Jonathan W. Valvano April 18, 2001, 9:00am-9:50am

(30) Question 1. Battery-backed SRAM interface.

Part a)

Read Data Available = (later (AdV+t_{AVQV}, ↓ $\overline{E1}$ +t_{E1LQV}) , earlier (AdN+t_{AXQX}, ↑ $\overline{E1}$ +t_{E1HQZ}))

= (60+t_{AVQV}, 510)

Read Data Required = (t_1 - 30, t_1) = (470, 500)

so 60+t_{AVQV} ≤ 470 or t_{AVQV} ≤ 410 ns

Part b)

Write Data Available = (t_2 + t_{13}, t_1 + t_{14}) = (60 + 46, 500 + 20) = (106,520)

Write Data Required = (↑ $\overline{E1}$ - t_{DVWH} , ↑ $\overline{E1}$ + t_{WHDX}) = (510 - t_{DVWH} , 510)

so 106 ≤ 510 - t_{DVWH} or t_{DVWH} ≤ 404 ns

(35) Question 2. Starting with the original Lab 17 files, you will develop a **Sleep** OS primitive.

Part a) Show the implementation of the `OS_Sleep` function.

```
void OS_Sleep(unsigned short delay){
  RunPt->SleepCounter = delay; // time in ms to sleep
  TC3 = TCNT+15;               // suspend this thread
}
```

Part b) The new modified `threadSwitch`.

```
void threadSwitch(void){ // do most of the work here
  RunPt = RunPt->Next;
  while(RunPt->SleepCounter){ // find one with counter equal to zero
    RunPt = RunPt->Next;
  }
  PORTJ = RunPt->Id;     /* PortJ shows which thread is running */
}
```

Part c) Once every ms, decrement the `SleepCounter` for all threads with a nonzero `SleepCounter`.

```
#pragma interrupt_handler OC0Handler
void OC0Handler(void){ unsigned int thread;
  TC0 = TC0 + 8000;   // interrupt every 1 ms
  TFLG = 0x01;        // acknowledge interrupt by clearing C0F
  for(thread=0; thread<NumThread; thread++){
    if(TCB[thread].SleepCounter){
       TCB[thread].SleepCounter--;  // awake when 0
    }
  }
}
```

(35) Question 3.

Part a) The DAC resolution, ΔV=range/precision=10/4096=2.44mV

Part b) $0.00244 \geq 2^n$ or $\log_2(0.00244) \geq n$ or $-8.678 \geq n$ so choose n = -9.

Part c) $-1.000 = I \bullet 2^{-9}$ so I = -512.

Part d) `dacData = 2048*binaryData/2560 + 2048` , which can be simplified to

`dacData = 4*binaryData/5 + 2048`

Part e) The following is essentially program 7.20 found on page 407. Add the C implementation of your equation.

```
void DACout(short binaryData){
unsigned short dacData;
unsigned char dummy;
  dacData = (4*binaryData+10242)/5;  // extra +2 for rounding
  SP0DR = 0x00FF&(dacData>>8); // msbyte
  while((SP0SR&SPIF)==0);      // gadfly wait
  dummy = SP0DR;               // clear SPIF
  SP0DR = 0x00FF&dacData;      // lsbyte
  while((SP0SR&SPIF)==0);      // gadfly wait
  dummy = SP0DR;               // clear SPIF
  PORTS &= ~0x80;              // PS7=LD=0
  PORTS |= 0x80; }             // PS7=LD=1
```