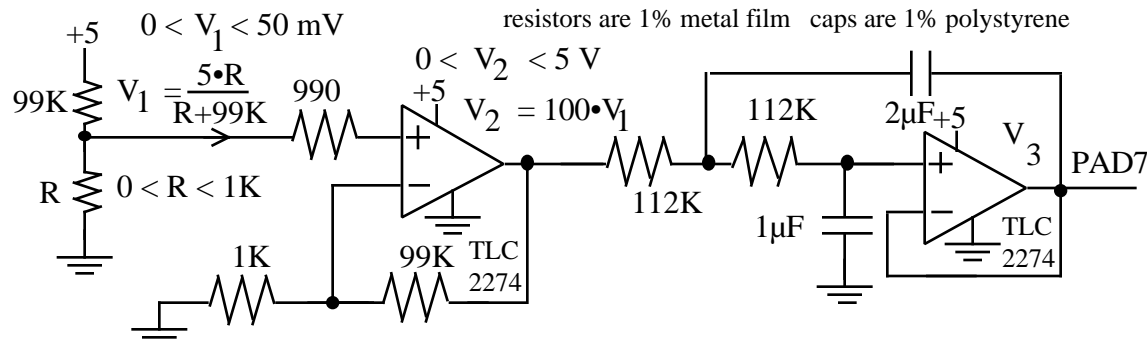


Jonathan W. Valvano

(20) Question 1. Design a wind direction measurement instrument using the 8-bit A/D converter.**(10) Part a)** Show the analog interface between the transducer and the A/D port channel 7.**(10) Part b)** Function measures the wind direction and returns a result with units of degrees.

void Ritual(void){

ATDCTL2 = 0x80; } // Activate A/D

#define SCF 0x80

unsigned int WindDirection(void){ unsigned int direction;

ATDCTL5=7; // Start A/D

while ((ATDSTAT & SCF) == 0){};

direction=(ADROH*360)>>8;

return(direction); }

(20) Question 2. Design a wind direction measurement instrument using the input capture technique.

unsigned int WindDirection(void) { unsigned int direction, First;

TCTL3=(TCTL3&0x3F)|0x40; // Rising edge on IC7

TFLG1=0x80; // clear C7F

while(TFLG1&0x80==0){}; // Wait for rise

First=TC7; // TCNT at first rising edge

TFLG1=0x80; // clear C7F

while(TFLG1&0x80==0){}; // Wait for second rising edge

direction=TC7-First-6098; // 0 to 3049

direction=(2*direction)/17; // 0 to 359

return(direction); }

void Ritual(void){

TIOS&=0x7F; // clear bit 7

DDRT&=0x7F; // PT7 is input capture

TSCR=0x80; // enable, no fast clear

TMSK2=0x32; // 500 ns clock

TMSK1=0x00; // no interrupts

(15) Question 3. Consider the 8K by 8 bit static RAM interface using the MCM60L64.**(10) Part a)** Develop the equations that specify the maximum allowable t_{AVQV} .

$$\begin{aligned} \text{Read Data Available} &= (\text{later}(\text{AdV} + t_{AVQV}, \overline{\text{E1}} + t_{E1LQV}), \\ &\quad \text{earlier}(\text{AdN} + t_{AXQX}, \overline{\text{E1}} + t_{E1HQZ})) \\ &= (\text{later}(60 + t_{AVQV}, 60 + t_{E1LQV}), \text{earlier}(1 + 20 + 20, 1 + 10)) \end{aligned}$$

Recall that

$$\text{RDR} = \text{Read Data Required} = (1 - 11, 1 + 12) = (95, 125)$$

Address $60 + t_{AVQV}$ 95 and 165 125E1 $60 + t_{E1LQV}$ 95 and 135 125

Thus both

$$t_{AVQV} \leq 35 \quad \text{and} \quad t_{E1LQV} \leq 35$$

(5) Part b) Develop the equations that specify the maximum allowable t_{DVWH} .

$$\text{WDA} = \text{Write Data Available} = (2 + 13, 1 + 14) = (106, 145)$$

Since we have chosen to synchronize $\overline{\text{E1}}$, that occurs at 135,

$$\text{Write Data Required} = (\overline{\text{E1}} - t_{DVWH}, \overline{\text{E1}}) = (135 - t_{DVWH}, 135)$$

For write data available to overlap write data required,

$$106 \leq 135 - t_{DVWH}$$

so $t_{DVWH} \leq 29\text{ns}$

(5) **Question 4.** In single chip mode ports A,B,C, D and E are regular I/O ports.

In expanded narrow mode, ports A,B contain the 16 bit address, port C has the 8 bit data, and port E has bus signals.

In expanded wide mode, ports A,B contain the 16 bit address, ports C,D has the 16 bit data, and port E has bus signals.

(10) **Question 5.** If the controller is executed too infrequently, then the error will be very large because the controller can not respond quickly enough to changes in the physical plant. If the controller is executed too frequently, then the system will revert to bang-bang.

(30) **Question 6.** The objective of this problem is to develop a message passing facility.

<pre> Part a) Computer 1 int mode; // 0=stop, 1=wait rise, 2=wait fall void Ritual(void){ asm(" sei"); // make atomic TIOS &= ~0x40; // PT6 input capture DDRT &= ~0x40; // PT6 is input DDRT = 0x80; // PT7 is output TSCR = 0x80; // enable TCNT TMSK2 = 0x32; // 500ns clock PORTT &= ~0x80; // PT7=0 mode=0; DDRH=0xFF; // PortH are outputs InitFifo(); // clears fifo asm(" cli"); } void Start(unsigned char data){ asm(" sei"); // make atomic PORTH=data; TCTL3 =(TCTL3&0xCF) 0x10; // rising of IC6 TMSK1 = 0x40; // Arm IC6 TFLG1 = 0x40; // initially clear PORTT = 0x80; // PT7=1 mode=1; // means waiting for rise asm(" cli"); } void PutMsg(unsigned char data){ if(mode) PutFifo(data); // previous in progress else Start(data); // was idle, so start it up } #pragma interrupt_handler IC6Han() void IC6Han(void){ unsigned char data; TFLG1=0x40; // acknowledge if(mode==1){ TCTL3 =(TCTL3&0xCF) 0x20; // falling of IC6 PORTT &= ~0x80; // PT7=0 mode=2;} else { if(GetFifo(&data)) // returns 0 if empty Start(data); // continue else mode=0; // no more } } </pre>	<pre> Part b) Computer 2 int mode; // 0=wait for rise, 1=wait for fall void Ritual(void){ asm(" sei"); // make atomic TIOS &= ~0x80; // PT7 input capture DDRT &= ~0x80; // PT7 is input DDRT = 0x40; // PT6 is output TSCR = 0x80; // enable TCNT TMSK2 = 0x32; // 500ns clock PORTT &= ~0x40; // PT6=0 TCTL3 =(TCTL3&0x3F) 0x40; // rising of IC7 TMSK1 = 0x80; // Arm IC7 TFLG1 = 0x80; // initially clear mode=0; // means waiting for rise DDRH=0x00; // PortH are inputs InitFifo(); // clears fifo asm(" cli"); } unsigned char GetMsg(void){ unsigned char data; while(!GetFifo(data)){}; return(data); } #pragma interrupt_handler IC7Han() void IC7Han(void){ TFLG1=0x80; // acknowledge if(mode==0){ // mode=0 means rising edge PutFifo(PORTH); // read and save TCTL3=(TCTL3&0x3F) 0x80; // falling of IC7 PORTT = 0x40; // PT6=1 mode=1;} else { TCTL3 =(TCTL3&0x3F) 0x40; // rising of IC7 PORTT &= ~0x40; // PT6=0 mode=0; // look for more } } </pre>
---	--