Jonathan W. Valvano May 10, 2002, 9am 12 noon

(20) Question 1. The ISR will simply count the number of these edges.

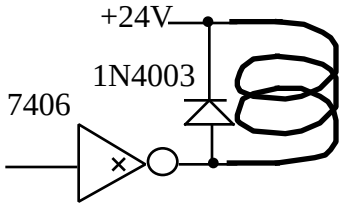
Part a) Show the ritual.

```
void Init(void){
asm(" sei");
  DDRJ  = 0x00;  // all are inputs
  KPOLJ = 0xF0;  // rise on PJ7-4, fall on PJ3-0
  KWIEJ = 0xFF;  // arm all
  KWIFJ = 0xFF;  // clear call flags
  Count = 0;
asm(" cli");}
```

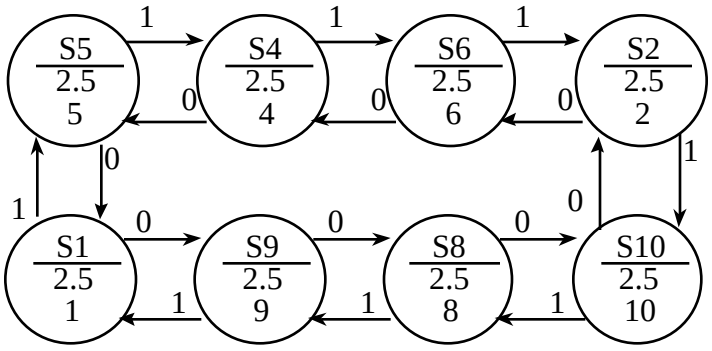
Part b) Show the PORTJ key wakeup ISR. Pay careful attention to consider the case where two or more edges occur at approximately the same time. For example, if PJ0 falls and PJ7 rises, then `Count` should be incremented twice.

```
#pragma interrupt_handler KeyWakupJhandler()
void KeyWakupJhandler(void){
unsigned char flag;
  for(flag=0x80; flag; flag=flag>>1){
    if(KWIFJ&flag){
      Count++;
      KWIFJ = flag;  // clear flag
    }
  }
}
```

(10) Question 2. The current is only 24mA, so a 7406 can be used. The diode is required to handle the back EMF.



(10) Question 3. To spin at 1 rps, there must be 400 steps/sec or 2.5 ms per step.



(10) Question 4. Consider the SCI interrupting system shown as assembly Programs 7.10 through 7.12 in the book.

Part a) Translate the 6812 assembly `OutChar`, show in Program 7.11, into C code.

```
void OutChar(char data){
  while(!TxFifo_Put(data)); // spin if Fifo is empty
  SC0CR2 = 0xAC;            // arm TDRE
}
```

Part b) If `OutChar` is called when the Fifo is full, the while loop will spin forever.

(5) Question 5. The ADC resolution is 10V/1024 or about 10 mV.

(5) Question 6. The Nyquist Theorem states that if you sample at 100 times per second, the frequencies from 0 to 50 Hz are reliably represented in the digital samples.

(10) Question 7.

Part a) The setup and hold times are before and after the rising edge of CSD, so this edge writes data into the RAM.

Part b) WDA=(t_2+t_{13},t_1+t_{14})=(60+46,t_1+20)=(106,t_1+20)

and WDR=(t_1-100,t_1), so 106 ≤ t_1-100, so 206 ≤ t_1, which required 1 stretch

(10) Question 8. The following operations that occur as a typical RTI interrupt is processed. I didn't ask for the order, but these events occur in this order.

D) The RTIF flag is set as a result of the RTI hardware;
J) The PC, Y, X, A, B, CCR registers are pushed on the stack;
H) The I bit is automatically set (I=1) by the hardware (not the result of executing a specific instruction);
K) The RTIF flag is cleared as a result of executing a specific instruction (e.g., `RTIFLG=0x80;`);
E) The `rti` instruction is executed;

(10) Question 9. Consider the case of the serial port interrupt-driven input interface as implemented with SCI12A.C, which is similar to Figure 4.4 in the book. The **RxFifo** queue is used to buffer the data between the background (hardware producer which is the RDRF interrupt service routine) and the foreground (software consumer which is the function `SCI_InChar`). The average serial port input rate is 10 bytes/sec. I.e., the human operator can type 10 characters/sec. The average time for the foreground software to process each character is 1 second. I.e., the average rate at which the foreground calls `SCI_InChar` is 1 Hz. This system does not work properly, and the purpose of this question is to evaluate the problem, and suggest a correction.

Part a) The bandwidth is limited to 1 byte/sec by the rate at which the foreground calls `SCI_InChar`. Thus, the system is CPU bound.

Part b) Since the average rate of calls to `RxFifo_Put` is 10 times the average rate of calls to `RxFifo_Get`, the **RxFifo** will definitely fill up, regardless of Fifo size.

Part c) No, the problem can NOT be fixed by increasing the RxFifo size, because the average arrival rate exceeds the average service rate. There are two solutions. One, you could decrease the typing rate to less than 1 character per second by hiring slower typists. Two, you could use a faster computer or better processing algorithm so that the foreground can process data at rate faster than 10 characters per second.

(10) Question 10. Place the letter code in the empty box for the best definition for the following terms

C	**asynchronous**
H	**atomic**
G	**critical**
I	**noninstrusive**
B	**open collector**
E	**private**
F	**promotion**
J	**real-time**
A	**stabilize**
D	**synchronous**

(A) A debugging technique that specifies all its inputs, so, the output results are consistently repeatable.
(B) A digital logic output that has two states low and HiZ.
(C) A protocol where the two devices have separate and distinct clocks
(D) A protocol where the two devices share the same clock.
(E) An object that can be accessed only by software modules in that group.
(F) Increasing the precision of a number for convenience or to avoid overflow errors during calculations.
(G) Locations within a software module, which if an interrupt were to occur at one of these locations, then an error could occur
(H) Software execution that can't be divided or interrupted.
(I) The collection of information itself does not affect the parameters being measured.
(J) There is an upper bound on the delay between when software is supposed to be run and when it is actually run.