CPE301 – SPRING 2019

Design Assignment DA2A

Student Name: Cody McDonald

Student #: 5000382538

Student Email: mcdonc4@unlv.nevada.edu

Primary Github address: https://github.com/elev8rProcrastinator/submission\_da.git

Directory:

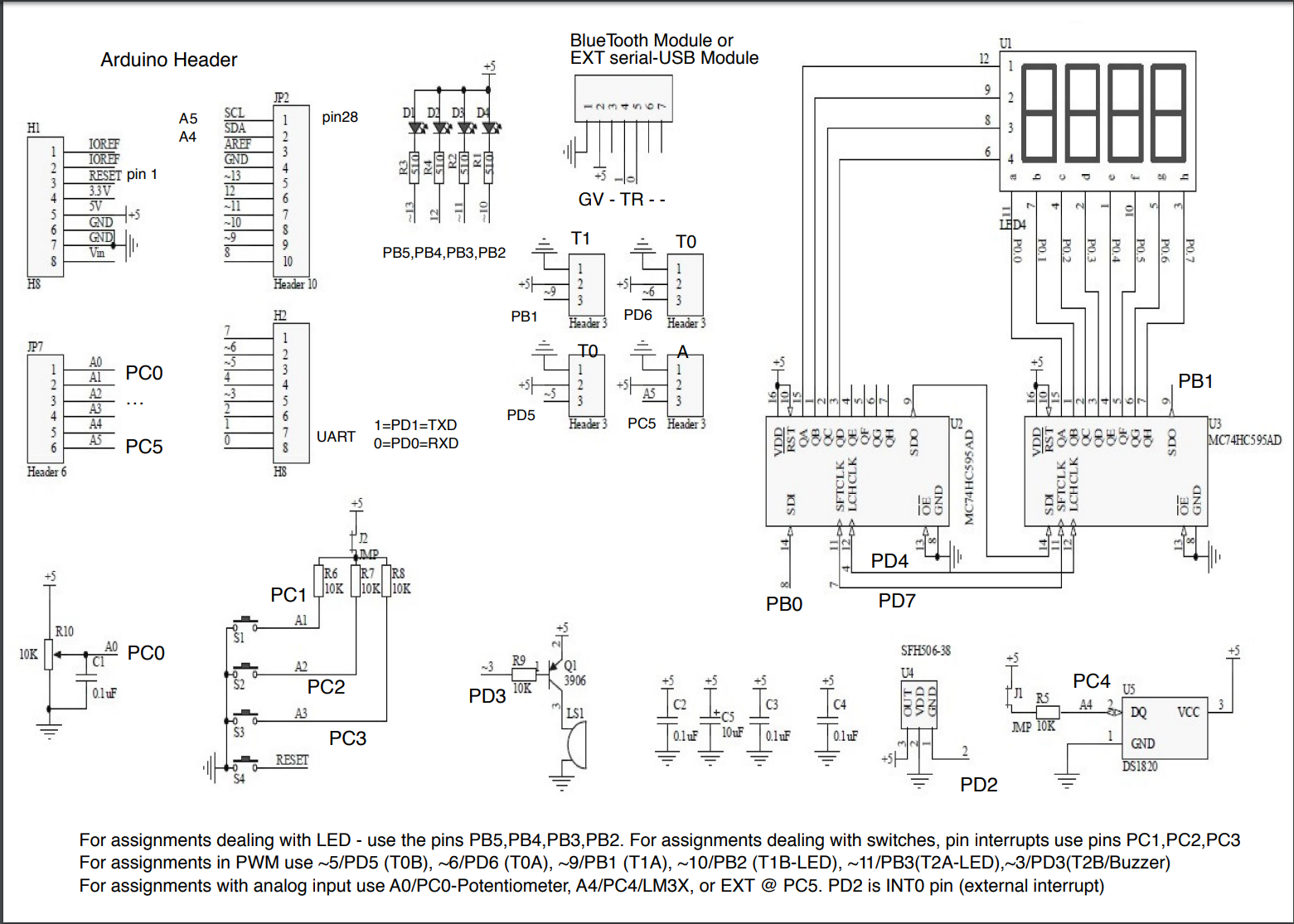
Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

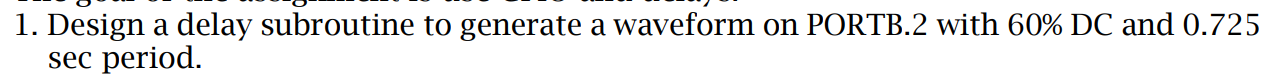
1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

ATMega328P mini

Multi-shield



1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A. ASSEMBLY AND C-CODE:**



ASSEMBLY PORTION:

; CPE301\_DA2\_assembly.asm

;

; Author : Cody

; TASK1

;0.725s period

;60% = 0.435s. TCNT = 6796 = 1A8C

;40% = 0.29s. TCNT = 4530 = 11B2

.org 0

LDI R16, 0b00111100 ;representing PB

OUT DDRB, R16 ;enable PB as output

LDI R20,5 ;set clock prescaler to 1024

STS TCCR1B,R20

begin:

LDI R20,0x00 ;resetting the counter to 0

STS TCNT1H,R20

STS TCNT1L,R20

RCALL delay ;calling timer to wait for 1 sec

LDI R17,0b00111100 ;XOR to toggle led

OUT PORTB,R17 ;output to port b

;RUN SEQUENCE FOR LED OFF

LDI R20,0x00 ;resetting the counter to 0

STS TCNT1H,R20

STS TCNT1L,R20

RCALL delay2 ;calling timer to wait for 1 sec

LDI R17,0b00111000 ;XOR to toggle led

OUT PORTB,R17

RJMP begin ;repeat main loop

delay:

LDS R29, TCNT1H ;loading upper bit of counter to R29

LDS R28, TCNT1L ;loading lower bit of counter to R28

CPI R28,0x8C ;comparing if lower 8 bits of timer is 0x8C

BRSH body ;if lower bits of timer have reached desired amount, check the upperbits

RJMP delay ;otherwise, keep checking lower bits

body:

CPI R29,0x1A ;check to see if upper timer bits have reached the desired value

BRLT delay ;if not, recheck the lower bits

RET ;once the timer reached the desired value, toggle the LED

delay2:

LDS R29, TCNT1H ;loading upper bit of counter to R29

LDS R28, TCNT1L ;loading lower bit of counter to R28

CPI R28,0xB2 ;comparing if lower 8 bits of timer is 0xB2

BRSH body2 ;if lower bits of timer have reached desired amount, check the upperbits

RJMP delay2 ;otherwise, keep checking lower bits

body2:

CPI R29,0x11 ;check to see if upper timer bits have reached the desired value

BRLT delay2 ;if not, recheck the lower bits

RET ;once the timer reached the desired value, toggle the LED

C-CODE PORTION:

/\*

\* CPE301\_DA2\_C\_task1.c

\*

\* Author : Cody

\*/

#include <stdio.h>

#include <avr/io.h>

//0.725s period

//60% = 0.435s. TCNT = 6796 = 1A8C

//40% = 0.29s. TCNT = 4530 = 11B2

int main(void)

{

DDRB |= (1<<2); // Set all PORTB out

TCCR1B = 0x0D; // Set pre-scaler to 1024 and CTC Mode

OCR1A = 0x1A8C; // Set top of OCR1A to 0.435s

int flag = 0; // set flag variables

int flag2 = 1;

// Run loop for 60% DC

while(1){

flag = TIFR1 & 0x04;

if (flag == 0x04 && flag2 == 0){

PORTB &= ~(1<<2); //turn off LED

TIFR1 = 0x04; //reset clear flag

OCR1A = 0x11B2; //set delay for LED off

flag2 = 1;

}

else if(flag== 0x04 && flag2 == 1){

PORTB |= (1<<2); // turn on LED

TIFR1 = 0x04; // reset clear flag

OCR1A = 0x1A8C; // set delay for LED on

flag2 = 0;

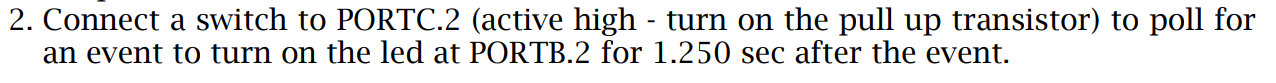
}

}

return 0;

}

1. **DEVELOPED MODIFIED CODE OF TASK 2. ASSEMBLY AND C-CODE:**



ASSEMBLY:

; CPE301\_DA2\_assembly\_task2.asm

;

; Created: 3/2/2019 3:31:09 PM

; Author : Cody

; TASK2

;TCNT for 1.25s = 0x4C4A

.ORG 0x00 ;start memory at 0x00

LDI R20,5 ;set clock pre-scaler to 1024 and normal mode

STS TCCR1B,R20 ;attach value to correct register

CBI DDRC, 2 ;set pin 2 to an input

LDI R16,0b00111100 ;set led pins to outputs

OUT DDRB, R16 ;attach to ddr

LDI R16, 0xFF ;make sure all leds are off

OUT PORTB, R16

AGAIN:

LDI R16, 0b00111100 ;turn led's off

OUT PORTB, R16 ;output to port b

SBIC PINC,2 ;skip next line if switch 2 is pressed

RJMP OVER ;jump to over subroutine

LDI R16, 0b00111000 ;turn on led

OUT PORTB, R16 ;output to port

LDI R20,0x00 ;reset the counter to 0

STS TCNT1H,R20

STS TCNT1L,R20

RCALL DELAY ;go to delay subroutine

RJMP AGAIN

OVER: ;subroutine to loop if button isn't pressed

SBI PORTB,2

RJMP AGAIN

DELAY:

LDS R29, TCNT1H ;loading upper bit of counter to R29

LDS R28, TCNT1L ;loading lower bit of counter to R28

CPI R28,0x4A ;comparing if lower 8 bits of timer is 0x4A

BRSH BODY ;if lower bits of timer have reached desired amount, check the upperbits

RJMP DELAY ;otherwise, keep checking lower bits

BODY:

CPI R29,0x4C ;check to see if upper timer bits have reached the desired value

BRLT DELAY ;if not, recheck the lower bits

RET ;once the timer reached the desired value, toggle the LED

C-CODE:

/\*

\* CPE301\_DA2\_C\_task2.c

\*

\* Created: 3/2/2019 12:04:23 AM

\* Author : Cody

\*/

#define *F\_CPU* 16000000UL //set clock speed to 16MHz

//include important header files

#include <avr/io.h>

#include <stdio.h>

#include <util/delay.h>

int main(void){

//initialize registers

DDRB |= (1<<2); //set portb 2 to output

PORTB |= (1<<2); //set pb2 to high

DDRC &= (0<<2); //set ddrc to input

PORTC |= (0<<2); //set portc2 to low

while (1) {

if (!(PINC & (1 << PINC2))) //check for button press

{

PORTB &= ~(1<<2); //if pressed then turn on led

*\_delay\_ms*(1250); //keep led on for 1.25s

}

else {

PORTB |= (1<<2); //if not pressed then keep led off

}

}

return 0;

}

1. **SCHEMATICS**

N/A

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

**TASK 1/A:**

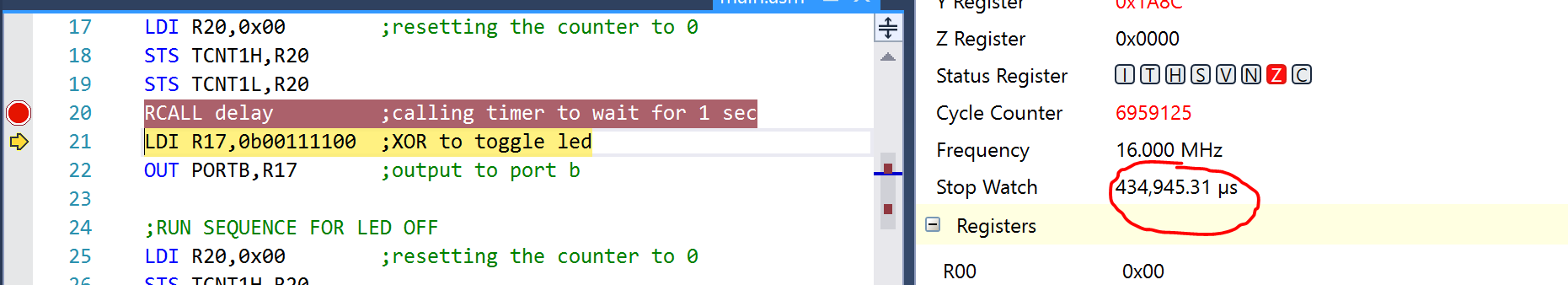
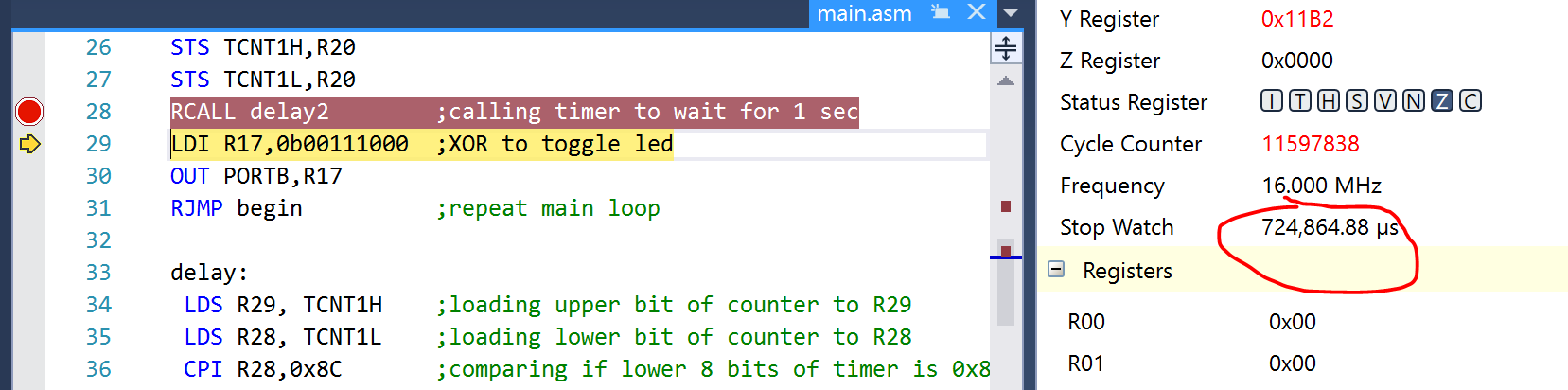


Figure 1 Here is my runtime for the 60% duty cycle delay where the LED is on. This indicates the correct delay of 0.435s

Figure 2 This is my runtime after approaching the second delay block that indicates the 40% duty cycle delay to complete the full 0.725s period

**TASK 1/B:**

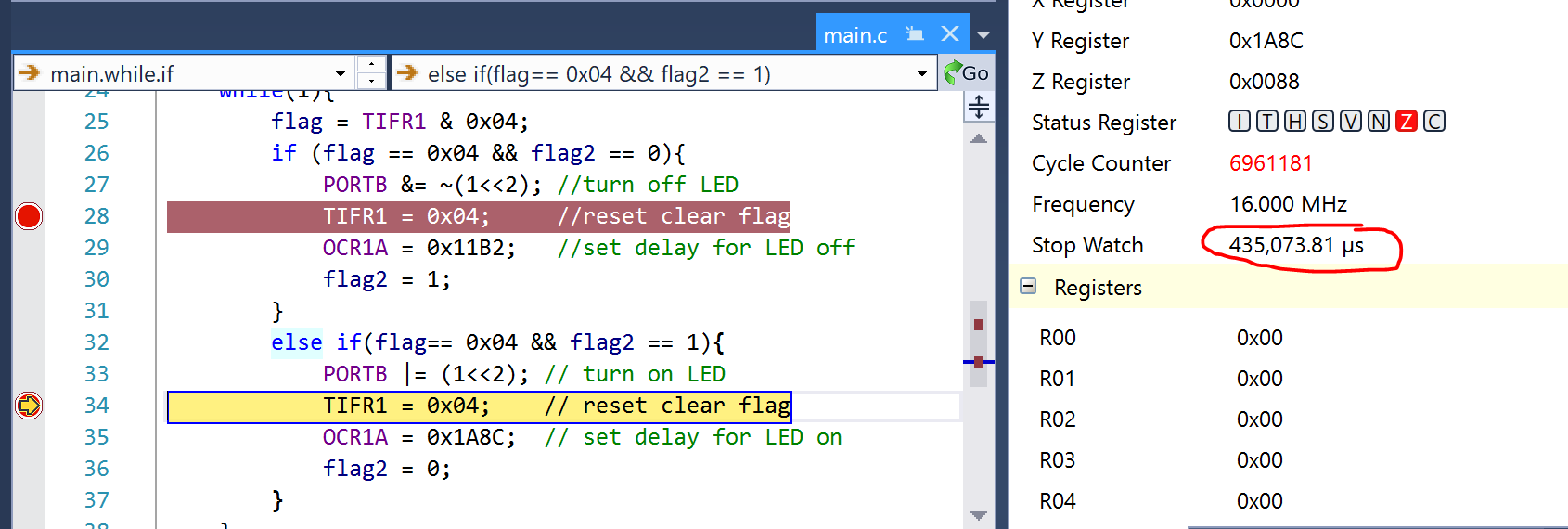


Figure 3 This the runtime after my code completes it's first loop with the LED on. The runtime is 0.435s as expected.

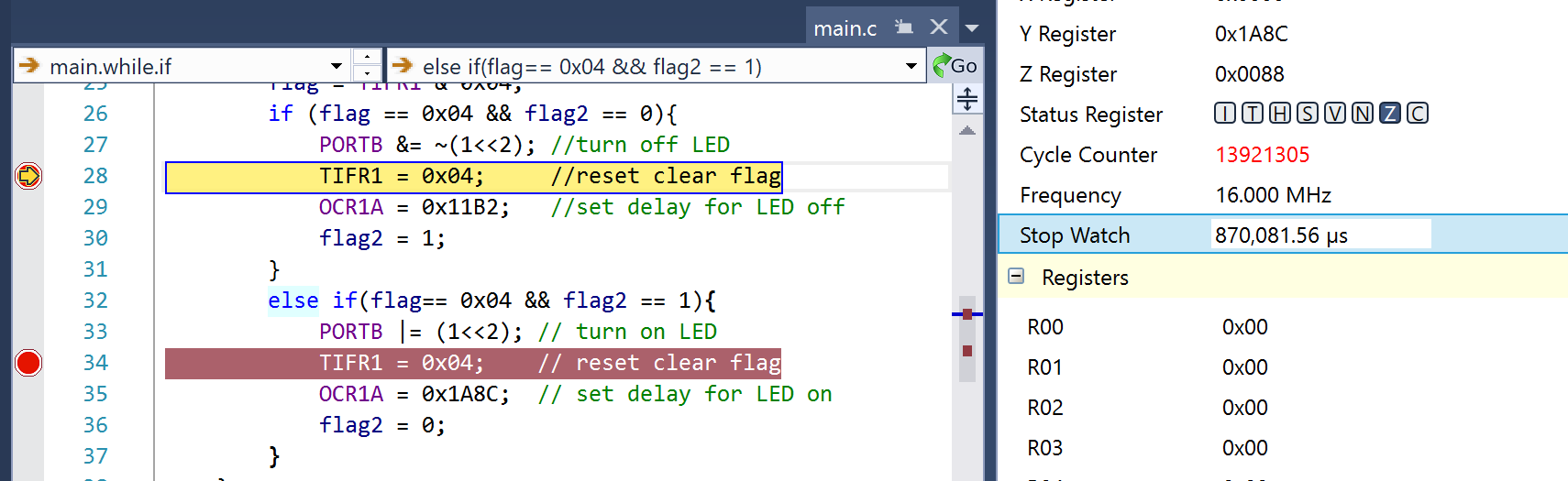


Figure 4 This is my runtime after the second loop. The runtime is 0.87s, which is slightly longer than expected, but I had advanced the steps a couple times to enter the next portion, which may have distorted the time.

**TASK 2/A:**

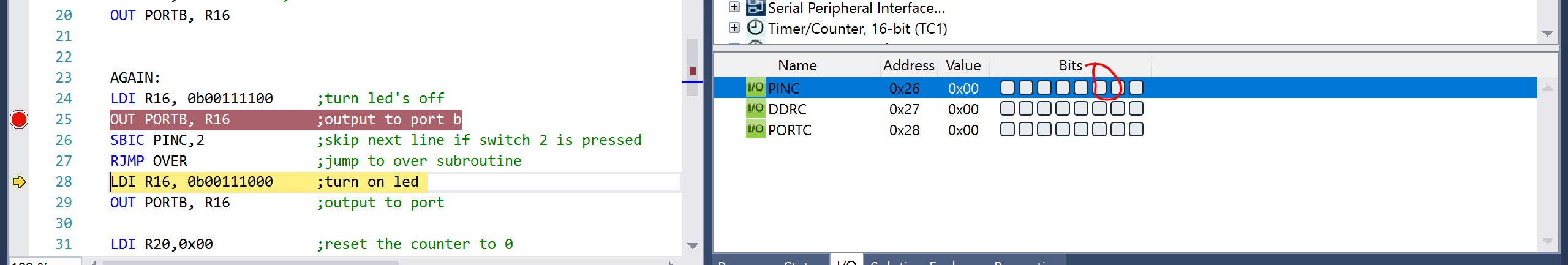


Figure 5: If the button is depressed it will indicate a logic zero and bypass the checker.

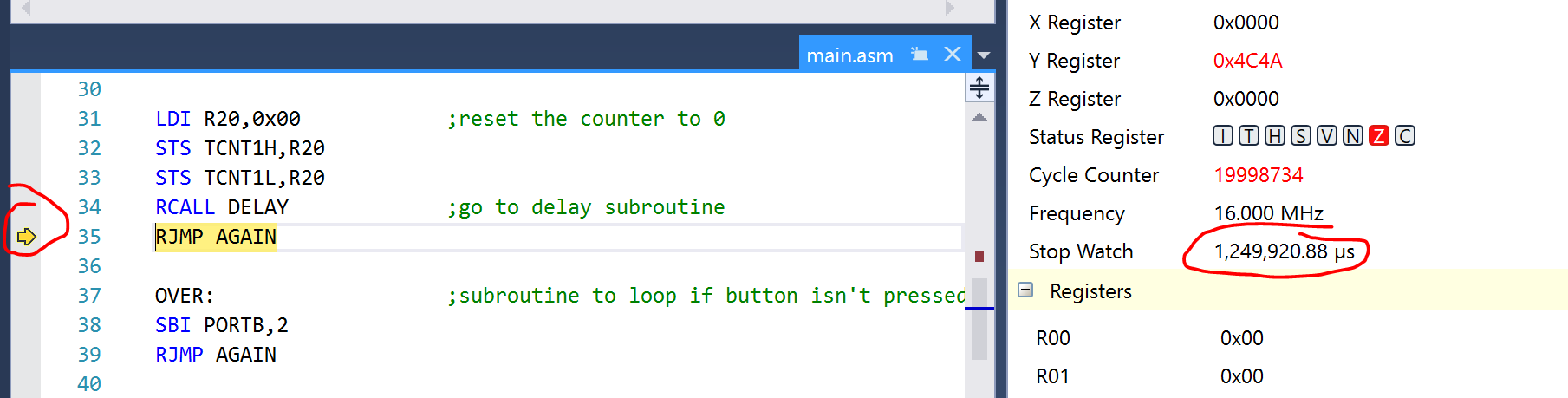


Figure 6: After executing the delay subroutine, the runtime indicates exaclty 1.25s the LED will be on when the button is pressed.

**TASK 2/B:**

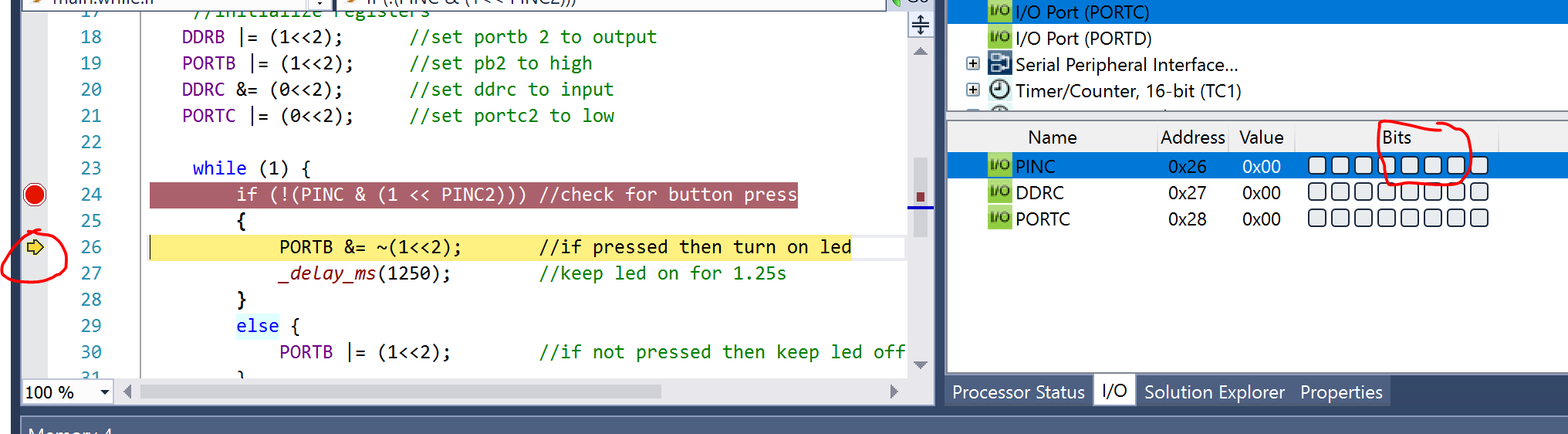


Figure 7: The output indicates a movement towards the next line after the button is pressed

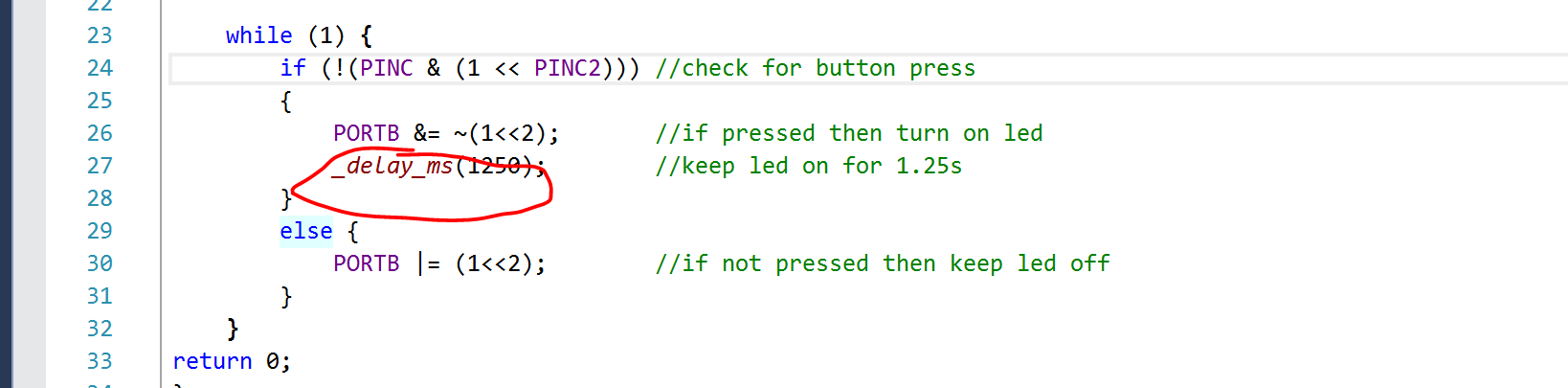
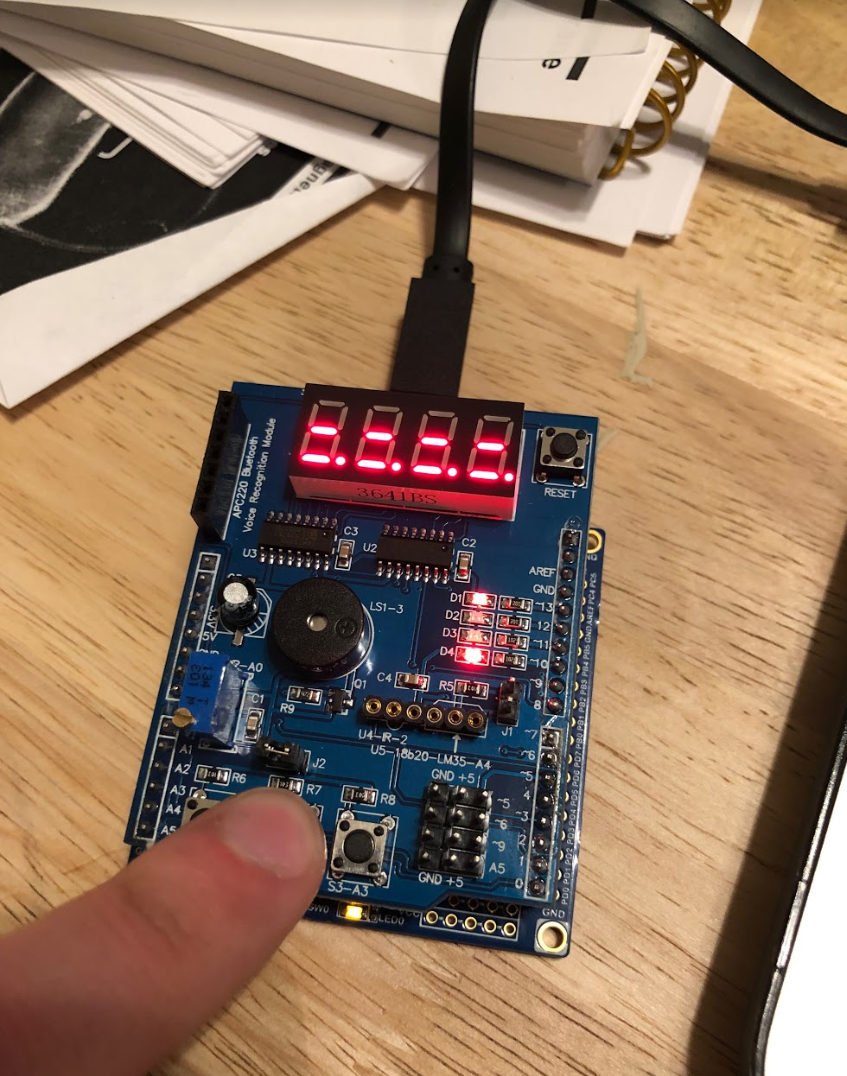
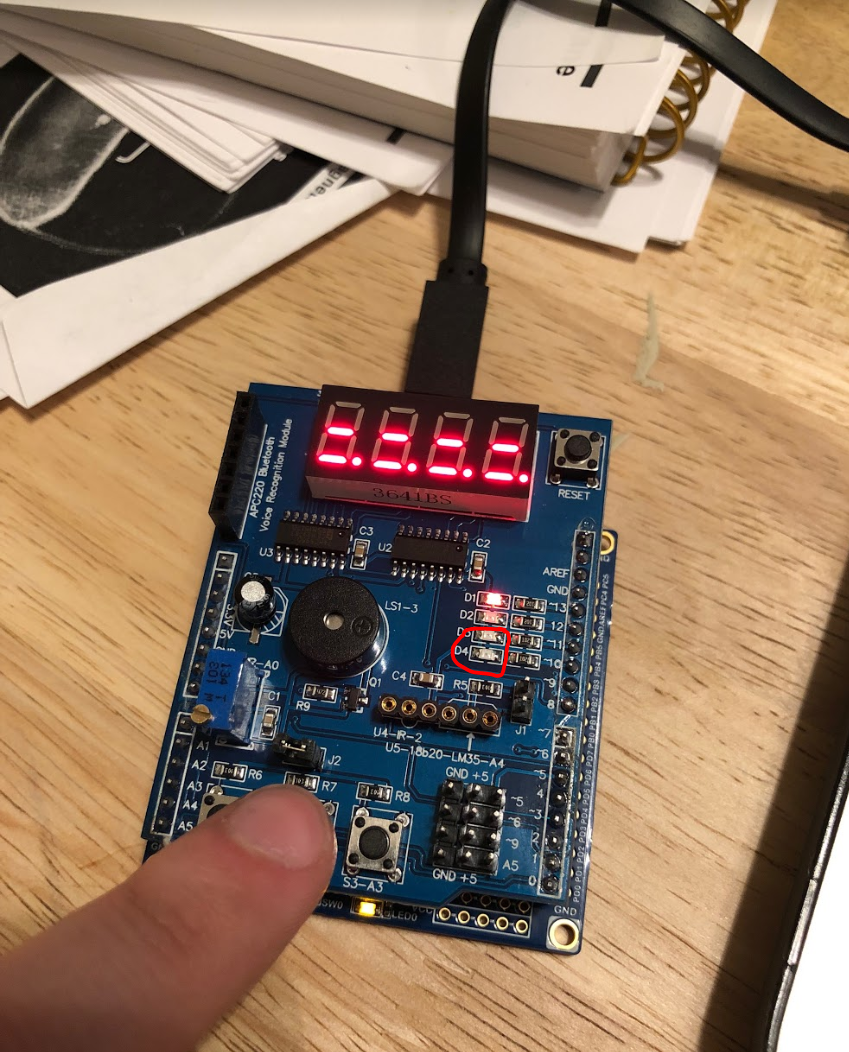


Figure 8:The delay from the delay header file indicating a 1.25s delay.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

**The board setup of each task is the same so only one set of pictures will be listed. The first task just featured the bottom LED blinking**

**Below is the multi-shield sitting on top of the AVR. The two pictures demonstrate the effect of a button press for the second task.**



1. **VIDEO LINKS OF EACH DEMO**

**All 4 demos are featured in the below link.**

https://www.youtube.com/watch?v=PkXHkp8\_aFA&feature=youtu.be

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Cody McDonald