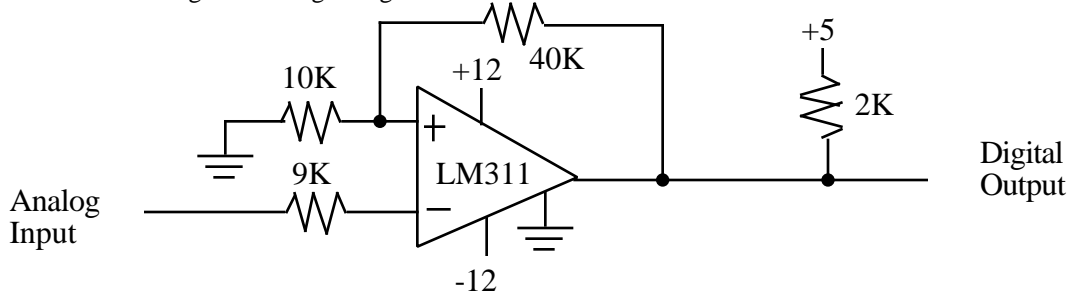


Jonathan W. Valvano March 13, 1998, 11:00 am-12:30pm

(20) Solution 1. Design an analog to digital converter.

The hysteresis is created by the positive feed back. Notice the voltage of the +terminal of the 311 is a function of the current output (+1v if output is high, 0 if the output is low.) The magnitude of the hysteresis is determined by $5v \cdot 10K / (10K + 40K)$

(30) Solution 2. The objective of this problem is to measure the phase angle.

Part a) The precision is 360 alternatives or about 9 bits.

Part b) The ritual which initializes the measurement software. Pulse is the time between the rise of PT2 and the rise of PT1. We will interrupt only on the rise of PT1, and measure Pulse using a 500 ns resolution. We could have used any time resolution better than 10 μ s because even at 125ns, the measurement will not overflow at 3.6ms

```
int phase; // range -179 to +180, units of degrees
```

```
void Ritual(void) {
    asm(" sei"); // make atomic
    TIOS&=0xF9; // clear bits 2, 1
    DDRT&=0xF9; // PT2, PT1 are input captures
    TSCR=0x90; // enable, fast clear
    TMSK2=0x32; // 500 ns clock
    TCTL4=(TCTL4&0xC3)|0x14; // IC2 Rise, IC1 Rise
    TMSK1|=0x02; // arm IC1, not IC2
    TFLG1=0x02; // clear C1F
    asm(" cli");}
```

Part c) Show the interrupt handler.

```
#pragma interrupt_handler TIC1handler()
void TIC1handler(void) { unsigned int Pulse; // uses fast clear acknowledgement
    Pulse=(TC1-TC2)/20; // time from rise of PT2 to rise of PT1 in 10 $\mu$ s units
    if (Pulse>180)
        Phase=Pulse-360; // convert to -179 to -1
    else
        Phase=Pulse;}
```

(50) Solution 3. The objective of this problem is to implement a real time data acquisition system.

Part a) Show the ritual

```
#define size 25;
int Xdata[size];
int yn; // FIR filter output
unsigned int n; // index to current
#define x(m) Xdata[(n+m)%size]
// x(n-m) is the sample m times ago
void ritual(void) { unsigned int m;
    asm(" sei"); // make atomic
    DDRA=0; DDRB=0xF0;
    TIOS|=0x20; // enable OC5
    TSCR=0x90; // enable, fast clear
    TMSK2=0x32; // 500 ns clock
    TMSK1|=0x20; // Arm OC5
    for(m=0; m<size; m++) Xdata[m]=0;
    n=size-1;
    TFLG1=0x20; // clear C5F
    TC5=TCNT+10000; // First one in 5 ms
    asm(" cli"); }
```

Part b) Show the OC5 interrupt handler

```
#pragma interrupt_handler TOC5handler()
void TOC5handler(void) { int data;
    data=((PORTB&0x0F)<<8) | PORTA;
    // raw 12 bit sample
    if(data&0x0800)
        data=data|0xF000; // sign extend
    if(n)
        n--;
    else
        n=size-1;
    x(n)=data; // save in MACQ
    yn=(x(n)+x((n+24)%25))>>1; // FIR
    TC5=TC5+10000; // every 5 ms
    PORTB |= 0x10;
    PORTB &= 0xEF; } // start next A/D
```