TECH 3157 MICROCONTROLLERS APPLICATIONS SECTION 01

LAB 4: TIMERS/COUNTERS IN ATmega328P

PROFESSOR: Dr. JIN ZHU

STUDENT: DIDAN JUNQUEIRA RIBEIRO

NOVEMBER 16, 2015
CEDAR FALLS, IA

Objectives

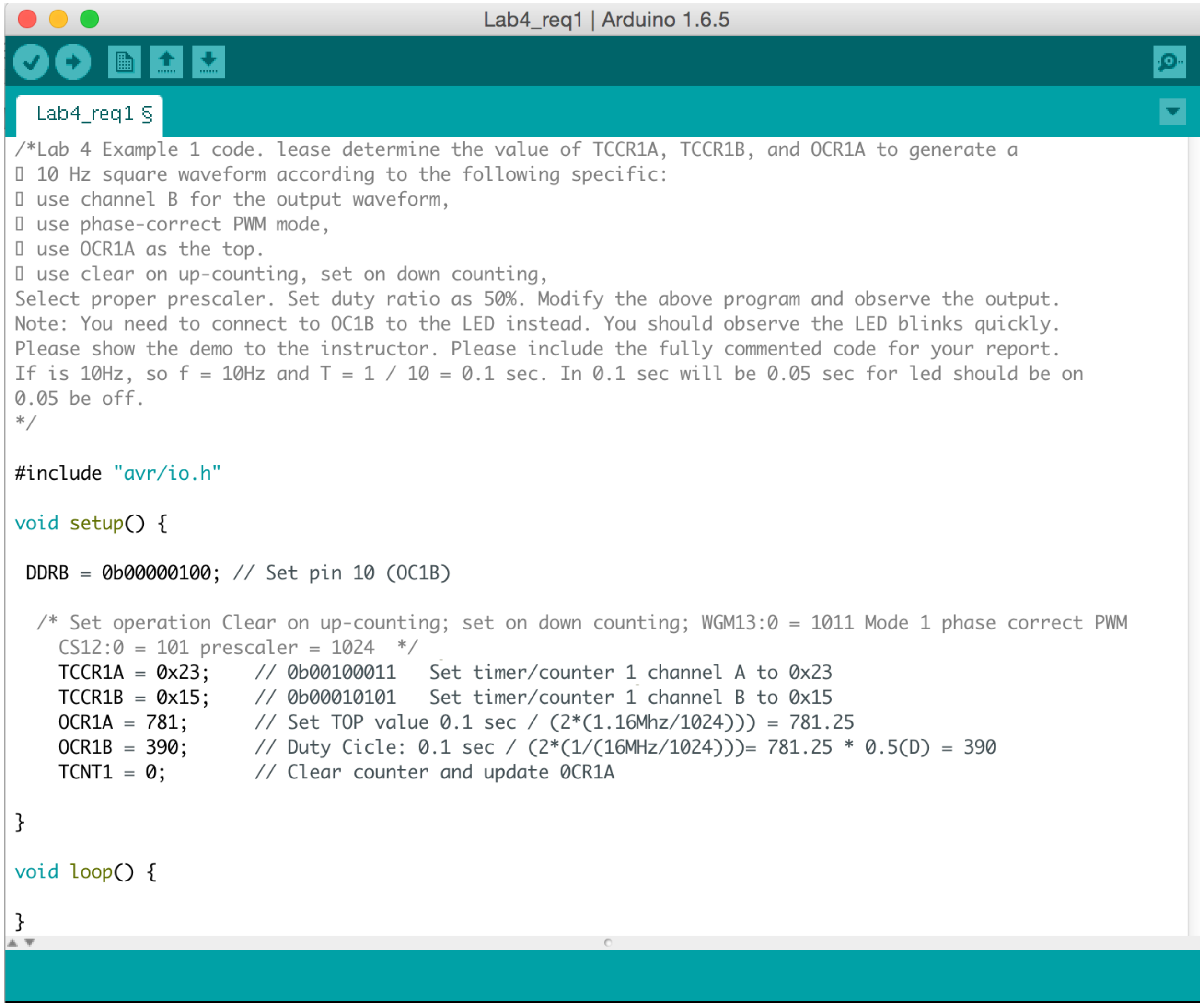
1. Understand the operation modes of the timers/counters in ATmega328P.
2. Learn to program timers to generate time delays and waveforms.
3. Learn to program counters to count the external event.

Equipment

BOE-BOT Shield Kit, Red LED, Pushbutton, 22k ohm resistor, 16 nF Capacitor, Oscilloscope

Procedure and Design

The first lab requirement asked to determine the value of TCCR1A, TCCR1B and OCR1A to generate a 10 Hz square waveform using channel B for the output waveform, phase-correct PWM mode, set OCR1A as the top and use clear on up-counting and set on down counting. The result of get a duty ratio of 50% was achieved and was possible to observe the red LED blinks quickly. The program used to perform that is shown below:



```
/*Lab 4 Example 1 code. lease determine the value of TCCR1A, TCCR1B, and OCR1A to generate a
 10 Hz square waveform according to the following specific:
 use channel B for the output waveform,
 use phase-correct PWM mode,
 use OCR1A as the top.
 use clear on up-counting, set on down counting,
Select proper prescaler. Set duty ratio as 50%. Modify the above program and observe the output.
Note: You need to connect to OC1B to the LED instead. You should observe the LED blinks quickly.
Please show the demo to the instructor. Please include the fully commented code for your report.
If is 10Hz, so f = 10Hz and T = 1 / 10 = 0.1 sec. In 0.1 sec will be 0.05 sec for led should be on
0.05 be off.
*/

#include "avr/io.h"

void setup() {

 DDRB = 0b00000100; // Set pin 10 (OC1B)

  /* Set operation Clear on up-counting; set on down counting; WGM13:0 = 1011 Mode 1 phase correct PWM
    CS12:0 = 101 prescaler = 1024  */
    TCCR1A = 0x23;     // 0b00100011   Set timer/counter 1 channel A to 0x23
    TCCR1B = 0x15;     // 0b00010101   Set timer/counter 1 channel B to 0x15
    OCR1A = 781;       // Set TOP value 0.1 sec / (2*(1.16Mhz/1024))) = 781.25
    OCR1B = 390;       // Duty Cicle: 0.1 sec / (2*(1/(16MHz/1024)))= 781.25 * 0.5(D) = 390
    TCNT1 = 0;         // Clear counter and update 0CR1A

}

void loop() {

}
```

Figure 1: Code of Lab Requirement 1

The second part of the lab was use PWM to get analog signal using Timer/Counter 1 to generate a 200 Khz square waveform. Then was used a RC as the LPF to observe analog voltage. Finally the duty ratio was adjusted thought OCR1B for 20% and 80% to get an analog output. The results was achieved when changed dutyRatio function was adjusted for the desired value. For RC values was used the formula Fc = 1 / 2*pi*R*C and to set the Timer 1 registers was used the Timer/Counter 1 register table to determine which bit to set according to the specifications as will be showing on the program used to perform that lab shown below:

```
/* In this part, we will use PWM to get analog signal that we need. First,
 *  use Timer/Counter 1 to generate a 200 KHz square waveform similar
 *  to part 1. Then Use a RC as the LPF to observe the output analog voltage.
 *  Modify the duty ratio by adjusting the OCR1B for 20% and 80% and observe
 *  the analog output. Remember in order for the pwm to work, we need
 *  fanalog << fc << fpwm. */
// Function Prototype
void PWMInit();
void OutputVoltage(float);
#include “avr/io.h"
void setup() {
  char message[80];
  DDRB = 0b00000100;   // Set pin 10 (OC1B)
  Serial.begin(9600);
```

```
    /* Set operation Clear on up-counting; set on down counting; WGM13:0 = 1011 Mode 1
phase correct PWM  CS12:0 = 001 prescaler = No pre scaling  */

     TCCR1A = 0x23;    // 0b00100011    Set Timer/Counter 1 Channel A to 0x23

     TCCR1B = 0x11;    // 0b00010001    Set Timer/Counter 1 Channel B to 0x11

     PWMInit();            // OCR1A = 40

     sprintf(message,"OCR1A: %d \n",  OCR1A); // Check the value of OCR1A on Serial Monit

     Serial.write(message);

     OutputVoltage(0.5);      // Get a duty ratio of 50% in this example

     sprintf(message,"OutputVoltage: %d \n \t", OCR1B );

     Serial.write(message);

   //   OCR1A = 40;   // Set TOP value 5us sec / (2*(1/(16 MHz/1024))) = 40

   //    OCR1B = 32;  // Duty Cycle(D): 0.1 sec / (2*(1/(16 MHz/1024)))= 781.25 * 0.5(D) =
390

      TCNT1 = 0;        // Clear counter and update 0CR1A

  }

  void loop() {

  }

  void PWMInit()              // Determine function PWMInit()

  {

   OCR1A = 40;                // OCR1A = 40

  }

  void OutputVoltage(float dutyRatio)       // Determine function OutputVoltage
```

```
{
  OCR1B = OCR1A * dutyRatio;              // Get OCR1B
}
```

The connect circuit used for this part is shown below:

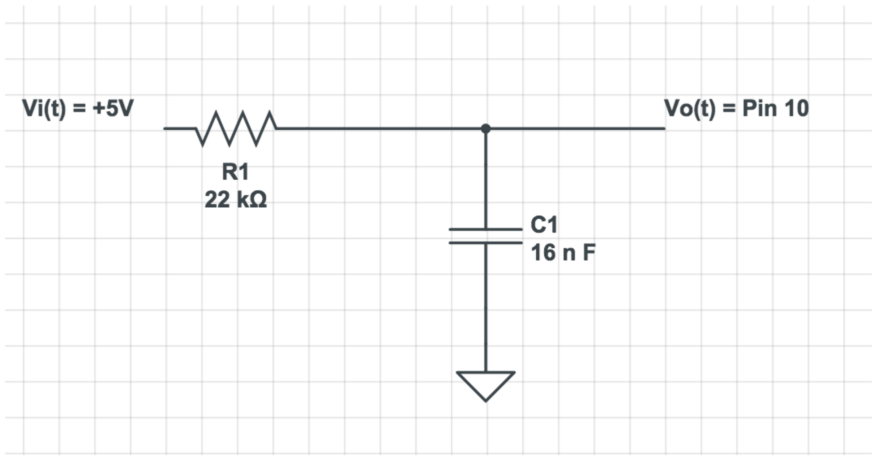


Figure 2: LPF

The third part was required to generate a triangle waveform of approximately 500 Hz. Timer 0 and Timer 1 registers was defined according to the specifications as shown in the code below:

```
/* In this part, you are required to generate a triangle waveform of approximately 500 Hz, as
*shown in  Figure 4.3. You may use N different duty ratios (for example N= 10, Duty ratio
*should be in sequence of 0,  0.1, 0.2,...., 0.9, 1, 0.9, 0.8, ...0.1, 0 and repeats) to get the needed
```

```
*analog voltage levels (0V, 0.5V, 1V, 1.5V, ...4.5V, 5V) for your triangle waveform. Please reuse
*the functions you implemented  in previous parts.  */
#include "avr/io.h"
//Function Prototype
void My100SecDelay();
void PWMInit();
void OutputVoltage(float);


void setup() {
  DDRD = 0b00100000;    // Set pin 6 (OC0A)
  DDRB = 0b00000100;    // Set pin 10 (OC1B)
  /* Set operation Clear on up-counting; set on down counting; WGM12:0 = 101 Mode 5 phase
correct PWM CS12:0 = 010 prescaler = 8  */
  OCR0A = 100;          //Set TOP value 100us sec / (2*(1/16 MHz/8))) = 100
  OCR0B = 50;           // Duty Cycle(D): 100us sec / (2*(1/(16 MHz/8)))= 100 * 0.5(D) = 50
  TCCR0A = 0x21;        // 0b00100001
  TCCR0B = 0x0A;        // 0b00001010;
  TCCR1A = 0x23;        // 0b00100011
  TCCR1B = 0x11;        // 0b0010001
  PWMInit();            // Call function PWMInit()
  OutputVoltage(0.7); // Call function Output Voltage()
  TCNT1 = 0;  // Clear counter and update 0CR1A
```

```
 TCNT0 = 0; // Clear counter and update 0CR0A
}
void loop() {
   double sample;        // Variable for the sample in analysis
   unsigned char currentIndex; // For loop interaction
   static const float m_triangleWaveform[] = { 0.000, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600,
0.700, 0.800, 0.900, 1.000, 0.900, 0.800, 0.700, 0.600, 0.500, 0.400, 0.300, 0.200, 0.100 };
// Array to store different duty ratio
   for (currentIndex = 0; currentIndex < 21; currentIndex ++) // For loop to initialize array
  {
   sample = m_triangleWaveform[currentIndex];
   currentIndex = (currentIndex+1) % 20;
   OutputVoltage(sample);
   My100SecDelay();
  }
}
void PWMInit()          // Define PWMInit function
{
 OCR1A = 40;
}
void OutputVoltage(float dutyRatio)  // Define OutputVoltage function
{
  OCR1B = OCR1A * dutyRatio;
}
void My100SecDelay()          // Define My100SecDelay function
{
  do { }
  while ((TIFR0 & (0x1 << TOV0)) == 0);   // wait till TOV0 to be set
}
```

The flowchart for the program is shown below:

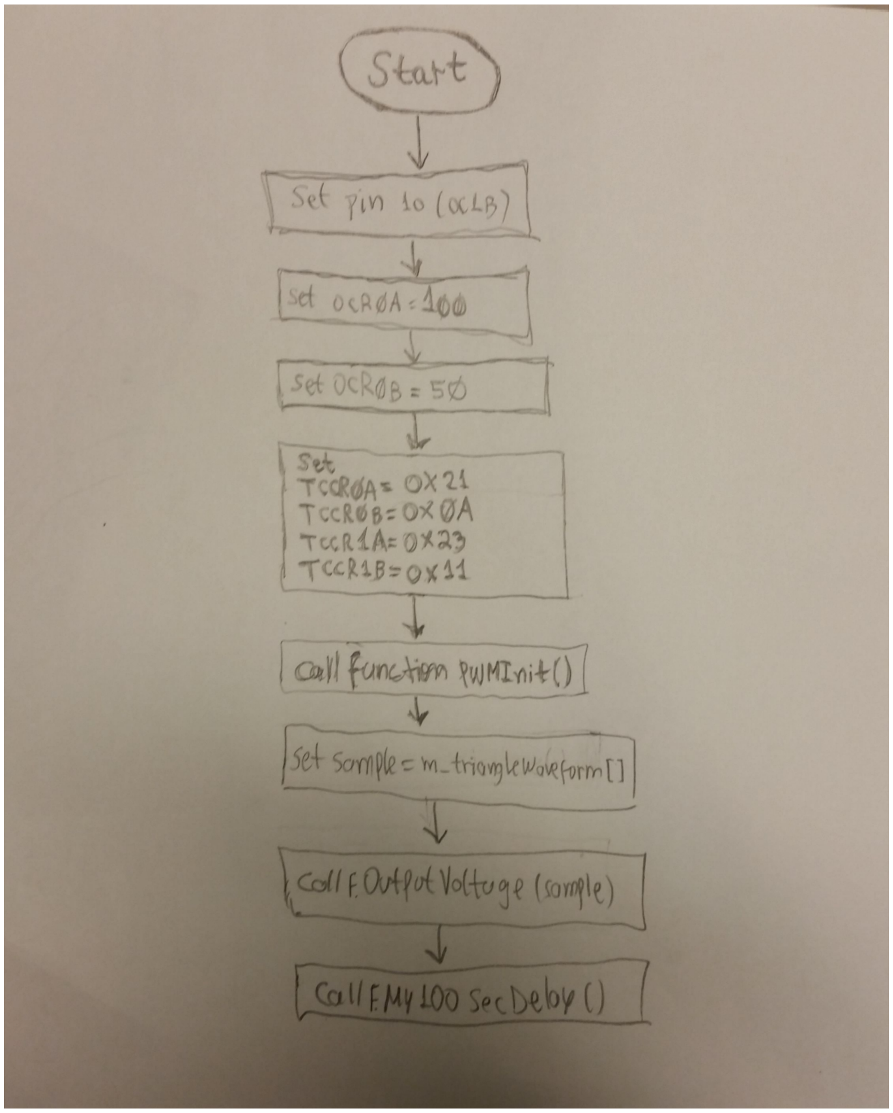
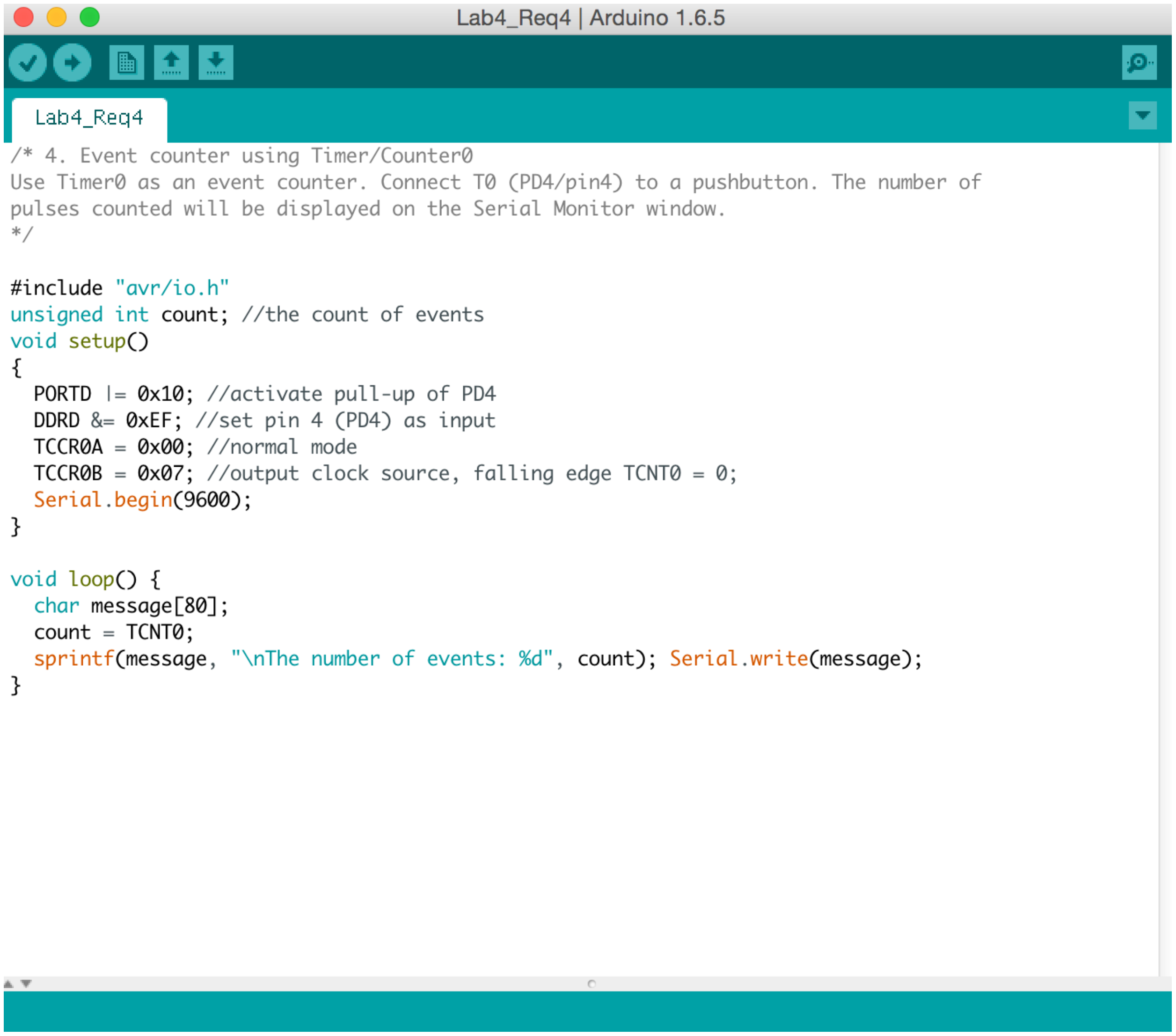


Figure 3: Lab requirement 3 Flowchart

The last part asked to use Timer0 as an event counter. T0 was connect to pin4 to a pushbutton. Then, the number of pulses counted is show in a Serial Monitor Window. Only one event is counted when the pushbutton is press one time. However, if keep pressing fast it will count multiple times due debounce effect. The code is shown below:



```
/* 4. Event counter using Timer/Counter0
Use Timer0 as an event counter. Connect T0 (PD4/pin4) to a pushbutton. The number of
pulses counted will be displayed on the Serial Monitor window.
*/

#include "avr/io.h"
unsigned int count; //the count of events
void setup()
{
  PORTD |= 0x10; //activate pull-up of PD4
  DDRD &= 0xEF; //set pin 4 (PD4) as input
  TCCR0A = 0x00; //normal mode
  TCCR0B = 0x07; //output clock source, falling edge TCNT0 = 0;
  Serial.begin(9600);
}

void loop() {
  char message[80];
  count = TCNT0;
  sprintf(message, "\nThe number of events: %d", count); Serial.write(message);
}
```

Figure 4: Code for Lab requirement 4

Conclusion

All the lab requirements was achieved. Following the hints in the lab 4 manual and using some programming background in arrays and how to initialize was possible to implement the lab requirement 3 to generate the triangle waveform on oscilloscope.