

EET 250 PIC LAB – Build a 7 SEGMENT Display Driver

7 Segment Displays play an important role in Microcontroller based products. This lab will deal with 7 Segment Display Driver using PIC. Part 1 demos the basic display. Part 2 integrates switch bounce and count with display. Part 3 simulates a single dice.

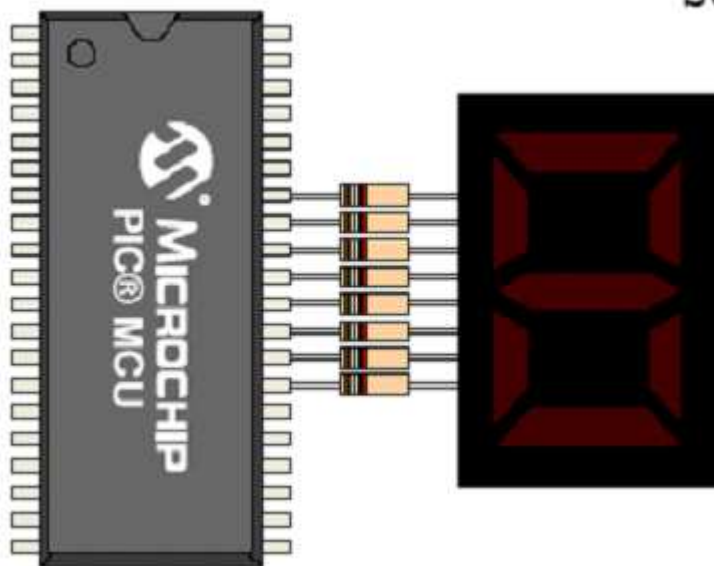
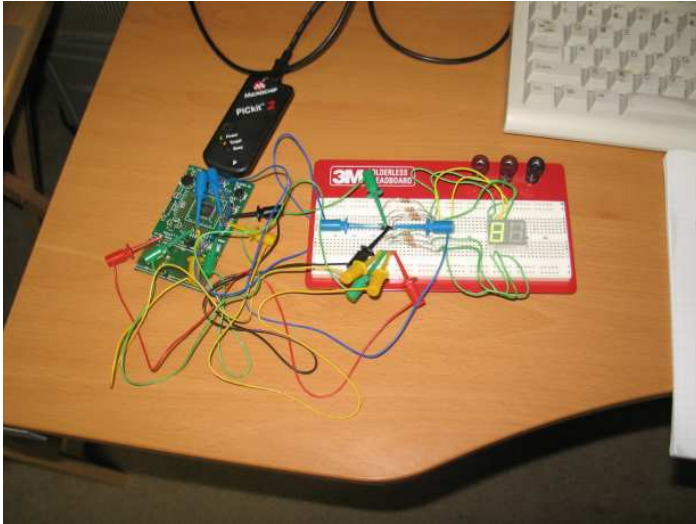


Figure 1 Notional Hook-up

The schematic for the PICKIT2 kit is shown. It contains all 8 LEDs connected to PORTD of the PIC and a Pushbutton connected to PORTB pin 0. Remove jumper JP1 from the

board. Use 470 ohm resistors between PIC and designated Display pins. The 7 SEGMENT Display needs to be hooked as follows:

VDD to Common Anode

RD7 to 220 ohms to A

RD6 to 220 ohms to B

RD5 to 220 ohms to C

RD4 to 220 ohms to D

RD3 to 220 ohms to E

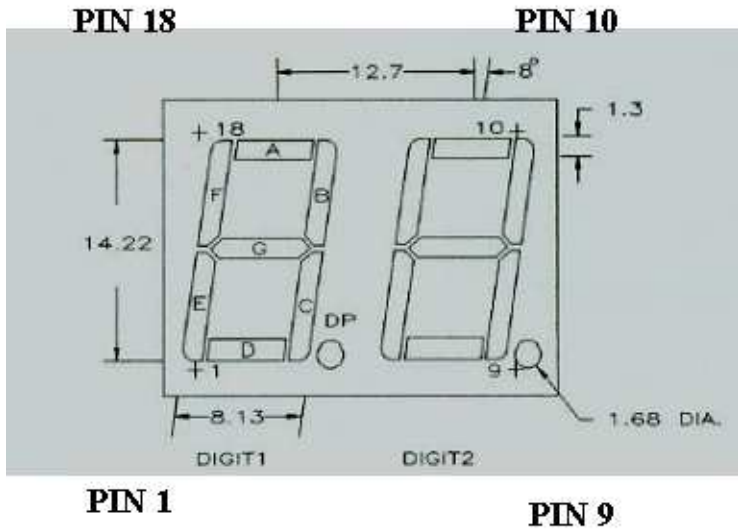
RD2 to 220 ohms to F

RD1 to 220 ohms to G

RD0 (no connected)

This lab will use the PICKIT2 as debugger and programmer for this lab. The lab consists of core assembly language file SEG7.ASM (Listing provided)

The labs will also make us of Specific Function Registers for PORT and TRIS Control. Bank switching will be required to probably access this registers. Note that to light an LED segment since the display is common anode is to provide the inverse of the LED segment pin for lighting. What happens to all segments when 0xff is written to PORTD? What happens when 0X00 is outputted?



UA56X2-11
COMMON ANODE

PIN NO.	FUNCTION
1	CATHODE E (DIGIT 1)
2	CATHODE D (DIGIT 1)
3	CATHODE C (DIGIT 1)
4	CATHODE DP (DIGIT 1)
5	CATHODE E (DIGIT 2)
6	CATHODE D (DIGIT 2)
7	CATHODE G (DIGIT 2)
8	CATHODE C (DIGIT 2)
9	CATHODE DP (DIGIT 2)
10	CATHODE B (DIGIT 2)
11	CATHODE A (DIGIT 2)
12	CATHODE F (DIGIT 2)
13	DIGIT 2 COMMON ANODE
14	DIGIT 1 COMMON ANODE
15	CATHODE B (DIGIT 1)
16	CATHODE A (DIGIT 1)
17	CATHODE G (DIGIT 1)
18	CATHODE F (DIGIT 1)

PIC	SEGMENT
RD7	A PIN16
RD6	B PIN15
RD5	C PIN 3
RD4	D PIN 2
RD3	E PIN 1
RD2	F PIN18
RD1	G PIN17
RD0	Not connected

Figure 2 7 SEG Display

1.0 Experiment 1 Displaying 7 Segments

This code configures PORTD for output 7 SEG LED and Delay Subroutine. The code simply counts and uses counter to look-up the 7 SEGMENT display. Display should vary from 0 to 7 and repeat. Delay is used to pause between updates. Build and run program. Then update the program and Note areas where student must originate code.
Instructor Signoff_____

```
#include <p16F917.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF & _MCLRE_OFF &
_CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF & _FCMEN_OFF)
```

```
        cblock 0x020
Delay1      ; Assign an address to label Delay1
Delay2
INDEX
        endc

        org     0x0000
RESET_V     goto     START      ;Reset Vector

START clrf     PORTD      ;Clear PORTD output latches
        bsf     STATUS,RP0 ;Switch to bank 1
        movlw  b'00000000' ;Load value to make lower 4 bits outputs
        movwf  TRISD     ;Move value to TRISD
        bcf     STATUS,RP0 ;Switch to bank 0
        clrf   INDEX     ;Clear index into table
        movlw  HIGH TABLE ;Load W with high byte of TABLE address
        movwf  PCLATH    ;Move W to PCLATH

LOOP  movf   INDEX,w      ;Move INDEX to W
        call  TABLE     ;Call TABLE
        movwf PORTD     ;Move W to PORTD
        movlw .255
        call  Delay      ;Call delay
        movlw .255
        call  Delay      ;Call delay
        incf  INDEX,f    ;Increment INDEX
***** 2nd part only; student to modify max count *****


---


        movlw  0x08      ;Load W with max count+1
        subwf  INDEX,w    ;Subtract W from INDEX, result in W
        btfsc STATUS,Z    ;Is Z bit in STATUS set?
        clrf  INDEX     ;if yes, clear INDEX
        goto  LOOP      ;Repeat
TABLE addwf  PCL,f      ;Add offset to program counter

        retlw  b'00000011' ;Table entry 0 I
        retlw  b'10011111' ;Table entry 1 I
        retlw  b'00100101' ;Table entry 2 I
        retlw  b'00001101' ;Table entry 3 I
        retlw  b'10011001' ;Table entry 4 I
        retlw  b'01001001' ;Table entry 5 I
        retlw  b'01000001' ;Table entry 6 I
        retlw  b'00011111' ;Table entry 7 I
```

******2nd part only; student to add code below*

```
;      retlw  b'xxxxxxx' ;Table entry 8 I
```

```
;      retlw  b'xxxxxxx' ;Table entry 9 I
```

Delay:

```
movwf  Delay2      ;
```

DelayLoop:

```
decfsz Delay1,f    ; Waste time.
```

```
goto   DelayLoop   ; The Inner loop takes 3 instructions per loop * 256 loops = 768 instructions
```

```
decfsz Delay2,f    ; The outer loop takes an additional 3 instructions per lap * 256 loops
```

```
goto   DelayLoop   ; (768+3) * 256 = 197376 instructions / 1M instructions per second = 0.197 sec.
```

```
                ; call it two-tenths of a second.
```

```
return
```

```
END
```

1. Enter code and select as source for your project. Build the code and download into DEMO board. Press RUN icon on MPLAD—you should see an activity display at bottom of MPLAB indicating DEMO is running. What happens?
2. Modify code to display all digits from zero to nine. Show instructor

2.0 Experiment 2 counting switch closures.

This code expands on SEG7.ASM and adds input push button as the source for count. Look at earlier program DEBOUNCE.Asm for guidance. Add switch and debounce feature so that each press of pushbutton causes the 7 Segment display to increment from 0 to 9 and repeat.

Instructor Signoff_____

```
#include <p16F917.inc>
```

```
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF &  
_MCLRE_OFF & _CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF &  
_FCMEN_OFF)
```

```
cblock 0x020
```

```
Delay1      ; Assign an address to label Delay1
```

```
Delay2
```

```
Counter     ; define a variable to hold the display
```

```
endc
```

```
org 0x0000
```

```
RESET_V goto START ;Reset Vector
```

```

START    clrf   PORTD           ;Clear PORTD output latches
         bsf    STATUS,RP0 ;Switch to bank 1
         movlw  b'00000000' ;Load value to make lower 4 bits outputs
         movwf  TRISD          ;Move value to TRISD
         movlw  0x01
         movwf  TRISB          ; Make RBO pin input (switch)
         bcf    STATUS,RP0 ;Switch to bank 0
         movlw  b'00000011'   ; turn off all leds
         movwf  PORTD
         movlw  0x00          ; turn off all leds
         movwf  Counter       ;zero counter

```

loop:

```

         btfsc  PORTB,0        ; test input switch
         goto  loop           ;wait for button to turn on

```

```

         movlw  .13           ; Delay another 10mS plus whatever
         call  Delay
         btfsc  PORTB,0        ; still depressed?
         goto  loop           ;wait for button to turn on
         incf  Counter
         movf  Counter,w
         Call  Update;

```

```

wait:    btfss  PORTB,0        ;wait for button to be released
         goto  wait
         goto  loop          ;

```

turnon:

```

         bsf   PORTD,0        ; turn on LED RD0 (DS0)
         goto  loop          ; loop back

```

turnoff:

```

         bcf   PORTD,0        ; turn off LED RD0
         goto  loop          ; loop back

```

Update:

```

         movlw 0x0A           ;Load W with 10
         subwf  Counter,w     ;Subtract W from INDEX, result in W
         btfsc STATUS,Z      ;Is Z bit in STATUS set?
         clrf  Counter       ;if yes, clear INDEX

```

```

         movlw HIGH TABLE;Load W with high byte of TABLE address
         movwf PCLATH        ;Move W to PCLATH
         movf  Counter,w
         call  TABLE        ;Call TABLE
         movwf PORTD        ;Move W to PORTD

```

```

        return
TABLE    addwf PCL,f      ;Add offset to program counter

        retlw b'00000011' ;Table entry 0 I
        retlw b'10011111' ;Table entry 1 I
        retlw b'00100101' ;Table entry 2 I
        retlw b'00001101' ;Table entry 3 I
        retlw b'10011001' ;Table entry 4 I
        retlw b'01001001' ;Table entry 5 I
        retlw b'01000001' ;Table entry 6 I
        retlw b'00011111' ;Table entry 7 I
        retlw b'00000000' ;Table entry 8 I
        retlw b'00011000' ;Table entry 9 I

Delay:
        movwf Delay2      ;
DelayLoop:
        decfsz Delay1,f   ; Waste time.
        goto DelayLoop   ; The Inner loop takes 3 instructions per loop * 256 loopss =
768 instructions
        decfsz Delay2,f   ; The outer loop takes and additional 3 instructions per lap * 256
loops
        goto DelayLoop   ; (768+3) * 256 = 197376 instructions / 1M instructions per
second = 0.197 sec.
        ; call it two-tenths of a second.
        return
        END

```

3.0 Simulating a DICE using Hardware setup.

This program reuses a lot of experiment 2 to display a random number at the display between 1 and 6 when the input button is pressed. Notice new subroutine roll.

```
#include <p16F917.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF &
_MCLRE_OFF & _CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF &
_FCMEN_OFF)
```

```
; SEG7 debounce modified for dice
cblock 0x020
Delay1 ; Assign an address to label Delay1
Delay2
Counter ; define a variable to hold the display

endc
```

```
org 0x0000
RESET_V goto START ;Reset Vector

START clrf PORTD ;Clear PORTD output latches
bsf STATUS,RP0 ;Switch to bank 1
movlw b'00000000' ;Load value to make lower 4 bits outputs
movwf TRISD ;Move value to TRISD
movlw 0x01
movwf TRISB ; Make RBO pin input (switch)
bcf STATUS,RP0 ;Switch to bank 0
movlw 0xff ; turn off all leds
movwf PORTD
movlw 0x06 ; turn off all leds
movwf Counter ;zero counter
```

```
loop: call roll
; test input switch
btfsc PORTB,0 ;wait for button to turn on
goto loop
movlw .13 ; Delay another 10mS plus whatever
call Delay
call roll
btfsc PORTB,0 ; still depressed?
goto loop ;wait for button to turn on
;call nexnum
; incf Counter
; movf Counter,w
Call Update;
wait: btfss PORTB,0 ;wait for button to be released
```

```

                goto wait
                goto loop    ;
roll: decfsz Counter;
        return;
reload: movlw 0x06
        movwf Counter;
        return;

```

Update:

```

        movlw HIGH TABLE;Load W with high byte of TABLE address
        movwf PCLATH          ;Move W to PCLATH
        movf Counter,w
        call TABLE           ;Call TABLE
        movwf PORTD          ;Move W to PORTD
        return

```

```

TABLE    addwf PCL,f          ;Add offset to program counter

```

```

        retlw b'00000011' ;Table entry 0 I
        retlw b'10011111' ;Table entry 1 I
        retlw b'00100101' ;Table entry 2 I
        retlw b'00001101' ;Table entry 3 I
        retlw b'10011001' ;Table entry 4 I
        retlw b'01001001' ;Table entry 5 I
        retlw b'01000001' ;Table entry 6 I
        retlw b'00011111' ;Table entry 7 I
        retlw b'00000000' ;Table entry 8 I
        retlw b'00011000' ;Table entry 9 I

```

Delay:

```

        movwf Delay2        ;
DelayLoop:
        decfsz Delay1,f    ; Waste time.
        goto DelayLoop    ; The Inner loop takes 3 instructions per loop * 256 loopss =
768 instructions

```

```
    decfsz Delay2,f    ; The outer loop takes and additional 3 instructions per lap * 256
loops
    goto DelayLoop    ; (768+3) * 256 = 197376 instructions / 1M instructions per
second = 0.197 sec.
                    ; call it two-tenths of a second.
return
    END
```