Dominique Anguiano

CPE301 – SPRING 2016

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)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2A + 3A |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4A |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 1B |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2B + 3B |  |  |
| 6. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4B |  |  |
| 7. | SCHEMATICS |  |  |
| 8. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 9. | SCREENSHOT OF EACH DEMO |  |  |
| 10. | VIDEO LINKS OF EACH DEMO |  |  |
| 11. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

Equipment Used

* Atmega328p
* Eight 300Ω resistors
* One Red LED
* One LED Bar graph

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1A |  |  |

;

; Da 2 Task 1.asm

;

; Created: 3/1/2016 1:49:55 PM

; Author : Dominique

;

.macro initstack ; Initialize Stack

ldi r16**,** high**(**RAMEND**)**

**out** sph**,** r16

ldi r16**,** low**(**RAMEND**)**

**out** spl**,** r16

.endmacro

Start**:**

sbi DDRC**,** 0 ; Set port C bit 0 as an output

WaveLoop**:**

**call** Delay ; Call the Delay function

ldi r16**,** 1

**out** PORTC**,** r16 ; Output a 1 at PORTC.0

FallingEdge**:**

**call** Delay ; Call the Delay subroutine

ldi r16**,** 0

**out** PORTC**,** r16 ; Output a 0 at PORTC.0

**jmp** WaveLoop ; Restart the Loop

; Subroutine for delay

; This subroutine will cause a delay of approximately .25 Seconds

; Assuming an 8MHz clock

Delay**:** ; Total of 2,000,455 Cycle Delay

**push** r16 ; 2 Cycles each push

**push** r17 ; This saves the values that were

**push** r18 ; originally in these registers

ldi r16**,** 63 ; 1 Cycle

L1**:**

ldi r17**,** 125 ; 1 \* 63 = 63 Cycles

L2**:**

ldi r18**,** 83 ; 1 \* 63 \* 125 = 7,875 Cycles

**nop** ; 1 \* 63 \* 125 = 7,875 Cycles

**nop** ; 1 \* 63 \* 125 = 7,875 Cycles

L3**:**

**dec** r18 ; 1 \* 63 \* 125 \* 83 = 653,625 Cycles

brne L3 ; 2/1 \* 63 \* 125 \* 83 = 1,299,375 Cycles

**dec** r17 ; 1 \* 63 \* 125 = 7,875 Cycles

brne L2 ; 2/1 \* 63 \* 125 = 15,687 Cycles

**dec** r16 ; 1 \* 63 = 63 Cycles

brne L1 ; 2/1 \* 63 = 125 Cycles

**pop** r18 ; 2 Cycles each pop

**pop** r17 ; Return the values to the registers

**pop** r16

**ret** ; 4 Cycles

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2A + 3A |  |  |

;

; Da 2 Task 2.asm

;

; Created: 3/1/2016 1:49:55 PM

; Author : Dominique

;

.macro initstack ; Initialize Stack

ldi r16**,** high**(**RAMEND**)**

**out** sph**,** r16

ldi r16**,** low**(**RAMEND**)**

**out** spl**,** r16

.endmacro

Start**:**

sbi DDRB**,** 0 ; Set all the bits in Port B as outputs

sbi DDRB**,** 1

sbi DDRB**,** 2

sbi DDRB**,** 3

sbi DDRB**,** 4

sbi DDRB**,** 5

sbi DDRB**,** 6

sbi DDRB**,** 7

sbi DDRC**,** 5 ; SEt port C bit 5 as an output

sbi DDRC**,** 4 ; Set port C bit 6 as an output

ldi r16**,** 0 ; Load r16 to 0 to initialize the counter

**mov** r1**,** r16 ; Copy r16 to r1. r1 will be our 8 bit

; counter

**mov** r20**,** r16 ; Counter to determine when to toggle

WaveLoop**:**

**call** Delay ; Call the Delay function

**inc** r1 ; Increment the 8-bit counter

**out** PORTB**,** r1 ; Output the bits of the 8-bit counter

**inc** r20 ; Increment the toggle counter

cpi r20**,** 5 ; Compare to 4 different immediate values

breq Toggle5th ; to determine which bits to toggle

cpi r20**,** 10

breq Toggle10th

cpi r20**,** 15

breq Toggle15th

cpi R20**,** 20

breq Toggle20th

FallingEdge**:**

**call** Delay ; Call the Delay subroutine

**jmp** WaveLoop ; Restart the Loop

Toggle5th**:**

ldi r16**,** 0x20 ; Set bit 5 to high and clear bit 4

**out** PORTC**,** r16

**jmp** FallingEdge

Toggle10th**:**

ldi r16**,** 0x10 ; Clear bit 5 and set bit 4

**out** PORTC**,** r16

**jmp** FallingEdge

Toggle15th**:**

ldi r16**,** 0x30 ; Set bit 5 and 4

**out** PORTC**,** r16

**jmp** FallingEdge

Toggle20th**:**

ldi r20**,** 0 ; Clear r20

ldi r16**,** 0x00 ; Clear bit 5 and 4

**out** PORTC**,** r16

**jmp** FallingEdge

; Subroutine for delay

; This subroutine will cause a delay of approximately .25 Seconds

; Assuming an 8MHz clock

Delay**:** ; Total of 2,000,455 Cycle Delay

**push** r16 ; 2 Cycles each push

**push** r17

**push** r18

ldi r16**,** 63 ; 1 Cycle

L1**:**

ldi r17**,** 125 ; 1 \* 63 = 63 Cycles

L2**:**

ldi r18**,** 83 ; 1 \* 63 \* 125 = 7,875 Cycles

**nop** ; 1 \* 63 \* 125 = 7,875 Cycles

**nop** ; 1 \* 63 \* 125 = 7,875 Cycles

L3**:**

**dec** r18 ; 1 \* 63 \* 125 \* 83 = 653,625 Cycles

brne L3 ; 2/1 \* 63 \* 125 \* 83 = 1,299,375 Cycles

**dec** r17 ; 1 \* 63 \* 125 = 7,875 Cycles

brne L2 ; 2/1 \* 63 \* 125 = 15,687 Cycles

**dec** r16 ; 1 \* 63 = 63 Cycles

brne L1 ; 2/1 \* 63 = 125 Cycles

**pop** r18 ; 2 Cycles each pop

**pop** r17

**pop** r16

**ret** ; 4 Cycles

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4A |  |  |

;

; Da 2 Task 3.asm

;

; Created: 3/3/2016 3:45:59 PM

; Author : Dominique

;

.org 0x00

rjmp Start

.org 0x16

rjmp TCNT1\_COMPA

.macro initstack ; Initialize Stack

ldi r16**,** high**(**RAMEND**)**

**out** sph**,** r16

ldi r16**,** low**(**RAMEND**)**

**out** spl**,** r16

.endmacro

Init**:**

Start**:**

sbi DDRB**,** 0 ; Set all the bits in Port B as outputs

sbi DDRB**,** 1

sbi DDRB**,** 2

sbi DDRB**,** 3

sbi DDRB**,** 4

sbi DDRB**,** 5

sbi DDRB**,** 6

sbi DDRB**,** 7

sbi DDRC**,** 5 ; SEt port C bit 5 as an output

sbi DDRC**,** 4 ; Set port C bit 6 as an output

ldi r16**,** 0 ; Load r16 to 0 to initialize the counter

**mov** r1**,** r16 ; Copy r16 to r1. r1 will be our 8 bit

; counter

**mov** r20**,** r16 ; r2 is the toggle counter

; Code to use Timer 1 on CTC mode for the interrupt

ldi r22**,** 0x0F ; Load the value 1954 (0x7A2) into OCR1

sts OCR1aH**,** r22 ; Load the high bits of Output Compare Register 1

ldi r22**,** 0x44

sts OCR1aL**,** r22 ; Load the low bits of Output Compare Register 1

ldi r22**,** 0x00

sts TCCR1A**,** r22 ; Timer 1, CTC mode, Normal port Operation

ldi r22**,** 0x0D ; Turn clock on, 1024 prescaler

sts TCCR1B**,** r22

ldi r22**,** 0x02 ; Enable the Timer 1 Output Compare Match Interrupt

sts TIMSK1**,** r22

sei ; Enable the global Interrupt

WaveLoop**:**

rjmp Waveloop

TCNT1\_COMPA**:**

ldi r22**,** 0x02 ; Load a 1 into bit 1 to clear OCF1A flag

**out** TIFR1**,** r22

**inc** r1

**out** PORTB**,** r1

Toggler**:**

**inc** r20 ; Increment the toggle Counter

cpi r20**,** 5 ; Compare to 4 different immediate values

breq Toggle5th ; to determine which bits to toggle

cpi r20**,** 10

breq Toggle10th

cpi r20**,** 15

breq Toggle15th

cpi r20**,** 20

breq Toggle20th

**jmp** TogglerEnd

Toggle5th**:**

ldi r16**,** 0x20 ; Set bit 5 to high and clear bit 4

**out** PORTC**,** r16

**jmp** TogglerEnd

Toggle10th**:**

ldi r16**,** 0x10 ; Clear bit 5 and set bit 4

**out** PORTC**,** r16

**jmp** TogglerEnd

Toggle15th**:**

ldi r16**,** 0x30 ; Set bit 5 and 4

**out** PORTC**,** r16

**jmp** TogglerEnd

Toggle20th**:**

ldi r20**,** 0 ; Clear r2

ldi r16**,** 0x00 ; Clear bit 5 and 4

**out** PORTC**,** r16

TogglerEnd**:**

reti

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 1B |  |  |

/\*

\* Da 2 Task 4.c

\*

\* Created: 3/4/2016 9:33:14 AM

\* Author : Dominique

\*/

#include <avr/io.h>

#include <util/delay.h>

#define F\_CPU 8000000UL // 8MHz

void delay250ms**();**

int main**(**void**)**

**{**

DDRC **=** 0x01**;** // Set portC.0 as an output

PORTC **=** 0**;** // Initialize portC to output 0

**while** **(**1**)**

**{**

delay250ms**();** // Call the Delay Function

PORTC **=** 1**;** // Output a 1 at PORTC.0

delay250ms**();**

PORTC **=** 0**;** // Output a 0 at PORTC.0

**}**

**}**

// Function to Delay for approximately 250ms

void delay250ms**(){**

**for(**int i **=** 399**;** i **>** 0**;** i**--)**

\_delay\_ms**(**5**);**

**}**

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2B + 3B |  |  |

/\*

\* Da2 Task 5.c

\*

\* Created: 3/4/2016 10:36:45 AM

\* Author : Dominique

\*/

#include <avr/io.h>

#include <util/delay.h>

#define F\_CPU 8000000UL // 8MHz

void delay250ms**();**

int main**(**void**)**

**{**

int i **=** 0**;** // Counter to be displayed on LEDs

int c **=** 0**;** // Counter to determine the 5th and 10th rising pulse

DDRC **=** 0x30**;** // Set PortC.4 and PortC.5 as an output

DDRB **=** 0xFF**;** // Set all of PortB as an output

PORTB **=** 0**;** // Initialize PortB to output 0

PORTC **=** 0**;** // Initialize PortC to output 0

**while** **(**1**)**

**{**

delay250ms**();** // Call the Delay Function

i**++;** // Increment the 8 bit-counter

PORTB **=** i**;** // Output the 8-bit counter to port B

c**++;** // Increment the toggle counter

**switch(**c**)** **{** // Switch Statement to determine which bits to toggle

**case** 5 **:** PORTC **=** 0x20**;** **break;**

**case** 10 **:** PORTC **=** 0x10**;** **break;**

**case** 15 **:** PORTC **=** 0x30**;** **break;**

**case** 20 **:** PORTC **=** 0x00**;** c **=** 0**;** **break;**

**}**

delay250ms**();**

**}**

**}**

// Delay Function to delay for 250ms

void delay250ms**(){**

**for(**int i **=** 399**;** i **>** 0**;** i**--)**

\_delay\_ms**(**5**);**

**}**

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4B |  |  |

/\*

\* Da2 Task 6.c

\*

\* Created: 3/4/2016 10:36:45 AM

\* Author : Dominique

\*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define F\_CPU 8000000UL // 8MHz

int c **=** 0**;** // Counter to determine the 5th and 10th rising pulse

int i **=** 0**;** // Counter to be displayed on LEDs

// This interrupt is triggered whenever

// a match occurs

ISR **(**TIMER1\_COMPA\_vect**)** **{**

i**++;** // Increment Counter

PORTB **=** i**;** // Output Counter to Port

c**++;** // Toggle Counter

**switch(**c**)** **{**

**case** 5 **:** PORTC **=** 0x20**;** **break;** // PC5 on, PC4 Off

**case** 10 **:** PORTC **=** 0x10**;** **break;** // PC5 off, PC4 On

**case** 15 **:** PORTC **=** 0x30**;** **break;** // PC4 and PC5 On

**case** 20 **:** PORTC **=** 0x00**;** c **=** 0**;** **break;** // PC4 and PC 5 Off

**}**

**}**

int main**(**void**)**

**{**

DDRC **=** 0x30**;** // Set PortC.4 and PortC.5 as an output

DDRB **=** 0xFF**;** // Set all of PortB as an output

PORTB **=** 0**;** // Initialize PortB to output 0

PORTC **=** 0**;** // Initialize PortC to output 0

OCR1A **=** 0x0F44**;** // Load value 1954 into Output Compare Register

TCCR1A **=** 0x00**;** // Timer 1, CTC mode, Normal port Operation

TCCR1B **=** 0x0D**;** // Turn the clock on with 1024 prescaler

TIMSK1 **=** 0x02**;** // Enable the Timer 1 Output Compare Match Interrupt

sei**();** // Enable Global Interrupts

**while** **(**1**)**

**{**

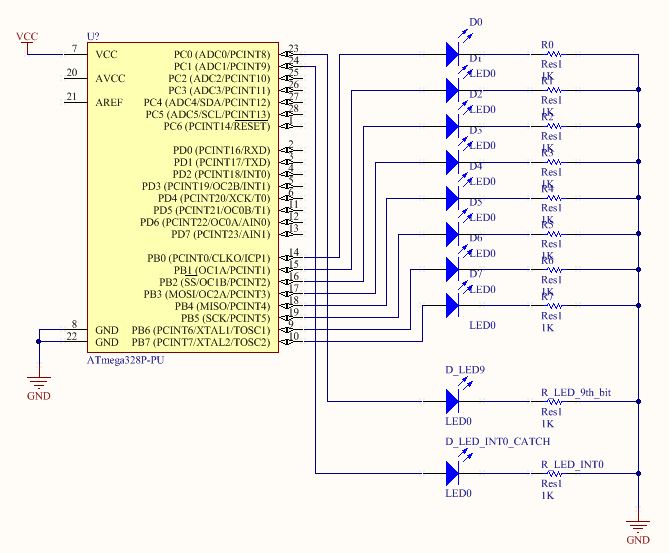
// Do Nothing

// Interrupt should occur every .5 seconds.

**}**

**}**

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |

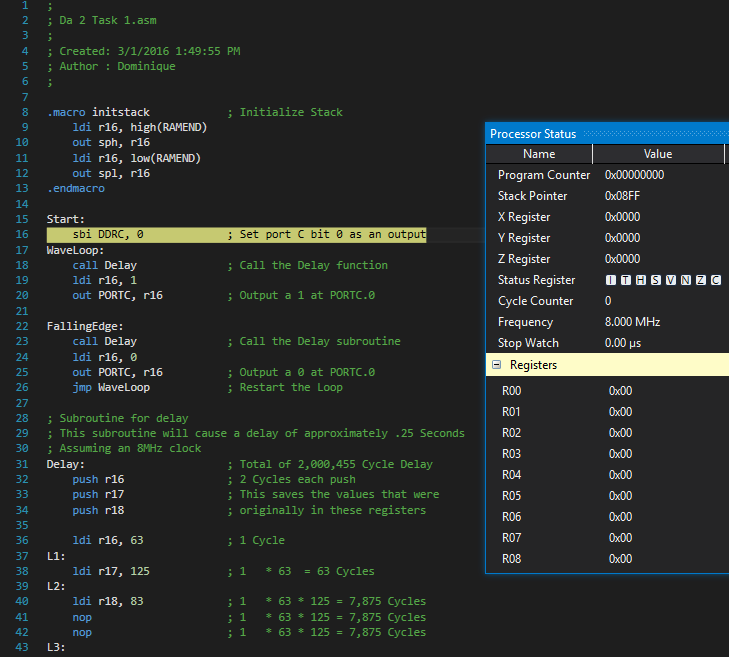


|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

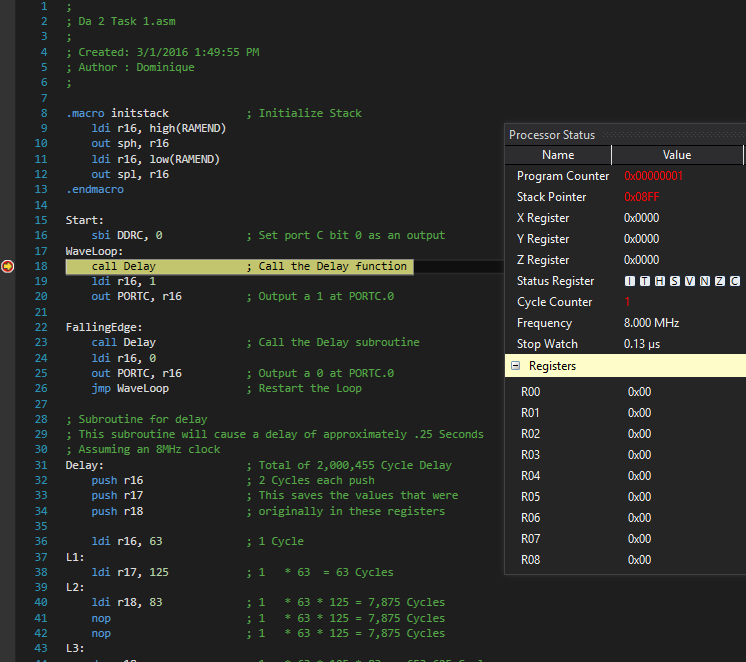
TASK 1A:

Verify duty cycle and period: 50% duty cycle, period = 0.5 second

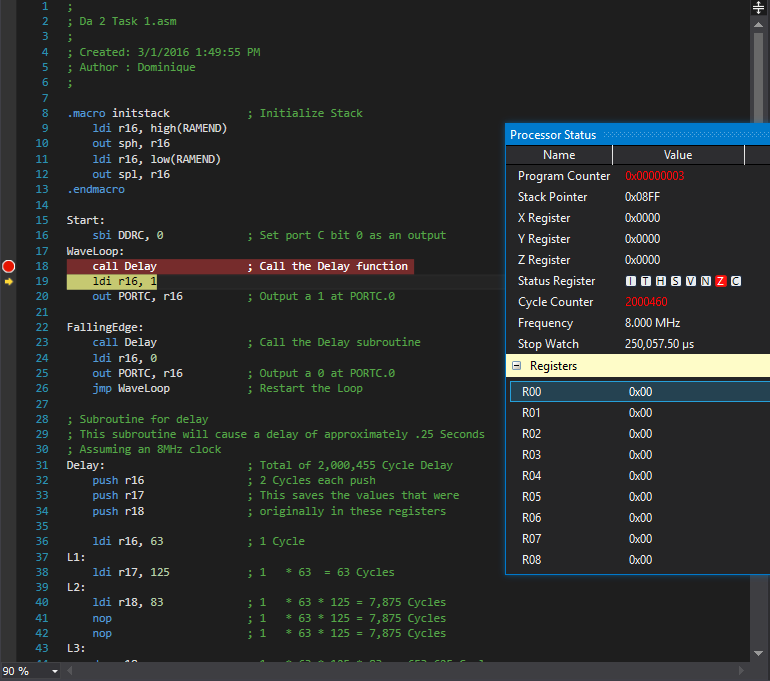
Before Running the program



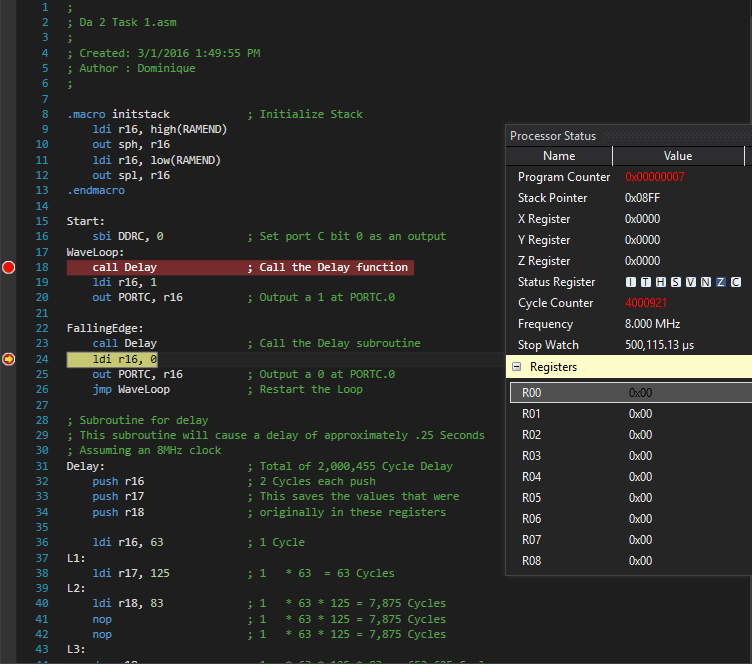
Before executing the Delay Subroutine



After Executing the Delay subroutine once. Delay is approximately 250ms, this is half the desired period.



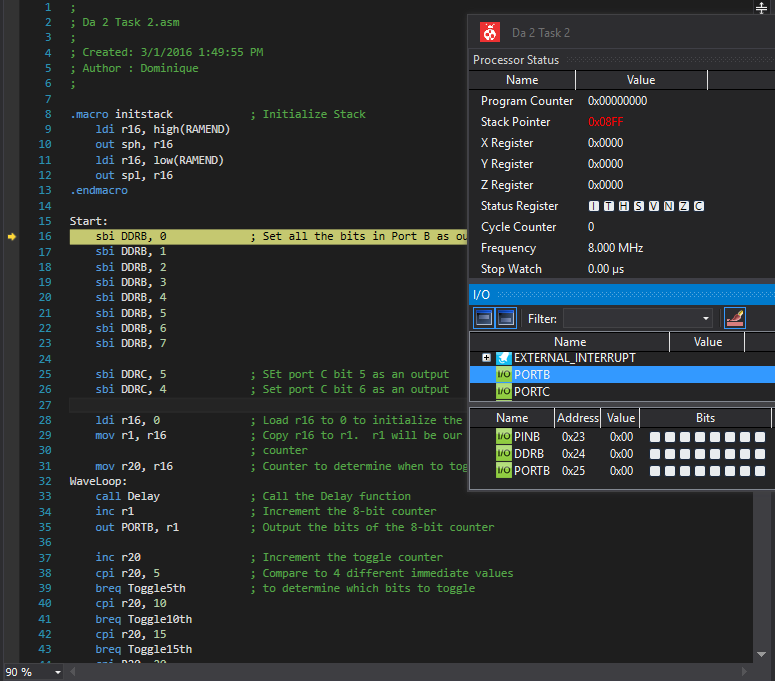
Screenshot after applying the second Delay. This results in a total period of approximately 500ms or .5 Seconds.



Task 2A + 3A:

An 8 bit counter that counts on the rising edge of each waveform as well as two ports that toggle every 5th and 10th rising pulse.

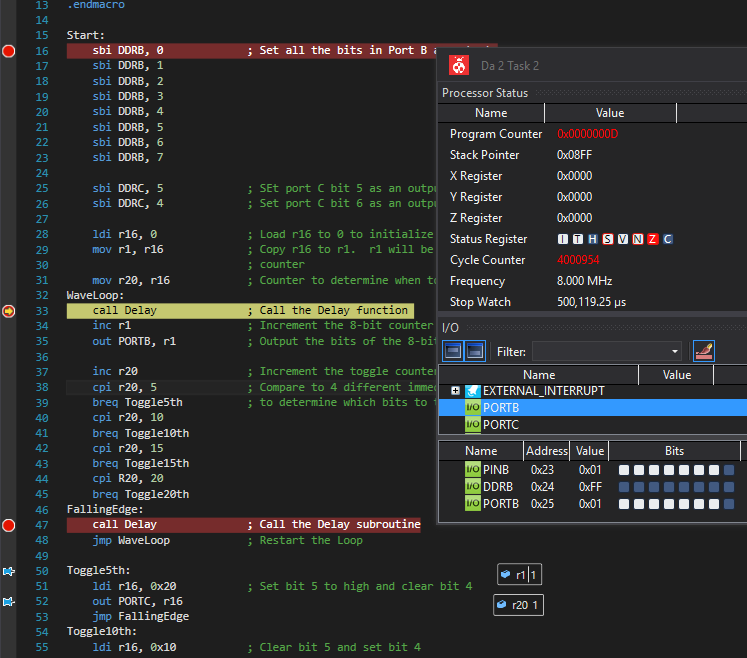
Program before any execution of task 1A



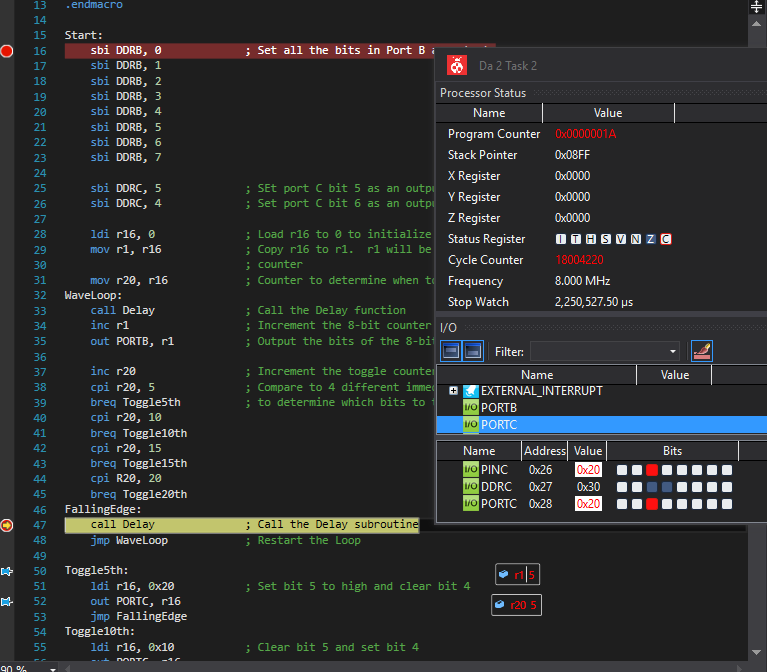
Program after executing the Rising edge delay of the loop of Task 2A + 3A.



Program after Executing the Falling Edge portion of the Delay loop in task 2A + 3A.



Program toggling the value of bit 5 after the 5th rising edge.



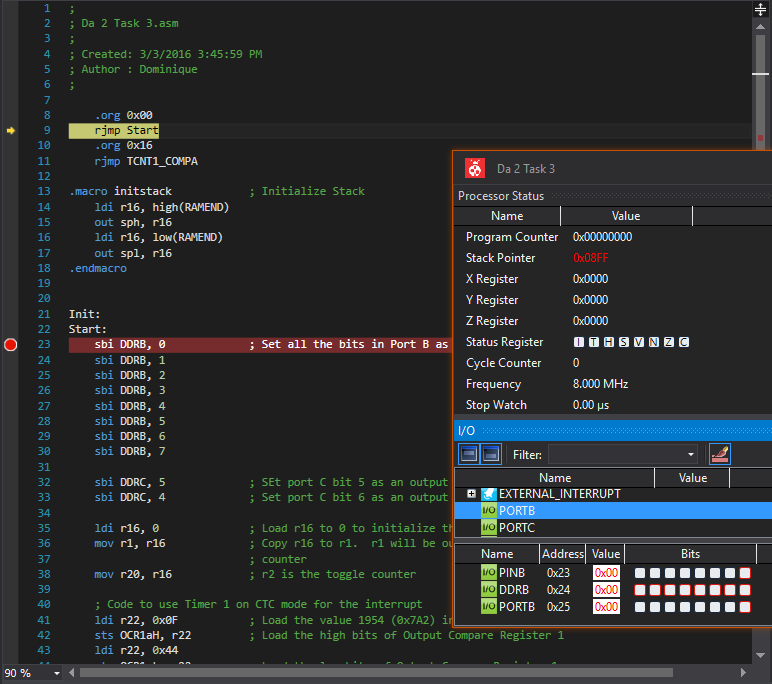
Program toggling the values of bit 5 after another 5 rising edges. It also toggles the value of bit 4 due to this being the 10th rising edge of the waveform.



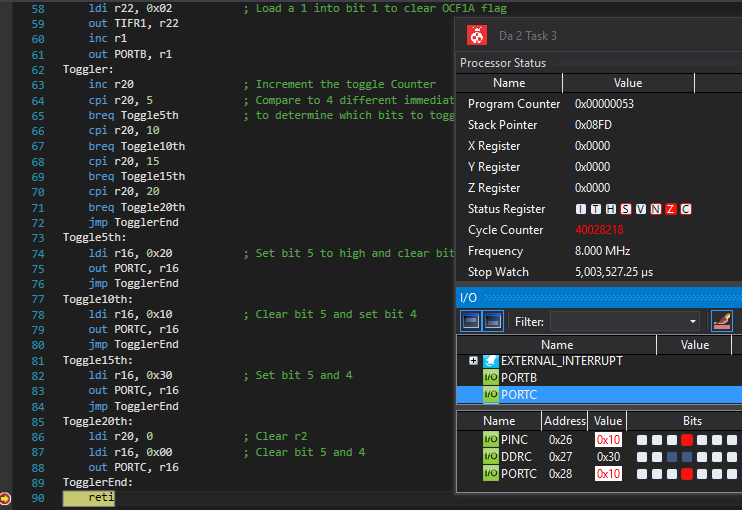
Task 4A

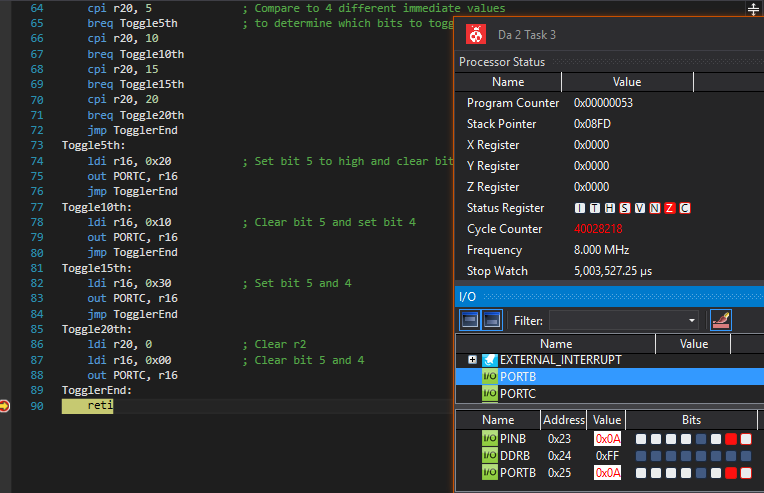
This task required the same things as task 2A+3A except we were required to utilize interrupts.

Task 4A before any program execution



Program after incrementing the counter 10 times. Both bits in PORTC have been toggled and 10 is being output at PORTB.

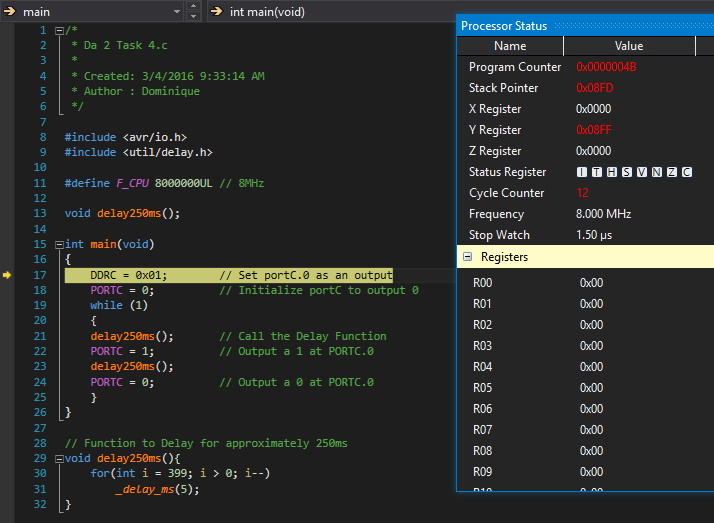


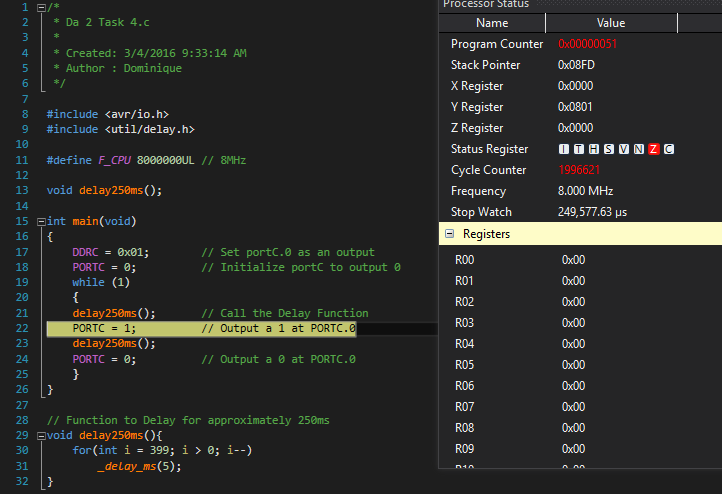


Task 1B required the same results as 1A except that the program is now written in C.

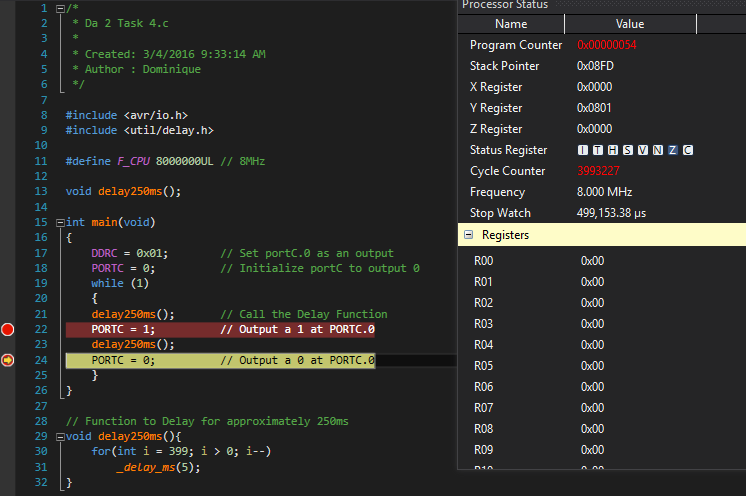
So a .5 second period with a 50% duty cycle is expected.

Program before any execution



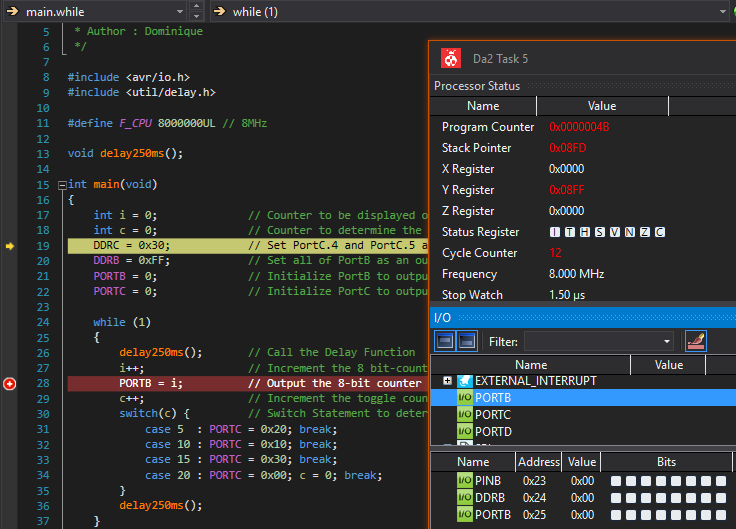
Program after calling the delay function once. Shows a delay of approximately 250ms for half the period.

After calling the delay function again, a delay of 499ms is shown for the entire period.

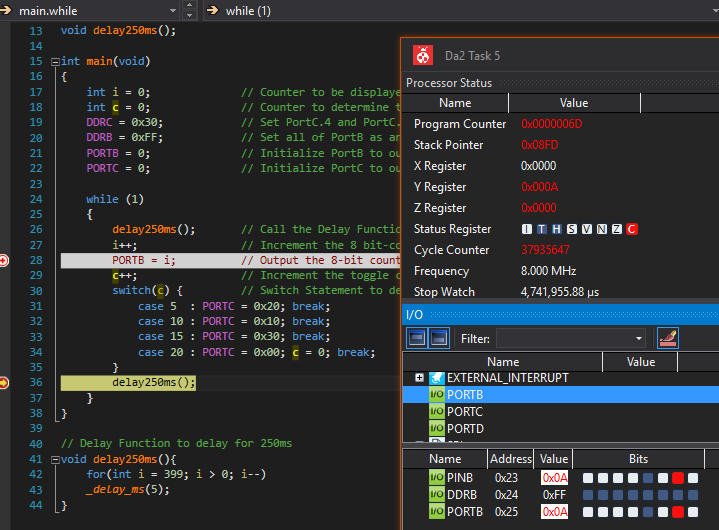


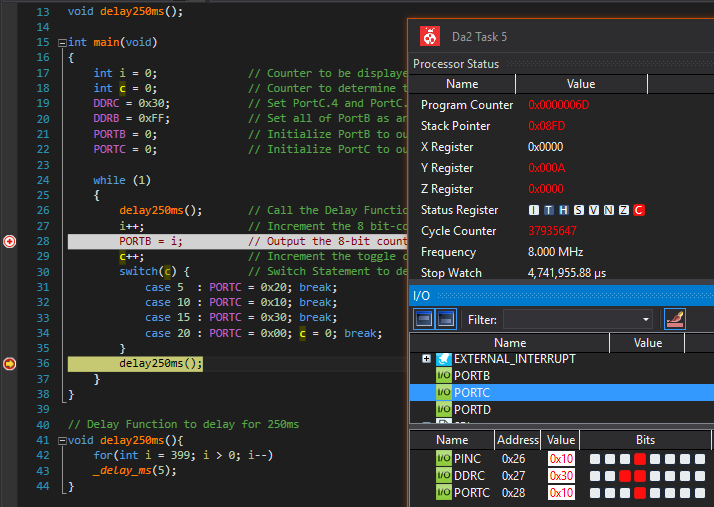
Task 2B + 3B required an 8-bit counter to be implemented with a bit that toggles every 5th rising edge and another bit that toggles every 10th rising edge.

Program at the start of execution



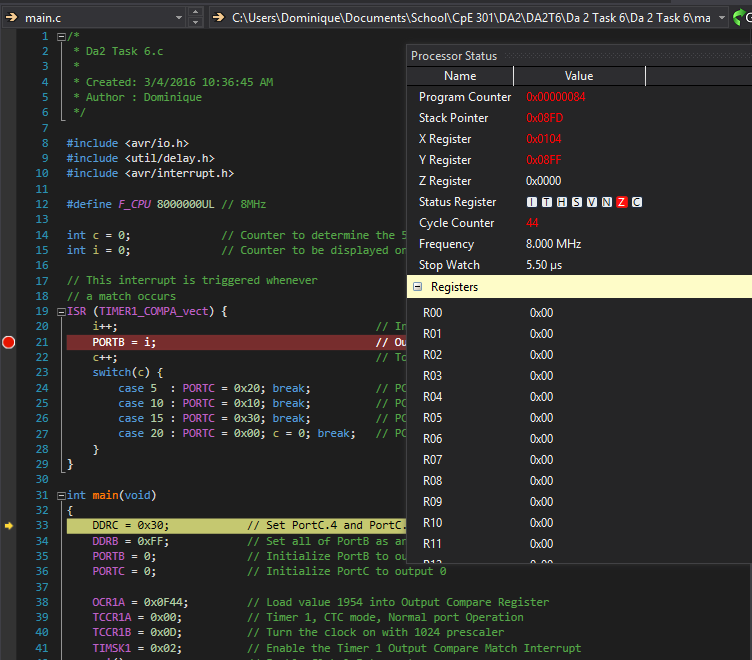
Program after 10 rising edges. The value 10 is being output at PORTB as well as the two bits at PINC being toggled due to this being the 10th rising edge.



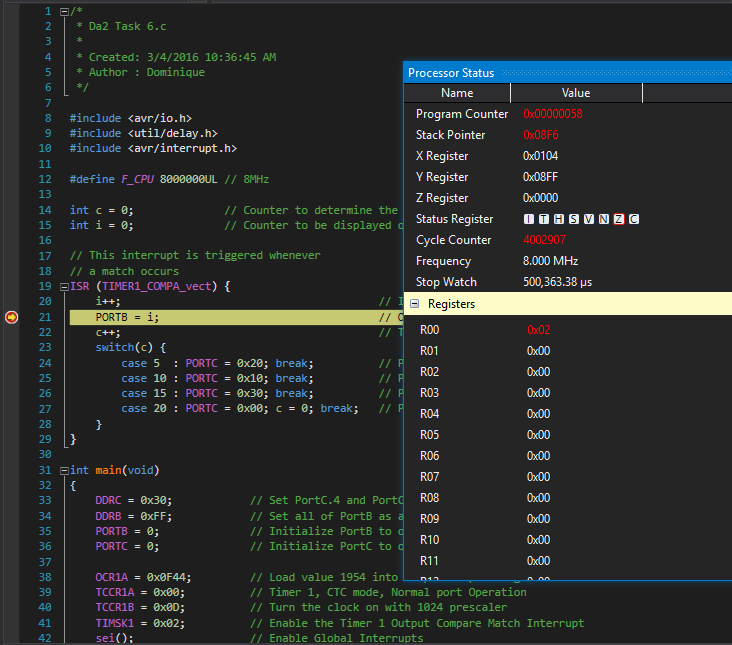


Task 4B required the same things as the assembly code in task 4A. So we must implement an interrupt and bits must be toggled as they were in tasks 2 and 3.

Program at the beginning of exection.

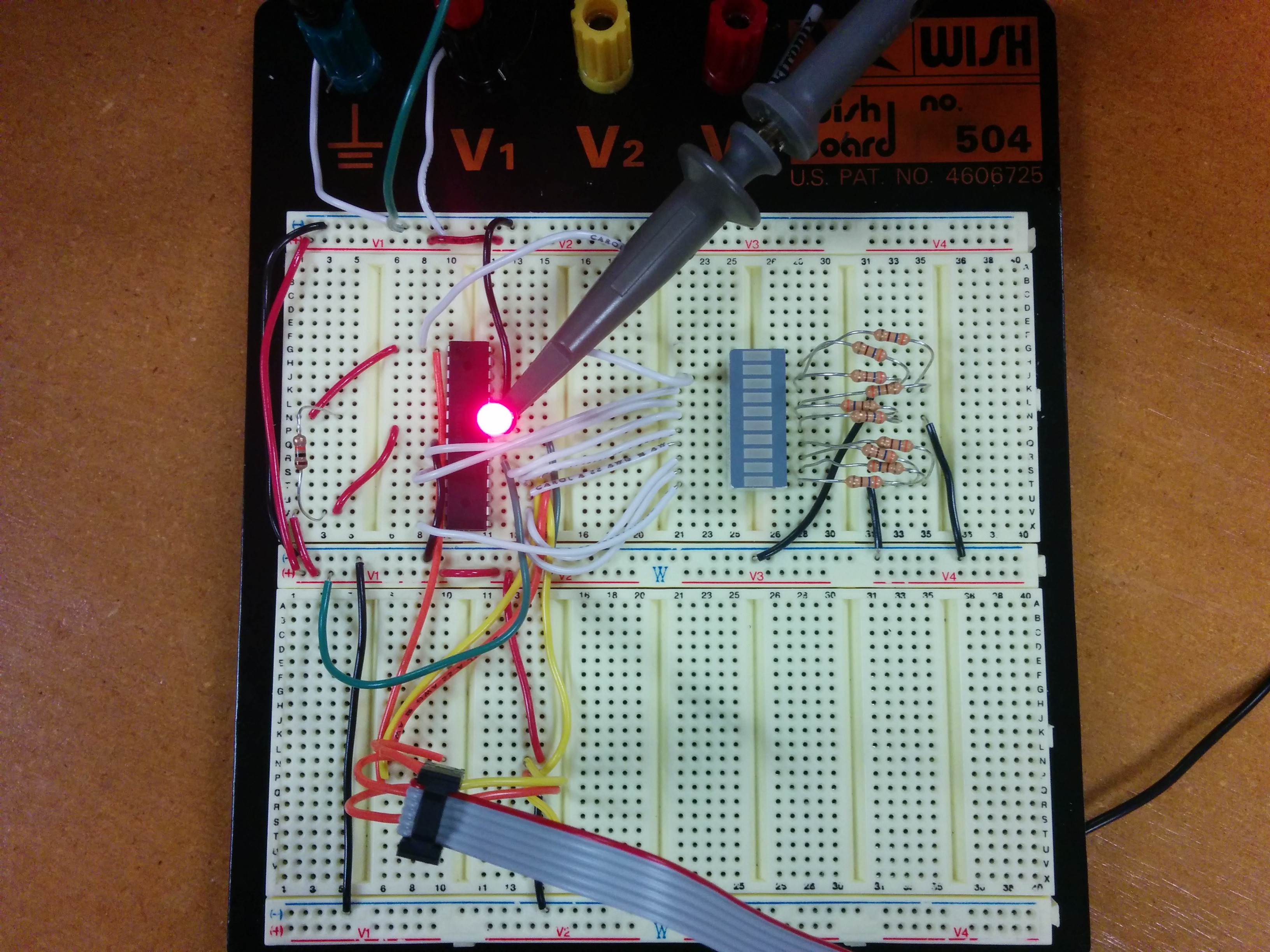


Program after entering the interrupt.

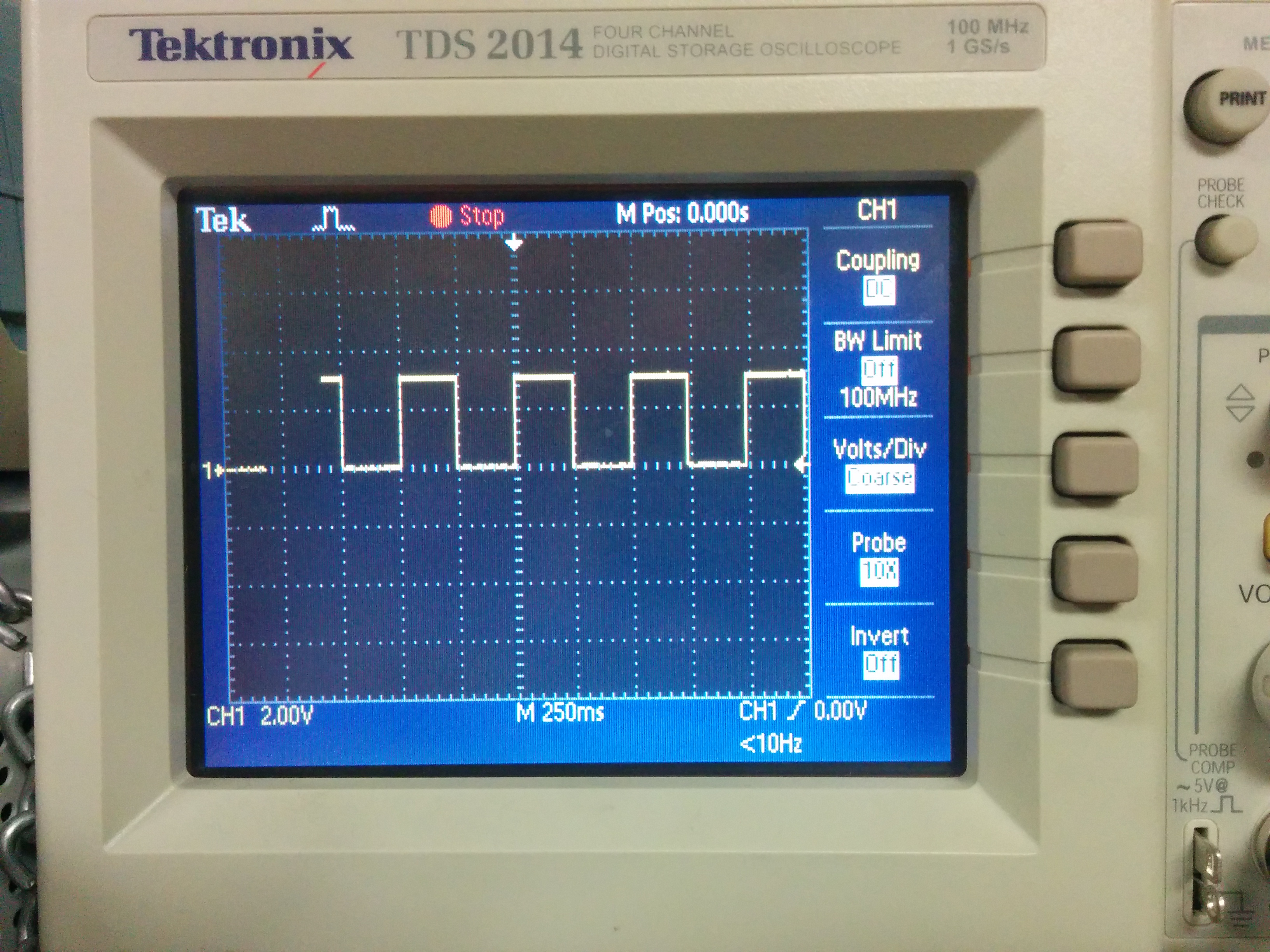


|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |

Task 1A and Task 1B : LED Turns on every .25 seconds



Task 1A and 1B: Verifying that the waveform produced is a square wave with a 50% duty cycle and a period of approximately .5 seconds.



The Output for Tasks 2A, 2B, 3A, 3B, 4A, and 4B may be seen In the video.

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| https://youtu.be/3m3Ix4uwnKY | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| https://github.com/Anguian3/anguian3-submissions | | | |

**Student Academic Misconduct Policy**

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

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

Dominique Anguiano