Jonathan W. Valvano October 9, 2009, 2:00pm-2:50pm.

(5) Question 1. *Latency* is the time between when main power is lost (rise of input) to the time the software disconnects the main power and connects to the battery backup (change output from 01 to 10). To be *real time*, this delay must always be less than 10 ms.

(10) Question 2. The nonreentrant code occurs because of the read-modify-write to **`RxPutPt`**. The critical section is between these two lines (i.e., between the read and the write of the read-modify-write sequence)

```
  tempPt = RxPutPt;
```
and
```
  RxPutPt = tempPt;      /* Success, so update pointer */
```
For data to be lost, there must be two or more threads that call **`RxFifo_Put`**. At least one of these threads must be running with I=0 (interrupts enabled), and at least one thread must be invoked by an interrupt service routine. In this example assume thread1 runs with I=0 and thread2 is an ISR

1) thread1 starts **`RxFifo_Put`** and executes past the line **`tempPt = RxPutPt;`**
2) thread2 is invoked by an interrupt and also calls **`RxFifo_Put`**
3) since I=1 at this point, thread2 finishes **`RxFifo_Put`** putting its data into the Fifo
4) thread1 resumes executing **`RxFifo_Put`**
5) because **`tempPt`** is local (not the same as thread2), the line **`RxPutPt = tempPt;`** will cause one piece of data to be lost.

This is a classic read-modify-write to a shared global critical section.

(10) Question 3. For the output parameters we look in the XBee data sheet

I_{OL} = 2mA, I_{OH} = 2mA, V_{OL} = 0.5V, $V_{OH} = V_{CC}$-0.5V = 2.8V

For the input parameters we look in the 9S12 data sheet

I_{IL} = 1μA, I_{IH} = 1μA, V_{IL} = 1.75V, V_{IH} = 3.25 V

The voltage requirements are

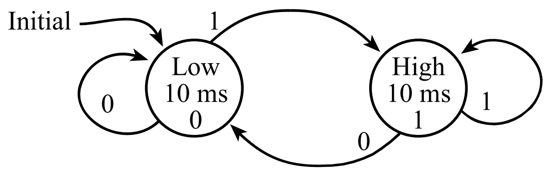
$V_{OL} \le V_{IL}$ and $V_{OH} \ge V_{IH}$

The current requirements are

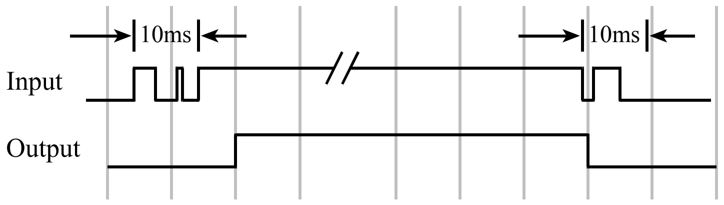
$I_{OL} \ge I_{IL}$ and $I_{OH} \ge I_{IH}$

It does not work because $V_{OH} < V_{IH}$ or 2.8 < 3.25

(5) Question 4. Value = integer*resolution= -384/256 = -1.5



(20) Question 5. The basic idea is to read the switch at a rate slower than every 10 ms but faster than every 100 ms. The time delay could be any number from 10 to 100 ms, but 10 ms minimizes the delay from input change to output change. There will be a variable delay from 0 to 20 ms between the change in the switch and the change in the output, however one switch touch will always cause one rising edge of the output, and one switch release will always cause one falling edge of the output.



(5) Question 6. Consider a stepper motor.

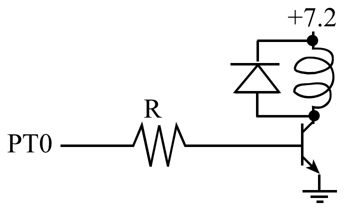
Part a) If θ is the angle of the motor, dθ/dt is the rotational velocity in radians/sec, $d^2\theta/dt^2$ is the rotational acceleration in radians/sec^2, and $\mathbf{d^3\theta/dt^3}$ is the **rotational jerk** in radians/sec^3.

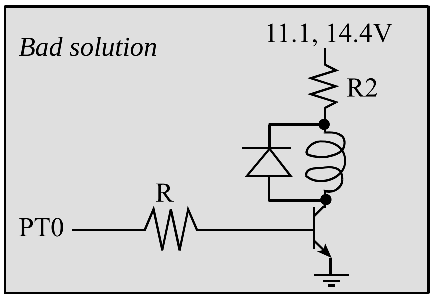
Part b) The units of jerk are **radians/sec^3**

(15) Question 7. 200 mA will require the 2N2222 (because it can handle up to 500 mA of I_{CE}). We could have used any NPN with I_{CE} > 200mA, e.g., TIP120, IRF540. The V_{CE} on voltage of the 2N2222 is 0.3V. 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.

Part a)

I suggest making R much less than 1.8 kΩ (e.g., 1 kΩ) because it will force the NPN into saturation, independent of the V_{OH} of the 9S12, the V_{BE} of the 2N2222, the h_{fe} of the 2N2222, and the resistance of the coil. Therefore, when the digital output is high, the voltage across the relay will be 6.9V. You might have been tempted to use a higher voltage supply, like the "Bad solution" and use a series resistor (R2) to drop the voltage down to 6 to 8V. There are two fundamental problems with the "Bad solution". First, the solution wastes power, the power delivered into R2 is lost as heat. Second, the resistance of the coil is a function of the mechanical load on the electromagnet. The coil resistance can not be assumed to be constant.





Part b) V = L dI/dt

(30) Question 8. This first solution implements explicit software outputs of PT1

```
void Question8_Init(void){
  TSCR1 = 0x80;    // Enable TCNT 4 MHz in run mode
  TSCR2 = 0x03;    // divide by 8 TCNT prescale, 1us
  TIOS |= 0x02;    // activate TC1 as output compare
  TIE |= 0x02;     // arm OC1
  DDRT |= 0x02;    // PT1 output
  PTT_PTT1 = 0;    // initial output
  TC1 = TCNT+2000; // low for 2 ms
  TFLG1 = 0x02;    // clear flag
  asm cli
}
interrupt 9 void TC1handler(void){
  TFLG1 = 0x02;     // acknowledge OC1
  if(PTT_PTT1){
    PTT_PTT1 = 0;   // was high, now low
    TC1 = TC1+2000; // low for 2 ms
  } else{
    PTT_PTT1 = 1;   // was low, now high
    TC1 = TC1+3000; // high for 3 ms
  }
}
```

This second solution implements hardware outputs of PT1

```
void Question8_Init(void){
  TSCR1 = 0x80;    // Enable TCNT 4 MHz in run mode
  TSCR2 = 0x03;    // divide by 8 TCNT prescale, 1us
  TIOS |= 0x02;    // activate TC1 as output compare
  TIE |= 0x02;     // arm OC1
  TCTL2 = (TCTL2&0xF3)|0x04; // toggle PT1 on OC1
  DDRT |= 0x02;    // PT1 output
  PTT_PTT1 = 0;    // initial output
  TC1 = TCNT+2000; // low for 2 ms
  TFLG1 = 0x02;    // clear flag
  asm cli
}
interrupt 9 void TC1handler(void){
  TFLG1 = 0x02;     // acknowledge OC1
  if(PTT_PTT1){
    TC1 = TC1+3000; // high for 3 ms
  } else{
    TC1 = TC1+2000; // low for 2 ms
  }
}
```