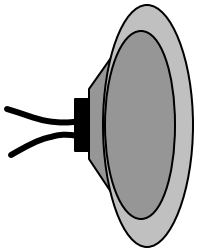
Analog Inputs and Outputs

This week, we are again using the BASIC Stamp. This time, we will use some specialized features of the Stamp to monitor and control analog inputs and outputs. Such tasks are especially hard for our usual TTL circuits, but are not so bad with the Stamp

Speakers

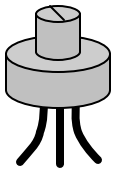


A speaker is an output device that is fundamentally similar to a resistor in that it restricts current running through it. The resistance of a speaker is small (usually less than 10Ω). The speaker also has a small membrane that vibrates at the same rate as the voltage difference across its two legs. This membrane pushes on the nearby air, which generates sound. Higher voltages generate louder sounds. People can hear sounds ranging from about 40 Hz to around 20, 000 Hz, depending on the person.

We will generally connect one leg of the speaker to ground, and the other to a pin on the Stamp. PBASIC has a command called ***FREQOUT pin, duration, frequency*** which can be used to send oscillating voltage to a speaker. For example, the following command generates a sound of frequency 200 Hz for one and a half seconds on pin 4:

```
FREQOUT 4, 1500, 200
```

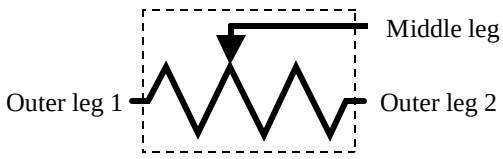
Potentiometers



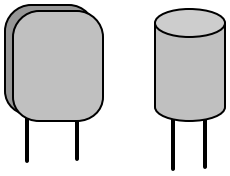
We already know about resistors. A potentiometer, or "pot" for short, is a resistor having a variable value. Pots commonly appear as devices with a knob to be turned. Most pots have 3 output wires. The full resistance sits between the outer two terminals, and never changes (it is R_{max}). As the knob is turned, the resistance between the middle terminal and one of the outer wires changes from 0 to R_{max}. At the same time, the resistance between the middle wire and the other outer wire changes from R_{max} to 0. For example, consider a pot having a total resistance of 1000 Ω. As the knob is turned, here are some possible outcomes:

Terminal 1 to 3:	Terminal 1 to 2	Terminal 2 to 3:
1000 Ω	0 Ω	1000 Ω
1000 Ω	200 Ω	800 Ω
1000 Ω	500 Ω	500 Ω
1000 Ω	800 Ω	200 Ω
1000 Ω	1000 Ω	0 Ω

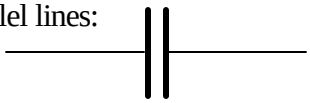
The symbol for a pot is a resistor with an extra leg attached to an arrow:



Capacitors



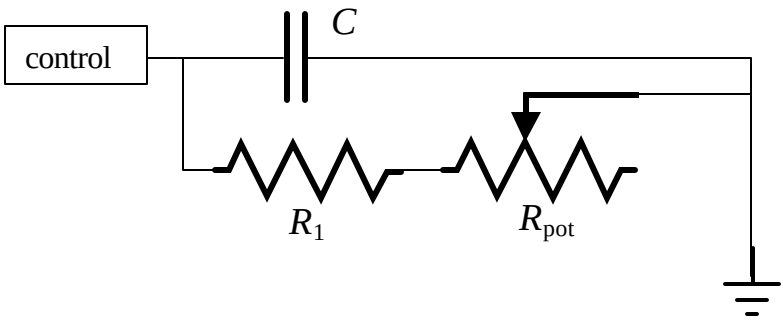
A capacitor is a device that stores charge. It has two legs, like a resistor. As current flows into it, charge accumulates like water in a swimming pool. However, the more it fills up, new charge is accepted into it more slowly. Eventually, the capacitor will become completely filled up, unless something is done to empty it. Once it is full, no new charge can enter, and therefore, no current can flow along any wires going into it. The units of capacitance are Farads, named after Michael Faraday. A one Farad capacitor is huge; you will probably not see one in your life. The symbol for a capacitor is two parallel lines:



Time Constants

When resistors are used simultaneously with capacitors, the resistor can slow down the amount of current entering the capacitor. Interestingly, the amount of time to fill the capacitor to any desired amount (for example, to 68% full) can be related to the product of the resistance and the capacitance. Part of what this means is that Ohms times Farads equals seconds, which you may not have guessed! The product of R times C is called the "time constant" for a circuit that has both. It is not only related to the amount of time needed to fill the capacitor, it is related to how much time is required to drain the charge out of the capacitor.

Consider the following circuit:



Notice that one of the outer legs of the pot is not used. Suppose that the control starts off high (+5V) for some time, and is then disconnected. Note that it is *not* set to zero, it is merely disconnected.

While the control was high, a lot of charge was stored on the left side of the capacitor. The +5V kept most of it there; some current was also running from the +5V control, through both resistors, and into the ground. The first resistor R_1 is there for safety: if the knob of the pot is ever turned all the way to one side, then the resistance of the pot would be zero. Without R_1, this would result in a short circuit from ground to the control.

Once the control is released, the charge on the capacitor has no reason to stay, and it will eventually flow through the two resistors until it reaches ground. The amount of time needed to drain it depends on how much charge there is in the first place (i.e., on the size of the capacitor), and on how fast it can get out (i.e., on the total resistance). Although it is not exact, the amount of time needed to drain the capacitor, (t, in seconds), can be computed using:

$$t \approx (R_1 + R_{pot})C$$

If the knob is turned all the way to one side so that R_1 is zero, then $t_{min} \approx R_1 C$. If the knob is turned all the way to the other side, then $t_{max} \approx (R_1 + R_{max})C$.

PBASIC Command: RCTIME

If you attach the control in the above circuit to one of the pins on the Stamp, then PBASIC can perform the following tasks:

a. Use the "control" pin in the above circuit as an output, and set it high to charge the capacitor.
b. Wait a moment to make sure the capacitor is full. The amount of time needed to ensure that it is full is about $5 \times t_{max}$.
c. Change the pin to be an input. This releases the voltage on that pin, effectively disconnecting it from the voltage source.
d. Repeatedly monitor the status of the input. At first, it will be high. But as charge leaves the capacitor, the voltage will shrink. Eventually, it will get low enough that the Stamp will see the voltage as a low signal (about 1.4V, actually).
   During this process, the Stamp will count how many times it has to check the voltage before it thinks the input has become low. Although it is not exact, the Stamp will check the voltage about 620,000 times per second while it waiting, and when the process is done, it will tell you how many total times it checked the voltage.

A program to do all of that is shown here:

```
' {$STAMP BS2}
' {$PBASIC 2.5}
checks VAR Word
HIGH 0
PAUSE 100
RCTIME 0, 1, checks          ' does both steps c and d from above list
DEBUG DEC5 checks, CR
END
```

This program will output a value called "checks", which is the number of times that the program looked at the input (pin 0, in this case) while the capacitor was draining. Two new statements in this program are:

DEBUG DEC5 *varname* → Formats *varname* into ##### format before printing. So, if checks = 427, it prints 00427.

RCTIME *pin*,1, *varname* → changes *pin* to an input, keeping track (in *varname*) of how many voltage checks it does before it senses that this pin is low. The "1" means that we are watching the capacitor drain; 0 can instead be used (to watch it fill).

Assuming that you know both R_1 and C, the resulting value can be used to tell you how far the knob has been turned, which is virtually impossible with normal digital circuits.

Instructions

Part A

1. Choose a pot. Then, choose an appropriate resistor and capacitor to enable the **RCTIME** statement to perform a reasonable number of checks when the pot is at each end of its range. Here is some advice:
    a. Determine R_{max} for the pot, then choose R_1 so that it is no smaller than 50Ω, and no larger than about one third of R_{max}.
    b. Choose a capacitor of about $C \approx \frac{1\text{sec}}{6000 \cdot R_1}$.
    c. These are only guidelines; it will almost certainly be true that you can't find the perfect numbers suggested here, since we only have a few sizes of resistor and capacitor. Do the best you can!

:

2. Write and test a program on the Stamp to determine the number of counts your circuit generates when your pot is fully turned in each direction.
3. Using the information you learned in parts 1 and 3, write and test another program on the Stamp that continually monitors the pot, and calculates how far the pot is turned, as a percentage. Output that percentage on a pair of 7 segment displays. The legal output range is 0% to 99%. Values greater than 99% should indicate 99%, and values less than 1% should indicate 0%.
4. Add a stop switch.

Part B

1. Connect a speaker between ground and the clock output of the PAD-234. Experiment with a range of frequencies and wave shapes.
2. Write a program to monitor the same potentiometer that you used in part A, and generate an integer *N* which ranges from 0 to 7 rather than from 0 to 99. Do not display this number.
3. Compute a new integer *f*, depending on *N*, using this chart:
4. Use the **FREQOUT** command (with a duration of 200 ms and a frequency *f*) to control a speaker.
5. Add a stop switch.

N	**f**
0	440
1	494
2	554
3	587
4	659
5	740
6	831
7	880