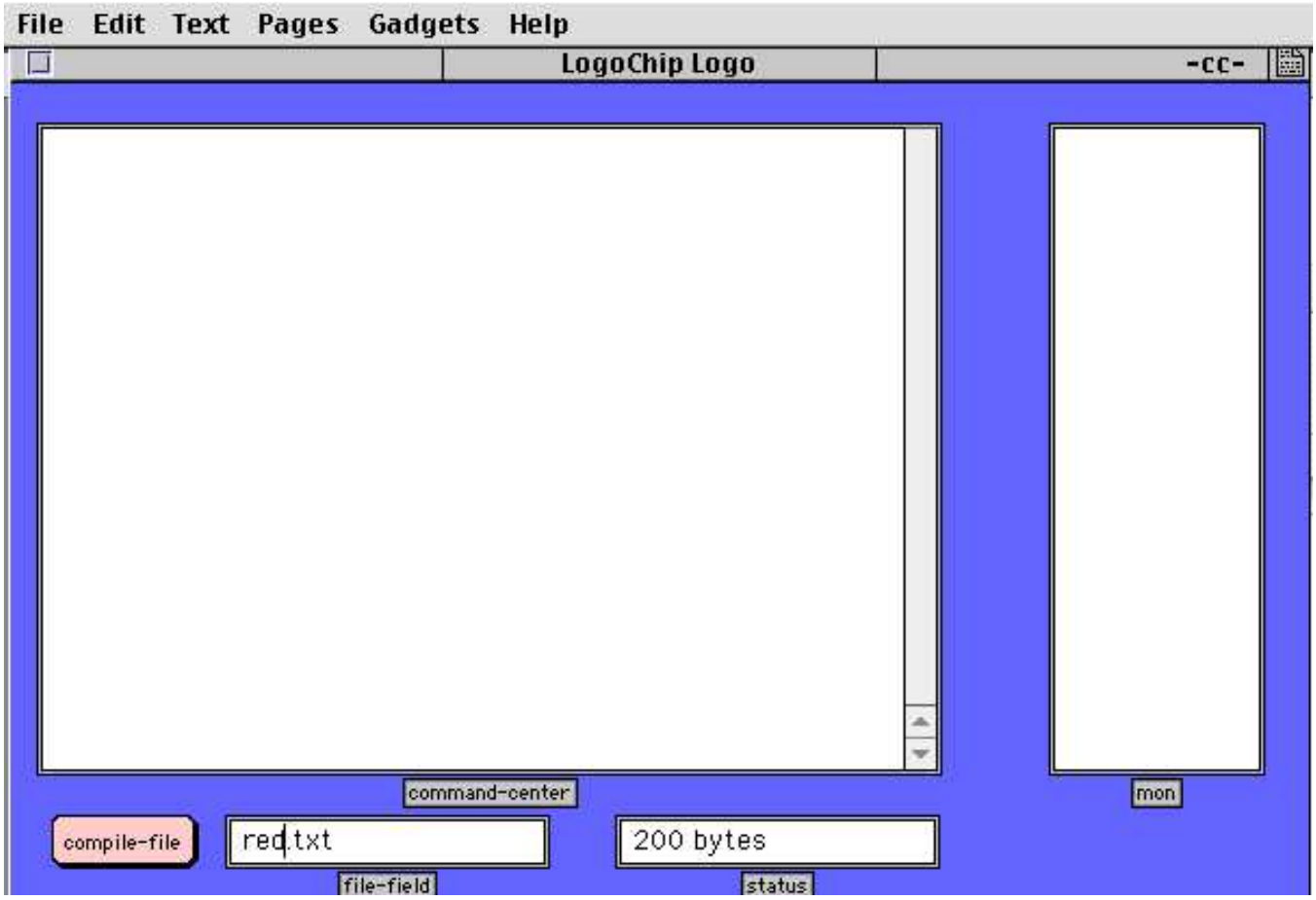
# An Introduction to LogoChip Logo



## 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**.



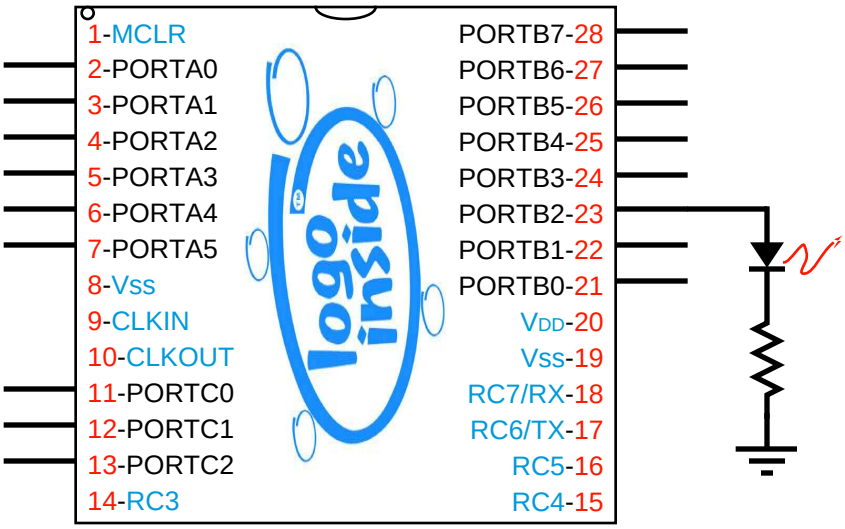

| | |
|---|---|
| √`flash` | The red /green indicator LED flashes! |
| √ `flash wait 20 flash` | Flashes, waits 20 tenths of a second, & flashes again. |
| | **`wait`** grabs control |
| √ `repeat 4 [flash wait 20]` | Repeats flash/wait 4 times |
| √`loop [flash wait 20]` | Repeats flash/wait infinitely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

√

If press START/STOP button to stop the 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 indictor LED is off, it means the LogoChip is not receiving power.

If you press the START/STOP button when no program is running (LED is red) then the LogoChip will execute the very last command that was run from the command-center. 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 program by powering up the LogoChip and pressing the `START/STOP` button.

# 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:

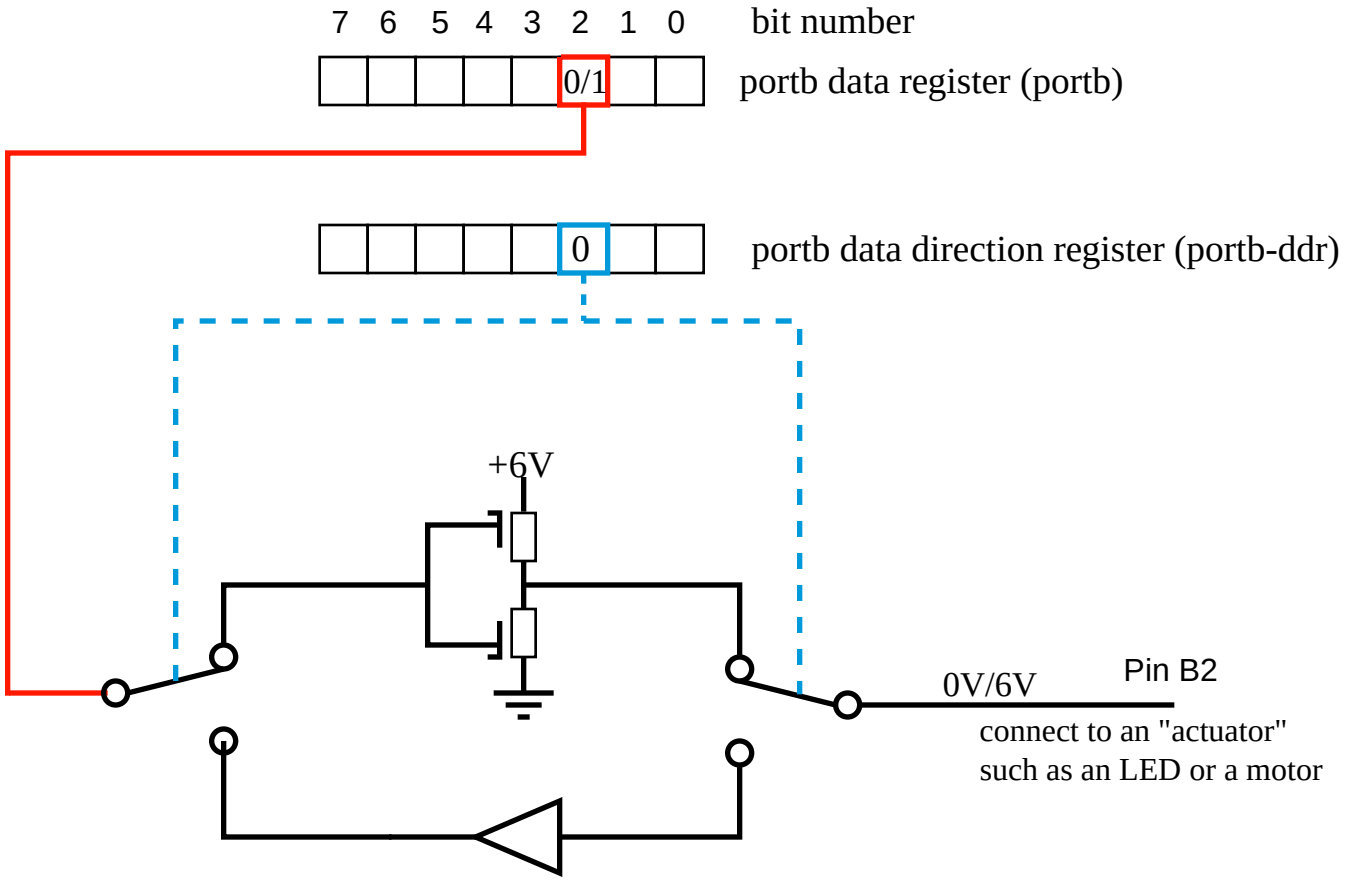
```
constants
  [[portb 6][portb-ddr $86]]

to light-on
clearbit 2 portb-ddr            ; turns B2 into an output
setbit 2 portb                  ; makes the level on B2 HIGH (+5V)
end


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

to flashy
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 LC Logo software.
- Click on the <compile-file> button on the screen to download code to the LogoChip.
- Try typing the command `light-on` in the command-center. Next try `light-off`. Try `flashy`.

# 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”. (Later in the course we will have a lot more to say about the way that these registered are implemented electronically. For now we will just think of hem in these rather abstract terms.)

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.)



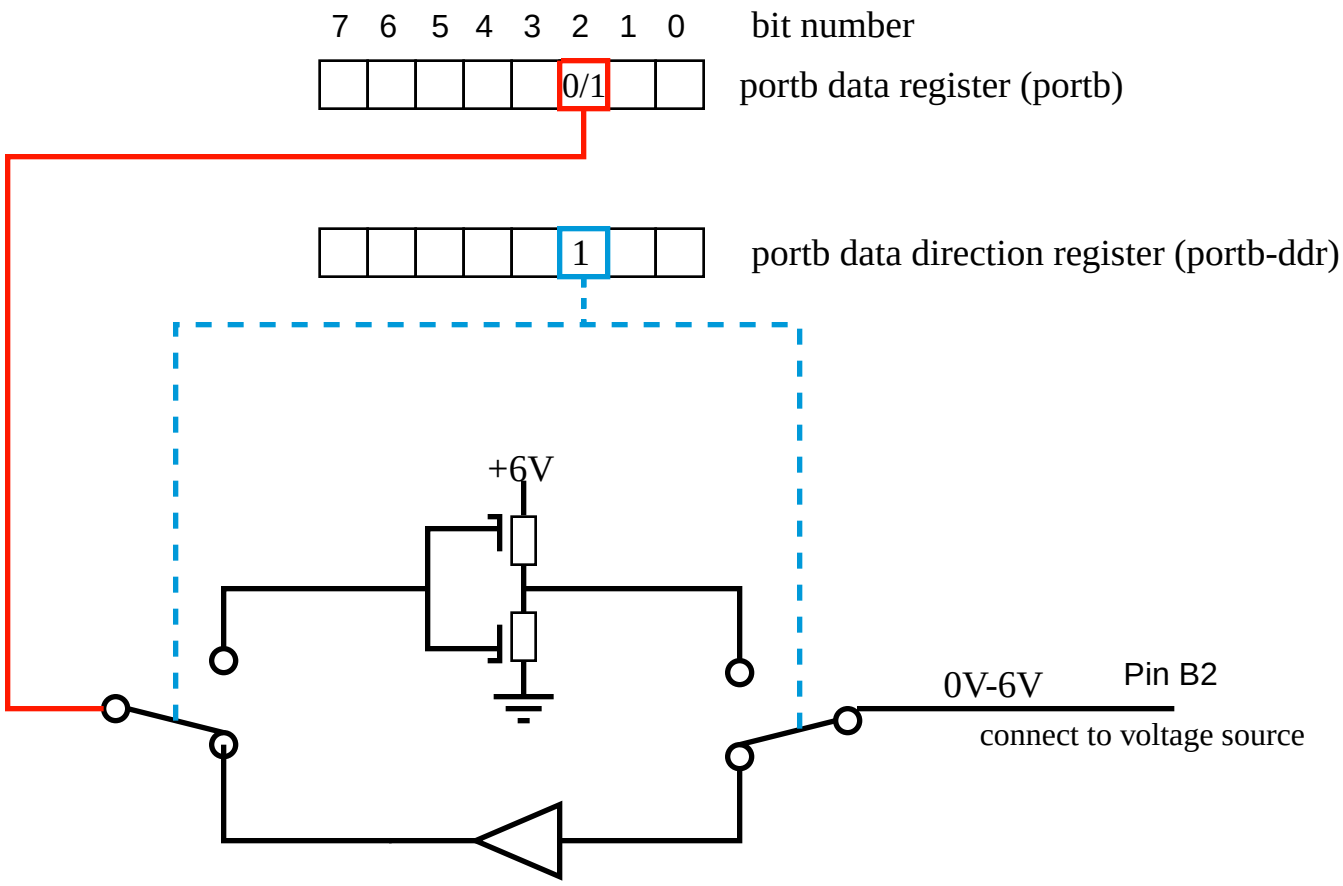

## Pin B0 configured as an output

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 has 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. Here bit 2 of the portb data direction register contains a "0". 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 tun "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 direction register were to contain a "1", this would cause the "bottom transistor" would tun "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 HGH 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 "set" (made into a "1"). Pin B2 is now configured as an input and the switches ore both moved into the positions shown in the figure below.




Pin B2 configured as an input

In this position pin B0 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".

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, *except B0* (B0 is a special pin used for bus communications). You can turn all of the portb pins into outputs by creating the following procedure

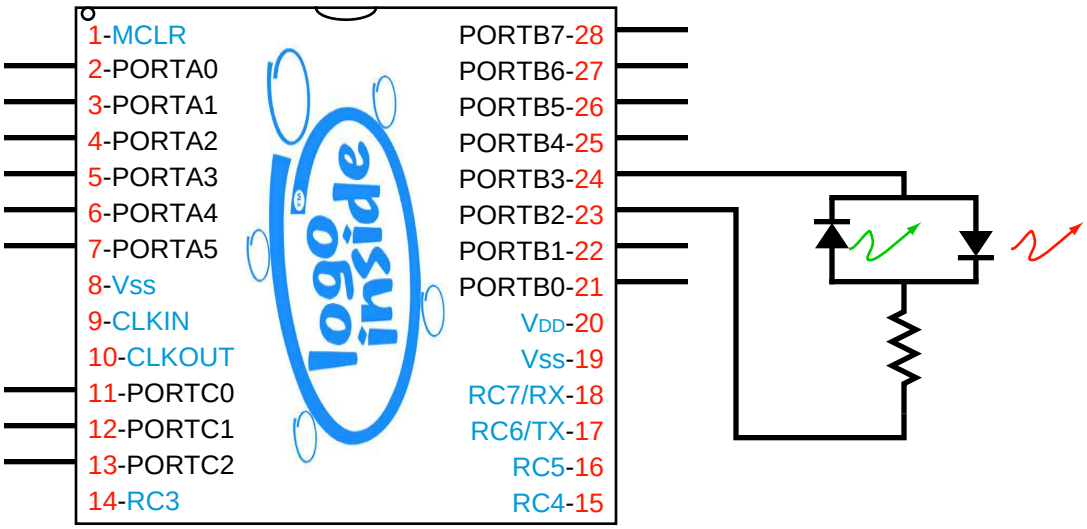
```
to init
write portb-ddr 1                ; turn portb into outputs except B0
end
```

When the LogoChip is turned off, the data direction registers reset to be all inputs. So each time you power on, you must run `init` again.

## 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 (Remember to turn the portb pins into outputs by running `init`.)

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

or green:

```
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 motor 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 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 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
repeat :n [no-op]
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
```

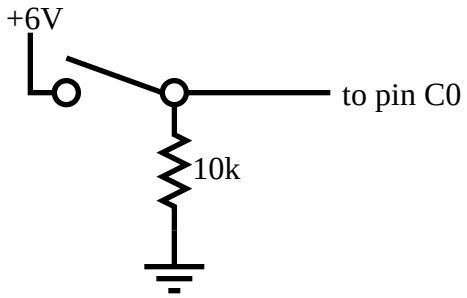
That is, `note` has two "input parameters". The first parameter determines the pitch of the note and the second parameter determines the duration of the note.

Can you compose a melody?

# Getting a Sense of the World

## *Digital Sensors*

Connect a pushbutton witch 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
```

and the definition

```
[portc 7]
```

to the list of constant definitions.

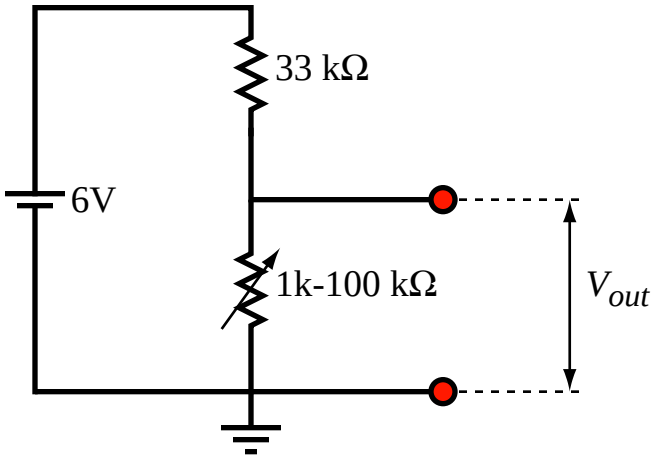
Try typing the following command in the command-center:

```
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.




Load and compile the file named **lc-tools.txt** and type the following commands in the command-center:

```
print read-ad 0

loop [print read-ad 0 wait 5]

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

You've built your first analog sensor!