Industrial Control

Student Guide

Version 1.0

Note regarding the accuracy of this text:
Many efforts were taken to ensure the accuracy of this text and the experiments, but the potential for errors still exists. If you find errors or any subject requiring additional clarification, please report this to stampsinclass@parallaxinc.com so we can continue to improve the quality of our documentation.

PARALLAX INC

Warranty

Parallax warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, Parallax will, at its option, repair, replace, or refund the purchase price. Simply call for a Return Merchandise Authorization (RMA) number, write the number on the outside of the box and send it back to Parallax. Please include your name, telephone number, shipping address, and a description of the problem. We will return your product, or its replacement, using the same shipping method used to ship the product to Parallax.

14-Day Money Back Guarantee

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a full refund. Parallax will refund the purchase price of the product, excluding shipping / handling costs. This does not apply if the product has been altered or damaged.

Copyrights and Trademarks

Disclaimer of Liability

Parallax, Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs or recovering, reprogramming, or reproducing any data stored in or used with Parallax products. Parallax is also not responsible for any personal damage, including that to life and health, resulting from use of any of our products. You take full responsibility for your BASIC Stamp application, no matter how life threatening it may be.

Internet Access

We maintain Internet systems for your use. These may be used to obtain software, communicate with members of Parallax, and communicate with other customers. Access information is shown below:

E-mail: stampsinclass@parallaxinc.com
Web: http://www.parallaxinc.com and http://www.stampsinclass.com

Internet BASIC Stamp Discussion List

We maintain two e-mail discussion lists for people interested in BASIC Stamps (subscribe at http://www.parallaxinc.com under the technical support button). The BASIC Stamp list server includes engineers, hobbyists, and enthusiasts. The list works like this: lots of people subscribe to the list, and then all questions and answers to the list are distributed to all subscribers. It's a fun, fast, and free way to discuss BASIC Stamp issues and get answers to technical questions. This list generates about 40 messages per day.

The Stamps in Class list is for students and educators who wish to share educational ideas. To subscribe to this list, go to http://www.stampsinclass.com and look for the E-groups list. This list generates about five messages per day.

Table of Contents

Preface

Industrial process control is a fascinating and challenging area of electronics technology and nothing has revolutionized this area like the microcontroller. The microcontroller has added a level of intelligence to the evaluation of data and a level of sophistication in the response to process disturbances. Microcontrollers are embedded as the "brains" in both manufacturing equipment and consumer electronic devices.

Process control involves applying technology to an operation that alters raw materials into a desired product. Virtually everything that you use or consume has undergone some type of automatic process control in its production. Automatic process control also provides higher productivity and better product consistency while reducing production costs.

This text is intended to introduce you to the concepts and characteristics of microcontroller-based process control. The hardware needed in the experiments to simulate the process has been kept to a bare minimum. While the microcontroller is the "brains" of the process, it is not the "muscle." Actual applications require the microcontroller to read and control a wide variety of input and output (I/O) devices. Information included in the experiments will help you understand the electrical interfacing of "real world" I/O devices to the BASIC Stamp.

The physical nature of the elements in a system determines the most appropriate mode of control action. The dynamics of a process include a study of the relationship of input disturbances and output action on the measured variables. It is difficult to understand the dynamics of a process without being able to "see" this relationship. For the authors, this defined a need to develop a graphical interface for the BASIC Stamp; hence the creation and release of StampPlot Lite. This software allows digital and analog values to be plotted on graphs, and time-stamped data and messages to be stored. StampPlot Lite is used throughout the experiments, and is especially helpful as you investigate the various modes of process control. Typical screen shots from program runs are included.

The authors of this material are Martin Hebel and Will Devenport. We are instructors at Southern Illinois University in Carbondale and co-owners of the consulting company, SelmaWare Solutions. We invite your comments and feedback. Contact us at www.selmaware.com, and copy all error changes to Parallax at stampsinclass@parallaxinc.com so the text may be revised.

We would like to thank our editors Ms. Cheri Barrall and Dale Kretzer, and of course Ken Gracey and Russ Miller of the Parallax staff for their review and improvement of this text.

Audience and Teacher's Guide

This text is aimed at an audience ages 17 and older. Effective during the first publication of this text in June, 2000, there is no Teacher's Guide edition planned. If a Teacher's Guide were to be published, it would likely be available the first part of year 2001. Solving these experiments presents no difficult technical hurdles, and can be done with a bit of patience.

Copyright and Reproduction

Experiment #1: Flowcharting and StampPlot Lite

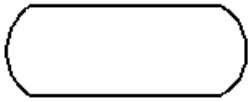
A flowchart is a detailed graphic representation illustrating the nature and sequencing of an operation on a step-by-step basis. A flowchart may be made of an everyday task such as driving to the store. How many steps are involved in this simple task? How many decisions are made in getting to the store? A formalized operation such as baking cookies can be flowcharted, whether on a small-scale process in your kitchen or on a very large scale in a commercial bakery. And, of course, a flowchart also may be made of the steps and decisions necessary for a computer or microcontroller to carry out a task.

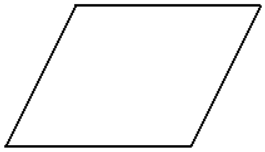
A relatively simple process is usually easy to understand and flows logically from start to finish. In the case of baking cookies, the steps involved are fairly easy. A recipe typically requires mixing the required ingredients, forming the cookies and properly baking them. There are several decisions to make: Are the ingredients mixed enough? Is the oven pre-heated? Have the cookies baked for the recommended time?

As processes become more complex, however, it is equally more difficult to chart the order of events needed to reach a successful conclusion. A BASIC Stamp program may have several dozen steps and possibly a number of **`if - then`** branches. It can be difficult to grasp the flow of the program simply by reading the code.

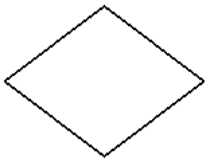
A flowchart is made up of a series of unique graphic symbols representing actions, functions, and equipment used to bring about a desired result. Table 1.1 summarizes the symbols and their uses.

Table 1.1: Flowchart Symbols

	Start/Stop box indicates the beginning and end of a program or process.
	Process box indicates a step that needs to be accomplished.

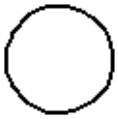


Input/Output box indicates the process requires an input or provides an output.

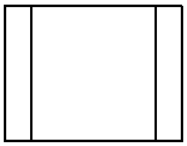


Decision box indicates the process has a choice of taking different directions based on a condition. Typically, it is in the form of a yes-no question.

Flowline is used to show direction of flow between symbols.



Connector box is used to show a connection between points of a single flowchart, or different flowcharts.



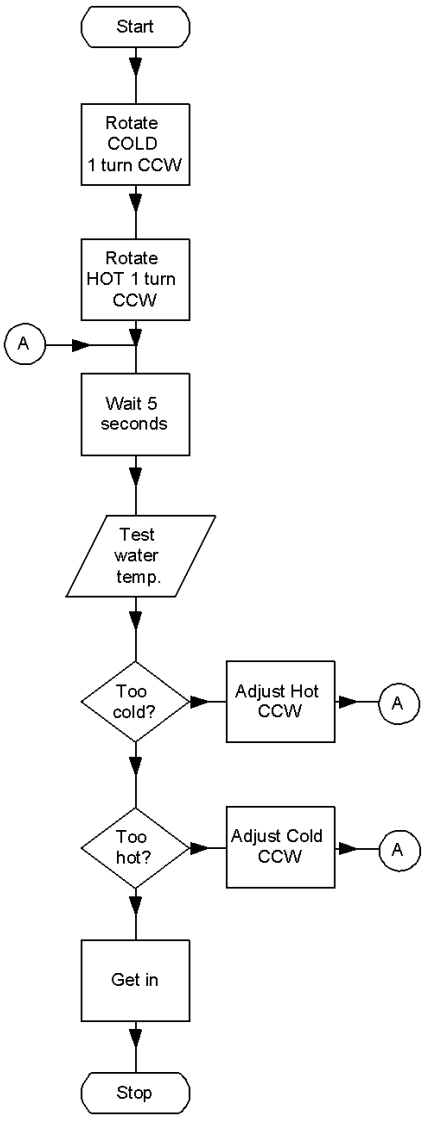
Sub-routine or sub-process box indicates the use of a defined routine or process.

Example #1: Adjusting the Temperature of a Shower

Let's take an example flowchart of an everyday task: adjusting the temperature for a shower. The process of adjusting the water temperature has several steps involved. The water valves are initially opened, we wait a while for the temperature to stabilize, test it, and make some decisions for adjustments accordingly. If the water temperature is too cold, the hot valve is opened more and we go back to test it again. If the water is too hot, the cold valve is opened more. Once we make this adjustment, we go back to the point where we wait for a few seconds before testing again. Of course this doesn't take into account whether the valves are fully opened. Steps may be inserted during the temperature adjustment procedure to correct for this condition. Figure 1.2 shows a flowchart of this process.

This example demonstrates a process that may be used in adjusting the temperature, but could it also be the steps in a microcontroller program? Sure! The valves may be adjusted by servos, and the water temperature determined with a sensor. In most cases, a simple process we go through can be quite complex for a microcontroller. Take the example of turning a corner in a car. Can you list all the various inputs we process in making the turn?

Figure 1.1:
Shower Temperature Example

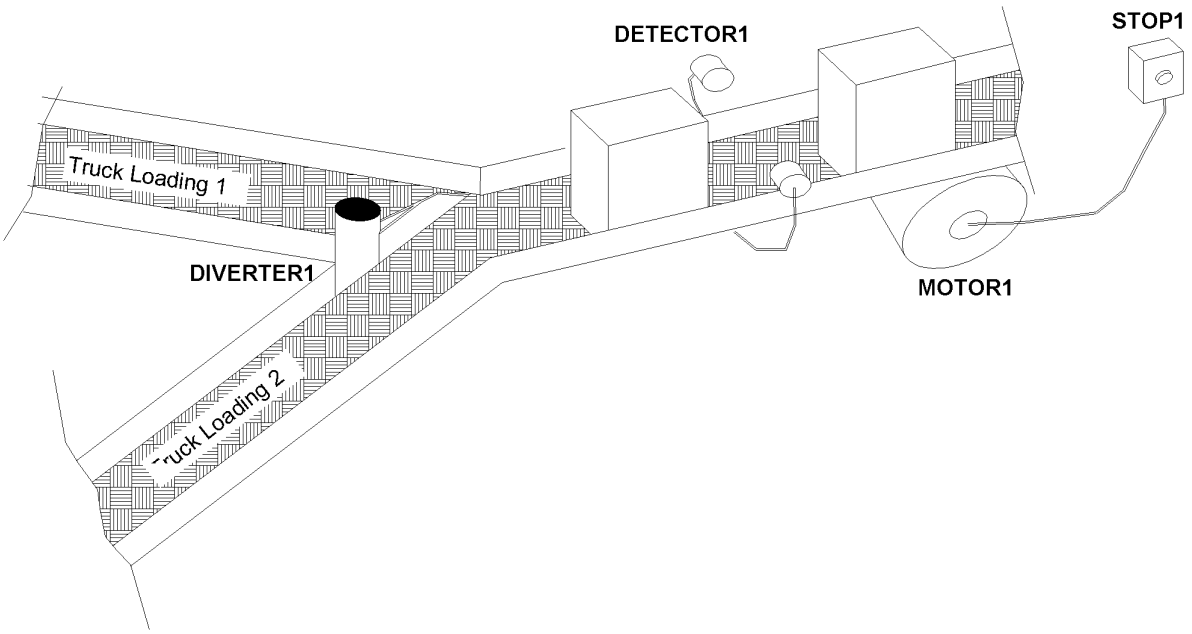


Example #2: Conveyor Counting Example

Let's look at a real scenario and develop a flowchart for it. In a manufacturing plant, items are boxed and sent down a conveyor belt to one of two loading bays with trucks waiting. Each truck can hold 100 boxes. As the boxes arrive, workers place them on the first truck. After that truck is full, the boxes must be diverted to the second truck so the loaded truck can be moved out and an empty one moved into position. Also, in the event of an emergency or problem, there must be a means of stopping the conveyor.

The physical aspects of the scenario are illustrated in Figure 1.2. The motor for the belt is labeled MOTOR1. The sensor to detect the boxes as they pass is labeled DETECTOR1. The lever to direct boxes to one truck conveyor or the other is labeled DIVERTER1. The emergency stop button is labeled STOP1.

Figure 1.2: Conveyor Counting Example



Let's list in order a brief description of what must occur:

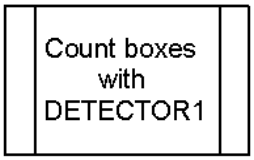
- Start the conveyor motor.
- Count the boxes as they pass.
- When 100 boxes have passed, switch the diverter to the opposite position.
- Whenever the emergency stop is pressed, stop the conveyor.

Now that we know the basic steps involved, let's develop a flowchart for the process. Let's begin by looking at the simple process flow in Figure 1.3.

Notice the placement of the Input/Output box for checking the emergency stop button, STOP1. It ensures the button is tested during every cycle. What if we had placed it following the 100-count decision box? How long would it have taken from when the button was pushed until the conveyor stopped?

Does the flowchart describe everything our program needs to do? Definitely not, but it is a good start at determining the overall flow of the process. Look at the "Count Boxes with DETECTOR1" Process box. How exactly is this carried out? We may need to develop a flowchart to describe just this routine. If a process needs further detailing, we might replace the Process box with a Sub-Process box as shown in Figure 1.4.

Figure 1.4: Sub-Process Box



How involved is it to simply count a box passing by a detector? If DETECTOR1 is activated by "going low," do we count? When the detector stays low, how do we keep from recounting it again the next time our program passes that point? What if the box bounces on the conveyor as it enters our beam? How do we keep from performing multiple counts of the box? These answers may not be as simple as they seem. Even when performing a task as simple as counting a passing box, many variables must be taken into account.

Another consideration is the output of our detector. Can we directly measure the output using one of the BASIC Stamp inputs, or is there some circuitry needed to condition the signal first?

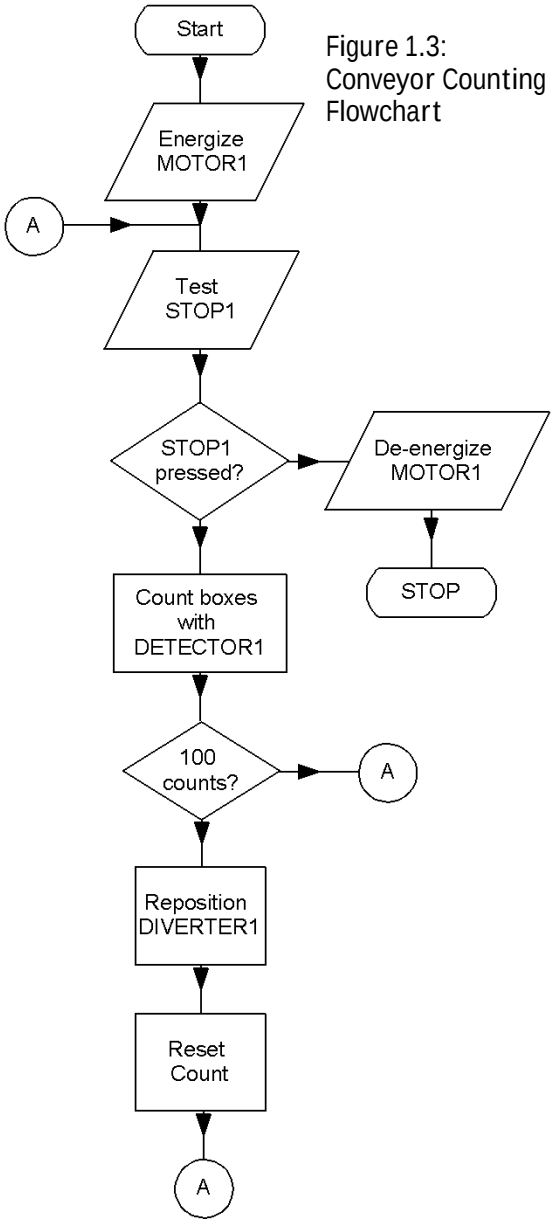


Figure 1.3: Conveyor Counting Flowchart

Let's consider an output in our conveyor counting example. How do we energize the motor? It is doubtful the 5-volt, milliamp-rated output of the BASIC Stamp will be able to drive a motor of sufficient horsepower to move a conveyor! How do we condition an output of the BASIC Stamp to control a higher voltage and current load?

These issues will be considered as you work through the chapters in this text. What may seem simple for us to do as humans may require some sophisticated algorithms for a microcontroller to mimic. We will use readily available electronic components, a BASIC Stamp module, and the Board of Education to simulate some complex industrial control processes.

Exercises

Exercise #1: Flowchart Design

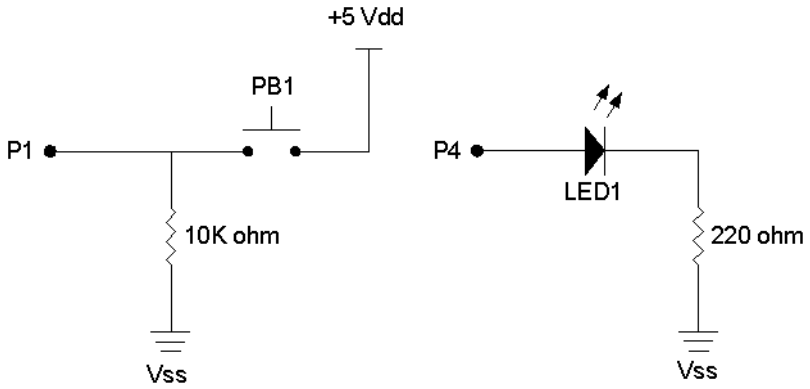
Draw a flowchart that will energize a heater below 100 degrees and de-energize it above 120 degrees.

Exercise #2: LED Blinking Circuit

We'll use a simple circuit to demonstrate a flowchart process and the program to perform the task. You'll need to build the circuit shown in Figure 1.5. The following parts will be required for this experiment:

(1) LED, green
(2) 220-ohm resistors
(1) 10K-ohm resistor
(1) Pushbutton
(1) 10K-ohm multi-turn potentiometer
(1) 1 uF capacitor
(miscellaneous) jumper wires

Figure 1.5: Exercise #2 Blinking Circuit Schematic

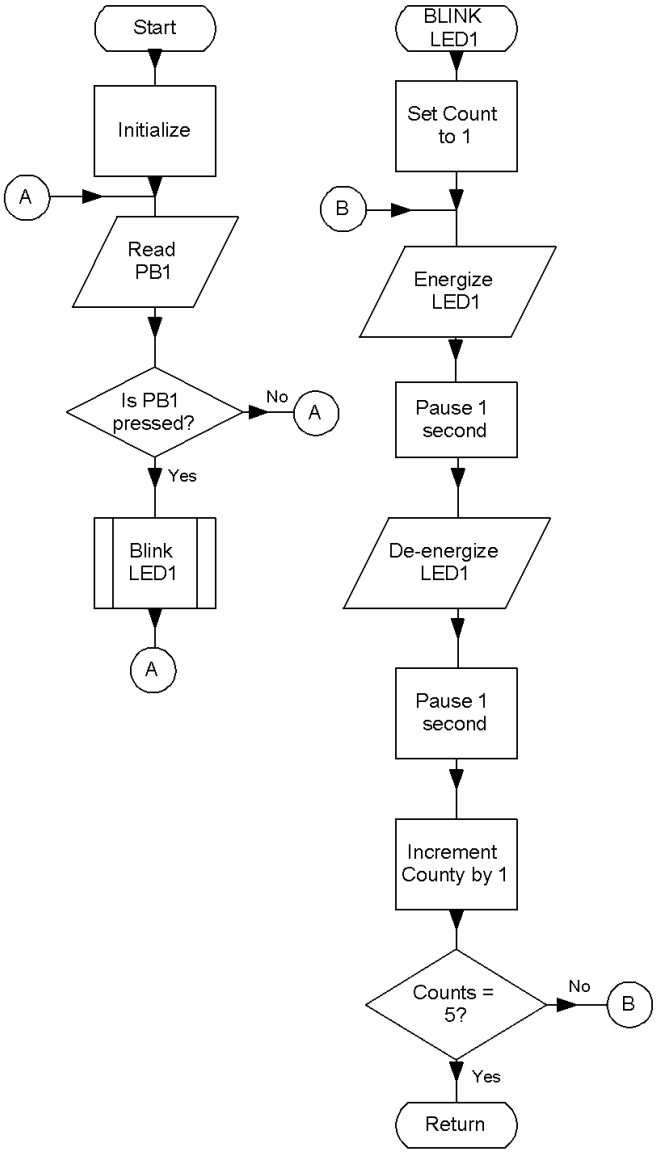


The circuit you are building consists of a single input button and a single output LED. Here is the process we want to perform: when the button (PB1) is pressed, blink the green LED (LED1) five times over 10 seconds. The flowchart for our process is shown in Figure 1.6.

Notice a few things about the flowchart. Our main loop is fairly simple. In the Initialize process box, we will define any variables needed and set initial outputs (LED off) and will loop unless PB1 is pressed, which calls our subroutine, **`blink_led1`**. Our subroutine doesn't begin with "Start," but the name of the process, so that we can identify it. The flowchart describes a process that we will repeat five times, alternately energizing and de-energizing our LED for one second each time.

Now that we have a flowchart to describe the process, how do we program it in PBASIC? Programmatically, we can sense PB1 using the **`in`** statement. We have two ways we can call our subroutine. If the condition is true (1), then we can branch to our subroutine directly using an **`if-then`** statement. This would be treated as a PBASIC **`goto`**. Once this completes, we would need to **`goto`** back to our main loop. Or, if the condition is false (0), we can branch back to our main loop from the **`if-then`**, and use a **`gosub`** command to branch to our subroutine when true. We can then use a **`return`** when our subroutine is complete.

Figure 1.6: Exercise #2 Blinking Circuit Flowchart



In our **blink_led1** subroutine, we need a loop to repeat five times. Choices for accomplishing this task may be to set up a variable we increment and check during each repetition, or use the **for-next** statement to accomplish it for us.

The flowchart describes the general steps involved in accomplishing a process. The code required is flexible as long as it faithfully completes the process as described. The same flowchart may be used in multiple languages or systems and even for humans!

Program 1.1 is one way to write the code for our blinking LED process. Enter the text in the BASIC Stamp editor, download it to the BASIC Stamp, and press the pushbutton of the circuit you built. If it works properly, the LED will blink five times after the pushbutton is pressed.

```
'Program 1.1, Blinking LED Example
cnt    var    byte          'A variable for counting
pb1    con    1             'PB1 is on P1
led1   con    4             'LED1 is on P4

input pb1                   'Setup PB as input
output led1                 'Setup LED as output

low led1                    'Turn OFF LED

loop:
  if in1 = 0 then loop      'Not pressed? Go back to start
  gosub blink_led1          'If it was pressed, then perform subroutine
goto loop                   'After return, go back to start

blink_led1:                 'Subroutine to blink LED
  for cnt = 1 to 5          'Set up loop for 5 repetitions
  high led1                 'Turn on LED
  pause 1000                'Wait 1 second
  low led1                  'Turn off LED
  pause 1000                'Wait 1 second
next                        'Repeat loop until done

return                      'Return to after the gosub in Loop
```

Programming Challenge

Flowchart and program a process where the LED will blink four times a second while the pushbutton is NOT pressed!

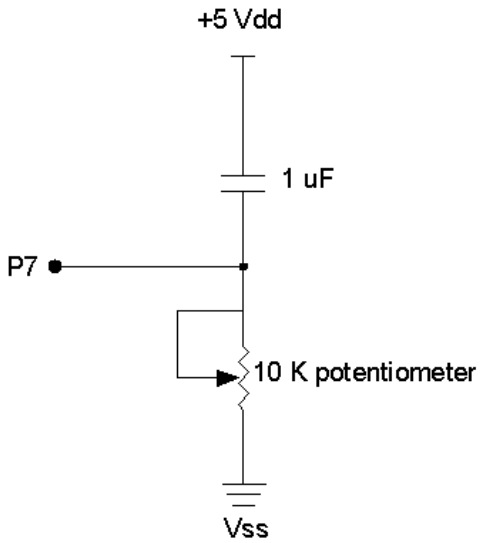
Exercise #3: Analog Data

In many instances a process involves analyzing and responding to analog data. Digital data is simply on or off (1 or 0). This is comparable to the simple light switches in our homes. The light is on or it is off. Analog data on the other hand is a range of values. Some examples include the level of lighting if we use a dimmer switch instead of an on/off switch, or the temperature of the water coming out of our shower nozzle.

There are several methods to bring analog data into a microcontroller, such as using an analog-to-digital (A/D) converter that changes analog values into digital values that may be processed by the microcontroller. Another method used by the BASIC Stamp is a resistor/capacitor network to measure the discharge or charge time of the capacitor. By varying the amount of the resistance, we can affect and measure the time it takes the capacitor to discharge. In this experiment, resistance is set by manually adjusting a variable resistor. But the device may be more sophisticated, such as a photo-resistive cell that changes resistance depending on the amount of light shining on it, or a temperature sensor. More discussion on analog data is found in later sections, but for now let's perform a simple process-control experiment using an analog value.

Add the RC network shown in Figure 1.7 to your circuit from the previous experiment.

Figure 1.7: Schematic for Analog Data circuit added to Exercise #3



PBASIC Command Quick Reference: RCTime

```
RCTIME pin, state, resultvariable
```

- `Pin` is the I/O pin connected to the RC network.
- `State` is the input voltage of that pin.
- `Resultvariable` is normally a word-length variable containing the results of the command.

.

The PBASIC command we will use to read the analog value of the potentiometer is **`rctime`**. A typical block of code to read the potentiometer is as follow:

```
high 7
pause 10
rctime 7, 1, pot
```

In order for it to read the potentiometer, the routine needs to take the following steps:

- +5 V (HIGH) is applied to both sides of the capacitor to discharge it.
- The BASIC Stamp pauses long enough to ensure the capacitor is fully discharged.
- When **`rctime`** is executed, Pin 7 becomes input. Pin 7 will initially read a high (1) because an uncharged capacitor acts as short.
- As the capacitor charges through the resistor, the voltage at Pin 7 will fall.
- When the voltage at Pin 7 reaches 1.4 V (falling), the input state is read as low (0), stopping the process and storing a value in **`pot`** proportional to the time required for the capacitor to charge.

The greater the resistance the longer the time required for a capacitor to discharge; therefore, the higher the value of **`pot`**. In this manner, we can acquire an analog value from a simple input device.

Let's write a process-control program to make use of this input. Our process will be one where temperature is monitored and a heater energizes below 100 degrees and de-energized above 120 degrees. The potentiometer will represent a temperature sensor and the LED will represent the heater being energized. We will use the debug window to display our temperature and the status of the heater. The maximum potentiometer value, with this combination of resistor and capacitor, may reach 5000, so we will divide it by 30 to scale it to a more reasonable range.

Enter and run Program 1.2. Monitor the value in debug window while adjusting the potentiometer and note what occurs as the value rises above 120 and below 100.

```
'Program 1.2, Simple Heater
led1   con    4                  'LED1 is on P4
rc     con    7                  'RC network is on Pin 7
pot    var    word               'Pot is a variable to hold results

output led1                      'Setup LED as output

goto HeaterON                    'Initially turn the heater on

loop:
  high rc                        'Read Potentiometer
  pause 10
  rctime rc, 1, pot
  pot = pot/30                   'Scale the results down
  debug "temp = ",dec pot, cr
  if (pot < 100) and (out4 = 0) then HeaterON
                                 'If temp < 100, and the output is
                                 'Off then turn it on.
  if (pot > 120) and (out4 = 1) then HeaterOFF
                                 'If temp => 120, and the output is
                                 'On then turn it off.
  pause 100
goto loop

HeaterON:                        'Energize and display
  high led1
  debug "The heater energized",cr
goto loop

HeaterOFF:                       'De-energize and display
  low led1
  debug "The heater de-energized", CR
goto loop
```

Programming Challenge

Modify the process so the LED indicates an air conditioner cycling between 70 and 75 degrees.

Exercise #4: Using StampPlot Lite

While the debug window for the BASIC Stamp is very useful for obtaining data and information from the BASIC Stamp, it can be difficult to visualize the information without careful scrutiny. Is the temperature increasing or decreasing? How quickly is it changing? At what point did the output change? What temperature is it cycling around? If you have not yet installed StampPlot Lite, install it on your computer by downloading it from http://www.stampsinclass.com. Double-click the setup button and install it in your designated directory.

Review Appendix A for an overview of StampPlot Lite. StampPlot Lite intercepts signals sent by the BASIC Stamp to the debug window.

Let's take another look at Program 1.2, our simple heater, but this time using StampPlot Lite to help visualize the process. Program 1.2 has been rewritten as Program 1.3 to utilize StampPlot Lite.

```
'Program 1-3, Simple Heater with plotting
'Configure StampPlot
pause 500
debug "!SPAN 50,150",CR                 'Set span for 50-150
debug "!TMAX 60",CR                     'Set for 60 seconds
debug "!PNTS 500",CR                    '500 data points per plot
debug "!TITL Simple Heater Control",CR  'Title the form
debug "!PLOT ON",CR                     'Enable plotting
debug "!RSET",CR                        'Reset Plot

led1   con    4                        'LED1 is on P4
rc     con    7                        'RC network is on Pin 7
pot    var    word                     'Pot is a variable to hold results

output led1                             'Setup LED as output

goto HeaterON                           'Initially turn the heater on

loop:
  high rc                               'Read Potentiometer
  pause 10
  rctime rc, 1, pot
  pot = pot/30                          'Scale the results down
  debug dec pot,cr,ibin1 out4,cr
  if (pot < 100) and (out4 = 0) then HeaterON
                                        'If temp < 100, and the output is Off
                                        'then turn it on.
  if (Pot > 120) and (out4 = 1) then HeaterOFF
       'If temp > 120, and the output is On
```

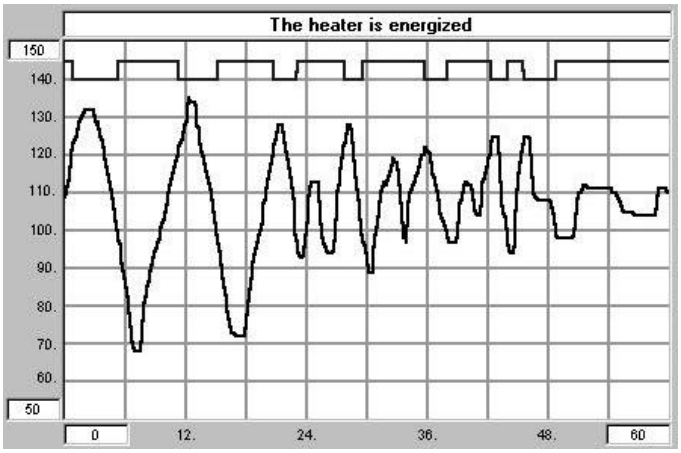
```
                                        'then turn it off.
  pause 100
goto loop

HeaterON:                               'Energize and display
  high led1
  debug "!USRS The heater is energized",cr
goto loop

HeaterOFF:                              'De-energize and display
  low led1
  debug "!USRS The heater is de-energized", cr
goto loop
```

Download this program to your BASIC Stamp, and follow these instructions to use StampPlot Lite.

Start StampPlot Lite by going to Start/Programs/StampPlot/StampPlot Lite. Enter and run Program 1.3 on your BASIC Stamp. Close the BASIC Stamp editor's blue debug window. Select the correct COM port in StampPlot Lite and click 'Connect.' Reset the BASIC Stamp by pushing the button on the Board of Education. Now you're ready to use this unique software utility.

At this point you should see data being plotted. Adjust the 10K-ohm potentiometer with your fingers or a small screwdriver. The analog line displays the value of the potentiometer. The digital trace at the top displays the status of the LED indicator. Figure 1.8 is a sample capture of the plot from our circuit.

Figure 1.8: StampPlot Lite Graph of Exercise #4



Note the correlation between the analog value and the switching of the digital output. Use the various controls on StampPlot Lite to become familiar with the functions and features. Analyze Program 1.3 and note the various configuration settings and data sent to the application. Refer to Appendix A for additional information on StampPlot Lite if you are having problems understanding the basics of the software utility.

Programming Challenge

Modify your air conditioner challenge from Exercise #2 to use StampPlot Lite. Configure your program to transmit data approximately every 0.5 seconds. Calculate the number of data points needed to fill the screen within a maximum of 60 seconds, and test.

Just for fun!

Enter and run the following program. The potentiometer simulates a single-handle shower (mixer) valve with adjustment delay. Adjust the shower temperature for a constant 110 degrees. See how fast you can stabilize the temperature at the set point! Press the reset button on the Board of Education and try again. We'll leave it up to you to figure out the program.

```
'Program 1-4, Adjust the shower!
settemp var byte
curtemp var byte
Diff var byte
Pot var word
Settemp = 110

Pause 500
debug "!RSET",CR,"!SPAN 0,200",CR,"!TMAX 30",CR,"!PLOT ON",CR
debug "!TSMP ON",CR,"!MAXS",CR,"!PNTS 100",13

Start:
debug "!USRS Adjust the pot for ",DEC Settemp,CR

Loop:
  high 7
  pause 10
  rctime 7, 3,pot
  pot = pot/ 30

  IF pot > curtemp then higher
  IF pot < curtemp then lower

goto display

higher:
  diff = pot - curtemp/5
  curtemp = curtemp + diff
goto display

lower:
  diff = curtemp - pot/5
  curtemp = curtemp - diff

display:
  debug dec curtemp,cr
  if curtemp <> settemp then skipbeep
  debug "At Setpoint!",CR,"!BELL",CR

SkipBeep:
  pause 250
goto loop
```

Questions and Challenge

1. List one everyday human process that involves a decision. List the steps in performing the process and the decisions needed to be made.

2. Develop a simple flowchart for the process in Question #1.

3. List an example of an electronics process in your home or school (such as that of an electric or microwave oven control, alarm clock, etc). Develop a simple flowchart to describe the process.

4. Develop the flowchart and code for the following process: The potentiometer simulates a temperature sensor. If the temperature exceeds 100 degrees, lock on the alarm (LED). Do not clear the alarm until the pushbutton is pressed.

5. Modify the program from Question #4 to use StampPlot Lite to display the temperature, alarm bit and status of the alarm.

Experiment #2: Digital Input Signal Conditioning

Process control relies on gathering input information, evaluating it, and initiating action. In industrial control, input information most often involves monitoring field devices whose outputs are one of two possible states. A switch is the most common example of a "bi-state" device. It is either open or closed.

Switches can provide control of an operation in three ways. One may be wired directly with the load and therefore control the full current and voltage. A switch also can be wired in the input circuit of a relay. In this case, the switch controls the relay's relatively low power input and the output contacts control load power. The on/off status of a switch may also provide a digital input to a programmable controller.

How many switches have you used today? And, what processes were affected by the toggling of those switches? Table 2.1 lists a few possibilities, starting at the beginning of your day:

Table 2.1: Switch Possibilities at the Beginning of your Day

Switch Status	Result
First, you may slap the "SNOOZE" button on your alarm clock.	The buzzing stops and -- Ah! 5 more minutes of sleep!
Next, stumble to the bathroom and flip "ON" the bathroom light.	Ouch! Turn it "OFF." Those vanity lights hurt!
Now, into the kitchen, start your coffeemaker, press down the toaster, and program your microwave. Open the refrigerator and the light comes on .	Breakfast is ready. And who knows if that light really goes off when you close the refrigerator?
Turn on the thermostat.	Heat or AC – your choice. What temperature? A setpoint is usually just a "switching point."
Turn on your TV, change the channel, turn up the volume.	The pushbuttons on the front or the flashing infrared LED in your remote– they all still just switch data.
Make a phone call. Lift the receiver and check for dial tone. Key in the phone number.	The limit switch held down by the handset now is in its "off the hook" position. Each switch on the keypad allows a specific tone to be generated.
Boot your PC. Switch on the monitor. Left click the mouse to check your e-mail.	These are only three obvious ones. There are many more switches behind the scenes in your PC.
You are up to 15 switches and you haven't even left your house!	

Some of the switches listed in Table 2.1 probably have direct control of electrical continuity to the loads involved. For example, the bathroom light switch controls the actual current flowing to the vanity light bulbs. The thermostat is an example of a switch controlling a low-voltage system that controls a relay in your furnace or air conditioner.

Most of the switches in Table 2.1, however, probably are providing a digital high or low signal being monitored by an electronic control system. It is the status of this input signal that is evaluated and used to determine the appropriate state of the outputs involved. The snooze button isn't physically opening the alarm circuit of your clock radio. When you "slapped" it, the momentary change of state was recognized by a programmable circuit. As a result, the program instructed the output to go off and add five minutes to the programmed alarm time. The start button on your microwave doesn't have to carry the actual current that powers the magnatron, inside light, and ventilation fan. However, pressing it creates an input causing the oven's microcontroller to close relays that do handle these loads.

Most often we think of switches as mechanical devices that make and break continuity between contact points in a circuit. In the case of the manual pushbutton and the limit switches pictured in Figure 2.1, this is exactly the case.

Figure 2.1: A Variety of Manual Pushbutton and Mechanical Limit Switches

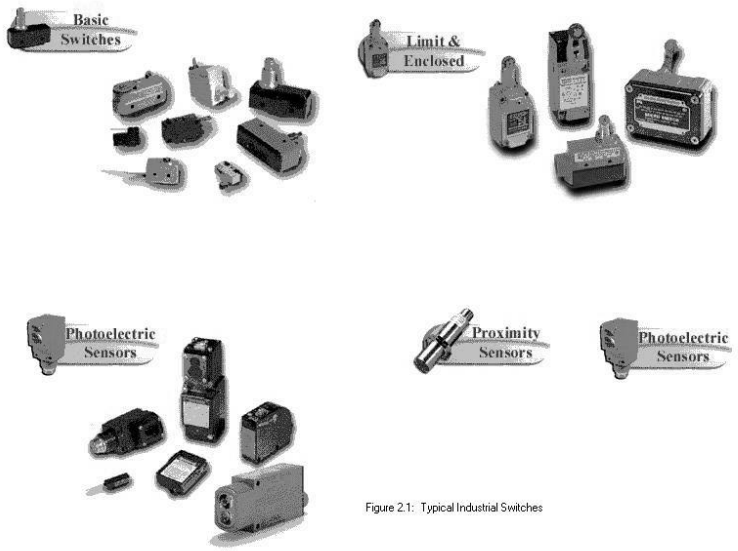
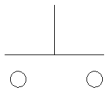
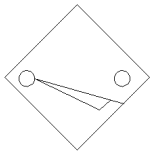
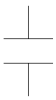
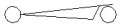
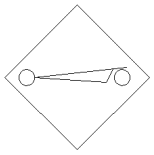


Figure 2.1: Typical Industrial Switches

Table 2.2 shows the schematic representation of various industrial switches. The symbols are drawn to represent the switch's "normal" state. Normal state refers to the unactuated or rest state of the switch. The pushbutton switches in this exercise kit are Normally Open (N.O.). Pressing the pushbutton results in a plunger shorting the contacts. The resistance goes from its open value of nearly infinite ohms to a value very near zero. A similar mechanism produces a like action in a Normally Open limit switch.

Table 2.2: Schematic Representation of Various Industrial Switches

	Push-button	Mechanical Limit	Proximity Switch	Relay Contacts
Normally Open				
Normally Closed				

While the concept of the switch doesn't get any simpler, there seems to be no limit to the physical design of switches that you will find in industrial control applications. Switches also may be designed as Normally Closed (N.C.); they are closed when at rest and actuation causes their contacts to open. As a technician, programmer, or system designer, you must be aware of the Normal (resting) position of a switch.

Figure 2.2: Schematic Representation of Pushbutton Switches

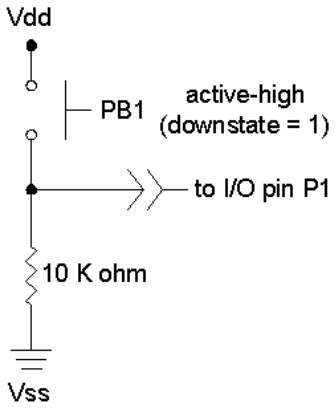


Figure 2.2a

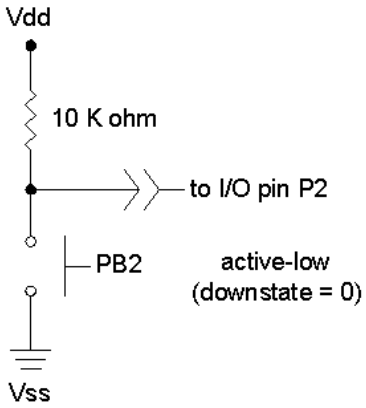


Figure 2.2b

What's a...

Digital Input (TTL, CMOS, ECL, etc.)?

Logic devices are built with a variety of processes that operate at different voltages. The manufacturer's datasheet will list several critical values for each device. Absolute Maximum Ratings are voltages and currents which must not be exceeded to avoid damaging or destroying the chip. I/O pins on the BASIC Stamp II should not exceed 0.6 V or Vdd+0.6 V (5.6V) with respect to Vss.

The logic transition between high and low is specified in the DC characteristics of the datasheet. A voltage of 0.2 Vdd (1 V on the BASIC Stamp II) is guaranteed to be low, which 0.45 Vdd (2.25 V) or higher is guaranteed to be high. There is a gray area between these two voltages where the actual transition will occur. It is dependant on temperature and supply voltages where the actual transition will occur. It also varies with temperature and supply voltage but will normally occur at about 1.4 volts.

The input pins of the BASIC Stamp do not detect "changes in resistance" between the switch's contacts. These inputs expect appropriate voltage levels to represent a logic high or a logic low. Ideally, these levels would be +5 volts for a logic high (1) and 0 volts for a logic low (0).

To convert the two resistive states of the switch into acceptable inputs, it must be placed in series with a resistor across the +5 volt supply of the BASIC Stamp. This forms a voltage divider circuit in which the resistive status of the switch is compared to the resistive value of the reference resistor. Figure 2.2 shows the two possibilities for our simple N.O. pushbutton switch. Figure 2.2a will result in +5 volts being fed to the input pin when it is pressed. When the switch is open, there is no continuity; therefore, no current flows through the 10K resistor and the input pin is grounded.

What's a...

Reference Resistor:

The 10K-ohm fixed resistor in Figures 2.2a and 2.2b is required to get dependable logic levels. It is wired in series with the switch. Its value must be much greater than the closed resistance of the switch and much less than its open resistance. When the switch is open in Figure 2.2a, the resistor gets no voltage and the input point is "pulled down" to ground. In Figure 2.2b, the open switch causes the input to be "pulled up" to +5 volts. You must consider the use of pull-up and pull-down resistors when working with all mechanical switches and some electronic switches.

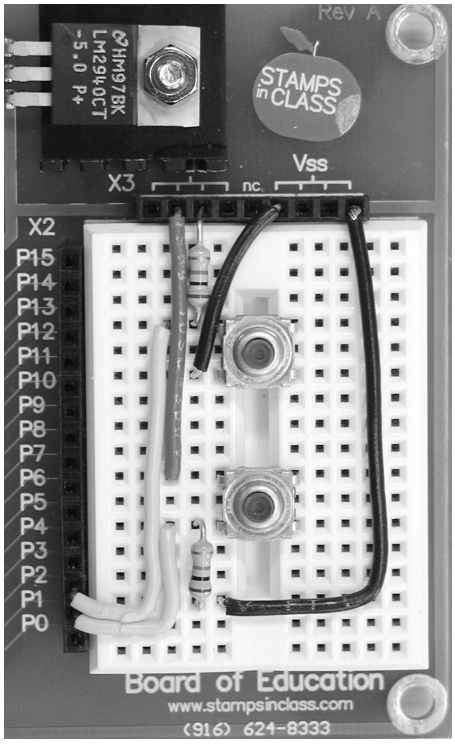
In Figure 2.2b, the switch closure results in grounding of the input pin. Zero volts is a logic low. When the switch is opened, there is again no voltage drop across the 10K-ohm resistor and the voltage at the input is +5, a logic high. The circuits are essentially the same, although the results of pressing the switch are exactly opposite. From a programming standpoint, it is important to know with which configuration you are dealing.

Exercises

Exercise #1: Switch Basics

To begin an investigation of programming for simple switch activity, wire the two pushbutton switches shown in Figure 2.2 onto the Board of Education breadboard. Connect the active-high configuration (Figure 2.2a) to I/O Pin 1 and the output of the active-low configuration (Figure 2.2b) to Pin 2. Note which one is which. As stated earlier, this is important. Figure 2.3 shows a pictorial of how the circuit is built on the Board of Education.

Figure 2.3: Pictorial of Parts Layout for circuits of Figure 2.2

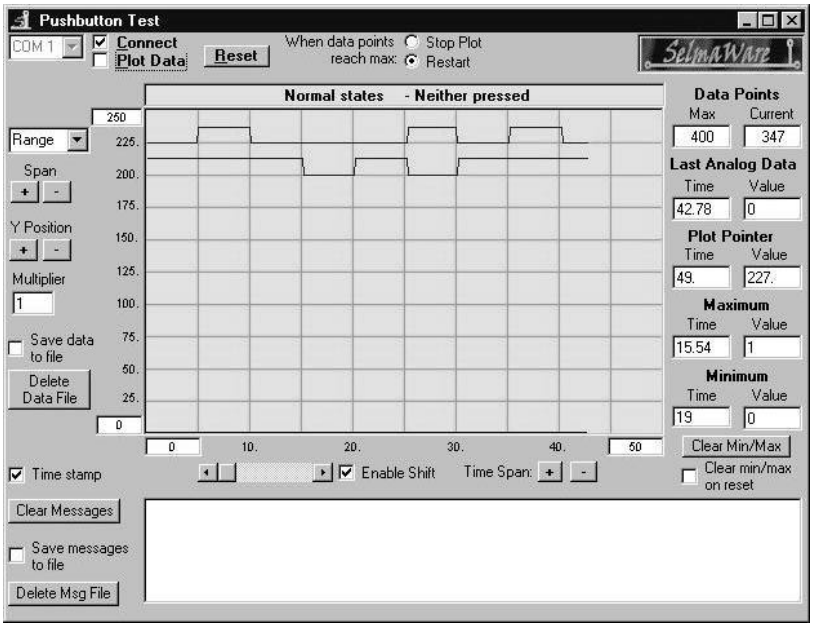


The following program is written to use the StampPlot Lite interface for displaying the status of the switches. The procedure will be the same as you followed in Experiment #1, Flowcharting and StampPlot Lite. First, enter Program 2.1. You may omit from the program all comments which include the apostrophe (') and the text that follows.

```
'Program. 2.1:  Switch Level Detection with StampPlot Lite Interface

debug "!TITL Pushbutton Test",13        ' Titles the StampPlot screen

input 1                                 ' Set P1 as an input
input 2                                 ' Set P2 as an input

Loop:
  pause 10                              ' Slow the program loop
  debug ibin in1, bin in2, 13           ' Plot the digital status
  debug dec 0, 13                       ' Output a 0 to allow for screen shift
  if (in1 = 1) and (in2 = 0) then both  ' Test for both pressed
  if in1 = 1 then PB1                   ' Test if active-high PB1 is pressed
  if in2 = 0 then PB2                   ' Test if active-low PB2 is pressed
debug "!USRS Normal states    - Neither pressed", 13
                                        ' Report none pressed
goto Loop

PB1:                                    ' Report PB1 pressed
  debug "!USRS Input 1 is High  - PB 1 is pressed ", 13
goto Loop

PB2:                                    ' Report PB2 pressed
  debug "!USRS Input 2 is Low   - PB 2 is pressed ", 13
goto Loop

Both:                                   ' Report both pressed
  debug "!USRS P1 High & P2 Low - Both pressed",  13
  debug "!BELL", 13                     ' Sound the bell.
goto Loop
```

Run the program. **Debug** will scroll the switch status and the input's digital value. Close the debug screen and open StampPlot Lite. Select the appropriate COM port and check the Connect and Plot Data boxes. Press the reset switch on your Board of Education and the trace of **in1** and **in2** should start across the screen. Your display should look similar to Figure 2.4. Press the pushbuttons and become familiar with the operation of your system. Next, we will look at how the program works.

Figure 2.4: Typical Screen Shot of StampPlot Monitoring the Status of Pushbuttons



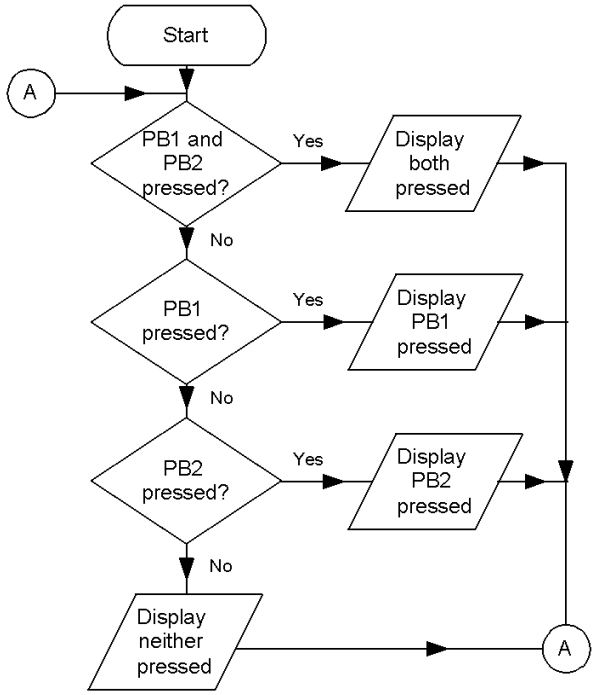
The purpose of this program is to run code based on the pressed or not-pressed condition of the two pushbuttons. This simple exercise gives insight to several considerations when dealing with digital inputs, programming multiple if-then statements, and using some of the PBASIC logical operators.

First, the statements in1 and in2 simply return the logic value of the input pins: +5 V = logic 1 and 0 V = logic 0. The active-high PB1 returns a 1 if pressed. The active-low PB2 returns a 0 when it is pressed. The program is testing for the "logical" status of the inputs; as the programmer, you must understand how this correlates to the "pressed" or "not pressed" condition of the pushbuttons involved. This is evident in the first line of the program loop where the logic operator AND is being used.

When you consider our switch configurations, it makes logical sense that if **in1** returns a logic high and **in2** returns a logic low then both switches are pressed. Output actions of industrial controllers often are dependent upon the status of multiple switches and contacts. A review of the PBASIC logical operators, including **and**, **or**, **xor** and **not,** can provide useful tools in meeting these requirements using the BASIC Stamp.

Another aspect of Program 2.1 is to notice the flow of the program loop. The **if-then** structure tests for a condition and **if** the condition is met, **then** the program execution is passed to the **label**. In this case, the label routine simply prints the conditions of the switches to the StampPlot Lite Status box. In industrial applications, this portion of the program would cause the appropriate output action to occur. Since the last line of each label is **goto loop**, program execution returns to the top of the loop and any code below that **if-then** statement is circumvented. The flowchart in Figure 2.5 shows how the program executes.

Figure 2.5: Flowchart for Program 2.1



If both switches are pressed, "**`if (in1 =1) and (in2 = 0`**)" is true. Program execution then would go to the **`both`** label. The "both pressed" condition would be indicated in the User Status Bar and your computer bell would ring. After this, program execution is instructed to go back to **`loop`** and test the switches again. As long as both switches remain pressed, the result of this test is continually true and looping is occuring only within this part of the program.

If either or both switches become not pressed, the next three lines of code will do a similar test for the condition.

Pressing PB1 results in "**`if in1 = 1`**" being true, execution is passed to the PB1 label action, and a return to the top of the loop; "**`if in2 = 0`**" is never tested. Is this good or bad? Neither, really. But, understanding the operation of multiple **`if-then`** statements can be a powerful tool for programming applications. Forgetting this can result in frustrating and not-so-obvious bugs in your program. For instance, what would happen in our program if the test for both switches being pressed "**`if (in1 =1) and (in2 = 0) then both"`** was put after the individual switch tests?

Quick Challenge

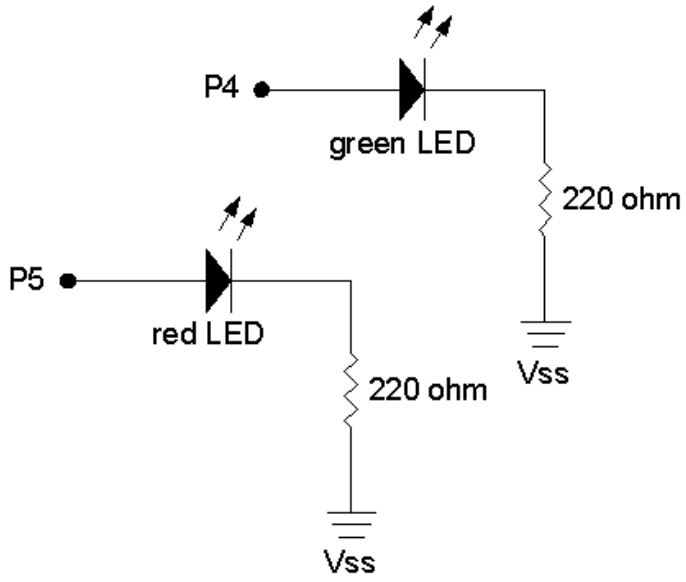
While running the program, try to reproduce the switch status shown in the screen shot of Figure 2.4.

Exercise #2 – Switch Bounce and Debouncing Routines

In the previous exercise, the steady-state level of the switch was being reported. The routine of reporting the switch status was performed on each program loop. What if you wanted to quickly press the switch and have something occur only once? There are two issues with which to contend. The first is: How quickly can you press and release the switch? You have to do it within the period of one program cycle. The second problem is contending with switch bounce. Switch bounce is the tendency of a switch to make several rapid on/off actions at the instant it is pressed or released.

The following program will demonstrate the difficulty in accomplishing this task. Two light-emitting diodes have been added as output indicators on Pin 4 and Pin 5. Wire the LEDs relative to Figure 2.6.

Figure 2.6: Active-High LED Circuit to be Added to the Schematic in Exercise #1



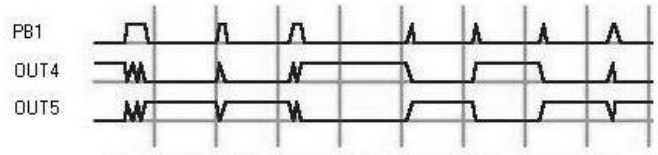
Enter and run the program according to StampPlot Lite procedures. The status of the pushbutton and the LEDs is being indicated. When PB1 is pressed, the LEDs will toggle. Can you be quick enough to make them toggle only once on alternate presses? Try it.

```
'Program 2.2 No Debouncing

pause 500
debug "!TITL Toggle Challenge",13' Titles the StampPlot screen
debug "!TMAX 25", 13                    ' Sets the plot time (seconds)
debug "!PNTS 300", 13                   ' Sets the number of data points

input 1                                 ' Set P1 as an input
input 2                                 ' Set P2 as an input
output 4                                ' Red LED
out4 = 1                                ' Initialize ON
output 5                                ' Green LED
out5 = 0                                ' Initialize OFF

Loop:
  debug ibin in1, bin in4, bin in5, 13  ' Plot the digital status.
  debug dec 0, 13                       ' Output a 0 to allow for screen shift
  if in1 = 1 then Action                ' Test the switch
                                        ' Optional pause 5 if StampPlot locks up
goto Loop

Action:                                 ' Toggle last state
  toggle 4
  toggle 5
GOTO Loop
```

If StampPlot Lite isn't responding to data sent by the BASIC Stamp, you may need to insert a very short delay in the **loop:** routine. A **pause 2** or **pause 5** (even up to 10 on slower computers) will alleviate any transmission speed problems you may encounter.

It is nearly impossible to press and release the pushbutton fast enough to perform the action only once. The problem is twofold as Figure 2.7 indicates. The program loop executes very fast. If you are slow, the program has a chance to run several times while the switch is closed. Add to this several milliseconds of switch bounce, and you may end up with several toggles during one press.

Figure 2.7: Slow Response and no Debounce Can be a Problem

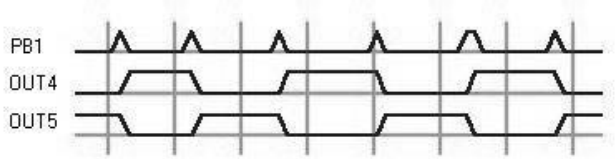


Further slowing the execution time of the program loop can help remedy the problem. (If the above program didn't work properly with StampPlot Lite, a delay in execution speed will allow for serial data transmission). Add a delay of 250 milliseconds to the **`Action:`** routine. This allows 250 milliseconds for the switch to settle after closing and then return to its open position.

Modify your program to include "Pause 250" to increase the loop time and negate switch bounce.

```
Program 2.3 (modify program 2-2 to slow it down)
Action:                    ' Toggle last state
  toggle 4
  toggle 5
  pause 250                ' Added to allow for settling time

goto loop
```

Figure 2.8: Adding a Pause Makes the Toggle Challenge Much Easier



By allowing settling time and pressing the button quickly, you make it much easier to get the **`Action:`** to occur only once. This technique helps debounce the switch and gives you enough time to release it before the next program cycle. The **`pause`** must be long enough to allow for these factors. If the **`pause`** is too long, however, a switch closure may occur and never be seen.

Exercise #3 – Edge Triggering

Counting routines pose additional problems for digital input programming. Exercise #2 used the **pause** command to eliminate switch bounce, which is compounded in industrial applications such as counting products on a conveyor. Not only does the switch have inherent bounce, but the product itself may have irregular shape, be wobbling, or stop for some time while activating the switch. There may be only one product, but the switch may open and close several times. Also, if the one product stays in contact with the switch for several program loop cycles, the program still should register it only once, not continually like in Program 2.2.

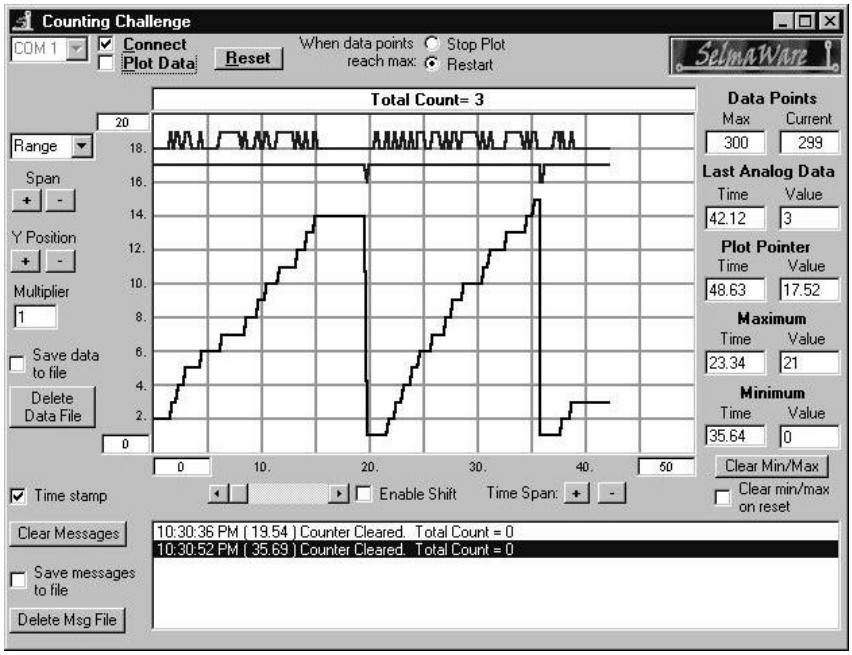
Program 2.4 uses a flag variable to create a program that responds to the initial low-to-high transitions of the switch. Once this "leading edge" of the digital input is detected, **Action:** will be executed. Then the flag will be set to prevent subsequent executions until the product has cleared and the switch goes low again. Enter Program 2.4.

```
' Program 2.4: Switch Edge Detection
' Count and display the number of closures of PB1.
' Reset total count with a closure of PB2.

pause 500
debug "!TITL Counting Challenge",13     ' Titles the StampPlot screen
debug "!TMAX 50",13                     ' Sets the plot time (seconds)
debug "!PNTS 300",13                    ' Sets the number of data points
debug "!AMAX 20",13                     ' Sets vertical axis (counts)
debug "!MAXR",13                        ' Reset after reaching maximum data points

input 1
input 2
Flag1 var bit                           ' flag for PB1
Flag2 var bit                           ' flag for PB2

Counts var word                         ' word variable to hold count
Flag1= 0                                ' clear the flags and Counts
Flag2 = 0
Counts = 0

Loop:
  pause 50
  debug "!USRS Total Count = ",dec Counts,13
  ' Display total counts in Status box
  debug dec Counts, 13                  ' Show counts on analog trace
  debug ibin in1, bin in2,13                  ' Plot the digital status.
```

```
  if in1 = 1 then Count_it              ' If pressed, count and display
  Flag1 = 0                             ' If not pressed, reset flag to 0
  if in2 = 0 then Clear_it              ' If PB2 is pressed, clear counts to 0
  Flag2 = 0
goto Loop

Count_it:
  if (in1 = 0) or (Flag1 = 1) then Loop ' If no longer pressed OR the
                                        ' flag is set, skip
  Counts = Counts +1                    ' Increment Counts
  Flag1 = 1                             ' Once Action executes, set Flag to 1
goto Loop

Clear_it:
  if(in2 = 1) or (Flag2 = 1) then Loop
  ' If no longer pressed,Or the flag is set, skip
  Counts = 0                            ' Clear counts to 0
  Flag2 = 1                             ' Prevents from clearing it again
  debug "Counter Cleared. Total Count = ", DEC Counts, 13
goto Loop
```

When PB1 is pressed, the program branches to the **`Count_it`** routine. Notice that the first line of this routine tests to see if the switch is open or **`Flag1`** is set. Neither is true upon the first pass through the program. Therefore, **`Counts`** is incremented, **`Flag1`** is set to 1 and program execution goes back to **`Loop`**. If PB1 still is being held down, **`Count_it`** is run again. This time, however, with **`Flag1`** set, the **`if-then`** statement sends the program back to **`Loop`** without incrementing **`Counts`** again. No matter how long the pushbutton is pressed, it will only register one "count" upon each closure. Although you are only incrementing the **`Count`** variable in this program, it could be part of a routine called for in an industrial application. Figure 2.9 is a screen shot that is representative of what you may see when running the program.

Figure 2.9: Running Program 2.4 - Edge Trigger Counting



Programming Challenge!

Use the indicating LEDs on output Pins 4 and 5, along with the two pushbuttons, to simulate a parking lot application. Assume your parking lot can hold 24 cars. Pushbutton PB1 will be counting cars as they enter the lot. Pushbutton PB2 will count cars as they leave. Write a program that will keep track of the total cars in the parking lot by counting "up" with PB1 and "down" with PB2. Have the green LED on as long as there is a vacancy in the lot. Turn the red LED on when the lot is full. Continually display how many parking spaces are available in the User Status window (!USRS). Plot continually the number of cars in the parking lot.

Additional StampPlot Lite Challenge

Keep a file of the number of times your parking lot went from "Vacancy" to "Full" (see Appendix A and the StampPlot Lite help file for information on using the Save Data to File option).

BUTTON Command: PBASIC's Debouncing Routine

Debouncing switches is a very common programming task. Parallax built into the PBASIC2 instruction set a command specifically designed to deal with digital input signal detection. The command is called **`button`**. The syntax for the command is shown below.

PBASIC Command Quick Reference: BUTTON

BUTTON pin, downstate,delay,rate,bytevariable,targetstate, address

- *Pin:* (0-15) The pin number of the input.
- *Downstate:* (0 or 1) Specifiying which logical state occurs when the switch is activated.
- *Delay:* (0-255) Establishes a settling period for the switch. Note: 0 and 255 are special cases. If delay is 0, Button performs no debounce or auto-repeat. If delay is 255, Button performs debounce but no auto-repeat.
- *Rate:* (0-255) Specifies the number of cycles between autorepeats.
- *Bytevariable:* The name of a byte variable needed as a workspace register for the BUTTON instruction.
- *Targetstate:* The state of the pin on which to have a branch occur.
- *Address:* The label to branch to when the conditions are met.

To try it with our counting routine, load and run program Program 2.5.

```
' Program 2.5: Button Exercise with StampPlot Interface
' Use Button to count and display the number of closures of PB1.
' Reset total count with a closure of PB2.

Pause 500
debug "!TITL Counting Challenge",13     ' Titles the StampPlot screen
debug "!TMAX 50",13                     ' Sets the plot time (seconds)
debug "!PNTS 300",13              ' Sets the number of data points
debug "!AMAX 20",13               ' Sets vertical axis (counts)
debug "!MAXR",13

Wkspace1  var  byte               ' Workspace for the BUTTON command for PB1
Wkspace1 = 0                      ' Must clear workspace before using BUTTON
Wkspace2  var  byte               ' Workspace for the BUTTON command for PB2
Wkspace2 = 0                      ' Must clear workspace before using BUTTON

Counts var word                   ' Word variable to hold count
Counts = 0
```

```
Loop:
  pause 50
  button 1,1,255,0,Wkspace1,1,Count_it
                                ' Debounced edge trigger detection of PB1
  button 2,0,255,0,Wkspace2,1,Clear_it
                                ' Debounced edge trigger detection of PB2

  debug "!USRS Total Count = ", DEC Counts,13
                                ' Display total counts in Status box
  debug dec Counts, 13          ' Show counts on analog trace
  debug ibin in1, bin in2,13    ' Plot the digital status.
goto Loop

Count_it:
  Counts = Counts +1                  ' Increment Counts
goto Loop

Clear_it:
  Counts = 0                    ' Clear counts to 0
  debug "Counter Cleared. Total Count = ", DEC Counts, 13
                                ' Display in Text Box
goto Loop
```

Review the documentation concerning the **`button`** command in the BASIC Stamp Programming Manual Version 1.9. This is a very handy command for industrial applications. Experiment by changing the delay time from 50 to 100 and to 200. See if you can press the switch more than one time but only get one **`Action`** to take place. What would be the risk of allowing for too much settling time in "high speed" counting applications? Save this program; it will be modified only slightly for use with the next programming challenge.

Electronic Digital Input Sources

It is very common for digital inputs to come from the outputs of other electronic circuits. These inputs may be from a variety of electronic sources, including inductive or capacitive proximity switches, optical switches, sensor signal-conditioning circuits, logic gates, and outputs from other microcontrollers, microprocessors, or programmable logic control systems.

There are several things to consider when interfacing these sources to the BASIC Stamp. Primarily, "Are they electrically compatible?"

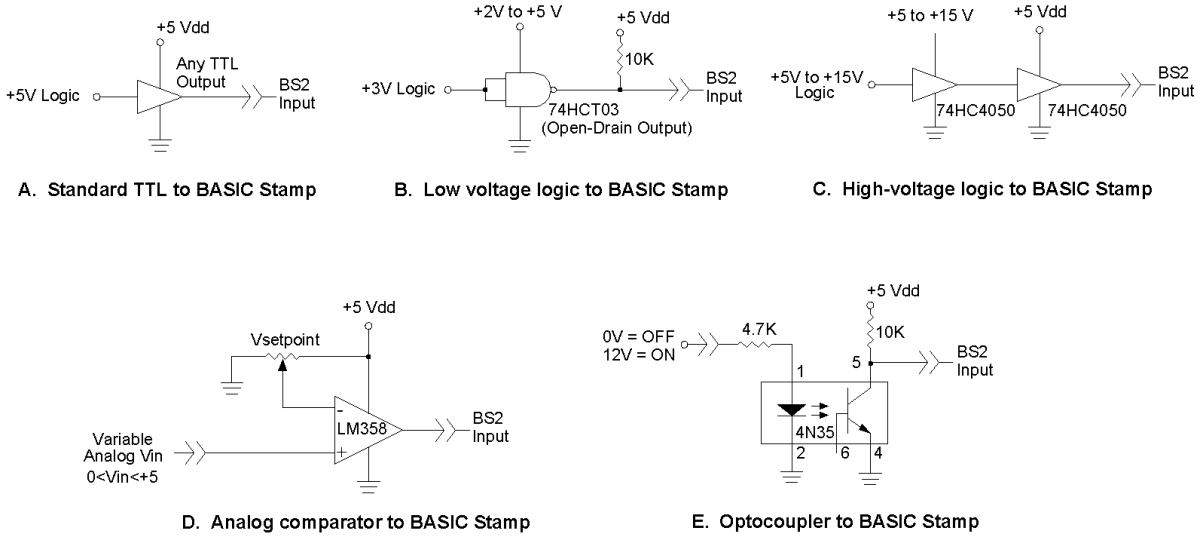
1. Is the source's output signal voltage within BASIC Stamp input limits?
2. Is the ground reference of the circuit the same as that of the BASIC Stamp?
3. Is protection of either circuit from possible electrical failure of the other a concern such that isolation may be necessary?

Once a compatible signal is established, the next question becomes, "Is the program compatible to respond to the input?"

1. Is digital bounce an issue?
2. How fast is the data? What is its frequency? What is the minimum pulse time?
3. Is action to be taken based on the data's steady-state level or on its leading or trailing edges?

Figure 2.10 shows a variety of electrical interfacing possibilities you may face.

Figure 2.10: Input Interfacing of Electronics to the BASIC Stamp



A. Standard TTL to BASIC Stamp

B. Low voltage logic to BASIC Stamp

C. High-voltage logic to BASIC Stamp

D. Analog comparator to BASIC Stamp

E. Optocoupler to BASIC Stamp

(a) TTL and CMOS logic inputs powered from a +5-volt supply can be applied directly to the BASIC Stamp's input pins. If the two systems are supplied from the same 5 volts, great. If not, at least the grounds must be common (connected together).

(b) Low-voltage (+3 V) devices can be interfaced using a 74HCT03 or similar open-drain gate with a pull-up resistor to the BASIC Stamp's +5-volt supply. Supply the chip with the low-voltage supply and make the grounds common.

(c) Higher-voltage digital signals can be interfaced using a 74HC4050 buffer or 74HC4049 inverter powered at +5 volts. These devices can safely handle inputs up to 15 volts. Again, the grounds must be common.

(d) A referenced comparator op-amp configuration can establish a High/Low output based on the analog input being above or below the setpoint voltage. The LM35810 is an op-amp whose output will go nearly rail to rail on a single-ended, +5-volt supply. It will be used in the upcoming application.

(e) An opto-coupler may be used to interface different voltage levels to the BASIC Stamp. The LED's resistor holds current to a safe level while allowing enough light to saturate the phototransistor. The input circuit can be totally isolated from the phototransistor's BASIC Stamp power supply. This isolation provides effective protection of each circuit in case of an electrical failure of the other.

Exercise #4: An Electronic Switch

Electronic switches that provide "non-contact" detection are very popular in industrial applications. No physical contact for actuation means no moving parts and no electrical contacts to wear out. The pushbutton switch used earlier should be good for several thousand presses. However, its return spring eventually will fatigue, or its contacts will arc, oxidize, or wear to the point of being unreliable.

Industrial electronic switches operate on one of three principles.

- Inductive proximity switches sense a change in an oscillator's performance when metal objects are brought near it. Most often the metal objects absorb energy via eddy currents from the oscillator causing it to stop.
- Capacitive proximity switches sense an increase in capacitance when any type of material is brought near them. When the increase becomes enough, it causes the switch's internal oscillator to start oscillating. Circuitry is then triggered and the output state is switched.
- Optical switches detect the presence or absence of a narrow light beam, often in the infrared range. In retroreflective optical switches, the light beam may be reflected by a moving object into the switch's optical sensor. Through-beam optical switches are set up such that the object blocks the light beam by going between the light source and the receiver.

The output of the an electronic switch is a bi-state signal. It's final stage may be any one of the types seen in Figure 2.10. As a technician and application developer, you must consider the nature of this signal circuit and condition it for the digital input of the microcontroller. The manufacturer's datasheet will give you information on the operating voltage for the switch and typical load connections.

Although you can think of the BASIC Stamp's digital input pin as the load, the electronic switch may require a reference resistor as used earlier in Figure #2.2. Most likely, the output of the proximity switch will be very near 0 volts in one state and near its supply voltage in the other state. It is always a good idea to test the switch's output states with a voltmeter before applying them to the unprotected input of the microcontroller. If the output voltages are not within the compatible limits of the BASIC Stamp, you will need to use one of the circuits in Figure 2.10 as an appropriate interface.

The following exercise focuses on the design and application of an optical switch. We will use this switch to detect and count objects. Then the switch will be used as a tachometer to determine RPM.

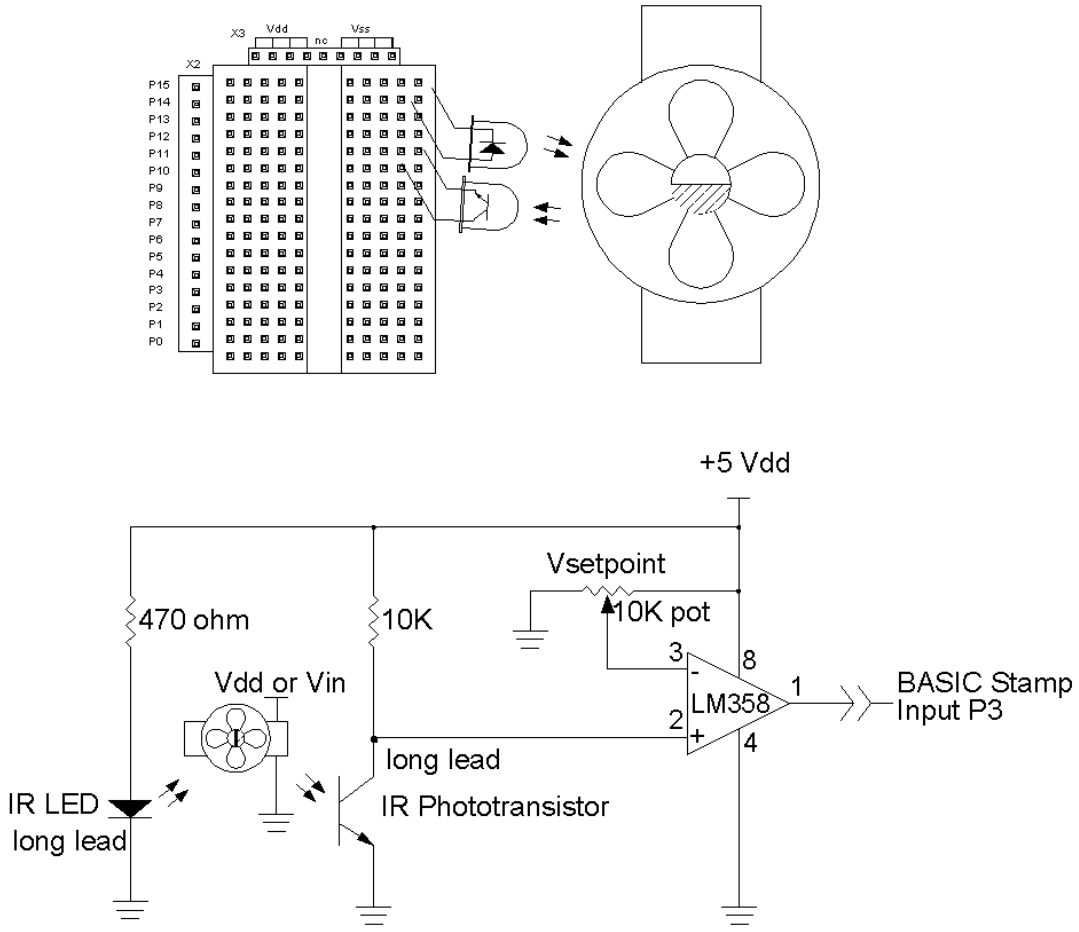
In Figure 2.11, the infrared light-emitting diode (LED) and the infrared phototransistor form a matched emitter/detector pair. Light emitted by the LED will result in phototransistor collector current. An increase in collector current drives the phototransistor toward saturation (ground). If the light is prevented from striking the phototransistor, it goes toward cutoff and the collector voltage increases positively. These conditions of light and no-light will most likely not provide a legal TTL signal at the collector of the transistor. Applying this signal to the input of a referenced comparator will allow us to establish a setpoint somewhere between the two conditions. The output of the comparator will be a compatible TTL logic signal. It's level is dependant on which side of the setpoint the phototransistor's output is on. The LM358 op-amp is a good choice for this application. It can operate on a +5-volt single supply and its output saturation voltages are almost equal to the supply potentials of +5 and ground.

Carefully construct the circuit in Figure 2.11 on the Board of Education breadboard. Mounting the devices near one end as pictured in the diagram allows for additional circuits in upcoming exercises. Make a 90^{o} bend in the LED and phototransistor leads so the devices lie parallel to the the benchtop. The phototransistor and infrared LED should be placed next to each other, pointing off the edge of the breadboard.

The LED in Figure 2.11 is emitting a continuous beam of infrared light. With the LED and phototransistor side-by-side, there is little or no light coming into the phototransistor because there is nothing reflective in front of it. If an object is brought toward the pair, some of the LED light will bounce back into the phototransistor. When light strikes the phototransistor, the collector current will flow and the collector voltage will drop. In this setup, the scattered reflection of light off an object as it passes in front of the pair will be sensed by the phototransistor. The amount of reflected light into the sensor depends on the optical reflectivity of the target object and the geometry of the light beam. We will attempt to determine the presence of a flat-white object. With the emitter and detector mounted side-by-side, you will try for detection of the object at a distance of one inch.

You must make a couple of voltage measurements to calibrate the presence of the object. Begin by placing a voltmeter across the phototransistor's collector and emitter. Measure the voltage when there is no object in front of the sensor. Record this value in Table 2.3. Next, move a white piece of paper toward and away from the pair and notice the variation in voltage. As the paper is brought near the IR pair, the reflected light increases collector current and drives the transistor toward saturation –"low." Record the voltage reading with the white paper approximately one inch in front of the sensor in Table 2.3. The difference between these measurements may be quite small, like 0.5 V, but that will be enough to trigger the op-amp. This signal is applied to the inverting input of the LM358 comparator. The potentiometer provides the non-inverting input reference voltage. This reference should be a value between the "no reflection" and "full reflection" readings. Adjust the potentiometer to provide the proper reference voltage, which is halfway between the measurements.

Figure 2.11: Retro-reflective Switch Pictorial and Schematic



Testing the output of the LM358 should result in a signal compatible with the BASIC Stamp. The output should be low with no object and high when the white object is placed in front of the emitter/detector pair. Measure these two output voltages of the LM358 and record the values in Table 2.3. If the output signal is compatible, apply it to the BASIC Stamp's Pin 3. Detecting light reflected by an object is called retro-reflective detection.

Table 2.3: LM358 Values

Condition	Phototransitor Voltage	LM358 Output Voltage
No object – no reflection		
Object – full reflection		
Reference voltage setpoint		

This ability to yield a switching action based on light received lends itself to many industrial applications such as product counting, conveyor control, RPM sensing, and incremental encoding. The following exercise will demonstrate a counting operation. You will have to help, though, by using your imagination.

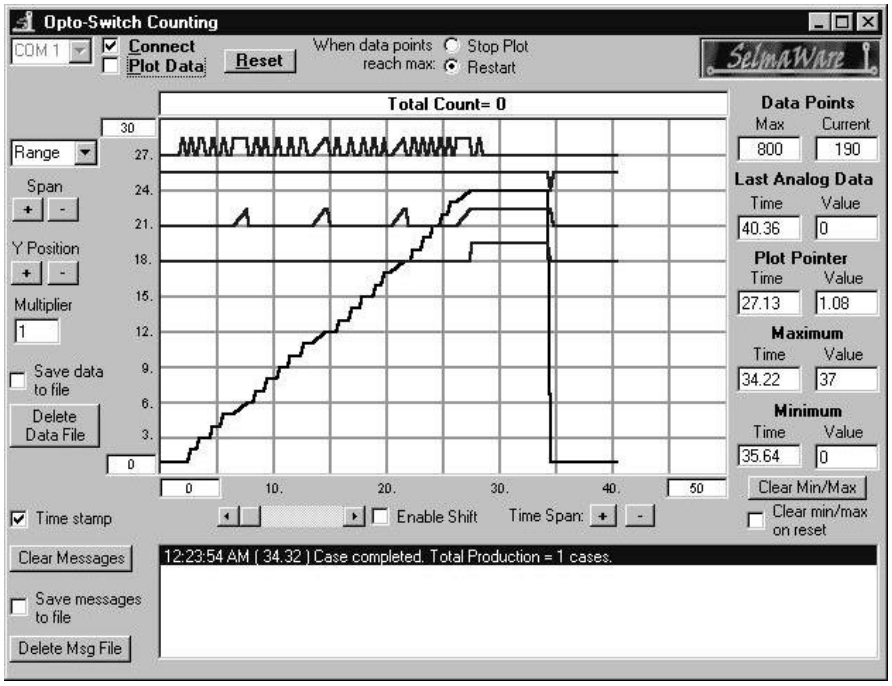
Let's assume that bottles of milk are being transferred on a conveyor between the filling operation and the case packer. Cut a strip of white paper to represent a bottle of milk. Passing it in front of our switch represents a bottle going by on the conveyor. Only a slight modification of the previous program is necessary to test our new switch. If you have Program 2.5 loaded, simply modify the first **`button`** instruction by changing the input identifier from Pin 1 to 3. The modified line would look like this:

```
' Program 2.6 (modification to Program 2.5
' for the retroreflective switch input)

button 3,1,255,0,Wkspace1,1,Count_it
' Debounced edge trigger detection of optical switch
```

Programming Challenge: Milk Botttle Case Packer

Refer back to Experiment #1 and consider the conveyor diverter scenario in Figure 1.2. We will assume that the controller is counting white milk bottles and controlling a diverter gate and a case packer drop chute. Use the green and red indicating LEDs attached to the StampPlot Lite output Pins 4 and 5 to represent these outputs. Modify your program to count bottles of milk on the conveyor. Let the diverter gate be used to direct the bottles to a case-packing machine. Change the status of the diverter each time six bottles have been counted. Upon 24 bottles being counted (the 4th six-pack), turn the "case packer" LED on for two seconds. Use the StampPlot Lite interface to monitor the operation. Report to the User Status window the present count of bottles in the case. Monitor the bottle count and the digital output status using the analog and digital graphs. Clear it after each case. In the lower Text Box, time-stamp each time a case is filled and report how many total cases have been processed. The interface should appear similar to the example screen shot in Figure 2.12.

Figure 2.12: Example Screen Shot of Milk Bottle Challenge



Exercise #5: Tachometer Input

Monitoring and controlling shaft speed is important in many industrial applications. A tachometer measures the number of shaft rotations in a unit of time. The measure is usually expressed in revolutions per minute (RPM).

A retroreflective switch can open and close fast enough to count white and black marks printed on a motor's shaft. Counting the number of closures in a known length of time provides enough information to calculate RPM. Figure 2.13 represents five possible encoder wheels that could be attached to the end of a motor shaft. If the optical switch is aimed at the rotating disk, it will pulse on-off with the alternating segments as they pass. The number of white (or black) segments represent the number of switch cycles per revolution of the shaft. The first encoder wheel has one white segment and one black segment. During each revolution, the white segment would be in front of our switch half the time, resulting in a logic high for half the rotation.

During the half rotation the black segment is in front of the disk, it absorbs the infrared light and with no reflected light, the switch will be low. One cycle of on-off occurs each revolution. The PBASIC2 instruction set provides a very useful command called **count** that can be used to count the number of transitions at a digital input occuring over a duration of time. Its syntax is shown below.

PBASIC Command Quick Reference: COUNT

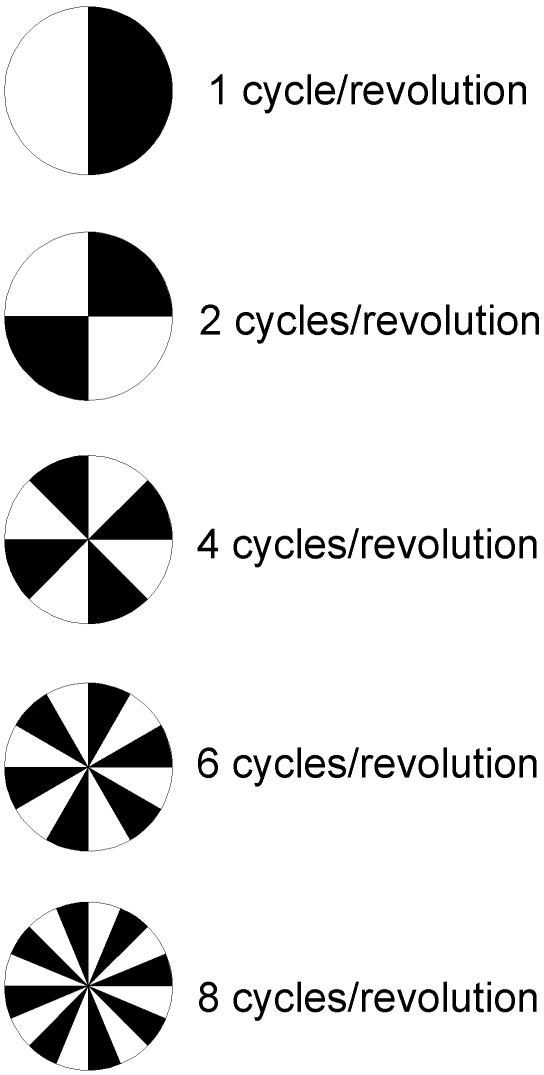
COUNT pin, period, variable

- *Pin: (0-15) Input pin identifier.*
- *Period: (0-65535) Specifies the time in milliseconds during which to count.*
- *Variable: A variable in which the count will be stored.*

The following exercise uses the count instruction, the optical switch, and the shaft encoder wheels to capture speed data.

Begin by cutting out the first encoder wheel. Use a small piece of cellophane tape to hold the encoder to the shaft hub of the fan motor (a full-size encoder wheel set may be pulled from Appendix B of this text). Wire a 10K-ohm trimpot in series with your fan so its speed can be varied. The fan is rated at 12 V. Its speed changes with varying voltages from 12 V down to approximately 3.5 V. This is the dropout voltage of the brushless motor control circuitry. The fan should be located so the encoder wheel is pointed at the emitter/detector pair. Pin 20 of connector X1 provides access to the 12-volt unregulated supply.

Figure 2.13: Retro-reflective Encoder Wheels
(cutouts are available in Appendix B)



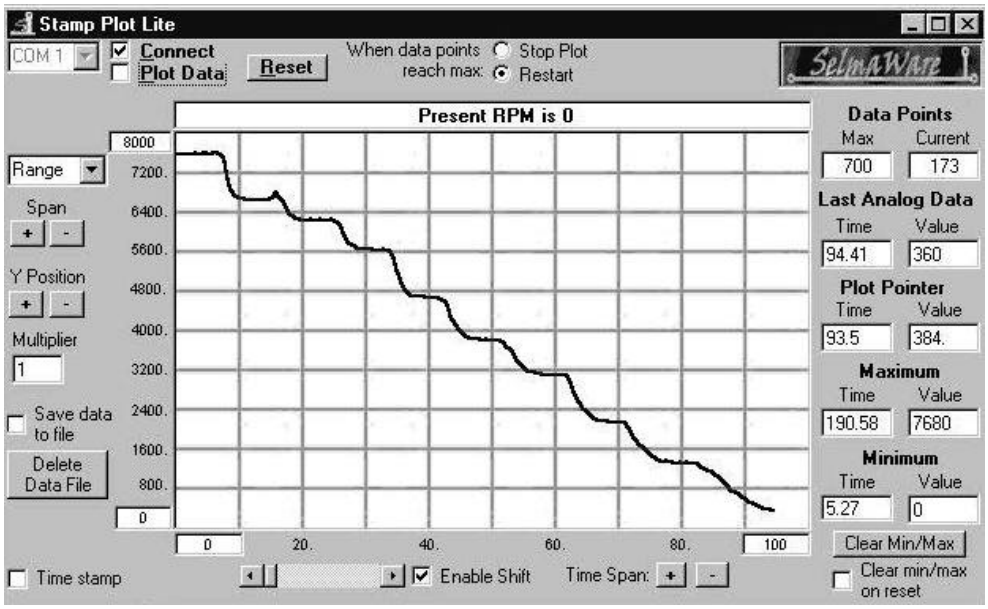
The first encoder wheel has one white and one black segment on it. As it rotates, the opto-switch should cycle on-off once for each revolution. Enter the Tachometer Test Program 2.7 below.

```
' Program 2.7 Tachometer Test - with the StampPlot Interface

' Initialize plotting interface parameters.
' (Can also be set or changed on the interface)
debug "!AMAX 8000",13            ' Full Scale RPM
debug "!AMIN 0",13               ' Minimum scaled RPM
debug "!TMAX 100",13             ' Maximum time axis
debug "!TMIN 0",13               ' Minimum time axis
debug "!AMUL 1",13               ' Analog scale multiplier
debug "!PNTS 600",13             ' Plot 600 data points
debug "!PLOT ON",13              ' Turn plotter on
debug "!RSET",13                 ' Reset screen

Counts var word                  ' Variable for results of count
RPM var word                     ' Variable for calculated RPM
Counts = 0                       ' Clear Counts

Loop:
  count 3,1000, Counts           ' Count cycles on pin 3 for 1 second
  RPM = Counts * 60              ' Scale to RPM
                                 ' Send out RPM value to plotter and status bar
  debug dec rpm,13
  debug "!USRS Present RPM is ", DEC RPM,13
goto Loop
```

As the fan spins, the cycling of the photo switch will be counted for 1000 milliseconds (one second). With the duration of the **`Counts`** routine being one second and one cycle occurring with each rotation, we get the cycles per second of fan rotation. Most often, the speed of a rotating shaft is described in terms of revolutions per minute (RPM). Multiplying the revolutions per second times 60 converts cycles per second to RPM.

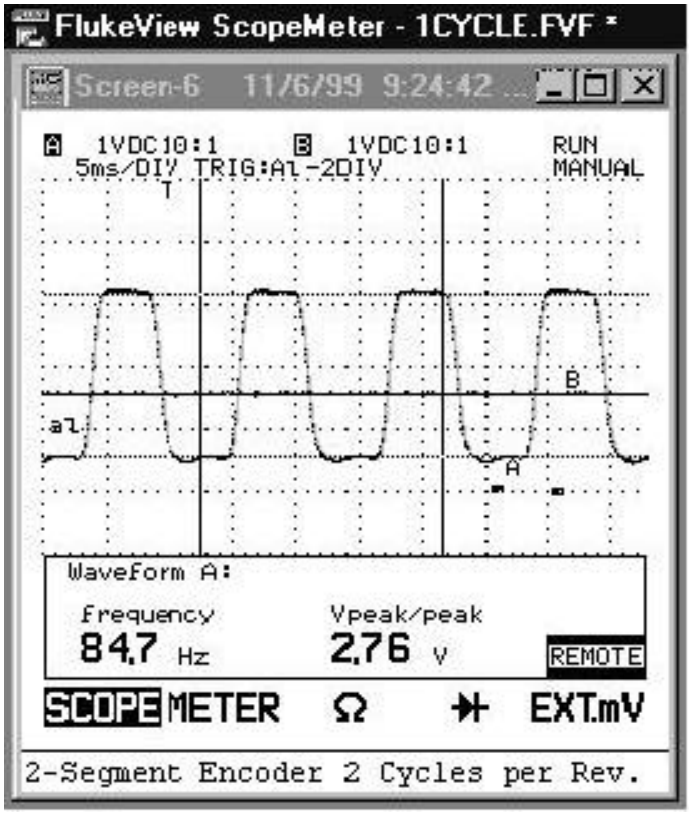
Run your program. The debug window will first appear with the serial information for configuration and display of the StampPlot Lite interface. Close the BASIC Stamp debug window and open StampPlot Lite. Check "Connect and Plot Data" and click on "Restart." Press the Board of Education reset button and your interface should start plotting. Figure 2.14 shows a representative screen shot of the interface plotting RPM at various motor voltages.

Figure 2.14: RPM of the Brushless DC Fan at Varying Voltages



The spinning encoder wheel may result in a slightly different phototransistor output for "light" and "no-light" conditions. If your system is not reporting correctly, change the setpoint by adjusting the potentiometer to the new average value. If you have access to an oscilloscope, measure the peak-to-peak output of the phototransistor and your potentiometer setpoint being applied to the comparator. Placing the setpoint midway between the peak-to-peak DC voltage levels would allow for optimal performance. Notice the frequency and wave shape of the signal. An example of the oscilloscope reading is pictured in Figure 2.15. The 84.7 Hz equated to a debug readout of "Counts = 84 RPM = 5040." The 84.7 Hz measured by the oscilloscope reflects an actual RPM of 84.7 x 60 = 5,082. Only 84 complete cycles fell within the one-second capture time of our routine.

Figure 2.15: Two-Segment Encoder Oscilloscope Trace



Record your tachometer readout when maximum voltage is applied to the motor. You can use the Board of Education's Vin (unregulated 9 V) for high speed, or the Vdd (regulated 5 V) for different speeds.

Counts = _____ RPM = _____

When testing your tachometer, notice the effects of slowing the motor with slight pressure from your finger. The counts will decrease by factors of one. In the Figure 2.13 example, it would decrease from 83 to 82 to 81, etc., and the resulting RPM readings drop by a factor of 60 (4980 to 4920 to 4860, etc.).

Because we are counting for one second and we get one cycle per revolution, the program can resolve RPM only to within an accuracy of 60. To get a more accurate assessment of RPM, you have a couple of choices: increase the time you count cycles, or increase the cycles per revolution.

Let's try the first choice. Increase the count time in Program 2.7 from 1000 milliseconds to 2000 milliseconds. By doing so, you are now reading during a two-second window and RPM is equal to 30 times this value. {(Counts/2 seconds) x 30 = RPM} and resolution is now to within 30 RPM. Change the scaling value for RPM from 60 to 30 in the **`RPM =`** line of the program.

Increasing the count duration time increases the accuracy of the RPM reading. Refer to Table 2.4.

Table 2.4: Given Encoder Frequency of 84.7 Hz
From the 1 cycle/second Encoder is an RPM of 5082

Duration	Counts	Scaler	RPM
1000 mS	84	60	5040
2000 mS	169	30	5070
3000 mS	254	20	5080
60000 mS	5082	1	5082

As you can see, to gain a resolution of one RPM, our count routine had to be one full minute (60,000) in duration. Unless you are very patient, this is unacceptable! In terms of programming, the BASIC Stamp is tied up with the **`count`** routine for the total duration. During this time, the rest of the program is not being serviced. For this reason, long duration also is not good.

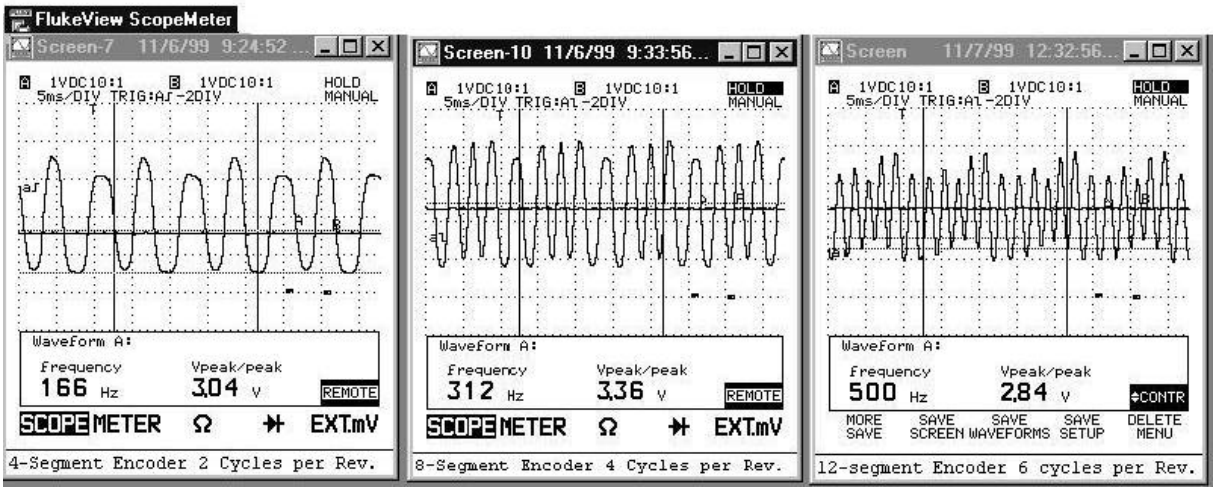
Another method of improving resolution is to increase the number of cycles per revolution. Cut out the second encoder wheel and tape it to your fan motor hub. This wheel has two white segments and will produce two count cycles per revolution. During a one-second-count duration, this encoder will produce twice as many pulses as the first encoder. The RPM calculation line of the code would be **`RPM = Counts x 60 / 2`** for this encoder, or **`RPM = Counts x 30`**. Try it!

The third encoder wheel yields even more resolution by with four cycles per revolution. Tape this encoder to your motor's hub and change the program's RPM line to **`RPM = Counts * 15`**. You may have to vary the setpoint potentiometer as you switch from one encoder wheel to another.

If you use the six-cycle encoder, what value would you use to scale the Counts to RPM? Fill in your answer in Table 2.5.

Figure 2.16 includes oscilloscope traces recorded from using the two-cycle, four-cycle, and six-cycle encoder wheels on a shaft rotating at 4,980 RPM. It is the focal properties of the emitter/detector pair that will limit the maximum number of segments on the encoder wheel. You may find it difficult to use the six-cycle encoder wheels without devising some sort of shielding and/or focusing of the light beam.

Figure 2.16: Two-cycle, Four-cycle, and Six-cycle Encoder Wheel Oscilloscope Traces



The accuracy required of a tachometer system is dependent on the application. Commercial shaft encoders are available with resolutions greater than 500 counts per revolution. Fill in the appropriate values in Table 2.5 for an encoder with a resolution of 360 counts per revolution.

Table 2.5: Given a Shaft Speed of 4,980 RPM

Cycles per Revolution	Counts	Scaler	RPM
1	83	60	4,980
2	166	30	4,980
4	312	20	4,980
6	498	_____	4,980
360	_____	_____	_____

Challenge! Analyze your Motor's Characteristics

It would be interesting to determine the speed voltage characteristics of your motor. Beginning at full voltage, adjust the motor's voltage to several different levels throughout its operating range. Allow the motor to stabilize at each voltage level and record those levels. Be quick or increase the time span of StampPlot Lite. Once you have gone through the range from full voltage to stall voltage, stop the plot. Use the mouse cursor to read the stable RPM at each step and record the RPM alongside its corresponding voltage. Plot your data and summarize the speed/voltage characteristics of your motor. Figure 2.14 is a screen shot resulting from performing this challenge. The motor we used had fair linearity for voltages from 6V to 12V. At low voltages, its linearity was poor, and it stalled at 3.4 V.

Questions and Challenge

Questions

1. An industrial device whose output is either one of two possible states is termed _____.

2. What is the "ideal" resistance of a mechanical switch in the open state? In the closed state?

   Open-state resistance = _____ and, Closed-state resistance = _____

3. Explain the purpose of placing a resistance in series with a switch for conditioning a digital input signal.

4. A normally-open pushbutton switch configured in an "active low" state will be read as a logic _____ when not being pressed.

5. What is the absolute maximum input voltage to the BASIC Stamp?

6. For some CMOS devices, an input of 1.3 volts is in the _____ area of operation.

7. Low-voltage logic devices operate on _____ volts DC.

8. What type of proximity switch activates only on metal objects?

9. When light strikes the base of a phototransistor, the collector current will _____ and collector to emitter voltage will _____.

10. A car's six-cylinder engine RPM can be determined by counting the pulses delivered to the ignition coil. Six pulses are required for one revolution. If 20 pulses occur in one second, what is the RPM of the engine?

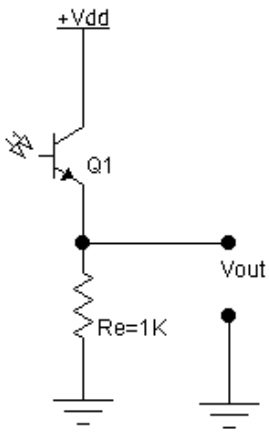
Design it!

1. Draw a diagram of a normally-open pushbutton switch and its "pull-up" resistor. The diagram should be drawn so pressing the switch results in a logic "low" output.

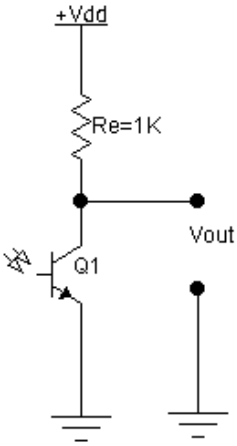
2. Draw a diagram of a normally-closed pushbutton switch and a 10K-ohm series resistor. The diagram should be drawn so pressing the switch results in a logic-low output.

Analyze it!

1. Consider the two phototransistor circuits below. Which one has an increasing output voltage with increases in light level? Why? What is the output voltage of Circuit B if the light level saturates transistor Q1?

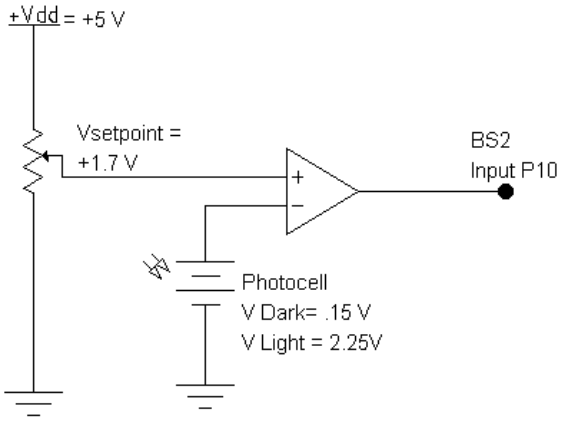


Circuit A



Circuit B

2. The comparator circuit below is used to determine when to turn on and off a dusk-to-dawn security lamp. What would be the output status of the comparator during "light" conditions? Would it be better to program for detecting the voltage level or the edge triggering of this circuit? Why?



Night time security light.

Program it!

1. Pretend that your retro-reflective tachometer is providing the input to an anti-lock braking system on an automobile. In conjunction with this input, use a pushbutton to model the brake pedal switch. An active high LED will represent the braking action. Write a program that will detect the pressing of the brake pedal and slow the motor, also turning on the LED as long as speed is above zero. When shaft speed drops to zero, turn off the LED. Use a potentiometer to set initial motor speed. Configure the two pushbutton switches as active-high inputs. Wire one LED as an active-high output.

2. Write programs to duplicate the operation of an OR, AND, and XOR gate.

OR Gate		
PB1	PB2	LED
0	0	0
1	0	1
0	1	1
1	1	1

AND Gate		
PB1	PB2	LED
0	0	0
1	0	0
0	1	0
1	1	1

XOR Gate		
PB1	PB2	LED
0	0	0
1	0	1
0	1	1
1	1	0

Field Activity

How many digital (bi-state) field devices can you identify in a new car? List as many as you can. Make a note as to whether you suspect that the field device directly controls load current , drives some sort of relay, or if you think its status is being monitored by a microcontroller.

Experiment #3: Digital Output Signal Conditioning

The outputs of a microcontroller are used to control the status of output field devices. Output devices are those devices that do the work in a process-control application. They deliver the energy to the process under control. A few common examples include motors, heaters, solenoids, valves, and lamps. The low-power output capability of the BASIC Stamp (or any microcontroller) prevents it from providing the power required by these loads. With proper signal conditioning, the BASIC Stamp can control power transistors, thyristors, and relays. These are the devices that can deliver the load current and voltage demands of the field devices. In some applications, you may use a BASIC Stamp output to communicate with another microcontroller or electronic circuit. There may be compatibility issues of different logic families, separate power supplies, or uncommon grounds that require special consideration. The focus of this experiment is to present some of the signal conditioning techniques used to interface your BASIC Stamp to output field devices.

Appropriate signal conditioning design begins with a brief look at the characteristics and limitations of the BASIC Stamp's outputs. The output of the BASIC Stamp is considered "standard TTL" level. As we discussed in Experiment #2, this means it can switch between logic high of approximately 5 volts or logic low of nearly 0 volts. According to the BASIC Stamp's datasheet, each output can sink 25 mA and source 20 mA of current. Relating this to the partial diagram in Figure 3.1, notice how the load can be connected. In Figure 3.1a, the load is wired from the output pin to ground. When you set an output pin high, five volts appear across the load resistor (RL). Load current will flow from ground through the resistor and into the output pin. This is the current source mode, and the BASIC Stamp can deliver a maximum of 20 mA to the load.

Figure 3.1: BASIC Stamp Output Pin Current Capability

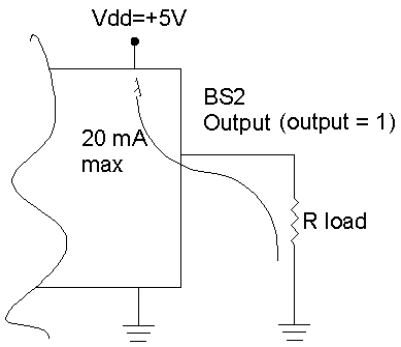


Figure 3.1a: Current Source

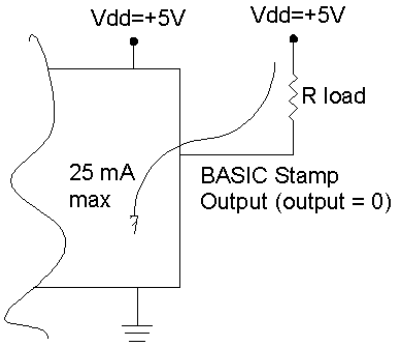


Figure 3.1b: Current Sink

In Figure 3.1b, the load is between the output pin and the +5-volt Vdd supply. Current will flow through the load now when the BASIC Stamp output pin is set Low (ground). Current will flow out of the output pin and up through the load resistor to Vdd. This is the current sinking mode, and the BASIC Stamp can deliver a maximum of 25 mA to the load when configured in this manner.

Output Capability of Digital Circuits

The output capability of digital circuits is listed in the manufacturer's datasheet. Devices usually can "sink" more current than they can "source." Some devices do not have the capability to source current because the internal path from their output to +V is not present. You may see this output design referred to as a device with an "open collector" output.

Outputs have been used to drive LEDs in previous exercises as pictured in Figure 3.2a. When the BASIC Stamp output is low, the diode is forward-biased at approximately one volt, and the remaining four volts, dropped across the 220-ohm resistor, limit current flow to approximately 22 mA. The light emitted by the diode gives visual indication of the output action. In previous programming challenges, you have assumed that the on-off status of an LED represents process action taking place. This is a valid assumption when you consider the operation of a solid-state relay (SSR). Figure 3.2b is a schematic representation of the solid-state relay. The input circuit (terminals 1-2) is equivalent to Figure 3.2a. The +3 to +24 V DC input identifies a range of control voltages. Control voltages must be above the minimum voltage to produce enough LED current for turn-on. Exceeding the maximum control voltage may cause damaging amounts of current to flow in the input LED. The light generated in the SSR strikes an optically controlled output circuit. The detail of this circuit is not shown, but is represented by the normally-open contact symbol. The current and voltage limitations of the output are listed in the device's datasheet and are usually printed on the device itself.

Figure 3.2: BASIC Stamp to LED and Solid-State Relay Schematics

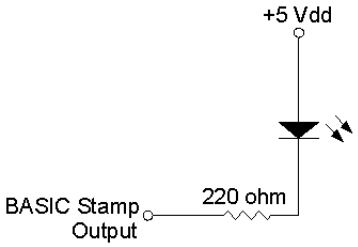


Figure 3.2a: BASIC Stamp to LED

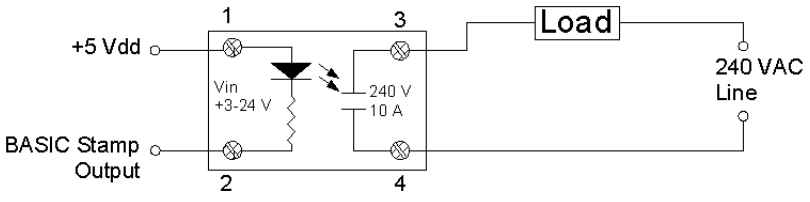


Figure 3.2b: BASIC Stamp to Solid-State Relay

Solid-state relays are available in a wide variety of output ranges. They may be designed to drive either AC loads or DC loads. The load you are driving defines the minimum specification of the SSR required.

An added benefit of the solid-state relay is electrical isolation. The BASIC Stamp is controlling the load by an optically-coupled signal. There is no electrical connection between the microcontroller and the high-power load device. Electrical failures of the load, or power line problems such as spikes, are not fed back to the BASIC Stamp. The datasheet for the Potter-Brumfield SSR may be found in Appendix C. Refer to this datasheet and find the following information.

Input voltage range: _____
Input current requirement @ five volts: _____
Maximum output load current: _____
Maximum output load voltage: _____
Electrical isolation: _____

A word of caution when selecting and implementing solid-state relays:

1. Do not push the specifications to their limits. Oversize the output capability of your selection by at least 20%.
2. Pay close attention to any heatsink requirements. Maximum load current capability is usually dependent upon incorporating a proper heatsink.
3. The load's supply source and all wiring and connections must be able to conduct the load's current. If relays are placed on the breadboard for prototyping, be aware that the breadboard traces are rated at only 1 amp.
4. Respect the output circuit voltage. Be sure all connections are solidly secured and correct before applying line voltage. There is the risk of electrical shock. Take measures to prevent contact with high-voltage potentials. Shield or encase these contacts. Clearly identify high-voltage potentials with appropriate labeling.
5. Some electronic relays will not contain an internal current-limiting resistor on the input. In these cases, an external current-limiting resistor must be added in series with the internal LED. The value of the resistor is based on your control voltage and the manufacturer's recommended input current specification.

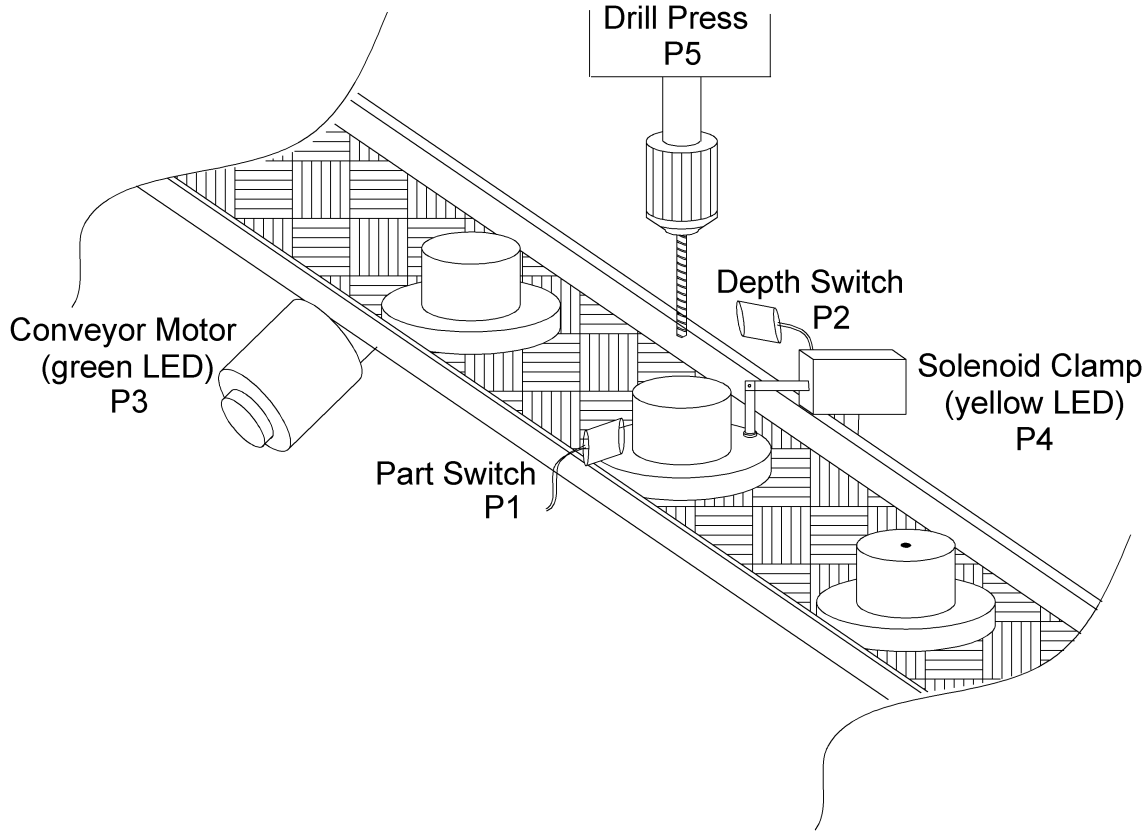
Solid-state relays provide an easy interface for controlling loads in an industrial application. Become familiar with the SSR datasheet specifications in order to make the right selection for your application.

Exercises

Exercise #1: Sequential Control

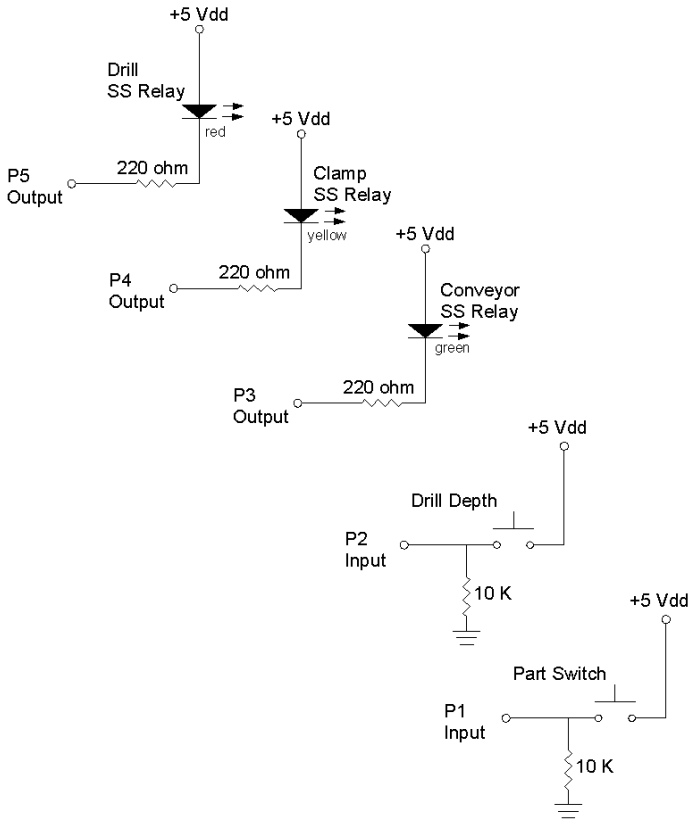
The BASIC Stamp is well suited to perform sequential control operations. Many processes depend on the orderly performance of operations. Consider the machining operation pictured in Figure 3.3a.

Figure 3.3a: AutoDrill Sequence Operation



3

Figure 3.3b: Sequential Control

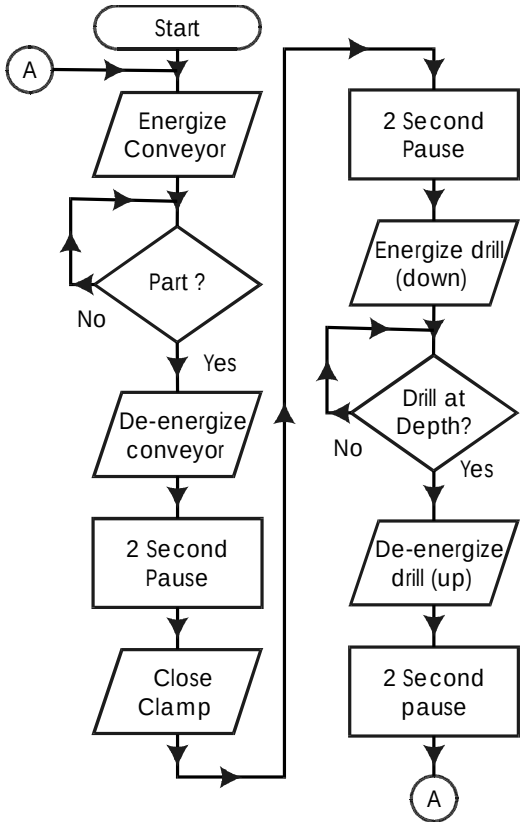


A conveyor is moving parts through a machining station. When a part is detected in the staging area, the conveyor is turned off. After a short pause, the solenoid clamp is activated to hold the part; another short pause, and then the drill is brought down to the part. A proximity switch detects proper depth of the hole. When the depth switch closes, the "drill down" command is stopped and the drill is retracted. After allowing a short time for the drill to retract, the clamp is released and the conveyor is started. This moves the processed part out of the staging area, and the conveyor continues until a new part is detected. Upon detecting another part, the sequential process continues.

With proper signal conditioning, the BASIC Stamp can easily control this sequence. For our exercise purposes, you are asked to use your imagination to allow LEDs to simulate the SSRs that could control the conveyor, clamp, and drill. Two pushbuttons and your two fingers must simulate the part coming into position and the drill coming down to proper depth. Construct Figure 3.3b on your Board of Education. For easier identification, use your green LED for the conveyor, your yellow LED for the clamp, and the red LED for the drill.

Sequential control lends itself well to flowcharting. The time required to develop your flowchart will be quickly saved as you write your program. Compare the flowchart in Figure 3.4 to the description of the machining process.

Figure 3.4: Sequential Control Flowchart



Program 3.1 follows the flowchart very closely. Study the program; compare its structure to the flowchart. Enter the program and try it. Again, you will have to use your imagination to simulate our process.

When the program starts, the green LED will be on. This represents the conveyor starting. To simulate a part coming into the staging area, you must press and hold Pushbutton P1. The conveyor LED will instantly turn OFF and the yellow LED indicating the Clamp relay will turn on after a one-second delay. After the Clamp has had two seconds to secure the part, the drill will come down toward the part as indicated by the red LED. At this time bring another finger down to simulate the drill. Pushbutton P2 represents a proximity switch, which will indicate when proper drill depth has been reached. Your "drill" finger pressing P2 will be turned OFF; the red LED indicating the drill is retracting. Your finger now coming off of the P2 pushbutton indicates the bit has started retracting and two seconds will be allowed for the drill to clear the part. After this delay, the clamp will be opened (yellow light OFF) and the conveyor will start again. The part is completed and leaves the staging area. From this point the sequence starts again.

Run the program a few times. Other than the DEBUG report that a part has been completed, there is no need for your computer. Unplug the serial cable from the Board of Education and continue to simulate the sequential process. The BASIC Stamp could function as the "embedded controller" in this application. Wiring the actual field devices to the BASIC Stamp would allow it to continuously repeat the process.

After understanding this sequential process, we will redefine your two inputs and three outputs to simulate another operation. You will be challenged to develop the program necessary for this embedded control application.

```
'Program 3.1: Sequential Process Control Machining Operation - Embedded

INPUT 1                          ' Part Detection Switch
INPUT 2                          ' Drill Depth Switch
OUTPUT 3                         ' Conveyor motor relay (green)
OUTPUT 4                         ' Clamp solenoid relay  (yellow)
OUTPUT 5                         ' Drill press relay (red)

OFF con 1                        ' Current sink loads
ON  con 0                        ' Negative logic

OUT3 = OFF                       ' Initialize outputs off
OUT4 = OFF
OUT5 = OFF
Start:
  OUT3 = ON                      ' Conveyor on
  IF IN1 = 1 THEN Process        ' If pressed, start "Process"
```

```
GOTO START

Process:                          ' The process begins
  OUT3 = OFF                      ' Stop conveyor
  PAUSE 1000
  OUT4 = ON                       ' Begin clamping part in place
  PAUSE 2000                      ' Wait 2 seconds to turn drill on

Drill_down:
  OUT5 = ON                       ' Turns on drill and drill begins dropping
  IF IN2 = 1 Then Pull_drill      ' If drill is deep enough, pull drill
GOTO Drill_down

Pull_drill:
  OUT5 = OFF                      ' Turns off drill and drill retracts
  IF IN2 = 0 Then Drill_up        ' Indicates drill is moving up
GOTO Pull_drill

Drill_up:
  PAUSE 2000                      ' Continue pulling drill up for 2 seconds

Release:
  OUT4 = OFF                      ' Open clamp to release part
  PAUSE 1000                      ' Wait 1 second
  OUT3 = ON                       ' Conveyor on
  IF IN1 = 0 Then Nextpart        ' Finished part leaves process area
GOTO Release

Nextpart:
  PAUSE 1000                      ' Wait 1 second
  DEBUG "Part leaving clamp. Starting next cycle", CR
GOTO Start
```

The real beauty of microcontrollers is to have the capability of embedding all of the intelligence necessary to perform sophisticated control within the equipment.

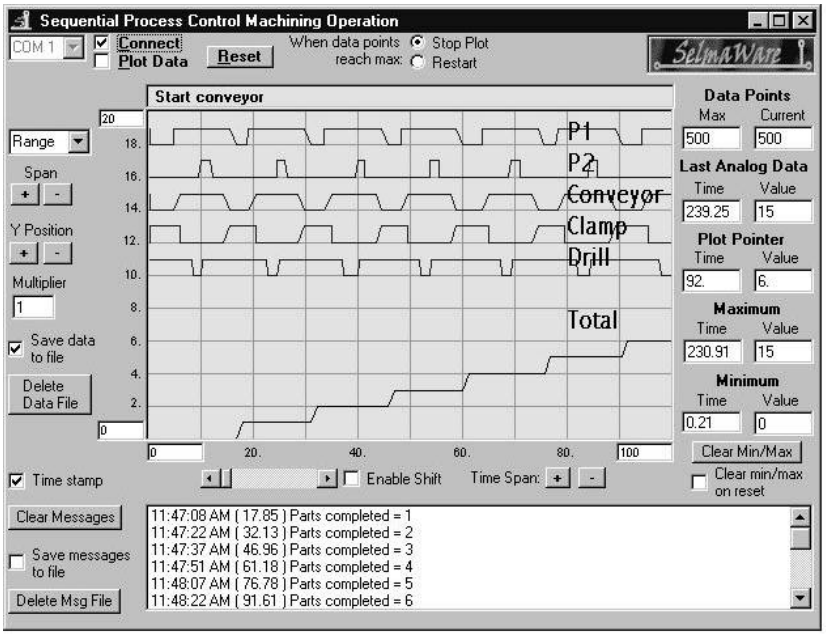
There are times, however, that being able to retrieve information from the microcontroller adds to its capabilities. The StampPlot Lite interface can be effectively used to monitor the sequential machining process. Program 3.2 uses this interface. The machine functions in Program 3.2 are the same as those in the previous program. **Debug** commands have been embedded to send data to the StampPlot Lite interface. The program plots the status of the digital I/O, reports process steps in the User status bar, and keeps a time-stamped list of the total parts produced. Figure 3.5 is a representative screen shot of the sequential process being monitored by StampPlot Lite. Load Program 3.2 and run it. Study the StampPlot Lite **debug** commands that have been added to the original program. Become familiar with their use. Graphical user interfaces such as this are very useful in the maintenance and data acquisition of embedded control systems. Use StampPlot to monitor the Sequential Control Mixing Challenge at the end of this section.

```
Program 3.2: Sequential Process Control Machining Operation with StampPlot
Interface

Pause 500
DEBUG "!TITL Sequential Process Control Machining Operation", 13
' StampPlot title
DEBUG "!TMAX 100", 13                  ' Set sweep plot time (seconds)
DEBUG "!PNTS 500", 13                  ' Sets the number of data points
DEBUG "!AMAX 20",  13                  ' Sets vertical axis (counts)
DEBUG "!CLRM", 13                      ' Clear List Box
DEBUG "!CLMM", 13                      ' Clear Min/Max
DEBUG "!RSET", 13                      ' Reset all plots
DEBUG "!DELD", 13                      ' Delete old data file
DEBUG "!PLOT ON", 13                   ' Turn Plot on
DEBUG "!TSMP ON", 13                   ' Time-stamp part completion
DEBUG "!SAVD ON", 13                   ' Save data to file

INPUT  1                               'Part Detection Switch
INPUT  2                               'Drill Depth Switch
OUTPUT 3                               'Conveyor motor relay (green)
OUTPUT 4                               'Clamp solenoid relay (yellow)
OUTPUT 5                               'Drill press relay    (red)

OFF con 1                              'Current sink mode
ON  con 0                              'Negative logic

OUT3 = OFF                             ' Initialize outputs off
OUT4 = OFF
OUT5 = OFF

Parts var byte
Parts = 0
```

```
Start:
  GOSUB Plot_data                        ' Plot the status
  OUT3 = ON                              ' Conveyor on
  DEBUG "!USRS Start conveyor",13        ' User status prompt
  IF IN1 = 1 THEN Process                ' If pressed, start "Process"
  PAUSE 100
GOTO START

Process:                                 ' The process begins
  GOSUB Plot_data                        ' Plot the status
  OUT3 = OFF                             ' Stop conveyor
  DEBUG "!USRS Detected part. Stop conveyor",13      ' User status prompt
  PAUSE 1000

  GOSUB Plot_data                        ' Plot the status
  OUT4 = ON                              ' Begin clamping part in place
  DEBUG "!USRS Clamp part.",13           ' User status prompt
  GOSUB Plot_data                        ' Plot the status
  PAUSE 2000                             ' Wait 2 seconds to turn drill on

Drill_down:
  GOSUB Plot_data                        ' Plot the status
  OUT5 = ON                              ' Turns on drill and drill drops
  DEBUG "!USRS Drill coming down!",13 ' User status prompt
  IF IN2 = 1 Then Pull_drill             ' If drill is deep enough, pull drill
  PAUSE 100
GOTO Drill_down

Pull_drill:
  GOSUB Plot_data                        ' Plot the status
  OUT5 = OFF                             ' Turns off drill and drill retracts
  DEBUG "!USRS Stop Drill and Retract",13
                                         ' User status prompt
  IF IN2 = 0 Then Drill_up               ' Indicates drill is moving up
  PAUSE 100
GOTO Pull_drill

Drill_up:
  GOSUB Plot_data                        ' Plot the status
  DEBUG "!USRS Drill coming up!!",13     ' User status prompt
  PAUSE 2000                             ' Pull drill for 2 seconds

Release:
  GOSUB Plot_data                        ' Plot the status
  OUT4 = OFF                             ' Open clamp to release part
  DEBUG "!USRS Clamp released. Conveyor moving.",13
'User status prompt
```

```
  PAUSE 1000                    ' Wait 1 seconds
  OUT3 = ON                     ' Conveyor on
  IF IN1 = 0 Then Nextpart
  GOTO Release

Nextpart:
  GOSUB Plot_data               ' Plot the status
  DEBUG "!USRS Part Complete. Start next cycle",13
' User status prompt
  Parts = Parts + 1             ' Parts counter
  PAUSE 1000                    ' Wait 1 seconds
  DEBUG "Parts completed = ", DEC Parts,13
                                ' Post parts count in the List Box
GOTO Start

Plot_data:
  DEBUG IBIN IN1,BIN IN2,BIN OUT3,BIN OUT4, BIN OUT5,13
'Plot the digital status.
  DEBUG DEC Parts,13            'Plot analog count
RETURN
```

Figure 3.5: Screen Shot of the Sequential Machining Process using StampPlot Lite



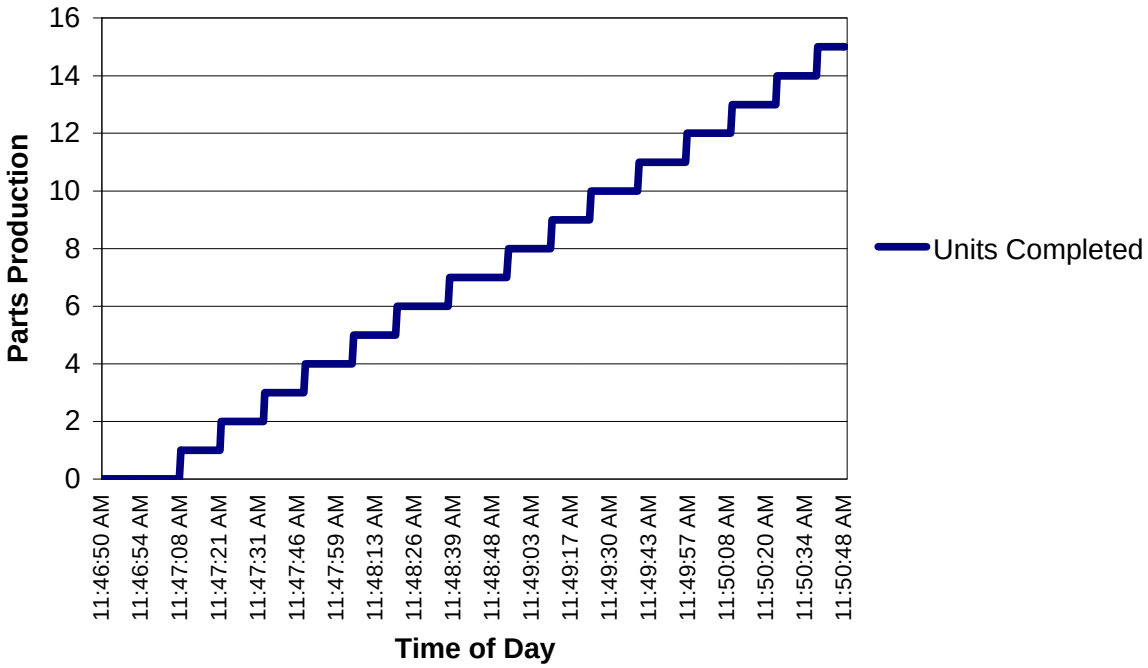
Note that the traces appear from top to bottom in the order which they were listed in the Debug digital plot command. Therefore, the top two traces are of the active high pushbuttons **IN1** (product in position) and **IN2** (depth switch). The next three traces are outputs **OUT3** (conveyor), **OUT4** (clamp), and **OUT5** (drill). Remember that the outputs are wired in the current sink mode. A High is OFF and a Low is ON.

Notice that in the initial setting for the StampPlot Lite interface, "Save data to file" (!SAVD) is ON. During the production run, the data at each sample point is saved into a text file, stampdat.txt. The data includes the time of day and program time that the sample was taken, the sample number, and the analog and digital values at the time of each sample. The data are comma delimited (separated by commas), and therefore, ready to be brought into a variety of spreadsheet or database software packages. Once the data is in the package, it is available for analysis and manipulation. Figure 3.6 represents a portion of the production run data, as it would appear in a Microsoft Excel spreadsheet. The complete file contains 500 samples (rows of data). Figure 3.7 is an Excel graph constructed from the data file.

Figure 3.6: Sequential Control Production Run (samples only)

Time of Day	Run Time	Sample number	Units Completed	Sample number	Digital Status
11:46:50 AM	0.21	1	1	1	111
11:46:50 AM	0.21	2	2	2	11
11:46:50 AM	0.21	3	3	3	11
11:46:50 AM	0.21	4	4	4	11
11:46:50 AM	0.27	5	5	5	11
11:46:50 AM	0.27	6	6	6	11
11:46:50 AM	0.27	7	7	7	11
11:46:50 AM	0.27	8	8	8	11
11:46:50 AM	0.27	9	9	9	11
11:46:50 AM	0.27	10	10	10	11
11:46:50 AM	0.32	11	11	11	11
11:46:50 AM	0.32	12	12	12	11
11:46:51 AM	0.43	13	13	13	11
11:46:51 AM	0.50	14	14	14	11
11:46:51 AM	0.71	15	15	15	11
11:46:51 AM	0.98	16	16	16	11
11:46:51 AM	1.26	17	17	17	11

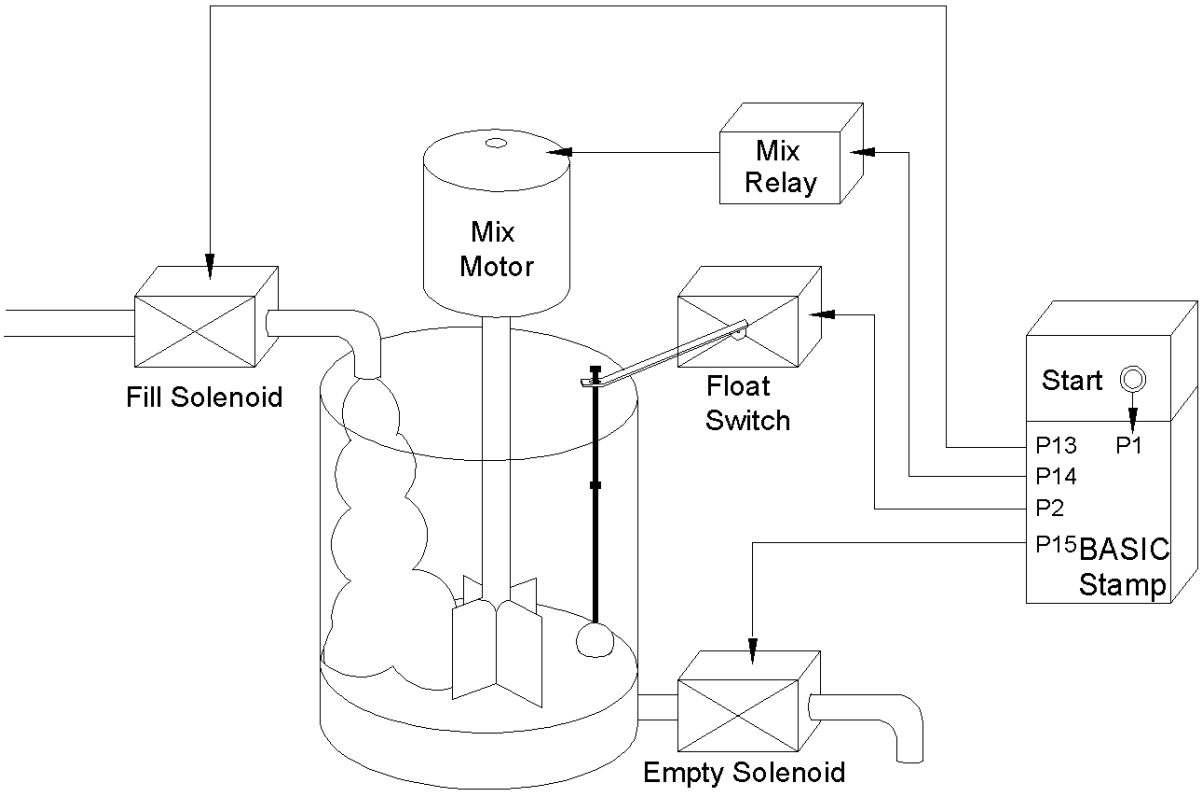
Figure 3.7: Graph of Sequential Control Production Run



Programming Challenge: Sequential Mixing Operation

A mixing sequence is pictured in Figure 3.8. In this process, an operator momentarily presses a switch to open a valve and begin filling a vat. A mechanical float rises with the liquid level and closes a switch when the vat is full. At this time, the "fill" solenoid is turned off, and a mixer blends the vat contents for 15 seconds. After the mixing period, a solenoid at the bottom of the vat is opened to empty the tank. The mechanical float lowers, opening its switch when the vat is empty. At this point, the "empty" solenoid is turned off and the valve closes. The process is ready for the operator to start another batch.

Figure 3.8: Mixing Sequential Control Process



Assign the following to the BASIC Stamp inputs and outputs to simulate the operation.

Operator pushbutton	Input P1 (N.O. active high)
Float switch	Input P2 (N.O. active high)
Fill Solenoid	Output P13 (red LED)
Mix Solenoid	Output P14 (yellow LED)
Empty Solenoid	Output P15 (green LED)

Construct a flowchart and program the operation.

Exercise #2 Current Boosting the BASIC Stamp

The BASIC Stamp's output current and/or voltage capability can be increased with the addition of an output transistor. Consider the circuit in Figure 3.9a. The circuit values should be designed such that a high (+5V) output of the BASIC Stamp drives Q1 into saturation without drawing more current than the BASIC Stamp can source.

Figure 3.9: Sequential Machining Process Schematic

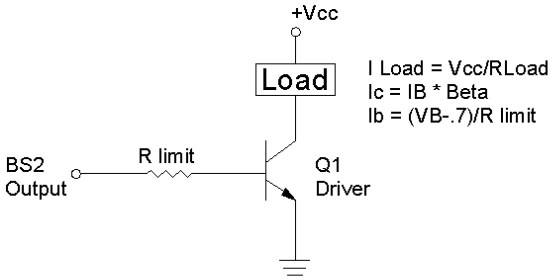


Figure 3.9a: BASIC Stamp to Transistor Driver

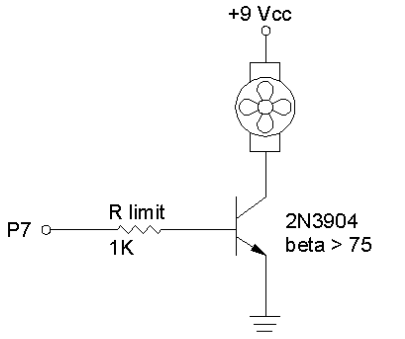


Figure 3.9b: BASIC Stamp Driving the Fan

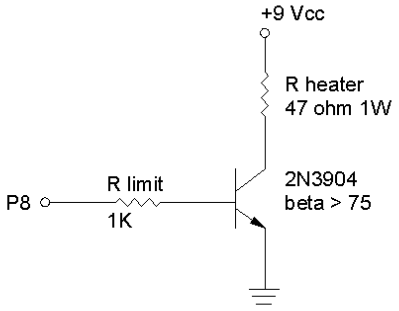


Figure 3.9c: BASIC Stamp Driving the Heater

Circuit component values stem from the load current and voltage requirements. The process of determining minimum component values is as follows: Since Q1 acts as an open collector current sink to the load, the load's supply voltage is not limited to the BASIC Stamp's +5-volt supply. If separate supplies are used, however, their common ground lines must be connected. When Q1 is driven into saturation, virtually all of the supply voltage will be dropped across the load and the load current will be equal to Vss/R_{load}. Q1's maximum collector current capability must be higher than this load current. The Q1 base current required to yield the collector current may be calculated by dividing the load current by the "beta" of Q1. $Ib = Ic/\beta_{Q1}$. Given a 20 mA maximum BASIC Stamp output current, a minimum transistor beta may be calculated by rearranging this formula. $\beta_{Q1(min)} = Ic/Ib$. Where Ib is the 20 mA maximum BASIC Stamp drive current.

A transistor must be chosen that meets, and preferably exceeds, these minimum requirements. Exceeding the minimum values by 50 to 100% or more would be best. Once the transistor is chosen, an appropriate base-limiting resistor value can be determined. This value must allow more base current than that defined by $Ic/\beta_{Q1,}$ yet less than the 20 mA BASIC Stamp limit. The voltage drop across R_{limit} is equal to the +5 V BASIC Stamp output minus the PN junction drop of Q1 (approximately +5V-.7V, or 4.3V).

For this exercise, and upcoming experiments, we have two loads that we wish to drive in this manner. They are a brushless DC fan and a 47-ohm, half-watt resistor. The brushless fan specifications include a full line voltage of +12 V and line current of 100 mA. The resistor will draw approximately 190 mA when powered by the +9 V Vin power supply.

Following the procedure outlined above, the transistors must handle collector currents of at least 190 mA and have a beta specification greater than 10. Figures 3.9b and 3.9c represent a 2N3904 as Q1. This transistor has collector current capability of 300 mA and a minimum Beta of 75. Added features to the selection of this transistor include that it is very common, it is inexpensive, and it can deliver the load current without need of a heat sink. R_{limit} values were selected based on minimum beta specifications and a desire to keep the BASIC Stamp output current demand well below 20 mA. These 1K-ohm resistors allow approximately 5 mA of base current, which ensures saturation.

Construct the two transistor-driver circuits in Figure 3.9b and 3.9c on your Board of Education. To test the devices, enter the short Program 3.3 to configure Pin 7 and Pin 8 as outputs, and turn them on.

```
Program 3.3: Current-boost for the fan and heater.

Output 7            ' Output for Fan drive
Output 8            ' Output for Heater drive
Out7 = 1            ' Fan ON (active high)
Out8 = 1            ' Heater ON (active high)
```

The fan will be running at full speed, and the resistor will be warming up due to the current flowing through it. Use your multi-meter to measure the collector emitter voltage of each transistor. If they are fully saturated, this voltage should measure less that 300 mV. If the transistors are not fully saturated, the R_{limit} value can be reduced.

In the next experiment, we will use the resistor to simulate a heating element. The fan will simulate a process disturbance that cools the heater. Our objective will be to investigate various types of control to maintain a constant temperature. Leave these circuits constructed on your Board of Education.

Before we leave this exercise, it is worth mentioning some other interfacing challenges that you may be confronted with as a designer. Consider the circuits in Figure 3.10.

Figure 3.10: BASIC Stamp Output Interfacing

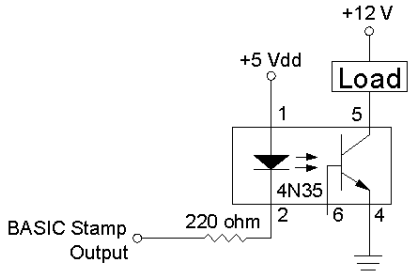


Figure 3.10a: BASIC Stamp to Optocoupler

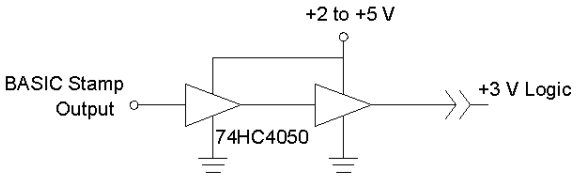


Figure 3.10b: BASIC Stamp to HCMOS

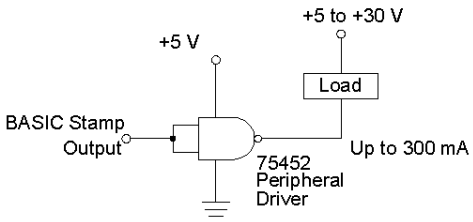


Figure 3.10c: BASIC Stamp to High Current Driver

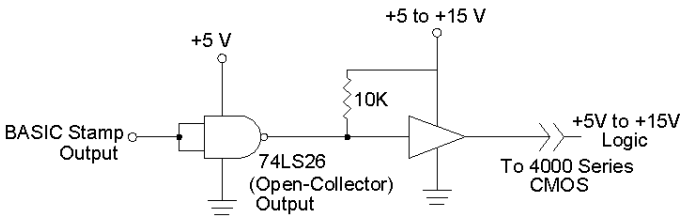


Figure 3.10d: BASIC Stamp to +15 V Logic

(a) The opto-coupler can be used to interface different voltages and to electrically isolate an output from the microcontroller circuit in Figure 3.10a.

(b) Figure 3.10b can be used to interface to HCMOS or 4000-series CMOS devices. The 74HC4050 can be operated on low voltages, allowing interfacing to +3-volt logic.

(c) There is a large variety of peripheral driver chips available. The 75452 driver depicted in Figure 3.10c can sink up to 300 mA of load current. Its open-collector output allows for loads up to 30 volts.

(d) Figure 3.10d includes the 74LS26 NAND gate. This is one of a family of open-collector gates. With the 10K-ohm pull-up resistor referenced to the next circuit stage, the BASIC Stamp can be interfaced to higher-voltage CMOS circuits.

Questions and Challenge

Questions

1. Output field devices are those devices that do the _____ in a process control application.

2. Field devices usually require more power than the BASIC Stamp can deliver. List three power interface devices that can control high-power circuits and be turned on and off by the BASIC Stamp.
   a. _____
   b. _____
   c. _____

3. The BASIC Stamp output is acting as a current sink when the load it is driving is connected between the output pin and _____.

4. The BASIC Stamp can source _____mA per output.

5. Electronic and electromagnetic relays offer a level of protection to the microcontroller because they provide electrical _____ between the BASIC Stamp and the power devices.

6. The input circuit of an SSR is usually an _____ ,which provides light that optically triggers an output device.

7. The current rating of an SSR should be oversized by at least _____ percent of the continuous load current demand.

8. Maximum continuous current ratings of solid-state relays usually involve applying a _____ for proper heat dissipation.

9. _____ control involves the orderly performance of process operations.

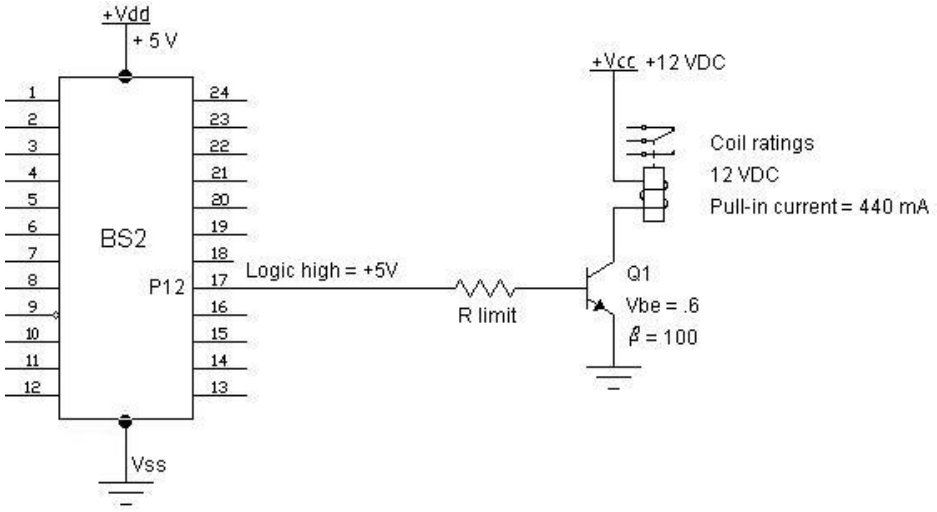
10. When the output current from the BASIC Stamp is not sufficient to turn on the control device, an output _____ may be used for current boosting.

11. The contacts of an electromagnetic relay are shown in schematics in the "normal" position. Normal means the relay's coil is _____ energized.

12. The "contacts" of an AC solid-state relay are actually the main terminals of a TRIAC. These contacts would be depicted in a schematic as being normally _____.

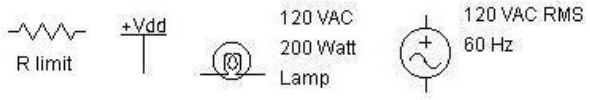
Design It!

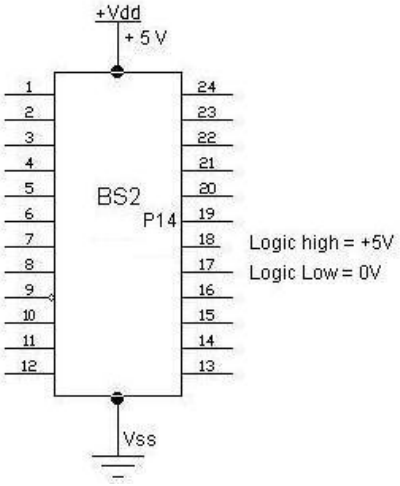
1. Given the figure below, solve for the maximum value of the base limiting resistor (Rlimit) that would allow the 440 mA of coil current to flow when the BASIC Stamp output Pin 12 is high.

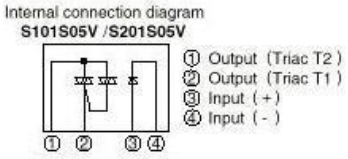


2. To ensure deep saturation of transistor Q1, the value of Rlimit should be _____ than this value.

3. The internal connection diagram of the SHARP S101S05V solid-state relay is given below. Notice that its input circuit is just an LED. The datasheet specifies that the LED has a forward voltage drop of 1.2 volts and that 15 mA through the LED will turn on the relay. Use the following components to complete the diagram for controlling the SSR with Pin 14 of the BASIC Stamp. Configure it as a current sink. Calculate the proper value of Rlimit. Draw the lamp and 120 VAC source as they would be connected to the SSR outputs.



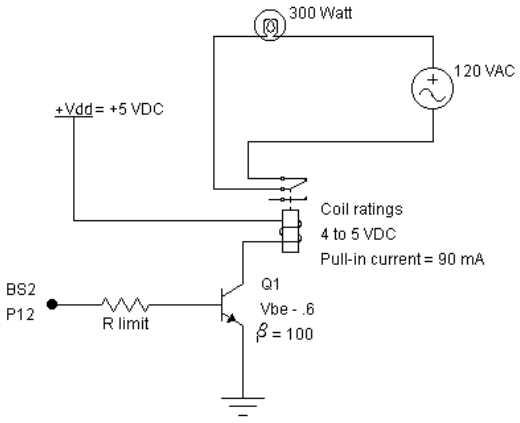




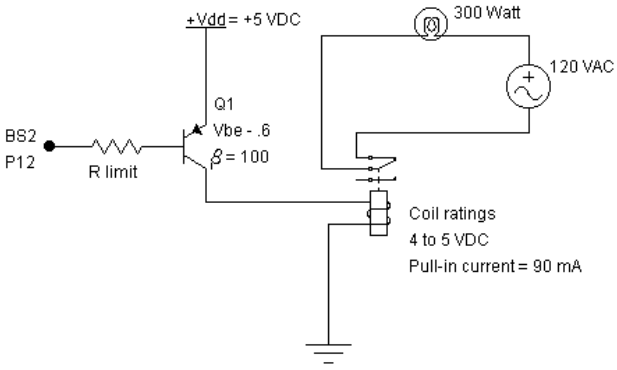
Analyze it!

1. Consider circuits A and B below. Write a line of BASIC Stamp code that will result in turning the lamp ON for each.

   Circuit A _____. Circuit B _____.



Circuit A: Current source drive and normally closed contacts.



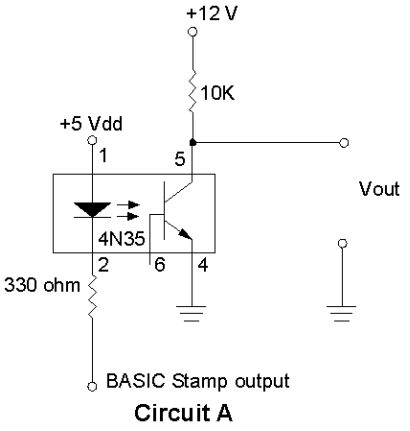
Circuit B: Current sink drive and normally open contacts.

2. Study the three figures shown below. Would you write a logic High or a logic Low to the BASIC Stamp output to yield a 12-volt Vout value?

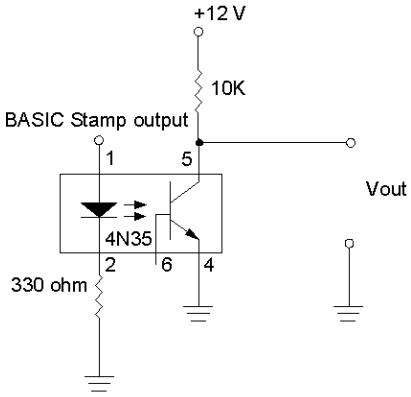
   Circuit A _____

   Circuit B _____

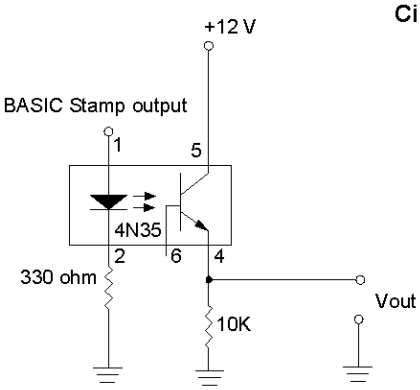
   Circuit C _____



Circuit A



Circuit B



Circuit C

Program it!

1. Given the input and outputs pictured back in Figure 3.3b, write a sequential program that will do the following:

   Momentarily pressing P1 will cause P3 to turn ON for three seconds and then go OFF. Pressing P1 a second time will cause P4 to come ON for three seconds and then go OFF. When P4 goes OFF, P5 will come ON until P2 is pressed.

2. Try this one. Using the same I/O, write a program that will do the following:

   Press and hold P1 and P3 goes ON. Holding P1 and pressing P2 causes P3 to go OFF and P4 to come ON. Releasing P1 while continuing to hold P2 turns OFF P4 and ON P5. And lastly, releasing P2 will turn all three outputs ON for three seconds, then all OFF, and the process is set to repeat.

Experiment #4: Continuous Process Control

Continuous process control involves maintaining desired process conditions. Heating or cooling objects to a certain temperature, holding a constant pressure in a steam pipe, or setting a flow rate of material into a vat in order maintain a constant liquid level, are examples of continuous process control. The condition we desire to control is termed the "process variable." Temperature, pressure, flow rate, and liquid level are the process variables in these examples. Industrial output devices are the control elements. Motors, valves, heaters, pumps, and solenoids are examples of devices used to control the energy determining the outcome of the processes.

The control action taken is based on the dynamic relationship between the output device's setting and its effect on the process. Generally speaking, process control can be classified into two types: open-loop and closed-loop. Closed-loop control involves determining the output device's setting based on measurement and evaluation during the process. In open-loop control, no automatic check is made to see whether corrective action is necessary.

A simple example of open-loop control would be cooling your bedroom on a hot summer evening. Your choices are using a window fan or an air conditioner. The window fan is a device that you set – low, medium, or high speed – based on your evaluation of what the situation needs for control. This evaluation involves an understanding of what the cause-and-effect relationship is of your speed setting vs. the room conditions. There is also an element of prediction involved. Once you make the setting decision, you are in for the night. You are setting up an open-loop control system. If your evaluations are correct, you will have a great night's sleep. If they are not, you may wake up shivering and cold or sweaty and hot! On the other hand, a room air conditioner allows you to set a certain desired temperature. A thermostat continuously compares the desired temperature with a measurement of actual room temperature. When room temperature is over the desired setpoint, the air conditioner is turned on. As the room cools below the setpoint, the air conditioner is turned off. As the night goes on and the outside temperature cools down, this closed-loop system will automatically spend less time on than off. This is an example of closed-loop feedback control, because the action is taken based on measurement of room temperature.

Which is better? Arguably, some people prefer air conditioning to a fan, but others do not. If the objective is to maintain a comfortable sleeping temperature, they both have their advantages. In terms of industrial control, the lower cost and simplicity of setting the window fan in an open-loop mode is very attractive. On the other hand, the automatic control of the closed-loop air conditioner ensures a more consistent bedroom temperature as the outside temperature changes.

Determining the best control action for an application and designing the system to provide this action is what the field of process control engineering is all about.

Microcontrollers have proven to be a dependable, cost-effective means of adding a level of sophistication to the simplest of control schemes. The next three exercises will focus on the characteristics of various methods of continuous control. We will develop an environment in which we can model process control, get process variable data into the BASIC Stamp, and study open-loop control principles. The first two items will take a little time and effort, but will be worthwhile, because the setup and circuitry will be used again for Experiments #5 and #6.

Temperature is by far the most common process variable that you will encounter. From controlling the temperature of molten metal in a foundry to controlling liquid nitrogen in a cryogenics lab, the measurement, evaluation, and control of temperature are critical to industry. The objective of this exercise is to show principles of microcontroller-based process control and enlighten you about interfacing the controller to real-world I/O devices. The exercises are restricted to circuits that fit on the Board of Education and to output devices that can be driven by its 9-volt, 300-mA power supply. As you monitor and control the temperature of a small environment, realize that through proper signal conditioning, the applications for which you can apply the BASIC Stamp are limitless.

Exercises

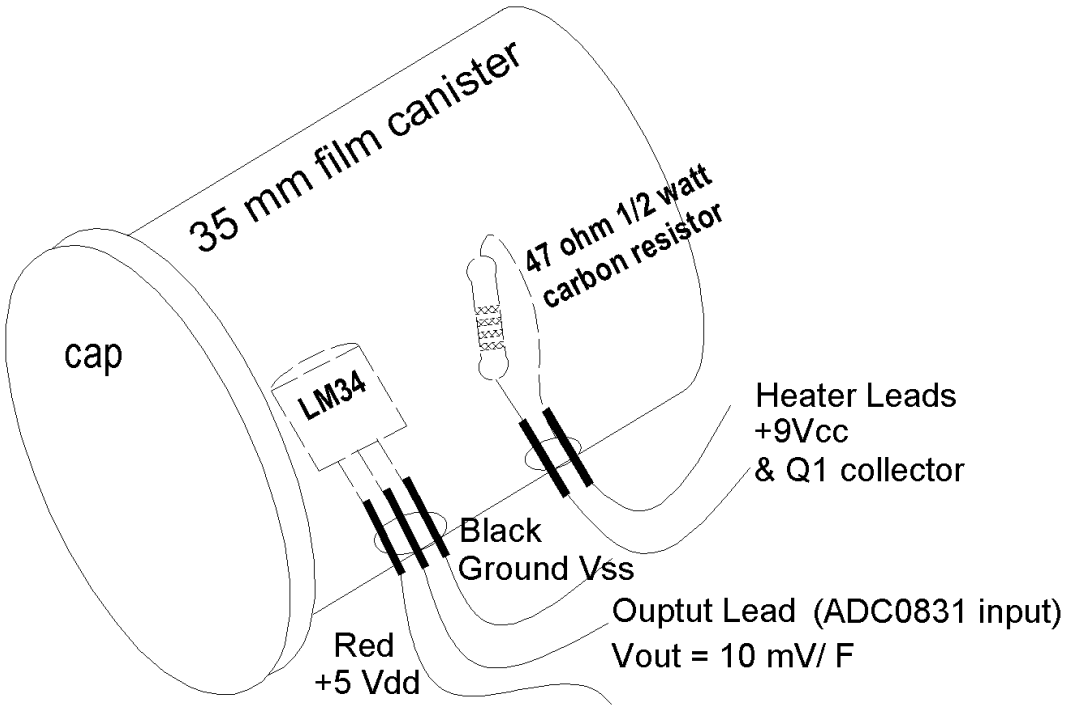
Exercise #1: Closed-Loop, On-Off Control

To set up our small environment, you will need the following parts:

- 35 mm plastic film container
- Six feet of 22 to 28 gauge hook-up wire
- 47-ohm, half-watt carbon resistor
- LM34DZ integrated circuit temperature sensor
- Electrical tape or heat-shrink tubing
- Soldering iron, solder, and wire cutter/strippers

You will put the 47-ohm resistor and the temperature sensor inside the 35-mm film canister. Leads from these devices will come back to the Board of Education. Placing the cap on the canister creates a closed environment. High current through the resistor will heat the environment, and the sensor will convert temperature to an analog voltage. A current-boost transistor from Experiment #3 will drive the resistor/heater, and you will add an analog-to-digital converter to get binary temperature information into the BASIC Stamp. Figure 4.1 depicts this construction step. Follow the procedures on the next page to construct the canister environment and signal-conditioning circuitry.

Figure 4.1: Film Canister Heated Environment



Preliminary Preparation

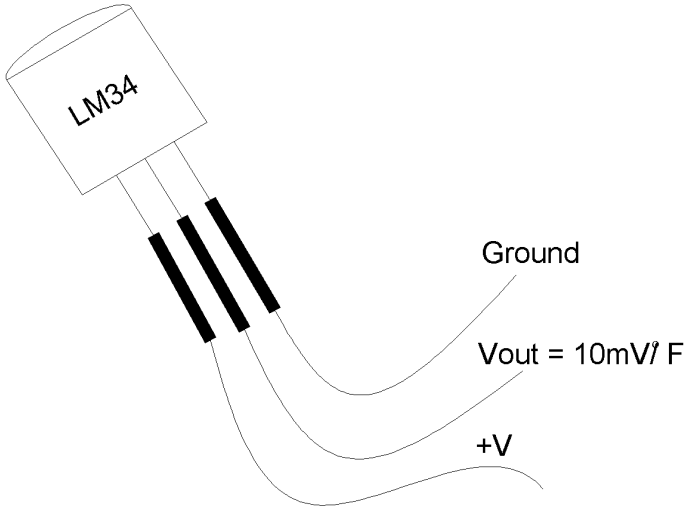
The 35-mm film canister will need two holes punched or drilled in it. A one-hole handheld paper punch could be used to create the holes. Place one hole as far toward the bottom as possible and the other near the top. The second hole should not be so near the top that it interferes with the cap. The sensor and the heater will be placed into these holes; some separation between the two is important.

In the last exercise, we ended with the ability to control the on-off status of a 47-ohm resistor acting as a heating element. The current-boost transistor acted as a switch, controlling the unregulated 9-volt supply to the resistor. As you should have seen, when the transistor turned on, the 9-volt supply was placed across the resistor and it became quite warm. It is no surprise considering that the power consumed is $P = V^2/R = 9^2/47$, or 1.7 watts! This is beyond the resistor's half-watt rating, but we are using it as a heater. It may become discolored, but should be all right otherwise.

Solder 12" lead wires onto the resistor so it can be placed in the film canister. A small amount of electrical tape or heat-shrink tubing over the connections should be used to avoid shorting in the canister. Place the resistor through the lower hole in the canister. Bend the leads and tape them down to the outside of the canister so the resistor is suspended in the middle of the canister.

Next, connect leads to the LM34 temperature sensor to act as a temperature probe. The LM34 precision IC temperature sensor is an excellent device in terms of its linearity, cost, and simplicity. Appendix D contains the datasheet for this very useful device. The sensor's output voltage changes 10 mV per degree Fahrenheit and is referenced at 0 degrees. With a DC power supply and a voltmeter, you have a ready-made Fahrenheit temperature sensor. Soldering some short leads onto the sensor and insulating them will result in a convenient probe. Twelve-inch lengths of red, black, and one other color wire will allow for easy identification of the +V, Ground, and Output leads. Short pieces of heat-shrink tubing or electrical tape should be used to insulate the leads from one another. Refer to Figure 4.2 and the device's datasheet in Appendix D.

Figure 4.2: LM34 Temperature Probe



Once your leads have been connected, test your temperature probe. Connect your probe to the +5-volt Vdd supply and ground. Use your voltmeter to monitor the LM34 output voltage of .01 volts per degree F. Simply move the decimal of the meter reading two places to the right to convert to temperature. For example; .75 V = 75 ^{o}F; .825 V = 82.5 ^{o}F; 1.05 V = 105 ^{o}F, etc. Then, try this:

- Complete your probe construction, then measure and record the room temperature.
- Hold the device between your fingers and watch the temperature rise.
- Hold it until the temperature becomes stable. How hot are your fingertips?
- The LM34 can measure temperatures up to 300 degrees. Briefly wave a flame under it and monitor higher temperatures.

Now, insert the sensor through the top hole of the film canister. Bend and tape its leads to the canister so the sensor is suspended inside. Cap your canister, and your model environment is complete. Keep in mind that although our laboratory setup is small and low power, it could represent controlling the temperature of a large kiln, a brewing vat, or an HVAC system. Appropriate output signal conditioning identified in Experiment #3 can allow the BASIC Stamp to control almost any industrial device.

Next, let's turn our attention to the Board of Education and set up the circuitry necessary for the experiment. Figure 4.3 represents the four circuits used in the exercises. All four circuits can fit on the Board of Education; you just have to be efficient with your use of the space. Figure 4.3a is a simple active-high pushbutton switch. It will be used to toggle the heater on and off. The output drive circuit depicted in Figure 4.3b also is very simple. Your board will contain the current-boost transistor (from Experiment 3) to drive the heater. Notice that an LED and current-limiting resistor have been added to indicate the status of this drive circuit. Note also that this circuit goes to the +5- Vdd supply, not the 9-volt unregulated supply.

Since the BASIC Stamp microcontroller does not have analog input capability, we must add an analog-to-digital converter to change the analog output of the LM34 temperature sensor to digital data. A one-step solution to signal conditioning is to use the serial analog-to-digital converter shown in Figure 4.3c. National's ADC0831 is a suitable A/D converter for our application, and will prove to be a very useful device for this and future applications. It's worth a moment at this point to look at the device's features and limitations.

This converter will take an input voltage and convert it to 8-bit digital data with a range of 256 possible binary representations. Potentiometers can be used to externally set the analog voltage range spanned by the 0 to 255_{10} digital output capability of the ADC0831. The voltage at Vin (-) Pin 3 sets the zero value. The Vref voltage at Pin 5 sets the voltage span above Vin (-) over which the resolution of 255 is spread.

Figure 4.3: Process Control Circuitry

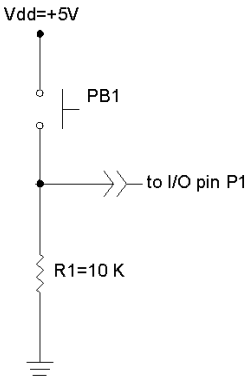


Figure 4.3a: Pushbutton

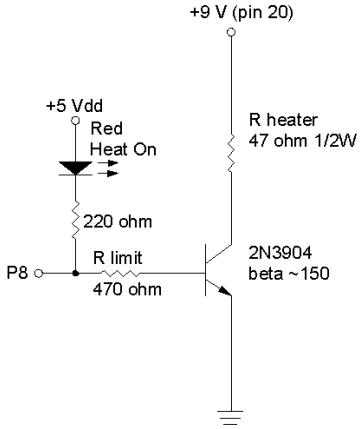


Figure 4.3b: BASIC Stamp Driving Heater with LED Indicator

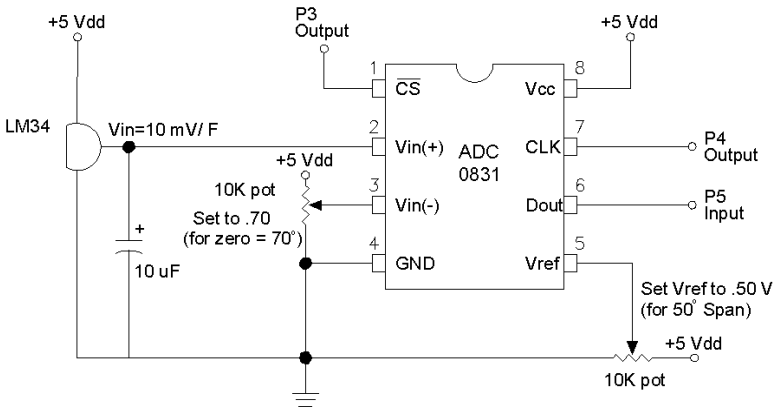


Figure 4.3c: ADC0831 Serial A-to-D

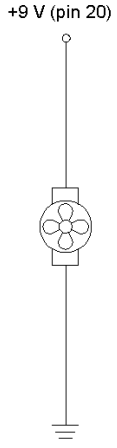


Figure 4.3d: Brushless Fan Directly Connected

These voltages are used to focus the range of the A/D device to cover the analog voltage being converted. The application will operate from room temperature (70^{o}) up to 120 ^{o}F. This temperature range equates to an LM34 voltage output of .70V to 1.20V. To maximize resolution, the span of interest is .5 V (1.2V-.7v); and the zero reference, termed the *offset*, is .7 V. Since we focus on just this range, each binary step represents approximately 0.2 degrees. This allows us to accurately resolve the temperature.

Construct the A/D converter circuit on the Board of Education. By using multi-turn trim potentiometers, the reference and span voltage levels can be set very accurately. Carefully measure and adjust these potentials. Attach the output of the LM34 to the input of the A/D converter.

Controlling the ADC0831 is relatively simple. A program control line tells the device to make a conversion. Once a conversion has been performed, the binary value can be output one bit at a time to the BASIC Stamp. The serial flow of data from the converter is controlled by output pins of the BASIC Stamp driving the "chip select" and "clock" lines. The chip select line (CS) is set low, followed by a low-to-high clock pulse. This starts the conversion. Subsequent clock pulses initiate the transfer of each binary bit starting with the most significant bit first. Parallax provides a convenient instruction, called **`shiftin`**, specifically designed for controlling synchronous serial communication. Once the binary data is clocked into the BASIC Stamp, it is converted to temperature, based on the zero and spanning values. (The use of the ADC0831 is detailed in the Parallax Basic Analog and Digital text, which can be used as a further reference to this section.)

Program 4.1 has been written to test your converter and driver circuits, and exercise the StampPlot Lite Interface. The first section of the program configures StampPlot Lite. A section that establishes variables and constants follows this. The program is designed for the circuit of Figure 4.3. Double-check the proper connections to I/O Pins 1,3,4,5 and 8. Accurately set the Zero and Span voltages of the ADC0831 to .7 and .5, respectively.

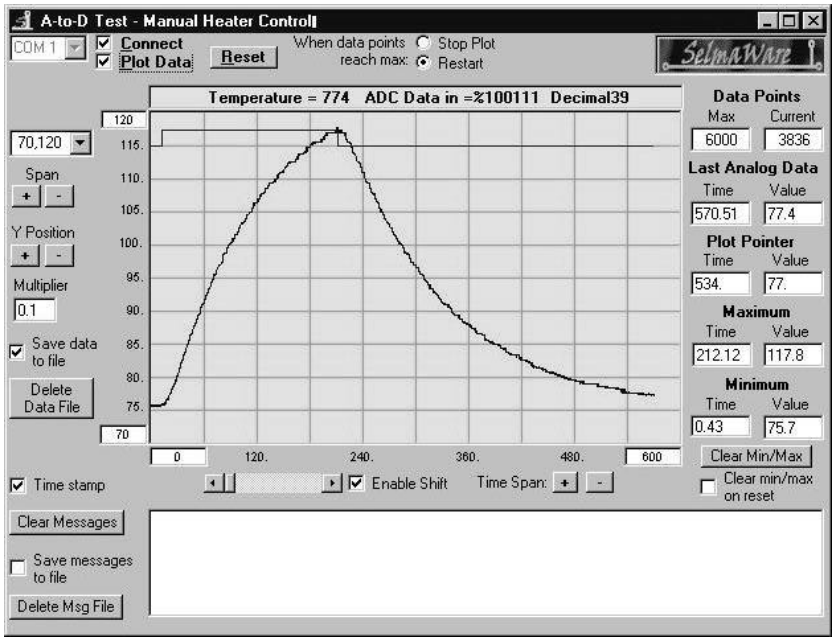
Load the program. Running the program will result in the DEBUG window opening and scrolling values and messages to the screen. Close the DEBUG window and open the StampPlot Lite Interface. At this point, select the appropriate COM port and check "Connect." Momentarily press the "Reset" button on the BASIC Stamp Board of Education to load the configuration and begin plotting the data. The user-status box will report the current temperature and the output of the A/D converter's binary and decimal values. The temperature and status of the heater will be plotted on the interface. Pressing PB1 will **`toggle`** the heater ON and OFF. (Feel free to omit the comments from this code, if you wish).

```
'Program 4.1: Analog-to-Digital & ON-OFF test with StampPlot Interface

'Pushbutton P1 toggles the heater fully ON and OFF.  It then establishes
'constants and variables used to acquire data from the ADC0831 serial A-to-D.
'StampPlot is used to graphically display results.  Program assumes that the
'circuitry is set according to Figure 4.3. ‘ADC0831: "chip select" CS = P3, "clock"
'Clk=P4, & serial 'data output"Dout=P5.  ‘Zero and Span pins: Digital 0 = Vin(-) =
'.70V and Span = Vref = .50V.


'Configure Plot
Pause 500                          'Allow buffer to clear
DEBUG "!RSET",CR                   'Reset plot to clear data
DEBUG "!TITL HEATER CONTROL SAMPLE",CR  'Caption form
DEBUG "!PNTS 6000",CR              '2000 sample data points
DEBUG "!TMAX 600",CR               'Max 300 seconds
DEBUG "!SPAN 70,120",CR            '50-300 degrees
DEBUG "!AMUL .1",CR                'Multiply data by .1
DEBUG "!DELD",CR                   'Delete Data File
DEBUG "!SAVD ON",CR                'Save Data
DEBUG "!TSMP ON",CR                'Time Stamp On
DEBUG "!CLMM",CR                   'Clear Min/Max
DEBUG "!CLRM",CR                   'Clear Messages
Debug "PLOT ON”,CR                 'Start Plotting
DEBUG "!RSET",CR                   'Reset plot to time 0

' Define constants & variables

CS con 3                           ' 0831 chip select active low from BS2 (P3)
CLK con 4                          ' Clock pulse from BS2 (P4) to 0831
Dout con 5                         ' Serial data output from 0831 to BS2 (P5)
Datain var byte                    ' Variable to hold incoming number (0 to 255)
Temp var word                      ' Hold the converted value representing temp

TempSpan var word                  ' Full Scale input span in tenths of degrees.
TempSpan = 5000                    ' Declare span.  Set Vref to .50V and
                                   ' 0 to 255 res. will be spread over 50
                                   ' (hundredths).

Offset var word                    ' Minimum temp.  @Offset, ADC = 0
Offset = 700
                                   'Declare zero Temp.  Set Vin(-) to .7 and
                                   'Offset will be 700 tenths degrees.  At these
                                   'settings, ADC output will be 0 - 255 for temps
```

```
                                'of 700 to 1200 tenths of degrees.

Wkspace1  var  byte             ' Workspace for the PB1's BUTTON command
Wkspace1 = 0                    ' Clear the workspace before using BUTTON
LOW 8                           ' Initialize heater OFF

Main:
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOSUB Display
  GOTO Main

Getdata:                        'Acquire conversion from 0831
  LOW CS                        'Select the chip
  LOW CLK                       'Ready the clock line.
  PULSOUT CLK,10                'Send a 10 uS clock pulse to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8]       'Shift in data
  High CS                       'Stop conversion
RETURN

Calc_Temp:                      'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset     'temp based on Span &
RETURN                    'Offset variables.

Control:                  ' Manual heater control
  Button 1,1,255,0,Wkspace1,1,Toggle_it
RETURN

Toggle_it:
  TOGGLE 8
  RETURN

Display:                  'Plot Temp, binary ADC, & Temp status
  DEBUG DEC Temp,CR
  DEBUG IBIN OUT8,CR
  DEBUG "!USRS Temperature = ", DEC Temp,"   ADC Data in =%", BIN Datain,  "
Decimal", DEC Datain, CR
RETURN
```

The StampPlot Lite interface will give you a dynamic representation of temperature changes in your canister. Toggle the heater ON and OFF and watch the response. The screen shot in Figure 4.4 represents the closed canister heating to 120 degrees and then cooling after the heater is turned off. Play with your system to become more familiar with its response; then, let's take a little closer look at the subroutines that make up the program.

Figure 4.4: Screen Shot using Program 4.1

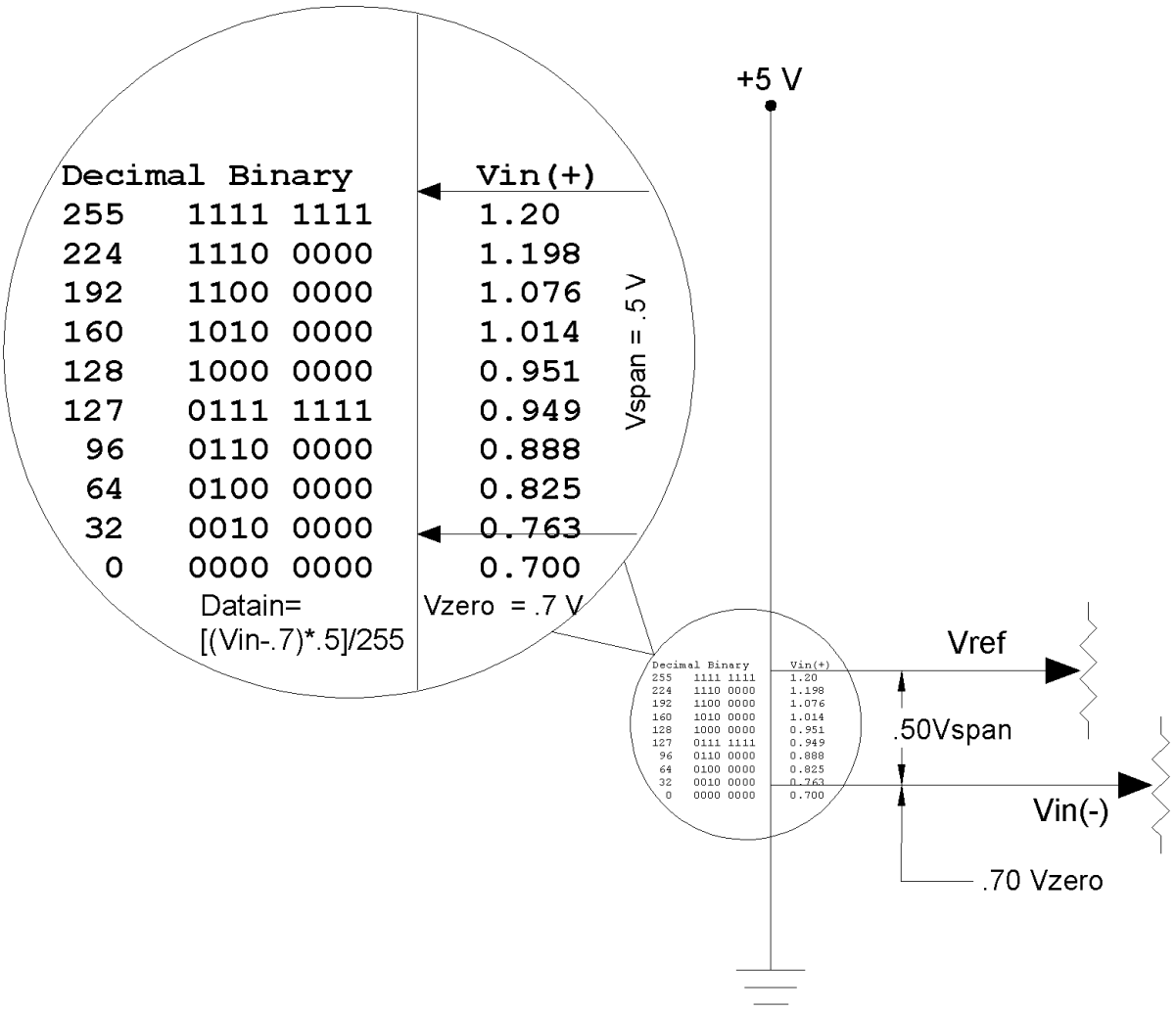


The main loop of this program simply executes three subroutines, **`Getdata`**, **`Calc_Temp`**, and **`Display`**. When running, the BASIC Stamp jumps back to the **`Getdata`** subroutine first. The last line of this routine instructs the processor to **`RETURN`** to the main loop and executes the next instruction, **`GOSUB Calc_Temp`**. The **`Calc_Temp`** subroutine executes, and it ends with a return. The BASIC Stamp returns to **`GOSUB_Display`**. After **`Display`** executes, its **`RETURN`** goes back to the instruction of **`GOTO Main`** and the process starts over. This is an organized approach to structuring our program. Later, when we include evaluation and control in our program, we simply add another subroutine, such as **`GOSUB_Control.`**

Let's take a closer look at the two primary subroutines of program 4.1. The **`Getdata`** subroutine begins with a high-to-low transition on the "chip select" line. This readies the A/D for operation.

The **LOW CLK** and **pulsout CLK,10** instructions tell the A/D converter to make a conversion of the Vin(+) voltage at this time. The ADC0831 is an 8-bit successive approximation converter. It's 256 possible digital combinations are spread over a voltage range determined by the potentials at the $V_{in(-)}$ and V_{ref} pins. $V_{in(-)}$ defines the voltage for which 0000 0000 would be the conversion. V_{ref} defines the range of input voltages above this point over which the other 255 digital combinations are spread. Figure 4.5 represents the Zero and Span settings for our application.

Figure 4.5: Zero and Span Settings for Our Application



With these settings, the ADC0831 is focused on a temperature range of 70 to 120 degrees. There can be an infinite number of possible temperature values within the .7 to 1.2-volt output range of the LM34. Only a few representative values are given. Since the 8-bit A/D converter has a resolution of 255, it can resolve this range of 50-degree temperatures to within .31 degrees. The conversion will be a binary number equal to [(Vin - .7) /.5] *255. Let's try a value within the range. Let's say the temperature is 98.6, which results in an LM34 output of .986 volts.

If Vin = .986, what would be the binary equivalent?

> [(.986-.7)/.5] *255 = 145.86. The answer is truncated to the whole integer of 145.
> The binary word would be 1001 0010.
> The binary conversion will be held and ready for transfer.

The **`shiftin`** instruction is designed for synchronous communication between the BASIC Stamp and serial devices such as the ADC0831. The syntax of the instruction is **`SHIFTIN dpin, cpin, mode, [result\bits]`**. The parameters indicate:

- which pin data will arrive on (dpin),
- which pin is the clock (cpin),
- which bit comes first, the least significant (LS) or most significant (MS), and on which edge of the clock it is released, rising (PRE) or falling (POST),
- and, what the word width is and where you want it stored [Datain\8].

For our system, we previously declared Pin 5 as dpin and Pin 4 as the clock (CLK) pin. The ADC0831 outputs the most significant bit first on the trailing edge of the clock. Therefore, MSPOST is the mode. And, finally, the 8-bit data will be held in a byte variable that we declared as **`Datain`**.

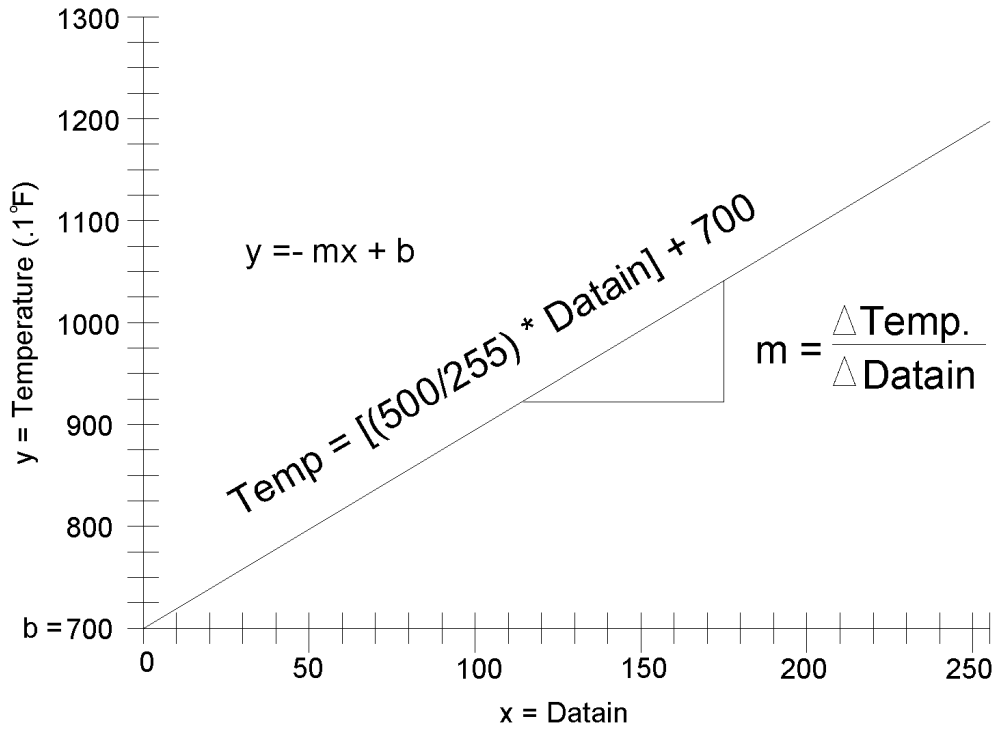
After the binary data is brought into the BASIC Stamp, it is available for our program to use. It would be most convenient to use if it were expressed in terms of the actual measurement units. For our application, that would be in degrees. The next subroutine, **`Calc_Temp`**, does just that. By knowing the zero and span transfer function of the conversion process, we use the standard y = mx + b formula. Where: y =Temperature, m = slope of the transfer function, and b is the offset. Temperature will be resolved and expressed in tenths of degrees.

Refer to the Calc_Temp formula: **`Temp = Tempspan/255 * 255/10 + 700`**

To increase the accuracy in resolving the slope (m), the **Tempspan** variable is scaled up by 10, to 5000 hundredth degrees. The slope is therefore, 5000/255 ~ 19 or .19 degrees per bit. Multiplying 19 times **Datain** tells you how far the measurement is into the span. This is in one one-hundredth of a degree at this point; therefore, divide by 10 to scale it back to tenths. Adding this to the Zero value of 700 (70 degrees) results in the actual temperature in tenths of a degree. Resolution is approximately .2 degrees over a range of inputs from 70.0 to 120.0 degrees.

The graph in Figure 4.6 plots the transfer function of the input A/D decimal equivalent input to temperature of the canister. Changing the span of coverage changes the slope of the transfer function. Changing the Zero value changes the y intercept.

Figure 4.6: Transfer Function



An additional word of caution about the BASIC Stamp math operation:

- A formula will be executed from left to right unless bracketing is used to set precedence.
- At no point can any subtotal exceed 32,759 or -32,760.
- Also, all remainders will be truncated, not rounded up.

Challenge: Change the Zero and Span voltages and edit the program to match the new range.

1. Your system should be able to raise the temperature of the closed canister beyond the 120-degree limit set by Program 4.1. Change the Zero and Span potentiometers for coverage of a temperature range from 75 degrees to 200 degrees. This allows for a wider range of coverage, but what is the resolution of your system now? Be patient, and let your system stabilize. Record the maximum temperature of your system.

2. Set the Zero and Span of your system to focus on the very narrow range of one degree below your room temperature to four degrees above it. Set the **`Calc_Temp`** variables to display in hundredths of degrees. Track these changes by leaving the cap off of the canister and simply touching the sensor with your warm finger. As you see, the resolution is great, but the trade-off is a decreased range of operation.

Having the ability to control the span and reference of the ADC0831 allows you to focus on a range of analog input. This helps maximize the resolution and accuracy of your system. The following exercise will require the original range of 70 to 120 degrees. Return the Zero and Span potentiometers back to .7 and .5 volts, respectively.

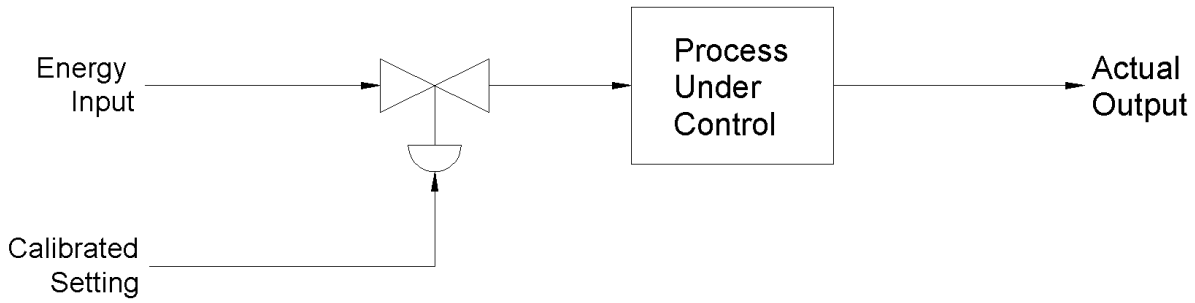
Now, after all of that, we can get back to a study of control theory!

Exercise #2: Open-Loop vs. Closed-Loop Control

Open-Loop Control

The simplest form of control is open loop. The block diagram in Figure 4.7 represents a basic open-loop system. Energy is applied to the process through an actuator. The calibrated setting on the actuator determines how much energy is applied. The process uses this energy to change its output. Changing the actuator's setting changes the energy level in the process and the resulting output. If all of the variables that may affect the outcome of the process are steady, the output of the process will be stable.

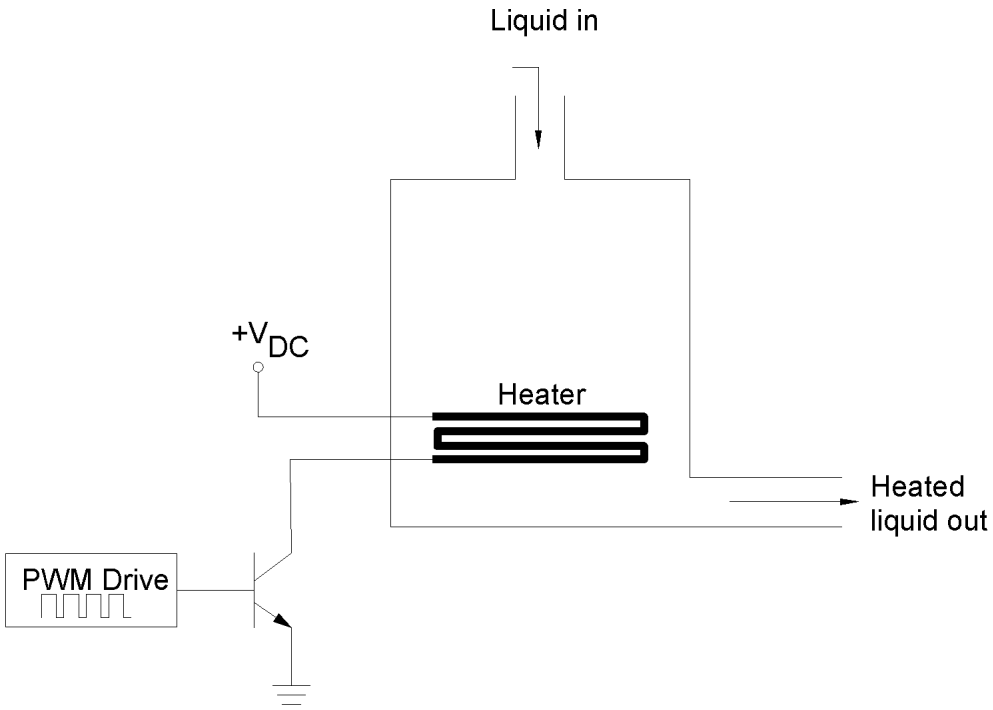
Figure 4.7: Open-Loop Control



The fundamental concept of open-loop control is that the actuator's setting is based on an understanding of the process. This understanding includes knowing the relationship of the effects of the energy on the process and an initial evaluation of any variables disturbing the process. Based on this understanding, the output "should" be correct. In contrast, closed-loop control incorporates an on-going evaluation (measurement) of the output, and actuator settings are based on this feedback information.

Consider the temperature control process shown in Figure 4.8. The material being drawn from the tank must be kept at a 101^{o} temperature. Obviously, this will require adding a certain amount of heat to the material. The drive on the transistor determines the power delivered to the heating element. The question becomes "How much drive is necessary?"

Figure 4.8: Open-Loop Heating Application



For a moment, consider the factors that would affect the output temperature. Obviously, ambient temperature is one. Can you list at least three others? How about:

- The rate at which material is flowing through the tank.
- The temperature of the material coming into the tank.
- And, the magnitude of air currents around the tank.

These are all factors that represent BTUs of heat energy taken away from the process. Therefore, they also represent BTUs that must be delivered to the process if the desired output is to be achieved. If the drive on the heating element were adjusted to deliver the exact BTUs being lost, the output would be stable.

In theory, the drive level could be set and the desired output would be maintained continuously, as long as the disturbances remained constant.

Let's now assume that it is your objective to keep the interior of your film canister at a constant temperature. A good real-world example would be that of an incubator used to hatch eggs. To hatch chicken eggs, it is important to maintain a 101°F environment.

Turning on the heater will warm up the interior of the canister. In our earlier test, you turned on the heater's drive transistor, and the temperature rose above 101°F. Obviously, to maintain the desired temperature, we will not need to have full power applied to the resistor. Through a little testing, you can determine just what drive level would be needed to yield the correct temperature.

The drive to the power transistor in Figure 4.8 is labeled as PWM. This is the acronym for pulse-width modulation. PWM is a very efficient method of controlling the average power to loads such as heating elements. The square wave is driving the transistor as a current-sinking switch. When the drive is high, the transistor is saturated, and full power is applied to the heater. A logic Low applied as base drive puts the transistor in cutoff; therefore, no current is applied to the load. Multiplying the percentage of the total time that the load receives full power times the full power will give the average power to the load. This average on-time is the duty cycle and is usually stated as a percentage. A 50% duty cycle would equate to half of the full power drive, 75% duty cycle is three-quarters full power, etc. It was stated earlier that the 47-ohm "heater" resistor in our canister would receive 1.7 watts when fully powered by the 9-volt unregulated source supply. The pushbutton switch was used to toggle the power on and off. If you were to press the switch rapidly at a constant rate, the resistor would receive 1.7 watts during the ON time and 0 watts during the OFF time. This 50% duty cycle would result in an average power consumption of .85 watts ($P_{average} = P_{full} * \text{duty cycle}$). Complete the table in Figure 4.9 below for power consumed at duty cycles of 75% and 25% for your system.

Figure 4.9: Average Power

$P_{average} = P_{full} * \text{duty cycle}$		
Full Power (P_{full})	Duty cycle	Average Power (P_{avg})
1.7 W	100%	1.7 W
1.7 W	75%	
1.7 W	50%	0.85 W
1.7 W	25%	
1.7 W	0%	0 W

PBASIC provides a useful instruction for providing pulse-width modulation.
Its syntax is: **`PWM pin, duty, duration`**

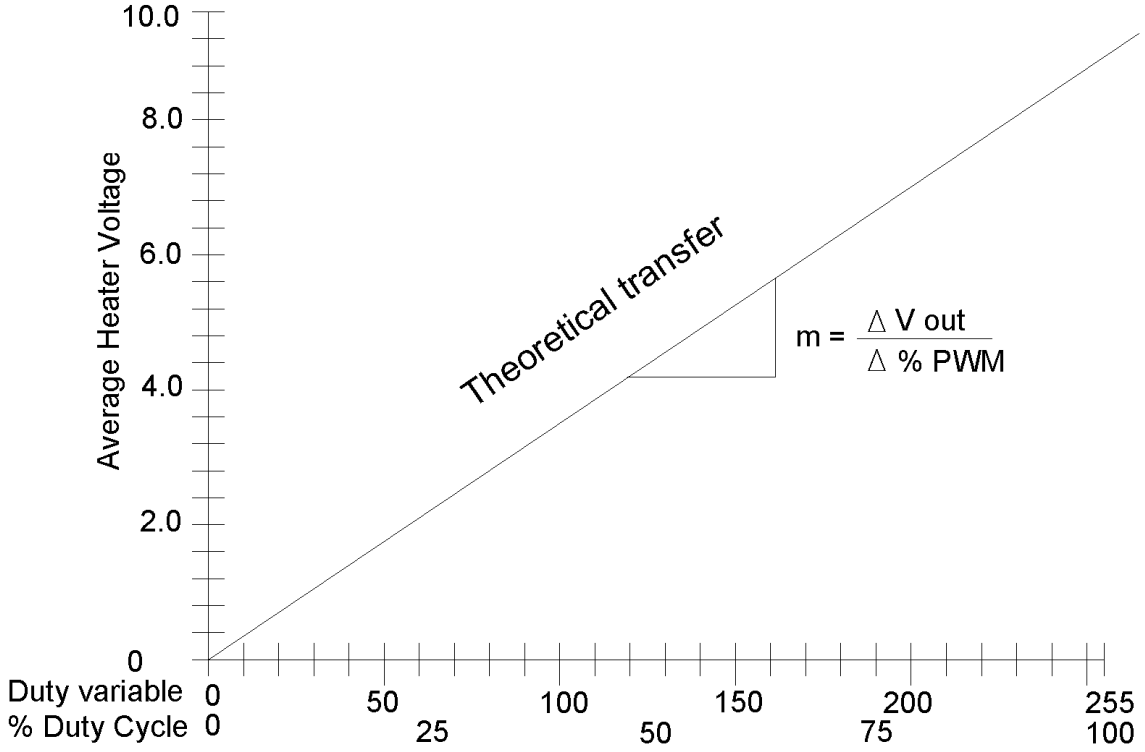
Where: **`pin`** is the output pin you are driving.
`duty` is the duty cycle relative to 255 being 100%.
`duration` is the window of time in milliseconds over which the duty cycle is provided.

Challenge! Graphing PWM duty vs. Vout

Use your multi-meter to measure the average voltage across the heating element at various PWM commands. Change the duty variable in Program 4.2 to increments between 0 and 255. Plot the average voltage on the graph in Figure 4.10.

```
'Program 4.2:  PWM vs. Vout

' Change the Duty = 50 in increments of 10 between 0 and 100.  Measure the
' average output voltage that results.

DutyCycle    var    byte
Duty         var    byte

DutyCycle = 50                          ' Begin with a 50% duty cycle

Loop:
  Duty = (DutyCycle * 255/100)          'Scale DutyCycle to PWM (0-255) duty
  PWM 8, Duty, 200                      ' Apply a Duty Cycle of to the heater
  DEBUG " Testing at a Duty Cycle of  ",  DEC DutyCycle, "%.", CR
GOTO Loop
```

Figure 4.10: Graph of Heater Voltage vs. PWM Duty Cycle



If you are not familiar with PBASIC's PWM instruction, refer to the BASIC Stamp Manual Version 1.9 pp. 293-295. One aspect of using the command should be understood. PWM applies pulses for a period of time defined by the duration value. During the time when the rest of the program is executing, there is no output applied to the load. As a result, the average voltage at a 100% duty cycle (duty =255) will result in a value less than the full voltage expected. The slower your program cycle time, the greater is this disparity. To get a better understanding of cycle time, place a **PAUSE 200** in Program 4.2. Compare the resulting output voltage with earlier readings. Change the length of the pause and notice the results.

Challenge! Analyzing your Open-Loop System

The following program is developed to study the relationship between PWM drive on your heater and the resulting stable temperature. The program will apply PWM drive levels in 10% increments. Each increment will last approximately four minutes. The program will end after 100% drive has been applied. StampPlot Lite will give you a graphical representation of your system's response, along with time stamp information in the list box. Furthermore, if you are really interested, the StampPlot Lite data file can be imported into a spreadsheet, applied to a graph and analyzed.

Figure 4.11: Screen Shot of PWM Drive vs. Temperature

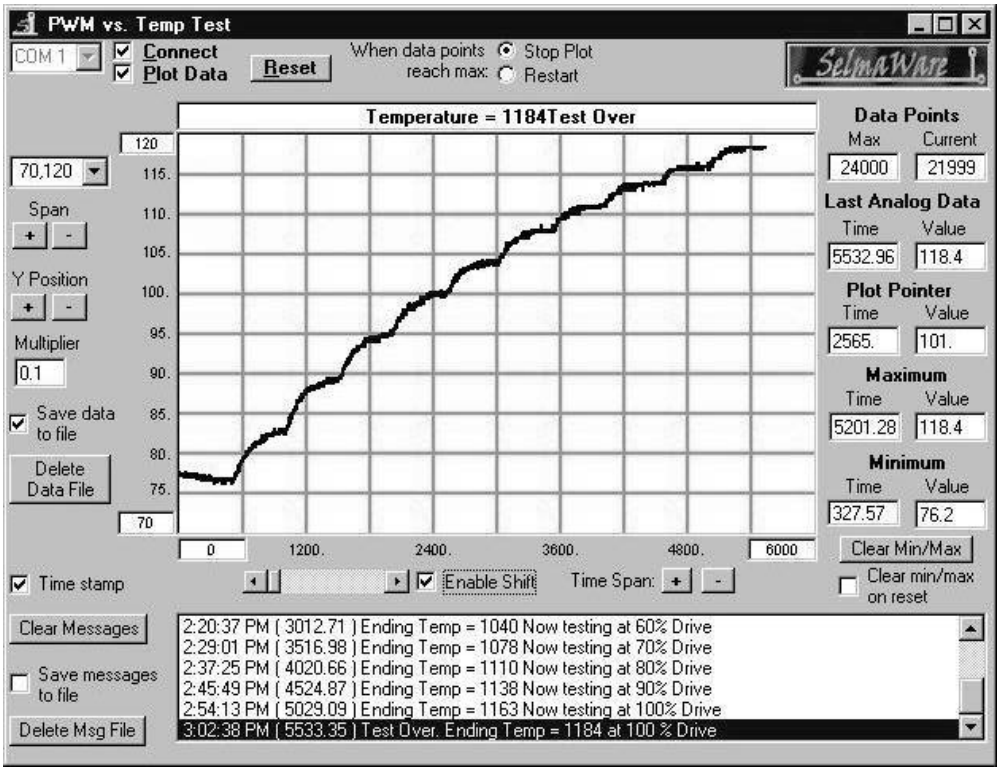


Figure 4.11 is typical of a StampPlot Lite screen shot resulting from this test. Load Program 4.3. Before running the program, be sure your canister has cooled to room temperature. Place the cap on your canister and start the program. When the DEBUG window appears, close it and start StampPlot Lite. Connect using StampPlot Lite and press the restart button to reload the program and begin the test.


```
'Program 4.3: PWM vs. Temp Test with StampPlot Interface

'This program tests the canister's temperature rise for incremental increases of
'PWM drive. Program runtime is approximately 40 minutes.  This can be adjusted
'by 'changing the "tick" and/or "Drive" increments.
'Program assumes that the circuitry is set according to Figure 4.3.
'ADC0831: '"chip select" CS = P3, "clock" 'Clk=P4, & serial data output"Dout=P5.
'Zero and Span pins: Digital 0 = Vin(-) = .70V and Span = Vref = .50V.

'Configure Plot
Pause 500                          'Allow buffer to clear
DEBUG "!RSET",CR                   'Reset plot to clear data
DEBUG "!TITL PWM vs. Temp Test",CR 'Caption form
DEBUG "!PNTS 24000",CR             '2000 sample data points
DEBUG "!TMAX 6000",CR              'Max 300 seconds
DEBUG "!SPAN 70,120",CR            '50-300 degrees
DEBUG "!AMUL .1",CR                'Multiply data by .1
DEBUG "!DELD",CR                   'Delete Data File
DEBUG "!SAVD ON",CR                'Save Data
DEBUG "!TSMP ON",CR                'Time Stamp On
DEBUG "!CLMM",CR                   'Clear Min/Max
DEBUG "!CLRM",CR                   'Clear Messages
DEBUG "!PLOT ON",CR                'Start Plotting
DEBUG "!RSET",CR                   'Reset plot to time 0

' Define constants & variables

CS con 3                           ' 0831 chip select active low from BS2 (P3)
CLK con 4                          ' Clock pulse from BS2 (P4) to 0831
Dout con 5                         ' Serial data output from 0831 to BS2 (P5)
Datain var byte                    ' Variable to hold incoming number (0 to 255)
Temp var word                      ' Hold the converted value representing temp

TempSpan var word                  ' Full Scale input span in tenths of degrees.
TempSpan = 5000                    ' Declare span.  Set Vref to .50V and
                                   ' 0-255 res. will be spread over 50
                                   '(hundredths).

Offset var word                    ' Minimum temp.  @Offset, ADC = 0
Offset = 700                       ' Declare zero Temp.  Set Vin(-) to .7 and
```

```
                                ' Offset will be 700 tenths degrees.  At these
                                ' settings, ADC output will be 0 - 255 for temps
                                ' of 700 to 1200 tenths of degrees.

LOW 8                           ' Initialize heater OFF

Drive var word                  ' % Drive
duty var word                   ' variable for PWM duty cycle
tick var word



Drive = 0                       ' Initialize variable to 0
tick = 0
duty = 0

' Get and display initial starting values.

  GOSUB Getdata
  GOSUB Calc_Temp
  DEBUG "Temp = ", DEC Temp, " Duty = ", DEC duty,CR
  DEBUG "!USRS Begining Test! -- Testing at ", DEC Drive, "% Drive.",CR

Main:                           ' main loop
  PAUSE 10
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOSUB Display
  GOTO Main

Getdata:                        'Acquire conversion from 0831
  LOW CS                        'Select the chip
  LOW CLK                       'Ready the clock line.
  PULSOUT CLK,10                'Send a 10 uS clock pulse to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8]       'Shift in data
  High CS                       'Stop conversion
RETURN

Calc_Temp:                      'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset 'temp based on Span &
RETURN                          'Offset variables.

Display:                        'Plot present temperature
  DEBUG DEC Temp,CR
RETURN
```

```
Control:                          ' Testing system at different % duty cycles
  PWM 8,duty,200                  ' PWM
  tick = tick + 1                 ' increment tick variable
  IF tick = 2000 Then Increase    ' Program cycles per drive level change
RETURN

Increase:                         'Bump up the drive
  Drive = Drive + 10              'Drive increments = 10%
  duty = (Drive * 255/100)        'Scale %Drive to Duty
  If duty > 256 Then Stopit       'Stop test after 100% PWM
  DEBUG "Ending Temp = ", DEC Temp, " Now testing at ", DEC Drive, "% Drive", CR
  DEBUG "!USRS Testing at ", DEC Drive, "% Drive",CR
  tick = 0
RETURN

Stopit:                           ' Stop and print summary
  DEBUG "Test Over. Ending Temp = ", DEC Temp," at 100 % Drive",CR
  DEBUG "!USRS Temperature = ", DEC Temp,"Test Over",CR
END
```

Challenge! Open-Loop Control -- Desired Setpoint = 101° Fahrenheit

It is our objective to maintain a constant canister temperature of 101° Fahrenheit. Follow these procedures. Record values in the table of Figure 4.12.

1. Study the StampPlot Lite analysis that resulted from running Program 4.2. From the Text Box listing, record in the table the beginning ambient temperature, the temperature at the end of the 50% drive test, and the ending maximum temperature after 100% drive.

2. Use your cursor to find the Drive level that resulted in a temperature of 101 degrees.

3. Next, modify the **`Control`** subroutine of Program 4.2 so that the duty cycle remains at the constant value declared initially. Do this by removing the two lines indicated below.

```
Control:                                ' Testing system at different % duty cycles
       PWM 8,duty,200                   ' PWM
       tick = tick + 1                  ' increment tick variable
       IF tick = 2000 Then Increase     ' Program cycles per drive level change
       RETURN
```

4. At the beginning of the program, declare the **`DutyCycle`** to be the value that yielded 101° in our test StampPlot Lite. Run the program and allow the system to stabilize. How close was your estimation? Bump it up or down accordingly to find the setting that yields the desired result. In line #5 of the table, record the percent of drive that places the system at or near 101 degrees. Let it run for a moment and take note of the system stability. Once a drive setting has been established, an open-loop system will stabilize; and, as long as the disturbances that affect the process stay constant, so will the output.

5. Plug in the brushless fan across the Vdd supply and aim it directly toward the canister. The moving air represents a change in the disturbance on your process. According to theory, heat will be removed from the process at a greater rate and the new stable temperature will be lower than 101°. With the fan blowing on the canister, try to find the new "correct" drive for this condition. Record your data in line #6 of the table.

Figure 4.12: Open-Loop Control Table

Line#	Condition	% Drive	Temp
#1	Desired Temperature		101°
#2	Ambient Temperature	0	
#3	50 % Drive Temperature	50%	
#4	Full Drive Temperature	100%	
#5	Appropriate % Drive for 101° Without fan disturbance		101°
#6	Appropriate % Drive for 101° With direct fan disturbance		101°
#7	Appropriate % Drive for 101° With partial fan disturbance		101°

6. Finally, leaving the proper setting established in line #6, change the position of the fan so it is blowing less directly on the canister. This represents a medium disturbance level on the system. Assess the situation and make your best guess as to the proper drive setting required by this new condition. Program the BASIC Stamp for this drive level. Once the system stabilizes, record your results in line #7 of the table.

Challenge!

1. Select a new "desired temperature" for your system. Predict and program an open-loop drive value that will maintain this temperature.

2. Place a couple of glass marbles in your canister. See how increasing the mass of the system affects the response and the drive setting necessary to maintain the new condition. What conclusions can you draw from the system's behavior?

There are many variables that can affect the relationship of drive level and temperature in your small environment. Given some time to experiment and become familiar with the dynamic relationship between temperature, drive level, and disturbances, you could get pretty good at assessing the conditions and setting the right amount of drive in an open-loop manner. As we see, however, if any condition of our process changes, so will the output. Open-loop control can be useful in some applications. When a process requires that its output remain constant for all conditions, then closed-loop control must be employed. In closed-loop control, action is taken based on an evaluation of the measurement and the desired setpoint. This evaluation results in what is called an "error signal."

Questions and Challenge

1. Give two examples of continuous process control other than those given in the text.

2. How is the drive level in open-loop control determined?

3. What is the primary advantage of open-loop control?

4. What is the primary disadvantage of open-loop control?

5. The ADC0831 will convert a range of analog input to one of 256 possible binary values. The number 256 identifies the _____ of the converter.

6. The purpose of the chip select and clock lines to the ADC0831 are to _____ the conversion process.

7. If the LM34 were placed in a 98.6-degree environment, the expected output would be _____volts.

8. In pulse-width modulation, the amount of drive action is based on the _____ of time ON over the total time.

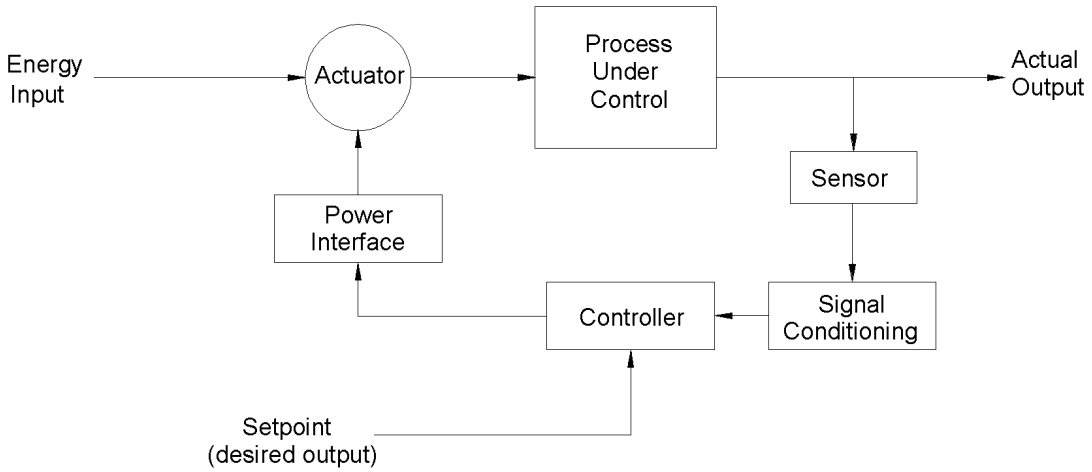
9. If a 40-watt heater were pulse-width modulated at a 75% duty cycle, the average power consumed would be _____ watts.

10. When disturbances change in an open-loop process, so does the _____.

Experiment #5: Closed-Loop Control

An open-loop control system can deliver a desired output if the process is well understood and all conditions affecting the process are constant. However, Experiment #4 showed us that an open-loop control system couldn't guarantee the desired output from a process that was subject to even mild disturbances. There is no mechanism in an open-loop system to react when disturbances affect the output. Although you were able to find a drive setting that would yield the desired temperature in Experiment #4, when the fan was moved closer or further from the heater, the fixed setting was no longer valid. Closed-loop control provides automatic adjustment of a process by collecting and evaluating data and responding to it accordingly. A typical block diagram of an automatic control system is depicted in Figure 5.1.

Figure 5.1: Closed-Loop Control



In this diagram, an appropriate sensor is measuring the Actual Output. The signal-conditioning block takes the raw output of the sensor and converts it into data for the Controller block. The Setpoint is an input to the Controller block that represents the desired output of the process. The controller evaluates the two pieces of data. Based on this evaluation, the controller initiates action on the Power Interface. This block provides the signal conditioning at the controller's output. Experiment #3 discussed several methods of driving power interface circuits. The Power Interface has the ability to control the Actuator. This may be a relay, a solenoid valve, a motor drive, etc. The action taken by the Actuator is appropriate to drive the Actual Output toward the desired value.

As you can see, this control scenario forms a loop, a closed-loop. Furthermore, since it is the process's output that is being measured, and its value determines actuator settings, it is a feedback closed-loop system. The input changes the process output → the output is monitored for evaluation → the evaluation changes the input → that changes the process output, etc., etc.

The type of reaction that takes place upon evaluation of the input defines the process-control mode. There are five common control modes. They are on-off, on-off with differential gap, proportional, integral, and derivative. The fundamental characteristic that distinguishes each control mode is listed below in Table 5.1.

Table 5.1: Five Common Control Modes

Process Control Mode	Evaluation	Action
On-off	Is the variable above or below a specific desired value?	Drive the output fully ON or fully OFF.
On-off with differential gap	Is the variable above or below a range defined by an upper and lower limit?	Output is turned fully ON and fully OFF to drive the measured value through a range.
Proportional	How far is the measured variable away from the desired value?	Take a degree of action relative to the magnitude of the error.
Integral	Does the error still persist?	Continue taking more forceful action for the duration the error exists.
Derivative	How fast is the error occurring?	Take action based on the rate at which the error is occurring.

This exercise will focus on converting the open-loop temperature control system of Experiment #4 into an on-off closed-loop system. Our system will show advantages and disadvantages to this method of control. The characteristics of the system being controlled determines how suitable a particular control mode will be. Experiment #6 will use the same circuitry to overview and apply proportional, integral, and derivative control modes. Leave the circuit constructed after completing this step.

Figure 5.2 is a schematic of the circuitry necessary for the next two exercises. As you see, this is identical to Experiment #4. The 35-mm film canister provides the environment we wish to control. The heater drive provides full power for developing heat in the resistor. The LED is also driven by Pin 8. Remember that the LED is driven by the +5-Vdd supply, and the heater is driven by the +9-volt unregulated line supply. The LM34 sensor will provide temperature data. In closed-loop control, we will monitor the temperature and use it to determine control levels. The fan's air currents will act as a disturbance to the process.

Figure 5.2: Closed-Loop Control Circuitry

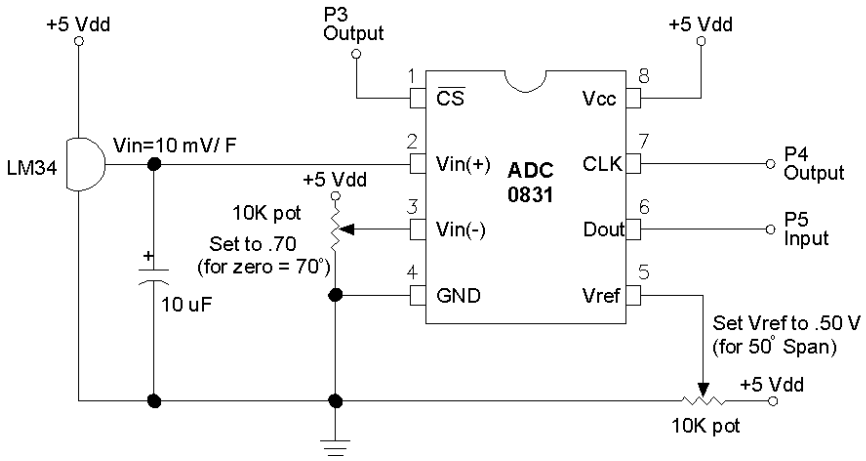


Figure 5.2a: LM34 to ADC0831 Serial A-to-D

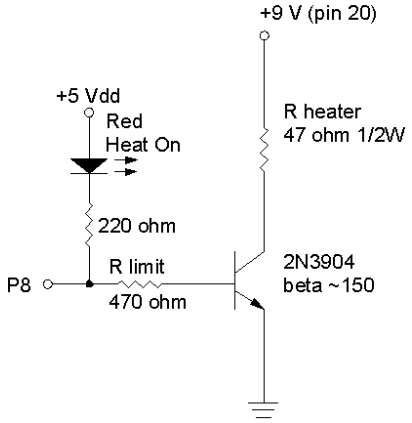


Figure 5.2b: BASIC Stamp Driving the Heater with LED Indicator

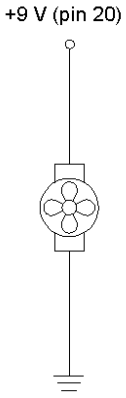


Figure 5.2c: Brushless Fan - Directly Connected

If the circuit isn't already on your board, carefully construct it. Use space on the small Board of Education efficiently to allow for the circuitry. Take your time, plan your layout, and be careful not to inadvertently short any wires. Refer back to Experiment #4 for details on the film canister construction, the operation of the LM34, and the use of the ADC0831 analog-to-digital converter.

Double-check the Zero and Span voltages of the ADC0831. Use your voltmeter to set the Zero voltage ($Vin_{(-)}$) to .7 and the Span ($V_{(ref)}$) to .5 volts. This will establish a full-scale temperature measurement range from 70 to 120 degrees F.

Exercises

Exercise #1: Establishing Closed-Loop Control

Let's assume it is our objective to maintain temperature within the canister at 101.50 ^{o}F $\pm$ 1 degree. This would be representative of the requirements of an incubator used for hatching eggs. Maintaining the eggs at the setpoint temperature of 101.5 ^{o}F is perfect, but the temperature could go up to 102.50 or down to 100.50 without damage to the embryos. Although it may be hard to imagine an incubator when you look at your film canister, the BASIC Stamp would be well suited as the controller in a large commercial hatchery incubator.

To maintain temperature at the desired value seems like a pretty "common sense" task. That is, simply measure temperature; if it is above the setpoint, turn the heater OFF; and, if it is below, turn the heater ON. The simplest kind of control mode is on-off control. There are drawbacks to this control mode, however. During the following exercise, you will establish on-off control of your model incubator. Pay close attention to the characteristics exhibited by your model. These characteristics would also apply to real control applications.

Procedure

Programming for this application requires data acquisition, evaluation, and control action. Our display routine will also include storing and displaying the minimum and maximum overshoot in the process.

The structure and much of the content of Program 4.1 may be used to acquire and calculate our measurement. Instead of turning the heater on continually, a new subroutine will be added to evaluate and control it. Evaluation will be based on a **`setpoint`** variable. Refer to Program 5.1 following.

```
'Program 5.1: Simple ON/OFF Control with the StampPlot Interface

'This program establishes simple ON/OFF control of the model incubator.
'Program I/O is based on the circuitry of Figure 5.2.
'Zero and Span voltages: Digital 0 = Vin(-) =.70V and Span = Vref = .50V.

'Configure Plot
Pause 500                         'Allow buffer to clear
DEBUG "!RSET",CR                  'Reset plot to clear data
DEBUG "!TITL Simple ON/OFF Control",CR         'Caption form
DEBUG "!PNTS 60000",CR            '2000 sample data points
DEBUG "!TMAX 300",CR              'Max 300 seconds
DEBUG "!SPAN 70,120",CR           '70-120 degrees
DEBUG "!AMUL .1",CR               'Multiply data by .1
DEBUG "!DELD",CR                  'Delete Data File
DEBUG "!CLMM",CR                  'Clear Min/Max
DEBUG "!CLRM",CR                  'Clear Messages
DEBUG "!USRS  ",CR                'Clear User status bar
DEBUG "!SAVD ON",CR               'Save Data
DEBUG "!TSMP ON",CR               'Time Stamp On
DEBUG "!SHFT ON",CR               'Enable plot shift
DEBUG "!PLOT ON",CR               'Start Plotting
DEBUG "!RSET",CR                  'Reset plot to time 0

' Define constants & variables

CS con 3                          ' 0831 chip select active low from BS2 (P3)
CLK con 4                         ' Clock pulse from BS2 (P4) to 0831
Dout con 5                        ' Serial data output from 0831 to BS2 (P5)
Datain var byte                   ' Variable to hold incoming number (0 to 255)
Temp var word                     ' Hold the converted value representing temp

TempSpan var word                 ' Full Scale input span in tenths of degrees
TempSpan = 5000                   ' Declare span 50 (1/100ths degrees)

Offset var word                   ' Minimum temp. Offset, ADC = 0
Offset = 700  ' Declare zero Temp. Set Vin(-) to .7 and
                                  'Offset will be 700 tenths degrees.  At these
                                  'settings, ADC output will be 0 - 255 for temps
                                  'of 700 to 1200 tenths of degrees.
Setpoint var word
Setpoint = 1015                   ' Initialize setpoint to 101.5 degrees

MMFlag var bit
MMFlag = 0
```

```
LOW 8                                   ' Initialize heater OFF

Main:
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOSUB Display
GOTO Main

Getdata:                                'Acquire conversion from 0831
  LOW CS                                'Select the chip
  LOW CLK                               'Ready the clock line.
  PULSOUT CLK,10                        'Send a 10 uS clock pulse to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8] 'Shift in data
  High CS                               'Stop conversion
RETURN

Calc_Temp:                              'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset     'temp based on Span &
RETURN                                  'Offset variables.

Control:                                'ON/OFF control
  IF Temp > Setpoint THEN OFF
  High 8                                'Heater ON
RETURN

OFF:                                    'Heater OFF
  LOW 8
RETURN

Display:                                'Plot Temp and heater status
  IF OUT8 = 0 AND MMFlag = 0 THEN MMClear ' Clear Min/Max
  DEBUG DEC Temp,CR
  DEBUG IBIN OUT8,CR
RETURN

MMClear:                          'Clear Min/Max When setpoint is first reached.
  DEBUG "!CLMM",CR
  DEBUG "!USRS Overshoot/Undershoot Test Ready",CR
  MMFlag = 1
RETURN
```

Run the program and observe the behavior of the system. StampPlot Lite will graphically plot the temperature response of the system and the on-off status of Output 8. Follow the StampPlot Lite procedures of running the program, closing the debug window, opening StampPlot Lite, and pressing the "reset" button.

When you start your system, the heater will be on as indicated by the LED. The heater/resistor becomes quite hot when full power is applied. This heat transfers through the environment and warms the temperature sensor. When the sensor has heated to 101.5, the BASIC Stamp will turn off the heater. For a period after the heater is turned off, the temperature continues to rise. This is called overshoot. At this point, it is important to understand the dynamics of your system. The heat held within the mass of the resistor will continue to dissipate into the air, the air becomes warmer, and the LM34 reports that overshoot has occurred. Similar to the mechanical inertia of a moving object, this phenomenon is called thermal inertia. Overshoot becomes large when the heat energy contained in the mass of the resistor is large, relative to the heat already in the canister. The 35-mm canister is small, but the mass of the half-watt resistor also is small. As a result, the overshoot of your system will probably be less than one degree.

When the temperature does turn around and begin to fall, as it passes the setpoint, the heater is once again turned on. Undershoot will occur for similar reasons as did the overshoot. During the time the heater is coming up in temperature, the ambient temperature has continued downward. Continuous cycling above and below the desired setpoint is typical of on-off control. The rate of this cycling and the degree of the overshoot depend on the characteristics of the system. On-off control is suitable for processes that have large capacity, can tolerate sluggish response, and sustain a relatively constant level of disturbance. If our incubator were large, well insulated, and kept in a constant room environment, on-off control would be acceptable. After the process has had a chance to cycle a few times, record the minimum and maximum overshoot values.

Maximum overshoot _____ Minimum Overshoot _____

Using your cursor, investigate time between cycles. Record these times below.

Time at which the heater first turned OFF (T_{1off}): _____

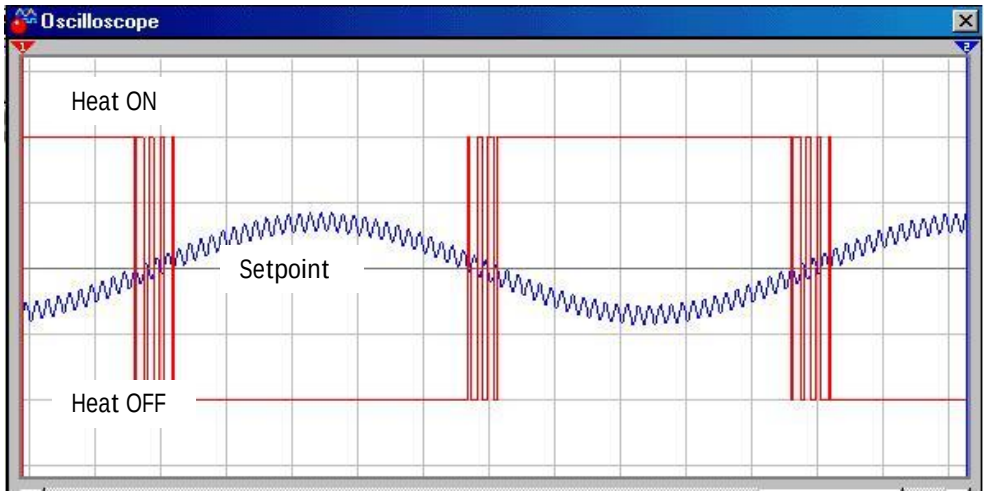
Time at which the heater turned back ON (T_{1on}): _____

Time at which the heater turned OFF again (T_{2off}): _____

Cycle time = (T_{2off}) – (T_{1off}): _____

A major problem with on-off control is that the output drive may cycle rapidly as the measurement hovers about the setpoint. Noise riding on the analog sensor measurement would be interpreted as rapid fluctuation above and below the setpoint. The timing diagram in Figure 5.3 represents this problem

Figure 5.3: On-Off Control When Noise Rides on the Data



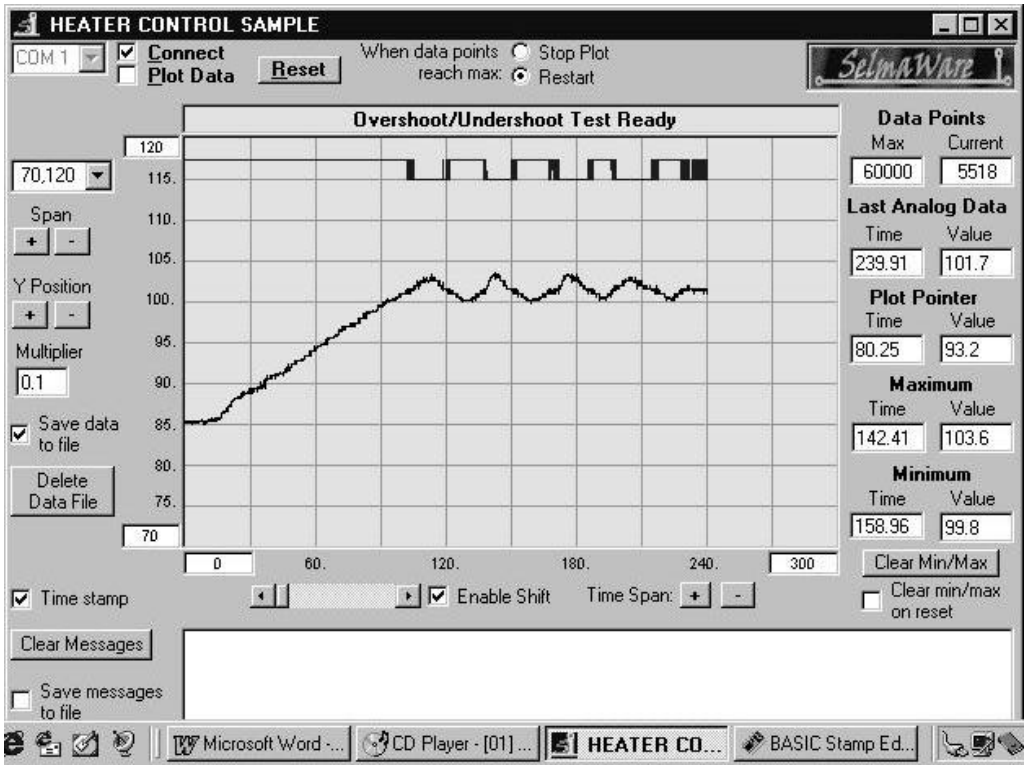
The slow-moving data that is cycling through the setpoint has a high-frequency noise component riding on it. As you can see, the coupled effects of the noise results in the data passing above and below the setpoint several times. The microcontroller would attempt to turn the heating element on and off accordingly.

In an actual incubator application where larger amounts of power are controlled, this rapid switching could cause unwanted RF noise. This rapid cycling could also be damaging to electromechanical output elements such as motors, relays, and solenoids.

Do you observe the rapid cycling of the LED as the temperature approaches the setpoint? _____

Remove the 22-uF capacitor from across the sensor output. Does this increase the cycling problem? Why? Figure 5.4 is a typical screen shot resulting from this experiment. Overshoot and rapid cycling are a problem. Is this similar to your system's response?

Figure 5.4: Simple ON/OFF Control



Challenge! Change the Dynamics of Your System

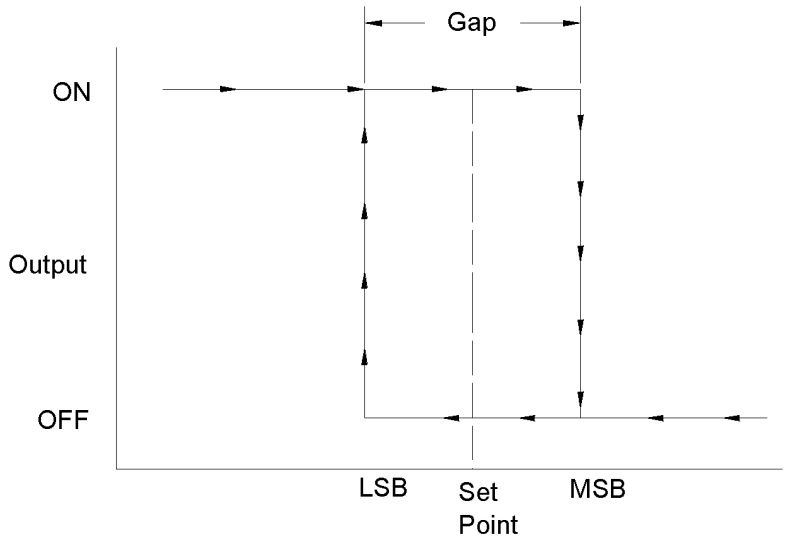
1. Connect your brushless fan to the Vcc supply and aim it toward the canister. Reset the program and watch the control action. Describe any effect that the new level of disturbance has on the overshoot and/or the cycling. Summarize how you have changed the dynamics of your system.

2. Place a single glass marble in your canister. Reset the program and watch the control action. Describe any effect that changing the mass of your system has on the overshoot and/or the cycling. Summarize how you have changed the dynamics of your system.

Exercise #2: Differential-Gap Control

The rapid cycling resulting from noise or the measurement hovering around a single setpoint is the biggest disadvantage of simple on-off control. Most practical on-off control systems lend themselves to allowing a minimum and maximum value of measurement. The incubator system is a good example. Although our desired temperature is 101.5 degrees, it allows $\pm$ 1 degree of variance about the setpoint. Differential-gap control is a mode of control that takes action based on the measurement crossing a defined upper and lower limit. When the measured value goes beyond one limit, full appropriate action is taken to drive the temperature to the opposite limit. Full opposite action is then taken to drive the process back again. Figure 5.5 graphically diagrams the action taken by differential-gap control. When the system is started and its temperature is below the Lower limit, the heater will come on and the temperature rise. When the temperature passes the upper limit, the heater is turned OFF, heat will begin to leave the process, and the temperature will begin to drop to below the Lower limit. The heat then is turned back on and the cycle begins again.

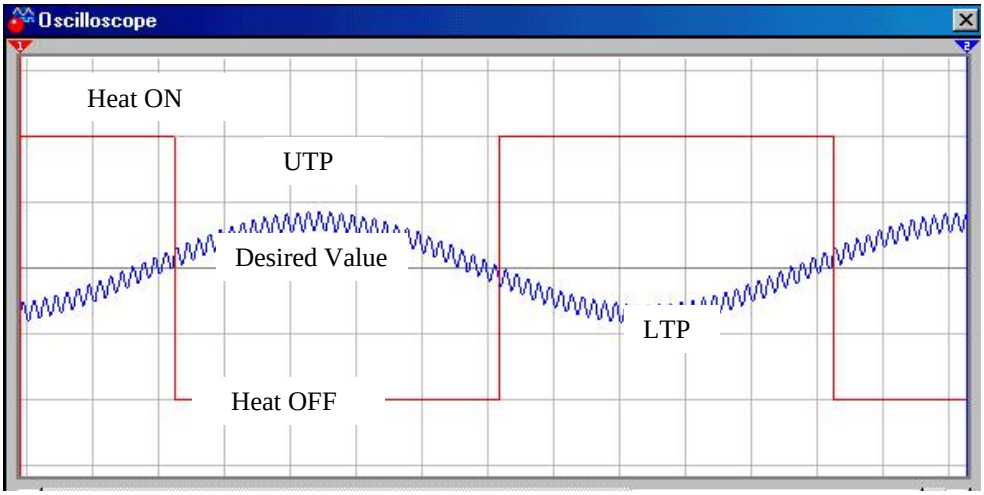
Figure 5.5: Differential-Gap Control Action



The result is a slower on-off cycle time and no cycling resulting from noisy data. Notice how the timing diagram in Figure 5.6 differs from the earlier one depicting noisy data in a simple on-off control mode.

Whenever the measurement is anywhere between these limits, the heater state is not changed. The result is a slower ON/OFF cycle time and no cycling from noisy data. The heater is switched when the data (+noise) passes a limit. After that, the data (+noise) has to exceed the other limit before switching will occur again. Because the differential gap is wider than the effect of the noise, rapid cycling is eliminated.

Figure 5.6: Differential-Gap Control Action when Noise is Riding on the Data



These advantages come at the compromise of allowing the measured variable to drift further from the desired "average" value. The thermal inertia of our system will still result in some amount of overshoot and undershoot. We are accepting a wider variance in temperature. When processes allow this variance, differential-gap control is usually preferred over simple on-off control.

The **`Control`** subroutine can be easily changed to accommodate differential-gap control. Make the following modifications to Program 5.1.

Declare the new variables of **`Upper_limit`** and **`Lower_limit`** at the beginning of the program and initialize them to 102 ^{o}F and 101 ^{o}F respectively.

```
Upper_limit var word
Lower_limit var word
Upper_limit = 1020                              ' 102 degrees in 1/10ths
Lower_limit = 1010                              ' 101 degrees in 1/10ths
```

Next, replace the on-off **`Control`** subroutine with the following code.

```
Control:
  IFTemp > Upper_limit THEN OFF   ' Over upper limit then Heat OFF
  IF Temp < Lower_limit THEN ON   ' Under lower limit then Heat ON
RETURN                            ' return leaving heat in last state

OFF:
  Out8 = 0                        ' Heater Off
  maxflag = 1
RETURN

ON:                               ' Heater On
  Out8 = 1
RETURN
```


Run the program and observe the behavior of your system.

Challenge! Observe and Evaluate Differential-Gap Control

Allow the program to cycle a few times. Report on the following:

Record the minimum and maximum overshoot temperatures.

Maximum overshoot _____ Minimum Overshoot _____

With your cursor, investigate time between cycles. Record these times below.

Time at which the heater first turned OFF (T_{1off}) _____

Time at which the heater turned back ON (T_{1on}) _____

Time at which the heater turned OFF again (T_{2off}) _____

Cycle time = (T_{2off}) – (T_{1off}) _____

Project the switching point of the digital output down to the plotted temperature. Can you determine the temperature at which switching occurred? Momentarily remove the 22-uF filter capacitor. Does the increase in noise cause rapid cycling about the limits?

Use the fan to change the disturbances to the process. Reset the program and watch the control action. Describe any effect the new level of disturbance has on the overshoot and/or the cycling. Summarize how you have changed the dynamics of your system.

Place a single glass marble in the canister. Restart the program and summarize how increasing the mass has affected the process control action. Investigate the cycle time and overshoot.

Simple ON/OFF or Differential Gap?

Hopefully, the data that you have observed and recorded will reveal some important characteristics of these two control modes. They both have advantages and disadvantages. Simple on-off control results in rapid cycling of the heating element. Reported cycle times of less than one second could easily result if your system has fast recovery or there is noise on the analog line. Rapid cycle time would not be acceptable if our heater were being controlled by an electromechanical relay.

Notice, however, that the overshoot is approximately a half-degree and our average temperature is at the desired setpoint. Compare this control response to that observed when Differential Gap has been added to the On/OFF control.

With Differential-Gap control, you will notice fundamental differences in the control action.

1. Rapid cycling about the setpoint no longer occurs.
2. The minimum and maximum values still overshoot, but now beyond the limits.
3. Total cycle time between ON/OFF conditions is longer.

Increased cycle time and noise immunity about the setpoint are definite improvements over simple on-off control. The tradeoff, however, is allowing the process to vary further from the desired temperature setpoint. Obviously, an understanding of your process and its hardware will determine the appropriate control mode.

Both modes took appropriate control action to maintain temperature under changing disturbance levels and load conditions. The drawback to either mode of ON/OFF control is that the controlled variable is constantly on the move. The fully-ON and fully-OFF conditions of the final control element are continually forcing the measurement past the limits.

If you recall, in the Open-Loop Control exercise of Experiment #3, a value of drive between off and fully-on was found to be appropriate to hold the temperature at the setpoint. If all disturbances to the process remained constant, the temperature would stay at the setpoint when the right percentage of drive was applied. We also saw that as conditions changed, so did the measurement. Experiment #6 will investigate controls that take an appropriate amount of action based on an evaluation of the measurement. Applying Proportional, Integral, and Derivative control theory can be employed to maximize the effectiveness of the control system.

Programming Challenges

1. Alter your program so a ±1 degree differential gap can be calculated automatically, based on the desired setpoint.

2. Add a variable called Differential_gap so the operator needs only to enter the Setpoint and the amount of Differential gap and the program automatically performs the desired action.

Questions

1. Write one complete sentence that clearly states the fundamental difference between Open-loop and Closed-loop controls.

2. Simple ON/OFF control compares the measurement of the process variable to a _____.

3. If the final control element to our model incubator were an air conditioner instead of a heater, what would change about Program 5.1?

4. When the process variable continues to increase after the final control element is turned OFF, it is termed _____.

5. Rapid cycling about the setpoint of an ON/OFF control system is a result of _____ riding on the measurement data.

6. List three devices that could not withstand rapid cycling of power.

   _____

   _____

   _____

7. Process cycle time is directly affected by the amount of _____ in the system.

8. Differential-gap control takes full action when the process variable crosses the _____ points.

9. Cycle time will be _____ if differential-gap control is used in a system rather than simple ON/OFF control.

10. Rapid cycling is not a problem in differential-gap control if the measurement data plus the _____ is less than the differential gap.

11. When the measurement is between the limits, the output will be in the mode determined by which limit was exceeded _____.

12. Adding more mass to a Differential-gap control system will _____ the amount of overshoot and _____ the cycle time of the process.

Experiment #6: Proportional-Integral-Derivative Control

PID is an acronym for Proportional-Integral-Derivative Control. In this section, we will explore each of these methods and how they work together to efficiently control a system.

Proportional Control

In the previous exercise, we used various means of cycling the heater of our incubator to maintain a desired temperature. Using differential gap, we created an allowable band in which the heater would cycle, causing the temperature to cycle above and below the setpoint. In Experiment #4, we saw how we could use pulse-width modulation (PWM) to add energy to our system in duty cycles between 0% (fully off) and 100% (fully on). In proportional control, we will combine a defined band of temperature and closed-loop feedback to adjust the PWM to our heater, maintaining a desired setpoint.

The first important step to understanding proportional control is to acknowledge that every system has both energy gains and energy losses. In our incubator, we add energy to the system by heating our resistor. The energy lost from our system is due to ambient losses. Heat changes in our container due to the temperature of the surrounding air. Consider another system, such as a fluid-flow system. Suppose we want a specific flow of oil at 10 gallons per minute. Energy is added to the system by a pump used to push the oil. A major loss in the system is from the restriction of the pipes to the oil flow. Friction between the oil and the pipe walls removes flow energy. In our automobiles, the engine adds energy to propel our car. The friction of the tires on the pavement, gravity and wind flow around the car all affect energy levels.

When the energy added to the system equals the energy lost, the system is balanced. In the incubator, if a 50% PWM added sufficient energy to make up for all the losses in the system, it would be balanced and the temperature would remain constant. If our losses exceeded the energy added to the system, the temperature would decrease. How far would it decrease? Until it found a new equilibrium of loss equaling gain. As our system cools less, there is less of a temperature difference between the inside and outside of the canister. The losses will decrease until they meet and equal energy added. Conversely, if gains are greater than losses, the temperature will increase until a new equilibrium is found at a higher temperature.

Think about when you are driving. You press the accelerator to add sufficient energy to propel the car to 65 MPH. On a flat, straight road, there may not be any need to change how far the accelerator is pressed to maintain speed. What happens when you start up a hill? Since your car is inclined, there is greater energy lost due to gravity pulling it. Without a change in how far the accelerator is pressed, the car begins to slow. Does it eventually come to a full stop? No, because the amount of energy from the engine reaches a point where it equals the new amount of losses, allowing a slower but stable speed. The system is back in equilibrium.

Let's think about our incubator operating in equilibrium. If using a PWM duty cycle (drive) of 50% adds exactly the amount of heat being lost by the system, the temperature will remain constant. What temperature will that be? That is dependent on the losses. Let's say the room temperature is 90 F and a PWM of 50% drive stabilizes the temperature at 101 F. If room temperature drops to 80 F, with the drive still at 50%, temperature might stabilize at 95 F.

But we want the temperature to stay at 101 F. How do we get the temperature to return to this value? We have two options: reduce the losses of the system, or increase the energy added to the system. Trying to adjust losses isn't a satisfactory solution to our problem. It could mean altering conditions that might be difficult or impossible to change. In our driving example, removing the hill from the road so we wouldn't slow down might sound like a nice idea, but isn't feasible. What do we do instead? We add energy (fuel) to the system via the accelerator.

In proportional control, feedback is used to monitor the system. For average losses, the amount of energy needed to maintain the setpoint is known as the "bias." The bias energy will equal anticipated losses, providing a steady-state system at the setpoint. As our losses change, however, the system is driven above or below the biased setpoint to compensate. The difference between the setpoint and the actual system drive is called the "amount of error." The amount of energy added to or removed from the system is proportional to the amount of error.

In our automobiles, we can use a cruise control to automatically adjust the fuel to our engine to maintain a constant speed, such as 65 MPH. At equilibrium on a flat, straight road as we rest our foot on the cruise-controlled accelerator, its position does not change while maintaining this setpoint. This would be its biased position. As our car proceeds up a hill, however, the speed decreases for the same accelerator position. The cruise control senses through feedback an error between the setpoint speed and the actual speed. It then drives our accelerator down, increasing the amount of energy and increasing speed. The further we fall from our setpoint, the greater the error and the further down the accelerator goes. If the hill is long enough, speed will increase until we arrive back very near our setpoint speed. Once it crests the hill, the car will increase speed because of the position of the accelerator, driving us beyond the setpoint. This causes another error condition and the cruise control will 'let up' on the accelerator, allowing the car to slow until the accelerator is up back at its biased position to maintain 65 MPH.

We will control the incubator in the same fashion. For anticipated losses, we will drive the heater at a biased PWM to maintain a constant temperature. As losses change, such as a disturbance in the system, we will drive the heater at a higher or lower PWM, based on the distance from the setpoint, to bring the temperature back to the setpoint.

Let's say our setpoint is 101.5 degrees, and a bias of 50% PWM is sufficient to maintain this temperature. If the temperature drops to 101.0, giving a -0.5 degree error, do we want to drive at 100% PWM, or something lower, to bring the temperature back to the setpoint? Let's look at a concept known as Proportional Band.

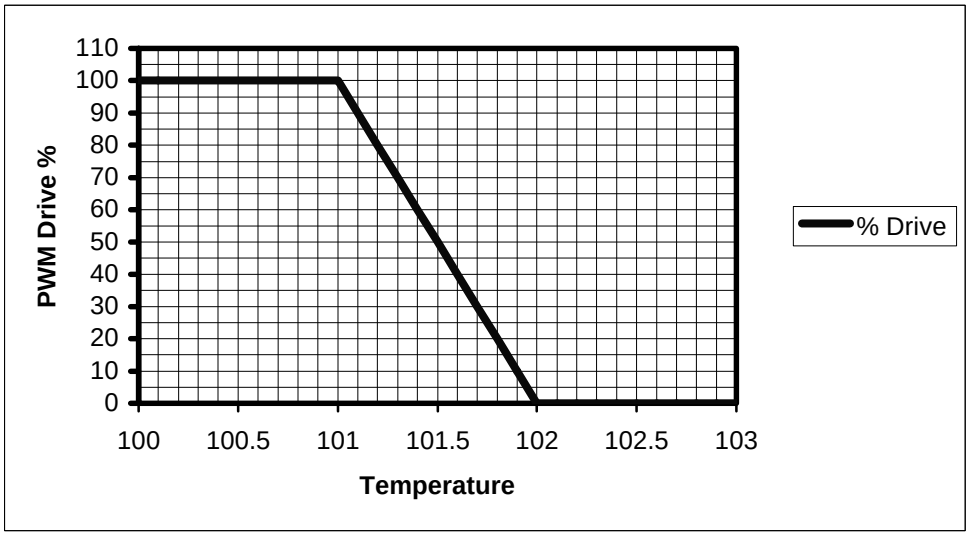
Exercises

Exercise #1: Proportional Band

For our incubator example, we want to maintain a temperature of 101.5 degrees. The steadier we maintain this temperature, the healthier our eggs will be. It is allowable, though, to go 0.5 degrees above or below this setpoint. So our setpoint will be 101.5 with a band of +/- 0.5 degrees, or 101.0 to 102.0 degrees.

If any error correction reaches the upper or lower limits, we want to take full action to return temperature back to the setpoint. Any error between the setpoint and the limits will provide a proportional amount of drive action. Figure 6.1 is a graphical representation of the incubator temperature verses the amount of drive needed for the given band.

Figure 6.1: Temperature vs. Drive



From Figure 6.1, if the temperature is at 101.5 F, 50% drive will be used to add heat energy to our system. If the temperature drops to 101 F, 100% drive will be used to increase the temperature, and any temperature in between will result in a proportional amount of drive.

If the temperature rises above 101.5 F, the drive will proportionally decrease to allow a lower temperature until at 102 F there is no drive (0%).

What would the drive be at 101.2 degrees? _____; at 101.7 degrees? _____.

This is termed a 100% Proportional Band because the drive covers 100% of our allowable band. Does this mean the temperature will not exceed our high and low limits? No. This simply means that at those limits our system will be taking full action to get the temperature back to the setpoint. The amount of drive that is added to or subtracted from the bias of 50% is the error drive. At 0.5 degrees away from the setpoint, we want our system to add or subtract 50% of drive to be fully on or fully off. To perform the math for this, we will need to consider the system gain. Over a change of one degree, we want to be able to change the drive 100%; our output change is 100% for an input change of one degree.

Proportional Gain = 100% / 1 degree

Since we have been working with tenths of degrees in the BASIC Stamp, for every tenth of a degree away from the setpoint, we will add or subtract 10% Drive for our error.

The Total Drive is the Bias Drive plus the Error Drive:

Total Drive = Bias Drive + Error Drive

The Error Drive is equal to the amount of error (Setpoint – Actual) multiplied by the Proportional Gain.

Error Drive = (Setpoint – Actual) x Proportional Gain

Let's try some numbers here. Assume that 50% bias is sufficient to maintain the incubator at 101.5 degrees under normal conditions. With normal conditions, the temperature would be 101.5 from the bias drive alone, thus no error.

Total Drive = Bias Drive + Error Drive

Total Drive = 50% + 0 % = 50%.

Looking at Figure 6.1, we see this is the correct drive for this situation.

Now, imagine we turn on a ceiling fan that blows air over our incubator, causing greater losses than expected. Energy removed is now greater than energy added, so the temperature drops. At 101.1 degrees, what amount of drive would our controller use to return the temperature to the setpoint?

Drive Error = (Setpoint – Actual) x Gain

Drive Error = (101.5 – 101.1) * 100 % = 40%.

Total Drive = 50% + 40% = 90 %.

Again, looking at Figure 6.1, we see this would be the correct amount of drive for the setpoint.

Now, let's turn off the ceiling fan and open a window so sunlight falls on our incubator. Losses would be less than anticipated, and the temperature would rise above the setpoint.

At 101.6 degrees, what would be the total drive of the system? _____.

Ok, enough talk! Let's look at this in our actual system. We have a few things to consider first. A real egg incubator is made with insulated materials and has high-energy heaters and stirring fans to equalize temperatures throughout the incubator. In our incubator:

- A bias of 50% PWM drive is insufficient to provide a temperature of 101.5.
- Every incubator will have a different stable temperature for 50% bias drive due to many factors.
- Our incubator is a fragile, steady state system. Factors such as room temperature and vents or fans blowing on the canister will shift the temperature.
- Moving or bumping the incubator will cause air mixing and affect the measured temperature.

Keeping all of this in mind, we will deviate somewhat from our incubator temperatures in testing out this section. You will use the program first to find the temperature at 50% bias drive for your incubator, and then use a band of that temperature +/- 1.0 degree for your experiments.

Note that this biased temperature band may shift over time because of varying conditions such as room temperature. You may occasionally need to redefine your bias drive.

Since our band is two degrees wide, the gain of our system will be:

Gain = 100%/2 degrees = 50%, or 5% drive change per tenth of a degree error

Circuit Construction

We will use the same circuit from Exercise #5, but we will connect the fan to Pin 19, the 5V supply, instead of Pin 20. In our control program, while individual drive values may exceed 0 and 100%, the total drive will not. Following is the full program for this section. We will change values in the program in testing the different areas of control.

```
'Program 6.1: PID Control with the StampPlot Interface

'Configure Plot
Pause 500                       'Allow buffer to clear
DEBUG "!RSET",CR                'Reset plot to clear data
DEBUG "!TITL PID Control",CR    'Caption form
DEBUG "!PNTS 400",CR            '400 sample data points
DEBUG "!TMAX 600",CR            'Max 600 seconds
DEBUG "!SPAN 80,100",CR         '70-120 degrees
DEBUG "!AMUL .1",CR             'Multiply data by .1
DEBUG "!DELD",CR                'Delete Data File
DEBUG "!CLMM",CR                'Clear Min/Max
DEBUG "!CLRM",CR                'Clear Messages
DEBUG "!USRS  ",CR              'Clear User status bar
DEBUG "!SAVD ON",CR             'Save Data
DEBUG "!TSMP ON",CR             'Time Stamp On
DEBUG "!SHFT ON",CR             'Enable plot shift
DEBUG "!PLOT ON",CR             'Start Plotting
DEBUG "!MAXS",CR                'Stop at Max
DEBUG "!RSET",CR                'Reset plot to time 0

' Define constants & variables

CS          con    3            ' 0831 chip select active low from BS2 (P3)
CLK         con    4            ' Clock pulse from BS2 (P4) to 0831
Dout        con    5            ' Serial data output from 0831 to BS2 (P5)
Datain      var    byte         ' Variable to hold incoming number (0 to 255)
Temp        var    word         ' Hold the converted value representing temp
TempSpan    con    5000         ' Full Scale input span in tenths of degrees.
                                ' Declare span.  Set Vref to .50V and
                                ' 0 to 255 res. will be spread over 50
'(1/100ths).

Offset      con    700          ' Minimum temp. Offset, ADC = 0
```

```
' Declare zero Temp. Set Vin(-) to .7 and
' Offset will be 700 tenths degrees.
' At these 'settings, ADC output will be 0 - 255 for temps
' of 700 to ‘1200 tenths of degrees.

Error        Var    Word   ' Amount of drive due to error
Drive        var    Word   ' Amount of total drive
Integral     var    word   ' Amount of Integral Drive
Integral = 0

PWMCount     var    byte   ' Counter for amount time to apply PWM

LastTemp     var    Word   ' Holds last temperature for derivative drive
LastTemp = 0

Deriv  var    Word         ' Holds amount of Derivative Drive
Deriv = 0

IntCount     var    byte   ' Holds variable for counting cycles for integral drive
                           ' One cycle one PWM Time
PWMTime      Con    20     ' Number of repetitions for PWM, 20 = ~ 5 seconds

'********* ESTABLISH CONTROL SETTING ************

Bias         con    50     'Bias drive setting

Setpoint     con    924
'Initialize setpoint to YOUR bias Temp (mine was 92.4 F)

PropGain            con    5
'Set gain. 5 = 100% Proportional Gain, 1= 500%, 10 = 50%
IntTime      con    0
'How often to update integral amount in 5 second increments (6 = ~30 sec)
IntStep      con    0
'The amount to step integral drive each update
DerivGain    con    0
'Gain for each tenth degree change from previous


'************************************************
' Update Status text box
DEBUG "!USRS Setpoint:", DEC Setpoint, "  P-Gain:", DEC PropGain
DEBUG "  Int-Time:", DEC IntTime * 5,"  Int-Step:",DEC IntStep,"  Deriv-Gain:", DEC
DerivGain,CR
```

```
Main:
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOTO Main

Getdata:                          'Acquire conversion from 0831
  LOW CS                          'Select the chip
  LOW CLK                         'Ready the clock line.
  PULSOUT CLK,10                  'Send a 10 uS clock pulse
                                  'to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8]       'Shift in data
  High CS                         'Stop conversion
RETURN

Calc_Temp:                        'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset     'temp based on Span &
RETURN                            'Offset variables.

Control:
  GOSUB PropCalc                  'Perform proportional error calcs
  GOSUB IntCalc                   'Perform Integral Calcs
  GOSUB DerivCalc                 'Perform Derivative calcs
  Drive = (Bias + Error + Integral + Deriv)          'calculate total drive
  IF Drive < 10000 then HighAdjust      'adjust for signed number exceeding 0-100
  Drive = 0                       'Min of 0

HighAdjust:
  If (Drive <= 100) Then DriveOK
  Drive = 100                     'Max of 100

DriveOK:
  Drive = 50        '***** TEMPORARY TO FIND INCUBATOR BIAS TEMPERATURE
  DEBUG DEC Temp,CR               'Plot and display data
  DEBUG "Temp:", DEC Temp, "  Bias:", DEC BIAS, "  Error:", SDEC Error
  Debug "  Int:", SDEC Integral, "  Deriv:", SDEC Deriv, "  Drive:",DEC Drive, CR
  FOR PWMCount = 1 TO PWMTime     'Pwm at 218 mSec for each PWMTime repetition
  PWM 8,drive * 255/100,218
NEXT
RETURN


'***********Proportional Drive Error
PropCalc:
  Error = Setpoint - Temp * PropGain  'Calculate error
Return
```

```
'********** Integral Drive
IntCalc:
  IntCount = IntCount + 1               'Add one to IntCount
  If IntCount = IntTime then IntAdjust
'Every IntTime counts (~5 seconds each) adjust Integral
Return

IntAdjust
  If Temp = Setpoint THEN IntDone:
'If temp > setpoint, subtract one step
  If Temp < Setpoint Then IntAdd
  Integral = Integral - IntStep
Goto IntDone

IntAdd:                                 'Otherwise add it
  Integral = Integral + IntStep

IntDone:
  IntCount = 0                          'Reset integral counter
Return

'*********** DERIVATIVE DRIVE
DerivCalc:

If LastTemp = 0 then DerivDone          'If first reading, skip
  Deriv = (LastTemp - Temp) * DerivGain
                                        'Calculate amount of derivative drive
                                        'based on the difference two temps
                                        'mutliplied by the deriv. gain
DerivDone
  LastTemp = Temp   'Store temp for next deriv calc
Return
```

Once the code is entered:

Enter and run the above program. Ensure the fan is pointed well away from the incubator.

1. Start StampPlot Lite and monitor the temperature.

2. Once the system is fairly stable, record the temperature at 50% Bias Drive.

3. In the control setting, make the Setpoint temperature the same as your bias-driven temperature. Remember, it is in tenths of degrees (924 = 92.4 degrees).

4. Remark out the line 'Drive = 50' in the Drive Heater section by placing an apostrophe in front of it.

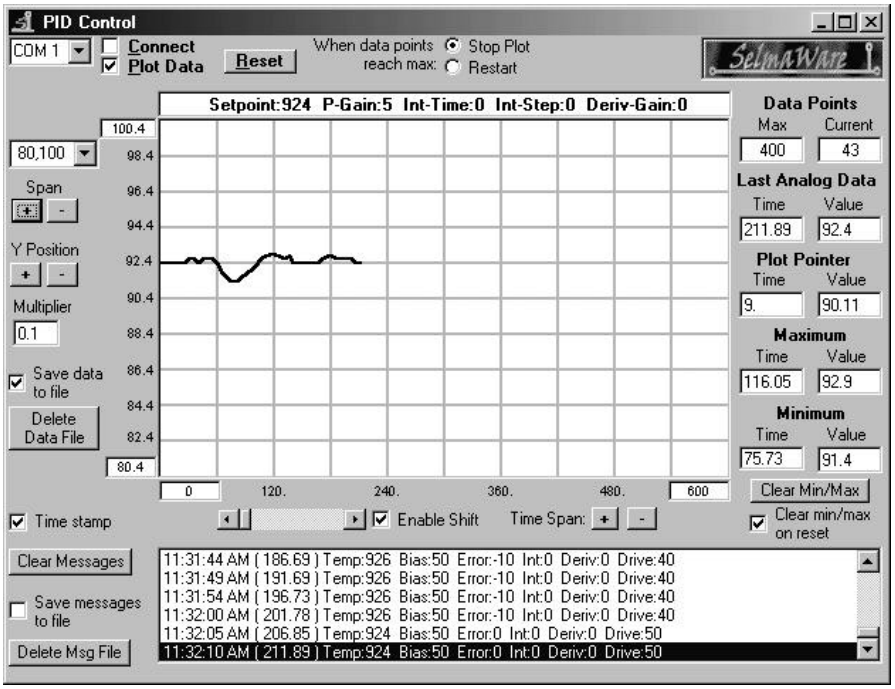
5. Disconnect the StampPlot Lite program, download the new program, close Debug and reconnect on StampPlot Lite.

6. Reset the BASIC Stamp by pressing the reset button on the Board of Education.

After 60 seconds of running a graph (1st grid line), create a system disturbance by blowing on the incubator at very close range for 10 seconds and monitor the results.

Figure 6.2 shows the results of our test with a temperature biased at 92.4 degrees with a disturbance of 60 seconds. The analog values were adjusted to best display the plot.

Figure 6.2: Test Results with a Temperature Biased at 92.4 Degrees and a Disturbance of 60 Seconds



In the message section of StampPlot Lite, we can see the amount of Bias, Error, and (Total) Drive. In an example at 92.4 degrees, we see:

BIAS: 50 Error: 0 Drive: 50

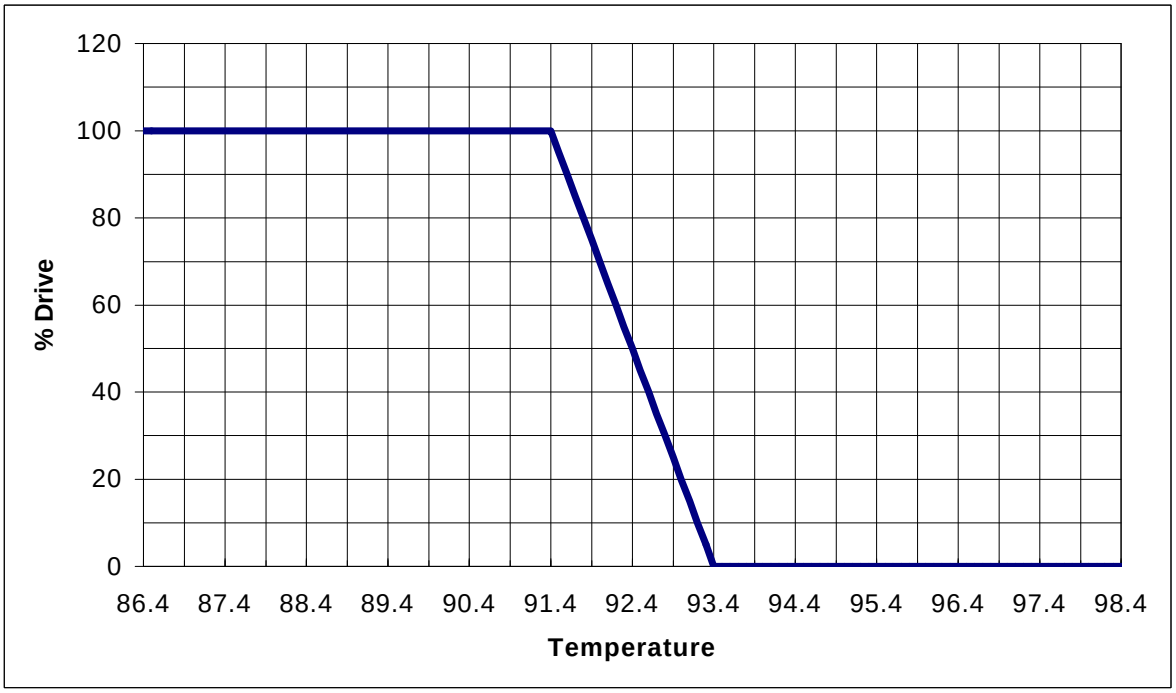
The drive is only from the Bias, since we are at the setpoint.

At 92.6 degrees: BIAS, 50; Error, -10; Drive, 40. Since we are two tenths above, we have an error.

Drive Error = (Setpoint – Actual) x Gain
Drive Error = (92.4 – 94.6) * 50 % = -10%
Total Drive = 50% + (-10%) = 40 %

When the temperature drops below the setpoint, the error becomes positive, and the total drive rises above 50% in an effort to bring the temperature up. Figure 6.3 shows the amount of drive versus the temperature for our plot.

Figure 6.3: 100% Proportional Band with 92.4 F Setpoint, +/- 1 Degree Band

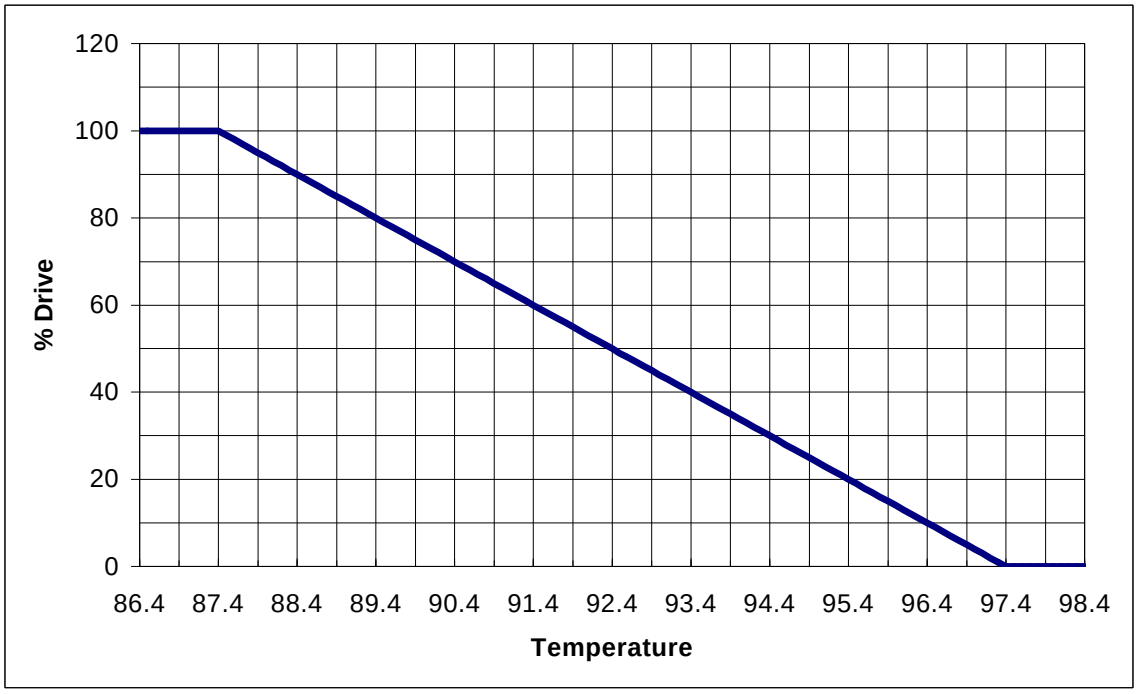


Notice that as the proportional control drives the heater to raise temperature, there is hunting. The temperature varies above and below the setpoint, hunting (adding and subtracting drive) until it finally stabilizes after several cycles.

The amount of drive we want to return to our setpoint is based on our proportional band and, therefore, our gain. For example, what if it was more important to stabilize our temperature than it was to return quickly and overshoot and undershoot getting there. In our cars, it would be an interesting ride if they sped up and slowed down repeatedly while re-adjusting for a new road condition!

In using a different proportional band setting, neither our setpoint nor limits change, we just set the gain so that we have the same amount of error drive over a greater range. For instance, in this example, our limits are still 91.4-93.4 degrees (92.4 +/- 1), but we will take full action over +/- 5.0 degrees. Full control would cover 500% of our allowable band.

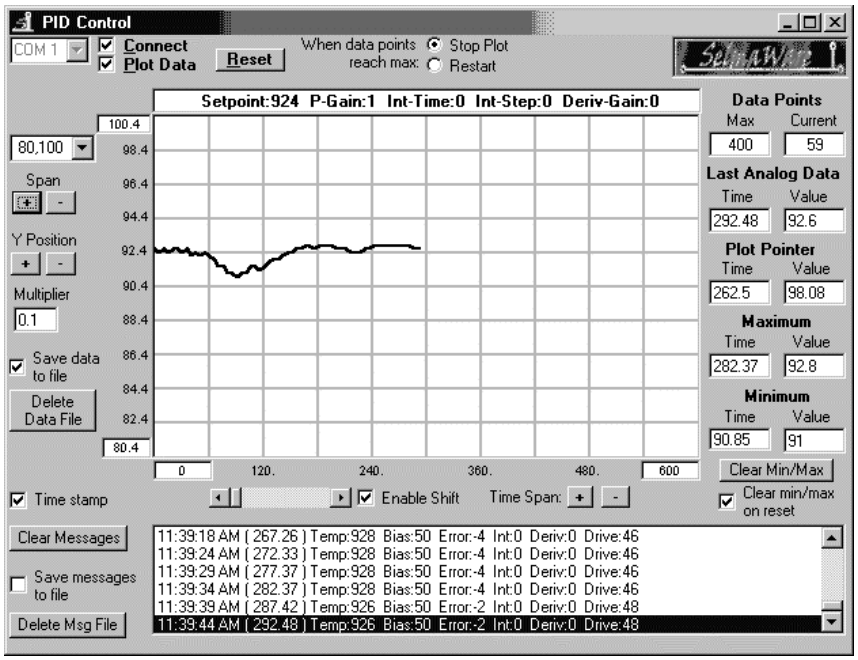
Figure 6.4: 500% Proportional Band with 92.4 F Setpoint, +/- 5 Degree Band



1. In your program locate **`PropGain = 5`** in the Control Settings section and change it to 1. (100% / 10 degrees = 10% or 1% per tenth of degree).
2. Re-run the plot with the same disturbance.

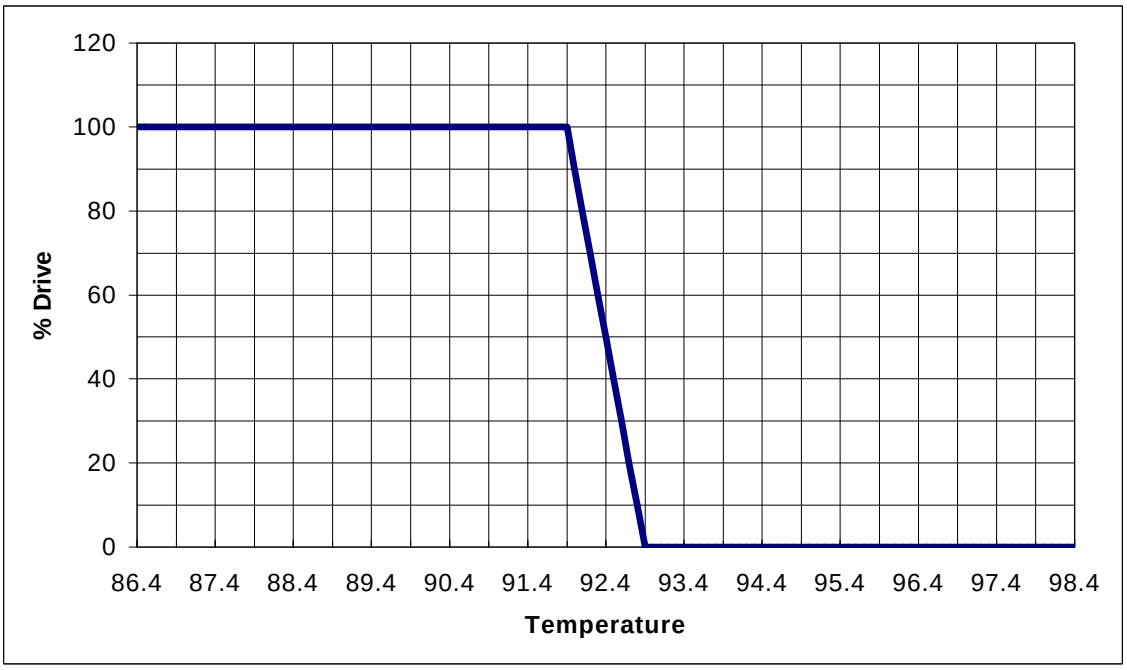
Figure 6.5 shows the plot of our control action. Notice that there is little overshoot and hunting, though it takes longer to reach our temperature since the drive for the same error is five times less.

Figure 6.5: System Response with 500% Proportional Band



Conversely, let's say we needed faster response and overshoot wasn't a concern. We could take full drive over 50% of our allowable band, so that at +0.5 we would have 0% drive and at -0.5 we would have 100% drive. Figure 6.6 illustrates the Drive for 50% Proportional Band.

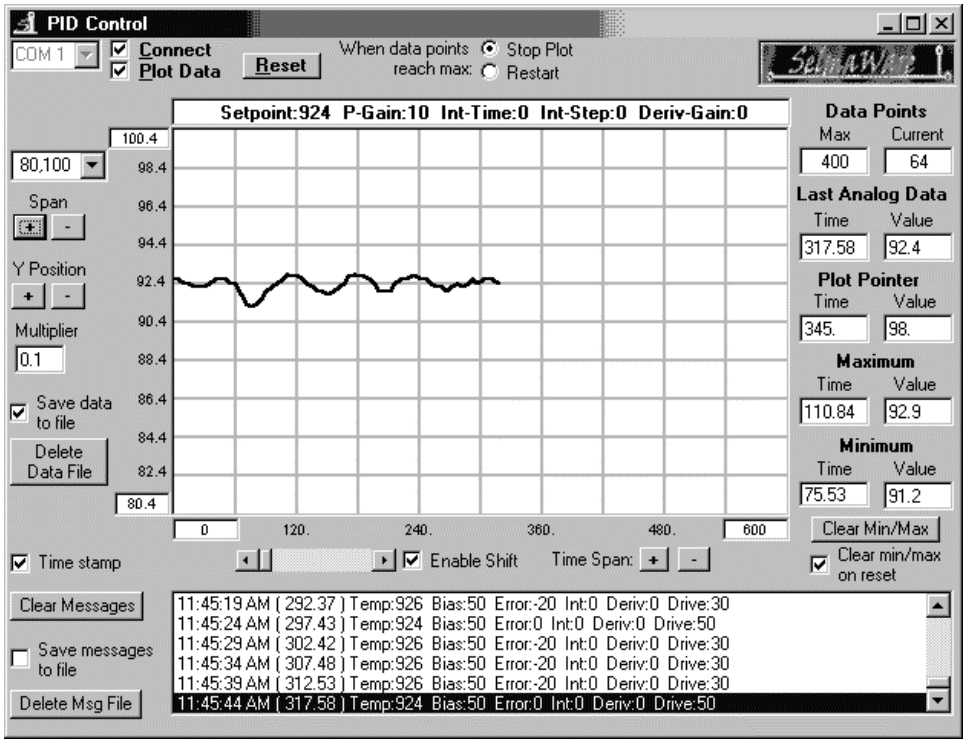
Figure 6.6: 50% Proportional Band with 92.4 F Setpoint, +/- .5 Degree Band



What would our new gain be? 100%/1 degree or 10%/tenth of a degree.

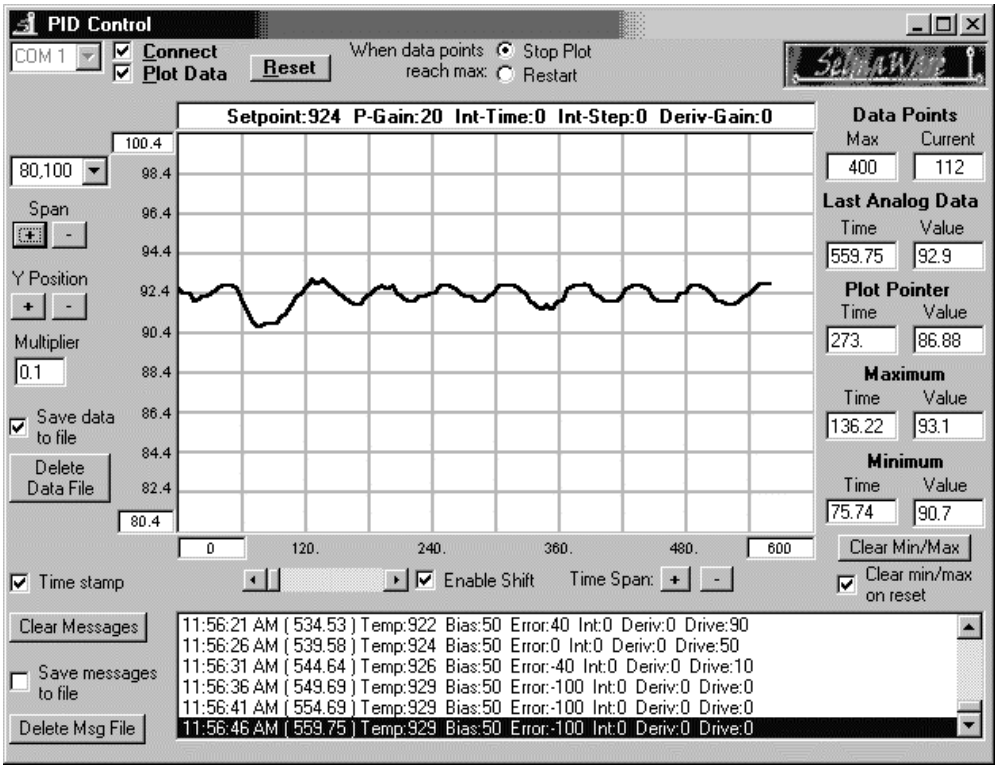
Once the plot is stable, change the gain to 10 and re-plot. Figure 6.7 shows our results. Notice that while response is faster, there is a lot more hunting before the temperature finally stabilizes.

Figure 6.7: System Response with 50% Proportional Band



In system control, there can be too much of a good thing. When we try to drive a system faster than it can respond, the resulting action could destabilize the system and lead to undesirable behavior. Figure 6.8 shows a plot of our incubator using a 25% proportional band for a gain of 20. This means that for every one-tenth of a degree change, we adjust the drive by 20%!

Figure 6.8: System Response Using a 25% Proportional Band



By driving the system faster than it can properly respond, the temperature not only hunts dramatically, but the oscillations begin getting larger, leading to system instability.

Challenge!

1. Adjust the system gain for a:
   - 100% Band
   - 200% Band
   - 50 % Band

2. Complete the following values for the Error Drive and Total Drive following a disturbance: (If the temperatures do not reach all values, fill in calculated values and circle)

Temp. from Setpoint	50% Band			100% Band			200 % Band		
	Bias Drive	Error Drive	Total Drive	Bias Drive	Error Drive	Total Drive	Bias Drive	Error Drive	Total Drive
+1.0									
+.8									
+.6									
+.4									
+.2									
0									
-.2									
-.4									
-.6									
-.8									
-1.0									

Exercise #2: Proportional-Integral Control

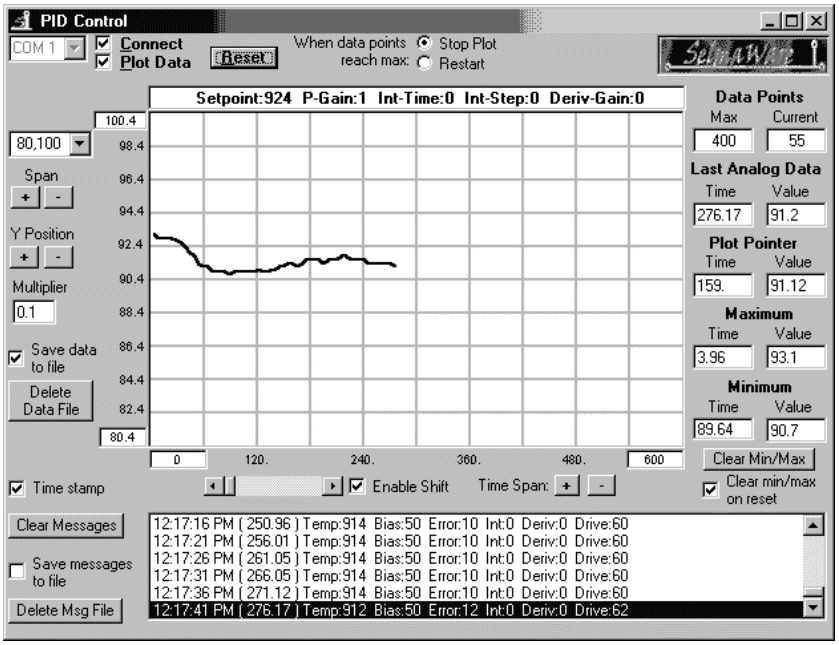
So far we've looked at what occurs when quick disturbances occur to our system in equilibrium. Proportional control may be used to drive the temperature back to the desired setpoint. But what happens when the disturbance is long enough to affect the equilibrium of our system over a long period of time?

Try this:

1) Setup Program 6.1 for a 500% Proportional Band (Gain of 1).
2) Allow the system to stabilize.
3) Point the fan at the incubator from a distance of about two feet (if you see no response after 30 seconds, move it closer in six-inch increments and try again).
4) Allow the system to stabilize with this new system loss.

Figure 6.9 shows our results of this test with an equilibrium setpoint of 92.4 F.

Figure 6.9: Long-Term Disturbance Effects



Notice that even with a drive of 62%, the temperature is approximately 1.2 degrees away from the setpoint with an error of 12%. With a short disturbance, this error would have been sufficient to drive the temperature back up. Now we have a long-term loss from our system and the error drive is insufficient to make up for it. As long as the system is out of equilibrium, where heat loss is greater than heat gain at 50% drive, the temperature will not be stable again at our setpoint.

If we were to increase the gain, to say 10, it would cause a greater drive and push the temperature closer to the setpoint, but there will always exist some error. Let's look at why. Let's say that we could reach the setpoint of 92.4 F. How much drive would there be? Since the Error drive would be 0, there would be 50%, and all from Bias. However, we know with our disturbance that 50% is insufficient to maintain that temperature; therefore, it would have to be less the 92.4 F, providing an error. If the temperature returns to the setpoint, there will be no error and thus no error drive, but we need the error to have sufficient energy to increase the temperature, so some error MUST remain. It can be confusing, but it has to work this way if you think about it.

With the long-term disturbance, what we reach is a new equilibrium where the energy added to meet the loss is equal to the bias + the Error drive at a lower temperature. In this case, the disturbance was a fan, but it may have been the temperature of the room changing over the course of the day, or even the season of the year. (Though it is easier to discuss greater losses, we could also have started at a lower room temperature and have it heated over the day, reducing our losses and causing our system to stabilize at a higher temperature with an error!)

So how do we maintain the desired setpoint in the face of long-term losses to our system? We use integral control. Integral control is used to slowly add to the total drive over a long period of time, driving the Actual value back to the setpoint and reducing the error.

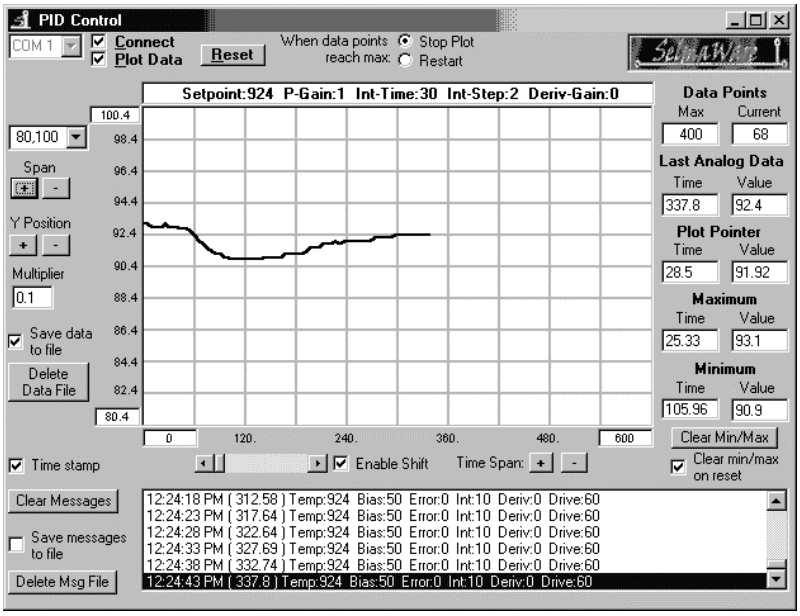
Set the control settings to the following:

```
Bias          con    50            'Bias drive setting

Setpoint      con    YOURS         'Initialize setpoint to YOUR bias Temp in
                                   'tenths of degrees

PropGain      con    1             'Set gain. 5 = 100% Proportional Gain,
                                   '1= 500%, 10 = 50%
IntTime       con    6             'How often to update integral amount in
                                   '5 second increments (6 = ~30 sec)
IntStep       con    2             'The amount to step integral each update
DerivGain con        0             'Gain for each tenth degree change from previous
```

We just added integral control to our process. In the **`IntCalc`** routine, every six repetitions (about 30 seconds), we adjust Integral drive by a step of two. If the temperature is below the setpoint, two will be added to the integral control. If above the setpoint, two will be subtracted. In this manner, the program will add or subtract drive over a long period of time, adjusting until it eventually drives away any error.

Run this program testing the long-term disturbance of the fan. Figure 6.10 shows our results.

Figure 6.10: Long-Term Disturbance Using Integral Control



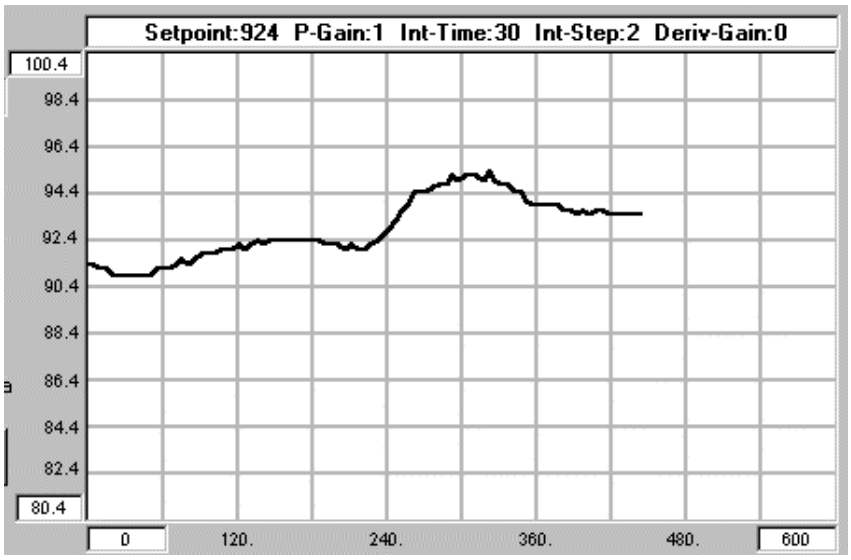
Notice that every 30 seconds or so (every half-division), the temperature bumps up slightly. At the end of the run, there is a 60% drive: 50% from bias, 0% from Error and 10% from integral.

Total Drive = Bias Drive + Error Drive + Integral Drive

Over long periods of time, as environmental temperatures change or perhaps our oil piping erodes and causes greater friction, integral control may be used to drive out these changes in equilibrium, returning the system to its desired operating point.

Of course, one drawback is what occurs when the long-term disturbance is suddenly removed. Figure 6.11 shows what occurs when the fan is turned off.

Figure 6.11: Control After the Disturbance Removal



When the disturbance leaves our system, all of that added integral drive still is present, causing the temperature to go way above the setpoint of 92.4 F. The error drive then has to compensate until, over time, the integral drive slowly is subtracted away.

Challenges!

1) If possible, heat the room to higher-than-normal temperature. Establish equilibrium with 50% bias at the higher room temperature. Return the room temperature to normal, and observe the effects on the system.

2) Change the amount of integral drive added (IntStep) from two to five. Test the system using the fan and notice the changes in the system's response time.

3) Increase the frequency adjustment rate for the integral drive (change IntCount from six to two). Test the system response.

Exercise #3: Derivative Control

In derivative control, we take action based on a change in readings over a very short time, typically from one reading to the next. This allows for an immediate amount of drive change to counter a rapid disturbance, quickly stabilizing the system.

Derivative Drive = (Change in temp) / (Change in time) x Derivative Gain

In our program's **`DerivCalc`** routine, we take the difference between the last temperature reading and the current temperature reading and multiply it by the derivative gain. Let's look at an example for a 500% proportional band (Gain = 1).

If a disturbance causes the temperature to drop from 92.4 degrees to 92.0 degrees, the amount of drive from the proportional error would increase the drive by only 4%. The rapid change in temperature, though, with a derivative gain of five, would add 20% drive instantly in an effort to dampen the sudden drop.

Last reading: 92.4. This reading: 92.0

Derivative Drive = (92.4-92.0)/1 unit * 5% = 20%

Total Drive = Bias + Error + Integral + Derivative

Total Drive = 50% + 4% + 0% + 20% = 74%

Remember, the amount of derivative drive is only based on the long-term change in temperature. If, at the next reading, the added total of 24% drive was sufficient to stop the sudden drop, and the reading was again 92.0 degrees, the only additional drive would be from Proportional at 4%.

Last reading: 92.0 This reading: 92.0

Derivative Drive = (92.0-92.0)/1 unit * 5% = 0%

Total Drive = 50% + 4% + 0% + 0%

If there were a sudden increase from 92.0 to 92.6, what would the total drive be?

Last reading: 92.0. This reading: 92.6

Derivative Drive = (92.0-92.6)/1 unit * 5% = -30%

Total Drive = Bias + Error + Integral + Derivative

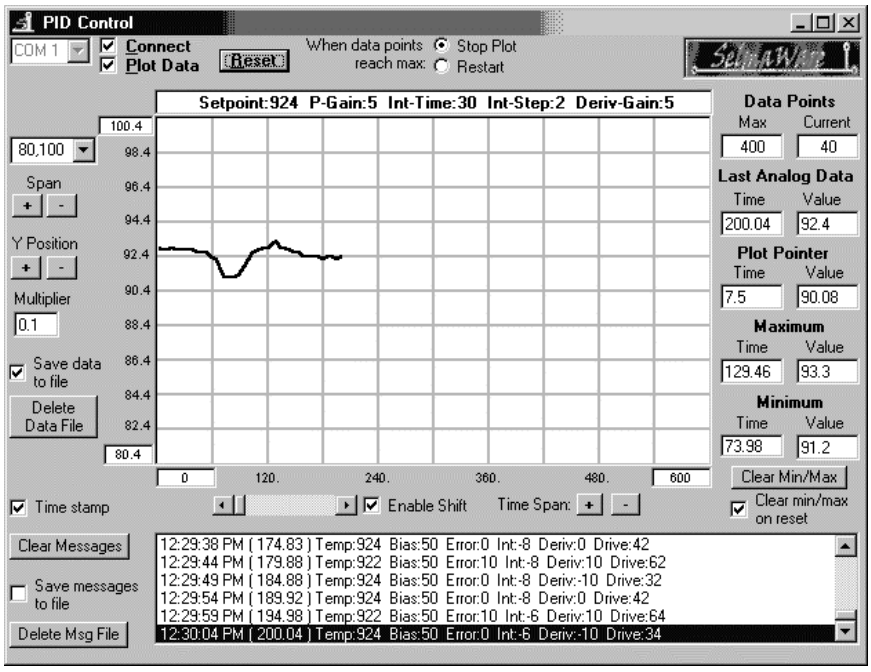
Total Drive = 50% + 2% + 0% + (-30%) = 22%

As we can see, if the temperature suddenly jumped from below the setpoint to above it, the amount of derivative drive would be negative for the instant, setting the PWM drive very low in an effort to slow the rapid change.

Let's look at some examples using StampPlot Lite and our incubator. Configure your program for the following 100% Proportional band (PropGain = 5), the integral drive, and a gain of five for Derivative drive, and run your plot with a 10-second blow disturbance.

Figure 6.12 shows the result of our test. Compare it to Figure 6.2.

Figure 6.12: PID Control: 100% Proportional Band



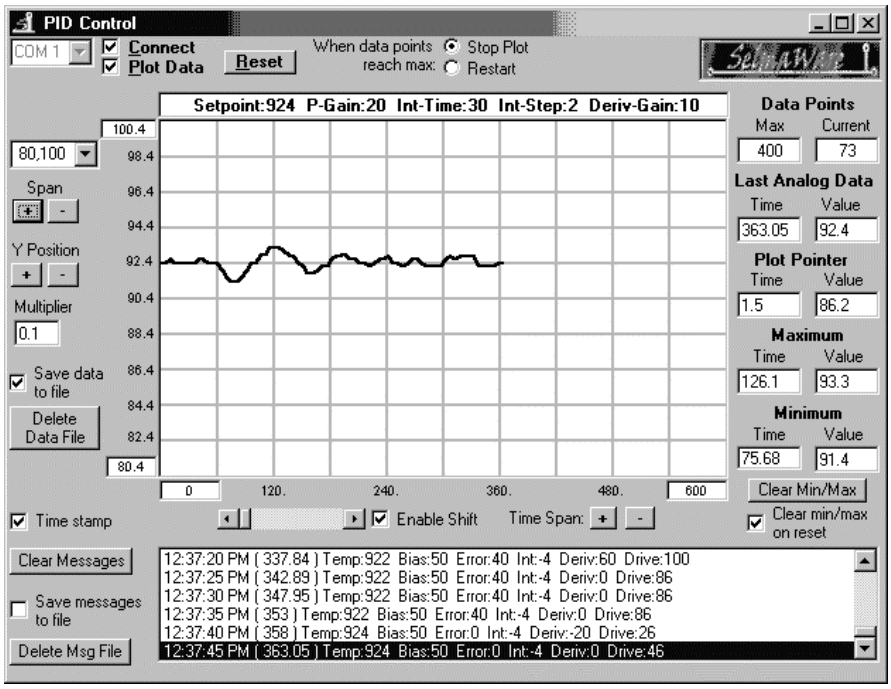
Compared to Figure 6.2, the sudden drop turned more quickly and the plot stabilized faster. Note in the message window the small fluctuations in the final temperature, creating derivative drive to try and stabilize the system.

Look back at Figure 6.8, where we plotted only proportional control with a gain of 20. The temperature oscillated endlessly, trying to stabilize. In your program, change the Proportional Gain to 20 and the derivative gain to 10 and try it again.

Figure 6.13 shows the results of our test. Revise your program as follows:

```
Bias                con    50     'Bias drive setting

Setpoint            con    YOURS  'Initialize setpoint to YOUR bias Temp in tenths

PropGain            con    5      'Set gain. 5 = 100% Proportioanl Gain,
                                  '1= 500%, 10 = 50%
IntTime             con    6      'How often to update integral amount in
                                  '5 second increments (6 = ~30 sec)
IntStep             con    2      'The amount to setp integral each update
DerivGain           con    5      'Gain for each tenth degree change from previous
```

Figure 6.13: PID Control for 25% Band



In this example with derivative control, the temperature more quickly stabilized as the derivative gain helped dampen overshoot and undershoot.

Once again, too much of a good thing can be bad. Improper derivative gains can totally destabilize a system where slight disturbances can dramatically affect the control. We didn't have any luck stabilizing our slowly responding system with derivative control. Can you?

Challenge!

For a 100% Proportional band, test various derivative gains with a disturbance. What setting allows the fastest return to the setpoint with minimal overshoot?

Proportional, Integral, Derivative Summary

Using PWM control, we can drive our system in small increments over a wide range. Through the use of proportional control, we can add to or subtract from that drive based on the error between the actual reading and the setpoint. By defining our band and adjusting the error gain, we can establish how quickly the system will take action based on the error. Small amounts of gain allow a slow, controlled return following a disturbance. High amounts of gain allow a faster return, though it may lead to instabilities.

Long-lasting disturbances prevent the actual value from reaching the setpoint because of the imbalance and the error drive needed to make up for the disturbance. Using integral control over long periods of time, we can add or subtract drive to compensate for the error and return the system to the setpoint.

Rapid disturbances can be effectively dampened through the use of derivative control. Derivative control adds to or subtracts drive based on changes in readings over a very short time. Effective use of derivative control can rapidly stabilize systems.

As a final example, consider that some inertial navigation systems use a tiny ball spinning at thousands of revolutions per second while suspended in a magnetic field to measure the speed and direction of a craft. Imagine the PID control employed to keep the ball suspended and rotating. Improper control would cause the ball to slam against the enclosure, destroying it. Not all systems are as slow as our incubator. Some must react to changes in the system within hundredths or millionths of a second. Proficiency at tuning some systems for proper PID control can take years of education and experience. Entire volumes have been written on the subject. Improper control of a system can lead to expensive, and sometimes disastrous, consequences.

Questions and Challenge

1. Would on/off control of the system be suitable for PID control? Explain.

2. Which type of control, proportional, integral, or derivative, would be best suited for the following?

   a. To return a system to the setpoint based on the difference between Actual temperature and the setpoint due to a short-lived disturbance: _____.
   b. To minimize the effect a quick disturbance has on the system: _____.
   c. To reduce the effect a long-term disturbance has on the system: _____.

3. A system has a setpoint of 101.5 degrees, and an allowable band of +/- 0.5 degrees. For a 50% proportional band, what would be the proportional gain? _____.

4. A system has a setpoint of 101.5 with a gain of 10%/0.1 degrees. If the Actual temperature were 101.2, what would be the drive due to proportional error? _____

5. A system increments the Integral drive by 1% every two hours. If a long-term disturbance creates an error of 8%, how long before the error is driven away? _____.

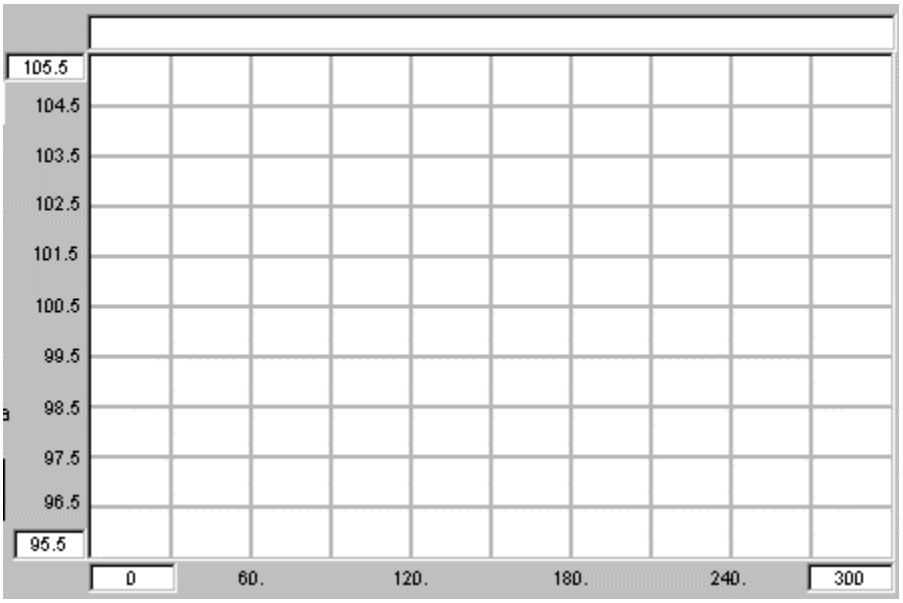
6. A system has a derivative gain of 10%/ 0.1 F. If the temperature dropped from 101.8 to 101.3 between readings, with a setpoint of 101.5, what would be the error due to derivative drive? _____.

Final Control Challenge

From a cold condition (incubator at room temperature), find the values of PID control which will bring the incubator to an operating temperature of 95 degrees the quickest with minimal overshoot and hunting. Graph and record your results (note the graph scales):

PropGain: _____ IntTime: _____ IntStep: _____ DerivGain: _____

Graph your results:



Time first reached 101.5: _____ Maximum value reached: _____

Next minimum reached: (Time)_____ (Value) _____

Find a system:
Find an example of a system that either does or could employ PID control. Discuss how PID control may be implemented to control it.

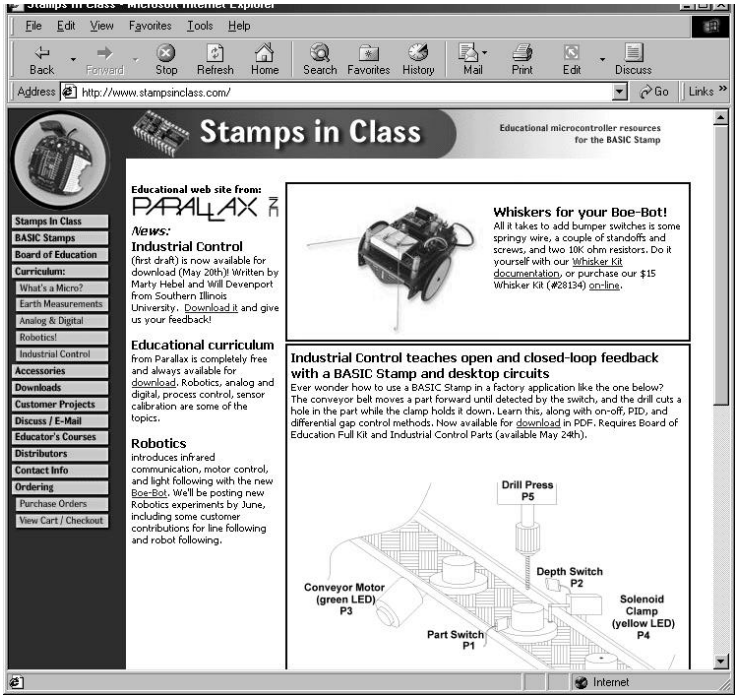
Appendix A: StampPlot Lite

StampPlot Lite is an application developed by SelmaWare Solutions for the Industrial Control series. The application allows plotting and capture of analog, digital and general data.

Downloading and Installing StampPlot Lite

StampPlot Lite may be downloaded from the Stamps in Class web site at http://www.stampsinclass.com. The program is installed by double-clicking on the setup.exe icon and accepting the default directories.

To download StampPlot Lite, click on the "Downloads" button on our web site and scroll down to the "Industrial Control" section.



Data from the BASIC Stamp is processed in one of four ways by the application:

Analog Values

Any string sent beginning with a numeric value will be processed as an analog value and graphed.

```
Debug DEC 100, 13          'Plot the number 100
```

Digital Values

Any string sent beginning with '%' will be processed as digital values. A separate digital plot will be started for each binary value in the string. For example, "%1001" will plot four digital values. Up to a 9-bit value may be sent. Once digital plots are started, caution should be used to always send the same number of bits since the plots are position-order dependent.

```
Debug IBIN4 INC,13         'Plots 4 digital values
```

Control Settings

Any string beginning with '!' will be processed as a control setting. The various settings of the application may be controlled from the BASIC Stamp using specified control words and values, if required.

```
Debug "!AMAX 200",13       'Sets analog maximum for plot to 200
Debug "!RSET",13           'Resets the plot
```

Other Strings

All other strings simply will be added to the running message list box.

```
Debug "Hello world!",13
```

Note that each instance of data MUST end with a carriage return (13 or CR).

The steps for using BASIC Stamp programs with StampPlot Lite are as follows:

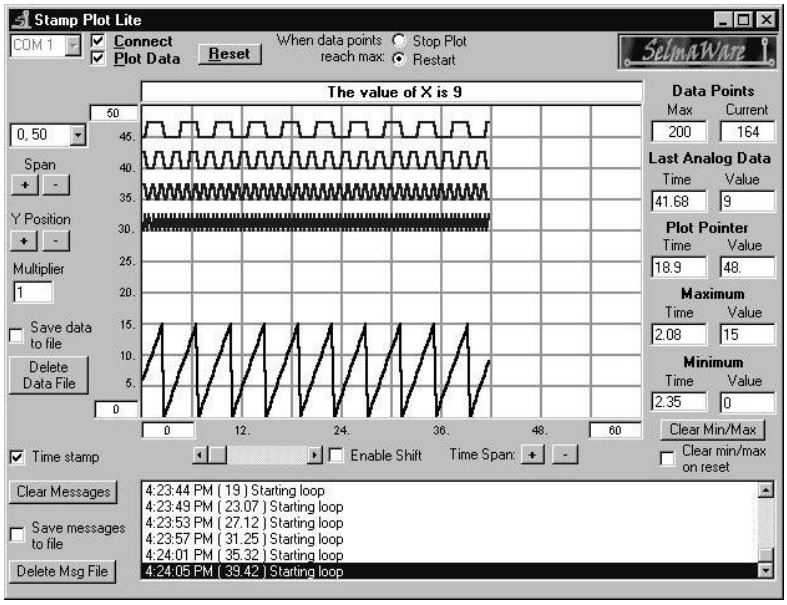
1. Start StampPlot Lite through your Start/Programs/StampPlot/StampPlot Lite icon.
2. Enter and run your BASIC Stamp program from the BASIC Stamp editor.
3. Close the blue BASIC Stamp debug window by clicking on the "close" box.
4. Select the COM port and click 'Connect' checkbox.
5. Click the 'Plot Data' checkbox.
6. Some programs may require you to reset the Board of Education (BASIC Stamp) to catch initial configuration and control settings. Do this by pressing the Reset button on the board.
7. Prior to downloading (running) another program to the BASIC Stamp, be sure to uncheck the StampPlot 'Connect' checkbox or your COM port will be locked by StampPlot Lite.

The plot will acquire analog and digital data and store it temporarily so that it may be resized or shifted on the screen. The number of data points collected is adjustable. Once the data points reach maximum, the plot must either be stopped or reset.

The following program will perform some configuration settings, continually plot and display the value of X on StampPlot, and plot the four digital bits of the value of X. Enter the program and use the steps above to test it with StampPlot Lite.

```
'Appendix A Program, StampPlot Example
'Configure StampPlot              'Variable for counting
Pause 500: Debug "!RSET",CR       'Short pause and reset
Debug "!SPAN 0, 50",CR            'Span the analog range
Debug "!TSMP ON",CR               'Time Stamp the messages
Debug "!TMAX 60",CR               'Set plot to 60 seconds max
Debug "!RSET",CR                  'Reset the plot

X var Byte
Loop:
Debug "Starting loop", CR         'Message that loop is resetting.
For X = 0 to 15                   'For-Next loop to count to 15
Debug DEC X, CR                   'Plot Analog value of X
Debug IBIN4 X, CR                 'Plot digital bits of X
                                  'Change the User Status message.
Debug "!USRS The value of X is ", DEC X, CR
      Pause 200                   'Short pause
Next
Goto Loop                         'Restart
```

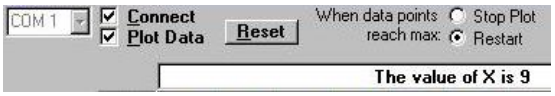
Below is a screen shot of StampPlot Lite showing the above program plotted.



Tool Help

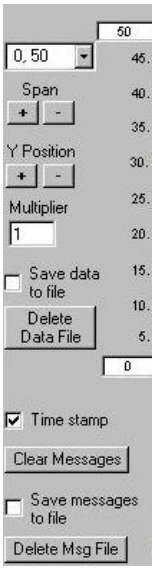
If a copy of StampPlot Lite is running on your computer, you may place the cursor over each control for 'Tool Text Help.' The following is a brief summary of each control. The BASIC Stamp programmable command, where applicable, is in brackets:

Top Section: General Controls



- Com 1: Drop down to select the applicable COM port.
- Connect: Connects the application to the selected COM port.
- Plot Data: Allows plotting of incoming data. Deselecting this control will stop the plotting of data but will allow messages and other actions to continue. [!PLOT ON/OFF]
- Reset: Clears the plot, resets to time 0, clears minimum and maximum value (optional). [!RSET]
- Stop Plot: When maximum data points are reached, the plot stops (Plot Data becomes unchecked). [!MAXS]
- Reset Plot: When maximum data points are reached, the plot resets. [!MAXR]
- User Status: (showing "The value of X is 9") Optional status messages from the BASIC Stamp may place data here. [!USRS message]

Left Section: Primarily for Setting the Analog Plot



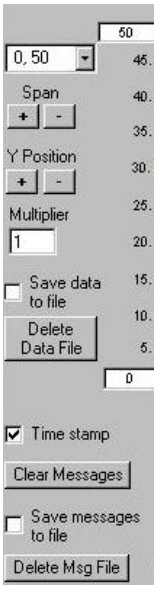
- Span Drop-Down box: Allows a selection of pre-defined plot ranges. Use of the BASIC Stamp command !SPAN will add a range to this drop-down. [!SPAN minvalue, maxvalue]
- + and - buttons: Respectively double or halve the span. The minimum value does not change.
- Multiplier: Defines the amount incoming BASIC Stamp analog data will be multiplied by prior to plotting or saving to file. [!AMUL value]
- Save Data to File: Saves the incoming data to a text file in the application directory called "stampdat.txt." If time stamping is enabled, each record will be marked with the current system time and the number of seconds since the last reset. The value of the data point for analog and digital values will also be recorded. Each record will have the following form:
  - Time of day, seconds since reset, analog data point, analog value, digital data point, digital data value. Note that each record is comma delineated for importing into a spreadsheet, if desired.
  - *Note: Data is saved ONLY when ANALOG data arrives. To force saving when no analog data is recorded, debug a value such as zero (DEBUG DEC 0, CR).[!SAVD ON/OFF]*
- Delete Data File: Deletes "stampdat.txt." If data saving is enabled, the file will be re-created after deleting it. [!DELD]
- Analog Minimum and Maximum values: These may be manually changed. Tab off, or click another control, to set the new value. [!AMIN value !AMAX value <or> !SPAN minvalue, maxvalue]
- Time Stamp: Enables time stamping of messages and data to the file. It includes both the current time and seconds since the last reset. [!TSMP ON/OFF]
- Clear Messages: Clears the messages in the listbox. [!CLRM]
- Save messages to file: Saves messages to the file "stampmsg.txt" in the application directory. Messages will be saved the same way they appear in the message box. [!SAVM ON/OFF]
- Delete Msg file: Deletes "stampmsg.txt" in the application directory. If the "Save Messages..." is enabled, the file will be re-created. [!DELM]

Bottom Section: Plot Shift and Time Span

- The minimum and maximum times of the plot may be set manually. Tab-off or click another control to apply the value. [!TMIN value and !TMAX value]
- Scroll Bar: If the plot extends beyond the current limits, the scroll bar may be used to reposition the plot (if collecting data, Enable Shift must be on).
- Enable Shift: Allows the plot to shift automatically when maximum plotted time is exceeded. Also enables operation of the scroll bar when collecting data. Note: Shifting of the plot during data collection may cause time errors in the data as the plot refreshes. [!SHFT ON/OFF]
- +/-: Respectively doubles or halves the time span of the plot. The minimum value of the plot will not change.

Right Section: Y Axis Control and Data File Saving

- Data Points: To allow the plot to be manipulated, data is stored in memory. The maximum number of points (either analog or digital) that may be recorded is set by Max. 'Current' displays the current data point being stored. Once the maximum is reached, the plot will either reset or stop, depending on the configuration.
- Last Analog Data: Displays the time since reset and the last analog value plotted.
- Plot Pointer: Moving the plot pointer on the display shows the current analog value and time for that point on the plot.
- Maximum: Records the maximum analog value and the time it was reached.
- Minimum: Records the minimum analog value and the time it was reached.
- Clear Min/Max: Clears the recorded minimum and maximum values. [!CLMM]
- Clear min/max on reset: Allows a reset to clear the minimum and maximum values. [!CMMR]



Display Control and Zoom

- Moving the cursor on the plot will set the plot pointer time and value to the current position.
- Double-clicking the plot will shift display modes from yellow to white background and thin lines to thick lines for better printing (ALT-Print Screen to capture form to the clipboard) and projected display.
- Shift-Click (hold) and Drag allows you to specify an area of the plot to zoom.

BASIC Stamp Control and Configuration Commands

The majority of the plot configuration and controls may be set from within the BASIC Stamp program. To use these commands, simply debug from the BASIC Stamp. All commands must end with a CR (ASCII 13): DEBUG "!PLOT ON", CR.

Command	Description
!TITL message	Sets the title of the form to the message
!USRS message	Sets the User Status box to display the message
!BELL	Sounds the bell on the PC
!AMAX value	Sets the plot maximum analog value
!AMIN value	Sets the plot minimum analog value
!SPAN minValue, maxValue	Sets the plots analog maximum and minimum as above (also adds the range to the Range Drop-Down box).
!AMUL value	Sets the value to multiply incoming data by
!TMAX value	Sets the plot maximum time (seconds)
!TMIN value	Sets the plot minimum time (seconds)
!PNTS value	Sets the number of data points to collect
!PLOT ON/OFF	Enables/disables the plotting of data
!RSET	Resets the plot and all data
!CLRM	Clears the message list
!CLMM	Clears the min/max recorded values
!CMMR ON/OFF	Enables/Disables clearing of Min/Max recorded values on reset
!MAXS	Sets the plot to STOP when data points are full
!MAXR	Sets the plot to RESET when data points are full
!SHFT ON/OFF	Enables/disables the plot from shifting when recording data (may cause a loss of data accuracy if enabled)
!TSMP ON/OFF	Enables time stamping of list messages; messages and data saved to files
!SAVD ON/OFF	Enables saving of analog and digital data to files
!SAVM ON/OFF	Enables saving of messages to a file
!DELD	Deletes the saved data file
!DELM	Deletes the saved message file

Additional Application Notes

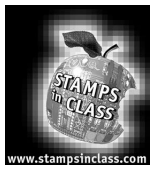
The amount of the plot that is used is dependent on the number of data points and the rate at which they are transmitted. For example, if you wish 60 seconds of data to fill the screen and are transmitting from the BASIC Stamp at a maximum rate of 100 msec (Pause 100 + processing time): 60/.1 = 600 data points.

The application needs a minimum regular pause in the reception of data for complete processing. A pause of 10 msec is typically sufficient for a fairly fast computer. If the application senses it cannot keep up with incoming data, a message box will appear and the application will disconnect. Some indications that the computer cannot keep up are: garbled data, no plotting, and the inability to affect any controls (locks up).

The greater the number of data points and the higher the current data point, the longer the plot will take to respond to plot shifts as it redraws. For faster, reliable configuration from the BASIC Stamp on initial power up or resetting, the following is recommended:

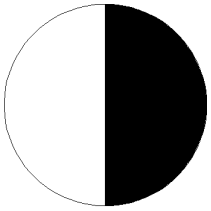
- Pause for 500msec at the start to allow StampPlot's buffer to clear.
- Perform a StampPlot RESET (!RSET) prior to making configuration changes to allow the data points to be cleared so redrawing is not performed.
- Reset (!RSET) at the end of the configuration resets the plot to time 0.

As with any application, the best way to learn it is to play. Have fun!

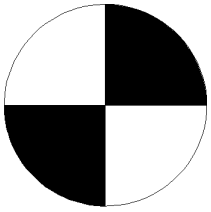


Appendix B: Fan Encoder Printouts

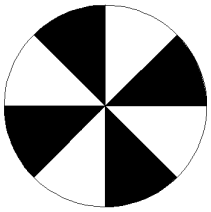
These printouts are full size, and should be an ideal fit for your fan used as a digital switch in Experiment #2.



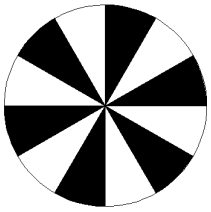
1 cycle/revolution



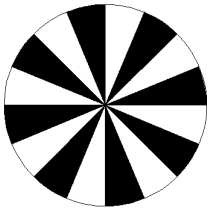
2 cycles/revolution



4 cycles/revolution



6 cycles/revolution



8 cycles/revolution

Appendix C: Potter Brumfield SSR Datasheet

Appendix C consists of the Potter Brumfield "Hockey Puck" Solid State Relay. Their datasheets may be downloaded from http://www.pandbrelays.com/.

PB

tyco / Electronics

SSRT series

"Hockey Puck"
Solid State Relay With
Snubberless Triac Output

File E29244
File E29244 UL Recognized for Canada

Features

- Standard "hockey puck" package.
- Exposed ceramic base plate for reduced themal resistance.
- Floating terminal design.
- Low cost snubberless triac outputs.
- 10A & 25A rms versions.
- AC & DC input versions.
- 4000V rms isolation.

Engineering Data

Form: 1 Form A (SPST-NO).
Duty: Continuous.
Isolation: 4000V rms minimum, input - output.
Isolation Resistance: 10^{10} ohms @ 500VDC minimum.
Capacitance: 10 pf maximum (input to output).
Temperature Range:
Storage: –40°C to +120°C
Operating Temperature: –25°C to + 80°C
Case Material: Plastic, UL rated 94V-0.
Base Plate Material: Ceramic.
Case and Mounting: Refer to outline dimension.
Termination: Refer to outline dimension.
Approximate Weight: 3.5 oz. (98g).

Ordering Information

Sample Part Number ►	SSRT	-240	D	10

1. **Basic Series:** SSRT = "hockey puck" triac output solid state relay
2. **Line Voltage:** 240 = 24-240 VAC
3. **Input Type & Voltage:** A = 90-280 VAC/VDC linear
   D = 3-32 VDC constant current
4. **Maximum Switching Rating:** 10 = .05-10A rms, mounted to heatsink
   25 = .05-25A rms, mounted to heatsink

Stock Items – The following items are normally maintained in stock for immediate delivery.

SSRT-240A10 SSRT-240D10
SSRT-240A25 SSRT-240D25

Input Specifications

Parameter	AC/DC Input/AC Output	DC Input/AC Output
Control Voltage Range V_{IN}	90-280VAC/VDC	3-32VDC
Must Operate Voltage $V_{IN(OP)}$ (Max.)	90VAC/VDC	3VDC
Must Release Voltage $V_{IN(REL)}$ (Min.)	10VAC/VDC	1VDC
Input Current	15mA Max. @ 90VAC	15mA Max. @ 5VDC

PB tyco Electronics

Parameter	Conditions	Units	SSRT-240A10 & SSRT-240D10	SSRT-240A25 & SSRT-240D25
Load Voltage Range V_L		V rms	24-240	
Repetitive Blocking Voltage (Min.)		V peak	±600	
Load Current Range I_L*	Resistive	A rms	.05-10	.05-25
Single Cycle Surge Current (Min.)		A peak	100	200
Leakage Current (Off-State) (Max.)	f = 60 Hz. V_L = Nom. (120 or 240 V rms)	mA rms	0.5	
On-State Voltage Drop (Max.)	I_L = Max.	V peak	1.7	
Static dv/dt (Off-State) (Min.)		V/µs	200	
Thermal Resistance, Junction to Case ($R_{\theta J-C}$) (Max.)		° C/W	0.6	0.6
Turn-On Time (Max.)	f = 60 Hz.	ms	8.3	
Turn-Off Time (Max.)	f = 60 Hz.	ms	8.3	
I^2t Rating	t = 8.3 ms	A^2 Sec.	60	260
Load Power Factor Rating	I_L = Max.		0.5-1.0	

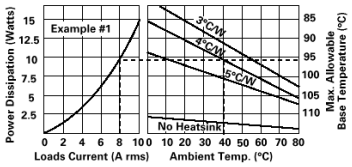
*See Derating Curves

Electrical Characteristics (Thermal Derating Curves)

How To Use These Curves

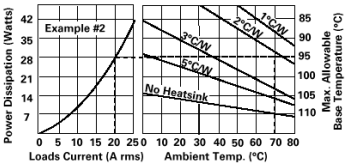
Knowing maximum load current and maximum ambient temperature, use derating curves to determine the minimum required heat sink and maximum allowable base plate temperature. On left hand power dissipation curve, locate the point corresponding to maximum load current. Extend a line to the right from that point to the intersection of vertical line on right hand chart corresponding to maximum ambient temperature. From heat sink curve, read directly or extrapolate required heat sink size. Extend the line farther to the right and read on the right hand scale the maximum allowable base plate temperature.

10A Triac Units



Example #1:
Given: IL 8A rms @ 40°C
Find: Heatsink required
Solution: From 10A curve
Heatsink = 4°C/W

25A Triac Units



Example #2:
Given: IL = 20A rms @ 70°C
Find: Required heatsink
Solution: Heatsink = 2°C/W

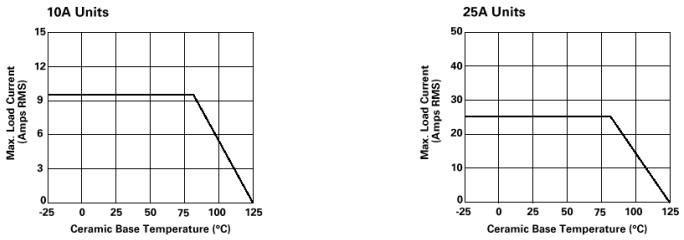
Heatsink Dimensions

- We recommend that solid State Relay Modules be mounted to a heatsink sufficient to maintain the module's base temperature at less than 85°C under worst case ambient temperature and load conditions.
- The heatsink mounting surface should be a smooth (30-40 micro-inch finish), flat (30-40 micro-inch flatness across mating area), un-painted surface which is clean and free of oxidation.
- An even coating of thermal compound (Dow Corning DC340 or equivalent) should be applied to both the heatsink and module mounting surfaces and spread to a uniform depth of .002" to eliminate all air pockets.
- The module should be mounted to the heatsink using two#10 screws. The mounting screws should be torqued to 10 inch-pounds by alternately tightening the screws one quarter turn at a time.

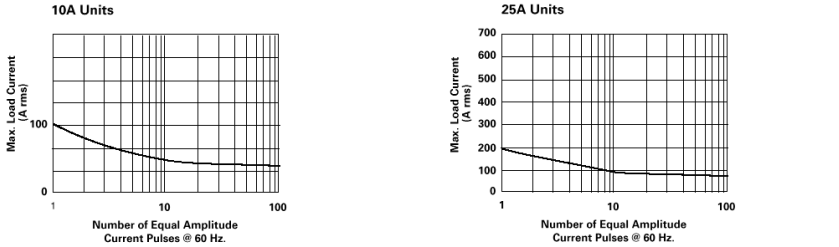
PB

tyco / Electronics

Load Current vs. Base Temperature

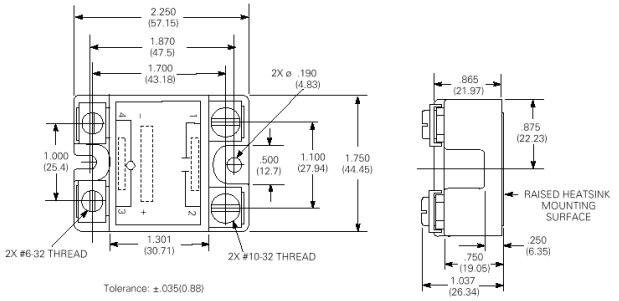


Allowable Peak Surge vs. Duration/Expected Lifetime



Operating Diagrams

Outline Dimensions



Tyco Electronics
700 Westpark Drive
Peachtree City, GA 30269-1498

Specifications and availability subject to change without notice.
13C7778 Printed in U.S.A. IH/2-00

Appendix D:
National Semiconductor
LM34 Datasheet

Appendix D consists of the National Semiconductor LM34 datasheet. This appendix includes the first five (5) pages of the 12-page datasheet. Should you wish to see more applications of the LM34 than are shown in this datasheet, the entire document may be downloaded from http://www.national.com/ds/LM/LM34.pdf.

National Semiconductor

July 1999

LM34
Precision Fahrenheit Temperature Sensors

General Description

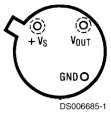
The LM34 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Fahrenheit temperature. The LM34 thus has an advantage over linear temperature sensors calibrated in degrees Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Fahrenheit scaling. The LM34 does not require any external calibration or trimming to provide typical accuracies of ±½˚F at room temperature and ±1½˚F over a full −50 to +300˚F temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM34's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies or with plus and minus supplies. As it draws only 75 µA from its supply, it has very low self-heating, less than 0.2˚F in still air. The LM34 is rated to operate over a −50˚ to +300˚F temperature range, while the LM34C is rated for a −40˚ to +230˚F range (0˚F with improved accuracy). The LM34 series is available packaged in hermetic TO-46 transistor packages, while the LM34C, LM34CA and LM34D are also available in the plastic TO-92 transistor package. The LM34D is also available in an 8-lead surface mount small outline package. The LM34 is a complement to the LM35 (Centigrade) temperature sensor.

Features

- Calibrated directly in degrees Fahrenheit
- Linear +10.0 mV/˚F scale factor
- 1.0˚F accuracy guaranteed (at +77˚F)
- Rated for full −50˚ to +300˚F range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 5 to 30 volts
- Less than 90 µA current drain
- Low self-heating, 0.18˚F in still air
- Nonlinearity only ±0.5˚F typical
- Low-impedance output, 0.4Ω for 1 mA load

Connection Diagrams

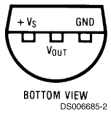
TO-46
Metal Can Package
(Note 1)



DS006685-1

Order Numbers LM34H, LM34AH, LM34CH, LM34CAH or LM34DH
See NS Package Number H03H

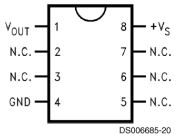
TO-92
Plastic Package



BOTTOM VIEW
DS006685-2

Order Number LM34CZ, LM34CAZ or LM34DZ
See NS Package Number Z03A

SO-8
Small Outline
Molded Package



DS006685-20

N.C. = No Connection

Top View
Order Number LM34DM
See NS Package Number M08A

Note 1: Case is connected to negative pin (GND).

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

 DS006685 www.national.com

Typical Applications

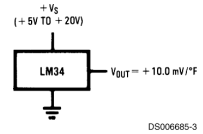


FIGURE 1. Basic Fahrenheit Temperature Sensor (+5° to +300°F)

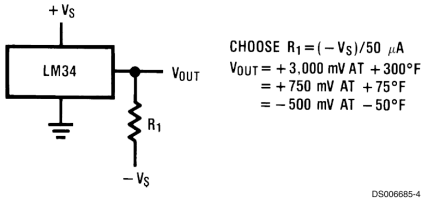


FIGURE 2. Full-Range Fahrenheit Temperature Sensor

Absolute Maximum Ratings (Note 11)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temperature,	
TO-46 Package	−76˚F to +356˚F
TO-92 Package	−76˚F to +300˚F
SO-8 Package	−65˚C to +150˚C
ESD Susceptibility (Note 12)	800V
Lead Temp.	
TO-46 Package (Soldering, 10 seconds)	+300˚C
TO-92 Package (Soldering, 10 seconds)	+260˚C
SO Package (Note 13)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
Specified Operating Temp. Range (Note 3)	
	T_{MIN} to T_{MAX}
LM34, LM34A	−50˚F to +300˚F
LM34C, LM34CA	−40˚F to +230˚F
LM34D	+32˚F to +212˚F

DC Electrical Characteristics (Notes 2, 7)

Parameter	Conditions	LM34A			LM34CA			Units (Max)
		Typical	Tested Limit (Note 5)	Design Limit (Note 6)	Typical	Tested Limit (Note 5)	Design Limit (Note 6)	
Accuracy (Note 8)	T_A = +77˚F	±0.4	±1.0		±0.4	±1.0		˚F
	T_A = 0˚F	±0.6			±0.6		±2.0	˚F
	$T_A = T_{MAX}$	±0.8	±2.0		±0.8	±2.0		˚F
	$T_A = T_{MIN}$	±0.8	±2.0		±0.8		±3.0	˚F
Nonlinearity (Note 9)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.35**		**±0.7**	**±0.30**		**±0.6**	˚F
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚F, min mV/˚F, max
Load Regulation (Note 4)	T_A = +77˚F	±0.4	±1.0		±0.4	±1.0		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
	$0 \le I_L \le 1$ mA							
Line Regulation (Note 4)	T_A = +77˚F	±0.01	±0.05		±0.01	±0.05		mV/V
	$5V \le V_S \le 30V$	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 10)	V_S = +5V, +77˚F	75	90		75	90		µA
	V_S = +5V	**131**		**160**	**116**		**139**	µA
	V_S = +30V, +77˚F	76	92		76	92		µA
	V_S = +30V	**132**		**163**	**117**		**142**	µA
Change of Quiescent Current (Note 4)	$4V \le V_S \le 30V$, +77˚F	+0.5	2.0		0.5	2.0		µA
	$5V \le V_S \le 30V$	**+1.0**		**3.0**	**1.0**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.30**		**+0.5**	**+0.30**		**+0.5**	µA/˚F
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L = 0	+3.0		+5.0	+3.0		+5.0	˚F
Long-Term Stability	$T_j = T_{MAX}$ for 1000 hours	±0.16			±0.16			˚F

Note 2: Unless otherwise noted, these specifications apply: −50˚F ≤ T_j ≤ + 300˚F for the LM34 and LM34A; −40˚F ≤ T_j ≤ +230˚F for the LM34C and LM34CA; and +32˚F ≤ T_j ≤ + 212˚F for the LM34D. V_S = +5 Vdc and I_{LOAD} = 50 µA in the circuit of *Figure 2*; +6 Vdc for LM34 and LM34A for 230˚F ≤ T_j ≤ 300˚F. These specifications also apply from +5˚F to T_{MAX} in the circuit of *Figure 1*.

Note 3: Thermal resistance of the TO-46 package is 720˚F/W junction to ambient and 43˚F/W junction to case. Thermal resistance of the TO-92 package is 324˚F/W junction to ambient. Thermal resistance of the small outline molded package is 400˚F/W junction to ambient. For additional thermal resistance information see table in the Typical Applications section.

Note 4: Regulation is measured at constant junction temperature using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 5: Tested limits are guaranteed and 100% tested in production.

Note 6: Design limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 7: Specification in **BOLDFACE TYPE** apply over the full rated temperature range.

DC Electrical Characteristics (Notes 2, 7) (Continued)

Note 8: Accuracy is defined as the error between the output voltage and 10 mV/°F times the device's case temperature at specified conditions of voltage, current, and temperature (expressed in °F).

Note 9: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line over the device's rated temperature range.

Note 10: Quiescent current is defined in the circuit of *Figure 1*.

Note 11: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions (Note 2).

Note 12: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

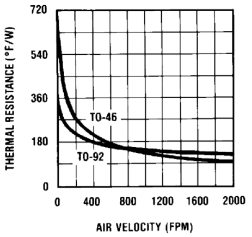
Note 13: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

DC Electrical Characteristics (Notes 2, 7)

Parameter	Conditions	LM34			LM34C, LM34D			Units (Max)
		Typical	Tested Limit (Note 5)	Design Limit (Note 6)	Typical	Tested Limit (Note 5)	Design Limit (Note 6)	
Accuracy, LM34, LM34C (Note 8)	T_A = +77°F	±0.8	±2.0		±0.8	±2.0		°F
	T_A = 0°F	±1.0			±1.0		±3.0	°F
	T_A = T_{MAX}	±1.6	±3.0		±1.6		±3.0	°F
	T_A = T_{MIN}	±1.6		±3.0	±1.6		±4.0	°F
Accuracy, LM34D (Note 8)	T_A = +77°F				±1.2	±3.0		°F
	T_A = T_{MAX}				±1.8		±4.0	°F
	T_A = T_{MIN}				±1.8		±4.0	°F
Nonlinearity (Note 9)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.6**		**±1.0**	**±0.4**		**±1.0**	°F
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°F, min mV/°F, max
Load Regulation (Note 4)	T_A = +77°F	±0.4	±2.5		±0.4	±2.5		mV/mA
	$T_{MIN} \le T_A \le$ +150°F	**±0.5**		**±6.0**	**±0.5**		**±6.0**	mV/mA
	$0 \le I_L \le 1$ mA							
Line Regulation (Note 4)	T_A = +77°F	±0.01	±0.1		±0.01	±0.1		mV/V
	$5V \le V_S \le 30V$	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 10)	V_S = +5V, +77°F	75	100		75	100		µA
	V_S = +5V	**131**		**176**	**116**		**154**	µA
	V_S = +30V, +77°F	76	103		76	103		µA
	V_S = +30V	**132**		**181**	**117**		**159**	µA
Change of Quiescent Current (Note 4)	$4V \le V_S \le 30V$, +77°F	+0.5	3.0		0.5	3.0		µA
	$5V \le V_S \le 30V$	**+1.0**		**5.0**	**1.0**		**5.0**	µA
Temperature Coefficient of Quiescent Current		**+0.30**		**+0.7**	**+0.30**		**+0.7**	µA/°F
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L = 0	+3.0		+5.0	+3.0		+5.0	°F
Long-Term Stability	T_j = T_{MAX}, for 1000 hours	±0.16			±0.16			°F

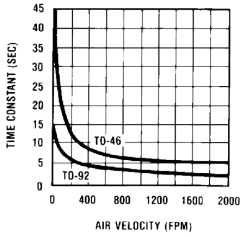
Typical Performance Characteristics

Thermal Resistance Junction to Air



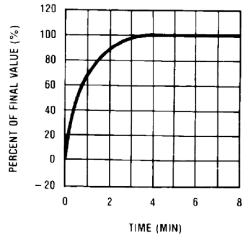
DS006685-22

Thermal Time Constant



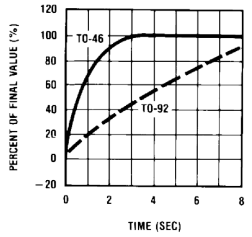
DS006685-23

Thermal Response in Still Air



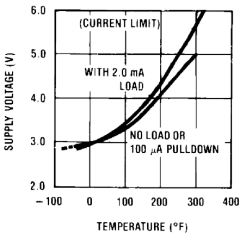
DS006685-24

Thermal Response in Stirred Oil Bath



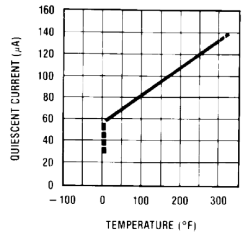
DS006685-25

Minimum Supply Voltage vs. Temperature



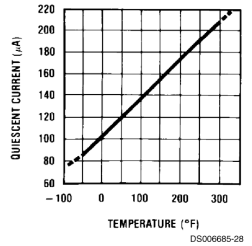
DS006685-26

Quiescent Current vs. Temperature (In Circuit of *Figure 1*)



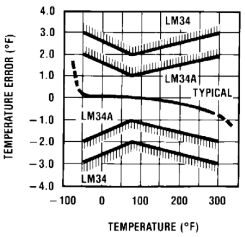
DS006685-27

Quiescent Current vs. Temperature (In Circuit of *Figure 2*; $-V_S$ = −5V, R1 = 100k)



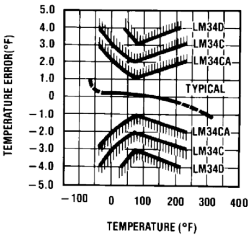
DS006685-28

Accuracy vs. Temperature (Guaranteed)



DS006685-29

Accuracy vs. Temperature (Guaranteed)



DS006685-30

Appendix E:
National Semiconductor
LM358 Datasheet

Appendix D consists of the National Semiconductor LM358 datasheet. This appendix includes the first five (5) pages of the 23-page datasheet. Should you wish to see more applications of the LM358 than are shown in this datasheet, the entire document may be downloaded from http://www.national.com/ds/LM/LM158.pdf.

National Semiconductor

January 2000

LM158/LM258/LM358/LM2904
Low Power Dual Operational Amplifiers

General Description

The LM158 series consists of two independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, dc gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM158 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

The LM358 is also available in a chip sized package (8-Bump micro SMD) using National's micro SMD package technology.

Unique Characteristics

- In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage.
- The unity gain cross frequency is temperature compensated.
- The input bias current is also temperature compensated.

Advantages

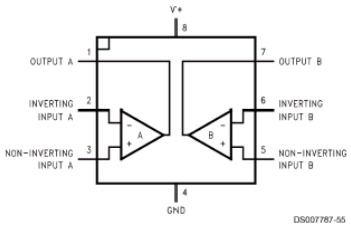
- Two internally compensated op amps
- Eliminates need for dual supplies
- Allows direct sensing near GND and V_{OUT} also goes to GND
- Compatible with all forms of logic
- Power drain suitable for battery operation
- Pin-out same as LM1558/LM1458 dual op amp

Features

- Available in 8-Bump micro SMD chip sized package, (See AN-1112)
- Internally frequency compensated for unity gain
- Large dc voltage gain: 100 dB
- Wide bandwidth (unity gain): 1 MHz (temperature compensated)
- Wide power supply range:
  — Single supply: 3V to 32V
  — or dual supplies: ±1.5V to ±16V
- Very low supply current drain (500 µA) — essentially independent of supply voltage
- Low input offset voltage: 2 mV
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Large output voltage swing: 0V to V+ − 1.5V

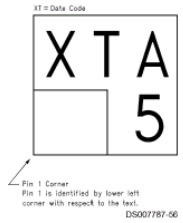
Connection Diagrams

8-Bump micro SMD



Top View
(Bump Side Down)

micro SMD Marking Orientation

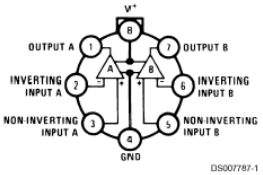


Bumps are numbered counter-clockwise.

Top View

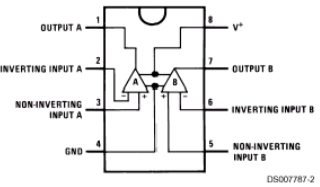
Connection Diagrams (Continued)

Metal Can Package



Top View

DIP/SO Package



Top View

Ordering Information

Package	Temperature Range				NSC Drawing
	−55˚C to 125˚C	−25˚C to 85˚C	0˚C to 70˚C	−40˚C to 85˚C	
SO-8			LM358AM LM358M	LM2904M	M08A
8-Pin Molded DIP			LM358AN LM358N	LM2904N	N08E
8-Pin Ceramic DIP	LM158AJ/883(Note 1) LM158J/883(Note 1) LM158J LM158AJLQML(Note 2) LM158AJQMLV(Note 2)				J08A
TO-5, 8-Pin Metal Can	LM158AH/883(Note 1) LM158H/883(Note 1) LM158AH LM158H LM158AHLQML(Note 2) LM158AHLQMLV(Note 2)	LM258H	LM358H		H08C
8-Bump micro SMD			LM358BP LM358BPX		BPA08AAA

Note 1: LM158 is available per SMD #5962-8771001
LM158A is available per SMD #5962-8771002
Note 2: See STD Mil DWG 5962L87710 for Radiation Tolerant Devices

Absolute Maximum Ratings (Note 11)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

	LM158/LM258/LM358 LM158A/LM258A/LM358A	LM2904
Supply Voltage, V^+	32V	26V
Differential Input Voltage	32V	26V
Input Voltage	−0.3V to +32V	−0.3V to +26V
Power Dissipation (Note 3)		
Molded DIP	830 mW	830 mW
Metal Can	550 mW	
Small Outline Package (M)	530 mW	530 mW
micro SMD	435mW	
Output Short-Circuit to GND		
(One Amplifier) (Note 4)		
$V^+ \leq 15V$ and $T_A = 25^\circ C$	Continuous	Continuous
Input Current ($V_{IN} < -0.3V$) (Note 5)	50 mA	50 mA
Operating Temperature Range		
LM358	0°C to +70°C	−40°C to +85°C
LM258	−25°C to +85°C	
LM158	−55°C to +125°C	
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C
Lead Temperature, DIP		
(Soldering, 10 seconds)	260°C	260°C
Lead Temperature, Metal Can		
(Soldering, 10 seconds)	300°C	300°C
Soldering Information		
Dual-In-Line Package		
Soldering (10 seconds)	260°C	260°C
Small Outline Package		
Vapor Phase (60 seconds)	215°C	215°C
Infrared (15 seconds)	220°C	220°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.		
ESD Tolerance (Note 12)	250V	250V

Electrical Characteristics

$V^+ = +5.0V$, unless otherwise stated

Parameter	Conditions	LM158A			LM358A			LM158/LM258			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	(Note 7), $T_A = 25^\circ C$		1	2		2	3		2	5	mV
Input Bias Current	$I_{IN(+)}$ or $I_{IN(-)}$, $T_A = 25^\circ C$, $V_{CM} = 0V$, (Note 8)		20	50		45	100		45	150	nA
Input Offset Current	$I_{IN(+)} - I_{IN(-)}$, $V_{CM} = 0V$, $T_A = 25^\circ C$		2	10		5	30		3	30	nA
Input Common-Mode Voltage Range	$V^+ = 30V$, (Note 9) (LM2904, $V^+ = 26V$), $T_A = 25^\circ C$	0		$V^+-1.5$	0		$V^+-1.5$	0		$V^+-1.5$	V
Supply Current	Over Full Temperature Range $R_L = \infty$ on All Op Amps										
	$V^+ = 30V$ (LM2904 $V^+ = 26V$)		1	2		1	2		1	2	mA
	$V^+ = 5V$		0.5	1.2		0.5	1.2		0.5	1.2	mA

Electrical Characteristics

V^+ = +5.0V, unless otherwise stated

Parameter	Conditions	LM358			LM2904			Units
		Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	(Note 7) , T_A = 25˚C		2	7		2	7	mV
Input Bias Current	$I_{IN(+)}$ or $I_{IN(-)}$, T_A = 25˚C, V_{CM} = 0V, (Note 8)		45	250		45	250	nA
Input Offset Current	$I_{IN(+)} - I_{IN(-)}$, V_{CM} = 0V, T_A = 25˚C		5	50		5	50	nA
Input Common-Mode Voltage Range	V^+ = 30V, (Note 9) (LM2904, V^+ = 26V), T_A = 25˚C	0		$V^+-1.5$	0		$V^+-1.5$	V
Supply Current	Over Full Temperature Range $R_L = \infty$ on All Op Amps							
	V^+ = 30V (LM2904 V^+ = 26V)		1	2		1	2	mA
	V^+ = 5V		0.5	1.2		0.5	1.2	mA

Electrical Characteristics

V^+ = +5.0V, (Note 6), unless otherwise stated

Parameter		Conditions	LM158A			LM358A			LM158/LM258			Units
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Large Signal Voltage Gain		V^+ = 15V, T_A = 25˚C, $R_L \geq 2$ kΩ, (For V_O = 1V to 11V)	50	100		25	100		50	100		V/mV
Common-Mode Rejection Ratio		T_A = 25˚C, V_{CM} = 0V to $V^+-1.5$V	70	85		65	85		70	85		dB
Power Supply Rejection Ratio		V^+ = 5V to 30V (LM2904, V^+ = 5V to 26V), T_A = 25˚C	65	100		65	100		65	100		dB
Amplifier-to-Amplifier Coupling		f = 1 kHz to 20 kHz, T_A = 25˚C (Input Referred), (Note 10)		−120			−120			−120		dB
Output Current	Source	V_{IN}^+ = 1V, V_{IN}^- = 0V, V^+ = 15V, V_O = 2V, T_A = 25˚C	20	40		20	40		20	40		mA
	Sink	V_{IN}^- = 1V, V_{IN}^+ = 0V, V^+ = 15V, T_A = 25˚C, V_O = 2V	10	20		10	20		10	20		mA
		V_{IN}^- = 1V, V_{IN}^+ = 0V, T_A = 25˚C, V_O = 200 mV, V^+ = 15V	12	50		12	50		12	50		µA
Short Circuit to Ground		T_A = 25˚C, (Note 4), V^+ = 15V		40	60		40	60		40	60	mA
Input Offset Voltage		(Note 7)			4			5			7	mV
Input Offset Voltage Drift		R_S = 0Ω		7	15		7	20		7		µV/˚C
Input Offset Current		$I_{IN(+)} - I_{IN(-)}$			30			75			100	nA
Input Offset Current Drift		R_S = 0Ω		10	200		10	300		10		pA/˚C
Input Bias Current		$I_{IN(+)}$ or $I_{IN(-)}$		40	100		40	200		40	300	nA
Input Common-Mode Voltage Range		V^+ = 30 V, (Note 9) (LM2904, V^+ = 26V)	0		V^+-2	0		V^+-2	0		V^+-2	V

Electrical Characteristics (Continued)

V⁺ = +5.0V, (Note 6), unless otherwise stated

Parameter		Conditions		LM158A			LM358A			LM158/LM258			Units
				Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Large Signal Voltage Gain		V⁺ = +15V (V_O = 1V to 11V) $R_L \geq 2\ k\Omega$		25			15			25			V/mV
Output Voltage	V_{OH}	V⁺ = +30V	R_L = 2 kΩ	26			26			26			V
		(LM2904, V⁺ = 26V)	R_L = 10 kΩ	27	28		27	28		27	28		V
Swing	V_{OL}	V⁺ = 5V, R_L = 10 kΩ			5	20		5	20		5	20	mV
Output Current	Source	V_{IN}^+ = +1V, V_{IN}^- = 0V, V⁺ = 15V, V_O = 2V		10	20		10	20		10	20		mA
	Sink	V_{IN}^- = +1V, V_{IN}^+ = 0V, V⁺ = 15V, V_O = 2V		10	15		5	8		5	8		mA

Electrical Characteristics

V⁺ = +5.0V, (Note 6), unless otherwise stated

Parameter		Conditions	LM358			LM2904			Units
			Min	Typ	Max	Min	Typ	Max	
Large Signal Voltage Gain		V⁺ = 15V, T_A = 25˚C, $R_L \geq 2\ k\Omega$, (For V_O = 1V to 11V)	25	100		25	100		V/mV
Common-Mode Rejection Ratio		T_A = 25˚C, V_{CM} = 0V to V⁺−1.5V	65	85		50	70		dB
Power Supply Rejection Ratio		V⁺ = 5V to 30V (LM2904, V⁺ = 5V to 26V), T_A = 25˚C	65	100		50	100		dB
Amplifier-to-Amplifier Coupling		f = 1 kHz to 20 kHz, T_A = 25˚C (Input Referred), (Note 10)		−120			−120		dB
Output Current	Source	V_{IN}^+ = 1V, V_{IN}^- = 0V, V⁺ = 15V, V_O = 2V, T_A = 25˚C	20	40		20	40		mA
	Sink	V_{IN}^- = 1V, V_{IN}^+ = 0V V⁺ = 15V, T_A = 25˚C, V_O = 2V	10	20		10	20		mA
		V_{IN}^- = 1V, V_{IN}^+ = 0V T_A = 25˚C, V_O = 200 mV, V⁺ = 15V	12	50		12	50		µA
Short Circuit to Ground		T_A = 25˚C, (Note 4), V⁺ = 15V		40	60		40	60	mA
Input Offset Voltage		(Note 7)			9			10	mV
Input Offset Voltage Drift		R_S = 0Ω		7			7		µV/˚C
Input Offset Current		$I_{IN(+)} - I_{IN(-)}$			150		45	200	nA
Input Offset Current Drift		R_S = 0Ω		10			10		pA/˚C
Input Bias Current		$I_{IN(+)}$ or $I_{IN(-)}$		40	500		40	500	nA
Input Common-Mode Voltage Range		V⁺ = 30 V, (Note 9) (LM2904, V⁺ = 26V)	0		V⁺−2	0		V⁺ −2	V

Appendix F: Parts Listing and Sources

All components (next page) used in the Industrial Control experiments are readily available from common electronic suppliers. Customers who would like to purchase a complete kit may also do so through Parallax. To use this curriculum you need three items: (1) a BASIC Stamp II module (available alone, or in the Board of Education - Full Kit); (2) a Board of Education (available alone or in a Board of Education Full Kit); and 3) the Industrial Control Parts Kit.

Board of Education Kits

The BASIC Stamp II (BS2-IC) is available separately or in the Board of Education Full Kit. If you already have a BS2-IC module, then purchase the Board of Education Kit. Individual pieces may also be ordered using the Parallax stock codes shown below.

Board of Education – Full Kit (#28102)

Parallax Code#	Description	Quantity
28150	Board of Education	1
800-00016	Pluggable wires	10
BS2-IC	BASIC Stamp II module	1
750-00008	300 mA 9 VDC power supply	1
800-00003	Serial cable	1

Board of Education Kit (#28150)

Parallax Code#	Description	Quantity
28102	Board of Education and pluggable wires	1
BS2-IC	Pluggable wires	6

This printed documentation is very useful for additional background information:

BASIC Stamp Documentation

Parallax Code#	Description	Internet Availability?
27919	BASIC Stamp Manual Version 1.9	http://www.stampsinclass.com
27341	Industrial Control Text	http://www.stampsinclass.com

The Industrial Control experiments require the Industrial Control Parts Kit (#27340)

Similar to all Stamps in Class curriculum, you need a Board of Education with BASIC Stamp and the Parts Kit. The contents of the Industrial Control Parts Kit is listed below, broken down by experiment. Replacement parts in the kit may be ordered from http://www.stampsinclass.com.

Industrial Control Parts Kit (#27340)

Stock#	Description	#1	#2	#3	#4	#5	#6	Total/Kit
150-01020	1K ¼ watt resistor		2					2
150-01030	10K ¼ watt resistor	1	2					2
150-02210	220 ohm ¼ watt resistor	1	2					4
150-04710	470 ohm ¼ watt resistor			2	2	2	2	4
152-01031	10K ¼ watt multi-turn pot	1	1	2	2	2	2	2
201-01050	1uF capacitor			1	1	1	1	1
201-01060	10uF 10V capacitor				1	1	1	1
350-00001	LED green	1	1					1
350-00006	LED red		1					1
350-00007	LED yellow		1					1
350-00017	IR Led w/ shrink tube		1					1
350-00018	Infrared Phototransistor		1					1
400-00002	Pushbutton	1	2					2
500-00001	2N3904 Transistor			1		1	1	2
602-00015	LM358 Dual Op-Amp				1	1	1	1
700-00039	35 mm Black Film Canister		1	1	1	1	1	1
700-00040	12 VDC Brushless Fan		1	1	1	1	1	1
800-00027	LM34 Temperature Probe				1	1	1	1
800-00028	47 Ohm Resistor Heater			1	1	1	1	1
ADC0831	ADC 0831 8-bit A/D converter					1	1	1