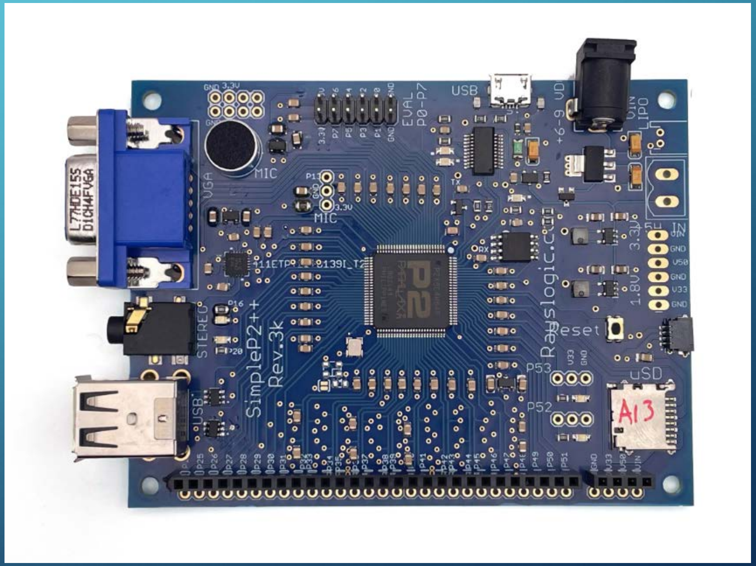
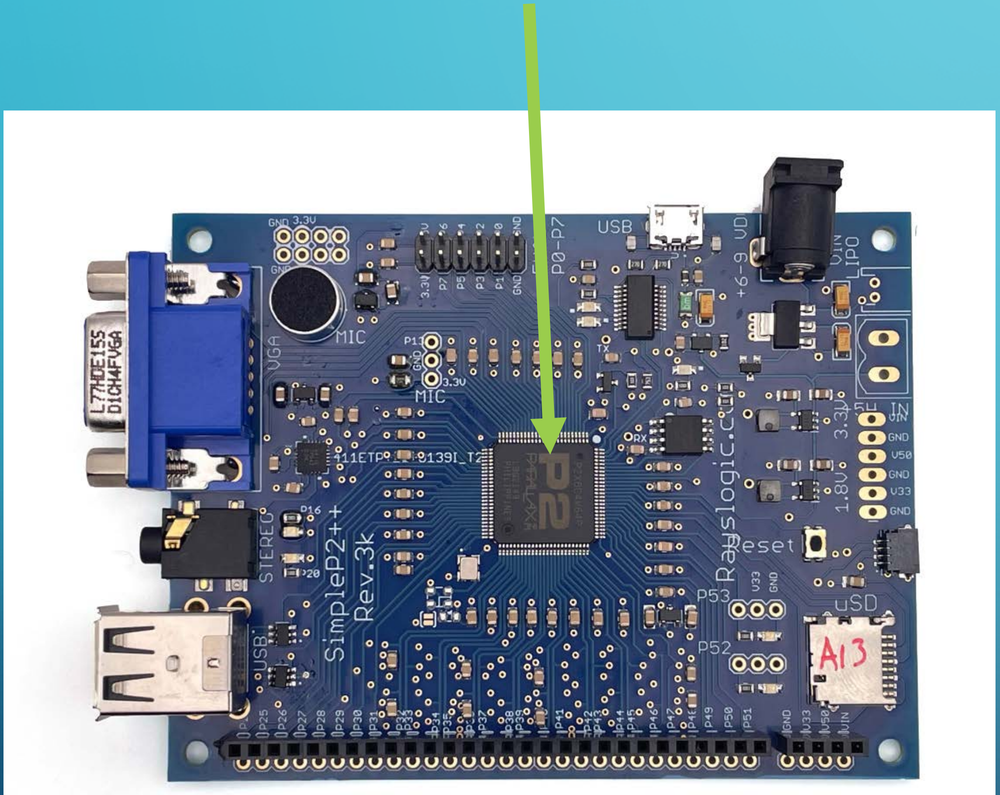
START CODING WITH THE SIMPLE P2++ AND SPIN2

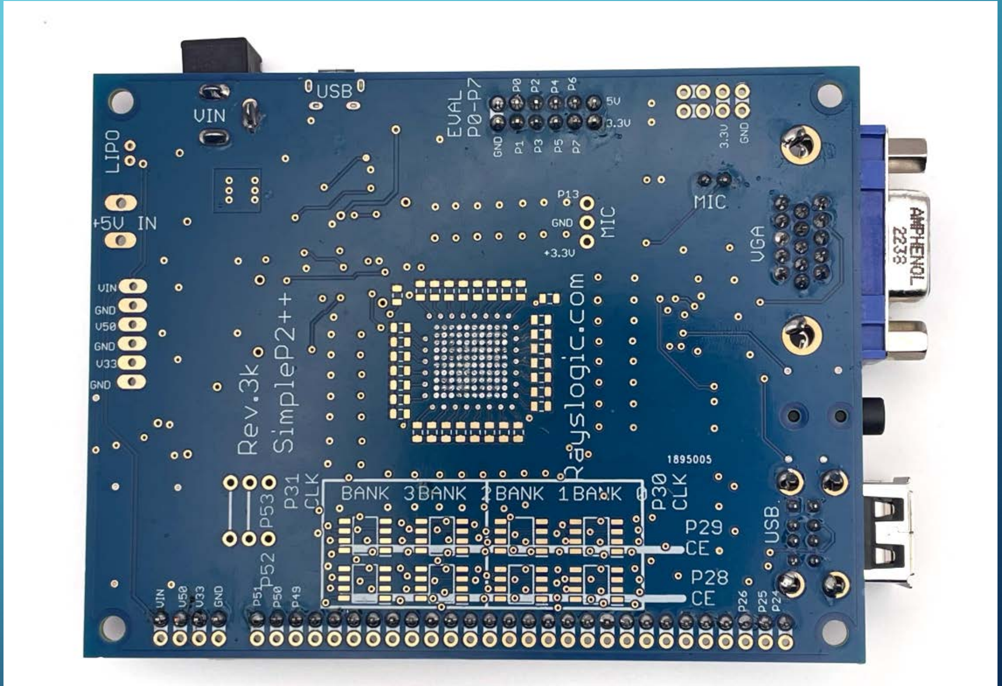
RAY ALLEN

RAY@RAYSLOGIC.COM



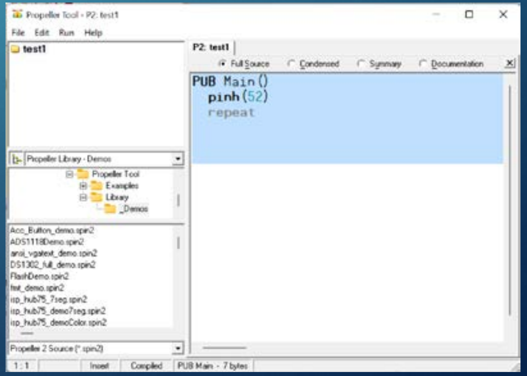
THE SIMPLE P2++ BOARD FEATURES THE PARALLAX PROPELLER II MICROCONTROLLER


Front View


Back View

Step 1: Download and Install the Propeller Tool Software for PC

- The Prop Tool software is a free download from the Parallax Website
  - Note: It is recommended to install the Prop Tool software before connecting the SimpleP2++ to a computer in order to ensure that the best FTDI USB serial driver is installed
- Note: If you don't have a Windows PC or don't like the Prop Tool, there are now some other great options:
  - Spin Tools IDE download latest release
    - Looks and acts a lot like the Prop Tool, but works on Mac and Linux
  - FlexProp IDE download latest release
    - A more basic editor, without the Spin2 background colors
    - Also supports the C and Basic languages, in addition to Spin2



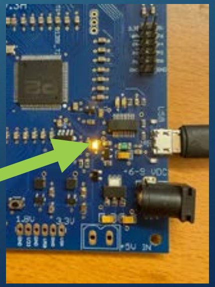
```
PUB Main()
  pinh(52)
  repeat
```

Acc_Button_demo.spin2
ADS1118Demo.spin2
ansi_vgatext_demo.spin2
DS1302_full_demo.spin2
FlashDemo.spin2
hst_demo.spin2
isp_hub75_7seg.spin2
isp_hub75_demo7seg.spin2
isp_hub75_demoColor.spin2

STEP 2: CONNECT PC TO MICRO-USB PORT ON SIMPLEP2++ BOARD

- This step requires one to have a USB data cable with one end being a Micro-B type plug and the other end pluggable into computer (usually USB Type A)
- If everything is good, the red USB serial LED should flash momentarily and then the yellow power LED should stay lit
  - At this point the PC is providing up to 500 mA of +5 VDC power to the SimpleP2+ board
  - If the yellow LED does not light check that the cable is a data cable and not just a charge cable also make sure that the Propeller Tool software was installed (so that the FTDI driver was installed)
  - If still won't work try a different cable (usually the longer ones are data cables) and/or a different computer
  - If still stuck here you may have a hardware issue with your SimpleP2+ board, email ray@rayslogic.com for support

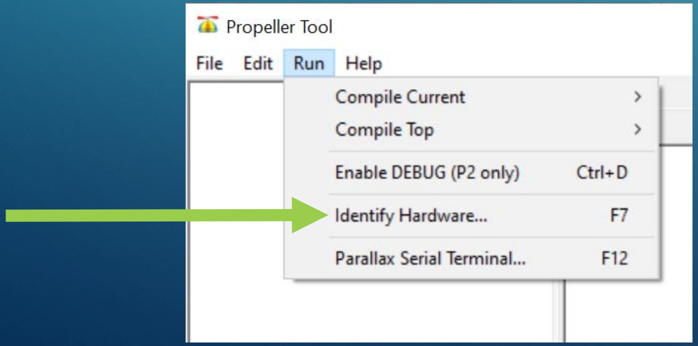
USB Type A Plug

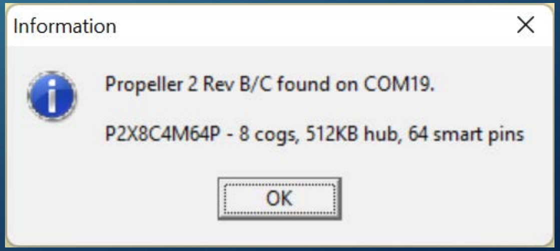


USB plug images from Wikipedia

STEP 3: IDENTIFY YOUR P2 WITH THE PROPELLER TOOL SOFTWARE

- The "Identify Hardware" feature of the Prop Tool is a great way to make sure communications and hardware are in good shape
- From the Prop Tool menu, select "Run"→"Identify Hardware" (or press F7 on keyboard)
- If everything is good, you will get the "Information" window shown below
  - The specific COM port will likely be different, this is picked automatically by the FTDI driver
  - Note that a COM port is how a PC communicates with a serial device. The FTDI driver acts like a COM port and then sends serial traffic over USB cable to the FTDI USB chip on the SimpleP2++ board. The FTDI chip then sends this data to/from the Propeller serial interface pins (P62 and P63)
  - The Red and Green USB activity LEDs will flash during the identify process. The Red LED is lit when data is being sent from P2 to PC and the Green LED is lit when data is sent from PC to P2




Information

Propeller 2 Rev B/C found on COM19.

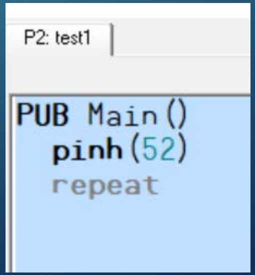
P2X8C4M64P - 8 cogs, 512KB hub, 64 smart pins

STEP 4: WRITE YOUR FIRST SPIN2 PROGRAM

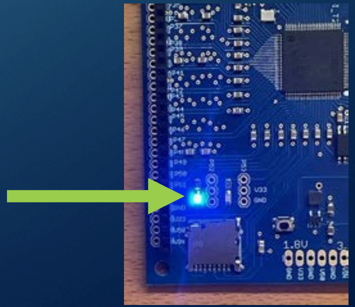
- In the Prop Tool, we code in a language called Spin2. This language has special instructions specifically designed for the P2 microcontroller. Spin2 is a lot simpler than the most common microcontroller language, C, with similar complexity as the BASIC language.
  - Note that Spin2 is not case sensitive and so upper and lower case characters are treated the same
  - Note that, like the computer language PYTHON, the indentation or leading spaces on each line is important
- When started, the Prop Tool should give you an empty white text window for code. Enter in this tiny code, noting the two space indent on second and third lines:
- Save the file to a folder, such as C:\Propeller2, making sure to select to save with ".spin2" extension
- The code should appear exactly as shown below with a "P2:" in front of the file name indicating a .Spin2 file extension and with "pinh" in dark black, indicating that it is recognized as a Spin2 keyword
- Finally, program this code into the P2 using the "Run"→"Compile Current"→"Load RAM" (or press F10)
- The P52 blue LED should now be lit on the SimpleP2++ board
- Congrats! You just programmed the P2.

```
PUB Main()
  pinh(52)
  repeat
```

If having problems, download the program here


P2: test1

```
PUB Main()
  pinh(52)
  repeat
```



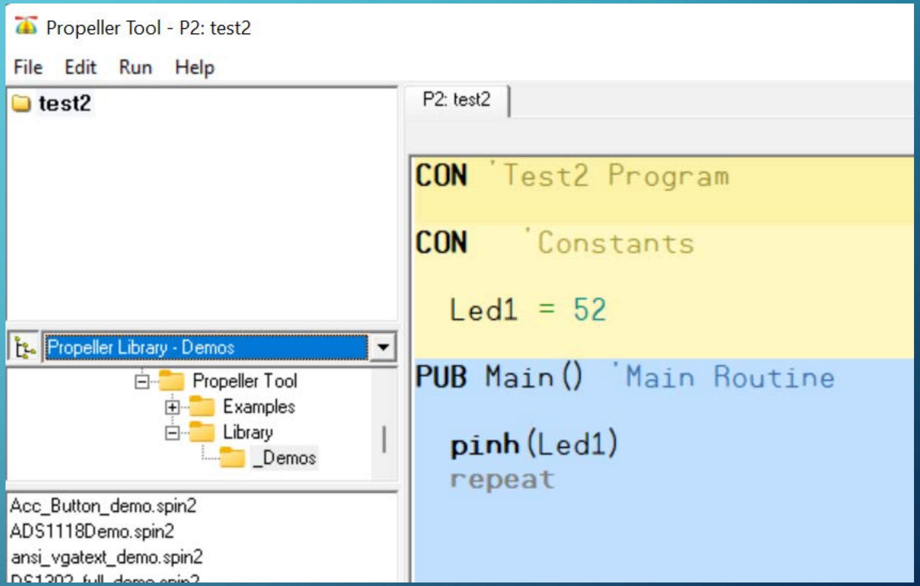
STEP 5: WHAT DID WE JUST PROGRAM?

```
PUB Main()
  pinh(52)
  repeat
```

- So how does this code work?
  - First, code goes into either PUB (short for public) or PRI (short for private) method block. Every top level program must have at least one PUB method so the program knows where to start.
    - The first PUB method in the file is the main method that starts first, regardless of the name of the method.
    - We could have renamed "Main" to "Start" and have the same result.
    - PUB method names must end in parenthesis to show if there are any parameters being passed into the routine.
    - The top level main method cannot get any parameters passed into it, so these parenthesis are always empty for the case of the main method.
    - Note that the "PUB" block line can have no indentation.
    - PUB method blocks have a bluish background color. Other types of blocks have different colors.
  - Next, the second line is indented just for clarity. Indentation is important though. Still, we don't need to indent the second and third lines here, it just looks better this way.
  - "PINH" is a built in method to drive an I/O pin into the high state. Usually, this means outputting +3.3 V. The "(52)" is saying to drive I/O pin #52 high. We happen to have a blue LED there, so it gets lit.
  - The "repeat" line (with no more lines under it) causes the program to stop. It is always highly recommended to have the main routine stop with a "repeat" line or some other form of infinite loop, otherwise the behavior is unpredictable.

STEP 6: ADD CON BLOCK AND COMMENTS

- Almost all Spin2 programs start with a CON (constant) block. These show up with a yellow background
  - The first CON block is usually used for comments that describe the program and how to use it and doesn't have any actual constants
  - See how this code is using a second CON section to define a constant to represent the pin to light, Led1.
  - Note how there is a slight change in background color between CON sections, this helps separate them and happens for all block types
- It's a good idea to add a lot of comments to your code so that others can understand it and also for yourself when you come back to it after a long time
  - A comment in Spin2 begins with a ' character
  - For example, here "Test2 Program" is a comment
  - The comment can start anywhere on a line
- Update the test1.spin2 code to be as shown here on the right, save as test2.spin2 and try running it
  - Try changing the value Led1 from 52 to 53 and see how the other blue LED is lit
  - This code should produce the exact same result as test1.spin2



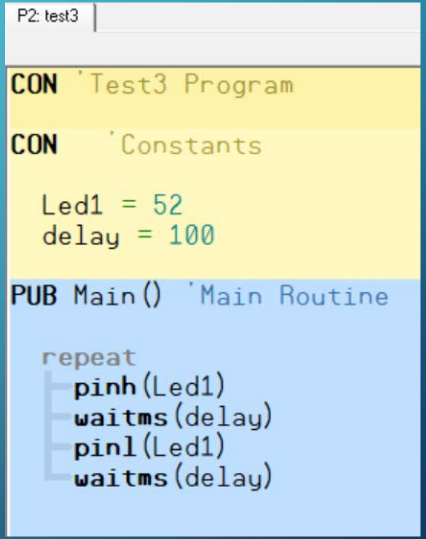
```
CON 'Test2 Program

CON    'Constants

  Led1 = 52

PUB Main() 'Main Routine

  pinh(Led1)
  repeat
```

If having problems, download the program here

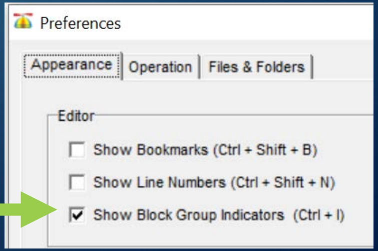
STEP 7: ADD DELAYS AND INFINITE LOOP TO BLINK THE LED

- The "repeat" method is how we have pieces of code repeat, either finite with lines like "repeat 10" to repeat 10 times, or by itself to have an infinite loop that repeats forever
- Delays are often needed in Spin2 codes. One usual way to cause a delay is with the waitms() method. This tells the P2 to stop for the given number of milliseconds.
  - Note that your heart beats at a rate of about 1000 ms (1 second)
  - The human response time is about 100 ms
  - Blinking the LED between 100 and 1000 ms is a good place to start
- The built in method "PinL()" does the opposite of PinH() and drives the I/O pin to the low state, usually ground
- Update the code to look like that on the right here and save as test3.spin2
- Note the faint lines under "repeat" show the indentation. This is called "Block Group Indicators" and it is highly recommended to have this turned on under Edit→Preferences
- Run the program and see how the LED is flashing in an infinite loop
  - Try changing the value of the constant "delay"

If having problems, download the program here



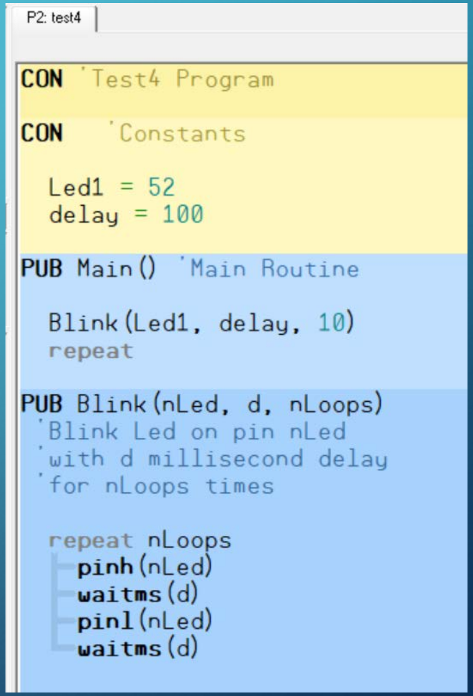
```
CON 'Test3 Program

CON   'Constants

  Led1 = 52
  delay = 100

PUB Main() 'Main Routine

  repeat
    pinh(Led1)
    waitms(delay)
    pinl(Led1)
    waitms(delay)
```



- [ ] Show Bookmarks (Ctrl + Shift + B)
- [ ] Show Line Numbers (Ctrl + Shift + N)
- [✓] Show Block Group Indicators (Ctrl + I)

STEP 8: CALLING YOUR OTHER CODED METHODS

- Usually the majority is a code is not in the main method, but in sub methods that are called by the main method
- Here we will demonstrate putting the code into a sub method and then calling it with parameters
- Adjust the previous code to look like this and then save as test4.spin2
- Here we call the method "Blink" and pass three parameters that tell it which pin to blink, what delay to use, and how many repeats to do
- This code will blink the P52 Led 10 times and then stop
- Experiment with other pins, delays, and loop counts
  - There are also Blue LEDs on P53, P16, and P20

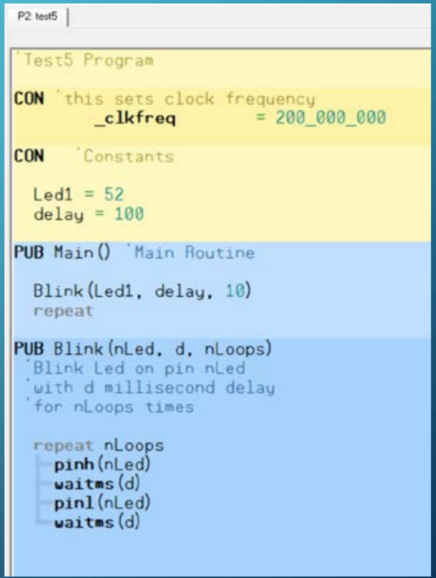


```
CON 'Test4 Program

CON   'Constants

  Led1 = 52
  delay = 100

PUB Main() 'Main Routine

  Blink(Led1, delay, 10)
  repeat

PUB Blink(nLed, d, nLoops)
 'Blink Led on pin nLed
 'with d millisecond delay
 'for nLoops times

  repeat nLoops
    pinh(nLed)
    waitms(d)
    pinl(nLed)
    waitms(d)
```

If having problems, download the program here

STEP 9: SETTING THE SYSTEM CLOCK AND USING OBJECTS

- The P2 MCU is a synchronous chip, meaning that things only change on the rising edge of the P2 system clock
- The faster the clock, the faster your code will run
  - Note that waitms() take the system clock speed into account, so will always be right
- So far, we've been using the default clock, RCFAST, at ~ 20 MHz (20 million cycles per second). But, the system clock can run much faster. A normal speed is around 180 MHz. Anything over 300 MHz is considered "over clocking".
- There is a special constant, _clkfreq, that one sets to define the system clock speed
  - When your code is compiled a section is added to the beginning of it to set the clock when program is first started
- Add the clock setting CON section to your code as shown on the right, save as test5.spin2 and run it.
  - Code will work the same way as before because nothing here depends too much on the clock speed. Program will be a hair slower, but you won't notice because the delays are much longer that the time to run this code and the waitms() method takes the clock speed into account.

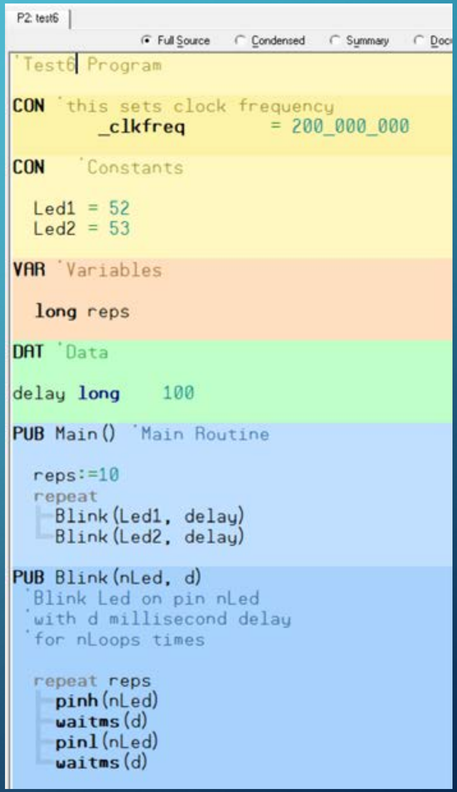


```
'Test5 Program

CON 'this sets clock frequency
        _clkfreq          = 200_000_000

CON   'Constants

  Led1 = 52
  delay = 100

PUB Main() 'Main Routine

  Blink(Led1, delay, 10)
  repeat

PUB Blink(nLed, d, nLoops)
 'Blink Led on pin nLed
 'with d millisecond delay
 'for nLoops times

  repeat nLoops
    pinh(nLed)
    waitms(d)
    pinl(nLed)
    waitms(d)
```

If having problems, download the program here

STEP 10: USING VAR AND DAT BLOCKS

- VAR (short for variable) blocks are where one can define variables that be used anywhere in the .spin2 file where they are declared
  - One use for a VAR block variable definition is to avoid having to pass parameters to sub methods
  - Especially useful if would otherwise have to pass the parameter to a lot of places
- DAT (short for data) blocks are a good place to things that you want to start out with some value
  - Like VAR variables, DAT variables are available for use anywhere in the file
- Edit the previous code to look like this →
  - Here we use a VAR section to declare a long variable, "reps", and we use a DAT section to define the long variable, "Delay", and have it initialized to a value of 100
  - Note that signed "longs" (four bytes) are the usual thing to use in Spin2, but "words" (two bytes) and "bytes" (one byte) can also be used to save memory



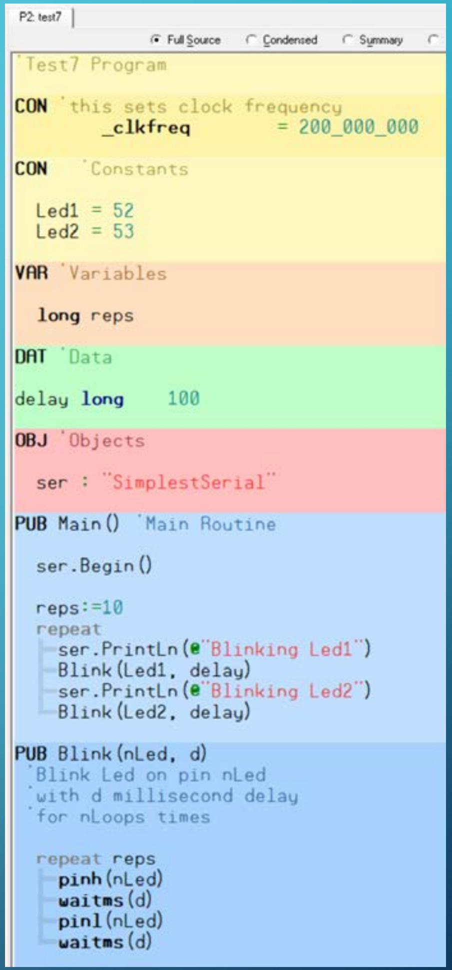
```
'Test6 Program

CON 'this sets clock frequency
        _clkfreq          = 200_000_000

CON   'Constants

  Led1 = 52
  Led2 = 53

VAR 'Variables

  long reps

DAT 'Data

delay long    100

PUB Main()  'Main Routine

  reps:=10
  repeat
    Blink(Led1, delay)
    Blink(Led2, delay)

PUB Blink(nLed, d)
 'Blink Led on pin nLed
 'with d millisecond delay
 'for nLoops times

  repeat reps
    pinh(nLed)
    waitms(d)
    pinl(nLed)
    waitms(d)
```

If having problems, download the program here

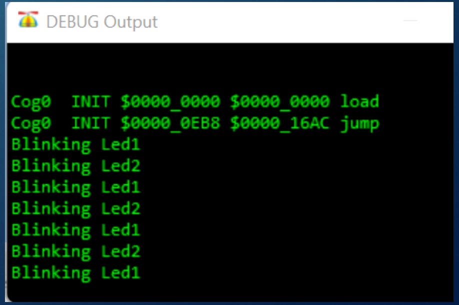
STEP 11: USING OBJ BLOCKS

- In this final step we will use an OBJ (object) block to output serial messages.
  - In Spin2, an object is another .spin2 file that we want to include in this program.
  - Objects make it easier to do things without writing new code yourself
- The "SimplestSerial" object does not come with the Prop Tool at this time, so best way to get it is to download the code using the link on the bottom of this page and save to a folder, such as C:\Propeller2, and then open Test7.spin2
  - This file was created with the Prop Tool's "Archive" feature that saves the main code and all the sub-objects into one .zip file, this can be very convenient
- Here we are adding the SimplestSerial object with the line in the Object block and giving it the name "ser", although could be anything
  - SimplestSerial uses some special Smart Pin features of P2 and so needs to be initialized with the ser.Begin()
- Here we are printing two text lines in the loop with ser.PrintLn()
  - We start a string with the @ symbol and then include the text inside parenthesis
- There are several other serial objects one can use, some are included in the Prop Tool Library
  - The special thing about this one is that it can use the "Debug" window as a serial terminal
  - Make sure this is enabled under "Run"→"Enable Debug (P2 Only)" and the debug windows should open when the program starts and display the text

Download this program archive here



```
'Test7 Program

CON 'this sets clock frequency
        _clkfreq          = 200_000_000

CON   'Constants

  Led1 = 52
  Led2 = 53

VAR 'Variables

  long reps

DAT 'Data

delay long    100

OBJ 'Objects

  ser : "SimplestSerial"

PUB Main() 'Main Routine

  ser.Begin()

  reps:=10
  repeat
    ser.PrintLn(@"Blinking Led1")
    Blink(Led1, delay)
    ser.PrintLn(@"Blinking Led2")
    Blink(Led2, delay)

PUB Blink(nLed, d)
  'Blink Led on pin nLed
  'with d millisecond delay
  'for nLoops times

  repeat reps
    pinh(nLed)
    waitms(d)
    pinl(nLed)
    waitms(d)
```



```
Cog0  INIT $0000_0000 $0000_0000 load
Cog0  INIT $0000_0EB8 $0000_16AC jump
Blinking Led1
Blinking Led2
Blinking Led1
Blinking Led2
Blinking Led1
Blinking Led2
Blinking Led1
```

NEXT STEPS?

- Look through the Spin2 and Propeller II documentation
- Visit the product page at rayslogic.com for more info and example code
- Explore the “Propeller Library – Demos” using the left side of the Prop Tool window
- Visit the OBEX (Object Exchange) where you can find Spin2 objects to do various things
- Email ray@rayslogic.com for SimpleP2++ board questions
- Visit the P2 forum where you can see and ask questions about the Propeller II and Spin2