```
    .title "EL308"
    .sbttl "Keyboard Buffer Example"
    .equ __24FJ256GB110, 1
    .include "p24FJ256GB110.inc"

    .global __reset          ;The label for the first line of code.
    .global __T1Interrupt    ;Declare Timer 1 ISR name global
    .global __CNInterrupt

.bss
    LCD_line1:  .space 16
    LCD_line2:  .space 16
    LCD_ptr:    .space 2
    LCD_cmd:    .space 2
    LCD_offset: .space 2
    BUFF_start: .space 2
    BUFF_stop:  .space 2
    BUFF_cnt:   .space 2
    BUFF_array: .space 8        ; buffer depth = 4
.section .const,psv
    line1:      .ascii "                "
    line2:      .ascii "                "
    lookup:     .ascii "0123456789ABCDEF"
.text                             ;Start of Code section
__reset:
    mov     #__SP_init, W15     ; Initalize the Stack Pointer
    mov     #__SPLIM_init, W0   ; Initialize the Stack Pointer Limit Register
    mov     W0, SPLIM
    nop                         ; Add NOP to follow SPLIM initialization

    call    init_PSV
    call    init_LED
    call    init_LCD
    call    init_message
    call    init_keypad
    call    init_buzzer

    call    init_timer

    bclr    IFS1, #CNIF         ; clear CN interrupt flag

    bclr    IPC4, #CNIP2
    bclr    IPC4, #CNIP1
    bset    IPC4, #CNIP0        ; set CN interrupt priority to 001

    bset    CNEN1, #CN15IE      ; enable interrupts for CN15 (port D bit 6)
    bset    IEC1, #CNIE         ; enable CN interrupt

    mov     #0, W0
    mov     W0, BUFF_cnt        ; set buffer count ot 0
    mov     W0, BUFF_start      ; set start-of-buffer pointer to 0
    mov     W0, BUFF_stop       ; set end-of buffer pointer to 0

    bset    AD1PCFG, #PCFG7     ; use the ADC as digital I/O (1-bit ADC)

    mov     #LCD_line1, W5      ; W5 acts as pointer to LCD array
    mov     #0, W6              ; W6 is used as an offset to W5

main_loop:
    btsc    PORTB, #7           ; POT reading > 1.65 V ?
    bra     main_loop           ; If yes, block the execution.

    mov     BUFF_cnt, W0
    cp0     W0                  ; Any key code in buffer
    bra     Z, no_key           ; If NO, do nothing
    dec     W0, W0              ; Decrement character count in buffer
    mov     W0, BUFF_cnt        ; store new count value

    mov     #BUFF_array, W0     ; get pointer to buffer
    mov     BUFF_stop, W1       ; get offset in buffer
    mov.b   [W0+W1], W2         ; get the current character
    mov.b   W2, [W5+W6]         ; display on LCD
    inc     W6, W6              ; increment display pointer
    and     W6, #0x1F, W6       ; wrap from 32 to 0
    inc     W1, W1              ; increment end-of-buffer pointer
    and     W1, #0x03, W1       ; wrap from 4 to 0
```

```
    mov     W1, BUFF_stop       ; store new value

no_key:
    bra     main_loop

; -------------------------------------------------------
; !!!!!!!!!!!!!!!!! CNInterrupt !!!!!!!!!!!!!!!!!!!!!!!!
; -------------------------------------------------------

__CNInterrupt:
    push    W0
    push    W1
    push    W2

    btss    PORTD, #6           ; is the key pressed or released?
    bra     key_release         ; releases

    mov     BUFF_cnt, W0
    cp      W0, #4              ; any space in buffer?
    bra     Z, buffer_full      ; NO, the buffer is full
    inc     W0, W0              ; YES, increment buffer counter
    mov     W0, BUFF_cnt        ; store new value

    mov     PORTD, W0
    and     W0, #0x000F, W0
    mov     #psvoffset(lookup), W1
    mov.b   [W0+W1], W2         ; get character code
    mov     #BUFF_array, W0
    mov     BUFF_start, W1
    mov.b   W2, [W0+W1]         ; store in buffer
    inc     W1, W1              ; increment start-of-buffer pointer
    and     W1, #0x03, W1       ; wrap from 4 to 0
    mov     W1, BUFF_start      ; store new value
    bra     done_CNint          ; we are done

buffer_full:
    bset    PORTD, #13          ; buffer full, turn buzzer on
    bra     done_CNint

key_release:                    ; key released, turn buzzer off
    bclr    PORTD, #13

done_CNint:
    bclr    IFS1, #CNIF         ; clear CNInterrupt flag

    pop     W2
    pop     W1
    pop     W0                  ; restore registers
    retfie

; -------------------------------------------------------
; !!!!!!!!!!!!!!!!! Functions !!!!!!!!!!!!!!!!!!!!!!!!!
; -------------------------------------------------------

init_PSV:
    mov     #psvpage(line1), W0
    mov     W0, PSVPAG      ; set PSVPAG to page that contains hello
    bset.b  CORCONL,#PSV    ; enable Program Space Visibility
    return

init_timer:
    bclr    T1CON, #TON     ; turn timer1 OFF

    bset    T1CON, #TCKPS1
    bset    T1CON, #TCKPS0  ; set prescaler to 256

    bclr    T1CON, #TCS     ; select internal clock

    mov     #0x0000, W0
    mov     W0, TMR1        ; clear TMR1 register
    mov     #0x0040, W0
    mov     W0, PR1         ; set timer1 period to 0x0040 -> f=2e6/256/64=122 Hz

    bclr    IPC0, #14
    bclr    IPC0, #13
    bset    IPC0, #12       ; set timer1 priority to 001
```

```
    bclr    IFS0, #T1IF     ; clear timer1 interrupt status flag
    bset    IEC0, #T1IE     ; enable timer1 interrupts

    bset    T1CON, #TON     ; turn timer1 ON
    return

init_LED:
    bclr    TRISF, #0
    bclr    TRISF, #1
    bclr    TRISF, #2
    bclr    TRISF, #3       ; LED array
    return

init_LCD:
    bclr    TRISB, #15
    bclr    PORTD, #4       ; make sure LCD is disabled before port is set to output mode
    bclr    TRISD, #4
    bclr    TRISD, #5
    mov     #0xFF00, W0
    mov     W0, TRISE

    bclr    PORTD, #5       ; select LCD WR mode

    mov     #0x0038, W0     ; init LCD
    call    sendcomm
    call    dly
    call    dly
    call    dly

    mov     #0x000C, W0     ; LCD on, cursor off
    call    sendcomm
    mov     #0x0001,W0      ; clear LCD
    call    sendcomm
    return

sendcomm:
    bclr    PORTB,#15   ; select LCD command register
    mov     W0, PORTE   ; output command
    bset    PORTD, #4
    call    dly
    nop
    bclr    PORTD, #4
    call    dly
    return

dly:
    mov     #0x2000,W0
dlyloop:
    sub     W0, #1, W0
    bra     NZ, dlyloop
    return

init_message:
    mov     #0x0000, W0
    mov     W0, LCD_ptr
    mov     W0, LCD_offset
    mov     #0x00C0, W0
    mov     W0, LCD_cmd
    mov     #psvoffset(line1), W1
    mov     #LCD_line1, W2
    repeat  #15
    mov.b   [W1++], [W2++]
    mov     #psvoffset(line2), W1
    mov     #LCD_line2, W2
    repeat  #15
    mov.b   [W1++], [W2++]
    return

init_keypad:
    bset    TRISD,#0    ; DATA A
    bset    TRISD,#1    ; DATA B
    bset    TRISD,#2    ; DATA C
    bset    TRISD,#3    ; DATA D
    bset    TRISD,#6    ; DATA Available
    return
```

```
init_buzzer:
    bclr    PORTD, #13  ; buzzer initially OFF
    bclr    TRISD, #13  ; enable output
    return


;.....................................................................
;Timer 1 Interrupt Service Routine
;Example context save/restore in the ISR performed using PUSH.D/POP.D
;instruction. The instruction pushes two words W4 and W5 on to the stack on
;entry into ISR and pops the two words back into W4 and W5 on exit from the ISR
;.....................................................................
__T1Interrupt:
    push.s
    push.d  W4                  ; Save context using double-word PUSH

        ;<<insert user code here>>
    bclr    IFS0, #T1IF           ; Clear the Timer1 Interrupt flag Status
                                  ; bit.

    clrwdt  ; !!!!!!!!!!! Very bad practice! !!!!!!!!!!!!!!!

    mov     LCD_ptr, W2
    mov     #0x0010, W1
    cp      W1, W2
    bra     NZ, send_LCD_data
    mov     LCD_cmd, W0
    bclr    PORTB, #15      ; select LCD command register
    mov     W0, PORTE       ; output command
    bset    PORTD, #4
    nop
    bclr    PORTD, #4
    btg     W0, #6
    mov     W0, LCD_cmd
    mov     #0x0000, W2
    mov     W2, LCD_ptr
    mov     LCD_offset, W0
    btg     W0, #4
    mov     W0, LCD_offset
;   btg     PORTF, #2
    bra     done_T1interrupt
send_LCD_data:
    mov     LCD_offset, W3
    add     W3, W2, W3
    mov     #LCD_line1, W1
    mov.b   [W1+W3], W0
    bset    PORTB, #15      ; select LCD data register
    mov     W0, PORTE       ; output command
    bset    PORTD, #4
    nop
    bclr    PORTD, #4
    inc     W2, W2
    mov     W2, LCD_ptr

    mov     PORTB, W0
    mov     #0x0080, W1
    and     W0, W1, W0
    lsr     W0, #4, W2
    mov     BUFF_cnt, W0
    add     W0, W2, W2
    mov     LATF, W0
    mov     #0xFFF0, W1
    and     W0, W1, W0
    ior     W0, W2, W0
    mov     W0, LATF

done_T1interrupt:
    pop.d W4                  ;Retrieve context POP-ping from Stack
    pop.s
    retfie                    ;Return from Interrupt Service routine

;--------End of All Code Sections ---------------------------------------------

.end                              ;End of program code in this file
```