



## **Prop-2 Programming Basics**

### **Team EFX**

teamefx@efx-tek.com

www.efx-tek.com

## **Why Use a Microcontroller?**

- 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

## Prop-1/Prop-2 Tech Specs

	Prop-1	Prop-2
<i>Speed (instructions / second)</i>	~2,000	~4,000
<i>Input / Output Connections</i>	8	16 + Serial
<i>RAM Variables (bytes)</i>	14* + 2	26 + 6
<i>Program Memory (bytes)</i>	256	2048
<i>Program Length (instructions)</i>	~80	~500
<i>PBASIC Commands</i>	32	42
<i>Programming Connection</i>	Serial 4.8k	Serial 9.6k



## MacBS2 Editor (3<sup>rd</sup> Party Software)



## Prop-2 Programming

<b>Name</b>	<b>PIN</b>	<b>PinNum</b>
<b>Name</b>	<b>CON</b>	<b>SomeValue</b>
<b>Name</b>	<b>VAR</b>	<b>Type</b>

**PIN**, **CON**, and **VAR** are used to give meaningful names to I/O pins, to constant values, and to variables.

<b>Pir</b>	<b>PIN</b>	<b>14</b>
<b>IsOn</b>	<b>CON</b>	<b>1</b>
<b>pntr</b>	<b>VAR</b>	<b>Byte</b>

## Prop-2 Variable Types

Variables used to store values that change. All variables are cleared to zero on reset and are lost when power is removed.

**Bit** : (0 to 1)

**Nib** : 4 bits (0 to 15)

**Byte** : 8 bits, 2 nibs (0 to 255)

**Word** : 16 bits, 4 nibs, 2 bytes (0 to 65,535)

## Prop-2 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-2 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-2 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-2 Programming

### **GOTO *Label***

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

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

## Prop-2 Example

```
Led    PIN    8           ' LED is connected to P8

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-2 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 (Pir = IsOff) THEN Main
```

When using PBASIC 2.5, the Prop-2 also supports

**IF-THEN-ELSE-ENDIF.**

## Prop-2 Example (Triggered Flasher)

```
Pir    PIN    14
Led    PIN    8
IsOff  CON    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-2 Example (Triggered Event & Delay)

```
MatSw  PIN    14
Valve  PIN    8
No     CON    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-2 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-2 Example (Triggered Chaser)

```
MatSw  PIN    14
No      CON    0
pinNum VAR    Nib
```

Main:

```
IF (MatSw = No) THEN Main      ' wait for "victim"
FOR pinNum = 8 TO 13          ' cycle through trainer LEDs
  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-2 Programming (Advanced)

### **RANDOM Variable**

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

```
RANDOM timer
```

## Prop-2 Example (Random Pre-Event Delay)

```
MatSw  PIN    14
Valve  PIN    8
No      CON    0
timer  VAR    Word
delay  VAR    Word
```

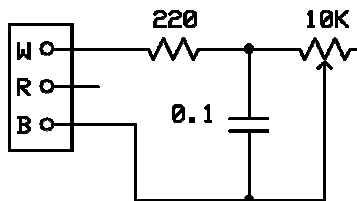
```
Main:
RANDOM timer          ' stir random generator
IF (MatSw = No) THEN Main ' wait for "victim"
delay = timer // 5 + 1 ' create delay, 1 to 5 seconds
delay = delay * 1000  ' convert to milliseconds
PAUSE delay          ' hold for random delay
HIGH Valve           ' open solenoid to lift prop
PAUSE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

## Prop-2 Programming (Advanced)

### **RCTIME** *Pin, Mode, Variable*

**RCTIME** is used to read a variable resistance (e.g., potentiometer, photo-resistor, etc.). *Mode* defines the RC circuit configuration.

```
RCTIME Dial, 1, level
```



## Prop-2 Example (Light-Activated Chaser)

```
LSense PIN      15          ' light sensor
level  VAR      Word       ' light level
pinNum VAR      Nib

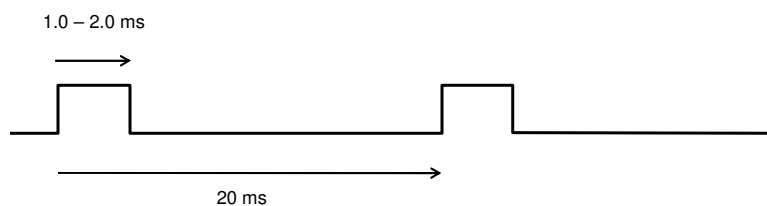
Main:
DO
  HIGH LSense          ' charge capacitor
  PAUSE 1              '   for 5 x R x C (min)
  RCTIME LSense, 1, level ' get current light level
LOOP UNTIL (level > 150) ' wait for dark
FOR pinNum = 8 TO 13   ' cycle through trainer LEDs
  HIGH pinNum          ' LED on
  PAUSE 100           ' hold 0.1 second
  LOW pinNum           ' LED off
NEXT
GOTO Main              ' back to Main
```

## Hobby Servos



## Servo Control

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



## Prop-2 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 2  $\mu$ s units).

```
PULSOUT Servo, 750      ' 1.5 ms pulse (center servo)
```

## Prop-2 Example (Servo Direct)

```
Servo  PIN    0
pos    VAR    Word      ' servo position
delay  VAR    Nib

Setup:
  LOW Servo          ' P0 is output, all others inputs

Main:
  FOR pos = 500 TO 1000 STEP 4 ' 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-2 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, Baud, ["Props are FUN!"]
```



## Prop-2 Example (RC-4 Control)

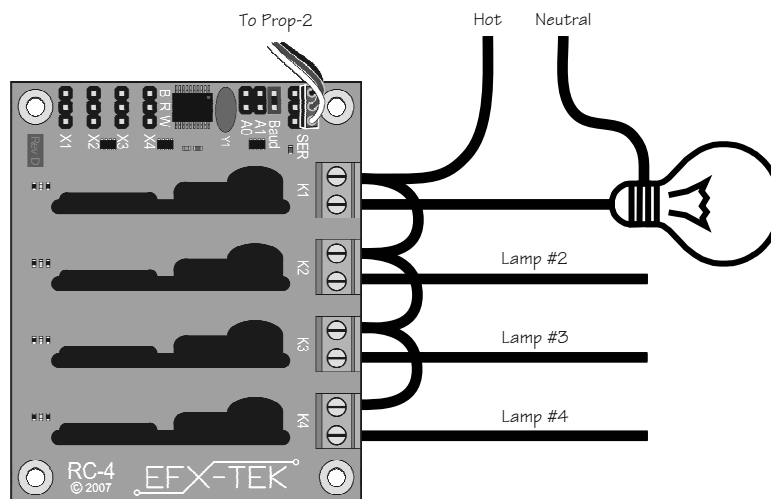
```

TX      PIN    15
MatSw   PIN    14
Baud    CON    $8000 | 6      ' OT38400
No      CON    0
idx     VAR    Nib
lights  VAR    Byte
delay   VAR    Byte
lotto   VAR    Word

Main:
  FOR idx = 1 TO 3
    RANDOM lotto          ' stir random generator
  NEXT
  SEROUT TX, Baud, ["!RC4", %00, "X"]
  IF (MatSw = No) THEN Main ' wait for "victim"
  lights = lotto.NIB0      ' randomize lights
  SEROUT TX, Baud, ["!RC4", %00, "S", lights]
  delay = lotto // 201 + 50 ' create 50 to 250 ms delay
  PAUSE delay              ' hold lights
  GOTO Main                ' back to Main

```

## Prop-2 Example (RC-4 Control)



## Prop-2 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
```

## Prop-2 Example (Timer Subroutine)

```
Led    PIN    8
tix    VAR    Byte

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:
  DO WHILE (tix > 0)     ' check for end of timer
    PAUSE 100           ' hold for 1 tic (0.1 secs)
    tix = tix - 1       ' update tix count
  LOOP                  ' re-check for end of timer
  RETURN                ' go back to main program
```

```
graph TD
  Main --> Run_Timer
  Run_Timer --> Main
```

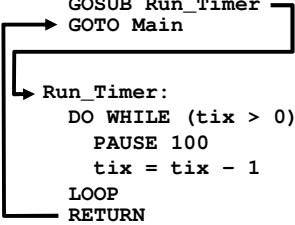
## Prop-2 Example (Timer Subroutine)

```
Led    PIN    8
tix    VAR    Byte
```

```

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:
DO WHILE (tix > 0) ' check for end of timer
PAUSE 100         ' hold for 1 tic (0.1 secs)
tix = tix - 1    ' update tix count
LOOP             ' re-check for end of timer
RETURN          ' go back to main program
```



## Prop-2 Programming – Review

### Essentials

```
PIN, CON, VAR
HIGH Pin
LOW Pin
PAUSE Period
GOTO Label
IF Condition THEN Label
FOR Variable = StartVal TO EndVal ... NEXT
```

### Advanced

```
RANDOM Variable
RCTIME Pin, Mode, Variable
PULSOUT Pin, Period
SEROUT Pin, Baudmode, [Data]
GOSUB Label ... RETURN
DO Statements LOOP
```

# Prop-2 Programming – More

## Additional Instructions

DEBUG *Data, ...*  
DATA {*@Location,*} {*Word*} *Value, ...*  
READ *Location,* {*Word*} *Variable {, Variable, ...}*  
PWM *Pin, Duty, Cycles*  
FREQOUT *Pin, Duration, Freq1 {, Freq2}*  
SERIN *Pin, Baudmode, [Variable, ...]*  
TOGGLE *Pin*

## Advanced Programming Techniques

Learn to use DIRS and OUTS for I/O setup and control  
Master the // (modulus) operator  
Learn to use \*\* and /\* to multiply by fractional values  
Learn to use ON Variable GOSUB Label... for state-driven programs

## Contact

- [www.efx-tek.com](http://www.efx-tek.com)
- [teamefx@efx-tek.com](mailto:teamefx@efx-tek.com)
- 916-616-1658
  
- [shop.efx-tek.com](http://shop.efx-tek.com)
- [forums.efx-tek.com](http://forums.efx-tek.com)