Chapitre 4

Appel d'un ROM_CALL

Maintenant que nous savons écrire, et compiler, des programmes ne faisant rien, nous allons pouvoir (mieux vaut tard que jamais !) réaliser un programme... faisant "quelque chose".
Principalement, pour commencer du moins, nous nous intéresserons à l'utilisation des fonctions intégrées à la ROM de la TI. Lorsque nous appellerons l'une de ces fonctions, nous réaliserons un "appel à la ROM", traditionnellement appelé "ROM_CALL". Par extension, et abus de langage (ça ne fait qu'un de plus... ça doit être ce qu'on appelle l'informatique :-)), nous emploierons généralement ce terme pour désigner les fonctions inclues à l'AMS en elles-mêmes.
L'ensemble des ROM_CALLs constitue une librairie de fonctions extrêmement complète, et qui, en règle générale, s'enrichit à chaque nouvelle version de ROM. Cet ajout de nouvelles fonctions peut être source d'incompatibilités entre votre programme et des anciennes versions d'AMS. (Jusqu'à présent, il n'y a que de très rares fonctions qui aient disparues, et celles-ci ont été supprimées parce qu'elles présentaient un danger potentiel pour la machine, tout en n'ayant qu'une utilisé limitée.) ; cependant, le plus souvent possible, nous veillerons à conserver la plus grande compatibilité possible.
A titre d'exemple, les ROM 2.0x sont toutes dotées de plus d'un millier de ROM_CALLs, qui permettent de faire tout ce que le TIOS fait ! Nous n'utiliserons qu'une infime partie de ces fonctions pour ce tutorial, et il est quasiment certain que vous n'utiliserez jamais plus du tiers de tous les ROM_CALLs ! (tout simplement parce que vous n'aurez pas l'usage des autres, à moins de développer des programmes un peu particuliers).

Il est possible de passer des paramètres à un ROM_CALL, si celui-ci en attend. Par exemple, pour une fonction affichant un texte à l'écran, il sera possible de préciser quel est ce texte ; pour un ROM_CALL traçant une ligne entre deux points, il faudra passer en paramètres les coordonnées de ces points.
Pour savoir quels sont les paramètres attendus par un ROM_CALL, je vous invite à consulter la documentation de TIGCC, fournie dans le pack que vous avez installé.
Nous verrons tout ceci au fur et à mesure de notre avancée dans ce chapitre...

I:\ Appel d'un ROM_CALL sans paramètre

A: Un peu de théorie

Comme nous l'avons dit plus haut, un ROM_CALL est une fonction inclue dans la ROM de la machine. Un ROM_CALL ne présente aucune différence, aux yeux du programmeur, qu'une fonction qu'il aurait écrit lui-même, si ce n'est qu'il ne l'a pas écrit.
Un ROM_CALL a donc un type de retour, et une liste d'arguments, exactement comme la fonction _main que nous avons étudié au chapitre précédent. Un ROM_CALL exécute une, ou plusieurs, action(s), sans que vous sachiez exactement comment il le fait (à moins d'être très curieux, et de désassembler la ROM... et encore vous faudra-t-il comprendre le code ASM donné par le désassembleur) : tout ce que vous avez besoin de savoir, c'est comment l'utiliser dans votre programme.

Une fonction n'attendant pas d'argument, est appelée en utilisant la syntaxe suivante :

```
nom_de_la_fonction();
```

Veillez à ne pas oublier le point-virgule en fin d'instruction, qui permet au compilateur de, justement, repérer la fin de l'instruction.

B: Un exemple simple

Sur nos TIs, il existe un ROM_CALL, nommé ngetchx, qui attend un appui sur une touche, et qui ne prend pas de paramètre. Nous allons écrire un programme, dans lequel nous utiliserons ce que nous avons dit au chapitre précédent, ainsi que la théorie que nous venons d'expliquer. Ce programme, une fois lancé, attendra que l'utilisateur appuie sur une touche (sauf les modificateurs, tels que [2nd], [<>], [shift], [alpha] sur 89, et [HAND] sur 92+/V200), et rend le contrôle au système d'exploitation de la TI (TI Operating System, abrégé en TIOS, qui est généralement utilisé comme synonyme d'AMS, ou de ROM).
Voila le code source de ce programme :

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

#include <tigcclib.h>

// Main Function
void _main(void)
{
    ngetchx();
}
```

Exemple Complet

Si vous compilez et exécutez ce programme, vous pourrez constater que, comme nous l'attendions, il attend une pression sur une touche, et, une fois la touche pressée, se termine.

II:\ Appel d'un ROM_CALL avec paramètres

A: Un peu de théorie

Appeler un ROM_CALL en lui passant des paramètres est chose extrêmement facile, une fois qu'on a compris le principe. Pour chaque ROM_CALL connu (et documenté), la documentation de TIGCC vous fourni la liste des paramètres qu'on doit lui passer. Il vous suffit de respecter ce qui est indiqué dans la documentation.
Pour appeler une fonction attendant des paramètres (on utilise aussi bien le terme de "paramètre" que celui "d'argument"), il suffit d'écrire son nom, suivi, entre parenthèses, de la liste des arguments.

B: Appel d'un ROM_CALL travaillant avec un quelque_chose *

Comme premier exemple, nous allons utiliser un ROM_CALL permettant d'afficher une ligne de texte dans la barre de status, en bas de l'écran. Ce ROM_CALL s'appelle ST_helpMsg. Il faudra, naturellement, préciser à la fonction quel message afficher ; nous le passerons en paramètre, comme la documentation de TIGCC précise que nous devons agir.
D'ailleurs, si l'on regarde ce qu'on appelle le "prototype" de ce ROM_CALL, on peut voir qu'il est comme reproduit ci-dessous :

```
void ST_helpMsg (const char *msg);
```

Comme nous pouvons le constater, ce ROM_CALL retourne void, c'est-à-dire rien.
Par contre, il attend un const char * en paramètre, ce qui correspond à une chaîne de caractères. Ne prêtez pas attention au nom msg donné à cette chaîne de caractère, il ne sert absolument à rien (et ne servira pas plus dans la suite de ce tutorial), et n'est là que pour que l'écriture soit plus "jolie".
En C, ce qu'on appelle chaîne de caractère correspond à du texte, délimité par des guillemets double.
Par exemple :

```
"Ceci est une chaîne de caractères"
```

Donc, pour appeler le ROM_CALL ST_helpMsg en lui demandant d'inscrire le message "Hello World !", il nous faudra utiliser cette syntaxe :

```
ST_helpMsg("Hello World !");
```

Afin de laisser le temps à l'utilisateur, on effectuera un appel au ROM_CALL ngetchx, afin que le programme ne se termine pas immédiatement.
Au final, notre code source sera celui-ci :

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

#include <tigcclib.h>

// Main Function
void _main(void)
{
    ST_helpMsg("Hello World !");
    ngetchx();
}
```

Exemple Complet

C: Appel d'un ROM_CALL attendant plusieurs paramètres, de type entier

Maintenant que nous avons vu les bases des appels de ROM_CALL avec paramètres, nous allons étendre notre connaissance au passage de plusieurs paramètres, et, pour varier, nous les prendrons cette fois de type entier.
Nous travaillerons ici avec le ROM_CALL `DrawLine`, qui permet de tracer une ligne entre deux points de l'écran, et qui nous permet de choisir le mode d'affichage que nous souhaitons.

Voici la façon dont ce ROM_CALL est déclaré dans la documentation de TIGCC :

```
void DrawLine (short x0, short y0, short x1, short y1, short Attr);
```

Pour l'appeler, nous agirons exactement de la même façon qu'avec le ROM_CALL ST_helpMsg : nous préciserons les arguments, dans l'ordre indiqué.
Par exemple, pour tracer une ligne entre les points de coordonnées (10 ; 30) et (70 ; 50), en mode Normal, c'est-à-dire en noir sur blanc, avec une épaisseur de trait de un pixel, nous utiliserons cette instruction :

```
DrawLine(10, 30, 70, 50, A_NORMAL);
```

Les différents modes de dessin pour cette fonction sont précisés dans la documentation de TIGCC, justement à l'entrée correspondant au ROM_CALL DrawLine.
Les coordonnées sont données en pixels, à partir du coin supérieur gauche de l'écran, qui a pour coordonnées (0 ; 0).
Le coin inférieur droit de l'écran a pour coordonnées, sur TI-89, (159 ; 99), et, sur TI-92+ et V-200, (239 ; 127).

Je pense que nous en avons assez vu pour ce chapitre. Vous pouvez, et je vous encourage à le faire, vous entraîner à utiliser d'autres ROM_CALL, tels, par exemple, DrawStr, qui permet d'afficher un texte à l'écran à la position que l'on désire (il est fort probable que, de toute façon, nous étudions ce ROM_CALL plus loin dans ce tutorial, de part sa grande importance), en vous basant sur la documentation de TIGCC ; cela ne peut vous faire que du bien. Je vous conseille cependant de tester vos programmes sur VTI, afin d'éviter toute mauvaise surprise.
Au chapitre prochain, nous commencerons par voir comment effacer l'écran avant de dessiner quelque chose, afin de rendre nos affichages plus jolis, puis nous verrons comment il se fait que l'écran soit redessiné en fin de programme, et comment empêcher ceci.