TP N° 01 Prise en main de l'environnement du logiciel Proteus ISIS et de compilateur MikroC PRO pour PIC.

1. Objectif

L'objectif de ce premier TP est de se familiariser avec l'environnement du logiciel ***Proteus*** et de compilateur ***MikroC PRO*** pour programmer un microcontrôleur ***PIC***.

2. Introduction

Un microcontrôleur est un circuit électronique encapsulé dans un circuit de haut niveau d'intégration. Les microcontrôleurs sont commercialisés par différents fabricants comme ***Motorola***, ***Intel***, ***Philips***, et ***Microchip***.

Microchip en particulier, est un fabricant des circuits électroniques. Dans leurs lignes de production, on trouve les microcontrôleurs ***PIC***, qui sont disponibles dans des différentes familles, tel que ***12F***, ***16F***, ***18F***, ***24F***, ***30F***, ***33F*** …etc.

Pour ce TP, nous avons opté pour le PIC 16F877 de ***Microchip*** qui dispose de 40 broches et des modules tels que : Timer, ADC, USART, I2C, PWM, entre autres.

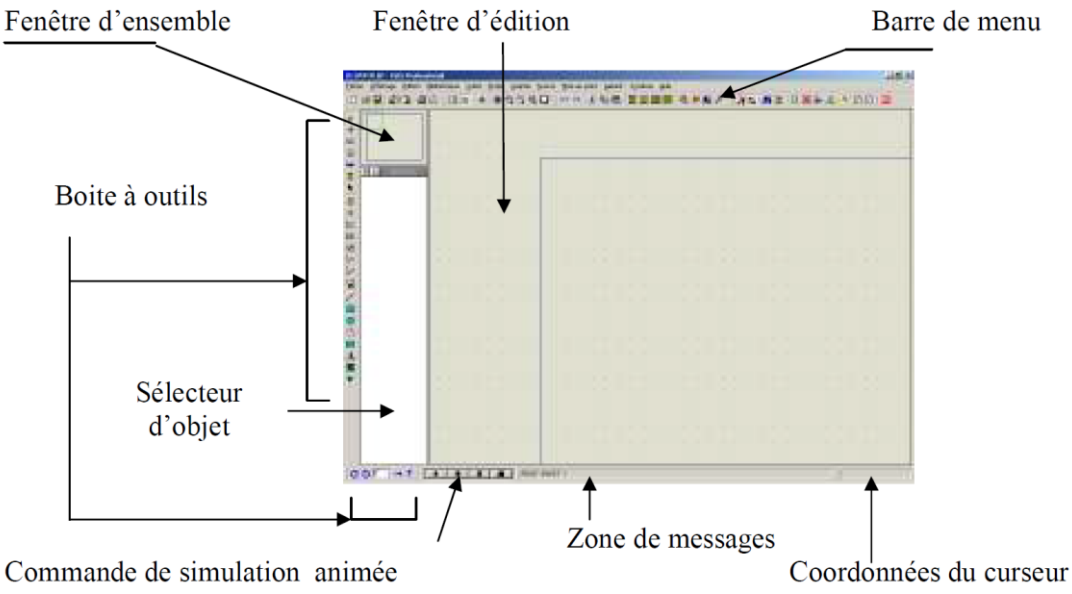
3. Isis Proteus

Le ***«Proteus»*** est une suite de logicielle permettant la CAO électronique éditée par la société *Labcenter Electronics*. Proteus est composé de deux logiciels principaux :

ISIS, qui est un très bon logiciel de simulation en électronique. Il est un éditeur de schémas qui intègre un simulateur analogique, logique ou mixte. ***ARES***, dédié à la création de circuits imprimés.

Grâce à des modules additionnels, ISIS est également capable de simuler le comportement de différents microcontrôleurs (PIC, Atmel, 8051, ARM, HC11...) et son interaction avec les composants qui l'entourent.

Interface utilisateur de Proteus



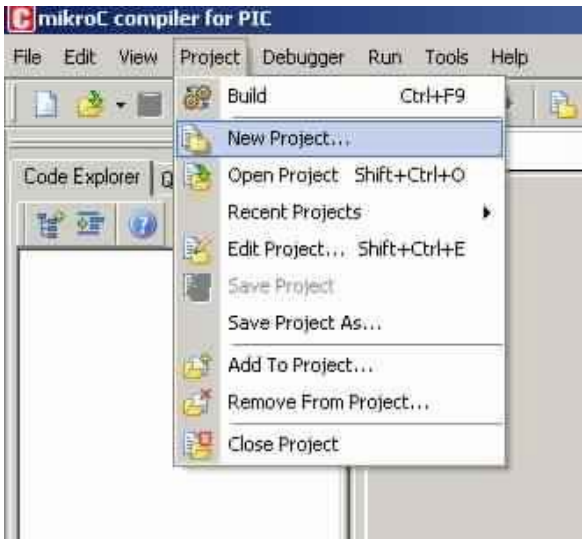
Note : La simulation permet d'ajuster et de modifier le circuit comme si on manipulait un montage réel. Ceci permet d'accélérer le prototypage et de réduire son coût. Il faut toujours prendre en considération que les résultats obtenus de la simulation sont un peu différents de celles du monde réel.

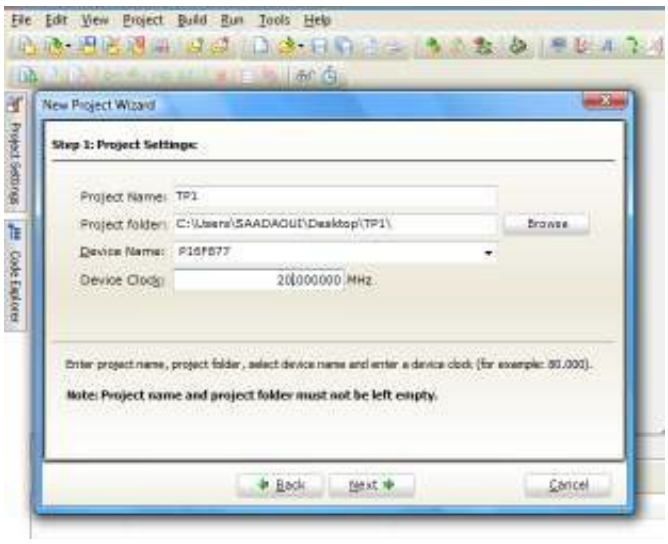
4. MikroC PRO :

Le « ***mikroC PRO*** » est un compilateur pour PIC Conçu par la société «***Mikroelektronika*** », le compilateur C nouvelle génération "***MikroC PRO***" pour microcontrôleurs PIC bénéficie d'une prise en main très facile. Il comporte plusieurs outils intégrés (mode simulateur, terminal de communication, gestionnaire 7 segments,...). Il a une capacité à pouvoir gérer la plupart des périphériques rencontrés dans l'industrie (Bus I2C, 1Wire, SPI, RS485, Bus CAN, cartes compact Flash, signaux PWM, afficheurs LCD et 7 segments...), de ce fait il est un des outils de développement incontournable et puissant. Il contient un large ensemble de bibliothèques de matériel, de composant et la documentation complète.

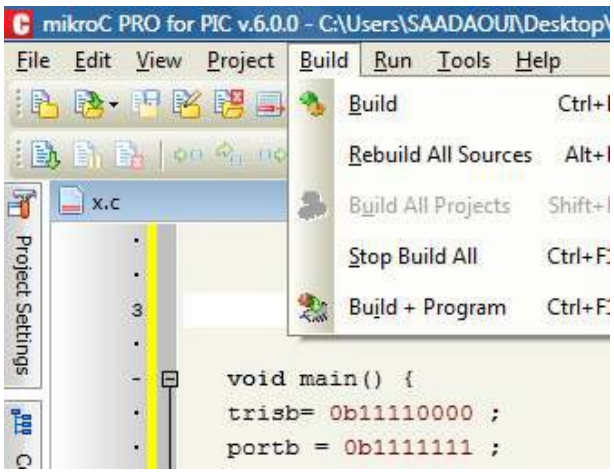
Création d'un nouveau projet sous MicroC

Avec ***mikroC PRO***, on crée un nouveau projet (*Project --> New Projet*) , puis on choisit le PIC16f877 et un quartz de 20Mhz.



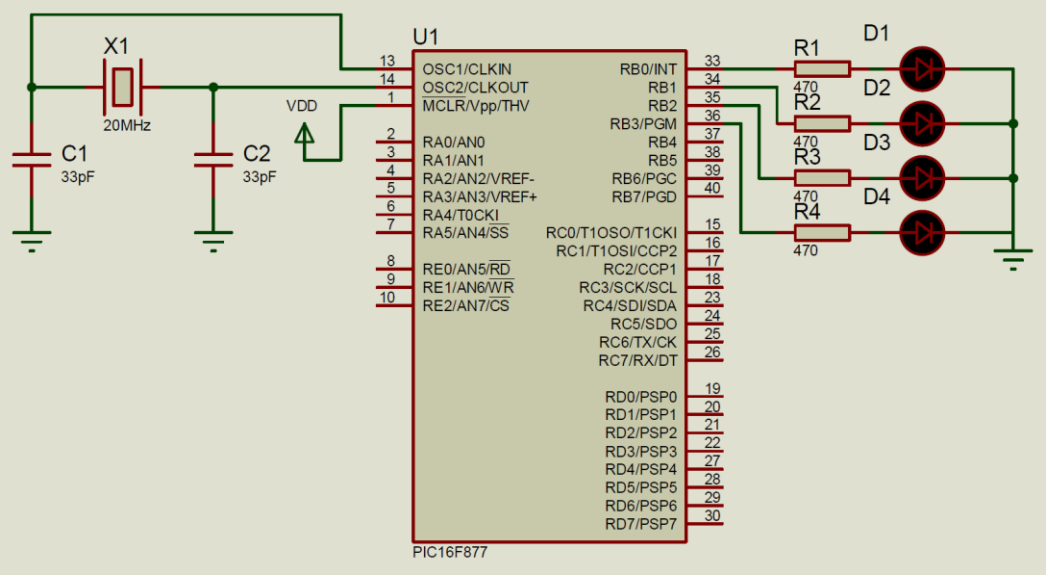


On Saisit ensuite le programme puis on le compile, le compilateur crée automatiquement le code assembleur et un code enregistré dans un fichier avec l'extension ****. Hex***,



5. Travail demandé :

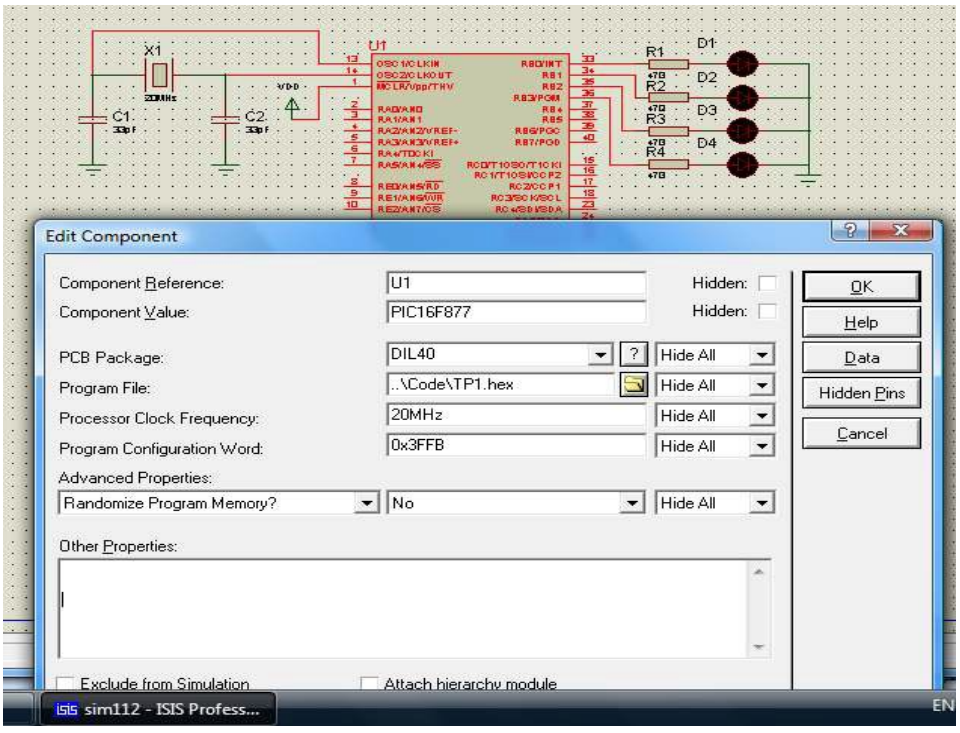
- **4.1.** Installer Proteus et mikroC PRO.
- **4.2.** Lancer ISIS et le Compilateur mikroC PRO.
- **4.3.** Réaliser le circuit de test ci-dessous sous ISIS :



- **4.4.** Ouvrir un nouveau projet sous mikroC PRO, écrire le programme ci-dessous :

```
void main() {
  trisb= 0b11110000 ;    // direction du port b (1 : Entrée, 0 : Sortie)
  while(1)               // boucle infinie
 {
  portb = 0b00001111 ;  //les 4 leds allumées
 }
 }
```

Après compilation et s'il n'y a pas d'erreurs un fichier ***TP1.hex*** est créé. Dans ***ISIS***, on double click sur le microcontrôleur et on spécifie le fichier ***TP1.hex*** qui se trouve dans le dossier TP puis OK.



Par la suite lancer la simulation (le bouton *Play* du panneau de contrôle de l'animation), les 4 Leds seront allumées.

- **4.5.** Refaire **4.4** pour allumer 4 autres Leds liées au port c du pic.

Maintenant on va transformer notre programme pour que les Leds clignoteront.

- **4.6.** Refaire **4.4** pour le programme suivant :

```
void main() {
   trisb= 0b11110000 ;
while(1)
{
 portb = 0b11110000 ;
delay_ms(200) ;
 portb = 0b11110001 ;
delay_ms(200) ;
portb= 0b11110010  ;
delay_ms(200) ;
portb =  0b11110100 ;
delay_ms(200) ;
portb =  0b11111000 ;
delay_ms(200) ;
  }
                }
```

- **4.7.** Dans notre exemple on a allumé les Leds en agissant sur le port en entier, on peut accéder individuellement à chaque bit en utilisant les identifiants ***F0, ..., F7***, c.-à-d. pour allumer la première Led, on écrit ***Portb.F0 = 1***.

```
void main() {
trisb= 0b11110000 ;
// 1 1 1 1 0 0 0 0
while(1)
{
portb = 0b11110000 ;
delay_ms(200) ;
portb.f0 = 1 ;
delay_ms(200) ;
portb.f0 =0 ;
portb.f1 = 1 ;
delay_ms(200) ;
portb.f1 = 0 ;
portb.f2 = 1 ;
delay_ms(200) ;
portb.f2 = 0 ;
portb.f3 = 1 ;
delay_ms(200) ;
}
}
```

- **4.8.** Ecrire un programme qui permet de faire clignoter les diodes paires pendant une seconde et les diodes impaires pendant une seconde en le testant avec ISIS.
- **4.9.** Rajouter 4 autres Led sur le port B (RB4, RB5, RB6, RB7), puis programmer un chenillard simple : La Led allumée se déplaçant sur le PORTB (de haut en bas).
- **4.10.** Programmer un chenillard double : un chenillard de haut en bas et simultanément de bas en haut qui se croisent.