

Microperipheral and Interfaces

Alexander Price

by Alex Price

Submission date: 08-Jan-2025 03:40PM (UTC+0000)

Submission ID: 247658975

File name: 51693_Alex_Price_Microperipheral_and_Interfaces_Alexander_Price_621295_856098734.docx
(718.45K)

Word count: 2248

Character count: 10817



MICROPERIPHERALS AND INTERFACES ASSEMBLY REPORT



Alex Price (2306891)
UWTSD L4-Microperipherals and Interfaces

Table of Contents

Introduction	2
Circuit and Function	2
List of Components	2
PIC16F84A	3
7805 voltage regulator	3
Crystal Oscillator	4
Buzzer	4
7 Segment Display Explanation	4
Tasks	6
Task 1	6
Task 2	7
Task 3	8
Task 4	9
Task 5	10
Conclusion	10
References	10
Appendix A	12
Task 1	12
Appendix B	14
Task 2	14
Appendix C	15
Task 3	15
Appendix D	16
Task 4	16
Appendix E	20
Task 5	20
Figure 1 Circuit	2
Figure 2 7805 Voltage regulator internal circuit	3
Figure 3 7 Segment Display Pinout	5
Figure 4 Task 1 Flowchart	6
Figure 5 Task 2 Flowchart	7
Figure 6 Task 3 Flowchart	8
Figure 7 Task 4 Flowchart	9
Figure 8 Task 5 Flowchart	10

Introduction

This task requires me to program a PIC16F84A for several functions with external I/O in a simulated environment. These included button inputs, displaying to a 7-segment display, and sounding a buzzer. This allowed me to explore a wide range of methods to get functionality out of the PIC and the circuit. This also allowed me to learn about the assembly language and allow me to learn about the inner workings of a RISC chip.

Circuit and Function

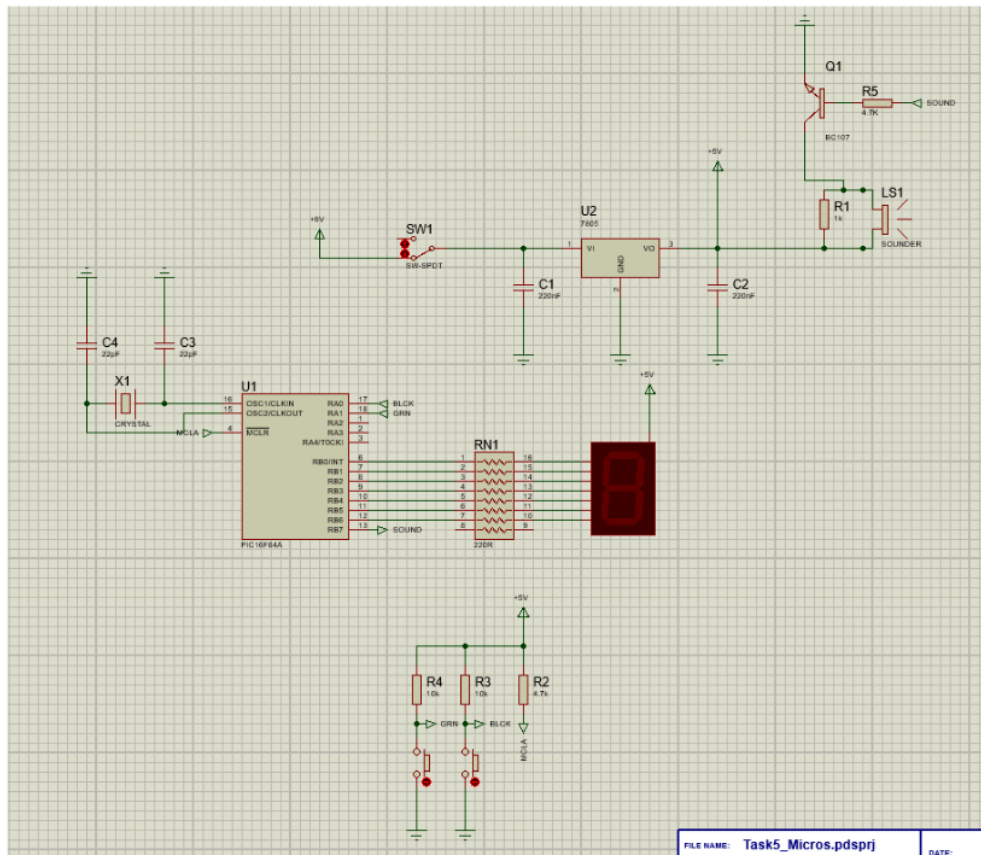


Figure 1 Circuit

List of Components

- PIC16F84A
- 7805 voltage regulator
- Crystal Oscillator
- Capacitors
- Buzzer

PIC16F84A

The PIC16F84A is a microcontroller in the PIC family made by Microchip. It is based on an 8-bit architecture with CMOS flash and EEPROM. This chip is ideally used for prototyping and small-scale device production. The chip has a 1024-word program memory so it can have a lot of operations configured to be executed within the chip, with a wide amount of EEPROM and Data RAM. The chip also has 13 I/O pins to ensure plenty of I/O expansion to the chip. [1]

The PIC chip is based on the RISC architecture unlike most traditional computer chips that run on CISC (Complex instruction Set Computing) This means unlike the x86 instruction set that 1000s of instructions [2] [3] [4] the PIC only has 35 instructions to learn [1]. This makes programming it in assembly far easier and in most cases more resource and power efficient. Overall this PIC is suited to the task at hand as it has plenty of computation power and registers to complete the tasks as shown below.

7805 voltage regulator

The 7805 voltage regulator is a component that takes an input voltage and regulates it to a fixed value depending on the component specifications. The pinout of the 7805 is very simple as there are 3 pins. The output that outputs a regular 5 volts, the input where the different voltage is inputted to the system, and then finally the ground pin to ground the component.

The chip is made up mostly of resistors and transistors internally. The principle is based on a single transistor that controls the current between the input and output port therefore controlling the current. However, before this, there are a few steps to get accurate values inside the circuit.

schematic

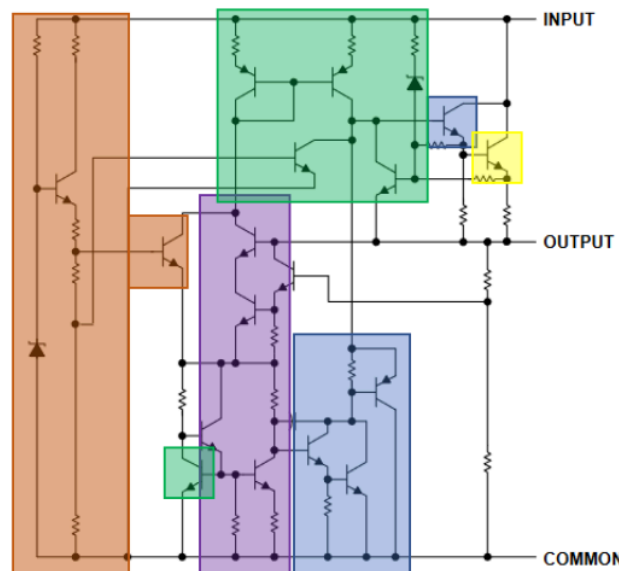


Figure 2 7805 Voltage regulator internal circuit

Figure 2, above, shows the internal circuit of the 7805 as we can see there is a zone in dark blue is used to amplify the error signal this amplifier then controls the output which is seen at the smaller dark blue transistor. The purpose of this signal is an error signal to see if the voltage output is too

high when compared to the output. This makes a feedback loop that always keeps the voltage regulator outputting the correct amount of voltage no matter the temperature or conditions around the IC. The rest of these transistors have 3 main jobs. The first is the orange section which is the start-up circuit, this stops the circuit from getting caught in an infinite booting cycle. The purple part of the circuit is a band gap circuit this circuit is designed to keep the voltage stable and allow for the circuit to carry out its intended function. [5]

Crystal Oscillator

The crystal oscillator in the circuit is used to keep timing across the chip and the entire circuit. The component does this by having a small crystal that expands or contracts at a fixed frequency, this is called resonance. This crystal oscillator's output frequency is either the crystal's fundamental resonance or a harmonic of this. [6] The crystal oscillator has resistors across it to load the crystal. The process of choosing this capacitor is based on the crystal resonance if the capacitor is too strong the crystal will oscillate too slowly and if it is too low the crystal will oscillate too quickly and break the IC. The equation shown below shows the relation between the capacitance of the capacitors and the spec of the crystal:

$$CL = (C1 * C2) / (C1 + C2) + Cstray$$

Therefore, when rearranged for calculating the capacitance of the capacitors around the crystal it becomes this:

$$C1 = C2 = 2 * (CL - Cstray)$$

The value of Cstray is usually around 5pF as it comes from the pins and parasitic elements on the circuit. [7] However, the recommended practice is to look at the data sheet and use the manufacturer's recommended circuit and capacitance values as seen in the datasheet.

Buzzer

The buzzer in the circuit operates on the piezo-electric effect. This entire principle is based on applying an alternating voltage across the device, which then flexes the piezoelectric material, making it resonate at a set frequency audible to humans. The circuit has a transistor in it as it is used to ground the system and allow it to return to a settled state when current isn't passing through the system and then restart immediately. This allows the system to start and restart incredibly reliably and quickly. [8]

7 Segment Display Explanation

To begin with, the output of the circuit first is the 7-segment display. This is one of the key outputs of the circuit and allows the system to display whatever is needed. The display has 7 addressable inputs, allowing the chip to send a combination of high and low outputs to display what is required on the chip. In the circuit, there is a +5V input to the chip so to turn off segments of the display the chip merely outputs +5V from the PIC to create no potential difference and turn off that segment of the display. This output from the PIC goes through a bank of 220Ω so the 7-segment display isn't destroyed due to too much voltage. This would cause the display to burn out rapidly and destroy the system.

For the 7-segment display to output hexadecimal numbers first, the controller chip would have to convert segments of the display to project the correct image onto it. This would be done by looking at the specification sheet and finding the pin layout.

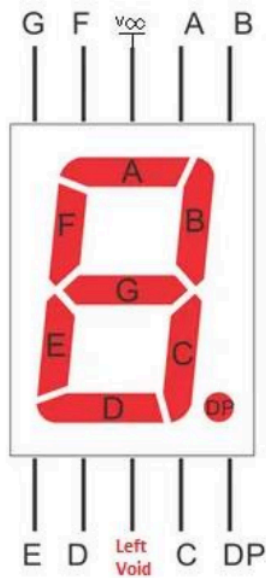


Figure 3 7 Segment Display Pinout

Above is the pinout for a 7-segment display and as shown making the potential difference +5V it will illuminate. So knowing what pins apply to what segment the system can construct a binary sequence to push to the chip to display the correct value. An example of this would be the hex value of 0x05. This is represented by the value '00010010'. This represents what segments are high and what segments are low. The 0 segments will be illuminated as there is a +5V potential difference, then the 1 segments will be off as there is a +5V from the PIC and from the supply to the display so creates a 0V potential difference. This needs to be outputted to the correct pins to display on the 7-segment display. Then this binary value is sent to port B of the chip from the working register and this has all the pinouts to the 7-segment display within it. Then this is pushed through the resistor bank and passed through to the 7-segment display. As seen in some of the examples like the task 4 project with incrementing from 0-9 it is this process but just with 6 more characters to display.

Tasks

Task 1

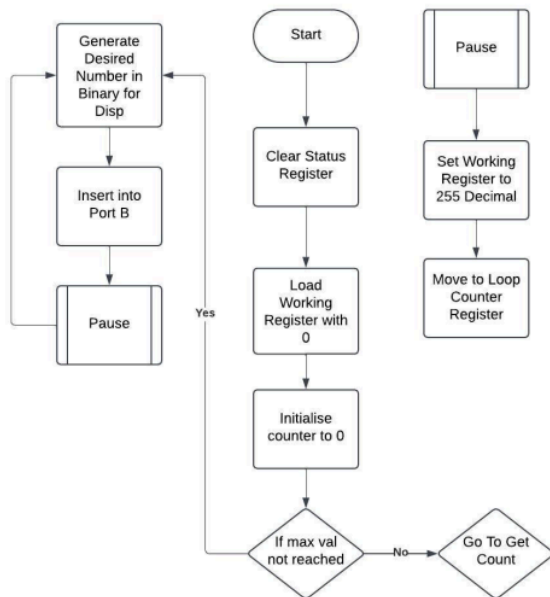


Figure 4 Task 1 Flowchart

Task 1 involves incrementing through the hexadecimal number system and then looping back to the beginning of the system. This is done by first doing the binary conversion from the hexadecimal character to be shown on the 7-segment display. This involves looking at the datasheet for the 7-segment display and then turning on and off the correct pins linking to the microcontroller. As shown in Figure 1 Circuit there is a +5V source connected to the 7-segment display to illuminate it. Then the code sets the correct pins high to cancel the potential difference of the 7-segment display and display the correct number or letter. It does this for each number and letter then calls the pause subroutine to add a delay and then moves to the next character and loops infinitely.

Task 2

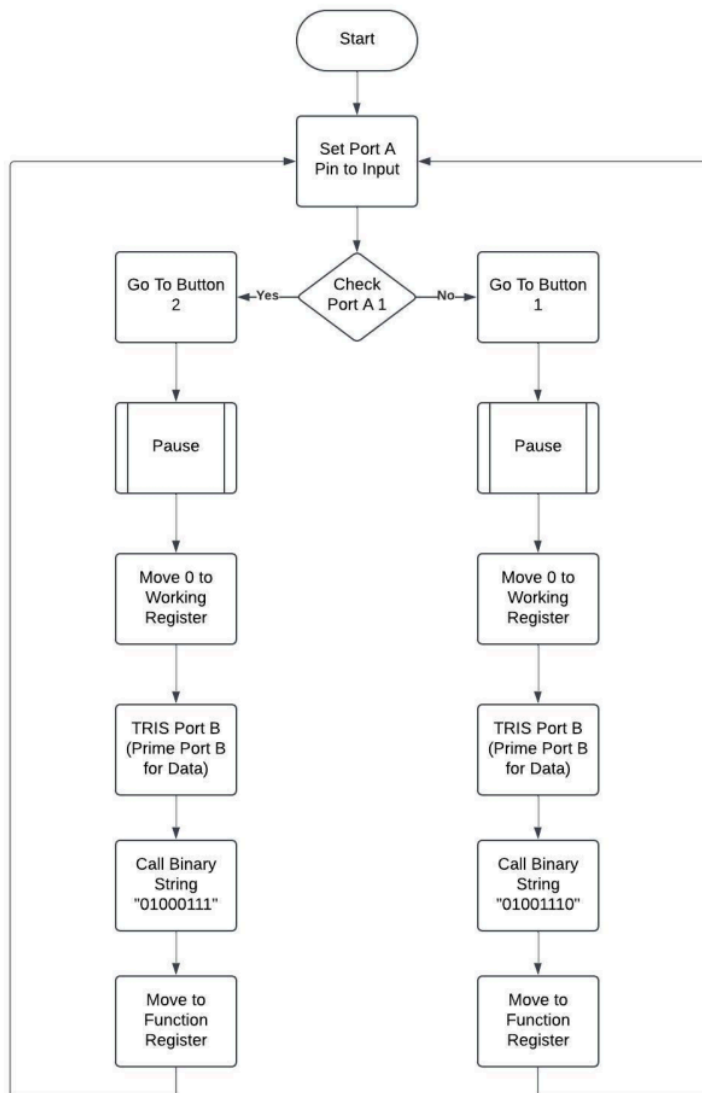


Figure 5 Task 2 Flowchart

Task 2 was a task of implementing a left and right system via the PIC. This has a simple system of a left and right button that changes what is displayed. Much like task 1 firstly the 7-segment conversion to binary is needed to project it onto the display once this is done then the main function can begin.

Firstly, a check is done on the F register to see if there is a 1 or 0 input to see what should first be displayed either L or R. Then depending on this it goes to either the L or R subroutine. They are the same except for the binary string sent to the 7-segment display. It begins with clearing the W

register and then inputting the literal binary string of the desired displayed item. Then move this string to the function register and out to the display and then loop the system back and repeat.

Task 3

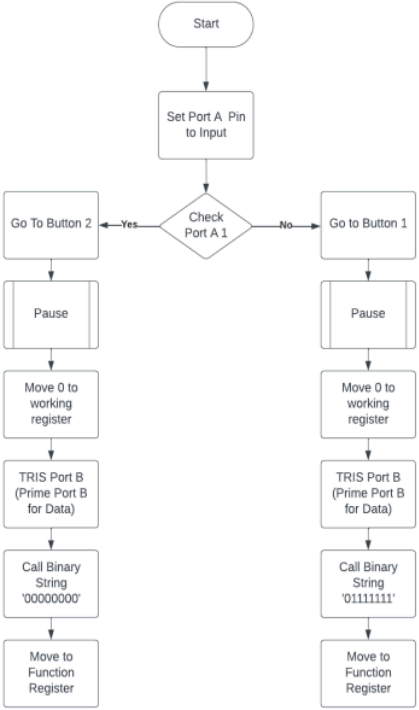


Figure 6 Task 3 Flowchart

Task 3 was the same as task 2 except for the bit string sent to the 7-segment display this created either a fully turned on or off 7-segment display.

Task 4

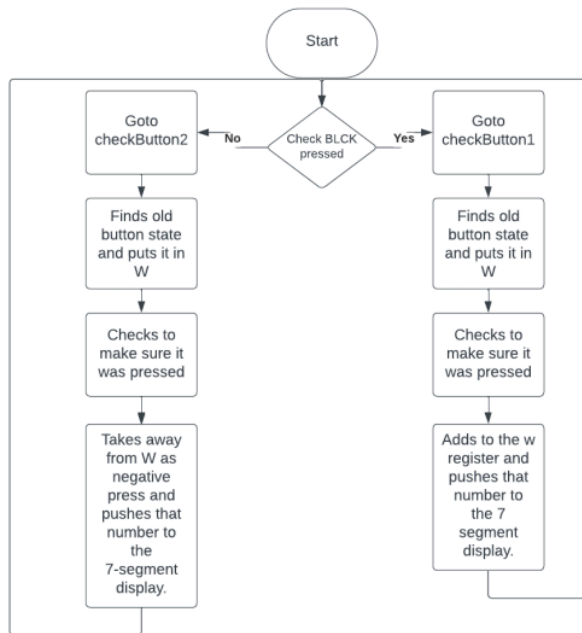


Figure 7 Task 4 Flowchart

The flowchart above is highly simplified but will be explained in depth within this paragraph, along with each line being commented on in the code in the appendix D.

The main loop starts with checking if the increment or decrement button is pressed once one of these is selected the process is the exact same except for adding one value or taking away another.

Once a button has been pressed the main loop will drop into one of the subroutines to check the button has been pressed this starts with the old button state being called and loaded into the W register. Then the increment or decrement routine is called depending on what the button pressed was.

The increment subroutines first calls the count of the display currently. It checks if it is either 9 for addition or 0 for subtraction. If so it will skip the mathematical action and return to the main loop. However, if the addition or subtraction needs to happen it will increment or decrement the count in the f register and then call the display subroutine.

The display subroutine then moves the count to the W register. Then this starts the chain of checking what number the count is and pushing the correct binary string out of portB and returns to the main loop after pushing the correct number.

Once the number has been pushed the old button state is set to 1 to represent it was pressed. This the loop checking for a button press and going through the steps to keep the number correct.

Task 5

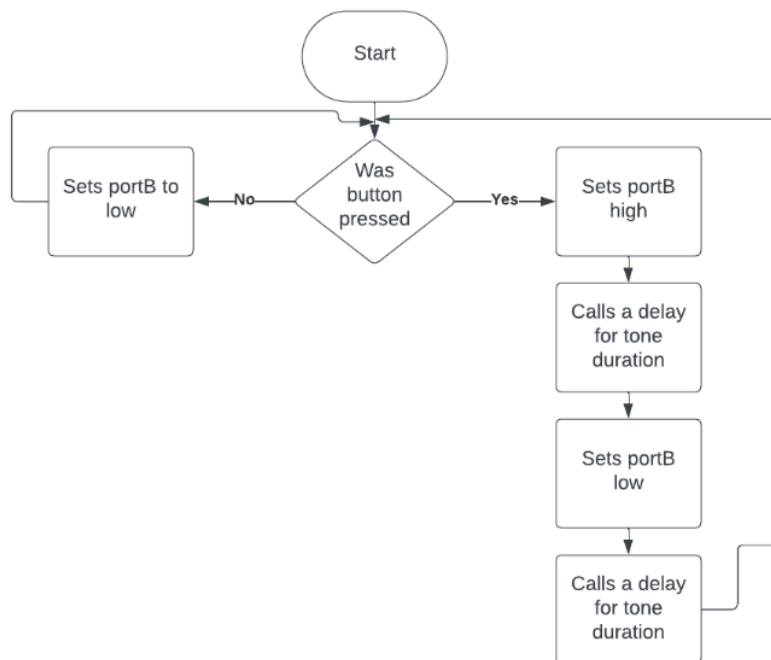


Figure 8 Task 5 Flowchart

Task 5 is a simple PWM buzzer script to push a button and the tone is produced. Simply, it checks for input from the button and if this is done then the buzzer is set high for a fixed time and is then set it high and low over the time the button is pressed. This gives the buzzer its tone and allows it to give off noise. Then once the button isn't pressed it sets the portB low and then repeats the loop.

Conclusion

Out of the 6 tasks 5 were complete and described here along with how the circuit works and how the 7-segment display outputs for the system.

References

- [1] Microchip, "PIC16F84A Data Sheet," 2001. [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf>. [Accessed 07 November 2024].
- [2] NASM.US, "nasm.us assembler docs," 14 February 2015. [Online]. Available: <https://www.nasm.us/xdoc/2.16.03/html/nasmdoc0.html>. [Accessed 08 November 2024].

- [3] University of Virginia Computer Science, "x86 Assembly Guide," 27 April 2006. [Online]. Available: <https://www.cs.virginia.edu/~evans/cs216/guides/x86.html>. [Accessed 08 November 2024].
- [4] N. A. A. R. Shahla Gul, "A comparison between RISC and CISC Microprocessor Architecture," *International Journal of Science Engineering and Microprocessor Architecture*, vol. 4, no. 5, pp. 254-259, 2016.
- [5] Texas Instruments, "A7800 Series Positive Voltage Regulators," May 2003. [Online]. Available: <https://www.sparkfun.com/datasheets/Components/LM7805.pdf>. [Accessed 08 November 2024].
- [6] M. A. Lombardi, *Encyclopedia of Physical Science and Technology*, Tarzana: Academic Press, 2003.
- [7] Microchip, "Calculating Crystal Load Capacitor," Microchip, 11th September 2020. [Online]. Available: <https://microchip.my.site.com/s/article/Calculating-crystal-load-capacitor>. [Accessed 3 December 2024].
- [8] RS Components, "How do Piezo Buzzers Work?," RS Components, 29 January 2024. [Online]. Available: <https://uk.rs-online.com/web/content/discovery/ideas-and-advice/how-do-piezo-buzzers-work>. [Accessed 3 December 2024].

Appendix A

Task 1

```
; Main.asm file generated by New Project wizard
;
; Created:   Alexander Price
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)

; DEFINITIONS
;-----
#include p16f84a.inc      ; Include register definition file

portB    equ 0x06         ; portb 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

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

; RESET and INTERRUPT VECTORS
;-----

; Reset Vector
RST      code 0x00
        goto Start

; MAIN
PGM      code
Start
        bcf     status,RP0      ; clears status registry
        bcf     status,RP1

        bsf     status,RP0      ; select bank 1
        movlw   0x00            ; load w with 0x00
        movwf   trisB          ; make trisb all outputs
        bcf     status,RP0      ; go back to bank 0

        clrf    portB          ; clear all wires to low
        clrf    count          ; start counter at 0

display
; The need for 0 at the start of each binary sequence keeps RB7 low at not display anything.

        movlw   b'01000000'    ;0
        movwf   portB          ; move W to portb and edit the display
        call    pause          ; delay subroutine

        movlw   b'0111001'     ;1
        movwf   portB          ; as above
        call    pause          ; as above

        movlw   b'00100100'    ;2
        movwf   portB          ; as above
        call    pause          ; as above

        movlw   b'00110000'    ;3
        movwf   portB          ; as above
        call    pause          ; as above

        movlw   b'00011001'    ;4
        movwf   portB          ; as above
        call    pause          ; as above

        movlw   b'00010010'    ;5
        movwf   portB          ; as above
        call    pause          ; as above

        movlw   b'00000011'    ;6
        movwf   portB          ; as above
        call    pause          ; as above
```

```

movlw b'01111000' ;7
movwf portB ; as above
call pause ; as above

movlw b'00000000' ;8
movwf portB ; as above
call pause ; as above

movlw b'00010000' ;9
movwf portB ; as above
call pause ; as above

movlw b'00001000' ;A
movwf portB ; as above
call pause ; as above

movlw b'00000011' ;B
movwf portB ; as above
call pause ; as above

movlw b'01000110' ;C
movwf portB ; as above
call pause ; as above

movlw b'00100001' ;D
movwf portB ; as above
call pause ; as above

movlw b'00000110' ;E
movwf portB ; as above
call pause ; as above

movlw b'00001110' ;F
movwf portB ; as above
call pause ; as above

goto display ; 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

```

Appendix B

Task 2

```
; Main.asm file generated by New Project wizard
;
; Created: Alexander Price
; Processor: PIC16F84A
; Compiler: MPASM (Proteus)

; DEFINITIONS
;-----
#include p16f84a.inc      ; Include register definition file

portb    equ 0x06         ; portb address
trisb    equ 0x06         ; trisb address in 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         ; porta address
trisa    equ 0x05         ; trisa address in bank 0

; 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
;-----

org 0
movlw 0x01      ; set the port A pins
movwf trisa     ; set up for input pins
bcf status,5    ; switch back to bank 0
goto Button_1   ; go to Button_1

MainLoop
Button_1 BTFSC porta,0 ; if porta is Low skip the next instruction if high go to next instruction
goto Button_2     ; goes to Button_2
call pause        ; delay
movlw 0x00        ; makes the w register 0
tris portb        ; change banks to 1
movlw b'01001110' ; bit value to display r
movwf portb       ; send to display

Button_2 BTFSC porta,1 ; if porta is Low skip the next instruction if high go to next instruction
goto Button_1     ; goes back to the top of the Loop
call pause        ; delay
movlw 0x00        ; makes the w register 0
tris portb        ; change banks to 1
movlw b'01000111'  ; bit value to display l
movwf portb       ; send to display

goto MainLoop     ; go back to MainLoop

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
```

Appendix C

Task 3

```
; Main.asm file generated by New Project wizard
;
; Created: Alexander Price
; Processor: PIC16F84A
; Compiler: MPASM (Proteus)

; DEFINITIONS
;-----
#include p16f84a.inc ; Include register definition file

portb equ 0x06 ; portb address
trisb equ 0x06 ; trisb address in 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 ; porta address
trisa equ 0x05 ; trisa address in bank 0

; 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
;-----
org 0
movlw 0x01 ; set the port A pins
movwf trisa ; set up for input pins
bcf status,5 ; switch back to bank 0
goto Button_1 ; go to Button_1

MainLoop
Button_1 BTFSC porta,0 ; if porta is Low skip the next instruction if high go to next instruction
goto Button_2 ; goes to Button_2
call pause ; delay
movlw 0x00 ; makes the w register 0
tris portb ; change banks to 1
movlw b'00000000' ; bit value to turn all LEDs on
movwf portb ; send to display

Button_2 BTFSC porta,1 ; if porta is Low skip the next instruction if high go to next instruction
goto Button_1 ; goes back to the top of the loop
call pause ; delay
movlw 0x00 ; makes the w register 0
tris portb ; change banks to 1
movlw b'01111111' ; bit value to turn all LEDs off
movwf portb ; send to display

goto MainLoop ; go back to MainLoop

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
```

Appendix D

Task 4

```
; Main.asm file generated by New Project wizard
;
; Created: Alexander Price
; Processor: PIC16F84A
; Compiler: MPASM (Proteus)

; DEFINITIONS
;-----
#include p16f84a.inc      ; Include register definition file

portb    equ 0x06         ; portb 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

porta    equ 0x05         ; porta address
trisa    equ 0x05         ; trisa address in register bank

; VARIABLES
;-----
count     equ 0x0C        ; Variable to hold the count
nCount    equ 0x0D        ; Pause routine inner loop counter
oldStateButton_1 equ 0x0E ; Old state for button 1
oldStateButton_2 equ 0x0F ; Old state for button 2

; MAIN
org 0
goto Start

Start
; Setting porta as button inputs
bsf status, RP0        ; Select bank 1
movlw 0xFF             ; Fills the w register with 1s
movwf trisa            ; Set porta as input

; Set portb as output for the display
clrf trisb             ; Set porta pins as output
bcf status, RP0        ; Select bank 0
clrf oldStateButton_1  ; Initialise old button state to 0
clrf oldStateButton_2  ; Initialise old button state to 0

movlw b'00010010'      ; Binary for outputting 5
movwf portb            ; Send initial display to portb
movlw 0x05             ; Initialize count at 5
movwf count

MainLoop
    BTFSS porta, 0      ; Check if RA0 is pressed
    goto checkButton1   ; If pressed, check for release
    clrf oldStateButton_1 ; Clear previous state if not pressed
    call pause

    BTFSS porta, 1      ; Check if button RA1 is pressed
    goto checkButton2   ; If pressed, check for release
    clrf oldStateButton_2 ; Clear previous state if not pressed
    call pause

    goto MainLoop       ; Repeat the Loop

checkButton1
    movf oldStateButton_1, w ; Load previous state for button 1
    btfsc status, 2         ; If previous state was 0 (not pressed), proceed
    call pos_increment      ; Call count up routine
    movlw 0x01             ; Add 1 to the w register
    movwf oldStateButton_1 ; Update previous state to 1 (pressed)
```

```

checkButton2
    movf oldStateButton_2, w    ; Load previous state for button 2
    btfsc status, Z            ; If previous state was 0 (not pressed), proceed
    call neg_increment         ; Call count down routine
    movlw 0x01                 ; Add 1 to a w register
    movwf oldStateButton_2     ; Update previous state to 1 (pressed)

    goto MainLoop

; Subroutines
;-----

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

neg_increment
    movf count, w              ; Load current count into W
    btfsc status, Z            ; If count is 0, skip decrease
    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                 ; calls the check loop to display the correct number
    call check                  ; calls the check loop to display the correct number
    return

;check if count = 0
;-----
check
    movf 0x0C, w               ;memory location of count load into w
    movlw 0x00                 ;load w with 0
    subwf 0x0C, w              ;subtract 0 from count
    btfss status, Z            ;if the zero flag is set skip next line else continue
    goto check1                ;check next value
    movlw b'01000000'          ;load correct value for display
    movwf portb                ;send to display
    return

;check if count = 1
;-----
check1
    movlw 0x01                 ;load w with 1
    subwf 0x0C, w              ;subtract 1 from count
    btfss status, Z            ;if the zero flag is set skip next line else continue
    goto check2                ;check next value
    movlw b'01111001'          ;load correct value for display
    movwf portb                ;send to display
    return

;check if count = 2
;-----
check2
    movlw 0x02                 ;load w with 2
    subwf 0x0C, w              ;subtract 2 from count
    btfss status, Z            ;if the zero flag is set skip next line else continue
    goto check3                ;check next value

```

```

    goto check3      ;check next value
    movlw b'00100100' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 3
;-----
check3
    movlw 0x03      ;load w with 3
    subwf 0x0C, w    ;subtract 3 from count
    btfss status, Z  ;if the zero flag is set skip next line else continue
    goto check4      ;check next value
    movlw b'00110000' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 4
;-----
check4
    movlw 0x04      ;load w with 4
    subwf 0x0C, w    ;subtract 4 from count
    btfss status, Z  ;if the zero flag is set skip next line else continue
    goto check5      ;check next value
    movlw b'00011001' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 5
;-----
check5
    movlw 0x05      ;load w with 5
    subwf 0x0C, w    ;subtract 5 from count
    btfss status, Z  ;if the zero flag is set skip next line else continue
    goto check6      ;check next value
    movlw b'00001001' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 6
;-----
check6
    movlw 0x06      ;load w with 6
    subwf 0x0C, w    ;subtract 6 from count
    btfss status, Z  ;if the zero flag is set skip next line else continue
    goto check7      ;check next value
    movlw b'00000011' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 7
;-----
check7
    movlw 0x07      ;load w with 7
    subwf 0x0C, w    ;subtract 7 from count
    btfss status, Z  ;if the zero flag is set skip next line else continue
    goto check8      ;check next value
    movlw b'01111000' ;load correct value for display
    movwf portb      ;send to display
    return

;check if count = 8
;-----
check8
    movlw 0x08      ;load w with 8

```

```

    subwf 0x0C, w      ;subtract 8 fromt count
    btfss status, Z    ;if the zero flag is set skip next line else continue
    goto check9        ;check next value
    movlw b'00000000'   ;load correct value for display
    movwf portb        ;send to display
    return

;check if count = 9
;-----
check9
    movlw 0x09         ;load w with 9
    subwf 0x0C, w      ;subtract 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 correct value for display
    movwf portb        ;send to display
    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

```

Appendix E

Task 5

```
; Main.asm file generated by New Project wizard
;
; Created: Alexander Price
; Processor: PIC16F84A
; Compiler: MPASM (Proteus)

; DEFINITIONS
-----
#include p16f84a.inc ; Include register definition file

portb equ 0x06 ; portb 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

porta equ 0x05 ; porta address
trisa equ 0x05 ; trisa address in register bank

; VARIABLES
-----

delay equ 0x20 ; Delay counter

; MAIN
-----

org 0 ; Reset vector
goto Start ; Jump to start

Start

bsf status, RP0 ; Select Bank 1
movlw 0xFF ; Set porta direction (RA0 as input, rest as output)
movwf trisa ; Set porta direction (0=output, 1=input)
clrf trisb ; Switches portb to output pins
bcf status, RP0 ; Switch to Bank 0

MainLoop

movlw b'01111111'
movwf portb
btfsc porta, 0 ; Check if button is pressed (RA0)
goto Sound_Off ; If not pressed, turn sound off
call Sound_On ; If pressed, generate sound
goto MainLoop ; Repeat loop

Sound_On

bsf portb, 7 ; Set speaker pin high
call DelayRoutine ; Call delay for tone duration
bcf portb, 7 ; Set speaker pin low
call DelayRoutine ; Call delay for tone duration
return

Sound_Off

bcf portb, 7 ; Ensure speaker is off
goto MainLoop ; Go back to main loop

DelayRoutine

movlw 0xFF ; Load delay counter
movwf delay ; Moves w to the delay register

Delayloop

decfsz delay, f ; Decrement delay counter
goto Delayloop ; Repeat until zero
return

end
```

Microperipheral and Interfaces Alexander Price

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

65/100

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

GRADING FORM: GJ_2019

ALEX PRICE

325

ENGAGEMENT WITH LITERATURE



Some reasonable sources cited, but there could have been even more and covered all aspects.

65

KNOWLEDGE AND UNDERSTANDING



A fair explanation of most of the core concepts. More EWL would have helped obtain better marks here though.

65

COGNITIVE AND INTELLECTUAL



Good attempts at 5 out of the 6 programming problems, however most solutions do reply on the 'brute force' approach, so this section has lower marks.

55

PRACTICAL APPLICATION



Good circuit implementation and most software problems attempted.

70

TRANSFERABLE SKILLS



Reasonably well laid out and presented.

70