

ECE 331 Sample Final Exam

Closed Notes & Book but OPEN PIC Manual

(4 hours) Coverage PIC Assembly & C and I/O material
(No 9512C32 material!)

Part 1. PIC16F877 Assembly Language Programming


1. MPASM Assembly Language (10 points)

In the PIC16F877 assembly language program below, fill in the missing code that branches to a program location labeled "my_reg_less_than_or_equal_45" if the contents of RAM (register file) location "my_reg" is less than or equal to the constant value 0x45 in an unsigned sense, and otherwise continues on to the next instruction at location "my_reg_greater_than_45".

```
list    p=16F877A
#include "p16f877A.inc"
radix   dec
test_val EQU 0x85 ;Change this test value to each of the following values and re-run:
                ; 0xff, 0x44, 0x45, and 0x46

CBLOCK 0x20
my_reg
ENDC
org 0
goto start
org 5
start    banksel test_val
        movlw test_val ;Move test value (test_val) into my_reg
        movwf my_reg
        movf my_reg,w ;put my_reg in w (this instruction may be left out, as w already holds test value)
        sublw 0x45 ; Evaluate 0x45 - test_val  $\Rightarrow C = \overline{\text{Borrow}} = \overline{0} = 1$  if test_val  $\leq$  0x45
;Insert the missing code HERE!
```

btfss STATUS, 0 ; Bit 0 of status Reg = "C" flag



```
my_reg_greater_than_45
    goto my_reg_greater_than45 ;Loop here if my_reg is greater than 0x45 in an unsigned sense
my_reg_less_than_or_equal_45
    goto my_reg_less_than_or_equal_45 ;Loop here if my_reg is less than or equal to 0x45 in an unsigned sense
end
```

2. **MPASM Assembly Language Program (27 points, 3 pts per blank)**

Assuming a **13.500 MHz** crystal oscillator, fill in the blanks in the MPASM program below that uses **TIMER 0 with interrupts disabled** to implement a "Poor Man's A/D converter" to sense the value of a variable resistor, as discussed in class. Assume you are using Pin RB0, with a 0.22 microfarad capacitor connected between RB0 and ground, and a resistive temperature transducer that varies between 1 kilohms and 18.454 kilohms connected between RB0 and the Vdd = 5.0 V dc power supply. Assume that the logic 1 threshold of RB0 is 2.25 V. Your program should first discharge the capacitor to 0 V, allowing 2.0 ms for the capacitor to be completely discharged before it is allowed to begin charging toward 5.0 V, past the 2.25 V logic 1 threshold. This program should place a value in a RAM location labeled RESULT. This value should = 0xFF when the transducer resistance is at its maximum value of 18.454 kilohms, and this number should decrease as the transducer resistance decreases. Hint: Recall the RC first-order circuit formula discussed in class: $V_c(t) = V_f - (V_f - V_i)\exp(-t/(RC))$

***** PIC16F877 MPASM Test 2 Problem 2*****

```

; POOR MAN'S A/D CONVERTER
; Senses resistance value connected between RB0 and Vdd =5V with a 0.22 uF capacitor connected
; between RB0 and ground. Assume that the RB0 logic high threshold = 2.25 V.
; RESULT = 0xFF for R = 18.454 kilohms (RESULT value decreases as R is decreased from this value.)
*****

```

```

list      p=16F877A
#include  "p16f877A.inc"
radix    dec                ;make default numbers decimal values
org      0                  ;Reset vector is at prog loc 0x00.
goto     startpgm          ;Skip over INT vector at prog loc 0x04.
CBLOCK  0x20                ;Reserve register file locations, starting at location 0x20
RESULT
ENDC
org      0x05                ;Start assembling program at location 5 in program space.

```

startpgm

banksel OPTION_REG

movlw 0x84 (BLANK 1)

; scale by 32 ⇒ tick = (4 / 13.5EG) 32 = 9.481 μs
; ALSO TURN OFF PORTB pull-ups!

;NOTE IN BLANK 1 YOU WILL HAVE TO CONSIDER CAREFULLY HOW EACH BIT OF THE
;OPTION_REG MUST BE SET TO PERMIT THE DESIRED OPERATION OF THE CIRCUIT. YOU
;MUST FIRST CALCULATE THE CAPACITOR'S CHARGING TIME WHEN THE TRANSDUCER
;RESISTANCE IS AT ITS MAXIMUM VALUE OF 18.454 KILOHMS. FROM THIS YOU CAN
;DECIDE THE PROPER PRESCALING RATIO TO YIELD RESULT = 0xFF, ASSUMING
;A 13.50 MHZ CRYSTAL FREQUENCY.

movwf OPTION_REG

doitagain

banksel PORTB

CLRF PORTB (BLANK 2)

banksel TRISB

movlw 0xfe (BLANK 3)

↔ bcf TRISB, 0 ; Lower RB0 (short out cap)

movwf TRISB

banksel TMR0

movlw 256-211 (BLANK 4)

2ms / tick time = 2ms / 9.481 μs = 210.43 ≈ 211

movwf TMR0

bcf INTCON, 2 (BLANK 5)

; clear TOIF flag

wl 2ms

btfss INTCON, 2 (BLANK 6)

; wait here for TMR0 to overflow

goto wt2ms

banksel TRISB

bsf TRISB, 0 (BLANK 7)

; make RB0 float

banksel PORTB

; thereby allowing capacitor to begin charging

clrf TMR0

wtRB0high

```

btfss PORTB, 0 (BLANK 8) ; Wait here until cap charges above
goto wtrb0high ; logic 1 threshold = 2.25V
movf TMR0, W (BLANK 9) ; Get result (TMR0 value) into "W" register
movwf RESULT
goto doitagain
end

```

3. **Poor Man's A/D Analysis 8 points, 2 points per question)**

In the program of Problem 2 above, RESULT = 0xFF => the transducer resistance was at its maximum value of 18.454 kilohms.

What value of resistance corresponds to RESULT = 0x80? 9.23 kΩ (Show your calculations below.)

$$t_x = \frac{0x80}{128} * (4/13.5EG) * 32 = 1.214 \text{ ms}$$

$$2.25 = 5 - (5-0)e^{-t_x/RC} = 5 - 5e^{-\frac{1.214 \text{ ms}}{R(0.22 \mu\text{F})}} \Rightarrow R = 9.23 \text{ k}\Omega$$

What value of resistance corresponds to RESULT = 0x20? 2.31 kΩ (Show your calculations below)

$$t_x = \frac{0x20}{128} * (4/13.5EG) * 32 = 0.3034 \text{ ms}$$

$$2.25 = 5 - 5e^{-\frac{0.3034 \text{ ms}}{R(0.22 \mu\text{F})}} \Rightarrow R = 2.31 \text{ k}\Omega$$

Explain what practical problem arises with this circuit if the transducer resistance is allowed to go below about 333 ohms.

The RB0 pin has to sink $\geq \frac{5V}{333 \Omega} = 15 \text{ mA}$

and this exceeds the output current spec! => Pin RB0 will burn out

→ Explain why the MSB (Bit #7) of the ~~OPTION~~ REG must be set to 1 if this application is to work properly. (You may have to go back and modify BLANK 1 above if you did not catch this earlier!)

IF MSB of ~~OPTION~~ REG = RBPU = 0, ^{internal} pull-up resistors are enabled which interfere with the external pull-up resistor being measured thus the measured resistance will be in error!

4. **PICLITE C Language Program 33 points, 3 points per blank)**

Fill in the 11 blanks in the PICLITE C program below that plays music in a format that is much more convenient to enter from a musical score than the format of Example 2 on p. 17 of the MPLAB PICC Tutorial that was distributed in class. In this example, the TONE array is loaded with values N = 0 - 23 which represent two octaves of the musical scale: A1, Bb1, B1, C1, Db1, D1, Eb1, E1, F1, F#1, G1, Ab1, A2, Bb2, B2, C2, Db2, D2, Eb2, E2, F2, F#2, G2, Ab2. Furthermore, let the value N = 24 correspond to the special case of silence (a musical rest). Let A1 correspond to 220 Hz, then A2 must correspond to 440 Hz, one octave above A1. Since the Western musical scale varies in 12 logarithmically-spaced steps between octaves, the frequency of each note in this scale is given by

5.

$$f = 220 \cdot 2^{N/12} \text{ Hertz, where } N = \text{the note number } (0 - 23)$$

Likewise, let the DURATION array be loaded with values that range from 1 up to 16. Let "1" represent the shortest possible note duration, let's call it a 16th note, then "2" represents a note that is twice as long (an 8th note), "4" represents a note that is 4 times as long (a quarter note), "8" represents a half note, and "16" represents a whole note. Note that with this scheme, a dotted 8th note, which is 1.5 times the length of a regular 8th note, would be represented by "3", and a dotted quarter note would be represented by "6", etc.

// PIC16F877 - Test 2 Music Playing Program

#include <pic.h>

#define SONGSIZE 10 //There are 10 notes in this song

void interrupt music_isr();

void music_init();

```

char getnoteflag, noteptr, note_number;
long int dur_nr_half_cycles;
long int SIXTEENTHNOTE = 337500; // SIXTEENTHNOTE = nr of timer ticks for 0.2 sec sixteenth note
int tone_val, dur_counter;
const char duration[SONGSIZE]={2,2,3,1, 1, 1, 2, 2, 3, 4}; //This is the RHIT fight song,
const char tone[SONGSIZE]= {6,7,8,10,11,13,15,13, 11,24}; // "Dear Old Rose!"

const int tone_table[25]={7670, 7240, 6834, 6450, 6088, 5746, 5424, //****BLANK 1
5119, 4832, 4560, 4305, 4063, 3835, 3620,
3417, 3225, 3044, 2873, 2712, 2560, 2416, //****BLANK 2
2280, 2152, 2032, 1918};

void main(void)
{
    music_init();
    for(;;);
}

void music_init()
{ getnoteflag = 1; // Set getnoteflag = 1, so first interrupt will fetch note
// from tone[ ] and duration[ ] arrays.

noteptr = 0; // Make noteptr to point to first note in tone[ ] and duration[ ].
dur_counter = 0; // Clear Duration Counter, which counts nr. of half cycles a note is played.
TICKPS1 = 0;
TICKPS0 = 1; // tick_period = (4/13.5E6)*2 = 0.592 us
TMR1ON = 1; // Start TMR 1
TMR1CS = 0; // TMR1 clock source set to internal clock (Fosc/4).
RBO = 0;
TRISB = 0xFE; // Make RBO an output
TMR1IE = 1; // Enable TMR1 interrupts
PEIE = 1; // Enable peripheral interrupts
GIE = 1; // Globally enable interrupts
TMR1ON = 0; // Turn off Timer 1
TMR1H = 0xff;
TMR1L = 0;
TMR1ON = 1; //Schedule the first interrupt in 256 ticks and turn on TMR 1
TMR1IF = 0; //Clear TMR1 Flag (shut the baby up!)
}

// This interrupt routine is entered every time TMR1 overflows, which should be every half of a note cycle.
void interrupt music_isr( )
{
    int sched_val;
    if(getnoteflag == 1)
    {
        getnoteflag = 0;
        note_number = tone[noteptr] //Look up the number of the next note ***BLANK 3
        if(note_number > 24) note_number = 24; // If an invalid note number (> 24) is entered,
// make it a rest = 24.
        tone_val = tone_table[note_number] // tone_val = nr of ticks in half cycle of note **BLANK 4
        dur_nr_half_cycles = duration[noteptr] * (SIXTEENTHNOTE / tone_val) // **BLANKS 5-6
// THIS EXPRESSION MUST INVOLVE the constant
// "SIXTEENTHNOTE", which sets the speed at which
//the musical composition is played. Note 'dur_nr_cycles'
// is the number of cycles in the note.

        dur_counter = 0; // Reset duration counter
        noteptr++; // Increment noteptr.
        if(noteptr > SONGSIZE) noteptr = 0; // If song complete, wrap back to beginning. ** BLANK 7
    }
}

```

else

```

{
    if (note_number < 24) RB0 = !RB0; // Toggle RB0 output pin ***BLANK 8
    dur_counter++; // increment duration counter
    if (dur_counter > dur_nr_half_cycles) getnoteflag = 1; //** BLANK 9
    // Set getnoteflag = 1 if at the end of the note

}

TMR1ON = 0; //Turn off Timer 1
sched_val = 65536 - tone_val; //*****BLANK 10
TMR1L = sched_val & 0xFF; //Schedule next half-cycle interrupt
TMR1H = sched_val >> 8;
TMR1ON = 1; //Turn Timer 1 back on
TMR1IF = 0; //Relax the Timer 1 interrupt flag (shut the baby up.)
//** BLANK 11
}
    
```

6. LCD Multiplexing (22 points)

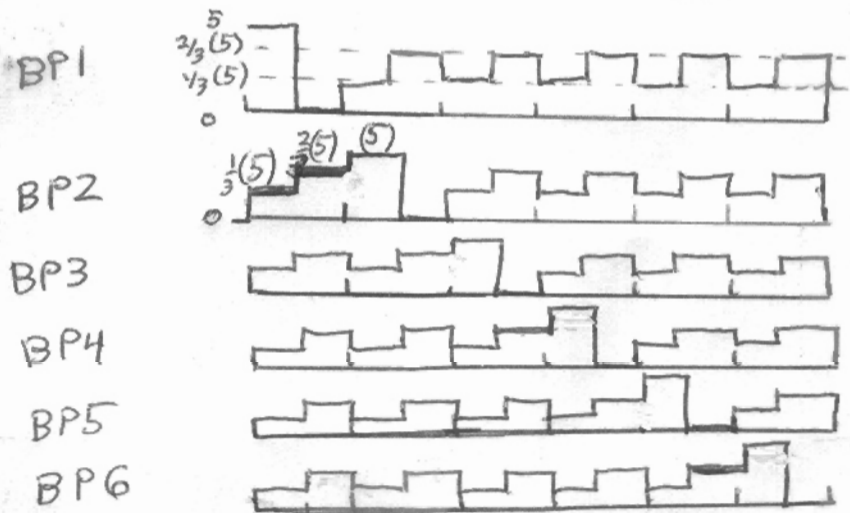
- a. A custom LCD display for a new product has 500 segments that must be individually controlled (turned on or off). If we choose to use 1/4 multiplexing on this display, implying 4 backplane signals are needed, what is the total number of wires (backplane wires plus frontplane wires) that must be connected to this display?

Total # Wires = 129 $4 + \frac{500}{4} = 4 + 125 = 129$

- b. Repeat Part A for 1/8 multiplexing. Total # Wires = 71 $8 + \frac{500}{8} = 8 + 62.5 = 71$
 $= 63$

- c. Repeat Part A for 1/16 multiplexing. Total # Wires = 48 $16 + \frac{500}{16} = 16 + 31.25 = 48$
 $= 32$

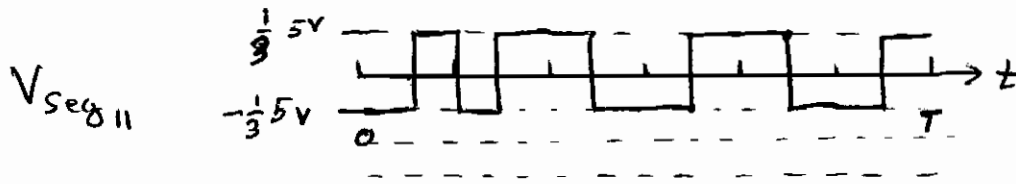
- d. For the case of 1/6 LCD multiplexing, there are 6 backplane signals, BP1, BP2, BP3, BP4, BP5, and BP6. Assume that $V_{cc} = 5V$, so the waveform voltage levels are 5V, 3.33V, 1.66V, and 0V. Sketch one frame of each of the six backplane signals.



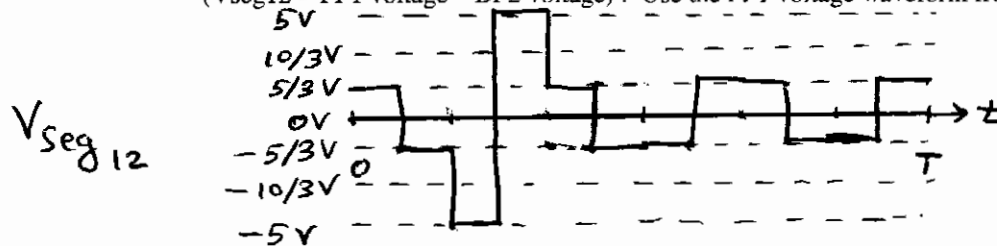
- e. Sketch one frame of a single frontplane signal, FP1, where the segments that pass over BP2, BP4, and BP6 are to be ON, while the remaining three segments are to be OFF.



- f. Sketch one frame of the voltage waveform V_{seg11} , which represents the voltage across the "turned off" segment that lies between $\underline{\text{FP1}}$ and $\underline{\text{BP1}}$. ($V_{\text{seg11}} = \underline{\text{FP1 voltage}} - \underline{\text{BP1 voltage}}$). Use the FP1 voltage waveform from Part e above.



- g. Sketch one frame of the voltage across the "turned on" segment that lies between $\underline{\text{FP1}}$ and $\underline{\text{BP2}}$, V_{seg12} . ($V_{\text{seg12}} = \underline{\text{FP1 voltage}} - \underline{\text{BP2 voltage}}$). Use the FP1 voltage waveform from Part e above



- h. For the case of 1/6 LCD multiplexing, find the RMS value of the V_{seg11} waveform, which corresponds to the waveform of a turned OFF segment, and also the RMS value of the V_{seg12} voltage waveform, which corresponds to a turned ON segment. *Hint: Recall that in the class notes, it was shown (in Figure 7.21) that for the case of 1/4 multiplexing, the RMS voltage across a segment that is ON is $V_{\text{rmson}} = 2.899 \text{ V, rms}$; and the RMS voltage across a segment that is OFF is $V_{\text{rmsoff}} = 1.67 \text{ V, rms}$. Show your calculations in the space below.*

$$(V_{\text{seg11}})_{\text{RMS (OFF seg)}} = \sqrt{\frac{1}{T} \int_0^T V_{\text{seg11}}^2(t) dt} = \sqrt{\frac{1}{T} (25T)} = \frac{5}{3} = 1.67 \text{ V, rms}$$

$$(V_{\text{seg12}})_{\text{RMS (ON seg)}} = \sqrt{\frac{1}{T} \int_0^T V_{\text{seg12}}^2(t) dt} = \sqrt{\frac{1}{T} \left[\frac{5}{6} T \left(\frac{25}{9}\right) + \frac{1}{6} T (25) \right]} = 2.54 \text{ V, rms}$$

RMS value of $V_{\text{seg11}} = \underline{1.67} \text{ V, rms}$

RMS value of $V_{\text{seg2}} = \underline{2.54} \text{ V, rms}$

F. Based upon comparing the results for 1/4 and 1/6 multiplexing,

(1) Which multiplexing method requires fewer connections? 1/6

(2) Which multiplexing method yields higher contrast? (Greater variation in RMS on and off values) 1/4

6 **SCR Crowbar Protection Circuit (5 points)**

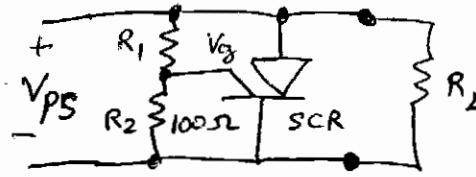
Recalculate the two resistor values in the SCR Crowbar circuit on p. 16-7c of the course notes that will cause the fuse to blow when the power supply voltage rises above 10 V.

SCR turns ON when $V_{PS} \geq 10V$

$$V_g = 0.7V = (10V) \left(\frac{R_2}{R_1 + R_2} \right)$$

Let $R_2 = 100\Omega$

$$\Rightarrow R_1 = 1.328 k\Omega$$

7 **Stepping Motor (5 points)**

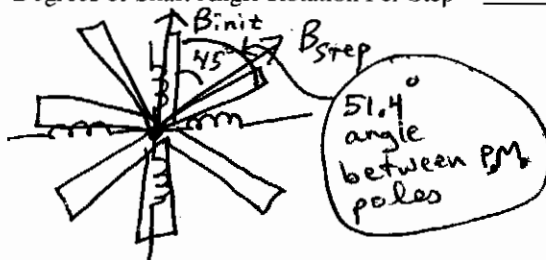
Referring to the stepping motor circuit diagram on page 6-19 of the course notes, imagine that the two bottom rows of 7406 hex inverters are removed, leaving us with just one row of 2N6427 power Darlington transistors. Then imagine that a PIC microcontroller has RB3 connected to the base of the left-most power Darlington, RB2 to the next one, RB1 to the next, and finally RB0 to the right-most power Darlington.

- a. List the sequence of eight 4-bit numbers that would have to be output on the low 4 bits of PORTB in order to make the magnetic field vector developed by the stepping motor spin CW, with 8 steps per revolution (45 degrees per step). Let your first number correspond to the magnetic field pointing directly up.

0001, 0011, 0010, 0110, 0100, 1100, 1000, 1001

- b. Assuming a 7-pole permanent magnet rotor, determine the number of steps per revolution of the shaft using the 8-value sequence of Part A. Do this by drawing, in the space provided below, the 7-pole rotor (showing only the north poles) with one of the 7 poles aligned with the initial B field. Then when the B field steps 45 degrees to its next position, determine which north pole is closest to the new position of the B field, and hence is pulled into alignment. Determine the angle through which it rotates, and determine its direction (CW or CCW).

Degrees of Shaft Angle Rotation Per Step = 6.428° Step Direction = CCW



$$\text{Angle/step} = 51.428^\circ - 45^\circ = 6.428^\circ$$

- c. What is the best name for the four 1N4001 power diodes in this stepping motor circuit? (circle one)
1. transient voltage suppression diodes
 2. turn-on speedup diodes
 3. turn-off speedup diodes
 4. load current limiter diodes
- d. What is the best name for the purpose of the 22-ohm resistor in this stepping motor circuit? (circle one)
1. turn-on speedup resistor
 2. turn-off speedup resistor
 3. load current limiter
 4. voltage transient suppression resistor