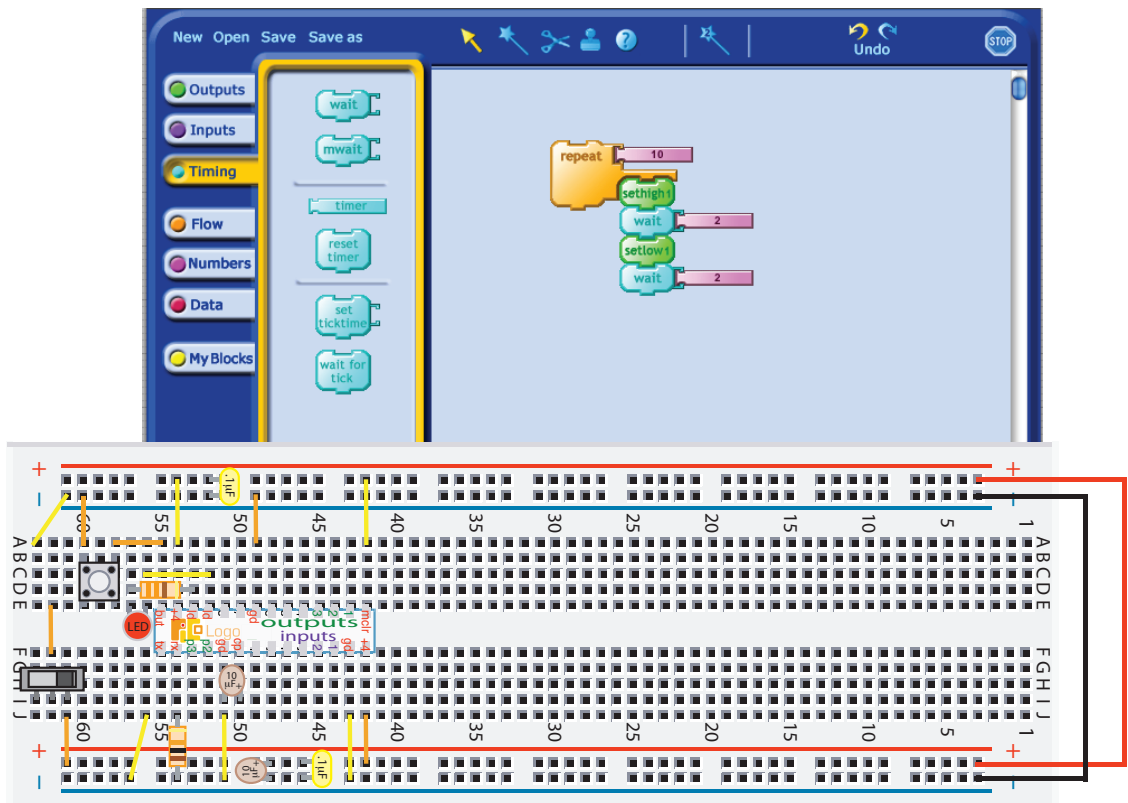


Getting Started with the



Getting Started with the PICO LogoChip

Electronics for Everyone	4
Build Your Own <i>LogoChip</i>.....	4
<i>LogoChip Parts List</i>	4
Tools suggested.....	5
<i>Overview of the Construction Process.....</i>	6
<i>Preliminaries.....</i>	7
The art of debugging	7
The LogoChip sticker.....	8
<i>Step by Step Instructions</i>	8
Step 1 - Connect the power and ground busses.....	9
Step 2 - Add the LogoChip	9
Step 3 - Connect the LogoChip to the ground bus	10
Step 4 - Connect the LogoChip to the power bus	11
Step 5 - Connect the LED	11
Step 6 - Make a “bootflash”	12
Step 7 - Add the capacitors	13
Step 8 - Add a power switch	14
Step 9 - Add the start/stop button.....	15
Step 10 - Add the serial connection	16
<i>LogoChip Schematic Drawing.....</i>	17
PicoBlocks for LogoChip.....	18
<i>Install PicoBlocks for LogoChip.....</i>	18
Troubleshooting	20
<i>Programming your LogoChip with PicoBlocks</i>	21
Tools.....	21
Getting Flashy	23
<i>Hello World Projects.....</i>	24
My Blocks	26
Making Colors.....	27
Making Music	31
<i>Getting a Sense of the World.....</i>	33
Digital Sensors	33
Analog Sensors.....	34
Theremin	36
Scoping out data	36
Try Colors	41
Clap sensor	41

Let's Get Moving	41
<i>Toothbrush hack</i>	44
Microphone PicoBlocks Language Reference	44
<i>Blocks Summary</i>	45
Outputs	45
Inputs	46
Timing	47
Flow.....	48
Numbers	49
Data	50
My Blocks	51
<i>The PicoBlocks Text Language</i>	51
Procedures	53
Make Your Own Blocks.....	55
<i>Variables</i>	56

Electronics for Everyone

Eventually the art went out of radio tinkering. Children forgot the pleasures of opening and eviscerating their parents' old Kadettes and Clubs. Solid electronic blocks replaced the radio set's messy innards—so where once you could learn by tugging at soldered wires and staring into the orange glow of the vacuum tubes, eventually nothing remained but featureless ready-made chips, the old circuits compressed a thousandfold or more. The transistor, a microscopic quirk of silicon, supplanted the reliably breakable tube, and so the world lost a well-used path into science.

— James Gleick,
Genius: The Life and Science of Richard Feynman
[1992]









Electronics lies at the center of much of technology of our times. We find ourselves surrounded by ever more powerful and, in many ways, ever more complicated and mysterious electronic gizmos and gadgets. In spite of the central role that these devices often play in our lives, the inner workings of these “black boxes” are almost completely hidden from view and are often poorly understood by their users. Electronics is typically viewed as a formidable subject that is best tackled by expert practitioners.

The *LogoChip Kit* is designed to help novices get started playing with the design and construction of interesting electronic circuits. The *LogoChip* is an easily programmable and inexpensive embedded microcontroller. Users can develop programs using a graphical programming language called *PicoBlocks*. *PicoBlocks* combines all the power and elegance of the Logo programming language with the ability to directly control the individual pins on the *LogoChip*.

Build Your Own *LogoChip*

LogoChip Parts List

You can build the *LogoChip* on a solderless breadboard following the instructions below. The *LogoChip* consists of a PIC24FJ32 microcontroller (made by Microchip) that has a Logo “virtual machine” installed on it. The parts that you need to construct the basic *LogoChip* circuit are listed below.

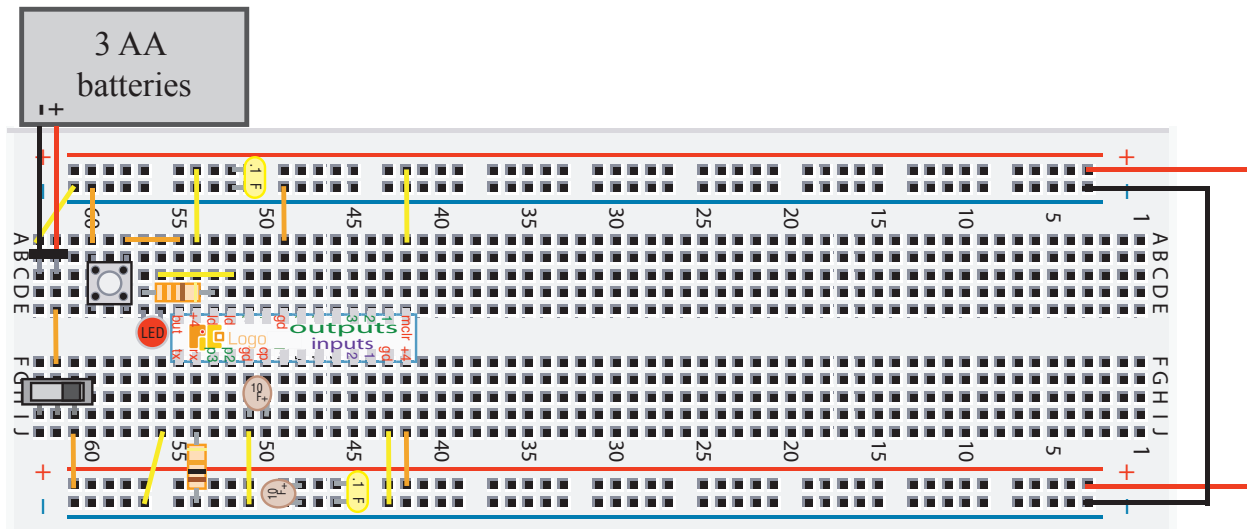
Parts List for the Basic <i>LogoChip</i> Circuit			
Quantity	Part	Manufacturer	Digi-Key Part Number
1	LogoChip 	Microchip	PIC24FJ32GA002-I/SP-ND
1	solderless breadboard	Twin	438-1045-ND
1	pushbutton 	Omron	SW400-ND
2	10 uF capacitors 	Panasonic	478-1837-ND
2	0.1 microfarad capacitors 	Kemet	399-2127-ND
1	bi-color LED 	Lite-on	160-1058-ND
1	10k resistor 	Yageo	10KQBK-ND
1	330 ohm resistor 	Yageo	330QBK-ND
1	power switch 	E-Switch	EG1903-ND
1	"AA" battery holder (3 cells)	Keystone	2465K-ND
3	"AA" batteries	Panasonic	P107-ND
1	serial interface board	PICO	
1	USB cable	Qualtek	Q361-ND

Tools suggested

- additional #22 solid hookup wire on a spool (such as Digi-Key part number C2117W-100-ND).
- a good wire stripper (such as Digi-Key part number WS6-ND). Be careful when stripping the wire. Make sure the aperture in the stripper is the right size for #22 wire. (If it's too small you will nick the wire, which can lead to nasty problems down the road.)
- A "multi-meter" for measuring voltage, current, and resistance. Either an analog (such as Digi-Key part number BK117B-ND) or a digital (such as Digi-Key part number BK2408-ND) meter will do.
- a small soldering iron (such as Digi-Key part number WP25-ND) for connecting solid hookup wire to the battery pack.

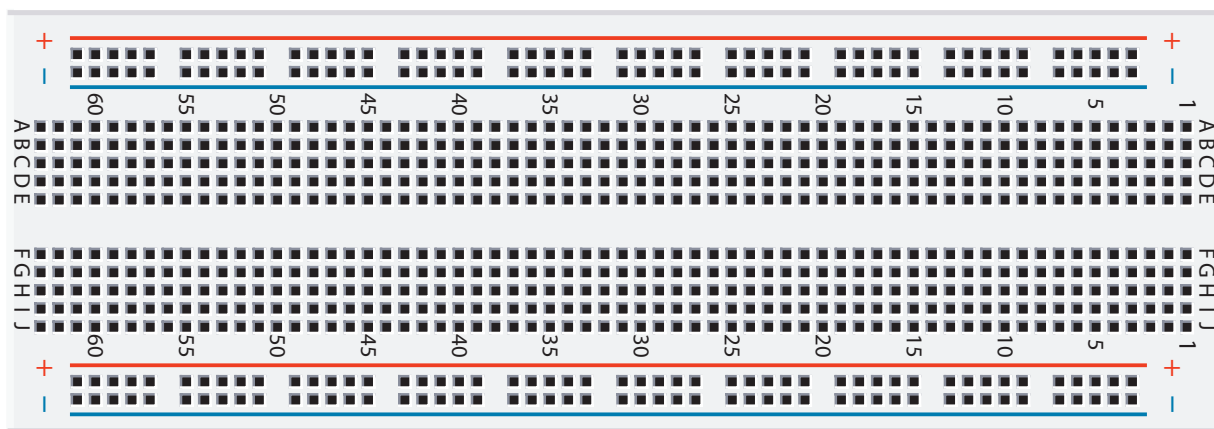
Overview of the Construction Process

A completed LogoChip circuit, constructed using the parts listed above, is shown in the figure below:



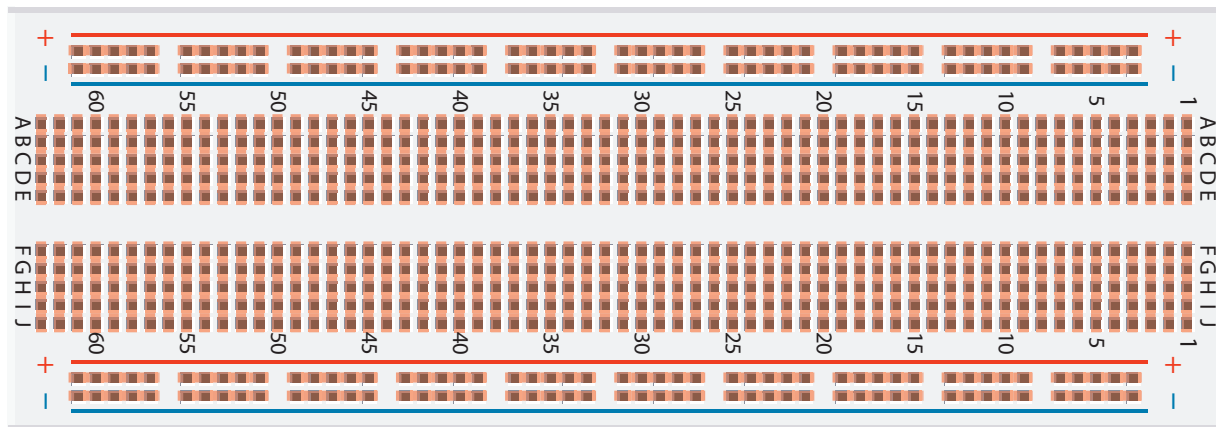
A completed LogoChip circuit

The circuit is constructed on a “solderless breadboard”:



A blank solderless breadboard

The red shaded regions in the figure below indicate which holes are connected on your breadboard.



Breadboard connections

Preliminaries

First, some general advice and preliminary steps before you start building the LogoChip circuit.

The solderless breadboard has an adhesive backing and needs to be mounted on a rigid backing in order to function properly. The breadboard comes packaged with a thin metal plate that can serve as a backing. Peel off the protective covering from the adhesive backing and attach the metal plate to the back of the breadboard.

The art of debugging

The last bit of preliminary advice is perhaps the most important point, though it is by far the hardest to express in words. As you build this circuit it is fairly likely that you will make some mistakes. Very likely, things won't work the first time through. Figuring out how to “debug the circuit”, that is, learning how to track down the cause of the problem, is an invaluable skill to develop. But the process of “debugging”, whether it involves an electronic circuit or some other complicated system, is something of an art, and therefore it is hard to give precise rules for how to go about doing it. There are however some basic underlying principles that underlie the art as it pertains to building the LogoChip. Two of the most important are:

- *Test frequently as you build.* A common mistake that novices make when building a circuit is to build the entire circuit before doing any testing. With this approach, if the fully assembled circuit fails to work properly, the root cause could be in any one of a large number of places. By breaking the assembly process into well defined steps and then trying to devise a test after each step to check if things are

working as expected, you will be in a much better position to catch a wiring mistake or a defective component. The step by step instructions are explicitly intended to introduce you to this habit of frequent testing.

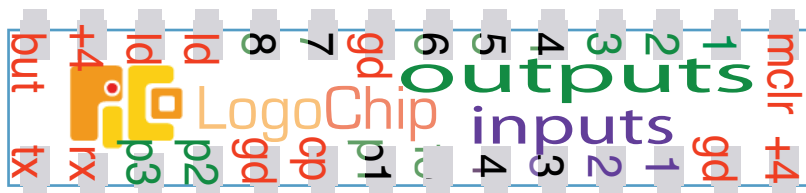
- *Be neat in your wiring.* It is hard to find a wiring error in a rat's nest of tangled wires. Keep your wires as short as possible and make sure when stripping the wires you only expose as much bare wire as needed to insert the wire in the breadboard. The pre stripped, color coded wires that come with your LogoChip Kit encourage this, since they are designed especially for use with solderless breadboards. Each color wire has a different length, but all the lengths are multiples of 0.1", and thus match up perfectly with the spacing between the holes in the breadboard.

The LogoChip sticker

The top of the LogoChip has a sticker, shown below, to help identify the pins on the LogoChip. (If your LogoChip has lost its sticker, you can print out a copy of this figure, carefully cut out a sticker, and use a glue stick to attach the sticker to a PIC24FJ32 microcontroller. **It is critical that the sticker be attached with the proper orientation. The top of the sticker (the end with the pins labeled **mclr** and **+4**) must be attached to the end of the microcontroller that has a notch in it.**



LogoChip Sticker -actual size



An enlarged view of a LogoChip sticker

Step by Step Instructions

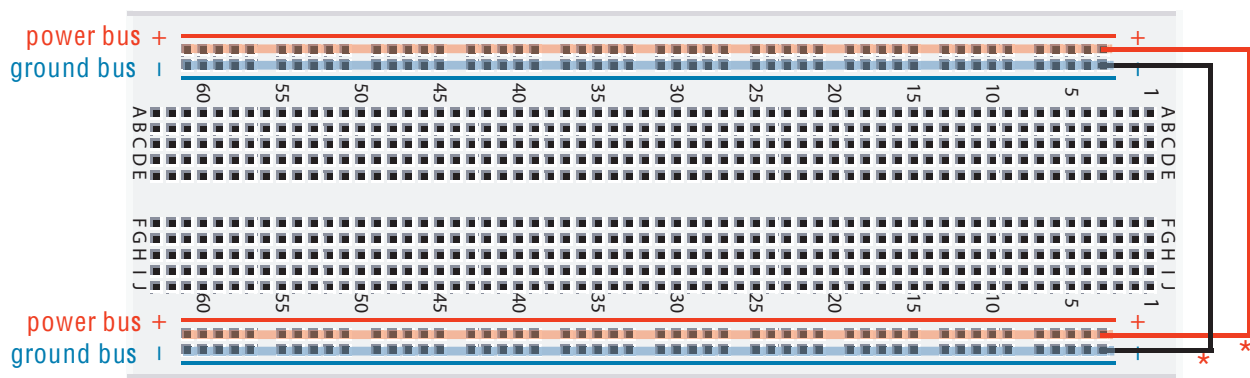
Note: In the following step-by-step red asterisks (*) denote the new connections that are to be made in a given step.

These instructions are best viewed on your computer screen using software

designed for viewing PDF format documents, such as Adobe Reader. The colors of the parts and wires will be readily visible and you can use Reader’s “zoom feature” to get a close-up view of the area you are wiring.

Step 1 - Connect the power and ground busses

Make power and ground “busses” as shown in the figures below.¹ The outer long rows of the breadboard will be connected to the +4.5V terminal of the battery (the LogoChip will operate over voltage range from +2.5 V to + 4.5V) and will form the “power bus” while the inner long rows will be connected to the negative terminal of the battery and will form the “ground bus”.



step 1

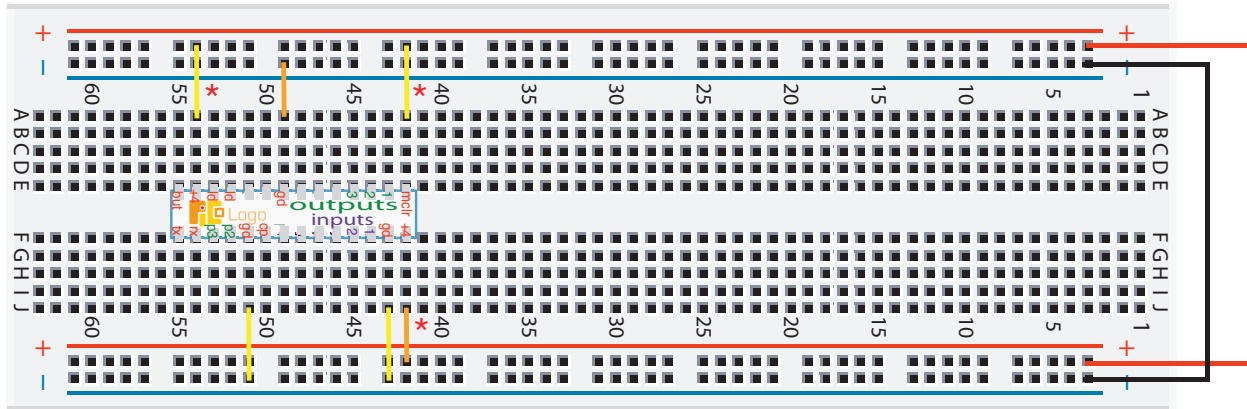
Step 2 - Add the LogoChip

Carefully position the LogoChip, as shown in the figure below. The left side of the should be plugged into row 55 of the breadboard.

¹ The name “bus” is meant to evoke the image of a public bus line, providing riders with frequent access points.

Step 4 - Connect the LogoChip to the power bus

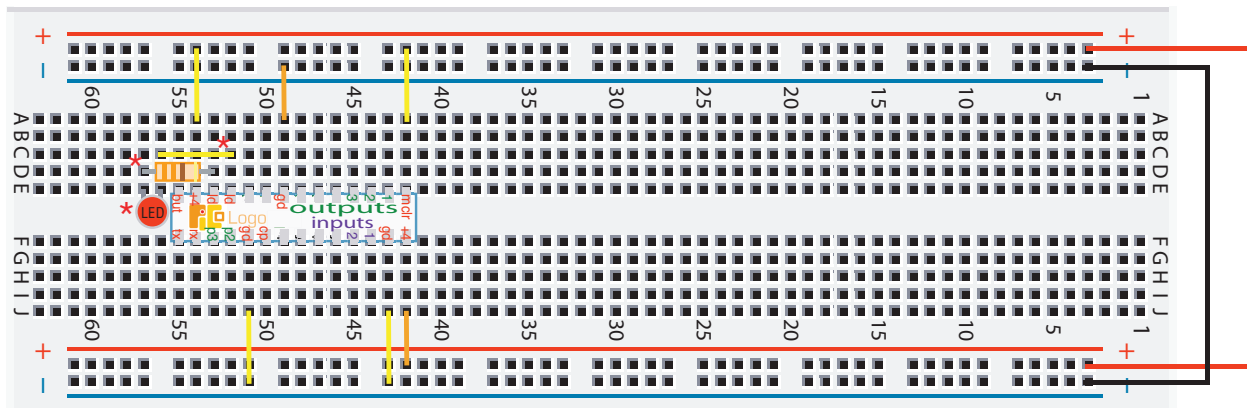
Three LogoChip pins (two are labeled **+4** and the third is labeled **mclr** on the sticker) must be connected to the power bus:



Step 4

Step 5 - Connect the LED

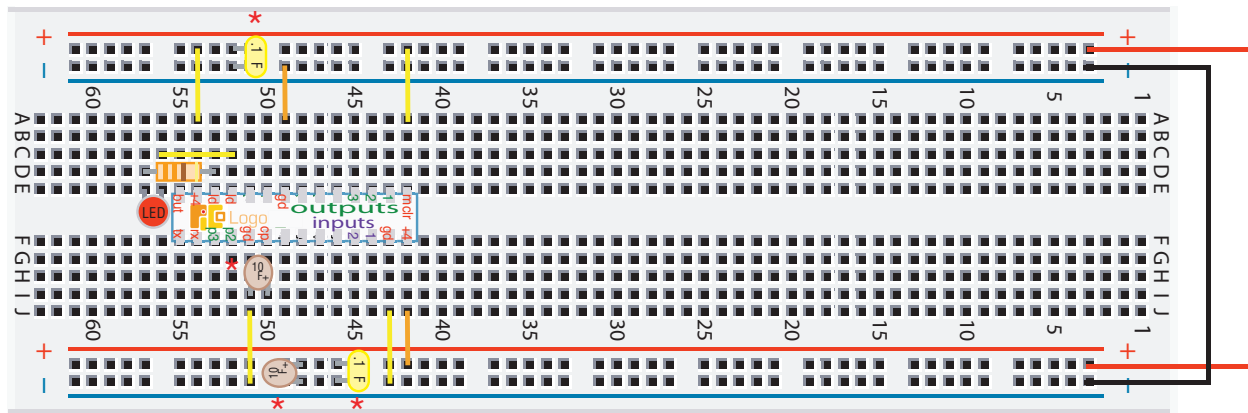
Add the red/green indicator light emitting diode (LED) and the 330 Ω resistor, as shown in the figure below. (The resistor and LED are wired “in series” between the two pins labeled **ld** on the LogoChip.) The LED comes with one lead shorter than the other. The shorter lead should be the one connected closest to the LogoChip. We suggest that you trim the leads on the LED and resistor with a wire cutter so that when you plug in the parts in they sit close to the surface of the breadboard.



Step 5

Step 7 - Add the capacitors

Add the two 10 μF capacitors and the two 0.1 μF “bypass” capacitors, as shown. **The polarity of the 10 μF capacitors matters.** The little “+” symbol on the 10 μF capacitors must go in the orientation shown in the figure. The polarity of 0.1 μF capacitors doesn’t matter; you can plug them in either way.

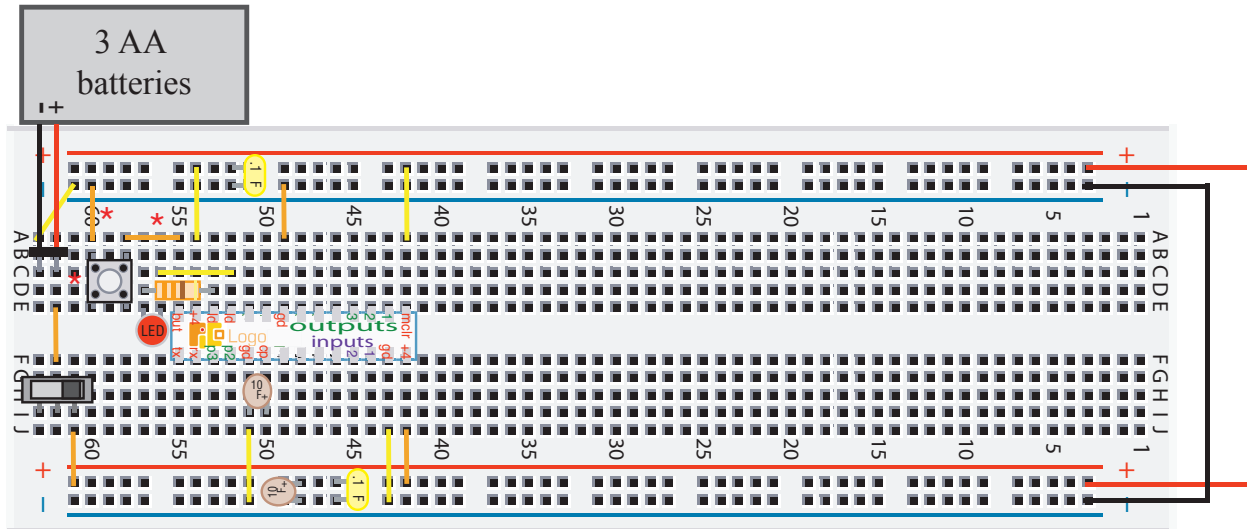


Step 7

The purpose of adding the capacitors is to “keep the busses quiet”; that is to try to help the battery maintain a constant voltage difference between the power and ground busses.

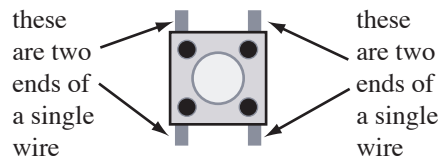
Step 9 - Add the start/stop button

Connect the start/stop button to the pin labeled **but** on the LogoChip, as shown in the figure below. Pressing this button will allow you to start and stop PicoBlocks programs running on the LogoChip.



Step 9

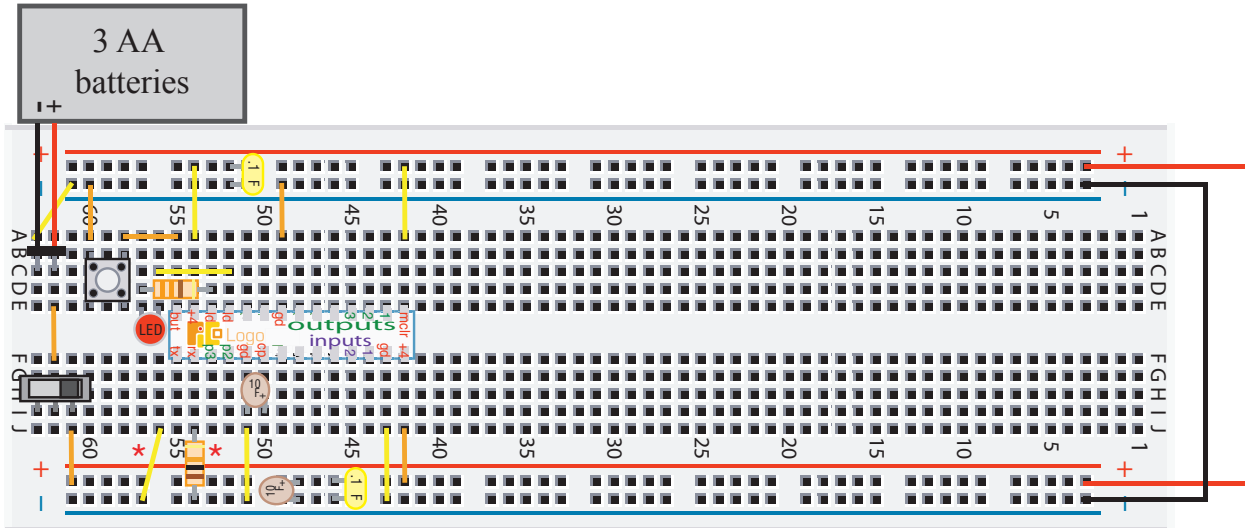
Please note: A common mistake is to get the orientation of the start/stop button wrong by 90 degrees. The figure below shows the proper orientation for inserting the button in step 9. Note that the pushbutton consists of two vertically oriented wires. The wires are brought into electrical contact whenever the button is pressed. This means that the orientation shown below is the one in which the two ends of a given wire are plugged into breadboard holes that lie in the same column. This insures that the two distinct wires that make up the pushbutton will remain unconnected until the pushbutton is pressed.



If you have a voltmeter you can easily perform a test to see if you connected the button circuit properly. Connect the voltmeter to measure the voltage between the “button pin” on the LogoChip and the ground bus. With the LogoChip power on, pressing the button should change the voltage from about 4.5 volts to 0 volts.

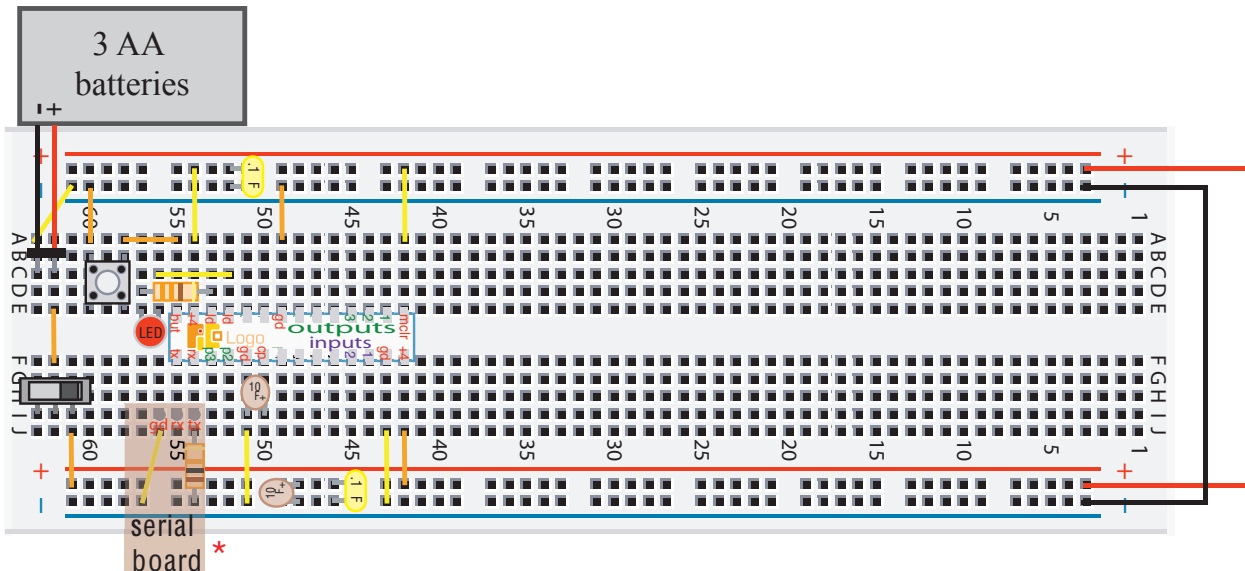
Step 10 - Add the serial connection

The LogoChip communicates with a host (desktop or laptop) computer via a serial connection. The LogoChip comes with a **serial interface board** and a **USB cable** that can be used to connect your LogoChip to the serial port of a host computer. The first step in making the serial connection is to install the 10 k Ω resistor and the ground connection in the locations shown below:



Step 10

Now plug the serial interface board into the position shown:

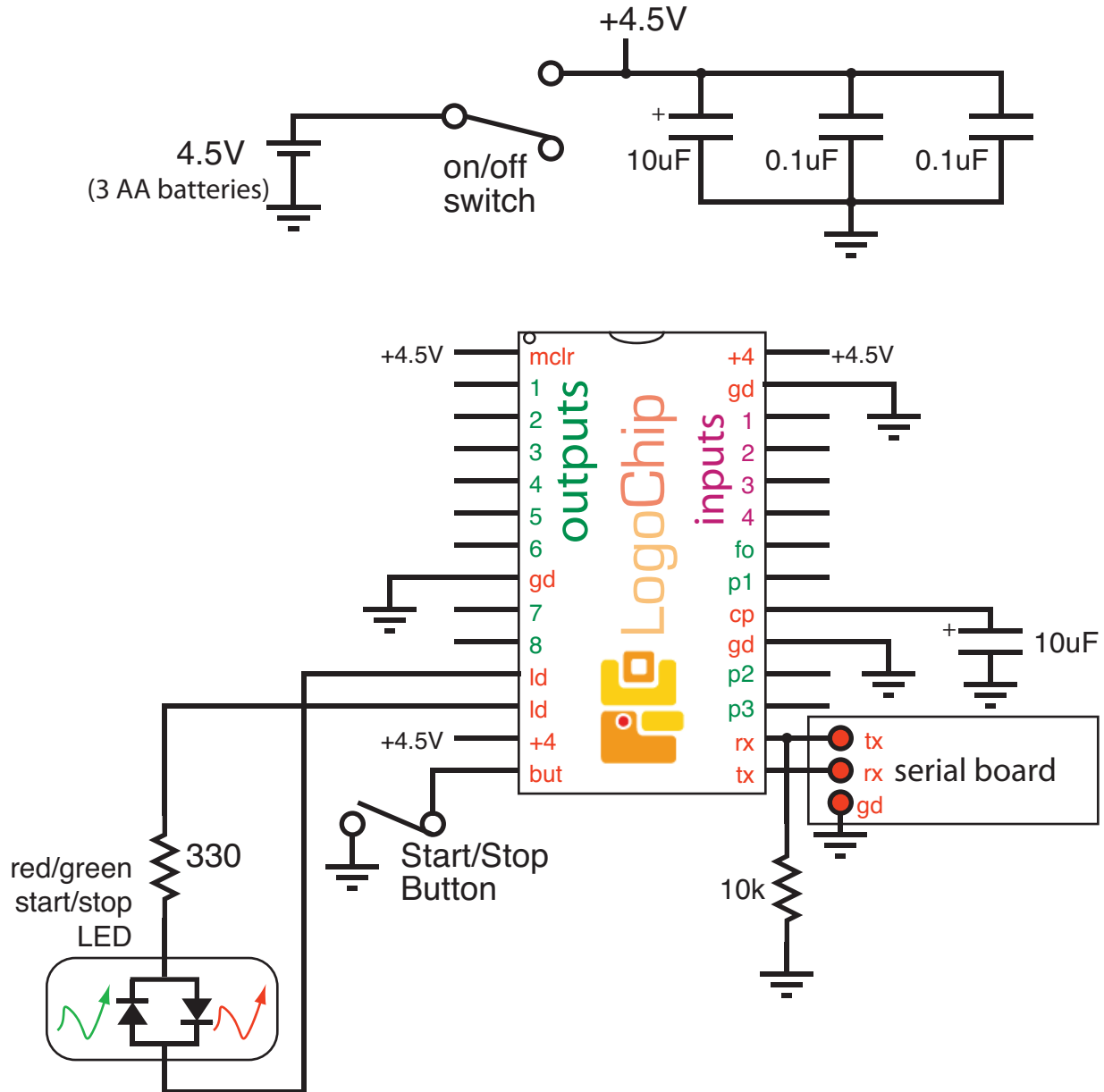


The completed LogoChip circuit

That's it! You're done building the basic LogoChip circuit. Now it's time to connect the LogoChip to your computer so that you can program it.

LogoChip Schematic Drawing

The figure below shows a **schematic drawing** of the LogoChip circuit you've just built. This drawing summarizes all of the external parts and power connections that need to be wired to the LogoChip.



PicoBlocks for LogoChip

You can compose programs for your LogoChip on a desktop or laptop computer and then download these programs to the LogoChip through the computer's serial port. The programming language we will use is a special version of the Logo programming language called PicoBlocks.

Install PicoBlocks for LogoChip

1) Download and Unzip the alpha version of the desktop LogoChip software from the following URL:

<http://www.wellesley.edu/Physics/Rberg/logochip/v3/alpha>

2) Check Java. The desktop PicoBlocks for LogoChip software runs on machines running either the Macintosh OS X or Microsoft Windows operating systems. It is a Java application, so you must have a recent version of Java installed on your machine.

Windows users: You must be running Java version 1.6.0 or later. To check the java version on your computer, open the Command Prompt (Windows). Type

```
java -version
```

If Java is not yet installed on your machine, you can download the latest Java version from :

<http://java.sun.com/javase/downloads>

(Select the JRE version.)

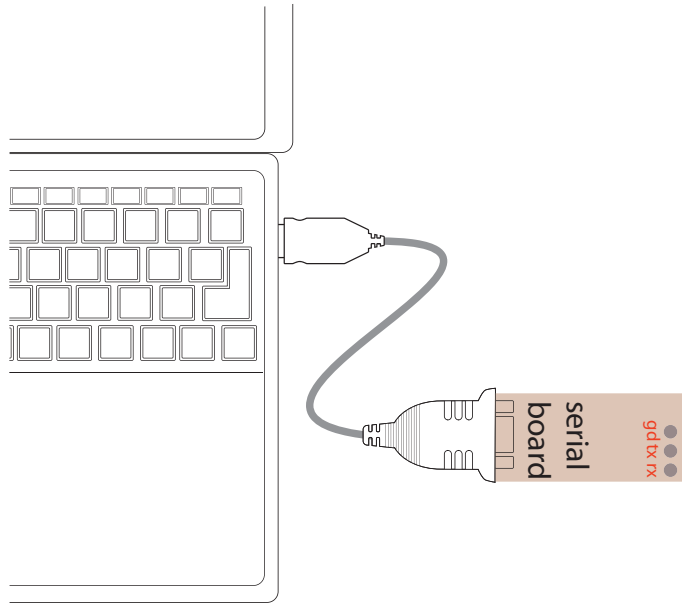
Mac OS X users: Java comes installed with Mac OS X. **Mac OS X will also have to install a Java Serial Communications driver**, a copy of which is available at:

<http://www.wellesley.edu/Physics/Rberg/logochip/distribution/macjavacomm/JavaCommInstaller.bin>

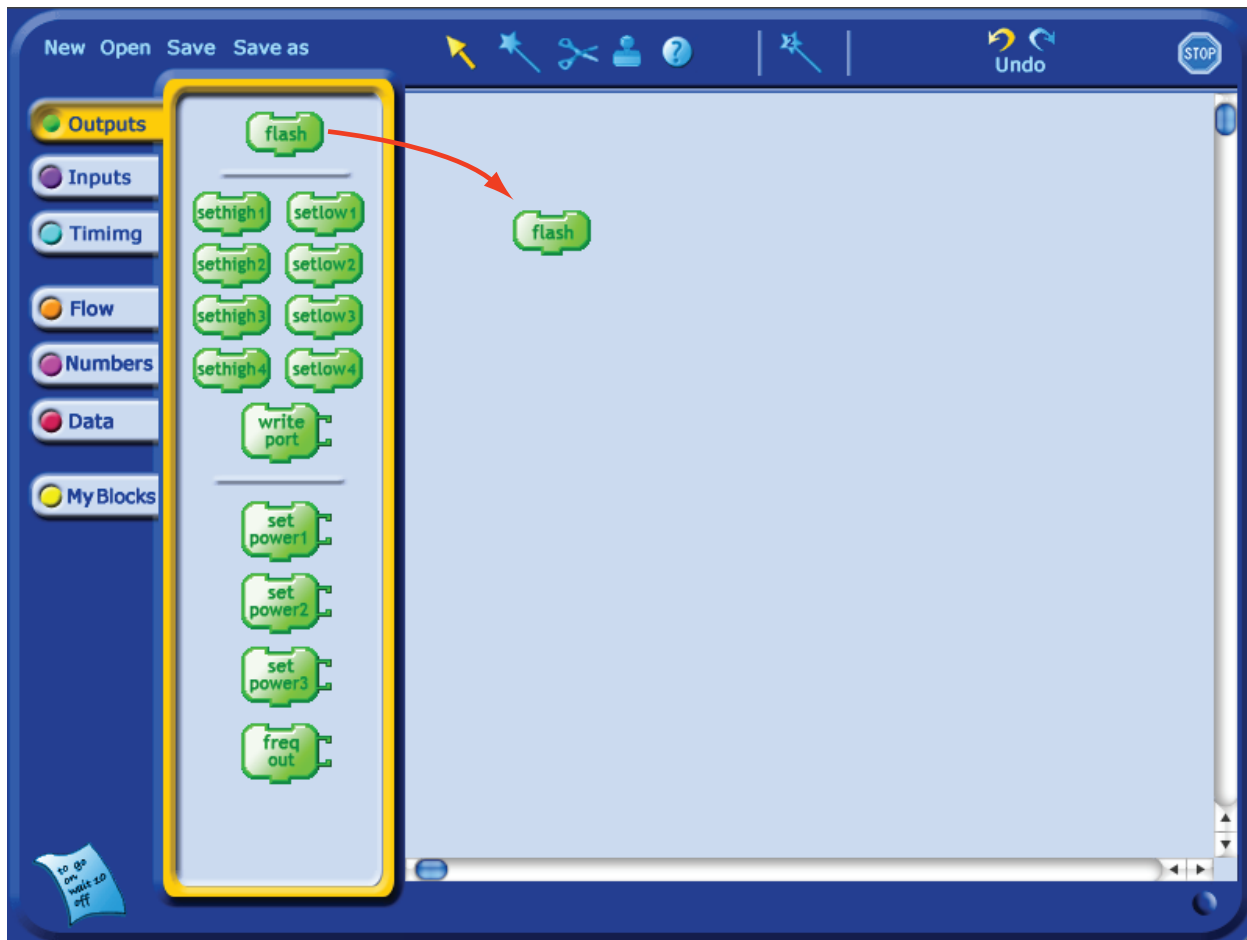
3) Install the drivers for the USB serial cable. Follow steps 1 and 2 at the following URL:

<http://www.picocricket.com/picoboardsetup.html>

2) Attach the serial cable serial board to your computer:



5) Open PicoBlocks for LogoChip by double-clicking on the file called `PicoBlocks.jar`. Make sure the LogoChip is powered up (LED is red) and is connected to your computer via the serial cable. Now, drag out a `flash` block and double-click on it:



The red/green indicator LED on the LogoChip board should flash just as it did upon powering up.

Troubleshooting

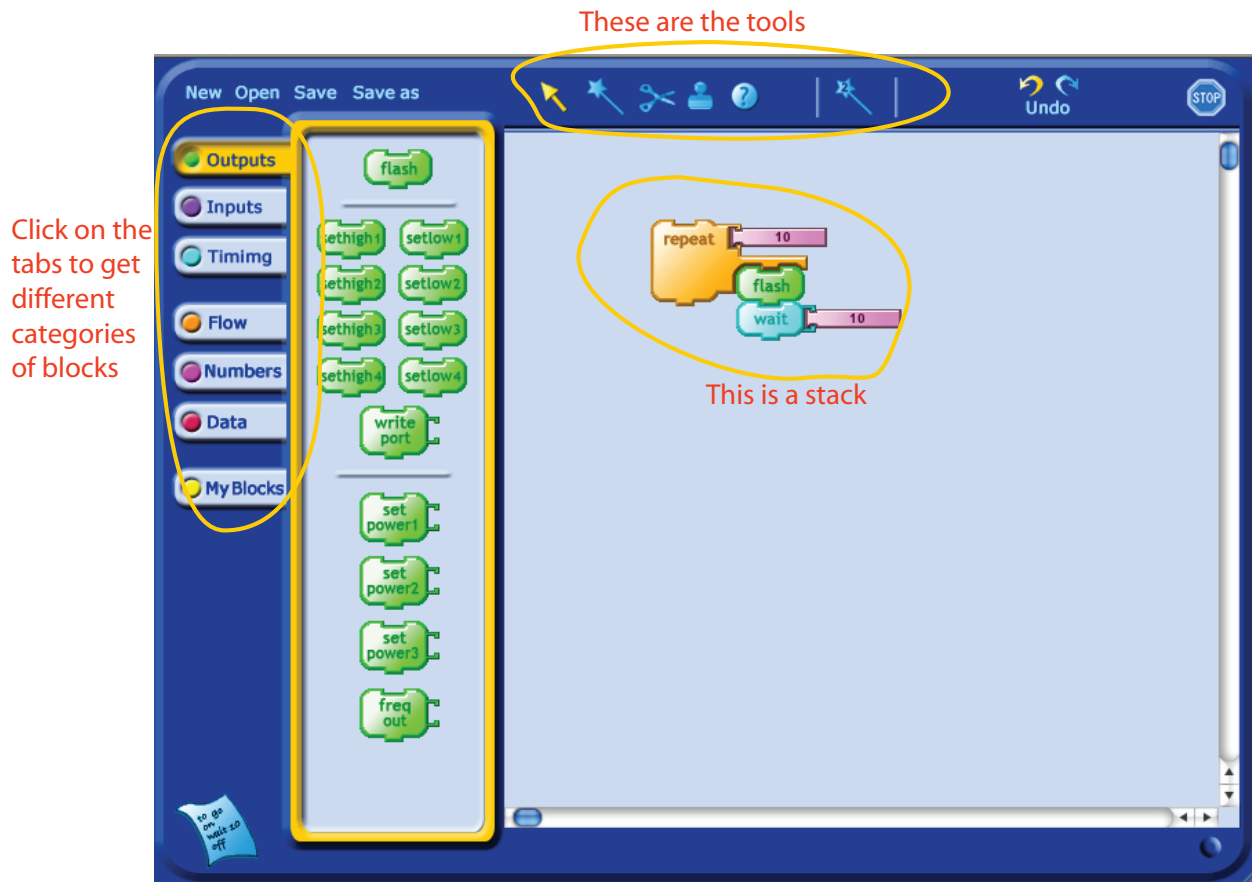
If you get a “Communications Error” message when you double-click on the `flash` block, a number of things could be wrong. Make sure that the LogoChip is powered up; the start/stop LED should be red. On Windows machines, look at the taskbar to make sure that there is only one version of PicoBlocks running. Also check to make sure that no other programs are competing for use of the serial port.

Are the batteries low? If your LogoChip is acting strangely (or not working at

all), it could be your batteries are running low. You should check the batteries with a battery tester.

Programming your LogoChip with PicoBlocks

With PicoBlocks, you create programs by snapping graphical blocks together into stacks.





This section contains some basic information that you need to start programming your LogoChip using PicoBlocks. More details of the PicoBlocks language are found in the *PicoBlocks Language Reference* section at the end of this document.

Tools

The tools are used for taking different actions on the blocks, such as copying blocks, deleting blocks, or sending blocks to the Cricket. When you click on a tool, the cursor turns into that tool.




 Use the **Arrow** to drag blocks from the Blocks Palette to the Workspace, or within the Workspace. To move an entire stack, drag from the top block. Dragging a block from the middle of a stack will also move any blocks attached beneath the one you are dragging. You can get rid of blocks by dragging them off of the Workspace, in any direction. If you delete blocks by mistake, click Undo .



Use the **Wand** to run a program. When you click on a block or stack with the Wand, it sends the program to the LogoChip and tells the LogoChip to run the program. Another way to run a program is by double-clicking with the Arrow, instead of single-clicking with the Wand.



Use the **Scissors** to get rid of blocks. Clicking on a block in a stack will also cut all blocks attached below it. If you delete blocks by mistake, click Undo . You can also get rid of blocks by dragging them out of the Workspace.



Use the **Stamper** to copy blocks and stacks. Click once to copy, then move to a new location and click a second time to paste. When you click on a block in a stack with the Stamper, it also copies all blocks attached below it.



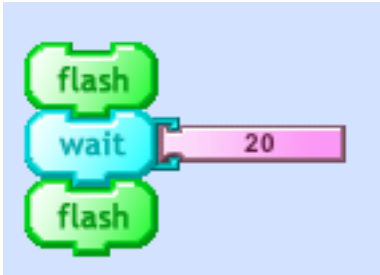
Use the **Help** tool to get more information. If you click on a block with the Help tool, a window with more information will appear.



Use the **Second Wand** to tell the LogoChip to run a second stack at the same time as it is running something else. For example, you can use the Second Wand to display sensor values while another program is running. To run two stacks, double-click on the first stack with the arrow, then click on the second stack with the Second Wand.

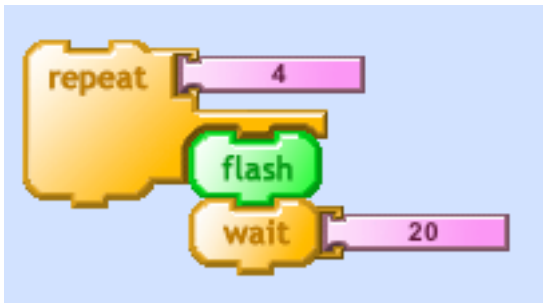
Getting Flashy

To get started, try running the following programs to make your *LogoChip*'s LED flash on and off in different ways:



Flash, waits two seconds, then flash again.

You can change how long the program waits between flashes by clicking on the number block with the arrow and then typing in a new number.)show how to change number



Repeats flash/wait 4 times.



Repeats flash/wait infinitely.

Press the *LogoChip*'s START/STOP button to stop the forever loop (or any PicoBlocks program) from running. You can tell if a program is running by looking at the indicator LED. "Green" means a program is running, "red" means the *LogoChip* is receiving power, but no program is running. If the indicator LED is off, it means the *LogoChip* is not receiving power.

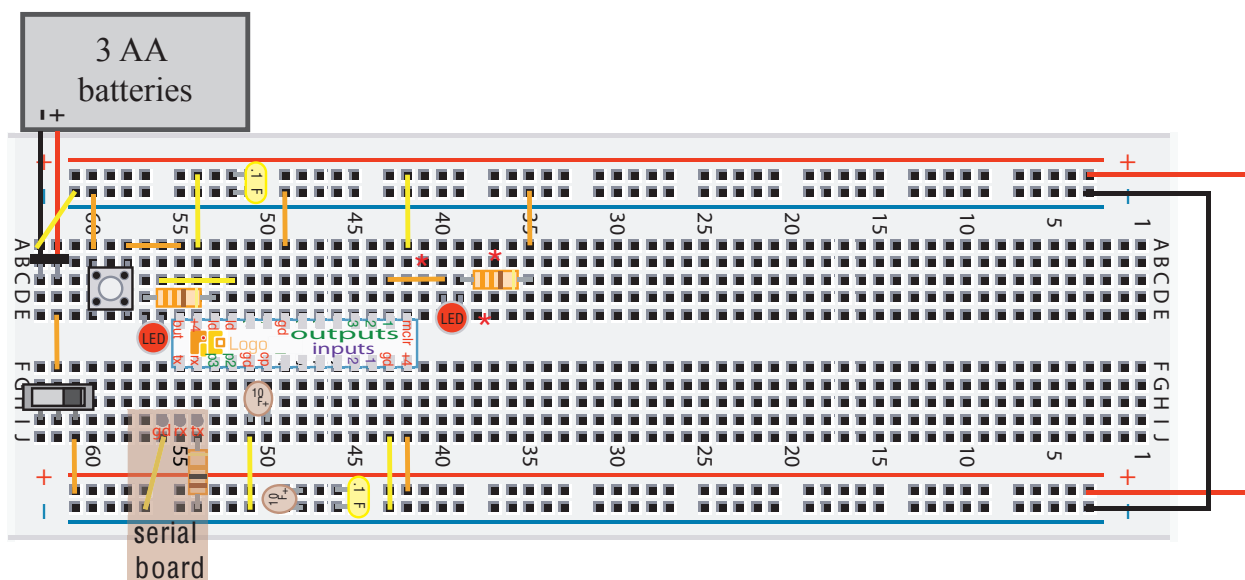
The *LogoChip* remembers the last program it ran, even after the power has been

turned off and the *LogoChip* has been disconnected from the computer. You can start this program by powering up the *LogoChip* and pressing the START/STOP button. This is a very useful feature, since it enables you to run your *LogoChip* programs without being connected to a computer.

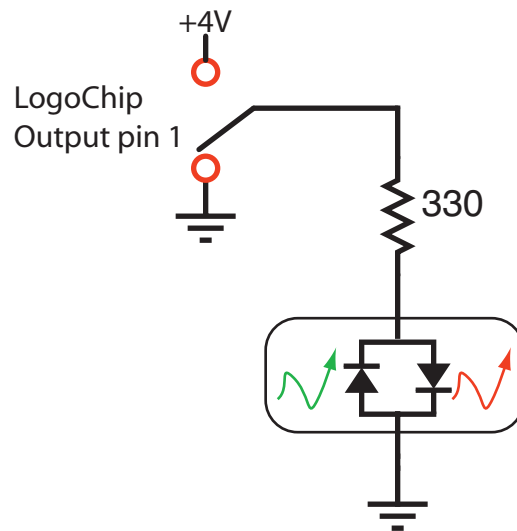
Hello World Projects

Here are some simple first projects to get started playing with your *LogoChip*.

Suppose a new red / green LED is connected to output pin 1 of the *LogoChip* as shown below:

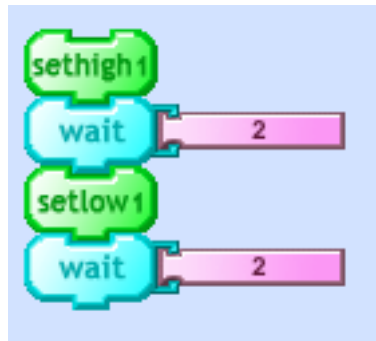


The red/green LED is simply a red LED and a green LED located in close proximity in a common housing and oriented in opposite directions, as indicated in the schematic drawing below:



In *PicoBlocks* there are commands that determine whether an output pin is “set high” (connected to +4 V) or “set low” (connected to ground, or 0 V).

Here is a *PicoBlocks* program that would make the LED flash once:



or forever:



Does your LED light up red or green? You can get it to turn the other color by reversing the LED’s orientation in your breadboard.

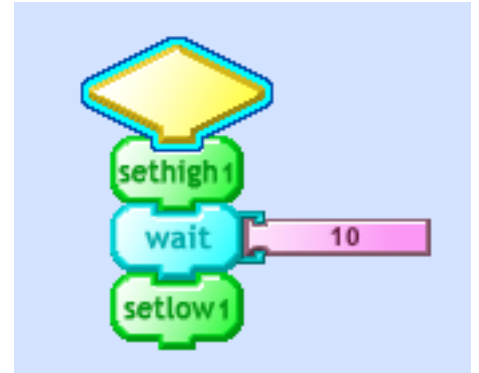
My Blocks



You can use this diamond shaped block to give a name to a stack of blocks - and create a new block that does the same thing as the

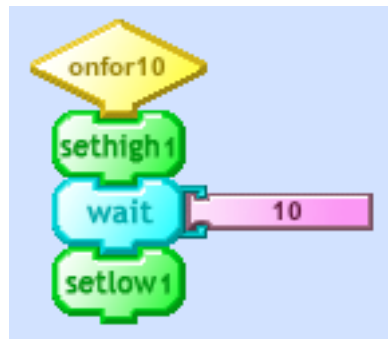
entire stack. Here's how:

1. Attach the diamond shaped block to the top of the stack you want to name.



2. Click and type a name:

(no spaces in name)



3. A new block with that name appears in



4. Use this new block in other stacks:





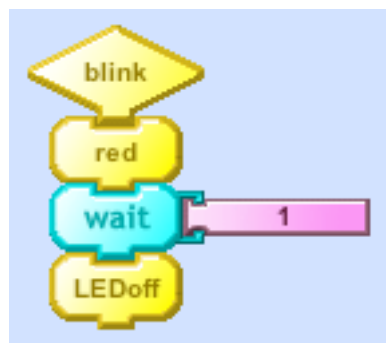
or the other:



You can turn the LED off:



or make it blink:



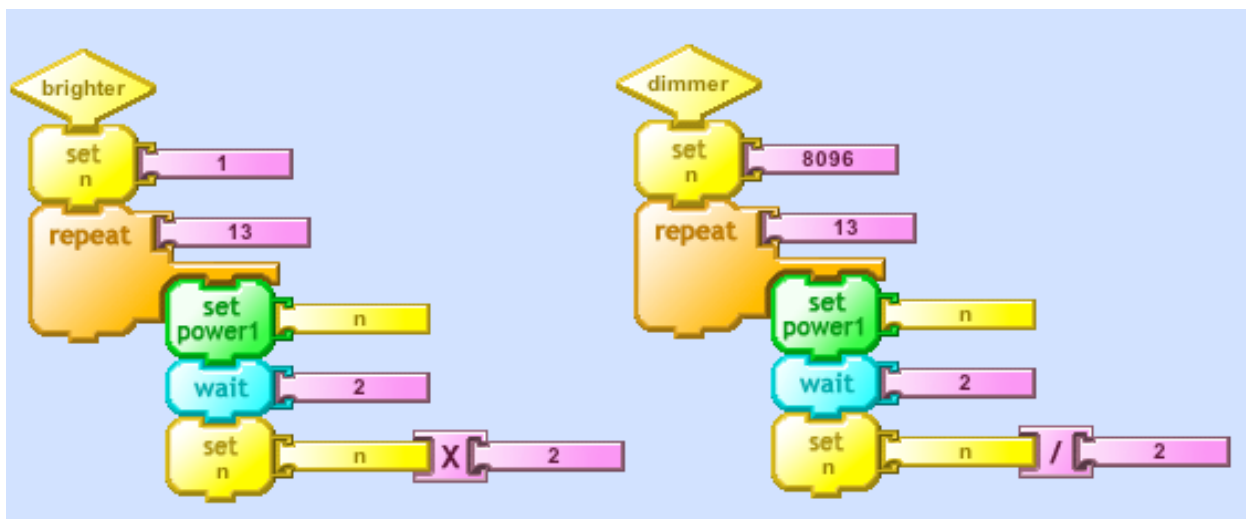
By rapidly switching between the red and green states, you can make the LED appear yellow:

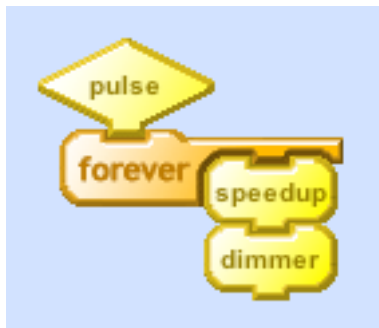


or some other color:



You changing the size of the number stored in on of the *LogoChip's* **variables**, you can make the LED get brighter or dimmer

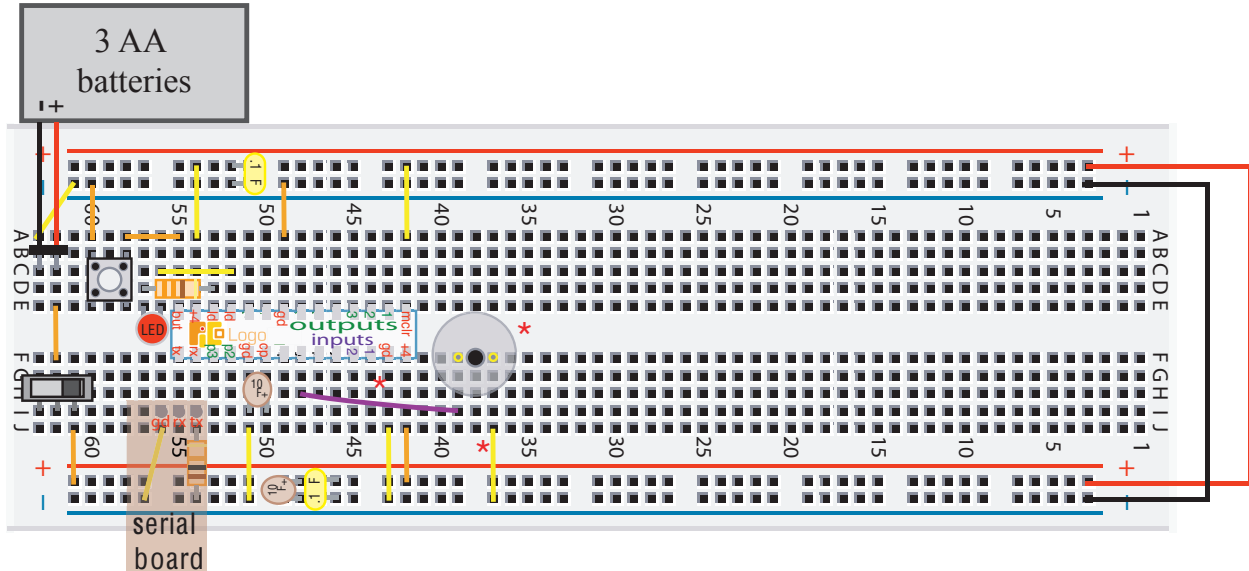




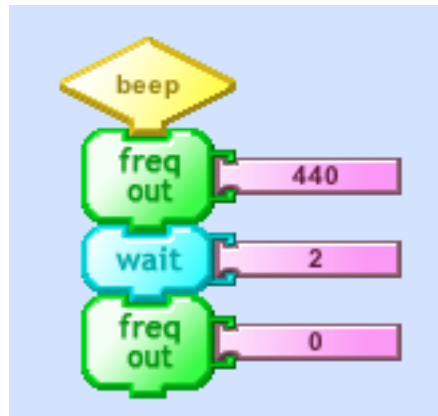


Making Music

Now connect the piezoelectric beeper to the freqout pin, **fo**, as shown below:



Here's a program that will make your beeper beep:

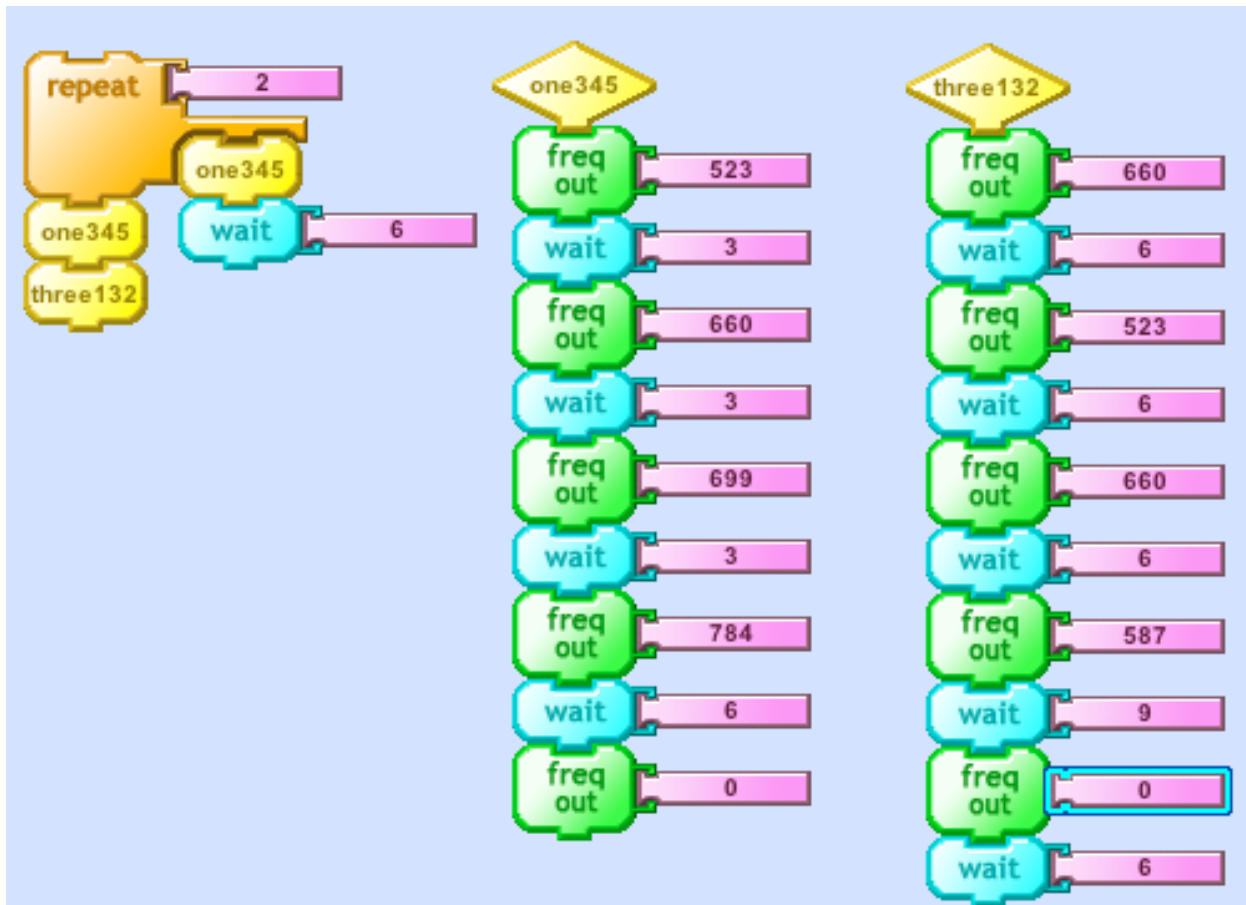
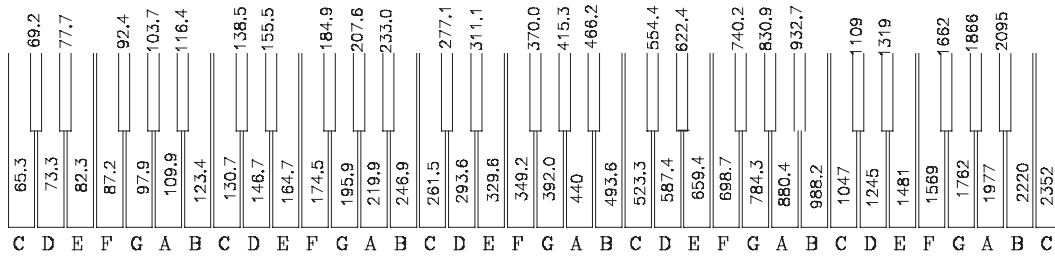


Play with the pitch and duration parameters in beep until you get a beep of your liking.

Can you compose a melody with the help of the keyboard shown below?



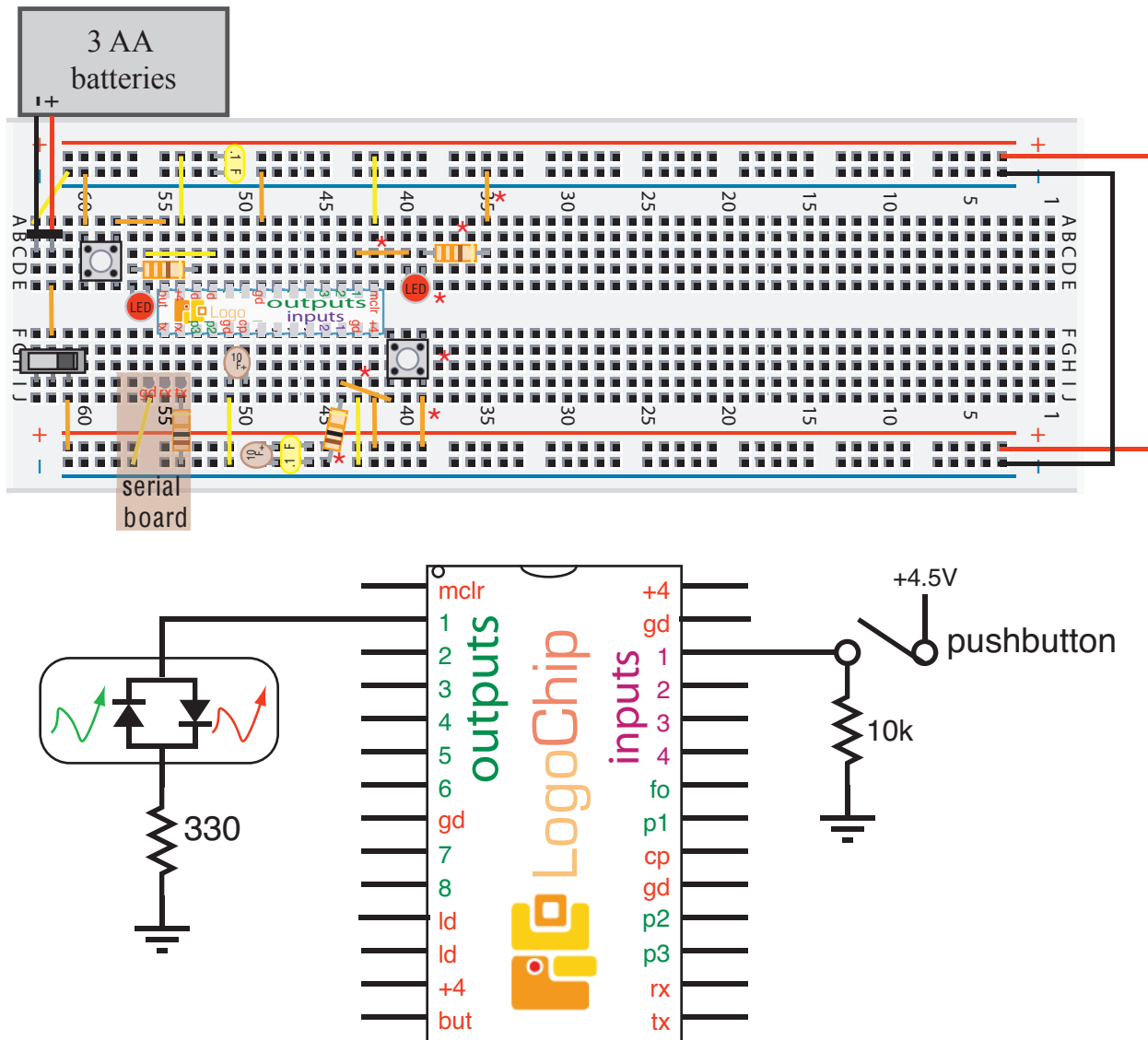
Keyboard Frequencies



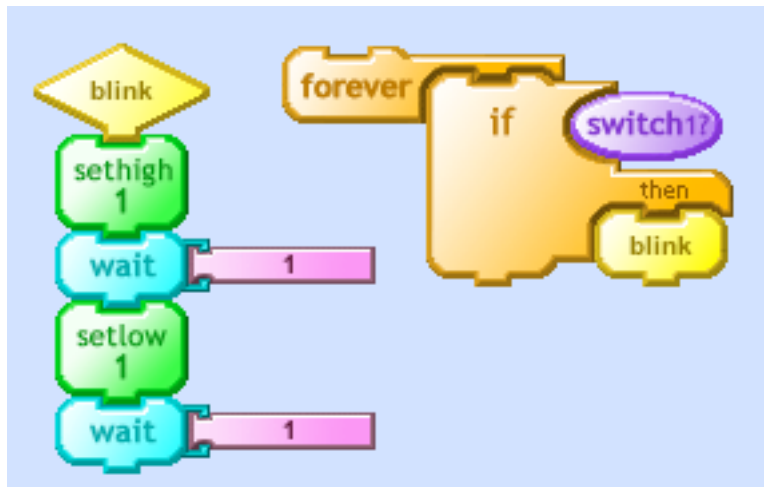
Getting a Sense of the World

Digital Sensors

Connect a pushbutton switch to the sensor 1 input pin, as shown in the figure below:



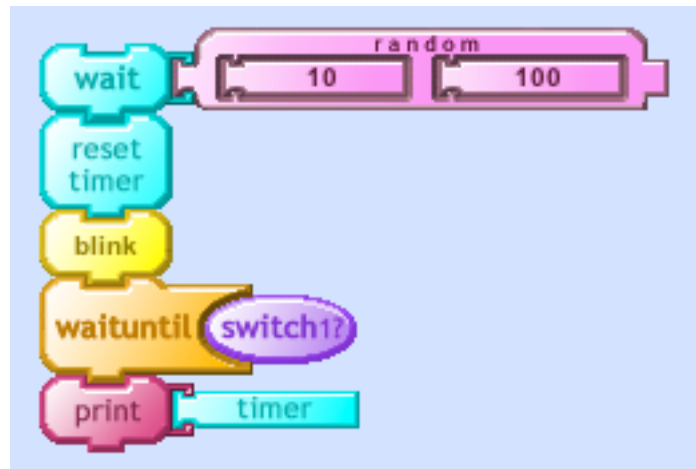
Now your *LogoChip* can sense when the pushbutton is pressed. For example, with the program below, an LED connected to output #1 will blink as long as you are pressing the pushbutton sensor.



You've built your first digital sensor!

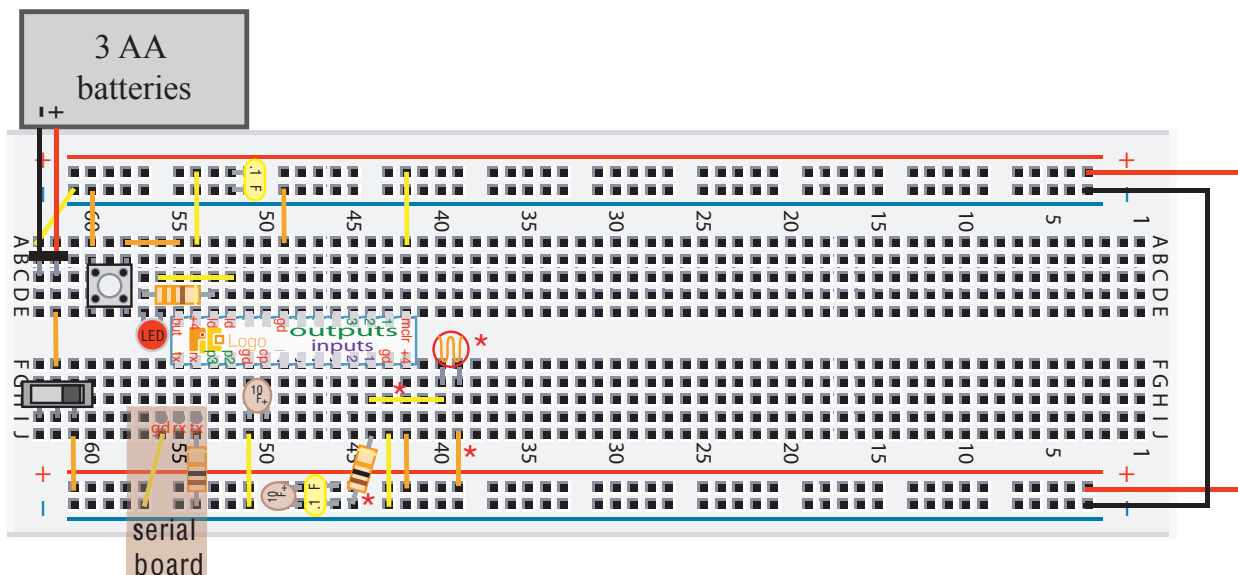
Reaction Game

The program below makes the LED flash on and off. Can you stop the program with the START/STOP button and consistently leave the LED in the on state?



Analog Sensors

Try replacing the pushbutton sensor with a **photocell**. A photocell has the property that its electrical resistance varies, depending on the brightness of the light striking its surface; the brighter the light the lower the resistance. The voltage measured at the sensor input pin will depend on how the photocell's resistance compares to the fixed 10 kΩ resistor, and hence on the brightness of the light.

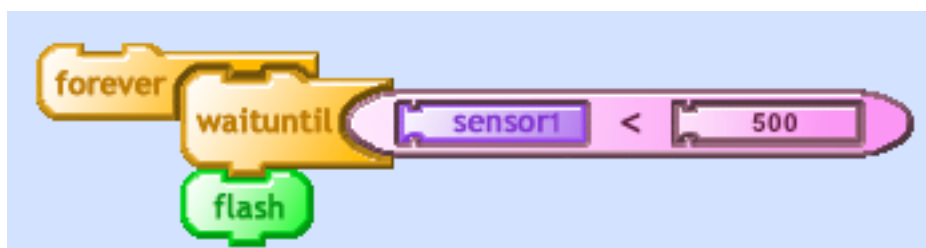


Try:



The `sensor1` block reports the value of the 10-bit analog to digital converter associated with input pin 1. The `print` block causes the result to be displayed in the message window at the bottom of the PicoBlocks screen. The brighter the light, the higher the reading you should get.

You can use your photocell as a “shadow detector”. Can you make the LED flash when you cast a shadow a on the photocell. (To get it to work, you may have to adjust the “threshold” by changing the number in the “<” block.)



Congratulations! You’ve built your first analog sensor. (Also included in your *LogoChip Kit* is a **thermistor**, an element whose electrical resistance depends on

temperature. Can you make a temperature sensor?)

Theremin

A “theremin” is a rather strange electronic musical instrument, invented by Leon Theremin nearly ninety years ago. You didn’t have to touch it to play it. You just wave your hand near a theremin and it makes weird sounds.

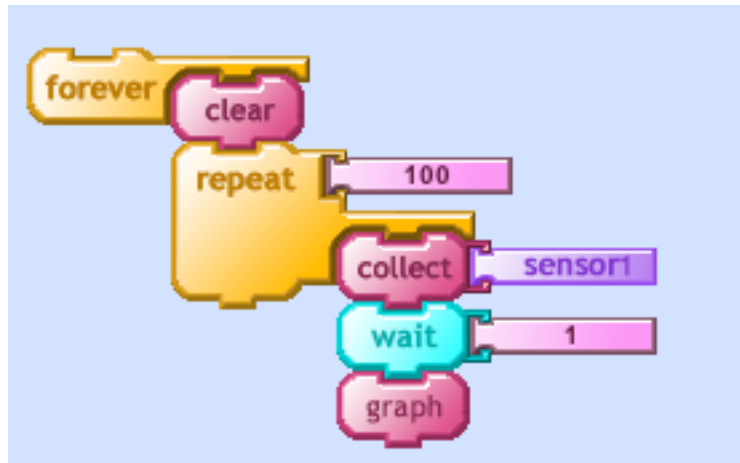
You can make your own theremin-like instrument. Just connect a beeper to the freqout pin and wave your hand over the light sensor, casting shadows:




Scoping out data

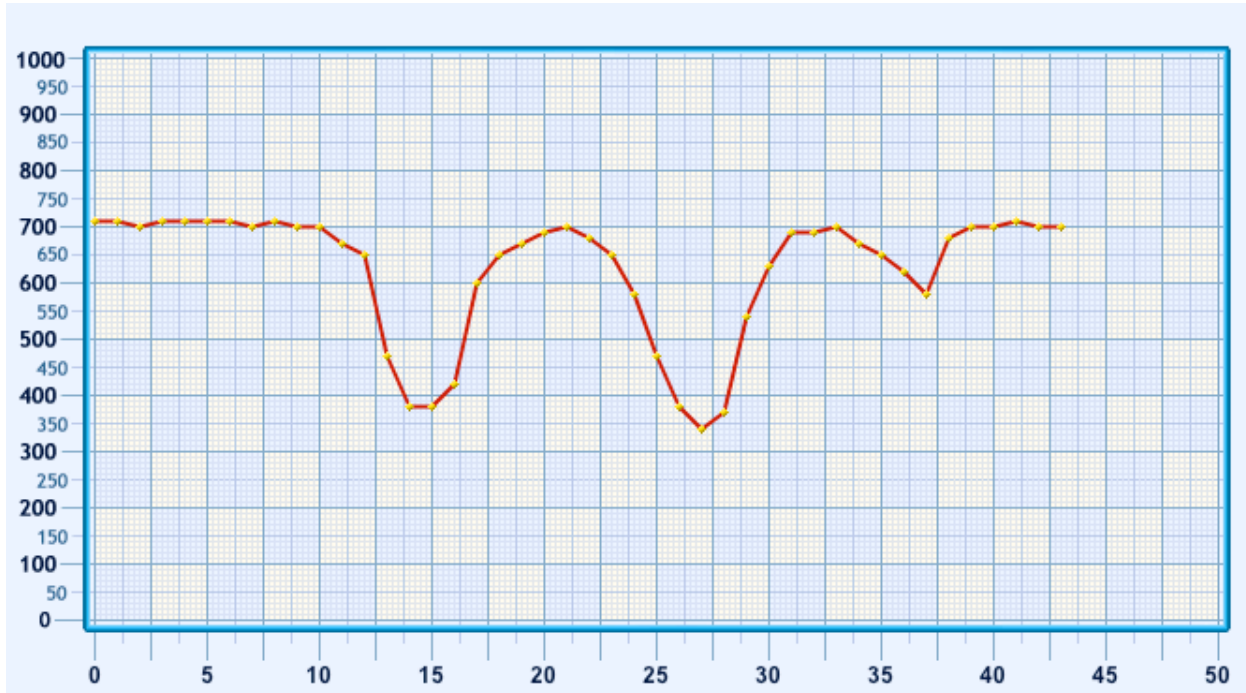
The PicoBlocks for LogoChip software let’s you collect data and then make graphs of your data. This makes it possible to use PicoBlocks as an “oscilloscope” to view how sensor readings are changing over time.

With your photocell connected as above try running the following program:

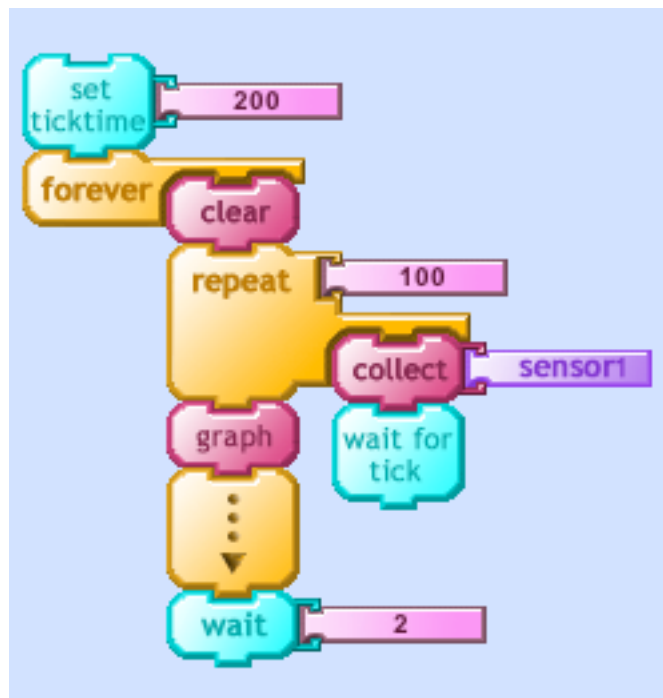


To see a real time graph of the data, click on the  button at the bottom of the data palette. You should be able to see the graph change as you cast

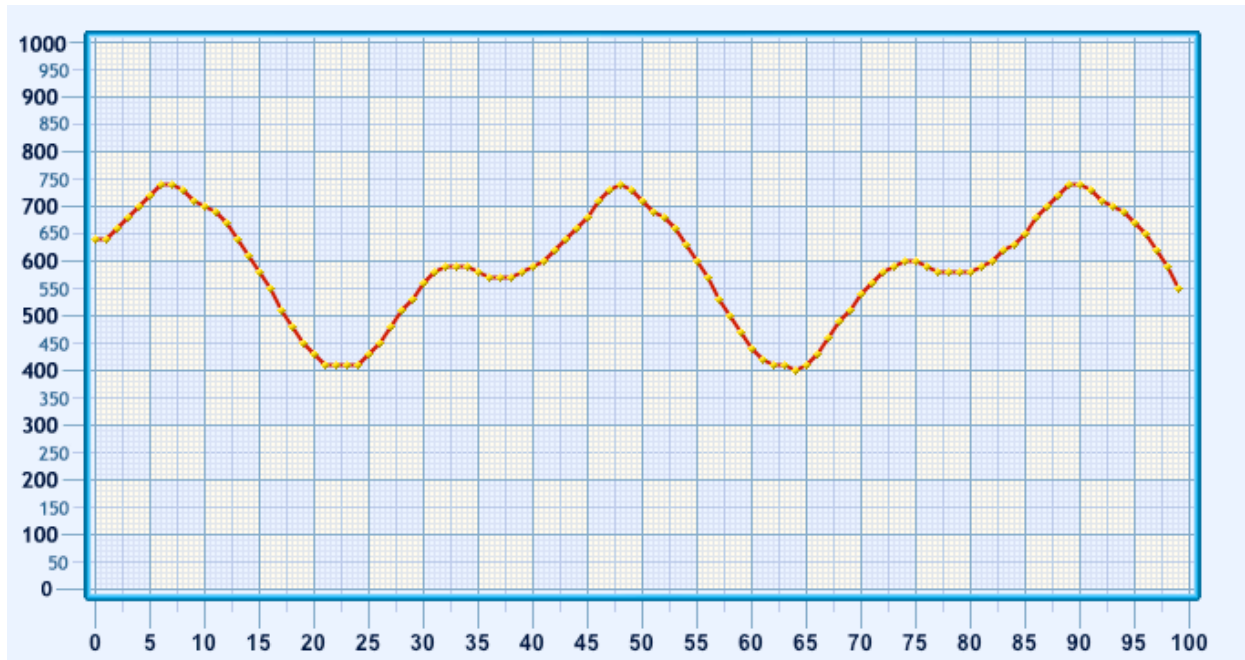
shadows on the photocell:



You can also use the scope to view more rapidly changing data. Try replacing the photocell with a phototransistor, which has a faster time response than a photocell. Change your program to:

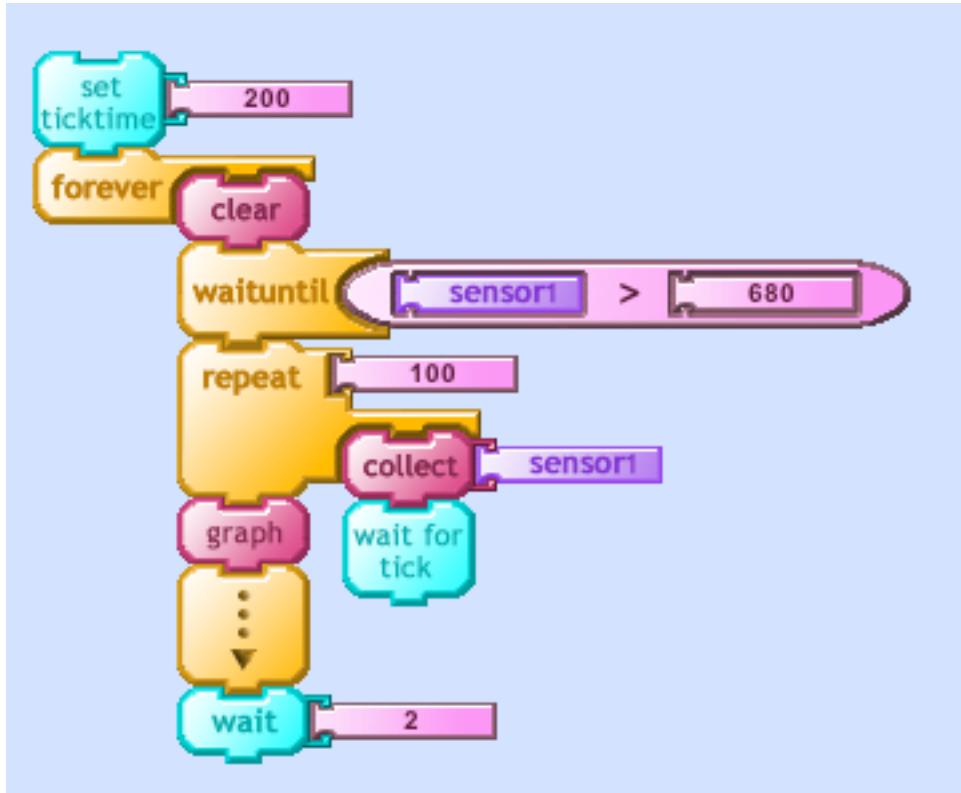


If you're in a room with fluorescent lights you may see something like this:

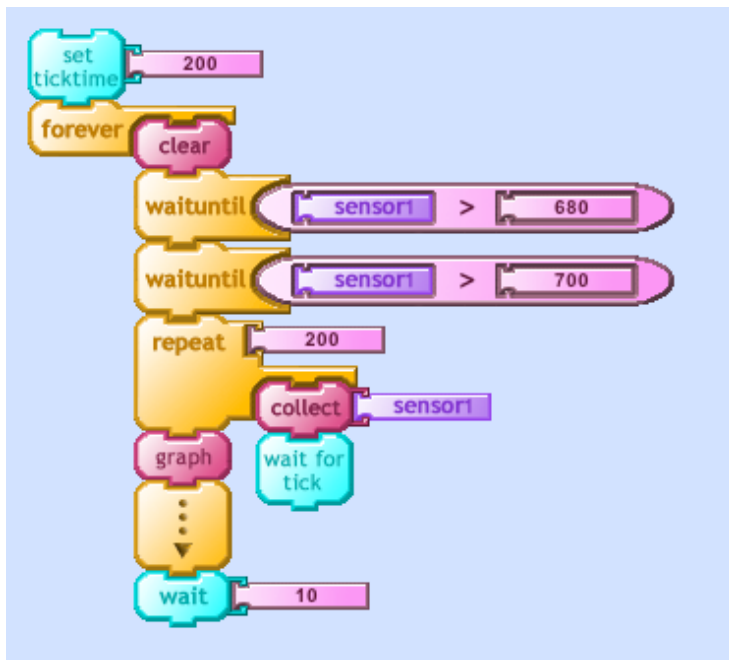


(Fluorescent lights exhibit a significant amount of “flicker”, varying their intensity 120 times per second. This rate is too fast for your eyes – or a photocell – to detect. But phototransistors can easily respond to light sources that are varying at this rate.)

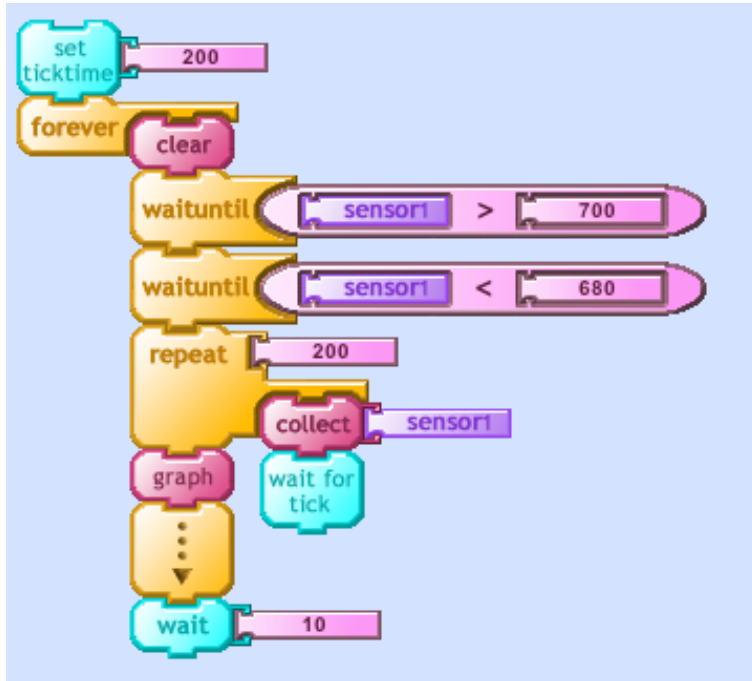
(You can add a “waituntil” block to “trigger” the scope at a particular level:



Trigger on positive slope:

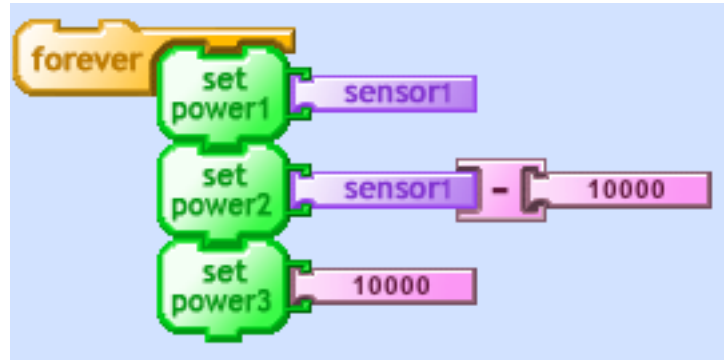


Trigger on negative slope:



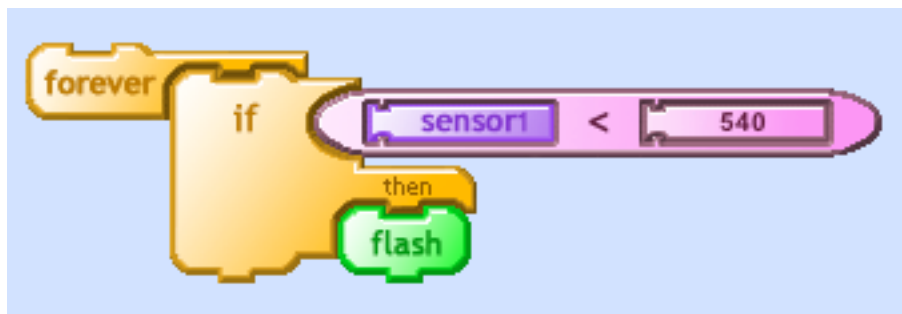
Try Colors

Connect a tri-color LED to the three power pins and you can make any color of the rainbow. You can control the color based on input from a sensor. For example:



Clap sensor

Wire up a microphone as shown below:



Let's Get Moving

Now add a small motor (such as a LEGO motor) between output pins 3 and 4. Try playing with the following new procedures:



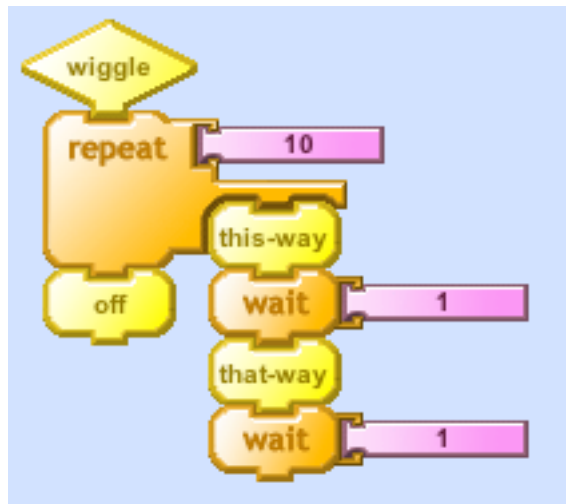
thisway Turns on motor in the “thisway” direction.



thatway Turns on motor in the “thatway” direction.



off Turns off the motor.

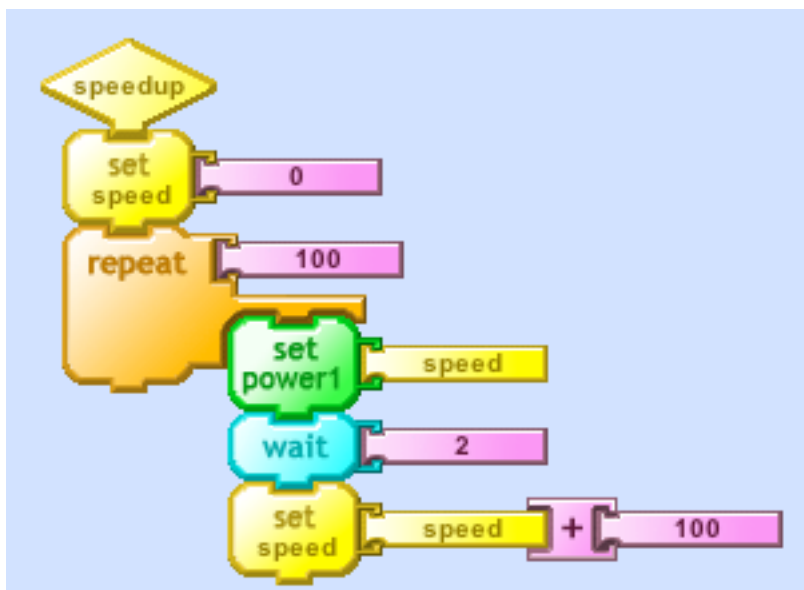


wiggle Makes the motor wiggle back and forth.

Now connect to the motor to power pin 1. You can turn the motor on at “full power” with the following program:



Choosing power settings less than 10,000 will deliver proportionately lower amounts of power to the motor. Here’s a program to make the motor speed slowly ramp up:





Toothbrush hack

Microphone

PicoBlocks Language Reference

Blocks Summary

Outputs



Make the LogoChip's LED flash red and green four times.



Turn on digital output #1. (Set its voltage to +4 volts.) **redraw figure**



Turn off digital output #1. (Set its voltage to 0 volts.) **redraw figure**

Similar blocks exist for digital outputs 2, 3, and 4. Digital outputs 5 - 8 can be controlled using `on` and `off` commands in the text language.

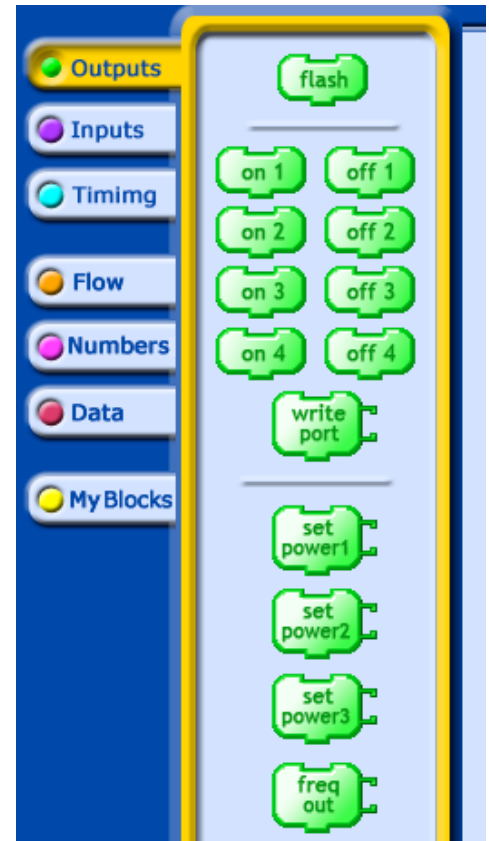


Sets the power delivered to pin `p1`. The argument of `setpower1` is any number between 0 and 10,000. (A square wave is set up on pin `p1` that is "on" for a fraction of time equal to the argument / 10,000.)

Similar blocks exist for controlling power pins `p2` and `p3`.



Sets up a square wave on pin `f0`. The argument of `freqout` determines the frequency of the square wave (in Hertz) and can be any number between 0 and 65,535.



Inputs



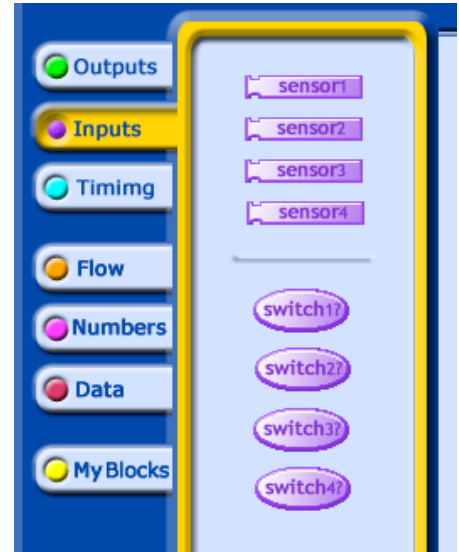
Reports a number from 0 to 1023 proportional to the voltage on input #1.

Similar blocks exist for inputs 2 - 4.



Reports "true" or "false" depending on whether the voltage on input #1 is HIGH or LOW.

Similar blocks exist for inputs 2 - 4.



Timing



Wait for a number of tenths of a second.



Wait for a number of *milliseconds*.



Reports the reading of the LogoChip's internal timer in *milliseconds*.



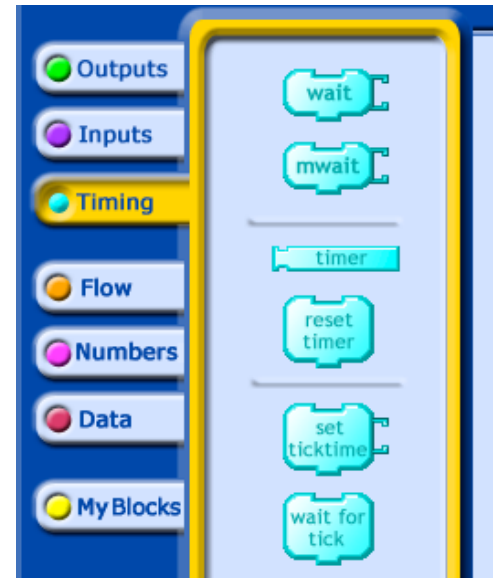
Resets the LogoChip's internal timer.



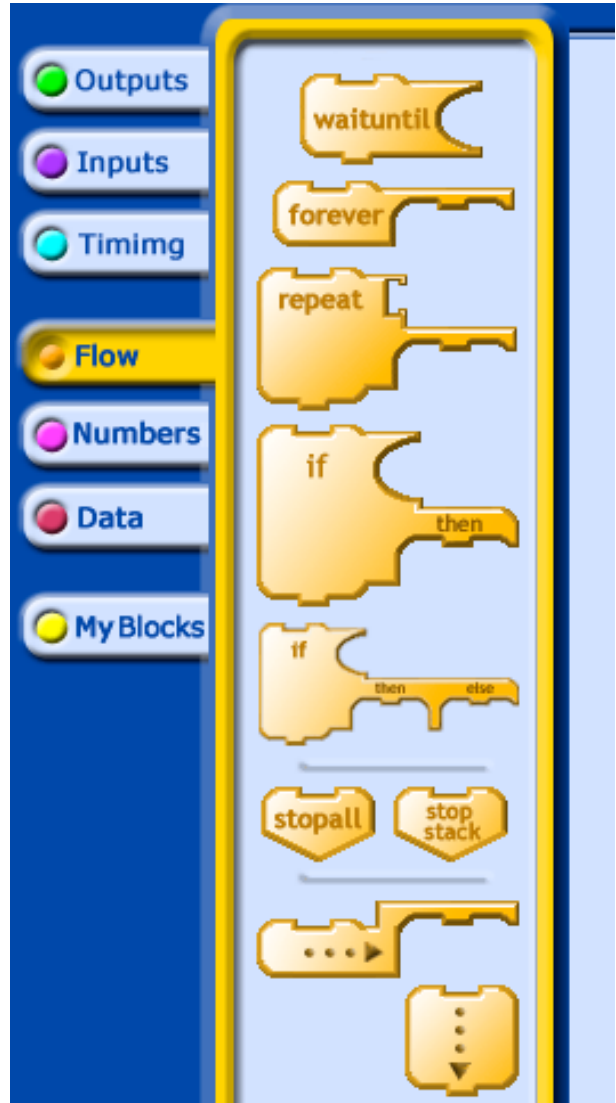
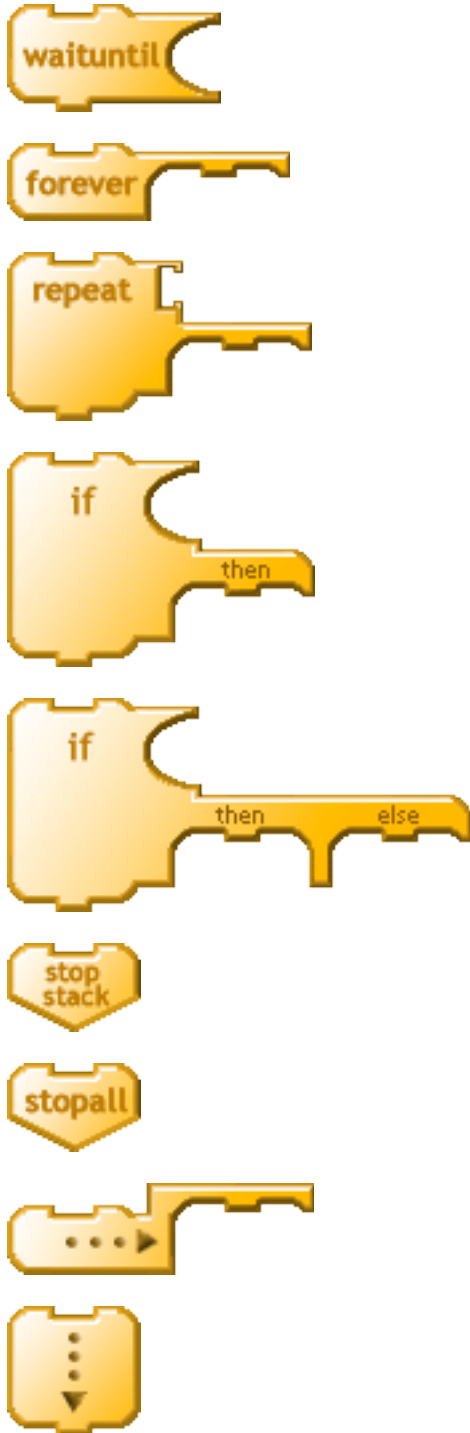
Sets the time in *microseconds* between ticks of the LogoChip's fast internal clock.



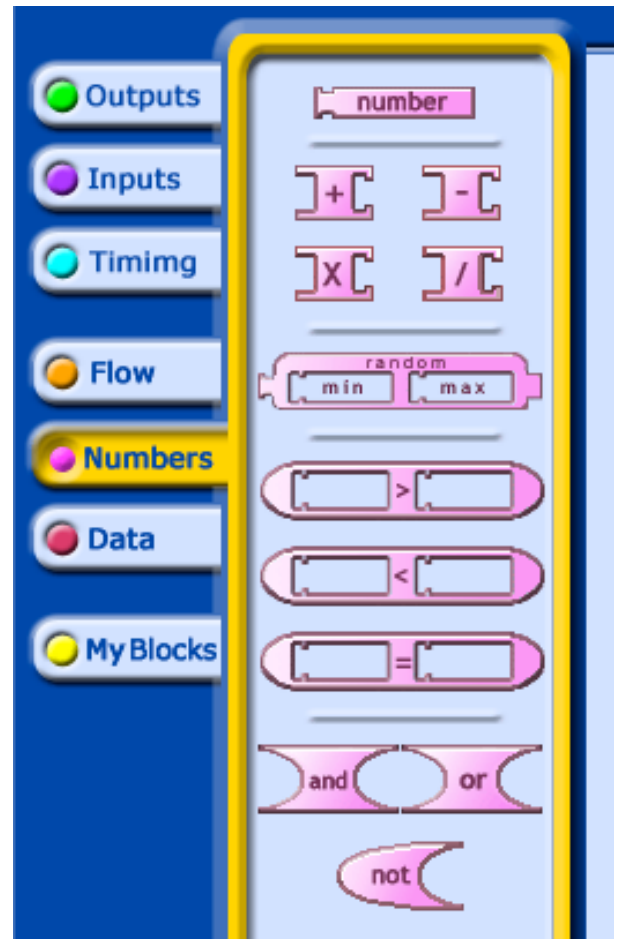
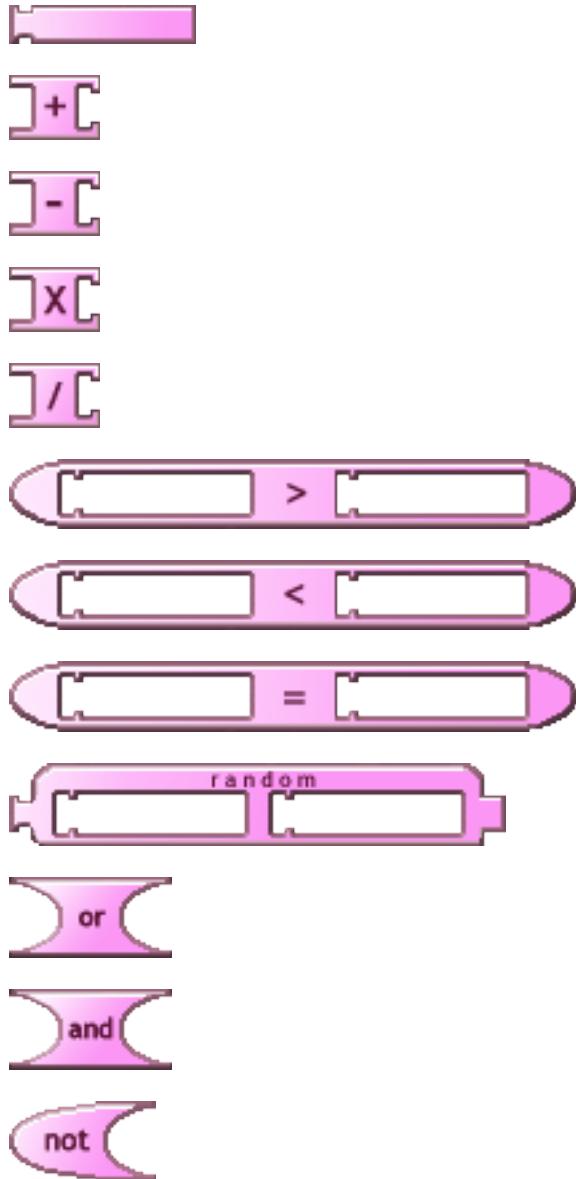
Wait for the next tick of the LogoChip's fast internal clock.



Flow



Numbers



Data



Erases all the data stored in the LogoChip's data buffer.



Stores a number at the next spot in the LogoChip's data buffer.



Goes back to the beginning of the LogoChip's data buffer.



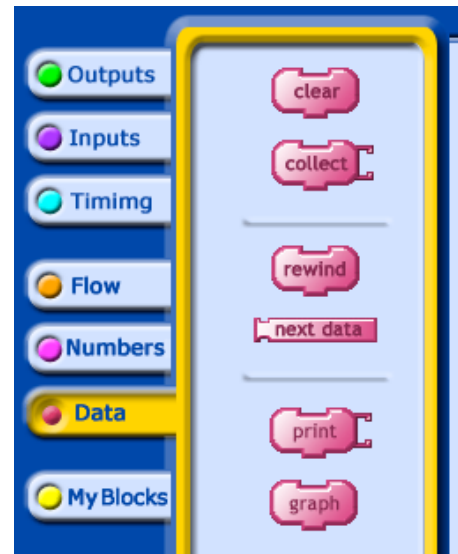
Reports the next number in the LogoChip's data buffer.



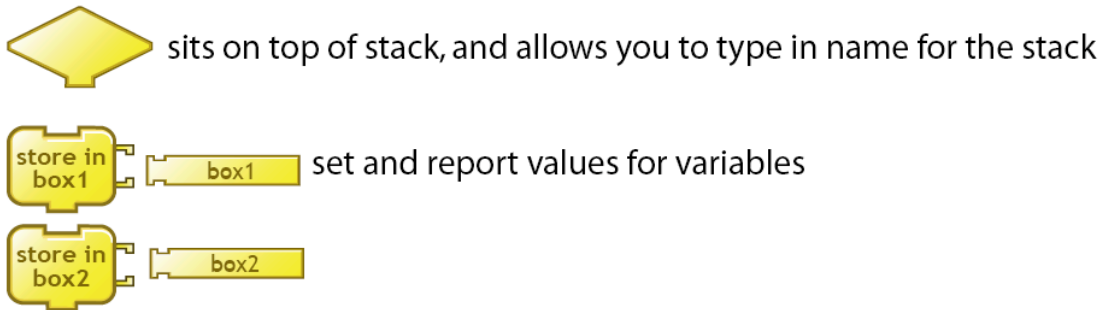
Prints a number in the message window at the bottom of the PicoBlocks screen.



Graphs the collected data on the PicoBlocks "scope".



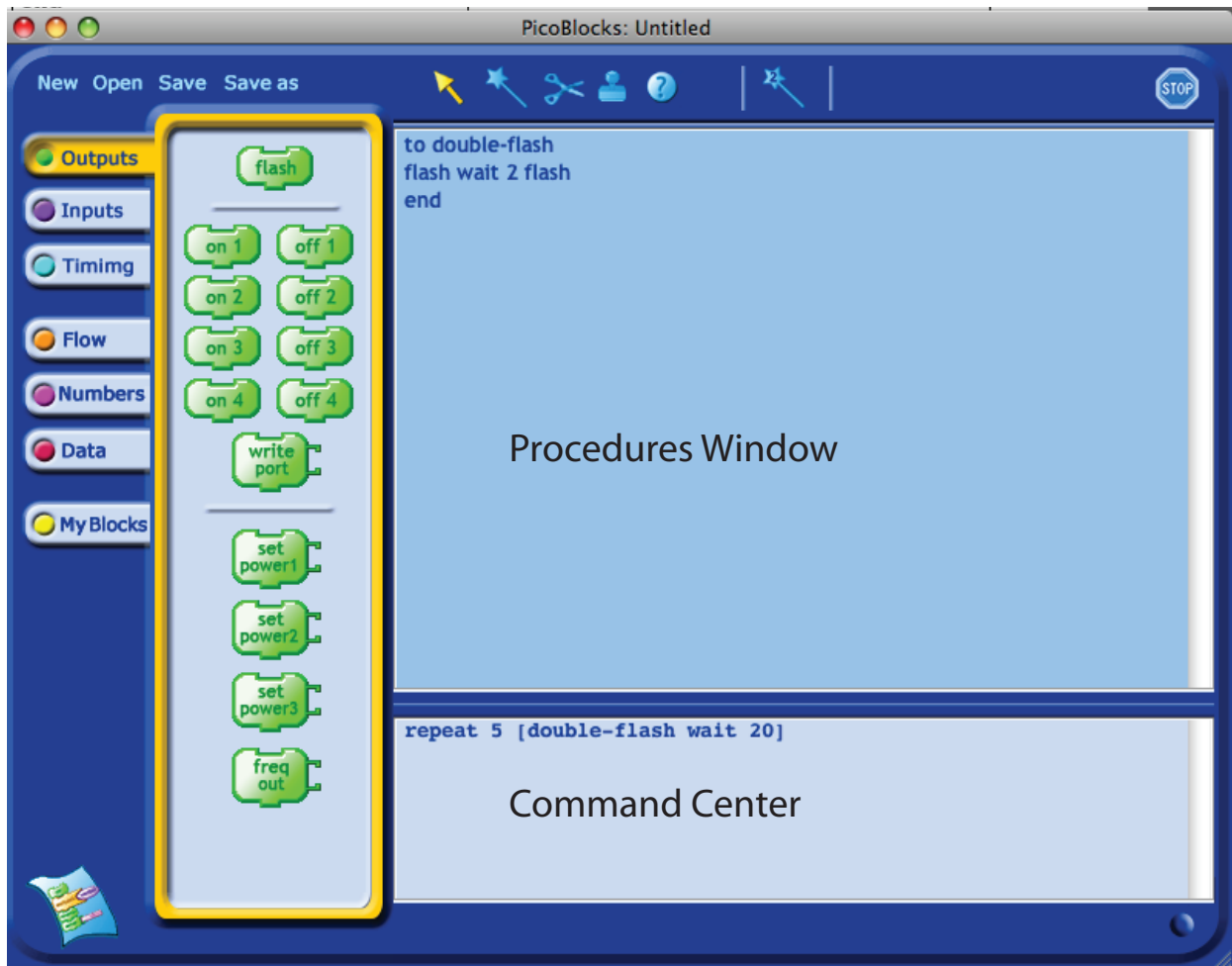
My Blocks



redraw figure

The PicoBlocks Text Language

PicoBlocks lets you construct computer programs by snapping different blocks together, like pieces of a puzzle. This approach makes it easy to get started; different kinds of blocks have different shapes and can only snap together in certain ways. The shapes of the pieces provide a lot of guidance as to how structure your programs, helping you build programs that work the way you want them to work. But if you try to write longer programs you may find that using the blocks can get cumbersome. For example, all of your stacks may no longer fit a the screen at the same time. Also, there are some advanced programming features that are not available using the blocks. As your programs become longer and more complex, you may want to consider using the Text Language that is available from within the PicoBlocks software. The PicoBlocks Text Language is similar to the computer language Logo. (For more information on Logo see www.logofoundation.org.) To begin writing programs in the Text Language, click on the text icon located in the lower left hand corner of the PicoBlocks Workspace. The Text Language Workspace, shown in the figure below, will appear.



redraw figure

Try typing the following commands in the Command Center. (Press the Enter key to run each line.)

```
flash
```

The LogoChip flashes!

```
flash wait 2 flash
```

The LogoChip flashes twice. Note that spaces are very important in the Text Language; they are used to determine where words start and end. For example, if you leave out the space between wait and 2 in the last example you will get an error message that says “I don’t know how to wait2”. This is because PicoBlocks does not understand the word wait2. Each block in PicoBlocks has an associated word in the text language. The general rule for finding the name of the word in the text language that is associated with a particular block is a very simple one:

Just copy what's written on the block, leaving out a space if there is one.

For example, plug an LED into output #1 your LogoChip and type the following in the Command Center:

```
on1 20
```

The light turns red Note that there are no spaces in the word on1.

Now try typing the following line in the Command Center:

```
repeat 5 [flash wait 2]
```

In the Text Language square brackets are used to enclose the list of words that are to be repeated. You can get guidance about how the text word corresponding to a particular block is meant to be used by clicking on that block, which you can find on the left hand side of the Text Language Workspace. For example if you click on the “repeat” block the following text will appear in the Command Center:

```
repeat 10 [ WHATEVER ]
```

In this message “WHATEVER” is meant as a place holder for the list of commands that you want repeated.

Procedures

You can “teach” your LogoChip to understand new words by defining procedures in the Procedures Area of the Text Language Workspace. Procedures are defined using the following general form:

```
to procedure-name  
  procedure-body  
end
```

The procedure definition starts with the keyword `to`, followed by the name of the procedure. Next comes the body of the procedure, which is a list of words that describe what the procedure is to do. The keyword `end` is used to complete the procedure definition. For example, in the *Procedure Area* type:

```
to flash-twice  
  flash wait 5 flash  
end
```

You’ve now taught your LogoChip a new word; if you type the word `flash-twice` in the Command Center, the LogoChip will chirp twice.

Procedures with Inputs and Outputs

You can create procedures that take inputs or produce an output. For example, you can create a procedure named `flashes` that takes an input that is used as the counter in a repeat loop. Note that the name of the input must begin with a colon:

```
to flashes :num
  repeat :num [flash wait 5]
end
```

Typing

```
flashes 10
```

in the *Command Center* will cause the LogoChip to flash 10 times. You can create a procedure that takes two inputs that will determine both the number of flashes and the duration of the pause between flashes:

```
to 2flashes :num :pause
  repeat :num [flash wait :pause]
end
```

Typing

```
2flashes 10 3
```

in the *Command Center* will cause the LogoChip to flash 10 times, with a pause of 3 tenths of a second between flashes.

Procedures may return values using the `output` command. For example, you can define a procedure called `half` by:

```
to half
  output sensor1 / 2
end
```

Then, with a light sensor wired to input #1, try typing in the *Command Center*:

```
print half
```

Or, you can define a procedure called `dark?` that returns a true or false result:

```
to dark?
  output (sensor < 200)
end
```

Now if try typing in the *Command Center*:

```
forever [if dark? [flash]]
```

your LogoChip should flash when it is dark enough.

Make Your Own Blocks

You can use the Text Language to make different kinds of blocks, which you can then use to build programs in the PicoBlocks Workspace. This feature allows you to continue doing much of your programming using the Blocks Language, with the Text Language being used as a supplement when its advanced features are needed. New blocks are defined by typing in the Procedures Area. The block definitions begin with the keyword `block`. For example:

```
block 1flash
flash wait 3 flash
end
```



Makes a block that chirps twice.

Blocks that you have defined will automatically be created and placed in the “My Blocks” area. You can create blocks with up to three inputs. For example, to create a block with one input:

```
block flashes :num
repeat :num [flash wait 3]
end
```



To create a block with two inputs:

```
block flash-v2 :num :pause
repeat :num [flash wait :pause]
end
```



When you define a block that outputs a number, PicoBlocks will automatically create a number-shaped reporter block:

```
block sensor+
output sensor1 + 100
end
```



Use the keyword `block` when you want to make a block that outputs the result of a true/false condition:

```
block small?
output sensor1 < 50
```





end

Variables