**Report**

**Laboratory #2**

*“Timing Loops & 7-Segment Displays”*

by

Nayara Oliva Ferreira

09/27/2012

# 1. Introduction

The purpose of this lab is to write 0 through 9 repetitively in the 7-segment LEDs connected to PORTD. Each number should appear in the 7-segment LEDs for half a second. To achieve this purpose it’s necessary the use of nested loops.

# 2. Design and Implementation

To solve the problem the first thing done was to calculate and figure it out how to make a nested loop that would take half a second. Studding nested loop it’s easy to notice that if there’s on loop inside other this will take 1+(4+3.CIN)\*COUT instructions to execute and inserting another loop inside this two loops it’s going to take 3 + 4COUT + 4MID\*COUT + 3CIN\*MID\*COUT where COUT it the outer loop , MID it’s the middle loop and CIN it’s the inner loop. Substituting COUT for 5 , MID for 130 and COUT for 255 we have 499,873 instructions that will be executed. Multiplying 499,873 for 1us , time that the PIC takes to execute one instruction , we have 0.499873 seconds executing the loop ,time required for the this lab.

After knowing how to keep the 7-segment LEDs on for half a second was necessary to know how to write the numbers. To make that first it’s necessary to select bank 1 so you can set RB1,RB2,RB3,RB4 of TRISB to 0 and the rest of the pins to 1 .After that it’s necessary to set TRISD to output. The next step is to select bank 0 so it’s possible to set RB1 in PORTB to 1 to display our count on the left-most 7-segment display. Then it’s necessary to write the numbers to PORTD and call the delay , after writing the last number (9) return to the beginning (0).

# 3. Test Results

Debugging the program in the device , this made exactly as wished. Counted 0 through 9 repetitively. To make sure the program was displaying the number for half a second I made right click on the project selected properties there I selected the simulator. Then I clicked on Window/Debugging/Stop Watch . After that I “clean and build” the code , and inserted a break point as showed in the figure 1.

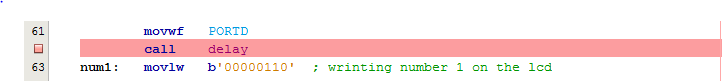


Figure 1

The next step is to “Launch the debugger” and “countinue(F5)” to debbug the program. The Stop Watch will show 13us of execution until calling the delay ( nested loops). To know exactly the time of the delay , it’s necessary now to insert another break point as showed in the figure 2.



Figure 2

Just clicking in “Contunue(F5)” we’ll find that the time of the delay was of 499.875 ms which is pretty close to the calculated time and required time.

# 4. Summary and Conclusions

As a precious finding I point the use of three nested loops to delay a period of half of a second to display each number in the 7-segment LEDs. Without this delay it will not be possible to see the numbers because this would the display them for approximately 2us. Like this lab in some cases it will be necessary to “waist” some time , and the best way to do it when it’s a “lot of time” (each instruction it takes 1us) is using the nested loops.

My code followed exactly the flow chart and also fulfilled the requirements needed for this lab. The figure 3 shows the execution time of the function delay.

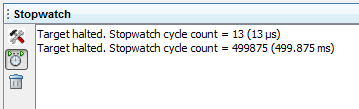


Figure 3

# Appendix A. Flow-chart

­­­­­

call delay

Display 9

call delay

Display 8

call delay

Display 7

call delay

Display 6

Display 5

call delay

Display 4

call delay

call delay

Display 3

call delay

Display 2

call delay

Display 1

call delay

Display 0

Display numbers 0-9

Display numbers 0-9

Set RB1 to 1

Select bank 0

Set TRISD to output

Set bit 1,2,3,4 of TRIB to 0

Main

Select bank 1

­­­­­

Return to the beginning

Delay

Set count = 5

Set middle count = 130

Set inner count = 255

Return

Y

Y

Y

N

N

N

Inner counter 0?

Decrement inner counter

Inner counter 0?

Inner counter 0?

Decrement inner counter

Decrement inner counter

# Appendix B

# Appendix B. Source Code

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Laboratory # 2

;\*

;\* This lab involves calculating timing loops for the PIC microcontroller. Using the 7-

;\* segment LEDs connected to PORTD, write assembly code that will display the numbers

;\* ?0? through ?9? on the 7-segment display. After displaying ?9? on the display start over by

;\* displaying ?0?. Each number should be displayed for approximately ½ second.

;\*

;\* EE3954 Microprocessors and Microcontrollers

;\* Student: Nayara Oliva Ferreira

;\*

;\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

list p=16f877 ; Assembler directive to turn list on

; and give 16F877 listing codes

; CONFIG

\_\_CONFIG 0x3F39 ; Code protect OFF, Debug DISABLED, Flash Write ENABLED

; Data EE Protection OFF, Low Voltage Programming DISABLED,

; Brown-out detection DISABLED, Power-up Timer DISABLED,

; Watchdog Timer DISABLED, XT Oscillator slected

;

; Define some 8-bit file register labels

; See PIC16F877 datasheet for location of the Special Function Registers

;

PORTB equ 0x06 ; in BANK 0

TRISB equ 0x06 ; in BANK 1

PORTD equ 0x08 ; in BANK 0

TRISD equ 0x08 ; in BANK 1

STATUS equ 0x03 ; in all BANKS

L equ 0x30

K equ 0x31

J equ 0x32

W equ d'0' ; destination=W ('working' register)

F equ d'1' ; destination=f

RP1 equ d'6' ; bank select high bit in status reg.

RP0 equ d'5' ; bank select low bit in status reg.

org 0x000 ; program starts at Program Memory location 0x0000

nop ; nop required by older MPLAB ICD system

;

; First allow program to initialize registers

;

Init: clrf STATUS ; clear status reg, to select BANK0

bsf STATUS,RP0 ; select bank 1 so we can access the TRISD register

movlw B'11100001' ; set up I/O data for configuring PortB

movwf TRISB ; set bit 0,5,6,7 of PORTB to 1:input; all other bits to 0:outputs

movlw B'00000000' ; set up I/O data for configuring PortB

movwf TRISD ; set all bits to 0:outputs

bcf STATUS,RP0 ; select bank 0 so we can access the PORTD register

clrf PORTD ; set all bits to '0' - turns off all segments

movlw B'00000010' ; move the binary number B'00000010' to PORTB,

movwf PORTB ; this will turn on the left-most 7-segment display

num0: movlw b'00111111' ;writing number 0 on the lcd

movwf PORTD

call delay

num1: movlw b'00000110' ; wrinting number 1 on the lcd

movwf PORTD

call delay

num2: movlw b'01011011' ;wrinting number 2 on the lcd

movwf PORTD

call delay

num3: movlw b'01001111' ;wrinting number 3 on the lcd

movwf PORTD

call delay

num4: movlw b'01100110' ;wrinting number 4 on the lcd

movwf PORTD

call delay

num5: movlw b'01101101' ;wrinting number 5 on the lcd

movwf PORTD

call delay

num6: movlw b'01111100' ;wrinting number 6 on the lcd

movwf PORTD

call delay

num7: movlw b'00000111' ;wrinting number 7 on the lcd

movwf PORTD

call delay

num8: movlw b'01111111' ;wrinting number 8 on the lcd

movwf PORTD

call delay

goto num0

num9: movlw b'01100111' ;wrinting number 9 on the lcd

movwf PORTD

call delay

delay: movlw d'5' ;this is the OUT\_CNT starting value.

movwf L ;store it in the ?L? register (0x32).

middle: movlw d'130' ;this is the MID\_CNT starting value.

movwf K ;store it in the ?K? register (0x31).

inner: movlw d'255' ;this is the IN\_CNT starting value.

movwf J ;store it in the ?J? register (0x30).

nxt: decfsz J,F ;decrement the inner counter value.

goto nxt ;if not zero, go decrement again.

decfsz K,F ;decrement the middle counter.

goto inner ;if middle cntr is not zero, do inner again.

decfsz L,F ;decrement the outer counter.

goto middle ;if outer cntr is not zero, do middle again.

return

end