

Micro, Peripherals & Interfaces Report (LS)

by Louis S

Submission date: 13-Jan-2025 10:38PM (UTC+0000)

Submission ID: 248341393

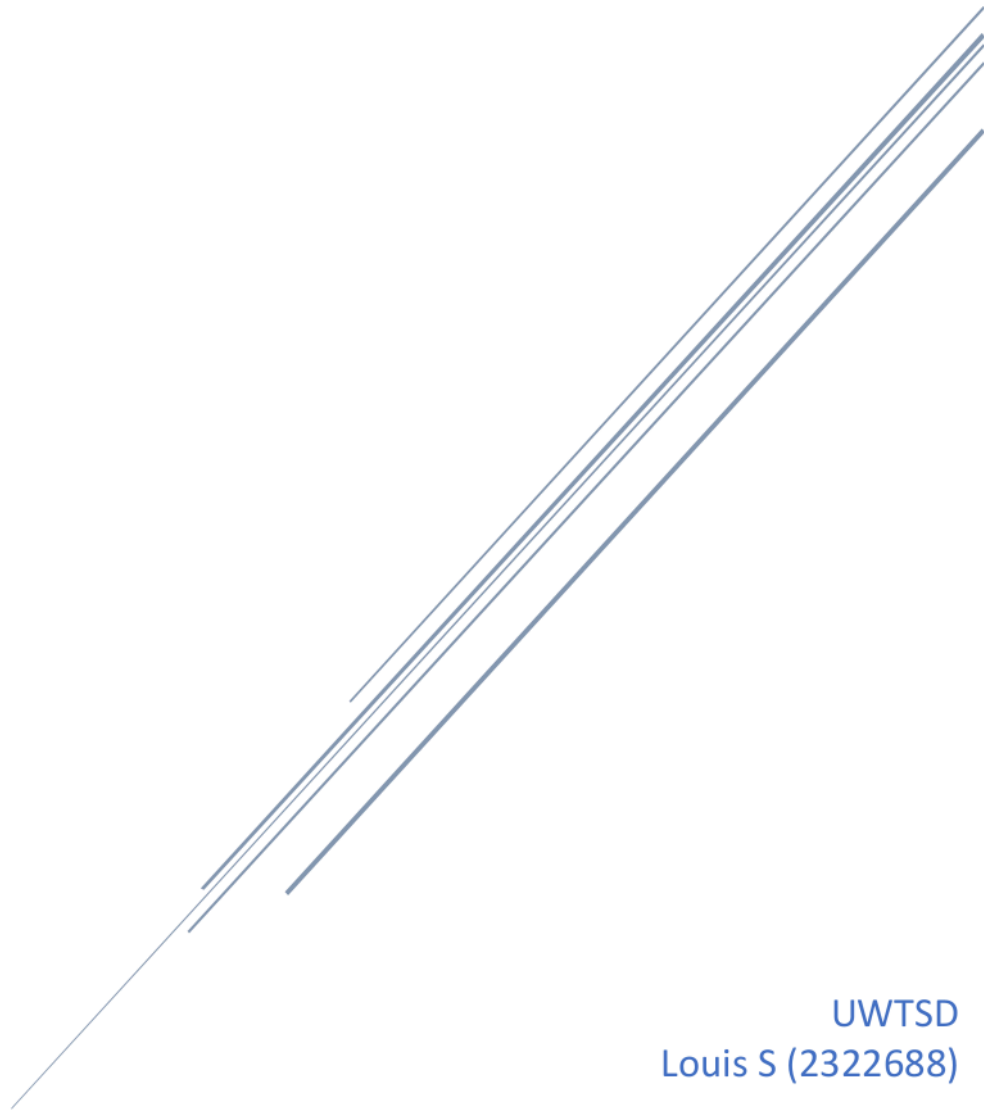
File name: 52258_Louis_S_Micro__Peripherals__Interfaces_Report_LS__621295_1491548029.docx (2.24M)

Word count: 4076

Character count: 18685

USING AND PROGRAMMING THE 7 SEGMENT DISPLAY PROTOTYPE CIRCUIT

Micros, Peripherals & Interfaces for Electronics (ACDA4004)
(EES Version)



UWTSD
Louis S (2322688)

Contents

Introduction	4
Section 1 – Circuit Operation	4
Circuit Diagram.....	4
Devices & Functions	4
PIC	4
7 Segment Display	6
Crystal.....	7
Resistors	7
Capacitors.....	8
Buttons.....	8
Switch	8
Buzzer/Sounder	9
Transistor.....	9
Connector.....	10
Voltage regulator.....	11
Circuit Modules	12
Button Inputs	12
Clock Inputs.....	12
Sounder	13
Power Input.....	13
PIC16F84A & 7 Segment Display	14
Section 2 – 7 Segment Display Operation	15
Section 3 – Coding Task.....	18
Coding Task 1 Flow Chart	19
Coding Task 2 Flow Chart	20
Coding Task 3 Flow Chart	21
Coding Task 4 Flow Chart	22
Coding Task 5 Flow Chart	23
Bibliography	23
Appendix.....	24
Task 1 Code	24

Task 2 Code 27

Task 3 Code 28

Task 4 Code 29

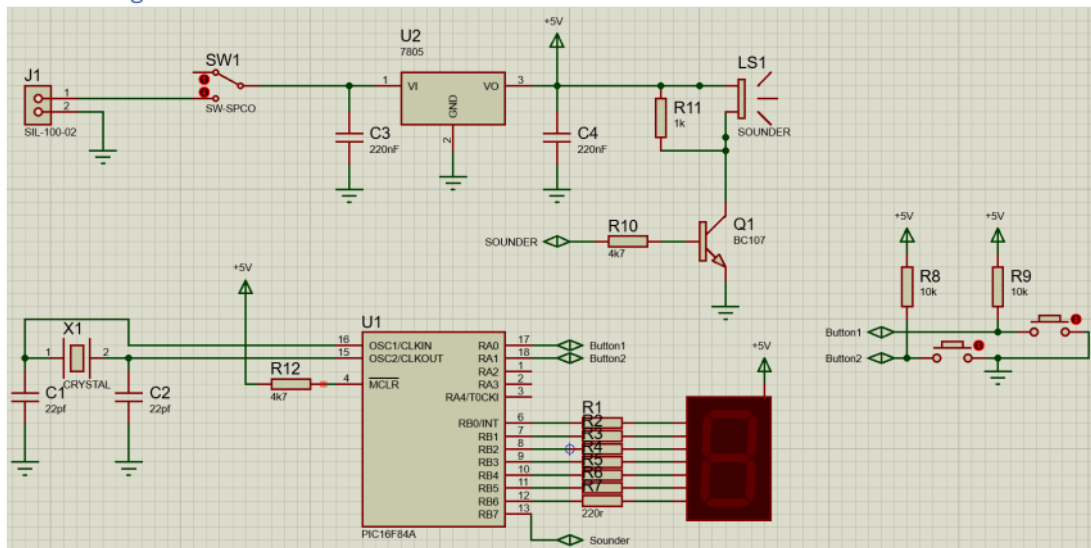
Task 5 Code 31

Introduction

This report contains the device and circuit operation for a PIC16F84A, programming of a 7 Segment-Display, as well as coding tasks as outlined by the report specification.

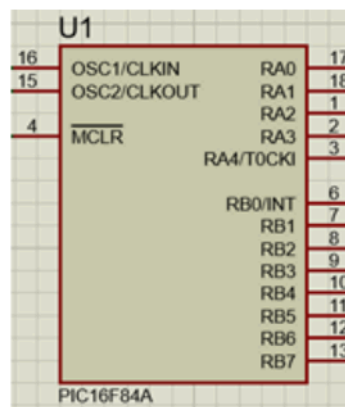
Section 1 – Circuit Operation

Circuit Diagram



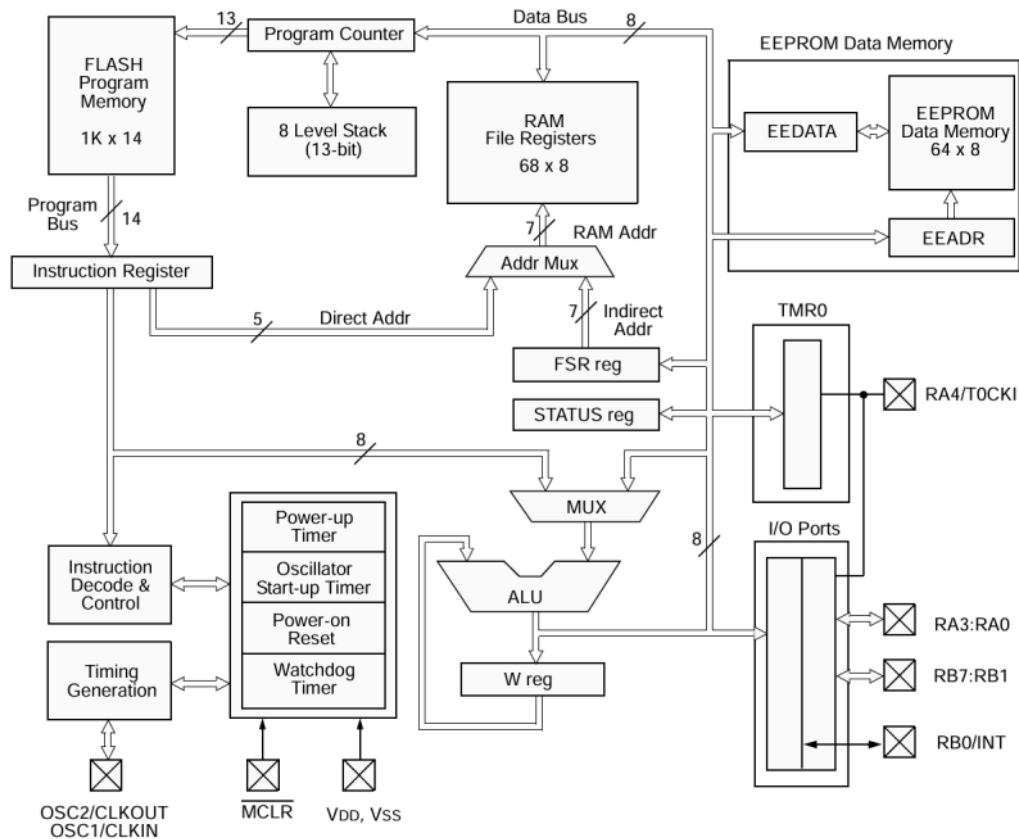
Devices & Functions

PIC



The PIC is a type of microcontroller, often used in embedded systems for its relative ease of use and versatile applications. The specific PIC16F84A has 35 instructions and is able to be programmed with languages such as C or Assembly.

Below shows the block diagram for the PIC16F84A.



[1]

Port A (RA0 – 4) and Port B (RB0 - 7) are the PIC16F84A's 13 I/O pins, meaning they can be configured as either inputs or outputs through the PIC's loaded program. They utilise TTL (Transistor Transistor Logic) shown below in the block diagrams for RA/B0 - 3

FIGURE 4-1: BLOCK DIAGRAM OF PINS RA3:RA0

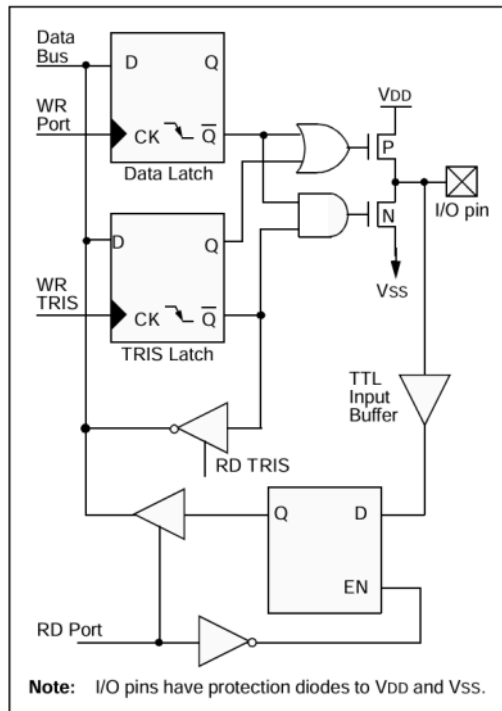
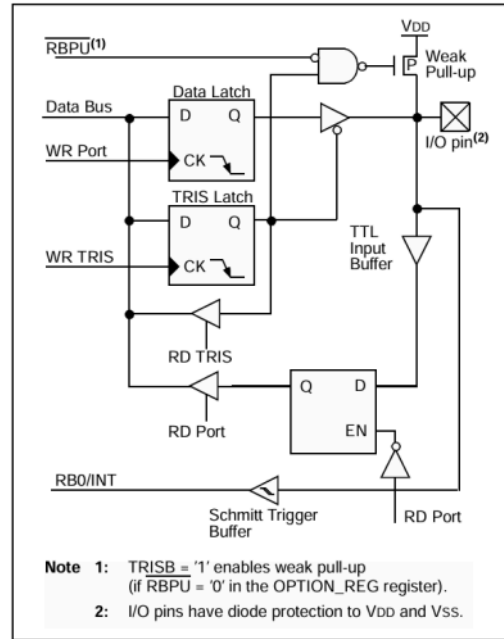


FIGURE 4-4: BLOCK DIAGRAM OF PINS RB3:RB0

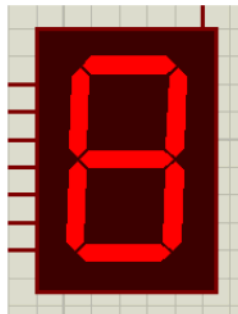


[1]

The PIC also contains a watchdog timer which is enabled/disabled via the program loaded onto the pic, a watchdog timer is a counter that will restart the device if the countdown is not reset, to ensure that program is not stuck in an infinite loop.

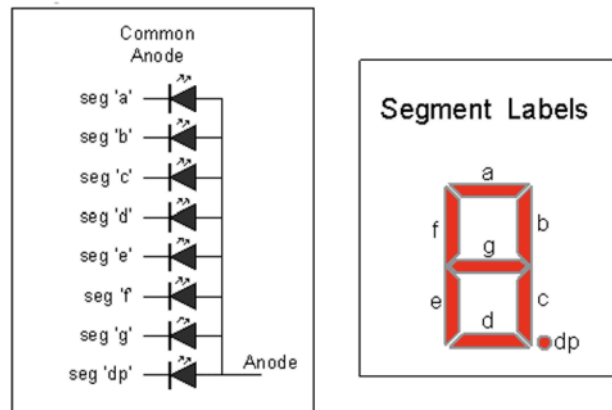
The MCLR Pin, when grounded and set low, will reset the PIC, thus needs to be held high during operation.

7 Segment Display



The seven-segment display is made up from 7 LEDs, with the device having 8 pins; 1 for supply, and 7 for the return path for each of the LEDs for this case by being a common anode, common cathode is essentially the same but inverted, with 7 supply pins for each of the LEDs and a single return pin.

Below is an example of the internals of a Common Anode 7 Segment-Display, similar to the one used in the simulation circuit, with the addition of a Decimal Point Pin.



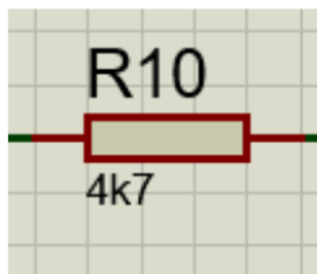
[2]

Crystal



Crystals, typically quartz, are used due to their piezoelectric effect which generate a charge when mechanically pressured or vibrates when an electrical signal is applied to it. With the frequency dependant on shape and dimension of the crystal, the latter effect can be used to produce precise waveforms that can be used for timing of microcontrollers, among other things.

Resistors



The resistors are used to limit current, and protect components from being damaged, by adding resistance to the path with the same voltage, the current passing through is lower.

$$I = \frac{V}{R}$$

V = Voltage (V), I = Current (A), R = Resistance (Ω)

Capacitors



Capacitors retain charge with the use of two internal conductive plates separated with a dielectric in the form of an electric potential.

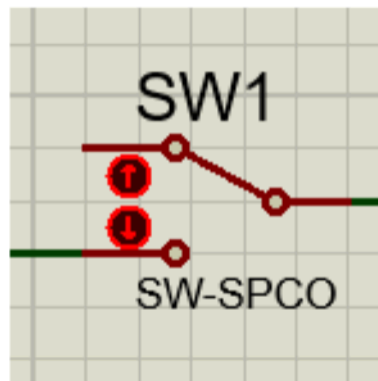
They have numerous uses such as filters for DC signals; use in RC circuits to create time delays; Voltage stabilisation to smooth out voltage spikes.

Buttons



A simple component that, in this case, creates an electrical connection when the button is depressed, returning to an open state when released with use of a spring. There are various types of buttons for differing applications, however, are similar in their operation.

Switch



The switch used in the circuit is a SPCO (Single Pole Change Over), meaning the switch has two states with a common contact on the left and two contacts on the left that the switch can alternate between

which is near identical to a SPDT (Single Pole Double Throw) switch with the only main distinction being their typical use case. Those being SPCO is typically used for more automated applications such as relays, where SPDT switches are used for manual operation like user operated switches.

Buzzer/Sounder



The buzzer creates a tone based on the voltage and signal of the input; a physical buzzer is made up from a small electromagnet placed behind a flexible diaphragm with a ferromagnetic disk connected to it. As the electromagnet has a current passed through its coil, this attracts the ferromagnetic disk, deforming the diaphragm producing sound waves

Transistor



The transistor also operates as both a switch and as an amplifier, in this case the former. When there is current supplied to the base pin, current can flow from collector to emitter. The BC107 is a common NPN BJT (Bipolar Junction Transistor).

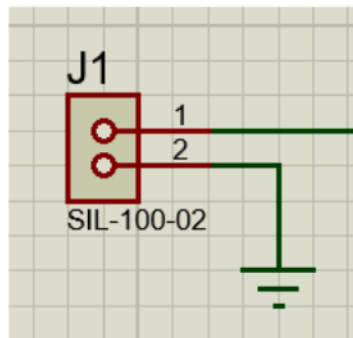
The maximum ratings for the BC107/108/109 can found below.

Absolute Maximum Ratings

DESCRIPTION	SYMBOL	BC107	BC108	BC109	UNIT
Collector-Emitter Voltage	V_{CE0}	45	25	25	V
Collector-Base Voltage	V_{CBO}	50	30	30	V
Emitter-Base Voltage	V_{EBO}	6.0	5.0	5.0	V
Collector Current Continuous	I_C		0.2		A
Power Dissipation at $T_a = 25^\circ\text{C}$	P_D		0.6		W
Derate Above 25°C			2.28		mW/ $^\circ\text{C}$
Power Dissipation at $T_c = 25^\circ\text{C}$	P_D		1.0		W
Derate Above 25°C			6.67		mW/ $^\circ\text{C}$
Operating And Storage Junction	T_j, T_{stg}		-65 to +200		$^\circ\text{C}$
Temperature Range					
Thermal Resistance					
Junction to Case	$R_{th(j-c)}$		175		$^\circ\text{C/W}$

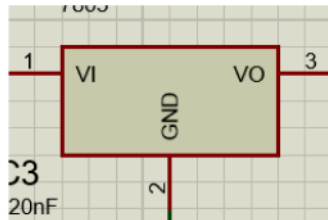
[3]

Connector



Serves as a connection point for the circuit to receive power input, this specific one used is a SIL-100-02, SIL referring to Single in Line arrangement of the terminal block. Terminal blocks are used to securely connect wires often to a PCB. Typical constructions had a conductive screw above a slot, that the wire is fed into and tightened to hold it in place, pins on the underside then connect the terminal to the circuit.

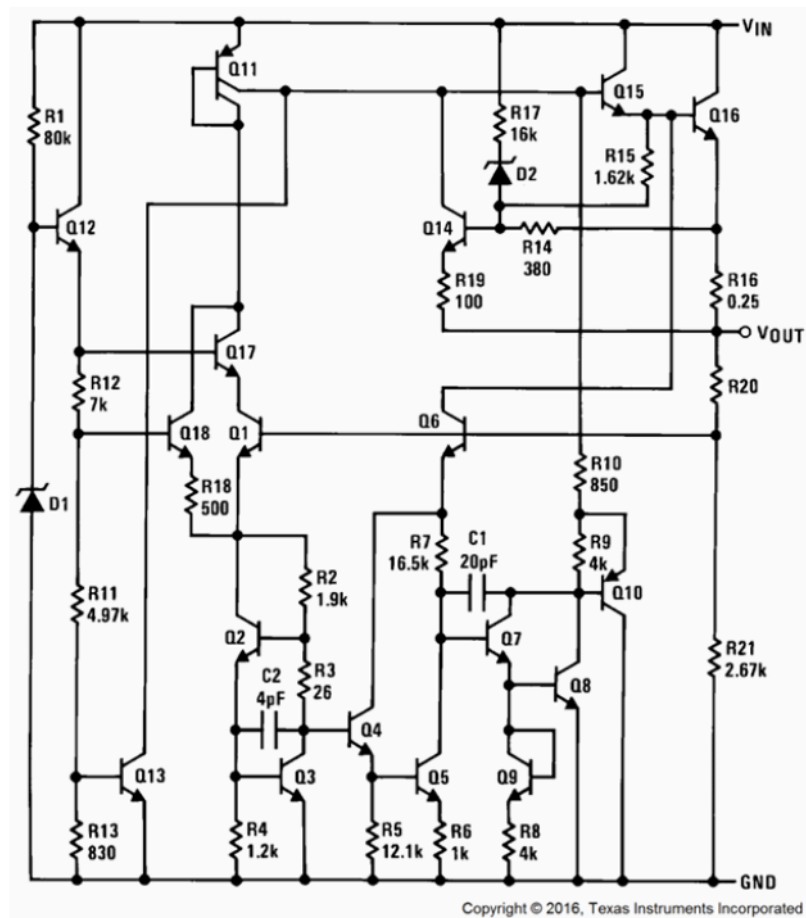
Voltage regulator



This Specific Voltage regulator (LM7805) Supplies the circuit with a stable and regulated +5V, denoted by the '05' from the LM78xx series of linear voltage regulators. The regulator takes in a higher voltage than its output, typically at least +2V above, to output a smoother 5V.

This specific regulator also uses a combination of internal current limiting, thermal shutdown, and safe-area compensation to prevent any potential damage to the component, which essentially mean the IC's circuit has a 'sensor' to limit power dissipation

Below shows the block diagram for the LM7800 series of voltage regulators

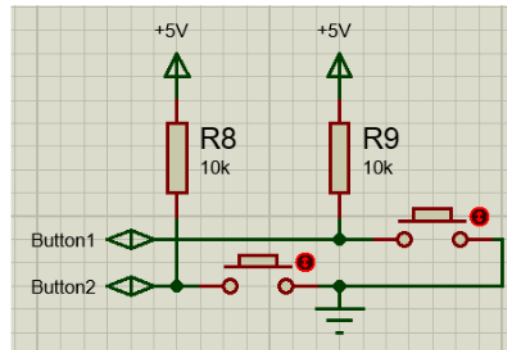


[4]

For more information regarding operation, see the power input module

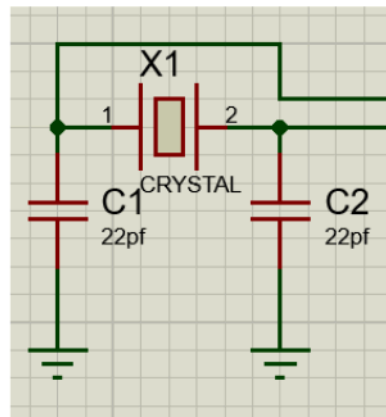
Circuit Modules

Button Inputs



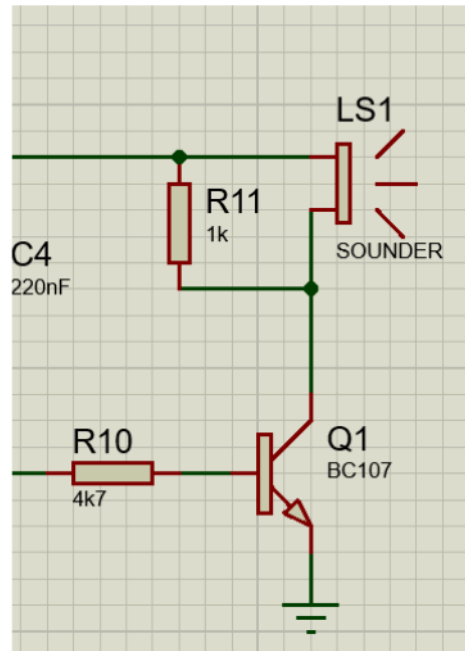
The above module operates as the main input for the overall circuit, when either of the buttons are pressed, they bring their respective PIC pins low, acting as an input for the PIC to detect. The pull up resistors are used to give definitive logic high (+5V) when the buttons aren't being pressed.

Clock Inputs



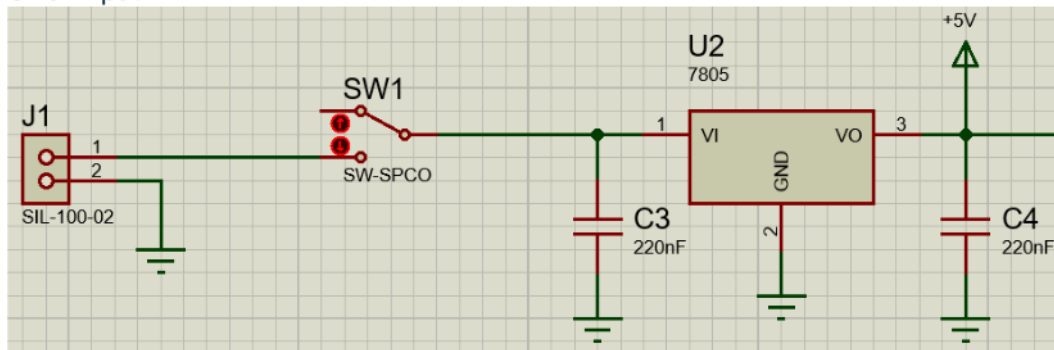
The External clock module dictates and synchronises the PIC's operations and the time it takes to execute instructions based on the clock speed. The capacitors are used in conjunction with the crystal to allow it to operate at its desired frequency.

Sounder



The sounder module is driven by the power input module but is operated based on the state of the transistor switching, which itself is driven via an output from the PIC. When current is supplied to the base of the transistor, current can flow from the collector to the emitter, allowing the sounder to produce output in the form of noise. Both resistors in this module are acting as current limiters for both the sounder and transistor.

Power Input

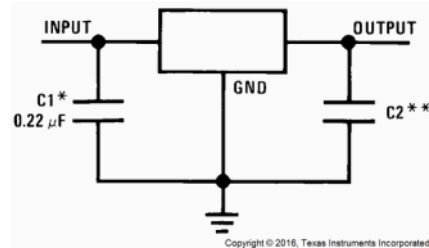


The circuit is powered by from the input terminal block leading to a relay which allows switching of the 'power input' by an external lower power circuit. The relay then feeds into a voltage regulator which provides a stable 5v for the rest of the circuit. Decoupling capacitors are used either side of the voltage regulator to reduce any noise or fluctuations from the supply.

Below shows the basic implementation of the regulator as per the datasheet

8.2.1 Fixed Output Voltage Regulator

The LM340x and LM7805 Family devices are primarily designed to provide fixed output voltage regulation. The simplest implementation of LM340x and LM7805 Family is shown in [Figure 18](#).

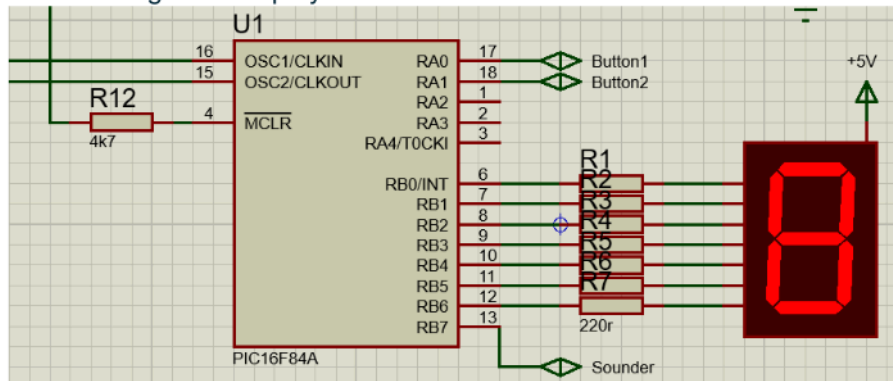


*Required if the regulator is located far from the power supply filter.

**Although no output capacitor is needed for stability, it does help transient response. (If needed, use 0.1- μ F, ceramic disc).

[4]

PIC16F84A & 7 Segment Display



The PIC has its MCLR pin (Master Clear) held high to avoid accidental resets during circuit operation.

The PIC is also connected to the common anode 7 Segment-Display via the RB0 – 6 pins with current limiting resistors to protect the LEDs of the display from potential over current.

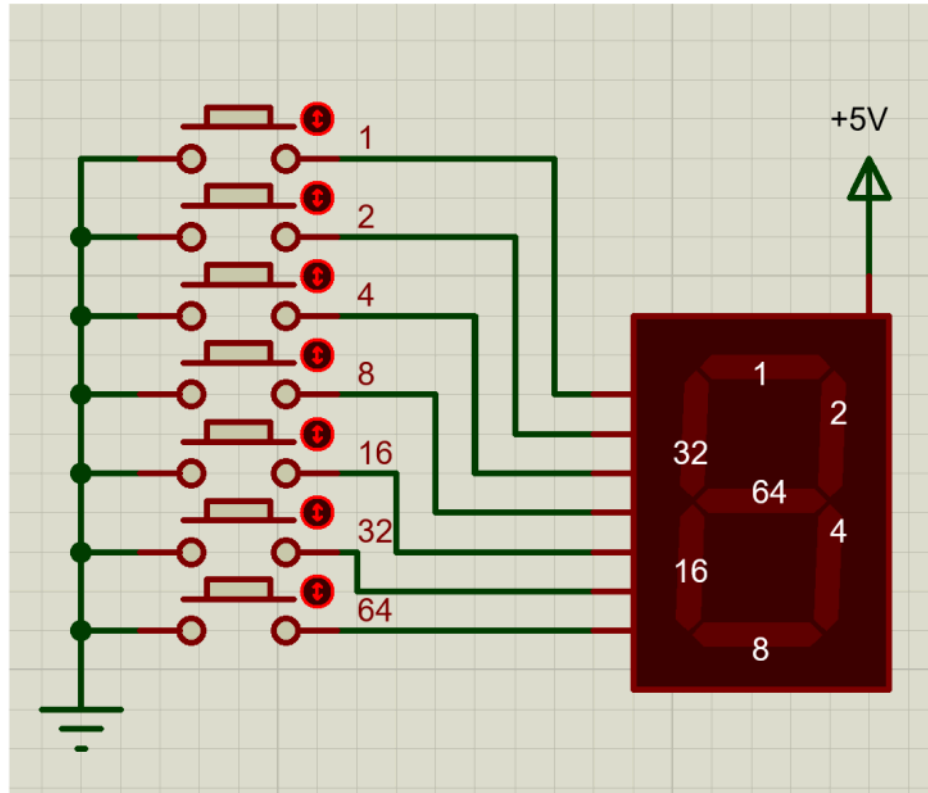
In this circuit, the RA0 and RA1 pins are configured in software to function as inputs, to detect change in the buttons.

OSC1/CLKIN and OSC2/CLKOUT are connected to the Clock module that are acting as the PIC's external oscillator

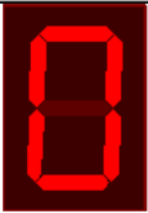
RB7 is configured to function as an output that leads to the base of the transistor detailed in the sounder module

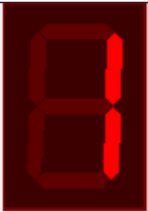
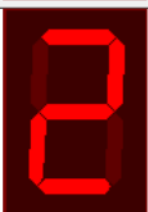
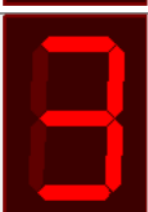
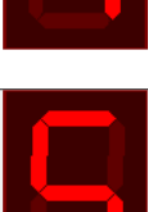
Section 2 – 7 Segment Display Operation

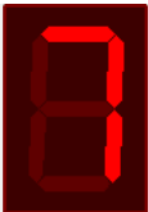
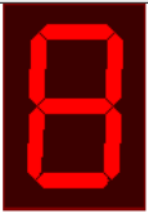
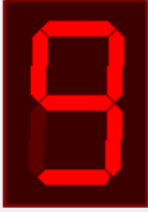
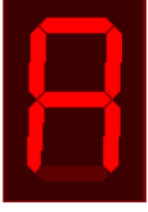
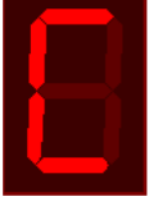
To have the seven-segment display light correctly, each the pins require a connection to ground due the seven-segment display being a common anode.

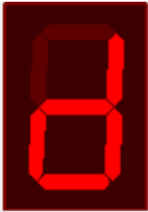
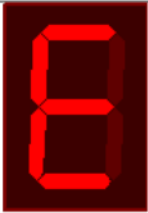


Each pin has a 'value' that the total dictates the LED's lit, and due to the 7 Segment-Display being common anode, the total value equates to those component value pins being unlit. For example, to display a zero, the middle LED needs to be off, thus the value would equal 64. Running this through the PIC results in all but the centre LED being on.

7 Segment	Example	Decimal Value	Code
0		64	0x40

1		121	0x79
2		36	0x24
3		48	0x30
4		25	0x19
5		18	0x12
6		2	0x02

7		120	0x78
8		0	0x00
9		16	0x10
A		8	0x08
B		3	0x03
C		70	0x46

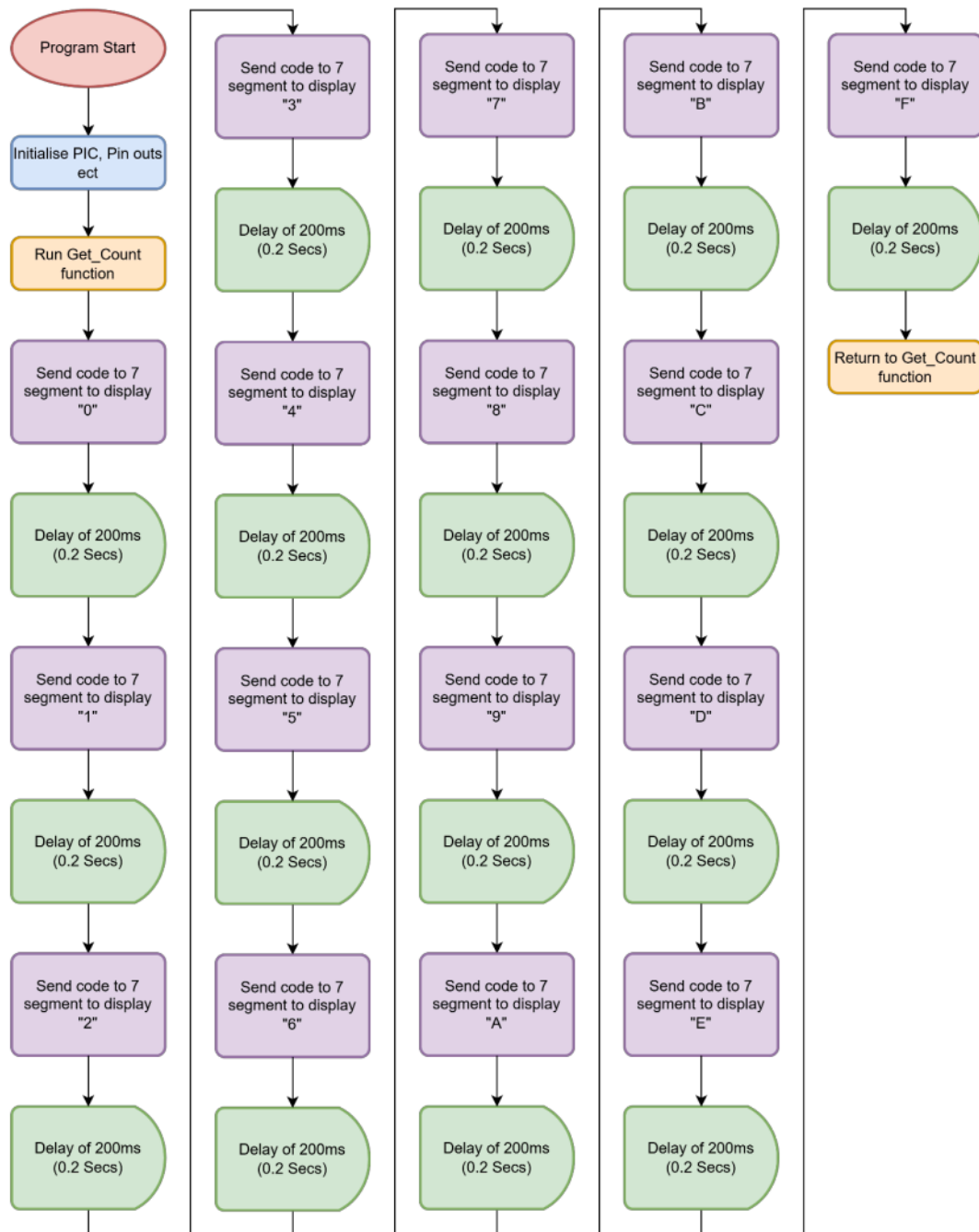
D		33	0x21
E		6	0x06
F		14	0x0e

Section 3 – Coding Task

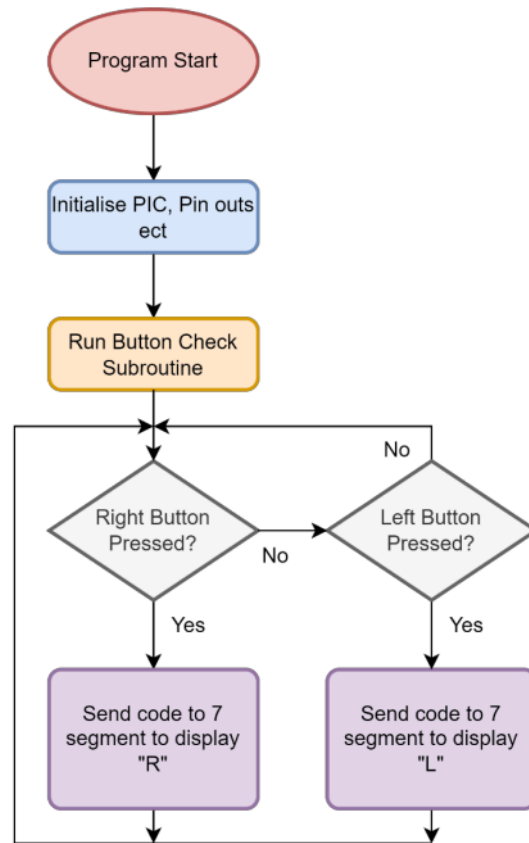
The table below outlines the coding tasks and flowcharts created, for the code in its entirety, see Appendix

Task	Flowchart	Code
1	Completed	Completed
2	Completed	Completed
3	Completed	Completed
4	Completed	Non-Functional
5	Completed	Completed
6	Not Attempted	Not Attempted

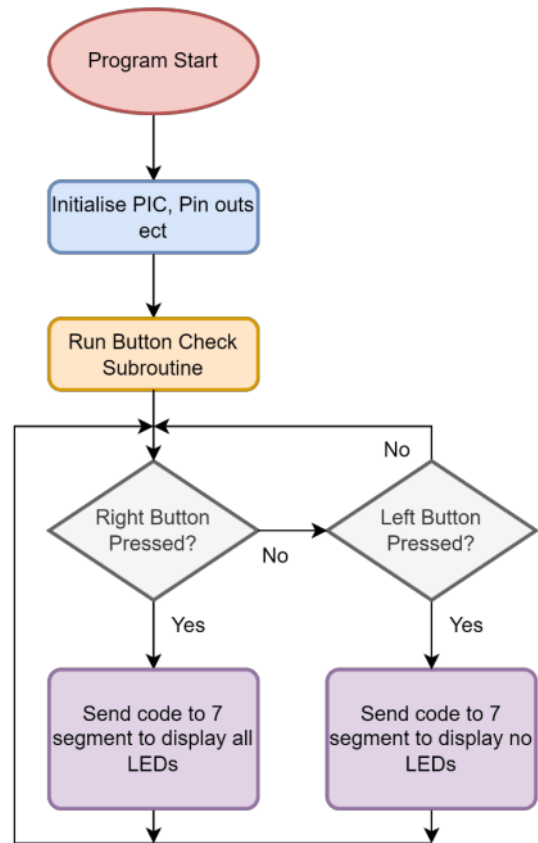
Coding Task 1 Flow Chart



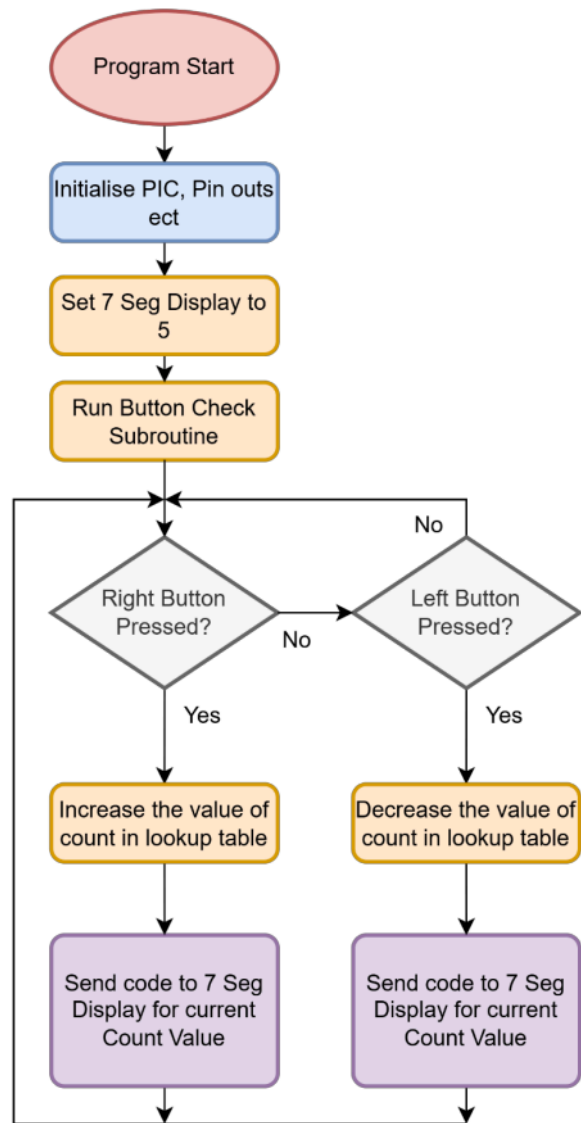
Coding Task 2 Flow Chart



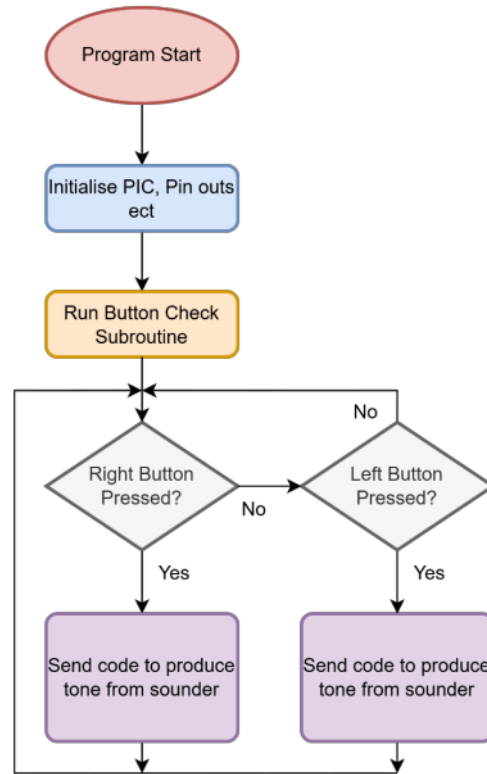
Coding Task 3 Flow Chart



Coding Task 4 Flow Chart



Coding Task 5 Flow Chart



Bibliography

- [1] Microchip Technology, "PIC16F84A Data Sheet," Microchip Technology Inc, 2001. [Online]. Available: <https://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf>. [Accessed 13 January 2025].
- [2] Cypress Semiconductor Corporation, "7-Segment LED Controller Datasheet," Cypress Perform, 2 July 2014. [Online]. Available: https://www.infineon.com/dgdl/Infineon-LED7SEG_User_Module-Software%20Module%20Datasheets-v01_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0fa9f7d51aaf. [Accessed 13 January 2025].
- [3] Farnell, "BC107/ BC108/ BC109 Low Power Bipolar Transistors," Multicomp, 30 May 2005. [Online]. Available: <https://www.farnell.com/datasheets/314715.pdf>. [Accessed 13 January 2025].
- [4] Texas Instruments, "LM340, LM340A and LM7805 Family Wide VIN 1.5-A Fixed Voltage Regulators," Texas Instruments, 2016. [Online]. Available: <https://www.ti.com/lit/ds/symlink/lm7800.pdf>. [Accessed 13 January 2025].

Appendix

Task 1 Code

```
list    p=16F84    ; select processor type
radix   hex        ; define radix as hexadecimal
;-----
portB   equ 0x06    ;port B address
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

count   equ 0x0c    ; counter 0 to 255
nCount  equ 0x0d    ; pause routine inner loop counter
mCount  equ 0x0e    ; pause routine outer loop counter
;-----
org     0x000
start
    bcf status,RP0  ; clear bank selection bits in the status reg
    bcf status,RP1

    bsf status,RP0  ; select bank 1
    movlw 0x00      ; load w with 0x00
    movwf trisb     ; load it into trisb to make portb all outputs
    bcf status,RP0  ; reselect bank 0

    clrf portB      ; clear all lines low
    clrf count      ; initialize counter to 0

get_count
    movf count, w   ; move count to W
    movwf portB     ; move W to port B to update the display
    movlw 0x40      ; load w with 0x40 (0)
    movwf count     ; save W in count
    call pause      ; delay 200ms by subroutine
    call pause      ; five pause executions equals ~ 1 second

    movf count, w   ; move count to W
    movwf portB     ; move W to port B to update the display
    movlw 0x79      ; load w with 0x79 (1)
    movwf count     ; save W in count
    call pause      ; delay 200ms by subroutine
    call pause      ; five pause executions equals ~ 1 second

    movf count, w   ; move count to W
    movwf portB     ; move W to port B to update the display
    movlw 0x24      ; load w with 0x24 (2)
    movwf count     ; save W in count
    call pause      ; delay 200ms by subroutine
```

```

call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x30    ; load w with 0x30 (3)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x19    ; load w with 0x19 (4)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x12    ; load w with 0x12 (5)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x02    ; load w with 0x02 (6)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x78    ; load w with 0x78 (7)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x00    ; load w with 0x00 (8)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x10    ; load w with 0x10 (9)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

movf    count, w    ; move count to W
movwf   portB    ; move W to port B to update the display
movlw   0x08    ; load w with 0x08 (A)
movwf   count    ; save W in count
call    pause    ; delay 200ms by subroutine
call    pause    ; five pause executions equals ~ 1 second

```

```

    movf    count, w      ; move count to W
    movwf   portB         ; move W to port B to update the display
    movlw   0x03          ; load w with 0x03 (B)
    movwf   count         ; save W in count
    call    pause         ; delay 200ms by subroutine
    call    pause         ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W
    movwf   portB         ; move W to port B to update the display
    movlw   0x46          ; load w with 0x46 (C)
    movwf   count         ; save W in count
    call    pause         ; delay 200ms by subroutine
    call    pause         ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W
    movwf   portB         ; move W to port B to update the display
    movlw   0x21          ; load w with 0x21 (D)
    movwf   count         ; save W in count
    call    pause         ; delay 200ms by subroutine
    call    pause         ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W
    movwf   portB         ; move W to port B to update the display
    movlw   0x06          ; load w with 0x06 (E)
    movwf   count         ; save W in count
    call    pause         ; delay 200ms by subroutine
    call    pause         ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W
    movwf   portB         ; move W to port B to update the display
    movlw   0x0e          ; load w with 0x0e (F)
    movwf   count         ; save W in count
    call    pause         ; delay 200ms by subroutine
    call    pause         ; five pause executions equals ~ 1 second

    goto    get_count     ; repeat 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      ; 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 subroutine

end

```

Task 2 Code

```
list    p=16F84    ; select processor type
radix   hex        ; define radix as hexadecimal
;-----
portA   equ 0x05
portB   equ 0x06    ;port B address
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

count   equ 0x0c      ; counter 0 to 255
nCount  equ 0x0d      ; pause routine inner loop counter
mCount  equ 0x0e      ; pause routine outer loop counter
;-----
org     0x000
start
    bcf status,RP0    ; clear bank selection bits in the status reg
    bcf status,RP1

    bsf status,RP0    ; select bank 1
    movlw 0x00        ; load w with 0x00
    movwf trisb       ; load it into trisb to make portb all outputs
    bcf status,RP0    ; reselect bank 0

    clrf portB        ; clear all lines low
    clrf count        ; initialize counter to 0

buttons
    btfsc portA, 0
    goto segL         ; goto the segL Subroutine
    movf count, w
    movlw 0x4e        ; Displays R on the 7 Segment display
    movwf portB
    movwf count
    call pause        ; Calls the Pause Subroutine

segL
    btfsc portA, 1
    goto buttons      ; loop back to buttons subroutine
    movf count, w
    movlw 0x47        ; Displays L on the 7 Segment display
    movwf portB
    movwf count
    call pause        ; Calls the Pause Subroutine
    goto buttons      ; loop back to buttons subroutine

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 subroutine

end

```

Task 3 Code

```

list    p=16F84      ; select processor type
radix   hex          ; define radix as hexadecimal
;-----
portA   equ 0x05
portB   equ 0x06      ;port B address
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

count   equ 0x0c      ; counter 0 to 255
nCount  equ 0x0d      ; pause routine inner loop counter
mCount  equ 0x0e      ; pause routine outer loop counter
;-----
org     0x000
start
    bcf status,RP0    ; clear bank selection bits in the status reg
    bcf status,RP1

    bsf status,RP0    ; select bank 1
    movlw 0x00        ; load w with 0x00
    movwf trisb       ; load it into trisb to make portb all outputs
    bcf status,RP0    ; reselect bank 0

    clrf portB        ; clear all lines low
    clrf count        ; initialize counter to 0

all
    btfsc portA, 0
    goto none         ; goto the none Subroutine
    movf count, w
    movlw 0x00        ; Displays all LEDs on the 7 Segment display
    movwf portB
    movwf count
    call pause        ; Calls the Pause Subroutine

none
    btfsc portA, 1
    goto all          ; loop back to all subroutine
    movf count, w
    movlw 0xff        ; Displays no LEDs on the 7 Segment display
    movwf portB
    movwf count

```

```

    call    pause      ; Calls the Pause Subroutine
    goto    all        ; loop back to all subroutine

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 subroutine

end

```

Task 4 Code

```

    list    p=16F84    ; select processor type
    radix   hex        ; define radix as hexadecimal
;-----
portA equ 0x05
portB equ 0x06        ;port B address
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

count equ 0x0c        ; counter 0 to 255
nCount equ 0x0d       ; pause routine inner loop counter
mCount equ 0x0e       ; pause routine outer loop counter
;-----
    org     0x000
start
    bcf     status,RP0 ; clear bank selection bits in the status reg
    bcf     status,RP1

    bsf     status,RP0 ; select bank 1
    movlw   0x12       ; load w with 0x00
    movwf   trisb      ; load it into trisb to make portb all outputs
    bcf     status,RP0 ; reselect bank 0

    clrf    portB      ; clear all lines low
    movlw   count      ; initialize counter to 0

```

Table

```

    retlw   0x40       ; 0 on 7 Segement Display
    retlw   0x79       ; 1 on 7 Segement Display
    retlw   0x24       ; 2 on 7 Segement Display

```

```

retlw 0x30 ; 3 on 7 Segement Display
retlw 0x19 ; 4 on 7 Segement Display
retlw 0x12 ; 5 on 7 Segement Display
retlw 0x02 ; 6 on 7 Segement Display
retlw 0x78 ; 7 on 7 Segement Display
retlw 0x00 ; 8 on 7 Segement Display
retlw 0x10 ; 9 on 7 Segement Display

```

IncrementR

```

btfsc portA, 0
goto DecrementL ; goto the segL Subroutine
movf count, w
; Displays R on the 7 Segment display
movwf portB
movwf count
call pause ; Calls the Pause Subroutine
movf count, w
; Displays R on the 7 Segment display
movwf portB
movwf count
call pause

```

DecrementL

```

btfsc portA, 1
goto IncrementR ; loop back to buttons subroutine
movf count, w
; Displays L on the 7 Segment display
movwf portB
movwf count
call pause
movf count, w
; Displays L on the 7 Segment display
movwf portB
movwf count
call pause ; Calls the Pause Subroutine
goto IncrementR ; loop back to buttons subroutine

```

pause

```

movlw 0x07 ; set w = 255 decimal
movwf mCount ; mCount = w: outer loop counter
loadN
movlw 0xc8 ; 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 subroutine

```

end

Task 5 Code

```
list    p=16F84    ; select processor type
radix   hex        ; define radix as hexadecimal
;-----
portA   equ 0x05
portB   equ 0x06    ;port B address
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

count   equ 0x0c      ; counter 0 to 255
nCount  equ 0x0d      ; pause routine inner loop counter
mCount  equ 0x0e      ; pause routine outer loop counter
;-----
org     0x000
start
    bcf status,RP0    ; clear bank selection bits in the status reg
    bcf status,RP1

    bsf status,RP0    ; select bank 1
    movlw 0x00        ; load w with 0x00
    movwf trisb       ; load it into trisb to make portb all outputs
    bcf status,RP0    ; reselect bank 0

    clrf portB        ; clear all lines low
    clrf count        ; initialize counter to 0

ToneR
    btfsc portA, 0
    goto  ToneL       ; goto the ToneL Subroutine
    movf count, w
    movlw 0xff         ; Switches Transistor High
    movwf portB
    movwf count
    call pause         ; Calls the Pause Subroutine
    movf count, w
    movlw 0x7f         ; Switches Transistor low
    movwf portB
    movwf count
    call pause

ToneL
    btfsc portA, 1
    goto  ToneR       ; loop back to buttons subroutine
    movf count, w
    movlw 0xff         ; Switches Transistor High
    movwf portB
    movwf count
    call pause         ; Calls the Pause Subroutine
    movf count, w
    movlw 0x7f         ; Switches Transistor low
```

```

    movwf    portB
    movwf    count
    call     pause      ; Calls the Pause Subroutine
    goto     ToneR      ; loop back to buttons subroutine

pause
    movlw    0x07        ; set w = 255 decimal
    movwf    mCount      ; mCount = w: outer loop counter
loadN
    movlw    0xc8        ; 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 subroutine

end

```

Micro, Peripherals & Interfaces Report (LS)

GRADEMARK REPORT

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

PAGE 31

PAGE 32

GRADING FORM: GJ_2019

LOUIS S

301

ENGAGEMENT WITH LITERATURE



While there are four reasonable sources cited, you really need more for an assignment such as this,

and they should cover as many aspects of the report as possible.

40

KNOWLEDGE AND UNDERSTANDING



The K&U demonstrated is reasonable. But more EWL would have helped obtain an even higher mark here

68

COGNITIVE AND INTELLECTUAL



Fairly straightforward coding, so as this section if the marks is specifically for programming flair and innovation, this mark isn't particularly high.

50

PRACTICAL APPLICATION



Circuit is well implemented and most (although not all) of the tasks are attempted reasonably well. Nice flowcharts!

68

TRANSFERABLE SKILLS



Generally very well laid out and presented.

75