Microcontrollers

Module 5: Motion Control

PREPARED BY

IAT Curriculum Unit

August2008

Module 5: Motion Control

Module Objectives

Upon successful completion of this module, students will be:

1. able to demonstrate knowledge of microcontroller motion control applications.
2. familiar with the basics of servo motors and control.
3. capable of programming the BASIC Stamp Microcontroller to control the angle and speed of the Servo Motor.

Module Contents:

	Topic	**Page No.**
5.1	Introduction to Microcontroller Motion Control	5
5.2	Introduction to Servo Motor	6
5.3	Servo Motor Control	7
5.4	Programming the BASIC Stamp for Servo Control	9
5.5	Practical Task 1: Connecting and testing a Servo Motor	12
5.6	Practical Task 2: Servo Control with Debug program	15

5.1 Microcontroller Motion Control

Microcontrollers control the motion of many things in our daily lives; a few examples are given below:

- Printer head movement
- DVD and VCR mechanisms
- automatic doors
- Robotic movement

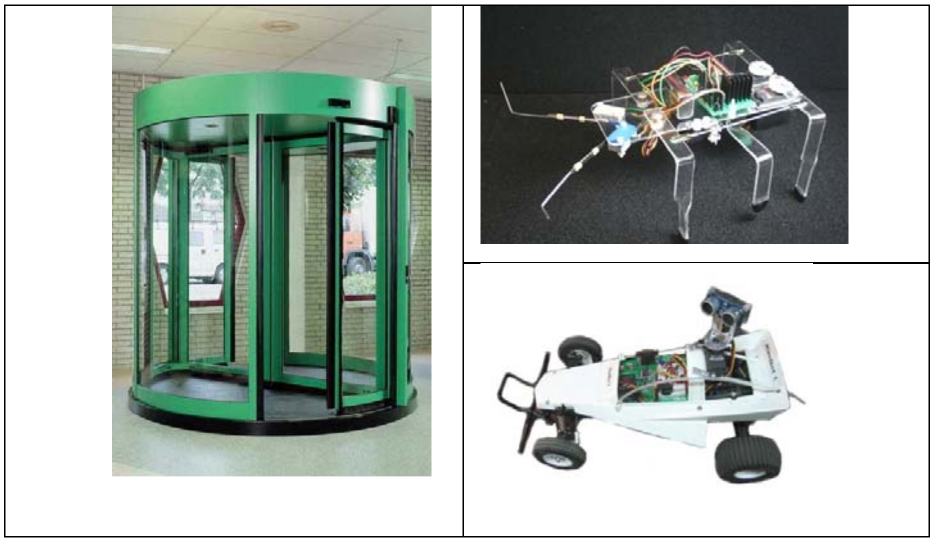


Figure 5.1: Microcontroller Motion Control Applications

Instead of simple ON/OFF signals, most of these motion devices require very fast pulses of signals for position control or movement.The examples of motion devices/motors are as follows:

- DC Motors
- AC Motors
- Stepper Motors
- Servo Motors

All the motion devices mentioned above can be controlled from the BASIC Stamp, though many need additional electronic circuitry or additional mechanical components.

5.2 Introduction to Servo Motor

The hobby servo or the Servo motor is the simplest and most directly useful of all DC motors that could be controlled from the BASIC Stamp. In this module, you will learn how to control a Servo motor.

A servo motor can be found in many devices that we use every day. For example a tiny built in servo motor controlled by a microcontroller causes the following:

- Movement of the print head in an inkjet printer
- Automatic eject feature in VCR and DVD players.
- Opening and closing of automatic doors in shopping malls, hospitals and so on.

Figure 5.2 shows a Parallax Servo Motor.

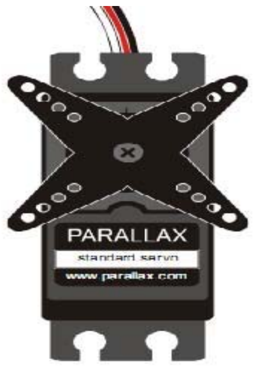


Figure 5.2: Parallax Servo Motor

What makes a servo motor different from other typical motors?

Unlike other motors, a servo motor does not have a continuous shaft rotation, and **can only rotate between two defined angles** specified by the manufacturer.

5.3 Servo Motor Control

Unlike the other standard microcontroller outputs, the **control signal of the servo motor is separate from the power signal**. Therefore, to use the servo motor, all three wires should be connected as shown in Figure 5.3.

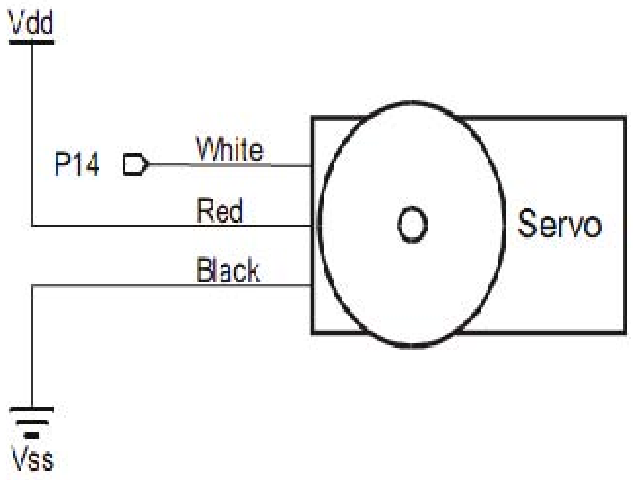
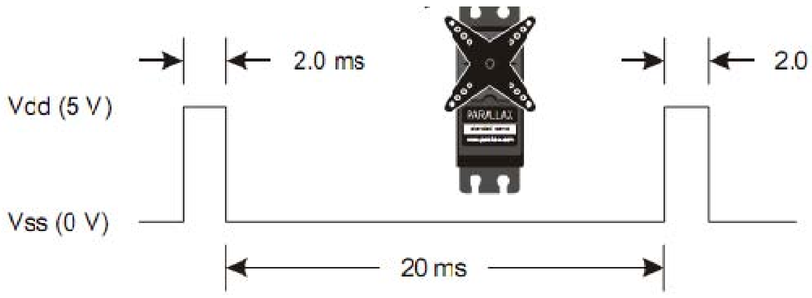


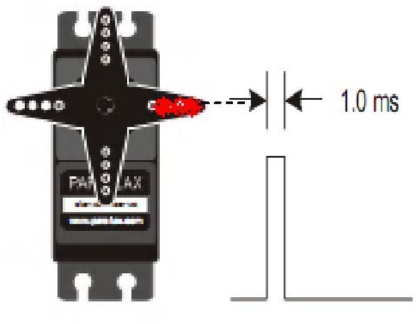
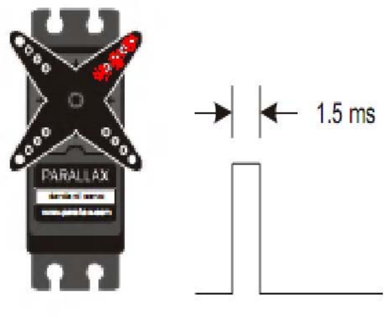
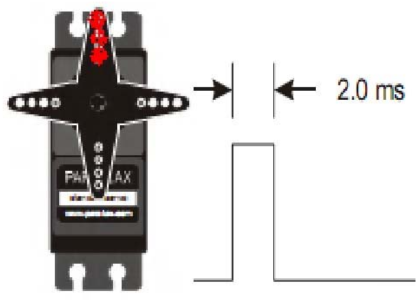
Figure 5.3: Servo Motor Connection Diagram

Wire	Connected to	Purpose
White	P14 or any other microcontroller output pin	The control signal from the microcontroller will control the rotation of the motor towards a targeted angle.
Red	Vdd or +5V	To supply power to the motor
Black	Vss	Ground connection

A servo motor is controlled by very **short high signals** or pulses spaced 20ms apart as illustrated in figure below. The duration of a pulse is between 1ms and 2ms.



The duration of the high signal determines the targeted angle the servo motor should be positioned at. For example, 1ms high signal duration causes the servo motor (red marked corner) to be positioned at angle zero degree, 1.5ms sets it at 45 degrees and 2ms sets the angle at 90 degrees as shown in figures 5.4, 5.5 and 5.6.

	Signal duration	Targeted angle
 Fig 5.4: A pulse width of 1ms corresponds to 0 angle	1ms	0 degrees
 Fig 5.6: A pulse width of 1.5 ms corresponds to 45 angle	1.5ms	45 degrees
 Fig 5.5: A pulse width of 2ms corresponds to 90 angle	2ms	90 degrees

5.4 Programming the BASIC Stamp for Servo Control

PULSOUT Command: The PULSOUT command is used to send the signal to control the movement of the servomotor.

The command syntax is as follows:

PULSOUT pin, duration
Pin: defines which I/O pin to use
Duration: defines the duration of a single pulse in microseconds.

The PULSOUT duration is in 2 microsecond (us) increments.
1 μs = 10^{-6} seconds.
1000μs = 1ms

For example, the command PULSOUT 14,750 would be sending a pulse that lasts 750 x 2 μs = 1500 μs or 1.5ms on pin 14

To calculate the duration if the time period is in ms, the following formula could be used:

Duration = Time (ms) X 500

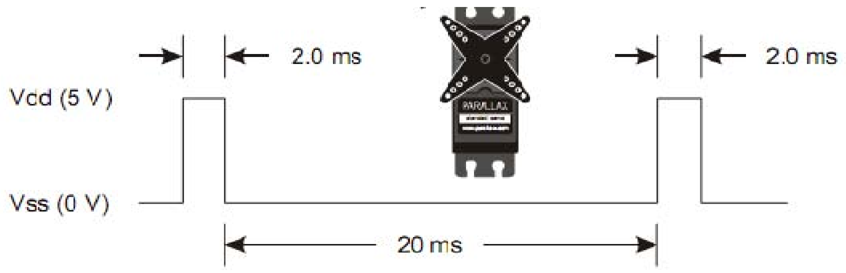
The table below shows the duration for the pulses with the following time period:

Time (ms)	**Duration**
1ms	500
1.5ms	750
2ms	1000

The following program code could be used to generate 150 pulses, each of duration 1000 (or 2ms) with 20ms pauses.

```
FOR counter = 1 TO 150
  PULSOUT 14, 1000
  PAUSE 20
NEXT
```

The timing diagram will be as follows:



Exercise:

1. Write the BASIC Stamp program code to generate 200 pulses, each of duration 1.5ms.

   ……………………………………………………………………………………………………

   ……………………………………………………………………………………………………

   ……………………………………………………………………………………………………

   ……………………………………………………………………………………………………

2. Write the BASIC Stamp program code to generate 150 pulses, each of duration 500.

   ……………………………………………………………………………………………………

   ……………………………………………………………………………………………………

   ……………………………………………………………………………………………………

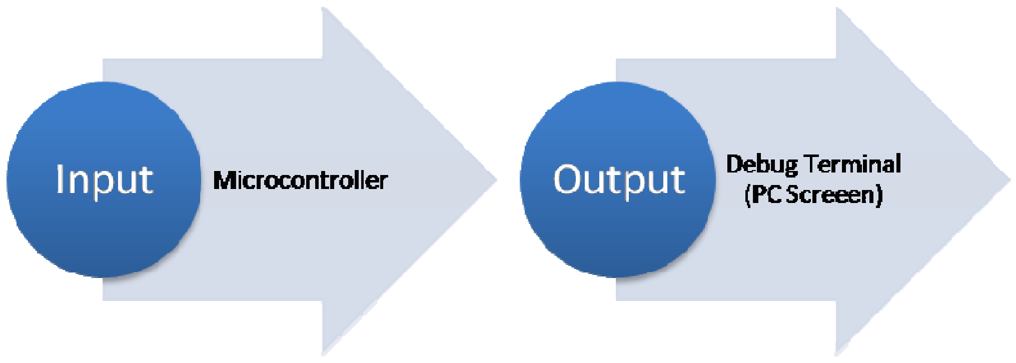
Now, try Practical Task-1 on page 12

<u>DEBUGIN Command:</u> The DEBUGIN command is used to send a value from the debug terminal to the microcontroller, and the value is stored in the declared variable. *Command Syntax: DEBUGIN DEC Variable*

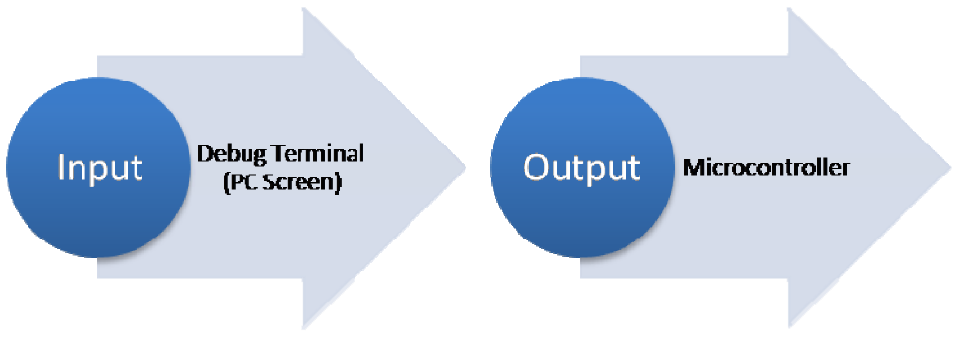
For example, in the command DEBUGIN DEC Pulses, a decimal value will be sent from the debug terminal to be stored in the variable Pulses. DEC indicates that the value stored is a decimal number.

Compare the DEBUG command and the DEBUGIN command:

DEBUG Command



DEBUGIN Command



What do you think is the difference?

………..

………..

Now, try Practical Task-2 on page 15

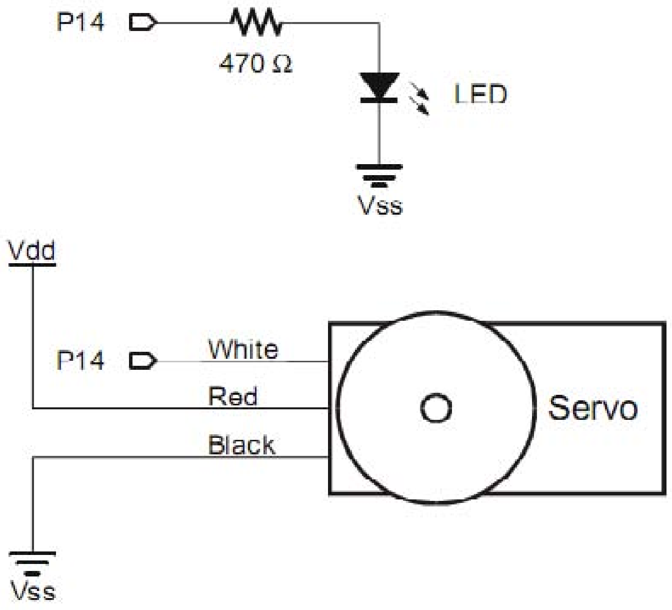
5.5 Practical Task 1: Connecting and testing the servo motor

In this experiment, you will connect the servo motor to the power supply and the BASIC Stamp microcontroller and test its operation.

Parts needed:

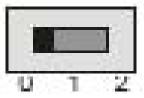
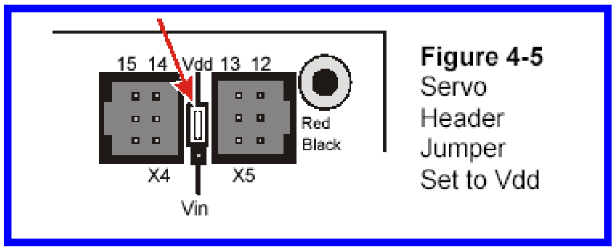
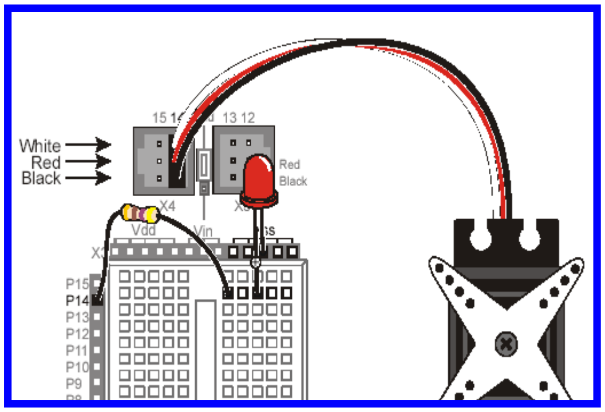
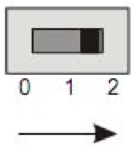
BASIC Stamp BoE
Servo motor
LED-any color
Resistor-470Ω

Schematic Diagram:



Procedure:

Hardware connection

Set the 3-position switch on the BoE to position-0 to turn it OFF as in Figure 5.7	 Figure:5.7
Set the jumper to Vdd as shown in figure 5.8. The jumper is the removable black rectangular piece right between the servo headers, and it is used to select either Vdd or Vin to supply power to the servo motor.	 15 14 Vdd 13 12 Red Black X4 X5 Vin Figure 4-5 Servo Header Jumper Set to Vdd Figure:5.8
Build the circuit shown in figure 5.9. Make sure the motor cable is plugged in the right direction (white wire on top)	 White Red Black Red Black P15 P14 P13 P12 P11 P10 P9 Figure:5.9
Power on the BoE by adjusting the three position switch to 2 as shown in figure 5.11 to turn on the servo motor as well.	 Figure:5.10

Program:

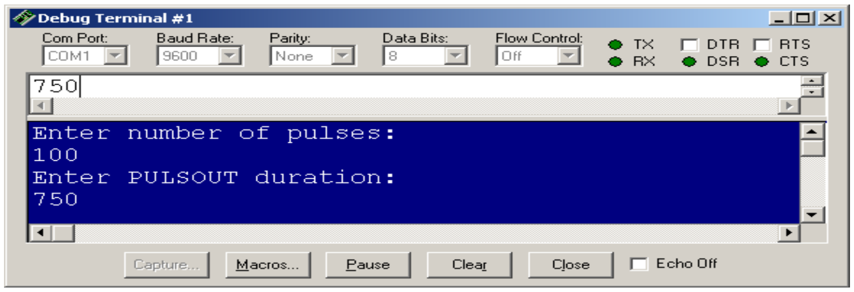
Run the following program, and observe the motor angle at each step of the three steps. Observe the motor angle through which the servo horn is rotated during each step and note down the result in the table below:

```
counter VAR Word
DEBUG "Counterclockwise 10 o'clock", CR
FOR counter = 1 TO 150
  PULSOUT 14, 1000
  PAUSE 20
NEXT

DEBUG "Clockwise 2 o'clock", CR
FOR counter = 1 TO 150
  PULSOUT 14, 500
  PAUSE 20
NEXT

DEBUG "Center 12 o'clock", CR
FOR counter = 1 TO 150
  PULSOUT 14, 750
  PAUSE 20
NEXT

DEBUG "All done."
END
```

Step-1	Step-2	Step-3
PULSOUT 14, 1000	PULSOUT14, 500	PULSOUT 14, 750
Angle:………………………..	Angle:………………………..	Angle:………………………..

5.6 Practical Task 2: Servo control with Debug program

In this experiment, you will use the DEBUG window or DEBUG terminal to control the servo by entering the number of pulses and the duration as shown in Figure 5.11.



```
750
```

```
Enter number of pulses:
100
Enter PULSOUT duration:
750
```

Figure: 5.11

The **DEBUGIN** command is used in the program below to send data FROM the computer TO the BASIC Stamp. Use the same circuit as in experiment 1, enter the program code and run the program.

```
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word
pulses VAR Word
duration VAR Word

DO
  DEBUG CLS, "Enter number of pulses:", CR
  DEBUGIN DEC pulses
  DEBUG "Enter PULSOUT duration:", CR
  DEBUGIN DEC duration
  DEBUG "Servo is running...", CR
  FOR counter = 1 TO pulses
    PULSOUT 14, duration
    PAUSE 20
  NEXT
  DEBUG "DONE"
  PAUSE 1000
LOOP
```

Run the program and enter the value ‘100’ when the debug terminal prompts for the ‘number of pulses’. Run the program thrice by entering different values for PULSOUT duration and note down the output angle for each trial in the table below:

Duration	Angle

Note that the value that can be entered for duration can be between 500 and 1000. Any number beyond that range will force the servo to rotate to a position beyond its mechanical limits which could shorten its lifespan. The value for the number of pulses can be any value between 1 and 65534.

What does the DEBUG formatter ‘CLS’ stand for? What is its function?

……..

……..

……..