**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 1**

**Using the MPLAB IDE Simulator**

**Prof. Hakan Peckan**

Jonathan Martinez

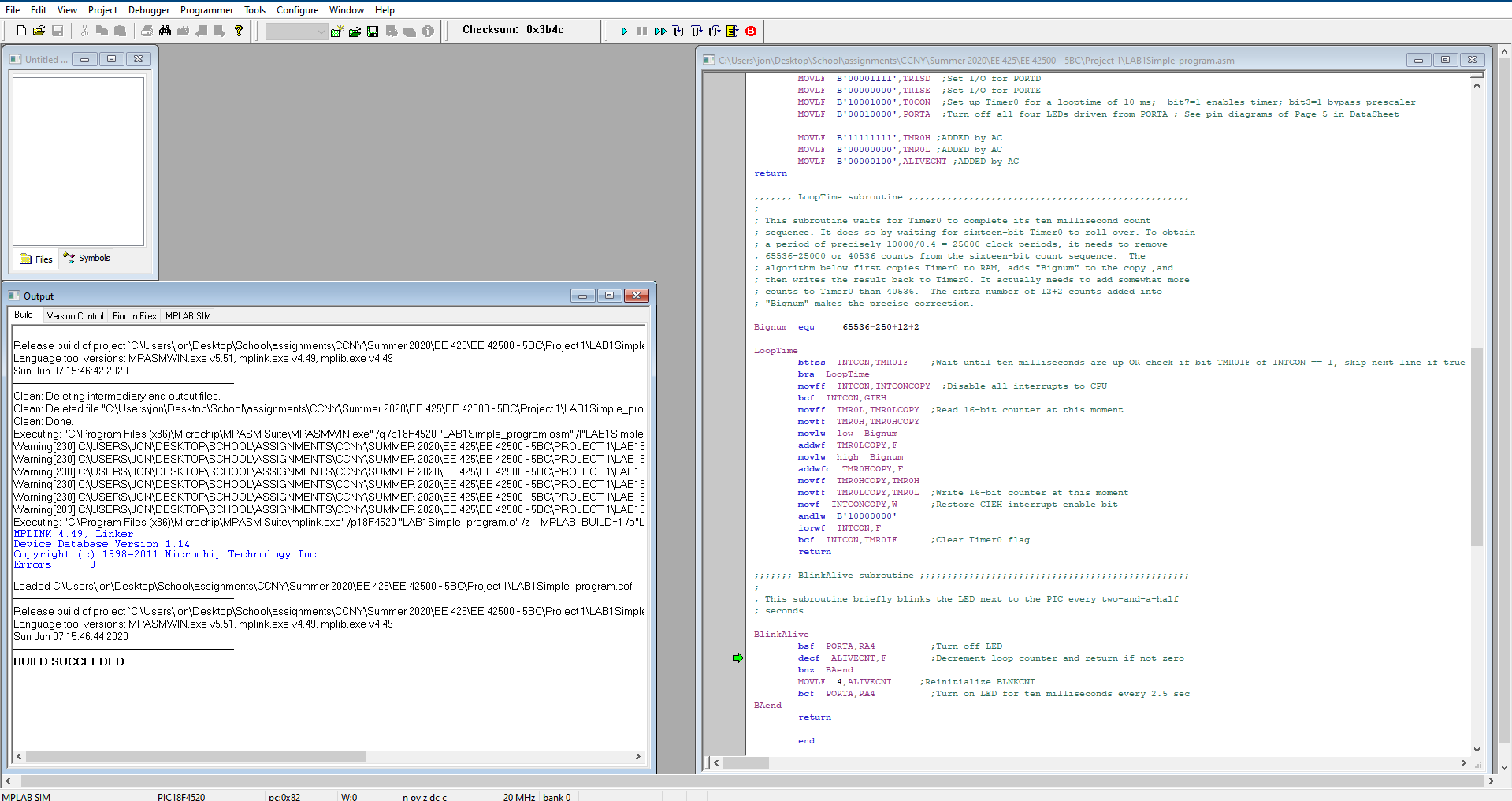
Summer 2020

## Introduction

The goal of this lab is to utilize the MPLAB IDE simulator and learn the key functions of the sample code provided. In addition we will be running this sample code and analyzing the functions that it performs. We will compare our analysis of the code to the execution of the code in the simulator. The first step however is to initialize MPLAB and load the sample program given, after achieving this figure 1 shows what MPLAB looks like with the code loaded in and compiled successfully, from this point on we can open a variety of windows and monitor specific variables of interest, these variables will be made apparent further into the report.

## Analysis

Figure 1: MPLAB with sample program



Before running the code we will answer the following questions:

How does this sample code perform the following functions:

**a**. Output a 50% duty cycle pulse train, with **0.1ms** half-period , at pin **RC2** .

**b**. Output a 75% duty cycle pulse train, with **0.4ms** full-period , at pin **RA4** .

To answer question we look at the following portion of the code

**a**.

Bignum equ 65536-250+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

We know that the MCU has an internal clock frequency of 2.5Mhz, therefore a period of T=0.4us. We also see that 250 cycles of half period have been removed by looking at

Bignum equ 65536-250+12+2

Therefore:

We also look at the following code and we see that the PORTC pin 2, or RC2 is toggled every rollover period, therefore every 0.1ms. We can now state that RC2 has a half period of 0.1ms and a full period of 0.2ms.

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

**b.**

PORTA pin 4, otherwise known as RA4 is governed by the following portion of the code. In the Initialization of variables we see ALIVECNT initialized to 4:

MOVLF B'00000100',ALIVECNT ;ADDED by AC

Then we look at the subroutine of BlinkAlive:

BlinkAlive

bsf PORTA,RA4 ;Turn off LED

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

bnz BAend

MOVLF 4,ALIVECNT ;Reinitialize BLNKCNT

bcf PORTA,RA4 ;Turn on LED for ten milliseconds every 2.5 sec

BAend

Return

ALIVECNT has been set to 4 and it will count down to 1 before reinitializing to 4. So this portion of the program has 4 times the rollover period for a full cycle to complete. In the event that ALIVECNT reaches 0 the branch is not performed, ALIVECNT is reinitialized to 4 and RA4 is turned off. This repeats every 4 iterations of the program. Therefore we make the following conclusions:

RA4 is turned on for 3 of every 4 of the iterations or “rollovers” of the code, meaning a 75% duty cycle.

We can also multiply our rollover of meaning we have a full period of 0.4ms.

## Code Execution

We will first attempt to understand the loop function from within the simulation

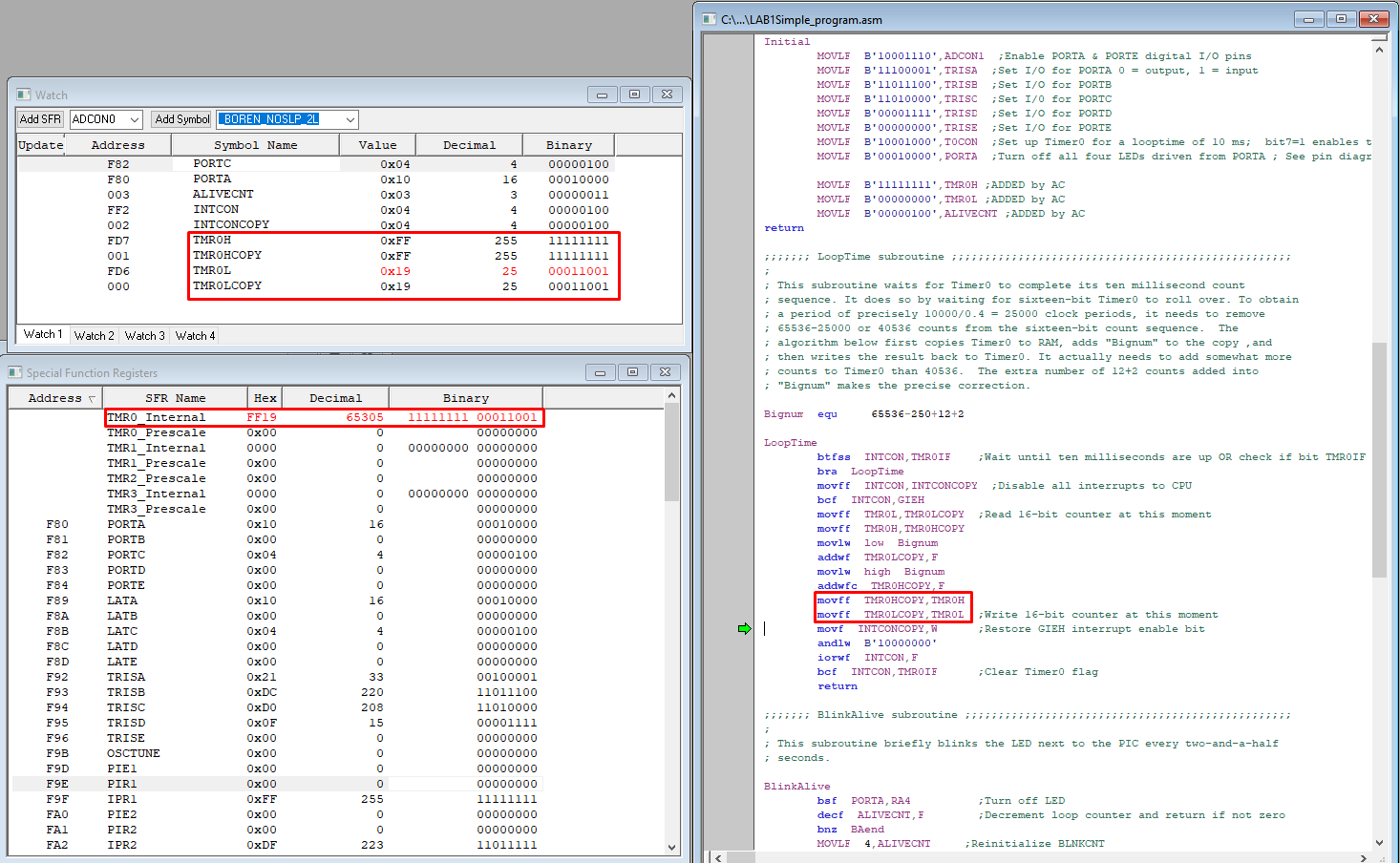


Figure2

Figure 2 shows how Bignum is used to perform a timed loop. Bignum equates to 65300 in decimal, after the loop is performed are an addition 5 steps to record and reinitialized the timers, both high and low. Therefore we see the internal timer at 65305 in the special function register windows, right after the Bignum is used to set it. This is followed by the toggling of RC2 and the counting of ALIVECNT to determine if RA4 must turn on or off. We can now combine our knowledge of the rollover period with how the simulation works. Since we know how the timer works now we will focus on RC2, RA4 and ALIVECNT.

We run the code and we look at some variables of interest. I will be looking at PORT C, PORT A and ALIVECNT. I chose ALIVECNT because this variable determines the behavior of PORT A. In order to achieve these results we need fine control of the execution of this code. I will stop and run the code in order to record the values of the variables.

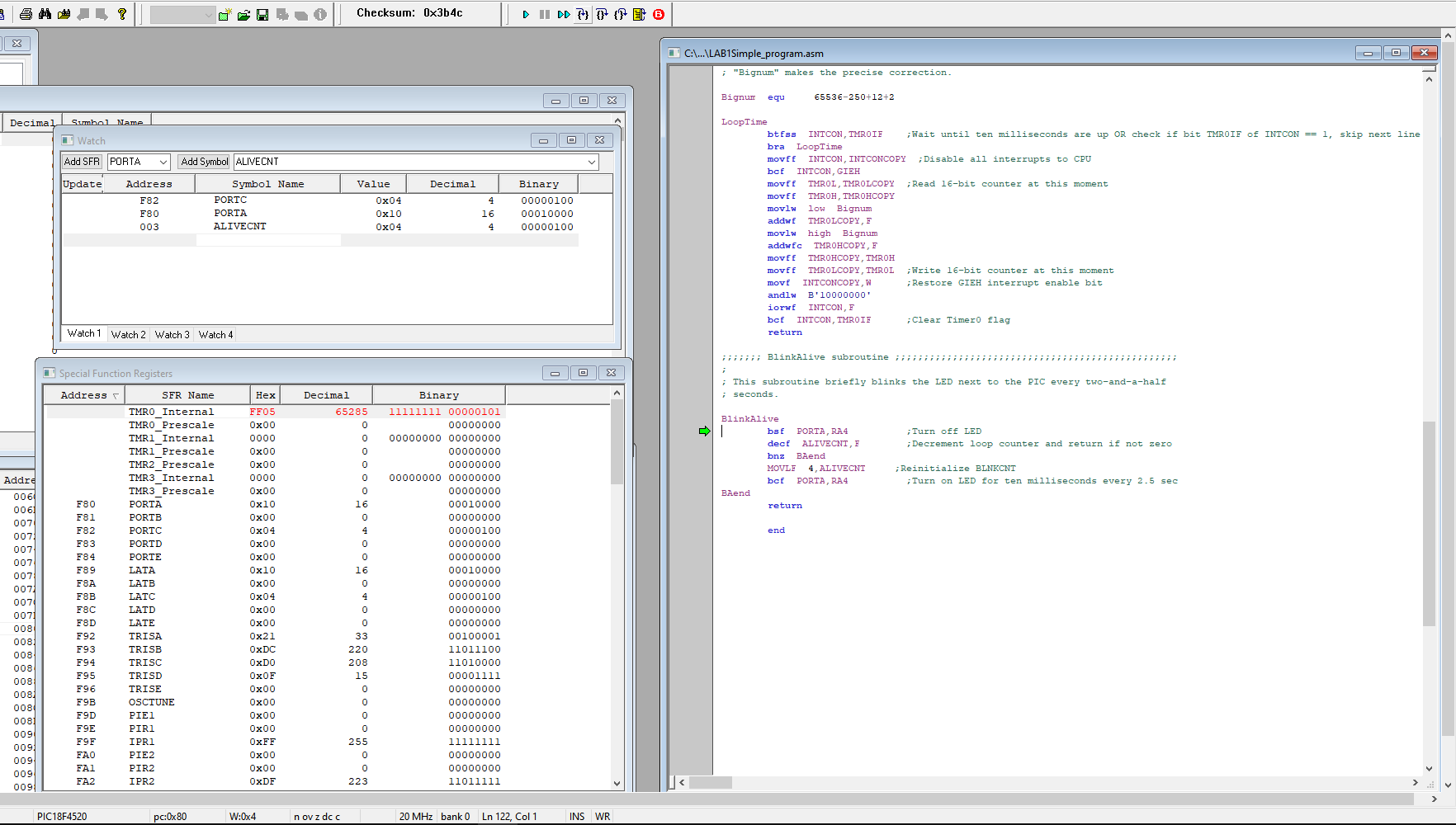


Figure 3: Variables at first initialization

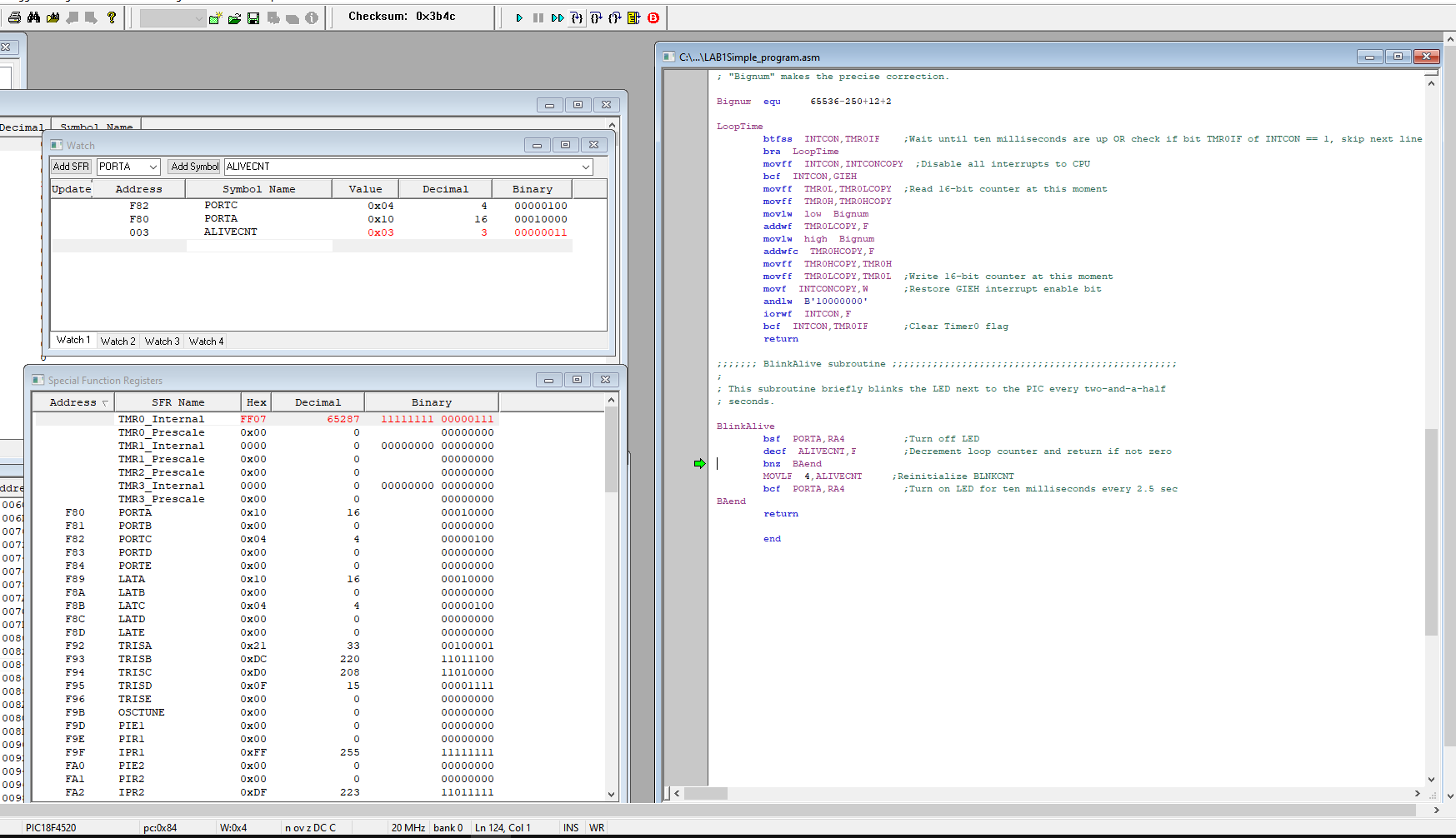


Figure 4: ALIVECNT has been decreased by 1 before the first loop

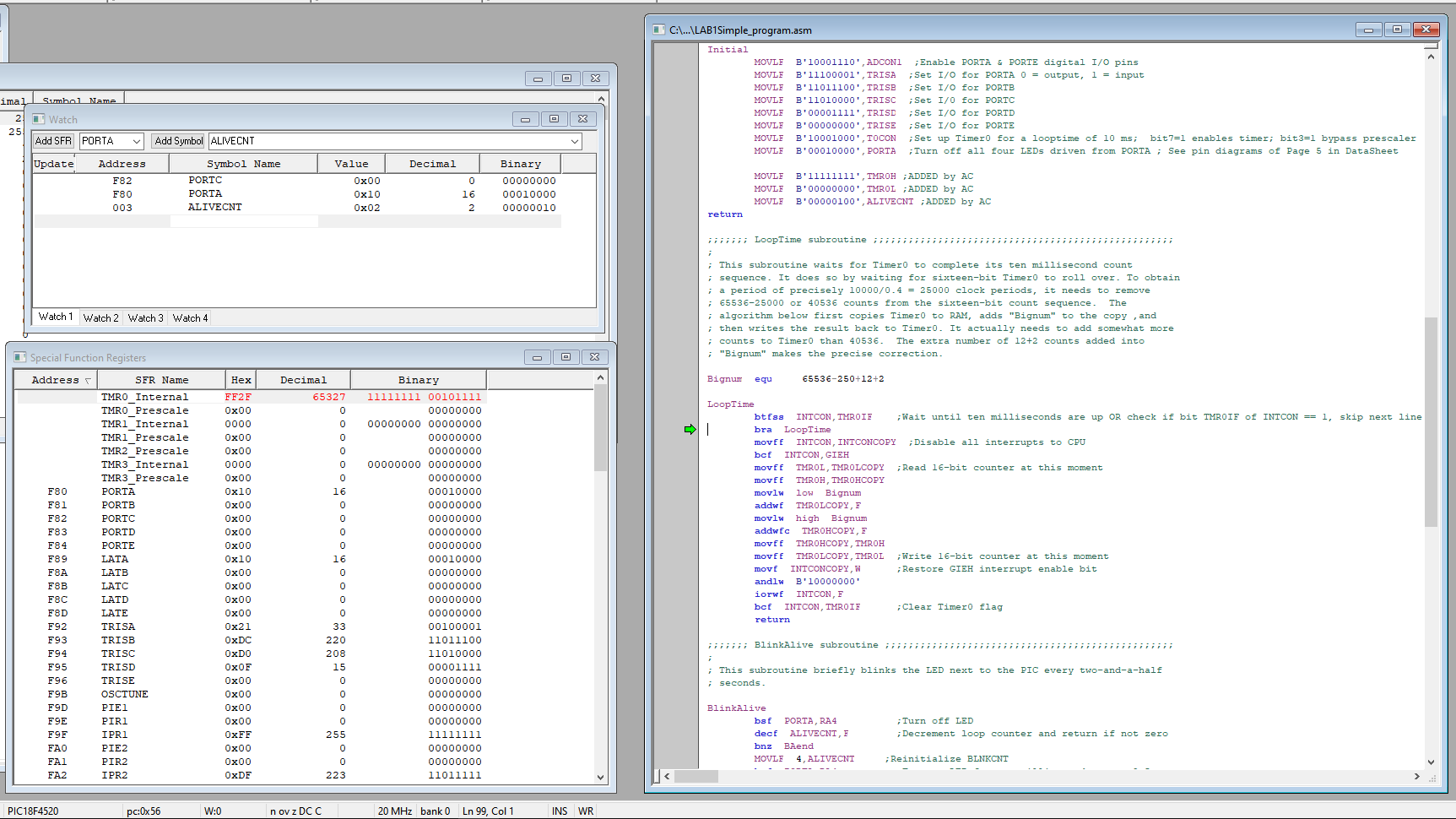


Figure 5: After the loop and the decrement of ALIVECNT

In figure 5 we see that RC2 has been toggled like it was supposed to considering the MainLine portion of the code, we also see that RA4 has not changed and ALIVECNT has decreased by 1 which once again follows what we’re expecting with the BlinkAlive subroutine.

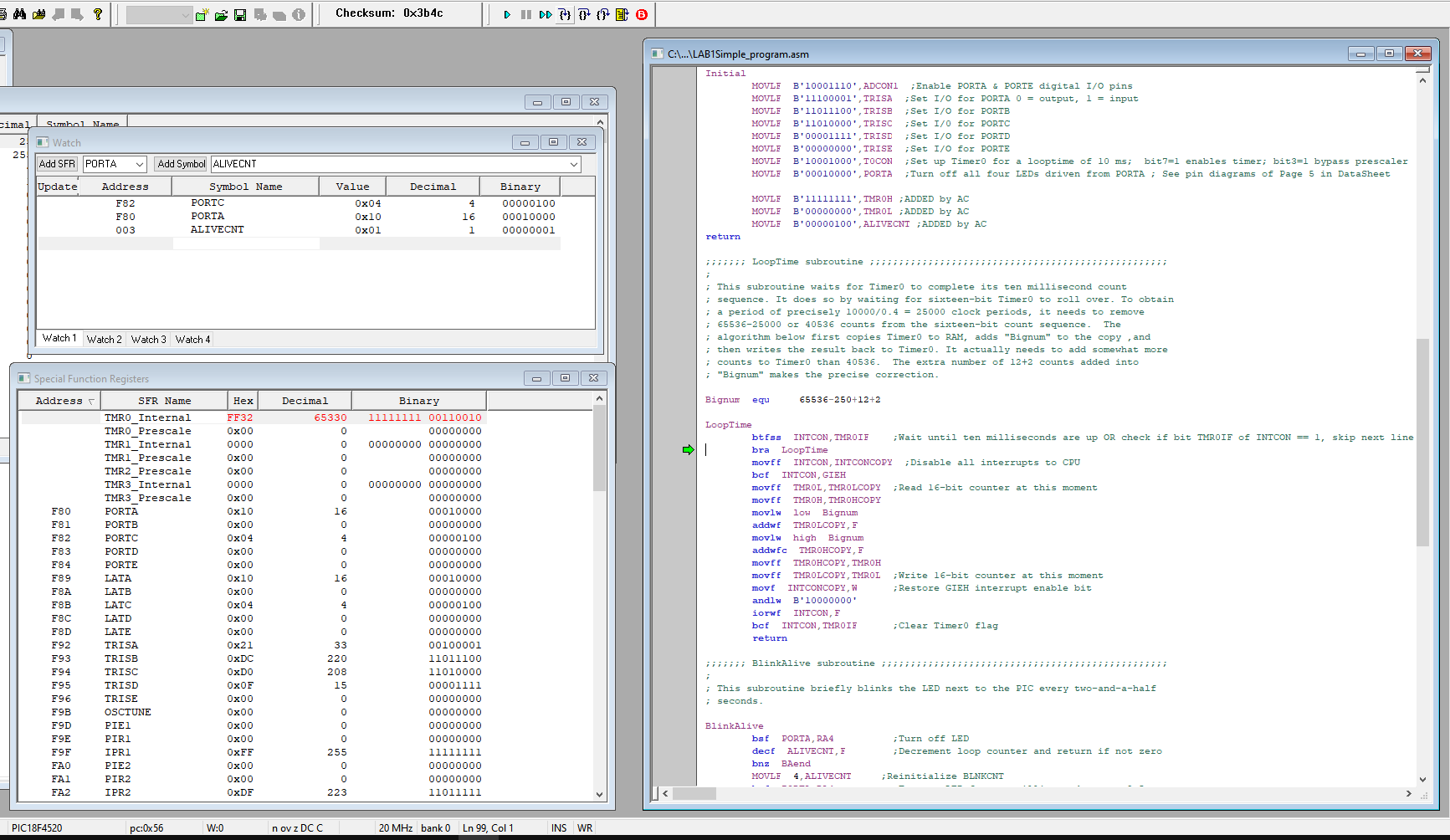


Figure 6

Figure 6 shows RC2 toggling and ALIVECNT decreasing by 1.

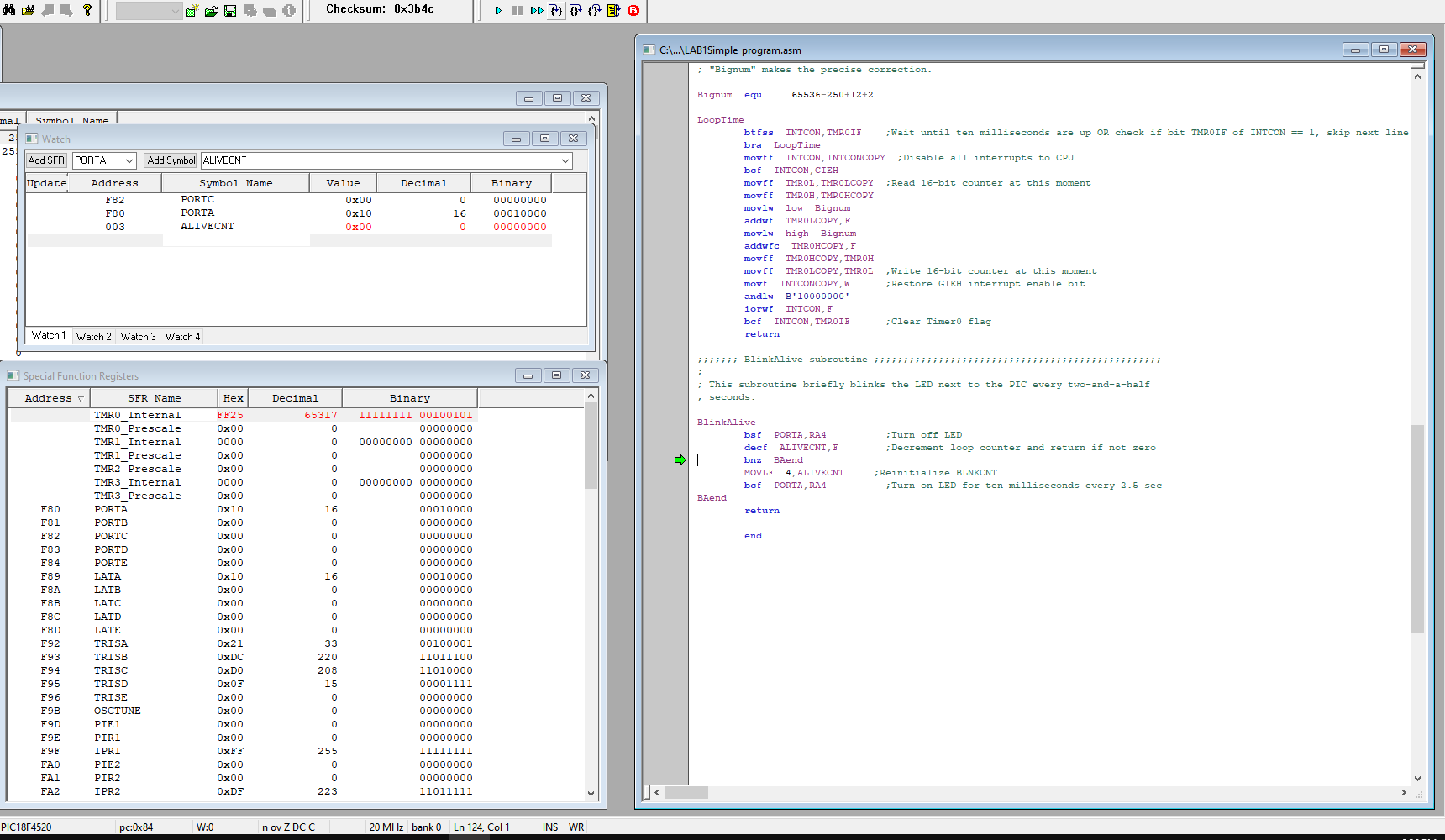


Figure 7

Figure 7 shows ALIVECNT reaching 0 for a brief moment, according to the BlinkAlive subroutine we should see ALIVECNT reinitialize to 4 and RA4 turn off, figure 8 proves this:

BlinkAlive

bsf PORTA,RA4 ;Turn off LED

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

bnz BAend

MOVLF 4,ALIVECNT ;Reinitialize BLNKCNT

bcf PORTA,RA4 ;Turn on LED for ten milliseconds every 2.5 sec

BAend

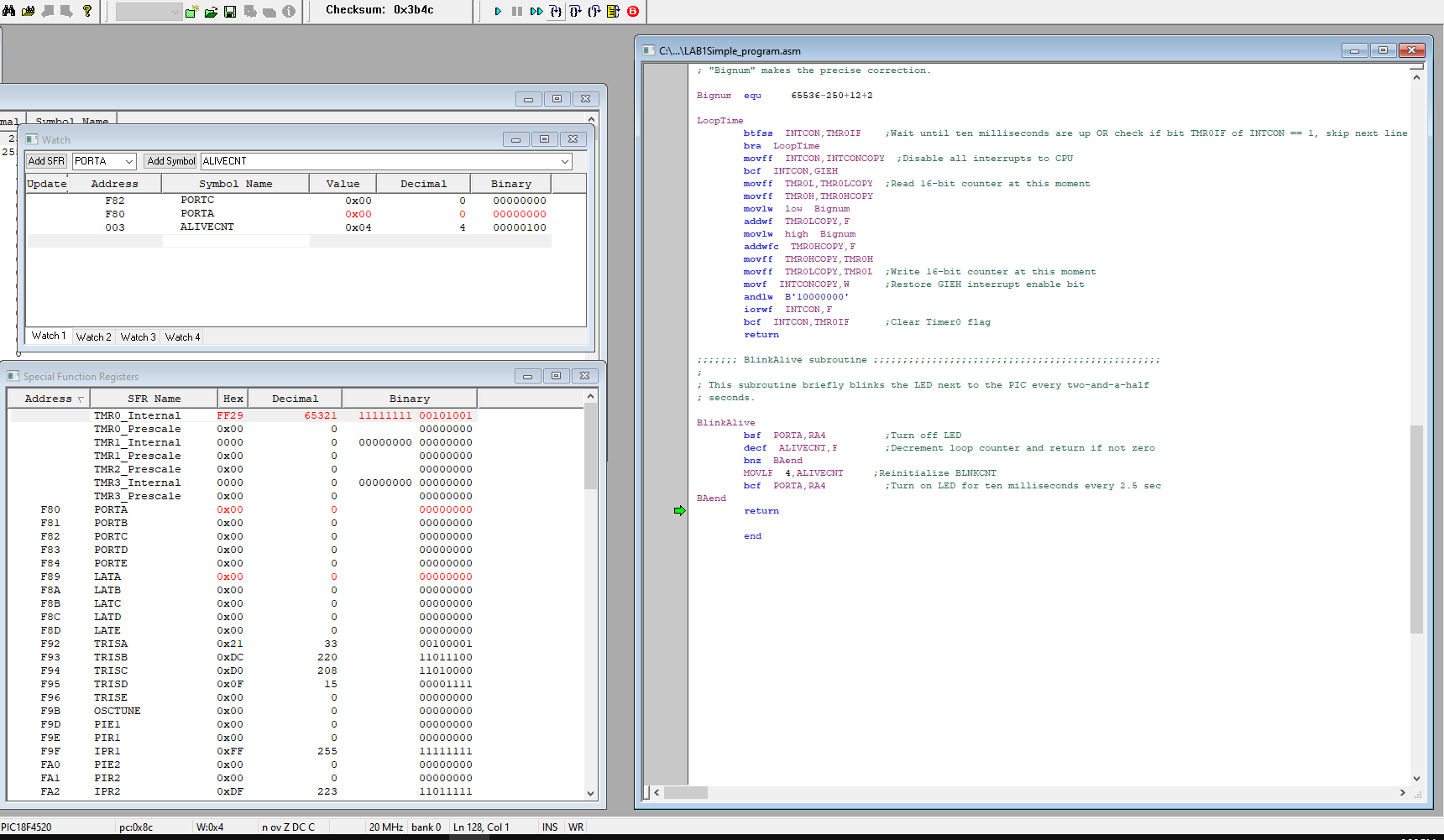


Figure 8

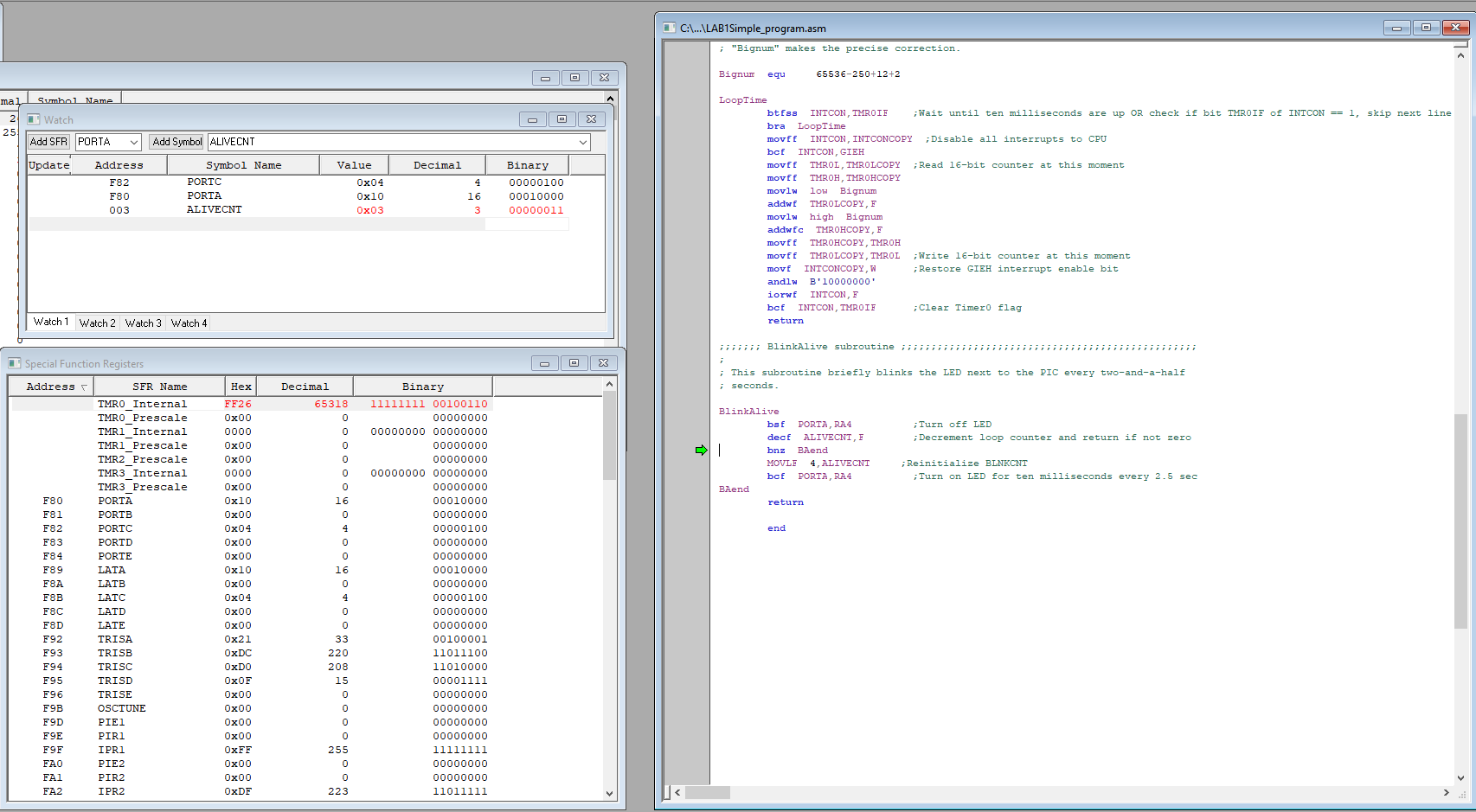


Figure 9

At this stage of the execution the loop has been made, RC2 has been toggled, RA4 has been turned on and ALIVECNT has been decreased by 1, this is shown in figure 9. We are now that the same stage from where we started (figure 4).

## Conclusion

From the previous analysis we can see that this process repeats itself indefinitely, we can make a table to predict the outcome of our code. For pins RC3 and RA4, 0 represents OFF, 1 represents ON.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rollover # / Iteration (each at 0.1ms)** | | | | | | | | | |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Variable** | RC2 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| RA4 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| ALIVECNT | 3 | 2 | 1 | 4 | 3 | 2 | 1 | 4 | 3 |

Table 1

From table 1 we can visualize the state that PORTC and PORTA would be at with respect to rollovers or iterations, we can then deduce what the . From our code analysis we had stated the following

**a**. At pin RC2 we have an output of 50% duty cycle pulse train, with 0.1ms half-period.

**b**. At pin RA4 we have an output a 75% duty cycle pulse train, with 0.4ms full-period.

If we compare table 1 to statements **a.** and **b.** we see that they agree, therefore our analysis matches our results. This concludes experiment 1, the entire sample code has been posted below.

## Sample 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'00010000',PORTA ;Turn off all four LEDs driven from PORTA ; See pin diagrams of Page 5 in DataSheet

MOVLF B'11111111',TMR0H ;ADDED by AC

MOVLF B'00000000',TMR0L ;ADDED by AC

MOVLF B'00000100',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 65536-250+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

bsf PORTA,RA4 ;Turn off LED

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

bnz BAend

MOVLF 4,ALIVECNT ;Reinitialize BLNKCNT

bcf PORTA,RA4 ;Turn on LED for ten milliseconds every 2.5 sec

BAend

return

end