RALPH MAGO

CPE301 – SPRING 2016

Design Assignment 2

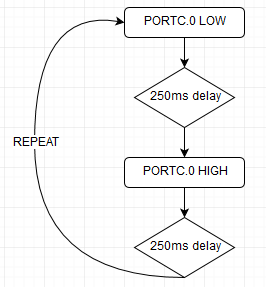
**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

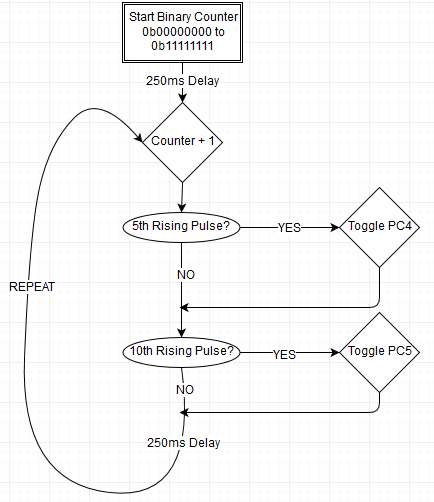
|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | FLOWCHART OF THE CODE | Y |  |
| 1. | COMPONENTS LIST | Y |  |
| 2. | INITIAL CODE OF TASK 1/A | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 1/B | Y |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 1/C | Y |  |
| 5. | INITIAL CODE OF TASK 2/A | Y |  |
| 6. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 7. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/C | Y |  |
| 8. | SCHEMATICS | Y |  |
| 9. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 10. | SCREENSHOT OF EACH DEMO | Y |  |
| 11. | VIDEO LINKS OF EACH DEMO | Y |  |
| 12. | GITHUB LINK OF THE DA | Y |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | FLOWCHART OF THE CODE |  |  |

Flowchart of blinking PC0 with 0.5s period:



Flowchart of Binary Counter with PC4 and PC5 rising edge toggles with 0.5s period:



|  |  |  |  |
| --- | --- | --- | --- |
| 1. | COMPONENTS LIST |  |  |

|  |  |
| --- | --- |
| ATMega328P-PU | 1 |
| 330 Ω Resistor | 11 |
| 10-LED Bar | 1 |
| Green LED | 1 |
| Breadboard | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INITIAL CODE OF TASK 1/A |  |  |

;

; DA2T1.asm

;

; Created: 3/8/2016 3:11:13 PM

; Author : r

;

; REQUIREMENTS : Generate a 50% DC with 0.5sec period on PORTC.0

; FOR 8MHZ CLOCK

SBI DDRC, 0 ; set PC0 output

CBI DDRC, 0 ; clear PC0 bit

PC0\_Blink:

CALL delay\_250ms ; call 1/4 second delay procedure (current counter≈250ms)

LDI R17, 1 ; R17 = 1

OUT PORTC, R22 ; set PC0 high

CALL delay\_250ms ; call 1/4 second delay procedure (current counter≈500ms)

LDI R17, 0 ; R17 = 0

OUT PORTC, R22 ; set PC0 low

JMP PC0\_Blink ; repeat procedure

; DELAY CALCULATION:

; 4 \* 55 \* 90 \* 100 = 1,980,000

; We are using 8MHz frequency, so 1,980,000 / 8,000,000 = 0.2475 ≈ 250ms

;

delay\_250ms:

PUSH R18 ; push R18

PUSH R19 ; push R19

PUSH R20 ; push R20 (registers for creating looping branch cycles for delay)

LDI R18, 55 ; R18 = 55

delay1:

LDI R19, 90 ; R19 = 90

delay2:

LDI R20, 100 ; R20 = 100

delay3:

NOP ; NOP adds extra cycle

DEC R20

BRNE delay3 ; repeats previous loops to add to clock cycle

DEC R19

BRNE delay2

DEC R18

BRNE delay1

POP R20 ; pop back registers used for loops

POP R19

POP R18

RET ; finished delay procedure, go back to main

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 1/B |  |  |

;

; DA2T2.asm

;

; Created: 3/8/2016 4:32:04 PM

; Author : User

;

; REQUIREMENTS: Use PORTB to display an 8-bit binary counter. PC4 will toggle every 5th pulse

; and PC5 will toggle every 10th pulse. Non-interrupt method.

SBI DDRB, 0 ; output PB0

SBI DDRB, 1 ; output PB1

SBI DDRB, 2 ; output PB2

SBI DDRB, 3 ; output PB3

SBI DDRB, 4 ; output PB4

SBI DDRB, 5 ; output PB5

SBI DDRB, 6 ; output PB6

SBI DDRB, 7 ; output PB7

SBI DDRC, 0 ; output PC0 (pulse)

SBI DDRC, 4 ; output PC4 (toggle 5th pulse)

SBI DDRC, 5 ; output PC6 (toggle 10th pulse)

LDI R22, 0x00 ; R22 is counter used for 8-bit counter

LDI R23, 0x00 ; R23 is counter for 5th or 10th rising pulse check

binaryCounter:

CALL delay\_250ms ; 250ms delay procedure

INC R22 ; increment 8-bit counter

OUT PORTB, R22 ; output 8-bit counter on PORTB

INC R23 ; increment count for detecting 5th or 10th rising pulse

; PULSE CHECK PROCEDURE

CPI R23, 5 ; first 5th pulse

BREQ rising\_5 ; branch if equal to 5

CPI R23, 10 ; next 5th (10th pulse)

BREQ rising\_10 ; branch if equal to 10

CPI R23, 15 ; next 5th (15th pulse)

BREQ rising\_15 ; branch if equal to 15

CPI R23, 20 ; next 5th (20th pulse)

BREQ rising\_20 ; branch if equal to 15

JMP next\_250ms ; if does not match any case, toggle nothing

rising\_5:

LDI R24, 0x10

OUT PORTC, R24 ; 5th pulse where PC4 = 1 and PC5 = 0

JMP next\_250ms ; finish pulse output, complete the delay

rising\_10:

LDI R24, 0x20

OUT PORTC, R24 ; 10th pulse where PC4 = 0 and PC5 = 1

JMP next\_250ms ; finish pulse output, complete the delay

rising\_15:

LDI R24, 0x30

OUT PORTC, R24 ; 5th pulse where PC4 = 1 and PC5 = 1

JMP next\_250ms ; finish pulse output, complete the delay

rising\_20:

LDI R24, 0x00

OUT PORTC, R24 ; second 10th pulse where PC4 = 0 and PC5 = 0

LDI R23, 0x00 ; set pulse check counter back to zero to redo cases

JMP next\_250ms ; finish pulse output, complete the delay

next\_250ms:

CALL delay\_250ms ; delay another 250ms to complete 0.5sec period

JMP binaryCounter ; count again, don't care about overflow

; DELAY CALCULATION:

; 4 \* 55 \* 90 \* 100 = 1,980,000

; We are using 8MHz frequency, so 1,980,000 / 8,000,000 = 0.2475 ≈ 250ms

;

delay\_250ms:

PUSH R18 ; push R18

PUSH R19 ; push R19

PUSH R20 ; push R20 (registers for creating looping branch cycles for delay)

LDI R18, 55 ; R18 = 55

delay1:

LDI R19, 90 ; R19 = 90

delay2:

LDI R20, 100 ; R20 = 100

delay3:

NOP ; NOP adds extra cycle

DEC R20

BRNE delay3 ; repeats previous loops to add to clock cycle

DEC R19

BRNE delay2

DEC R18

BRNE delay1

POP R20 ; pop back registers used for loops

POP R19

POP R18

RET ; finished delay procedure, go back to main

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

;

; DA2T3.asm

;

; Created: 3/8/2016 4:59:29 PM

; Author : r

;

.ORG 0x00 ; PC = 0x00

RJMP INIT ; Initialize Stack at PC = 0

.ORG 0x16 ; TIM1\_COMPA in interrupt table is 0x16

RJMP TIM1\_COMPA ; Timer 1 Compare A Handler

INIT: ; Macro that initializes stack pointer

.MACRO INITSTACK

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

.ENDMACRO

INITSTACK

RJMP main

main:

SBI DDRB, 0 ; output PB0

SBI DDRB, 1 ; output PB1

SBI DDRB, 2 ; output PB2

SBI DDRB, 3 ; output PB3

SBI DDRB, 4 ; output PB4

SBI DDRB, 5 ; output PB5

SBI DDRB, 6 ; output PB6

SBI DDRB, 7 ; output PB7

SBI DDRC, 0 ; output PC0 (pulse)

SBI DDRC, 4 ; output PC4 (toggle 5th pulse)

SBI DDRC, 5 ; output PC6 (toggle 10th pulse)

LDI R20, 0x00 ; R20 is counter used for 8-bit counter

LDI R23, 0x00 ; R23 is counter for 5th or 10th rising pulse check

LDI R24, 0x0F ; R24 used as temporary value register. load 0x0F into R24

STS OCR1AH, R24 ; store 0x0F into the high register of OCR1A

LDI R24, 0x42 ; load 0x42 into R24

STS OCR1AL, R24 ; store 0x42 into the low register of OCR1A

LDI R24, 0x00 ; load 0x00 into R24

STS TCCR1A, R24 ; set CTC and have OC1A be disconnected to not change bits on bargraph

LDI R24, 0x0D ; load 0x0D into R24

STS TCCR1B, R24 ; set prescaler to 1024 and WGM12 bit high

LDI R24, 0x02 ; load 0x02 into R24

STS TIMSK1, R24 ; set OCIEA1 high to enable interrupts for OCF1A

SEI ; enable global interrupts

loop:

RJMP loop ; relative jump, looping forever while interrupt occurs at 0.5 sec intervals

; INTERRUPT PROCEDURE

TIM1\_COMPA: ; Timer 1 Compare A Handler

INC R20 ; binary counter is incremented

OUT PORTB, R20 ; update value of counter on LED bargraph

INC R23 ; increment register checking for 5th/10th rising edge

; DETECT RISING PULSES

CPI R23, 5 ; compare immediate, 5th pulse

BREQ rising\_5 ; if it is the 5th pulse, branch

CPI R23, 10 ; compare immediate, 10th pulse

BREQ rising\_10 ; if it is the 10th pulse, branch

CPI R23, 15 ; compare immediate, 15th pulse

BREQ rising\_15 ; if it is the 15th pulse, branch

CPI R23, 20 ; compare immediate, 20th pulse

BREQ rising\_20 ; if it is the 20th pulse, branch

JMP negEdge ; jump to negative edge if not a 5th or 10th pulse

rising\_5:

LDI R16, 0x10

OUT PORTC, R16 ; PC4 = 0 and PC5 = 1, Toggle PC4

JMP negEdge ; toggle done, jump to negative edge

rising\_10:

LDI R16, 0x20

OUT PORTC, R16 ; PC4 = 1 and PC5 = 0, Toggle PC5

JMP negEdge ; toggle done, jump to negative edge

rising\_15:

LDI R16, 0x30

OUT PORTC, R16 ; PC4 = 1 and PC5 = 1, Toggle PC4 and PC5

JMP negEdge ; toggle done, jump to negative edge

rising\_20:

LDI R16, 0x00

OUT PORTC, R16 ; second 10th pulse where PC4 = 0 and PC5 = 0

LDI R23, 0x00 ; set counter back to zero to redo toggle check

JMP negEdge ; toggle done, jump to negative edge

negEdge:

LDI R24, 0x02 ; load 0x02 into R24

STS TIFR1, R24 ; clear OCF1A flag

RETI ; return and enable interrupts

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | INITIAL CODE OF TASK 2/A |  |  |

/\*

\* DA2T4.c

\*

\* Created: 3/8/2016 5:24:22 PM

\* Author : r

\*/

// REQUIREMENTS: Blink PC0 with 50% DC and 0.5sec period.

#define *F\_CPU* 8000000UL // 8MHz Frequency for Simulator

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRC |= (1 << PC0); // PC0 is output

while(1) // loop forever blinking PC0

{

*\_delay\_ms*(250); // 250ms delay

PORTC |= 0x01; // set PC1 high

*\_delay\_ms*(250); // 250ms delay (completes 0.5sec period)

PORTC = 0; // set PC0 low

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |

/\*

/\*

\* DA2T5.c

\*

\* Created: 3/8/2016 11:55:01 PM

\* Author : r

\*/

// REQUIREMENTS : Output 8-bit counter on PB0-PB7. 0.5sec delay. PC4 toggled every 5th

// rising edge and PC5 is toggled every 10th rising edge.

#define *F\_CPU* 8000000UL // 8MHz Frequency for Simulator

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

int binaryCounter = 0; // set counter for LED bargraph starting at 0

int risingCounter = 0; // set counter for 5th/10th rising edge starting at 0

DDRB = 0xFF; // PB0-PB7 set to outputs

DDRC = 0x30; // PC4 and PC5 set to outputs

while(1) //count sequence loop forever

{

*\_delay\_ms*(250); // 250ms delay

binaryCounter++; // increment counter after delay

PORTB = binaryCounter; // output counter to LED

risingCounter++; //used to detect every 5th and 10th rising edge

if(risingCounter%5 == 0) //PC4 toggles every 5 rising pulses

PORTC ^= (1<<PC4);

if(risingCounter%10 == 0) //PC5 toggles every 10 rising pulses

PORTC ^= (1<<PC5);

*\_delay\_ms*(250); // 250ms delay (completes 0.5sec period)

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 7. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/C |  |  |

/\*

\* DA2T6.c

\*

\* Created: 3/9/2016 12:13:03 AM

\* Author : r

\*/

// REQUIREMENTS : Output 8-bit counter on PB0-PB7. PC0 generates 50% DC and 0.5sec period.

// PC4 toggles every 5th rising pulse, PC5 toggles every 10th rising pulse.

// Toggles are handled by interrupts

#define *F\_CPU* 8000000UL //set clock frequency to 16MHz for compiler

#include <avr/io.h>

#include <avr/interrupt.h> //use interrupt library

void delay\_msec(int ms); //delay function procedure

//calculation using clock freq. (here we are using 8MHz)

int main(void)

{

DDRC |= 0x31; // set PC4 an PC5 as outputs on PORTC

DDRB |= 0xFF; //set every pin on PORTB as an output

//////////////

//TIMER 0

//////////////

DDRD &= ~(1<<PD4); //set PORTD.4 as input. pin PD4 uses TIMER0 and PC0 connects to this pin

//for the binary counter to properly count on the LEDs.

TCCR0A = 0; //timer 0 in normal mode

TCCR0B = 0x07; //external clock on TIMER0

TCNT0 = 0; //initialize binary count sequence

//////////////

//INTERRUPTS

//////////////

PCICR |= (1<<PCIE1); //turn on pin change interrupts for PORTC

PCIFR |= (1<<PCIF1); //turn on pin change interrupt flag for PORTC

PCMSK1 |= (1<<PCINT8); //turn on PC0 (PCINT8) for interrupt usage (PCMSK1 is for PORTC)

sei(); //set interrupts

//////////////

//BINARY COUNTER

//////////////

while(1) //generate waveform of 50% DC with 0.5 sec period

{

PORTC &= ~(1<<PC0); //PORTC.0 low

delay\_msec(250); //delay for 250 ms

PORTC |= (1<<PC0); //PORTC.0 high

PORTB = TCNT0+1; //binary count sequence incremented by 1 each loop

delay\_msec(250); //delay for 250 ms

}

}

//////////////

//DELAY PROCEDURE

//////////////

//////////////

void delay\_msec(int ms)

{

double timerCalc = 65536 - (((double)ms)/1000)\*((double)*F\_CPU*/64);

//set timer calculation = 2^16 - delay in sec / (clock period/ prescaler 64)

TCNT1H = ((int)timerCalc) >> 0x08; //move upper byte of TCNT

TCNT1L = ((int)timerCalc) & 0x00FF; //move lower byte of TCNT

TCCR1A = 0; //Timer 1 Normal Mode

TCCR1B = 0x03; //Prescaler = 64 (we can easily change prescaler here for emulation)

while(!(TIFR1 & (1<<TOV1))) { //loop until overflow

}

TCCR1B = 0; // stop timer 1

TIFR1 = 1; // clear overflow flag

return;

}

//////////////

//INTERRUPT SERVICE ROUTINE

//////////////

ISR (PCINT1\_vect) //PC0 is in the PCINT1\_vect

{

if(!(PORTC & (1<<PC0)) == 0) //if PC0 is changed from low to high

{

static int risingCounter = 0; //used to detect every 5th and 10th rising edge

risingCounter++;

if(risingCounter%5 == 0) //PC4 toggles every 5 rising pulses

PORTC ^= (1<<PC4);

if(risingCounter%10 == 0) //PC5 toggles every 10 rising pulses

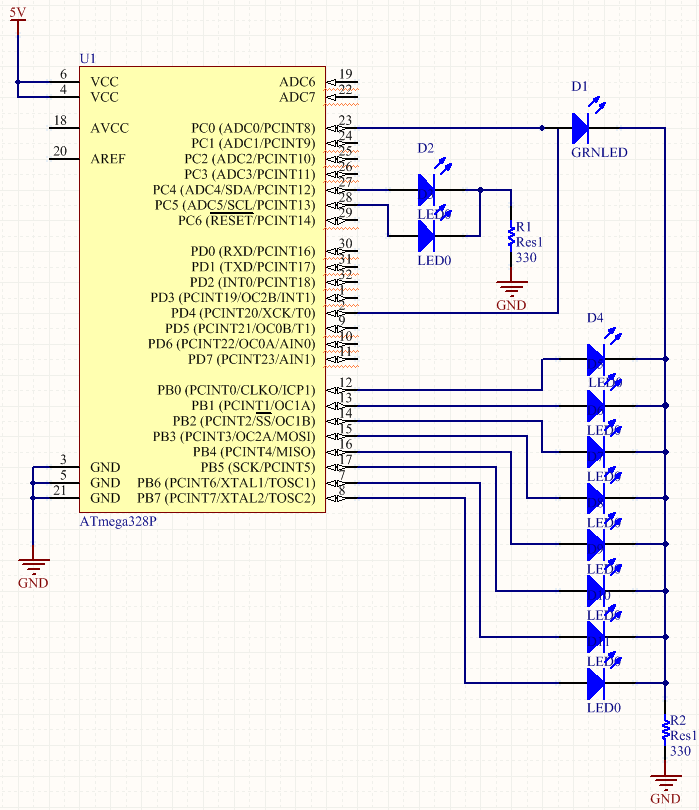
PORTC ^= (1<<PC5);

}

return;

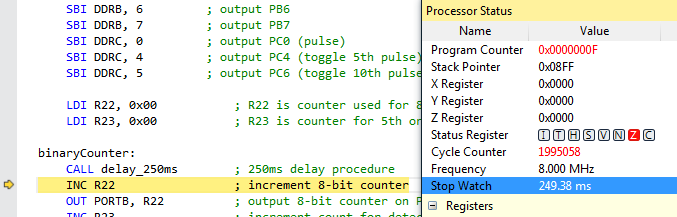
}

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCHEMATIC |  |  |

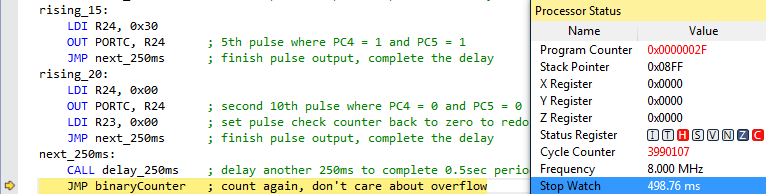


|  |  |  |  |
| --- | --- | --- | --- |
| 9. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

**Assembly Version – Simulation**

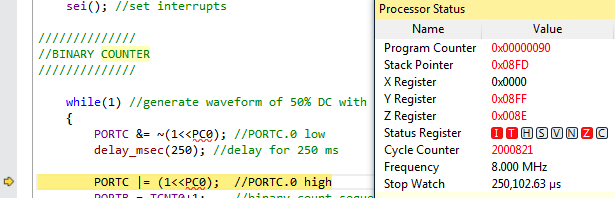


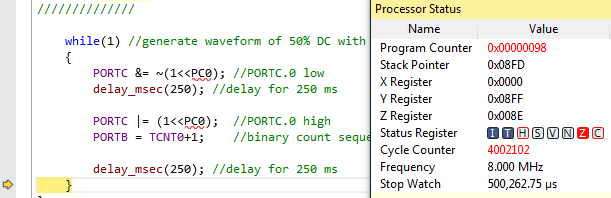
Half Period -> 249.38ms -> Cycle Counter = 1995058



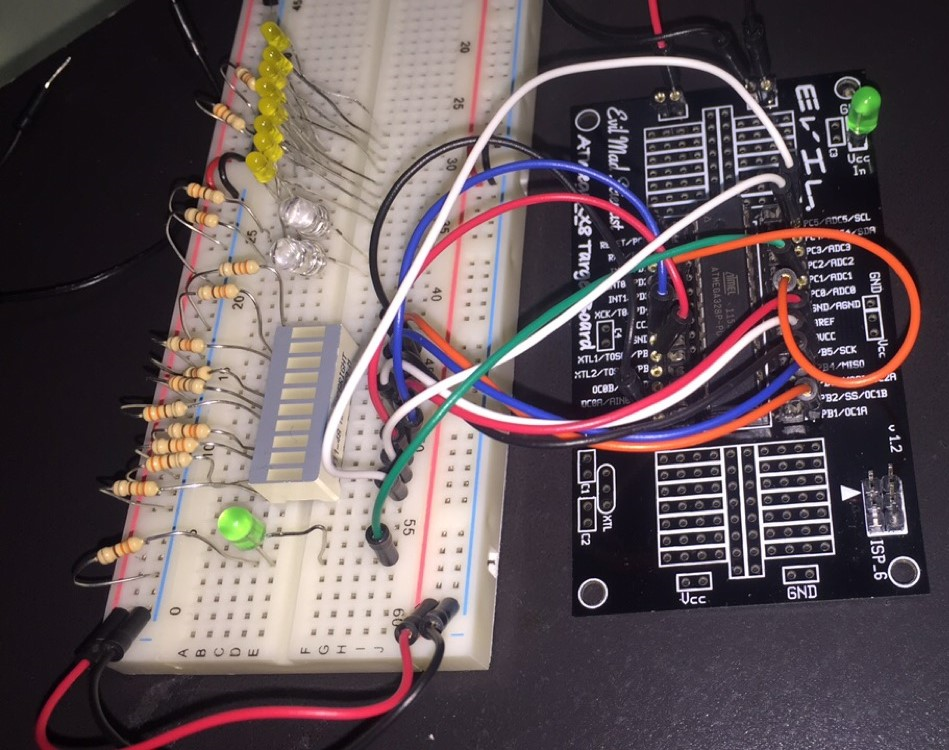
Full Period -> 498.76ms ≈ 0.5 sec period -> Cycle Counter = 3990107 (≈ 50% Duty Cycle)

**C Version – Simulation**

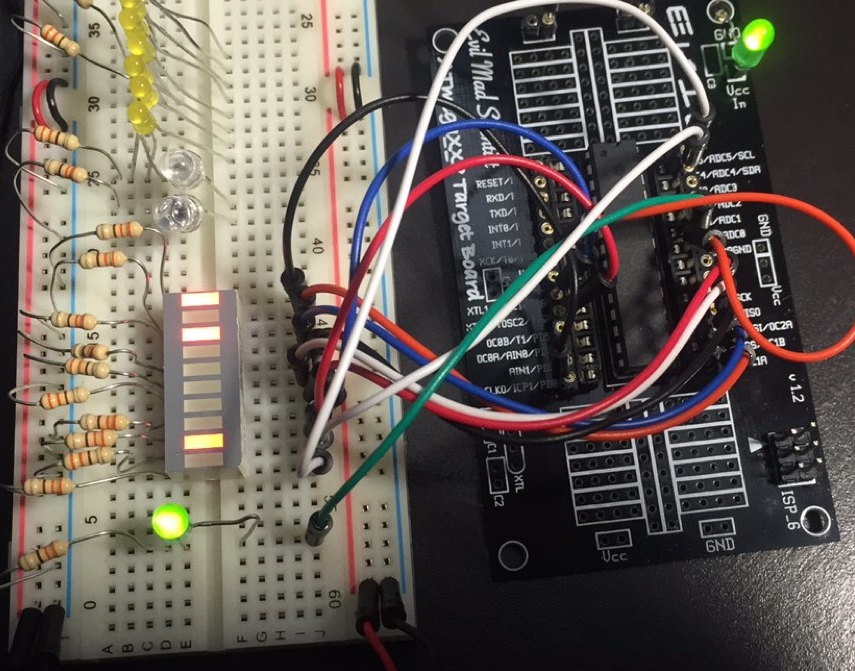
 Half Period -> 250.10ms -> Cycle Counter = 2000821

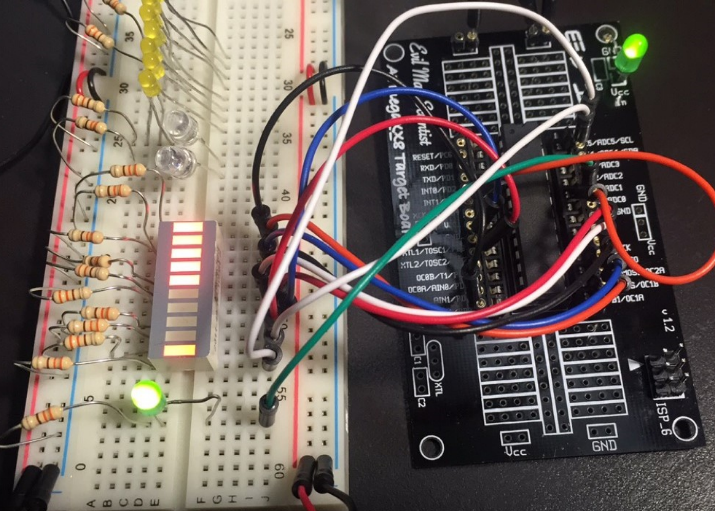
 Full Period -> 500.26ms ≈ 0.5 sec period -> Cycle Counter = 4002102(≈ 50% Duty Cycle)

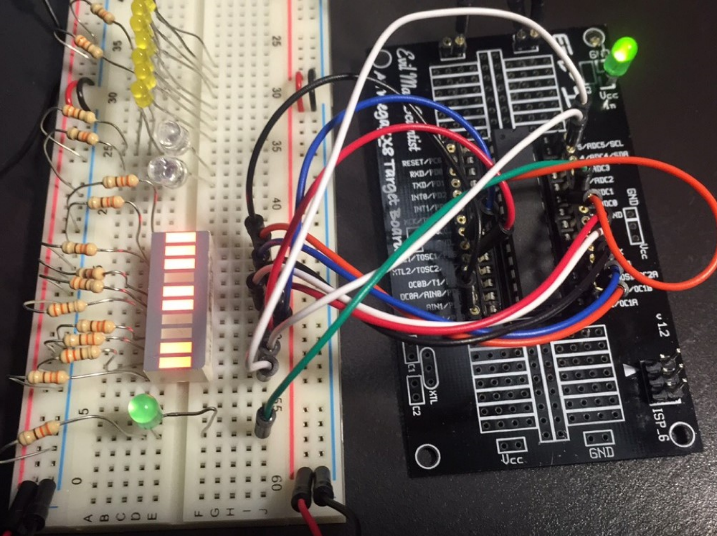
|  |  |  |  |
| --- | --- | --- | --- |
| 10. | SCREENSHOT OF EACH DEMO |  |  |



From top to bottom on LED bar respectively: PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7, PC4, PC5. The 8-bit counter is displayed from PB0-PB7 and it counts from 0 to 255 (with overflow). PC4 toggles for every 5th pulse, and PC5 is toggled for every 10th pulse. The green LED connects to PC0 and it is used to generate a half second pulse for the counter (with 50% DC).

 9th bit is toggled for every 5th rising pulse

 10th bit is toggled for every 10th rising pulse

 9th bit and 10th bit can both be toggled

|  |  |  |  |
| --- | --- | --- | --- |
| 11. | VIDEO LINK OF DEMO |  |  |
|  | https://www.youtube.com/watch?v=5Q0N-Fxzzh0 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 12. | GITHUB LINK OF THE DA |  |  |
|  | https://github.com/magor1/embedded-design-VM.git |  |  |

**Student Academic Misconduct Policy**

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

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

RALPH MAGO