ECE 362 Lab Verification / Evaluation Form

Experiment 8

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	Coding	**7**	
2	Interfacing	**7**	
3	Demonstration	**7**	
4	Thought Questions	**4**	
	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 8: Coach "Hope Against Hope" Reaction Timer

Instructional Objectives:
- To illustrate how the 9S12C32 TIM module can be used to implement a precision time base
- To provide additional experience with various 9S12C32 peripherals and ISRs

Parts Required:
- 2 x 16 LCD and 16-pin single-row header (available from the Instrument Room, EE 168)
- One 8-bit 74HC164 shift register (previous experiment)
- RED, YELLOW, and GREEN LEDs plus current limiting resistors (previous experiment)
- Breadboard and wire

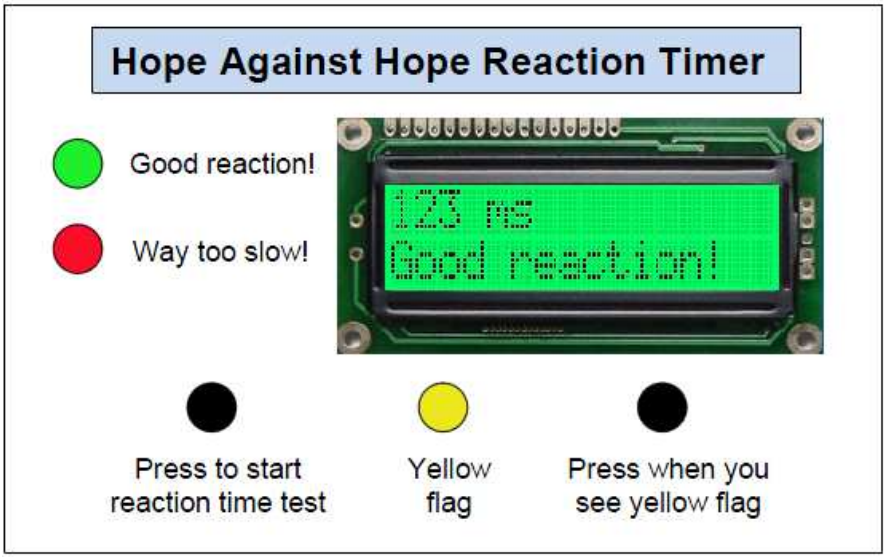
Preparation:
- Read this document in its entirety
- Review the material on the TIM subsystem
- Solder the 16-pin single-row header to your LCD
- Create your solution in "C" using Code Warrior

Introduction
Midway through the football season, eternally optimistic Coach "Hope Against Hope" (HAH) got an idea (i.e., an *ah-HAH* moment). Determined to be as quick as possible to challenge all the "bad calls" that he's sure will be made at the big ***Battle for the Bit Bucket*** game with **Boo-U**, he needs a reaction timer to test the swiftness of his protest skills. This device will feature an LCD to display the status of the reaction timer as well as the reaction result. This design features both portability and on-the-go reaction timing. Not to mention, you're sure to have an exciting experience seeing your reaction time light up the eyes of your Liberal Arts friends.

Coach HAH's reaction timer will use the docking module pushbuttons as follows: the left pushbutton starts the reaction test which will display the message "Ready, Set…" on the first line of the LCD and initiate a random delay. When this delay expires, "Go Team!" will be printed to the second line of the LCD, the yellow LED will be illuminated and the reaction timer will start. The right pushbutton stops the test, at which point the reaction time will be displayed on the first line of the LCD in the format "RT = NNN ms" and an appropriate message will be displayed on the second line. Also, the YELLOW LED should be turned off and if the resulting reaction time was less than 250 milliseconds, the GREEN LED should be turned on. If the reaction time was greater than 1 second, the RED LED should be turned on. The left LED on the docking module will be used to indicate the reaction time test is "stopped" (i.e., ready to start a new test), while the right LED will be used to indicate a reaction time test is "in progress".

In addition to the reaction time, appropriate messages (i.e., messages you wouldn't mind if your Mom saw) should be displayed on the second line of the LCD (e.g., "Ready to start!" upon reset, "Way to go HAH!!" if a really fast reaction time is recorded, etc.). It should be noted that LCD messages should be limited to 16 characters due to the size of the screen. If, for some reason, the reaction time exceeds 999 milliseconds, an appropriate suggestion should be displayed on the second line of the LCD (e.g., "Wake up!").

Lucky for Coach HAH, there are plenty of underpaid and overworked ECE 362 students armed with 9S12C32 microcontroller kits willing to work for the opportunity to buy full-price **Boo-U** game tickets. Time's running out, though, so gather up your parts kit and get to work!



Software Overview

Timer Channel 7 will be used to provide periodic interrupts at precise 1.000 millisecond intervals for the purpose of gauging the reaction time. The Timer Channel 7 interrupt service routine should count of the elapsed time using the "react" variable provided in the skeleton file. This variable should be kept in BCD format.

The RTI will be used to sample the pushbuttons on the docking board every 2.048 milliseconds as well as increment the "random" counter. Data for the LCD display will be shifted out to an external shift register (74HC164) and will be interfaced to the SPI module through Port M.

A key element will be providing a "random" delay between the instant the "start test" (left pushbutton) is pressed and the "Go Team!" message is displayed on the LCD. Here, the "random" counter maintained by the RTI service routine will be used to trigger the start of a reaction time test.

In order to help with LCD interfacing, the subroutine skeletons outlining a simple driver have been provided to you:

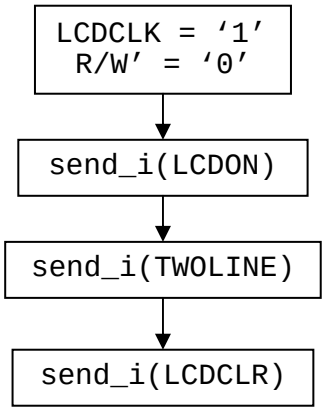
Subroutine	**Function**
shiftout	Transmits the contents of A to a shift register using SPI
lcdwait	Delays for 2 milliseconds
send_byte	Writes contents of A to the LCD
send_i	Sends instruction byte in A to LCD
linechg	Move LCD cursor to the cursor position passed in A
print_c	Print character passed in register A on LCD
pmsglcd	Print to the LCD all characters passed by call until an ASCII NULL character (decimal 0) is found.

You will be required to write all of these subroutines (or equivalent) to interface with the LCD. Additionally, many of these routines depend on others (for instance, `pmsglcd` requires `print_c` be written). You should build the driver from the ground up starting at `shiftout` and designing toward `pmsglcd`. The `pmsglcd` routine should take in a string passed by call and print every character to the LCD until it reads an ASCII NULL character ("0"). You should make multiple calls to `print_c` to print each individual character to the LCD screen.

Before printing characters to the LCD, it must first be initialized by writing instruction bytes. The relevant LCD instructions are given in the skeleton file and are replicated below:

Instruction	Byte	Function
LCDON	$0F	Turn on the LCD driver chip
LCDCLR	$01	Clear the LCD
TWOLINE	$38	Enable two line display mode
CURMOV	$FE	Cursor move command (requires second cursor position byte)

To initialize the LCD, you must send the LCD commands in the following order. NOTE: LCDCLK and R/W' are names of signals, not instructions. Also, keep in mind `send_i` is a subroutine that takes arguments passed by value in register A.



NOTE: *The LCD initializations call subroutines that you must write. It is recommended that you validate these subroutines before attempting initialization.*

The subroutine `chgline` moves the cursor on the LCD to the cursor position passed by value in register A. To accomplish this, you must first send the CURMOV instruction, followed a second instruction byte telling the LCD which position you want the cursor to move. These cursor position bytes are given in the table below.

	Cursor Position						
	0	**1**	**2**	**…**	**15**	**…**	**40**
Line 1	$0	$1	$2				$26
Line 2	$40	$41	$42		$5F		$66
Line 3	**$80**	**$81**	**$82**		**$9F**		$A6
Line 4	**$C0**	**$C1**	**$C2**		**$DF**		$E6

This is the cursor position byte map for our LCD driver (Hitachi HD44780). The area shaded in green corresponds to the valid cursor positions for our 2 x 16 character LCD. For example, the first character of the first line corresponds to cursor position $80 and first character of the second line corresponds to cursor position $C0. It should be noted that both of these cursor positions are given as variables in your skeleton file (`LINE1`/`LINE2`).

Keep in mind that the LCD driver chip is designed to accommodate up to a 4x40 character display (ours is only 2 x 16). This means that you will be able to write to cursor positions that are not within the viewable area of your LCD.

Hardware Overview

On the docking module, the left pushbutton will be used to initiate a reaction time test, while the right pushbutton will be used to stop the test in progress (and display the reaction time results). The left LED will be used to indicate the reaction timer is stopped (ready to start a new test), while the right LED will be used to indicate a reaction time test is in progress.

A GREEN LED on PTT[5] will be used to indicate a reaction time less than 250 milliseconds, a RED LED on PTT[6] will be used to indicate a reaction time greater than 999 milliseconds and a YELLOW LED on PTT[7] will indicate the start of the reaction time test.

An external 8-bit shift register (74HC164) will be used to interface LCD to the microcontroller via the SPI module (MOSI, port pin PM[4]; and SCK, port pin PM[5]). 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)	PTT[2]
5	R/W' (LCD read/write)	PTT[3]
6	LCD Clock	PTT[4]
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: DB[#] are the LCD data inputs and Q[#] are the data outputs of the shift register.

NOTE: ***The LCD does not come with pre-soldered headers. You will need to solder a 16-pin header to the LCD so that you will be able to plug it in to your breadboard.***

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. Coding
Complete the "C" skeleton file provided 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.

Step 2. Interfacing
Interface the LCD and LEDs to your microcontroller kit as described in the Hardware Overview section.

Step 3. Demonstration
Demonstrate your working hardware and software to your lab TA.

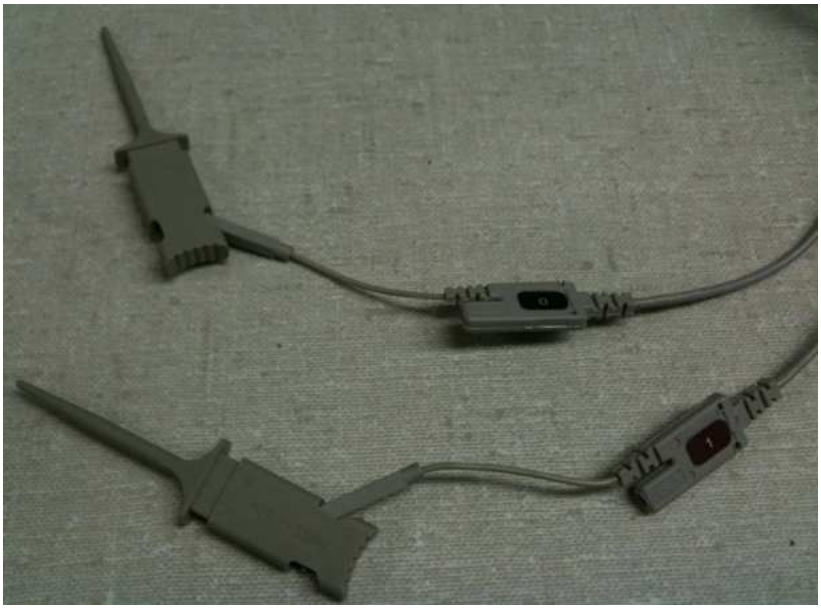
Step 4. Thought Questions
Answer the following thought questions in the space provided below:

(a) Comment on the differences between the microcontroller-to-shift-register interface implemented in this lab (using SPI) vs. the interface implemented in Lab 6. Include both hardware and software considerations.

(b) Based on the AC timing parameters listed in the 74HC164 data sheet, determine the maximum 9S12C32 SPI data rate that could be used for the LCD interface. Are there any potential advantages or disadvantages (or potential problems) associated with operating the interface at the maximum data rate possible?

Debugging Tips

When debugging the LCD circuit, you should use the oscilloscope logic analyzer if you're trying to watch more than two signals at a time (e.g. looking at all eight shift register outputs at one time. The only difference in using this rather than using the `A1/A2` analog oscilloscope probes is that instead, you'll be using the `D0-D7` probes. If these probes aren't immediately visible, please check the bag on top of the oscilloscope.



D0 and D1 LA probes

First, you should attach a probe to each pin you want to measure. Next, you should press the `D0-D15` button in the lower right corner of the oscilloscope controls. This display allows you to enable `D0-D7` and `D8-D15`. Enable `D0-D7` by pressing the "`On`" control for these probes (the third button from the left below the display). You can also choose to disable the `A1/A2` probes if you want more display room for `D0-D7`.

To trigger on any of these ports, you can press the `Edge` button and select the `D bit` as your trigger source (third button from the left below the display). You can select among the D bits after you have pressed this button by turning the knob marked "`Select`" under the `D0-D15` button. You may also choose whether to trigger on a rising or falling edge in the same menu.

The last thing to check is under the `Mode/Coupling` button menu. You'll want to select "`Normal`" trigger mode if you want a one-shot trigger. If you have a periodic waveform, select "`Auto`" as your trigger mode. Leave the coupling as "`DC`" for non-periodic waveforms. The rejection functions are situation-dependent and can be changed at will.

You may now run your code and the LA will trigger based on the settings you've entered.

