

ACDA4004 Student ID2204725

by Gabriel Pan

Submission date: 08-Jan-2025 11:02AM (UTC+0000)

Submission ID: 247607565

File name: 52080_Gabriel_Pan_ACDA4004_Student_ID2204725_621295_661476613.docx (2.04M)

Word count: 4510

Character count: 23139

Using and Programming the 7 Segment Display Prototype Circuit

ACDA4004: Micros, Peripherals and Interfaces

Student ID: 2204725

Word count: 4510

Contents:

Title.....	1
Contents.....	2
Introduction.....	3-4
Section 1.....	5-13
Section 2.....	14-17
Section 3.....	18-29
References.....	30

Introduction:

This assignment will identify and explain all the components of a circuit, show how and why the circuit works and demonstrate competency in C-programming by showing a circuit which has been modified by inputting different codes into a microcontroller which have been written with different desired outcomes/results in mind. The circuit being used uses a 7-segment display and a 'sounder', which is a virtual component resembling and acting as a speaker; these are used to show different outputs depending on the code written. The circuit is commanded by a PIC16F84 microcontroller this manages all inputs and outputs through the code written. The circuit also contains many other useful components such as resistors, capacitors and a crystal oscillator. This is shown below in fig. 1 and accompanying key.

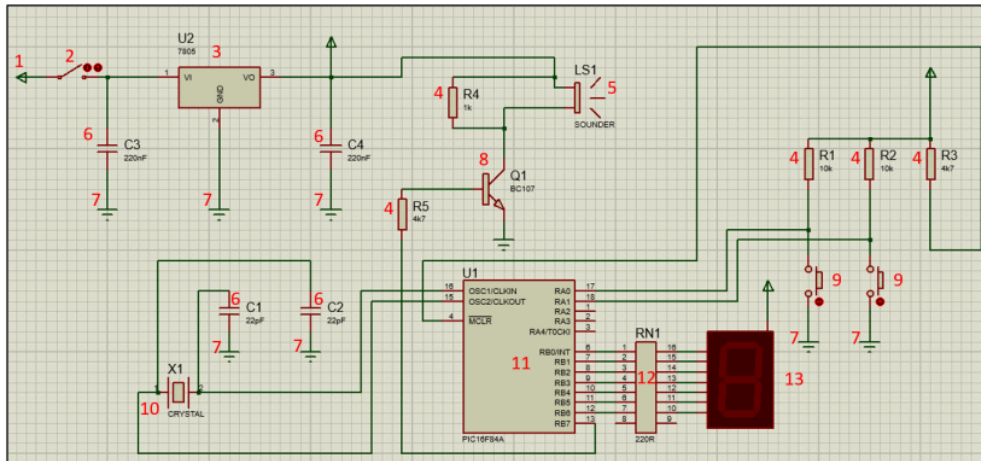


Fig. 1

Key:

1 – Power input	2 – Switch	3 – Voltage Regulator
4 – Resistor	5 – Sounder/Speaker	6 – Capacitor
7 – Ground	8 – Transistor	9 – Button
10 – Crystal Oscillator	11 – PIC Microcontroller	12 – 8-Resistor Chip
13 – 7-Segment Display		

The code which was provided is shown below in fig. 2 and 3; this code provides the basic framework to work from with annotations along each input to allows for greater understanding and confidence with modifying this code for the desired outputs required.

```

1  #include p16f84a.inc          ; Include register definition file
2
3  portB    equ 0x06             ;port B address
4  trisB    equ 0x06             ;trisB address in register bank 1
5
6  status   equ 0x03             ;status register
7  RP0      equ 5                ;bit 5 of status register
8  RP1      equ 6                ;bit 6 of status register
9
10 count    equ 0x0c             ;counter 0 to 255
11 nCount    equ 0x0d            ;pause routine inner loop counter
12 mCount    equ 0x0e            ;pause routine outer loop counter
13
14          ; Reset Vector
15 RST      code 0x00
16          goto Start
17
18 PGM      code
19 Start
20          bcf     status,RP0     ; clear bank selection bits in the status reg
21          bcf     status,RP1
22
23          bsf     status,RP0     ; select bank 1
24          movlw   0x00           ; Load w with 0x00
25          movwf   trisB         ; Load it into trisb to make portb all outputs
26          bcf     status,RP0     ; reselect bank 0
27
28          clrf    portB         ; clear all lines low
29          clrf    count         ; initialize counter to 0
30
31 get_count
32          movf    count, w       ; move count to W
33          movwf   portB         ; move W to port B to update the display
34          addlw   1              ; add +1 to W
35          movwf   count         ; save W in count
36
37          call    pause         ; delay subroutine by 200ms
38          call    pause
39          call    pause
40          call    pause
41          call    pause         ; five pause executions equals ~ 1 second
42
43          goto    get_count      ; repeat forever
44
45 pause
46          movlw   0xff          ; set w = 255 decimal
47          movwf   mCount        ; mCount = w: outer loop counter

```

Fig. 2

```

48 loadN
49          movlw   0xff          ; set w = 255 decimal
50          movwf   nCount        ; nCount = w: inner loop counter
51
52 decN
53          decfsz  nCount, f      ; nCount--
54          goto    decN          ; if nCount != 0 then repeat nCount--
55
56          decfsz  mCount, f      ; else decrement Count
57          goto    loadN         ; if mCount != 0 then
58                                   ; reload nCount to 255 and decrement
59          return                ; else exit subroutine

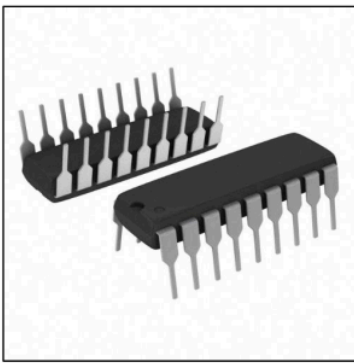
```

Fig. 3

Section 1

This section will explain and elaborate on all the components used in the 7-segment display circuit to greater understand the circuit and what it could be capable of.

PIC16F84A Microcontroller:



The PIC16F84A is an 8-bit microcontroller which is from the 'PIC' family of microcontrollers. The PIC16F84A was made as an updated version of the PIC16C84, the most significant update being an increase of RAM (Random Access Memory) available in the PIC16F84A; meaning it can more effectively execute and deliver programs at higher speeds and at increase complexity. The microcontroller proved to be ruthlessly effective with its utilisation of a serial programming algorithm and EEPROM (Electrically Erasable Programmable Read-Only Memory).

[1] Fig. 4

In this circuit, however, the PIC16F84A is acting as the brain of the circuit and has complete control of all operations and outputs occurring in the circuit simulation. As you may notice in the fig. 5 all of the pins have identifiers next to them, this is to help show their possible functions and designate them in later programming. Not all pins have to be used in any given circuit, for example pins RA2, RA3 and RA4, this is simply because there is no need in this circuit for any additional inputs or outputs for the eventual outputs required.

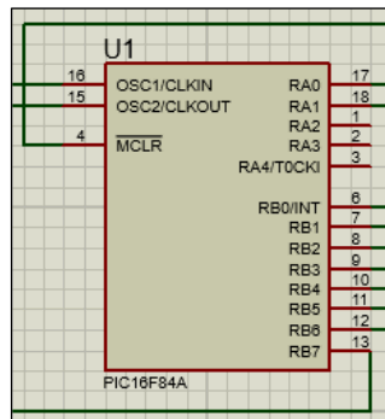


Fig. 5

The pins in the 'A' port, this is identified by the second letter in the designations, are used in this circuit as inputs from two different buttons with each having the ability, if the appropriate button is pressed, to 'say' 'yes' or 'no'.

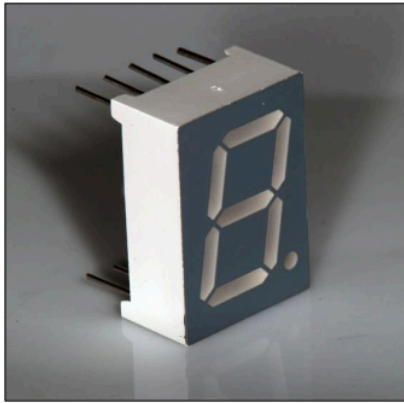
The pins in the 'B' port are utilised as outputs with RB0, RB1, RB2, RB3, RB4, RB5 and RB6 being connected to the 7-segment display through a resistor chip; these outputs directly effect the display as each output pin either turns off or on each segment of the display making it possible, through the code, to display any possible shape on the display of which the display allows.

The pin 'RB7' is connected to a system with multiple components which results in the speaker activating when the output is sent out this pin by the code. There are three other pins which, it may be noticed, don't have a "port designation", this is due to the pins being needed to perform the same function regardless of the code which is inputted.

The pins 'OSC1/CLKIN' and 'OSC2/CLKOUT' are used to keep track of time by connecting them to a crystal oscillator or another type of clock, this helps the code massively, for example, say the PIC is programmed to output through a pin for one second the output on another, without these pins being used the PIC would keep outputting on the first pin because it has no reference of time to say when it stops that function.

The final pin, MCLR (Master Clear), is used to reset the program to the "top of the program execution or memory location 0x00"[2]. This resets the device to the first command programmed into it, this is useful so that the PIC cant get 'stuck' in a loop or glitch in the code inputted.

7-Segment display:



[3] Fig. 6

In this circuit a 7-segment display is used as an output from the PIC controller, this is useful because it displays a specific and desired outcome which is instantly checked on observation of the simulation running, acting as a verifier and could be useful if the simulation circuit is constructed.

The component operates based on 8 pins acting as inputs; 7 pins of which are directly connected to each segment of the display with an off or an on function, the 8th pin powers the component with the necessary amount to light up any combination of segments to achieve the desired display.

Depending on the inputs on the first 7 pins, this component can display any possible light configuration allowed by the positioning of the light segments. The pin layout in physical construction can be observed in the fig. above and the appearance of the component in the simulation can be seen on the right in fig. 7.

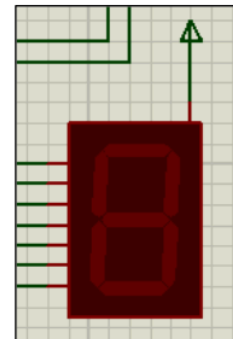


Fig. 7

Crystal Oscillator:

A crystal oscillator is an essential component of any semi-intelligent circuit or at least any circuit which requires a microcontroller; this is because a crystal oscillator acts as an internal clock to keep time on any time-based operation which is being performed by the circuit.

This component achieves this by utilising the frequency given by applying a voltage source to a fragment of quartz crystal which then produces a phenomenon known as the 'piezo-electric effect' in which the crystal oscillates or 'vibrates' at a high frequency which, largely, remains constant [5]. Dependent on how and to what shape the crystal has been cut directly determines the frequency at which the crystal oscillates at. For example, in the fig. 8 to the right the size of the internal crystal means it oscillates at 16 MHz as noted on the external print.



[4] Fig. 8

In the circuit, however, the crystal is observed as the symbol shown in fig. 9 on the left. In the simulation the oscillator is directly connected to the PIC controller at pins 'OSC1/CLKIN' and 'OSC2/CLKOUT' ensuring that the code is followed to any required timings demanded by the code in the PIC controller. In the

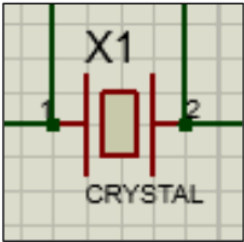


Fig. 9

example of the simulation, the crystal is set to oscillate at the frequency of 1MHz, as seen in fig. on the right.

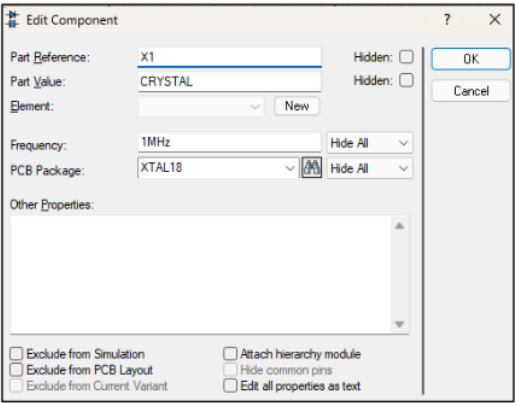


Fig. 10

Sounder/Speaker:

Present in this circuit, also, is a 'sounder', which in the simulation represents a widely used component known as a 'piezo buzzer' which is a type of "audio signalling device" which utilises piezoelectricity as a source of power. This works by the component containing 'piezoelectric element' which is driven by an oscillating input which then creates the sound



[7] Fig. 11

of that oscillation[6]. The type of component can be found in many different sizes and designs which can make different types of sounds, also. This can be seen in fig. 10 on the left.

In the circuit, the 'sounder' is directly powered by an external power source and then has an input from one of the pins on the PIC controller to initiate its function.

The simulated component is shown as the symbol in fig. on the right.

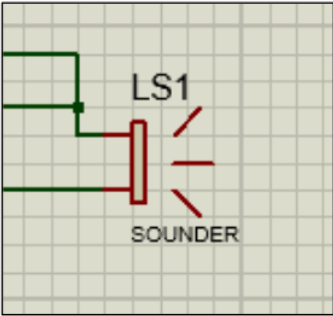
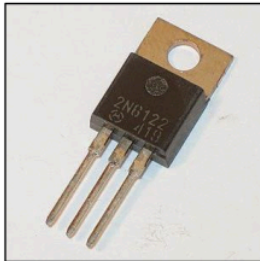


Fig. 12

Transistor:

A transistor is present in nearly all circuits as an essential component. Transistors operate as a form of switch that can block the flow of electricity along a wire, to allow the flow, a high enough voltage is sent to the transistor. A physical example can be seen on the right in fig. 13.

In the simulation circuit, however, this component is used in one instance in which it blocks flow to a ground from a sounder which has a power source connected, and the 'base' pin is connected directly to the PIC controller. This means that



[8] Fig. 13

when this pin is 'on', through the inputted code, the sounder/speaker will activate and create a sound to confirm this. An example of the simulated transistor symbol from the circuit can be seen in fig. 14 on the left.

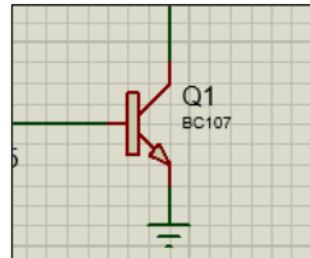
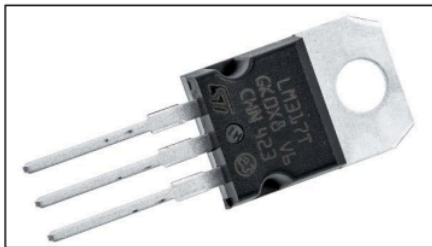


Fig. 14

Voltage Regulator:

In short, a voltage regulator regulates voltage; however, it does this by maintaining a constant voltage from one 'Vin' pin to one 'Vout' pin and dissipates any remaining voltage, when the designated voltage is reached, out the ground pin. Many different sizes of voltage regulator exist to regulate any voltage required by the circuit. A physical voltage regulator is shown in fig. 15. In the simulated circuit the voltage regulator is positioned to regulate the voltage powering the sounder/speaker; the simulated representation is depicted in fig. 16.



[9] Fig. 15

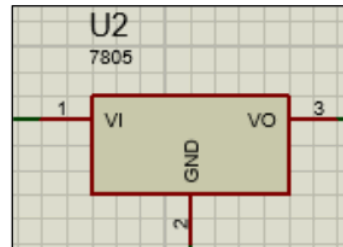
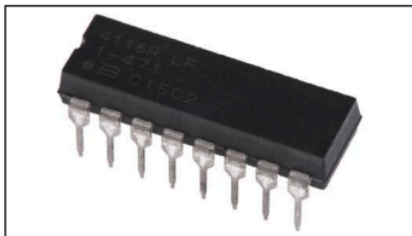


Fig. 16

8-Resistor Chip:

Resistor chips are used to make manufacturing, constructing and fault-finding circuits easier. The component, basically, is just as effective as, in this case, 8 resistors which are parallel to one another. In this simulation this chip is used as a resistor for all the connections from the PIC controller to the 7-segment display. Examples of both the physical component and the simulated chip in the circuit are found below in fig. 18 and fig. 17.



[10] Fig. 17

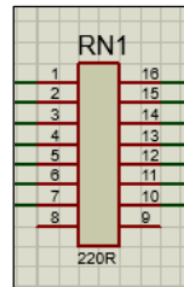


Fig. 18

Resistor:

A resistor is one of the most basic of all components. A resistor operates by resisting the current the wire it is in, all resistors have a measured amount of resistance, just as every material, calculated in Ohms. The component is constructed by picking the appropriate material and selecting the amount of material with the correct amount of ohms, then two pins are connected to each side. The physical component is often marked by an arrangement of ring of different colours, this determine the amount of resistance between the two pins it has. Examples of both the physical component and the simulated component in the circuit are found below in fig. 20 and fig. 19.



[11] Fig. 19

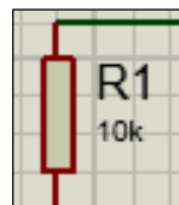
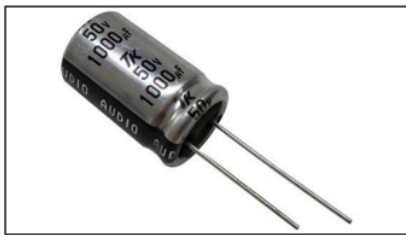


Fig. 20

Capacitor:

Capacitors store and release power, they store the power when a power source is connected to one pin of the capacitor and no components are drawing power on the other pin, this means that when no power is needed the capacitor will store it automatically until it is. However, when it is, the capacitor will discharge the power to the components that are 'needing' it to operate. This is very useful in case of intentional or unintentional power outages to the components in the circuit. This means that capacitors are widely used in almost every type of circuit and can be used in many different ways given these properties of this component. For example in the simulated circuit two capacitors are connected to both of the pins on the crystal oscillator, and therefore will prevent this essential component from losing power at any moment. Examples of both the physical component and the simulated component in the circuit are found below in fig. 21 and fig. 22.



[12] Fig. 21



Fig. 22

Button:

A button is a very simple component which, simply, is placed in a wire connection and, when not pressed, leaves the connection broken so no electricity can run through the wire and when pressed it connects the break and allows the current along the wire to flow. In the simulation circuit this component is used as an input to influence what task the PIC controller is performing, whilst the simulation is running. Examples of both the physical component and the simulated component in the circuit are found below in fig. 23 and fig. 24.



[13] Fig. 23



Fig. 24

Switch:

The switch as an electrical component operates the exact same way as a button by having the ability to break a physical connection of a wire, apart from in one affect which is that the switch has no default setting of operation when not activated whereas when the button is not activated it will return to its default setting where the connection of the wire is broken. In other words, you cycle between the switch being in a constant state of 'on' or 'off' each time it is activated. This component is also useful in the same ways as a button by having the ability to influence the operations of a circuit whilst it is actively operating. In the simulated circuit it is connected to a power supply on one pin and on the other connected to the sounder/speaker subsystem to provide additional power to it when activated. Examples of both the physical component and the simulated component in the circuit are found below in fig. 25 and fig. 26



[14] Fig. 25



Fig. 26

Power Input:

Power inputs in the simulated circuit are used to power it and all of its operations, the simulated components are very convenient, however in a physical circuit you would need to implant a battery or a source of power from a larger system such as a house hold mains plug. Examples of both the physical component and the simulated component in the circuit are found below in fig. 27 and fig. 28.



[15] Fig. 27

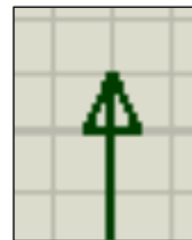


Fig. 28

Ground:

A way for a circuit to connect to a ground source is very useful and simplifies the construction of many circuits. This is useful because a ground source dissipates any current flowing into it and therefore can get rid of any unnecessary electrical energy from the circuit. For example, the voltage regulator component explained earlier in this report is connected to a ground source because that is where it is designed to flow any electrical energy that is ignored and allowed to flow into the circuit would be over the amount of voltage the voltage regulator is being used to maintain. For physical circuits the most basic way for a ground source to be present is to have the wire you need grounded planted in the actual ground. Examples of both the physical example and the simulated component connected to the voltage regulator are found below in fig. 29 and fig. 30.



[16] Fig. 29

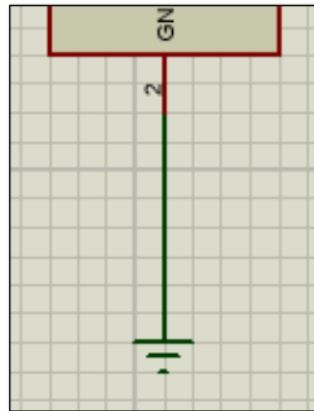


Fig. 30

Section 2:

The properties of this simulated circuit, with all of the components laid out in the previous section, mean that it is able to display any output on the 7-segment display. This is possible because the 7-segment display is connected directly in to the PIC controller to 7 output pins which can be influenced to be 'on' or 'off' by the inputted code. In this section I will detail what is required for the code to influence the simulated circuit to, when activated, display all of the single characters/digits of the hexadecimal number system (0 – F).

To edit the code to be able to achieve this, I will need to first define 'port B' and let it have the ability, when specified to output on these 7 pins, in which the port is the housing of all the pins which control the 7-segment display. This, however, has already been completed in the provided base code, as seen below in fig. 31.

```
15 portB    equ 0x06      ;port B address
16 trisb    equ 0x06      ;trisB address in register bank 1
17
```

Fig. 31

To explain:

- Line 15 is written to simply name the address of the pins, in this code it is specified that 'portB', which is the row of pins connected to the 7-segment display, 'equ', which translates to =, '0x06', this specifies that the 'address' of 'portB' is in the 6th byte register of the PIC controller's command list. This line is essential to use the 7-segment display, without this the PIC controller would not be able to register/understand the specification 'portB'.
- Line 16 is written to give 'portB' a place in the control register, by specifying 'trisb' with the same address as 'portB', which means that when you specify a pin in 'portB', in binary form, as a 1, it inputs data and as a 0, it outputs data.

The next modification I would have to make for the code to produce this is to produce the code which specifies the pins that needs to be output in 'portB' for the display to show the characters/digits required, and in the required order. See below in fig. 32 for an example and explanation.

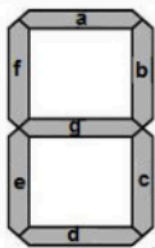
54	<code>movf</code>	<code>count, w</code>	<code>; move count to W</code>
55			
56	<code>movlw</code>	<code>b'11000000'</code>	<code>; move the following binary data to the W register</code>
57	<code>movwf</code>	<code>portB</code>	<code>; move W to port B to update the display</code>
58	<code>call</code>	<code>pause</code>	<code>; delay subroutine by 200ms</code>

Fig. 32

To explain:

- Line 54 moves 'count' to the w register which allows the code in 'count' to output as a base operation
- Line 56 moves the accompanying binary data to the w register with the function 'movlw'. The following data is 'b'11000000'', where 'b' specifies that the following data is in the number system of binary.
- Line 57 moves the w register to 'portB', this allows the binary written in the last line of code to be activated on 'portB', which is the port that controls the 7-segement display and therefor will display on the component when the simulation is running.
- Line 58 is the 'call pause' function which delays the next operation in the code from being ran by 200ms or 0.2 seconds. This allows the display to be observed before the code continues with the next character or digit in the hexadecimal system.
- The binary output in fig. above shows the digit of 0 on the display when it is ran into the 7-segment display. The theory of why this is so, however, is due to each digit of the byte of binary representing each pin on 'portB' being outputting or inputting. Because of the 'trisb' function being active, 0 is a pin outputting and 1 is a pin inputting, because the display has no possible output to the connections to the PIC controller, the pins that are inputting are, essentially, 'off'.

The byte of binary '11000000' displays a zero in this circuit because of how the display is set to operate, use fig. 33 and fig. 34. below as supplementary information. In fig. 33, it labels each segment as a character of the alphabet, following this the byte of binary, due to how the circuit was constructed, as said before each digit represents each pin and therefor each segment also. So, the byte can be simplified to outline each segment of the display as '1gfedcba'; where the first digit is not applicable to this operation because, as seen in fig., the pin is not connected to the 7-segment display, but the sounder/speaker subsystem. Alternatively, the byte of binary could be represented as pin allocations as 'RB7, RB6, RB5, RB4, RB3, RB2, RB1, RB0', see fig. below for clarification.



[17] Fig. 33

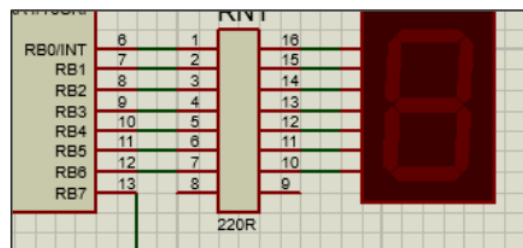


Fig. 34

Now, by observing the previous fig. 32., all is needed is to repeat this code with different bytes of binary data being moved to the w register that display the remaining 15 characters/digits. This section of code is all in a loop of code which restarts from the top once it is completed so that the display will cycle through all of these operations then start again, once the simulation is running. See code for this below in fig. 35 and fig. 36.

```

main.asm
54      movf    count, w      ; move count to W
55
56      movlw   b'11000000'   ; move the following binary data to the W register
57      movwf   portB         ; move W to port B to update the display
58      call    pause        ; delay subroutine by 200ms
59
60      movlw   b'11001111'   ; move the following binary data to the W register
61      movwf   portB         ; move W to port B to update the display
62      call    pause        ; delay 200ms by subroutine
63
64      movlw   b'10100100'   ; move the following binary data to the W register
65      movwf   portB         ; move W to port B to update the display
66      call    pause        ; delay 200ms by subroutine
67
68      movlw   b'10110000'   ; move the following binary data to the W register
69      movwf   portB         ; move W to port B to update the display
70      call    pause        ; delay 200ms by subroutine
71
72      movlw   b'10011001'   ; move the following binary data to the W register
73      movwf   portB         ; move W to port B to update the display
74      call    pause        ; delay 200ms by subroutine
75
76      movlw   b'10010010'   ; move the following binary data to the W register
77      movwf   portB         ; move W to port B to update the display
78      call    pause        ; delay 200ms by subroutine
79
80      movlw   b'10000010'   ; move the following binary data to the W register
81      movwf   portB         ; move W to port B to update the display
82      call    pause        ; delay 200ms by subroutine
83
84      movlw   b'11111000'   ; move the following binary data to the W register
85      movwf   portB         ; move W to port B to update the display
86      call    pause        ; delay 200ms by subroutine
87
88      movlw   b'10000000'   ; move the following binary data to the W register
89      movwf   portB         ; move W to port B to update the display
90      call    pause        ; delay 200ms by subroutine
91
92      movlw   b'10011000'   ; move the following binary data to the W register
93      movwf   portB         ; move W to port B to update the display
94      call    pause        ; delay 200ms by subroutine
95
96      movlw   b'10001000'   ; move the following binary data to the W register
97      movwf   portB         ; move W to port B to update the display
98      call    pause        ; delay 200ms by subroutine
99
100     movlw   b'10000011'   ; move the following binary data to the W register
101
102     movwf   portB         ; move W to port B to update the display
103     call    pause        ; delay 200ms by subroutine
104
105     movlw   b'11000110'   ; move the following binary data to the W register
106     movwf   portB         ; move W to port B to update the display
107     call    pause        ; delay 200ms by subroutine
108
109     movlw   b'10100001'   ; move the following binary data to the W register
110     movwf   portB         ; move W to port B to update the display
111     call    pause        ; delay 200ms by subroutine
112
113     movlw   b'10000110'   ; move the following binary data to the W register
114     movwf   portB         ; move W to port B to update the display
115     call    pause        ; delay 200ms by subroutine
116
117     movlw   b'10001110'   ; move the following binary data to the W register
118     movwf   portB         ; move W to port B to update the display
119     call    pause        ; delay 200ms by subroutine

```

Fig. 35 and 36

Section 3:

This section will explain and prove how each of the coding tasks have been completed from the assignment brief.

Task 1:

This task is to edit the coding of the simulated circuit, so that when running the 7-segment display will cycle through all the single digits/characters of the hexadecimal number system (0 to F) in the correct order of progression from the number system, then start from the first digit/character when the program has cycled all of the values. It is also specified that there be a time delay between each digit/character of 200ms or 0.2 seconds. In the previous section, this report has outlined precisely what is required and how this code completes this task, therefore this text will not elaborate further on this subject.

Please see below in figs. 37, 38, 39 for the written code, and accompanying annotations.

```
main.asm
1  #include p16f84a.inc      ; Include register definition file
2
3  portB    equ 0x06         ; portB address
4  trisB    equ 0x06         ; trisB address in register bank 1
5
6  status   equ 0x03         ; status register
7  RP0      equ 5           ; bit 5 of status register
8  RP1      equ 6           ; bit 6 of status register
9
10
11 count    equ 0x0c         ; counter 0 to 255
12 nCount    equ 0x0d        ; pause routine inner loop counter
13 mCount    equ 0x0e        ; pause routine outer loop counter
14
15
16 RST      code 0x00        ; Reset Vector
17      goto Start
18
19
20 PGM      code
21
22 Start
23      bcf    status,RP0     ; clear bank selection bits in the status reg
24      bcf    status,RP1     ; clear bank selection bits in the status reg
25
26      bsf    status,RP0     ; select bank 1
27      movlw  0x00           ; Load w with 0x00
28      movwf  trisB          ; Load it into trisb to make portb all outputs
29      bcf    status,RP0     ; reselect bank 0
30
31      clrf   portB          ; clear all lines low
32      clrf   count          ; initialize counter to 0
33
34 get_count
35      movf   count, w       ; move count to W
36
37      movlw  b'11000000'    ; move the following binary data to the W register
38      movwf  portB          ; move W to port B to update the display
39      call   pause          ; delay subroutine by 200ms
40
41      movlw  b'11001111'    ; move the following binary data to the W register
42      movwf  portB          ; move W to port B to update the display
43      call   pause          ; delay subroutine by 200ms
44
45      movlw  b'10100100'    ; move the following binary data to the W register
46      movwf  portB          ; move W to port B to update the display
47      call   pause          ; delay subroutine by 200ms
```

Fig. 37

```

48
49     movlw b'10110000' ; move the following binary data to the W register
50     movwf portB       ; move W to port B to update the display
51     call  pause       ; delay subroutine by 200ms
52
53     movlw b'10011001' ; move the following binary data to the W register
54     movwf portB       ; move W to port B to update the display
55     call  pause       ; delay subroutine by 200ms
56
57     movlw b'10010010' ; move the following binary data to the W register
58     movwf portB       ; move W to port B to update the display
59     call  pause       ; delay subroutine by 200ms
60
61     movlw b'10000010' ; move the following binary data to the W register
62     movwf portB       ; move W to port B to update the display
63     call  pause       ; delay subroutine by 200ms
64
65     movlw b'11111000' ; move the following binary data to the W register
66     movwf portB       ; move W to port B to update the display
67     call  pause       ; delay subroutine by 200ms
68
69     movlw b'10000000' ; move the following binary data to the W register
70     movwf portB       ; move W to port B to update the display
71     call  pause       ; delay subroutine by 200ms
72
73     movlw b'10011000' ; move the following binary data to the W register
74     movwf portB       ; move W to port B to update the display
75     call  pause       ; delay subroutine by 200ms
76
77     movlw b'10001000' ; move the following binary data to the W register
78     movwf portB       ; move W to port B to update the display
79     call  pause       ; delay subroutine by 200ms
80
81     movlw b'10000011' ; move the following binary data to the W register
82     movwf portB       ; move W to port B to update the display
83     call  pause       ; delay subroutine by 200ms
84
85     movlw b'11000110' ; move the following binary data to the W register
86     movwf portB       ; move W to port B to update the display
87     call  pause       ; delay subroutine by 200ms
88
89     movlw b'10100001' ; move the following binary data to the W register
90     movwf portB       ; move W to port B to update the display
91     call  pause       ; delay subroutine by 200ms
92
93     movlw b'10000110' ; move the following binary data to the W register
94     movwf portB       ; move W to port B to update the display

```

Fig. 38 and Fig. 39

```

95     call  pause       ; delay subroutine by 200ms
96
97     movlw b'10001110' ; move the following binary data to the W register
98     movwf portB       ; move W to port B to update the display
99     call  pause       ; delay subroutine by 200ms
100
101
102     goto  get_count   ; repeat forever
103
104 pause
105     movlw 0xff         ; set w = 255 decimal
106     movwf mCount      ; mCount = w: outer loop counter
107
108 loadN
109     movlw 0xff         ; set w = 255 decimal
110     movwf nCount      ; nCount = w: inner loop counter
111
112 decN
113     decfsz nCount, f   ; nCount--
114     goto  decN         ; if nCount != 0 then repeat nCount--
115
116     decfsz mCount, f   ; else decrement Count
117     goto  loadN        ; if mCount != 0 then reload nCount to 255 and decrement
118
119     return            ; else exit subroutine
120 end

```

Task 2:

The second task specified in the assignment brief is to modify the code so that when one of the buttons is pressed, the 7-segment display will show the letter 'r' and when the other button is pressed the display will read 'L', however whilst no buttons are pressed the display will be blank.

To achieve this, loops were needed to add more complexity to the code as the addition of buttons adds more variables in outputs. One main loop named 'LOOP', in this code, is used to specify if either button is being pressed, and if not to send no outputs to the display so nothing is displayed. However, if either of the buttons are pressed, for example the button that is designated to display 'r' when pressed, the code continues but in a loop named 'r' which displays 'r' on the 7-segment display, then the code continues in the main loop (LOOP) to check if the button is still being pressed or the other button.

The other modification that was essential was to specify and define 'portA' with the same code used to achieve this with 'portB'.

Please see below in figs. 40, 41 for the written code, and accompanying annotations.

```

main.asm
1  #include p16f84a.inc           ; Include register definition file
2
3  portA    equ 0x05             ; port A address
4  trisa    equ 0x85             ; trisa address in register bank 1
5
6  portB    equ 0x06             ; port B address
7  trisb    equ 0x86             ; trisB address in register bank 1
8
9  status   equ 0x03             ; status register
10 RP0      equ 5                 ; bit 5 of status register
11 RP1      equ 6                 ; bit 6 of status register
12
13
14 count    equ 0x0c             ; counter 0 to 255
15 nCount   equ 0x0d             ; pause routine inner loop counter
16 mCount   equ 0x0e             ; pause routine outer loop counter
17
18
19 RST      code 0x00             ; Reset Vector
20          goto Start
21
22
23 PGM      code
24 Start
25          bcf    status,RP0      ; clear bank selection bits in the status reg
26          bcf    status,RP1      ; clear bank selection bits in the status reg
27
28          bsf    status,RP0      ; select bank 1
29          movlw  0x00             ; load w with 0x00
30          movwf  trisb           ; load it into trisb to make portb all outputs
31          bcf    status,RP0      ; reselect bank 0
32
33          clrf   portB           ; clear all lines low
34          clrf   count           ; initialize counter to 0
35
36 LOOP:
37
38          BTFSS  portA,0          ; if button '0' pressed then complete next command
39          GOTO  L                 ; continue in loop 'L'
40
41          BTFSS  portA,1          ; if button '1' pressed then complete next command
42          GOTO  r                 ; continue in loop 'r'
43
44          movlw  b'11111111'     ; move the following binary data to the W register
45          movwf  portB           ; move W to port B to update the display
46          call  pause            ; delay subroutine by 200ms
47

```

Fig. 40

```

48      GOTO LOOP                ; continue in loop 'LOOP'
49
50  L:
51      movlw b'11000111'        ; move the following binary data to the W register
52      movwf portB              ; move W to port B to update the display
53      call pause               ; delay subroutine by 200ms
54      GOTO LOOP                ; continue in loop 'LOOP'
55
56  r:
57      movlw b'10101111'        ; move the following binary data to the W register
58      movwf portB              ; move W to port B to update the display
59      call pause               ; delay subroutine by 200ms
60      GOTO LOOP                ; continue in loop 'LOOP'
61
62
63  pause
64      movlw 0xff               ; set w = 255 decimal
65      movwf mCount             ; mCount = w: outer loop counter
66
67  loadN
68      movlw 0xff               ; set w = 255 decimal
69      movwf nCount             ; nCount = w: inner loop counter
70
71  decN
72      decfsz nCount, f          ; nCount--
73      goto decN                ; if nCount != 0 then repeat nCount--
74
75      decfsz mCount, f          ; else decrement Count
76      goto loadN               ; if mCount != 0 then
77                               ; reload nCount to 255 and decrement
78                               ; else exit subroutine
79      return
80  end

```

Fig. 41

Task 3:

As describe in the assignment brief, task 3 is to modify the code once again so that the circuit performs a “latching function”, in which when one button is pressed, all the segments of the display turn ‘on’, but when the button is released the segments continue to be ‘on’. Only when the button is pressed again, all the segments turn ‘off’.

To achieve this, in the most efficient way, the code for task 2 will be modified. This is efficient because all of the needed ports have already been defined and are working. Also, a very similar loop structure is needed for this function. The only differences that are needed is to delete all of the time-based functions, the ‘call pause’ function, and for the main loop (LOOP) to not have a default output; this way the code will continue to output from the last loop selected by the last button pressed, and therefor will perform the ‘latching’ function. The only other changes would be cosmetic (naming of the loops) and changing the binary to display all the segments or none for each outputting loop.

Please see below in figs. 42, 43 for the written code, and accompanying annotations.

```

main.asm
1  #include p16f84a.inc          ; Include register definition file
2
3  portA    equ 0x05             ; port A address
4  trisa    equ 0x05             ; trisa address in register bank 1
5
6  portB    equ 0x06             ; port B address
7  trisb    equ 0x06             ; trisB address in register bank 1
8
9  status   equ 0x03             ; status register
10 RP0      equ 5                 ; bit 5 of status register
11 RP1      equ 6                 ; bit 6 of status register
12
13 count     equ 0x0c             ; counter 0 to 255
14 nCount    equ 0x0d             ; pause routine inner Loop counter
15 mCount    equ 0x0e             ; pause routine outer Loop counter
16
17
18 RST       code 0x0             ; Reset Vector
19 goto     Start
20
21
22 PGM       code
23 Start
24     bcf    status,RP0          ; clear bank selection bits in the status reg
25     bcf    status,RP1          ; clear bank selection bits in the status reg
26
27     bsf    status,RP0          ; select bank 1
28     movlw  0x00                ; Load w with 0x00
29     movwf  trisb               ; Load it into trisb to make portb all outputs
30     bcf    status,RP0          ; reselect bank 0
31
32     clrf   portB               ; clear all lines low
33     clrf   count               ; initialize counter to 0
34
35 LOOP:
36
37     BTFSS  portA,0              ; if button '0' pressed complete next command
38     GOTO  ON                    ; continue in loop 'ON'
39
40     BTFSS  portA,1              ; if button '1' pressed complete next command
41     GOTO  OFF                    ; continue in loop 'OFF'
42
43
44     GOTO  LOOP                  ; continue in loop 'LOOP'

```

Fig. 42

```

47 ON:
48     movlw  b'10000000'         ; move the following binary data to the W register
49     movwf  portB                ; move W to port B to update the display
50     GOTO  LOOP                  ; continue in loop 'LOOP'
51
52 OFF:
53     movlw  b'11111111'         ; move the following binary data to the W register
54     movwf  portB                ; move W to port B to update the display
55     GOTO  LOOP                  ; continue in loop 'LOOP'
56
57
58 pause
59     movlw  0xffff               ; set w = 255 decimal
60     movwf  mCount               ; mCount = w: outer loop counter
61
62 loadN
63     movlw  0xff                 ; set w = 255 decimal
64     movwf  nCount               ; nCount = w: inner loop counter
65
66 decN
67     decfsz nCount, f            ; nCount--
68     goto  decN                  ; if nCount != 0 then repeat nCount--
69
70     decfsz mCount, f           ; else decrement Count
71     goto  loadN                 ; if mCount != 0 then reload nCount to 255 and decrement
72
73     return                      ; else exit subroutine
74 end

```

Fig. 43

Task 4:

Task 4 is to modify the code one again so that the 7-segment display displays the digits 0-9, but when one button is pressed the display shows the next digit in consecutive order, and when the other button is pressed the display shows the previous digit in consecutive order. However, similar to the function of task 3, if no buttons are pressed it will display the last digit displayed, additionally when the display reaches 0 or 9 the code will not allow the sequence to cycle, so that if the display reaches 9 due to the inputs of the buttons and the button is pressed to display the next character in consecutive order it will not cycle to display 0, it will instead continue to display 9. This is also true if the display is on 0 and the button is pressed which displays the previous digit it will not display 9, instead it will continue displaying 0.

To achieve this, a theory was trialled that this code did not need a main loop like the previous codes, all this code needs to function is a loop per digit of the display, making this code have 10 loops, and the code simply repeats the loop it is currently running and only moves loops when a button is pressed to move to the next or the previous loop in the sequence. To better explain this theory, see fig. 44 below.

```
86  THREE:  movlw  b'10110000'  ; move the following binary data to the W register
87          movwf  portB      ; move W to port B to update the display
88          call   pause      ; delay subroutine by 200ms
89          call   pause      ; delay subroutine by 200ms
90          call   pause      ; delay subroutine by 200ms
91
92
93
94          BTFSS  portA,0      ; if button '0' pressed then complete next command
95          GOTO   FOUR        ; continue in loop 'FOUR'
96
97          BTFSS  portA,1      ; if button '1' pressed then complete next command
98          GOTO   TWO         ; continue in loop 'TWO'
99
100         BTFSC  portA,0      ; if button '0' is not pressed then complete next command
101         BTFSC  portA,1      ; if button '1' is not pressed then complete next command
102         GOTO   THREE        ; continue in loop 'THREE'
```

Fig. 44

This is an example of one of the loops in this finished code. Each loop starts by displaying the appropriate digit then is delayed by 600ms to make the digit more easily observed. Then the code 'asks' if button 0 has been pressed and if so move to the next loop, which in this example would be loop 'FOUR' displaying '4' on the display. Similarly, the code then 'asks' if button 1 has been pressed and is so continue on loop 'TWO' which will be displaying '2'. Then, the code 'double checks' if either button has been pressed and if not continue to loop 'THREE', which is the same loop, so that the display continues to display the digit '3'.

All of the 10 loops are identical to this functionally, apart from loop 'NINE' and 'ZERO' which where the code would continue to the previous loop on 'ZERO' and the next loop on 'NINE', the code instead continues to the same loop so that digit is still displayed.

This code achieves this function, see fig. 45, 46, 47, 48, 49 below for the complete code and supplementary annotations.

```

main.asm
1  #include p16f84a.inc          ; Include register definition file
2
3  portA equ 0x05                ; port A address
4  trisa equ 0x05                ; trisa address in register bank 1
5
6  portB equ 0x06                ; port B address
7  trisb equ 0x06                ; trisb address in register bank 1
8
9  status equ 0x03               ; status register
10 RP0 equ 5                     ; bit 5 of status register
11 RP1 equ 6                     ; bit 6 of status register
12
13 count equ 0x0c                ; counter 0 to 255
14 nCount equ 0x0d               ; pause routine inner loop counter
15 mCount equ 0x0e               ; pause routine outer loop counter
16
17 RST code 0x00                 ; Reset Vector
18 goto Start
19
20
21 PGH code
22 Start
23   bcf status,RP0               ; clear bank selection bits in the status reg
24   bcf status,RP1               ; clear bank selection bits in the status reg
25
26   bsf status,RP0               ; select bank 1
27   movlw 0x00                   ; load w with 0x00
28   movwf trisb                  ; load it into trisb to make portb all outputs
29   bcf status,RP0               ; reselect bank 0
30
31   clrf portB                   ; clear all lines low
32   clrf count                   ; initialize counter to 0
33
34
35 ZERO:
36   movlw b'11000000'           ; move the following binary data to the W register
37   movwf portB                  ; move W to port B to update the display
38   call pause                   ; delay subroutine by 200ms
39   call pause                   ; delay subroutine by 200ms
40   call pause                   ; delay subroutine by 200ms
41
42
43   BTFSS portA,0                ; if button '0' pressed then complete next command
44   GOTO ONE                      ; continue in loop 'ONE'
45
46   BTFSS portA,1                ; if button '1' pressed then complete next command
47   GOTO ZERO                    ; continue in loop 'ZERO'

```

Fig. 45

```

48      BTFSC  portA,0      ; if button '0' is not pressed then complete next command
49      BTFSC  portA,1      ; if button '1' is not pressed then complete next command
50      GOTO   ZERO        ; continue in Loop 'ZERO'
51
52  ONE:
53      movlw  b'11001111'  ; move the following binary data to the W register
54      movwf  portB         ; move W to port B to update the display
55      call   pause        ; delay subroutine by 200ms
56      call   pause        ; delay subroutine by 200ms
57      call   pause        ; delay subroutine by 200ms
58
59      BTFSS  portA,0      ; if button '0' pressed then complete next command
60      GOTO   TWO         ; continue in Loop 'TWO'
61
62      BTFSS  portA,1      ; if button '1' pressed then complete next command
63      GOTO   ZERO        ; continue in Loop 'ZERO'
64
65      BTFSC  portA,0      ; if button '0' is not pressed then complete next command
66      BTFSC  portA,1      ; if button '1' is not pressed then complete next command
67      GOTO   ONE         ; continue in Loop 'ONE'
68
69  TWO:
70      movlw  b'10100100'  ; move the following binary data to the W register
71      movwf  portB         ; move W to port B to update the display
72      call   pause        ; delay subroutine by 200ms
73      call   pause        ; delay subroutine by 200ms
74      call   pause        ; delay subroutine by 200ms
75
76      BTFSS  portA,0      ; if button '0' pressed then complete next command
77      GOTO   THREE       ; continue in Loop 'THREE'
78
79      BTFSS  portA,1      ; if button '1' pressed then complete next command
80      GOTO   ONE         ; continue in Loop 'ONE'
81
82      BTFSC  portA,0      ; if button '0' is not pressed then complete next command
83      BTFSC  portA,1      ; if button '1' is not pressed then complete next command
84      GOTO   TWO         ; continue in Loop 'TWO'
85
86  THREE:
87      movlw  b'10110000'  ; move the following binary data to the W register
88      movwf  portB         ; move W to port B to update the display
89      call   pause        ; delay subroutine by 200ms
90      call   pause        ; delay subroutine by 200ms
91      call   pause        ; delay subroutine by 200ms
92
93      BTFSS  portA,0      ; if button '0' pressed then complete next command
94

```

Fig. 46 and 47.

```

95      GOTO   FOUR        ; continue in Loop 'FOUR'
96
97      BTFSS  portA,1      ; if button '1' pressed then complete next command
98      GOTO   TWO         ; continue in Loop 'TWO'
99
100     BTFSC  portA,0      ; if button '0' is not pressed then complete next command
101     BTFSC  portA,1      ; if button '1' is not pressed then complete next command
102     GOTO   THREE       ; continue in Loop 'THREE'
103
104  FOUR:
105     movlw  b'10011001'  ; move the following binary data to the W register
106     movwf  portB         ; move W to port B to update the display
107     call   pause        ; delay subroutine by 200ms
108     call   pause        ; delay subroutine by 200ms
109     call   pause        ; delay subroutine by 200ms
110
111     BTFSS  portA,0      ; if button '0' pressed then complete next command
112     GOTO   FIVE        ; continue in Loop 'FIVE'
113
114     BTFSS  portA,1      ; if button '1' pressed then complete next command
115     GOTO   THREE       ; continue in Loop 'THREE'
116
117     BTFSC  portA,0      ; if button '0' is not pressed then complete next command
118     BTFSC  portA,1      ; if button '1' is not pressed then complete next command
119     GOTO   FOUR        ; continue in Loop 'FOUR'
120
121  FIVE:
122     movlw  b'10010010'  ; move the following binary data to the W register
123     movwf  portB         ; move W to port B to update the display
124     call   pause        ; delay subroutine by 200ms
125     call   pause        ; delay subroutine by 200ms
126     call   pause        ; delay subroutine by 200ms
127
128     BTFSS  portA,0      ; if button '0' pressed then complete next command
129     GOTO   SIX         ; continue in Loop 'SIX'
130
131     BTFSS  portA,1      ; if button '1' pressed then complete next command
132     GOTO   FOUR        ; continue in Loop 'FOUR'
133
134     BTFSC  portA,0      ; if button '0' is not pressed then complete next command
135     BTFSC  portA,1      ; if button '1' is not pressed then complete next command
136     GOTO   FIVE        ; continue in Loop 'FIVE'
137
138  SIX:
139     movlw  b'10000010'  ; move the following binary data to the W register
140     movwf  portB         ; move W to port B to update the display
141     call   pause        ; delay subroutine by 200ms
142     call   pause        ; delay subroutine by 200ms
143     call   pause        ; delay subroutine by 200ms
144

```

```

142      call    pause           ; delay subroutine by 200ms
143
144
145      BTFSS   portA,0          ; if button '0' pressed then complete next command
146      GOTO    SEVEN           ; continue in Loop 'SEVEN'
147
148      BTFSS   portA,1          ; if button '1' pressed then complete next command
149      GOTO    FIVE            ; continue in Loop 'FIVE'
150
151      BTFSC   portA,0          ; if button '0' is not pressed then complete next command
152      BTFSC   portA,1          ; if button '1' is not pressed then complete next command
153      GOTO    SIX             ; continue in Loop 'SIX'
154
155      SEVEN:  movlw   b'11111000' ; move the following binary data to the W register
156      movwf   portB           ; move W to port B to update the display
157      call    pause           ; delay subroutine by 200ms
158      call    pause           ; delay subroutine by 200ms
159      call    pause           ; delay subroutine by 200ms
160
161
162      BTFSS   portA,0          ; if button '0' pressed then complete next command
163      GOTO    EIGHT           ; continue in Loop 'EIGHT'
164
165      BTFSS   portA,1          ; if button '1' pressed then complete next command
166      GOTO    SIX             ; continue in Loop 'SIX'
167
168      BTFSC   portA,0          ; if button '0' is not pressed then complete next command
169      BTFSC   portA,1          ; if button '1' is not pressed then complete next command
170      GOTO    SEVEN           ; continue in Loop 'SEVEN'
171
172      EIGHT:  movlw   b'10000000' ; move the following binary data to the W register
173      movwf   portB           ; move W to port B to update the display
174      call    pause           ; delay subroutine by 200ms
175      call    pause           ; delay subroutine by 200ms
176      call    pause           ; delay subroutine by 200ms
177
178
179      BTFSS   portA,0          ; if button '0' pressed then complete next command
180      GOTO    NINE            ; continue in Loop 'NINE'
181
182      BTFSS   portA,1          ; if button '1' pressed then complete next command
183      GOTO    SEVEN           ; continue in Loop 'SEVEN'
184
185      BTFSC   portA,0          ; if button '0' is not pressed then complete next command
186      BTFSC   portA,1          ; if button '1' is not pressed then complete next command
187      GOTO    EIGHT           ; continue in Loop 'EIGHT'
188      NINE:

```

Fig. 48 and 49.

```

189      movlw   b'10011000' ; move the following binary data to the W register
190      movwf   portB           ; move W to port B to update the display
191      call    pause           ; delay subroutine by 200ms
192      call    pause           ; delay subroutine by 200ms
193      call    pause           ; delay subroutine by 200ms
194
195
196      BTFSS   portA,0          ; if button '0' pressed then complete next command
197      GOTO    NINE            ; continue in Loop 'NINE'
198
199      BTFSS   portA,1          ; if button '1' pressed then complete next command
200      GOTO    EIGHT           ; continue in Loop 'EIGHT'
201
202      BTFSC   portA,0          ; if button '0' is not pressed then complete next command
203      BTFSC   portA,1          ; if button '1' is not pressed then complete next command
204      GOTO    NINE            ; continue in Loop 'NINE'
205
206
207      pause
208      movlw   0xffff          ; set W = 255 decimal
209      movwf   mCount          ; mCount = W: outer Loop counter
210
211      loadN:  movlw   0xff      ; set W = 255 decimal
212      movwf   nCount          ; nCount = W: inner Loop counter
213
214      decN:   decfsz  nCount, f ; nCount--
215      goto    decN            ; if nCount != 0 then repeat nCount--
216
217      decfsz  mCount, f ; else decrement Count
218      goto    loadN          ; if mCount != 0 then reload nCount to 255 and decrement
219
220      return ; else exit subroutine
221      end

```

Task 5:

Task 5, in comparison to task 4, is far more straight forward. The task is to modify the code once more so that when one of the buttons are pressed the sounder/speaker subsystem will activate and sound a noise whilst the button is pressed and stop when the same button is not pressed.

On observation of the circuit the sounder/speaker subsystem is activated by allowing the current to flow by a transistor, controlled by the pin from 'portB' that is not connected to the 7-segment display. After observation, all that is needed is to have the same button loops used before and just to output on the first bit of the 'portB' output byte when the button is pressed.

This code achieves this function, see fig. 50, 51. below of the complete code and supplementary annotations.

```
*main.asm
1  #include p16f84a.inc          ; Include register definition file
2
3  portA    equ 0x05             ; port A address
4  trisa    equ 0x05             ; trisa address in register bank 1
5
6  portB    equ 0x06             ; port B address
7  trisb    equ 0x06             ; trisb address in register bank 1
8
9  status   equ 0x03             ; status register
10 RP0      equ 5                ; bit 5 of status register
11 RP1      equ 6                ; bit 6 of status register
12
13 count    equ 0x0c             ; counter 0 to 255
14 nCount   equ 0x0d             ; pause routine inner loop counter
15 mCount   equ 0x0e             ; pause routine outer loop counter
16
17 RST      code 0x0             ; Reset Vector
18      goto Start
19
20 PGM      code
21 Start
22      bcf    status,RP0         ; clear bank selection bits in the status reg
23      bcf    status,RP1
24
25      bsf    status,RP0         ; select bank 1
26      movlw  0x00               ; load w with 0x00
27      movwf  trisb              ; load it into trisb to make portb all outputs
28      bcf    status,RP0         ; reselect bank 0
29
30      clrf   portB              ; clear all lines low
31      clrf   count              ; initialize counter to 0
32
33 LOOP:
34
35      movlw  b'11111111'        ; move the following binary data to the W register
36      movwf  portB              ; move W to port B to update the display
37
38      BTFSS  portA,0             ; if button '0' pressed complete next command
39      GOTO   SPEAKER            ; continue to 'SPEAKER' loop
40
41      BTFSC  portA,0             ; if button '0' is not pressed complete next command
42      BTFSC  portA,1             ; if button '1' is not pressed complete next command
```

Fig. 50

```

48      GOTO    LOOP          ; continue to 'LOOP' loop (start again)
49
50
51  SPEAKER:
52
53      movlw   b'01111111'    ; move the following binary data to the W register
54      movwf   portB          ; complete at 'portB'
55
56
57
58
59  pause
60      movlw   0xffff          ; set w = 255 decimal
61      movwf   mCount          ; mCount = w: outer loop counter
62
63  loadN
64      movlw   0xff            ; set w = 255 decimal
65      movwf   nCount          ; nCount = w: inner loop counter
66
67  decN
68      decfsz  nCount, f        ; nCount--
69      goto    decN            ; if nCount != 0 then repeat nCount--
70
71      decfsz  mCount, f        ; else decrement Count
72      goto    loadN           ; if mCount != 0 then reload nCount to 255 and decrement
73
74      return                  ; else exit subroutine
75  end

```

Fig. 51

References:

- [1]. Digikey, PIC16F84. [Online]. Available: <https://www.digikey.co.uk/en/products/detail/microchip-technology/PIC16F84-10-P/243464>
- [2]. Microchip, Master Clear Pin. [Online]. Available: <https://developerhelp.microchip.com/xwiki/bin/view/products/mcu-mpu/8bit-pic/reset-options/master-clear-pin-external-reset/#:~:text=MCLR%3A%20The%20Master%20Clear%20pin,controlled%20by%20a%20configuration%20setting.>
- [3]. Wikipedia, seven segment display. [Online]. Available: https://en.wikipedia.org/wiki/Seven-segment_display
- [4]. Wikipedia, Crystal Oscillator. [Online]. Available: https://en.wikipedia.org/wiki/Crystal_oscillator#/media/File:16MHZ_Crystal.jpg
- [5]. Electronics Tutorials, Crystal Oscillator. [Online]. Available: <https://www.electronics-tutorials.ws/oscillator/crystal.html>
- [6]. RS - Online, Piezo Buzzers. [Online]. Available: https://uk.rs-online.com/web/c/passive-components/sounder-buzzer-microphone-components/piezo-buzzers/?srsltid=AfmBOopqBjRZsf9njv_8Kv9Z7Dwdqv-KFJcnL4pQ8vFsGxOyEPWsiuEM
- [7]. Murata, Sounder. [Online]. Available: <https://www.murata.com/products/sound/sounder>
- [8]. Electro Technik, Transistor Overview. [Online]. Available: https://www.electrotechnik.net/2015/01/transistor-overview_18.html
- [9]. RS - Online, Voltage Regulators. [Online]. Available: <https://uk.rs-online.com/web/p/voltage-regulators/1686809>
- [10]. RS - Online, Resistor Arrays. [Online]. Available: <https://uk.rs-online.com/web/p/resistor-arrays/5223220>
- [11]. RS - Online, Through Hole Resistors. [Online]. Available: <https://uk.rs-online.com/web/p/through-hole-resistors/1742637>
- [12]. RS - Online, Aluminium Capacitors. [Online]. Available: <https://uk.rs-online.com/web/p/aluminium-capacitors/1443346>
- [13]. Cool Components, Component Buttons. [Online]. Available: <https://coolcomponents.co.uk/collections/component-buttons>
- [14]. Future Electronics, Switches. [Online]. Available: <https://www.futureelectronics.com/c/electromechanical/switches>
- [15]. Canford, Duracell AA. [Online]. Available: https://www.canford.co.uk/Products/59-107_DURACELL-MN1500-BATTERY-AA-size-alkaline-1.5V-pack-of-4
- [16]. Wikipedia, Ground, Electricity. [Online]. Available: https://en.wikipedia.org/wiki/Ground_%28electricity%29
- [17]. Electronics Fur U, seven segment display. [Online]. Available: <https://www.electronicshobby.com/resources/7-segment-display-pinout-understanding/#:~:text=Each%20segment%20in%20a%207%2Dsegment%20display%20corresponds%20to%20a,or%20character%20on%20the%20display.>

FINAL GRADE

GENERAL COMMENTS

60/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

GRADING FORM: GJ_2019

GABRIEL PAN

299

ENGAGEMENT WITH LITERATURE



A few relevant sources cited - could have included even more though (the links to the images/components don't really count for EWL)

60

KNOWLEDGE AND UNDERSTANDING



Some ok comments, some not so much (e.g. having more RAM doesn't mean it can deliver higher speeds), and statements like 'proved to be ruthlessly effective' are somewhat inappropriate.

62

COGNITIVE AND INTELLECTUAL



A reasonable attempt at the programming, but generally a 'brute force' method was used, so the marks for this section will be reduced.

52

PRACTICAL APPLICATION



A good circuit implementation.
Software attempts ok, but no flowcharts or pseudocode as requested ??

55

TRANSFERABLE SKILLS



Generally well laid out and presented

70