Jonathan W. Valvano First:________________ Last:____________________
October 19, 2007, 1:00pm-1:50pm. This is a closed book exam. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting***.

(5) Question 1. Why is it *highly intrusive* to add this code to an interrupt service routine, then observe the output on HyperTerminal?

```
     SCI_OutUDec(time); SCI_OutUDec(data); SCI_OutChar(CR);
```

(10) Question 2. You have the following two interrupt service routines. Add minimally-intrusive debugging instruments to determine which interrupt service routine is executed first. In particular, set **bTOF** if TOF is first and set **bRTI**, if RTI is first.

```
short bTOF=0; // true if first TOF occurs before first RTI
short bRTI=0; // true if first RTI occurs before first TOF
interrupt 16 void TOFhandler(void){




  TFLG2 = 0x80;       // acknowledge TOF
  Stuff1();
}
interrupt 7 void RTIhandler(void){




  CRGFLG = 0x80;     // acknowledge RTIF
  Stuff2();
}
```

(10) Question 3. Consider two debugging instruments. The first one is called from the main program

```
  PTT ^= 0x01; // toggle PT0
```

It is compiled into

```
          ldaa $0240
          eora #$01
          staa $0240
```

The second one is called from an interrupt service routine

```
  PTT ^= 0x02; // toggle PT1
```

It is compiled into

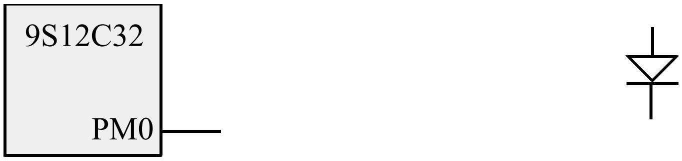
```
          ldaa $0240
          eora #$02
          staa $0240
```

Do these read-modify-write sequences constitute a critical section? Answer yes or no. If yes, specify how you would change the system to correct the error. If no, justify why there can be no error.

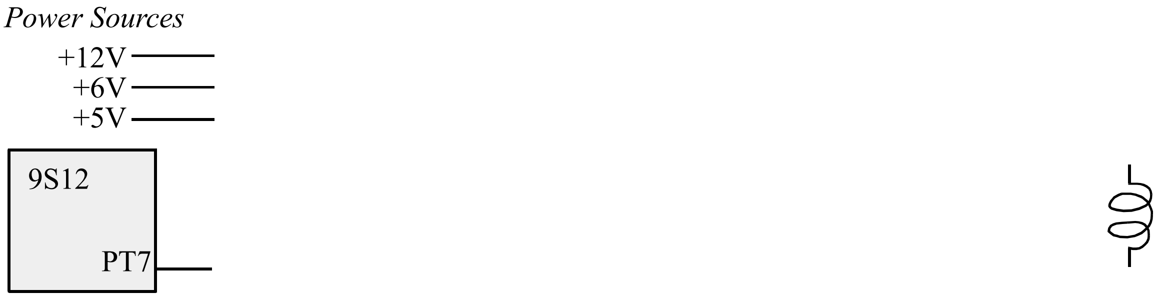
(10) Question 4. Consider an input interface using the SCI device. When the main program wants an input, it calls **`SCI_InChar`**. **`SCI_InChar`** uses busy-wait synchronization (waits for **`RDRF`**=1, then reads data from **`SCIDRL`**). In this system, define *interface latency*.

(5) Question 5. What are the minimum and maximum values of an 8-bit signed decimal fixed-point system which has a resolution of 0.1V?

(10) Question 6. An LED voltage requires 2 V at 25 mA to activate. Interface this LED to the 9S12C32 port pin PM0, such that when the software outputs a one, the LED comes on, and when the software outputs a zero, the LED goes off. If more than one possibility exists, choose the cheapest method. Label all interface components and resistor values. You can specify resistor values using an equation, rather than calculating the exact number.



(10) Question 7. Interface a 12V solenoid to the 9S12. To activate, the coil needs 11.3 to 12.7 V at 200 mA. Include protection against back EMF. Label all interface components and resistor values. You can specify resistor values using an equation, rather than calculating the exact number.



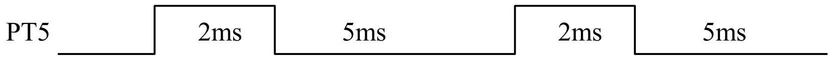
(20) Question 8. You will draw a Moore finite state graph that controls a stepper-motor robot. You will be just drawing the state graph (no hardware or software). There is 1 binary input

0 means the robot should turn left

1 means the robot should go forward.

There are 8 binary outputs to two stepper motors attached to separate rear wheels. Each 4-bit stepper motor must always be one of the accepted values 5, 6, 10, or 9. Each output in the sequence $5,$6,$A,$9,$5,$6,$A,$9… will rotate a wheel forward. If both wheels are rotated forward, the robot moves forward in a straight line. If just the right wheel moves forward, the robot turns left, pivoting about the left wheel. For each stepper, the only changes allowed are from $5 to $6, $6 to $A, $A to $9, or $9 to $5. (no going backwards and no skipping from $5 to $A). You may assume the state controller will output, wait 10ms, input, then go to the next state (depending on the input). Specify the output for each state as an 8-bit hex number, e.g., $5A means $5 to the left motor and $A to the right motor. It is acceptable to delay a short amount of time (0 to 40 ms) after the input changes before the new output pattern is started. Show the state graph where each state has one 8-bit output and 2 next-state arrows. You do not need to add state names.

(20) Question 9. Use output compare 5 interrupts to generate the following periodic output on PT5. Assume the PLL is off, so the E clock is 4 MHz (250ns).



Part a) Show the ritual that is executed once.

Part b) Show the output compare interrupt service routine that outputs to PT5.

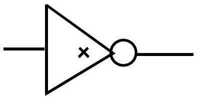
PTT is 8-bit bi-directional I/O port
DDRT is the associated direction register for Port T (0 means input, 1 means output)
PTM is 6-bit bi-directional I/O port
DDRM is the associated direction register for Port M (0 means input, 1 means output)

TSCR1 is the first 8-bit timer control register
bit 7 **TEN**, 1 allows the timer to function normally, 0 means disable timer including **TCNT**
TSCR2 is the second 8-bit timer control register
bits 2,1,0 are **PR2**, **PR1**, **PR0**, which select the rate, let **n** be the 3-bit number formed by **PR2**, **PR1**, **PR0**
without PLL **TCNT** is 4MHz/$2^{\mathbf{n}}$, with PLL **TCNT** is 24MHz/$2^{\mathbf{n}}$, **n** ranges from 0 to 7
TCNT is 16-bit up counter
TIOS is the 8-bit output compare select register, one bit for each channel (1 = output compare, 0 = input capture)
TIE is the 8-bit output compare arm register, one bit for each channel (1 = armed, 0 = disarmed)
TC0 TC1 TC2… **TC7** are the eight 16-bit output compare registers, one register for each channel
TFLG1 is the 8-bit flag register, one bit for each channel,
(with output compare, flags are set when **TCNT** equals **TC0 TC1 TC2**… **TC7**)
flags become zero when software writes a 1 to it (e.g., **TFLG1=0x08;** clears channel 3 flag)
SCIDRL 8-bit data serial data register
SCIBD is 16-bit SCI baud rate register, let **n** be the 16-bit number Baud rate is 12MHz/**n**
SCICR1 is 8-bit SCI control register
bit 4 M, Mode, 0 = One start, eight data, one stop bit, 1 = One start, eight data, ninth data, one stop bit
SCICR2 is 8-bit SCI control register
bit 7 TIE, Transmit Interrupt Enable, 0 = TDRE interrupts disabled, 1 = interrupt whenever TDRE set
bit 5 RIE, Receiver Interrupt Enable, 0 = RDRF interrupts disabled, 1 = interrupt whenever RDRF set
bit 3 TE, Transmitter Enable, 0 = Transmitter disabled, 1 = SCI transmit logic is enabled
bit 2 RE, Receiver Enable, 0 = Receiver disabled, 1 = Enables the SCI receive circuitry.
SCISR1 is 8-bit SCI status register
bit 7 TDRE, Transmit Data Register Empty Flag
Set if transmit data can be written to SCDR
Cleared by **SCISR1** read with TDRE set followed by **SCIDRL** write.
bit 5 RDRF, Receive Data Register Full
set if a received character is ready to be read from **SCIDRL**
Clear the RDRF flag by reading **SCISR1** with RDRF set and then reading **SCIDRL** .

ATDDIEN ADC digital enable register, 1 to make corresponding pin digital, 0 to make corresponding pin analog
PTAD is 8-bit bi-directional I/O port
DDRAD is the associated direction register for digital pins of Port AD (0 means input, 1 means output)

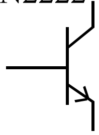
0xFFD6	**interrupt 20 SCI**
0xFFDE	**interrupt 16 timer overflow**
0xFFE0	**interrupt 15 timer channel 7**
0xFFE2	**interrupt 14 timer channel 6**
0xFFE4	**interrupt 13 timer channel 5**
0xFFE6	**interrupt 12 timer channel 4**
0xFFE8	**interrupt 11 timer channel 3**
0xFFEA	**interrupt 10 timer channel 2**
0xFFEC	**interrupt 9 timer channel 1**
0xFFEE	**interrupt 8 timer channel 0**
0xFFF0	**interrupt 7 real time interrupt**

7406



V_{OL} = 0.5V
I_{OL} = 40 mA
V_{IL} = 0.7V
I_{IL} = 1.6mA

2N2222



V_{ce} = 0.3V
V_{be} = 0.6V
h_{fe} = 100
I_{ce} =500mA max

9S12C32 parameters

I_{OL} = 10mA, I_{OH} = 10mA, I_{IL} = 1µA, I_{IH} = 1µA,
V_{OL} = 0.8V, V_{OH} = 4.2V, V_{IL} = 1.75V, V_{IH} = 3.25 V