Jonathan W. Valvano November 3, 2004, 1 to 1:50pm

(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) The **'J'** goes in the shift register, and the **'V'** goes in the data register.

Part b)

```
 ATDCTL5 = 0xA5          ;
// bit 7 1=right justified, 0=left justified
// bit 6 0=unsigned
// bit 5 1=continuous, 0=single
// bit 4 0=single channel
// bit 2-0 101 channel number 5
SCICR2 =  0x0C or 0x08          ;
/*  7   TIE, no transmit interrupts on TDRE
    6   TCIE, no transmit interrupts on TC
    5   RIE, no receive interrupts on RDRF
    4   ILIE, no interrupts on idle
    3   TE, yes enable transmitter
    2   RE, yes/no enable receiver
    1   RWU, no receiver wakeup
    0   SBK, nosend break */
 TIOS |=   0x01    ;  // TC0 as output compare
 TIE  |=  0x01     ;  // arm OC0
```

Part c)

```
interrupt 8 void OC0Handler(void){ unsigned short data;
  data = ATDDR0;      // read most recent 10bit ADC sample
  SCIDRL = data>>8;   // top 2 bits
  SCIDRL = 0xff&data; // bottom 8 bits
}
```

Part d) Match the SCI bandwidth with the ADC bandwidth

Baudrate(bits/sec)*1frame/10bits*1byte/frame > 2bytes/sample*1000samples/sec

Baudrate> 20000 bits/sec

(25) Question 2. implement a ***three-thread rendezvous***

Semaphore sij=1 means thread i allows thread j to proceed. sij=0 means thread i is not done with its start code and requests that thread j to wait.

`void thread1(void){`
`Init(&s12,0);`
`Init(&s13,0);`
`Init(&s21,0);`
`Init(&s23,0);`
`Init(&s31,0);`
`Init(&s32,0);`
	`void thread2(void){`	`void thread3(void){`

```
 init1();
 while(1){
   start1();
   Signal(&s12);
   Signal(&s13);
   Wait(&s21);
   Wait(&s31);
   body1();
   end1();
 }
}
```

```
init2();
 while(1){
   start2();
   Signal(&s21);
   Signal(&s23);
   Wait(&s12);
   Wait(&s32);
   body2();
   end2();
 }
}
```

```
 init3();
 while(1){
   start3();
   Signal(&s31);
   Signal(&s32);
   Wait(&s23);
   Wait(&s13);
   body3();
   end3();
 }
}
```

(25) Question 3. In this question you will write C code to implement a spinlock binary semaphore.

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. The jitter is caused by a delayed service of the producer background ISR, so
D) Reduce the amount of time the system runs with interrupts disabled.

(10) Question 5. Select the best term that describes each definition.
Part a) **aging**
Part b) **starvation**
Part c) **deadlock**
Part d) **blocked**
Part e) **bounded waiting**