Chapitre 1

Premiers Pas sous TIGCC

Comme chacun sait, toute machine informatique ne travaille qu'avec des 0 et des 1 (en règle générale, le courant passe ou ne passe pas). Ainsi le langage de plus bas niveau (celui qui agit directement sur le processeur) n'est qu'une manipulation de 0 et de 1 : le langage de programmation qui s'en approche le plus est l'assembleur. En concevant un programme en assembleur, on n'écrit bien évidement pas avec des 0 et des 1 ; sur un ordinateur, on crée un fichier dans lequel on écrit notre programme avec des instructions de base telles que multiplier, comparer des valeurs, déplacer des données en mémoire... Ensuite, un logiciel appelé " Assembleur " va assembler le fichier (appelé le fichier source) dans le langage de la TI : les 0 et les 1. Mais l'assembleur est souvent difficile à comprendre et à manipuler, en particulier pour les débutants en programmation. Des langages d'un niveau plus haut (plus proche du "langage naturel", mais en anglais dans la majeure partie des cas) ont donc été créé : le TI-BASIC est un exemple parmi d'autres. Pour ce type de langage, il n'y a plus besoin de compilateur, la machine (la TI dans la cas qui nous intéresse) lit le programme tel quel (on dit qu'elle l'interprète). Le TI-BASIC a l'avantage d'être relativement stable (ne plante pratiquement jamais) mais est extrêmement lent. L'assembleur a les propriétés totalement inverses : rapide mais la difficulté à programmer entraîne trop souvent des plantages pour les débutants.

Une alternative à ces difficultés est le C : un peu plus complexe que le TI-BASIC mais pratiquement aussi rapide que l'assembleur.
Son secret est qu'après que vous ayez écrit votre programme en C dans un fichier, le compilateur va s'occuper de vérifier sa cohérence, puis le traduire en langage Assembleur ; celui-ci sera enfin assemblé en langage machine. Le langage C, parmi les trois disponibles pour programmer sur nos calculatrices, offre ainsi, au yeux d'un grand nombre de programmeurs, le meilleur rapport entre facilité de programmation et performances.
De plus, le langage C offre beaucoup plus de possibilités que ne le propose le langage TI-BASIC : par exemple les structures, les tableaux à plusieurs dimensions, ainsi que l'accès à quasiment tout ce que la machine permet de faire.
Notons tout de même une chose dont l'importance n'est pas grande lorsqu'il s'agit d'écrire des programmes de taille raisonnable, mais qui croit avec la complexité, et les besoins de rapidité et d'optimisation : en règle générale, pour des architectures limitées du genre de celle dont nous traitons dans ce tutorial, un bon programmeur en C produira un programme plus rapide et plus optimisé qu'un mauvais programmeur en langage d'Assembleur, mais un bon programmeur en ASM produira toujours un programme de meilleure qualité qu'un bon programmeur C. Cela tenant au fait que le programmeur ASM dit exactement à la machine quoi faire, alors que le programmeur C doit passer par un traducteur, le compilateur. Une solution souvent retenue par les programmeurs connaissant à la fois le C et l'Assembleur, est de profiter des avantages des deux langages, en écrivant la majeure partie de leur programme en C, pour la facilité et rapidité de développement qu'il permet, mais en programmant les routines critiques en ASM. (Mais cela implique de connaître le C, et de bien connaître l'ASM !).

I:\ Création d'un projet sous TIGCC IDE

Sous l'IDE fournie dans le pack de TIGCC, à chaque fois que l'on veut développer un programme, il faut commencer par créer un projet, qui contiendra les différents fichiers utiles à son écriture. Celui-ci regroupera bien sûr votre (ou vos) fichier(s) source(s) (*.c) ; il pourra, et nous le conseillons, contenir des fichiers textes (*.txt) dans lesquels vous pourrez mettre quelques notes ou des fichiers lisez-moi. Vous pourrez y ajouter vos propres headers (*.h), ou encore des fichiers contenant du code Assembleur (*.s pour de l'Assembleur GNU, et *.asm pour de l'Assembleur A68k, qui est traité dans le tutorial de la TCI sur le langage ASM).

A: Avec TIGCC 0.94 et versions précédentes

Pour créer ce projet, cliquez sur *File*, puis sur *New*, et enfin sur *Projet*.
Une fois le projet créé, il convient de lui ajouter au minimum un fichier source, dans lequel nous pourrons écrire le texte de notre programme. Pour cela, encore une fois, cliquez sur *File*, puis *New*, et ensuite, cette fois-ci, puisque nous souhaitons écrire un programme en langage C, choisissez "*C Source File*".
Le logiciel va alors vous présenter une série d'écrans avec des cases à cocher, qui vous permettrons d'affiner quelques options. Pour l'instant, laissez les options par défaut, qui permettent de créer des programmes tout à fait raisonnables, et cliquez sur *Next* à chaque fois. Au final, le fichier C est créé ; il ne reste plus qu'à lui donner un nom. Étant donné que c'est le fichier source principal du projet, vous pouvez, par exemple, le nommer "*main*" (l'extension est automatiquement rajoutée par l'IDE, et n'apparait pas).

A présent, il faut enregistrer le projet sur le disque dur; Pour cela, cliquez sur *File*, puis "*Save All...*". Choisissez ensuite où enregistrer le projet, et donnez lui le nom que vous voulez que votre programme ait au final, une fois envoyé sur la TI (8 caractères maximum, le premier étant une lettre, et les suivants des lettres ou chiffres).

Avec la version 0.94 SP4 de TIGCC, le code généré avec les options par défaut est le suivant :

```
// C Source File
// Created 03/07/2003; 18:46:33

#define USE_TI89 // Compile for TI-89
#define USE_TI92PLUS // Compile for TI-92 Plus
#define USE_V200 // Compile for V200

// #define OPTIMIZE_ROM_CALLS // Use ROM Call Optimization

#define MIN_AMS 100 // Compile for AMS 1.00 or higher

#define SAVE_SCREEN // Save/Restore LCD Contents

#include <tigcclib.h> // Include All Header Files

// Main Function
void _main(void)
{
  // Place your code here.
}
```

Exemple Complet

B: Avec TIGCC 0.95

Avec la version 0.95 de TIGCC, sortie début octobre 2003, on commence par créer un projet, en cliquant sur File, New, Project.
Une fois le projet créé, il nous faut créer, puisque nous souhaitons programmer en langage C, un fichier source C. Pour cela, cliquez sur File, New, "*C Source File*". Donnez un nom à ce fichier ; étant donné que c'est le fichier source principal du projet, vous pouvez, par exemple, le nommer "*main*" (l'extension est automatiquement rajoutée par l'IDE, et n'apparait pas).

A présent, il faut enregistrer le projet sur le disque dur; Pour cela, cliquez sur *File*, puis "*Save All...*". Choisissez ensuite où enregistrer le projet, et donnez lui le nom que vous voulez que votre programme ait au final, une fois envoyé sur la TI (8 caractères maximum, le premier étant une lettre, et les suivants des lettres ou chiffres).

Avec TIGCC 0.95, le code généré est le suivant :

```
// C Source File
// Created 08/10/2003; 14:17:20

// Delete or comment out the items you do not need.
#define COMMENT_STRING          "Place your comment here."
#define COMMENT_PROGRAM_NAME    "Place your program name here."
#define COMMENT_VERSION_STRING "Place your version string here."
#define COMMENT_VERSION_NUMBER 0,0,0,0
/* major, minor, revision, subrevision */
#define COMMENT_BW_ICON \
    {0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000}
#define COMMENT_GRAY_ICON \
    {0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000}, \
    {0b0000000000000000, \
     0b0000000000000000, \
     0b0000000000000000, \
```

```
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000, \
      0b0000000000000000}

#include <tigcclib.h>

// Main Function
void _main(void)
{
    // Place your code here.
}
```

Exemple Complet

Beaucoup de choses définies dans ce code ne nous servirons pas avant quelques temps... Nous allons donc les supprimer, afin d'avoir un code source plus petit (et donc un programme moins gros), ce qui nous donnera des exemples plus court, et plus lisibles !
En ne conservant que ce qui nous est actuellement utile, et nécessaire, nous obtenons le code source qui suit :

```
// C Source File
// Created 06/10/2003; 00:02:40

#include <tigcclib.h>

// Main Function
void _main(void)
{
    // Place your code here.
}
```

Exemple Complet

Lorsque nous fournirons des exemples de source dans ce tutorial, nous supposerons que vous avez créé un projet, ainsi qu'un fichier source C, et que vous avez, comme nous venons de le faire, supprimé tout ce qui est inutilement, à notre niveau, inséré par TIGCC.

Notez que, au cours de ce tutorial, nous supposerons que vous utilisez au minimum la version 0.95 de TIGCC.
Il sera assez rare, je pense, que des explications soient données concernant les versions antérieures, principalement du fait que la version 0.95 apporte beaucoup de nouveautés, et énormément de changements.
En fait, jusqu'au chapitre 8 de ce tutorial, il est possible que nous parlions encore de la version 0.94, en annexe de ce que nous dirons pour la version 0.95. Une fois le chapître 8 passé, il est fort peu probable que l'on fasse encore mention de la version 0.94, et, si le cas se produit, ce sera sûrement de façon anecdotique.
Cela est du au fait que les huit premiers chapitres de ce tutorial ont été rédigé avant la sortie de la version 0.95 (ils ont naturellement été revus, une fois la version 0.95 sortie, afin de lui correspondre), alors que les suivants l'ont été après sa sortie.

II:\ Compilation d'un projet sous TIGCC IDE

Enfin, pour créer votre programme exécutable pour la TI (une fois votre programme fini bien sûr), cliquez sur l'icone "*Make*", ou, si vous êtes un grand adepte des raccourcis clavier, appuyez sur la combinaison de touches [Alt+F9].
Le compilateur transformera votre fichier source en fichier exécutable pour TI, si votre source est sans erreur. En cas d'erreur, ou de choses que le compilateur juge douteuses, il apparaîtra peut-être des Warning ou des Errors. Les Errors indiquent qu'il y a une faute de programmation et que la compilation est impossible. Warning indique seulement que le programme présente un problème, mais que cela n'empêche pas la création du fichier exécutable. Cela dit, souvent, en cas de Warning, le programme plantera ou ne fera pas ce qu'il faut (En réalité, certains warnings sont sans importance, ou leur affichage peut être affiché supprimé en ajoutant quelques options de compilation. Cependant, tant que vous ne maîtriserez pas parfaitement la technique de compilation, et surtout dans les cas où vous ignorez la signification des warnings, nous vous encourageons fortement à prêter attention à TOUS ceux qui peuvent apparaître !). N'ayez par contre pas peur si vous voyez afficher une multitude d'erreurs : bien souvent une seule erreur très simple de programmation entraîne une dizaine d'incompréhension pour le compilateur.

III:\ Exécution du programme

Si la compilation s'est déroulée sans erreur, TIGCC aura créé un programme exécutable (fichier comportant l'extension *.89z, ou *.9xz, selon la machine). Vous pouvez à présent l'exécuter.
Pour cela, vous avez naturellement la solution d'envoyer le programme sur votre calculatrice, et de le lancer. Cela dit, lorsque l'on développe, en particulier lorsque l'on débute la programmation, il est fréquent que les programmes que l'on écrit soient buggués, et puisse planter la calculatrice.
C'est pour cette raison que nous vous encourageons à toujours utiliser un émulateur, tel VTI (Virtual TI Emulator) si vous êtes sous Windows, ou TI-Emu si vous êtes sous Linux, pour tester vos programmes, directement sur l'ordinateur.
De plus, envoyer un programme à l'émulateur est nettement plus rapide que de l'envoyer à la calculatrice.

Sous TIGCC-IDE, pour lancer un programme sous VTI, vous pouvez cliquer sur l'icone représentant une flèche verte, ou presser la touche F9. Naturellement, VTI doit être lancé pour que cela fonctionne ; et, de plus, il vaut mieux que vous ne touchiez pas au PC avant que le programme ne soit lancé sur VTI (utiliser le PC en cours de transfert de TIGCC vers VTI peut parfois empêcher le bon déroulement de ce transfert).

Nous n'entrerons pas plus dans les détails, et nous n'expliquerons pas ce que fait le code source présenté en exemple plus haut. Nous en reparlerons plus tard, mais avons besoin d'expliquer d'autres choses auparavant, je pense.

Nous avons, au cours de ce premier chapitre, vu comment créer un projet, un fichier source C, comment le compiler et l'exécuter. Au cours du chapitre suivant, nous discuterons de différents modes de programmation qui s'offrent à nous, afin de vous permettre de choisir celui que vous utiliserez généralement.
Une fois ceci fait, nous passerons à la création de notre premier, et simple, programme.