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

PARALLAX INC

599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

Infrared Remote AppKit (#29122)

A Wireless Keypad for Your BASIC Stamp® Microcontroller Module

With a universal remote and an infrared receiver, you can add a wireless keypad to your BASIC Stamp Applications. The IR receiver is inexpensive, and only takes one I/O pin. Universal remotes are also inexpensive, easy to obtain and replace, and have enough buttons for most applications. The parts in this kit along with the example programs make it possible to enter values and control your projects in the same way you might with a TV, VCR, or other entertainment system component.

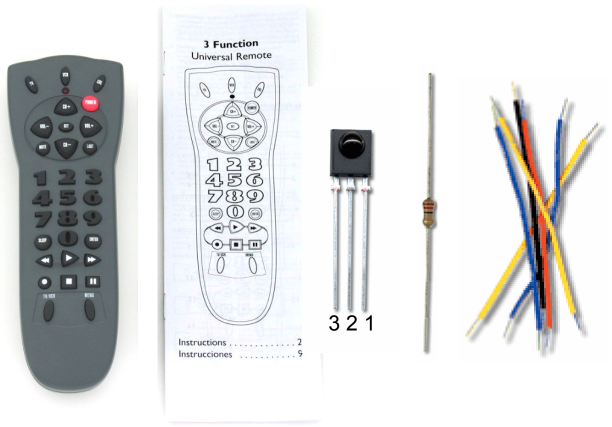
IR Remotes can also add zing to your robotics projects. While this package insert provides you with the essential background information, circuits, and example programs to get started, you can learn lots more with *IR Remote for the Boe-Bot* by Andy Lindsay of Parallax Inc. This text is, for the most part, a continuation of *Robotics with the Boe-Bot*, but with an IR remote twist. It follows the same format in terms of introducing new hardware, explaining how things work, and demonstrating new PBASIC techniques. IR remote applications for the Boe-Bot® robot include remote control, keypad entry control, hybrid autonomous and remote control, and remote motion sequence programming.

Kit Contents*

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

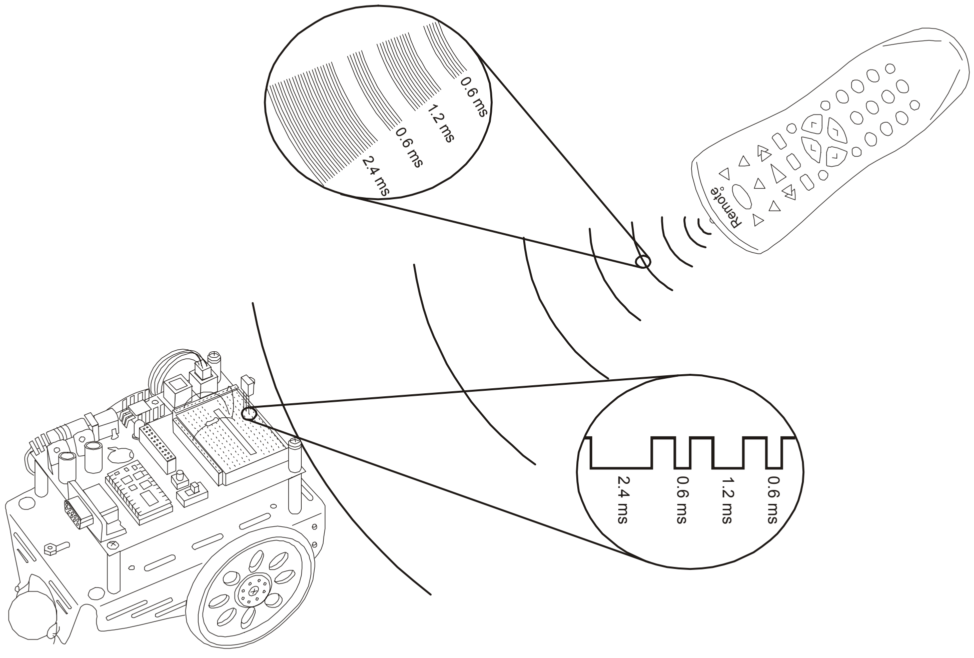
*Requires 2 alkaline AA batteries, not included


3 Function
Universal Remote
Instructions 2
Instrucciones 9
3 2 1

How IR Communication Works

The universal remote sends messages by strobing its IR LED at 38.5 kHz for brief periods of time. The actual data is contained in the amount of time each strobe lasts. Each IR protocol is different. In general, the amount of time each 38.5 kHz signal lasts transmits some kind of message. One duration might indicate the start of a message, while another indicates a binary-1, and still another indicates a binary-0.

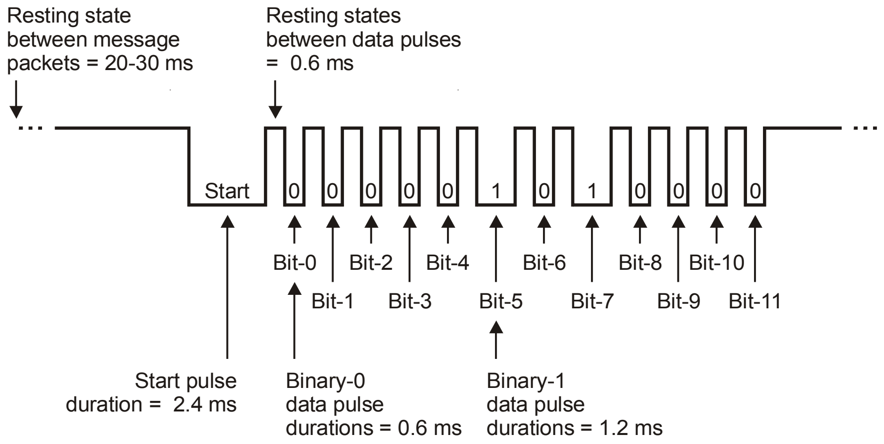
The IR detector's output pin sends a low signal while it detects the 38.5 kHz IR signal, and a high signal while it does not. So, a low signal of one duration might indicate the start of a message, while another indicates a binary-1, and still another indicates a binary-0. This communication scheme is called pulse width modulation (PWM), because when it is graphed against time, the IR detector's high/low signals form pulses of different widths that correspond to their durations.



Handheld Remote Infrared Messages

Excerpt from *IR Remote for the Boe-Bot* text.

The examples here will rely on the protocol that universal remotes use to control SONY® TV sets. This protocol strobes the IR thirteen times with roughly a half-millisecond rest between each pulse. It results in thirteen negative pulses from the IR detector 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 to be sent to a TV, VCR, CD, DVD player, etc. The pulses are transmitted in LSB-first order, so 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 re-sent after a 20 to 30 ms rest.

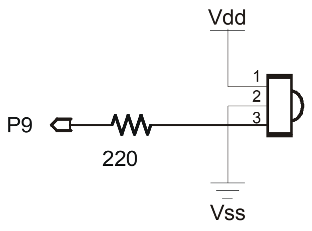

Resting state between message packets = 20-30 ms
Resting states between data pulses = 0.6 ms
Start
0
0
0
0
0
1
0
1
0
0
0
0
Bit-0
Bit-1
Bit-2
Bit-3
Bit-4
Bit-5
Bit-6
Bit-7
Bit-8
Bit-9
Bit-10
Bit-11
Start pulse duration = 2.4 ms
Binary-0 data pulse durations = 0.6 ms
Binary-1 data pulse durations = 1.2 ms

IR Message Timing Diagram

Values are approximate and will vary from one remote to the next

IR Detection Circuit

For testing purposes, all you need is this IR detector circuit and the BASIC Stamp Editors's Debug Terminal.



IR Detector Circuit

IR detector viewed from the top. Also see Kit Contents figure for pin map.

BASIC Stamp 2 "Bare-Bones" Example – IrRemoteCodeCapture.bs2

This example program demonstrates how to capture and display a remote code with the BASIC Stamp 2. If you modify the $STAMP directive, it can also be used with the BASIC Stamp 2e or 2 pe.

√ First, make sure to use the documentation that comes with your universal remote to configure it to control a SONY®TV.
√ Press the TV button on your remote so that you know it is sending TV signals.
√ Download or hand enter and run IrRemoteCodeCapture.bs2
√ Point the remote at the IR detector, and press/release the digit keys.
√ Also try POWER, CH+/-, VOL+/-, and ENTER to view the codes for these values.

```
' Ir Remote Application - IrRemoteCodeCapture.bs2
' Process incoming SONY remote messages & display remote code.

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

' SONY TV IR remote variables

irPulse         VAR     Word                    ' Stores pulse widths
remoteCode      VAR     Byte                    ' Stores remote code

DEBUG "Press/release remote buttons..."

DO                                              ' Main DO...LOOP

  remoteCode = 0

  DO                                            ' Wait for end of resting state.
    RCTIME 9, 1, irPulse
  LOOP UNTIL irPulse > 1000

  PULSIN 9, 0, irPulse                          ' Get data pulses.
  IF irPulse > 500 THEN remoteCode.BIT0 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT1 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT2 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT3 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT4 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT5 = 1
  RCTIME 9, 0, irPulse
  IF irPulse > 300 THEN remoteCode.BIT6 = 1

  ' Map digit keys to actual values.
  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  DEBUG CLS, ? remoteCode

LOOP                                     ' Repeat main DO...LOOP
```

How IrRemoteCodeCapture.bs2 Works

Each time through the outermost DO…LOOP, the value of remoteCode is cleared. There's also an inner DO…LOOP with an RCTIME command to detect the end of a high signal that lasted longer than 2 ms. This indicates that the rest between message packets just ended, and the start pulse is beginning. The first PULSIN command captures the first data pulse, and the IF…THEN statement that follows uses the value of the irPulse variable to either set (or leave clear) the corresponding bit in the remoteCode variable. Since the next data pulse has already started while the IF…THEN statement was executing, the remainder of the next data pulse is measured with an RCTIME command. This next value is again used to either set (or leave clear) the next bit in remoteCode. This is repeated five more times to get the rest of the useful part of the IR message and set/clear the rest of the bits in remoteCode.

The BS2sx and BS2p handle remote codes a little differently. The programs usually search for the actual start pulse with a PULSIN command instead of searching for the resting state between messages. They also use PULSIN commands to capture all the pulses since the IF…THEN statements that sets bits in the remoteCode variable complete before the starting edge of the next data pulse. To see a code example that does this, see the #CASE statement for the BS2sx and BS2p inside the next example program's Get_Ir_Remote_Code subroutine.

BASIC Stamp 2 Series Application Example – IrRemoteButtonDisplay.bs2

You can use this application example with BASIC Stamp 2, 2e, 2sx, 2p or 2pe modules to test your remote and display which key you pressed.

- √ As with the previous example program, make sure your remote is configured to control a SONY TV first.
- √ Update the $Stamp directive for the BASIC Stamp module you are using.
- √ Download or hand enter, then run IrRemoteButtonDisplay.bs2.
- √ Point the remote at the IR detector, press and release buttons
- √ Make sure the Debug Terminal reports the correct button. Start with digits, channel, volume, etc.

You can modify or expand the SELECT…CASE statement to test for VCR keys defined in the Constants section (Play, Stop, Rewind, etc.). There are usually several different codes for configuring universal remotes to control SONY VCRs, so you may need to try a few before finding the code that makes the remote speak the same PWM language as the TV controller. You can determine if the code worked because number, CH/VOL+-, and POWER keys will still work after you have pressed the VCR button.

```
' -----[ Title ]-----------------------------------------------------------
' Ir Remote Application - IrRemoteButtonDisplay.bs2

' Process incoming SONY remote signals and display the corresponding button
' in the Debug Terminal.

' {$STAMP BS2}                               ' BS2, 2sx, 2e, 2p, or 2pe
' {$PBASIC 2.5}

' -----[ Revision History ]------------------------------------------------

' V1.0 - Supports most SONY TV and VCR control buttons.
'        Supports BASIC Stamp 2, 2SX, 2e, 2p, and 2pe modules.

' -----[ I/O Definitions ]-------------------------------------------------

' SONY TV IR remote declaration - input receives from IR detector

IrDet           PIN     9                    ' I/O pin to IR detector output

' -----[ Constants ]-------------------------------------------------------

' Pulse duration constants for SONY remote.

#SELECT $stamp
  #CASE BS2, BS2E, BS2PE                     ' PULSE durations
    ThresholdStart CON 1000                  ' Message rest vs. data rest
    ThresholdPulse CON 500                   ' Binary 1 vs. 0 for PULSIN
    ThresholdEdge  CON 300                   ' Binary 1 vs. 0 for RCTIME
  #CASE BS2P, BS2SX
    ThresholdStart CON 2400                  ' Binary 1 vs. start pulse
    ThresholdPulse CON 500  * 5 / 2          ' Binary 1 vs. 0 for PULSIN
  #CASE #ELSE
    #ERROR This BASIC Stamp NOT supported.
#ENDSELECT

' SONY TV IR remote constants for non-keypad buttons

Enter           CON     11
ChUp            CON     16
ChDn            CON     17
VolUp           CON     18
VolDn           CON     19
Mute            CON     20
Power           CON     21
TvLast          CON     59                   ' AKA PREV CH
```

```
' SONY VCR IR remote constants

' IMPORTANT: Before you can make use of these constants, you must
' also follow the universal remote instructions to set your remote
' to control a SONY VCR.  Not all remote codes work,  so you may have to
' test several.

VcrStop        CON     24
VcrPause       CON     25
VcrPlay        CON     26
VcrRewind      CON     27
VcrFastForward CON     28
VcrRecord      CON     29

' Function keys

FnSleep        CON     54
FnMenu         CON     96

' -----[ Variables ]-------------------------------------------------------

' SONY TV IR remote variables

irPulse        VAR     Word                  ' Stores pulse widths
remoteCode     VAR     Byte                  ' Stores remote code

' -----[ Initialization ]--------------------------------------------------

DEBUG "Press/release remote buttons..."

' -----[ Main Routine ]----------------------------------------------------

' Replace this button testing DO...LOOP with your own code.

DO                                           ' Main DO...LOOP

  GOSUB Get_Ir_Remote_Code                   ' Call remote code subroutine

  DEBUG CLS, "Remote button: "               ' Heading

  SELECT remoteCode                          ' Select message to display
    CASE 0 TO 9
      DEBUG DEC remoteCode
    CASE Enter
      DEBUG "ENTER"
    CASE ChUp
      DEBUG "CH+"
    CASE ChDn
```

```
      DEBUG "CH-"
    CASE VolUp
      DEBUG "VOL+"
    CASE VolDn
      DEBUG "VOL-"
    CASE Mute
      DEBUG "MUTE"
    CASE Power
      DEBUG "POWER"
    CASE TvLast
      DEBUG "LAST"
    CASE ELSE
      DEBUG DEC remoteCode, " (unrecognized)"
  ENDSELECT

LOOP                                          ' Repeat main DO...LOOP

' -----[ Subroutine - Get_Ir_Remote_Code ]--------------------------------

' SONY TV IR remote subroutine loads the remote code into the
' remoteCode variable.

Get_Ir_Remote_Code:

  remoteCode = 0

  #SELECT $stamp
    #CASE BS2, BS2E, BS2PE
      DO                                      ' Wait for end of resting state.
        RCTIME IrDet, 1, irPulse
      LOOP UNTIL irPulse > ThresholdStart
      PULSIN IrDet, 0, irPulse                ' Get data pulses.
      IF irPulse > ThresholdPulse THEN remoteCode.BIT0 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT1 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT2 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT3 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT4 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT5 = 1
      RCTIME IrDet, 0, irPulse
      IF irPulse > ThresholdEdge  THEN remoteCode.BIT6 = 1
    #CASE BS2SX, BS2P
      DO                                      ' Wait for start pulse.
        PULSIN IrDet, 0, irPulse
      LOOP UNTIL irPulse > ThresholdStart
      PULSIN IrDet, 0, irPulse               ' Get data pulses.
      IF irPulse > ThresholdPulse THEN remoteCode.BIT0 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT1 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT2 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT3 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT4 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT5 = 1
      PULSIN IrDet, 0, irPulse
      IF irPulse > ThresholdPulse THEN remoteCode.BIT6 = 1
    #CASE #ELSE
      #ERROR "BASIC Stamp version not supported by this program."
  #ENDSELECT

  ' Map digit keys to actual values.
  IF (remoteCode < 10) THEN remoteCode = remoteCode + 1
  IF (remoteCode = 10) THEN remoteCode = 0

  RETURN
```

BASIC Stamp 2 Series Example - Multi-Digit Application

You can use the remote for keypad entry of values by replacing the DO…LOOP in IrRemoteButtonDisplay.bs2's main routine one shown below. It works for values from 0 to 65535; just type in the value on the digital keypad, then press the remote's ENTER key.

√ Add this declaration to the IrRemoteButtonDisplay.bs2's Variables section:

```
value          VAR     Word                  ' Stores multi-digit value
```

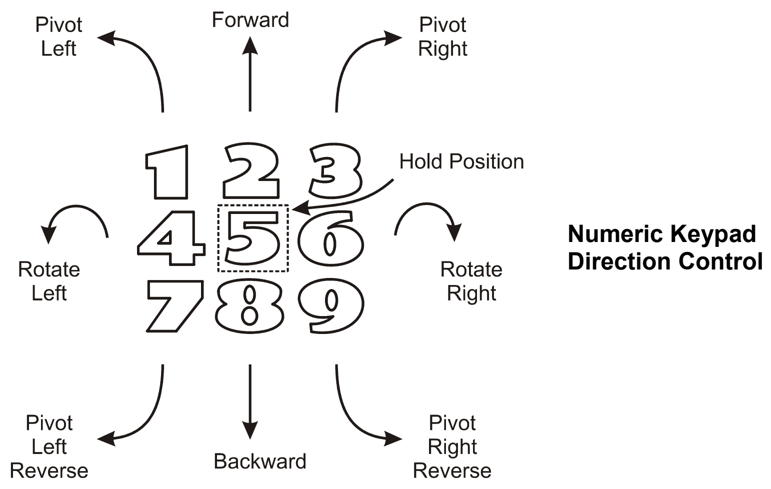
√ Replace the DO…LOOP in IrRemoteButtonDisplay.bs2's main routine with the one shown below.
√ Run the program and follow the Debug Terminal's prompts.

```
' Replace the DO...LOOP in the Main Routine with this one for multi-digit
' value acquisition (up to 65535). Value stored in value variable.

DEBUG CR, CR, "Type value from", CR, "0 to 65535,", CR,
      "then press ENTER", CR, CR

DO
  value = 0
  remoteCode = 0
  DO
    value = value * 10 + remoteCode
    DO
      GOSUB Get_Ir_Remote_Code
      IF (remoteCode > 9) AND (remoteCode <> Enter) THEN
```

```
        DEBUG "Use digit keys or ENTER", CR
        PAUSE 300
      ELSE
        DEBUG "You pressed: "
        IF remoteCode = Enter THEN
          DEBUG "Enter", CR
        ELSE
          DEBUG DEC remoteCode, CR
        ENDIF
        PAUSE 300
      ENDIF
    LOOP UNTIL (remoteCode < 10) OR (remoteCode = Enter)
  LOOP UNTIL (remoteCode = Enter)
  DEBUG ? value, CR, "Ready for next value...", CR
LOOP
```

Boe-Bot Application for the BASIC Stamp 2

This next application requires a Boe-Bot robot with a BASIC Stamp 2 module which you will be able to control by pressing and holding the numeric keys to execute the maneuvers shown in the figure. In addition, you can use CH+ = forward, CH- = backward, VOL+ = rotate right, VOL- = rotate left.



The routine below is for a Boe-Bot robot with Parallax Continuous Rotation servos. Its left servo should be connected to P13, and its right servo connected to P12. If you have Parallax PM servos, use 500 in place of 650 and 1000 in place of 850 for the PULSOUT command duration arguments.

√ Replace the DO…LOOP in the IrRemoteButtonDisplay.bs2's main routine with this one, run it, and operate the Boe-Bot with your remote. Have fun!

```
DEBUG CR, CR, "Press and hold digit", CR, "or CH+/-, VOL+/- keys", CR,
      "to control the Boe-Bot..."

DO
  GOSUB Get_Ir_Remote_Code
  SELECT remoteCode
    CASE 2, ChUp                            ' Forward
      PULSOUT 13, 850
      PULSOUT 12, 650
    CASE 4, VolDn                           ' Rotate Left
```

```
      PULSOUT 13, 650
      PULSOUT 12, 650
    CASE 6, VolUp                         ' Rotate Right
      PULSOUT 13, 850
      PULSOUT 12, 850
    CASE 8, ChDn                          ' Backward
      PULSOUT 13, 650
      PULSOUT 12, 850
    CASE 1                                ' Pivot Fwd-left
      PULSOUT 13, 750
      PULSOUT 12, 650
    CASE 3                                ' Pivot Fwd-right
      PULSOUT 13, 850
      PULSOUT 12, 750
    CASE 7                                ' Pivot back-left
      PULSOUT 13, 750
      PULSOUT 12, 850
    CASE 9                                ' Pivot back-right
      PULSOUT 13, 650
      PULSOUT 12, 750
    CASE ELSE                             ' Hold position
      PULSOUT 13, 750
      PULSOUT 12, 750
  ENDSELECT
LOOP
```

More Resources

These resources are available from www.parallax.com.

Lindsay, Andy. *IR Remote for the Boe-Bot*, Student Guide, Version 1.0, California: Parallax, Inc., 2004.

This book is discussed on the first page of this package insert.

Williams, Jon. *The Nuts and Volts of the BASIC Stamps*, Volume 3, California: Parallax, Inc, 2003.

Column #76: *Control from the Couch* introduces capturing and decoding SONY TV IR control signals with the BASIC Stamp 2SX (or 2p).