Luis Ruiz

Assignment 2

CPE 301 – 1001

0. PART 0

This assignment took me about 3 hours to do, I had to modify my code quite a bit and generating flow charts always take me a good hour.

1. PART A

DA2_1: Assembly No Interrupt

This assembly code generates a clock with a period of about 0.5 second at 50% duty cycle. The clock toggles at every 0.25 sec from LOW to HIGH. Aside from generating a clock the, the code has a count that is being outputted to PORTD, I do realize the homework says to output to PORTB; however, my development board on PORTB only has 5 pins. In addition to outputting the count to PORTD, PORTC has few outputs as well, where PORTC.0 is the clk, PORTC.4 is a signal that toggles every 5 rising edges, and PORTC.5 is a signal that toggles every 10 rising edges. The code consists of a loop that loops forever and a loop that generates a delay for the clk that I am generating. After the delay portion is where I output my count to PORTD and the signals to PORTC of course what gets outputted depends if the clk recently hit a rising edge of falling edge. The signals excluding the clk are displayed on a led bar where the clk is displayed on a separate LED.

DA2_2: Assembly with Interrupts

The assembly code with Interrupts is very similar to the code described above, the most significant difference is that this code uses interrupts. Before I move on I would like to mention that both assembly codes use a pre scalar of 1024 generating a clock rate of 7.81 kHz. Moving forward, this code starts off similar to the code above except instead of manually polling the flag, I enable global interrupts and an allow forever loop to generate interrupts. Of course I enable Timer0 and global interrupts before entering the forever loop. Once an interrupt is generated the interrupt service routine (ISR) is called; once in the ISR we basically do the same as the assembly code above, it updates my count, toggles a few signals and outputs them. To recall we output the count to PORTD and the other signals to PORTC.0, PORTC.4, and PORTC.5. Just a note PORTC.4 is the signal that gets toggled every 5 rising edges and

PORTC.5 is the signal that gets toggled every 10 rising edges, I did this because my development board doesn't have a PORTC.6.

DA2_3: C/C++ No Interrupt

This code like the two above assembly code, it generates a clk and outputs a count every rising edge of the clk. However, this code is written in C instead of assembly which makes a little bit neater to read. Continuing with the discussion the code enables Timer0 with a pre scalar of 1024 after doing this we go into a loop that generates a delay in order to have 0.5 sec delay. The loop that generate the delay is a nested for loop that is used to generate 1953 clock cycles before toggling the clk. I basically translated the assembly code with no interrupts into this C code with no interrupts.

DA2_4: C/C++ with Interrupts

Like before this C code is a translation of the assembly code with interrupts. They both have the same flow to them, the only difference is the language in which they were written in. Aside from that this code like every other one outputs a count to PORTD and signals that toggle to PORTC. It has loop that goes forever in order to generate an interrupt and then go in to the ISR function which within generates a delay and outputs given signals. This is done over and over until the user terminates the program and or erases the program off the chip.

PART B

DA2_1: Assembly No Interrupt

```
;
; DA2_1.asm
;
; Created: 2/7/2017 5:36:18 PM
; Author : Luis
;


.CSEG ;Code Segment

;DEFINE REG'S AND SOME CONST
.DEF COUNT = R16     ;Define Count as R16
.DEF COUNT5 = R17    ;Define Count5 as R17
.DEF COUNT10 = R18   ;Define Count10 as R18
.EQU FIVE = 0x05     ;FIVE = 5
.EQU TEN = 0x0A             ;TEN = 10
```

```
.DEF CLK = R19              ;Define CLK as R19
.DEF PC_5 = R20             ;PB5 will be used to toggle PC.5
.DEF PC_6 = R21             ;PB6 will be used to toggle PC.6
.DEF ZERO = R2              ;Define Zero as R2

main:
       CLR CLK                      ;R19 = 0
       CLR ZERO             ;R2 = 0
       CLR R23                      ;R23 = 0
       LDI R22, 0XFF ;R22 = 0b1111_1111
       OUT DDRD,R22  ;PD.0-7 are set as outputs
       LDI R22, 0X31 ;R22 = 0b0011_0001
       OUT DDRC, R22 ;PC.5 and PC.4 are set as outputs

       FOREVER:
              OUT TCNT0, ZERO             ;Load Timer0 = 0
              OUT TCCR0A, ZERO     ;Timer0: normal mode, internal clock

              ;Timer0: enabled, CLk(i/o)/1024
              LDI R23,(1<<CS00)
              ORI R23, 0X04
              OUT TCCR0B, R23

              ;Create nested loop with the outer loop being similar to
              ;for(int i = 0; i < 7; i++)
              ;the inner is a counts up till 255 till exiting
              CLR R26                                ;R26 = 0
              LDI R24, 7                      ;R24 = 7
              DELAY:                          ;do{
                     OUT TCNT0,ZERO           ;Load Timer0 = 0
                     CALL AGAIN_LOOP          ;Loop used to count for overflow
                     INC R26                         ;R26 = R26 + 1
                     CP R26, R24
                     BRLT DELAY               ;}while(R26<7);

              ;Another small delay to aquire a 0.25sec delay
              ;more accuratley
              LDI R26,95                      ;set the TCNTO to 95
              OUT TCNT0,R26
              DELAY2:                                ;do{
                     IN R25,TIFR0
                     SBRS R25,0               ;check if TOV0 is set
                     RJMP DELAY2              ;while(TCNT0 < 255)

              ;Disable Timer and Clear the TOV0 flag
              LDI R25, (1<<TOV0)
              OUT TIFR0,R25
              CLR R25                         ;R25 = 0
              OUT TCCR0B, R25           ;Disable Timer0

              ;Change the CLK
              LDI R25, 0XFF        ;R25 = 255
              EOR CLK, R25         ;toggle the pulse

              ;Check if the clock is Posedge/Negedge
              CP CLK,ZERO
              BREQ NEGEDGE         ;Branch if CLK  == 0
```

```
		;increment counter
		INC COUNT5
		INC COUNT10

		;call the function
		CALL CHECK_TOG_5

		;call the function
		CALL CHECK_TOG_10

		;OUTPUT TO PORTC
		CLR R25                     ;CLEAR R25
		;R25 bit 0 is set since CLK did a posedge
		ORI R25, 0x01 ;R25 = 0b0000_0001
		OR R25, PC_5  ;R25 = 0b0001_0001
		OR R25, PC_6  ;R25 = 0b0011_0001
		OUT PORTC, R25       ;OUTPUT TO PORTC

		INC COUNT            ;COUNT++
		OUT PORTD,COUNT

		NEGEDGE:
			IN R25, PORTC
			LDI R26, 0x01 ;EXOR PORTC.0 which represents the clock
			EOR R25, R26
			OUT PORTC, R25       ;OUTPUT TO PORTC

		RJMP FOREVER

	end:
    rjmp end
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;FUNCTIONS


	CHECK_TOG_5:
		LDI R25,FIVE         ;if(!(COUNT5 == 5))
		CP COUNT5,R25        ;      goto NOT_EQ_5
		BRNE NOT_EQ_5        ;else
		LDI R25,0x10         ;      toggle PC.4
		CLR COUNT5
		EOR PC_5,R25
		NOT_EQ_5:
	RET

	CHECK_TOG_10:
		LDI R25,TEN                 ;if(!(COUNT10 == 10))
		CP COUNT10,R25              ;      goto NOT_EQ_10
		BRNE NOT_EQ_10              ;else
		LDI R25,0x20         ;      toggle PC.5
		CLR COUNT10
		EOR PC_6,R25
		NOT_EQ_10:
	RET

	;THE ITERATIONS TILL TCNTO > 255 and
	;TOV0 is set
```

```
	AGAIN_LOOP:
	;wait for overlow
		AGAIN:
			IN R25,TIFR0
			SBRS R25,0
			RJMP AGAIN
		LDI R25, (1<<TOV0)
		OUT TIFR0,R25
	RET
```

DA2_2: Assembly with Interrupts

```
;
; DA2_2.asm
;
; Created: 2/10/2017 10:33:31 AM
; Author : Luis
;

.CSEG ;Code Segment

.org 0
	jmp main
.org 0x20
	jmp TIM0_OVF

.DEF COUNT = R16	;Define Count as R16
.DEF COUNT5 = R17	;Define Count5 as R17
.DEF COUNT10 = R18	;Define Count10 as R18
.EQU FIVE = 0x05	;FIVE = 5
.EQU TEN = 0x0A		;TEN = 10
.DEF CLK = R19		;Define CLK as R19
.DEF PC_5 = R20		;PB5 will be used to toggle PC.5
.DEF PC_6 = R21		;PB6 will be used to toggle PC.6
.DEF ZERO = R2		;Define Zero as R2
.DEF i = R24		;Define i as R24

main:
	CLR CLK			;R19 = 0
	CLR ZERO		;R2 = 0
	CLR R23			;R23 = 0
	LDI i,6			;i = 6
    LDI R22, 0XFF	;R22 = 0b1111_1111
	OUT DDRD,R22	;PD.0-7 are set as outputs
	LDI R22, 0X31 ;R22 = 0b0011_0010
	OUT DDRC, R22 ;PC.5 and PC.4 are set as outputs

	FOREVER:
		OUT TCNT0, ZERO			;Load Timer0 = 0
		OUT TCCR0A, ZERO		;Timer0: normal mode, internal clock
		LDI R23,(1<<CS02)|(1<<CS00) ;Timer0: enabled, CLk(i/o)/1024
		OUT TCCR0B, R23

		;enable overflow interrupt
		LDI R23,(1<<TOV0)
		STS TIMSK0, R23
		SEI				;enable global interrupts

		LOOP_FOREVER:
```

```
                    RJMP LOOP_FOREVER


;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;FUNCTIONS

CHECK_TOG_5:
       LDI R25,FIVE         ;if(!(COUNT5 == 5))
       CP COUNT5,R25        ;      goto NOT_EQ_5
       BRNE NOT_EQ_5        ;else
       LDI R25,0x10         ;      toggle PC.4
       CLR COUNT5
       EOR PC_5,R25
       NOT_EQ_5:
RET

CHECK_TOG_10:
       LDI R25,TEN                ;if(!(COUNT10 == 10))
       CP COUNT10,R25             ;      goto NOT_EQ_10
       BRNE NOT_EQ_10             ;else
       LDI R25,0x20         ;      toggle PC.5
       CLR COUNT10
       EOR PC_6,R25
       NOT_EQ_10:
RET


;THE ITERATIONS TILL TCNTO > 255 and
;TOV0 is set
AGAIN_LOOP:
;wait for overlow
       AGAIN:
              IN R25,TIFR0
              SBRS R25,0
              RJMP AGAIN
       LDI R25, (1<<TOV0)
       OUT TIFR0,R25
RET


TIM0_OVF:

              LDI R25, 6
              CP R24, R25
              BRNE IF_NEQ

              CLR R24                           ;i = 0

              ;Disable Timer
              CLR R25                           ;R25 = 0
              OUT TCCR0B, R25             ;Disable Timer0

              ;CLEAR Timer0 Overflow Flag (TOV0)
              LDI R25, 1                  ;R25 = 0X01
              OUT TIFR0, R25

              ;Change the CLK
```

```
            LDI R25, 0XFF        ;R25 = 255
            EOR CLK, R25         ;toggle the pulse

            ;Check if the clock is Posedge/Negedge
            CP CLK,ZERO
            BREQ NEGEDGE         ;Branch if CLK  == 0

            ;increment counter
            INC COUNT5
            INC COUNT10

            ;call the function
            CALL CHECK_TOG_5

            ;call the function
            CALL CHECK_TOG_10

            ;OUTPUT TO PORTC
            CLR R25                     ;CLEAR R25
            ;R25 bit 0 is set since CLK did a posedge
            ORI R25, 0x01 ;R25 = 0b0000_0001
            OR R25, PC_5  ;R25 = 0b0001_0001
            OR R25, PC_6  ;R25 = 0b0011_0001
            OUT PORTC, R25       ;PORTC = 0

            INC COUNT            ;COUNT++
            OUT PORTD,COUNT

            NEGEDGE:
                  IN R25, PORTC
                  LDI R23, 0x01 ;EXOR PORTC.0 which represents the clock
                  EOR R25, R23
                  OUT PORTC, R25       ;OUTPUT TO PORTC


            LDI R23,(1<<CS02)|(1<<CS00) ;Timer0: enabled, CLk(i/o)/1024
            OUT TCCR0B, R23

            ;enable overflow interrupt
            LDI R23,(1<<TOV0)
            STS TIMSK0, R23

            RJMP END_TIM0_OVF

      IF_NEQ:
            INC R24              ; i++

            ;CLEAR Timer0 Overflow Flag (TOV0)
            LDI R25, 1                  ;R25 = 0X01
            OUT TIFR0, R25

      END_TIM0_OVF:

RETI
```

DA2_3: C/C++ No Interrupt

```
/*
 * DA2_3.c
 *
 * Created: 2/9/2017 5:05:37 PM
 * Author : Luis
 */

#include <avr/io.h>

char TOGG_5();
char TOGG_10();

volatile unsigned char count5 = 0;
volatile unsigned char count10 = 0;

int main(void)
{
	unsigned char clk = 0;
	unsigned char count = 0;

    DDRD = 0xFF;     //PD.0-7 are set as outputs
	DDRC = 0x31;  //PC.5 and PC.4 and PC.0 are set as outputs
	PORTC = 0;           //PC.0-7 OUT LOW
	PORTD = 0;           //PD.0-7 OUT LOW

    while (1)
	{
		//reset count
		if( count == 255)
			count = 0;

		TCCR0A = 0;                                                    //TIMER0: Normal
Mode, internal clock
		TCCR0B = (1<<CS02)|(1<<CS00);            //TIMER0: enable, prescalar 1024 -
8MHz/1024

		//Inner loop count from 0 to 255
		//and this is done 7 time to genereate a period of about 0.5s
		for(unsigned char i = 0; i < 7; i++)
		{
			TCNT0 = 0;                                                                 //LOAD
TIMER0 = 0
			while((TIFR0 & (1<< TOV0)) == 0);        //wait for an overflow
			TIFR0 &= (1<<TOV0);                                        //clear the
overflow(TOV0)
		}

		//A Few more cycle to generate a more accurate
		//clock period
		TCNT0 = 95;                                                    //LOAD TIMER0 = 95
		while((TIFR0 & (1<<TOV0))== 0 );
		TIFR0 &= (1<<TOV0);                                  //clear the overflow(TOV0)
		TCCR0B = 0;                                                    //TIMER0: disable
timer

		if(clk == 1)
		{
			clk = 0;                                               //RESET CLK
```

```
                        count++;                                        //Count = Count + 1;
                        count10++;                                      //Count10 = Count10
+ 1;
                        count5++;

                        PORTC = (TOGG_5()|TOGG_10()| (0x01));
                        PORTD = count;                                          //OUTPUT
Count
                }

                else if(clk == 0)
                {
                        clk = 1;                                                //SET CLK
                        PORTD = count;                                          //OUTPUT
CLOCK
                        PORTC ^= 0X01;                                          //toggle the
CLK
                }

        }
}

char TOGG_5()
{
        unsigned char TOG5 = 0;
        TOG5 = 0x10 & PORTC;        //Grab the PC.4
        if(count5 == 5){                    //Check if it's at it 5th rising edge
                count5 = 0;
                TOG5 ^= 0x10;               //toggle and reset counter
        }
                                                        //else do nothing
        return TOG5;
}
char TOGG_10()
{
        unsigned char TOG10 = 0;
        TOG10 = 0x20 & PORTC;       //Grab the PC.5
        if(count10 == 10){          //Check if it's at it 10th rising edge
                count10 = 0;
                TOG10 ^= 0x20;              //toggle and reset counter
        }
                                                //else do nothing
        return TOG10;
}
```

DA2_4: C/C++ with Interrupts

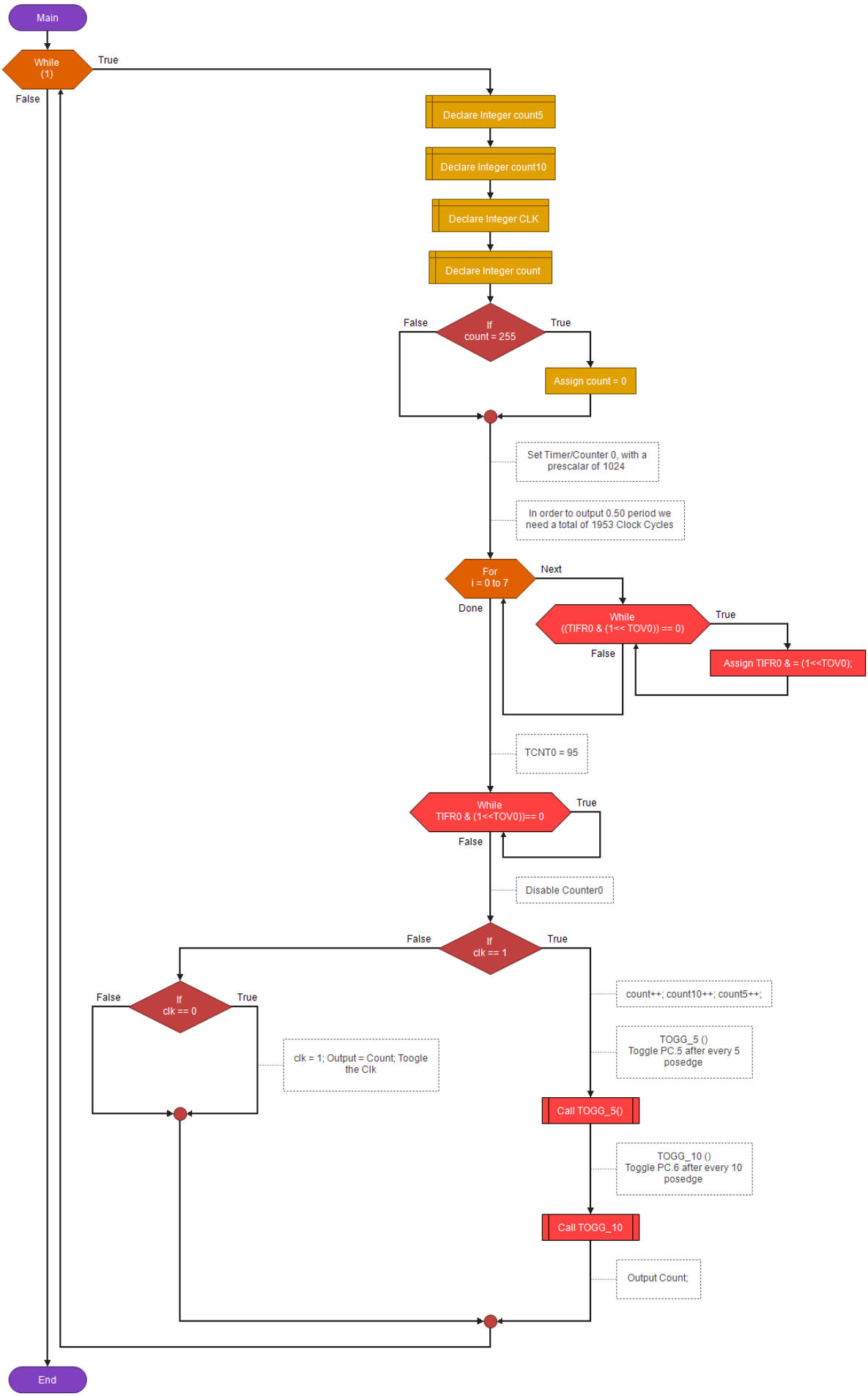
```
/*
 * DA2_4.c
 *
 * Created: 2/10/2017 7:28:47 AM
 * Author : Luis
 */

#include <avr/io.h>
#include <avr/interrupt.h>
```

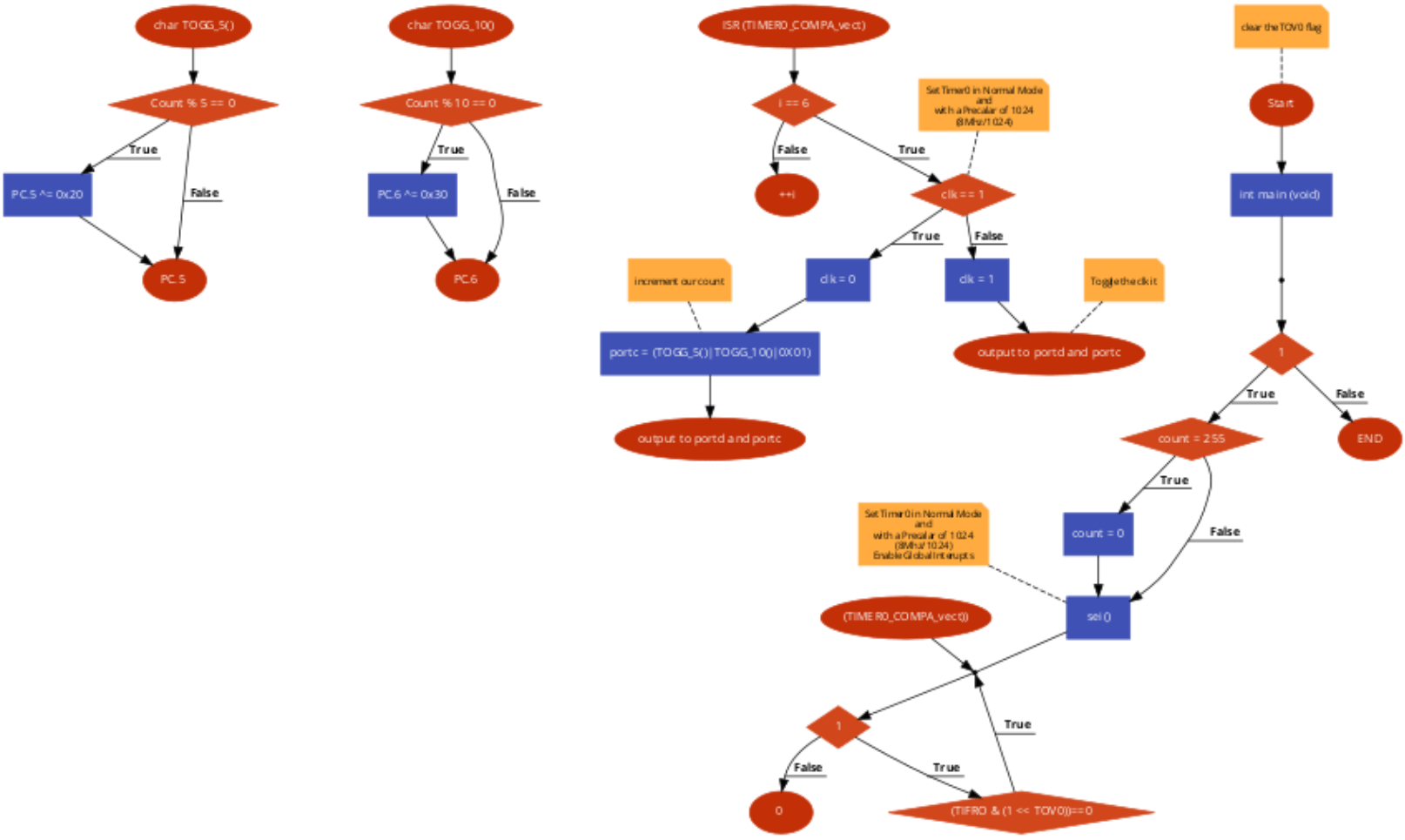
```
char TOGG_5();
char TOGG_10();
ISR (TIMER0_COMPA_vect);

volatile unsigned char count5 = 0;
volatile unsigned char count10 = 0;
volatile unsigned char count = 0;
static unsigned char i = 0 ;
static unsigned char clk = 0;

int main(void)
{

	DDRD = 0xFF;  //PD.0-7 are set as outputs
	DDRC = 0x31;  //PC.5 and PC.4 and PC.0 are set as outputs
	PORTC = 0;           //PC.0-7 OUT LOW
	PORTD = 0;           //PD.0-7 OUT LOW

	while (1)
	{
		TCCR0A = 0;
	//TIMER0: Normal Mode, internal clock
		TCCR0B = (1<<WGM02)|(1<<CS02)|(1<<CS00);        //TIMER0: enable, prescalar
1024 - 8MHz/1024
		TCNT0 = 0;
	//LOAD TIMER0 = 0
		OCR0A = 255;
		TIMSK0 |= (1<<OCIE0A);
	//Enable compare interrupts
		sei();

		while(1);
	//wait for an overflow

	}
}
//*****************************************************************
//*****************************************************************

char TOGG_5()
{
	unsigned char TOG5 = 0;
	TOG5 = 0x10 & PORTC;        //Grab the PC.4
	if(count5 == 5){                    //Check if it's at it 5th rising edge
		count5 = 0;
		TOG5 ^= 0x10;               //toggle and reset counter
	}
	//else do nothing
	return TOG5;
}
char TOGG_10()
{
	unsigned char TOG10 = 0;
	TOG10 = 0x20 & PORTC;       //Grab the PC.5
	if(count10 == 10){          //Check if it's at it 10th rising edge
		count10 = 0;
		TOG10 ^= 0x20;              //toggle and reset counter
	}
```

```
	//else do nothing
	return TOG10;
}

ISR(TIMER0_COMPA_vect)
{
	if(i == 6)
	{
		i = 0;
		TCNT0 = 95;                                          //LOAD TIMER0 = 95
		while((TIFR0 & (1<<TOV0))== 0 );
		TIFR0 &= (1<<TOV0);                                  //clear the overflow(TOV0)
		TCCR0B = 0;                                          //TIMER0: disable
timer

		if(clk == 1)
		{
			clk = 0;                                     //RESET CLK
			count++;                                     //Count = Count + 1;
			count10++;                                   //Count10 = Count10
+ 1;
			count5++;

			PORTC = (TOGG_5()|TOGG_10()|0x01);
			PORTD = count;                                       //OUTPUT
CLOCK
		}

		else if(clk == 0)
		{
			clk = 1;                                             //SET CLK
			PORTD = count;                                       //OUTPUT
CLOCK
			PORTC ^= 0x01;
		}

		TCCR0B = (1<<WGM02)|(1<<CS02)|(1<<CS00);        //TIMER0: enable, prescalar
1024 - 8MHz/1024
		TCNT0 = 0;
	//LOAD TIMER0 = 0
		OCR0A = 255;
		TIMSK0 |= (1<<OCIE0A);                               //Enable compare
interrupts
	}

	else
	{
		++i;
		TIFR0 &= (1<<OCF0A);                                 //clear the
overflow(TOV0)
	}
}
```

PART C

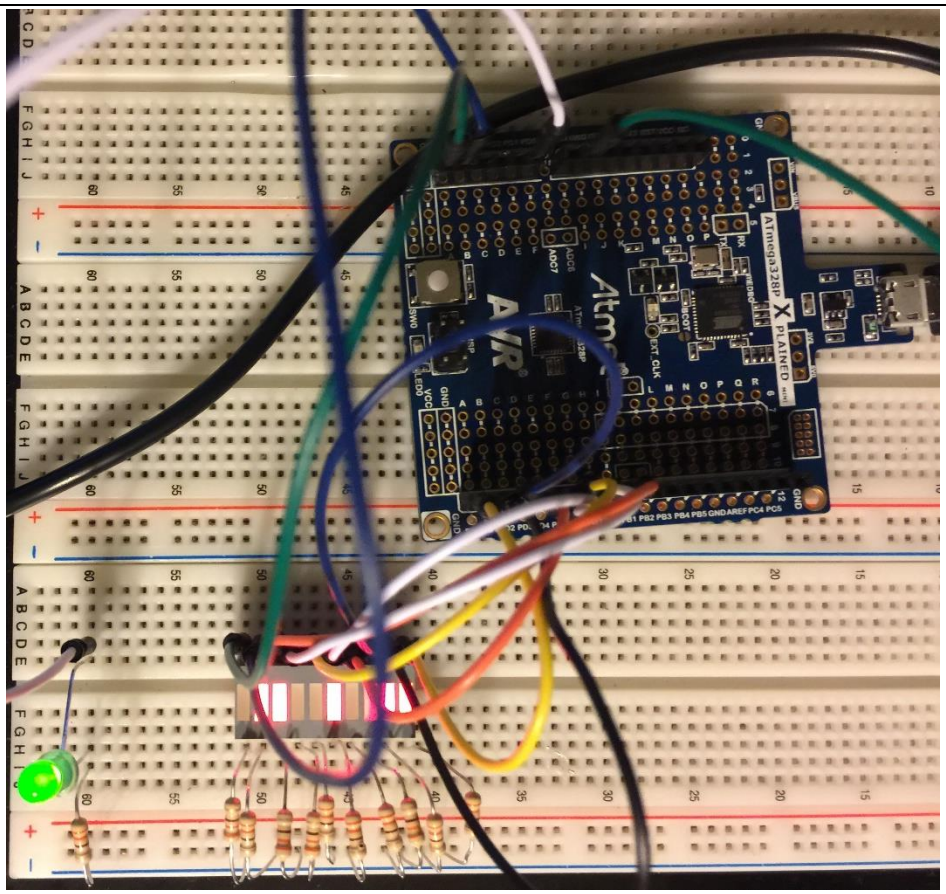
The flow chart below describes the flow of how DA2_1 an DA2_3 were coded. The only significant difference between the two was the language and size of the programs. However, I tried to keep the snippets of code that did not use interrupts very similar

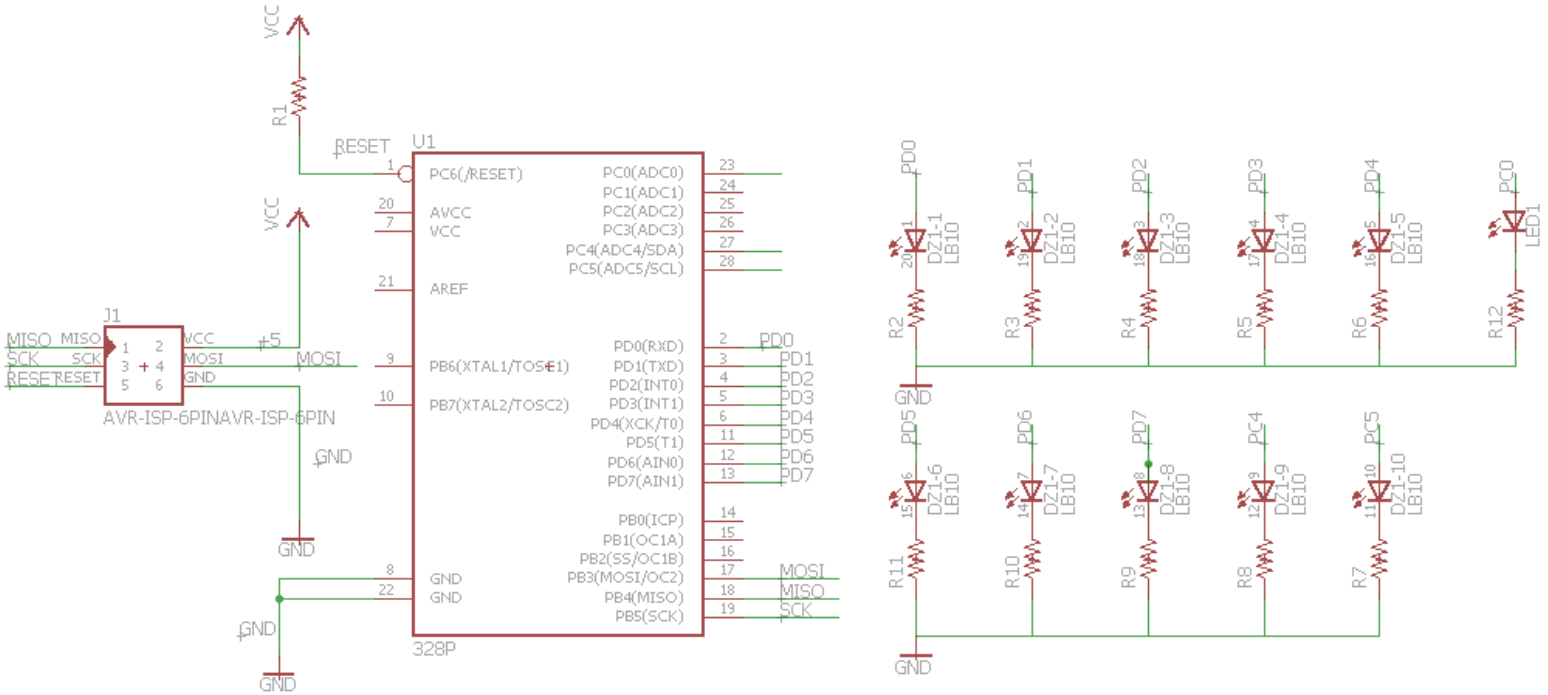

Main
While (1)
True
False
Declare Integer count5
Declare Integer count10
Declare Integer CLK
Declare Integer count
If count = 255
False
True
Assign count = 0
Set Timer/Counter 0, with a prescalar of 1024
In order to output 0.50 period we need a total of 1953 Clock Cycles
For i = 0 to 7
Next
Done
While ((TIFR0 & (1<< TOV0)) == 0)
True
False
Assign TIFR0 & = (1<<TOV0);
TCNT0 = 95
While TIFR0 & (1<<TOV0))== 0
True
False
Disable Counter0
If clk == 1
False
True
If clk == 0
False
True
clk = 1; Output = Count; Toogle the Clk
count++; count10++; count5++;
TOGG_5 () Toggle PC.5 after every 5 posedge
Call TOGG_5()
TOGG_10 () Toggle PC.6 after every 10 posedge
Call TOGG_10
Output Count;
End



The flow chart above demonstrates flow of DA2_2 and DA2_4, as before I tried to design both snippets of code that were very similar; however, the main difference was the language. They both had three functions: char TOGG_5, char TOGG_10 and ISR(TIMER0_COMPA_vect).

PART D: Schematic





PART E: YouTube Link

URL: https://www.youtube.com/watch?v=o1vD-zR6FNs&feature=youtu.be