Computer Engineering and Mechatronics MMME3085

Solution Sheet 2: Timers and Counters

---

1. **Pulse measurement:** This works by clocking the T/C (and incrementing it each pulse until it overflows and wraps around) using the input T5, no waveform generation is needed so all the WGM50 … WGM53 are zero. There are no output changes on compare register, so COM5A0 etc are all zero.
   CS0 … 3 are all set to 1 because we are clocking the timer from the external input, not from the 16 MHz clock.
   We do enable an interrupt which occurs on register overflow. This is so that we can keep track of how many times it has overflowed by calling an interrupt service routine (ISR) which increments bigLaps every time the counter overflows.
   Pulse generation: We configure the T/C1 as CTC output so that we can choose (via the comparison register OCR1A) the TOP value at which the counter resets and toggles an output pin. We choose which output pin (there are three possibilities, OC1A=pin 11, OC1B=pin12, OC1C=pin 13) using the COM10 … 1 bits in TCCR1A. We choose the waveform generation by setting the WGM10 … 3 bits in TCCR1A and TCCR1B. We choose the CS10 … CS12 bits to set the prescale value by which to divide 16 MHz in order to get an appropriate tick rate to get T/C1 into the right range. You can then experiment with different combinations of prescaler settings CS10 … CS12 and output compare register OCR1A values (in range 0-65535) or, which will give a final pulse rate in the range around 0.12 Hz up to 8 MHz.

2.
```
#include <avr/io.h>

void setup()
{
  DDRB |= (1<<7);  // Set pin 13 as output
  TCCR1A = (1<<WGM11) | (1<<COM1C1);
  TCCR1B = (1<<WGM12) | (1<<WGM13) | (1<<CS10);
  TCCR1C = 0;     // No force output compare
  ICR1 = 799;    // Set PWM frequency noting prescale if any
  OCR1C = 79;  // Set PWM duty cycle as a fraction of ICR1
}

void loop()
{
}
```