Utilisation de MPLAB X

Introduction

Tous les outils logiciels nécessaires à la programmation du contrôleur de robot sont regroupés dans un environnement de développement intégré (*IDE : **I**ntegrated **D**evelopment **E**nvironment*) appelé *MPLAB X*.

Un *IDE* a pour but d'être l'interface unique par laquelle le programmeur peut accéder à tous les outils dont il a besoin :

- un éditeur de texte, qui permet d'écrire le code source ; il est spécialement adapté à cette tâche : par exemple; le texte est coloré pour le rendre plus lisible, l'indentation du texte (ajout de tabulation pour montrer la hiérarchie des boucles) est automatique, ...
- un gestionnaire de projet, qui permet de gérer et d'accéder aux fichiers composant le projet.
- une fenêtre de sortie, dans laquelle les messages d'informations et d'erreurs des outils s'affichent.
- un tableau de bord qui résume les informations de configuration du projet.
- des barres d'outils et des menus permettant de lancer les autres outils : le compilateur, le debugger, le simulateur et le programmateur.

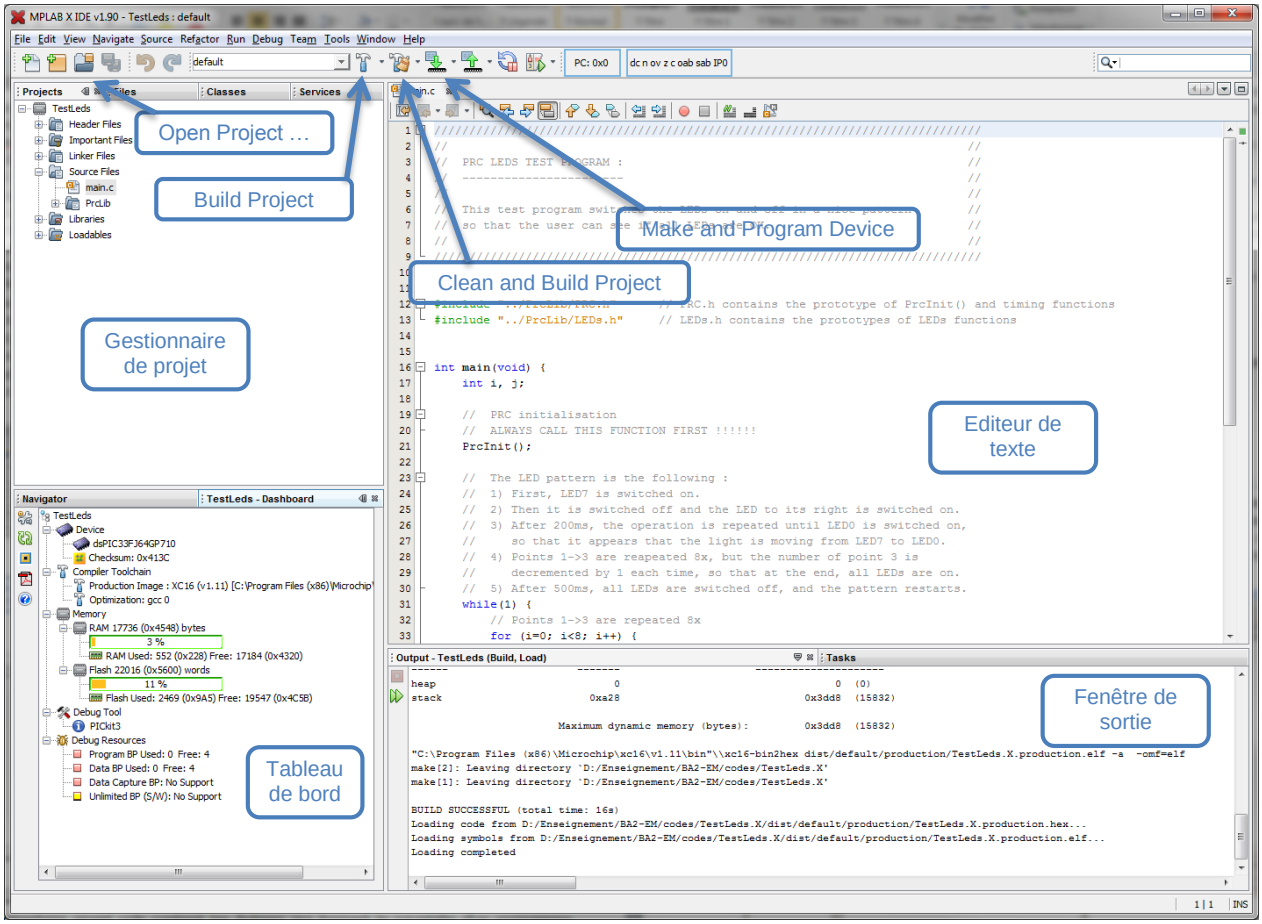


Figure 1 : Fenêtre principale de *MPLAB X*

Démarrer un nouveau projet

EmptyProj

Pour vous éviter de devoir configurer entièrement votre projet, nous avons créé un projet adapté à l'Explorer16. Ce projet est situé dans le répertoire *EmptyProj.*
Le projet contient un fichier *main.c* qui est le fichier principal du projet, il contient un squelette de fonction *main()*, contenant le minimum vital pour initialiser l'Explorer16.

Copie d'un projet existant

Pour créer votre projet :

- Ouvrez le projet sur lequel vous voulez vous baser (*EmptyProj* par exemple).
- Cliquez-droit sur le nom du projet dans le *gestionnaire de projet*, et choisissez *copy…* dans le menu déroulant ;
- Dans la fenêtre de dialogue, donnez un nom à votre projet :

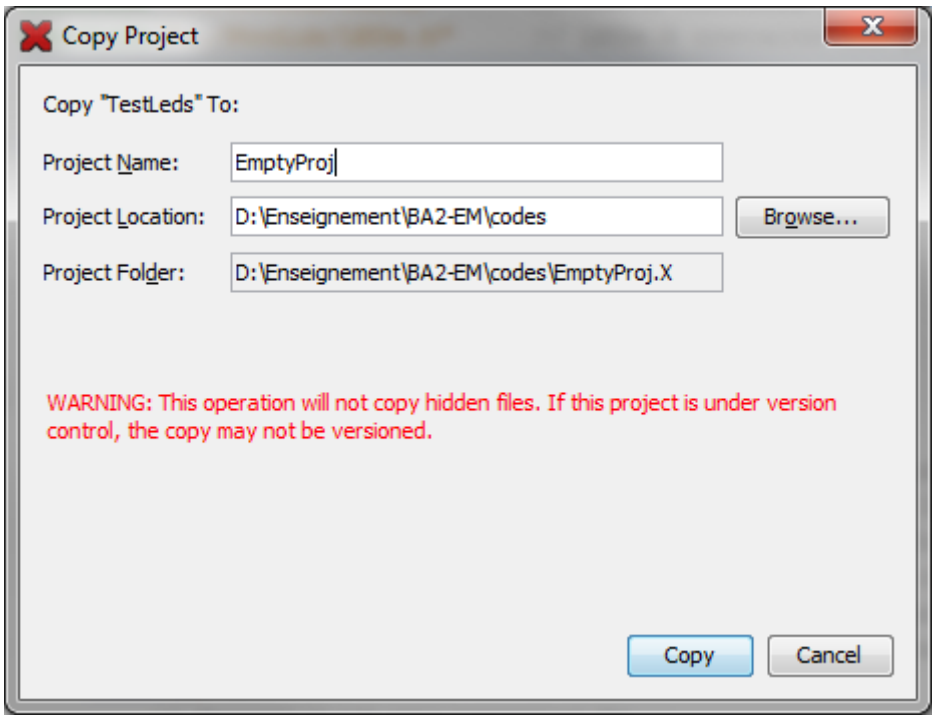


Figure 2 : Boite de dialogue *Copy Project*

Remarquez que vous avez maintenant deux projets ouverts. *MPLAB X* supporte cela sans problème, mais nous vous conseillons de n'avoir qu'un seul projet ouvert, pour éviter les confusions.

Compiler un projet

Pour compiler votre programme, il existe deux commandes :

- *Build Project*, qui permet de ne compiler que les fichiers sources qui ont été modifiés depuis la dernière compilation. Pour cela, le compilateur compare les dates des fichiers sources avec celles des fichiers compilés correspondants. Cette commande permet d'accélérer la compilation des projets où il y a plusieurs fichiers sources dont certains ne sont pas modifiés souvent (comme les fichiers des périphériques).
- *Clean and Build Project*, qui efface d'abord tous les fichiers compilés, obligeant le compilateur à compiler tous les fichiers sources. Cela prend évidemment plus de temps, mais permet de s'assurer que tous les fichiers sont correctement compilés. En effet, comme *Build project* se base sur les dates des fichiers pour décider

lesquels compiler, il se peut qu'il se trompe (par exemple si vous remplacer un fichier source par une version plus ancienne après vous être rendu compte que vos dernière modifications n'étaient en fait pas une si bonne idée).

Corriger les erreurs de compilation

Erreurs lors de l'écriture

MPLAB X analyse constamment votre code et vous indique vos erreurs sans que vous ayez besoin de compiler votre projet. Dans l'exemple de la *Figure 3*, il indique qu'il ne s'attend pas à trouver le mot réservé *while* à cet endroit. Cette erreur est en fait due à l'absence d'un ';' à la fin de la ligne 25.

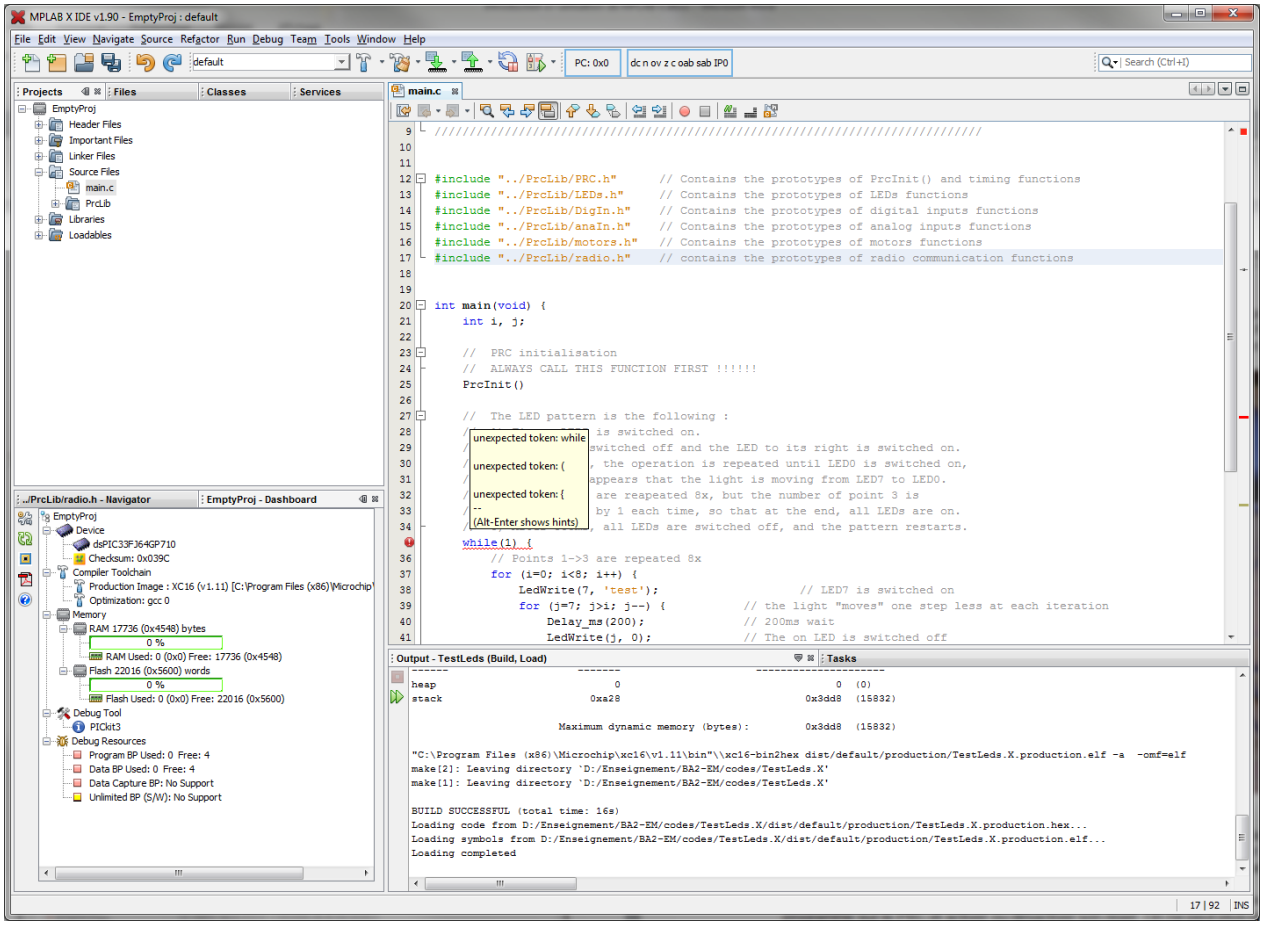


Figure 3 : Exemple de détection d'erreur à l'écriture

Grace à cette correction permanente, vous corrigerez toutes les erreurs de syntaxe avant d'essayer de compiler votre code.

Erreurs lors de la compilation

Toutefois, *MPLAB X* ne peut pas détecter toutes les erreurs sans tenter de compiler votre code. Dans l'exemple de la *Figure 4*, La fonction *ErrFunction()* est déclarée au début de *main.c* et appelée dans la fonction *main()*. Par contre, elle n'est définie nulle part. *MPLAB X*

ne détecte pas ce type d'erreur avant la compilation, car il suppose que la fonction est définie dans un autre fichier '.c' du projet.

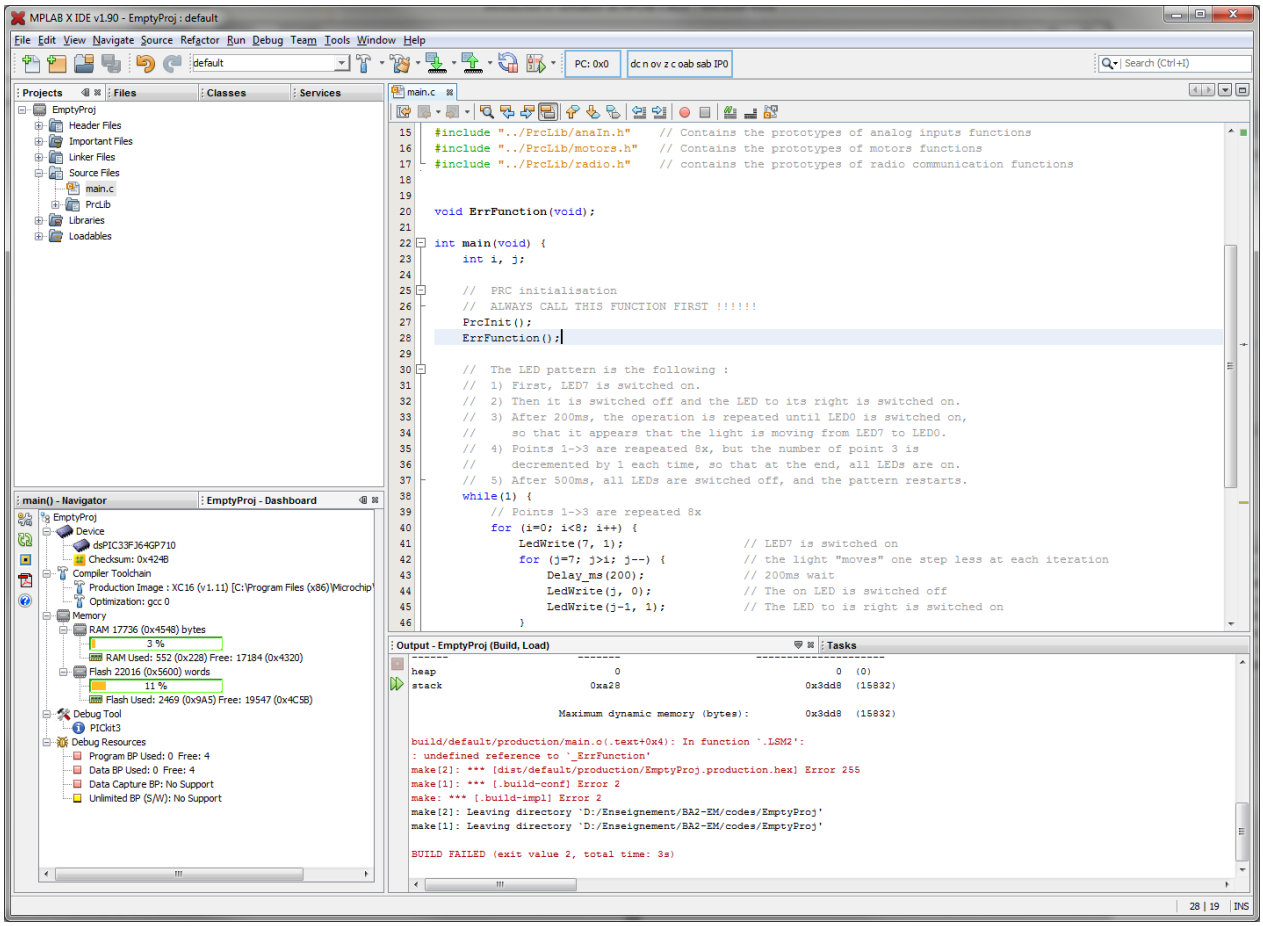


Figure 4 : Exemple de détection d'erreur à la compilation

De plus, il arrive qu'il détecte quelque chose de suspect, mais qui n'est pas nécessairement une erreur ; dans ce cas, il vous laisse le bénéfice du doute et compile votre code sans provoquer d'erreur. Néanmoins, il vous indique ce qu'il a détecté sous forme d'un *Warning* dans la fenêtre de sortie.

Dans l'exemple de la *Figure 5*, il a remarqué que la variable *err* n'est pas initialisée avant d'être utilisée comme argument de la fonction *LedWrite*, ce qui provoquera un comportement indéterminé du PRC. Toutefois, il ne provoque pas d'erreur et compile le code, au cas où nous aurions omis d'initialiser *err* intentionnellement.

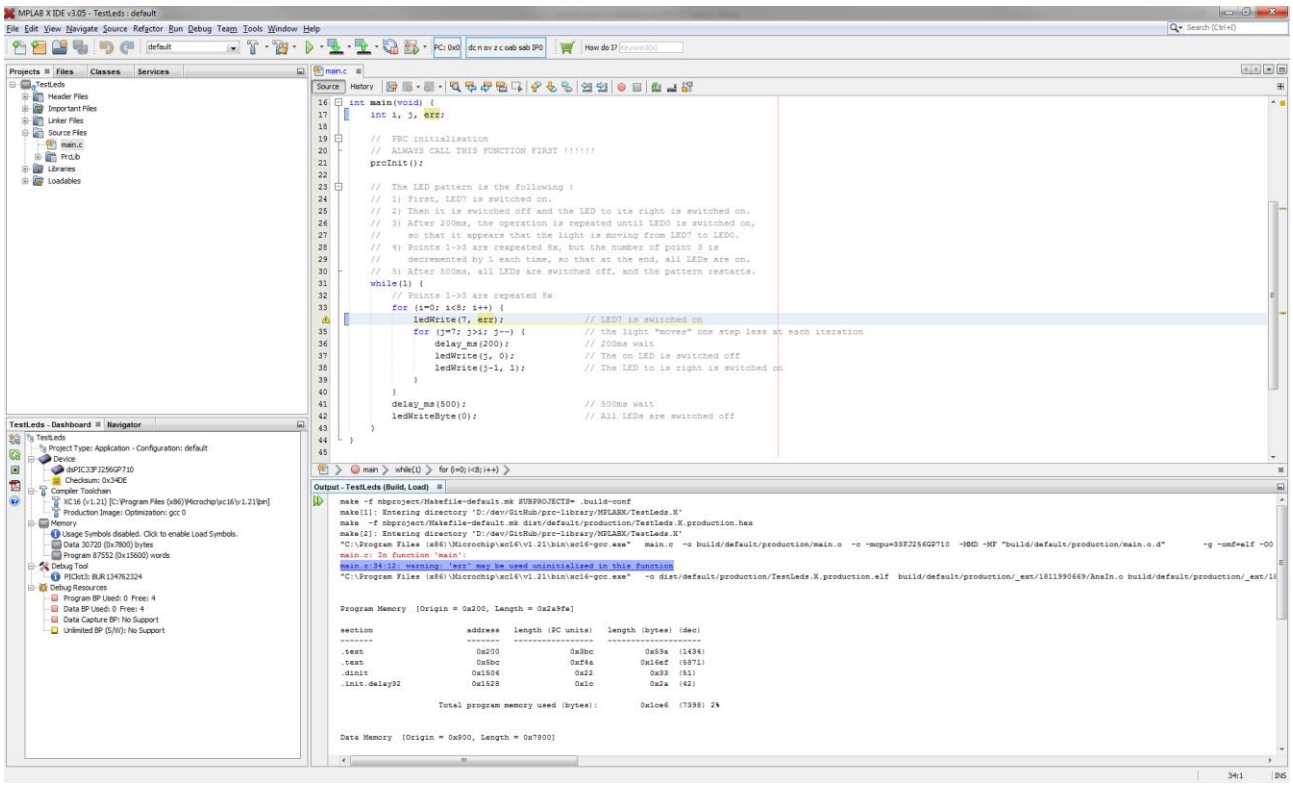


Figure 5 : Exemple de *Warning*

Exécuter le programme sur le PRC

Pour transférer le programme compilé sur le PRC, on doit utiliser un dispositif d'interface appelé PicKit3. Il se connecte sur un port USB du PC et sur le connecteur de programmation du PRC (connecteur à 5 pattes situé sur un de ses côtés).
Le PicKit3 peut fonctionner dans deux modes :

- Le mode *Debugger* : dans ce mode, l'exécution du programme est contrôlée par *MPLAB X*, on peut le démarrer, le mettre en pause, l'exécuter pas-à-pas, utiliser des *breakpoints*, observer l'état des variables, … Lorsque le PRC est programmé dans ce mode, il ne peut fonctionner sans être connecté au PC.
- Le mode *Programmer* : dans ce mode, le PicKit3 sert uniquement à transférer le programme sur le PRC. On ne peut plus utiliser les outils de debug, mais le PRC peut fonctionner seul.

Utilisation en mode *Programmer*

Pour télécharger votre programme dans le PRC, vous devez :

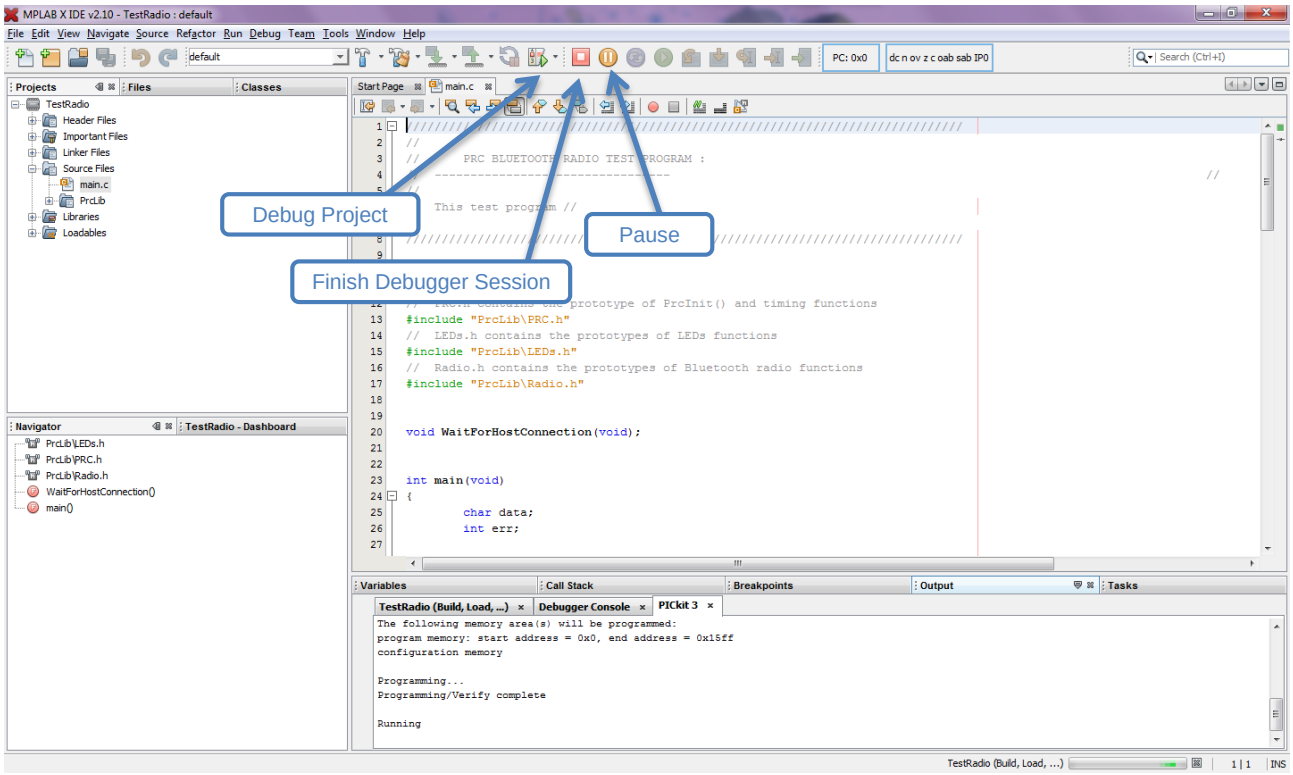
- Connecter le PRC au PC au moyen du PicKit3
- Alimenter le PRC au moyen de l'alimentation fournie ou de la batterie
- Cliquer sur Make and Program Device

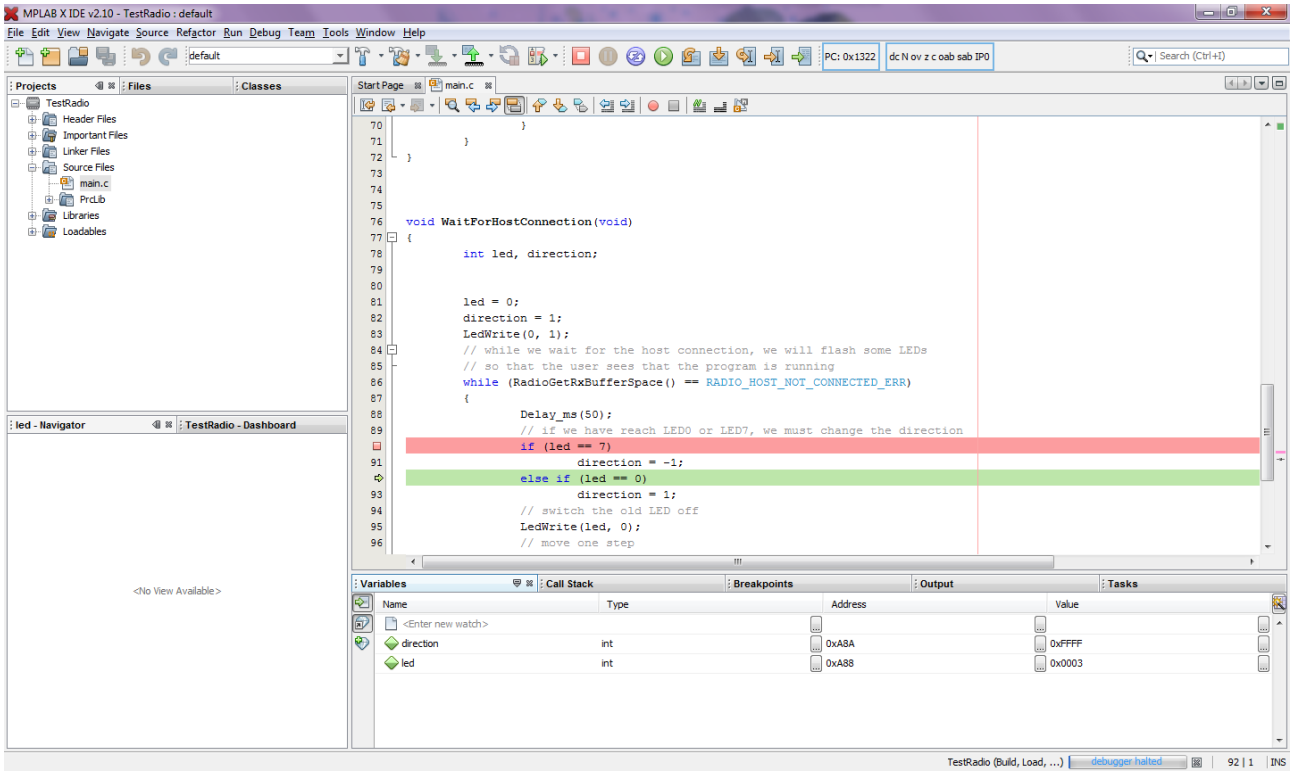
Lorsque le PRC est programmé, votre programme commence à s'exécuter. Vous pouvez déconnecter le PicKit3.
Le PRC stocke votre programme dans une mémoire non volatile (du même type que celle des clés UCB) ; vous pouvez donc retirer l'alimentation du PRC et la rebrancher plus tard sans devoir le reprogrammer.
Pour que vous puissiez décider du moment où l'exécution du programme débute, le kit contient un interrupteur que vous pouvez brancher sur le connecteur de reset du PRC (connecteur à 2 pattes à côté du connecteur de programmation). Il vous permet alors de contrôler le *Reset* du PRC.

ATTENTION : ne connectez pas l'interrupteur de Reset et le PicKit3 au PRC simultanément, l'interrupteur peut perturber le fonctionnement du PicKit3.

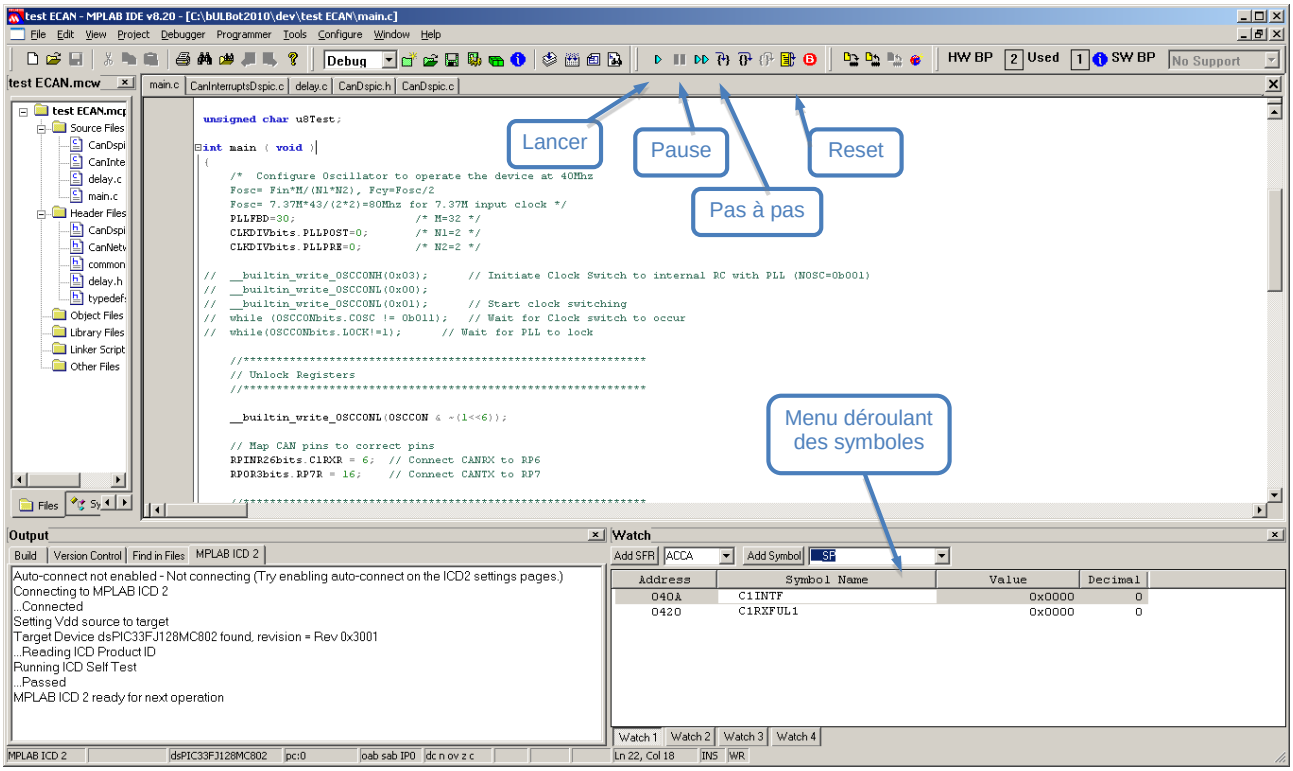
Utilisation en mode *Debugger*

Programmer le PRC et démarrer le programme





En appuyant sur le bouton Play, le programme est exécuté par le processeur. Il est possible de l'interrompre à tout moment avec le bouton pause.
Lorsque le programme est en pause, il est possible de visualiser l'état des variables dans la *watch window*. Pour cela, il suffit de sélectionner le nom de votre variable dans le menu déroulant des symboles, puis de cliquer sur *Add Symbol*.
Il est également possible de faire avancer le programme ligne par ligne avec le bouton pas-à-pas, ce qui facilite la recherche d'erreur.
Le bouton *Reset* permet de revenir au début du programme, pour pouvoir recommencer son exécution depuis le début.

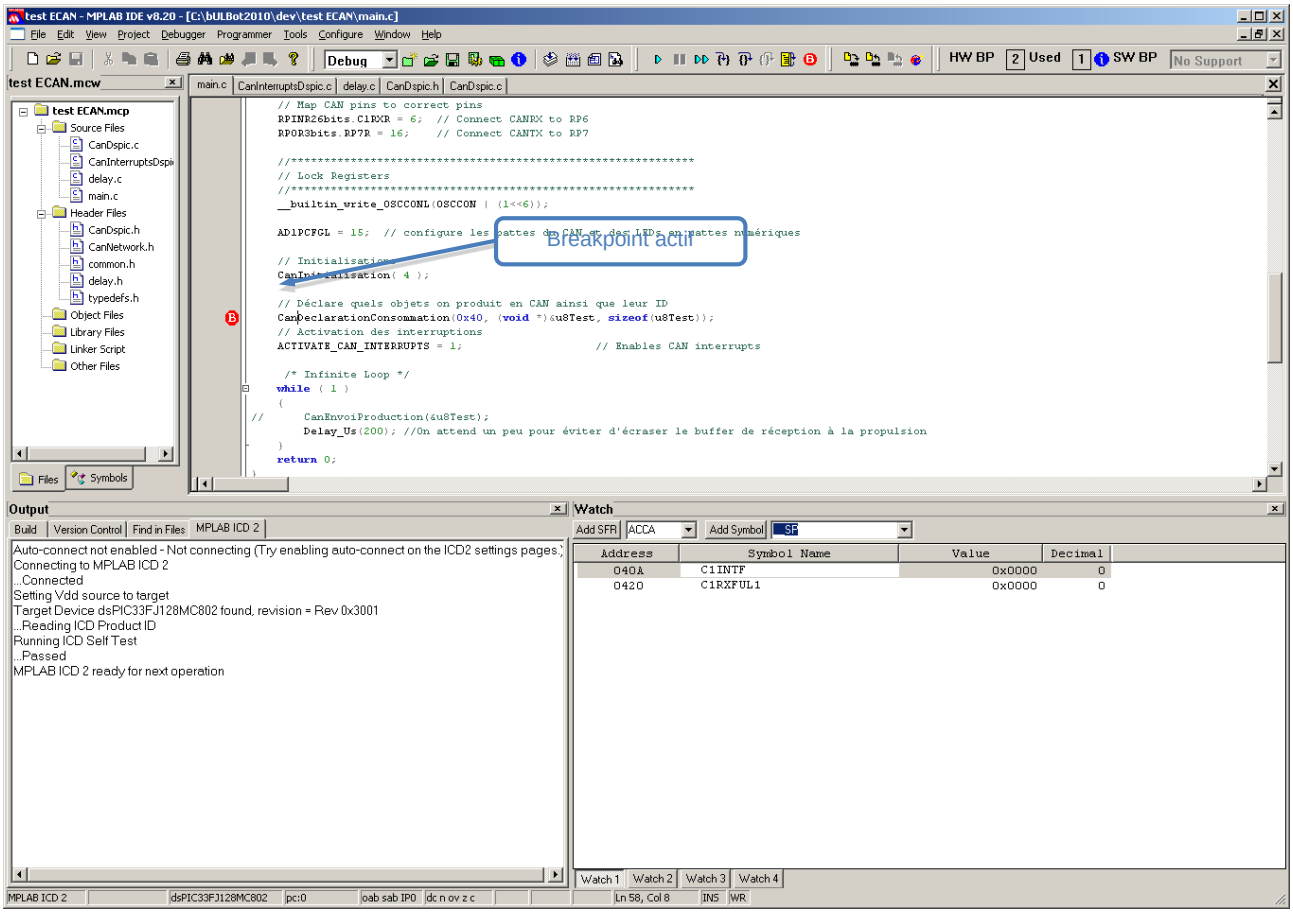


Debugger le programme avec les breakpoints

Repérer des erreurs de programmation n'est pas chose facile avec un microcontrôleur car les commandes du type *cout* et *cin* n'existent pas. C'est pourquoi l'on utilise très souvent les *breakpoints*.
Le principe est le suivant : lorsque le programme exécute une ligne où se trouve un breakpoint, le processeur se met automatiquement en pause. Il est dès lors possible de visualiser l'état de toutes les variables et ainsi de vérifier le bon fonctionnement du

programme. C'est également une manière simple de s'assurer que le programme exécute bien certaines lignes.

Pour ajouter un breakpoint sur une ligne, il suffit de double cliquer dessus.



Remarque : Normalement, le programme s'arrête avant d'exécuter la ligne où se situe le *breakpoint.* Toutefois, il arrive qu'il s'arrête sur la ligne d'après.

Pour sélectionner l'ICD2 en mode *programmer*, il faut sélectionner le menu *Programmer/Select Tool/MPLAB ICD 2*. L'IDE se modifie en conséquence (voir les figures ci-dessous). En particulier de nouvelles barres d'outils apparaissent.