Jonathan W. Valvano First Name: __________________ Last Name:___________________________

December 19, 2000, 9 am to 12 noon

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.

(15) Question 1. In this problem you will modify a pulse-width measurement system extending the range to 100 ms. The digital signal is connected to PT1.

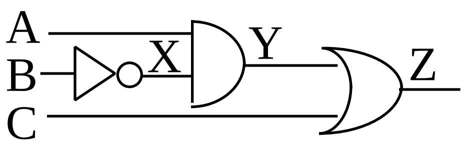
(5) Part a) What is the best pulse-width resolution possible that allows pulses up to 100 ms to be measured?

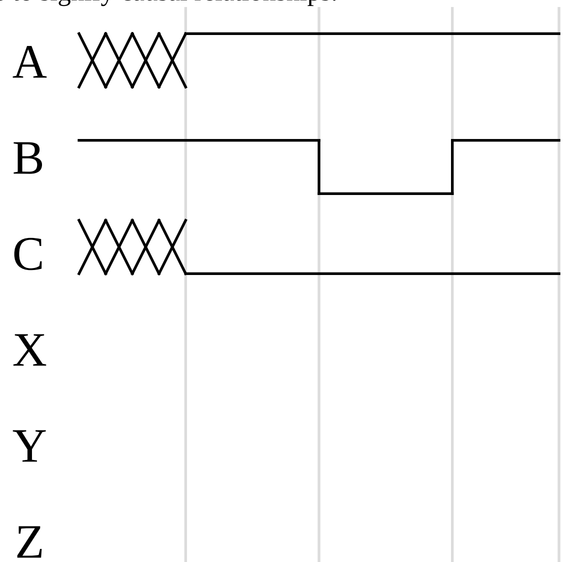
(5) Part b) The following is the pulse-width measurement program from Chapter 6. Modify this program so that the pulse width measurement range includes 100 ms. Add comments to explain *each change* you make.

```
unsigned int Rising; // TCNT at rising
unsigned char Done;  // Set each falling
#define PT1 0x02     // the input signal
#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
  if(PORTT&PT1){    // PT1=1 if rising
     Rising=TC1;}   // Setup for next
  else{
     PW=TC1-Rising; // the measurement
     Done=0xFF;}    // ack, fast clr C1F=0
void Ritual(void) {
 asm(" sei");  // make atomic
 TIOS&=~PT1;   // clear bit 1
 DDRT&=~PT1;   // PT1 is input capture
 TSCR=0x90;    // enable, fast clear
 TMSK2=0x32;   // 500 ns clock
 TCTL4|=0x0C;  // Both edges IC1
 TMSK1|=0x02;  // arm IC1
 TFLG1=0x02;   // clear C1F
 Done=0;
 asm(" cli");}
```

(5) Part c) List the factors affect the minimum pulse-width that can be measured with this new system?

(20) Question 2. Consider the following digital circuit with three inputs A,B,C. Assume a 10 ns gate delay through each gate. Draw the timing signals for X,Y,Z. Use arrows to signify causal relationships.





(20) Question 3. Consider the timing between the computer and a DS1225AB-150.

Part a) Give an equation for Read Data Available using terms like $\uparrow\overline{OE}$.

Part b) Using write cycle 2, give an equation for Write Data Required using terms like $\uparrow\overline{OE}$.

(10) Question 4. The following two-channel DAS has a bug. The FIFO works fine, but the problem is in how the FIFO is used. It is not a problem with reentrancy. Instead, look carefully at what happens when the FIFO becomes full.

```
#define FIFOSize 20
unsigned char FIFO[FIFOSize];
unsigned char *PutPt=&FIFO[0];
unsigned char *GetPt=&FIFO[0];
// 8-bit data stored in FIFO
// returns -1 if full, unsuccessful
// returns 0 if stored OK
int PutFifo(unsigned char data){
unsigned char *PPt;
  PPt=PutPt;     // Temporary pointer
  *PPt++=data;   // Try and store
  if(PPt==&FIFO[FIFOSize])
    PPt=&FIFO[0]; // Wrap
  if(PPt==GetPt) return(-1); // Full
  PutPt=PPt;
  return(0);} // OK
// 8-bit data removed from FIFO
// returns -1 if empty (no data)
// returns 0 if stored OK
int GetFifo(unsigned char *data){
  if(PutPt==GetPt) return(-1); // Empty
  *data=*GetPt++;  // remove
  if(GetPt==&FIFO[FIFOSize])
    GetPt=&FIFO[0]; // Wrap
  return(0);} // OK
```

```
#define Rate 2000
#define OC5  0x20
unsigned int FullError=0;
#pragma interrupt_handler TOC5handler()
void TOC5handler(void){
   TFLG1=OC5;       // ack OC5F
   TC5=TC5+Rate;    // Executed every 1 ms
  if(PutFifo(A2D(0)))
    FullError++;
  if(PutFifo(A2D(1)))
    FullError++;}
void main(){ unsigned char x0,x1;
  TIOS|=OC5;     // enable OC5
  TSCR|=0x80;    // enable
  TMSK2=0x32;    // 500 ns clock
  TMSK1|=OC5;    // Arm output compare 5
  TFLG1=OC5;     // Initially clear C5F
  TC5=TCNT+Rate; // First one in 1 ms
asm(" cli");
  while(1) {
    while(GetFifo(&x0)){};
    while(GetFifo(&x1)){};
    process(x0,x1);
  }
}
```

Part a) Explain the sequence of steps that might occur, such that the data streams get reversed. In particular, how could it be that the variable `x0` gets the data from ADC channel 1, and `x1` gets the data from channel 0?

Part b) Briefly explain how you could eliminate this bug.

(20) Question 5. Print statements are easy-to-use debugging instruments, but they suffer from two limitations. The first problem is intrusiveness, which means if it takes too long to print, system timing will be affected. The second problem is removing them all after debugging is complete. In this question, we will address both these limitations. In this problem you can assume the serial port is used exclusively for debugging. You can also assume the interrupt-driven version in SCI12A.C is being used.

(10) Part a) First, we will address intrusiveness. Below is the code from SCI12A.C for `OutChar`

```
void OutChar(char data){
   while (TxPutFifo(data) == 0){};
   SC0CR2=0xAC;}/* arm TDRE */
```

First, we make the TxFifo as big as possible. Next, we change the way `OutChar` works by discarding data when the output channel is too busy. This means we will tolerate occasional situations where a debugging output issued but it is never transmitted. **Rewrite `OutChar` so that it is minimally intrusive** (reduce the upper bound on execution speed of this function.) Think about what causes long execution speeds and eliminate that situation.

(10) Part b) The DEBUG macro can be used to switch between debugging and release modes. In release mode, the line with DEBUG at the front becomes a comment. _DEBUG will be true in debug mode and false in release mode.

```
#define SLASH(s) /##s
#if _DEBUG
#define  DEBUG
#else
#define DEBUG SLASH(/)
#endif
```

Consider the following debugging initialization

```
void main(void){
  Init();   // initialize COP, Port T
  DEBUG InitSCI();
  DEBUG OutString("Debug active\n");
```

If `_DEBUG` is true, the macros expand to

```
void main(void){
  Init();   // initialize COP, Port T
  InitSCI();
  OutString("Debug active\n");
```

If `_DEBUG` is false, the macros expand to

```
void main(void){
  Init();   // initialize COP, Port T
  // InitSCI();
  // OutString("Debug active\n");
```

Write a macro called `SCAN` that has two parameters. The first is a string, and the second is a variable to observe. For example, if `_DEBUG` is true then `SCAN("a=",a)` gives

```
  OutString("a="); OutUDec(a); OutChar(13);
```

If `_DEBUG` is false then `SCAN("a=",a)` produces no code.

(15) Question 6. Consider the fuzzy control system in Section 13.5.1 in the book. You can find it on pages 749-756. Your job in this question is to work through the fuzzy equations for the first control iteration. I.e., assume the motor is at rest, so both T and Told are zero (T'(n)=T'(n-1)=0). Assume also that the desired speed is S*=55 rpm (T*=14). Notice that all three adjustment parameters TE, TD and TN are 20.

Part a) Calculate the crisp inputs

E the error in motor speed in 3.9rpm ___________________

D the change in motor speed in 3.9rpm/time ___________________

Part b) Calculate the input fuzzy membership sets

Slow True if the motor is spinning too slow ___________________

OK True if the motor is spinning at the proper speed ___________________

Fast True if the motor is spinning too fast ___________________

Up True if the motor speed is getting larger ___________________

Constant True if the motor speed is remaining the same ___________________

Down True if the motor speed is getting smaller. ___________________

Part c) Calculate the output fuzzy membership sets

Decrease True if the motor speed should be decreased ___________________

Same True if the motor speed should remain the same ___________________

Increase True if the motor speed should be increased ___________________

Part d) Calculate the crisp output

ΔN change in output, N=N+ΔN in DAC units ___________________