Jonathan W. Valvano First:_________________ Last:_____________________

February 22, 2008, 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. Who was your EE319K instructor? Circle one

a) Valvano	b) Bill Bard	c) Gary Daniels	g) G. Jack Lipovski
d) Mark Welker	e) Nur Touba	f) Nachiket Kharalkar	h) other

(5) Question 2. Assume bit 0 of PTT is an output, which is connected to an oscilloscope. The 9S12 is running at 4 MHz (E clock period is 250 ns). We added the following debugging monitor to an interrupt service routine:

```
  PTT ^= 0x01;
```

The compiler generated the following assembly code

```
  0014 f60000       [3]    LDAB  _PTT
  0017 c801         [1]    EORB  #1
  0019 7b0000       [3]    STAB  _PTT
```

Part a) Is this debugging monitor *nonintrusive*, *minimally intrusive*, or *highly intrusive*?

Part b) Justify your answer.

(5) Question 3. You are developing a pressure monitor. The pressure can range from 0 to 250 psi (lbs/in^2). Your system has a pressure resolution of 0.1 psi. Which number system would you use to store the pressure data without error? If more than one answer is possible, choose the most space efficient.

A) 8-bit unsigned integer
B) 8-bit unsigned decimal fixed-point system with a resolution of 0.1
C) 8-bit unsigned decimal fixed-point system with a resolution of 0.01
D) 16-bit unsigned binary fixed-point system with a resolution of 1/8
E) 16-bit unsigned binary fixed-point system with a resolution of 1/16
F) 32-bit unsigned integer
G) 32-bit floating point

Put answer in this box

(15) Question 4. The objective is to implement the following quadratic equation using fixed-point calculations. **n** is a 10-bit unsigned number (0 to 1023), and **m** is the output of the equation, which is a 16-bit unsigned number. Assume **n** and **m** are defined as unsigned short variables in our system. **Write C code to implement** (no floating point allowed)

$$\mathbf{m = 0.012*n^2 + 3.4*n + 7.7}$$

This table shows some typical results

n	m
0	8
250	1608
500	4708
750	9308
1000	15408

```
unsigned short n,m;
```

You are allowed to define additional variables. Hand execute your code with **n**=1000 to verify it calculates **m**=15408 using integer math.

(10) Question 5. Consider an output interface that uses interrupt synchronization. When the output device is idle, an interrupt is requested. During execution of the interrupt service routine, another data value is given to the output device. Interface latency, defined to be the time between output device idle and the software writing data, must be less than 1ms. Assume there are two other interrupts in this system used for other unrelated tasks. How do we design the system to meet this real-time requirement? Choose the best answer.

a) Activate the PLL so the software runs at 24 MHz.
b) Use an output compare interrupt with a periodic rate of 1ms.
c) Set the HPRIO register so this output device is highest priority.
d) Execute `sei` in the ISR of the output device so other interrupts can't suspend this task.
e) Execute `cli` in the ISR of the output device so other interrupts can suspend this task.
f) Run with the I bit equal to 0 for no more than 0.5ms at a time.
g) Run with the I bit equal to 1 for no more than 0.5ms at a time.
h) Switch over to periodic polling synchronization
i) Switch over to busy-wait synchronization

Put answer in this box

(5) Question 6. Assume PTT is an input and PTM is an output. The following C function is used to test the hardware

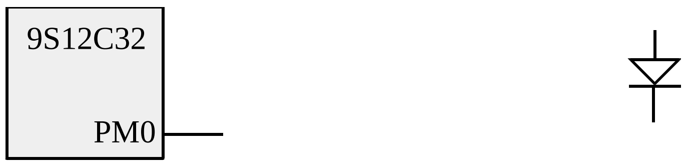
```
void test(void){ unsigned char i;
  for(i=0; i<4; i++){
    PTM = PTT+i;
  }
}
```

The compiler generated this output. (I added the `LL` to the Metrowerks code for clarity).

```
  0000 f60240       [3]test LDAB  PTT
  0003 070d         [4]     BSR   LL
  0005 52           [1]     INCB
  0006 070a         [4]     BSR   LL
  0008 cb02         [1]     ADDB  #2
  000a 0706         [4]     BSR   LL
  000c cb03         [1]     ADDB  #3
  000e 7b0250       [3]     STAB  PTM
  0011 3d           [5]     RTS
  0012 7b0250       [3] LL  STAB  PTM
  0015 f60240       [3]     LDAB  PTT
  0018 3d           [5]     RTS
```

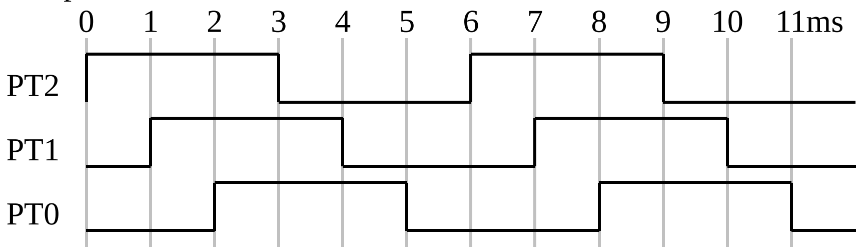
Where is the local variable **i** stored in this example? Be very explicit.

(15) Question 7. An infrared LED requires 1.7 V at 100 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.



(5) Question 8. Assume a 9S12 port pin configured as an input. What voltages would be recognized as a logic low? Assuming we are avoiding negative voltages, 0.0V is the smallest possible voltage that would be considered a logic low. Give the maximum possible value.

(35) Question 9. The following output will spin a 3-phase synchronous motor at 300 rpm. Write software that creates the following outputs on PT2 PT1 PT0 (the pattern 4,6,7,3,1,0 repeats over and over). You write all the C code required to spin this motor at 300 rpm. You must use a FSM and output compare interrupt 1.



Part a) Draw the FSM graph that has 3 outputs and no inputs.

Part b) Edit this FSM from the starter code **Moore_9S12** to solve this problem

```
const struct State{
  unsigned char Out;
  unsigned short Time;
  const struct State *Next[4];
};
typedef const struct State StateType;
typedef StateType *StatePtr;
#define SA &fsm[0]
#define SB &fsm[1]
#define SC &fsm[2]
#define SD &fsm[3]
#define SE &fsm[4]
#define SF &fsm[5]
StateType fsm[6]={
     {0x01,250,{SB,SC,SD,SE}},
     {0x02,250,{SA,SC,SD,SE}},
     {0x03,100,{SA,SC,SD,SE}},
     {0x00,100,{SA,SC,SD,SE}},
     {0x00,1000,{SA,SC,SD,SF}},
     {0x03,1000,{SA,SC,SD,SE}}
};
```

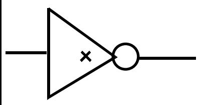
Part c) Show the main program that sets the direction register for PTT, sets up output compare 1, initializes the FSM, and then executes a do-nothing loop. You should not activate the PLL. (E clock is 4 MHz). Outputs to PTT will occur in the background, not here in the foreground.

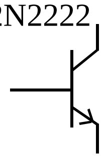
Part d) Show the output compare interrupt 1 service routine that outputs to PT2, PT1, and PT0. You need not be friendly. Basically, you should run the FSM here in the background.

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