

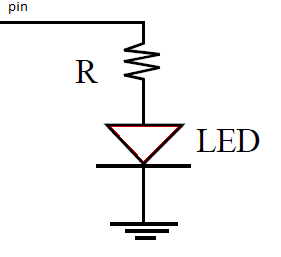
Department of Electrical and Computer Engineering

ENCS438- Interfacing Techniques

**First Hour Exam**

Student Name: Student ID: Sec:

Question#1: (8 points)

1. Write a portion of code to generate a PWM signal with frequency 2kHz and duty cycle 20%.
2. Write one use of each of the following memories in Arduino board (or ATMEGA328P): Flash memory, SRAM, EEPROM.
3. What is the value of R so that the current in the diode is 15 mA when 5 Volt is applied at the pin.
4. We have the following code:

DDRB=0x3C

PORTB=3

Which pins are input and which pins are output?

For output pins, are they HIGH or LOW?

Question#2: (15 points)

A-

1. Draw a 2-bit R-2R ladder DAC with the resistor R=10k Ohm and the resistor 2R=20 k Ohm. Assume that logic 1 is 5 Volt and logic 0 is 0 Volt, what is the output voltage for the input 3.

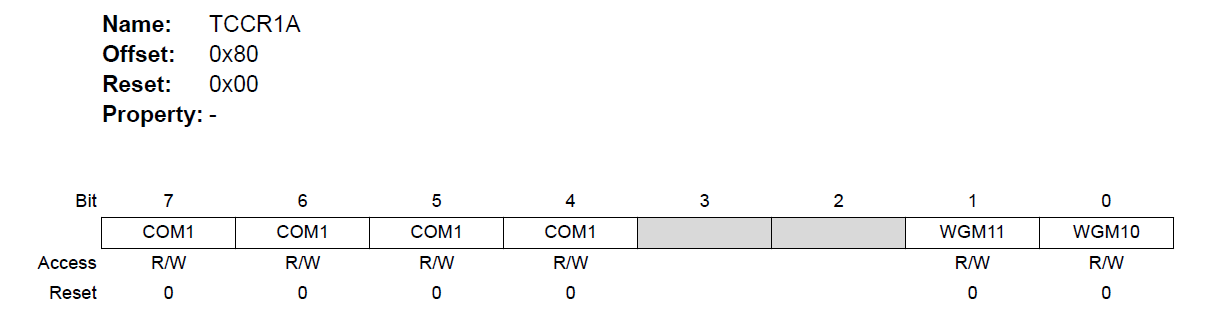
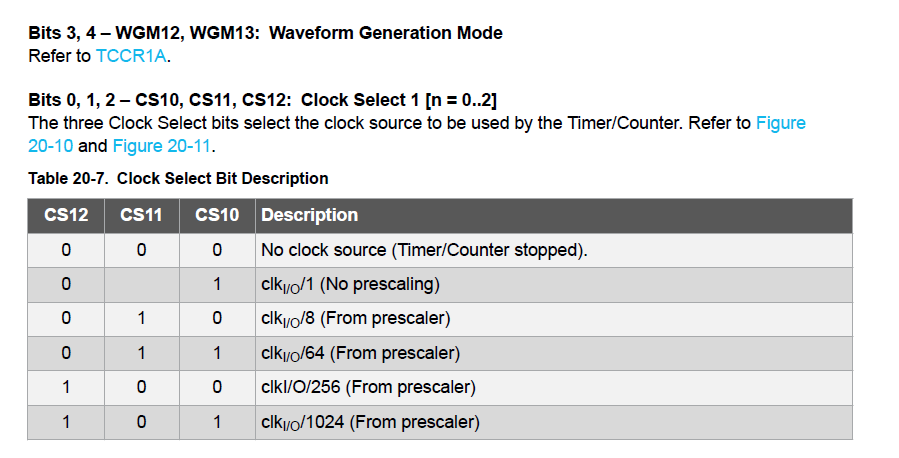
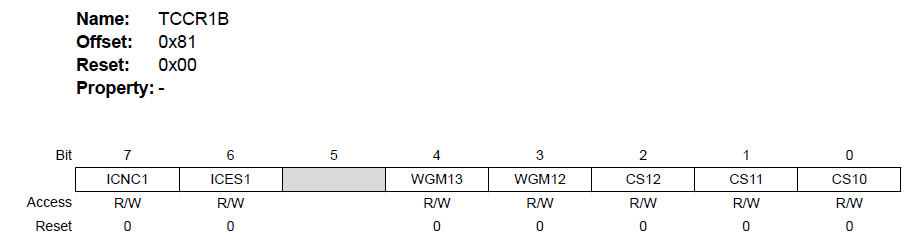
1. Now assume that the resistor 2R= 22 k Ohm, what is the output voltage of the input 3.

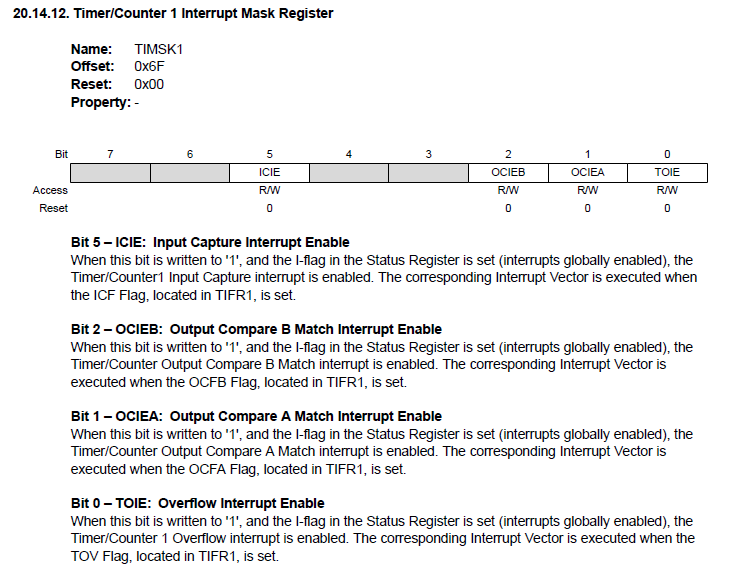
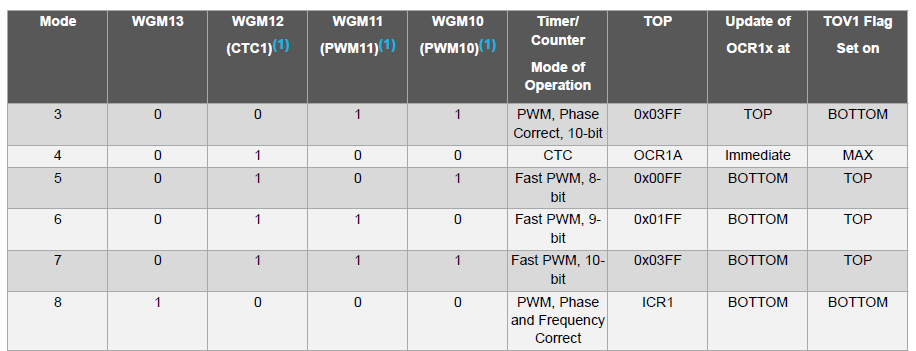
B- Use Normal Mode (Timer Overflow) to read a sensor connected at pin 3 every 7 microseconds. Show the configuration of the registers.

Question#3: (7 points)

In this task we want to measure the human reaction. A person is going to press a push button after a LED is ON. Using Arduino UNO, buzzer, a push button, and resistors (if needed), write an Arduino code that does the following:

1. Make sure that the LED is OFF.
2. Wait a random time
3. If the push button is pressed during the LED is OFF, generate a beep signal with 500 Hz and wait a new random time.
4. After the random time is elapsed, turn the built-in LED ON
5. Measure the time between turning the LED on and the push button is pressed
6. Turn the LED OFF and display the time on the screen using the Serial monitor.



TimerOne.

**initialize(period)**  
You must call this method first to use any of the other methods. You can optionally specify the timer's period here (in microseconds), by default it is set at 1 second. Note that this breaks analogWrite() for digital pins 9 and 10 on Arduino.

**setPeriod(period)**  
Sets the period in microseconds. The minimum period or highest frequency this library supports is 1 microsecond or 1 MHz. The maximum period is 8388480 microseconds or about 8.3 seconds. Note that setting the period will change the attached interrupt and both pwm outputs' frequencies and duty cycles simultaneously.

**pwm(pin, duty, period)**  
Generates a PWM waveform on the specified pin. Output pins for Timer1 are PORTB pins 1 and 2, so you have to choose between these two, anything else is ignored. On Arduino, these are digital pins 9 and 10, so those aliases also work. Output pins for Timer3 are from PORTE and correspond to 2,3 & 5 on the Arduino Mega. The duty cycle is specified as a 10 bit value, so anything between 0 and 1023. Note that you can optionally set the period with this function if you include a value in microseconds as the last parameter when you call it.

**attachInterrupt(function, period)**  
Calls a function at the specified interval in microseconds. Be careful about trying to execute too complicated of an interrupt at too high of a frequency, or the CPU may never enter the main loop and your program will 'lock up'. Note that you can optionally set the period with this function if you include a value in microseconds as the last parameter when you call it.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Random

**random()**

[Random Numbers]

**Description**

The random function generates pseudo-random numbers.

**Syntax**

random(max)  
random(min, max)

**Parameters**

min - lower bound of the random value, inclusive (optional)

max - upper bound of the random value, exclusive

**Returns**

A random number between min and max-1 (long) .

**randomSeed()**

[Random Numbers]

**Description**

randomSeed() initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence. This sequence, while very long, and random, is always the same.

If it is important for a sequence of values generated by random() to differ, on subsequent executions of a sketch, use randomSeed() to initialize the random number generator with a fairly random input, such as analogRead() on an unconnected pin.

Conversely, it can occasionally be useful to use pseudo-random sequences that repeat exactly. This can be accomplished by calling randomSeed() with a fixed number, before starting the random sequence.

**Parameters**

seed - number to initialize the pseudo-random sequence (unsigned long).