Jonathan W. Valvano First:__________________ Last:______________________

April 10, 2002, 11:00am-11:50am

This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting.***

(50) Question 1. The objective of this problem is to design a frequency measurement system. Frequency (in cycles/minute) is defined as the number of cycles (i.e., the number of falling edges counted by the key wakeup ISR) in a fixed time (1 minute time delay implemented by the TOF interrupts.) The digital wave is connected to Port H bit 7 (PH7). The frequency ranges from 0 to 600,000 cycles/min. Your software system will count the number of falling edges on PH7 in one minute using a combination of key wakeup and TOF interrupts. Your system has exclusive access to Port H, key wakeup on H, and the TOF timer system. TOF interrupts will be used to wait the fixed time (1 minute.) Every one minute the TOF interrupt handler will update the following public global variable with a new frequency measurement.

```
long Frequency;   // public, units are cycles/min
```

Part a) Define additional global variables that you require. Specify whether these variables are public or private.

Part b) Show the ritual that initializes your frequency measurement software system. Once initialized your measurement process runs in the background without the help of any foreground software. The foreground software simply reads the 32-bit global `Frequency`. See the example main program in Part d).

Part c) Write the interrupt service routines for key wakeup H and TOF. Include the vectors.

```
#pragma interrupt_handler KeyHHndlr
void KeyHHndlr(void){

}
#pragma abs_address:
void (*KeyH_vect[])() = { KeyHHndlr};
#pragma end_abs_address
```

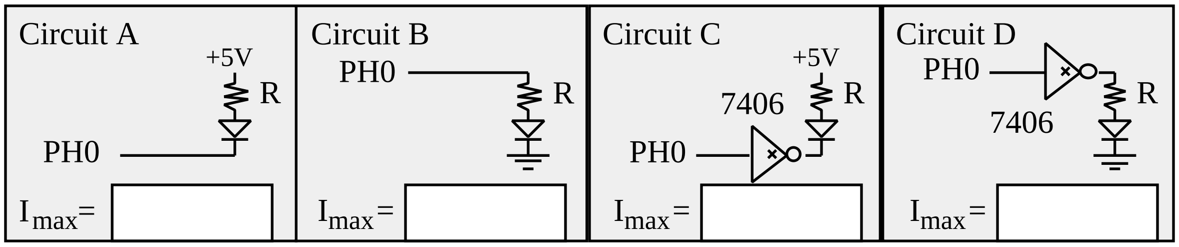
```
#pragma interrupt_handler TOFhndlr
void TOFhndlr(void){

}
#pragma abs_address:
void (*TOF_vect[])() = { TOFhndlr};
#pragma end_abs_address
```

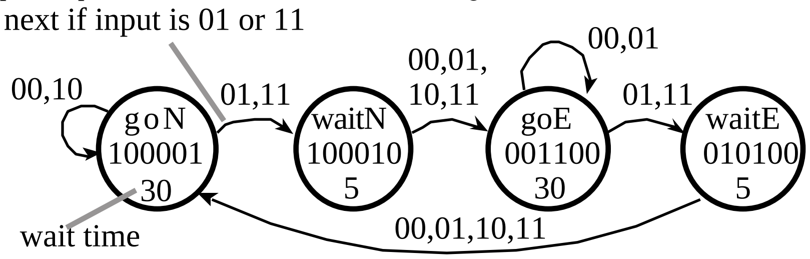
Part d) Consider the following `main` program. This is the only access to the global variable `Frequency`. Will there be any critical sections? Justify your answer.

```
void main(void){
  SCI_Init();
  Freq_Init();   // your ritual defined in part b)
  while(1){
    if(Frequency>1000){  // think about the assembly code for this
      SCI_OutString("too big");
      SCI_OutChar(CR);
    }
  }
}
```

(12) Question 2. Consider the following four LED interface circuits. Determine the range of LED currents allowable for each circuit. In particular, give the maximum LED current each circuit can drive. If a circuit cannot be used at all, specify NONE. PH0 is an output pin of the 6812.



(38) Question 3. You will be writing the C program that controls traffic at an intersection using a Moore FSM. The 2-bit inputs and the 6-bit outputs are given in binary. The wait times are given in seconds. The controller sequence is output, wait, input, and jump to next state, depending on the input. You are NOT allowed to make any modifications to the linked structure. Port A bits 7,6 are sensor inputs, and Port A bits 5-0 are traffic light outputs. You may NOT change the hardware connections. *Notice that the sensor inputs are connected to pins 7 and 6, and not 1 and 0.* All input/output uses Port A. The initial state is `goN`.



The `State` structure uses an index rather than a pointer to specify the next state. For example, if `CurrentState` is the index of the current state, and the input is 2 (binary 10), then the following code properly updates the index

```
 CurrentState = fsm[CurrentState].next[2];
```

The syntax and definition of the FSM cannot be changed.

```
const struct State {
  unsigned char out;        // Output to Port A
  unsigned char time;       // Time in sec to wait
  unsigned char next[4];};  // Next if input=00,01,10,11
typedef const struct State StateType;
#define goN   0
#define waitN 1
#define goE   2
#define waitE 3
StateType fsm[4]={
  {0x21, 60,{goN,waitN,goN,waitN}}, // goN   state
  {0x22,  5,{goE,goE,goE,goE}},     // waitN state
  {0x0C, 30,{goE,goE,waitE,waitE}}, // goE   state
  {0x14,  5,{goN,goN,goN,goN}}};    // waitE state
```

Part a) Show the ritual that initializes the traffic light system.

Part b) Show the main program that runs the controller in the foreground. No interrupts are needed. You may assume there is a helper function, `Wait1Sec(unsigned char)`, that waits the specified number of seconds. You do not have to write the function `Wait1Sec`.