CPE301 – SPRING 2019

Design Assignment 2C Part 1

Student Name: Benjamin Johnson

Student #: 5003284489

Student Email: johnsb18@unlv.nevada.edu

Primary Github address: https://github.com/johnsb18/ClassRepository

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

// Part1\_Task1

#define *F\_CPU* 16000000UL

#include <avr/io.h>

int main(void)

{

DDRB |= (1<<2); // set PORTB.2 for output (LED)

TCCR0A = 0; // normal operation

TCCR0B |= (1<<CS02) | (1<<CS00); // prescaler = 1024

int count = 0; // overflow counter

while (1) {

count = 0; // reset counter

TCNT0 = 0x00; // reset timer

while (count < 27){ // loop until 60% DC is met

if(TCNT0 == 0xFF){ // if timer overflows

count++; // increment counter

TCNT0 = 0x00; // reset timer

}

}

PORTB ^= (1<<2); // toggle LED off

count = 0; // reset counter

TCNT0 = 0x00; // reset timer

while (count < 18){ // loop until remaining 40% DC is met

if(TCNT0 == 0xFF){ // if timer overflows

count++; // increment counter

TCNT0 = 0x00; // reset timer

}

}

PORTB ^= (1<<2); // toggle LED on

}

}

// Part1\_Task2

#define *F\_CPU* 16000000UL

#include <avr/io.h>

int main(void)

{

DDRC &= (0<<2); // connect PORTC.2 to switch as input

PORTC |= (1<<2); // enable pull-up

DDRB |= (1<<2); // set PORTB.2 for output (LED)

TCCR0A = 0; // normal operation

TCCR0B |= (1<<CS02) | (1<<CS00); // prescaler = 1024

int count = 0; // overflow counter

while (1) {

if(!(PINC & (1<<PINC2))) { // check if pin in low

PORTB &= ~(1<<2); // set LED on

count = 0; // reset counter

TCNT0 = 0x00; // reset timer

while(count < 77) { // loop until 1.25 sec delay met

if(TCNT0 == 0xFF){ // if timer overflows

count++; // increment counter

TCNT0 = 0x00; // reset timer

}

}

}

else {

PORTB |= (1<<2); // set LED off

}

}

return 0;

}

// Part2\_Task1

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

*uint8\_t* OVFCOUNT = 0; // overflow count multiplier

*uint8\_t* OVFLIMIT = 18; // overflow count limit, set for 40% of DC

int main(void)

{

DDRB |= (1<<2); // set PORTB.2 for output (LED)

TCCR0A = 0; // normal operation

TCCR0B |= (1<<CS02) | (1<<CS00); // prescaler = 1024

TCNT0 = 4; // timer value for .725 sec

TIMSK0 = (1<<TOIE0); // enable timer0 overflow interrupt

sei(); // enable interrupts

while (1) { // wait for timer interrupt

}

}

ISR (TIMER0\_OVF\_vect)

{

OVFCOUNT++; // increment counter

if (OVFCOUNT == OVFLIMIT) {

PORTB ^= (1<<2); // toggle LED

if(OVFLIMIT == 18) {

OVFLIMIT = 27; // set ovf limit for 60% of DC

} else {

OVFLIMIT = 18; // set ovf limit for 40% of DC

}

OVFCOUNT = 0; // reset counter

}

TCNT0 = 4; // timer value for .725 sec

}

// Part3\_Task1

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

*uint8\_t* OVFCOUNT = 0; // overflow count multiplier

*uint8\_t* OVFLIMIT = 18; // overflow count limit, set for 40% of DC

int main(void)

{

DDRB |= (1<<2); // set PORTB.2 for output (LED)

PORTB |= (1<<2); // set LED off

OCR0A = 252; // TOP = 256 - 4

TCCR0B |= (1<<WGM02) | (1<<CS02) | (1<<CS00); // prescaler = 1024, CTC mode

TIMSK0 = (1<<OCIE0A); // enable timer0 compare match int

TCNT0 = 0; // initialize timer to 0

sei(); // enable interrupts

while (1) { // wait for interrupt

}

}

ISR (TIMER0\_COMPA\_vect)

{

OVFCOUNT++; // increment counter

if (OVFCOUNT == OVFLIMIT) {

PORTB ^= (1<<2); // toggle LED

if(OVFLIMIT == 18) {

OVFLIMIT = 27; // set overflow limit for 40% DC

} else {

OVFLIMIT = 18; // set overflow limit for 60% DC

}

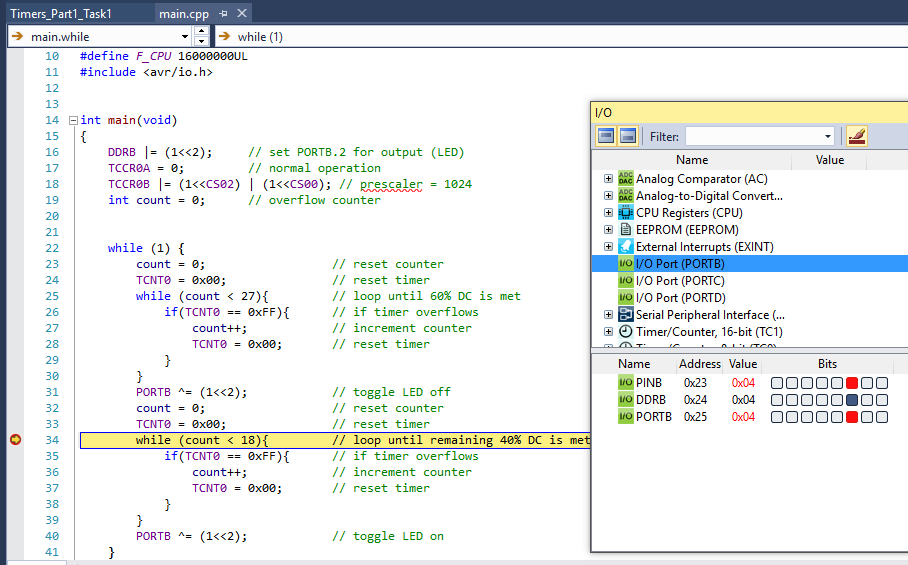
OVFCOUNT = 0; // reset counter

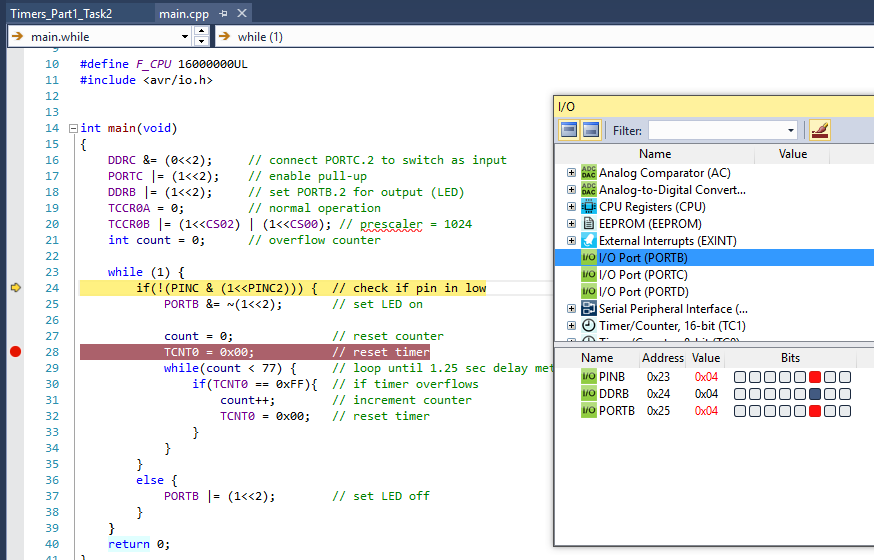
}

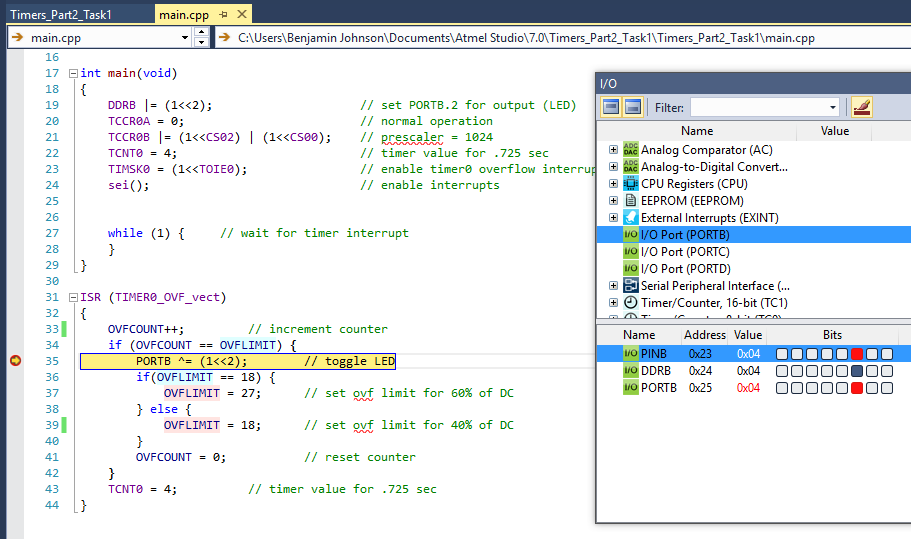
TCNT0 = 0; // reset timer

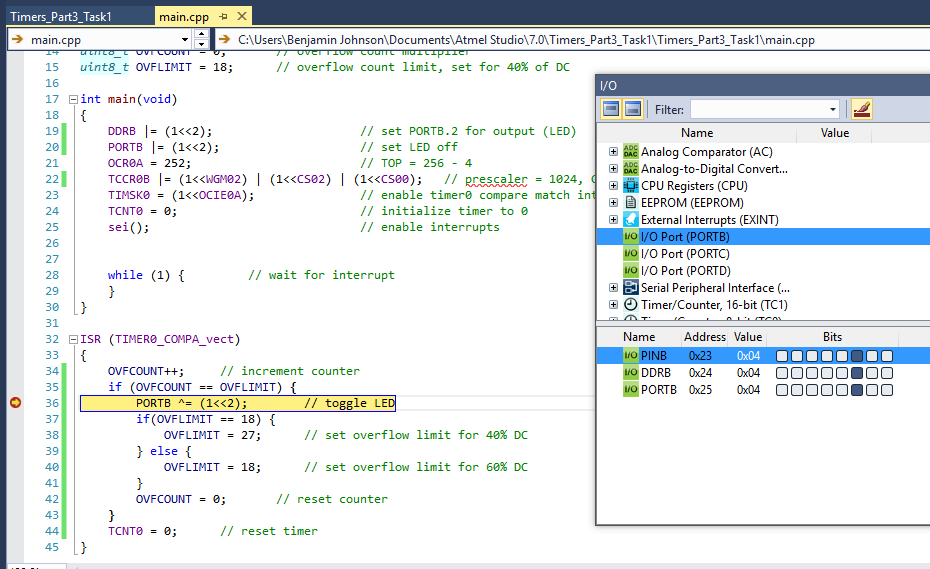
}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**
2. **SCHEMATICS**
3. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**









1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**
2. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/5sN--0Uo614

https://youtu.be/8FwetAzQMVs

1. **GITHUB LINK OF THIS DA**

https://github.com/johnsb18/ClassRepository/tree/master/DesignAssignments/DA2C

**Student Academic Misconduct Policy**

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

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

Benjamin Johnson