

Project 1: Octave DC Motor Interface

Thomas Boynton

CET 346

3/8/2010

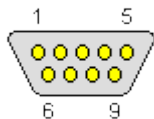
Project 1: CET 346

Abstract:

This project's purpose is to make an interface with the GNU Octave high level language and a DC motor. This motor should be able to be controlled at a constant speed even if the load and voltage are varied. The motor to be used will be # GM8224S021 12V Pittman Servo Motor. This motor will be controlled by a NTE 251 power transistor. The circuit will have a resistor in series with the computer's output pin and control the base circuit. The emitter will be connected to a ground and the collector will be attached to the motor's ground supply. The power side to the motor will be attached to its own 12V power supply. A snubber diode will be placed between the collector and emitter to protect the transistor from the inductive spike caused when the motor is turned off (channels power around the transistor). The output of the computer will be controlled by the Octave program and the necessary script will be written to enable this motor control.

Project Set-up and Build

When starting this project, the first step is planning the circuit to be interfaced with the computer. The second step will be to build this interface and the third step will be to try and control this interface with the Octave language. I start with trying to figure out from which port to control the motor. I chose the serial port DB9 connector instead of the USB as I can easily identify it as the COM1 port; I have a serial cable I can use for the interface



DB-9 Pin	IDC internal pin name*	Name	Dir	Description
1	1	CD	←	Carrier Detect
2	3	RXD	←	Receive Data
3	5	TXD	→	Transmit Data
4	7	DTR	→	Data Terminal Ready
5	9	GND	—	System Ground
6	2	DSR	←	Data Set Ready
7	4	RTS	→	Request to Send
8	6	CTS	←	Clear to Send
9	8	RI	←	Ring Indicator

The servo motor to be used with the computer interface control is a cylindrical, inch and three quarter diameter motor with a black and red twisted pair coming from the rear of the tube and what appears to be an encoder device on the rear of it. This encoder has a five wire connector on it. A bracket is then fabricated to hold the motor in place on the bench. The design that is chosen is just a simple aluminum L-channel with the appropriate mounting holes machined in it. This gives the user the option of whether to simply rest it on the bench for test or holes can be drilled in the bracket to fix it to the assembly it will be used in.

To begin to understand the problem at hand, we have to start by researching the motor that we have. Souping up the computer in the Utility Muffin Research Kitchen we enter the model number; it is an Ametek/Pittman part number GM8224S021. After a few hours of searching the Internet, it is determined that it is an 8224 Series 12V Brush DC Servo Motor with a spur gearbox on the front and a Pittman E30B Incremental Optical Encoder on the back of it. The pin-outs are obtained for the optical encoder and are listed below. The twisted black and red wires on the motor are for motor control.

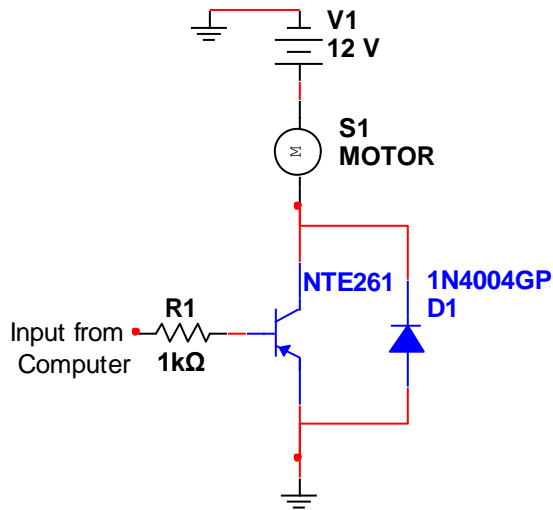
E30B Optical Encoder Motor Wire Designations

E30B Connection Chart		
Pin	Color	Function
1	Black	Ground
2	Green	Index
3	Yellow	Channel A
4	Red	Vcc
5	Blue	Channel B

Now that the pin/wire designations are known and the motor voltage is known, a circuit will be designed for it. Since the circuit coming from the computer is a low current signal circuit, the motor has to be controlled with something that has enough current to control the DC Servo motor

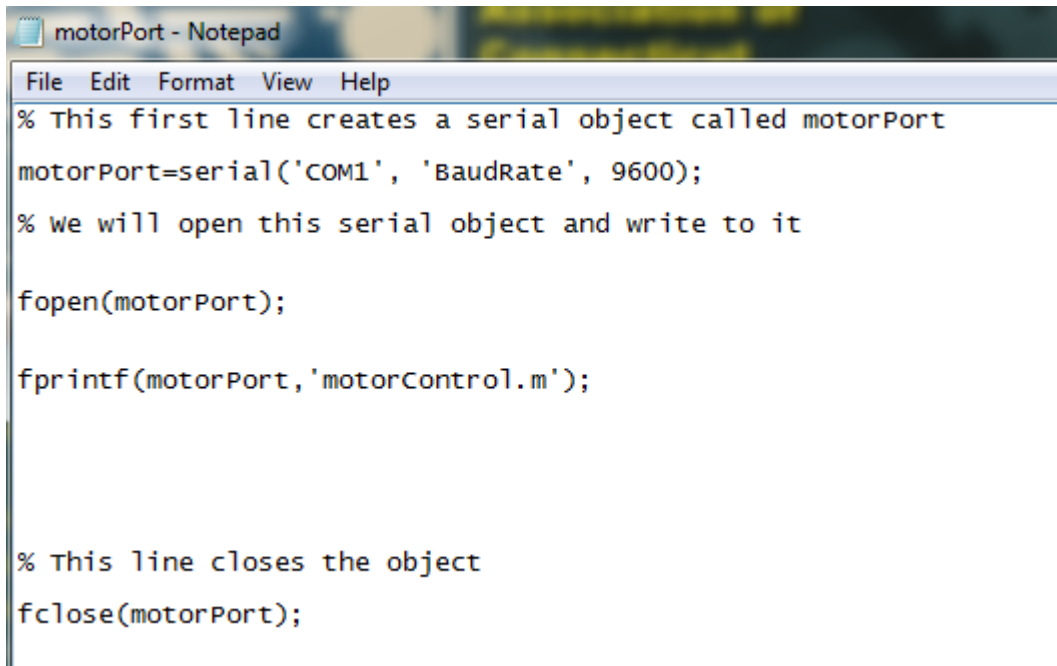
A transistorized design is chosen with the motor positive lead going to an external 12 volt source and the negative motor lead being switched to ground through the transistor's collector-emitter circuit. The base voltage will be cycled by the computer's serial data port through a current limiting resistor. The transistor will be protected from the inductive kick caused when the motor is shut off by a snubbing 1N4004 diode connected in a reverse bias fashion across the collector emitter circuit. This kick will be conducted to ground. The hardest part now, is figuring out the RS 232 protocol and how to control this output using the Octave interface. The trick is use an Octave outputted square wave and control the duty cycle of this wave, more "on time" the faster the motor spins. The Channel A of the encoder can give feedback to the computer about actual motor speed. The initial motor circuit is shown on the next page.

Motor Control Circuit



Pin three of the serial data cable is the transmitting line, pin 2 is the receiving line and pin 5 is system ground. In this circuit we will attach the input to the transistor circuit to pin three of the COM1 port and ground and attach the output from Channel A to pin 2 of the COM1 port and ground.

Next the Octave interface has to be written. We now have to figure out how to interface with the serial port, an object gets created in Notepad and is assigned to the COM1 port. Our initial file looks like the one shown on the next page. Once the port is opened it will have to send a pulse width modulated cycle with the duty cycle somehow controlled by feedback from the optical channel A in the receiving line in the serial port.



```
motorPort - Notepad
File Edit Format View Help
% This first line creates a serial object called motorPort
motorPort=serial('COM1', 'BaudRate', 9600);
% we will open this serial object and write to it

fopen(motorPort);

fprintf(motorPort, 'motorControl.m');

% This line closes the object
fclose(motorPort);
```

In this script the serial port is opened and is written to with a file called motorControl.m. In this motor control file, the function square from Octave can be written to generate a square wave and the output square wave from the optical encoder can be read. Then an “if” statement can be written to have the duty cycle on these two match after the incoming square wave is conditioned to have the same limits. The square wave function is shown below:

$$s = \text{square}(t, \text{duty})$$

This generates a square wave with period 2π with limits $+1/-1$.

If the duty cycle is specified, the square wave is $+1$ for that portion of the time

$$\text{Duty cycle} = \frac{\text{on time}}{\text{on time} + \text{off time}}$$

While I have not quite figured out how to control this motor and have still not made it spin; given the time, this is what I would accomplish. I will first attempt to put a signal on the transistor input and see if I can verify that the motor turns and puts out a signal from the optical encoder. Next I can see if I can see an output on the oscilloscope coming from the transmit line on the computer output. Once I can get these two items verified, I can then begin the adjustments to the code to fine tune the motor speed to my desired level.