

Assignment submission

by Ben Scott

Submission date: 13-Jan-2025 10:27PM (UTC+0000)

Submission ID: 248340236

File name: 51445_Ben_Scott_Assignment_submission_621295_1621908714.docx (431.96K)

Word count: 2289

Character count: 12428

Using and programming the 7-segment display prototype
circuit.

L4 - Micros Peripherals and Interfaces (EES version)
(ACDA4004)

BEng Embedded Electronics

Benjamin Scott (2306925)
2306925@student.uwtsd.ac.uk

Table 1 - Binary and bitwise calculations	9
Figure 1 - HND/BENG PIC16F84 Assignment Board	3
Figure 2 – capacitor	4
Figure 3 - crystal oscillator:	5
Figure 4 – PIC16F84A	5
Figure 5 – Resistor.....	6
Figure 6 – button	6
Figure 7 – 7 segment display	7
Figure 8 – power	7
Figure 9 - ground	8
Figure 10 - sounder.....	8
Figure 11- 7 segment display (Enable Education , n.d.)	9

Contents

Section 1	3
Explanation of different components inside assignment circuit.....	3-9
Section 2.....	9
Explanation of 7 segment display.....	9
Table of binary and bitwise calculations.....	10
Section 3	10
Flowcharts of explanation of code.....	11-14
Bibliography	15
Abstract	16

Section 1:

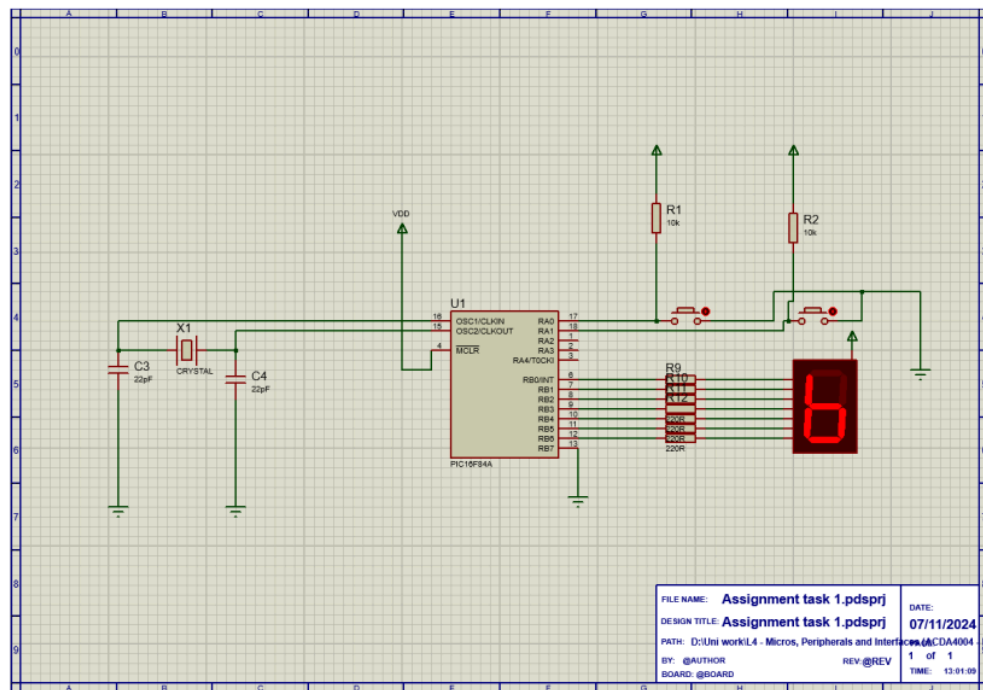


Figure 1 - HND/BENG PIC16F84 Assignment Board

PIC16F84 - Count circuit:

In the above figure is the count project for one 7 segment display. when the code is run, the count circuit will start running on the code written on the PIC. The code will output the numbers 1 – 9 on the 7-segment display by setting the corresponding pins high and low – see section 2 for explanation of the 7-segment display and how the numbers are displayed on the screen.

The components in the circuit are as follows:

capacitor:



Figure 2 – capacitor

A capacitor is a small electronic component that stores electrical energy. It consists of two metal plates separated by an insulating material, known as the dielectric. When electricity flows into the capacitor, it stores energy and can later release it when needed. The amount of energy a capacitor can store is called capacitance, measured in farads or F. Capacitors have a voltage rating, which indicates the maximum voltage they can safely handle. In the circuit the capacitor is paired next to a crystal oscillator (see next explanation). The capacitor works with the oscillator and makes sure that it works at the intended frequency. The capacitors control and stabilize the oscillations to ensure that there isn't any feedback and it gets rid of any noise. (Pollette, n.d.)

crystal picaxe oscillator:

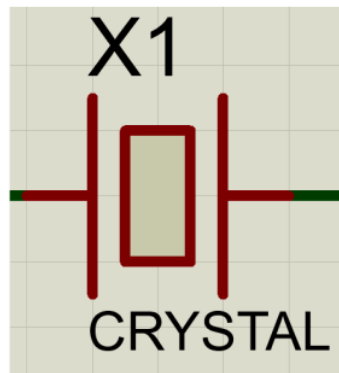


Figure 3 - crystal oscillator:

A crystal oscillator is a type of electronic oscillator that uses the mechanical resonance of a quartz crystal to create a highly stable and accurate frequency signal. These oscillators are crucial in applications where precise timing is required, such as in microcontrollers. A crystal oscillator uses a quartz crystal to create a stable frequency. When electricity is applied to the crystal, it vibrates at a specific frequency, which is used to generate a continuous oscillating signal. The oscillator circuit amplifies these vibrations to keep the signal going. Capacitors are added to help fine-tune the frequency and keep the oscillation stable. This steady signal is then used as a clock for devices like microcontrollers, ensuring accurate timing and synchronization. Crystal oscillators are widely used in many electronic devices because they provide reliable and precise frequency control. (Electrical4U, 2024)

PIC16F84A:

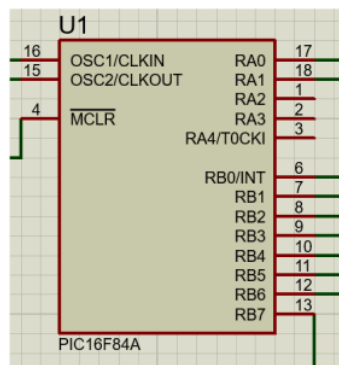


Figure 4 – PIC16F84A

The PIC16F84 is a small, low-cost microcontroller from Microchip Technology, used in simple electronic projects. It has 1 KB of memory for programs, 68 bytes of RAM for temporary data, and 64 bytes of EEPROM for saving data even when the power is off. It has 18 pins that can be used for input or output, and it works with an external crystal for timing, usually running up to 20 MHz. The microcontroller is low power, making it ideal for battery-powered devices. It's easy to program and widely used in hobby projects, basic control systems, and educational applications. The MCLR is Used for resetting the microcontroller, Pins RA0 to RA4: These can be used as analogue inputs or

digital I/O pins. Pins RB0 to RB7: Primarily for digital I/O, with special functions like external interrupts (RB0) or programming control. (MicrocontrollersLAB, 2025)

resistors:

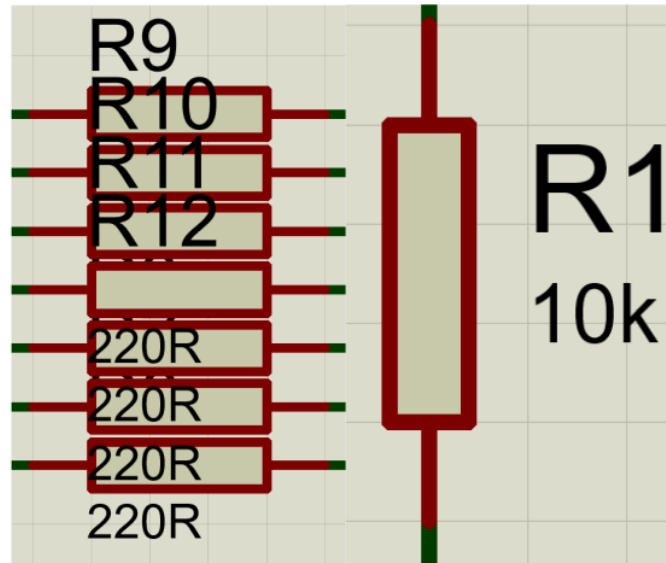


Figure 5 – Resistor

A resistor is an electrical component that limits the flow of electric current in a circuit. They add resistance to the current, which means it makes it harder for the current to pass through. The resistance is measured in ohms and the resistor acts like a bottleneck in an electrical circuit. It controls how much current can flow, helping to protect components from receiving too much current, ensuring proper voltage levels. The resistor is often used to limit the current flowing to LEDs, which prevents them from burning out. The resistor is used in this circuit to protect the 7-segment display from any over current which allows the display to not burn out or get damaged. The resistors are also used in the button section of the circuit to make sure that there isn't too much current flowing when the button is pressed. When the button is pressed, current will flow and the resistor makes sure that the circuit isn't over charged and damaged. (Braza, n.d.)

button:



Figure 6 – button

A button is a simple electrical component used to make or break a connection in a circuit. It typically consists of a switch mechanism that opens or closes the circuit when pressed. When the button is not pressed, the circuit is open, meaning there is no electrical connection, and no current can flow. However, when the button is pressed, the circuit becomes closed, allowing electrical current to flow through it and complete the circuit.

In many applications, a button can be used to trigger an action, like turning on a light, starting a motor, or sending a signal to a microcontroller. For example, in a microcontroller circuit, pressing a button can send a signal to the microcontroller, which then responds by executing a task. Buttons can be either temporary (only active while pressed) or latching (stays in an on or off state after being pressed once). In the circuit, there are two buttons that give high and low outputs. The buttons trigger the PIC to go through the code and act upon the different commands. (jimblom, n.d.)

7 segment display:

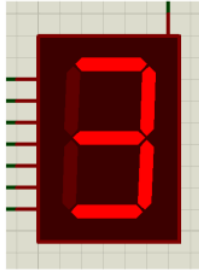


Figure 7 – 7 segment display

A 7-segment display is a type of display that shows numbers by lighting up different segments. It has 7 LED segments arranged in a figure-8 shape. Each segment can be turned on or off to form numbers (0-9). For example, to show the number 8, all the segments are turned on, and to show the number 1, only two segments (on the right) light up. Bits are sent to the display which turns on and off different LEDs on the display to be able to display information such as the temperature in a car or the time on a clock. The bits that are sent to the different pins correlate to different LEDs on the screen. Turning on and off the different LEDs at the same time allow for different information to be displayed such as numbers or letters. (ElectronicsTutorial, 2025) See section 2.

power sources:

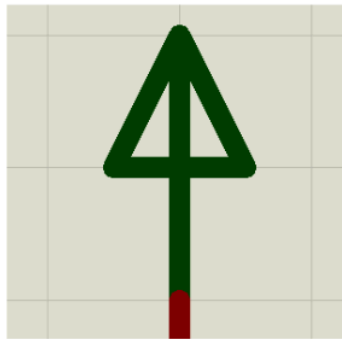


Figure 8 – power

A power source provides electrical energy to a circuit by supplying voltage and current. It pushes electrical current through the circuit, allowing components like LEDs or motors to function. The power source connects to the circuit at two points: positive and negative terminals. When the circuit is closed, the current flows through the components, which use the energy for tasks like lighting up or performing calculations. The current then returns to the power source, completing the cycle. (Advanced Conversion technology, 2025)

ground components:

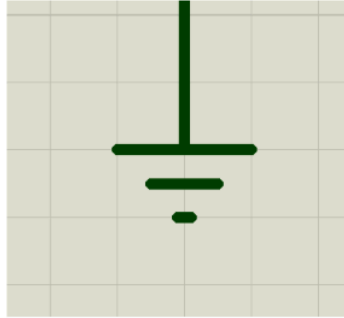


Figure 9 - ground

In an electrical circuit, ground serves as a common reference point for all components, essentially providing a return path for electric current. It helps to complete the circuit by offering a low-resistance path for current to flow back to the source. (Dahl, 2025)

Sounder

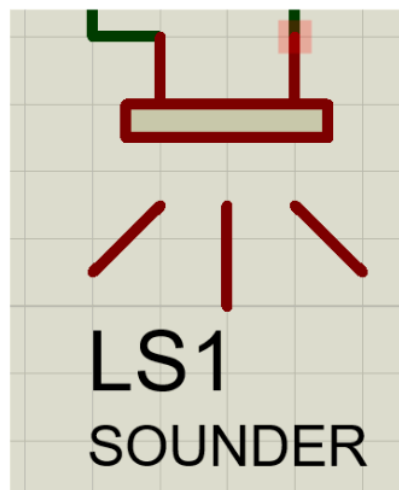


Figure 10 - sounder

In an electrical circuit, the sounder converts an electric signal into an audible signal. There is a coil of wire around a metal reed which when it has current flowing through it will generate an

electromagnetic field which attracts and repels the ‘reed’ inducing a sound. The tone that the sounder ‘sounds’ at is correlated to the frequency that the reed is attracted and repelled. (RS, 2024) The faster or slower is moved will change the tone or sound of the component. The sounder can be used in things like alarm systems for alerting workers or people of dangers. (pola, 2024)

Section 2:

To display the Hexadecimal characters 0-F there needs to be an understanding that the pins on the display are as a standard ‘high’. This means that the switch activation for the different ports of the display need to be low. As there is an inverted function, the pin must go low for the LED to go ‘high’ on the display.

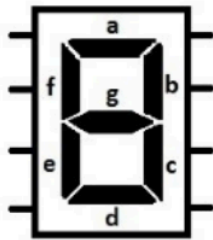


Figure 11- 7 segment display (Enable Education , n.d.)

Figure 2 displays the identifiers of the different lights on the 7-segment display. When an output is sent to each of the different pins, the corresponding light on the display will light up.

Below is the bitwise calculation to turn each of the pin on and off for the characters 0-9 and A-F

What is displayed?	What pins need turning on?	What is the binary?	What is the hexadecimal?
1	B, C	0111 1001	79
2	A, B, G, E, D	0010 0100	24
3	A, B, G, C, D	0011 0000	30
4	F, G, B, C	0001 1001	19
5	A, F, G, C, D	0001 0010	12
6	A, F, G, C, D, E	0000 0010	02
7	A, B, C	0111 1000	78
8	A, B, C, D, E, F, G	0000 0000	00
9	A, B, C, D, F, G	0001 0000	10
0	A, B, C, D, E, F	0100 0000	40
A	A, B, C, E, F, G	0000 1000	08
B	F, G, C, D, E	0000 0011	03
C	A, F, E, D	0100 0110	46
D	B, C, D, E, G	0010 0001	21
E	A, F, G, E, D	0000 0110	06
F	A, F, G, E	0000 1110	0E

Table 1 - Binary and bitwise calculations

To convert the binary to hexadecimal, take the first 4 bits and convert them into a number.
i.e. 1011

8421

1001 = 9

Because this is saying there is one 8, zero 4 or 2s and one 1 which sum to 11.

This process is then repeated for the next 4 bits.

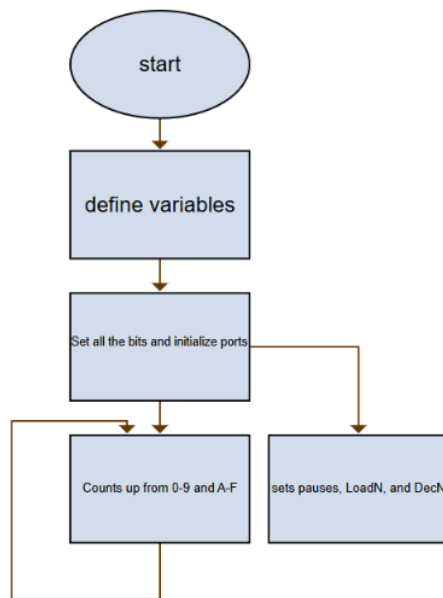
If the 4 bits sum to a number above 9 such as 11 the number is converted into an alphabetic character.

For the example 11 the hex character is B so 01001011 would be 4B

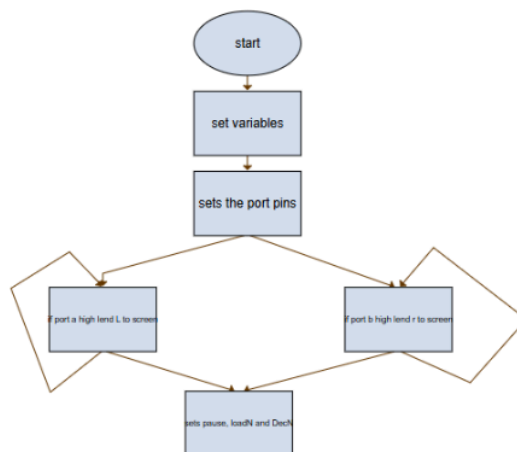
(BinaryHex Converter, n.d.)

Section 3:

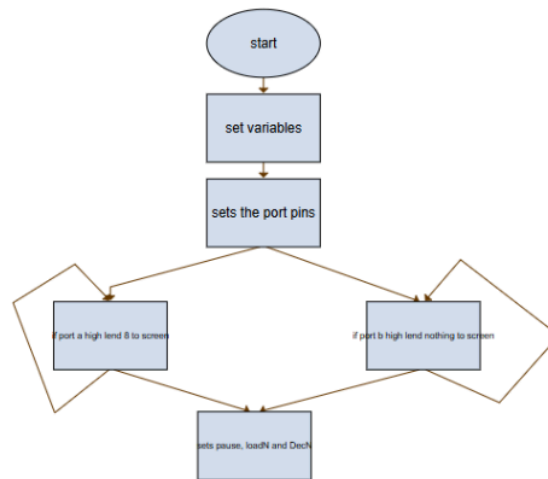
1. Modify "Count.asm" so that the seven-segment display counts from 0 to F in hexadecimal numbers in 0.2 second increments.



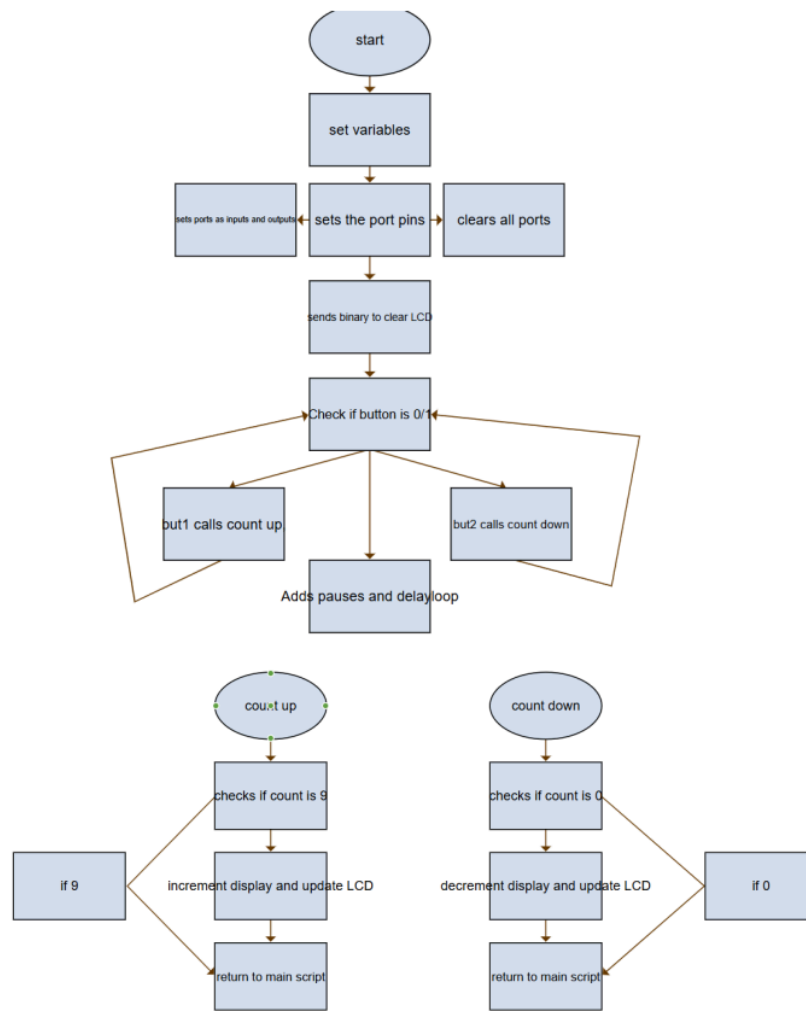
2. Write a program so that when you press one of the two buttons, it displays "L" on the 7 segment LED display, then when you press the other one, it displays "r".



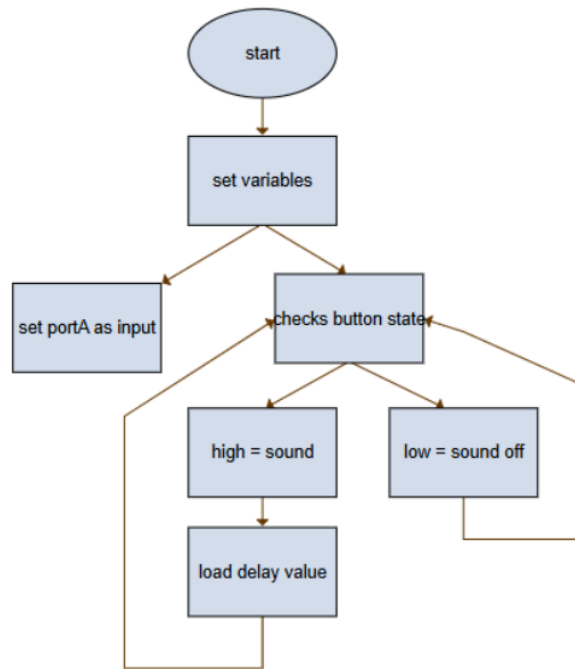
3. Write another program, so one button causes all of the LED segments to light, and the other switch turns them off. (the switch activations should be 'latching' i.e. you don't have to hold the button down for the on/off effect to remain)



4. Then combine all of these techniques together, by writing a new program where you start by displaying '5' but you can count up (to a maximum of 9) if you press one button, and count down (to a minimum of 0) if you press the other.



5. Write a new program so that the piezo sounder produces a tone while one of the switches/buttons is pressed. As you are using a simulation program and not the actual hardware, see if you can get Proteus to output the virtual speaker/sounder to the actual PC speaker, but if this fails, attach a virtual oscilloscope to the speaker/sounder and monitor the waveform visually.



All flow charts were created using the free online software 'eyed Live' (yEd Live, 2025)

To conclude, this assignment displays all the different components in the assignment circuit and provides an in-depth explanation of what the component does and what it does inside the circuit. It explains what a 7-segment display is and the calculation to turn each of the pins on and off for the characters 0-9 and A-F. In the final section, the author had to write code to make the display count from 0-9 with set increments and making sure all code worked and bitwise calculations were correct. The author has created flowcharts that easily explain the code they have written and have added clear and descriptive comments that explain what the code does and how it should work.

Bibliography

- Advanced Conversion technology*. (2025, January 12). Retrieved from What Is a Power Supply, and How Does It Work?: <https://www.actpower.com/blog/what-is-a-power-supply-and-how-does-it-work/>
- BinaryHex Converter*. (n.d.). Retrieved December 28, 2024, from Binary to hexadecimal converter: <https://www.binaryhexconverter.com/binary-to-hex-converter>
- Braza, J. (n.d.). *Circuit Basics*. Retrieved from How Resistors Work: <https://www.circuitbasics.com/what-is-a-resistor/>
- Dahl, Ø. N. (2025, January 13). *buildelectroniccircuits*. Retrieved from What is Ground in Electronic Circuits?: <https://www.build-electronic-circuits.com/what-is-ground/>
- Electrical4U. (2024, June 24th). *Electrical4U*. Retrieved from Crystal Oscillator: Circuit, Frequency & Working Principle: <https://www.electrical4u.com/crystal-oscillator/#:~:text=Working%20Principle%3A%20The%20oscillator%20works%20by%20applying%20an,mode%20%28low%20impedance%29%20or%20parallel-resonant%20mode%20%28high%20impedance%29>.
- ElectronicsTutorial*. (2025, January 10). Retrieved from 7-segment Display: <https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>
- Enable Education . (n.d.). *Education.ni*. Retrieved from Digital clock project: <https://education.ni.com/teach/resources/1211/digital-clock-project>
- jimblom. (n.d.). *Sparkfun*. Retrieved January 6, 2025, from Button and Switch Basics: <https://learn.sparkfun.com/tutorials/button-and-switch-basics/all>
- MicrocontrollersLAB. (2025, January 6). *MicrocontrollersLAB*. Retrieved from PIC16F84A Microcontroller introduction and features: <https://microcontrollerslab.com/pic16f84a-microcontroller-introduction-and-features/>
- pola, s. (2024, February 12th). *Learn Electronics India*. Retrieved from Anatomy of a Buzzer: Unveiling the Science Behind the Buzzer: <https://www.learnelectronicsindia.com/post/anatomy-of-a-buzzer-unveiling-the-science-behind-the-beep#:~:text=These%20buzzers%20rely%20on%20the%20classic%20electromagnetism%20principle.,reed%2C%20causing%20it%20to%20vibrate%20and%20produce%20sound>.
- Pollette, M. B. (n.d.). *How stuff works*. Retrieved January 4, 2025, from How capacitors work: <https://electronics.howstuffworks.com/capacitor.htm>
- RS. (2024, April 9). Retrieved from Understanding the Role of Electronic Sounders in Industrial Safety: <https://nz.rs-online.com/web/content/discovery/ideas-and-advice/electronic-sounders-guide>
- yEd Live*. (2025, January 4th). Retrieved from yworks: <https://www.yworks.com/>

Abstract:

```

;FILE: count.asm
;AUTHOR: Ben Scott
;DATE: 06/11/2024
;DESCRIPTION this code counts from 0-9 and then A-F and repeats

include p16f84.inc

;-----
; definition
portB equ 0x06 ;port B address links to the data sheet
trisB equ 0x06 ;trisB address in register bank 1
status equ 0x03 ;status register
RP1 equ 6 ;bit 6 of status register
RP0 equ 5 ;bit 5 of status register
;-----

;variables
count equ 0x0c ;counter 0 to 255
nCount equ 0x0d ;pause routine inner loop counter
mCount equ 0x0e ;pause routine outer loop counter
;-----

RST code 0x0 ;reset
goto start
;-----

PGM code

start
    bcf status,RP0 ;clear bank selection bits in the status reg
    bcf status,RP1
    bcf status,RP0 ;make bank 1
    movlw 0x00 ;load w with 0x00
    movwf trisB ;load it into trisb to make port b all outputs
    bcf status,RP0 ;reselect bank 0
    clrf portB ;clear all lines low
    clrf count ;initialize counter to 0
;-----

;the different numbers and letters to display
get_count
    movlw 0x000000 ;0
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x01110011 ;1
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00100100 ;2
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00100000 ;3
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00011001 ;4
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00010010 ;5
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00000111 ;6
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x01111000 ;7
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00000000 ;8
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00010000 ;9
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00001000 ;A
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00000011 ;B
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x01000110 ;C
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00100001 ;D
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00000110 ;E
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    movlw 0x00001110 ;F
    movwf portB ;move W to port B to update the display
    call pause ;delay 200ms
    goto get_count ;cycles forever!!
;-----

pause
    movlw 0xff ;set w = 255 decimal
    movwf mCount ;mCount = w; outer loop counter
loadN
    movlw 0xff ;set w = 255 decimal
    movwf nCount ;nCount = w; inner loop counter
decN
    decfsz nCount,f ;if nCount != 0 then repeat nCount--
    goto decN
    decfsz mCount,f ;else decrement Count
    goto loadN ;if mCount != 0 then reload nCount to 255 and decrement
    return ;xx it subroutine
;-----

end

```

```
;FILE:          count.asm
;AUTHOR:        Ben Scott
;DATE:          06/11/2024
;DESCRIPTION
;this code counts from 0-9 and then A-F and repeats
```

```
include p16f84.inc
```

```
;-----
;              ; definition
```

```
portB  equ 0x06      ;port B address
trisB  equ 0x06      ;trisB address - register bank 1
status equ 0x03      ;status register
RP0    equ 5         ;bit 5 of status register
RP1    equ 6         ;bit 6 of status register
portA   equ 0x05     ;port A address links to the data sheet. enables the inputs
trisA   equ 0x05     ;Tris A address in register bank
```

```
;-----
;              ;variables
```

```
count  equ 0x0c      ;counter 0 to 255
nCount equ 0x0d      ;pause routine inner loop counter
mCount equ 0x0e      ;pause routine outer loop counter
```

```
;-----
;              ;main code section
```

```
org 0

movlw 0x01          ;set the port A pins
movwf trisA         ;set to input
bcf status,5        ;switch back to bank 0
goto Button1        ;go to button1 line
Loop
Button1 BTFSC portA,1 ;if portA bit 1 is 0 skip next instruction if 1 complete next instruction
        goto Button2 ;returns to button2
call pause          ;pauses the code

movlw 0x00
tris portB
movlw b'01000111'   ;send l to screen
movwf portB         ;send to screen
Button2 BTFSC portA,0 ;if portA bit 1 is 0 skip next instruction if 1 complete next instruction
        goto Button1 ;goes back to the top of the loop
call pause          ;pauses the code

movlw 0x00
tris portB
movlw b'01001110'   ;send r to screen
movwf portB         ;send to display

goto Loop           ;go back to loop
```

```
;-----
;              ;pause
```

```
pause
        movlw 0xff          ;set w = 255 decimal
        movwf mCount        ;mCount = w: outer loop counter

loadN
        movlw 0xff          ;set w = 255 decimal
        movwf nCount        ;nCount = w: inner loop counter

decN
        decfsz nCount, f     ;nCount--
        goto    decN         ;if nCount != 0 then repeat nCount--

        decfsz mCount, f     ;else decrement Count
        goto    loadN        ;if mCount != 0 then reload nCount to 255 and decrement
        return              ;exit subroutine
end
```

```

;FILE:      count.asm
;AUTHOR:    Ben Scott
;DATE:      06/11/2024
;DESCRIPTION:
;when one of the buttons is pressed it either turns all the 7 segment displays on or off

```

```
include p16f84a.inc
```

```

;-----
; definition

```

```

portB equ 0x06      ;port B address
trisB equ 0x06      ;trisB address - register bank 1
status equ 0x03     ;status register
RP0 equ 5           ;bit 5 of status register
RP1 equ 6           ;bit 6 of status register
portA equ 0x05      ;port A address links to the data sheet. enables the inputs
trisA equ 0x05      ;Tris A adress in reigster bank

```

```

;-----
;variables

```

```

count equ 0x0c      ;counter 0 to 255
nCount equ 0x0d     ;pause routine inner loop counter
mCount equ 0x0e     ;pause routine outer loop counter

```

```

;-----
;main code section

```

```

org 0

movlw 0x01          ;set the port A pins
movwf trisA         ;set to input
bcf status,5        ;switch back to bank 0
goto Button1        ;go to button1 line
Loop
Button1 BTFSC portA,1 ;if portA bit 1 is 0 skip next insturction if 1 complete next insturction
      goto Button2    ;returns to button2
call pause          ;pauses the code

movlw 0x00
tris portB
movlw b'01111111'    ; send 8 to screen
movwf portB          ;send to screen
Button2 BTFSC portA,0 ;if portA bit 1 is 0 skip next insturction if 1 complete next insturction
      goto Button1    ;goes back to the top of the loop
call pause          ;pauses the code

movlw 0x00
tris portB
movlw b'00000000'    ;send blank to screen
movwf portB          ;send to display

goto Loop           ;go back to loop

```

```

;-----
pause
      movlw 0xff      ; set w = 255 decimal
      movwf mCount    ; mCount = w: outer loop counter

loadN
      movlw 0xff      ; set w = 255 decimal
      movwf nCount    ; nCount = w: inner loop counter

decN
      decfsz nCount, f ; nCount--
      goto decN        ; if nCount != 0 then repeat nCount--
      decfsz mCount, f ; else decrement Count
      goto loadN       ; if mCount != 0 then reload nCount to 255 and decrement
      return          ; else exit

end

```

```
;FILE: count.asm
;AUTHOR: Ben Scott
;DATE: 06/11/2024
;DESCRIPTION:
;Starting at 5, when a button is pressed it increment up to and stop at 9. when the other is pressed it will stop at 0. it counts one number at a time.
```

```
include p16f84a.inc
```

```
;-----
; definition
```

```
portB equ 0x06 ;port B address links to the data sheet
trisB equ 0x06 ;trisB address in register bank 1
status equ 0x03 ;status register
RP0 equ 5 ;bit 5 of status register
RP1 equ 6 ;bit 6 of status register
portA equ 0x05 ;port A address links to the data sheet
trisA equ 0x05 ;tris A address in register bank
```

```
;-----
;variables
```

```
count equ 0x0C ;Variable to hold the count
nCount equ 0x0D ;Pause routine inner loop counter
Button1 equ 0x0E ;Previous state for button 1
Button2 equ 0x0F ;Previous state for button 2
```

```
;-----
;main code section
```

```
org 0
goto Start
Start:
; Setting PORTA as inputs for the buttons
bsf status, RP0 ; Select bank 1
movlw 0xFF
movwf trisA ; Set PORTA as input

; Set PORTB as output for the display
clrf trisB ; Set PORTB pins as output
bcf status, RP0 ; Switch back to bank 0

clrf Button1 ; Initialize previous button states to 0
clrf Button2

; Initialize display with '5' and start count at 5
movlw b'00010010' ; Binary for displaying 5
movwf portB ; Send initial display to PORTB
movlw 0x05 ; Initialize count at 5
movwf count

mainLoop:
;Button 1 (count up)
BTFSS portA, 0 ; Check if button 1 (RA0) is pressed
goto checkReleaseBut1 ; If pressed, check for release
clrf Button1 ; Clear previous state if not pressed
call pause

; Button 2 (count down)
BTFSS portA, 1 ; Check if button 2 (RA1) is pressed
goto checkReleaseBut2 ; If pressed, check for release
clrf Button2 ; Clear previous state if not pressed
call pause

goto mainLoop ; Repeat the loop
```

;Button 1 release detection (count up)

checkReleaseBut1:

```
    movf Button1, w          ; Load previous state for button 1
    btfsc status, Z          ; If previous state was 0 (not pressed), proceed
    call count_up            ; Call count up routine
    movlw 0x01
    movwf Button1            ; Update previous state to 1 (pressed)
```

goto mainLoop

;Button 2 release detection (count down)

checkReleaseBut2:

```
    movf Button2, w          ; Load previous state for button 2
    btfsc status, Z          ; If previous state was 0 (not pressed), proceed
    call count_down          ; Call count down routine
    movlw 0x01
    movwf Button2            ; Update previous state to 1 (pressed)
```

goto mainLoop

;coding the numbers

count_up:

```
    movf count, w            ; Load current count into W
    sublw 0x09               ; Check if count is 9
    btfsc status, Z          ; If count is 9, skip increment
    return
    incf count, f            ; Increment count
    call display              ; Update display
    return
```

count_down:

```
    movf count, w            ; Load current count into W
    btfsc status, Z          ; If count is 0, skip decrement
    return
    decf count, f            ; Decrement count
    call display              ; Update display
    return
```

display:

```
    movf count, w            ; Move count value to W
    movwf portB              ; Send to display on PORTB
    call pause
    call check                ;calls the check loop to display the correct number
    return
```

check:

```
    movf 0x0C, w             ;memory location of count load into w
    movlw 0x00               ;load w with 0
    subwf 0x0C, w            ;take 0 from count
    btfss status, Z          ;if the zero flag is set skip next line else continue
    goto count_1             ;check next value
    movlw b'01000000'        ;load value for display
    movwf portB              ;send to display
    return                   ;return
```

count_1:

```
    movlw 0x01               ;load w with 1
    subwf 0x0C, w            ;take 1 from count
    btfss status, Z          ;if the zero flag is set skip next line else continue
    goto count_2             ;check next value
    movlw b'01111001'        ;load value display
    movwf portB              ;send to display
    return                   ;return
```

```

count_2:
    movlw 0x02           ;load w with 2
    subwf 0x0C, w        ;take 2 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_3         ;check next value
    movlw b'00100100'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_3:
    movlw 0x03           ;load w with 3
    subwf 0x0C, w        ;take 3 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_4         ;check next value
    movlw b'00110000'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_4:
    movlw 0x04           ;load w with 4
    subwf 0x0C, w        ;take 4 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_5         ;check next value
    movlw b'00011001'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_5:
    movlw 0x05           ;load w with 5
    subwf 0x0C, w        ;take 5 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_6         ;check next value
    movlw b'00010010'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_6:
    movlw 0x06           ;load w with 6
    subwf 0x0C, w        ;take 6 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_7         ;check next value
    movlw b'00000011'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_7:
    movlw 0x07           ;load w with 7
    subwf 0x0C, w        ;take 7 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_8         ;check next value
    movlw b'01111000'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_8:
    movlw 0x08           ;load w with 8
    subwf 0x0C, w        ;take 8 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    goto count_9         ;check next value
    movlw b'00000000'    ;load value for display
    movwf portB          ;send to display
    return              ;return
count_9:
    movlw 0x09           ;load w with 9
    subwf 0x0C, w        ;take 9 from count
    btfss status, Z      ;if the zero flag is set skip next line else continue
    return              ;check next value
    movlw b'00010000'    ;load value for display
    movwf portB          ;send to display
    return              ;return

```

```
pause:
    movlw 0xFF          ; Load W with 255 for outer loop
    movwf nCount        ; Set outer loop counter
delayLoop:
    decfsz nCount, f    ; Decrement counter
    goto delayLoop      ; Repeat until nCount reaches 0
return                  ; Exit delay

end
```

```
;FILE:    count.asm
;AUTHOR:  Ben Scott
;DATE:    06/11/2024
;DESCRIPTION:
;This program uses a button connected to RA0 to control the output of a speaker connected to RB7.
;When the button is pressed, the program generates a sound. If not pressed, the speaker is turned off.
;The program operates in a loop, checking the button's state repeatedly.
```

```
-----
;definition
portB equ 0x06      ;port B address links to the data sheet
trisB equ 0x06      ;trisB address in register bank 1
status equ 0x03     ;status register
RP0 equ 5           ;bit 5 of status register
RP1 equ 6           ;bit 6 of status register
portA equ 0x05      ;port A address links to the data sheet
trisA equ 0x05      ;tris A address in register bank
;Define constants
delay equ 0x20      ;Delay counter for tone duration
butt equ 0          ;RA0 pin is used for the input button
speaker equ 7        ;RB7 pin is used for output to the speaker
-----
```

```
;Reset Vector: Jump to start of the program
org 0               ;Reset vector
goto start          ;Jump to start of the main program
-----
```

```
;Main program starts here
start:
; Set PORTA for input (RA0 is button, other pins as output)
bsf status, RP0     ;Select Bank 1
movlw 0x01          ;Set RA0 as input (1= input, 0= output)
movwf trisA         ;Store to trisA register
```

```
clrf trisB          ;Store to trisB register
bcf status, RP0     ;Return to Bank 0
```

```
main_loop:
;Check if button is pressed (RA0 = 0 means pressed)
movlw b'01111111'   ;Binary for clearing screen (all blank)
movwf portB          ;Send initial value to portB
btfsc portA, butt    ;Test if button (RA0) is pressed
goto sound_off       ;If not pressed, skip to sound off
call sound           ;If pressed, generate sound
goto main_loop       ;Repeat the loop again!
-----
```

```
;Subroutine to generate sound on the speaker
sound:
bsf portB, speaker   ;Set RB7 high to turn on the speaker
call delay_routine   ;Call delay for tone duration
bsf portB, speaker   ;Set RB7 low to turn off the speaker
call delay_routine   ;Call delay for tone duration
return               ;Return to the main loop
-----
```

```
;Subroutine to ensure the speaker is off
sound_off:
bcf portB, speaker   ;Set RB7 low to make sure speaker is off
goto main_loop       ;Return to the main loop
-----
```

```
;Delay routine to create a tone duration
delay_routine
movlw 0xff           ;Load delay counter with a value (max delay)
movwf delay          ;Store it to DELAY register
delay_loop:
decfsz delay, f      ;delay counter
goto delay_loop      ;Loop until DELAY reaches 0
return               ;Return to caller
```

```
;End of program
end
```



Assignment submission

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

61 /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

GRADING FORM: GJ_2019

BEN SCOTT

307

ENGAGEMENT WITH LITERATURE



A reasonable range of reference sources used - note however, you need to use the IEEE convention for your UWTSD work!

70

KNOWLEDGE AND UNDERSTANDING



A fair level of K&U, although the voltage reg doesn't seem to be in the circuit, neither is the sounder (and the description in the report is for the wrong type of buzzer)

62

COGNITIVE AND INTELLECTUAL

	Fairly conventional approach to solving the tasks, so low marks in this section which is for 'flair and innovation'	50
---	---	-----------

PRACTICAL APPLICATION

	Circuit missing the voltage reg and sounder, but the rest of it is there, and 5 of the tasks attempted well. Flowcharts a little 'unconventional' in places!	65
---	---	-----------

TRANSFERABLE SKILLS

	Fairly basic layout - could have spent more time in making a more professional looking document - see study skills notes!	60
---	---	-----------