Chapter 4

Calling of a ROM_CALL

Translated by Spectras ; Many thanks to him/her! - Website: http://www.tiwiki.org/

I:\ Calling AMS functions

We have reviewed all the process of setting up a project and creating a basic program that does nothing. It is now time to make the program do something useful. Let's begin with a simple example:

```
#include <tigcclib.h>

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

Try copying this code into your project, compiling it and running it on your calculator. You should notice that your program keeps running until you press a key. It is the intended behaviour: *ngetchx* is one of the functions provided by AMS. (Be careful, the C language is case-sensitive. That is, lowercase and uppercase letters are not the same, thus *ngetchx* is not the same as *Ngetchx* or *NGETCHX*). Notice how the function name is followed by empty parenthesis. Thoses parenthesis tell the compiler to run the function, with no parameters. Finally, the ; symbol tells the compiler the current instruction is finished. It is mandatory, since the C language does not take blanks nor line limits into account; the ; is the only instruction delimiter it knows.

Thus the previous code sample executes the ngetchx romcall, which the documentation says takes no parameter, waits for a keypress and returns the value of the key that was pressed. Well, it works exactly that way, though we simply discard the return value. There are many such AMS functions (usually called romcalls because they live in the read-only part of the memory) covering all ranges of TI-89 features, like memory management, filesystem, link, drawing on the screen, etc. New functions appear in almost every new version of the AMS. Using them, however, breaks the compatibility with older AMS versions, something you will want to be careful about since not all users update there AMS on a regular basis.

II:\ Functions that take parameters

Most romcalls take parameters, since they have to know what data they have to work on. For instance, a function computing a square root has to know which value you want the square root of. For any romcall one can get the list of required parameters in the documentation, along with the description of what parameters do, and what prerequisites exist on each of them, if any. One then calls the function the usual way, supplying the parameters in the parenthesis, separated by commas. Here is an example of the declaration of a romcall taken from the documentation:

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

We see this romcall takes a string (char*) parameter and returns nothing. The *const* modifier states that ST_helpMsg will not modify the string we supply (although it could make an internal copy and modify it). The documentation explains that the *msg* parameter is a message to be displayed in the status line of the calculator. We can now call it this way:

```

#include <tigcclib.h>

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

Notice the string must be enclosed in double quotes. This program will display the message "Hello World!" (without the quotes) in the status line, then wait for a keypress.

III:\ Another example

This code sample demonstrates the use of some drawing romcalls, namely DrawLine and DrawStr. A quick lookup in their documentation shows their declaration:

```
void DrawLine (short x0, short y0, short x1, short y1, short Attr);void DrawStr
(short x, short y, const char * msg, short Attr);
```

The x and y parameters should be obvious. Just keep in mind that (0,0) is the upper-left pixel, not the lower-left one, and that the lower-right pixel is at (159,99) on TI-89s and (239,127) on TI-92s. The *msg* parameter for DrawStr is the message to draw onto the screen. The Attr parameters are a bit more tricky. The documentation says it changes the way the line and string are drawn. Using the special value A_NORMAL will make the pixels on the line black, which is what we want here.

Here is a simple example, feel free to try changing the different parameters and see the effects:

```
#include <tigcclib.h>

void _main(void)
{
      DrawLine("Hello World!");
      ngetchx();
}
```

You have probably taken notice of the fact that the screen is not cleared at the beginning of your program, but gets, however, refreshed at its end, preventing you from exiting without waiting for a keypress. This will be explained and solved in the next chapter.