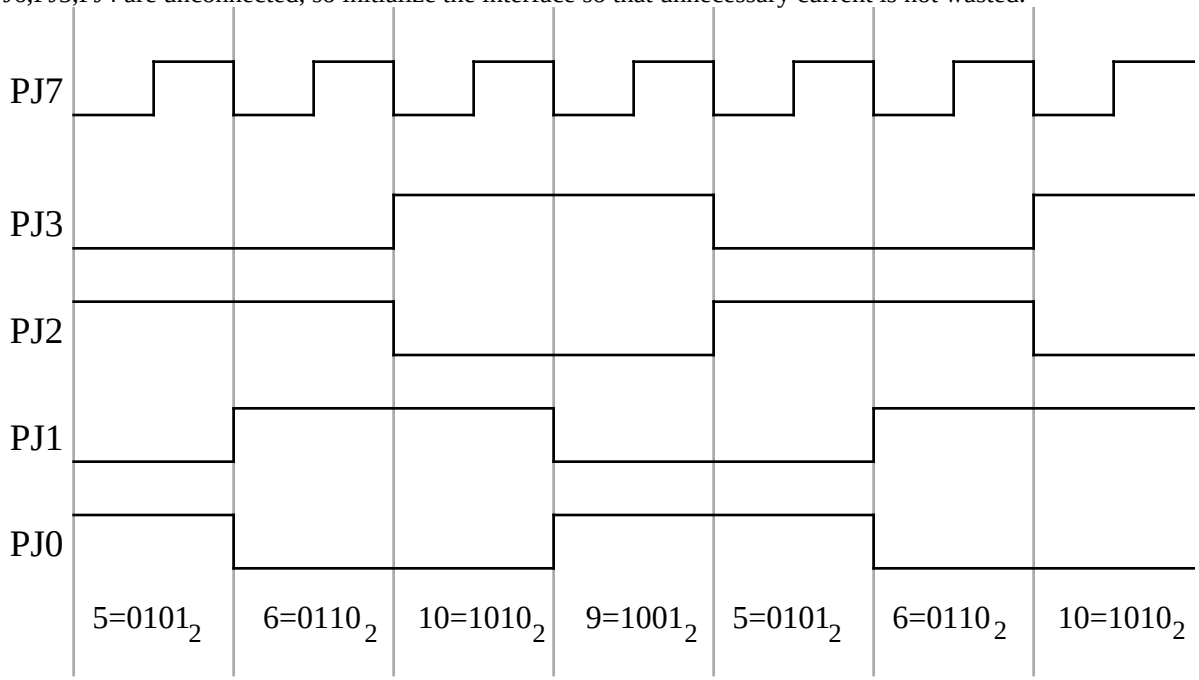
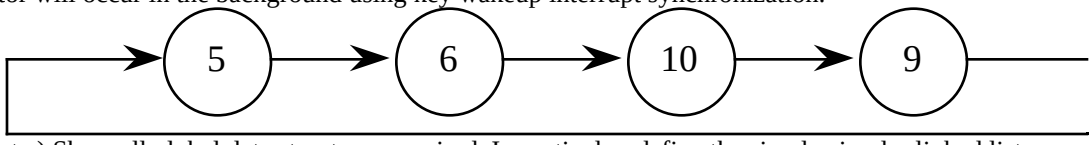
Jonathan W. Valvano First:_____ Last:_____
March 1, 2000, 11:00am-11:50am

This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting.***

The first two questions on this exam deal with the software interface to the same stepper motor. No hardware design is required, but if you are curious, Figures 8.79 and 13.6 show typical hardware interfaces. In this situation, four bits of PORTJ (PJ3, PJ2, PJ1, PJ0) are connected to the stepper motor interface. To spin the motor, your software will output on PJ3-0 to the stepper interface the sequence 5,6,10,9,5,6,10,9,5,6,10,9,… over and over again. The motor will step (rotate 7.5 degrees) each time a new value in this sequence is written to PORTJ. Therefore, the speed of the motor is determined by the time delay between outputs. In these two problems, the time between outputs will be determined by an external signal connected to PJ7. Each time there is a falling edge (1 to 0 transition) on PJ7, your software will "step" the motor (change from 5 to 6, 6 to 10, 10 to 9, or 9 to 5.) This operation is summarized in the following timing diagram. Remember that PJ7 is an input to the 6812 and PJ3,PJ2,PJ1,PJ0 are outputs. PJ6,PJ5,PJ4 are unconnected, so initialize the interface so that unnecessary current is not wasted.



(45) Question 1. In the first implementation, you will use a simple circular linked list like Program 13.2, and key wakeup interrupts to perform the desired input/output functions. Set it up so there is a key wakeup interrupt on each fall of PJ7, and have the ISR change the output from 5 to 6, 6 to 10, 10 to 9, or 9 to 5. You may assume the time between falling edges of PJ7 is long enough so the ISR has time to execute and motor has time to respond. The main program, which you do not write, will call your ritual, then perform unrelated operations. I.e., the outputs to the motor will occur in the background using key wakeup interrupt synchronization.



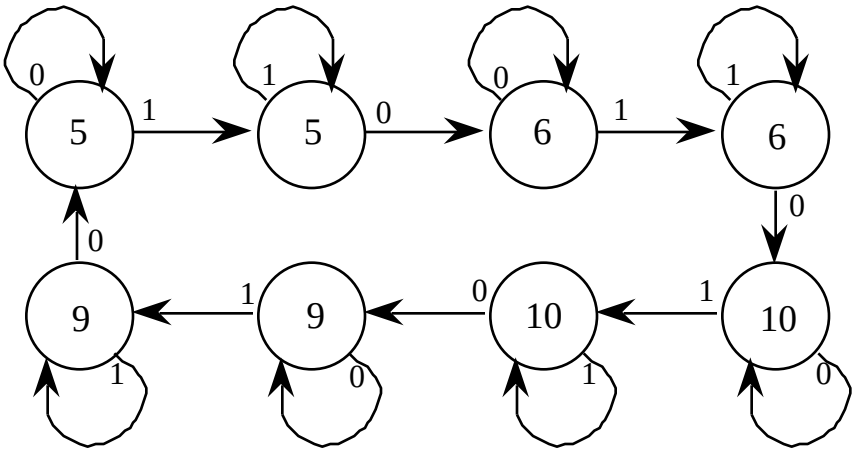
(10) Part a) Show all global data structures required. In particular, define the simple circular linked list.

(15) Part b) Show the ritual that initializes DDRJ, key wakeup on PORTJ, and data structures. Arm and enable interrupts.

(10) Part c) Show the PORTJ key wakeup ISR, which should be executed on the fall of PJ7.

(5) Part d) Show the C code that establishes the key wakeup interrupt vector.

(45) Question 2. In the second implementation, you will use a Moore Finite State Machine, and 976.56 Hz real time interrupts (RTI) to perform the desired input/output functions. Configure it so there is a RTI interrupt every 1024 µsec, and have the ISR execute one sequence of the Moore FSM 1) input, 2) change state (depending on the input, 3) output. You may assume the time between falling edges of PJ7 is large enough so the RTI ISR has time to execute and motor has time to respond. The main program, which you do not write, will call your ritual then perform unrelated operations. I.e., the outputs to the motor will occur in the background using periodic interrupt synchronization.



(15) Part a) Show all global data structures required. In particular, define the linked data structure.

(15) Part b) Show the ritual that initializes RTI, DDRJ, and data structures. Arm and enable interrupts.

(15) Part c) Show the real time interrupt ISR, which should execute the FSM every 1024 μsec. (don't worry about the interrupt vector.)

(**10) Question 3.** Assume the foreground thread calls `SetBit0()` (with interrupts enabled) and a single background thread calls `SetBit1()`. Is there a critical section? Justify your answer.

```
void SetBit1(void){ PORTJ |= 0x02; }
void SetBit0(void){ PORTJ |= 0x01; }
```