François SCHNEIDER – Lycée Victor-Hugo BESANÇON.

STS SE

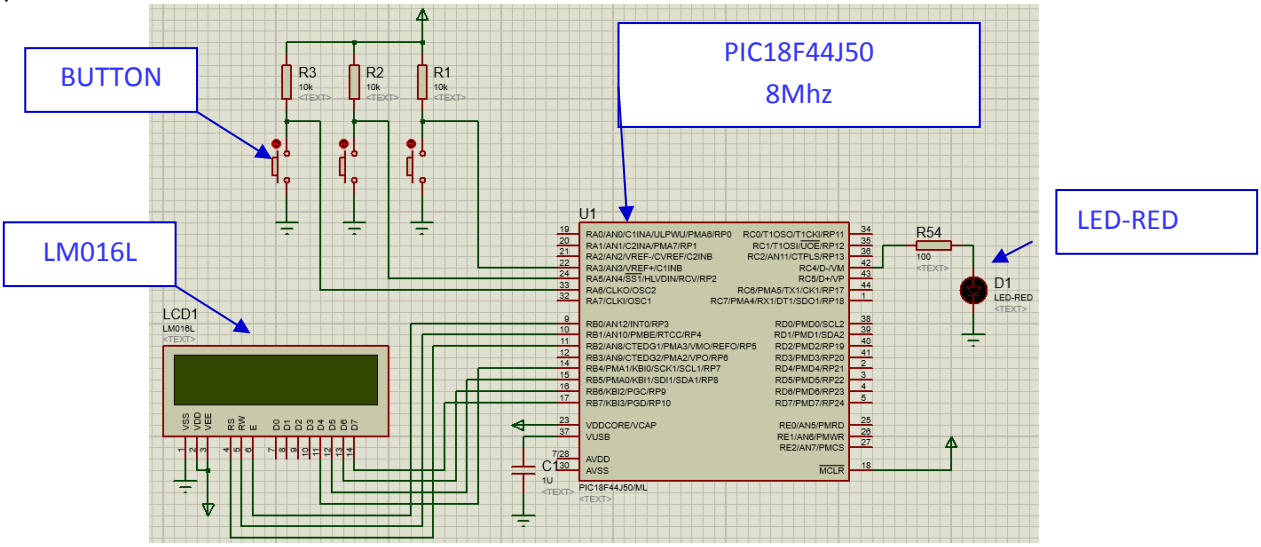
Développement de microcontrôleurs Microchip avec PICC
validation fonctionnelle PROTEUS

Pilotage d'un afficheur
Utilisation Wizard et PROTEUS
Simulation Validation

Prérequis : langage C, PROTEUS ISIS simulation d'un microprocesseur.

I. Le projet.

Le schéma du montage est donné page suivante avec le nom des composants en bleu (librairie ISIS). Vous dessinez le schéma, vous le sauvegardez avec pour nom « Tp2.dsn » dans un dossier nommé Tp2.



Le fonctionnement est le suivant :

- Un appui sur le bouton B1, la LED éclaire et l'afficheur affiche 'B1'.
- Un appui sur le bouton B2, la LED s'éteint et l'afficheur affiche 'B2'.
- Un appui sur le bouton B3, l'afficheur affiche 'B3'.

PICC : le wizard, production du squelette du programme.

Nous allons utiliser le logiciel PICC pour produire le programme. La programmation se fait en langage C. PICC nous permet pour les microcontrôleurs 8 et 16 bits de marque microchip :

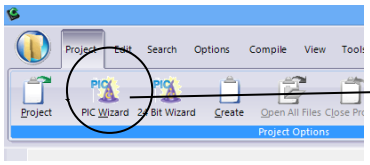
- De produire le squelette et la configuration de base du programme.
- D'éditer le programme en langage C.
- De compiler le programme source pour obtenir le programme en langage machine. Deux versions sont produites :
    - .HEX : programme binaire simple.
    - .COF : programme binaire contenant les éléments pour la simulation ou l'émulation en pas à pas.
- De programmer les microcontrôleurs.
- De tester à l'aide d'une sonde les programmes dans la cible.

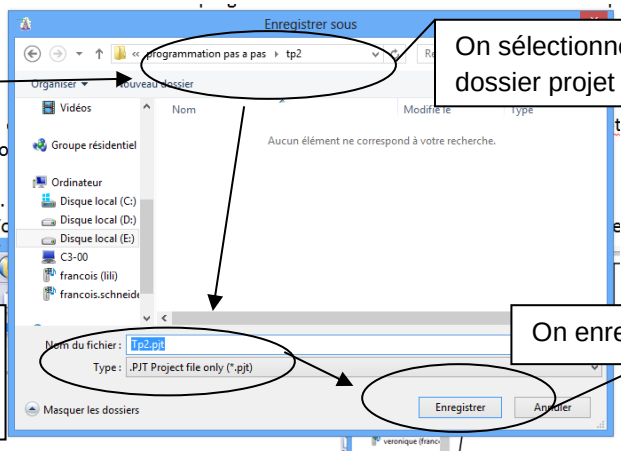
Dans ce document nous testerons les programmes par simulation avec Proteus. Vous devez avoir l'option pour la simulation des processeurs microchip, ici uniquement 8bits.

Nous allons compléter le programme en plusieurs phases. En dernière phase nous verrons comment utiliser un afficheur différent.

1. Création du squelette du programme en langage C avec PICC.

Vous lancez PICC. Nous allons utiliser le wizard pour produire le squelette et la configuration.



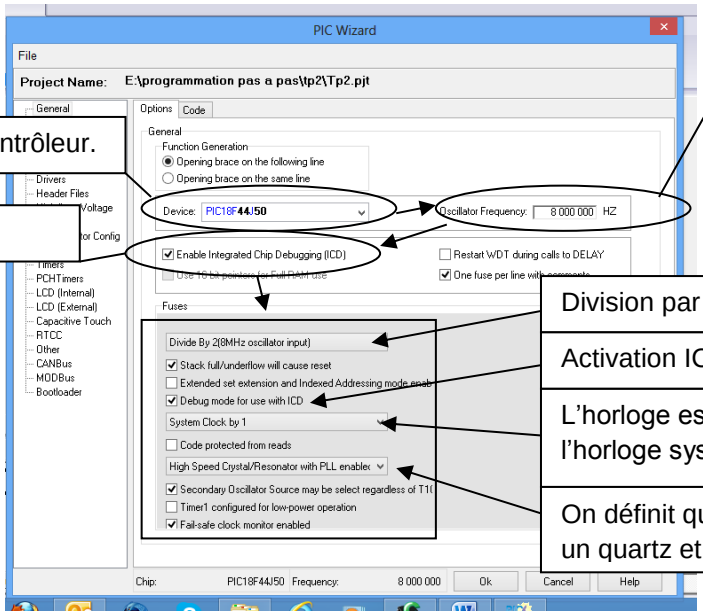


On sélectionne le dossier projet : Tp2

On donne un nom au projet : Tp2.pjt
Ici on donne le même nom au dossier et au projet, mais ce n'est pas obligatoire.

On enregistre le projet

Nous obtenons la première page du Wizard, il faut alors définir les valeurs des différentes ressources utilisées. Ici seul l'ICD et l'oscillateur sont utilisés.



Définir le microcontrôleur.

Définir la fréquence du quartz.

Activation de l'ICD.

Division par 2 de la fréquence du quartz.

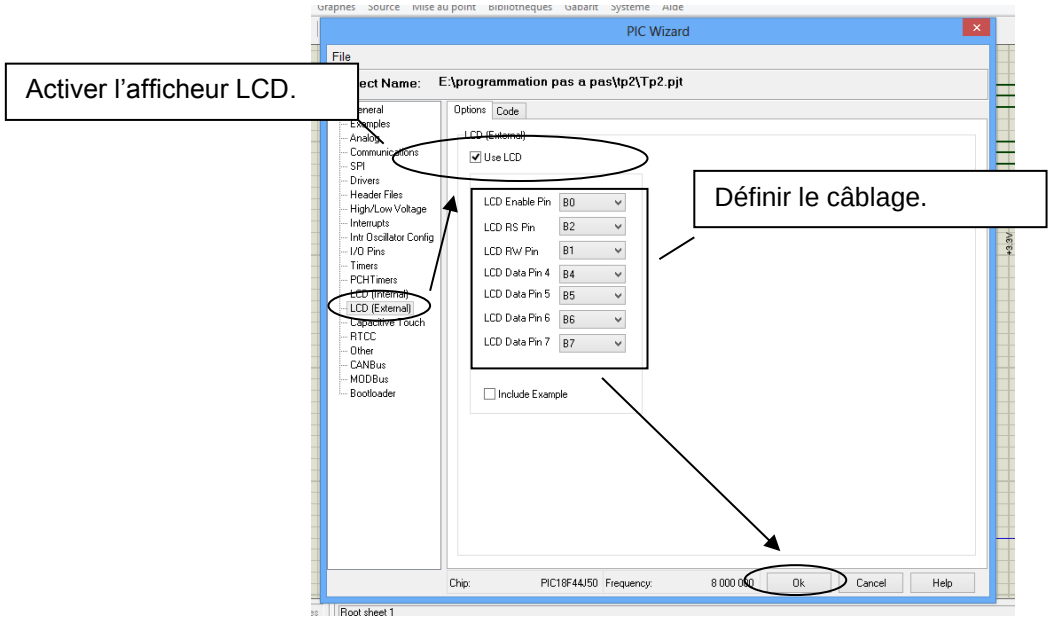
Activation ICD.

L'horloge est produite en divisant l'horloge système par 1.

On définit que l'horloge est produite par un quartz et on active la PLL.

Remarque : pour avoir plus de détails sur la configuration de l'horloge, il faut se référer à la notice technique du circuit PI18F44J50. Ici la configuration est prête pour utiliser l'USB en mode haute vitesse.

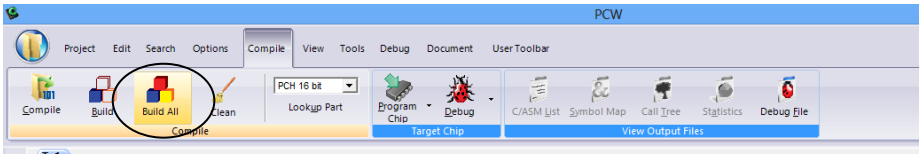
Nous allons maintenant configurer la section « LCD Externe ». Remarque il est configuré en mode 4 bits.



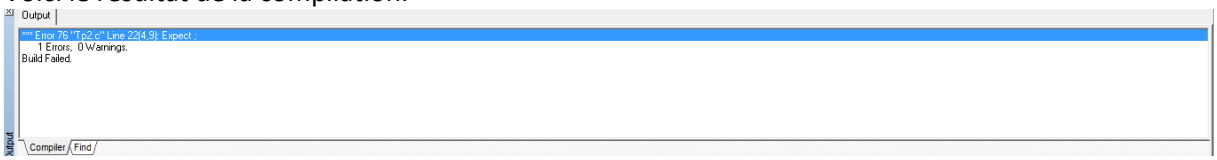
En activant le bouton Ok, le squelette du programme est réalisé : 2 fichiers sont produits.

- TP2.c : fichier contenant les instructions dont la fonction principale 'void main()'.
- TP2.h : fichier contenant les déclarations.

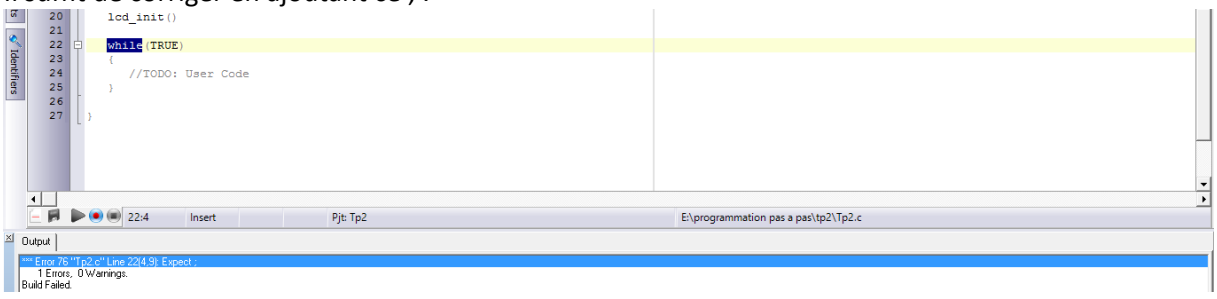
Nous pouvons le compiler le projet en cliquant depuis l'onglet 'Compile' en cliquant sur le bouton 'Build' :



Voici le résultat de la compilation.

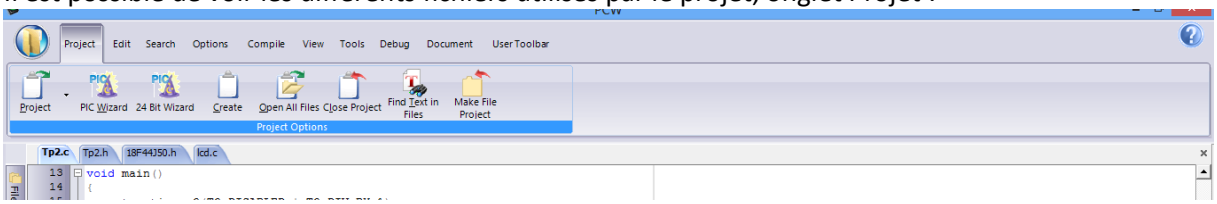


Nous voyons une erreur affichée. Sice n'est pas le cas pour vous c'est aussi bien.
Il faut alors cliquer 2x sur l'erreur et l'instruction while (TRUE) apparait marquée. L'erreur indique qu'il manque un ; (Expect ;), effectivement l'instruction lcd_init() n'est pas suivie par un ;
Il suffit de corriger en ajoutant ce ; .



Une nouvelle compilation donne 0 erreurs.

Il est possible de voir les différents fichiers utilisés par le projet, onglet Projet :



4 fichiers sont utilisés par le projet :

- Tp2.c : fichier produit par le wizard, contenant le squelette du programme.
- Tp2.h : fichier produit par le wizard contenant les déclarations.
- 18F44J50.h : fichier provenant de la librairie de PICC, contenant les déclarations particulières du composants 18F24J50. En jetant un coup d'œil, nous voyons l'affectation des ports, des registres, …
- Lcd.c : contient les librairies pour l'afficheur.

Quelques commentaires sur le programme :

Un programme est constitué d'une suite d'instructions, qui contiennent des variables, des constantes, des fonctions, des opérateurs, des structures conditionnelles… mais aussi des directives de compilation. Dans ce programme nous trouvons 2 directives essentielles :

#include : permet d'inclure un fichier dans le programme.

- Dans ce programme « #include <lcd.c> » permet d'inclure la librairie pour les afficheurs LCD.

#define : permet de créer des macros en donnant des équivalences. Elle est utilisée ici pour affecter les bornes de l'afficheur.

```
#define ENABLE_PIN B0
#define RS_PIN B2
#define RW_PIN B1
#define Data4 B4
#define Data5 B5
#define Data6 B6
#define Data7 B7
```

Nous avons intérêt à créer nos propres macros pour les boutons. Ainsi en cas de modification du câblage, il suffira de redéfinir les macros et non pas de modifier le programme.

```
#define bp1 PIN_RA3
#define bp2 PIN_RA4
#define bp3 PIN_RA4
```

Nous trouvons la fonction « lcd_init(); », qui permet d'initialiser l'afficheur.

2. Compléter le programme.
Le fichier Tp2.C

```
#include <Tp2.h>
#define ENABLE_PIN B0
#define RS_PIN B2
#define RW_PIN B1
#define Data4 B4
#define Data5 B5
#define Data6 B6
#define Data7 B7

#define bp1 PIN_RA3
#define bp2 PIN_RA4
#define bp3 PIN_RA4

#include <lcd.c>

void main()
{
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  setup_timer_4(T4_DISABLED,0,1);

  setup_comparator(NC_NC_NC_NC);//

  lcd_init();

  while(TRUE)
  {
    //TODO: User Code
  }
}
```

En rouge, la partie correspondant à la gestion de l'afficheur.

En bleu les macros ajoutées.

En marron la boucle du programme principal.

La structure conditionnelle while(TRUE) { } :

En langage C, il existe différentes structures conditionnelles, qui permettent de faire des instructions sous conditions ainsi que des boucles sous conditions.

L'instruction *while* permet d'exécuter plusieurs fois la même série d'instructions.

La syntaxe de cette expression est la suivante :

```
while (condition réalisée) {
         liste d'instructions;

}
```

Cette instruction exécute la liste d'instructions tant que (while est un mot anglais qui signifie tant que) la condition réalisée est vraie. Les accolades permettent de délimiter le début et la fin de la liste d'instruction.

Pour avoir plus d'informations cliquez ici.

Nous revenons au programme « Tp2.c ». La boucle

```
while(TRUE)
{
  //TODO: User Code
}
```

Les instructions de configuration se place avant cette boucle.
La boucle s'effectue de façon perpétuelle car la condition est TRUE (vraie).
//TODO: User Code : indique ou il faut mettre la suite d'instruction du programme principal.

Nous allons afficher le message « Bonjour » au démarrage du système et ensuite attendre.
Comment faire ?

Il faut utiliser une fonction, qui affiche le message et la placer avant la boucle. Quelles sont les instructions utilisables :

a. En début du fichier « lcd.c » sont décrites les différentes fonctions associées à l'afficheur LCD. Vous trouvez par exemple la fonction lcd_putc(c). Nous allons donc utiliser cette fonction pour afficher le premier caractère du message.
L'instruction est lcd_putc('B') ;

La fonction main() devient :

```
void main()
{
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  setup_timer_4(T4_DISABLED,0,1);

  setup_comparator(NC_NC_NC_NC);// This device COMP currently not supported by the
PICWizard

  lcd_init();
  lcd_putc('B') ;  // affichage de la lettre B sur l'afficheur LCD.
  while(TRUE)
  {
    //TODO: User Code
  }
}
```

On compile, on teste le fonctionnement avec Proteus, rien ne s'affiche. Il existe une erreur au niveau du code généré par le Wizard de Picc connue au niveau des macros de l'afficheur, il faut remplacer les lignes correspondants à l'afficheur par :

```
#define LCD_ENABLE_PIN PIN_B0
#define LCD_RS_PIN PIN_B2
#define LCD_RW_PIN PIN_B1
#define LCD_Data4 PIN_B4
#define LCD_Data5 PIN_B5
#define LCD_Data6 PIN_B6
#define LCD_Data7 PIN_B7
```

Vous modifiez le programme, compilez et relancez la simulation. Bravo cela fonctionne.

Pour afficher le message complet, il suffit soit d'écrire les caractères les uns après les autres ou de faire une boucle sur la longueur du message.

Pas simple, il existe une fonction pour faire cela. La fonction printf().

b. Utilisation de la fonction printf().

Printf sait afficher une chaine de caractère.
Pour afficher le message ii suffit d'écrire :

- printf(lcd_putc,"Bonjour"); //lcd_putc permet d'envoyer l'affichage vers l'afficheur LCD.

La fonction main() devient :

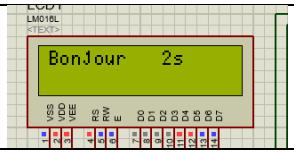
```
void main()
{
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  setup_timer_4(T4_DISABLED,0,1);

  setup_comparator(NC_NC_NC_NC);// This device COMP currently not supported by the
PICWizard

  lcd_init();
  printf(lcd_putc,"Bonjour");
  while(TRUE)
  {
    //TODO: User Code
  }
```

c. Utilisation avancée de la fonction printf().

Je vous propose d'utiliser la fonction main() suivante afin de voir la puissance de la fonction printf() :

```
void main()
{
  unsigned int i = 0;  // creation variable I avec valeur innitiale = 0
  setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
  setup_timer_4(T4_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  lcd_init();
  printf(lcd_putc,"Bonjour"); // affichage Bonjour vers LCD
  while(TRUE)
  {
   delay_ms(1000);
   i = i + 1;
   printf(lcd_putc,"\fBonjour %3us",i);    // affichage "Bonjour xxxs" vers LCD
  }
}
```

Effectuez les essais.

printf(lcd_putc,"\fBonjour %3us",i); - \f : permet d'effacer l'écran - %3d : définie comment i est affiché. Ici comme un entier non signé sur 3 caractères.	

Vous pouvez avoir plus de détails dans l'aide de PICC.

3. Vous utilisez un afficheur différent par exemple un afficheur « EA DOGM162E-A ».

Nous constatons qu'en utilisation 4bits (data) l'écriture et la lecture sont identiques avec un afficheur LCD traditionnel.
Les 2 afficheurs ont un fonctionnement identique en mode 4 et 8 bits.
La différence se résume à écrire une fonction d'initialisation de l'afficheur différente. Je vous propose celle-ci pour un afficheur alimenté en 3,3v. Il s'agit de la fonction lcd_init(void) de la librairie « lcd.c ».

Nous allons la nommer lcd_init_DOGM162E.

```
void lcd_init_DOGM162E (void)
 {
 #if defined(__PCB__)
   set_tris_lcd(LCD_OUTPUT_MAP);
 #else
  #if (defined(LCD_DATA4) && defined(LCD_DATA5) && defined(LCD_DATA6) && defined(LCD_DATA7))
   output_drive(LCD_DATA4);
   output_drive(LCD_DATA5);
   output_drive(LCD_DATA6);
   output_drive(LCD_DATA7);
  #else
   lcdtris.data = 0x0;
  #endif
   lcd_enable_tris();
   lcd_rs_tris();
   lcd_rw_tris();
 #endif

   lcd_output_rs(0);
   lcd_output_rw(0);
   lcd_output_enable(0);

//wait for a short period, maybe the voltage needs to stabilize first
   delay_ms(50);

//ici on initialize le mode 4 bits.

   lcd_send_nibble(3); // envoie mot de 4 bits.
   delay_ms(2);
   lcd_send_nibble(3);
   delay_ms(2);
   lcd_send_nibble(3);
```

```
  delay_ms(2);
  lcd_send_nibble(2); // MODE 4 BITS
  delay_ms(2);
  // ici on configure l'afficheur : …..
  lcd_send_byte(0,0x29);  // envoie mot de 8 bits.
  lcd_send_byte(0,0x14);
  lcd_send_byte(0,0x55);
  lcd_send_byte(0,0x6D);
  lcd_send_byte(0,0x78);
  lcd_send_byte(0,0x28);
  lcd_send_byte(0,0x0C);
  lcd_send_byte(0,0x01);
  lcd_send_byte(0,0x06);

 #if defined(LCD_EXTENDED_NEWLINE)
  g_LcdX = 0;
  g_LcdY = 0;
 #endif
}
```

Pour l'utiliser, il suffit de modifier le fichier Tp2.c de la façon suivante :

- On recopie le texte ci-dessus en début du fichier Tp2.c après la directive #include <lcd.c>.
- On remplace l'appel à la fonction lcd_init(void) par l'appel à la fonction lcd_init_DOGM162E(void).

Effectuer les modifications et vérifier le fonctionnement. Vous pouvez ensuite tester sur votre carte.

4. Création d'une libraire pour l'afficheur « EA DOGM162E-A ».

Il est intéressant de créer un fichier, qui contient le programme d'initialisation de l'afficheur. Il sera alors simple de l'utiliser dans un autre projet. Pour ce il faut :

- Créer un fichier « lcd_init_DOGM162E.c »
- Copier la fonction void lcd_init_DOGM162E (void) dans ce fichier. ?
- Inclure ce fichier dans le fichier TP2.c avec la ligne #include " lcd_init_DOGM162E.c".

Il est intéressant d'ajouter en début du fichier "cd_init_DOGM162E.c" la directive #include <lcd.c>, ainsi en début du programme principal il n'y a que la directive #include " lcd_init_DOGM162E.c" à placer.

Effectuer la modification et valider le fonctionnement.

5. Gestion des boutons. A faire…