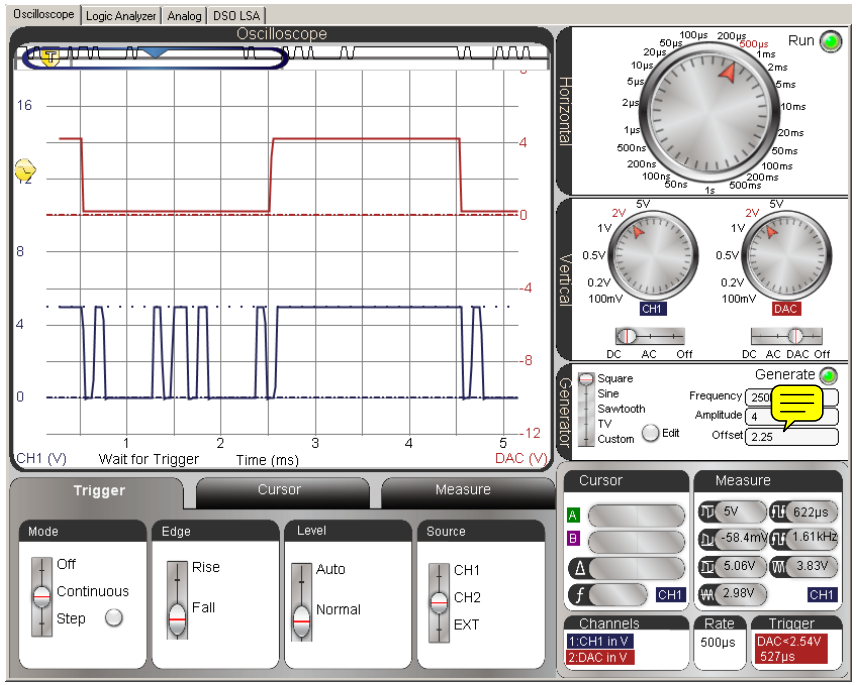
- ✓ Adjust the Horizontal dial to 500 µs/division.
- ✓ Change the Trigger Edge to Fall, and adjust the Trigger Time Control so that the DAC trace's negative edge lines up with the first time division line.

Comment [AL66]: Oops, go back through the rest of the book and clean this up.

Decide what to do about function generator vs. DAC trace.

Also, clean up DAC trace vs. CH2 trace.

Figure 6-19: Two "A" Bytes Per Cycle



Your Turn: Does Flow Control Stop at the Bit or the Byte?

In Figure 6-19, it's hard to tell for sure whether the flow control signal transitioning from low to high stops the serial transmit at the bit or the byte . 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.~~

Comment [AL67]: To-do: Double check this.

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 Manage Probes settings so that the Probe 1 Gain is 10 X. For the location of the probe's X1 X10 switch and the PropScope software's Manage Probes window, see Chapter 1, Activity #2.
- ✓ 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 to the CH1 input. Another option is to follow the connection instructions in the Set DC Voltages with the PropScope's DAC CARD section on page 43. Either way will work fine.

✓ Configure the PropScope's:

- Trigger tab to Mode = Continuous, Edge = Rise, Level = Auto, and Source = CH1.
- Horizontal = 200 µs/div, Vertical = 1V/div for both CH1 and DAC.
- Function generator to transmit a square wave with: Frequency = 1 kHz, Amplitude = 3 Vpp, and Offset = 0 V

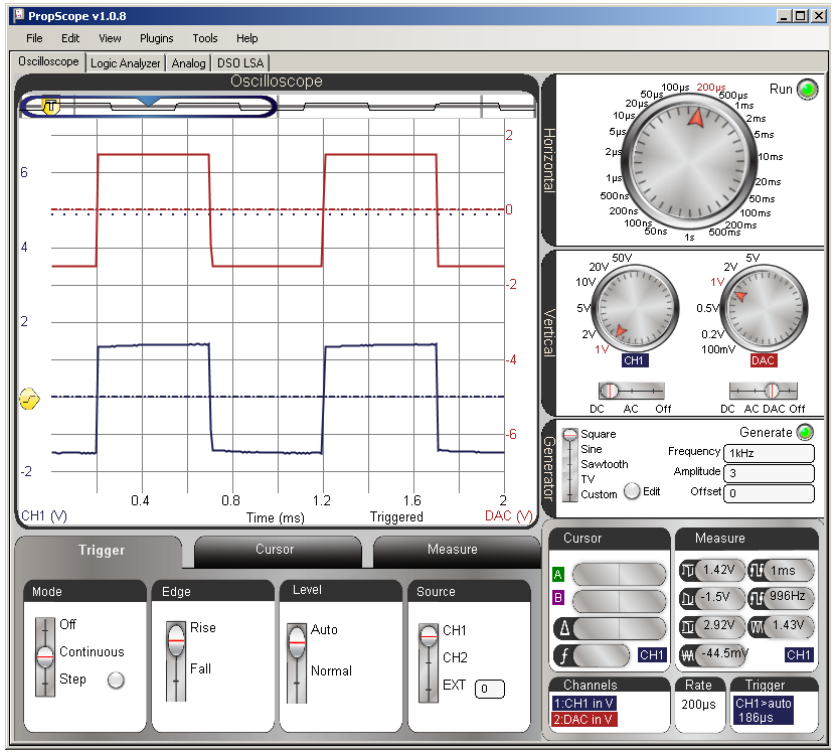


Figure 6-20 Measuring DAC CARD Function Generator to CH1

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.

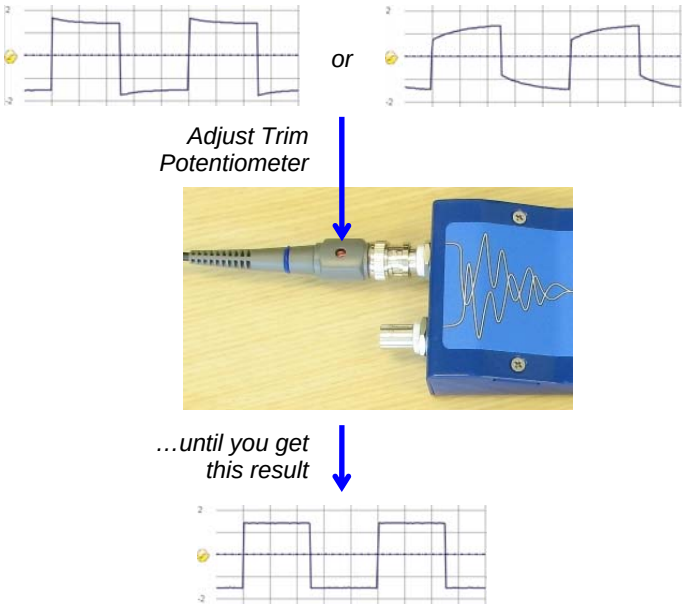


Figure 6-21
Adjust the trim pot in the CH1 BNC Connector

… to correct the shape of the square wave and the make the Square Wave Square

6

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 X1 X10 switch on the probe connected to the CH1 BNC connector to X10.

- ✓ Verify that the X1 X10 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 probe with the blue probe markers to the PropScope's CH1 BNC port.
- ✓ Reconnect the probe with the red probe markers 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~~ X1 X10 switches to X10.

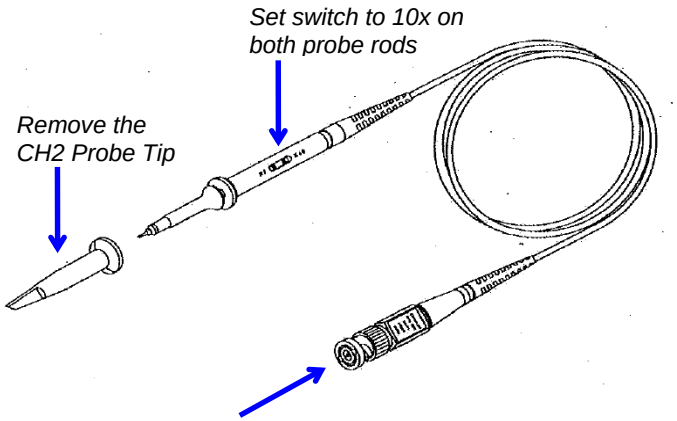


Figure 6-22
X1 X10 Switch on the Probe Rod, Probe Tip, and BNC Connector

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 and select Manage Probes. Set the probe gain to 10 for both probes.

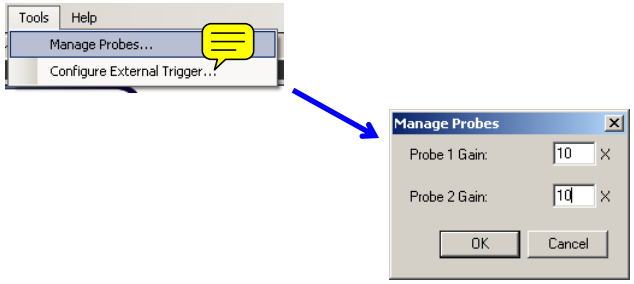


Figure 6-23
Configure Probe Gain to 10x

Serial Port Transmit Test Code

PC Serial Transmit.bs2 sends "A" to P11 and another "A" to the BASIC Stamp's 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.

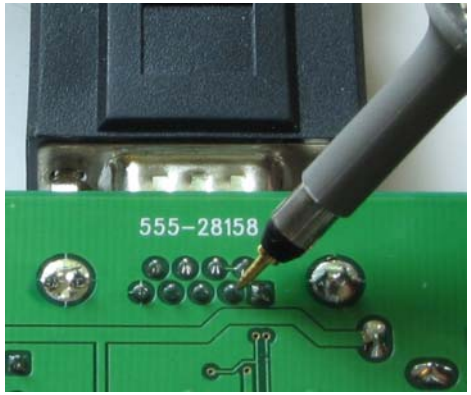
```
' PC Serial Transmit.bs2
' Sends a true serial byte to P11 as well as to P16 (SOUT).
' The byte to SOUT gets inverted by a circuit before it is transmitted.

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



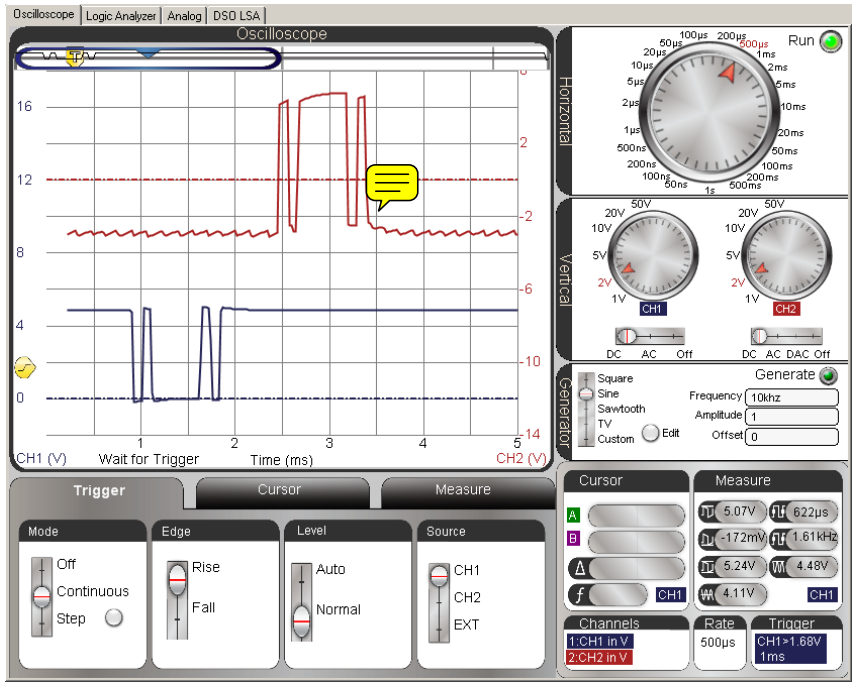
BASIC Stamp HomeWork Board
(Underneath the Serial Port Connecter)

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



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

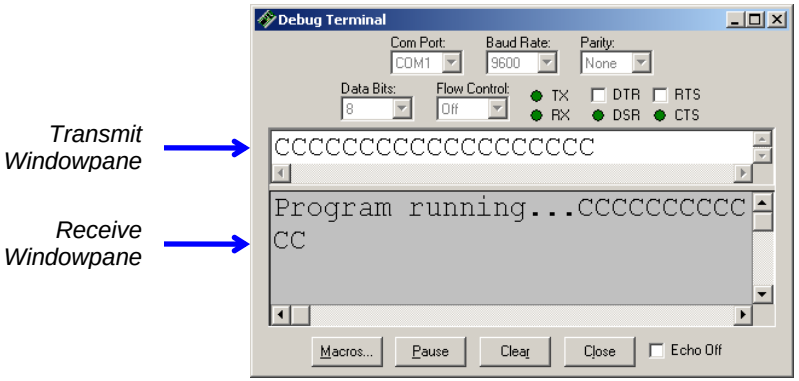
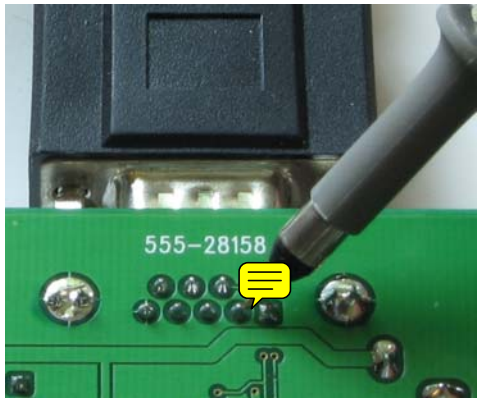


Figure 6-27 shows locations for testing the incoming signal from a PC on various boards. In both cases, the incoming signal test point is adjacent to the outgoing signal's test point.

- ✓ 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

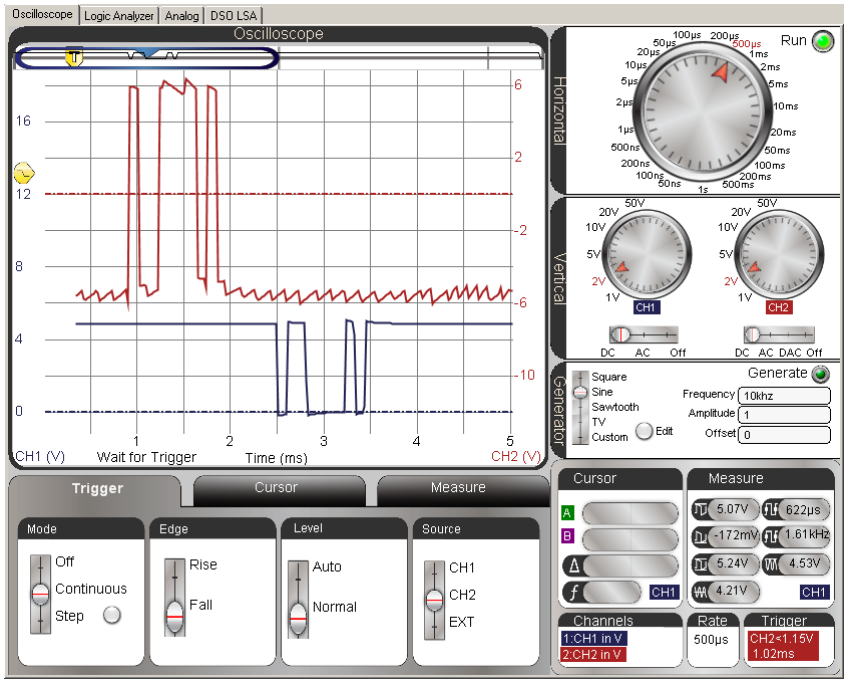


BASIC Stamp HomeWork Board
(Underneath the Serial Port Connecter)

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.

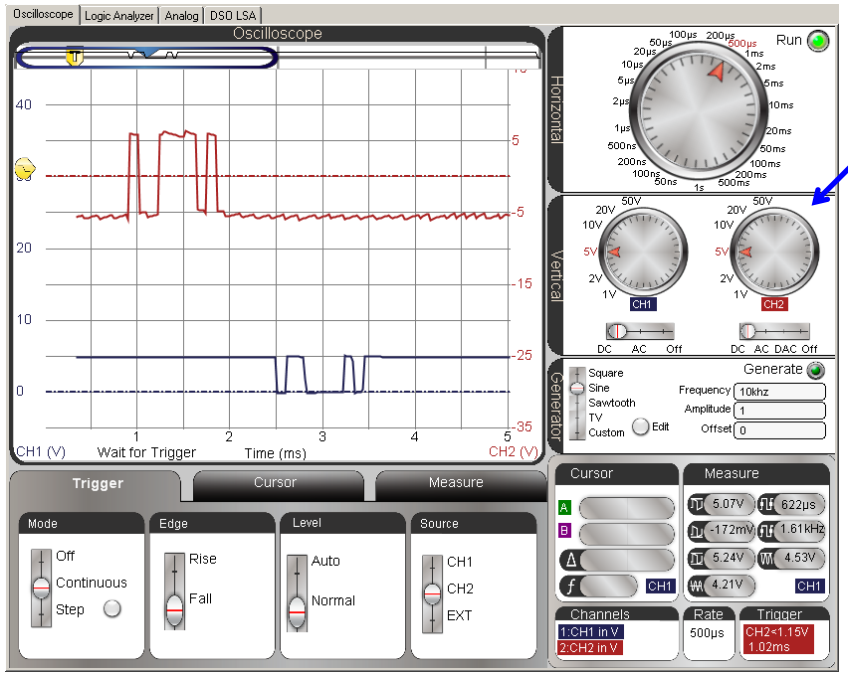
Figure 6-28: Inverted “C” at SIN Pin (upper CH2 trace), True “C” at P11 (lower CH1 trace)



Especially 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, and none of ~~this~~ chapter's projects ~~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.

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.

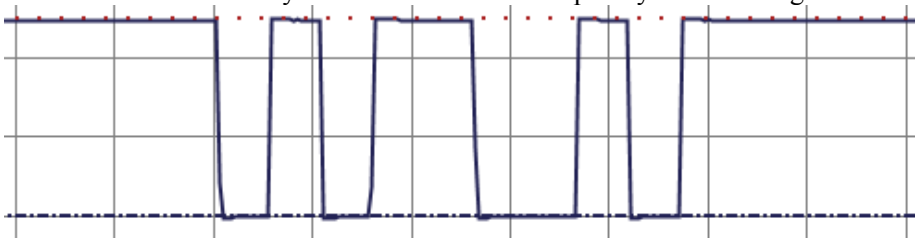
Questions

1. How do synchronous and asynchronous serial communication differ?
2. What does ASCII stand for?
3. What is a common way that ASCII codes are conveyed from one device to another?
4. If the BASIC Stamp sends the character 13 to the Debug Terminal, what happens?

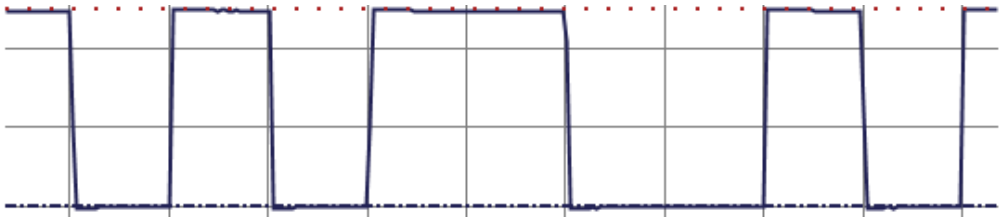
5. If the BASIC Stamp transmits the number 97, what value would you expect the Debug Terminal to display?
6. What order of bits do asynchronous serial signals transmit?
7. How many bit times are in an 8N1 message?
8. What does the N in 8N1 stand for?
9. What is the resting state of a true signal?
10. What does the value 84 in SEROUT 11, 84, [char] do?
11. How do you set a custom units per division in the PropScope software?
12. How does an inverted signal differ from a true signal?
13. What trigger edge is most appropriate for an inverted signal?
14. Does a device transmitting data send or listen for flow control signals? (Assume hardware flow control is used.)
15. How does signal amplitude through a 10x probe compare to amplitude through a 1x probe?
16. What two actions are required to change a probe connected to the PropScope from 1x to 10x?

Exercises

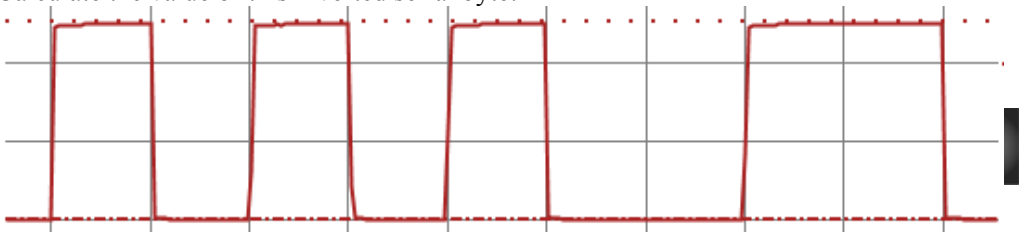
1. Modify the FOR…NEXT conditions in Printable ASCII Chart.bs2 to display from the letter E to the letter e.
2. Write a DEBUG command that sends the character "V" with a number.
3. Write a DEBUG command that sends a number equivalent to DEBUG CLS.
4. Calculate the bit period for 19.2 kbps.
5. Calculate the value of a true serial byte if bit 3 and bit 5 are high.
6. Calculate the value of an inverted serial byte if bit 3 and bit 5 are high.
7. Calculate the minimum time it should take to send four serial bytes at 19200 bps.
8. Determine if this serial byte is true or inverted. Explain your reasoning.



9. Calculate the value of this true serial byte:



10. Calculate the value of this inverted serial byte:



Projects

1. Use the logic analyzer to view "A" transmitted by P7 "B" transmitted by P6 "C" transmitted by P5, and "D" transmitted by P4
2. Do a search in the BASIC Stamp Editor's Help for parity and find the explanation for how it works. Test with the PropScope and verify that SEROUT command's parity bit functions according to the explanation.

Solutions

Q1. Asynchronous serial communication depends on a separate signal to synchronize the transfer of bits, asynchronous serial does not.
Q2. American Standard Code for Information Exchange.
Q3. With asynchronous serial communication.
Q4. The Debug Terminal prints a carriage return, placing the cursor one line down, and all the way to the left.
Q5. Lower-case a.
Q6. Least significant bit first. Bit-0, then bit-1, then bit-2, and so on ….
Q7. Ten including ~~an~~ a start bit, eight data bits, and one stop bit.
Q8. No parity.
Q9. High.
Q10. ts the asynchronous serial signaling to true, 9600 bps, 8N1.
Q11. Right-click the Oscilloscope screen and enter the units/division value into the (timescale, Value) cell.

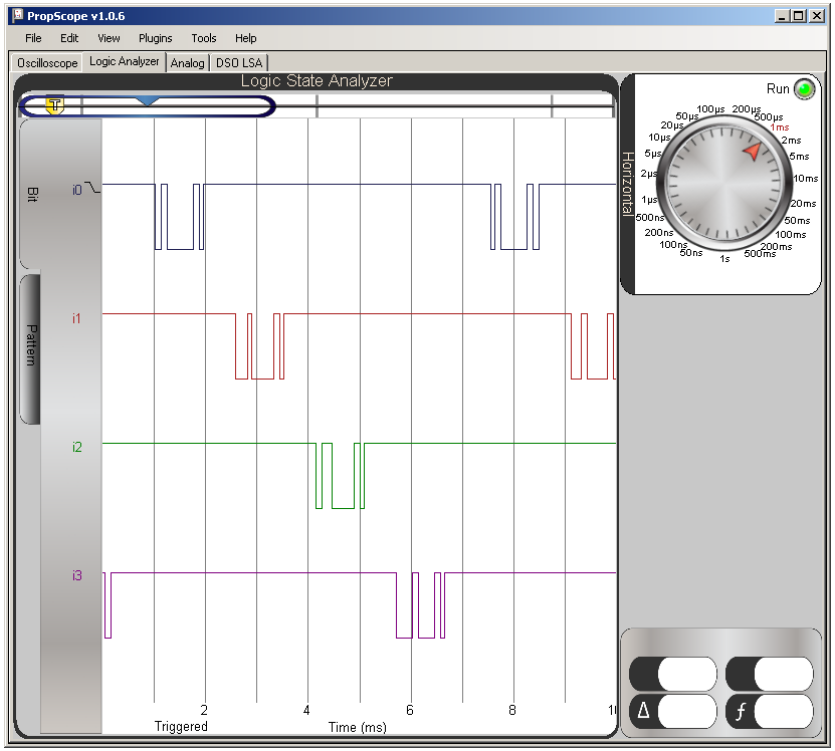
Q12. All voltage levels in an inverted signal are opposite of what they would be in a true signal.
Q13. Rise or positive edge trigger.
Q14. It listens for them.
Q15. The signal a 10x probe forwards to the oscilloscope's input circuit is 1/10 of what is with a 1x probe.
Q16. Adjust the X1 X10 switch on the probe, and update the software's probe settings.

E1. FOR char = "E" TO "e".
E2. DEBUG 86
E3. DEBUG 0
E4. t_{bit} = 1/baud rate = 1/19200 bits/s ≈ 52.083 µs/bit
E5. 8 + 32 = 40
E6. Binary 11010111 = 128 + 64 + 32 + 4 + 2 + 1 = 231
E7. 4 bytes × 10 bit periods per byte × 1 bit period = 40 × (1/19200) ≈ 2.083 ms. (Remember that a bit period is 1/baud rate.)
E8. True because it appears to have a high resting state and low start bit.
E9. 1 + 4 + 8 + 64 = 77
E10. 1 + 4 + 16 + 32 = 53

P1. Connections: Vss to GND, P7 to DAC CARD 0, P6 to DAC CARD 1, P5 to DAC CARD 2 and P4 to DAC CARD 3.
PBASIC Test Code:

```
' {$STAMP BS2}
' {$PBASIC 2.5}
PAUSE 1000
DEBUG "Program running..."
DO
  SEROUT 7, 84, ["A"]
  SEROUT 6, 84, ["B"]
  SEROUT 5, 84, ["C"]
  SEROUT 4, 84, ["D"]
LOOP
```

Verification



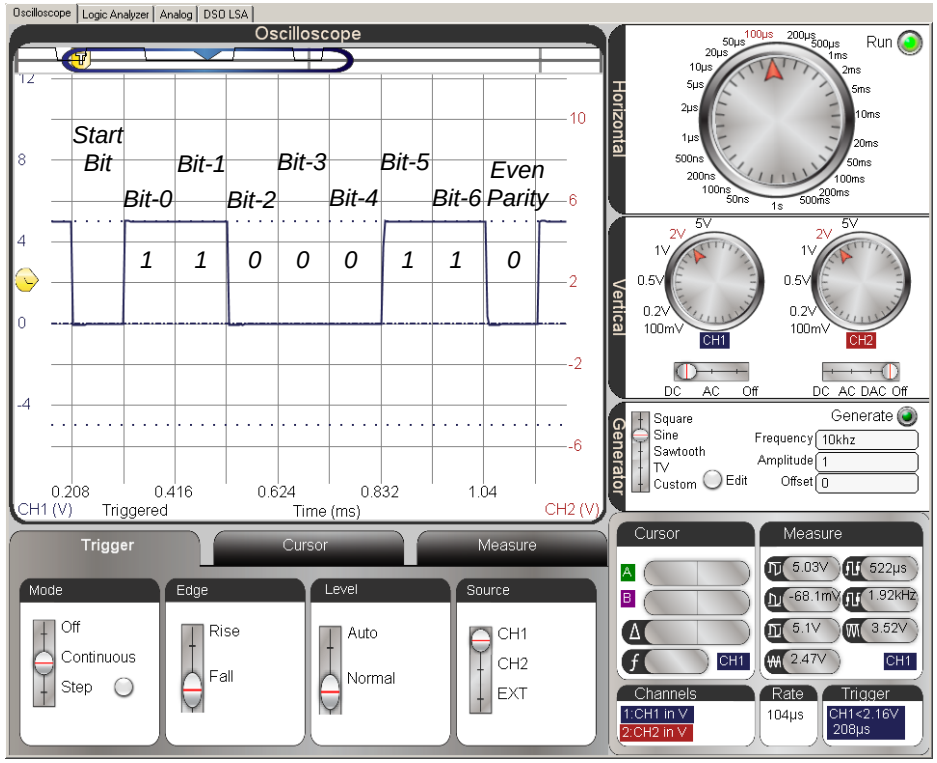
P2. According to the BASIC Stamp Editor Help's SEROUT command documentation, 7-bits with even parity uses bit-7 (the 8th data bit) as a parity bit. The SEROUT command chooses the value of this bit to make the total number of data bits that are 1 an even number. So if the value has four 1s, the parity bit will be 0 to keep the number of 1 bits even. If there are five data bits, then the parity bit will be 1 to round out the number of 1 bits to an even 6. The SEROUT Command Documentation's Common Baud Rates and Corresponding Baudmodes table for the BASIC Stamp 2 lists 8276 as the Baudmode argument for a 9600 bps true signal. A convenient approach would be to send a couple of 7-bit binary numbers, one with an even number of data bits, and the other with an odd number, and check to see if bit-7 is set and cleared accordingly.

PBASIC Test Code

```
' {$STAMP BS2}                        ' Target module = BASIC Stamp 2
' {$PBASIC 2.5}                       ' Language = PBASIC 2.5
PAUSE 1000
DEBUG "Program running..."
DO
```

```
  SEROUT 11, 8276, [%01100011]    ' Even number of 1 bits, parity
                                  '  should be 0
'  SEROUT 11, 8276, [%01100111]    ' Odd number of 1 bits, parity
                                  ' should be 1

  PAUSE 1000
LOOP
```

SEROUT 11, 8276, [%01100011] - parity bit is 0 to keep 1 count at even four.



SEROUT 11, 8276, [%01100111] – parity bit is 1 to increase five 1 bits to even six