

La programmation des PIC en C

Installation des programmes et première simulation

Réalisation : HOLLARD Hervé.
<http://electronique-facile.com>
Date : 29 juillet 2003
Révision : 1.2

Sommaire

Sommaire	2
Introduction	3
Structure de ce document.....	4
Le matériel nécessaire.....	4
Installation des deux logiciels	4
Installation de MPLAB 6.3.....	4
Modification lors de l'installation de version postérieure à 6.42.....	4
Installation de CC5X	4
Déclaration du compilateur CC5X dans MPLAB.....	5
Création d'un projet.....	6
Déclaration des options	7
Choix du composant	7
Configuration des bits d'option	7
Déclaration du répertoire des fichiers inclus	8
Choix de l'option de simulation.....	8
Choix de la fréquence du quartz pour la simulation	8
Sauvegarde du projet.....	8
Edition et compilation.....	9
Simulation	12

Introduction

Les microcontrôleurs PIC de la société Microchip, sont depuis quelques années dans le "hit parade" des meilleures ventes. Ceci est en partie dû à leur prix très bas, leur simplicité de programmation, les outils de développement que l'on trouve sur le NET.

Aujourd'hui, développer une application avec un PIC n'a rien d'extraordinaire, et tous les outils nécessaires sont disponibles gratuitement. Voici l'ensemble des matériels qui me semblent les mieux adaptés.

Ensemble de développement (éditeur, compilateur, simulateur) :

MPLAB de MICROCHIP <http://www.microchip.com>

Logiciel de programmation des composants:

IC-PROG de Bonny Gijzen <http://www.ic-prog.com>

Programmeur de composants:

PROPIC2 d'Octavio Noguera voir notre site <http://electronique-facile.com>

Pour la programmation en assembleur, beaucoup de routines sont déjà écrites, des didacticiels très complets et gratuits sont disponibles comme par exemple les cours de **BIGONOFF** dont le site est à l'adresse suivante <http://abcelectronique.com/bigonoff>.

Les fonctions que nous demandons de réaliser à nos PIC sont de plus en plus complexes, les programmes pour les réaliser demandent de plus en plus de mémoires. L'utilisateur est ainsi à la recherche de langages "évolués" pouvant simplifier la tâche de programmation.

Depuis l'ouverture du site <http://electronique-facile.com>, le nombre de questions sur la programmation des PIC en C est en constante augmentation. Il est vrai que rien n'est aujourd'hui disponible en français.

Mon expérience dans le domaine de la programmation en C due en partie à mon métier d'enseignant, et à ma passion pour les PIC, m'a naturellement amené à l'écriture de ce **didacticiel**. Celui-ci se veut **accessible à tous** ceux qui possèdent une petite expérience informatique et électronique (utilisation de Windows, connaissances minimales sur les notions de base en électronique comme la tension, le courant, les résistances, les LEDS...).

Je ne **traiterai ici que la programmation et la simulation**. Si vous désirez essayer vos œuvres avec un microcontrôleur, il vous faudra réaliser tout seul un programmeur de composant et trouver un logiciel de programmation.

Structure de ce document

Ce document est composé de chapitres. Chaque chapitre dépend des précédents. Si vous n'avez pas de notion de programmation, vous devez réaliser chaque page pour progresser rapidement.

Le type gras sert à faire ressortir les termes importants.

Vous trouverez la définition de chaque **terme nouveau** en **bas de la page** où apparaît pour la première fois ce terme. *Le terme est alors en italique.*

La couleur **bleue** est utilisée pour vous indiquer que ce **texte est à taper** exactement sous cette forme.

La couleur **rouge** indique des **commandes informatiques** à utiliser.

Le matériel nécessaire

Un *éditeur*¹, *compilateur assembleur*², *simulateur*³.

Nous utiliserons **MPLAB version 6.3 ou supérieure**. Cet ensemble est gratuit et disponible sur le NET (voir l'adresse plus haut).

Un *compilateur C*⁴:

Nous utiliserons **CC5X version 3.11 ou supérieure**, dont une version gratuite pour moins de 1024 octets de code est disponible sur le NET (Voir l'adresse sur le site).

CC5X est compatible avec MPLAB 6. Une fois copié sur le disque dur et déclaré, son utilisation est transparente.

Installation des deux logiciels

Installation de MPLAB 6.3

Vous possédez le fichier MPLAB630full.zip, décompressez ce fichier, vous obtenez un fichier exécutable unique MP3000.exe. Exécutez ce programme et suivez les indications (placez le logiciel à l'endroit que vous désirez).

Modification lors de l'installation d'une version postérieure à 6.42

Pour accéder au debugging, il est nécessaire de corriger le fichier "TLCC5X.INI" qui se trouve dans le répertoire "LegacyLanguageSuites" de MPLAB. Il faut remplacer "Target=HEX" par "Target=COD".

Installation de CC5X

Vous possédez le fichier cc5x3lfree.zip. Créez un répertoire CC5X où vous voulez. Décompressez tous les fichiers de cc5x3lfree.zip, dans le répertoire CC5X que vous venez de créer. Voilà, c'est fini.

¹ Logiciel permettant l'écriture du programme encore appelé code source.

² Transforme le code assembleur, en une suite de nombres compréhensibles par le microcontrôleur.

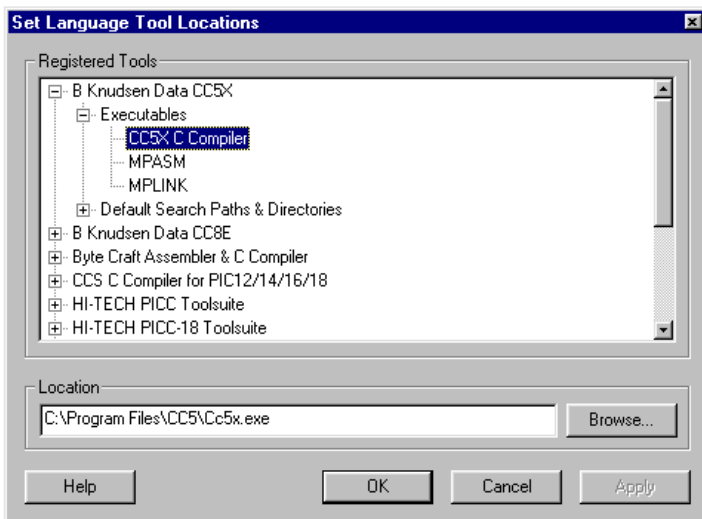
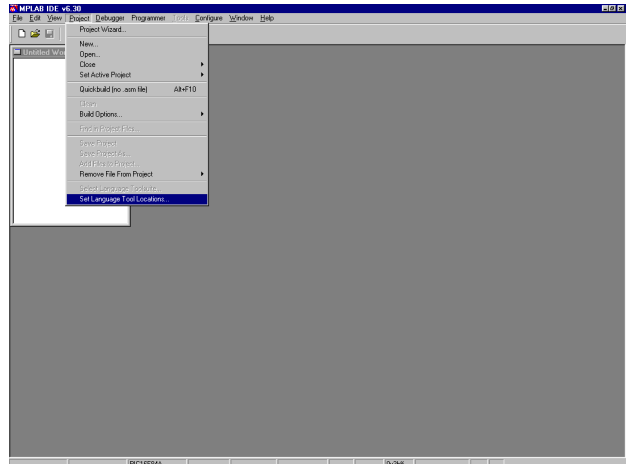
³ Permet de simuler l'état du microcontrôleur avec un ordinateur.

⁴ Transforme le code C en code assembleur.

Déclaration du compilateur CC5X dans MPLAB

La **déclaration** du compilateur s'effectue **une seule fois**.

Lancez **MPLAB IDE**
Ouvrez la fenêtre **Projet > Set Language Tool Locations...**

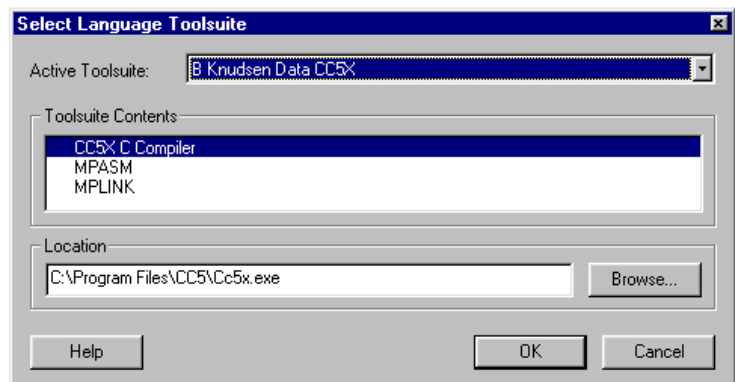


Déclarez l'emplacement du compilateur CC5X comme ci-contre.

Validez par **OK**.

Ouvrez la fenêtre **Projet > Select Language Toolsuite...**

Choisissez l'outil CC5X comme ci-contre, validez par **OK**



Le compilateur est maintenant déclaré, **cette opération n'est plus à refaire**, même si vous créez plusieurs projets, plusieurs programmes.

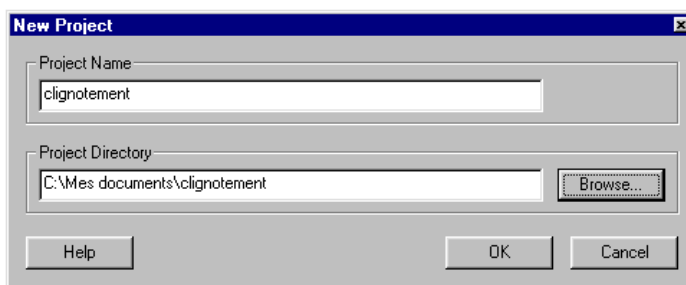
Création d'un projet

Le *projet*⁵ que nous allons créer s'appelle clignotement.

Nous allons modifier l'état de la patte RA0 d'un PIC 16F84A (patte 17), cadencé à 4 Mhz par un quartz. Cette patte passera de l'état 1 (+5V mesurés au voltmètre sur la patte du PIC) à l'état 0 (0V mesuré au voltmètre sur la patte du PIC), puis de l'état 0 à l'état 1 et recommencera infiniment.

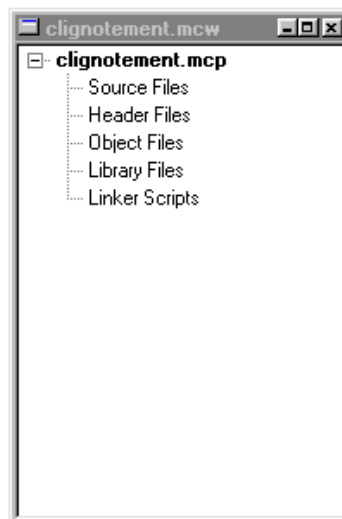
Créez un **répertoire** que vous appellerez **clignotement** dans le dossier "Mes documents".

Créez un nouveau projet sous MPLAB IDE grâce à la commande **Projet > New...**, et remplissez la fenêtre comme ci-dessous.



Validez par **OK**.

L'ensemble des fichiers utiles au projet s'affiche dans la fenêtre nommée clignotement .mcw.



⁵ Fichier contenant toutes les indications nécessaires à la programmation du PIC.

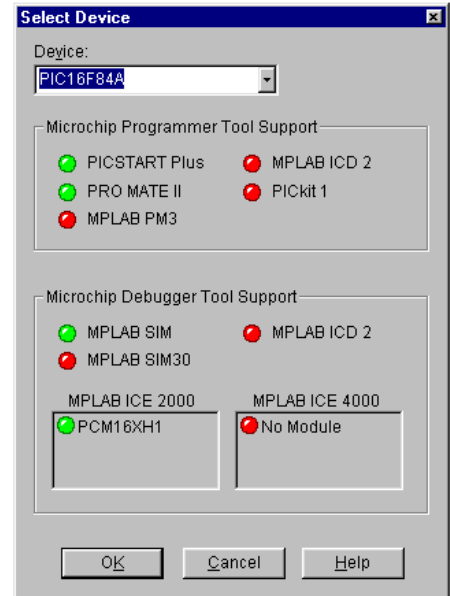
Déclaration des options

Il est nécessaire de :

- choisir un composant;
- configurer les bits d'option;
- déclarer le répertoire des fichiers inclus
- sélectionner l'option "simulation";
- choisir la fréquence du quartz,
- sauvegarder le projet.

Choix du composant

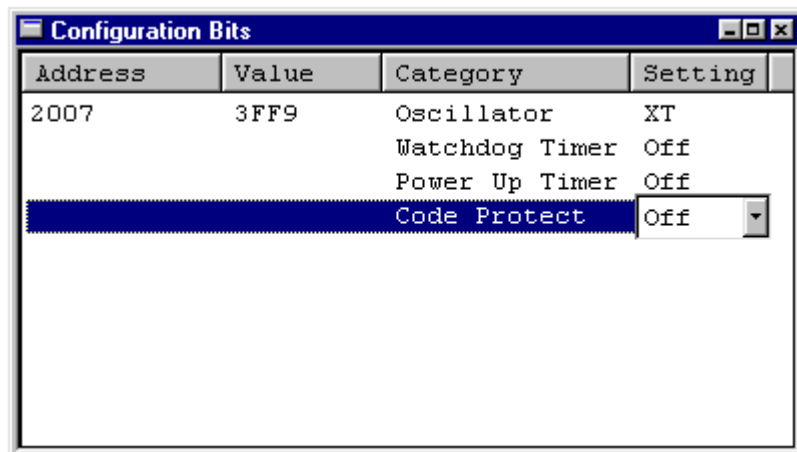
Chaque microcontrôleur possède des caractéristiques distinctes. Il est donc indispensable de choisir dès le départ un composant. Ouvrez la fenêtre **Configure > Select device...** Choisissez le composant **PIC16F84A** comme ci-contre. Validez par **OK**.



Configuration des bits d'option

Certaines fonctions du composant doivent être déclarées ou non. Cette déclaration servira pour le simulateur et le programmeur de composant.

Ouvrez la fenêtre **Configure > Configuration Bits...**, remplissez la fenêtre comme ci-dessous.



- Oscillator : choix du type de composant servant à cadencer le microcontrôleur.
- Watchdog Timer : initialisation du composant en cas de problème.
- Power Up Timer : temporisation au démarrage permettant la stabilisation de la tension d'alimentation.
- Code Protect : code de protection du composant.

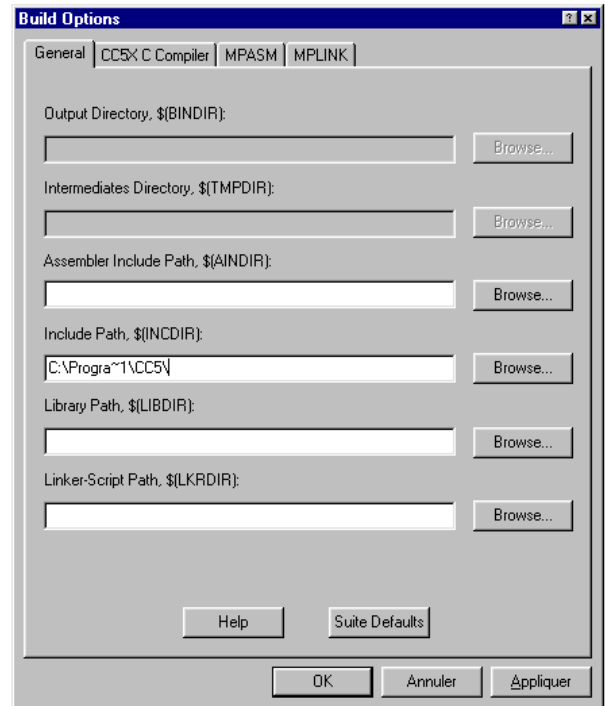
Fermez la fenêtre par l'icône fermer en haut à droite.

Déclaration du répertoire des fichiers inclus

Certaines données spécifiques au composant et nécessaires pour CC5X sont disponibles dans des fichiers du répertoire CC5X. Il est indispensable de déclarer ce répertoire.

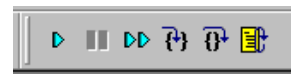
Ouvrez la fenêtre **Project > Build Options... > Project**. Remplissez la ligne Include Path par le chemin du répertoire où se situe le fichier .inc.

Selon le système d'exploitation utilisé, les **noms longs ne sont pas acceptés**. Il est alors indispensable de remplacer ce nom par son nom sous DOS comme ci-contre pour le répertoire Program Files.



Choix de l'option de simulation

Nous souhaitons faire une simulation de notre programme. Il est obligatoire de le déclarer afin de créer lors de la compilation les fichiers nécessaires. Cliquez sur **Debugger > Select Tool > MPLAB SIM**. Une barre d'outil contenant les icônes de simulation se rajoute.

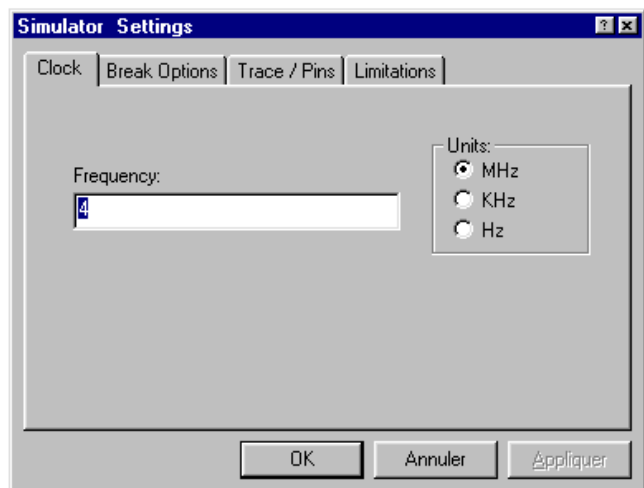


Choix de la fréquence du quartz pour la simulation

Il est maintenant nécessaire de choisir une fréquence de quartz pour la simulation. Ouvrez la fenêtre **Debugger > Setting...**

Choisissez une fréquence de quartz de 4 MHz. En tapant **4** pour Fréquence et en cochant l'unité **MHz**.

Validez par **OK**



Sauvegarde du projet

Sauvegardez vos options par la commande **Project > Save Project**.

Edition et compilation

Le projet est sauvegardé, vous pouvez le fermer si vous le désirez. Pour l'ouvrir à nouveau cliquez sur **File > Open Workspace...**

Nous allons écrire le programme correspondant à notre application. Cliquez sur **File > New**. Une fenêtre nommée Untitled* s'ouvre. **Saisissez** dans cette fenêtre **le programme** ci-dessous correspondant à l'application que nous désirons. Attention : ce programme comporte volontairement une erreur, que nous allons résoudre avec le compilateur.

```
// Ceci est un commentaire car il commence par //

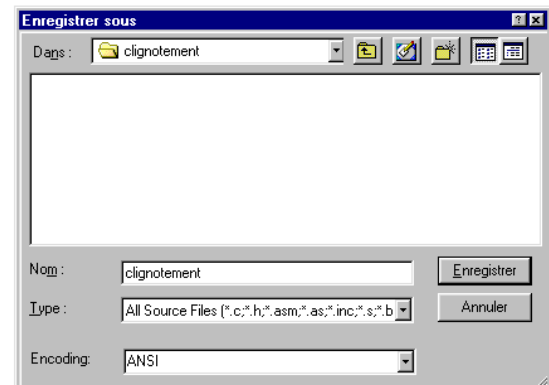
// La sortie RA0 du composant clignote
// RA0 est la patte 17 du microcontrôleur

// Attention de respecter les majuscules

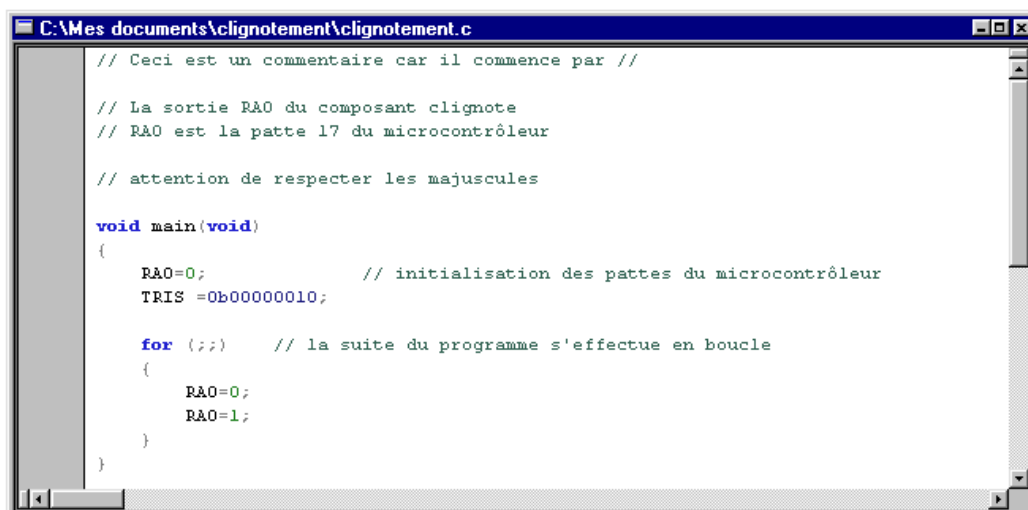
void main(void)
{
    RA0 = 0;                // Initialisation des pattes du microcontrôleur
    TRIS = 0b00000000;

    for (;;) // La suite du programme s'effectue en boucle
    {
        RA0 = 0;
        RA0 = 1;
    }
}
```

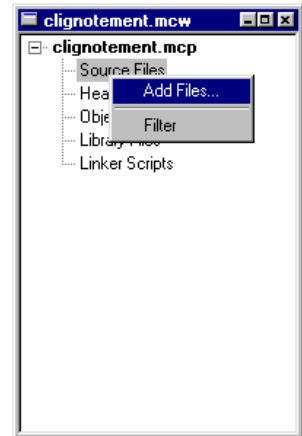
Enregistrez votre programme avec **File > Save**. Et remplissez la fenêtre comme ci-contre, cliquez sur **Enregistrer**.



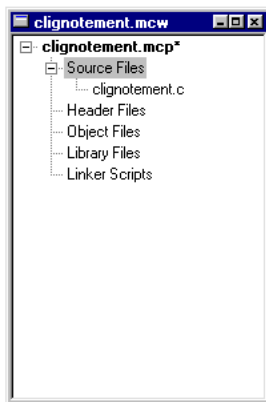
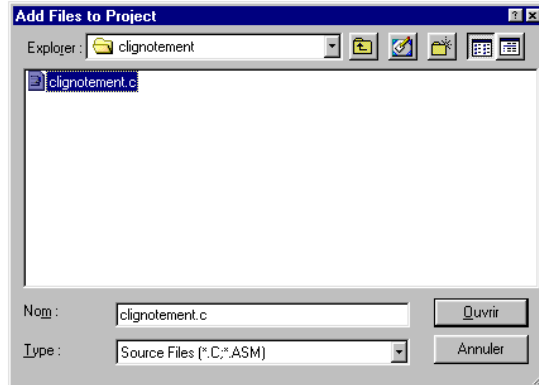
La fenêtre change alors de nom et de nouvelles couleurs apparaissent.




Le fichier que nous venons de créer doit maintenant être déclaré dans le projet comme fichier source. Pour cela, **cliquez sur le bouton droit de la souris lorsque le curseur est sur "Source Files"**. Un menu apparaît choisissez la commande **Add Files**.



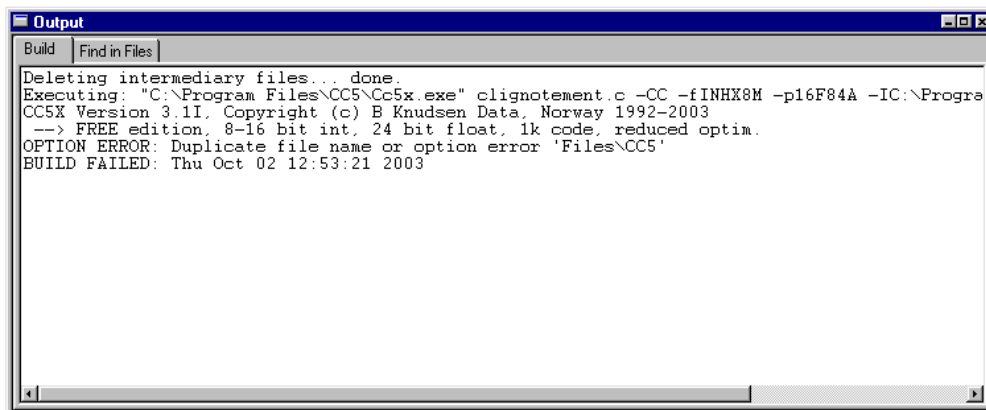
Déclarez votre fichier clignotement.c comme source et cliquez sur **Ouvrir**.



La fenêtre prend alors l'apparence ci-contre.

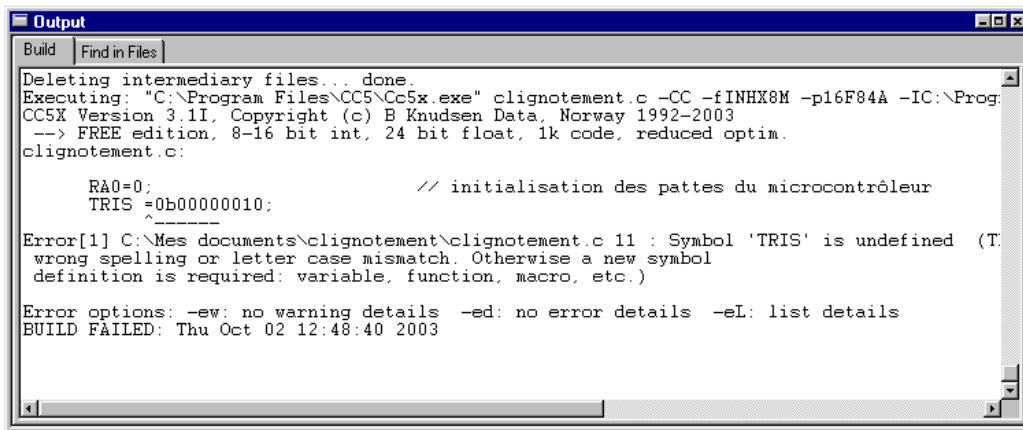
Maintenant, tout est prêt pour la compilation. Cliquez alors sur l'icône **Build**  pour compiler votre programme.

Une fenêtre s'ouvre. Si celle-ci ressemble à la **fenêtre ci-dessous**, votre système ne prend pas en charge les noms longs. Il faut alors revenir à la déclaration du répertoire des fichiers inclus afin de modifier "Program Files\CC5" par "Progra~1\CC5".



La programmation des PIC en C – Installation et première simulation

Vous devez, si tout va bien, après compilation obtenir la fenêtre ci-dessous.

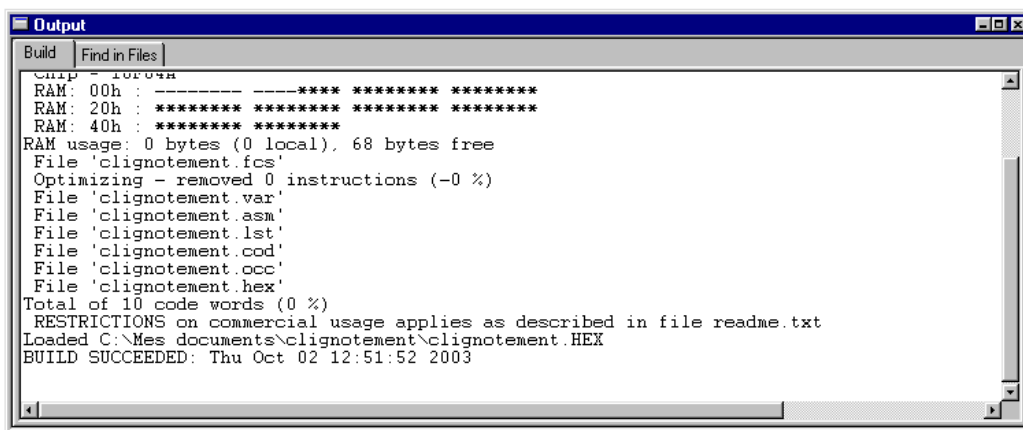


```
Output
Build Find in Files
Deleting intermediary files... done.
Executing: "C:\Program Files\CC5\Cc5x.exe" c:\notement.c -CC -fINHX8M -p16F84A -IC:\Prog
CC5X Version 3.11, Copyright (c) B Knudsen Data, Norway 1992-2003
--> FREE edition, 8-16 bit int, 24 bit float, 1k code, reduced optim.
c:\notement.c:
    RA0=0;
    TRIS =0b00000010;           // initialisation des pattes du microcontrôleur
    ^-----
Error[1] C:\Mes documents\cignotement\cignotement.c 11 : Symbol 'TRIS' is undefined (T
wrong spelling or letter case mismatch. Otherwise a new symbol
definition is required: variable, function, macro, etc.)

Error options: -ew: no warning details -ed: no error details -eL: list details
BUILD FAILED: Thu Oct 02 12:48:40 2003
```

Cette fenêtre nous annonce que la commande TRIS n'est pas définie. En effet la vraie commande est TRISA. Tapez **TRISA =0b00000000;** au lieu de **TRIS =0b00000000;**

Cette fois, votre programme est corrigé. Compilez à nouveau pour obtenir la fenêtre ci-dessous.



```
Output
Build Find in Files
Chip - 16F84A
RAM: 00h : -----
RAM: 20h : *****
RAM: 40h : *****
RAM usage: 0 bytes (0 local), 68 bytes free
File 'cignotement.fcs'
Optimizing - removed 0 instructions (-0 %)
File 'cignotement.var'
File 'cignotement.asm'
File 'cignotement.lst'
File 'cignotement.cod'
File 'cignotement.ooc'
File 'cignotement.hex'
Total of 10 code words (0 %)
RESTRICTIONS on commercial usage applies as described in file readme.txt
Loaded C:\Mes documents\cignotement\cignotement.HEX
BUILD SUCCEEDED: Thu Oct 02 12:51:52 2003
```

Certaines indications importantes apparaissent :

- les octets en RAM⁶ utilisés;
- les lignes optimisées (ici 0%);
- les différents fichiers créés;
- la partie de mémoire utilisée (ici 10 instructions machines, soit moins de 1% noté 0%);

⁶ Mémoire à accès rapide où le microcontrôleur peut écrire à tout moment.

Simulation

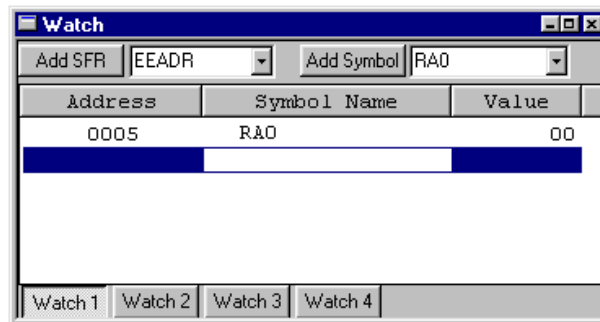
Nous arrivons à la dernière phase: tester notre programme par simulation.

Nous allons dans un premier temps "Animer" le programme, puis le faire fonctionner en "pas à pas", ensuite l'exécuter jusqu'à une instruction.


Avant tout, il nous faut ouvrir certaines fenêtres afin d'observer le fonctionnement interne du PIC.


Il est possible :


- d'observer **un bit**⁷, **un octet**⁸, **une sortie**⁹ grâce à la commande **"Watch"** du menu **"View"**.
Pour observer RA0, **Choisissez RA0** dans le menu déroulant à côté de "Add Symbol", puis cliquez sur **"Add Symbol"**.



- d'observer la **RAM** du microcontrôleur par la fenêtre **"File Registers"** du menu **"View"**.
- d'observer **le programme** dans le microcontrôleur par la fenêtre **"Program Memory"** du menu **"View"**.
- d'observer certains **octets importants** (Registres) grâce à la fenêtre **"Special Function Registers"** du menu **"view"**.

Cliquez sur l'icône **"Reset"**  (touche F6) pour **mettre à 0** le programme.

Cliquez sur **"Animate"**  pour **voir le programme** se dérouler. Une flèche verte se positionne sur l'instruction qui va s'exécuter, Certaines identités du microcontrôleur se modifient (observez surtout RA0), elles sont signalées par du rouge. dans les différentes fenêtres. Evidement, vu la vitesse de traitement, nous ne voyons pas grand chose du programme.

Cliquez sur **"Halt"**  (touche F5) pour **arrêter** l'exécution.

Pour ne simuler **qu'une seule instruction**, cliquez sur **"Step Into"**  (touche F7).

Pour exécuter le programme de la **position actuelle jusqu'à une autre position**, déplacez le curseur de souris sur la ligne correspondante à la position à atteindre, cliquez sur le bouton droit de la souris, un menu s'affiche, choisissez la commande **"Run to Cursor"**.

Voilà, vous connaissez maintenant les opérations les plus importantes pour utiliser MPLAB en langage C.

⁷ Le plus petit élément informatique. Il peut prendre l'état 0 ou 1.

⁸ Ensemble de 8 bits.

⁹ Patte du microcontrôleur sur laquelle on veut faire apparaître un 0 (0V) ou un 1 (5V).