ECE 362 Lab Verification / Evaluation Form

Experiment 6

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	**6**	
2	Coding	**6**	
3	Demonstration	**10**	
4	Thought Questions	**3**	
	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 6: NBA Shot Clock

Instructional Objectives:
- To illustrate how the 9S12C32 RTI can be used as a clock time base
- To illustrate use of the 9S12C32 port pins as both inputs and outputs

Parts Required:
- Two common anode 7-segment displays (available from the Instrument Room, EE 168)
- Two 8-bit 74HC164 shift registers (available from the Instrument Room, EE 168)
- Breadboard and wire (from ECE 270 parts kit)

Preparation:
- Read this document in its entirety
- Fill the schematics for the pre-lab and submit it at the beginning of your lab session
- Review the reference material on the RTI subsystem
- Create your solution in "C" using Code Warrior

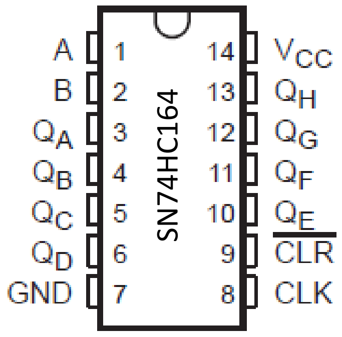
Introduction

Coach *Phil Jokeson* of the *LA Lackers* has a problem – on a recent practice slam dunk attempt, 7' 6" star draftee *Ray Deo Shaq* smashed the entire backboard along with the 24-second shot clock, just as the NBA basketball season was posed to wipe away sports fans' collective baseball fatigue! Lucky for the *Lackers*, there are currently plenty of ECE 362 students armed with 9S12C32 microcontroller kits anxious to see something other than America's lamest pastime (watching the Cubs lose *again*) on TV.

Hardware Overview

Your job is to implement a 24-second shot clock. It should utilize the two pushbuttons ("PAD7" and "PAD6") on the docking module: the left pushbutton ("PAD7") should reset the shot clock to 24 seconds (change of possession), while the right pushbutton ("PAD6") should start/stop the shot clock (out of bounds/time out). When the time left is 9.9 seconds or less, the middle decimal point should be illuminated and the display should decrement every tenth of a second. The left LED (connected to port pin "PT1") should be on when the shot clock is in the "run" state and off when it is in the "stop" state; the right LED (connected to port pin "PT0") should be turned on when the count reaches "0.0". Note that the clock should stop counting down once it reaches "0.0", and that the right LED should remain on until the shot clock is reset; the left-most decimal point of the display should also be illuminated when the clock reaches "0.0".

The two-digit BCD value should be displayed on a pair of common-anode 7-segment LEDs (similar to the one provided in the ECE 270 DK parts kit). To interface the microcontroller kit to the 7-segment LEDs, use the two 74HC164 8-bit shift registers. Use Port T bit 3 ("PT3") to supply the SHIFT DATA signal, and Port T bit 4 ("PT4") to supply the SHIFT CLOCK signal. Note that the decimal point (DP) of each display should also be connected.



Software Overview
Once the boot-up and initialization sequence is completed, the "main" program should simply "loop" − looking at various "flags" (SRAM locations used as Boolean variables) to see if some action should be taken. Four such flags will be needed: (a) the "tenth second" flag, (b) the left pushbutton ("reset shot clock") flag, (c) the right pushbutton ("start/stop") flag, and (d) the "run/stop" flag.

If the "tenth second" flag is set, the following actions should take place:

- if the "run/stop" flag is set (i.e., shot clock is running), then
  - ➢ decrement the shot clock by one tenth of a second (note that the shot clock value needs to be stored internally as a three-digit BCD number)
  - ➢ update the seven-segment display (using upper two digits of the BCD value if the time left is greater than or equal to 10 seconds; or the lower two digits of the BCD value, with the middle DP illuminated, if the time left is 9.9 seconds or less)
  - ➢ clear the "tenth second" flag

If the left pushbutton ("reset shot clock") flag is set, the following actions should take place:

- clear the left pushbutton flag
- reset the display to "24"

If the right pushbutton ("start/stop") flag is set, the following actions should take place:

- clear the right pushbutton flag
- toggle the "run/stop" flag
- toggle the "run/stop" LED

The "main" program should also, in its polling loop, check to see if the shot clock has reached "0.0" – if it has, the following actions should take place:

- clear the "run/stop" flag
- turn on the "time expired" LED and the left-most DP on the 7-segment display
- turn off the "run/stop" LED

Producing a periodic interrupt every 1.408 ms, the RTI service routine will serve two purposes:

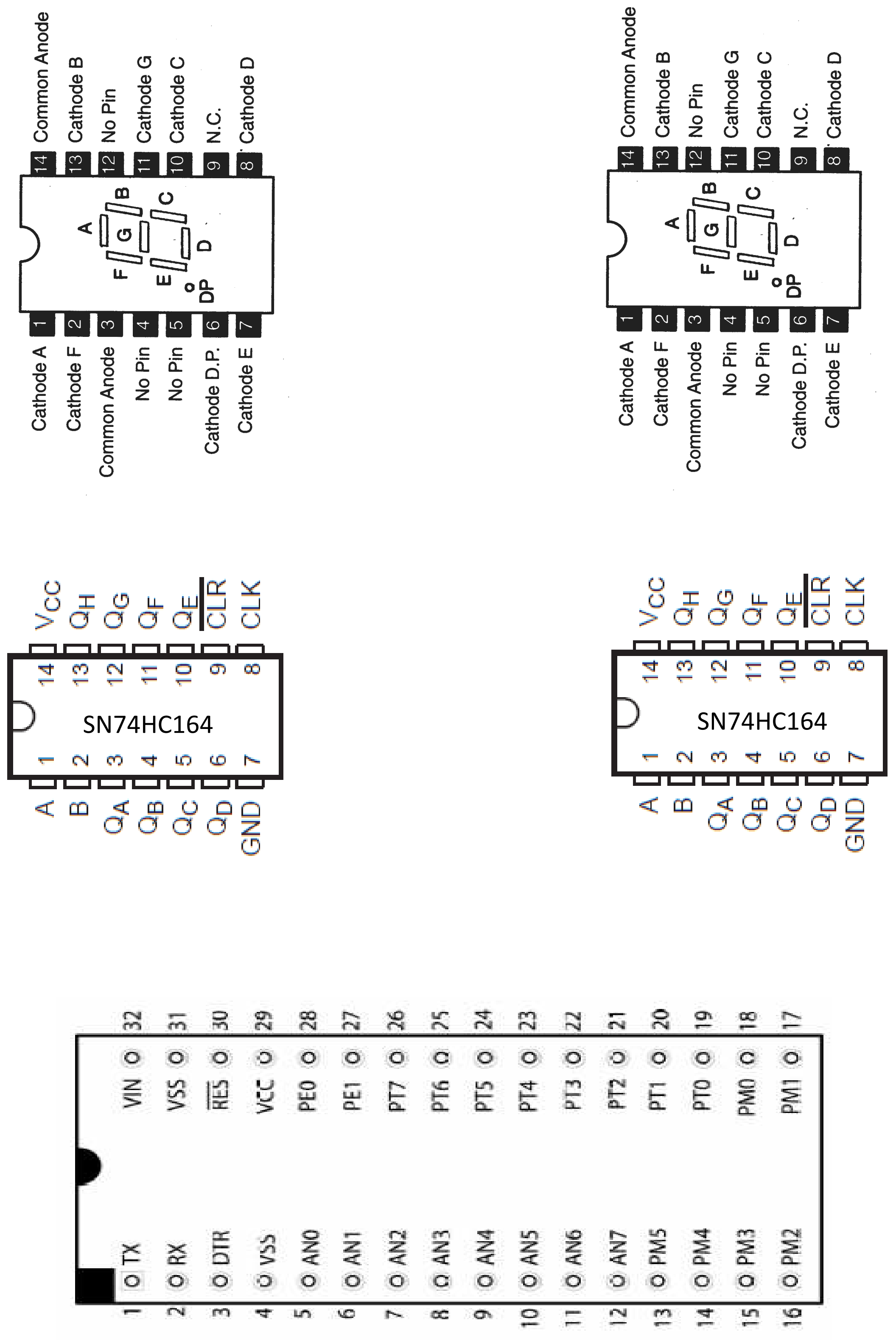
- On each interrupt, sample the state of the pushbuttons; if the *previous state* was high and the *current state* is low (note that the pushbuttons are contact closures to ground, and that their default state is "high"), set the corresponding "button flag" to trigger actions in the "main" program-driven polling loop (note that this implies the need for an SRAM location that saves the "previous" state of the pushbuttons).
- When one tenth of a second's worth of RTI interrupts has accumulated, set the "tenth second flag" to trigger actions in the "main" program-driven polling loop.

Software tips:

- Store the time in packed BCD format as a word (0dd.d), where "d" is a BCD digit (note that the "decimal point" is *virtual*).
- Decrement the 16-bit BCD time value by adding $9999 to it, followed by a DAA.
- Switch the display mode from "seconds" to "tenths" when the upper byte of the time reaches zero.
- Debug your countdown logic using the emulated terminal on the PC (TeraTerm).

Step 1. Interfacing

Complete the wiring diagram below. The microcontroller should be connected to the shift registers, and the shift registers should be connected to the 7-segment displays.



Step 2. 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.

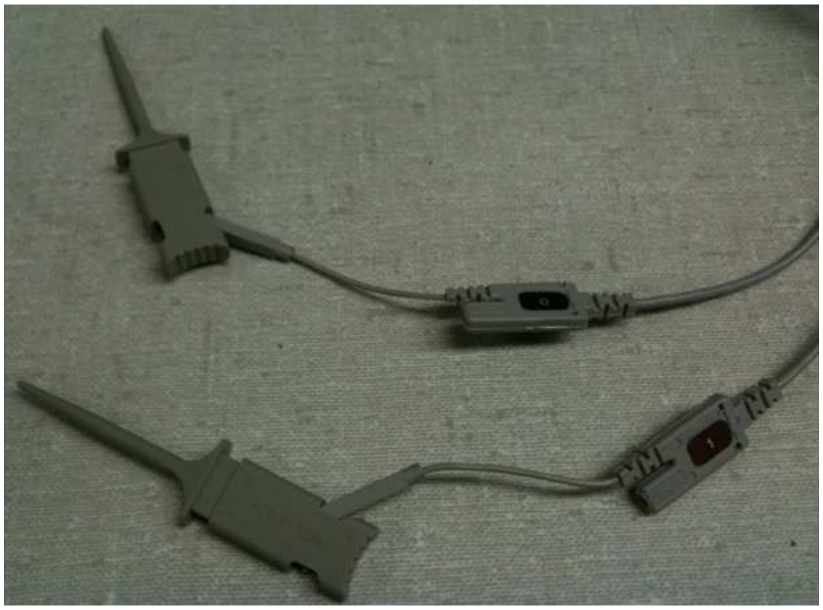
Step 3. Demonstration
Demonstrate your working hardware and software to your lab TA, and be prepared to explain your answers to the Thought Questions (below).

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

(a) Why do we need to know the previous state of the pushbuttons, and set "flags" based on their transition from high to low?

(b) How accurate is your shot clock, given that its time base is a 1.408 ms periodic interrupt rate (e.g., how "far off" is it after running 24 seconds)? Does your design run "slow" or "fast"? Is this accuracy acceptable (i.e., is it "close enough" so that not even *Ray Deo Shaq* or *Coach Jokeson* will notice any error)? Show any calculations necessary to prove your answer.

Debugging Tips
When debugging your shift register/display 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.