Jonathan W. Valvano First:________________ Last:____________________

October 10, 2008, 2:00pm-2:50pm. 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. Who was your EE319K instructor?

A) Valvano B) Bill Bard C) Gary Daniels G) G. Jack Lipovski
D) Mark Welker E) Nur Touba F) Nachiket Kharalkar H) other

Put answer in this box

(10) Question 2. Consider two debugging instruments. They are both used as monitors to visualize when the interrupts have been serviced. The first one is called from output compare 0 ISR.

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

It is compiled into

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

The second one is called from an output compare 1 ISR.

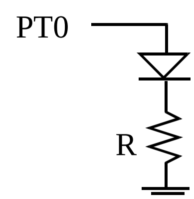
```
  PTT ^= 0x02; // toggle PT1
```

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 3. You are given the 9S12 voltage and current parameters for PT0: V_{OH}, V_{OL}, V_{IH}, V_{IL}, I_{OH}, I_{OL}, I_{IH}, and I_{IL}. The desired LED voltage is V_D and the desired current is I_D. Assume the LED current is small enough, so the LED can be interfaced directly to the 9S12 PT0 pin as shown. Give the equation to calculate the resistance R in terms of V_D, I_D, V_{OH}, V_{OL}, V_{IH}, V_{IL}, I_{OH}, I_{OL}, I_{IH}, and I_{IL}.

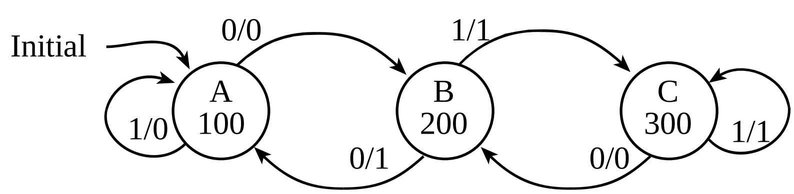


(15) Question 4. There are four binary fixed point numbers called **x1**, **x2**, **x3**, and **x4**. Each fixed-point number has a corresponding integer part called **I1**, **I2**, **I3**, and **I4**. The format is unsigned binary fixed point with a resolution, $\Delta = 2^{-4}$ (1/16). Assume **I1 I2 I3** and **I4** are defined as unsigned short variables in our system. The objective is **write C code** to implement the following equation using fixed-point calculations, no floating point allowed. To the best of your ability, minimize the errors due to overflow and dropout.

x4 = x1*x2 + x3

Basically, you need to calculate **I4** as a function of **I1**, **I2**, and **I3**. You are allowed to define additional variables. Hand execute your code with **x1**=1.5, **x2**=2, **x3**=0.25 to verify it calculates **x4**=3.25.

(5) Question 5. Consider the following Mealy FSM, where the initial state is A. The labels on the arrows mean input/output. The numbers in the circles (100,200,300) are time delays for each state. The sequence is 1) wait; 2) input; 3) do output depending on input and state; then 4) set next state depending on input and state. If the input were to be a constant 0, after a while, what happens?



A) Eventually the system ends up in state C with the output high.
B) The system oscillates between state A and state B with the output low.
C) Eventually the system ends up in state A with the output low.
D) The system oscillates between state A and state B with the output toggling high and low, with the output being high for a longer time than the output is low.
E) The system oscillates between state B and state C with the output toggling high and low, with the output being high for a longer time than the output is low.
F) The system oscillates between state A and state B with the output toggling high and low, with the output being low for a longer time than the output is high.
G) The system oscillates between state B and state C with the output toggling high and low, with the output being low for a longer time than the output is high.
H) None of the above.

Put answer in this box

(8) Question 6. Consider the following C program, which is implemented on an embedded system. Where are each of the four variables stored? For each variable specify A, B or C:

A) Global means permanently allocated at a fixed location in volatile memory.
B) Stack means temporarily allocated in RAM or a register, used, then deallocated.
C) EEPROM means permanently allocated at a fixed location in nonvolatile memory.

Please note that the variable names in this example do not follow the standard naming conventions.

```
const char v1=100;
static char v2=10;
char add3(const char v3){
static char v4;
  v4 = v1+v3;
  return(v4);
}
void main(void){
  v2 = add3(v2);
}
```

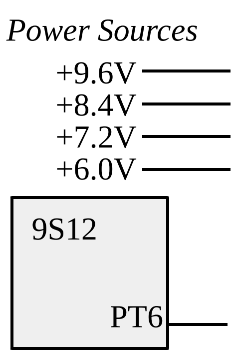
Part a) Where is **v1** allocated?

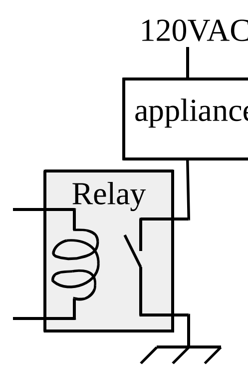
Part b) Where is **v2** allocated?

Part c) Where is **v3** allocated?

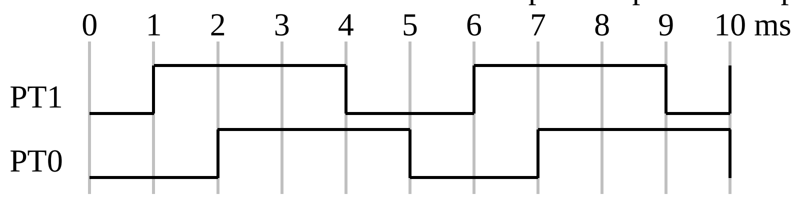
Part d) Where is **v4** allocated?

(12) Question 7. The NiMH battery cell voltage is 1.2V. Using multiple cells, we can create a power source at integer multiples of 1.2V. Interface a 6V electromagnetic relay to the 9S12. To activate, the relay needs anywhere from 5.4 to 6.6 V at 100 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. You must select which NiMH battery to use.





(35) Question 8. The following output will spin a 2-phase synchronous motor. Write software that creates the following outputs on PT1 PT0 (the pattern 0,2,3,1 repeats over and over every 5 ms). Notice that the 3 state (PT1=1, PT0=1) is 2 ms long, while the other states are 1 ms long. You write all the C code required to spin this motor. You must use a FSM and output compare interrupt 7.



Part a) Draw the FSM graph that has 2 outputs and no inputs.

Part b) Define the FSM structure. Give the `struct` definition.

```
const struct State{



};
typedef const struct State StateType;
typedef StateType *StatePtr;
```

Part c) Give the C code to place the linked list in ROM.

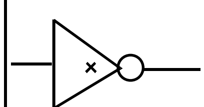
Part d) Show the main program that sets the direction register for PTT, sets up output compare 7, initializes the FSM, and then executes a do-nothing loop. You should not activate the PLL. (9S12C32 E clock is 4 MHz, 9S12DP512 E clock is 8 MHz). Outputs to PTT will occur in the background, not here in the foreground.

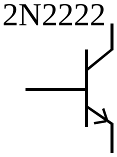
Part e) Show the output compare interrupt 7 service routine that outputs to PT1, and PT0. You need not be friendly. Basically, you should run the FSM here in the background. No backward jumps or conditional branching is allowed.

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 (8-bits on the 9S12DP512)
DDRM is the associated direction register for Port M (0 means input, 1 means output)
PTP is 8-bit bi-directional I/O port
DDRP is the associated direction register for Port P (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**
    9S12C32 without PLL **TCNT** is $4MHz/2^{\mathbf{n}}$, with PLL **TCNT** is $24MHz/2^{\mathbf{n}}$, **n** ranges from 0 to 7
    9S12DP512 without PLL **TCNT** is $8MHz/2^{\mathbf{n}}$, 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** .

```
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

9S12C32/9S12DP512 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