

Micros Assignment, Emily De Lange

by Emily De Lange

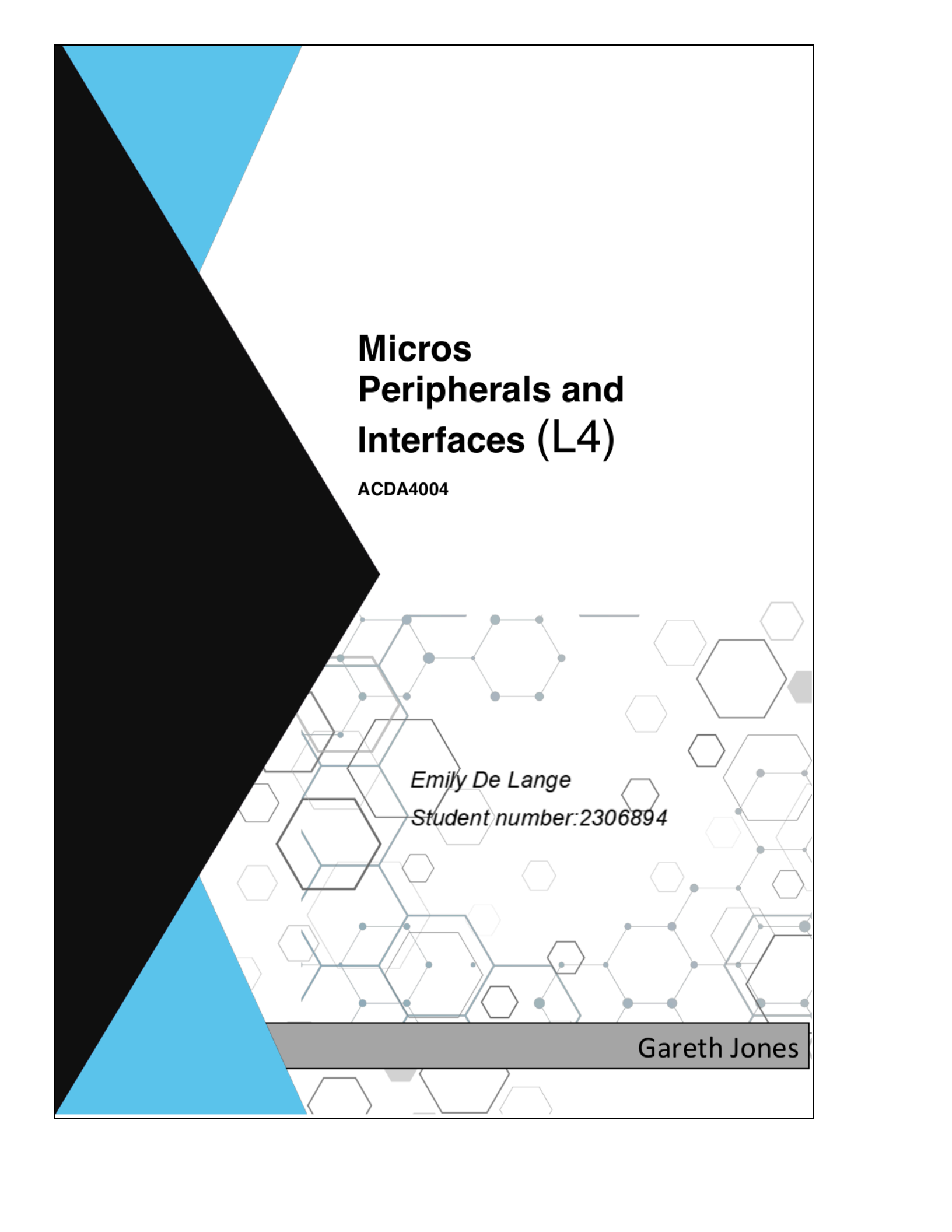
Submission date: 13-Dec-2024 09:21AM (UTC+0000)

Submission ID: 245941594

File name: 51449_Emily_De_Lange_Micros_Assignment__Emily_De_Lange_621295_187316716.docx (1.49M)

Word count: 3246

Character count: 15686



Micros Peripherals and Interfaces (L4)

ACDA4004

Emily De Lange

Student number: 2306894

Gareth Jones

Contents

Introduction	4
PIC16F84A circuit	4
Components used	5
PIC considerations.....	8
PIC data	8
Programming the Pic.....	8
Ports	9
Pins.....	9
ALU	9
W Register	10
How ALU and the W Register work together.....	10
Register file map	10
Watchdog timer	10
Fuses	10
Binary	11
Hexadecimal.....	11
Displaying hexadecimal characters on a seven-segment display	11
Coding questions.....	11
Code set up	11
(Definitions)	12
(Variables)	12
(Reset and interrupt vectors).....	12
1. Modify "Count.asm" 0 to F in hexadecimal numbers in 0.2 second increments.	13
2. Two buttons, it displays "L" or "r" the 7 segment LED display	15
.....	15
3. LED segments to light all on or all off	16
.....	17
4. Display5 count up to 9 and back down to 0.....	17
5. Make piezo sounder produce a tone while one of the buttons is pressed	20
Summary.....	21
References	22
Appendices.....	23

Figure Table

Figure 1 PIC16F84A circuit	4
Figure 2, Original circuit provided.....	5
Figure 3, LED display	5
Figure 4, Display configuration [1].....	6
Figure 5, Truth table.....	6
Figure 6, 5V regulator with capacitors.....	6
Figure 7, Transistor and buzzer.....	6
Figure 8, Buttons with pull up resistors	7
Figure 9, Crystal and capacitors	7
Figure 10, PIC, ...	7
Figure 11, Block diagram on PIC [4]	8
Figure 12, Pin location	9
Figure 13, Register file map	10
Figure 14, Binary for 1-F.....	11
Figure 15, Code set up	12
Figure 16, End of each code.....	13
Figure 17, flow chart 0-f display.....	13
Figure 18, Code for question 1 (1)	14
Figure 19, Code for question 1 (2)	15
Figure 20, Flow chart for "L" or "R" button press	15
Figure 21, code for question 2	16
Figure 22, Flow chart for all LED's on/off.....	16
Figure 23, Code for question 3.....	17
Figure 24, Flow chart, question 4.....	18
Figure 25, Question 4 (1)	18
Figure 26, Question 4 (2)	19
Figure 27, Question 4 (3)	19
Figure 28, Question 4 (4)	20
Figure 29, Flow chart for question 5	20
Figure 30, Question 5 code	21

Introduction

This report includes a simulated circuit from the “count” project, originally designed for the HND/BEng PIC16F84 board assignment using Proteus. The PIC16F84A microcontroller is part of the PIC16 family and is typically used in simpler embedded systems. In this project, the microcontroller drives the circuit and interfaces with a seven-segment display through its General-Purpose Input/Output (GPIO) pins. The report will provide detailed explanations of the components and their functions, supported by external research. Later, it will include examples of the assembly code used, illustrating how these components interact within the circuit. The code will be annotated, and flowcharts will be provided to outline the coding process.

PIC16F84A circuit

The circuit shown in Figure 1 PIC16F84A circuit, is built off the original circuit shown in Figure 2, Original circuit provided. It has an 8-bit microcontroller from Microchip which is the central controller reading button inputs and controlling outputs to the LED display and buzzer. There are seven 220Ω resistors in series to limit current flow through each segment of the LED display. Two buttons with 10K pull up resistors

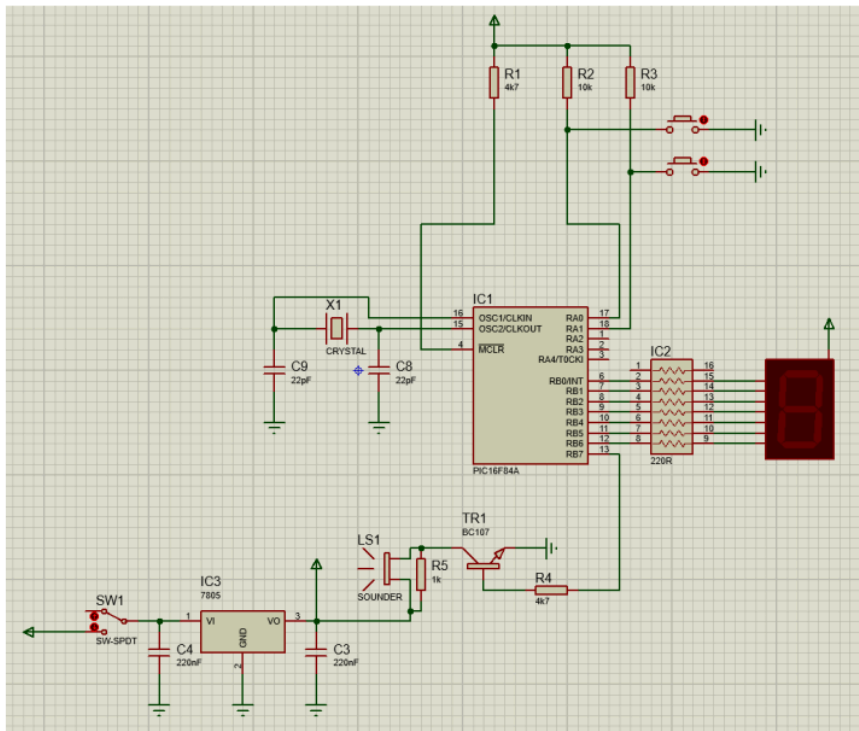


Figure 1 PIC16F84A circuit

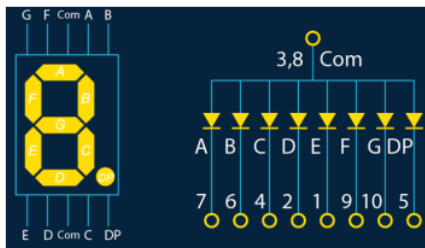


Figure 4, Display configuration [1]

Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	x	x	x	x	x	x	
1		x	x				
2	x	x		x	x		x
3	x	x	x	x			x
4		x	x			x	x
5	x		x	x		x	x
6	x		x	x	x	x	x
7	x	x	x				
8	x	x	x	x	x	x	x
9	x	x	x			x	x

Figure 5, Truth table

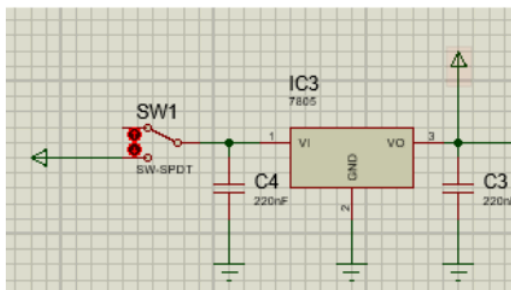


Figure 6, 5V regulator with capacitors

5V Fixed 1A positive power supply regulator (7805) - This component is a 5V regulator meaning it restricts the output voltage to 5V on various ranges of input voltages. This will add safety to the circuit and protect against voltage fluctuations, as seen in Figure 6, 5V regulator with capacitors. In this circuit capacitors are added to help with smoothing and stop the output voltage from oscillating, by charging and discharging. Pin 2 is the ground, Pin 1 and 3 are connected to 220nF capacitors.

One downfall from using the 7805 regulator is it can produce a lot of heat.

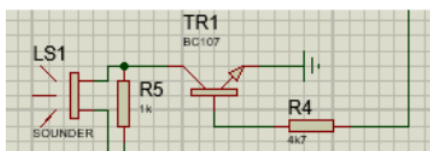


Figure 7, Transistor and buzzer

Silicon NPN low power bipolar transistor (BC107) - Seen in Figure 7, Transistor and buzzer. This is a semi conductive device, the function of this is to amplify or switch the signal. A primary use of these NPN (Negative-Positive-Negative) transistors are in amplification circuits. They are used to amplify a weak

signal into a larger output signal. In this circuit it is used along with the sounder see Figure 1 PIC16F84A circuit. This transistor is connected through port B, which is also connected to a sound buzzer. The transistor flexing sends a pulse to the buzzer. To do this the pin needs to go on and off over and over to make a sound.

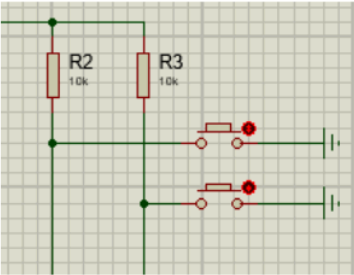


Figure 8, Buttons with pull up resistors

Button – The buttons used in this circuit are push buttons that don't latch when pressed. These buttons, have been added to the circuit with 10K pull up resistors, see Figure 8, Buttons with pull up resistors. Adding these resistors ensures that when the button isn't pressed it remains on logic 1. The buttons are grounded so when the button is pressed the current on the one side is able to flow to the other, however when the button is released the circuit is broken and the button is floating. As the voltage is intermittent it needs to be connected to ground to ensure 0V is read. [2] Having 10K resistors means less current which results in less impact when

shorted to ground, too much resistance causes noise on the line. When working out resistance values for LED's the equation used was Ohms law seen in Equation 1, Ohms law.

$$V = IR$$

Equation 1, Ohms law

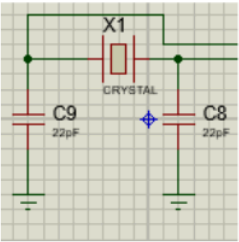


Figure 9, Crystal and capacitors

Crystal – A crystal is a two-pin device, with a wafer of quartz crystal or ceramic inside with electrodes connected to it. [3] The waves produced are square and tend to be very accurate. The crystal oscillator along with the 22pF capacitors will produce an external clock used to help keep track of time and can stabilise frequencies, see Figure 9, Crystal and capacitors. It works by a changing the shape of the quartz under an electronic field, known as piezoelectricity. When creating a circuit, you can pick from an external oscillator or an internal one built into the Pic. When using proteus it is important to note that the crystal may behave differently than in a real-world scenario. Crystals can also be used in walkie talkies, but their functionality changes slightly in that they are used the crystal stabilise the frequency appose to creating a sound.

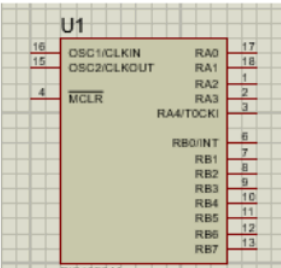


Figure 10, PIC,

PIC16F84A – The PIC seen in Figure 10, PIC, is made by Microchip and belongs to the mid-range family of the PICmicro microcontroller devices, [4] A block diagram of the chip internals is shown in Figure 11, Block diagram on PIC . The memory contains 1k words which means it can translate 1024 instructions. The RAM (Read Only Memory) contains 68 bytes and EEPROM (Electrically Erasable Programmable Read-Only Memory) is 64 bytes. This Pic has 13 0/1 pins which are configurable on a pin to pin basis. Proteus can supply 5V but when creating this circuit in a practical sense a power supply

would need to be added. As per the data sheet [4] the sink/source Max Per Pin is 25mA before the pins or device get damaged.

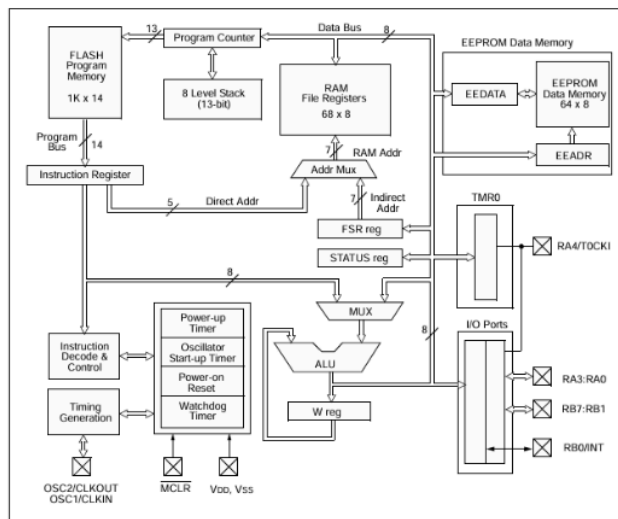


Figure 11, Block diagram on PIC [4]

PIC considerations

When choosing the right PIC microcontroller, keep in mind the performance you require, budget and understand the design of your circuit.

Consider the following:

- 'F' and 'C' type variants: most are 'f' types, but some are 'C' types which mean you can only flash them once.
- Oscillators and speed: If the PIC has an internal oscillator. 4MHz is a common speed. For reading buttons, multiplexing displays or turning LED's or motors on or off, is plenty. [5]
- Compiler: Many compilers are available, be sure to pick one that is compatible with the language being used.

PIC data

The PIC used in this circuit is the PIC16F84A. It is an 18-pin Enhanced FLASH/EEPROM, 8-bit Microcontroller.

- 18pin: refers to the number of pins the PIC has.
- EEPROM: (Electrically Erasable Programmable Read-Only Memory) allows for data to be written erased and rewritten electronically.
- 8-bit: Processes data in 8-bits (1 byte) this is what this PIC can handle in one operation
- 68 bytes of RAM: This is a relatively small amount, suitable for smaller projects.

Programming the Pic

Programming the PIC16F84A involves writing a code, the code language for this Pic is assembly and carried out on proteus. Programming this PIC can also be done in MPLAB along side a suitable

compiler like 'MPLAB XC8'. MPLAB is a programme developed by the same company who developed the PIC. [6]

Ports

Port A has five available ports. Port B has seven available ports. This PIC has two banks, A is for inputs or outputs B is to manipulate the data.

Bank 0 contains the lower 128 registers (addresses 0x00 to 0x7F).

Bank 1 contains the upper 128 registers (addresses 0x80 to 0xFF).

To switch between Bank 0 and Bank 1, the RP0 bit in the STATUS register is used:

RP0 = 0: The microcontroller is in Bank 0.

RP0 = 1: The microcontroller is in Bank 1.

Pins

Pins are used to attached power, ground, inputs and outputs and communications. See Figure 12, Pin location for pin locations.

- RA0, RA1, RA2, RA3 and RA4, these are assigned to port A and can be configured as inputs or outputs.
- RB0-RB7 are assigned to port B and have multiple options. These can be used as inputs or outputs or used as an interrupt. For example, putting the chip to sleep via an external interrupt.
- MCLR, Master/Memory Clear Reset. Is used when starting the device, resetting it means you know where you are starting from. The bar over the top means pulse 'LOW' to clear. No bar means pulse 'HIGH' to clear.
- OSC is the oscillator.
- VDD is +5V (note: not in proteus)
- VSS is -5V (note: not in proteus)

The Pins on this PIC have a default state of high/true, when buttons are not pressed. When the button is pressed it goes to logic 0 state. They are inverted.

PDIP, SOIC

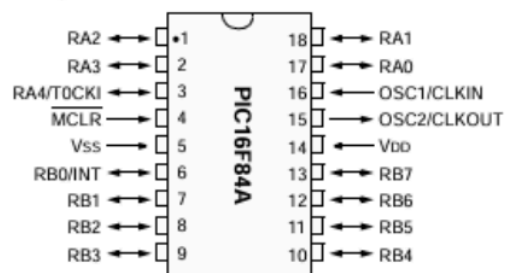


Figure 12, Pin location

ALU

The Arithmetic Logic Unit (ALU) is an internal digital circuit that operates in binary data. It performs two categories of operation logical operations (AND, OR, NOT etc) and arithmetic operations like addition and subtraction.

W Register

The W Register (Working register) is a single 8 bit register that is a temporary storage area for data being processed by the microcontroller. You can put any value that you wish, once assigned, you can add to it or move it but if you assign another value it will be overwritten.

In this Pic the destination designator specifies where the result of the operation is to be placed, if 'd' is '0' it is placed in the W register, if '1' it is placed in the specified file register. [4]

Using interrupts with the W register can cause issues as it may move the value inputted before the interrupt and fail to return to it.

How ALU and the W Register work together

Data is loaded into the W register using instructions like *MOVLW* (Move literal to W register). When the ALU receives one operation from the W register and one from another memory location, The ALU will take the W register contents and perform the operation to give the result back to the port called upon. The result will either be stored in the W register or the A file register.

Register file map

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A

File Address	Indirect addr. (R)	Indirect addr. (R)	File Address
00h			80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EEDCON1	88h
09h	EEDADR	EEDCON2 ⁽¹⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch
4Fh	6B General Purpose Registers (SRAM) Mapped (accessed) in Bank 0		CFh
50h			D0h
7Fh	Bank 0 Bank 1		FFh

□ Unimplemented data memory location, read as '0'.
 Note 1: Not a physical register.

The PIC16F84A has a reasonably simple file register map, seen in Figure 13, Register file map. The file map refers to the organisation of internal RAM, where the special function registers and general-purpose registers are located. There are two banks on this PIC 0 and 1. To call a bank use the *RPO* bit, (*RPO* = 0 is for bank 0, *RPO* = 1 is for bank 1). You can change between banks by clearing the *RPO* bit in the *STATUS* register. The status register is used for controlling the program flow and interacting with the microcontroller operations.

Figure 13, Register file map

Watchdog timer

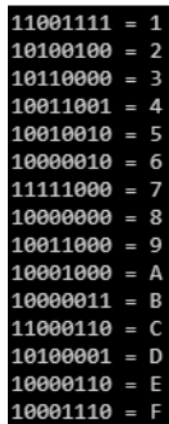
A watchdog timer is a failsafe for a programme that will continuously run. A watchdog timer is there to reset a programme that may have got stuck in a loop. Inside the PIC there is a resistor/capacitor network working as an internal clock. When it starts it will be at 00 and go up in one increment until it reaches FF before it returns to 00. Note that these times are irrespective of an external clock frequency. It is important to know if the watch dog timer is running, glitches could cause an error. When active you can add a clear watch dog command.

[Fuses](#)

The fuses (fuse bits) built into the PIC are not to be confused with fuses you'd use on a breadboard. These are however electronic switches that are 'blown' by the programmer. [7] They are used for control settings such as clock speed. If you want the watchdog timer, these fuses need to be 'blown' for the watchdog timer to work. To do this a couple of lines can be written at the beginning of the programme, which will enable certain fuses.

[Binary](#)

The PIC16F84A works with an 8-bit data meaning each register holds an 8-bit binary value. 00000000 to 11111111 in binary. As binary numbers are only 1's and 0's they need more digits to represent large values. The length of a binary number describes how many bits it is made of. Four bits are 1 nibble, eight bits are a byte. In Figure 14, Binary for 1-F you see an example later used in the code of the binary structure to enable the seven segment LED display to go from 1 to F.



11001111	= 1
10100100	= 2
10110000	= 3
10011001	= 4
10010010	= 5
10000010	= 6
11111000	= 7
10000000	= 8
10011000	= 9
10001000	= A
10000011	= B
11000110	= C
10100001	= D
10000110	= E
10001110	= F

Figure 14, Binary for 1-F

[Hexadecimal](#)

The PIC16F84A works with an 8-bit data, meaning each register holds an 8-bit binary value. 00000000 to 11111111 in binary or 0x00 to 0xFF in Hexadecimal. Hexadecimal are base-16, like decimal numbers are 0-9 for the tenth value. Numbers 11 to 15 are represented by letters. Each four bits can be converted to its hexadecimal equivalent.

[Displaying hexadecimal characters on a seven-segment display](#)

To display the hexadecimal characters of 0-F on the seven segments display each segment will light up individually depending on which value is being called to display. To do this each hexadecimal digit will need to be called to its corresponding 7-bit binary pattern. This binary pattern will need to be sent to the appropriate GPIO pins. See figure Figure 14, Binary for 1-F for the correct binary pattern for the 1-F.

[Coding questions](#)

Below are flow charts and annotated code for Section 3 of the assignment.

Code set up

Below is how most of the code is set up before you start getting into the loops, which is more specific code. See Figure 15, Code set up. The end of each code is also identical in that it used the same *count*, *loadN* and *decN*. See Figure 16, End of each code. Assembly code language is used to programme the PIC.

(Definitions)

First calling upon the *main.asm* source file and including the *p16f84a.inc* register definition file.

portB = Defining/naming value (something later will use the value)

trisb = (tri state) input or output pins (direction register)

status = Status register = 8 bits

RP0 = bit 5 of status register

RP1 = 6 of status register

(Variables)

count = set counter 0 to 255

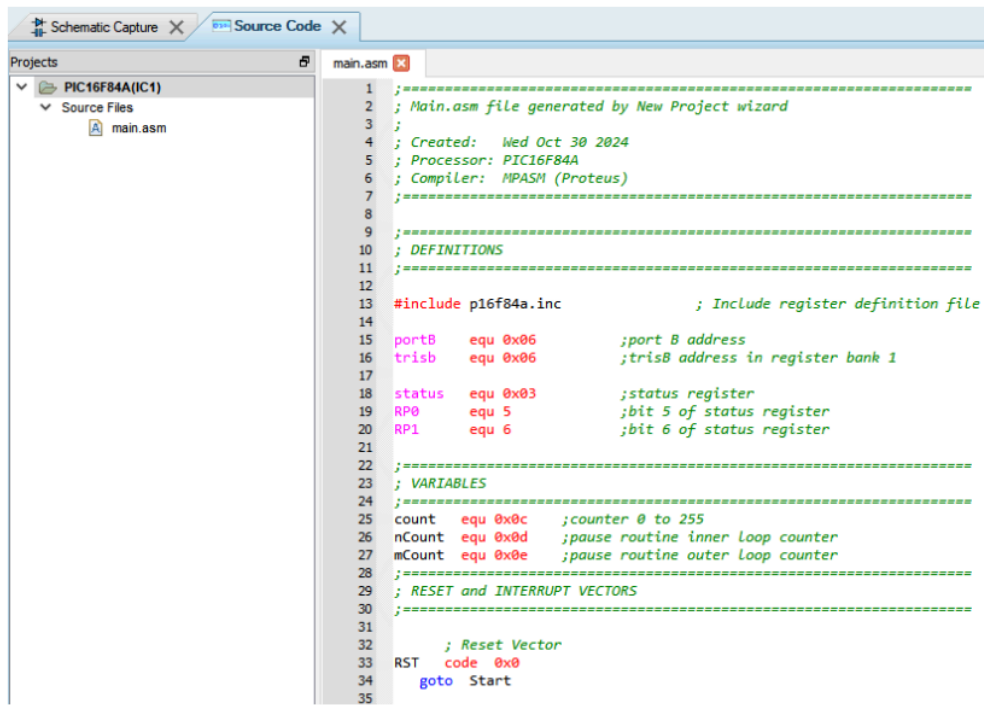
nCount = pause routine inner loop counter

mCount = pause routine outer loop counter

(Reset and interrupt vectors)

RST = Jump to named label "goto start" Before loop must be set up.

goto = Go back to start of code



```
1 ;=====
2 ; Main.asm file generated by New Project wizard
3 ;
4 ; Created: Wed Oct 30 2024
5 ; Processor: PIC16F84A
6 ; Compiler: MPASM (Proteus)
7 ;=====
8
9 ;=====
10 ; DEFINITIONS
11 ;=====
12
13 #include p16f84a.inc ; Include register definition file
14
15 portB equ 0x06 ;port B address
16 trisB equ 0x06 ;trisB address in register bank 1
17
18 status equ 0x03 ;status register
19 RP0 equ 5 ;bit 5 of status register
20 RP1 equ 6 ;bit 6 of status register
21
22 ;=====
23 ; VARIABLES
24 ;=====
25 count equ 0x0c ;counter 0 to 255
26 nCount equ 0x0d ;pause routine inner loop counter
27 mCount equ 0x0e ;pause routine outer loop counter
28 ;=====
29 ; RESET and INTERRUPT VECTORS
30 ;=====
31
32 ; Reset Vector
33 RST code 0x00
34 goto Start
35
```

Figure 15, Code set up

```

124 ;-----
125 ;subroutine pause: Lets waste some time...
126 ;
127 ;inputs: nothing
128 ;outputs: nothing
129 ;
130 ;does: nothing but waste some time
131 ;corrupts: register W
132 ;
133 ;description: two loops using "mCount" and "nCount" register files
134 ;             "mCount" is the outer loop counter (255 down to zero)
135 ;             "nCount" is the inner loop counter (255 down to zero)
136 ;
137 ;-----
138 pause
139     movlw 0xff          ; set w = 255 decimal
140     movwf mCount        ; mCount = w: outer loop counter
141 loadN
142     movlw 0xff          ; set w = 255 decimal
143     movwf nCount        ; nCount = w: inner loop counter
144 decN
145     decfsz nCount, f    ; nCount--
146     goto decN           ; if nCount != 0 then repeat nCount--
147
148     decfsz mCount, f    ; else decrement Count
149     goto loadN          ; if mCount != 0 then
150                         ; reload nCount to 255 and decrement
151     return              ; else exit subroutine
152 end
153 ;-----
154 ; Configuration! Important!
155 ;
156 ; at blast time, select:
157 ;     memory unprotected
158 ;     watchdog timer disabled
159 ;     standard crystal (4 MHz)
160 ;     power-up timer on
161

```

Figure 16, End of each code

1. Modify "Count.asm" 0 to F in hexadecimal numbers in 0.2 second increments.

Note that the start and end of the code are shown in Figure 15, Code set up and Figure 16, End of each code.

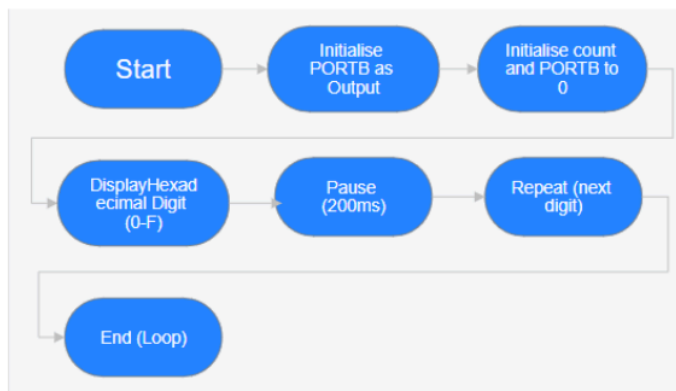


Figure 17, flow chart 0-f display

```

main.asm x
39
40 PGM code
41 Start
42     bcf     status,RP0      ; clear bank selection bits in the status reg
43     bcf     status,RP1
44
45     bsf     status,RP0      ; select bank 1
46     movlw   0x00            ; load w with 0x00
47     movwf   trisb           ; load it into trisb to make portb all outputs
48     bcf     status,RP0      ; reselect bank 0
49
50     clrf    portB           ; clear all lines low
51     clrf    count           ; initialize counter to 0
52
53     get_count                ; loop start (LED number/letter change)
54
55     movlw   b'01000000'      ;display 0
56     movwf   portB            ; move W to port B to update the display
57     call    pause            ; delay 200ms by subroutine
58
59     movlw   b'01111001'      ;display 1
60     movwf   portB            ; move W to port B to update the display
61     call    pause            ; delay 200ms by subroutine
62
63     movlw   b'00100100'      ;display 2
64     movwf   portB            ; move W to port B to update the display
65     call    pause            ; delay 200ms by subroutine
66
67     movlw   b'00110000'      ;display 3
68     movwf   portB            ; move W to port B to update the display
69     call    pause            ; delay 200ms by subroutine
70
71     movlw   b'000011001'      ;display 4
72     movwf   portB            ; move W to port B to update the display
73     call    pause            ; delay 200ms by subroutine
74
75     movlw   b'00010010'      ;display 5
76     movwf   portB            ; move W to port B to update the display
77     call    pause            ; delay 200ms by subroutine
78
79     movlw   b'00000011'      ;display 6
80     movwf   portB            ; move W to port B to update the display
81     call    pause            ; delay 200ms by subroutine
82
83     movlw   b'01111000'      ;display 7
84     movwf   portB            ; move W to port B to update the display
85     call    pause            ; delay 200ms by subroutine

```

Figure 18, Code for question 1 (1)

```

86      movlw b'00000000'      ;display 8
87      movwf portB            ; move W to port B to update the display
88      call pause             ; delay 200ms by subroutine
89
90
91      movlw b'00010000'      ;display 9
92      movwf portB            ; move W to port B to update the display
93      call pause             ; delay 200ms by subroutine
94
95      movlw b'00001000'      ; display a
96      movwf portB            ; move W to port B to update the display
97      call pause             ; delay 200ms by subroutine
98
99      movlw b'00000011'      ;display b
100     movwf portB            ; move W to port B to update the display
101     call pause             ; delay 200ms by subroutine
102
103     movlw b'01000110'      ;display c
104     movwf portB            ; move W to port B to update the display
105     call pause             ; delay 200ms by subroutine
106
107     movlw b'00100001'      ;display d
108     movwf portB            ; move W to port B to update the display
109     call pause             ; delay 200ms by subroutine
110
111     movlw b'00000110'      ; display e
112     movwf portB            ; move W to port B to update the display
113     call pause             ; delay 200ms by subroutine
114
115     movlw b'00001110'      ;display f
116     movwf portB            ; move W to port B to update the display
117     call pause             ; delay 200ms by subroutine
118
119
120     ;call pause             ; five pause executions equals ~ 1 second
121
122     goto get_count          ; repeat forever
123

```

Figure 19, Code for question 1 (2)

2. Two buttons, it displays "L" or "r" the 7 segment LED display

Note that the start and end of the code are shown in Figure 15, Code set up and Figure 16, End of each code.

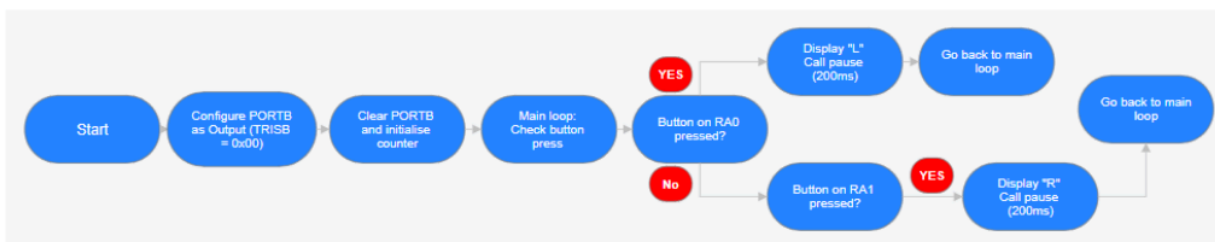


Figure 20, Flow chart for "L" or "R" button press

```

39 ;=====
40 ; CODE SEGMENT Loop
41 ;=====
42
43 PGH code
44 Start
45     bcf    status,RP0    ; clear bank selection bits in the status reg
46     bcf    status,RP1
47
48     bsf    status,RP0    ; select bank 1
49     movlw  0x00          ; Load w with 0x00
50     movwf  trisb          ; Load it into trisb to make portb all outputs
51     bcf    status,RP0    ; reselect bank 0
52
53     clrf   portb          ; clear all lines low
54     clrf   count         ; initialize counter to 0
55
56 main                ; function
57
58     BTFSS  portA, 0      ; port a (button) checking if on.
59     GOTO   Display_L     ; If button pressed show L
60
61     BTFSS  portA, 1      ; port a (button) checking if on
62     GOTO   Display_R     ; If button is pressed show R
63
64     movlw  b'11111111'   ; clear display
65     movwf  portb         ;
66     call   pause         ; delay 200ms by subroutine
67
68     goto   main          ; Loop back to beginning
69
70 Display_L
71
72     movlw  b'11000111'   ; display L
73     movwf  portb         ; move W to port B to update the display
74     call   pause         ; delay 200ms by subroutine
75     GOTO   main          ; When port a is pressed display L then go back to main
76
77 Display_R
78
79     movlw  b'11001110'   ; display r
80     movwf  portb         ; move W to port B to update the display
81     call   pause         ; delay 200ms by subroutine
82     GOTO   main          ; When port a is pressed display R then go back to main
83
84

```

Figure 21, code for question 2

3. LED segments to light all on or all off

Note that the start and end of the code are shown in Figure 15, Code set up and Figure 16, End of each code.

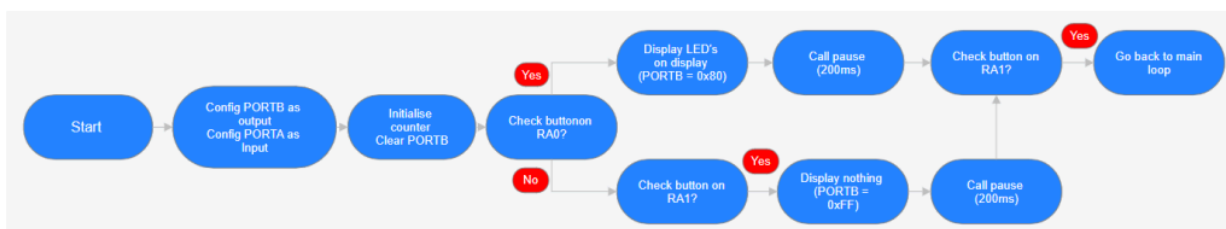


Figure 22, Flow chart for all LED's on/off

```

42
43 PGM code
44 Start
45     bcf     status,RP0      ; clear bank selection bits in the status reg
46     bcf     status,RP1
47
48     bsf     status,RP0      ; select bank 1
49     movlw   0x00            ; Load w with 0x00
50     movwf   trisb           ; Load it into trisb to make portb all outputs
51     bcf     status,RP0      ; reselect bank 0
52
53     clrf    portB           ; clear all lines low
54     clrf    count           ; initialize counter to 0
55
56 main                                ; function
57
58     BTFSS   portA, 0        ;port a (button) checking if on.
59     GOTO    Display_LED     ; If button pressed show LED
60
61     BTFSS   portA, 1        ; port a (button) checking if on
62     GOTO    Display_Nothing ; If button is pressed show Nothing
63
64
65     goto    main            ; Loop back to beginning
66
67 Display_LED
68
69     movlw   b'10000000'     ; display all LEDs
70     movwf   portB           ; move W to port B to update the display
71     call    pause           ; delay 200ms by subroutine
72     BTFSC   portA, 1        ; Check if on
73     GOTO    main            ; When port a is pressed display LEDs then go back to main
74
75
76 Display_Nothing
77
78     movlw   b'11111111'     ; display nothing
79     movwf   portB           ; move W to port B to update the display
80     call    pause           ; delay 200ms by subroutine
81     BTFSC   portB, 0        ;Check if off
82     GOTO    main            ; When port a is pressed display R then go back to main
83

```

Figure 23, Code for question 3

4. [Display5 count up to 9 and back down to 0](#)

The start-up for question four is different to the other questions. The outputs were set as variables, so only a word was needed as opposed to the binary pattern to clean up the code and make it easier to read. The variables were made from the hexadecimal conversion from the binary pattern needed to represent the number outputting on the LCD display.

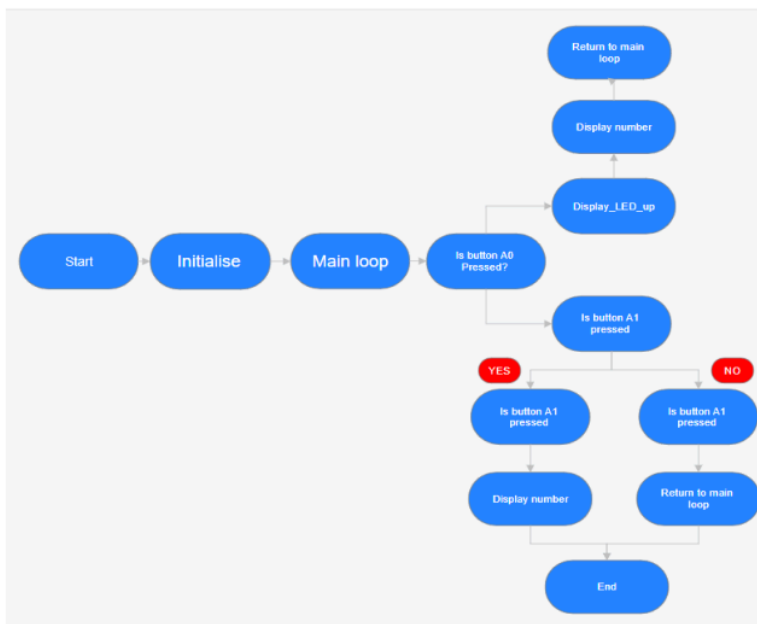


Figure 24, Flow chart, question 4

```

main.asm
1 ;=====
2 ; Main.asm file generated by New Project wizard
3 ;
4 ; Created: Wed Oct 30 2024
5 ; Processor: PIC16F84A
6 ; Compiler: MPASM (Proteus)
7 ;=====
8 ;
9 ;=====
10 ; DEFINITIONS
11 ;=====
12
13 #include p16f84a.inc ; Include register definition file
14
15 portA equ 0x05 ; port a address
16 trisa equ 0x85 ; trisa address
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 5 ;bit 5 of status register
22 RP1 equ 6 ;bit 6 of status register
23
24 ;=====
25 ; VARIABLES
26 ;=====
27 count equ 0x0c ;counter 0 to 255
28 nCount equ 0x0d ;pause routine inner Loop counter
29 mCount equ 0x0e ;pause routine outer Loop counter
30 nine equ 0x10 ;Variable representing 9 for LCD display
31 eight equ 0x00 ;variable representing 8 for LCD display
32 seven equ 0x78 ;variable representing 7 for LCD display
33 six equ 0x03 ;variable representing 6 for LCD display
34 five equ 0x12 ;variable representing 5 for LCD display
35 four equ 0x19 ;variable representing 4 for LCD display
36 three equ 0x30 ;variable representing 3 for LCD display
37 two equ 0x24 ;variable representing 2 for LCD display
38 one equ 0x79 ;variable representing 1 for LCD display
39 zero equ 0x40 ;variable representing 0 for LCD display
40 ;=====
41 ; RESET and INTERRUPT VECTORS
42 ;=====
  
```

Figure 25, Question 4 (1)

```

main.asm
40 ;=====
41 ; RESET and INTERRUPT VECTORS
42 ;=====
43
44 ; Reset Vector
45 RST code 0x0
46 goto Start ; essential code
47
48 ;=====
49 ; CODE SEGMENT Loop
50 ;=====
51
52 PGM code
53 Start ;Run through this function once to set up screen
54
55 bcf status,RP0 ; clear bank selection bits in the status reg
56 bcf status,RP1
57
58 bsf status,RP0 ; select bank 1
59 movlw 0x00 ; load W with 0x00
60 movwf trisb ; load it into trisb to make portb all outputs
61 bcf status,RP0 ; reselect bank 0
62
63 clrf portB ; clear all lines low
64 clrf count ; initialize counter to 0
65
66 movlw b'00010010' ;Set register to a value of 5
67 movwf portB ; move W to port B to update the display
68
69 button ;main function to check button status
70 BTFSS portA, 0 ;port a (button) checking if on.
71 call Display_LED_up ; Call the display function to count up from 5
72
73 BTFSS portA, 1 ;port a (button) checking if on.
74 call Display_LED_down ; Call the display function to count down from 9
75
76 goto button ;Loop through the main function to check for button
77
78 Display_LED_up ; Loop start (LED number/Letter change)
79
80 movlw five ;display 5
81 movwf portB ; move W to port B to update the display
82 call pause ; delay 200ms by subroutine
83
84

```

Figure 26, Question 4 (2)

```

main.asm
85 movlw six ;display 6
86 movwf portB ; move W to port B to update the display
87 call pause ; delay 200ms by subroutine
88
89 movlw seven ;display 7
90 movwf portB ; move W to port B to update the display
91 call pause ; delay 200ms by subroutine
92
93 movlw eight ;display 8
94 movwf portB ; move W to port B to update the display
95 call pause ; delay 200ms by subroutine
96
97 movlw nine ;display 9
98 movwf portB ; move W to port B to update the display
99 call pause ; delay 200ms by subroutine
100
101 return
102
103
104
105 Display_LED_down
106
107 movlw nine ;display 9
108 movwf portB ; move W to port B to update the display
109 call pause ; delay 200ms by subroutine
110
111 movlw eight ;display 8
112 movwf portB ; move W to port B to update the display
113 call pause ; delay 200ms by subroutine
114
115 movlw seven ;display 7
116 movwf portB ; move W to port B to update the display
117 call pause ; delay 200ms by subroutine
118
119 movlw six ;display 6
120 movwf portB ; move W to port B to update the display
121 call pause ; delay 200ms by subroutine
122
123 movlw five ;display 5
124 movwf portB ; move W to port B to update the display
125 call pause ; delay 200ms by subroutine
126
127 movlw four ;display 4
128 movwf portB ; move W to port B to update the display
129

```

Figure 27, Question 4 (3)

```

129      movwf    portB      ; move W to port B to update the display
130      call     pause      ; delay 200ms by subroutine
131
132      movlw    three      ;display 3
133      movwf    portB      ; move W to port B to update the display
134      call     pause      ; delay 200ms by subroutine
135
136      movlw    two        ;display 2
137      movwf    portB      ; move W to port B to update the display
138      call     pause      ; delay 200ms by subroutine
139
140      movlw    one        ;display 1
141      movwf    portB      ; move W to port B to update the display
142      call     pause      ; delay 200ms by subroutine
143
144      movlw    zero       ;display 0
145      movwf    portB      ; move W to port B to update the display
146      call     pause      ; delay 200ms by subroutine
147
148      ;call     pause      ; five pause executions equals ~ 1 second
149
150      return

```

Figure 28, Question 4 (4)

5. Make piezo sounder produce a tone while one of the buttons is pressed

Note that the start and end of the code are shown in Figure 15, Code set up and Figure 16, End of each code.

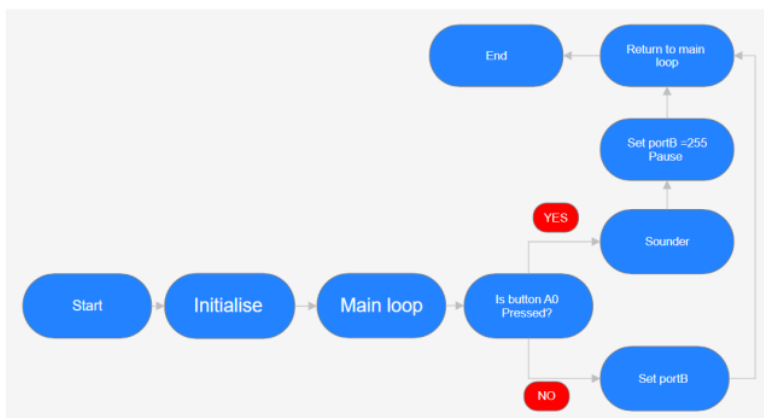


Figure 29, Flow chart for question 5

```

42
43 PGM    code
44 Start
45     bcf    status,RP0    ; clear bank selection bits in the status reg
46     bcf    status,RP1
47
48     bsf    status,RP0    ; select bank 1
49     movlw  0x00          ; load w with 0x00
50     movwf  trisb         ; load it into trisb to make portb all outputs
51     bcf    status,RP0    ; reselect bank 0
52
53     clrf   portB         ; clear all lines low
54     clrf   count        ; initialize counter to 0
55
56 main
57                                     ; function
58     BTFSS  portA, 0      ; port a (button) checking if on.
59     GOTO   Sounder      ; If button pressed sound
60
61     movlw  b'00000000'   ; set port b to not active
62     movwf  portB
63
64     goto   main          ; loop back to beginning
65
66 Sounder
67
68     movlw  b'11111111'   ; sound when port B pressed
69     movwf  portB         ; move W to port B to active
70     call   pause         ; delay 200ms by subroutine
71     GOTO   main          ; When port a is pressed sound, then go back to main
72

```

Figure 30, Question 5 code

Summary

This report has shown the “count” project, which utilizes a seven-segment LED display and a simulated circuit developed in Proteus using the PIC16F84A microcontroller. It features screenshots of the circuit and provides an explanation of each component and their interactions. To understand these components further, additional research was conducted, included diagrams and information from external sources. The report also details the method for displaying hexadecimal characters (0-F) on the seven-segment display, highlighting the corresponding binary representations that activate specific segments. Finally, the report includes flowcharts and annotated assembly language code, with annotations clarifying the purpose of each instruction and its role in the overall functionality of the program. This section shows the significance of clear logic and structured programming.

References

[Proteus, "Proteus," lab centre , 2024. [Online]. Available: <https://www.labcenter.com/blog/sim-17seg-displays/>. [Accessed 04 November 2024].

]

[creative technology lab, "How to connect a push button or switch," Creative technology lab , 2 [Online]. Available: <https://lab.arts.ac.uk/books/physical-computing/page/how-to-connect-a-push-button-or-switch#:~:text=When%20the%20button%20is%20pressed,ensure%20the%20Arduino%20reads%200V..> [Accessed 05 November 2024].

["Crystal oscillator," Wikipedia, 26 October 2024. [Online]. Available: https://en.wikipedia.org/wiki/Crystal_oscillator#:~:text=Crystal%20is%20the%20common%20term,circuits%2C%20such%20as%20crystal%20filters.. [Accessed 04 November 2024].

[Microchip, *PIC16F84A Data sheet*, Microchip technology inc, 2001.

4

]

[Dan, "PIC Microchip Guide, Tools and Tips," Electro-dan : Electronics and Computing, July 2022. 5 [Online]. Available: https://electro-dan.co.uk/electronics/PIC_guide.aspx. [Accessed 05 November 2024].

[Microchip, *MPLAB*, Arizona: Microchip technology inc.

6

]

[mstracy, *PIC programming in assembly*.

7

]

Appendices

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

ASCII Character Value Hexadecimal, binary numbers, angle, text png

Mnemonic, Operands	Description	Cycles	14-Bit Instruction Word				Status Affected	Notes
			MSb		LSb			
BYTE-ORIENTED FILE REGISTER OPERATIONS								
ADDWF f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRWF -	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECf f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF f	Move W to f	1	00	0000	1fff	ffff		
NOP -	No Operation	1	00	0000	0xxx	0000		
RLF f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS								
ADDLW k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDt -	Clear Watchdog Timer	1	00	0000	0110	0100	TO,PD	
GOTO k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE -	Return from interrupt	2	00	0000	0000	1001		
RETLW k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN -	Return from Subroutine	2	00	0000	0000	1000		
SLEEP -	Go into standby mode	1	00	0000	0110	0011	TO,PD	
SUBLW k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Micros Assignment, Emily De Lange

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

69/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

GRADING FORM: GJ_2019

EMILY DE LANGE

345

ENGAGEMENT WITH LITERATURE



Put citation numbers typically at end of section,
not start of sentence.

Some good sources cited, but could have used
even more to explain the components etc.

65

KNOWLEDGE AND UNDERSTANDING



A very good level of component and general circuit
explanation included.

75

COGNITIVE AND INTELLECTUAL

 Good attempts at the coding, however mostly it uses a brute force approach, and things like the buzzer routing don't look as if they are going to work (needs oscillating, rather than just turning the bit on)

60

PRACTICAL APPLICATION

 Well implemented circuit, with most key software tasks at least attempted.

70

TRANSFERABLE SKILLS

 Generally well written and presented.

75