

# Prop-1 Programming Basics

## Team EFX-TEK

teamefx@efx-tek.com www.efx-tek.com

## Why Use a Programmable Controller?

- No off-the-shelf product exists that meets the requirements of your application
- Off-the-shelf product is price-prohibitive
- Control requirement will evolve
- You're an OEM with several products and want to simplify control inventory
- Custom control = Unique product

### Microcontroller Essentials

- A microcontroller is a "computer on a chip"
- Handles Input, Processing (instructions), and Output
- Flexible I/O (Input-Output) structure
- Advanced microcontrollers offer simple and sophisticated I/O control

### The BASIC Stamp Microcontroller

- Single-Board-Computer
- Handles Input, Processing (instructions), and Output
- Flexible I/O (Input-Output) structure
- Simple and Sophisticated I/O commands
- Program storage is non-volatile
  - will not be lost when power removed
- Programming Language: PBASIC
  - specialized, yet easy-to-use variant of BASIC

### The BASIC Stamp Microcontroller

### BASIC

Beginner's All-purpose Symbolic Instruction Code

### The BASIC Stamp Microcontroller

Parallax Beginner's All-purpose Symbolic Instruction Code

# **BASIC Stamp 1 Tech Specs**

Speed (instructions per second)	~2,000
Input / Output Connections	8
RAM Variables (bytes)	14 + 2
Program Memory (bytes)	256
Program Length (lines of code)	~80
PBASIC 1.0 Commands	32
Programming Connection	Serial 4.8k

### Prop-1 Controller (#31101)

- 6-24 vdc input
- TTL I/O, and high-current (Vin) outputs
- Program with BASIC Stamp Editor, v2.1+



### Parallax BASIC Stamp Editor

- Program all BASIC Stamp modules
- Win98, Win2K, WinNT, WinXP
- Serial or USB Interface for programming (Prop-1 requires BS1 Serial Adapter, #27111)



BS1 Serial Adapter (#27111)

## Parallax BASIC Stamp Editor

BASIC Stamp - C:\Documents and Sett	ings\Ac	iministrator\My Documents\EFX-TEK\Products\31101 Prop-1 Controller\Code\Spooky_Eyes.BS1
<u>File E</u> dit <u>D</u> irective <u>R</u> un <u>H</u> elp		
□ ☞ 🖬 🗖 🖨 👗 🖻 🛱		[11] ····· 《 《 《 《 《 · ③ ④ ④ □ III III Ⅰ 图 ▶ 象   象 象 象 》
E:\\31101 Prop-1 Controller\Code	Spooky	_Eyes.BS1
🖻 🛅 Dvx100a 🛛 🔥	1	· A
	2	
EFX-TEK	3	' File Spooky Eyes.BS1
🔁 Artwork	4	' Purpose Random "eyes" controller for Halloween haunts
	5	' Author Jon Williams EFX-TEK
Documentation Templates     MySpace	6	' E-mail jwilliams@efx-tek.com
Products	7	' Started
in oducts	8	Updated 01 JUN 2006
	9	
- Con uMP3	10	{\$STAMP BS1}
Control		(opinie poi)
+ 🗁 31198 Prop Pot	11	(\$PBASIC 1.0)
🗉 🦰 31199 Prop-1 Trainer	12	
🗄 🦰 31204 RC-4	13	
🕂 🦳 31216 DC-16 🛛 💆	14	
< >	15	
Candle_On.BS1	16	'[ Program Description ]
Candles.BS1	17	
Candles-8x.BS1	18	' While triggered, this program will randomly select an set of "eyes" and
Crate-Beast.BS1	19	' bring them on gradually, hold them on for a moment, then extinguish
DCthrasher.bs1	20	' them. The process will complete as long as the trigger is present.
Dual-Sequencer.BS1	21	
Dual-Sequencer-Advanced.BS1	22	
Dual-Sequencer-MaxOuts.BS1	23	'[ Revision History ]
Exorcist_Room.BS1	24	[ NOVIDION NIDEOLY ]
Expander.BS1	25	' 15 JAN 2005 - First version; tested on Parallax BS1 Project Board
Flash.BS1	26	13 JAN 2003 - First Version, tested on Parallax BSI Project Board
Hangman.BS1 HC1 Program Pad.BS1		
Jack.BS1	27	
MoodSkull.bs1	28	'[ I/O Definitions ]
MoodSkulla.bs1	29	
MoodSkullb.bs1		SYMBOL Trigger = PIN7
MoodSkullc.bs1	31	
Multi-Sequence.BS1	32	
MultiSequencer.bs1	33	'[ Constants ]
OE-8X Ping-Pong.BS1	34	
OE-8X Test.BS1	35	SYMBOL ISON = 0 'active low input
or-and-demo.bs1	36	SYMBOL ISOFF = 1
Police_Lights.BS1	37	
Police_Lights_EZ.BS1		SYMBOL EyePorts = 7 ' # eye control pins
POT_Demo.BS1	39	
BASIC Stamp files (*.bs1;*.bas;*.bs2;*.bse;*.bsx,	<	
8:28 Modified INS		

8:28 Modified INS

## Prop-1 Trainer (#31199)

- Training / Experimenting / Prop UI
- 6 LEDS, Trigger button, POT circuit (requires simple ULN2803 modification/replacement to use POT circuit)



## Prop-1 Variables (Internal Names)

Word Name	Byte Name	Bit Name	Special Notes	
PORT	PINS	PIN0 – PIN7	I/O pins; bit-addressable	
	DIRS	DIR0 – DIR7	I/O pins direction; bit-addressable	
WO	B0	BITO – BIT7	Bit-addressable	
	B1	BIT8 – BIT15	Bit-addressable	
W1	B2			
	B3			
W2	B4			
	B5			
W3	B6			
	B7			
W4	B8			
	B9			
W5	B10			
	B11			
W6	B12		Used as stack for GOSUB-RETURN	
	<b>B13</b>			

**SYMBOL** Name = [Variable | Value] **SYMBOL** is used to give meaningful names to I/O pins, to constant values, and to variables.

- SYMBOL Pir = PIN6
- SYMBOL Active = 1
- SYMBOL pntr = B2

### HIGH Pin

**HIGH** is used to make an I/O pin an output and set it to a high (+5 vdc) state.

HIGH 0

Better example:

HIGH Eyes

' eyes on

### LOW Pin

# **LOW** is used to make an I/O pin an output and set it to a low (O vdc) state.

LOW 0

Better example:

LOW Eyes

' turn off

### PAUSE Period

**PAUSE** is used to suspend program operation for the specified period (in milliseconds; 1/1000 second). After the **PAUSE**, program operation is automatically resumed.

PAUSE 1000

' hold for 1 second

### GOTO Label

**GOTO** is used to redirect the program to the specified program label.

GOTO Main

' back to Main

## Prop-1 Example (Simple Flasher)

CIVIDAT

PAUSE 500

PAUSE 500

- GOTO Main

LOW Led

SIMBOL Led	- 0	LED IS CONNECTED TO FO	
► Main:			
HIGH Led		' turn LED on	

- ' hold for 1/2 second
- ' turn LED off
- ' hold for 1/2 second
- ' back to Main

### IF Condition THEN Label

**IF-THEN** is used to redirect the program to the a specified program label if the <u>condition</u> evaluates as True.

Main:

```
IF PIN6 = 0 THEN Main
```

Better example:

```
IF Pir = IsOff THEN Main
```

## Prop-1 Example (Triggered Flasher)

SYMBOL	Pir	=	PIN6
SYMBOL	Led	Η	0
SYMBOT.	TsOff	=	0

#### Main:

IF Pir = IsOff THEN Main	T	wait :	for	PIR	activity
HIGH Led	T	turn :	LED	on	
PAUSE 500	T	hold	for	1/2	second
LOW Led	T	turn 3	LED	off	
PAUSE 500	T	hold	for	1/2	second
GOTO Main	T	back	to M	lain	

### **Prop-1 Example** (Triggered Event with Pelay)

SYMBOL	MatSw	= PIN6
SYMBOL	Valve	= 0
SYMBOL	No	= 0

#### Main:

IF MatSw = No THEN Main ' wait for "victim" **PAUSE 3000** HIGH Valve **PAUSE 5000** LOW Valve **PAUSE 20000** GOTO Main

- ' 3 second pre-delay
- ' lift prop
- ' hold for 5 seconds
- ' retract prop
- ' 20 second post-delay
- ' back to Main

### Prop-1 Programming (Advanced)

FOR Var = StartVal TO EndVal NEXT

**FOR-NEXT** is used to repeat a section of code for a specific number of iterations.

FOR cycles = 1 TO 10
 ' statement(s)
NEXT

### Prop-1 Example (Triggered Chaser)

SYMBOL MatSw = PIN6 SYMBOL No = 0SYMBOL pinNum = B2

#### Main:

IF MatSw = No THEN Main ' wait for "victim" FOR pinNum = 0 TO 5 ' cycle through pins HIGH pinNum PAUSE 100 LOW pinNum NEXT GOTO Main

' turn selected pin on ' hold for 0.1 second ' turn selected pin off

' back to Main

### Prop-1 Programming (Advanced)

RANDOM Variable

**RANDOM** is used to generate the next pseudo-random value in variable.

RANDOM timer

### Prop-1 Example (Random Pre-Event Delay)

SYMBOL	MatSw	= <b>PIN6</b>
SYMBOL	Valve	= 0
SYMBOL	No	= 0
SYMBOL	timer	= W1
SYMBOL	delay	= W2

#### Main:

#### RANDOM timer

```
IF MatSw = No THEN Main ' wait for "victim"
delay = delay * 1000
PAUSE delay
HIGH Valve
PAUSE 5000
LOW Valve
PAUSE 20000
GOTO Main
```

- ' stir random generator
- delay = timer // 5 + 1 ' create delay, 1 to 5 seconds
  - ' convert to milliseconds
  - ' hold for random delay
  - ' open solenoid to lift prop
  - ' hold for 5 seconds
  - ' retract prop
  - ' 20 second post-delay
  - ' back to Main

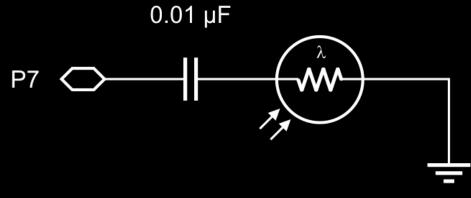
### Prop-1 Programming (Advanced)

### POT Pin, Scale, Variable

**POT** is used to read a variable resistance (e.g., potentiometer, photo-resistor, etc.). Scale value derived from Editor utility.

POT LSense, 135, lightLevel

Light level circuit:



### Prop-1 Example (Light-Activated Chaser)

SYMBOL LSense = 7SYMBOL level0 = B2 ' initial light level SYMBOL level1 = B3SYMBOL pinNum = B4

```
Setup:
 POT LSense, 135, level0 ' get initial light level
```

#### Main:

```
POT LSense, 135, level1 ' get current light level
IF level1 > level0 THEN Main ' wait for light drop
FOR pinNum = 0 TO 6 ' cycle through pins
 HIGH pinNum
 PAUSE 100
 LOW pinNum
NEXT
GOTO Main
```

- ' light level sensor

  - ' current light level

```
' LED on
' hold 0.1 second
 ' LED off
```

```
' back to Main
```

### Prop-1 Programming (Advanced)

### PULSOUT Pin, Period

**PULSOUT** is used to generate a pulse on an I/O pin. The output state will be inverted for the specified period (in 10  $\mu$ s units).

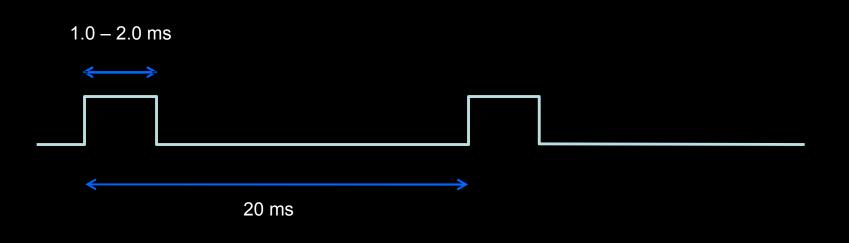
PULSOUT Servo, 150 ' 1.5 ms pulse (center servo)

# Hobby Servos



### Servo Control

- 5 vdc power input (nominal)
- 1.0 ms to 2.0 ms (typical) control pulse
- Refresh every 20 ms



### Prop-1 Example (Servo Direct)

```
SYMBOL Servo = 0
SYMBOL pos = B2 ' servo position
SYMBOL delay = B3
Setup:
DIRS = %00000001 ' P0 is output, all others inputs
Main:
FOR pos = 100 TO 200 STEP 2 ' sweep left-to-right
FOR delay = 1 TO 3 ' hold position
PULSOUT Servo, pos ' refresh servo
PAUSE 20
NEXT
NEXT
GOTO Main ' back to Main
```

### Prop-1 Programming (Advanced)

### **SEROUT** *Pin, Baudmode, (Data)* **SEROUT** is used to transmit asynchronous serial data

on an I/O pin at the specified baud rate and mode.

SEROUT Lcd, T2400, ("Props are FUN!")

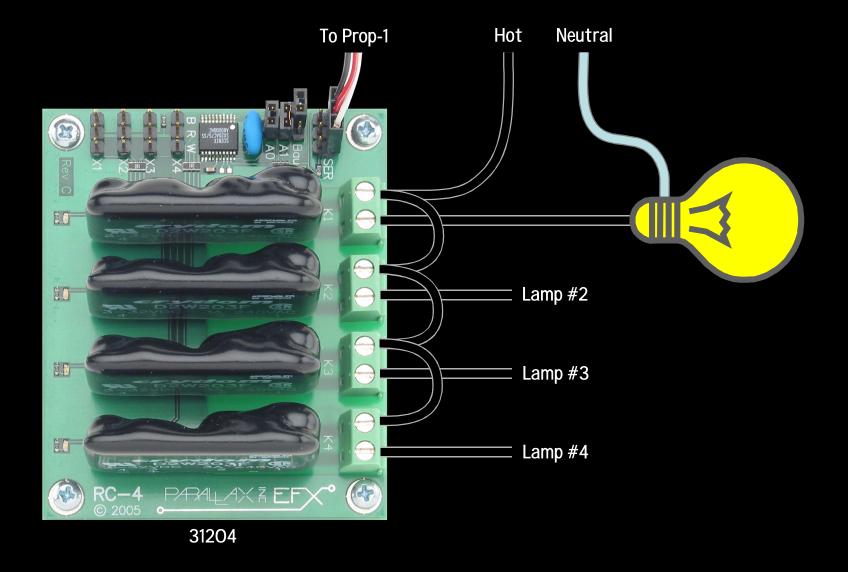
### Prop-1 Example (RC-4 Control)

```
SYMBOL MatSw=PIN6SYMBOL TX=5SYMBOL No=0SYMBOL idx=B2SYMBOL lights=B3SYMBOL timer=W2SYMBOL delay=W3
```

```
Main:
```

```
FOR idx = 1 TO 3
  RANDOM timer ' stir random generator
NEXT
SEROUT TX, OT2400, ("!RC4", %11, "X")
IF MatSw = No THEN Main ' wait for "victim"
lights = timer // 16 ' randomize lights
SEROUT TX, OT2400, ("!RC4", %11, "S", lights)
delay = timer // 201 + 50 ' create 50 to 250 ms delay
PAUSE delay ' hold lights
GOTO Main ' back to Main
```

# Prop-1 Example (RC-4 Control)



### Prop-1 Programming (Advanced)

### GOSUB Label ... RETURN

**GOSUB** is used to redirect the program to the specified code section that ends with **RETURN**, which sends the program back to the line that follows the calling **GOSUB**.

tix = 35	Ţ	timer =	3.5 seconds
GOSUB Run_Timer	Ţ	run the	timer

Remember... GOSUB uses W6, so you can't use this variable (or B12 or B13) in your program.

### **Prop-1** Example (Timer Subroutine)

```
SYMBOL Led = 0
SYMBOL tix = B2
Main:
  HIGH Led
                                ' Led on
  tix = 23
  GOSUB Run Timer
LOW Led
                               ' Led off
  tix = 7
  GOSUB Run Timer
  GOTO Main
Run Timer:
  IF tix = 0 THEN Timer Done ' check for end of timer
    PAUSE 100
    tix = tix - 1
    GOTO Run Timer
Timer Done:
  RETURN
```

```
' set timer for 2.3 seconds
' start the timer
' set timer for 0.7 seconds
' start the timer
```

```
' hold for 1 tic (0.1 secs)
' update tix count
```

```
' re-check for end of timer
```

```
' go back to main program
```

### **Prop-1** Example (Timer Subroutine)

```
SYMBOL Led = 0
 SYMBOL tix = B2
 Main:
   HIGH Led
   tix = 23
   GOSUB Run Timer
   LOW Led
   tix = 7
   GOSUB Run Timer -
  - GOTO Main
> Run Timer:
   IF tix = 0 THEN Timer Done ' check for end of timer
     PAUSE 100
     tix = tix - 1
     GOTO Run Timer
 Timer Done:
   RETURN
```

```
' Led on
' set timer for 2.3 seconds
' start the timer
' Led off
' set timer for 0.7 seconds
' start the timer
```

```
' hold for 1 tic (0.1 secs)
' update tix count
' re-check for end of timer
```

```
' go back to main program
```

### Prop-1 Programming – Review

### Essentials

SYMBOL Name = [Variable | Value] HIGH Pin LOW Pin PAUSE Period GOTO Label IF Condition THEN Label FOR Variable = StartVal TO EndVal ... NEXT

### Advanced

RANDOM Variable POT Pin, Scale, Variable PULSOUT Pin, Period SEROUT Pin, Baudmode, (Data) GOSUB Label ... RETURN

## Prop-1 Programming – Going Further

### Additional Instructions

DEBUG Data EEPROM {Location, }(Value, Value, ...) READ Location, Variable PWM Pin, Duty, Cycles TOGGLE Pin SERIN Pin, Baudmode, {(Qualifier, ...)}, {#}Variable, ...

### **Advanced Programming Techniques**

Learn to use **DIRS** and **PINS** for I/O setup and control Master the // (modulus) operator Learn to use **\*\*** to multiply by fractional values (less than zero)