Jonathan W. Valvano First Name: _____ Last Name:_____
October 15, 2003, 1 to 1:50pm

This is an open book, open notes exam. You may put answers on the backs of the pages, but please don't turn in any extra sheets.

(20) Question 1. Assume an input serial channel is operating at a baud rate of 100 bits/sec. The interface must satisfy a real-time constraint, in order to guarantee that no data is lost. The channel uses interrupt synchronization, and the ISR is shown below in part b). The header file for **RxFifo** is

```
#define RXFIFOSIZE 8
/* Number of characters in Fifo, is full when it has FifoSize-1 characters */
void RxFifo_Init(void);

/* Enter one character into the fifo
   Inputs: 8-bit data
   Outputs: true if data is properly saved */
int RxFifo_Put(char);

/* Remove one character from the fifo
   Inputs: pointer to place to save 8-bit data
   Outputs: true if data is valid */
int RxFifo_Get(char *);

/* Check the status of the fifo
   Inputs: none
   Outputs: true if there is any data in the fifo */
int RxFifo_Status(void);
```

(10) Part a) What specific timing constraint must be satisfied to prevent loss of data? Be as specific as possible, giving a quantitative answer.

(10) Part b) Modify the following code to count the number of characters lost? Put the result in a global.

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

  if(SC0SR1 & RDRF){

    RxFifo_Put(SC0DRL); // clears RDRF

  }

}
```

(25) Question 2. Consider a problem of running two foreground threads using a preemptive scheduler with semaphore synchronization (like Lab 17.) There are two shared 16-bit global variables which contain the position of an object:

```
short TheX,TheY;
```

Both, the `client` and `server` wish to access this data. The basic shell of this operation is given. Define one or more semaphores, then add calls to the following three functions in order to properly synchronize the interactions between `client` and `server`. The goal is to prevent corrupted data.

```
int OS_InitSemaphore(Sema4Type *semaPt, short value);
void OS_Wait(Sema4Type *semaPt);
void OS_Signal(Sema4Type *semaPt);
```

You will define one or more semaphores and calls to the semaphore functions, otherwise no other changes are allowed. Assume `server` is run first. You may assume the only accesses to `TheX` and `TheY` in the entire software system are explicitly shown here.

```
void server(void){
 short sX,sY;
 TheX = 5;
 TheY = 6;
 sInit();   // initialization
 while(1){
  sX= TheX;
  sY= TheY;
  sProcess(&sX,&sY); // body
 }
}
```

```
void client(void){
 short cX,cY;
 cInit();   // initialization
 while(1){
  cX= TheX;
  cY= TheY;
  cProcess(cX,cY);  // body
 }
}
```

If you wish to make no changes at all, explain why, otherwise for each semaphore you add, explain what it means to be 0, 1 etc.

(20) Question 3. Consider the following thread switch system, which is essentially the initial system used in Lab 17 with a status field added to implement blocking. The changes to the original Lab17 starter file are shown in **bold face**.

```
struct TCB{
 Sema4Type *status;
 struct TCB *Next;
 unsigned char *StackPt;
 unsigned char Id;
 unsigned char MoreStack[100];
 unsigned char InitialCCR;
 unsigned char InitialRegB;
 unsigned char InitialRegA;
 unsigned int InitialRegX;
 unsigned int InitialRegY;
 void (*InitialPC)(void);
};
void threadSwitch(void){

  RunPt = RunPt->Next;

  while(RunPt->Status)

    RunPt = RunPt->Next;

  PORTJ = RunPt->Id;
}
```

```
void threadSwitchISR(void){

  asm(" ldx _RunPt\n"

  " sts 2,x");

  RunPt = RunPt->Next;

  PORTJ = RunPt->Id;

  TC3 = TCNT+TimeSlice;

  TFLG1 = 0x08;


  asm(" ldx _RunPt\n"

  " lds 2,x");

}
```

(10) Part a) Explain why this system crashes just because **`status`** was added to the TCB.

(10) Part b) Without moving the position of the **`status`** entry (leaving it the TCB structure unchanged), make other changes to the above program to fix the bug.

(20) Question 4. Lab 17 measured a time-jitter. It was usually a small number.
(10) Part a) How is time-jitter defined? What does it mean?

(10) Part b) The time-jitter causes a voltage measurement error. Specifically, what is the relationship between time-jitter and voltage error?

(15) Question 5. What bad thing might happen if **`OS_Wait`** does not disable interrupts? I.e., what can go wrong with the following spinlock implementation?

```
void OS_Wait(Sema4Type *semaPt){
  while(semaPt->Counter <= 0){
  }
  (semaPt->Counter)--;
}
```