Chapter 5

Clearing the Screen ; Printing a Message on Quitting

This page is being translated.
Translated by Spectras ; Many thanks to him/her! - Website: http://www.tiwiki.org/

Now we can draw onto the screen, we will see how to erase the screen at the beginning of our program, and why the screen gets restored automatically at the end of our program.

I: \ Erasing the screen contents

Erasing the screen contents is very easy, as AMS provides a simple romcall taking no parameters. ClrScr, standing for 'Clear Screen', blanks all pixels. We can then make nice drawings without the ugly AMS home screen in the background.

Here is an example that draws a single line and waits for a keypress:

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

// Main Function
void _main(void)
{
      ClrScr();
      DrawLine(10, 30, 70, 50, A_NORMAL);
      ngetchx();
}
```

This program runs as it should. Also, you will notice that the screen contents get restored automatically after you press a key, despite the absence of any instruction to do so in your source code. Actually, your binary program does contain the code that does this. Tigcc generates it for you automatically because most program need it, for AMS does not redraw the screen by itself.

II : \ Manual control of the screen save and restore operations

Screensaving code generation is triggered by the definition of a specific symbol, SAVE_SCREEN. The compiler detects its presence and generates the additional instructions accordingly. There are two ways to define this symbol : either in the compiler options or with a special preprocessor directive.

- The compiler option is actually a flag on the compiler command line, but TIGCC IDE provides a simpler way to configure the command line. One can go through the project options menu, select the compilation tab then click the Program Options button. There can be found a list of checkboxes, one of them labelled as 'Save/Restore LCD Contents'. It is checked by default, which us why our program did it. To take over the process or remove it completely, simply uncheck the box.
- The special preprocessor directive should be at the top of the main source file, and read: '*#define SAVE_SCREEN*'. It will force screensaving code whether or not the box is checked in the project options, though one should avoid using both at a time.

You can try unckecking the box, compile your program again, and make sure your program leaves a garbled screen. Well not completely, since parts of the screen are redrawn, and one can restore almost all the other parts by using the calculator a bit. Still, it is not very satisfying for the user (and the bottom line separating the status bar from the command line editor cannot be restored this way).

So, you are wondering why one would want to disable the automatic screen restore. Well, there are quite a lot of reasons. Suppose you want to exit your program and leave an error message in the status line, for instance under critical error conditions. The message will not be visible, since the automatic screen restore will overwrite it. You could wait for a keypress, but it is not as practical. Here is the solution.

Firstly, the automatic screen restore must be disabled, since we are going to handle it by ourselves. So we have to save the screen ourselves and restore it before we write our message. We will use a LCD_BUFFER variable for this purpose. We declare it thusly:

```
LCD_BUFFER saved_screen;
```

This line tells the compiler we want a new variable that will fit a screen capture, and that it shall be named *saved_screen*. Therefore, any further reference to *saved_screen* will be interpreted as references to this variable. Variables will be explained in another chapter. We then use the *LCD_save* function to dump the screen contents into our variable, this way:

```
LCD_save(saved_screen);
```

At this point, the original screen contents is safe, and we can draw anything we like onto the screen. Once we are done, we restore the original screen contents from our variable using *LCD_restore* this way:

```
LCD_restore(saved_screen);
```

At this point, our program will work exactly the way it used to when we let the compiler generate the code that saves and restores the contents of the screen. Except we can add other instructions after we restore it. For instance, we can display a message in the status line, and it will remain visible even after our program exits. We do this using the usual function:

```
ST_helpMsg("Goodbye");
```

And finally, a complete program you can try on your own calculator:

```
#include <tigcclib.h>

// Main Function
void _main(void)
{
      LCD_BUFFER saved_screen;
      LCD_save(saved_screen);
      ClrScr();
      DrawLine(10, 30, 70, 50, A_NORMAL);
      ngetchx();
      LCD_restore(saved_screen);
      ST_helpMsg("Goodbye");
}
```

When run, this program saves the screen contents, then draws a line on a blank screen and waits for a keypress. When it comes, it restores the original screen contents, and leave a message for the user to see after the program exits. Status line messages disappear as soon as the user presses a key.

[Translation not finished yet]