PARALLAX INC.
599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Toll-Free: (USA & CAN) (888) 512-1024
Fax: (916) 624-8003

Toll Free Tech Support:
(USA & CAN) (888) 99-STAMP
Email: support@parallax.com
Web: www.ScribblerRobot.com

Hacker's Hints for the Scribbler Robot

Programming Tips for Experienced BASIC Stamp Enthusiasts

Advanced users who want direct control of the Scribbler's motors, sensors, speaker, and lights have the option of writing PBASIC programs with the BASIC Stamp Editor. If you are familiar with BASIC Stamp programming, these hints will give you the inside information you need to write programs that control each of these systems:

- User-programmable LEDs – 3 programmable light emitting diodes can be programmed to act as a user interface, or to provide feedback for debugging.
- Sound Generation – An onboard speaker with amplifier can generate tones and musical notes.
- Light Sensor – 3 forward looking narrow angle light sensors enable light level measurement and navigation towards or away from light sources.
- Object Detection – Dual infrared emitters with a single center-mounted sensor detect objects, enabling collision-free navigation of indoor environments.
- Line Detectors – 2 downward pointing infrared line detector pairs sense black lines on a white background.
- Drive Train - Dual high ground clearance differentially steered drive wheels are driven by a 127:1 reduced DC motor. A bidirectional motor controller provides programmable speed and duration set points.
- Stall Sensor – A motor stall sensor provides a redundant collision sensor to the obstacle detector.

If you have no previous experience with PBASIC, go to www.ScribblerRobot.com, and download the free *Scribbler PBASIC Programming Guide* and the *BASIC Stamp Syntax and Reference Manual.* If you need to restore the original factory Scribbler Demo program, it can be found on the Scribbler software CD and on the website.

I/O Declarations

The Scribbler utilizes all 16 of the BASIC Stamp input/output pins. The pin declarations below describe what is connected to each I/O pin. These pin names are used in all of the code segment examples below. Remember, unless you intentionally initialize a pin with `HIGH` or `LOW`, all I/O pins default to input.

```
LightRight    PIN 0   ' right light sensor
LightCenter   PIN 1   ' center light sensor
LightLeft     PIN 2   ' left light sensor
LineEnable    PIN 3   ' power to line follower IR emitters
LineRight     PIN 4   ' right line follower IR detector
LineLeft      PIN 5   ' left line follower IR detector
ObsRx         PIN 6   ' obstacle sensor IR detector
Stall         PIN 7   ' stall sensor
LedRight      PIN 8   ' right green LED
LedCenter     PIN 9   ' center green
LedLeft       PIN 10  ' left green LED
Speaker       PIN 11  ' speaker
MotorRight    PIN 12  ' right wheel motor
MotorLeft     PIN 13  ' left wheel motor
ObsTxRight    PIN 14  ' right obstacle sensor IR emitter
ObsTxLeft     PIN 15  ' left obstacle sensor IR emitter
```

User LEDs – UserLEDs.bs2
There are three green user-controlled light emitting diodes on the rear panel of the robot. These serve as sensor indicator lights in several of the Demo modes. The user LEDs can be turned on and off in any combination by your own code. You can design a user interface with them, to display the value of sensors or the program status. Or, you can just make cool flashing patterns.

To turn on a user LED, set its pin to `HIGH`. To turn it off, set it to `LOW`.

```
  HIGH LedLeft       'Turn on each LED one at a time, 500 ms apart
  PAUSE 500
  HIGH LedCenter
  PAUSE 500
  HIGH LedRight
  PAUSE 500

  LOW LedLeft       ' Turn them all off virtually at once
  LOW LedCenter
  LOW LedRight
```

Speaker – Speaker.bs2
The Scribbler can make a wide range of sounds on its speaker, both single notes and two notes played together. To play notes, simply send a `FREQOUT` command to the Speaker pin:

```
  'Play the single note A = 1760 Hz for 500 ms
  FREQOUT Speaker, 500, 1760
```

In this example, the first number after specifying the Speaker pin is the duration, in ms, of the note to be played. The second number is the frequency of the note in Hz. If you include a comma and a third number at the end, it will play a second frequency at the same time:

```
'Play dual notes A = 1760 Hz, E = 1319 Hz for 500 ms
  FREQOUT Speaker, 500, 1760, 1319
```

Detailed instructions for programming the BASIC Stamp 2 to play music can be found in *What's a Microcontroller?*

Light Sensors – LightSensors.bs2

The Scribbler has three photoresistors on front of the robot, deeply set to narrow their focus. The center sensor looks straight ahead, and the other two sensors look 30 degrees to each side. This positioning allows the Scribbler use the photoresistors to navigate towards or away from light.

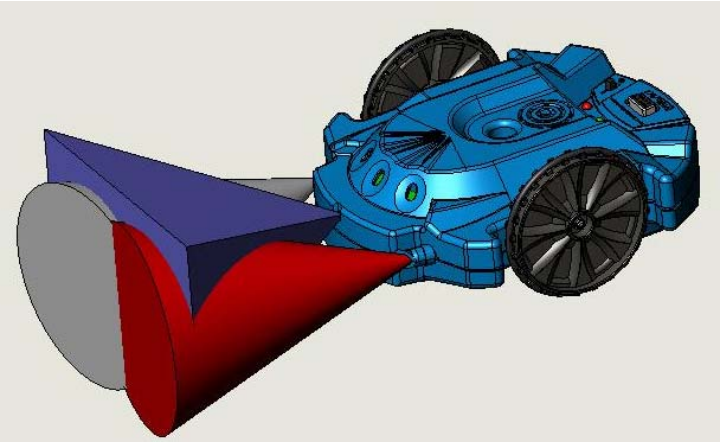
The code segment below will activate the photoresistors and record the level of light detected by each sensor as a number. These values will be stored in the variables `light_left_value`, `light_center_value`, and `light_right_value`. Bright light will return a low value, below 100 in a sunlit room. Very dim light, such as in a dark room or under a box, might return values up to 10,000.

```
' Set the pins high to charge the capacitors
HIGH LightLeft
HIGH LightCenter
HIGH LightRight

' Wait while the capacitor charges
PAUSE 3

' Time how long it takes each one to discharge
RCTIME LightLeft, 1, light_left_value
RCTIME LightCenter, 1, light_center_value
RCTIME LightRight, 1, light_right_value
```

No two photoresistors are likely to return the exact same values given the very same light. Therefore, your program should either calibrate each sensor or only rely on imprecise comparisons between the sensors. For more examples using the `RCTIME` command with this type of photoresistor, see *What's a Microcontroller?* and *Robotics with the Boe-Bot.*

Object Detection – ObjectDetect.bs2

Several of the Scribbler Demo modes rely on the object detection system. With it, the Scribbler can sense an object in front of it and detect whether the object is on the right or left-hand side of the robot's field of vision.



The two IR LEDs on the front of the Scribbler emit modulated infrared light. The IR detector between them is tuned to see this light reflected off of objects.

In this sample code, the two IR LEDs alternate emitting. This way if the IR detector senses reflections only when the right IR LED is emitting, you know the object is on the right. Likewise, it uses the left IR LED to check whether an object is on the left. If it can sense an object when either IR LED is emitting, it knows the object is in front.
The object detection system code uses the `FREQOUT` command to make the IR LED's emit light modulated to the desired frequency. The command

```
FREQOUT ObsTxRight, 1, 38500
```

sends a signal to the right IR LED pin, for 1 ms, at a frequency of 38500 Hz.

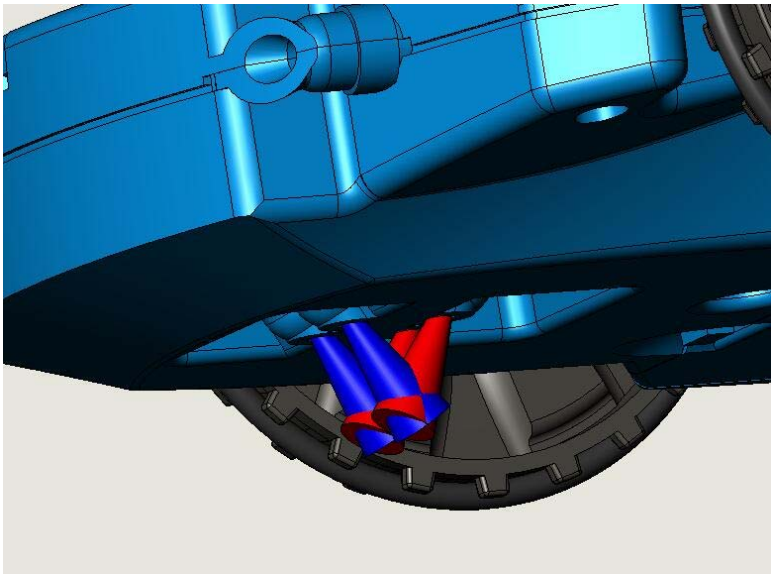
If this light is reflected off an object and seen by the IR detector, the bit variable `obstacle_right` will be 0; if there is no obstacle it will be 1. Then, the right IR LED is turned off and the process is repeated by the left IR LED.

```
' Output IR light modulated at the right frequency
FREQOUT ObsTxRight, 1, 38500

' Check if reflected light was detected
object_right = ObsRx

' Turn off the right-side IR LED
LOW ObsTxRight

' Do the same thing for the left-hand side
FREQOUT ObsTxLeft, 1, 38500
object_left = ObsRx
LOW ObsTxLeft
```

The Scribbler can be programmed to avoid obstacles, find objects, or follow something that is moving, such as your hand or another Scribbler. Examples can be found in *Robotics with the Boe-Bot.*

Line Sensors – LineSensors.bs2

Like the object detection system, the line following system relies on infrared light. The pair of line sensors on the bottom of the Scribbler allows it to see a dark line on a light background.



Each line sensor consists of an IR LED and an IR detector. While the IR LED emits infrared light, the IR detector measures the amount of infrared light being reflected off the surface under the Scribbler. A lot of reflected light means the surface is white below the detector; only a little reflected light means the surface looks black. This is how the Scribbler "sees" the lines that it follows. You could also use the line sensors to recognize other patterns, such as finding a black patch on the floor.

The line sensor system works by lighting both of its IR LEDs at the same time. Each beam of infrared light is precisely aimed to bounce of the surface under the Scribbler, and directly hit its own IR detector. This light does not have to be modulated, so you do not need to use the `FREQOUT` command. Instead, turn on both IR emitters at the same time with the command `HIGH LineEnable`. Then, check the status of each IR detector's pin. If a pin returns a 1, that detector is seeing the surface as black. If it returns a 0, it sees the surface as white.

```
'Turn on the Line Sensor IR LEDs
HIGH LineEnable

IF (LineRight = 1) THEN
 DEBUG "Right= Black"
ELSE
 DEBUG "Right= White"
```

```
  ENDIF

  IF (LineLeft = 1) THEN
   DEBUG "Left= Black"
  ELSE
   DEBUG "Left= White"
  ENDIF
```

Hints:

- The line sensors will give their best guesses no matter what color surface is beneath the Scribbler. Light colors will be seen as white, and dark colors as black, but this isn't as reliable as using true black and white.
- Line following works best when your track is placed on a smooth, hard, flat surface.
- The line sensors are precisely positioned. If you increase or decrease the distance between the Scribbler bottom and the surface it is on, the line sensors will be out of focus and will not give accurate results.
- To conserve your battery power, the line sensor IR LEDs should be turned off with the command `LOW LineEnable` when you are not using them.

Drive Motors – Motors.bs2

The Scribbler moves when velocity instructions are sent to both the right and left DC motors. These motors are controlled by Pulse Code Modulation (PCM) signals using the `PULSOUT` command.

```
PULSOUT Pin, Duration
```

Pin will be either `MotorRight` or `MotorLeft`. *`Duration`* defines the signal pulse width, in 2 microsecond units. The DC motors will accept a Period value between 1000 and 3000.

A *`Duration`* value of 1000 will make a wheel go full speed in reverse, 2000 will make it stop, and 3000 will make it go full speed forward. You can also send any *`Duration`* value in between for intermediate velocities. Once a wheel has been given a velocity command, it will keep turning at that velocity until it gets another command.

```
  'Drive straight forward at half speed
  PULSOUT MotorRight, 2500
  PULSOUT MotorLeft, 2500
```

The DC motors also have a special function that lets you control the Scribbler's movements precisely. Any velocity instruction can be followed with a `PAUSE` of 1 ms, and then a second `PULSOUT` command that dictates how long the velocity command should execute.

This duration instruction also uses *`Duration`* values in the 1000 to 3000 range, with 1000 producing a duration of zero seconds, and 3000 producing a duration of five seconds. Any intermediate *`Duration`* value will return an intermediate duration; for instance, a *`Duration`* value of 1400 will make the wheel turn for one second. Once a motor has rotated for its specified duration, the motor will stop and wait for the next command.

```
  ' Drive in an arc to the right for 3 seconds

  PULSOUT MotorRight, 2400
  PAUSE 1
  PULSOUT MotorRight, 2200

  PULSOUT MotorLeft, 2600
  PAUSE 1
```

```
PULSOUT MotorLeft, 2200
```

These DC motors use PCM, where hobby servos use PWM (Pulse Width Modulation).

The Demo programs and GUI use calibrated motor commands and light sensor values. The calibration values and checksum are stored in the first 16 bytes of the EEPROM. If you write a PBASIC that uses all of the available EEPROM you may accidentally overwrite these values and lose the calibration. So, if you are writing large PBASIC programs, be sure to read the first 16 EEPROM values before you load your code and restore them before you use the Demo program or the GUI again.

Stall Sensor – StallSensor.bs2
Object sensors are never perfect, so it's always good for a roaming robot to have a backup plan. The stall sensor detects when the motors are running but the wheels have stopped turning because the Scribbler has hit an object.

To use the stall sensor, simply check the stall sensor pin as shown in the code segment below. The stall sensor is connected to both DC motors, so it can check both wheels for a stall at the same time by monitoring a single pin. If either or both wheels are stalled, the stall sensor pin will return a 1. If neither wheel is stalled, it will return a 0.

```
IF (Stall = 1) THEN
  DEBUG "Stall"
ELSE
  DEBUG "No Stall"
ENDIF
```

The stall sensor can be used as a backup while roaming with object detection or light seeking, or it can be used on its own to make a robot that just bumps around the room.