

Robert Webb 2308625 Micros Peripherals and Interfaces Assignment

by Bobby Webb

Submission date: 13-Jan-2025 11:11PM (UTC+0000)

Submission ID: 248345139

File name:

51462_Bobby_Webb_Robert_Webb_2308625_Micros_Peripherals_and_Interfaces_Assignment_621295_185869301.docx
(3.18M)

Word count: 3555

Character count: 17735

Micros, Peripherals and Interfaces

Robert Webb

2308625

Embedded Electronic Degree Apprenticeship

L4 – Micros, Peripherals and interfaces

ACDA4004

Table of Contents

List of Figures.....	3
Introduction:.....	3
1.0.Count Project Circuit:.....	4
1.1. PIC16F84A:.....	5
1.2. Seven Segment Display:.....	6
1.3. Push latching switches:.....	6
1.4. Resistors:.....	7
1.5. Resistor Bank:.....	7
1.6. Capacitors:.....	7
1.7. Positive Voltage Regulator (7805):.....	8
1.8. Crystal Oscillator:.....	8
1.9. Sounder:.....	9
2.0. Display the Hexadecimal characters 0-F on the 7-segment display.	10
3.0. Section 3: Coding Tasks.....	12
3.1. Coding Task 1:.....	12
3.2. Coding Task 2:.....	14
3.3. Coding Task 3:.....	15
3.4. Coding Task 4:.....	16
3.5. Coding Task 5:.....	18
4.0.Conclusion:.....	19
References	20

List of Figures

Figure 1: Prototype Circuit Screenshot	4
Figure 2: Author's Version of Circuit	4
Figure 3: PIC16F84A Pins [13]	5
Figure 4: 7 Segment display [3]	6
Figure 5: Push Momentary Switch [4]	6
Figure 6: 10K Resistor Bands	7
Figure 7: Resistor Circuit Symbol	7
Figure 8: Resistor Bank	7
Figure 9: Capacitors [6]	8
Figure 10: LM7805 Pin Layout Diagram [7]	8
Figure 11: Crystal Oscillator [9]	9
Figure 12: Sounder Circuit Symbol	9
Figure 13: Code Task 1. Pt1	12
Figure 14: Code Task 1. Pt2	12
Figure 15: Code Task 1. Pt 3	13
Figure 16: Code Task 1 Flow Chart	13
Figure 17: Code Task 2. Pt 1	14
Figure 19: Code Task 2 Flow Chart	14
Figure 20: Code Task 3. Pt 1	15
Figure 22: Code Task 3 Flow Chart	15
Figure 23: Code Task 4. Pt 1	16
Figure 24: Code Task 4. Pt 2	16
Figure 25: Code Task 4. Pt 3	17
Figure 26: Code Task 4. Pt 4	17
Figure 28: Code Task 4 Flow Chart	18
Figure 29: Code Task 5. Pt 1	18
Figure 31: Code Task 5 Flow Chart	19

Introduction:

In this report, you shall read about a proteus board which was given to the author to write more code to allow additional tasks to be completed with the 7-segment display. The circuits will involve PIC16F84A microcontrollers which will be used throughout this assignment. Following this the author will explain how they programmed a circuit to count up from 0-F in hexadecimal. After this there are coding tasks which have been completed which relates mostly to the same circuit which enables it to do different tasks, for example, getting the screen to count up or down from 5 depending on which button has been pressed. This is followed by a flowchart for each task about how the code/circuit executes the task.

1.0.Count Project Circuit:

For this task, the author was introduced to a 'count' circuit which runs a seven-segment common anode display with a PIC16F84. The diagram shown below in Fig.1 is the original diagram included in the tutorial. The circuit comprises of a range of components in a simple layout which is easy to understand and visualise. Within section 1, the author was tasked to recreate this circuit as closely as possible using Proteus software, Fig.2 shows the author's version of the circuit layout.

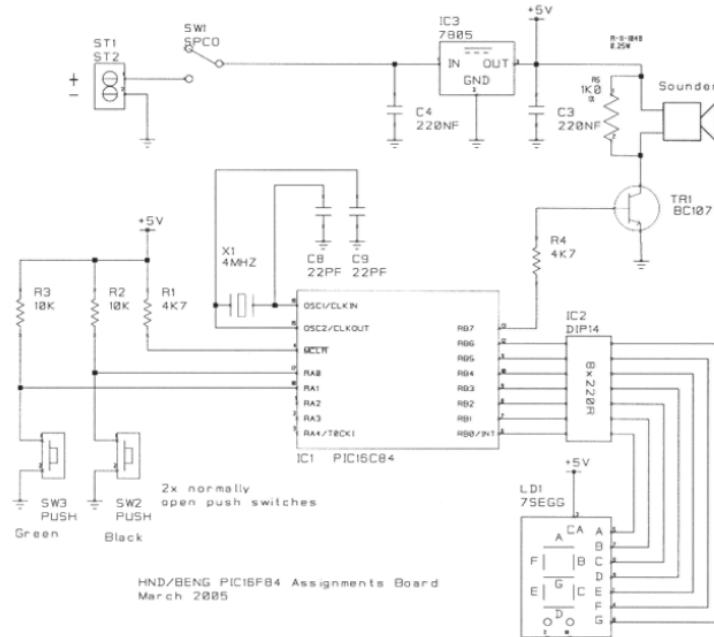


Figure 1: Prototype Circuit Screenshot

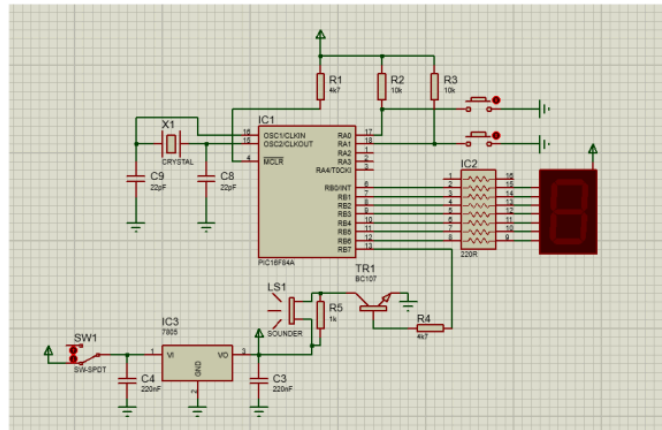


Figure 2: Author's Version of Circuit

As previously stated, fig.2 shows the authors version of the original circuit diagram. The circuit is very similar and well thought out in terms of component placement, meaning the diagram doesn't look messy and is simple to trace where the components are being connected.

1.1. PIC16F84A:

In this proteus circuit, a PIC16F84A has been used which is part of the microcontroller family. This PIC utilises two memory blocks, one being program memory and the other being data memory, each one has its own bus which allows each block to access during the same oscillator cycle. [1]

The data memory also contains EEPROM memory which is memory that isn't directly mapped into the data memory. There are indirect address pointers that specify the address of the EEPROM to read/write memory. EEPROM is a form of non-volatile ROM which allows individual bytes of data to be erased and then reprogrammed, they can also be called byte erasable chips. The data memory which can also be known as the RAM (Random Access Memory) contains 68 bytes and the EEPROM is 64 bytes. These sizes are more than enough to carry out the simple task of counting up in the author's count circuit.

The program memory contains 1000 words which equates to 1024 different instructions for the one PIC, due to each 14-bit program memory word being the same width as each device's instruction.

Figure 3 shows the pin output of the PIC which is used to call certain ports within the program which can be used to display data on an LCD screen, and send outputs to LED's or power to motors. It can also accept inputs from buttons when pressed and components alike to allow the program to work properly.

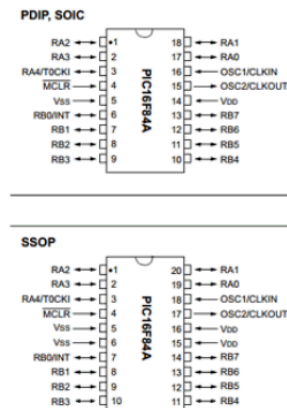


Figure 3: PIC16F84A Pins [13]

Within the count circuit, this microcontroller is crucial for the circuit and code to work, without this one component everything else is useless, this could be referred to as the 'CPU' or 'brains' of the circuit as this is where all of the code is uploaded to. Figure 3 is the PIC utilised in the author's circuit which has all of the ports displayed, within the authors circuit nearly all ports are being utilised, these are for the buttons, crystal oscillators and the main one being the Seven segment display which shows if everything is working as it should as it would display the Hexidecimal values as stated in the count code.

1.2. Seven Segment Display:

Within the circuit there is a seven-segment LED display which is used to display the Hexadecimal 0-F, this display is what shows the numbers that will be outputted via the code and PIC when the simulation is run. The display is known as a seven-segment common anode, these are also available as a common cathode depending on its requirements. These displays are great for this circuit due to the factor of using a relatively low current, an average of 96mW per segment [2]. In the circuit each segment is controlled by LEDs which give off Light when power is supplied via a specific pin from the input, these inputs are controlled by the PIC which means the display only gets a series of power, on or off, high or low, on each pin which makes the LED turn on or not in the desired segments. [3]

The common Cathode name for this display comes from the factor of the cathode (negative terminal) of each led segment is connected to the common negative or a common negative ground. There is also a common anode display which is where all of the anodes (positive) of each LED segment are connected together and supplied with a given voltage.

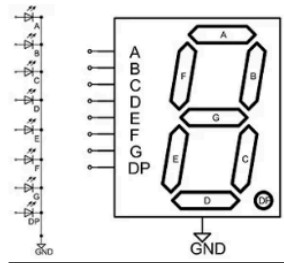


Figure 4: 7 Segment display [3]

The author's circuit display is used to show the current selected number from the circuit, which would be between 0 and G in Hexadecimal on the original count circuit. The purpose of this display is to show the hexadecimal digits as well as allow the author to understand if their code is correct and works as it should.

1.3. Push latching switches:

In the original circuit, SPST momentary push button switches are used to control the display depending on what code is used. I.e. either to count up or down, turn the screen off or on etc. These switches are known as 'Push to make' buttons [4]. A push button switch is used in applications which requires momentary on or off switching, this works by them being in an 'off' state to start with, when the button is pressed the connection closes and the contacts touch, the circuit is completed and the power flows. Most of these buttons have a current rating of 3 amps at 125 V and 1A at 250 V. The power within the circuit is well within this tolerance.

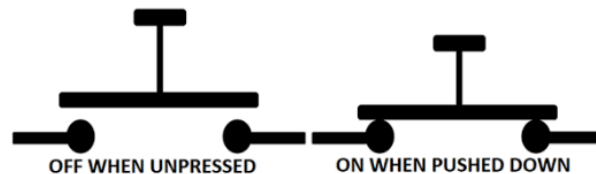


Figure 5: Push Momentary Switch [4]

In this circuit, there are 2 of these buttons, for the count program they are not utilised however further on they are used to control elements such as the count up or count down, the screen on or the screen off to name but a few.

1.4. Resistors:

Within the circuit there are many resistors, these are used to ensure that the components get the required current they need and no more. They act as a 'safety' measure for the components, meaning that they will not get too much current supplied to the components which could cause them to blow due to a surge in power/current. This would then mean the circuit won't work due to a 'gap' in the circuit (dependent on which component blew).



Figure 6: 10K Resistor Bands



Figure 7: Resistor Circuit Symbol

1.5. Resistor Bank:

In the circuit there is the use of a resistor bank, this is used just before the Seven-segment display on the output from the PIC. The resistor bank in this circuit is to tidy up the outputs from the PIC and makes it easier to visualise which ports the connections are coming from and going to on the Seven Segment display.

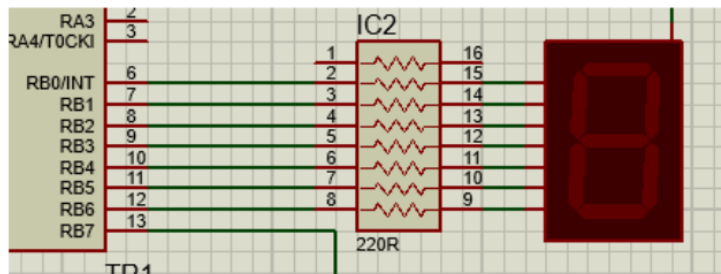


Figure 8: Resistor Bank

As shown in the figure above, within the count circuit the resistor bank is used to make the connections of the seven-segment display look neater and easier to see which pin goes where rather than 7 individual resistors on the diagram. A resistor bank is effectively a component with 8 (or more) resistors within it, the one being used in the Fig. 8 diagram is an 8-way resistor network.

1.6. Capacitors:

Within the author's circuit, there are a lot of capacitors dotted all over the place. 2 are placed around the crystal oscillator connected to the ground and the others are on either side of the voltage regulator. The capacitors used are generic non-electrolytic capacitors, these capacitors use dielectric materials which are formed through mechanical processes, these types of capacitors can also be called electrostatic capacitors. [5]

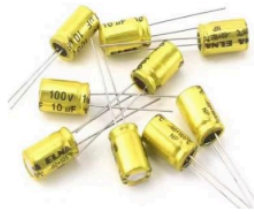


Figure 9: Capacitors [6]

1.7. Positive Voltage Regulator (7805):

The 7805 Positive Voltage Regulator is a member of the 78 series of fixed linear voltage regulators. These regulators are utilised within circuits to maintain voltage in the circuit. The 7806 can receive between 7V and 35V of unregulated voltage. This then outputs a steady 5V DC to the component(s) it is connected to. These work by the differences between the input and the output being released as heat. These components generally have heatsinks on them to help dissipate this heat as it could cause the regulator to get damaged if it can't release the heat fast enough. On the circuit there are capacitors placed around it, these are there to filter residual AC noise, and these regulators perform more efficiently on a clean DC voltage being inputted to it. [7]

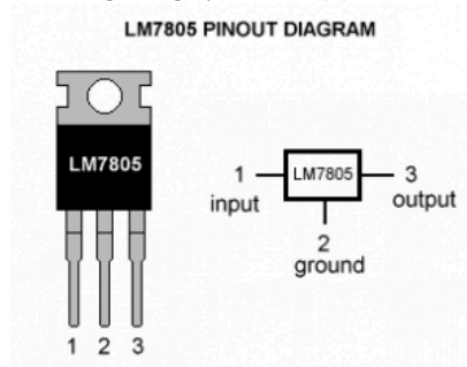


Figure 10: LM7805 Pin Layout Diagram [7]

Within the count circuit, this component is placed around the sounder and power inputs, this is so the component can regulate part of the circuit to ensure there is no overvolting causing other components to get damaged and no longer work. Within this part of the circuit, it is not required a huge amount due to the sounder not being required within this section but can be utilised at a later stage when the sounder is wanted to be used. This would then ensure that the input to the sounder and the rest of the circuit is at a steady clean 5V and not a dirty voltage which could damage other components within the circuit.

1.8. Crystal Oscillator:

A crystal oscillator is one of the components within the count circuit board it is located between the two capacitors within the circuit just branching off from the PIC pins 15 and 16. The crystal is an electronic oscillator circuit which uses piezoelectric crystals as the frequency-selective element [8]. This component is usually utilised to keep track of time. The oscillator works by the quartz crystal ever so slightly when under an electric field, which is known as inverse piezoelectricity. This is when an AC voltage is applied to the electrodes connected to the crystal which causes it to change its shape by resonating which will then start the whole effect and make the component work.

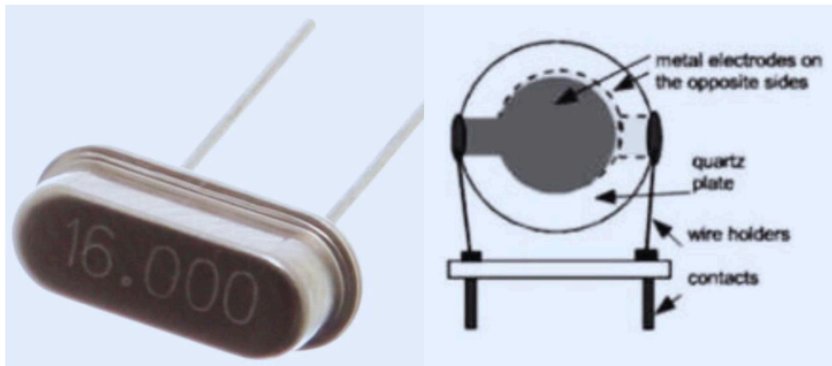


Figure 11: Crystal Oscillator [9]

Figure 11 shows the component and what the 'internals' look like. As shown the electrodes are on opposite sides with the quartz plate which is where it 'changes shape' and the piezoelectricity takes place. These components are available in a large frequency range, ranging from 0.0320 MHz to 1075 MHz [10]. The frequencies of the component will enable it to be used in various applications. In the author's circuit, it is set to 4 MHz, where 4 – 8 MHz is the specific frequency range for microprocessors and TV remotes. At the higher end of the frequency level, it can be utilised on Radar systems and satellite communications, overall this is a key component which has massive amounts of uses in everyday life and plays a crucial role in the author's circuit.

In the circuit, the Crystal oscillator is used to allow the circuit to create a clock for the circuit to count on as well as to help with timings such as requested intervals. Within the author's circuit, this is required due to it being required to count at specific time intervals when counting up from 0-F. The interval is controlled by this component.

1.9. Sounder:

Within the author's circuit, a sounder is used for a task later on to be able to emit a sound when a button is depressed. The count code does not use this component but gets utilised at a later stage. A sounder is used when a noise is wanted to be emitted, this could be to tell the user that a function has been completed or an error etc.

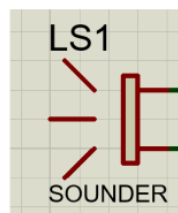


Figure 12: Sounder Circuit Symbol

Figure 12 shows the circuit symbol of the sounder which is utilised in the count circuit. This works by sound being produced by driving a piezoelectric sounder depending on the input signals from the PIC. When the voltage is applied across the 2 connectors the piezoelectric material within the sounder deforms and the sound is emitted because of the movement of the material. [11]

2.0. Display the Hexadecimal characters 0-F on the 7-segment display.

For this task, the author was required to program the circuit to display Hexadecimal characters 0-F on the seven-segment display. The Hexadecimal 0-F had to be converted to binary to allow the PIC to be able to understand which numbers wanted to be displayed. This was done by some simple mathematics of converting the Hexadecimal into the irrespective binary value.

This conversion is from the base-16 number system to the base-2 number system. [12] The Hexadecimal number system is 16 due to 16 values being used to represent the numbers (0-F). When the value gets to '9' we use A-F to represent the numbers '10-15'. The binary is a base 2 number system meaning it uses only 0's and 1's to represent all numbers. These 0 and 1's are also known as bits. [12]

Table 1 below shows the Hexadecimal to Binary conversions which was done to enable the author to output the Hex digits onto the Seven-segment display. This is a basic conversion.

Decimal	Hex	Binary
0	0	00000000
1	1	00000001
2	2	00000010
3	3	00000011
4	4	00000100
5	5	00000101
6	6	00000110
7	7	00000111
8	8	00001000
9	9	00001001
10	A	00001010
11	B	00001011
12	C	00001100
13	D	00001101
14	E	00001110
15	F	00001111

Table 1: Binary Table

The hexadecimal can be converted straight to binary form, there are two methods, an indirect method and a direct method. To do the indirect method the of Hexidecmial to decimal the author had to multiply each digit by its respective place value and then add the products. After this step, the author then converted the decimal into binary, the method for this was to divide the decimal by 2 continuously until the quotient value equalled 0. [12]

Hexadecimal Value	Value is being displayed.	Used Pins for the desired value being displayed	Binary Value for the code.
79	1	B, C	01111001
24	2	A, B, D, E, G	00100100
30	3	A, B, C, D, G	00110000
19	4	B, C, F, G	00011001
12	5	A, C, D, F, G	00010010
02	6	A, C, D, E, F, G	00000010
78	7	A, B, C	01111000
00	8	A, B, C, D, E, F, G	00000000
10	9	A, B, C, D, F, G	00010000
40	0	A, B, C, D, E, F	01000000
08	A	A, B, C, E, F, G	00001000
03	B	C, D, E, F, G	00000011
46	C	A, D, E, F	01000110
21	D	B, C, D, E, G	00100001
06	E	A, D, E, F, G	00000110
0E	F	A, E, F, G	00001110

Table 2: Bit Calculation

The code for the binary to be displayed was to output the binary value to the ports of the PIC, depending on the value will depend on the output high or low on each pin which controls each segment of the display. This output is delayed for the specified time within the code between each clear and send command, the binary bit is output after the clear and time has been completed. This then outputs the next binary value to the display pins which is either high or low power to the desired pins which could make the digit. This was done by the binary for each value being loaded to the correct port to then be refreshed and then sent/displayed on the screen, the value will show for the defined delay of 200ms (0.2s). After each output, the next number is loaded to the register read for the screen to be cleared, once it has been cleared the value to the ports will be pushed and then displayed on the screen. This process continues all the way through from 0 – F, once F is reached it loops around and starts again.

3.0. Section 3: Coding Tasks

3.1. Coding Task 1:

```
main.asm
9 #include p16f84a.inc      ; Include register definition file
10
11 portB equ 0x06           ; portB address Links to the data sheet
12 trisB equ 0x06           ; trisB address in register bank 1
13
14 status equ 0x03          ; status register
15 RP0 equ 5                ; bit 5 of status register
16 RP1 equ 6                ; bit 6 of status register
17
18 count equ 0x0c           ; counter 0 to 255
19 innerCount equ 0x0d      ; pause routine inner loop counter
20 outerCount equ 0x0e      ; pause routine outer loop counter
21
22
23 RST code 0x0             ; Reset Vector
24 goto Start
25 ; CODE:
26 PGH code
27 Start
28 bcf status,RP0           ; clear bank selection bits in the status reg
29 bcf status,RP1
30
31 bsf status,RP0           ; select bank 1
32 movlw 0x00              ; load w with 0x00
33 movwf trisB              ; load it into trisb to make portb all outputs
34 bcf status,RP0           ; reselect bank 0
35
36 clrf portB               ; clear all lines low
37 clrf count              ; initialize counter to 0
38
39 get_count
40
41 movlw b'01000000'       ; Binary for 0
42 movwf portB              ; move w to port B to refresh the display
43 call pause               ; delay 200ms by subroutine
44
45 movlw b'01111001'       ; Binary for number 1
46 movwf portB              ; move w to port B to refresh the display
47 call pause               ; delay 200ms by subroutine
48
49 movlw b'00100100'       ; Binary for number 2
50 movwf portB              ; move w to port B to refresh the display
51 call pause               ; delay 200ms by subroutine
52
53 movlw b'00110000'       ; Binary for number 3
54 movwf portB              ; move w to port B to refresh the display
55 call pause               ; delay 200ms by subroutine
56
57
```

Figure 13: Code Task 1. Pt1

```
55 call pause               ; delay 200ms by subroutine
56
57 movlw b'00011001'       ; Binary for number 4
58 movwf portB              ; move w to port B to refresh the display
59 call pause               ; delay 200ms by subroutine
60
61 movlw b'00010010'       ; Binary for number 5
62 movwf portB              ; move w to port B to refresh the display
63 call pause               ; delay 200ms by subroutine
64
65 movlw b'00000011'       ; Binary for number 6
66 movwf portB              ; move w to port B to refresh the display
67 call pause               ; delay 200ms by subroutine
68
69 movlw b'01111000'       ; Binary for number 7
70 movwf portB              ; move w to port B to refresh the display
71 call pause               ; delay 200ms by subroutine
72
73 movlw b'00000000'       ; Binary for number 8
74 movwf portB              ; move w to port B to refresh the display
75 call pause               ; delay 200ms by subroutine
76
77 movlw b'00010000'       ; Binary for number 9
78 movwf portB              ; move w to port B to refresh the display
79 call pause               ; delay 200ms by subroutine
80
81 movlw b'00001000'       ; Binary for Letter A
82 movwf portB              ; move w to port B to refresh the display
83 call pause               ; delay 200ms by subroutine
84
85 movlw b'00000011'       ; Binary for Letter B
86 movwf portB              ; move w to port B to refresh the display
87 call pause               ; delay 200ms by subroutine
88
89 movlw b'01000110'       ; Binary for Letter C
90 movwf portB              ; move w to port B to refresh the display
91 call pause               ; delay 200ms by subroutine
92
93 movlw b'00100001'       ; Binary for Letter D
94 movwf portB              ; move w to port B to refresh the display
95 call pause               ; delay 200ms by subroutine
96
97 movlw b'00000110'       ; Binary for Letter E
98 movwf portB              ; move w to port B to refresh the display
99 call pause               ; delay 200ms by subroutine
100
101 movlw b'00001110'       ; F the first 0 turns off RB7 then the rest control each led segment on the display
102 movwf portB              ; move w to port B to refresh the display
103
```

Figure 14: Code Task 1. Pt2

```

main.asm
87 call pause ; delay 200ms by subroutine
88
89 movlw b'01000110' ; Binary for letter C
90 movwf portB ; move W to port B to refresh the display
91 call pause ; delay 200ms by subroutine
92
93 movlw b'00100001' ; Binary for letter D
94 movwf portB ; move W to port B to refresh the display
95 call pause ; delay 200ms by subroutine
96
97 movlw b'00000110' ; Binary for letter E
98 movwf portB ; move W to port B to refresh the display
99 call pause ; delay 200ms by subroutine
100
101 movlw b'00001110' ; F the first 0 turns off RB7 then the rest control each led segment on the display
102 movwf portB ; move W to port B to update the display
103 call pause ; delay 200ms by subroutine
104
105 goto get_count ; repeat forever
106
107
-----
108 pause
109
110 movlw 0xff ; set w = 255 decimal
111 movwf outerCount ; mCount = w: outer Loop counter
112 loadw
113 movlw 0xff ; set w = 255 decimal
114 movwf innerCount ; innerCount = w: inner Loop counter
115 decw
116 decfsz innerCount, f ; innerCount--
117 goto dech ; if nCount != 0 then repeat nCount--
118
119 decfsz outerCount, f ; else decrement Count
120 goto loadw ; if mCount != 0 then
121 ; reload nCount to 255 and decrement
122 return ; else exit subroutine
123
124 end ; End of code

```

Figure 15: Code Task 1. Pt 3

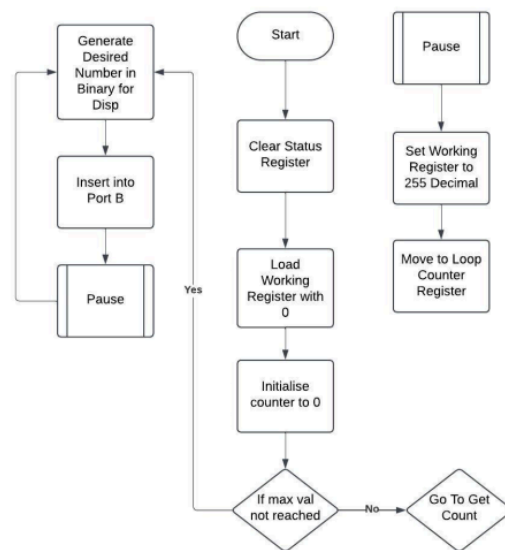


Figure 16: Code Task 1 Flow Chart

3.2. Coding Task 2:

```

5 #include p16f84a.inc           ; Include the register definition file
6
7 portB equ 0x06                 ; Port B address links to the data sheet
8 trisB equ 0x06                 ; TrisB address in register bank 1
9
10 status equ 0x03                ; Status register
11 RP0 equ 5                      ; Bit 5 of status register
12 RP1 equ 6                      ; Bit 6 of status register
13
14 portA equ 0x05                 ; Port A address links to the data sheet
15 trisA equ 0x05                 ; Tris A address in register bank
16
17 Count equ 0x0c                 ; Counter 0 to 255
18 innerCount equ 0x0d            ; Pause routine inner loop counter
19 outerCount equ 0x0e            ; Pause routine outer loop counter
20 org 0                           ; Set the starting address to 0x00
21 movlw 0x01                     ; Set the port A pins
22 movwf trisA                    ; Set to the input
23 bcf status,5                   ; Switch back to bank 0
24 goto But1                      ; Go to But1 line
25 Loop
26 But1 btfsc portA,1             ; If portA bit 1 is 0 skip the next instruction if 1 complete next the instruction
27 goto But2                      ; Goes back to the top of the loop
28 call pause                     ; Call for delay
29 movlw 0x00                     ; Load 0x00 into W
30 tris portB                     ; Tris state register with port B
31 movlw b'01001110'             ; Bit value to output 'r' on seven seg
32 movwf portB                    ; Output to the display - seven segment
33 But2 btfsc portA,0             ; If portA bit 1 is 0 skip next instruction if 1 complete next instruction
34 goto But1                      ; Go to the top of the loop
35 call pause                     ; Call for delay
36 movlw 0x00                     ; Load 0x00 into W
37 tris portB                     ; Tris state register with port B
38 movlw b'01000111'             ; Bit value to display 'L' on the seven seg
39 movwf portB                    ; Output to the display
40 goto Loop                      ; Go back to the loop
41 pause
42 movlw 0xff                     ; Set w = 255 decimal
43 movwf outerCount               ; outerCount = w: outer loop counter
44 loadn
45 movlw 0xff                     ; Set w = 255 decimal
46 movwf innerCount              ; innerCount = w: inner loop counter
47 decl
48 decfsz innerCount, f           ; outerCount--
49 goto decl                     ; If innerCount != 0 then repeat innerCount--
50 decfsz outerCount, f           ; Else decrement Count
51 goto loadn                    ; If outerCount != 0 then
52                               ; Reload innerCount to 255 and decrement
53                               ; Else, exit the subroutine
53 return                        ; End of the code
54 end

```

Figure 17: Code Task 2. Pt 1

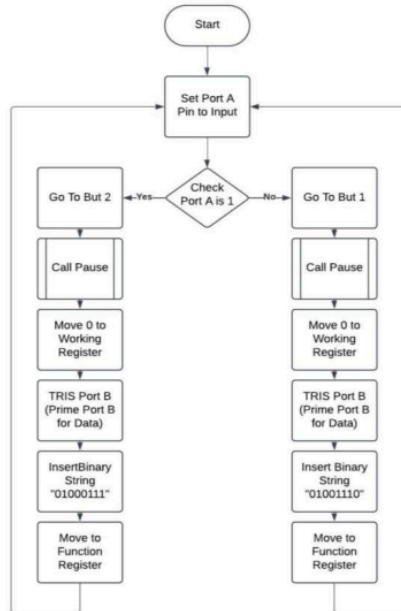


Figure 18: Code Task 2 Flow Chart

3.3. Coding Task 3:

```

4 ; Compiler: MPASM (Proteus)
5 #include p16f84a.inc           ; Include register definition file
6
7 portB    equ 0x06              ; Write 0xFF to PORTB (set all Port B pins high)
8 trisB    equ 0x06              ; Define TRISB register address
9
10 status  equ 0x03              ; Define STATUS register address
11 RP0     equ 5                  ; Bit 5 of status register
12 RP1     equ 6                  ; Bit 6 of status register
13
14 portA    equ 0x05              ; Port A address Links to the data sheet
15 trisA    equ 0x05              ; Tris A address in register bank
16
17 count    equ 0x0c              ; Counter 0 to 255
18 innerCount equ 0x0d            ; Pause routine inner loop counter
19 outerCount equ 0x0e            ; Pause routine outer loop counter
20
21 org 0                             ; Program memory Location 0
22 movlw 0x00                        ; Load 0x00 into W
23 tris portB                        ; Tri State register assigned to portB
24 movlw 0x01                        ; Set the port A pins
25 movwf trisA                      ; Set to input
26 bcf status,5                     ; Switch back to bank 0
27 goto But1                         ; Go to but1 line
28
29 But1: BTFSC portA,1               ; If portA bit 1 is 0 skip next instruction if 1 complete next instruction
30      goto But2                   ; Goes back to the top of the loop
31      call pause                   ; Call for delay
32      movlw b'00000000'           ; The Bit value to set each segment of the LED's to on
33      movwf portB                 ; Send to seven segment display
34      BTFSC portA,0               ; If portA bit 0 is 0 skip next instruction if 1 complete next instruction
35      goto But1                   ; Go to the top of the loop
36      call pause                   ; Call for a delay
37      movlw b'01111111'           ; Bit value to set each segment of LED's to off
38      movwf portB                 ; Send to seven segment display
39      goto Loop1                  ; Go to the Loop1
40
41      pause                       ; Call for pause
42      movlw 0xFF                  ; Set w = 255 decimal
43      movwf outerCount             ; OuterCount = w: outer loop counter
44      loadN                        ; Set w = 255 decimal
45      movlw 0xFF                  ; InnerCount = w: inner loop counter
46      decfsz innerCount, f         ; InnerCount--
47      goto decimalN               ; If innerCount != 0 then repeat nCount--
48
49      decfsz outerCount, f         ; Else decrement Count
50      goto loadN                  ; If outerCount != 0 then
51      reload innerCount to 255 and decrement
52      ; Else exit subroutine
53      return                      ; End of the code
54 end

```

Figure 19: Code Task 3. Pt 1

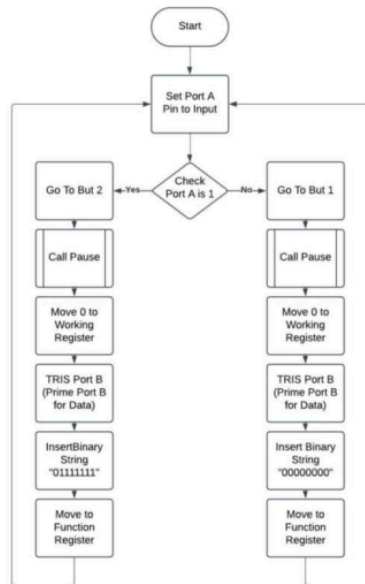


Figure 20: Code Task 3 Flow Chart

3.4. Coding Task 4:

```

4 #include p16f84a.inc ; Include register definition file
5
6 portB equ 0x06 ; port B address links to the data sheet
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 portA equ 0x05 ; port A address links to the data sheet
14 trisA equ 0x85 ; Tris A address in register bank
15
16 count equ 0x0C ; Variable to hold the count
17 innerCount equ 0x0D ; Pause routine inner loop counter
18 prevStateBut1 equ 0x0E ; Previous state for button 1
19 prevStateBut2 equ 0x0F ; Previous state for button 2
20
21 org 00
22 goto Start ; GoTo start of program
23 Start ; Start of program
24 ; Set PORTA as inputs for the buttons
25 bsf status, RP0 ; Select bank 1
26 movlw 0xFF ; Load W with 0xFF
27 movwf trisA ; Set PORTA as input
28 ; Set PORTB as output for the seven segment display
29 clrf trisB ; Clear TRISB, set all Port B pins as outputs
30 bcf status, RP0 ; Set RP0 to select Bank 1
31 clrf prevStateBut1 ; Initialize previous button state to 0
32 clrf prevStateBut2 ; Initialize previous button state to 0
33
34 ; Start the display with number '5' and start count from 5
35 movlw b'00010010 ; Binary for displaying digit '5' on display
36 movwf portB ; Send initial display to PORTB
37 movlw 0x05 ; Initialize count at 5
38 movwf count
39
40 mainLoop
41 ; Button 1 press to count up
42 btfsf portA, 0 ; Check if button 1 is pressed
43 goto checkReleaseBut1 ; If pressed check for release
44 clrf prevStateBut1 ; Clear previous state if not pressed
45 call pause
46
47 ; Button 2 press to count down
48 btfsf portA, 1 ; Check if button 2 is pressed
49 goto checkReleaseBut2 ; If pressed check for release
50 clrf prevStateBut2 ; Clear previous state if not pressed
51 call pause
52
53 goto mainLoop ; Repeat the Loop

```

Figure 21: Code Task 4. Pt 1

```

53 goto mainLoop ; Repeat the Loop
54
55 ; Button 1 press release count up
56 checkReleaseBut1
57 movf prevStateBut1, w ; Load previous state for button 1
58 btfsf status, Z ; If previous state was 0 (button not pressed) continue
59 call count_up_increment ; Call count up routine
60 movlw 0x01 ; Load W with 1
61 movwf prevStateBut1 ; Update previous state to 1 (button pressed)
62
63 goto mainLoop
64 ; Button 2 press release count down
65 checkReleaseBut2
66 movf prevStateBut2, w ; Load previous state for button 2
67 btfsf status, Z ; If previous state was 0 (button not pressed) continue
68 call count_down_increment ; Call count down routine
69 movlw 0x01 ; Load W with 1
70 movwf prevStateBut2 ; Update previous state to 1 (button pressed)
71
72 goto mainLoop
73 ; Count up increment
74 count_up_increment
75 movf count, w ; Load current count into W
76 sublw 0x09 ; Check if count is 9
77 btfsf status, Z ; If count is 9 skip increment
78 return ; Return
79 incf count, f ; Increment count
80 call display ; Update display
81 return
82 ; Count down increment
83 count_down_increment
84 movf count, w ; Load current count into W
85 btfsf status, Z ; If count is 0 skip decrement
86 return ; Return
87 decf count, f ; Decrement count
88 call display ; Refresh the display
89 return
90 ; Display the value loop
91 display
92 movf count, w ; Move count value to W
93 movwf portB ; Send to display on PORTB
94 call pause ; Calls for a pause
95 call check ; Calls for the check loop to display the correct number/value
96 return
97 ; Check if the count value is 0;
98 check
99 movf 0x0C, w ; Location memory of count to load into w
100 movlw 0x00 ; Load w with 0
101 subwf 0x0C, w ; Subtract 0 from count value
102 btfsf status, Z ; If the zero flag is set skip the next line, else continue
103 goto check_count_1 ; Check next value

```

Figure 22: Code Task 4. Pt 2

```

103 goto check_count_1 ; Check next value
104 movlw b'01000000' ; Load the correct value for display
105 movwf portB ; Send to the seven seg display
106 return ; Return
107 ; Check if count value is 1;
108 check_count_1
109 movlw 0x01 ; Load w with 1
110 subwf 0x0C, w ; Subtract 1 from count value
111 btfsf status, Z ; If the zero flag is set skip next line else continue
112 goto check_count_2 ; Check the next value
113 movlw b'01111001' ; Load correct value for the display
114 movwf portB ; Send value to the display
115 return ; Return
116 ; Check if count value is 2;
117 check_count_2
118 movlw 0x02 ; Load w with 2
119 subwf 0x0C, w ; Subtract 2 from count value
120 btfsf status, Z ; If the zero flag is set skip next line else continue
121 goto check_count_3 ; Check the next value
122 movlw b'00100100' ; Load correct value for the display
123 movwf portB ; Send value to the display
124 return ; Return
125 ; Check if count value is 3;
126 check_count_3
127 movlw 0x03 ; Load w with 3
128 subwf 0x0C, w ; Subtract 3 from count value
129 btfsf status, Z ; If the zero flag is set skip next line else continue
130 goto check_count_4 ; Check the next value
131 movlw b'00110000' ; Load correct value for the display
132 movwf portB ; Send value to the display
133 return ; Return
134 ; Check if count value is 4;
135 check_count_4
136 movlw 0x04 ; Load w with 4
137 subwf 0x0C, w ; Subtract 4 from count value
138 btfsf status, Z ; If the zero flag is set skip next line else continue
139 goto check_count_5 ; Check the next value
140 movlw b'00011001' ; Load correct value for display
141 movwf portB ; Send value to the display
142 return ; Return
143 ; Check if count value is 5;
144 check_count_5
145 movlw 0x05 ; Load w with 5
146 subwf 0x0C, w ; Subtract 5 from count value
147 btfsf status, Z ; If the zero flag is set skip next line else continue
148 goto check_count_6 ; Check the next value
149 movlw b'00010010' ; Load correct value for the display
150 movwf portB ; Send value to the display
151 return ; Return
152 ; Check if count value is 6;
153 check_count_6

```

Figure 23: Code Task 4. Pt 3

```

145 movlw 0x05 ; Load w with 5
146 subwf 0x0C, w ; Subtract 5 from count value
147 btfsf status, Z ; If the zero flag is set skip next line else continue
148 goto check_count_6 ; Check the next value
149 movlw b'00010010' ; Load correct value for the display
150 movwf portB ; Send value to the display
151 return ; Return
152 ; Check if count value is 6;
153 check_count_6
154 movlw 0x06 ; Load w with 6
155 subwf 0x0C, w ; Subtract 6 from count value
156 btfsf status, Z ; If the zero flag is set skip next line else continue
157 goto check_count_7 ; Check the next value
158 movlw b'00000011' ; Load correct value for display
159 movwf portB ; Send value to display
160 return ; Return
161 ; Check if count value is 7;
162 check_count_7
163 movlw 0x07 ; Load w with 7
164 subwf 0x0C, w ; Subtract 7 from count value
165 btfsf status, Z ; If the zero flag is set skip next line else continue
166 goto check_count_8 ; Check the next value
167 movlw b'01111000' ; Load correct value for display
168 movwf portB ; Send value to display
169 return ; Return
170 ; Check if count value is 8;
171 check_count_8
172 movlw 0x08 ; Load w with 8
173 subwf 0x0C, w ; Subtract 8 from count value
174 btfsf status, Z ; If the zero flag is set skip next line else continue
175 goto check_count_9 ; Check the next value
176 movlw b'00000000' ; Load correct value for the display
177 movwf portB ; Send value to display
178 return ; Return
179 ; Check if count value is 9;
180 check_count_9
181 movlw 0x09 ; Load w with 9
182 subwf 0x0C, w ; Subtract 9 from count value
183 btfsf status, Z ; If the zero flag is set skip next line else continue
184 return ; Check the next value
185 movlw b'00010000' ; Load the correct value for display
186 movwf portB ; Send value to display
187 return ; Return
188 pause ; Pause in code
189 movlw 0xFF ; Load W with 255 for outer loop
190 movwf InnerCount ; Set the outer loop counter
191 delayLoop
192 decfsf InnerCount, f ; Decrement counter
193 goto delayLoop ; Repeat until InnerCount reaches 0
194 return ; Exit the pause
195 end ; End of code

```

Figure 24: Code Task 4. Pt 4



Figure 25: Code Task 4 Flow Chart

3.5. Coding Task 5:

```

3 #include p16f84a.inc      ; Include register definition file
4
5 portB equ 0x06           ; Define the address of PORTB register as 0x06
6 trisB equ 0x06           ; TRISB register (sets direction of Port B pins: 1 = input, 0 = output)
7
8 status equ 0x03         ; Status register
9 RP0 equ 5                ; Selects bit 5 of status register
10 RP1 equ 6                ; Selects bit 6 of status register
11
12 portA equ 0x05          ; Port A Data register
13 trisA equ 0x05          ; Tris A address in register bank
14
15 Delay EQU 0x20          ; Delay counter
16 org 00                  ; Set program start address to 0x00
17 goto Start              ; Go to the start
18 Start
19 bcf status,RP0          ; Set RP0 bit in STATUS register to 1, select Bank 1
20 movlw 0xFF              ; Load W register with the value 0xFF, all bits to 1
21 movwf trisA             ; Move the value in W to TRISA register, set all PortA pins to inputs
22 clrf trisB              ; Clear TRISB, set all Port B pins as outputs
23 bcf status,RP0          ; Clear RP0 bit in STATUS register, select Bank 0
24
25 main_loop
26 movlw b'01111111'      ; Blank/clear the display
27 movwf portB             ; Move W to port B to update the display
28 btfsf portA,0           ; Check if button is pressed
29 goto turn_sound_off     ; If button not pressed turn sound off
30 call turn_sound_on      ; If button pressed turn sound on
31 goto main_loop          ; Repeat the loop
32
33 turn_sound_on
34 bsf portB,7             ; Set speaker pin to high
35 call pause              ; Call for a pause within code
36 bcf portB,7             ; Set speaker pin to low
37 call pause              ; Call pause for a delay in code
38 return
39
40 turn_sound_off
41 bcf portB,7             ; Turn the speaker/buzzer off
42 goto main_loop          ; Goto the main loop
43 pause                   ; Call for a Pause within code
44 movlw 0xFF              ; Load delay counter
45 movwf Delay             ; Move Delay to F
46 delay_loop
47 decfsz Delay,F          ; Call for a delay in the loop
48 goto delay_loop         ; Decrement delay counter
49 return                  ; Repeat until number is zero
50 end                     ; End of the code

```

Figure 26: Code Task 5. Pt 1

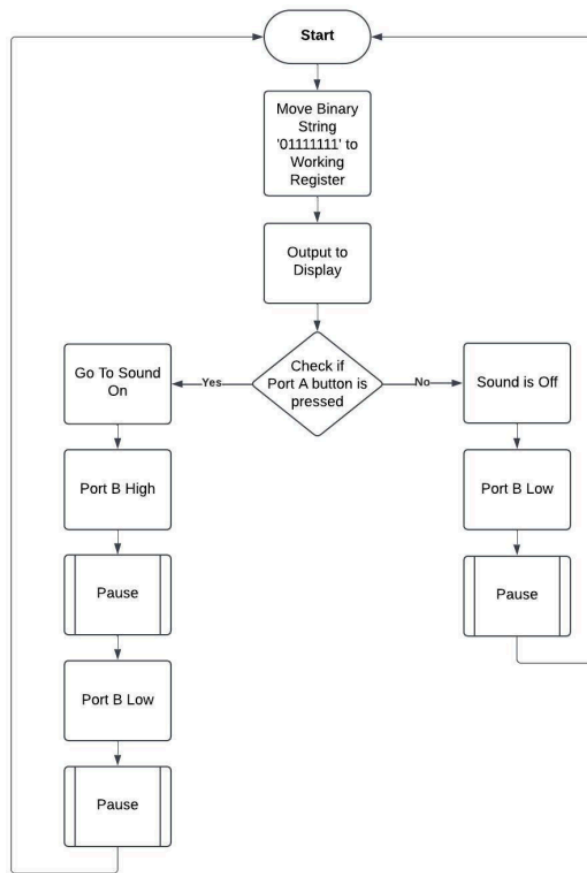


Figure 27: Code Task 5 Flow Chart

4.0.Conclusion:

Within this report, the author has explained the components included in the circuit given to a point where they should be easy to understand for the reader as well as links to the relevant data sheets to confirm certain details of a component, whether this be the exact resistance value etc or max power rating. Within Task B, the author then went through the code for the count circuit of how it counts from 0-9 and A-F on the seven-segment display, as required by the coding task. Tables of the bitwise calculations have been included for the reader to follow and understand where the binary values had come from which were used in the code to display each value. Within task 3, there are the screenshots of the code up to task 5 with the corresponding flow chart to complement the code. The code is commented which should enable the reader to understand what the code is doing and its process via the flowcharts.

References

- [1] MICROCHIP, "PIC16F84A Data Sheet," MICROCHIP Technology Inc., [Online]. Available: <file:///C:/ProgramData/Labcenter%20Electronics/Proteus%208%20Professional/Datasheets/pdfs/pic16/pic16f84a.pdf>. [Accessed 14 Nov 2024].
- [2] A. Technologies, "HDSP-740x Series Data Sheet," 30 August 2010. [Online]. Available: <https://docs.rs-online.com/0100/0900766b8151e3c8.pdf>. [Accessed 22 November 2024].
- [3] P. Kane, "Working with Seven Segment LED Displays," Jameco, [Online]. Available: <https://www.jameco.com/Jameco/workshop/TechTip/working-with-seven-segment-displays.html>. [Accessed 01 January 2025].
- [4] COMPONENTS 101, "SPST Momentary Push Button," COMPONENTS 101, [Online]. Available: <https://components101.com/switches/spst-momentary-push-button>. [Accessed 01 January 2025].
- [5] R. Wiens, "What Is a Capacitor and What Are the Different Types?," DigiKey, 02 June 2023. [Online]. Available: <https://www.digikey.co.uk/en/articles/what-is-a-capacitor-and-what-are-the-different-types#:~:text=Electrolytic%20capacitors%20use%20a%20dielectric,mechanical%20processes%20and%20are%20not.> [Accessed 05 January 2025].
- [6] Made-In-China, "High Quality Non Polarized Np Electrolytic Capacitor," [Online]. Available: <https://novatech.en.made-in-china.com/product/yafUQJOETmWk/China-High-Quality-Non-Polarized-Np-Electrolytic-Capacitor.html>. [Accessed 06 January 2025].
- [7] E. Bureau, "7805 Voltage Regulator IC Pinout, Circuit, and Applications," 11 June 2024. [Online]. Available: <https://www.electronicsforu.com/technology-trends/learn-electronics/7805-ic-voltage-regulator>. [Accessed 05 January 2025].
- [8] "Crystal Oscillator," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Crystal_oscillator. [Accessed 03 January 2025].
- [9] A. Satheesh, "Crystal Oscillator: Frequency, Circuit and Working," 13 June 2023. [Online]. Available: <https://solderingmind.com/crystal-oscillator-frequency-circuit-and-working/>. [Accessed 03 January 2025].
- [1] Cadence PCB Solutions, "Crystal Oscillator Frequency Ranges and Applications," Cadence, [Online]. Available: <https://resources.pcb.cadence.com/blog/2024-crystal-oscillator-frequency-ranges-and-applications>. [Accessed 04 January 2025].
- [1] B. Rose, "Buzzer Basics - Technologies, Tones, and Drive Circuits," same sky, [Online]. Available: <https://www.sameskydevices.com/blog/buzzer-basics-technologies-tones-and-driving-circuits#:~:text=When%20a%20voltage%20is%20applied,the%20speaker%20cone%20mentione d%20above..> [Accessed 06 January 2025].

- [1] SplashLearn, "Convert Hexadecimal To Binary – Definition, Table, Examples, FAQs," Splash
- 2] Learn, [Online]. Available: <https://www.splashlearn.com/math-vocabulary/hexadecimal-to-binary>. [Accessed 02 January 2025].

- [1] Microchip, "PIC16F84A Data Sheet," [Online]. Available:
- 3] <https://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf>. [Accessed 27 December 2024].

Robert Webb 2308625 Micros Peripherals and Interfaces Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

69/100

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

GRADING FORM: GJ_2019

BOBBY WEBB**346****ENGAGEMENT WITH LITERATURE**


A good range of research sources used in component identification and operation description.
IEEE referencing well implemented.

75**KNOWLEDGE AND UNDERSTANDING**


Excellent explanation of the components functions and operations.

78**COGNITIVE AND INTELLECTUAL**


This part is the weakest, as the code deploys a very 'brute force' approach, and there are a few logical errors,

50**PRACTICAL APPLICATION**



Well created circuit, and while the coding could be improved (and some of the flowcharting doesn't really match up exactly with what is happening!), a reasonable attempt at solving the majority of the tasks.

65

TRANSFERABLE SKILLS



Very professionally written and presented.

78