TP0 : PRISE EN MAIN DE L'MPLAB XC8

1. MPLAB IDE

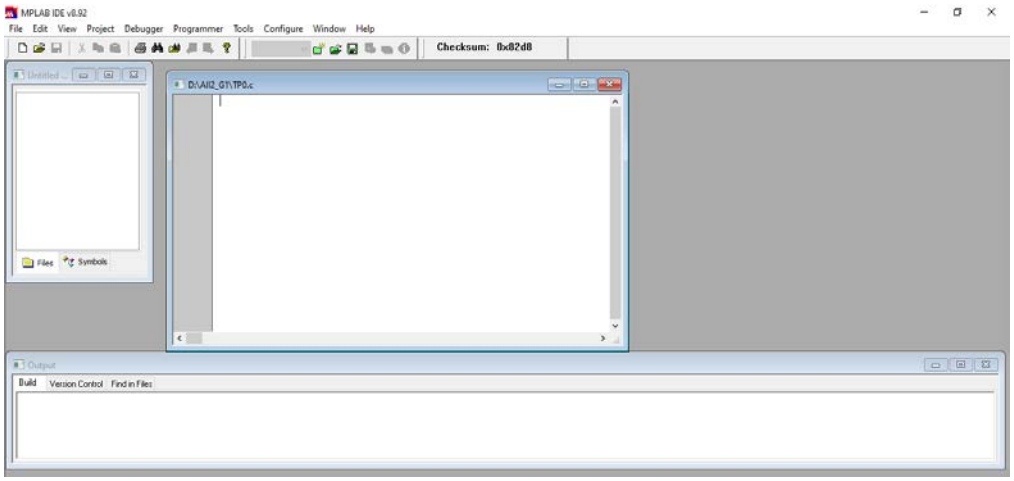
MPLAB IDE est un outil de développement complet pour les microcontrôleurs PIC. Cet environnement intègre tous les outils nécessaires à la programmation des microcontrôleurs (Assembleur, Compilateurs, simulateur, déboguer et programmateur des circuits).

2. Création d'un projet avec MPLAB XC8

L'environnement **MPLAB** sauvegarde vos applications au sein de projets qui s'apparentent à un fichier ***Projet*** unique (avec l'extension. mcp). Ce projet comporte un ensemble de fichiers : fichiers Sources « .c », fichiers Entêtes « .h », fichiers Objets « .o » et fichiers librairies « .lib. Pour créer un projet, il faut suivre les étapes suivantes :

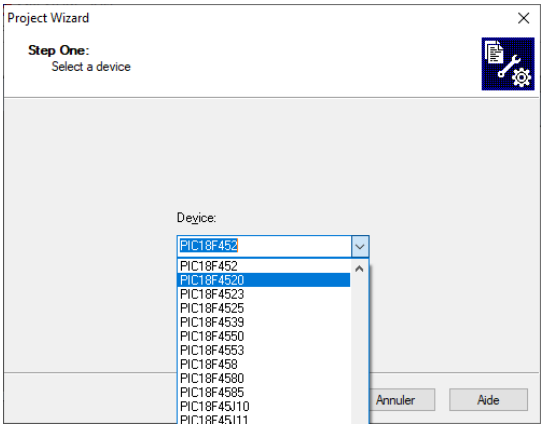
Avant ce lancer le logiciel MPLAB IDE, créer votre répertoire de travail dans le disque D: « **D:\NomClasse_GoupeNum** », Vous devez travailler dans ce répertoire le long du semestre.

Etape 1 : Lancez le logiciel MPLAB, choisir la commande ***New*** du menu ***File*** pour créer le fichier source. Enregistrez le nouveau fichier vide sous le nom **TP0.c**, Attention, ce fichier doit être sauvegardé dans votre répertoire de travail.

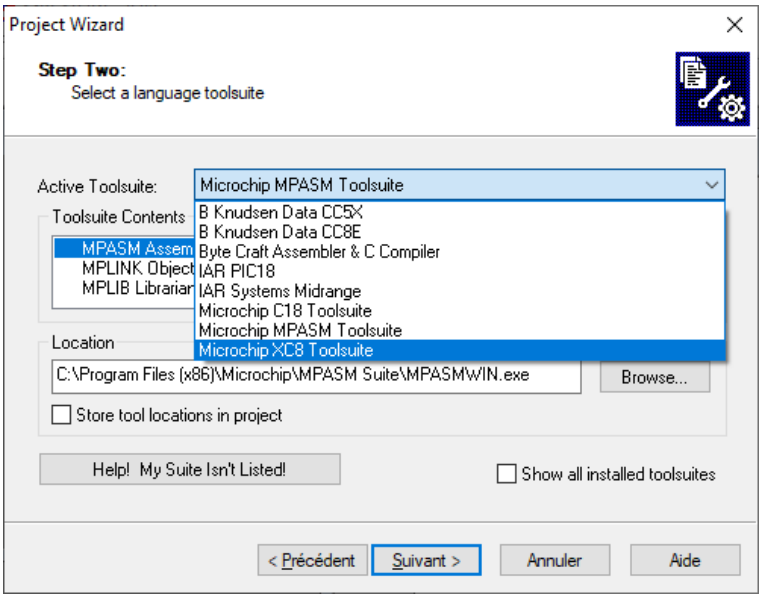


Etape 2 : Choisir ***Project>Wizard…*** pour créer votre projet. Dans la fenêtre qui s'ouvre, cliquez sur **suivant**.

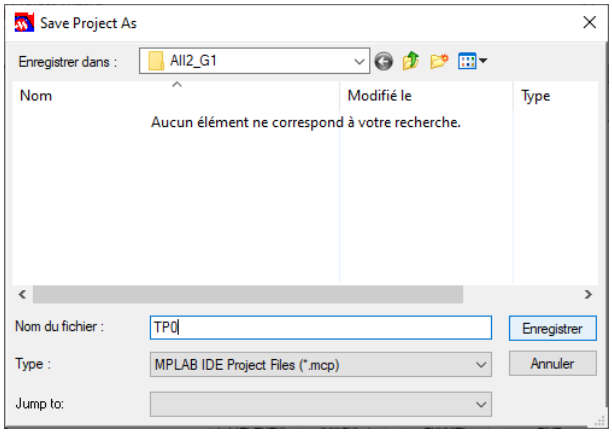
Etape 3 : sélectionnez le microcontrôleur **PIC18F4520**, puis cliquez sur s**uivant**.



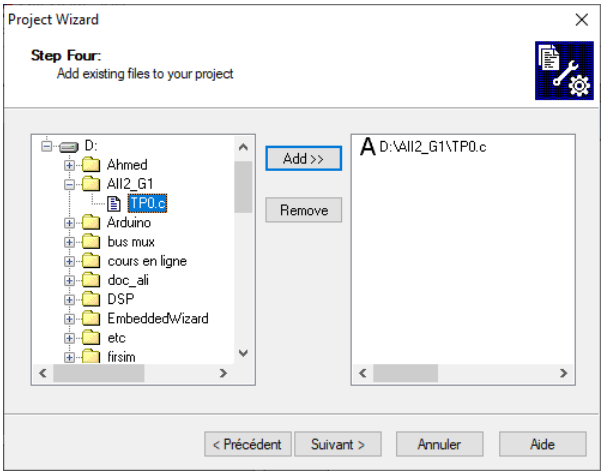
Etape 4 : sélectionnez, le compilateur **Microchip XC8 Toolsuite**, puis cliquez sur s**uivant**.



Etape 5 : Dans la fenêtre qui s'ouvre, cliquez sur le bouton **Browse….** Sélectionnez votre dossier de travail (All2_G1 dans notre exemple) puis donnez un nom à votre projet (**TP0** par exemple). Enregistrer votre projet et cliquez sur le bouton **suivant**.

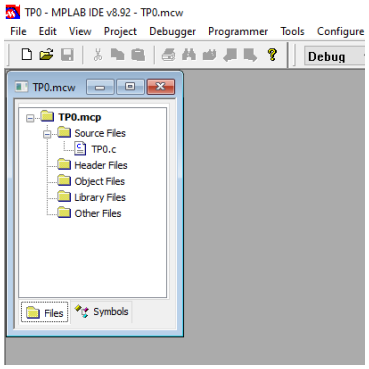


Etape 5 : Dans la fenêtre qui s'ouvre, ne faites rien, cliquer sur le nom le bouton **suivant.** Puis cliquez sur le bouton **terminer** dans fenêtre suivante.



Etape 6 : Si la fenêtre du Project n'est pas visible, affichez-la, en activant la commande ***Project*** du menu ***View***. Positionnez le pointeur de la sourie sur le dossier ***Source Files***, cliquez sur le

bouton droit de la sourie. Dans la fenêtre qui s'ouvre choisissez la commande ***Add Files…*** et sélectionnez votre fichier source (**TP0.c** dans cet exemple).



Le projet est maintenant créé, vous pouvez saisir votre programme. Ouvrez le fichier source **TP0.c** et tapez le programme suivant :

```
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF, DEBUG = OFF, PWRT = OFF
unsigned long int count ;
void main(){
     TRISC = 0x00;
     PORTC = 0x00;
     while(1)
     {    PORTC = ~PORTC;
          count = 1000000;
          do{
               count = count - 1;
          }while(count != 0);
     }
}
```

3. Compilation et chargement du programme

Une fois que vous aurez créé votre projet et écrit le code source, vous pouvez compiler votre programme en sélectionnant la commande ***Build All*** du menu ***Project***, ou cliquez sur le bouton ***Build All***. Vous devez avoir le message « **`Build successfull`** », sinon corrigez vos erreurs et compilez à nouveau votre programme.

Pour tester votre programme sur le kit, suivez les étapes suivantes :

- ⇨ Branchez le kit **µc-pic2A** sur le réseau.
- ⇨ Connectez-le au PC via la lisaison USB
- ⇨ Démarrez le logiciel **PICkit2,** le logiciel doit détecter automatiquement le composant « PIC18F4520 ».
- ⇨ Chargez le fichier «TP0.hex », en utilisant la commande ***Import HEX*** du menu **File**.
- ⇨ Démarrez la programmation du microcontrôleur en cliquant sur le bouton **Write.**

Pour augmenter le temps d'attente, remplacez la boucle d'attente par les instructions suivantes :

```
for (count = 0 ; count < 10 ; count--)
     __delay_ms(50) ;
```

TP1 : PROGRAMMATION DES PORTS D'ENTREES-SORTIES

1. Objectifs

- ⇨ Etre capable de gérer les ports parallèles en langage C
- ⇨ Savoir utiliser l'accès individuel aux bits.
- ⇨ Utiliser les fonctions en langage C.

2. Environnement de travail

- Compilateur MPLAB XC8 et le logiciel de programmation PICkit2.
- Une carte d'étude µc-PIC2A permettant l'implémentation et la vérification pratique des applications. Le programmateur PICkit2 est intégré dans le kit.

3. Travail demandé

3.1 PORTS E/S

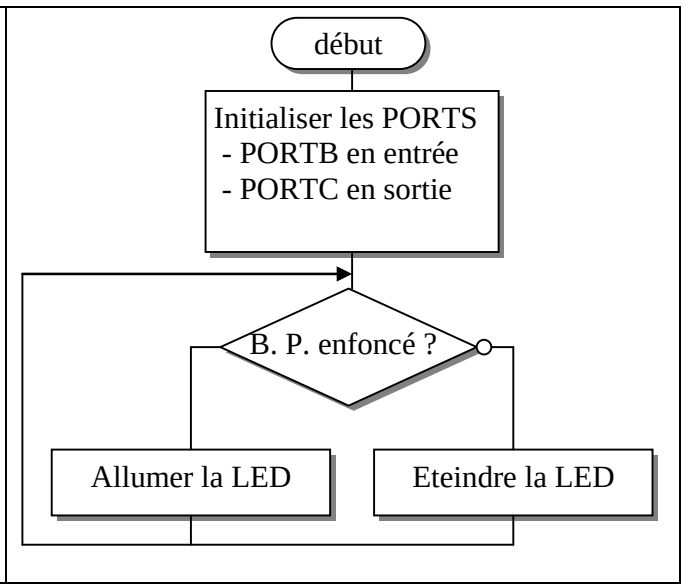
Programmez le microcontrôleur pour réaliser les tâches suivantes :

1) *Compteur simple* : utilisez le PORTC pour réaliser un compteur binaire qui s'incrémente une fois par seconde. Affichez l'état du PORTC sur les LEDs.
2) *Chenillard simple* : faites défiler l'allumage des LEDs de droite à gauche. Fixer le temps de défilement à 200ms.
3) Le PORTC est initialisé à la valeur 55H, faites clignoter la LED connecté à la broche RC2 sans changer l'état des autres bits.

3.2 Scrutation des entrées

1) Complétez le programme suivant afin d'allumer la LED connecté à la broche RC2 lorsqu'on appui sur le Bouton Poussoir RB0.

```
#include <xc8.h>
#define _XTAL_FREQ 8000000
#pragma config ……
void main()
{    ADCON1 = 0x0F ;
     TRISB = 0xFF ;
     TRISC = 0x00 ;
     PORTC = 0x76 ;
  ……
     if(……)
     …… ;
     else
     …… ;
  ……
}
```



2) Il est possible d'accéder directement à un seul bit sans avoir recours à des opérations de masquage en utilisant une union des structures anonymes des bits adressables « PORTxbits ».

Exemple : l'instruction **PORTCbits.RC0 = 1** : met le bit 0 du PORTC à 1, sans modifier les autres bits du PORTC. Le compilateur MPLAB XC8, accepte aussi l'accès individuel aux bits, au sens du compilateur **Hi-Tech**, en écrivant simplement **RC0 = 1**.

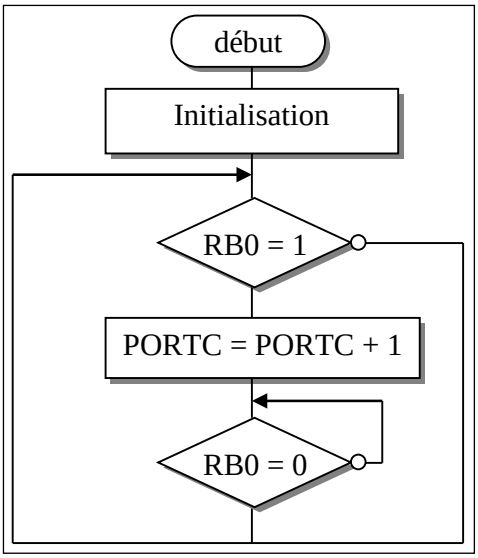
Reprendre alors le programme précédent en utilisant l'accès individuel aux bits.

3) Pourquoi doit-on faire la scrutation du bit RB0 dans une boucle.

3.3 Détection de front

On se propose ici de compter le nombre de paquets passant devant une cellule photoélectrique. La détection de passage d'un paquet est simulée par l'appui sur le bouton poussoir AR1 qu'on le branche à l'entrée RB0. Chaque appui sur le bouton AR1 incrémente le PORTC d'une unité. L'état du PORTC est affiché sur les LEDs.

1. Ecrire le programme de comptage des paquets.
2. Est-ce que le compteur s'incrémente convenablement ? Pourquoi ?
3. Afin d'éviter l'incrémentation du PORTC avec un pas aléatoire, on vous propose l'organigramme ci-contre.



Réécrire et tester à nouveau votre programme sur le kit. Que constatez-vous ? Conclure.

Connectez maintenant l'entrée RB0 à la borne « SW1 » au lieu de la borne « AR1 ». Appuyez sur le bouton RESET et exécutez de nouveau le programme. Que constatez-vous ? Apportez les modifications nécessaires à votre programme afin d'éliminer l'effet de rebondissement du bouton poussoir.

3.4 Programmation structurée (utilisation des fonctions)

Le corps d'une fonction est délimité par des accolades { ..... }. Un programme C doit se composer de plusieurs fonctions pour permettre une meilleure lisibilité et maintenance du programme.

La fonction main () est exécutée en premier, elle gère le bon déroulement du programme et l'échange des données entres les autres fonctions.

Exemple d'application : Traduire l'algorithme suivant en un programme C, comportant les fonctions suivantes :

- une fonction « *Init()* » pour l'initialisation des ports
- une fonction de temporisation « *delay_ms(int n)* »
- une fonction principe « *main()* ».

```
début
     TRISC ← 0x00;          // Initialisation
//-------------------------------------------------------
     Tant que (toujours vrai)
          PORTC ← PORTC;
//-------------------------------------------------------
          Attente 500 ms     // temporisation
//-------------------------------------------------------
     Fin tant que
fin
```