Twenty Dollars

Cromemco®

68000 Debugger

Instruction Manual

CROMEMCO, Inc.
280 Bernardo Avenue
Mountain View, CA. 94043

Part No. 023-4054

February 1983

Cromemco®

68000 Debugger

Instruction Manual

CROMEMCO, Inc.
280 Bernardo Avenue
Mountain View, CA. 94043

Part No. 023-4054

February 1983

This manual was produced using a Cromemco System Three computer with a Cromemco HDD-22 Hard Disk Storage System running under the Cromemco Cromix® Operating System. The text was edited with the Cromemco Cromix Screen Editor. The edited text was proofread by the Cromemco SpellMaster™ Program and formatted by the Cromemco Word Processing System Formatter II. Camera-ready copy was printed on a Cromemco 3355B printer.

TABLE OF CONTENTS

Chapter 1

LOADING AND RUNNING DEBUG68

LOADING DEBUG68

The Cromemco Debug68 program makes it possible to test, debug, and trace through 68000 user programs. Debug68 is loaded into memory at a bank boundary (e.g., 20000h, 30000h, etc.) and occupies 128K of memory, including user program space and stack. In the standard configuration, the length of the user program can be up to 112K. A minimum of 256K ≈ 50K bytes of user system memory is required to run the debugger. If you need to change the size of the debugger, you can easily change the long word size of the program at location 8 by using the Patch utility.

Load Debug68 by entering the following command:

DEBUG68 [path-name program-params]

where **path-name** is the pathname of the program to be tested and **program-params** are optional program command line parameters.

The command is used without any options to load the Debug68 program into memory and to run the debugger alone.

With options, the command is used to load Debug68 and the file to be tested into memory with the optional parameters properly passed to the program being debugged. **Debug68 can load only bit-mapped 68000 code .bin files** that are linked with Cromemco Link68 or Crolinker. If you try to load any other type of file, an error message results and execution of Debug68 is aborted.

CONTROL CHARACTERS

Control characters are used in Debug68 to help enter commands. These control characters are the same as those used by the Cromix Operating System.

CONTROL-C (^C)	abort program and exit to the Cromix Operating System (then run Restrp to restore system trap vector)
CONTROL-H (^H)	delete character and backspace on CRT
CONTROL-U (^U)	delete line
RUBout (DEL)	delete character and backspace on CRT

While displaying new information (such as that displayed by the Display Memory command) the following characters may be used:

CONTROL-S (^S)	stop listing to the console.
CONTROL-Q (^Q)	resume listing to the console.
SPACE	(or any other character) will abort the listing or console display. The debugger will prompt and wait for the next command.

COMMAND FORMAT

Debug68 is controlled by one- and two-character commands from the terminal. The format is free-form with respect to SPACEs. Remember that all commands must be terminated by pressing the RETURN key. The following examples all display memory starting at location 31000h and ending at location 310FFh.

```
D31000 310FF
DM31000S100
DM 31000 310FF
D 31000 S100
```

CURRENT PROGRAM COUNTER LOCATION ($)

The current address (the current value of the program counter) may be represented by a dollar sign ($). The following command begins program execution at the current value of the program counter and stops execution (by the use of a temporary break point) when the program counter reaches its current value plus 3.

```
G/$3
```

COMMAND STARTING ADDRESS

The Display Memory (DM or D), List in Assembler Mnemonics (L), and Substitute Memory (SM or S) commands <u>each</u> maintain a starting address to use if none is given with the command. This address is changed each time one of the specified commands is executed so that the next execution of the command commences where the last one ended. When Debug68 is entered, the starting addresses of the DM and SM commands are set to address FIRST, and the L command starting address is set to address START.

@ REGISTER

Debug68 is designed to aid the programmer in testing relocatable programs. The @ register tells Debug68 where the module you wish to debug is located. This address can be found from the map generated by the Cromemco 68000 linkers. To change the @ register, type:

```
@
```

on the console. Debug68 will then display:

```
@ = xxxxxx
```

where xxxxxx is the current value of the @ register. The computer then waits for a new address. If only a RETURN is typed, the register remains unchanged. If an address and a RETURN are typed, the register will contain the new address. The @ register may now be used as part of an address:

```
G/@ @A3 24000
```

This is an example of the Go command. Break points will be set at the beginning of the current module, relative location A3h in the current module, and at location 24000h. This feature allows you to test a module without having to calculate absolute addresses.

The relative address is displayed in addition to the absolute address whenever addresses are displayed unless the @ register equals zero.

ADDRESS EXPRESSIONS

For ease in specifying addresses, an expression can be used. Within these expressions, addition, subtraction, the relative address (@) register, and the current program counter location ($) may be used. If many modules are being tested, addition can be used to specify relative addresses. Whenever it makes sense, the debugger will check for an even address and will abort if the address is odd.

Expressions

Special Symbols:

@	@ register
$	Program counter
(expr)	Contents of memory addressed by "expr"
'xy'	ASCII value of "xy"
ddddd.	Decimal number
hhhh	Hexadecimal number

Binary Operators:

+	Addition
-	Subtraction

An expression can be used in Debug68 anywhere that an address is needed. Expressions must not contain any embedded SPACEs. The precedence of operators is as follows:

1. Special symbols (highest precedence)
2. Binary operators

Among operators of the same precedence, evaluation proceeds from left to right.

SWATH OPERATOR

There are two ways to specify the address range of many commands. The first is to list the beginning and ending addresses (and, where appropriate, the destination address). In this example:

```
M24000 24fff 27000
DM23000 23002
```

The first command moves the contents of memory from 24000h through 25000h to memory starting at location 27000h. The second command displays the contents of memory between addresses 23000h and 23002h.

Another way to do the same thing is to use the Swath operator to specify the width of the address range, rather than explicitly stating the end address.

```
M24000 S1000 27000
DM 23000S3
```

ERRORS

Any errors made during command entry may be corrected by typing CONTROL-U (^U) to abort the line or by backspacing and correcting the line. If a RETURN has already been entered and Debug68 detects an error, the line will not be accepted and a question mark will be displayed. The command must then be re-entered correctly.

Chapter 2

DEBUG68 COMMANDS

A - ASSEMBLE INTO MEMORY

This command allows you to enter assembly language mnemonics from the console and have them assembled into memory. The command has two formats:

```
A
A beginning-addr
```

The first format assembles into memory starting at the default current address (initially first). With the second format, assembly starts at beginning-addr.

You are prompted with the absolute address and the current instruction located at that address.

Debug68 reads from the console and assembles the instruction into memory. Mnemonics for the MC68000 instructions are contained in Motorola's MC68000 User's Manual. Two operation codes are not implemented, Reset and Illegal, but the Assembler recognizes the Jsys op code followed by an immediate operand. If there is no error in the instruction, it is assembled into memory and you are prompted for the next instruction. In the following example the @ register contains 25000h.

```
A@
25000  ORI     #00,D0        move   #1, dl
25004  ORI     #00,D0        lea     25010h, a0
2500A  ORI     #00,D0        jsys   #19h
2500E  ORI     #00,D0        jsys   #46h
25012  ORI     #00,D0
```

If you type only a RETURN character, Debug68 does not alter the current instruction and goes to the next instruction in memory.

The A command terminates when a line starting with a period (.) is entered. If there is an error in the input line, the command is not accepted, a question mark is displayed, and you are prompted again.

With the following exceptions, the syntax for this Assembler is very similar to that used by the Cromemco 68000 Macro Assembler (refer to the 68000 Macro Assembler Manual).

1. Expressions are limited to signed or unsigned decimal or hexadecimal numbers.

2. Whenever an address is specified, it is taken as program counter relative.

3. Absolute addresses are marked with size extension (.W or .L).

4. When program counter relative with index addressing mode is specified, index register must have a size extension, e.g., CMP 23000h(D0.L),D7.

5. An immediate operand can be a string of appropriate size, e.g., MOVE.B #'a',D0 or MOVE #'ab',D0. However, the instruction MOVE.B #'ab',D0 will give an error.

B - SET OR DISPLAY PERMANENT BREAK POINTS

The B command is used to set permanent break points. Using this feature, it is not necessary to repeatedly set break points when using the G command. A permanent break point will remain in effect until it is explicitly removed by the BX command.

It is possible to set a total of 16 break points of which 8 are permanent breakpoints (B command), and 8 are temporary breakpoints (G command).

To display all of the currently active permanent break points, enter the command:

B

To set a permanent break point enter the command:

B breakpoint-1 breakpoint-2...breakpoint-n

A break point is specified by an even address.

```
B 338BC
```

This will establish a permanent break point at memory location 338BCh. Whenever the program counter (PC) is equal to 338BCh, program execution will stop and the registers will be displayed in a standard format.

BX - DELETE PERMANENT BREAK POINTS

This command is used to delete permanent break points. It has two formats:

```
BX
BX addr-1 addr-2 ...
```

The first format deletes all permanent break points. The second format deletes the break points at each of the specified addresses.

C - TRACE OVER CALLS

```
C
```

The Trace Over Calls command functions similarly to the simple Trace command. The difference appears when a JSR or BSR instruction is encountered during instruction trace.

When a JSR or BSR instruction is encountered during the execution of the C command, no tracing is performed in the called subroutine. A temporary break point is inserted after the call instruction, and execution will stop after the RTS or RTR instruction is executed in the called subroutine.

D or DM - DISPLAY MEMORY

The contents of memory is displayed in hexadecimal notation. Each line of the display is preceded by the address of the first byte and is followed by the ASCII representation of the hexadecimal bytes. For example:

```
-> dm33000 s30
033000  3132 3334  3536 3738   3930 6162  6364 6566  1234567890abcdef
033010  6768 696A  6B6C 6D6E   0000 0000  0000 0000  ghijklmn........
033020  0000 0000  0000 0000   0000 0000  0000 0000  ................
```

If the @ register is not equal to zero, the relative address will be displayed as follows (assume the @ register has been set to 33000h):

```
-> dm33000 s30
033000  000000'  3132 3334  3536 3738   3930 6162  6364 6566  1234567890abcdef
033010  000010'  6768 696A  6B6C 6D6E   0000 0000  0000 0000  ghijklmn........
033020  000020'  0000 0000  0000 0000   0000 0000  0000 0000  ................
```

The formats of this command are as follows:

DM
DM beginning-addr
DM beginning-addr ending-addr
DM beginning-addr S swath-width
DM S swath-width

The M is optional in all formats.

The first format displays memory from the current display address, initially at FIRST, and continues for four lines. The second format displays from the beginning address and continues for four lines. The third format displays from the beginning address to the ending address. The fourth format displays from the beginning address for a length specified by the swath width. The fifth format displays from the current display address for a length specified by the swath width.

DR - DISPLAY REGISTERS

When Debug68 is re-entered from a break point, the user registers are saved. The registers may be displayed in response to the Debug68 prompt by typing the following command:

```
-> dr
        0        1        2        3        4        5        6        7
Dn = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
An = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0004FFF4
XNZVC SR=001F PC=033000 000000' ORI.B   #00,D0
```

Notice that the relative address of the program counter (PC) is displayed only if the @ register is not equal to zero.

The letters XNZVC on the fourth row represent the flags in the user part of the status register (CCR). If the flag is on, it is displayed; if the flag is off, a space is displayed. If only the carry and zero flag are set then " Z C" would be displayed. The flags are described below.

X - Extend flag, X=1 if the operation performed generated carry or borrow. Transparent to data movement.

N - Negative flag, N=1 if the MSB of the result is one.

Z - Zero flag, Z=1 if the result of an operation is zero.

V - Overflow flag. This flag is affected by arithmetic operations. If an overflow occurs during an arithmetic operation, the flag is set to one.

C - Carry flag, C=1 if it is generated out of the most significant bit of the operands for an addition. Also set if a borrow is generated in a subtraction.

The numbers on the first line are the register numbers. On the second line, data registers are displayed followed by the address registers on the third line.

After the flags on the fourth line, the status register (SR) is displayed. The program counter value is then displayed as both an absolute and relative address. The opcode pointed to by the program counter is then displayed as an instruction mnemonics.

If the disassembled opcode includes an address, the relative value of this address is displayed as the last item on the line.

E - EXAMINE INPUT PORT

The data port is read and displayed as a hexadecimal number. The format of the command is:

E data-port

Port number is in the range of 0 to 0FFFFh, and it is converted to an address in the range of 0FF0000h to 0FFFFFFh to enable access to memory mapped input-output ports. In the following example, the data port 3 is read and displayed on the console.

```
-> E3
23
```

EX - EXIT TO OPERATING SYSTEM

The command:

EX

closes all files, restores break point trap and trace vectors to original values, and exits to the operating system.

If you exit to the operating system using CONTROL-C, run the Restrp utility to restore the system trap vector.

F - SPECIFY FILENAME

This command allows the programmer to specify the pathname of the file to be read into Debug68 user space. The pathname can be any valid Cromix pathname up to 128 characters long.

```
        -> F/usr/anyfile.bin
```

This command can be used with the R command to read disk files.

G - GO

The GO command has the following format:

G starting-addr/break point-1 break point-2...break point-n

Each of the addresses is optional. If the starting address is omitted, the contents of the program counter is used. The registers are loaded from the user registers (these are the values displayed by the DR command). Execution begins with the starting address or the contents of PC (the program counter). If break points were specified, a TRAP #3 is inserted at each break point address. When a break point is executed, control is returned to Debug68, and all of the user registers are saved and displayed. All temporary break points (those established by the G command) are then removed from the user program. Note the following about break points:

1. Break points can only be set in programs residing in RAM. This is because a TRAP #3 is inserted at each break point location. (The original contents of these locations are saved so that they can be restored after a break point is executed.)

2. Up to 16 break points can be set. If you attempt to enter more than 16 break points, the command will not be accepted. The 16 breakpoints may be composed of 8 permanent and 8 temporary breakpoints.

Refer to the B and BX commands for a description of permanent and conditional permanent break points.

H - HEXADECIMAL ARITHMETIC

H expr

The command evaluates the expression and displays the result in hexadecimal and decimal notation.

In the following example, two hexadecimal numbers (1234h and 321h) are added and subtracted.

```
-> h 1234+321
00001555 5461.

-> h 1234-321
00000F13 3859.
```

L - LIST IN ASSEMBLER MNEMONICS

The List command is used to list the contents of memory in assembly language mnemonics. The formats of this command are:

L
L starting-addr
L starting-addr ending-addr
L starting-addf S swath-width
L S swath-width

The first format lists 16 lines of disassembled code, starting from the current list address (initially START). The second format lists 16 lines from the starting address. The third format lists from the starting address to the ending address. The fourth format lists from the starting address for a length specified by the swath width. The fifth format lists from the current address for a length specified by the swath width.

The first address of the listing is the absolute address, and the second address is the relative address. If the disassembled instruction contains an address, the absolute address is displayed in the instruction in hexadecimal notation, and the relative address is displayed to the right of the disassembled line. In the example that follows, the @ register contains 40000h.

```
-> l @ s20
040000  000000' MOVEA.L A7,A5
040002  000002' ADDQ.L  #2,A5
040004  000004' MOVEQ   #01,D1
040006  000006' CLR     D5
040008  000008' LEA     04002E,A0          00002E'
04000C  00000C' MOVE.L  (A5)+,-(A7)
04000E  00000E' BEQ.S   040022             000022'
040010  000010' MOVE    D5,-(A7)
040012  000012' JSYS    #1B
040016  000016' MOVE    (A7)+,D5
040018  000018' ADDA.L  #00000004,A7
04001E  00001E' ADDQ    #1,D5
```

M - MOVE MEMORY

The formats of this command are:

```
M source-addr source-end destination-addr
M source-addr S swath-width destination-addr
```

The first format moves the contents of memory, beginning with the source address and ending with the source-end to the destination address. The second format uses the swath width to determine the length of the move.

The Move command can be used to fill a block of memory with a constant. In the following example, a zero has been entered into location 23100h using the SM command. The following command will move zeros from location 23100h through 23108h.

```
-> M23100 S7 23101
```

Take care not to move memory over Debug68 or the Cromix Operating System or some other user's program while in unprotected memory mode.

O - OUTPUT TO DATA PORT

This command outputs data to a data port. The following is the command format:

O data-byte port-number

Q - QUERY MEMORY

The format of the Query command is:

```
Q beginning-addr ending-addr string-of-bytes
Q beginning-addr S swath-width string-of-bytes
```

This command is used to search through a specified area of memory for a certain string of bytes. The string of bytes is in the same format as the S or SM command. If the string of bytes is found, 16 bytes, starting at the first byte which matches, are displayed as in the DM command.

Example:

```
Q 20000 50000 'cromix.sys'
```

R - READ DISK FILE

The R command is used with the F command to allow the programmer to read a disk file. Only files with bit-mapped M68000 code can be loaded with this command. The F command is used to specify the file name, and the R command causes the file to be read into memory. The format of the R command is:

```
R
```

After receiving the command, Debug68 checks if the specified file contains the M68000 code. Further, code size is determined and Debug68 tries to load the file at a bank boundary (i.e., 30000h, 40000h, etc.). If the file is bigger than 60K, Debug68 loads the file at the lowest free memory location, rounded to 1000h (e.g., 23000h, 24000h, etc.).

When the R command is executed, if there is an error, Debug68 displays either a question mark (pathname buffer empty) or an appropriate error message (file does not exist, program too big, etc.). If there is no error, the following message is displayed:

```
[FIRST] = xxxxxx
[NEXT]  = yyyyyy
[START] = zzzzzz
```

Where xxxxxx is the beginning address of the program loaded, yyyyyy is the first free location in the memory after the loaded program, and zzzzzz is the starting address of the program (where execution begins).

S or SM - SUBSTITUTE MEMORY

This command is used to substitute memory. The formats of this command are:

SM
SM starting-addr

Note that the M is optional. If the first format is used, the last substituted location (initially FIRST) will be the starting address.

Debug68 displays the absolute address, followed by the relative address, followed by the contents of the memory byte. One of the following may then be entered.

1. A data byte value, followed by a RETURN. The data byte value is stored at the address of the prompt. The address is then incremented by one and displayed on the next line.

2. A string enclosed between apostrophes (') followed by a RETURN. The string is stored beginning at the address of the prompt. The address is then incremented past the string and displayed on the next line.

3. Any number of 1 and 2, above, can be entered on one line with one RETURN terminating the line. The address is then incremented past the bytes that were stored and the new address is displayed on the next line.

4. A minus sign (-). A minus sign does not store a byte. The address will be decremented to the previous address. The minus sign can be used to back up to a previous location in case an error was made.

5. A RETURN only. If no entry is made on the line, the memory byte remains unchanged. The address is incremented by one and displayed on the next line.

6. A period(.). A period ends the input mode and returns control to the command level.

In the example that follows, assume that the @ register contains the value 28000h:

```
-> SM@100
029000 000100' 32 0
029001 000101' 17 00
029002 000102' 31 'this is an ASCII string'
029019 000119' 7A 'AAAA' 0 0 1 2 3 4 5 6 7 8 9
029028 000128' 22
029029 000129' 29
02902A 00012A' 87 -
029029 000129' 55
02902A 00012A' 87 .
```

SR - SUBSTITUTE REGISTER

The SR command alters the user registers. The format of the substitute registers command is:

SR
SR reg-name

The first form of the command allows the user to change the contents of the registers in the manner of a circular list. The list starts with register D0 through D7 and A0 through A7, followed by SR, PC, and USP. One register and its contents is displayed at a time. You can either change its contents by entering a new value, followed by a RETURN, or leave the register unchanged and skip to the next register in the list by entering a RETURN only, or backspace to the previous register in the list by entering a minus sign (-) followed by a RETURN. A period (.) ends the substitution mode and returns control to the command level.

The second form of the command can be used to substitute the contents of a single register specified by reg-name. The section SUMMARY OF REGISTER NAMES gives a summary of the names that can be substituted. An example of the first form of the command follows:

```
-> SR
D0 = 00000000   12
D1 = 00000000   34
D2 = 00000000   56
D3 = 00000000   78
D4 = 00000000   48567
D5 = 00000000   aaaa
D6 = 00000000   bbbb
D7 = 00000000   cccc
A0 = 00000000   dddd
A1 = 00000000   eeee
A2 = 00000000   ffff
A3 = 00000000   1000
A4 = 00000000   2000
A5 = 00000000   3000
A6 = 00000000   4000
A7 = 0004FFF4   5000
SR = 0000       1f
PC = 00033000   76
USP = 00004000  -
PC = 00000076   34000
USP = 00005000  (CR)
D0 = 00000012   .
```

Program area <u>must be</u> unprotected for SRPC to work after G with a breakpoint.

Note that only the whole register contents can be changed (i.e., all values are considered long words).

If no value is entered, or if an error occurs, the value of the register remains unchanged. In the following example, the D0 register is changed to contain 41h.

```
-> SR d0
D0 = 00123456   41
```

T - TRACE

The format of the Trace command is:

```
T
```

The trace command will perform tracing of one instruction. After every instruction is traced, the values of the registers are displayed.

A program can be traced through RAM or ROM. The trace command is performed by M68000 trace mode. First the user registers are loaded and the trace mode is turned on. After executing one instruction, the registers are resaved and displayed.

U - UNPROTECT MEMORY

The format of the Unprotect command is:

U
U beginning-addr ending-addr
U beginning-addr S swath-width

During the execution of Debug68 program, memory access is limited to the assigned user space in order to protect the rest of the system's memory from inadvertent changes that may cause the system to crash. Every attempt to access memory outside of the user space will cause an error to be displayed and the command to be ignored.

With the U command, the addresses of the beginning and the end of unprotected memory space can be displayed or modified for special debugging purposes.

The first form of the command displays the beginning and ending addresses of the unprotected memory space. The second form of the command changes the unprotected memory space to be in the range of beginning-addr to ending-addr. The third form of the command unprotects memory space from beginning-addr swath length.

V - VERIFY MEMORY

This command verifies that the block of memory between source address and source end contains the same value as the block beginning at the destination address. The addresses and contents are displayed for each discrepancy found. The format of the command is:

V source-addr source-end destination-addr
V source-addr S swath-width destination-addr

The command works by reading bytes from the source and destination and comparing them.

If a discrepancy is found, it is displayed in the following order: source address, source contents, destination contents, destination address. In the example that follows, the contents of memory locations 30003h and 31003h are not the same. The same is true of locations 30008h and 31008h.

```
-> V 30000 S30 31000
030003 32  12 031003
030008 7A  5A 031008
```

Z - ZAP MEMORY

Z beginning-addr ending-addr string-of-bytes
Z beginning-addr S swath-width string-of-bytes

This command is used to initialize a portion of memory. The memory from beginning-addr to ending-addr is initialized with the string of bytes repeated over and over. The string of bytes is in exactly the same format as in the S or SM command.

```
        -> Z 33100 S400 0   (zeros from 33100h to 334FFh) -> Z
        31000 S10 0 1 'ABC'
        -> D 31000 S10
031000  0001 4142  4300 0141   4243 0001  4142 4300  ..ABC..ABC..ABC.
```

Appendix A

SUMMARY OF DEBUG68 COMMANDS

The following is an alphabetical list of the Debug68 commands.

Command	Description
A	Assemble into memory
B	Set and display break points
BX	Delete break points
C	Trace over calls
D or DM	Display memory
DR	Display register
E	Examine input port
EX	Exit to the operating system
F	Specify disk filename
G	Go
H	Hexadecimal arithmetic
L	List in assembler mnemonics
M	Move memory
O	Output to data port
Q	Query memory
R	Read disk file
S or SM	Substitute memory
SR	Substitute register (refer to summary of register names)

T	Trace
V	Verify memory
U	Unprotect memory
Z	Zap memory

Appendix B

SUMMARY OF REGISTER NAMES

The following register names are displayed by the DR command and may be used with the SR command.

Register	Description
SR	Status register. The following flags may be changed. X - eXtend flag N - Negative flag Z - Zero flag V - oVerflow flag C - Carry flag The interrupt enable bits I1, I2, and I3 may also be changed.
D0	Data register 0
D1	Data register 1
D2	Data register 2
D3	Data register 3
D4	Data register 4
D5	Data register 5
D6	Data register 6
D7	Data register 7
A0	Address register 0
A1	Address register 1
A2	Address register 2
A3	Address register 3
A4	Address register 4
A5	Address register 5
A6	Address register 6
A7	Address register 7
PC	Program counter

Appendix C

USING DEBUG68 UNDER THE D-SERIES CROMIX OPERATING SYSTEM

After Debug68 is loaded into memory and begins execution, it checks the values of the trap and trace vectors. If the vectors point to the operating system, Debug68 replaces them with values pointing to its trace and trap handling routines. If the vectors point somewhere else, Debug68 assumes that some other user is already running Debug68 and aborts execution with the message:

```
Debugger already running.
```

To avoid the possibility of leaving the trap and trace vectors pointing to the wrong place after finishing a debugging session, use the Ex command to exit to the operating system. By doing so, the trap and trace vectors will be restored to the original values.

If Debug68 is aborted by pressing CONTROL-C, the trap and trace vectors will not be restored. To restore the original values of both vectors run the program Restrp.

The suggested way of calling Debug68 is to use a command file **deb.cmd**, which is supplied with the Debug68 program. If a debugged program inadvertently exits to the operating system, the Restrp program executes and it restores the trap values.

~~ah~~ after logging in

type "System"

Then you [illegible] type "[illegible]" to see [illegible] [illegible] [illegible] etc.

all the [illegible] [illegible] [illegible] in directory SWI

[illegible] to it

type "D /SWI"

then L will [illegible] all files

[illegible] — to debug something say debug68 *.bin

— to create or edit [illegible] say Ed *. Asm

— to Assemble say Asm68 *.[illegible]

then Link68 * ← no need for extension [illegible] [illegible]

frame [illegible] [illegible] [illegible]

```
OPT_DEFAULT   EQU   ^0|^2|^4
   BRA_S      EQU   8
```

[illegible] OPT [illegible]

[illegible] [illegible] [illegible]

```
 _EMIT     EQU   46H
```

```
 OPT       OPT_DEFAULT|^BRA_S
```

HIS LOVE IS ETERNAL

CROMIX

I) To put floppy disk B online, you must

a) have a directory called "b" at top level on disk A. If not, use

CREATE a b to make it

b) type MOUNT FDB b to mount disk.

c) remember to UNMOUNT FDB b before power down.

[illegible] 7/22/85 [illegible]

MOVE [illegible] to [illegible] columns

Debug67

[illegible] 045000

& can go to [illegible]

→ 970000 [illegible] to put program [illegible].

```
[illegible] LEA      ΦFFFΦEHL, AΦ          memory [illegible]
         MOVE.B     D1, (AΦ)
         LEA      #FFFΦΦ1L, A1
         MOVE.B   (A1), DΦ
         ANDI.B   #7FH, DΦ
         CMPI.B   #Φ1H, DΦ
         BNE.S    7Φ0ΦΦH
7ΦΦ[illegible] NOP
         BRA.S    7ΦΦΦΦH [illegible], return
```

Set breakpoint at 70[illegible]A, [illegible] with desired [illegible] byte, & G700[illegible] to [illegible].

[illegible] ^A [illegible] on terminal to cause break, & G to resume.

[illegible] ^C [illegible] RESTRP if done.

[illegible] 1:

```
7ΦΦΦΦ LEA      ΦB[illegible]
      MOVE.B   #[illegible]
      [illegible]   D[illegible]
      MOVE.B   (A[illegible]
      ADDQ.L   #1, A[illegible]
      MOVE.B   #[illegible] DΦ
      MOVE.B   D[illegible]
      MOVE.B   ([illegible]
      BRA[illegible]
```