RSC_Viewer

A Tool To Dump, Disassemble and Patch Macintosh Resources.

Version 6.3.1 (02/08/1992)

_____

Written by François Menneteau

_____

— 2 —

Written in *THINK's LightspeedC*
Portions © 1986, THINK Technologies, Incorporated.

RSC_Viewer Documentation written and copyright © 1992 by F. Menneteau

TABLE OF CONTENTS

INTRODUCTION

This application is a tool to dump, disassemble and patch any kind of resources.

It is not a Resource Editor like *ResEdit* because resource manipulations are very limitated (e.g. you cannot create, add, delete or append resources).

In addition, It is not a real debugger like *TMON* or *MacsBug* because you cannot execute your program and see the corresponding results. However one can say *RSC_Viewer* is a "static" debugger.

Indeed *RSC_Viewer* is more oriented toward understanding how applications work. Thus many useful information are available into different **windows** (like the list of traps currently recognized, the low memory global variables addresses, etc), and they can be consulted at any time.

RSC_Viewer offers many other possibilities such as saving or printing dumped or disassembled resources in order to examine them (or use them) later, modifying files or resources attributes, searching for ASCII or hexadecimal strings, searching for Macintosh traps, etc.

THE WINDOWS

In the *RSC_Viewer* application there are nine windows. We give here their "symbolic" names and a brief description of their contents:

- **The RSRC FILE WINDOW.** This is the main window. It contains the current resource in dump or disassembly mode.

- **The SYMBOLS WINDOW.** It contains the list of all Macsbug symbols (see MacsBug manual for more information about symbol formats).

Note: during the loading phase, the application informes you if some sym-bols are available .

- **The GLOBAL VARIABLES WINDOW.** It contains a list of most of the low memory global variables.
    It is possible to update this list, see the **RSCV_Compiler.Doc** file.

- **The TOOLBOX TRAPS WINDOW.** It contains a list of the Toolbox traps currently recognized. This list is taken from Inside Macintosh Volume I through V and from various Technical Notes.

For each trap, its hexadecimal value and its name are displayed. In addition, if the trap is a dispatch one, the symbol "/d" is appended to its name, and if the trap is an element of a dispatch or package trap, the symbol "/value" (where value is the value in register D0, A0 or in the stack) is append to its name (exemple: A260: PBGetFCBInfo /8).

- **The OS TRAPS WINDOW.** It contains a list of the OS traps currently recognized. The syntax used for the Toolbox traps also applies here.

- **The three MC680xx WINDOWS.** They contain information about MC68000 and MC68020 format, MC68020 operations and MC68000 ins-tructions. If you want to update them (for example with some 68020 and 68030 instructions), you must modify the corresponding PICT resources.

- **The AVAILABLE MEMORY WINDOW.** It contains the value of the current free memory under finder, and the remaining room of your partition under multifinder.

THE DISASSEMBLER

A great effort has been made to produce a MC680xx code as clear as possible. The format of each line of code is as follows:
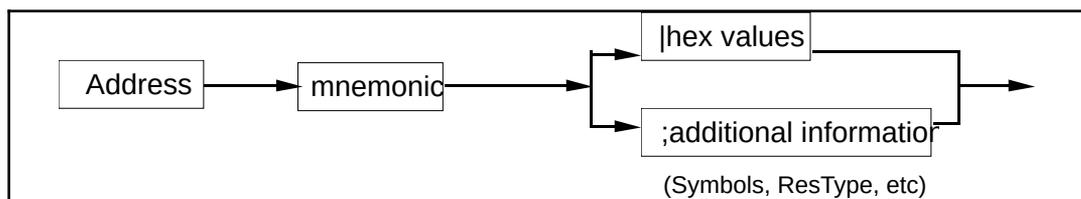
The syntax for a mnemonic symbol is as close as possible to the one used in MacsBug. Howe-ver, there are little changes (for example the address representation), but it is generally for a clarity purpose.

Note: FPU instructions are not disassembled (and I don't think they will be!).

Any kind of resources can be disassembled (see **Filter…** and **File Rsrc List…** in the File menu for more information), and the following functionalities are provided:

• Conversion of a d(PC) instruction into a more suitable address i.e. **Offset[resource number]** (Offset is a long hexadecimal value and resour-ce number is a decimal value).

• Conversion of a trap value into its corresponding name. Trap dispatches and Packages are also converted, but sometimes, it is impossible to give their real name because the program does not succeed in finding the value put into the stack (due to static disassembly).

```
00000830   TST.B      D1                                    |4A01
00000832   BNE.B      *$00000846[1]                         |6612
00000834   _Read                                            |A002
00000836   CMPI.W     #-$0027,D0                             |0C40FFD9
```

• Disassembly at a specific address (remember: address = Offset [id]), by pressing the Command key and selecting it by double-clicking.

```
▤□▓▓▓▓▓▓▓▓▓▓▓▓≡ CODE ID= 1 SIZE= 4590 (11EE). "" ≡▓▓▓▓▓▓
            _%_GETPUT
000007FC   LINK.W      A6,#$0000                              |4E560000
00000800   MOVEA.L     (SP)+,A6                               |2C5F
00000802   CLR.B       D1                                     |4201
00000804   BRA.B       *$0000080A[1]                          |6004
00000806   MOVE.W      #$0001,D1                              |123C0001
```

Some additionnal features are also available when disassembling resources of type CODE. These are:

- Resolution of external references, by scanning the jump table (CODE ID 0) and converting a d(A5) instruction into an Offset [resource number] address.

- Translation of the Mac-OS low memory addresses (global variables) into their corresponding names (if possible). The names are those found in *Inside Macintosh* Volume I through VI and in various other sources (Com-pilers, Technical Notes, etc).

- When Macsbug symbols are available, translation of the Offset [rsrc number] address into the corresponding symbolic name.

```
≡□≡≡≡≡≡≡≡≡≡≡≡ CODE ID= 1 SIZE= 4590 (11EE). "" ≡≡≡≡≡≡≡
000005FA   LEA      $0248(A0),A1                          |43E80248
000005FE   MOVE.L   A1,$0132(A0)                          |21490132
00000602   TST.B    -$0008(A0)                            |4A28FFF8
00000606   BNE.B    *$00000612[1]                         |660A
00000608   PEA.L    -$0010(A0)                            |4868FFF0
0000060C   JSR      *$000007FC[1]                         |; _%_GETPUT
00000610   RTS                                            |4E75
```

- Display of the application entry point, if possible (see **Resource Table**).

- Jump of the four first bytes of any CODE resources (except for the "Jump Table" where 16 bytes are jumped), because these bytes are used by the System and so they have no interest from the disassembler point of view.

- Attempt not to disassemble the symbolic information stored in the code, in order to produce a better listing (it sometimes fails).

- Disassembly at a specific symbol address, by pressing the Command key and double-clicking on the correspondind symbol in the SYMBOLS WINDOW.

THE ASSEMBLER

In **Patch** mode (and not in **Dump** mode) you can now use an assembler. The assembler works by searching for special symbols, located on various places in the main window (see below).
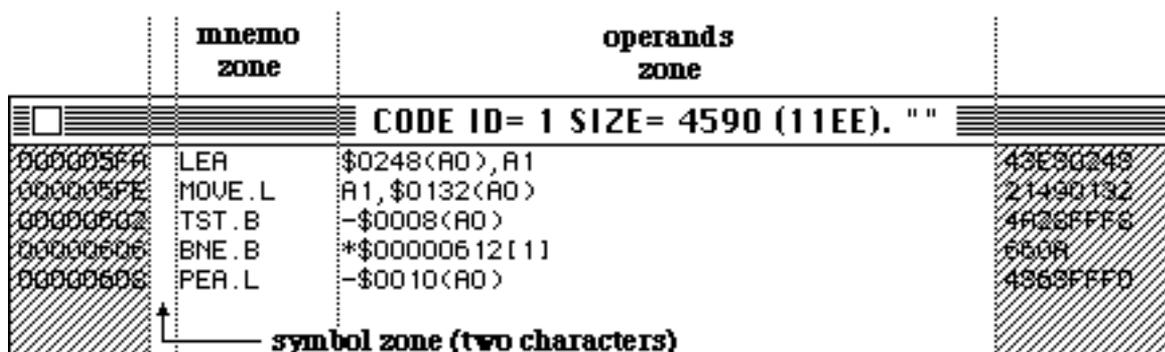
In order to simplify the lines acquisition to be assembled, some useful features are provided:

   • When you are in the **mnemo zone**, and you press the **tab** key, the cursor is moved at the beginning of the **operands zone**, and the characters between the old cursor location and the new one are replaced with white spaces.

   • When you are in the **operands zone**, and you press the **return** key, the cursor is moved at the beginning of the **mnemo zone** of the next line, and the end of the previous line is filled with white spaces.

The **symbol zone** (which contains two characters) is divided into two parts. The first charac-ter describes the action to perform, and the second one characterizes the action:

:	Start assembly at the corresponding address. Don't stop until a next ':' symbol is found. The second character can be:	
	w	write mode i.e. write the result into the resource.
	c	check mode i.e. don't write the result into the resource.
	space	end assembly mode.
=	start assembly — this line only. only **w** or **c** character is allowed after this symbol.	
@	label definition. The second character can only be 0, 1, …, 9. Thus there are only 10 labels available which are: @0 to @9.	

If you mix **:w** and **:c** sequences, the last one will determine the mode (check or write).

It is an error to include a "**=?**" sequence within the ":**?**" "**:** " pairs.

The **mnemo zone** contains the standards MC680xx mnemonics as well as trap names, and the **DC**

instruction. You can enter any trap name that are available in the two traps windows. The assembler is able to generate the full sequence of codes needed when the trap is an element of a dispatch trap.

Example.   **_PrClose** is translated into:

   MOVE.x #$D0000000, -(SP)
   _PrGlue

The **DC** instruction can be used with the **.B**, **.W** or **.L** suffix. When **.B** is used, If the number of characters is odd, the assembler adds a 0 at the end of the sequence, to be word boundary.

Possible **DC** sequences are:

DC.B	'ascii string'	(22 characters maximum. '\0'… \9, \n, \r, \t, \b are recognized).
DC.W	expr1	
DC.L	expr1	
DC.L	'xxxx'	(for Restype)

expr1	=	$value	(hexadecimal value)
	or	-$value	(negative hexadecimal value)
	or	value	(decimal value)
	or	-value	(negative decimal value)

The **operands zone** contains the standard MC680xx operands. **BE CAREFUL**: no white space are authorized between the operands. If you insert even one, the parser will fail.

An operand is a combination of one or more effective address (**ea**), separated by some special symbols (see the instructions description of for more information).

ea	=	Dn	: data register direct (lower case also allowed).
	or	An	: address register direct.
	or	(An)	: address register indirect.
	or	(An)+	: address register indirect with postincrement.
	or	-(An)	: address register indirect with predecrement.
	or	$d_{16}(An)	: address register indirect with displacement.
	or	$d_8(An,Ri.Si)	: address register indirect with index, Si = [W, L].
	or	mi(An)	: memory indirect **[MC68020]**.
	or	*expr2	: absolute address.
	or	$d_{16}(PC)	: program counter indirect with displacement.
	or	$d_8(PC,Ri.Si)	: program counter indirect with index.
	or	mi(PC)	: program counter memory indirect **[MC68020]**.
	or	#expr4	: immediate data.

mi(Xn)	=	$d_{8/16/32}(Xn,Ri.Si*Sc), Sc = [1, 2, 4, 8]	**not implemented**.
	or	($d_{16/32}[Xn],Ri.Si*Sc,$d_{16/32})	**not implemented**.
	or	($d_{16/32}[Xn,Ri.Si*Sc],$d_{16/32})	**not implemented**.

expr2	=	$address	: if < 8 characters, the address is considered as short.
	or	Global variable	: from the GLOBAL VARIABLES WINDOW.
	or	expr3	: converted into a relative PC address.

expr3	=	@label	: the label <u>MUST</u> exist.
	or	$address[ID]	: for PC relative, ID = current resource ID.

expr4 = exp1
or 'xxxx' : ResType.
or _Trap+ one space : only simple trap. Flags (ASYNC, etc) are allowed.

Note: complex expressions with a combination of +, -, * and / signs are not implemented.

The instructions currently recognized are:

Inst.	x			Dy,Dx
B/W/L				
Operands				
Inst.				
B/W/L				
Operands ABCD				-(Ay),-(Ax)
ADDA		x	x	ea,An
ADDI	x	x	x	#expr4,ea
ADDQ	x	x	x	#[1..8],ea
ADD	x	x	x	ea <-> Dn
ADDX	x	x	x	Dy,Dx -(Ay),-(Ax)
ANDI	x	x	x	#expr4,ea
AND	x	x	x	ea <-> Dn
ASd[1]	x	x	x	Dx,Dy #expr4,Dy ea
Bcc[2]	x	S	x	*expr3
BCHG	x		x	Dn,ea #data,ea
BCLR	x		x	Dn,ea #data,ea
BFCHG				ea, {#off or Dn1:#width or Dn2}
BFCLR				ea, {#off or Dn1:#width or Dn2}
BFEXTS				ea, {#o. or Dn1:#w. or Dn2},Dn3
BFEXTU				ea, {#o. or Dn1:#w. or Dn2},Dn3
BFFFO				ea, {#o. or Dn1:#w. or Dn2},Dn3
BFINS				ea, {#off or Dn1:#width or Dn2}
BFSET				ea, {#off or Dn1:#width or Dn2}
BFTST				ea, {#off or Dn1:#width or Dn2}

---

[1] **'d'** = R or L
[2] **'cc'** = [CCCSEQGEGTHILELSLTMINEPLVCVS]

BKPT				#vector
BRA	x	S	x	*expr3
BSR	x	S	x	*expr3
BSET	x		x	Dn,ea
				#data,ea
BTST	x		x	Dn,ea
				#data,ea
CAS2		x	x	Dc1:Dc2,Du1:Du2,Rn1:Rn2
CAS	x			Dc,Du,ea
CALLM				#data,ea
CHK2	x	x	x	ea,Rn
CHK		x	x	ea,Rn
CLR	x	x	x	ea
CMP2	x	x	x	ea,Rn
CMPA		x	x	ea,An
CMPI	x	x	x	#expr4,ea
CMP	x	x	x	ea,Dn
CMPM	x	x	x	(Ay)+,(Ax)+
DBcc[2]		x		Dn,*expr3
DIVx[3]		x	x	ea,Dn
				ea,Dr:Dq
DIVxL[3]			x	ea,Dr:Dq
EORI	x	x	x	#expr3,ea
EOR	x	x	x	Dn,ea
EXG			x	Dx,Dy
				Ax,Ay
				Dx,Ay
EXT		x	x	
EXTB			x	
ILLEGAL				
JMP				*expr3
JSR				*expr3
LEA			x	ea,An
LINK		x	x	An,#expr4
LSd[1]	x	x	x	Dx,Dy
				#expr4,Dy
				ea
MOVE	x	x	x	ea,ea
MOVEA		x	x	ea,An
MOVEM		x	x	<D0-D7/A0-A7> <-> ea
MOVEQ			x	#data,Dn
MOVES	x	x	x	ea <-> Dn
MULx[3]		x	x	ea,Dn
				ea,Dh:Dl
NBCD	x			ea
NEG	x	x	x	ea
NEGX	x	x	x	ea
NOP				

[3]**x** '= S or U

NOT	x	x	x	ea
OR	x	x	x	ea <-> Dn
ORI	x	x	x	#expr4,ea
PACK				-(Ax),-(Ay),#adjust Dx,Dy,#adjust
PEA			x	ea
RESET				
ROd[1]	x	x	x	Dx,Dy #data,Dy ea
ROXd[1]	x	x	x	Dx,Dy #data,Dy ea
RTD				#displacement
RTE				
RTM				
RTR				
RTS				
SBCD	x			Dx,Dy -(Ax),-(Ay)
Scc[2]	x			ea
STOP				#data
SUBA		x	x	ea,An
SUB	x	x	x	ea <-> Dn
SUBI	x	x	x	#expr4,ea
SUBQ	x	x	x	#data,ea
SUBX	x	x	x	Dx,Dy
				-(Ax),-(Ay)
SWAP		x		Dn
TAS	x			ea
TRAP				#vector
TRAPcc[2]		x	x	#data
TRAPV				
TST	x	x	x	ea
UNLK				An
UNPK				-(Ax),-(Ay),#adjust Dx,Dy,#adjust

If the assembly process fails, on each line where an error was detected, a rather obscur mes-sage is displayed. We give their meaning in the following lines:

BAD mnemo	The mnemonic you enter is not a valid one.
BAD operand	One or more components of one or more operands is (are) wrong. Good chance!!!
BAD lab val	Occurs if the label is not between the range 0..9, or is a name.
LAB exists	Once a label is defined, you can't redeclare it, even if you are in an other **::** sym. pairs.
BAD lab ref	You try to reference a label that is not defined yet.
BAD pc rel	Either the resource ID is wrong, or the displacement is out of the resource range.
BAD address	The address you give is not valid.
BAD asm sym	The symbol you enter in the s**ymbol zone** is not valid.
ADD not set	When you try to define a label out of a **: :** symbol pairs.
BAD trap	Obvious.
BAD assoc	Occurs when there is no match between the mnemonic and the operands.
INS ignored	For instructions are thar not implemented (floating points instruction for example).
OPE is out	For an operand out of range (a long instead of a byte for example).

A VERY SHORT USER'S GUIDE

The principle is as follows: after selecting a file, the application displays in the RSRC FILE WINDOW, the first resource available which type is the last specified (the default is CODE), and in dump or disassembly mode (the default is dump). If some symbols are available you can find them in the SYMBOLS WINDOW.

Then you can view the other resources by specifying their IDs (see **Resource Table** item in the ℝsℝc menu for the list of the available resources).

Each resource is shown by block of 512 bytes in dump mode, and by block of roughly 128 bytes otherwise. The vertical scrolling bar allows you to view the hidden part of the current block (the screen is generaly too short to display the entire block), whereas the horizontal scrolling bar allows you to view the other blocks of the current resource.

To patch into a resource, use the **Pach Mode** item in the ℝsℝc menu, and valid your modifications with the **Write Block** item. If you have made a mistake, the **Restore Block** item may be used.

Note: If you have **modified** the current block and you decide to quit the application or chan-ge the current block (there are many way to do this, see menu descriptions), **all the modifications are lost**. However if you have **saved** them, a dialog box appears asking you if you really want the changes to be permanent.

You can also save or print disassembled or dumped resources. This process can be cancelled at any time.

The other capabilities of this application are described menu by menu, in the next pages.

— 18 —

USEFUL FEATURES

- **Easy way to update the list of OS and Toolbox traps**.

  See the *RSCV_Compiler* documentation. However if you update these lists, please send me back the corresponding text file, or at least post it on the net, to allow every one else to update their own files.

- **Easy way to change the 'abort' character when saving or printing rsrc**.

  As there exist different keyboards (for example AZERTY and QWERTY), it is sometimes useful to modify this character. The default one is ';' but if you want to change it, simply edit the '**STR** ' resource (**ID 127**), and replace the first character of the string. **Note**: the string must be at least one-byte long or unpredictable results could arise.

- **Easy way to change the list of resources for symbols scanning.**

  When *RSC_Viewer* opens a file,it first scans all the resources of the current type to search for symbolic information. However resources which do not contain 680xx code never have symbols (MENU resources for example). So in order to avoid scanning all the resources, a list is provided.

  This list contains all the resources where symbolic information can be found (CODE, DRVR, FKEY, etc). If you want to restrict it or expand it, simply edit the '**FSYM**' resource (list of ResTypes).

- **If you launch *RSC_Viewer* by double-clicking on its help file, the help panel is dis- played before any other actions are performed.**

- **Easy way to choose the *RSC_Viewer* preferences.**

  The preferences for *RSC_Viewer* are now stored in a file which is called *RSC_Viewer.Pref*. The algorithm that searchs for this file is as follows (it is based on the one given in the **Usenet Macintosh Programmer's Guide, § Macintosh One-Liners, p. 234**, with a little modification):

  «First look in the current directory for the prefs file. If is found, use that. The Poor Man's Search Path will ensure that any file in the System Folder will be found as well. […]. If no file is found, look for a Preferences folder in the System Folder and look for the file there. If it is not found, create a **Preferences** folder and save a file with the default settings in that folder.»

  If you rename this file and you launch *RSC_Viewer* by double-clicking on it, the

application will take the preferences contained in that file, instead of those contained in the default file. This allow you, for example, to keep many different configurations within the same folder.

- **Easy way to change the name of the default files and folder.**

  "RSC_Viewer.Help" is in resource STR# 131 "FILESNAMEid", item 1.
  "RSC_Viewer.Pref" is in resource STR# 131 "FILESNAMEid", item 2.
  "Preferences" is in resource STR# 131 "FILESNAMEid", item 3 (**note**: under System 7, the application no more use this resource…).

- **Support for Required open and quit Apple events, and drag & drop feature.**

THE PACKAGE

The *RSC_Viewer* package is distributed as **shareware**. If you find it useful, please send $35 to the following address, to become a registered user:

François Menneteau
3 Chemin de L'Eglise
38100 Grenoble
FRANCE.

Please send NO cheques, as banks tend to charge substantial amounts of money for cashing foreign cheques. So, send cash or money order only.

In return, I will send you the latest version of the package, which is made of the following files:

1. **RSC_Viewer**   : the main application;
2. **RSC_Viewer.Help**   : its help file;
3. **ReadMe**   :

   RSCV_COMPILER
   4. **RSCV_Compiler**   : an application to update RSC_Viewer resources;
   5. **GlobalVariables**   : the list of Macintosh global variables;
   6. **ListOfTraps**   : the list of Macintosh traps;

   DOCUMENTATION
   7. **RSC_Viewer.Doc**   : this file;
   8. **RSCV_Compiler.Doc**   : obvious!

The corresponding icons are:

          RSC_Viewer

          File created by RSC_Viewer

          RSC_Viewer.Help

          RSC_Viewer.Pref

          RSCV_Compiler

The *RSC_Viewer* application requires HFS, System 4.1 or later and at least 500 Kbytes of free memory. However, when viewing big applications such as 4D or MS Word for example, you certainly need more memory.

WARRANTY

No warranty, expressed or implied, is provided with *RSC_Viewer*. If you find a bug, let me know and I'll fix it.

Since this application manipulates resources, you **can** easily corrupt them. Thus, when using it, be careful to work on files copy. Otherwise, **I am not responsible for any incidental or consequential losses resulting from its use or misuse**.

BUG REPORTS

RSC_Viewer is at least compatible with Macintosh Plus, SE, SE/30, II, IIsi, IIci and Classic, and has been successfully tested under System 4.3, 6.0.5 and 7.0.

It is multifinder friendly, however, like many resource-editors, some problems may occur (see below).

Any questions or bug reports can be sent to the author at the following network address:

iron@imag.fr

Known possible problems:

> — Under Multifinder, the first time you paste something from *RSC_Viewer* into the Album, the content is sometimes not what you expect. Then paste again and it will work fine. Under 6.0.5 many applications seems to have the same problem.

> — Though it is possible to open a file which is already open (like the System file for example), it is not recommanded to do so, especially under Multifinder. Some crashes are possible even after you quit *RSC_Viewer* .

CREDITS

Countless thanks go out to O. Taramasco, who has patiently waited for his Mac to be available. Every body should be as lucky as I am to have such a great friend.

Many thanks also to Ando Sonenblick who sent me a very interesting list of low memory global variables, to hugues Marty, and to every one on the net who helps me solving my software development problems.

FILE MENU DESCRIPTION

| File	
Open File...	⌘O
Close	⌘W
Get File Rsrc List...	
Get File Info...	⌘I
Get Rsrc List for this File...	⌘Y
Get Info for this File...	
Save Block	⌘B
Save Resource...	⌘S
Save Resource As Text...	
Preferences...	
Background Setting...	
Print...	⌘P
Page Setup...	
Quit	⌘Q

• Open file

Presents a custom Standard File package box (see fugure). The files which are listed have all a Type and a Creator specified by the filter button. The default is **APPL** for the File Type, nothing for the File Creator, and **CODE** for the File Resource.
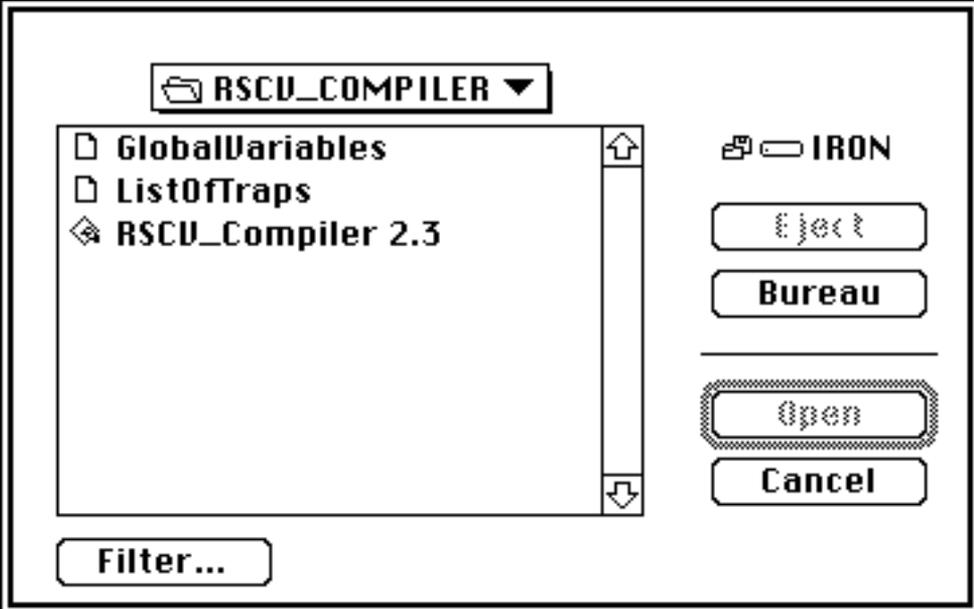
Under System 7, there is also a check box (called **Open Alias**) which allows you to open the alias file instead of the real file.

The **Filter...** button presents a dialog box (see figure), in which you can specify a "File Type" and a "File Creator" for the filter procedure used by the Standard File Package. If you want any kind of "File Type", or "File Creator", put a blank or ???? value. In addition, you can choose the type of the resource you want to study.
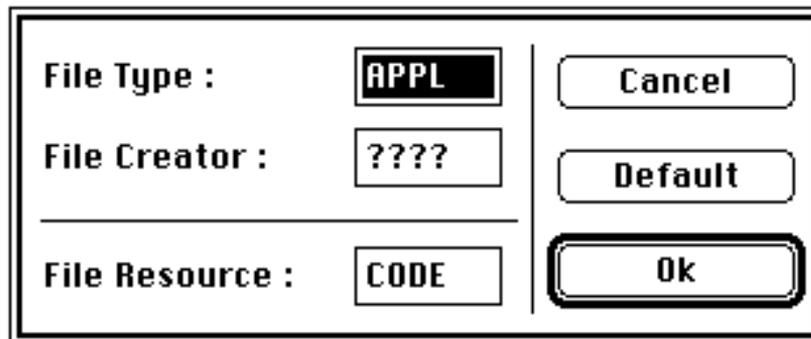

• <u>Close</u>

Closes the front window (which can belong to the application or to a Desk Accessory under Finder).

• <u>Get File Rsrc List</u>

Presents a Standard File Package dialog box (with the same filter procedure as in **<u>Open</u>** item). Once you have chosen a file, a dialog box appears, displaying the list of all the resources available in that file.

You can choose a specific resource by selecting it and double-clicking in the list-window or by writing its name directly in the edit box (don't forget a resource type is <u>four</u> bytes long). The selected resource becomes the current File resource (see **Filter** dialog box).

Then the application opens the file and loads the required resources.

File resources list for :

RSCU_Compiler 2.3

```
RSCC
FREF
BNDL
ICN#
ALRT
DITL
STR#
```

Cancel   **Select**
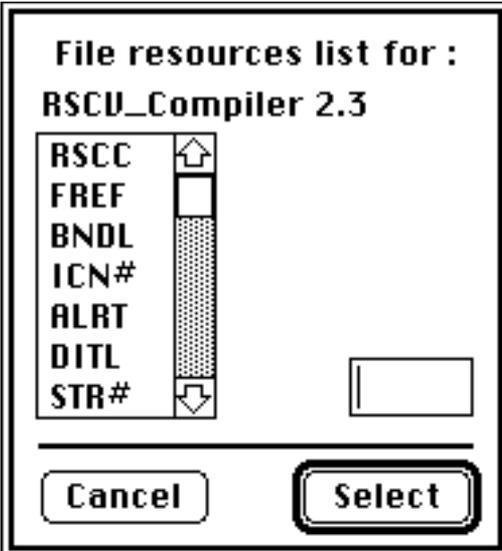
•**Get File Info**

Shows the main attributes (file creator, file type, etc) about the file you have selected. You are allowed to modify them. Depending on the version of the system, the file attributes are not exactly the same (see figures). **Note**: the items that are in gray cannot be modifed, and the **File Protected** attribut is not implemented yet.

Under System 7:

Name : **RSCU_Compiler 2.3**

Type : `APPL`   Creator : `RSCC`

☐ Shared   ☐ No INITs   ☒ Inited
☐ Name Locked   ☒ Has BNDL   ☐ Invisible

☐ Use Custom Icon
☐ Stationary
☐ Alias

☐ File Locked   ☐ File In Use (Busy)
☐ Resources Locked   ☐ File Protected

Created : Samedi 18 Mai 1991, 20:12
Modified : Dimanche 1 Septembre 1991, 13:04

Data Size : 0   Rsrc Size : 48347

[Cancel]   [Save]

• <u>Get Rsrc List for this File</u>

Same as **<u>Get File Rsrc List</u>** item. However it is for the current application.

• <u>Get Info for this File</u>

Same as **<u>Get File Info</u>** item. However it is for the current application, and you **cannot modify** the attributes (the save button is disabled).

• <u>Save Block</u>

Saves the current dumped or disassembled block. The first time you use this item, a standard file package box proposes a file name (**<file name>.RSRC.Samp**). Then, each time you save an other block, it is appended to that file. This item may be very useful if you need only to retrieve some pieces of resource(s).

• <u>Save Resource</u>

Saves the dumped or disassembled resource. The default mode is save current resource. You can change this mode by selecting the **<u>Option</u>** item in the 𝕸𝖎𝖘𝖈 menu. Each time you use this item, a standard file package box asks for a file name, which will contain the dumped or di-sassembled resource(s) specified by the **<u>Option</u>** item.

The name of the save file is the concatenation of the current file and a suffix which characterize the set of resources saved:

    <filename>.<ResType>.<id>   for an unique resource.
    <filename>.<ResType>.List for a list of resources of the given type.
    <file name>.<ResType>.All if all the resources of the given type are saved.

• <u>Save Resource As Text</u>

Saves the resource without formatting it. You can only save resources that do not belong to the '**FSYM**' resource i.e. resources that do not contain instructions. This item is really useful if you want to save the entire content of a '**STR#**' resources for example. It is an extension of the **<u>Copy As Text</u>** item in the menu 𝕰𝖉𝖎𝖙 (see Edit Menu description for more information).

• <u>Preferences</u>

This item allows the user to save the following settings:

- The application's different windows current locations,
- The default File Type and File Creator,
- The default resource type to load and its default mode (dump or disassembly).
- Must the Standard File Package dialog box be open at startup ?

This setting will be take into account the next time you launch the *RSC_Viewer* application.

Depending on what you set or unset the effects are very different:

• If you activate the checkbox of a window (see dialog box below), its current location is saved when you choose **OK**. Otherwise its default location is saved. Of course, when you open again *RSC_Viewer*, each window is displayed (if visible) at its saved location.

• The **Default** button restores the default position for all the windows (very useful when you switch from a Mac II to a Mac SE screen for exemple), the default File Type i.e. **APPL**, the default file Creator (none), the default resource type i.e. **CODE**, the default mode i.e. **dump**, and sets the SFP box default status to open.

```
┌─────────────────────────────────────────────────┐
│              ━━━━━ Windows ━━━━━                   │
│   ┌──────────────────────┬──────────────────────┐ │
│   │ ☐ RSRC FILE          │ ☒ MC680xx FORMAT     │ │
│   │ ☒ MACSBUG SYMBOLS    │ ☒ MC680xx OPERATION  │ │
│   │ ☒ GLOBAL VARIABLES   │ ☒ MC680xx CODEOP     │ │
│   │ ☒ TOOLBOX TRAPS      │ ☒ MEMORY AVAILABLE   │ │
│   │ ☒ OS TRAPS           │ ☐ (not used)         │ │
│   └──────────────────────┴──────────────────────┘ │
│              ━━━━━━ File ━━━━━━                     │
│   ┌─────────────────────────────────────────────┐ │
│   │ Open with File Type : [APPL], File Creator : [????] │ │
│   │ View resource of type : [CODE]  In : ☒ Dump Mode   │ │
│   └─────────────────────────────────────────────┘ │
│              ━━━━━━ Misc ━━━━━━                     │
│   ┌─────────────────────────────────────────────┐ │
│   │ ☐ Open dialog at startup                     │ │
│   └─────────────────────────────────────────────┘ │
│   ( Cancel )      ( Default )      (( Save ))      │
└─────────────────────────────────────────────────┘
```

• <u>Multifinder Setting</u>

As you can open up to nine windows, it is sometimes desirable to close those cumbersome windows that hide your screen, when you switch from *RSC Viewer* to an other application (under Multifinder or Finder for System 7).

However, it is also a good idea to keep at least one window visible, to quickly return to the *RSC Viewer* application.

This item allows you to configure the way your different windows will react to a context switch. It presents a dialog box which look like this:
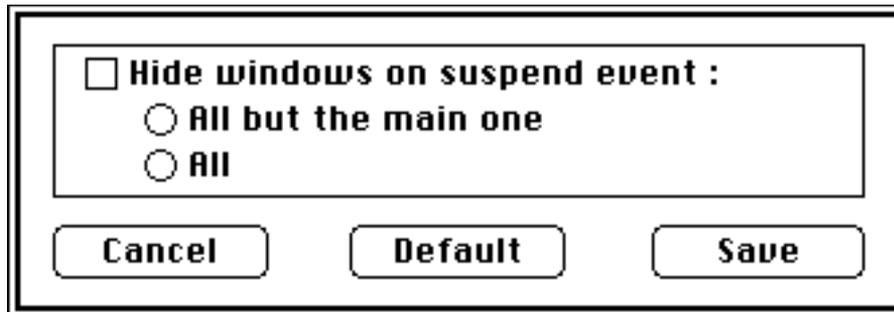
You can choose either to hide **"all the windows on suspend event"** or **"all but the main one"** (the RSRC FILE WINDOW).

If you do not want to hide the windows, click into the corresponding check box. The dialog will then look like this (**Note**: This setting is the default one):
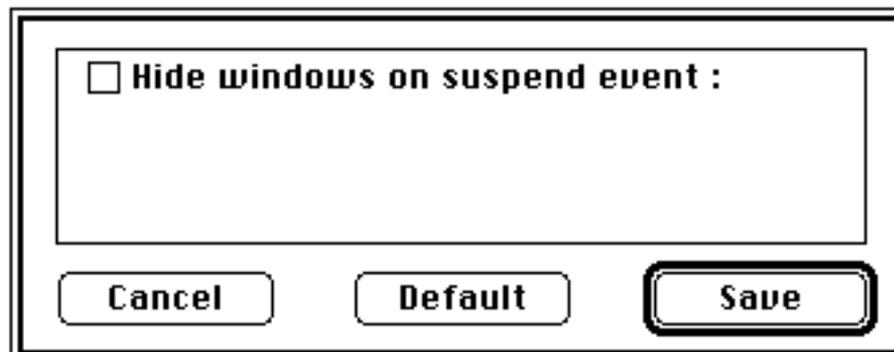
• **Print**

by default, prints the entire current resource. You can change this default mode by using the **Option** item which allows you to specify which resource(s) and within which addresses range you want to print. It is possible to cancel a printing operation (default is the ⌘ - ';' sequence, see the USEFUL FEATURES chapter if you want to change it).

• **Page Setup**

Classic…

• **Quit**

Leaves the application.

EDIT MENU DESCRIPTION

```
Edit
Undo              ⌘Z

Cut               ⌘H
Copy              ⌘C
Paste             ⌘U

Select All        ⌘A

Copy As Text
Switch Zone       ⌘U
```

Only copy operations are possible in *RSC_Viewer* i.e. Cut and Paste operations won't work. In fact, the Paste operation do work in assembly mode (See **Paste** item).

However the ⌘-C/V/X mechanism is fully available in dialogs (thus, for example, you can copy a trap name from the OS TRAPS WINDOW, and paste it in the find dialog box).

• **Undo**

Not implemented yet! (in fact will never be implemented…).

• **Cut**

You are not allowed to cut any pieces of resources.

• **Copy**

Nothing special to say.

• **Paste**

You are not allowed to paste something into any windows but the main one (if you are in assembly mode). It is mostly for safety purpose (patching a resource is sufficiently dangerous to introduce no more problems) and data consistency purpose (for example it is difficult to prevent pasting some ASCII text fragments into the hexadecimal zone of a dumped resource).

In assembly mode, a paste operation acts as a substitute operation (the text in the main win-dow is replaced by the new stuff you paste). however past operations are only allowed in the **symbol**, **mnemo** and **operands zones** (see the assembler for more information).

• **Select All**

Selects the content of the entire window. For the RSRC FILE WINDOW only the content of the current block is selected.

• **Copy As Text**

As in dump mode, you get both hexadecimal and ASCII values, it is sometimes preferable to retrieve only the ASCII part (think about '**STR#**', or '**STR** ' resources for example).

The only restriction is that, whatever you select, the minimum you can copy is 16 bytes (i.e. the ASCII part of a line). Thus, in short, selecting even any one byte of a line, selects in fact the whole ASCII part of that line.

Note: If the current resource type belongs to the list of executable resources (those described in the '**FSYM**' resource), this item is disabled.

• **Switch Zone**

If the cursor is in the hexadecimal part of a dumped line, then it goes to its corresponding position in the ASCII part (and vice versa in the ASCII part). Seems very useful…

RSRC MENU DESCRIPTION

• __Resource Table__

Shows the list of all resources of a given type present in the file. For each resource its ID, its SIZE (in decimal) and its NAME (if available) is displayed. There is also a summary which gives the total number of available resources and their total sizes.

If you want to view a particular resource, you can select it directly by simply double-clicking the corresponding line. In addition you can choose in which mode (dumped or disassembled) you want to see it .

When the resource is of type CODE, this option shows the main entry point of the application.

Unfortunately, some jump tables are not standard, and in that case, it is impossible to give the correct entry point. You can however find it, by disassembling the jump table at address $00000010.

• <u>Show Current Rsrc</u>

Allows the window which contains the current resource (FILE RSRC WINDOW) to become the front window.

• <u>Show Symbol Table</u>

Allows the window which contains the list of symbol names (SYMBOLS WINDOW) to become the front window. If the window is not visible, then it is first displayed.

• <u>Get Attributes</u>

Displays the attributes of the current resource. It is possible to modify all items but the **resChanged** and **resInHeap** attributes (for obvious reasons). If you set the **resProtected** bit on and if you have modified the current block, you won't be authorized to save it. In addition, if you are in **Pach Mode**, this mode becomes disabled.

```
┌─────────────────────────────────────┐
│         resource attributes         │
│  ┌───────────────────────────────┐  │
│  │ ☐ resChanged   ☒ resPurgeable │  │
│  │ ☐ resSysHeap   ☐ resLocked    │  │
│  │ ☐ resPreload   ☒ resProtected │  │
│  └───────────────────────────────┘  │
│   ( Cancel )         ( Save )        │
└─────────────────────────────────────┘
```

• <u>Change Resource</u>

Changes the current resource ID. You can enter the resource ID in hexadecimal form (**HexVal** checkbox active), or in decimal form (**HexVal** checkbox inactive). In addition, you can choose to disassemble the resource (**Dump** checkbox inactive) or to dump it (**Dump** checkbox active). This is the default selected field.

For more convenience, it is also possible to change the default address (0x00000000) where the application jumps for a new resource. This is very useful when the address is really a couple (address, resource ID) — the entry point for example.

By default, the dialog displays the current resource ID, the current mode, and a 0x00000000 address.

In addition, the resource ID value is in decimal and the address is in hexadecimal.

```
                    CODE ID= 0 SIZE= 248 (F8). ""
00000000 : 0000 0108 0000 0E3E 0000 00E8 0000 0020 | .......>.......
00000010 :
00000020 :     ┌──────────────────────────────────────────────┐
00000030 :     │  Resource ID :     [0          ]   □ Hex value │
00000040 :     │                                                │
00000050 :     │                    ⊠ View Rsrc in Dump Mode    │
00000060 :     │                                                │
00000070 :     │  Address :         [ 00000000 ]   ⊠ Hex value  │
00000080 :     │                                                │
00000090 :     │  ──────────────────────────────────────────    │
000000A0 :     │  (  Cancel  )              (( Ok ))             │
000000B0 :     └──────────────────────────────────────────────┘
000000C0 :
000000D0 :
000000E0 : 2604 3F3C 0003 A9F0 25DE 3F3C 0003 A9F0 | &.?<..э.%.?<..э.
000000F0 : 23DA 3F3C 0003 A9F0                     | #.?<..э.
```
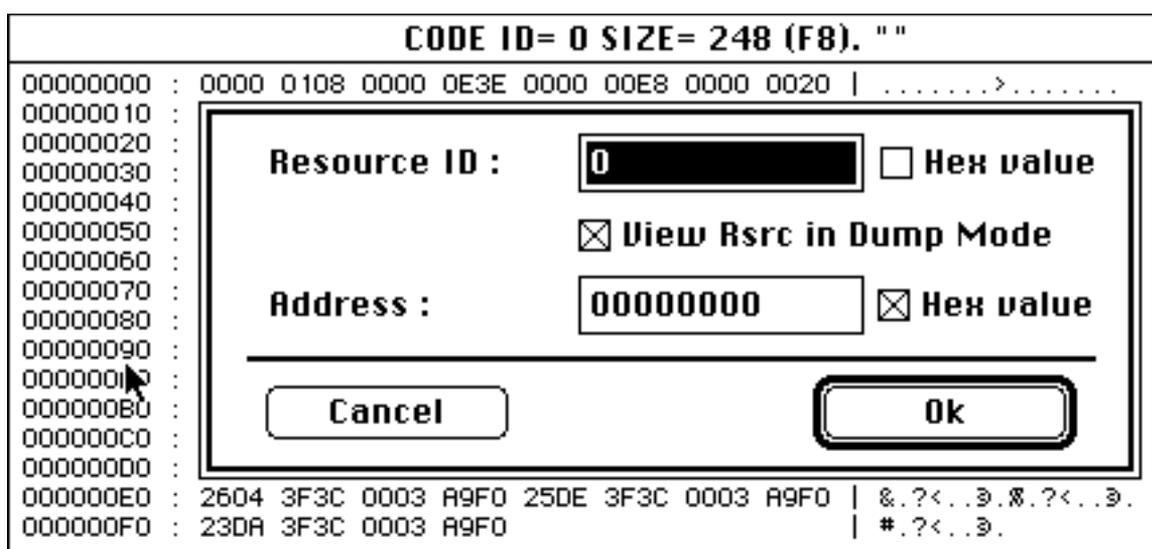
If the current block has been modified and saved, the following dialog box is displayed:

```
┌────────────────────────────────────┐
│ WARNING!                           │
│ The content of the current block has│
│ been changed and saved.            │
│ Are you absolutely sure you want the│
│ modifications to be permanent ?    │
│ ─────────────────────────────────── │
│  (( No ))            (  Ok  )       │
└────────────────────────────────────┘
```
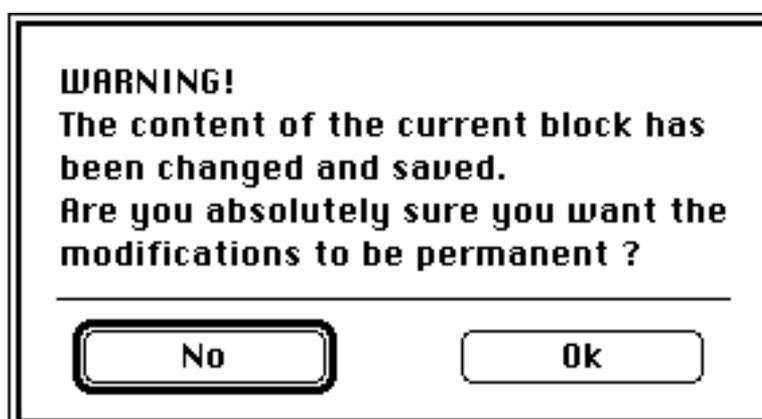
If you choose OK, you change the current resource (so the modifications are permanent, and the **Restore Block** item is disabled), otherwise you remain in the same resource, and you still have the possibility to use the **Restore Block** item.

This dialog box will appear as soon as you decide to leave a block that is modified and saved.

The actions that can trigger the apparition of this dialog are: **changing the current address** (menu com-mands, click in the horizontal scroll bar), **changing the current resource** (ID), **switching from dump mode to disassembly mode**, and l**eaving the application**.

• Change Address

Use it, when you want to select a new address within the current resource.

The behavior of this dialog is the same as in **Change Resource**. However the default selected field is the address one.
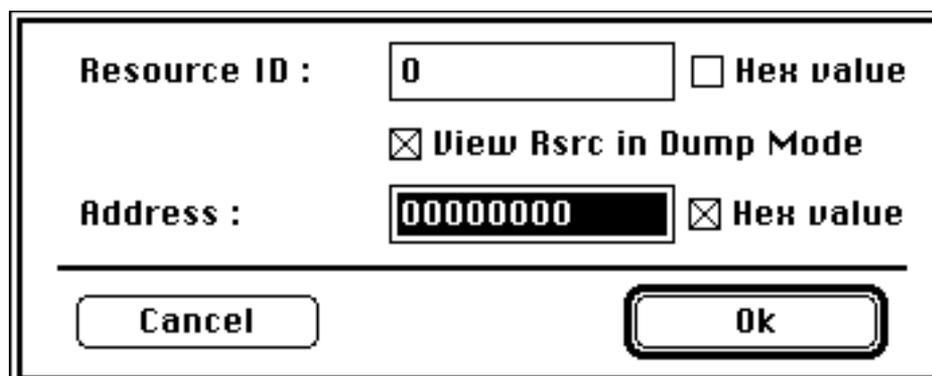
```
┌─────────────────────────────────────────────────┐
│                                                   │
│  Resource ID :    ┌─────────┐   ☐ Hex value      │
│                   │ 0       │                      │
│                   └─────────┘                      │
│                     ☒ View Rsrc in Dump Mode      │
│  Address :        ┌─────────┐   ☒ Hex value      │
│                   │00000000 │                      │
│                   └─────────┘                      │
│  ─────────────────────────────────────────        │
│  ┌──────────┐              ┌──────────┐           │
│  │  Cancel  │              │   Ok     │           │
│  └──────────┘              └──────────┘           │
└─────────────────────────────────────────────────┘
```

• <u>Dump mode</u>

Allows you to switch between dump and disassembly mode within the current resource.

• <u>Patch Mode</u>

Enters/leaves the patch mode, which allows you to write into the resource. **Be careful...**

If patching is allowed, a ** Patch ** menu appears (it is an empty one and it is only for information purpose).

If you are in dump mode, only hexadecimal values or ASCII characters can be entered. If you are not in dump mode, only assembly instructions can be entered (see the assembler for more information).

• <u>Write block</u>

Writes the current block into the current resource.

Note: if you change the current block or the current resource before saving it, all the modifications are lost i.e. the resource is not changed.

• <u>Restore Block</u>

Restores the old values (those before the write operation).

MISC MENU DESCRIPTION

• <u>Options</u>

Allows you to change the default mode for finding strings and printring or saving resour-ces. You can select the addresses range, and the resource IDs range (of course, resource IDs must be valid ones).

The **Default** button restores the default values i.e. actions occur within the current resource, from its beginning to its end. **Note:** Each time you select a new resource, the options are set to their default values.

• **Magic Return**

Each time you call **Change resource**, **Change address** or each time you select an address or a symbol by double-clicking, the previous address (offset[rsrc number]) is recorded in a LIFO stack.

This stack is able to store **ten** different addresses. Thus choosing **Magic Return** allow you to return to previous addresses within the limit of the ten stored adresses in the stack.

This item is very useful when there is a JSR instruction somewhere in a resource and you want to see the corresponding code, but you don't want to loose the address where the JSR was.

• **Mini Calc**

Emulates a mini calculator in hexadecimal and/or decimal mode. This calculator is word or long-word oriented (a byte is interpreted as a word). The available operators are:

- [two operands]   :   + (ADD), - (SUB), * (MUL), / (DIV), % (MOD),
- [two operands]   : | (OR), & (AND), > (ASR), < (ASL), ^ (XOR),
- [one operand (the top one)]   :   ~ (NOT), = (EQU).

The '=' operator is useful to convert number (decimal to hexadecimal or hexadecimal to decimal conversion).
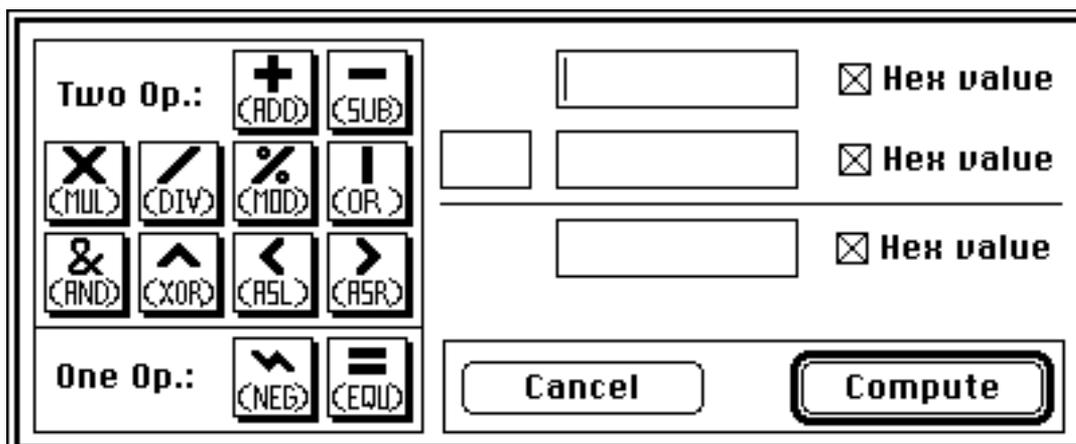
• **Show Global Variables**

Allows the window which contains the list of global variables (GLOBAL VARIABLES WINDOW) to become the front window. If the window is not visible, then it is first displayed.

If you want to view a low memory address, you can do it simply by pressing the command key and double-clicking on the address part ($xxx:size).

• <u>Show Toolbox Traps</u>

Allows the window which contains the list of toolbox traps (TOOLBOX TRAPS WINDOW) to become the front window. If the window is not visible, then it is first displayed.

• <u>Show OS Traps</u>

Allows the window which contains the list of OS traps (OS TRAPS WINDOW) to become the front window. If the window is not visible, then it is first displayed.

• <u>Available Memory</u>

Allows the window which contains the value (in Kbytes) of the free memory (AVAI-LABLE MEMORY WINDOW) to become the front window. If the window is not visible, then it is first displayed.

• <u>Help</u>

Presents a help panel. In fact a condensed version of this documentation.

SEARCH MENU DESCRIPTION

```
Search
Find String...   ⌘F
Find Trap...      ⌘T
..........................
Find Again        ⌘G
```

• Find String

If you choose the **ASCII** checkbox, the string is interpreted as an ASCII one, otherwise it is interpreted as an hexadecimal value, and in that case, you can find a word (two bytes) or a long (four bytes).

```
Enter string, please
[                    ]
☐ ASCII search
( Ok )      ( Cancel )
```
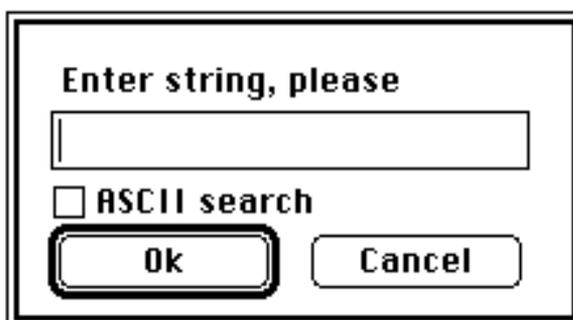
When the application finds a matching string, it displays the following dialiog box:

```
The String you search is located in
resource ID : 2 in resource ID=
00000038

( View )    ( Next )    ( Leave )
```
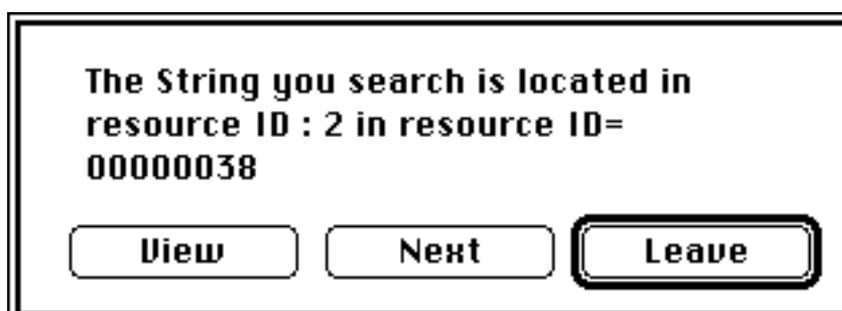
Then, you can decide to view the resource at the corresponding adress (resource ID, address), to search for an other possible string, or to stop the searching process.

Note: the current mode (dump or disassemblage) is used when you choose the **view** button.

• Find Trap

Trap name search (always in **ASCII** mode).

It is possible to search for a trap with some bits set (bit 9 and 10 for OS-traps [SYS, CLEAR, MARK, CASE, HFS, ASYNC, IMMED, A0RESULT] and bit 10 for TB-traps [AUTOPOP]).

Examples:

   \x(NewHandle,CLEAR),
   \x(NewHandle,CLEAR,SYS),
   \x(GetTrapAddress,NEWTOOL),
   etc.

```
┌─────────────────────────────────────────┐
│                                         │
│   Enter trap name.                      │
│                                         │
│  ┌───────────────────────────────────┐  │
│  │ |                                 │  │
│  └───────────────────────────────────┘  │
│                                         │
│  ╔═══════════════╗   ┌───────────────┐  │
│  ║      Ok       ║   │    Cancel     │  │
│  ╚═══════════════╝   └───────────────┘  │
│                                         │
└─────────────────────────────────────────┘
```
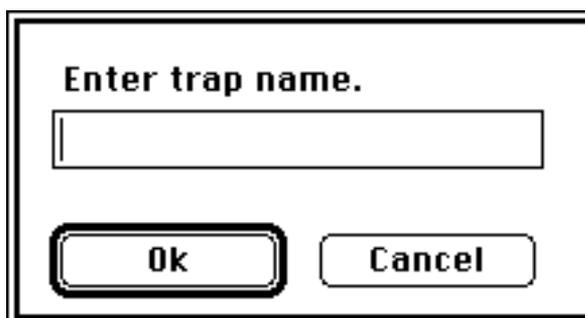
Note:
• If you want to search for the \x(_HOpen) trap, you must enter \x(_Open,HFS), etc.

• As the \x(_HFSDispatch) trap value can be 0xA060 or 0xA260, this trap has been renamed:
     \x   (_FSDispatch) when the value is 0xA060 and,
     \x(_FSDispatch,HFS) otherwise.

• <u>Find Again</u>

As soon as a search string or a trap string is defined, this item becomes available. But if you change the current window, it is disabled since the search parameters are no longer valid.

MC680XX MENU DESCRIPTION

```
MC680xx
┌─────────────────────────────┐
│ Show Format        ⌘5       │
│ Show Operation     ⌘6       │
│ Show Codeop        ⌘7       │
└─────────────────────────────┘
```

• <u>Show Format</u>

Allows the window which contains the 680xx format to become the front window. If the window is not visible, then it is first displayed.

Assembly syntax	sub-field mode	sub-field register	addressing category				68020
			Data	Memory	Control	Modif	
Dn	000	reg n°	*			*	
An	001	reg N°				*	
(An)	010	reg N°	*	*	*	*	
(An)+	011	reg N°	*	*		*	
-(An)	100	reg N°	*	*		*	
d(An)	101	reg N°	*	*	*	*	
d(An,Ri)	110	reg N°	*	*	*	*	(x)
(d[An],Ri,od)	110	reg N°	*	*	*	*	X
(d[An,Ri],od)	110	reg N°	*	*	*	*	X
$xxxx	111	000	*	*	*	*	
$xxxxxxxx	111	001	*	*	*	*	
d(PC)	111	010	*	*	*		
d(PC,Ri)	110	011	*	*	*		(x)
(d[PC],Ri,od)	110	011	*	*	*		X
(d[PC,Ri],od)	110	011	*	*	*		X
#value	111	100	*	*			

This panel is very useful, if you want to build an instruction on the fly (good luck!). Assuming you know its internal representation, see Codeop window for some examples.

• __Show Operation__

Allows the window which contains the 680xx operation to become the front window. If the window is not visible, then it is first displayed.

16 bits (15-12)	Operation
0x0...	Bit manipulation/MOVEP/Immediate
0x1...	Byte transfert
0x2...	Long transfert
0x3...	Word transfert
0x4...	Misc (NOP/LINK/RTS/TRAP/JUMP/...)
0x5...	ADDQ/SUBQ/Scc/DBcc/TRAPcc
0x6...	Bcc/BSR/BRA
0x7...	MOVEQ
0x8...	OR/DIV/SBCD
0x9...	SUB/SUBX
0xA...	(Unassigned, Reserved)[Mac traps]
0xB...	CMP/EOR
0xC...	AND/MUL/ABCD/EXG
0xD...	ADD/ADDX
0xE...	Shift/rotate/Bit Field
0xF...	(Not Used,or Coprocessor Interface)

• **Show Codeop**

Allows the window which contains some 68000 instructions to become the front window. If the window is not visible, then it is first displayed. Update it if you need.

MOVE : `0` `0` `Size` | Destination Reg | Mode | Source Mode | Reg

Size : .B = 01; .W = 11; .L = 10

PEA : `0` `1` `0` `0` `1` `0` `0` `0` `0` `1` Effect. Addr

PEA d(A5): 0x486D; **PEA d(PC)**: 0x487A

LEA : `0` `1` `0` `0` Reg `1` `1` `1` Effect. Addr

JSR : `0` `1` `0` `0` `1` `1` `1` `0` `1` `0` Effect. Addr

JSR d(A5): 0x4EAD; **JSR d(PC)**: 0x4EBA

JMP : `0` `1` `0` `0` `1` `1` `1` `0` `1` `1` Effect. Addr

Bcc : `0` `1` `1` `0` `Cond` 8 bits displacemt

Cond : "T F HILSCCCSNEEQVCVSPLMIGELTGTLE"

RTS : 0x4E75 | **NOP** : 0x4E71 | **BRA = BT** (0x60dd)