Common Registers

TIOS

	Bit 7	6	5	4	3	2	1	Bit 0
R W	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
RESET:	0	0	0	0	0	0	0	0

IOS[7:0] — Input Capture or Output Compare Channel Configuration
1 = The corresponding channel acts as an output compare.
0 = The corresponding channel acts as an input capture.

TFLG1

	Bit 7	6	5	4	3	2	1	Bit 0
R W	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
RESET:	0	0	0	0	0	0	0	0

These flags are set when an input capture or output compare event occurs. Clear a channel flag by writing one to it.

TMSK1 (A4)

TIE (C32)

	Bit 7	6	5	4	3	2	1	Bit 0
	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
RESET:	0	0	0	0	0	0	0	0

C7I–C0I — Input Capture/Output Compare "x" Interrupt Enable.

TSCR (A4)

TSCR1(C32)

	Bit 7	6	5	4	3	2	1	Bit 0
	TEN	TSWAI	TSBCK	TFFCA	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

TEN — Timer Enable
1 = Allows the timer to function normally.
0 = Disables the main timer, including the counter.

N= 0 to 7 as determined by PR2, PR1, PR0

MC68HC812A4 TCNT frequency = $8/2^N$ MHz

TMSK2

	Bit 7	6	5	4	3	2	1	Bit 0
	TOI	0	TPU	TDRB	TCRE	PR2	PR1	PR0
RESET:	0	0	1	1	0	0	0	0

PR2, PR1, PR0 — Timer Prescaler Select
These three bits select the frequency of the timer prescaler clock derived from the Bus Clock

9S12C32 TCNT frequency = $24/2^N$ MHz

TSCR2

	Bit 7	6	5	4	3	2	1	Bit 0
R W	TOI	0	0	0	TCRE	PR2	PR1	PR0
RESET:	0	0	0	0	0	0	0	0

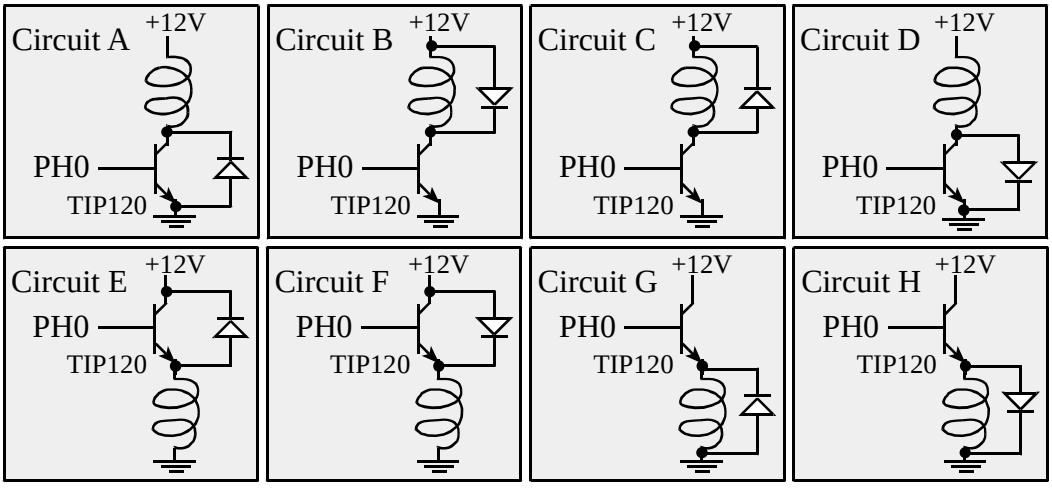
PR2, PR1, PR0 — Timer Prescaler Select
These three bits select the frequency of the timer prescaler clock derived from the Bus Clock

(40) Question 1. All code must be friendly.

PLACE THE CORRECT CHOICE FOR EACH QUESTION SHOWN ON THE ANSWER PAGE.

```
A)  asm(" cli");
B)  asm(" rti");
C)  asm(" set");
D)  asm(" tap");
E)  asm(" swi");
F)  DDRT &= ~0x08;
G)  DDRT &= ~0x10;
H)  DDRT |= 0x08;
I)  DDRT |= 0x10;
J)  DDRT = 0x08;
K)  DDRT = 0x10;
L)  TIOS &= ~0x08;
M)  TIOS &= ~0x10;
N)  TIOS |= 0x08;
O)  TIOS |= 0x10;
P)  TIOS = 0x08;
Q)  TIOS = 0x10;
R)  TMSK1&= ~0x08;  //812A4
    TIE  &= ~0x08;  //9S12C32
S)  TMSK1&= ~0x10;  //812A4
    TIE&= ~0x10;    //9S12C32
T)  TMSK1 |= 0x08;  //812A4
    TIE |= 0x08;    //9S12C32
U)  TMSK1 |= 0x10;  //812A4
    TIE |= 0x10;    //9S12C32
V)  TMSK1 = 0x10;   //812A4
    TIE = 0x10;     //9S12C32
W)  TMSK1 = 0x08;   //812A4
    TIE = 0x08;     //9S12C32
X)  TFLG1 &= ~0x10;
Y)  TFLG1 &= ~0x08;
Z)  TFLG1 |= 0x10;
AA) TFLG1 |= 0x08;
BB) TFLG1 = 0x10;
CC) TFLG1 = 0x08;
DD) TFLG1 = 0x00;
EE) TFLG1 = 0xFF;
FF) TSCR &= ~0x80;   //812A4
    TSCR1&= ~0x80;   //9S12C32
GG) TSCR |= 0x80;    //812A4
    TMSK2 = 0;       //812A4
    TSCR1|= 0x80;    //9S12C32
    TSCR2 = 0;       //9S12C32
HH) TC3 = TCNT+1000; // ritual
II) TC4 = TCNT+1000; // ritual
JJ) TC3 = TC3+1000;  // ISR
KK) TC4 = TC4+1000;  // ISR
LL) None of the above
```

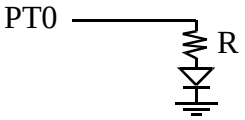
(5) Question 2. The objective of this question is to interface a solenoid to PH0 using a Darlington transistor. Choose the appropriate circuit for the interface. You may assume PH0 is an output pin of the 6812. *PLACE THE CORRECT LETTER ON THE ANSWER PAGE.*



(5) Question 3 Assume you have a buffered input interface. In other words, this is a producer-consumer problem When new input is ready, it generates an interrupt, and the new data is **Put** into a FIFO queue. The foreground thread (main program) will **Get** data out of the FIFO queue and process it. Debugging instruments placed on the FIFO have recorded the condition that the FIFO is usually full. What term best describes this situation?
PLACE THE CORRECT LETTER ON THE ANSWER PAGE.

A) I/O bound.
B) CPU bound.
C) FIFO bound.
D) Reentrant.
E) Nonreentrant.
F) Single-threaded.
G) Critical section.
H) Real-time system.

(10) Question 4. A low-current LED can be interfaced directly to the 6812 as shown below



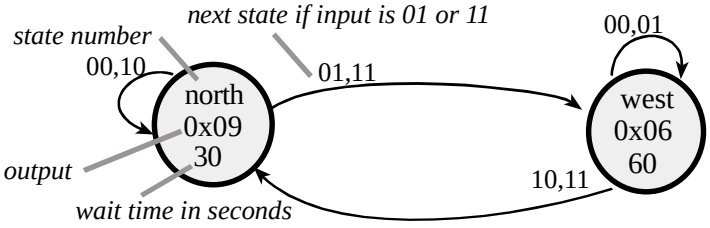
The LED voltage is 2V and its current is 2 mA.
Part a) For this interface to work, which inequality must be true? *PLACE THE CORRECT LETTER ON THE ANSWER PAGE.*

A) $I_{IH} > 2mA$
B) $I_{IL} > 2mA$
C) $I_{OH} > 2mA$
D) $I_{OL} > 2mA$
E) $I_{IH} < 2mA$
F) $I_{IL} < 2mA$
G) $I_{OH} < 2mA$
H) $I_{OL} < 2mA$

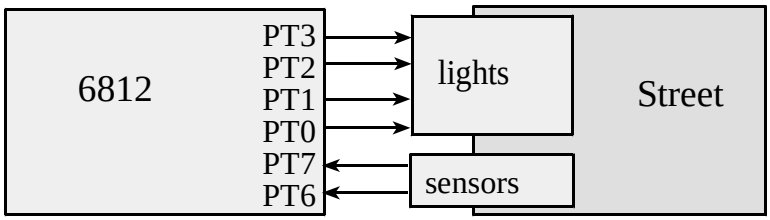
Part b) Assuming the $\mathbf{V_{OH}}$ of the 6812 is approximately +5V, Choose the appropriate resistor value for the interface. *GIVE YOUR ANSWER IN OHMS.*

(5) Question 5. An analog signal is sampled at 1000 Hz using an ADC. According to fundamental mathematical principles, what range of frequencies is reliably represented in the digital signals?

(35) Question 6. You will implement this traffic light finite state machine. Each state has a 4-bit output value, and four next state pointers. The sequence is output, wait, input, next.



The hardware uses 6 bits of PORTT, so your software will configure PORTT bits 7,6 to be inputs and bits 3-0 to be output. *For this question, don't worry about being friendly.*



You will write two regular C functions, **`ritual()`** and **`machine()`**. My **`main`** program calls your **`ritual()`** once before I enable interrupts. My RTI **`handler`** will call your **`machine()`** every 1 second. *You are not allowed to change my code.*

```
unsigned short Count;
void interrupt 7 handler(){ // interrupts at 244Hz
  CRGFLG = 0x80;    // acknowledge
  if((--Count)==0){ // every 1 second
    machine();      // your program
    Count = 244;
  }
}
void RTI_Init(void){
  CRGINT = 0x80; // arm
  RTICTL = 0x33; // period=250ns*4096*4=4096us or 244.14Hz
  Count = 244;   // interrupt counter
  asm cli
}
void main(void){
  ritual()       // your program
  RTI_Init();
  for(;;){
  }
}
```

Part a) I am giving you the specification of the linked data structure. You can not change this specification. Show the C code that defines this finite state machine

```
const struct State {
  unsigned char out;             // 4-bit output
  unsigned short wait;           // time in seconds to wait
  const struct Node *next[4];};  // Next if 2-bit input is 0-3
typedef const struct State StateType;
```

Part b) You may add global variables as needed.

Part c) Write the **`ritual()`** function. Among other things the ritual should initialize the PORTT direction register, **`DDRT`** and set the initial state to **`north`**.

Part d) Write the **`machine()`** function that implements the FSM. In particular, you will perform input and output using **`PORTT`**. Since this is executed in the background, no backwards jumps (**`do while`**, **`while`**, or **`for`**) are allowed.

If you combine my C code that I put on this page with your code that you place in parts a,b,c,d, it will constitute the entire software system to run this FSM.

Jonathan W. Valvano First:_____ Last:_____

April 7, 2004, 1:00pm-1:50pm. This is a closed book exam. You must put your answers on the boxes on the last two pages only. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting.***

(40) Question 1. Place one answer A-LL for each. (4 points each)

Arms output compare 4.	
Acknowledges output compare 3.	
Enables interrupts.	
Disables interrupts.	
Disarms output compare 3.	
Clears all 8 output compare flags.	
Specifies channel 4 is output compare.	
Specifies the period of the OC3 interrupt	
Activates TCNT timer at the fastest rate	
Specifies the time of the first OC3 interrupt	

(5) Question 2.	Letter A-H
(5) Question 3.	Letter A-H
(5) Question 4a.	Letter A-H
(5) Question 4b. Resistor value, in Ω	
(5) Question 5. frequency range in Hz	Minimum frequency= Maximum frequency=

(10) Question 6a. Show the C code that defines the finite state machine in ROM.

(5) Question 6b. Show the C code that defines necessary RAM-resident global variables.

(5) Question 6c. Show the `ritual()` function that initializes the finite state machine.

(15) Question 6d. Show the `machine()` function that executes the finite state machine.