Building an RC Time Circuit for the BASIC Stamp

Figure 5-11 shows a schematic and wiring diagram for the fast RC-time circuit. This is the circuit that you will use to monitor the position of the potentiometer's knob with the help of the BASIC Stamp and a PBASIC program.

✓ Build the circuit shown in Figure 5-11.

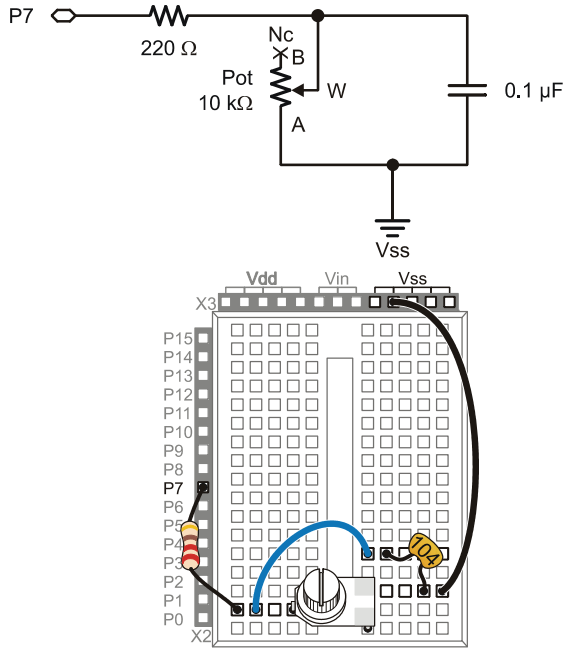


Figure 5-11
Schematic and wiring diagram for BASIC Stamp RCTIME Circuit with Potentiometer

Programming RC-Time Measurements

The example program in Activity #2 measured RC decay time by checking whether **`IN7`** = 0 every 100 ms, and it kept track of how many times it had to check. When **`IN7`** changed from 1 to 0, it indicated that the capacitor's voltage decayed to 1.4 V. The result when the program was done polling was that the **`timeCounter`** variable stored the number of tenths of a second it took for the capacitor's voltage to decay to 1.4 V.

This next example program uses a PBASIC command called **`RCTIME`** that makes the BASIC Stamp measure RC decay in terms of 2 µs units. So, instead of tenths of a

second, the result **RCTIME 7, 1, time** stores in the **time** variable is the number of two-millionths of a second units that it takes for the capacitor's voltage to decay below 1.4 V. Since the **RCTIME** command has such fine measurement units, you can reduce the capacitor size from 3300 µF to 0.1 or even 0.01 µF, and still get time measurements that indicate the resistor's value. Since the resistance between the potentiometer's A and W terminals changes as you turn the knob, the **RCTIME** measurement will give you a time measurement, which corresponds to the position of the potentiometer's knob.

Example Program: ReadPotWithRcTime.bs2

- ✓ Enter and run ReadPotWithRcTime.bs2
- ✓ Try rotating the potentiometer's knob while monitoring the value of the **time** variable using the Debug Terminal.

> **Remember to apply a little downward pressure** to keep the potentiometer seated on the breadboard as you twist its knob. If your servo starts twitching back and forth unexpectedly instead of holding its position, an un-seated pot may be the culprit.

```
' What's a Microcontroller - ReadPotWithRcTime.bs2
' Read potentiometer in RC-time circuit using RCTIME command.

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

time VAR Word

PAUSE 1000

DO

  HIGH 7
  PAUSE 100
  RCTIME 7, 1, time
  DEBUG HOME, "time = ", DEC5 time

LOOP
```

Your Turn – Changing Time by Changing the Capacitor

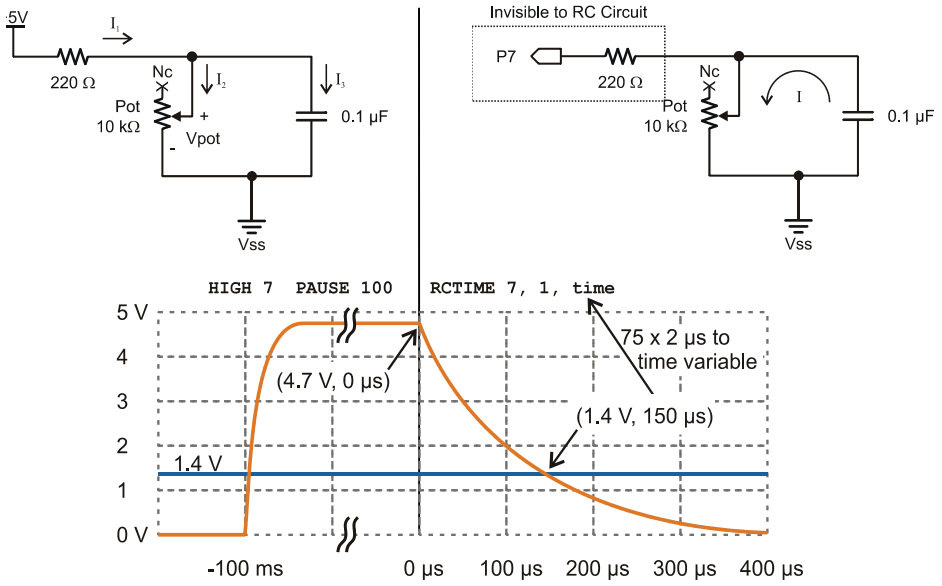
- ✓ Replace the 0.1 µF capacitor with a 0.01 µF capacitor.
- ✓ Try the same positions on the potentiometer that you did in the main activity and compare the value displayed in the Debug Terminal with the values obtained for the 0.1 µF capacitor. Are the **RCTIME** measurements about one tenth the value for a given potentiometer position?

- ✓ Go back to the 0.1 µF capacitor.
- ✓ With the 0.1 µF capacitor back in the circuit and the 0.01 µF capacitor removed, turn the pot's knob to its limit in both directions and make notes of the highest and lowest values for the next activity. Highest:_____ Lowest:_____

How ReadPotWithRcTime.bs2 Works

Figure 5-12 shows how the ReadPotWithRcTime.bs2's **`HIGH`**, **`PAUSE`** and **`RCTIME`** commands interact with the circuit in Figure 5-11.

Figure 5-12: Voltage at P7 through HIGH, PAUSE, and RCTIME



On the left, the **`HIGH 7`** command causes the BASIC Stamp to internally connect its I/O pin P7 to the 5 V supply (Vdd). Current from the supply flows through the potentiometer's resistor and also charges the capacitor. The closer the capacitor gets to its final charge (almost 5 V), the less current flows into it. The **`PAUSE 100`** command is primarily to make the Debug Terminal display update at about 10 times per second; **`PAUSE 1`** is usually sufficient to charge the capacitor. On the right, the **`RCTIME 7, 1, time`** command changes the I/O pin direction from output to input and starts counting time in 2 µs increments. As an input the I/O pin no longer supplies the circuit with 5 V.

In fact, as an input, it's pretty much invisible to the RC circuit. So, the capacitor starts losing its charge through the potentiometer. As the capacitor loses its charge, its voltage decays. The **RCTIME** command keeps counting time until P7 senses a low signal, meaning the voltage across the capacitor has decayed to 1.4 V, at which point it stores its measurement in the **time** variable.

Figure 5-12 also shows a graph of the voltage across the capacitor during the **HIGH**, **PAUSE**, and **RCTIME** commands. In response to the **HIGH 7** command, which connects the circuit to 5 V, the capacitor quickly charges. Then, it remains level at its final voltage during most of the **PAUSE 100** command. When the program gets to the **RCTIME 7, 1, time** command, it changes the I/O pin direction to input, so the capacitor starts to discharge through the potentiometer. As the capacitor discharges, the voltage at P7 decays. When the voltage decays to 1.4 V (at the 150 µs mark in this example), the **RCTIME** command stops counting time and stores the measurement result in the **time** variable. Since the **RCTIME** command counts time in 2 µs units, the result for 150 µs that gets stored in the **time** variable is 75.

I/O Pin Logic Threshold: 1.4 V is a BASIC Stamp 2 I/O pin's logic threshold. When the I/O pin is set to input, it stores a 1 in its input register if the voltage applied is above 1.4 V or a 0 if the input voltage is 1.4 V or below. The first pushbutton example back in Chapter 3, Activity #2 applied either 5 V or 0 V to P3. Since 5 V is above 1.4 V, **IN3** stored a 1, and since 0 V is below 1.4 V, **IN3** stored a 0.

RCTIME *State* Argument: In ReadPotWithRcTime.bs2, the voltage across the capacitor decays from almost 5 V, and when it gets to 1.4 V, the value in the **IN7** register changes from 1 to 0. At that point, the **RCTIME** command stores its measurement in its ***Duration***, which is the **time** variable in the example program. The **RCTIME** command's ***State*** argument is 1 in **RCTIME 7, 1, time**, which tells the **RCTIME** command that the **IN7** register will store a 1 when the measurement starts. The **RCTIME** command measures how long it takes for the **IN7** register to change to the opposite state, which happens when the voltage decays below the I/O pin's 1.4 V logic threshold.

For more information: Look up the **RCTIME** command in either the BASIC Stamp Manual or the BASIC Stamp Editor's Help.

Figure 5-13 shows how the decay time changes with the potentiometer's resistance for the circuit in Figure 5-11. Each position of the potentiometer's knob sets it at a certain resistance. Turn it further one direction, and the resistance increases, and in the other direction, the resistance decreases. When the resistance is larger, the decay takes a longer time, and the **RCTIME** command stores a larger value in the **time** variable. When the resistance is smaller, the decay takes a shorter time, and the **RCTIME** command stores a

smaller value in the **`time`** variable. The **`DEBUG`** command in ReadPotWithRcTime.bs2 displays this time measurement in the Debug Terminal, and since the decay time changes with the potentiometer's resistance, which in turn changes with the potentiometer knob's position, the number in the Debug Terminal indicates the knob's position.

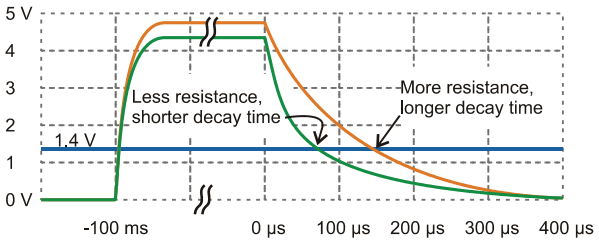


Figure 5-13
How Potentiometer Resistance Affects Decay Time

Why does the capacitor charge to a lower voltage when the potentiometer has less resistance?

Take a look at the schematic in the upper-left corner of Figure 5-12 on page 153. Without the 220 Ω resistor, the I/O pin would be able to charge the capacitor to 5 V, but the 220 Ω resistor is necessary to prevent possible I/O pin damage from a current inrush when it starts charging the capacitor. It also prevents the potentiometer from drawing too much current if it is turned to 0 Ω while the I/O pin sends its 5 V high signal.

With 5 V applied across the 220 Ω resistor in series with the potentiometer, the voltage between them has to be some fraction of 5 V. When two resistors conducting current are placed in series, which results in an intermediate voltage, the circuit is called a *voltage divider*. So the 220 Ω resistor and potentiometer form a voltage divider circuit, and for any given potentiometer resistance (Rpot), you can use this equation to calculate the voltage across the potentiometer (Vpot):

Vpot = 5 V × Rpot ÷ (Rpot + 220 Ω)

The value of Vpot sets the ceiling on the capacitor's voltage. In other words, whatever the voltage across the potentiometer would be if the capacitor wasn't connected, that's the voltage the capacitor can charge to, and no higher. For most of the potentiometer knob's range, the resistance values are in the kΩ, and when you calculate Vpot for kΩ Rpot values, the results are pretty close to 5 V. The 220 Ω resistor doesn't prevent Vpot from charging above 1.4 V until the potentiometer's value is down at 85.6 Ω, which is less than 1% of the potentiometer's range of motion. This 1% would have resulted in the lowest measurements anyhow, so it's difficult to tell that measurements of 1 in this range are anything out of the ordinary. Even with the additional 220 Ω resistors built into BASIC Stamp HomeWork board I/O pin connections, only the lowest 1.7% of the potentiometer's range is affected, so it's still virtually unnoticeable.

So the 220 Ω resistor protects the I/O pin, with minimal impact on the RC decay measurement's ability to tell you where you positioned the potentiometer's knob.