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

IR Remote for the Boe-Bot

By Andy Lindsay

VERSION 1.1

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 1-928982-31-X

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

AN AUTONOMOUS ROBOT AND A HANDHELD REMOTE

The handheld remote may be the tool of choice for channel surfing couch potatoes worldwide, but this device can also be used to send messages to your Boe-Bot® robot. The press of a button on the remote's keypad opens up an array of new Boe-Bot possibilities. Once you understand how a remote uses infrared to transmit messages to a TV, programming the BASIC Stamp® 2 microcontroller to detect and process these messages is pretty easy. Once these tasks are reduced to subroutines, programming the Boe-Bot to take action based on these messages is a snap.

Here are a few Boe-Bot applications that you will have the opportunity to try in the next three chapters:

- Press and hold keys on the remote's keypad to control your Boe-Bot like a remote-controlled car.
- Send messages to your Boe-Bot while it autonomously roams to change the way it behaves.
- Remotely enable/disable the Boe-Bot program with the power on/off key.
- Tell the Boe-Bot which program to run.
- Remotely program the Boe-Bot with motion sequences.

Along the way, you will learn about pulse width modulation (PWM) for sending electronic messages, the binary number system, the PBASIC **`PULSIN`** command, and new uses for the **`RCTIME`** and **`SELECT…CASE`** commands.

AUDIENCE

This text is organized so that it can be used by the widest possible variety of students as well as independent learners. Middle school students can try the examples in this text in a guided tour fashion by simply following the check-marked instructions and instructor supervision. At the other end of the spectrum, pre-engineering students' comprehension and problem-solving skills can be tested with the questions, exercises, and projects (with solutions) in each chapter summary. The independent learner can work at his or her own pace, and obtain assistance through the Stamps in Class® Discussion Forum cited below.

EDUCATOR RESOURCES

IR Remote for the Boe-Bot 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 *IR Remote for the Boe-Bot* 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., 2005
"*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 Parallax team assembled to produce this text includes: application design and technical writing by Andy Lindsay, illustration by Rich Allred, cover design by Larissa Crittenden and Jen Jacobs, technical review by Kris Magri, and technical editing by Stephanie Lindsay. Thanks also to Parallax customers Robert Ang and Sid Weaver for their advanced feedback on the original 1.0 version. The Stamps in Class program was founded by Ken Gracey, and Ken wishes to thank the Parallax staff for the great job they do. Each and every Parallaxian has made contributions to this and every Stamps in Class text.

Chapter 1: Infrared Remote Communication

GETTING STARTED

The IR Remote AppKit has two documents you can use to get started:

- This book – takes you from beginner to advanced in a step-by-step format.
- IR Remote AppKit Documentation – a quick start guide that comes with the AppKit as a package insert and can be found in this book in Appendix A.

This book is, for the most part, a continuation of *Robotics with the Boe-Bot*. It follows the same format in terms of introducing new hardware, explaining how things work, and demonstrating new PBASIC techniques. By doing the activities, questions, exercises, and projects, you will build your programming, electronics, and robotics skills as you learn about infrared communication and control. The knowledge and skills you will gain will be useful for future robot and/or product designs of your own. NOTE: You will need a fully assembled Parallax Boe-Bot robot to complete the material in this text.

IR Remote AppKit Documentation (Appendix A) summarizes selected activities from this book in a few pages. It's mainly the bare essentials that an intermediate to advanced BASIC Stamp programming enthusiast needs to understand how IR communication works, and how to use the applications developed in this text. You can also find this document as a package insert in the Parallax IR Remote AppKit. NOTE: Unlike this text, a Boe-Bot robot is not required for the activities in the AppKit documentation; you may use your own customized BASIC Stamp project.

This book and the **IR Remote AppKit Documentation** are both available for free download from www.parallax.com.

KIT CONTENTS

The IR Remote AppKit contains a universal remote similar to the one shown in Figure 1-1. The activities in this chapter will make use of the signals sent by a universal remote after it has been configured to control a SONY® television set. Most universal remotes can be configured to send messages to a SONY TV. If you want to try these activities with a universal remote you already own, or one purchased at a local store, you will need to read the documentation that comes with the remote to find out how to configure it to control a SONY TV.

Infrared Remote Parts List:*

(1) 020-00001 Universal Remote and Universal Remote Manual**
(1) 350-00014 IR detector
(1) 150-02210 Resistor – 220 Ω
(1) 800-00016 Jumper wires – bag of 10

* Batteries sold separately
** The remote and instruction sheet/booklet may not be identical to the ones shown Figure 1-1.

Figure 1-1
Contents of IR Remote Kit

(batteries not included)

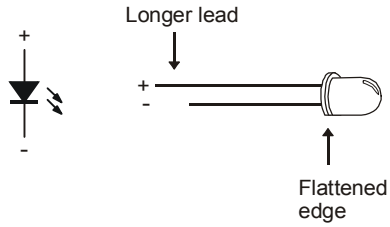
In addition to a fully assembled Boe-Bot, there are also some parts that you will need from your original Boe-Bot Robot Kit (see Figure 1-2). The activities in this chapter will make use of the same infrared detection circuit used in Chapters 7 and 8 of *Robotics with the Boe-Bot*. The resistors, LEDs, and piezospeaker should also be familiar from earlier chapters. Only one IR detector is required to capture messages sent by the remote. However, some of the later activities in this chapter will make use of the entire IR detection circuit for autonomous roaming combined with remote control.

Figure 1-2
Parts from Boe-Bot Parts Kit

Parts List:

(2) Infrared detectors

(2) IR LEDs – (clear case)



(2) IR LED shield assemblies

(2) Resistors – 220 Ω

(2) Resistors – 1 kΩ

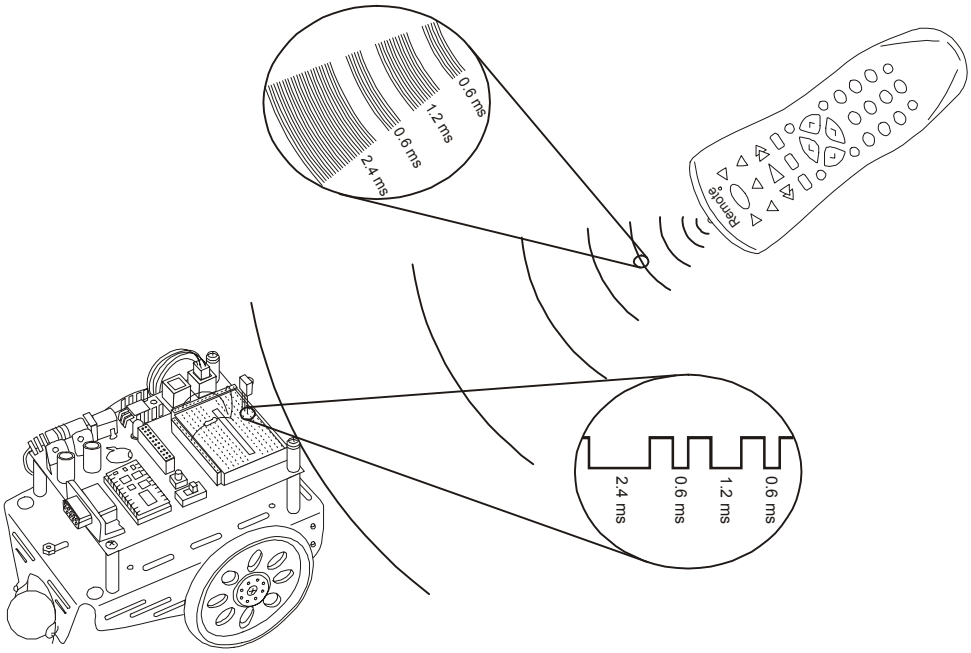
(1) Piezospeaker

(2) LEDs – red

HOW THE REMOTE SENDS MESSAGES

Figure 1-3 shows how the handheld remote can be used to send messages to a Boe-Bot. When a button on the remote is pressed, it flashes its IR LED on/off at 38.5 kHz to broadcast a code for that button. The code is produced by controlling the brief amounts of time the IR LED flashes on and off. The sequence of IR broadcast times shown in the figure corresponds to the beginning of the code that tells a SONY TV the 3 button has been pressed. To send messages to a SONY TV, the remote has to broadcast IR for 2.4 ms signaling that a message is about to start. The message that follows consists of a combination of 1.2 ms (binary-1) and 0.6 ms (binary-0) broadcasts. Notice how the IR detector on the Boe-Bot sends low signals that match the time pattern broadcast by the remote. The BASIC Stamp can then be programmed to detect, measure, record and interpret the durations of low pulses in this pattern to figure out which key on the remote was pressed.

Figure 1-3: Handheld Remote Infrared Messages



The IR receiver the Boe-Bot used for infrared object detection in *Robotics with the Boe-Bot* is the same detector found in many TVs and VCRs. This detector sends a low signal whenever it detects IR flashing on/off at 38.5 kHz and a high signal the rest of the time. When the IR detector sends low signals, the processor inside a TV or VCR measures how long each of the low signals lasts. Then, it uses these measurements to figure out which key was pressed on the remote. Like the processor inside a TV or VCR, the BASIC Stamp 2 can be programmed to detect, measure, store, and interpret the sequence of low pulses it receives from the same IR detector.

Pulse width modulation (PWM): Pulse durations are used in many applications, a few of which are digital-to-analog conversion, motor control, and communication. Since the IR detector sends low pulses that can be measured to determine what information the IR remote is sending, it's an example of using PWM for communication.

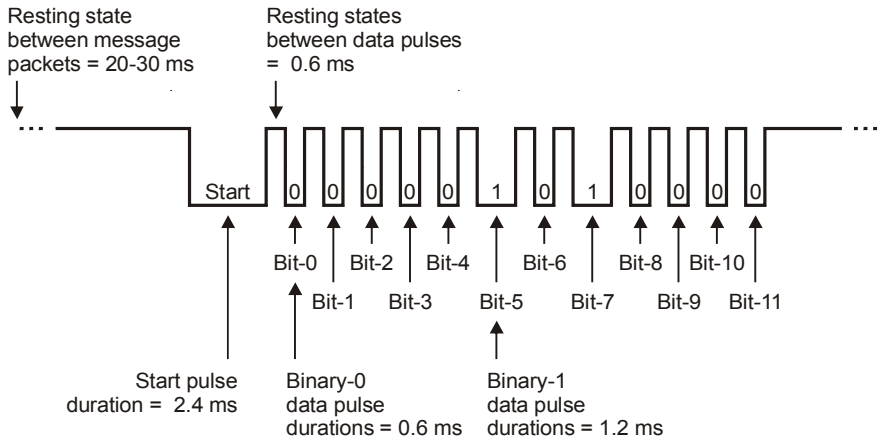
Carrier signal: The IR remote uses a 38.5 kHz "carrier signal" to transmit the pulse durations from the remote to the IR detector.

Communication protocol: A communication protocol is a set of rules for devices that have to exchange electronic messages. Protocols tend to have rules for voltages, the amount of time signals last, carrier signal frequencies and/or wavelengths, and much more. When two or more devices follow the rules of a given protocol, they should be able to communicate and exchange information.

There are many different communication protocols that a universal remote can use to transmit PWM messages to entertainment system components. This text will focus on the SONY protocol. It is easy to understand, and it works well with the same IR detector that we used for infrared object and distance detection in *Robotics with the Boe-Bot*.

Figure 1-4 shows a timing diagram example for an example signal the IR detector might send to the BASIC Stamp when it receives a SONY TV control message from the IR remote. This message consists of thirteen negative pulses that the BASIC Stamp can easily measure. The first pulse is the start pulse, which lasts for 2.4 ms. The next twelve pulses will either last for 1.2 ms (binary-1) or 0.6 ms (binary-0). The first seven data pulses contain the IR message that indicates which key is pressed. The last five pulses contain a binary value that specifies whether the message is intended being sent to a TV, VCR, CD, DVD player, etc. The pulses are transmitted in LSB-first order. This stands for least significant bit first, meaning the first data pulse is bit-0, the next data pulse is bit-1, and so on. If you press and hold a key on the remote, the same message will be sent over and over again with a 20 to 30 ms rest between messages.

Figure 1-4: IR Message Timing Diagram



In this chapter, you will start with simple programs to make your BASIC Stamp 2 module interpret and act on the pulse patterns sent by the infrared remote. You will then expand these programs to guide the Boe-Bot with the IR remote.

ACTIVITY #1: CONFIGURING YOUR REMOTE

In this activity, you will program your universal remote so that it sends PWM messages to a television set using the SONY protocol. In this case, the term "programming" means a sequence of key-presses on the remote that tells it to send signals to a SONY TV.

If you change the batteries in your remote, you will probably have to repeat the steps in this activity. Why? Because, when you remove the batteries, the remote will probably forget that it was programmed to be a SONY TV controller.

Don't throw away the instruction sheet/booklet that comes with your remote! The batteries for these remotes might last longer than your memory of the code and procedure for entering it. A sticker on the back of the remote with the code and button sequence can also come in really handy.

Infrared Remote Parts

(1) Universal remote
(1) Instruction sheet/booklet for the universal remote
Compatible batteries

How to Configure the Universal Remote to Send SONY TV Protocol Signals

These instructions are for the remote included in the IR Remote AppKit.

- √ Remove the battery compartment cover and determine how many and what kind of batteries to use (AA, AAA, etc).
- √ Load the battery compartment with new (or freshly charged rechargeable) batteries. Do not mix battery types.
- √ Find the TV setup codes section in the instruction sheet/booklet. Here are some examples of titles for that section: "Setup Codes for TV", "Setup Codes for Television", "TV Code List".
- √ Find the code for SONY from the TV code list, and make a note of it.
- √ Find the section that explains how to manually program a TV code into your remote. Here also, are examples of titles for that section: "Programming Your Remote", "To Manually Program Your Remote Control", "Programming for TV".
- √ Follow the instructions in the manual programming section for entering the SONY TV code into your remote.

The instructions might tell you to test it on your TV, but if it's not a SONY, the test probably won't work. In the next activity, you will test to make sure the code was correctly entered by verifying that the signals that the remote transmits have the SONY TV protocol characteristics.

Missing instruction booklets: If you want to try these activities with a remote you already own, you will probably need the instruction booklet for that remote. If it has been misplaced or lost, there may be a copy published on the World Wide Web.

Inexpensive universal remotes can also be purchased from local department stores, and most include instruction booklets. Check the package before you purchase a particular remote to make sure it can be configured to control a SONY TV. If the packaging says something like, "compatible with most/all major brands...", it will most likely work with SONY TVs.

Your Turn – Testing the Remote on Other Devices

√ If you'd like to test it out on your brand of TV/VCR, etc., try following the instructions in the remote manual to see if you can get it to work.

√ **Before moving on to the next activity:** If you followed the instruction in the previous checkmark, make sure to reprogram the remote to control a SONY TV!

ACTIVITY #2: CHARACTERIZING THE IR MESSAGES

This activity focuses on measuring the pulses the Boe-Bot's IR detector sends when it detects messages from the handheld remote. Figure 1-5 shows a timing diagram of the IR detector's output at the beginning of an IR remote message. The IR detector sends a low signal while it detects infrared flashing on/off at 38.5 kHz. While the IR detector does not detect 38.5 kHz infrared, it sends a high signal. Before the message can be interpreted by the BASIC Stamp module, the durations of these low signals have to be measured and stored.

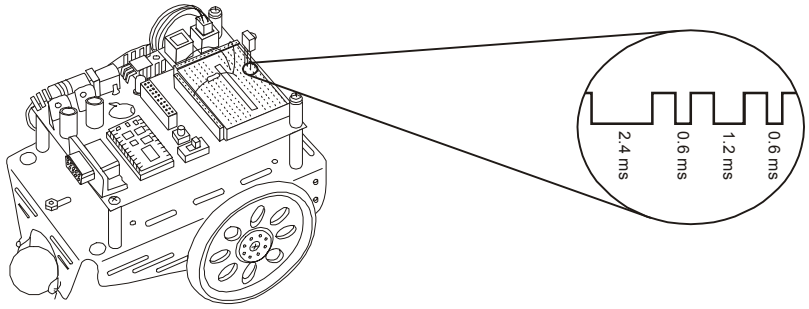


Figure 1-5
IR Detector Output for IR Remote Message

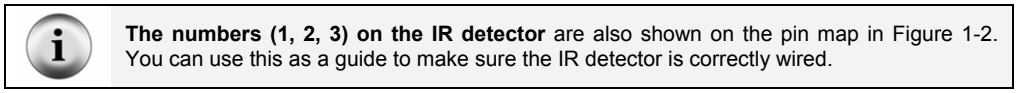
Infrared Detection Parts

All parts from the previous activity
(1) Infrared detector
(1) Resistor – 220 Ω (red-red-brown)
(2) Jumper wires

The IR Detection Circuit

It only takes one IR detector to capture messages from the IR remote (see Figure 1-6).

√ Build this circuit on your Boe-Bot's prototyping area.



The numbers (1, 2, 3) on the IR detector are also shown on the pin map in Figure 1-2. You can use this as a guide to make sure the IR detector is correctly wired.

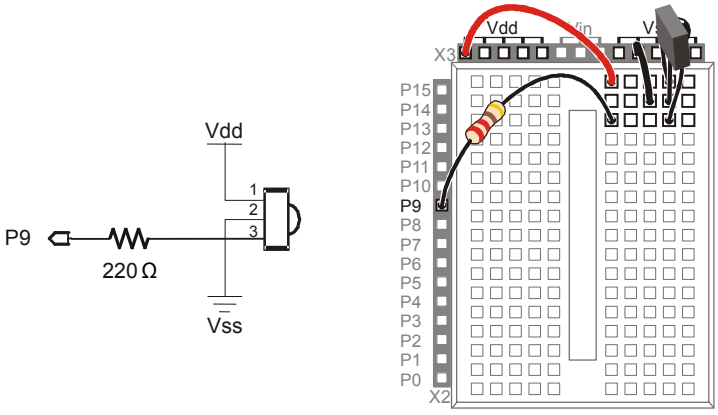


Figure 1-6
IR Detection Circuit

Measuring Start and Data Pulses

The **PULSOUT** command you've been using to send pulses to the Boe-Bot servos has a complementary command called **PULSIN**. The syntax for the **PULSIN** command is

PULSIN *Pin, State, Variable*

Pin is, of course, used to select the I/O pin for measuring the pulse. ***State*** is used to determine whether the pulse is a high pulse (1) or a low pulse (0). ***Variable*** stores the pulse duration measured by the BASIC Stamp.

High pulse vs. low pulse: If the voltage a BASIC Stamp I/O pin senses starts low, then goes high for a while before returning to low, that's a **high pulse**. The term **positive pulse** is also commonly used. The **PULSIN** command measures the amount of time the signal is high if the ***State*** argument is 1.

A **low pulse** is the opposite: the signal will be high, then drop low for a while before returning to the high state. It is also called a **negative pulse**. The **PULSIN** command measures the amount of time the signal is low if the ***State*** argument is 0.

Let's say your program has a word variable named **time** for storing the measured pulse duration. The 2.4 ms, 1.2 ms, and 0.6 ms pulses shown in Figure 1-7 are negative pulses. To measure them with the IR detector circuit, you will have to use the command:

```
PULSIN 9, 0, time
```

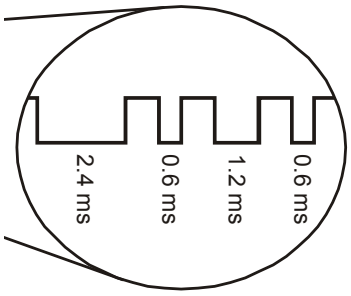


Figure 1-7
A Closer Look at the Pulses

The high time between two low pulses can be measured as a positive pulse. True, it doesn't contain any data, but it can be useful for figuring out how long an entire IR message takes to transmit. By changing the **PULSIN** command's ***State*** argument from 0 to 1, you can measure the duration of a positive pulse, like this:

```
PULSIN 9, 1, time
```

Example Program: CountStartPulses.bs2

The first pulse that we will examine using the **PULSIN** command is the 2.4 ms start pulse. This pulse won't be exactly 2.4 ms (2400 µs), but it should be fairly close, give or take 250 µs. This is the pulse that signifies that twelve more data pulses are about to be sent by the remote.

CountStartPulses.bs2 counts the number of low pulses that it receives with durations that fall in the 1.95 to 2.85 ms range. Figure 1-8 shows what your Debug Terminal should display after you have pressed a numeric key for a couple of seconds while pointing the remote at the IR detector. The duration shown in the Debug Terminal for a sample start pulse is 2562 µs. Although it's not exactly 2400 µs, it is well within +/- 250 µs.

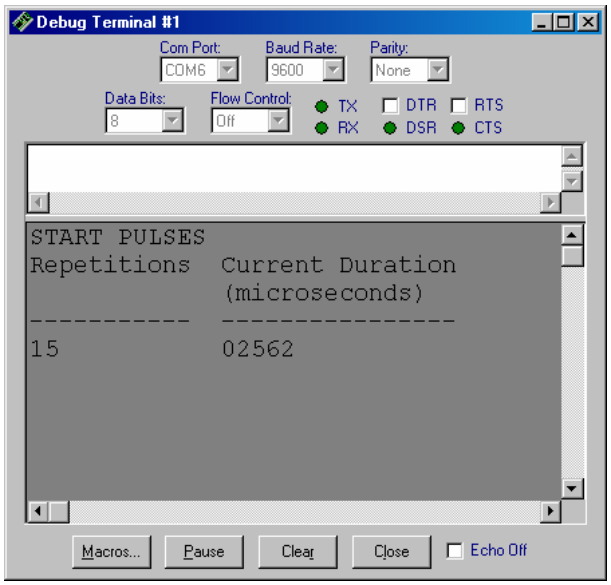


Figure 1-8
Debug Terminal for CountStartPulses.bs2

- √ Enter and run CountStartPulses.bs2.
- √ Point the remote at the bubble on the front of the infrared detector.
- √ Press and hold one of the numbered keys (0 – 9) on the remote.
- √ Make sure pulses in the 2.25 to 2.75 ms range are detected (that's 2250 to 2750 µs displayed in the Debug Terminal).
- √ Make a note of the actual value of the start pulse for your remote here: _____. If the value is different each time, make a guess at the average.

If the program does not detect the start pulse, make sure your remote is set to control a SONY TV set. First, press the TV button on the remote, then try beaming your BASIC Stamp another message. If that doesn't work, repeat the steps from Activity #1. Still no luck? Check your IR detector's wiring, then check your program for errors.

Do not press CBL, VCR, or TV/VCR. Your remote may or may not have these buttons. If it does and you press one of those keys, it tells the remote to send messages to a different device, either a VCR or cable box. In either case, the messages the remote sends will use a non-SONY TV protocol. The result will be that the programs in this book will seem to stop working.

If you accidentally press one of those keys: just press the TV key to get the remote back to sending (SONY) TV signals.

```
' IR Remote for the Boe-Bot - CountStartPulses.bs2
' Capture and count the number of 2.4 ms (low) start pulses.

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

time              VAR     Word
counter           VAR     Word

DEBUG "START PULSES", CR,
      "Repetitions  Current Duration", CR,
      "             (microseconds)  ", CR,
      "-----------  ----------------"

DO

  PULSIN 9, 0, time

  IF (time > 975) AND (time < 1425) THEN

    counter = counter + 1

    DEBUG CRSRXY, 0, 4,
          DEC counter,
          CRSRXY, 13, 4,
          DEC5 time * 2

  ENDIF

LOOP
```

How CountStartPulses.bs2 Works

This program starts by declaring two word variables, **`time`** to store the start pulse duration, and **`counter`** to store the number of start pulses received. Then, a **`DEBUG`** command adds some column headings for displaying the variables.

```
time              VAR     Word
counter           VAR     Word
DEBUG "START PULSES", CR,
      "Repetitions  Current Duration", CR,
      "             (microseconds)  ", CR,
      "-----------  ----------------"
```

Inside the **DO…LOOP**, the program executes the command **PULSIN 9, 0, time**, which measures pulses. Whenever a pulse duration is between 975 (1.95 ms) and 1425 (2.85 ms), the **IF…THEN…ENDIF** code block increments the **counter** variable and displays the **counter** and **time** variables under the **Repetitions** and **Current Duration** headings.

```
DO

  PULSIN 9, 0, time

  IF (time > 975) AND (time < 1425) THEN

    counter = counter + 1

    DEBUG CRSRXY, 0, 4,
          DEC counter,
          CRSRXY, 13, 4,
          DEC5 time * 2

  ENDIF

LOOP
```

What's the * 2 in the DEBUG command? Remember that the **PULSIN** command measures duration in 2 µs units. That's what gets stored in the **time** variable, the number of 2 µs units the **PULSIN** command measured. By multiplying **time** by 2 with the ***** operator, the **DEBUG** command displays the actual number of microseconds for the pulse measurement.

Your Turn – Measuring the Binary-1 and Binary-0 Pulses

By modifying the **DEBUG** and **IF…THEN** statement in the example program, you can capture and display the duration the binary-1 and binary-0 pulses. Here's how to:

√ Save CountStartPulses.bs2 as MeasureBinary1Pulses.bs2.

√ Change

```
DEBUG "START PULSES", CR,
```

to

```
DEBUG "BINARY-1 PULSES", CR,
```

√ Change

```
IF (time > 975) AND (time < 1425) THEN
```

to

```
IF (time > 450) AND (time < 750) THEN
```

√ Save your modified program.
√ Run the program.
√ Press and hold one of the numbered keypad keys until the Debug Terminal displays a pulse duration.
√ Record the pulse duration for binary-1 here _____.
√ Save MeasureBinary1Pulses.bs2 as MeasureBinary0Pulses.bs2.
√ Change

```
DEBUG "BINARY-1 PULSES", CR,
```

to

```
DEBUG "BINARY-0 PULSES", CR,
```

√ Change

```
IF (time > 450) AND (time < 750) THEN
```

to

```
IF (time > 150) AND (time < 450) THEN
```

√ Save your modified program.
√ Run the program and use one of the remote's numbered keypad keys to send messages to your Boe-Bot.
√ Record the pulse duration for binary-0 here _____.

Measuring the Resting State between Messages

When you press and hold a given key on the remote, the remote sends the code for that key, then waits a while and sends it again. CountStartPulses.bs2 also can be modified to search for this resting time between messages. This resting time is shown in Figure 1-9, and it turns out to be an important factor in Boe-Bot navigation.

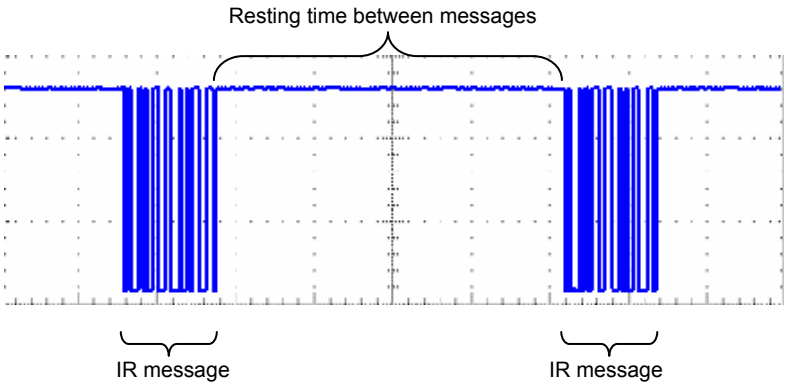


Figure 1-9
Two IR Messages

This screen capture from the Parallax USB Oscilloscope shows two IR messages and the resting state between them.

To measure this resting state with the BASIC Stamp module, all you have to do is think of the high time between messages as a positive pulse with a very long duration. The **`PULSIN`** command must be modified to search for a positive pulse by changing the ***`State`*** argument from 0 to 1. The **`IF…THEN`** statement also has to be modified so that it takes no action if the measured time is less than 1000 (2 ms). This ensures that the brief high pulses between the start pulse and data bits won't be reported. Instead, only the longer high time between messages will be reported.

Example Program: CountRestingStates.bs2

- √ Enter and run CountRestingStates.bs2.
- √ Point the remote at the IR detector, and press and hold the 5 key.
- √ Make sure high pulses in the 20 to 35 ms range (20,000 to 35,000 µs) are detected.
- √ Make a note of the actual duration of the resting state between messages here: _____.