CPE301 – SPRING 2019

Design Assignment 2C

Student Name: Clayton Higbee

Student #: 5001189864

Student Email: higbee@unlv.nevada.edu

Primary Github address: <https://github.com/claytonjhigbee/CODSWORTH_MAIN>

Directory: <https://github.com/claytonjhigbee/CODSWORTH_MAIN/tree/master/DA/DA2C>

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

Components used:

Atmega328P Xplained Mini Board

Female to Male Wires (For logic analyzer Connections)

Logic Analyzer

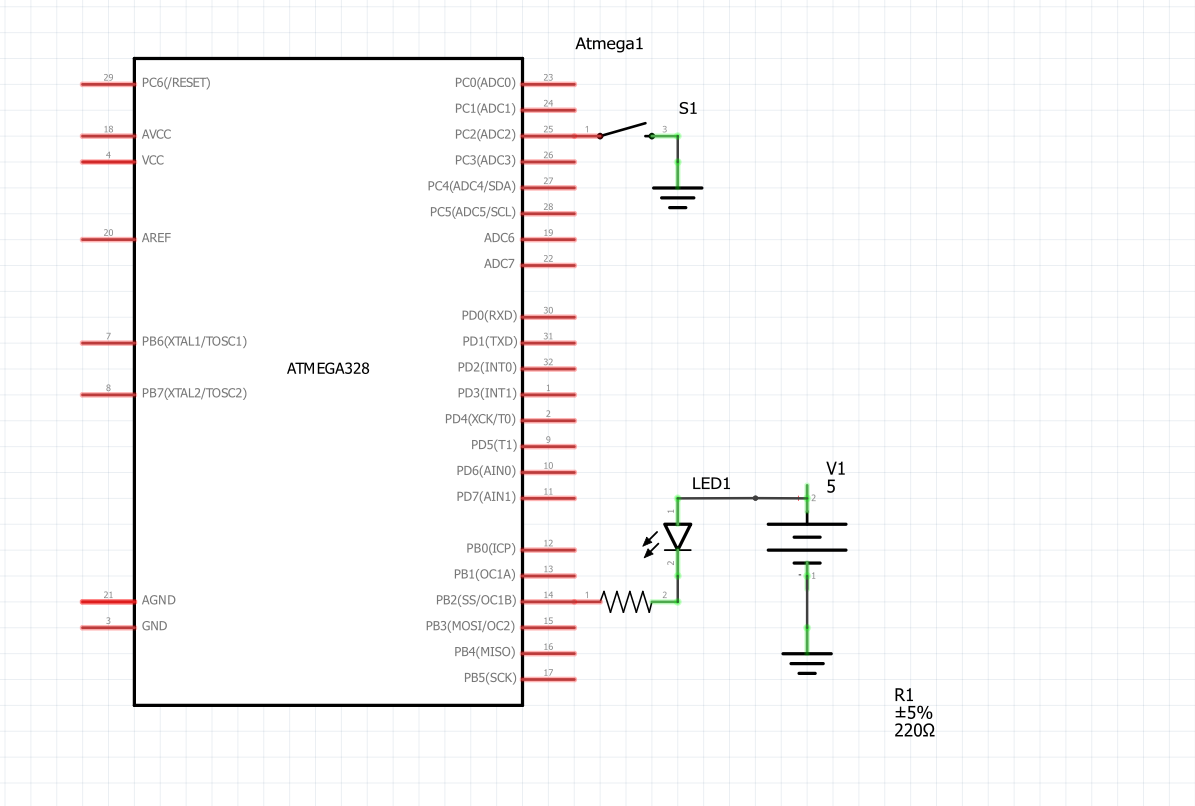
Multi-Function Development Board Shield

Block diagram with pins used in the Atmega328P

(Schematic only for Task 2 as Task 1 required no parts)

PortC.2 – Input for Switch

PortB.2 – Output for LED, used with Danger Shield



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

Task 1.1

/\*

\* DA2C\_T1.c

\*

\* Created: 3/15/2019 10:24:15 PM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#define LED 2

// from vanki

int main() {

*uint8\_t* OVFCount = 0; // Setup Placeholder for overflow counter

DDRB |= 1 << LED; // Configure PORTB.2 as an output

TCNT0 = 0x4; // Reset timer0 for first delay

TCCR0B = (1<<CS02)|(1<<CS00); // Setting prescaler value for timer0 to 1024 and start timer

while (1){

while((TIFR0 & 0x01) == 0){} // When the flag is not set, do nothing and wait

OVFCount++; // Increment overflow flag counter

// Count up during the first delay period

if(OVFCount < 27){

TCNT0 = 0x4; // Reset timer0 for first delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

// If count is met, set PinB.2 low

if(OVFCount == 27) {

PORTB = 0x00; // Set PinB.2 Low

}

// Count up for next delay

if(27 < OVFCount < 45) {

TCNT0 = 0x4; // Reset timer0 for second delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

// If count met, set high and start delay period over

if(OVFCount == 45) {

PORTB = 0xFF; // Set PinB.2 high

OVFCount = 0; // Start delay counter over

}

}

}

Task 1.2

/\*

\* DA2C\_T12.c

\*

\* Created: 3/16/2019 9:25:11 AM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

*uint8\_t* OVFCount = 0; // Setup Placeholder for overflow counter

DDRB = 0xFF; // Set Port B as an output

DDRC = 0x00; // Set Port C as an input

PORTC = 0xFF; // Set Pull up resistors in Port C

TCNT0 = 0x2; // Reset timer0 for delay

TCCR0B = (1<<CS02)|(1<<CS00); // Setting prescaler value for timer0

while (1)

{

OVFCount = 0; // Retain Overflow count to zero if the button has not been pushed

PORTB = 0b00111100; // Set LED Port B to Low (REVERSE LOGIC)

if(PINC == 0x7D) // If switch push is detected, then set LEDs and begin timer delay sequence

{

TCNT0 = 0x3; // Reset timer0 for delay

PORTB = 0b00000000; // Set LED Port B to High (REVERSE LOGIC)

while(OVFCount < 77){

while((TIFR0 & 0x01) == 0){

// When the flag is not set, do nothing O.o and wait

}

OVFCount++; // When flag is set, increment overflow flag counter

if(OVFCount < 77){

TCNT0 = 0x3; // Reset timer0 for delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

}

}

}

}

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 2 from Task 1**

Task 2.1

/\*

\* DA2C\_T2.c

\*

\* Created: 3/16/2019 11:23:22 AM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

volatile int OVFCount; // Setup global overflow counter

int main(void){

OVFCount = 0; // Initialize global overflow counter

DDRB = 0b100; // Configure PORTB.2 as an output

TIMSK0 |= (1<<TOIE0); // Activate Timer 0 Interrupt Flag

TCNT0 = 56; // Desired timer initial value to count up to 255

sei(); // Activate global interrupts

TCCR0B |= (1<< CS02)|(1<<CS00); // Set prescalar to 1024 and starts timer

while(1){

// Do nothing and wait for the interrupt to take place O.o

}

}

// Interrupt C Code

ISR(TIMER0\_OVF\_vect) // Timer0 overflow interrupt sequence

{

OVFCount++; // Increment overflow flag counter

int count; // Initialize local counter

count = OVFCount; // Let local counter mimic global counter variable

// First delay period

if(count < 27){

TCNT0 = 0x4; // Reset timer0 for first delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

// If first delay done, then set PinB.2 low

if(count == 27) {

PORTB = 0x00;

}

// Second delay period

if(27 < count & count < 45) {

TCNT0 = 0x4; // Reset timer0 for second delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

// If second delay done, set PinB.2 low and start counter over

if(count == 45) {

PORTB = 0xFF; // Set PinB.2 Low

OVFCount = 0; // Reset Counter

}

}

Task 2.2

/\*

\* DA2C\_T22.c

\*

\* Created: 3/16/2019 12:38:34 PM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

volatile int OVFCount;

int main(void){

OVFCount = 0; // initialize Overflow counter

DDRB = 0xFF; // Configure PORTB as an output

TIMSK0 |= (1<<TOIE0); // Activate Timer 0 Interrupt Flag

TCNT0 = 56; // Desired timer initial value to count up to 255

sei(); // Activate global interrupts

TCCR0B |= (1<< CS02)|(1<<CS00); // Set prescalar to 1024 and starts timer

while(1){

PORTB = 0b11111111; // Set LED PortB.2 to Low (REVERSE LOGIC)

if(PINC == 0x7D) // If switch push is detected, then set LEDs

{

OVFCount = 0; // initialize Overflow value for this button press

TCNT0 = 0x2; // Reset timer0 for delay

PORTB = 0xFB; // Set LED Port B to High (REVERSE LOGIC)

while(OVFCount < 77)

{

// Do nothing O.o, waiting for entire overflow to occur

}

}

}

}

// Interrupt C Code

ISR(TIMER0\_OVF\_vect) // Timer0 overflow interrupt sequence

{

OVFCount++; // Increment overflow flag counter

int count;

count = OVFCount;

if(count < 77){

TCNT0 = 0x2; // Reset timer0 for delay

TIFR0 = 0x01; // Clearing Timer1 overflow flag

}

}

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 3 from Task 2**

Task 3.1

/\*

\* DA2C\_T3.c

\*

\* Created: 3/16/2019 1:11:32 PM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int OVFCount; // Initialize global overflow counter variable

int main(void)

{

OVFCount = 0; // initialize Overflow counter

DDRB = 0xFF; // Configure Port B as an output

OCR0A = 252; // Load compare register 0A Value

TCCR0A |= (1<<WGM01); // Set to CTC Mode

TIMSK0 |= (1<<OCIE2A); // Set interrupt on compare match

TCCR0B = 0b101; // Set prescalar to 1024

sei(); // Enable Global Interrupts

while(1)

{

// Do nothing O.o

}

}

ISR(TIMER0\_COMPA\_vect){

OVFCount++; // Increment global overflow flag counter

int count; // Initialize local overflow counter variable

count = OVFCount; // let local counter mimic global overflow counter

// If counter reaches 27, set PinB.2 low and continue counting

if(count == 27) {

PORTB = 0x00; // Set PinB.2 low

}

// If counter reaches 45, set PinB.2 high and restart counter

if(count == 45) {

PORTB = 0xFF; // Set PinB.2 high

OVFCount = 0; // Restart Overflow counter

}

}

Task 3.2

/\*

\* DA2C\_T32.c

\*

\* Created: 3/16/2019 3:22:39 PM

\* Author : clayt

\*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int OVFCount; // initialize global overflow counter variable

int main(void)

{

OVFCount = 0; // initialize Overflow counter

DDRB = 0xFF; // Configure Port B as an output

DDRC = 0x00; // Configure Port C as an Input

PORTC = 0xFF; // Turn Port C Pull up resistors

OCR0A = 253; // Load compare Reg Value

TCCR0A |= (1<<WGM01); // Set to CTC Mode

TIMSK0 |= (1<<OCIE2A); // Set interrupt on compare match

TCCR0B = 0b101; // Set prescalar to 1024 and starts PWM

sei(); // Enable Global Interrupts

while(1)

{

PORTB = 0b11111111; // Set LED PortB.2 to Low (REVERSE LOGIC)

if(PINC == 0x7D) // If switch push is detected, then set LEDs

{

OVFCount = 0; // initialize Overflow value for this button press

TCNT0 = 0x0; // Reset Timer for Delay

PORTB = 0xFB; // Set LED Port B to High (REVERSE LOGIC)

while(OVFCount < 77)

{

// Do nothing O.o, waiting for entire overflow count to occur

}

}

}

}

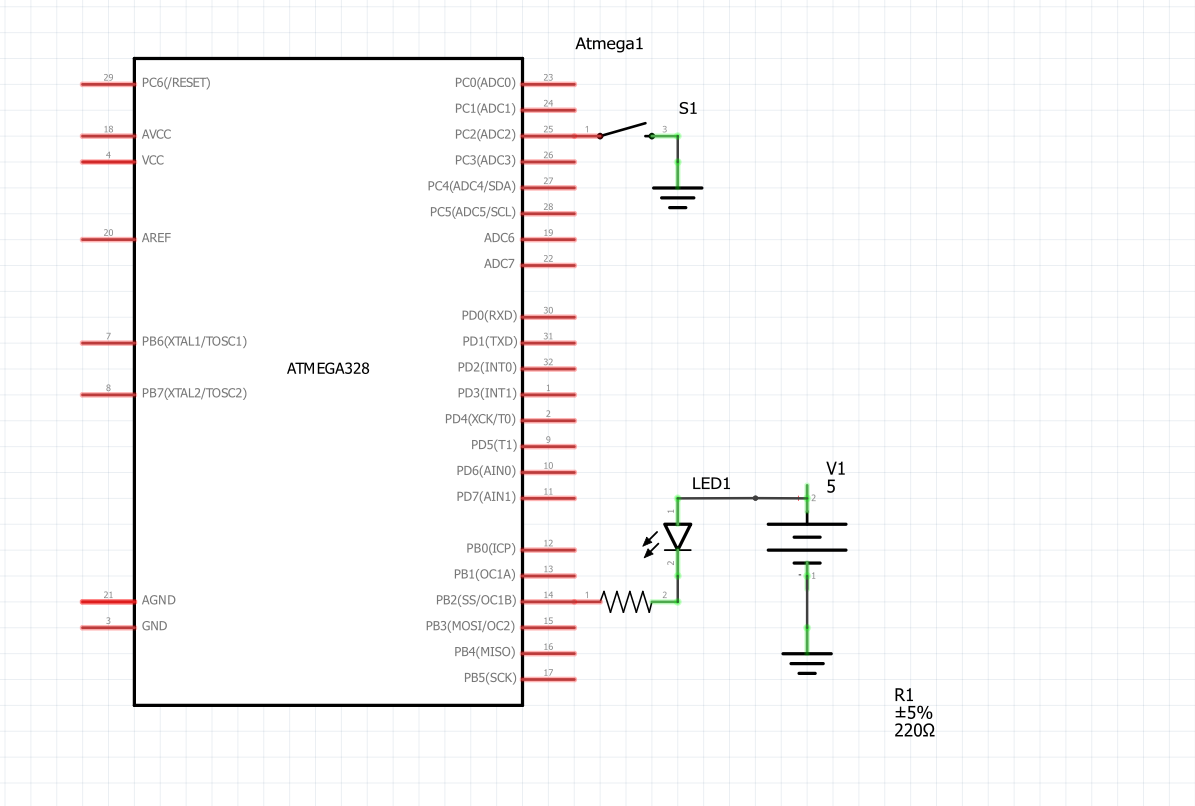
// Continue to increment counter until it reaches 77, then turn LED off

ISR(TIMER0\_COMPA\_vect){

OVFCount++; // Increment overflow flag counter

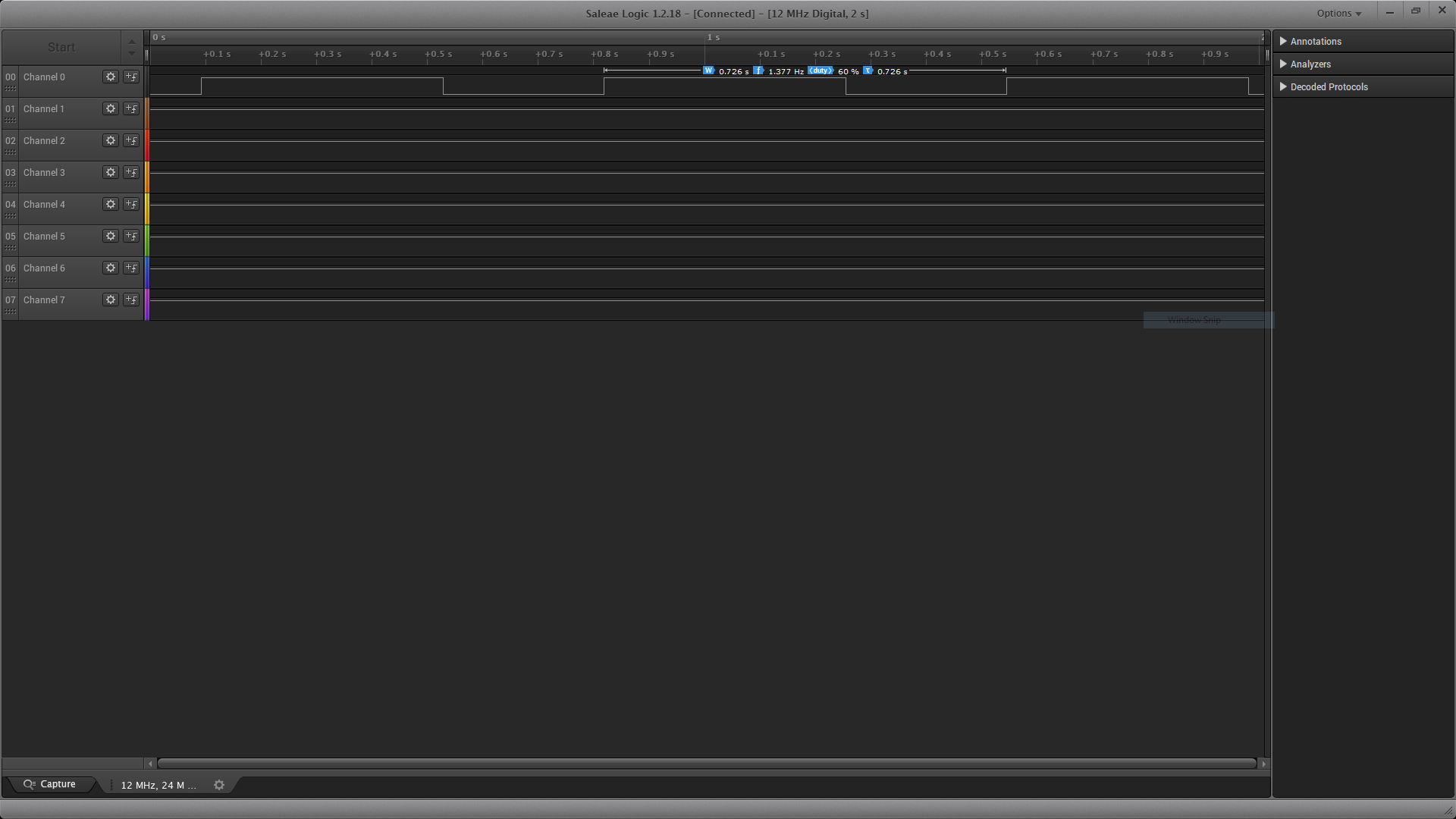
}

1. **SCHEMATICS**

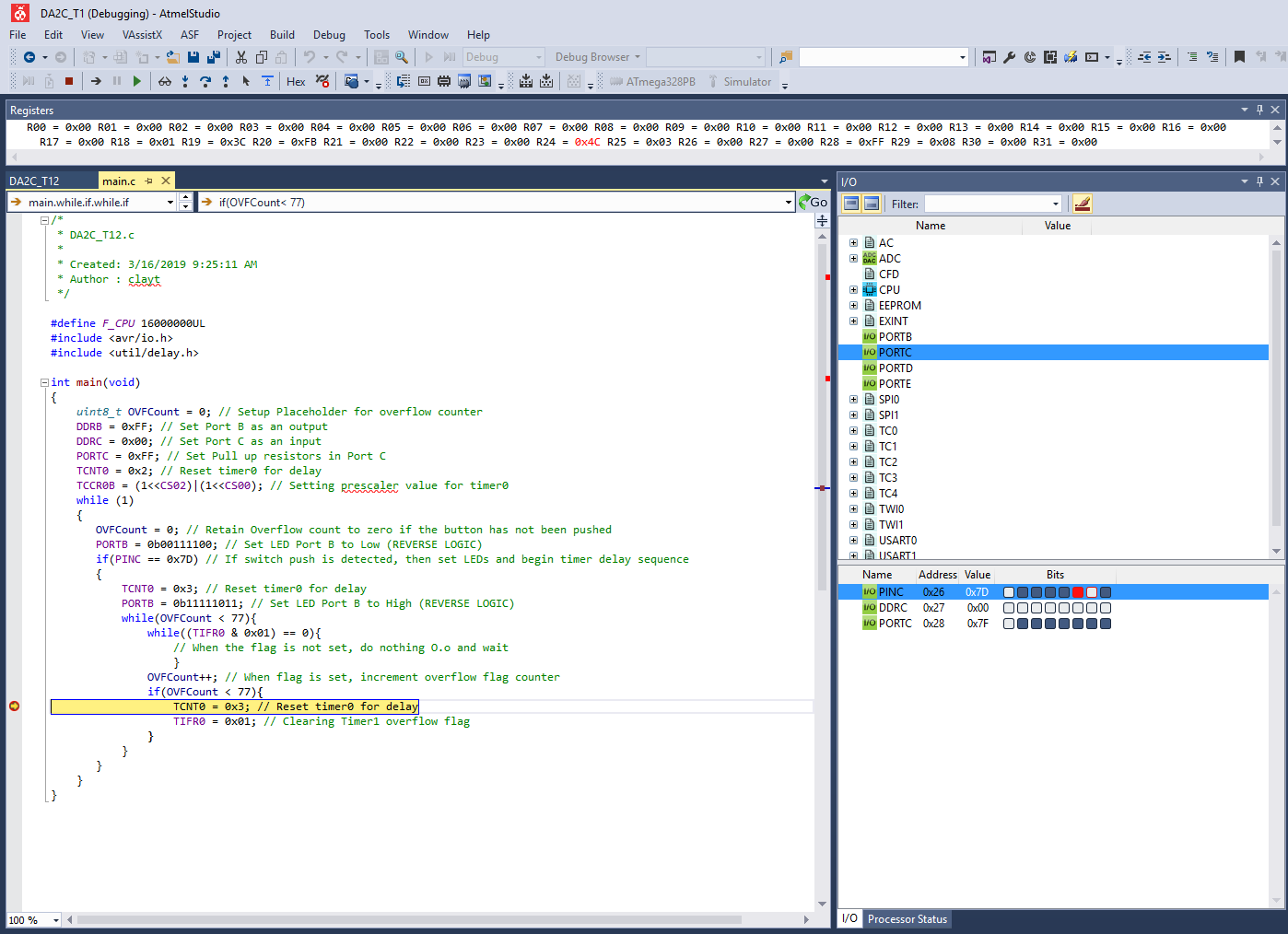


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

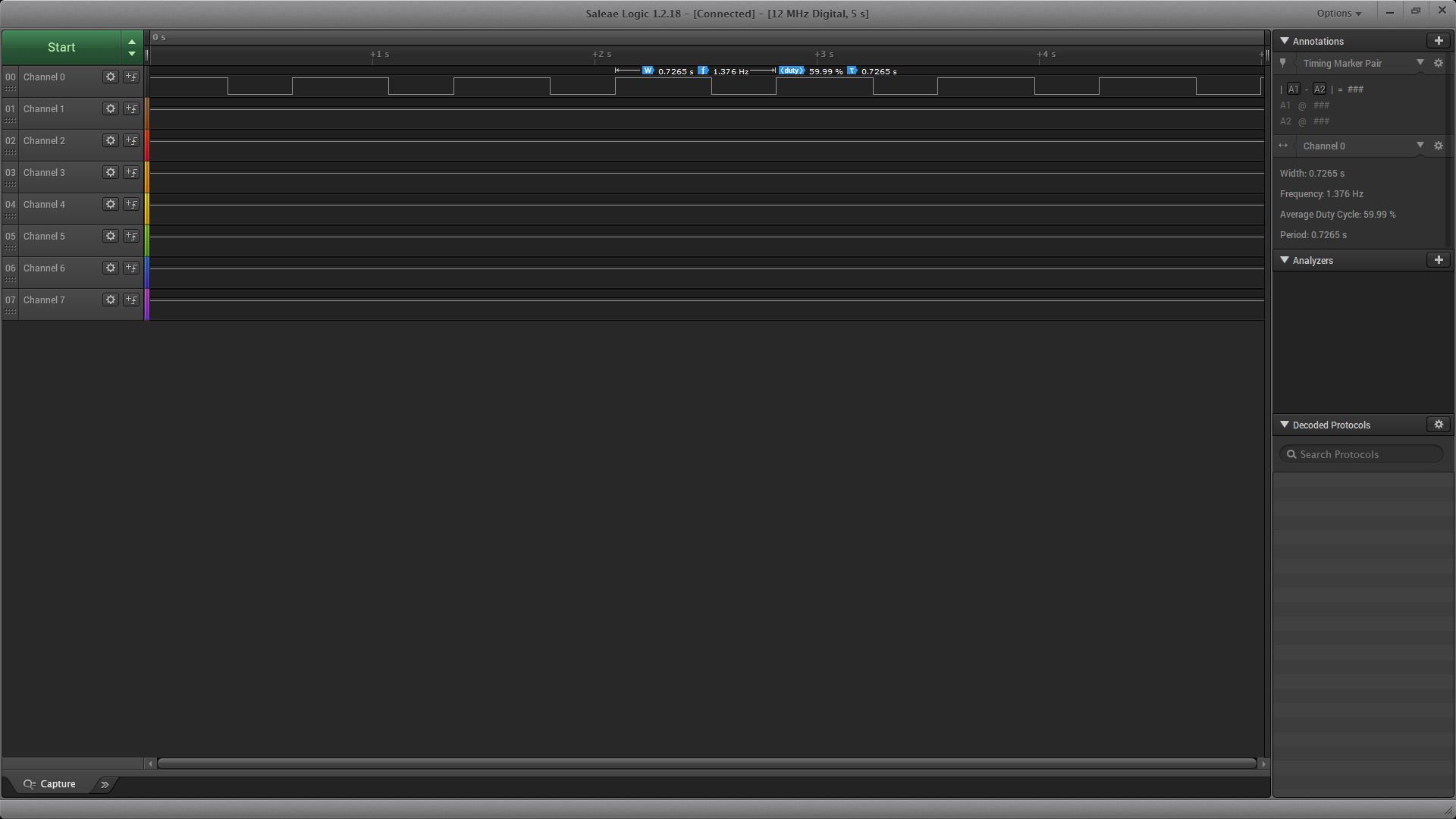
Task 1.1



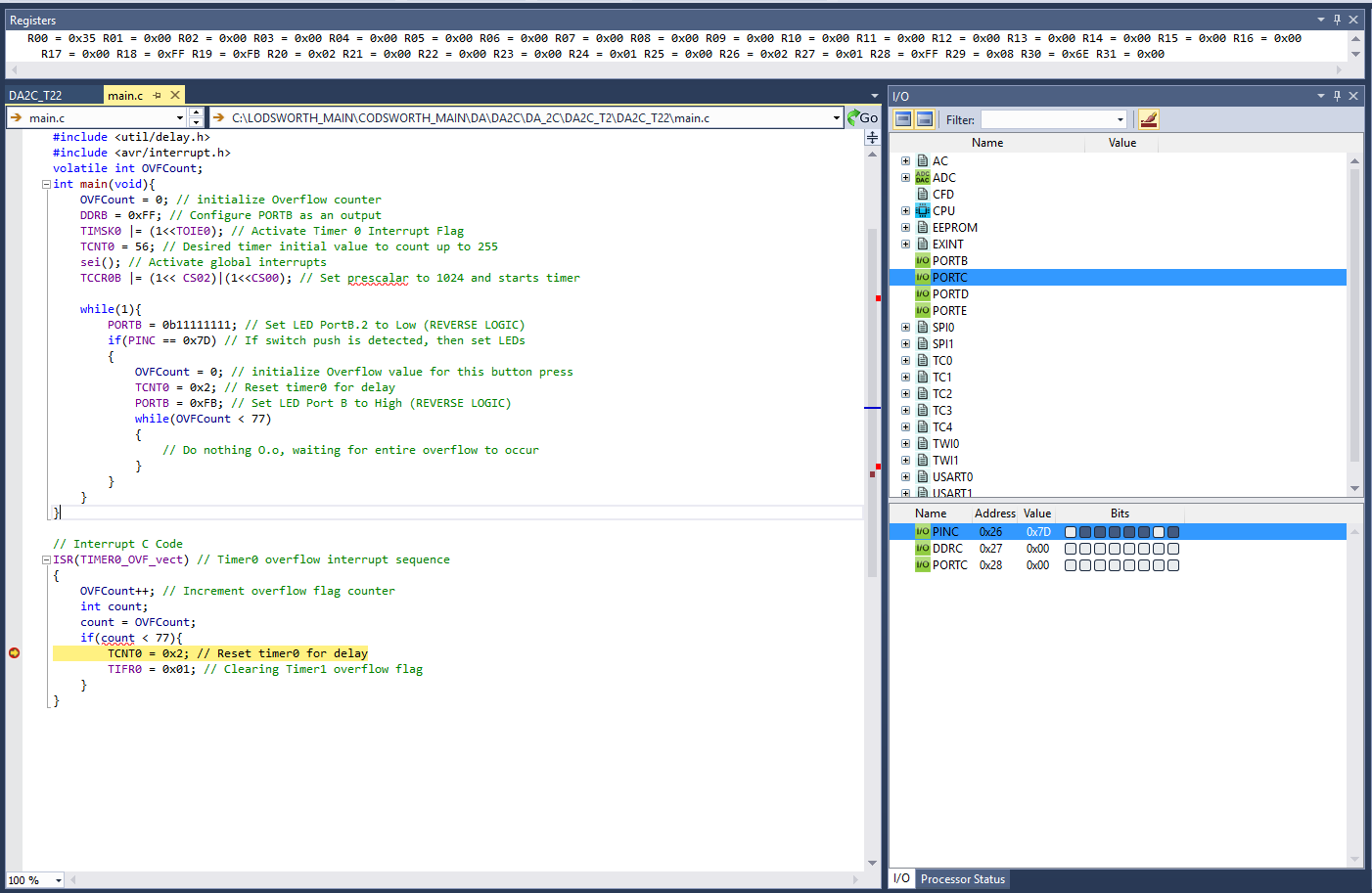
Task 1.2



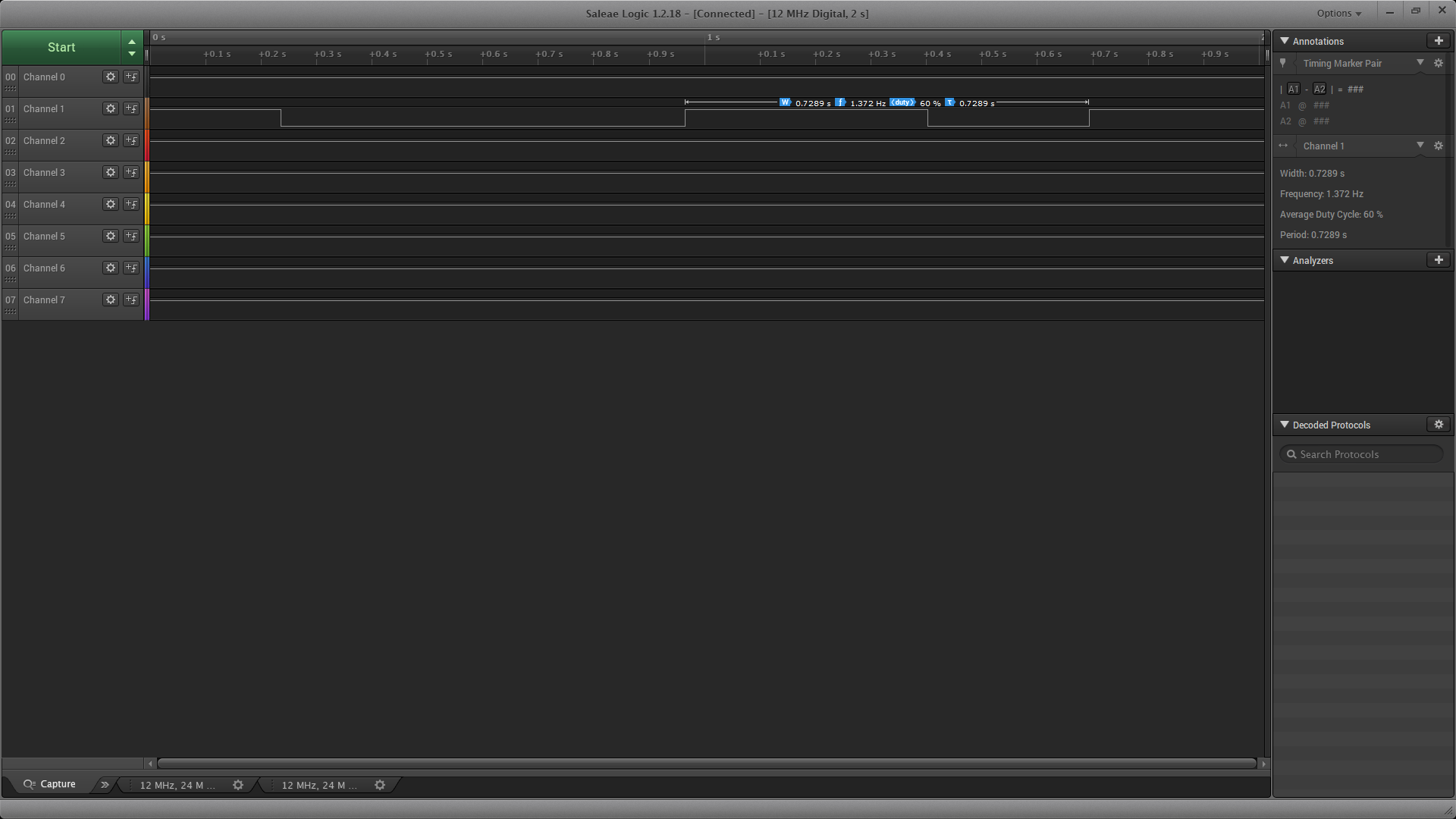
Task 2.1



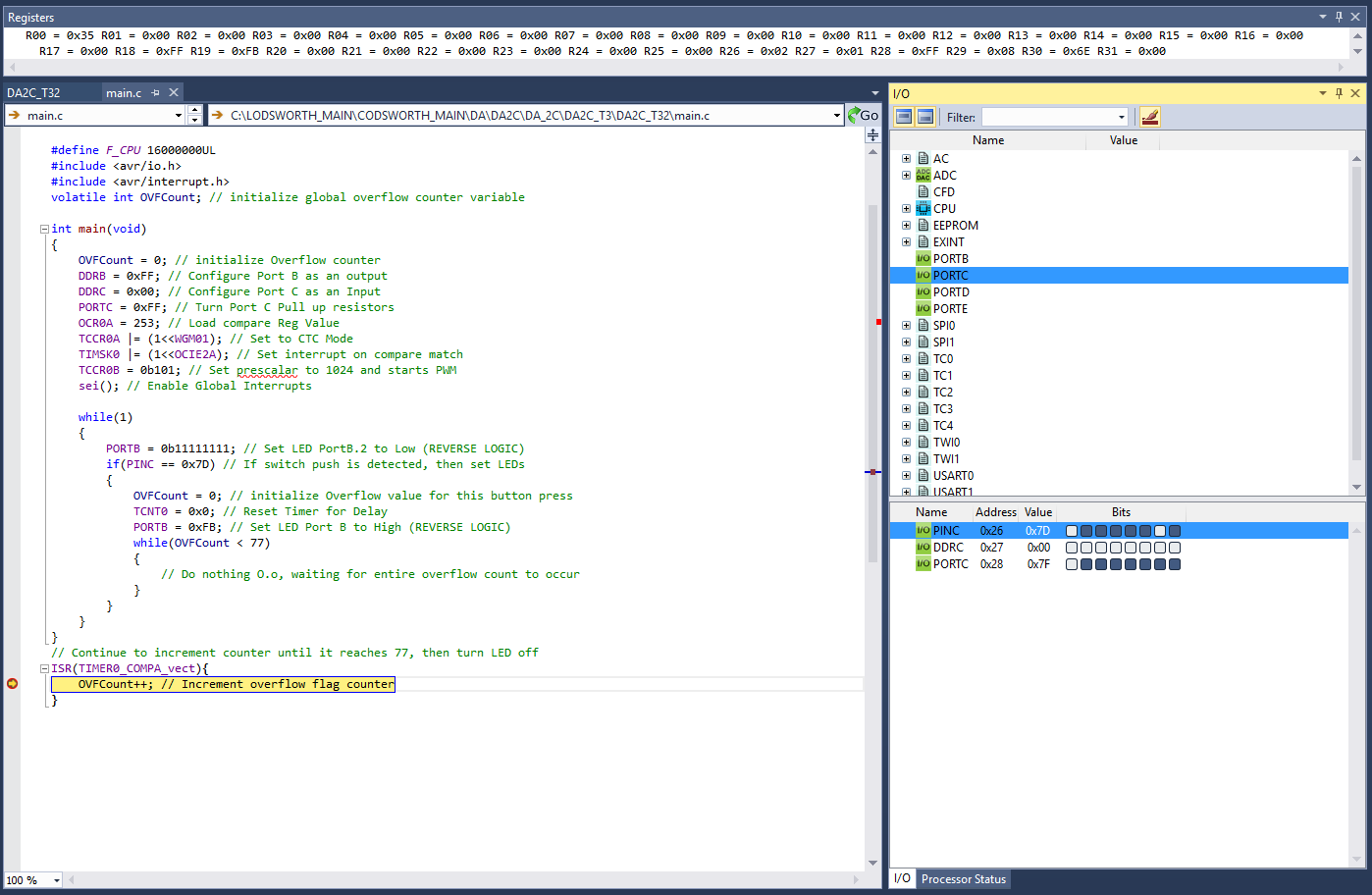
Task 2.2



Task 3.1



Task 3.2



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

Tasks 1.1, 2.1, and 3.1



Tasks 1.2, 2.2, and 3.2



1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/D3mHCsE3zBM>

1. **GITHUB LINK OF THIS DA**

<https://github.com/claytonjhigbee/CODSWORTH_MAIN/tree/master/DA/DA2C>

**Student Academic Misconduct Policy**

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

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

Clayton Higbee