**THE CITY COLLEGE OF NEW YORK**

160 Convent Avenue, New York, NY 10031

*The City University of New York*



**EE 42500 Computer Engineering Lab**

**Experiment 2**

**Square Wave Generation**

**Prof. Hakan Peckan**

Jonathan Martinez

Summer 2020

## Introduction

The goal of this lab is to utilize the MPLAB IDE simulator and generate a square wave with a duty cycle of 50% and a half period of 0.5ms. We will then generate another wave with a duty cycle other than 50%. We will achieve this by utilizing two pins, one pin from PORTC and another pin from PORTA. The sample code provides a good base to perform this task.

## Analysis

## 

From previous experiment we know that the subroutine LoopTime cycles once and toggles RC2 once. Given that LoopTime performs the same loop every time depending the value of Bignum we already know that RC2 will automatically have a duty cycle of 50%. Now we just need to adjust Bignum to determine the number of cycles to be removed to produce the rollover period required, this rollover period is the same as the half-period of RC2. We know that the MCU has an internal clock frequency of 2.5Mhz, therefore a period of T=0.4us. With this information we can determine the number of cycles to be removed to achieve this wave.

Therefore we can set Bignum now:

Bignum = 65536-1250+12+2

Now we have a signal on RC2 with a duty cycle of 50% and a half period of 0.5ms. We will now configure a different pin to have an output different to 50%. I will choose a period of less than 50%. We know that ALIVECNT is a powerful variable and we use it to our advantage. To achieve this we will turn off the pin at the beginning of the BlinkAlive subroutine and it will be turned on when ALIVECNT reaches 0 from where it was set initially. I will choose 33.33% duty cycle therefore ALIVECNT should be initialized to 3 in the initial subroutine and reinitialized to 3 within the BlinkAlive subroutine. First we initialize the variables to the values that we want

Initial

.

.

.

MOVLF B'11111011',TMR0H

MOVLF B'00101100',TMR0L

MOVLF B'00000011',ALIVECNT

The timer has been set to that of Bignum which is 64300, so TMR0H is 11111011 and TMR0L is 00101100. ALIVECNT has also been changed to 3 from 4, so ALIVECNT is 00000011. We will now go to the BlinkAlive subroutine now and modify a few lines.

BlinkAlive

bcf PORTA,RA4 ; turns off RA4

decf ALIVECNT,F ;Decrement loop counter and return if not zero

bnz BAend

MOVLF 3,ALIVECNT ;Reinitialize BLNKCNT

bsf PORTA,RA4 ;turns on RA4

BAend

return

end

Rather than turning on RA4 we will now be turning it off until ALIVECNT has decreased to 0, when ALIVECNT reaches 0 we branch, reinitialize and then turn on RA4 and continue with the loop. Once it loops around once and comes back to the BlinkAlive subroutine RA4 is turned off once again. This will provide us with a duty cycle of 33.33%. Now we can proceed to running the code.

## Code Execution

The code is opened and compiled, there are a few variables of interest. We will be monitoring these variables and determining if the code works as intended.

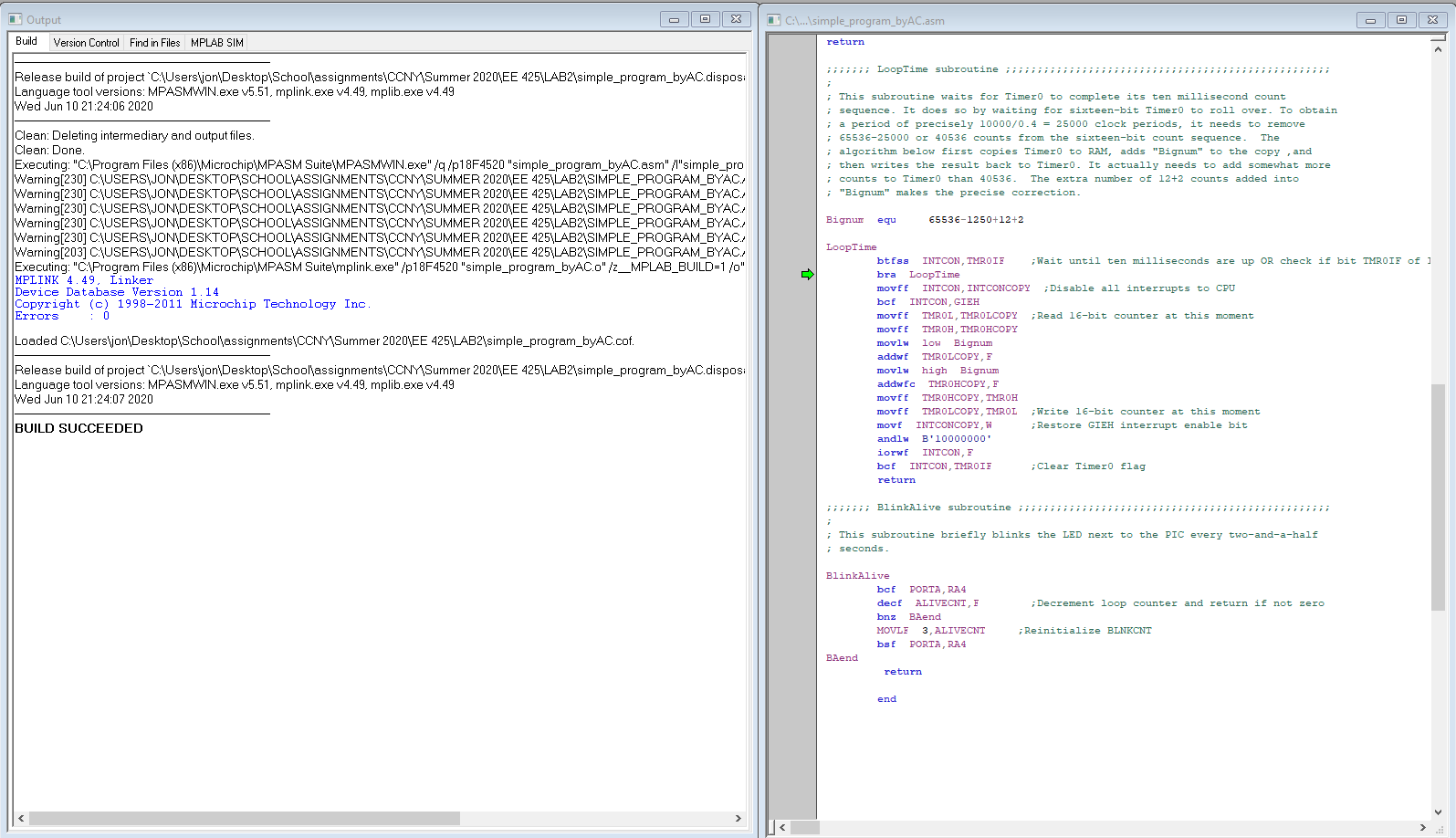


Figure 1

Once compiled successfully we will be making use of the logic analyzer to prove that we have programed our pins to work as intended.

Figure 2

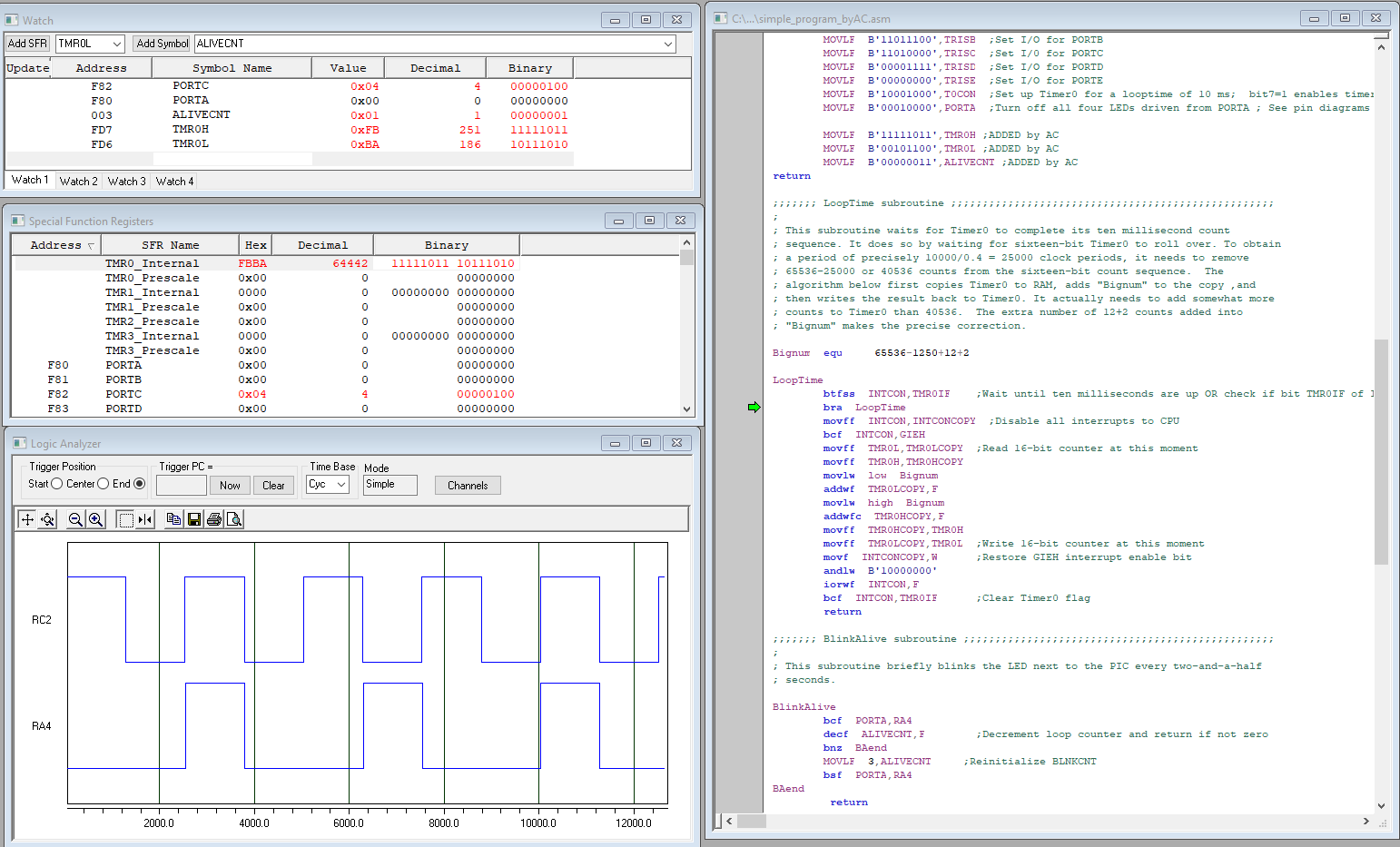
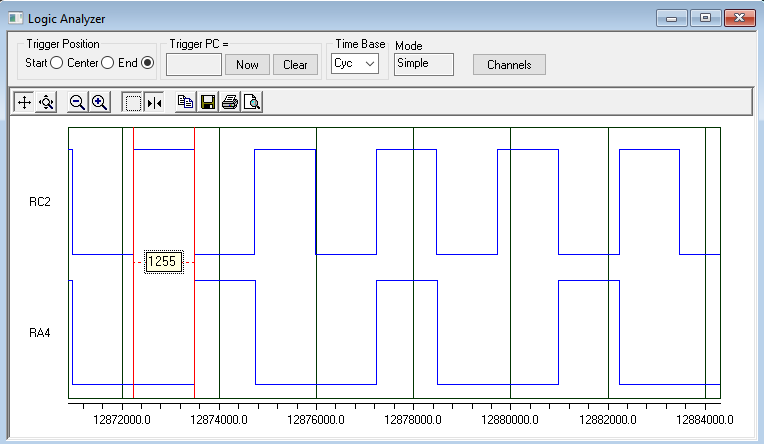


Figure 3: Logic analyzer

Here we can see the behavior of RC2 and RA4, and as expected we see that RC2 has a duty cycle of 50% with a half period of 0.5ms. We can deduce then that RA4 as a full period of 3 \* 0.5 = 1.5ms with a duty cycle of 1/3 = 33.33%. We also see that we have 1255 cycles per iteration or rollover of the program, which matches what we expect to see given our modification of Bignum.

## Conclusion

We can conclude that ALIVECNT is a powerful tool to use when generating a square wave of varying duty cycles. For duty cycles greater than 50% all we needed to do is change ALIVECNT to a number greater than 2. For a duty cycle less than 50% we need to begin the BlinkAlive subroutine by clearing the bit from port RA4 and setting the bit at the end of the subroutine, once we learn how to do this any duty cycle is possible. Bellow I will demonstrate how these ports act considering ALIVECNT.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rollover # / Iteration (each at 0.5ms)** | | | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Variable** | RC2(state) | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| RA4(state) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| ALIVECNT  (counter) | 2 | 1 | 3 | 2 | 1 | 3 | 2 | 1 | 3 |

Table 1

From table 1 we can relate ALIVECNT more closely to RA4 and confirm that this program works as intented

## Code Used

;;;;;;; P1 for QwikFlash board ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Use 10 MHz crystal frequency.

; Use Timer0 for ten millisecond looptime.

; Blink "Alive" LED every two and a half seconds.

; Toggle C2 output every ten milliseconds for measuring looptime precisely.

;

;;;;;;; Program hierarchy ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; Mainline

; Initial

; BlinkAlive

; LoopTime

;

;;;;;;; Assembler directives ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

list P=PIC18F4520, F=INHX32, C=160, N=0, ST=OFF, MM=OFF, R=DEC, X=ON

#include <P18F4520.inc>

\_\_CONFIG \_CONFIG1H, \_OSC\_HS\_1H ;HS oscillator

\_\_CONFIG \_CONFIG2L, \_PWRT\_ON\_2L & \_BOREN\_ON\_2L & \_BORV\_2\_2L ;Reset

\_\_CONFIG \_CONFIG2H, \_WDT\_OFF\_2H ;Watchdog timer disabled

\_\_CONFIG \_CONFIG3H, \_CCP2MX\_PORTC\_3H ;CCP2 to RC1 (rather than to RB3)

\_\_CONFIG \_CONFIG4L, \_LVP\_OFF\_4L & \_XINST\_OFF\_4L ;RB5 enabled for I/O

errorlevel -314, -315 ;Ignore lfsr messages

;;;;;;; Variables ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

cblock 0x000 ;Beginning of Access RAM

TMR0LCOPY ;Copy of sixteen-bit Timer0 used by LoopTime

TMR0HCOPY

INTCONCOPY ;Copy of INTCON for LoopTime subroutine

ALIVECNT ;Counter for blinking "Alive" LED

endc

;;;;;;; Macro definitions ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOVLF macro literal,dest

movlw literal ;move literal value to WREG

movwf dest ;move WREG to f= dest, which is specified by user

endm

;;;;;;; Vectors ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org 0x0000 ;Reset vector, READ Section 5.7

nop

goto Mainline ;goes to Mainline; thus skipping the interrupts below

org 0x0008 ;High priority interrupt vector

goto $ ;Trap

org 0x0018 ;Low priority interrupt vector

goto $ ;Trap

;;;;;;; Mainline program ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Mainline

rcall Initial ;Initialize everything

Loop

btg PORTC,RC2 ;Toggle pin, to support measuring loop time

rcall BlinkAlive ;Blink "Alive" LED

rcall LoopTime ;Make looptime be ten milliseconds

bra Loop

;;;;;;; Initial subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; This subroutine performs all initializations of variables and registers.

Initial

MOVLF B'10001110',ADCON1 ;Enable PORTA & PORTE digital I/O pins

MOVLF B'11100001',TRISA ;Set I/O for PORTA 0 = output, 1 = input

MOVLF B'11011100',TRISB ;Set I/O for PORTB

MOVLF B'11010000',TRISC ;Set I/0 for PORTC

MOVLF B'00001111',TRISD ;Set I/O for PORTD

MOVLF B'00000000',TRISE ;Set I/O for PORTE

MOVLF B'10001000',T0CON ;Set up Timer0 for a looptime of 10 ms; bit7=1 enables timer; bit3=1 bypass prescaler

MOVLF B'00000000',PORTA ;Turn off all four LEDs driven from PORTA ; See pin diagrams of Page 5 in DataSheet

MOVLF B'11111011',TMR0H ;ADDED by AC

MOVLF B'00101100',TMR0L ;ADDED by AC

MOVLF B'00000011',ALIVECNT ;ADDED by AC

return

;;;;;;; LoopTime subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; This subroutine waits for Timer0 to complete its ten millisecond count

; sequence. It does so by waiting for sixteen-bit Timer0 to roll over. To obtain

; a period of precisely 10000/0.4 = 25000 clock periods, it needs to remove

; 65536-25000 or 40536 counts from the sixteen-bit count sequence. The

; algorithm below first copies Timer0 to RAM, adds "Bignum" to the copy ,and

; then writes the result back to Timer0. It actually needs to add somewhat more

; counts to Timer0 than 40536. The extra number of 12+2 counts added into

; "Bignum" makes the precise correction.

Bignum equ 65536-1250+12+2

LoopTime

btfss INTCON,TMR0IF ;Wait until ten milliseconds are up OR check if bit TMR0IF of INTCON == 1, skip next line if true

bra LoopTime

movff INTCON,INTCONCOPY ;Disable all interrupts to CPU

bcf INTCON,GIEH

movff TMR0L,TMR0LCOPY ;Read 16-bit counter at this moment

movff TMR0H,TMR0HCOPY

movlw low Bignum

addwf TMR0LCOPY,F

movlw high Bignum

addwfc TMR0HCOPY,F

movff TMR0HCOPY,TMR0H

movff TMR0LCOPY,TMR0L ;Write 16-bit counter at this moment

movf INTCONCOPY,W ;Restore GIEH interrupt enable bit

andlw B'10000000'

iorwf INTCON,F

bcf INTCON,TMR0IF ;Clear Timer0 flag

return

;;;;;;; BlinkAlive subroutine ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;

; This subroutine briefly blinks the LED next to the PIC every two-and-a-half

; seconds.

BlinkAlive

bcf PORTA,RA4 ; turns off RA4

decf ALIVECNT,F ;Decrement loop counter and return if not zero

bnz BAend

MOVLF 3,ALIVECNT ;Reinitialize BLNKCNT

bsf PORTA,RA4 ; turns on RA4

BAend

return

end