# ECED 3204 Microprocessor

# Assignment #5 Reference Solution

# http://www.jasongu.org/3204/assignments.html

Assignment #5 contains the following problems:

**E11.1** No. of clock cycles required to create a time delay of 7 ms with 2 MHz timer clock is
$7 \times 10^{-3} \times 2 \times 10^{6} = 14{,}000$ clock cycles

**E11.4** Write a sequence of AVR instructions and C statements to be executed by an ATMega640/1280/2560 MCU to generate a 1-kHz PWM waveform with 30% duty cycle from the OC2B pin assuming that $f_{clk\_IO} = 16$ MHz.

**Solution**: We will use the fast PWM, non-inverting mode to generate this waveform. We will set the clock source to **clk_I/O ÷ 64**. The TOP value is calculated to be $16 \times 10^{6} \div 64 \div (1 \times 10^{3}) = 250$. The value to set the duty cycle is $250 \times 30\% = 75$.

```
.def    tmp = r16
ldi     r16,0x40
sts     DDRH,r16        ; configure PH6/OC2B pin for output
ldi     tmp,0x23        ; select non-inverting fast PWM mode
sts     TCCR2A,tmp      ; use OCR2A as TOP value
ldi     tmp,250         ; set period to 1 ms
sts     OCR2A,tmp       ;     “
ldi     tmp,75          ; set duty cycle to 30%
sts     OCR2B,tmp       ;     “
ldi     tmp,0x0C        ; select clk_I/O / 64 as timer clock input
sts     TCCR2B,tmp      ;     “
```

The sequence of C statements to generate the same waveform is as follows:

```
DDRH          |= 0x40;      // configure OC0B/PG5 pin for output
TCCR2A    = 0x23;      // select non-inverting PWM mode
OCR2A     = 250;       // set up signal period 1 kHz
OCR2B     = 75;        // set up waveform duty cycle to 30%
TCCR2B    = 0x0C;      // select clk_I/O / 64 as timer 2 clock input
```

**E11.7** Write a C program to measure the duty cycle of a periodic square wave connected to the ICP1 pin of the ATmega640/1280/2560 device.

**Solution**: We need to capture three edges: a rising edge, a falling edge, and a rising edge. This program takes the slow signal into account. We will use **ovCnt1** and **ovCnt2** to keep track of the number of times that Timer1 overflows during the high and low interval in a cycle. We will force Timer1 to count up from 0 during its high and low interval. The total clock cycle counts for the signal high and low intervals are given by the following two expressions:

hiWidth = captured TCNT1 value + ovCnt1 * 65536

loWidth = captured TCNT1 value + ovCnt2 * 65536

The duty cycle can then be calculated as follows:

duty = (hiWidth * 100) / (hiWidth + loWidth)

The following C program measures the duty cycle of the signal connected to the ICP1 pin:

```
#include <avr/io.h>
#include <avr/interrupt.h>
unsigned char ovCnt1, ovCnt2; // timer1 overflow count in high and low interval
unsigned char hiOlo; // indicator for high or low interval

int main(void)
{
    unsigned long int hiWidth, loWidth, period;
    unsigned char duty;
    hiOlo       = 1;    // waveform starts with high
    ovCnt1      = 0;
    ovCnt2      = 0;
    TCCR1A      = 0;            // configure Timer1 to normal mode
    DDRD            &= 0xEF;    // configure ICP1/PD4 pin for input
    TIFR1       = 0x2F;         // clear all flags related to Timer 1
    TCCR1B      = 0x42;         // capture rising edge, use clk_IO/8 as Timer1 clock input
    while(!(TIFR1 & (1 << ICF1))); // wait for the arrival of the first rising edge
    TCNT1       = 0;            // force timer1 to count up from 0
    TIFR1       = 0x21;         // clear ICF1 and TOV1 flags
    TIMSK1      = 0x01;         // enable TOV1 interrupt
    TCCR1B      = 0x02;         // capture the falling edge
    sei();                      // enable interrupt
    while(!(TIFR1 & (1 << ICF1))); // wait for the arrival of the falling edge
    hiOlo       = 0;            // enter low interval
    hiWidth     = TCNT1;        // save clock count in high interval
    TCNT1       = 0;            // force Timer1 to count up from 0 again
    TIFR1       = 0x21;         // clear ICF1 and TOV1 flags
    TCCR1B      = 0x42;         // prepare to capture rising edge
    while(!(TIFR1 & (1 << ICF1)));  // wait for the arrival of the second rising edge
    TCCR1B      = 0;            // stop Timer 1
    loWidth     = TCNT1;        // save the clock count in low interval
    hiWidth     += (unsigned long int)ovCnt1 * 65536;
    loWidth     += (unsigned long int)ovCnt2 * 65536;
    period = hiWidth + loWidth;
    duty        = (hiWidth * 100) / period;
    while(1);
}
// ------------------------------------------------------------------------------------
// Timer1 overflow interrupt service routine.
// ------------------------------------------------------------------------------------
ISR(TIMER1_OVF_vect)
{
    if(hiOlo){
        ovCnt1++;
    }
    else{
        ovCnt2++;
    }
}
```

**E11.9** Write a set of C functions that can create a time delay that is a multiple of 50 μs, 1 ms, 10 ms, 100 ms, and 1 s, respectively. The multiple is passed to these functions as a parameter with the type of **unsigned int**. These functions are to be executed by an MEGA640/1280/2560 MCU running with a 16-MHz crystal oscillator.

**Solution**:

```
// ------------------------------------------------------------------------------------------------
// This function creates a delay which is a multiple of 50 us. The multiple is passed to this function.
// ------------------------------------------------------------------------------------------------
void delayby50us(unsigned int k)
{
    TCCR5A    = 0x00;      // configure Timer 5 to CTC mode with clock
    TCCR5B    = 0x19;      // set to clk_IO / 1 (WGM3:0 = 12)
    TCNT5     = 0;         // TCNT5 counts up from 0
    ICR5      = 800;       // ICR5 is the TOP value
    TIFR5     = 1<<ICF5;   // clear ICF5 flag
    while(k) {
        while (!(TIFR5 & (1<<ICF5))); // wait for 1 ms
        TIFR5 = 1<<ICF5;  // clear ICF5 flag
        k--;
  }
}
// ------------------------------------------------------------------------------------------------
// This function creates a delay which is a multiple of 1 ms.
// ------------------------------------------------------------------------------------------------
void delayby1ms(unsigned int k)
{
    TCCR5A    = 0x00;      // configure Timer 5 to CTC mode with clock
    TCCR5B    = 0x1A;      // set to clk_I/O / 8 (WGM3:0 = 12)
    TCNT5     = 0;         // TCNT5 counts up from 0
    ICR5      = 2000;      // ICR5 is the TOP value
    TIFR5     = 1<<ICF5;   // clear ICF5 flag
    while(k) {
        while (!(TIFR5 & (1<<ICF5))); // wait for 1 ms
        TIFR5 = 1<<ICF5;  // clear ICF5 flag
        k--;
  }
}
// ------------------------------------------------------------------------------------------------
// This function creates a delay which is a multiple of 1 ms.
// ------------------------------------------------------------------------------------------------
void delayby10ms(unsigned int k)
{
    TCCR5A    = 0x00;      // configure Timer 5 to CTC mode with clock
    TCCR5B    = 0x1A;      // set to clk_I/O / 8 (WGM3:0 = 12)
    TCNT5     = 0;         // TCNT5 counts up from 0
    ICR5      = 20000;     // ICR5 is the TOP value
    TIFR5     = 1<<ICF5;   // clear ICF5 flag
    while(k) {
        while (!(TIFR5 & (1<<ICF5))); // wait for 1 ms
        TIFR5 = 1<<ICF5;  // clear ICF5 flag
```

```
            k--;
    }
}
// ------------------------------------------------------------------------------------------------------------------
// The following function creates a time delay that is a multiple of 100 ms using Timer 5 CTC mode.
// ------------------------------------------------------------------------------------------------------------------
void delayby100ms(unsigned int k)
{
    TCCR5A    = 0x00;       // configure Timer 5 to CTC mode with clock
    TCCR5B    = 0x1B;       // set to clk_I/O / 64 (WGM3:0 = 12)
    TCNT5     = 0;          // TCNT5 counts up from 0
    ICR5      = 25000;      // ICR5 is the TOP value
    TIFR5     = 1<<ICF5;    // clear ICF5 flag
    while(k) {
        while (!(TIFR5 & (1<<ICF5))); // wait for 1 ms
        TIFR5 = 1<<ICF5;  // clear ICF5 flag
        k--;
    }
}
// ------------------------------------------------------------------------------------------------------------------
// The following function creates a time delay that is a multiple of 1 s using Timer 5 CTC mode.
// ------------------------------------------------------------------------------------------------------------------
void delayby1s(unsigned int k)
{
    TCCR5A    = 0x00;       // configure Timer 5 to CTC mode with clock
    TCCR5B    = 0x1C;       // set to clk_I/O / 256 (WGM3:0 = 12)
    TCNT5     = 0;          // TCNT5 counts up from 0
    ICR5      = 62500;      // ICR5 is the TOP value
    TIFR5     = 1<<ICF5;    // clear ICF5 flag
    while(k) {
        while (!(TIFR5 & (1<<ICF5))); // wait for 1 ms
        TIFR5 = 1<<ICF5;  // clear ICF5 flag
        k--;
    }
}
```

**E11.13** Write an AVR C program to be run on an ATmega640/1280/2560 demo board to generate a periodic square wave with 2-kHz frequency and 70% duty cycle using the Timer 1 CTC mode. The MEGA640/1280/2560 is running with a 16-MHz crystal oscillator.

**Solution**: Suppose we select 1 as the prescaler to the Timer1 clock input, a period of the 2-kHz square will consist of 8,000 timer clock cycles. To generate a 2-kHz square with a 70% duty cycle, we need to alternate the TOP value (of the CTC mode) between 5600 and 2400. The following assembly program will generate the specified waveform:

```
#include   <avr/io.h>
#include   <avr/interrupt.h>
#define    hiTime      5600
#define    loTime      2400
unsigned   char        hiOlo;

int main(void)
{
```

```
    DDRB            |= 0x20;       // configure PB5/OC1A pin for output
    TCCR1A      = 0xC0;       // select set high as compare match pin action
    TCCR1C      |= 0x80;       // force output compare OC1A
    TCCR1A      = 0x40;       // select Toggle as CTC mode pin action
    ICR1        = hiTime;     // waveform start with high
    hiOlo       = 0;          // next waveform interval will be low
    TIMSK1      = 0x01;       // enable ICF1 interrupt
    TIFR1       = 0x3F;       // clear all timer flags
    TCCR1B      = 0x19;       // select clk_IO as clock input to Timer1
    sei();
    while(1);
}
ISR(TIMER1_ICP1_vect)
{
    if(hiOlo){ // should next interval be high?
        ICR1  = hiTime;
        hiOlo = 0;
    } else {
        ICR1  = loTime;
        hiOlo = 1;
    }
}
```