

Jake Rudd MPI

by Jake Rudd

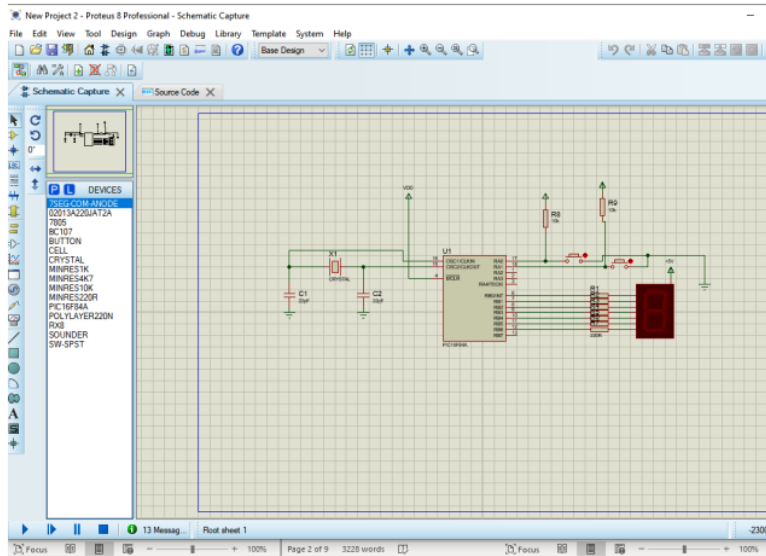
FILE	24425_JAKE_RUDD_JAKE_RUDD_MPI_315192_1337154339.DOCX (729.88K)		
TIME SUBMITTED	18-JAN-2021 12:40PM (UTC+0000)	WORD COUNT	2389
SUBMISSION ID	142530481	CHARACTER COUNT	10720

Level 4 All Electronics Streams

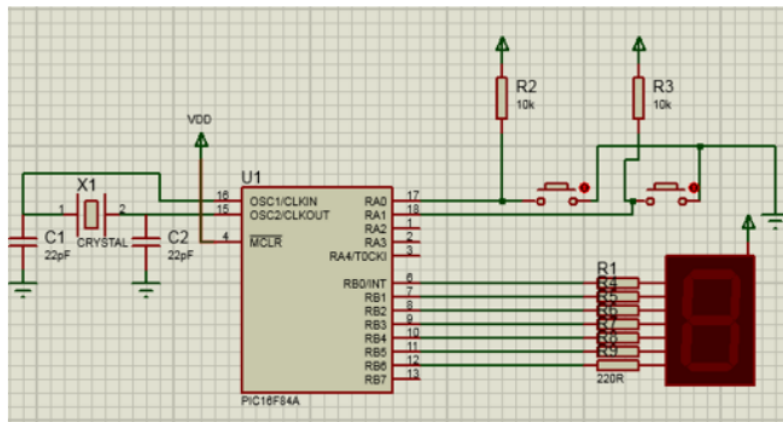
HNC/HND/B.ENG

Micros Peripherals and Interfaces. ACDA4004

Section 1:



The above is my version of the below original circuit.



Section 2:

Proteus generates code based on the PIC used.

Upon running the simulation the 7 segment display flashes through different segments, not showing any number in particular. I modified the code to display number 0-9 and also A – F (Hex)

The below shows the original code before modification.

```
=====
; Main.asm file generated by New Project wizard
;
; Created: Wed Nov 4 2020
; 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

RP0       equ 5              ;bit 5 of status register

RP1       equ 6              ;bit 6 of status register

;=====

; VARIABLES

;=====

count     equ    0x0c        ; counter 0 to 255

nCount    equ    0x0d        ; pause routine inner loop counter

mCount    equ    0x0e        ; pause routine outer loop counter

;=====

; RESET and INTERRUPT VECTORS

;=====

; Reset Vector

RST code 0x0

    goto Start

;=====

; CODE SEGMENT

;=====

PGM code

Start

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

        bcf      Status,RP1

        bsf      Status,RP0 ; select bank 1

    movlw    0x00          ; load w with 0x00

    movwf    trisb         ; load it into trisb to make portb all outputs

    bcf      Status,RP0 ; reselect bank 0

    clrf     portB         ; clear all lines low

    clrf     count        ; initialize counter to 0

get_count

```

```

        movf    count, w    ; move count to W
movwf    portB              ; move W to port B to update the display
addlw 1                    ; add +1 to W
movwf    count              ; save W in count

        call    pause       ; delay 200ms by subroutine
        call    pause
        call    pause
        call    pause
        call    pause       ; five pause executions equals ~ 1 second

        goto    get_count    ; repeat forever

```

;subroutine pause: lets waste some time....

;description: two loops using "mCount" and "nCount" register files

; "mCount" is the outer loop counter (255 down to zero)

; "nCount" is the inner loop counter (255 down to zero)

;

;-----

pause

```

        movlw   0x5          ; set w = 255 decimal

```

```

        movwf   mCount       ; mCount = w: outer loop counter

```

loadN

```

        movlw   0xff         ; set w = 255 decimal

```

```

        movwf   nCount       ; nCount = w: inner loop counter

```

decN

```

        decfsz  nCount, f    ; nCount--

```

```

        goto    decN         ; if nCount != 0 then repeat nCount--

```

```

        decfsz  mCount, f    ; else decrement Count

```

```

        goto    loadN        ; if mCount != 0 then

```

```

                                ; reload nCount to 255 and decrement

```

```

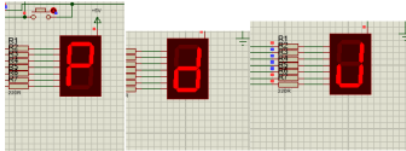
        return              ; else exit subroutine

```

;=====

END

Here are a few screenshots showing what is being displayed based on the code above.



To begin to modify the code I had to ascertain which codes in Hex would light different segments of the display. To do this, based on the 7 pins of the display, I had to work out and convert the binary number to Hex using a calculator. For Example, 1100 0000, converts to C0 in Hex, thus I could use the Hex code and input this into my code to display the number Zero.

The reason this works is because each '0' is actually on/off for a segment on the display. I input different numbers into a calculator in binary format and converted them to Hex, I then modified the code with the Hex digit and run the simulation, to obtain different results. After doing this enough times and through trial and error, I was able to know which segment related to which digit in binary.

For the purpose of this assignment I have condensed my findings into a table. Below the table shows my original workings.

Binary Number	Segments Lit	Hex	Digit Displayed
1100 0000	A,B,C,D,E,F	C0	0
1111 1001	B,C	F9	1
1010 0100	A,B,D,E,G	A4	2
1011 0000	A,B,C,D,G	B0	3
1001 1001	B,C,F,G	99	4
0001 0010	A,F,G,C,D	12	5
1000 0010	A,C,D,E,F,G	82	6
0111 1000	A,B,C	78	7
1000 0000	A,B,C,D,E,F,G	80	8
1001 0000	A,B,C,D,F,G	90	9
1000 1000	A,B,C,E,F,G	88	A
1000 0000	A,B,C,E,D,F,G	80	B
0100 0110	A,D,E,F	46	C
1100 0000	A,B,C,D,E,F	C0	D
0000 0110	A,B,D,E,F,G	84	E
1110	A,E,F,G	E	F

- 1
- 1

- 0
- 0 = A,B,C,D,E,F = C0 = 0
- 0
- 0-C
- 0-B
- 0-A

- 1
- 1
- 1 =B,C = F9 = 1
- 1
- 1
- 0-C
- 0-B
- 1

- 1
- 0-G
- 1 =A,B,D,E,G = A4 = 2
- 0-E
- 0-D
- 1
- 0-B
- 0-A

- 1
- 0
- 1 =A,B,C,D,G = B0 = 3
- 1
- 0
- 0-C
- 0-B
- 0-A

- 1
- 0
- 0 = B,C,F,G = 99 = 4
- 1
- 1
- 0
- 0
- 1

- 0
- 0
- 0 = A,F,G,C,D = 12 = 5

- 1
- 0
- 0
- 1
- 0

- 1
- 0
- 0 =A,C,D,E,F,G = 82 = 6
- 0
- 0
- 0-C
- 1
- 0-A

- 0
- 1
- 1 =A,B,C = 78 = 7
- 1
- 1
- 0-C
- 0-B
- 0-A

- 1
- 0
- 0 =A,B,C,D,E,F,G = 80 = 8
- 0
- 0
- 0-C
- 0-B
- 0-A

- 1
- 0
- 0 =A,B,C,D,F,G = 90 = 9
- 1
- 0
- 0-C
- 0-B
- 0-A

- 1
- 0
- 0 = A,B,C,E,F,G = 88 = A
- 0
- 1
- 0 - C
- 0 - B
- 0 - A

- 1
- 0
- 0 = A,B,C,D,E,F,G = 80 = B
- 0
- 0
- 0
- 0
- 0
- 0

- 0
- 1
- 0 = A,D,E,F = 46 = C
- 0
- 0
- 1
- 1
- 0

- 1
- 1
- 0
- 0 = A,B,C,D,E,F = C0 = D
- 0
- 0
- 0
- 0

- 0
- 0
- 0 = A,B,D,E,F,G = 84 = E
- 0
- 0
- 1
- 1
- 0

- 1
- 1
- 1 = A, E, F, G = E = F
- 0
-
-
-
-

- 1
- 0
- 0 = A, B, D, E, F, G = 84 = e
- 0
- 0
- 1
- 0
- 0

- 1
- 1
- 1 = A, E, F, G = E = F
- 0
-
-
-
-

From this process, I could confidently say which binary digit related to which segment:

- 0 -
- 0 – Segment G
- 0 – Segment F
- 0 – Segment E
- 0 – Segment D
- 0 – Segment C
- 0 – Segment B
- 0 – Segment A

Below is the modified code with the Hex digits.

```
=====
```

```
; Main.asm file generated by New Project wizard
```

```

;
; Created: Wed Nov 4 2020
; 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
RP0 equ 5                 ;bit 5 of status register
RP1 equ 6                 ;bit 6 of status register

;=====
; VARIABLES
;=====

count equ 0x0c           ; counter 0 to 255
nCount equ 0x0d          ; pause routine inner loop counter
mCountequ 0x0e           ; pause routine outer loop counter

;=====
; RESET and INTERRUPT VECTORS
;=====

```

```

        ; Reset Vector
RST code 0x0
        goto Start

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

PGM code
Start
        bcf      Status,RP0      ; clear bank selection bits in the status reg
        bcf      Status,RP1

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

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

get_count
        movf     count, w        ; move count to W

        movlw    0xC0
        movwf    portB           ; move W to port B to update the display

        addlw    1               ; add +1 to W
        movwf    count           ; save W in count

```

```

    call    pause          ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause          ; five pause executions equals ~ 1 second

    movf    count, w       ; move count to W

movlw    0xF9
    movwf   portB          ; move W to port B to update the display


    addlw   1              ; add +1 to W
    movwf   count          ; save W in count


    call    pause          ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause          ; five pause executions equals ~ 1 second

    movf    count, w       ; move count to W

movlw    0xA4
    movwf   portB          ; move W to port B to update the display


    addlw   1              ; add +1 to W
    movwf   count          ; save W in count

```

```

    call    pause          ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause          ; five pause executions equals ~ 1 second

    movf    count, w       ; move count to W

movlw    0xB0
    movwf   portB          ; move W to port B to update the display

    addlw   1              ; add +1 to W
    movwf   count          ; save W in count

    call    pause          ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause          ; five pause executions equals ~ 1 second

    movf    count, w       ; move count to W

movlw    0x99
    movwf   portB          ; move W to port B to update the display

    addlw   1              ; add +1 to W
    movwf   count          ; save W in count

    call    pause          ; delay 200ms by subroutine

```

```

    call    pause
    call    pause
    call    pause
    call    pause        ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W

    movlw   0x12
    movwf   portB         ; move W to port B to update the display

    addlw   1              ; add +1 to W
    movwf   count         ; save W in count

    call    pause         ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause         ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W

    movlw   0x82
    movwf   portB         ; move W to port B to update the display

    addlw   1              ; add +1 to W
    movwf   count         ; save W in count

    call    pause         ; delay 200ms by subroutine

```

```
call    pause
call    pause
call    pause
call    pause    ; five pause executions equals ~ 1 second
```

```
movf    count, w    ; move count to W
```

```
movlw   0x78
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```
call    pause
```

```
call    pause
```

```
call    pause
```

```
call    pause
```

```
movf    count, w    ; move count to W
```

```
movlw   0x80
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```
call    pause
call    pause
call    pause
call    pause    ; five pause executions equals ~ 1 second
```

```
movf    count, w    ; move count to W
```

```
movlw   0x90
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```
call    pause
```

```
call    pause
```

```
call    pause
```

```
call    pause    ; five pause executions equals ~ 1 second
```

```
movf    count, w    ; move count to W
```

```
movlw   0x88
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```

    call    pause
    call    pause
    call    pause
    call    pause        ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W

movlw    0x80
    movwf   portB        ; move W to port B to update the display

    addlw   1            ; add +1 to W
    movwf   count        ; save W in count

    call    pause        ; delay 200ms by subroutine
    call    pause
    call    pause
    call    pause
    call    pause        ; five pause executions equals ~ 1 second

    movf    count, w      ; move count to W

movlw    0x46
    movwf   portB        ; move W to port B to update the display

    addlw   1            ; add +1 to W
    movwf   count        ; save W in count

    call    pause        ; delay 200ms by subroutine

```

```
call    pause
call    pause
call    pause
call    pause    ; five pause executions equals ~ 1 second
```

```
movf    count, w    ; move count to W
```

```
movlw   0xC0
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```
call    pause
```

```
call    pause
```

```
call    pause
```

```
call    pause    ; five pause executions equals ~ 1 second
```

```
movf    count, w    ; move count to W
```

```
movlw   0x6
```

```
movwf   portB    ; move W to port B to update the display
```

```
addlw   1    ; add +1 to W
```

```
movwf   count    ; save W in count
```

```
call    pause    ; delay 200ms by subroutine
```

```
call    pause
call    pause
call    pause
call    pause        ; five pause executions equals ~ 1 second
```

```
movf    count, w      ; move count to W
```

```
movlw   0xE
```

```
movwf   portB         ; move W to port B to update the display
```

```
addlw   1              ; add +1 to W
```

```
movwf   count         ; save W in count
```

```
call    pause         ; delay 200ms by subroutine
```

```
call    pause
```

```
call    pause
```

```
call    pause
```

```
call    pause        ; five pause executions equals ~ 1 second
```

```
        ; five pause executions equals ~ 1 second
```

```
        ;subroutine pause: lets waste some time....
```

```
        ;description: two loops using "mCount" and "nCount" register files
```

```
        ;                "mCount" is the outer loop counter (255 down to zero)
```

```
        ;                "nCount" is the inner loop counter (255 down to zero)
```

```
        ;
```

```
        ;-----
```

```
        pause
```

```

        movlw 0xff          ; set w = 255 decimal
        movwf mCount       ; mCount = w: outer loop counter
loadN
        movlw 0xff          ; set w = 255 decimal
        movwf nCount       ; nCount = w: inner loop counter
decN
        decfsz nCount, f    ; nCount--
        goto   decN         ; if nCount != 0 then repeat nCount--

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

```

```

;=====
END

```

I wrote movlw and 0x(Hex digits) in order to load the W register with the Hex code and movwf portB, to move W to port B to update display. I used five pauses as each are equal to 200ms.

I have also re-created this circuit in real life. I have created a Youtube account to visibly demonstrate my working. I have also shown the ability to speed up the process so that the interval between the digits is faster; to do this I could either decrease the outer loop set or I could increase or decrease the amount of pauses. I hope to incorporate a switch to control the speed; this will come with more knowledge.

https://youtu.be/jPmUwXXUP_4

To program this I used Arduino.

FINAL GRADE

GENERAL COMMENTS

Instructor

22/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

GRADING FORM: GJ_2019

JAKE RUDD**110****ENGAGEMENT WITH LITERATURE**

"this section will be concerned with your ability to obtain relevant information on the components used in the "HND/BENG PIC16F84 Assignment Board" circuit." Unfortunately - this section has not been attempted at all

KNOWLEDGE AND UNDERSTANDING

"Marks will be allocated based on the demonstration and extent of knowledge and understanding of the general concepts and underlying principles of the circuit operation and the individual components." Unfortunately - this section has not been attempted at all

COGNITIVE AND INTELLECTUAL

"For each task below, produce a flowchart or pseudo-code " This has not been done.

20**PRACTICAL APPLICATION**



A working proteus simulation produced and the first programming task completed. Good description of the LED number calculation method. Could have done with more tasks attempted though for higher marks.

50

TRANSFERABLE SKILLS



Reports should have a cover page with the name of the student as well as the assignment title. Also should have a table of contents, and formatted with appropriate sections etc. Should have a references section. Needs more content - much of the assignment simply hasn't been done.

40