MICROCHIP PICs

Travaux Pratiques

Objectifs : ce TP a pour but d'illustrer le fonctionnement des PICs et de découvrir les outils de développement proposés par Microchip pour programmer ce type de microcontrôleurs. On s'intéresse ici au PIC 16F84a qui fait partie des PICs d'entrée de gamme de Microchip, donc plus facile à prendre en main. On programmera dans un 1er temps en langage assembleur avec le jeu d'instructions propre au processeur afin de mieux comprendre son fonctionnement. On verra ensuite comment programmer les PICs en langage C de plus haut niveau.

Environnement de travail

La chaîne d'outils utilisée pour le développement est MPLAB X, fournie gratuitement par MICROCHIP. Il s'agit d'un IDE (Integrated Development Environment) qui intègre tous les outils nécessaires pour la mise au point d'un projet :

- éditeur de fichier source
- compilateur + assembleur
- éditeur de lien (permet d'obtenir le fichier exécutable de sortie)
- "débuggeur" (simulateur et/ou vrai débuggeur si le circuit cible le permet)
- programmateur

Cet IDE offre l'avantage de pouvoir effectuer toutes ces opérations dans un même environnement de travail et facilite grandement les interactions entre les modules (gestion et mise à jour automatique des fichiers d'échanges, exécution d'un seul clic des modules permettant l'obtention du fichier exécutable...) d'où un gain évident de productivité.

Nous utiliserons MPLAB X en mode Simulator pour tester notre application sur Proteus.

Le logiciel MPLAB X, le compilateur XC8 et l'outil de conception Proteus doivent être installés au préalable pour les besoins du TP.

TP

Création d'un projet

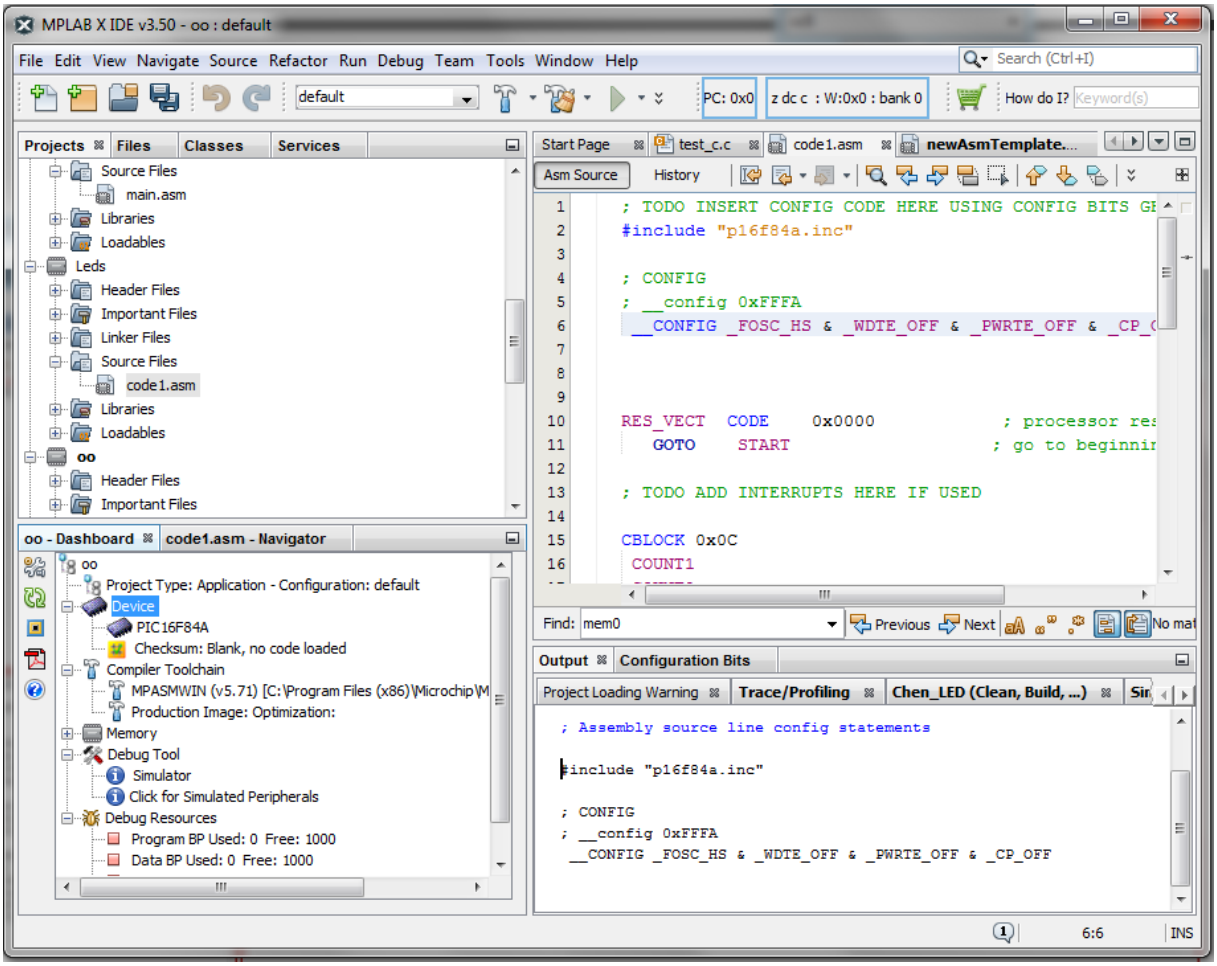
→ FILE > NEW PROJECT

1- Choose project : choisir les options Microchip Embedded et Standalone Project

2- Select Device : choisir le PIC16F84A dans la famille Mid-Range 8-bit MCUs (PIC10/12/16/MCP)

3- Select Tool : choisir « Simulator »

Une fois le projet crée, l'environnement de travail de MPLAB X se compose de 4 zones d'affichage arrangées selon la figure ci-dessous :



- La fenêtre "Projects" permet d'accéder à tous les fichiers qui composent le projet, classés par catégories

- La fenêtre d'édition permet de visualiser voire modifier les fichiers sélectionnés par un double clic

- La fenêtre "Dashboard" résume les paramètres du projet (ceux-ci sont modifiables à tout moment en double cliquant sur le nom du projet)

- La fenêtre "Output" affiche toutes les informations liées au déroulement du projet. C'est ici que s'affichent automatiquement tous les résultats à l'issu de chaque phase de la conception (compilation, programmation…). On peut également y ajouter différents onglets (via le menu WINDOW) permettant de configurer le PIC, obtenir des informations sur la simulation, voir la mémoire du PIC…

Tâche 1

Faire un chenillard simple sur 8 leds. Cela consiste à créer un mouvement lumineux en allumant et éteignant successivement une série de LED. L'effet se traduit par un déplacement de cette lumière dans un sens choisi, par exemple de droite à gauche pour commencer.

Nous allons nous basé sur le code LED Blink (du cours) :

```
 ; TODO INSERT CONFIG CODE HERE USING CONFIG BITS GENERATOR

#include "p16f84a.inc"

; CONFIG

; __config 0xFFFA

 __CONFIG _FOSC_HS & _WDTE_OFF & _PWRTE_OFF & _CP_OFF


RES_VECT  CODE    0x0000            ; processor reset vector

  GOTO    START                   ; go to beginning of program


; TODO ADD INTERRUPTS HERE IF USED


MAIN_PROG CODE                      ; let linker place main program

 START

   BSF STATUS, RP0

   MOVLW 0xFE

   MOVWF TRISB

   BCF STATUS, RP0


 MAIN
```

```
   BSF PORTB,0
   CALL DELAY
   BCF PORTB,0
   CALL DELAY
   GOTO MAIN

DELAY
   LOOP1 DECFSZ COUNT1, 1
   GOTO LOOP1
   DECFSZ COUNT2,1
   GOTO LOOP1
   RETURN
END
```

Remarque : ce qui vient après le « ; » est un commentaire.

Indications :

- Initialiser un registre (une variable d'un octet) « memo » par exemple

- Utiliser l'instruction RLF (qui fait que le contenu du registre F est décalé de 1 bit vers la gauche à travers le flag de retenue)

```
MOVLW  0xE6          ;    W= 0xE6  =  11100110
MOVWF memo           ; memo = 11100110 et supposons C=0
RLF   memo,1                ; memo = 11001100 et C = 1
```

Tâche 2

Rajouter un premier bouton pour déclencher le chenillard (par exemple au port RA0).

Indications :

Utiliser l'instruction **btfss f,b** (Si le b ème bit du registre f est égal à 1, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place).

Tâche 3

Rajouter un deuxième bouton pour changer le sens (par exemple au port RA1).

Tâche 4

Code la même application en C.

Exemple :

```
// PIC16F84A Configuration Bit Settings

// 'C' source line config statements

// CONFIG
#pragma config FOSC = HS       // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF      // Watchdog Timer (WDT disabled)
#pragma config PWRTE = ON      // Power-up Timer Enable bit (Power-up Timer is enabled)
#pragma config CP = OFF        // Code Protection bit (Code protection disabled)

#include <xc.h>

init (void)     // initialisation du PIC
```

```
{
?????
}

main (void)
{
init ();

while (1)          // boucle infinie indispensable dans la fonction main () !
{
?????
}
}
```