CSCE 236 Embedded Systems, Spring 2012
Quiz/Test 2

Thursday, April 12, 2012

Instructions: You will have the full class period to complete this test. **Make sure to show your work to ensure you receive partial credit if your final answer is incorrect.** This is a closed book quiz, no computers, textbooks, notes, etc. are allowed. Unless otherwise specified, assume problems refer to the Arduino board we are using.

Name (5 pts.):

Problem 1. *Hex and bit operations (all references to bit locations are zero referenced). For each bit operation subproblem write a* **single line** *of* `C` *code to achieve the desired result.*

a) (5 pts.). *Set the lower 4 bits in the variable* `var` *to 0x3.*

b) (5 pts.). *Clear the top two bits in the 16-bit variable* `var`.

c) (5 pts.). *What is the value of* `((5<<3) | (1<<6))` *in hex?*

d) (5 pts.). *What is the value of* `((7>>2) + 6)` *in hex?*

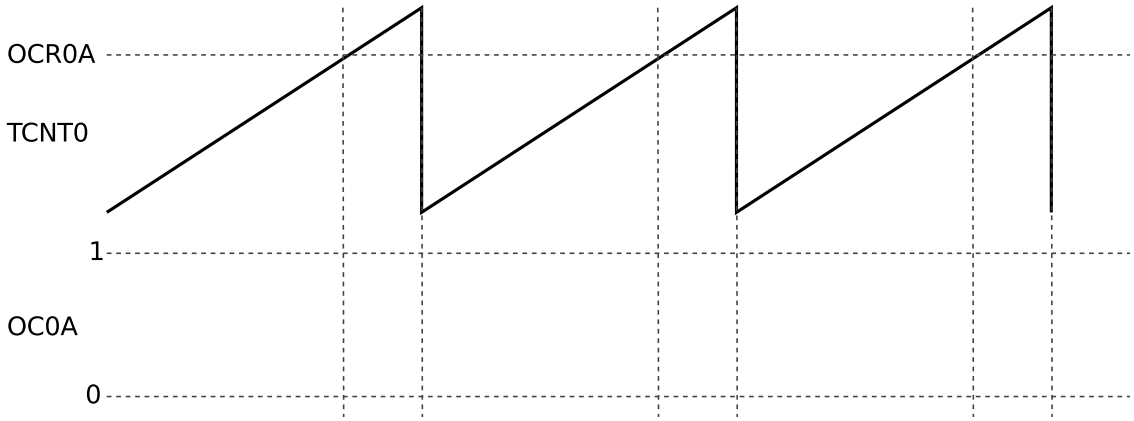
e) (5 pts.). *Set the upper 3 bits in the 8-bit variable* `var` *to the lower 3 bits in the variable* `config`. *(Remember to do this in a single line of code.)*

Problem 2. *Timer/PWM*

a) (5 pts.). *What frequency will the interrupt be triggered on our Arduino with the following configuration (comments should provide all the needed information)? (You can leave the answer as a fraction.)*

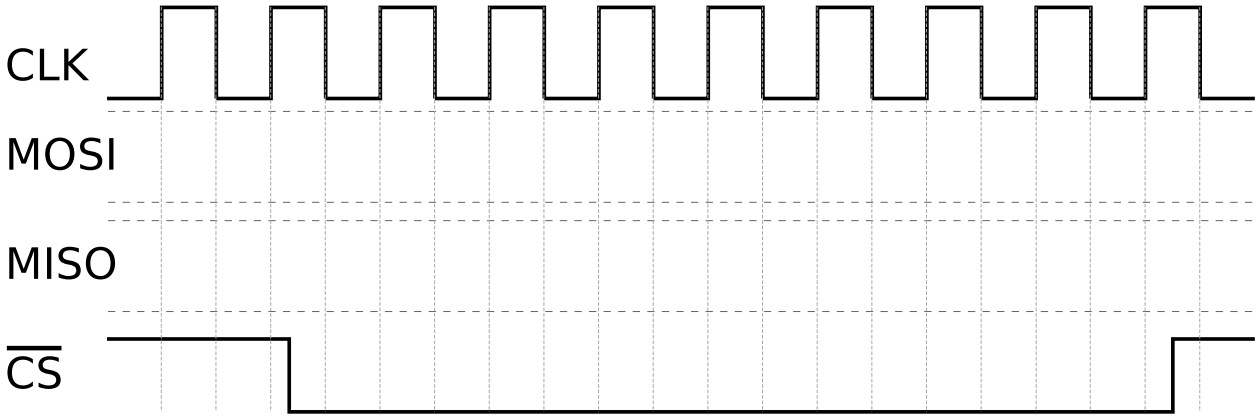
```
void setup(){
  //Set to 10-bit Phase Correct Mode (top is 0x3FF)
  TCCR1A  = (1<<WGM11) | (1<<WGM10);
  //Set clock to clk/1024
  TCCR1B = (1 << CS12) | (1 << CS10);
  //Enable interrupt on match with OCR1A
  TIMSK1 = (1 <<OCIE1A);
  //Set the value of the output compare register
  OCR1A = 200;
}

SIGNAL(TIMER1_COMPA_vect){
  //Interrupt handler code goes here
}
```

b) (5 pts.). *In the following figure, draw the signal output on the pin* `OC0A` *given the clock value* `TCNT0` *(zig-zag line) and the specified output compare register value,* `OCR0A`*. Assume that this is in fast PWM mode, with OC0A configured to non-inverting mode ("Clear* `OC0A` *on Compare Match, set* `OC0A` *at BOTTOM").*



Problem 3. *Communication*

a) (10 pts.)**.** *In the following figure, draw the proper signal for sending the from the master to the slave the value* `0xA3` *on the rising edge and from the slave to the master the value* `0x4B` *on the falling edge.*



b) (5 pts.)**.** *In I^2C, describe how arbitration works when there are two masters. What is unique at the physical layer (compared to UART, SPI, etc.) that enables multiple masters with I^2C?*

c) (5 pts.)**.** *Queuing bytes to be sent with an interrupt over the UART/Serial port makes sense, especially when the speed that it is sending is significantly slower than the processor speed (so it does not need to wait for the output buffer to be free). If queuing a byte requires 60 cpu cyles and handling the transmit interrupt requires 100 cycles, then which serial baud rates should use a transmit queue out of the following baud rates (bits per second) options: 10,000; 50,000; 100,000; 500,000; 1,000,000; 1,500,000; 5,000,000. Assume placing a byte in the output buffer requires just 1 cpu cycle as long as the buffer is free. Explain your reasoning.*

Problem 4. *Interrupt Example Code. For this problem, refer to the following code. This code monitors two of the external interrupts and increments a counter every time one of the interrupts is triggered. It turns on the green LED when INT0 has been triggered more, the red LED when INT1 has been triggered more, and turns off both LEDs when the counts are equal. Both LEDs should not be on at the same time.*

```
01: uint32_t int0Count = 0;
02: uint32_t int1Count = 0;
03:
04: SIGNAL(INT0_vect){
05:   //Turn on the green LED when int0Count is larger than int1Count
06:   if(int0Count++ > int1Count){
07:     digitalWrite(LED_GREEN,HIGH);
08:   }
09: }
10: SIGNAL(INT1_vect){
11:   //Turn on the red LED when int1Count is larger than int0Count
12:   if(int1Count++ > int0Count){
13:     digitalWrite(LED_RED,HIGH);
14:   }
15: }
16:
17: void loop(){
18:   //If both counts are equal turn off the LEDs
19:   if(int1Count == int0Count){
20:     digitalWrite(LED_RED,LOW);
21:     digitalWrite(LED_GREEN,LOW);
22:   }
21: }
```

a) (5 pts.)**.** *This code does not work properly. Describe how this code could end up with: 1) both LEDs on; and 2) both LEDs off when* `int1Count != int0Count`*. Refer to line numbers to help in your explanation.*

b) (5 pts.)**.** *Is it possible to fix one or both of these problems by only modifying the main loop code? If so, what changes need to be made to the main loop and explain why this fixes one or both problems. If one or both of the problems cannot be fixed, explain why not. Remember that* `cli()` *disables interrupts and* `sei()` *enables interrupts.*

c) (5 pts.). *How would you rewrite the interrupt handlers to fix both problems.*

d) (5 pts.). *What are the steps that occur to switch from executing the main code to executing interrupt handler code when an interrupt occurs?*

Problem 5. *Analog to Digital Converters*

a) (5 pts.)**.** *If a 12-bit ADC, with a 5.0V reference reports a value of 981, what is the voltage being read on the pin? (You can leave the answer as a fraction.)*

b) (5 pts.)**.** *Write the Arduino C code to read the analog value on pin* `A0` *and add it to the analog value read from* `A3`*. Store the result in a variable* `sum`*. Make sure to declare this variable before using it.*

Problem 6. *Embedded Operating Systems*

a) (5pts.)**.** *What is the difference between a non-cooperative multi-tasking operating system and a cooperative multi-tasking operating system?*

b) (5pts.)**.** *What does it mean when an OS has a fully preemptive scheduler? What is one advantage and one disadvantage of a fully preemptive scheduler?*