Jonathan W. Valvano First:_____ Last:_____
March 6, 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.***

(20) Question 1. Find the sequence of execution that leads to the incorrect value being displayed. `Display` is a function that is called from the foreground, while `RTIHan` is a background interrupt service routine periodically activated by the RTIF flag. If the interrupt occurs at a bad time, and `time` is a certain value, then `Display` will output an incorrect value.
Part a) Mark each line of executable C with a "1", "2", "3",… symbol signifying the order of execution that yields a bad display. Also specify the value of `Time` before the interrupt (there is more than one).

```
unsigned short Time;

// called from the foreground
void Display(void){

  SCI_OutChar(Time/10+0x30); // tens

  SCI_OutChar(Time%10+0x30); // ones

}
```

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

  Time++;   // 0 to 59

  if(Time == 60) Time=0;

  RTIFLG = 0x80;
}
#pragma abs_address:0xfff0
void (*RTI_vector[])()={RTIHan};
#pragma end_abs_address
```

Part b) Rewrite the `Display` function to eliminate the bug.

(50) Question 2. You will write a software device driver for an input/output device. The output goes to a LCD display, and the input comes from a keypad. The I/O data are ASCII characters. The ritual should set the direction registers for ports A and B, initializing Port B to inputs. Your software should use busy-waiting (Gadfly) synchronization. All operations should be friendly. The hardware connections are as follows:

PA7	-->	**start**	your software sets this high to start an I/O operation
PA6	<--	**done**	this signal goes high when the I/O operation is complete
PA5	-->	**R/W**	1 means input, and 0 means output
PB7-PB0	<->	**data**	bidirectional, output to display, input from keypad

To output a letter to the LCD,
1) set **R/W** to 0
2) make Port B outputs
3) write Port B with the 8-bit **data** to display,
4) set **start** high
5) wait for **done** to be 1
6) set **start** low
7) wait for **done** to be 0
8) make Port B inputs again

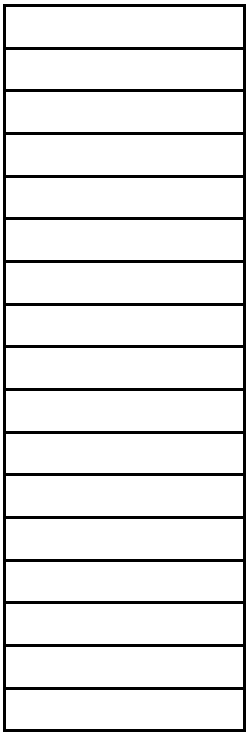
To input a letter from the keypad,
1) set **R/W** to 1
2) set **start** high
3) wait for **done** to be 1
4) read input data
5) set **start** low
6) wait for **done** to be 0

Part a) Show the C code that goes in the `IO.h` header file.

Part b) Show the C code that goes in the `IO.c` implementation file.

(30) Question 3. The overall goal is to draw a picture of the stack that exists while in the middle of the `RTIHan`. Starting at the top of `main`, hand-execute this software system; `main` calls `function`; the RTI interrupt occurs at the specified spot within `function`, and the `RTIHan` runs until the "`show stack`" comment. Local variables, parameters, and return values will be pushed on the stack. For each element pushed on the stack, give a general symbolic description (e.g., return pointer, old X, i1, f0) rather than its specific value (e.g., $F08A, $0000, 100, 4.)

```
#pragma interrupt_handler RTIHan()
void RTIHan(void){
short i1;
short static i2=100;
  i2--;
  i1 = i2;
// show stack at this point
  if(i1 == 0 ) i2=100;
  RTIFLG = 0x80;
}
#pragma abs_address:0xfff0
void (*RTI_vector[])()={RTIHan};
#pragma end_abs_address
short function(short f0){
  short f1;
  f1 = f0+4;
  // interrupt occurs here
  return f1;
}
void main(void){ short m1;
  m1 = function(4);
  while(1){}
}
extern void _start();
#pragma abs_address:0xfffe
void (*reset_vector[])()={_start};
#pragma end_abs_address
```



```
                        .area data
--- 0000          L2:   .blkb 2
                        .area idata
--- 0000 0064          .word 100
                        .area text
 F03B  34          _RTIHan::  pshx
 F03C  B775              tfr s,x
 F03E  1B9E              leas -2,sp
 F040  FC0800            ldd L2
 F043  830001            subd #1
 F046  7C0800            std L2
 F049  18011E0800        movw L2,-2,x
                ;  // show stack at this point
 F04E  ED1E              ldy -2,x
 F050  8D0000            cpy #0
 F053  2606              bne L3
 F055  180300640800      movw #100,L2
 F05B  180B800015  L3:   movb #128,0x15
 F060  B757              tfr x,s
 F062  30                pulx
 F063  0B                rti
                        .org 0xfff0
 FFF0              _RTI_vector::
 FFF0  F03B              .word _RTIHan
                        .area data
                        .area text
 F064              _function::
 F064  3B                pshd
 F065  34                pshx
 F066  B775              tfr s,x
 F068  1B9E              leas -2,sp
 F06A  EC02              ldd 2,x
 F06C  C30004            addd #4
 F06F  6C1E              std -2,x
            ; // interrupt occurs here
 F071  EC1E              ldd -2,x
 F073  B757              tfr x,s
 F075  30                pulx
 F076  1B82              leas 2,sp
 F078  3D                rts
 F07F  34          _main:: pshx
 F080  B775              tfr s,x
 F082  1B9C              leas -4,sp
 F084  CC0004            ldd #4
 F087  16F064            jsr _function
 F08A  6C1C              std -4,x
 F08C  B746              tfr d,y
 F08E  6D1E              sty -2,x
 F090  20FE        L7:   bra L7
 F092  B757              tfr x,s
 F094  30                pulx
 F095  3D                rts
                        .org 0xfffe
 FFFE              _reset_vector::
 FFFE  F000              .word __start
```