

assignment

by Owen Miller

Submission date: 14-Jan-2025 01:44PM (UTC+0000)

Submission ID: 248415986

File name: 52086_Owen_Miller_assignment_621295_416552205.docx (133.45K)

Word count: 984

Character count: 4829

Micros Peripherals and Interfaces

ACDA4004

Student number: 2308626

Contents

Section 1.....3

Section 2.....4

Section 3.....5

 Task 1 5

 Task 2 5

 Task 3 6

 Task 4 6

 Task 5 6

 Task 6 7

Section 1

```

; Main.asm file generated by New Project wizard
;
; Created:   Mon Feb 22 2018
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)
;
;=====
;
; DEFINITIONS
;=====
#include p16f84a.inc          ; Include register definition file

portB equ 0x06               ;port B address
trisB equ 0x06               ;trisB address in register bank 1

status equ 0x03              ;status register
int0 equ 5                    ;bit 5 of status register
int1 equ 6                    ;bit 6 of status register

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

;=====
; RESET and INTERRUPT VECTORS
;=====
; Reset Vector
RST code 0x0
goto Start

;=====
; CODE SEGMENT
;=====

```

Figure 1

Before we begin with the actual coding process, it's essential to establish the initial setup. This involves configuring the registers, which are critical for controlling the flow of data and instructions within the system. Additionally, we need to define the inputs and outputs to ensure that the program can correctly interact with external devices or data sources. By carefully preparing these elements, we create a solid foundation for our coding endeavours.

```

PGM code
Start

    bcf      status,RP0      ; clear bank selection bits in the status reg
    bcf      status,RP1

    bsf      status,RP0      ; select bank 1
    movlw    0x40             ; Load w with 0x00
    movwf    trisB           ; Load it into trisB to make portB all outputs
    bcf      status,RP0      ; reselect bank 0

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

```

Figure 2

Figure two shows the setting up of the program

To ensure proper functionality, I am in the process of resetting the counter and clearing both the registers and the ports. I noticed that the value was set to 0x40, which could potentially lead to complications in the code execution. Therefore, I have changed this value to 0x00 to resolve any issues and maintain the stability of the system.

Figure 3

```

pause      movlw 0x0f      ; set w = 255 decimal.
           movwf mCount    ; mCount = w: outer loop counter

loadli     movlw 0x0f      ; set w = 255 decimal.
           movwf mCount    ; mCount = w: inner loop counter

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

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

```

Figure 4

Section 2

Focusing on values A through F, I conducted a detailed analysis to determine which specific segments of the seven-segment display needed to be illuminated to accurately depict each character. Each segment can be visualized as a bit in a binary number, where it can either be activated (represented as 1) or deactivated (represented as 0).

I meticulously mapped out the segments associated with the letters A to F, translating the necessary segments into binary numbers. This involved identifying which segments needed to glow to form the distinct shapes of these hexadecimal characters. After establishing the correct binary configurations for each value, I then converted these binary representations into their corresponding hexadecimal codes.

This systematic approach ensures that the appropriate segments light up for each character on the display, resulting in a clear and precise representation of the hexadecimal values ranging from 0 to F. By illuminating the correct segments, the display vividly presents each character, facilitating easy reading and interpretation.

This has given me the following values:

Value	binary	hex
A	0001000	0x08
B	0000011	0x03
C	1000110	0x46
D	1011110	0x21
E	0000110	0x06
F	0001110	0x0e

Section 3

Task 1

Start

Task 2

This code is designed to provide a simple interaction where pressing one button will display the letter "l" on the screen. Conversely, when you press the other button, the letter "r" will appear instead. This functionality allows for a clear and intuitive way to distinguish between the two actions through visual feedback.

Clear all ports

Set port b to an output

Start

if button 1 is high, set output b to 0x47 so the seven-segment display prints out an L

if button 2 is high, set output b to 0x4e so the seven-segment display prints out an r

else set output b to 0x00

return to start

Task 3

This pseudo code demonstrates the functionality of two buttons—button 1 and button 2—in relation to the LED lights. When button 1 is pressed, the code activates all the LEDs, causing them to light up simultaneously. Conversely, when button 2 is pressed, the code deactivates all the LEDs, resulting in them turning off completely. This simple interface allows for clear and immediate control over the LED lights.

Clear all ports

Set port b to an output

Start

When button 1 is high, set output b to 0x7f to set all the segments to high

If button 2 is high, set output b to 0x00 to clear the seven-segment display

Return to start

Task 4

Clear all ports

Set port b to an output

Create a segment including the values 0-9 in hex

Set digit to 5

Start

Display digit on output b

Subroutine button 1

When button 1 is high, increase the digit value by 1.

Subroutine button 2

When button 2 is high, decrease the digit value by 1 =.

if digit is 10

Set digit to 9

If digit is -1

Set digit to 0

Go to start

Task 5

Clear all ports

Set port b to output

If button 1 is pressed

Send output through pin b 8 creating a sound through the buzzer

Task 6

References

Pic16f84a data sheet used throughout the assignment

[PIC16F84A Data Sheet](#)

assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

13/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

GRADING FORM: GJ_2019

OWEN MILLER

66

ENGAGEMENT WITH LITERATURE



No citations used, and no proper reference list.

KNOWLEDGE AND UNDERSTANDING

 No circuit included and no component descriptions/explanation.

10**COGNITIVE AND INTELLECTUAL**

 Only a simplistic brute force approach applied to the single piece of code included.

10**PRACTICAL APPLICATION**

 No evidence provided of the circuit. Minimal programming attempted.

10**TRANSFERABLE SKILLS**

 Very poor presentation of the report.
Listings too small to see properly.
Extremely brief as is.

35