(15) Question 1. In this problem you will modify a pulse-width measurement system extending the range to 100 ms.
(5) Part a) 2us is the best pulse-width resolution possible that allows pulses up to 100 ms to be measured.

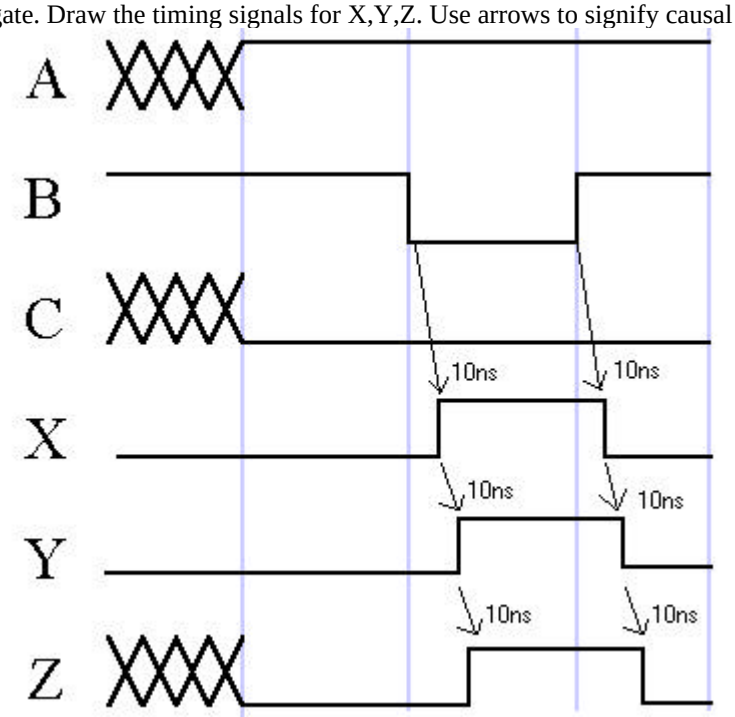
Resolution	Precision	Maximum PW
125ns	65536	8ms
250ns	65536	16ms
500ns	65536	32ms
1us	65536	64ms
2us	**65536**	**128ms**
4us	65536	256ms

(5) Part b) Modify this program so that the pulse width measurement range includes 100 ms. Only change is

```
TMSK2=0x34;   // 2us clock
```

(5) Part c) The factors that affect the minimum pulse-width that can be measured include 1) time to execute the ISR, 2) time to finish the instruction being executed when the interrupt is requested, and 3) the longest time the software runs with interrupts disabled.

(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)RDA= (later(AdV+t_{ACC},↓CE+t_{CO},↓OE+t_{OE}),earlier(AdN+t_{OH},↑OE,↑CE)
= (later(AdV+150,↓CE+150,↓OE+70),earlier(AdN+5,↑OE,↑CE)

Part b) WDR=(↑CE-t_{DS},↑CE+t_{DH2})=(↑CE-60,↑CE+10)

(10) Question 4. The following two-channel DAS has a bug.
Part a) One possible sequence of steps that might occur is
1) the fifo is almost full, when an interrupt occurs
2) the ISR puts channel 0 data into the FIFO properly
3) a full error occurs when the ISR puts channel 1 data into the fifo, this data is lost
4) the foreground runs and gets many data points, so there is lots of room in the fifo
5) the next interrupt occurs and the ISR puts channel 0 and channel 1 data
Part b) There are two solutions. If you use two fifos, one for each channel, then it is possible for a time shift between samples to occur if one fifo is full and the other almost full at the time of an interrupt. The best solution is to pack the samples into a single structure and put and get them together. E.g.,.

```
#define FIFOSize 20
unsigned short FIFO[FIFOSize];
unsigned short *PutPt=&FIFO[0];
unsigned short *GetPt=&FIFO[0];
// 16-bit data stored in FIFO
// returns -1 if full, unsuccessful
// returns 0 if stored OK
int PutFifo(unsigned short data){
unsigned short *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
// 16-bit data removed from FIFO
// returns -1 if empty (no data)
// returns 0 if stored OK
int GetFifo(unsigned short *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){
unsigned short data;
  TFLG1=OC5;     // ack OC5F
  TC5=TC5+Rate;  // Executed every 1 ms
  data=(A2D(0)<<8)+ A2D(1);
  if(PutFifo(data))
    FullError++;}
void main(){ unsigned short data;
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(&data)){};
    x0=data>>8; x1=data&0x00ff;
    process(x0,x1);
  }
}
```

(20) Question 5. Print statements are easy-to-use debugging instruments, but they suffer from two limitations.

(10) Part a) The new code for `OutChar`

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

(10) Part b) The DEBUG macro can be used to switch between debugging and release modes.

```
#define SCAN(s,d) DEBUG OutString(s); OutUDec(d); OutChar(13);
```

(15) Question 6. Consider the fuzzy control system in Section 13.5.1 in the book.

Part a) Calculate the crisp inputs

E = T*-T' = 14

D = T'(n)-T'(n-1) = 0

Part b) Calculate the input fuzzy membership sets

Slow=(255*E)/TE=(255*14)/20=178

OK=255-Slow=77

Fast=0

Up=0

Constant=255

Down=0

Part c) Calculate the output fuzzy membership sets

Same=(OKandConstant)=(min(77,255))=77

Decrease=(OKandUp)or(FastandConstant)or(FastandUp)=max(min(77,0),min(0,255)min(0,0))=0

Increase=(OKandDown)or(SlowandConstant)or(SlowandDown)=max(min(77,0),min(178,255)min(0,0))=178

Part d) Calculate the crisp output

ΔN=(-20*Decrease+Same•0+20*Increase)/(Decrease+Same+Increase)=(0•-20+77•0 +178•20)/255=13