Microcomputers EEL 4746

Assignment 3 – **Daniel Taylor**

1. Review Chapter 9 from “The AVR Microcontroller and Embedded System Using Assembly and C”
2. Submit the solution for the following problems.

Problem 1

Find the TCCR0A and TCCR0B values for Normal mode, no presacral, with the clock coming from the AVR’s crystal (16MHz)

TCCR0A =00000000 = **0x00**

TCCR0B = 00000001 = **0x01**

Problem 2

Assume the XTAL=8MHZ. Find the TCNT0 value needed to generate a time delay of 5ms. Use Normal mode, and the largest pre-scaler possible.

Largest prescaler for timer 0 is 1024

With 8000000Hz and 1024 pre-scalar, frequency is 8MHz / 1024 = 7812.5 times a second.

TCCR0A = 00000000 = 0x00 (Normal Mode)

TCCR0B =00000101 = 0x05 (1024 Pre-Scaler)  
  
Delay = (1/f)\*number of cycles, f = 8000000  
Number of cycles = delay\*f = (5\*10^-3 \* 7812.5) = 39.0625 clock cycles on 1024 prescaler. Round down to 39.

**TCNT0 = 256 - 39**

Problem 3

Assume the XTAL=1MHz. Find the OCR0A value needed to generate a time delay of 0.2ms. Use CTC mode, and no prescaler mode.

Delay = 0.2 ms, number of clock cycles = (0.2 \* 10^-3) \* (1000000) = 200

TCCR0A = 00000010 (CTC Mode)  
TCCR0B = 00000001 (No Prescaler)  
  
OCR0A = n – 1 = **199**

Problem 4

Program Timer0 in C to generate a square wave of 2kHz. Assume that XTAL=16MHz (use CTC mode)(

Square wave with freq = 2000, meaning period is 1/2000 (0.5 ms), so the square wave is on for 0.25 ms, off for 0.25 ms.

Delay is 0.25 ms, so number of clock cycles is (.25\*10^-3) \* (16000000) = 4000. Using timer0, so a prescaler is needed. Can do clk/64 prescaler, so n = 62.5.

62.5 – 1 = 61.5, round to 62

**#include <avr/io.h>  
int main(void) {  
DDRB |= 1<<2; //pin 2 on port B will generate the wave  
PORTB &= ~(1<<2); //clear pin 2 on port B to set it LOW before wave is generated**

**while(1) {**

**//start clock**

**TCCR0A = 0x02; //(CTC Mode)**

**TCCR0B = 0x03; //(clk / 64 prescalar)**

**OCR0A = 62;**

**//monitor**

**while(TIFRO & (1<<OCF0A) == 0) {}**

**//instructions**

**PORTB ^= (1<<2);**

**}  
return 0;  
}**

Problem 5

State the difference between Timer0 and Timer1.

**The main difference between Timer0 and Timer1 is the size of each timer’s main bit register. Timer 0 (as well as Timer2) is 8bit, while Timer1 is 16-bit. This allows Timer1 to have a lot more counter modes than Timer0 or Timer2.**

Problem 6

Program Timer1 in C to generate a square wave of 3kHz. Assume that XTAL=10MHz (use Normal mode).

# of clock cycles is (1/2 \* 1/3000)\*(10000000) =1666.66 clock cycles, round down to 1666 clock cycles. Can be done with Timer 1 no pre-scalar

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

**int main(void) {**

**DDRC |= 1<<6; //Data direction, will display square wave on port C pin 6 so make it output**

**PORTC &= ~(1<<6); //set portC pin 6 low before generating square wave**

**While(1) {**

**//start clock**

**TCNT1 = 65536 – 1666; //Number of clock cycles  
TCCR1A = 0x00; //Normal mode**

**TCCR1B = 0x01; //No pre-scalar**

**//monitoring**

**while( (TIFR1 & (1<<TOV1)) ==0) {}**

**//stop & reset clock**

**TCCR1B = 0;**

**TIFR1 = 0x01;**

**//instruction, toggle PORTC with delay for a square wave**

**PORTC ^= (1<<3);**

**}  
return 0;  
}**

Problem 7

Program Timer0 in C to be an event counter. Use Normal mode and display the binary count on PORTD continuously. Set the initial count to 20

**#include <avr/io.h>  
  
int main(void) {**

**//data direction, all of PORTD is an output**

**DDRD = 0xFF;**

**//initially clear PORTD**

**PORTD = 0x00;**

**while (1) {**

**//start clock**

**TCNT0 = 20;**

**TCCR0A = 0x00; //Normal mode**

**TCCR0B = 0x01; //No prescaler**

**while((TIFRO & 1<<TOV0) == 0) {**

**//display current TCNT0 value on PORTD. Inside while loop as TCNT0 increments**

**PORTD = TCNT0;**

**}**

**//stop clock**

**TCCR0B = 0;**

**TIFR0 = 0x01;**

**}**

**return 0;  
}**

//Wasn’t sure if PORTD = TCNT0 should be inside the timer while loop, since we’re wanting to display TCNT on PORTD. I think displaying it after the while loop would be when TCNT reaches overflow and resets back down to 0 before being set to 20. Also, if wanting to be displayed, you might want to throw a delay inside the monitor/counting while loop, so that the “display” can actually be seen moving in real time with human eyes. That would look like this:

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

**#include <util/delay.h>  
  
int main(void) {**

**//data direction, all of PORTD is an output**

**DDRD = 0xFF;**

**//initially clear PORTD**

**PORTD = 0x00;**

**while (1) {**

**//start clock**

**TCNT0 = 20;**

**TCCR0A = 0x00; //Normal mode**

**TCCR0B = 0x01; //No prescaler**

**while((TIFRO & 1<<TOV0) == 0) {**

**//display current TCNT0 value on PORTD. Inside while loop as TCNT0 increments. Have delay of 1 second so the display can actually be seen**

**\_delay\_ms(1000);**

**PORTD = TCNT0;**

**}**

**//stop clock**

**TCCR0B = 0;**

**TIFR0 = 0x01;**

**}**

**return 0;  
}**