Department of Mechanical, Materials and Manufacturing Engineering

The University of Nottingham

Computer Engineering and Mechatronics MMME3085

Exercise Sheet 3: Timers and Counters

1. The following sketch uses hardware facilities (T/C1 and T/C5) to:
   - Generate a high frequency signal (can be done on any output labelled "~").
   - Count pulses on the clock input to one of the T/C (in fact this facility is only available for T/C5 on pin 47 as other timers either don't have this pin accessible or are used for other things).

   Specifically, if pin 13 is connected to pin 47 using a piece of wire, then you can generate a very wide range of frequencies by configuring T/C1's "TOP" value and its prescale value, and measure the frequency achieved using T/C5 configured as a counter.

   a. Work through the program with the Arduino Mega 2560 data sheet and see if you can work out exactly how it works.
   b. If you have an Arduino Mega, have a go with this and confirm that it generates a frequency of 1 MHz.
   c. Now have a go at changing the prescaler values (e.g., via TCCR1B) and the output compare register value (via OCR1A). Look in the Atmega2560 datasheet to see the function of the different bits in these registers.
   d. One version (to give a 100 Hz output) has been given but is commented out, so you will need to replace the previous two lines with these commented ones.
   e. How low can the frequency go? How high? (Note: the value of OCR1A can be anything from 0 to 65535 on Mega.

   This code is available on Moodle as **Generate_and_Measure_Mega.ino**.

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

unsigned long bigLaps; // To record how many times T/C5 overflows

void setup()
{
  Serial.begin(9600);

  // ===Timer/Counter 5 settings to count input pulses ====

  //set pin T5 as input (external); this is pin 47 (Arduino Mega pin mapping)
  pinMode(47, INPUT);// set pin to input

  TCCR5A=0;// No waveform generation needed (T/C is counting only)
  TCCR5B=(1<<CS50)|(1<<CS51)|(1<<CS52);// Normal mode, external clock on rising edge
  TCCR5C=0;//No force output compare (we will not study about this in the module)
  TCNT5=0;//Initialise counter register to zero
  TIMSK5=(1<<TOIE5);//When the T/C reach its maximum, an interrupt will be activated
  bigLaps=0;//When an overflow happens, we will increase this bigLaps by 1

  sei();// Enable all active interrupts

  // === Timer/Counter 1 settings to generate pulses with different frequencies ===

  DDRB|=(1 << 7);//T/C1 channel C is connected to pin 13, set it as output
  TCCR1B=(1 << WGM12 );//Configure T/C1 for CTC mode
  TCCR1A=(1 << COM1C0 );//Enable T/C1 Compare Output channel C in toggle mode
 // OCR1A= 7;//Set CTC compare value to 2 MHz at 16MHz AVR clock with no prescaling
 // TCCR1B|= (1 << CS10 ) ;//Start T/C1 at Fcpu with no prescaling (i.e., 16MHz)
  OCR1A=1249;//Set CTC compare value to 200 Hz at 16MHz AVR clock with 1/64
prescaling
  TCCR1B|= (1 << CS10 )|(1 << CS11 );// Start T/C1 at Fcpu (i.e., 16MHz)
}

// === This is a routine which will be called if an overflow happens in T/C5 ===
ISR(TIMER5_OVF_vect )
{
  bigLaps++; //when this runs, you had 65365 pulses counted
}

 unsigned long Current_Count=0;//A variable to store the total number of pulses

void loop() {
 Current_Count = TCNT5 + bigLaps*65536;
  Serial.println(Current_Count); // Print the current number of pulses
  delay(1000); // Intervals between each two consecutive prints
}
```

2. You need to configure T/C 1 to output PWM (“fast PWM”) on pin 13 at 20 kHz.
   - Several possible solutions but suggest:
   - T/C 1 fed with 16 MHz clock signal directly (divisor 1, i.e. no prescaling)
   - Fast PWM: TCNT1 counts up until matches value in ICR1 (“TOP”), resets to 0, restarts
   - This sets output high at each cycle start
   - Output C goes low when counter register TCNT1 matches value in OCR1C (i.e., “clears on compare match”)

   Can you come up with the register-level code to implement this? (See lecture 3 for some further hints and diagrams showing how this might work). Implement this in the following skeleton program:

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


void setup()
{
  DDRB |= ????;  // Set pin 13 as output
  TCCR1A = ????;
  TCCR1B = ????;
  TCCR1C = 0;    // No force output compare
  ICR1 = ????;   // Set PWM frequency noting prescale if any
  OCR1C = ????;  // Set PWM duty cycle as a fraction of ICR1
}

void loop()
{
}
```