Jonathan W. Valvano
November 16, 2007, 1:00pm-1:50pm. This is an open book test No electronic devices allowed. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting***.

(15) Question 1.
The rise of E occurs at 500ns, the fall of OE occurs at 500+[5,15], RDA starts at 500+[5,15]+ $\mathbf{t_a}$. The worst case is the later RDA = 515+ $\mathbf{t_a}$. RDR starts at 950ns. RDA must overlap RDR, so 515+ $\mathbf{t_a} \le$ 950ns, or $\mathbf{t_a} \le$ 435ns.
This was no part of the question, but what is the largest value possible for $\mathbf{t_s}$ so that write data available overlaps write data required?
The fall of E occurs at 1000ns, the rise of WE occurs at 1000+[5,15], WDR starts at 1000+[5,15]- $\mathbf{t_s}$. The worst case is the earlier WDR = 1005- $\mathbf{t_s}$. WDA starts at 500+128ns=628ns. WDA must overlap WDR, so 628ns $\le$ 1005- $\mathbf{t_s}$, or $\mathbf{t_s} \le$ 377ns.

(25) Question 2. The goal of this interface is to provide 8 digital inputs using a 74HC165
Part a) There is only one answer, it must be CPOL=1, CPHA=0

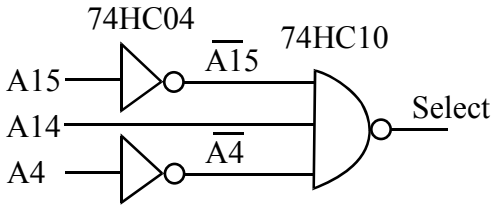
```
void SPI_Init(void){
  DDRM = (DDRM&0x03)|0x38; //PM2 in,PM5-3 out
  PTM |= 0x08;   // PL=1, disable latch
  SPICR1 = 0x58; // master, CPOL=1, CPHA=0
  SPICR2 = 0x00; // normal output
  SPIBR = 0;     // fastest speed
}
```

Part b) Write to SPIDR to start transmission, read from SPIDR when done

```
unsigned char SPI_In(void){
  PTM &= ~0x08;         // clear PL to parallel latch
  PTM |= 0x08;          // PL=1, latching data into 74HC165
  while(((SPISR&0x20)==0)){};// 1) wait for SPTEF=1,
  SPIDR = 0x00;              // 2) dummy output to start SPI
  while(((SPISR&0x80)==0)){};// 3) wait for SPIF=1,
  return SPIDR;              // 4) read result, clear SPIF
}
```

(10) Question 3. Design a minimal-cost negative-logic address decoder for *YourDevice*. Choose A4 to separate from I/O, choose A14 to separate from RAM, choose A15 to separate from ROM. Activate *YourDevice* if A4=0, A14=1, A15=0.

Select = not(not(A4) * A14 * not(A15)) = A4 + not(A14) + A15



(30) Question 4. Use output compare 7 to create a sampling rate of 1000 Hz.
Part a) To what value should you initialize **ATDCTL2**? Give your answer in hex.

```
ATDCTL2 = 0x80; // turn on
```

Part b) The goal is to initialize the ADC so that one start command will sample PAD3 exactly four times, no more no less. To what value should you initialize **ATDCTL3**? Give your answer in hex.

```
ATDCTL3 = 0x20; // sequence length = 4
```

Part c) In order to reduce noise, the 10-bit ADC will be operated at the slowest possible rate.

```
// choose m so 2(m+1)/Eperiod is 2us, 2(m+1)/0.25=2, m=3
     ATDCTL4 = 0x63; //s=18, m=3
```

Part d) Show the output compare 7 ritual.

```
void OC7_Init(void){
  TIOS |= 0x80;     // activate TC7 as output compare
  TSCR1 = 0x80;     // Enable TCNT, 4MHz in run mode
  TSCR2 = 0x02;     // divide by 4 TCNT prescale, TCNT at 1MHz
  PACTL = 0;        // timer prescale used for TCNT
  TIE  |= 0x80;     // arm OC7
  TC7   = TCNT+50;  // first interrupt right away
  asm cli
}
```

Part e) Show the entire output compare 7 interrupt service routine at implements the 1 kHz sampling.

```
interrupt 15 void TC7handler(void){ // executes at 1000 Hz
  TFLG1 = 0x80;         // acknowledge OC7
  TC7 = TC7+1000;       // 1 ms
  ATDCTL5 = 0x83;       // start sequence of four conversions
  while((ATDSTAT0&0x80)==0){};    // wait for SCF
  Average = (ATDDR3+ATDDR2+ATDDR1+ATDDR0)/4;
}
```

(20) Question 5. Design a circuit with three analog inputs V_0, V_1, V_2 and one analog output V_{out}
Add ground gain so sum of gains is 1

$V_{out} = 1/2\ V_0 + 1/4\ V_1 + 1/8\ V_2 + 1/8\ V_g$

Choose R_f (10 kΩ) a common multiple of gains, 1/2 1/4 1/8
Choose input resistors to get the desired gain

$R_f / R_0 = 1/2$ thus $R_0 = 20$ kΩ,
$R_f / R_1 = 1/4$ thus $R_1 = 40$ kΩ,
$R_f / R_2 = 1/8$ thus $R_2 = 80$ kΩ,
$R_f / R_g = 1/8$ thus $R_g = 80$ kΩ.

Build circuit, all inputs are positive, so they go to + terminal of op amp. (This is a 3-bit DAC, similar to the 4-bit DAC we currently build in EE319K)