EE-3306 HC6811 Lab #6

DC Motor Control Project

The purpose of this lab is to synthesize the programming concepts learned in the previous 68HC11 labs along with some circuit design into a working controller for a DC Motor with user input from the LCD display. The PWM modulation technique learned in the third lab will be used to control the speed of the motor while pulse detection will also be implemented to detect the RPM of the motor.

Objectives

- Understand how to code the microcontroller to run a DC motor.
- Understand how to use the 68HC11 to detect pulses.
- Understand the circuitry used to drive a DC Motor using the 68HC11.
- Apply Pulse Width Modulation (PWM) to the application of controlling the speed of a DC Motor.

Prelab:
Answer the question given in the website and also design the circuit.

References

Equipment for this lab:
1. 68HC11 trainer kit, to include 68HC11 EVBU and prototyping strips
2. IBM compatible PC to connect to the trainer kit via an RS-232 serial cable
3. Agilent E3649A DC power supply
4. Agilent 54621D oscilloscope
5. Floppy disk provided by the student

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Part Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photo Emitter</td>
<td>160-1063-ND</td>
<td>1</td>
</tr>
<tr>
<td>Photo Detector</td>
<td>160-1065-ND</td>
<td>1</td>
</tr>
<tr>
<td>Mounted DC Motor with Slotted Disk</td>
<td>FE-260-18130</td>
<td>1</td>
</tr>
<tr>
<td>NPN General Transistor</td>
<td>2N222A</td>
<td>≤2</td>
</tr>
<tr>
<td>Rectifier Diode</td>
<td>1N4004</td>
<td>1</td>
</tr>
<tr>
<td>Resistors and Capacitors (Varying Sizes as Needed)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Laboratory Exercise

Timing
It is anticipated that this lab will take two weeks to complete.

Notes
1. In this lab, all numbers such as addresses and data are given in hexadecimal format (“hex”) unless otherwise indicated. In completing the lab, record all information in hexadecimal unless directed differently.
2. Make sure your development EVBU board is connected to power (green LED on board ON) and the serial port of the EVBU is connected to the serial port of your development PC containing the AXIDE software, configured to the correct port at the correct baud rate, etc.
3. Be sure to bring your RTF file from the Timers lab. The final code you submit should be named DCMotor.RTF.
4. The motor’s speed should not exceed 6000 RPM. If the motor is shaking enough to move around the table, this is a good indication that the motor is moving too quickly.
5. IMPORTANT: Do not touch the slotted disk while it is spinning. Doing so could result in bodily harm.

Part 1—Fully Functional DC Motor Control

In this lab you will create a control interface and use the knowledge from lab 3 to create a PWM square wave to drive a DC motor. To accomplish this, you will need to write a code that allows the user to control the speed of the motor manually. This is done by altering the duty cycle of the PWM wave proportional to the speed entered by the user. This will be dictated by the equation used.

\[ \text{Duty Cycle} = A \times \text{Speed} + B \]

In this equation, A and B are constants. For convenience, B is set to be ‘0’. The value of ‘A’ is chosen as 200. Your program will ONLY need to handle POSITIVE speeds. The OC3 function is used to create the square wave with the duty cycle proportional to the speed entered by the user. Note that the number that the user can enter can only be less than 200, since the maximum duty cycle for a period of 40000 is 99%.

The diode in Figure 1 is included because of the use of Pulse Width Modulation. During the period of time at which the 68HC11 output is logically low, it creates an alternate path for any small current that can drive the motor. However, the addition of the diode allows current to continue to flow through an alternate path.
Program:
The following steps are to be followed in order to create a PWM wave in accordance with the speed entered by the user. The output is then used to drive the DC motor through the circuit as shown in the figure.

Step1:
- Assign labels to all the registers which will be used in the program.
- Set the absolute addresses for the interrupt vector subroutine and the main program.
- Allocate memory to different variables that will be used in the program.
- Set up a lookup table for the characters on the keypad.
- Allocate memory to the messages to be displayed on the screen.

Step2:
- The main program starts with initializing the LCD, PORTD and PORTA
- Configuring the control registers for the output compare function (OC3).
- The registers that are to be configured are TCTL1, TMSK1 and TFLG1.
- The OC3 should be set to toggle through the TCTL1 reg.
- The OCI pin and OCF pin should be set using the TMSK1 and TFLG1 registers respectively.
- Initialize the ‘which’ variable to ‘1’.
- Enable the interrupts using the global interrupt enable.

Start a loop within the main program where in actually the input is obtained from the user and processed.
The welcome message should be first displayed followed by the message which prompts the user to enter the speed.
The speed entered by the user should be terminated by a ‘#’ sign so that the controller knows that the speed is entered completely.
The speed entered should be converted to decimal from hexadecimal format since one digit is considered as input at a time.

This is done by adding the new incoming digit to the previous number multiplied by 10.

For ex: The speed entered is as follows:

23 #

The code should be written such that 2 is multiplied by 10 and 3 is later added to the product i.e, \((2\times10)+3=23\)

Multiply the speed by 100.

Multiply it again by 2 using the shift instruction.

Assign this number to the ‘high’ variable to update the wave.

Subtract this value from 40000 and store it in ‘low’ variable.

The int_TOC3 interrupt vector subroutine is similar to what was done in the previous labs i.e, to create a square wave with a given duty cycle. Repeat the same thing.

Note: You will need all the subroutines you used in the previous labs in order to display messages, get input from the keypad..etc.

**Part 2**

In the second part you will be writing a separate code to calculate the RPM of the motor. In addition to the code you will need a circuit to calculate the RPM.

The circuit that you will need will be a pulse detector circuit using the photo emitter-detector pair provided. The disk attached to the motor will spin between the emitter and detector, blocking the light except when the slot passes by them. The emitter of this pair is simply a light emitting diode (LED). When the light from the emitter strikes the detector, a voltage is created across the terminals of the transistor (the light basically takes the place of a base current). Your program must then use the output of the detector as input for the 68HC11 to read in the generated pulses. These pulses can then be used to calculate the RPM of the motor and display it on the LCD display. The general circuit is shown in Figure 2. Please enter the values of R1 and R2 used in your circuit below (you just design the circuit so that \(I_r=20\ mA\)).

**Program:**

To find the RPM of the motor the PACNT register is used. When the slot passes between the emitter and detector a signal is obtained indicating that one revolution of the motor is completed. This signal is sent as an input to the controller through the PAI pin. The signal acts as a clock input to the counter PACNT. Thus, PACNT counter keeps count of these signals. By reading this counter every second, the RPM of the motor for every second can be obtained. The counter PACNT is enabled by setting the PAEN pin the PACTL register.
The program most importantly needs the following:

1. A new subroutine to display the RPM (or a modification of the subroutine that displays the speed and approximation type).
2. A new portion of code within the interrupt subroutines that counts the number of pulses accumulated over a known period of time.
3. 

The following steps should be followed while writing the code:

- Assign labels to all the registers which will be used in the program.
- Set the absolute addresses for the interrupt vector subroutine (int_TOC2 in this case) and the main program.
- Allocate memory to different variables that will be used in the program.
- Allocate memory to the messages to be displayed on the screen.

The main program begins with displaying the welcome message and configuring the registers TCTL1, TMSK1, TCTL1 (for TOC2) and PACTL for RPM calculation. A variable ‘rpm’ which is set aside to store the RPM value is initialized to 0. Use the global enable interrupt.

**int_TOC2 (Interrupt vector subroutine):**

- Use the counter variable ‘cnt’ to count down from 100 to 0 to keep count of the number of seconds. Since the counter value chosen is 100, interrupt is generated every 10ms.
- Once the counter reads 0, one second is complete and at that point read PACNT and store the value read in the variable ‘rpm’.
- Multiply the ‘rpm’ value by 60. This will give you the revolutions per minute.
- Initialize the value of ‘cnt’ to 100.

Note: Remember to clear the flag bit in TFLG2 and store the value 20000 in TOC2 every time an interrupt is generated (every time ‘cnt’ variable decrements) at the end of the subroutine.

**Displaying the ‘rpm’:**

- The character string ‘RPM:’ is displayed using the ‘cputs’ subroutine.
- This is followed by the ‘rpm’ variable after converting it to decimal format.

This subroutine can be called within the ‘int_TOC2’ subroutine.

The rest of the subroutines needed for the program can be obtained from the previous lab codes.
Figure 2: RPM Detector Circuit

Demonstrate to the T. A. that your program is running. T. A. Initials:
______________________

Part 3—Questions

When you turn in the lab report, please answer the following questions. Provide flow charts and/or assembly language programs as necessary to explain your answers.

11. What did you find easiest about this lab? What was most challenging?

22. Take at least 5 points of data for varying speeds. Name one way of more accurately controlling the speed of the motor so it reflects the speed entered by the user.

33. Take at least 10 points of data for the RPM calculated by your program and the RPM calculated from the oscilloscope. What could cause a difference between the actual and computed RPM? Draw a circuit that could aid in the accurate counting of pulses.

14. The circuit designed in the prelab was only one way of stepping up the current to drive the motor. Design an alternate circuit and name at least 2 advantages and 2 disadvantages of your new circuit as compared to the one designed in the prelab (hint, you can use online resources such as www.digikey.com to determine parts, prices, electrical characteristics, etc.)