Jonathan W. Valvano First Name: _____ Last Name:_____
November 3, 2004, 1 to 1:50pm

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

**(30) Question 1**. The overall goal is to sample channel 5 of the 10-bit ADC at 1000 samples/sec and output the raw data through the SCI Transmitter *without executing any backward jumps in the ISR*. The E clock, the M clock, and the TCNT are all 24 MHz.

**Part a)** If the SCI transmitter is idle (its data register and shift register are both empty) you can output two data bytes to the SCI one right after the other, and have both bytes properly transmitted. I.e., if the SCI transmitter is idle, this code will transmit my initials. Explain why. In particular, where does the **'J'** go, and where does the **'V'** go?

```
SCIDRL = 'J';
SCIDRL = 'V';
```

**Part b)** Put hexadecimal values in the four boxes to make the ADC enabled in continuous sampling mode right-justified and unsigned, the SCI enabled but not armed, and the output compare 0 channel enabled and armed.

```
void Ritual(void){
  ATDCTL2 = 0x80; // power up ADC
  ATDCTL4 = 0x05; // 10-bit ADC
 ATDCTL5 = [          ];
// bit 7 DJM Result Register Data Justification,
//       1=right justified, 0=left justified
// bit 6 DSGN Result Register Data Signed or Unsigned Representation
//       1=signed, 0=unsigned
// bit 5 SCAN Continuous Conversion Sequence Mode
//       1=continuous, 0=single
// bit 4 MULT Multi-Channel Sample Mode
//       1=multiple channel, 0=single channel
// bit 3 0
// bit 2-0 CC,CB,CA channel number 0 to 7
  SCIBD = [**discussed in part d**];   // 24MHz/(16*BaudRate)
  SCICR1 = 0;// 1 start, 8 data, 1 stop,  no parity
 SCICR2 = [          ];
/*  7   TIE, transmit interrupts on TDRE
    6   TCIE, transmit interrupts on TC
    5   RIE, receive interrupts on RDRF
    4   ILIE, interrupts on idle
    3   TE, enable transmitter
    2   RE, enable receiver
    1   RWU, receiver wakeup
    0   SBK, send break */
 TIOS  |= [          ];  // TC0 as output compare
  TSCR1 = 0x80;   // Enable TCNT
  TSCR2 = 0x00;   // 24MHz TCNT, TOI disarm
  PACTL = 0;      // timer prescale used for TCNT
```

```
  TIE  |= [              ];     // arm OC0
  TC0    = TCNT+50; // first interrupt right away
  asm cli
}
```

**Part c)** Write the output compare 0 ISR that first reads the most recent 10-bit ADC sample, then outputs the binary data (not ASCII) using two SCI transmission frames. This ISR code should run without any backwards jumps. In Metrowerks, OC0 is interrupt 8.

**Part d)** What is the slowest SCI baud rate that can be used to solve this real time system? Show the calculations used to determine this baud rate.

**(25) Question 2.** Consider a problem of running three foreground threads using a preemptive scheduler with semaphore synchronization. Each thread has a central **`body()`** containing code that should be executed together. The basic shell of this system is given. Define one or more semaphores, then add semaphore function calls to implement a ***three-thread rendezvous***. Basically, each time through the **`while`** loop, the first two threads to finish their **`start()`** code will wait for the last thread to finish its **`start()`** code. Then, all three threads will be active at the same time as they execute their corresponding **`body()`**. The semaphore prototypes:

```
void Init (Sema4Type *semaPt, short value);
void Wait(Sema4Type *semaPt);
void Signal(Sema4Type *semaPt);
```

You will define one or more semaphores and add calls to these three semaphore functions, otherwise no other changes are allowed. You may assume **`thread1`** runs first.

```
void thread1(void){
 init1();
 while(1){
   start1();
   body1();
   end1();
 }
}
```

```
void thread2(void){
 init2();
 while(1){
   start2();
   body2();
   end2();
 }
}
```

```
void thread3(void){
 init3();
 while(1){
   start3();
   body3();
   end3();
 }
}
```

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.

**(25) Question 3**. In this question you will write C code to implement a spinlock binary semaphore. Binary semaphores have two values, 0 means busy, 1 means free. They can be used to implement mutual exclusion. Assume the semaphore type is a simple signed integer, such as

```
short s1;  // semaphore
```

Write three functions to implement this spinlock binary semaphore. Full credit will be given to solutions that do not utilize increment, or decrement.

**Part a)** The initialization function will initialize the semaphore value to 0 or 1.

```
void OS_InitSemaphore(short *semaPt, short value){
  *semaPt = value;
}
```

**Part b)** The wait function will test the semaphore value. If the value is zero it will wait and test it again. If the value is one, it will set it to zero and return.

```
void OS_Wait(short *semaPt){
  asm sei      // Test and set is atomic
  while(*semaPt== 0){  // disabled
    asm cli            // disabled
    asm nop            // enabled
    asm sei            // enabled
  }
  *semaPt = 0;
  asm cli              // disabled
}                      // enabled
```

**Part c)** The signal function will set the value to one and return.

```
void OS_Signal(short *semaPt){
  *semaPt = 1;
}
```

**(10) Question 4.** In Lab 17, we defined time-jitter, $d\mathbf{t}$, as the difference between when a periodic task is supposed to be run, and when it is actually run. The goal of a real-time DAS is to start the ADC at a periodic rate, $\Delta t$. Let $t_n$ be the nth time the ADC is started. In particular, the goal to make $t_n - t_{n-1} = \Delta t$. The jitter is defined as the constant, $d\mathbf{t}$, such that

$\Delta t - d\mathbf{t} < t_i - t_{i-1} < \Delta t + d\mathbf{t}$ for all i.

Consider the situation where the time jitter is unacceptably large. Which modification to the system will cause the largest improvement in time jitter? Just circle your selection.

A) Run the ADC in continuous mode

B) Convert from spinlock semaphores to blocking semaphores

C) Change from round robin to priority thread scheduling

D) Reduce the amount of time the system runs with interrupts disabled.

E) Increase the size of the DataFifo

**(10) Question 5.** Select the best term that describes each definition.

Part a) A technique to periodically increase the priority of low-priority threads so that low priority threads occasionally get run. The increase is temporary.

Part b) The condition where low priority threads never get run.

Part c) The condition where thread 1 is waiting for a resource held by thread 2, and thread 2 is waiting for a resource held by thread 1.

Part d) The condition where a thread is prevented from running because it needs something that is unavailable.

Part e) The condition where once a thread blocks, there are a finite number of threads that will be allowed to proceed before this thread is allowed to proceed.

**word bank**

| | |
|---|---|
| active | nonrentrant |
| aging | preemptive scheduler |
| blocked | producer-consumer |
| bounded buffer | reentrant |
| bounded waiting | rendezvous |
| critical section | round robin scheduler |
| deadlock | sleeping |
| killed | spin lock |
| maximum latency | starvation |