LES Debugger AmigaGuide Documentation

COLLABORATORS

	TITLE : LES Debugger AmigaGuide Documentation	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 20, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

LES Debugger AmigaGuide Documentation

1.1 Main Menu

LES Debugger v1.21
==========

Written by Stephen McNamara
of
Leading Edge Software

Original debugger by
Mark Sibly of Acid Software

Release Date:
4/1/1995

Note
See the section New Stuff for additions and changes in this release
of the debugger.

This program is OS2.0+ only and requires
the reqtools.library

1.2 The Index Page

LES Debugger v1.21
==========

Index Page

Please select the topic you wish to read :

Legal stuff
About the debugger
Debugger windows
Installation
New stuff for v1.21-1.0

IMPORTANT NOTE:
Installing the Debuglib.obj file into BlitzLibs:System/ is no longer optional. IT MUST BE INSTALLED there. See the section on Installation for more information.

1.3 Disclaimer, Copyright Notice & Public Domain Notice

Disclaimer, Copyright Notice & Public Domain Notice

God I hate formality..... here we go anyway.....

This program is public domain. People may spread it to anyone they like, as long as all the relevant files are included when it is spread. Relevant files include the actual program and all documentation files supplied with the program. PD libraries etc may not charge more than a reasonable price for copying and disks. This program may not be distributed in a commercial package without the permission of the author (Stephen McNamara).

This program, and all associated files are the work of Stephen McNamara. The main debugger program is based on code by Mark Sibly of Acid Software and is used with permission. This program file may not be disassembled or modified in anyway and the docs must not be altered.

The author will not be held liable for any damage, however caused, whether indirect or direct as a result from the (mis)use of this software.

This program uses the reqtools.library which is (C) Nico Francois.

1.4 About the debugger

About the debugger

This program replaces the default debugger supplied with B.U.M. 7. It is only of use for people who have version 1.9 of Blitz2 (which was supplied with B.U.M. 7) or later. It cannot be run stand-alone - it can only run from inside Blitz2.

The standard debugger supplied with Blitz2 is very basic in that it only allows you to step programs and evaluate variables. It opened a window on the Workbench screen, in which it had program listing and some program control gadgets. This debugger goes for a full MonAm (the disassembler/debugger supplied with Devpac) approach in that it opens its own screen. On this screen are a series of windows that allow you to watch what you're program is doing whilst running it, see the section

WINDOWS for information on them.

There are a few things that should be pointed out about using this program. Firstly, it is big and memory hungry, thus you will need to have at least 1meg to use it alongside Blitz2. Also, at the moment anyway, the program uses the ReqTools library via Neil O'Rourke's blitz library. You will thus have to have reqtools.library installed on your machine.

1.5 Windows In The Debugger

Windows In The Debugger
---

* Program Control Window

This window contains the gadgets that were found in the original debugger plus a few extra. The extra ones are:

BLITZ: Click to view blitz mode display (if program is in blitzmode). Press a mouse button to return to debugger.
REGS : Toggle register window (open/closed)
MEM1 : Toggle memory window 1 (open/closed)
MEM2 : Toggle memory window 2 (open/closed)
VARS : Toggle variable trace window (open/closed)
ASM : Toggle disassembly window (open/closed)
COP : Toggle the copper disassembly window (open/closed)
PS : Toggle procedure control on/off (highlighted=on)
See New Stuff for information.
BC : Toggle Blitz mode control on/off
See New Stuff for information.

Old gadgets are:
STOP : Causes program execution to stop and the debugger top take over
STEP : Execute the current instruction (whilst program stopped)
SKIP : Ignore current instruction and move to the next one
TRACE: Trace the program (like running, except the debuggers windows are updated. Your program will run very slowly in trace mode, but you'll be able to see exactly what is going on.
RUN : Resume normal speed execution of the program
< : Move backwards through history buffer
> : Move forwards through history buffer
EXEC : Execute a basic instruction
EVAL : Evaluate a value (e.g. variable, addition etc)

Note: All gadgets have keyboard shortcuts associated with them. See the menu option 'Keyboard Shortcuts' inside the debugger for the list of keys.

Closing the program control window causes the debugger to quit and your program to end.

V1.2
This window now displays the current mode of your program inside its title bar. Possible modes are: AMIGA, QAMIGA and BLITZ.

* Register window

This window allows you to view the contents of the 8 data and address registers. It will mainly be of use to people who have a little knowledge of assembler. As well as the register contents, you are also shown the 6 words that the address registers point to. The status register flags are also shown in this window, as is the program counter (which will be mainly used to make the disassembly window open at the current instruction).

* Source window

This window shows you the source code of the program currently being traced. Whilst it is active, you can use the cursor keys to move up and down through your code.

* Memory windows

This windows allow you to look through memory. They are both identical in that they show memory in both hex and ascii form, there are two so that you can keep tabs on different addresses at the same time.
The windows can be moved through memory be using the cursor keys (and shift for page scroll). You can also press the 'm' key to enter an address to jump to. Valid entry modes for the address are:

  * Effective address: e.g. $dff0,565560
  * Address or data register: e.g. a5,d0,d2
  * Address/data register indirect: e.g. (a5) jumps to the address that a5 holds.
  * Label: jump to a given label, e.g. MyLabel:

V1.2
You can now edit memory directly inside the Memory windows.
See New Stuff for more information

* Object trace window(s)

These windows allow you to keep tabs on particular objects whilst your program is being traced or single-stepped.
They hold the structure of an object and will show you the objects items in the relevant format (e.g. strings will appear as actual strings, not addresses).
Open another object window by selecting a new one from the object menu strip.
Change the object number being traced by dragging the slider which is located on the right side of this window.

V1.2 You can have upto 10 object windows. You must now close them yourself - they are nor automatically closed

when you select a new object. See New Stuff for more information

* Variable trace window

This window allows you to trace different variables whilst your program is running. It has been completely fixed to work fine with all variable types (strings now work all the time!).
Add variables to the trace list by selecting the Add Trace... menu item. You'll be asked to enter the name of the variable to trace, then the type, and then the output mode (if relevant to your type). Output mode is either DECimal, HEX or BINary. Variables can be added whilst this window is closed, but you must open it yourself before tracing will start.
Variables can also be added directly by the program being traced by using the AddVarTrace library command (see documentation for the RIDebug Library).

NOTE Variables must be added *after* they have been defined.

New option: there is now a menu option that sorts all variables added into alphabetical order.

V1.2 As of this version, you can define the type of a variable with its name, doing this *validly* causes the type requester to be skipped. The following are now valid in the name requester (after the name of a variable):

```
$
.s
.b
.w
.l
.q
.f
```

* 680x0 disassembly window

This window displays disassembly of memory into 680x0 instructions. See New Stuff for more information (specifically: V1.0 new stuff).

* Copper disassembly window
V1.2
This window displaya dissassembly of memory into copper instructions. The following 3 instructions will be shown:

```
MOVE <$VALUE> <REGISTER>
WAIT <$xx,$yy> MASK <$xx,$yy>
SKIP <$xx,$yy> MASK <$xx,$yy>
```

The WAIT and SKIP instructions will sometimes have [BFD] written after them. If present, this means that the instruction has "blitter finish disable' on.

See New Stuff for more information.

* Search memory window
V1.21
This window allows you to control the debuggers memory searching facilities. From here, you can enter the start and end addresses for the search, the value to search for and the type of the value. You can search for the all 6 basic variable types: byte, word, longword, string, quick and float.

1.6 Installation/How To Use

Installation/How To Use

Installing is done is two parts, theses being installing the debugger executable and installing the library object files needed to let your basic programs communicate with the debugger.

Installing the debugger program couldn't be easier. Just copy the defaultdbug file into your Blitz2:Dbug/ drawer and load up Blitz2. Before you do this, though, copy the original version of the defaultdbug file to a storage area (or rename it to something like olddbug). Do this in case you don't like this debugger, or find it unusable for some reason (tell me about this though!). Now whenever the debugger is invoked (by running a program that crashes, or by pressing Ctrl-Alt-C) the new debugger will pop up and offer you its services.

There are two library objects that need to be installed, they both need to be copied to folders in your Blitzlibs: volume. The files are:

RIDebugLib.obj - copy to Blitzlibs:Userlibs/. This library gives you the commands that allow you to order the debugger about.

Debuglib.obj - copy to Blitzlibs:System/. This library in an Acid Software one that has been extended. It is required by the debugger. If you do not install this library, the debugger will not function at all.

switchlib.obj - copy to Blitzlibs:System/. This library in an Acid Software one that has been extended. It is required by the debugger. If you do not install this library, the debugger will not function at all.

When both these objects have been copied to the relevant folders, you should run the program MakeDefLibs which can be found on your original Blitz 2 disks. This will remake your Blitz2:deflibs file based on the contents of the Blitzlibs: volume. You should be aware that on floppy this will be very slow. When MakeDefLibs has finished just load blitz and you'll have access to the new debugger and the additional commands supplied by the RIDebug library. Please see the additional docs for the RIDebugLib

library for information about commands.

1.7 New Stuff

```
                          New Stuff for v1.21
                          -------------------
```

This version is a slight update to v1.2 (which was a beta version) that fixes some problems found. There were somw problems regarding blitz mode control that have been sorted out, as well as some library problems.

One major update has been made to the debugger in that you can now search memory for values and strings. This is done via the search window, which is opened by selecting the Search option from the menus.

There has also been an extension to the save configuration function that lets you save the position, size and status of every window in the debugger except object trace windows.

Finally, there are now several example programs that show off a few features of the debugger to you.

```
                          New Stuff for v1.2
                          ------------------
```

Quite a few additions and improvements have been made to this version of the debugger. New features have been added as well as improvements and bugs fixes (which have come about from reports by people who've tried the debugger out).

Summary of new features:

- Procedure control
- Blitz mode control
- Coplist Object tracing
- Copperlist Disassembly
- Multiple Object Windows
- Configuration Saving
- Memory Edit Mode
- Data/Address register edit
- Program mode display

Improvements made:

- Loads of bug fixes
- Speed improved for redrawing etc

```
Procedure Control
-----------------
```

This function allows you to skip over procedures as if they were single instructions when either STEPing or TRACEing your program. When running, this function has no effect. When you use this function, the procedure runs exactly the same except that it no longer causes the debugger to update its windows whilst it is executing.

In effect, what this function does it to, at the start of a procedure, which the debugger into RUN mode, and then at the end chuck it back into its previous mode.

Blitz mode control

This function is for all those people who're getting annoyed at the debugger constantly rebuilding the blitz mode display every time they step an instruction or evaluate an expression. With this function on, singular calls to the debugger like evaluating and execing no longer cause Blitz to redraw the display.

To understand this you have to have a little knowledge of what blitz does when it goes into blitz mode. When blitz mode is activated, Blitz disables multitasking, sets up a custom copperlist and owns the blitter. What this function does is to stop blitz setting a custom copperlist. The program being debugged *IS* in blitz mode, it just doesn't have the display. This may not be useful at times (for when you need, for example, to check to see where a blit is occuring) - you must decide yourself when and where to enable/disable it (default is disabled). That said, this function is very useful, since it stops all flashing, including flashes for opening object windows etc.

Coplist Object Tracing

The Coplist object type (used in the Display library) has been added to the debuggers list. The whole object could not be included, as it is fairly massive, so the object holds *most* of the object definition (all the important bits are in). This object was added mainly for use with copperlist disassembly.

Copperlist Disassembly

You can now disassemble copperlists directly from inside the debugger. Just select the option and you'll be given a nice window, where you can use cursors to scroll up and down, plus use the 'm' key to enter a start address. All this comes in useful when combined with the Coplist object, and some custom copperlist coding. Now its possible to check what Blitz is doing with your custom commands in the display library.

Multiple Object Windows

You can now have upto 10 independent object windows opened at once inside the debugger. This should be more than enough for everyone. A quick warning, though, with this many windows open, unless you have an accelerated Amiga, you will slow the debugger down a great deal. Like all the windows in the debugger, you should only have open the windows that you are actually using.

Because of the additional object windows, selecting a new object from the menu strip *will* not replace any current windows. You must close individual object windows yourself after you have finished with them. Also

note that it is perfectly allowable to have, for example, two bitmap windows open. Thus allowing you to trace more than one object of each type at once.

Configuration Saving

A default config file for the debugger can be saved now from inside the debugger. Inside the config file, are size, position and status information for the following windows:

```
Source    - MUST BE OPEN!
Registers
Control   - MUST BE OPEN!
Memory1
Memory2
```

When the debugger first runs, it will look for the volume ENV:. If it finds it it'll then attempt to load the file ENV:BB_DBug.prefs. If it can find this, your prefered window layout will be loaded, otherwise the default layout will be used.

Please note that the debugger will not bring up any requesters if it cannot find the file. When the debugger saves its preferences file, it saves to both ENV: and ENVARC: so both of these should be made available if you want to save your config (this will probably mean floppy users having to put there boot disk into a drive).

At the moment, the configuration file is 64 bytes in size.

Memory Edit Mode

Whilst a memory window is active, pressing <SPACE> will put you into edit mode. Here you can move a cursor round the windows hex display, and edit using 0-9/A-F on the keyboard. Press <SPACE> again to exit, or deactivate the window by selecting another (or pressing TAB).

NOTE: You should be careful what you change in the edit window. Only change/edit memory that your program owns! Do not go messing around through memory as you could cause a crash.

Register Edit

Clicking on registers d0-d7/a0-a6 in the register window will now bring up a requester into which you can type a replacement value for the register. The value can be typed in as a number or as a 4 character string, surrounded by quotes. Thus the following are allowed:

```
$fff0
12466
%1001111111100000
"CMAP"
"ANHD"
```

Program Mode Display

The current mode of your program is now shown in the title of the Program Control Window. The mode will be one of AMIGA, QAMIGA or BLITZ.

SPEED.....

Speed improvements have been made to several areas of the debugger. The main improvements have been in text printing and window updating. By using the Configuration Saving, though, you can speed up debugger initialisation by closing extra windows.

Odd Addresses

Odd addresses can now be displayed properly in the register window and the two memory windows. Also the memory windows can be moved one byte at a time using the left and right cursor keys.

Variable Window

You can now sort all currently traced variables into alphabetical order by selecting the 'Sort Names' menu item.

Bug Fixes

Just a few bugs that have been fixed:

- Hex display in memory and register window was sometimes displayed incorrectly due to a mistake in conversion. This effected odd addresses only.
- BLITZ gadget has been sorted so that clicking out of the BLITZ screen will not immediately reactivate the gadget.
- Loads of OS2.0/68000 compatibility problems sorted (e.g. peek.l at an odd address).
- Wrong font used inside the Object window (used default WB font rather than topaz.8).
  Problems with string printing inside the object window sorted

New Stuff for v1.0

So whats new in this release of the debugger? Well there are loads of changes and updates that have been made to the debugger, these being:

680x0 Disassembly

You can now disassemble your blitz basic programs directly into 680x0 (68000 to 68020) instructions. When runtimes are on in a blitz basic program, the TRAP instruction is used before every basic instruction in the program. When the disassembler finds the correct trap (#1 is used by blitz) it automatically replaces this instruction, plus the 8 data bytes after it, with the line of blitz basic code that the following 680x0 belongs to. This doesn't always work, though (for some reason - I'll blame blitz for this :)) so you *might* not always display the correct line of source (multiple commands on one line, plus blank or commented lines bugger up).

The cursor keys, plus [SHIFT] can be used to move up and down through memory when the disassembly window is active. Also, pressing 'm' will bring up a requester where you can enter an address to move the disassembly window to.

Blitz Gadget

Clicking this gadget allows you to view the blitzmode display of the program being traced. When you have finished viewing the display, press the mouse button to return to the debugger.

AutoEVAL

The EVAL button in the debugger allows you to evaluate any variables, find label addresses etc. AutoEVAL takes this a step further and allows you to enter a line that will be evaluated after *EVERY* instruction is single stepped (using the STEP gadget). To enter a line to AutoEVAL just select the menu option, when a non-null string is entered, the AutoEVAL will be enabled and will display its output after every step at the bottom of the source window.

An example of how this can be used is to evaluate an expression like mymap(x,y) after every instruction is stepped. This can be used to easily keep track of what your program is doing. The output of AutoEVAL for this example instruction will look like this:

```
mymap(x,y)=50
```

You can only AutoEVAL one line at a time. Also, you should note that the AutoEVAL does degrade the speed of your program, since the EVAL command itself is slow.

Remove the AutoEVAL by selecting the menu option and clearing the command string.

Standard Gadgets

The menu items that mirror the standard gadgets in the program control window have been removed. They have been replaced with keyboard shortcuts for the gadgets that do not require the right amiga to be pressed to activate them.

New Objects

The list of objects that can be traced inside the debugger has been extended to include all the main objects I could think of. The full list of objects is:

Bitmap (extended to hold the *full* bitmap definition)
Blitzfont
Buffer
File
Intuifont
Module
Palette
Queue
Slice
Sound
Sprite
Shape
Stencil
Tape
Window

RIDebug Library

This library allows you to give the debugger 4 basic instructions. It mainly allows you control over the variable tracing facilities inside the debugger. Please see the RIDebugLib docs for more information about how to use the library.

Bug Fixes

Too many to mention :)

1.8 Problems...

Problems...

A couple problems that I know of that people have had with the debugger in the recent past. As more problems are reported, I'll extend this list so that it becomes like a FAQ for the debugger.

A quick plea to possible bug reporters:

When you report bugs to me about this program, can you please include details of the machine the debugger is running on, including amount of memory and an 'add-ons' you may have, and the operating system version number you are using. The more I know about what situation the debugger is running in, the easier it'll be for me to track down possible bugs or help you with setting it up.

Prob 1: I cannot trace any variables. The debugger always responds with "Variable not found" when I do.

Solution: There are two main reasons for this happening:

a) You are typing the variable name wrong. Remember that Blitz names are case sensitive, also make sure that any strings defined with a '$' instead of '.s' must be added with '$' after there name.

b) You are being affected by a bug in the current version of Blitz. The debugger uses the EVAL function (the gadget on the debugger screen) that Blitz provides to locate a variable in memory. What happens, though, is that if a particular library is not included in your program when you compile, you CANNOT access the EVAL function. Thus the debugger gets 0 back from its locate request, and tells you the variable doesn't exist.
To get round this bug, you must have a command like Print used in your program. The command DOESN'T have to run, it is only there to make sure that Blitz includes the correct library when compiling your program. What you can do is just add 'Print ""' to the end of your program (this is what I do)

Prob 2: I cannot trace objects. I just get an 'No object found' message in the object window.

Solution: Again, this problem is caused by one of two things:

a) The objects don't actually exist in your program. For example, if you never use the Blitz Slice library to create Blitz displays, you won't be able to look at slice objects since there won't be any in your program.

b) The Blitz bug crops up again. This function is effected in exactly the same way as Variable tracing by a bug in the EVAL function. Add a print command to your program to solve this problem.
See Problem 1 for more info about this bug.

Prob 3: CTRL-ALT-C doesn't make the debugger pop up.

Solution: Quite a few for this one:

a) Your program may have crashes. Check this by going to the debugger screen (is possible) and trying to evaluate an expression or variable. If you don't get an answer back - your program has gone bye-byes.

b) The debugger has crashed. This may or may not cause your program to crash. Most likely a crash will result eventually.

c) You have a ZAPPO CD-ROM drive installed (like me!). I have had problems with using programs that want to allocate CIA resources with the driver software loaded. I get a lot of 'Resource already in use' messages when I run particular software. When I have the drive installed, Blitz refuses to acknowledge any keyboard responses (except through windows). I have tracked down the fault to the fact that the CD driver software is in memory, but I do not know what is causing the problem.

1.9 Version history

```
                              Version history
                              ---------------

For those interested in the development of this program, here's the version
history for it as taken from the top of the source code.  Pretty pointless
putting it here but what the hell - somone might read it ;-).

18/1/95
   - Release VERSION: 1.21
17/1/95
   - Fixed bug: when pressing escape to quit the memory search,
     the program exited - the ESC key hit was being sent as an
     event.  Fixed by flushing all keyboard events (1024) from
     the event list.
11/1/95
   - Search memory in and mostly working.
     Searches for: Bytes, Words, Longs, Quicks & Floats.  Strings
     are to be added.
   - Windows automatically activate when opened AFTER end_init
     goes non-zero (at the end of initialisation)
   - CTRL \ now closes the current window (except: #_winSource,
     #_winBackdrop & #_winControl)
   - Slight changes to RedrawVarWin{} - now uses FPrint instead
     of window printing
   - Config file now saves Var/Cop/680x0 windows so these can
     now be open by default.
   - Bug fixes in blitz mode handling
7/1/95
   - BETA TEST: Version 1.2
   - Config file now 2 bytes bigger.  Saves ProcStatus and
     blitzcontrol at end of file.
   - NEW BLITZ MODE HANDLING!  Can now stop Blitz mode display
     being recreated whilst stepping/tracing/EXECing/EVALing
6/1/95
   - Procedure stepping now controlled by debuglib library.  The
     library will call d_prochandler if variable procstatus is
     non-zero
   - Extra gadget PS: procedure tracing status.  If highlighted,
     procedures are treated as a single command.
5/1/95
   - New debug command causes copper disassembly window to
     open at the given address
   - When adding a variable trace from the debugger you can
     now specify the type directly, e.g. a.s,b$,d.q etc
   - Debug messages have changed: addresses are given as one
     longword rather than a hex string
4/1/95
   - VERSION: V1.2
   - Edit mode toggled by pressing <SPACE> *WHILST*
     a memory window is active
   - Program running mode now shown in control window title
   - Exiting MemEdit mode now redraws both windows (in case
     the windows' addresses overlap)
   - COPPER gadget added - Copper Dis... menu item removed
```

```
   - PREFERENCES FILE NOW SAVED TO ENV: and ENVARC:
     Loaded from ENV: (if available!)
   - Preferences file now has a program version number
     in first byte and revision number in second byte.
     Then two byte pad before window data (could hold number
     of window definitions?)
1/1/95
   - d_eval{} and d_exec{} now increment stepcnt themselves
31/12/94
   - Bug when moving mem window to label address. Was only
     adjusting MEM1 instead of MEM1 and MEM2
   - TAB key now cycles all windows properly
   - MemEdit now exits when window gets deactivated
   - Can now move mem windows into negative numbers from 0
30/12/94
   - Menustate command was trying to deactivate an unitialised
     menu item.  Command has been removed
   - Fucked up .xtra file..... causing gurus (not enough
     windows.......)
26/12/94
   - Trial custom font printing
24/12/94
   - VERSION: V1.1
   - Object window: data types now shown on left of central
     line and are not rewritten every refresh (only during
     full redraw)
   - Added copper instruction disassembling
   - New command in library: GetCopIns$ - convert copper
     instructions into strings
   - Library changes:
     - D_CheckNMove now takes an optional count parameter
     - Changed the way D_GetSR works
   - Fixed BLITZ gadget disable/enable (AGAIN!)
   - Capital letter for object names
   - Objects can now contain quick and float types
   - Added coplist object type - whole definition won't fit
     inside a 256 tall screen though :)
   - Added checks in NewTypeWindow{} to make sure the window
     will never be larger than can fit on the screen
21/12/94
   - CheckNMove syntax change: now GIVE address to take 4 bytes
     from rather than giving the actual longword of data
20/12/94
   - Added 'Sort names" menu item for variable window - sorts
     names into alphabetical order
   - Click in register window to modify d0-d7/a0-a6
   - Memory window have been widened to 208 pixels to allow
     odd addresses to be displayed properly
   - Left/Right cursor now move memory windows 1 byte at a
     time
19/12/94
   - Bug when workbench screen was 640 wide:
     Control window was opened at x=#_winControl rather than x=0
17/12/94
   LOADS of stuff :)
   - Bug fixed in 'hexwordodd - wrong mask values
   - All windows now use the winstatus() array to say whether
```

```
     they are open or not
   - Default window layout can be saved into
     Blitz2:Dbug/dbug.prefs.  Only windows 0-5 though
15/12/94
   - Trial thingy: mybuffer is loaded from SetHandler{} rather
     than the Statehandler_ as before
   - NOTE: Register window contents aren't always valid when
     program is first run.  The statehandler_ routine MUST be
     called before they become valid
14/12/94
   - RELEASE 1.0
   - Removed 'Disassemble...' menu item
   - Fixed 'Del Trace...' to make it update the variable window
   - Fixed font cockup in Object window - printing was being
     done with default font rather than topaz8.
     Thanx to Rupert Henson for the bug report
12/12/94
   - Fixed UpdateDisAsm{} to show blitz2 code properly
2/12/94
   - Removed standard nine gadgets from menu strip - keyboard
     shortcuts stay the same (but obviously minus the RAmiga)
   - Removed all occurences of CPUCls
   - Added checks in RedrawMem and HexWord{} for odd addresses
     (for 68000 compatibility) - thanx S.Le for the help with
     identifying that as a prob.
   - 'Bumped' OS version number to 39 for use of ScreenTags
     command.
24/11/94
   - Added load of new object types
   - Added option to display extra info for string tracing
       (can display length and maxlen of string,
        AddVarTrace: use output=2 for extra info)
   - STRING tracing is now sound :)
20/11/94
   - Added Del Trace... menu option
   - Added Auto EVAL - does an auto evaluation after every STEP
16/11/94
   - Added DisAssembly window.  Can be moved with cursors [+shift]
     or positioned using the EVAL function
15/11/94
   - Changed HexWord{} to improve efficiency ;-)
 6/11/94
   - Added menutitle: Variables with options to add/del traces
 5/11/94
   - Variable trace window put in - no definable traces yet
```

1.10 Tell me what you think

```
                          Tell me what you think
                          ----------------------

  Please mail me with what you think of this program.  Mail any
comments/questions/moans/money offers to:

                          SIS3149@SIS.PORT.AC.UK
```

When I originally released this program, I said that full source would be available for it. Source, though, hasn't been available for a number of reasons. The biggest one is that the debugger relies on certain Blitz 2 libraries by Acid software that I've modified too make them more useful. I have sent these changes to Simon Armstrong at Acid but haven't received a 'seal of approval' for them so I can't distribute them with the debugger - I'll have to wait to see if the changes find there way into the next blitz update.
Another reason for source being unavailable is that I would assume that most people will be quite happy with the debugger as it stands, and would not have enough time to work out how it works anyway. Thus the benefits of them having the source will be very small. There will be a number of people with the knowledge, or desire, to look at the source code for this program. There may even be people who wish to customize it to there programs (to make it easy to debug *specific* programs). These people should contact me directly to discuss source, they will most probably be able to get it off me, as well as some details of how it works (I'll be doing some internal docs for the debugger for people interested in customizing it).

1.11 Anything new on the way?

Anything new on the way?

This program is in continuous development at the moment. This means that as I think of them, new ideas get put into the program. Thus ideas for the debugger are coming from just one person - me - rather than a selection of people. What I would like is for people who either use this program, or have looked at it to, to make suggestions about what is needed in it. This sort of program grows best when people use it and from using it, discover shortcoming in it. Please, if you have any suggestions, then email me at the address given above.

Quick thanx to all the people who have mailed suggestions to me, especially as some of the suggestions genuinely make the program better. A few credits for suggestions should go to:

Steve Matty - Editing memory (although I was thinking about this anyway - he just *made* me put it in :)).
Saving prefs to ENV: and ENVARC:.
Loads of other small things.

Steve Innel - Multiple object windows.

Rupert Henson - Wanting the default window positions to be different (you can define them yourself now Rupert! ;-)).
Also for wanting font sensitivity..... (thats not in yet though)

Son H. Le - Loads of suggestions regarding OS2.0/68000 compatibility.
Wanting to be able to step over statements/functions when tracing (this can be done now Son!).

Lastly a quick mention of Simon Armstrong at Acid software who supplied us with library code for blitz (which made this debugger to be as good as it is) and the disassembler.

1.12 Thanx go to the following

Thanx go to the following

OS2.0/68000 testing and fixing:
Son H. Le

Suppling the disassembler code:
Simon Armstrong (Acid software)

Help and criticism:
Steven Matty
Steven Innell
And loads of chappies on the Blitz-list

For feedback and bug reports:
Son H. Le
Rupert Henson
And the others I've forgotten :)

Offers of money:
No-one yet :)

Requests to publish the debugger with Blitz:
None yet :)

Quick hello's to:
Martin 'it doesn't work with StarWoids' Kift
Jurgen 'AmigaGuide' Valks
Mark 'Virtual Worlds' Tiffany
Nigel 'Cascade' Hughes
And everyone on #amiga/Blitz-list