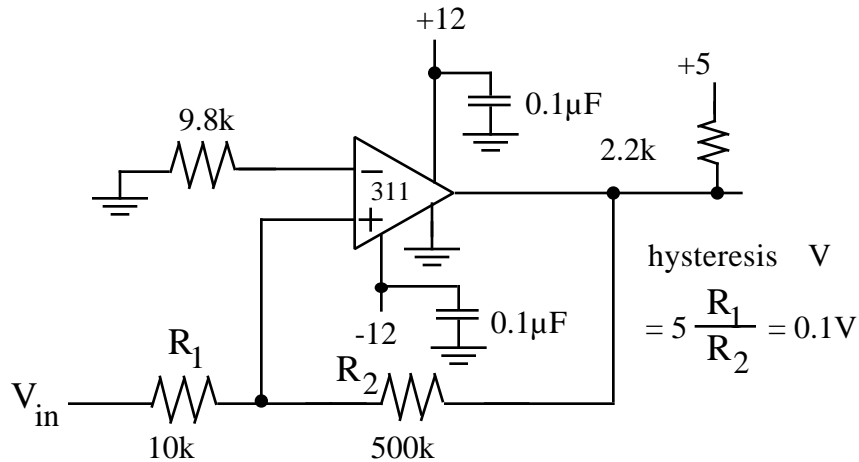


Jonathan W. Valvano November 5, 1999, 11:00am-12:45pm

(40) **Question 1.** In this problem you will design an ultrasonic doppler velocity meter.

(5) Part a) The period measurement resolution is
 $p = 25.25 - 25 = 0.25 \mu\text{s}$

(10) Part b) The hardware interface between the $\pm 5 \text{ V}$ sine wave and the Port T PT2



(10) Part c) The ritual subroutine initializes the interface.

```
// PT2/IC2 input = external signal, rising edge to rising edge
// resolution = 250ns, Range = 25 μs to 2.5 ms,
// output compare PT3 used for overflow checking
unsigned short v; // resolution=1 cm/sec, range is 0 to 100 cm/sec
unsigned short Period; // resolution=250ns, range is 100 to 10000
unsigned short First; // TCNT first edge
#define OC3 0x08
#define IC2 0x04
void Ritual(void) {
    asm(" sei "); // make atomic
    TIOS |= OC3; // PT3 output compare
    TIOS &= ~IC2; // PT2 input capture
    DDRT &= ~IC2; // PT2 is input
    TSCR = 0x80; // enable TCNT
    TMSK2 = 0x31; // 250ns clock
    TCTL4 = (TCTL4 & 0xCF) | 0x10; // rising
    First = TCNT; // first measurement will be wrong
    TFLG1 = OC3 + IC2; // Clear C3F, C2F
    TC3 = TCNT + 10000; // period larger than 2.5 ms
    TMSK1 |= OC3 + IC2; // Arm OC3, IC2
    asm(" cli "); }
```

(15) Part d) The interrupt service routines measure velocity.

```
#pragma interrupt_handler TIC2handler()
void TIC2handler(void) {
    Period = TC2 - First;
    v = 10000 / Period; // calculate velocity in cm/sec
    First = TC2; // Setup for next
    TFLG1 = OC3 + IC2; // Clear both C3F, C2F
    TC3 = TCNT + 10000; // period larger than 2.5 ms
```

```
#pragma interrupt_handler TOC3handler()
void TOC3handler(void) {
    v=0; // period larger than 2.5 ms
    First = TCNT; // next measurement will be wrong
    TFLG1=0C3+IC2; // Clear both C3F, C2F
    TC3=TCNT+10000; // period larger than 2.5 ms
}
```

(20) Question 2. In this problem you will write a C function that implements this digital filter.

(10) Part a) To convert to fixed-point, multiply by 256.

$$y(n) = (243x(n) - 487x(n-1) + 243x(n-2) + 486y(n-1) - 231y(n-2))/256$$

(10) Part b) Show the C code for the filter function. Needs 16 by 16 into 32 bit multiplies, 32 bit additions, and a 32 bit by 16 bit division. The inefficient way is to write it in C. The efficient way is to use assembly code (see the emacs instruction).

```
unsigned short filter(unsigned short data) {
    static short xn, xn1, xn2, yn, yn1, yn2; // MACQ
    xn2=xn1; xn1=xn; xn=data; yn2=yn1; yn1=yn; // shift data in MACQ
    yn=(243*(long)xn-487*(long)xn1+243*(long)xn2+486*(long)yn1-231*(long)yn2)/256;
    return yn; }
xn    ds    10    ; x(n),x(n-1),x(n-2),y(n-1),y(n-2)
yn    ds    2
acc   ds    4    ; temporary 32-bit result
cc    dc.w 243,-487,243,486,-231
; new data in Reg D
filter
    movw xn+2,xn+4 ; x(n-2)=x(n-1)
    movw xn,xn+2   ; x(n-1)=x(n)
    std  xn        ; x(n) = new data
    movw xn+6,xn+8 ; y(n-2)=y(n-1)
    movw yn,xn+6   ; y(n-1)=y(n)
    ldx  #xn       ; pointer to x(n),x(n-1),x(n-2),y(n-1),y(n-2)
    ldy  #cc       ; pointer to coefficients
    movw #0,acc    ; initially clear temporary result
    movw #0,acc+2
    ldaa #5        ; number of terms
loop  emacs acc ;acc=acc+{X}*{Y}
    leax 2,x
    leay 2,y
    dbne A,loop
    ldy  acc
    ldd  acc+2 ;Y:D=243x(n)-487x(n-1)+243x(n-2)+486y(n-1)-231y(n-2)
    ldx  #256
    edivs
    sty  yn
    ldd  yn      ; result in Reg D
    rts
```

(40) Question 3. In this problem you will interface a 74LS374 as an output port.

(5) Part a) We will use expanded narrow mode.

(10) Part b) Design the minimal cost positive logic address decoder

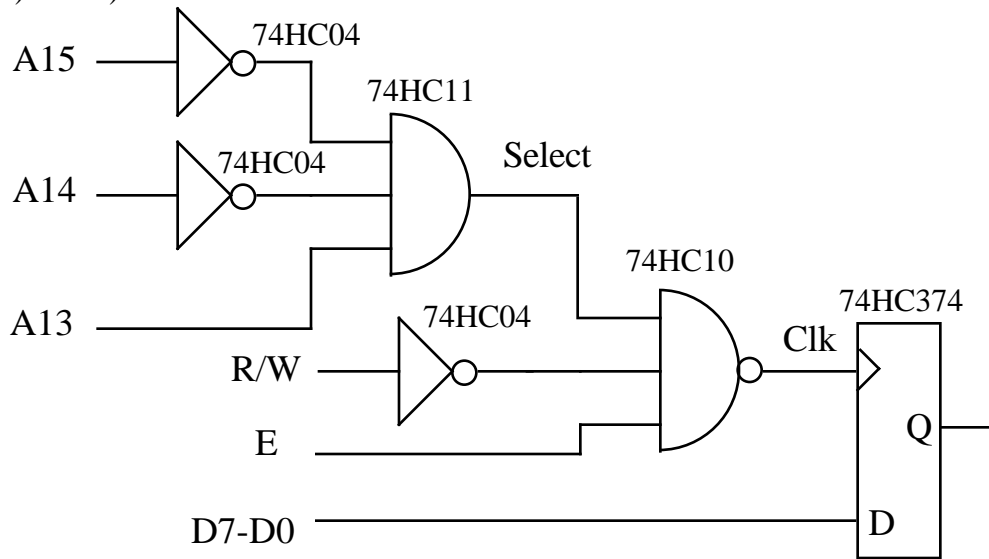
Ports	\$0000 to \$01FF	0000,000X,XXXX,XXXX
RAM	\$0800 to \$0BFF	0000,10XX,XXXX,XXXX
Your	\$3000 to \$3000	0011,0000,0000,0000
ROM	\$5000 to \$5FFF	0101,XXXX,XXXX,XXXX
ROM	\$6000 to \$7FFF	011X,XXXX,XXXX,XXXX
ROM	\$8000 to \$FFFF	1XXX,XXXX,XXXX,XXXX

Choose A15,A14,A13 (or A15 A14,A12), draw K-map

$$\text{Select} = \overline{A15} \cdot \overline{A14} \cdot A13$$

(5) Part c) Synchronized negative logic, so the rising edge of Clk clocks data into the 74LS374.

(5) Part d)



(5) Part e) The 74LS374 is so fast, no cycle stretches are needed. See the next part.

(10) Part f) Show the combined **write cycle** timing diagram.

