Build Your Own Programmable Brick: An Introduction to the LogoChip

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.[1]

Why is it that most people view the subject of electronics as inaccessible and esoteric? Well,

[1] Even among the "technically minded", one finds a surprising amount of discomfort with even relatively "simple" electronics. For example it is striking how often in casual conversation working scientists will volunteer that they feel under prepared in their knowledge of electronics.

for starters, the electronic world is inherently a more abstract and less familiar one than, say, the mechanical world. We interact with the mechanical world directly with our senses, which has lead to a multitude of ways in everyday life for us to play with mechanical ideas, building with blocks or taking apart a car engine for example. There have never been many good opportunities for novices to play with electronics, but in recent times the situation has gotten noticeably worse. The ever increasing complexity of our electronic devices has served over time to diminish the depth of our experience with the inner workings of the electronic world. In the days before the coming of solid state electronics and integrated circuits, one could more easily look at a circuit and see how the electron stream flowed. Over the decades the capabilities of our electronic technologies may have progressed at incredibly rapid rates, but our access to good ways to for non-experts to creatively play with electronic circuits and ideas has lagged far behind. For a young Richard Feynman, the process of tinkering with a broken radio could serve as an ideal playground for exploring electronics (and also act as a launching pad for making connections to other scientific ideas). But for the present age of digital electronics, few comparable experiences beckon.

The standard treatments of electronics in books or the curriculum are often too formal and theoretical. Or they adopt the style of a cookbook, providing recipes for constructing interesting electronic gadgets, but not aiming to provide a deep understanding of the underlying principles. By only constructing circuits that have been designed by others, readers miss an important opportunity to learn the art of debugging a flawed design. In either case these approaches fail students along a critical dimension; they typically do not bring most to the point where they are able or even want to design their own electronic circuits or build their own electronic inventions.

In most of the standard introductory electronics experiences the devices being constructed often do not have a particularly compelling use. Typical are “bulbs and batteries” exercises, in which students use a kit consisting of a few batteries, flashlight bulbs and wires to build and explore simple dc circuits. While the underlying ideas that are in play here (*e.g.* series vs. parallel circuits, Ohm’s Law, short circuits, *etc.*) may be accessible at a fairly early age, it is hard to see how someone would be inspired to extend this kind of activity into a long term project. The remarkable electronic gadgets that surround us in the world today raise expectations to a very high level. Kids are bound to be somewhat disappointed with an exercise that results simply in turning on a flashlight bulb.

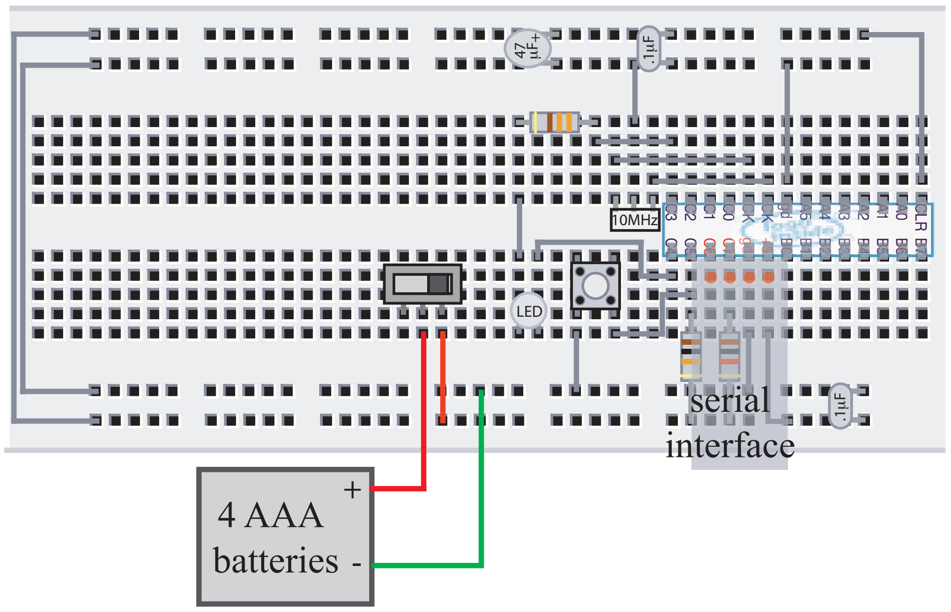
Introductory electronics books and kits usually adopt a bland style that appeals to a narrow segment of geeky hobbyists. To see what the client base for these books and kits looks like, just walk into your neighborhood *Radio Shack* and check out the customers, particularly the group hovering in the electronic components section: The group tends to be overwhelmingly male and the projects that are highlighted represent only a small range of the possibilities. (One telltale symptom: Few if any craft materials appear in the books.)

But this need not be so. With a more eclectic set of sample projects and a more imaginative exposition there are many more people who would become fluent designers and creators of electronic inventions. For example, there are many ways to make connections to the arts community by using electronics to create interactive art.

If the important and powerful ideas of electronics are to be more widely accessible, new tools and new methods are needed. We envision an approach that is playful in spirit while still seeking to engage learners in the deep underlying principles of electronics and the broad connections of these ideas to other areas of engineering and science.

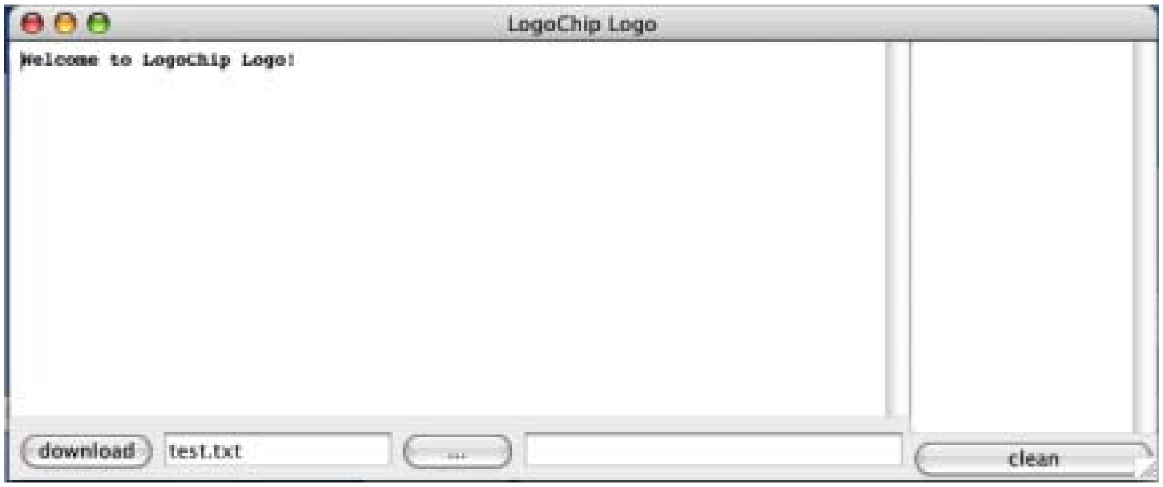
LogoChip: A new electronic construction kit

Paradoxically, the same microelectronic technologies that have contributed to the black-boxing of modern electronic circuits can also be used as the basis for a new kind of construction kit that will help reintroduce a vigorously creative and playful dimension into the design of electronic inventions. The basic building block that we have developed for this kit is an easily programmable and inexpensive embedded microcontroller called the LogoChip. Users can develop programs using a special version of the Logo programming language on a desktop or laptop computer and then download these programs to the LogoChip. LogoChip Logo combines all the power and elegance of the Logo programming language with the ability to directly configure and control the individual pins on the LogoChip.



Hello World!

You can write 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 LogoChip Logo that was written by Brian Silverman with help from Bakhtiar Mikhak and Robbie Berg. The user interface for LogoChip Logo is shown below. A brand new LogoChip knows how to execute certain “primitive” commands already. Try typing the following commands in the **command-center**.

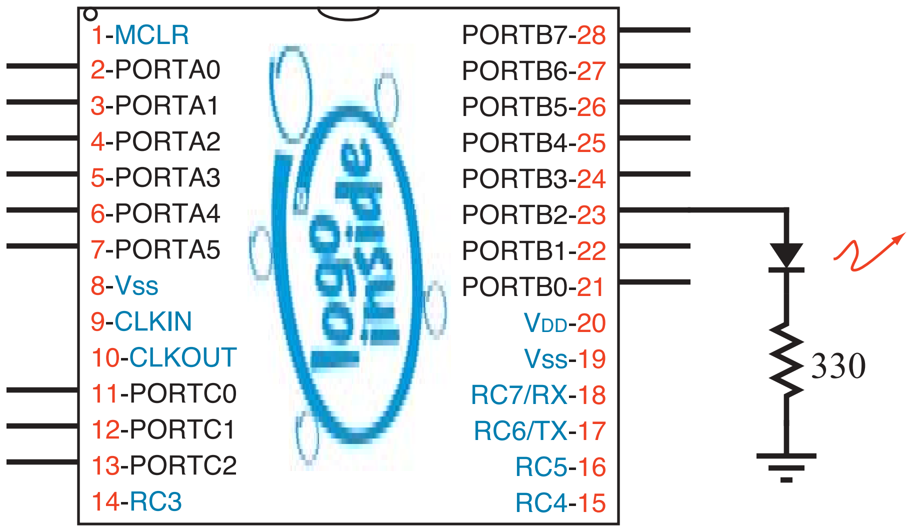


```
Welcome to LogoChip Logo!
```

√flash	The red /green indicator LED flashes!
√ flash wait 20 flash	Flashes, waits 20 tenths of a second, & flashes again.
	wait grabs control
√ repeat 4 [flash wait 20]	Repeats flash/wait 4 times
√ loop [flash wait 20]	Repeats flash/wait infinitely
√	

Press START/STOP button to stop loop (or any LogoChip Logo program). 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 powered up, but no program is running. If the indicator LED is off, it means the LogoChip is not receiving power.

Getting Flashy



Suppose an LED is connected to pin B2 as shown above. Here is what the Logo code that would make the LED flash once might look like:

```
to light-on
clearbit 2 portb-ddr            ; turns B2 into an output
                                ; see LogoChip Ins and Outs
                                ; discussion below
setbit 2 portb                  ; makes the level on B2 HIGH (+6V)
end


to light-off
clearbit 2 portb         ;makes pin 2 of portb 0V
end

to blink
light-on
wait 1
light-off
end
```

- Try typing this code in a text file and save the text file in the same folder as the LogoChip Logo software.

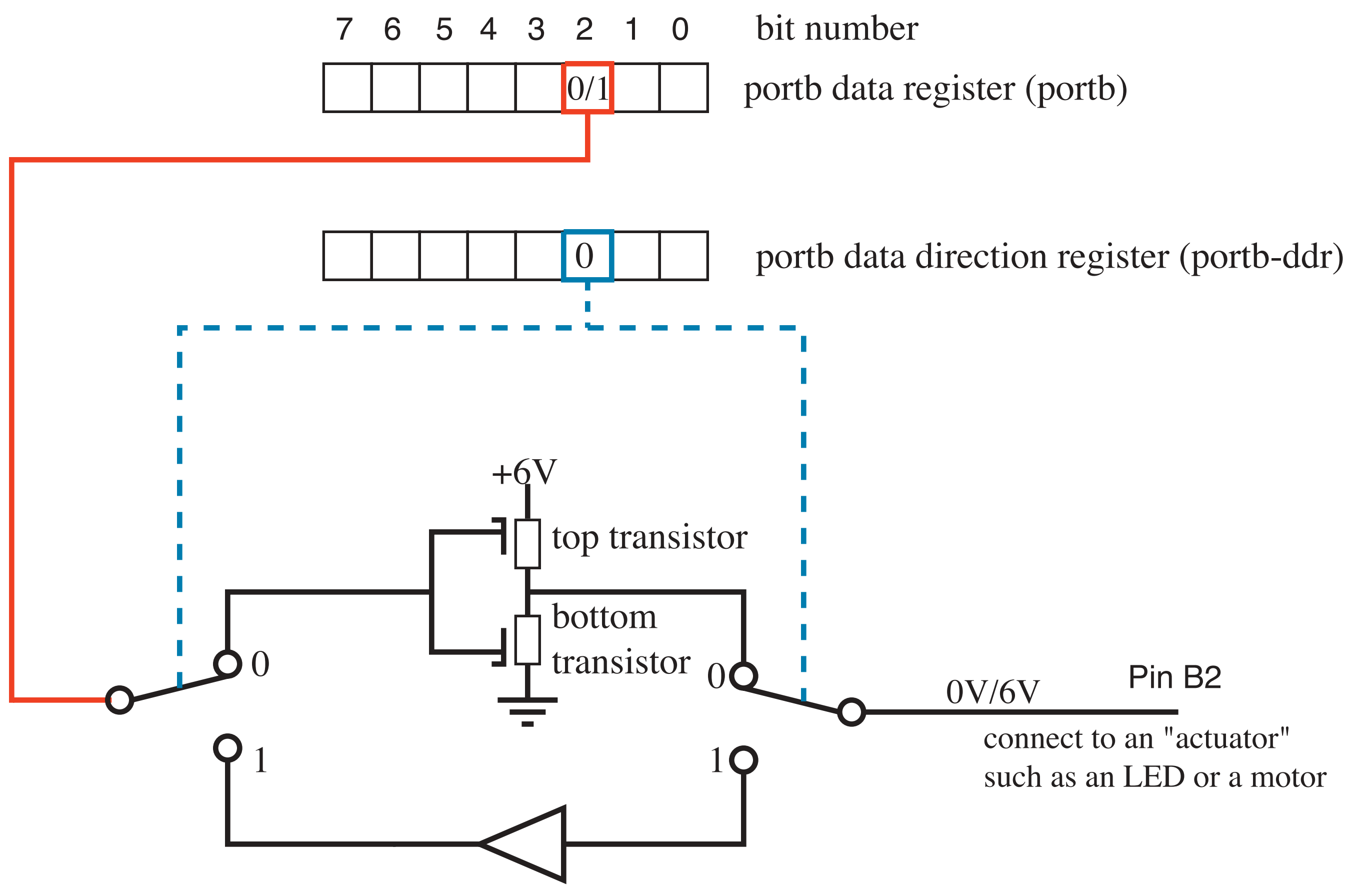
- Type the name of the text file in the text box in the lower left hand corner of the LogoChip Logo screen. Click on the <download> button on the screen to download the code to the LogoChip.

- Try typing the command `light-on` in the command-center. Next try `light-off.` Try `blink`.

LogoChip Ins and Outs

The inside of the LogoChip is organized into entities that we will call **registers**. Each register is a collection of 8 bits (1 byte). Each bit can be either a “0” or a “1”.

There are lots of registers inside the LogoChip, but for now let ‘s just focus on two of them that are associated with the 8 pins on the LogoChip that we refer to collectively as **portb**. These registers are illustrated in the somewhat cartoonish drawing below: and are called the **portb data register** and the **portb data direction register.** (Similar registers exist for porta and portc.)


7 6 5 4 3 2 1 0
bit number
0/1
portb data register (portb)
0
portb data direction register (portb-ddr)
+6V
top transistor
bottom transistor
0
1
0
1
0V/6V
Pin B2
connect to an "actuator" such as an LED or a motor

Pin B2 configured as an output

Each of the portb pins on the LogoChip can be configured to be either an “input” or an “output”. The “direction” of each pin is set by a corresponding “data direction bit” the data direction register. If a data direction bit is “set” (made equal to “1”) then the corresponding pin becomes an input. If the data direction bit is “cleared” (made equal to a “0”) the corresponding pin comes an output. We can think of the data direction bit as controlling the positions of the two switches in the above drawing.

A pin that is configured as an output will have a low output resistance and will have a voltage that is either 0V or 6V depending on the contents of the corresponding bit in the data register. Consider the example illustrated above. Suppose bit 2 of the portb data direction register is made to contain a “0” by running the command

```
clearbit 2 portb-ddr
```

This causes the switches to be as shown in the drawing, making pin B2 an **output**. Now, if bit 2 of the portb data register were to contain a “0” then the “bottom transistor” would turn “on” while the “top transistor” would turn “off”, thereby connecting pin B2 to 0 V.

If, on the other hand, bit 2 of the portb data register were made to contain a “1” by typing

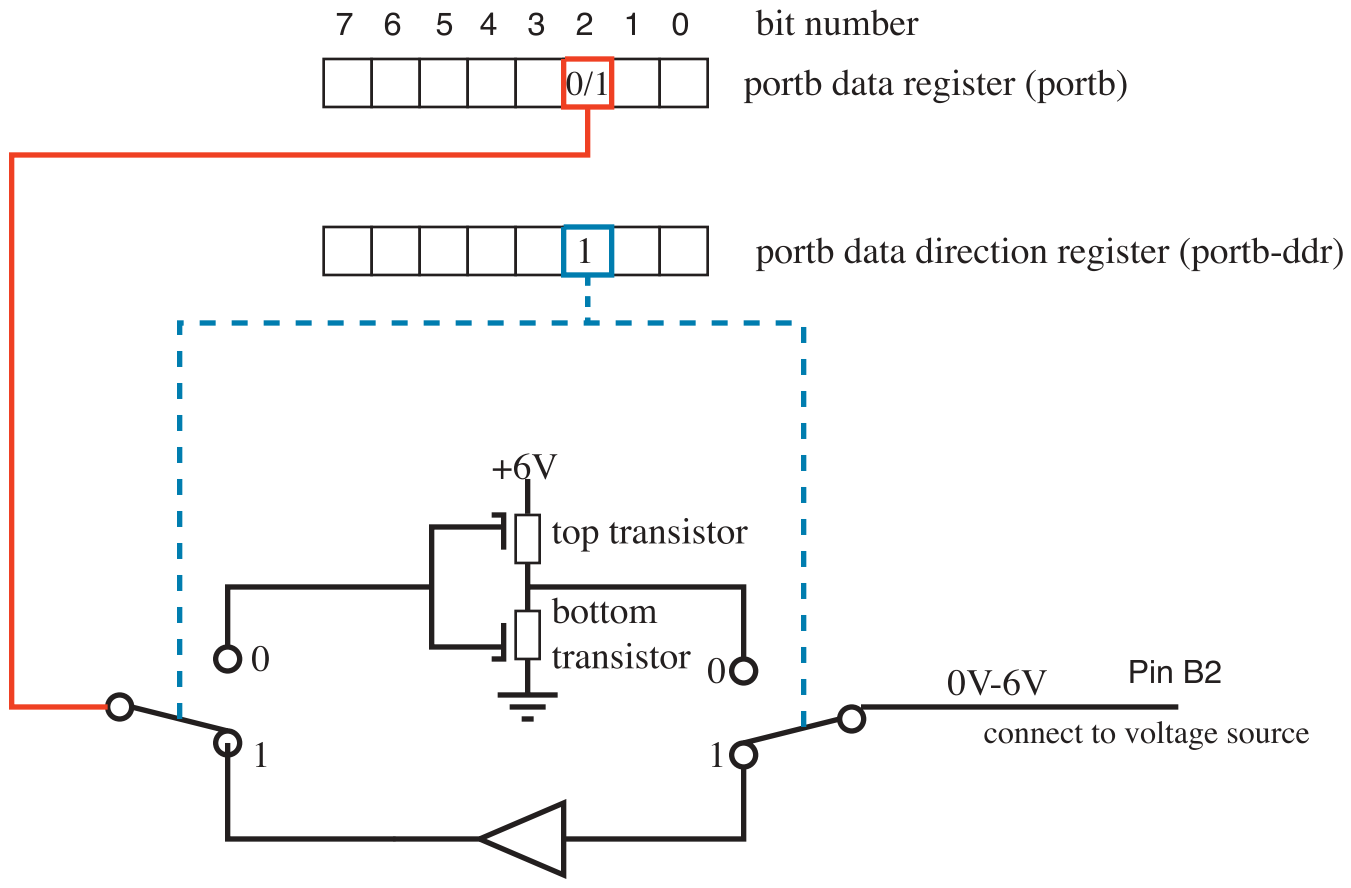
```
setbit 2 portb
```

then this would cause the “bottom transistor” would turn “off” while the “top transistor” would turn “on”, thereby connecting pin B2 to 6 V. Thus in the case where a pin is configured as an output the corresponding bit in the data register determines whether the output is HIGH or LOW. When configured as an output a pin can be connected to and used to control various “**actuators**”, such as lights, beepers, or motors.

Now suppose bit 2 of the portb data direction register is “set” (made into a “1”):

```
setbit 2 portb-ddr
```

Pin B2 is now configured as an **input** and the switches are both moved into the positions shown in the figure below.


7 6 5 4 3 2 1 0
bit number
0/1
portb data register (portb)
1
portb data direction register (portb-ddr)
+6V
top transistor
bottom transistor
0
0
1
1
0V-6V
Pin B2
connect to voltage source

Pin B2 configured as an input

In this position pin B2 has a high input resistance and can easily be driven “HIGH”(6 V) or “LOW” (0 V) by an external voltage source, such as some sort of “sensor”.

The Autonomous LogoChip

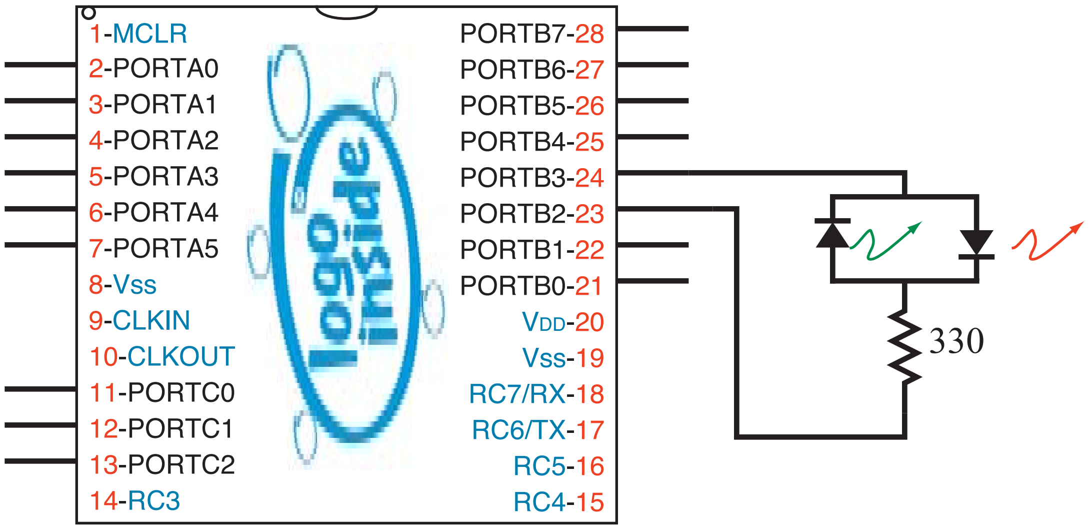
Upon powering up, all of the user accessible pins on the LogoChip are configured as “inputs”. In the first set of exercises below you will be connecting the portb pins to various actuators. Therefore we would like to configure all of the pins on portb as outputs. You can automatically turn all of the portb pins into outputs when you first turn on the LogoChip by creating the following special `powerup` procedure.

```
to powerup
write portb-ddr 0              ; turns all of portb into outputs
end
```

If you have downloaded a procedure with the name `powerup` then this procedure is run automatically when the LogoChip is turned on. Similarly, if you have downloaded a procedure with the name `startup` then this procedure is run whenever the white `start/stop` button on the LogoChip breadboard is pressed. This is an important feature. It allows the LogoChip to become autonomous from the desktop computer. You can take the LogoChip away from the computer (and even turn off the power for a long time). You can then run the `startup` procedure by powering up the LogoChip and pressing the `START/STOP` button.

Making Colors

Suppose we now connect a "red/green" LED, between pins B2 and B3 of the LogoChip. (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,)



We can use the procedures below to make the LED turn red.[2]

```
to red
setbit 2 portb
clearbit 3 portb
end
```

or green:

[2] Remember, for these procedures to work, make sure you have configured the portb pins as outputs, by running the command

```
write portb-ddr 0
```

as described above.

```
to green
clearbit 2 portb
setbit 3 portb
end
```

or, by rapidly switching between red and green states, we can make it appear yellow:

```
to yellow
loop [red green]
end
```

or some other color:

```
to other-color
loop [red red green]
end
```

Let's Get Moving

Now trying adding a red LEGO micromotor between B4 and B5 and playing with the following new procedures.

```
to on-thisway
setbit 4 portb
clearbit 5 portb
end

to on-thatway
clearbit 4 portb
setbit 5 portb
end

to off
clearbit 5 portb
clearbit 4 portb
end

to rd
togglebit 4 portb
togglebit 5 portb
end

to on-thisway-for :n
on-thisway
wait :n
off
end

to on-thatway-for :n
on-thatway
```

```
wait :n
off
end
```

One you've compiled these procedures, try typing the following commands in the command-center:

on-thisway	Turns on motor in the "thisway" direction
on-thatway	Turns on motor in the "thatway" direction
rd	Reverses direction of a spinning motor
off	Turns off the motor
on-thisway-for 20	Turns on the motor in the "thisway" direction for 2 seconds
repeat 4 [on-thisway-for 10 wait 10]	Turns motor on and off 4 times
repeat 4 [on-thisway-for 10 on-thatway-for 10]	Motor moves back and forth 4 times

Making Music

Now connect a piezoelectric beeper between pin B6 and ground and add the following new procedures

```
to click-on
setbit 6 portb
end

to click-off
clearbit 6 portb
end


to beep
repeat 100 [click-on delay 50 click-off delay 50]
end

to delay :n                 ; a variable length short delay
repeat :n [no-op]           ; no-op does nothing except take about 10
                            ; micro-seconds to execute

end
```

Play with parameters in `beep` until you get a beep of your liking.

Try writing your own `note` command of the form:

```
to note :pitch :duration
     ….
end
```

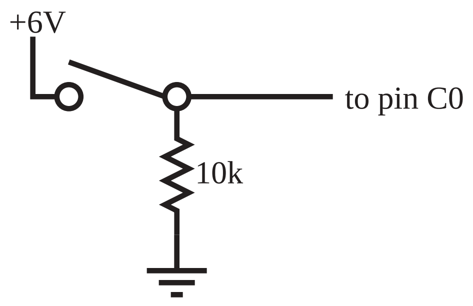
That is, `note` has two "input parameters". The first parameter determines the pitch of the note and the second parameter determines the duration of the note.

Can you compose a melody?

Getting a Sense of the World

Digital Sensors

Connect a pushbutton switch to pin C0, as shown in the figure below:



a "touch sensor"

Add the following procedure to your procedures file.

```
to touch?
output testbit 0 portc
end
```

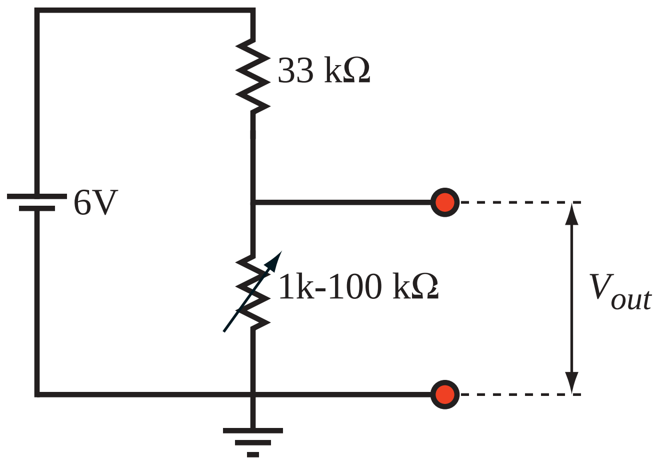
Try typing the following command in the command-center:

```
waituntil [touch?] beep
```

You've built your first digital sensor!

Analog Sensors

Build the following simple voltage divider on your breadboard. Try using a photocell or perhaps a thermistor as the variable resistor. Then connect the output of the voltage divider to pin A0.



Try typing the following commands in the command-center:

```
print read-ad 0
```

The `read-ad` primitive reports the value of the 10-bit analog to digital converter associated with pin A0. The `print` primitive causes the result to be displayed in the text box on the right side of the LogoChip Logo screen.

```
loop [print read-ad 0 wait 5]

waituntil [(read-ad 0) < 500] beep
```

You’ve built your first analog sensor!