

Micros, Peripherals and Interfaces Assignment

by Freya Vafadari

Submission date: 28-Nov-2022 02:36PM (UTC+0000)

Submission ID: 190322809

File name:

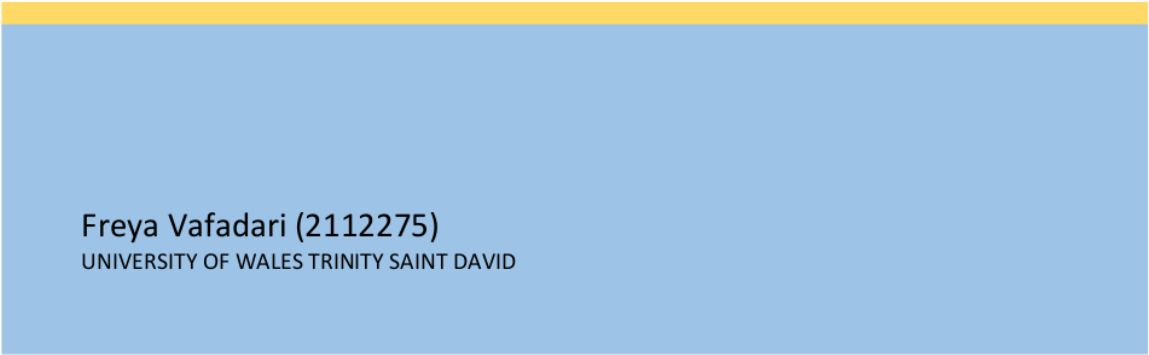
32483_Freya_Vafadari_Micros__Peripherals_and_Interfaces_Assignment_466098_220422761.docx (4.86M)

Word count: 2663

Character count: 13254

Micros, Peripherals, and Interfaces University Assignment

PROGRAMMING THE 7 SEGEMENT LED DISPLAY PIC16F84A
CIRCUIT IN PIC ASSEMBLY IN PROTEUS SIMULATION
SOFTWARE



Freya Vafadari (2112275)
UNIVERSITY OF WALES TRINITY SAINT DAVID

List of Figures

Figure 1 PIC 16F84A from Microchip Technologies

Figure 2 PIC 16F84A pin out diagram

Figure 3 7 segment LED display circuit with two non-latching buttons controlled by a PIC 16F84A board

Figure 4a Mapping of the terminals A – H to the LED segments on the external peripheral

Figure 4b Binary and HEX conversion truth tables for displaying 0 – F in the binary and hexadecimal numbering systems

Figure 5a Registers, ports and variables definitions for portB, Status, count and LED_0 – LED_F in Task 1

Figure 5b Start and get_count function loops in Task 1

Figure 5c One_sec_pause, loadN and decN functions in Task 1

Figure 5d Proof of completing Task 1 – the 7 segment LED display increments from 0 – F in hex in 0.5 sec delays

Figure 5e Pseudo code of Task 1

Figure 6a Ports, registers, indf, fsr, Counts, LEDs, SWs, BTNs, temp and flag definitions in Task 2

Figure 6b Start and test_switch functions in Task 2

Figure 6c Black_switch, green switch, update_screen, pause, loadN and decN functions in Task 2

Figure 6d Proof of completing Task 2 – the 7 segment LED display will display '1' if the black button is selected and '2' if the green button is selected

Figure 6e Pseudo code of Task 2

Figure 7a Ports, registers, indf, fsr, Counts, LEDs, SWs, BTNs, temp and flag definitions in Task 3

Figure 7b Start, test_switch and black_switch functions in Task 3

Figure 7c Green_switch, update_screen, pause, loadN and decN functions in Task 3

Figure 7d Proof of completing Task 3 – the 7 segment LED display will display '1' if the black button is selected and '2' if the green button is selected

Figure 7e Pseudo code of Task 3

Figure 8 Task 4 Figure 8a Ports, registers, indf, fsr, Count_A, LEDs 0 – F, SWs, start_address, temp, LED_index and event_flag definitions in Task 4

Figure 8b Start, check_count and centre_loop functions in Task 4

Figure 8c Event_sw_black, continue and event_sw_green functions in Task 4

Figure 8d Proof of completing Task 4 – the 7 segment LED display will increment up from 0 - F if the black button is selected and will decrement down from 0 - F if the green button is selected

Figure 8e Pseudo code of Task 4

Figure 9a Ports, registers, indf, fsr, LEDs, SWs, first_loop, second_loop, temp, pause_count and event_flag definitions in Task 5

Figure 9b Main_code and event_black_sw functions in Task 5

Figure 9c Buzzer, loadN_buzzer, decN_buzzer, short_pause and decN functions in Task 5

Figure 9d Proof of completing Task 5 – the buzzer will sound if the black button is selected and display the noise output on the digital oscilloscope in the virtual terminal

Figure 9e Buzzer sound displayed on the digital oscilloscope in the virtual terminal for Task 5

Figure 9f Pseudo code of Task 5

Introduction

The aim of the assignment is to write successful PIC assembly code for several tasks that modifies the Proteus circuit in Figure 3 which is a 7 segment LED display circuit with two non-latching buttons, a buzzer and a crystal oscillator all controlled by a PIC 16F84A board.

Assembly language is any low – level programming language with a strong correspondence between the instructions in the language and the architecture's machine code instructions. Assembly language usually has one statement per machine instruction and is specific to a single computer architecture. PIC 16F84A shown in Figure 2 is an 8 - bit microcontroller manufactured by Microchip Technologies that contains 35 assembly language instructions and has a maximum operating frequency of 20MHz and an operating voltage of 2 – 5.5v. PIC microcontroller is based on Harvard architecture and supports RISC architecture (Reduced Instruction Set Computer). Harvard architecture is a type of computer architecture that has separate storage and signal pathways for both instructions and data. [4, Sec. 1 PIC 16F84A Microcontroller features]

Binary numbers are the transfer of information in the form of 0's and 1's implemented in digital computers as machine code. Data in digital circuits is transmitted as a series of 0's and 1's where it can be interpreted and executed as machine code in the PIC microcontroller's CPU specifically in the ALU which is the unit inside the CPU responsible for processing all arithmetic operations and logical decisions. [1, Ch. Binary numbers] Hexadecimal numbers are a different number system that group binary numbers into 4 - digit strings to allow for the conversion of 16 binary digits into a more readable form. Binary and hexadecimal numbers are positional number systems and use different bases. Binary numbers use base – 2 and hexadecimal numbers use base – 16. [2, Ch. Hexadecimal numbers]



Figure 1 PIC 16F84A from Microchip Technologies [4, Sec. 1 PIC 16F84A Microcontroller features]

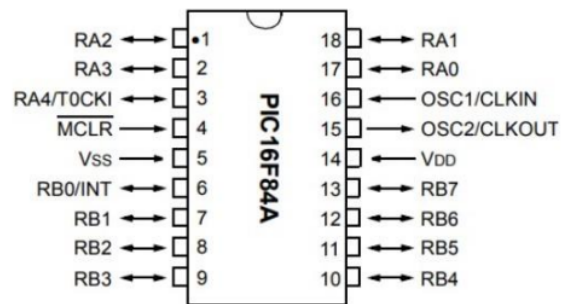


Figure 2 PIC 16F84A pin out diagram [4, Sec. 1 PIC 16F84A Microcontroller features]

Section 1

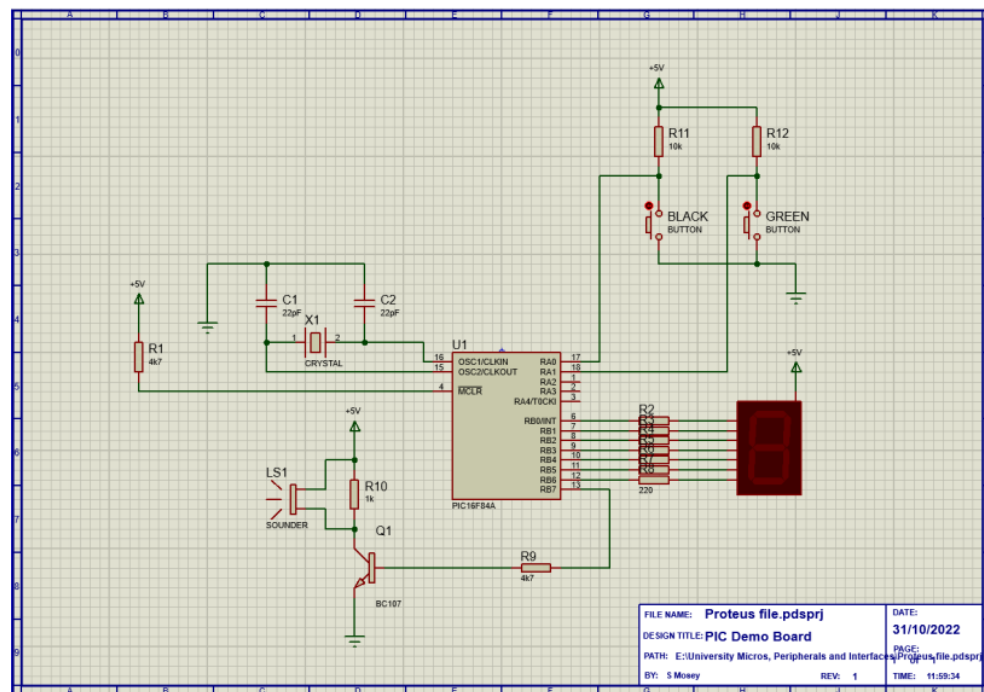


Figure 3 7 segment LED display circuit with two non-latching buttons, a buzzer and a crystal oscillator all controlled by a PIC 16F84A board

- $R1 = 4k7\Omega$

R1 is a current limiting pull up resistor connected in series with a + 5v power supply terminal to the Memory Clear Pin 4 on the PIC 16F84A. R1 is a current limiting pull up resistor which is implemented in the circuit to reduce the current and connected in series with the V_{CC} power terminal to physically interrupt the current connection to ground from the Memory Clear Pin 4. Pin is an active low pin at resets the device when connected to ground.

- $C1 \text{ \& } C2 = 22pF$

C1 & C2 are two aluminium electrolytic capacitors connected to the two terminals of the crystal oscillator X1 and connected to ground in Figure 3. The capacitors are connected to the terminals of the crystal oscillator to attenuate all unwanted oscillations to the resonant frequency that the CPU can operate at with no noise.

- X1 = crystal oscillator

X1 is a crystal oscillator which work on the principle of the inverse piezoelectric effect where an alternating voltage applied across the crystal surfaces causes it to vibrate at its natural frequency and these vibrations are transferred into oscillations as the kinetic energy in the system increases. Crystal oscillators like X1 are implemented into many microcontrollers because they act as precise and very accurate internal clock for the microcontroller in this case as a PIC 16F84A and its peripherals.

- LS1 = buzzer

LS1 is a buzzer that has a sample rate of 44100 and a playback time of 500ms

- V_{CC} Power = +5v
- R10 = 1k Ω

R10 is a current limiter pull up resistor connected in series to LS1 to reduce the current flow to LS1 as it prevents overload current increasing the overall resistance load.

- Q1 = BC107

Q1 is a BC107 NPN bipolar transistor. NPN (negative – positive – negative) transistors are three terminal devices which can function as amplifiers or switches. In Figure 3 Q1 is acting as an amplifier to the buzzer LS1 because in a transistor it's the movement of electrons through the base region that causes the transistor to function and amplify because the base terminal of Q1 exerts a resistive force upon the collector to emitter current flowing to ground. Q1 is utilised as an amplifier to amplify the signal through LS1 so the buzzer can sound.

- U1 = PIC 16F84A

U1 is a an 8 – bit PIC 16F84A microcontroller. It has a bus width of 8 bits and has 1024 words of program memory. The PIC 16F84A has a current sourcing and sinking limit of 25mA. It has 13 GPIO pins that can be utilised as digital inputs or outputs and are divided between port A and port B. Figure 2 shows the pin out diagram for the PIC 16F84A microcontroller. Each other 13 GPIO pins can either source or sink a maximum current of 25mA so every pin could power an LED because the current requirement to power an LED is less than 10mA. The processor in a 16F84A has a 5 million instructions per second speed and a program memory of 1750 bytes of which 68 bytes is RAM memory. The memory logic unit in PIC microcontrollers like 16F84A consists of three types of memories which are program, data and data EEPROM. Program memory contains the written program after its programmed onto the microcontroller where the counter executes commands stored in the memory sequentially. PIC microcontrollers can have 8K words $\times$ 14 bits of flash memory that can be programmed. Data memory is a RAM type that stores the data briefly in the data registers. The RAM memory is separated into banks where each bank is a maximum of 128 bytes. Banks contain both Special Function Registers (SFR) and General - Purpose Registers (GPR). GPRs are utilised for simple arithmetic and logical operations then storing the results permanently in other registers where the data can be accessed by the CPU. SFRs perform a single function which is determined at the time of manufacturing and three important SFRs are the status register which

changes the bank, port registers that assign logic values 0 or 1 to the ports and tris registers which are data direction registers for inputs and outputs. Data EEPROM memory allows variables to be stored as result of the microcontroller being programmed and is readable and writable during operation and indirectly accessible through the SFRs. [3, pp. 1, 5 - 12 & 37 - 42 PIC 16F84A Data & Instruction Set]

- $R9 = 4k7\Omega$

R9 is a current limiting pull resistor connected in series with Q1 to prevent an overload of forward current into the base terminal of Q1 which is an NPN bipolar transistor.

- $R11 \text{ \& } R12 = 10k\Omega$

R11 & R12 are current limiting pull up resistors connected to the black and green switches.

R11 & R12 are pull up resistors are utilised on the switches to ensure the input pin is in a constant HIGH state which the switch is not selected. If this was not implemented the input would be floating and its possible that the input randomly changes between HIGH and LOW as noise increases.

Section 2

A 7 – segment LED display is a form of electronic display peripheral that consists of seven LEDs arranged in a rectangular for. Each LED is called a segment that maps to one of the terminals A – G. Therefore, by assigning the binary digit inputs to segments A – G, its possible to control the illumination of the segments so the decimal numbers 0 – 9 and the letters A – F can be displayed on the LED peripheral. The hexadecimal 7 – segment LED display decodes 4 binary - digit inputs into a hexadecimal number which is displayed using the LED segments.

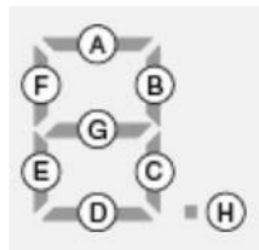


Figure 4a Mapping of the terminals A – H to the LED segments on the external peripheral

Figure 4a shows the mapping of the terminals A – H to the LED segments.

It's possible to calculate the hexadecimal numbers for each character by forming two 4 binary – digit inputs depending on which of the mapping terminals A – H on the LED display are lit up and then the two binary numbers can be converted into a two - digit hexadecimal number.

'0' = LED segment is ON/HIGH

'1' = LED segment is OFF/LOW

Example calculating the hexadecimal number for the character 'F'

'A' = 0 'B' = 1 'C' = 1 'D' = 1

'E' = 0 'F' = 0 'G' = 0 'H' = 0

Therefore, by following the pattern of the table in Figure 4b I formed to calculate the hexadecimal numbers for the characters 0 – F then the two 4 binary – digit inputs for the character 'F' are: 0000 1110 = 0x0e

	LED display								
Hexadecimal	Buzzer 'h'	"g"	"f"	"e"	"d"	"c"	"b"	"a"	Character
0x40	0	1	0	0	0	0	0	0	0
0x79	0	1	1	1	1	0	0	1	1
0x24	0	0	1	0	0	1	0	0	2
0x30	0	0	1	1	0	0	0	0	3
0x19	0	0	0	1	1	0	0	1	4
0x12	0	0	0	1	0	0	1	0	5
0x02	0	0	0	0	0	0	1	0	6
0x78	0	1	1	1	1	0	0	0	7
0x00	0	0	0	0	0	0	0	0	8
0x10	0	0	0	1	0	0	0	0	9
0x08	0	0	0	0	1	0	0	0	A
0x03	0	0	0	0	0	0	1	1	B
0x46	0	1	0	0	0	1	1	0	C
0x21	0	0	1	0	0	0	0	1	D
0x06	0	0	0	0	0	1	1	0	E
0x0e	0	0	0	0	1	1	1	0	F

Figure 4b Binary and HEX conversion truth tables for displaying 0 – F in the binary and hexadecimal numbering systems

Figure 4b shows the binary and hexadecimal equivalence truth table for the decimal numbers 0 – 9 and the letters A – F where each character that can be displayed on the LED peripheral is formed of two 4 binary – digit inputs.

Section 3

Task 1

The aim of Task 1 is to modify the standard circuit in Figure 3 so the 7 segment LED display increments from 0 – F in hexadecimal number form with 0.5 second delays between each character.

```

15  ;;;; TASK 1 - increment from 0 - F in hex with a 0.5 second delay
16
17  portB    equ 0x06                ;port B address
18  trisB    equ 0x06                ;trisB address in register bank 1
19
20  Status    equ 0x03                ;status register
21  RP0       equ 0x05                ;bit 5 of status register
22  RP1       equ 0x06                ;bit 6 of status register
23
24  count     equ 0x0c                ; counter 0 to 255
25  nCount    equ 0x0d                ; pause routine inner loop counter
26  mCount    equ 0x0e                ; pause routine outer loop counter
27
28  LED_0     equ 0x40                ; 7 segment LED character definitions
29  LED_1     equ 0x79
30  LED_2     equ 0x24
31  LED_3     equ 0x30
32  LED_4     equ 0x19
33  LED_5     equ 0x12
34  LED_6     equ 0x02
35  LED_7     equ 0x78
36  LED_8     equ 0x00
37  LED_9     equ 0x10
38  LED_A     equ 0x08
39  LED_B     equ 0x03
40  LED_C     equ 0x46
41  LED_D     equ 0x21
42  LED_E     equ 0x06
43  LED_F     equ 0x0e
44

```

Figure 5a Registers, ports and variables definitions for portB, Status, count and LED_0 – LED_F in Task 1

Micros, Peripherals and Interfaces Assignment November 2022

```

46
47
48      org      0x000
49
50      start
51      bcf      Status,RP0      ; clear bank selection bits in the status reg
52      bcf      Status,RP1      ;
53
54      bsf      Status,RP0      ; select bank 1
55      movlw    0x00            ; load w with 0x00
56      movwf    trisb           ; load it into trisb to make portb all outputs
57      bcf      Status,RP0      ; reselect bank 0
58
59      clrf     portB           ; clear all lines low
60      clrf     count          ; initialize counter to 0
61
62      get_count
63      movlw    LED_0           ; load the character "0" into the W register
64      movwf    portB           ; move character "0" from W register to port B
65      call     one_sec_pause    ; wait a second
66
67      movlw    LED_1           ; load the character "1" into the W register
68      movwf    portB           ; move character "1" from W register to port B
69      call     one_sec_pause    ; wait a second
70
71      movlw    LED_2           ; load the character "2" into the W register
72      movwf    portB           ; move character "2" from W register to port B
73      call     one_sec_pause    ; wait a second
74
75      movlw    LED_3           ; load the character "3" into the W register
76      movwf    portB           ; move character "3" from W register to port B
77      call     one_sec_pause    ; wait a second
78
79      movlw    LED_4           ; load the character "4" into the W register
80      movwf    portB           ; move character "4" from W register to port B
81      call     one_sec_pause    ; wait a second
82
83      movlw    LED_5           ; load the character "5" into the W register
84      movwf    portB           ; move character "5" from W register to port B
85      call     one_sec_pause    ; wait a second
86
87      movlw    LED_6           ; load the character "6" into the W register
88      movwf    portB           ; move character "6" from W register to port B
89      call     one_sec_pause    ; wait a second
90
91      movlw    LED_7           ; load the character "7" into the W register
92      movwf    portB           ; move character "7" from W register to port B
93      call     one_sec_pause    ; wait a second
94
95      movlw    LED_8           ; load the character "8" into the W register
96      movwf    portB           ; move character "8" from W register to port B
97      call     one_sec_pause    ; wait a second
98
99      movlw    LED_9           ; load the character "9" into the W register
100     movwf    portB           ; move character "9" from W register to port B
101     call     one_sec_pause    ; wait a second

```

Figure 5b Start and get_count functions loops in Task 1

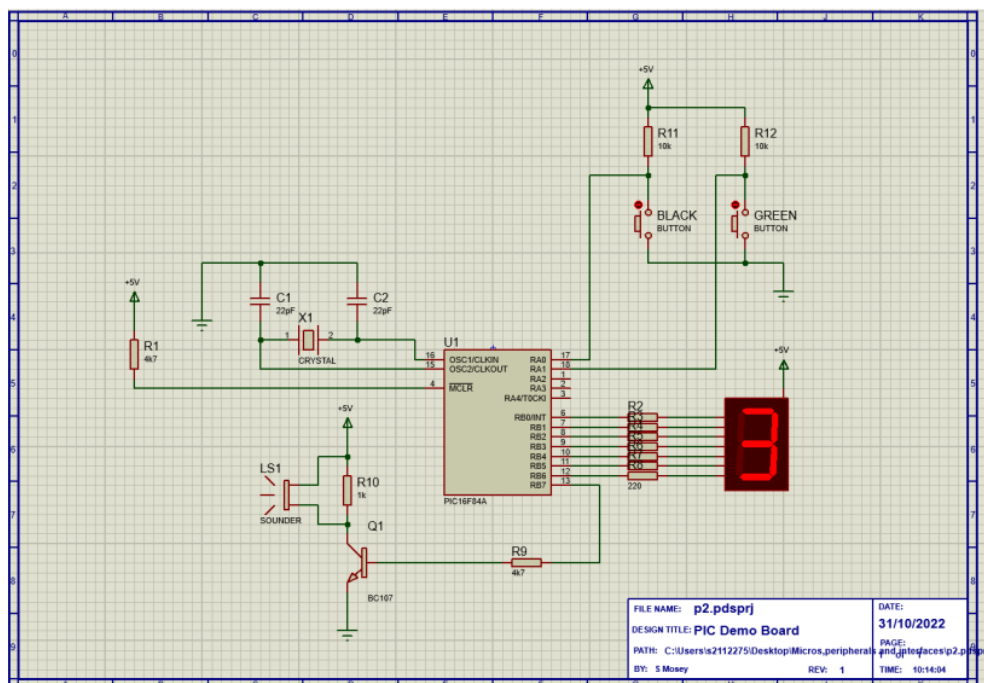
Micros, Peripherals and Interfaces Assignment November 2022

```

002      movlw   LED_A           ; Load the character "A" into the W register
003      movwf  portB          ; move character "A" from W register to port B
004      call   one_sec_pause ; wait a second
005
006      movlw   LED_B           ; Load the character "B" into the W register
007      movwf  portB          ; move character "B" from W register to port B
008      call   one_sec_pause ; wait a second
009
010      movlw   LED_C           ; Load the character "C" into the W register
011      movwf  portB          ; move character "C" from W register to port B
012      call   one_sec_pause ; wait a second
013
014      movlw   LED_D           ; Load the character "D" into the W register
015      movwf  portB          ; move character "D" from W register to port B
016      call   one_sec_pause ; wait a second
017
018      movlw   LED_E           ; Load the character "E" into the W register
019      movwf  portB          ; move character "E" from W register to port B
020      call   one_sec_pause ; wait a second
021
022      movlw   LED_F           ; Load the character "F" into the W register
023      movwf  portB          ; move character "F" from W register to port B
024      call   one_sec_pause ; wait a second
025
026      goto    get_count       ; repeat forever
027
028  one_sec_pause
029      call    pause           ; delay 200ms by subroutine
030      call    pause
031      call    pause
032      call    pause
033      call    pause           ; five pause executions equals ~ 1 second
034      return
035
036      pause
037      movlw   0x80            ; set W = 255 decimal
038      movwf  nCount          ; mCount = w; outer Loop counter
039
040      movlw   0xff            ; set W = 255 decimal
041      movwf  nCount          ; nCount = w; inner Loop counter
042
043  decN     decfsz nCount, f    ; nCount--
044         goto  decN           ; if nCount != 0 then repeat nCount--
045
046      decfsz mCount, f        ; else decrement Count
047      goto    loadN           ; if mCount != 0 then
048                               ; reload nCount to 255 and decrement
049      return                  ; else exit subroutine
050
051      end

```

Figure 5c One_sec_pause, loadN and decN functions in Task 1



Micros, Peripherals and Interfaces Assignment November 2022

Figure 5d Proof of completing Task 1 – the 7 segment LED display increments from 0 – F in hex in 0.5 sec delays

```
Define port B as an output port and equate memory addresses in tris b.

Define RP0 and RP1 registers to Pins 17 (connected in series to the black button switch) and 18
(connection in series to the green button switch) – then equate to memory addresses

Define nCount – inner loop counter and mCount – outer loop counter then equate to memory
addresses

Define hexadecimal numbers to LEDs 0 – F characters

Initialise start loop. Clear banks in the status registers and assign space in status register to blank
memory then load into tris b to make port B an output. Define RP0 as an output on port B clear port
B register and count address

Initialise get_count – load the character '0' into the W register

Move the character '0' from the W register to port B

Call one_sec_pause - Wait one second

Repeat get_count for all characters 0 – F

Initialise one_sec_pause. To call five pause executions ~ 1 second.

Initialise loadN. Initialise c

Initialise decN. Decrement nCount in the working register f then call decN. If nCount! = 0 then repeat
if nCount = 0 carry on and decrement mCount in the working register f, then call loadN. If mCount! =
0 then repeat nCount and decrement.

If not exit subroutine and finish.
```

Figure 5e Pseudo code for Task 1

Task 2

The aim of Task 2 is to modify the standard circuit in Figure 3 so if the black button switch is selected either latching or non-latching “1” is displayed or if the green button switch is selected “2” is displayed on the LED display.

Micros, Peripherals and Interfaces Assignment November 2022

```

1  ; FILE:          count.asm
2  ; AUTHOR:       Freya Vafadari
3  ; DATE:         November 2022
4  ; DESCRIPTION:  Count to 255 at 1 second intervals and display on a seven
5  ;               segment display in binary.
6  ;               The pause subroutine takes apx. 0.5 sec.
7  ;               Pause is called 5 times
8
9  list    p=16F84      ; select processor type
10 radix   hex          ; define radix as hexadecimal
11 ; #include "p16f84.inc"
12 ;-----
13 ; cpu equates (memory map)
14 ;;; TASK 2 - Select the black button so "1" is displayed on the LED display
15 ;;; or select the green button so "2" is displayed on the LED display
16 portA   equ    0x05      ;port A address
17 trisa   equ    0x05      ;trisa address in register bank 1
18
19 portB   equ    0x06      ;port B address
20 trisb   equ    0x06      ;trisB address in register bank 1
21
22
23 indf    equ    0x00      ;indirect file register
24 fsr     equ    0x04      ;indirect address register
25
26 Status  equ    0x03      ;status register
27 RP0     equ    0x05      ;bit 5 of status register
28 RP1     equ    0x06      ;bit 6 of status register
29
30 count   equ    0x0c      ;counter 0 to 128
31 nCount  equ    0x0d      ;pause routine inner Loop counter
32 mCount  equ    0x0e      ;pause routine outer Loop counter
33
34 LEDA    equ    0x79      ; if LED = on display 1
35 LEDB    equ    0x24      ; if LED = on display 2
36 SW_Black equ    0x01      ; Black switch bit of Port A
37 SW_Green equ    0x02      ; Green switch bit of Port B
38 BTN_Black equ    0x03      ; Black button bit of Port A
39 BTN_Green equ    0x04      ; Green button bit of Port B
40
41 temp    equ    0x10
42 flag    equ    0x11
43 SSD     equ    0x12
44 ;-----
45
46 org     0x000

```

Figure 6a Ports, registers, indf, fsr, Counts, LEDs, SWs, BTNs, temp and flag definitions in Task 2

```

48 start
49 bcf     Status,RP0      ; clear bank selection bits in the status register RP0
50 bcf     Status,RP1      ; clear bank selection bits in the status register RP1
51
52 bsf     Status,RP0      ; select register RP0 bank 1
53 movlw   0x00            ; Load w with 0x00 trisb = bank 1
54 movwf   trisb           ; Load it into trisb to make portA all outputs
55 movlw   0x03            ; Load w with 0x00 trisb = bank 1
56 movwf   trisa           ; Load it into trisb to make portA all inputs
57
58 bcf     Status,RP0      ; reselect bank 0
59
60 clrf    portB           ; clear all lines Low
61 clrf    count           ; initialize counter to 0
62
63 test_switch             ; run test_switch function
64
65 movf    portA, w        ; move w to portA
66 andlw   SW_Black        ; AND literal working file + SW_Black together
67 movwf   temp            ; move w into the file register temp
68 btfsf   temp, 0         ; test bit 0 of temp (black switch)
69 call    black_switch    ; call black_switch function
70
71 movf    portA, w        ; move w to portA
72 andlw   SW_Green        ; AND literal working file + Test green switch
73 movwf   temp            ; move w into the file register temp
74 btfsf   temp, 1         ; test bit 0 of temp (black switch)
75 call    green_switch    ; call green_switch function
76
77 movlw   0xff            ; moves a literal value into the working register 0x01 - RAM location 0x01
78 movwf   portB           ; move to working file register to portB
79 goto    test_switch     ; case statement one-way transfer to test_switch function
80

```

Figure 6b Start and test_switch functions in Task 2

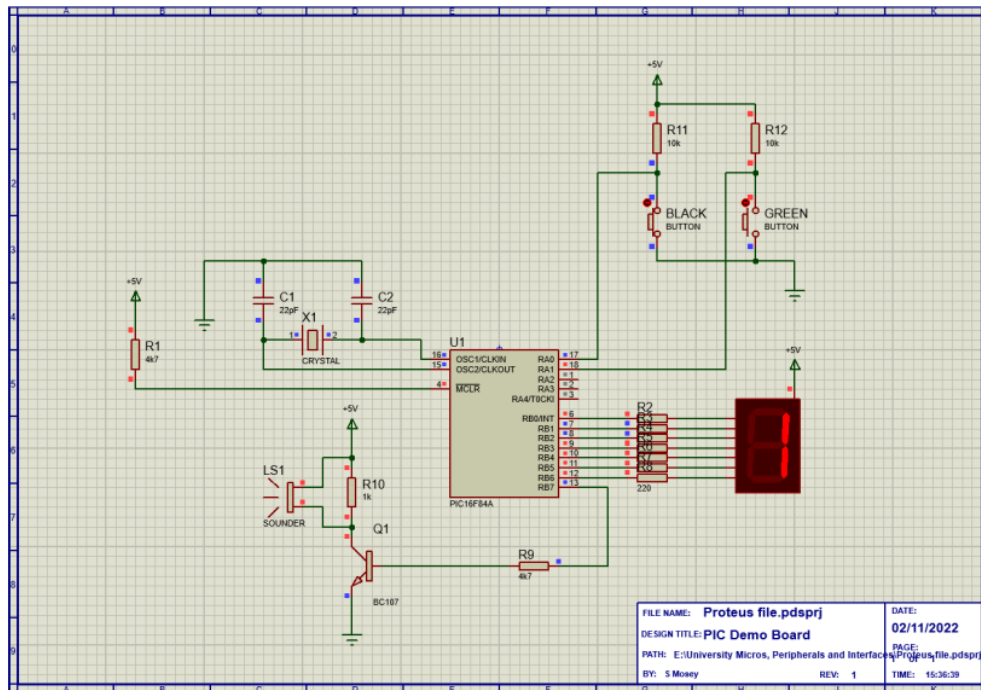
Micros, Peripherals and Interfaces Assignment November 2022

```

80      black_switch      ; call black_switch function
81      movlw 0x01        ; moves a literal value into the working register @x01 - RAM Location @x01
82      movwf flag        ; move flag variable into the working file
83      movlw LEDA        ; move a literal (address of the byte that displays 2 in HEX) - clear segment a bit (turn on LEDA)
84      goto update_screen ; case statement one-way transfer to update_screen function - check button status
85
86
87      green_switch      ; run green_switch function
88      movlw 0x01        ; moves a literal value into the working register @x01 - RAM Location @x01
89      movwf flag        ; move flag variable into the working file
90      movlw LEDB        ; move a literal (address of the byte that displays 1 in HEX) - clear segment a bit (turn on LEDB)
91      goto update_screen ; case statement one-way transfer to update_screen function - check current button status
92
93
94      update_screen      ; run update_screen
95      movwf portB        ; move working file to portB
96      call pause         ; call pause function
97      goto test_switch   ; case statement one-way transfer to test_switch function
98
99      pause             ; set w = 128 decimal @x80 hex is 128 in decimal - displays "1" or "2" for 0.5 secs depending on the switch selection
100     movlw 0x80          ; mCount = w: outer Loop counter
101     movwf mCount
102     loadN              ; set w = 128 decimal
103     movlw 0x80          ; nCount = w: inner Loop counter
104     movwf nCount
105     decN              ; nCount--
106     decfsz nCount, f    ; if nCount != 0 then repeat nCount--
107     goto decN
108
109     decfsz mCount, f    ; else decrement Count
110     goto loadN          ; if mCount != 0 then
111     ; reload nCount to 128 and decrement
112     ; else exit subroutine
113     return
114     end

```

Figure 6c Black_switch, green switch, update_screen, pause, loadN and decN functions in Task 2



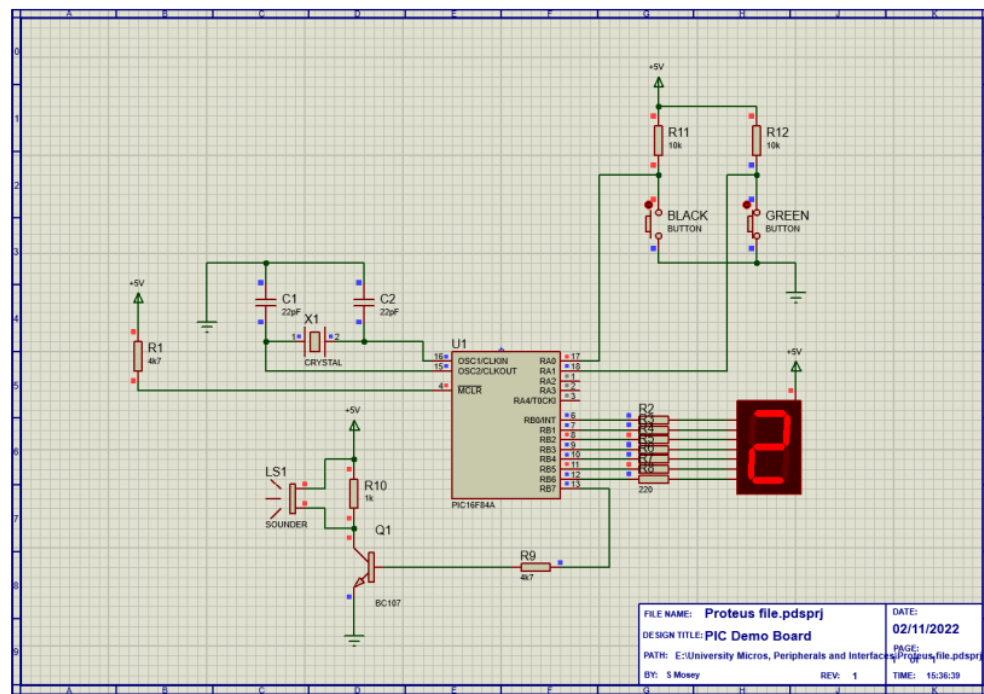


Figure 6d Proof of completing Task 2 – the 7 segment LED display will display '1' if the black button is selected and '2' if the green button is selected

Define port A and port B addresses and tris A and tris b addresses in register bank 1

Define indf and fsr address registers

Define RPO and RP1 as status registers

Define LEDs and SWs – equate to memory addresses

Initialise test_switch. Move w to port A then and literal working file + SW_Black together. Move w into the file register temp and test bit 0 of temp (black switch). Call black_switch function. Repeat for the green_switch. Return to update_screen.

Initialise black_switch. Move a literal working value into the working register 0x01 – RAM location 0x01. Move flag variable into the working file. Move a literal (address of the byte that displays 2 in HEX) – clear segment a bit (turn on LED). Check button status. Repeat for green_switch.

Initialise update_screen. Move working file to port B and call pause. Return to test_switch.

Initialise loadN. Set time counter.

Initialise decN. Set increment number counter on the display. If completed exit subroutine

Task 3

The aim of Task 3 is to modify the standard circuit in Figure 3 so if the black button switch is latched all the LED segment displays are switched to HIGH and if the green button switch is latched all the LED segment displays are switched to LOW.

Micros, Peripherals and Interfaces Assignment November 2022

```

1 ; FILE: count.asm
2 ; AUTHOR: Freya Vafadari
3 ; DATE: November 2022
4 ; DESCRIPTION: Count to 255 at 1 second intervals and display on a seven
5 ; segment display in binary.
6 ; The pause subroutine takes apx. 0.5 sec.
7 ; Pause is called 5 times
8
9 list p=16F84 ; select processor type
10 radix hex ; define radix as hexadecimal
11 #include "pi6f84.inc"
12
13 ;-----
14 ; cpu equates (memory map)
15 ;;; TASK 3 - Select the black button so all the LEDs are switched on or select the green button so all the LEDs are switched off
16 portA equ 0x05 ;port A address
17 trisa equ 0x05 ;trisa address in register bank 1
18 portB equ 0x06 ;port B address
19 trisb equ 0x06 ;trisb address in register bank 1
20
21
22 indf equ 0x00 ;indirect file register
23 fsr equ 0x04 ;indirect address register
24
25 Status equ 0x03 ;status register
26 RP0 equ 0x05 ;bit 5 of status register
27 RP1 equ 0x06 ;bit 6 of status register
28
29 count equ 0x0c ;counter 0 to 128
30 nCount equ 0x0d ;pause routine inner loop counter
31 mCount equ 0x0e ;pause routine outer loop counter
32
33 LEDA equ 0x79 ; if LED = on display 1
34 LEDB equ 0x24 ; if LED = on display 2
35 SW_Black equ 0x01 ; Black switch bit of Port A
36 SW_Green equ 0x02 ; Green switch bit of Port B
37 BTN_Black equ 0x03 ; Black button bit of Port A
38 BTN_Green equ 0x04 ; Green button bit of Port B
39
40 temp equ 0x10
41 flag equ 0x11
42 SSD equ 0x12
43 ;-----

```

Figure 7a Ports, registers, indf, fsr, Counts, LEDs, SWs, BTNs, temp and flag definitions in Task 3

```

47 start
48 bcf Status,RP0 ; clear bank selection bits in the status register RP0
49 bcf Status,RP1 ; clear bank selection bits in the status register RP1
50
51 bsf Status,RP0 ; select register RP0 bank 1
52 movlw 0x00 ; Load w with 0x00 trisb = bank 1
53 movwf trisb ; Load it into trisb to make portA all outputs
54 movlw 0x03 ; Load w with 0x03 trisb = bank 1
55 movwf trisa ; Load it into trisb to make portB all inputs
56
57 bcf Status,RP0 ; reselect bank 0
58
59 clrf portB ; clear all lines Low
60 clrf count ; initialize counter to 0
61
62 test_switch ; call test_switch function
63
64 movf portA, w ; move w to portA
65 andlw SW_Black ; AND Literal working file + Test black switch
66 movwf temp ; move w into the file register temp
67 btfss temp, 0 ; test bit 0 of temp (black switch)
68 call black_switch ; call black_switch variable
69
70 movf portA, w ; move w to portA
71 andlw SW_Green ; AND Literal working file + Test green switch
72 movwf temp ; move w into the file register temp
73 btfss temp, 1 ; test bit 0 of temp (black switch)
74 call green_switch ; call green_switch variable
75
76 goto test_switch ; case statement one-way transfer to test_switch function
77
78
79 black_switch ; call black_switch function
80 movlw 0x01 ; move literal value of 0x01 to the working register
81 movwf flag ; move flag variable into the working file
82 movlw 0x00 ; move literal value of 0x00 to the working register
83 goto update_screen ; case statement one-way transfer to update_screen function - check button status

```

Figure 7b Start, test_switch and black_switch functions in Task 3

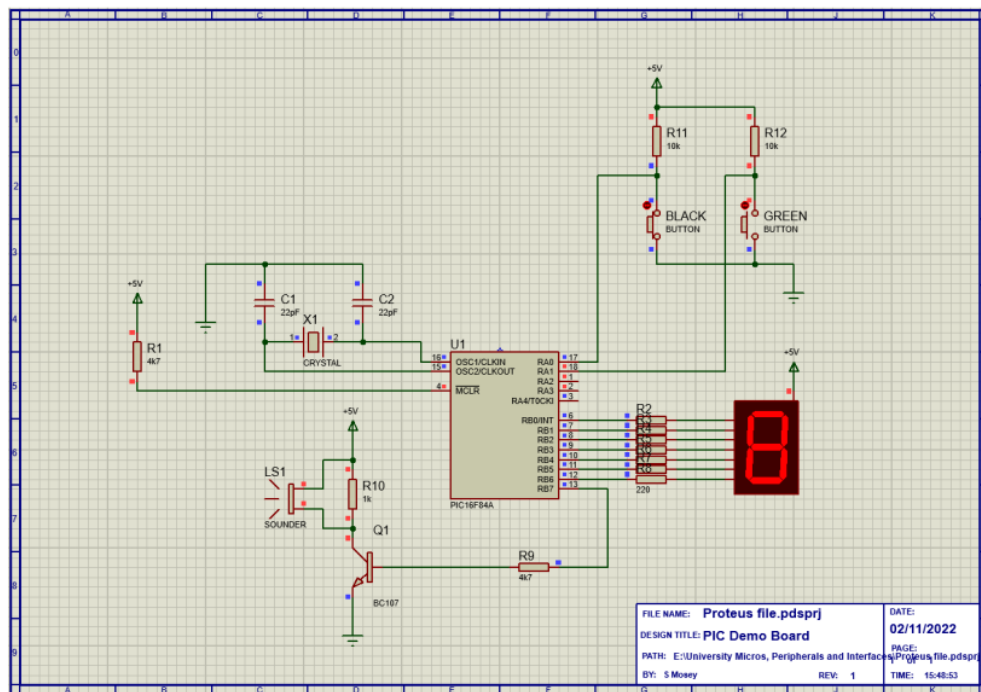
Micros, Peripherals and Interfaces Assignment November 2022

```

79 black_switch      ; call black_switch function
80   movlw 0x01      ; move literal value of 0x01 to the working register
81   movwf flag      ; move flag variable into the working file
82   movlw 0x00      ; move literal value of 0x00 to the working register
83   goto update_screen ; case statement one-way transfer to update_screen function - check button status
84
85 green_switch      ; run green_switch function
86   movlw 0x01      ; move literal value of 0x01 to the working register
87   movwf flag      ; move flag variable into the working file
88   movlw 0xff      ; move literal value of 0xff to the working register
89   goto update_screen ; case statement one-way transfer to update_screen function - check current button status
90
91
92 update_screen      ; run update_screen
93   movwf portB      ; move working file to portB
94   call pause       ; call pause function
95   goto test_switch ; case statement one-way transfer to test_switch function
96
97
98 pause             ; set w = 128 decimal 0x80 hex is 128 in decimal - 0.5 secs delay in LED flashes
99   movlw 0x80      ; mCount = w: outer Loop counter
100  movwf mCount
101
102 loadN             ; set w = 128 decimal
103   movlw 0x80      ; nCount = w: inner Loop counter
104   movwf nCount
105
106 decN             ; nCount--
107   decfsz nCount, f ; if nCount != 0 then repeat nCount--
108   goto decN
109
110   decfsz mCount, f ; else decrement Count
111   goto loadN      ; if mCount != 0 then
112   return          ; reload nCount to 128 and decrement
113   ; else exit subroutine
114
115 end

```

Figure 7c Green_switch, update_screen, pause, loadN and decN functions in Task 3



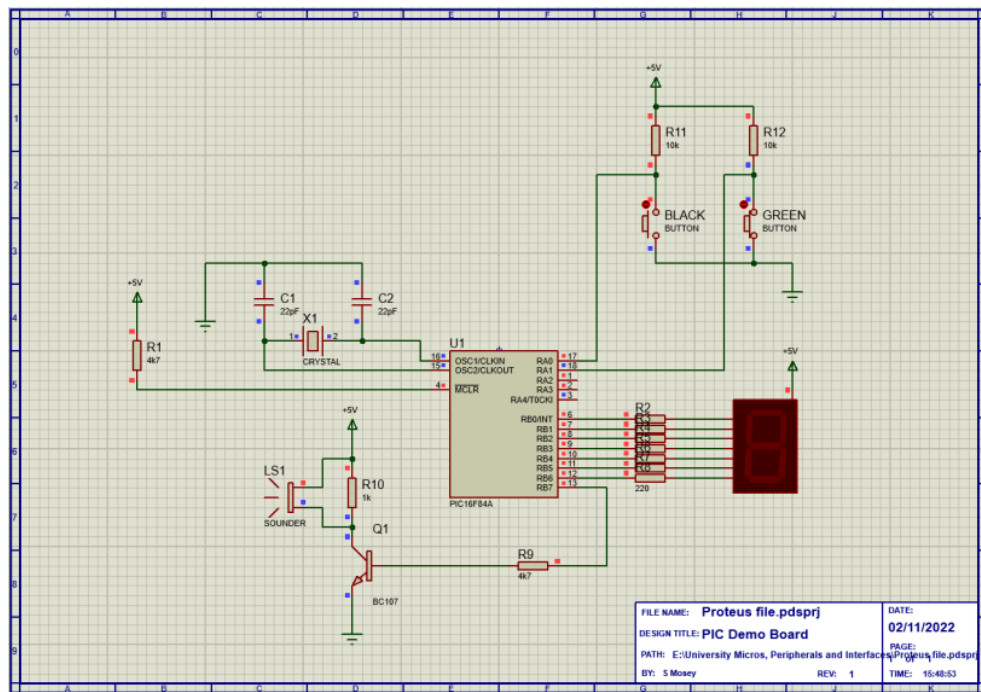


Figure 7d Proof of completing Task 3 – the 7 segment LED display will display '1' if the black button is selected and '2' if the green button is selected

Define port A as an input port and equate memory addresses in tris a.

Define port B as an output port and equate memory addresses in tris b.

Define RP0 and RP1 registers to Pins 17 (connected in series to the black button switch) and 18 (connected in series to the green button switch) – then equate to memory addresses.

Define nCount – inner loop counter and mCount – outer loop counter then equate to memory addresses

Define hexadecimal numbers to LEDs A & B, and SWs Black & Green and bitmask onto ports A and B

Define flag and temp variables

Initialise start loop. Select register RP0 bank 1 and load w with 0x00 tris b = bank 1. Load it into tris b to make port A all outputs then repeat so port B is all inputs in tris b. Clear banks in the status registers and assign space in status register to blank memory then load into tris b to make port B an output. Define RP0 as an output on port B clear port B register and count address

Initialise test_switch. Move w to port A then and literal working file + test black switch. Move w to into the temp file register and test bit 0 of temp (black switch). Repeat for green switch and return to test_switch.

Call black_switch and move literal working value of 0x01 into the working register then move flag into the working file. Move literal value of 0x00 to the working register then call update_screen to check button status. Repeat for the green_switch.

Initialise update_screen. Move working file to port B and call pause then loop back. Initialise pause so the counter increments with a delay. If finished exit subroutine.



Figure 7e Pseudo code for Task 3

Task 4

The aim of Task 4 is to modify the standard circuit in Figure 3 so elements of the previous three tasks are combined therefore if the black button is selected to HIGH, the LED display increments up from 0 – F in hexadecimal and if the green is selected to HIGH, the LED display decrements up from 0 – F in hexadecimal.

```

1 ; FILE: count.asm
2 ; AUTHOR: Freya Vafadari
3 ; DATE: November 2022
4 ; DESCRIPTION: Count to 255 at 1 second intervals and display on a seven
5 ; segment display in binary.
6 ; The pause subroutine takes apx. 0.5 sec.
7 ; Pause is called 5 times
8
9 list p=16F84 ; select processor type
10 radix hex ; define radix as hexadecimal
11 #include "p16f84.inc"
12
13 -----
14 ; cpu equates (memory map)
15
16 ;;;; Task 4 - increment from 0 - F in hex with 0.5 second delays if the black switch is selected and
17 ;;;; decrement from 0 - F in hex with 0.5 second delays if the green switch is selected
18
19 portA equ 0x05 ; port A address
20 trisa equ 0x05 ; trisa address in register bank 1
21
22 portB equ 0x06 ; port B address
23 trisB equ 0x06 ; trisB address in register bank 1
24
25 indf equ 0x00 ; indir file reg
26 fsr equ 0x04 ; indir address reg
27
28 status equ 0x03 ; status register
29 RP0 equ 5 ; bit 5 of status register
30 RP1 equ 6 ; bit 6 of status register
31
32 start_address equ 0x11 ; start addr of character data
33 LED_index equ 0x0F ; indexing variable
34 COUNT_A equ 0x0C ; intermediate variable for calculating addresses
35
36 event_flag equ 0x10 ; event_handler start flag
37
38 temp equ 0x1 ; temp variable
39
40 SW_Black equ 0x01 ; SW_Black bitmask of portA
41 SW_Green equ 0x02 ; SW_Green bitmask of portA

```

Figure 8a Ports, registers, indf, fsr, Count_A, LEDs 0 – F, SWs, start_address, temp, LED_index and event_flag definitions in Task 4

Micros, Peripherals and Interfaces Assignment November 2022

```
42 LED_0 equ 0x40 ; 7 segment LED display character definitions
43 LED_1 equ 0x79
44 LED_2 equ 0x24
45 LED_3 equ 0x30
46 LED_4 equ 0x19
47 LED_5 equ 0x12
48 LED_6 equ 0x02
49 LED_7 equ 0x78
50 LED_8 equ 0x00
51 LED_9 equ 0x10
52 LED_A equ 0x08
53 LED_B equ 0x03
54 LED_C equ 0x46
55 LED_D equ 0x21
56 LED_E equ 0x06
57 LED_F equ 0x0E
58
59 org 0x000
60
61 start
62 bcf status,RP0 ; clear bank selection bits in the status register RP0
63 bcf status,RP1 ; clear bank selection bits in the status register RP1
64
65 bsf status,RP0 ; select bank 1
66 movlw 0x00 ; Load W with 0x00
67 movwf trisb ; Load it into trisb to make portB all outputs
68 bcf status,RP0 ; reselect bank 0
69
70 movlw 0xFF ; clear all lines Low
71 movwf portB
72
73 movlw LED_0 ; Load the character "0" into the W register
74 movwf 0x11 ; move the character "0" from W to start address of character data RAM
75 movlw LED_1 ; Load the character "1" into the W register
76 movwf 0x12
77 movlw LED_2 ; Load the character "2" into the W register
78 movwf 0x13
79 movlw LED_3 ; Load the character "3" into the W register
80 movwf 0x14
81 movlw LED_4 ; Load the character "4" into the W register
82 movwf 0x15
83 movlw LED_5 ; Load the character "5" into the W register
84 movwf 0x16
85 movlw LED_6 ; Load the character "6" into the W register
86 movwf 0x17
87 movlw LED_7 ; Load the character "7" into the W register
88 movwf 0x18
89 movlw LED_8 ; Load the character "8" into the W register
90 movwf 0x19
91 movlw LED_9 ; Load the character "9" into the W register
92 movwf 0x1A
93 movlw LED_A ; Load the character "A" into the W register
94 movwf 0x1B
95 movlw LED_B ; Load the character "B" into the W register
```

Micros, Peripherals and Interfaces Assignment November 2022

```

91      movlw LED_9      ; Load the character "9" into the W register
92      movwf 0x1A      ; Load the character "A" into the W register
93      movlw LED_A      ; Load the character "A" into the W register
94      movwf 0x1B      ; Load the character "B" into the W register
95      movlw LED_B      ; Load the character "B" into the W register
96      movwf 0x1C      ; Load the character "C" into the W register
97      movlw LED_C      ; Load the character "C" into the W register
98      movwf 0x1D      ; Load the character "D" into the W register
99      movlw LED_D      ; Load the character "D" into the W register
100     movwf 0x1E      ; Load the character "E" into the W register
101     movlw LED_E      ; Load the character "E" into the W register
102     movwf 0x1F      ; Load the character "F" into the W register
103     movlw LED_F      ; Load the character "F" into the W register
104     movwf 0x20
105
106 check_count
107
108     movlw 0x11      ; Load 0x11 to W register - maximum count value
109     movwf LED_index ; set index variable to max count value
110
111 centre_loop
112     ; SW_Black
113     movf portA,w     ; read portA
114     movwf temp       ; move temp working to file
115     comf temp,w      ; complement f with temp
116     andlw SW_Black   ; test SW_Black
117     andwf event_flag,w ; test event_flag
118     movwf temp       ; stores w register value (SW_Black status) in temp
119     btfsc temp,0     ; test bit 0 of temp (black switch), if button not pressed skip
120     call event_sw_black
121
122     movf portA,w     ; read portA
123     movwf temp       ; move temp working to file
124     comf temp,w      ; complement f with temp
125     andlw SW_Black   ; test black switch
126     movwf temp       ; move working to temp
127     btfss temp,0     ; test bit 0 of temp
128     bsf event_flag,0 ; set flag to allow event_sw_black
129
130     ; SW_Green
131     movf portA,w     ; read portA
132     movwf temp       ; move temp working to file
133     comf temp,w      ; complement f with temp
134     andlw SW_Green   ; test SW_Green
135     andwf event_flag,w ; test event_flag
136     movwf temp       ; stores w register value (SW_Green status) in temp
137     btfsc temp,1     ; test bit 1 of temp (black switch), if button not pressed skip
138     call event_sw_green

```

Figure 8b Start, check_count and centre_loop functions in Task 4

```

138
139     movf portA,w     ; read portA
140     movwf temp       ; move working to file
141     comf temp,w      ; complement f with temp
142     andlw SW_Green   ; test black switch
143     movwf temp       ; move working to file
144     btfss temp,1     ; test bit 0 of temp
145     bsf event_flag,1 ; set flag to allow event_sw_green
146
147     goto centre_loop ; one way transfer - loop back to the beginning of the centre_loop
148
149 event_sw_black
150     decfsz LED_index,f ; decrement f in the LED_index register - skip if 0
151     goto continue     ; one way transfer - loop back to the beginning of continue
152     goto check_count  ; one way transfer - loop back to the beginning of check_count
153
154 continue
155     ; check portB and count up 0 - F in hex
156     bcf event_flag,0 ; clear flag to show event complete
157     ; j = 15 - LED_index
158     movf LED_index,w ; move file 'LED_index' into working reg
159     sublw 0x10        ; 16 (F) - LED_index
160     movwf COUNT_A     ; store value into COUNT_A
161     ; FSR = start_address + COUNT_A
162     addlw start_address ; add COUNT_A to start_address
163     movwf fsr         ; move start_address to fsr
164     ; PortB = INDF
165     movf indf,w       ; pull data from RAM and store in W reg
166     movwf portB       ; copy data to portB
167
168     return
169
170 event_sw_green
171     incf LED_index,f ; increment f in the LED_index register
172     ; check portB and count down F - 0 in hex
173     bcf event_flag,1 ; clear flag to show event complete
174     COUNT_A = 15 - LED_index
175     movf LED_index,w ; move file 'LED_index' into working register
176     sublw 0x10        ; 16 (F) - LED_index
177     movwf COUNT_A     ; move file 'LED_index' into working register
178     ; FSR = start_address + COUNT_A
179     addlw start_address ; add COUNT_A to start_address
180     movwf fsr         ; move start_address to fsr
181     ; PortB = INDF
182     movf indf,w       ; pull data from RAM and store in W reg
183     movwf portB       ; copy data to portB
184
185     return
186     ; else exit subroutine
187

```

Figure 8c Event_sw_black, continue and event_sw_green functions in Task 4

Micros, Peripherals and Interfaces Assignment November 2022

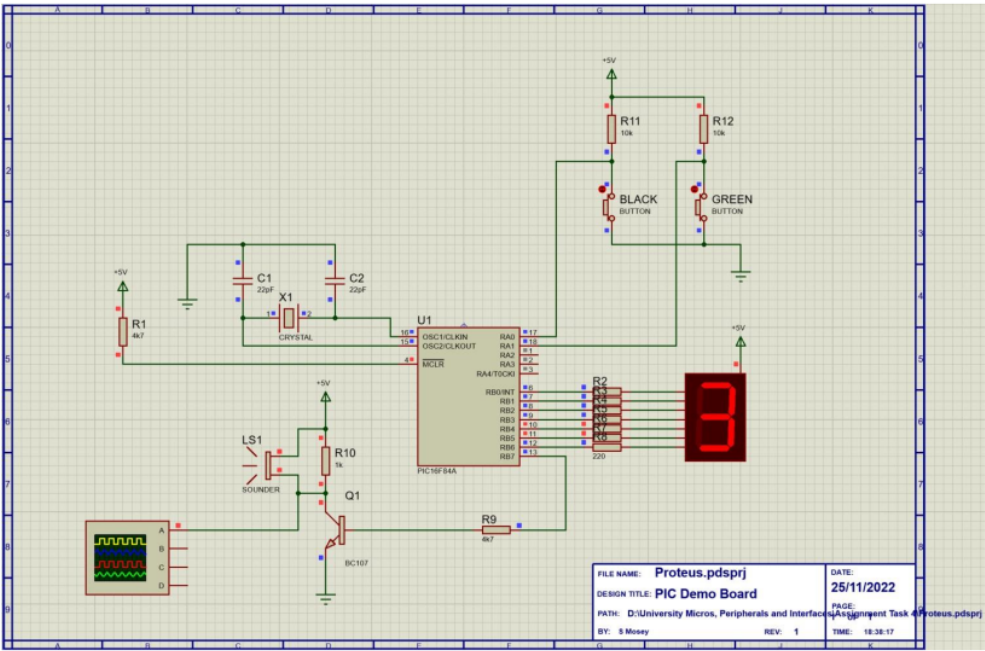
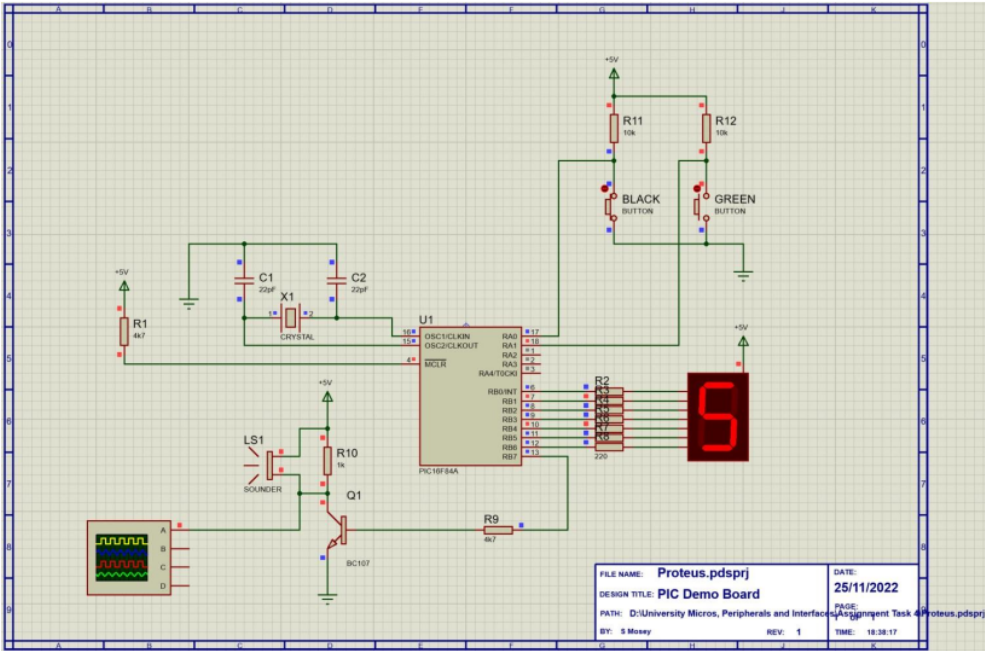


Figure 8d Proof of completing Task 4 – the 7 segment LED display will increment up from 0 - F if the black button is selected and will decrement down from 0 - F if the green button is selected

Micros, Peripherals and Interfaces Assignment November 2022

Define port A as an input port and equate memory addresses in tris a.

Define port B as an output port and equate memory addresses in tris b.

Define RP0 and RP1 registers to Pins 17 (connected in series to the black button switch) and 18 (connected in series to the green button switch) – then equate to memory addresses.

Define start_address variable – start address in memory of character data

Define LED_index variable – indexing variable in memory

Define COUNT_A – intermediate variable for calculating addresses

Define event_flag – event_handler start flag and temp – temp variable

Define SW_Black and SW_Green – bitmasks of port A

Define hexadecimal numbers for the 7 – segment LED display for all characters 0 – F

Initialise start. Clear bank selection bits in the status registers RP0 and RP1. Define 0x00 to be empty memory address in bank 1 and load into port B. Load the character '0' into the W register. Move the character '0' from W to start address of the character data RAM. Repeat for all characters 0 – F.

Initialise check_count. Load 0x11 into W register – maximum count value. Set index variable to max count value.

Initialise centre_loop. For SW_Black – read port A then move temp working register to file. Complement f with temp register and then test SW_Black. Test event_flag and move result in the w register into temp. Test bit 0 of temp (black switch) – if button not pressed then skip. Repeat for SW_Green. Return to beginning of centre_loop.

Initialise event_sw_black. Decrement f in the LED_index register but skip if 0. For event_sw_green decrement f in the LED_index register but skip if 0.

Initialise continue. Check port B and count 0 – F in hex. Move LED_index into working register to store value in COUNT_A. Add COUNT_A to start_address and move to fsr. Pull data from RAM and store in W register then copy to port B. Repeat for event_sw_green.

Figure 8e Pseudo code for Task 4

Task 5

The aim of Task 5 is to modify the standard circuit in Figure 3 so if the black switch is selected the buzzer will sound and the noise output will be displayed on the digital oscilloscope in the virtual terminal.

Micros, Peripherals and Interfaces Assignment November 2022

```

1 ; FILE:      count.asm
2 ; AUTHOR:    Freya Vafadari
3 ; DATE:      November 2022
4
5     list     p=16F84      ; select processor type
6     radix    hex          ; define radix as hexadecimal
7 ;
8     #include "p16f84.inc"
9
10    ;;; Task 5 - Buzzer sounds when either the black or green switch is selected
11
12    portA     equ    0x05    ; portA address
13    trisa     equ    0x05    ; trisa address in register bank 1
14
15    portB     equ    0x06    ; portB address
16    trisb     equ    0x06    ; trisb address in register bank 1
17
18    indf      equ    0x00    ; indir file reg
19    fsr       equ    0x04    ; indir addr reg
20
21    status    equ    0x03    ; status register
22    RP0       equ    5       ; bit 5 of status register
23    RP1       equ    6       ; bit 6 of status register
24
25    first_loop equ    0x0d    ; pause routine inner loop counter (first loop)
26    second_loop equ    0x0e    ; pause routine outer loop counter (second loop)
27    pause_count equ    0x0f
28
29    event_flag equ    0x11    ; event handler flag start
30
31    temp      equ    0x10    ; temporary variable
32
33    LED_A     equ    0x00    ; LED side of a bit of portB
34    LED_B     equ    0x01    ; LED side of a bit of portB
35
36    SW_B      equ    0x01    ; bitmask black switch of portA
37    SW_G      equ    0x02    ; bitmask green switch of portA
38
39    org       0x000
40
41    start
42    bcf       status,RP0      ; clear bank selection bits in the status register RP0
43    bcf       status,RP1      ; clear bank selection bits in the status register RP1
44
45    bsf       status,RP0      ; select bank = 1
46
47    movlw     0x03            ; Load w with 0x03 move the literal value 0x03 in the working register w
48    movwf     trisa           ; set switch pins SW_B & SW_G on portA as inputs
49
50    movlw     0x00            ; Load w with 0x00 move the literal value 0x00 in the working register w
51    movwf     trisb           ; Load it into trisb to make portB all outputs
52    bcf       status,RP0      ; reset bank = 0
53
54    clrf      portB           ; clear all lines low
55    bsf       event_flag,0    ; make sure no events registered
56
57    main_code
58    ; select black switch

```

Figure 9a Ports, registers, indf, fsr, LEDs, SWs, first_loop, second_loop, temp, pause_count and event_flag definitions in Task 5

Micros, Peripherals and Interfaces Assignment November 2022

```

38      org      0x000
39
40      start
41      bcf      status,RP0      ; clear bank selection bits in the status register RP0
42      bcf      status,RP1      ; clear bank selection bits in the status register RP1
43
44      bsf      status,RP0      ; select bank = 1
45
46      movlw   0x03             ; Load w with 0x03 move the Literal value 0x03 in the working register w
47      movwf   trisa            ; set switch pins SW_B & SW_G on portA as inputs
48
49      movlw   0x00             ; Load w with 0x00 move the Literal value 0x00 in the working register w
50      movwf   trisb            ; Load it into trisb to make portB all outputs
51      bcf      status,RP0      ; reset bank = 0
52
53      clrf    portB            ; clear all lines low
54      bsf      event_flag,0     ; make sure no events registered
55
56      main_code
57      ; select black switch
58
59      movf     portA,w          ; read portA
60      movwf    temp             ; move data in portA to the working file temp
61      comf     temp,w           ; complement data in register temp with w
62      andlw    SW_B             ; test black switch
63      andwf    event_flag,w     ; test event_flag
64      movwf    temp             ; stores w register value (SW_B status) in temp
65      btfsc    temp,0           ; test bit 0 of temp (black switch), if switch not set HIGH skip
66      call     event_black_sw
67
68      movf     portA,w          ; read portA
69      movwf    temp             ; move data in portA to the working file temp
70      comf     temp,w           ; complement data in register temp with w
71      andlw    SW_B             ; test black switch
72      movwf    temp             ; move data in SW_B to the working file temp
73      btfsc    temp,0           ; test bit 0 of temp
74      bsf      event_flag,0     ; set flag to allow event_black_sw
75
76      goto     main_code        ; return to start of the Loop
77
78      event_black_sw
79      ; check portB to see if LED is HIGH or LOW
80
81      bcf      event_flag,0     ; clear flag to show event complete
82      movf     portB,w          ; read portB
83      xorlw    0x01            ; complement portB bit 0
84      movwf    portB           ; write value back to portB
85      call     buzzer           ; short beep
86      return

```

Figure 9b Main_code and event_black_sw functions in Task 5

```

77      event_black_sw
78      ; check portB to see if LED is HIGH or LOW
79
80      bcf      event_flag,0     ; clear flag to show event complete
81      movf     portB,w          ; read portB
82      xorlw    0x01            ; complement portB bit 0
83      movwf    portB           ; write value back to portB
84      call     buzzer           ; short beep
85      return
86
87      buzzer
88      ; set pulse rate for buzzer : 0x7FF - 2047 Pulses
89
90      movlw   0x07             ; set w = 07 decimal
91      movwf   first_loop       ; first_loop = w: second Loop counter
92
93      loadN_buzzer
94      movlw   0xFF             ; set w = 255 decimal
95      movwf   second_loop      ; second_loop = w: first Loop counter
96
97      decN_buzzer
98      ; sound buzzer
99
100     bsf      portB,7           ; set buzzer output high
101     call     short_pause       ; wait 125us
102     bcf      portB,7           ; set buzzer output low
103     call     short_pause       ; wait 125us
104
105     decfsz   second_loop, f     ; second_loop--
106     goto     decN_buzzer        ; if second_loop != 0 then repeat second_loop--
107
108     decfsz   first_loop, f      ; else decrement second_loop
109     goto     loadN_buzzer       ; if first_loop != 0 then return
110
111     return
112
113     ; pause function
114     ; Generates ~4kHz inner clock (2000 pulses == 4kHz)
115     ; 2048 pulses 125us ON 125us OFF
116
117     short_pause
118     ; short_pause 125us delay
119
120     movlw   0x29             ; set w = 41 decimal
121     movwf   pause_count       ; pause_count = w: first Loop counter
122
123     decN
124     ; check buzzer status repeat if buzzer status is set to HIGH
125
126     decfsz   pause_count, f     ; pause_count--
127     goto     decN               ; if pause_count != 0 then repeat pause_count--
128
129     return                     ; else exit subroutine

```

Figure 9c Buzzer, loadN_buzzer, decN_buzzer, short_pause and decN functions in Task 5

The screenshot displays a digital oscilloscope interface. On the left is a large black rectangular area representing the oscilloscope screen. To the right of the screen are six control panels arranged in a 2x3 grid:

- Trigger:** Includes a 'Level' control with a scale from -10 to 10, 'AC' and 'DC' coupling buttons, a 'Position' control with a scale from 110 to 130, an 'Invert' button, and a 'Source' control with a scale from 1 to 10. It also features 'Auto', 'One-Shot', and 'Cursors' buttons.
- Channel A:** Includes a 'Position' control with a scale from 110 to 130, 'AC', 'DC', 'GND', and 'OFF' buttons, an 'Invert' button, and a 'Source' control with a scale from 1 to 10. It also features a 'V' control with a scale from 5 to 10 mV.
- Channel B:** Includes a 'Position' control with a scale from 110 to 130, 'AC', 'DC', 'GND', and 'OFF' buttons, an 'Invert' button, and a 'Source' control with a scale from 1 to 10. It also features a 'V' control with a scale from 5 to 10 mV.
- Channel C:** Includes a 'Position' control with a scale from 110 to 130, 'AC', 'DC', 'GND', and 'OFF' buttons, an 'Invert' button, and a 'Source' control with a scale from 1 to 10. It also features a 'V' control with a scale from 5 to 10 mV.
- Channel D:** Includes a 'Position' control with a scale from 110 to 130, 'AC', 'DC', 'GND', and 'OFF' buttons, an 'Invert' button, and a 'Source' control with a scale from 1 to 10. It also features a 'V' control with a scale from 5 to 10 mV.
- Horizontal:** Includes a 'Source' control with a scale from 1 to 10, a 'Position' control with a scale from 190 to 210, and a 'V' control with a scale from 5 to 10 mV.

Figure 9e Buzzer sound displayed on the digital oscilloscope in the virtual terminal for Task 5

References

References pic instruction set, microchip pic datasheet, hex number system flowchart

- [1 M. W. Storr, "Binary Numbers," Electronics-tutorials.ws, 08-Jul-2022. [Online]. Available:
] https://www.electronics-tutorials.ws/binary/bin_1.html#google_vignette. [Accessed: 27-Nov-2022].
- [2 M. W. Storr, "Hexadecimal Numbers," *Electronics-tutorials.ws*, 08-Jul-2022. [Online]. Available:
] https://www.electronics-tutorials.ws/binary/bin_3.html. [Accessed: 27-Nov-2022].
- [3 Microchip Technologies, "PIC16F84A Data and Instruction Set," Microchip.com. [Online].
] Available:
<https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/35007C.pdf>. [Accessed: 27-Nov-2022].
- [4 M. Lab, "PIC16F84A Microcontroller introduction and features," Microcontrollers Lab, 10-Dec-
] 2018. [Online]. Available: <https://microcontrollerslab.com/pic16f84a-microcontroller-introduction-and-features/>. [Accessed: 27-Nov-2022].

Micros, Peripherals and Interfaces Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

65/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



Comment 1

ceramic capacitors



Comment 2

These are only the properties of how Proteus uses the sound card of the computer to emulate the component. LS1 is actually a piezo buzzer. This relies on the piezoelectric effect which causes a diaphragm to deform when a voltage is applied (operation is very similar to a loudspeaker) the applied voltage has to be an AC signal to create vibrations in the air in response to the movements of the diaphragm.



Comment 3

As the Piezo buzzer is actually a capacitive device, R10 is actually there to discharge the sounder when the transistor is off



Comment 4

Its a switch, not an amplifier. The IO pin of the PIC is either 0V or 5V so the transistor is either off or on. When driven with a square wave from the PIC the transistor will switch on and off driving the piezo sounder, vibrating its diaphragm to create a sound.



Comment 5

Good



Comment 6

This would have been better if you used the INDR and FSR registers as it would have made the code reusable



Comment 7

What happens to the program after it exits the subroutine? Where does it go? Does it just stop?



Comment 8

This is a better method than task 1. well done

GRADING FORM: SMU GENERIC_SMOSEY

FREYA VAFADARI

64.60

ENGAGEMENT WITH LITERATURE

including reading, referencing, academic conventions and academic honesty



Fair description of the components on the PIC demo board

12.6

KNOWLEDGE & UNDERSTANDING

Knowledge of the basic underlying concepts and principles of a subject.



Excellent clear understanding of the operation of all the devices on the board - evidenced through descriptions and code operation

13.6

COGNITIVE & INTELLECTUAL

Evaluate underlying concepts and principles of a subject and interpret qualitative and quantitative data in order to develop lines of argument

and make sound judgements.



Mostly clear and concise code, although task 1 could have been better. Good solutions to the problems

13

PRACTICAL APPLICATION SKILLS

Different approaches to solving problems in particular contexts.



Simulations will work. Problems solved with good coding methods

12.4

TRANSFERABLE SKILLS

Communicate the results of their study/work accurately and reliably, and with structured and coherent arguments; the qualities needed for employment requiring the exercise of some personal responsibility within a structured and managed environment.



Generally well written, well structured report.

13