Jonathan W. Valvano February 22, 2008, 1:00pm-1:50pm.

**(5) Question 1.** I am looking to see if there is a correlation between EE319K instructor and EE345L grades. So far, the two seem uncorrelated (which of course is good).

**(5) Question 2.** `PTT ^= 0x01;`

Part a) This debugging monitor is *minimally intrusive*

Part b) Because it takes only 7*250ns to execute.

**(5) Question 3.** The issue is the problem says "to store the pressure data" not "to output the results". Internally, we must choose a resolution better than desired resolution. We use binary fixed-point when we need to make calculations, and we use decimal fixed-point when we need to input/output.

E) 16-bit unsigned binary fixed-point system with a resolution of 1/16

Resolution is better than 0.1 psi and the range is 0 to 65535/16= 4095.9375

**(15) Question 4.** Promote to 32 bits to handle overflow. Divide last to handle dropout.

$$\mathbf{m = (12*n^2 + 3400*n + 7700)/1000}$$

For unsigned calculations you can add half the divisor, before the division to implement rounding.

```
unsigned long ln,lm;
  ln = (unsigned long)n;   // promote to 32 bits
  lm = (12*ln*ln + 3400*ln + 7700+500)/1000;
  m = (unsigned short)lm;   // demote back to 16 bits
```

**(10) Question 5.** The largest component of latency is running with interrupts disabled. The delay from output idle is the sum of three parts (the maximum time running with I=1, time to switch from foreground to background, and time to execute the ISR).

**g) Run with the I bit equal to 1 for no more than 0.5ms at a time.**

One by one, consider why the other answers are wrong.

a) Activate the PLL so the software runs at 24 MHz.

The PLL makes it run faster but it does not guarantee real-time behavior

b) Use an output compare interrupt with a periodic rate of 1ms.

h) Switch over to periodic polling synchronization (same as b)

Having an output compare interrupt, which itself could be delayed, has a slower response than having the output idle flag interrupt directly.

c) Set the HPRIO register so this output device is highest priority.

Setting the HPRIO register will help if two interrupts are simultaneously requested, allowing the output device to go first. But if a low priority interrupt occurs first and this low priority interrupt service routine takes more than 1ms, then the output device timing will be lost, and HPRIO is no help.

d) Execute `sei` in the ISR of the output device so other interrupts can't suspend this task.

This is nonsense (and has no effect), because the I bit is already set.

e) Execute `cli` in the ISR of the output device so other interrupts can suspend this task.

This is nonsense, because it would allow lower priority interrupts to suspend this high priority task.

f) Run with the I bit equal to 0 for no more than 0.5ms at a time.

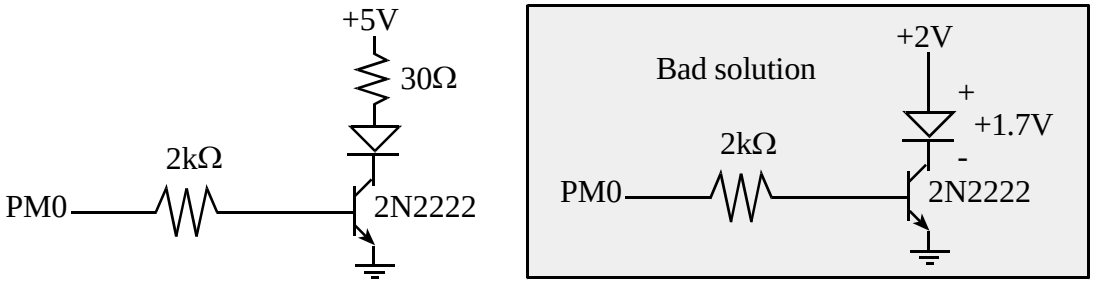
Basically, you want to run with I=0 all the time (interrupt enabled)

i) Switch over to busy-wait synchronization

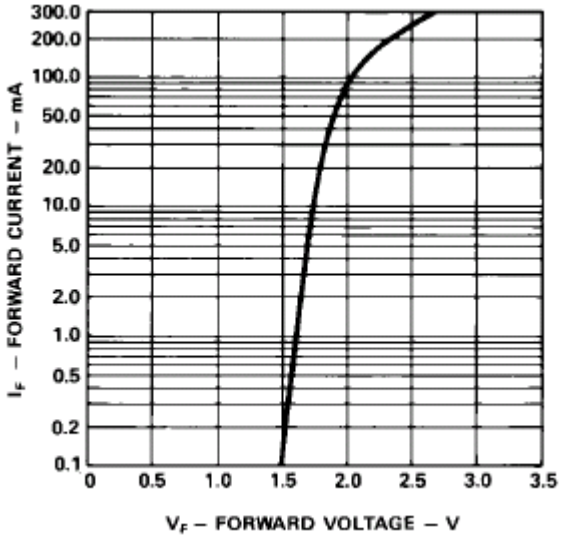
Busy-wait can improve bandwidth in simple systems with one I/O task. In a complex system with three I/O tasks, one can not design it in such a way that the latency is bounded.

**(5) Question 6.** The variable i is embedded in the machine code itself. The **`i`**=0 case is implied when PTT is copied to PTM. The **`i`**=1 case is implemented by the **`incb`** instruction. The **`i`**=2 case is implemented by the **`#2`** in the **`addb #2`** instruction. The **`i`**=3 case is implemented by the **`#3`** in the **`addb #3`** instruction. In this code the variable **`i`** is never in Reg B nor on the stack. To get this Metrowerks code, I activated the loop-unrolling option in the optimization configuration.

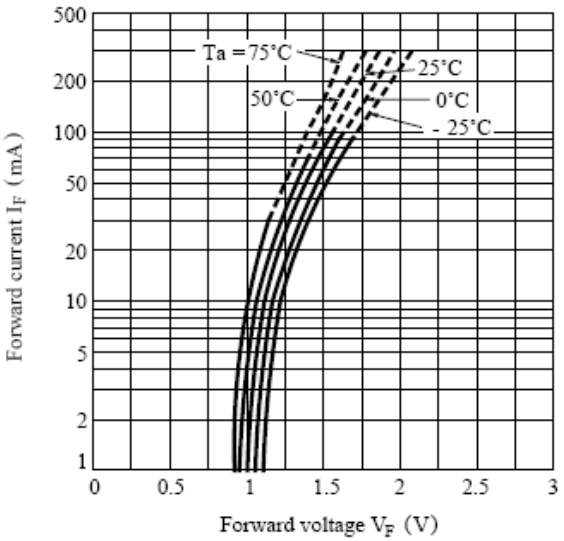
**(10) Question 7.** 100 mA will require the 2N2222 (because it can handle up to 500 mA of $I_{CE}$). The $V_{CE}$ on voltage of the 2N2222 is 0.3V. The resistor value is calculated as R = (5-1.7-0.3)/0.1 = 3V/0.1A = 30Ω. Because the current gain is 100 ($h_{fe}$) the base current needs to be 100mA/100 = 1mA. The $I_{OH}$ of the 9S12 can supply this 1mA ($I_{OH}$ can be up to 10 mA). Because the $V_{OH}$ of the 9S12 is 4.2V (or greater) and the $V_{BE}$ if the 2N2222 is 0.6V (or less), the resistor from the 9S12 to the 2N2222 base must be less than (4.2-0.6V)/1mA = 3.6/0.001 = 3.6 kΩ.




You might have been tempted to interface a LED using a voltage-controlled scheme, like the "Bad solution". The trouble with the "Bad solution" is if the +2V supply increases by 10% or if the VCE of the 2N2222 decreases by 0.2V, then the voltage on the diode goes from 1.7 to 1.9V, and the current could increase by a factor of 5 or more.




The 1 mA LED used in Lab 8,11,12.



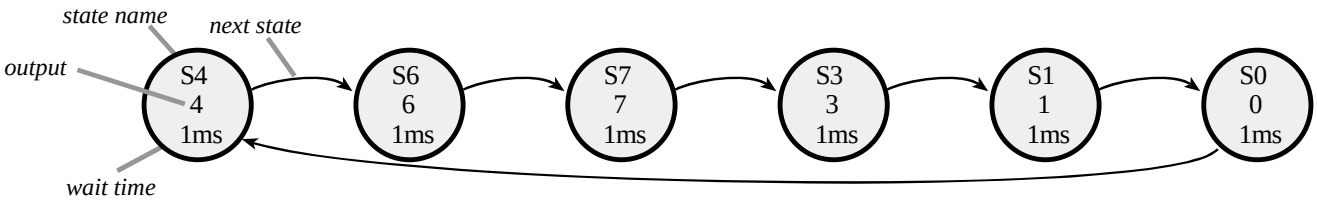

100 mA IR LED

With the current-controlled circuit on the left, a 10% error in +5V, 30Ω, or $V_{CE}$ causes a 10% error in the LED current, because I = (5-1.7- $V_{CE}$)/30Ω.

**(5) Question 9.** Maximum voltage recognized as a logic low is $V_{IL}$ = 1.75 V.

**(35) Question 10.**

Part a) Output sequence is 4, 6, 7, 3, 1, 0,…




Part b) Change from four pointers to one pointer.

```
const struct State{
  unsigned char Out;
  unsigned short Time;
  const struct State *Next;
};
typedef const struct State StateType;
typedef StateType *StatePtr;
#define S4 &fsm[0]
#define S6 &fsm[1]
#define S7 &fsm[2]
#define S3 &fsm[3]
#define S1 &fsm[4]
#define S0 &fsm[5]
StateType fsm[6]={
      {0x04,1000,S6},
      {0x06,1000,S7},
      {0x07,1000,S3},
      {0x03,1000,S1},
      {0x01,1000,S0},
      {0x00,1000,S4}
};
```

Part c) Show the main program that initializes the system and then executes a do-nothing loop.

```
StatePtr Pt; // pointer to current state
void main(void){
  DDRT |= 0x07; // PT2,PT1,PT0 outputs
  TIOS |= 0x02; // activate TC1 as output compare
  TSCR1 = 0x80; // Enable TCNT 4MHz in run mode
  TSCR2 = 0x02; // divide by 4 TCNT prescale, 1us
  TIE |= 0x02;  // arm OC1
  TC1 = TCNT+50; // first interrupt soon
  Pt = S4;
  asm cli
  for(;;);
}
```

Part d) You could have replaced **`Pt->Time`** with a constant to create 1ms periodic interrupt

```
interrupt 9 void TC1handler(void){
  TFLG1 = 0x02;       // acknowledge OC1
  PTT = Pt->Out;      // perform output for this state
  TC1 = TC1+Pt->Time; // time for this state
  Pt = Pt->Next;      // next state
}
```