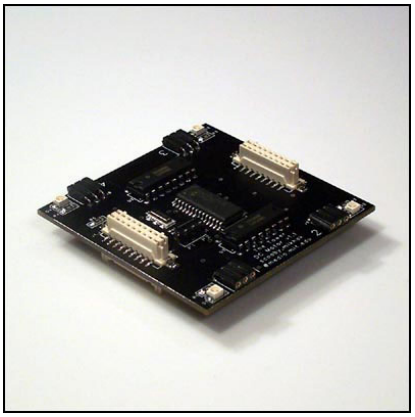
DC Motor Layer



Description

The DC Motor layer can drive up to four DC motors at high currents. Each motor can be individually controlled to drive in either direction, at a variable speed, and can be easily accelerated or decelerated over a desired period of time.

Hardware Detail

A full H-Bridge circuit drives each motor, in order to allow bi-directional high-powered control. Two L293D integrated circuits are used to drive two motors each. The on-board PIC controls the enable pins on each H-Bridge, to allow for high-voltage velocity control. The chips themselves contain the flyback diodes needed to prevent damage to the chip due to the field collapse when the motor's inductive load is switched off rapidly. The motors can be powered off of either the primary or secondary bus as selected by the switch on the layer, and can be driven up to 36 volts if desired. A bicolor LED near each motor port will light up red or green depending on the power and direction that the motor is being driven.

Layer Code

The include file for the DC Motor layer contains four functions, one each for turning the motor forwards and backwards, one for stopping abruptly, and one for just cutting power and letting the motor coast to a stop. *(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 **motor-on-forward** function is used to turn on a motor in the forward direction. The function takes three arguments from the user, the motor number (a value from 1 to 4), the desired speed (from 0 to 255), and the time in hundredths of a second (0 to 255) that it should take to ramp up or down to the new speed setting. A total of four arguments are being sent to the layer itself. First a "0" is sent, indicating that motor forward drive function is being called. Then, the three arguments are sent in order, and the motor will assume its new state.

```
to motor-on-forward :n :speed :time
    i2c-start
    i2c-write-byte 26
    i2c-write-byte 4
    i2c-write-byte 0
    i2c-write-byte (:n - 1)
    i2c-write-byte :speed
    i2c-write-byte :time
    i2c-stop
end
```

The **motor-on-reverse** function is used to turn on a motor in the reverse direction. The function takes three arguments from the user, the motor number (a value from 1 to 4), the desired speed (from 0 to 255), and the time in hundredths of a second (0 to 255) that it should take to ramp up or down to the new speed setting. A total of four arguments are being sent to the layer itself. First a “1” is sent, indicating that motor reverse drive function is being called. Then, the three arguments are sent in order, and the motor will assume its new state.

```
to motor-on-reverse :n :speed :time
    i2c-start
    i2c-write-byte 26
    i2c-write-byte 4
    i2c-write-byte 1
    i2c-write-byte (:n - 1)
    i2c-write-byte :speed
    i2c-write-byte :time
    i2c-stop
end
```

The **motor-stop** function is used to cut power to the motor, but will still allow it to coast to a stop. The function takes a single argument, the motor number (a value from 1 to 4), and sends it to the layer, right after a “2” is sent, signifying a motor-stop call.

```
to motor-stop :n
    i2c-start
    i2c-write-byte 26
    i2c-write-byte 2
    i2c-write-byte 2
    i2c-write-byte (:n - 1)
    i2c-stop
end
```

The **motor-brake** function is used to immediately stop the motor from turning. By turning on both forward and reverse drive simultaneously, the voltage across the motor becomes fixed, resisting further motion by fighting against the voltage created by its turning. The function takes a single argument, the motor number (a value from 1 to 4), and sends it to the layer, right after a “3” is sent, signifying a motor-brake call.

```
to motor-brake :n
    i2c-start
    i2c-write-byte 26
    i2c-write-byte 2
    i2c-write-byte 3
    i2c-write-byte (:n - 1)
    i2c-stop
end
```

Examples of Use

Using the DC Motor layer is as simple as just turning a motor on and off. Let's say we want to turn on the motor in port number 1 going forwards. We could say something like this:

```
motor-on-forward 1 255 0
```

The first argument to the function is saying that we want to turn on motor 1. The 255 being sent is indicating that we want it to go to full power, and the 0 says that we want it to happen instantly. Now, let's cut the speed by half:

```
motor-on-forward 1 128 100
```

This time, we sent a value of 128 for the speed setting, which is about half of the 255 we sent before. For the third argument, we sent a value of 100, indicating that we wanted to it to ramp down to this new speed over the course of one second. It is important to note that the motor may actually be spinning faster than half of its original speed. This is because the speed argument is actually a power setting, so if there is not a significant load on the motor, you won't see a huge speed change even if the power is cut in half.

Let's now tell the motor to go the other direction:

```
motor-on-reverse 1 255 200
```

When this code is run, the motor should switch directions, and begin ramping to full power in the opposite direction. The 200 argument at the end is telling it to take 2 seconds to reach that new speed. Let's cut power to the motor and let it coast to a stop:

```
motor-stop 1
```

The motor should slow down, and eventually stop. If you try moving the axel with your hand, you should find it easy to spin. For a comparison, let's turn the motor back on:

```
motor-on-forward 1 255 0
```

And now have it stop abruptly:

```
motor-brake 1
```

This time, the motor should stop spinning almost immediately. If you try to spin the axel by hand, you should notice that there is much more resistance to motion than there was in the previous case.