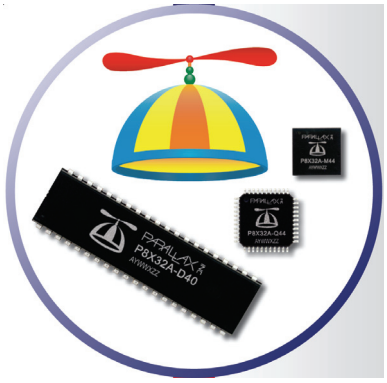
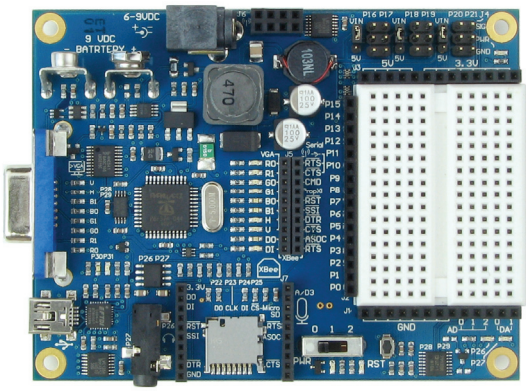
Propeller Board of Education





A.K.A. the PropBoE

Presented by: Andy Lindsay
Email: alindsay@parallax.com
Event: Propeller Meetup Group – 10/27/2011

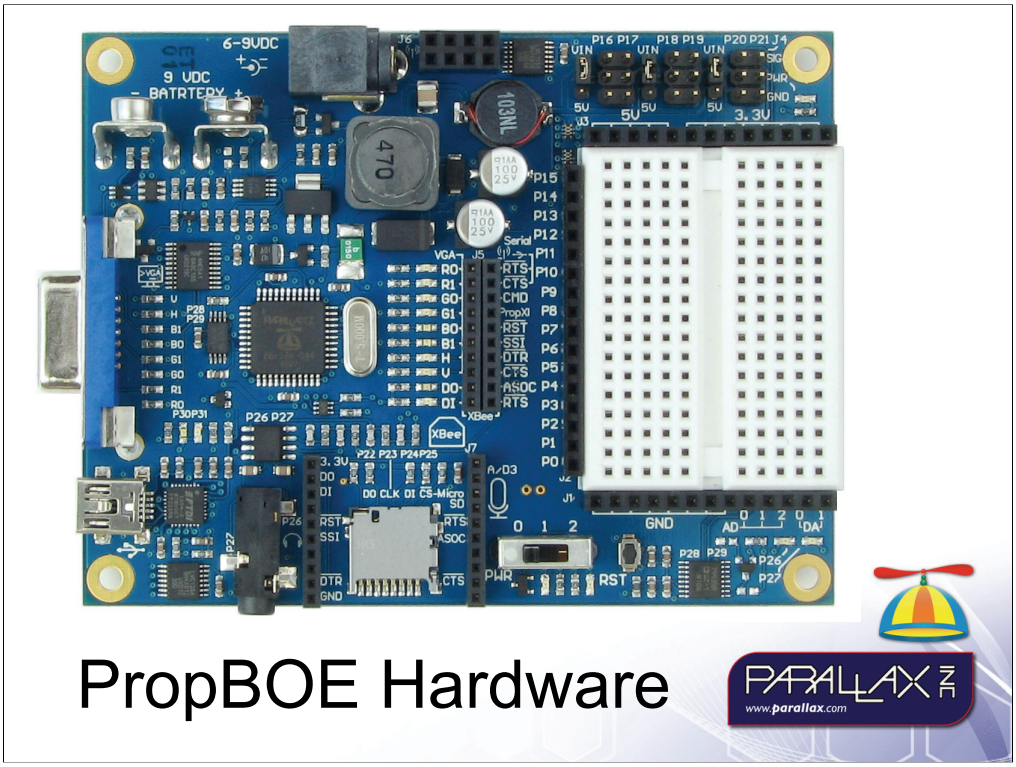
PARALLAX INC
www.parallax.com

My name is Andy Lindsay, and I work for the Parallax Education department. This presentation is about the soon to be released Propeller Board of Education, also known as the PropBOE.

Topics

- PropBOE Hardware
- PropBOE Education Program
- Status Update
- PropBOE Tutorial Excerpts

Today, we'll take a look at the PropBOE's hardware features, education program, a status update on when you might expect to see it for sale on our web site, and as many tutorial examples as time permits.



PropBOE Hardware

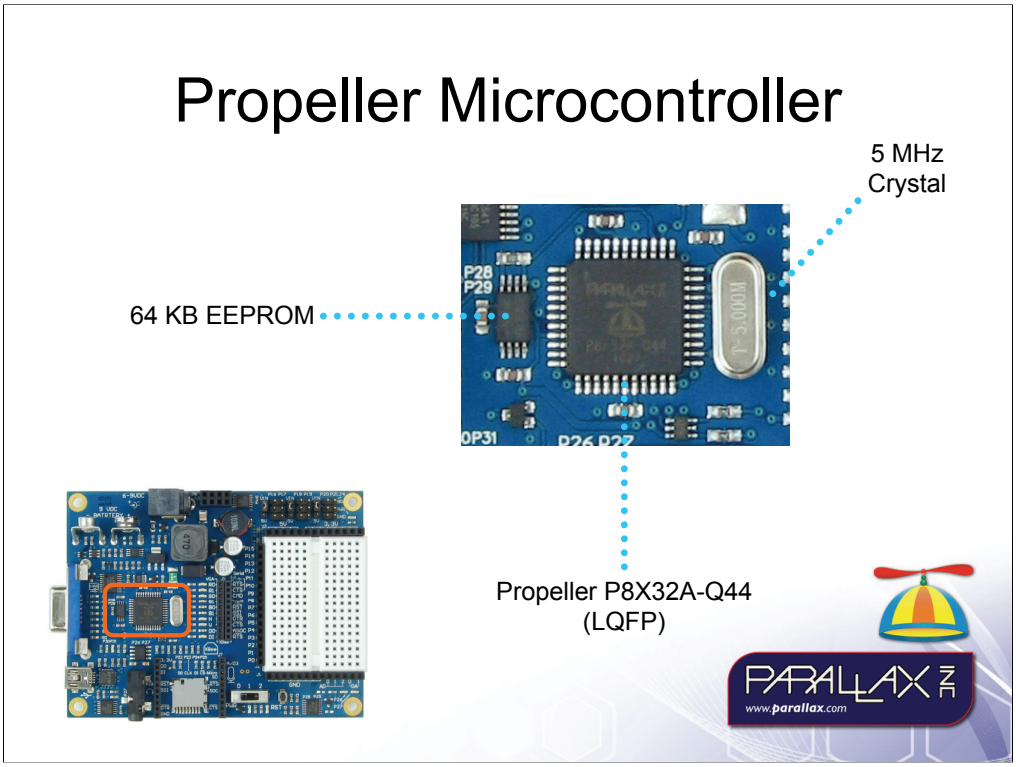
Here's a close-up of the PropBOE in its second prototype incarnation. (The third prototype printed circuit boards just got here, but have not been populated yet.) The outside dimensions and mounting hole positions are identical to the BASIC Stamp Board of Education (BOE), and familiar features like the breadboard, servo ports, on/off switch and programming port are all in roughly the same locations as they are on the BASIC Stamp BOE.

Major functionality and form factor differences (PropBOE vs. BASIC Stamp BOE) include:

(1) Socketed BASIC Stamp replaced with onboard Propeller chip

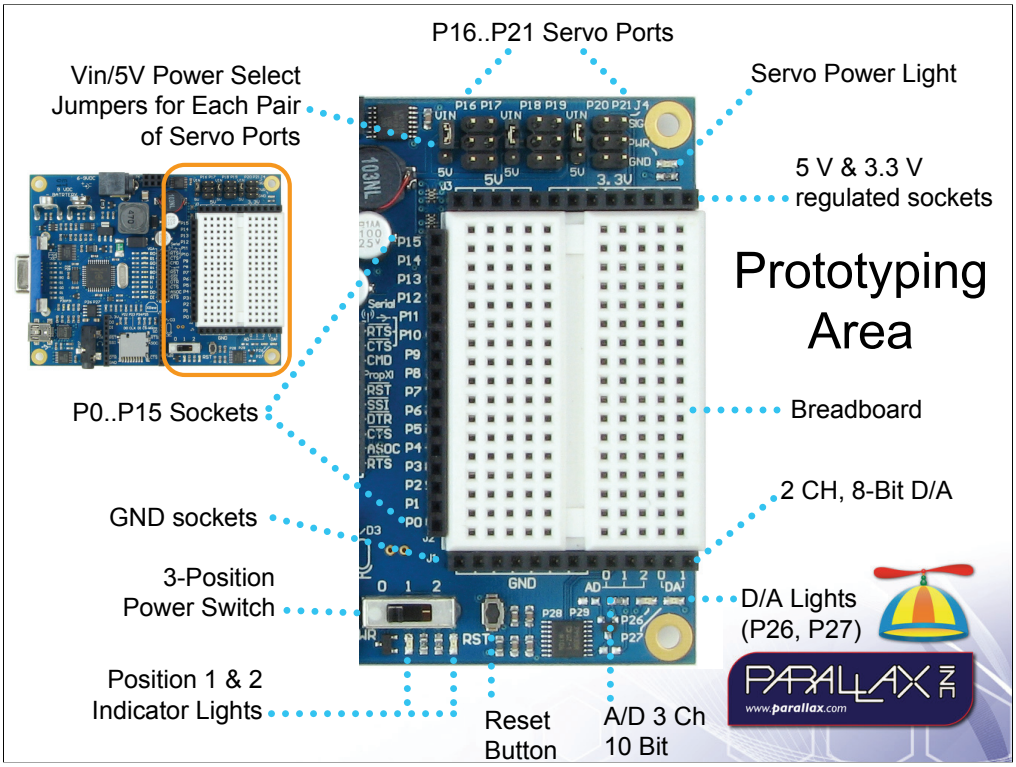
(2) PropBOE has a lots more peripheral ports, including VGA, stereo audio, and onboard D/A & A/D,

(3) The AppMod header is replaced with a peripheral port connection header. You can run jumper wires between the I/O pin sockets and the peripheral header sockets for connections to VGA, XBee, and the FTDI chip's serial flow control connections.



The location where the Propeller, EEPROM program memory, and crystal oscillator reside is highlighted in the lower-left picture, and the upper-right picture shows a close-up.

The EEPROM is 64 KB. Since the program image for the Propeller chip occupies 32 KB of the EEPROM, you can use the upper 32 kB for non-volatile data storage. This EEPROM is on an I2C bus connected to the Propeller I/O pins P28 and P29. Whenever there are dedicated connections like this, you'll see silkscreen I/O pin numbers by the chip or port.



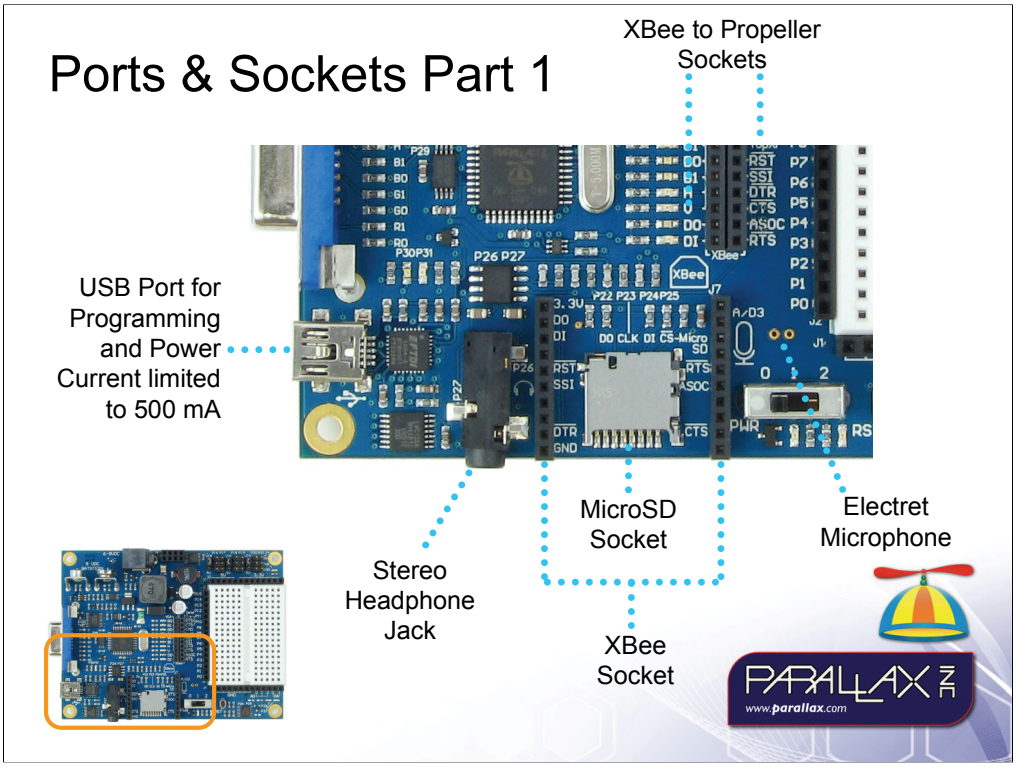
Here's the prototyping area on the right side of the board. The breadboard is the same 17-row by 2 column that's on the BASIC Stamp Board of Education.

To the left of the breadboard, we have sockets that you can use to connect to Propeller I/O pins P0 through P15. Unlike the BASIC Stamp Board of Education, these I/O pins are not shared with servos; all of them dedicated for prototyping and are not shared with any other peripherals.

Up top, we have six (as opposed to four on the BASIC Stamp Board of Education) servo connections with signals routed to Propeller I/O pins P16..P21. The servo headers are separated into three pairs, each with its own power supply selection jumper. So, each pair of servo ports can be set to draw power from either unregulated Vin or regulated 5 VDC.

Along the top edge of the breadboard, we have sockets for 5 V and 3.3 V, and along the bottom edge, we have sockets for GND. By placing power connections on the upper right and ground on the lower left, it should help reduce crossed wires for running connections to DIP chips. (BASIC Stamp BOE had 5 V on top-left and GND on top-right.)

There are three A/D and two D/A sockets on the lower-right side of the breadboard. The D/A sockets are connected to P26 and P27 and have voltage indicator lights below them. The LEDs receive the duty modulated high/low signals directly and respond in brightness to the D/A setting. The duty modulated signals pass through Sallen-Key filters on their way to the I/O pins, and the filter's op amp buffers the



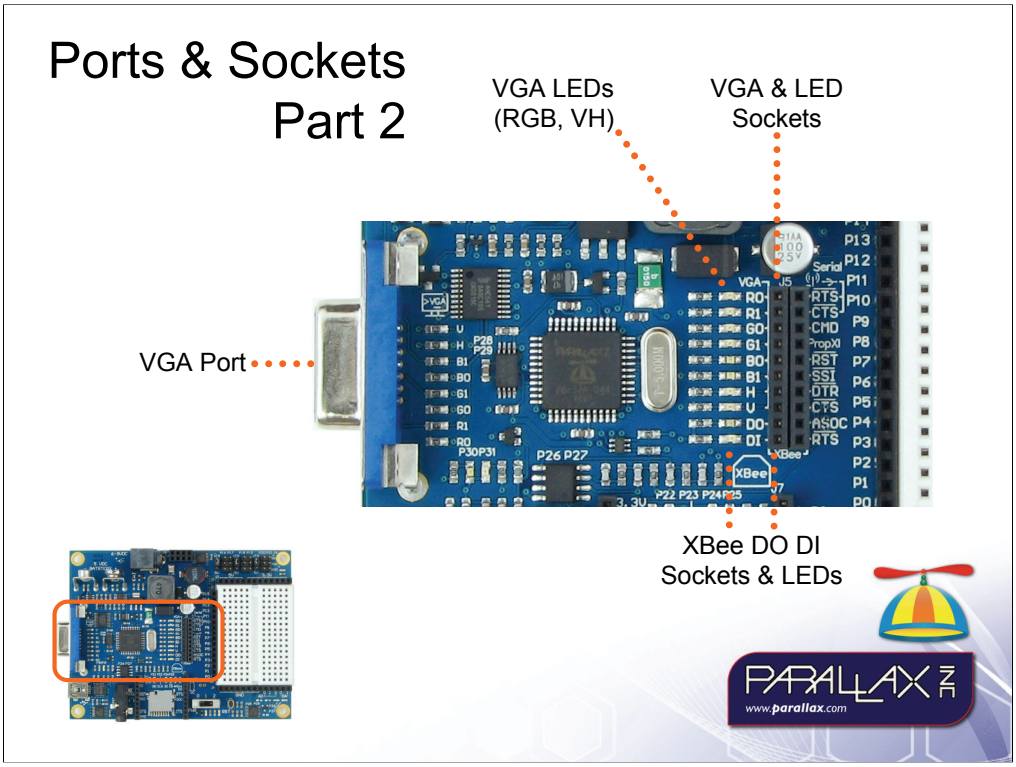
Just above the 3-position switch there's an electret mic. Its output passes through an amplifier to a fourth onboard ADC channel. The other three are connected to sockets below the breadboard.

To the left of the Power switch, we have a MicroSD card socket flanked on both sides by the two rows of an XBee radio socket. The MicroSD card is hardwired to I/O pins P22, P23, P24, and P25. The relationship of Propeller I/O pins to MicroSD pins is indicated by the silkscreen just above the socket, with P22 connected to DO, P23 to CLK, and so on…

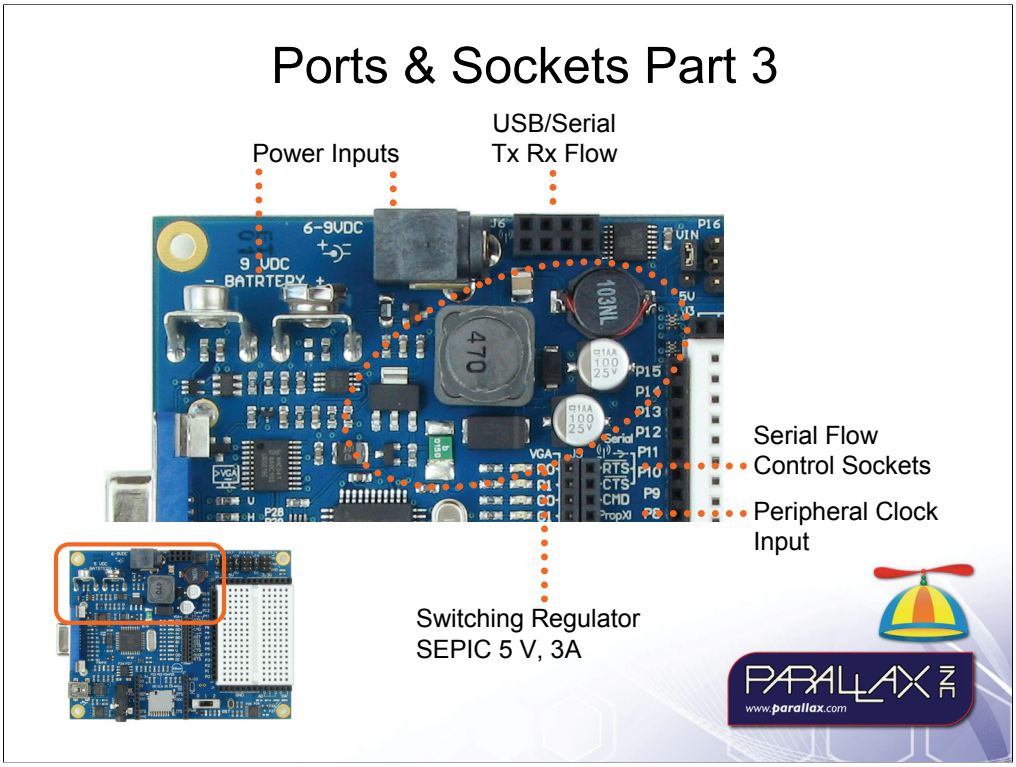
Many of the XBee socket connections are routed to the XBee to Propeller Sockets at the top of the picture. The typical setup involves running jumper wires from I/O pins to DO and DI, but the Xbee to Propeller sockets keep the option open for other applications that might involve flow control, signal strength monitoring, and XBee analog and digital I/O operations. To add any of these features, just run a jumper wire to the corresponding XBee to Propeller socket, and of course update your code accordingly.

Take a look just to the left of the Xbee to Propeller DO socket, there's a blue LED that indicates a signal from the XBee to the Propeller. Likewise, there's a red LED next to the DI socket to indicate signal activity from the Propeller to the XBee. The same convention is used for serial programming with the Propeller Plug and other serial connections.

The stereo headphone jack is the same D/A output that goes to the D/A sockets



The Propeller I/O to Peripheral header just to the left of the I/O pin sockets also provides optional connections to the VGA port. To the left of the VGA sockets, red green, and blue signal LEDs are color coded to indicate red, green, and blue VGA channel activity, and the horizontal and vertical channels are color coded yellow. This color coding has a pretty cool effect when there's more of a particular color on the VGA display because the color coded LEDs for that channel will be brighter. The VGA LEDs are also buffered instead of directly connected to the VGA circuits. This makes the VGA LEDs convenient for simple LED light experiments because it removes any interaction with the VGA circuits. (Without the buffer, a high signal to one LED might make an adjacent LED turn on as well.)



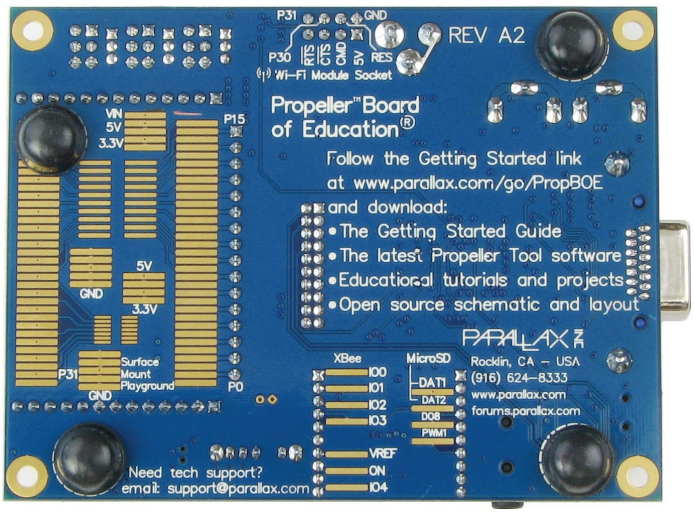
The PropBOE is designed to draw power from either the 9 V battery or 2.1 mm center positive supply if connected. If neither is connected, it'll draw power from the USB port (if that's connected). The board has a 5 V, 3 A SEPIC switching regulator that is designed to hold a 5 V output with input voltages as low as 4 V. (After testing, the silkscreen may be changed from 6-9 VDC to a wider range.)

Just to the right of the power inputs, there's a serial connection. One of the rows is compatible with the Propeller Plug with Rx, Tx, /RES, and Gnd. The second row has access to Serial/USB flow control. It was originally designed for a Wi-Fi module what will not be manufactured, but was kept on the board for its alternate programming connection and for flow control, which can be used for 2+ Mb/s serial communication over USB with the PropBOE's onboard FTDI USB/serial chip.

The Propeller I/O to Peripheral header to the left of the I/O pin sockets also has serial flow control sockets as well as PropXI input for applications that use an a programmable oscillator or other external clock source.

…and, on the other side…

- Pad access to Vin, and regulated 5 V, and 3.3 V
- Pad access to all 32 I/O pins with 0.1 inch spacing
- I/O pads alternate with pads routed to SOIC and MSOP IC pads
- I/O pad access to any XBee and MicroSD pin not available on top side socket
- Serial Socket map
- Silkscreen contact info and links to documentation



The back side of the PropBOE has a surface mount playground with access to all Propeller I/O pins and supply voltages as well as pads for any XBee and MicroSD connections that are not available on the top side.

The I/O pads alternate with pads that are routed to SOIC and MSOP IC footprints. In addition to convenient I/O connection to surface mount ICs, the I/O pads also have 1/10th inch spacing so that common a 1/10 inch SMT pad to socket header can be connected for prototyping access to the higher bank of I/O pins (P16..P31).

The serial header's pin map wouldn't fit on the top, so silkscreen for that is on the bottom layer. Note: The labels are not correct on this photo of an earlier engineering prototype.

PropBOE and Propeller Education

- The PropBOE hardware design will be open source
- The PropBOE will have a lifetime guarantee
- The PropBOE will kick off with a getting started web tutorial
- Additional educational resources and projects for the PropBOE will be community driven
- Languages: Spin initially, C language with multi-platform IDE support when ready

The PropBOE is going to be a fully open source design, and it'll also have a lifetime guarantee. When the board is released, it'll have a getting started web tutorial with how-to's for most of the board's features. We'll be inviting teachers who use the board to contribute material they use in their classroom on a new web site: learn.parallax.com.

Although initial support is in Spin language, the program will be switching to C language as soon as it's available. With the GCC compiler under way, we are currently evaluating options for making it easy to use for high school and Jr. High school students. C libraries can simplify the code to a PBASIC level, but we also need to look at an IDE with reduced buttons and settings.

Web Tutorial

- Beta test site is being migrated to learn.parallax.com
- The web site will go live about the same time that PropBOEs come into stock

Material was initially developed on a private site available to Propeller Educators Course attendees who test drive the hardware, objects, and tutorial.

Site is being migrated to learn.parallax.com, which with a planned opening to coincide with when PropBOEs come into stock.

Status Update

- Third revision prototype boards arrived
- Parts are on their way
- If testing 100%, will be out before Christmas

The status update is pretty simple: The third and final prototype revision printed circuit boards arrived, and parts are on their way. If the board tests 100% functional, we'll be able to start a production run and the boards should be available before Christmas. If not, our engineers will have to do another revision, and it'll probably be out early next year instead.

Web Tutorial Excerpts

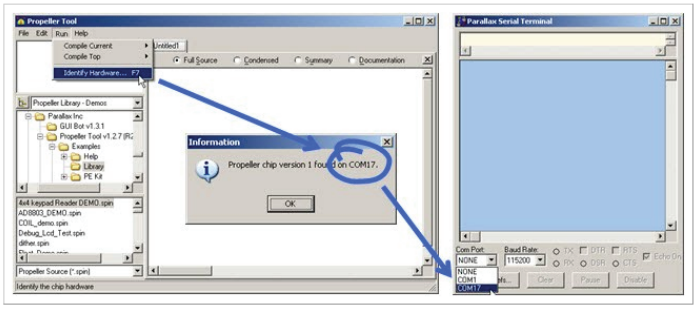
Here are screen captures from the PropBOE Tutorial and PropBOE-Bot Projects contents pages. In addition to the basics, the PropBOE-Tutorial has lots of projects posted that utilize its peripheral connectors, including servos, A/D and D/A in the Measure voltage lesson, speech synthesis with the stereo jack, wireless XBee communication, SD card data storage, and WAV file player.

The PropBOE-Bot tutorial is currently in progress. I'm just about done with navigation, and will be moving into navigation WITH sensors next.

Web Tutorial Excerpts

Configure the Software

- Click the Run menu and select Identify Hardware to find out what COM port your PropBOE is connected to.
- Set your Parallax Serial Terminal with your PropBOE's COM port.



NEXT

Web Tutorial Excerpts

Let's take a look at a few examples from out web tutorial. Here's an example from the software setup. Wherever possible, I tried to combine written instructions with pictures and diagrams. This particular step shows how to figure out what COM port your Propeller chip is connected to for the sake of setting the Parallax Serial Terminal to that same port.

Lesson 2: Led Light Control

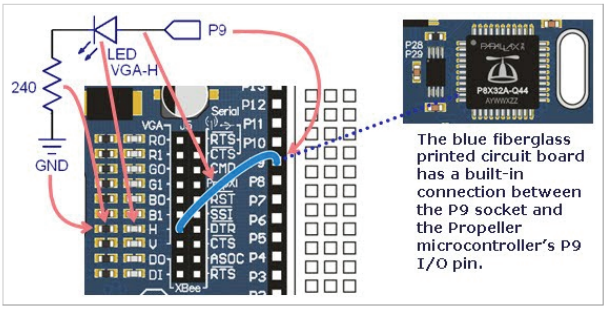
Blink a Light

The Propeller microcontroller has 32 input/output pins, abbreviated "I/O" pins. The Propeller chip can be programmed to set these I/O pins to certain output voltages and monitor for certain input voltages. (Voltage is abbreviated V.) In this first activity, we'll connect a light emitting diode circuit (abbreviated LED) to an I/O pin, and then write a program that makes the Propeller chip alternately supply 3.3 V and 0 V to the I/O pin. The result will be that the light in the LED circuit will alternately turn on and off.

Build a Test Circuit

It takes one wire to connect a Propeller I/O pin to one of the PropBOE's built-in LED circuits.

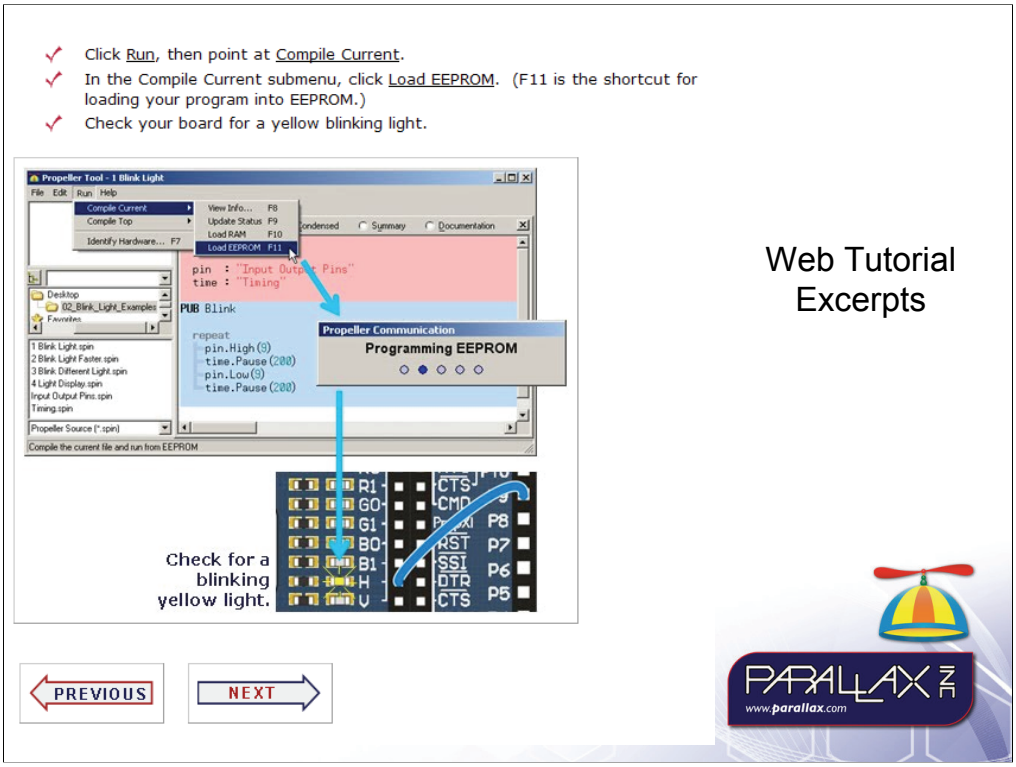
- ✓ Take a jumper wire and connect one end into the socket labeled P9, and plug the other end into the socket labeled H.



The figure shows how the components in the circuit schematic relate to the PropBOE. The LED is the tiny component just to the left of the H label, and to the left of the LED is a resistor with a value of 240 Ω. The ohm is a measure of how strongly the device resists current, and it's abbreviated with the Greek letter omega Ω.

Web Tutorial Excerpts

The blinking light lesson became interesting with the bank of VGA LEDs next to the Propeller I/O to Peripheral header. Note how the diagram shows relationships between the schematic and parts on the board as well as which pin on the Propeller connects to the P9 socket.



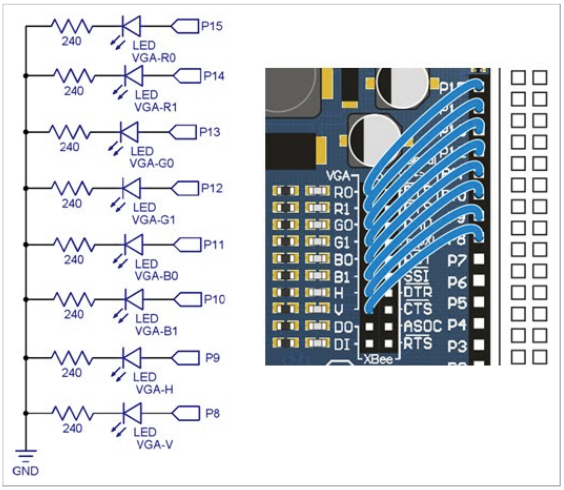
Here's an example of the light blinking code. I talked about the abstraction layer provided by the Input Output Pins and Timing objects during the January Propeller Meetup Group's Spin Tips section. So far, the approach has been well received by educators and beta testers.

Individual vs. Group I/O Operations

The Input Output Pins.spin object can also control multiple I/O pins and groups of I/O pins.

Modify the Circuit

✓ Add seven more jumper wires to build the circuit shown here:

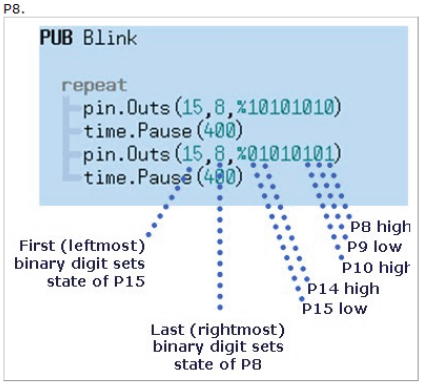


Web Tutorial Excerpts

It's nice to have ten pre-wired LEDs right next to the I/O pins; it sure made short work of this bank of eight indicator lights.

How the Spin Code Works

The Input Output Pins object's Outs method expects three numbers: the first pin number in the group of I/O pins, the last pin number, and a binary number that describes the high low pattern. In Spin, you indicate a binary number with a percent sign in front, like this: %01010101. Here's what that does to the high/low states of the group of I/O pins starting with P15 and ending with P8.



Web Tutorial Excerpts

Your Turn – A Different Pattern

Let's try turning 4 lights that are next to each other on, while the other four are off. Your binary values would have to be %11110000 and %00001111.

Try it.

PREVIOUS NEXT

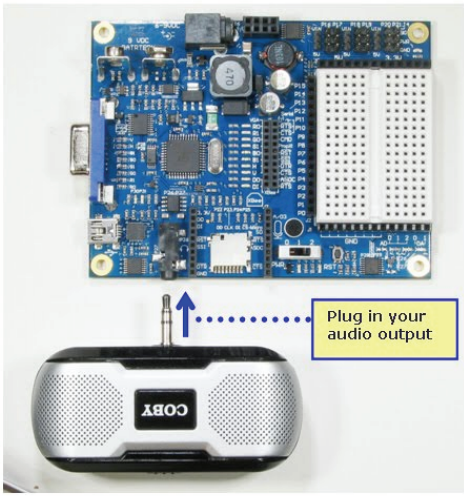
After introductions to objects, how to call their methods and how to read their documentation, the tutorial shows how to find a method for controlling groups of I/O pins, along with an example program that makes all eight lights blink in an alternating on/off pattern.

Circuit

This is one of the PropBOE's convenience features, instant audio without having to build any circuits. The circuits that allow the Propeller to generate audio are built right into the board.

Wiring

- ✓ Plug your audio device into the PropBOE's audio jack.
- ✓ If it requires batteries and on/off switch adjustment, do it now.



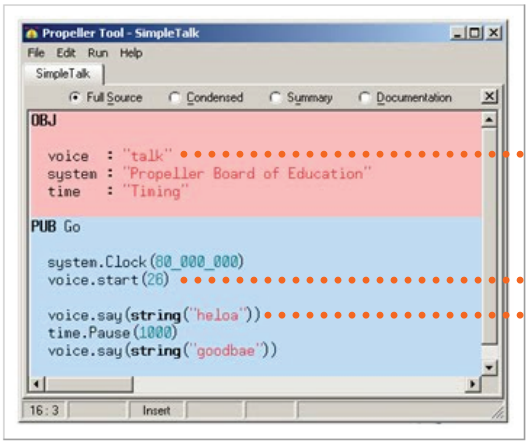
Web Tutorial Excerpts

Here's an excerpt from the start of a speech synthesis activity that comes a little later in the PropBOE tutorial.

Simple Talk

Simple Talk.spin passes strings of text that represent the phonemes in "hello" and "goodbye" to the talk object's say method.

- ✓ Download 06_Speech_Synthesis.zip.
- ✓ Unzip it to a folder.
- ✓ Remember that you have to open programs from within the folder, not within the zip.
- ✓ Run Simple Talk.spin and listen carefully.



Just add three lines of code and your application can talk!

What's really cool about object support for the multiprocessing Propeller is that getting it to say "Hello" with Phil Pilgrim's Talk object is about the same level of difficulty as blinking a light. Furthermore, since the Propeller DOES multi-process, you can add speech to just about any application with no additional integration steps.

- From the underside of the chassis, pull any excess servo and battery cable through the hole with the rubber grommet.
- Tuck the excess cable lengths between the servos and the chassis.
- Remove the L and R labels from the servo cables.

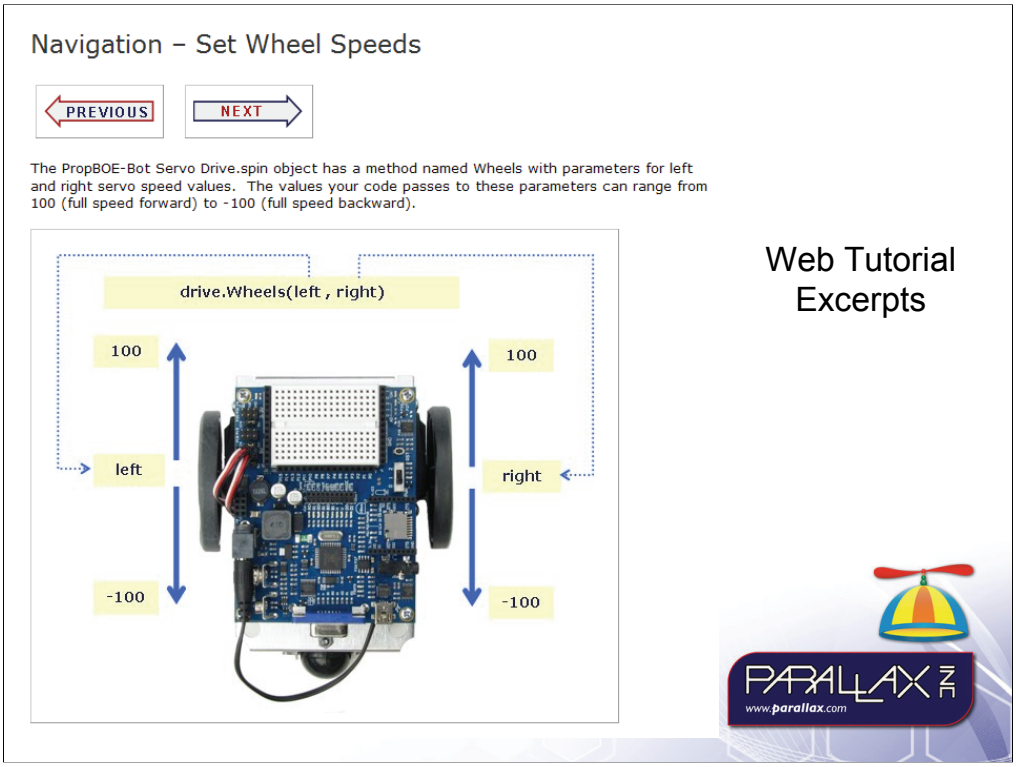
PREVIOUS NEXT

Web Tutorial Excerpts

PARALLAX INC
www.parallax.com

As you saw earlier, there's also a PropBOE-Bot tutorial. This excerpt includes the final assembly steps.

After assembly, we get to basic navigation, and this is it's contents page with links to example code and activities for each example.

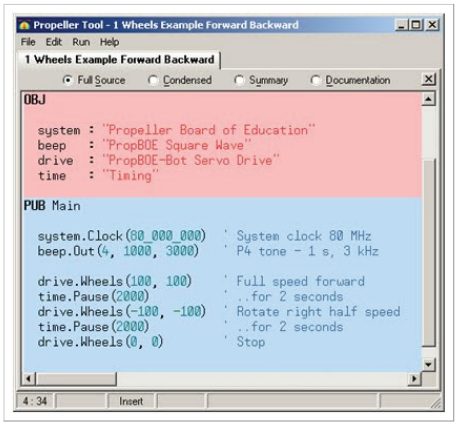


Here's a little diagram that kicks off discussion on how values passed to the PropBOE Servo Drive object's Wheels method affect wheel speeds and directions.

1 Wheels Example.spin

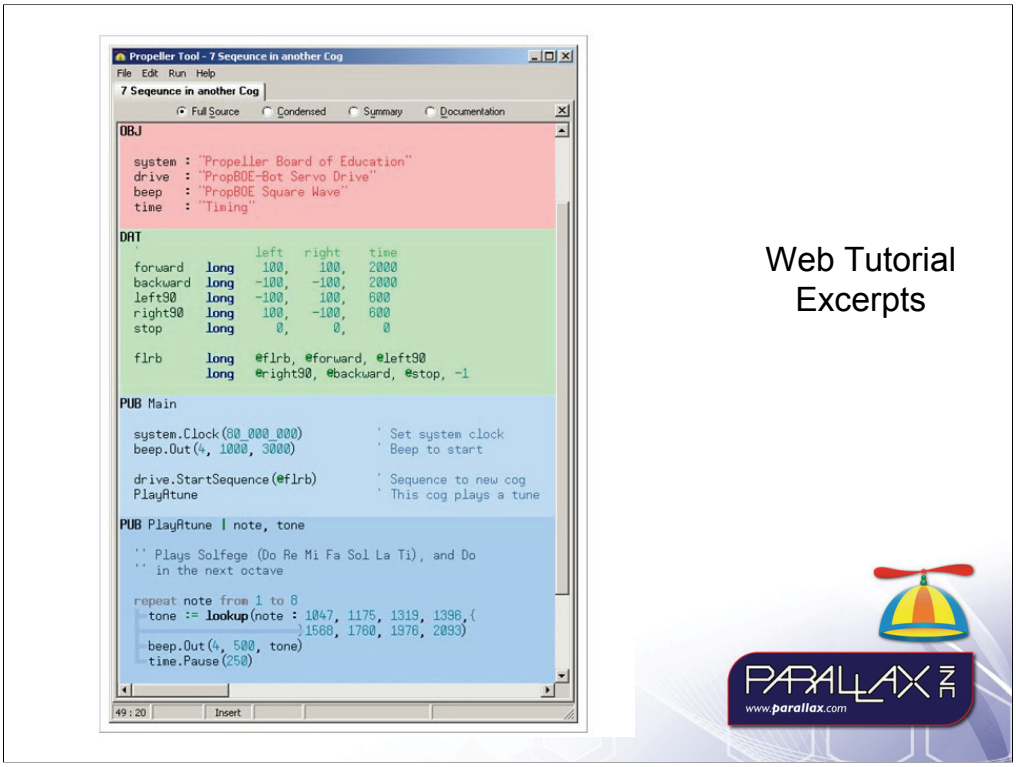
The first example program makes the PropBOE-Bot go full speed forward for two seconds, then full speed backward for two seconds, then stop.

- ✓ Open your PropBOE-Bot_Navigation folder (not the zip file).
- ✓ Double click the 1 Wheels Example.spin object to open it into the Propeller Tool.
- ✓ Use F11 to load the program into EEPROM.
- ✓ Verify that the PropBOE-Bot rolls forward for two seconds, then backward for two seconds.



Web Tutorial Excerpts

…and here is an example program that puts the concepts to use. With the servo control tended by another object that uses another cog to send repeated control pulses, all this program has to do is set the wheel speed, and then wait for the PropBOE-Bot to travel a certain distance before changing the wheel speed for the next maneuver. Of course, if it had sensors to check it could also do that during the maneuver, but that's part of a different lesson.



This example demonstrates how to use the PropBOE-Bot Servo Drive object's StartSequence method to assign another cog with performing a sequence of maneuvers. After the code passes the address of the sequence labeld flrb (forward, left, right, backward) to the StartSequence method, the cog executing this code calls the PlayAtune method, which plays some musical notes while the another cog walks the PropBOE-Bot through the sequence of maneuvers.

Questions and Comments

- My email address is alindsay@parallax.com
- Next up: Chip Gracey to talk about the Propeller 2

Here's my email address in case you have any questions or comments after the meeting.

Stick around; Chip Gracey is up next to talk about the Propeller 2.