Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Process Control

Student Guide

VERSION 1.0

WARRANTY

Parallax Inc. warrants its products against defects in materials and workmanship for a period of 90 days from receipt of product. If you discover a defect, Parallax Inc. will, at its option, repair or replace the merchandise, or refund the purchase price. Before returning the product to Parallax, call for a Return Merchandise Authorization (RMA) number. Write the RMA number on the outside of the box used to return the merchandise to Parallax. Please enclose the following along with the returned merchandise: your name, telephone number, shipping address, and a description of the problem. Parallax 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 Inc. will refund the purchase price of the product, excluding shipping/handling costs. This guarantee is void if the product has been altered or damaged. See the Warranty section above for instructions on returning a product to Parallax.

COPYRIGHTS AND TRADEMARKS

This text is available in printed format from Parallax Inc. Because we print the text in volume, the consumer price is often less than typical retail duplication charges.

ISBN 1-928982-36-0

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, or any costs of recovering, reprogramming, or reproducing any data stored in or used with Parallax products. Parallax Inc. 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 DISCUSSION LISTS

We maintain active web-based discussion forums for people interested in Parallax products. These lists are accessible from www.parallax.com via the Support → Discussion Forums menu. These are the forums that we operate from our web site:

- BASIC Stamp – This list is widely utilized by engineers, hobbyists and students who share their BASIC Stamp projects and ask questions.
- Stamps In Class® – Created for educators and students, subscribers discuss the use of the Stamps in Class educational products in their courses. The list provides an opportunity for both students and educators to ask questions and get answers.
- Parallax Educators –Exclusively for educators and those who contribute to the development of Stamps in Class. Parallax created this group to obtain feedback on our educational products and to provide a forum for educators to develop and obtain Teacher's Guides and other resources.
- Translators – The purpose of this private list is to provide a conduit between Parallax and those who translate our documentation to languages other than English. Parallax provides editable Word documents to our translating partners and attempts to time the translations to coordinate with our publications. To join, email aalvarez@parallax.com.
- Robotics – Designed for Parallax robots, this forum is intended to be an open dialogue for robotics enthusiasts. Topics include assembly, source code, expansion, and manual updates. The Boe-Bot®, Toddler®, SumoBot®, HexCrawler and QuadCrawler robots are discussed here.
- SX Microcontrollers and SX-Key – Discussion of programming the SX microcontroller with Parallax assembly language SX – Key® tools and 3rd party BASIC and C compilers.
- Javelin Stamp – Discussion of application and design using the Javelin Stamp, a Parallax module that is programmed using a subset of Sun Microsystems' Java® programming language.
- ParallaxEFX – For animators, theatre prop builders, and those who create Halloween and other holiday displays using the ParallaxEFX product line.
- Propeller Chip – Forum for those using the Parallax Propeller™ chip.

ERRATA

While great effort is made to assure the accuracy of our texts, errors may still exist. If you find an error, please let us know by sending an email to editor@parallax.com. We continually strive to improve all of our educational materials and documentation, and frequently revise our texts. Occasionally, an errata sheet with a list of known errors and corrections for a given text will be posted to our web site, www.parallax.com. Please check the individual product page's free downloads for an errata file.

Table of Contents

Preface iii
Educator Resources iv
The Stamps In Class Educational Series v
Foreign Translations vi
Special Contributors vi

Chapter 1: Process Control and Flowcharts 1
What is Process Control? 1
Activity #1: Flowcharts for Representing Processes 3
Activity #2: Sequential Flow and Code 6
Activity #3: Flow and Coding with Conditional Branches 12
Activity #4: Predefined Processes with Subroutines 16
Activity #5: Conditional Looping 20
Conclusion 24
Solutions to Chapter 1 Challenges 25

Chapter 2: PC Based Monitoring and Control 29
Activity #1: Using StampPlot for Monitoring and Control 29
Activity #2: StampPlot Interactive Control 35
Activity #3: StampPlot Toolbar Controls 37
Activity #4: Specialized Interface Controls 40
Conclusion 43
Solutions to Chapter 2 Challenges 43

Chapter 3: Digital Input Conditioning 45
Activity #1: Measuring the Threshold Voltage 45
Activity #2: Night-Light Process 51
Activity #3: Uncommited Inputs and Conditioning Switches 53
Activity #4: The Transistor as a Switch 57
Activity #5: Effects of Resistor Sizing 66
Activity #6: Switching Configuration Comparisons 70
Activity #7: Typical Industrial Switches 73
Conclusion 78
Solutions to Chapter 3 Challenges 79

Chapter 4: Sequential Processes and Optical Switches 83
Activity #1: Connecting and Testing the Opto-Reflective Switch 84
Activity #2: Batch or Sequential Process Control 91
Activity #3: Production Logs 101
Activity #4: Box Conveyor Belt – Counting and Edge Detection 104
Activity #5: Input Bounce and Spurious Signals 111
Activity #6: Tachometer – High-Speed Counting 114

Activity #7: Increasing Sensor Response 127
Conclusion 130
Solutions to Chapter 4 Challenges 130

Chapter 5: High Current Drive and PWM Control 135
Activity #1: DC Fan On-Off Control 135
Activity #2: Pulse Width Modulation 141
Activity #3: PWM Filtering 152
Activity #4: Op-Amp Buffer and Active-Filters 159
Activity #5: Op-Amp Non-Inverting Amplifier 162
Activity #6: Driving the Fan with the Op-Amp 166
Additional Devices of Interest 170
Conclusion 177
Solutions to Chapter 5 Challenges 177

Chapter 6: Open Loop Continuous Process Control 181
Activity #1: Testing the LM34 182
Activity #2: Signal Conditioning 186
Activity #3: Manual Control of Incubator 202
Activity #4: Open Loop PWM Control 211
Conclusion 218
Solutions to Chapter 6 Challenges 219

Chapter 7: Closed Loop Process Control 221
Activity #1: On-Off Control 223
Activity #2: Differential Gap Control 231
Conclusion 239
Solutions to Chapter 7 Challenges 240

Chapter 8: Proportional-Integral-Derivative Control 245
Activity #1: Bias Drive 256
Activity #2: Bias and System Response 274
Activity #3: Proportional Control at Bias Temperature 282
Activity #4: Proportional Control Not at Bias Temperature 288
Activity #5: Proportional-Integral Control 290
Activity #6: Proportional-Derivative Control 294
Conclusion 303
Solutions to Chapter 8 Challenges 305

Appendix A: Cut-Outs 309

Appendix B: Parts Listing 311

Index 313

Preface

Industrial process control (PC) 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. In this respect, you may hear that microcontrollers are embedded as the "brains" in much of our manufacturing equipment and consumer electronic devices. But in reality, the real "brains" of the system is the process control technician.

Although embedded process control centers around the microcontroller, it is only one piece of the total control system. The process control technician must be part control engineer, electronics technician, and computer programmer. This Process Control text uses its experiment-based chapters to build a good foundation from which to analyze and understand the many facets of embedded control technology.

The text builds this foundation through hands-on laboratory circuits and experiments that reinforce short, relative discussions of control theory. You will experiment with event-based and time-based sequential control as well as various open-loop and closed-loop continuous control modes. You will understand the characteristics of these modes of control and how they lend themselves to different types of control applications. Converting the control scenario and the mode of control chosen into a program flowchart is the first major step toward bringing automated intelligence into the system. Clear, well-commented PBASIC programs demonstrate how the Basic Stamp can be programmed to provide the control action.

An exciting and powerful software application comes with this text to help you visually understand the dynamics of a system as well as allow you to develop computer-based monitoring and control of your Basic Stamp. StampPlot's multiple-channel graphing feature is used throughout the text to allow you to monitor and compare input and output relationships to better understand the dynamics of the control system. You will also see how virtual controls, such as gauges, pushbuttons, sliders, textboxes, etc. can be used to build interactive visual interfaces for supervisory control and data acquisition of your Basic Stamp projects.

The hardware needed in the experiments to simulate the process has been kept to a bare minimum. While the microcontroller is programmed to be 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. The experiments include information on proper I/O signal conditioning. The process control technician must have a good understanding of the electronics required to get proper input voltages into the microcontroller from switches, contacts, and sensors as well as understand how to interface it to high-power output elements through the use of relays and power semiconductors.

After working with the sample control scenarios in the book, students quickly find themselves considering the countless automated control applications all around them. The most exciting aspect of this Process Control text is its ability to give you the tools to apply control theory, flowchart diagramming, and input/output signal conditioning to your own real-world applications.

Martin Hebel and Will Devenport
Southern Illinois University Carbondale
Electronic Systems Technologies
http://www.siu.edu/~isat/est
-- and -- SelmaWare Solutions
http://www.selmaware.com

Editor's Note: *Process Control* is a newly written text covering similar subject matter to Industrial *Control*, which it now replaces in the Stamps In Class educational series. *Process Control* is an advanced book, and we strongly recommend that students first learn the electronics and PBASIC programming concepts introduced in *What's a Microcontroller?* – the gateway to the Stamps in Class series.

EDUCATOR RESOURCES

Process Control has a supplemental set of exercises and solutions in an editable Word document that are made available only to teachers. These materials and other Stamps in Class resources can be obtained by joining the free, private Parallax Educators forum.

Both students and teachers are invited to join the public Parallax Stamps in Class forum, where they can discuss their experiences using Process Control or any other Stamps in Class text in the classroom. Students are encouraged to come here for assistance with working through the projects in the text, and teachers are encouraged to offer support. Parallax staff moderate and participate in this forum.

To join the Stamps in Class forum, go to forums.parallax.com. After joining Stamps in Class, educators may email stampsinclass@parallax.com for instructions to join the Parallax Educators forum. Proof of status as an educator will be required.

THE STAMPS IN CLASS EDUCATIONAL SERIES

The Stamps In Class series of texts and kits provides affordable resources for electronics and engineering education. All of the books listed are available for free download from www.parallax.com. The versions cited below were current at the time of this printing. Please check our web sites www.parallax.com or www.stampsinclass.com for the latest revisions; we continually strive to improve our educational program.

Stamps in Class Student Guides:

What's a Microcontroller? is the recommended entry level text to the Stamps In Class educational series. Some students instead start with *Robotics with the Boe-Bot*, also designed for beginners.

"*What's a Microcontroller?*", Student Guide, Version 2.2, Parallax Inc., 2004
"*Robotics with the Boe-Bot*", Student Guide, Version 2.2, Parallax Inc., 2004

You may continue on with other Educational Project topics, or you may wish to explore our other Robotics Kits.

Educational Project Kits:

The following texts and kits provides a variety of activities that are useful to hobbyists, inventors and product designers interested in trying a wide range of projects.

"*Process Control*", Student Guide, Version 2.0, Parallax Inc., 2006
"*Applied Sensors*", Student Guide, Version 1.3, Parallax Inc., 2003
"*Basic Analog and Digital*", Student Guide, Version 1.3, Parallax Inc., 2004
"*Elements of Digital Logic*", Student Guide, Version 1.0, Parallax Inc., 2003
"*Experiments with Renewable Energy*", Student Guide, Version 1.0, Parallax Inc., 2004
"*Understanding Signals*", Student Guide, Version 1.0, Parallax Inc., 2003

Robotics Kits:

To gain experience with robotics, consider continuing with the following Stamps in Class student guides, each of which has a corresponding robot kit:

"*IR Remote for the Boe-Bot*", Student Guide, Version 1.0, Parallax Inc., 2004
"*Applied Robotics with the SumoBot*", Student Guide, Version 1.0, Parallax Inc., 2005
"*Advanced Robotics: with the Toddler*", Student Guide, Version 1.2, Parallax Inc., 2003

Reference

This book is an essential reference for all Stamps in Class Student Guides. It is packed with information on the BASIC Stamp series of microcontroller modules, our BASIC Stamp Editor, and our PBASIC programming languages.

"*BASIC Stamp Manual*", Version 2.2, Parallax Inc., 2005

FOREIGN TRANSLATIONS

Parallax educational texts may be translated to other languages with our permission (e-mail stampsinclass@parallax.com). If you plan on doing any translations please contact us so we can provide the correctly-formatted MS Word documents, images, etc. We also maintain a private discussion group for Parallax translators which you may join. This will ensure that you are kept current on our frequent text revisions.

SPECIAL CONTRIBUTORS

The authors would also like to thank the 2004 and 2005 EST 212 classes at Southern Illinois University Carbondale for testing much of the text during its development. Also, thanks to Barry Shahian and Clark Radcliffe for their feedback and suggestions.

Parallax Inc. would like to recognize their Education Team members: Project Manager Aristides Alvarez, Technical Illustrator Rich Allred, Graphic Designer Jen Jacobs, and Technical Editor Stephanie Lindsay. Special thanks also go to Andrew Lindsay, Chris Savage, and Kris Magri for their insightful consulting and review, and, as always, to Ken Gracey, the founder of Parallax Inc.'s Stamps in Class educational program.

Chapter 1: Process Control and Flowcharts

BEFORE YOU START

To perform the experiments in this text, you will need to have your Board of Education connected to your computer, the BASIC Stamp Editor software installed, and to have verified the communication between your computer and your BASIC Stamp. For detailed instructions, see *What's a Microcontroller?* - a free download from www.parallax.com. You will also need the parts contained in the Process Control Parts Kit. For a full listing of system, software, and hardware requirements, see Appendix B.

WHAT IS PROCESS CONTROL?

Process control refers to the control of one or more system parameters, such as temperature, flow rate or position. While most systems are a continual process, such as maintaining a temperature, other processes may be a sequence of actions, for example, the assembly of a product.

Control systems can be very simple or very complex. Figure 1-1 is a block diagram of a simple continuous control system. For control of the process, an input (such as a setpoint control or switch) is required into the controller. Based on the input, the controller will drive an actuator to cause the desired effect on the process.

Examples of actuators are heaters for temperature, pumps for flow, and servos for positioning.

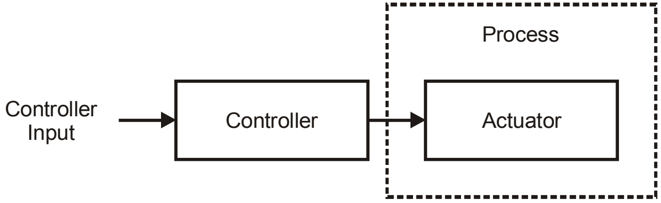


Figure 1-1
Simple Process Control Block Diagram

Consider the example of a common car heating system. The driver adjusts a temperature control to change the heat output of the vents. If the driver becomes too warm when weather conditions change, the temperature control must be adjusted to return to a comfortable temperature. This is a very simple system in that most automobiles do not monitor the cabin with temperature sensors to automatically control the heat output of the vents.

A more sophisticated system would have a sensor to monitor temperature and provide feedback to the controller. The controller would automatically adjust the actuator to regulate the controlled parameter - temperature. The controller would drive the heating system to maintain the temperature near the defined set point. An example of this is your home heating system.

Consider the difference between how the cabin temperature of the automobile is controlled versus the temperature in a home. In the automobile, the heat output is variable but has no sensors that directly affect the heat output and maintain temperature. In home heating a sensor is used to monitor the temperature, but the output of the heating system is not variable; it is either on or off and cycles to maintain temperature in a comfortable band. The controller itself may be very simple, such as a metallic coil that expands and contracts, or more complex, such as a microcontroller similar to the BASIC Stamp.

These are two very unique means of controlling a process. First, the types of drive employed may be variable or on/off. Second, whether feedback from the system may or may not be used in the control of the system.

INPUT, DRIVE AND MONITORING

Just as important as the type of control employed are the methods used for the input into the controller. Will the inputs provide a simple on/off input to the controller? If using an analog (variable level) input instead of a digital one (only two levels), how can it be conditioned for on/off input if needed? If analog input is required, what are methods to bring this data into the BASIC Stamp? How is the data represented in the BASIC Stamp and how can it be converted to meaningful information?

In terms of the drive of a system, there are several questions as well. Do we need to employ on/off control of the actuator, such as turning a heater or pump on or off? Does the process require variable control of the drive such as regulating heat or flow output between on or off? Does the process actuator require higher current or voltage than is provided by the BASIC Stamp? How can the BASIC Stamp outputs be used to control these actuators?

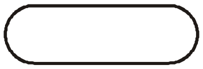
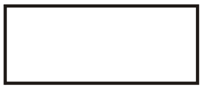
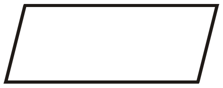
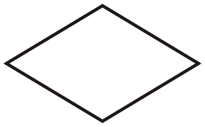
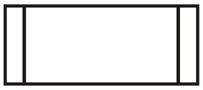
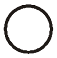
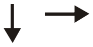
In industry, monitoring of systems is often required in order to ensure proper control action and to determine response in order to adjust this control action. Another monitoring aspect is data logging, or being able to collect real time data from the system for analysis.

This text explores these areas of process control through simple circuits using the BASIC Stamp microcontroller, and illustrates use with much larger systems.

ACTIVITY #1: FLOWCHARTS FOR REPRESENTING PROCESSES

When you hear the word 'flowchart', it may bring to mind programming, but a flowchart is often used for more than programming. A flowchart is a graphical representation of steps and decisions used to arrive at a logical outcome. It can be used to arrive at management decisions, system troubleshooting decisions, and other processes that involve well-defined steps and outcomes. Table 1-1 shows the most popular symbols used in flowcharting. These blocks, connected with flow lines, are used to describe the actions and flow of the program.

Table 1-1: Flowcharting Symbols

	Start/Stop: Indicates the beginning or end of a program or routine.
	Process: Indicates an internal process, such as calculations or delays.
	Input/Output: Indicates an input from an external source or output to an external source.
	Decision: Indicates a decision to continue flow in one of two directions based on a condition.
	Predefined Process: Indicates a predefined process, such as a subroutine, to be performed.
	Matching connectors indicate a connection between two locations in the flowchart.
	Flow lines: Indicates direction of flow between symbols.

While flowcharting has fallen out of fashion in many programming circles due to the advent of object-oriented programming (PBASIC used by the BASIC Stamp is procedural language), it is still an excellent tool when planning program flow. Flowcharting is particularly useful in process control because it can be used to visually represent the steps and decisions required to perform control of the system.

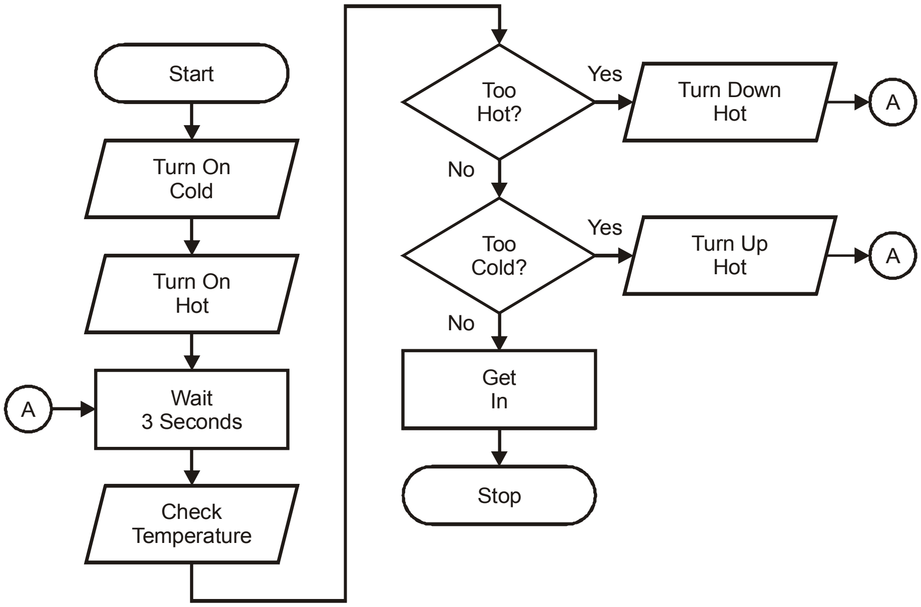
Take the everyday task of preparing the temperature of the shower before stepping into it. In pseudocode, or English statements outlining the steps to take, this is how we would proceed:

```
1. Turn on cold water.
2. Turn on hot water.
3. Wait 3 seconds for temperature to stabilize.
4. Test water temperature.
5. If too hot, then:
      a. Turn hot water down.
      b. Go back to step 3.
6. If too cold, then:
      a. Turn hot water up.
      b. Go back to step 3.
7. If just right then get in shower.
```

While it's not too difficult to read through these steps to see what actions should be taken, as a program or procedure becomes more complex it becomes more difficult to visualize the flow of the process and what actions and branches are needed. For example, how much more complex would the flow be if the hot water valve becomes fully open before the optimum temperature is reached?

As complexity increases, a flowchart makes it easier to visualize how the process will flow. Take a look at the flowchart in Figure 1-2, which describes the same process as the pseudocode above.

Figure 1-2 Adjusting Shower Temperature Flowchart



Note how each of the symbols is used.

- Typically, an input/output symbol is used when bringing data or information into the controller (in this case the person adjusting the temperature by sensing and adjusting the water actuators).
- The processing symbol is used when the controller is performing internal processing of data or a task, such as waiting or calculations.
- Finally, decision blocks are used to guide the flow of the procedure in one direction or another based on the decision results.

A decision can take one of two forms:

- Questions resulting in Yes or No.
- Statements resulting in True or False.

As humans, we typically work with questions resulting in yes/no. In flowcharting, it is better to use statements that result in true/false due to the logical nature of programming where conditions are checked to be true or false. Take the following example for the shower process:

- Is the water too hot? YES – Turn down the hot.
- The water is too hot. TRUE – Turn down the hot.

In programming, a typical condition may be:

```
IF (Water_Temp > 95) THEN …
```

In this example, when the condition is checked, the equality will either be true or false. Using true/false statements makes the transition from the flowchart to the programming language easier.

Challenge 1-1: Modify the Flowchart for True/False

√ Modify the flowchart in Figure 1-2 to use true/false statements instead of yes/no questions.

ACTIVITY #2: SEQUENTIAL FLOW AND CODE

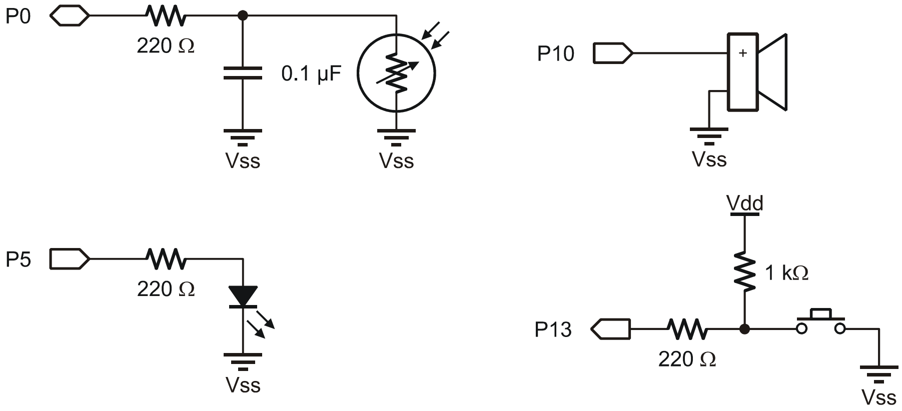
"Sequential flow" means moving from one operation to the next with no branches being made. In this activity a simple circuit will be used to illustrate principles of sequential flow and how the PBASIC language is used in programming the BASIC Stamp.

Parts Required

(3) Resistors – 220 Ω
(1) Resistor – 1 kΩ
(1) Photoresistor
(1) Pushbutton – Normally Open
(1) LED – Red
(1) Piezospeaker
(1) Capacitor – 0.1 μF

√ Construct the photoresistor, LED, piezospeaker, and pushbutton circuits shown in Figure 1-3.

Figure 1-3 Test Circuit Schematics



For an introduction to building basic circuits with these components, please see *What's a Microcontroller?*, the recommended starting point for the Stamps in Class series. It is available for free download or purchase from www.parallax.com.

Figure 1-4 is a flowchart to have the circuit continuously perform a sequence of operations. Without knowing any programming, can you determine what should occur when the program is entered and run?

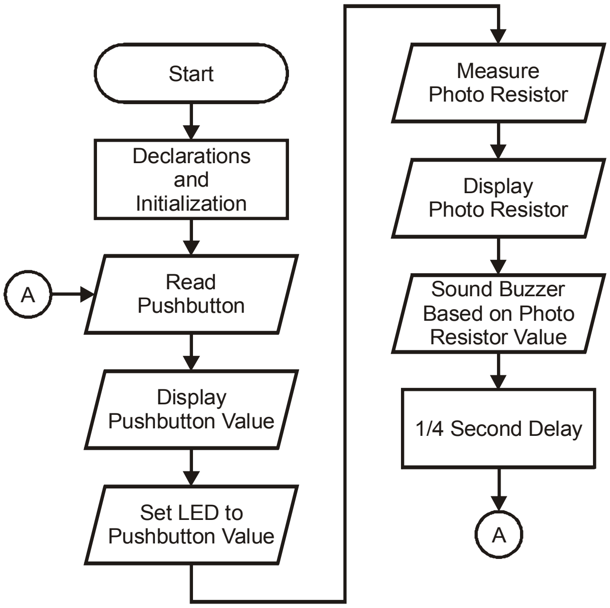


Figure 1-4
Simple Sequential Operation Flowchart

AVOID TYPOS! All of the BASIC Stamp (.bs2) programs listed in this text are available for free download from the Process Control product page at www.parallax.com.

Example Program: SimpleSequentialProgram.bs2

√ Enter and run SimpleSequentialProgram.bs2.

```
' -----[ Title ]-----------------------------------------------------------
' Process Control - SimpleSequentialProgram.bs2
' Tests and illustrates sequential flow using a simple test circuit.
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Declarations ]----------------------------------------------------
Photo     PIN  0              ' Alias for photo resistor circuit on P0
LED       PIN  5              ' Alias for LED on P5
Buzzer    PIN  10             ' Alias for buzzer on P10
PB        PIN  13             ' Alias for pushbutton on P13
PBVal     VAR  Bit            ' Bit variable to hold pushbutton value
PhotoVal  VAR  Word           ' Word variable to hold RC Time value
```

```
BuzzerDur CON  250            ' Constant for duration of tone for buzzer


' -----[ Initialization ]--------------------------------------------------
OUTPUT LED                    ' Set LED pin to be an output
OUTPUT Buzzer                 ' Set Buzzer pin to be an output

' -----[ Main Routine ]----------------------------------------------------
DO
  ' ******** Read Pushbutton
  PBVal = PB                  ' Read Pushbutton value and assign to PBVal
                              ' Display Pushbutton value

  ' ******** Display pushbutton value

  DEBUG CLS, "Pushbutton Value = ", DEC PBVal,CR

  ' ******** Set LED to pushbutton value
  LED = PBVal                 ' Set LED based on Pushbutton value

  ' ******** Measure Photoresistor
  HIGH Photo                  ' Charge photoresistor's RC network Capacitor
  PAUSE 10                    ' Allow 10 milliseconds to charge fully
  RCTIME Photo, 1, PhotoVal   ' Measure discharge time through photoresistor

  ' ******** Display photoresistor value
  DEBUG "Photo RC Time Value = ", DEC PhotoVal,CR

  ' ******** Sound buzzer at set duration at frequency of PhotoVal
  FREQOUT Buzzer,BuzzerDur,PhotoVal

  ' ******** 1/4 seconds delay
  PAUSE 250                   ' 1/4 second pause
LOOP                          ' Loop back to DO to repeat continuously
```

√ Test the circuit by pressing the pushbutton and varying the light falling on the photoresistor.

- o When the button is pressed does the state of the pushbutton change from 1 to 0 in the Debug Terminal?
- o When the button is pressed does the LED change from on to off?
- o When the sensor is darkened does the photoresistor RC time value change in the Debug Terminal?
- o Does the frequency output of the buzzer change in relation to the photoresistor's RC time value? Note that the buzzer has a very limited frequency response range.

√ If your circuit does not operate properly, verify your circuit connections and code.

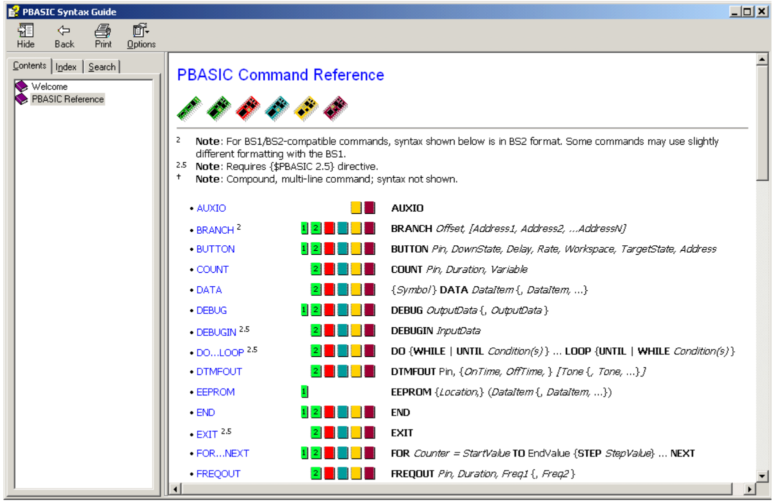
Code Discussion

As you read through the program, you can see that the coding that corresponds to the various elements of the flowchart are well highlighted using comments.

The pushbutton switch is active-low, meaning that its value is 0 when pressed. This is because the pushbutton is pulled up to Vdd when not pressed and brought to Vss when pressed. (This will be explored more in Chapter 3.)

Note that the flowchart block for 'Measure Photo Resistor' takes 3 lines of code. The flowchart just describes the process and is not intended to be a line-by-line description. This flowchart could be used for coding or designing any number of devices in any number of languages.

Looking it Up: The PBASIC commands and programming techniques used here were introduced in *What's a Microcontroller?*, the recommended prerequisite to *Process Control*. If you would like a refresher about specific program elements, you can look it up quickly in the BASIC Stamp Editor's Help file. Or, refer to the *BASIC Stamp Syntax and Reference Manual*, available for purchase or free download from www.parallax.com.



PBASIC Command Reference

[2] **Note**: For BS1/BS2-compatible commands, syntax shown below is in BS2 format. Some commands may use slightly different formatting with the BS1.
[2.5] **Note**: Requires {$PBASIC 2.5} directive.
[†] **Note**: Compound, multi-line command; syntax not shown.

• AUXIO	🟨 🟫	**AUXIO**
• BRANCH [2]	1 2 🟥 🟦 🟨 🟫	**BRANCH** *Offset, [Address1, Address2, ...AddressN]*
• BUTTON	1 2 🟥 🟦 🟨 🟫	**BUTTON** *Pin, DownState, Delay, Rate, Workspace, TargetState, Address*
• COUNT	2 🟥 🟦 🟨 🟫	**COUNT** *Pin, Duration, Variable*
• DATA	2 🟥 🟦 🟨 🟫	{*Symbol*} **DATA** *DataItem* {, *DataItem, ...*}
• DEBUG	1 2 🟥 🟦 🟨 🟫	**DEBUG** *OutputData* {, *OutputData* }
• DEBUGIN [2.5]	2 🟥 🟦 🟨 🟫	**DEBUGIN** *InputData*
• DO...LOOP [2.5]	2 🟥 🟦 🟨 🟫	**DO** {**WHILE** | **UNTIL** *Condition(s)* } ... **LOOP** {**UNTIL** | **WHILE** *Condition(s)* }
• DTMFOUT	2 🟥 🟦 🟨 🟫	**DTMFOUT** Pin, {*OnTime, OffTime,* } *[Tone* {, *Tone, ...*} *]*
• EEPROM	1	**EEPROM** {*Location,*} (*DataItem* {, *DataItem, ...*})
• END	1 2 🟥 🟦 🟨 🟫	**END**
• EXIT [2.5]	2 🟥 🟦 🟨 🟫	**EXIT**
• FOR...NEXT	1 2 🟥 🟦 🟨 🟫	**FOR** *Counter = StartValue* **TO** EndValue {**STEP** *StepValue*} ... **NEXT**
• FREQOUT	2 🟥 🟦 🟨 🟫	**FREQOUT** *Pin, Duration, Freq1* {, *Freq2* }

Figure 1-5
BASIC Stamp Editor's Help Files

The PBASIC Syntax Guide places information and examples for all commands at your fingertips.

Challenge 1-2: Coding from a Flowchart

Figure 1-6 is a flowchart for a different sequence of operations, using the same circuit. Code a program to match this sequence of events. Hints for coding are provided in the flow symbols.

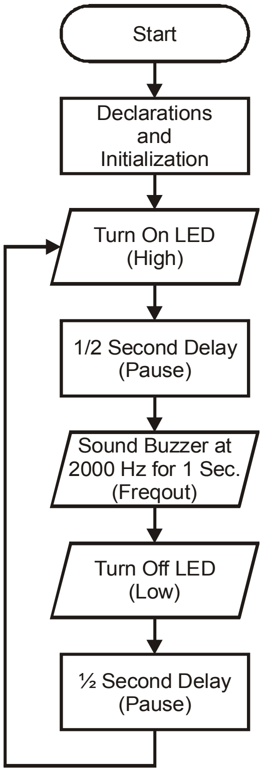


Figure 1-6
Challenge 1-2
Flowchart

ACTIVITY #3: FLOW AND CODING WITH CONDITIONAL BRANCHES

In most processes, measurements are made and decisions are then based on those measurements (such as, in the shower example, whether to turn up or down the hot water based on the current temperature). In the BASIC Stamp, there are multiple ways to code decisions and conditional branches.

Parts Required

Same as Activity #2

Consider the flowchart in Figure 1-7. What should occur when the button is pressed, and when it is not pressed?

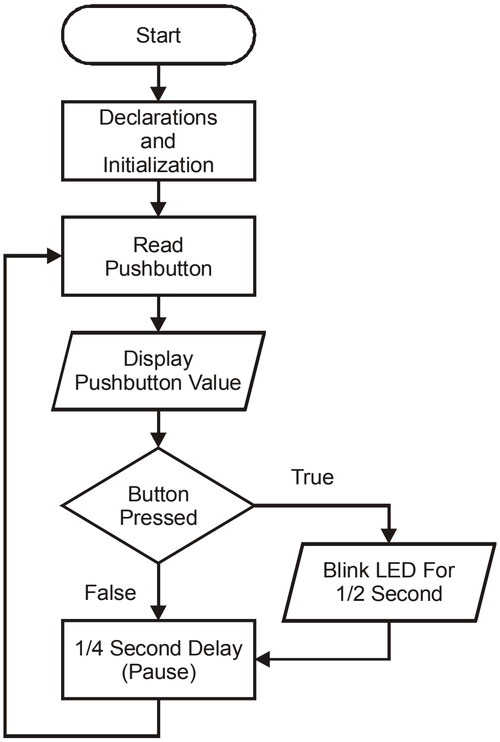


Figure 1-7
Conditional LED Blink Flowchart

If you said the LED would blink on for ½ second when the button is pressed, and not at all when not pressed, you would be correct.

Example Program: ConditionalLEDBlink.bs2

√ Enter, save and run ConditionalLEDBlink.bs2.

```
' -----[ Title ]-----------------------------------------------------------
' Process Control - ConditionalLEDBlink.bs2
' Blinks the LED based on state of Pushbutton
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Declarations ]----------------------------------------------------
Photo     PIN  10       ' Alias for photo resistor circuit on P0
LED       PIN  5        ' Alias for LED on P5
Buzzer    PIN  10       ' Alias for buzzer on P10
PB        PIN  13       ' Alias for pushbutton on P13
PBVal     VAR  Bit      ' Bit variable to hold pushbutton value
PhotoVal  VAR  Word     ' Word variable to hold RC Time value
BuzzerDur CON  250      ' Constant for duration of tone for buzzer

' -----[ Main Routine ]----------------------------------------------------
DO
  ' ******** Read Pushbutton
  PBVal = PB            ' Read Pushbutton Value and assign to PBVal

  ' ******** Display Pushbutton value
  DEBUG CLS,"Pushbutton value = ", DEC PBVal,CR

  ' ******** Button Pressed Conditional and Code
  IF (PBVal = 0) THEN   ' If pushbutton pressed is true then,
        HIGH LED        ' blink the LED
        PAUSE 500
        LOW LED
  ENDIF
  ' ******** 1/4 second pause
  PAUSE 250
LOOP                    ' Loop back to DO to repeat continuously
```

Code Discussion

The **IF…THEN…ENDIF** block is used to test the condition. Based on the result, the program will execute the code within the block if true or skip over it if false.

```
IF (PBVal = 0) THEN   ' If condition is true then,
      HIGH LED        ' blink the LED
      PAUSE 500
      LOW LED
ENDIF
' ******** 1/4 second pause
PAUSE 250
```

When the button is not pressed, the conditional test of **`PBVal=0`** will result in false because the value of **`PBVal`** is 1. Execution will branch to after the **`ENDIF`**, executing the **`PAUSE 250`**.

When the button is pressed, **`PBVal`** will in fact equal 0; **`PBVal=0`** will be true, the code within the block will be executed, and the LED will blink.

Code Formatting Tip: While indents in lines are not required, they do help to visually represent code that is common to sections.

Challenge 1-3: Code for True and False Conditions

Many times, different code must be executed depending on whether a condition is true or false. The **`IF…THEN…ELSE...ENDIF`** structure can be used to perform this task. If the condition is false, the code in the **`ELSE`** section will be executed.

```
IF (condition) THEN
  Code to run if true
ELSE
  Code to run if false
ENDIF
```

√ Figure 1-8 is a flowchart that requires different code depending on whether the button is pressed or not. Modify ConditionalLEDBlink.bs2 to match the flowchart's operation.

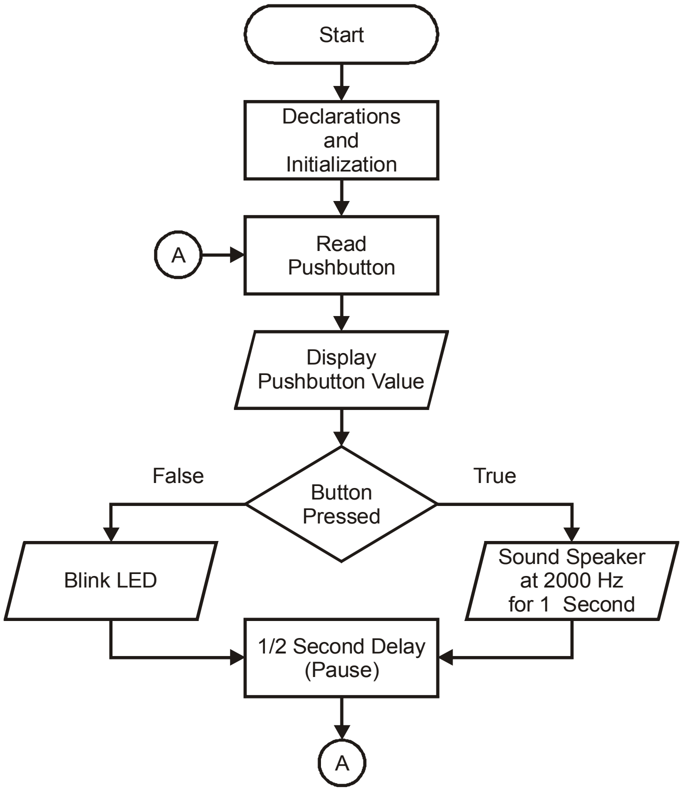


Figure 1-8
Conditional LED Blink or Tone Flowchart