Université Mohammed Premier
Ecole Nationale des sciences appliquées
- Oujda –

Génie des Systèmes Electronique
Informatique et Réseaux
GSEIR

TP 1 – Informatique embarquée

Prise en main de la carte EasyPic7 et mikroC PRO for PIC

Encadré par :

EMHARRAF Mohamed

1. Objectif

L'objectif de ce travail pratique est de se familiariser avec la carte de test EasyPic7 et ses différents composants, tout en acquérant les compétences de base pour la configuration, la compilation et l'exécution d'un code sur cette carte. Plus spécifiquement, cette manipulation portera sur la gestion de la mémoire RAM du microcontrôleur 18F4520.

2. Contenu du kit EasyPic7 et logiciels

Un kit, par binôme ou trinôme selon les possibilités offertes par la salle Robotique.

Le kit contient :

- La carte de test EasyPic7 avec son microcontrôleur 18F45K22

Les logiciels nécessaires au déroulement de vos séances des travaux pratique ont déjà été installés sur les machines. Il s'agit de :

- MikroC PRO (compilateur)
- MikroICD (outil de débogage)
- MikroProg Suite (programmation de la puce)
-

3. Observation de la platine EasyPic7

La carte de test EasyPic composée de différents modules dont les principaux sont numérotés sur la figure ci-dessous :

Compléter le tableau suivant :

Modul	n°
Potentiomètres - conversion analogique/numérique (CAN)	
Programmateur	
Connecteur USB (programmateur et alimentation)	
Afficheur 7 segments	
Connecteur RS-232	
LEDs de contrôle des ports	
Buzzer intégré	
Afficheur LCD 2x16	
Quartz	
Connecteur USB-UART	
Afficheur GLCD	
Microcontrôleur	
Zone des switches	
Potentiomètre de réglage de la luminosité des LCDs	
Boutons poussoirs de contrôle des ports	
Bouton reset de la platine	
Zone d'alimentation	
Support capteur de T° DS1820	
Connecteur USB	
Support carte fille	

2. Création d'un premier projet

mikroC permet de créer des projets qui contiendront un seul fichier projet (fichier avec l'extension **.mcppi**) et un ou plusieurs fichiers sources (fichiers avec l'extension .c). Il est possible de gérer plusieurs projets à la fois. Les fichiers sources ne peuvent être compilés que s'ils font partie d'un projet.

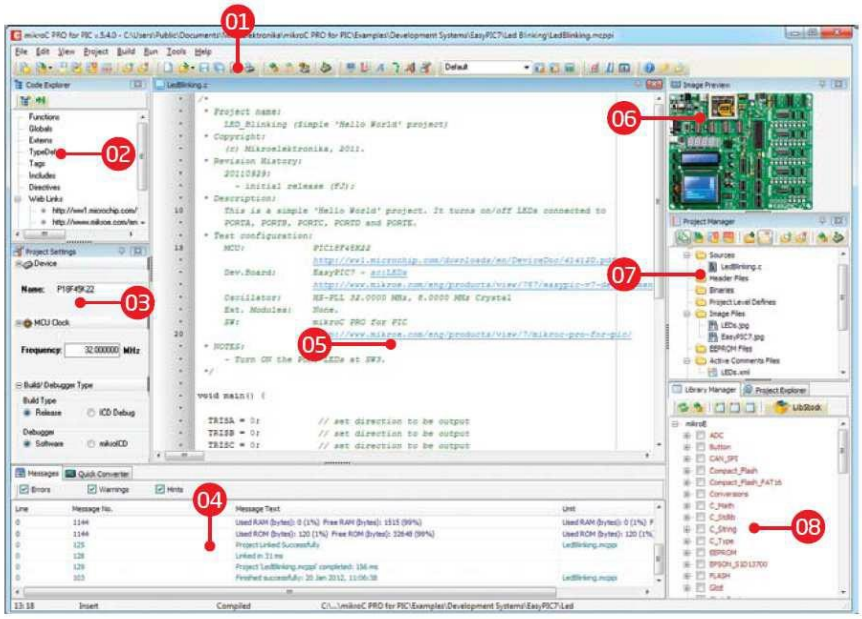
Un fichier "projet" contient les informations suivantes :

- le nom du projet et éventuellement une description,
- le µC utilisé,
- la fréquence d'horloge,
- la liste des fichiers sources,
- des fichiers binaires etc.

A présent, nous allons créer un nouveau projet, écrire du code, le compiler et le tester. Le but est de charge une constante dans un registre de RAM de notre µC.

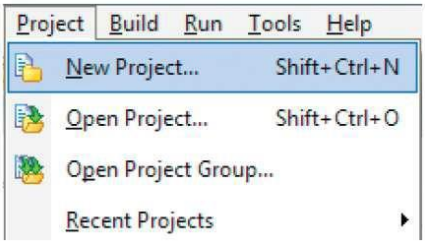
A-Double cliquer sur l'icône du compilateur **MikroC** du menu Start ou sur le raccourci du bureau de votre ordinateur pour le faire démarrer.

L'environnement de travail intégré du compilateur s'affiche sur l'écran. Il contient plusieurs fenêtres que l'on peut afficher ou non. Il est également possible dès les déplacer ou encore de les réduire, ceci permet à chacun de configurer son propre espace de travail. Un arrangement possible est montré sur la figure suivante :



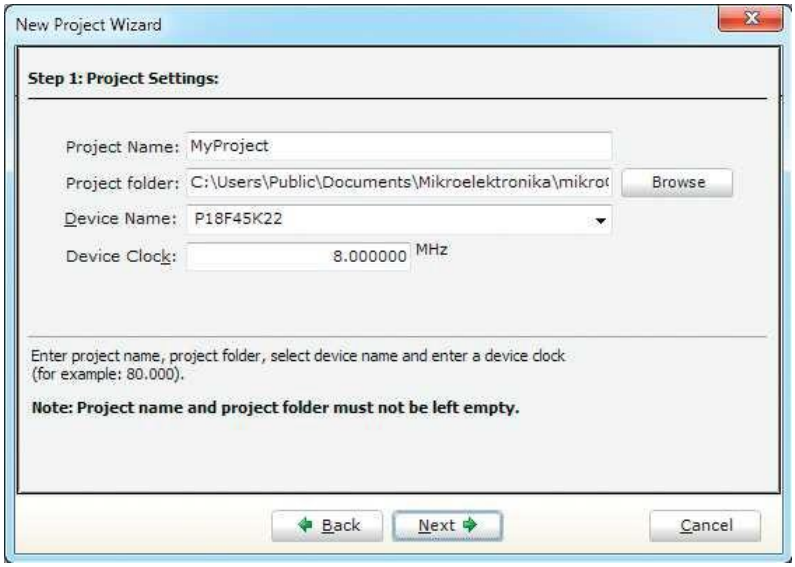
01 Main Toolbar
02 Code Explorer
03 Project Settings
04 Messages
05 Code Editor
06 Image Preview
07 Project Manger
08 Library Manager

B-Vous allez maintenant pouvoir démarrer un nouveau projet. Sélectionnez l'option "**New Project**" dans le menu Project ou cliquez directement sur l'icône "New Project" dans la barre d'outils "Project". Il ne reste plus qu'à se laisser guider, appuyer sur le bouton "Next"



C-Configuration du projet (Project Settings)

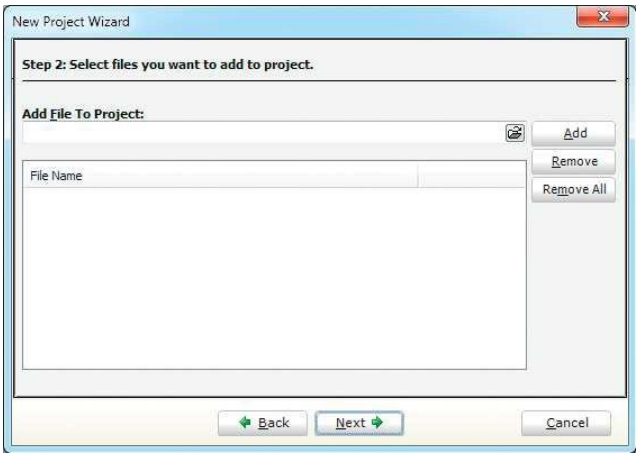
La première des choses à faire est de spécifier des informations générales sur le projet, son nom, son emplacement, le type de µC et la fréquence de l'horloge (8 MHz).



Modifier le nom du projet (GRAM) et le chemin du projet (cliquer sur Browse puis, sur le bureau, créer un répertoire TP1/GRAM). Le µC renseigné par défaut étant le PIC 18F45K22, ne pas modifier la case Device Name. Idem pour l'horloge de 8 MHz. Puis cliquer sur Next.

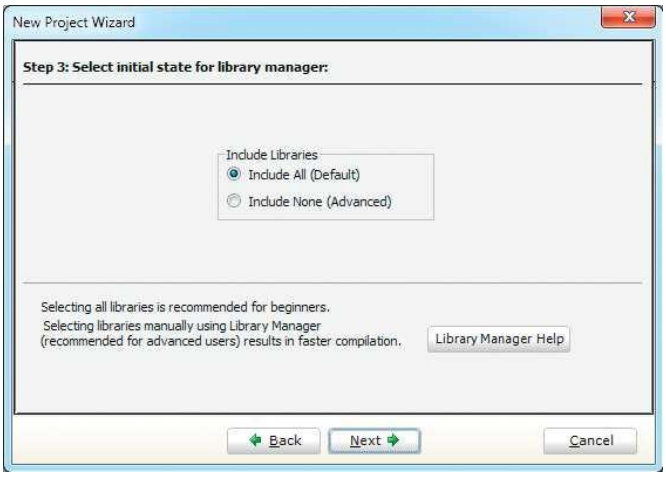
D-Ajout de fichiers (Add files)

Si vous devez intégrer des fichiers déjà existants dans votre projet, vous pouvez le faire à ce stade. Ce n'est pas encore le cas ici, contentez-vous de cliquer sur Next.



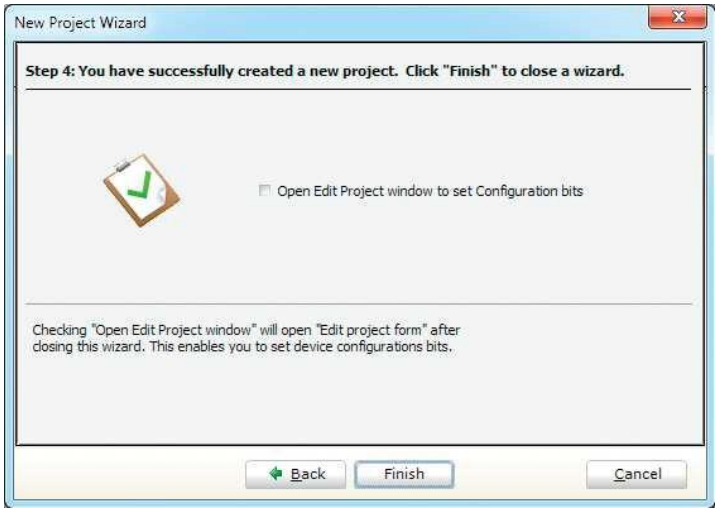
E-Inclure les librairies (Include Libraries)

Cette étape vous permet d'inclure ou non toutes les librairies dans votre projet. Le fait d'inclure toutes les librairies ne sera pas pénalisant au niveau de mémoire. En effet seules les librairies explicitement appelées par le programme seront activées. Au total, ce sont 500 fonctions qui peuvent être appelées dans le code. Elles peuvent être visualisées dans le "Code Assistant", [CTRL+space].

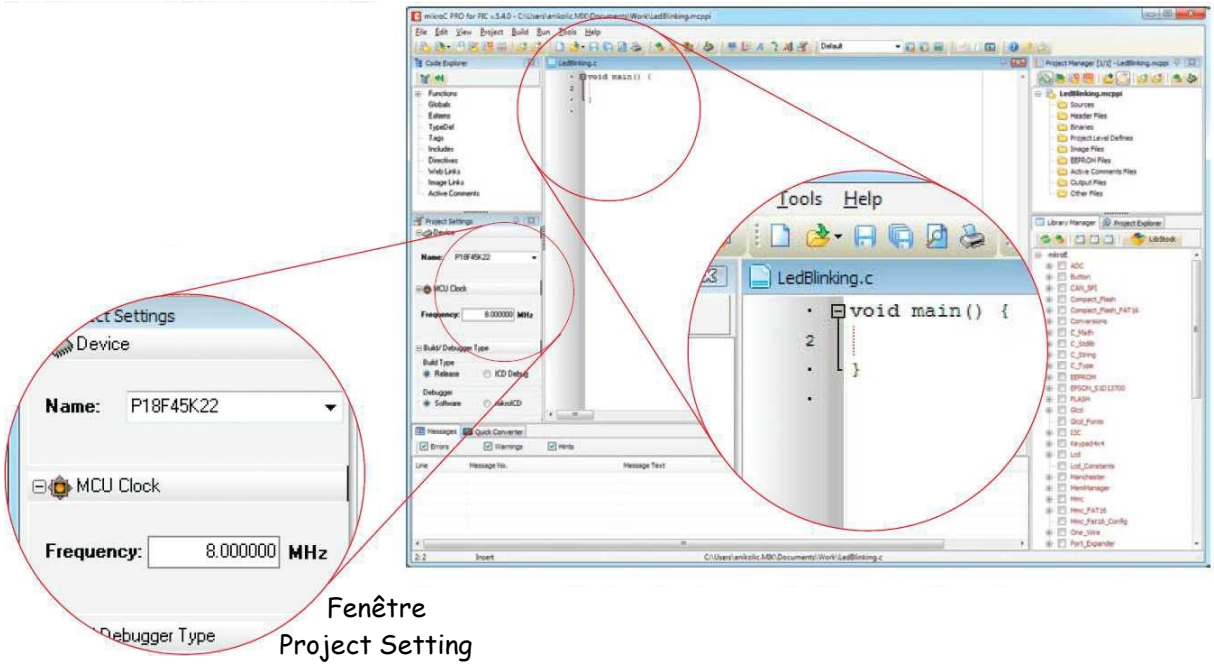


Inclure toutes les librairies et cliquer sur Next. Puis Fin

La dernière fenêtre permet de configurer la source d'horloge et la PLL ainsi que quelques bits de configuration. Nous utiliserons ici la configuration par défaut (oscillateur HS et PLL inactive), ne pas cocher la case et cliquer sur Finish.



Votre nouveau projet vient d'être créé. Il inclue un fichier source appelé "GRAM.c" qui contient la fonction principale void main(). Vous remarquerez également que votre projet est configuré avec les paramètres que vous avez renseignés précédemment.



Fenêtre Project Setting

3. Exemple de code

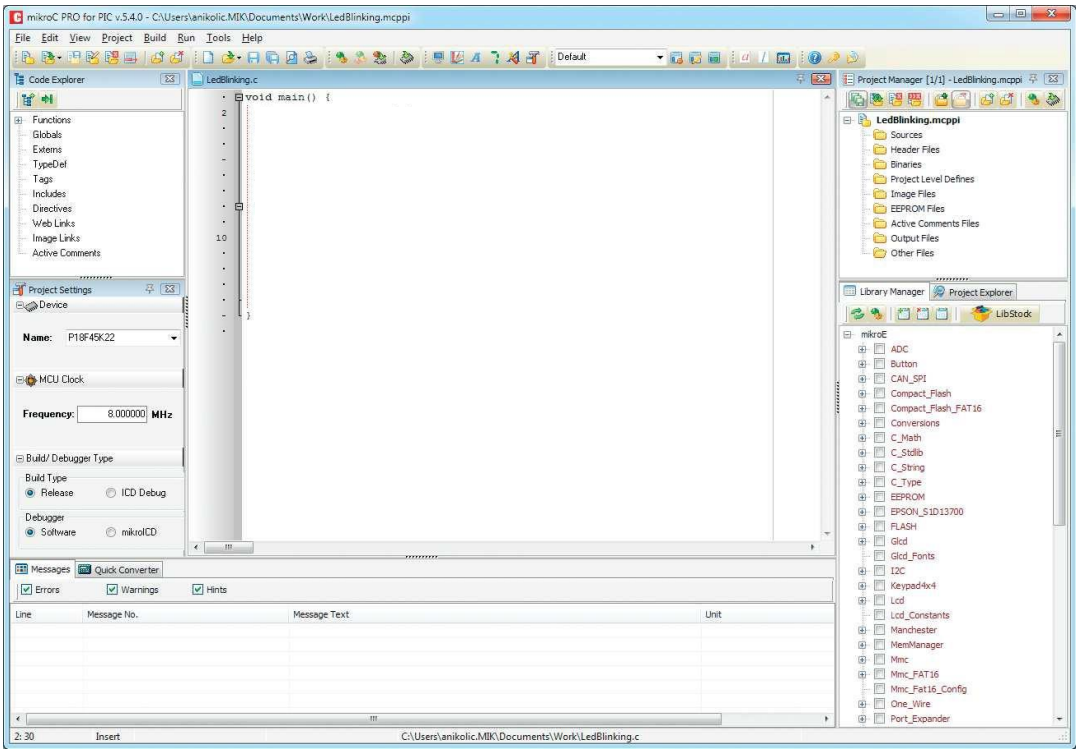
Nous allons maintenant pouvoir écrire notre premier programme en C. Nous initialisons un registre (par exemple R0, case mémoire d'adresse 0x000) avec des 1 sur tous les bites :

```
R0=0xFF ;
```

Finalement, dans une boucle while(), nous allons affecté périodiquement la valeur au registre R0, et initialise sa valeur à zéro.

```
while (1) {
R0=0xFF ;
R0=0x00 ;
R0=0xF0 ;
R0=0x0F;
}
```

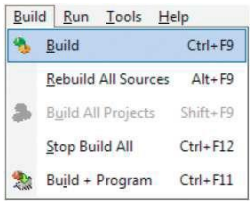
Tapez ce code dans la fenêtre principale, puis enregistrer.



Remarque : Si vous devez afficher une fenêtre particulière, suivez le chemin View et activer la ligne correspondante. Les fenêtres peuvent être insérées directement dans l'espace de travail principal ou réduites sur les côtés (épingles).

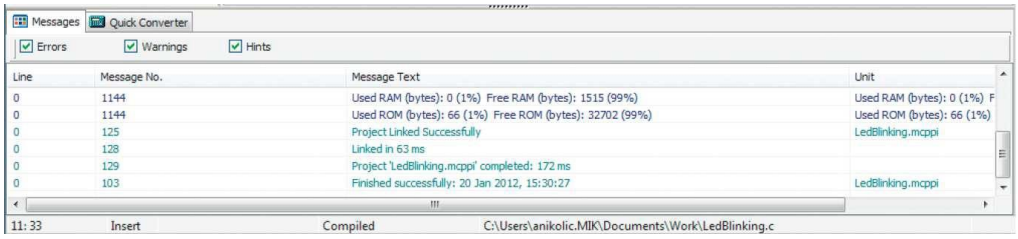
4. Compilation

A présent, nous allons compiler le projet afin de créer le fichier .hex qui sera chargé dans le µC. La compilation inclue ici la compilation à proprement parler (génération d'un code machine par fichier), l'édition des liens ou linking (lien entre fichiers et bibliothèques) et l'optimisation, tâches qui seront faite de manière automatique.



Pour compiler le projet, cliquer soit sur l'icône de la barre des tâches  ou dans le menu "Build", cliquer sur Build [CTRL+F9].

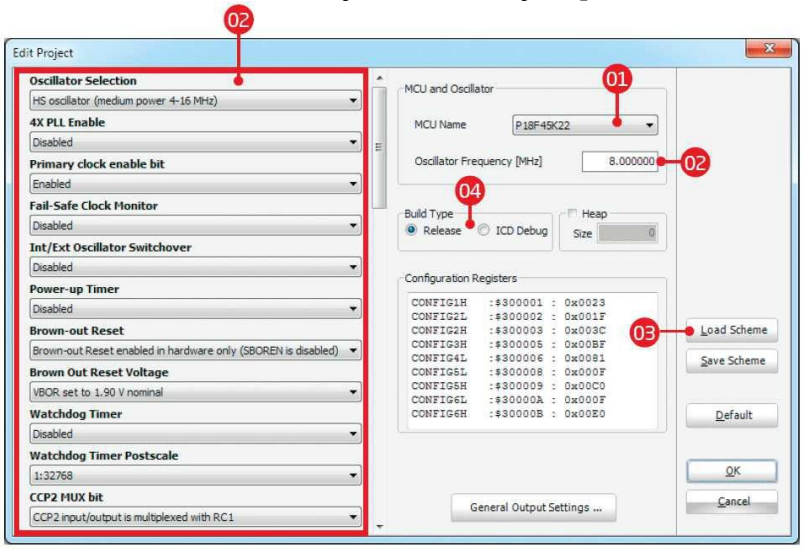
La fenêtre "message", si elle est activée, contient des détails sur le résultat de la compilation. Le compilateur créé automatiquement les fichiers de sortie, dont le fichier GRAM.hex.



5. Modification des paramètres

Si vous devez changer certains paramètres (type de µC, horloge), vous n'êtes pas dans l'obligation de reprendre tout le processus. Cela peut se faire très vite à partir de la fenêtre "Edit Project" :

Project › Edit Project [CTRL+SHIFT+E].



01 To change your MCU, just select the desired microcontroller from the dropdown list.

02 To change your settings enter the oscillator value and adjust configuration register bits using drop-down boxes.

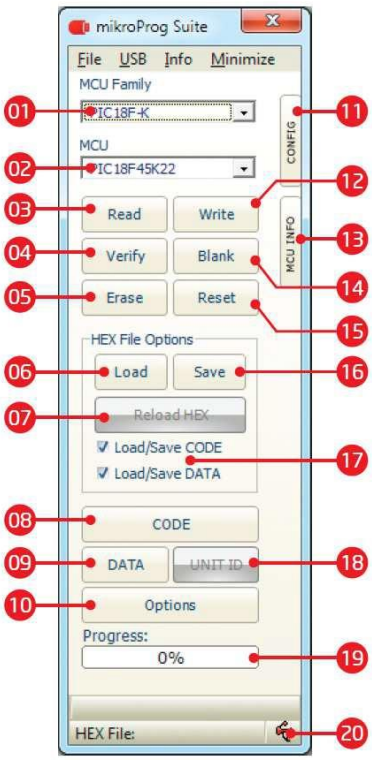
03 Several most commonly used settings can be loaded using the provided oscillator "schemes". Load the desired scheme by clicking the **Load Scheme** button.

04 Select whether to build a **Debug HEX**, which is necessary for hardware debugging, or a final **Release HEX**.

6. Programmation de la puce (µC)

Le programmateur mikroProg embarqué sur la platine de test fonctionne avec un programme spécial appelé mikroProg Suite for PIC. Pour le lancer, cliquer sur le raccourci situé sur le bureau de votre machine (icône représenté à droite)

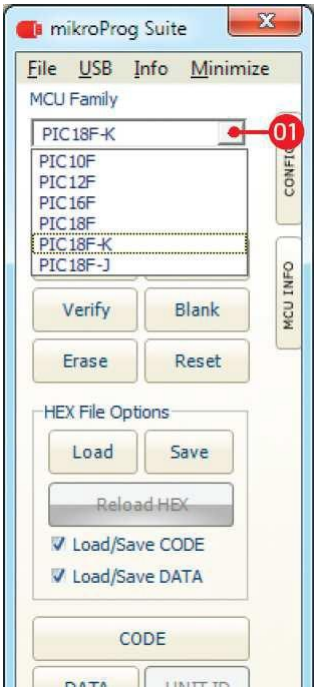
il faut lancer l'application en tant que administrateur !

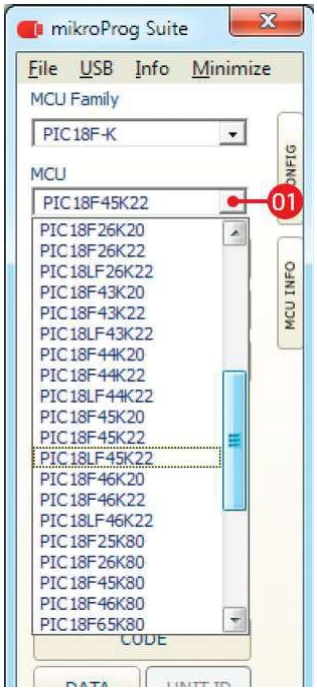


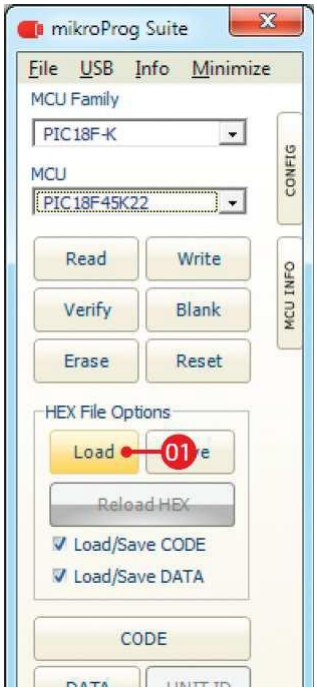
01 MCU family selection list
02 MCU type selection list
03 Read program from MCU
04 Verify the loaded program
05 Erase MCU memory contents
06 Browse for a .hex file on your PC
07 Reload previously loaded .hex file
08 Preview program which is in buffer and ready for uploading in MCU FLASH memory
09 Preview program which is in buffer and ready for uploading in MCU EEPROM memory
10 Various settings of visual, advanced and programming options.
11 Expand configuration bits menu
12 Upload .hex file in to MCU memory
13 Expand MCU info menu
14 Check whether the MCU is empty
15 Reset the microcontroller
16 Save buffer to a .HEX file
17 Load/Save CODE/DATA in buffer
18 Used for some MCU-s ID
19 Progress bar
20 Shows that programer is connected to USB port on a PC (red if connected)

Connecter maintenant le programmateur avec le câble USB fournit, allumer la platine de test (alimentation via le câble USB). L'icône représenté en bas à droite de la fenêtre doit passer au rouge.

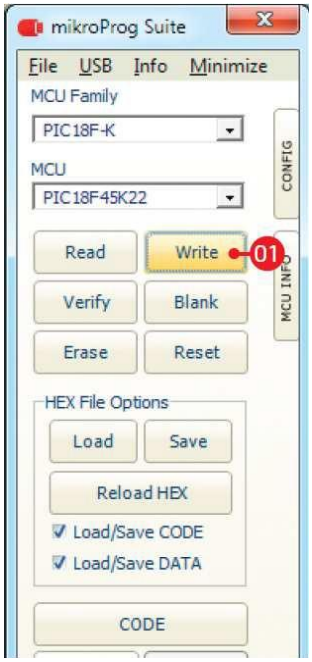
Afin de charger le fichier.hex, vous devez dans l'ordre sélectionner la famille du µC (PIC18F-K), ensuite le modèle (PIC18F45K22), sélectionner le fichierhex. (Load puis GRAM.hex).

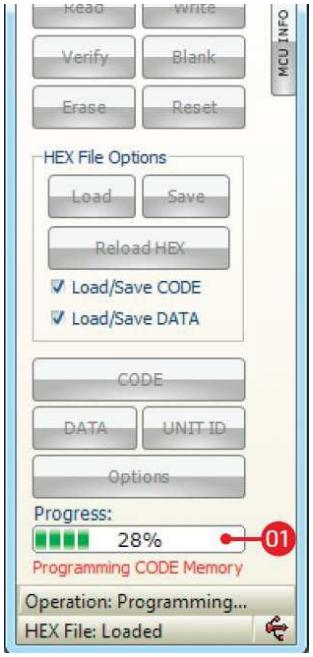






Cliquer sur le bouton Write, le chargement est terminé lorsque l'opération passe à None.





7. Exécution pas-à-pas, débogage

L'expérience le prouve, même le programme le plus soigneusement écrit peut ne pas se comporter comme attendu. Pour le faire fonctionner correctement, vous devez découvrir les erreurs qui se cachent dans le code. Le processus consistant à découvrir et à corriger des erreurs de programmation est appelé débogage.

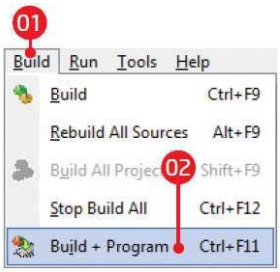
Il y a 2 manières de procéder, soit par simulation logicielle (simulation de ce qui est supposé se passer au niveau du µC lorsque les lignes s'exécutent), soit par test direct sur le circuit (In-Circuit debugging). Dans ce cas, il ne s'agit plus d'une simulation, le code s'exécute effectivement sur le composant, c'est-à-dire au niveau matériel.

Le programmateur embarqué MikroProg et le compilateur MikroC supportent mikroICD, un outil de débogage au niveau matériel. Il permet d'exécuter le code directement sur le µC et de vérifier les valeurs des variables, des registres spéciaux (SFR), du contenu de la mémoire RAM, programme ou EEPROM etc.

1. Revenir sous MikroC Pro for PIC et ouvert le projet s'est-il fermer. Le quartz sur la carte est toujours de 8 MHz, compiler le projet et charger le fichier GRAM.hex dans le µC.

2. Il vous faut maintenant autoriser le débogueur mikroICD. Revenir sous MikroC, suivi le lien "Project Settings – Build /Debugger Type", sélectionner l'option "ICD Debug" pour autoriser la création d'un fichier .hex de débogage puis l'option "mikroICD" afin d'autoriser l'utilisation du débogueur.

3. A présent, il nous faut recompiler et reprogrammer la puce. Pour cela, cliquer sur Build › Build + Program [CTRL+ F11] ou directement sur l'icône correspondante de la barre d'outils.



Via le menu Build

Via la barre d'outils

Le compilateur va alors directement compiler le projet et démarrer le logiciel mikroProg Suite for Pic.

4. Pour démarrer mikroICD, cliquer sur le menu Run puis sur Start Debugger [F9].

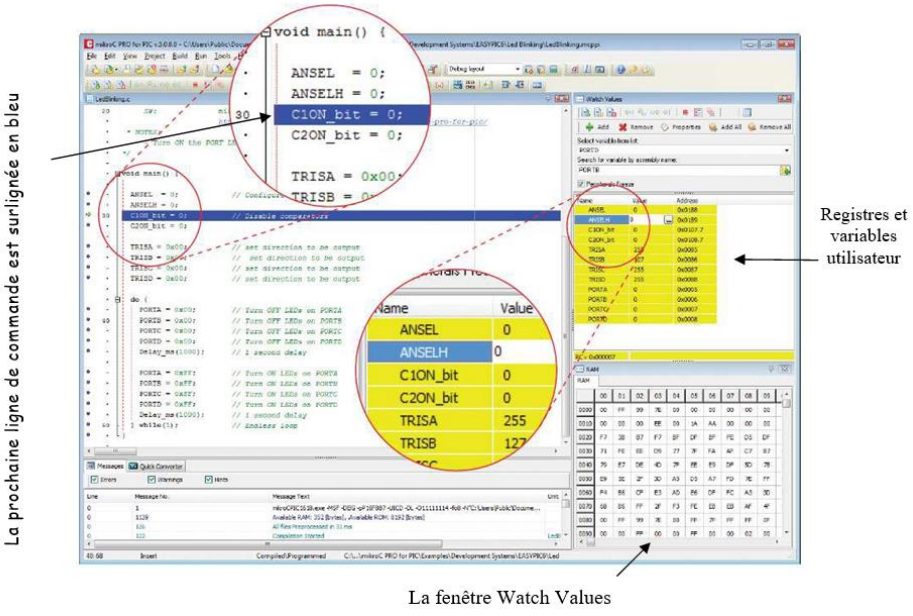
5. Pour afficher la fenêtre des variables à surveiller, cliquer sur View › Debug Windows › Watch Window [Shift+F5] ou directement sur l'icône de la barre d'outils.

Pour afficher la fenêtre de la RAM (Data) et visualiser le contenu de registre 145h, cliquer sur View › Debug Windows › RAM Window. Les valeurs affichées dans les cases mémoires sont exprimées en hexadécimal. Vous pouvez les modifier à tout moment, il suffit pour cela de rentrer une nouvelle valeur et d'appuyer sur la touche Enter de votre clavier.

De même, le contenu des mémoires Programme (Flash) et EEPROM peuvent être visualisées

Flash : View › Debug Windows > CODE Window
EEPROM : View › Debug Windows > EEPROM Window

Après avoir validé l'affichage des paramètres et si vous le souhaitez de la RAM, la fenêtre de MikroC doit ressembler à la figure suivante :



La fenêtre Watch Values

Dans cette fenêtre, vous pouvez décider quels registres ou variables vont être surveillés. Durant l'exécution du code, les valeurs éventuellement impactées vont évoluer en temps réel.

Différentes couleurs permettent de repérer le type de variable, en mauve pour les registres SFR et les sbits et en noir pour les variables utilisateur ou celles utilisées par les librairies.

La barre d'outils du débogueur

Debug commands

The first three icons on the toolbar are used for starting/stopping debugger:

- Start debugger [F9]
- Run/Pause Debugger [F6]
- Stop Debugger [Ctrl + F2]

Execution commands

Next set of icons enables you to execute program in real time:

- Step Into [F7]
- Step Over [F8]
- Step Out [Ctrl + F8]
- Run To Cursor [F4]

Managing breakpoints

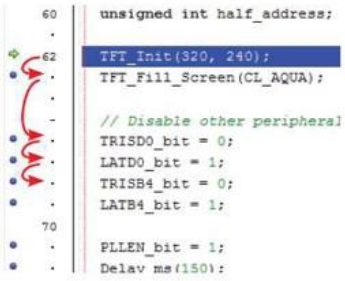
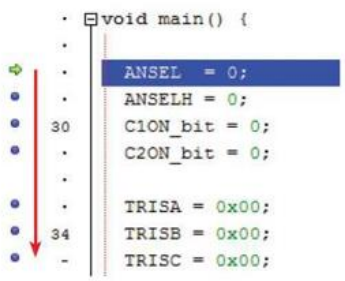
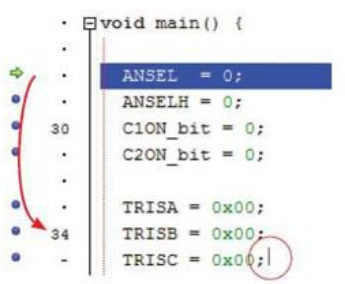
Last set of icons is related to breakpoints and interrupt option:

- Toggle Breakpoint [F5]
- Show/Hide breakpoints [Shift+F4]
- Clears breakpoints [Shift+Ctrl+F5]
- Jump to interrupt [F2]

Toolbar Icon	Name	Shortcut	Description
	Start Debugger	[F9]	Starts Debugger.
	Run/Pause Debugger	[F6]	Run/Pause Debugger.
	Stop Debugger	[Ctrl + F2]	Stops Debugger.
	Step Into	[F7]	Executes the current program line, then halts. If the executed program line calls another routine, the debugger steps into the routine and halts after executing the first instruction within it.
	Step Over	[F8]	Executes the current program line, then halts. If the executed program line calls another routine, the debugger will not step into it. The whole routine will be executed and the debugger halts at the first instruction following the call.
	Step Out	[Ctrl + F8]	Executes all remaining program lines within the subroutine. The debugger halts immediately upon exiting the subroutine.
	Run To Cursor	[F4]	Executes the program until reaching the cursor position.
	Toggle Breakpoint	[F5]	Toggle breakpoints option sets new breakpoints or removes those already set at the current cursor position.
	Show/Hide breakpoints	[Shift+F4]	Shows/Hides window with all breakpoints
	Clears breakpoints	[Shift+Ctrl+F5]	Deletes selected breakpoints
	Jump to interrupt	[F2]	Opens window with available interrupts (doesn't work in mikroICD™ mode)

Débogage en temps réel

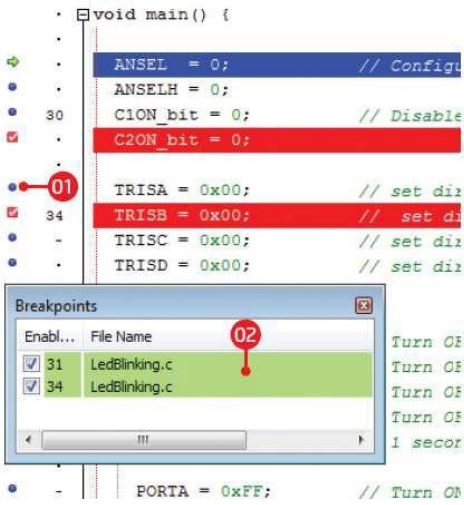
Il existe trois possibilités pour faire du débogage en temps réel :

Step by Step	Execute remaining lines	Execute to cursor
		
Pour exécuter le programme ligne par ligne : Step Into [F7] Step Over [F8]	Pour exécuter l'ensemble des lignes restantes : Step Out [Ctrl+F8]	Pour exécuter les lignes jusqu'à l'emplacement du curseur : Run to Cursor [F4]

Les points d'arrêt (breakpoint)

MikroICD permet l'utilisation des points d'arrêts. Cela signifie que vous pouvez marquer une ou plusieurs lignes particulières du programme afin de stopper ou de mettre en pause son exécution à des fins de test.

Les points d'arrêt sont placés en cliquant sur les points bleus à gauches des lignes de code ou sur l'icône [F6]. Dans ce cas, le µC exécute le programme à partir de la ligne active, surlignée en bleu, jusqu'à la ligne marquée par un point d'arrêt, surlignée en rouge. A ce stade, le débogueur fait une pause.



Remarque : il existe des points d'arrêt matériels et logiciels (cf. guide de l'utilisateur de mikroICD).

8. Manipulation

- Suivi pas à pas l'exécution du programme (Step Into par exemple).
- Vérifier la valeur de registre.
- Ajouter le suivi des registre 145h lorsque vous êtes en mode pas à pas.
- Contrôler la bonne évolution des valeurs de ce variable.
- Expérimenter les autres possibilités du débogueur.

9. Exercices :

Exercice 1:

Donner le programme permet de copier l'alphabet majuscule ‘A’ dans la RAM a la position R0.

Exercice 2 : Accès à la RAM par pointeur

1. Donner le programme permet de copier la valeur 040h, dans 10 case mémoire à partir de l’adresse R0,
2. Donner le programme qui copie les valeurs 0,1... 10, dans les cases mémoires à partir de l’adresse 0x90.