Jonathan W. Valvano

(4) Question 1. Write a C function that outputs (transmits) one byte using the SCI port.

```c
void SCI_OutChar(unsigned char data){
    while((SCISR1&0x80)==0){};  // wait for TDRE to be 1
    SCIDRL = data;              // start output
}
```

(4) Question 2. Give an example C code that has a dropout error.

```c
Result = 1234*(Input/1000);  // multiply by 1.234
```

(4) Question 3. Give the four inequalities that must be true in order for the interface to operate properly.

<table>
<thead>
<tr>
<th>( V_{OH} \geq V_{IH} )</th>
<th>( I_{OH} \geq I_{IH} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{OL} \leq V_{IL} )</td>
<td>( I_{OL} \geq I_{IL} )</td>
</tr>
</tbody>
</table>

(4) Question 4. Write a C function that outputs (transmits) one byte using the SPI port.

```c
void SPI_OutChar(unsigned char data){
    unsigned char dummy;
    while((SPISR&0x20)==0){};  // wait for SPTEF to be 1
    SPIDR = data;              // start output
    while((SPISR&0x80)==0){};  // wait for SPIF to be 1
    dummy = SPIDR;             // clear SPIF
}
```

(4) Question 5. Use long type (signed 32 bit) because we need to store -100000 to +100000.

(4) Question 6. Give a mathematical equation, defining sampling jitter, \( \delta t \) in terms of \( f_s \), \( t_{i-1} \) and \( t_i \).

\[
\delta t_i = (t_i - t_{i-1}) - \frac{1}{f_s}
\]

To be real time, we must be able to place an upper bound, \( k \), on the time-jitter.

\[-k \leq \delta t_i \leq +k\]

for all \( i \)

(8) Question 7. Review your EE411 before going on the interview trail.

\[
R \times C = 1 \mu s. \text{ From 0 to 100ns, } V_{out} = 5 - 5e^{-t/RC} = 5 - 5e^{-t/1000ns}
\]

At 100ns, \( V_{out} = 5 - 5e^{-100ns/1000ns} = 0.48V \)

![Graph](image)

(4) Question 8. This one captures the data in the least amount of time

A) \( \text{if (n<100)} \{ \text{BufT[n]= TCNT; BufX[n]=x; n++;} \}

(4) Question 9. Rationalizations an engineer might have for being unethical.

1) Too expensive to do the right thing; lawsuits and jail time are more expensive.
2) Everyone else is doing it; no, everyone else is not doing it.
3) Won’t get caught; do the right thing because it is the right thing.
4) Didn’t know it was wrong; you are responsible, it is your job to know.
5) My boss told me to do it; hit men for the mob go to jail too.
6) Arrogance, I am right, so I could not have made a mistake; mistakes happen.
7) It all averages out; you can not balance a little bit of evil with a lot of good. Right and wrong are absolute. Search the morality associated with the expression “color of money”.

(10) **Question 10.** The controller sequence repeats these four operations: output, wait, input, next.

(10) **Question 11.** PT0 controls the On/Off and PT1 controls the direction of rotation.

(15) **Question 12.** Address decoder at A15 must be 1, activate second half of the cycle E must be 1, activate for read cycle R/W must be 1. Negative logic output. Part a)

Part b) **Read Data Available** = [500+20+120, 1000+10+10] = [640,1020]
(10) Question 13. You will use a Sharp GP2Y0A21YK0F infrared object detector to measure distance. These sensors are $10 on digikey.com

Part a) Map the 0.5 to 3V range of the transducer into the 0 to +5V range of the ADC.
\[ V_2 = 2*V_1 - 1 \]
0.5 → 0, 3 → 5

Part b) \[ V_{ref} = 2.50V, V_g = 0V \]
\[ V_2 = 2*V_1 - 0.4*V_{ref} \]
offset terms generated with voltage reference
\[ V_2 = 2*V_1 - 0.4*V_{ref} - 0.6*V_g \]
sum of gains equals 1

Common multiple of 2, 0.4, 0.6 = 24 kΩ
\[ V_2 \text{ Gain of 2 is } 24 \text{ kΩ} / 12 \text{ kΩ} \]
\[ V_{ref} \text{ Gain of 0.4 is } 24 \text{ kΩ} / 60 \text{ kΩ} \]
\[ V_g \text{ Gain of 0.6 is } 24 \text{ kΩ} / 40 \text{ kΩ} \]

(15) Question 14. Add the other 7 switch patterns to the table, then sort by PTM value. This joystick is $12 (part number JS-5) on http://www.allelectronics.com.

<table>
<thead>
<tr>
<th>Joystick position</th>
<th>Switch U</th>
<th>Switch R</th>
<th>Switch D</th>
<th>Switch L</th>
<th>PTM</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Left</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Down</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Down and Left</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Right</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>255</td>
</tr>
<tr>
<td>Down and Right</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>255</td>
</tr>
<tr>
<td>Up</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Up and Left</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>255</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>255</td>
</tr>
<tr>
<td>Up and Right</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>255</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>255</td>
</tr>
<tr>
<td>Bad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>255</td>
</tr>
</tbody>
</table>
Part a) Simple positive logic switch interfaces

![Logic Switch Diagram]

Part b) Show the initialization code for the system, enable and arm output compare 7. Initialize Port M.

```c
void JoyStick_Init(void){
  DDRM &= ~0x0F; // PM3-PM0 are inputs from joystick
  Fifo_Init();
  TIOS |= 0x80; // channel 7 is output compare
  TSCR1 = 0x80; // activate timer
  TSCR2 = 0x05; // TCNT at 4MHz/32 = 125 kHz
  TIE |= 0x80; // arm channel 5
  TC7 = TCNT+50; // interrupt right away
  asm cli // enable
}
```

Part c) Show the output compare interrupt 7 ISR. Use the last entry in table to map PTM into Code.

```c
const unsigned char CodeTable[16]= {
  0,7,5,6,3,255,4,255,1,8,255,255,2,255,255,255};

interrupt 15 void TC7handler(void){ // executes at 10 Hz, 100ms
  unsigned char in; // 0 to 15 input from PTM
  unsigned char code; // 0 to 8 code, specifying joystick position
  TFLG1 = 0x80; // acknowledge OC7
  TC7 = TC7+12500; // 100 ms (4,000,000/32/12500= 10 Hz)

  in = PTM&0x0F; // switch input, 0 to 15
  code = CodeTable[in]; // current position of joystick, 0 to 8
  Fifo_Put(code); // send to foreground
}
```