

## Chapter #2: Lights On – Lights Off

---

### INDICATOR LIGHTS

Indicator lights are so common that most people tend not to give them much thought. Figure 2-1 shows three indicator lights on a laser printer. Depending on which light is on, the person using the printer knows if it is running properly or needs attention. Here are just a few examples of devices with indicator lights: car stereos, televisions, VCRs, disk drives, printers, and alarm system control panels.



**Figure 2-1**  
Indicator  
Lights

Turning an indicator light on and off is a simple matter of connecting and disconnecting it from a power source. In some cases, the indicator light is connected directly to the battery or power supply, like the power indicator light on the Board of Education. Other indicator lights are switched on and off by a microcontroller inside the device. These are usually status indicator lights that tell you what the device is up to.

### MAKING A LIGHT EMITTING DIODE (LED) EMIT LIGHT

Most of the indicator lights you see on devices are called light emitting diodes. You will often see a light emitting diode referred to in books and circuit diagrams by the letters LED. The name is usually pronounced as three letters: “L-E-D”. You can build an LED circuit and connect power to it, and the LED emits light. You can disconnect the power from an LED circuit, and the LED stops emitting light.

An LED circuit can be connected to the BASIC Stamp, and the BASIC Stamp can be programmed to connect and disconnect the LED circuit's power. This is much easier than manually changing the circuit's wiring or connecting and disconnecting the battery. The BASIC Stamp can also be programmed to do the following:

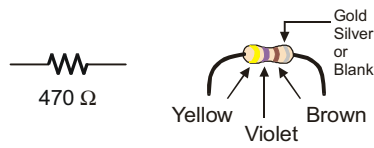
- Turn an LED circuit on and off at different rates
- Turn an LED circuit on and off a certain number of times
- Control more than one LED circuit
- Control the color of a bi-color (two color) LED circuit

### **ACTIVITY #1: BUILDING AND TESTING THE LED CIRCUIT**

It's important to test components individually before building them into a larger system. This activity focuses on building and testing two different LED circuits. The first circuit is the one that makes the LED emit light. The second circuit is the one that makes it not emit light. In the activity that comes after this one, you will build the LED circuit into a larger system by connecting it to the BASIC Stamp. You will then write programs that make the BASIC Stamp cause the LED to emit light, then not emit light. By first testing each LED circuit to make sure it works, you can be more confident that it will work when you connect it to a BASIC Stamp.

#### **Introducing the Resistor**

A resistor is a component that 'resists' the flow of electricity. This flow of electricity is called current. Each resistor has a value that tells how strongly it resists current flow. This resistance value is called the ohm, and the sign for the ohm is the Greek letter omega:  $\Omega$ . The resistor you will be working with in this activity is the 470  $\Omega$  resistor shown in Figure 2-2. The resistor has two wires (called leads and pronounced "leeds"), one coming out of each end. There is a ceramic case between the two leads, and it's the part that resists current flow. Most circuit diagrams that show resistors use the jagged line symbol on the left to tell the person building the circuit that he or she must use a 470  $\Omega$  resistor. This is called a schematic symbol. The drawing on the right is a part drawing used in some beginner level Stamps in Class texts to help you identify the resistor in your kit.



**Figure 2-2**  
470 Ω Resistor Part Drawing

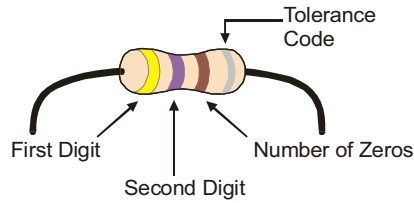
*Schematic symbol (left) and Part Drawing (right)*

Resistors like the ones we are using in this activity have colored stripes that tell you what their resistance values are. There is a different color combination for each resistance value. For example, the color code for the 470 Ω resistor is yellow-violet-brown.

There may be a fourth stripe that indicates the resistor’s tolerance. Tolerance is measured in percent, and it tells how far off the part’s true resistance might be from the labeled resistance. The fourth stripe could be gold (5%), silver (10%) or no stripe (20%). For the activities in this book, a resistor’s tolerance does not matter, but its value does.

Each color bar that tells you the resistor’s value corresponds to a digit, and these colors/digits are listed in Table 2-1. Figure 2-3 shows how to use each color bar with the table to determine the value of a resistor.

Digit	Color
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White



**Figure 2-3**  
Resistor Color Codes

Here is an example that shows how Table 2-1 and Figure 2-3 can be used to figure out a resistor value by proving that yellow-violet-brown is really 470 Ω:

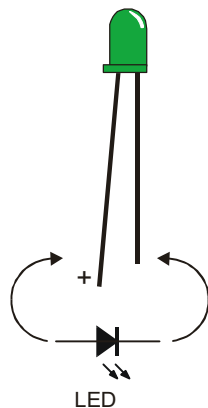
- The first stripe is yellow, which means the leftmost digit is a 4.
- The second stripe is violet, which means the next digit is a 7.
- The third stripe is brown. Since brown is 1, it means add one zero to the right of the first two digits.

Yellow-Violet-Brown = 4-7-0.

### Introducing the LED

A diode is a one-way current valve, and a light emitting diode (LED) emits light when current passes through it. Unlike the color codes on a resistor, the color of the LED usually just tells you what color it will glow when current passes through it. The important markings on an LED are contained in its shape. Since an LED is a one-way current valve, you have to make sure to connect it the right way, or it won't work as intended.

Figure 2-4 shows an LED's schematic symbol and part drawing. An LED has two terminals. One is called the anode, and the other is called the cathode. In this activity, you will have to build the LED into a circuit, paying attention to make sure the leads connected to the anode and cathode are connected to the circuit properly. On the part drawing, the anode lead is labeled with the plus-sign (+). On the schematic symbol, the anode is the wide part of the triangle. In the part drawing, the cathode lead is the unlabeled pin, and on the schematic symbol, the cathode is the line across the point of the triangle.



**Figure 2-4**  
LED Part Drawing  
and Schematic  
Symbol

*Part Drawing (above)  
and schematic symbol  
(below).*

*The LED's part  
drawings in later  
pictures will have a +  
next to the anode leg.*

When you start building your circuit, make sure to check it against the schematic symbol and part drawing. For the part drawing, note that the LED's leads are different lengths. The longer lead is connected to the LED's anode, and the shorter lead is connected to its cathode. Also, if you look closely at the LED's plastic case, it's mostly round, but there is a small flat spot right near the shorter lead that tells you it's the cathode. This really comes in handy if the leads have been clipped to the same length.

### **LED Test Circuit Parts**

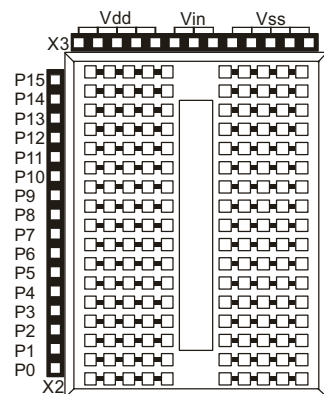
- (1) LED – Green
- (1) Resistor – 470  $\Omega$  (yellow-violet-brown)



**Identifying the parts:** In addition to the part drawings in Figure 2-2 and Figure 2-4, you can use the photo on the last page of the book to help identify the parts in the kit needed for this and all other activities. For more information on the parts in this photo, see Appendix B: Equipment and Parts Lists.

### **Building the LED Test Circuit**

You will build a circuit by plugging the LED and resistor leads into small holes called sockets on the prototyping area shown in Figure 2-5. This prototyping area has black sockets along the top and along the left. The black sockets along the top have labels above them: Vdd, Vin, and Vss. These are called the power terminals, and they will be used to supply your circuits with electricity. The black sockets on the left have labels like P0, P1, up through P15. These are sockets that you can use to connect your circuit to the BASIC Stamp module's input/output pins. The white board with lots of holes in it is called a solderless breadboard. You will use this breadboard to connect components to each other and build circuits.



**Figure 2-5**  
Prototyping Area

*Power terminals (black sockets along top), I/O pin access (black sockets along the side), and solderless breadboard (white sockets)*



**Input/output pins** are usually called I/O pins, and after connecting your circuit to one or more of these I/O pins, you can program your BASIC Stamp to monitor the circuit (input) or send on or off signals to the circuit (output). You will try this in the next activity.

Figure 2-6 shows a circuit schematic, and a picture of how that circuit will look when it is built on the prototyping area. The breadboard is separated into rows of five sockets. Each row can connect up to five leads, or wires, to each other. For this circuit, the resistor and the LED are connected because each one has a lead plugged into the same 5-socket row. Note that one lead of the resistor is plugged into Vdd so the circuit can draw power. The other resistor lead connects to the LED's anode lead. The LED's cathode lead is connected to Vss, or ground, completing the circuit.

You are now ready to build the circuit shown in Figure 2-6 (below) by plugging the LED and resistor leads into sockets on the prototyping area. Follow these steps:

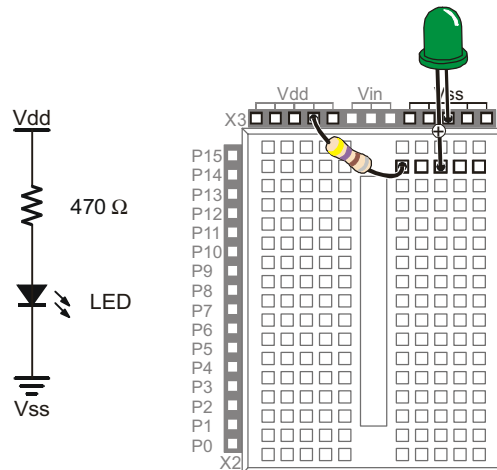
- √ Disconnect power from your Board of Education or HomeWork Board.
- √ Use Figure 2-4 to decide which lead is connected to the LED's cathode. Look for the shorter lead and the flat spot on the plastic part of the LED.
- √ Plug the LED's cathode into one of the black sockets labeled Vss on the prototyping area.
- √ Plug the LED's anode (the other, longer lead) into the socket shown on the breadboard portion of the prototyping area.
- √ Plug one of the resistor's leads into the same breadboard row as the LED's anode. This will connect those two leads together.

- ✓ Plug the resistor's other lead into one of the sockets labeled Vdd.



**Direction does matter for the LED, but not for the resistor.** If you plug the LED in backward, the LED will not emit light when you connect power. The resistor just resists the flow of current. There is no backwards or forwards for a resistor.

- ✓ Reconnect power to your Board of Education or HomeWork Board.
- ✓ Check to make sure your green LED is emitting light. It should glow green.



**Figure 2-6**  
LED On

*Schematic (left) and  
Wiring Diagram (right)*

If your green LED does not emit light when you connect power to the board:

- ✓ Some LEDs are brightest when viewed from above. Try looking straight down onto the dome part of the LED's plastic case from above.
- ✓ If the room is bright, try turning off some of the lights, or use your hands to cast a shadow on the LED.

If you still do not see any green glow, try these steps:

- ✓ Double check to make sure your cathode and anode are connected properly. If not, simply remove the LED, give it a half-turn, and plug it back in. It will not hurt the LED if you plug it in backwards, it just doesn't emit light. When you have it plugged in the right direction, it should emit light.

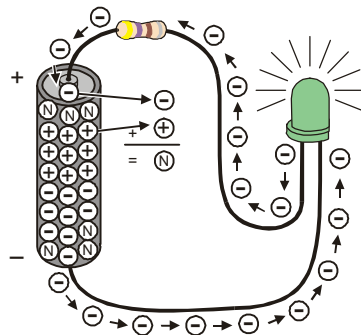
- √ Double check to make sure you built your circuit exactly as shown in Figure 2-6.
- √ If you are using a What's a Microcontroller kit that somebody used before you, the LED may be damaged, so try a different one.
- √ If you are in a lab class, check with your instructor.



**Still stuck?** If you don't have an instructor or friend who can help, you can always check with the Stamps in Class discussion group. The first pages of this book has Internet Access information on where to find the Stamps in Class discussion group. If the group is unable to help you solve the problem, you can contact the Parallax Technical Support department by following the Support link at [www.parallax.com](http://www.parallax.com).

### How the LED Test Circuit Works

The Vdd and Vss terminals supply electrical pressure in the same way that a battery would. The Vdd sockets are like the battery's positive terminal, and the Vss sockets are like the battery's negative terminal. Figure 2-7 shows how applying electrical pressure to a circuit using a battery causes electrons to flow through it. This flow of electrons is called electric current, or often just current. Electric current is limited by the resistor. This current is what causes the diode to emit light.



**Figure 2-7**  
LED On Circuit Electron Flow

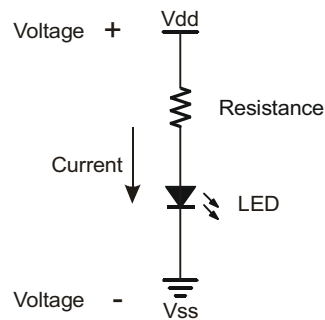
*The minus signs with the circles around them are used to show electrons flowing from the battery's negative terminal to its positive terminal.*





**Chemical reactions** inside the battery supply the circuit with current. The battery's negative terminal contains a compound that has molecules with extra electrons (shown in Figure 2-7 by minus-signs). The battery's positive terminal has a chemical compound with molecules that are missing electrons (shown by plus-signs). When an electron leaves a molecule in the negative terminal and travels through the wire, it is called a free electron (also shown by minus-signs). The molecule that lost that extra electron no longer has an extra negative charge; it is now called neutral (shown by an N). When an electron gets to the positive terminal, it joins a molecule that was missing an electron, and now that molecule is neutral too.

Figure 2-8 shows how the flow of electricity through the LED circuit is described using schematic notation. The electric pressure across the circuit is called voltage. The + and – signs are used to show the voltage applied to a circuit. The arrow shows the current flowing through the circuit. This arrow is almost always shown pointing the opposite direction of the actual flow of electrons. Benjamin Franklin is credited with not having been aware of electrons when he decided to represent current flow as charge passing from the positive to negative terminal of a circuit. By the time physicists discovered the true nature of electric current, the convention was already well established.



**Figure 2-8**  
LED-On Circuit  
Schematic Showing  
Conventional Voltage  
and Current Flow

*The + and – signs show voltage applied to the circuit, and the arrow shows current flow through the circuit.*



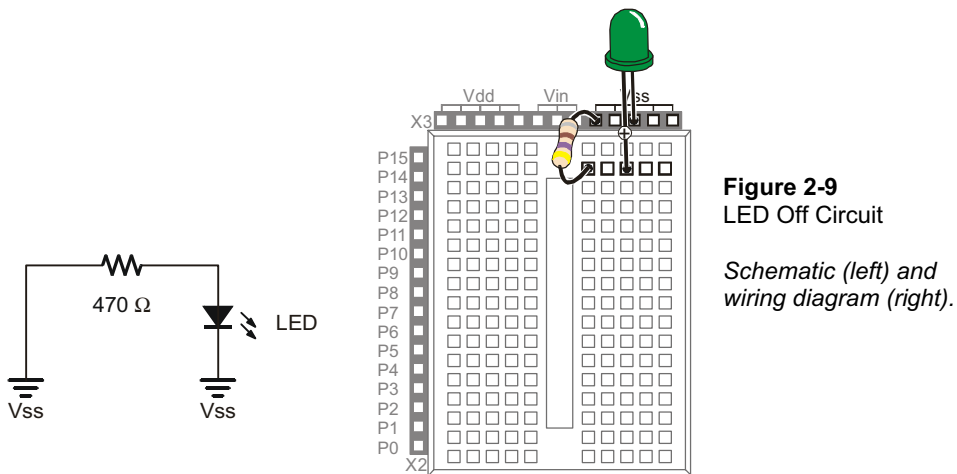
**A schematic drawing** (like Figure 2-8) is a picture that explains how one or more circuits are connected. Schematics are used by students, electronics hobbyists, electricians, engineers, and just about everybody else who works with circuits.

**Appendix F: More about Electricity:** This appendix contains some glossary terms and an activity you can try to get more familiar with measurements of voltage, current and resistance.

### Your Turn – Modifying the LED Test Circuit

In the next activity, you will program the BASIC Stamp to turn the LED on, then off, then on again. The BASIC Stamp will do this by switching the LED circuit between two different connections, Vdd and Vss. You just finished working with the circuit where the resistor is connected to Vdd, and the LED emits light. Make the changes shown in Figure 2-9 to verify that the LED will turn off (not emit light) when the resistor's lead is disconnected from Vdd and connected to Vss.

- √ Disconnect power from your Board of Education or HomeWork Board.
- √ Unplug the resistor lead that's plugged into the Vdd socket, and plug it into a socket labeled Vss as shown in Figure 2-9.
- √ Reconnect power to your Board of Education or HomeWork Board.
- √ Check to make sure your green LED is **not emitting light**. It should not glow green.



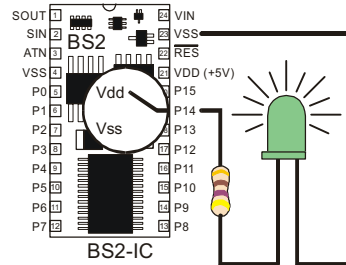
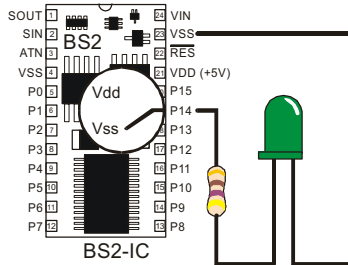
**Figure 2-9**  
LED Off Circuit

*Schematic (left) and wiring diagram (right).*

### ACTIVITY #2: ON/OFF CONTROL WITH THE BASIC STAMP

In Activity #1, two different circuits were built and tested. One circuit made the LED emit light while the other did not. Figure 2-10 shows how the BASIC Stamp can do the same thing if you connect an LED circuit to one of its I/O pins. In this activity, you will connect the LED circuit to the BASIC Stamp and program it to turn the LED on and off.

You will also experiment with programs that make the BASIC Stamp do this at different speeds.



**Figure 2-10**  
BASIC Stamp  
Switching

*The BASIC Stamp can be programmed to internally connect the LED circuit's input to Vdd or Vss.*

There are two big differences between changing the connection manually and having the BASIC Stamp do it. First, the BASIC Stamp doesn't have to cut the power when it changes the LED circuit's supply from Vdd to Vss. Second, while a human can make that change several times a minute, the BASIC Stamp can do it thousands of times per second!

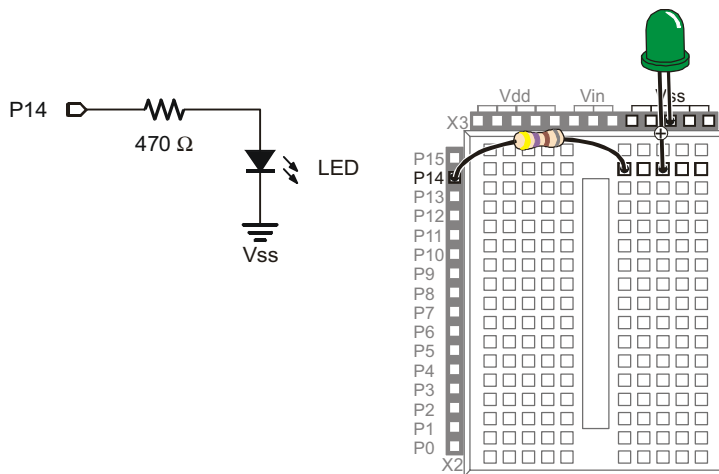
### **LED Test Circuit Parts**

Same as Activity #1.

### **Connecting the LED Circuit to the BASIC Stamp**

The LED circuit shown in Figure 2-11 is wired almost the same as the circuit in the previous exercise. The difference is that the resistor's lead that was manually switched between Vdd and Vss is now plugged into a BASIC Stamp I/O pin.

- √ Disconnect power from your Board of Education or HomeWork Board.
- √ Modify the circuit you were working with in Activity #1 so that it matches Figure 2-11.



**Figure 2-11**  
BASIC Stamp  
Controlled LED  
Circuit

*The LED circuit's  
input is now  
connected to a  
BASIC Stamp I/O  
pin instead of Vdd  
or Vss.*



**Resistors are essential.** Always remember to use a resistor. Without it, too much current will flow through the circuit, and it could damage any number of parts in your circuit, BASIC Stamp, or Board of Education or HomeWork Board.

### Turning the LED On/Off with a Program

The example program makes the LED blink on and off one time per second. It introduces several new programming techniques at once. After running it, you will experiment with different parts of the program to better understand how it works.

#### **Example Program: LedOnOff.bs2**

- ✓ Enter the LedOnOff.bs2 code into the BASIC Stamp Editor.
- ✓ Reconnect power to your Board of Education or HomeWork Board.
- ✓ Run the program.
- ✓ Verify that the LED flashes on and off once per second.
- ✓ Disconnect power when you are done with the program.

```
'What's a Microcontroller - LedOnOff.bs2
'Turn an LED on and off. Repeat 1 time per second indefinitely.

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

DEBUG "The LED connected to Pin 14 is blinking!"
```

```
DO
HIGH 14
PAUSE 500
LOW 14
PAUSE 500
LOOP
```

### How LedOnOff.bs2 Works

The command `DEBUG "The LED connected to Pin 14 is blinking!"` makes this statement appear in the Debug Terminal. The command `HIGH 14` causes the BASIC Stamp to internally connect I/O pin P14 to Vdd. This turns the LED on.

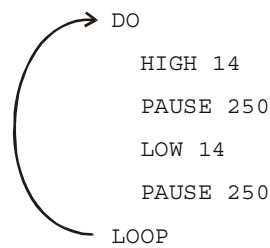
The command `PAUSE 500` causes the BASIC Stamp to do nothing for ½ a second while the LED stays on. The number 500 tells the `PAUSE` command to wait for 500/1000 of a second. The number that follows `PAUSE` is called an argument. If you look up `PAUSE` in the BASIC Stamp Manual, you will discover that it calls this number the *Duration* argument. The name duration was chosen for this argument to show that the `PAUSE` command pauses for a certain ‘duration’ of time, in milliseconds.



**What's a Millisecond?** A millisecond is 1/1000 of a second. It is abbreviated as ms. It takes 1000 ms to equal one second.

The command `LOW 14` causes the BASIC Stamp to internally connect I/O pin P14 to Vss. This turns the LED off. Since `LOW 14` is followed by another `PAUSE 500`, the LED stays off for half a second.

The reason the code repeats itself over and over again is because it is nested between the PBASIC keywords `DO` and `LOOP`. Figure 2-12 shows how a `DO...LOOP` works. By placing the code segment that turns the LED on and off with pauses between `DO` and `LOOP`, it tells the BASIC Stamp to execute those four commands over and over again. The result is that the LED flashes on and off, over and over again. It will keep flashing until you disconnect power, press and hold the Reset button, or until the battery runs out.



**Figure 2-12**  
DO...LOOP

*The code between the keywords DO and LOOP get executed over and over again.*

### **A Diagnostic Test for your Computer**

A very few computers, such as some laptops, will halt the PBASIC program after the first time through a `DO . . . LOOP` loop. These computers have a non-standard serial port design. By placing a `DEBUG` command the program `LedOnOff.bs2`, the open Debug Terminal prevents this from possibly happening. You will next re-run this program without the `DEBUG` command to see if your computer has this non-standard serial port problem. It is not likely, but it would be important for you to know.

- √ Open `LedOnOff.bs2`.
- √ Delete the entire `DEBUG` instruction.
- √ Run the modified program while you observe your LED.

If the LED blinks on and off continuously, just as it did when you ran the original program with the `DEBUG` command, your computer will not have this problem.

If the LED blinked on and off only once and then stopped, you have a computer with a non-standard serial port design. If you disconnect the serial cable from your board and press the Reset button, the BASIC Stamp will run the program properly without freezing. In programs you write yourself, you should add a single command:

```
DEBUG "Program Running!"
```

right after the compiler directives. This will open the Debug Terminal and keep the COM port open. This will prevent your programs from freezing after one pass through the `DO...LOOP`, or any of the other looping commands you will be learning in later chapters. You will see this command in some of the example programs that would not

otherwise need a **DEBUG** instruction. So, you should be able to run all of the remaining programs in this book even if your computer failed the diagnostic test.

### Your Turn – Timing and Repetitions

By changing the **PAUSE** command's *Duration* argument you can change the amount of time the LED stays on and off. For example, by changing both the *Duration* arguments to 250, it will cause the LED to flash on and off twice per second. The **DO...LOOP** in your program will now look like this:

```
DO
    HIGH 14
    PAUSE 250
    LOW 14
    PAUSE 250
LOOP
```

- √ Open LedOnOff.bs2
- √ Change the **PAUSE** command's *Duration* arguments from 500 to 250, and re-run the program.

If you want to make the LED blink on and off once every three seconds, with the low time twice as long as the high time, you can program the **PAUSE** command after the **HIGH 14** command so that it takes one second using **PAUSE 1000**. The **PAUSE** command after the **LOW 14** command will have to be **PAUSE 2000**.

```
DO
    HIGH 14
    PAUSE 1000
    LOW 14
    PAUSE 2000
LOOP
```

- √ Modify and re-run the program using the code snippet above.

A fun experiment is to see how short you can make the pauses and still see that the LED is flashing. When the LED is flashing very fast, but it looks like it's just on, it's called

persistence of vision. Here is how to test to see what your persistence of vision threshold is:

- √ Try modifying both of your **PAUSE** command's *Duration* arguments so that they are 100.
- √ Re-run your program and check for flicker.
- √ Reduce both *Duration* arguments by 5 and try again.
- √ Keep reducing the *Duration* arguments until the LED appears to be on all the time with no flicker. It will be dimmer than normal, but it should not appear to flicker.

One last thing to try is to create a one-shot LED flasher. When the program runs, the LED flashes only once. This is a way to look at the functionality of the **DO...LOOP**. You can temporarily remove the **DO...LOOP** from the program by placing an apostrophe to the left of both the **DO** and **LOOP** keywords as shown below.

```
' DO  
  
HIGH 14  
PAUSE 1000  
LOW 14  
PAUSE 2000  
  
' LOOP
```

- √ Modify and re-run the program using the code snippet above.
- √ Explain what happened, why did the LED only flash once?



**Commenting a line of code:** Placing an apostrophe to the left of a command changes it into a comment. This is a useful tool because you don't actually have to delete the command to see what happens if you remove it from the program. It is much easier to add and remove an apostrophe than it is to delete and re-type the commands.

### ACTIVITY #3: COUNTING AND REPEATING

In the previous activity, the LED circuit either flashed on and off all the time, or it flashed once and then stopped. What if you only want the LED to flash on and off ten times? Computers (including the BASIC Stamp) are great at keeping running totals of how many times something happens. Computers can also be programmed to make



decisions based on a variety of conditions. In this activity, you will program the BASIC Stamp to stop flashing the LED on and off after ten repetitions.

### **Counting Parts and Test Circuit**

Use the example circuit shown in Figure 2-11 on page 48.

### **How Many Times?**

There are many ways to make the LED blink on and off ten times. The simplest way is to use a **FOR...NEXT** loop. The **FOR...NEXT** loop is similar to the **DO...LOOP**. Although either loop can be used to repeat commands a fixed number of times, **FOR...NEXT** is easier to use.

The **FOR...NEXT** loop depends on a variable to track how many times the LED has blinked on and off. A variable is a word of your choosing that is used to store a value. The next example program uses the word `counter` to ‘count’ how many times the LED has been turned on and off.



#### **Picking words for variable names has several rules:**

1. The name cannot be a word that is already used by PBASIC. These words are called reserved words, and some examples that you should already be familiar with are: **DEBUG**, **PAUSE**, **HIGH**, **LOW**, **DO**, and **LOOP**.
2. The name cannot contain a space.
3. Even though the name can contain letters, numbers, or underscores, it must begin with a character.
4. The name must be less than 33 characters long.

### **Example Program: LedOnOffTenTimes.bs2**

The program LedOnOffTenTimes.bs2 demonstrates how to use a **FOR...NEXT** loop to blink an LED on and off ten times.

- √ Your test circuit from Activity #2 should be built (or rebuilt) and ready to use.
- √ Enter the LedOnOffTenTimes.bs2 code into the BASIC Stamp Editor.
- √ Connect power to your Board of Education or HomeWork Board.
- √ Run the program.
- √ Verify that the LED flashes on and off ten times.

- √ Run the program a second time, and verify that the value of `counter` shown in the Debug Terminal accurately tracks how many times the LED blinked. Hint: instead of clicking *Run* a second time, you can press and release the Reset button on your Board of Education or HomeWork Board.

```
' What's a Microcontroller - LedOnOffTenTimes.bs2
' Turn an LED on and off. Repeat 10 times.

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

counter VAR Byte

FOR counter = 1 TO 10

  DEBUG ? counter

  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500

NEXT

DEBUG "All done!"

END
```


### How LedOnOffTenTimes.bs2 Works

This PBASIC statement

```
counter VAR Byte
```

tells the BASIC Stamp Editor that your program will use the word `counter` as a variable that can store a byte's worth of information.

**What's a Byte?** A byte is enough memory to store a number between 0 and 255. The BASIC Stamp has four different types of variables, and each can store a different range of numbers:

 **Table 2-2: Variable Types and Values They Can Store**

Variable type	Range of Values
Bit	0 to 1
Nib	0 to 15
Byte	0 to 255
Word	0 to 65535

The question mark formatter before a variable in a **DEBUG** command tells the Debug Terminal to display the name of the variable and its value. This is how the command

```
DEBUG ? counter
```

displays both the name and the value of the **counter** variable in the Debug Terminal.

The **FOR...NEXT** loop and all the commands inside it are shown below. The statement **FOR counter = 1 to 10** tells the BASIC Stamp that it will have to set the **counter** variable to 1, then keep executing commands until it gets to the **NEXT** statement. When the BASIC Stamp gets to the **NEXT** statement, it jumps back to the **FOR** statement. The **FOR** statement adds one to the value of **counter**. Then, it checks to see if **counter** is greater than ten yet. If not, it repeats the process. When the value of **counter** finally reaches eleven, the program skips the commands between the **FOR** and **NEXT** statements and moves on to the command that comes after the **NEXT** statement.

```
FOR counter = 1 to 10

  DEBUG ? counter, CR

  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500

NEXT
```

The command that comes after the **NEXT** statement is:

```
DEBUG "All done!"
```

This command is included just to show what the program does after ten times through the **FOR...NEXT** loop. It moves on to the command that comes after the **NEXT** statement.

### Your Turn – Other Ways to Count

- √ Replace the statement

```
FOR counter = 1 to 10    with this:    FOR counter = 1 to 20
```

in `LedOnOffTenTimes.bs2` and re-run the program. What did the program do differently, and was this expected?

- √ Try a second modification to the **FOR** statement. This time, change it to

```
FOR counter = 20 to 120 STEP 10
```

How many times did the LED flash? What values displayed in the Debug Terminal?

## ACTIVITY #4: BUILDING AND TESTING A SECOND LED CIRCUIT

Indicator LEDs can be used to tell the machine's user many things. Many devices need two, three, or more LEDs to tell the user if the machine is ready or not, if there is a malfunction, if it's done with a task, and so on.

In this activity, you will repeat the LED circuit test in Activity #1 for a second LED circuit. Then you will adjust the example program from Activity #2 to make sure the LED circuit is properly connected to the BASIC Stamp. After that, you will modify the example program from Activity #2 to make the LEDs operate in tandem.

### Extra Parts

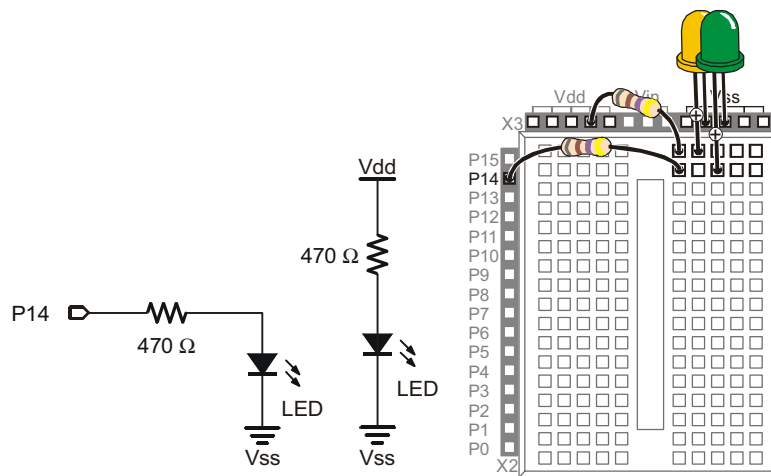
In addition to the parts you used in Activities 1 and 2, you will need these parts:

- (1) LED - yellow
- (1) Resistor – 470  $\Omega$  (yellow-violet-brown)

### **Building and Testing the Second LED Circuit**

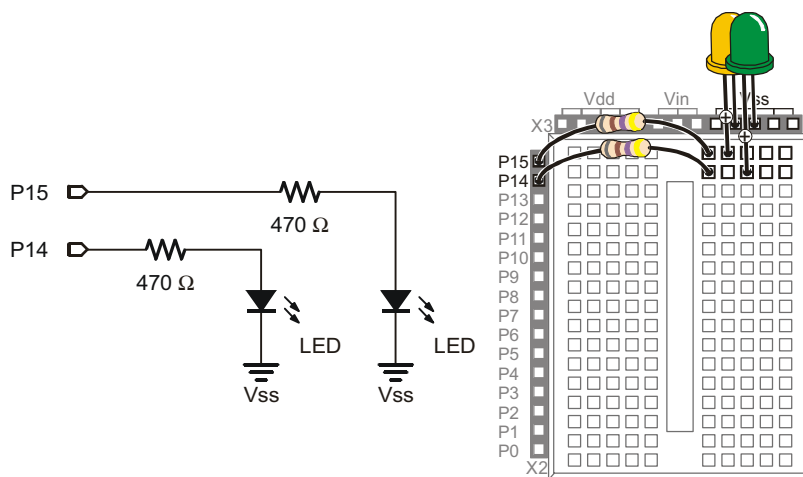
In Activity #1, you manually tested the first LED circuit to make sure it worked before connecting it to the BASIC Stamp. Before connecting the second LED circuit to the BASIC Stamp, it's important to test it too.

- √ Disconnect power from your Board of Education or HomeWork Board.
- √ Construct the second LED circuit as shown in Figure 2-13.
- √ Reconnect power to your Board of Education or HomeWork Board.
- √ Did the LED circuit you just added turn on? If yes, then continue. If no, Activity #1 has some trouble-shooting suggestions that you can repeat for this circuit.



**Figure 2-13**  
Manual Test  
Circuit for  
Second LED

- √ Disconnect power to your Board of Education or HomeWork Board.
- √ Modify the second LED circuit you just tested by connecting the LED circuit's resistor lead (input) to P15 as shown in Figure 2-14.



**Figure 2-14**  
Connecting  
the Second  
LED to the  
BASIC  
Stamp

*Schematic  
(left) and  
wiring  
diagram  
(right).*

### Using a Program to Test the Second LED Circuit

In Activity #2, you used an example program and the **HIGH** and **LOW** commands to control the LED circuit connected to P14. These commands will have to be modified to control the LED circuit connected to P15. Instead of using **HIGH 14** and **LOW 14**, you will use **HIGH 15** and **LOW 15**.

#### **Example Program: TestSecondLed.bs2**

- ✓ Enter TestSecondLed.bs2 into the BASIC Stamp Editor.
- ✓ Connect power to your Board of Education or HomeWork Board.
- ✓ Run TestSecondLED.bs2.
- ✓ Make sure the LED circuit connected to P15 is flashing. If the LED connected to P15 flashes, move on to the next example (Controlling Both LEDs). If the LED circuit connected to P15 is not flashing, check your circuit for wiring errors and your program for typing errors and try again.

```
' What's a Microcontroller - TestSecondLed.bs2
' Turn LED connected to P15 on and off.
' Repeat 1 time per second indefinitely.

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

DEBUG "Program Running!"
```

```
DO
HIGH 15
PAUSE 500
LOW 15
PAUSE 500
LOOP
```

### **Controlling Both LEDs**

Yes, you can flash both LEDs at once. One way you can do this is to use two **HIGH** commands before the first **PAUSE** command. One **HIGH** command sets P14 high, and the next **HIGH** command sets P15 high. You will also need two **LOW** commands to turn both LEDs off. It's true that both LEDs will not turn on and off at exactly the same time because one is turned on or off after the other. However, there is no more than a millisecond's difference between the two changes, and the human eye will not detect it.

### **Example Program: FlashBothLeds.bs2**

- √ Enter the FlashBothLeds.bs2 code into the BASIC Stamp Editor.
- √ Run the program.
- √ Verify that both LEDs appear to flash on and off at the same time.

```
' What's a Microcontroller - FlashBothLeds.bs2
' Turn LEDs connected to P14 and P15 on and off.

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

DEBUG "Program Running!"

DO

HIGH 14
HIGH 15
PAUSE 500
LOW 14
LOW 15
PAUSE 500

LOOP
```

### Your Turn – Alternate LEDs

You can cause the LEDs to alternate by swapping the **HIGH** and **LOW** commands that control one of the I/O pins. This means that while one LED is on, the other will be off.

- √ Modify FlashBothLeds.bs2 so that the commands between the **DO** and **LOOP** keywords look like this:

```
HIGH 14  
LOW 15  
PAUSE 500  
LOW 14  
HIGH 15  
PAUSE 500
```

- √ Run the modified version of FlashBothLeds.bs2 and verify that the LEDs flash alternately on and off.

### ACTIVITY #5: USING CURRENT DIRECTION TO CONTROL A BI-COLOR LED

The device shown in Figure 2-15 is a security monitor for electronic keys. When an electronic key with the right code is used, the LED changes color, and a door opens. This kind of LED is called a bi-color LED. This activity answers two questions:

1. How does the LED change color?
2. How can you run one with the BASIC Stamp?



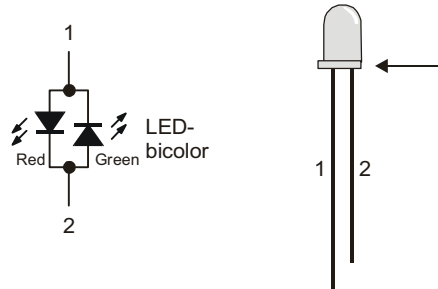


**Figure 2-15**  
Bi-color LED in a Security Device

*When the door is locked, this bi-color LED glows red. When the door is unlocked by an electronic key with the right code, the LED turns green.*

### Introducing the Bi-Color LED

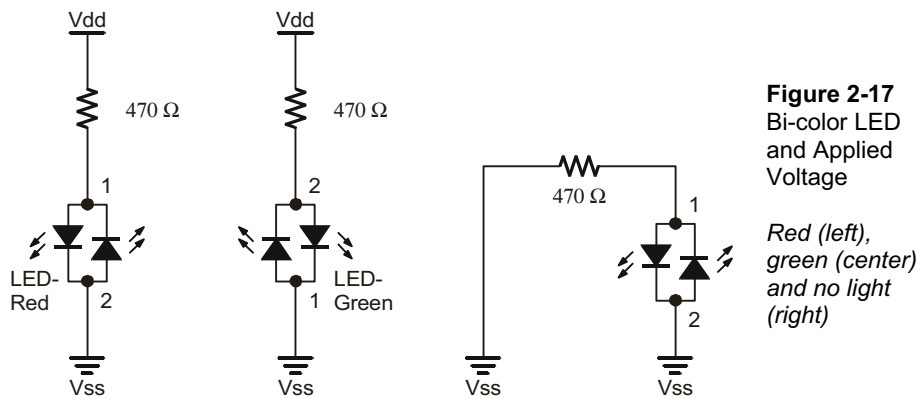
The bi-color LED's schematic symbol and part drawing are shown in Figure 2-16.



**Figure 2-16**  
Bi-color LED

*Schematic symbol (left) and part drawing (right).*

The bi-color LED is really just two LEDs in one package. Figure 2-17 shows how you can apply voltage in one direction and the LED will glow red. By disconnecting the LED and plugging it back in reversed, the LED will then glow green. As with the other LEDs, if you connect both terminals of the circuit to V<sub>ss</sub>, the LED will not emit light.



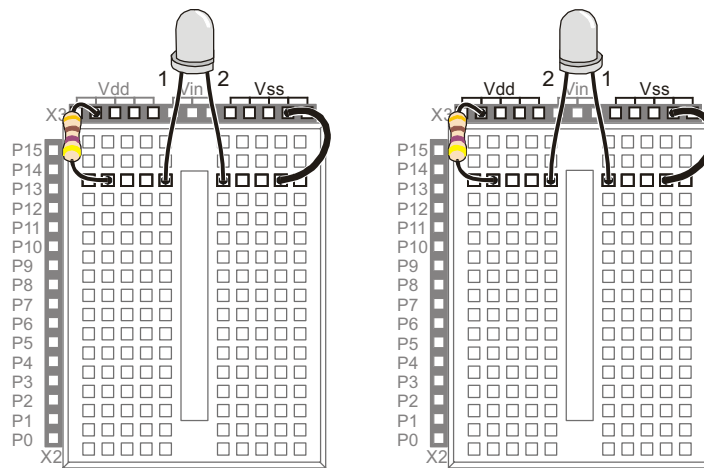
### **Bi-Color LED Circuit Parts**

- (1) LED – bi-color
- (1) Resistor 470  $\Omega$  (yellow-violet-brown)
- (1) Jumper wire

### **Building and Testing the Bi-Color LED Circuit**

Figure 2-18 shows the manual test for the bi-color LED.

- √ Disconnect power from your Board of Education or HomeWork Board.
- √ Build the circuit shown on the left side of Figure 2-18.
- √ Reconnect power and verify that the bi-color LED is emitting red light.
- √ Disconnect power again.
- √ Modify your circuit so that it matches the right side of Figure 2-18.
- √ Reconnect power.
- √ Verify that the bi-color LED is now emitting green light.
- √ Disconnect power.

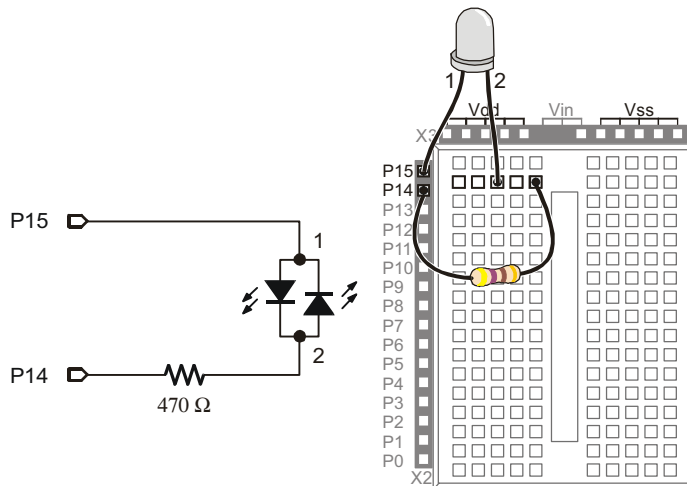


**Figure 2-18**  
Manual bi-color LED Test

*Bi-color LED red (left) and green (right).*

Controlling a bi-color LED with the BASIC Stamp requires two I/O pins. After you have manually verified that the bi-color LED works using the manual test, you can connect the circuit to the BASIC Stamp as shown in Figure 2-19.

- √ Connect the bi-color LED circuit to the BASIC Stamp as shown in Figure 2-19.

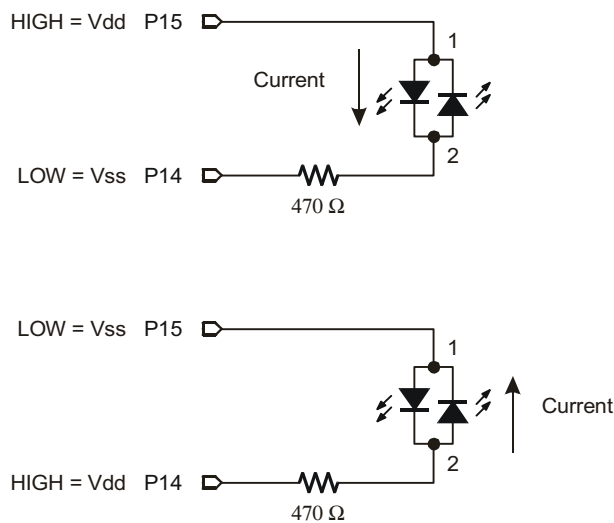


**Figure 2-19**  
Bi-color LED Connected to BASIC Stamp

### **BASIC Stamp Bi-Color LED Control**

Figure 2-20 shows how you can use P15 and P14 to control the current flow in the bi-color LED circuit. The upper schematic shows how current flows through the red LED when P15 is set to Vdd and P14 is set to Vss. This is because the red LED will let current flow through it when electrical pressure is applied as shown, but the green LED acts like a closed valve and does not let current through it. The bi-color LED glows red.

The lower schematic shows what happens when P15 is set to Vss and P14 is set to Vdd. The electrical pressure is now reversed. The red LED shuts off and does not allow current through. Meanwhile, the green LED turns on, and current passes through the circuit in the opposite direction.



**Figure 2-20**  
Manual bi-color  
LED Test

*Current through  
Red LED (above)  
and Green LED  
(below).*

Figure 2-20 also shows the key to programming the BASIC Stamp to make the bi-color LED glow two different colors. The upper schematic shows how to make the bi-color LED red using **HIGH 15** and **LOW 14**. The lower schematic shows how to make the bi-color LED glow green by using **LOW 15** and **HIGH 14**. To turn the LED off, send low signals to both P14 and P15 using **LOW 15** and **LOW 14**. In other words, use **LOW** on both pins.



**The bi-color LED will also turn off** if you send high signals to both P14 and P15. Why? Because the electrical pressure is the same at P14 and P15 regardless of whether you set both I/O pins high or low.

2

### Example Program: TestBiColorLED.bs2

- √ Reconnect power.
- √ Enter the TestBiColorLed.bs2 code into the BASIC Stamp Editor.
- √ Run the program.
- √ Verify that the LED cycles through the red, green, and off states.

```
' What's a Microcontroller - TestBiColorLed.bs2
' Turn bi-color LED red, then green, then off in a loop.

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

DEBUG "Program Running!"

DO

HIGH 15          ' Red
LOW 14
PAUSE 500

LOW 15          ' Green
HIGH 14
PAUSE 500

LOW 15          ' Off
LOW 14
PAUSE 500

LOOP
```

### Your Turn – Lights Display

In Activity #3, a variable named **counter** was used to control how many times the LED blinked. What happens if you use the value **counter** to control the **PAUSE** command's *Duration* argument?

- √ Rename and save TestBiColorLed.bs2 as TestBiColorLedYourTurn.bs2.
- √ Add a counter variable declaration before the **DO** statement:

```
counter VAR BYTE
```

√ Nest the **FOR...NEXT** loop below within the **DO...LOOP**.

```
FOR counter = 1 to 50

    HIGH 15
    LOW 14
    PAUSE counter

    LOW 15
    HIGH 14
    PAUSE counter

NEXT
```

When you are done, your code should look like this:

```
counter VAR BYTE

DO

    FOR counter = 1 to 50

        HIGH 15
        LOW 14
        PAUSE counter

        LOW 15
        HIGH 14
        PAUSE counter

    NEXT

LOOP
```

At the beginning of each pass through the **FOR...NEXT** loop, the **PAUSE** value (*Duration* argument) is only one millisecond. Each time through the **FOR...NEXT** loop, the pause gets longer by one millisecond at a time until it gets to 50 milliseconds. The **DO...LOOP** causes the **FOR...NEXT** loop to execute over and over again.

√ Run the modified program and observe the effect.

## SUMMARY

The BASIC Stamp can be programmed to switch a circuit with a light emitting diode (LED) indicator light on and off. LED indicators are useful in a variety of places including many computer monitors, disk drives, and other devices. The LED was introduced along with a technique to identify its anode and cathode terminals. An LED circuit must have a resistor to limit the current passing through it. Resistors were introduced along with one of the more common coding schemes you can use to figure out a resistor's value.

The BASIC Stamp switches an LED circuit on and off by internally connecting an I/O pin to either Vdd or Vss. The **HIGH** command can be used to make the BASIC Stamp internally connect one of its I/O pins to Vdd, and the **LOW** command can be used to internally connect an I/O pin to Vss. The **PAUSE** command is used to cause the BASIC Stamp to not execute commands for an amount of time. This was used to make LEDs stay on and/or off for certain amounts of time. This amount of time is determined by the number used in the **PAUSE** command's *Duration* argument.

The **DO...LOOP** can be used to create an infinite loop. The commands between the **DO** and **LOOP** keywords will execute over and over again. Even though this is called an infinite loop, the program can still be re-started by disconnecting and reconnecting power or pressing and releasing the Reset button. A new program can also be downloaded to the BASIC Stamp, and this will erase the program with the infinite loop.

Current direction and voltage polarity were introduced using a bi-color LED. If voltage is applied across the LED circuit, current will pass through it in one direction, and it glows a particular color. If the voltage polarity is reversed, current travels through the circuit in the opposite direction and it glows a different color.

## Questions

1. What is the name of this Greek letter:  $\Omega$ , and what measurement does  $\Omega$  refer to?
2. Which resistor would allow more current through the circuit, a 470  $\Omega$  resistor or a 1000  $\Omega$  resistor?
3. How do you connect two wires using a breadboard? Can you use a breadboard to connect four wires together?

4. What do you always have to do before modifying a circuit that you built on a breadboard?
5. How long would `PAUSE 10000` last?
6. How would you cause the BASIC Stamp to do nothing for an entire minute?
7. What are the different types of variables?
8. Can a byte hold the value 500?
9. What will the command `HIGH 7` do?

### **Exercises**

1. Draw the schematic of an LED circuit like the one you worked with in Activity #2, but connect the circuit to P13 instead of P14. Explain how you would modify `LedOnOff.bs2` on Page 48 so that it will make your LED circuit flash on and off four times per second.
2. Explain how to modify `LedOnOffTenTimes.bs2` so that it makes the LED circuit flash on and off 5000 times before it stops. Hint: you will need to modify just two lines of code.

### **Project**

1. Make a 10-second countdown using one yellow LED and one bi-color LED. Make the bi-color LED start out red for 3 seconds. After 3 seconds, change the bi-color LED to green. When the bi-color LED changes to green, flash the yellow LED on and off once every second for ten seconds. When the yellow LED is done flashing, the bi-color LED should switch back to red and stay that way.

### **Solutions**

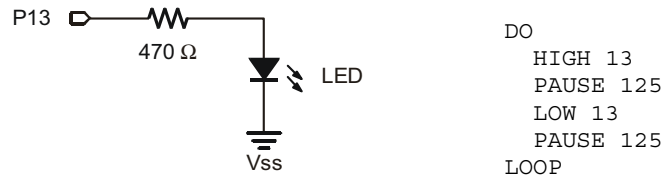
- Q1. Omega refers to the ohm which measures how strongly something resists current flow.
- Q2. A 470  $\Omega$  resistor: higher values resist more strongly than lower values, therefore lower values allow more current to flow.
- Q3. To connect 2 wires, plug the 2 wires into the same row of 5 sockets. You can connect 4 wires by plugging all 4 wires into the same row of 5 sockets.
- Q4. Disconnect the power.
- Q5. 10 seconds.
- Q6. `PAUSE 60000`
- Q7. Bit, Nib, Byte, and Word



Q8. No. The largest value a byte can hold is 255. The value 500 is out of range for a byte.

Q9. **HIGH 7** will cause the BASIC Stamp to internally connect I/O pin P7 to Vdd.

E1. The **PAUSE Duration** must be reduced to  $500\text{ms} / 4 = 125\text{ms}$ . To use I/O pin P13, **HIGH 14** and **LOW 14** have been replaced with **HIGH 13** and **LOW 13**.



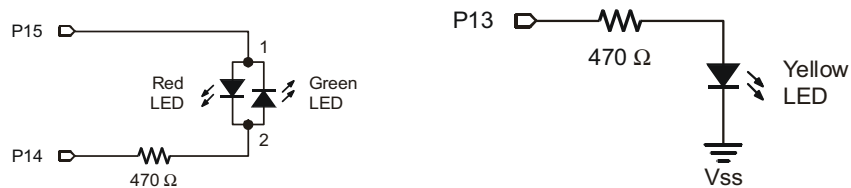
E2. The **counter** variable has to be changed to **word** size, and the **FOR** statement has to be modified to count from 1 to 5000).

```

counter VAR Word
FOR counter = 1 to 5000
  DEBUG ? counter, CR
  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500
NEXT

```

P1. The bi-color LED schematic, on the left, is unchanged from Figure 2-20 on page 64. The yellow LED schematic is based on Figure 2-11 on page 48. For this project P14 was changed to P13.



```

' What's a Microcontroller - Ch02Prj01_Countdown.bs2
' 10 Second Countdown with Red, Yellow, Green LED
' Red/Green: Bicolor LED on P15, P14. Yellow: P13
' {$STAMP BS2}
' {$PBASIC 2.5}
DEBUG "Program Running!"

counter VAR Byte

```

```
' Red for three seconds          ' Bi-color LED Red
HIGH 15
LOW 14
PAUSE 3000

' Green for 10 seconds...       ' Bi-color LED Green
LOW 15
HIGH 14

' ...while the yellow LED is flashing
FOR counter = 1 TO 10
  HIGH 13                      ' Yellow LED on
  PAUSE 500
  LOW 13                       ' Yellow LED off
  PAUSE 500
NEXT

' Red stays on                 ' Bi Color LED Red
HIGH 15
LOW 14
```

### **Further Investigation**

The resources listed here are available for free download from the Parallax web site and are also included on the Parallax CD.

#### ***“BASIC Stamp Manual”, Users Manual, Version 2.0c, Parallax Inc., 2000***

The *BASIC Stamp Manual* has more examples you can try and information that further explains the following commands: **HIGH**, **LOW**, **PAUSE**, the **DEBUG ?** formatter, and **FOR...NEXT**.

#### ***“Basic Analog and Digital”, Student Guide, Version 2.0, Parallax Inc., 2003***

*Basic Analog and Digital* uses LEDs to describe counting in binary, describes analog conditions, and it introduces new ways to adjust an LED's brightness.

#### ***“BASIC Stamp Editor Help File”, PBASIC 2.5 Version 2.0 Parallax Inc., 2003***

The PBASIC 2.5 Help File has information on **DO...LOOP**, which is new to PBASIC 2.5 and not included in the BASIC Stamp Manual. You can find this information by clicking the book icon on your BASIC Stamp Editor task bar, then selecting PBASIC Reference from the menu in the left sidebar window. This will open the PBASIC Command Reference alphabetical listing in the main window. Detailed information can be found by clicking on each command.