Prop-1 Programming Basics

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

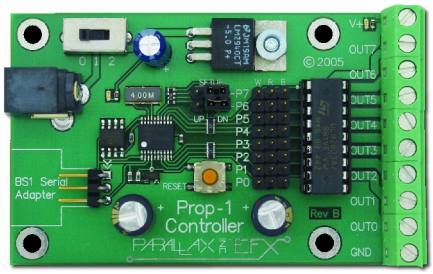
Parallax
Beginner's
All-purpose
Symbolic
Instruction
Code

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

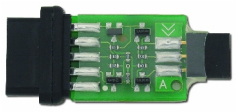
Prop-1 Controller (#31101)

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

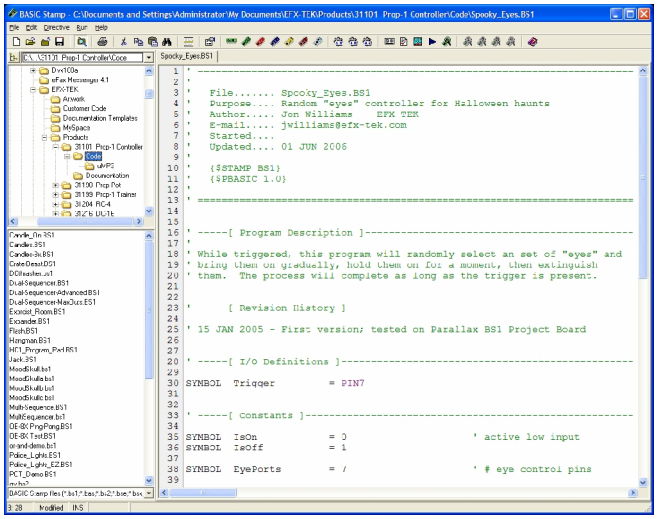


Parallax BASIC Stamp Editor

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



Parallax BASIC Stamp Editor



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
W0	B0	BIT0 – 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
	B13		

Prop-1 Programming

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
```

Prop-1 Programming

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
```

Prop-1 Programming

LOW *Pin*

LOW is used to make an I/O pin an output and set it to a low (0 vdc) state.

```
LOW 0
```

Better example:

```
LOW Eyes                     ' turn off
```

Prop-1 Programming

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
```

Prop-1 Programming

GOTO ***Label***

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

```
GOTO Main                   ' back to Main
```

Prop-1 Example (Simple Flasher)

```
SYMBOL Led     = 0              ' LED is connected to P0

Main:
  HIGH Led                      ' turn LED on
  PAUSE 500                     ' hold for 1/2 second
  LOW Led                       ' turn LED off
  PAUSE 500                     ' hold for 1/2 second
  GOTO Main                     ' back to Main
```

Prop-1 Programming

IF *Condition* THEN *Label*

IF-THEN is used to redirect the program to the a specified program label if the *condition* 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      = 0
SYMBOL IsOff    = 0

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

Prop-1 Example (Triggered Event with Delay)

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

Main:
  IF MatSw = No THEN Main       ' wait for "victim"
  PAUSE 3000                    ' 3 second pre-delay
  HIGH Valve                    ' lift prop
  PAUSE 5000                    ' hold for 5 seconds
  LOW Valve                     ' retract prop
  PAUSE 20000                   ' 20 second post-delay
  GOTO Main                     ' 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      = 0
SYMBOL pinNum  = B2

Main:
  IF MatSw = No THEN Main        ' wait for "victim"
  FOR pinNum = 0 TO 5            ' cycle through pins
    HIGH pinNum                  ' turn selected pin on
    PAUSE 100                    ' hold for 0.1 second
    LOW pinNum                   ' turn selected pin off
  NEXT
  GOTO Main                      ' 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    = PIN6
SYMBOL Valve    = 0
SYMBOL No       = 0
SYMBOL timer    = W1
SYMBOL delay    = W2

Main:
  RANDOM timer                  ' stir random generator
  IF MatSw = No THEN Main       ' wait for "victim"
  delay = timer // 5 + 1        ' create delay, 1 to 5 seconds
  delay = delay * 1000          ' convert to milliseconds
  PAUSE delay                   ' hold for random delay
  HIGH Valve                    ' open solenoid to lift prop
  PAUSE 5000                    ' hold for 5 seconds
  LOW Valve                     ' retract prop
  PAUSE 20000                   ' 20 second post-delay
  GOTO Main                     ' back to Main
```

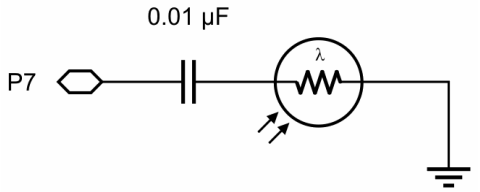
Prop-1 Programming (Advanced)

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  = 7              ' light level sensor
SYMBOL level0  = B2             ' initial light level
SYMBOL level1  = B3             ' current light level
SYMBOL pinNum  = B4

Setup:
  POT LSense, 135, level0       ' get initial light level
  level0 = level0 * 3 / 4       ' adjust to 75%

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                 ' LED on
    PAUSE 100                   ' hold 0.1 second
    LOW pinNum                  ' LED off
  NEXT
  GOTO Main                     ' back to Main
```

Prop-1 Programming (Advanced)

PULSOUT *Pin, Period*

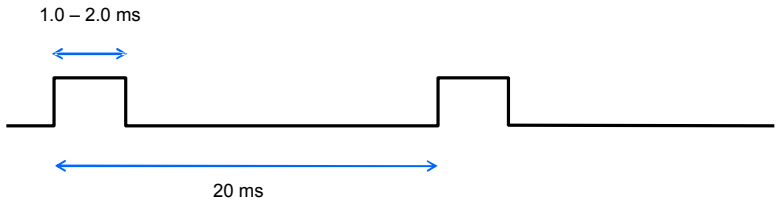
PULSOUT is used to generate a pulse on an I/O pin. The output state will be inverted for the specified period (in 10 µs units).

```
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.

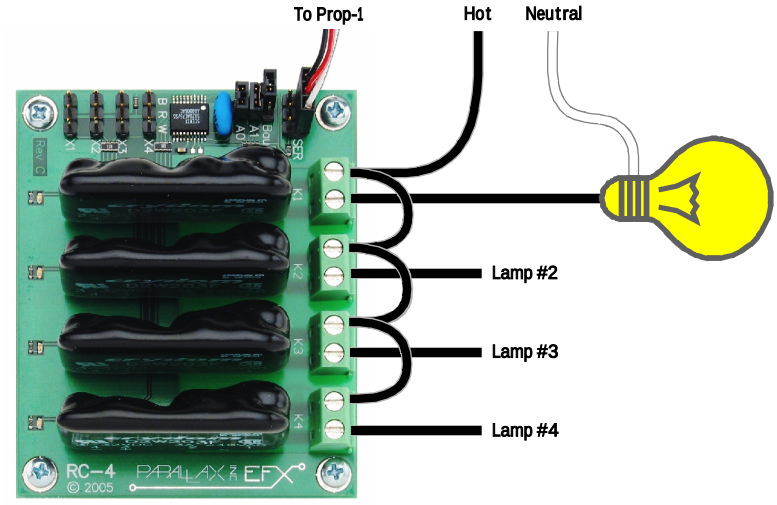
```
SEROUT Lcd, T2400, ("Props are FUN!")
```

Prop-1 Example (RC-4 Control)

```
SYMBOL MatSw  = PIN6
SYMBOL TX     = 5
SYMBOL No     = 0
SYMBOL idx    = B2
SYMBOL lights = B3
SYMBOL timer  = W2
SYMBOL 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, #31204)



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                        ' timer = 3.5 seconds
GOSUB Run_Timer                 ' 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                          ' set timer for 2.3 seconds
  GOSUB Run_Timer                   ' start the timer
  LOW Led                           ' Led off
  tix = 7                           ' set timer for 0.7 seconds
  GOSUB Run_Timer                   ' start the timer
  GOTO Main

Run_Timer:
  IF tix = 0 THEN Timer_Done        ' check for end of timer
    PAUSE 100                       ' hold for 1 tic (0.1 secs)
    tix = tix - 1                   ' update tix count
    GOTO Run_Timer                  ' re-check for end of timer

Timer_Done:
  RETURN                            ' 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

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

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)