ECE 362 Lab Verification / Evaluation Form

Experiment 10

Evaluation:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
1	Interfacing (completed prior to your scheduled lab period)	**5**	
2	Software (completed during your scheduled lab period)	**10**	
3	Submission (completed following demonstration)	**5**	
TQ	Thought Questions	**5**	
Bonus	"The Hummer"	**5**	
	TOTAL	**25+**	

Signature of Evaluator: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. *You will not receive credit for this lab experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Printed Name: _____ Class No. __ __ __ __ - __

Signature: _____ Date: _____

Experiment 10: D.C. Motor Speed Control and Digital Tachometer

Instructional Objectives:

- To illustrate a real-time embedded control application and the use of interrupts
- To provide experience using various 9S12C32 integrated peripherals, including the pulse width modulation (PWM) unit, the analog-to-digital (ATD) unit, the serial communications interface (SCI), the pulse accumulator (PA) unit, and the serial peripheral interface (SPI)

Parts Required:

- 4N28 optical isolator (included with DK-3 parts kit)
- TIP 120 or TIP 122 NPN transistor (included with DK-3 parts kit)
- 2 x 16 LCD and 16-pin single-row header (previous experiment)
- GAL22V10 programmed as an 8-bitshift register (previous experiment)
- Breadboard and wire

Preparation:

- Read this document in its entirety
- Review material on the ATD, PWM, TIM/PA, SCI, and SPI
- Create your solution in "C" using Code Warrior

NOTE: The "C" code for "main" and the ISRs must be written and debugged ***during your scheduled lab period*** (LCD device driver should be written and tested as part of your pre-lab preparation.)

1. Introduction

In lecture you have been introduced to a number of peripheral subsystems included on the 9S12C32 microcontroller. The best way to understand how these independent functional units operate, as well as to gain an appreciation for how they might be utilized in real-world applications, is to illustrate their use in a real-time control experiment. In this lab you will investigate the use of several key HC(S)12 peripherals: the pulse width modulation (PWM) unit, the analog-to-digital converter (ATD), the serial communications interface (SCI), the pulse accumulator (PA) unit (itself part of an extensive timer subsystem), and the serial peripheral interface (SPI). In this real-time application, you will control the speed of a small D.C. motor using a pulse-width modulated signal derived from an analog input control voltage.

Your solution should utilize the two pushbuttons ("PAD7" and "PAD6") on the docking module: the left pushbutton ("PAD7") should stop the motor (if it is running), while the right pushbutton ("PAD6") should start the motor (if it is stopped). The left LED (connected to port pin "PTT1") should be on when the motor is stopped; the right LED (connected to port pin "PTT0") should be turned on when the motor is running.

The 3-digit (geared down) drive shaft speed (in RPM) as well a percent-of-maximum bar graph will be displayed on an LCD (as shown below). An external shift register is used to interface the LCD to the microcontroller via the SPI module (MOSI, port pin PM4; and SCK, port pin PM5), as in Experiment 8.

2. Software Overview

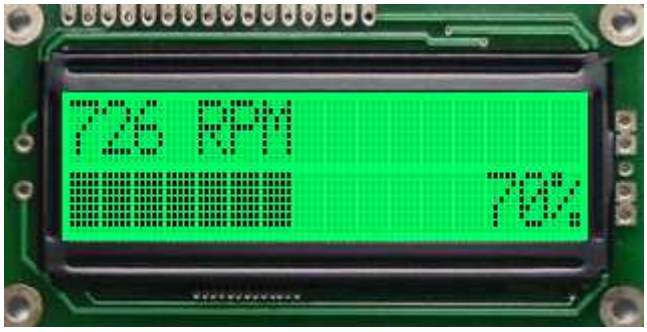
The objective of this experiment is to control the speed of a small D.C. motor using pulse width modulation. The PWM duty cycle will be a function of an input analog D.C. voltage (in the range of 0 to 5 volts). The motor speed will be determined by the number of pulses detected by the pulse accumulator; its instantaneous value will be estimated based on the number of pulses accumulated from the motor's 64-slot "chopper" over a one-second integration period. An updated calculation of the (geared-down) drive shaft RPM (in BCD format) will be displayed on the terminal screen once every second. The motor, to which the chopper is attached, drives a gear-head, which then drives the external shaft; the gear ratio is approximately 28:1. Therefore, the three-digit value displayed for the drive shaft speed should be the motor speed divided by 28.

The timer module (TIM) will be used to establish the ATD sampling rate as well as the tachometer integration period and display update rate. The real time interrupt (RTI) will be used to establish the pushbutton sampling rate.

The SCI will be used to print the RPM of the motor to the terminal in a buffered, interrupt-driven fashion. To print characters you should call `bco` which writes them into the buffer `TBUF`. Also in `bco`, the transmit interrupt should be enabled so that the buffer can be emptied. The SCI transmit ISR will send out characters when the buffer is not empty. When the transmit ISR encounters an empty buffer, it should turn off the SCI transmit interrupt so that no unintended characters are sent. This methodology allows the processor can do other things in-between character transmissions.

The RTI will be used to sample the pushbuttons on the docking board every 2.048 milliseconds. Data for the LCD display will be shifted out to an external shift register (GAL22V10) and will be interfaced to the SPI module through Port M (PTM).

You should reuse the LCD driver you created for Lab 8 in this experiment for all interfacing. On the first line of the LCD, you should display the RPM, updated every second. On the second line of the LCD, one box character (ASCII $FF) should be displayed for each 10%. Additionally, the percentage-of-max should be right aligned. Noting the fact that each line of the LCD is 16 characters will be useful in doing this.



NOTE**:** *Before generating the bar graph, the maximum RPM must be determined.*

3. Hardware Overview

The basic interface circuit you will need to construct is illustrated in Figure 1. PWM output Channel 3 (routed to port pin PTT3 based on MODRR register setting) is used to drive an (optically isolated) NPN switching transistor. The PWM sampling frequency should be approximately 100 Hz. Attached to the motor shaft is a 64-slot "chopper" disk that passes through an optical detector. Signal conditioning circuitry included with the motor assembly produces a pulse each time a hole in the disk passes through the aperture of the optical detector. The signal produced by the optical detector is fed to the pulse accumulator input PTT7. Finally, a potentiometer is used to provide a reference D.C. voltage (in the range of 0 to 5 volts) for ATD input Channel 0 (PAD0). This value, when digitized, is used to control the PWM duty cycle.

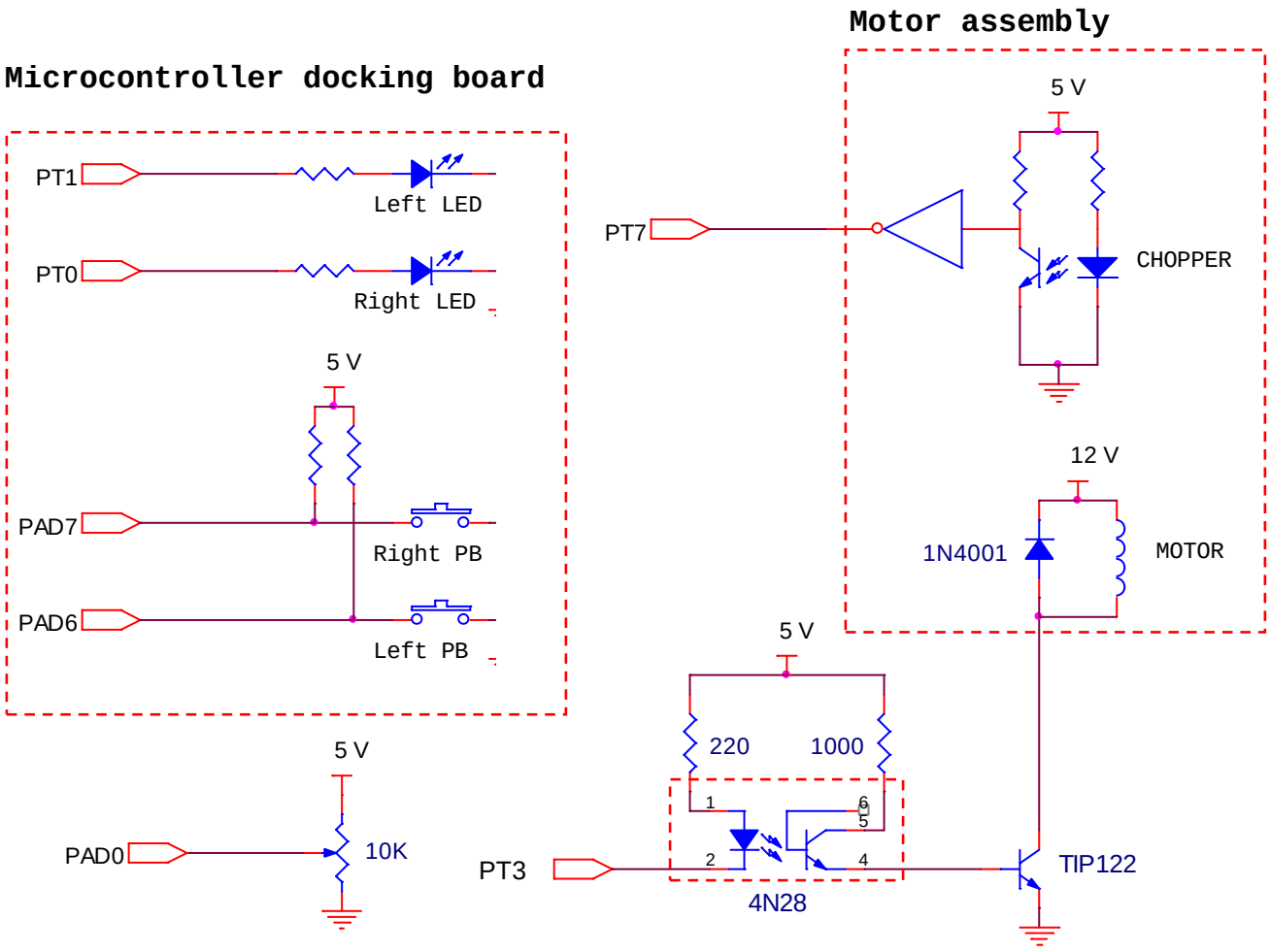


Figure 1. Wiring diagram for potentiometer and motor interface.

NOTE: *Motor assemblies will only be available for checkout during times that a course staff member is on duty. Debug your software using an oscilloscope (to view PWM output generated on PT3) and a function generator (connected to PT7) to simulate the chopper.*

You will interface to the LCD in the same way that you did in Experiment 8 (except for the Port T connections) The description of the LCD interface is reproduced below for your convenience.

An external 8-bit shift register (GAL22V10) will be used to interface LCD to the microcontroller via the SPI module (MOSI, port pin PTM4; and SCK, port pin PTM5). The LCD will be interfaced as described to the microcontroller module as described in the table below:

LCD Pin #	LCD Pin Description	Connected to Microcontroller
1	Vss (ground)	Vss (ground)
2	Vcc (+5V)	Vcc (+5V)
3	VEE (contrast adjust)	Vss (ground)
4	R/S (register select)	PTT4
5	R/W' (LCD read/write)	PTT5
6	LCD Clock	PTT6
7	DB[0] (LSb)	Q[0]
8	DB[1]	Q[1]
9	DB[2]	Q[2]
10	DB[3]	Q[3]
11	DB[4]	Q[4]
12	DB[5]	Q[5]
13	DB[6]	Q[6]
14	DB[7] (MSb)	Q[7]
15	Not connected	
16	Not connected	

NOTE: Port T connections are different than those used in previous experiments.

Some of the LCD pins require a bit more explanation:

Mnemonic	Name	Description
RS	Register select	This pin is logic 0 when sending an instruction command over the data bus and logic 1 when sending a character.
R/W'	Read/write	This pin is logic 0 when writing to the LCD, logic 1 when reading from it. For this lab, we will only write to the LCD.
LCDCLK	LCD clock	This pin latches in the data on the data[7:0] bus on the falling edge. Therefore, this line should idle as logic 1.

Step 1. Interfacing

Interface the LCD and LEDs to your microcontroller kit as described in the Hardware Overview section **(note that different Port T pins are used than in previous experiments – see table)**. This will require programming your GAL22V10 to function as an 8-bit shift register, as described in problem 5 of the Module 2 homework. Complete all the interface wiring on your breadboard as part of your pre-lab preparation.

Step 2. Software
Prior to your scheduled lab period, copy the LCD device driver routines you wrote for Lab 8 and test them with the interface circuitry completed for Step 1. Complete the remainder of the "C" skeleton file provided on the course website ***during your scheduled lab period.*** Note that the "finished product" should work in a "turn key" fashion, i.e., your application code should be stored in flash memory and begin running upon power-on or reset. Demonstrate the completed motor speed control system to your lab T.A.

Step 3. Submission
Submit your ".C" solution file on-line after demonstrating it to your lab T.A. (but before leaving lab). ***Be sure identifying information*** (i.e., name, class number, and lab division) ***is included in the files you submit – credit will not be awarded if identifying information is omitted.***

Bonus Credit
Add a "musical" (frequency modulation) mode to the motor speed control function. Here, use the left pushbutton to toggle motor run/stop, and the right button to toggle between potentiometer (voltage) speed control ("stock" solution) and a frequency control mode. Indicate the mode of operation in column 16 of the first line of the LCD display: "V" for voltage, "F" for frequency. For the frequency control mode, use the function generator at your lab station as a "musical" tone: connect the SYNC output of the function generator (CMOS compatible square wave) to PTT2 (timer channel 2), configured for input capture mode. Write a function that converts input capture timestamps to frequency, and convert the frequency to a proportional (8-bit) PWM duty cycle: the lower the frequency, the slower the motor should run, and the higher the frequency, the faster the motor will run. So, for example, if you wish to allow a frequency control range of 1000 Hz, you will want to divide the frequency detected by four to produce a PWM duty range of 250 (assuming a PWM period of 255).

While beyond the scope of this experiment (but well within the realm of an exciting potential Mini-Project), a microphone/preamp could be added along with some input conditioning circuitry to create a " hummer" – a device that converts musical tones to different motor speeds. The speed of a color wheel (or disco ball?) could be modulated by music.

As they tend to screech on most infomercials, "Amazing!"…

Thought Questions

Answer the following thought questions in the space provided below:

(a) Why is an optical isolator used to interface the 9S12C32 PWM output pin with the NPN switching transistor that controls the motor? Could an NPN switching transistor like a TIP 120 be interfaced to a 9S12C32 port pin *without* using an optical isolator?

(b) What are the advantages of using PWM to control the speed of a D.C. motor? (Compare PWM to any alternatives you think might be feasible.)

(c) Why not just connect the motor directly to the potentiometer to control its speed? What would happen? (DO NOT TRY THIS!)

(d) What would happen if a higher PWM sampling frequency were utilized? Is there a practical limit to the sampling frequency that can/should be utilized? (This one you CAN safely try!!)

(e) Given a 64-slot chopper and a 1.0 second integration period, what is the RESOLUTION of the tachometer (i.e., the difference in RPM estimate caused by pulse count difference of one over the integration period)?

