Lab 6: Controlling external devices — motors

Goals: The major goal is to interface the AVR to an external device, and have the AVR act as a controller for that device. The lab will use motors as the external devices.

In particular, it will interfacing the AVR to a L293D, to control a DC motor. It will use the same device, with different control code, to control a stepping motor.

Motor control

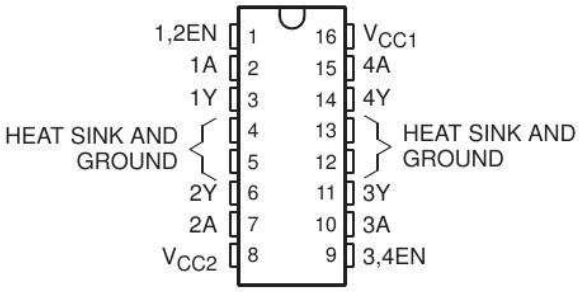
Controlling real-world devices is perhaps the largest application for embedded processing. The capability to perform complex control algorithms, possibly adapting to different environments, is a key application for such systems.

We will look at controlling two different types of motors commonly found in small and large control applications.

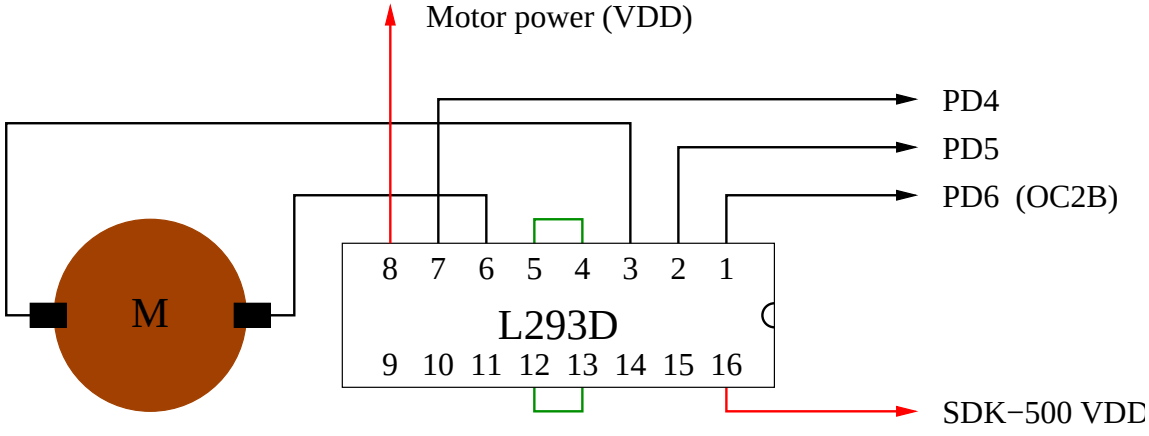
Driving a simple DC gear motor with the L293D

The L293D is a half H-bridge driver for DC electric motors. They can provide bi-directional control, and isolate the motor power supply from the logic driver (the microprocessor, in our case.) They are often used with pulse width modulation.

Following is the pinout for the L293D:



Construct the circuit depicted in the following diagram. Note that only output 1 and 2 are connected to the motor. Pin 8 is connected to the target power supply for the motor (a battery pack, in our case.) Pin 16 is connected to the logic power supply (the SDK-500, in our case.)



The speed of the DC motor can be controlled with Pulse Width Modulation (PWM). Write and test a program that accepts commands from a user that controls the motor's direction and the duty cycle of the PWM. The program should accept a line from the USART (terminated by a '`\n`'). The first character should be either an 'f' for forward or 'b' for backward. An integer specifying the duty cycle from 0 to 255 should follow the 'f' or 'b'.

A duty cycle of 100 percent is specified with 255. The `OC2B` (PWM output) is connected to the `enable1,2` line on the L293D. The period of the PWM is set to 8192μS.

Your program should be constructed with a main routine that accepts user commands via the USART. If the first character is a 'f', then "Input 1" should be 1 and "Input 2" should be 0. If the character was a 'b', then "Input 1" should be 0 an d "Input 2" should be 1. The following integer should set the duty cycle.

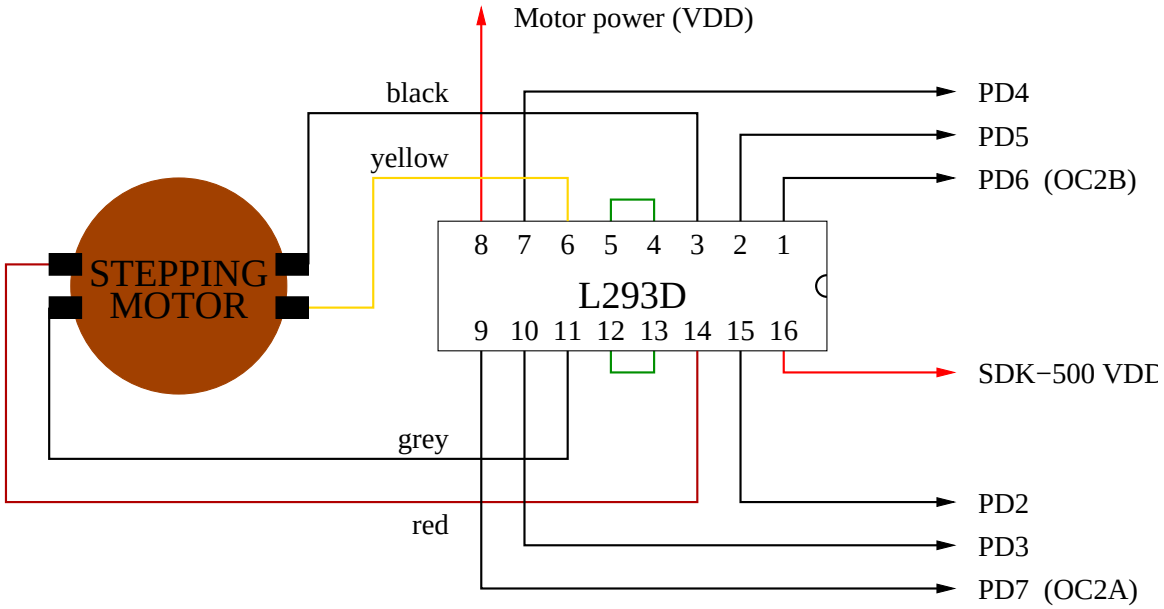
Test the performance of the motor for a reasonable set of input commands, and note the results.

Optional:

Add the required connections to drive a second motor using the same timer and L293D controller. The second motor speed and direction should be controlled independently of the first motor. This is a technique often used steer mobile wheeled robots.

Driving a bipolar stepping motor with the L293D

Connect the stepping motor as indicated in the following diagram. Note that only the motor connections change from the first section.



Using the program in the first section as a base, modify the program to accept as input a line from the USART containing a direction and a time period. The direction is indicated by the character 'f' or 'b', the time period by an integer. The integer determines the number of milliseconds between changes to the stepping motor control. A value of 10 would mean that the stepping motor would perform a step every 10 milliseconds. Configure timer 0 to interrupt the microcontroller every 1 mS.

The stepping motor should be programmed to perform full and half steps all together (e.g., the motor will go from a full step to half step position, and then to the next full step

position). Every time the specified period has occurred, the software will change the L293D to cause the next step (see the class notes).

For the motor used in the lab, the red–gray wire pair are terminals for one coil, and the yellow–black pair for the other coil.

The sequence for full step mode is:

Step	R	Y	G	B
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1
5	1	1	0	0
6	0	1	1	0
.	.	.	.	.

The timers need not be used to generate the steps for the motor; this can be done under program control by setting the appropriate bits in PORTD.

To reverse the motor, the sequence is applied backwards.

Test the performance of the motor for a reasonable set of input commands, and note the results. In particular, note what happens when the direction is changed.