Hw3

Please follow the example code attached in the package.

1. (20%) Atmega128 and Pic24e have the reset interrupt at the program address 0x0. Write a function reset() that works for both chips to reset a program. Copy and paste your code in report.

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define soft\_Reset()

#define F\_CPU 4000000UL //4mhz

volatile bool \_\_ran\_interupt = 0;

void \_\_reset\_Interupt(void){

do {

wdt\_enable(WDTO\_15MS);

for(;;) {

}

} while(0)

PORTB ^= 0x00;

\_\_ran\_interupt = true;

}

int main(void) {

PORTB ^= 0xFF;

OCR1A = 100;

OCR1B = 10;

TCCR1A = 0;

TCCR18 |= (1 << WGM12) | (1 << CS12);

// enable timer compare interrupt

TIMSK1 |= (1 << OCIE1A);

sei(); // Turns interupts on

while(!\_\_ran\_interupt)

{

\_\_reset\_Interupt();

}

}

2. (20%) Briefly explain what are the functions declared in each of the header files stdlib.h, string.h, math.h, ctype.h and assert.h? (Do not list and explain each function.)

The functions that are declared within the top portion are their respective extern functions, that handle the initialization of their defined values at the beginning of the file. What we find is that the files will have the overall structure of first the #define method to a certain extern function declaration and then define then function call with typically the \_ATTR\_CONST\_\_ method type for the call. Therefore, the functions in the beginning of all the files are for initialization of constants or setting up calculations that will be used throughout.

stdlib.h - This header defines several general purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting.

string.h - defines several functions to manipulate C strings and arrays.

math.h - declares a set of functions to compute common mathematical operations and transformations.

ctype.h - declares a set of functions to classify and transform individual characters.

assert.h - defines one macro function that can be used as a standard debugging tool.

3. (20%) Configure and code clock for Pic24e.

a) Report how to configure bits to use its fast RC oscillator with PLL to run at 40MHz. Assume the RC's frequency is 7.5MHz.

In order to configure bits with the information above we get the following code below. Overall one has to determine the Fosc by using the simple formula of Fin \* (M/(n1\*n2)) and from that number it allows us to match it up with the oscillator frequency. Which would t=be twice as much as the PLL which in this case is 75. Hence why the numbers below make sense for the configuration of the bits.

\_FOSCSEL(FNOSC\_FRCPLL); // FRC Oscillator

\_FOSC(FCKSM\_CSECMD & OSCIOFNC\_OFF & POSCMD\_NONE);

\_FNOSC=FRCPLL, FRCDIV=0, PLLFBD=62, PLLPRE=1, PLLPOST=0 (7.5MHz)

Int main(void) {

/\* Configure Oscillator to operate the device at 40Mhz \*/

// Fosc= Fin\*M/(N1\*N2), Fcy=Fosc/2

// Fosc= 7.5M\* (40/(2\*2)) which is around 75Mhz for 7.5Mhz input clock

**// Configuring Bits**

**PLLFBD=38; // M=40-2**

**CLKDIVbits.PLLPOST=0; // N1=2-2**

**CLKDIVbits.PLLPRE=0; // N2=2-2**

}

b) Report how to configure bits to use its primary crystal (XT) with PLL to run at 40MHz. Assume the external clock is a crystal oscillator of 8MHz.

Refer to the example code included in case10.clock. Modify the code and copy and paste the code in your report (excluding the comments).

FNOSC=XTPLL, PLLFBD=38, PLLPRE=0, PLLPOST=0 (8MHz)

PLLFBD = 38;

CLKDIVbits.PLLPRE = 0;

CLKDIVbits.PLLPOST = 0;

4. (20%) Assume four LEDs are connected to Pins A4..7. Implement the following functions with bit operators only (|, &, ^, ~, <<, |=, &=, ^=).

a) Show the code for Atmega128.

(1) Initialize the LEDs.

void initLeds() {

// All inputs

DDRA |= 0xF0;

}

(2) Turn one selected LED on.

void ledOn(uint8\_t sel) {

PORTA |= 1<<(sel+4);

}

(3) Turn one selected LED off.

void ledOff(uint8\_t sel) {

PORTA &= ~(1<<(sel+4));

}

(4) Toggle one selected LED.

void ledToggle(uint8\_t sel) {

PORTA ^= (1<<(sel+4));

}

(5) Set a value to the LEDs.

void ledSet(uint8\_t val) {

PORTA = (PORTA & 0xF8) | (val & 0xF0);

}

b) Show the code for Pic24ep512gu810.

(1) Initialize the LEDs.

void initLeds() {

// All inputs

TRISA &= 0xFF0F;

}

(2) Turn one selected LED on.

void ledOn(uint8\_t sel) {

LATD |= 1<<(sel+5);

}

(3) Turn one selected LED off.

void ledOff(uint8\_t sel) {

LATD &= ~(1<<(sel+5));

}

(4) Toggle one selected LED.

void ledToggle(uint8\_t sel) {

LATD ^= (1<<(sel+5));

}

(5) Set a value to the LEDs.

void ledSet(uint8\_t val) {

LATD = (LATA & 0xFF0F) | ((val & 0x7) << 1);

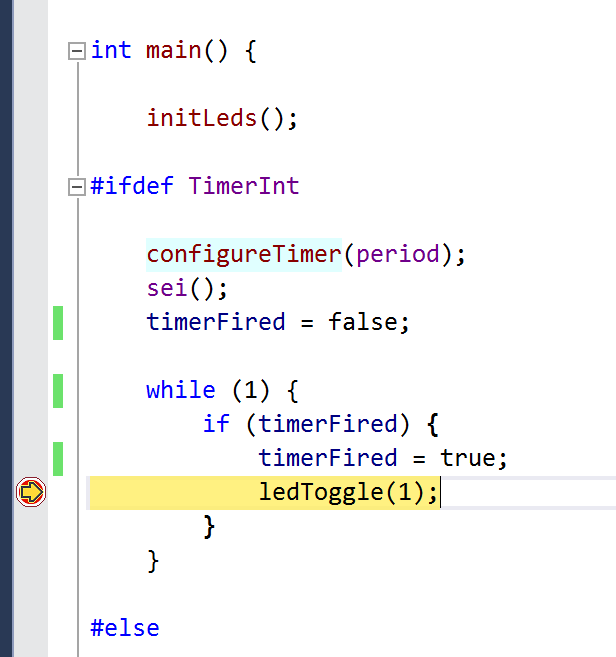
}

Pic24 Difference included having (sel +1) instead of sel. Also instead of using PORTA the pic24 chipset uses LATD.

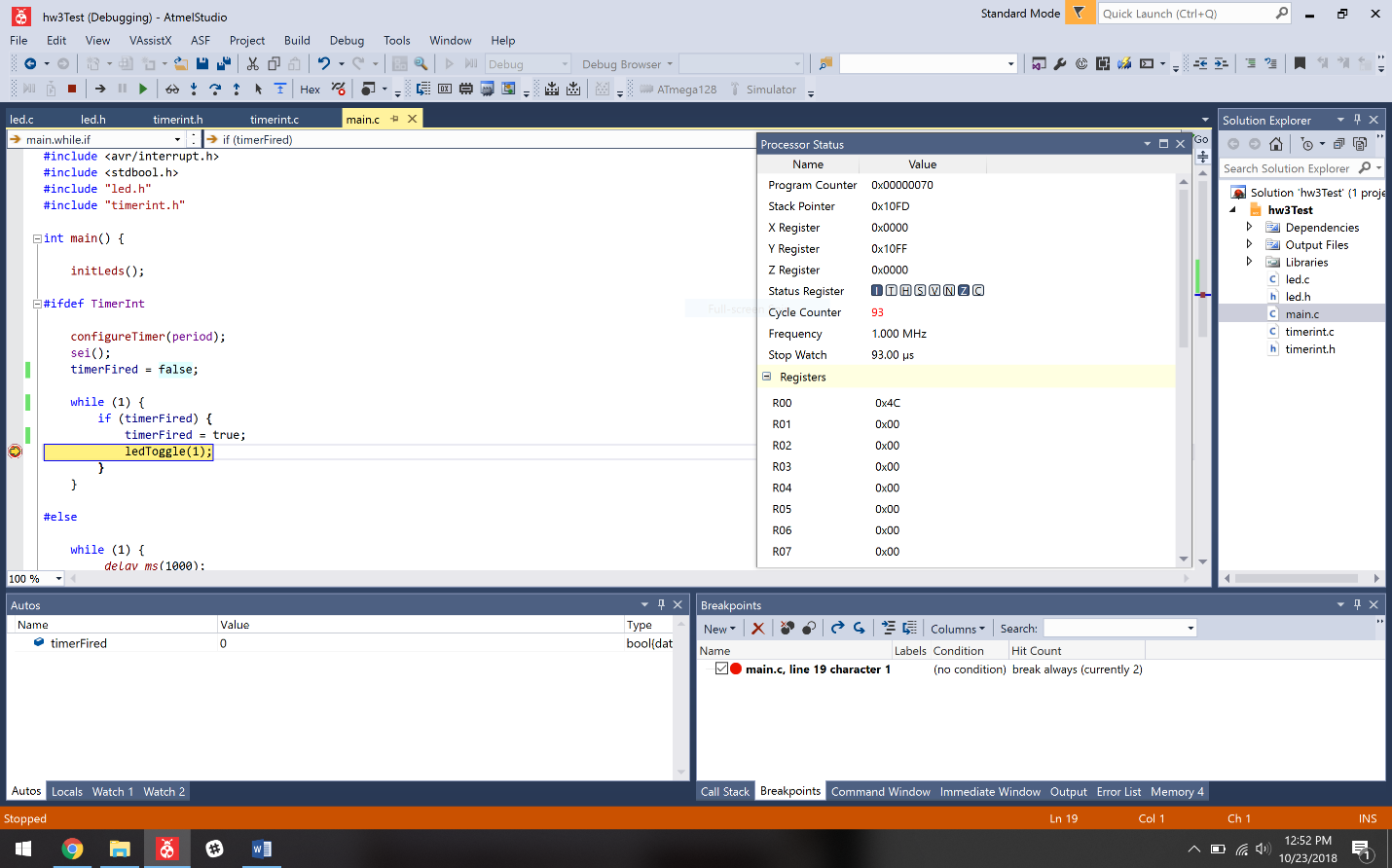
You may refer to case12.led for example code.

5. (20%) Make a program to blink/toggle a LED every second with the Atmega128 chip.

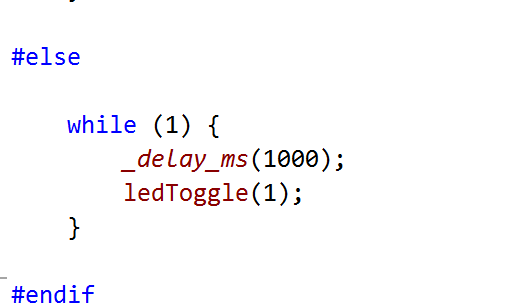
a.1) Show the code to use the timer interrupt for implementation.



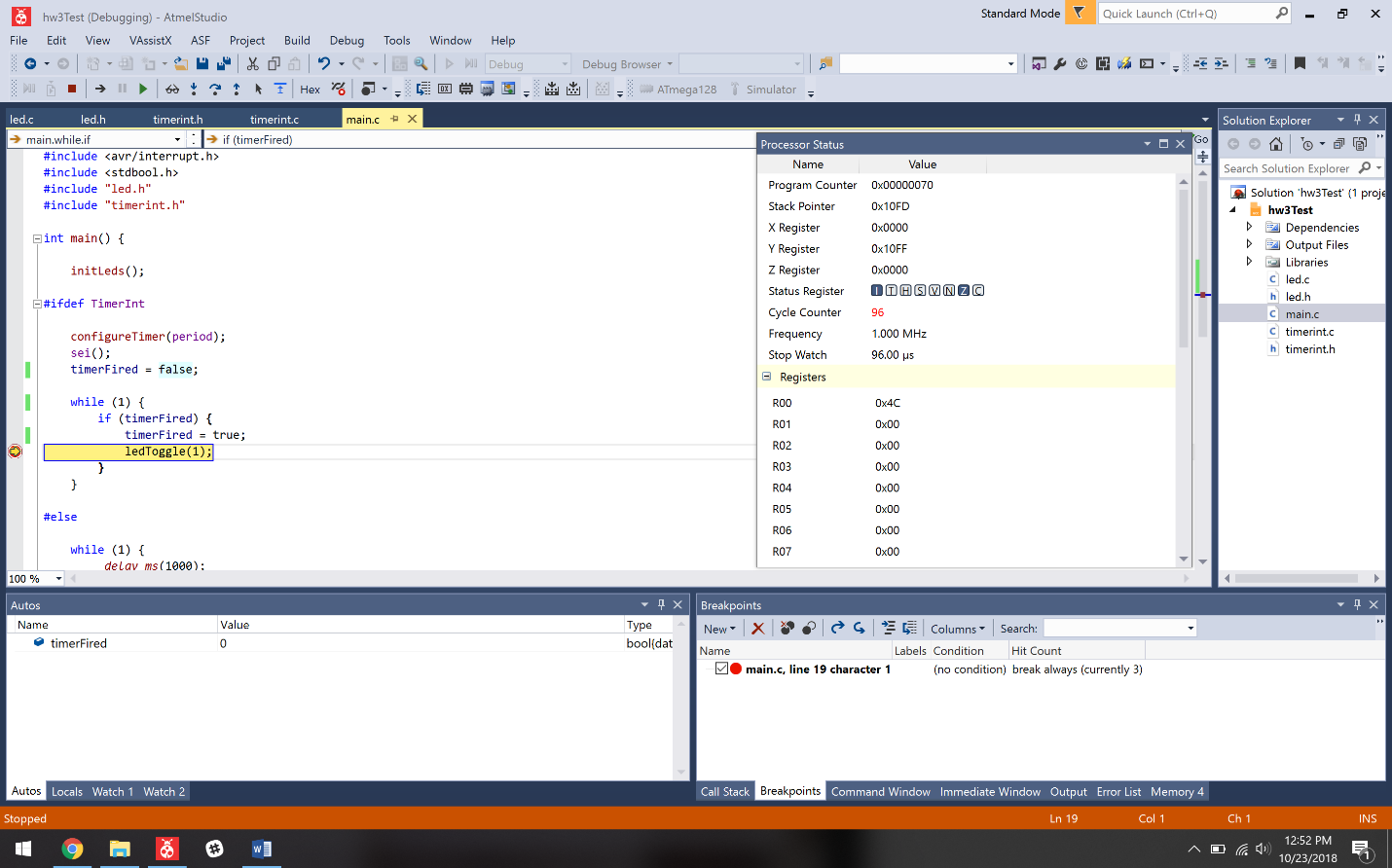
a.2) Show the screen shot of the stopwatch on the second toggle in AVR Studio.

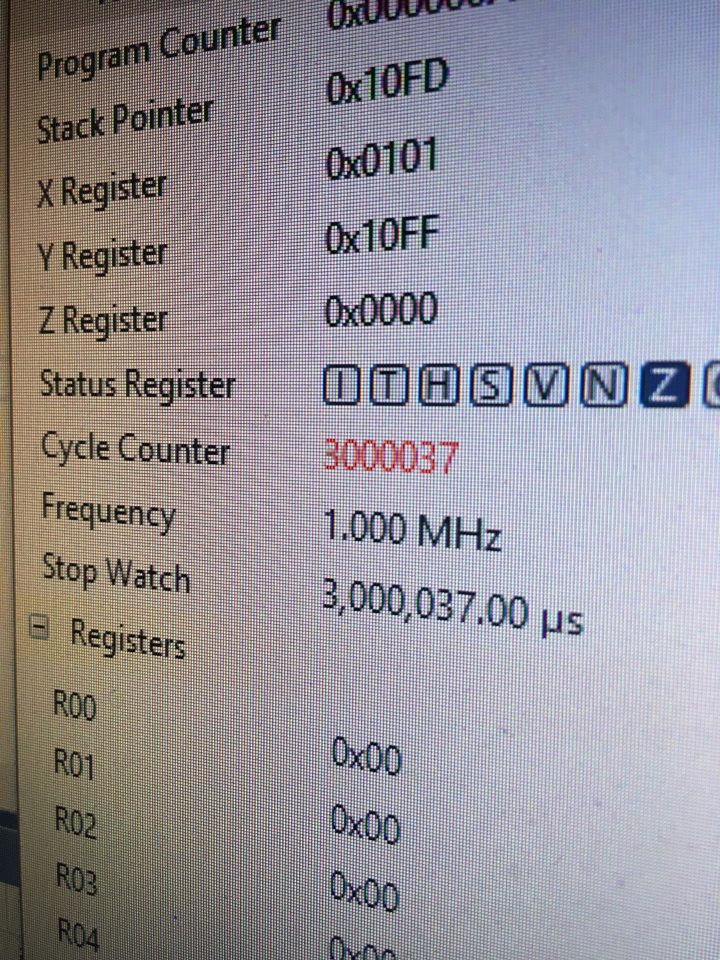


b.1) Show the code to use the delay function for implementation.



b.2) Show the screen shot of the stopwatch on the third toggle in AVR Studio.





You may call the toggle function in Question 4 to toggle the led and do not need to show the toggle function again.

Hint: Because the delay function has a maximum delay limit, you will need an internal counter to accumulate delays to one second.

You may refer to case14.blink for example code.