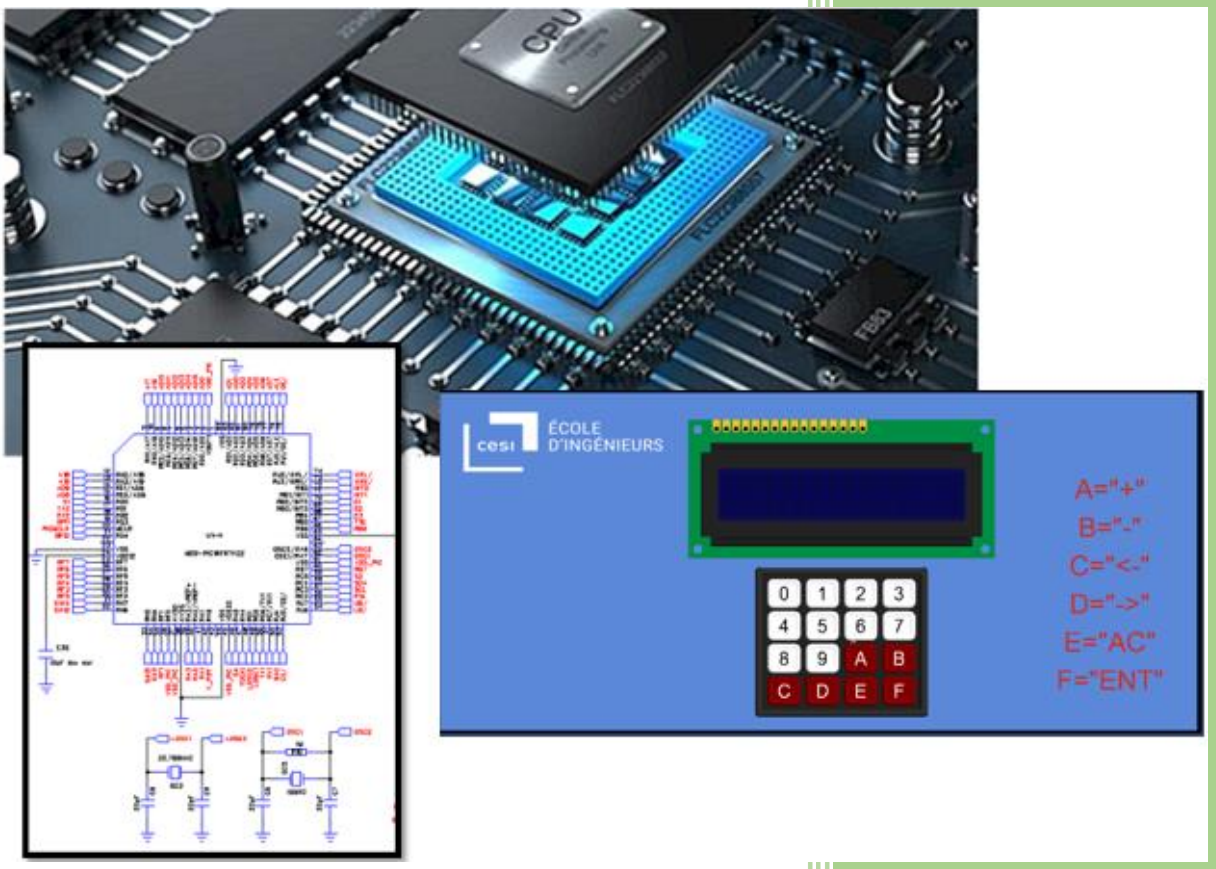
2022

TP2 FISA 26 Microcontrôleur



ANTINI Nicolas

CESI

01/01/2022

Mise en œuvre des mémoires « et gestion des Interruptions

Objectifs du TP :

Découvrir l'environnement de développement.

Se familiariser avec la mémoire du PIC 18f87k22 et sa gestion des interruptions.

Matériel utilisé :

- Malette MDET,
- Un PC équipé des logiciels MPLABX et son compilateur XC8.
- Un analyseur logique.
- Documentation des logiciels et du microcontrôleur.

Points abordés :

- La gestion de la mémoire et des interruptions pour le PIC18.

Objectifs du TP :

- Gestion d'un clavier matricé
- Gestion d'un afficheur LCD et création d'un jeu de fonctions répertoriées dans un fichier LCD.C.
- Gestion des interruptions du PIC18F87K22 (Haute et basse)
- Cartographier la mémoire du PIC18F87K22 (mémoire données, programmes, externe, EEROM).

Etape 1 :

Configuration du bus externe

Nous utilisons le bus mémoire externe, donc il faut le configurer :
Run > Set project configuration > Customize

A gauche dans l'arborescence, étendre conf en cliquant dessus puis XC8

Cliquer sur XC8 linker

A droite sélectionner memory model dans option categories

A droite de RAM range, taper : default,20000-1FFFFF

c'est la zone mémoire adressable par le PIC

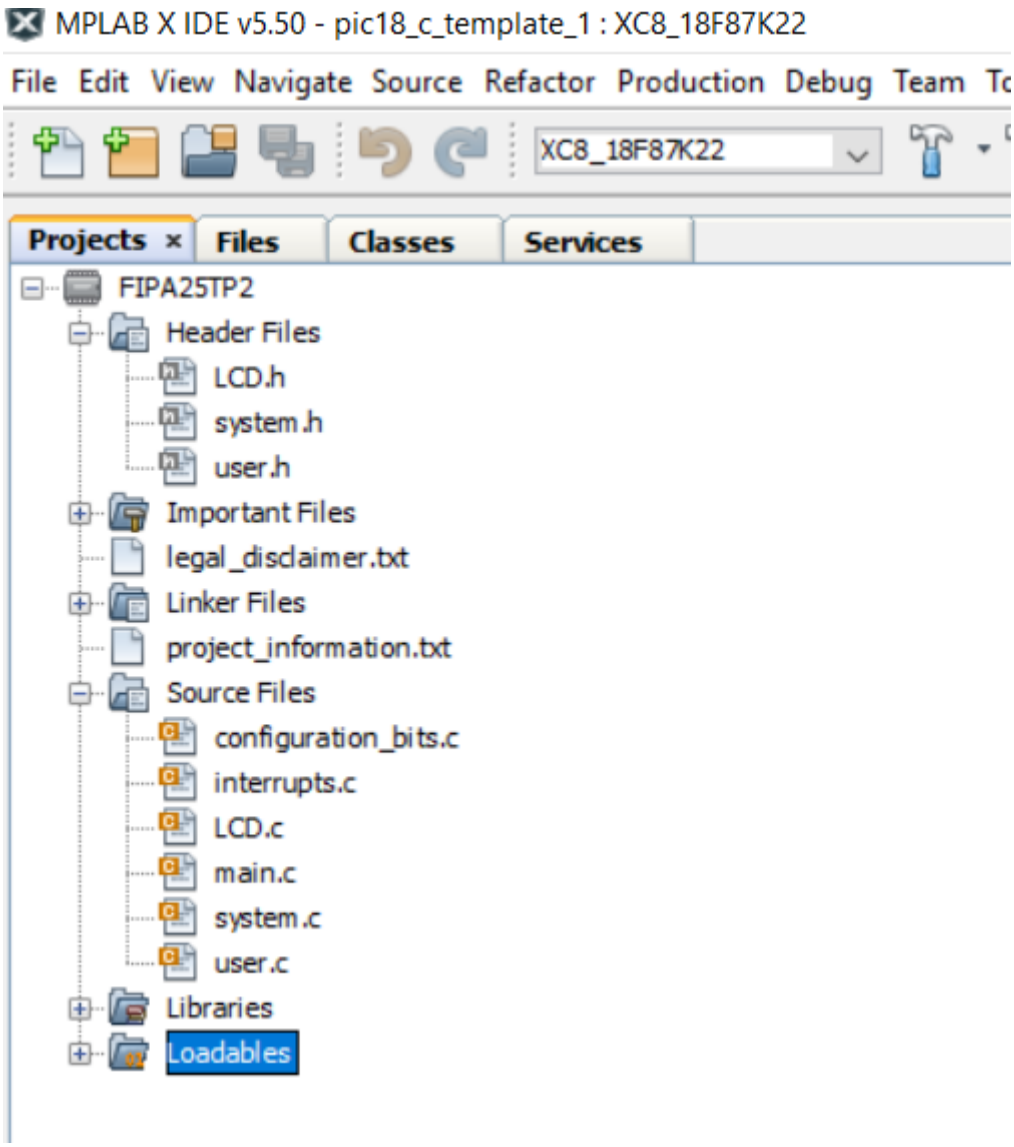
Valider avec OK

Utiliser impérativement la version1 XC8 1.12

Recopier picc_18.ini dans le répertoire DAT :

Recopier le fichier PICC-18.INI dans le répertoire suivant :
C : \program files(x86)\Microchip\xc8\v1.20\dat
Afin de prendre en compte la zone externe de l'espace mémoire du PIC18F87K22.

Etape 2) Créer le projet FIPA25TP2.



user.h regroupe les entêtes de mes fonctions.

Ex : void Init(void) ;

system.h regroupe les entêtes des fonctions spécifiques au système.

LCD .h regroupe les entêtes des fonctions spécifiques a la gestion de l'afficheur LCD.

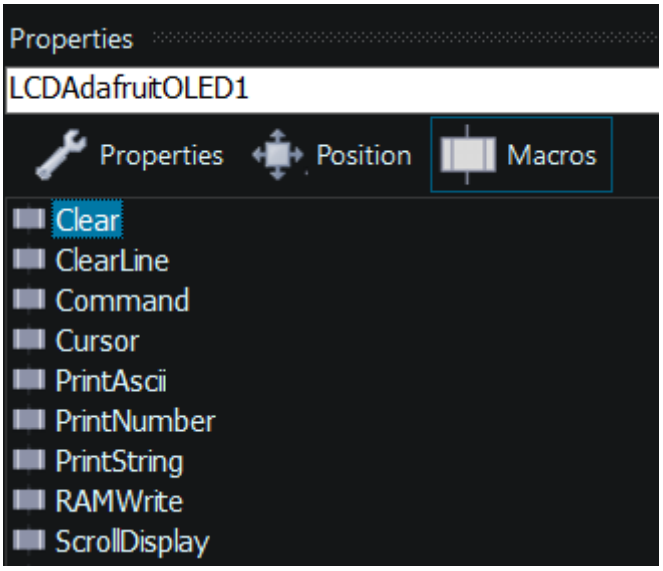
Source Files :

Configuration_bits.c représente le fichier de configuration du système

Interrupt.c regroupes les fonctions liées à la gestion des ISR

LCD.c fonctions liées à la gestion de l'afficheur LCD

Equivalence LCDAdafruitOLED1



MainDemoFIPA25.c

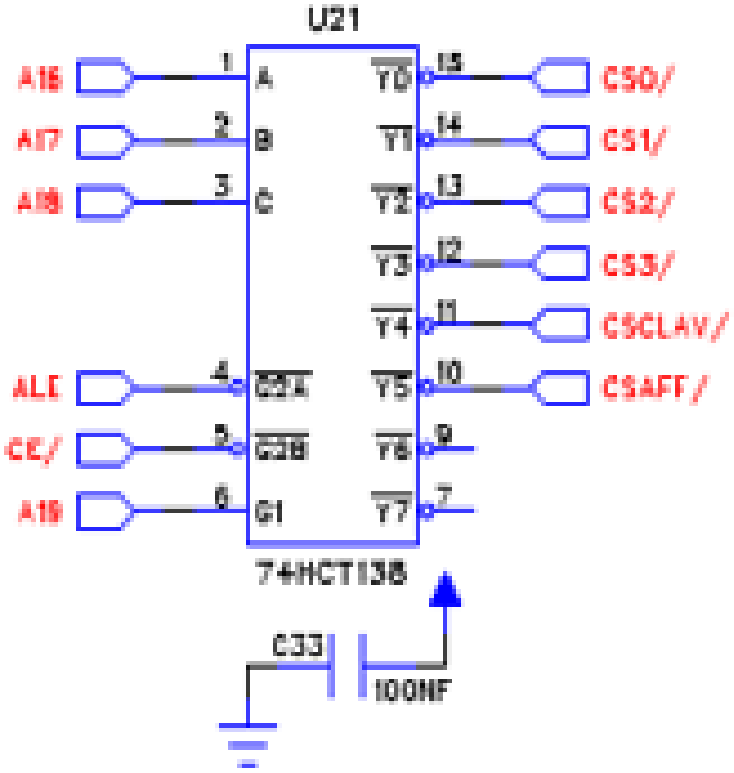
C'est la fonction Main qui comportera l'ordonnanceur séquentiel mono tâche.

```
#include <xc.h>

#include "user.h"

#include "LCD.h"



void main(void) {
   Init();
   while (1)
        {
             /*Rafraichir l'afficheur LCD
             Ecrire EEROM octets 0,9
             Lire EEROM octets 0 à 9*/
        }
   return;
}
```

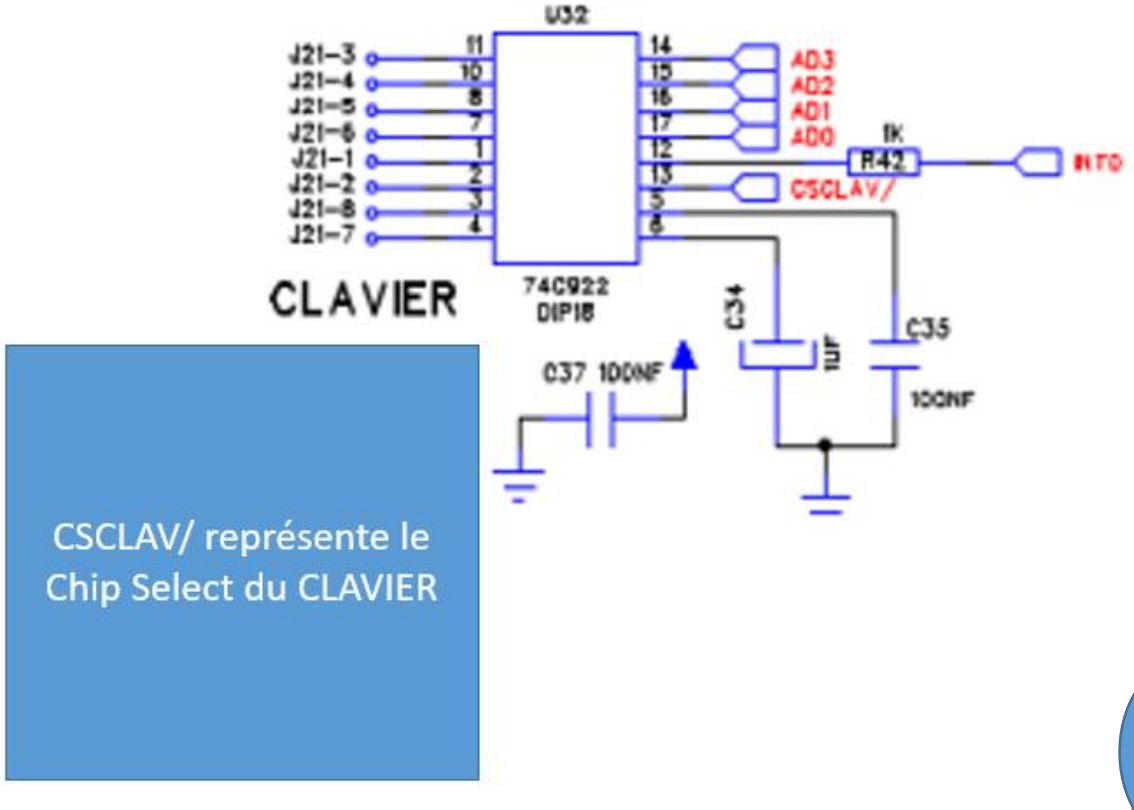
ETAPE 3) Décodage d'adresse

Selon Van NEWMAN .

Exploitation de la DATASHEET associeé au décodeur d'adresse (Démultiplexeur 74138)


U21
A16
A17
A18
ALE
CE/
A19
1
2
3
4
5
6
A
B
C
G2A
G2B
G1
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
15
14
13
12
11
10
9
7
CS0/
CS1/
CS2/
CS3/
CSCLAV/
CSAFF/
74HCT138
C33
100NF

Clavier :



Bit b à 0

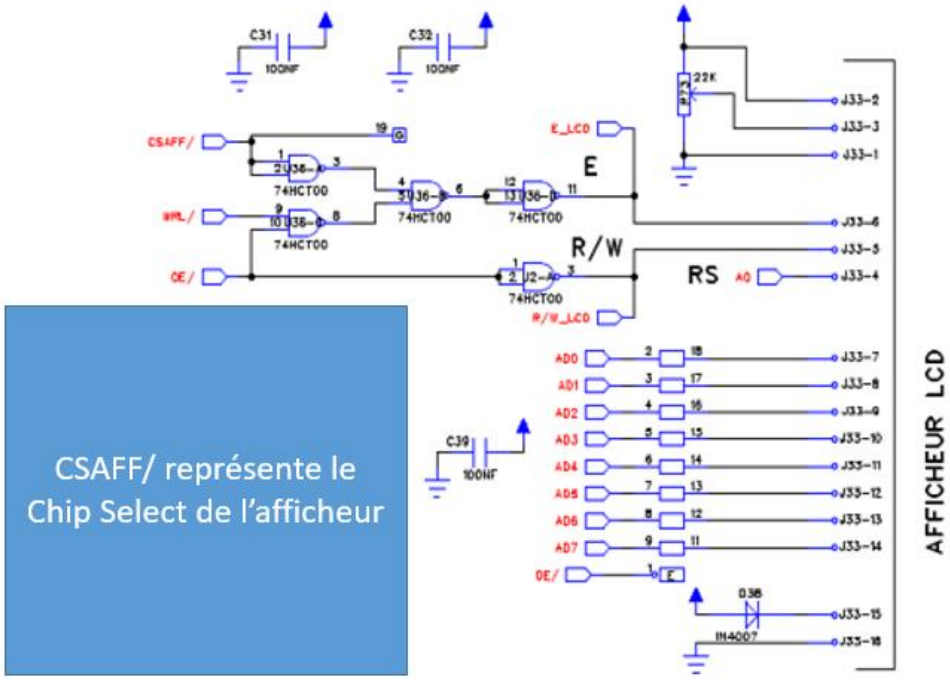
Calculer l'adresse du clavier. L'adresse se présentera sous la forme (21 bits)

A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 0

Déclaration d'une variable CLAVIER de type :

far unsigned char CLAVIER @ 0X_ _ _ _ _ _

Afficheur 2*16



Bit b à 0

Calculer les 2 adresses de l'afficheur LCD . Les adresses se présenteront sous la forme (21 bits)

A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 0

Déclaration d'une variable AFFICHEUR_WR_DATA de type :

far unsigned char AFFICHEUR _DATA @ 0X_ _ _ _ _ _

Déclaration d'une variable AFFICHEUR_FUNC de type :

far unsigned char AFFICHEUR _DATA @ 0X_ _ _ _ _ _

ETAPE 4) Gestion du Clavier

L'appui d'une touche de clavier déclenche un signal sur la sortie RBO (INT0).

4.1) Ecrire une application **get_clav.c** :

- Lecture d'une touche,

- Conversion du code lu en un code exploitable par l'afficheur LCD (Code ASCII pour les valeurs comprises entre 0 et 9).

-Mémorisation de la touche lue.

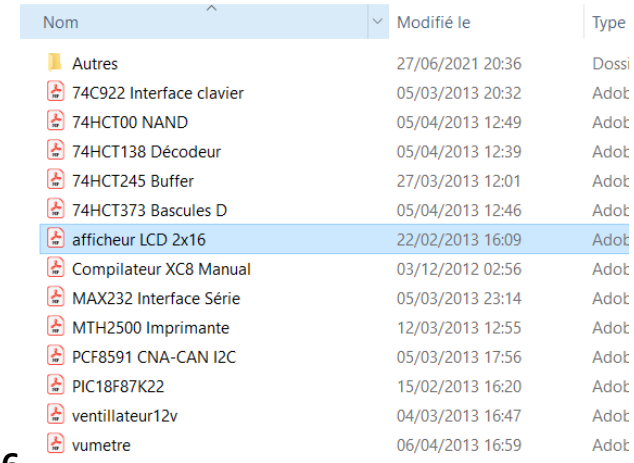
4.2) Gérez le clavier en interruption sachant qu'INT0 est une entrée d' interruption normalisée de notre microcontrôleur.

Déclaration d' une fonction de type :

void hight_priority interrupt NOMFONCTION(void)

ETAPE 4) Gestion de l'afficheur

Créer un répertoire de fonction LCD.C selon la datasheet du composant



afficheur LCD 2*16

Etape 5) Gestion des touches du clavier et affichage LCD :

En cas d'appui d'une touche au Clavier on affiche la valeur sur l'afficheur à la position du curseur.

+,- sans effet

<- et -> déplacement du curseur à gauche et à droite pour modification du caractère saisi.

ENT Enter

AC efface l'écran, positionne le curseur en haut à gauche.

Etape 5) Gestion de l'EEROM

On ne travaille que sur les 10 premiers octets de la mémoire EEROM.

Exemple :

Write : 05=20 ENT *écrire 20 en hexadécimal à l'emplacement 05 de l'EEROM.*

0	6	=	2	0	ENT										
2	0														

Après une écriture on effectue une lecture de vérification

Read : 06 ENT Lire le contenu de l'adresse 06 (résultat 01)

0	6	ENT													
0	1														

Seules les interruptions Clavier et TIMER0 sont en priorité haute.

Ici lecture / Ecriture EEROM sont gérées en priorité Basse

Déclaration d' une fonction de type :

void low_priority interrupt NOMFONCTION(void)

ETAPE 6) Cartographie mémoire

Donnez la cartographie de votre système taux d'occupation, espace mémoire occupée , Ram, Flash, Eerom .

Annexes :

1) Afficheur 2*16 :

Séquence d'init recommandée :

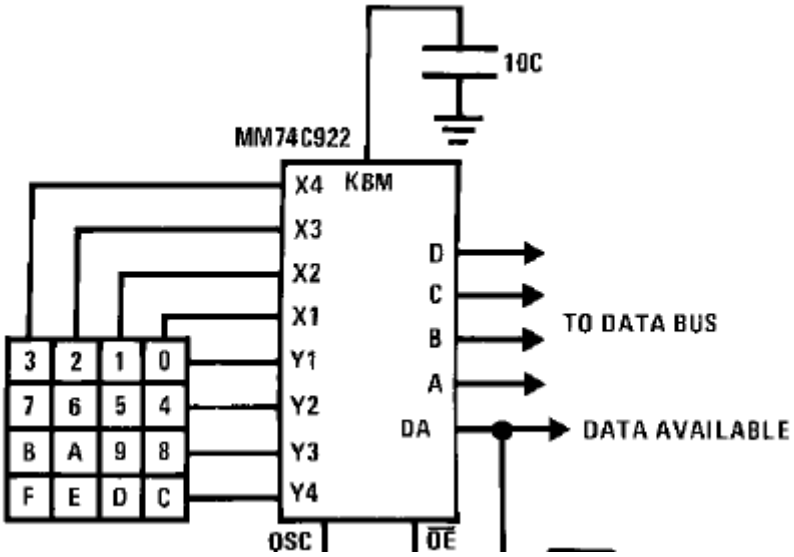
```
//Définitions des fonctions LCD
#define DISPLAY_CLEAR            0x01
#define RETURN_HOME              0x02
#define ENTRY_MODE_SET_CI_DNS    0x06 //Cursor increase, is not shifted
#define ENTRY_MODE_SET_CI_DS     0x07 //Cursor increase, shifted
#define DISPLAY_ON_CUR_ON_BLINK_ON  0x0F
#define DISPLAY_ON_CUR_ON_BLINKOFF  0x0E
#define SHIFT_DISPLAY_RIGHT      0x1C
#define SHIFT_DISPLAY_LEFT       0x18
#define SHIFT_CURSOR_RIGHT       0x14
#define SHIFT_CURSOR_LEFT        0x10
#define SET_FUNC_8BIT_2LINE_5x10   0x3C
#define SET_FUNC_8BIT_2LINE_5x7    0x38

void init_aff_lcd(void)
{
   delai_ms(100);
   LCD_FUNC=0x38;                  delai_ms(5);
   LCD_FUNC=0x38;                  delai_ms(1);
   LCD_FUNC=0x38;                  delai_ms(1);
   LCD_FUNC=SET_FUNC_8BIT_2LINE_5x7;     delai_ms(1);
   LCD_FUNC=SHIFT_CURSOR_RIGHT;          delai_ms(1);
   LCD_FUNC=DISPLAY_ON_CUR_ON_BLINKOFF;   delai_ms(1);
   LCD_FUNC=ENTRY_MODE_SET_CI_DNS;       delai_ms(1);
   LCD_FUNC=RETURN_HOME;             delai_ms(2);
   LCD_FUNC=DISPLAY_CLEAR;           delai_ms(2);
```

}

// delai_ms () est une fonction de temporisation qui utilise **__delay_ms() voir TP1.**

3) Niveau d'IT

Selon exemple Ci-dessous on imagine un bouton poussoir ou la ligne DA reliée à INT0 :



1 seul niveau d'IT

1- Un seul niveau d'IT
RCONbits.IPEN=0;
2- validation IT globales:
INTCONbits.GIE=1
3- validation IT périphériques
INTCONbits.PEIE= 1 ou 0
4- validation individuelle des IT
Validation du périphérique xxx: **xxxxIE=1**
5- acquittement des IT
Si IT en suspens l'acquitter : **xxxIF=0**

```
// exemple bouton poussoir sur RBO en IT
RCONbits.IPEN=0; // 1 seul nv
INTCONbits.GIE=1; // IT autorisé
INTCONbits.PEIE=0; // pas de périph en IT
INTCONbits.INT0IE=1; // IT bouton poussoir autorisé
INTCON2bits.INTEDG0=0; // choix du front
INTCONbits.INT0IF=0; // acquittement init
```

2 niveaux d'IT

1- Deux seul niveau d'IT
RCONbits.IPEN=1;
2- validation IT hautes:
INTCONbits.GIEH=1 ***(ou INTCONbits.GIE=1)***
3- validation IT périphériques
INTCONbits.GIEL=1 ***(ou INTCONbits.PEIE= 1)***
4 – Pour chaque source d'IT choisir si priorité haute ou basse
xxxxIP=……; //1= haute priorité 0=basse

5- validation individuelle des IOT
Validation du périphérique xxx: **xxxxIE=1**
6- acquittement des IT
Si IT en suspens l'acquitter : **xxxIF=0**