FoldMaster

Macro Tutorial

An Introduction to Writing Macros for FoldMaster

FoldMaster the Folding Editor
Internet: http://www.foldmaster.de
E-Mail: support@foldmaster.de

Issue 1. from 12/16/2006

Armin Raible
Bernhardstr. 47
88677 Markdorf
Germany

and

Michael Leute
Rathausplatz 6
88697 Bermatingen
Germany

Table of Contents

1 Scope

This document describes how to write macros for the FoldMaster editor. Writing macros for FoldMaster is easy. FoldMaster is equipped with a powerful - Pascal like - macro language. Nearly all commands which can be performed manually using the mouse and keyboard can also be done by a macro. This document is not an introduction in Pascal. Basic knowledge about Pascal is assumed. But if you are not familiar with Pascal itself, but with any other programming language, it is easy and fast to learn.

For more detailed information of all the features and possibilities of FoldMaster refer to the online help. Further information and a short tutorial will be found on the FoldMaster website at http://www.foldmaster.de.

2 System requirements

For work through this tutorial the following minimum system requirements are recommended:

- FoldMaster V1.63 or newer (licensed or demo)
- 90 MHz compatible Pentium-class computer
- 32 MB RAM
- 10 MB of free hard disk space
- MS Windows 95, 98, ME, Win2000, XP or NT

3 Introduction to a FoldMaster Macro

3.1 Viewing the Macro List

To view a list of all current macros known by FoldMaster select:

Macros > Show modules…

The window *Edit Macro Modules* opens, containing a list of all macros currently known by FoldMaster. The Icon in front of every macro indicates the current status of it:

No tick: The macro is only known by FoldMaster, ready to edit.

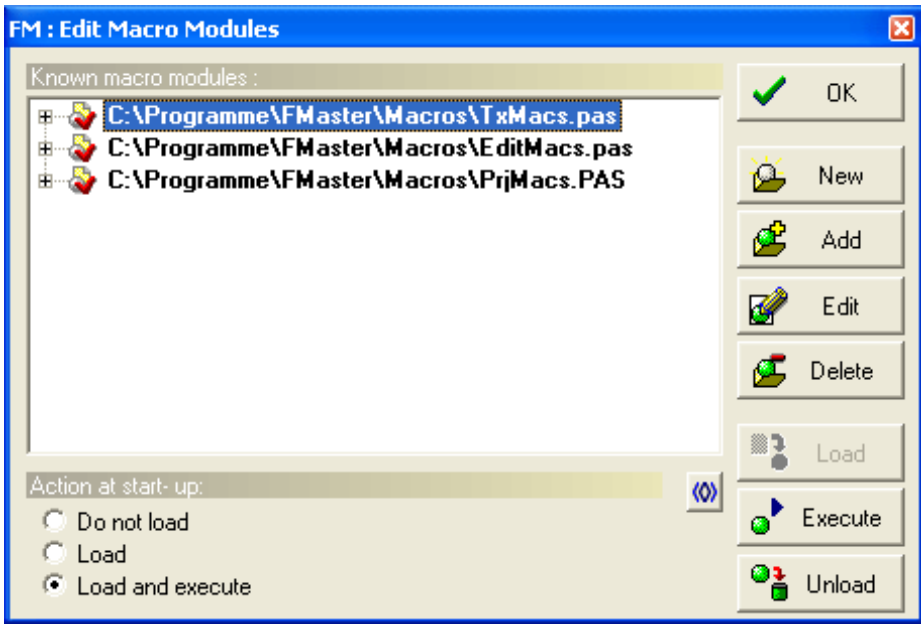
Blue tick: The macro has loaded and compiled.

Red tick: The macro has loaded, compiled and executed.

The option *Action at start up* defines for every macro individually what happens at the start up of FoldMaster. So it is possible to load, compile and execute a macro at start up, providing the contained functions automatically to the user. For example some user defined hot keys can be added.

Note: If you have no valid license (running only the demo version) the macros are limited to the three provided modules. Nevertheless you are able to do the tutorial, but after a FoldMaster restart the list of known macro modules will only

contain the three provided macros. The licensed version do not have any restrictions.



The buttons on the right have the following functions:

OK Closes the *Edit Macro Modules* window.

New Let you create a new macro module or include file.

Add Opens the *Insert Macro Module* window to add new macros to the list.

Edit Opens the selected macro from the list in the macro editor.

Delete Removes the selected macro from the list. The macro file itself will not be deleted.

Load Loads and compiles the selected macro.

Execute Runs the selected macro. The macro has to be loaded and compiled.

Unload Unloads the selected macro. The macro has to be loaded and compiled. The functions of the macro are no longer available.

Toggles between the normal and a reduced version of the window. In the reduced version only the name of the macro modules will be displayed. All buttons are only show symbols. This mode saves space on the desktop.

3.2 Installing a Macro

Installing a new macro (for example one provided at the FoldMaster homepage) is simple:

Copy the macro file and maybe some include files and / or bitmaps for buttons, into the macros subdirectory of the FoldMaster installation directory. Normally this would be `C:\Programme\fmaster\macros`. Of course you can use any other place on your hard disk too.

In FoldMaster select: **Macros > Show modules…** The *Edit Macro Modules* window opens.

Click on the **Add** button to open the *Insert Macro Module* window. Browse to the macro file, select it and click the **Add to** button, to add the macro to the list of known macros. The

window doesn't close automatically, allowing you to add another macro. When ready, close the window.

The new macro appears in the list of the known macro modules. Select it with a single click and click on the **Load** button to load and compile it.

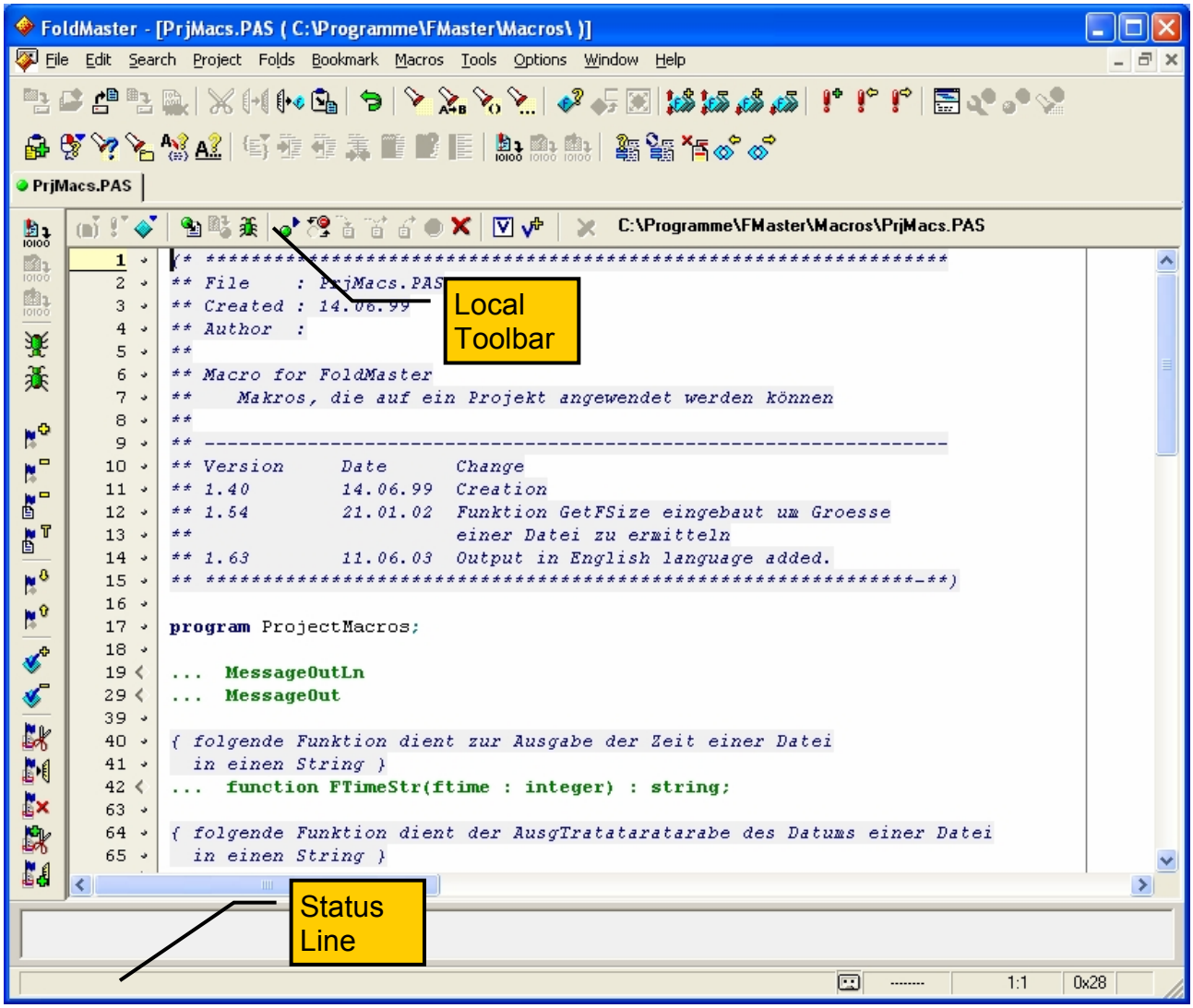
After successful compilation click on the **Execute** button to run the macro.

Important: If you want FoldMaster automatically load, compile and run the macro at start up, select the *Load and execute* option. This is useful to provide the macro functions immediate after start up.

Click on **OK** to close the *Edit Macro Modules* window.

3.3 Open a Macro in the Macro Editor

To open an existing macro in the macro editor select the macro in the *Edit Macro Modules* window and then click the **Edit** button. The macro editor opens and loads the macro.



At the first view it may look like any other editor window. But on the upper side you can see the local toolbar containing the buttons of the macro editor to compile, run and debug the macro.

The local toolbar of the macro editor:

The buttons provide the following functions:

Inserts a macro function. A window appears showing all available macro commands. After selection, any command can be inserted at the current cursor position. The macro commands are sorted in different groups:

EdCmd	Editor commands.
EdPF	Editor procedures and functions.
StdP	Standard procedures.
StdF	Standard functions.
Math	Arithmetic functions.

This function is very useful for browsing through all macros especially if you don't know the exact name of a macro function.

Compiles the macro. If any error occurs, a window containing a description of the error appears, and the cursor is placed at the error.

Enables the debugger. The macro can be run in single step and variables can be displayed.

Runs the macro. If the debugger is disabled the macro is executed at full speed. With the debugger enabled, the macro can now be executed in single step mode using the single step button, or in fast step mode by clicking the run button again. In fast step mode the observed variables are updated after every macro command. In this mode you can watch the macro working.

Sets a breakpoint at the current cursor position. The execution of the macro stops at the breakpoint and the debugger is enabled.

Single step. Let you execute the macro step by step and observe the variables.

Steps over a function or procedure. The function or procedure is executed at full speed, avoiding long single step sessions into proved functions or procedures.

Steps out of the current function or procedure. The remaining part of the current function or procedure is executed in fast step mode (the variables can be observed). The macro is stopped at the first command outside the function or procedure.

Stops the macro execution in fast execution mode for further debugging.

Cancels the macro. The running macro is terminated. This is useful if the macro hangs in an infinite loop.

Opens the watch window inside the debugger. You can resize the window using the mouse.

Adds the variable at the current cursor position to the watch window. To remove a variable from the watch window right click on it and select **Remove Watch**.

Now close the macro editor by selecting the cross in the upper right of the macro editor.

4 Writing the First FoldMaster Macro

4.1 Creating a New Macro

If not still open, open the *Edit Macro Modules* window by selecting **Macros > Show modules…**.

In the *Edit Macro Modules* window select the **New** button. You are asked either to create a macro module or a include file. Select the option *Macro Module* and click **OK**. Now browse to a directory of your choice (normally the macros directory, for example `C:\Programme\fmaster\macros`) and enter a file name for the macro. For this tutorial enter `mactut.pas` and click on the **Save** button.

FoldMaster now creates a new macro using the template from the Pascal syntax filter. The syntax of the macro language is Pascal with many special extensions providing control over the editor.

4.2 Writing the First Statements

Now we write a simple macro for demonstration purposes only, having no useful function. You can do either, type the program lines into the provided template, or clear the file and type the lines into the editor without any folding.

First we define the global variables under the VAR section:

```
VAR
  i,
  y  : INTEGER;
```

Then we define a little function, using local variables:

```
FUNCTION TestFunction(a: INTEGER) : INTEGER;

VAR
  i,
  sum : INTEGER;

BEGIN
  FOR i := 1 TO a DO BEGIN
    sum := sum + i;
  END;
  TestFunction := sum;
END;
```

The main program consists of the following statements:

```
BEGIN
   i := 0;                    //for easy debugging
   y := 0;
   MacroConsoleOpen;          //open the macroconsole
   WriteLn('Hello world!');   //write to the macroconsole
   i := 1000;
   y := TestFunction(i);      //call the TestFunction
   WriteLn('f(',i,') = ',y);  //write the result
END.
```

While writing a command, FoldMaster displays all possible commands at the status line. For example, after writing `wri` the status line displays `Write | WriteLn` as possible commands. Using the F8 key, FoldMaster auto completes the command to the first of the list. If enabled, in the *Help on Parameters* window some useful information about the parameter used by the current command will be displayed.

To save the macro to disc select **File > Save**, or Ctrl + s, or use the  (save file) button from the toolbar.

4.3 Compiling the Macro

To compile the macro click on the  (compile macro module) button. The macro should compile without any error. If not, correct the typing error and try again.

Congratulations! You successfully written your first FoldMaster macro.

4.4 Running the Macro

Click on the  (run macro) button to run the macro.

The macroconsole opens and gets focus, displaying the following text:

```
Hello world!
f(1000) = 500500
```

The macroconsole is the standard output for any write from a macro. This is useful to keep the user informed about the status of a macro, especially if a macro runs very long.

5 Using the Macro Debugger

5.1 Invoking the Debugger

Invoke the debugger by selecting the  (enable debugger) button from the macro editor toolbar. The locked button shows that the debugger is active. Now run the macro again. The macro will not be proceed at full speed as before, execution is stopped in front of the first statement and the statement is highlighted.

Now step through the macro using the  (single step) button. Every time you click on the single step button, one statement is executed.

5.2 Watching Variables

If you reach the third statement (`i := 1000;`) stop and invoke the watch window by selecting the  (open watch window) button from the local toolbar. Resize the window as required using the mouse. Place the cursor at the `i` variable. To add the variable to the watch window select the  (add to watch) button. The variable at the current cursor position is added for watching. In this way add all other variables of the macro, including the local ones of the function.

Note that the local variables of the function are not available yet and the local `i` is not identical to the global one.

To remove a variable from the watch window, select it with a left click, then right click on it and select **Remove Watch**.

Stepping now through the macro you can see the variables changing.

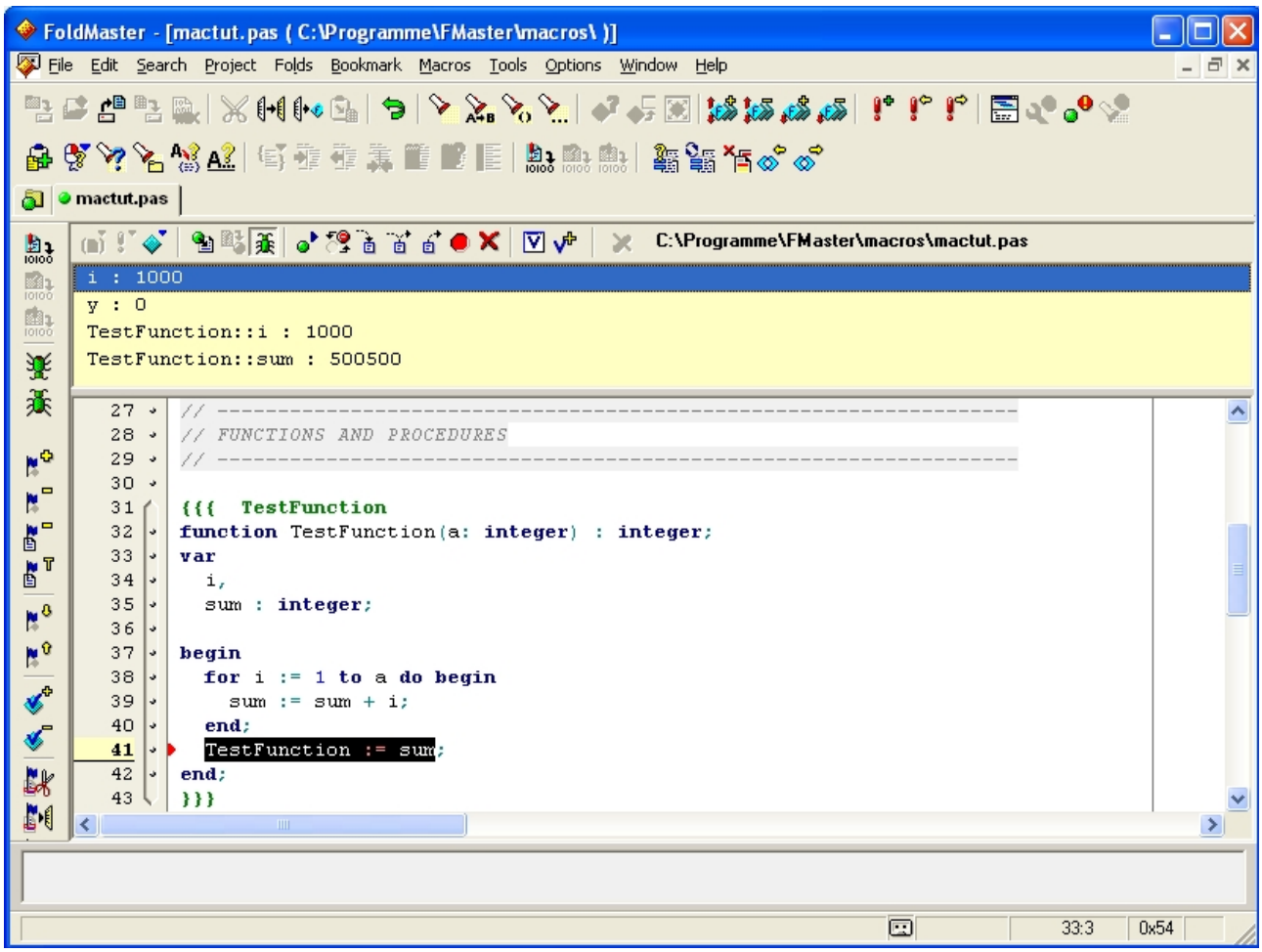
5.3 The Fast Step Mode

If you click on the  (run macro) button while watching any variables, the macro will be executed in fast step mode. In fast step mode the observed variables are updated after every macro command. In this mode you can watch the macro working through its end, or reaching the breakpoint. To avoid this and run the macro at full speed instead, you must disable the watch window by clicking on the  (open watch window) button again. The viewed variables are not lost. Open the watch window again, displays all the earlier variables.

5.4 Using the Breakpoint

If debugging long macros it could be very boring and hard work, too, reaching the point of interest using the single step function. Using the breakpoint avoid this problem.

Place the cursor at the line you wish to set the breakpoint and click on the  (set breakpoint) button. For example place the cursor on the last line inside the TestFunction and click on the *set breakpoint* button. The line will be marked at the left side with a little red arrow, indicating the breakpoint.



Now run the macro again, using either normal run or fast execution mode. The macro will stop running before the execution of the line with the breakpoint. The debug mode is entered, allowing further execution in single step or fast step mode.

6 Conclusion

6.1 Further Macro Examples

On the FoldMaster homepage at www.foldmaster.de at the *Macros & Tips* section you will find many macros providing various functions. They can also serving as examples for own macros.

The Keymap.zip for example containing a macro providing some functions which are of special interest for a beginner in FoldMaster macros. It shows how to link macro functions with hotkeys or place user defined buttons to the toolbar.