

Micros Assignment 2307075

by Rhys Cheedy

Submission date: 13-Jan-2025 05:54PM (UTC+0000)

Submission ID: 248308063

File name: 51889_Rhys_Cheedy_Micros_Assignment_2307075_621295_127647776.docx (244.09K)

Word count: 2214

Character count: 10431

Title : Using and programming the 7 segment display prototype circuit

Module : Micros peripherals and interfaces

Module code : ACDA4004

Lecturer : Gareth Jones

Program : Electronic engineering Beng
2307075

Contents

1. The circuit and its components
 - 1.1 The function of the circuit components
2. How to get the circuit to display hexadecimal characters
 - 3.1 Task number one
 - 3.2 Task number two
 - 3.3 Task number three

Introduction

This assignment will be discussing the programming , assembly and function of the HND/BENG PIC16F84 Assignment Board. The assignment will include images of the circuit in simulation software as well as annotated coding explaining the commands needed to execute the given tasks. Mainly involving the hexadecimal characters, with all of their respective binary calculations displayed and explained.

1. the circuit and its components

in figure 1 the circuit is displayed in the simulation software. There are many components, however, all are necessary for the tasks to be completed and for the microcontroller to operate with the seven segment LED display.

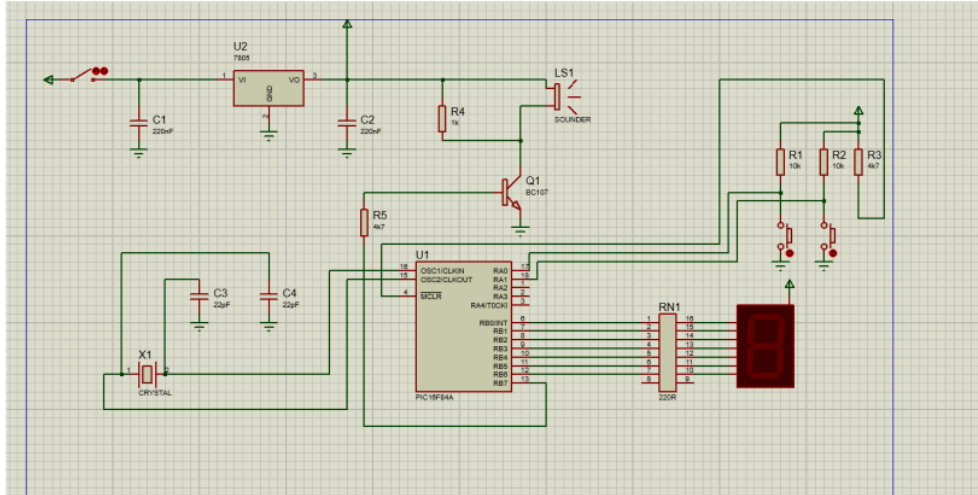


Figure 1 – a schematic of the HND/BENG PIC16F84 Assignment Board

As mentioned, the circuit contains a seven segment LED display. This connects directly to the microcontroller via seven resistors, one port for each segment. The seven resistors connect to the microcontroller the PIC16F84. Coming out of the MCLR (micro controller) port there is a wire connecting the microcontroller to a resistor and power supply. The microcontroller runs off of 2-5 volts. Connected to the same power supply are two more resistors along with two grounded buttons.

On the upper left hand side of the schematic, the subsystem starts with a power supply connected to a switch. This switch is connected to both a grounded capacitor and a 5V fixed power supply regulator, which is also connected to the ground. Following the circuit path further there is another power supply connecting from the top, opposite another grounded capacitor connecting from the bottom. Also connecting from below further down, there is a resistor. This is connected before the sounder on one side and then the connection is completed after the sounder, as displayed in fig.1. Finally, following the sounder, there is a grounded transistor and another resistor before connecting into pin B7 of the microcontroller.

The final subsystem, located in the bottom left hand side of the schematic, is simply two grounded capacitors each connecting to one side of the oscillating crystal and connecting to pins C1 and C2 of the microcontroller.

1.1 The function of the circuit components

In total the circuit contains 14 different components, each serving a different purpose, but all working together to make the circuit work. The following is a list of all the components and their functions.

Power: The power source is represented as a small triangle. This is displayed at the top central left hand side of fig.1. The purpose of the power source is to provide the circuit with power and allow it to function. This circuit contains 3 power sources.

Capacitors: The schematic contains four capacitors, two measured at 220nF, and two measured at 22pF. The purpose of capacitors is to store energy and release the energy when it is no longer powered by the energy of the circuit. An example of this is in the schematic the two capacitors release their charge into the oscillator crystal to power it when needed.

Switch: Located at the very top left of fig.1, the switch acts with an on or off function. This means when the switch is down the circuit is connected to power and operating, however, when the switch is not down and off, no power goes to the circuit.

Resistor: Dotted throughout, there are five resistors in the circuit. Their purpose is to regulate current in a circuit. This can be for both restricting and increasing current in the circuit as you can have step up and step down resistors depending on their implementation and resistance.

Sounder: The sounder in the circuit is a component that is programmable to sound when certain functions happen, depending on what has been written in the assembly code.

7 segment LED display: Like the sounder the 7 segment LED displays function depends on what the assembly code states. Its primary function is the 7 pins that come out of the display each relate to one segment, number pin 7 connects to the middle, and the rest of the numbers go up in a clockwise pattern with pin 1 connecting to the top.

Button: There are 2 buttons in the circuit. Like the switch they connect the circuit when pressed down and disconnect when depressed.

Ground: The ground terminals in the circuit ground the components with a negative connection.

Transistor: Transistors act as current amplifiers in circuits, they act like a binary switch to amplify currents to the appropriate level for the other components to operate correctly. There is only one transistor in the schematic.

Fixed power supply voltage regulator: The purpose of this component is to ensure the voltage running through the circuit is maintained. As this is a 5V power supply regulator then it ensures the voltage is at 5 volts throughout the circuit.

Oscillator crystal: The crystal produces an analog signal into the circuit. This helps the circuit with timing as it resonates at a set frequency.

RN1: Connected directly to the 7 segment LED display, this component is essentially 8 resistors in one line. These operate at 220R and regulate the current in the circuit.

PIC16F84: The microcontroller in the circuit is the centerpiece, exchanging and passing information from one component to another depending on the assembly code. The microchip can be programmed to do many functions and contains 18 bidirectional ports. [1] [2]

2. How to convert hexadecimal numbers into binary and have the circuit display them

In order for the 7 segment LED display to work, as each port is powered individually with the light being either on or off, then the hexadecimal values must be converted into binary for this to work. When converting hexadecimal into binary, the first thing to understand is the hexadecimal values. Hexadecimal operates showing values 1-16. With the numerical representing their true values up until 10. 10 is represented by A, 11 by B and so on until the end of the hexadecimal values where F represents 16. This means the binary value will only

ever be 5 numbers long when converting hexadecimal into binary as the decimal value of 16 is 1 0 0 0 in binary.

Binary numbers are displayed in a format similar to a table where each number represents a different value. For example in table.1 it is showing the 12.

16	8	4	2	1
0	1	1	0	0

Table.1

In the table you can see some values have a 0 below them, and some numbers have a 1. This tells the user what they need to add together to obtain the decimal value. For example $8+4=12$. Therefore the binary value 0 1 1 0 0 is equal to 12 or C in hexadecimal. [3]

3.1 task one

The first task of the assignment was to modify the assembly code so that the seven segment LED display counts from 0 to F in hexadecimal numbers in 0.2 second increments. See figure 2 for a screenshot of the modified code. In this code there are a few elements of the code that need adjusting so that the pick can execute the code in the correct manor.

```
4
5 #include p16f84a.inc           ; Include register definition file
6
7 portB equ 0x06                ;port B address
8 trisB equ 0x06                ;trisB address in register bank 1
9
10 status equ 0x03               ;status register
11 R0 equ 5                      ;bit 5 of status register
12 R1 equ 6                      ;bit 6 of status registers
13
14 .
```

figure.2

In figure 2 it can be seen that there are a function called portB. This addresses all of the B ports on the microcontroller. This is because the B ports from the microcontroller are connected to the 7 segment LED display.

In figure 2 there are 3 functions called. The first function is the MOVLW function. This moves the literal value to the working register. Each value is represented off to the right of the command. These numbers are telling the 7 segment LED display which lights need to be on and off in order to count up in hexadecimal values.

The MOVWF function moves the working register value into the f register which is portB. This is the port the 7 segment LED is input into. This will transfer the command and light up the lights. Finally, the call pause command at the end is to hold the display in order to be read and prevent bouncing or flickering.

See figures.3 and .5 for flowchart and modified ASM code.

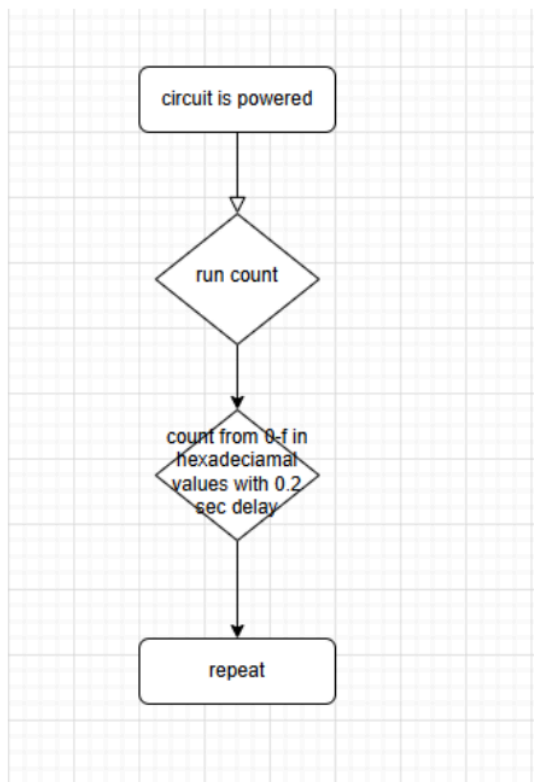


fig.5. flowchart diagram of the code

```

55 get_count
56 movf count, W ; move count to W
57
58 movlw b'11000000' ;0
59 movwf portB ; move W to port B to update the display
60 call pause
61
62 movlw b'11001111' ;1
63 movwf portB ; move W to port B to update the display
64 call pause
65
66 movlw b'10100100' ;2
67 movwf portB ; move W to port B to update the display
68 call pause
69
70 movlw b'10110000' ;3
71 movwf portB ; move W to port B to update the display
72 call pause
73
74 movlw b'10011001' ;4
75 movwf portB ; move W to port B to update the display
76 call pause
77
78 movlw b'10010010' ;5
79 movwf portB ; move W to port B to update the display
80 call pause
81
82 movlw b'10000010' ;6
83 movwf portB ; move W to port B to update the display
84 call pause
85
86 movlw b'11111000' ;7
87 movwf portB ; move W to port B to update the display
88 call pause
89
90 movlw b'10000000' ;8
91 movwf portB ; move W to port B to update the display
92 call pause
93
94 movlw b'10011000' ;9
95 movwf portB ; move W to port B to update the display
96 call pause
97
98 movlw b'10001000' ;a
99 movwf portB ; move W to port B to update the display
.00 call pause
.01
.02 movlw b'10000011' ;b
.03 movwf portB ; move W to port B to update the display
.04 call pause
.05
.06 movlw b'11000110' ;c
.07 movwf portB ; move W to port B to update the display
.08 call pause
.09
.10 movlw b'10100001' ;d
.11 movwf portB ; move W to port B to update the display
.12 call pause

```

figure.3 the modified count.asm code.

3.2 task two

Task 2 of the assignment was to modify the code to allow the 7 segment LED to display the 'l' character when one button was pressed and display the 'r' character when the other button was pressed. For this task, compared to task one, the setup was different as new

loop functions had to be implemented. This was because there were 2 inputs and the PIC needed to be aware of when the input was changing. For this separate 'r' and 'l' functions were made.

```

3 main_loop:
4
5     BTFSS portA,0      ; if the button that is connected to pin A0 pressed then complete next command
6     GOTO l_function   ; continue in loop 'l_function'
7
8     BTFSS portA,1      ; if the button that is connected to pin A1 pressed then complete next command
9     GOTO r_function   ; continue in loop 'r_function'
10
11    movlw b'1111111'    ; activating connections and making the LED display blank
12    movwf portB         ; move W to port B to update the display
13    call pause          ; delay 200ms by subroutine, this prevents bouncing
14
15    GOTO main_loop      ; continue in main loop (main_loop)
16
17 l_function:
18    movlw b'11000111'   ; display an 'L' on the 7 segment LED display
19    movwf portB         ; move W to port B to update the display
20    call pause          ; delay 200ms by subroutine, this prevents bouncing
21    GOTO main_loop      ; continue in main loop (main_loop)
22
23 r_function:
24    movlw b'10101111'   ; display an 'r' on the 7 segment LED display
25    movwf portB         ; move W to port B to update the display
26    call pause          ; delay 200ms by subroutine, this prevents bouncing
27    GOTO main_loop      ; continue in main loop (main_loop)

```

figure.4. the

introduction of main loop.

The first command in main_loop is BTFSS, this means bit test f skip if selected. This essentially means to select f if the input is A0, one of the button inputs. However, if the button is already selected then to skip and just execute the line below. The following command GOTO is directing the code to the function to be executed if port A0 had an input value of 1.

The next command is similar to the first however it contains a different port number, this is because it is for the other button. This will send the PIC to the 'r' function to display 'r' on the seven segment LED.

The next line of MOVLW meaning move the literal value to the w register, is telling the 7 segment LED display to show a blank display, while waiting for the buttons to be pressed. The blank display should light up every segment to show the circuit is on and awaiting user input from one of the buttons. To prevent bouncing and flickering of the LEDs there is a delay on the lights before going back to the main loop to await input.

Next is the 'l' and 'r' functions. Firstly there is a MOVLW function to load the w register with the literal value. Following this there is a function to move the value in the w register to the f register to display. Once this has happened there is a pause to mitigate bouncing. After the command is executed the code goes back to the main loop to await further input.

See figure.6. for a flowchart representation of the ASM code.

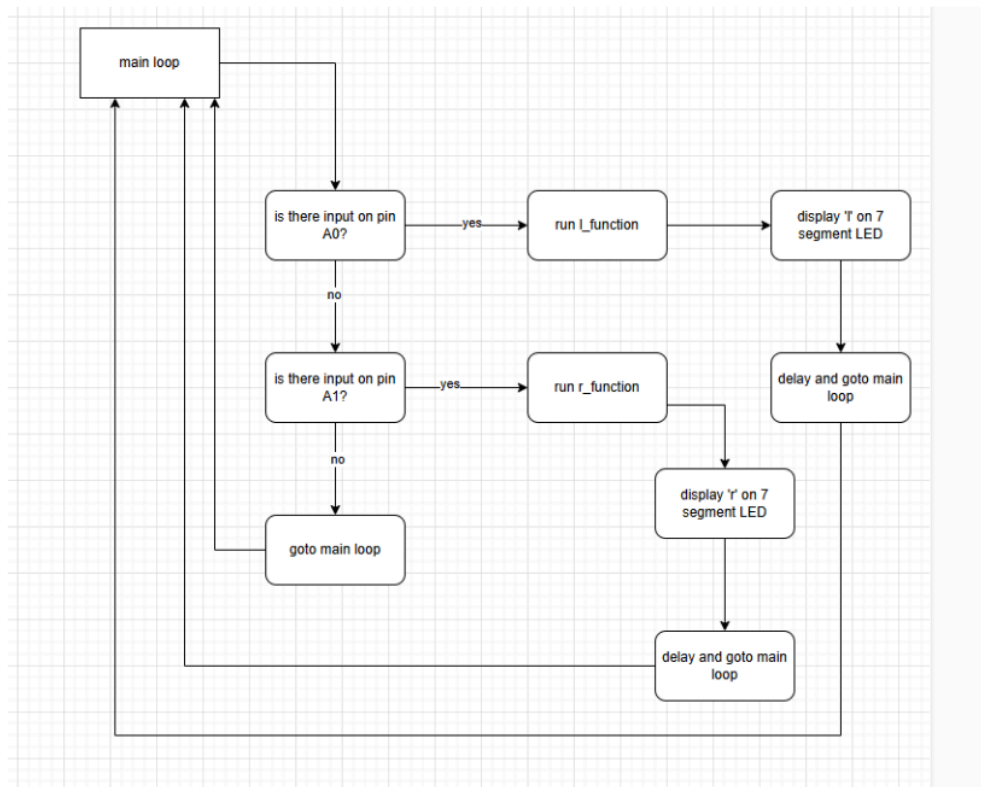


figure.6. a flowchart representation of the task 2 modified asm code

3.3 task three

The objective of task three was to modify the code to create a system that would light up all seven segments when one button was pressed and have the lights maintained on until the other button was pressed, creating a latching effect. For this task a new feature had to be introduced which is 'buttonstate' this is a feature where the PIC can update itself with the state of which button has last been pressed, creating a latching effect with no latching buttons. There are two button state functions displayed in the modified .asm code. One for turning the LED segments on and one for turning them off. These are found in the main loop where the PIC checks the inputs A0 and A1 to see if a button has a state of 1 (on) or 0 (off). Depending on these button states the PIC will then be directed to the function that it is required to execute.

This can be seen in fig.7. which is a flow chart of the code for the system to execute.

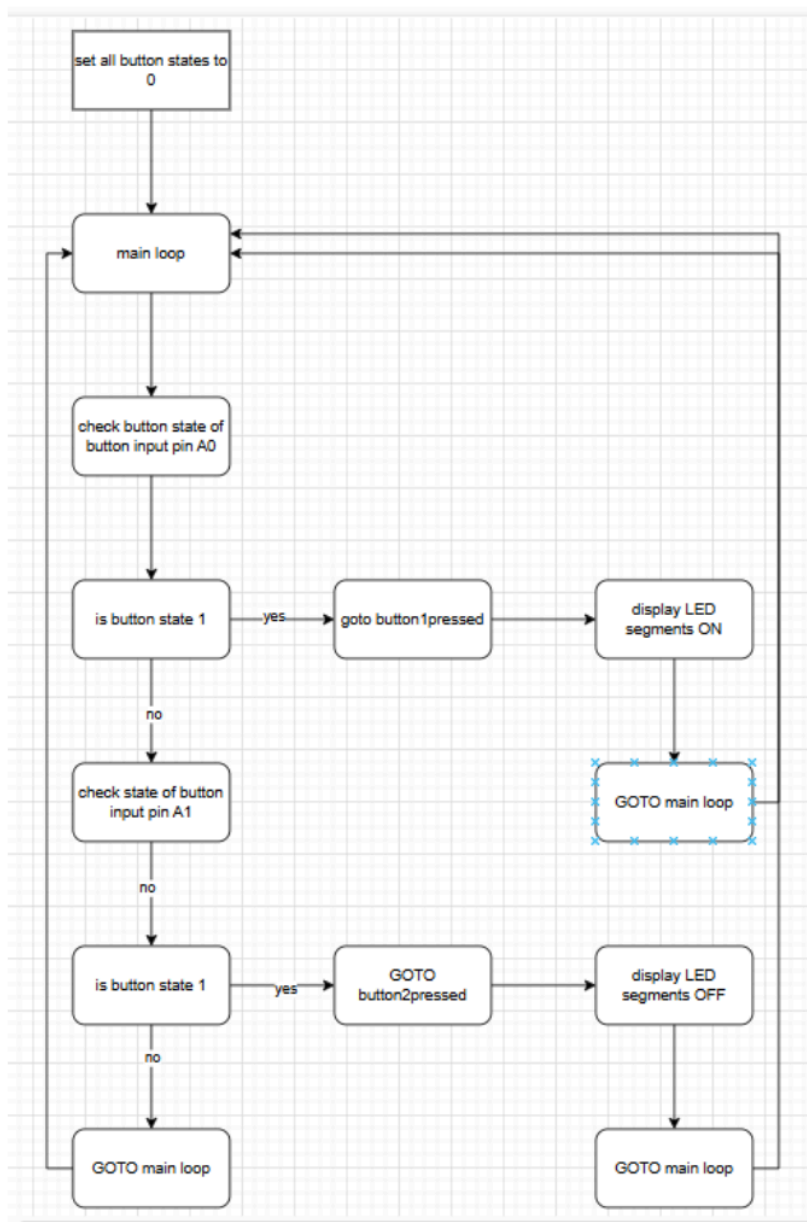


fig.7. flow chart

of task 3 execution

As seen in figure 7 the flowchart goes through each button individually and checks the state of their output to see if the LED segments need to be updated. This is different to the previous task as the button states create the latch as if the states remain the same a delay is put on in order to keep the function running while the states are continuously checked. The main loop function is once again used for the code to revert back to in order to repeat the steps.

Seen in fig.8. the modified .ASM code contains 3 sections: main loop, button1pressed and button2 pressed. Before the main loop is the setup. This is vital as it tells the PIC that the pins A0 and A1 will be the inputs for this coding task. This means that the buttons will be registered with the PIC when pressed and the task will be executed correctly. As seen at the bottom of the main loop, if no buttons are pressed it shall go back to the top and start the main loop process again.

```

1
2
3 PGM code
4 Start
5   bcf status, RP0    ; Clear bank selection bits in the status reg
6   bcf status, RP1
7
8   ; Set up port B for output (LED segments)
9   bsf status, RP0
10  movlw 0x00          ; Set all bits of portB to output (0x00)
11  movwf trisb
12  bcf status, RP0    ; reselect bank 0
13
14  ; Set up port A for input (buttons)
15  bsf status, RP0
16  movlw 0x03          ; Set RA0 and RA1 as inputs (binary 11)
17  movwf trisa
18  bcf status, RP0    ; reselect bank 0
19
20  clrf portb          ; Clear all LED segments initially
21  clrf buttonState    ; Initialize button state to 0 (off)
22
23 MainLoop
24  ; Read button states (RA0 and RA1)
25  btfscc portA, 0     ; Check if button 1 (RA0) is pressed
26  goto Button1Pressed
27
28  btfscc portA, 1     ; Check if button 2 (RA1) is pressed
29  goto Button2Pressed
30
31  ; No buttons pressed, continue to Loop
32  goto MainLoop
33
34 Button1Pressed
35  ; Button 1 pressed: Turn on all LED segments
36  bsf buttonState, 0  ; Set button state to 1 (on)
37  movlw b'11111111'  ; All segments ON (8 bits for 7-segment display)
38  movwf portb
39  goto MainLoop
40
41 Button2Pressed
42  ; Button 2 pressed: Turn off all LED segments
43  bcf buttonState, 0  ; Set button state to 0 (off)
44  clrf portb          ; Clear all LED segments
45  goto MainLoop
46
47 ;-----

```

References

- [1] O. a. M. N. Jakobsson, Student papers in computer architecture, 2006.
- [2] M. technology, "PIC16F84 Datasheet," microchip, [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf>. [Accessed 13 1 2025].

[3] C. Springer, "Fundamentals of digital electronics," *Lecture notes in electrical engineering*, vol. 623, 2020.

Micros Assignment 2307075

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

38/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

GRADING FORM: GJ_2019

ENGAGEMENT WITH LITERATURE

 only three sources cited - this is too limited - need FAR more and they should cover ALL areas and ALL components etc.

25

KNOWLEDGE AND UNDERSTANDING

 The K&U is weak, with a number of errors. Most of the explanation is just stating what the components ARE or how they are connected rather than explaining exactly what they DO in the circuit - lack of referencing and external sources is mostly to blame here.

35

COGNITIVE AND INTELLECTUAL

 Only half the tasks attempted and these use a very conventional approach, demonstrating little flair or creative thinking.

30

PRACTICAL APPLICATION

 The circuit is simulated well, but only half the tasks attempted

50

TRANSFERABLE SKILLS



Report layout is fair, but limited in content, could have used far more images in the explanation etc. More detailed TOC needed and a nicer title page wouldn't have gone amiss either!

50