Jonathan W. Valvano May 8, 2003, 2-5pm

(5) Question 1. I/O ports are considered like Global variables

(5) Question 2. Any order as long as all three occur

(5) Question 3. Software crashes because the interrupts is requested over and over.

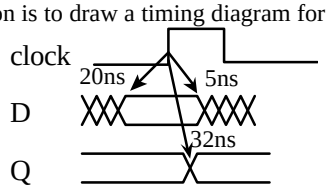
(5) Question 4. ADC resolution=range/precision = 20V/4096 = 5 mV

(5) Question 5. R=(5-2.6-0.4V)/2mA = (2V)/2mA = 1000O

(10) Question 6. Assume an expanded mode 6812 is initialized to have 2 stretches.
The chip select CSP0 becomes 0 and the address becomes valid at 60 ns. The data becomes valid t_{ACC} later. Adding these two together yields the time, $60+t_{ACC}$, when the data first becomes available. The data is required 30 ns before the end of the cycle. With three stretches the cycle time is 375ns. So data is required at 345ns. The maximum allowable value for t_{ACC}

$60+t_{ACC} < 345$ ns
or $t_{ACC} < 285$ ns

(5) Question 7. The objective of this question is to draw a timing diagram for this D flip-flop.



(5) Question 8. First multiply numerator and denominator by 1000 to remove floating point,
Y = (123*X)/1000 + 4560/1000
then perform the division last (overflow can not occur)

```
Y = (123*X + 4560)/1000;
```

(2) Question 9. A high speed communication protocol where both the clock and data are passed from transmitter to receiver.
HH) synchronous serial

(2) Question 10. A communication system that can transfer data in two directions, but only one direction at a time.
O) half-duplex

(2) Question 11. The difference between the true value and the measured value.
A) accuracy

(2) Question 12. A debugging technique that uses paper and pencil to determine in advance specific input/output behaviors of our software, then runs the system and comparing actual results with expected values.
J) desk check

(2) Question 13. The number of information bits transferred per second.
D) bandwidth

(2) Question 14. The amount of time from when new input data is ready until the time the computer to reads the data.
P) latency

(2) Question 15. The interrupt mechanism, like RDRF and TDRE, where multiple potential interrupt requests share the same interrupt vector, but have separate interrupt flags, separate interrupt arm bits, and separate acknowledge sequences.
U) polled interrupt

(2) Question 16. A variable or function that can only be accessed by functions within the same module (e.g., functions within the same file).
W) private

(2) Question 17. A debugging term that means the act of debugging itself has a small but not too noticeable effect on the system being tested.
Q) minimally intrusive

(2) Question 18. A multithreaded system where the direct operations of input and output occur in background interrupt service routines, the foreground thread (main program) processes inputs and generates new outputs, and FIFO queues are used to pass data between the foreground and background.
H) buffered I/O

(10) Question 19. Write 1 2 or 3 lines of C code that acknowledges each type of interrupt.

Part a) Clearing RDRF has two steps

```
  if(SC0SR1&0x20)   // read status with RDRF set
    data = SC0DRL;  // read serial data register
```

Part b) Clearing TDRE has two steps

```
  if(SC0SR1&0x80)   // read status with TDRE set
    SC0DRL = data;  // write serial data register
```

Part c) Clear KWIFH.2, by writing a one to the flag

```
  KWIFH = 0x04;     // clear flag
```

Part d) Clear TOF by writing a one to the flag

```
   TFLG2 = 0x80;    // Acknowledge by clearing TOF
```

Part e) Clear RTIF by writing a one to the flag

```
   RTIFLG = 0x80;   // Acknowledge by clearing RTIF
```

(20) Question 20. Linked data structure and output compare interrupts to implement this Mealy finite state machine.

```
StatePtr = pt;        // current state
unsigned short count; // 1 ms counter used to create time delays
void Initialization(void){
  asm(" sei");    // make atomic
  TIOS |= 0x80;   // PT7 is output compare
  TSCR = 0x80;    // enable TCNT
  TMSK2= 0x33;    // 1us clock
  pt = S0;        // initial state
  DDRH = 0x3F;    // PH4,3,2,1,0 outputs, PH7,6,5 inputs
  count = pt->wait; // time to wait in initial state
  TFLG1 = 0x80;   // Clear C7F
  TMSK1 |= 0x80;  // Arm output compare 7
  asm(" cli");
}
#pragma interrupt_handler TC7handler()
void TC7handler(void){ unsigned char input;
  TC7 = TC7+1000;   // interrupts every 1ms
  if(--count == 0){
    input = PORTH>>5;         // 0 through 7
    PORTH = pt->out[input];   // output depends on input and state
    pt = pt->next[input];     // next depends on input and state
    count = pt->wait;         // time to wait in this new state
  }
  TFLG1=0x80;  // ack by clearing C7F
}
```