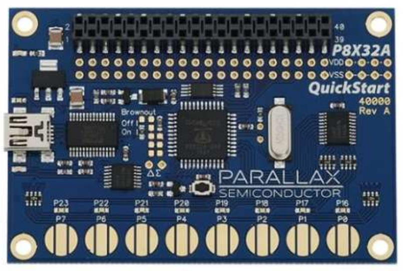
DIY Portable Radio!

The Propeller microcontroller is pretty awesome, if we do say so ourselves. With 8 different processors (or as we call them, cogs) operating simultaneously, the development possibilities are endless!

Today, you'll use the Propeller chip's native programming language, Spin to create your own portable radio. Here's what we'll be using:

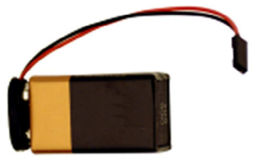
The FM Radio Receiver Module. The FM Radio Receiver Module uses a stereo radio tuner chip, which is how we'll get the Propeller to receive local FM radio stations.



The P8X32A QuickStart. The QuickStart is an open-source evaluation board for the Propeller. We'll be using it to:

- Interface to the FM Radio Receiver Module
- Adjust the volume and change stations using the onboard resistive touch-buttons
- View current volume using the LEDs

USB A to mini B Cable & Headphones. Used to power and program the QuickStart & listen to your radio!



Battery & Connecter. So you can power your radio without a laptop!

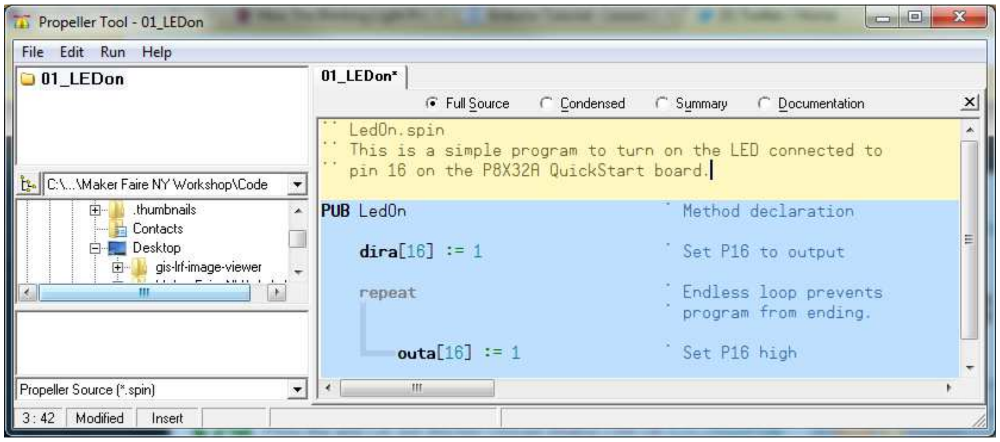
Let's get started!

Getting Started: Turning on LEDs

First, let's take a look at a simple program to turn on an LED.

- Open **DIY Portable Radio -> Code -> 01_LEDon.spin**, which is located on your desktop.

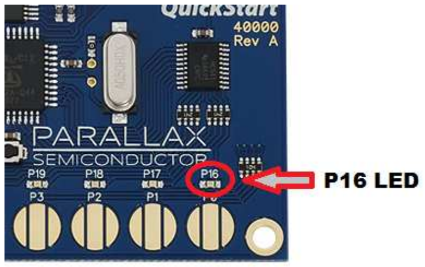
Opening this file opens the Propeller Tool, which should look something like this:



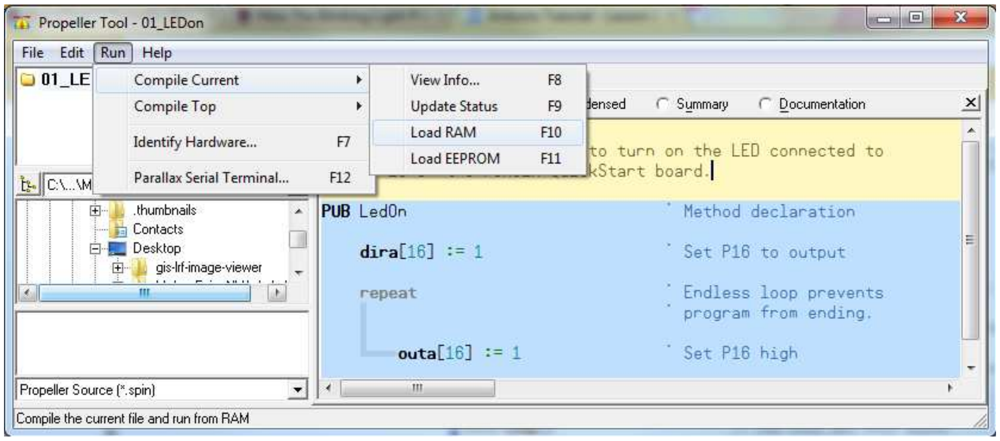
```
'' LedOn.spin
'' This is a simple program to turn on the LED connected to
'' pin 16 on the P8X32A QuickStart board.

PUB LedOn                          ' Method declaration

  dira[16] := 1                    ' Set P16 to output

  repeat                           ' Endless loop prevents
                                   ' program from ending.
    outa[16] := 1                  ' Set P16 high
```

The QuickStart has eight LEDs onboard, hardwired to pins 16-23 on the Propeller chip.

This program will turn on the LED connected to pin 16 on the Propeller.



- ✓ First, connect your QuickStart board to the computer using a USB A to mini B cable.
- ✓ Then, load the program to the QuickStart by selecting **Run -> Compile Current -> Load RAM**.



Congratulations! You just turned on a LED!

What Happened?

While turning on an LED is cool, it's even cooler to know how we did it. Let's take a take a look at some of the key pieces of this program.

Command	Description
`PUB LedOn`	Declares a public *method* named LedOn. A method is a block of code, and making the code public means you can use it in other programs too. We'll cover this more in detail later.
`dira[16] := 1`	Sets a pin's direction – whether it's an input (0) or an output (1). In this case, we're going to be controlling an LED, so we want it to output signals to turn the LED on or off.
`repeat`	This command executes any code indefinitely.
`outa[16] := 1`	Sets the output state of a pin. In this case, we want to turn the LED on, so we set it to 1.

Have some fun and play with the code to see which LEDs you can turn on and off. For example:

- Can you turn the pin 16 LED off?
- Can you turn the pin 17 LED on?
- Can you turn all LEDs on?

Once you feel comfortable turning LEDs on and off…

- ❖ Open and run **DIY Portable Radio -> Code -> 02_LedBlink.spin**.

You should now see every LED blinking on and off alternately.

Shortcuts!

This program uses a couple of "shortcuts" to help keep the code short and sweet. Below is a list of the code used, and their extended counterparts. See if you can tell what each line does!

In Program	**We Could Have Used…**

`dira[16..23] := %11111111`	`dira[16] := 1`
	`dira[17] := 1`
	`dira[18] := 1`
	`dira[19] := 1`
	`dira[20] := 1`
	`dira[21] := 1`
	`dira[22] := 1`
	`dira[23] := 1`

`outa[16..23] := %10101010`	`outa[16] := 1`
	`outa[17] := 0`
	`outa[18] := 1`
	`outa[19] := 0`
	`outa[20] := 1`
	`outa[21] := 0`
	`outa[22] := 1`
	`outa[23] := 0`

Using Objects

The QuickStart Board has eight gold resistive touch buttons hardwired to pins 0-7 on the Propeller microcontroller.

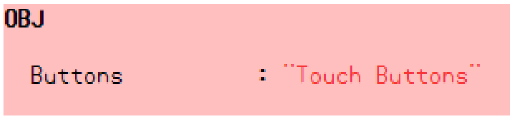
Let's turn the LEDs above the buttons on when the buttons are touched.

- ❖ Open and run **DIY Portable Radio -> Code -> 03_LedButtons.spin**.
- ❖ Make sure when you press the touch button, the LED above it turns on.

A neat feature of this program is that we're using an *object* to read the states of the touch pads. An object is another Spin program that you can use with your code.

We declare objects using an OBJ block.

The object will be called "Buttons" in our program and the filename of the Spin program is "Touch Buttons".



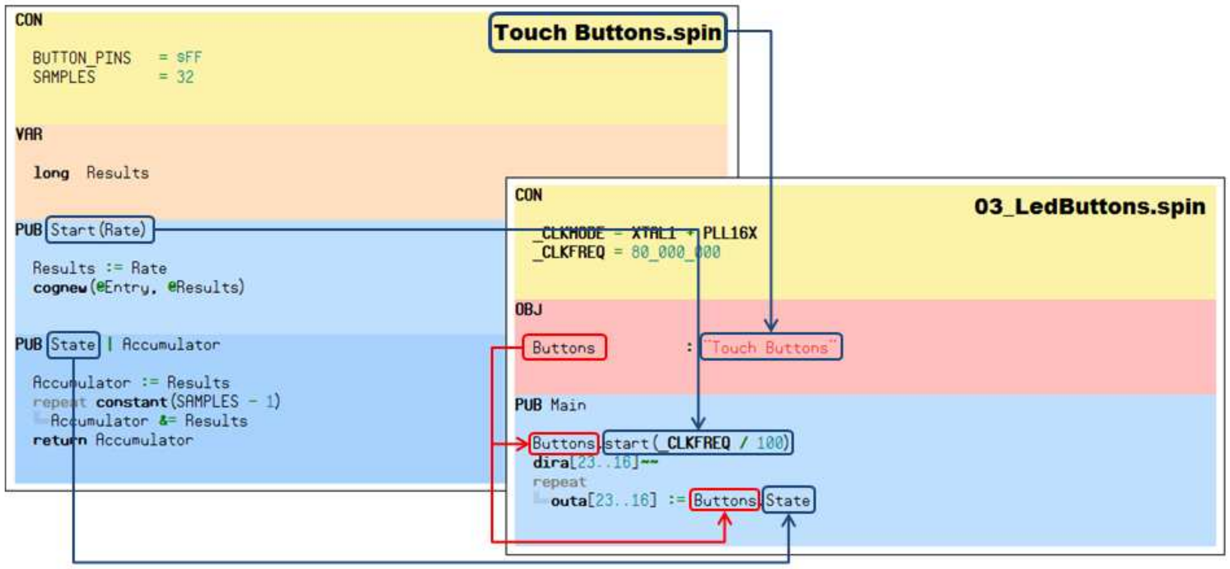
```
OBJ

  Buttons              : "Touch Buttons"
```

- ❖ Open **DIY Portable Radio -> Code -> Touch Buttons.spin**.

In the Touch Buttons object, there are two public method we can use: Start(Rate) and State.

To use the methods in our code, we use the *nickname.methodName* syntax, for example: Buttons.State.

The picture below shows the relationship between the Touch Buttons object and our program.



```
CON

  BUTTON_PINS    = $FF
  SAMPLES        = 32

VAR

  long  Results

PUB Start(Rate)

  Results := Rate
  cognew(@Entry, @Results)

PUB State | Accumulator

  Accumulator := Results
  repeat constant(SAMPLES - 1)
    Accumulator &= Results
  return Accumulator
```

```
CON

  _CLKMODE = XTAL1 + PLL16X
  _CLKFREQ = 80_000_000

OBJ

  Buttons              : "Touch Buttons"

PUB Main

  Buttons.start(_CLKFREQ / 100)
  dira[23..16]~~
  repeat
    outa[23..16] := Buttons.State
```

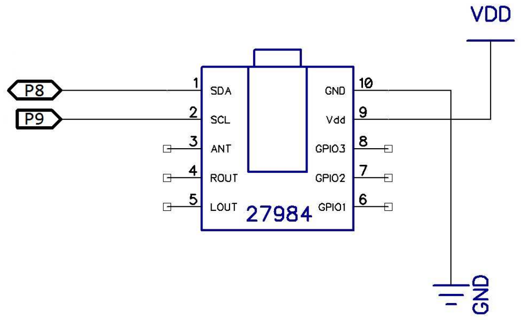
Making Our Radio

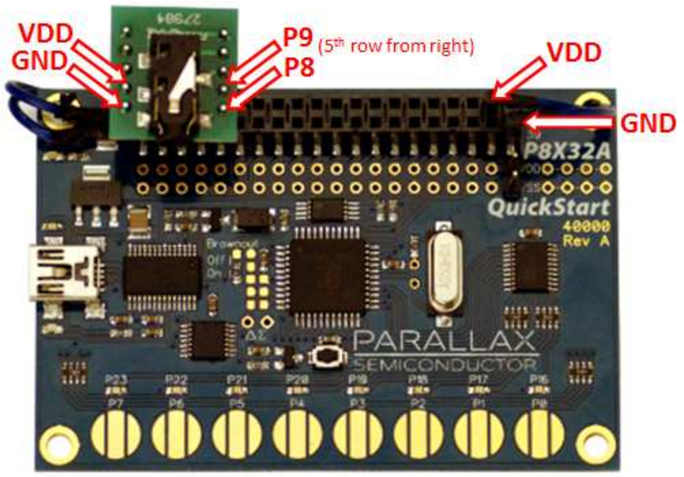
With object-oriented programming, you can easily and quickly make your own portable radio. With a bit of code and a pair of headphones you'll be ready to jam!

Connect the FM Radio Receiver Module

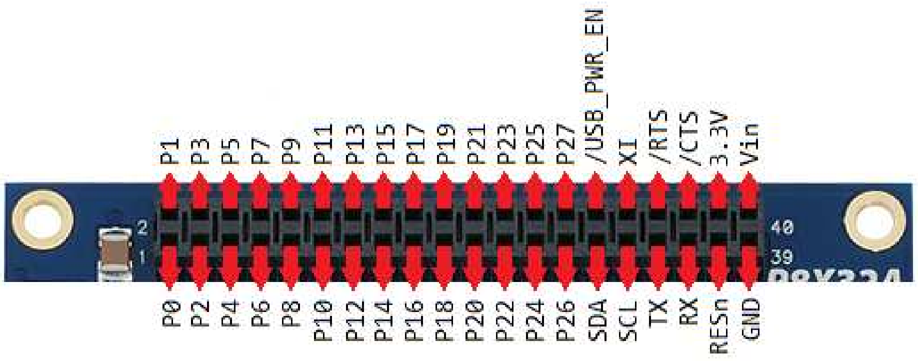
First, bend the female end of your pluggable wires so they're at a 90-degree angle.

Then, use the schematic and picture below to connect your FM Radio Receiver Module to the QuickStart Board.





You can also use this pinout to make the connection to the QuickStart Board's socket.


P1 P3 P5 P7 P9 P11 P13 P15 P17 P19 P21 P23 P25 P27 /USB_PWR_EN XI /RTS /CTS 3.3V Vin
P0 P2 P4 P6 P8 P10 P12 P14 P16 P18 P20 P22 P24 P26 SDA SCL TX RX RESn GND
2
1
40
39

Run the Program

- Open and run **DIY Portable Radio -> Code -> DIYPortableRadio.spin**.
  - Instead of choosing **Run -> Compile Current -> Load RAM**, choose **Run -> Compile Current -> Load EEPROM.** This option will keep the program on the board when power is disconnected.

Now that your QuickStart is programmed and ready to go, you'll need to setup an alternative power source (unless you're planning to carry around a laptop computer with you all day). Unplug the USB cable from the QuickStart and plug in the 9V battery clip as shown in the picture.

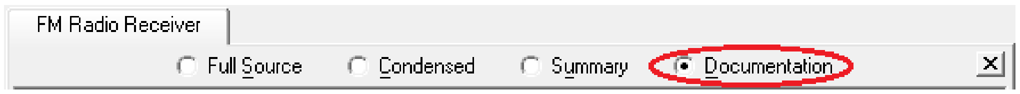
You are now ready to jam out to your favorite stations!

Before you go, take a look at how this program works…

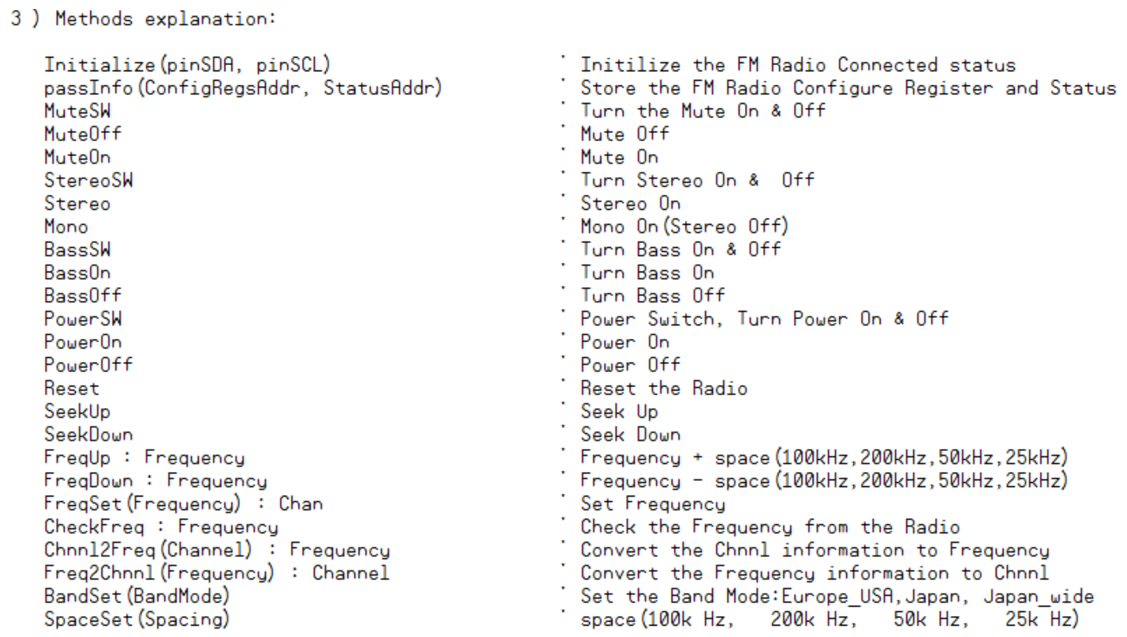
Most of the work was done for us with two objects: Touch Buttons.spin and FM Radio Receiver.spin

If you want to customize your radio, there's a quick way we can see all the methods available to us in these programs:

- Open **DIY Portable Radio -> Code -> FM Radio Receiver.spin** and click the 'Documentation' radio button at the top of the screen.



This button is extremely helpful to get a quick view of the available methods that you can call in your program. You'll see there are many options to choose from: change stations, adjust volume & stereo, etc.



```
3 ) Methods explanation:

   Initialize(pinSDA, pinSCL)                   ' Initilize the FM Radio Connected status
   passInfo(ConfigRegsAddr, StatusAddr)         ' Store the FM Radio Configure Register and Status
   MuteSW                                       ' Turn the Mute On & Off
   MuteOff                                      ' Mute Off
   MuteOn                                       ' Mute On
   StereoSW                                     ' Turn Stereo On &  Off
   Stereo                                       ' Stereo On
   Mono                                         ' Mono On(Stereo Off)
   BassSW                                       ' Turn Bass On & Off
   BassOn                                       ' Turn Bass On
   BassOff                                      ' Turn Bass Off
   PowerSW                                      ' Power Switch, Turn Power On & Off
   PowerOn                                      ' Power On
   PowerOff                                     ' Power Off
   Reset                                        ' Reset the Radio
   SeekUp                                       ' Seek Up
   SeekDown                                     ' Seek Down
   FreqUp : Frequency                           ' Frequency + space(100kHz,200kHz,50kHz,25kHz)
   FreqDown : Frequency                         ' Frequency - space(100kHz,200kHz,50kHz,25kHz)
   FreqSet(Frequency) : Chan                    ' Set Frequency
   CheckFreq : Frequency                        ' Check the Frequency from the Radio
   Chnl2Freq(Channel) : Frequency               ' Convert the Chnl information to Frequency
   Freq2Chnl(Frequency) : Channel               ' Convert the Frequency information to Chnl
   BandSet(BandMode)                            ' Set the Band Mode:Europe_USA,Japan, Japan_wide
   SpaceSet(Spacing)                            ' space(100k Hz,   200k Hz,   50k Hz,   25k Hz)
```

Pin	Method
7	SeekDown
6	SeekUp
5	VolumeDown
4	VolumeUp
3	CheckVolume

Just choose the methods you want executed when certain buttons are pressed.

These are the five methods used in DIYPortableRadio.spin.

Here are the blocks of code needed to get the portable radio to work:

- First, the radio and button spin programs are setup as objects. This is done with the `OBJ` block.

```
OBJ

  FM       :  "FM Radio Receiver"           ' FM Radio Receiver Module Object
  Buttons  :  "Touch Buttons"               ' Touch Button object
```

- Then we declare a variable, `vol`, which will store the current volume level. We do this using a VAR block.

```
VAR

  byte vol                                  ' Store volume level
```

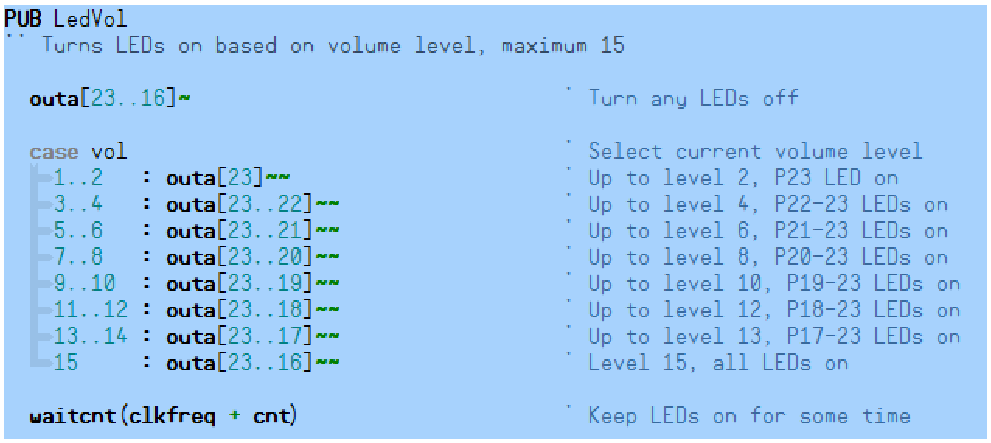
- Next we declare our main method and start the objects used in our code, as well as set some parameters.

```
PUB Main

  FM.Initialize(8, 9)                        ' Start FM Radio object
  FM.SeekThreshold(6)                        ' Set seek threshold
  FM.VolumeSet(1)                            ' Set volume level to 1

  dira[23..16]~~                             ' Set LEDs to outputs

  Buttons.start(_CLKFREQ / 100)              ' Start button object
```

- In one repeat loop we'll define what methods will run when certain buttons are pressed. We'll also turn on LEDs when the touch buttons are pressed.

```
  repeat
    outa[23..16] := Buttons.State
    if Buttons.State == %10000000            ' If the P7 button is touched,
      FM.SeekDown                            ' auto seek down for a station
    if Buttons.State == %01000000            ' If the P6 button is touched,
      FM.SeekUp                              ' auto seek up for a station
    if Buttons.State == %00100000            ' If the P5 button is touched,
      FM.VolumeDown                          ' turn the volume down
    if Buttons.State == %00010000            ' If the P4 button is touched,
      FM.VolumeUp                            ' turn the volume up
    if Buttons.State == %00001000            ' If the P3 button is touched,
      vol := FM.CheckVolume                  ' Store current volume level
      LedVol                                 ' Call LedVol Method
      outa[23..16]~                          ' Turn LEDs off

    waitcnt(clkfreq/4 + cnt)                 ' Short pause
```

❖ To turn on LEDs based on the current volume level, the LedVol method was created.



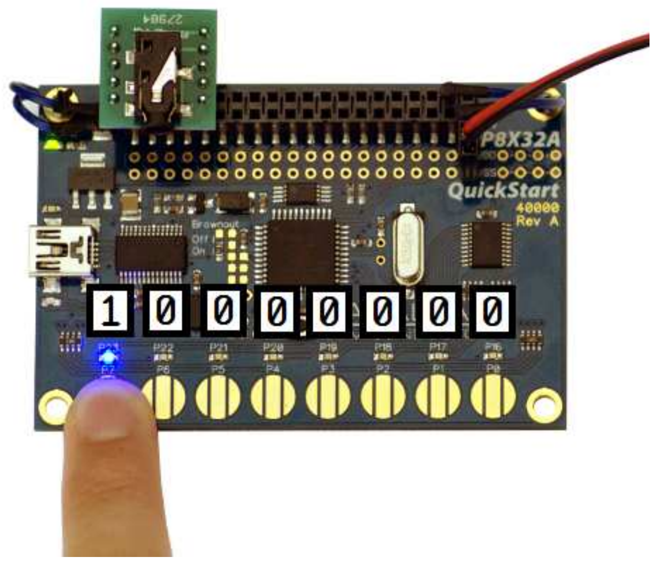
```
PUB LedVol
'' Turns LEDs on based on volume level, maximum 15

  outa[23..16]~                                   ' Turn any LEDs off

  case vol                                        ' Select current volume level
    1..2   : outa[23]~~                           ' Up to level 2, P23 LED on
    3..4   : outa[23..22]~~                       ' Up to level 4, P22-23 LEDs on
    5..6   : outa[23..21]~~                       ' Up to level 6, P21-23 LEDs on
    7..8   : outa[23..20]~~                       ' Up to level 8, P20-23 LEDs on
    9..10  : outa[23..19]~~                       ' Up to level 10, P19-23 LEDs on
    11..12 : outa[23..18]~~                       ' Up to level 12, P18-23 LEDs on
    13..14 : outa[23..17]~~                       ' Up to level 13, P17-23 LEDs on
    15     : outa[23..16]~~                       ' Level 15, all LEDs on

  waitcnt(clkfreq + cnt)                          ' Keep LEDs on for some time
```

And that's it!

Was the Button Pressed?

In this code, we need to check if a button was pressed. We do so using the `if` command, for example: `if Buttons.State == %10000000`.

The touch buttons output a high signal to the Propeller if a button is pressed, which is represented by a logical one. So if we press the P7 touch button, the Propeller sees `Button.State` like this:



New Commands

The DIYPortableRadio.spin object used a few new commands that we haven't been introduced to yet. Let's take a look at each of them in a little bit more detail.

Command	Description
`if Buttons.State == %10000000`	Compares the state of the buttons and executes code if the on is pressed is met. The % signifies a binary string of numbers.
`LedVol`	Call the LedVol method.
`case vol`	Check the value in the variable vol, compare to conditions.

Challenge Yourself!

You may not like the control pins chosen in this program, so feel free to customize this code to whatever you like! Here's how:

- ✓ Take a look at FM Radio Receiver.spin and choose what methods you want to use.
- ✓ Chose a touch button to call that method.
- ✓ Determine the binary string necessary for reading that button (i.e. P7 = %10000000).
- ✓ Write the code.
- ✓ Bask in the glory of your customized portable radio!

One for the Road

Once your QuickStart is programmed to be a portable radio, you may want to power the board from something other than your laptop's USB port.

By soldering a 2-pin header (already included in the 910-40000 kit) to the VIN and GND plated holes on the QuickStart, you can connect a 9 V battery using a battery clip (also included in the 910-40000 kit). **Make sure your USB cable is unplugged from the computer BEFORE connecting your 9 V battery.**

Proper battery connection is shown in the photo below.