Les GPIO du PIC16F877

cours n°4
LI326

Plan

- Les GPIO du pic
- Lecture de la documentation
- Usages
  - Commande de LEDs
  - Afficheur 7-segments
  - Bouton poussoir
  - Clavier matriciel
  - Clavier 1 fil

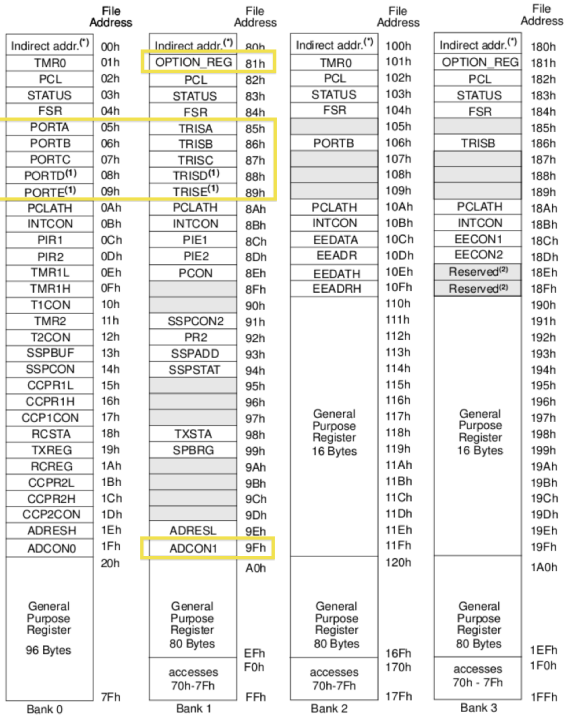
GPIO: General Purpose Input/Output

Selon Wikipedia:

- Les ports GPIO (General Purpose Input/Output, c'est-à-dire entrée/sortie pour un usage général) sont des ports d'entrée/sortie très utilisés dans le monde des microcontrôleurs, en particulier dans le domaine de l'électronique embarquée. Les périphériques GPIO comportent un ensemble de ports d'entrée/sortie qui peuvent être configurés pour jouer soit le rôle d'une entrée, soit le rôle d'une sortie.

- Lorsqu'un port GPIO est configuré en tant que sortie, on peut écrire dans un registre interne afin de modifier l'état d'une sortie. Lorsqu'il est configuré en tant qu'entrée, on peut détecter son état en lisant le contenu d'un registre interne.

- De plus, les périphériques GPIO peuvent produire des interruptions et des événements d'accès direct à la mémoire.



Registres impliqués

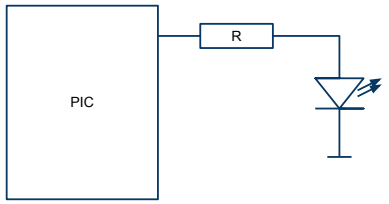
- 5 ports: A,B,C,D,E
- Tous les ports ne sont pas égaux.
- Certains ports peuvent lever des interruptions.

La documentation officielle

Pour chaque port:

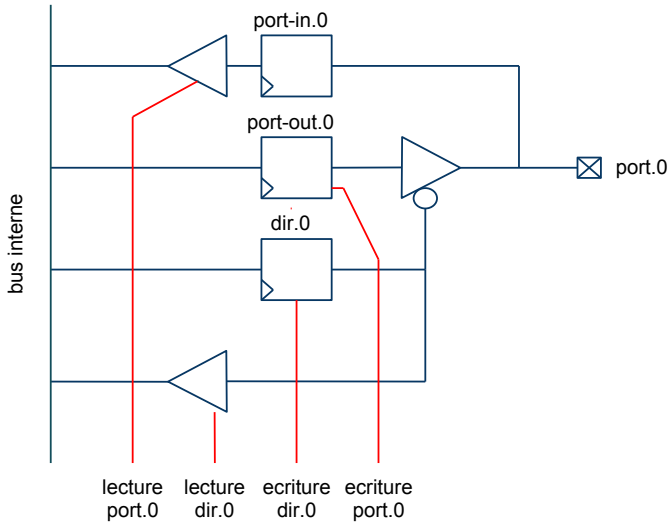
- Une description de la fonction
- Un schéma représentatif du comportement
- La liste des registres impliqués
  - La définition des registres spéciaux généraux se trouve aussi dans le chapitre sur la mémoire

Affichage par LEDs



- 1 sur la broche allume la LED
- 0 éteint la LED
- La résistance permet de limiter le courant (entre 2 et 20mA)

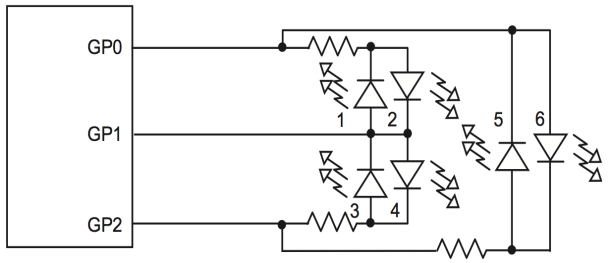
GPIO exemple du port D



Commandes 6 LEDs

Les GPIOs peuvent avoir 3 états : 0, 1 et Z
Ici, on peut allumer 1 led parmi les 6.
En généralisant,

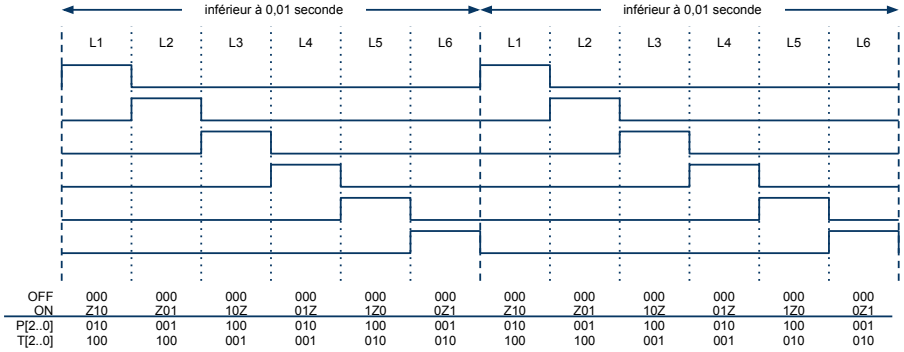
- avec GP ports, on peut commander GP*(GP-1) LEDs
- 2 LEDs avec 2 GPs, 6 avec 3, 12 avec 4, 20 avec 5, ...



GPx			LEDs					
0	1	2	1	2	3	4	5	6
0	0	0	0	0	0	0	0	0
0	1	Z	1	0	0	0	0	0
1	0	Z	0	1	0	0	0	0
Z	0	1	0	0	1	0	0	0
Z	1	0	0	0	0	1	0	0
0	Z	1	0	0	0	0	1	0
1	Z	0	0	0	0	0	0	1

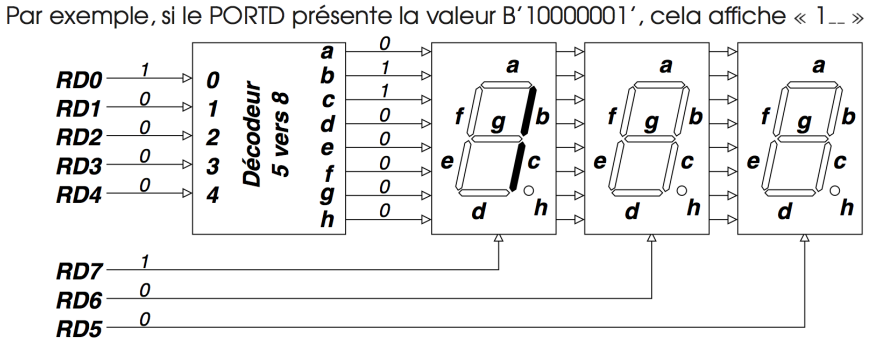
Multiplexage temporel

- Les LEDs s'allument et s'éteignent instantanément.
- L'œil humain ne distingue pas les clignotements > à 100Hz.
- Si on allume les LEDs à tour de rôle à une fréquence supérieure à 100Hz on peut "voir" plusieurs leds allumées.
- avec 6 leds:



Afficheurs 7 segments

commande de 3 afficheurs 7-segments (branchés sur le port D) grâce au multiplexage temporel.



Algorithme d'affichage

soit

- Vaff une valeur sur 6 bits à afficher

- Baff le numéro du bit à afficher (sera compris entre 0 et 5)
- Paff un tableau de 6 cases avec les codes à mettre sur les Ports pour afficher 1
  Paff [6] = {010,001,100,010,100,001};
- Taff un tableau de 6 cases avec les codes à mettre sur les Dir (Tris) pour afficher 1
  Taff [6] = {100,100,001,001,010,010};
- P La valeur du port
- T La direction du port

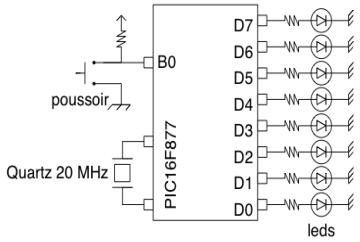
algo:

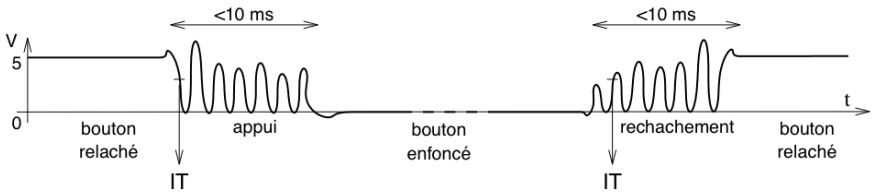
```
faire périodiquement (p. ex. 10ms)
    P = 0; T = 0;
    si ((Vaff >> Baff) & 1) alors
        T = Taff [ Baff ];
        P = Paff [ Baff ];
    finsi
    Baff = ( Baff + 1 ) % 6;
finfaire
```

Le bouton poussoir

Une entrée sur 1 bit numérique:

- relaché : 1
- appuyé : 0



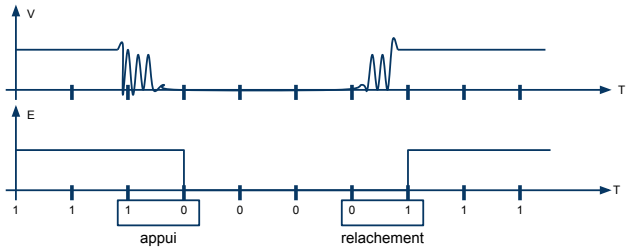


Traitement des rebonds

Il suffit d'échantillonner à une période supérieure aux rebonds, mais inférieure à la durée d'appui

soit
- B : une variable
- E : la valeur du bouton poussoir
- appui : un drapeau à 1 si appui
- relache : un drapeau à 1 si relachement

faire périodiquement
    B = (B << 1) & E
    si (B & 3) == 2 **alors**
        appui = 1
    finsi
    si (B & 3) == 1 **alors**
        relache == 1
    finsi
finfaire

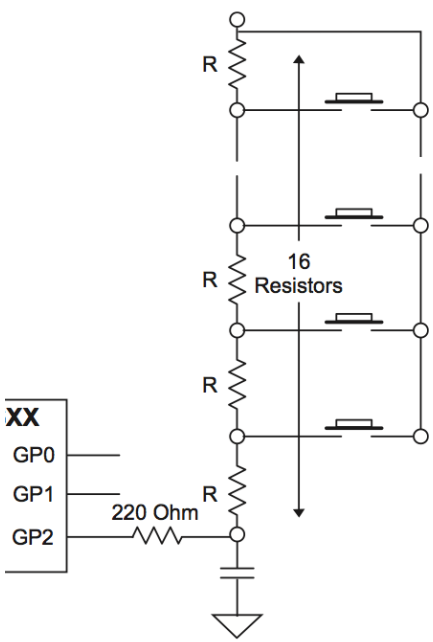


clavier 1 fil

- La charge d'une capacité est "proportionnelle" à la résistance de charge : si on double la resistance on double le temps de charge

- Algorithme
    GP2 = 0
    compteur = 0
    GP2 est mis en entrée
    tantque GP2 != 1
        compteur++
    fintantque
    touche = fct(compteur)

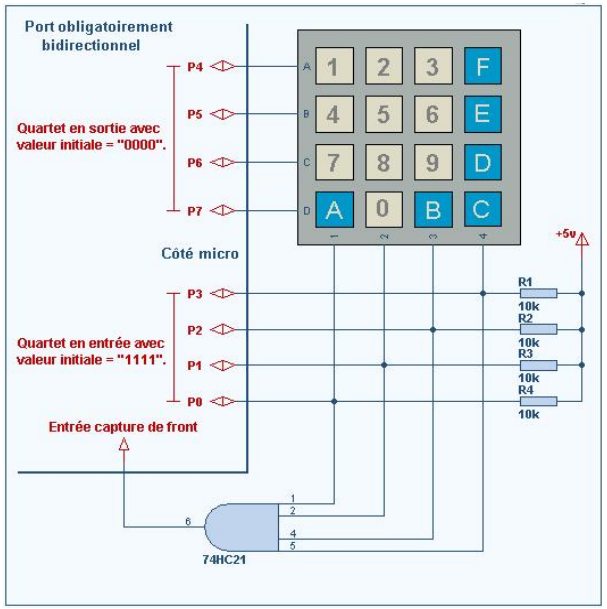
    garder les 4 bits de poids forts du compteur



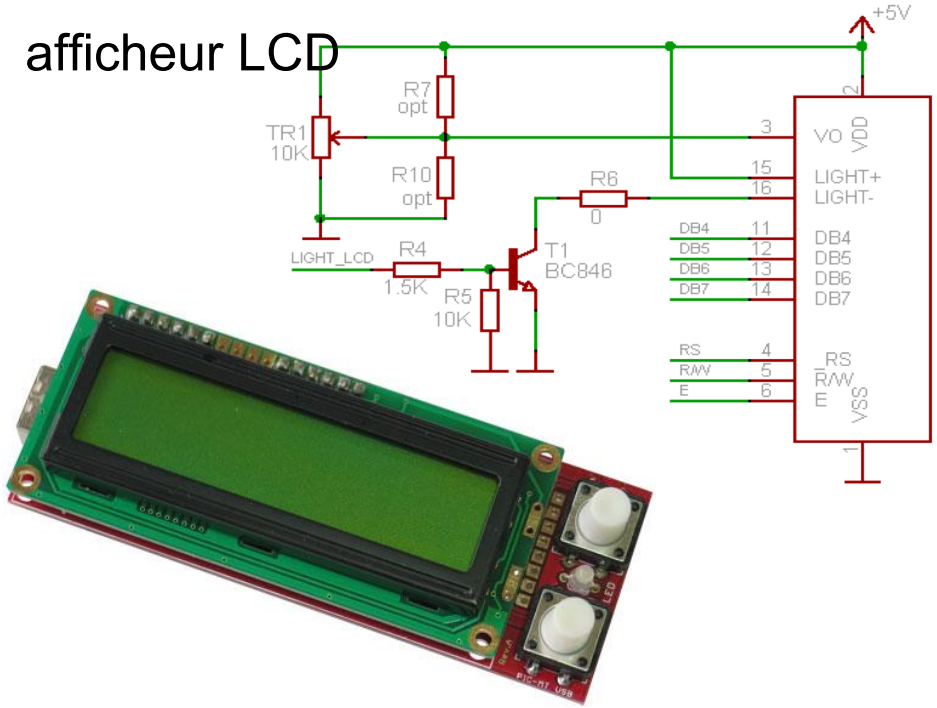
Clavier matriciel

P4 à P7 à 0 permet de détecter un appui

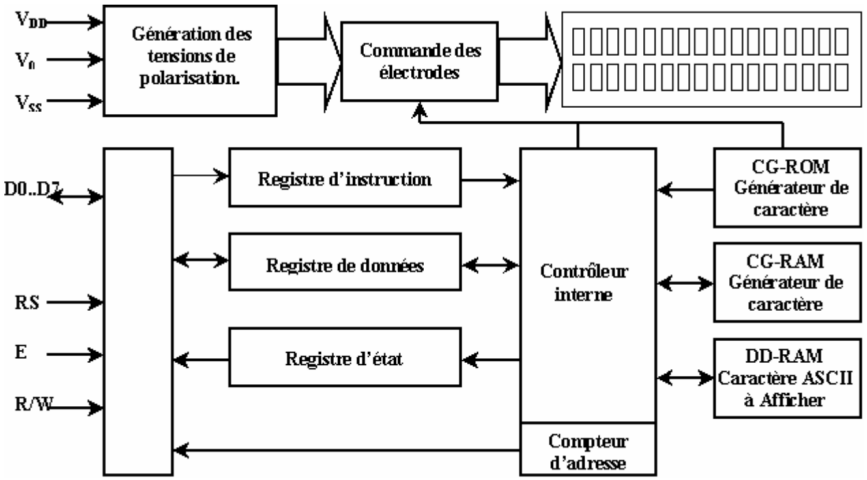
puis on place une des lignes à 0 les autres en entrée et on lit P0 à P3



afficheur LCD

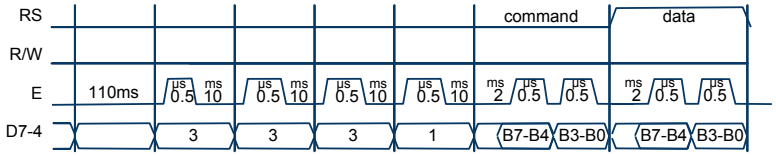


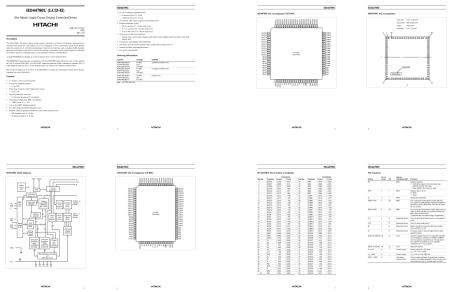
afficheur LCD



Chronogramme LCD

Cette information est extraite de la documentation





afficheur LCD

broche N°	Symbole	Fonction
1	VSS	0V
2	VDD	5V
3	V0	Réglage contraste[1]
4	RS	0 : Code instruction 1 : Donnee
5	R/W	1 : Lecture 0 : Ecriture
6	E	Horloge
7	D0	
8	D1	
9	D2	
10	D3	Bus de donnée Bidirectionnel[2]
11	D4	
12	D5	
13	D6	
14	D7	

1 : le réglage de contraste s'effectue en appliquant une tension variable sur cette broche.

2 : Lors de l'utilisation en mode 4 bits seul les bits D4 à D7 sont utilisés. Les lignes correspondant de D0 à D4 sont laissées en l'air. On communique le quarté de poids fort puis de poids faible.

lcd.inc

```
#define LCD_DIR             TRISD
#define LCD_PORT            PORTD
#define LCD_BL              LCD_PORT,3
#define LCD_E               LCD_PORT,2
#define LCD_RW              LCD_PORT,1
#define LCD_RS              LCD_PORT,0
#define LCD_CLR_DISP        0x01
#define LCD_DISP_ON         0x0C
#define LCD_DISP_OFF        0x08
#define LCD_CUR_HOME        0x02
#define LCD_CUR_OFF         0x0C
#define LCD_CUR_ON_UNDER    0x0E
#define LCD_CUR_ON_BLINK    0x0F
#define LCD_CUR_LEFT        0x10
#define LCD_CUR_RIGHT       0x14
#define LCD_CUR_UP          0x80
#define LCD_CUR_DOWN        0xC0
#define LCD_ENTER           0xC0
#define LCD_DD_RAM_ADDR     0x80
#define LCD_DD_RAM_ADDR2    0xC0
```

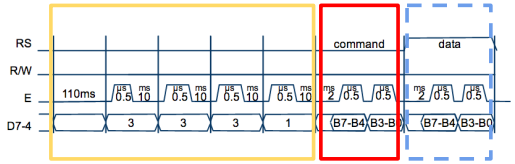
```
lcd_init macro
          call   _lcd_init
        endm
```

```
lcd_send_lcmd macro  cmd
          movlw  cmd
          bcf    LCD_RS
          call   _lcd_send
        endm
```

```
lcd_send_wdata macro
          bsf    LCD_RS
          call   _lcd_send
        endm

lcd_send_ldata macro  char
          movlw  char
          bsf    LCD_RS
          call   _lcd_send
        endm

lcd_send_fdata macro  reg
          movf   reg, w
          bsf    LCD_RS
          call   _lcd_send
        endm
```



lcd_main.asm

```
; ------------------------------------------------------------------------------
; programme : affichage LCD
; ------------------------------------------------------------------------------

        list       p=16f877      ; definit le processeur cible
        errorlevel -302          ; pas de message concernant les numéro de bank
        list       n=0           ; pour un fichier lst soit sur une seule page
        include    "p16f877.inc" ; declaration des noms de registres

        ; Definition du registre de configuration du PIC
        ; _CP_OFF   : le code n'est pas protege et peut etre relu
        ; _WDT_OFF  : pas de timer watch dog
        ; _PWRTE_ON : attente d'un délai apres le power on
        ; _HS_OSC   : oscillateur à quartz
        ; _LVP_OFF  : pas de mode programmation basse tension
        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC & _LVP_OFF

; réservation des registres
; ------------------------------------------------------------------------------
        CBLOCK  0x70             ; banc commun
           A0, A1, A2, A3, M0    ; registres de travail pour les programmes
        ENDC

       include "lcd.inc"

; reset + gestionnaire d'interruption
; ------------------------------------------------------------------------------
        org     0
        pagesel initialisation  ; pour etre compatible avec le bootloadder
        call    initialisation
        goto    main

        org     4               ; adresse du vecteur d'interruption
        return                  ; ne rien faire : effacer le bit GIE et sortir

; initialisation generale
; ------------------------------------------------------------------------------
initialisation
        lcd_init
        clrf    STATUS          ; pour être sûr d'être dans le banc 0
        return

; programme principal
; ------------------------------------------------------------------------------
main
        lcd_send_ldata  'L'
        lcd_send_ldata  'O'
        lcd_send_ldata  'L'
        goto $

       include "lcd.asm"

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

lcd.asm 1/3

```
 ; _lcd_wait permet d'attendre un multiple de 2ms
 ;-------------------------------------------------------------------------------------------------------------
_lcd_wait macro  delai
             movlw      delai/2
             call       _lcd_wait_ms
         endm

_lcd_wait_2ms
           clrf         _lcd_tmp
           call         $+5
           call         $+4
           call         $+3
           decfsz       _lcd_tmp
           goto         $-4
           call         $+2
           call         $+1
           return

_lcd_wait_ms
           call         _lcd_wait_2ms
           addlw        -1
           btfss        STATUS,Z
           goto         $-3
           return
```

lcd.asm 2/3

```
_lcd_e_pulse
        bsf             LCD_E
        goto            $+1
        goto            $+1
        bcf             LCD_E
        return

_lcd_init        ; sequence d'initialisation du lcd
        BANKSEL         LCD_PORT        ; initialiser le LCD_PORT
        clrf            LCD_PORT
        BANKSEL         LCD_DIR         ; place tous les signaux du LCD_PORT en sortie
        clrf            LCD_DIR
        clrf            STATUS          ; pour être sûr d'être dans le banc 0
        bcf             LCD_RS          ; RS=0
        bcf             LCD_RW          ; RW=0
        _lcd_wait       110             ; 110ms
        movlw           B'00110000'     ; LCD_PORT=0b00110000;
        movwf           LCD_PORT
        call            _lcd_e_pulse    ; E_Pulse();
        _lcd_wait       10              ; 10ms
        call            _lcd_e_pulse    ; E_Pulse();
        _lcd_wait       10              ; 10ms
        call            _lcd_e_pulse    ; E_Pulse();
        _lcd_wait       10              ; 10ms
        movlw           B'00100000'     ; LCD_PORT=0b00100000;
        movwf           LCD_PORT
        call      _     lcd_e_pulse     ; E_Pulse();
        lcd_send_lcmd   LCD_DISP_ON
        lcd_send_lcmd   LCD_CLR_DISP
        return
```

lcd.asm 3/3

```
#define _lcd_data A0
#define _lcd_tmp  A1

_lcd_send        ; envoi w
        movwf           _lcd_data
        bcf             LCD_RW          ; write
        _lcd_wait       2               ; 2ms
        movf            _lcd_data,w     ; LCD_PORT = (LCD_PORT & 0b00001111)|(lcd_data & 0b11110000)
        call            _lcd_send_half
        swapf           _lcd_data,w     ; LCD_PORT = (LCD_PORT & 0b00001111)|((lcd_data<<4) & 0b11110000)
        call            _lcd_send_half
        return

_lcd_send_half       ; envoi lcd_data
        andlw           B'11110000'
        movwf           _lcd_tmp
        movf            LCD_PORT,w
        andlw           B'00001111'
        iorwf           _lcd_tmp,w
        movwf           LCD_PORT
        goto            _lcd_e_pulse    ; E_Pulse();
```