Jonathan W. Valvano October 19, 2007, 1:00pm-1:50pm.

(5) Question 1. Why is it highly intrusive to add this code to an interrupt service routine?

1) because the debugging code takes a long time to execute compared to the time to run the ISR itself

2) because the presence of the debugging code itself affects the measurement

(10) Question 2. Add debugging instruments to determine which ISR is executed first.

```
interrupt 16 void TOFhandler(void){
  if((bRTI==0)&&(bTOF==0)) bTOF = 1;
  TFLG2 = 0x80;      // acknowledge TOF
  Stuff1();
}
interrupt 7 void RTIhandler(void){
  if((bRTI==0)&&(bTOF==0)) bRTI = 1;
  CRGFLG = 0x80;     // acknowledge RTIF
  Stuff2();
}
```

or

```
interrupt 16 void TOFhandler(void){
  bTOF = ~bRTI;
  TFLG2 = 0x80;      // acknowledge TOF
  Stuff1();
}
interrupt 7 void RTIhandler(void){
  bRTI = ~bTOF;
  CRGFLG = 0x80;     // acknowledge RTIF
  Stuff2();
}
```

(10) Question 3. Yes, there is a critical section, because the main program has a nonatomic read-modify-write to a global variable (PTT). If the interrupt occurs after the main program reads **PTT**, and before the main program writes **PTT**, the output becomes invalid

```
          ldaa $0240
..... trouble if interrupt occurs here.....
          eora #$01
..... trouble if interrupt occurs here.....
          staa $0240
```

Two ways to fix it

1) Move one of the debugging pins to another port like **PTM**

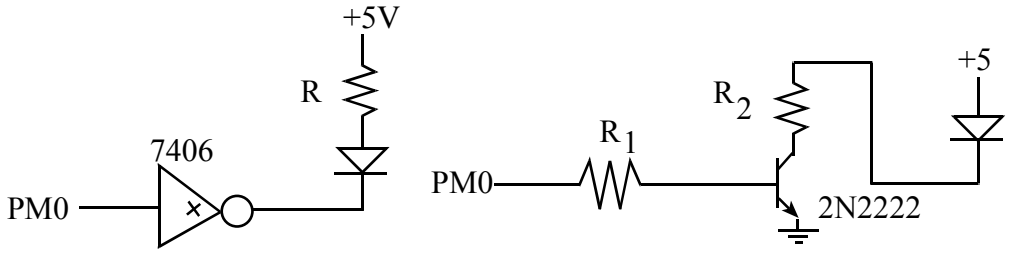
2) Make the read-modify-write atomic

```
asm sei
  PTT ^= 0x01; // toggle PT0
asm cli
```

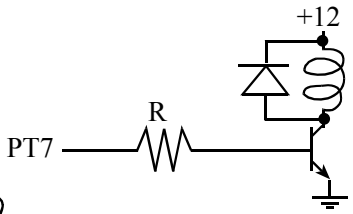
(10) Question 4. *Interface latency* is the time from when **RDRF** is set (by the hardware) to signify new data is available until the time the software reads **SCIDRL**. The *device latency* is the time from reading the **SCIDRL** (the time the software is asking for another input) until the time **RDRF** is set (by the hardware) signifying the I/O request is satisfied.

(5) Question 5. The range of 8-bit signed numbers is -128 to +127, thus **-12.8V to +12.7V**

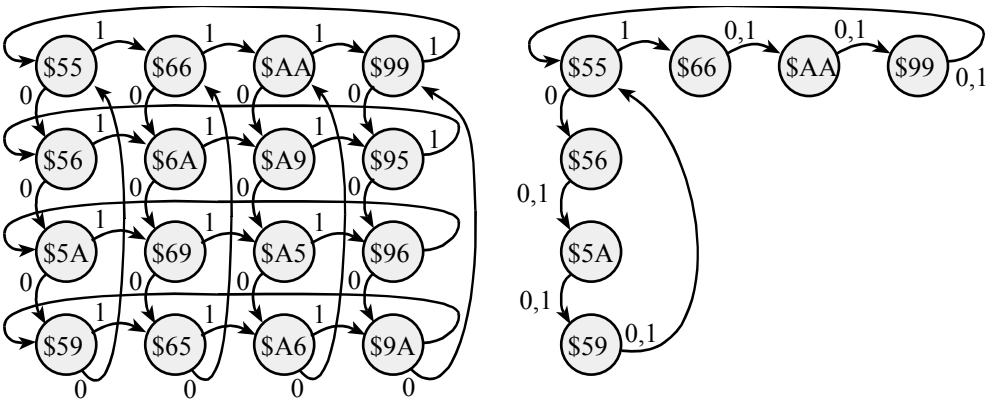
(10) Question 6. 25 mA will require either the 2N2222 or the 7406. The output low voltage of the 7406 is 0.5V. The resistor value is calculated as R = (5-2-0.5)/0.025 = 2.5V/0.025A = 100Ω.



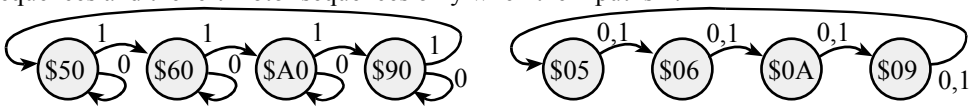
(10) Question 7. A 12V power is connected to one side of the solenoid, and the other side is switched to ground. To handle the 200 mA, the 2N2222 can be used (because it can handle up to 500 mA of I_{CE}). Because the current gain is 100 (h_{fe}) the base current needs to be 200mA/100 = 2mA. The I_{OH} of the 9S12 can supply this 2mA (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)/2mA = 3.6/0.002 = 1.8 kΩ. The 1N914 diode provides protection against back EMF.



(20) Question 8. For each motor there are 4 valid outputs. Therefore, for the two motors, there are 16 valid outputs. When the input is 1, both motors sequence 5,6,10,9. When the input is 0, just the right motor changes. If you are willing to delay up to 40ms, the FSM on the right could be used.



A third alternative is to implement two totally separate machines, and have the controller combine outputs. Again, the right motor always sequences and the left motor sequences only when the input is 1.



(20) Question 9. Use output compare 5 interrupts to generate the following periodic output on PT5.
Part a) Show the ritual that is executed once.

```
void OC5_Init(void){
  DDRT |= 0x20;     // PT5 output
  PTT &= ~0x20;     // PT5 initially low
  TIOS |= 0x20;     // activate TC5 as output compare
  TSCR1 = 0x80;     // Enable TCNT 1 MHz in run mode
  TSCR2 = 0x02;     // divide by 4 TCNT prescale, 1us
  TIE |= 0x20;      // arm OC5
  TC5 = TCNT+5000; // first interrupt in 5ms
  asm cli
}
```

Part b) Show the output compare interrupt service routine that outputs to PT5.

```
interrupt 13 void TC5handler(void){
  TFLG1 = 0x20;      // acknowledge OC5
  if(PTT&0x20){
    PTT &= ~0x20;   // PT5 used to be high, now low
    TC5 = TC5+5000; // low for the next 5ms
  } else{
    PTT |= 0x20;    // PT5 used to be low, now high
    TC5 = TC5+2000; // high for the next 2ms
  }
```

```
}
```