

This series of exercises introduces the microcontroller and some of the things you can do with it.

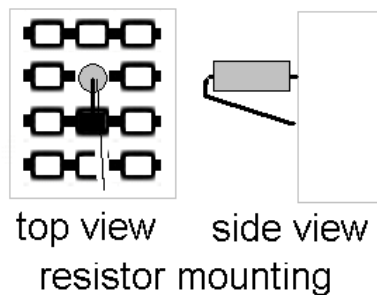
The microcontroller comprises a microprocessor, memory, and input/output (I/O) capability all on a single chip. The one we'll be using is the Parallax BASIC Stamp2 (BS2). This device is an integrated microcontroller system on a small carrier board that has the microcontroller chip plus auxiliary chips to enhance the memory and to provide a compatible interface to a PC. The PC runs the "BASIC Stamp Editor" that allows you to write a program in the PBASIC language and then download it to the BS2 where the program runs.

The Basic Stamp has 16 I/O lines or pins that can be set or read under program control. As an output the pin can be set to a logic low or 0 state (~0 volts) or logic hi or 1 (~5 volts). As an input it will be read as 0 or low when held at 0 volts by external circuitry (designated at Vss) and as 1 or high when held at 5 volts (designated at Vdd).

The setup we are using is made by Parallax Inc. and is called the "Board of Education" (BOE). Its main components are a serial connector that goes to the PC and a power connector on the left edge of the board, the BS2, a power switch at the bottom center, and at the right a small breadboard with a connecting strip at its immediate left for the 16 I/O pins and a strip immediately above for power connections.

Each exercise requires two steps: wiring up the external circuitry and writing the program.

You've already had some experience wiring circuits on breadboards. For these exercises you will be directed to wire the circuit at a specific location on the board. The breadboard is small and you will be adding components to those already there as you progress through each exercise. So look carefully at the breadboard layout and position the components exactly as shown. When you mount resistors on the board, mount them vertically, as shown below.



Writing programs is not hard. A program consists of a series of lines of text, each of which instructs the microcontroller to perform a specific function. A line of code can perform an input or output operation, assign a number to a variable, perform a mathematical or logical operation, change or repeat a sequence of instructions.

Because of the limited processing speed and memory available in a microcontroller, the mathematical operations are limited to integers.

For the division operator (/)

107/12 returns 8 (not 8.916 . .) 11/3 returns 3 (not 3.666..)

However the math includes the remainder operator (//)

107//12 returns 11 11//3 returns 2

A programming language has a tightly defined syntax that requires a line of code to have a specific form. Partial or complete code is provided for the experiments. When you change or add code to a program it is a good idea to use the File|Save As command to save it with a new name.

EXERCISE a - flash an LED

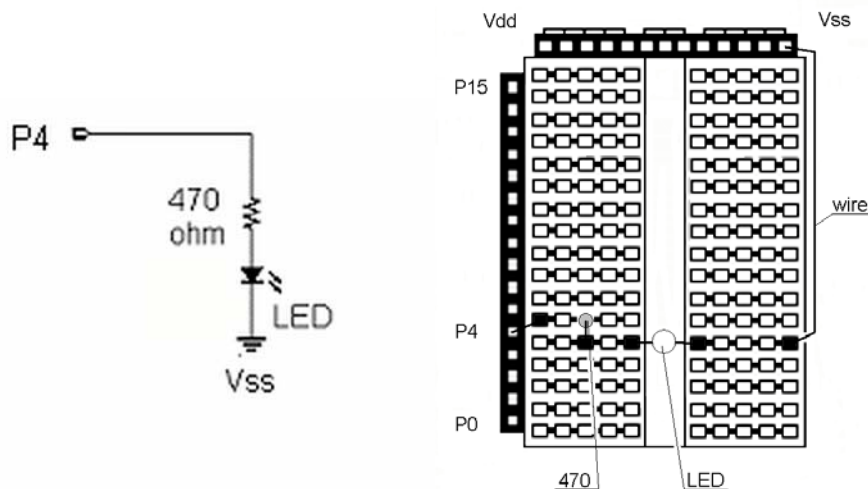
The first exercise is simply to connect an LED to one of the pins and write a program to make it flash on and off.

Wire:

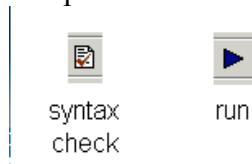
Make sure the power switch at the bottom center of the BOE is in the 0 (off) position. The green LED near the top center of the board should be off. Always turn off the board before making any wiring changes.

(NOTE: the A/D chip for the last exercise may already be installed at the top of the board, but not wired, ignore it for the moment.)

Install a red LED in the fifth row from the bottom of the board straddling the groove with the short lead to the right. Connect a jumper from the right side of the led to Vss. Connect a 470 ohm (anything from 470 to 560 ohms will work) resistor from the fifth row to the sixth row as shown, and a jumper from P4 to the sixth row.

**Program:**

Start the BASIC Stamp Editor. Using the folder tree at the left of the edit window, locate and open the pchem folder and double click the flash.bs2 file. On the Run menu click syntax check. This checks the structure of the program. On the Run menu click run to download the program to the BS2. The program starts running when the download is complete. Shortcuts to these two operations are on the toolbar.



Some notes on the program:

1) all text following a single quote (') is ignored by the compiler. This is used to enter comments in the code. Blank lines are also ignored.

(The following two special "comments" (called compiler directives) tell the compiler which chip and which variant of PBASIC are being used. They must appear before any code.)

```
' {$STAMP BS2}  
' {$PBASIC 2.5}
```

2) "flash:" is a label for a point that can be referenced by certain commands.

3) The "LOW 4" command makes P4 an output and sets it equal to 0 volts (Vss). This will turn off the LED.

4) The "HIGH 4" command makes P4 an output and sets it equal to 5 volts (Vdd). This will turn on the LED.

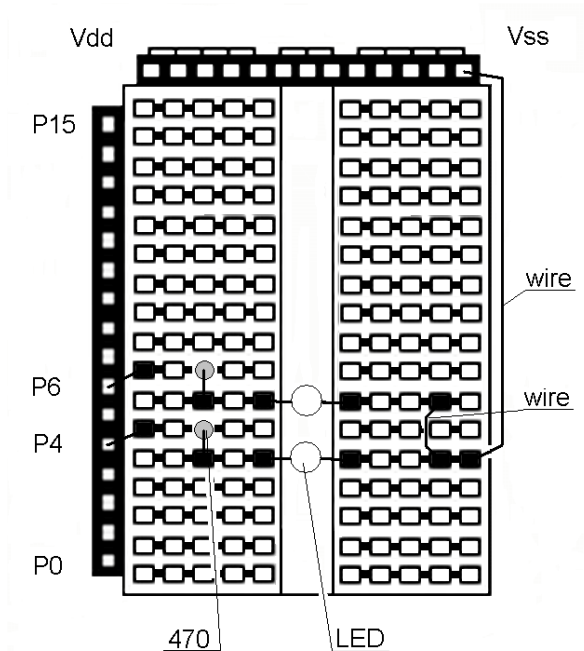
5) The "PAUSE 1000" command causes the processor to wait one second (1000 msec) before proceeding to the next command.

6) The "GOTO flash" command tells the program to find the label "flash:" and continue from there.

EXERCISE b - flash 2 LEDs

Wire:

Connect a second (green) LED to P6 in the same manner as above.



Program:

Modify the program to alternately flash the 2 LEDs.

(alternately flash: as in a railroad grade crossing signal)

Save this program as flash2.bs2

Modify the program so that the red LED is on twice as long as the green one.

Save this program as flash3.bs2

Turn off the BOE board. Carefully disconnect the black cable that runs to the laptop from the left side of the board. Turn the BOE on. The program should run just as before. Once the program has been downloaded from the PC it remains in the chip's memory and runs whenever the chip is powered up.

Turn off the BOE and reconnect the cable from the computer.

EXERCISE c - send data to the Debug screen

Program:

PBASIC has a command, "DEBUG", that can be used to send text and numbers to the PC for display in a DEBUG terminal window on the PC. (This simulates a display unit that could be attached to one of the I/O pins.)

Modify the 2 LED program to display on the DEBUG screen which LED is on.

Insert the program line

```
DEBUG "red "
```

in the part of the code that turns on the red led, and the line

```
DEBUG "green "
```

in the part of the code that turns on the green led.

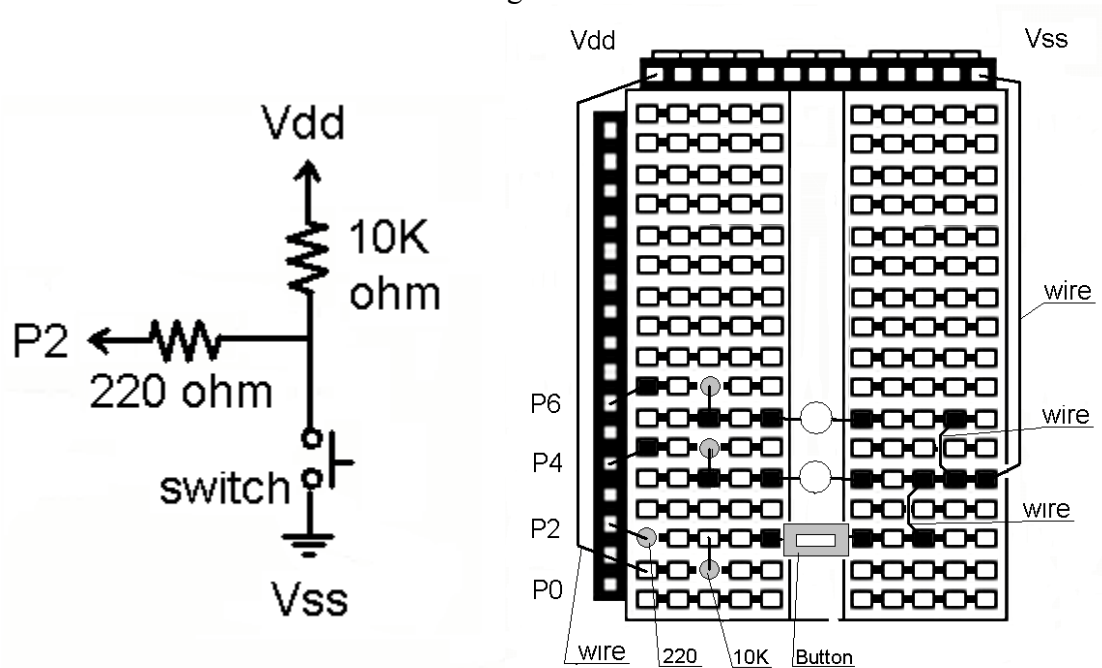
Save this program as flash4.bs2

Leave the LEDs and resistors connected.

EXERCISE d - reading a switch

In this section you will add a switch to the board and connect it to the microcontroller.

Wire a switch to P2 as shown in the diagrams.



When the program sets P2 as an input it does not draw any (significant) current. As a result the potential at P2 will be essentially 5 volts (Vdd) when the switch is open and 0 volts (Vss) when the switch is closed.

Program:

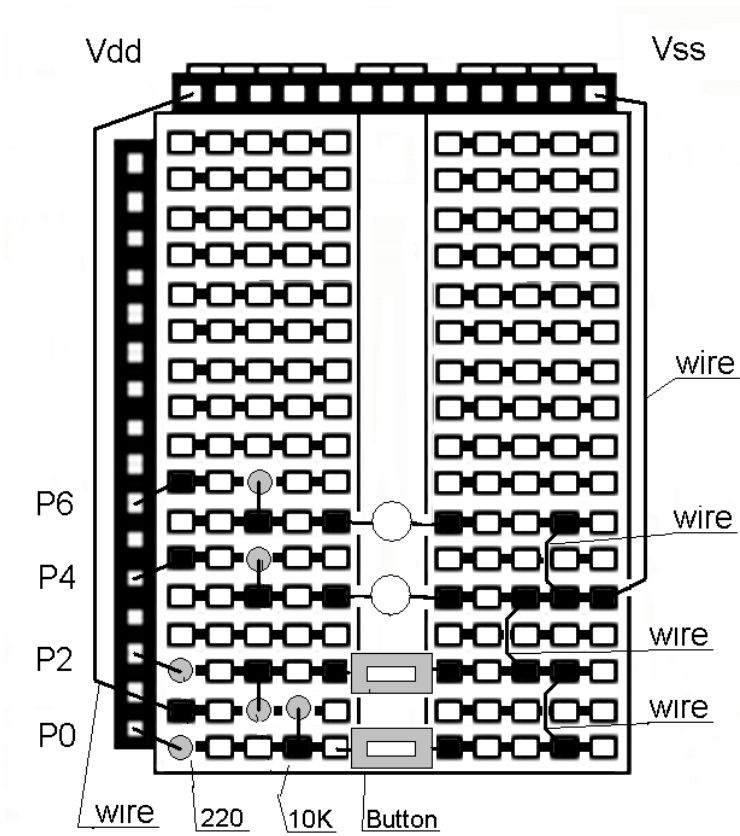
Load the program ezbttn2.bs2

This program illustrates the IF – THEN – ELSE command, a programming construct that allows one of two different sets of code to be executed depending on the result of a comparison. The form is:

```
IF conditional test THEN
  Code block 1
ELSE
  Code block 2
ENDIF
Code block 3
```

If the conditional test is true, code block 1 is executed followed by block 3.
If the conditional test is false, code block 2 is executed followed by block 3.

When you are finished, add a second button connected to P0 as shown in the following diagram. We'll use this later for a demonstration.

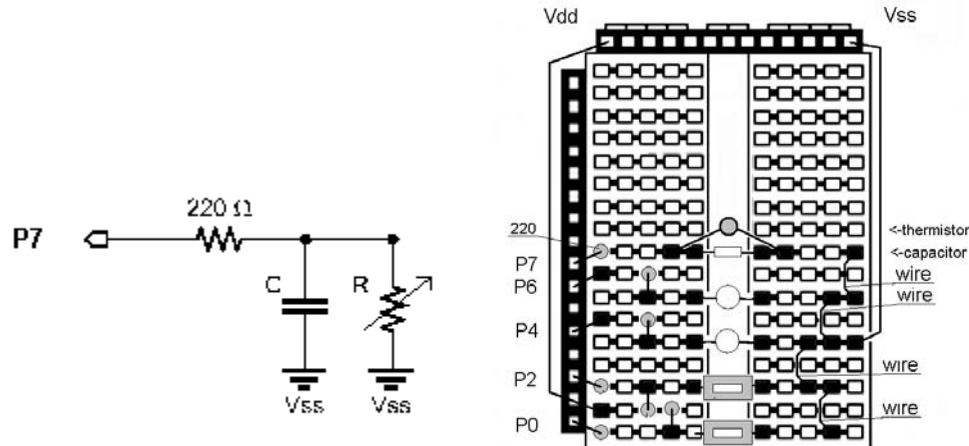


EXERCISE e - "pseudo" analog to digital conversion

If your sensor is resistive (thermistor) a simple way to measure changes in its value is to connect it in parallel with a capacitor, charge the capacitor, and time how long it takes for the capacitor to discharge.

Wire:

Connect a thermistor (R), 220 ohm resistor, and 0.1 uF capacitor (C) to P7.



Program:

Open the program RCTemp.BS2

PBASIC has a command, RCTIME, that returns a count proportional $R \times C$, so if C is fixed the count is proportional to R of the thermistor.

Illustrated here is the requirement to define a variable called "result" before using it to store a number and telling the compiler how much memory to allocate to it, in this case 16 bits (word). Also assigning the designation of pin 7 (P7) to RC. This is good practice if you end up with a program that refers to pin 7 in multiple lines of code and then find out that you would rather use pin 11. You then only have to change one line in your program.

What are the readings for air temperature and your skin temperature? (Use one of the lab digital thermometers to get air and skin temperatures.)

| source | counts | temperature |
|---------------|--------|-------------|
| Room air | | |
| Your skin | | |
| Partners skin | | |

How would you calibrate this digital thermometer?

Modify this program to turn the green LED on (and the red one off) when the sensor is somewhat above room temperature or less and turn the red LED on (and the green one off) when the temperature is definitely above room temperature. (Pick a break point about halfway between room temperature and your skin temperature.)

Use the IF-THEN-ELSE code structure that was illustrated in the button exercise

```
IF conditional test THEN
    (code1)
ELSE
    (code2)
ENDIF
```

“Conditional test” can have several different forms.

$a=b$, $a>b$, $a<b$, $a\neq b$ (not equal) where a and b may be variables (like “result”), numbers (235), or a function (e.g. IN2) that evaluates to a number.

Save this program as LEDTemp.BS2.

Stop at this point and get the lab coordinator or TA for a demonstration using the parts you’ve assembled so far. We’ll use the set up as a simple oven controller.

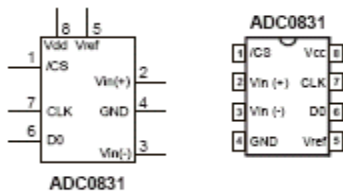
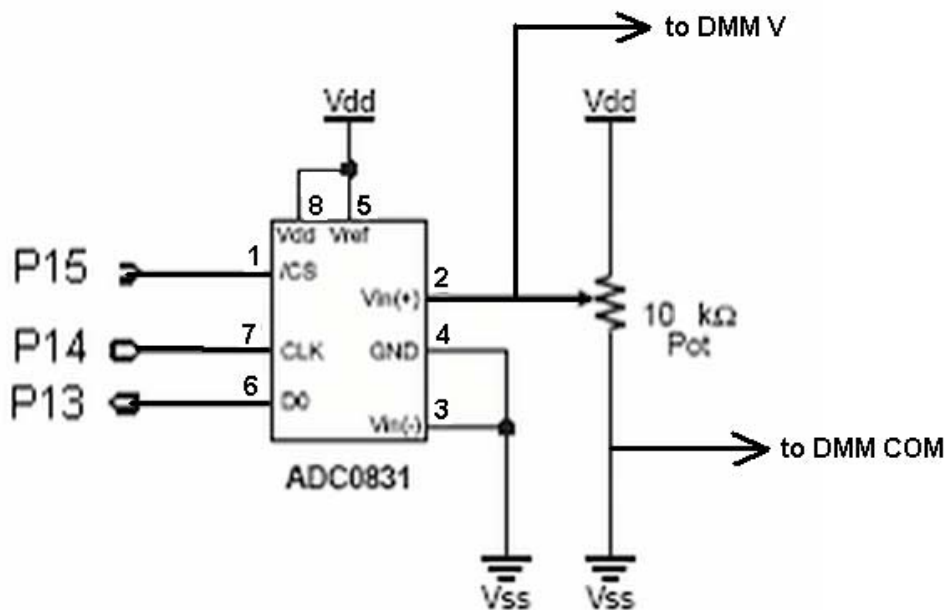
EXERCISE f - voltage measurement with an analog-to-digital converter (ADC0831 chip)

The ADC0831 is an 8-bit A/D converter on a chip with a serial interface to a microcontroller. For the setup as shown below, the input range is 0 to +5 volts. The output is an integer between 0 and 255 (decimal) or 00000000 to 11111111 (binary).

Wire:

At this point we have a lot mounted on the breadboard and space is getting tight.

Here is the schematic of the circuit:



ADC0831 Circuit Symbol and Pin Map.

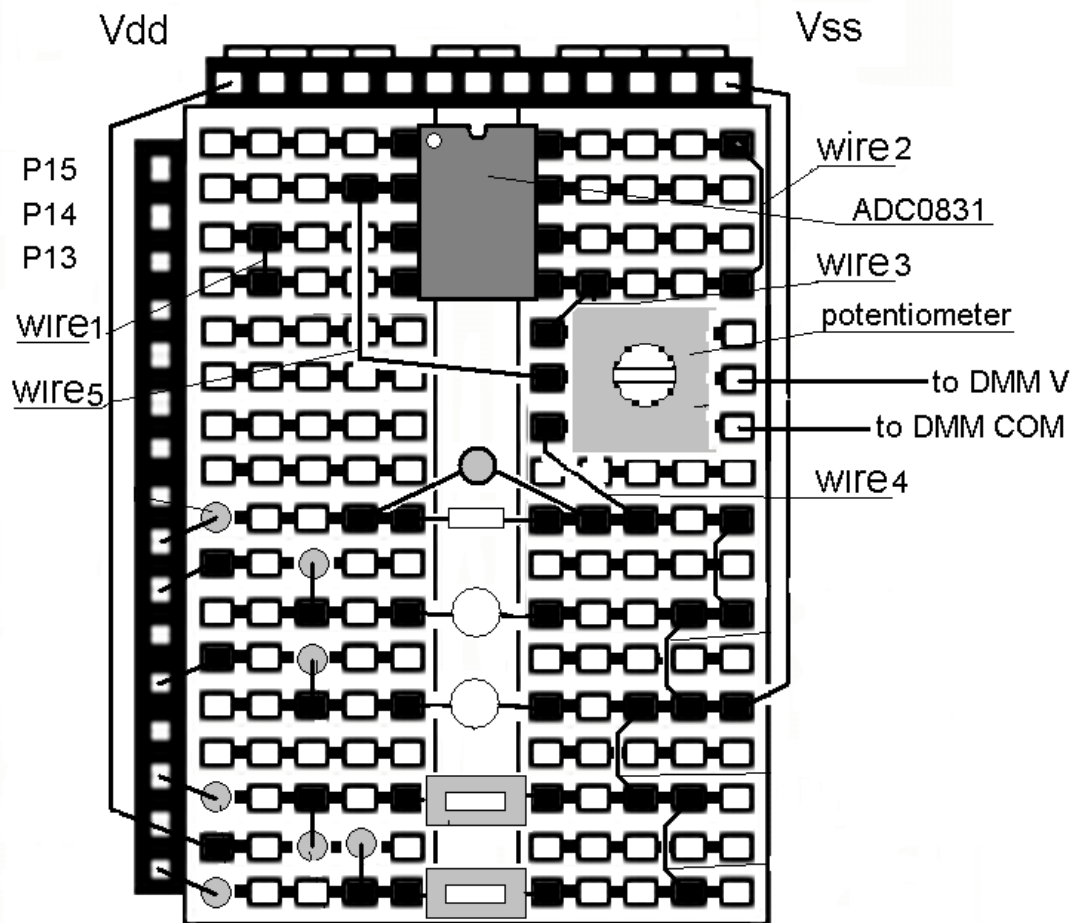
The pin map on the right shows the pins and labels according to where they are on the chip. The circuit symbol on the left also shows the pins and their labels, but it's typically drawn in a way that most conveniently fits into the schematic.



Potentiometer
"Pot"

Side view

Here is the board layout with SOME (but not all connections shown)



Assemble the circuit as follows: First - install the ADC0831 chip, oriented as shown and wire 1 and wire 2 if they are not already on the board.

Next add the potentiometer (again oriented as shown) and wires 3, 4, and 5. There are now five remaining connections to make which are NOT shown on the illustration. Do them in the order shown in the table.

| <i>ADC0831 pin</i> | <i>Connect to</i> |
|--------------------|-------------------|
| 1 | P15 |
| 8 | Vdd |
| 4 | Vss |
| 7 | P14 |
| 6 | P13 |

Finally, get a “binding post” adapter for the DMM. Using a long pair of wires (8”-10”), connect DMM COM and DMM V to the board as shown.

Program:

Load program "AD01.BS2" and run it. The DEBUG screen displays the value read from the ADC0831 in binary format. Adjust the potentiometer with a small screwdriver. The potentiometer in this configuration acts as a continuously variable voltage divider. The display should change as you turn the adjustment screw. At one extreme 0 volts (Vss) is applied to the input of the ADC and should provide a readout of 00000000. At the other extreme 5 volts (Vdd) is applied and the readout should be 11111111.

Record binary values for voltages of about 0,1,2,3,4,5 as shown on the DMM.

| Voltage (DMM) | Binary value |
|---------------|--------------|
| | |
| | |
| | |
| | |
| | |
| | |

Add the highlighted line to the program and download the program to the board.

```
Display:
DEBUG HOME, "8 bit binary value: ", BIN8 adcBits
DEBUG CR, CR, "Decimal value: ", DEC3 adcBits 'new line
RETURN
```

The DEBUG screen should now display both the binary and decimal values adcBits. With a voltage range of 0 to 5 volts and a converter count range of 0 to 255, what is the resolution of the analog to digital conversion?

_____ (Volts/Count)

If we want our answer in volts we need to scale the adcBits value to the range 0-5. So we add the two new lines shown below, download and run the program, and we are there.

```
Calc_Volts:
v = 5 * adcBits / 255 'new line
RETURN

Display:
DEBUG HOME
DEBUG "8-bit binary value: ", BIN8 adcBits
DEBUG CR, CR, "Decimal value: ", DEC3 adcBits
DEBUG CR, CR, "DVM Reading: ", DEC3 v, " Volts" 'new line
RETURN
```

Or are we? (Remember, P-Basic can only do integer math?)

The next code shows how to scale the result of the remainder operator (//) to get the fractional part of the voltage.

```
Calc_Volts:
v = 5 * adcBits / 255
r = 5 * adcBits // 255 'new line
v2 = 100 * r / 255 'new line
RETURN

Display:
DEBUG HOME
DEBUG "8-bit binary value: ", BIN8 adcBits
DEBUG CR, CR, "Decimal value: ", DEC3 adcBits
DEBUG CR, CR, "DVM Reading: " 'change line
DEBUG DEC1 v, ".", DEC2 v2, " Volts" 'new line
RETURN
```

You now have all the tools you need to build a basic analog or digital pH meter. Your assignment for Day 4 is to design a digital pH meter between now and next lab and build it during the next lab.

For the design we will assume the ideal pH electrode:

0 mV for pH7

-57 mV change per 1 pH unit change ($\Delta V = -57 \text{ mV} * \Delta \text{pH}$)

The analog to digital converter will be the ADC0831 on the BOE board.

Input range 0 to +5 volts.

The display will be the DEBUG window in PBASIC.

Each student should independently prepare their own design document.

The design document should include:

- 1) a sketch of the necessary op-amp circuitry (including resistor values or ratios and fixed voltages) necessary to match the electrode's output range to the input of the A/D converter.
- 2) a outline of a PBASIC program to calculate and display the pH to 0.1 pH units.

Have this ready to turn in at the beginning of the lab. I will review yours and your partners' designs and we will determine the final design that you, as a group, will build.

Start with the pH electrode: what is its output range (voltage at pH=0, voltage at pH=14)? What op-amp circuits are necessary to get that range to equal the input range of the analog to digital converter? What scaling factors are needed in the program to get the desired display (0.0 to 14.0 pH units)?

```

' Basic Analog and Digital (AD01.BS2)
' Program Listing 3.1 Revision 0.
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Declarations ]-----
adcBits VAR Byte
v VAR Byte
r VAR Byte
v2 VAR Byte
v3 VAR Byte

CS PIN 15
CLK PIN 14
DataOutput PIN 13

' -----[ Initialization ]-----
DEBUG CLS 'Start display.

' -----[ Main Routine ]-----
DO
GOSUB ADC_Data
GOSUB Calc_Volts
GOSUB Display
LOOP

' -----[ Subroutines ]-----
ADC_Data:
HIGH CS
LOW CS
LOW CLK
PULSOUT CLK, 210
SHIFTIN DataOutput,CLK,MSBPOST,[adcBits\8]
RETURN

Calc_Volts:
RETURN

Display:
DEBUG HOME
DEBUG "8-bit binary value: ", BIN8 adcBits
RETURN

```