**Report**

**Laboratory #4**

*<I/O Peripheral: The Analog-to-Digital Converter>*

by

<Randal Obringer and Bryon McElwee>

<7 March 2012>

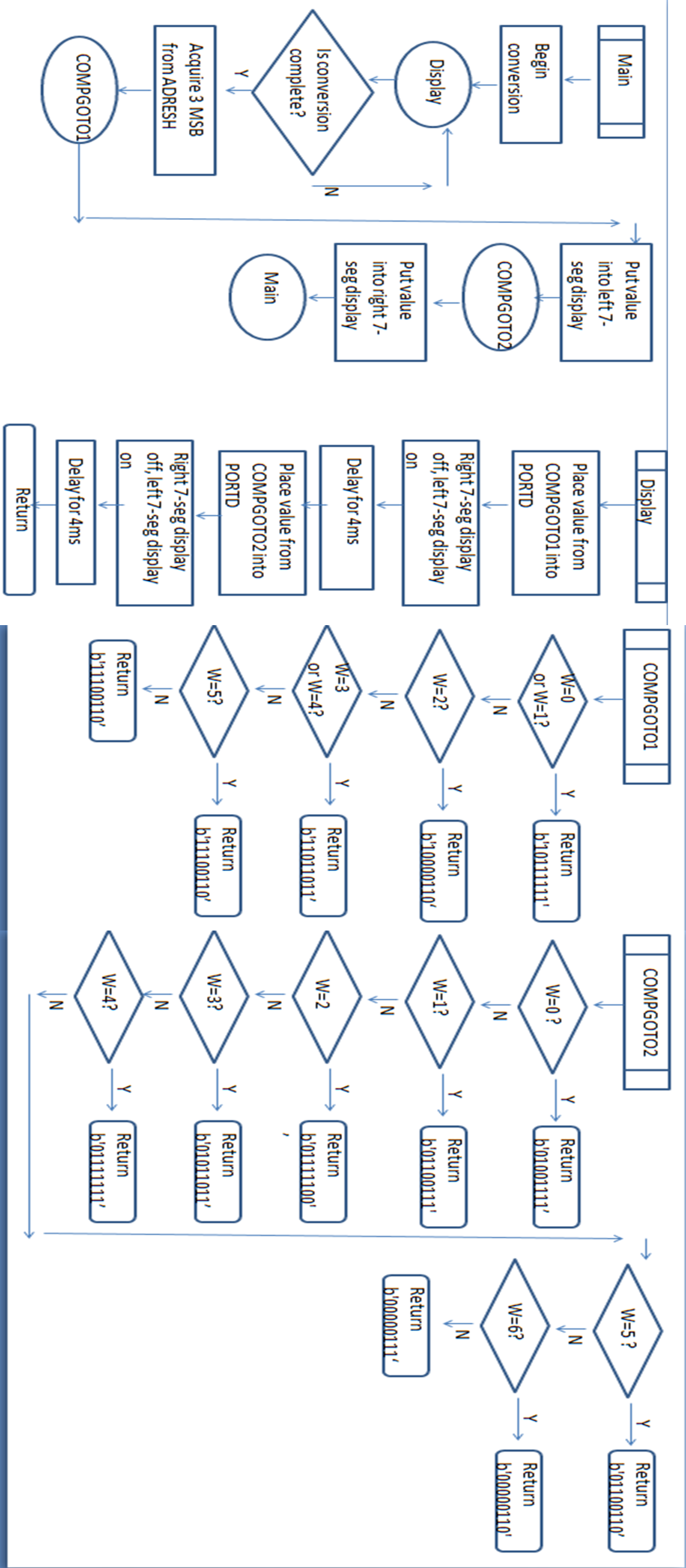
# 1. Introduction

In this lab we were given the problem of displaying the digital voltage measurement on two of the four 7-segment displays. The measurement is obtained from a potentiometer on the lab board.

Our approach was to first initialize registers. Then we would use the go bit in the ADCON0 register start our measurement of the voltage. Once we have a new value to display, it will be loaded into PORTD register to initiate its display. We would then use a delay and switch displaying between our two 7-segment displays so that it properly shows the voltage on the lab board. At the end of this process we check if the go/done bit is cleared. If it is we load the new value into the PORTD register to display the new value and then start the ADC conversion process all over again.

# 2. Design and Implementation

First, we initialize PORTB and PORTD to support output of the number segments. Then we start the ADC by setting the go/done bit in the ADCON0 register. We first display one 7-segment display then call a delay. We then display the other 7-segment display then call a delay. The constant switching and delaying of the displays gives the appearance that we are displayed the two numbers at the same time. At the end of displaying we check if there is a new value registered by the ADC converter by checking the go/done bit in the ADCON0 register. If it is still set it means we have not received a new measurement yet so we will continue displaying the two old 7-segment displays and thus showing the old value still. If the go/done bit is clear it means the conversion is complete and we will go through the process of displaying the new value. This process begins with moving the three most significant bits into the least significant three spots of the ADRESH register. This register is where the conversion value is held and completed by doing five bit shift operations. Since we only care about those three bits in question, we mask the bits and use an AND operation to move their values into w register so that we may perform a computed goto to receive the number that corresponds to that value for our display purposes. We repeat this process to receive our second display value. Once this part of the code is executed we can set the go/done bit to start receiving and converting another measurement. We then repeat the process of displaying the 7-segment displays so that we are constantly showing our measurements.



# 3. Test Results

The problem given to us provides that we must create a small timing delay to give the appearance of showing two displays at once, and we constantly switch between the two. Using 6 iterations for the outer loop and 221 iterations for the inner loop, we can produce a timing analysis algorithm of

iterations, and the inner loop being 221 iterations, we can produce a timing analysis algorithm of: 1+4(outer)+3(inner)(outer) = 1 + 4(6) + 3(221)(6) = 1 + 24 + 3978 = 4003 instruction cycles for our delay. With a 4MHz oscillator we have computed that our loop should be 4003 microseconds or approximately .004 seconds. This was sufficient time to produce the appearance of two displays going at once.

When turning on the lab board and programming it with our code we found that, when rotating the potentiometer, we could change the voltage. We were able to tell that we were changing the voltage because our 7-segment displays were displaying the voltage values we desired and no other value. Therefore we were able to receive and convert the value given by the potentiometer with the ADC correctly and furthermore display the suspected and correct voltage values.

# 4. Summary and Conclusions

Using the MPLAB simulator we were able to use the watchdog timer to test our own calculations and find a delay that correctly works with the assignment. Using the potentiometer to change the voltage values we were able to see if we were using the Anolog-to-Digital converter correctly. Through this process we were able to clarify exactly how the ADC works and better understand its conversion time and where it stores its conversions. We were also able to better understand how bit shift operations work.

# References

[1] “PICmicro Mid-Range MCU Family Reference Manual,” by Microchip Technology Inc. Literature # DS33023A.

# Source Code;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# ;\* Laboratory #4

# ;\* Program uses the microcontroller's ADC to output

# ;\* the a digital voltage measurement from RA0 to the

# ;\* 7-segment displays

# ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

# ; and give 16F877 listing codes

# ;Equivalence Statements for easier readability (Reference Manual Page 6-10)[1]

# Z equ 0x02 ;Z=2

# PCL equ 0x02 ;PCL = 0x02

# PORTA equ 0x05 ; in BANK 0

# TRISA equ 0x05 ; in BANK 1

# 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

# INTCON equ 0x0B ; in all BANKS

# K equ 0x20

# J equ 0x21

# TEMP\_W equ 0x22

# T\_STAT equ 0x23

# D1 equ 0x24

# D2 equ 0x25

# TMR0 equ 0x01

# OPTION\_REG equ 0x01

# ADCON0 equ 0x1F ;in BANK 0

# ADCON1 equ 0x94 ;in BANK 1

# ADRESH equ 0x1E

# 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.

# ;end of equ statements, begin program code:

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

# nop ; nop required by older MPLAB ICD system

# goto INIT

# INIT: clrf STATUS

# bsf STATUS, RP0 ;move to BANK1 to access TRISA, B and D

# movlw b'00000000'

# movwf TRISD ;set all pins of PORTD as outputs

# movlw b'11100001'

# movwf TRISB ;set bits 1-4 of TRISB to outputs

# ;and all other bits to inputs

# bsf TRISA, 0 ;PORTA bit 0 is now an input

# 

# movlw b'00000000'

# movwf ADCON1 ;configure ADC via ADCON1

# 

# bcf STATUS, RP0 ;return to BANK0

# 

# movlw 0x00 ;initialize displays to a value

# movwf D1

# movwf D2

# 

# movlw b'01000001' ;Fosc/4 and turn ADC ON (Reference Manual Page 23-3)

# movwf ADCON0

# 

# 

# MAIN: bsf ADCON0,2 ;set the go bit to start measurement (Reference Manual Page 23-3)

# MAIN2: movf D1,W ;move value of the display to w

# movwf PORTD ;display the value provided by the ADC

# movlw b'00000010' ;7seg display left-most

# movwf PORTB

# call DELAY

# movf D2,W ;move value of the display to w

# movwf PORTD ;display the value provided by the ADC

# movlw b'00000100' ;7seg display left-mid

# movwf PORTB

# call DELAY

# btfsc ADCON0,2 ;check if done, if done move it to be displayed, if not display old value

# goto MAIN2 ;display the old value

# rrf ADRESH,F ;process to display the new value (Reference Manual Page 23-2)[1]

# rrf ADRESH,F ;by moving the 3 most significant bits of our measurement into

# rrf ADRESH,F ;the least significant bits of the high address

# rrf ADRESH,F ;

# rrf ADRESH,F ;

# movlw b'00000111' ;mask the bits we want

# andwf ADRESH,W ;receive the actual values we want and store in w

# call FIRST\_VOL ;computed goto

# movwf D1 ;and store the returned value to be displayed

# movlw b'00000111';mask the bits we want

# andwf ADRESH,W ;receive the actual values we want and store in w

# call SEC\_VOL ;computed goto

# movwf D2 ;and store the returned value to be displayed

# goto MAIN ;start the ADC over again

# 

# ;0.004 second delay to display two voltage digits

# DELAY: movlw d'6' ;outer loop iterations

# movwf K

# INNER: movlw d'221' ;inner loop iterations

# movwf J

# NXT: decfsz J,F ;decrement inner loop and store in J

# goto NXT ;go back to decrementing inner loop if not yet 0

# decfsz K,F ;decrement outer loop and store in K

# goto INNER ;go back and do inner loop again if outer loop isn't 0

# return

# 

# ;computed goto for the 7-seg display WITH the decimal point

# FIRST\_VOL: addwf PCL,F

# retlw b'10111111' ;0.

# retlw b'10111111' ;0.

# retlw b'10000110' ;1.

# retlw b'11011011' ;2.

# retlw b'11011011' ;2.

# retlw b'11001111' ;3.

# retlw b'11100110' ;4.

# retlw b'11100110' ;4.

# 

# ;computed goto for the right-most 7-seg in our display

# SEC\_VOL: addwf PCL,F

# retlw b'01001111' ;3

# retlw b'01100111' ;9

# retlw b'01111100' ;6

# retlw b'01011011' ;2

# retlw b'01111111' ;8

# retlw b'01100110' ;4

# retlw b'00000110' ;1

# retlw b'00000111' ;7

# 

# end