Andrew Wells

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 / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

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

Xplained Mini: ATmega328P micro controller

10 Segment LED bar graph

10 100Ω Resistors

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

.macro MAKESTACK ;Start of Stack Macro

ldi r16, LOW(RAMEND) ;Copy the lower 8 bits of the end of the RAM into R16

out spl, r16 ;Copy the address in r16 to the lower portion of the stack pointer

ldi r16, HIGH(RAMEND) ;Copy the higher 8 bits of the end of hte RAM into R16

out sph, r16 ;Copy the address in r16 to the higher portion of the stack pointer

.endmacro

.cseg

.org 0x0

jmp START

.org 0x100 ;Start of Program

START:

SBI DDRC, 0 ;Enable DDRC.0

SBI DDRC, 4 ;Enable DDRC.4

SBI DDRC, 5 ;Enable DDRC.5

LDI R21, 0 ;Set R21 to 0

LDI R22, 0 ;Set R22 to 0

LDI R16, 0xFF ;Load 0xFF into R16

OUT DDRB, R16 ;Enable all pins of DDRB

MAKESTACK ; Call MAKESTACK macro to create the stack

COUNTING:

call DELAY ;Run DELAY function

CPI R17, 0x01 ;Compare R17 with 1

BRNE COUNTING ;If R17 is 0 go back to COUNTING and run DELAY again

INC R21 ;Increase R21 by 1

INC R22 ;Increase R22 by 2

OUT PORTB, R21 ;Copy value of R21 into PORTB

CHECK1:

CPI R22, 5 ;Compare R22 with 5

BRNE CHECK2 ;If not 5 skip to CHECK2

LDI R18, 0x20;Set R18 to 0x20, 0010 0000

OR R18, R17; OR R18 and R17, Keeps value in PORTC.0

OUT PORTC, R18;Copy R18 into PORTC

rjmp COUNTING ; Return to COUNTING to repeat loop

CHECK2:

CPI R22, 10 ;Compare R22 with 10

BRNE CHECK3 ;If not 10 skip to CHECK3

LDI R18, 0x10;Set R18 to 0x10, 0001 0000

OR R18, R17; OR R18 and R17, Keeps Value in PORTC.0

OUT PORTC, R18; Copy R18 into PORTC

rjmp COUNTING ;Return to COUNTING to repeat loop

CHECK3:

CPI R22, 15 ;Compare R22 with 15

BRNE CHECK4 ;If not 15 skip to CHECK4

LDI R18, 0x30; Set R18 to 0x30, 0011 0000

OR R18, R17; OR R18 and R17, Keeps value in PORTC.0

OUT PORTC, R18;Copy R18 into PORTC

rjmp COUNTING;Return to COUNTING to repeat loop

CHECK4:

CPI R22, 20 ;Compare R2 wtih 20

BRNE COUNTING; If not 20 Return to COUNTING to repeat loop

LDI R22, 0 ; Reset R22 to 0

LDI R18, 0x00;Load 0 into R18

OR R18, R17; OR R18 and R17, Keeps Value in PORTC.0

OUT PORTC, R18; Copy R18 into PORTC

rjmp COUNTING; Return to COUNTING to repeat loop

HOLD:

rjmp HOLD ;Infinite Loop

DELAY:

LDI R16, 0x01 ;Set R16 to 1

LDI R20, 0xF0 ;When using Chip on breadboard: TCNT1 = 0xF862, 63586

STS TCNT1H, R20 ;When using Xplained Mini board: TCNT1 = 0xF0BE, 61630

LDI R20, 0xBE ;When using Chip on breadboard: TCNT1H = 248, TCNT1L = 98

STS TCNT1L, R20 ;When using Xplained Mini board: TCNT1H = 240, TCNT1L = 190

LDI R20, 0x00 ;Set R20 to 0, 00000000

STS TCCR1A, R20 ;Normal Mode

LDI R20, 0x05 ;Set R20 to 5, 00000101

STS TCCR1B, R20 ;1024 prescaler

LOOP1:

IN R20, TIFR1 ;Copy TIFR1 into R20

SBRS R20, TOV1 ;skip next instruction if TOV1 = 1

rjmp LOOP1 ;Jump back to LOOP1

EOR R17, R16 ;XOR R17 and R16 and Save into R17

CPI R17, 0x00 ;Compare R17 with 0

BRNE NOTZERO ;Jump to NOTZERO if R17 is not zero

CBI PORTC, 0 ;Set PORTC.0 to 0

rjmp ZERO ;Jump to ZERO

NOTZERO:

SBI PORTC, 0 ;Set PORTC.0 to 1

ZERO:

LDI R19, 1<<TOV1;Clear TOV1

OUT TIFR1, R19 ;clear OCF1A

ret; Return to Main Program

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

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

#include <avr/io.h>

void DELAY();// Delay function

unsigned char a = 1;// Set a to 1 for XOR with PORTC.0

unsigned char num = 0;// Set num to 0, counting Variable

unsigned char byfive = 0;// Set byfive to 0, counting Variable

int main(void)

{

DDRB = 0xFF;// Activate all PORTB Pins

DDRC = 0x31;// Activate PORTC Pins 0, 4, 5

while (1) //Infinite loop

{

DELAY();// Run Delay function

if ((PORTC % 2) == 1) //

{

num++;// Increase num by 1

byfive++;// Increase byfive by 1

PORTB = num;// Copy the value in num into PORTB

if (byfive == 5)// Compare byfive with 5

{

PORTC |= 1 << PORTC4;// Set PORTC.4 to 1

PORTC &= 0 << PORTC5;// Set PORTC.5 to 0

}else

if (byfive == 10)// Compare byfive with 10

{

PORTC &= 0 << PORTC4;// Set PORTC.4 to 0

PORTC |= 1 << PORTC5;// Set PORTC.5 to 1

}else

if (byfive == 15)// Compare byfive with 15

{

PORTC |= 1 << PORTC4;// Set PORT.4 to 1

PORTC |= 1 << PORTC5;// Set PORT.5 to 1

}else

if (byfive == 20)// Compare byfive with 20

{

byfive = 0;// Reset byfive to 0

PORTC &= 0 << PORTC4;// Set PORT.4 to 0

PORTC &= 0 << PORTC5;// Set PORT.5 to 0

}

}

}

return 0;

}

void DELAY()// Function to cause delay

{

TCNT1 = 0xF0BE;// When using chip on breadboard: TCNT1 to 0xF862, 63586

// When using Xplained Mini board: TCNT1 = 0xF0BE, 61630

TCCR1A = 0x0;//Normal Mode

TCCR1B = 0x05;//1024 Prescaler

while ((TIFR1 & (1<<TOV1)) == 0)// Stay in loop while TOV1 = 0

{

}

TIFR1 |= 1 << 0;// Clear TOV1

PORTC ^= a << PORTC0;// XOR PORTC.0 with a to toggle on and off

return; // Return to main program

}

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

.macro MAKESTACK ;Start of Stack Macro

ldi r16, LOW(RAMEND) ;Copy the lower 8 bits of the end of the RAM into R16

out spl, r16 ;Copy the address in r16 to the lower portion of the stack pointer

ldi r16, HIGH(RAMEND) ;Copy the higher 8 bits of the end of hte RAM into R16

out sph, r16 ;Copy the address in r16 to the higher portion of the stack pointer

.endmacro

.cseg

.org 0x0

jmp START ;Jump to program start

.org 0x16

jmp TIM1\_COMPA ;Jump to Interrupt

.org 0x100 ;Start of Program

START:

SBI DDRC, 0 ;Enable DDRC.0

SBI DDRC, 4 ;Enable DDRC.4

SBI DDRC, 5 ;Enable DDRC.5

LDI R21, 0 ;Set R21 to 0

LDI R22, 0 ;Set R22 to 0

LDI R17, 0 ;Set R17 to 0

LDI R16, 0xFF ;Load 0xFF into R16

OUT DDRB, R16 ;Enable all pins of DDRB

MAKESTACK ; Call MAKESTACK macro to create the stack

LDI R20, 0x02 ;Set R20 to 2, 0000 0010

STS TIMSK1, R20 ;Enable Timer1 compare interrupt

SEI ;Global Interrupt Enable

LDI R20, 0x0F ;When using Chip on breadboard: TCNT1 = 0x079E, 1950

STS OCR1AH, R20 ;When using Xplained Mini board: OCR1A = 0x0F42, 3906

LDI R20, 0x42 ;When using Chip on breadboard: TCNT1H = 7, TCNT1L = 158

STS OCR1AL, R20 ;When using Xplained Mini board: TCNT1H = 15, TCNT1L = 66

LDI R20, 0x00 ;Set R20 to 0, 00000000

STS TCCR1A, R20 ;CTC Mode

LDI R20, 0x0D ;Set R20 to 13, 00001101

STS TCCR1B, R20 ;1024 prescaler

HOLD:

rjmp HOLD ;Infinite Loop

TIM1\_COMPA: ;OCF1A clears automatically when interrupt is run

LDI R16, 0x01 ;Set R16 to 1

EOR R17, R16 ;XOR R17 and R16 and Save into R17

CPI R17, 0x00 ;Compare R17 with 0

BRNE NOTZERO ;Jump to NOTZERO if R17 is not zero

CBI PORTC, 0 ;Set PORTC.0 to 0

rjmp COUNTING ;Jump to COUNTING

NOTZERO:

SBI PORTC, 0 ;Set PORTC.0 to 1

INC R21 ;Increase R21 by 1

INC R22 ;Increase R22 by 2

OUT PORTB, R21 ;Copy value of R21 into PORTB

CHECK1:

CPI R22, 5 ;Compare R22 with 5

BRNE CHECK2 ;If not 5 skip to CHECK2

LDI R18, 0x20;Set R18 to 0x20, 0010 0000

OR R18, R17; OR R18 and R17, Keeps value in PORTC.0

OUT PORTC, R18;Copy R18 into PORTC

jmp COUNTING ;Jump to COUNTING, skip rest of checks

CHECK2:

CPI R22, 10 ;Compare R22 with 10

BRNE CHECK3 ;If not 10 skip to CHECK3

LDI R18, 0x10;Set R18 to 0x10, 0001 0000

OR R18, R17; OR R18 and R17, Keeps Value in PORTC.0

OUT PORTC, R18; Copy R18 into PORTC

jmp COUNTING ;Jump to COUNTING, skip rest of checks

CHECK3:

CPI R22, 15 ;Compare R22 with 15

BRNE CHECK4 ;If not 15 skip to CHECK4

LDI R18, 0x30; Set R18 to 0x30, 0011 0000

OR R18, R17; OR R18 and R17, Keeps value in PORTC.0

OUT PORTC, R18;Copy R18 into PORTC

jmp COUNTING ;Jump to COUNTING, skip rest of checks

CHECK4:

CPI R22, 20 ;Compare R2 wtih 20

BRNE COUNTING; If not 20 Skip to COUNTING to exit interrupt

LDI R22, 0 ; Reset R22 to 0

LDI R18, 0x00;Load 0 into R18

OR R18, R17; OR R18 and R17, Keeps Value in PORTC.0

OUT PORTC, R18; Copy R18 into PORTC

COUNTING:

RETI ; return to main program

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

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

#include <avr/io.h>

#include <avr/interrupt.h>

unsigned char a = 1;// Set a to 1 for XOR with PORTC.0

unsigned char num = 0;// Set num to 0, counting Variable

unsigned char byfive = 0;// Set byfive to 0, counting Variable

int main(void)

{

DDRB = 0xFF;// Activate all PORTB Pins

DDRC = 0x31;// Activate PORTC Pins 0, 4, 5

OCR1A = 0x0F42;// When using Chip on breadboard: TCNT1 = 0x079E, 1950

//When using Xplained Mini board: OCR1A = 0x0F42, 3906

TCCR1A = 0x0;//CTC Mode

TCCR1B = 0x0D;//1024 Prescaler

TIMSK1 = (1<<OCIE1A); //Enable Timer1 compare interrupt

sei (); // Enable global interrupt

while (1) //Infinite loop

{

}

return 0;

}

ISR (TIMER1\_COMPA\_vect)

{

PORTC ^= a << PORTC0;// XOR PORTC.0 with a to toggle on and off

if ((PORTC % 2) == 1) //

{

num++;// Increase num by 1

byfive++;// Increase byfive by 1

PORTB = num;// Copy the value in num into PORTB

if (byfive == 5)// Compare byfive with 5

{

PORTC |= 1 << PORTC4;// Set PORTC.4 to 1

PORTC &= 0 << PORTC5;// Set PORTC.5 to 0

}else

if (byfive == 10)// Compare byfive with 10

{

PORTC &= 0 << PORTC4;// Set PORTC.4 to 0

PORTC |= 1 << PORTC5;// Set PORTC.5 to 1

}else

if (byfive == 15)// Compare byfive with 15

{

PORTC |= 1 << PORTC4;// Set PORT.4 to 1

PORTC |= 1 << PORTC5;// Set PORT.5 to 1

}else

if (byfive == 20)// Compare byfive with 20

{

byfive = 0;// Reset byfive to 0

PORTC &= 0 << PORTC4;// Set PORT.4 to 0

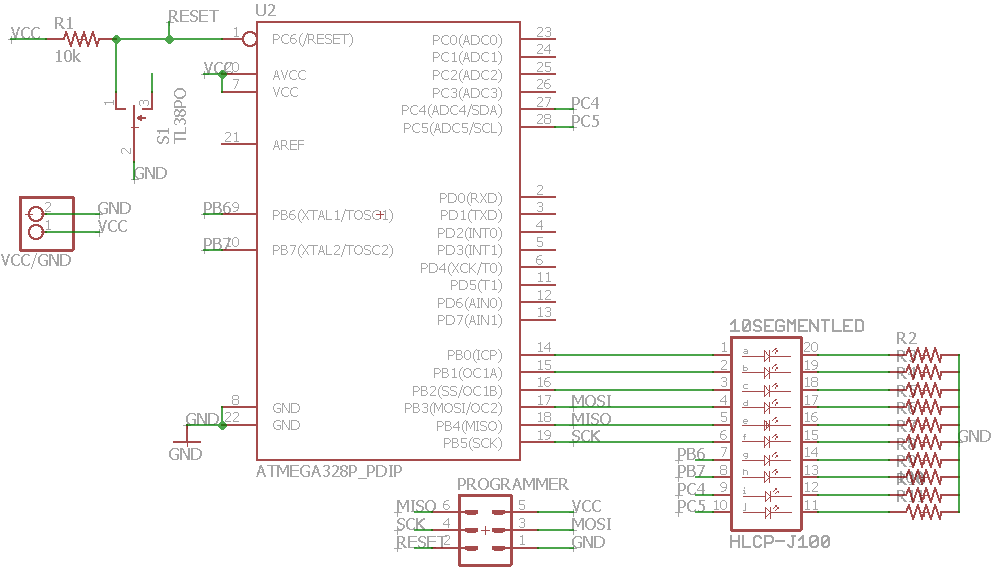
PORTC &= 0 << PORTC5;// Set PORT.5 to 0

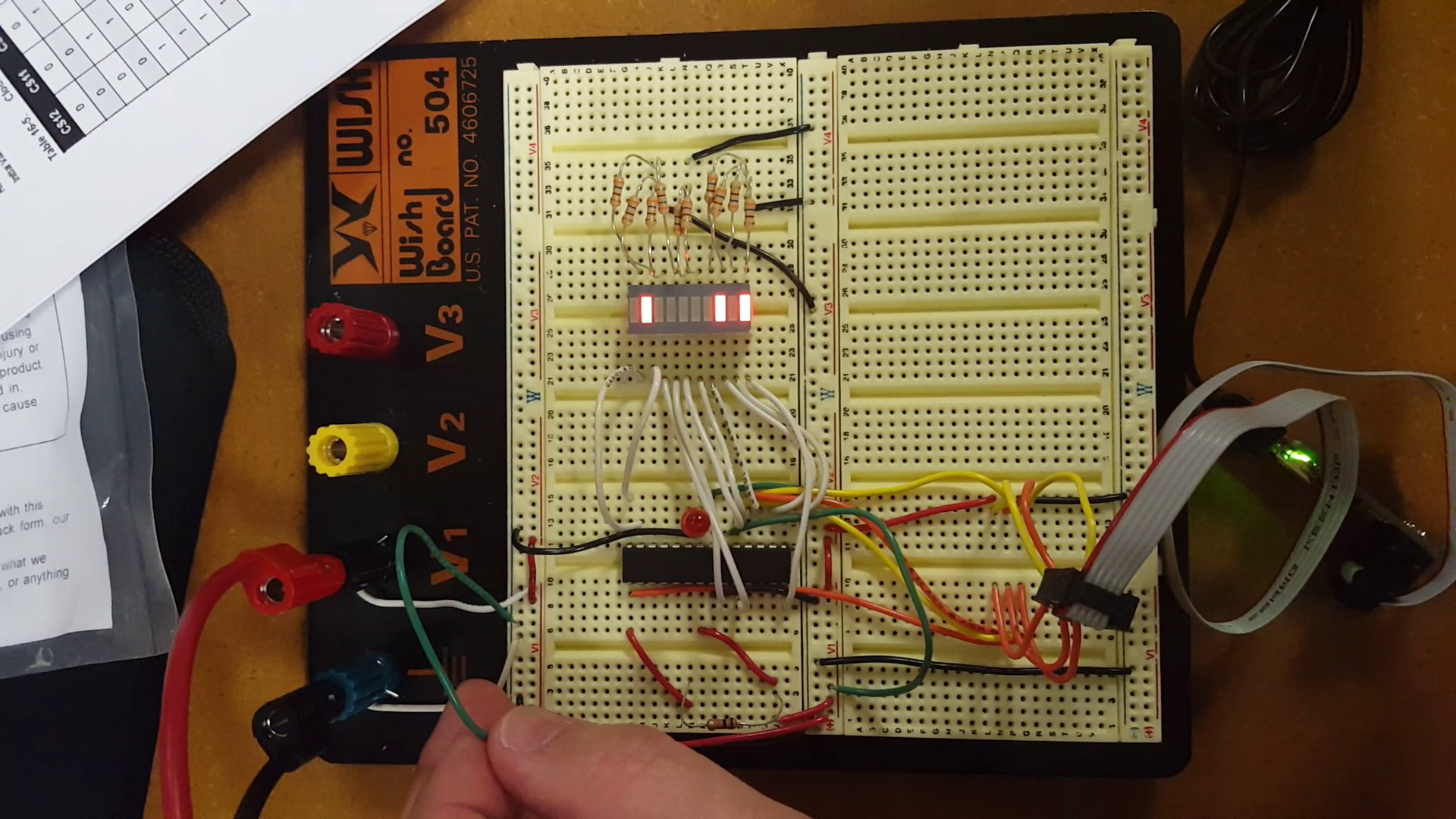
}

}

}

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



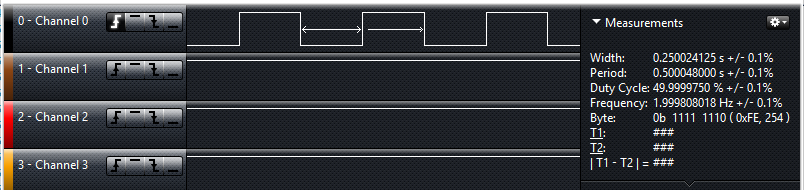


|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

TASK 1:

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

sec period.



TASK 2:

Implement a 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.

TASK3:

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.

TASK4:

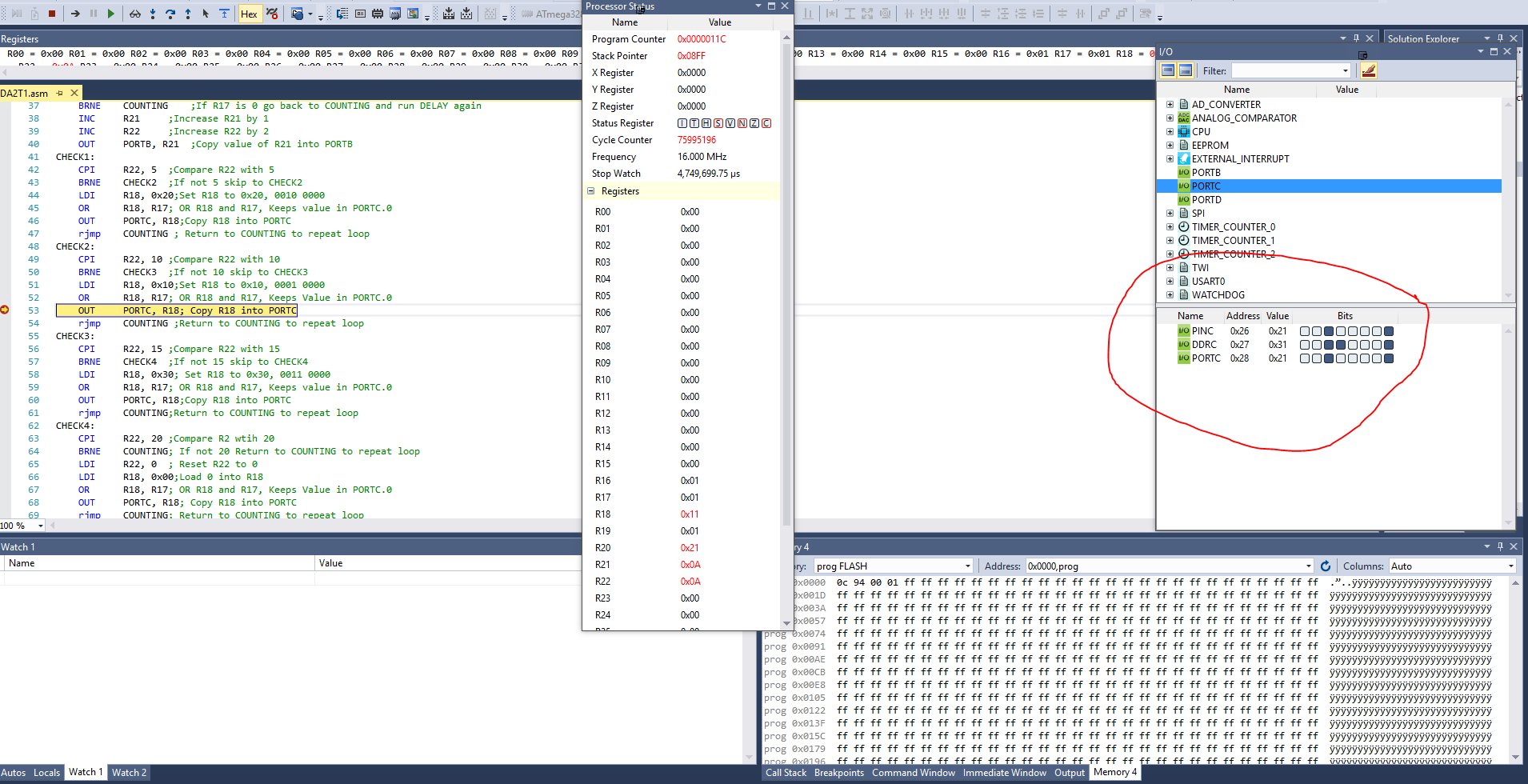
Modify the above code to use interrupts to update the status/toggle of the 9th and

10th LED.

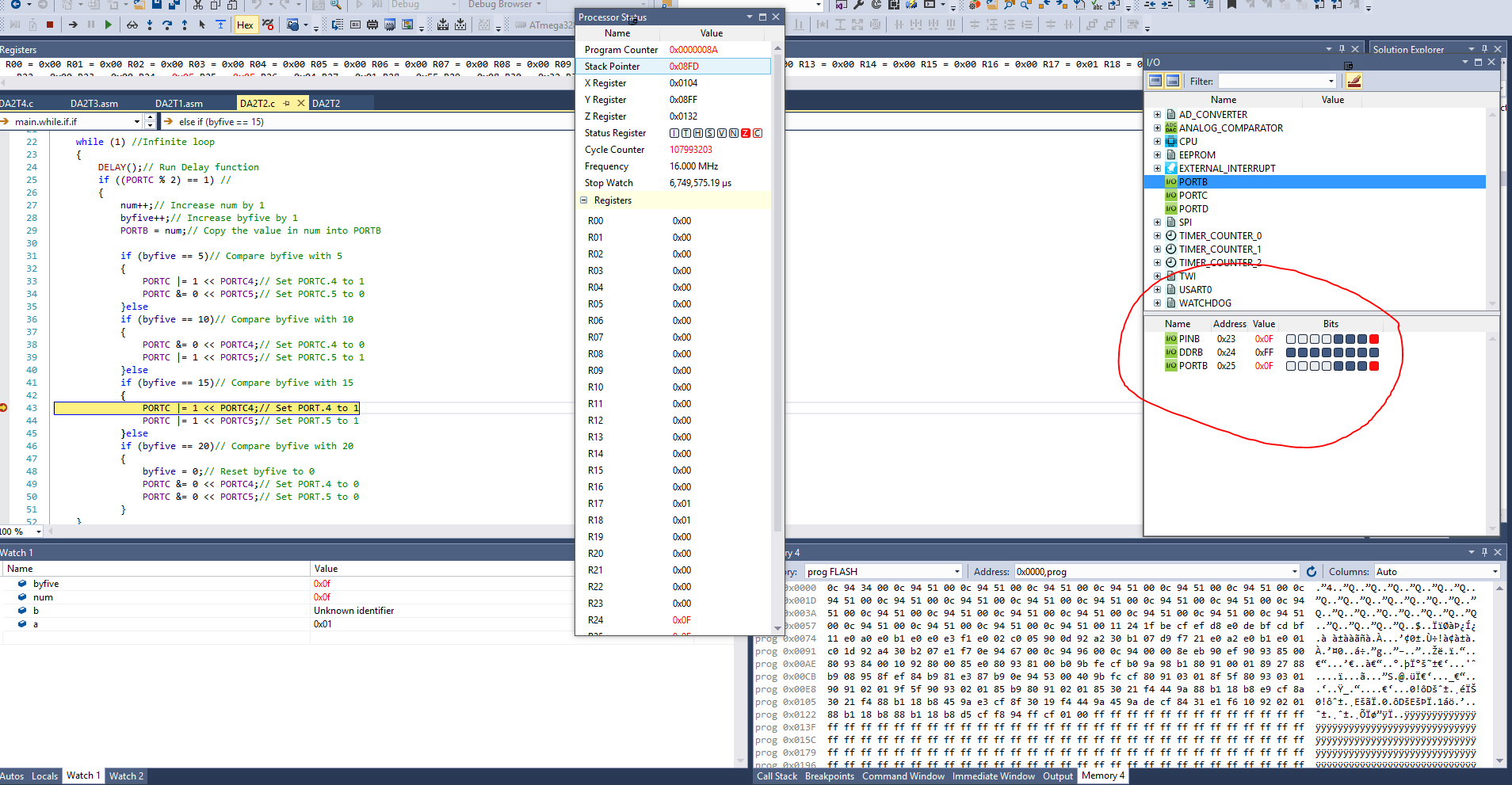
See video for Tasks 2-4.

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

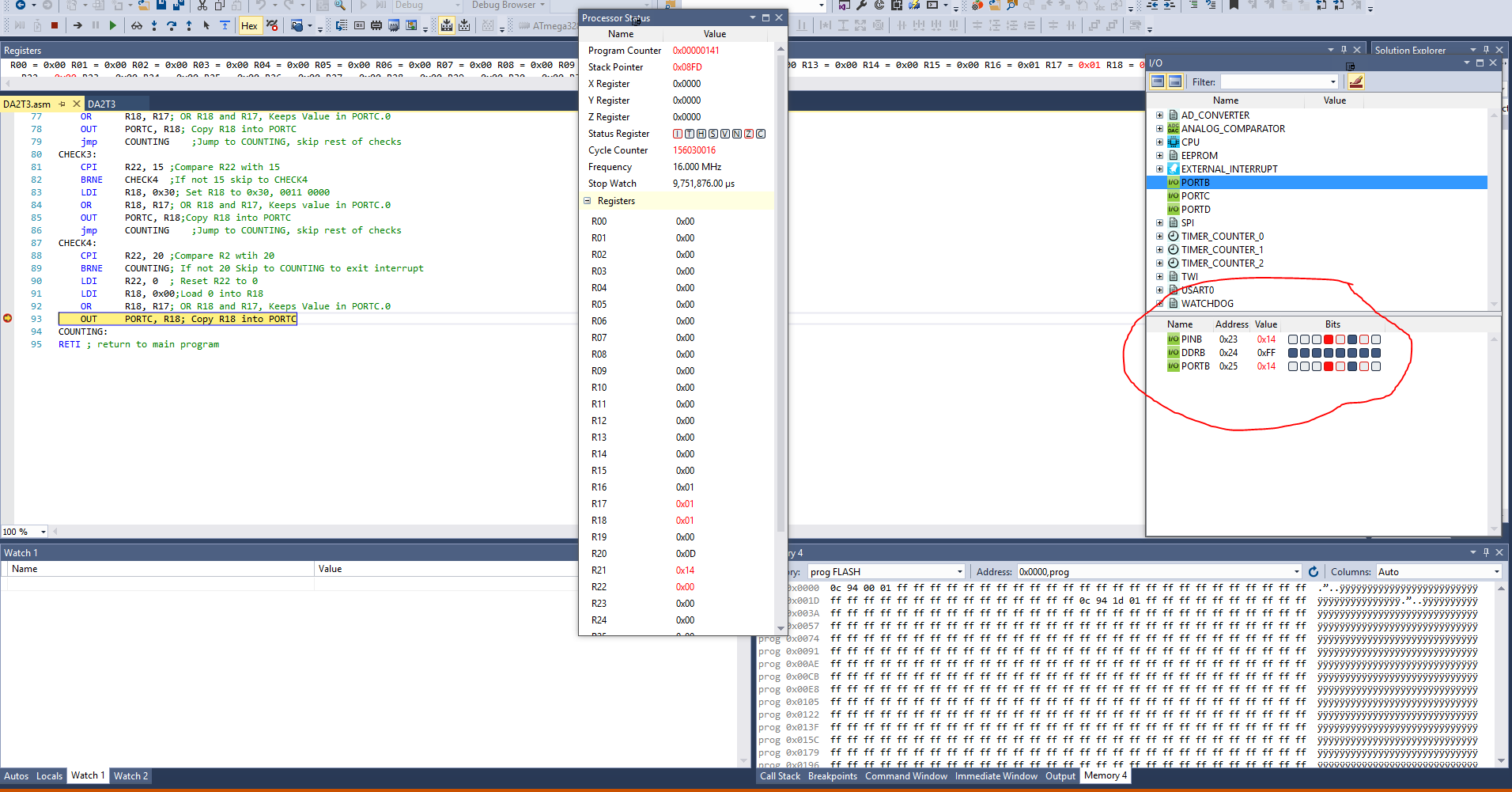
TASK 1: Assembly without interrupt



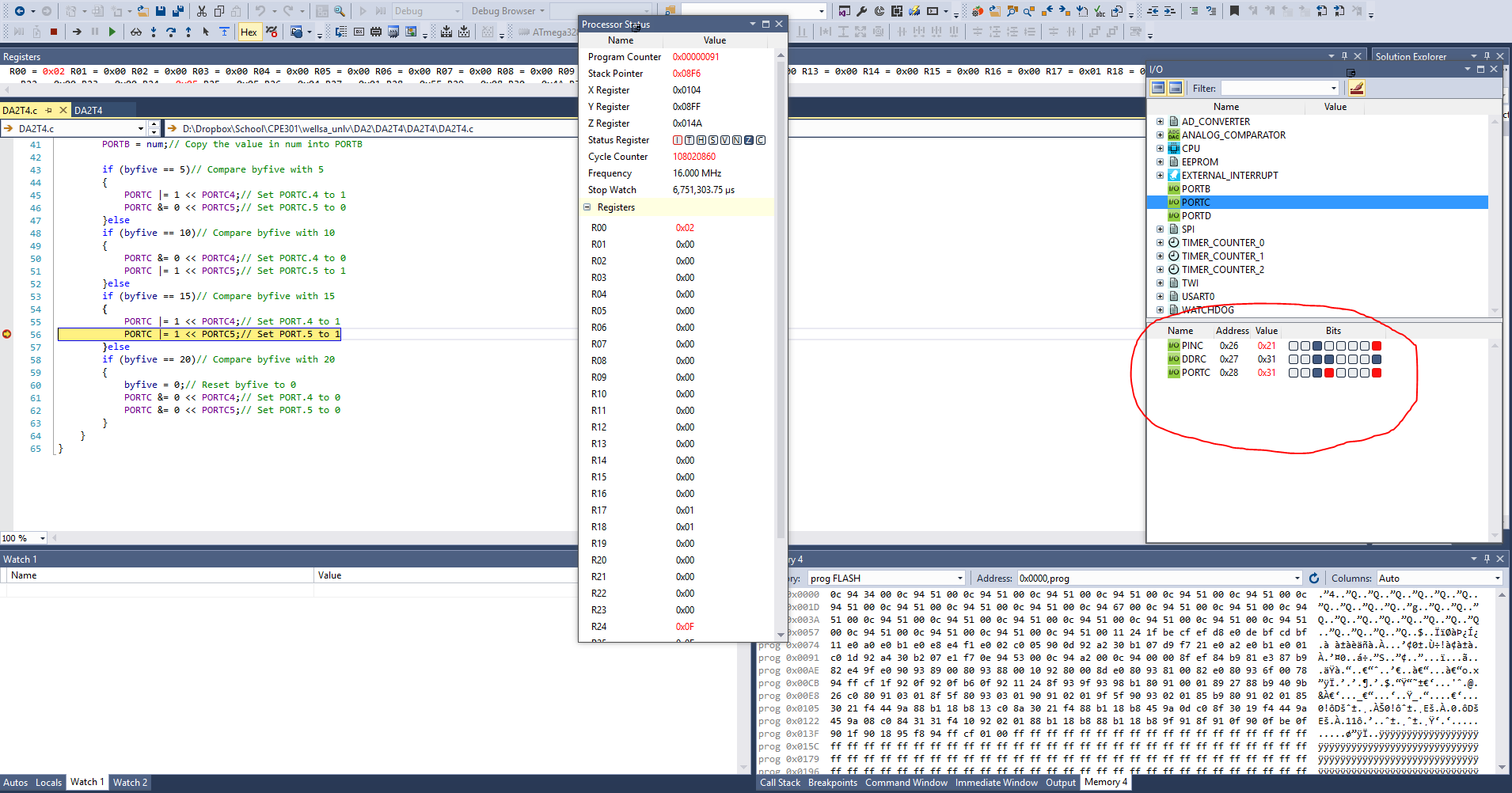
TASK2: C without interrupt



TASK3: Assembly with interrupt



TASK4: C with interrupt



|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| https://youtu.be/42GUsY9HwiE | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| https://github.com/Wellsa15/wellsa\_unlv | | | |

**Student Academic Misconduct Policy**

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

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

Andrew Wells