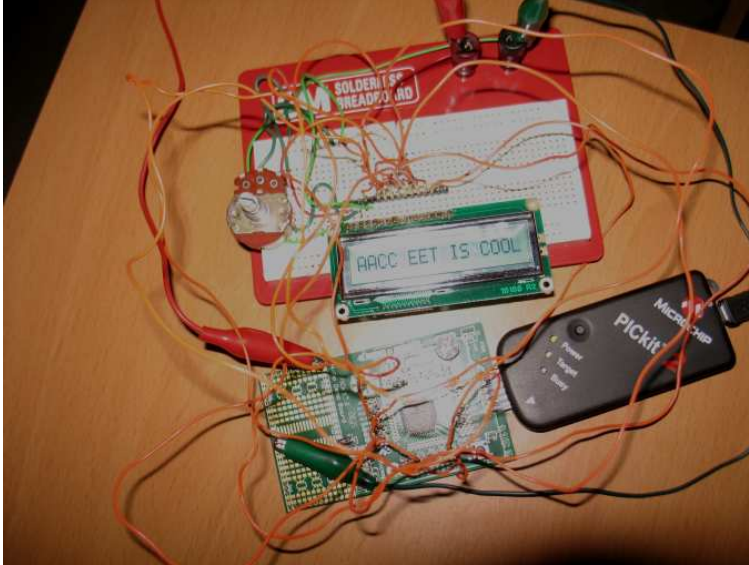


Lab Driving LCD Displays with PIC

This lab will illustrate use of PICKIT 2 and MPLAB in the process of assembly, execution and debug of an LCD 1x16 Character module. Code and hardware setups are shown. Part 1 demos a string output. Part 2 denounces a switch and maintains and displays a switch closure count on LCD from 0 to 255.



Part 1

Integrate a LCD with PICKIT2 module as shown

Make sure to use potentiometer provided as contrast control

Before power up show construction to instructor. Remove jumper JP1 form PICKIT.

Power up and adjust contrast for readability

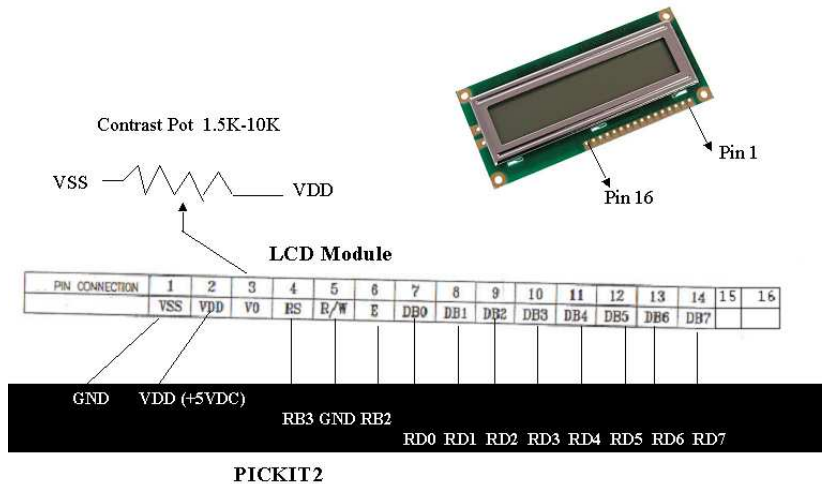


Figure 1 Hook-up Instructions

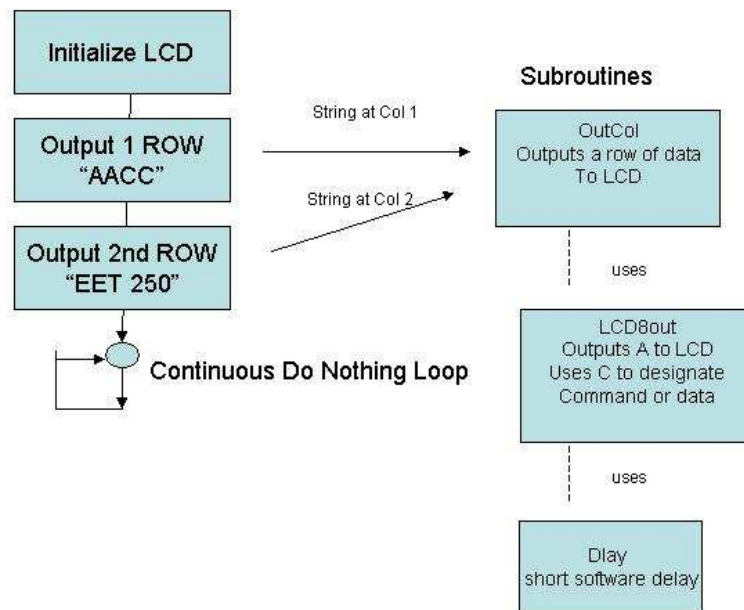
Enter Code

There are two basic subroutines one to command the LCD form proper operation, then other to output data. A long delay and short delay subroutine is also used for proper command and data timing to LCD. Note that the display is configured as two- 1x8 (one row –eight column) display. The display data has to be in ASCII. Try changing the message. Remember you are restricted to eight characters at a time for each half of the 16-character display.

Execute

Module Should Display "AACC EET 250 is Cool"

Part 1 Flowchart



Part 1 Source.

```
*****
;
;AACC LCD LED
;
;
; This turns on LCD
; LCD Control at:
; RD7:RD0 - LCD Bits
; RB3 - LCD RS Bit
; RB2 - LCD E Bit

#include <p16F917.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF & _MCLRE_OFF &
_CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF & _FCMEN_OFF)
#define E PORTB, 2
#define RS PORTB,3
CBLOCK 0x020 ; Start Registers at End of the Values

Delay1 ; Assign an address to label Delay1
Delay2
ENDC
org 0
Start:
    bsf STATUS,RP0 ; select Register Page 1
    clrf TRISD ; Make PortD LEDS all output
        movlw 0x00
        movwf TRISB ; Make RBO pin input (switch)

    bcf STATUS,RP0 ; back to Register Page 0
    movlw 0x00 ; turn off all leds
    movwf PORTD

    movwf PORTB
; Initialize the LCD
LCDINIT:
    call DelayL
    call DelayL
    movlw 0x38 ;enable cursor display
    call LCD8CMD
    call DelayL
    movlw 0x0f ;set cursor direction
    call LCD8CMD
    call DelayL
    movlw 0x06 ;enable cursor display
    call LCD8CMD
    call DelayL
    movlw 0x01 ;clear display
    call LCD8CMD
    call DelayL
    movlw 0x10 ;enable display cursor
    call LCD8CMD
    call DelayL
```

```

movlw 0x03      ;return cursor home
call LCD8CMD
call DelayL
call DelayS
movlw 'A'
call LCD8Data
call DelayS
movlw 'A'
call LCD8Data
call DelayS
movlw 'C'
call LCD8Data
call DelayS
movlw 'C'
call LCD8Data
call DelayS
movlw ' '
call LCD8Data
call DelayS
movlw 'E'
call LCD8Data
call DelayS
movlw 'E'
call LCD8Data
call DelayS
movlw 'T'
call LCD8Data
call DelayS
;switch to column 2
movlw 0xc0
call LCD8CMD
call DelayL
movlw ' '
call LCD8Data
call DelayS
movlw 'I'
call LCD8Data
call DelayS
movlw 'S'
call LCD8Data
call DelayS
movlw ' '
call LCD8Data
call DelayS
movlw 'C'
call LCD8Data
call DelayS
movlw 'O'
call LCD8Data
call DelayS
movlw 'O'
call LCD8Data
call DelayS
movlw 'L'
call LCD8Data
call DelayS

```

```

loop:
    goto loop

; Mainline
LCD8CMD: bcf RS
        goto LCDCOM
LCD8Data: bsf RS
LCDCOM:
        movwf PORTD

        bsf E
        bcf E
        return

; Delay Function. Enter with number of 771uS delays in Wreg
DelayL: movlw .20          ; 20 msec
        goto Delay
DelayS: movlw .1          ;771 uss
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

end

```

Part 2

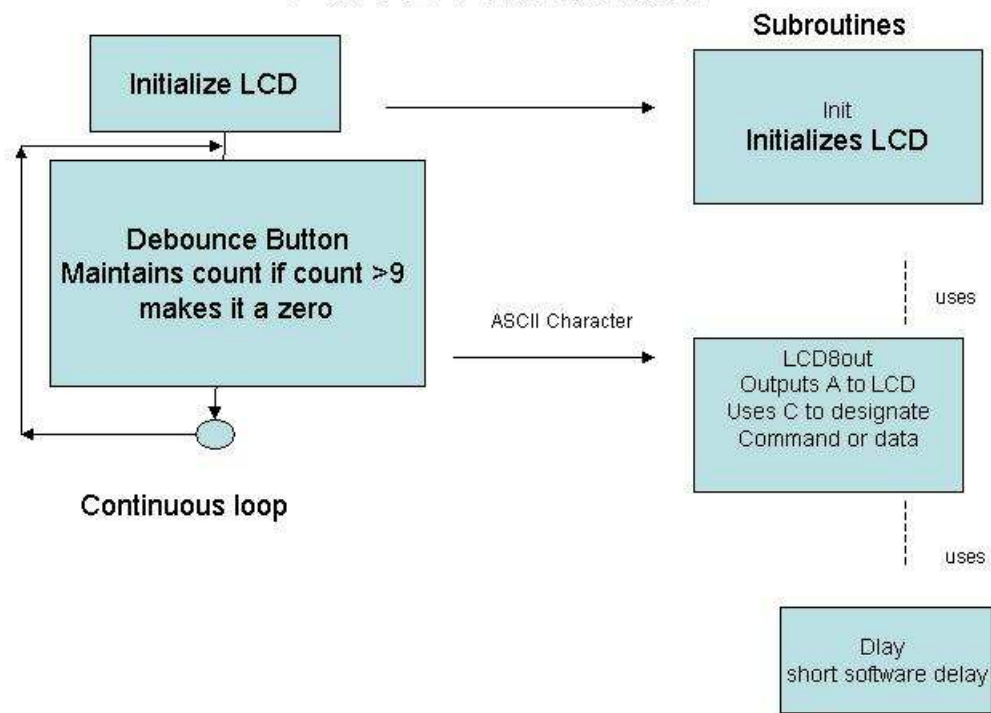
Use lab 1 setup and add pushbutton

Enter Code

Execute

Module should running count of number of pushbuttons from 0 to 9. it incorporates a new subroutine that converts binary byte to decimal three values units, tens and hundreds and then convert the decimal to ACSII by adding 0x30 header to each decimal byte. The software then outputs the ASCII string to the display as current button count in the column two position of the display..

Part 2 Flowchart



Part 2 Source Code

```
*****  
;AACC LCD Driver
```

```

; LCD test4 debounces switch runs a counter that is converted to ASCII and outputed to LCD
;
; This turns on LCD
; LCD Control at:
; RD7:RD0 - LCD Bits
; RB3 - LCD RS Bit
; RB2 - LCD E Bit
; RB0 is input switch

#include <p16F917.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF & _MCLRE_OFF &
_CP_OFF & _CPD_OFF & _BOD_OFF & _IESO_OFF & _FCMEN_OFF)
#define E PORTB, 2
#define RS PORTB, 3
CBLOCK 0x020 ; Start Registers at End of the Values
Delay1 ; Assign an address to label Delay1
Delay2
Hundreds
Tens
Ones ; Number ASCII Variables
Temp ; number to be converted
counter
ENDC
org 0
;initialize I/O
Start:

        bsf STATUS,RP0 ; select Register Page 1
        clrf TRISD ; Make PortD LEDES all output
        movlw 0x01
        movwf TRISB ; Make RBO pin input (switch) and
; all rest outputs

        bcf STATUS,RP0 ; back to Register Page 0
        movlw 0x00 ; turn off all leds
        movwf PORTD
        movwf PORTB

        movwf Temp
        movwf counter
        movlw '0' ; Initialize each of the Variables to "0"
        movwf Hundreds
        movwf Tens
        movwf Ones

; Initialize the LCD
LCDINIT:
        call DelayL ; good long delay to start
        call DelayL
        movlw 0x38 ;enable cursor display
        call LCD8CMD

        movlw 0x0f ;set cursor direction
        call LCD8CMD

        movlw 0x06 ;enable cursor display
        call LCD8CMD

```

```

movlw 0x01    ;clear display
call LCD8CMD

movlw 0x10    ;enable display cursor
call LCD8CMD

movlw 0x03    ;return cursor home
call LCD8CMD

```

```

movlw 'C'
call LCD8Data

```

```

movlw 'O'
call LCD8Data

```

```

movlw 'U'
call LCD8Data

```

```

movlw 'N'
call LCD8Data

```

```

movlw 'T'
call LCD8Data

```

```

movlw '='
call LCD8Data

```

```

movlw ' '
call LCD8Data

```

```

movlw ' '
call LCD8Data
movlw 0xc0
call LCD8CMD
call bintoASCII
movf Hundreds,w
call LCD8Data

```

```

movf Tens,w
call LCD8Data

```

```

movf Ones,w
call LCD8Data

```

```

;switch to column 2

```

```

loop:

```

```

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

```

```

call DelayS
    btfsc PORTB,0    ; still depressed?
    goto loop        ;wait for button to turn on
    incf counter

```

```

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

           movlw 0xc0
           call LCD8CMD
           movlw '0'             ; Initialize each of the Variables to "0"
           movwf Hundreds
           movwf Tens
           movwf Ones

movf counter,w
movwf Temp
call bintoASCII
movf Hundreds,w
call LCD8Data

movf Tens,w
call LCD8Data

movf Ones,w
call LCD8Data

goto loop

; Command to LCD
LCD8CMD: bcf RS
           movwf PORTD
           bsf E
           bcf E
           call DelayL
           return

;data to LCD
LCD8Data: bsf RS
           movwf PORTD
           bsf E
           bcf E
           call DelayS
           return

; Delay Function. Enter with number of 771uS delays in Wreg
DelayL: movlw .20                ; 20 msec
           goto Delay

DelayS: movlw .1                 ;771 uss
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
bintoASCII:
           ; Use TEMP to convert binary to ASCII
           movf Temp,w

```

```

HundredsLoop:          ; Return Here Each Time to Test for Temp >= 100
    movlw .100         ; Is Temp >= 100?
    subwf Temp, w
    btfss STATUS, C    ; If STATUS.C Set, then "yes"
    goto TensLoop
    movwf Temp         ; Save the Result
    incf Hundreds, f   ; Increment the Number of 100s
    goto HundredsLoop

TensLoop:              ; Return Here Each Time to Test for Temp >= 10
    movlw .10         ; Is Temp >= 10?
    subwf Temp, w
    btfss STATUS, C    ; If STATUS.C Set, then "yes"
    goto DoOnes
    movwf Temp         ; Save the Result
    incf Tens, f       ; Increment the Number of 10s
    goto TensLoop

DoOnes:                ; Remainder is the Number of Ones
    movf Temp, w
    addwf Ones, f

    return
    end

```