General Notice

When using this document, keep the following in mind:

1. This document is confidential. By accepting this document you acknowledge that you are bound by the terms set forth in the non-disclosure and confidentiality agreement signed separately and /in the possession of SEGA. If you have not signed such a non-disclosure agreement, please contact SEGA immediately and return this document to SEGA.

2. This document may include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new versions of the document. SEGA may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time.

3. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without SEGA'S written permission. Request for copies of this document and for technical information about SEGA products must be made to your authorized SEGA Technical Services representative.

4. No license is granted by implication or otherwise under any patents, copyrights, trademarks, or other intellectual property rights of SEGA Enterprises, Ltd., SEGA of America, Inc., or any third party.

5. Software, circuitry, and other examples described herein are meant merely to indicate the characteristics and performance of SEGA's products. SEGA assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples describe herein.

6. It is possible that this document may contain reference to, or information about, SEGA products (development hardware/software) or services that are not provided in countries other than Japan. Such references/information must not be construed to mean that SEGA intends to provide such SEGA products or services in countries other than Japan. Any reference of a SEGA licensed product/program in this document is not intended to state or simply that you can use only SEGA's licensed products/programs. Any functionally equivalent hardware/software can be used instead.

7. SEGA will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's equipment, or programs according to this document.

NOTE: A reader's comment/correction form is provided with this document. Please address comments to :

SEGA of America, Inc., Developer Technical Support (att. Evelyn Merritt)
150 Shoreline Drive, Redwood City, CA 94065

SEGA may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

SEGA OF AMERICA, INC.
Consumer Products Division

Sound Programming Debugger

User's Manual

Rev. 0.93

Doc. # ST-65-R1-031494

READER CORRECTION/COMMENT SHEET

Keep us updated!

If you should come across any incorrect or outdated information while reading through the attached document, or come up with any questions or comments, please let us know so that we can make the required changes in subsequent revisions. Simply fill out all information below and return this form to the Developer Technical Support Manager at the address below. Please make more copies of this form if more space is needed. Thank you.

General Information:

Your Name _____ **Phone** _____

Document number ST-65-R1-031494 **Date** _____

Document name Sound Programming Debugger User's Manual

Corrections:

Chpt.	pg. #	Correction

Questions/comments: _____

Where to send your corrections:

Fax: (415) 802-3963
Attn: Manager,
Developer Technical Support

Mail: SEGA OF AMERICA
Attn: Manager,
Developer Technical Support
275 Shoreline Dr. Ste 500
Redwood City, CA 94065

REFERENCES

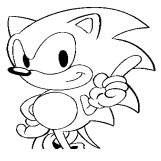
In translating/creating this document, certain technical words and/or phrases were interpreted with the assistance of the technical literature listed below.

1. *Dictionary of Science and Engineering, 350,000 words, 3rd Edition*
   Inter Press
   Tokyo, Japan
   1990

2. *Computer Dictionary*
   Kyoritsu Publishing Co., LTD.
   Tokyo, Japan
   1978

3. *IBM Dictionary of Computing*
   McGraw-Hill, Inc.
   New York, New York
   1994

CONTENTS



Introduction

SSBug is a symbolic remote debugger for the sound CPU (MC68EC000) on the Saturn development board for use in Apple Macintosh computers.

It supports the conventional character terminal interface and the standard Macintosh user interface, and debugging can be performed using either one or both of these interfaces.

SSBug can be used with the sound board alone, but when used in conjunction with the main CPU board, synchronized breaks with the main CPU are also supported.

1.0 Setup

Operating Environment

The following configuration is required to run SSBug.

- Apple Macintosh computer with a SCSI interface and KanjiTalk 7 or System 7 or later.
- Saturn development system (can be used with sound board alone)

SSBug distinguishes whether it is running under KanjiTalk 7 or any other Japanese environment (or more correctly, an environment in which a Japanese script is operating). It displays Japanese when it is possible to use Japanese and English when Japanese cannot be displayed, thus making it fully functional when running on an English system.

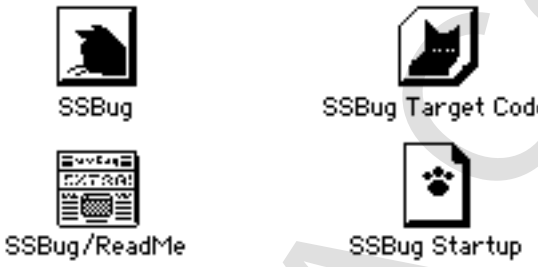
Setup

Refer to the Sound Development Manual regarding connection to a Macintosh. When only SSBug is used, connection can be made only via the SCSI port.

Also, since SSBug automatically distinguishes the SCSI ID of the sound board, no settings are required with respect to SSBug-side SCSI.

Installation

SSBug is supplied in the following four files:

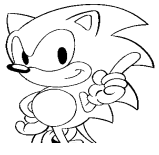


Descriptions of the files are given below.

- SSBug: the debugger program for the Macintosh
- SSBug Target Code: debugger program for sound board
- SSBug Startup: sample startup file described below
- SSBug/ReadMe: latest information regarding SSBug

To install, take the following steps:

1. Copy the SSBug Target Code file to the system folder on the startup disk by dragging it to the system folder.
2. Copy the remaining files to any desired location. This completes installation, and there is no need to restart the Macintosh.



Starting Up SSBug

SSBug can be started by double-clicking on its icon or by opening it up from the Finder. Since SSBug does not support Apple Events, SSBug will not startup even when double-clicking on files produced using SSBug or use drag&drop.

2.0 Definitions

Constants

The constants that can be used in SSBug are listed below.

- Hexadecimal constants: character strings of any desired length beginning with '$' and comprised of the characters [0-9A-Fa-f]
- Decimal constants: character strings of any desired length beginning with '\' and comprised of the characters [0-9]
- Binary constants: character strings of any desired length beginning with '_' and comprised of the characters [0, 1]

Evaluation is performed in the 32-bit mode without sign. Overflow amount is ignored, and the Hexadecimal prefix '$' can be omitted.

Variables

The variables that can be used in SSBug are listed below.

@0, @1, @2, ..., @9

(These are variables that the user can freely assign and reference with no 32-bit code.)

Unary Operators

The unary operators that can be used with SSBug are as follows.

+ positive sign
– negative sign
~ 1's complement

Binary Operators

The binary operators that can be used with SSBug are as follows.

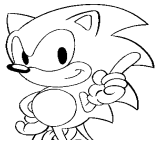
+ sum
– difference
* multiplication
/ division
& logical product
| logical sum
% remainder of the dyadic division
^ not-if-then logical sum
<< arithmetic left shift
>> arithmetic right shift

Symbols

In SSBug, symbols are referenced by attaching a '.' before the symbol name. The characters that can be used as symbols are listed below.

[A-Z], [a-z], [0-9] and $, _, ?, @

SSBug always distinguishes between upper case and lower in its symbols.



Wild Cards

Wild cards are meta characters which are used when specifying multiple character strings. The following wild cards can be specified in SSBug.

*	Matches a character string of a specified length. However, characters appearing after these characters are ignored.
?	Matches any desired character.

Registers

The CPU registers are also referenced in SSBug by attaching a period '.' at the beginning. The following register names can be used.

D0 D1 D2 D3 D4 D5 D6 D7
A0 A1 A2 A3 A4 A5 A6 A7
SSP USP PC SR CCR

- Case is ignored in register names.
- When symbol names and register names conflict, the symbol name takes precedence.
- The character "." is added at the start only when referencing the value in a expression; the character "." is not specified during line assemble or when substituting values per X command.
- The value of CCR becomes the value of the lower byte of the SR register.

Functions

The SSBug's built-in functions are listed below.

@B*(expr)*:	contents of one byte at address *expr* on sound board
@W*(expr)*:	contents of one word at address *expr* on sound board
@L*(expr)*:	contents of one long word at address *expr* on sound board
@BP*(expr)*:	address of the *expr*-th software breakpoint
@EXTBL*(expr)*:	value of *expr* code sign-extended from word to long word
@EXTBW*(expr)*:	value of *expr* code sign-extended from byte to word
@EXTWL*(expr)*	value of *expr* code sign-extended from word to long word

- Case is ignored in function names (characters between '@' and '(').
- BP*(expr)* returns a value regardless of whether or not the breakpoint is enabled.

Terms and Expressions

"Term" refers to character strings whose values can be evaluated.

Hexadecimal constants
Decimal constants
Binary constants
Symbols
Register names
Functions

The terms and expressions on the previous page are the least divisible units with their own values, and are referred to as elementary terms. Elementary terms can make up terms as described below.

Terms: elementary term
unary operator + elementary term
elementary term + binary operator + elementary term
{+ binary operator + elementary term {+ ...}}

There can be any (arbitrary) number of white space characters (tab characters or space characters) between an operator and an elementary term.

"Expression" refers to character strings comprising an arbitrary number of terms and with values that can be evaluated.

Expressions: term
unary operator + term
term + binary operator + term {+ binary operator + term {+ ...}}

An arbitrary expression may be enclosed in parentheses, where the part enclosed in the parentheses is treated as a single term.

Order of Precedence of Operators

The order of precedence is as follows.

High	0	() elementary term
	1	~unary + unary-
	2	*/%
	3	binary + binary-
	4	>> <<
	5	&
	6	^
Low	7	|

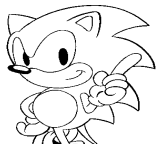
Statements

Character strings that specify operations in SSBug are called commands. Command names are formed by the following characters.

[A-F] [a-f] [0-9] ! # $ % & ( ) ? @

Some commands require an expression or an arbitrary character string as an argument. The character strings completed as command where include these required arguments are called statements. There must be at least one white space character between command names and arguments; further, multiple arguments must be separated by a comma ",". There may be any number of white space characters before or after commas.

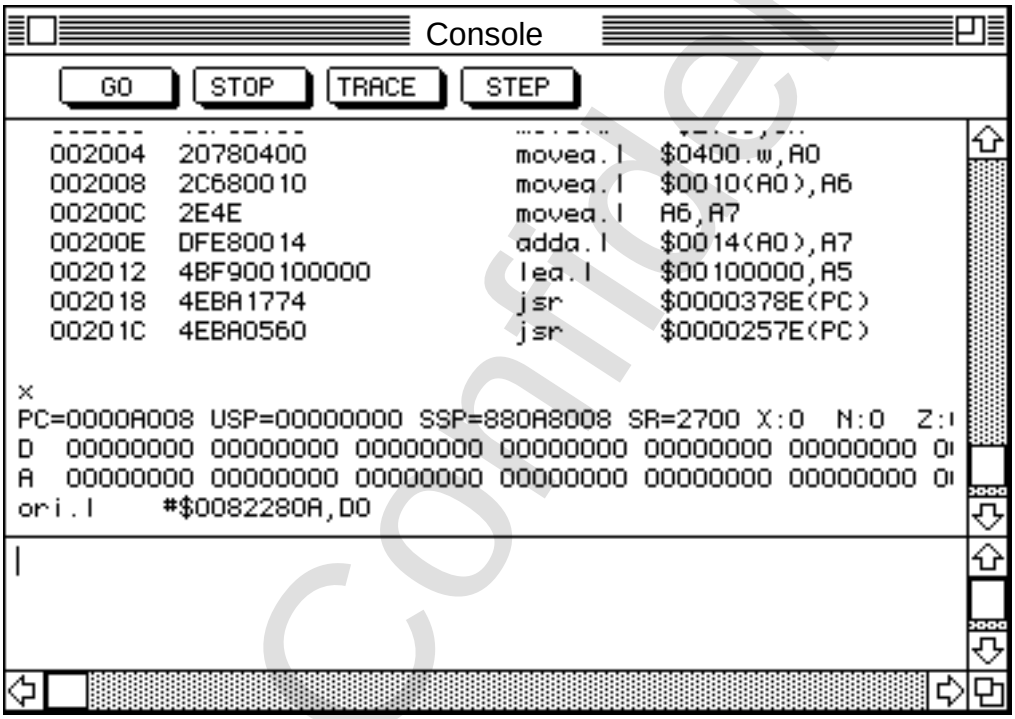
Statement: command argument{,argument{,argument{,...}}}



- In SSBug, the maximum length of completed character strings of commands or state ments is 255 characters.

3.0 Console Commands

The console command environment is an operating environment that uses a character terminal interface. Operation is progressed per a console window like that below by inputting commands from the keyboard. These are called console commands.



Refer to “Menu Environment” in the next chapter for more information on how to display console windows and operate them.

Terms used in the following explanations have the following meanings.

range specifies a range of addresses or values; 2 expressions separated by commas, where the first expression is referred to as “from” and the last expression is referred to as “to” and the range of values becomes from <= value <= to

regexp character strings including any selected number of the wild card characters ‘?’ and ‘*’; these are especially used when specifying multiple symbols.

In command notations, factors surrounded by brackets [] can be omitted.

Execution Control

G

Format: **G***[expr1][,expr2]*

Explanation: Executes a user program from address *expr1* and stops it at address *expr2*. When *expr1* is omitted, the value of the current PC is used. If *expr2* is omitted, execution of the user program continues unless it is stopped for some other reason.

T

Format: **T***[expr1][,expr2]*

Explanation: Executes *expr1* commands from address *expr2* and stops. The register contents and the command to be executed next are displayed with each command. If *expr2* is omitted, the value of the current PC is used. If *expr1* is omitted, "1" is used.

TU

Format: **TU***[expr1][,expr2]*

Explanation: This is the same as the **T** command except that the register contents of each command are not displayed.

S

Format: **S***[expr1][,expr2]*

Explanation: Executes *expr1* commands from address *expr2* and stops. The register contents are displayed with each command. If *expr2* is omitted, the value of the current PC is used. If *expr1* is omitted, 1 is used. This command differs from the **T** command with respect to the following points.

- bsr, jsr
  Treated as 1 command from the execution of a subroutine until return.
- trap, trapv, chk, A line trap, F line trap
  Treated as 1 command from exception processing until return.

SU

Format: **SU***[expr1][,expr2]*

Explanation: This is the same as the **S** command except that the register contents of each command are not displayed.

STOP

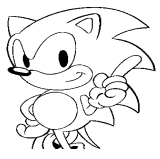
Format: **STOP**

Explanation: Stops the execution of the user program.

RESET

Format: **RESET**

Explanation: Resets the CPU registers and masks all the SCSP interrupt enable registers.



SCOPE

Format: **SCOPE***[arg]*

Explanation: Can periodically display in the register window the status of the CPU registers of the SSBug user program being executed. The SCOPE command performs this setting. Specify the following character strings for *arg*.
ON: switches display ON
OFF: switches display OFF
Omitted: displays the current settings

Register Operation

X

Format: **X**

Explanation: Lists the values of the CPU registers.

X

Format: **X***reg,expr*

Explanation: Sets the value *expr* in the CPU register *reg*.
A period is not inserted in the register name with this command.

Memory Operation

D,DB,DW,DL

Format: **D***[range]*
DB*[range]*
DW*[range]*
DL*[range]*

Explanation: Dump/displays the specified address area range to the display. The commands **DB**, **DW** and **DL** result in byte display, word display and long word display, respectively, and change the default display size to byte, word and long word.
The **D** command displays in the default display size.
Word display and long word display are possible from odd addresses in SSBug. Either 'from' or 'to' may be omitted per range. When 'from' is omitted, the next address where any of the commands **D**, **DB**, **DW** or **DL** is previously completed will result. When 'to' is omitted, 'from+$7F' will result.

F,FB,FW,FL

Format: **F***range,expr*
FB*range,expr*
FW*range,expr*
FL*range,expr*

Explanation: Fills the address range *range* with data *expr*. Fill means to put in as much as can be held. The **FB**, **FW** and **FL** commands specify byte, word and long word operation, respectively, and change the default fill size to byte, word and long word.

The **F** command executes fill using the default fill size.

ME,MEB,MEW,MEL

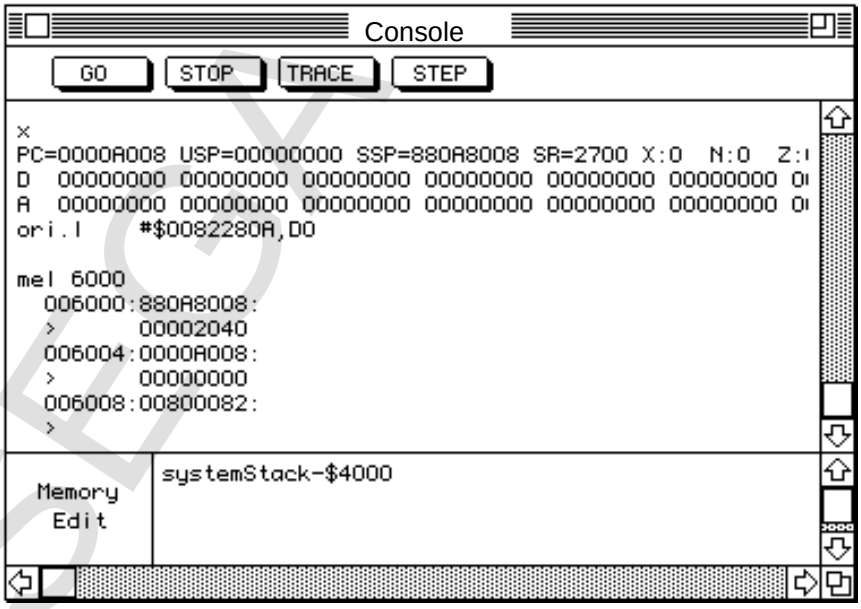
Format: **ME***expr1,expr2*
MEB*expr1,expr2*
MEW*expr1,expr2*
MEL*expr1,expr2*

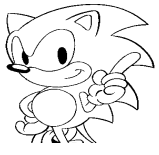
Explanation: Writes data *expr2* to address *expr1*. The **MEB**, **MEW** and **MEL** commands write in bytes, words and long words, respectively, and change the default write size to byte, word and long word. The **ME** command executes in the default write size.

ME,MEB,MEW,MEL

Format: **ME***[expr1]*
MEB*[expr1]*
MEW*[expr1]*
MEL*[expr1]*

Explanation: Performs interactive memory editing from address *expr1*. The **MEB**, **MEW** and **MEL** commands write in bytes, words and long words, respectively, and change the default write size to byte, word and long word.
The **ME** command executes in the default write size.
During interactive editing, a prompt like that shown next is displayed in the input area of the console window.





- Enter "/" or "." to terminate interactive editing.
- Return to the previous address by entering "^" or "-".
- By entering a space, advance to the next address without making any changes to the contents of the current address.

MS,MSB,MSW,MSL

Format: **MS***range,expr*
MSB*range,expr*
MSW*range,expr*
MSL*range,expr*

Explanation: Searches for the data *expr* in the address range *range*. The **MSB**, **MSW** and **MSL** commands specify byte, word and long word search, respectively, and change the default search size to byte, word and long word.
The **MS** command is executed in the default search size.

MM

Format: **MM***range,expr*

Explanation: Copies the contents of the address range *range* to after address *expr*.
• Since the **MM** command transfers the contents of the memory on the target board to the host and performs a replacement operation at the host, operation slows down if the transfer origin and the transfer destination overlap.

MC

Format: **MC***range,expr*

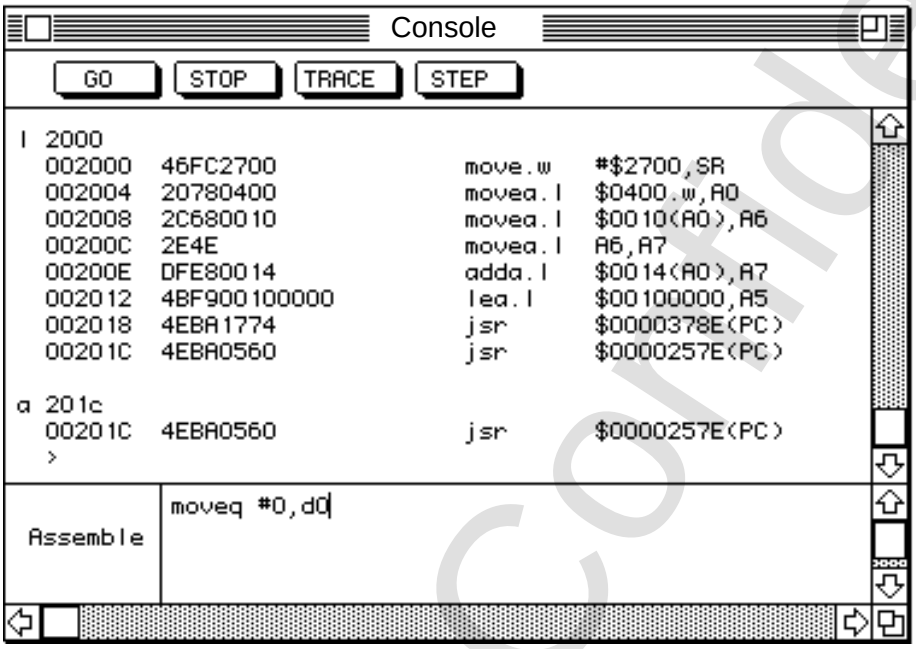
Explanation: Compares the contents of the address range *range* with the contents after address *expr*.

Assemble

A

Format: **A***[expr]*

Explanation: Performs line assemble from address *expr*. The mnemonic and addressing mode notation used follows that used by Motorola, but please note that constant notation and overflow in expression evaluation are ignored. When *expr* is omitted, the address following that used by the previous A command is used. During interactive line assembly, a prompt like that below is displayed in the input area of the console window.



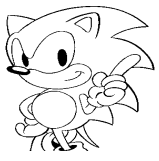
- Enter "/" or "." to terminate interactive line assembly.
- You can return to the previous address by entering "^" or "-".
- By entering a space, you can advance to the next address without making any changes to the contents of the current address.
- Specifications of absolute addresses are interpreted as described below.

  expr absolute long address
  expr.l absolute long address
  expr.w absolute short address

L

Format: **L***range*

Explanation: Disassembles and displays the address area range. When from is abbreviated, the next command where the previous L command is completed; when to is abbreviated, 'from+$1F' is adopted. If 'from' is an odd number, then it is made into an even number.



Software Breaks

A software break stops the execution of the user program by generating an exception when a specified address is rewritten to illegal command. Therefore, a software break can only be set at a location that is an even address in the DRAM area on the sound board, and is executed by the CPU as a command. SSBug has 16 of these kind of software breakpoints, and it is also possible to set the pass count.

The number of times they are passed through without stopping is referred to as the pass count, and the number of times they are actually passed through is referred to as the break count. When expressing software break numbers, specify them in an expression with values from 0 to 15.

B

Format: **B**
Explanation: Displays a list of the software breakpoint and hardware breakpoint settings.

B

Format: **B***expr1 [,expr2]*
Explanation: The software breakpoint is set at address *expr1*. The pass count is set in *expr2*. If it '0', break is continually applied. When a pass count other than '0' is specified, then break is applied continually when the break count exceeds the pass count. The values that can be set in *expr2* range from 0 to 65535, and becomes '0' if omitted. Of the software breakpoints currently canceled, the B command sets the most recent at address *expr1*.

BN

Format: **BN***expr1, expr2 [,expr3]*
Explanation: This is the same as the **B** command except that the number of the software breakpoint set is specified in *expr1*. The breakpoint set at number *expr1* is canceled up to that point. The address is specified by *expr2* and the pass count by *expr3*.

BC

Format: **BC***expr*
Explanation: Cancels the software breakpoint at number *expr*. When '*' is specified in *expr*, all software breakpoints are canceled.

BE

Format: **BE***expr*
Explanation: Enables the software breakpoint at number *expr* which has been canceled by **BC**. When '*' is specified in *expr*, all software breakpoints are enabled. When this command is executed with respect to a software break where no settings have been made since startup, the setting may occur per a meaningless address, so please use caution.

BR

Format: **BR***expr*

Explanation: Clears the break count for breakpoint number *expr*. When '*' is specified in *expr*, the break count of all software breaks is cleared.

Hardware Breaks

A hardware break monitors the generation of the specified number of bus cycles via CPU external hardware, and stops execution of the user program by providing an NMI to the CPU when the target bus cycles occur. The following bus cycle conditions can be set.

- Address bus
- Function codes
- Read/write
- Access data width
- Access count

Up to three hardware breaks can be set, and are called channel 0, channel 1 and channel 2 respectively.

Be careful a break does not come after the bus cycles are generated or after access is terminated. For example, in the case of write access, write is already terminated at the time of the break.

Also note that the MC68000 performs pre-fetch of program code and *movem* command data. Especially when a hardware break is used for program execution, the break may be applied before execution or a break may be generated by commands (immediately after *jmp*, etc.) which are not executed.

HB

Format: **HB**

Explanation: Displays a list of the settings of software breaks and hardware breaks.

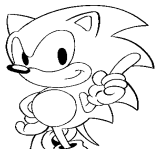
HB

Format: **HB***bpnum,adr[,mask[,cnt[,<fc>[,<ds>[,<rw>]]]]]*

Explanation: Sets a break point in the hardware break channel *bpnum*.

adrr—address
mask—address mask (described below)
cnt—break count
<*fc*>—function code
<*ds*>—access data width
<*rw*>—read/write

bpnum is 0, 1 or 2.
adr is the address in MC68000



mask is explained below. It's omission value is 0.
A value from 1 to 65535 can be specified for *cnt*. It's omission value is 1.
One of the following character strings is specified for <*fc*>.

user	(user mode access)
super	(supervisor mode access)
data	(data access)
prog	(program access)
ud	(user data access)
up	(user program access)
sd	(supervisor data access)
sp	(supervisor program access)

is set to an arbitrary mode when abbreviated.

One of the following character strings is specified for <*ds*>.

low	(lower byte access, odd addresses)
high	(upper byte access, even addresses)
word	(word access)

is set to an arbitrary mode when abbreviated.

One of the following character strings is specified for <*rw*>.

read	(read access)
write	(write access)

When abbreviated, both read and write will be the target.

Mask is equivalent to the undefined bit in most ICE, etc., and the bit for which mask is 1 is ignored during address comparison. For example, if $1000 is specified for the address and $00FF is specified for the mask, the address at which the break is actually generated becomes a selected address from $1000 to $100FF.

HB

Format: **HB***bpnum,adr,[mask],[cnt],PF*
Explanation: To apply a break to a program fetch, eliminate excessive specifications by writing the character string "PF" after *cnt* as shown here.
• This can be used in both the user mode and the supervisor mode.

HB

Format: **HB***bpnum,[level],[cnt],IA*
Explanation: To apply a break to an interrupt acknowledge cycle, Write the character string "IA" after *cnt* as shown here. The interrupt level is specified in *level*. If omitted, it becomes 'Don't care'.
• Do not specify 'Don't care' or 'level 1' or 'level 7'.

HBC

Format: **HBC***bpnum*

Explanation: Cancels the hardware break in channel *bpnum*.
All channels are canceled when the character '*' is specified in *bpnum*.

HBE

Format: **HBE***bpnum*

Explanation: Enables the hardware break in channel *bpnum*.
All channels are enabled when the character '*' is specified in *bpnum*.

- The **HBE** command enables break regardless of the setting.
- The pass count for the hardware break is initialized each time user execution is started.

Access Break

Access break sets whether or not read or write is allowed per every 4 Kbytes with respect to the entire CPU area, and detects pass cycles in violation of these settings via external hardware monitoring. As with hardware breaks, be careful that the pass cycles are terminated when a break is generated.

MAP

Format: **MAP**

Explanation: Displays a list of access break settings.

MAP

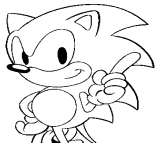
Format: **MAP***range, acc*

Explanation: Sets the access break condition *acc* in the address range *range*.
acc specifies one of the following character strings.

RW	read/write is enabled
RO	read only, break is generated with write access
NO	read is disabled, break is generated with a selected access

If the range does not have 4 Kbyte boundaries, then it is rounded up.

- In normal use, also set the following areas as read/write areas in addition to the area used by the user program.
  $600000-$67FFFF (ROM mounted area)
  $800000-$82BFFF (emulator-mounted area)
  $A00000-$A00FFF (MB89352A-mounted area)
  $E00000-$E07FFF (debugger area)
  $F00000-$FFFFFF (area used by sound tool only when sound tools exists together)
- These settings are written in the accompanying startup file SSBug Startup. Please use them as reference.



Bus Trace

Bus trace records the bus cycles using external hardware. The sound board can record up to 8192 bus cycles. The signals recorded are as follows.

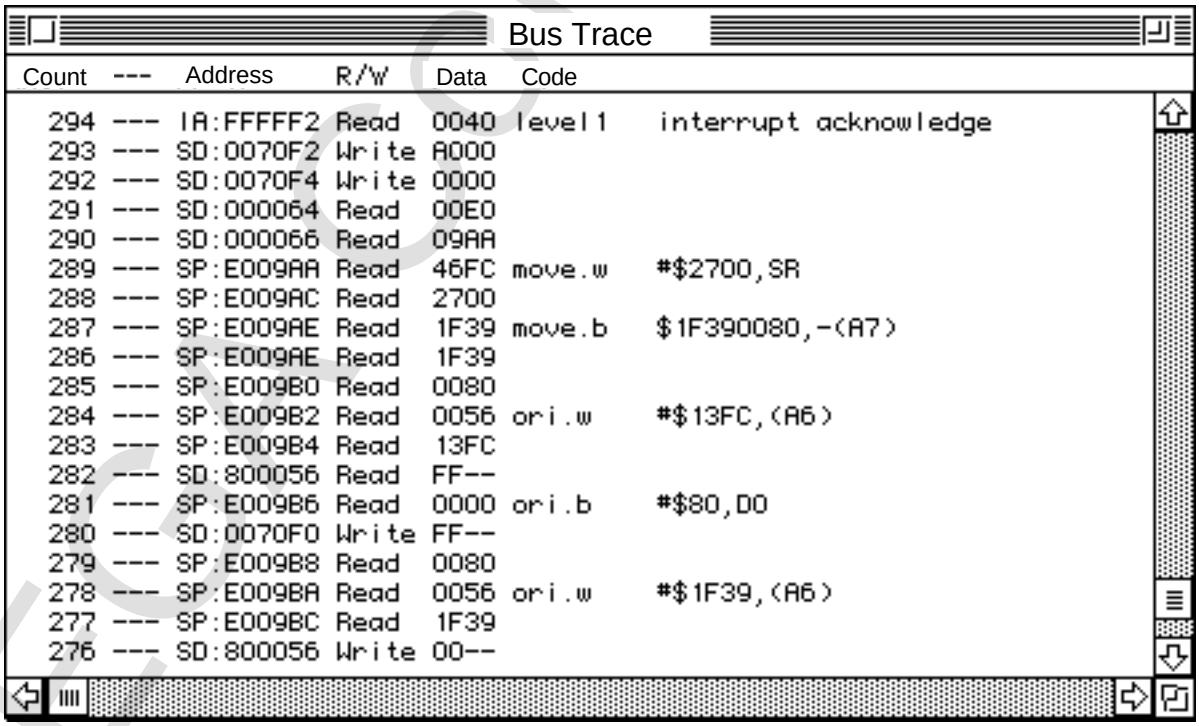
- Address bus
- Data bus
- FC
- R/W
- MCCS (B-BUS select signal)

BH

Format: **BH**expr

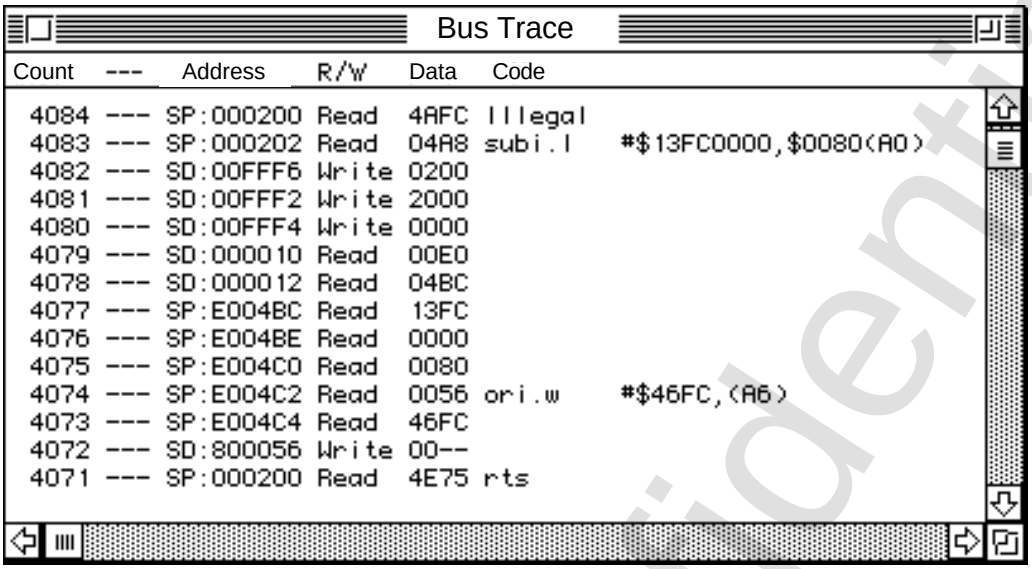
Explanation: Displays the past "expr" times worth of history since the previous stop. Due to the MC68000 command prefetch, disassembly display of this command is not necessarily correct. Specifically, the instruction after a conditional branch (instruction) is likely to be displayed incorrectly. Also, according to the hardware specification, the operation of the debugger could be recorded.

In the following examples, the SCSI communication condition of the debugger is recorded.



The contents from a SCSI interrupt is received at count #294 until the bus trace is stopped at #276 are recorded.

In the following example, a software break is set at $200. Since it has not reached the pass count, user execution is not stopped and is continued.



In this example, from 4084 to 4072 becomes the extra debugger operation that is recorded. In the interval between 4072 and 4071, the debugger code is executed but is not recorded. Also, there is a command prefetch that was not executed at 4083, therefore the disassembly display from 4077 is shifted.

File Operation

A working directory is introduced as a concept to operate files via console commands. The working directory is similar to the current directory in MS-DOS, etc., and settings are changed by selecting “Working Directory” from the file menu.

The file used by console commands include:

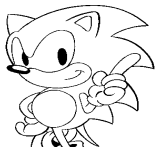
- the file when the full path name is specified
- the path specification in relation to the working directory when a partial path name is specified.

For example, if the working directory is “:Macintosh HD:Development:SSBug Folder”, then the command “r main.s28” would download the file “Macintosh HD:Development:SSBug Folder:main.s28”. SSBug only handles the data forks of files, and all files handled, including binary data, are ‘TEXT’.

PWD

Format: **PWD**
Explanation: Displays the current working directory.

- SSBug remembers the working directory when the previous session is terminated, and this is automatically set at startup.



R

Format: **R***[filename[,expr]]*

Explanation: Reads the file *filename* to offset address *expr*. If *expr* is omitted, 0 is used. The file formats Motorola S19, S28 and S37, Intelex Hex format, 2500 A.D. symbol file, ZAX symbol file and SDSS symbol file are automatically distinguished. Files that cannot be read are treated as binary files. In the case of symbol files, *expr* is added to the symbol value. In the case of S28 and other files with an address specification, *expr* is added to the specified address. In the case of binary files, the top address for read is used.

• In all of the file operation commands below, the same operation as selection from the menu is performed when the file name is omitted.

RB

Format: **RB***[filename[,expr]]*

Explanation: Treats the selected file *filename* as a binary file and reads it to address *expr*. If *expr* is omitted, 0 is used.

V

Format: **V***[filename,[expr]]*

Explanation: Compares the file *filename* with the contents of memory or the contents of symbol. The file formats Motorola S19, S28 and S37, Intelex Hex format and symbol files are automatically distinguished. Files that cannot be read are treated as binary files. In the case of symbol files, *expr* is added to the symbol value. In the case of S28 and other files with an address specification, *expr* is added to the specified address.

• When verifying symbol files, value comparison will be against the same-name symbols among those already loaded and those in the file.

VB

Format: **VB***[filename[,expr]]*

Explanation: Treats the selected file *filename* as a binary file and compares it with the contents of memory. If *expr* is omitted, 0 is used.

W

Format: **W***[filename,range[,offset]]*

Explanation: Writes the contents of the address range *range* to the file *filename* in the Motorola S28 format. *offset* is added to the address value of the S record. '0' is used when omitted.

WB

Format: **WB***[filename,range]*
Explanation: Writes the contents of the address range *range* to the file *filename* in binary format.

WS

Format: **WS***[filename]*
Explanation: Writes the currently loaded symbol to the file *filename*.

Symbol Operation

SR

Format: **SR***[regexp]*
Explanation: Displays a symbol matching *regexp*. When *regexp* is omitted, the character string "*" is used.

SV

Format: **SV***[regexp]*
Explanation: Treats the value of symbols matching *regexp* as signed integers, sorts in ascending order, and displays them. If *regexp* is omitted, then the character string "*" is used.

SU

Format: **SU***[regexp]*
Explanation: Treats the value of symbols matching *regexp* as unsigned integers, sorts in ascending order, and displays them. If *regexp* is omitted, then the character string "*" is used.

SN

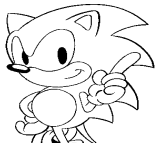
Format: **SN***[regexp]*
Explanation: Sorts symbols matching *regexp* in ascending order of the names and displays them. Upper case and lower case are sorted in dictionary order. If *regexp* is omitted, the character string "*" is used.

SC

Format: **SC***[regexp]*
Explanation: Displays the number of the symbol matching *regexp*. If *regexp* is omitted, the character string "*" is used.

SS

Format: **SS***expr*
Explanation: Searches for a symbol whose value is *expr* and displays it.



SD

Format: **SD***regexp]*
Explanation: Deletes a symbol matching *regexp*.

SA

Format: **SA***name,expr*
Explanation: Records the symbol whose symbol name is *name* using the value *expr*. If the same name is already recorded, the value is updated.

• The maximum number that can be sorted by the entire symbol display commands is 16,384. No more than 16,384 symbols can be displayed by "commands with sort," and only the SR command can be used when the number of symbols is more than 16,384.

Macro Operation

Macro's make it possible for the user to define a selected number of statements as one command. Macro execution is performed by using the macro name as a command name as with other built-in commands. When an argument is desired, a "special character" is used in the macro definition for reference. It is not necessary to describe a dummy argument.

MACRO

Format: **MACRO***name*
Explanation: Sets the macro with name *name*. Defining is terminated by inputting a blank line.

EXITM

Format: **EXITM***expr*
Explanation: This command can only be executed during macro development. If *expr* is '0', the macro being executed is exited.

MLIST

Format: **MLIST***[regexp]*
Explanation: Displays macro names matching *regexp* and their contents. Displayed after being sorted in macro name dictionary order. If *regexp* is omitted, the character string "*" is used.

• The same restrictions that apply to symbol operation commands apply to sort display.

WM

Format: **WM***filename*
Explanation: Writes the currently registered macro to file *filename*.

RM

Format: **RM***filename*
Explanation: Reads a macro definition from the file *filename*.

• Description and expansion example of argument
The *n*th macro argument is noted as "*n*" in the macro definition, where *n* is an integer from 0 to 9.

•Macro definition

MACRO	wordwrite
mew	'0,'1

•Command input

wordwrite	100,123

•Expansion result

mew	100,123

Batch Jobs

Batch job is the reading and execution of a file in which a selected number of statements are described. These files are called batch files.

B A

Format: **BA***filename*
Explanation: Executes the batch file *filename*.

The commands **RM** and **BA** are actually the same. Macros can be defined in a batch file and other statements can also be executed in a macro file.

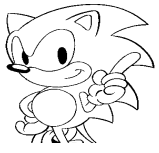
Startup Files

Startup files refer to batch files automatically executed when SSBug is started. When the text file "SSBug Startup" is in the working directory when SSBug is started up, this is treated as a startup file and executed.

Other

HELP

Format: **HELP**
Explanation: Opens the help window.



VERSION

Format: **VERSION**

Explanation: Displays the version of SSBug itself, the target program and the SCSP.

MEMFREE

Format: **MEMFREE**

Explanation: Displays the amount of memory that can be used for macros, symbols, batches, etc.

ECHO

Format: **ECHO***[arg]*

Explanation: **ECHO ON**
Echoes the input line from the current input destination (macro, console, or batch) to the console. This becomes a local setting for each nest of the macro, console or batch.
ECHO OFF
Switches echo off.
ECHO
Echoes other specified character strings to the console.

QUIT

Format: **QUIT**

Explanation: Terminates SSBug.

?

Format: **?**

Explanation: Displays a list of user variables.

?

Format: **?***expr*

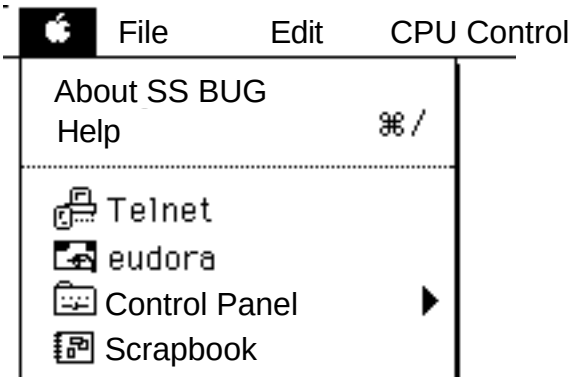
Explanation: Evaluates the expression *expr* and displays the result.

4.0 Menu Environment

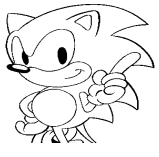
SSBug can also be operated via the standard Macintosh user interface.
The same expressions as in the console environment can be used at all locations that accept numerical input in dialogs.

Apple Menu

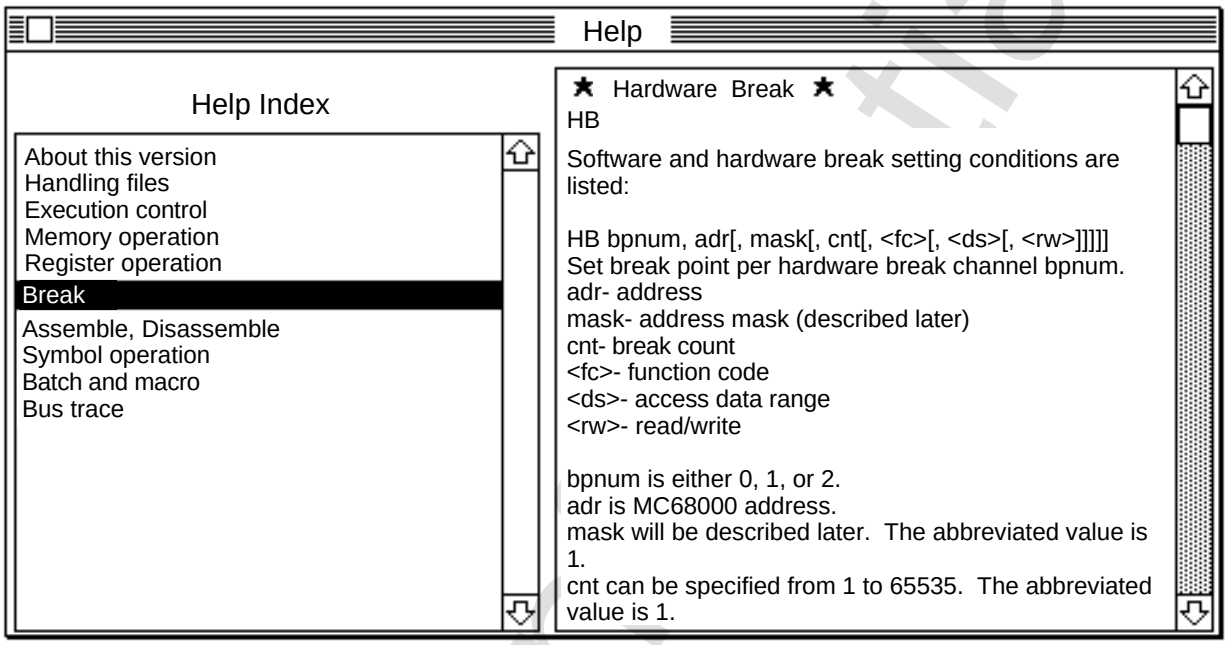
There are two items in the Apple menu.



"**About SSBug...**" displays the version and other information about SSBug. The window is closed by clicking in the window.

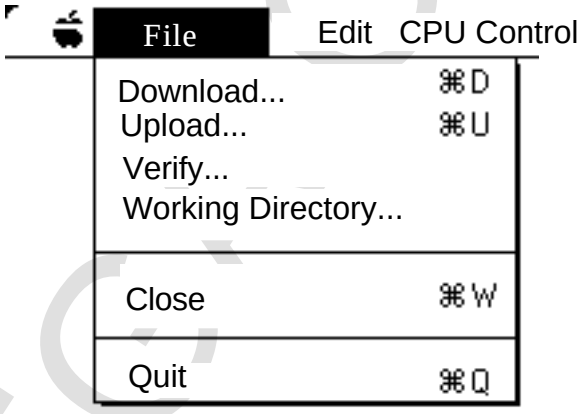


"**Help**" opens the help window shown below. Click on the subject to see in the "Help Index" in the left part of the window. A description of that subject will appear on the right side of the window.



File Menu

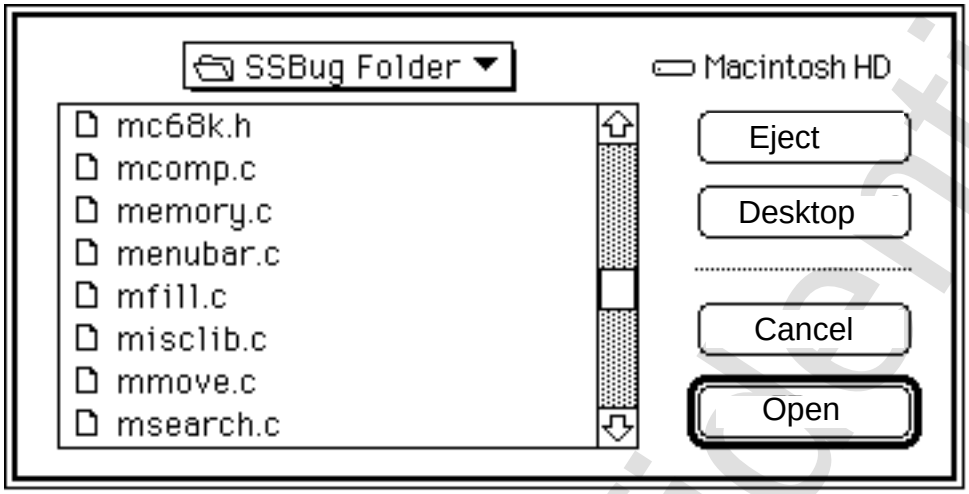
This menu deals mainly with file operations.



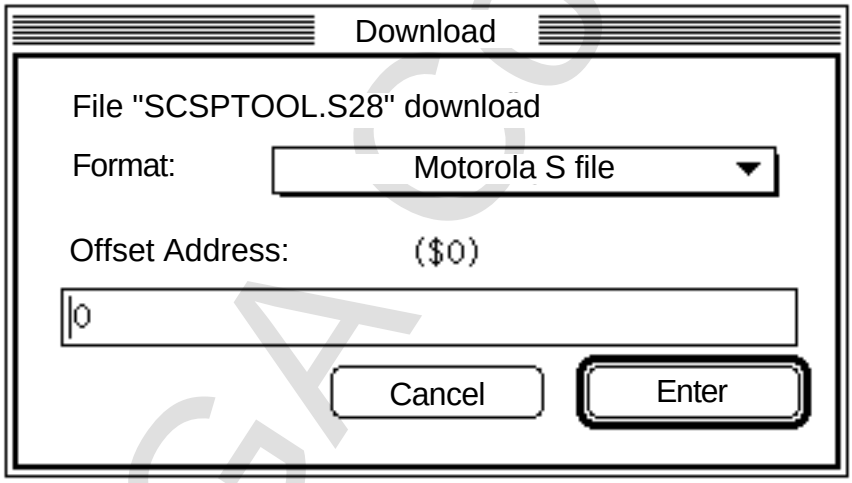
Download

Equivalent to the console commands R, **RB**, **RM** and **BA**. All file load operations for user programs, symbols, macros and batches can be performed from this menu.

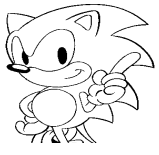
1. When "**Download**" is selected, the following dialog for selecting a file is dis played.



2. When a file is selected, the following dialog for specifying the format or offset address is displayed



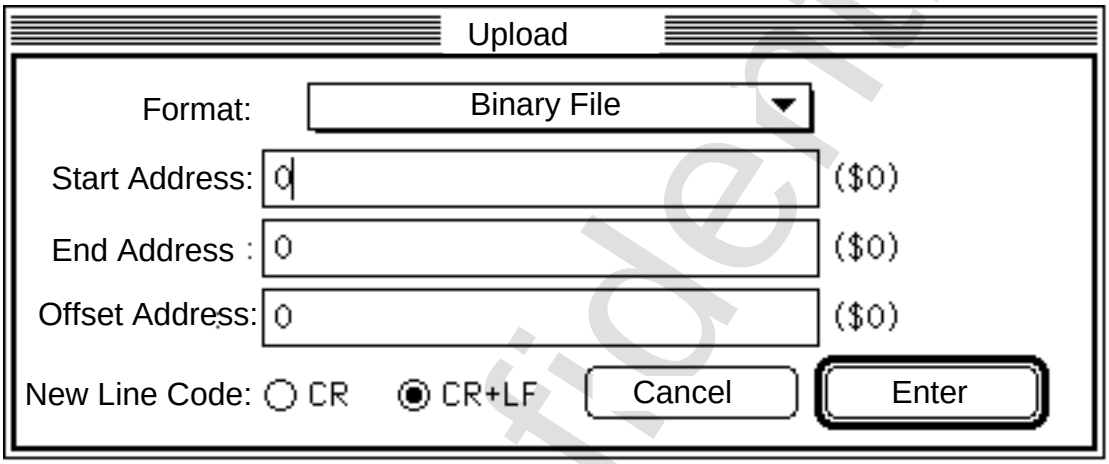
3. The format is set in a menu, and the offset address is set by inputting a formula.



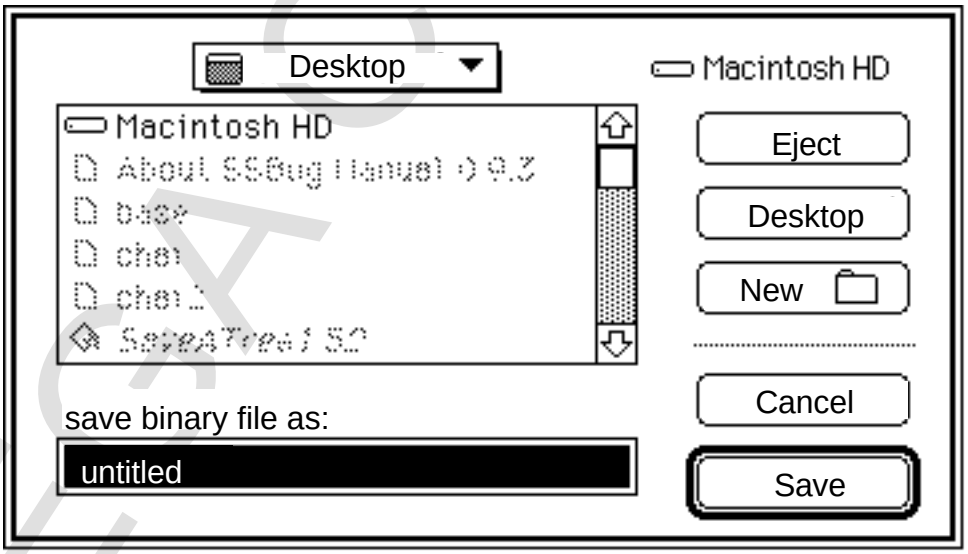
Upload

Is equivalent to the console commands **W**, **WB**, **WM** and **WS**. All file save operations for user programs, symbols, macros and batches can be performed from this menu.

1. When "**Upload**" is selected, the following dialog for specifying the file to be saved is displayed.

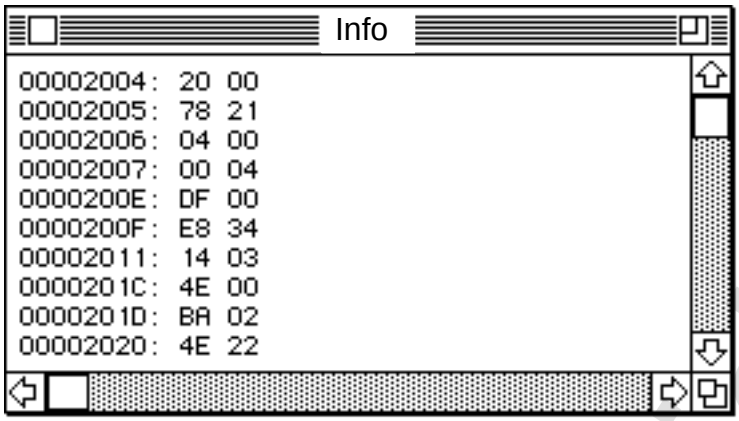


2. The format is set in a menu, and the offset address is set by inputting a formula. The new line designation is only valid when the output is text. The following dialog for inputting a file name is displayed next.



Verify

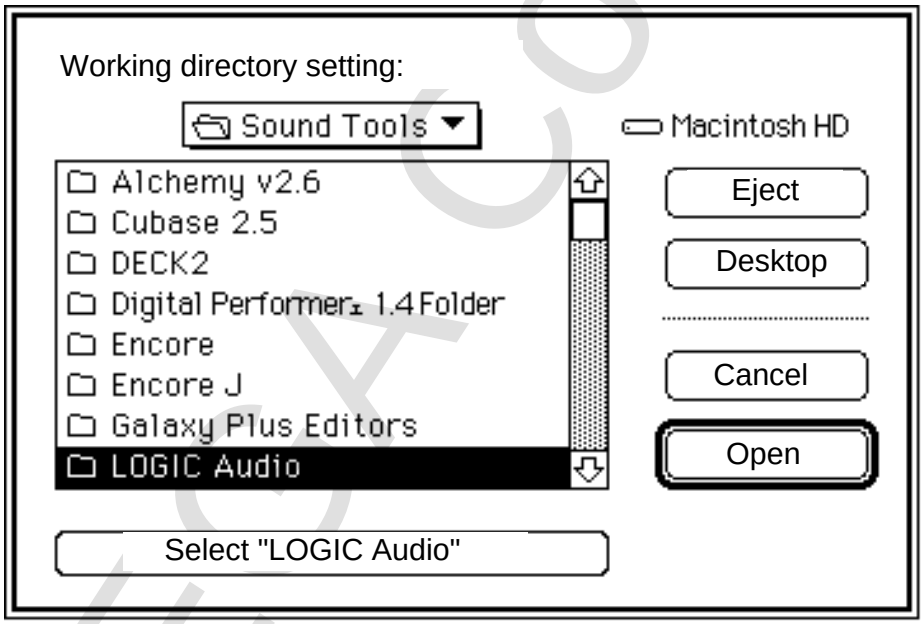
Verify is equivalent to the console commands **V** and **VB**. Operation is similar to download, but the result of verify is displayed in an information window like that below.



- If there are tool many results, then the older information is lost from the window.

Working Directory

When you select **Working Directory**, the following dialog will appear. Please select using the button at the bottom of the dialog box.

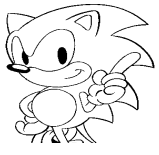


Close

Closes the front (active) window.

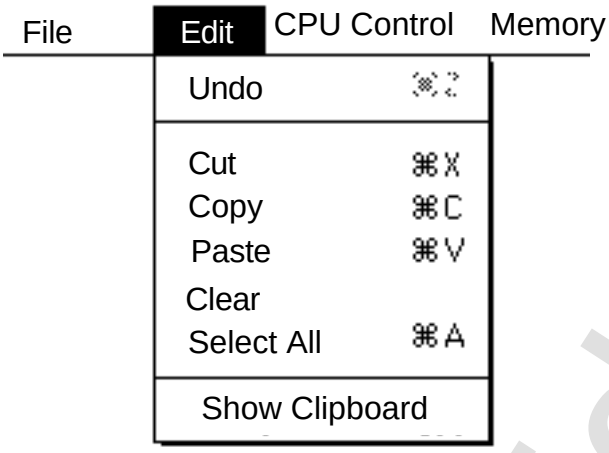
Quit

Quits SSBug.



Edit Menu

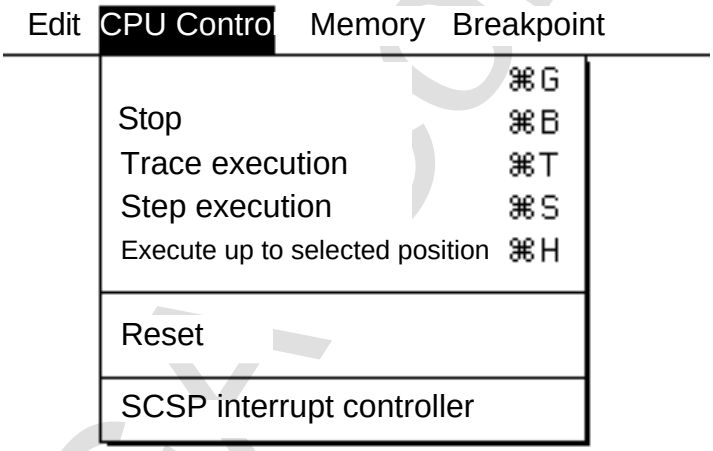
The **Edit** menu is used primarily to cut and paste text in accordance with the standard Macintosh interface.



• Undo is not supported.

CPU Control Menu

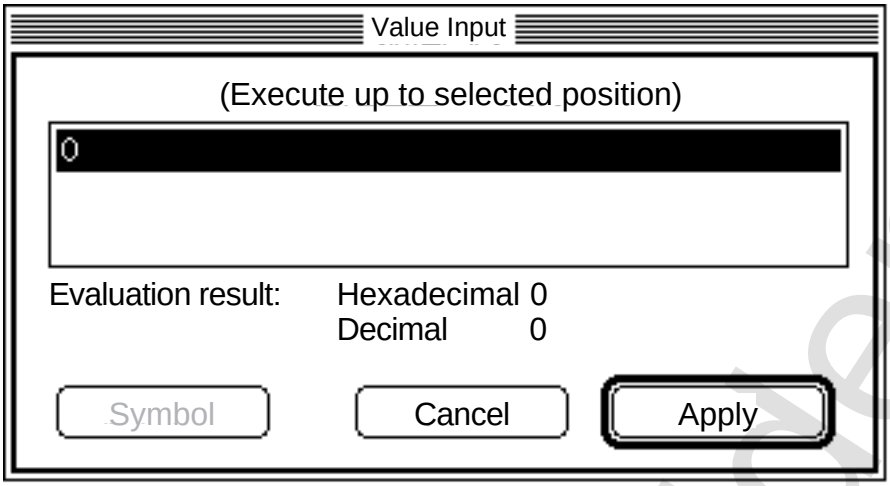
This menu is used to control execution of user programs.



Except for "SCSP Interrupt Controller", the following commands correspond to the console commands as shown below.

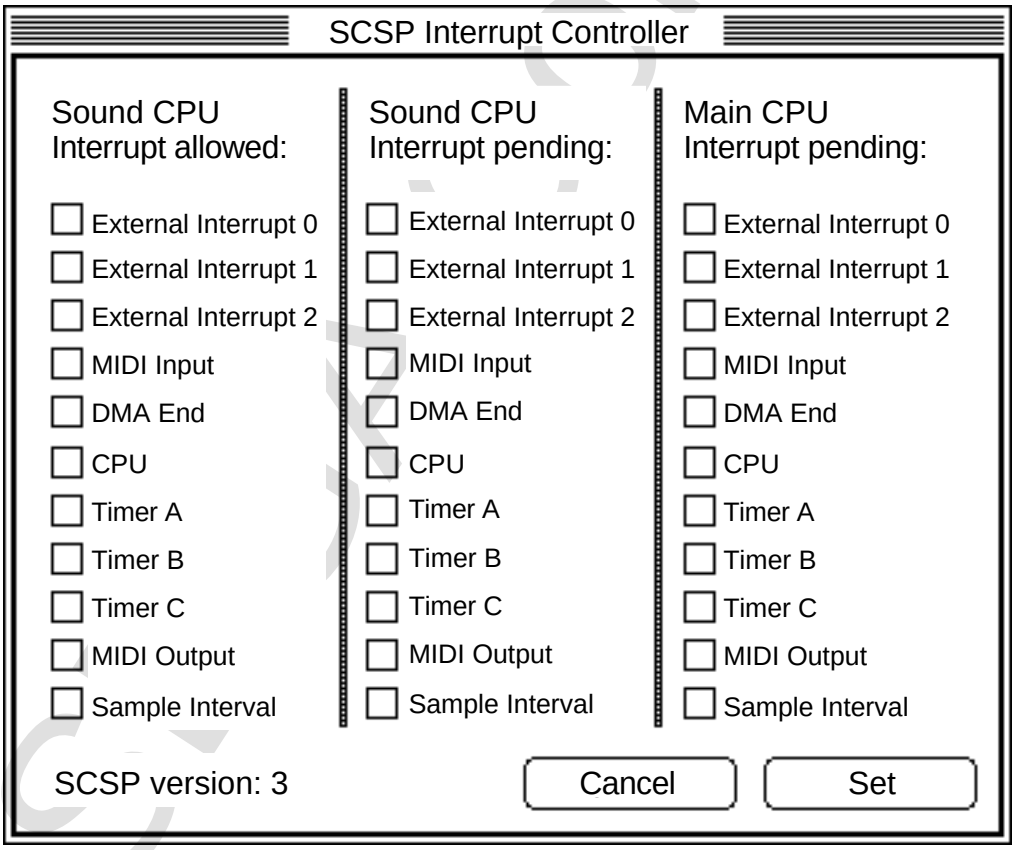
Run	G
Stop	STOP
Trace	T
Step	S
Run to Specified Point	G ,expr
Reset	RESET

The following dialog for specifying an address is displayed for **Run to Specified Point**.

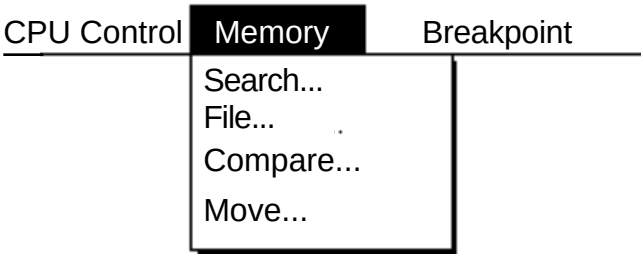


SCSP Interrupt Controller

Allows the operation of the interrupt enable register (SCIEB) and the interrupt pending registers (SCIPD, MCIPD) of the SCSP.

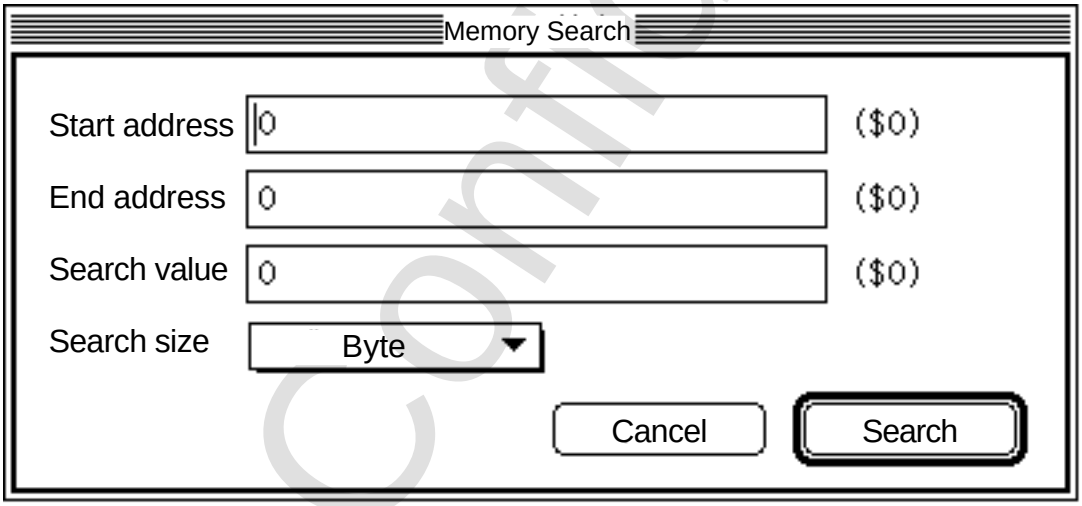


Memory Menu

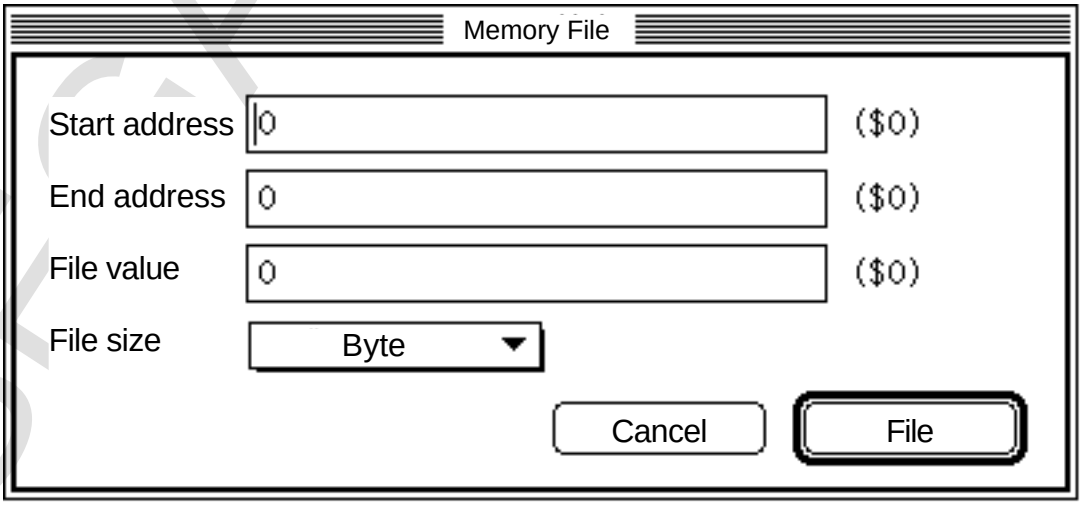


These correspond to the console commands as shown below. Through each selection, a movable modal type dialog for inputting arguments is displayed.

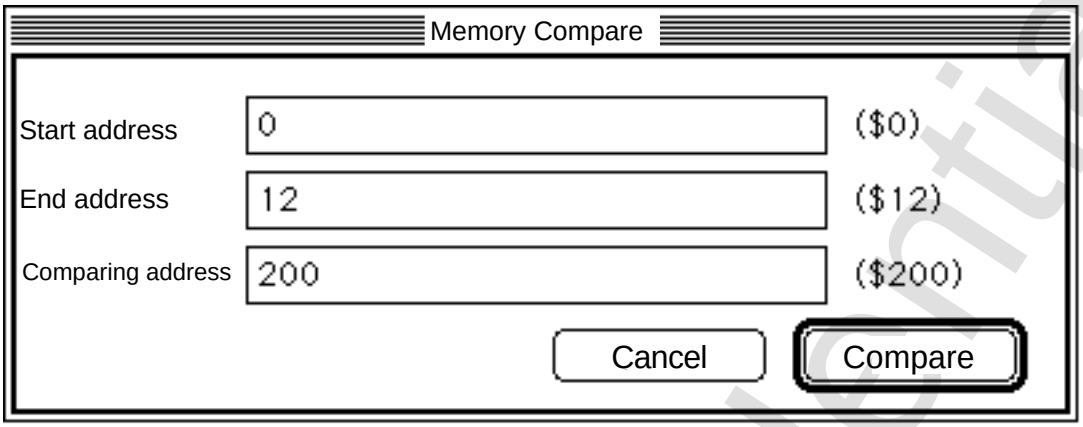
- **Retrieve** Corresponds to **MS** command. However, the results are displayed in an information window.



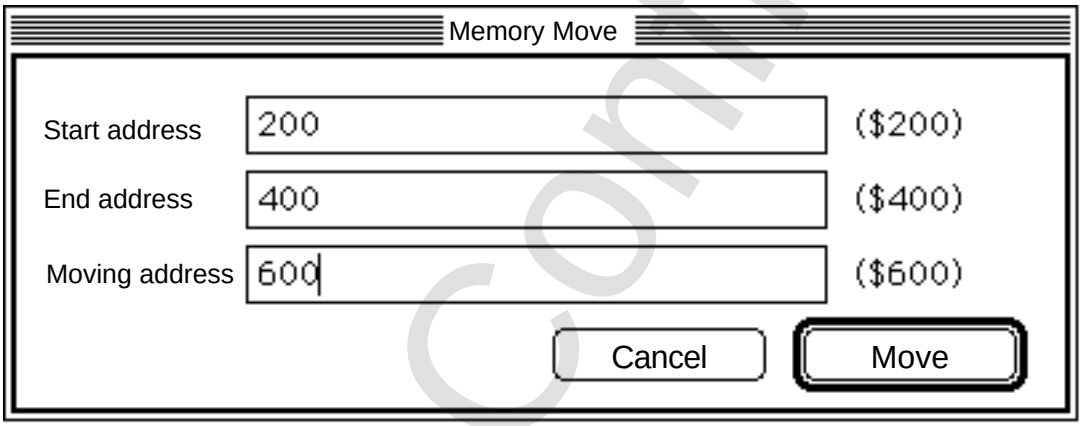
- **Fill** Corresponds to **F** command.

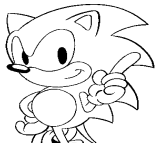


• **Compare** Corresponds to **MC** command. However, the results are displayed in an information window.

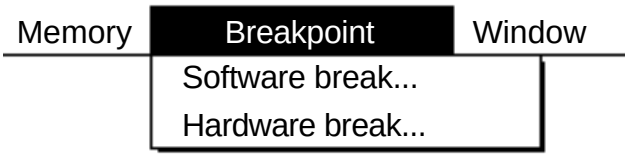


• **Move** Corresponds to **MM** command.

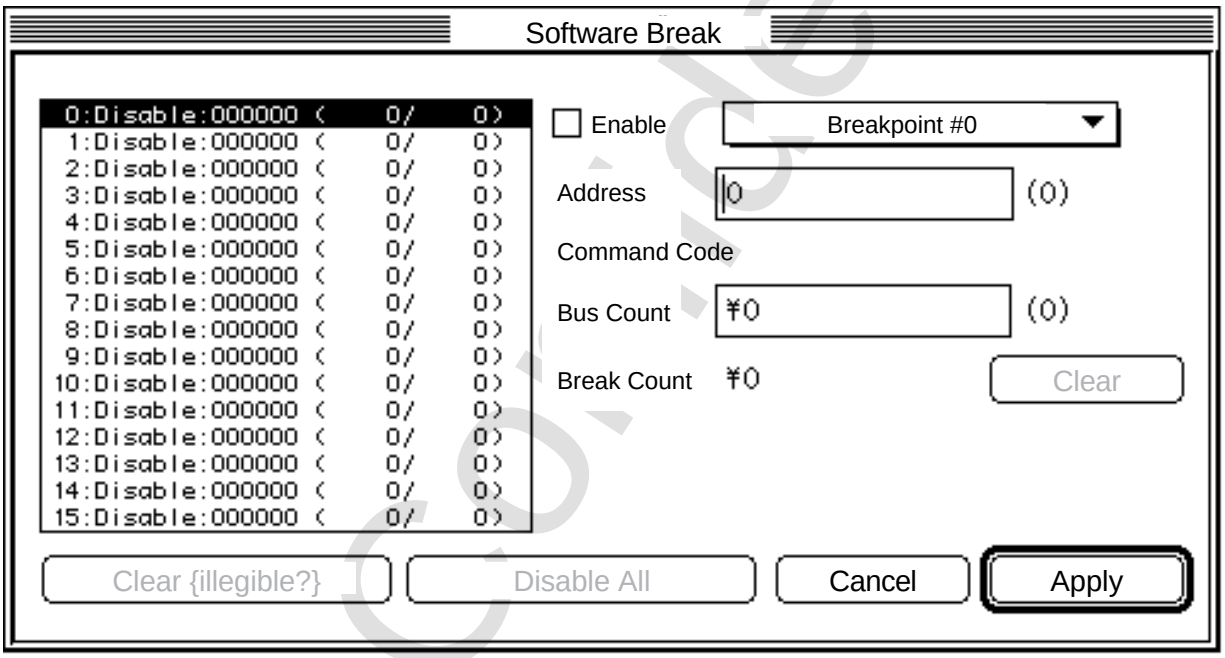




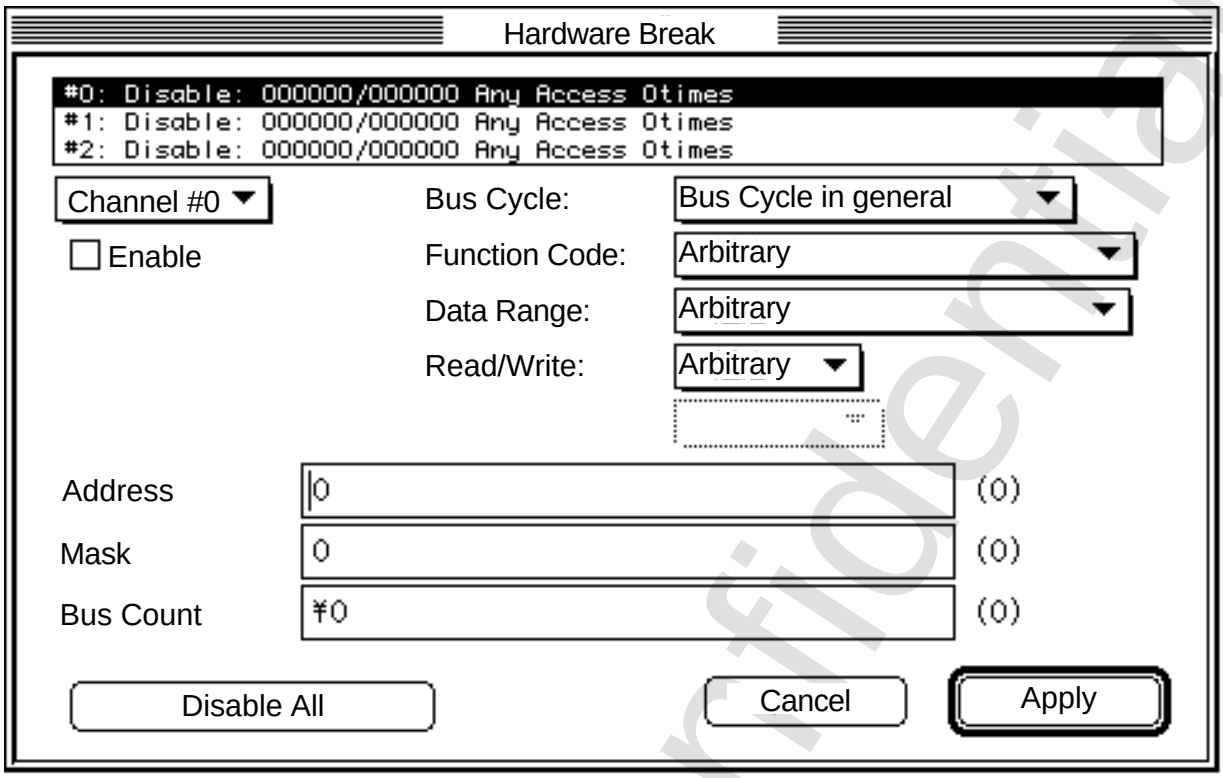
Breakpoint Menu

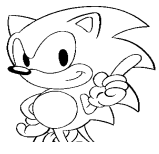


When selecting **Software Break,** the following movable modal dialog box will be displayed: The same operations as can be performed by the console commands **B**, **BE**, **BC** and **BR** can be performed.

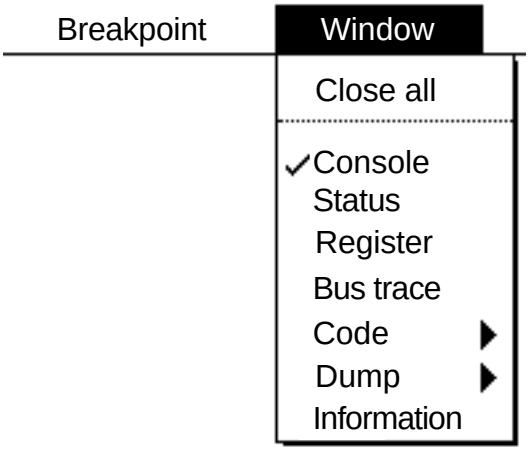


When selecting **Hardware Break,** the following movable modal dialog box will be displayed: The same operations as can be performed by the console commands **HB**, **HBE** and **HBC** can be performed.





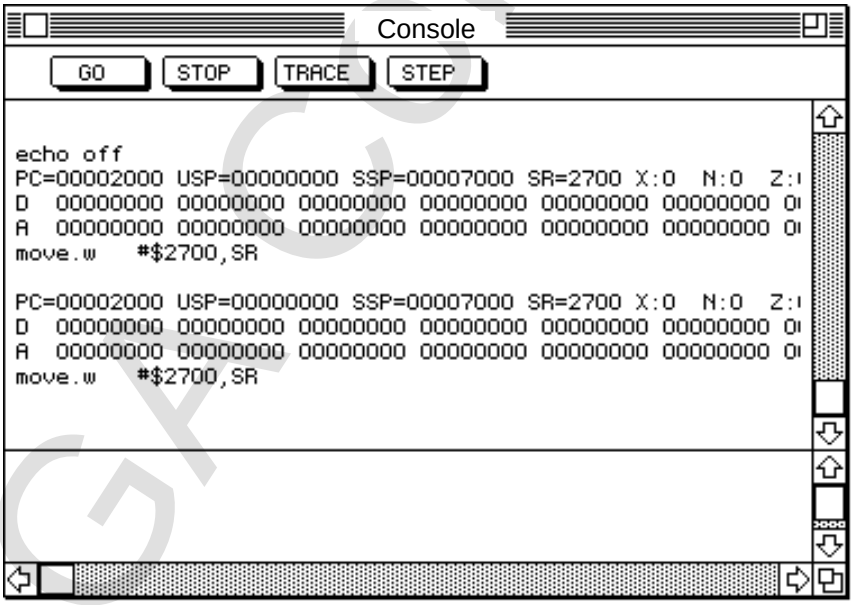
Window Menu



By selecting **Close All,** all open windows will be closed. Other menu items will open their corresponding windows. Individual windows cannot be closed from the window menu.

Console Window

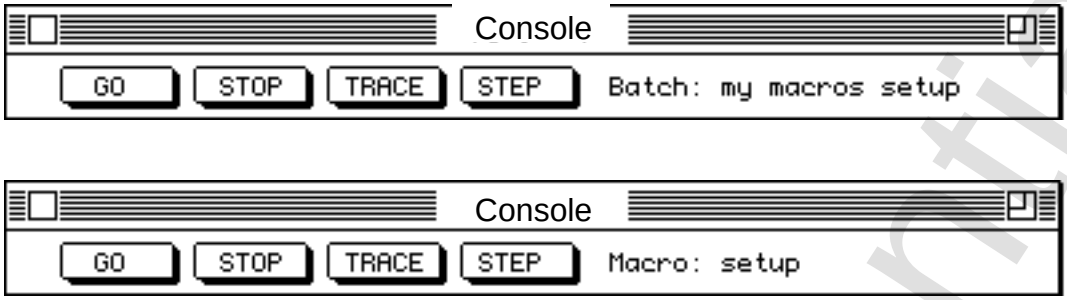
The following window is used to enter console commands.



The four buttons at the top of the window have a one-to-one correspondence with the console commands as shown below, and when a button is clicked, the respective console command is entered.

GO button	**G**
STOP button	**STOP**
TRACE button	**T**
STEP button	**S**

Further, if the command is not entered from the keyboard, i.e., by execution of a batch job or a macro, the file name or macro name is displayed as shown below.

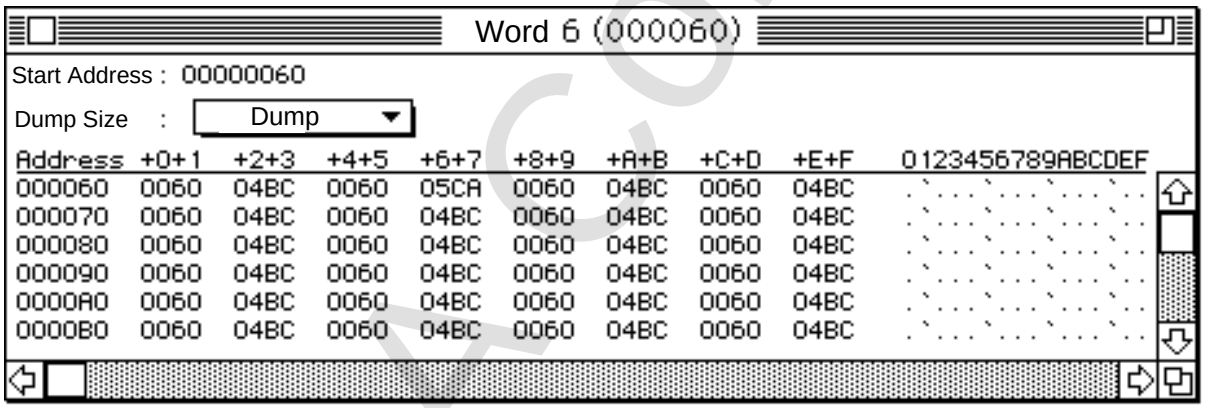


The bottom part of the window is the input area. This is normal edited text, so cut and paste can be used freely. The result of execution of console commands is displayed in the area between the input area and the buttons at the top. A backlog of results can be viewed by using the scroll bar.

- If the backlog becomes too full, the older information is lost.

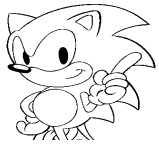
Dump Window

The contents of memory are displayed in this window. There is a total of eight dump windows.



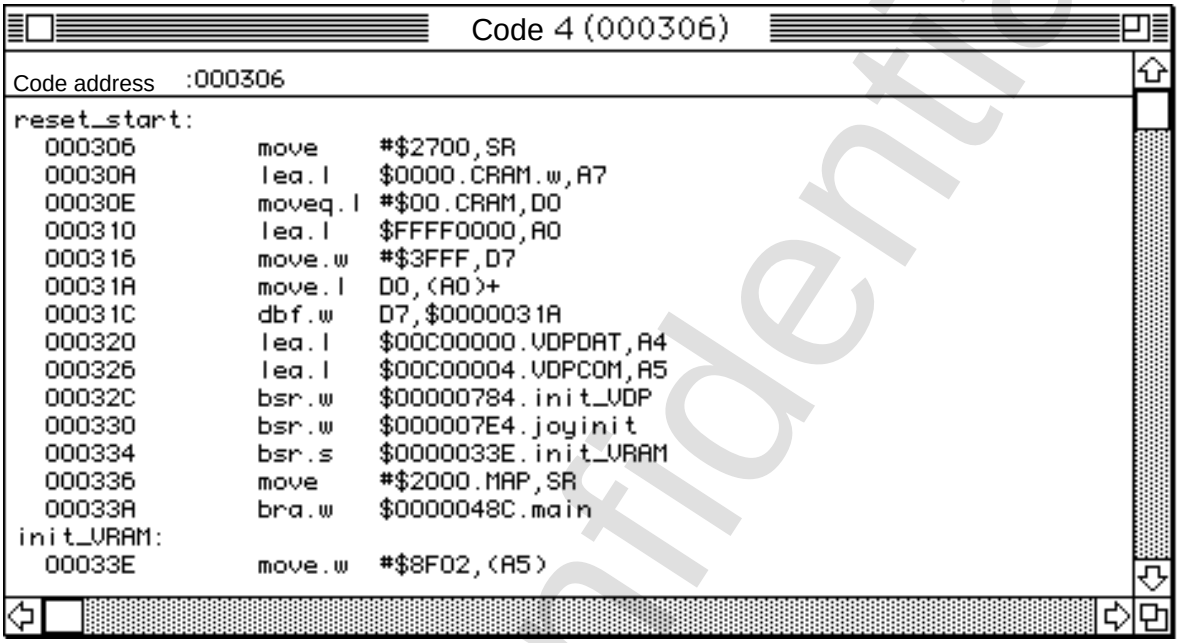
The dump start address can be input by clicking on **Dump Address**.
The display size can be selected (in byte, word or long word) from the **Dump Size** menu.
Also, by clicking on the dumped number in the window, the memory can be edited.

- The dump window is not updated during execution of a user program. Updating is performed when the user program is stopped by a break, etc.



Code Window

The contents of reverse assembly are displayed in this window. There are a total of eight code windows.

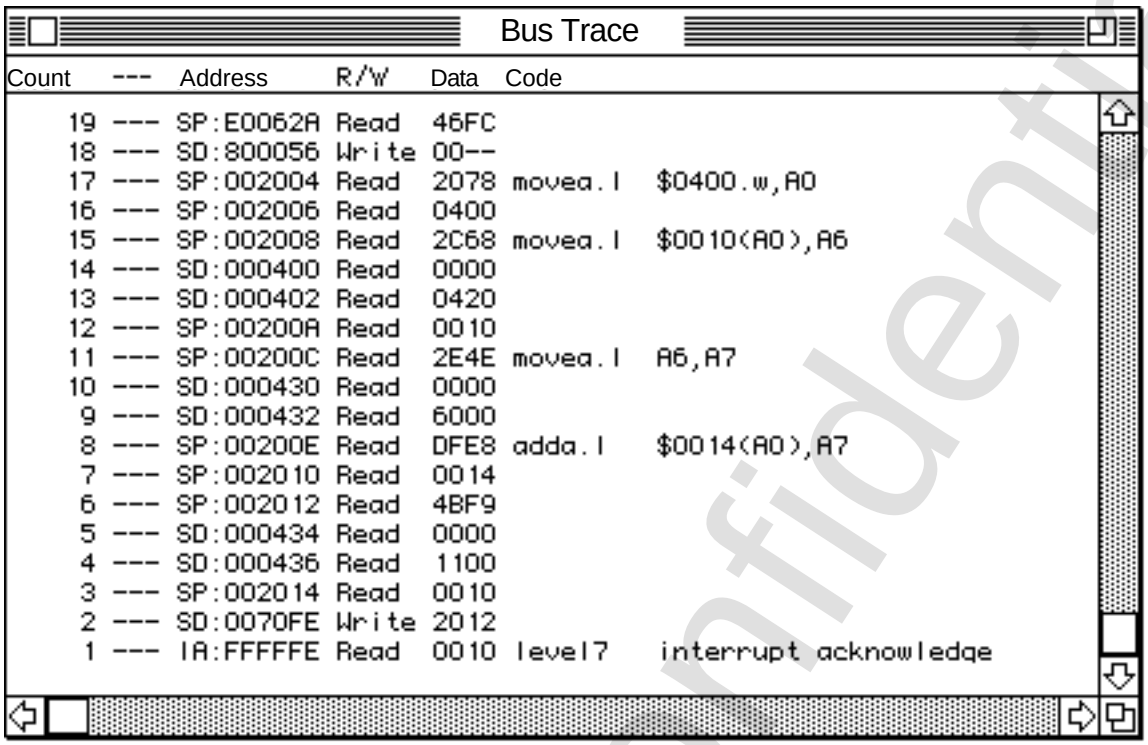


The reverse assembly start address can be input by clicking on **Code Address**. Also, line assembly can be performed by clicking on the reverse assembly display in the window.

- The dump window is not updated during execution of a user program. Updating is performed when the user program is stopped by a break, etc.

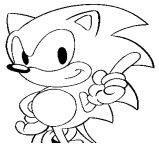
Bus Trace Window

Bus trace results are displayed in this window.



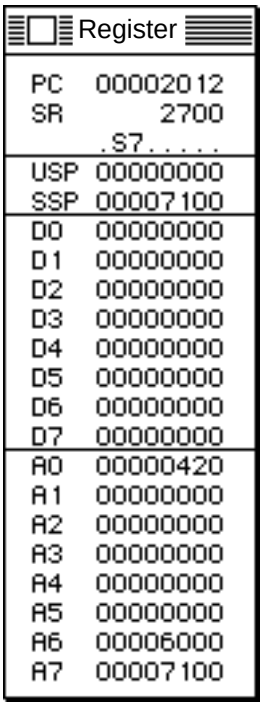
The bus cycles are distinguished as explained below when displaying on a color display.

Black Bus cycle for the MC68EC000
Red Bus cycle in which the EXT signal is asserted
Green Bus cycle in which control is completely passed to the debugger
Blue Bus cycle from B-BUS

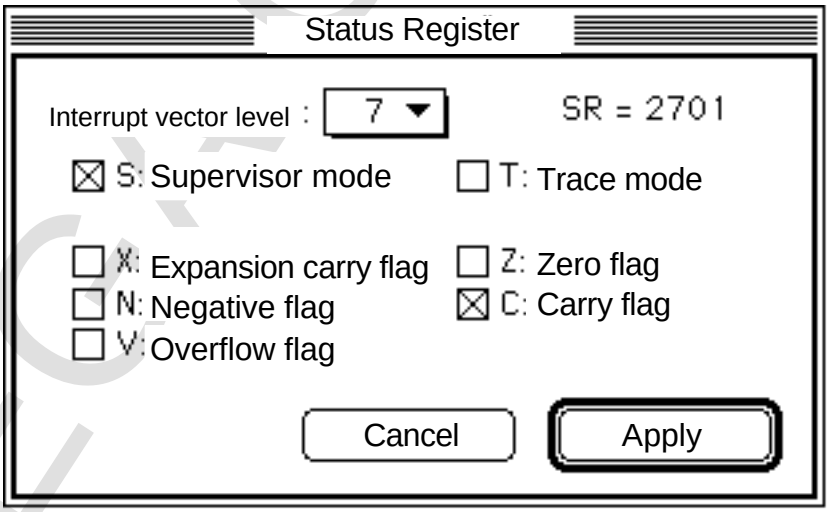


Register Window

Displays the contents of the CPU registers.

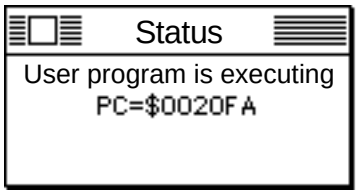


The register values can be input or changed by clicking on the respective register values. Also, by clicking on the flag indicator of the SR register, the value of the SR register can be changed in bit units as shown below.

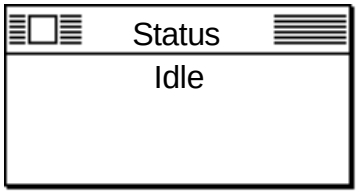


Status Window

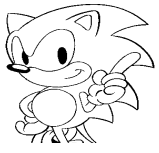
Displays execution of the user program. The following display appears during execution of a
user program.



When the user program is stopped, the following display appears.



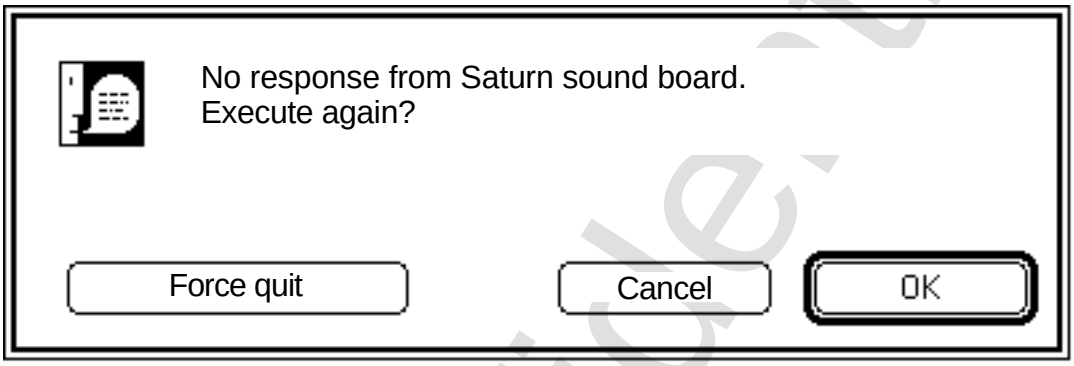
However, when a break is not applied without reaching the pass count with software breaks, an idle display or debugger program PC display may temporarily appear even when the user program is normally being executed.



5.0 Use Restrictions

Problems with SCSI Interrupts

Communication between SSBug and the sound board is done via the SCSI (level 1 auto vector). Therefore, when the sound board is at interrupt level one for long periods, the message (in the dialog box) “No response” is displayed.



If the sound board should hang up at this time, the SCSI bus is taken over and stopped, thus hanging up the Macintosh as well. This happens when display of the message (in the dialog box) “No response” is attempted, the resource cannot be read in because the SCSI bus has been taken over, thus resulting in a window being displayed with nothing in it and hanging up the system.

If this should happen, attempt execution again after releasing the SCSI bus by resetting the sound board or supplying a break from the main side.

Vectors Hooked by SSBug

The exception vectors (re-written at the time of execution of the user program) used by SSBug are listed below.

reset
bus error
address error
illegal instruction
zero divide
chk instruction
trapv instruction
privilege violation
trace
un-initialized interrupt
spurious interrupt
level 1 auto vector
level 7 auto vector

The operation of programs that automatically rewrite the above vectors or reference them during execution cannot be guaranteed under SSBug. Programs that hold the vector table static present no problem.

In a condition in which control has shifted to the debugger and not the user program, the two interrupts level 2 auto vector and level 3 auto vector are replaced by the following routine.

```
interrupt:  move.w #-1,SCSP_SCIRE
            rte
```

Therefore, as long as the sound tool rules (use only level 2 and 3 SCSP interrupts) are observed, interrupts will not be applied to the user program during debugger standby.

Memory Hidden by SSBug

Since SSBug hooks to the user vector, the first $80 bytes of memory is given special treatment, and the memory $000000-$00007F as viewed from the debugger is not actually the target $000000-$00007F. Therefore, the following restrictions apply to the $000000-$00007F memory area.

- When an ICE, etc., is also used, seemingly conflicting conditions may be experienced.
- User program code cannot be placed.
- Even if memory is dumped from SSBug, an exception vector onto which the aforementioned SSBug hooks cannot be viewed.

Critical Time Periods

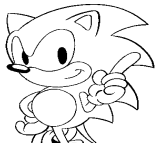
There are critical time periods in the operation of the current version of the debugger when an emulator break, etc., of a hardware break, etc., and a CPU exception, including software breaks or trace executions, occur almost simultaneously. This is currently being addressed.

SCSI Noise

There are cases in which the SCSI signal of SSBug appears as noise in the analog output of the sound board. If this should be a problem, open up a dialog box. SSBug does not communicate with the sound board during dialog display.

Carriage Returns

The carriage return characters in text files used by SSBug can by either a CR (Macintosh) or a CR+LF (MS-DOS) in all batch, symbol or hex files. However, the EOF character used in CP/M or MS-DOS is not supported.



Symbol Files

The SDSS symbol format is defined as follows in SSBug.

[<WS>]<symbol value (hex)><WS><symbol name><WS>, an arbitrary number of repeat + new character.

However, <WS>: an arbitrary number of white spaces, new line character: CR or CR+LF.

Therefore, by selecting the SDSS format, the following symbol files can also be used.

- Iwasaki Giken's IR80 or other CP/M class symbol files
- Computex company's ID68000 symbol files

Restrictions with Respect to Saturn

The TAS command cannot be used with the Saturn and development board MC68EC000. Also, external devices cannot be reset with the RESET command.