ID#1007139171 Monty Sourjah

CPE301 – SPRING 2018

Design Assignment 2

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS | Y |  |
| 2. | INITIAL CODE OF TASK 1/A | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | Y |  |
| 4. | SCHEMATICS | Y |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 5. | SCREENSHOT OF EACH DEMO | Y |  |
| 6. | VIDEO LINKS OF EACH DEMO | Y |  |
| 7. | GOOGLECODE LINK OF THE DA | Y |  |
|  |  |  |  |
|  |  |  |  |

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

List of Components used

1.Polulo AVR ISP Programmer.

2.Polulo Scope.

3.Atmel Xplained Mini

Block diagram with pins used in the Atmega328P

Part (a)

PIN B PORT B Polulo Scope



Atmel 328p

Pololu AVR ISP Programmer

ISP

USB Connection from Pololu to computer

1. **INITIAL/DEVELOPED CODE OF TASK 1/A**

;

; SimulationCPE301\_Assignment2\_Task1.asm

;

; Created: 2/26/2018 8:35:01 PM

; Author : Monty Sourjah

;

Assembly Code:

.INCLUDE "M328pDEF.INC"

start:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 as output

out DDRB, r20 ; Set PIN 2 on port B to output and all others to input

loop:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 to 1

out PORTB, r20 ; Set pin 2 to 1 on port B

call delay ; wait for 250ms

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 to 0

out PORTB, r20 ; Set pin 2 to 0 on port B

call delay ; wait for 250ms

rjmp loop ; repeat the loop

delay:

; By default on Xplained mini is extenal clock

; On 5V it is at 16MHz

; On 3.3V it is 8MHz

; We have to do a 250ms delay

; On 16MHz 1 clock is 1/16MHz = 62.5ns

; It takes 16 clocks for 1us, if there is a loop which will take 16 clocks (1us) executed 250 times it is 250us.

; Doing this a 250 times, it will elapse 62.5ms

; And doing this 4 times, we will have our 250ms of delay

; There will be some inaccurancy due to setting the values after end of loops.

ldi r16, 4 ; load 4 to r16 - 1 clock

cycle1: ; 4 \* 250 \* 250us loop = 250ms

ldi r17, 250 ; load 250 to r17 - 1 clock

cycle2: ; 250 \* 250us loop = 62.5ms

ldi r18, 250 ; load 250 to r18 - 1 clock

cycle3: ; 250us loop

dec r18 ; decement r18 - 1clock

nop ; no operation, just spend a clock cycle - 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

brne cycle3 ; branch if r18 not 0 - 2 clocks, 1 if no branching

; in total 16 clocks or 1 us

; it will execute 250 times, so it will spend 250us

dec r17 ; decrement r17 - 1 clock

brne cycle2 ; branch if r17 not 0 - 2 clocks, 1 if no branching

; it will execute 250 times, so it will spend 250 \* 250us = 62.5ms

dec r16 ; decrement r16 - 1 clock

brne cycle1 ; branch if r16 not 0 - 2 clocks, 1 if no branching

; it will execute 4 times, so it will spend 4 \* 62.5ms = 250ms

ret ; return

C Code:

#define *F\_CPU* 16000000

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = 0b00000100;

PORTB = 0b00000100;

while (1)

{

*\_delay\_ms*(250);

PORTB = 0b00000000;

*\_delay\_ms*(250);

PORTB = 0b00000100;

}

}

1. **MODIFIED CODE OF TASK 2/A from TASK 1/A**

start:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 as output

out DDRB, r20 ; Set PIN 2 on port B to output and all others to input

out PORTD, r20 ; Enable pull up resistors on port D

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 as input

out DDRD, r20 ; Set port D pins to input

loop:

; debouncing is not considered

in r20, PIND ; load port d pins to r20

andi r20, 0b00000100 ; clear all bits in r20 except pin 2

breq loop ; repeat the loop

out PORTB, r20 ; Set pin 2 to 1 on port B

call delay ; wait for 250ms

call delay ; wait for 250ms

call delay ; wait for 250ms

call delay ; wait for 250ms

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 to 0

out PORTB, r20 ; Set pin 2 to 0 on port B

rjmp loop ; repeat the loop

delay:

; By default on Xplained mini is extenal clock

; On 5V it is at 16MHz

; On 3.3V it is 8MHz

; We have to do a 250ms delay

; On 16MHz 1 clock is 1/16MHz = 62.5ns

; It takes 16 clocks for 1us, if there is a loop which will take 16 clocks (1us) executed 250 times it is 250us.

; Doing this a 250 times, it will elapse 62.5ms

; And doing this 4 times, we will have our 250ms of delay

; There will be some inaccurancy due to setting the values after end of loops.

ldi r16, 4 ; load 4 to r16 - 1 clock

cycle1: ; 4 \* 250 \* 250us loop = 250ms

ldi r17, 250 ; load 250 to r17 - 1 clock

cycle2: ; 250 \* 250us loop = 62.5ms

ldi r18, 250 ; load 250 to r18 - 1 clock

cycle3: ; 250us loop

dec r18 ; decement r18 - 1clock

nop ; no operation, just spend a clock cycle - 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

nop ; 1clock

brne cycle3 ; branch if r18 not 0 - 2 clocks, 1 if no branching

; in total 16 clocks or 1 us

; it will execute 250 times, so it will spend 250us

dec r17 ; decrement r17 - 1 clock

brne cycle2 ; branch if r17 not 0 - 2 clocks, 1 if no branching

; it will execute 250 times, so it will spend 250 \* 250us = 62.5ms

dec r16 ; decrement r16 - 1 clock

brne cycle1 ; branch if r16 not 0 - 2 clocks, 1 if no branching

; it will execute 4 times, so it will spend 4 \* 62.5ms = 250ms

ret ; return

C code:

#define *F\_CPU* 16000000 //Clock frequency

#include <avr/io.h> // io header file

#include <util/delay.h> // delay header file

int main(void)

{

/\* Replace with your application code \*/

DDRB = 0b00000100; // set PORTB.2 to output

PORTD = 0; // Enable pull up resistors on port D

DDRD = 0b00000000; // Set port D to all inputs

while (1)

{

if (PIND >> 2) // if PIND2 is 1

{

PORTB = 0b00000100; // SET PORTB2 to 1

*\_delay\_ms*(1000); // wait 1s

PORTB = 0; // Set PORTB2 to 0

}

}

}

**MODIFIED CODE OF TASK 3**

;

start:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 as output

out DDRB, r20 ; Set PIN 2 on port B to output and all others to input

loop:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 to 1

out PORTB, r20 ; Set pin 2 to 1 on port B

call delay ; wait for 250ms

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 to 0

out PORTB, r20 ; Set pin 2 to 0 on port B

call delay ; wait for 250ms

rjmp loop ; repeat the loop

delay:

; By default on Xplained mini is extenal clock

; On 5V it is at 16MHz

; On 3.3V it is 8MHz

; We have to do a 250ms delay

; On 16MHz 1 clock is 1/16MHz = 62.5ns

; It takes 16 clocks for 1us, a 8 bit counter counts 256 times for overflow, so there will be overflow on 256 / 16 = 16 us.

; To get 250ms, we need 250000 / 16 = 15625 overflows

; We can do 15625 as 125 \* 125 overflows

; There will be some inaccurancy due to setting of the values.

ldi r18, 0b00000001 ; load 1 to r18

out TCCR0B, r18 ; start timer 0

ldi r16, 125 ; load 125 to r16

cycle1: ; 125 \* 125 \* 16us loop = 250ms

ldi r17, 125 ; load 250 to r17 - 1 clock

cycle2: ; 125 \* 16us loop = 2ms

in r18, TIFR0 ; load TIFR0 register int r18

sbrs r18, TOV0 ; Check TOV0 bit (timer 0 overflow occured), skip next line if set

jmp cycle2;

ldi r18, 1 ; load 1 into r18

out TIFR0, r18; clear TOV0 flag

dec r17 ; decrement r17

brne cycle2 ; branch if r17 not 0

; it will execute 125 times, so it will spend 125 \* 16us = 2ms

dec r16 ; decrement r16

brne cycle1 ; branch if r16 not 0

; it will execute 125 times, so it will spend 125 \* 2ms = 250ms

ldi r18, 0b00000000 ; load 250 to r18 - 1 clock

out TCCR0B, r18 ; stop TIMER0

ret ; return

C code

#define *F\_CPU* 16000000 // CPU clock

#define const2ms 125

#include <avr/io.h>

void delay(int d)

{

int i = 0; // counter variable

TCCR0B = 1; //start TIMER0

TCNT0 = 0; // set TIMER0 counter to 0

while (i < d \* const2ms / 2) // check if time elapsed

{

if (TIFR0 && 1) // check if timer0 overflow

{

TIFR0 |= 7; // clear overflow bit

i++; // increment counter

}

}

}

int main(void)

{

/\* Replace with your application code \*/

DDRB = 0b00000100; // set PORTB2 to output

while (1)

{

delay(250); // delay 250ms

PORTB = 0b00000100; // Set PORTB2 to 1

delay(250); // delay 250ms

PORTB = 0; // Set PORTB2 to 0

}

}

**MODIFIED CODE OF TASK 4**

.INCLUDE "M328pDEF.INC"

.org 0x000

rjmp start

.org 0x020

rjmp timer0\_ovf

timer0\_ovf:

dec r17

reti

start:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 as output

out DDRB, r20 ; Set PIN 2 on port B to output and all others to input

ldi r20, 1 << TOIE0

sts TIMSK0, r20

sei

loop:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 to 1

out PORTB, r20 ; Set pin 2 to 1 on port B

call delay ; wait for 250ms

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 to 0

out PORTB, r20 ; Set pin 2 to 0 on port B

call delay ; wait for 250ms

rjmp loop ; repeat the loop

delay:

; By default on Xplained mini is extenal clock

; On 5V it is at 16MHz

; On 3.3V it is 8MHz

; We have to do a 250ms delay

; On 16MHz 1 clock is 1/16MHz = 62.5ns

; It takes 16 clocks for 1us, a 8 bit counter counts 256 times for overflow, so there will be overflow on 256 / 16 = 16 us.

; To get 250ms, we need 250000 / 16 = 15625 overflows

; We can do 15625 as 125 \* 125 overflows

; There will be some inaccurancy due to setting of the values.

ldi r18, 0b00000001 ; load 1 to r18

out TCCR0B, r18 ; start timer 0

ldi r18, 0

out TCNT0, r18

ldi r16, 125 ; load 125 to r16

cycle1: ; 125 \* 125 \* 16us loop = 250ms

ldi r17, 126 ; load 126 to r17 - 1 clock

dec r17 ; to set the flags

cycle2: ; 125 \* 16us loop = 2ms

brne cycle2 ; branch if r17 not 0

; it will execute 125 times, so it will spend 125 \* 16us = 2ms

dec r16 ; decrement r16

brne cycle1 ; branch if r16 not 0

; it will execute 125 times, so it will spend 125 \* 2ms = 250ms

ldi r18, 0b00000000 ; load 250 to r18 - 1 clock

out TCCR0B, r18 ; stop TIMER0

ret ; return

C code:

#define *F\_CPU* 16000000 // clock frequency

#define const2ms 125 // constant 2 ms

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int counter; // volatile global variable

ISR(TIMER0\_OVF\_vect) // interrupt function

{

counter--; // decrement the counter

}

void delay(int d) // makes a delay of d ms

{

int i = 0; // internal counter

TCNT0 = 0; // set timer0 counter to 0

TCCR0B = 1; // start timer 0

while (i < d \* const2ms / 2) // check if time elapsed

{

if (counter == 0) // check if counter is 0

{

counter = 1; // set counter to 1

i++; // increment internal counter

}

}

TCCR0B = 0; // stop timer0

}

int main(void)

{

/\* Replace with your application code \*/

DDRB = 0b00000100; // set PORTB2 to output

TIMSK0 = (1 << TOIE0); // enable timer 0 interrupt

counter = 1; // set counter to 1

sei(); // enable interrupts

while (1)

{

PORTB = 0b00000100; // set PORTB2 to 1

delay(250); // wait 250ms

PORTB = 0; // set PORTB2 to 0

delay(250); // wait 250ms

}

}

**MODIFIED CODE OF TASK 5**

.org 0x0000

rjmp start

.org 0x0002

rjmp int0\_int

.org 0x0020

rjmp timer0\_ovf

int0\_int:

dec r19

reti

timer0\_ovf:

dec r17

reti

start:

ldi r20, 0b00000100 ; load to r20 the mask to set pin 2 as output

out DDRB, r20 ; Set PIN 2 on port B to output and all others to input

out PORTD, r20 ; Enable pull up resistors on port D

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 as input

out DDRD, r20 ; Set port D pins to input

ldi r20, 3

sts EICRA, r20 ; rising edge

ldi r20, 1 << INT0

out EIMSK, r20 ; enable INT0

ldi r20, 1 << TOIE0

sts TIMSK0, r20 ; enable TIMER0 oveflow

sei

preloop:

ldi r19, 1 ; 1 not pressed

loop:

; debouncing is not considered

cpi r19, 0 ; compare with 0;

brne loop ; repeat the loop

ldi r20, 0b00000100

out PORTB, r20 ; Set pin 2 to 1 on port B

call delay ; wait for 250ms

call delay ; wait for 250ms

call delay ; wait for 250ms

call delay ; wait for 250ms

ldi r20, 0b00000000 ; load to r20 the mask to set pin 2 to 0

out PORTB, r20 ; Set pin 2 to 0 on port B

rjmp preloop

delay:

; By default on Xplained mini is extenal clock

; On 5V it is at 16MHz

; On 3.3V it is 8MHz

; We have to do a 250ms delay

; On 16MHz 1 clock is 1/16MHz = 62.5ns

; It takes 16 clocks for 1us, a 8 bit counter counts 256 times for overflow, so there will be overflow on 256 / 16 = 16 us.

; To get 250ms, we need 250000 / 16 = 15625 overflows

; We can do 15625 as 125 \* 125 overflows

; There will be some inaccurancy due to setting of the values.

ldi r18, 0b00000001 ; load 1 to r18

out TCCR0B, r18 ; start timer 0

ldi r18, 0

out TCNT0, r18

ldi r16, 125 ; load 125 to r16

cycle1: ; 125 \* 125 \* 16us loop = 250ms

ldi r17, 126 ; load 126 to r17 - 1 clock

dec r17 ; to set the flags

cycle2: ; 125 \* 16us loop = 2ms

brne cycle2 ; branch if r17 not 0

; it will execute 125 times, so it will spend 125 \* 16us = 2ms

dec r16 ; decrement r16

brne cycle1 ; branch if r16 not 0

; it will execute 125 times, so it will spend 125 \* 2ms = 250ms

ldi r18, 0b00000000 ; load 250 to r18 - 1 clock

out TCCR0B, r18 ; stop TIMER0

ret ; return

C code:

#define *F\_CPU* 16000000 // clock frequency

#define const2ms 125 // constant 2 ms

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int counter; // volatile global variable

volatile int button; // volatile global variable

ISR(TIMER0\_OVF\_vect) // interrupt function

{

counter--; // decrement the counter

}

ISR(INT0\_vect)

{

button = 1; // set button to 1

}

void delay(int d) // makes a delay of d ms

{

unsigned int i = 0; // internal counter

unsigned int j = d \* const2ms / 2;

TCNT0 = 0; // set timer0 counter to 0

TCCR0B = 1; // start timer 0

while (i < j) // check if time elapsed

{

if (counter == 0) // check if counter is 0

{

counter = 1; // set counter to 1

i++; // increment internal counter

}

}

TCCR0B = 0; // stop timer0

}

int main(void)

{

/\* Replace with your application code \*/

DDRB = 0b00000100; // set PORTB2 to output

PORTD = 0b00000100; // Enable pull up on PORTD2

PIND = 0; // SET all PINS on PORTD to input

TIMSK0 = (1 << TOIE0); // enable timer 0 interrupt

EICRA |= 3; // set rising edge

EIMSK |= 1; // enable INT0 interrupt

counter = 1; // set counter to 1

button = 0; // set button to 0

sei(); // enable interrupts

while (1)

{

while (button == 0);

button = 0; // set button to 0

PORTB = 0b00000100; // set PORTB2 to 1

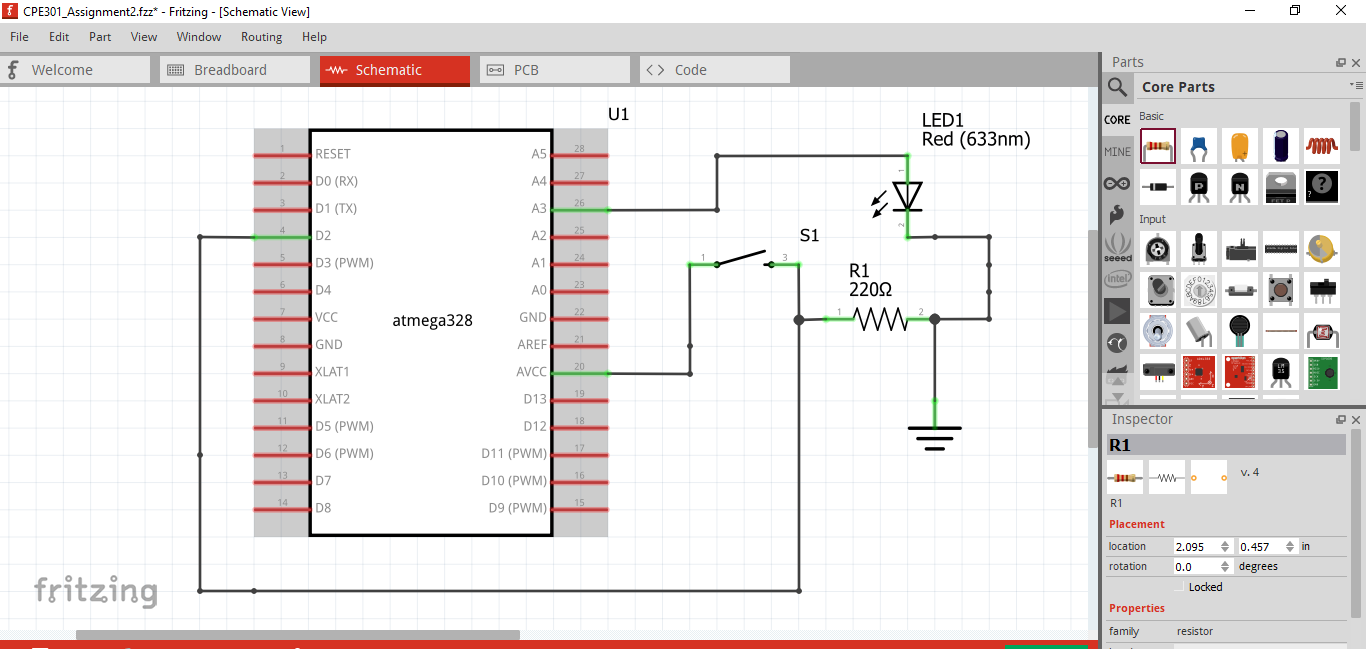
delay(1000); // wait 1s

PORTB = 0; // set PORTB2 to 0

}

}

1. **SCHEMATICS**



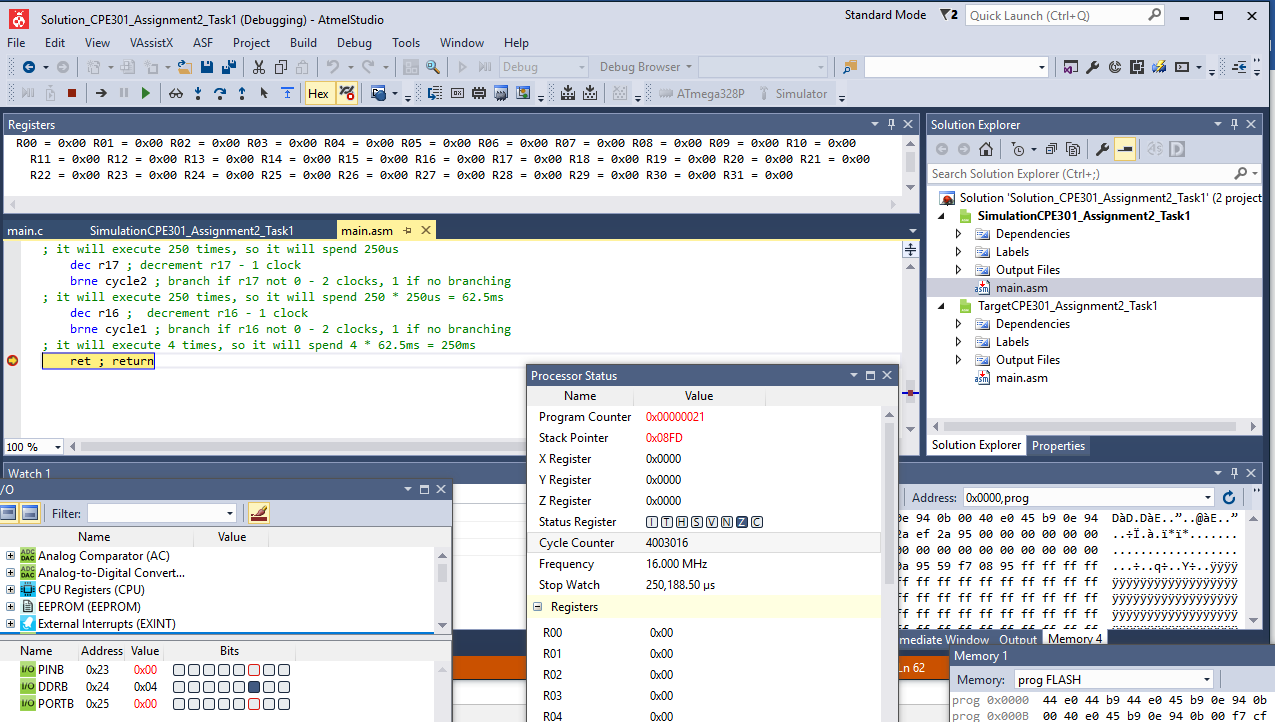
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Task( 1)

Start:

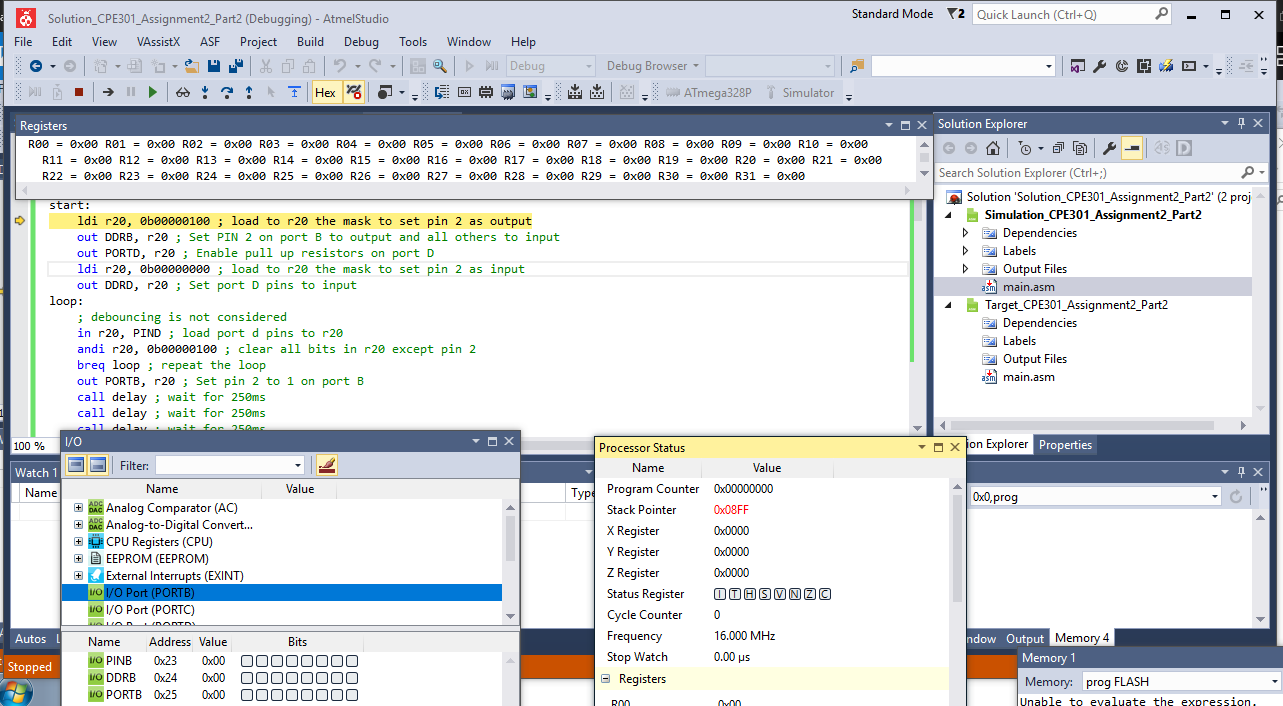


Task2:

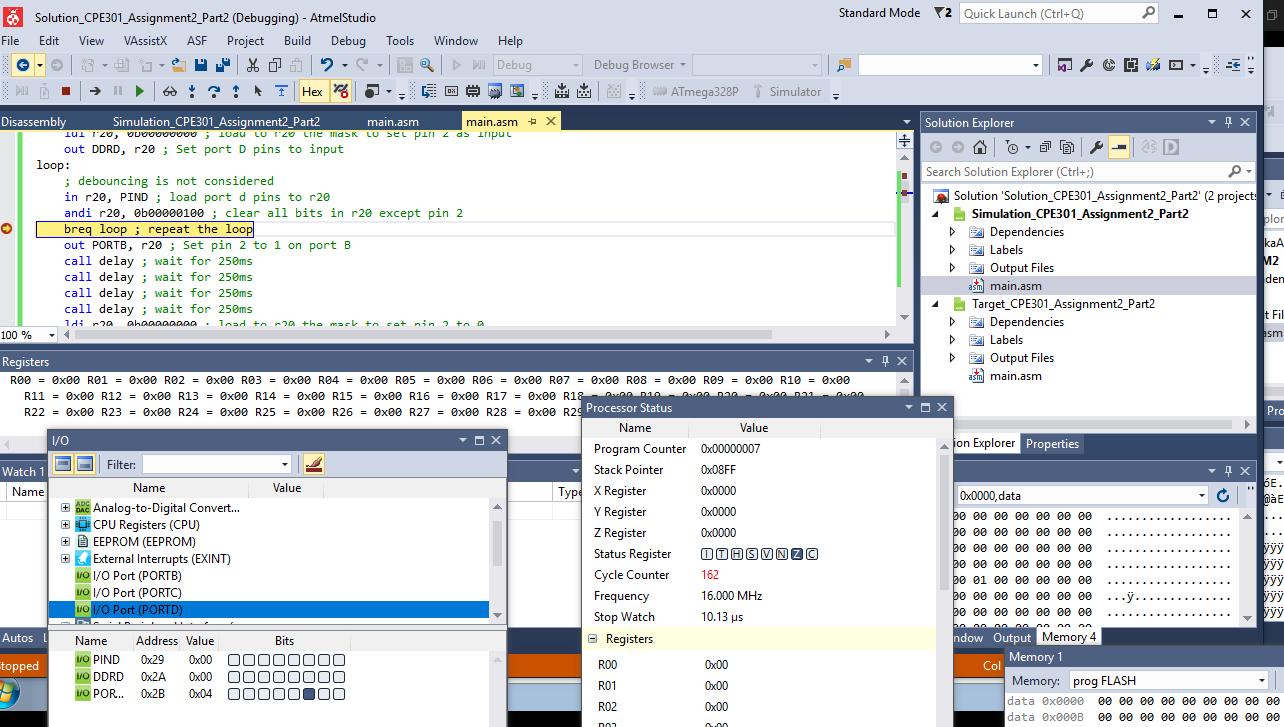


Part(2)

Start

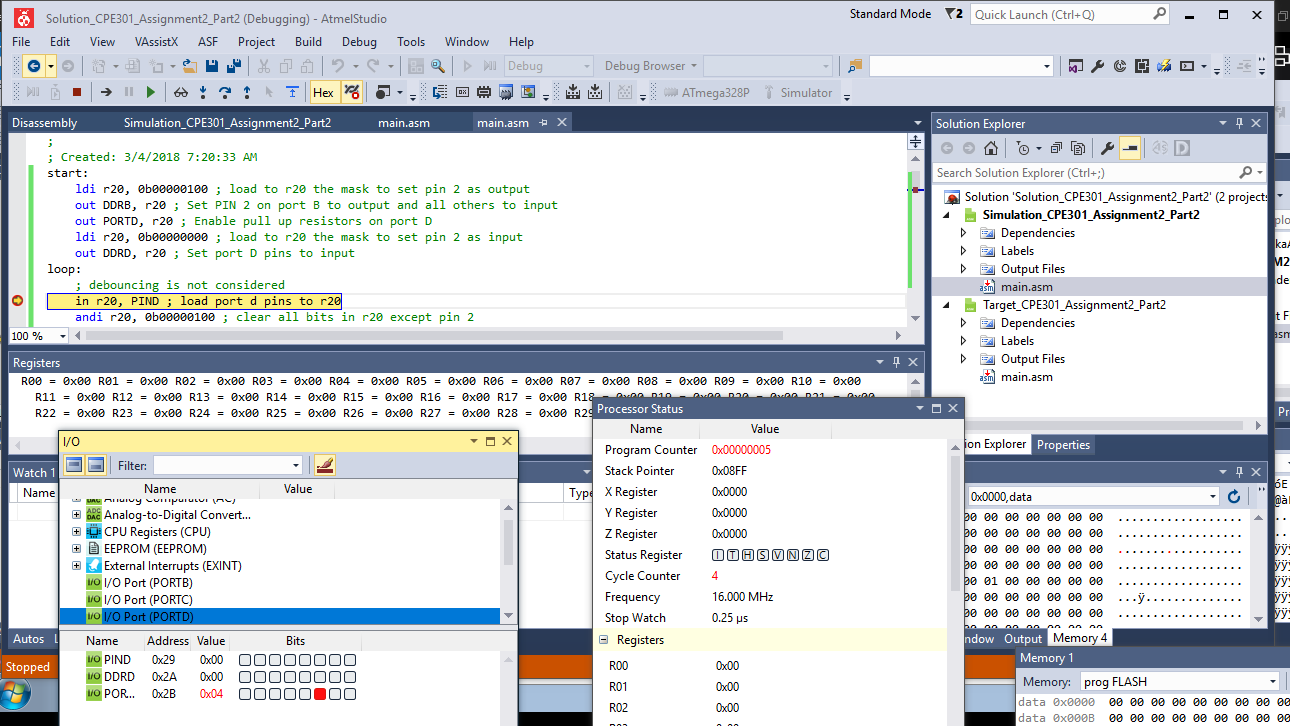


PinD after:

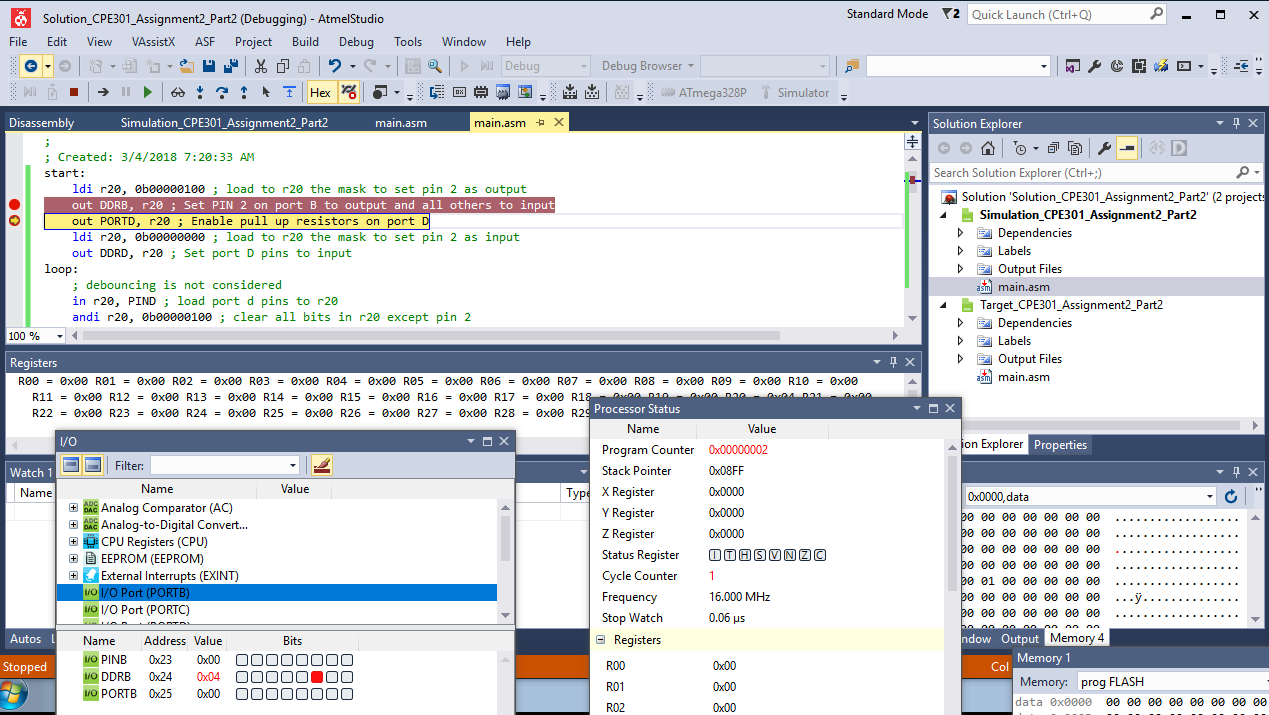


Port B after 1s:

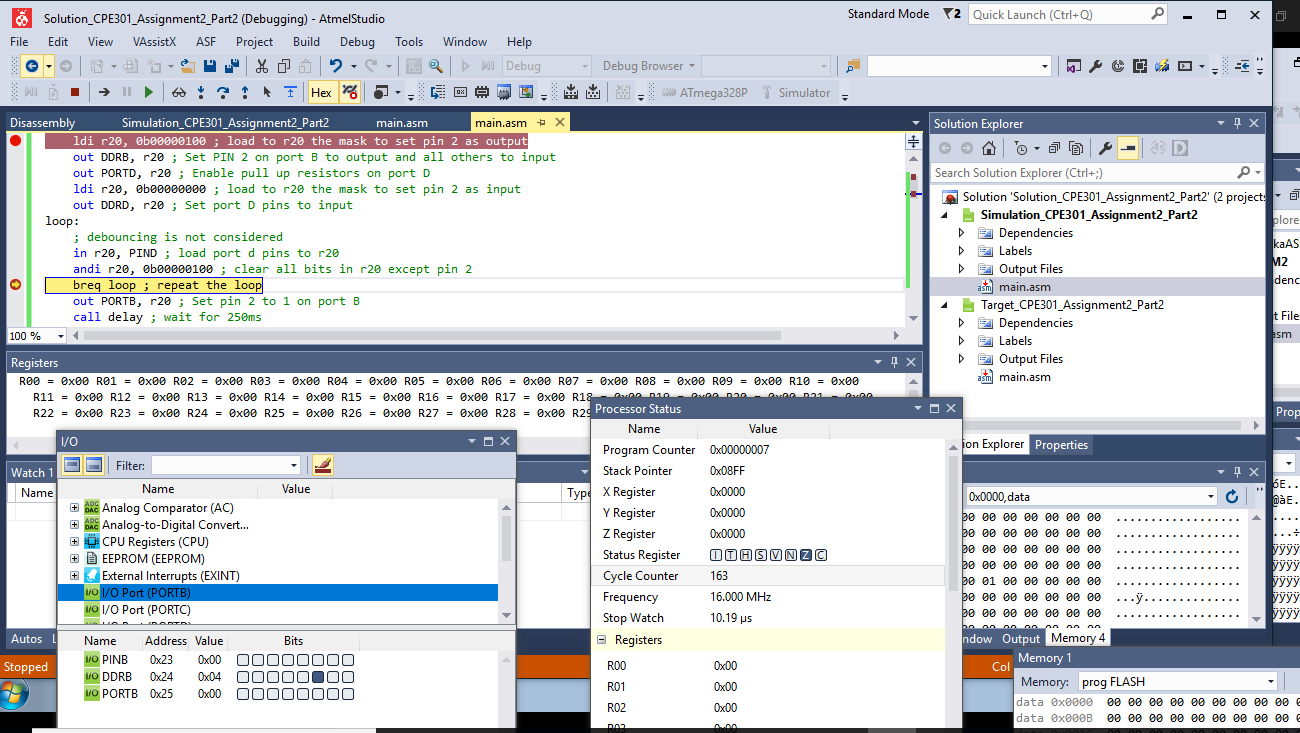
PortD Setup:



PortB Setup:

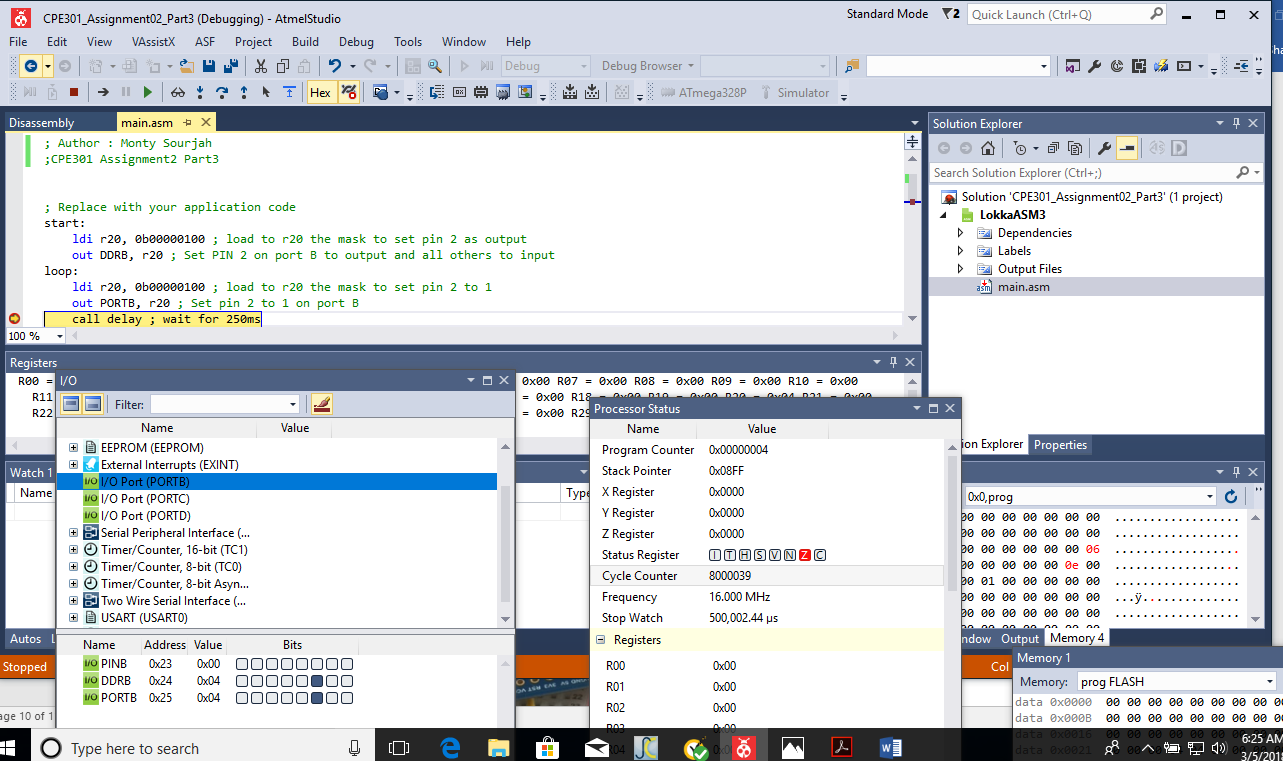


Port B Before:

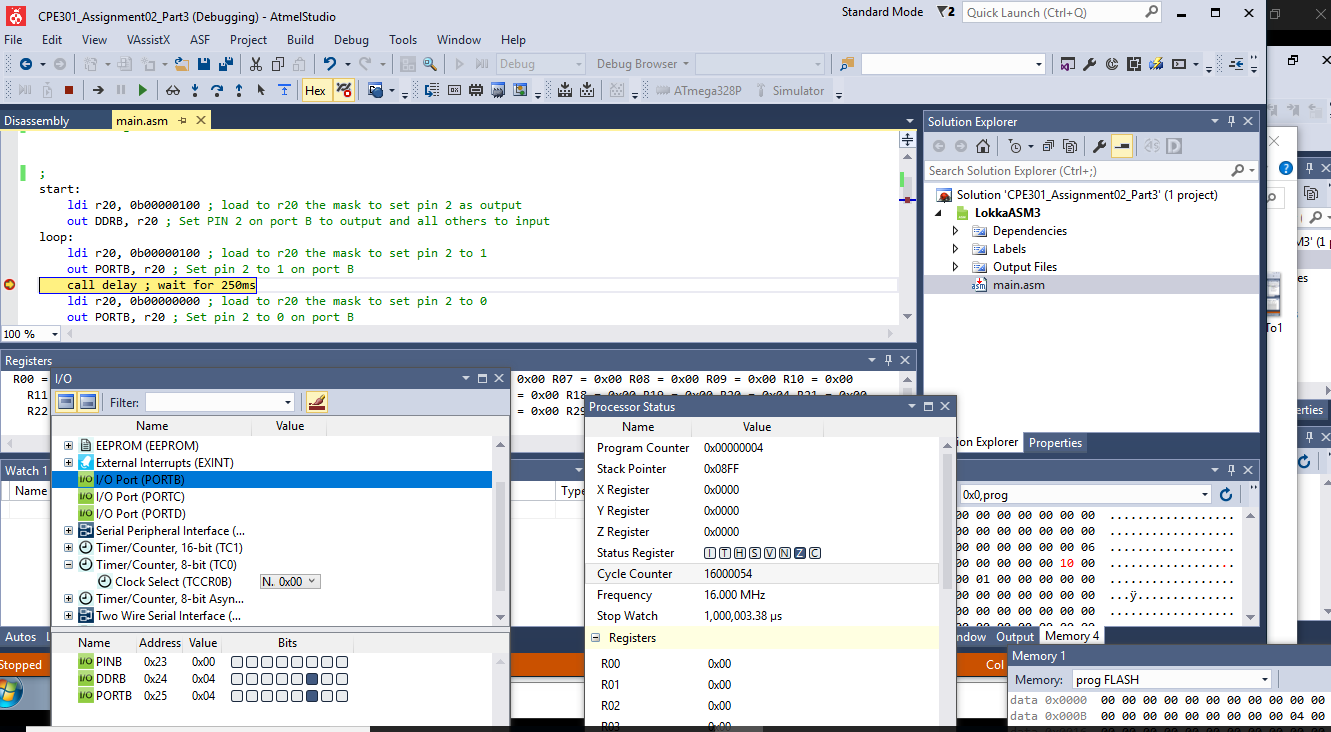


Task3:

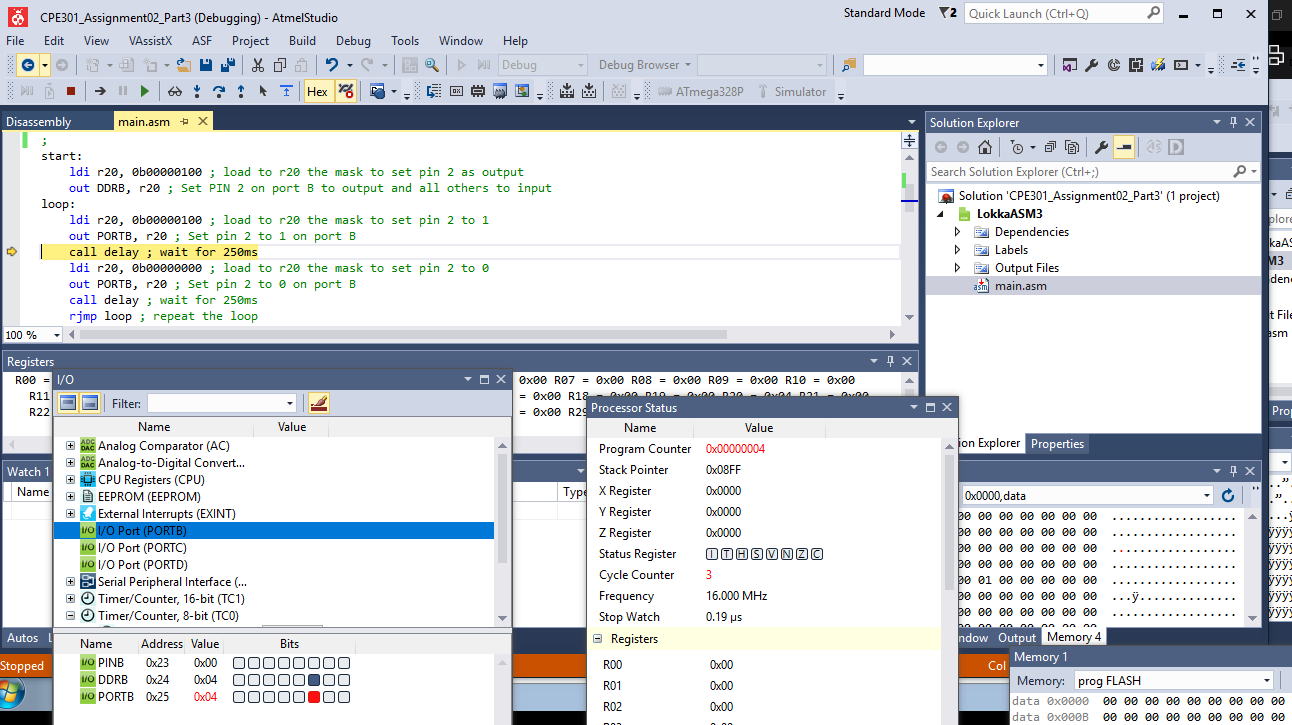
Port B set to 0 after 500ms:



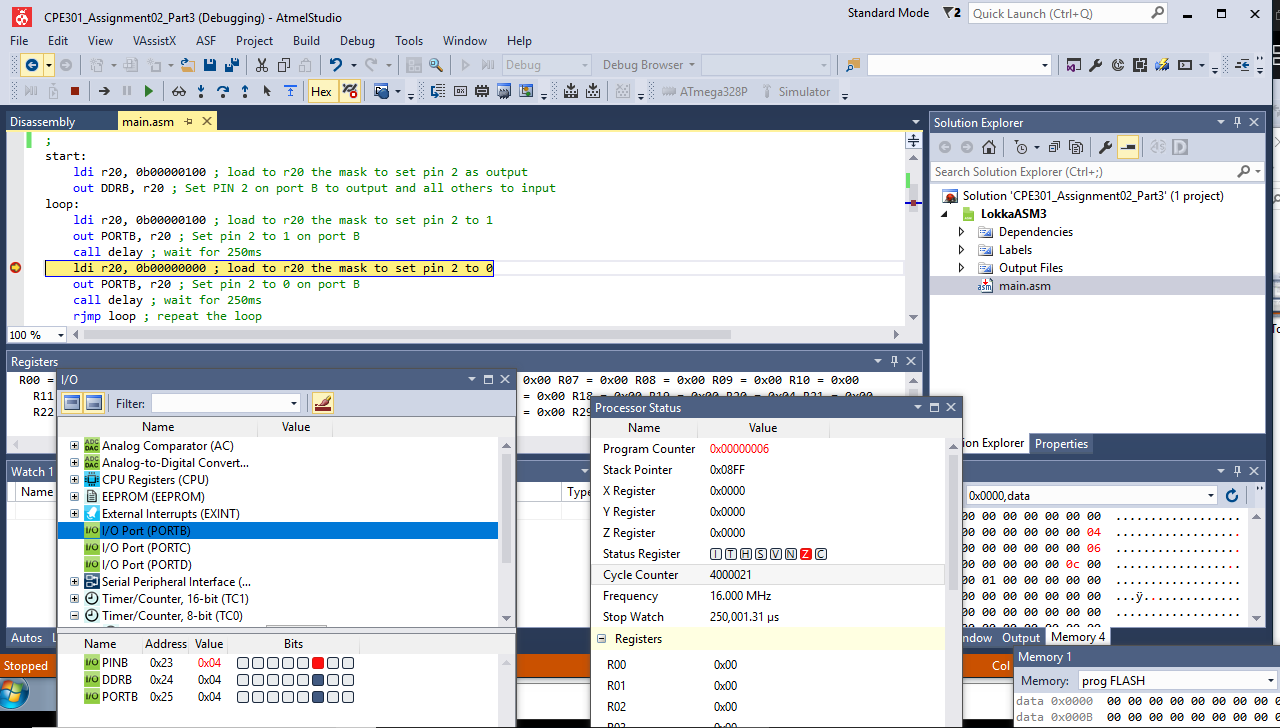
Port B Set To 0 After 1000 ms



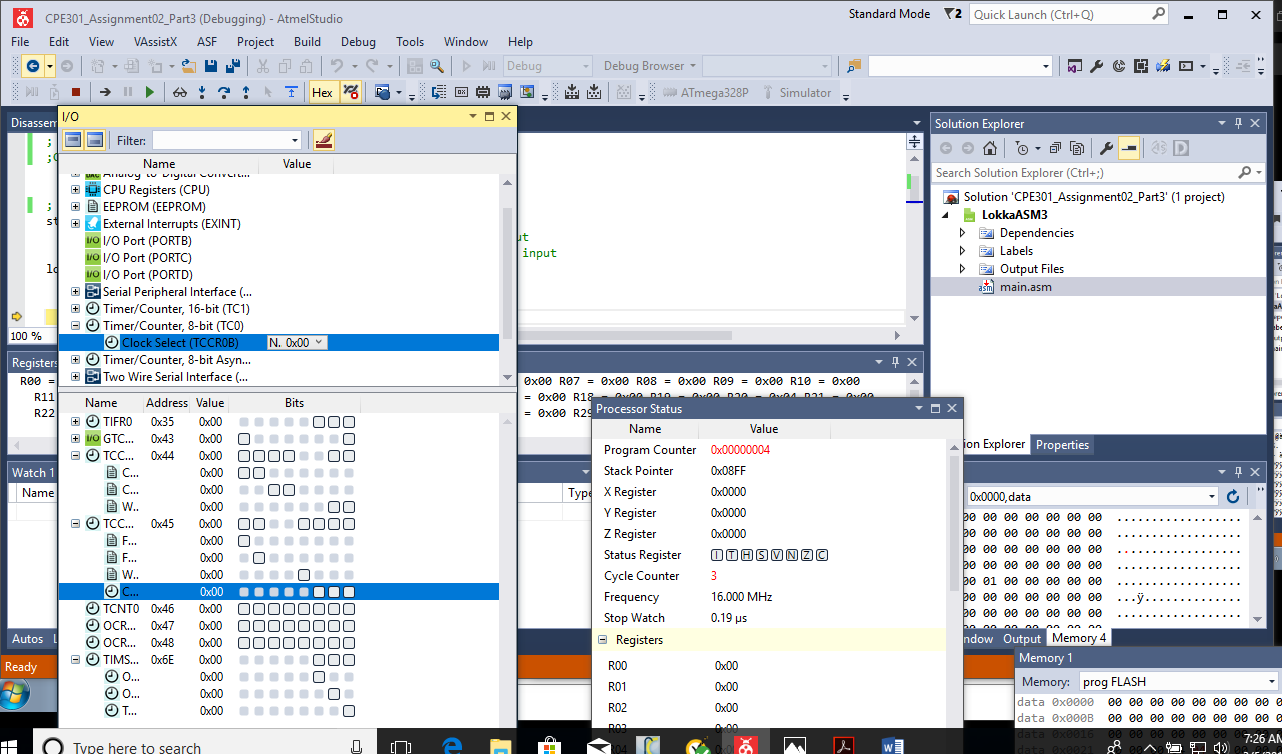
Port B set to 1



Port B Set to 1 After 250ms



Port B Set To 1 Timer0

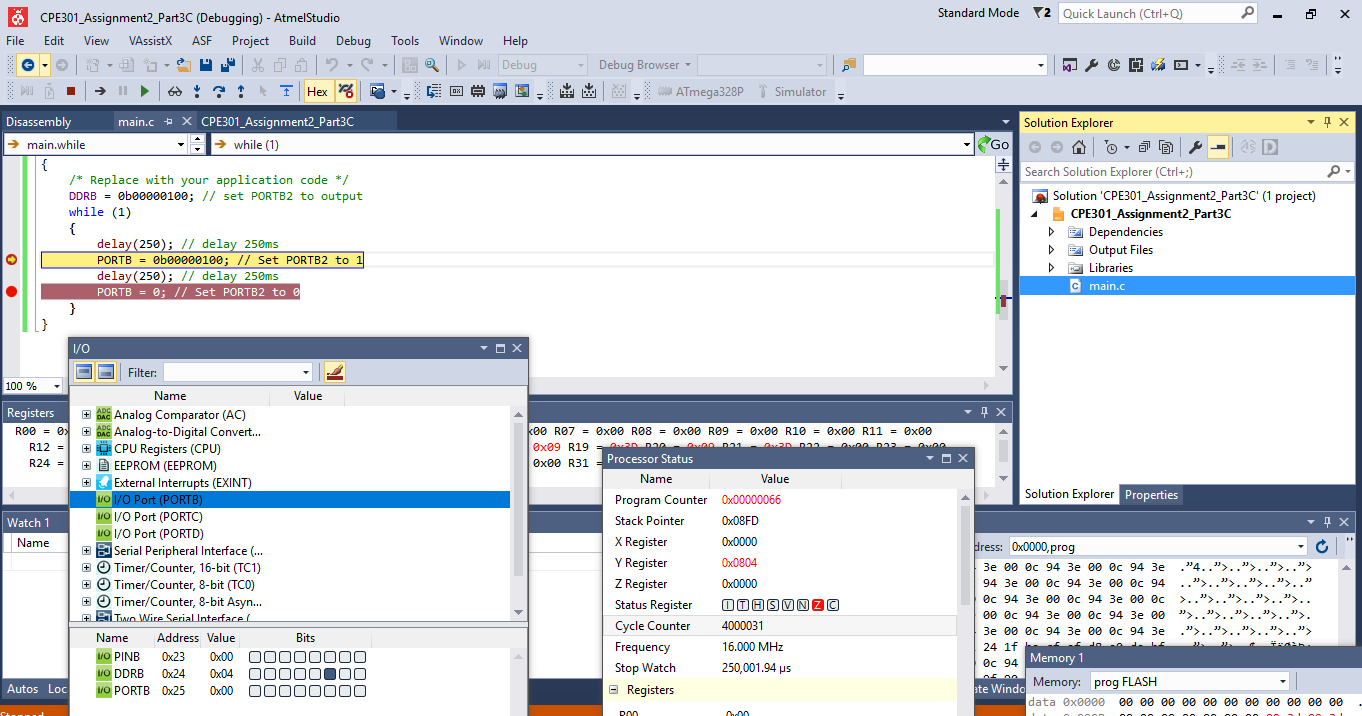


Port B Set To 1 Timer0 After 250ms



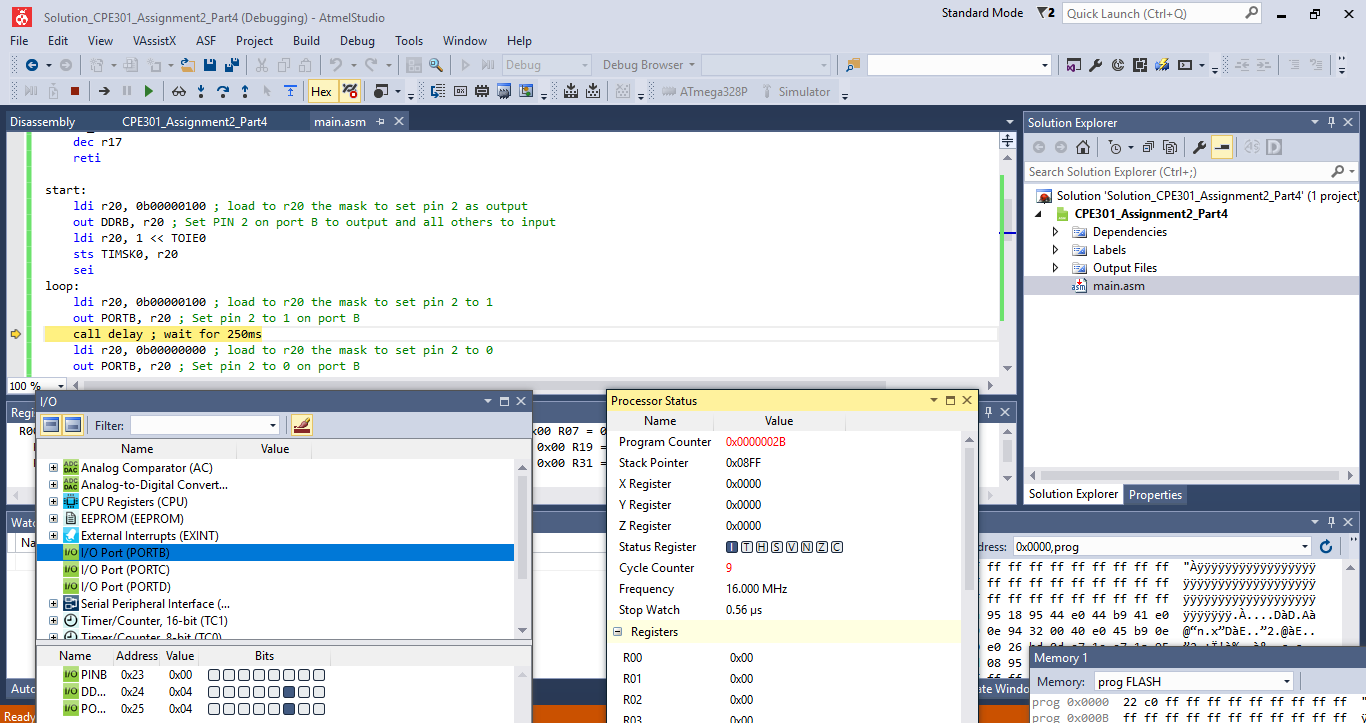
Task 3 C

Port B2 After Set 250ms

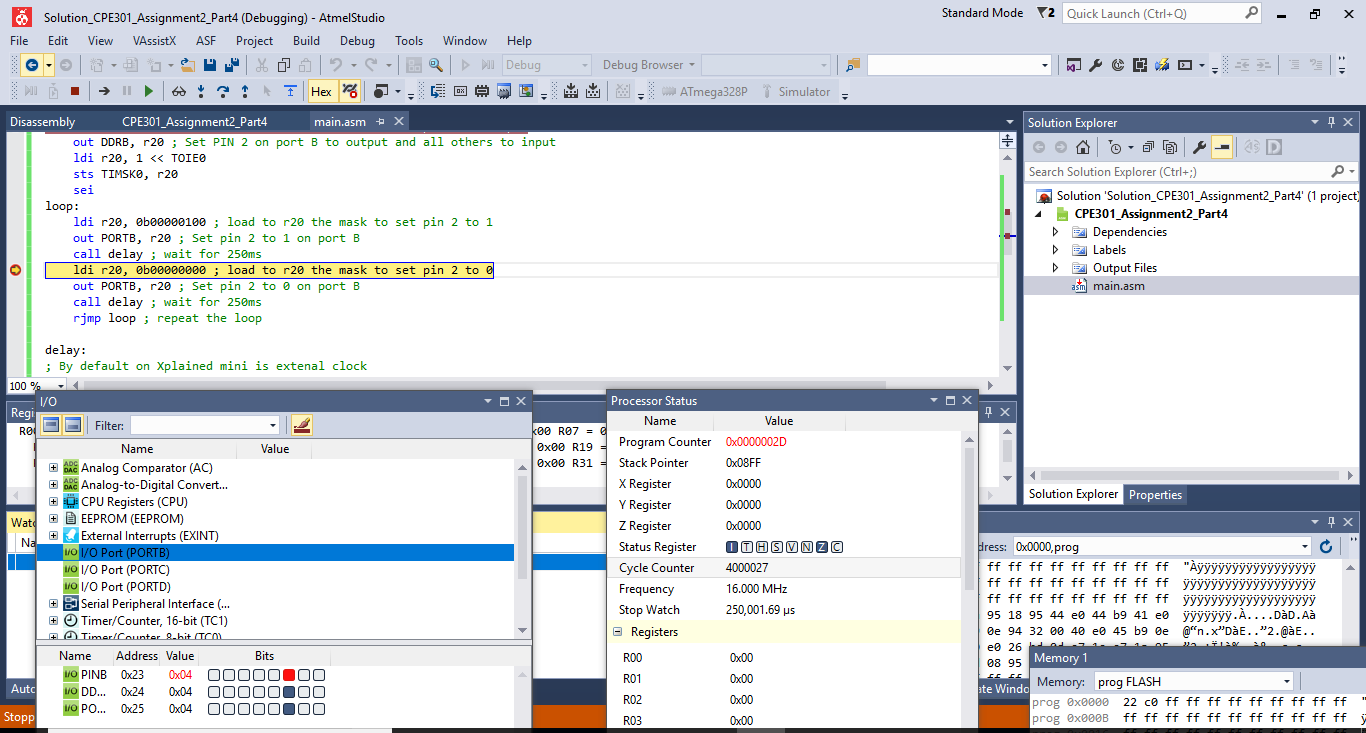


Task 4:

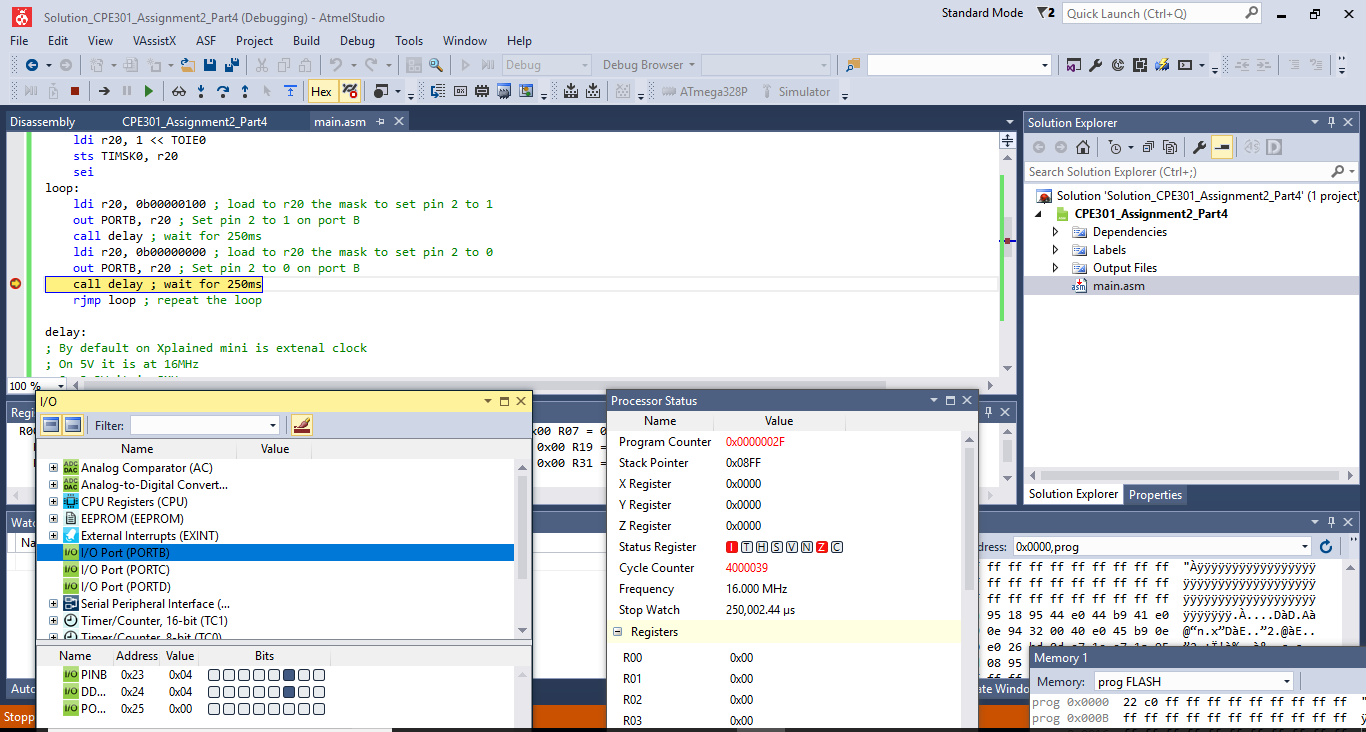
Port B Set on 0 ms:



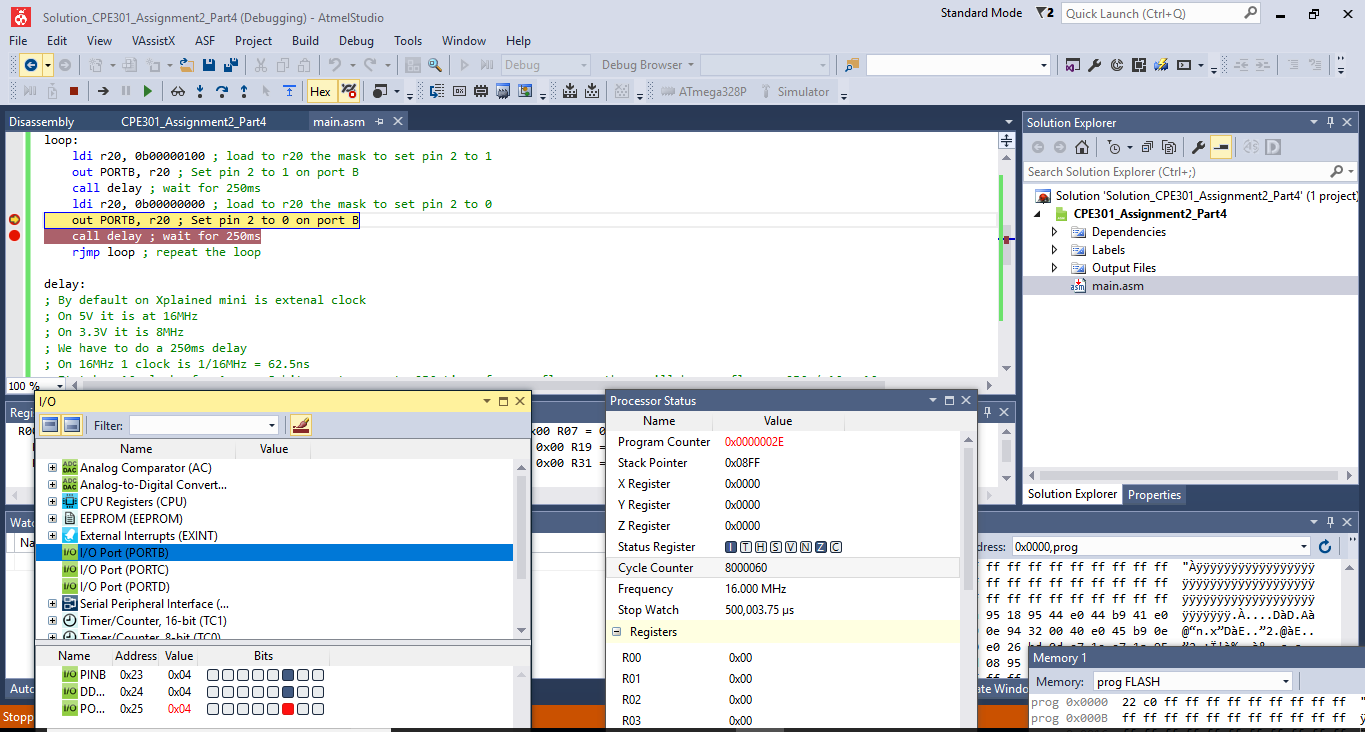
Port B Set On after250 ms:



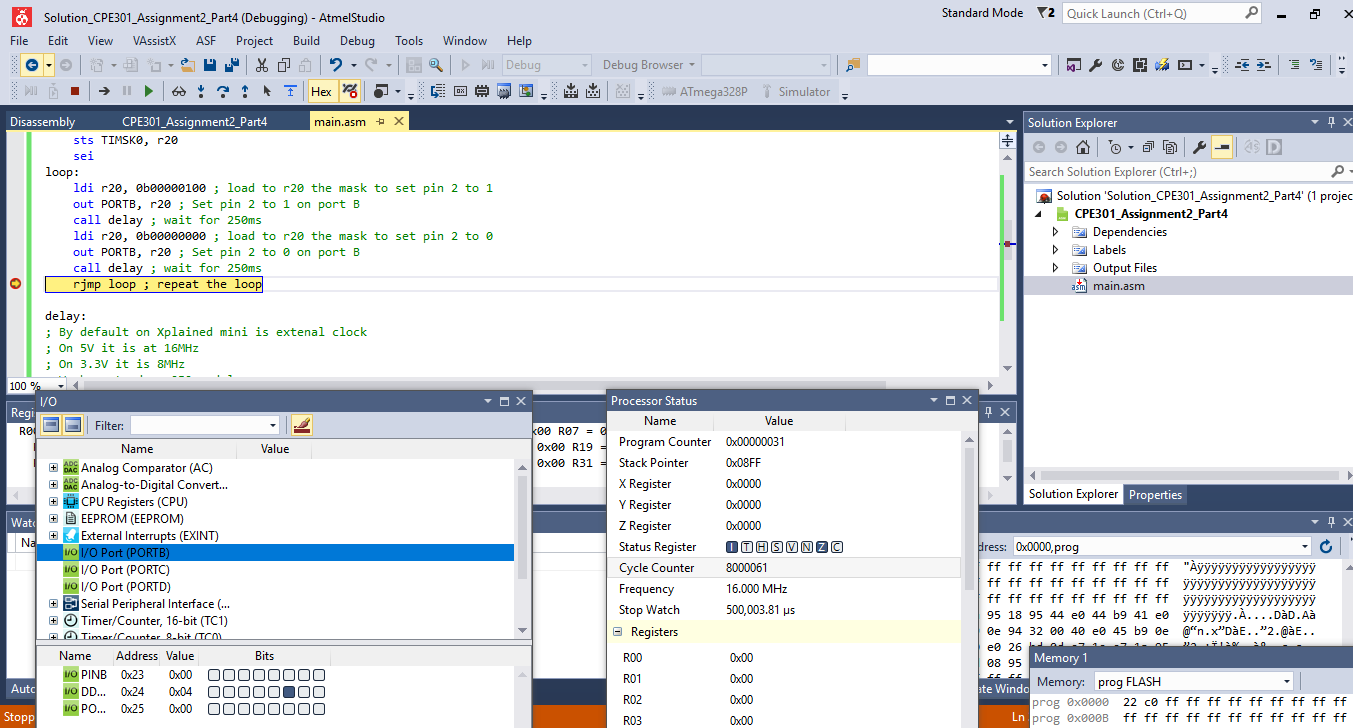
Port B Set Off after 250 ms:



Port B ON After 500ms:

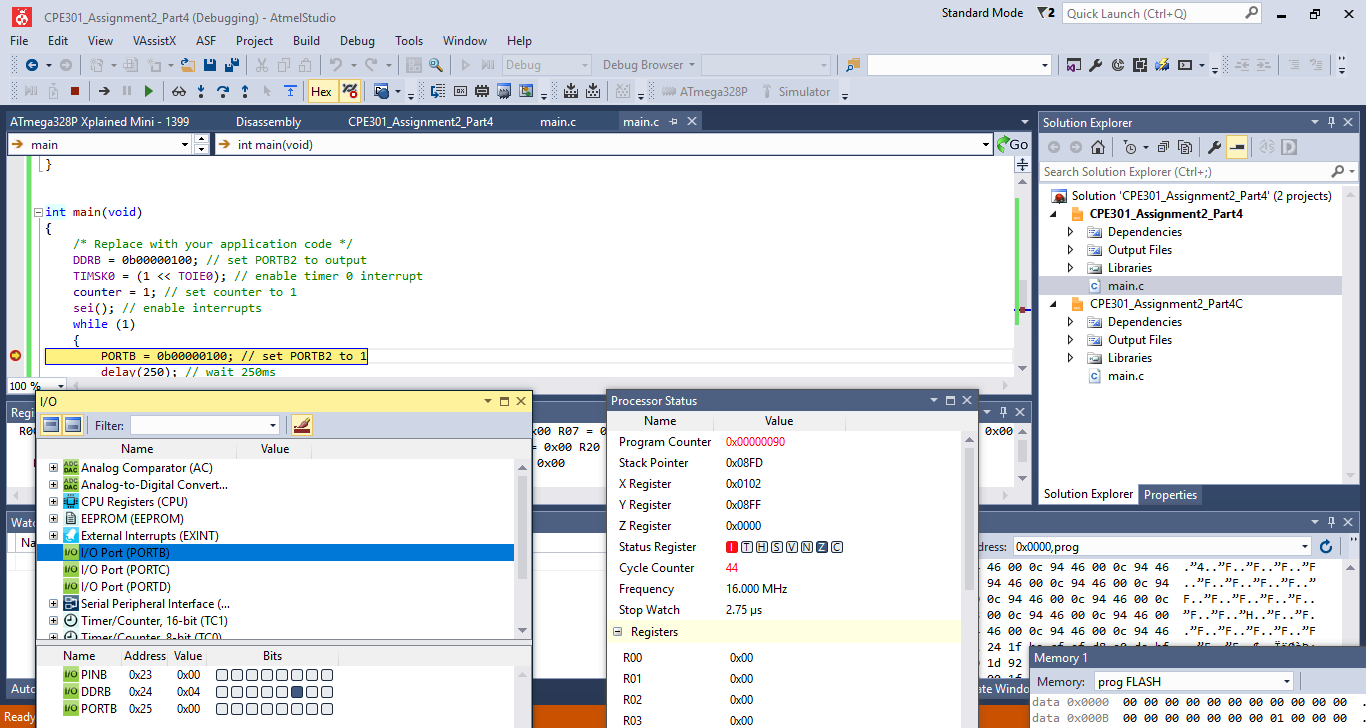


Port B OFF After 500ms:

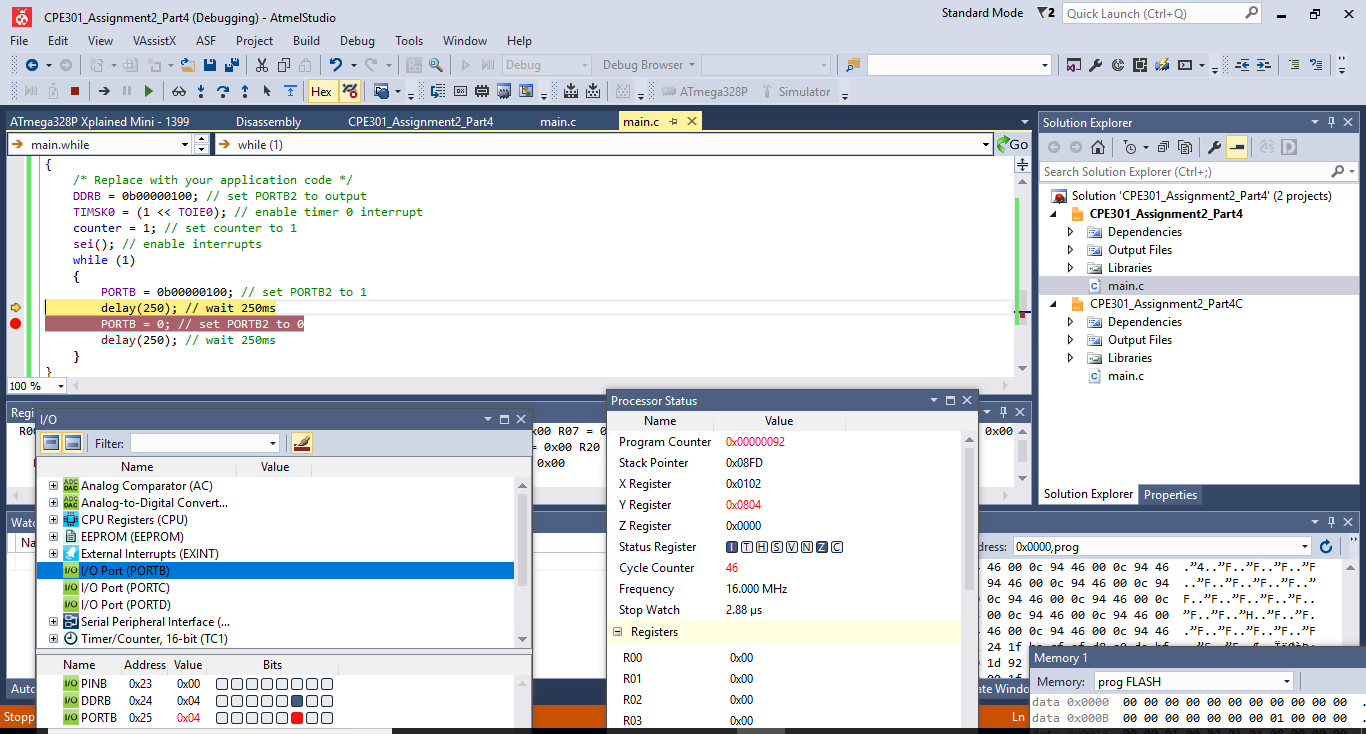


Task 4 Code:

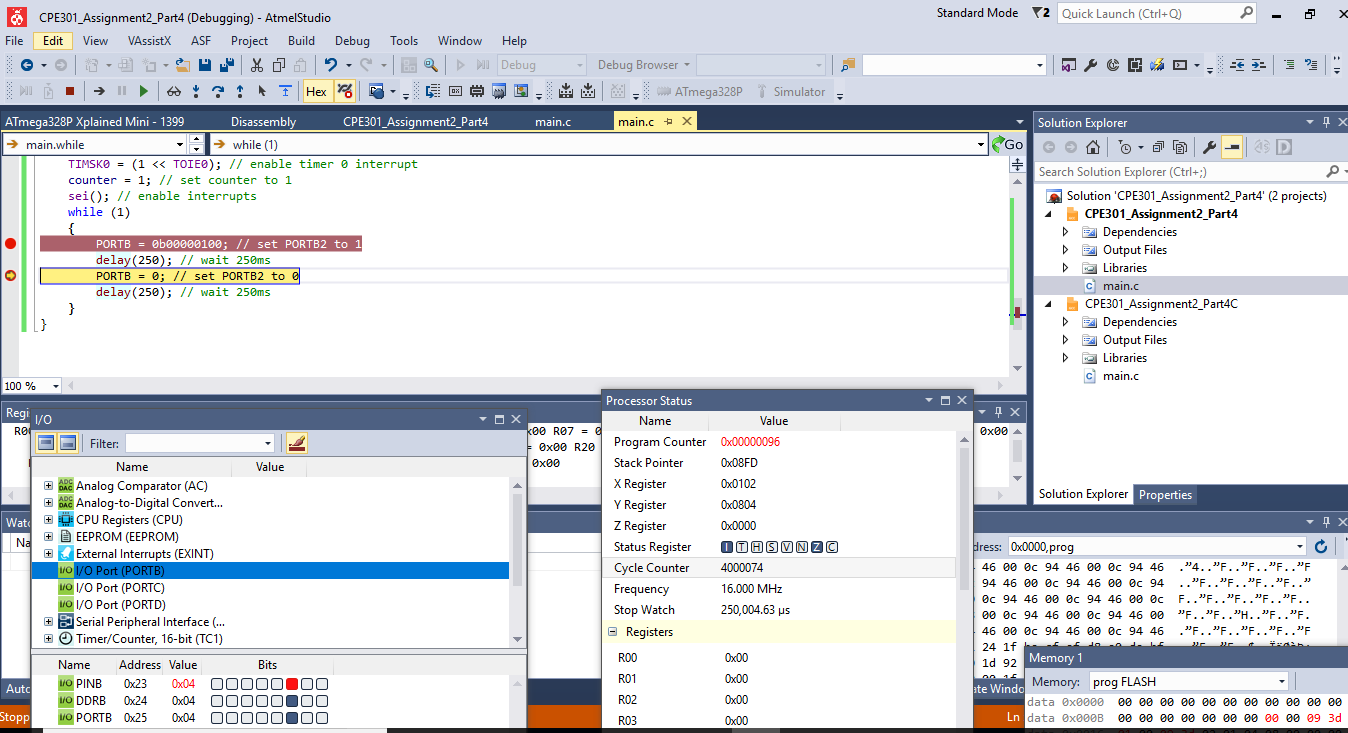
Port B2 0 Before Set:



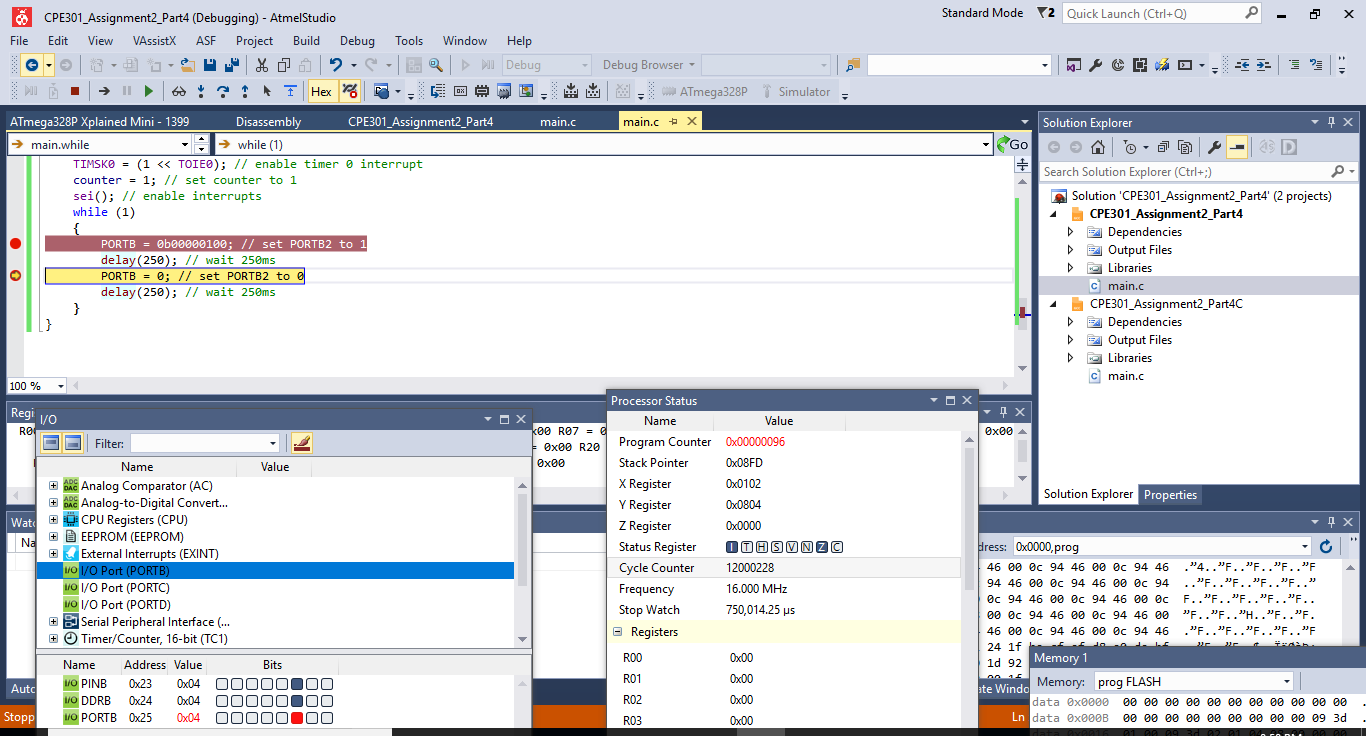
Port B 0 Set:



Port B Clear after 250ms:

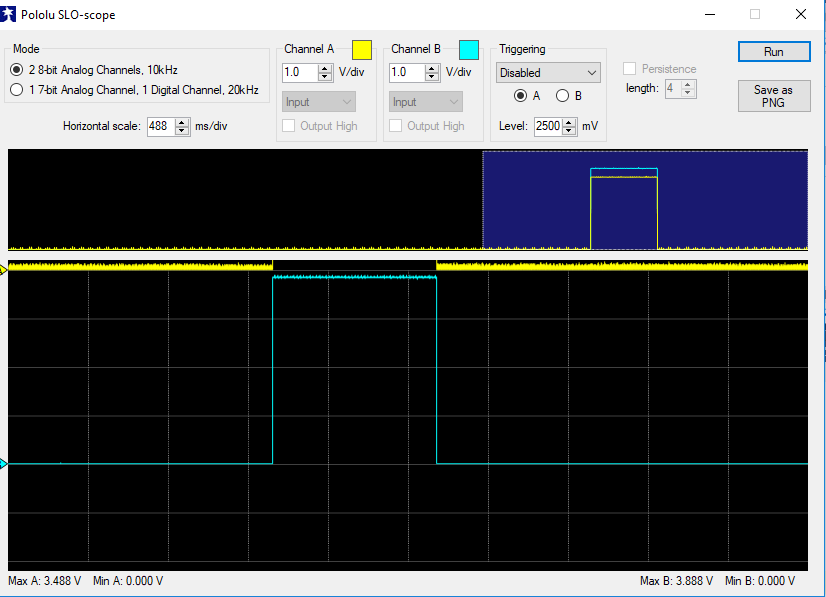


Port B2 Clear after 750ms:

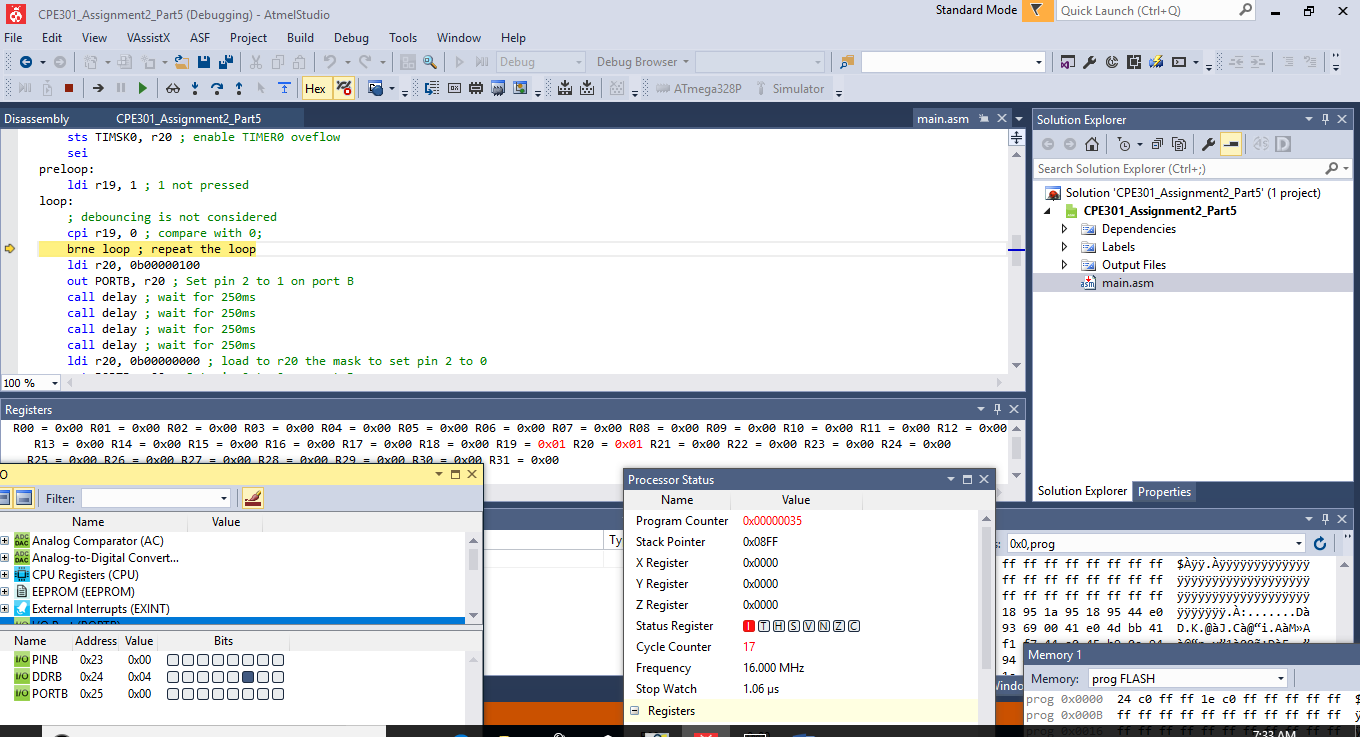


Task 5:

After press button wave form captured in Pololu Scope

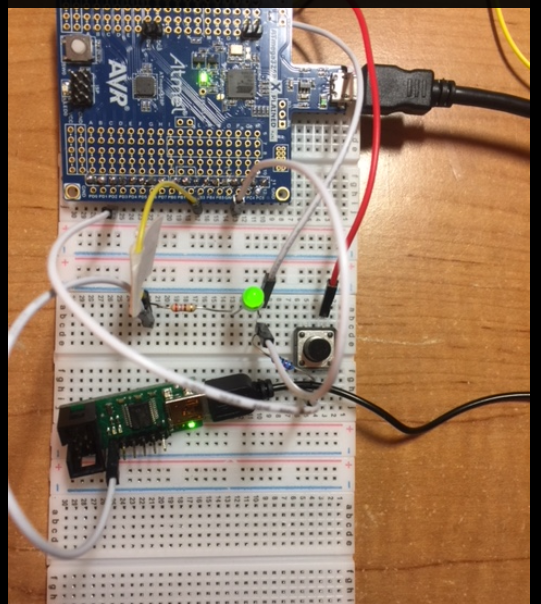


Port B 0ms:

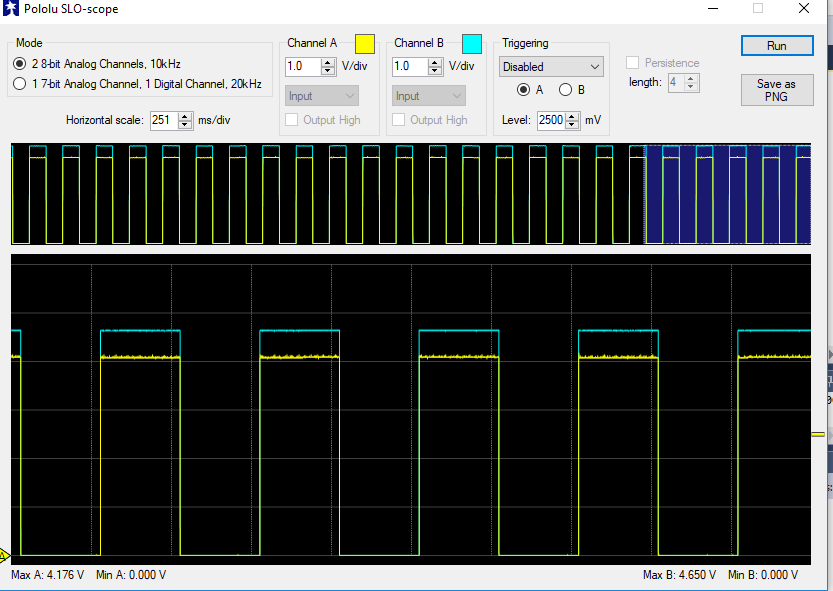


1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

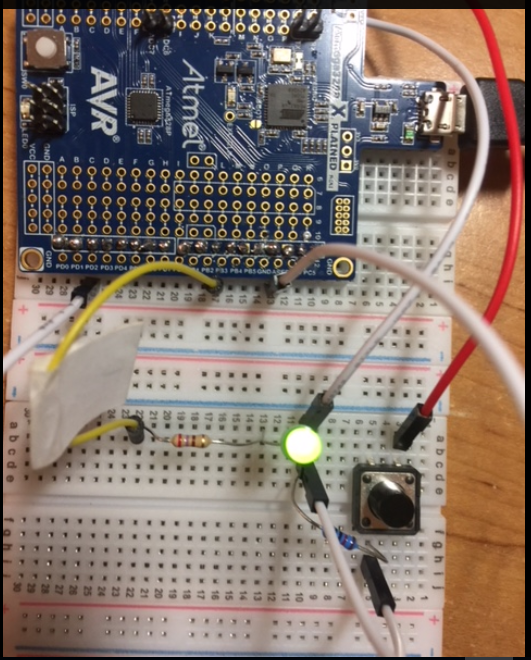
Part(1)



Polulo Scope:



Part(2)



1. **VIDEO LINKS OF EACH DEMO**

Part(1):

<http://www.youtube.com/watch?v=FmxLbcPPyNg>

Part(2):

<http://www.youtube.com/watch?v=fHGfqQuW-mU>

Part(3):

<http://www.youtube.com/watch?v=qP3z8v4OouQ>

Part(4):

<http://www.youtube.com/watch?v=PN2o1sqrVUA>

Part(5):

<http://www.youtube.com/watch?v=_uArf93o3Rg>

1. **GITHUB LINK OF THIS DA**

<https://github.com/ballasl/CPE301_Assigment2>

**Student Academic Misconduct Policy**

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

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

NAME OF THE STUDENT