**ECE3301 SPRING 2024 SESSION 3 FINAL**

Last Name: \_\_\_\_\_\_\_\_\_\_Ricciardelli\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

First Name: \_\_\_\_\_\_\_\_\_\_Daniele\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Total Score: \_\_\_\_\_\_ / 200

**Problem #1**– 40 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Indicate whether the following locations can be best accessed by the Access Bank (mark AB), by BSR (mark BSR) or None (mark None):

1. Table Pointer Upper (TBLPTRU) \_\_\_\_\_\_AB\_\_\_\_\_\_
2. Memory location 0x262 \_\_\_\_\_\_BSR\_\_\_\_\_\_
3. Status Register \_\_\_\_\_\_AB\_\_\_\_\_\_
4. PRODH \_\_\_\_\_\_AB\_\_\_\_\_\_
5. Memory location 0x80 \_\_\_\_\_\_AB\_\_\_\_\_\_

Indicates by **True of False** whether the indicated addressing mode is correct for each of the following instructions:

1. MOVWF PORTB - **Indirect** \_\_\_\_\_\_F\_\_\_\_\_\_
2. MOVLW 0x50 - **Immediate** \_\_\_\_\_\_T\_\_\_\_\_\_
3. RRNCF 0x30,W - **Bit Addressing**  \_\_\_\_\_\_F\_\_\_\_\_\_
4. ADDWF POSTINC0, W - **Direct** \_\_\_\_\_\_F\_\_\_\_\_\_
5. RESET - **Implied** \_\_\_\_\_\_T\_\_\_\_\_\_
6. CLRF POSTDEC1 - **Indirect** \_\_\_\_\_\_T\_\_\_\_\_\_
7. LFSR 2, 0x1000 - **Indirect** \_\_\_\_\_\_T\_\_\_\_\_\_
8. BCF 0x40,1 - **Direct**  \_\_\_\_\_\_T\_\_\_\_\_\_
9. True or False: In the instruction **‘BN LOOP’**, the decision to branch to the location LOOP is based on the last instruction affecting **the Z flag.**

Ans: \_\_\_\_\_\_\_F\_\_\_\_\_\_\_

1. True or False: The Status Register is part of the SFR.

Ans: \_\_\_\_\_\_\_T\_\_\_\_\_\_\_

1. True or False: The SFR registers are located in the Access Bank.

Ans: \_\_\_\_\_\_\_T\_\_\_\_\_\_\_

1. True or False: In Compare mode, the CCP1 pin must be programmed to be in input mode:

Ans: \_\_\_\_\_\_\_F\_\_\_\_\_\_\_

1. If we set PR2 to have the value of 50, with what value of the timer2 will the TMR2IF flag is set:

Ans: \_\_\_\_\_\_\_\_50\_\_\_\_\_\_

1. True or False: Timer2 cannot be used as a counter and strictly as a Timer.

Ans: \_\_\_\_\_\_\_\_T\_\_\_\_\_\_

1. What the ports affected by the ADCON1 register to program whether they operate in analog or digital mode?

Ans: \_\_\_\_\_\_\_\_PORTA (0-3,5) , PORTB (0-4), PORTE (0-2)\_\_\_\_\_\_

**Problem #2** - 20 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Find the sign, carry, and zero flags in the PIC18F for the following logic and arithmetic operations:

Assuming: [WREG] = 0x0A, [0x55] = 0x0A, [7F] = 0xF6

N flag Z flag C flag OV Flag

1. ANDLW 0xF5 \_\_\_0\_\_ \_1\_\_\_ \_\_X\_\_\_ \_\_X\_\_\_
2. SUBWF 0x55,W \_\_\_0\_\_ \_1\_\_ \_\_\_1\_\_ \_\_\_0\_\_
3. ADDWF 0x7F,F \_\_\_0\_\_ \_1\_\_ \_\_\_1\_\_ \_\_\_1\_\_
4. XORLW 0x1F \_\_\_0\_\_ \_0\_\_\_ \_\_X\_\_\_ \_\_X\_\_\_
5. INCF 0x55 \_\_\_0\_\_ \_\_0\_\_ \_X\_\_ \_\_\_0\_\_

**Problem #3** - 15 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Given that the following RAM locations 0x20 to 0x24 have already the content of the string ‘HELLO’:

(0x20) = ‘H’

(0x21) = ‘E’

(0x22) = ‘L’

(0x23) = ‘L’

(0x24) = ‘O’

Write an **assembly program using a loop** to move that string of 5 bytes to the **PORT C** using the following rule:

1. Use a FSR pointer to point to the string.
2. Use a counter for the loop
3. After a byte is moved to the PORT C, use the **call subroutine** to a library function DELAY\_ONE\_SECOND (**You don’t have to write this routine**)
4. Don’t forget to program the direction of the PORT C.

CODE:

ORG 0x0000 ;

START:

CLRF TRISC ; Clear TRISC

MOVLW 0x20 ; Load W with the starting address

MOVWF FSR ; Move starting address into FSR

MOVLW 5 ; Load W with the loop count (5 characters)

MOVWF COUNT ; Store the loop count into COUNT

LOOP:

MOVF INDF, W ; Move the byte by FSR into W

MOVWF PORTC ; Move the byte in W to PORTC

CALL DELAY\_ONE\_SECOND ; Call delay

INCF FSR, F ; Increment FSR to point to the next character

DECF COUNT, F ; Decrement COUNT

BTFSS STATUS, Z ; Check if zero flag is set (COUNT == 0)

GOTO LOOP ; If not zero, repeat the loop

END:

END ;

**Problem #4**– 20 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Assuming that the following locations have the data contents:

[0x10] = 0x55

[0x20] = 0xAB

[0x30] = 0xA5

[0x40] = 0x57

[0x50] = 0xFF

[0x60] = 0x00

ORG 0x100

MOVF 0x10,W ; w = 0x55

ADDWF 0x20, W ; w = 0x55 + 0xAB = 0x100 (0x00 W , C = 1)

ADDWFC 0x30, W ; 0x00 + 0xA5 + C = 0xA6

MOVWF 0x60 ; [0x60] = 0xA6

XORWF 0x50 ; [0x50] = 0xA6 |OR| 0xFF = 0x59

ADDLW 0xA5 ; w = 0x59 + 0xA5 = 0xFE

ADDWF 0x40,W ; w = 0xFE + 0x57 = 0x155 (0x55 W, C = 1)

MOVWF 0x40 ; [0x40] = 0x55

BNZ LABEL1 ; Does not occur as w=0x55, not 0.

XORLW 0xFF ; -------

LABEL1:

SLEEP ; continues here -> sleeps

END ; ends

Show the contents of the following memory locations:

[0x40] = \_\_\_\_\_\_0x55\_\_\_\_\_

[0x60] = \_\_\_\_\_\_0xA6\_\_\_\_\_

W = \_\_\_\_\_\_0x55\_\_\_\_\_

**Problem #5**– 20 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

**PART A)**

Assuming that the PIC18F4321 is operating at 2Mhz system clock. Find the values of TMR0H and TMR0L needed to create a time delay of 100 msec. Use the prescale of 16 and 16-bit operation**. Need to show all the calculations.**

2/4 = 0.5

0.5/16 = 0.03125 = 31.25 kHz

31.25/10 = 3125

0xFFFF – 0xC35 + 1 = 0xF3CB

**Ans:**

**TMR0H = 0x\_F3\_;**

**TMR0L = 0x\_CB\_;**

**PART B)**

Write the function T0\_Delay() to create the above mentioned delay. Determine beforehand the values of T0CON and then write the **C code** for that function:

CODE:

void T0\_Delay()

{

T0CON = 0x43;

TMR0H = 0xF3;

TMR0L = 0xCB;

INTCONbits.TMR0IF = 0;

T0CONbits.TMR0ON = 1;

while(INTCONbits.TMR0IF == 0);

T0CONbits.TMR0ON = 0;

}

**PART C)**

Write a main program to program the system clock to operate at 2 Mhz and preload an output PORT D with the value 0x5A. Then, perform an infinite loop to use the T0\_Delay() function above to flip the output of PORT D.

void main()

{

OSCCON = 0x50;

TRISD = 0x00;

PORTD = 0x5A;

While(1)

{

T0\_Delay();

PORTD = ~PORTD;

}

}

**Problem #6** – 15 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Assuming that the PIC18F is running at the frequency of 4 MHz.

1. What is the delay that can be generated if we use Timer 2 and we preload the value of PR2 with 0x7A while setting the Timer 2 with a prescale value of 1:4 and a postscale value of 1:16

PR2 = 0x7A = d`122

F\_osc = 4MHz prescale = 1:4 postcale = 1:16

F\_TIMER2 = f\_osc / (4\*prescale\*(PR2+1)\*postcale) = 127.033

TIMER2\_period = 1 / F\_TIMER2 = 0.007872s = 7.872ms

Ans: delay = \_\_\_\_\_7.872\_\_\_\_\_\_msec\_\_\_ (**must show calculation**)

**Problem #7** – 15 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Write a PIC18F C program to generate a 16 kHz PWM with a 75% duty cycle on the CCP1 pin of the PIC18F4321. Assume 4 MHz crystal. Make sure to initialize the OSCCON register and TRISx register(s).

Calculations:

**System Clock Frequency (SysFreq): 4 MHz**

Timer Clock Frequency (TmrFreq): SysFreq / 4 = 1 MHz

Frequency of PWM (Freq\_PWM): 16 kHz

**PWM Period Calculation:**

PR2 = (sysfreq/4Frq\_PWMxTMR2 prescale) – 1 = 62

*Duty cycle: 75%*

Duty cycle = CCPR1L x 4 + DC1B

CCPR1L = 0.75(62+1)/4 ≈ 11.81

CCPR1L = 47

CODE:

#include <p18f4321.h>

void initialize\_pwm(void);

void main(void)

{

initialize\_pwm();

// Main loop does nothing as PWM hardware handles the signal generation

while (1)

{

}

}

void initialize\_pwm(void)

{

OSCCON = 0x60; // Internal oscillator to 4 MHz

TRISCbits.TRISC2 = 0; // Set the CCP1 pin (RC2) as an output for the PWM signal

// PWM configuration

PR2 = 62;

CCPR1L = 47;

CCP1CON = 0x0C;

T2CON = 0x04; // Timer2 ON, prescaler = 1:1

TMR2 = 0; // Clear Timer2

// Enable the PWM output

CCP1CONbits.DC1B0 = 0;

CCP1CONbits.DC1B1 = 0;

// Start Timer2 to begin PWM output

T2CONbits.TMR2ON = 1;

}

**Problem #8** – 15 points Score: \_\_\_\_\_\_\_\_\_\_\_

Write a C language program that will toggle the CCP1 pin using the Compare mode after 30 msec. Use Timer3 with a 4MHz internal clock and 1:2 prescale. Make sure to have all the initializations done (OSCCON, CCP1CON, T3CON).

Calculations:

System Clock Frequency (SysFreq) = 4 MHz  
Timer Clock Frequency (TmrFreq) = SysFreq / 4 = 1 MHz   
Effective Timer Frequency = TmrFreq / 2 = 500 kHz  
Timer Period (TmrPeriod) = 1 / Effective Timer Frequency = 2 µs   
Required Timer Count for 30 ms = 30 ms / 2 µs = 15000

CODE:

#include <p18f4321.h>

void init\_microcontroller(void);

void main(void)

{

init\_microcontroller();

while (1)

{

// Wait until the compare event flags the interrupt (CCP1IF)

while (!PIR1bits.CCP1IF);

PIR1bits.CCP1IF = 0; // Clear the compare event flag

// Toggle CCP1 pin

LATCbits.LATC2 = !LATCbits.LATC2;

// Reset Timer3 for convenience

TMR3H = 0;

TMR3L = 0;

}

}

void init\_microcontroller(void)

{

OSCCON = 0x60; // Set internal oscillator to 4 MHz

TRISCbits.TRISC2 = 0; // Set CCP1 pin (RC2) as output

CCP1CON = 0x08; // Set CCP1 to Compare mode, interrupt on match

T3CON = 0x11; // 1:2 prescale on TIMER3

// Set CCPR1 to 15000 to get a 30ms delay

CCPR1H = 15000 >> 8; // Load the high byte

CCPR1L = 15000 & 0xFF; // Load the low byte

TMR3H = 0; // Clear Timer3 higher byte

TMR3L = 0; // Clear Timer3 lower byte

// Enable peripheral interrupts and global interrupts

INTCONbits.PEIE = 1;

INTCONbits.GIE = 1;

}

**PROBLEM #9** – 15 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Write a program to measure a voltage that is connected to the pin AN1 (RA1) and display its integer part using PORTC and the first digit after the decimal point on PORTD. Assume the following:

1. Use the function int Get\_Full\_ADC() already provided in the lab
2. the array Seven\_Seg[10] = (0x40,0x79,…} is already provided

Make sure to do the proper initialization of the ports’ directions and setup the proper values of ADCON0, ADCON1 and ADCON2 (refer to the labs)

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <xc.h>

#include <math.h>

#include <p18f4620.h>

#include <usart.h>

#pragma config OSC = INTIO67

#pragma config WDT = OFF

#pragma config LVP = OFF

#pragma config BOREN = OFF

unsigned int Get\_full\_ADC(void);

void Init\_ADC(void);

void Init\_Ports(void);

char Seven\_Seg[10] = {0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10};

int main(void)

{

unsigned int adc\_result;

unsigned int voltage\_integer;

unsigned int voltage\_first\_digit;

Init\_ADC();

Init\_Ports();

while(1)

{

adc\_result = Get\_full\_ADC();

float voltage = (adc\_result \* 5.0) / 1023.0;

voltage\_integer = (unsigned int)voltage;

voltage\_first\_digit = (unsigned int)((voltage - voltage\_integer) \* 10);

// Display integers at PORTs C and D

PORTC = Seven\_Seg[voltage\_integer];

PORTD = Seven\_Seg[voltage\_first\_digit];

}

return 0;

}

void Init\_ADC(void)

{

ADCON0 = 0x05; // Channel AN1 (RA1), ADC ON

ADCON1 = 0x0E; // Configure RA0/AN0 as analog, rest as digital

ADCON2 = 0xA9; // Given in class

//RIGHT JUSTIFY RESULT. SET BIT CONVERSION TIME //(TAD) AND ACQUITION TIME

}

void Init\_Ports(void)

{

TRISC = 0x00; // Set PORTC as output

TRISD = 0x00; // Set PORTD as output

}

unsigned int Get\_full\_ADC(void) // given

{

int result;

ADCON0bits.GO = 1; // Start conversion

while(ADCON0bits.DONE == 1); // Wait for conversion to be completed

result = (ADRESH \* 0x100) + ADRESL; // Combine result of upper and lower byte

return result;

}

**Problem** **#10** – 25 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Two push-button switches are connected to INT0 (Port B bit 0) and INT1 (Port B bit 1) and two common-anode 7-segment displays are connected respectively to Port C and Port D. In addition, a switch called ‘Up/Down’ is connected to Port A bit 0. Write a C program that will perform the following tasks.

1. If the Up/Down switch is 0, the operation is to count down. Else, it will be counting up.
2. If the push-button connected to INT0 is pressed, then the count displayed on Port C is increased or decreased based on the ‘Up/Down’ switch.
3. The same applies to the push-button connected to INT1 and the display on PORT D.
4. A while (1) loop is to be used to stay in an infinite loop.
5. External interrupt mode to be used for both INT0 and INT1.
6. The interrupts are to be generated on the **falling edge** of the INT0/INT1 pin.
7. Initialize two counters called CNT1 and CNT2 to be used to output to PORT C and PORT D respectively to 0. Make sure to output the value of those two counters to the corresponding PORT before going to the ‘while (1)’ loop.
8. Make sure to have all the ports properly initialized as well as the proper programming of the interrupt registers.

Hints:

* Use the array for the 7-segment used in the lab or provided on problem #7 of Mid-term
* You can directly output to port C or port D inside the Interrupt Service Routine (after updating the count).
* For simplicity, no need to check if the count is 0 if a count down mode is required. Same for the up count when the count is at maximum. In other words, ignore boundary checks.

CODE:

#include <p18f4321.h>

#include <delays.h>

#include <stdio.h>

#pragma config OSC = INTIO2

#pragma config WDT = OFF

#pragma config LVP = OFF

#pragma config BOR = OFF

#define UpDown PORTAbits.RA0

#define INT0 PORTBbits.RB0

#define INT1 PORTBbits.RB1

unsigned char CNT1 = 0;

unsigned char CNT2 = 0;

char LED\_7seg[10] = {0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10};

void interrupt isr(void)

{

if (INTCONbits.INT0IF == 1)

{

if (UpDown == 0)

{

CNT1--;

}

else

{

CNT1++;

}

PORTC = LED\_7seg[CNT1];

INTCONbits.INT0IF = 0;

}

if (INTCON3bits.INT1IF == 1)

{

if (UpDown == 0)

{

CNT2--;

}

else

{

CNT2++;

}

PORTD = LED\_7seg[CNT2];

INTCON3bits.INT1IF = 0;

}

}

void main(void)

{

TRISA = 0x01;

TRISB = 0x03;

TRISC = 0x00;

TRISD = 0x00;

INTCONbits.GIE = 1;

INTCONbits.INT0IE = 1;

INTCON3bits.INT1IE = 1;

INTCON2bits.INTEDG0 = 0;

INTCON2bits.INTEDG1 = 0;

PORTC = LED\_7seg[CNT1];

PORTD = LED\_7seg[CNT2];

while (1) {}

}

**Bonus Problem** – 10 points Score: \_\_\_\_\_\_\_\_\_\_\_\_

Determine the contents of the PORTx after all the instructions have been executed.

void main()

{

TRISA = 0x00;

TRISB = 0x00;

TRISC = 0x00;

TRISD = 0x00;

PORTA = 0x55;

PORTB = PORTA ^ 0xF0; // XOR operation

PORTC = PORTB | 0x05; // OR operation

PORTD = !PORTB; // Inverting operation

PORTC = PORTD >> 4; // shift right operation

PORTC = PORTB << 4; // shift left operation

PORTA = PORTC / 2; // division operation

while (1){}

}

Answer:

PORTA= \_\_\_\_\_\_\_0x28\_\_\_\_\_\_\_\_\_

PORTB = \_\_\_\_\_\_\_0xA5\_\_\_\_\_\_\_\_\_

PORTC = \_\_\_\_\_\_\_0x50\_\_\_\_\_\_\_\_\_

PORTD = \_\_\_\_\_\_\_0x00\_\_\_\_\_\_\_\_\_