Encoder Interfacing

Purpose

- To introduce the concept of an encoder for determining rotational speed and angle
- To develop familiarity with the IRL510 power MOSFET to control power devices
- To reinforce the concept of Pulse-Width Modulation to controlling the speed of a DC motor
- To explore the pulse counting capability of the OOPic microcontroller

Components

Qty.	Item
1	Oricom Technologies OOBOT 40-II robot controller board and serial port cable
1	1M Ω resistors
1	1N4148 diode
1	IRL510 power MOSFET
1	Pittman DC motor with Encoder

Introduction

In this lab you will investigate how rotational speed and rotational angle can be determined using a rotary encoder. A rotary encoder consists essentially of a disk with alternating opaque and clear radial regions. In operation, a light source is positioned on one side of the disk, and a photosensitive device, such as a phototransistor is positioned on the other side of the disk. As the disk rotates, the passage of the opaque and clear regions of the encoder disk alternately block and allow light to impinge on the receiver, which produces corresponding voltage pulses. The rotational speed of the encoder disk can be determined by counting pulses during a known time period. The angle of rotation corresponds directly to the number of pulses, since the number of pulses per revolution is constant.

We will also look at controlling the speed of a DC motor using the concept of Pulse-Width Modulation (PWM).

Procedure

Function Generator Output to Control Duty Cycle

1. Set up the function generator (FG) to output a 1 kHz square wave (remember to set the output termination to HIGH Z). Look at the signal on the ‘scope.
2. Set the amplitude to 5 V p-p
3. Offset the waveform by 2.5 V, so that you have a square wave that goes between 0 and 5 V.
4. Select the ‘% Duty’ function on the FG by pressing the Shift key, then the Offset key.

5. Rotate the knob and see what happens to the output waveform when you vary the duty cycle. **What are the limits you can set the duty cycle to? What does "duty cycle" mean? Describe in your own words.**

DC Motor Speed Contol Using Duty Cycle

6. Build the circuit shown in Figure 1. Don't forget the diode. **Important note**: MOSFET's are ***very*** sensitive to static electricity. Make sure that you are not carrying static charge before you handle these devices. It is best to work on a properly grounded anti-static surface with an anti-static bracelet on your wrist. If these precautions are not available, then discharge yourself by touching a grounded metal surface (such as the frame of the bench) before you handle a MOSFET. Always handle a MOSFET by its large metal tab and NOT by its leads.

   The MOSFET you will be using in this lab is the IRL510. The IRL510 can be fully turned on by logic level signals (+5 V), whereas most MOSFETS that can handle similar power are fully turned on when V_{GS} is approximately 10 V. The package style for the IRL510 is an industry standard TO-220. This package is somewhat awkward to use in a solderless breadboard, because the leads are relatively large. To avoid damaging the solderless breadboard, insert the IRL510 so that its metal tab is ***parallel*** to one of the 5-hole rows, but with each lead in a separate row. To do this, you will have to bend the leads slightly (see Figure 1).

   Why is the diode needed across the motor leads?

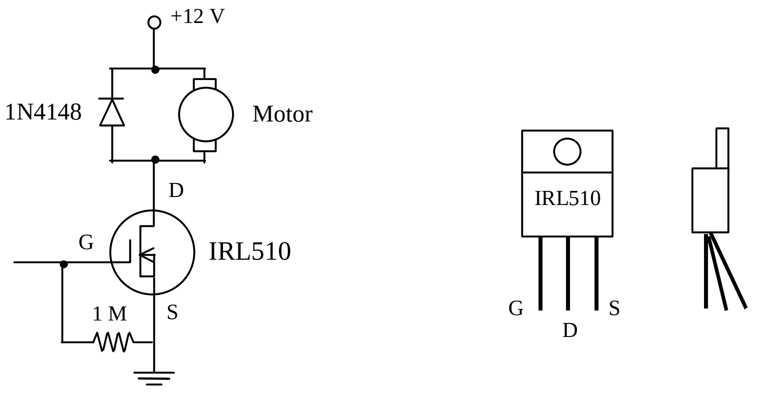


Figure 1 DC Motor Driver Circuit. The circuit uses an IRL510 power MOSFET. MOSFET's are very static sensitive, so handle them by the metal tab.

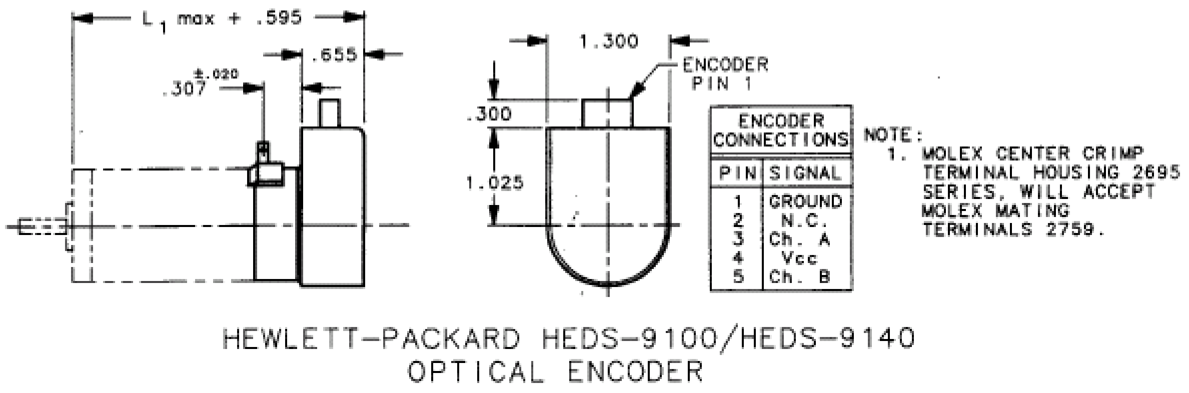
7. With the FG set up from Step 5, clip the red mini-hook lead from the function generator cable to the gate of the IRL510 and the black mini-hook lead to the common ground. Vary the duty cycle of the signal from the FG, and observe what happens to the motor. **Explain why the motor speed varies with the duty cycle.**

Using an Encoder with the OOPic Qencode Object

8. Unhook the red lead of the FG from the gate of the IRL510.

9. Using the information provided in Figure 2 below; connect +5V and Ground from the OOPic (use the pin nearest C5 and any of the servo ground pins respectively) to the corresponding pins of the motor's encoder (**Hint:** You may want to transfer the pins from

the encoder to your breadboard by using jumper wires. Then you can easily connect the pins on the OOPic to the desired pins of the encoder via the breadboard.). This will power the encoder's optointerrupter and provide 0-5V signals on channels A and B as the encoder wheel turns. Since this is a "quadrature encoder," the signals from channels A and B are 90 degrees out of phase. **Explain how the two signals from channel A and channel B can be used to determine the direction that the motor spins.**



ENCODER CONNECTIONS	
PIN	SIGNAL
1	GROUND
2	N.C.
3	Ch. A
4	Vcc
5	Ch. B

NOTE:
1. MOLEX CENTER CRIMP TERMINAL HOUSING 2695 SERIES, WILL ACCEPT MOLEX MATING TERMINALS 2759.

Figure 2 DC Motor with Encoder. The above picture shows the motor/encoder from the ***bottom*** view with Pin 1 of the encoder on the ***right***.

10. Now, connect the 'scope to the Ch. A and Ground pins of the encoder.
11. Power the OOBOT controller: Use the power cable, a red-green twisted pair with stripped leads on one end, and a connector on the other. Make sure that the power supply is turned OFF, then connect the stripped end of the red lead to the +12 V terminal of the power supply, and the stripped end of the green lead to the COMMON of the power supply. The connector at the other end of the twisted pair plugs into the power input terminal and can do so in only one orientation so as to avoid switching + and -.
12. Re-connect the red mini-hook of the FG to the gate of the IRL510, and run the motor as in Step 7. Observe the output of channel A on the 'scope. **Determine the speed of the motor at 5 different duty cycles that span the full range that the FG can output. (Hint: you will need to know how many counts per revolution the encoder wheel provides). Plot motor speed vs. % duty cycle in your report, and discuss your results.**

OOPic Measurement of Encoder Pulses

13. Disconnect the red mini-hook of the FG and turn off the 12 V power supply. Make sure that the OOBOT controller is **OFF**. Connect channel A of the encoder to B6 of OOBOT and channel B of the encoder to B7.
14. At this time, connect the serial cable to the COM port on your computer and to the OOBOT's COM port connector.

15. After **<u>DOUBLE CHECKING</u>** all of the connections to the OOBOT, turn on the 12V power supply to power up the OOBOT. Check to make sure that the power LED on the OOBOT is on. If the light does not come on, see the TA for help. Do <u>not</u> proceed if the board does not power up!
16. We are going to use the **oQencode** object to keep track of the encoder pulses from the motor. Use the OOPic help files to become familiar with the oQencode object. Also, we will use an **oEvent** object in a "Virtual Circuit" that will cycle once every second. The oEvent's interrupt routine will use a second oQencode object to keep track of the number of encoder ticks in every second. This can be translated to the motor's speed in encoder ticks per second. Finally, we will display the values for encoder position and speed by sending them to the serial port with the **oSerial** object. You can use a terminal program such as HyperTerminal or the Comm Control provided in the OOPic IDE to view the output from the serial port. If using HyperTerminal, configure the COM port to 9600, 8, N, 1, N and the ASCII settings should ***not*** append line feeds to incoming line ends.
17. Enter and run the following OOPic program using the C compiler. Then open the Comm Control or HyperTerminal window.

```
‘Tachometer Program
‘Put your name here
‘Put the date here

oSerial A = new oSerial;
oQencode Encoder = new oQencode;
oWord Speed = new oWord;
oQencode EncCount = new oQencode;
oEvent Clock = new oEvent;
oGate Wire = new oGate;

sub void main(void)
{
    // Initialize Variables and Settings
    OOPic.pullup = 1;
    A.Baud = cv9600;                    // set baud rate
    A.Operate = cvTrue;                 // turn on com port

    Encoder.IOLine1 = 14;               // IO line 14 (OOBOT Pin B6) is selected
    Encoder.IOLine2 = 15;               // IO line 15 (OOBOT Pin B7) is selected
    Encoder.Signed = cvFalse;           // Uses an unsigned integer
    Encoder.Operate = cvTrue;           // Turns on Encoder object

    EncCount.IOLine1 = 14;              // Will be used to count pulses for the motor speed
    EncCount.IOLine2 = 15;
    EncCount.Signed = cvFalse;
    EncCount.Operate = cvTrue;
    EncCount.Value = 0;

    Wire.Input1.Link(OOPic.Hz1);        // Virtual Circuit link to OOPic.Hz1 object
    Wire.Output.Link(Clock.Operate);    // Virtual circuit link to Clock Event
    Wire.Operate = cvTrue;              // Turn on Virtual circuit

    while (1)
    {
        A.String = str$(Encoder);
        A.Value = 9;                    // ouput a TAB character to the serial port
        A.String = str$(Speed);
```

```
            A.Value = 13;                  // output a carriage return to the serial port
        }   // end while
    }   // end main

    sub void Clock_Code(void)
    {
        Speed = EncCount;
        EncCount.Value = 0;                // reset EncCount to zero
    }   // end Clock_Code
```

Reconnect the red mini-hook of the FG to the gate of the IRL510. **Explain how this program works. Compare the speed you measure with the 'scope with the value your program outputs. How well do these values agree?**

Suppose the encoder were mounted to the shaft of a motor or the wheel of a vehicle. If the diameter of the wheel is 3 inches, write a program that will indicate the rotational speed of the wheel and tell how far the vehicle has traveled. (In order to accomplish this, you will need to know how to work around the fact that OOPic deals with integer arithmetic only. If you need to multiply a variable by a constant that has a fractional part, like π (pi) for example, use the * operator, and represent the constant as a two-byte number. The whole number portion will be the upper byte, and the fractional portion will be the lower byte in units of 256. So for pi, the whole number portion is 3, so the upper byte will be 0x03. The lower byte will be 0.14159*256, which is about 36 or 0x24. So if you need to multiply a variable by pi, use the * operator, and multiply by 804.)

18. How accurately can you calculate the speed and distance? Quantify your answer.

Run the program from step 18 with the setup you just used.

Acknowledgment

We greatly appreciate the support of Pittman, Inc. (http://www.pittmannet.com), who donated the motors used in this experiment.