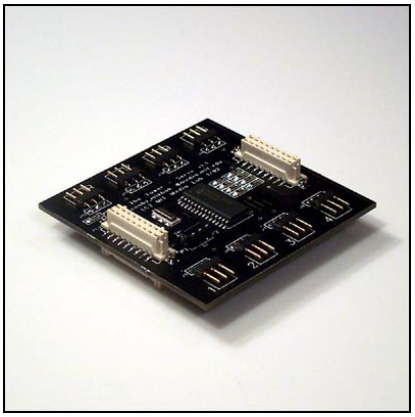
Servo Layer

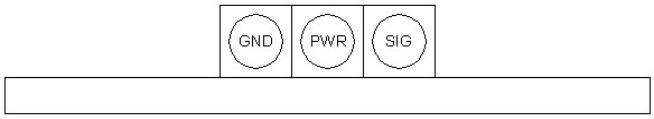


Description

The Servo layer has ports for eight standard servo-motor connections. Each servo can be independently turned on or off and can be set to any angle within a 100 degree range.

Hardware Detail

The Servo layer uses a PIC to generate the PWM (Pulse Width Modulation) signals needed to communicate with standard hobby servos. The connector is a 3-pin 0.1" header, compatible with Futaba J-Type servo connectors. The pin configuration of the port is shown below.



When powered on, there will be a pulse train present on the signal line, with a positive pulse width between 1ms and 2ms long. The width of the pulse corresponds directly to the servo angle, with a 1.5ms pulse aligning the servo to its center point.

Even though most servo-motors are capable of 180 degrees of rotation, they tend to have different offsets built in, so the range has been limited to 100 degrees to eliminate the possibility of unintentional damage to servos by over or under-driving them, and there is no need to calibrate a neutral position.

Layer Code

The include file for the Servo layer contains three functions, two for turning a servo on and off, and one for rotating the servo to the desired angle. *(All of these functions are written for the PIC Foundation. The include files used with other foundations differ slightly, due to the presence of local variables.)*

The **servo-on** function takes one argument, the servo number to turn on, a value from 1 to 8. Within the function, a total of two values are actually sent to the servo layer. First an "8" is sent, indicating that we want to turn a servo on. Then, the actual servo number is sent. It should be noted that the servo number has the number one subtracted from it before it is sent to the layer, since the layer is actually expecting a value from 0 to 7. The reason for the modulo 9, is that we want to ensure that the user cannot inadvertently send a value greater than 7 in that spot, to avoid possibly confusing the layer itself.

```
to servo-on :number
    i2c-start
    i2c-write-byte $04
    i2c-write-byte 2
    i2c-write-byte 8
    i2c-write-byte (:number % 9) - 1
    i2c-stop
end
```

The **servo-off** function operates similarly to the write function, taking one argument, the servo number to turn on, which is a value from 1 to 8. Within the function, a total of two values are actually sent to the servo layer. First a "9" is sent, indicating that we want to turn a servo off. Then, the actual servo number is sent. Just as in the previous function, the servo number is scaled down to the 0 to 7 range, and is forced to be kept in the desired range by the use of the modulus operator.

```
to servo-off :number
    i2c-start
    i2c-write-byte $04
    i2c-write-byte 2
    i2c-write-byte 9
    i2c-write-byte (:number % 9) - 1
    i2c-stop
end
```

The **turn-servo** function is what actually makes the servo move by changing the positive pulse width of the signal pin. The function takes two arguments, the servo number to be turned, and the angle that the servo should be turned to. The angle argument is represented in degrees, with a maximum swing from 0 to 100 degrees. If a value higher than 100 is sent, it will be interpreted just as 100 by the pulse modulation code on the layer. By default, all servos will start at the 0 degree point when they are turned on until they are changed.

```
to turn-servo :number :angle
    i2c-start
    i2c-write-byte $04
    i2c-write-byte 2
    i2c-write-byte (:number % 9) - 1
    i2c-write-byte :angle
    i2c-stop
end
```

Examples of Use

Probably the most common thing to do with the servo layer is to just turn on a servo and start moving it around. Let's start by turning on the servo plugged into port 3:

```
servo-on 3
```

Pretty simple, huh? Well, it's almost as easy to move it:

```
turn-servo 3 50
```

The servo motor should now turn to its midpoint, since 50 is halfway between 0 and 100. We can write a simple loop to make the servo swing back and forth between its endpoints:

```
loop [turn-servo 3 0 wait 5 turn-servo 3 100 wait 5]
```

The servo motor will now start moving between its two extremes, switching positions every half second. The speed at which it turns is not user controllable, and is set by the actual servo itself. It is possible to buy servos of a variety of powers and speeds, with the tradeoff being that the faster servos are generally much weaker. If a servo is too fast, it's always possible to slow it down manually, similar to this:

```
setn 0
repeat 101 [turn-servo 3 n wait 5 setn n + 1]
```

This code first sets the global variable "n" to 0. Then, it loops 101 times, sending a turn-servo command with n as its angle argument, incrementing n each time. This would cause the servo to turn one degree every half second, giving the illusion of a much slower movement. We looped 101 times in order to have both the 0 and 100 positions included.

If you remember, we said that every servo is at the "0" angle position when it is turned on. In some applications, that can cause a problem, depending on how things are set up in the mechanical system. If you want to have a servo be at a different angle when it is turned on, its as simple as just calling the **turn-servo** function before the **servo-on** function:

```
turn-servo 3 75
servo-on 3
```

In this case, the servo will be at a 75 degree angle when it is turned on. The last thing we might want to do, is turn the servo off:

```
servo-off 3
```

Turning off the servo will not cause the servo to move, it will simply stop driving power into it. If nothing is pushing on the servo, it will stay exactly at the angle it had been at. However, it's easy to rotate by hand when it is turned off. For a comparison, try turning a servo by hand while it is turned on. You should find it very difficult, and feel the motor actively pushing back against your efforts.