```
    .title "EL308"
    .sbttl "Digital Door Lock"
    .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 __T2Interrupt    ;Declare Timer 2 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
    KPD_buff:   .space 2
    KPD_stat:   .space 2
    State_Var:  .space 2
    Prev_State: .space 2
    Time_Count: .space 2
    Time_Inc:   .space 2
    Entry:      .space 4
.section .const,psv
    line1:      .ascii "Password ->     "
    line2:      .ascii "                "
    lookup:     .ascii "*0*E123*456*789B"
    stt0:       .byte 0,1,0,0, 1,1,1,0, 1,1,1,0, 1,1,1,0
    stt1:       .byte 1,2,1,1, 2,2,2,1, 2,2,2,1, 2,2,2,0
    stt2:       .byte 2,3,2,2, 3,3,3,2, 3,3,3,2, 3,3,3,1
    stt3:       .byte 3,4,3,3, 4,4,4,3, 4,4,4,3, 4,4,4,2
    stt4:       .byte 4,4,4,5, 4,4,4,4, 4,4,4,4, 4,4,4,3
    stt5:       .byte 5,5,5,5, 5,5,5,5, 5,5,5,5, 5,5,5,5
    password:   .ascii "1234"
    fn_table:   .word handle(fn0)
                .word handle(fn1)
                .word handle(fn2)
                .word handle(fn3)
                .word handle(fn4)
                .word handle(fn5)
.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

        ;<<insert more user code here>>

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

    call    init_timer
    call    init_timer2

    mov     #0, W0
    mov     W0, State_Var
    mov     W0, Time_Count
    mov     W0, Time_Inc

MainLoop:

;   mov     State_Var, W0
;   mov     #LCD_line2, W1
;   add.b   #'0', W0
;   mov.b   W0, [W1]

    mov     KPD_stat, W0
    cp0     W0
    bra     Z, NoKey
    mov     #0, W0
    mov     W0, KPD_stat
```

```
    mov     #psvoffset(stt0), W1
    mov     State_Var, W0
    mov     W0, Prev_State
    sl      W0, #4, W0          ; 16 bytes per state
    add     W0, W1, W1
    mov     KPD_buff, W0
    mov     #0, W2
    mov.b   [W0+W1], W2         ; W2 gets new state
    mov     W2, State_Var
    add     W2, W2, W2
    mov     #psvoffset(fn_table), W1
    mov     [W1+W2], W2
    call    W2

NoKey:

    mov     Time_Count, W0
    mov     #4, W1              ; 4 seconds is enough
    cp      W0, W1
    bra     NZ, no_timeout
    mov     #0, W0
    mov     W0, Time_Count
    mov     W0, State_Var
    call    fn0
no_timeout:
    clrwdt                      ; Kick the dog
    bra     MainLoop

fn0:
    push    W0
    push    W1
    mov     #LCD_line1, W0
    mov.b   #' ', W1
    mov.b   W1, [W0+12]
    mov.b   W1, [W0+13]
    mov.b   W1, [W0+14]
    mov.b   W1, [W0+15]
    mov     #LCD_line2, W0
    mov.b   W1, [W0+14]
    mov.b   W1, [W0+15]
    bclr    PORTF, #0
    pop     W1
    pop     W0
    return

fn1:
    push    W0
    push    W1
    push    W2
    mov     #0, W1
    mov     Prev_State, W2
    cp      W1, W2
    bra     NZ, prev_stat_not_0
    mov     #Entry, W1
    mov     #psvoffset(lookup), W2
    mov.b   [W2+W0], W0
    mov.b   W0, [W1+0]
prev_stat_not_0:
    mov     #LCD_line1, W0
    mov.b   #' ', W1
    mov.b   #'*', W2
    mov.b   W2, [W0+12]
    mov.b   W1, [W0+13]
    mov.b   W1, [W0+14]
    mov.b   W1, [W0+15]
    pop     W2
    pop     W1
    pop     W0
    return

fn2:
    push    W0
    push    W1
    push    W2
    mov     #1, W1
    mov     Prev_State, W2
```

```
    cp      W1, W2
    bra     NZ, prev_stat_not_1
    mov     #Entry, W1
    mov     #psvoffset(lookup), W2
    mov.b   [W2+W0], W0
    mov.b   W0, [W1+1]
prev_stat_not_1:
    mov     #LCD_line1, W0
    mov.b   #' ', W1
    mov.b   #'*', W2
    mov.b   W2, [W0+12]
    mov.b   W2, [W0+13]
    mov.b   W1, [W0+14]
    mov.b   W1, [W0+15]
    pop     W2
    pop     W1
    pop     W0
    return

fn3:
    push    W0
    push    W1
    push    W2
    mov     #2, W1
    mov     Prev_State, W2
    cp      W1, W2
    bra     NZ, prev_stat_not_2
    mov     #Entry, W1
    mov     #psvoffset(lookup), W2
    mov.b   [W2+W0], W0
    mov.b   W0, [W1+2]
prev_stat_not_2:
    mov     #LCD_line1, W0
    mov.b   #' ', W1
    mov.b   #'*', W2
    mov.b   W2, [W0+12]
    mov.b   W2, [W0+13]
    mov.b   W2, [W0+14]
    mov.b   W1, [W0+15]
    pop     W2
    pop     W1
    pop     W0
    return

fn4:
    push    W0
    push    W1
    push    W2
    mov     #3, W1
    mov     Prev_State, W2
    cp      W1, W2
    bra     NZ, prev_stat_not_3
    mov     #Entry, W1
    mov     #psvoffset(lookup), W2
    mov.b   [W2+W0], W0
    mov.b   W0, [W1+3]
prev_stat_not_3:
    mov     #LCD_line1, W0
    mov.b   #'*', W2
    mov.b   W2, [W0+12]
    mov.b   W2, [W0+13]
    mov.b   W2, [W0+14]
    mov.b   W2, [W0+15]
    pop     W2
    pop     W1
    pop     W0
    return

fn5:
    push    W0
    push    W1
    push    W2
    push    W3
    mov     #1, W0
    mov     W0, Time_Inc
    mov     #psvoffset(password), W1
```

```
    mov     #Entry, W0
    mov.b   [W1], W3
    mov.b   [W0], W2
    cp.b    W3, W2
    bra     NZ, wrong_password
    mov.b   [W1+1], W3
    mov.b   [W0+1], W2
    cp.b    W3, W2
    bra     NZ, wrong_password
    mov.b   [W1+2], W3
    mov.b   [W0+2], W2
    cp.b    W3, W2
    bra     NZ, wrong_password
    mov.b   [W1+3], W3
    mov.b   [W0+3], W2
    cp.b    W3, W2
    bra     NZ, wrong_password
    bset    PORTF, #0
    mov     #LCD_line2, W0
    mov     #'O', W1
    mov.b   W1, [W0+14]
    mov     #'K', W1
    mov.b   W1, [W0+15]
    bra     correct_password
wrong_password:
    mov     #LCD_line2, W0
    mov     #'X', W1
    mov.b   W1, [W0+15]
correct_password:
    pop     W3
    pop     W2
    pop     W1
    pop     W0
    return



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

__CNInterrupt:
    push    W0
    push    W1

    bclr    IFS1, #CNIF
    mov     PORTD, W0

    btss    W0, #6
    bra     key_released
    mov     #0x000F, W1
    and     W0, W1, W0
    mov     W0, KPD_buff
    mov     #1, W0
    mov     W0, KPD_stat
    btg     PORTF, #3
key_released:



    pop     W1
    pop     W0
    retfie

__T2Interrupt:
    push    W0
    push    W1
    bclr    IFS0, #T2IF     ; clear timer2 interrupt status flag
    btg     PORTF, #1
    mov     Time_Inc, W1
    mov     Time_Count, W0
    add     W0, W1, W0
    mov     W0, Time_Count
    mov     #4, W1
    cp      W0, W1
    bra     NZ, no_time_limit
```

```
    mov     #0, W1
    mov     W1, Time_Inc
no_time_limit:
    pop     W1
    pop     W0
    retfie

init_timer2:
    bclr    T2CON, #TON     ; turn timer2 OFF

    bset    T2CON, #TCKPS1
    bclr    T2CON, #TCKPS0  ; set prescaler to 64

    bclr    T1CON, #TCS     ; select internal clock

    mov     #0x0000, W0
    mov     W0, TMR2        ; clear TMR2 register
    mov     #31250, W0
    mov     W0, PR2         ; set timer2 period to 32150 -> f=2e6/64/31250=1 Hz

    bclr    IPC1, #T2IP2
    bclr    IPC1, #T2IP1
    bset    IPC1, #T2IP0    ; set timer2 priority to 001
    bclr    IFS0, #T2IF     ; clear timer2 interrupt status flag
    bset    IEC0, #T2IE     ; enable timer1 interrupts

    bset    T2CON, #TON     ; turn timer2 ON
    return


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

    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, KPD_stat

    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.

    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

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
```