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 by the end of your scheduled lab period)	**10***	
3	Submission (completed immediately following demonstration)	**5***	
TQ	Thought Questions	**5**	
Bonus	Time-of-Day Clock	**5**	
	TOTAL	**25+**	

** code must function as specified to receive full credit for software and submission scores (score for non-functioning code will be 20% of max for both software and submission)*

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 (DK-3 parts kit)
- TIP 120 or TIP 122 NPN transistor (DK-3 parts kit)
- 2 x 16 LCD (DK-3 parts kit)
- GAL22V10 programmed as an 8-bit shift register (DK-3 parts kit)
- Breadboard and wire

Preparation:
- Read this document in its entirety
- Review material on the ATD, PWM, TIM/PA, SCI, and SPI peripherals

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 timer module (TIM), the pulse accumulator (PA) unit (part of the 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. In the background, message strings of your choice will continuously be output via the SCI to an emulated terminal screen, at the rate of one (80-character) string each second.

Your solution should utilize the two pushbuttons ("PAD7" and "PAD6") on the docking module: the left pushbutton ("PAD7") should toggle the motor "run" state (i.e., start/stop its rotation), while the right pushbutton should toggle the display mode (for bonus credit option, the first line of the LCD will display either the motor shaft speed in RPM or the time-of-day clock). The left LED (connected to port pin "PTT1") should be on when the motor is running (and off when it is stopped). The right LED (connected to port pin "PTT0") should be on when RPM is being displayed on the first line of the LCD, and off when the time-of-day clock is displayed on the first line of the LCD. At startup, both LEDs should be off.

The 3-digit gear-head drive shaft speed (in RPM) as well a percent-of-maximum bar graph will be displayed on an LCD. An external shift register will be used to interface the LCD to the microcontroller via the SPI module, as was done in Experiments 8 and 9.

2. Software Overview

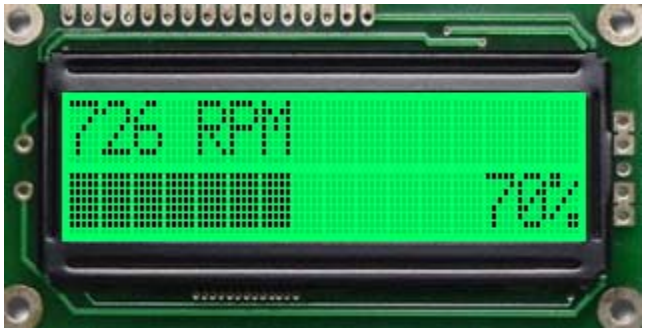
The objective of this experiment is to control the speed of a small D.C. motor using pulse width modulation. The PWM duty cycle will be generated based on a variable D.C. voltage (in the range of 0 to 5 volts) sampled on PAD0 every 0.1 second. A 10 KΩ potentiometer will be used to supply the variable input voltage. The motor speed will be estimated based on the number of pulses detected by the pulse accumulator over a one second integration period; a "chopper" attached to the motor shaft generates 64 pulses per revolution. 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 gear-head drive shaft speed can be calculated by dividing the motor shaft speed by 28. This 3-digit (BCD) value will be calculated and displayed on the first line of the LCD once each second.

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

The SCI will be used to print message strings to an emulated terminal in a buffered, interrupt-driven fashion. Function `bco(x)` will be used to place the character `x` in the next available location of the `tbuf` FIFO. Once a character has been placed in `tbuf`, the `bco` function will enable SCI transmit interrupts. The SCI transmit ISR will keep outputting characters from the `tbuf` FIFO each time it is called (i.e., every time the TDRE interrupt occurs) to the SCI as long as there are characters in `tbuf`. When the ISR empties the buffer, it will disable the transmit interrupt so that no superfluous interrupts are generated by the SCI. This methodology allows the processor to do other things in-between character transmissions (recall that, at 9600 baud, each character takes about 1 ms to transmit).

Similar to previous experiments, 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) interfaced to the SPI module through Port M (PTM).

On the first line of the LCD, the RPM should be displayed, 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.

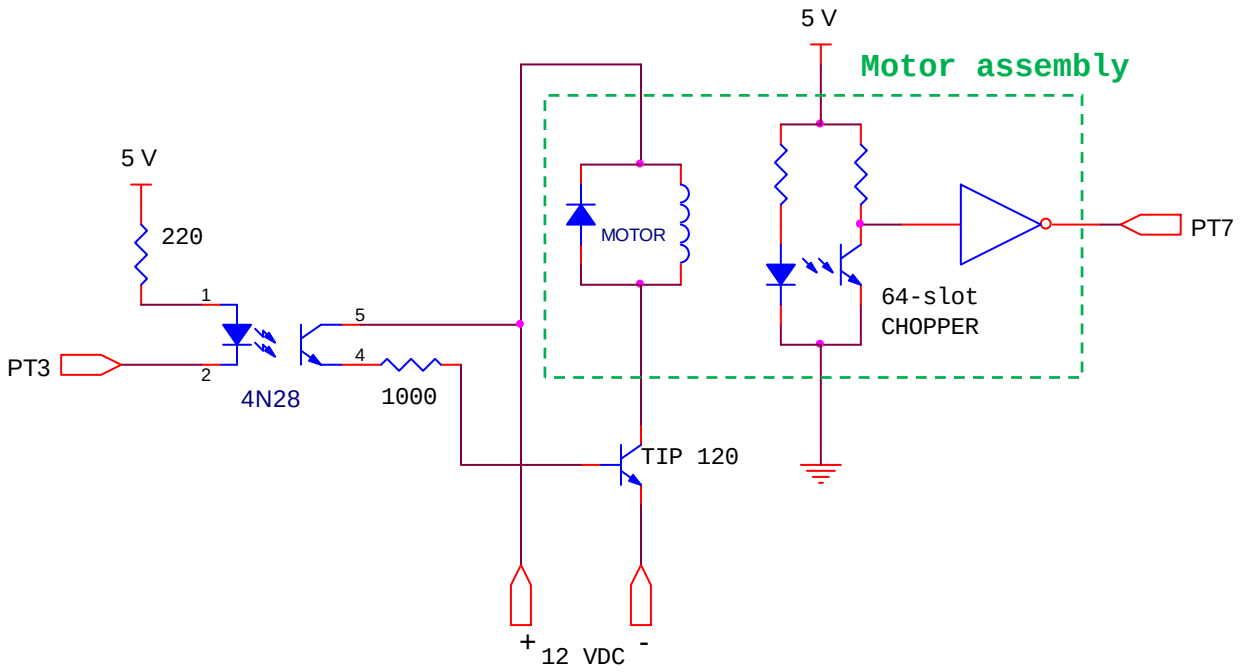
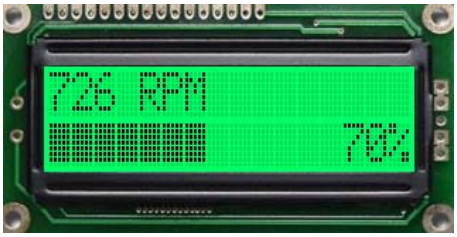


Figure 1. Wiring diagram for motor interface.

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 and 9 (***except*** for the **Port T** connections – see chart below). An external 8-bit shift register (GAL22V10) will once again 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 following table.



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.

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)**. Also, construct the optical isolation circuit, ***taking care to keep the motor "ground" separated from the logic circuit ground.*** Complete all the interface wiring on your breadboard as part of your pre-lab preparation.

Step 2. Software

Complete the "C" skeleton file provided in the Code Warrior **Lab10** project folder on the course website. 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

Zip your completed **Lab 10** Code Warrior project folder and submit it on-line (using the link at the bottom of the Lab Experiments page) *immediately after* demonstrating it to your lab T.A. ***Be sure identifying information*** (i.e., name, class number, and lab division) ***is included in the*** `main.asm` ***file you submit – credit will not be awarded if identifying information is omitted.***

Bonus Credit

Add the time-of-day ("tick tock") clock you created for Experiment 4 to your application, only now using the TIM module to provide the one-second timing reference needed. Use the right pushbutton to toggle the first line of the LCD between displaying the RPM and the current time. Format should be **`TIME 01:02:03 A/P`** (i.e., last character should be `A` or `P`). The dialog for setting the time should run upon reset, using the emulated terminal (same as Lab 4). Once the time has been set, the periodic message strings should start to be displayed on the terminal.

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)?

