Martin Jaime

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 1/A |  |  |
| 2. | INCREMENTAL CODE OF TASK 2 |  |  |
| 3. | INCREMENTAL CODE OF TASK 3 |  |  |
| 4. | INCREMENTAL CODE OF TASK 4 |  |  |
| 5. | INCREMENTAL CODE OF TASK 5 |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | Github LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

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

1. Atmega328P
2. Pololu USB AVR Programmer with USB mini cable
3. LEDs
4. 100 resistors
5. 10-bit LED block
6. Breadboard

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

Design a delay subroutine to generate a waveform on PORTC.0 with 50% DC and 0.5 sec period.

Assembly code:

.macro initstack

ldi r16, high(ramend)

out sph, r16

ldi r16, low(ramend)

out spl, r16

.endmacro

initstack

sbi ddrc, 0 ;; Set first pin as output

ldi r17, 0

ldi r16, 0x01

out portc, r16 ;; Set PC0 high.

start:

call delay

eor r17, r16 ;; toggle value written to PC0

out portc, r17 ;; output toggled value to PC0

rjmp start ;; Restart loop

;; Delay routine that delays for a 0.25 s

delay:

;; Set timer/counter1 to 0x85EE

push r20 ;; store r20 and restore before return.

ldi r20, high(34286)

sts tcnt1h, r20

ldi r20, low(34286)

sts tcnt1l, r20 ; tcnt1 = 34286

ldi r20, 0x00 ;

sts tccr1a, r20 ; wgm11:10 = 0x00. Normal operation

ldi r20, 0x03

sts tccr1b, r20 ; cs1 = 0x03. Prescaler = 64

again:

in r20, tifr1 ; read tifr

sbrs r20, tov1 ; if tov1 is set, skip next instruction

rjmp again

ldi r20, 0x00

sts tccr1b, r20 ; stop timer1. Resetting CS1

ldi r20, 0x01

out tifr1, r20 ; clear tov1 flag

pop r20

ret

C code:

#define *F\_CPU* 8000000UL //XTAL = 8 MHz

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRC = 1<<DDC0; // Set PC0 as an output

PORTC = 1<<PORTC0; // Set PC0 high

while (1)

{

*\_delay\_ms*(250); // delay for 0.25 s

PORTC ^= 1<<PORTC0; // toggle PC0

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INCREMENTAL CODE OF TASK 2 |  |  |

Implement an 8-bit counter to count on every rising edge of the above waveform. The state of the counter needs to be displayed (display 8 bits only) on a 10-bit LED bar connected to PORTB. Do not worry about the counter overflow.

Assembly code:

.def zero = r0

.macro initstack

ldi r16, high(ramend)

out sph, r16

ldi r16, low(ramend)

out spl, r16

.endmacro

initstack ;; initiate stack.

sub zero, zero ;; make zero register.

ldi r20, 0 ;; initiate counter.

; sbi ddrc, 0 ;; make PC0 an output.

ldi r16, 0xFF ;;

out ddrb, r16 ;; make all of portB an output.

ldi r17, 0

ldi r16, 0x01

; out portc, r16 ;; output initial value of 1.

start:

call delay

eor r17, r16 ;; toggle output value.

sbrc r17, 0 ;; if (r17 == 1)

inc r20 ;; r20++

; out portc, r17 ;; output 1/0

out portb, r20 ;; output count to portB

rjmp start

;; Delay routine that delays for a 0.25 s

delay:

;; Set timer/counter1 to 0x85EE

push r20 ;; store r20 and restore before return.

ldi r20, high(34286)

sts tcnt1h, r20

ldi r20, low(34286)

sts tcnt1l, r20 ; tcnt1 = 34286

ldi r20, 0x00 ;

sts tccr1a, r20 ; wgm11:10 = 0x00. Normal operation

ldi r20, 0x03

sts tccr1b, r20 ; cs1 = 0x03. Prescaler = 64

again:

in r20, tifr1 ; read tifr

sbrs r20, tov1 ; if tov1 is set, skip next instruction

rjmp again

ldi r20, 0x00

sts tccr1b, r20 ; stop timer1. Resetting CS1

ldi r20, 0x01

out tifr1, r20 ; clear tov1 flag

pop r20

ret

C code:

#define *F\_CPU* 8000000UL //XTAL = 8 MHz

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

unsigned char x = 1<<DDC0; // waveform var

unsigned char counter = 0; // counter to output

DDRB = 0xFF; // set PORTB as output

PORTB = counter; // output the count to PORTB

while (1)

{

*\_delay\_ms*(250); // delay for 0.25 s

x ^= 1<<PORTC0; // toggle x

if (x == 1) // if x is on rising edge,

PORTB = ++counter; // increment count.

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL CODE OF TASK 3 |  |  |

Also connect the 9th and 10th bit of the LED bar to PORTC.5 and PORTC.6 pins. Toggle PORTC.5 and PORTC.6 for every 5th rising pulses and every 10th rising pulse of the counter respectively.

**Assembly code:**

.def zero = r0

.def count5 = r25

.def five = r18 ;; bit 5 set

.def four = r19 ;; bit 6 set

.def outc = r21

.macro initstack

ldi r16, high(ramend)

out sph, r16

ldi r16, low(ramend)

out spl, r16

.endmacro

initstack ;; initiate stack.

ldi four, 0x10

ldi five, 0x20

sub zero, zero ;; make zero register.

ldi r20, 0 ;; initiate counter.

ldi count5, 0 ;; count5 = 0.

ldi r16, 0x30

out ddrc, r16 ;; make PC5 and PC6 an output.

ldi r16, 0xFF ;;

out ddrb, r16 ;; make all of portB an output.

out portb, r20 ;; output 0x00 to portB

ldi r17, 0

ldi r16, 0x01

; out portc, r16 ;; output initial value of 1.

start:

call delay

eor r17, r16 ;; toggle output value.

breq notRising ;; if (r17 != 0)

inc r20 ;; r20++

inc count5

out portb, r20 ;; output count to portB

cpi count5, 5

brne notRising

clr count5

in r1, portc ;; read pinC

eor r1, five

out portc, r1

sbrc r1, 5

rjmp notRising

eor r1, four

out portc, r1

; out portc, r17 ;; output 1/0

notRising:

rjmp start

;; Delay routine that delays for a 0.25 s

delay:

;; Set timer/counter1 to 0x85EE

push r20 ;; store r20 and restore before return.

ldi r20, high(34286)

sts tcnt1h, r20

ldi r20, low(34286)

sts tcnt1l, r20 ; tcnt1 = 34286

ldi r20, 0x00 ;

sts tccr1a, r20 ; wgm11:10 = 0x00. Normal operation

ldi r20, 0x03

sts tccr1b, r20 ; cs1 = 0x03. Prescaler = 64

again:

in r20, tifr1 ; read tifr

sbrs r20, tov1 ; if tov1 is set, skip next instruction

rjmp again

ldi r20, 0x00

sts tccr1b, r20 ; stop timer1. Resetting CS1

ldi r20, 0x01

out tifr1, r20 ; clear tov1 flag

pop r20

ret

**C code:**

#define *F\_CPU* 8000000UL //XTAL = 8 MHz

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

unsigned char x = 1<<DDC0; // waveform var

unsigned char counter = 0; // counter to output

// I decided to use pinC 4 and 5 instead since pin 6 is used for reset.

DDRC = (1<<DDC4) | (1<<DDC5);

DDRB = 0xFF; // set PORTB as output

PORTB = counter; // output the count to PORTB

while (1)

{

*\_delay\_ms*(250); // delay for 0.25 s

x ^= 1<<PORTC0; // toggle x

if (x == 1) { // if x is on rising edge,

PORTB = ++counter; // increment count.

if (counter % 5 == 0) { // if counter is div by 5,

PORTC ^= (1<<PINC5); // Toggle pinC5

if (counter % 10 == 0) // if also div by 10,

PORTC ^= (1<<PINC4); // toggle pinC4

}

} // end if(x == 1)

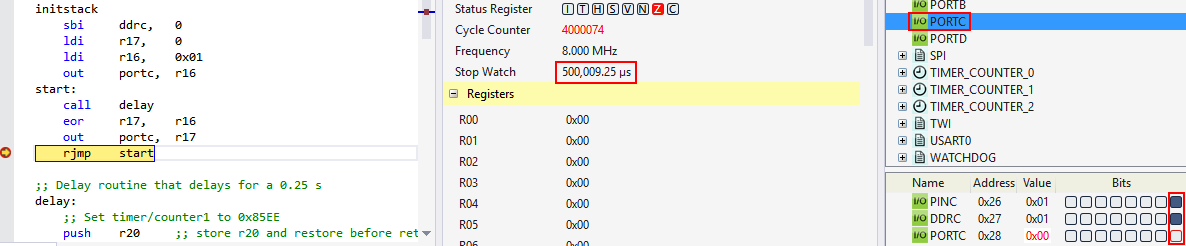
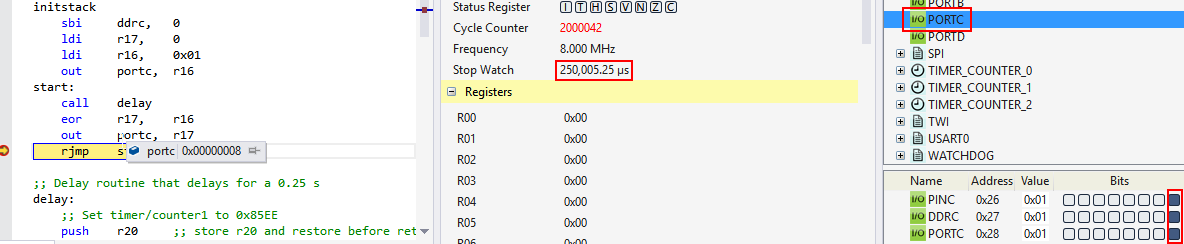
} // end while

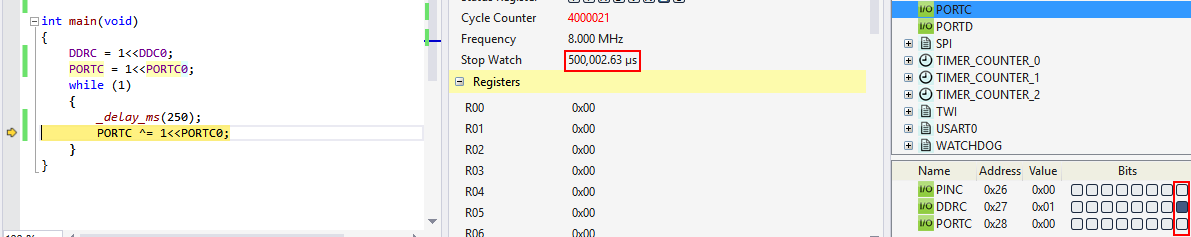
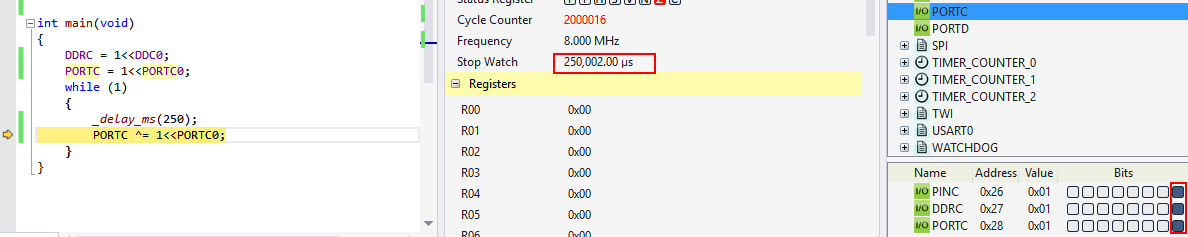
}

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

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

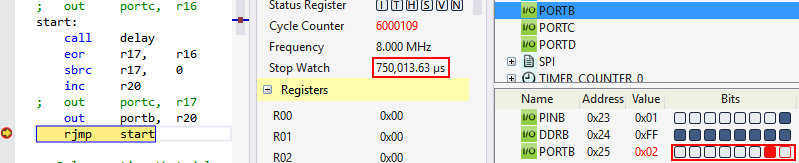
**TASK 1**: Verify duty cycle and period: 50% duty cycle, period = 0.5 second

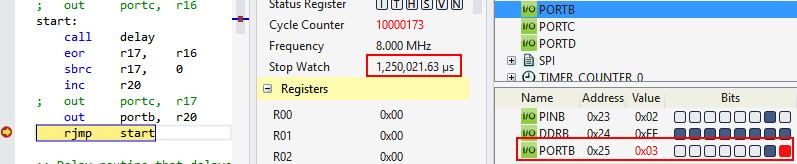
In assembly:

In C:

**TASK 2**: Implement an 8-bit counter to count on every rising edge of the above waveform. The state of the counter needs to be displayed (display 8 bits only) on a 10-bit LED bar connected to PORTB. Do not worry about the counter overflow.

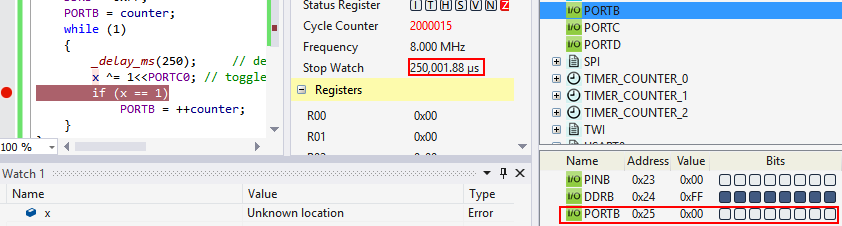
**Assembly Code:**

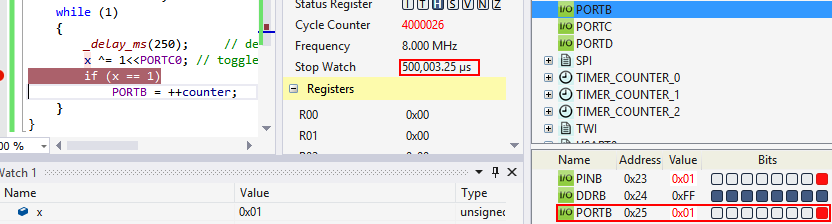




The counter counts up after one period.

**C code:**



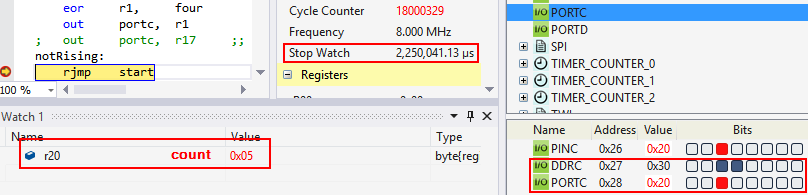


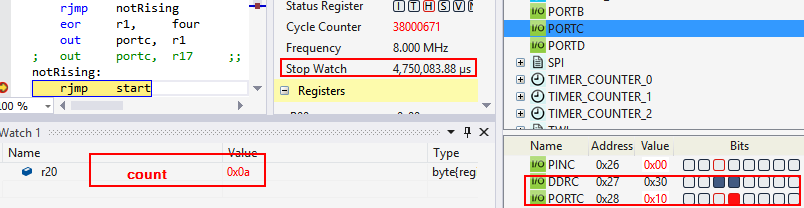
**TASK 3:** Also connect the 9th and 10th bit of the LED bar to PORTC.5 and PORTC.6 pins. Toggle PORTC.5 and PORTC.6 for every 5th rising pulses and every 10th rising pulse of the counter respectively.

For this task, I decided to use pins 4 and 5 from port C to avoid disabling the RSTDISBL fuse on the atmega328p. When I enabled it on one chip, I was not able to enter programming mode again, unless I used high voltage programming.

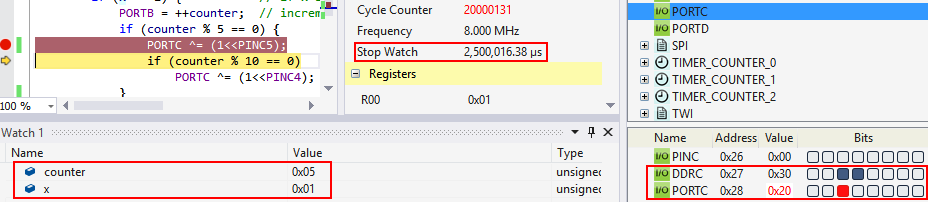
**Assembly code**:

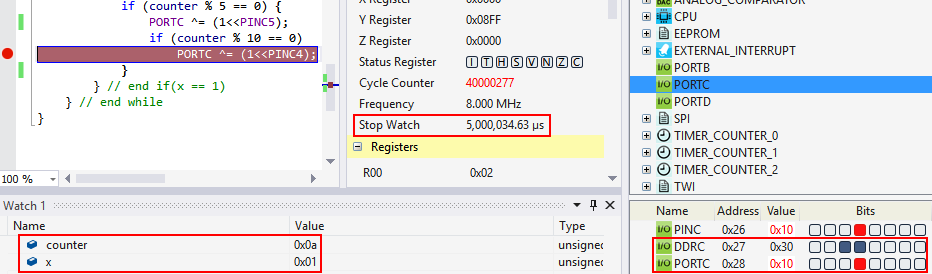
When pin 5 is toggled.





**C code:**

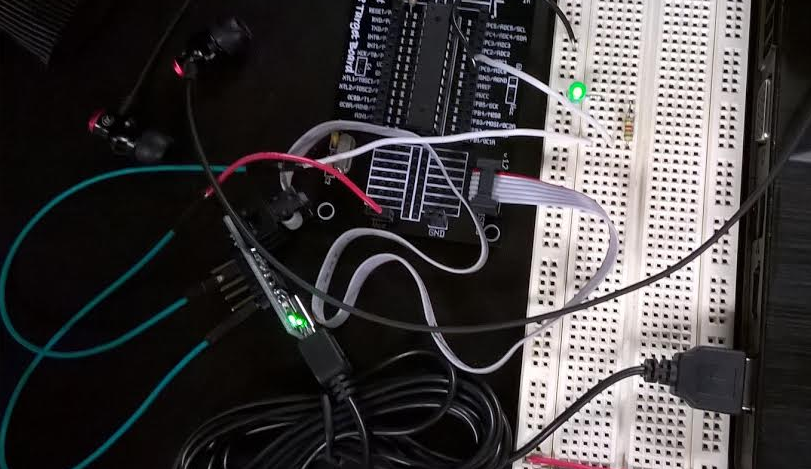




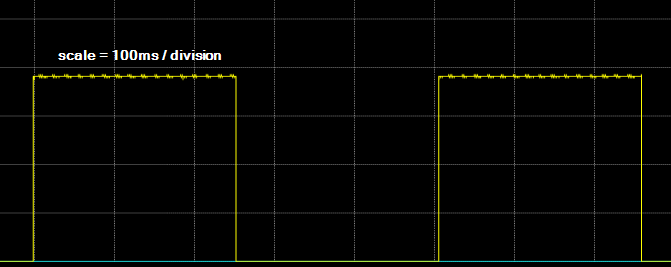
**Task 4:**

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

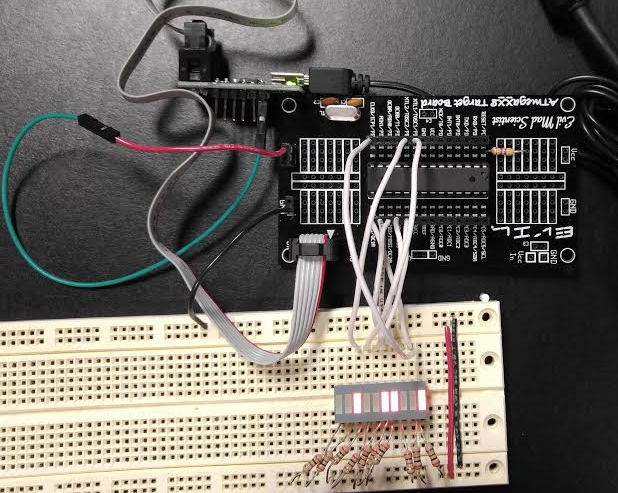
Task 1: LED blinks with a period of 0.5 second. It is on for 0.25 s, and off for 0.25 s.



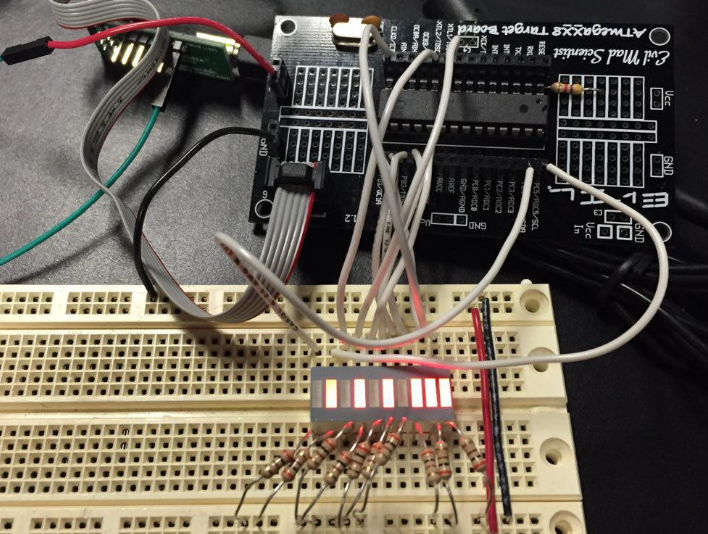
Signal output: as measured by Pololu Slo Scope



Task 2: 8-bit counter displayed on PortB through a 10-bit LED bar.



Task 3: Toggle pins 5 and 6 on port C every 5th count and every 10th count respectively. To avoid overriding the reset pin, I used pin 4 on port C instead of pin 6.



|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| All videos can be found on the following Google Drive link. Playable on a browser.  <https://drive.google.com/folderview?id=0B4ItVBjMqlnyVHdOS0d3V25nbm8&usp=sharing> | | | |
| 10. | GITHUB LINK OF THE DA |  |  |
| <https://github.com/martinjaime/CpE301_Assignments2016S.git> | | | |

**Student Academic Misconduct Policy**

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

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

Martin Jaime