Installation Guide and Technical Reference OF THE RAMSES SOFTWARE

VERSION 3

FOR APPLE® MACINTOSH® COMPUTERS[1]

Andreas Fischlin[2][3], Dimitrios Gyalistras[3] and Thomas J.Löffler[4]

July 2006[5]

Zürich/Schlieren, SWITZERLAND

Contents

[1]Macintosh is a registered trademark of Apple® Computer, Inc.

[2]To whom correspondence is to be addressed

[3]Terrestrial Systems Ecology Group, Swiss Federal Institute of Technology (ETH Zurich), Universitätstr. 16, CHN E 35.1, CH-8092 Zurich, SWITZERLAND. Tel +41-(0)1-633 60 90, Fax +41-(0)1-633 1031, E-Mail: mailto:RAMSES@env.ethz.ch

[4]Swiss Federal Institute of Technology (ETHZ), Institute of Geology, Department: Earth Sciences, Universitaetstr. 6, CAB E 10.4, CH-8092 Zuerich, SWITZERLAND

[5]This text refers to RAMSES version 3.4.0 or newer

1 What is RAMSES

RAMSES is an acronym for **R**esearch **A**ids for **M**odeling and **S**imulation of **E**nvironmental **S**ystems (FISCHLIN, 1991). It can be best understood as a rather huge box containing many, many tools useful for a large number of purposes such as modeling and simulation. In addition it contains many useful software tools to develop interactive programs, macro editors, compilers, statistical tools etc., in short software which is usually of use to scientists and engineers in their daily research. To learn more about RAMSES install it first (see below) and read the on-line helps. RAMSES has also a home page at

http://www.sysecol.ethz.ch/RAMSES/

Parts of the RAMSES software are also available for other computer platforms than the one described in this document, i.e. the "Dialog Machine" (FISCHLIN *et al.*, 1987; FISCHLIN & SCHAUFELBERGER, 1987) and ModelWorks (FISCHLIN *et al.*, 1994) for the IBM-PC[6] and the Atari Computer, and the RAMSES simulation server RASS (THOENY *et al.*, 1994) is available for Sun[7] workstations and Mac OS X machines. There exist separate documents explaining the installation and providing a technical reference to those RAMSES packages.

[6]IBM is a registered trademark of International Business Machines Corporation.

[7]Sun is a registered trademark of Sun Microsystems, Inc.

2 Installation Guide

2.1 Installation

The RAMSES software is available via internet from

http://www.sysecol.ethz.ch/SimSoftware/

or from the more compact download table at

http://www.sysecol.ethz.ch/SimSoftware/SimSoftware2.html

if you favor overview over explanations. Alternatively you can use anonymous file transfer ftp (File Transfer Protocol) from the host "ftp.ito.umnw.ethz.ch" in the ftp directory "pub/mac/RAMSES". The installation is simply an unpacking or copying of all the software onto any target machine. There is no other installation necessary and all your System Software, preferences etc. remain untouched. You find more details on this topic either on the internet

http://www.sysecol.ethz.ch/RAMSES/READ_ME.RAMSES.html

or in the file ***On RAMSES (READ ME!).pdf***. You find a second copy of this document also redundantly in the folder *Docu* of the RAMSES package.

Important note: Please NEVER nest the RAMSES folder within folders with long names. Instead try to keep all involved folder names as short as possible! Also try to avoid any nesting. E.g. put the folder *RMS* which results from the default installation process directly onto your hard disk, e.g. named only *HD*. The absolute path will remain short, i.e. "HD:RMS:". Such an installation will help to avoid the path too long problem. Note, normally MacMETH and RAMSES will use relative path, yet in many instances it is unavoidable to use absolute paths. Then it is advantageous to having used short names only. Remember: Not only keep folder names short within the RAMSES folder, but also those of the enclosing folders, including the name of the disk on which the RAMSES folder resides.

If you use this software under Mac OS X you have to install also Classic. Classic is available from your Mac OS X software installation CD or DVD, but requires a separate installation step[8]. Should you use an Intel based Macintosh computer, note, however, unless a Classic emulation becomes available , this variant of the RAMSES software will not run on such a Macintosh computer. However, it runs on any other Macintosh machine, regardless whether they are 68'000 (68K) or Power PC (PPC) based machines.

2.2 Getting Started

2.2.1 A Quick Intro for Programmers

It is important to be aware of the fact that RAMSES consists of 4 software levels, where each level forms a subset of the next higher level:

1) The first subset is MacMETH (no difference to the MacMETH you got from the separate release).

2) The second subset, based on MacMETH, is the 'Dialog Machine' (abbreviated DM), which goes along the same path Apple started with its toolbox, but which goes beyond the point Apple quit that path. This is the level, I think which makes the big difference and which is probably exactly what you need. The 'Dialog Machine' makes the toolbox obsolete, since it offers a very-high level modern, graphical user interface in a machine independent way. In the Macintosh implementation it fully and automatically conforms e.g. to the Apple's User Interface Guidelines (in contrast to almost all commercial, let alone other applications I know). Moreover, the 'Dialog Machine' does not require the programmer to understand the low level toolbox routines and to study first the whole Inside Macintosh before being able

[8]double-click the installation package Classic and follow the instructions. If your installation software does not include Classic, you can obtain it from other sources. Visit the RAMSES home (http://www.sysecol.ethz.ch/RAMSES) for latest information on the availability of Classic.

to write a decent interactive program. You are capable to program already quite fancy interactive programs with only about 30 DM-functions (see the article 'Introduction to the Dialog Machine' by Daniel Keller).

Furthermore, DM programs can be ported to other machines, like the IBM-PC under Windows with almost no changes. We have ported bigger software projects, e.g. consisting of more than ca. 50'000 lines of source code, from the Mac to the IBM PC (besides, also back to the Mac), within a couple of days. Up to now the 'Dialog Machine' has been successfully implemented on all Macs, on IBM PCs (under GEM (>=80286) and Windows (>=80386)), on Ataris, and on Suns (albeit only a prototype).

3) The third subset, a typical DM application, is ModelWorks (abbreviated MW), an interactive modeling and simulation environment.

4) The superset of everything is called RAMSES, a programing, modeling, simulation, and post-simulation analysis environment. It is again a DM application and uses currently also ModelWorks for the modeling and the interactive simulation sessions.

Depending on your needs, you can stop with any subset you want and ignore the higher supersets. So if you look for a programing environment, which allows to develop interactive, user friendly applications, use the 'Dialog Machine'. You won't have to think about the toolbox anymore, just think in terms of menus, windows, entry forms (modal dialog boxes), files and other objects. The basic philosophy is almost OOP (Object Oriented Programing), the only difference is inheritance, which is difficult to support in standard Modula-2.

To get acquainted with the RAMSES shell quickly follow these steps:

First a few preparatory steps: Start the RAMSES shell. It will launch the sample model 'Logistic'. But since this of little interest to you, perform the following changes to the RAMSES shell: Press simultaneously the keys Command^Shift^Capslock^B (B stands for big shell, i.e. you will leave the Mini shell and will enter the big shell, again in the simulation session with the same work object, i.e. 'Logistic'. Besides, the involved concepts are explained in the On-line Help of the Mini RAMSES shell, e.g. select command 'Help' in menu 'Shell'). Now select in the menu 'Shell' the command 'Programing' to quit the simulation session and to enter the programing session. This session acts similar to the MacMETH shell, the main difference is that it is more convenient to use under System 7 and that it is more efficient when loading large programs. Otherwise it is the similar to the MacMETH shell and therefore all user instructions contained in the MacMETH manual apply also.

Here comes a quicky to get acquainted with the 'Dialog Machine':

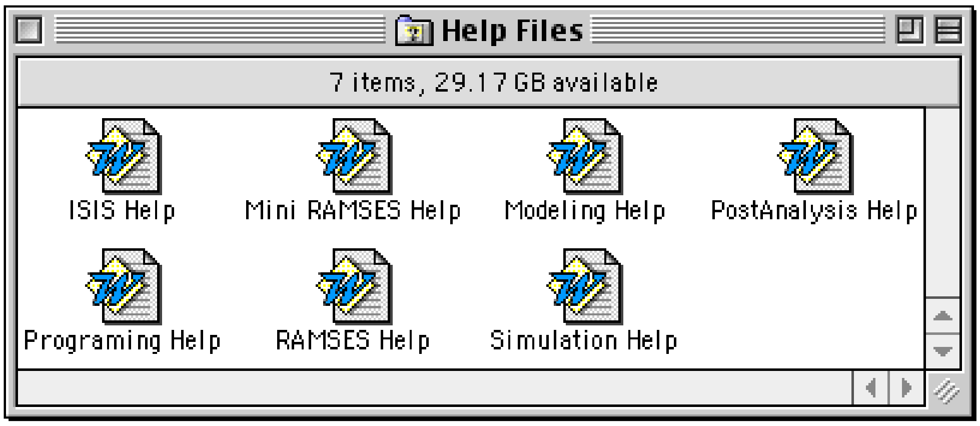
To have a look at the simplest possible 'Dialog Machine' program, consisting only of 10 words, proceed as follows: Select the menu command 'Compile' (press key Tab when the compiler prompts for compilation unit) and select the module 'Simple.MOD' in the folder 'Sample Progs'. Then select menu command 'Execute' and select the module 'Simple.OBM'. Look at what you get, quit 'Simple' and go to the editor (menu command 'Edit', enter your last name and your first name, only needed during the very first start-up of the editor, anything else just accept as suggested) to look at the program, i.e. 'Simple.MOD' you have just executed. Isn't it simple?

Then, you may wish to go back to the RAMSES shell by selecting the menu command 'Clear, save & launch' (Cmd^1) under menu 'Macros' of the editor and have a look at the next, less simpler sample program called 'AlmostSimple.MOD'. It demonstrates how to install a menu, and so on. For your convenience I have added (as enclosure with this mail) two additional sample programs ('LessSimple' and 'TableGraph') in an English version (to be released in the final release only). A bit a less experimental approach is to print the documentations contained in the folder 'Docu' and read them.

2.2.1 USING HELP

Whenever available use the on-line help facility. RAMSES software offers on-line help in form of a menu command starting with *Help...* For instance the RAMSES shell and most RAMSES sessions offer on-line help. To activate it choose the corresponding menu command or if available, press the key "Help". This will open a help window entitled *Help: Topics.* There you may select a topic of interest by clicking on its title as listed in the topics window. You may also navigate through the help facility by using the cursor keys or the page up and page down keys etc. To orient yourself read the window title, which is always of the following form *Help: Current topic*. To print a specific topic push button *Print.*

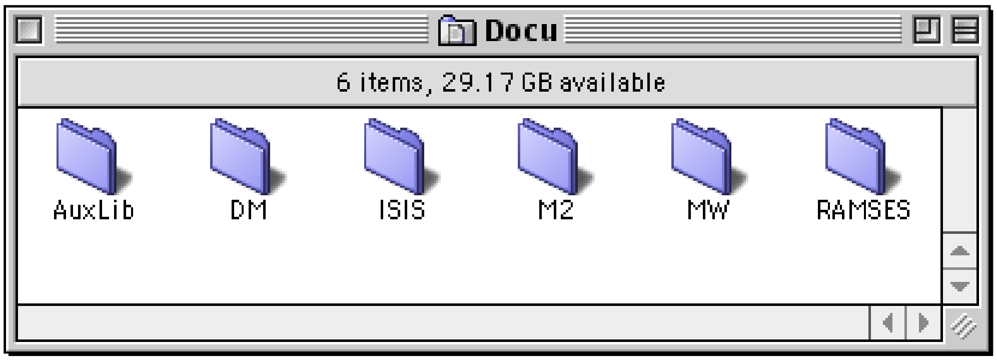
Note that this on-line help facility depends on the presence of so-called help-files. They are distributed within the folder *Help Files* (likely to look similar to Fig. below).



Don't rename this folder or don't drag it to another place relative to the RAMSES or MacMETH shell, unless you know exactly what you are doing.

2.2.2 Use Documentations

The RAMSES software package comes with extensive documentations (where this documentation is just a small part of). All documentations are contained in the folder *Docu* (except for the very first entry point, the "**On RAMSES (READ ME!).pdf"**) and are sorted according to the software layer. It is highly recommended to print them out and to **read** them, in particular the tutorials. Your work will be much more profitable and much more efficient.



This folder contains all documentation for the entire RAMSES software package including all definition modules (*.DEF files). You should not rename it nor the folders contained within, so that the Command-Click or Command^Control^Click onto an identifier of a procedure or a module from within the editor Alpha (given the M2 mode is active) will find the appropriate information. Command-Click finds the procedure definition within the so-called quick references. Command^Control^Click finds a procedure within the exporting definition module. See M2 mode and its help feature for further information on this topic.

2.2.3 Using MacMETH

Detailed instructions on how to use and develop programs using the MacMETH Modula-2 language system can be found in WIRTH *et al*. (1992) (see folder Docu). MacMETH follows largely the definition of the language as published in WIRTH (1985), i.e. the *Report on the Programming Language Modula-2*, 3rd edition.

2.2.4 Using the Dialog Machine

The "Dialog Machine" is a machine-independent software layer between an interactive program and the underlying hard- and software of a particular computer (FISCHLIN *et al*., 1987; FISCHLIN & SCHAUFELBERGER, 1987). It allows to develop so-called "Dialog Machine" programs in a machine independent and highly portable form. For instance all the RAMSES software is solely based on the "Dialog Machine". Large software layers such as ModelWorks (FISCHLIN *et al*., 1994) could be ported successfully several times to other computer platforms such as Macintosh, IBM-PC, and Sun machines (MANSOUR & SCHAUFELBERGER, 1989; THOENY *et al*., 1994).

To develop "Dialog Machine" programs it is recommended to use the session *Programming* from the RAMSES shell. Alternatively it is possible to use the MacMETH shell, but the loading of programs is less efficient than from within the RAMSES shell.

2.2.5 Using ModelWorks and RAMSES

If you wish to work with ModelWorks we highly recommend to consult FISCHLIN *et al*. (1994) (see folder Docu). Especially read the *Tutorial* and follow all herein given instructions step by step to learn about a typical working with ModelWorks and the development of model definition programs. It is recommended to develop models from within the RAMSES session *Modeling*. Basic concepts of the latter session are described in FISCHLIN (1991).

2.2.5.1 Testing the Installation for ModelWorks

Note the RAMSES shell remembers the settings it was in, when it was used the last time. Hence, in case the shell has been used before, confirm that you are really in the proper state to follow the instructions described in this tutorial. The needed settings (modes) are: Use the mode «Mini RAMSES Shell» and the current work object should be *Logistic.MOD* (resides in the folder *Work*, or there is an additional copy also in the folder *Sample Models*). In case the settings are different, use the menu command *Help…* or *Help RAMSES shell…* and follow the herein described instructions on how to change the settings of the RAMSES shell such, that they conform exactly with the settings needed for this tutorial.

(*Customize/Shell modes - Mini RAMSES Shell*)

In case the shell has been used before, confirm that you are really in the Mini RAMSES shell and that the current work object is set to Logistic (resides in the folder Work or in the folder . *Simulation session*. You may do so by checking any of the following indicators: 1) The apple menu contains below the item *About the RAMSES shell…* the following item *About Simulation Session…*. or 2) The command *Simulation* in the menu *Shell* is checked. or 3) The first menu to the right of the menu *Shell* is *Simulation*. or 4) If the window *Shell State* is open, its top line reads: *Currently in session Simulation*. In case this window is not open, you may open it by choosing the menu command *Show shell state…* from the menu *Shell*. Note that the RAMSES shell knows several sessions, but only a single one can be active at a given time. The listed indicators are always available and help you to determine in which RAMSES session you currently are. In case you should now not be in the session *Simulation,* please enter it by choosing the menu command *Shell/Simulation*.

To create a new model use a command contained in the window Shell State. In case the window is not open, choose first the menu command *Shell/Show shell state…*. Push the button *New...* . A dialog box appears, where you can specify the name of the new model definition program. Type *GrassAphids.MOD* and make sure the file is stored in the folder *Work*. Now choose *Modeling/Edit MDP* to start or reenter the text editor. Once you are in the editor you may conveniently open the current work model definition program by choosing the menu command *Macros/Open M2 File[s].*

Now choose *Modeling/Edit MDP* to start or reenter the text editor.

Once you are in the editor you may conveniently open the current work model definition program by choosing the menu command *Macros/Open M2 File[s].*

Once you have finished editing the new model, save it with the command *Save* in the program editor menu, and switch back to the RAMSES Shell by using the command */RAMSESShell* or the shortcut ⌥ $\boxed{\uparrow}$ (option-cursor up) or ⌥ $\boxed{\downarrow}$ (option-cursor down), if you are using MultiFinder. If you are using the Finder choose the command *Transfer/To application...* and open the RAMSES shell.

2.2.5.2 Compilation of the New Model

Use the RAMSES shell for the compilation and execution of ModelWorks models[3]. Select the command *Modeling/Compile MDP…*. The compiler will be started, where you may press return or type the name of your just written source program: *GrassAphids.MOD* (or hit the TAB-key to choose the file with the ordinary Macintosh file opening dialog box). Press return or click the mouse to start the compilation of your work model (or click into the *Open* button).

Should you encounter any problems, e.g. the message *File not found*, check whether you have forgotten to close the work file in the editor or check your installation[4]. If the compiler finds no errors in your program, the compilation will end with the message *+ :Work:GrassAphids.OBM* SIZE; else you will see the message *+ :Work:GrassAphids.RFM errors detected*. In both cases, quit the compiler by pressing the return key or clicking the mouse.

In case compiler errors have been detected, you must first correct them before you can continue. Choose *Modeling/Edit model* to insert the error messages at the violating locations in your model definition source and correct your errors in the model definition program. Errors are marked with the characters '†' which enclose the error number. It is *File[s]* (or type O) then locate the errors by the command *Macros/Find M2 Error* (or type

E). The cursor jumps to the next error mark and an alert with the type of the error found will be diplayed[1]. E.g. if your model definition program is missing the declaration of the parameter c_3, then your file may look similar to this:

```
PROCEDURE Dynamic;
BEGIN
  grassDot  :=  c1*grass - c2*grass*grass - c3† 50†*grass*aphids†117†;
  aphidsDot :=  c3† 50†*c4*grass†117†*aphids - c5*aphids†117†;
END Dynamic;
```

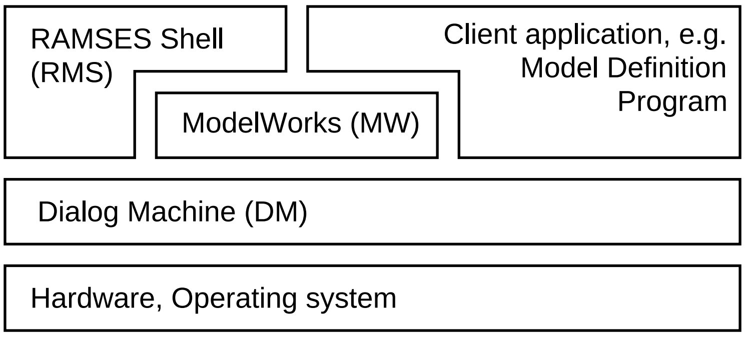
Insert your corrections and close the file with the command *Macros/Close M2 File* (use this command instead of the command *File/Close* to remove the error marks from your source file). Switch back to the RAMSES shell and choose *Modeling/Compile MDP…* . If needed, repeat the edit and the RAMSES shell Run-cycle until the compiler finds no more errors and lets you execute the model definition program.

If the compilation was succesfull, enter the simulation session by choosing the menu command *Shell/Simulation*. The model *GrassAphids* is loaded automatically and made ready to be simulated. Note, the name of the OBM-file is defined by the name of the module, and **not** by the name of the file containing the source program. To avoid confusion it is recommended to use always the same names for files and modules.

2.3 Software Description

2.3.1 RAMSES Architecture

The RAMSES software consists of several software layers built on top of each other.


RAMSES Shell
(RMS)
Client application, e.g. Model Definition Program
ModelWorks (MW)
Dialog Machine (DM)
Hardware, Operating system

Note that the "Dialog Machine" (DM) separates all upper layers from the hardware and system software, hereby enhancing the portability of client applicatins. For instance ModelWorks (MW) so-called model definition programs can usually be ported from one computer platform to the other without any changes (FISCHLIN *et al.*, 1994; THOENY *et al*., 1994, GARDI, 2005). The home page of RAMSES offers further details on this topic:

http://www.sysecol.ethz.ch/RAMSES

The RAMSES software is complex. A internet based reference is available at

http://www.sysecol.ethz.ch/RAMSES/Objects

This reference offers many different views onto the various objects provided by RAMSES. It should help any programer to quickly gain overview, yet find specific and detailed information on all RAMSES objects quickly. Using the M2 mode of AlphaX, you may even accomplish this by a mere Command-key modified double-click.

2.3.2 Availability

The newest version of the RAMSES software package can be obtained via anonymous internet file transfer ftp (File Transfer Protocol) from the host "baikal.ethz.ch" (Internet address 129.132.80.130) in the ftp directory "pub/mac/RAMSES" at no charge. Note however, all copyrights remain with the authors.

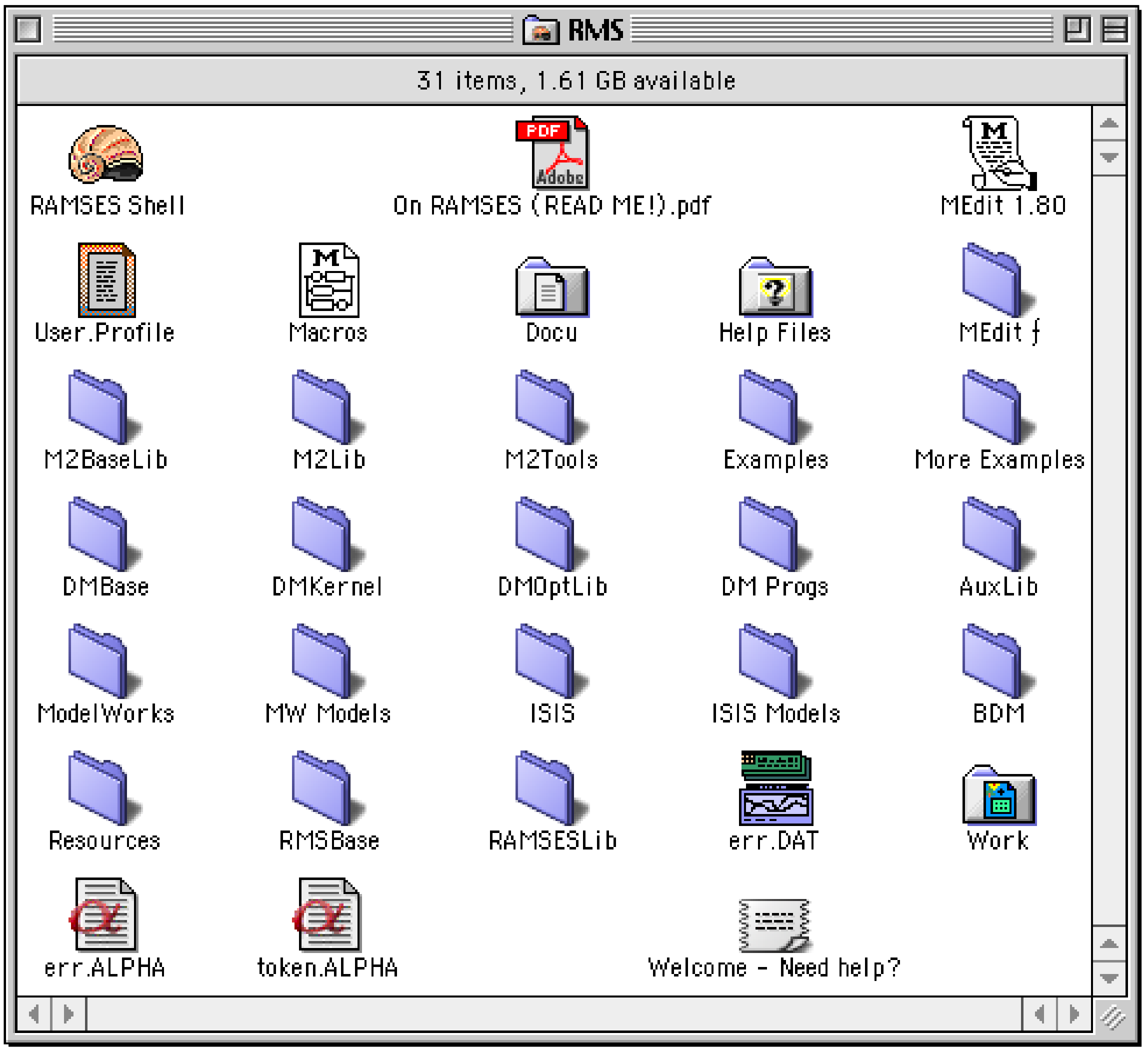
2.3.3 Distribution, and Copyrights

RAMSES is freeware but not public domain. This means two things: First, all rights are reserved and copyrights remain with the authors and Systems Ecology at the Institute of Terrestrial Ecology, Swiss Federal Institute of Technology Zürich, Switzerland. Second, the RAMSES software may not be altered or modified by any means.

2.4 Contents of the RAMSES Release

2.4.1 Applications and Common Files

Once installed you will obtain a folder containing folders and applications similar to this (may look a bit different, depending on the method you receive the software and the release version)[9]:



The "**RAMSES Shell**" is a double-clickable application supporting program development in Modula-2 by means of the 'Dialog Machine'. A main feature is the modeling, simulation and post-simulation analysis of complex respectively ill-defined systems. For starting up and working with the shell, all files holding the software and the documentation (in electronic form) are provided. Consult the read me file "**On RAMSES (READ ME!).pdf"** for a first overview and to get going.

"MEdit" is a program editor supporting the usage of user-written macros for various programming languages. **"Macros**" is the default MEdit macro-file. It contains macros for the interaction with the RAMSES Shell (handling of Modula-2 source files, display of compiler error messages), for an easy insertion of Modula-2 program-structures and several program editing helps.

[9]Under MacOS X it is recommended to run first the little utility **Arrange_RMS_X** to obtain a neatly arranged folder. You find this utility, actually an AppleScript, within the disk image "**Arrange OS X.dmg**". You find this disk image in the separately available release "**RAMSES Extras**" you can download from site http://www.sysecol.ethz.ch/SimSoftware/#RAMSES_Extras.

"User.Profile" is the MacMETH- and RAMSES Shell's configuration file. **"err.DAT"**[10] contains information on which file(s) were compiled during the last compilation or on the name of the current work object of the RAMSES shell.

2.4.2 MACMETH

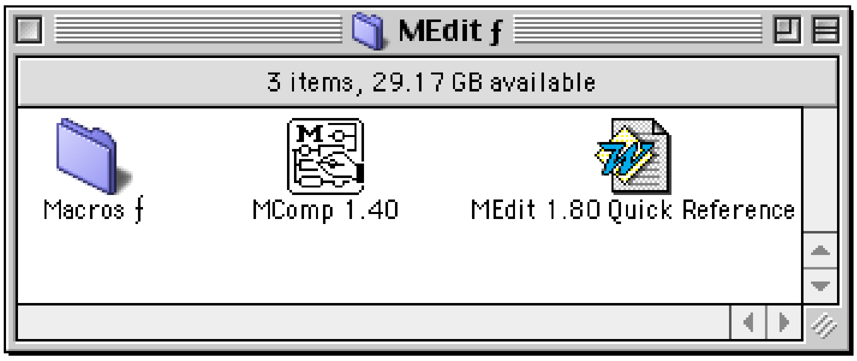
Currently all RAMSES software which runs on the Macintosh computer was built using the Modula-2 language system MacMETH (WIRTH *et al.*, 1992). In its current release (separate folder from the RAMSES folder) it comes with two editors, one is fully integrated into the MacMETH shell but does not follow the user interface guidelines of the Macintosh and the other is also MEdit, which was developed separately by M. Aebi and Systems Ecology. The MacMETH software is almost fully (except for the shell) also contained in the RAMSES release. This software layer consists mainly of the following tools, files and library modules:

2.4.2.1 Program Development Tools

Folder "**M2Tools**" contains Modula-2 tools and subprograms executable from within the MacMETH- or RAMSES-Shell. In the following we will only shortly describe some of the more important and useful program development tools. For further information see the MacMETH User Manual (WIRTH *et al.*, 1992).

"ReadProfile" rereads the configuration file "User.Profile" which also defines access paths to files and modules residing in folders different from the folder in which the file opening Modula-2 program resides.

"Compile", **"Compile20"**, **"Compile20N"**, **"CompileN"** are different versions of the MacMETH single-pass Modula-2 compiler. The suffix "20" denotes generation of instructions directly addressing an FPU for the basic arithmetic operations. "N" stands for "N̲O SANE" resulting in faster but less precise arithmetics.



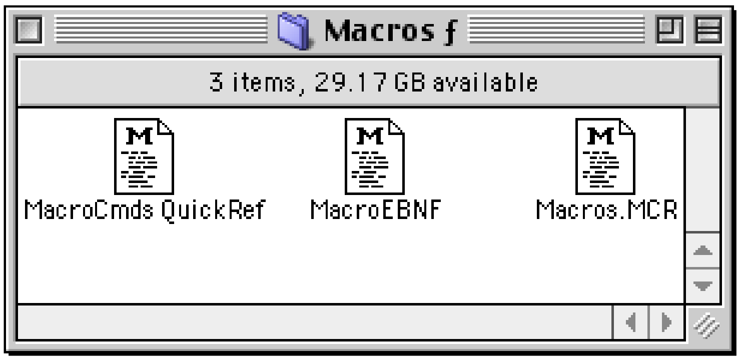
"Edit" is the MacMETH full-screen program editor "Sara". However, it is recommended to use "**Edit2"** instead of "Edit" to write and edit source code. The latter will launch an editor, by default *MEdit*, as defined in the *User.Profile*. The file "**ErrList.DOK**" contains a list of possible compiler error messages for error display by program editors.

A brief documentation of the commands available in MEdit's macro language is contained in "**MacroCmds QuickRef**", whereas the syntax of the Macro-language is given in file "**MacroEBNF".**

"Macros.MCR" contains a collection of macros useful for RAMSES users. The macro compiler **"MComp"** allows to compile the macro file *Macros.MCR* and to convert it to the form in which it can be loaded and used by *MEdit*.

If you don't like *MEdit* it is also possible to use different editors. We have succesfully integrated Alpha, a popular shareware editor written by Pete Keleher (pete@cs.rice.edu).

[10]When working with the «MacMETH Modula-2 Language System For the Apple Macintosh» you will often see two files name "err.DAT" and "err.LST". Each time you call the compiler, it produces these two files. "**err.DAT**" contains information on the file(s) which were compiled during the compilation session and if the compiler detected error(s) "err.DAT" holds also coded error messages, whereas "**err.LST**" contains a listing of the erroneous portions of the previously compiled source codes. Typically you may ignore these files, since their contents are automatically maintained by the compiler and the RAMSES shell. "err.LST" can be totally ignored in case you are using the MEdit or SARA editors (s.a. WIRTH *et al.*, 1992).



"Debug" is a symbolic debugger with multiple windows allowing to view the process and data state of an erroneous program and zoom into its data stuctures.

"Unload" removes any currently unused modules kept in memory and recovers heap space.

"Decode" disassembles an object code file and generates a textfile with mnemonics for the machine instructions.

"Link" is a utility program collecting the codes of separate modules into one file. It can also be used to easily create Macintosh stand-allone applications. **"Linker Extra"** is a stand-alone linker which allows to relink older applications in a downward-compatible fashion. Applications made with MacMETH versions older than 3.2 may no longer run on newer Macintosh models. Use *Linker Extra* to convert such applications and making them again executable on newer models. This will replace the old version of the moduel *System* which is likely to cause the incompatibility with newer machines; in case all software layers above are compatible, the relinked application might work again.

"HierSRC" is a module dependency analyzer which allows to tabulate all modules from which a starting module directly or indirectly imports. For more details see the documentation *HierSRC.DOC*.

2.4.2.2 MacMETH Library

Definition modules of the MacMETH language system are not provided in source form. Full documentation on MacMETH in general and the provided modules is available in WIRTH *et al.*, 1992 (see folder Docu).

"M2BaseLib" contains the minimum set of modules needed by (1) the MacMETH-Shell itself, (2) by the tools available from within the MacMETH-Shell (see description of folder *M2Tools* below), (3) by the Dialog Machine and its library modules and (4) by the RAMSES-Shell and its subprograms or libraries.

"M2Lib" contains utility modules which should normally not be used when working with the *Dialog Machine* or the *RAMSES-Shell*. *M2Lib* constitutes together with the modules of *M2BaseLib* a superset of the modules listed in WIRTH (1985).

Module "TerminalOut" exists in the two implementations "**TerminalOut.OBM (writes file)**" and **"TerminalOut .OBM (no file outp.)"**. The first implementation is the default. It copies all terminal-output to a file "Terminal.OUT" in the MacMETH- or RAMSES-Shell's start-up directory. The file gets rewritten each time the module "TerminalOut" is executed.

Module "MathLib" is available with or without Standard Apple Numeric Environment (SANE) support. The implementation "**MathLib.OBM (SANE)**" is the default. Usage of SANE implies conversion of single REAL numbers (4 bytes) to 10 bytes for arithmetic operations, resulting in a higher precision to the cost of performance. However, usage of "DMMathLib" (SANE/NO SANE) instead of "MathLib" is recommended.

The provided "**LongMathLib.OBM**" is a SANE-supporting implementation.

"System.OBM" is not included, since the module resides in the MacMETH/RAMSES-Shell's code resource.

2.4.2.3 Examples

The folders "**Examples**" and "**More Examples"** contain all sample programs as described in WIRTH *et al.* (1992).

2.4.3 The Dialog Machine

The "Dialog Machine" is distributed as a library divided into the following parts: First the internal base modules **("DMBase"**), second the kernel modules ("**DMKernel**"), and third the optional modules ("**DMOptLib").**

Sample programs are available in the folder "**Sample Progs**". They demonstrate the fundamental use of the "Dialog Machine", i.e. procedure *RunDialogMachine* and the elementary use of menus and windows. For a further introduction to the use of the "Dialog Machine" see the file "**On the DM**" in the folder *Docu* and KELLER (1989), FISCHLIN *et al*. (1987), and FISCHLIN & SCHAUFELBERGER (1987). See also the internet based reference at

http://www.sysecol.ethz.ch/RAMSES/Objects

2.4.4 MODELWORKS

ModelWorks is distributed as a library contained in the folder "**ModelWorks**", which holds all modules needed to develop and execute model definition programs (FISCHLIN *et al*., 1994).

Sample models demonstrating typical uses of ModelWorks from elementary to advanced levels are available in the folder "**Sample Models**". Explanations of these sample models are given in FISCHLIN *et al*. (1994) (see folder *Docu*). See also the internet based reference at

http://www.sysecol.ethz.ch/RAMSES/Objects

2.4.5 AUXILIARY LIBRARY

The following table contains a list and short description of all modules contained in folder "**AuxLib**". Further documentation is available in the corresponding definition modules. For source codes and definitions see also folder *Docu* and in particular the internet based reference at

http://www.sysecol.ethz.ch/RAMSES/Objects

2.4.6 RAMSES LIBRARY

The folders **"RMSBase**" and "**RAMSESLib**" contain all modules needed to execute the RAMSES shell. See also the internet based reference at

http://www.sysecol.ethz.ch/RAMSES/Objects

2.5 Work Organization

The folder “**Work**” has been prepared to allow for the running of the sample model “**Logistic**” as described in the tutorial of ModelWorks (Fischlin *et al*., 1994). Moreover it is ready to hold any work you wish to entertain.

For larger projects it is recommended to create new folders within this work folder. This method allows for easier project switching or removal and archiving of projects once they are finished. The «RAMSES Mini Shell» or the RAMSES session «Modeling» take care of the needed setup in the User.Profile and support also the working with so-called projects instead of a single-file work object.

2.5.1 Use of the Editor Alpha

The standard editor of the RAMSES Mini Shell is MEdit, which comes with the RAMSES software package as "Rain Forest Ware"[11]. In the big shell it is assumed you have the shareware editor Alpha installed. It is highly recommended to use an Alpha editor. Alpha can be obtained from here

http://www.kelehers.org/alpha/	Alpha
http://alphatcl.sourceforge.net/wiki/pmwiki.php/Software/Alpha8	Alpha8
http://alphatcl.sourceforge.net/wiki/pmwiki.php/Software/AlphaX	AlphaX
http://alphatcl.sourceforge.net/wiki/pmwiki.php/Software/Alphatk	AlphaTk

all based on

http://alphatcl.sourceforge.net/wiki/	AlphaTcl

More recently new versions AlphaX (Mac OS X) or AlphaTK (Windows) are recommended. As of version 7.5 and later all Alpha editors contain the M2 mode preinstalled[12], which supports in a sophisticated manner all working with the MacMETH and RAMSES software. You can choose for the RAMSES Mini Shell or any of the RAMSES sessions the editor of your choice, by performing a few customization steps. For instance to choose an Alpha editor for the programing session of RAMSES, execute the steps describe below.

Understand, you always have to customize a pair of involved applications: RAMSES and Alpha.

1. Customization from within RAMSES

One time initial customization:

- In the RAMSES folder you find the file 'User.Profile', open it.
- Insert the following strings into this file
    - In section "Alias" the string: <'Alpha' is ':*AlphaFolder*:*AlphaName*'>, where *AlphaFolder* is the name you have to give the folder you copy Alpha into and *AlphaName* is the name of the Alpha editor, i.e. the application typically named Alpha7;[13]
    - In section "FullMenu" extend the existing string by: <|AlphaEdit/A>, e.g. after 'Edit2/E;[14]
    - Save the file User.Profile.[15]
    - Read the file User.Profile by the menu command: Shell - Read profile.

[11] Similar to shareware, but donate some money to an organization protecting rain forests (see About of MEdit).

[12] Should this not be the case, you can still use the M2 mode. However, then you have to install it manually. This is easy. Download the RAMSES Extras and double-click the file *READ to AUTOINSTALL M2 Mode* (http://www.ito.umnw.ethz.ch/SysEcol/SimSoftware/SimSoftware.html#RAMSES_Extras). Follow the instructions given there.

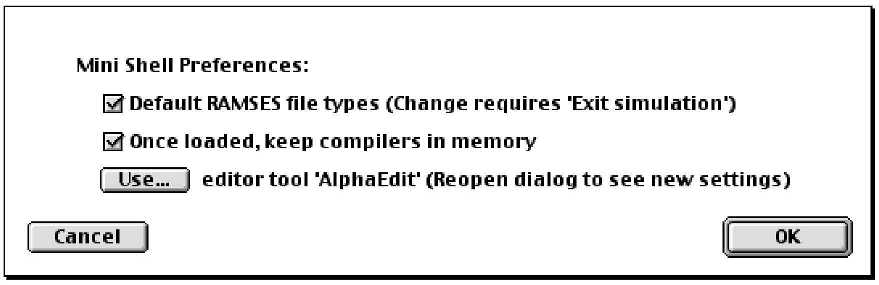
[13] This information will be used by all RAMSES software which are setup to use the Alpha editor (read by tool AlphaEdit)

[14] This information will only be used by the MacMETH shell

[15] From now on the User.Profile can support the use of MEdit or the Alpha editor. You can change the settings any time without having to edit the User.Profile anymore, given the Alpha editor stays at the same location relative to the RAMSES folder

Each component from the RAMSES software package can be configured individually. E.g. each session from within the RAMSES shell and notably the RAMSES Mini Shell can be configured independent from the other. Follow the procedure as described below.

In the RAMSES Mini Shell you have first to activate the power-user mode (consult also the on-line help of the RAMSES Mini Shell), and choose the menu command Shell > Power user > Preferences... There click onto the button "Use..." and select the tool 'AlphaEdit' in the folder M2Tools within the RAMSES folder. Inspect the result of the configuration by choosing the same menu command a second time. It should look similar to this:


Mini Shell Preferences:

☑ Default RAMSES file types (Change requires 'Exit simulation')

☑ Once loaded, keep compilers in memory

Use... editor tool 'AlphaEdit' (Reopen dialog to see new settings)

Below you find the steps needed to configure the session Programing of the big RAMSES shell.

- Make sure the tool file 'AlphaEdit' is within folder 'M2Tools' inside the RAMSES folder[16].
- Start the RAMSES shell and, should you have gotten the RAMSES Mini Shell, enter the big RAMSES shell[17]
- Choose menu command Shell - Programming.
- Choose menu command Programming - Customize - Menu 'Programming'.
- Choose menu command Programming - Edit (alternate 1)...
  - Choose a keyboard equivalent;
  - Click button 'calls (Modula-2 subprogram)' and open the file 'AlphaEdit' in folder "M2Tools" within the RAMSES folder;
  - Check option "Once loaded, keep in memory";[18]
  - Click the OK button;
- Deactivate the Customizing Mode of RAMSES by closing the window 'Customizing Mode'.

Repeat this procedure for any other RAMSES session for which you wish to use the Alpha editor.

2. Customization from with the editor Alpha

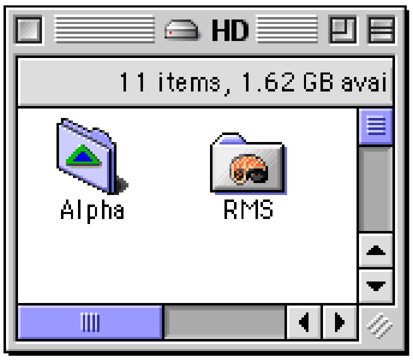
Alpha needs to know where the RAMSES software is installed. Normally this requires very little customization work from your part, given Alpha is installed at the right place relative to the RAMSES folder and you have a simple setup. Yet, it is recommended to perform the customization steps described below at least once, when you wish to work with Alpha.

- Make sure the Alpha folder resides exactly beside the RAMSES folder, i.e. inside the same folder where the RAMSES folder resides. For instance your setup may look similar to this:

[16] This is normally the case, unless you have changed this

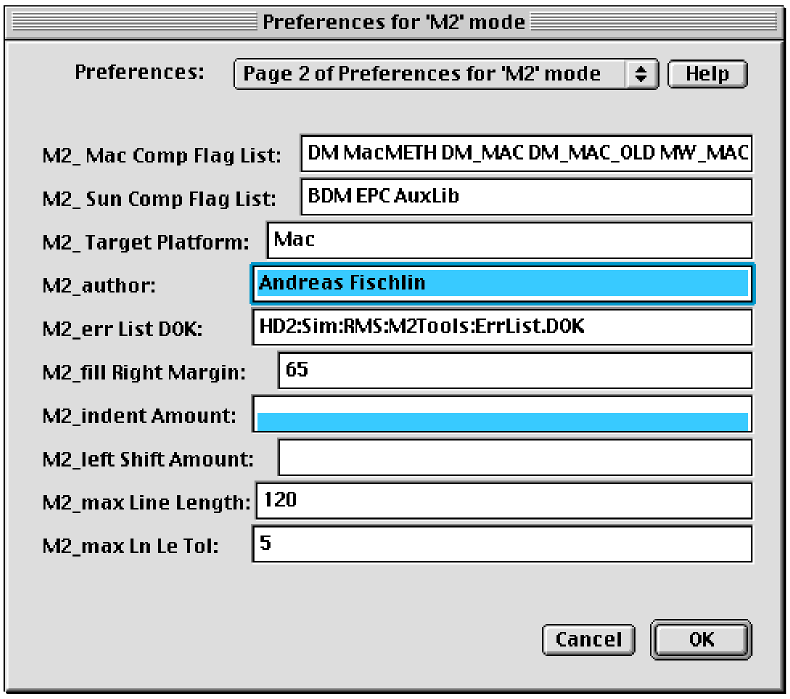
[17] Use the hidden keyboard shortcut Command^Shift^Capslock^B (B stands for big) as described in the Help of the RAMSES Mini Shell and the big RAMSES Shell.

[18] This is recommended for faster operation but not mandatory



- Start Alpha and press Control^0 to activate the M2 mode and open the current RAMSES working object

- You should see the M2 mode menu appear in the menu bar of Alpha. There it is recommended to initialize properly your name by using the function key F12 and enter your name for the item M2_author on the second page of the dialog, similar to this:


Preferences for 'M2' mode

Preferences: Page 2 of Preferences for 'M2' mode

M2_ Mac Comp Flag List:	DM MacMETH DM_MAC DM_MAC_OLD MW_MAC
M2_ Sun Comp Flag List:	BDM EPC AuxLib
M2_ Target Platform:	Mac
M2_author:	Andreas Fischlin
M2_err List DOK:	HD2:Sim:RMS:M2Tools:ErrList.DOK
M2_fill Right Margin:	65
M2_indent Amount:	
M2_left Shift Amount:	
M2_max Line Length:	120
M2_max Ln Le Tol:	5

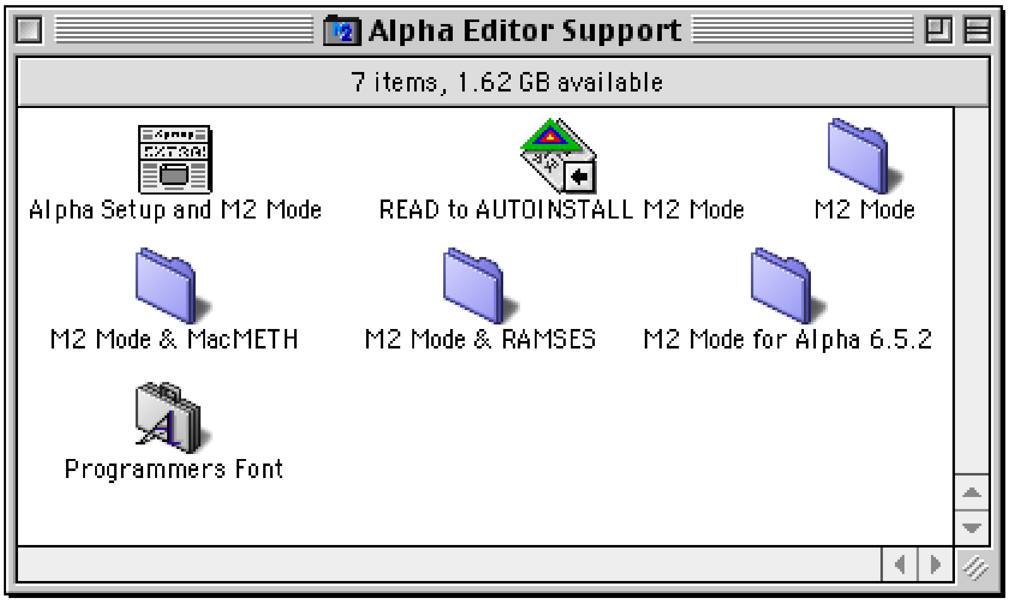
Normally there is no further customization necessary within Alpha. However, should you have several RAMSES folders and/or a MacMETH folders on the same machine, you have to tell Alpha which so-called Modula-2 shell.you wish to use. The automatic setup performed by the M2 mode during its first activation as described above has picked whichever Modula-2 shell it has found first. To gain control over this behavior, choose the menu command Configure Launching from within Alpha, once the M2 mode is active[19]. Then select the shell with which you plan to work. Typically this is either the MacMETH or the RAMSES shell. Both are so-called Modula-2 shells, i.e. applications which can call the compiler and execute your programs. You need to repeat this step each time you wish to use a dfferent Modula-2 shell.

The M2 mode of the Alpha editor is also released independently. It is available as part of the RAMSES Extras software package or from the compact download page at

http://www.sysecol.ethz.ch/SimSoftware/SimSoftware2.html

[19] Should you have trouble activating this mode by the method described here you have probably a non-standard installation. Then you have to setup manually Alpha yourself. Do this by opening from within Alpha a new window and forcing the M2 mode by clicking onto the pop-up menu button Text in the message bar at the bottom of the screen till the bar displays the current mode as M2 similar to this Wrap Mac M2 5 11. Then choose the menu command Configure Launching from the M2 menu and select the Modula-2 shell you wish to use. This should help in most cases to "force" a proper setup even in a non-standard installation. Note, in this case you might also have to adjust the User.Profile used by the Modula-2 shell accordingly, e.g. by setting a path for the Alpha application relative to the RAMSES shell.

The mode is contained in the folder "Alpha Editor Support".



You can reinstall the M2 mode which comes with Alpha anytime by doubleclicking the file " READ to AUTOINSTALL M2 Mode" and following the instructions.

You find also more details on the behavior of Alpha and how to use the M2 mode within Alpha. Most important, note that the M2 mode features a mode specific help which contains all the details needed to make really good use of the mode. It is highly recommended to consult this Help and you will be able to work much more efficiently.

Finally, should you prefer to use MEdit instead of Alpha, simply perform the steps described above within any RAMSES session or the RAMSES Mini Shell. In constrast to what is described above, do not select the tool *AlphaEdit*, instead select the tool *Edit2* from the folder *M2Tools* inside the RAMSES folder. Note, the actual choice of the editor is merely done by selecting a tool, i.e. *AlphaEdit* for Alpha or *Edit2* for MEdit[20].

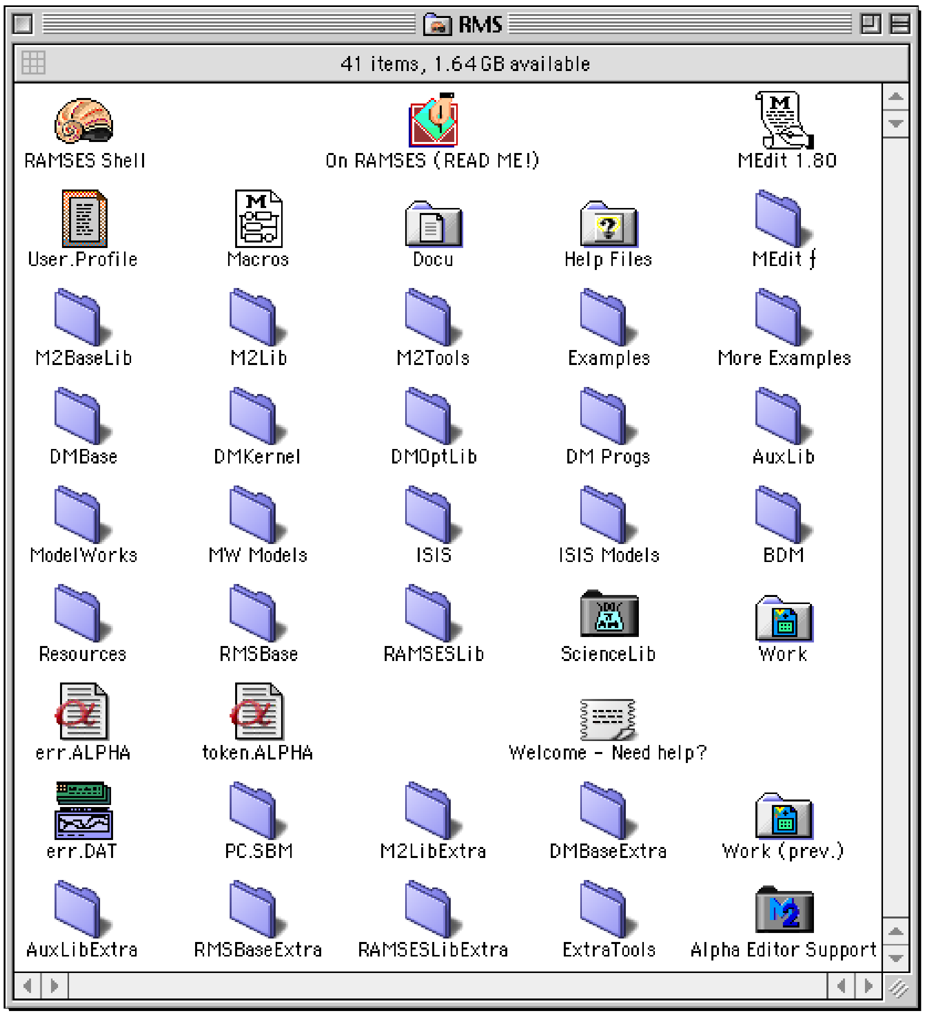
2.6 RAMSES Extras

The RAMSES Extras package extends the standard functionality of RAMSES. It can be downloaded from

http://www.sysecol.ethz.ch/SimSoftware/#RAMSES_Extras

To install it follow the instructions given in the file *READ ME RAMSES Extras.pdf* and run the utility "Arrange_RMS" (under MacOS 9.x) or "Arrange_RMS_X" (under MacOS X). The result should be a folder similar to the following:

[20] You could even select the tool *Edit* to use the Sara editor, However, the latter is no longer recommended, unless you are familiar with its exotic user interface, which originates from the outdated Lilith computer.



All further information can be found in the file *READ ME RAMSES Extras.pdf*.

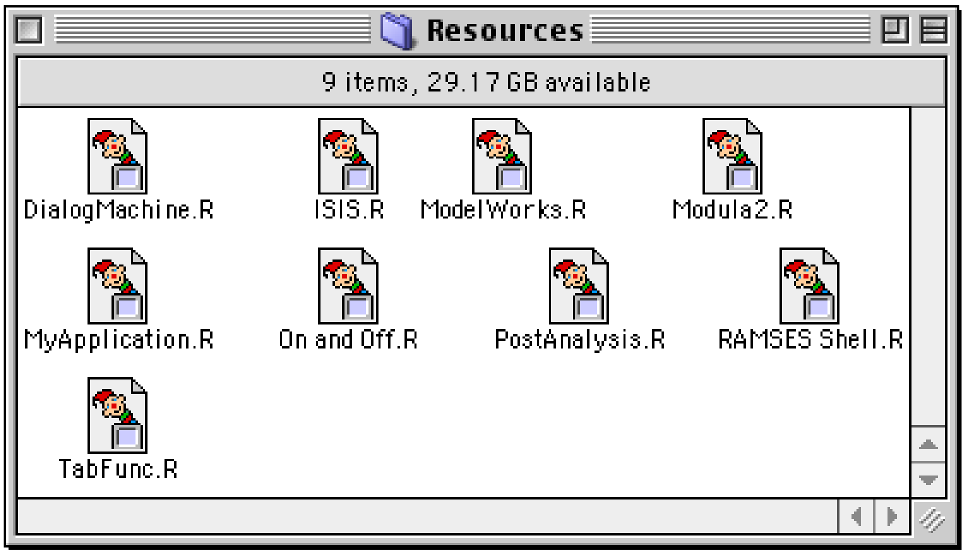
3 Technical Reference

3.1 Linking

3.1.1 Linking Applications

To link any subprogram oder model definition program into a stand-alone application, use the MacMETH utility program «Link» as described in the MacMETH manual WIRTH *et al*. (1992, pp.40-41) with the option application (/A). It is recommended to generate a separate resource file as described in WIRTH *et al*. (1992, p.41) to store all the needed resources by using *ResEdit*. Name this file with the name of the module (typically a program module) you wish to link and add the extension ".R".

The easiest technique is to start from the file "MyApplication.R" provided as part of the RAMSES release in the folder "Resources".



Step 1: **Duplicate** the file "**MyApplication.R**" and **rename** it to "**MyMain.R**" (it is recommended to name this file exactly as the program module name; just add the extension .R. This will make the linking much easier, because you can accept the linker's suggestion by simply hitting the Return key). Then move the file into the same folder where you have the program (or library module) you wish to link as an application. Then use *ResEdit* to modify this file as described below. If you don't know how to use *ResEdit* see ANONYMOUS (1991) and ALLEY & STRANGE (1991).

In case you wish to link only a plain-vanilla MacMETH (M2-level) program, duplicate the file "Modula2.R" (instead of "MyApplication.R"). This file holds only the bare minimum of application specific resources, resulting in a lean, small application.

There is a poor man's method ignoring the provided resource file "MyApplication.R". This is to link the application first once by answering with the key ESC to the linker's prompt and suggestion for a resource file (as described in WIRTH *et al.* (1992), p.41). This results in having a copy of all resources which are also used by the linking application, usually too many (!) in the resource fork of the newly linked application. Then the poor man uses *ResEdit* to open the linked application and copy all resources into a new *ResEdit*-file with the name "MyMain.R". There the poor man needs to remove all unwanted resources inherited from the linking application. Of course the poor man can leave all superfluous resources, e.g. the pictures needed by the RAMSES shell, in there, they won't disturb the functioning of his application much. However, he is likely to waste lots of disk space. Also the maintenance of his application specific resources is more error-prone and the firing up of the resulting application may become slower. Thus it is recommended to follow the procedure described in the following.

In general you ought to understand, there are three mechanisms by which the linker transfers resources into your final application during the linking process with option /A (Linker 3.2.4 or later):

a) First, some essential, mandatory resources are always copied from the linking application into your program (unless option /S would be specified; but the latter is useful only for developpers of RAMSES and are not intended for the general public).

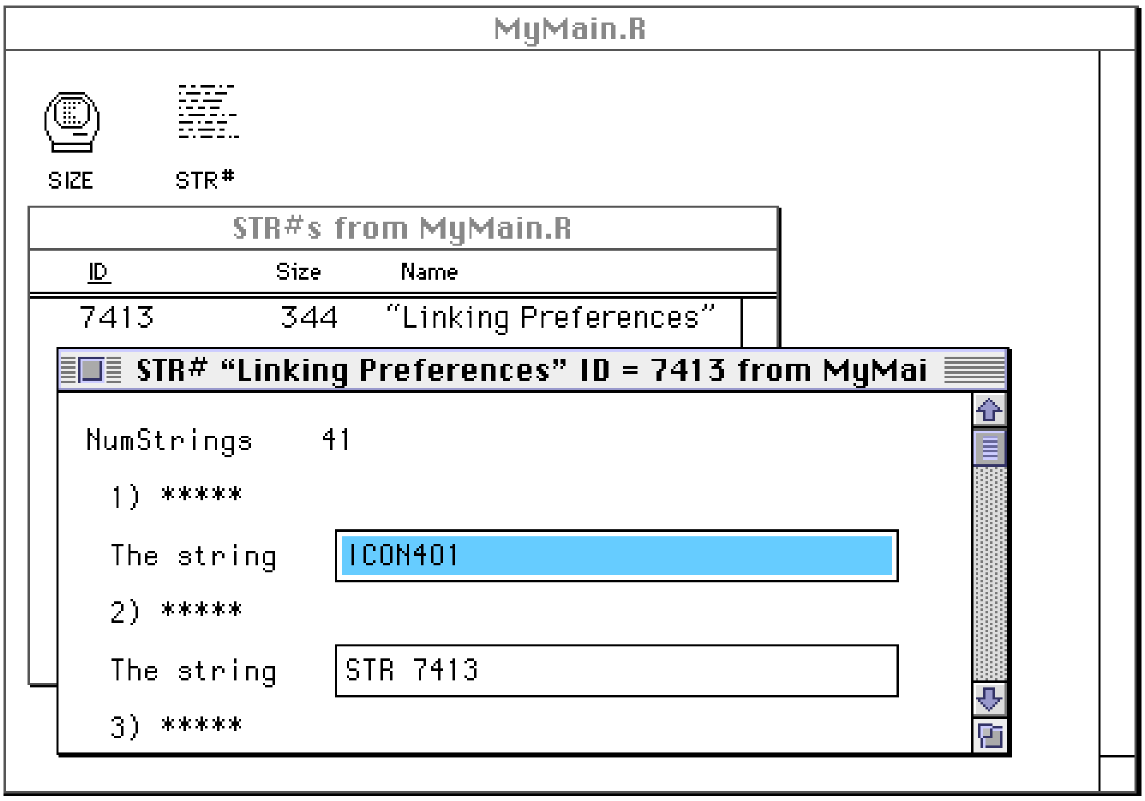
b) Secondly, resources listed within the so-called "**Linking Preferences**" are copied from the linking application into your application "MyMain".

c) Third, all resources contained within "MyMain.R" are copied into your application "MyMain".

Step 2: Secondly start **editing** "**MyMain.R**" using *ResEdit* by editing the so-called "**Linking preferences**". The "Linking preferences" specify which resources are transfered via mechanism b) into your application "MyMain". It lists all resources in form of strings containing the type and the ID of the wanted resource (see figure).

Depending on the software layer from which your application imports, you may not need all resources specified within your resource file "MyMain.R" or you may need more, because your application sits on a higher level than that assumed by "MyApplication.R". "MyApplication.R" assumes you are working on the level of ModelWorks. Use the table in section *Resources* (below) to determine which resources may be deleted (see column SL - software layer) or need to be added from any of the other files contained in the folder "Resources".

You can easily customize the "Linking Preferences" (the resource of type "STR#", with ID = 7413) with *ResEdit*. The "Linking Preferences" provided in "MyApplication.R" as distributed assumes you are working on the level of ModelWorks. Use the table in section *Resources* (below) to determine which resources may be omitted (see column SL - software layer) or need to be added. To omit a resource from ending up in your application "MyMain", simply delete the corresponding string in the "Linking Preferences" (to delete e.g. the third string select "3) ****" and hit backspace). To add a resource select a string and choose menu command "Insert New Field(s)" (Command^K). The string first contains exactly 4 characters specifying the type and then an integer number for the ID of the resource (no space inbetween).



ID	Size	Name
7413	344	"Linking Preferences"

```
NumStrings    41
  1) *****
  The string    ICON401
  2) *****
  The string    STR 7413
  3) *****
```

In case of a conflict, i.e. "MyMain.R" contains a resource of same type and ID as the one listed in the "Linking Preferences", the one in "MyMain.R" will end up in "MyMain" and not the one contained in the linking application. Thus you may simply override a default resource with a customized variant, e.g. to change the position or size of the busy message window used by DMMessages.DisplayBusyMessage, copy first the resource of type "WIND",ID=6789, from the file "DialogMachine.R" into "MyMain.R" and then customize it within "MyMain.R".

Another technique is to copy and paste additional resources from the files provided in folder 'Resources'. E.g. if "MyMain" uses the post analysis, open "PostAnalysis.R", select all, copy them to the clipboard and paste them into "MyMain.R". In this case it is recommended to answer "Yes", in case *ResEdit* asks whether to replace any resources with the same type and ID, since your program is likely to depend on exact type and ID numbers. The latter references would be disrupted if you let *ResEdit* choose unique IDs. However, whenever possible, it is recommended to list additionally needed resources in the "Linking Preferences" only and to refrain from the technique just described. This will avoid unnecessary duplicates and update problems. On the other hand you have to make sure that the linking application contains actually all the needed resources or your linking process will not succeed. If you use the RAMSES shell while linking, this should not occur, since its resource fork represents a superset of all resources needed by MacMETH, the 'Dialog Machine', ModelWorks, any auxiliary modules requiring resources, and finally of-course the RAMSES software layer itself.

To learn about the actual content and meaning of the resources, see below section «Resources».

Example: If your application "MyMain" does only import from the software layer "Dialog Machine" [DM] but from none above this layer, i.e. neither ModelWorks nor RAMSES itself, then you need only the resources listed in the table which are marked with DM or M2. You may therefore safely delete all strings contained in the "Linking preferences" (type "STR#", ID=7413) referencing a resource only needed by the ModelWorks layer. To make the identification of the software layers easy, the "Linking preferences" lists first all resources needed by the layers in the following sequence: Modula-2 (M2) (last is "STR 7413", = M2 preferences), Dialog Machine (DM) (last is "STR 7414", = DM preferences), ModelWorks (MW) (last is "STR 7418", = MW preferences), and TabFunc (auxiliary module) (last is "STR 7500", = TabFunc preferences).

For your convenience the following tables list the resources of interest while editing "MyMain.R" (for details and explanations see tables in section *Resources*):

The minimal or **essential resources** needed at all times (otherwise the application crashes[†]) are the following (will all be added by the linker in all cases):

Resource type	Resource ID
DLOG	301-304, 6002-6003 [†,◊,*]
DITL	301-304, 6002-6003 [†,◊,*]
CODE	0,1 [†,◊]

Note, all essential resources needed by MacMETH are in any case added by the linker from the linking application; in this case the linker will overwrite any copies of these resources eventually present in your separate resource file "MyMain.R" with the resources found in the resource fork of the linking application. The linker informs the user of this fact by the message line *Copying essential resources from linking program 'RAMSESShell'.*

It is highly recommended any Modula-2 (**M2**) program contains also the following resources (in order to warrant a correct behavior such as not skipping the display of messages etc.):

Resource type	Resource ID
ALRT	305 [†,◊]
CODE	0,1 [†,◊]
DLOG	-4000, 6000 [◊]
DITL	-4000, 305, 6000 [21,◊]
mst#	100-103 [€]
SIZE	-1
STR	7413 [◊]
vers	1,2
ev. WIND	ev. 6789 [22,€]

In addition it is recommended to leave in any "Dialog Machine" (**DM**) program also the following resources (in order to warrant a correct behavior):

Resource type	Resource ID
acur	0 [€]

[†] Linker checks their presence and produces error message if not present.

[◊] Automatically inserted by the linker during application linking.

[*] Actually all resources of type 'DLOG' and 'DITL' are copied from the linking application to the resulting application.

[◊] Automatically inserted by the linker during application linking.

[21] Partly checked by the linker

[€] Automatically added if linked from within RAMSES shell unless "Linking preferences" from "MyMain.R" override the default "Linking preferences" contained in the "RAMSES Shell", i.e. omit this resource by not listing it.

[22] Only needed if application imports from *M2BaseLib* library module *M2ErrMarks*.

CURS	4-12 or 4, 256-263 [23],€
ALRT	1003-1005 €
DITL	1003-1005 €
PICT	29800, 29801 €
STR	7414 €

In addition it is recommended to leave in any ModelWorks model definition program (**MW**) also the following resources (in order to avoid crashing and warranting correct behavior):

Resource type	Resource ID
MENU	151-154, ev. 160 [24],€
PICT	800-804 €
STR	7418, ev. 7500 [1],€

All tabulated resources are available in the *ResEdit* files residing within the folder *Resources.*

Step 3[25]: Note, any resource unique to your application "MyMain", e.g. its so-called signature, should be provided as an additional resource in "MyMain.R". It will be transferred into the final "MyMain" via mechanism c) (see above).

In this third step, add the bundle or signature, and the desktop icon for your application. First determine the creator (or signature), a four character string unique to your application; second create a BNDL resource within *ResEdit* and edit all associated icons (it is recommended to include colored and small icons also). As a result you will find the following additional resources in "MyMain.R":

Resource type	Resource ID
BNDL	130
FREF	130
icl4, icl8, and ICN# ics4, ics8, and ics#	130

Step 3 can also be omitted. As a consequence your application will obtain only the default desktop icon and any bundling of documents with your application will not be possible. Thus it is recommended to complete step 3 as described. If necessary, consult *ResEdit* documentation such as ANONYMOUS (1991) and ALLEY & STRANGE (1991).

Step 4: Finally add freely any additional resources such as pictures, sounds, strings etc. which are specifically needed by your application. This step is of course only necessary if your code actually needs additional resources. This would be the case if your code contains statements such as

```
    DisplayPredefinedPicture("", pictureID, rect);
```

or like

```
    PlayPredefinedMusic("", musicID);
```

etc.

The obtained file "MyMain.R" is now ready to be used during any future linking of your application. All needed resources will be added automatically if you enter to the linker's request for a resource file with "MyMain.R".

[23]Choice depends on your preferences. Of course you may add here also your own preferred cursors. Make sure resource acur, ID=0 matches your CURS resources.

[24]Only needed if application imports from auxiliary library module *TabFunc*.

[25]Step 3 can also be omitted. As a consequence your application will obtain only the default desktop icon and any bundling of documents with your application will not be possible. Therefore it is recommended to complete step 3 as described if you link a stand-alone application.

For details on the actual function of the resources used by the RAMSES software, see below section «Resources».

3.1.2 Behavior of a Linked Application

Any MacMETH stand-alone (double-clickable) application adopts the following default strategy to determine the configuration:

Section[26]	Entry	Predefined default
"PATH"	-	none[27]
"Default...Vol"	-	1
"Traps"	'All'	on
	'Arithmetic'	on
	'FPU'	on if FPU present
	'F-Line'	on
	'System'	on
	'Break'	on
"SANE"	'alwaysSANE'	on if CPU >= MC68020[28]
	'invalidHalt'	on
	'underflowHalt'	off
	'overflowHalt'	on
	'divByZeroHalt'	on
	'inexactHalt'	off
"Stack"	'Size'	10kB[29]
"Menu"	-	Only used by MacMETH shells
"FullMenu"	-	Only used by MacMETH shells
"System"	-	Only used by the MacMETH shell. The RAMSES shell maintains its own, user customizable preferences system (accessible via menu commands).
"Printer"	-	Only used by the tool *Print*
"Alias"	'Edit2'	" used by System, i.e. all MacMETH based applications such as the MacMETH or RAMSES shell. The tool 'Edit2' (see folder M2Tools) is actually the main user of this alias and uses it to determine which application (relative/absolute path and name) to launch.
	'Alpha'	" used only by the tool 'AlphaEdit' (see folder M2Tools). AlphaEdit uses this alias to determine which application (relative/absolute path and name) to launch.

[26] Section of the configuration file "User.Profile" as described in Wirth *et al.* (1992), chapter «The Configuration File "User.Profile"».

[27] Unless there is a configuration file "User.Profile" with a "PATH" section present, a MacMETH application can't find any files except for those residing in the same folder as the application.

[28] FALSE in the module «System (Extra)», e.g. contained in the application Linker (see below). This allows for easier relinking and supports upward compatibility of older MacMETH applications.

[29] in addition to the stack space which is already allocated by the Macintosh's operating system. The latter differs, depending on the machine: without Color Quickdraw, it is 8 kB, with Color Quickdraw it is 32 kB. The actual stack size is then the total from these defaults and the amount you specify in the resource "M2 preferences" or the User.Profile (see chapter «Resources», 'STR '/7413).

In contrast to «MacMETH 3.2» as described in Wirth *et al*. (1992) «MacMETH 3.2.x» features also a mechanism to override the above tabulated default strategy for the sections "Traps", "SANE", and "Stack" via the resource «M2 preferences» of type 'STR ' and ID 7413 (see chapter «Resources», 'STR '/7413).

In case there resides a configuration file "User.Profile" in the same folder as the linked application, and the start-up process can successfully open and read it, all specifications found in this file override any of the above configurations.

Finally note, depending on the current hardware or system software platform some of the requested configurations may be interpreted differently. This ensures a proper behavior of the MacMETH system under all conditions. For details see WIRTH *et al*. (1992). However, in contrast to the previous MacMETH version 3.2 «MacMETH 3.2.x» installs the Access Fault Trap (#2) always unless the machine is run with virtual memory turned on.

The file "User.Profile" can be reread any time during program execution (simply call System.ReloadPath, or execute utility "ReadProfile") and the current settings will be replaced by the ones found in the "User.Profile". There is one exception to this behavior, see next paragraph.

In contrast to «MacMETH 3.2» as described in WIRTH *et al*. (1992) «MacMETH >=3.2.3» features a mechanism to change the stack size. To accomplish this, we have added the new section "Stack" with the entry 'Size'. Stack sizes must be specified in kBytes, adding the unit is optional (but there must be no blank between the number and the memory unit!). However, note that the stack size can only be set once at the very beginning of an application while launching it. This means that once read, any rereading of the "User.Profile" lets you never change the stack space, i.e. the section "Stack" is simply ignored (module System calls SetApplLimit and then MaxApplZone, thus preventing any further changes of the memory partitioning; see «Inside Macintosh - Memory manager» for futher information). But if you quit and restart your application, e.g. quit the MacMETH or RAMSES shell, the new value found in the "User.Profile" will become effective. Stack sizes can be set to any amount, but you should always leave some heap space; the total amount of memory in which you let your application run (Get Info command from within Finder, "Memory requirements", minimum size) has to encompass the stack AND the heap.

3.1.3 PRELINKING MODULES

Any module, i.e. so-called OBM-file, can be prelinked by using the MacMETH linker (WIRTH *et al.*, 1992, p. 39ff.). Note that any prelinked module can only be found by the dynamic linking-loader if the module holding the prelinked modules is **programmed as the last in the import list** of any importing client. For instance, always make sure you import in your DM-program the prelinked module *DMMaster* always as the last of all kernel modules, i.e. after any import from prelinked modules such as *DMMenus*, *DMSystem* etc. Use the quick references to learn about which modules belong to the kernel and which don't.

3.2 Resources

The Macintosh system software features so-called resources, which are used to specify texts, layout and form of dialogs, pictures etc. independently from the actual program code (s.a. ANONYMOUS, 1991; ALLEY & STRANGE, 1991). This provides a versatile mechanism to modify the properties even of an application already in daily use or to hold pictures used for display during program development etc. (for how to use resources while linking applications, see above, section «Linking Applications»).

3.2.1 FUNCTIONS AND FORMATS OF RAMSES RESOURCES

Note that MacMETH, the Dialog Machine, ModelWorks, and all of the RAMSES software require resources in order to function properly. Especially the resources for which the ID is marked in bold are so-called essential resources and a program, e.g. the MacMETH or RAMSES shell or any program you linked into an application, is likely to crash or lead to other unpredictable behaviour without them. Note, while linking applications, the linker (linker version >= 2.6.2) tests for the presence of all essential resources and will warn and inform you about any missing, essential resource.

Resources are stored in the so-called resource fork of a file (besides the so-called data fork, which resembles conceptually the classical concept of a file). Each resource is recognized via its type (4 characters) and its ID (1 integer). Any number of resources can be stored in a single resource fork, as long as they are unique by type and ID. Throughout this text references to individual resources are therefore made by its type and its ID in the following format: 'TYPE'/ID. The RAMSES software knows only resources with a IDs >= 128. The RAMSES software uses the following resources (The column marked with N indicates which):

Resource type	Resource ID	SL[30]	Resource Name (not always present)	Purpose/Function
acur	0	DM	Ying Yang	Determines cursor sequence displayed during subsequent calls to *DMMaster.Show-WaitSymbol* for the Ying-Yang waiting cursor set
	128	DM	Hour-glass	Same as 'acur'/0 but for the sand hour-glass cursor set. Swap the IDs of the 2 sets to activate the hour-glass set, i.e. the currently actually used set is the one with the ID = 0.
ALRT [31]	**305**	M2	Low memory	Used by the MacMETH System >= 3.2.2. The *GrowMemory* function uses this alert to inform the users, that the heap memory is used up. An essential resource.
	1003	DM	Inform alert (mod/locDescr)	Used by the "Dialog Maschine" to bring an alert, typically an error message, to the attention of the user. In front of the actual message some information about the module as well as a description of the exact location where the error occurred, e.g. the procedure name, is displayed.
	1004	DM	Inform alert (plain)	A plain message window used by the "Dialog Maschine" to display some information for the user.
	1005	DM	Inform alert (head=locDescr)	A message window similar to 'ALRT'/1003, but the error location can be described without giving or knowing the module name.
BNDL	130	M2 RMS		Bundles the currently running application with desktop icons and the so-called application signature. The MacMETH and the RAMSES shell use the same ID to allow for easier resource pasting. For more details on signatures and desktop icons see chapter «*Desktop Icons*»
CODE	**0**	M2		Basic code containing the jump table which is needed to launch an application. An essential resource.
	1	M2		**The essential resource.** Code of module *System*
	3	M2		MacMETH user identification. Required to call a compiler from within an application.
CURS	4	DM		Cursor always shown if the procedure *DMMaster.ShowWaitSymbol* is called for the first time after a call to *DMMaster.HideWait-Symbol* .
	256-263	DM		Set of Ying-Yang cursors shown during a consecutive sequence of calls (i=0,1,2,...) to procedure *DMMaster.ShowWaitSymbol* ID = 256 + i MOD 8 (default cursor set)

[30] Lists the software layer which requires the resource. Note that upper levels imply the dependency on the lower levels according to the following level hierarchy: MacMETH - M2, Dialog Machine - DM, ModelWorks - MW, RAMSES - RMS in ascending order. Exceptions to the implication, e.g. the signature ETHM (=resource type) ID=0, are marked with an asterisk. In this example this means, that a higher level, e.g. a DM-program, does not require this resource and it should actually be removed from the resource fork of a DM-program if it is linked into an application (see also above, the section on "Linking Applications").

[31] Important Notice: Resources of type 'ALRT' describe only the dialog window, the other items needed during such a dialog are hold in the resources of type 'DITL" with the same IDs

	5-12	DM		Set of hour-glass cursors shown during a consecutive sequence of calls (i=0,1,2,...) to procedure *DMMaster.ShowWaitSymbol* ID = 256 + i MOD 8 (only needed if the corresponding 'acur' resource has the ID = 0)
DITL	-4000[32]	M2	Open dialog with prompt	Overrules the system's open file dialog resource to show the prompt informing the user about the current file dialog. Any RAMSES open file dialog supports the prompt.
	301 - 304	M2	Load status (IO-err) Load status Program status (small) Program status (big)	Used by MacMETH to display the loader or program status before or right after an abnormal call to a subprogram. Essential resources.
	305	M2	LowMemory	Used by MacMETH System, see resource 'ALRT' with the same ID. An essential resource.
	400	M2	MacMETH About	MacMETH About
	1003-1005	DM		Used by the "Dialog Maschine", see resources 'ALRT' with same IDs
	6000	M2	Inform/Message	Dialog used by the MacMETH procedure *FileUtil.Message* to display to the user a message.
	6002	M2	Warn/Halt	Halt message used by MacMETH to halt temporarily the program execution. Typically it is called to display a warning or other non-fatal error condition. Execution of the standard Modula-2 HALT statement or a call to procedure *System.Warn* cause a display of this message. The dialog allows the user to choose a program continuation (default), a call to the debugger (subprogram Debug), or to abort the program. An essential resource.
	6003	M2	Abort/Halt	Halt message similar to the 'DITL'/6002. It is used by MacMETH to halt the program execution when a fatal error has been encountered which disallows any program continuation. Encountering hardware traps such as arithmetic overflow or FPU exceptions (see MacMETH Manual, WIRTH *et al.*, 1992) or the execution of a call to procedure *System.Abort* cause a display of this message. Despite the fact that a further program continuation is no longer possible on the current subprogram level, often it may still be possible to call the debugger by pressing button *Debug* or typing key *D*. However, in some cases the memory of the machine or the hardware condition may be corrupted to such an extent, that this is no longer possible, resulting in unpredictable system behavior. An essential resource.

[32] Negative IDs and IDs <= 127 are reserved for resources used by the system.

DLOG [33]	-4000 - 400 resp. 6000 **6002 - 6003**	M2		Used by MacMETH (see above under corresponding 'DITL' resources). 6002-3 are essential resources.
ETHM		M2*		Signature (creator) of MacMETH containing version information
FREF	130-140	M2		Used in conjunction with 'BNDL' to link documents with desktop icons (see chapter «*Desktop Icons*»).
icl4, icl8, and ICN#	130-140	M2		Desktop icons (big). For details see chapter «*Desktop Icons*»
ICON	240, 401	M2		Small (icon sized) pictures used by MacMETH
ics4, ics8, and ics#	130	M2		Analogous as 'icl4', 'icl8', and 'ICN#' but for the small icons
MENU [34]	129	RMS	RAMSES Shell	Shell menu of the RAMSES shell function set *Shell* (not used by the *Mini RAMSES Shell*) (s.a. 'STR#'/7129).
	130	RMS		Reserved for internal use by the customization sub menu of the RAMSES shell.
	131	RMS		reserved for future use (Edit menu)
	132	RMS	RAMSES Programing	Menu of the function set *Programing* of the RAMSES session *Programing* (s.a. 'STR#'/7132).
	133	RMS		Reserved for internal use by the customization sub menu of the RAMSES session *Programming*.
	134	RMS	RAMSES Modeling	Menu of the function set *Modeling* of the RAMSES session *Modeling* (s.a. 'STR#'/7134). Note that the *Modeling* session uses the DY system which does not have a user interface with any menus.
	135	RMS		reserved for future use (customization sub menu)
	136	RMS	Modeling Edit	Standard edit menu used by the function set *Modeling* of the RAMSES session *Modeling* (s.a. 'STR#'/7136).

[33] Similar to resources of type 'ALRT' the resources of the type 'DLOG' are related to resources 'DITL'. Either a pair 'ALRT' and 'DITL' or 'DLOG' and 'DITL' specify a dialog fully. For a description of the dialogs corresponding (same ID) to the 'DLOG' resources see above under the 'DITL' resources.

[34] All resources of type 'MENU' may be customized for texts and keyboard equivalents freely. However, the number of commands (items) must never be changed, nor should the status of a menu or a command be modified. The latter may not cause much harm, except that it may have no effect at all, e.g. the "Dialog Maschine" program overrides the customization immediately or if the the "Dialog Maschine"-program does not maintain the menu status, such a customization may result in an useless application, e.g. commands may never again become active etc.

Note also that some of these menus form part of so-called functions sets. Besides being fully run-time customizable, a function set is some sort of a super menu, including several menus, commands, plus their associated "Dialog Maschine" procedures. In addition does a function set allow for a simple status management. To function properly a function set always requires additional information not contained in a standard Macintosh 'MENU' resource. These data are stored in resources of type 'STR#' with an ID 7000 + corresponding 'MENU' ID of the first menu in the set. The actual commands of the set are hold in the invidual strings of the 'STR#' resource. See also below 'STR#' resources.

	137	RMS	Modeling Elements	Menu of the function set *System and Model Elements Definition* used by the RAMSES session *Modeling* for system element definition (s.a. 'STR#'/7137).
	138	RMS	Modeling Current work model	Submenu of the function set *System and Model Elements Definition* used by the RAMSES session *Modeling* to display and select the currently active model to which system elements belong.
	139	RMS	Modeling Experiments	Menu of the function set *Experiment Definition* used by the RAMSES session *Modeling* to define experiments (data frames and experimental frames) (s.a. 'STR#'/7139).
	140	RMS	Modeling Windows	Menu of the function set *Window management* used by the RAMSES session *Modeling* for the window management (opening or bringing to the front) (s.a. 'STR#'/7140).
	141	RMS	RAMSES Modeling (simple)	Menu of the function set *Modeling* of the RAMSES session *Modeling (simple)*, which does not support any interactive modeling, but only editing and compiling of MDPs (Model Defintion Programs).
	142-148	RMS		reserved for future use
	149	RMS	RAMSES Simulation	Menu of the function set *Simulation* of the RAMSES session *Simulation* (s.a. 'STR#'/7149). Note that in current implementation the *Simulation* session calls ModelWorks which will result in a simultaneous display of this menu together with all the ModelWorks menus.
	150	RMS		Reserved for internal use by the customization sub menu of the RAMSES session *Simulation*.
	151	MW	ModelWorks File	ModelWorks' menu *File* (see FISCHLIN *et al.*, 1992).
	152	MW	ModelWorks Settings	ModelWorks' menu *Settings* (see FISCHLIN *et al.*, 1992)
	153	MW	ModelWorks Windows	ModelWorks' menu *Windows* (see FISCHLIN *et al.*, 1992)
	154	MW	ModelWorks Solve	ModelWorks' menu *Solve* (see FISCHLIN *et al.*, 1992)
	160	IC[35], e.g. MW	ModelWorks TabFuncs	ModelWorks' menu *TabFunc* used by the auxiliary module *TabFunc* (see FISCHLIN *et al.*, 1992)
	165-170	RMS		reserved for future use by RAMSES PostAnalysis
mst#[36]	100	M2	QuitMenus	Holds strings of the potential menu titles which may contain quitting commands. This information is only of use if the mode flag «*High level event aware*» in the resource 'SIZE' is currently off or if the program is run under a system with version < 7. In the latter situation the application can not be terminated during a system shut down, unless a string contained in this resource matches exactly a menu title.

[35] Importing client, i.e. any application, e.g. a DM-program, importing from auxiliary library module *TabFunc*.

[36] Internal format is exactly that of a 'STR#' resource.

	101	M2	QuitCmds	Similar to 'mst#'/100, but it holds strings of the potential menu commands (items) which may represent a quitting command. This information is only of use if the mode flag «*High level event aware*» in the resource 'SIZE' is currently off or if the program is run under a system with version < 7. In the latter situation the application can not be terminated during a system shut down, unless a string contained in this resource matches exactly the command text which can cause the application to quit. During shut-down, the system will emulate the matching menu command selection, exactly as if the user would have done it manually. Under System 7 and later, the library module *Menu* (*M2BaseLib* of MacMETH) uses this information to determine which menu commands ought to be executed (returned by calls to procedure *GetMenuCmd*) if an Apple core event 'quit' is received from the MacOS. The latter is for instance the case during a system shutdown. If this resource is not present or no string matches a menu command of the topmost subprogram level, the MacMETH based application will not quit automatically and the user will have to terminate the application manually.
	102	M2	OpenMenus	Similar to 'mst#'/100 and 102 but used for file opening dialogs. If the user double-clicks a document file from within the Finder, the system will emulate the matching file opening dialog, exactly as if the user would have selected the e.g. *Open…* menu command and would have selected the double-clicked file from within the dialog box manually.
	103	M2	OpenCmds	Similar to 'mst#'/100 and 102 but used for file opening dialogs.
PICT	795	RMS		Palette buttons used by the RAMSES session *Modeling* in the IO-window *AuxVars.*
	796	RMS		Palette buttons used by the RAMSES session *Modeling* in the IO-window *Expressions*.
	800	MW		Palette buttons used by ModelWorks in the IO-window *Models.*
	801	MW		Palette buttons used by ModelWorks in the IO-window *State Variables*.
	802	MW		Palette buttons used by ModelWorks in the IO-window *Model Parameters.*
	803	MW		Palette buttons used by ModelWorks in the IO-window *Monitoring Variables*.
	804	MW		Palette buttons used by ModelWorks in all IO-windows for scrolling.
	810	RMS		Palette buttons used by the RAMSES session *PostAnalysis*.
	811	RMS		Palette buttons used by the RAMSES session *PostAnalysis*.
	812	RMS		Palette buttons used by the RAMSES session *PostAnalysis*.
	3010	MW		ModelWorks logo used in About ModelWorks and About MDP *i* when launching a model as a subprogram from within the RAMSES Simulation Session
	7000	RMS		RAMSES hyeroglyphe (left)

	7001	RMS		RAMSES hyeroglyphe (right)
	7002	RMS		RAMSES big shell icon
	7003	RMS		RAMSES small shell icon
	29800	DM		"Dialog Maschine" copyright notice
	29801	DM		"Dialog Maschine" celtic knot used in default about message.
RAMS	0	RMS *		Signature (creator) of the RAMSES shell containing version information
SIZE	-1	M2		Contains mode flags. Make sure that in all DM, MW, or RAMSES applications the following flags are set: «*Accept suspend events*», «*32 Bit Compatible*». Beyond and inclusive the "Dialog Maschine", i.e. not plain-vanilla MacMETH the following flags should also be set «*Can background*» and «*High level event aware*». All other flags should always be turned off.
snd	1000 1003 11965 11966	IC[37] RMS	Opening Touche Whoops! Seagull topperware	Used the by the Mini RAMSES Shell (about) and the RAMSES Help mechanism.
STR	7412	RMS	Programming session preferences	Current preferences of the RAMSES session *Programming*.
	7413	M2	M2 preferences	Current preferences of the Modula-2 development system MacMETH: Section[38] Entry Index "Traps" 'All' 1 'Arithmetic' 2 'FPU' 3 'F-Line' 4 'System' 5 'Break' 6 reserved (internal use) 7 "SANE" 'alwaysSANE' 8 'invalidHalt' 9 'underflowHalt' 10 'overflowHalt' 11 'divByZeroHalt' 12 'inexactHalt' 13 reserved (internal use) 14 reserved (internal use) 15 "Stack" 'Size' 16..n blank n+1 [39] DefltCreator n+2..n+5 A character at index i = '1' turns the corresponding entry on, any other value off. If the string is too short to contain a character at position i, the corresponding entry is not modified and keeps its predefined default value.

[37] Importing client, i.e. any application, e.g. a DM-program, importing from auxiliary library module *Help.* The RAMSES shell is an example of such a client.

[38] If this resource is not empty, its content overrides the defaults set by the MacMETH module System as described in Wirth *et al.* (1992), chapter 1.4 «The Configuration File "User.Profile"». Note, in case that there is a User.Profile present, its content will completely override the settings defined by resource 'STR '/7413. However, in case of a stand-alone application, without a User.Profile, the resource 'STR '/7413 can be used to set the desired configuration.

[39] Note, this feature is not available via the file *User.Profile*, but only by using this resource. Thus, there is no corresponding section in the *User.Profile*.

The characters at indices 7,14, and 15 are reserved for internal use only.

The characters starting with index 16 specify the amount in kBytes by which the run-time stack size is increased in addition to the predefined one (as predefined by Apple, i.e. 8 kB for machines without and 32 kB for those with ColorQuickDraw).

```
Example:        110111–110110—10kB
        indices 123456789o123456789..
```

Effect (unless overwritten by a User.Profile) All = on, Arithmetic = on, FPU = off, F-Line = on, System = on, Break = on, alwaysSANE = on, invalidHalt = enabled, underflowHalt = disabled, overflowHalt = enabled, divByZeroHalt = enabled, inexactHalt = disabled; total stack size 18 or 42 kB (depending on machine). Note, these settings are the default settings. To simulate default settings of older MacMETH's (<=3.2), use the following settings:

```
Example:        110111–010110—10kB
        indices 123456789o123456789..
```

However, note, this emulation of older MacMETH's behavior uses the memory still differently, since there was no top-of heap memory cushion set aside between stack and heap (now 20 kB, to be used if heap gets scarce) and stack overflows were less detected, since the heap was not immediately maximally used (no call of MaxApplZone). Moreover, the stack-heap border line was set at the bottom of the block where now the memory cushion starts.

RMSMacMETH available in the Extra Release of RAMSES uses the following default settings:

```
Example:        110111–110110—20kB
        indices 123456789o123456789..
```

NOTE: If this resource is missing in the resource file you specify during linking, it will be copied into your application from the linking application (typically the MacMETH or RAMSES shell) together with all other essential resources.

DefltCreator is a 4 character string which starts after the first blank found in the resource. It is used by the library module *FileSystem* (*M2BaseLib* from MacMETH) to determine the default creator for any created files, e.g. by using procedure *Lookup*. Note, any files ending with the reserved extensions '.SBM', '.OBM', and '.RFM' are treated as belonging to the compiling application, such as the MacMETH or RAMSES shell, i.e., in all these exceptional cases DefltCreator is ignored. In case no DefltCreator can be found, *FileSystem* uses 'MEDT', the creator of the freeware editor *MEdit*.

	7414	DM	DM preferences	Preferences of the "Dialog Maschine". The first 4 characters denote the default creator for any files created by means of the library module *DMFiles*, e.g. by calling procedures *Lookup* or *CreateNewFile*. If this resource can't be found, *DMFiles* uses 'MEDT', the creator of the freeware editor *MEdit*. The latter part is used to specify a string of an alternative file, possibly containing the needed DM resources. This part of the resource is reserved for internal use only (development of the DM) and should normally be left blank.
	7415	RMS	Modeling session preferences	Current preferences of the RAMSES session *Modeling*. [40]
	7416	RMS	DY preferences	Current preferences of the interactive modeling system DY used by the RAMSES session *Modeling*.
	7417	RMS	Simulation session preferences	Current preferences of the RAMSES session *Simulation*.
	7418	MW	MW preferences	Current preferences of the interactive simulation system MW (ModelWorks) used by the RAMSES session *Simulation*.
	7419	RMS	Postanalysis session preferences	Current preferences of the RAMSES session *Simulation*.
	7420	RMS	PA preferences	Current preferences of the post-analysis system PA used by the RAMSES session *PostAnalysis*.
	7421	EMW	*Easy ModelWorks* preferences	Current preferences of the *Easy ModelWorks* application
	7422	M2	*LinkTool* preferences	Current preferences of the *LinkTool* application
	7500	IC[41] MW	TabFunc preferences	Current preferences of the table function library TF from the RAMSES auxiliary library *AuxLib*
	7501-7600	-	*AuxLib* preferences	reserved for future use by the RAMSES auxiliary library *AuxLib*
	7501	TF	TabFunc edit window preferences	Contains the preferences used by the TabFunc auxiliary module. It defines the default location and size of the edit window by 4 integer numbers (free format) listed in the string resource.
	7507	DF	DatFraViewer preferences	Contains the preferences used by the user interface (menu configuration, preferences) of the Data Frame Viewer provided by the RAMSES auxiliary library module *DatFraViewer* from the package *DataFrames* (DF).
	7508	SD & ISIS	SimDatAux & SysDatAux Preferences	Contains the preferences used by the user interface (menu configuration, preferences) of the Simulation Data Manager provided by the RAMSES auxiliary library modules *SimDatAux* and module*SysDatAux* from the packages *SimData* and ISIS, respectively.

[40] This and the other following preferences can be modified by the user interactively from within the corresponding session while running the RAMSES shell

[41] Importing client, i.e. any application, e.g. a DM-program, importing from auxiliary library module *TabFunc*.

	7652	RMS	Current State of RAMSES shell	Contains the current state of the RAMSES shell, so that the next time the shell is started it remembers the previous state.
	7653	RMS	Template for RAMSES Work Objects	Contains the information on the currently active template used for creating new work objects from within the RAMSES shell.
	7654	RMS	RAMSES Work Object	Contains the information on the currently active work object of the RAMSES shell.
	7660	ISIS	SysStructAux Preferences	Contains the preferences used by the user interface (menu configuration, preferences) of ISIS[42]
STR#	6500	DM	CPU names	Used by Dialog Machine to hold names of CPUs (index ≈ DMSystem.CPUType)
	6501	DM	FPU names	as above but for FPUs (index ≈ DMSystem.FPUType)
	6502	DM	ROM names	as above but for ROM (index ≈ DMSystem.ROMType)
	6999	RMS	RAMSES File Names	Specifies the names of the files containing resources (reserved for internal RAMSES development only) and the names of the help files used by the RAMSES shells.
	7000	RMS	RAMSES file types	Specifies the file types for DY-systems ('Modl'), textual MDPs ('MoTx'), compiled MDPs ('Mobj'), data frames ('XLS '), and RAMSES preferences and workspaces ('RMSp').
	7129	RMS	RAMSES Shell (FS data)	Data of the function set *Shell* of the RAMSES shell (s.a. 'MENU'/129-130).
	7131	RMS		reserved for future use
	7132	RMS	RAMSES Programing (FS data)	Data of the function set *Programming* of the RAMSES session *Programming* (s.a. 'MENU'/132-133).
	7134	RMS	RAMSES Modeling (FS data)	Data of the function set *Modeling* of the RAMSES session *Modeling* (s.a. 'MENU'/134-136).
	7137	RMS	Modeling Elements (FS data)	Data of the function set *System and Model Elements Definition* used by the interactive modeling system DY from within the RAMSES session *Modeling* (s.a. 'MENU'/137-138).
	7139	RMS	Modeling Experiments (FS data)	Data of the function set *Experiment Definition* used by the interactive modeling system DY from within the RAMSES session *Modeling* (s.a. 'MENU'/139).
	7140	RMS	Modeling Windows (FS data)	Data of the function set *Window management* used by the interactive modeling system DY from within the RAMSES session *Modeling* (s.a. 'MENU'/140).
	7141	RMS	RAMSES Modeling (simple) (FS data)	Data of the function set *Modeling* of the RAMSES session *Modeling (simple)* (s.a. 'MENU'/141).
	7149	RMS	RAMSES Simulation (FS data)	Data of the function set *Simulation* of the RAMSES session *Simulation* (s.a. 'MENU'/149-150).
	7165-70	RMS		reserved for future use by RAMSES PostAnalysis

[42] ISIS stands for Integrative Systems Implementation Software. ISIS is particularly suited to model complex systems, such as entire ecosystems, and forms part of RAMSES.

	7410	M2	Default linking prefs	Used by the Linker (>= version 3.2.4) in case an application is linked (option /A). Lists all resources to be copied from the linking application into the target application. This resource is only used if no resource STR#,7413 (see below) can be found in the resource file specified in the linking dialog.
	7413	M2	Linking preferences	Used by the Linker (>= version 3.2.4) in case an application is linked (option /A). Lists all resources to be copied from the linking application into the target application.
vers	1,2	M2		Contains the version information displayed by the Finder *Get Info* command.
WIND	6788	RMS	Shell State	Used by the RAMSES shell to display the current status.
	6789	IC[43] M2	Inform on M2 err insertion	Informs the user about the compiler error mark insertion respectively removing. Used by the MacMETH tool *Edit2* and some RAMSES sessions such as *Programming*.

Should you encounter any new run-time problems with the resulting application, which were never visible while running the unlinked program, check the contents of the resource fork with the resource editor. Depending on the highest software layer from which your application imports, resources have to be present according to the layer which is listed in the third column of above table. Note that a layer like DM implies all resources from the layer below, i.e. the M2-layer, unless the layer is marked with an asterisk (see footnote of column SL in table above).

Any listed resource missing in the application's resource fork will result in an non-functioning or not properly functioning application. However, additional resources, as long as they do not conflict with the listed types and IDs, should cause no harm.

3.2.2 ACCESSING RESOURCES

Note, the RAMSES software can also access resources which are not kept in the resource fork of the running application, e.g. the MacMETH shell. This is of course important for a modular modeling environment. In particular, while you are working on a project which requires access to its own resources you have several options to accomplish this. It is possible to store your resources in several files, or to have them all together in one file called "MyMain.R". It is recommended to follow the latter method, since this makes possible later linking easier. To this end create and edit the needed resource file by means of ResEdit as described above under section «Linking Applications».

There are several methods by which you can access the resources from within your software:

Method 1: Default strategy

Method 2: Explicit file reference

In the first case, your software provides the 'Dialog Machine' with an empty string, each time a file name has to be passed to a routine. E.g. the call

```
        DisplayPredefinedPicture("",myPictID);
```
results in the 'Dialog Machine' searching for a resource of type 'PICT' and ID myPictID in these locations:

1) First in the resource fork of the root module "MyModule.OBM" which you have loaded with the dynamic linking loader. For instance the "Mini RAMSES Shell" automatically inserts any resources it finds listed in a project file (extension. PRJ) into the resource fork of the program module's object file. There it will be accessible via the default strategy as soon as you load this module.

2) If a resource couldn't be found yet, the 'Dialog Machine' searches in other files as well. E.g. the 'Dialog Machine' searches first in the resource fork of the running application.

[43] Importing client, i.e. any application, e.g. a M2-program, importing from library module *M2ErrMarks* (in library *M2BaseLib*).

3) If it still can't locate the resource, it uses the resource of type 'STR ' with ID 7414 (called "DM preferences") in the resource fork of the running application to lookup the name of a file in which the resource might also be found. Of course the latter search fails if no such file is accessible (uses PATH definitions in the User.Profile to search) or if the the resource of type 'STR ' with ID 7414 is empty or missing.

4) Finally the system file is also searched.

Note, if all searches fail, e.g. while searching for a picture, you may see only a blank area instead of a graphic or, worse, even see nothing (e.g. the display of a warning message, which may depend on the accessability of ALRT and DITL resources (e.g. see ALRT/DITL 1003-1005 needed by the 'Dialog Machine' for proper message display under all circumstances) may result in no visible effect at all, not even a blank window. Thus the end-user may not learn anything about the occurrence of an error!).

In case of the second method, your software provides the 'Dialog Machine' with an exact file name. E.g. the call

```
        DisplayPredefinedPicture("MyResourceFile.R",myPictID);
```

results in the 'Dialog Machine' searching for a resource of type 'PICT' and ID myPictID only in the resource fork of the specified file. Again a file with the given name is found, even if it resides in another folder than the the running application (called also the current working directory), provided the 'Dialog Machine' can locate it via the PATH specifications given in the User.Profile sitting in the current working directory.

The first method is faster than the second. It is best used in final applications or in conjunction with the "RAMSES Mini Shell" or the RAMSES Session "Modeling" of the big "RAMSES Shell". Otherwise the second method is recommended during development. The second method allows also to edit the resource while the application which needs the resource is still running, since the 'Dialog Machine' closes the resource file immediately after each access.

3.3 Prelinked RAMSES Modules

Some modules are distributed in a prelinked form, either containing already all needed modules, or only a subset of these. Some other program modules are linked into double-clickable applications. If the latter are linked such that they contain all needed modules, they become a stand-alone application (see e.g. utility *Linker*). In the latter case it may be even possible to omit the User.Profile, e.g. by configuring the application via the resource «M2 preferences» (see chapter «Linking Applications»).

3.3.1 MACMETH

LIBRARY/PROGRAM	Linked Modules	Needed Modules
MacMETH[44]	MacMETH ...*System*	Windows FileSystem FileUtil EventBase Menu CursorMouse
RMSMacMETH[1]	MacMETH ...*System*[45]	Windows FileSystem FileUtil EventBase Menu CursorMouse

[44] Linked as application (file type 'APPL'). Note that any MacMETH application requires module System, which must reside in the resource fork (resources of type 'CODE', ID = 1,2, and 3).

[45] In contrast to the shell «MacMETH 3.2.1» does the shell «RMSMacMETH 3.2.1» contain also the following, additional resources: 'STR '/7413 (M2 preferences), 'STR '/7414 (DM preferences), 'STR '/7418 (MW preferences), 'STR '/7420 (PA preferences), and 'STR#'/6999 (RAMSES File Names). The content of the resources 'mst#', 'SIZE', and 'vers' are modified to account for the particular shell properties and replace the resources of same type and ID from the shell «MacMETH 3.2.1». Note that the shell «RMSMacMETH 3.2.1» is useful just for the development or testing of the RAMSES software itself. Hence, when working with RAMSES software, it is recommended to use the RAMSES shell only. In particular, the SIZE resource allows the «RMSMacMETH 3.2.1» shell to run also in the background; however, this behavior is supported properly only for "Dialog Maschine" programs, but not for simple MacMETH programs. Similarily, under System 7, high level events are only supported by the "Dialog Maschine" but not by simple MacMETH programs.

Compile[46]	Compile M2LM M2CM M2HM M2EM M2RM M2TM M2SM M2DM M2CL M2FP	FileUtil FileSystem Terminal TerminalOut TerminalIn EventBase CursorMouse
Compile20[1]	Compile20 M2LM20 M2CM20 M2HM20 M2EM20 M2RM20 M2TM20 M2SM20 M2DM20 M2CL20 M2FP20	FileUtil FileSystem Terminal TerminalOut TerminalIn EventBase CursorMouse
CompileN[1]	CompileN M2LA M2CA M2HA M2EA M2RA M2TA M2SA M2DA M2CLA M2FPA	FileUtil FileSystem Terminal TerminalOut TerminalIn EventBase CursorMouse
Compile20N[1]	Compile20N M2LA20 M2CA20 M2HA20 M2EA20 M2RA20 M2TA20 M2SA20 M2DA20 M2CLA20 M2FPA20	FileUtil FileSystem Terminal TerminalOut TerminalIn EventBase CursorMouse
Debug	Debug DA2U DA2M DA2C FPCR Heap RIO	FileSystem TerminalIn Menu CursorMouse Windows TextWindows EventBase
Decode	Decode MC68020 DisUtilities COPdata	InOut Terminal FileUtil FileSystem EventBase CursorMouse TerminalOut TerminalIn Conversions

[46] MacMETH compilers come in pairs: First there is the less efficient, but more general compiler *Compile*; it generates code executable on any MC68'000, MC68'010, MC68'020, MC68'030, and MC68'040 machine, regardless whether it has a FPU or not. Secondly there is the more efficient, but less general compiler *Compile20*; iit generates code which is only executable on MC68'020 or MC68'030 machines which have either a MC68'881 or MC68'882 FPU and on MC68'040 machines. Apart from a few exceptions (see WIRTH *et al.*, 1992), these 2 compilers accept Modula-2 source code as defined in the «Report on the Programming Language Modula-2» in WIRTH (1985, 3rd ed.). In addition to these basic versions of the compiler pair exists also the pair of the so-called N versions, *CompileN* and *Compile20N*. They are an implementation of Modula-2 as defined in the «Report on the Programming Language Modula-2» in WIRTH (1988, 4th ed.). All four compilers accept the same source code except for some of the mathematical, predefined standard functions (e.g. Sin, Exp etc.) exported by the pseudo module SYSTEM. The latter functions result in a native FPU code generation by the *Compile20/Compile20N* compilers. For more details see WIRTH *et al.* (1992).

Edit	Edit SaraDisplay SaraText SaraCompErr SaraBase SaraMashDep	Storage Printer FileUtil FileSystem TextWindows Windows Menu CursorMouse TerminalIn EventBase
Edit2	Edit2 M2ErrMarks HLEvent SysEnv	System EventBase FileSystem FileUtil CursorMouse
Link	Link FileMgrL	Terminal FileSystem FileUtil EventBase CursorMouse TerminalOut TerminalIn
Linker Extra[1]	Link Terminal FileSystem FileUtil FileMgrL EventBase CursorMouse TerminalOut TerminalIn ...*System* [47]	(none)[48]
Print	Print	FileUtil FileSystem TerminalIn Printer EventBase CursorMouse
ReadProfile	ReadProfile	System
SetDefVolume	SetDefVolume	FileUtil FileSystem EventBase CursorMouse System
Transfer	Transfer	FileUtil FileSystem EventBase CursorMouse System
Unload	Unload	System
UnmarkErrs	UnmarkErrs M2ErrMarks	CursorMouse FileUtil FileSystem EventBase System

[47] Due to the changed 'STR ' 7413 resource, the modules system follows a different default strategy for the single real (REAL) floating-point arithmetic (see chapter «Linking Applications»). The utility «Linker» is the tool Link linked into a stand-alone application, herewith allowing for an easy relinking of old, previously linked applications. Relink with the application Linker any linked, stand-alone application, regardless whether it has been made by MacMETH only, the "Dialog Maschine", ModelWorks, or RAMSES. According to our experience, once relinked, many applications are likely to become again executable on new hardware and system platforms in a fully upward compatible manner In contrast to the MacMETH shell, is the default numeric the extra linker

[48] In contrast to MacMETH shells already fully prelinked to a stand-alone application.

XREF	XREF3 DMMaster ResIdents FileInfo DMStrings TabHandler DMWindowIO DMWindows DMFiles DMMenus DMLanguage DMSystem DMPathBase DMStorage DMConversions DMAlerts DMHandlers DMMessages DMMasterBase DMDlgBase DMWindowBase DMMenuBase DMSys7Events DMBase DMLevels DMQuickDraw DMMemTypes DMLinkLoader DMDebugHelp DMHeapWatch	Menu FileSystem EventBase

In “M2BaseLib”

Terminal	Termbase	(none)
TerminalOut	TerminalOut HideTerminal	EventBase System

In “M2Lib”

EV24	DeviceMgr	(none)

Dialog is no longer prelinked.

3.3.2 MODELWORKS

LIBRARY/PROGRAM	Linked Modules	Needed Modules

SimMaster	SimMaster	DM2DGraphs
	MWDefaults	DMBase
	MWDocProcs	DMClipboard
	MWEntryForms	DMClock
	MWErrors	DMConversions
	MWFiling	DMDebugHelp
	MWFunctions	DMDlgBase
	MWGraphics	DMDlgErrors
	MWItemLists	DMEntryForms
	MWMenus	DMFiles
	MWMonitoring	DMHandlers
	MWObjects	DMHeapWatch
	MWRunTimeSys	DMLanguage
	MWSimLib	DMLevels
	MWSimLibAux0	DMLinkLoader
	MWTabulation	DMMaster
	MWTypes	DMMasterBase
	MWVars	DMMathLib
	MWWButtons	DMMemTypes
	MWWContHdls	DMMenuBase
	MWWMgmt	DMMenus
	MWWPlacing	DMMessages
	RMSDocUtils	DMPathBase
	TFBase	DMPrinting
		DMPTFiles
		DMQuickDraw
		DMSANEEnv
		DMStorage
		DMStrings
		DMSys7Events
		DMSystem
		DMWindowBase
		DMWindowIO
		DMWindows
		DMWPicIOBase
		DMWPictIO
		DMWTxtIOBase
		...JumpTab
		EventBase
		FileSystem
		Menu
SimGraphUtils	SimGraphUtils	
	SGUBase	

TabFunc	RMSDocUtils	DM2DGraphs
	TabFunc	DMBase
	TFBase	DMClock
	TFDocProcs	DMConversions
	TFEdit	DMDebugHelp
	TFFuncs	DMDlgBase
	TFMenus	DMDlgErrors
	TFTypesAndVars	DMEditFields
		DMEFBase
		DMEntryForms
		DMFiles
		DMHandlers
		DMHeapWatch
		DMLanguage
		DMLevels
		DMLinkLoader
		DMMaster
		DMMasterBase
		DMMathLib
		DMMemTypes
		DMMenuBase
		DMMenus
		DMMessages
		DMPathBase
		DMQuickDraw
		DMStorage
		DMStrings
		DMSys7Events
		DMSystem
		DMWindowBase
		DMWindowIO
		DMWindows
		EventBase
		FileSystem
		MatBase
		MatCopy
		Matrices
		MatShape
		Menu

3.3.3 PostAnalysis

LIBRARY/PROGRAM	**Linked Modules**	**Needed Modules**
PaMaster	PAButtonActions	Lists
	PAHandlers	...JumpTab
	PALists	ByteBlockIO
	PAScanner	SimBase
	PAParser	DMMaster
	PAItemsDisplay	DMEntryForms
	PASimInterface	DMFiles
	PAFunctions	DMWindows
	PAObjects	DMWindowIO
	PAFiles	DMMenus
	PAWindows	DMStrings
	PATypes	DMConversions
		DMSystem
		DMMessages
		DMStorage
		SimMaster
		Buttons
		DMEditFields

3.3.4 RAMSES Shell

LIBRARY/PROGRAM	Linked Modules	Needed Modules

RAMSESShell	DMBase DMClipboard DMClock DMConversions DMDlgBase DMDlgErrors DMEditFields DMEFBase DMEntryForms DMFileNames DMFiles DMHandlers DMHFSBase DMLevels DMLinkLoader DMMaster DMMasterBase DMMemTypes DMMenuBase DMMenus DMMessages DMOpSys DMPathBase DMPrinting DMPTFiles DMQuickDraw DMResBase DMResources DMStorage DMStrings DMSys7Events DMSystem DMTxtResBase DMWindowBase DMWindowIO DMWindows DMWPicIOBase DMWPictIO DMWTxtIOBase RMSDialogs RMSErrMarks RMSErrors RMSFctSets RMSHelp RMSShellBase RTFScanner	DMDebugHelp DMHeapWatch DMLanguage EventBase FileSystem Menu RMSDebugHelp

Modeling		RMSErrMarks RMSShellBase RMSFctSets RMSDialogs RMSErrors DMOpSys DMMessages DMClipboard DMEditFields DMClock DMFiles DMFileNames DMMaster DMStrings DMWindowIO DMWindows DMMenus DMConversions DMLanguage DMSystem DMDlgErrors DMEFBase DMWTxtIOBase DMWPicIOBase DMDlgBase DMWindowBase DMLevels DMMasterBase DMMenuBase DMHFSBase DMSys7Events DMBase DMMemTypes DMQuickDraw DMLinkLoader DMPathBase DMStorage DMResources FileSystem DMTxtResBase DMResBase DMHeapWatch Menu EventBase DMDebugHelp

3.4 Desktop Icons

File input into RAMSES (double-click, user interface (open dialog) or client interface). Double-click on a document does first launch the creator, e.g. the shell, and then attempts to launch a particular session. Open dialog allows to open for editing or for execution of a particular file. Opening a file via the client interface is left fully to the programer's responsibility.

• Launching by double click:

• Launching, e.g. by double click, or opening for editing:

- Type 'TEXT' can always be read, independent of creator, since RAMSES does analyze content before distributing the document to a particular session. Can't be customized, and priorities of distribution to sessions are:
    1) Modeling & Experiment Definition,
    2) PostAnalysis,
    3) Programing

• Launching, e.g. by double click or opening for execution:

3.4.1 Application Icons and Signatures (Creators)

```
    MacMETHCreator      = 'ETHM'; (* 'ICN#' ID 130 *)
    MEditCreator        = 'MEDT';
    shellCreator        = 'RAMS'; (* 'ICN#' ID 130 *)
```

3.4.2 Document Icons and Types

```
    textDocuType        = 'TEXT'; (* 'ICN#' ID 139 *)

    M2SymbolDocuType    = 'MSYM'; (* 'ICN#' ID 131 *)
    M2ObjectDocuType    = 'MOBJ'; (* 'ICN#' ID 132 *)
    M2RefDocuType       = 'MREF'; (* 'ICN#' ID 133 *)

    modelDocuType       = 'Modl'; (* 'ICN#' ID 134 *)
    modelTextDocuType   = 'MoTx'; (* 'ICN#' ID 135 *)
    modelObjectDocuType = 'Mobj'; (* 'ICN#' ID 136 *)
    dataFrameDocuType   = 'XLS '; (* 'ICN#' ID 137 *)
```

MS Excel 2.2a (creator 'XCEL') uses the types 'XLS ' (or 'XLBN'?) for normal documents, 'TEXT' for SYLK files. But a SYLK file with type 'XLS ' can be opened by Excel without any problems, and will be interpreted correctly as a SYLK file.

```
    dataFrameDocuType   = 'XLS '; (* 'ICN#' ID 137 *)
```

MS Word 4.00D uses type 'WDBN' for normal documents. However, it can open a file of this type, although it is a RTF file, but first warns the user in a confusing way about the fact, that the file's content is not of the correct format. Then it continues to interprete the content, finds out that it is RTF and asks the user, whether he/she wants to interprete it.

```
    shellPrefsDocuType  = 'RMSp'; (* 'ICN#' ID 140 *)
```

3.4.3 Behavior Rules

Predefined or currently defined default types, one for each session (after / indicated wether customization is possible, if no comment yes means fully customizable to any type, TEXT included or any creator):

```
                TYPE            CREATOR

            Programing
                'TEXT'/no       'MEDT'/yes
                'MSYM'/no       'RAMS'/yes (only for ETHM)
                'MOBJ'/no       'RAMS'/yes (only for ETHM)
                'MREF'/no       'RAMS'/yes (only for ETHM)

            Modeling & Exp. Def.
                'MoTx'/yes      'RAMS'/yes
                                            (Modula-2 module -
                                            (new MEdit does open 'MoTx'
                                            like 'TEXT' files);
                'TEXT'/no       'RAMS'/yes
                'Modl'/no       'RAMS'/no
                'MSYM'/no       'RAMS'/no
                'Mobj'/no       'RAMS'/no
                'MREF'/no       'RAMS'/no

                'XLS '/yes      'RAMS'/yes
                                            (Data frame -
                                            MS Excel can open it if
                                            it is a SYLK or text file)
            Simulation
                'Mobj'/no       'RAMS'/no
                'MOBJ'/no       'RAMS'/no
                'WDBN'/yes      'RAMS'/yes
                                            (Stash file - MS Word can open it
                                            as RTF or text file)
                'TEXT'/no       'RAMS'/yes
```

```
                                          (Stash file - MS Word can open it
                                          as RTF or text file)

            PostAnalysis
                'WDBN'/yes       'RAMS'/yes  (Stash file - MS Word can open it
                                          as RTF or text file)
                'TEXT'/no        'RAMS'/yes
                                          (Stash file - MS Word can open it
                                          as RTF or text file)
                'Mobj'/no        'RAMS'/no
                'MOBJ'/no        'RAMS'/no
```

4 Latest Changes

Please consult the file **On RAMSES (READ ME!).pdf** section "Latest Changes" to learn about the most uptodate information on this topic. An online html-version of this file is also available on the internet at

http://www.sysecol.ethz.ch/RAMSES/READ_ME.RAMSES.html

or alternatively at

http://www.sysecol.ethz.ch/SimSoftware/#RAMSES

link "READ ME".

5 Trouble Shooting

Please consult the file **On RAMSES (READ ME!).pdf** section on "Known Problems and Solutions" to learn about the most uptodate information on this topic. An online html-version of this file is also available on the internet at

http://www.sysecol.ethz.ch/RAMSES/READ_ME.RAMSES.html

or alternatively at

http://www.sysecol.ethz.ch/SimSoftware/#RAMSES

link "READ ME".

6 Bug Reporting

We welcome bug reports, but please send us bug reports only by using electronic mail. Send messages to the following internet address:

`mailto:ramses@env.ethz.ch`

7 Updating Your Software

Visit

`http://www.sysecol.ethz.ch`

and download whatever you need. It's free, courtesy ETH Zurich (but not public domain, since we (authors and ETH Zurich) retain all copyrights).

More specifically the software is available via internet from

http://www.sysecol.ethz.ch/SimSoftware/

or from the more compact download table at

http://www.sysecol.ethz.ch/SimSoftware/SimSoftware2.html

if you favor overview over explanations.

For latest changes see file **On RAMSES (READ ME!).pdf** section "Latest Changes". An online html-version of this file is available on the internet at

http://www.sysecol.ethz.ch/RAMSES/READ_ME.RAMSES.html

or alternatively at

http://www.sysecol.ethz.ch/SimSoftware/#RAMSES

link "READ ME".

8 Reference of RAMSES Objects

Visit

http://www.sysecol.ethz.ch/RAMSES/Objects

This reference can also be downloaded and installed into the Docu folder within the RAMSES folder for off-line access (set preferences in M2 mode accordingly). It is recommended you use "Arrange_RMS_X" to install it after moving the donwloaded archive into the RAMSES folder.

8 References

ANONYMOUS, 1991. ResEdit Reference: For ResEdit 2.1. Reading, Mass.: Addison-Wesley Publishing, 153pp. "An Apple development document" ISBN 0-201-57091-2.

ALLEY, P., & STRANGE, C., 1991. ResEdit complete. Reading, Mass.: Addison-Wesley, 3rd printing, 546pp. (includes program diskette).

GARDI, O., 2005. Analyse und Evaluation der RAMSES Software bezüglich Neuimplementation auf modernen OS X und UNIX/Linux Arbeitsplatzsystemen. Technische Semesterarbeit Fachgruppe Systemökologie, Institut für terrestrische Ökologie, ETHZ, Zürich, Switzerland, 159 pp.

FISCHLIN, A., MANSOUR, M.A., RIMVALL, M. & SCHAUFELBERGER, W., 1987. *Simulation and computer aided control system design in engineering education*. In: Troch,I., Kopacek,P. & Breitenecker, F. (eds.), Simulation of Control Systems, Pergamon Press, 459pp., Oxford a.o., 51-60pp.

FISCHLIN, A. & SCHAUFELBERGER, W., 1987. *Arbeitsplatzrechner im technisch-naturwissenschaftlichen Hochschulunterricht*. Bulletin SEV/VSE, **78** (Januar): 15-21.

FISCHLIN, A. 1991. Interactive modeling and simulation of environmental systems on workstations. In: Möller, D.P.F. (ed.), Analysis of dynamic systems in medicine, biology, and ecology. Proc. of the 4th Ebernburger Working Conference, April 5-7, 1990, Ebernburg, Bad Münster am Stein-Ebernburg, BRD, Informatik-Fachberichte 275, Springer, Berlin a.o.: 131-145.

FISCHLIN *et al.*, 1994. *ModelWorks - An interactive simulation environment for workstations*. Systems Ecology Group, Internal Report 14, Swiss Federal Institute of Technology Zürich, Switzerland, 323pp.

KELLER, D., 1989. *Introduction to the Dialog Machine*. Interner Bericht Nr. 5 (Nov.), Projekt-Zentrum IDA, Swiss Federal Institute of Technology Zürich (ETHZ), Switzerland, 37pp.

MANSOUR, M. & SCHAUFELBERGER, W., 1989. Software and laboratory experiments using computers in control education. IEEE Control Systems **9**: 19-24.

THOENY, J., FISCHLIN, A., & GYALISTRAS, D., 1994. *RASS: Towards bridging the gap between interactive and off-line simulation*. In: Halin, J., Karplus, W., & Rimane, R. (eds.), *CISS - First Joint Conference of International Simulation Societies Proceedings*, 816pp., August 22-25, 1994, Zurich, Switzerland, The Society for Computer Simulation International, P.O. Box 17900, San Diego, Cal. 92177, USA: 99-103.

WIRTH, N., 1985. *Programming in Modula-2, Third, Corrected Edition*. Springer-Verlag, Berlin a.o., 202pp.

WIRTH, N. 1988: *Programming in Modula-2*. Springer, Berlin a.o., 4th, corrected edition.

WIRTH, N., GUTKNECHT, J., HEIZ, W., SCHäR, H., SEILER, H., VETTERLI, C. & FISCHLIN, A., 1992: *MacMETH. A fast Modula-2 language system for the Apple Macintosh. User Manual.* 4th. completely revised ed., Departement Informatik ETH Zürich, Switzerland, 116pp.