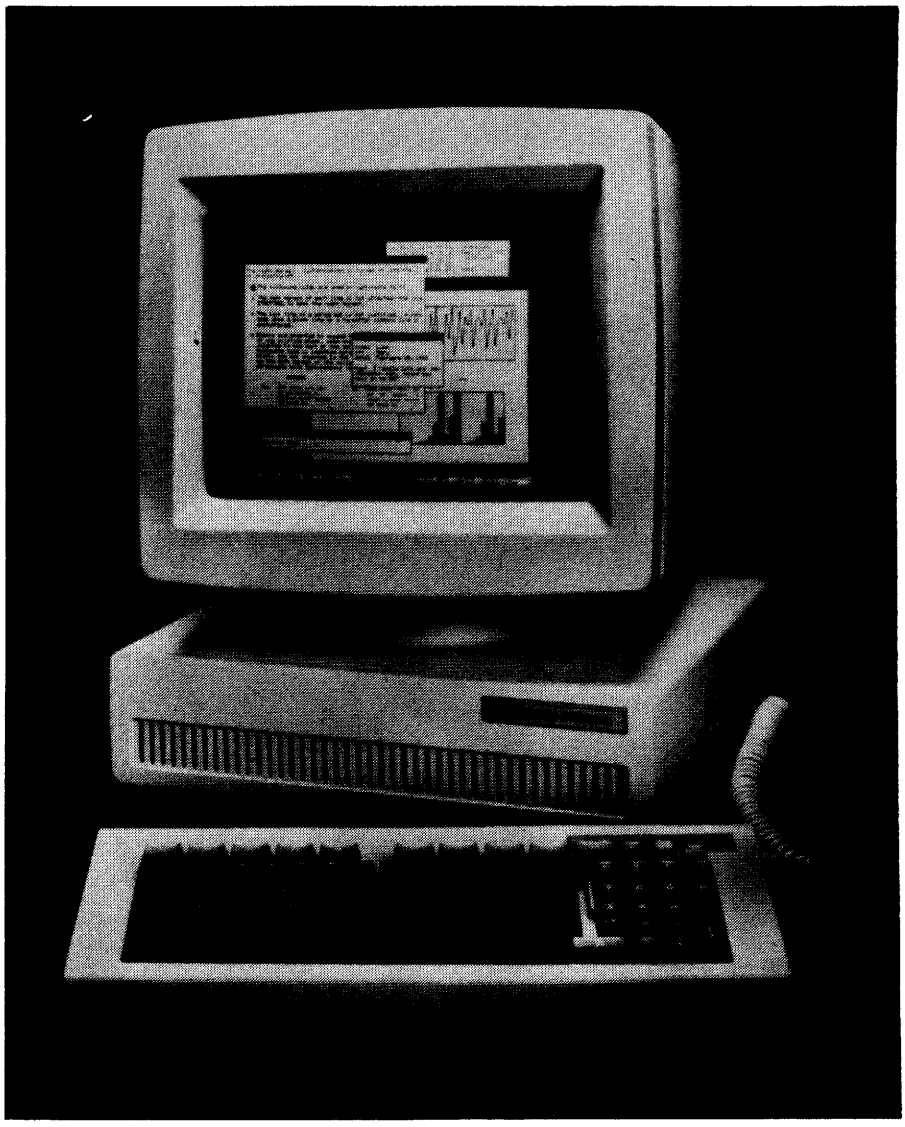
MACSbug
68000 Debugger
User's Manual

Corvus Concept™



CORVUS SYSTEMS

DISCLAIMER OF ALL WARRANTIES & LIABILITIES

Corvus Systems, Inc. makes no warranties, either expressed or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Corvus Systems, Inc. software is sold or licensed "as is." The entire risk as to its quality or performance is with the buyer and not Corvus Systems, Inc., its distributor, or its retailer. The buyer assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Corvus Systems, Inc. be liable for direct, indirect, incidental or consequential damages, even if Corvus Systems, Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liabilities for incidental or consequential damages, so the above limitation may not apply to you.

Every effort has been made to insure that this manual accurately documents the operation and servicing of Corvus products. However, due to the ongoing modification and update of the software along with future products, Corvus Systems, Inc. cannot guarantee the accuracy of printed material after the date of publication, nor can Corvus Systems, Inc. accept responsibility for errors or omissions.

NOTICE

Corvus Systems, Inc. reserves the right to make changes in the product described in this manual at any time without notice. Revised manuals and update sheets will be published as needed and may be purchased by writing to:

Corvus Systems, Inc.
2029 O'Toole Avenue
San Jose, CA 95131
Telephone: (408) 946-7700
TWX 910-338-0226

FCC WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accorance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to correct the interference. The following measures may help: reorient the receiving antenna, plug the computer in a different branch circuit, or relocate the computer.

CORVUS SYSTEMS

CORVUS CONCEPT MACSbug 68000 DEBUGGER USER'S GUIDE

PART NO. : 7100-01387

DOCUMENT NO. : CCC/60-33/1.0b

RELEASE DATE : November, 1982

CORVUS CONCEPT

MACSbug 68000 DEBUGGER

USER'S MANUAL

MACSbug

INSTALLATION AND OPERATING INSTRUCTIONS

1.1 INTRODUCTION

This document describes the Corvus Concept MACSbug Debugger Version 2.0. It includes a description of the commands for the resident firmware monitor, MACSbug, and examples of its use.

1.2 INSTALLATION PROCEDURES

NOTE: Before powering the base unit ON or OFF, ensure that there is no diskette in the floppy drive.

a) Power-off the Concept base and display.

b) Disconnect the keyboard cable and display monitor cable. Open the drawer of the base unit and remove the power supply cables connected at locations labeled J8 and J1 on the processor board and the memory board respectively. Remove any tap cables or interface cards which are currently in the drawer.

c) Lift up on the drawer assembly and completely remove it from the base unit.

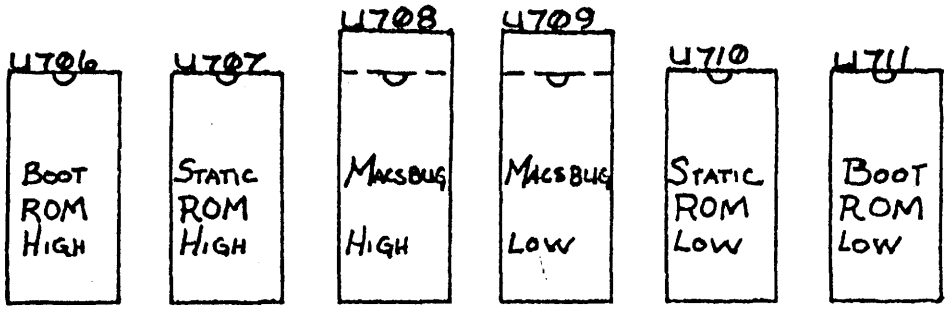
d) The procedure to install MACSbug ROMs is different for REV 03 processor boards and REV 04 processor boards. You can determine whether you have a REV 03 or REV 04 by the configuration of the Concept boot switches.

   On the REV 03 processor boards, there is a 2-switch microswitch on the right side of the processor board, opposite the I/O slots.

   On the REV 04 processor boards, there is a 8-switch microswitch on the right side of the processor board, opposite the I/O slots.

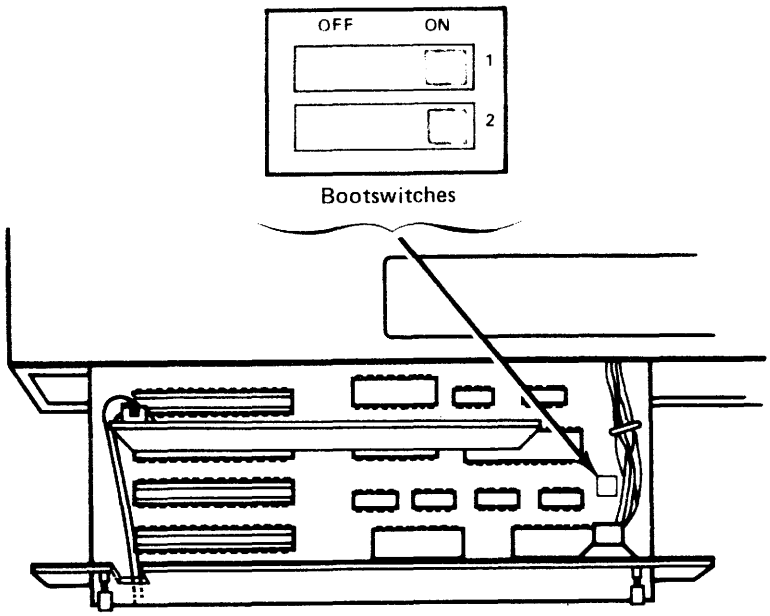
e) Revion 03 Installation Procedures

1. Locate the Boot ROMS on the processor board at locations U706 (ROM 0U) and U711 (ROM 0L). If they are not version 0.5 or later, remove the ROMs at these locations and place the ROM labeled CC 0.5 H or later in location U706 and place the ROM labeled CC 0.5 L or later in location U711 on the processor board.



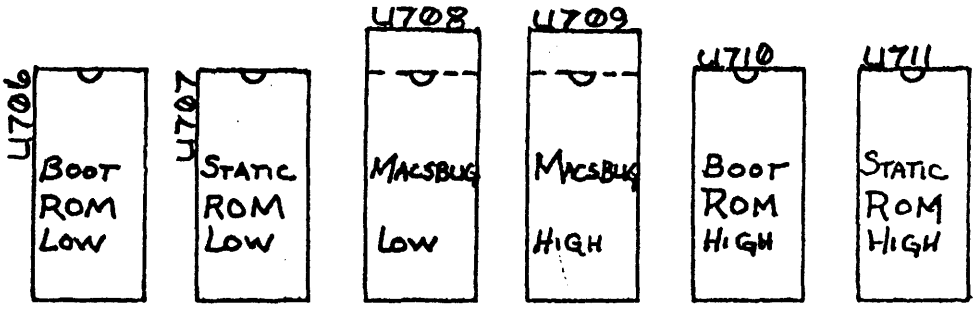
2. Place the ROM labeled MACSbug 2.0 L in location U709 and place the ROM labeled MACSbug 2.0 H in location U708 on the processor board. The MACSbug ROM sockets are 28 pin sockets, and the MACSbug ROMs are 24 pin devices. The sockets should have the top four pin locations unused (i.e. pins 1,2,27 and 28).

3. Place both microswitches in the ON position.



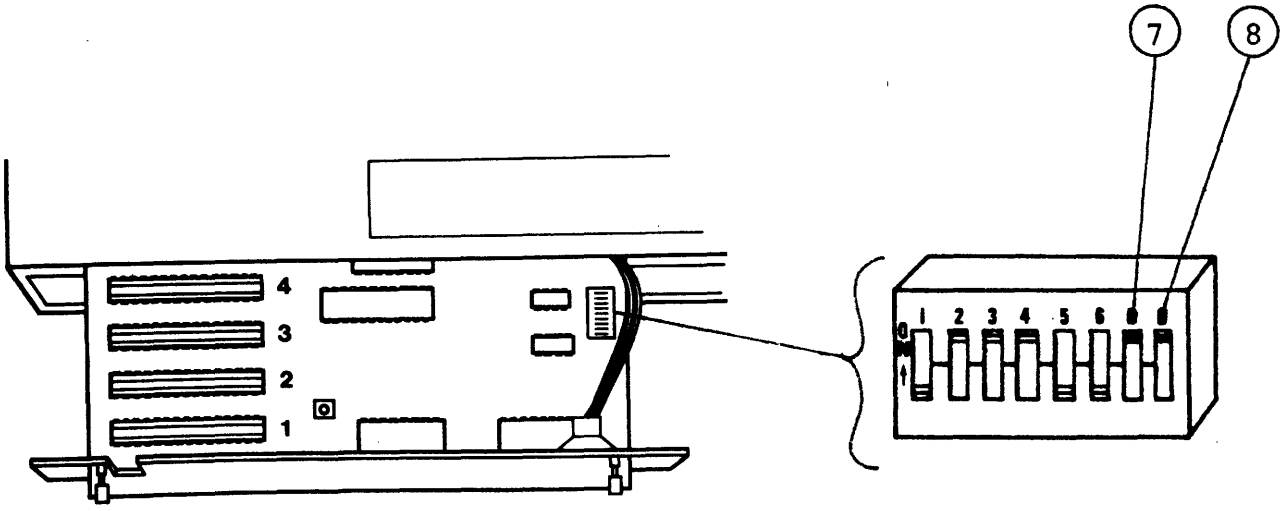
f) Revison 04 Installation Procedures

1. Locate the Boot ROMS on the processor board at locations U706 (ROM 0L) and U710 (ROM 0U). If they are not version 0.5 or later, remove the ROMs at these locations and place the ROM labeled CC 0.5 H or later in location U710 and place the ROM labeled CC 0.5 L or later in location U706 on the processor board.



2. Place the ROM labeled MACSbug 2.0 L in location U708 and place the ROM labeled MACSbug 2.0 H in location U709 on the processor board. The MACSbug ROM sockets are 28 pin sockets, and the MACSbug ROMs are 24 pin devices. The sockets should have the top four pin locations unused (i.e. pins 1,2,27 and 28).

3. Place microswitches 7 and 8 in the ON position.



g) Replace the drawer into the base unit and position the two power supply cables along the speaker tray channel to prevent chaffing of the cables. Reconnect the power supply cables to J8 on the processor board and J1 on the memory board.

h) Reconnect any tap cables or interface cards originally within the drawer.

i) Power on the display and then the base unit. The Concept will emit a beep, and then request input from the user regarding the boot device as follows:

Select the device : (D,F,L,O) :

D - Debug (MACSbug)
F - Floppy Disk Drive
L - Local Disk Drive
O - Omninet Drive

j) Select your normal disk boot option to run a quick check of the unit.

k) If the unit does not complete the boot, check the ROM locations and that all pins of the ROMs are installed correctly. Repeat the procedure until the system boots. If problems persist, contact your local servicing dealer or Corvus Customer Service.

1.3 COMMMUNICATING WITH MACSbug

Communication with MACSbug is performed through the two serial ports on the back of the Corvus Concept. When used with MACSbug, port 1 has a default data rate of 9600 BAUD, parity is disabled and an 8 bit character size is assumed. An ASCII terminal must be attached to port 1 with a null modem cable. This terminal is the MACSbug console.

MACSbug supports port 2 as a standard RS-232C data terminal connector with a default data rate of 4800 BAUD, parity disabled and a 8 bit data character size. Port 2 can be used to communicate with a host computer, a printer or other serial device.

This two port communication arrangement allows the Corvus Concept to be placed in series with an ASCII terminal and a host computer. The transparant mode in MACSbug can be used to bypass the Corvus Concept. This allows a program to be created on the host computer using the ASCII terminal and then when the program code file is generated, it can be downloaded into the Corvus Concept for execution and dubugging. This can all be performed without reconfiguring the cabling.

1.4 OPERATIONAL PROCEDURE

After the MACSbug ROMs has been installed, MACSbug can be entered before the Corvus Concept operating system is booted as follows:

a. Connect an ASCII terminal to port 1 of the Corvus Concept.

b. Ensure that the Concept boot switches are both in the ON position.

c. Power on the Corvus Concept.

d. Select option D, for Debugger, when prompted.

MACSbug will initialize and display on the ASCII terminal connected to port 1 with the following message:

```
MACSBUG 2.0
*
```

If these two lines do not print out, perform the following:

a. Check to see that the ASCII terminal is attached to RS-232C port 1 using a null modem cable.

b. Ensure that the terminal's BAUD rate is set to 9600, parity is disabled and an 8 bit character size is selected.

1.5 COMMAND LINE FORMAT

Commands are entered the same as in most other buffer organized computer systems. A standard input routine controls the system while the user types a line of input. The delete (RUBOUT) key or control H will delete the last character entered. A control X will cancel the entire line. Control D will redisplay the line. Processing begins only after the carriage return has been entered.

The format of the command line is:

*COmmand parameters ;options

where:

*	is the prompt from the monitor. The user does not enter this. In the examples given, the lines beginning with this character are lines where the user entered a command.
CO	is the necessary input for the command. Each command has one or two upper case letters necessary in its syntax. In the examples, the entire command may be used, but only those letters in upper case in the syntax definition are necessary. In actual usage, MACSbug converts all lower case characters to upper case.
mmand	is the unnecessary part of the command. It is given in the syntax definiton only to improve readability. If this part of the command was actually entered on the command line, it would be ignored.
parameters	depends upon the particular command. Data is usually in hex but most printable ASCII characters may be entered if enclosed in single quotes. The system also supports a limited symbolic feature allowing symbols to be used interchangeably with data values.
;options	modifies the nature of the command. A typical option might be to disregard the checksum while downloading.

1.6 MACSbug COMMAND SUMMARY

COMMAND	DESCRIPTION	SECTION
reg#	Print a register	1.6.1
reg# hexdata	Put a hex value in the register	
reg# 'ASCII'	Put hex-equivalent characters in register	
reg#:	Print the old value and request new value	
class	Print all registers of a class (A or D)	
class:	Sequence through-print old value request new	
DM start end	Display memory, hex-ASCII memory dump	1.6.2
SM address data	Set memory with data	
OPen address	Open memory for read/change	1.6.3
SYmbol NAME value	Define and print symbols	1.6.4
W#	Print the effective address of the window	1.6.5
W#.len EA	Define window length and addressing mode	
M# data	Memory in window, same syntax as register	
Go	Start running from address in program counter	1.6.6
Go address	Start running from this address	
Go TILL add	Set temporary breakpoint and start running	
BReakpoint	Print all breakpoint addresses	
BR add: count	Set a new breakpoint and optional count	
BR -address	Clear a breakpoint	
BR CLEAR	Clear all breakpoints	
TD	Print the trace display	1.6.7
TD reg#.format	Put a register in the display	
TD Clear	Take all registers out of the display	
TD ALI	Set all registers out of the display	
TD A.l D.l L.c	Set register blocks or line separator	1.6.8
T	Trace one instruction	1.6.9
T count	Trace the specified number of instructions	
T TILL Address	Trace until this address	
:*(CR)	Carriage return-trace one instruction	
OFfset address	Define the global offset	1.6.10
CV decimal	Convert decimal number to hex	1.6.11
CV $hex	Convert hex to decimal	
CV value,value	calculate offset or displacement	
REad;=test	Expect to receive S records	1.6.12
VErify;=text	Check memory against S records	
CAll address	JSR to user utility routine	1.6.13
P2	Enter transparent mode	1.6.14
*..data	Transmit command to host	

CTL-A	The control A key ends transparent mode (default)
CTL-D	The control D key redisplays the line
CTL-H	The control H key deletes the last character entered
CTL-X	The control X key cancels the entire line

68000 REGISTER MNEMONICS	DESCRIPTION
D0,D1,D2,D3,D4,D5,D6,D7	Data registers
A0,A1,A2,A3,A4.A5,A6,A7	Address registers
PC	Program counter
SR	Status register (condition codes)
SS	Supervisor stack pointer (A7 in supervisor mode)
US	User stack pointer (A7 in user mode)

COMMAND FORMATS	DESCRIPTION
reg# hexdata	Put a hex value into register 'reg#'
reg# 'ascii data'	Put hex value of ASCII into register 'reg#'
reg#:	Print register value and request in new value
reg#	Print register value
class (where class=D or A)	Print values of all registers in the class
class:	Cycle through all registers in the class printing old value and requesting new value

```
EXAMPLES                    COMMENTS

*A5 123                     Set address register A5 to hex value 123
*A5                         Command to print the value of register A5
A5=00000123                 Computer response
*D4 FFFFFF                  Set a data register
*D0:                        Command to print old value and take in new value
D0=0000000 ? 45FE           Computer prompts with old value; new value entered
*D:                         Command to cycle through all data registers
D0=000045FE ? 9EAB3         Change value of register DO from 45FE to 9EAB3

D1=00000000 ? (CR)          Carriage return (null line) means the value
                            remains the same
D2=00000000 ? (CR)
D3=00000000 ? (CR)
D4=00FFFFFF ? (CR)
D5=00000000 ? 55555         Change register D5 to a new value
D6=00000000 ? (CR)
D7=00000000 ? (CR)
*D                          Display all data registers
D0=0009EAB3 D1=00000000 D2=00000000 D3=00000000
D4=00FFFFFF D5=00055555 D6=00000000 D7=00000000
*PC:                        Display and request input for program counter
PC=0008B3 ? 2561            Set the program counter to new value
*SR 0                       Set status register to zero (user mode)
*A7 4321                    Set address register (same as US now)
*US                         Display user stack pointer
US=00004321
*SS FFC                     Set supervisor stack pointer
*SR 2000                    Set status register to supervisor mode
*A7                         Print A7 which is now the SS register
A7=00000FFC                 Initialize system stack pointer value from
*                           MACSbug
```

1.6.2 Display and Set Memory

COMMAND FORMAT	DESCRIPTION
DM start end	Display Memory in hex and ASCII where start < end
DM start count	Where start > count
DM2 start end	Send output to PORT 2
SM address data	Set Memory to hex
SM address 'ASCII'	Set Memory to ASCII
SM address data N	The 'N' as the last character means start a new line; the system will prompt with the current address

EXAMPLES — COMMENTS

```
*SM 92000 'ABC'                          Set memory to some ASCII data
*SM 92003 4445 46 'G'                    Set some more locations
*DM 92000 92010                          Command to dump memory

0092000 41 42 43 44 45 46 47 00 00 00 00 00 00 00 00 00 ABCDEFG.......
0092010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

                                         In the following usage of the DM
                                         command the second number is smaller than
                                         the first so it is decoded as a count.
*DM 92003 12
0092003 44 45 46 47 00 00 00 00 00 00 00 00 00 00 00 00 DEFG..........
0092013 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

*SM 91000 1 23 456 7890 ABCDE 12345678  Size can be up to 8 characters
*DM 91000
091000 01 23 04 56 78 90 0A BC DE 12 34 56 78 00 00 00................

*SM 91000     'TABLE      '  00005678 N      Use of the 'N' parameter to
                                               start a new line

0009100C?     'START      '  00023456


*DM 91000 20
091000 54 41 42 4C 45 20 20 20 00 00 56 78 53 54 41 52 TABLE....VxSTAR
091010 54 20 20 20 00 02 34 56 00 00 00 00 00 00 00 00 T.....4V........

*OFFSET 2030                               Global offset will be added to
*DM 91000                                  command parameters
093030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................

*SM 91005 1234 N              Global offset added to address 91005
00093037 ? AB
*DM 91000
093030 FF FF FF FF FF 12 34 AB FF FF FF FF FF FF FF FF ................

*SM 20000 AB CD EF            Trying to set ROM
ERROR                         Error message
*
```

1.6.3 Open Memory for Read/Change

COMMAND FORMAT	DESCRIPTION
OPen address	Open memory at specified address and enter subcommand mode

SUBCOMMAND FORMAT

(CR)	Go to next sequential location
^	Go to previous location
=	Stay at same location
.	Return to MACSbug(exit the OPen command)

EXAMPLES

ADDRESS	CONTENT	USER ENTERS	COMMENTS
*OP E00			Open memory location E00
000E00	= FF?	12	User enters data and system goes to next location
000E01	= AB?	(CR)	Carriage return means go to the next location
000E02	= 44?	34^	UP arrow means go to previous location
000E01	= AB?	^	Can be entered without data
000E00	= 12?	77=	Equal sign means stay at same address
000E00	= 77?	=	Can be used without any data
000E00	= 77?	.	Period means return to MACSbug
*			Returns to command level
*OP 21234			
021234	= FF?	99=	Example of trying to change ROM
NO CHANGE			Warning message
021234	= FF?	.	Does not abort command
*0P E00			
000E00 00? W			Enter invalid character
W IS NOT A HEX DIGIT			Print error message
*			Command is aborted

1.6.4 Define and Print Symbols SYMBOLS

COMMAND FORMAT	DESCRIPTION
SYmbol name hex value	Put a symbol in the symbol table with a hex value or assign a new value to a previously defined one. NAME can be 8 characters long, consisting of:A-Z,0-9,(period), and $(dollar sign). It must begin with letter (A-Z) or period.
SY -name	Remove a symbol from the symbol table
SY name	Print the current value of the symbol (absolute)
SY value	Print the first symbol with the given value
SY	Print the sorted symbol table

NOTE

Offset is not used by this command. Some commands recognize the words TILL, ALL. and CLEAR as key words and will not interpret them as symbols.

EXAMPLES	COMMENTS
*SY XYZ 5000	Puts the symbol in the table
*SY XYZ	Command prints out the symbol's current value
XYZ=5000	
*SY XYZ 123	Change a symbol's value
*SY ABC34 2500	Define another symbol
*SY Z17.RT5 XYZ	Define a symbol with value from another symbol
*SY 123	Print first symbol with value of 123
XYZ=123	
*SY B$67ABC 4300	Define some more symbols
*SY RFLAG 200	
*SY MVP2 9990	
*SY	Print the sorted symbol table

```
ABC34  00002500    B$67ABC    00004300    MVP2     00009990
RFLAG  00000200    XYZ        00000123    Z17.RT5  00000123
```

*SY TTT	Print a value for symbol not in table, when not found, it tries to convert parameter to number
T IS NOT A HEX DIGIT	
*SY 567	Attempt to print value for symbol not in table
00000567=567	

SYNTAX EXAMPLES	COMMENTS
*BR MVP2	Set a symbolic breakpoint
*CALL RFLAG	User define routine
*PC ABC34	Set a register
*DM MVP2 10	Display some memory

EXAMPLES OF KEY WORDS IN COMMANDS

*BR CLEAR	The word CLEAR is not considered a symbol here
*GO TILL Z17.RT5	The word TILL is part of the command
*T TILL ABC34	The word TILL is part of the command

1.6.5 Displaying and Accessing Memory through Windows WINDOWS

A "window" is an effective address through which the user can "see" memory. the windows are labeled W0 to W7 and are defined using the syntax listed below. The windows address corresponding memory locations labeled M0 to M7 which use the same syntax as registers. These memory locations can be examined, set or defined in the display the same as a register.

COMMAND FORMAT	DESCRIPTION
W#	Print the effective address of a given window
W#.len EA	Define a window size and effective address # is the window number 0 to 7 len is the length in bytes 1=byte; 2=word; 3=3 bytes; 4=long word 0=close a window (undefine it) EA is Effective Addressing mode (see EA SYNTAX EXAMPLES in table below) Pseudo registers have same syntax as registers

M# data or 'ASCII'

EA SYNTAX EXAMPLES	DESCRIPTION
FE8	Absolute address in hex
(A6)	Address register indirect in hex
100(A6)	Indirect with displacement in hex
-10(A6,D2)	Indirect with index and displacement in hex
-100(*)	Program counter with displacement in hex
10(*,D4)	Program counter with index and displacement in hex

```
EXAMPLES                        COMMENTS

*W3.4 (A6)                      Define a window:
*A6 92000                       Enter a value for the address register indirect
*W3                             Print the effective address of a window
W3.4 (A6)=92000
*M3 87342                       Set memory through the window
*M3                             Command to print memory through the window
M3=00087342
*DM 92000                       Display a line of memory
092000 00 08 73 42 00 00 00 00 00 00 00 00 00 00 00 00 . .sB.......
*TD CLEAR                       Clear all registers from the trace display
*TD PC.2 A6.3 M3.1              Define some registers for the display
*TD                             Command to print the trace display
PC=00A2 A6=092000 M3=42         NOTE:W3.4 and M3.1 only lowest byte displayed
*W3.2 (A6)                      Change width of window
*TD M3.2                        Change width of display
*TD
PC=00A2 A6=092000 M3=0008
*W0.1 10(*,A6)                  Define a new window:PC+A6+10
*W0                             Print effective address of window W0
W0.1 10(*,A6)=920B2
*W3.0                           Close window W3, undefine it
*TD
PC=00A2 A6=092000               Closed/undefined windows are not in the display
```

COMMAND FORMAT	DESCRIPTION
Go	Begin execution at address in PC register
Go address	Begin execution at this address
Go TILL address	Set a temporary breakpoint at the address and run until a breakpoint is encountered
BR	Print the address of all breakpoints (8 maximum)
BR address	Set a breakpoint at this address
BR -address	Remove the breakpoint at this address
BR address:count	Set a breakpoint at this address with a count
BR CLEAR	Remove all breakpoints

EXAMPLES (see example program in section 1.7)	COMMENTS
*PC E00	Set program counter to starting address
*TD CLEAR	
*TD PC.2 D0.1	Set trace display format
*TD	Print trace display
PC=0E00 D0=00	
*G TILL E08	Run until address
PC=0E08 D0=04	System displays when it stops
*BR 0E02	Set a breakpoint
*G	Run until breakpoint
PC=0E02 D0=01	Trace display
*BR E08:4	Set a breakpoint with a count
*BR	Print the breakpoints
BRKPTS= 0E02 0E08:4	
*G	Run
PC=0E00 D0=4	Decrements count, prints display, continues
PC=0E02 D0=1	Stops at breakpoint with zero count
*BR	Print the breakpoints
BRKPTS= 0E02 0E08:3	Count has been decremented by one
*BR -E02	Remove a breakpoint
*G	Run
PC=0E08 D0=4	Count from 3 to 2...
PC=0E08 D0=4	...2 to 1...
PC=0E08 D0=4	...1 to 0 and it stops here
*BR	Print the breakpoints
*BRKPTS= 0E08	No count for this breakpoint, does not reset back to count value
*BR E08:2	Reseting count
*G	
PC=0E08 D0=4	Count 2 to 1
PC=0E08 D0=4	Count 1 to 0 and stop
*BR E00	Set another breakpoint
*G E00	Start running from E00, bypass breakpoint at
PC=0E08 D0=4	starting address and stop at next breakpoint
*SY JUMPER E0A	Define a symbol
*BR JUMPER:5	Set a breakpoint at a symbolic address
*BR 123456:7897 11 22 33 44 55 66	Try to overflow table (holds 8)
TABLE FULL BRKPTS= E08 E00 E0A:5 123456: 7897 11 22 33 44	

Set the Trace Display Format (Individual Registers)

COMMAND FORMAT	DESCRIPTION
TD	Print the trace display
TD Clear	Take everything out of the display
TD ALI	Put all registers in display (see section 3.6.8)
TD reg#.format	Add or delete registers in display where reg# is D0-D7,A0-A7.W0-W7,M0-M7,PC.SR.US.SS.A.D, or L (see the next section). Format can be 0,1,2,3,4.Z,D,R. or S.

0=remove the item from the display
1,2,3,4=print this number of bytes as hex characters, include all leading zeros
Z=signed long word hex with zero suppress
D=signed long word decimal with zero suppress
R=subtract offset (see OFfset command) then print with Z format with letter 'R' at end
S=search symbol table for 4 byte value, if found print symbol name as 8 characters, if not found print hex value as 8 characters

EXAMPLES	COMMENTS
*PC 0	Initialize registers for example below
*D1 5	Initialize registers for example below
*A6 8F	Initialize registers for example below
*TD CLEAR	Turn off all the registers in display
*TD PC.3 D1.1	Define PC as 3 bytes and D1 as one
*TD	Command to display
PC=000000 D1=05	This is the trace display
*TD PC.0 A6	Remove PC and add A6 which defaults to 4 bytes
*TD	Display
D1=05 A6=0000008F	Display with two new registers
*W3.2 92000	Define a window
*M3 20	Set value of memory pseudo register
*TD M3.2	Add a memory pseudo register to the display
*TD	Display
D1=05 A6=0000008F M3=0020	New display
*TD A6.1 D1.3 M3.Z	Change length of registers already in display
*TD	Display
D1=000005 A6=8F M3=20	New display, M3 now suppresses leading zeroes
*TD D1.R M3.D	D1 is relative and M3 is decimal
*OFFSET 12345	Set the offset (see OFfset command)
*TD	Display
D1=-12340R A6=8F M3=32	5-offset=-12340r; 20 hex = 32 decimal
*SY TABLE 8F	Define a symbol (see SYmbol command)
*TD A6.S M3.0	Make A6 print symbol if value is in table
*TD	
D1=-1234OR A6=TABLE	Prints symbolic value
*A6 123	Set A6 to a value NOT inm symbol table
*TD	
D1=-1234OR A6=00000123	A6 prints value with 4 byte format

COMMAND FORMAT	DESCRIPTION
TD CLear	Take everything out of the display
TD D.1	Put all data registers in display as a block
TD A.1	Put all address registers in display as block (for D.1 and A.1 the format is fixed at 4 bytes)
TD L.character	Define a line separator at the end of display (.0 will reverse A.1, D.1, and L. char commands)
TD ALl	Same as keying: *TD PC.3 SR.2 US.4 SS.4 D.1 A.1 L.- does not affect other registers and windows that have been previously defined to display

EXAMPLES COMMENTS

```
*TD CLEAR                          Clear the display
*TD D.1                            Define all data registers in a block
*TD                                Print the trace display
D0=00000000 D1=00000000 D2=00000000 D3=00000000
D4=00000000 D5=00000000 D6=00000000 D7=00000000
*TD CLEAR
*TD A.1                            Define all address registers in a block
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00000FFC
*TD L.@                            Define a line separator (a row of '@')
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00000FFC
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*TD L.&                             Define a line separtator (a row of '&')
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00000FFC
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
*TD ALL                            Turn on commonly used registers...
*TD                                ... this is also the default or reset condition
PC=000000 SR=2000 US=00007F00 SS=00007FFE
D0=00000000 D1=00000000 D2=00000000 D3=00000000
D4=00000000 D5=00000000 D6=00000000 D7=00000000
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00000FFC
------------------------------------------------
*
```

COMMAND FORMAT	DESCRIPTION
Trace	Execute one instruction and print trace display
Trace count	Trace specified number of instructions
Trace TILL address	Trace to the given address (breakpoint will stop the trace)
:*(CR)	A colon (:) before the prompt indicates a special trace mode is in effect, a carriage return will trace the next instruction

EXAMPLES COMMENTS
(see example program in section 1.7)

```
*TD CLEAR                          Remove all of trace display
*TD PC.2 DO.1                      Display only PC and D0
*DM E00                            Example program in memory
000E00 70 01 70 02 70 03 70 04 70 05 4E F8 0E 00 FF FF
*PC E00                            Set the program counter
*TD                                Print the trace display
PC=0E00 D0=00
*T                                 Trace one instruction
PC=0E02 D0=01
:*(CR)                             Special prompt appears, carriage return will
PC=0E04 D0=02                      trace the next instruction
:*T3                               Trace three instructions
PC=0E06 D0=03
PC=0E08 D0=04
PC=0E0A D0=05
*T TILL E04                        Trace till instruction at address E04
PC=0E00 D0=05
PC=0E02 D0=01
PC=0E04 D0=02
*
```

The 68000 instruction set lends itself to relocatability and position independence. A general purpose, global offset feature has been provided. The single offset address applies to all of the commands listed below. Registers displayed in the trace display may have the offset subtracted by using R as the format. See paragrpah 1.6.7 on trace display.

The offset may be overriden by entering a comma and alternate offset. All commands do not use the offset but any number can be forced to be relative (have the offset added) by entering an R as the last character of the number.

WARNING: This is a very simple offset feature and may not be able to solve complex relocation problems. The user is encouraged to experiment with the global offset and the window features to determine their limitations and usefulness in a particular application.

COMMAND FORMAT	DESCRIPTION
OFfset	Display offset
OFfset hex value	Set the offset to a given value
OFfset 0	Set the offset to zero - begin absolute addressing
command data, alternate	Disregard offset, add alternate offset to data
command data,	Data is absolute, no offset added
command data,0R	Used in commands that do not normally use offset, adds offset to data

The offset affects the following commands:

TD reg.R	Trace display, substract offset from register value
BReakpoint	Set breakpoint (display is in absolute)
Go	All addresses
SM	All addresses
DM	All addresses (display is in absolute)
REad	All addresses

EXAMPLE	COMMENTS
*PC 2010	Set the program counter
*TD PC.R	Set trace display.R means nex long word minus offset
*TD	Display
PC=2010R	Displayed relative to offset (zero now)
*OF 2000	Set the offset ot 2000
*TD	Display
PC=10R	PC - offset = 2010-2000 = 10 Relative
*BR 6	Set a breakpoint: hex data+offset = 6+2000 = 2006
*BR	Display breakpoint
BRKPTS=2006	Breakpoints are always displayed as absolute hex
*BR 24,3000	Set a breakpoint with alternate offset 24+3000
*BR	
BRKPTS=2006 3024	

COMMAND FORMAT	DESCRIPTION
CV decimal or & decimal	Decimal to hex conversion
CV $hex	Hex to decimal conversion
CV symbol	Use value from symbol table
CV value,offset	Calculate offset or displacement

NOTE

This command DOES NOT automatically use the global offset. The default base for this command only is decimal. All numbers are signed 32 bit values.

EXAMPLES	COMMENTS
*CV 128	Command to convert decimal to hex
$80=&128	Computer response
*CV $20	Hex to decimal
$20=&32	
*CV -$81	Negative numbers
$FFFFFF7F=-$81=-&129	
*CV $444,111	Adding an offset (second number's base
$555=&1365	defaults to first number's)
*CV $444,-111	Subtracting an offset (forward displacement)
$333=&819	
*SY TEN &10	Defining a symbolic decimal constant
*SY THIRTY &30	
*CV TEN	Command can be used with symbols
$A=&10	
*CV -TEN	
$FFFFFFF6=-$A=-&10	
*CV THIRTY,-TEN	
$14=&20	
*OF 2000	Define a global offset
*CV $123R	R at the end of a number means add the
$2123=&8483	global offset
*CV TEN,0R	Symbolic relative
$200A=&8202	

COMMAND FORMAT	DESCRIPTION
REad;-CX =text	Load S records - default PORT 2 option -C means ignore checksum; option X means display data being read; if equal sign is used in this command line then everything after it is sent to PORT 2
VErify;=text	Verify memory with S records - print difference; verify does not use checksum

NOTE

These commands use the offset. No attempt is made to control the host transmissioins. For the REad and VErify, any line received not beginning with the letter S is ignored (see appendix A for S record formats). If an error occurs causing the system to take the time to print out an error message, one or more lines sent during the error message may have been ignored.

```
EXAMPLE                    COMMENTS

*READ;=COPY FILE.MX,#CN              Download from an EXORciser.
*DM E00 10                           Check to see if data was loaded
000E00 70 01 70 02 70 03 70 04 70 05 4E F8 0E 00 FF FF
*VERIFY;=COPY FILE.MX,#CN            Normal verify returns with prompt
*SM E05 FF                           Deliberately change memory to show verify
*DM E00                              Verify that 03 was changed to FF
000E00 70 01 70 02 70 FF 70 04 70 05 4E F8 0E 00 FF FF
*VERIFY;=COPY FILE.MX,#CN
S1110E00          03                 Displays only nonmatching data bytes
*RE;=COPY FILE2.MX,#CN               Example of file with bad character
S1110E007001700270/3700470054EF80E0049 NOT HEX=/
*RE;=COPY FILE2.MX,#CN               Example of file with bad checksum
S1110E00700170027003700470054EF80E0039 CHKSUM=49
*RE;=COPY FILE.MX,#CN                Normal read returns with prompt
*OF 5423
*RE;=COPY FILE.MX,#CN                Download with offset
*DM 1000                             Display memory. adds offset to parameters
006423 70 01 70 02 70 03 70 04 70 05 4E F8 0E 00 FF FF
```

1.6.13 The CALL Command CALL

The call command can be used to add commands. This is done by writing a subroutine which ends with an RTS.

The call command does not affect the user's registers and is not to be confused with the GO command. The user may use a symbol as the command parameter instead of an absolute starting address. Registers A5 and A6 point to the start and end of the I/O BUFFER (see RAM equate file listing, paragraph 1.11) so the user may pass additional parameters on the comand line.

COMMAND FORMAT	DESCRIPTION
CALL address	JSR to user subroutine, routine must end with RTS
EXAMPLE	**COMMENTS**
*CALL 3000 23 45 ZZ	JSR to user routine at location 3000 note that 23 45 & ZZ may be additional parameters that the user's subroutine will decode and are ignored by MACSbug
*SY FIXUP 2300	Define a symbol as absolute address 2300
*CALL FIXUP	JSR to symbolic address

1.6.14 Transparent Mode and Host Communication TRANSPARENT

COMMAND FORMAT	DESCRIPTION
P2 [char]	Enter transparent mode: The optional user defined exit character [char], defaults to control A ($01). This command logically connects port 2 (host) and port 1 (console). Host transmissions go directly to the console and console transmissions go directly to the host. The BAUD rates on the two ports may be the same or port 2 may be less.
(control A)	Default character to end the transparent mode, alternate character may be defined in P2 command
*...data...	Asterisk.*, as the first character of the console input buffer means transmit the rest of the buffer to the host (PORT 2), the BAUD rates of the two ports (1 and 2) do not have to be the same.

EXAMPLES	COMMENTS
MACSBUG 2.0	Start up or reset condition
*P2	Command to enter transparent mode
TRANSPARENT EXIT=$01	MACSbug prints this, the EXIT=$01 means to exit this mode, enter control A
}	User talks direct to the host, uses the editor, assembler, etc.
(CONTROL A)	Ends the transparent mode
MACSBUG	MACSbug prints this and system is ready for new command
**MAID	System prompts with * and user enters '*MAID'
**E800; G	Everything after the second * is sent to the host (NOTE: the BAUD rates do not have to be the same)
*P2 &	Enter transparent mode, '&' is the exit character
TRANSPARENT EXIT=$26	Displays exit character (&) as hex value 26
}&	User exits transparent mode by entering '&'
MACSBUG	
*	Command mode prompt

1.7 EXAMPLE OF COMMAND PROCEDURES

```
MACSBUG 2.0                          Start up condition
*P2                                  MACSbug prompts with * user enters P2 to
                                     enter transparent mode.
*TRANSPARENT* EXIT=$01               Message printed to indicate user is now
                                     directly connected with host system

 - NOTE: The following example is using a MOTOROLA EXORciser host system -

 MAID                                Boot up MDOS
 **E800;G
 MDOS3.0
 =MACS FILE;CO                       Assemble a source file (see M68000 Cross
                                     Macro Assembler manual)
FILE      MC68000 ASM REV= 1.OC - COPYRIGHT BY MOTOROLA 1978
   1                              *
   2                              *  EXAMPLE PROGRAM FOR 68000 MACSBUG
   3                              *  TO DEMONSTRATE TRACING, BREAKPOINTS, AND GO
   4           00000E00                 ORG $0E00
   5  000E00   7001               START   MOVE.L #1,D0    1 LOADED INTO REG DO
   6  000E02   7002                       MOVE.L #2,D0    2
   7  000E04   7003                       MOVE.L #3,D0    3
   8  000E06   7004                       MOVE.L #4,D0    4
   9  000E08   7005                       MOVE.L #5,D0    5
  10  000E0A   4EF80E00           JUMPER  JMP START         DO IT AGAIN
  11                                       END

******TOTAL ERRORS 0 - 0

SYMBOL TABLE
  JUMPER    000E0A START   000E00

=COPY FILE. MX, #CN                  MDOS command to list file on console
S00600004844521B                             Header record
S1110E00700170027003700470054EF80E0049    Data record
S9030000FC                                   End-of-file
=(control A)                         Ends transparent mode
*MACSBUG*                            Message put out by MACSbug to indicate user is
                                     now in MACSbug command mode
*READ ;=COPY FILE.MX,#C              Download from EXORciser host (see sec. 1.6.12)
*DM E00                              Display memory                (see sec. 1.6.2)
000E00 70 01 70 02 70 03 70 04 70 05 4E F8 0E 00 FF FF
*PC E00                              Set program counter to START  (see sec. 1.6.1)
*TD CLEAR                            Clear the trace display       (see sec. 1.6.7)
*TD PC.2D0.1                         Specify which registers to print in display
*TD                                  Print the trace display
PC=0E00 D0=00
*BR E04                              Set a breakpoint              (see sec. 1.6.6)
*T TILL 0                            Trace command                 (see sec. 1.6.9)
PC=0E02 D0=01
PC=0E04 D0=02                        Stopped at breakpoint
*GO                                                                (see sec. 1.6.6)
PC=0E04 D0=02                        Stopped at breakpoint
*                                    Program is ready to run
```

1.8 I/O SPECIFICATIONS

Provisions have been made for the user to substitute his own I/O routines and direct the I/O for some commands to these routines. There are three pairs of locations in RAM that hold the addresses of the I/O routines. (See paragraph 1.11 on the equate file of RAM locations used by MACSbug.) They are initialized when the system is booted to contain the addresses of the default routines in MACSbug ROMs.

INPORT1 and OUTPORT1 are defaulted to port 1 which is MACSbug's console. The MACSbug prompt, command entry, all error messages, and all other unassigned I/O use these addresses to find the I/O routines. Most commands do not need a port specifier to use PORT 1. The REad and VErify commands, however, default to PORT 2.

INPORT2 and OUTPORT2 are defaulted to port 2 which is the host system (an EXORciser or timesharing system, etc.). Output or input is directed to this port by including a port specifier in the command field of the command line.

For example: *RE2;-C

The 2 in the command RE2 specifies that the addresses for the I/O routines will be found in the RAM locations INPUT2 and OUTPUT2. Error messages, however, will be printed on PORT 1 - MACSbug's console.

INPORT3 and OUTPORT3 are inititalized to the same routine addresses as PORT 1 when the system is booted. The user can insert the addresses of his own I/O routines into these locations. I/O can then be directed to his configuration by using a 3 in the command field.

EXAMPLES	COMMENTS
*READ3;-C	Memory load from port 3; checksum ignored
*VERIFY1	Verify memory with 'S' records coming in from PORT 1
*DM2 50 80	Display memory sending output to PORT 2

The BAUD rates of the two RS-232C serial ports can be changed by setting memory locations $06BA and $06BC.

ADDRESS	PORT	VALUE
$06BA	1	1X
$06BC	2	1X

The Hex digit X can be set to select various BAUD rates as shown below:

X =	6	7	8	A	C	E	F
BAUD RATE =	300	600	1200	2400	4800	9600	19200

EXAMPLES	COMMENTS
SM 6BA 16	Set BAUD rate to 300 for port 1
SM 6BC 1F	Set BAUD rate to 19200 for port 2

1.9 USER I/O THROUGH TRAP 15

Format in user program:

```
            TRAP    #15       Call to MACSbug trap handler
            DATA.W  function  Valid functions listed below.
                              Program resumes with next instruction.
```

FUNCTION	DESTINATION	FUNCTION	BUFFER
0		Coded Breakpoint	
1	PORT1 console	Input line	A5=A6 is start of buffer.
2	PORT1 console	Output line	A5 to A6-1 is buffer.
3	PORT2 host	Read line	A5=A6 is start of buffer.
4	PORT2 host	Print line	A5 to A6-1 is buffer.

EXAMPLE PROGRAM:

```
                        1*
                        2* ;       file : MBUG.EX.TEXT
                        3*
                        4* ;       Example of using TRAP #15 facility in
                        5* ;       MacsBug.  This program is assembled with
                        6* ;       ASM68K then linked using the Concept
                        7* ;       LINKER.  It was executed by calling out
                        8* ;       the code file.
                        9* ;
                       10* ;       COMMAND LINE:            COMMENT:
                       11* ;       asm68k mbug.ex           assemble file
                       12* ;       linker mbug.ex           link
                       13* ;       mbug.ex                  execute
                       14* ;
                       15*
                       16*
0000  4BFA  001A+      17* START     LEA     BUFFER, A5     ;Init buffer
0004  2C4D             18*           MOVEA.L A5, A6         ;pointers
                       19*
                       20* ;       Input buffer from Port 1
                       21* ;
0006  4E4F             22*           TRAP    #15            ;echoes input
0008  0001             23*           DATA.W  1
                       24*
                       25* ;       Output buffer to Port 2
                       26* ;
000A  4E4F             27*           TRAP    #15
```

```
000C  0004           28*           DATA.W  4
                     29*
                     30* ;      Enter MacsBug   -   a coded breakpoint
                     31* ;
000E  4E4F           32*           TRAP    #15
0010  0000           33*           DATA.W  0
                     34*
                     35* ; if first char in buffer = "!" then exit
                     36* ;
0012  7021           37*           MOVEQ   #'!',D0
0014  B03A  0006+    38*           CMP.B   BUFFER, D0     ;1st char = "!"
0018  66E6           39*           BNE.S   START          ;no, do again
                     40*
001A  4E75           41*           RTS
                     42*
                     43* ; BUFFER
                     44* ;
001C  00000000       45* BUFFER    DATA.L  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0054  00000000       46*           DATA.L  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
008C  00000000       47*           DATA.L  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                     48*
      00000000+      49*           END     START

 BUFFER      00001C+  START       000000+
```

1.10 GENERAL INFORMATION

TRAP ERROR is the general message given when an unexpected trap occurs. Nearly all of the low vectors including the user traps, interrupts, divide by zero, etc. are initialized during booting to point to this simple error routine. No attempt is made to decipher which trap happened, but the user's registers are saved. The system usually retrieves the right program counter from the supervisor stack but some exception traps push additional information on to the stack and the system will get the program counter from the wrong place. It is recommended that the user's program reinitialize all unused vectors to his own error handler.

The REad command may have problems in some configurations. No attempt is made to control the equipment sending the information. When the system recognizes the end of a line it must process the buffer fast enough to be able to capture the first character of the next line. Normally the system can download from an EXORciser at 9600 baud. If the system is having problems, it might be worthwhile to experiment with lower BAUD rates.

The REad routine DOES NOT protect any memory locations. The routine will not protect itself from programs trying to overlay the I/O buffer. This will, of course, lead to errors during the download. Any location in memory can be loaded into, including MACSbug's RAM area. This allows the user to initalize such locations as the starting and ending address of the symbol table. All the registers may be initialized except the program counter which takes its address from the S8 or S9 record.

The REad command, supports the normal S0, Sl, S2, S8. and S9 record formats. (See Appendix for a description of these S Records.)

TRAP 15 is used by both the user I/O feature and breakpoints. When the program is running, the address of the breakpoint routine is normally in the TRAP 15 vector. When program execution is stopped, the I/O routine address is normally inserted into TRAP 15 vector. If I/O is not needed in the program, the user may change the vector with the SM command. If breakpoints are not needed, the program may change the vector while the program is running. It is recommended, however, that the user should use the other 15 vectors (or other programming techniques) and let MACSbug control TRAP 15.

1.11 EQUATE FILE OF RAM USED BY 68000 MACSbug 2.0

* WARNING TO USER: The addresses listed below and their usage as described in this document are intended for only this version (2.0) of MACSbug. Corvus does not guarantee the usage of these locations.

```
                ORG $400
400   REGPC     DS.L 1          USERS PROGRAM COUNTER
404   REGSR     DS.L 1          USERS CONDITION CODES
408   REGS      DS.B 4*2*8      4BYTES*3SECTIONS*8REG(OR MEM)
444   REGA7     EQU REGS+60     WHERE A7 REG IS
448   REGUS     DS.B 4          USER STACK
44C   OFFSET    DS.L 1          ASSUMED OFFSET
450   FORMAT    DS.B 36         TRACE DISPLAY FORMATS
474   ADALL     DS.L 1          SPECIAL FORMAT FLAGS
478   WINDOWS   DS.B8*8         WINDOW PARAMETERS
4B8   LOOPR1    DS.L 1          LOW RANGE FOR LOOP FEATURE
4BC   LOOPR2    DS.L 1          HIGH RANGE FOR LOOP FEATURE
4C0   BPADD     DS.L 8          BREAKPOINT ADDRESSES
4EO   BPTILL    DS.L 1          TEMPORARY BREAKPOINT
4E4   BPCNT     DS.L 9          BREAKPOINT COUNTS
508   BPDATA    DS.W 9          HOLD USER WORDS REPLACED BY TRAP IN SET
51A   SAVETRAP  DS.L 1          HOLDS USER'S TRAP 15 VECTOR
51E   NULLPADS  DS.B 2          CHARACTER NULL PADS
520   CRPADS    DS.B 2          CARRIAGE RETURN NULL PADS
522   SBIT      DS.B 2          STOP BITS (ACIA PROGRAM)
524   OUTTO     DS.B 4          HOLDS ADDRESS OF OUTPUT ROUTINE
528   INFROM    DS.B 4          HOLDS ADDRESS OF INPUT ROUTINE
52C   ALTACIA1  DS.L 1          ALTERNATE ACIA PORT#1
530   ALTACIA2  DS.L 1          ALTERNATE ACIA PORT#2
534   INPORT1   DS.L 1          INPUT ROUTINE ADDRESS
538   OUTPORT1  DS.L 1          ADDRESS FOR OUTPUT ROUTINE
53C   INPORT2   DS.L 1          ADDRESS FOR INPUT ROUTINE
540   OUTPORT2  DS.L 1          FOR OUTPUT ROUTINE
544   INPORT3   DS.L 1          PORT #3 INPUT ROUTINE
548   OUTPORT3  DS.L 1          PORT #3 OUTPUT ROUTINE
54C   TRACECNT  DS.L 1          TRACE COUNTER
550   TRACEON   DS.W 1          FLAG FOR TRACE ON
552   RUN       DS.W 1          1=SAVE USER REGISTERS;0=NOT
554   BPSTATUS  DS.W 1          1=BP ARE IN; 0=ARE OUT OF MEMORY
556   SCREEN1   DS.L 1          PRINT THIS BEFORE TRACE DISPLAY
55A   SCREEN2   DS.L 1          PRINT THIS AFTER
55E   BASE      DS.B 2          WORK VARIABLE
560   SIGN      DS.B 2          WORK VARIABLE
562   VECTOR    DS.B 2          WORK VARIALBE
564   TEMP      DS.B 4          WORK SPACE
568   WORK1     DS.L 1          WORK SPACE
56C   WORK2     DS.L 1          WORK SPACE
570   STRSYM    DS.L 1          START OF SYMBOL TABLE
574   ENDSYM    DS.L 1          END OF SYMBOL TABLE
578   CMDTABLE  DS.L 1          START OF COMMAND TABLE
57C   BUFFER    DS.B $128       WORKING STORAGE BUFFER
6A4             DS.B 20         ROOM FOR STACK
6B8   SYSTACK   DS.B 2          START OF STACK (GOES DOWN)
```

S RECORDS

An S record is a standard Motorola record format used in downloading programs and data with MACSbug.

There are ten possible standard S record types, five of which can be used with MACSbug. They are as follows:

S0	Header record
S1	16 bit address Data record
S2	24 bit address Data record
S8	24 bit address End of File/Execution Address record
S9	16 bit address End of File/Execution Address record

The standard S record is defined as follows:

FRAME	VALUE	DESCRIPTION	BYTE COUNTED	CHECK SUMMED
1	$53 (S)	Start of Record		
2	$30-$39 (0-9)	Record Type		
3,4		Byte Count		*
5-8		Address (for 16 bit)	*	*
5-10		Address (for 24 bit)	*	*
:			*	*
:		Data	*	*
:			*	*
N-1,N		Checksum	*	

The letter "S" and the Record Type are represented directly in ASCII.

The byte count, address, data, and checksum are represented in ASCII encoded hexadecimal; i.e., two frames per data byte, with the most significant digit in the leading frame.

The checksum is the 1's complement of the sum of all 8-bit data/address bytes from byte count to last data byte, inclusive.

TYPICAL OBJECT S-RECORD FORMAT

```
S00600004844521B
S1131000307C1000327C1FFE123C00804280428300
S1131010383C09964A016A0000121A18B0C96600E1
S1131020000AD2FC00026000002EE3113400E352F7
S113103002420008E30D050466000006E25860D48A
S1131040E2580840000F60CC4A016A00000A1A18EE
S1131050B0C96700002AE3113400E3520242000BD6
S113106005046600000CE35B08C300006000000890
S1131070E35B08830000E25808C0000F60CA31C374
S10710801FFE4E728B
S00600004844521B
S20A010000323C00035641ED
S9030000FC
```

First two characters	- S0 Starts of the first record.
	- S1 Indicates that the object data that follows will be at a two-byte memory address.
	- S2 Same as S1, but indicates a three-byte memory address.
	- S8 Same as S9, but indicates a three-byte memory address.
	- S9 Last Record
Third and fourth characters	- Hexadecimal byte count of the remaining characters in the record.
Fifth through eighth characters	- Hexadecimal memory address where the data that follows is to be loaded. If the record is "S2" or "S8" type, the fifth through tenth characters contain the memory address.
Last two characters	- Checksum of all characters from byte count to the end of data.