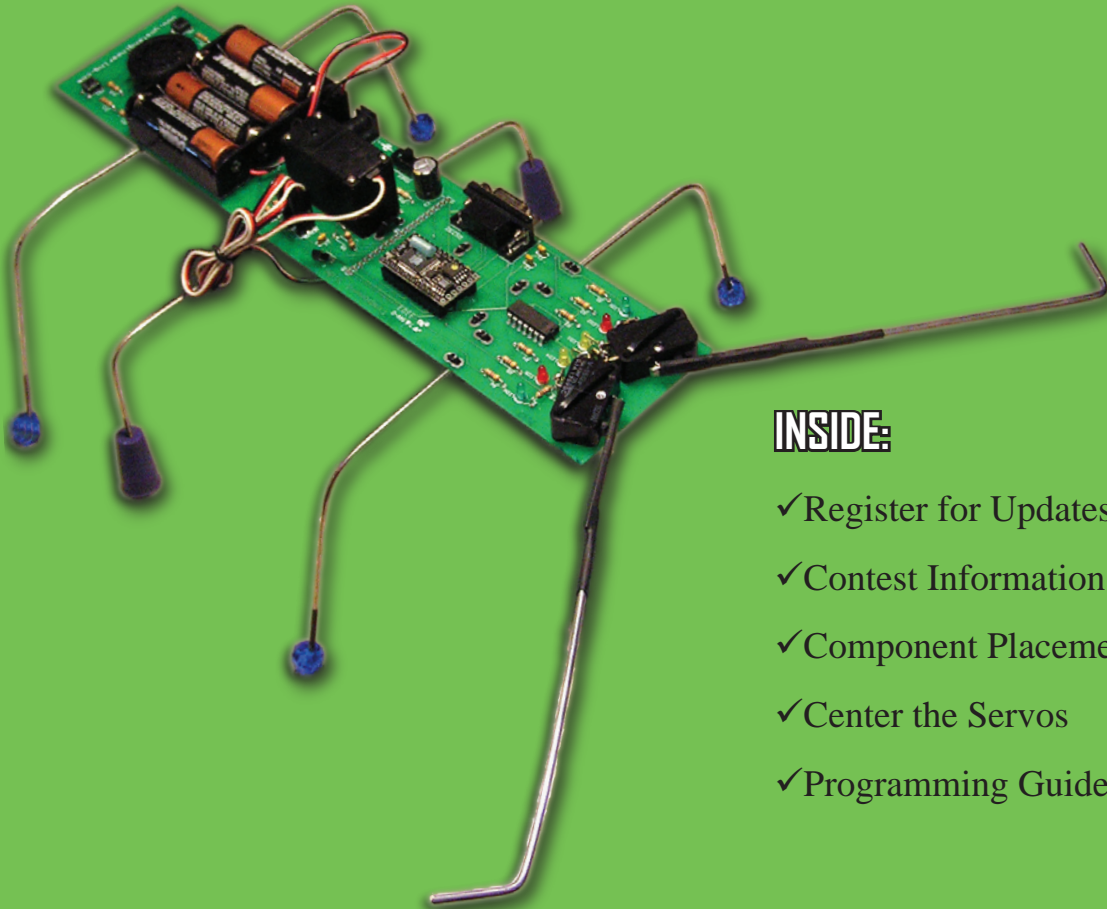


Educational Kit Series  
Part Number: BB6X

# BugBrain™ 3.0.2

## Resource Manual

(BasicX Version)



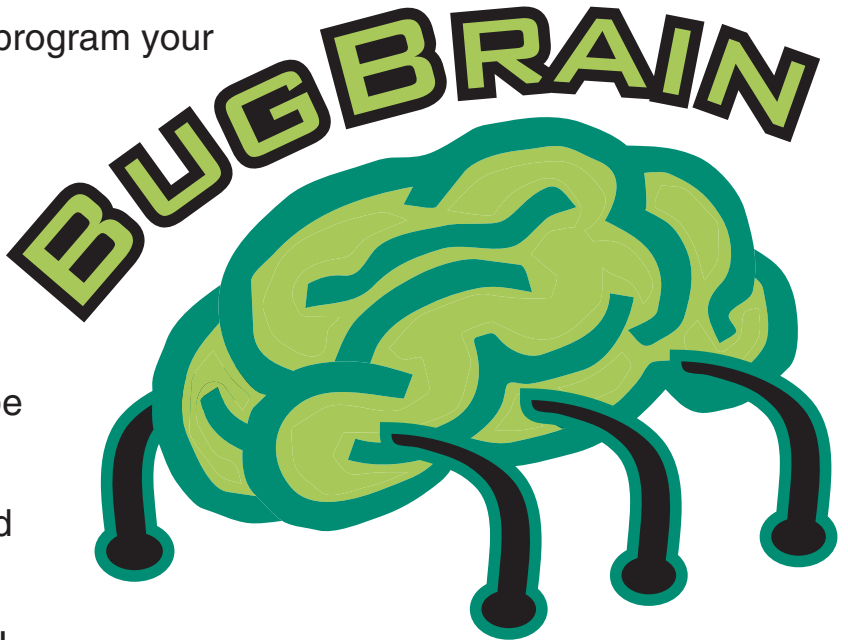
### INSIDE:

- ✓ Register for Updates
- ✓ Contest Information
- ✓ Component Placement Sheet
- ✓ Center the Servos
- ✓ Programming Guide

# Register Now!

## Why register your BugBrain™ ?

- ✓ Get notification of new features, parts, and add-on accessories
- ✓ Get sample code you can use to program your BugBrain
- ✓ Get pictures and tips about what other BugBrain owners are doing
- ✓ Find out about Contests
- ✓ Learn from the FAQs and Q&A columns in our newsletters, and be able to post your own questions
- ✓ Learn about special discounts and sales
- ✓ Tell us what you want to see next!



✂ -----

### REGISTRATION FORM

Complete the following Mail-In form or email this information to robots@YostEngineering.com:

Name: \_\_\_\_\_

If Under Age 18 Only: Age: \_\_\_\_\_ Parent Name: \_\_\_\_\_

Address: \_\_\_\_\_

City: \_\_\_\_\_ State: \_\_\_\_\_ ZIP: \_\_\_\_\_

Phone: \_\_\_\_\_

Email: \_\_\_\_\_

Where did you buy your BugBrain? \_\_\_\_\_

Mail this form to:  
Yost Engineering, Inc. 630 Second Street, Portsmouth, OH 45662  
or email this information to robots@YostEngineering.com

# BugBrain Contests!

Finished building your BugBrain?

Does your BugBrain look amazing? Did you program it to do something new and different? **Tell us about it, and you could win a prize!**

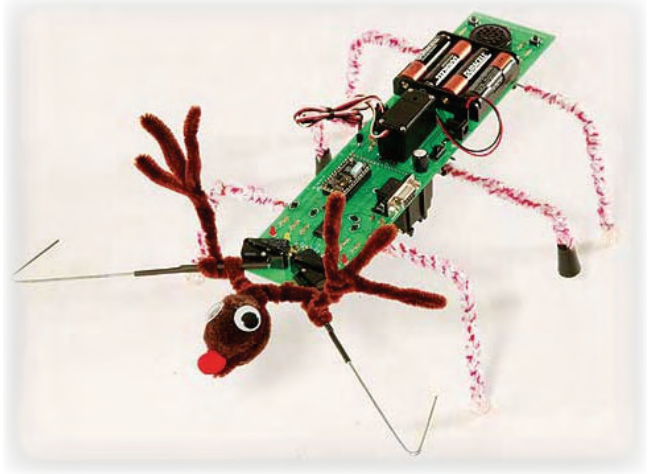
All participants get their name and BugBrain picture on our webpage, so go ahead and show off your creation!

Enter any time by sending your name, address, age, and email address, along with pictures and programming code (for Bug-Attack contest) to:

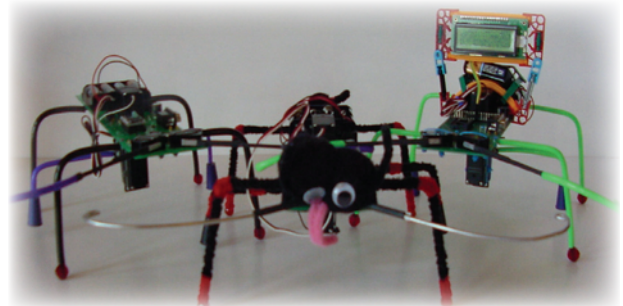
Robots@YostEngineering.com

Be sure to specify which contest(s) you are entering (you may enter both):

**Picture-Perfect Contest:** Send us one or more pictures of your assembled BugBrain, showing off your personalized body parts or decorations. One winner will receive a \$50 Amazon.com gift certificate. One runner-up receives a \$25 gift certificate.

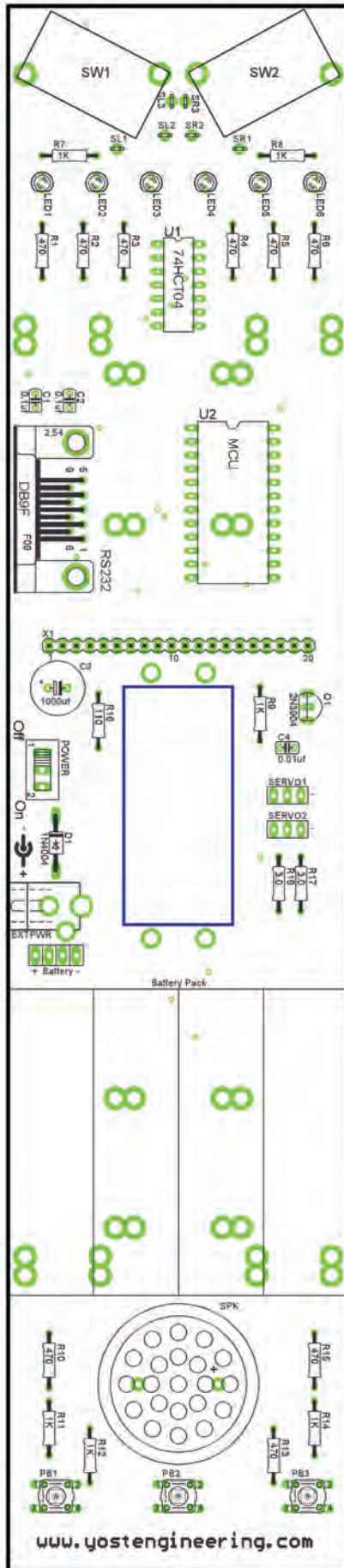


**Bug-Attack Contest:** Send us pictures of your BugBrain in action, along with the code that makes it happen (specify whether or not we can share the code with others). One winner gets a \$75 Amazon.com gift certificate, and one runner-up gets a \$50 gift certificate.



**Contests end December 31st and June 30th every year.**

# BugBrain v3.0.2 - Component Placement Sheet



**Resistors:**

- R1 = 470  $\Omega$
- R2 = 470  $\Omega$
- R3 = 470  $\Omega$
- R4 = 470  $\Omega$
- R5 = 470  $\Omega$
- R6 = 470  $\Omega$
- R7 = 1K  $\Omega$
- R8 = 1K  $\Omega$
- R9 = 1K  $\Omega$
- R10 = 470  $\Omega$
- R11 = 1K  $\Omega$
- R12 = 1K  $\Omega$
- R13 = 470  $\Omega$
- R14 = 1K  $\Omega$
- R15 = 470  $\Omega$
- R16 = no resistor
- R17 = 3.0  $\Omega$
- R18 = 3.0  $\Omega$

**Capacitors:**

- C1 = 0.1  $\mu$ f
- C2 = 0.1  $\mu$ f
- C3 = 1000  $\mu$ f
- C4 = 0.01  $\mu$ f

**Switches:**

- PB1 = Key Switch
- PB2 = Key Switch
- PB3 = Key Switch
- SW1 = Long Hinge Switch
- SW2 = Long Hinge Switch

**Diodes:**

- D1 = 1N4004 Diode
- LED1 = Light Emitting Diode
- LED2 = Light Emitting Diode
- LED3 = Light Emitting Diode
- LED4 = Light Emitting Diode
- LED5 = Light Emitting Diode
- LED6 = Light Emitting Diode

**Connectors:**

- EXTPWR - Power Jack ( 6VDC )
- DB9F = 9 Pin DSUB Connector
- X1 = Expansion Connector ( Optional )

**Other Devices:**


- SPK = Speaker
- U1 = 74HCT04 Inverter Integrated Circuit
- U2 = Socket for BasicStamp2 / BasicStamp2 Module
- Q1 = PN2222 Transistor

www.yostengineering.com

# Centering the Servos

## BasicX-24 Processor Option

During the construction of your BugBrain v.3.0.2 you will need to center the servo motors as directed in the instructions. Follow the directions on this sheet to center the servos.

1. **Install the BasicX 2.00 software on your computer.** To do this: place the BasicX Software CD into your computer's CD-ROM drive. When the menu appears click on "Install BasicX Software". Follow the on-screen prompts.
2. **Run the BasicX 2.00 development environment software.** To do this click on the Windows "Start" button, navigate to the "Programs" menu, then click on the "BasicX 2.00" menu selection.
3. **Open the editor window.** To do this click on the  icon. This will open a new editor window.
4. **Create a new project.** To do this click on the "File" menu and select the "New Project" menu selection. Type the project's name ( such as CenterServos ) in the "New project name" field then click the "Ok" button in the "New Project" dialog box.
5. **Type the CenterServos program.** To do this type the following program into the BasicX Editor window:


```

`CenterServos program starts
  Option Explicit

Public Sub Main()
  Dim i as Byte
  `turn off lights
  For i = 5 to 10
    Call putpin(i,0)
  Next
  `center servos
  Call moveservos(0.5,0.5,50)
End Sub

Sub moveservos(byval Position1 as single,byval Position2 as single, byval times as integer)
  Dim PulseWidth As Single
  Dim i as Integer
  For i = 1 to times
    PulseWidth = 0.0003 + (0.002 * Position1)
    Call PulseOut(14, PulseWidth, 1)
    PulseWidth = 0.0003 + (0.002 * Position2)
    Call PulseOut(13, PulseWidth, 1)
    Call delay(0.02)
  Next
End Sub
`CenterServos program ends

```

6. Download and run the program. To do this connect the BugBrain to your computer using the provided serial cable, connect power to your BugBrain, turn your BugBrain's power switch to the "On" position. Click the  button to download and run the program.

*That does it! Your BugBrain has now been programmed to center the servo motors whenever it is powered up. You may now turn the BugBrain off and disconnect the serial programming cable.*

# BugBrain™ 3.0.2 Programming Guide

( BasicX-24 Version )

With the addition of the BasicX Controller Module ( BasicX24 - available from NetMedia Inc. ) all you need to do is write some programs to get your robot to do whatever you'd like!

The BasicX Controller programming environment software comes on the BasicX CD. To install the software simply insert the CD and follow the on-screen prompts. The BasicX-24 programming environment may also be downloaded from [www.netmedia.com](http://www.netmedia.com).

The BasicX24 controller module connects to a 24 pin DIP socket and the pins are allocated as listed below. Note that pin 1 is in the upper left of the controller module board next to the square pad.

## Pin Descriptions:

Pin 1 - SOUT - Serial Port Output

Pin 2 - SIN - Serial Port Input

Pin 3 - ATN - Used for programming

Pin 4 - Vss - System Ground for BasicX24 Module and the BugBrain's circuitry.

Pin 5 - I/O Port Pin 5 - LED1 - Left-most LED ( looking from the back with the feelers facing away from you)

Pin 6 - I/O Port Pin 6 - LED2 ( counting from the left and looking from the back with the feelers facing away from you)

Pin 7 - I/O Port Pin 7 - LED3 ( counting from the left and looking from the back with the feelers facing away from you)

Pin 8 - I/O Port Pin 8 - LED4 ( counting from the left and looking from the back with the feelers facing away from you)

Pin 9 - I/O Port Pin 9 - LED5 ( counting from the left and looking from the back with the feelers facing away from you)

Pin 10 - I/O Port Pin 10 - LED6 - Right-most LED ( looking from the back with the feelers facing away from you)

Pin 11 - I/O Port Pin 11 - Not connected - available for use.

Pin 12 - I/O Port Pin 12 - Not connected - available for use.

Pin 13 - I/O Port Pin 13 - Control signal for upper servo ( SERVO2 on the board layout ).

Pin 14 - I/O Port Pin 14 - Control signal for lower servo ( SERVO1 on the board layout ).

Pin 15 - I/O Port Pin 15 - Speaker output signal.

Pin 16 - I/O Port Pin 16 - Left rear input button. ( looking from the back with the feelers away from you)

Pin 17 - I/O Port Pin 17 - Middle rear input button. ( looking from the back with the feelers away from you)

Pin 18 - I/O Port Pin 18 - Right rear input button. ( looking from the back with the feelers away from you)

Pin 19 - I/O Port Pin 19 - Right front feeler input. ( looking from the back with the feelers away from you)

Pin 20 - I/O Port Pin 20 - Left front feeler input. ( looking from the back with the feelers away from you)

Pin 21 - Regulated +5 volts DC coming from the BasicX24 Module.

Pin 22 - /RES - Active Low Reset for the BasicX24 Module.

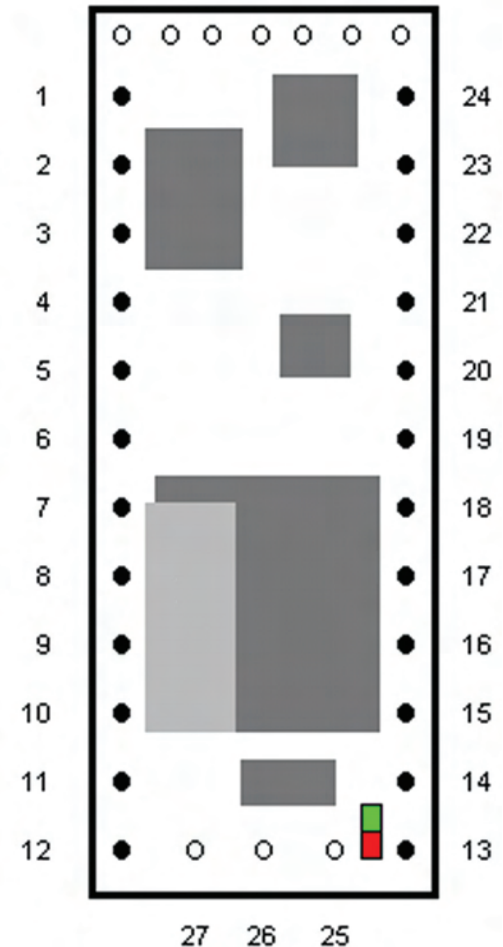
Pin 23 - Vss - System Ground for BasicX24 Module and the BugBrain's circuitry.

Pin 24 - Vin - The switched unregulated +6 volts DC coming from the batteries or power pack.

## Other Pins:

Pin 25 - Red LED- located on processor module.

Pin 26 - Green LED- located on processor module.



## Example 1:

```
`*****`
`** Example Program 1
`** Illustrates the use of the PutPin, Do Loop, and Delay Statements
`** The program lights one light at a time, goes to the next
`** light after one second and then repeats
`*****`
Option Explicit

Public Sub Main()
    dim i as byte
    `set all pins off
    for i = 5 to 20
        call putpin(i,0)
    next

    do
        call putpin(5,1)
        call delay(1.0)
        call putpin(5,0)
        call putpin(6,1)
        call delay(1.0)
        call putpin(6,0)
        call putpin(7,1)
        call delay(1.0)
        call putpin(7,0)
        call putpin(8,1)
        call delay(1.0)
        call putpin(8,0)
        call putpin(9,1)
        call delay(1.0)
        call putpin(9,0)
        call putpin(10,1)
        call delay(1.0)
        call putpin(10,0)

    loop

End Sub
```

**Example 2:**

```
*****
** Example Program 2
** Illustrates the use of the Delay Statement
** The program lights one light at a time, goes to the next
** light after 100 milliseconds and then repeats
*****
Option Explicit

Public Sub Main()
    dim i as byte
    `set all pins off
    for i = 5 to 20
        call putpin(i,0)
    next

    do
        call putpin(5,1)
        call delay(0.1)
        call putpin(5,0)
        call putpin(6,1)
        call delay(0.1)
        call putpin(6,0)
        call putpin(7,1)
        call delay(0.1)
        call putpin(7,0)
        call putpin(8,1)
        call delay(0.1)
        call putpin(8,0)
        call putpin(9,1)
        call delay(0.1)
        call putpin(9,0)
        call putpin(10,1)
        call delay(0.1)
        call putpin(10,0)
    loop

End Sub
```

**Example 3:**

```

*****
** Example Program 3
** Illustrates the use of the FREQOUT Statement to make sound.
** The program plays Mary Had a Little Lamb
** Note sequence: E,D,C,D,E,E,E,Rest,D,D,D,Rest,E,G,G,Rest,E,D,C,D,E,E,E,E,D,D,E,D,C
*****
Option Explicit
Sub Main()
    'E
    call freqout(15,659,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'C
    call freqout(15,523,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'Rest
    call delay(250)
    'D
    call freqout(15,587,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'Rest
    call delay(250)
    'E
    call freqout(15,659,0,0.25)
    'G
    call freqout(15,748,0,0.25)
    'G
    call freqout(15,784,0,0.25)
    'Rest
    call delay(250)
    'E
    call freqout(15,659,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'C
    call freqout(15,523,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'E
    call freqout(15,659,0,0.25)
    'D
    call freqout(15,587,0,0.25)
    'C
    call freqout(15,523,0,0.25)
End Sub

```

**Example 4:**

```
*****
** Example Program 4
** Illustrates the use of the GetPin, IF/THEN, For Next Statements
** The program reads the front bumper switches and then sends
** a light pattern across the LEDs away from the
** front switch that was pressed. This program shows how to
** check a switch or button for activity.
*****

Option Explicit
Public Sub Main()
    dim i as byte
    `set all pins off
    for i = 5 to 20
        call putpin(i,0)
    next

    `set front feeler pins as inputs
    call putpin(19,3)
    call putpin(20,3)

    do
        if getpin(19) = 1 then
            for i = 5 to 10
                call putpin(i,1)
                call delay(0.05)
                call putpin(i,0)
            next
        end if
        if getpin(20) = 1 then
            for i = 10 to 5 step -1
                call putpin(i,1)
                call delay(0.05)
                call putpin(i,0)
            next
        end if
    loop
End Sub
```

## Example 5:

```
*****
** Example Program 5
** Illustrates the use of the GetPin and IF/THEN Statements
** The program reads the back switches and then sends
** a light pattern to the LEDs depending which switch is pressed
*****
Option Explicit

Public Sub Main()
    dim i as byte
    `set all pins off
    for i = 5 to 20
        call putpin(i,0)
    next

    `set front feeler pins as inputs
    call putpin(19,3)
    call putpin(20,3)

    do
        if getpin(16) = 1 then
            call putpin(5,1)
            call putpin(6,1)
        else
            call putpin(5,0)
            call putpin(6,0)
        end if
        if getpin(17) = 1 then
            call putpin(7,1)
            call putpin(8,1)
        else
            call putpin(7,0)
            call putpin(8,0)
        end if
        if getpin(18) = 1 then
            call putpin(9,1)
            call putpin(10,1)
        else
            call putpin(9,0)
            call putpin(10,0)
        end if
    loop
End Sub
```

**Example 6:**

```

*****
**  Servo Moving Example Program
**  Illustrates the use of servo control.
**  The program moves the servos in various ways using
**  calls to the predefined moveservos sub program.
*****

```

```
Option Explicit
```

```
Public Sub Main()
```

```
    dim i as byte
```

```
    `turn off lights
```

```
    for i = 5 to 10
```

```
        call putpin(i,0)
```

```
    next
```

```
    `center servos
```

```
    call moveservos(0.5,0.5,30)
```

```
    `servo1 Forward
```

```
    call moveservos(0.7,0.5,30)
```

```
    `center servos
```

```
    call moveservos(0.5,0.5,30)
```

```
    `servo1 Backward
```

```
    call moveservos(0.3,0.5,30)
```

```
    `center servos
```

```
    call moveservos(0.5,0.5,30)
```

```
    `servo2 Forward
```

```
    call moveservos(0.5,0.7,30)
```

```
    `center servos
```

```
    call moveservos(0.5,0.5,30)
```

```
    `servo2 Backward
```

```
    call moveservos(0.5,0.3,30)
```

```
    `center servos
```

```
    call moveservos(0.5,0.5,30)
```

```
End Sub
```

```
(Example 6 Continued)
```

```
sub moveservos(byval Pos1 as single,byval Pos2 as single, byval times as integer)
```

```
    Dim PulseWidth As Single
```

```
    dim i as integer
```

```
    for i = 1 to times
```

```
        ` Translate position to pulse width. Resulting range is 1.0 to 2.0 ms,
```

```
        ` centered at 1.5 ms.
```

```
        PulseWidth = 0.0003 + (0.002 * Pos1)
```

```
        ` Generate a high-going pulse on the servo pin.
```

```
        Call PulseOut(14, PulseWidth, 1)
```

```
        PulseWidth = 0.0003 + (0.002 * Pos2)
```

```
        ` Generate a high-going pulse on the servo pin.
```

```
        Call PulseOut(13, PulseWidth, 1)
```

```
        Call delay(0.02)
```

```
    next
```

```
end sub
```

**Example 7:**

```

*****
** Walking Robot Example Program using servo motor control.
** Illustrates the use of servo motor locomotion.
** The program moves the servos to make your robot walk, bump
** into things and backup.
*****
Option Explicit

Public Sub Main()
    dim i as byte

    'turn off lights
    for i = 5 to 10
        call putpin(i,0)
    next

    'center servos
    call moveservos(0.5,0.5,30)

    call runLights
    call delay(4.0)

    'main loop
    call beginstep()
    do while true
        call astep()
        call bumpercheck()
        call bstep()
        call bumpercheck()
    loop

End Sub

sub bumpercheck()
    if getpin(19) = 1 then    ' Left Bumper
        call cstep()
        call dstep()
        call cstep()
        call dstep()
        call runlights()
        call backleft()
    end if
    if getpin(20) = 1 then    ' Right Bumper
        call dstep()
        call cstep()
        call dstep()
        call cstep()
        call runlights()
        call backright()
    end if
end sub

sub backleft()
    dim i as byte
    call moveservos(0.7,0.5,15)
    call moveservos(0.7,0.35,15)
    call moveservos(0.3,0.35,15)
    call moveservos(0.3,0.5,15)
    call moveservos(0.7,0.5,15)
    call moveservos(0.7,0.65,15)
    call moveservos(0.3,0.65,15)
    call moveservos(0.3,0.5,15)
end sub

```

**(Example 7 Continued)**

```
sub backright()
  dim i as byte

  call moveservos(0.3,0.5,15)
  call moveservos(0.3,0.35,15)
  call moveservos(0.7,0.35,15)
  call moveservos(0.7,0.5,15)
  call moveservos(0.3,0.5,15)
  call moveservos(0.3,0.65,15)
  call moveservos(0.7,0.65,15)
  call moveservos(0.7,0.5,15)
end sub

sub beginstep()
  call moveservos(0.7,0.5,15)
end sub

sub astep()
  call moveservos(0.7,0.65,15)
  call moveservos(0.3,0.65,15)
end sub

sub bstep()
  call moveservos(0.3,0.35,15)
  call moveservos(0.7,0.35,15)
end sub

sub cstep()
  call moveservos(0.3,0.65,15)
  call moveservos(0.7,0.65,15)
end sub

sub dstep()
  call moveservos(0.7,0.35,15)
  call moveservos(0.3,0.35,15)
end sub

sub fullstep()
  call astep()
  call bstep()
end sub

sub runLights()
  dim i as byte
  for i = 5 to 10
    call putpin(i,1)
    call freqout(15,100*cint(i),0,0.05)
    call putpin(i,0)
  next
  for i = 9 to 5 step -1
    call putpin(i,1)
    call freqout(15,100*cint(i),0,0.05)
    call putpin(i,0)
  next
  for i = 5 to 10
    call putpin(i,1)
    call freqout(15,100*cint(i),0,0.05)
    call putpin(i,0)
  next
  for i = 9 to 5 step -1
    call putpin(i,1)
    call freqout(15,100*cint(i),0,0.05)
    call putpin(i,0)
  next
end sub
```

**(Example 7 Continued)**

```
sub moveservos(byval Pos1 as single,byval Pos2 as single, byval times as integer)
  Dim PulseWidth As Single
  dim i as integer

  for i = 1 to times
    ` Translate position to pulse width. Resulting range is 1.0 to 2.0 ms,
    ` centered at 1.5 ms.
    PulseWidth = 0.0003 + (0.002 * Pos1)

    ` Generate a high-going pulse on the servo pin.
    Call PulseOut(14, PulseWidth, 1)

    PulseWidth = 0.0003 + (0.002 * Pos2)
    ` Generate a high-going pulse on the servo pin.
    Call PulseOut(13, PulseWidth, 1)
    Call delay(0.02)
  next
end sub
```



Yost Engineering Inc.  
630 Second Street  
Portsmouth, Ohio 45662

[www.YostEngineering.com](http://www.YostEngineering.com)

888.395.9029 | 740.355.9029  
Fax: 888.565.1170 | 740-354-1170  
Sales: [sales@yostengineering.com](mailto:sales@yostengineering.com)  
Support: [tech@yostengineering.com](mailto:tech@yostengineering.com)

**1-888-395-9029**

[www.YostEngineering.com/BugBrain](http://www.YostEngineering.com/BugBrain)