Programmer/debugger les ColdFire avec Eclipse et GCC

Introduction

Le présent document explique comment configurer un IDE de classe professionnelle, compiler et debugger un programme simple avec Eclipse et GCC (Gnu Compiler Collection).

En préalable, s'assurer que le PC est bien équipé d'un runtime « Java » de version supérieure à 1.5.0 (ce qui est le cas de la grande majorité des PC récents). Ceci peut se vérifier en tapant *java -version* dans une fenêtre DOS. Sinon, le télécharger de http://www.java.com/fr .

Il faudra également télécharger et installer sur le PC les « GNU CoreUtils », qui interprètent les commandes des shell Linux sous Windows :
http://gnuwin32.sourceforge.net/packages/coreutils.htm ou bien
http://sourceforge.net/projects/gnuwin32/

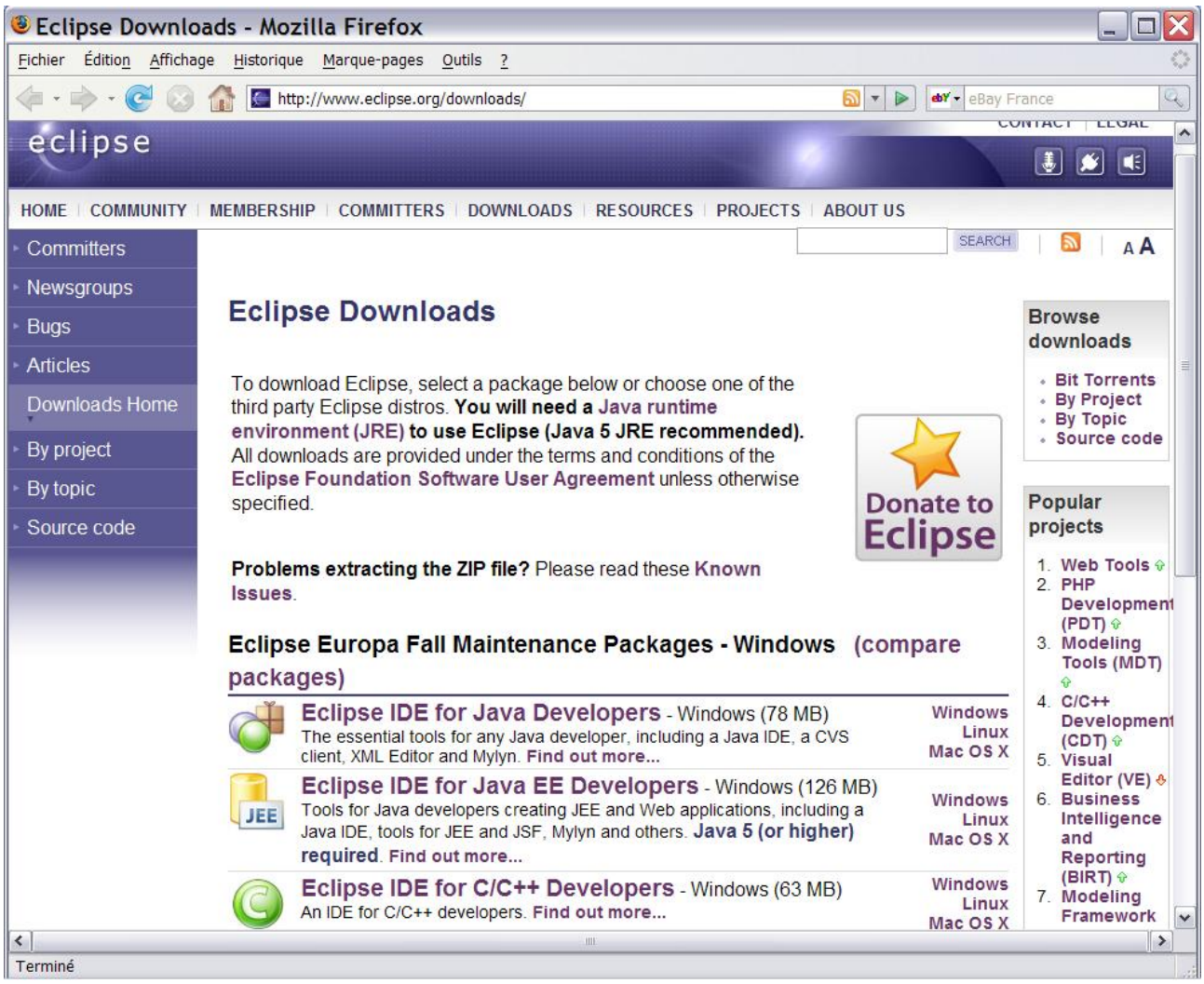
Remarque générale : J'ai personnellement tout installé dans un répertoire qui s'appelle D:\m68k. Dans les explications qui suivent, il faudra donc préciser le répertoire utilisateur approprié.

Installation de l'IDE Eclipse

L'installation de la chaine de développement nécessite d'installer :
- L'IDE eclipse; qui devra être complété par « CDT » (C Developpement Toolkit) et par « Zylin embedded CDT » (appelés « plugins »).
- Le compilateur GCC pour ColdFire (m68k-elf-gcc)

Configuration d'eclipse :

Se connecter à http://www.eclipse.org , y télécharger « Eclipse IDE for C/C++ developpers » :

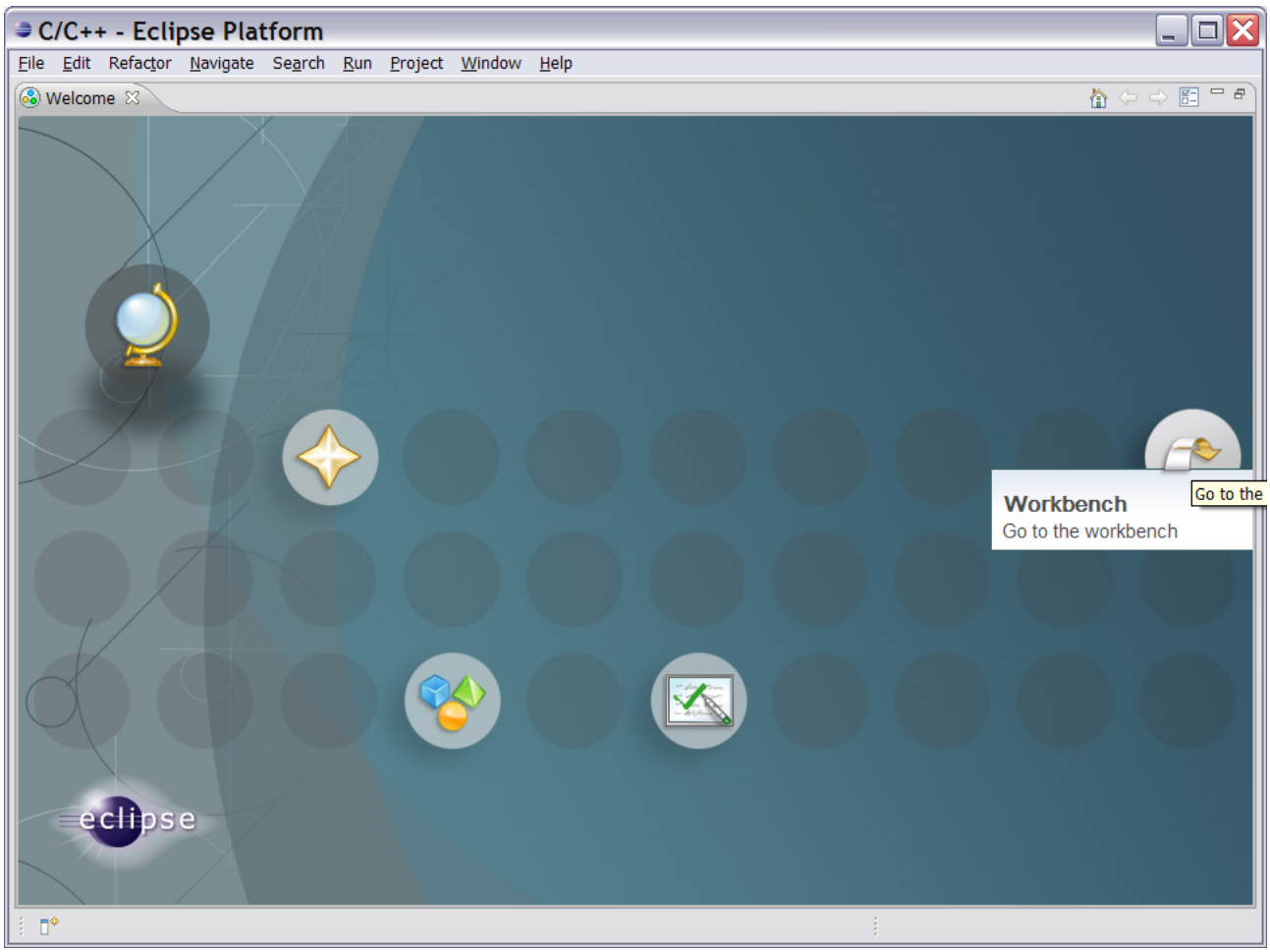


Le fichier qui sera téléchargé s'appelle : **eclipse-cpp-europa-fall2-win32.zip**

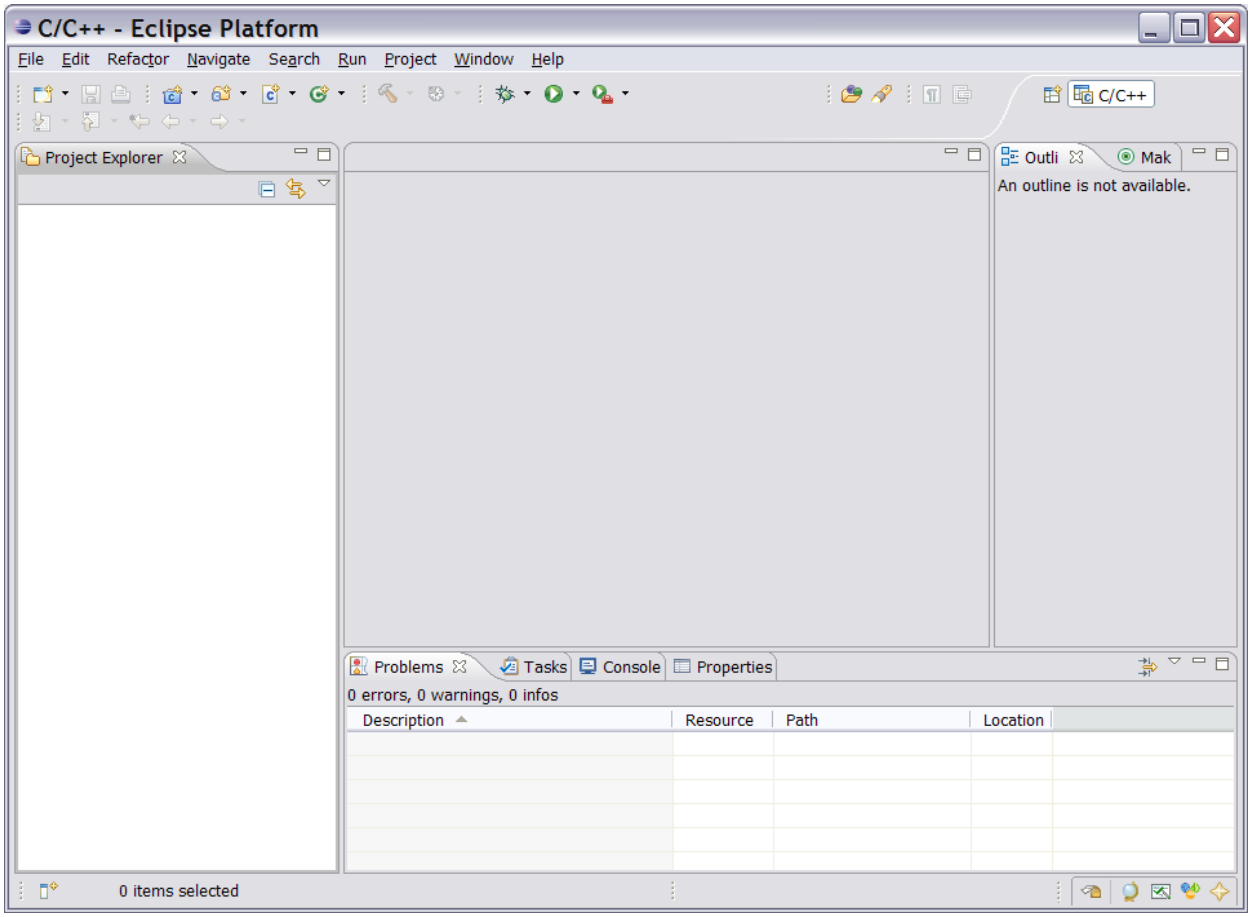
L'installation est simple : Ce zip contient un répertoire **eclipse** qu'il il suffit de décompresser à un endroit approprié.
L'étape suivante consiste maintenant à installer CDT et Zylin Embedded CDT.
Il y a d'autres méthodes que celle que je vais décrire, mais j'ai eu des problèmes d'incompatibilité entre versions; celle-ci évite ces ennuis.

Il faut lancer éclipse, simplement en cliquant sur « eclipse.exe »
Il proposera d'abord un « workspace », c'est le dossier de travail. Cliquer « OK » si celui qui est proposé convient, sinon choisir un autre.

Voilà l'écran d'accueil ; cliquer « goto the workbench »



Et voici comment cela se présente :



Cette vue, telle qu'elle apparaît à l'écran s'appelle une « Perspective ».
Ici, il s'agit de la « Perspective » « C/C++ »

Aller directement dans le menu : Help ->Software Update -> Find and Install ; sur l'écran qui appraît, choisir : « Search for new features to install »

Je ne vais pas décrire les manipulations en détail, la manière de procéder est parfaitement décrite ici :
http://subclipse.tigris.org/install.html (Ne pas installer subclipse, c'est simplement pour la manière de procéder)

Les URL des « new remote sites » desquels CDT et Zylin CDT seront téléchargés sont les suivants:

Pour CDT, le site est **http://download.eclipse.org/tools/cdt/releases/europa**

Pour Zylin embedded CDT, le site est **http://www.zylin.com/zylincdt**

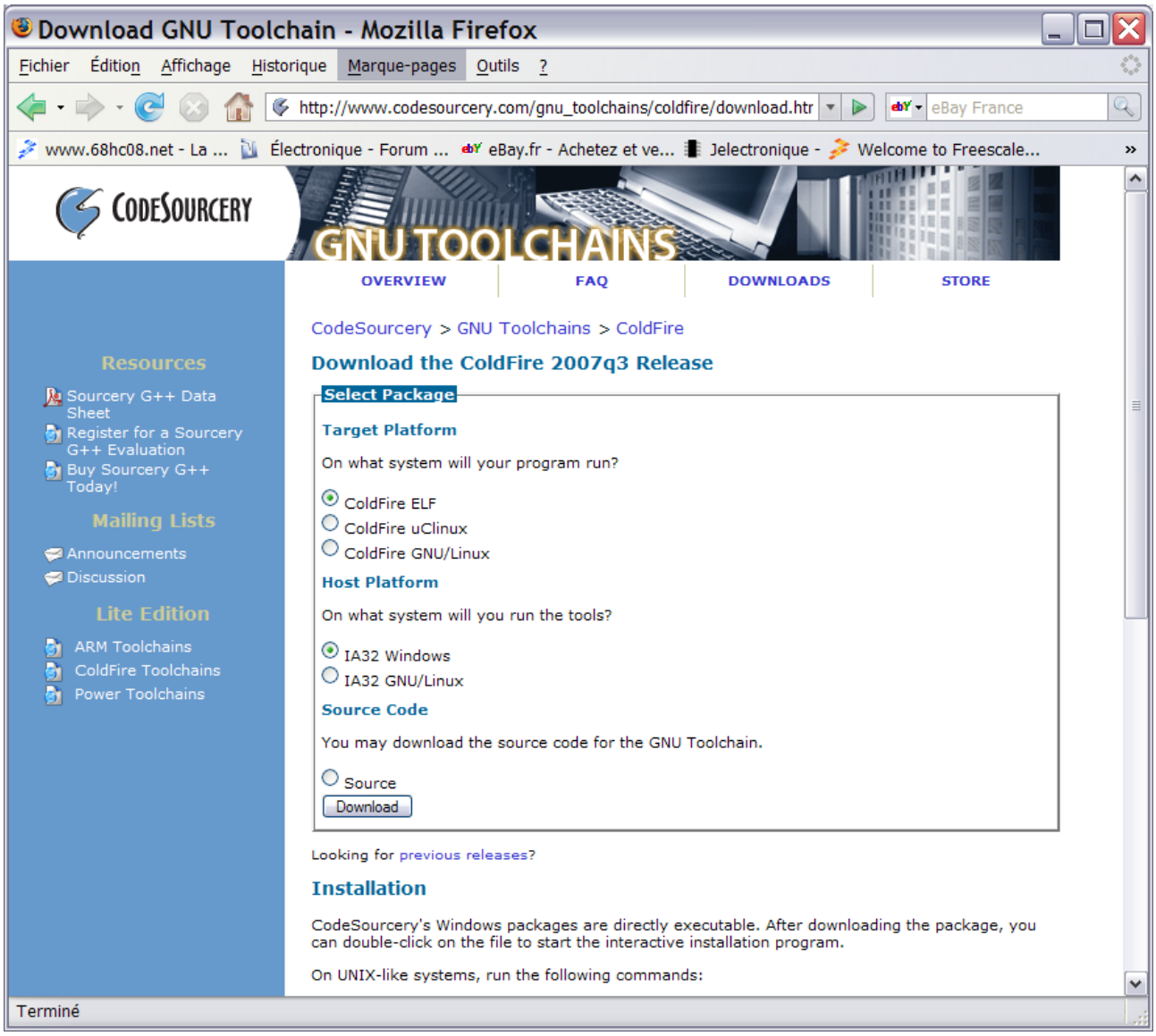
Comme « name », on peut mettre ce que l'on veut (par exemple CDT et ZylinCDT)
Sélectionner les updates trouvés, accepter les licences, et « OK » pour les menus.
Installer l'un et puis l'autre, eclipse se redémarrera après chaque installation.

Après ces deux installations, Eclipse est configuré.

Installation de la chaine GCC/GNU :

Elle est disponible gratuitement (en version LITE) ici :

http://www.codesourcery.com/gnu_toolchains/coldfire/download.html



Un clic sur « download » téléchargera le fichier :
freescale-coldfire-4.2-47-m68k-elf.exe (version actuelle à l'heure ou ces lignes sont écrites).

Exécuter ce fichier (avec choix des options par défaut installera la chaine GNU).
S'assure que la variable « PATH » du PC est bien modifiée : Pour cela, ouvrir une fenètre DOS et taper : *m68k-elf-gcc -v*
Il répondra avec la version et les options.

Quelques commentaires sur cette version :

- Elle comprend des « CS3 », start-up séquences automatisées qui comprennent :
  - L'initialisation « hard reset »
  - L'initialisation des varibles ram : En fait ; pour les variables ram initialisées, elles sont copiées en flash (section .data), à l'initialisation, leur valeur est copiée dans la ram.

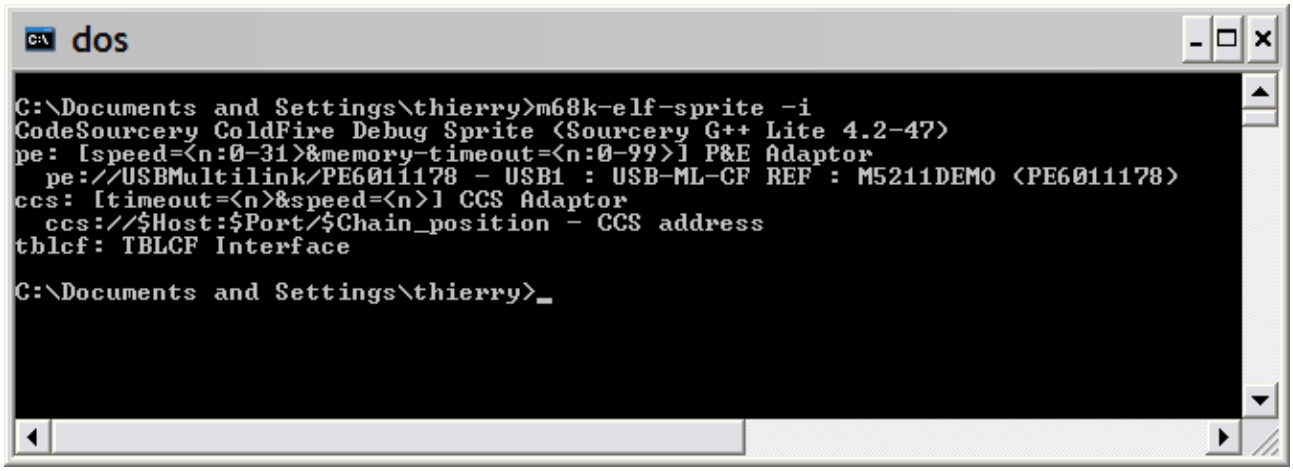
  - La déclaration des vecteurs d'interruption.

- Le BDM « open source » (TBLCF) est supporté.

- Les Flexis (ColdFire V1) sont supportés. Je n'ai cependant pas réussi à l'utiliser sur ma carte « DEMOQE128 » (sprite – voir ci après - n'a pas reconnu le BDM)

Voir le « getting started » de CodeSourcery pour des informations plus détaillées.

Pour s'assurer que la connection avec la carte est fonctionnelle : dans une fenêtre dos, après avoir connecté le BDM, taper « *m68k-elf-sprite -i* » :



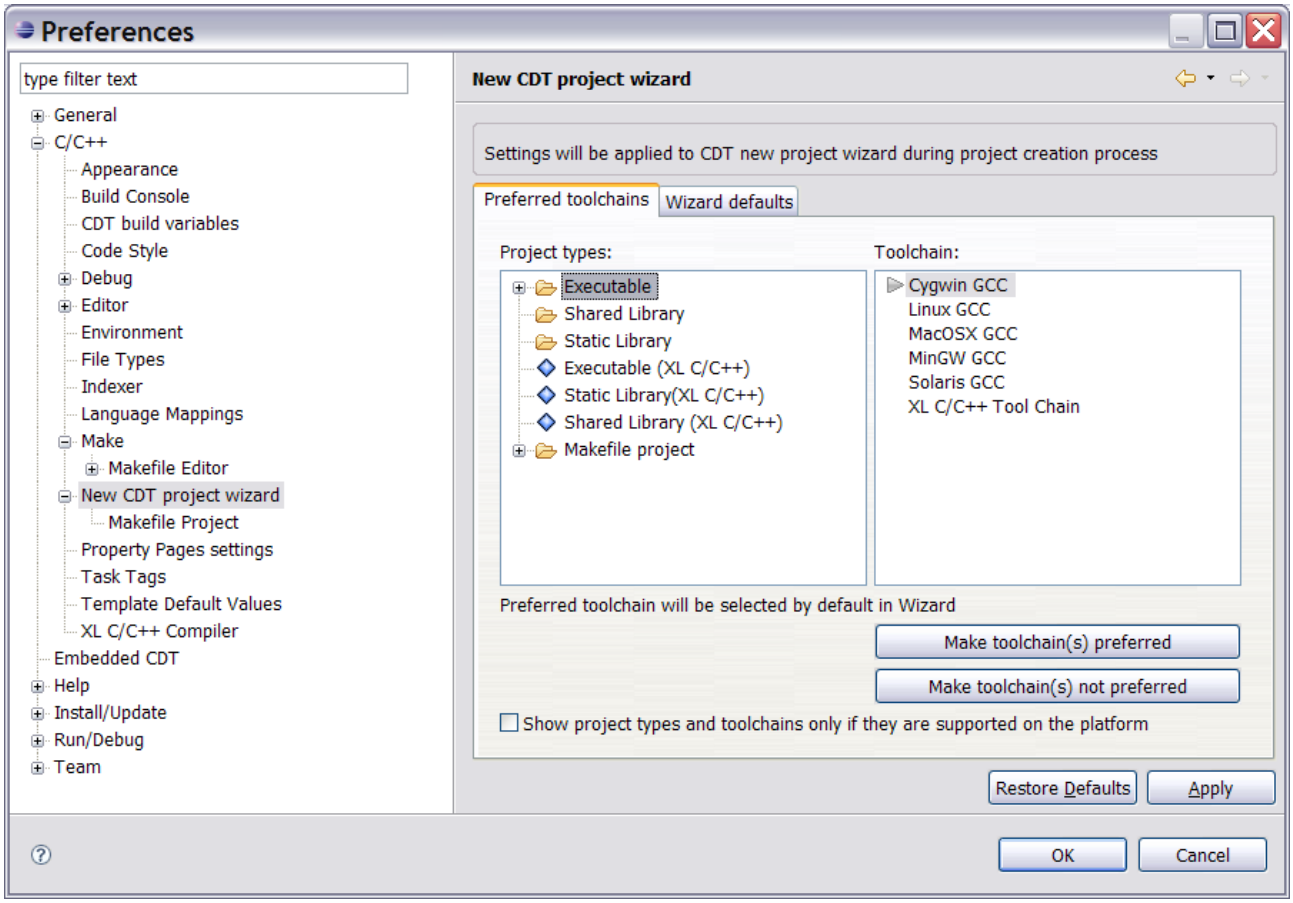
On voit la connection avec ma « M5211DEMO ».

En fait, j'ai acheté une M5211DEMO, puis j'ai désoudé le MCF5211 et remplacé par un MCF5212 (gratuit en sample chez Freescale) et ajouté un quartz 8Mhz. Cette manipulation d'une part double la mémoire disponible (16Kram, 128K Flash pour le 5211; 32k ram et 256 K flash pour le 5212) et d'autre part, permet d'utiliser cette carte comme une M5213EVB. La fréquence est limitée cependant à 66MHz.
Attention, cette manip est délicate, ces µC étant en LQFP 64 au pas de 0,5 mm, et il ne s'agit pas d'abîmer le circuit imprimé en retirant le 5211.

Le premier programme

Lancer Eclipse,

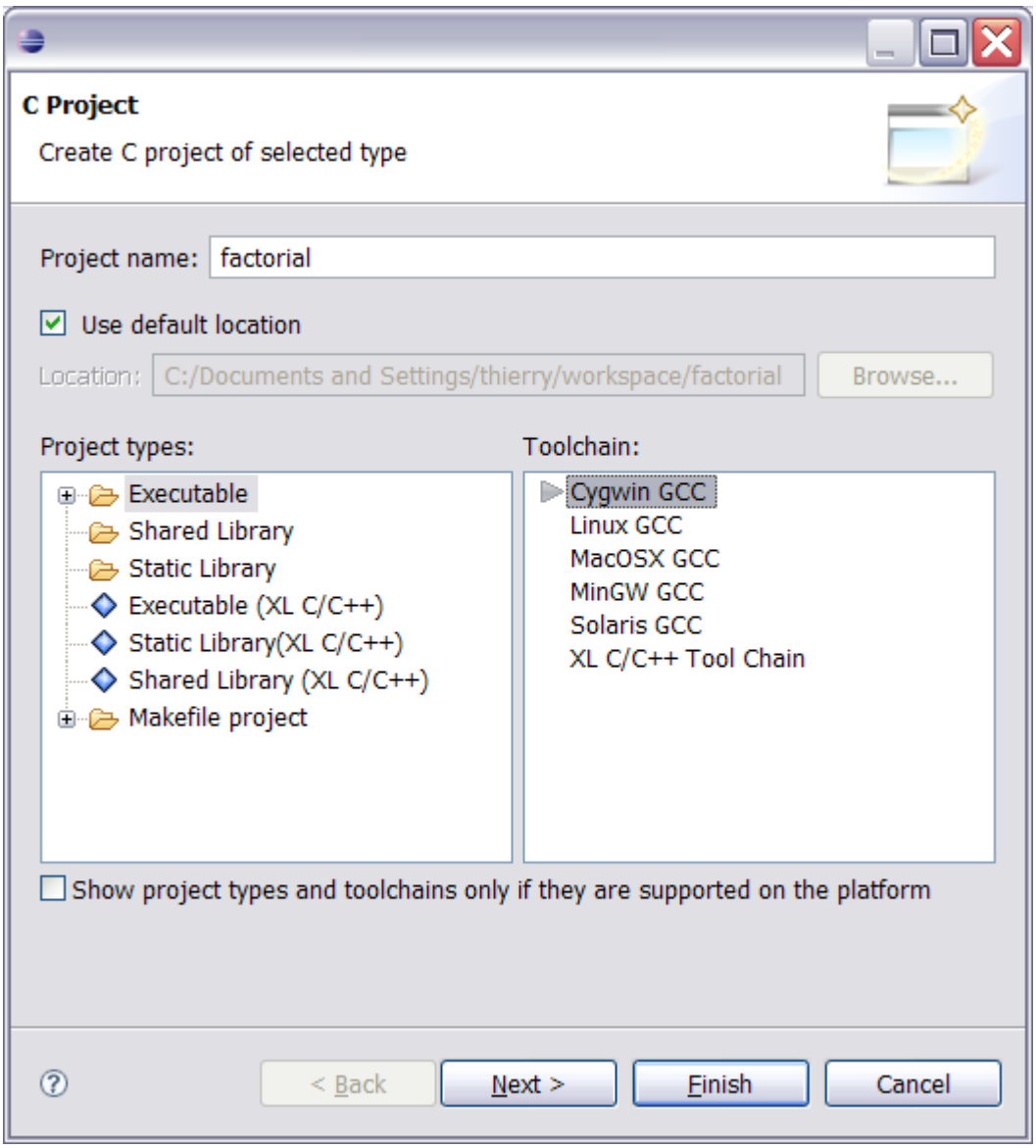
1ère étape importante : aller dans le menu : Window -> Preferences.. cliquer sur « New CDT project Wizzard » déselectionner le « Show projects types and toolchains only if they are supported on the plattform » , puis cliquer sur « Cygwin GCC » et « Make toolchain preferred »



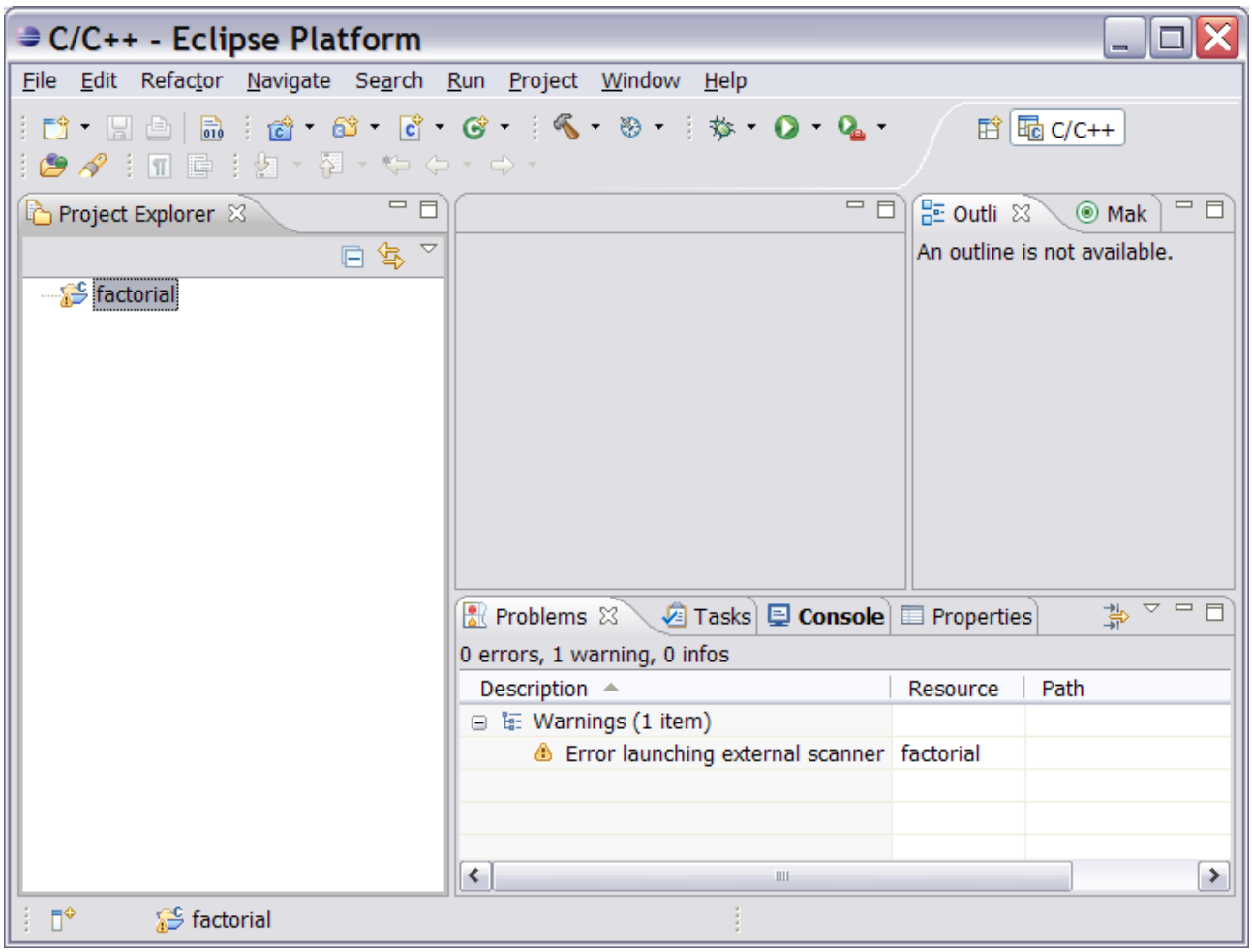
Le premier projet :

Aller dans le menu : File -> New -> C project

Dans le wizzard, taper le nom souhaité : *factorial* ; puis comme précédemment : déselectionner le « show projects and toolchain only if they are supported on the plattform » , puis cliquer sur « Cygwin GCC »



L'écran se présente alors comme ceci :



Un warning, mais ce n'est pas grave, il suffit de décocher l'option « Build Automatically » dans le menu « Project »

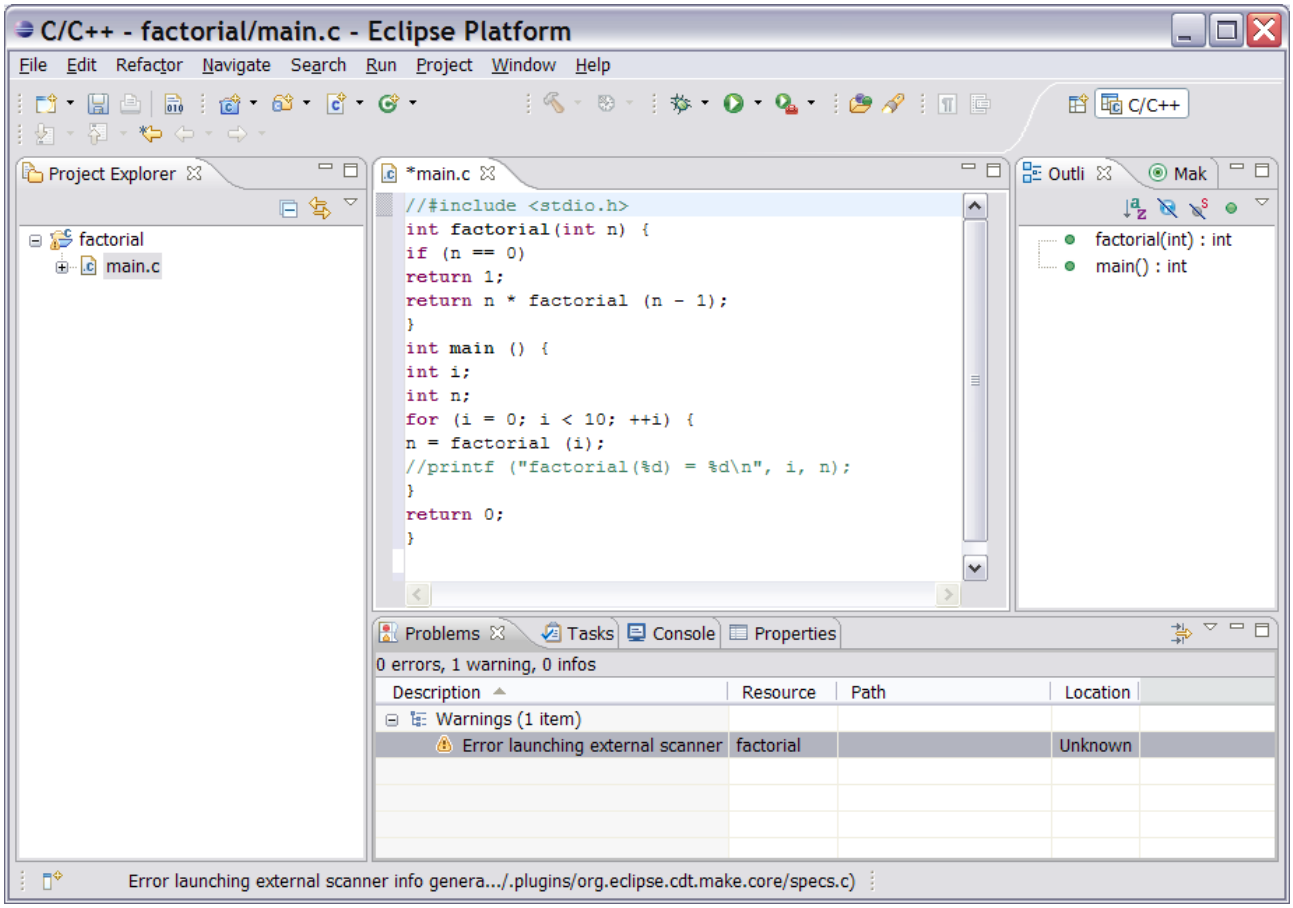
Il faut maintenant ajouter les sources C :

Cliquer sur le nom du projet (factorial), puis par un clic sur le bouton droit de la souris dans le menu contextel New -> C source File, entrer le nom : main.c, qui apparaît alors dans eclipse. Faire un copy-paste du programme suivant :

```
//#include <stdio.h>
int factorial(int n) {
if (n == 0)
return 1;
return n * factorial (n - 1);
}
int main () {
int i;
int n;
for (i = 0; i < 10; ++i) {
n = factorial (i);
//printf ("factorial(%d) = %d\n", i, n);
}
return 0;
}
```

Attention, rajouter une ligne blanche à la fin.
C'est le programme fourni en exemple dans le « getting started » de CodeSourcery. Cependant, pour le debug en ram; la fonction « printf » génère un code trop important. J'ai donc mis la ligne en commentaire.

L'écran se présente alors comme ceci :



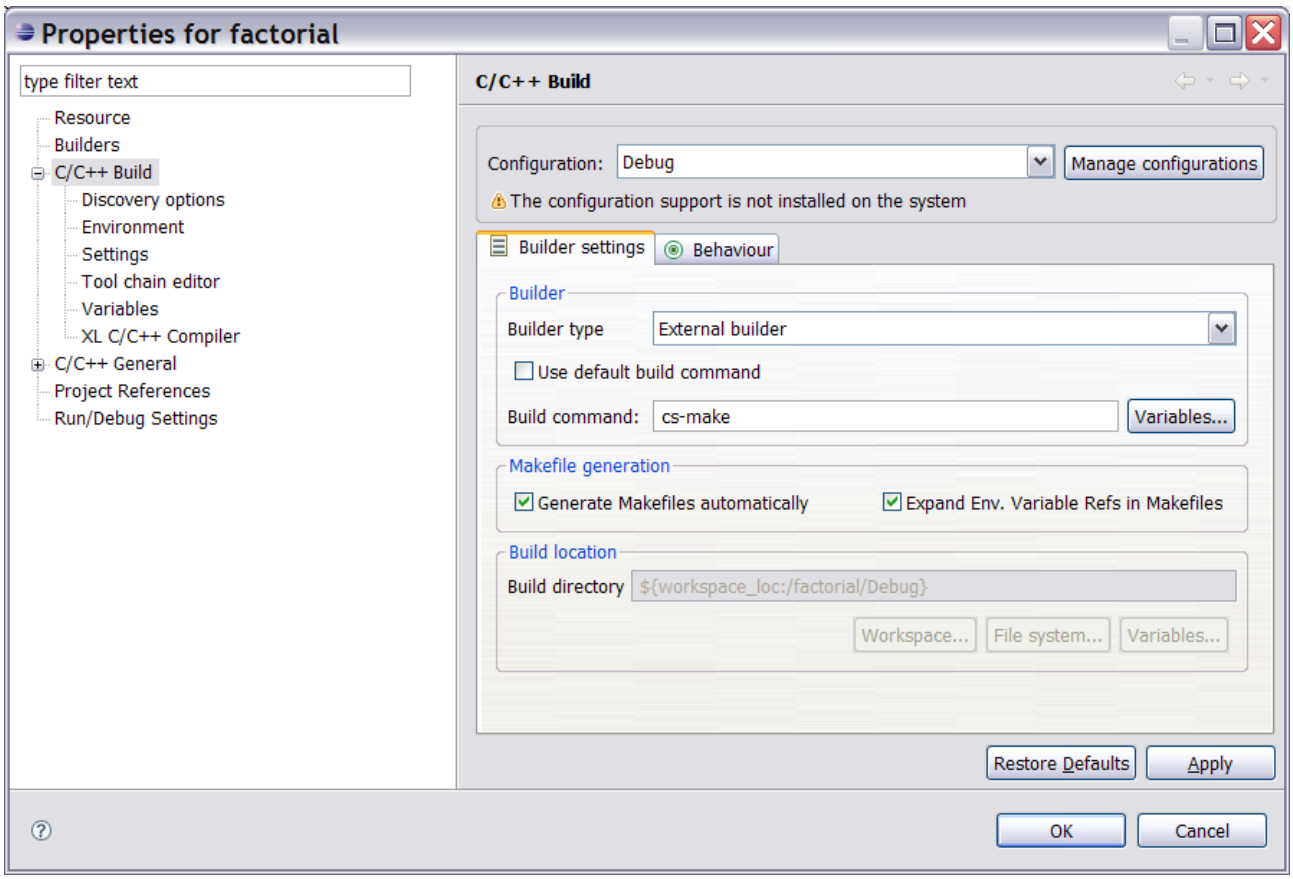
Attention, toute modification effectuée dans la fenètre centrale (édition) doit être suivie d'un save, sinon, elle n'est pas prise en compte. (attention donc avant de refaire un « buid »)

Il est également possible d'importer plusieurs sources/ sous folders en une seule opération, dans le menu contextuel, c'est l'option « Import » .. puis «File System »

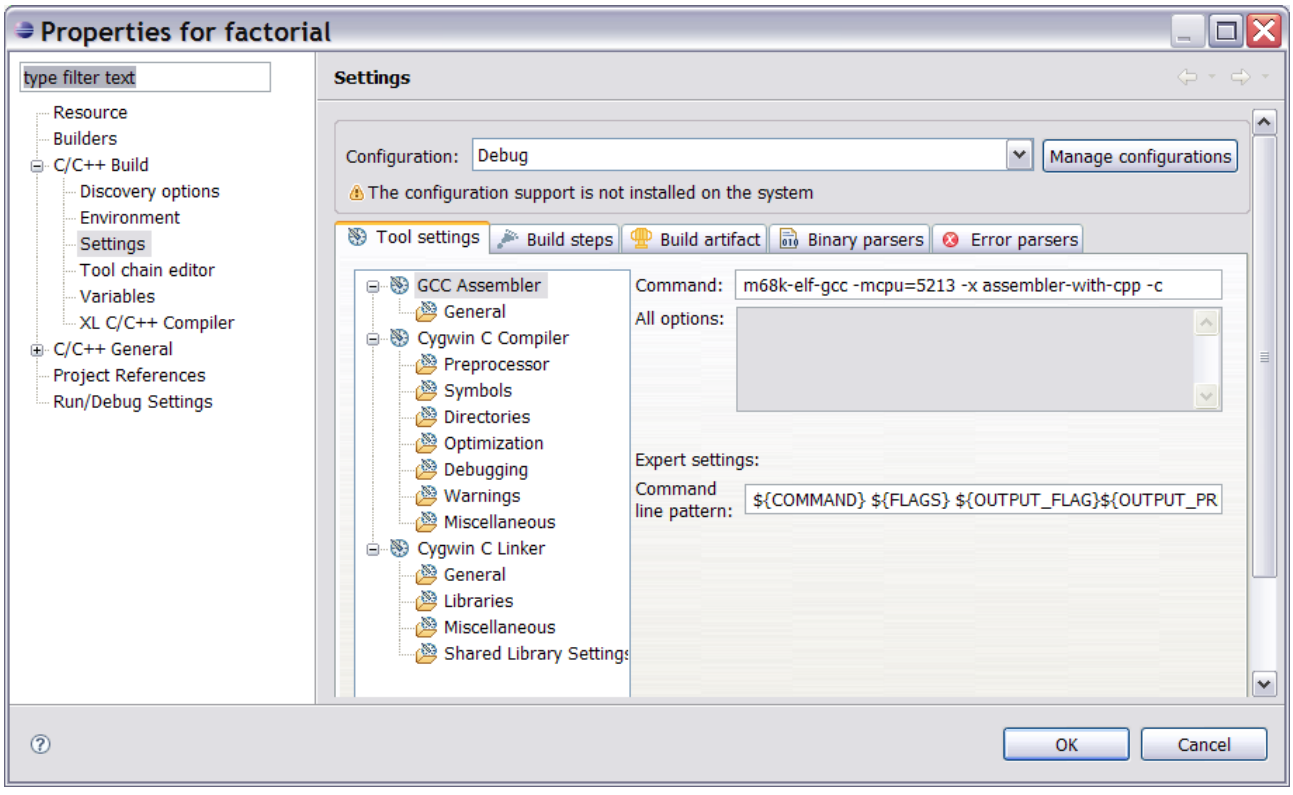
Commence alors une délicate étape de configuration, mais elle n'est répétée qu'une seul fois (par projet) et permet d'éviter l'écriture d'un Makefile assez complexe, qui sera généré automatiquement : cliquer sur le nom du projet, puis par un clic sur le bouton droit de la souris dans le menu contextuel : **Properties**

Cliquer sur le menu **C/C++ build** :

1. 1ère modification : décocher « Use default build command » et taper « **cs-make** » à la place de « make », puis « Apply »

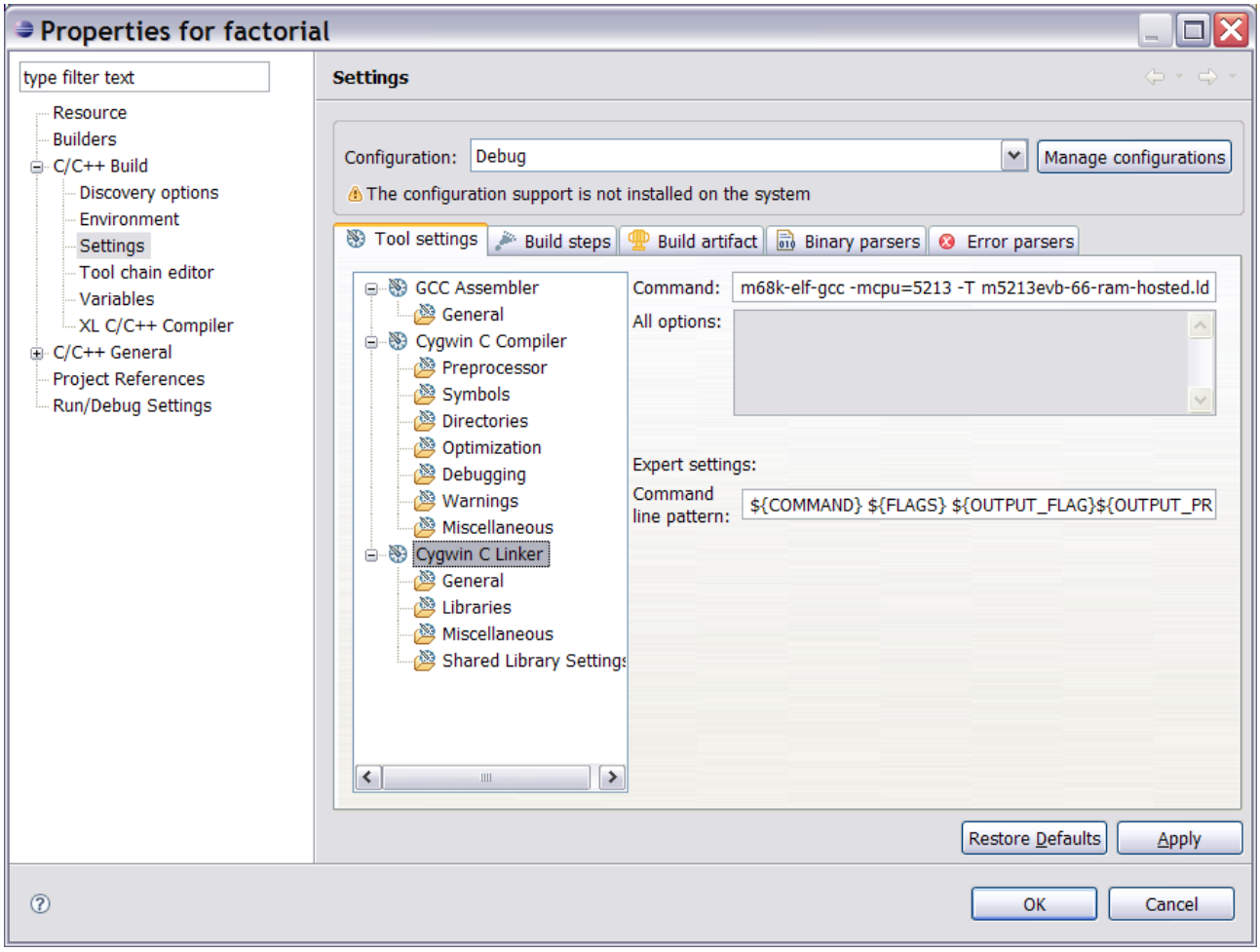


2. Cliquer maintenant sur « Settings » de C/C++ build, dans l'onglet « tools settings »
Il faut modifier les invocations de l'assembleur, du compilteur et du linker :
Modifier l'invocation de **l'assembler** « as »
par « **m68k-elf-gcc -mcpu=5213 -x assembler-with-cpp -c** »
Cette option dépend du CPU



3. De la même manière, Cygwin C **compiler** toujours de l'onglet « Tools settings » Command :
m68k-elf-gcc -mcpu=5213

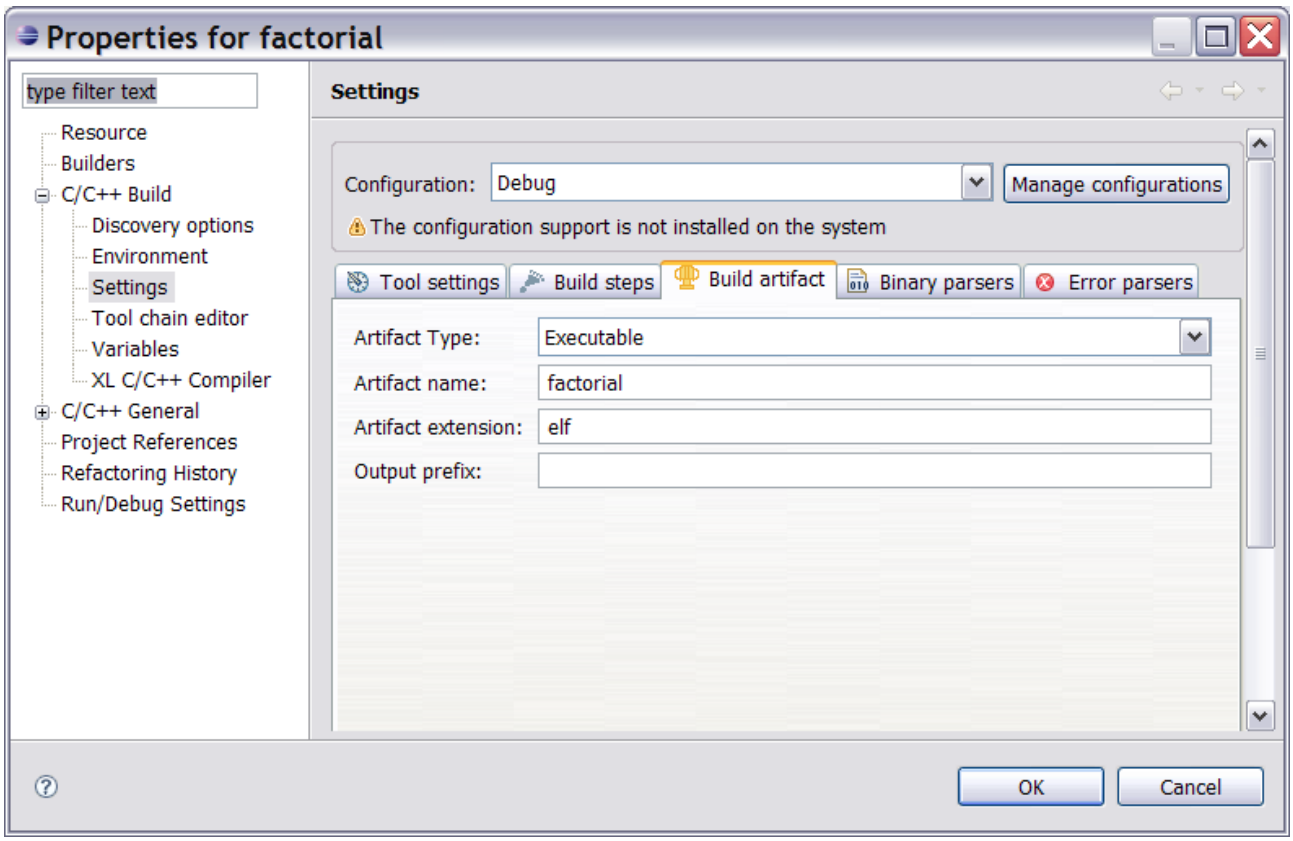
4. Et finalement, dans l'onglet « **linker** » :
m68k-elf-gcc -mcpu=5213 -T m5213evb-66-ram-hosted.ld



Une remarque importante :

Le fichier « m5213evb-66-ram-hosted.ld » est le script du linker. Différents scripts se trouvent dans le folder CodeSourcery\Sourcery G++ Lite\m68k-elf\lib. On choisira celui qui est approprié. Par exemple, pour générer un exécutable en flash, le fichier choisi sera m5213evb-66-**rom**-hosted.ld

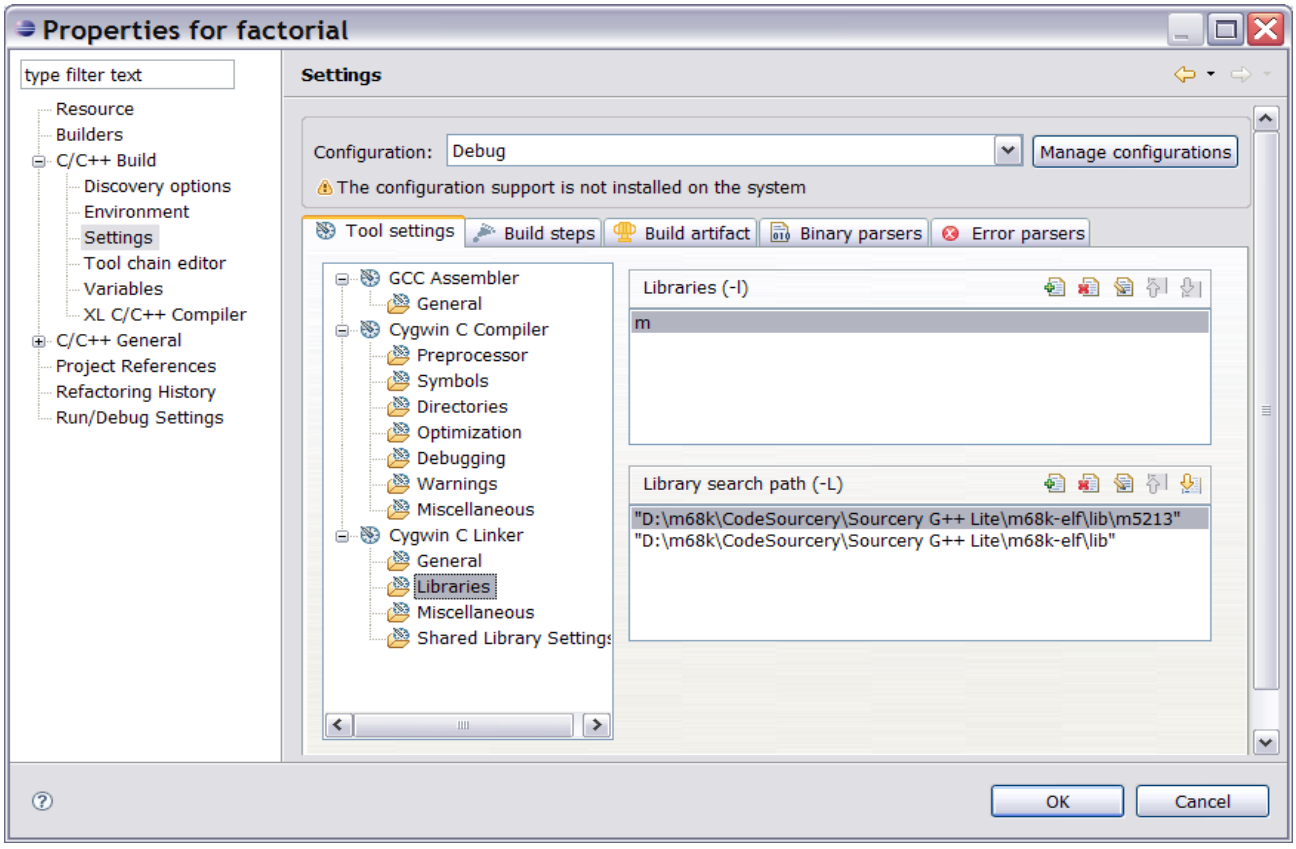
Dans l'onglet «Build artifact », remplacer l'Artifact extension « .exe », proposée par défaut, qui peut poser des problèmes, par « .elf »



Complément (non indispensable) : utilisation des float

Si l'on veut utiliser les fonctions « floating point » telles que sin(), cos(), sqrt(), il faut

- mettre « `#include <math.h>` » dans le main ;
- configurer le linker de la manière suivante :



afin qu'au link, il fasse appel à la librairie « libm.a »

Note: D:\m68k est le dossier dans lequel j'ai personnellement installé CodeSourcery. Préciser le dossier adéquat (par exemple « C:\Program Files »)

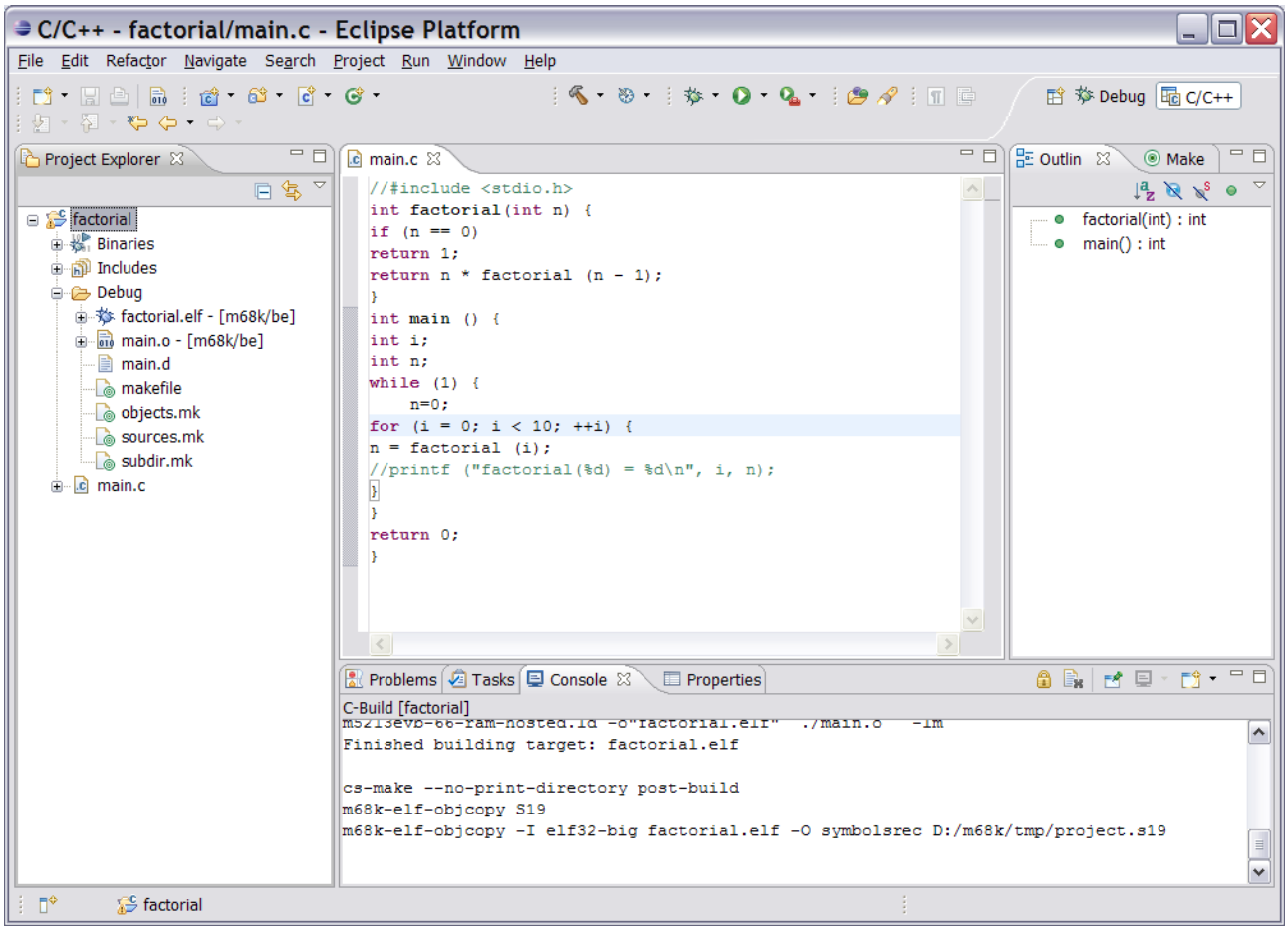
Les diférents paramètres s'ajoutent (ou s'enlèvent) en cliquant sur les icones

(-fin du complément-)

Cliquer sur « OK » pour fermer l'écran, le projet est alors prêt à être compilé. La création d'un exécutable s'appelle un « build ».

Dans l'écran principal, aller dans le menu « Project » puis « Clean.. », s'assurer que l'option « start a build immediately » est bien coché puis « OK ».

Après avoir cliqué sur « OK », 'écran se présente comme ceci :

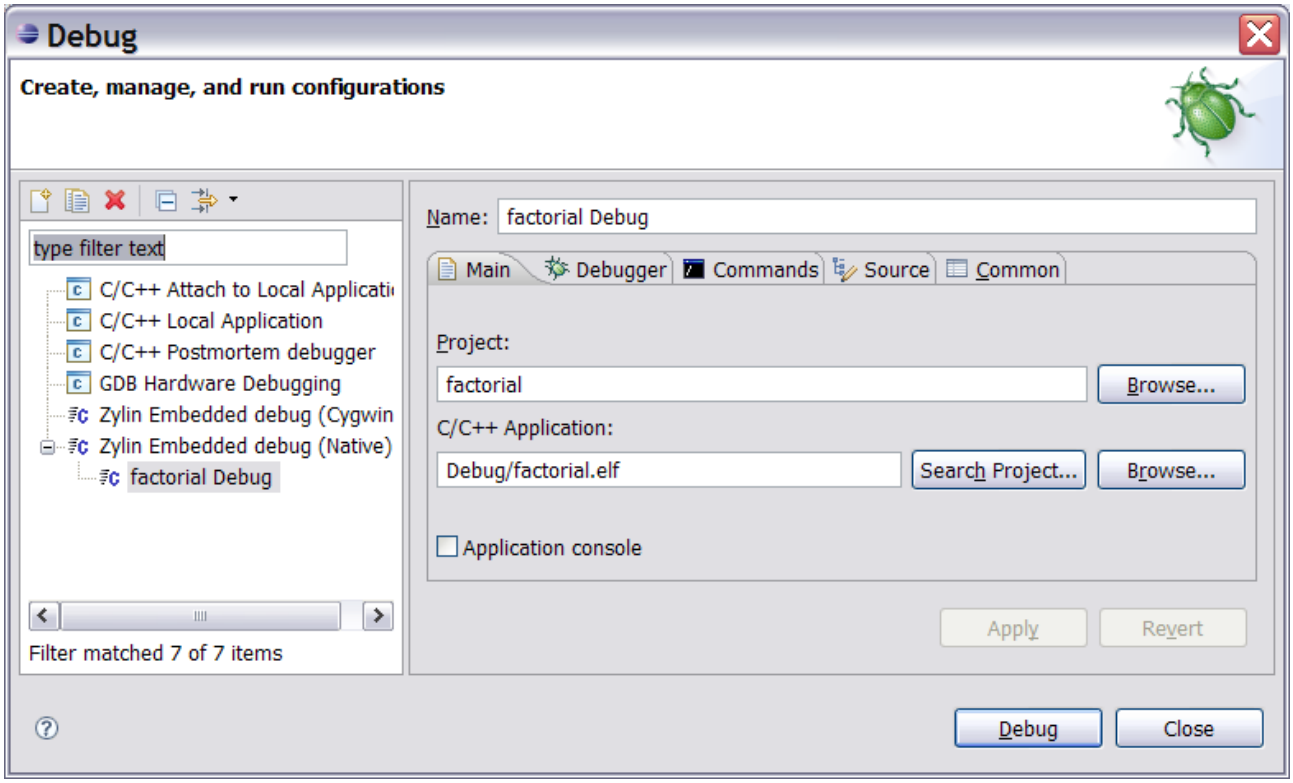


On peut constater que les différents « makefile » (et .mk) ont été rajoutés automatiquement.
Un exécutable « elf » est présent.

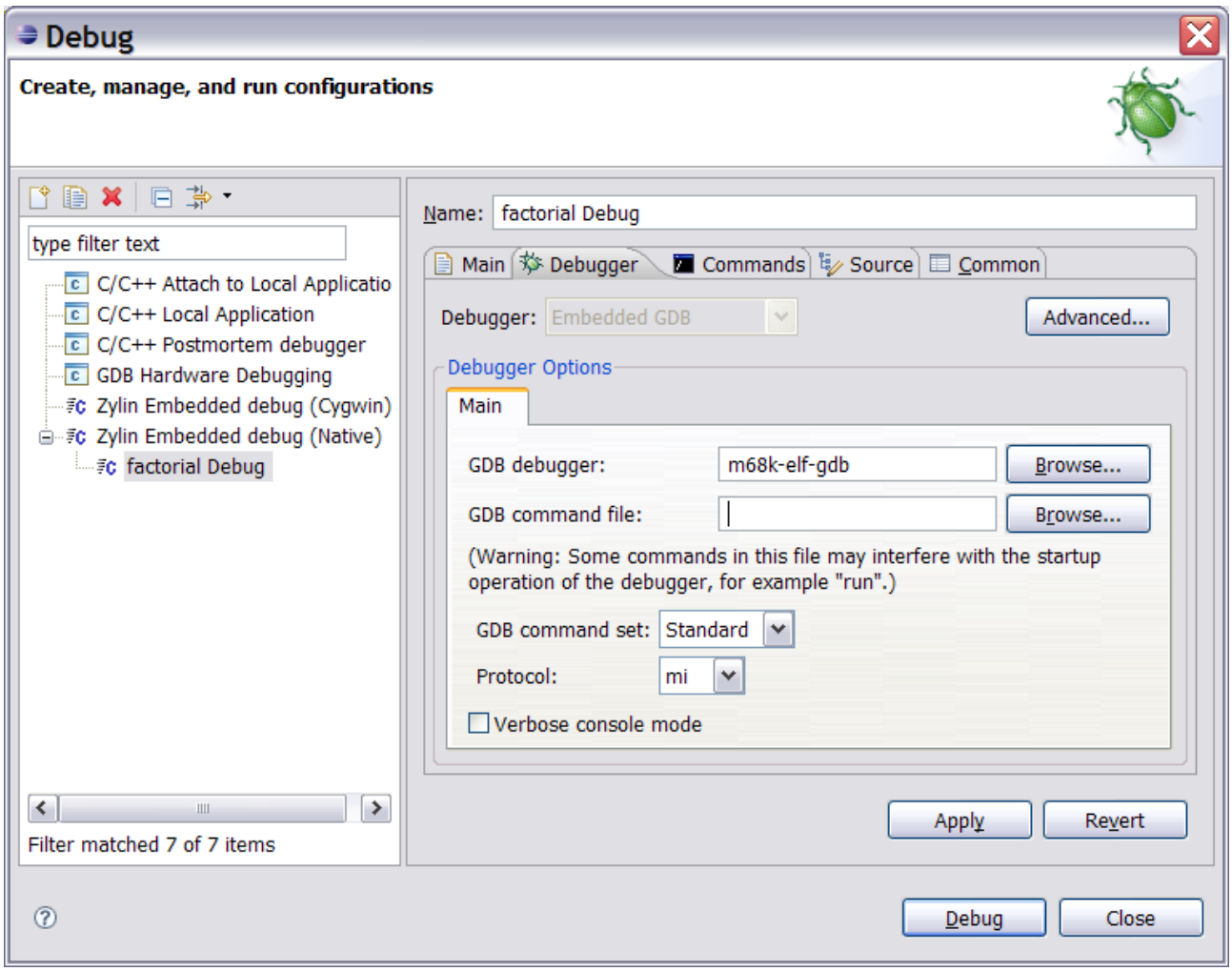
Debug en RAM

Il faut d'abord ouvrir la « perspective debug » : aller dans le menu
« Window » --> « Open Perspective » --> « Debug »

Dans le menu « Run » ou en cliquant sur la petite flèche (vers le bas) à côté de l'insecte vert, on choisit « Open debug dialog », puis en déroulant « Zylin Embedded debug (Native), il y a une configuration « factorial debug ».

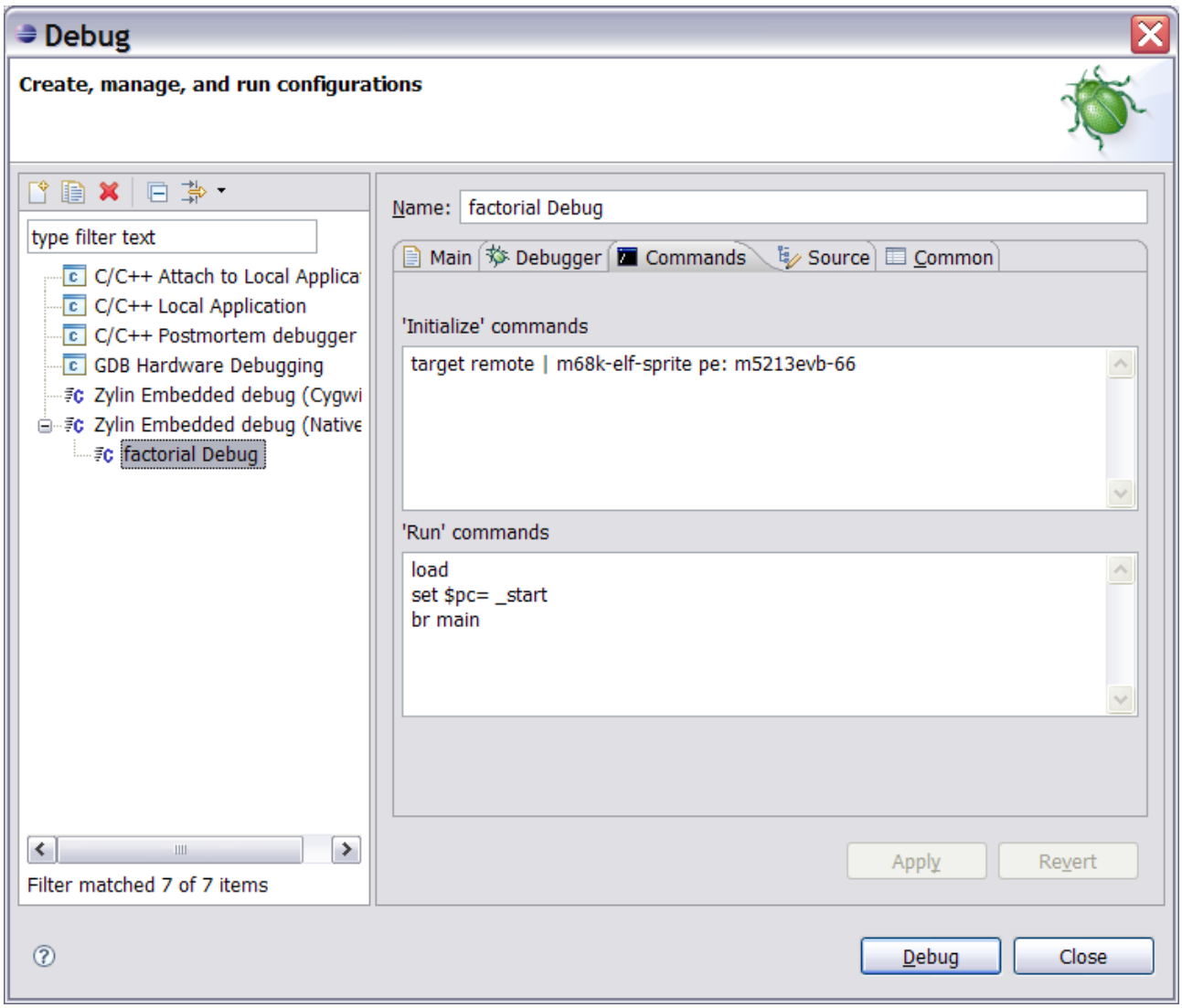


Il faut maintenant configurer le Debugger , dans l'onglet approprié:



Le debugger s'appelle m68k-elf-gdb. Ne pas oublier d'enlever le « GDB command file », proposé par defaut à .gdbinit

Dans l'onglet « Commands » , compléter « 'Initialize' commands » de la sorte :



Cette commande initialise le système (ici m5213evb à 66Mhz). Cela correspond à un fichier « xml » qui se trouve dans le repertoire : CodeSourcery\Sourcery G++ Lite\m68k-elf\lib\boards
Il est possible de créer des xml « custom » selon la configuration.

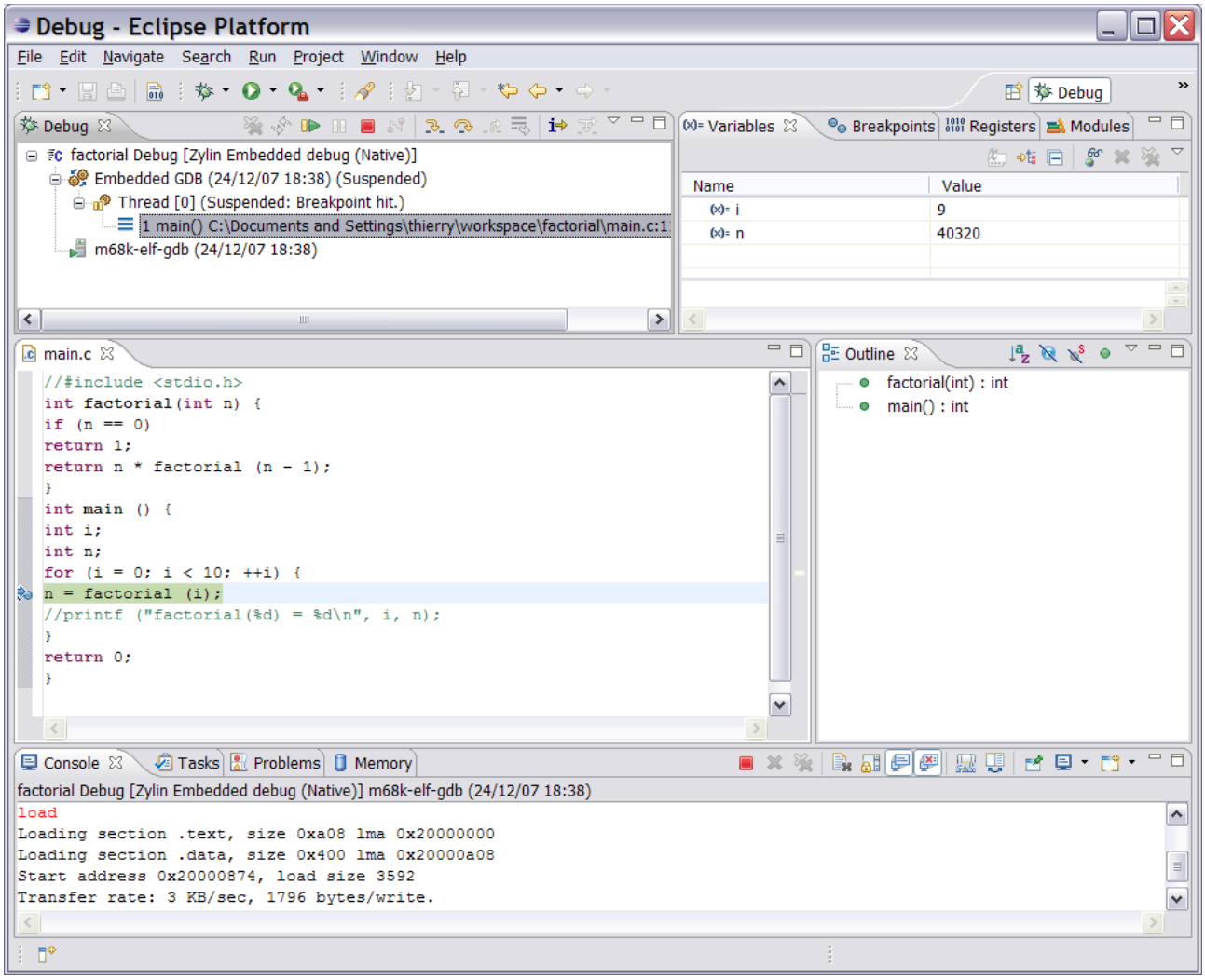
Connecter la carte ColdFire, puis cliquer sur le bouton « Debug ».

Voici ce qui doit apparaître dans la « Console » :

```
target remote | m68k-elf-sprite pe: m5213evb-66
m68k-elf-sprite: Opening P&E USBMultilink port 1 (USB1 : USB-ML-CF REF :
M5211DEMO (PE6011178))
m68k-elf-sprite: Target reset
0x00000000 in ?? ()
load
Loading section .text, size 0xa08 lma 0x20000000
Loading section .data, size 0x400 lma 0x20000a08
Start address 0x20000874, load size 3592
Transfer rate: 28736 bits in <1 sec, 1796 bytes/write.
set $pc= _start
br main
Breakpoint 1 at 0x200004f6: file ../main.c, line 11.
```

La suite relève de l'utilisation du debugger GDB. Il est par exemple possible de mettre un « Breakpoint » en positionnant la souris devant une ligne C et en cliquant avec le bouton droit,

dans le menu contextuel « toggle breakpoint », exécuter pas par pas, examiner les registres, la mémoire, l'asm, ... ou même d'introduire directement des commandes GDB dans la « console ». Le BDM réagit instantanément.



Pour arrêter le debug, cliquer sur « factorial debug » puis, avec le menu qui apparaît avec un clic du bouton droit : « Terminate and remove ».

Pour repasser à l'écran C du compilateur, aller dans le menu « Window --> Open Perspective -> C/C++. Si, par mégarde, on ferme une ou plusieurs fenêtres de la « Perspective », toujours dans le même menu windows, effectuer : « Reset Perspective ».

Les différentes fenêtres qui apparaissent dans une Perspective s'appellent des « Vues » View. Si l'on veut en ajouter, il faut aller dans le menu « Window » --> « Schow View ».

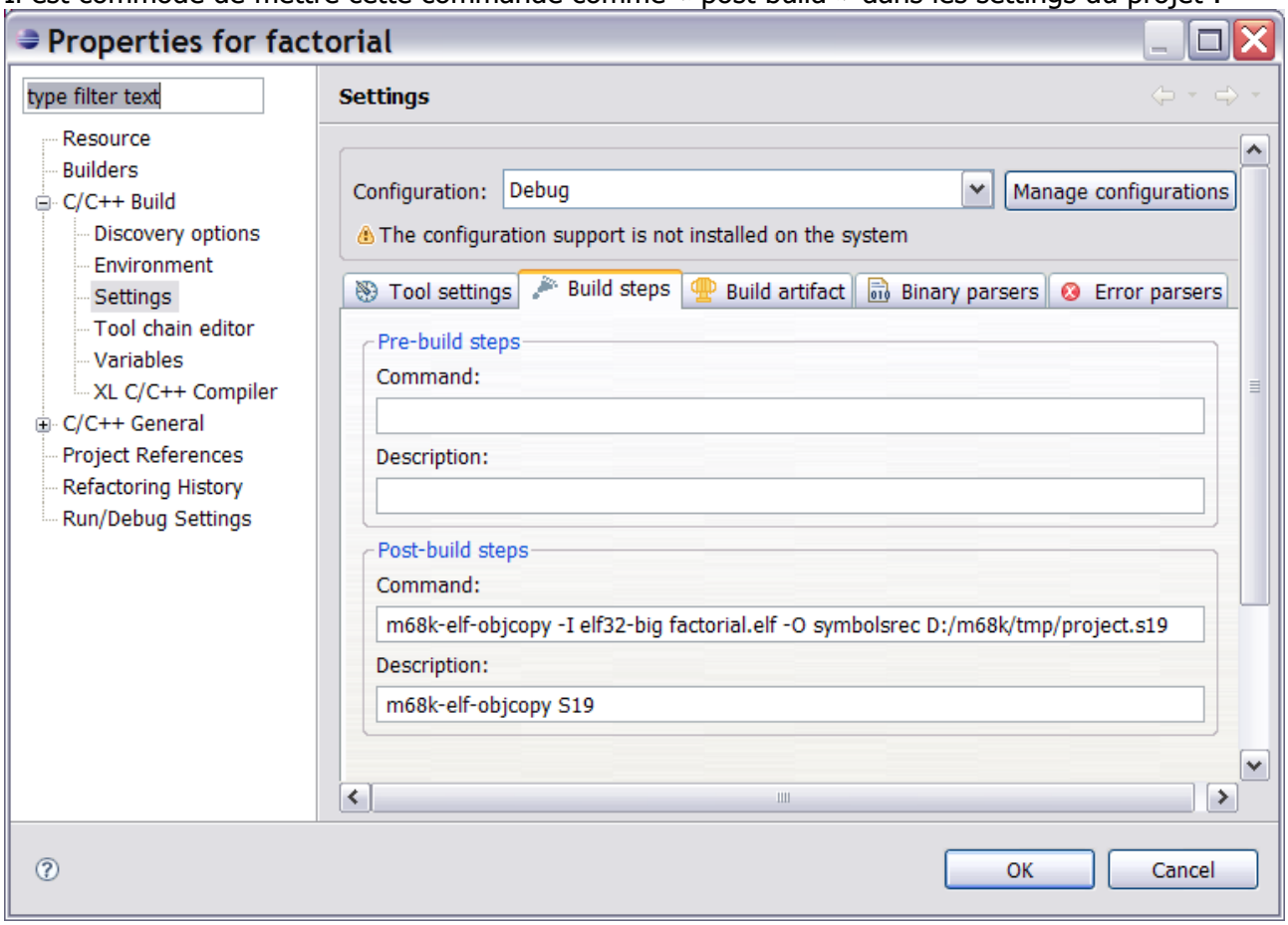
Debug en FLASH

Bien que la version « lite » de Code Sourcery ne soit pas censée debugger en Flash, il y a quand même moyen : On va utilser l'utilitaire gartuit CFFlasher fourni par Freescale. Auparavant, il faudra utilser le programme *m68k-elf-objcopy* , pour générer à partir de l'exécutable elf un S19.

La commande est la suivante :

m68k-elf-objcopy -I elf32-big factorial.elf -O symbolsrec D:\m68k\tmp\project.s19

Il est commode de mettre cette commande comme « post build » dans les settings du projet :



Ici, je crée ce S19 dans un répertoire temporaire D:\m68k\ pour des raisons de facilité de l'utilisation de CFFlasher.

-Complément non indispensable : intégration de CFFlasher dans Eclipse

Il est possible de lancer CFFlasher directement depuis Eclipse.

Pour cela :

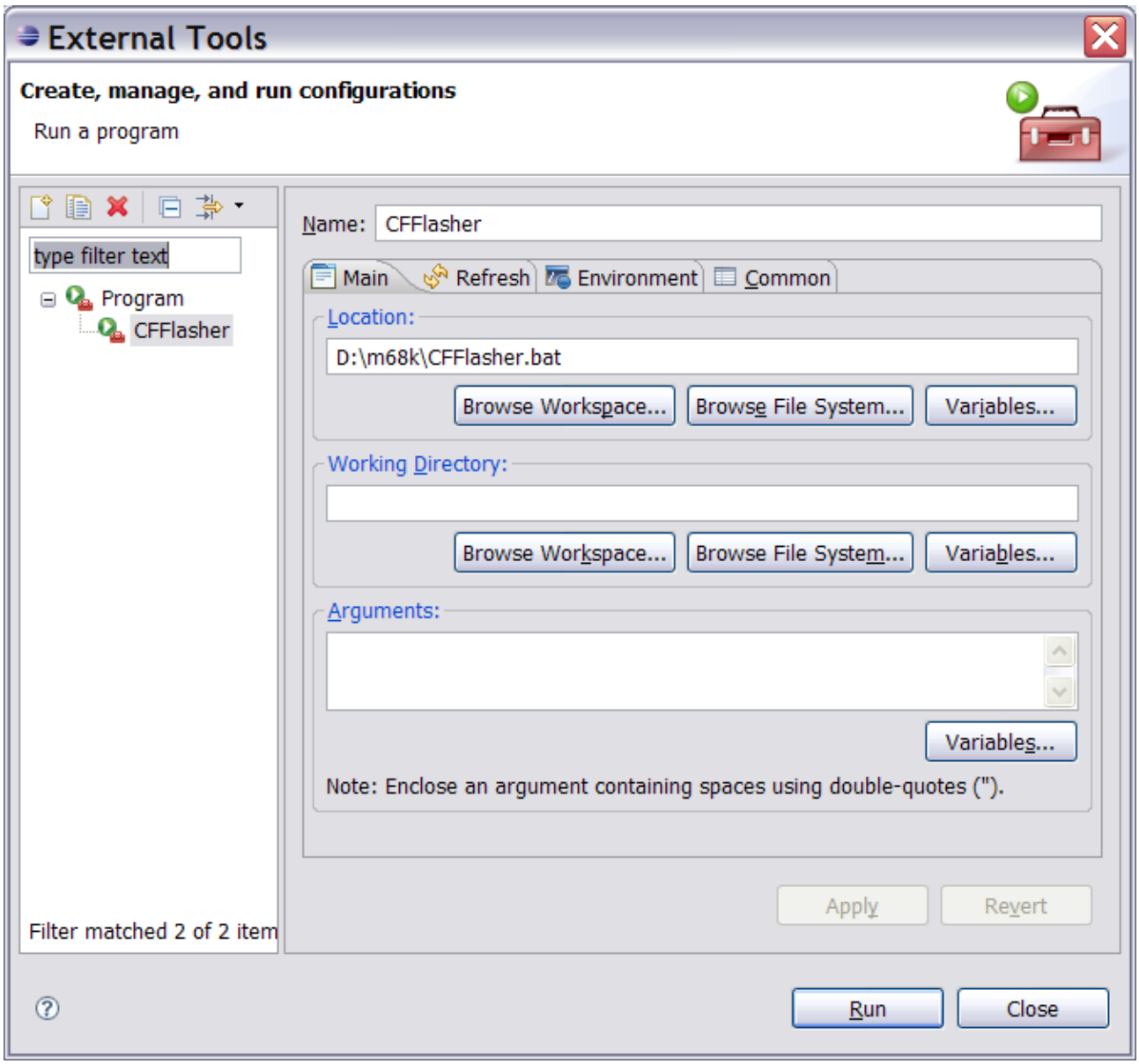
1. D'abord créer un fichier « CFFlasher.bat », qui contient :

```
cd D:\m68k
start CFFlasher.exe
```

(J'ai installé CFFlasher dans D:\m68k)

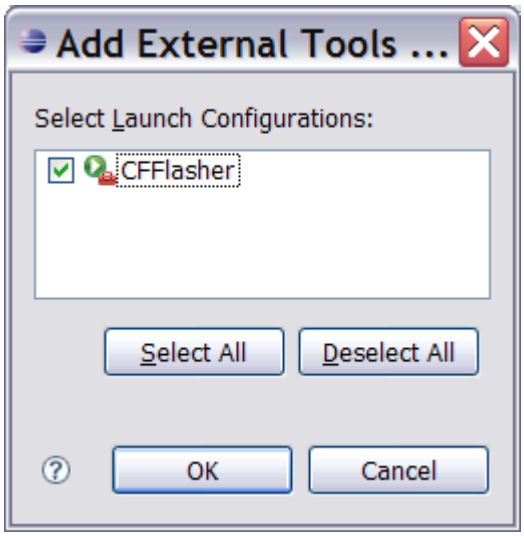
2. Ouvrir le menu « Run » --> « External tools » --> « Open External Tools dialog »

En cliquant deux fois sur « Program », s'ouvre une fenêtre que l'on complètera de la sorte (mettre par exemple CFFlasher dans « Name »; ce nom remplace automatiquement celui de New_Configuration »):



Cliquer sur « Close ».

Ensuite : (Ré-)ouvrir le menu : « Run » --> « External tools » --> « Organize Favrites » --> « Add » et dans la fenêtre, sélectionner « CFFlasher » :



Cliquer (Deux fois) sur « OK ». Quitter Eclipse et le relancer : CFFlasher apparaîtra directement dans le menu « Run » --> « External tools » ; d'où il pourra être lancé.

(-Fin du Complément non indispensable)

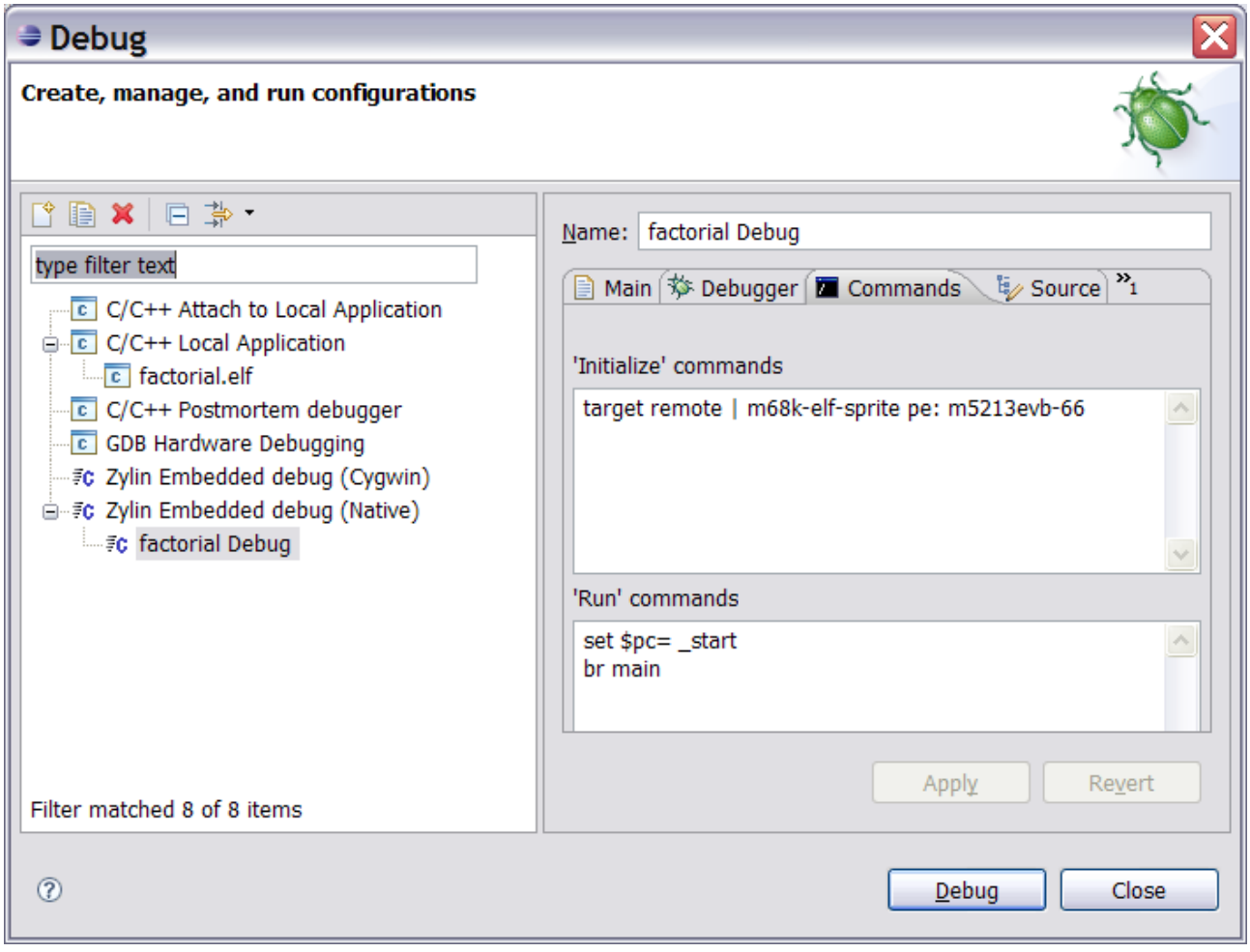
Ensuite :

1. Refaire un « Buid » du projet , en spécifiant bien cette fois, pour le link : **m68k-elf-gcc -mcpu=5213 -T m5213evb-66-rom-hosted.ld**

2. Charger le « Project.s19 » crée lors du buid dans la Falsh du Coldfire à l'aide de CFFlasher (ne pas oublier d'effacer avant de programmer)

3. Ouvrir la Perspective « Debug »;

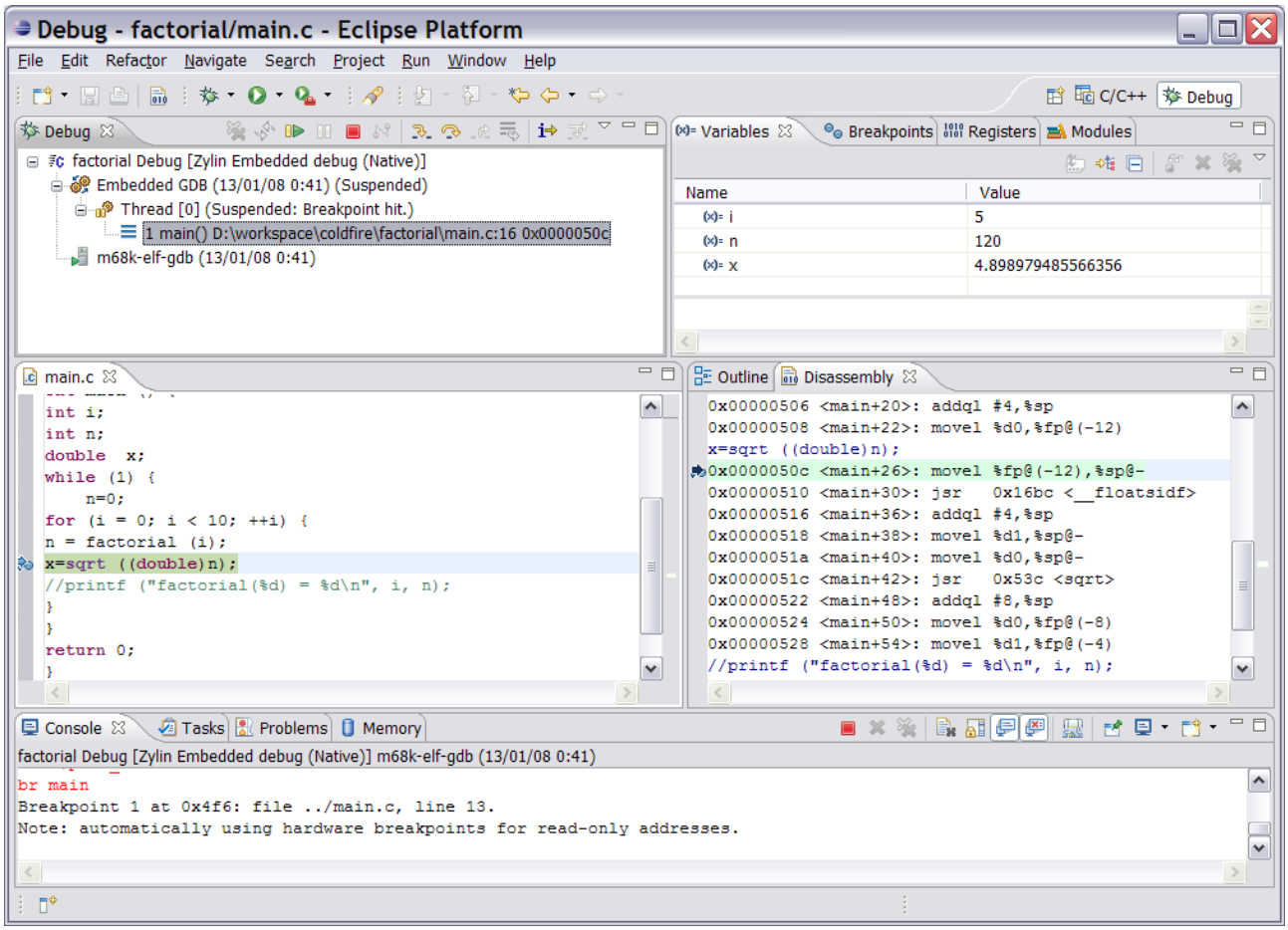
   Modifier les commandes d'initialisation de GDB de la manière suivante:



La commande « load » a été supprimée, ce qui est normal vu que l'exécutable a été chargé par CFFlasher

Cliquer ensuite sur « Debug »

CA MARCHE : on voir que le programme s'exécute à partir de la Flash : adresses 0x000....



Dernières notes et conclusion

Tout ceci a l'air bien compliqué, mais avec un peu d'habitude, cela devient de plus en plus simple et évident. Les différentes étapes ne sont faites qu'une seule fois par projet.
Il doit être possible aussi de créer des projets « template », afin d'éviter toutes les étapes de configuration.
La plupart des manipulations via les menus que j'ai décrites sont accessibles via des icones d'eclipse (par exemple le switch entre perspectives, le lancement du debug,..). Il faut cependant une certaine habitude, mais c'est nettement plus rapide.

- Eclipse est écrit en Java, donc son exécution est assez lente. Il faut donc un PC puissant; pas la peine d'essayer avec un Pentium 2 à 400MHz.
- Tout ce que j'ai décrit ci-dessus est parfaitement réalisable sous Linux. Télécharger les versions appropriées.
- Il y a d'autres programmes inclus dans la distribution GCC :
  - m68k-elf-objcopy : permet de convertir les exécutables en différents formats (en S19 par exemple).
  - m68k-elf-objdump : dump des symboles des objets, désassemblage,..
  - etc..
- Pour ce qui concerne les routines d'interruptions, voir le document « Getting Started » de CodeSourcery. Cela semble assez simple.
- L'utilisation de m68k-elf-gcc sans eclipse est parfaitement possible sous DOS, en ligne de commande et avec un éditeur standard pour les fichiers source.

- Eclipse peut être utilisé (simultanément) avec tout autre microcontrôleur supporté par GCC. Par exemple ARM, il suffira « juste » de paramétrer les options de compilation et du debugger. Il serait intéressant aussi d'essayer avec MC9S12 (http://www.gnu.org/software/m68hc11/m68hc11_pkg_zip.html)

Ce document est très loin de décrire toutes les possibilités de gcc et eclipse, qui sont en fait immenses. Les « header » en C des ColdFire sont disponibles sur le site de Freescale, et ne pas hésiter à se servir de ColdFire init de MicroAPL (http://www.microapl.com/CFInit/cfinit_main.html) pour initialiser les périphériques.

Pour approfondir GCC, beaucoup de docs sont librement téléchargeables sur internet : http://www.gnu.org/manual/ ; http://gcc.gnu.org/onlinedocs/

Pour comprendre le développement en C sous GCC, une compréhention basique de MAKE est utile.

Finalement, les pages précédentes résultent de longues semaines de galère et de recherche. Eclipse n'est pas simple ni intuitif, avec de très nombreux menus tout aussi tortueux les uns que les autres, mais il faut avouer que debugger son premier programme avec GDB rend aussi euphorique que le premier clignotement de led.

thierry