CPE301 – SPRING 2019

Design Assignment 2A

Student Name: Ali Asadi

Student #: 5004200625

Student Email: aliasadivet@gmail.com

Primary Github address: https://github.com/aliasadivet/Submitions\_DA\_AliAsadi

Directory: DessignAssignments

The goal of the assignment is use GPIO and delays

Introduction

Microcontrollers job is taking input , processing it and generating output ! Thus i/o registers and their correct settings is indispensable part of programming any microcontroller.

Registers

In Atmega328P, every port (B, C, D) has 3 registers associated with it each one with 8 bits. Every bit in those registers configure pins of particular port. Bit0 of these registers is associated with Pin0 of the port, Bit1 of these registers is associated with Pin1 of the port, …. and like wise for other bits.

These three registers are as follows :  
(x can be replaced by B,C,D as per the AVR you are using)  
– DDRx register  
– PORTx register  
– PINx register

DDRx register

DDRx (Data Direction Register) configures data direction of port pins. In other words its setting determines whether port pins will be used for input or output. Writing 0 to a bit in DDRx makes corresponding port pin as input, while writing 1 to a bit in DDRx makes corresponding port pin as output.

PINx register

PINx (Port IN) used to read data from port pins. In order to read the data from port pin, first you have to change port’s data direction to input. This is done by setting bits in DDRx to zero. If port is made output, then reading PINx register will give you data that has been output on port pins.

PORTx register

PORTx is used for two purposes.

1) To output data  :  when port is configured as output

2) To activate/deactivate pull up resistors – when port is configures as input

Programming

The following is the assembly code for the delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period as well as the second part of the task. I have explained the delay subroutine in the comments.

Also, since the assignment has two parts, for convenience in execution, I have written both parts’ functions in main loop and Loop loop. To execute each part I simmply change them to the related parts’ functions, and change the other ones to comment.

;

; DA2A.asm.asm

;

; Created: 2/27/2019 1:19:00 AM

; Author : Ali Asadi

;

rjmp main

.org 0x0100

main:

ldi r16, high (RAMEND)

out SPH, r16

ldi r16, low (RAMEND)

out SPL, r16

rcall Pin\_Init\_ASM ;change to comment when executing second part of the task (just for convenience)

;rcall Switch\_Init\_ASM;change to code instruction when executing second part of the task

loop:

rcall Pulse\_ASM;change to comment when executing second part of the task (just for convenience)

;rcall LED\_ASM ;change to code instruction when executing second part of the task

rjmp loop

Pin\_Init\_ASM:

sbi DDRB, 2 ; sbi - set bit I/O - set bit 2 in DDRB register - Output Mode

sbi PORTB, 2 ; cbi - clear bit I/O - clear bit 2 in PORTB register - Output Low

ret ; return from the function

Pulse\_ASM:

ldi r24, 87 ; ldi - load immediate load value of 87 in r24 - 87\*5mS = 435mS

cbi PORTB, 2 ; set bit 2 in PORTB register - Output High

rcall Delay\_ASM ; rcall - relative call , call Delay\_ASM Function

sbi 0x05, 2 ; clear bit 2 in PORTB register - Output Low

ldi r24, 58 ; load value of 58 in r24 - 58\*5mS = 290mS

rcall Delay\_ASM ; call Delay\_ASM Function

ret ; return from the function

; Delay\_ASM generates a delay of 5mS for each loop

; So to generate a delay of 100mS value of 100/5 = 20 is sent via r24 register

; ldi - 1 Cycle

; sbiw - 2 cycle

; brne - 2 cycle

; so sbiw and brne loop till the value in r24 decrements to zero

; for example if r24 = 1 , then those two lines executes 1 time which is 4 cycle

; if r24 = 10, then executes 10 times which is 4 \* 10 = 40 cycle

; frequency is 8Mhz then = each cycle is 1/8Mhz = 1/8uS

; therefore 4Cycle = 4\*(1/8)uS = 0.5uS delay

; to get 5mS = 5000uS , number of cycles = 5000uS / (1/8)uS = 5000 \* 8 cycles

; but this loop already generates delay of 4 cycles

; to take that into account 5000 \* 8 / 4 = 10000 cycles

; there 10000 \* 4 = 40000cycles \* (1/8uS) = 5000 uS

Delay\_ASM:

mov r16, r24 ; move r24 to r16

ldi r24, low(5000 \* 8 / 4) ; load lower byte of the delay value to r24

ldi r25, high(5000 \* 8 / 4) ; load higher byte of the delay value to r25

sbiw r24, 1 ; subtract Immediate from word - r25:r24 - 1

brne PC-1 ; branch if not equal ie branch till result of previous instruction is Zero

dec r16 ; decrement r16

brne PC-5 ; branch till r16 becomes zero

ret ; exit or return

Switch\_Init\_ASM:

cbi DDRD, 2 ; DDRD bit 2 = 0 INPUT Mode

sbi PORTD, 2 ; PORTD bit 2 = 1 PULLUP Enable

sbi DDRB, 2 ; PORTB.2 - Output

sbi PORTB, 2 ; PORTB.2 - Output Low

ret ; return from the function

LED\_ASM:

sbis PIND, 2 ; skip if bit is set in I/O register, skip if HIGH state

rjmp pressed ; relative jump - jump to pressed label if State is low

brtc return ; branch if T bit is cleared - check the previous state

ldi r24, 2 ; load r24 with 2 to generate a delay of 10mS

rcall Delay\_ASM ; call Delay\_ASM to generate debounce delay

sbis PIND, 2 ; check if the bit stays the same

rjmp return ; return if not

cbi PORTB, 2 ; PORTB.2 - Output High

ldi r24, 250 ; load r24 with 250 to generate a delay of 1250mS

rcall Delay\_ASM ; call delay

sbi PORTB, 2 ; PORTB.2 - Output Low

clt ; clear T , previous state is high

rjmp return ; return

pressed:

ldi r24, 2 ; if state is low, then generate a debounce delay of 10mS

rcall Delay\_ASM ; call delay

sbis PIND, 2 ; check if the bit stays the same

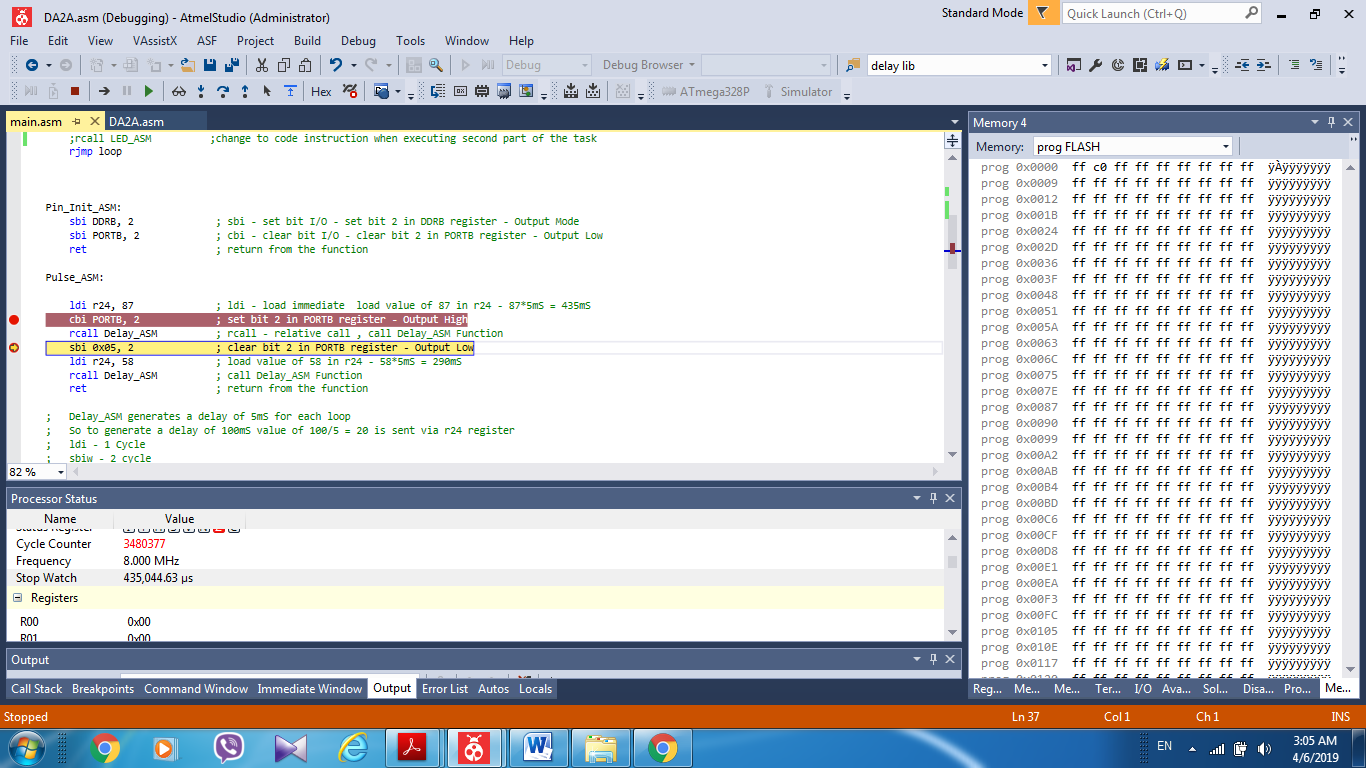
set ; set T bit to indicate previous state is Low

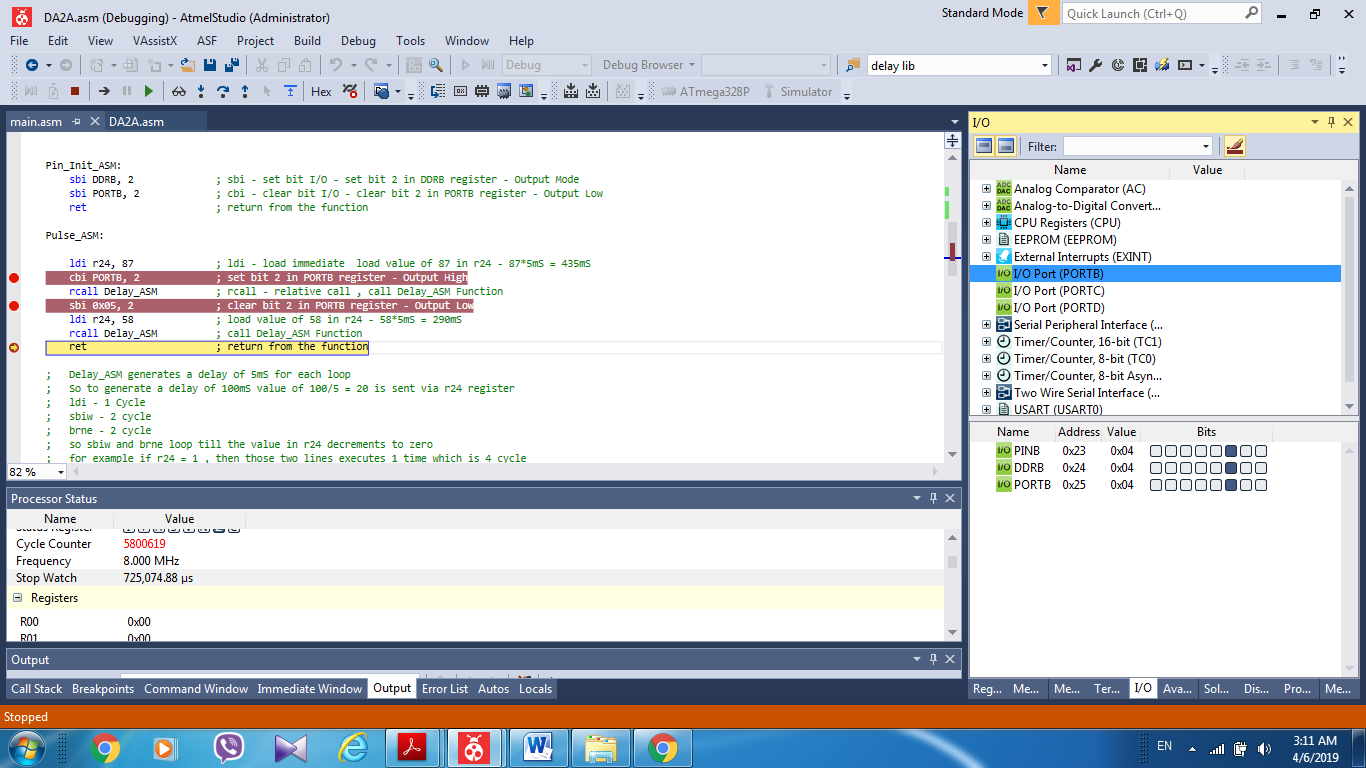
return:

ret ; return from the function

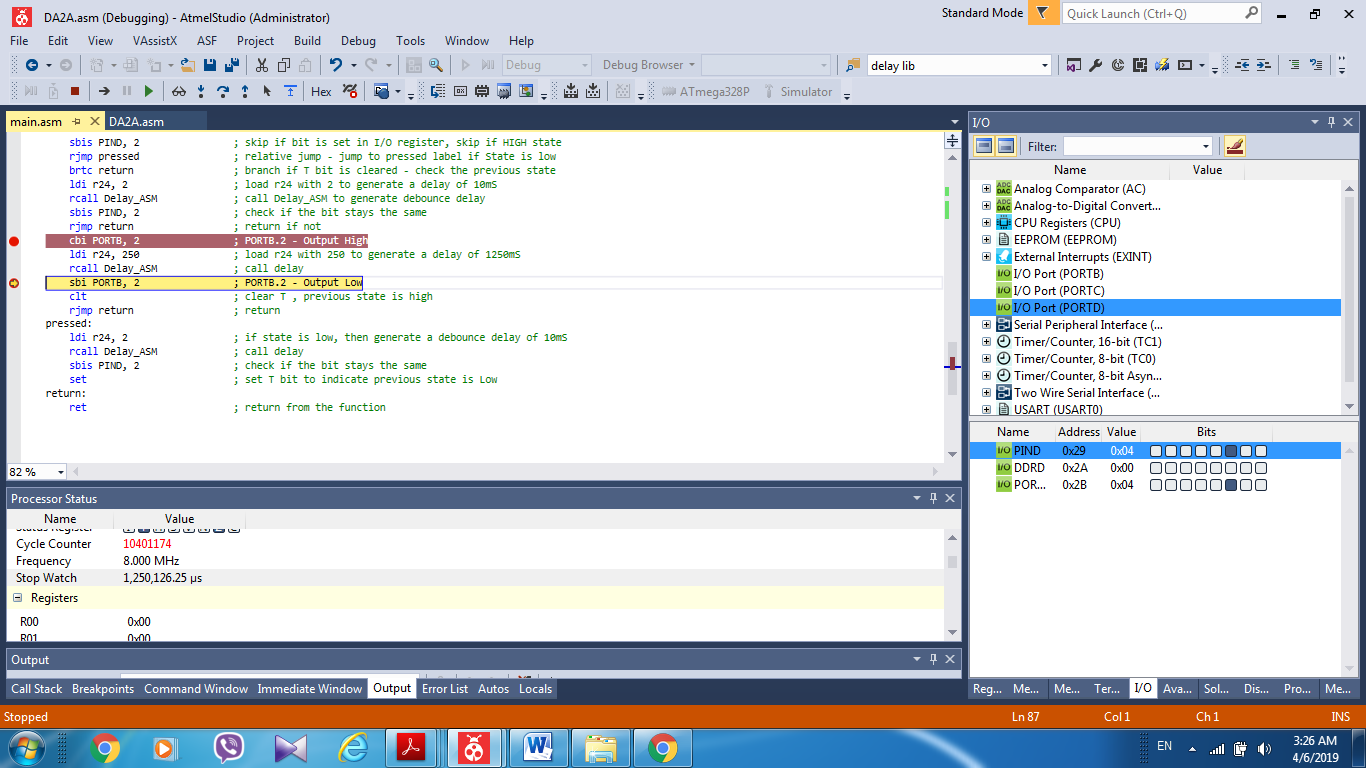
The following screenshots demonstrate the %60 of the period (435ms) and whole period (725 ms) respectively.

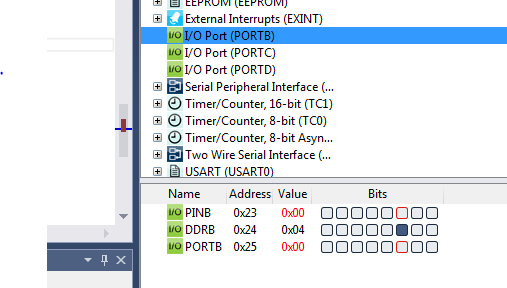
(part 1 of the assignment)





The following screenshot demonstrates the simulation result of the second part of the assignment (The LED connected to port B is turned on for 1.25 sec when the switch connected to port D is pressed).





The following is the C code for the delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period as well as the second part of the task.

/\*

\* DA2A.c

\*

\* Created: 2/27/2019 4:54:25 AM

\* Author : Ali Asadi

\*/

#include <avr/io.h>

#include <inttypes.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/sleep.h>

#include <util/delay.h>

void Pin\_Init()

{

DDRB |= \_BV(2); // PORTB Pin 2 Output if bit in DDRx register is set

// then corresponding Pin will be output

PORTB |= \_BV(2); // Set Pin in PortB High if bit in PORTx is Set then corresponding Pin is HIGH

} // \_BV(x) = (1<<x) ; \_Bv(1) = 1<<1 = 00000010

void Pulse()

{

PORTB &= ~\_BV(2); // clear the bit in PORTb , to set the output low

*\_delay\_ms*(435); // delay for 435 ms, built-in function

PORTB |= \_BV(2); // Set Pin in PortB High if bit in PORTx is Set then corresponding Pin is HIGH

*\_delay\_ms*(290); // delay for 290 ms , total of 725mS

}

void Switch\_Init()

{

DDRD &= ~\_BV(2); // Clearing DDRx register set that pin to INPUT mode

PORTD |= \_BV(2); // setting PORTx, for an INPUT pin , enables the pull-up resistor

DDRB |= \_BV(2); // PORTB pin 2 output

PORTB |= \_BV(2); // Set Pin in PortB High if bit in PORTx is Set then corresponding Pin is HIGH

}

void LED()

{

static *uint8\_t* pressed = 0; // variable to store previous value of the PIN

if((PIND & \_BV(2)) && pressed) // if PINx register is used to read the voltage level on that Pin

{

*\_delay\_ms*(10); // if current State is high, and previous state is low , delay 10ms

if(PIND & \_BV(2)) // for Debonuce

{ // if the pin maintains same state as before than Pin stable

PORTB &= ~\_BV(2); // Clear the PORTD pin 2

*\_delay\_ms*(1250); // delay 1.250 Secs

PORTB |= \_BV(2); // Set the PORTD pin 2

}

pressed = 0; // Previous state value to high

}

else if(!(PIND & \_BV(2))) // if current state is LOW

{

*\_delay\_ms*(10); // debounce delay

if(!(PIND & \_BV(2))) // still low

pressed = 1; // previous value is LOW

}

}

void main()

{

Pin\_Init(); //change to comment when executing second part of the task (just for convenience)

//Switch\_Init(); //change to code instruction when executing second part of the task

while (1)

{

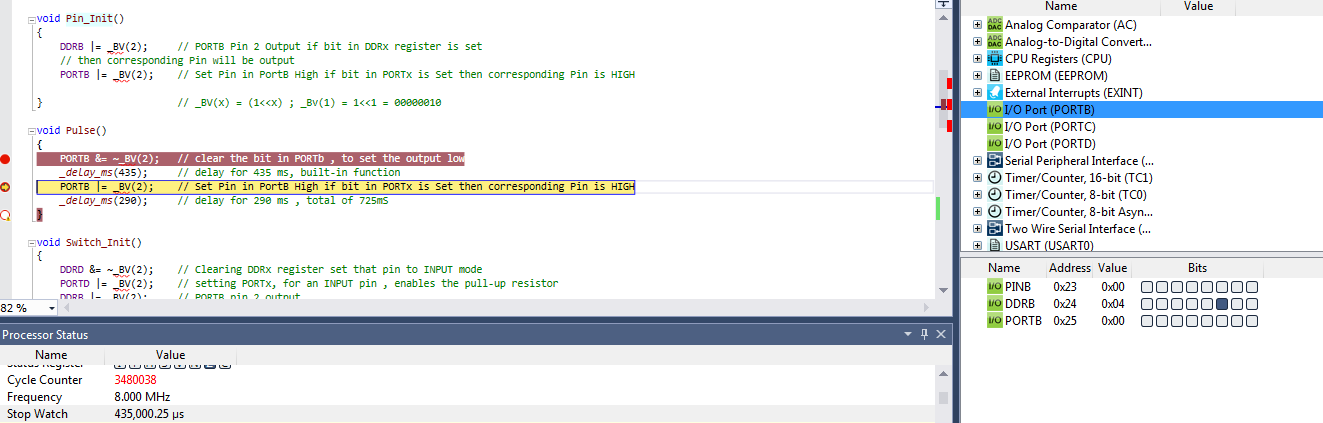
Pulse(); //change to comment when executing second part of the task (just for convenience)

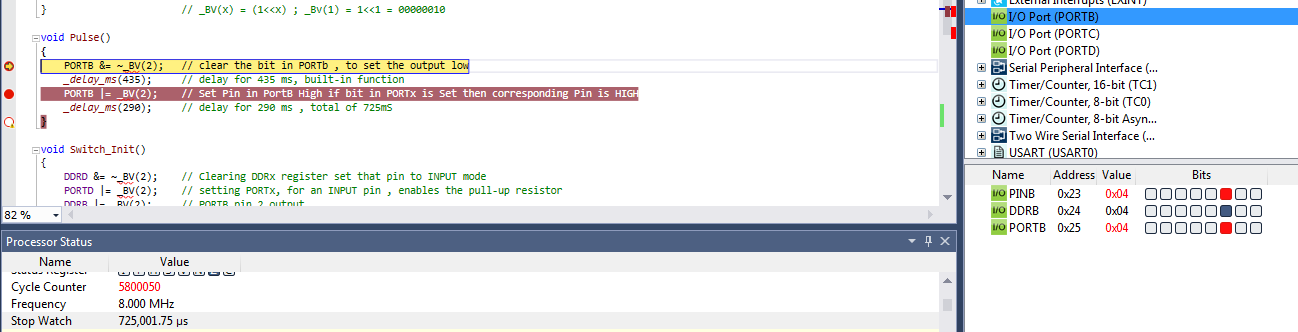
//LED(); //change to code instruction when executing second part of the task

}

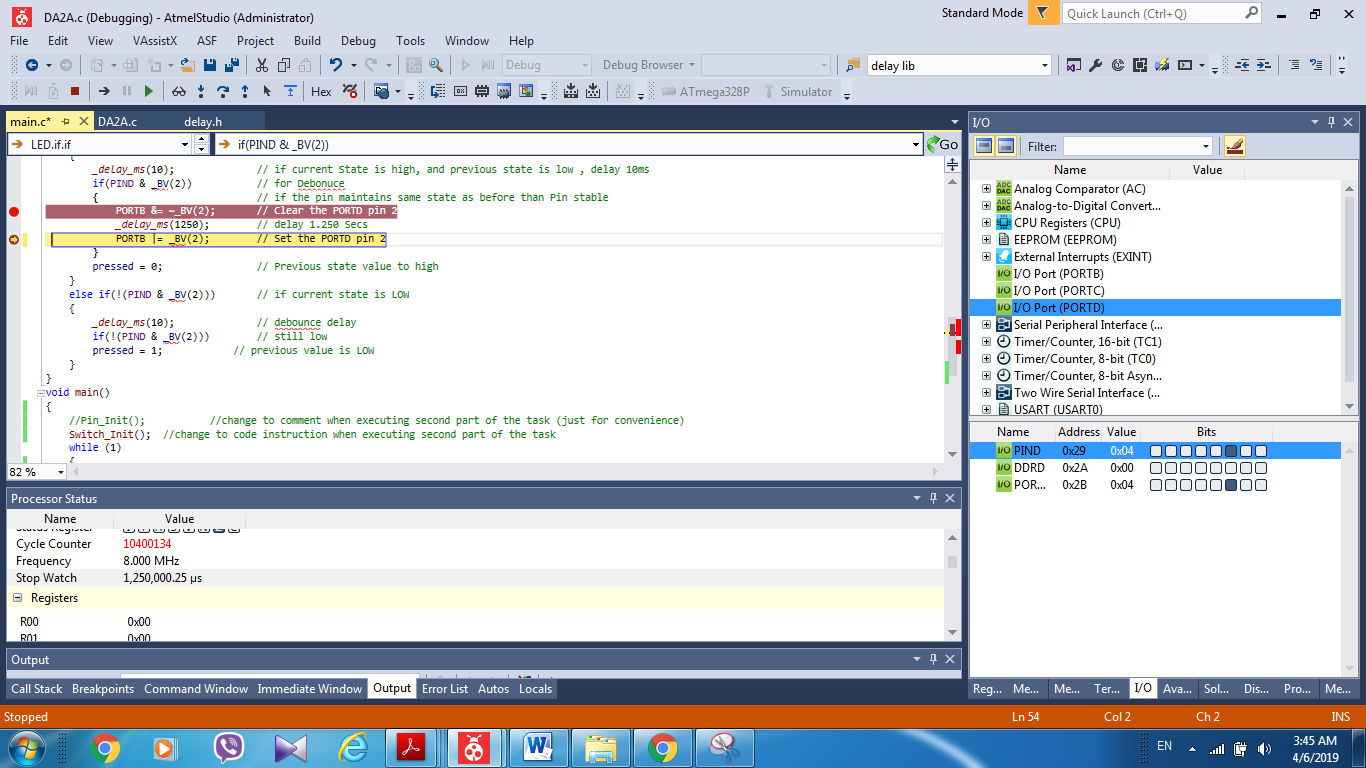
}

As it is shown in the screenshots below, the full period is 725 ms and %60 DC is 435 ms.





The following screenshot demonstrates the simulation result of the second part of the task in C code.



The simulation and emulation tesults are posted on YouTube, and can be found on linkes below:

<https://www.youtube.com/watch?v=06ImNqBi7Lg>

<https://www.youtube.com/watch?v=CDTJhhKZ8to>