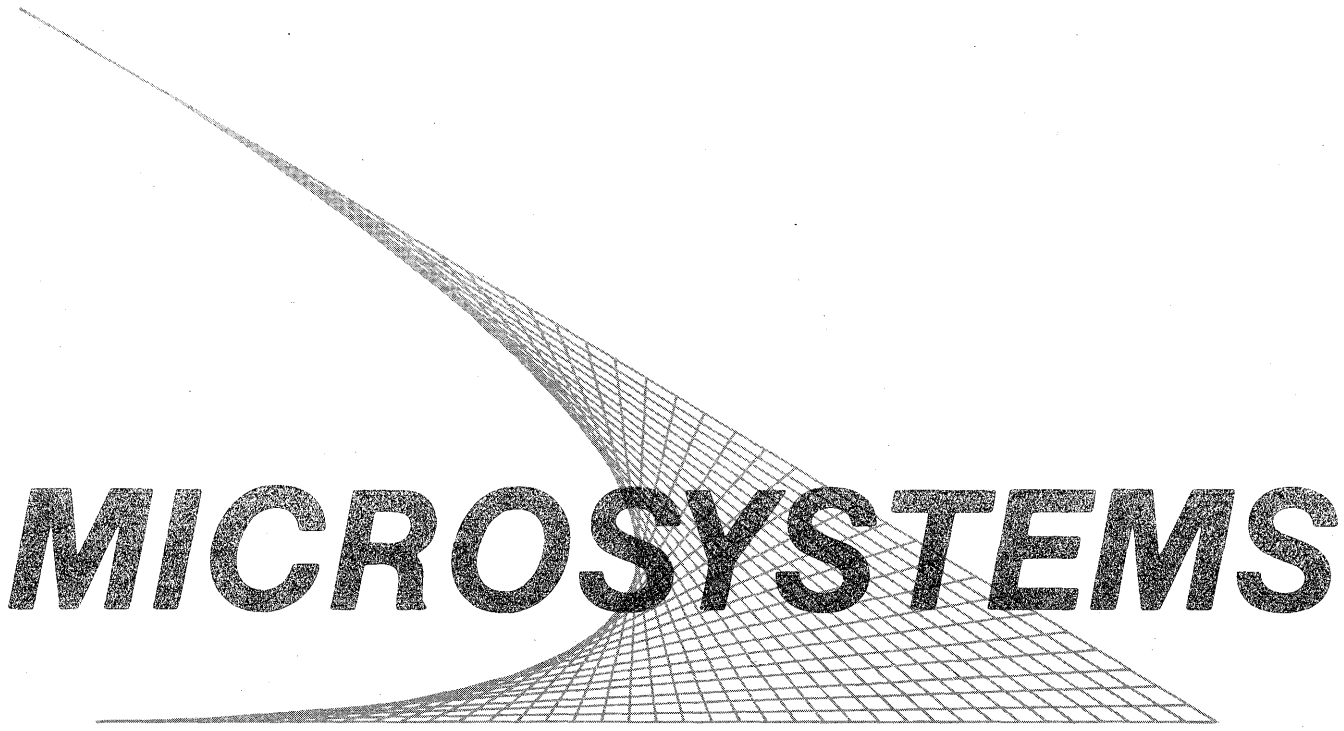
MOTOROLA

M68KTENBG/D2

TENbug Debugging Package
User's Manual



QUALITY • PEOPLE • PERFORMANCE

M68KTENBG/D2

MAY 1984

TENbug DEBUGGING PACKAGE

USER'S MANUAL

Second Edition

First Edition September 1983

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

This manual describes the debugging monitor TENbug as it is used in the VME/10 Microcomputer System, hereafter referred to as the VME/10.

1.2 DEFINITION OF TENbug

TENbug is the resident firmware debugging package for the VME/10. The 32K-byte firmware (stored in ROM or EPROM devices) provides a self-contained programming and operating environment. TENbug interacts with the user through predefined commands that are entered via the terminal. The commands fall into five general categories:

a. Commands which allow the user to display or modify memory.

b. Commands which allow the user to display or modify the various internal registers of the MC68010.

c. Commands which allow the user to execute a program under various levels of control.

d. Commands which control access to the various input/output resources on the board.

e. Commands which allow the user to select and test video features and graphics resolution.

An additional function called the TRAP #15 I/O handler allows the user program to utilize various routines within TENbug. The TRAP #15 handler is discussed in Chapter 6.

The operational mode of TENbug is described in Chapter 2.

1.3 TENbug INTERNAL STRUCTURE

1.3.1 Memory Map

The following abbreviated memory map for the VME/10 highlights addresses that might be of particular interest to TENbug users. Refer to the VME/10 Microcomputer System Reference Manual for a complete description of the memory maps for both high- and low-resolution graphics modes.

Note that addresses are assumed to be hexadecimal throughout this manual. In text, numbers may be preceded with a dollar sign ($) for identification as hexadecimal.

RAM LOCATION	FUNCTION
0-3FF	Vectors
400-AFF	Work area and stack for TENbug

SPECIAL LOCATIONS	FUNCTION
F00000-F00007	Area containing initial values for supervisor stack pointer, program counter, and vector base register after cold start
F14000-F14FFF	Area used to define programmable "soft" character set

I/O LOCATION	FUNCTION
F1C1C9	Serial port 2 (host), serial I/O card (optional)
F1C1CB	Serial port 3 (host), serial I/O card (optional)
F1C1E1	Parallel port 1 (printer), parallel I/O card (optional)
F1C1E9	Parallel port 2 (printer), parallel I/O card (optional)
F1C0D1	Base address of RWIN1 Disk Controller

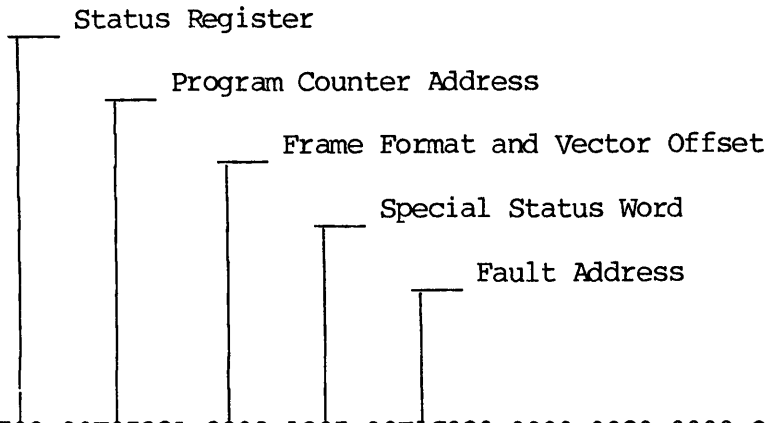
1.3.2 Vectors and Errors

TENbug shares resources with the target program under test -- that is, each affected resource can be used only by TENbug or the target program at any given time.

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle the exception. These vectors are initialized by TENbug in default memory locations 0 through $3FF during a cold- or warm-start sequence (see Chapter 2). If the target program uses any of these locations, the user values must be rewritten following each cold or warm start. If the target program uses any of the following locations, the associated function will be lost to TENbug.

MEMORY LOCATION	TENbug FUNCTION
10-13	Breakpoints (illegal instructions)
24-27	Trace
BC-BF	TRAP #15 user calls to TENbug
138-13B	ABORT pushbutton switch on VME/10 operator panel (refer to Appendix A)

The vectors with default memory locations of $80 through $3FF cause a ???? ERROR TRAP message to be displayed on the console terminal. In addition, several of the vectors cause display of appropriate information. (Refer to Appendix B for a list of error messages.) BUS and ADDR error traps also cause display of the exception status from the stack, in hexadecimal characters, as shown in the following example.

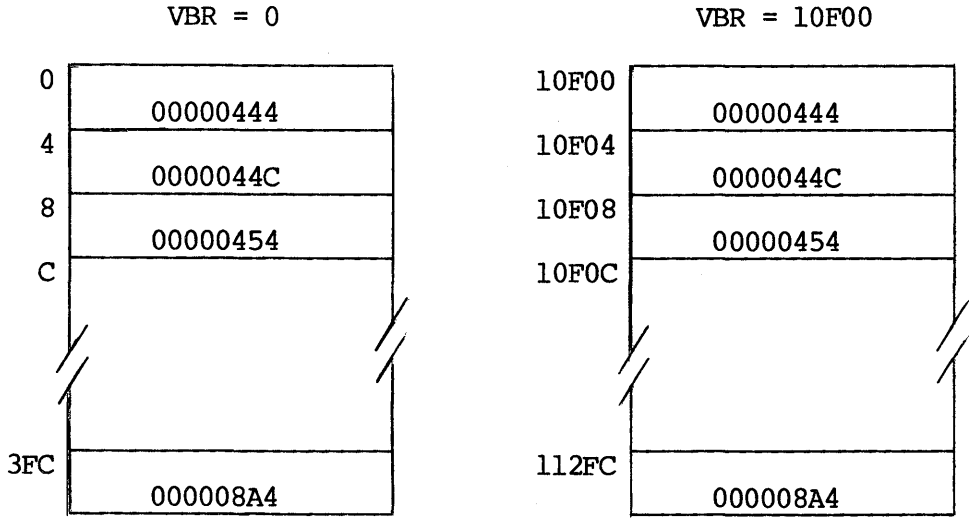


For additional information on this display, refer to the bus error, address error, and the reference classification descriptions in the exception processing chapter of the M68000 16/32-Bit Microprocessor Programmer's Reference Manual.

1.3.2.1 Resetting Vector Base Register. The MC68010 processor upon which the VME/10 is based features a Vector Base Register (VBR) which contains the base (starting) address for the VME/10 exception vectors. Exception vectors are located in memory addresses 0 through $3FF relative to the VBR. Upon reset (cold or warm start) of the MC68010, the value of the VBR is set to zero.

TENbug must have control of the exception vectors to function properly. If the user sets the VBR to a value other than its default value of zero, he must also establish a new set of exception vector memory locations for the VBR value. In other words, the user must copy all existing vector memory locations to the same relative location in the new VBR table.

In the following example, the VBR value is changed from 0 to 10F00. Exception vector memory locations must also be copied to this new location. Note that the content of each vector memory location (i.e., the appropriate routine address) remains the same.



1.3.3 Disk I/O

TENbug provides limited support of disk I/O through a Winchester Disk Controller. The commands supported are BH, BO, IOC, IOP, and IOT. Each of these commands does a read of the volume ID found on sector 0 of a disk.

NOTE

A sector is 256 bytes. The disk controller maps physical sectors on various disks into virtual 256-byte sectors at the controller interface.

The first 256 bytes of the media are the volume ID. Bytes $F8-$FF of the volume ID must contain either the ASCII character string "EXORMACS" or "MOTOROLA"; otherwise, an error message will result. For more information on interpreting the data displayed, see the Winchester Disk Controller User's Manual.

The other information used from the volume ID is:

BYTES	USED FOR
$14-$17	Starting sector address of program to be loaded (via BH, BO).
$18-$19	Number of 256-byte sectors to be loaded.
$1E-$21	Load address (first destination memory byte).
$90-$93	Sector address of media configuration parameters (refer to Appendix C).
$94	Length of configuration area (usually one 256-byte sector).

1.4 TENbug WITH SYSTEM V/68

The following paragraphs list information specific to the use of TENbug with SYSTEM V/68.

1.4.1 Operational Commands

In the following list, commands given in parentheses indicate the key that is to be pressed. Commands not given in parentheses are to be typed as shown.

BH	Boots the operating system from the fixed disk and halts.
BO	Boots the operating system from the fixed disk and gives control to the program loaded.
(BREAK)	Aborts command.
(DEL)	Deletes character.
(CTRL-D)	Redisplays line.
(CTRL-H)	Deletes character.
(CTRL-W)	Suspends output; any character continues output.
(CTRL-X)	Cancels command line.

1.4.2 Debugging Commands

The following commands may be useful for debugging, but should be used only in single-user mode after sync has executed. Use of these commands may result in the need for system reboot.

.A0-.A7	BARS, NOBARS *	HE
.D0-.D7	BF	IOC
.DFC	BM	IOP
.PC	BR, NOBR	IOT
.R0-.R6	BS	MD
.SFC	CH, NOCH	MM
.SR	CRT	MS
.SSP	CS	OF
.USP	DC	PA, NOPA
.VBR	DF	TR
	GD	TT
	GO	
	GR, NOGR	
	GT	

* This command modifies graphics memory and should be used only with an operating system configured to support graphics.

1.4.3 Non-Applicable Commands

The following commands should be used in a stand-alone mode; they should not be used with SYSTEM V/68.

BI	PF
BT	TM
DU	VE
LO	

1.5 REFERENCE MANUALS

Refer to the following documents for more information on the environments in which TENbug is used.

VME/10 Microcomputer System Overview Manual, M68KVSOM

VME/10 Microcomputer System Diagnostics Manual, M68KVSDM

VME/10 Microcomputer System Reference Manual, M68KVSREF

VERSAdos to VME Hardware and Software Configuration User's Manual, MVMEDOS

Winchester Disk Controller User's Manual, M68RWIN1

MVME400 Dual RS-232C Serial Port Module User's Manual, MVME400

MVME410 Dual 16-Bit Parallel Port Module User's Manual, MVME410

M68000 16/32-Bit Microprocessor Programmer's Reference Manual, M68000UM

CHAPTER 2

TENbug OPERATING PROCEDURE

2.1 INTRODUCTION

The following procedures enable the user to enter TENbug. For information on system installation, self-test diagnostic programs, and operating system initialization, refer to the VME/10 Microcomputer System Overview and Diagnostics manuals.

2.2 CHASSIS CONTROL SWITCHES

Before attempting to initiate TENbug, the user should be familiar with the operator panel located at the bottom left corner on the front of the VME/10 chassis. This panel contains the following control switches which are supported by TENbug. Use of these switches is described in paragraphs 2.5 through 2.7.

a. [0 | 1] - The amber-colored power on/off rocker-arm switch is used to turn on power to the VME/10 and initiate the power-up/reset self-test (PWRT). When the 0 side is pressed, power is off; when the 1 side is pressed, power is on.

b. KYBD LOCK - The KYBD LOCK key switch controls a bit in a register which is monitored by TENbug. When the key switch is in the locked (vertical) position, VME/10 performs an automatic BO command from device 0 (this usually starts the operating system). When the key switch is in the unlocked (horizontal) position, VME/10 enters TENbug. Also, when the key switch is in the locked position, the front panel pushbutton switches RESET and ABORT, as well as the keyboard, are inoperative. This feature provides protection from inadvertent panel interrupts during system usage.

c. RESET - When this momentary-action pushbutton switch is pressed, it resets the VME/10 logic circuits. If the VME/10 is in the operating system, TENbug is entered by pressing RESET (provided the KYBD LOCK key switch is in the unlocked position). Because pressing RESET can cause indeterminate results, read the warm start description in paragraph 2.7 before using this switch.

d. ABORT - When this momentary-action pushbutton switch is pressed (provided the KYBD LOCK switch is in the unlocked position), the VME/10 enters TENbug, but the VME/10 logic circuits are not reset. After an abort, the user can enter the character G to continue execution of the current program prior to the abort. Appendix A describes what occurs when the ABORT switch is pressed.

e. RESET and ABORT - These buttons may be used in combination to accomplish the same thing as the on/off switch (item a.) without cycling power. This simulated cold-start sequence is described in paragraph 2.6.

2.3 TERMINAL CONTROL CHARACTERS

Several keys are used as command line edit and control functions. The user should be familiar with these functions before using TENbug. The functions include:

a. DEL key or CTRL H - will delete the last character entered on terminal.

b. CTRL X - will cancel the entire line.

c. CTRL D - will redisplay the entire line.

d. <--| (carriage return) - will enter the command line and cause processing to begin.

e. CTRL W - will suspend system output to the terminal. To resume output to the terminal, any character can be entered.

f. BREAK - will abort commands that do any console I/O and return to the input routine.

For characters requiring the control key (CTRL), the CTRL should be pressed and held down, and then the other key (H, X, D, or W) should be pressed.

2.4 HEADER J2

The configuration of pins 5 and 6 on header J2, located inside the VME/10 chassis, determines whether the power-up reset (PWRT) self-test is performed upon system initialization. It also allows generation or suppression of the "Booting from ROM: xxxx" message at the close of the ROMBOOT procedure (refer to paragraph 2.10). When a jumper is placed on pins 5 and 6 of J2, as in initial VME/10 factory configuration, the PWRT self-test is performed during the cold-start and warm-start sequences described in the following paragraphs. This jumper also allows display of the ROMBOOT message when control is passed to TENbug. When the jumper is removed from pins 5 and 6 of J2, no PWRT self-test is performed and the ROMBOOT message is suppressed.

2.5 ENTERING TENbug DURING SYSTEM POWER-UP (COLD START)

Invoking TENbug using the cold-start technique causes the contents of all memory to be destroyed. It also causes the VME/10 system to place the contents of addresses $F00000-$F00003 into the supervisor stack, and the contents of $F00004-$F00007 into the program counter. These addresses are located in system ROM. Figure 2-1 illustrates a flow diagram of the VME/10 cold-start procedure. The following paragraphs assume that a jumper is present on pins 5 and 6 of J2.

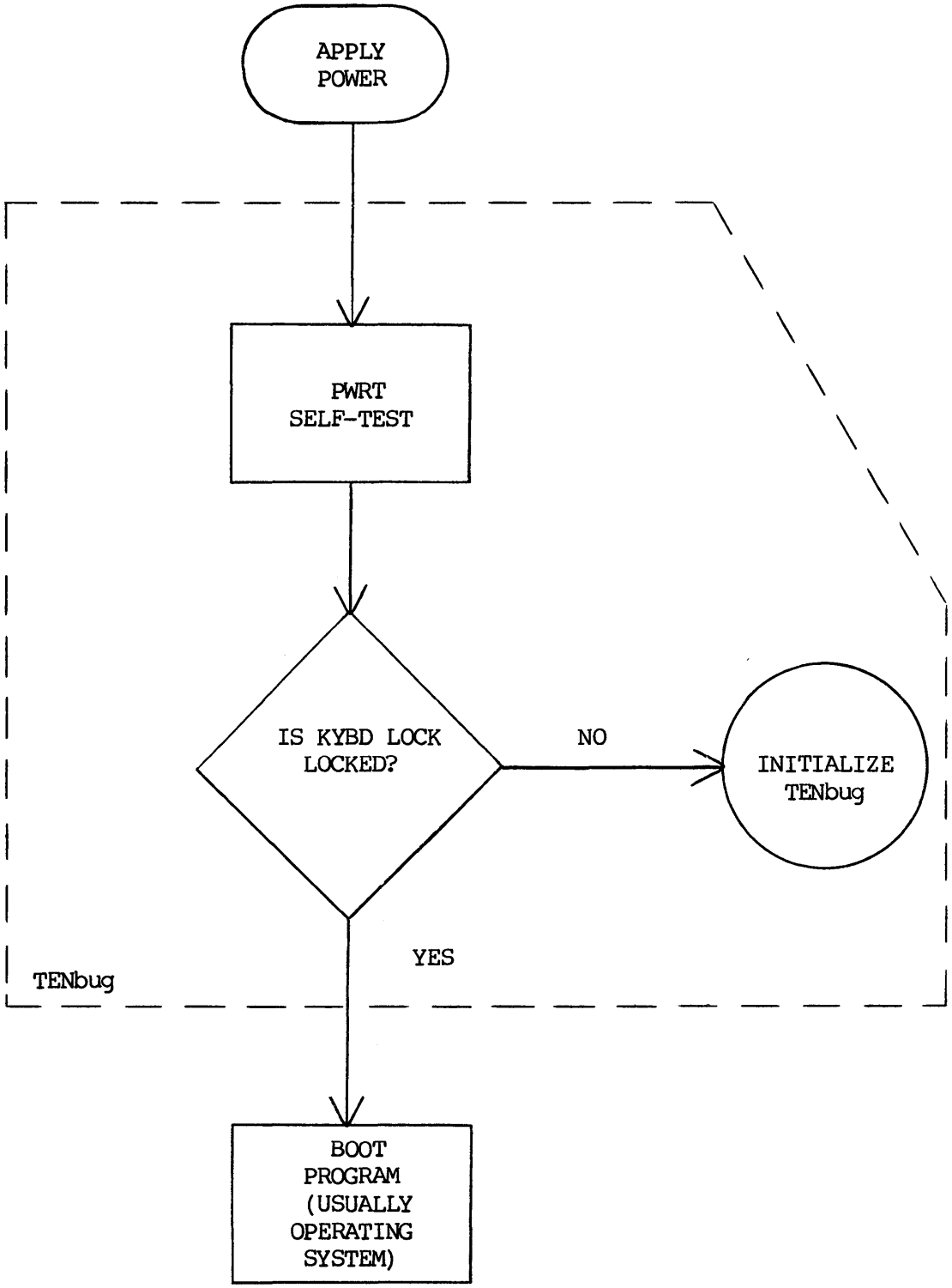


FIGURE 2-1. Flow Diagram of VME/10 Cold Start

2.5.1 Cold Start without MVME400 Module

This method allows the user to enter TENbug during system power up when no MVME400 (Dual RS-232C Serial Port) module is present in the VME card cage.

a. Set the KYBD LOCK key switch on the operator panel to the unlocked position.

b. Apply power to chassis. When power is applied, the PWRT self-test is initiated.

c. If PWRT self-test indicates no errors, the TENbug prompt and version number will appear on the screen:

```
TENbug 2.x >
```

2.5.2 Cold Start with MVME400 Module

This method allows the user to enter TENbug during system power up when an MVME400 module is present in the VME card cage.

a. Set the KYBD LOCK key switch on the operator panel to the unlocked position.

b. Apply power to chassis. When power is applied, the PWRT self-test is initiated.

c. If PWRT self-test indicates no errors, the firmware displays a prompt without a version number.

```
TENbug >
```

It then awaits input from the first device to be used, which will be the console terminal.

d. Select the terminal to serve as the console keyboard. This device will remain the console device until the VME/10 is restarted with a warm- or cold-start procedure.

e. Press the carriage return key on the chosen keyboard to obtain the complete TENbug prompt with version number.

```
TENbug 2.x >
```

2.6 ENTERING TENbug VIA SIMULATED COLD START

A cold-start sequence (the equivalent of turning the power off and on) can be simulated when the KYBD LOCK switch is set to the unlocked position. Use the RESET and ABORT buttons as follows:

a. Press and hold RESET button.

b. Press and release ABORT button.

c. Release RESET button.

d. When an MVME400 module is not present in the VME card cage, go to step c. of paragraph 2.5.1.

e. When an MVME400 module is present in the VME card cage, go to step c. of paragraph 2.5.2.

Like the true cold-start sequence, this method will erase all memory contents and will execute the PWRT self-test. It will also place the contents of ROM addresses $F00000-$F00003 into the supervisor stack, and the contents of $F00004-$F00007 into the program counter. In other words, it translates the ROM at $F00000 to location $000000, so that the RAM at location 0 is mapped out of the system.

2.7 ENTERING TENbug WITHOUT DESTROYING MEMORY CONTENTS (WARM START)

This method allows the user to enter TENbug without destroying the contents of the VME/10 memory. However, using the warm-start sequence (pressing RESET only) causes the VME/10 to place the contents of RAM addresses $0 through $3 into the supervisor stack, and the contents of $4 through $7 into the program counter. It also sets the processor to supervisor state.

CAUTION

BECAUSE THESE ADDRESSES ARE LOCATED IN RAM, THE USER CAN OVERLAY ANY DATA OR ADDRESS INTO THESE REGISTERS, IN WHICH CASE RESULTS ARE INDETERMINATE.

a. Set the KYBD LOCK key switch to the unlocked position.

b. Press the RESET button on the operator panel.

c. When an MVME400 module is not present in the VME card cage, go to step c. of paragraph 2.5.1.

d. When an MVME400 module is present in the VME card cage, go to step c. of paragraph 2.5.2.

2.8 TENbug COMMAND OPERATION

After TENbug initialization, the computer waits for a command line input from the console terminal. A standard input routine controls the system while the user types a line of input. Command processing begins only after the line has been entered, followed by a carriage return. When a proper command is entered, the operation continues in one of two basic modes. If the command causes execution of a user program, the TENbug firmware may or may not be reentered, depending on the discretion of the user. For the alternate case, the command will be executed under control of the TENbug condition. During command execution, additional user input may be required, depending on the command function.

Figure 2-2 illustrates the VME/10 operational mode.

NOTE

If a command causes the system to access an unused address (i.e., no memory or peripheral devices are located at that address), a bus trap error will occur. Unless default vectors have been overwritten, the terminal displays a trap error message and the contents of all MC68010 registers. Control is then returned to the TENbug monitor. A bus trap error also occurs if the system attempts to write to ROM.

2.9 WHEN TENbug PROMPT FAILS TO APPEAR

Refer to Chapter 2 of the VME/10 Microcomputer System Diagnostics Manual for instructions if the PWRT sequence fails and/or no TENbug prompt appears during one of the procedures listed in this chapter.

2.10 ROMBOOT FACILITY

When the VME/10 completes its preliminary initialization, pins 5 and 6 of header J2 are checked to determine whether the PWRT self-test should be executed. If not, TENbug receives control immediately; if so, it receives control after execution of the self-test. After control is passed to TENbug, a routine in ROM can be executed (if the ROM meets the format requirements). This feature, which provides the ability to transfer control to an external ROM routine at power up or cold start, is named ROMBOOT.

A module requiring the use of ROMBOOT linkage must meet the following three requirements:

a. The routine must be located in the VME/10 memory map between addresses $180000 to $FFE000.

b. The ASCII string "BOOT", followed by some linkage convention information, must be located on an 8K boundary within the memory range.

c. The routine must pass a checksum test applied from the first to the last byte of the module.

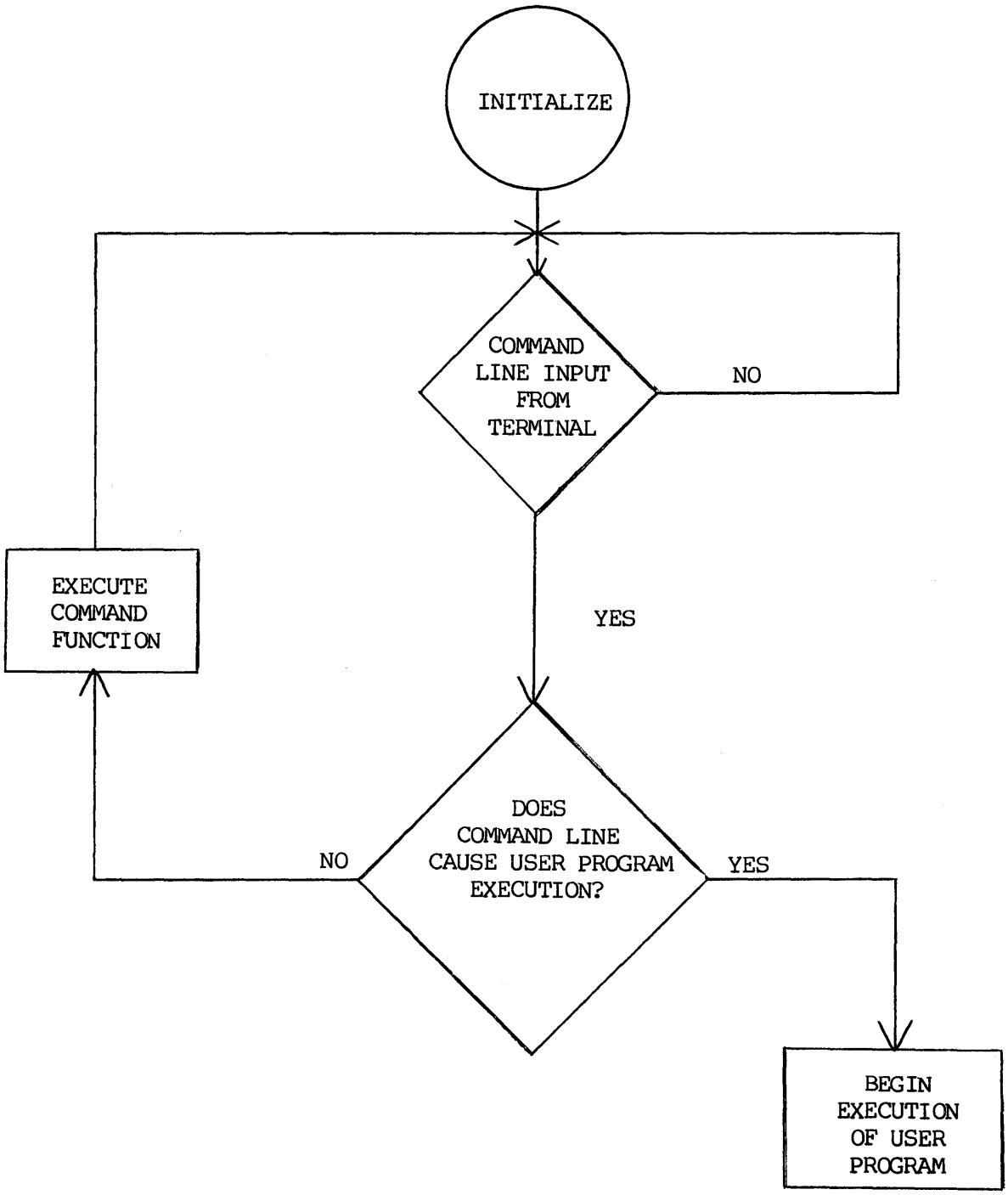


FIGURE 2-2. Flow Diagram of TENbug Operational Mode

NOTE

There is no requirement that the routine reside in ROM; it can be loaded into a RAM module and then invoked by a cold start.

To prepare a module for ROMBOOT, the CS command must be used. When the module is ready it can be loaded into RAM, and the checksum generated, installed, and verified with the CS command. (Refer to the Checksum command description and examples.)

The format of the beginning of the routine is as follows:

MODULE OFFSET	LENGTH	CONTENTS	DESCRIPTION
$00	4	BOOT	ASCII string indicating possible routine; checksum must be zero, too.
$04	4	Entry address	Longword offset from 8K boundary.
$08	4	Routine Length	Longword, includes "BOOT to end".
$0C	?	Routine name	ASCII string containing routine name (only four bytes displayed).

By convention within Motorola, the last three bytes of ROM contain the firmware version number, checksum, and socket number. In this environment, the length would contain the ASCII string "BOOT" (that was on the 8K boundary), through and including the socket number; however, the user wishing to make use of ROMBOOT does not have to fill a complete ROM. Any partial amount will be accepted, as long as the length reflects where the checksum will be correct.

The sequence used to validate a routine for execution begins at the high limit of memory first and checks for the "BOOT" indicator. Three events are of interest for any location being tested.

a. If there is no memory at that location, a bus error is generated. The ROMBOOT routine is required to move on to the next 8K boundary.

b. If memory is present, but the first four bytes do not contain BOOT, the ROMBOOT routine is required to move on to the next 8K boundary.

c. If the ASCII string "BOOT" is located on an 8K boundary, we are not assured that the routine is really one meant to gain control at power up or cold start. To verify that this is the case, the bytes starting from the 8K boundary through the end of the routine (as defined by the 8K boundary + 4-byte length at offset $8) are run through the self-test checksum routine. If both the even and odd bytes are zero, chances are very good that the routine was meant to be used for ROMBOOT.

The bus error routine address is replaced at location $8 before control is passed to the routine at the point specified within the header (8K boundary + contents of offset $4). A JSR instruction which loads the address of the next instruction within the ROMBOOT routine on the stack allows the routine to return control to TENbug following some temporary task such as initialization.

In most cases, right before control is actually given to the ROM routine, a message displaying the first four bytes of the routine name (8K boundary + contents of $C) is placed on the terminal in the following form:

```
Booting from ROM: xxxx
```

where xxxx are the first four bytes of the name.

If pins 5 and 6 of header J2 are not jumpered, the message is suppressed and the PWRT self-test programs are not run. This might be desirable if, for example, no CRT or disk was present on a system.

The following example returns control to TENbug 2.x after placing a test pattern in a portion of RAM. Notice the use of the CS command to calculate and verify the checksum.

SAMPLE ROMBOOT ROUTINE - Procedure for preparing checksum

Load ROMBOOT routine in RAM to generate checksum.

```
TENbug 2.x > MD 0+R3 40
000000+R3 42 4F 4F 54 00 00 00 14  00 00 00 A6 54 65 73 74  BOOT.......&Test
000010+R3 41 F9 00 01 F0 00 20 3C  00 00 EF FF 11 00 51 C8  Ay..p. <..o...QH
000020+R3 FF FC 4E 75 01 01 00 00  10 08 FF FF FF FF FF FF  .|Nu............
000030+R3 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
```

Display (in hex) contents of RAM containing the routine.

```
TENbug 2.x > M 10+R3;DI
000010+R3 41F90001F000          LEA.L   $0001F000,A0 ? (CR)
000016+R3 203C0000EFFF          MOVE.L  #61439,D0 ? (CR)
00001C+R3 1100                  MOVE.B  D0,-(A0) ? (CR)
00001E+R3 51C8FFFC              DBF.L   D0,$02001C? (CR)
000022+R3 4E75                  RTS      ? (CR)
000024+R3 0101                  BTST    D0,D1 ? (CR)
000026+R3 0000                  DC.W    $0000 ? (CR)

000028+R3 1008                  DC.W    $1008 ? (CR)
00002A+R3 FFFF                  DC.W    $FFFF ? (CR)
00002C+R3 FFFF                  DC.W    $FFFF ? (CR)
00002E+R3 FFFF                  DC.W    $FFFF ? (CR)
000030+R3 FFFF                  DC.W    $FFFF ? .
```

Display same memory using disassembler/assembler. This small routine loads a test pattern into RAM and returns to TENbug 2.x.

0101 is revision number of routine.
0000 is value to be replaced by checksum.
1008 are socket ID's U16 and U108.

NOTE: The socket ID's are the last two bytes of the routine.

```
TENbug 2.x > CS 0+R3 2A+R3
PHYSICAL ADDRESS=00020000 0002002A
      (EVEN ODD)=4B34
```

Calculate checksum from byte 0 to byte 2A (end of routine +1). Note that location where checksum is to be placed must be $0000 to produce correct checksum bytes.

```
TENbug 2.x > M 26+R3;W
000026+R3 0000 ? 4B34.
```

Enter calculated checksums with MM command.

```
TENbug 2.x > CS
PHYSICAL ADDRESS=00020000 0002002A
      (EVEN ODD)=0000
```

Issue CS command to verify zeros are produced. Notice the operands from the previous CS command are retained for verification.

SAMPLE ROMBOOT ROUTINE - Procedure for preparing checksum (cont'd)

```
TENbug 2.x > MD 0+R3 40
000000+R3 42 4F 4F 54 00 00 00 14  00 00 00 A6 54 65 73 74  BOOT.......&Test
000010+R3 41 F9 00 01 F0 00 20 3C  00 00 EF FF 11 00 51 C8  Ay..p. <..o...QH
000020+R3 FF FC 4E 75 01 01 4B 34  10 08 FF FF FF FF FF FF  .|Nu..K4........
000030+R3 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................

TENbug 2.x >
```

Display memory one more time with checksum in place.

2.11 OFFLINE KEYBOARD COMMAND

Pressing the SEL key places the VME/10 keyboard and local CRT display into local mode (offline to TENbug commands). Then uppercase, lowercase, and special characters are displayed on the local VME/10 CRT display when they are entered on the keyboard. To exit local mode, press the SEL key again; TENbug will resume control of the keyboard and CRT display.

Five function keys provide support for the character attributes as follows:

F1	Character blink
F2	Character underline
F3	Character inverse video
F4	Character protect
F5	Character color

Functions controlled by keys F1 through F4 are enabled by pressing the appropriate function key, whereas pressing SHIFT and the same function key disables the function. The character-color function increments through eight colors or shades of green each time the F5 key is pressed.

The offline feature is particularly useful in building sample screens for applications under development.

CHAPTER 3

COMMAND LINE FORMAT

3.1 INTRODUCTION

Commands are entered in buffer-organized fashion. A standard input routine controls the system while the user types a line of input. Processing begins only after the carriage return has been entered.

Many primitive commands can be altered by the options field. This provides the user several extensions to the primitive commands.

Several commands are set and reset pairs; i.e., rather than having two primitive commands, the form NO is added as the first two characters of the command. For example, the set breakpoint command is BR, and the reset breakpoint command is NOBR.

Command line formats are presented in a modified Backus-Naur Form (BNF). Certain symbols in the syntax may be used, where noted, in the real I/O. Others are metasymbols, which are used for definition only and are not entered by the user. These metasymbols and their meanings are as follows:

< >	Angular brackets enclose a symbol, known as a syntactic variable, that is replaced in a command line by one of a class of symbols it represents.
|	This symbol indicates that a choice is to be made. One of several symbols, separated by this symbol, should be selected.
[ ]	Square brackets enclose a symbol that is optional. The enclosed symbol may occur zero or one time.
[ ]...	Square brackets followed by periods enclose a symbol that is optional/repetitive. The symbol may appear zero or more times.

In the examples given in the following paragraphs, operator entries are shown underscored for clarity only -- i.e., the underscore is not to be typed. Operator entries are followed by a carriage return unless otherwise specified. The carriage return is not shown in examples except where it is the only entry, in which case it is shown as (CR).

3.2 TENbug COMMAND LINE FORMAT

The format of the TENbug command line is:

TENbug 2.x > [NO]<command>[<port number>] [<parameters>] [;<options>]

where:

TENBUG 2.x >	Is the basic TENbug prompt. For prompt variations, see appropriate command descriptions.
NO	Is the negative form (opposite) of primitive command.
command	Is the primitive command.
port number	Specifies the applicable device port.
parameters	Can be of the form <expression> or <address> and are usually separated by spaces.
options	Specifies applicable options; multiple options may be selected.

The basic command form consists of the primitive command field and the parameters field, although some primitives do not require parameters. Some primitive commands allow specification of alternate device ports. The additional command negation and options field can modify the primitive command.

If an option exists for a command, a semicolon (;) plus <options> field(s) are added to the command. Thus, several extensions can be provided to the user.

3.2.1 Expression as a Parameter

An <expression> can be one or more numeric values separated by the arithmetic operators plus (+) or minus (-). Numbers are assumed hexadecimal except for those preceded by an ampersand (&), which are decimal. In the assembler, numbers are assumed decimal unless preceded by a dollar sign ($).

3.2.2 Address as a Parameter

Many commands use <address> as a parameter. The syntax accepted by TENbug is the same as that accepted by the assembler, plus a memory indirect mode. Also, contained within TENbug are eight offset registers designated R0 through R7. These registers are software registers only, and are provided for easier debugging of relocatable code.

3.2.2.1 Address Formats.

FORMAT	EXAMPLE	DESCRIPTON
expression	140	Absolute address (offset register zero is added).
expression+offset	130+R5	Absolute address plus offset register five (not an assembler-accepted syntax).
expression+offset	150+R7	Absolute address (offset register seven is always zero; not an assembler-accepted syntax).
(A@)	(A5)	Address register indirect.
(A@,D@) (A@,A@)	(A6,D4)	Address register indirect with index.
expression(A@)	120(A3)	Register indirect with displacement.
expression(A@,D@) expression(A@,A@)	110(A2,D1)	Address register indirect with index plus displacement.
[expression]	[100]	Memory indirect (not an assembler-accepted syntax).

3.2.2.2 Offset Registers. Eight software registers (not actually hardware configured) are used to modify addresses contained in TENbug commands. The first seven registers (.R0-.R6) are used as general-purpose offsets, while .R7 (the eighth register) is always zero. The contents of the registers can be displayed by the offset command (OF), paragraph 4.2.29, and modified by the .<register> command, paragraph 4.2.1.

The offset registers are always reset to zero at power up. Thus, if their contents are not changed, the registers will have no effect on the entered address.

Unless another offset is entered, each command that expects an address parameter automatically adds offset R0 to the entered address -- that is, if R0 = 1000, the following commands are the same:

```
BR   10          (10 + 1000)       R0 is added by default
BR   10+R0       (10 + 1000)
BR   1010+R7     (1010 + 0)        R7 is always zero
```

The physical address for each of these commands is 1010.

Offset R0 is automatically added to the offset registers any time they are modified. The only exception to this is when another offset register is specifically added. Offset registers may be set to zero by adding R7 (always zero) to zero.

EXAMPLE:

```
.R0  0+R7      (R0 = 0 + 0 = 0)          R0 set to zero
.R1   8        (R1 = 8 + 0 = 8)          Offset R0 is zero, R1 is set to 8
.R0  100       (R0 = 100 + 0 = 100)      Offset R0 added
.R0  200       (R0 = 200 + 100 = 300)    Offset R0 added
.R3  100+R1    (R3 = 100 + 8 = 108)      Offset R0 not added
.R0  0+R7      (R0 = 0 + 0 = 0)          R0 set to zero
```

3.3 COMMAND VERIFICATION

As an aid to the user, TENbug displays for most commands its interpretation of the values entered as expression and address parameters. The results are displayed in either physical or logical format, depending upon the command entered.

EXAMPLES:

```
TENbug x.y > .R0 1000
TENbug x.y > .PC 0
```

Logical Format Example

```
TENbug x.y > MD 0
000000+R0 4E 71 4E 71 4E 71 4E 71  4E 71 00 00 0F 90 00 00   NqNqNqNqNq......
```

Physical Format Example

```
TENbug x.y > GT 8
PHYSICAL ADDRESS=00001008
PHYSICAL ADDRESS=00001000

AT BREAKPOINT
PC=00001008 SR=2700=.S7..... USP=00012C5C SSP=0000085E VBR=00000000 SFC=1 DFC=0
D0-7 00304E71 00001000 4E711000 00000000 00004E71 0000002C 00001008 00000000
A0-7 000004DA 00000000 00001000 0000053A 00001002 00000551 00000551 0000085E
 PC=000008+R0 4E71                  NOP
```

Commands entered are also checked for validity. For example, specifying an address parameter which would result in an error may cause the message INVALID ADDRESS=xxxxxxxx to be displayed on the console terminal. A table of TENbug error messages is provided in Appendix B.

CHAPTER 4

COMMAND SET

4.1 INTRODUCTION

Chapter 4 describes the command line syntax and provides one or more examples for each command in the TENbug command set. Table 4-1 lists TENbug command mnemonics by type. For SYSTEM V/68-specific information about TENbug commands, refer to paragraph 1.4.

TABLE 4-1. TENbug Commands by Type

COMMAND MNEMONIC	DESCRIPTION
MD	Memory display/disassembly
MM	Memory modify/disassembly/assembly
MS	Memory set
.A0-.A7	Display/set address register
.D0-.D7	Display/set data register
.DFC	Display/set destination function code
.PC	Display/set program counter
.SFC	Display/set source function code
.SR	Display/set status register
.SSP	Display/set supervisor stack pointer
.US	Display/set user stack pointer
.VBR	Display/set vector base register
DF	Display formatted registers
BF	Block of memory fill
BI	Block initialize
BM	Block of memory move
BS	Block of memory search
BT	Block of memory test
DC	Data conversion
.R0-.R6	Display/set relative offset register
OF	Display offsets

TABLE 4-1. TENbug Commands by Type (cont'd)

COMMAND MNEMONIC	DESCRIPTION
BR	Breakpoint set
NOBR	Remove breakpoint
GO	Execute program
GT	Go until breakpoint
GD	Go direct execute program
TR	Trace
TT	Trace to temporary breakpoint
PA	Printer attach
NOPA	Detach printer
BARS	Draw graphics bars test pattern
NOBARS	Clear graphics bars test pattern
CH	Display character data
NOCH	Remove character data from screen
CRT	Modify CRT control registers
CS	Checksum
GR	Display graphics RAM
NOGR	Remove graphics RAM from screen
PF	Port format
TM	Transparent mode
VM	Video map
HE	Help
DU	Dump memory (S-records)
LO	Load (S-records)
VE	Verify (S-records)
BH	Bootstrap halt
BO	Bootstrap operating system
IOC	I/O command for disk
IOP	I/O physical for disk
IOT	I/O teach for disk

4.2 TENbug COMMANDS

A complete description of each TENbug command is provided in the following paragraphs. Messages resulting from error conditions during command execution are described in Appendix B.

4.2.1 Display/Set Register (.<register>) .<register>

.<register> [<expression>]

The .<register> commands allow the user to display or modify individual registers. Commands with a leading period and the registers displayed/altered by these commands are:

.A0-.A7	address register
.D0-.D7	data register
.DFC	destination function code (used with MC68010 MOVES instruction)
.PC	program counter
.R0-.R6	relative offset register (software register) (refer to OF command)
.SFC	source function code (used with MC68010 MOVES instruction)
.SR	status register (in the MC68010)
.SSP	supervisor stack pointer
.USP	user stack pointer
.VBR	vector base register

EXAMPLE — COMMENT

```
TENbug 2.x > .PC                Display program counter.
.PC=00000A00

TENbug 2.x > .A7 F00            Set address register 7 to $F00.

TENbug 2.x > .R1 A00            Set relative offset register 1 to $A00.

TENbug 2.x > OF
R0-7 00000000 00000A00 00000000 00000000 00000000 00000000 00000000 00000000

                                Display all relative offset registers.

TENbug 2.x > DF
PC=00000A00 SR=2700=.S7..... USP=FFFFFFFF SSP=00000F00 VBR=00000000 SFC=2 DFC=7
D0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000F00
 PC=000000+R1 41F81000              LEA.L   $00001000,A0

                                Display all formatted CPU registers.
```

4.2.2 Draw Graphics Bars Test Pattern (BARS and NOBARS)

```
BARS
NOBARS
```

The BARS command provides a graphics test pattern that can be used to familiarize the user with a few of the graphics facilities. BARS will create a color or green scale consisting of eight horizontal and eight vertical bars. Each bar in a given axis is a different color or shade of green. Where a horizontal bar intersects a vertical bar, the result is the Exclusive-OR of the two colors or shades.

For more detailed information about the control registers and graphics RAM, refer to the VME/10 Microcomputer System Reference Manual.

If NOBARS is entered, the graphics RAM is cleared. If BARS is entered following a previous BARS command, the system automatically clears graphics RAM before redrawing the test pattern. If graphics RAM has been enabled (using the GR command), the test pattern can be seen while it is being drawn.

This diagnostic command will support high- or low-resolution mode, with the only observable difference being that the low-resolution version can be drawn in less time due to the reduced amount of RAM involved. For more information about changing from high- to low-resolution mode refer to the description of the VM command.

See also: [NO]CH, CRT, [NO]GR, VM

EXAMPLE	COMMENT
TENbug 2.x > GR	Display the contents of graphics RAM.
TENbug 2.x > BARS	Execute the graphics test pattern command. Notice both graphics and character data are displayed.
TENbug 2.x > VM	Change from high- to low-resolution mode.
TENbug 2.x m> GR	Enable graphics again.
TENbug 2.x m> NOCH	Disable the character RAM.
	NOTE The following commands will not be visible on the CRT display after execution of the NOCH command.
TENbug 2.x m> BARS	Draw test pattern, a little quicker this time.
TENbug 2.x m> GR 2	Allow only the color controlled by bit 2 in control register 1.
TENbug 2.x m> NOBARS	Clear graphics RAM.
TENbug 2.x m> CH	Restore the character display.
TENbug 2.x m> VM	Change from low- to high-resolution mode.

BF <address1> <address2> <pattern>

The BF command fills a specified block of memory with a specified binary pattern of word size. A word boundary (even address) must be given for the starting <address1> and ending <address2> of the block. The pattern word may be expressed in hexadecimal (default), decimal, octal, or binary format. Refer to the DC command for symbols used to denote numeric type. If a pattern of less than word size is entered, the data is right-justified and leading zeros are inserted by TENbug.

EXAMPLE

```
TENbug 2.x > MD 900
000900    FF FF 00 00 FF FF 00 00  FF FF 00 04 FF FF 00 00  ................

TENbug 2.x > BF 900 90E 4E75
PHYSICAL ADDRESS=00000900 0000090E

TENbug 2.x > MD 900
000900    4E 75 4E 75 4E 75 4E 75  4E 75 4E 75 4E 75 4E 75  NUNUNUNUNUNUNUNU

TENbug 2.x >
```

```
BH [<device>][,<controller>]
```

where:

device	Is a single hexadecimal digit (0-3) specifying the device to be used (default = 0).
controller	Is a single hexadecimal digit (0) specifying the controller to which the device is connected (default = 0).

The BH command causes data from disk to be loaded into memory and program control to be given to TENbug. If device and/or controller are not specified, device 0 and controller 0 are used.

This command works the same as BO, except that control is transferred to TENbug.

See also: BO

EXAMPLE COMMENT

```
TENbug 2.x > BH                 Boot Halt from default drive 0, default controller 0.
Booting from:  SYS              (Message appears only if first four bytes of volume
                                 ID are not null.)

PC=00001694 SR=2700=.S7..... USP=FFFFFFFF SSP=00040E00 VBR=00000000 SFC=2 DFC=7
D0-7 00000000 00000000 00000048 4D453455 4D505500 A987EDCB 00000000 0007FFFE
A0-7 00F1C0D1 00001694 0000067A 00F01E2C 00F01350 00000550 00000550 00040E00
  PC=001694   46FC2700            MOVE.W   #9984,SR

TENbug 2.x >


TENbug 2.x > BH 2               Boot Halt from drive 2, default controller 0.
Booting from:  TEN              (Message appears only if first four bytes of volume
                                 ID are not null.)

PC=000025D8 SR=2700=.S7..... USP=FFFFFFFF SSP=000014D8 VBR=00000000 SFC=2 DFC=7
D0-7 00000002 00000000 000000AC 4D453455 00000000 00000010 00000000 0007FFFE
A0-7 00F1C0D1 000025D8 00000682 00F01E2C 00F01350 00000552 00000552 000014D8
  PC=0025D8   41F814D8              LEA.L   $000014D8,A0

TENbug 2.x >
```

NOTE

To use the BH command, a valid stack value must be in locations $0-$3 of the file being loaded.

4.2.5 Block Initialize (BI) BI

BI <address1> <address2>

The BI command initializes word parity in a specified block of memory consisting of <address1> through <address2>. No data in any word is changed if parity in the word is correct. If parity in a word is incorrect, the characters "m?" ($6D3F) are written in that word to force correct parity. If the parity cannot be set in one or more words, the message BUS TRAP ERROR is displayed on the console. The BT (Block Test) command may be used to isolate the failure(s).

NOTE

Both addresses must be on word boundaries.

See also: BT

EXAMPLE

```
TENbug 2.x > BI 44000 4FFFE
PHYSICAL ADDRESS=00044000 0004FFFE

TENbug 2.x >
```

BM <address1> <address2> <address3>

where:

<address1> Is the starting address of the source memory block.

<address2> Is the ending address of the source memory block.

<address3> Is the starting address of the destination memory block.

The BM command is used to move (duplicate) blocks of memory from one area to another.

EXAMPLE

```
TENbug 2.x > MD B00 A;DI
000B00    1018                  MOVE.B  (A0)+,D0
000B02    0C000000              CMP.B   #0,D0
000B06    67F8                  BEQ.S   $000B00
000B08    4E75                  RTS

TENbug 2.x > MD A00 A;DI
000A00    FFFF                  DC.W    $FFFF
000A02    0000FFFF              OR.B    #-1,D0
000A06    0020FFFF              OR.B    #-1,-(A0)

TENbug 2.x > BM B00 B09 A00
PHYSICAL ADDRESS=00000B00 00000B09
PHYSICAL ADDRESS=00000A00

TENbug 2.x > MD A00 A;DI
000A00    1018                  MOVE.B  (A0)+,D0
000A02    0C000000              CMP.B   #0,D0
000A06    67F8                  BEQ.S   $000A00
000A08    4E75                  RTS

TENbug 2.x >
```

4.2.7 Bootstrap Operating System (BO) BO

BO [<device>][,<controller>][,<string>]

where:

device — Is a single hexadecimal digit (0-3) specifying the device to be used (default = 0).

controller — Is a single hexadecimal digit (0) specifying the controller to which the device is connected (default = 0).

string — Is an optional ASCII character string that is passed to the program being loaded from the specified device and controller.

The function of the BO command is to access a program on disk, transfer it into memory space, and give control to that program. Where to find the program and where in memory to store the program is contained in sector 0 of the disk corresponding to the specified device and controller. If the device and controller are not specified, the default value zero is used for each.

The following sequence occurs when the BO command is executed:

a. Starting at sector 0 (the volume ID), 256 bytes are read and transferred into TENbug workspace RAM.

b. If the volume ID (locations $0 through $3) is not null, these four ASCII bytes will be displayed as follows:

```
Booting from:  SYS
```

Where SYS is the volume ID. If null, the display is suppressed.

c. Motorola ID locations $F8-$FF are read to ensure that they contain either "EXORMACS" or "MOTOROLA".

d. The location of the program to be loaded and its destination in memory are identified by examining the first sector at the following locations.

LOCATIONS	CONTENTS
$14-$17	First 256-byte sector to transfer
$18-$19	Number of sectors to transfer
$1E-$21	Address of first destination byte (first memory address)

e. The location of the disk configuration area is identified by examining volume ID locations as shown:

LOCATIONS	CONTENTS
$90-$93	Sector address of the media configuration parameters (normally sector 1)
$94	Length of the configuration area (normally one 256-byte sector)

If there is no media configuration area specified, default media configuration values are used to read the disk. If there is a media configuration area specified, then that VERSAdos sector is read into the TENbug workspace, and these values are used to read the disk. Refer to Appendix C for additional information.

f. The program is read and transferred to its memory destination.

g. The status register is updated to reflect supervisor mode and interrupt level 7.

h. The stack pointer is loaded from locations $0-$3 relative to the destination memory.

i. The program counter is loaded from locations $4-$7 relative to the destination memory.

The registers are set up as defined below, and the program loaded by the BO command now has control of execution.

```
D0...DRIVE NUMBER
D1...IPC NUMBER
D2...DISK CONFIGURATION CODE
D3...FLAG FOR IPL; 'ME4U' = USE BUGS' DISK READ ROUTINE
A0...ADDRESS OF DISK CONTROLLER BOARD
A1...ADDRESS OF PROGRAM JUST LOADED
A2...ADDRESS OF DISK CONFIGURATION DATA
A3...ADDRESS OF BUGS' DISK READ ROUTINE
A4...ADDRESS OF THE DEBUGGER ENTRY POINT ("MACSBUG")
A5...START OF TEXT
A6...END OF TEXT+1 (WHERE THE NEXT CHARACTER WOULD GO)
A7...STACK OF PROGRAM JUST LOADED
SR...SUPERVISOR MODE AND LEVEL SEVEN
```

These registers can be used by IPL to load the file identified by the string field. If a string field is specified on the BO command line, registers A5 and A6 point to the first and last plus one characters of the string. If no string is specified, register A5 = A6. The file name may be followed by a semicolon and either or both of the options L=$<address> or H. Specifying H causes control to be returned to TENbug, rather than to the specified program.

Refer to the discussion of the bootload file, IPL.SY, in the M68000 Family VERSAdos System Facilities Reference Manual.

The devices and controllers currently supported by TENbug are assigned as follows:

DEVICE #	DESCRIPTION
0	Winchester hard disk
1	Winchester hard disk
2	5 1/4" Winchester floppy disk
3	5 1/4" Winchester floppy disk

CONTROLLER #	DESCRIPTION
0	RWIN1 disk controller

EXAMPLE	COMMENT
TENbug 2.x > BO Booting from: SYS	Boot from the default drive and controller, (drive is 0, controller is 0). Note: The volume ID is SYS.
TENbug 2.x > BO 2 Booting from: TEN	Boot from drive 2 (first floppy) on the default controller (RWIN1 number 0). Note: The volume ID is TEN.
TENbug 2.x > BO ,,0.VME10.KBD.SY Booting from: SYS	Boot from default device 0 on the RWIN1 (default controller 0) and pass the ASCII string that will request the program named KBD.SY under catalog VME10, account number 0, to be loaded by the IPL program. (IPL was the program booted into memory with the BO command.) Note: The volume ID is SYS.

4.2.8 Breakpoint Set and Remove (BR and NOBR)

```
BR (display only)
BR [<address>[;<count>]] [<address>[;<count>]]...
NOBR [<address>[<address>]...]
```

When encountered, a breakpoint causes target program execution to stop and control to be transferred to TENbug. The BR command may be used without parameters to cause display of current breakpoint addresses. The BR <address> command sets one or more addresses into the breakpoint address table. This table can hold up to eight breakpoint addresses. Multiple breakpoints (up to eight) may be specified with one call of the Breakpoint command. Addresses should be on even word boundaries. The range of <count> is a 32-bit integer.

The breakpoints are inserted into the target program when execution is called via a GO or GT command. The illegal instruction $4AFB is inserted at the addresses specified by the table. During execution of the program, a breakpoint occurs whenever this instruction is encountered. If program control is lost, control may be regained via the RESET or the ABORT button. ABORT is preferred because use of the RESET function may leave breakpoints ($4AFB) in the user program, whereas ABORT will recover properly (refer to Appendix A).

While executing a Trace command, the breakpoint addresses are monitored (i.e., the illegal instruction $4AFB is not placed in memory).

After stopping at a breakpoint, execution may be continued by typing the GO command.

The NOBR command removes one or more breakpoints from the internal breakpoint table. The NOBR command without parameters eliminates all breakpoints.

BR COMMAND FORMAT	DESCRIPTION
TENbug 2.x > BR	Display all breakpoints.
TENbug 2.x > BR <address>	Set a breakpoint.
TENbug 2.x > BR <address>;<count>	Set a breakpoint with a count. Count is decremented each time the breakpoint is encountered until <count> = 0. Execution stops as soon as count is decremented to zero. Thereafter, execution will stop each time the breakpoint is reached.

NOBR COMMAND FORMAT	DESCRIPTION
TENbug 2.x > NOBR	Clear all breakpoints.
TENbug 2.x > NOBR <address>	Clear a specific breakpoint.

See also: GT, TT

EXAMPLE

```
TENbug 2.x > .R4 4000

TENbug 2.x > BR 1010 2000;5 2040 4000

BREAKPOINTS
001010     001010
002000     002000;5
002040     002040
000000+R4  004000

TENbug 2.x > NOBR 1010 2040

BREAKPOINTS
002000     002000;5
000000+R4  004000

TENbug 2.x > NOBR

BREAKPOINTS

TENbug 2.x >
```

```
BS <address1> <address2> '<literal string>'
BS <address1> <address2> <data> [<mask>][;<option>]
```

The BS command has two modes: literal string search and data search. Both modes scan memory beginning at <address1> through <address2>, looking for a match.

The literal string mode is initiated if a single quote (') follows <address2>. The ASCII literal string can include lowercase letters. If a single quote does not follow <address2>, data search mode is assumed. In the data search mode, the optional mask (if used) is ANDed to data. The default mask is all ones. The options supported are:

```
;B   byte
;W   word
;L   longword
```

The default is byte.

In both modes of the BS command, if the search finds matching data, the data and the address(es) are displayed. If the search is in data search mode with a mask, and data is found that matches the data after the mask is ANDed, the data from memory before applying the AND mask is displayed.

EXAMPLE

```
TENbug 2.x > MD 41FF0 15
001FF0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ................
002000  43 43 45 45 00 00 00 00  00 00 00 00 00 00 00 00   CCEE............
```

```
TENbug 2.x > BS 41FF0 4200F 'CC'
PHYSICAL ADDRESS=00001FF0 0000200F
002000 'CC'
```

Successful search for literal string 'CC'.

```
TENbug 2.x > BS 41FF0 4200F 34 ;W
PHYSICAL ADDRESS=00001FF0 0000200F
```

Unsuccessful search for word-length data (with default mask).

```
TENbug 2.x > BS 41FF0 4200F 03 0F
PHYSICAL ADDRESS=00001FF0 0000200F
002000 43
002001 43
```

Successful search for byte-length data, with four most significant bits masked.

```
TENbug 2.x > BS 41000 47FFE 4AFB;W
PHYSICAL ADDRESS=00001000 00007FFE
001000 4AFB
```

Successful search for "leftover" breakpoints.

```
TENbug 2.x > md 10000 30
010000   54 68 69 73 20 69 73 20  61 20 6D 65 73 73 61 67  This is a messag
010010   65 20 66 6F 72 20 74 68  65 20 42 53 20 63 6F 6D  e for the BS com
010020   6D 61 6E 64 2E 20 20 20  20 20 20 20 20 20 20 20  mand.
```

Display of memory that will be searched for lowercase letters.

```
TENbug 2.x > bs 10000 20000 'is'
PHYSICAL ADDRESS=00010000 00020000
010002 'is'
010005 'is'

TENbux 2.x >
```

Block Search the address range for ASCII 'is'.

Successful search finding two occurrences.

BT <address1> <address2>

The BT command provides a destructive test of a block of memory. A word boundary (even address) must be given for the starting <address1> and ending <address2> of the block. If the test runs to completion without detecting an error, all memory tested will have been set to zeros.

Execution of this command may take several seconds for large blocks of memory.

When a problem is found in a memory location, the address, the data stored, and the data read are displayed. Control is then returned to TENbug.

See also: BI

EXAMPLE	COMMENT
TENbug 2.x > BT 44000 47FFE PHYSICAL ADDRESS=00044000 00047FFE	Successful memory test; no errors found.
TENbug 2.x > BT 44000 4FFFE PHYSICAL ADDRESS=00044000 0004FFFE FAILED AT 0480FE WROTE=FFFF READ=0000 TENbug 2.x >	Unsuccessful memory test; error data is listed.

CH [<bits>]
NOCH

The Character Display (CH) command provides access to specific bits within VME/10 control register 0 ($F19F05). These bits determine whether the character RAM is displayed upon the CRT built into the VME/10. They are called the character disable bits, and must be off to allow the color or shade of green being used to draw the characters on the display.

For more detailed information about the control registers, refer to the VME/10 Microcomputer System Reference Manual.

An optional <bits> parameter, 0-7, can be provided to replace the current configuration of bits 7, 6, and 5 of VME/10 control register 0. Default is 0 (or all bits off), allowing the character display to appear on the CRT.

If NOCH is entered, the current values of bits 7, 6, and 5 within control register 0 are first saved, and then replaced by 7 (all bits on). This removes any character data appearing on the screen. The data still resides in display RAM, unchanged; only the control register has been modified.

To again display character data within display RAM, the CH command can be entered. The value saved when NOCH was last executed is restored. If the optional <bits> parameter is provided, however, the new value is used in place of the bit pattern previously saved.

See also: [NO]BARS, CRT, [NO]GR, VM

EXAMPLE	COMMENTS
TENbug 2.x > NOCH	Remove any character data being displayed upon the built-in terminal.
	NOTE The keyboard is still operational; however, the data entered will not be displayed.
TENbug 2.x > CH	When CH is entered, all previous data that was on the screen (minus any lines that scrolled off the top) will be seen again.

CRT

The CRT command provides an easy way to access the VME/10 control registers that affect the CRT display. Entering the CRT command begins a sequence of prompts that displays the current result of a particular control register bit(s). The user is able to continue without changing the register, or to walk through a preselected set of parameters available for that bit within the register.

For more detailed information about the control registers, see the VME/10 Microcomputer System Reference Manual.

The first prompt presented is the video amplifier duty cycle. Bit 3 within control register 0 is toggled on and off, selecting 50% or 100% duty cycle. Parameters can be toggled by pressing any character on the keyboard; the value continues to alternate with each press of a key. When the desired option is selected, pressing ENTER (or carriage return) moves on to the next parameter.

The second prompt selects a cursor. In the same manner as before, three selections are presented: one for each character entered. When the cursor selection is complete, press ENTER (or carriage return) to proceed. Cursor selection is controlled by bits 5 and 6 within control register 1.

The third prompt selects an optional blinking cursor. Blinking alternates with a steady cursor for each character entered. Bit 4 within control register 0 controls cursor blink. After carriage control is entered the next prompt is displayed.

The fourth and last option selected is inverse video. Bit 2 within control register 0 is alternately set or cleared to select inverse or normal video. When the selection is made, press ENTER and the following message will be displayed on the built-in terminal:

```
    Video Set

TENbug 2.x >
```

See also: [NO]BARS, [NO]CH, [NO]GR, VM

CS [<address1>] [<address2>]

The Checksum command provides access to the same checksum routine used by the power-up/reset (PWRT) self-test firmware. This routine is used in two ways within TENbug.

a. At power up, if pins 5 and 6 of jumper J2 are connected, the self-test is executed. One of the many items verified is the checksum contained in TENbug ROM. If for any reason the contents of ROM were to change from the factory version, the checksum test is designed to detect the change and inform the user of the failure.

b. Following a valid self-test, TENbug 2.x examines the VME address space for code that needs to be executed. This feature (ROMBOOT) makes use of the checksum routine to verify that a routine in memory is really there to be executed at power up. For more information refer to paragraph 2.10, which describes the format of the routine to be executed and the interface provided upon entry.

This command is provided as an aid in preparing routines for the ROMBOOT feature. Since ROMBOOT does checksum validation as part of its screening process, the user needs access to the same routine in the preparation of EPROM/ROM routines.

The [<address>] parameters can be provided in two forms:

a. An absoute address (24-bit maximum).
b. An expression using a displacement + relative offset register.

Any previous addresses are saved as default addresses for CS commands invoked later. This is convenient since users typically enter the address range to calculate the checksum and then enter the results into memory (into bytes that were $0000 while the checksum was calculated). When the CS command is used to verify the content and location of the new checksum, the operands need not be entered since the command retains the addresses used to calculate the previous checksum. The even and odd byte result should be 0000, verifying that the checksum bytes were calculated correctly and placed in the proper locations.

The default operands at power up are the starting/ending addresses of the TENbug 2.x firmware. The results for even and odd bytes should be 0000.

The algorithm used to calculate the checksum is as follows:

a. $FF is placed in each of two bytes within a register. These bytes represent the even and odd bytes as the checksum is calculated.

b. Starting with the first address, the even and odd bytes are extracted from memory and XORed with the bytes in the register.

c. This process is repeated, word by word, until the ending address is reached. Note that the last word addressed is NOT included in the checksum. This technique allows use of even ending addresses ($D40000 as opposed to $D3FFFE).

EXAMPLE

```
TENbug 2.x > CS
PHYSICAL ADDRESS=00F00000 00F08000
      (EVEN ODD)=0000
```

COMMENT

CS command entered without operands right after power up (addresses are TENbug firmware limits by default).

```
TENbug 2.x > MD 20000 3F
```

Display routine requiring a checksum. Start at $20000; last byte is at $20029. Checksum will be placed in bytes at $20026 and $20027, so they are zero while calculating the checksum.

```
020000   42 4F 4F 54 00 00 00 14  00 00 00 A6 54 65 73 74  BOOT.......&Test
020010   41 F9 00 01 F0 00 20 3C  00 00 EF FF 11 00 51 C8  Ay..p. <..o...QH
020020   FF FC 4E 75 01 01 00 00  10 08 FF FF FF FF FF FF  .:Nu............
020030   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
```

```
TENbug 2.x > M 20010;DI
020010   41F90001F000          LEA.L   $0001F000,A0 ?(CR)
020016   203C0000EFFF          MOVE.L  #61439,D0 ?(CR)
02001C   1100                 MOVE.B  D0,-(A0) ?(CR)
02001E   51C8FFFC             DBF.L   D0,$02001C ?(CR)
020022   4E75                  RTS     ?(CR)
020024   0101                  BTST    D0,D1 ?(CR)
020026   0000                  DC.W    $0000 ?(CR)
020028   1008                  DC.W    $1008 ?(CR)
02002A   FFFF                  DC.W    $FFFF ?(CR)
02002C   FFFF                  DC.W    $FFFF ?(CR)
02002E   FFFF                  DC.W    $FFFF ?(CR)
020030   FFFF                  DC.W    $FFFF ?.(CR)
```

Display executable code plus revision number, checksum, socket ID, and a few unused bytes following the routine:

0101 is revision.
0000 is where checksum is to be placed.
1008 are socket locations U16 and U08.
FFFF is unused memory.
FFFF is unused memory.
FFFF is unused memory.
FFFF is unused memory.

```
TENbug 2.x > CS 20000 2002A
PHYSICAL ADDRESS=00020000 0002002A
      (EVEN ODD)=4B34
```

Request checksum of area using absolute addresses.

Checksum of even bytes is $4B.
Checksum of odd bytes is $34.

```
TENbug 2.x > M 20026;W
020026 0000 ?4B34.
```

Place these bytes in zeroed area used while calculating checksum.

```
TENbug 2.x > CS
PHYSICAL ADDRESS=00020000 0002002A
      (EVEN ODD)=0000
```

Verify checksum (no operands needed if same as previous entries).
Result is 0000, good checksum.

```
TENbug 2.x > .R3 2000
```

Define value of relative offset register 3.

```
TENbug 2.x > CS 0+R3 2A+R3
PHYSICAL ADDRESS=00020000 0002002A
      (EVEN ODD)=4B34
```

Request checksum of area using relative ofset.

Checksum of even bytes is $4B.
Checksum of odd bytes is $34.

```
TENbug 2.x > M 26+R3;W
000026+R3 0000 ?4B34.
```

Place these bytes in zeroed area used while checksum was calculated.

```
TENbug 2.x > CS
PHYSICAL ADDRESS=00020000 0002002A
     (EVEN ODD)=0000

TENbug 2.x >
```

Verify checksum (no operands needed if same as previous entries).

DC <expression>

The DC command is used to convert an expression into hexadecimal and decimal. The expression may be entered in hexadecimal, decimal, or mixed format; output will be shown both ways. Default input format is hexadecimal. Octal and binary values may also be converted to decimal and hexadecimal values.

The following symbols are used:

$	precedes hexadecimal value (default; may be omitted)
&	precedes decimal value
@	precedes octal value
%	precedes binary value

Except for .R0, offset registers may not be used with the DC command.

This command is useful in calculating displacements such as destination of relative branch instructions or program counter relative addressing modes.

COMMAND FORMAT	DESCRIPTION
TENbug 2.x > DC $<data>	Convert hexadecimal data into hexadecimal and decimal.
TENbug 2.x > DC &<data>	Convert decimal data into hexadecimal and decimal.

EXAMPLE

```
TENbug 2.x > DC &120
000078  =$78=&120

TENbug 2.x > DC &15+$4-$13
000000  =$0=&0

TENbug 2.x > DC -1000
FFF000  =$FFFFF000=-$1000=-&4096

TENbug 2.x > DC &15-$9+@14-%1100
000006  =$6=&6

TENbug 2.x >
```

DF

The DF command is used to display the MC68010 registers. The registers display is also provided whenever TENbug gains control of the program execution (i.e., at breakpoints and when tracing).

Note that any single register can be displayed with the .A0-.A7, .D0-.D7, and similar commands. Refer to the descriptions of the Display/Set Register command (.<register>) and the OF command.

EXAMPLE	COMMENTS
TENbug 2.x > DF	Display formatted registers. Notice that the values of register A7 and the user stack pointer are the same because the status register indicates user mode.

```
PC=00000000 SR=0000=..0..... USP=FFFFFFFF SSP=00000000 VBR=00000000 SFC=2 DFC=7
D0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 FFFFFFFF
 PC=000000    0000                  DC.W    $0000
```

TENbug 2.x > .A7 1100

TENbug 2.x > DF — Once again the values of A7 and the user stack pointer are the same. The latter was changed by altering A7 in the user mode.

```
PC=00000000 SR=0000=..0..... USP=00001100 SSP=00000000 VBR=00000000 SFC=2 DFC=7
D0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001100
 PC=000000    0000                  DC.W    $0000
```

TENbug 2.x > .SS B00 — Set supervisor stack pointer.

TENbug 2.x > .SR 2700 — Set supervisor state and interrupt level 7.

TENbug 2.x > .R3 A00 — Set relative offset register 3 to $A00.

TENbug 2.x > .PC 0+R3 — Set program counter to start of area using the relative offset register.

TENbug 2.x > DF

Display formatted registers again. Notice that the supervisor mode in the status register results in the supervisor stack pointer being displayed both as the SSP and A7. Other changes include the program counter now being displayed in two ways: on the first line as an absolute address, and on the fourth line relative to the closest offset register equal to or below the absolute address. Notice also that the current location of the program counter is displayed in both hexadecimal and disassembled M68010 source statements.

```
PC=00000A00 SR=2700=.S7..... USP=00001100 SSP=00000B00 VBR=00000000 SFC=2 DFC=7
D0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000B00
 PC=000000+R3 41F81000             LEA.L   $00001000,A0

TENbug 2.x >
```

DU[<port number>] <address1> <address2> [<text>]

The DU command formats memory data in S-record form and sends it to a specified port. The default port number is port 1, the VME/10 built-in CRT terminal/keyboard. The first record output is an S0 record, which will contain the characters entered in the text field on the command line, if any. The last record output is an S7, S8, or S9 terminator. See Appendix D for information on S-records.

To dump to a peripheral using the DU command, a dual serial I/Omodule, MVME400, or a dual parallel I/Omodule, MVME410, must be available on the I/O Channel. Note that serial ports 1 and 2 on the MVME400 correspond to TENbug ports 3 and 2, respectively, and that parallel ports 1 and 2 on the MVME410 correspond to TENbug ports 4 and 5, respectively.

Default destination is the console terminal. Specifying DU<port number> allows the output to be directed to another port.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 1 (VME/10 built-in terminal/keyboard).
1	Specifies TENbug port 1 (VME/10 built-in terminal/keyboard).
2	Specifies TENbug port 2 (MVME400 port 2 - 7201/B).
3	Specifies TENbug port 3 (MVME400 port 1 - 7201/A).
4	Specifies TENbug port 4 (MVME410 port 1 - PIA/A).
5	Specifies TENbug port 5 (MVME410 port 2 - PIA/B).

This command does not send control characters to start or stop peripheral devices.

See also: LO, PF, VE

NOTE

Offset R0 is added to the address field in each S-record.

EXAMPLE	COMMENT
TENbug 2.x > MD A00 30	Display memory where routine to be transferred exists.

```
000A00   41 F8 10 00 20 3C 00 00  02 FF 11 00 51 C8 FF FC  Ax.. <......QH.:
000A10   60 EE 4E 71 4E 71 4E 71  4E 71 4E 71 4E 71 4E 71  'nNqNqNqNqNqNqNq
000A20   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
TENbug 2.x > MD A00 7;DI      Display memory using the disassembler.

000A00    41F81000            LEA.L   $00001000,A0
000A04    203C000002FF        MOVE.L  #767,D0
000A0A    1100                MOVE.B  D0,-(A0)
000A0C    51C8FFFC            DBF.L   D0,$000A0A
000A10    60EE                BRA.S   $000A00
000A12    4E71                NOP
000A14    4E71                NOP

TENbug 2.x > DU A00 A15 TENBUG 2.X Test

                              Dump memory to default port, starting at $A00
                              through A15, and place title within S0 record.

PHYSICAL ADDRESS=00000A00 00000A15          Note: This appears on the default
S012000054454E42554720322E582054455354F0          CRT display.
S1130A0041F81000203C000002FF110051C8FFFC17
S1090A1060EE4E714E7110
S9030000FC

TENbug 2.x >
```

GD [<address>]

The GD command is similar to the GO command, except that GD does not set breakpoints, nor does it start by tracing one instruction. The GD command starts the target program at the location given as <address> without changing any of the exception vectors (default locations 0 through $3FF). If <address> is not specified, the GD command starts the target program at the address in the program counter.

See also: GO, GT

EXAMPLE

```
(Listing of program in memory at location 001900)
001900    1018                  MOVE.B    (A0)+,D0
001902    0C000000              CMP.B     #0,D0
001906    66F8                  BNE.S     $001900
001908    4E75                  RTS

TENbug 2.x > BR 1900 1908

BREAKPOINTS
001900    001900
001908    001908

TENbug 2.x > G 1900
PHYSICAL ADDRESS=00001900

AT BREAKPOINT
PC=00001900 SR=2704=.S7..Z.. USP=0000C19E SSP=00000C00 VBR=00000000 SFC=2 DFC=2
D0-7 00000000 00000000 00003048 4D453455 00000000 00000020 00000000 0007FFFE
A0-7 00001002 00001694 0000065C 00F01D72 00F0120C 00000538 00000538 00000C00
  PC=001900

TENbug 2.x > GD 1900
PHYSICAL ADDRESS=00001900
```

GO [<address>]
G [<address>]

The Go (G or GO) command causes the target registers (previously saved in RAM) to be placed into the actual MC68010 hardware registers, and any breakpoints previously requested to be placed into RAM. When this is completed, control is given to the target program by one of two methods. If no operands are provided with the G (or GO) command, the current value of the program counter is used. If an address is provided, this address will be placed into the program counter and then used to give control to the target code. The program starts by first tracing one instruction and then free running until one of the following events interrupts the program execution.

a. The target program encounters a breakpoint.

b. An abnormal program sequence causes exception processing (e.g., divide by zero).

c. The operator intervenes through use of the RESET or ABORT pushbuttons on the VME/10 operator panel.

NOTE

The execution will be in REAL TIME unless any breakpoints with <count> are encountered.

The [<address>] parameter can be provided in several formats:

a. An absolute address (24-bit maximum).

b. An expression using a displacement + relative offset register.

c. Address indirect, using the contents of RAM (or ROM) to acquire the new program counter contents.

d. Register indirect, using the contents of address registers 0 through 7 to acquire the new program counter contents.

EXAMPLE	COMMENT
TENbug 2.x > .PC A00	Set program counter to desired address.
TENbug 2.x > G PHYSICAL ADDRESS=00000A00 AT BREAKPOINT PC=000A0A 1100 MOVE.B D0,-(A0)	Enter Go command using existing PC.
TENbug 2.x > G A00 PHYSICAL ADDRESS=00000A00 AT BREAKPOINT PC=000A0A 1100 MOVE.B D0,-(A0)	Enter Go command with absolute address provided.
TENbug 2.x > M 20000;L 020000 00000000 ?A00.	Set RAM location to contain an execution address.
TENbug 2.x > GO [20000] PHYSICAL ADDRESS=00000A00 AT BREAKPOINT PC=000A0A 1100 MOVE.B D0,-(A0)	Enter Go command providing indirect address in RAM.
TENbug 2.x > .A1 A00	Set address register to contain an execution address.
TENbug 2.x > G (A1) PHYSICAL ADDRESS=00000A00 AT BREAKPOINT PC=000A0A 1100 MOVE.B D0,-(A0)	Enter Go command providing indirect addressing in A1.
TENbug 2.x > .R2 A00	Set relative offset register to contain start of module.
TENbug 2.x > GO 0+R2 PHYSICAL ADDRESS=00000A00 AT BREAKPOINT PC=00000A+R2 1100 MOVE.B D0,-(A0)	Enter Go command providing a displacement and offset register.
TENbug 2.x >	

Notice the physical address used in each example, though provided in a different way in each case, is identical.

4.2.19 Graphics RAM Display (GR and NOGR)

```
GR [<bits>]
NOGR
```

The Graphics Display (GR) command provides access to specific bits within VME/10 control register 1 ($F19F07). These bits determine whether the graphics RAM is displayed upon the CRT built into the VME/10. These bits are called the graphics enable bits and must be on to allow the respective colors, or shades of green, that they control to be displayed upon the screen.

For more detailed information about the control registers, see the VME/10 Microcomputer System Reference Manual.

An optional <bits> parameter, 0-7, can be provided to replace the current configuration of bits 3, 2, and 1 of VME/10 control register 1. Default is 7 (all bits on), allowing the character display to appear on the CRT.

If NOGR is entered, the current values of bits 3, 2, and 1 within control register 0 are first saved, and then replaced by zero (all bits off). This removes any graphics data appearing on the screen. The data still resides in display RAM; only the control register has been modified.

To once again display graphics data the GR command can be entered. The value saved when NOGR was last executed will be restored. If the optional <bits> parameter is provided, the new value is used in place of the bit pattern previously saved.

See also: [NO]BARS, [NO]CH, CRT, VM

EXAMPLE	COMMENT
TENbug 2.x > GR	Display the contents of graphics RAM.
TENbug 2.x > BARS	Execute the graphics test pattern command. Notice both graphics and character data are displayed.
TENbug 2.x > NOCH	Remove all character data from the screen.
Remember the following CANNOT be seen but is shown as a guide:	
TENbug 2.x > GR 4	Enable only the color, or shade of green, controlled by bit 2 of control register 1.
TENbug 2.x > NOGR	When the carriage return is pressed, all graphics and character data are gone.
TENbug 2.x > GR	The last value previously used in the graphics control register bits is restored (4 in this example).
TENbug 2.x > CH	Restore the character display.
Now the entire character screen is shown as well as graphics RAM.	
TENbug 2.x > NOGR	Remove the graphics display.
TENbug 2.x >	

4.2.20 Go Until Breakpoint (GT) GT

GT <temporary breakpoint address>

The GT command performs the following:

a. Sets the temporary breakpoint specified on the command line.

b. Sets breakpoints entered by the BR command.

c. Sets target program registers as displayed by the DF command.

d. Causes the target program to execute from the PC address (free run in real time).

When any breakpoint is encountered, the temporary breakpoint is reset.

See also: BR, DF, GD, GO, TR, TT

EXAMPLE

```
(Listing of program in memory at location 001900)
001900    1018                  MOVE.B   (A0)+,D0
001902    0C000000              CMP.B    #0,D0
001906    66F8                  BNE.S    $001900
001908    4E75                  RTS

TENbug 2.x > BR 1900 1908

BREAKPOINTS
001900    001900
001908    001908

TENbug 2.x > .PC 1900

TENbug 2.x > GT 1906
PHYSICAL ADDRESS=00001906
PHYSICAL ADDRESS=00001900

AT BREAKPOINT
PC=00001906 SR=2700=.S7..... USP=0000C19E SSP=00000BF8 VBR=00000000 SFC=2 DFC=2
D0-7 00000020 00000000 00003048 4D453455 00000000 00000020 00000000 0007FFFE
A0-7 0000160E 00001694 0000065C 00F01D72 00F0120C 00000538 00000538 00000BF8
  PC=001906

TENbug 2.x > BR

BREAKPOINTS
001900    001900
001908    001908

TENbug 2.x >
```

4.2.21 Help (HE)

HE

The HE command displays a list of available commands.

EXAMPLE

```
TENbug 2.x > HE

.PC .SR. .US .SS .VBR .DFC .SFC
.D0 thru .D7
.A0 thru .A7
.R0 thru .R7

CRT   CH    NOCH  GR    NOGR    (Control Registers)
BARS  NOBARS                    (Test Graphics RAM)
IOC   IOP   IOT                 (Physical Disk I/O)

CS    BF    BH    BI    BM    BO    BR    NOBR
BS    BT    DC    DF    DU    G     GD    GO
GT    HE    LO    M     MD    MM    MS    OF
PA    NOPA  PF    T     TM    TR    TT    VE
VM

TENbug 2.x >
```

IOC

The IOC command allows the user to issue commands directly to the RWIN1 controller.

When invoked, this command prompts for the drive and controller required. An address where the current RWIN1 command is located and hex display of that command are shown, followed by an "ARE YOU SURE?" prompt.

The command is used primarily as a debugging tool to issue commands to the RWIN1 controller to locate problems with either drives, media, or the controller itself. The RWIN1 commands are as follows:

0/0	Check drive status
0/1	Recalibrate
0/4	Format drive
0/6	Format track (refer to example)
0/7	Format default/alternate track
0/8	Read sectors (use IOP command)
0/9	Scan sectors
0/A	Write sectors (use IOP command)
0/B	Seek
0/C	Read track (Winchester only)
0/D	Read ECC (Winchester only)
0/E	Write ECC (Winchester only)
6/0	Configure drive

NOTE: For more information see the Winchester Disk Controller User's Manual.

The default values are the parameters left over from any previous controller request. An IOP command or, if an operating system has been booted and the debugger was reentered with the use of the ABORT button, the drive, controller and the last command (08 read sectors) issued by boot to the RWIN1 controller are the current default values.

While answering the prompts, there are four actions that can be taken following the question mark prompt:

? (CR) - Entering a carriage return indicates that the existing value for the current parameter is acceptable; go on to the next parameter.

? . - Entering a period indicates that this execution of the IOC command must be terminated now, without asking for more parameters.

? ^ - Entering a caret symbol indicates that a previous parameter requires a change and will logically back up one parameter each time it is entered (until the first entry is reached, where it will remain until one of the other responses is received).

? <data> - Entering the appropriate data requested (followed by a carriage return or ENTER). Often the parameters are checked for valid options (i.e., Y or N).

EXAMPLE	COMMENT
TENbug 2.x > IOP	Invoke the physical disk I/O command.
READ OR WRITE (R/W)=.....R ? W	Specify a write operation.
MEMORY ADDRESS FOR DISK I/O=.$00001000 ? 2000	Write to memory location $2000.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? 2	Write to drive 2 (floppy).
CONTROLLER NUMBER=.....$00 ? (CR)	Write to RWIN1 controller.
FIRST BLOCK NUMBER=.$00000000 ? 100	Write to block number $100.
NUMBER OF (256 BYTE) BLOCKS=.....$0001 ? (CR)	Write one 256-byte block.
ARE YOUR SURE (Y/N) ? Y	Last chance ... Sure? Yes
DISK ERROR: STATUS=06 4D 00 08 09 01 40 01 00 09	For this example an unformatted floppy disk was placed into drive 2. As expected an "06" indicates that the ID HEADER NOT FOUND message from the Winchester Disk Controller User's Manual is the correct description of the situation.

```
PC=00F04EBC SR=2700=.S7..... USP=FFFFFFFF SSP=000008CC VBR=00000000 SFC=2 DFC=7
D0-7 00000001 000000AC 00002100 00000109 00000000 00000010 00000028 00042700
A0-7 00F1C0D1 00002100 00000682 00F1C0D9 00001010 0000054E 0000054E 000008CC
 PC=F04EBC    4BFAFDC0              LEA.L   $00F04C7E(PC),A5
```

TENbug 2.x > IOC	The IOC command will be used to write the ID header for a specific track. To start, the RWIN1 command must be located. This is done by entering an IOC command. The RWIN1 command last used is then displayed along with its address. In this case, the last request was "0A" for the write that was attempted in the IOP.
DRIVE NUMBER (0&1=FIXED,2&3+FLOPPY)=.....$02 ? (CR)	
CONTROLLER NUMBER=.....$00 ? (CR)	
DISK COMMAND AT $00000634 = $0A 40 01 08 01 AC	
ARE YOU SURE (Y/N) ? N	

```
TENbug 2.x > M 634
000634 0A ? 06
```

Modify memory at the specified address ($634 in this example) and change the write ($0A) operation to a format track ($06) operation (Chapter 4 in the Winchester Controller User's Manual).

```
TENbug 2.x > IOC
DRIVE NUMBER (0&1=FIXED,2&3-FLOPPY)= .......$02 ? (CR)
                CONTROLLER NUMBER= .......$00 ? (CR)

DISK COMMAND AT $00000634 = $0A 40 01 08 01 AC

ARE YOU SURE (Y/N) ? Y
```

Invoke the IOC command and, after verifying the correct drive and controller and that the RWIN1 command is proper, reply Y in response to the ARE YOU SURE? prompt.

```
TENbug 2.x > IOP
              READ OR WRITE (R/W)=.........W  ? (CR)
       MEMORY ADDRESS FOR DISK I/O=.$00002000  ? (CR)
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.......$02  ? (CR)
                CONTROLLER NUMBER=.......$00  ? (CR)
               FIRST BLOCK NUMBER=.$00001000  ? (CR)
       NUMBER OF (256 BYTE) BLOCKS=.....$0001  ? (CR)

ARE YOU SURE (Y/N)  ? Y

"W" COMPLETE

TENbug 2.x >
```

With the return of the TENbug prompt the command is complete. Note that a total format will require longer than the "dead man" timer provides; the TENbug program will time out and send an error message, but the RWIN1 controller will proceed with the command until complete.

In this example the write can now be completed, as long as the one track that was formatted is all that is specified.

IOP

The IOP command allows the user to do physical reads or writes to the disk. When invoked, this command prompts for the information required to perform the Input/Output operation. The initial values for drive and controller are zeros, (hard disk 0 on the RWIN1 controller). However, once a parameter has been changed the new value will become the new default.

Notice that this command can only perform reads and writes through the RWIN1 controller. If for some reason any of the other RWIN1 commands are required, the IOC command can be used.

While answering the prompts, there are four actions that can be taken following the question mark prompt:

? (CR) - Entering a carriage return indicates that the existing value for the current parameter is acceptable; go on to the next parameter.

? . - Entering a period indicates that this execution of the IOC command must be terminated now, without asking for more parameters.

? ^ - Entering a caret symbol indicates that a previous parameter requires a change and will logically back up one parameter each time it is generated (until the first entry is reached, where it will remain until one of the other responses is received).

? <data> - Entering the appropriate data requested (followed by a carriage return or ENTER). Often the parameters are checked for valid options (i.e, Y or N).

NOTE

Care should be taken when using this diagnostic tool. Portions of the operating system could be destroyed if an incorrect area of a disk were modified.

EXAMPLE	COMMENT
TENbug 2.x > IOP	Request physical disk I/O to read a routine.
READ OR WRITE (R/W)=.....R ? (CR)	Use default of read.
MEMORY ADDRESS FOR DISK I/O=.$00000000 ? 1000	Change $0 default to $1000.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? ^	That isn't correct; back up one parameter.
MEMORY ADDRESS FOR DISK I/O=.$00001000 ? A00	Change $1000 to $A00.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? 2	Change default drive 0 to drive 2.
CONTROLLER NUMBER=.....$00 ? (CR)	Use controller 0.
FIRST BLOCK NUMBER=.$00000000 ? 500	Change block number to $500.
NUMBER OF (256 BYTE) BLOCKS=.....$0000 ? 1	Read one 256-byte block.
ARE YOU SURE? (Y/N) ? Y	Last change? Yes, all is ready.
"R" COMPLETE	The read is complete.
TENbug 2.x > MD A00 30	Display data read from disk (only first $30 bytes).
00A00 41 F8 10 00 20 3C 00 00 02 FF 11 00 51 C8 FF FC	Ax.. <.....QH.:
000A10 60 EE 4E 71 4E 71 4E 71 4E 71 4E 71 4E 71 4E 71	'nNqNqNqNqNqNqNq
000A20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
TENbug 2.x > M A12;L	Make change to RAM where routine was loaded.
000A12 4E714E71 ? -1	
000A16 4E714E71 ? -1	
000A1A 4E714E71 ? -1	
000A1E 4E710000 ? -1	
000A22 00000000 ? .	
TENbug 2.x > IOP	Request physical disk I/O to write to a disk.
READ OR WRITE (R/W)=.....R ? W	Change to W for write.
MEMORY ADDRESS FOR DISK I/O=.$00000A00 ? (CR)	Use previous address.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$02 ? (CR)	Drive is fine; no change.
CONTROLLER NUMBER=.....$00 ? (CR)	No change.
FIRST BLOCK NUMBER=.$00000500 ? (CR)	No change.
NUMBER OF (256 BYTE) BLOCKS=.....$0001 ? (CR)	No change.
ARE YOU SURE (Y/N) ? Y	Last chance; are you sure? Yes, all is ready.
"W" COMPLETE	Write is complete.
TENbug 2.x >	

IOT

The IOT command allows the user to change the configuration of the RWIN1 controller. When invoked, this command prompts for the information required to perform the configuration command.

Depending on the type of drive there are varying parameters required.

WINCHESTER HARD DISK	5.25-INCH FLOPPY DISK
Drive number	Drive number
Controller number	Controller number
Sector size	Sector size
Number of heads	Number of heads
Number of cylinders	Number of cylinders
Number of sectors per track	Number of sectors per track
	Motorola/IBM format
	Single- or double-sided media
	Single- or double-track density
	Single- or double-data density

The IOT command will present the appropriate questions based upon which drive has been specified. There are four actions that can be taken following a question mark prompt:

? (CR) - Entering a carriage return indicates that the existing value for the current parameter is acceptable; go on to the next parameter.

? . - Entering a period indicates that this execution of the IOC command must be terminated now, without asking for more parameters.

? ^ - Entering a caret symbol indicates that a previous parameter requires a change and will logically back up one parameter each time it is entered (until the first entry is reached, where it will remain until one of the other responses is received).

? <data> - Entering the appropriate data requested (followed by a carriage return or ENTER). Often the parameters are checked for valid options (i.e, Y or N).

Appropriate configuration information for specific disk types is listed in the "Mass Storage" chapter of the VERSAdos to VME Hardware and Software Configuration User's Manual.

EXAMPLE	COMMENT
TENbug 2.x > IOT	Teach RWIN1 controller a new configuration.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$02 ? 00	Select drive 0.
CONTROLLER NUMBER=.....$00 ? (CR)	Use default controller 0.
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ? .	Change of heart ... start over again.
TENbug 2.x > IOT	Invoke the IOT command again.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? 02	Configure drive 2 (first floppy).
CONTROLLER NUMBER=.....$00 ? (CR)	No change.
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ? (CR)	No change.
NUMBER OF HEADS, (ON THIS DRIVE)=.....$02 ? (CR)	No change.
NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0050 ? (CR)	No change.
SECTORS PER TRACK=.....$10 ? 8	Change from 16 sectors/track to 8.
MOTOROLA/IBM FORMAT (M/I)=.....I ? (CR)	No change.
SINGLE/DOUBLE SIDED MEDIA (S/D)=.....D ? (CR)	No change.
SINGLE/DOUBLE TRACK DENSITY (S/D)=.....D ? (CR)	No change.
SINGLE/DOUBLE DATA DENSITY (S/D)=.....D ? S	Change to single-data density.

```
TENbug 2.x > IOT

  DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.......$02 ? 0
                    CONTROLLER NUMBER=.......$00 ? (CR)
SECTOR SIZE(0=128,1=256,2=512,3=1024)=.........1 ? (CR)
     NUMBER OF HEADS, (ON THIS DRIVE)=.......$01 ? (CR)
 NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0001 ? ^
     NUMBER OF HEADS, (ON THIS DRIVE)=.......$01 ? 2
 NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0001 ? 132
                    SECTORS PER TRACK=.......$00 ? 20

TENbug 2.x >
```

Invoke IOT to change configuration of hard disk to allow use as a 5-megabyte hard disk.

Oops, passed the number of heads parameter; back up.
Change to 2 heads.
Change to $132 cylinders.
And $20 (32) sectors/track.

NOTE: These parameters are specifically for the 5-megabyte Winchester hard disk. To find out what a particular Winchester hard disk requires for configuration, boot the operating system and, while "inactive" (not updating critical files), press the ABORT button. Then enter an IOT command to see how the boot device is configured. If all is well, enter GO and continue with operating system control.

LO[<port number>] [;<options>] =<text>

The LO command prepares the VME/10 to receive S-records from the designated <port number> and then transmits the <text> following the = sign to the system connected to the <port number> indicated. As the S-records are received, the checksums are verified and the data placed into memory. If the automatic relative offset register, R0, contains a nonzero value, this offset is added to the address contained within the S-record before the data is moved into memory.

The following options are supported:

;-C Ignore validation of the checksum on each S-record while loading.

;X Echo the S-records read to the VME/10 built-in terminal. Different environments may dictate that the ;X option not be used. If printer attach is in effect, the data cannot be displayed upon the screen (and then printed on the printer) before the next record arrives. Thus data can be missed.

This command requires that a dual serial I/Omodule (MVME400) be available on the I/O Channel. Note that serial ports 2 and 1 on the MVME400 correspond to TENbug ports 2 and 3, respectively.

Default source is the TENbug port 2. Specifying LO<port number> allows the input to be received from other ports.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 2 (MVME400 port 2 - 7201/B).
1	Specifies TENbug port 1 (VME/10 built-in terminal/keyboard).
2	Specifies TENbug port 2 (MVME400 port 2 - 7201/B).
3	Specifies TENbug port 3 (MVME400 port 1 - 7201/A).

EXAMPLE	COMMENT
TENbug 2.x > BF A00 F00 2020 PHYSICAL ADDRESS=00000A00 00000F00	Fill RAM with spaces.
TENbug 2.x > MD B00	Display RAM.
000B00 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
TENbug 2.x > .R0 100	Set automatic relative offset register to $10 to reposition S-records to be loaded by + $100.
TENbug 2.X > LO =DU A00 A80 DU A00 A80	Load S-records from $A00 to $A80 (into $B00 to $B80).
TENbug 2.x > .R0 0+R7	Reset automatic relative offset register to zero.
TENbug 2.x > .R0 .R0=00000000	
TENbug 2.x > MD A00	Display memory at $A00 to verify S-records not loaded here.
000A00 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	
TENbug 2.x > MD B00 7F	Display memory at offset $100 from source location.

```
000B00   00 01 02 03 04 05 FF FF  08 09 0A 0B 0C 0D 0E 0F  ................
000B10   10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  ................
000B20   20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F   !"#S%&'()*+,-./
000B30   30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
000B40   40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
000B50   50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
000B60   60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  'abcdefghijklmno
000B70   70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}".
```

TENbug 2.x > BF A00 B00 4161 PHYSICAL ADDRESS=00000A00 00000B00	Fill RAM with pattern.
TENbug 2.x > MD A00	Display destination memory.
000A00 41 61 41 61 41 61 41 61 41 61 41 61 41 61 41 61 AaAaAaAaAaAaAaAa	

```
TENbug 2.x > LO ;X=DU A00 A80
```

Enter Load command specifying ;X option (echo S-records to CRT as memory is being loaded).

```
DU A00 A80
PHYSICAL ADDRESS=00000A00 00000A80
```

Notice S0, S1, and S9 records are displayed upon screen. (Refer to warning in command description about timing restrictions with the ;X option.)

```
S0030000FC
S1130A00000102030405FFFF08090A0B0C0D0E0F79
S1130A1010111213141516171819 1A1B1C1D1E1F5A
S1130A2020212223242526272829 2A2B2C2D2E2F4A
S1130A3030313233343536373839 3A3B3C3D3E3F3A
S1130A4040414243444546474849 4A4B4C4D4E4F2A
S1130A5050515253545556575859 5A5B5C5D5E5F1A
S1130A6060616263646566676869 6A6B6C6D6E6F0A
S1130A7070717273747576777879 7A7B7C7D7E7FFA
S1040A8080F1
S90300000FC
```

```
TENbug 2.x > MD A00 80
```

Display memory containing downloaded data.

```
000A00   00 01 02 03 04 05 FF FF  08 09 0A 0B 0C 0D 0E 0F  ................
000A10   10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  ................
000A20   20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F   !"#S%&'()*+,-./
000A30   30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
000A40   40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
000A50   50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
000A60   60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  'abcdefghijklmno
000A70   70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}~.
```

NOTE

The host system used to create and transmit the S-records was an MC68000 Educational Computer Board (MEX68KECB).

MD[<port number>] <address> [<count>][;<options>]

The MD command displays a portion of memory which begins at <address> and extends for the number of bytes or lines given as <count>. There are two formats that can be requested with the MD command.

a. The dump format begins each line with the starting or next hexadecimal memory address followed by 16 hex bytes per line with the ASCII equivalent shown to the right. The number of lines varies with the <count> entered (or default). There are no partial lines. If the byte count ends in the middle of a line, the complete line is displayed. (Default byte <count> is $10.)

b. The disassembler format provides:

   1. The starting or next hexadecimal memory address.

   2. The object code displayed in hexadecimal.

   3. The M68010 source statement that will assemble into the object code as described in 2. above.

   If the operation code is not valid, a "Define Constant" is constructed for one word. Notice that <count> for the disassembler mode is a number of source lines to be disassembled and displayed, not the number of bytes. (Default line <count> is $10.)

Default destination is the console terminal. Specifying MD<port number> allows the output to be directed to another port.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 1 (VME/10 built-in terminal/keyboard).
1	Specifies TENbug port 1 (VME/10 built-in terminal/keyboard).
2	Specifies TENbug port 3 (MVME400 port 1 - 7201/A).
3	Specifies TENbug port 2 (MVME400 port 2 - 7201/B).
4	Specifies TENbug port 4 (MVME410 port 1 - PIA/A).
5	Specifies TENbug port 5 (MVME410 port 1 - PIA/B).

Options supported are the disassembler and the screen option.

;DI Requests the disassembler option. The <count>, if provided, is a line count (default is $10).

;S Requests the display of a full screen of memory (16 lines of display in either dump or disassembler format). Notice that the default for disassembly is $10 (or 16 decimal) anyway. If the <count> and ;S option are both entered within the same MD command, the ;S option has priority.

All combinations are valid (e.g., ;DIS, ;SDI, ;S DI, ;DI S).

The MD command has a quick scroll facility that lets the terminal operator press CR repeatedly following the initial MD command. In the past, all but the first display were automatically 16 lines long. To enable control blocks to be examined conveniently, the <count> (either bytes or lines) is used for each iteration.

```
EXAMPLE                                  COMMENT

TENbug 2.x  > MD 10000 30                Display memory of a small routine in dump
                                         format.

010000   41 F8 10 00 20 3C 00 00  02 FF 11 00 51 C8 FF FC  Ax.. <......QH.:
010010   60 EE FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  'n..............
010020   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................

TENbug 2.x  > MD 10000 7;DI              Display memory of the same routine in
                                         disassembler format.

010000   41F81000            LEA.L    $00001000,A0
010004   203C000002FF        MOVE.L   #767,D0
01000A   1100               MOVE.B   D0,-(A0)
01000C   51C8FFFC           DBF.L    D0,$01000A
010010   60EE               BRA.S    $010000
010012   FFFF               DC.W     $FFFF
010014   FFFF               DC.W     $FFFF

TENbug 2.x  > MD A00                     Display memory without a <count>.

000A00   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................

TENbug 2.x  > (CR)                       Enter a carriage return; the next <count>
                                         bytes are displayed (default is $10).

000A10   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................
```

```
TENbug 2.x  > MD A00;S        Display  a  full  screen  of  hexadecimal  data.

000A00   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A10   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................
000A20   08 09 0A 0B 0C 0D 0E 0F  10 11 12 13 14 15 16 17  ................
000A30   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A40   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................
000A50   08 09 0A 0B 0C 0D 0E 0F  10 11 12 13 14 15 16 17  ................
000A60   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A70   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................
000A80   08 09 0A 0B 0C 0D 0E 0F  10 11 12 13 14 15 16 17  ................
000A90   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000AA0   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................
000AB0   08 09 0A 0B 0C 0D 0E 0F  10 11 12 13 14 15 16 17  ................
000AC0   18 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000AD0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000AE0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000AF0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................

TENbug 2.x > MD A00 18        Display memory with a <count> of $18.

000A00   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A10   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................

TENbug 2.x > (CR)             Enter a carriage return to display the next <count>
                              bytes (starting where the last request ended, even
                              if in the middle of the line).

000A18   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A28   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................

TENbug 2.x > (CR)

000A30   00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ................
000A40   10 11 12 13 14 15 16 17  00 01 02 03 04 05 06 07  ................

TENbug 2.x >
```

M[M] <address>[;<options>]

The function of the Memory Modify (M or MM) command is to change data in memory. An address and options are specified on the initial command line.

For convenient viewing and changing of object data, four variations of data updating capability are offered. These are enhanced by five options: the data size options, word and longword (the default size is byte); odd or even address access options (byte-size only); and a nonverification option for write-only operations. Action provided by an option specified on the initial command line is utilized in all four data updating submodes and remains in force until the M command is exited.

The five memory change mode options are:

- ;W Set size to word (i.e., two bytes).
- ;L Set size to longword (i.e., four bytes).
- ;O Set size to byte; access only odd addresses.
- ;V Set size to byte; access only even addresses.
- ;N No verification. Do not read data after updating. If used, ;N must be preceded by one of the above options (the semicolon (;) is required between multiple options).

When the memory change mode is entered on execution of the initial command line, object data in the specified locations is displayed in hexadecimal format, and the M command prompt (?) is presented at the right of the data. The data can then be changed using any of the subcommands described below. If desired, the action of the subcommand can be obtained without entering new data. For example, the contents of the preceding location(s) can be viewed by typing "^(CR)" alone after the ? prompt, or the M command can be exited by typing ".(CR)".

[<data>](CR)	Update location and sequence forward.
[<data>]^(CR)	Update location and sequence backward.
[<data>]=(CR)	Update location and reopen same location.
[<data>].(CR)	Update location and terminate.

Disassemble/Assemble Mode (the ;DI option)

On execution of an initial M command line with the ;DI option selected, the disassemble/assemble mode is entered. Starting from the specified location, data is disassembled into a source instruction line, and both object data (in hexadecimal) and the source line are displayed. The M command prompt (?) is displayed to the right of the disassembled source line. If desired, a new source instruction may be typed and assembled to replace the existing instruction (the first character must be a space, which is recognized as the label field delimiter by the TENbug one-line assembler). Assembly is initiated by typing a carriage return. After assembly and updating, data in the following locations is disassembled and the next source line displayed. Note that the update and sequence backward (^(CR)) and the update and reopen the same location (=(CR)) features are not available in the disassemble/assemble mode. Typing ".(CR)" while in this mode provides exit from the M command.

EXAMPLE		COMMENT
TENbug 2.x	> M 10000;L	Memory modify location $10000 a longword at a time.
010000	00000200 ? (CR)	No change to this longword.
010004	FFFFFFFF ? -5.	Change this longword to $-5 (or $FFFFFFFB) and stop.
TENbug 2.x	> MM 10008;W;N	Modify memory location $10008 a word at a time; do not read data after updating.
010008	? 1111	Place a word of 1's into this location.
01000A	? 2222	Place a word of 2's into this location.
01000C	? 3333	Place a word of 3's into this location.
01000E	? 4444	Place a word of 4's into this location.
010010	? ^	Back up one word.
01000E	? 5555	Place a word of 5's over the 4's in this location.
010010	? 6666.	Place a word of 6's into this location.
TENbug 2.x	> M 20001;N;O	Place a byte into the odd locations only, without reading the data.
020001	? 1=	Place a 1 into this memory location and remain at
020001	? 8=	the same location. This technique is very useful
020001	? 1=	for debugging I/O devices.
020001	? 7=	Place a 7 at this memory location and remain at the same location.
020001	? .	Exit the MM command.

```
TENbug 2.x  > M 49528;DI        Modify memory starting at $49528 using the
                                disassembler.

049528 48B800010406             MOVEM.W D0,$0406 ? (CR)
04952E 40F80406                 MOVE.W  SR,$0406 ? (CR)
049532 48E7FFFE                 MOVEM.L D0-D7/A0-A6,-(A7) ? (CR)
049536 4FF8095A                 LEA.L   $095A,A7 ? MOVE.B

049536 4FF8095A                 X?(CR)                    (response to incomplete,
049536 4FF8095A                 LEA.L   $905A,A7 ? (CR)       incorrect entry)
04953A 1600                     MOVE.B  D0,D3 ? (CR)
04953C 04444281                 SUB.W   #17025,D4 ? .

TENbug 2.x  >
```

NOTE

Refer to Chapter 5 for more information about the assembler/disassembler.

MS <address> <data>

The Memory Set (MS) command changes the contents of memory. The data entered is placed at the location specified by <address>. If the data entered requires word alignment and <address> is not even, the byte at <address> is bypassed and the data is placed in the next even address.

Memory Set allows both hexadecimal and ASCII string data within the same line. The length of hexadecimal values can also vary. A space is used to delimit each field, and an apostrophe must be used to enclose each ASCII string.

Notice that lowercase is supported within the ASCII string. TENbug's command and parameter parser automatically converts all lowercase input into uppercase. The only exceptions are the ASCII strings within apostrophes and the data entered while in transparent mode. This provides support for users wishing to use the terminal in lowercase. The commands and operands will work because they are converted to uppercase, and, where lowercase is specifically needed, it is supported (BS, MS, and TM).

The maximum number of bytes that can be entered with one MS <address> <data> command is limited to the size of the command line buffer, or 128 bytes. When the character in the last position of the first line is entered, an automatic CR/LF is sent to the display allowing the user to continue and still read the input characters entered.

EXAMPLE COMMENT

TENbug 2.x > MD A00 30 Display memory at start before the MS command.

```
000A00   20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
000A10   20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
000A20   20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
```

TENbug 2.x > MS A01 'MS Test' 41 61 42434445 20313233 'End of Data'

Enter ASCII string, two bytes, two longwords, and another string. (Notice lowercase input.)

TENbug 2.x > md a00 30 Now display results with the MD command. Notice command was entered in lowercase.

```
000A00   20 4D 53 20 54 65 73 74  20 41 61 42 43 44 45 20   MS Test AaBCDE
000A10   31 32 33 45 6E 64 20 6F  66 20 44 61 74 61 2E 20   123End of Data.
000A20   20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
```

TENbug 2.x > BF A00 B00 4161

Fill memory with Aa so changes will stand out.

PHYSICAL ADDRESS=00000A00 00000B00

```
TENbug 2.x  > MD A70          Display memory near end of what a full buffer can
                              change.

000A70   41 61 41 61 41 61 41 61  41 61 41 61 41 61 41 61

TENbug 2.x > MS A00 'This example shows that the MS command allows more than one
line of data in the buffer at a time.  Note the CR/LF sent'.

                              Enter long ASCII string.

TENbux 2.x  > md a00 80       Display a full 128 bytes.

000A00   54 68 69 73 20 65 78 61  6D 70 6C 65 20 73 68 6F  This example sho
000A10   77 73 20 74 68 61 74 20  74 68 65 20 4D 53 20 63  ws that the MS c
000A20   6F 6D 6D 61 6E 64 20 61  6C 6C 6F 77 73 20 6D 6F  ommand allows mo
000A30   72 65 20 74 68 61 6E 20  6F 6E 65 20 6C 69 6E 65  re than one line
000A40   20 6F 66 20 64 61 74 61  20 69 6E 20 74 68 65 20   of data in the
000A50   62 75 66 66 65 72 20 61  74 20 61 20 74 69 6D 65  buffer at a time
000A60   2E 20 4E 6F 74 65 20 74  68 65 20 43 52 2F 4C 46  . Note the CR/LF
000A70   20 73 65 6E 74 61 41 61  41 61 41 61 41 61 41 61   sentaAaAaAaAaAa

TENbug 2.x  >
```

OF

The OF command displays the offsets used to assist with relocatability and position-independent code.

Linked segments of code will each have a different load address or offset. For user convenience, seven general purpose offsets (.R0-.R6) are provided. Offset .R7 is always zero, which provides a convenient technique for entering an address without an offset. If no value is assigned to one of the general purpose offsets, it will have the default value of zero.

Unless another offset is entered, each command that expects an address parameter automatically adds offset R0 to the entered address -- that is, if R0 = 1000, the following commands are the same:

```
BR 10           (10 + 1000)    Offset R0 added by default.
BR 10+R0        (10 + 1000)
BR 1010+R7      (1010 + 0)     R7 is always zero.
```

The physical address for each of these commands is 1010.

```
EXAMPLE                              COMMENT

TENbug 2.x > .R1 1000                Set offset R1.

TENbug 2.x > .R3 33000

TENbug 2.x > .R4 440000

TENbug 2.x > .R5 0                   Reset offset R5.

TENbug 2.x > .R6 -1

TENbug 2.x > OF                      Display offsets.

R0=00000000 R1=00001000 R2=00000000 R3=00033000
R4=00440000 R5=00000000 R6=FFFFFFFF R7=00000000

TENbug 2.x > .R0 1200                Set offset R0.

TENbug 2.x > MM 10
000010+R0 61 ? .                     Offset R0 is added to the address.

TENbug 2.x > MM 10+R7                R7 is always 0 and, when entered, overrides
000010 00 ? .                        R0.

TENbug 2.x >
```

To set R0 to 0 after it has been set to a non-zero value, use the command ".R0 0+R7". The command ".R0 0" will not alter R0.

```
PA[<port number>]
NOPA
```

The PA command allows the user to attach the line printer so that information sent to the console terminal will also be printed. (The printer is connected to a port on a dual 16-bit parallel port I/Omodule, MVME410, attached to a printer board which communicates with the VME/10 system via the I/O Channel. Refer to the initial setup instructions in the VME/10 Microcomputer System Diagnostics Manual.) The board has two PIA's. TENbug takes the lower addressed PIA as port 4 and the higher as port 5. Default is always port 4.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 4 (MVME410 port 1 - PIA/A).
4	Specifies TENbug port 4 (MVME410 port 1 - PIA/A).
5	Specifies TENbug port 5 (MVME410 port 2 - PIA/B).

The printer can also be called by the Memory Display (MD4 or MD5) command.

If the printer is deselected or not ready, the message PRINTER NOT READY will be sent to the console terminal. TENbug will wait until the printer is ready or the BREAK key is pushed.

NOTES

1. Execution of this command when no dual 16-bit parallel port module is connected to the I/O Channel may require pressing the RESET pushbutton in order to return control to TENbug.
2. Only one printer port can be attached at a time.

The NOPA command allows the user to detach the line printer at port 4 or port 5 from the console terminal. Output will then be displayed on the console terminal only; it will not be printed.

See also: MD

```
EXAMPLE                     COMMENT

TENbug 2.x > PA5

TENbug 2.x > MD 800 90

000800    FF FF 24 18 FF 7F 0C 00  FF 7F 30 04 FF FF 00 00  ..$.......0.....
000810    FF 31 FF FF FF 01 FF FF  FF 26 FF FF 7F 22 FF FE  .1.......&...".~
  .
  .                         Output is displayed on console terminal and printed
  .                         at port 5.

TENbug 2.x > NOPA           Future output will be displayed on console terminal
                            only.
TENbug 2.x >
```

PF[<port number>]

The Port Format (PF) command displays or changes the automatic null insertion for each character and for each carriage return. Two additional pieces of information are displayed:

a. The address of RAM used to initialize the two 7201 serial ports when the optional MVME400 module is installed.

b. The address of the TENbug option bytes in RAM (affecting transparent mode parameters and the environment while tracing).

There are two responses when the Port Format (PF) command is entered. If the optional MVME400 module is installed, the character null count and the carriage return null count are displayed. If the dual serial port is not installed, a message notifying the user is displayed. In either event, the RAM locations for 7201 initialization and the TENbug options are then displayed.

```
TENbug 2.x > PF

OPTIONAL MVME400 NOT INSTALLED

7201 RAM @ xxxxxx
OPTIONS  @ YYYYYY
```

```
TENbug 2.x > PF

             PORT2   PORT3
CHAR NULL= 00          00
C/R NULL = 00          00

7201 RAM @ xxxxxx
OPTIONS  @ YYYYYY
```

NOTE

TENbug port 2 corresponds to MVME400 port 2, (7201/B).
TENbug port 3 corresponds to MVME400 port 1, (7201/A).

If either PF2 or PF3 is entered, the CHAR NULL count (number of nulls to be inserted after each character) is displayed followed by a ? prompt. The user may then enter a new count or enter a carriage return to move on to the next field without change to this parameter. C/R NULL (or the number of nulls to be inserted after each carriage return) is the second and last parameter displayed for the user to accept or modify as desired.

If the configuration of either or both of the 7201 serial ports requires modification (e.g., number of stop bits, parity), the PF command cannot directly make the changes. Instead, the address where RAM is located for the initialization of the ports is displayed on the general display. A Memory Modify command can be used to alter these hex locations. After a change has been made, the serial ports must be forced to reinitialize by entering a PF <port number> command. This will place the new RAM data into the 7201 registers. The contents of the 7201 RAM area at power up or cold start are as follows.

RELATIVE OFFSET	PORT2 (7201/B)	PORT3 (7201/A)	DESCRIPTION OF MOVE	FUNCTION
0000	18	18	$18 to ctl reg 0	Issue channel reset
0002	02	02	$02 to ctl reg 0	Set up R2 for a read
0004	00	00	$00 to ctl reg 2	Disable DMA & interrupt vector
0006	14	14	$14 to ctl reg 0	Set up R4 for a read & reset
0008	44	44	$44 to ctl reg 4	# stop bits & parity
000A	01	01	$01 to ctl reg 0	Set up R1 for a read
000C	00	00	$00 to ctl reg 1	Disable interrupts
000E	05	05	$05 to ctl reg 0	Set up R5 for a read
0010	EA	EA	$EA to ctl reg 0	Set bits/character (write)
0012	03	03	$03 to ctl reg 0	Set up R3 for a read
0014	E1	E1	$E1 to ctl reg 3	Set bits/character (read) AUTO ENABLE

See the examples following this discussion for 7201 reconfiguration.

Three-wire interfaces (TXD, RXD, GND) and other interfaces that do not provide full modem flow control (DSR, RTS, CTS, DCD, DTR) can be supported by disabling the AUTO ENABLE feature within the 7201 serial controller (register 3, bit 5). In this mode of operation there will not be any flow control in either direction with the modem control lines.

If flow control is desired, for transmission of S-records to and from an EXORmacs through an MCCM for example, the AUTO ENABLE feature can be enabled. In addition there are specific jumpers that must be in place to support this configuration. For more information on the serial port jumper configuration refer to the paragraph in the MVME400 user's manual entitled "CTS Control Headers".

The following jumper configurations provide flow control with modem control lines:

TENbug PORT 2
MVME400 PORT 2

J7

1-3, 2-4

MVME400 J9 IS JUMPERED
(TO TERMINAL)

EXORmacs
MCCM (TO MODEM)

TENbug PORT 3
MVME400 PORT 1

J16

1-3, 2-4

MVME400 J15 IS JUMPERED
(TO TERMINAL)

EXORmacs
MCCM (TO MODEM)

For compatibility, the format of the "options" RAM area is the same as that used in the Educational Computer Board (ECB) firmware TUTOR 1.x. Several items are not supported within TENbug, but those that are reside in the same offsets. The XON/XOFF characters and auto line feed control are not supported within TENbug 2.x; as a result the first three bytes in the "options" area are unused.

Offset $3 within the "options" RAM, a non-$00 byte will inhibit the register display at breakpoints and traces.

Offset $4 contains the Transparent Mode (TM) "trailing" character (the character transmitted through port 2 to inform a host system that any characters that might be left over from this port are to be flushed). If this character is $00 or NUL, there will not be any transmission of a character upon termination of transparent mode.

Offset $5 contains the character that when entered will terminate transparent mode and return control to TENbug.

See also: TM

EXAMPLE

```
TENbug 2.x  > PF

        PORT2  PORT3
CHAR NULL= 00     00
C/R NULL = 00     00

7201 RAM @ 001314
OPTIONS  @ 001152

TENbug 2.x  > M 1314;W
001314   1818 ? (CR)
001316   0202 ? (CR)
001318   0000 ? (CR)
00131A   1414 ? (CR)
00131C   4444 ? 4144
00131E   0101 ? (CR)
001320   0000 ? (CR)
001322   0505 ? (CR)
001324   EAEA ? AAEA
001326   0303 ? (CR)
001328   E1E1 ? 61E1.

TENbug 2.x  > PF
CHAR NULL= 00 ? (CR)
C/R NULL = 00 ? (CR)
TENbug
```

COMMENT

Reconfigure for 7 bits/character with even parity.

Display the address of the 7201 initialization RAM.

(For this example, the address of RAM is $1314. This address will vary by TENbug version).

Modify areas affecting transmitted/received bits/character, number of stop bits, and parity.

NOTE

Only the first byte is modified within each pair of bytes. In this example, only TENbug port 2 (MVME400 port 2 7201/B) is being changed.

After the RAM has been changed, the 7201 must be forced to initialize. PF2 or PF3 will issue serial port initialization after accepting the two optional variables.

In the example below, the AUTO ENABLE bit in control register 3 is turned on for ports 2 and 3 to provide modem control line flow control.

```
TENbug 2.x  > M 1314;W
001314   1818 ? (CR)
001316   0202 ? (CR)
001318   0000 ? (CR)
00131A   1414 ? (CR)
00131C   4144 ? 4444
00131E   0101 ? (CR)
001320   0000 ? (CR)
001322   0505 ? (CR)
001324   AAEA ? EAEA
001326   0303 ? (CR)
001328   61E1 ? C1C1.
```

Reconfigure both ports to support a 3-wire interface (TXD, RXD, and ground only).

NOTE

In this example both the first and second bytes are modified (contents of control register 3), disabling the port 2 and port 3 AUTO ENABLE feature.

```
TENbug 2.x  > PF2
CHAR NULL= 00 ? (CR)
C/R NULL = 00 ? (CR)
TENbug
```

Reinitialize the serial ports using the modified RAM.

The following are examples both with and without register information within trace and breakpoints.

```
TENbug 2.x  > BR A10
```

Use the "options" bytes displayed by PF.

```
BREAKPOINTS
000A10    000A10
```

Set breakpoint at end of a routine.

```
TENbug 2.x  > .PC A00
```

Set program counter at start of routine.

```
TENbug 2.x  > T
PHYSICAL ADDRESS=00000A00
PC=00000A04 SR=2704=.S7..Z.. USP=FFFFFFFF SSP=00000B00 VBR=00000000 SFC=2 DFC=7
D0-7 0000FFFF 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000B00
 PC=000A04    203C000002FF         MOVE.L  $767,D0
```

Trace one instruction. (Note: Complete register display for each trace and breakpoint.)

```
TENbug 2.x :> (CR)                                          Assume Trace (CR after previous trace).
PHYSICAL ADDRESS=00000A04
PC=00000A0A SR=2700=.S7..... USP=FFFFFFFF SSP=00000B00 VBR=00000000 SFC=2 DFC=7
D0-7 000002FF 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000B00
 PC=000A0A    1100                MOVE.B  D0,-(A0)


TENbug 2.x :> GO                                            Allow free running of routine to be stopped by
PHYSICAL ADDRESS=000000A0A                                  breakpoint.

AT BREAKPOINT                                               Once again, full register display.
PC=00000A10 SR=2704=.S7..Z.. USP=FFFFFFFF SSP=00000B00 VBR=00000000 SFC=2 DFC=7
D0-7 0000FFFF 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-7 00000D00 00000000 00000000 00000000 00000000 00000000 00000000 00000B00
 PC=000A10    60EE                BRA.S   $000A00


TENbug 2.x  > PF                                            Request "options" RAM location with the PF command.

        PORT2  PORT3
CHAR NULL= 00     00
C/R  NULL= 00     00

7201 RAM @ 001314
OPTIONS  @ 001152                                           Note:  Address will vary from example.


TENbug 2.x > M 1152                                         Use address to display, then modify, the fourth byte
001152   00 ? (CR)                                          (offset $3) from zero to nonzero to suppress complete
001153   00 ? (CR)                                          register display.
001154   00 ? (CR)
001155   00 ? 1.
```

```
TENbug 2.x > .PC A00

TENbug 2.x > T
PHYSICAL ADDRESS=00000A00
 PC=000A04     203C000002FF      MOVE.L #767,D0

TENbug 2.x :> (CR)
PHYSICAL ADDRESS=00000A04
 PC=000A0A     1100              MOVE.B D0,-(A0)

TENbug 2.x :> (CR)
PHYSICAL ADDRESS=00000A0A
 PC=000A0C     51C8FFFC          DBF.L D0,$000A0A

TENbug 2.x  :> GO
PHYSICAL ADDRESS=00000A0C

AT BREAKPOINT
 PC=000A10     60EE              BRA.S $000A00

TENbug 2.x  >
```

Restart routine again.

Trace one instruction.

Trace assumed when previous trace followed with a carriage return.

Note: Only physical address where execution begins; the relative displacement and offset (if used) for the PC; the hex value of the instruction; followed by the disassembled instruction.

Breakpoint displays the same abbreviated information.

TM [<exit character> [<trailing character>]]

The TM command, together with an MVME400 dual RS-232C serial port I/Omodule, provides terminal support in a dumb terminal fashion until the exit character is received.

Multiple baud rates are supported, however, because the processor supports the CRT display in a polling fashion; each time the screen requires scrolling there is a delay that can result in lost characters. Baud rates down through 4800 baud can miss characters on the display.

The <exit character> is entered right after the TM command itself, though an optional space is permitted. Note that CTRL-X, CTRL-D, CTRL-H, CTRL-J, and CTRL-M cannot be specified as exit or trailing characters from the TM command. These characters provide control for the terminal (e.g., line delete, redisplay the line, backspace) and will not be passed to the TM command from the terminal handler. All other characters, both CTRL and non-CTRL, can be entered. (Default <exit character> is CTRL-A.)

The <trailing character> is transmitted to the host port upon receipt of the <exit character>. With systems using VERSAdos the standard trailing character is a CTRL-X, which cancels anything that might have made it to an input buffer. By default CTRL-X is specified as the trailing character so there is no problem using it (though if another trailing character were selected, CTRL-X could not be entered to reinstate it).

There is an alternate way of setting the exit and trailing characters. Since they are stored in RAM, any user knowing the location of these characters could use a Memory Modify (MM) command to alter those values. By issuing a Port Format (PF) command the address of the TENbug option bytes is displayed at the end of the display.

```
Trailing character is located at offset $4
    Exit character is located at offset $5
```

In systems where no data at all must be sent upon exit of transparent mode, the trailing character can be changed to $00 using Memory Modify within the option bytes. With the trailing character as NUL, there is no data sent at exit time.

Only TENbug port 2 (MVME400 port 2, 7201/B) is supported as the transparent port. Even though the other ports may have output directed to them (e.g., MD3, DU3), they are not supported for transparent mode.

EXAMPLE	COMMENT
TENbug 2.x > TM *TRANSPARENT* EXIT=$01 = CTL A	Request default exit and trailing character by entering only TM. TENbug responds with both hex and CTRL display of exit character.
TENbug 2.x > TM (CTRL-S) *TRANSPARENT* EXIT=$13 = CTL S	Specify CTRL-S as exit character with TM (hold down the CTRL key and press the S key). TENbug responds with hex 13 and CTL-S, verifying what was entered.
TENbug 2.x > PF PORT2 PORT3 CHAR NULL= 00 00 C/R NULL = 00 00 7201 RAM @ 001314 OPTIONS @ 001152	Use PF command to show where the TENbug option variables are located within RAM.
TENbug 2.x > M 1152;W 001152 0000 ? (CR) 001154 0000 ? (CR) 001156 1813 ? 001C.	Use Memory Modify command to display the trailing character ($18) and the current exit character ($13). Then change the trailing character to prevent transmission of any data upon exit (refer to the discussion of null trailing characters) and change exit character to a CTRL-\ (control backslash).
TENbug 2.x > TM *TRANSPARENT* EXIT=$1C = CTL \	Enter a TM command without any operands and the previous values will be used, as changed by Memory Modify.
TENbug 2.x > M 1156;W 001156 001C ? 1801.	Change trailing character back to CTRL-X and exit character to CTRL-A.
TENbug 2.x > TM *TRANSPARENT* EXIT =$01 = CTL A TENbug 2.x >	Enter TM without operands and the CTRL-A exit character is verified.

4.2.33 Trace (TR)

TR
T

T[R] [<count>]

The Trace (T or TR) command executes instructions one at a time, beginning at the location pointed to by the program counter. After execution of each instruction, the MC68010 registers are displayed.

When the trace mode is entered, the prompt includes a colon (i.e., TENbug 2.x :>). While in this mode, typing only a carriage return will cause one instruction to be traced.

Breakpoints and breakpoint counts are in effect during trace.

Trace cannot be used to step through interrupts or exceptions (e.g., TRAP).

COMMAND FORMAT	DESCRIPTION
TENbug 2.x > T	Trace one instruction.
TENbug 2.x :> TR <count>	Trace <count> instructions.
TENbug 2.x :> (CR)	Carriage return (CR) executes next instruction.
TENbug 2.x :> MD 1000	Typing the next command exits trace mode.

NOTE

If the program counter contains an address that falls between the starting and ending addresses of the TENbug program, the warning message .PC within "DEBUGGER" will be returned. Processing will continue with unexpected results if stack pointers and/or registers are not handled properly.

See also: DF, GO, GT, TT

EXAMPLE:

```
TENbug 2.x > .PC 2000

TENbug x.y > TR
PHYSICAL ADDRESS=00002000
PC=00002002 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002002

TENbug x.y :> (CR)
PHYSICAL ADDRESS=00002002
  PC=00002004
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002004

TENbug x.y :> T 2
PHYSICAL ADDRESS=00002004
PC=00002006 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002006
PC=00002008 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002008

TENbug 2.x :>
```

TT <breakpoint address>

The TT command performs the following:

a. Sets a temporary breakpoint at the address specified.

b. Starts program execution in the trace mode.

c. Traces until any breakpoint with a zero count is encountered.

d. Resets the temporary breakpoint.

The temporary breakpoint is not displayed by the BR command.

See also: DF, GO, GT, TR

EXAMPLE

```
TENbug 2.x > .PC 2000

TENbug 2.x > TT 2006
PHYSICAL ADDRESS=00002006
PHYSICAL ADDRESS=00002000
PC=00002002 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002002
PC=00002004 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002004

AT BREAKPOINT
PC=00002006 SR=2700=.S7..... USP=FFFFFFFF SSP=00000800 VBR=00000000 SFC=2 DFC=2
D0-7 00304E71 00002000 C05E2000 00000000 1796AF30 00000020 00000000 00000000
A0-7 00F021CA 00000000 00002000 00000458 00000410 00000551 00000551 00000800
  PC=002006

TENbug 2.x :>
```

VE[<port number>] [;<options>] =<text>

The VE command prepares the VME/10 to receive S-records from the designated <port number> and then transmits the <text> following the = sign to the system connected to the <port number> indicated. As the S-records are received, the bytes are compared one by one with the contents of memory. If all bytes are correct, the TENbug prompt is returned. However, if any data does not compare, a message in the following format is displayed:

S1LLaaaa.-.-.-.-.-.-.-.-.-.-DD.-.-.-.-.-

or

S2LLaaaaaa.-.-.-.-.-.-.-.-.-.-DD.-.-.-.-.-

where:

S1 or S2	Is the S-record type (note size of address).
LL	Is the length of the data contained within the checksum.
aaaa or aaaaaa	Is the address that this S-record verifies.
.-.-.-	Are characters that verify correctly.
DD	Represents the contents of the S-record where data was found to be different.

Refer to Appendix D for a discussion of S-record content.

The following options are supported:

;-C	Ignore validation of the checksum on each S-record while loading.
;X	Echo the S-records read to the VME/10 built-in terminal.

The VE command requires that a dual serial I/Omodule (MVME400) be available on the I/O Channel. Note that serial ports 2 and 1 on the MVME400 correspond to TENbug ports 2 and 3, respectively.

Default source is the TENbug port 2. Specifying VE<portnumber> allows the input to be received from other ports.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 2 (MVME400 port 2 - 7201/B).
1	Specifies TENbug port 1 (VME/10 built-in terminal/keyboard).
2	Specifies TENbug port 2 (MVME400 port 2 - 7201/B).
3	Specifies TENbug port 3 (MVME400 port 1 - 7201/A).

EXAMPLE	COMMENT
TENbug 2.x > BF D00 E00 4161 PHYSICAL ADDRESS=00000D00 00000E00	Initialize RAM to known pattern.
TENbug 2.x > MD D00 000D00 41 61 41 61 41 61 41 61 41 61 41 61 41 61 41 61 AaAaAaAaAaAaAaAa	Display memory.
TENbug 2.x > LO =DU D00 D80 D00 D80	Enter LO command which contains a Dump (DU) command for an Educational Computer Board (ECB).
TENbug 2.x > MD D00 8F	Display RAM to see if data was transferred.

```
000D00   00 01 02 03 04 05 FF FF   08 09 0A 0B 0C 0D 0E 0F  ................
000D10   10 11 12 13 14 15 16 17   18 19 1A 1B 1C 1D 1E 1F  ................
000D20   20 21 22 23 24 25 26 27   28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
000D30   30 31 32 33 34 35 36 37   38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
000D40   40 41 42 43 44 45 46 47   48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
000D50   50 51 52 53 54 55 56 57   58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
000D60   60 61 62 63 64 65 66 67   68 69 6A 6B 6C 6D 6E 6F  'abcdefghijklmno
000D70   70 71 72 73 74 75 76 77   78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}".
000D80   80 61 41 61 41 61 41 61   41 61 41 61 41 61 41 61  .aAaAaAaAaAaAaAa
```

TENbug 2.x > M D78 000D78 78 ? 88.	Alter two bytes with Memory Modify to see if Verify can detect incorrect bytes.
TENbug 2.x > M D1A 000D1A 1A ? 0A.	
TENbug 2.x > VE =DU D00 D80 DU D00 D80 S1130D10.-.-.-.-.-.-.-.-.-.-1A.-.-.-.-.- S1130D70.-.-.-.-.-.-.-.-78.-.-.-.-.-.-.-	Enter the Verify command requesting the same S-records be transmitted. Two errors are detected.
TENbug 2.x > M D1A 000D1A 0A ? 1A.	Use Memory Modify to change the two bytes back to the correct values.
TENbug 2.x > M D78 000D78 88 ? 78.	
TENbug 2.x > VE =DU D00 D80	Invoke the Verify command again; no errors are detected this time.
TENbug 2.x >	

VM

The VM command toggles the high-resolution bit (bit 4 of control register 1), causing VME/10 RAM to be remapped. This enables use of both low- (800 x 300-pixel matrix) and high- (800 x 600-pixel matrix) resolution graphics display mode.

WARNING

WHEN THE VM COMMAND IS USED, ALL VME/10 RAM IS REARRANGED AND MUST BE RELOADED.

The user enters TENbug in the high-resolution mode (bit set to 1), with memory mapped accordingly. The basic TENbug prompt appears:

```
TENbug 2.x >
```

When the VM command is entered (bit changed to 0), the memory map is reorganized for low resolution mode and a modified prompt appears which signifies that low mode is in effect:

```
TENbug 2.x m>
```

However, the actual display matrix does not change. By remapping VME/10 memory, RAM space is provided for the user to modify the display by reprogramming the CRT controller device (MC6845) that defines the video screen.

Entering the VM command again will remap RAM for high-resolution mode.

Descriptions and memory maps for both low- and high-resolution modes appear in the VME/10 Microcomputer System Reference Manual.

EXAMPLE	COMMENT
TENbug 2.x > VM	While in high-resolution mode, use VM command to enter low-resolution mode.
TENbug 2.x m> VM	New prompt indicates low-resolution mode. Enter VM command again to return to high-resolution mode.
TENbug 2.x >	

4.3 COMMAND SUMMARY

The commands and options available to TENbug 2.x users are summarized in Table 4-2.

TABLE 4-2. TENbug Command and Option Summary

COMMAND	DESCRIPTION
[NO]BARS	Draw graphics test pattern.
BF <address1> <address2> <pattern>	Block fill.
BH [<device>][,<controller>]	Boot and halt.
BI <address1> <address2>	Block initialize.
BM <address1> <address2> <address3>	Block move.
BO [<device>][,<controller>][,<string>]	Boot operating system.
[NO]BR [<address>[;<count>]]...	Set and remove breakpoints.
BS <address1> <address2> '<literal string>' <address1> <address2> <data>[<mask>][;<option>]	Block search; options ;B ;W ;L.
BT <address1> <address2>	Block test.
[NO]CH [<bits>]	Alter character display map.
CRT	Alter CRT control registers.
CS [<address1>] [<address2>]	Checksum.
DC <expression>	Data conversion/evaluation.
DF	Display formatted registers.
DU[<port number>] <address1> <address2> [<text>]	Dump memory (S-records).
GD [<address>]	Go direct.

TABLE 4-2. TENbug Command and Option Summary (cont'd)

COMMAND	DESCRIPTION
G[O] [<address>]	Install breakpoints and go.
[NO]GR [<bits>]	Alter graphics display map.
GT <temporary breakpoint address>	Go until address.
HE	Display commands/registers.
IOC	Issue RWIN1 command.
IOP	Issue physical read/write.
IOT	Teach RWIN1 a configuration.
LO[<port number>] [;<options>] =<text>	Load (S-records).
MD[<port number>] <address> [<count>][;<options>]	Memory display; options ;DI ;S.
M[M] <address>[;<options>]	Memory modify; options ;W ;L ;O ;V ;N ;DI.
MS <address> <data>	Memory set (also ASCII).
OF	Offset register display.
[NO]PA[<port number>]	Printer attach/detach.
PF[<port number>]	Port format.
TM [<exit character> [<trailing character>]]	Transparent mode.
T[R] [<count>]	Trace.
TT <breakpoint address>	Trace until address.
VE[<port number>] [;<options>] =<text>	Verify (S-records).
VM	Toggle video map.

TABLE 4-2. TENbug Command and Option Summary (cont'd)

COMMAND	DESCRIPTION
(See description of .<register> commands)	
.A0 - .A7 [<expression>]	Display/set address register.
.D0 - .D7 [<expression>]	Display/set data register.
.DFC [<expression>]	Display/set destination function code.
.PC [<expression>]	Display/set program counter.
.R0 - .R6 [<expression>]	Display/set relative offset register.
.SFC [<expression>]	Display/set source function code.
.SR [<expression>]	Display/set status register.
.SSP [<expression>]	Display/set supervisor stack pointer.
.USP [<expression>]	Display/set user stack pointer.
.VBR [<expression>]	Display/set vector base register.
Key Functions:	
(BREAK)	Abort command or process.
(DEL)	Delete character.
(CTRL-D)	Redisplay line.
(CTRL-H)	Delete character.
(CTRL-W)	Suspend output; any character continues.
(CTRL-X)	Cancel command line.
(<—┘)	Process current/previous command line.

CHAPTER 5

USING THE ASSEMBLER/DISASSEMBLER

5.1 INTRODUCTION

Included as part of the VME/10 TENbug 2.x firmware is an assembler/disassembler function. The assembler/disassembler is an interactive assembler/editor in which the source program is not saved. Each source line is translated into the proper MC68010 machine language code and is stored in memory on a line-by-line basis at the time of entry. In order to display an instruction, the machine code is disassembled, and the instruction mnemonic and operands are displayed. All valid MC68010 instructions are translated.

The VME/10 assembler is effectively a subset of the MC68010 Resident Structured Assembler. It has more limitations than the resident assembler, such as not allowing line numbers and labels; however, it is a powerful tool for creating, modifying, and debugging MC68010 code.

5.1.1 M68010 Assembly Language

The symbolic language used to code source programs for processing by the assembler is called M68010 assembly language. This language is a collection of mnemonics representing:

- Operations
  - MC68010 machine-instruction operation codes
  - Directive (pseudo-op)
- Operators
- Special symbols

5.1.1.1 Machine-Instruction Operation Codes. That part of the assembly language that provides mnemonic machine-instruction operation codes for the MC68010 machine instructions is described in the M68000 16/32-Bit Microprocessor Programmer's Reference Manual. The user should refer to this manual.

5.1.1.2 Directives. The assembly language can contain mnemonic directives which specify auxiliary actions to be performed by the assembler. Directives are not always translated to machine language.

Assembler directives assist the programmer:

- In controlling the assembler output
- In defining data and symbols
- In allocating storage

The VME/10 assembler recognizes only one directive called define constant (DC.W). This directive is used to define data within the program. Refer to paragraph 5.2.4 for a description of this directive.

5.1.2 Comparison with MC68000 Resident Structured Assembler

There are several major differences between the VME/10 assembler and the MC68000 Resident Structured Assembler. The resident assembler is a two-pass assembler that processes an entire program as a unit, while the VME/10 assembler processes each line of a program as an individual unit. Due mainly to this basic functional difference, the capabilities of the TENbug 2.x assembler are more restricted:

a. Label and line numbers are not used. Labels are used to reference other lines and locations in a program. The one-line assembler has no knowledge of other program lines and, therefore, cannot make the required association between a label and the label definition located on a separate line.

b. Source lines are not saved. In order to read back a program after it has been entered, the machine code is disassembled and then displayed as mnemonic and operands.

c. Limited error indication. The one-line assembler will show a question mark (?) under the portion of the source statement where an error probably occurred, or will display the word "ERROR" or another short message. In contrast, the resident assembler generates specific error messages for over 60 different types of errors.

d. Only one directive (DC.W) is accepted.

e. No macro operation capability is included.

f. No conditional assembly is used.

g. Several symbols recognized by the resident assembler are not included in the VME/10 assembler character set. These symbols include !, >, and <. Two other symbols, * and /, each have multiple meanings to the resident assembler, depending on the context, but only one meaning to the VME/10 assembler. Finally, the ampersand character (&) specifies a decimal number when used with the TENbug 2.x assembler (although numbers with no prefix are assumed to be decimal), while this symbol represents a logical AND function to the resident assembler. Paragraph 5.2.1.5 describes the VME/10 assembler character set.

Although functional differences exist between the two assemblers, the one-line assembler is a true subset of the resident assembler. The format and syntax used with the TENbug 2.x assembler are acceptable to the resident assembler except as described in g. above.

5.2 SOURCE PROGRAM CODING

A source program is a sequence of source statements arranged in a logical way to perform a predetermined task. Each source statement occupies a line and must be either an executable instruction or a DC.W assembler directive. Each source statement follows a consistent source line format.

5.2.1 Source Line Format

Each source statement is a combination of operation and, as required, operand fields; line numbers, labels, and comments are not used. The general format is:

<sp> <operation field> [<operand field>]

The space (<sp>) must be the first character of each line. This is to be consistent with the resident assembler, which expects the first field of each line to be either a space or a label. Because the TENbug 2.x assembler never allows a label, the first character must always be a space.

5.2.1.1 Operation Field. The operation field must follow at least one space (more can be used) and entries can consist of one of two categories:

a. Operation codes which correspond to the MC68010 instruction set.

b. Define constant directive (DC.W) which is recognized to define a constant in a word location. This is the only directive recognized by the assembler.

The size of the data field affected by an instruction is determined by the data size code. Some instructions and directives can operate on more than one data size. For these operations, the data size code must be specified or a default size applicable to that instruction will be assumed. The size code need not be specified if only one data size is permitted by the operation. The data size code is specified by a period (.), appended to the operation field, and followed by B, W, or L, where:

B = Byte (8-bit data).
W = Word (the usual default size; 16-bit data).
L = Longword (32-bit data).

The data size code is not permitted, however, when the instruction or directive does not have a data size attribute.

Examples (legal):

LEA	2(A0),A1	Longword size is assumed (.B,.W not allowed); this instruction loads effective address of first operand into A1.
ADD.B	(A0),D0	This instruction adds the byte whose address is (A0) to lowest order byte in D0.
ADD	D1,D2	This instruction adds low order word of D1 to low order word of D2. (W is the default size code.)
ADD.L	A3,D3	This instruction adds entire 32-bit (longword) contents of A3 to D3.

Example (illegal):

SUBA.B	#5,A1	Illegal size specification (.B not allowed on SUBA). This instruction would have subtracted the value 5 from the low order byte of A1; byte operations on address registers are not allowed.

5.2.1.2 Operand Field. If present, the operand field follows the operation field and is separated from the operation field by at least one space. When two or more operand subfields appear within a statement, they must be separated by a comma. In an instruction like ' ADD D1,D2' the first subfield (D1) is generally applied to the second subfield (D2) and the results placed in the second subfield. Thus, the contents of D1 are added to the contents of D2 and the result is saved in register D2. In the instruction ' MOVE D1,D2' the first subfield (D1) is the sending field and the second subfield (D2) is the receiving field. In other words, for most two-operand instructions, the general format '<opcode> <source>,<destination>' applies.

5.2.1.3 Disassembled Source Line. The disassembled source line may not look identical to the source line entered. The disassembler makes a decision on how to represent a numerical value based on how it interprets the number's use. If the number is determined to be an address or a "would-be" address, it is displayed in hexadecimal; everything else is decimal. For example,

```
MOVE.L   #$1234, $5678
```

disassembles to

```
005000     21FC000012345678     MOVE.L   #4660,$00005678
```

Also, for some instructions, there are two valid mnemonics for the same opcode, or there is more than one assembly language equivalent. The disassembler may choose a form different from the one originally entered. As examples:

a. BRA is returned for BT
b. DBF is returned for DBRA

NOTE

The assembler recognizes two forms of mnemonics for two branch instructions. The BT form (branch conditionally true) has the same opcode as the BRA instruction. Also, DBRA (decrement and branch always) and DBF (never true, decrement, and branch) mnemonics are different forms for the same instruction. In each case, the assembler will accept both forms.

5.2.1.4 Mnemonics and Delimiters. The assembler recognizes all MC68000 instruction mnemonics except ILLEGAL. Numbers are recognized as both decimal and hexadecimal, with decimal the default case (note that this is reverse to the TENbug 2.x commands):

a. Decimal is a string of decimal digits (0-9) without a prefix (default) or preceded by an optional ampersand (&). Examples are:

```
1234
&1234
```

b. Hexadecimal is a string of hexadecimal digits (0-9, A-F) preceded by a dollar sign ($). An example is:

```
$AFE5
```

One or more ASCII characters enclosed by apostrophes (') constitute an ASCII string. ASCII strings are left-justified and zero-filled (if necessary), whether stored or used as immediate operands. This left justification will be to a word boundary if one or two characters are specified, or to a longword boundary if the string contains more than two characters.

```
005000    5300                  DC.W     'S'
005002    223C41424344          MOVE.L   #'ABCD',D1
005008    3536                  DC.W     '56'
```

NOTE

The MC68000 has seventeen 32-bit registers (D0-D7, A0-A6, SSP, USP) in addition to a 32-bit program counter (24 bits available) and a 16-bit status register. Registers D0-D7 are used as data registers for byte, word, and longword operations. Registers A0-A6 and SSP and USP are used as software stack pointers and base address registers; they may also be used for word and longword data operations. All 17 registers may be used as index registers. Register A7 is a pseudo register, used as the system stack pointer corresponding to either SSP or USP, depending on the operating state.

The following register mnemonics are recognized by the assembler:

D0-D7	Data registers.
A0-A7	Address registers.
SSP	Address register 7 represents the supervisor stack pointer of the active system state.
USP	User stack pointer. Used only in privileged instructions which are restricted to supervisory state.
CCR	Condition code register (low 8 bits of SR).
SR	Status register. All 16 bits may be modified in the supervisor state. Only low 8 bits (CCR) may be modified in user state.
PC	Program counter. Used only in forcing program counter-relative addressing.
VBR	Vector base register, contains the 32-bit absolute address of the beginning of the exception vector.
SFC	Source function code.
DFC	Destination function code.

5.2.1.5 Character Set. The character set recognized by the TENbug 2.x assembler is a subset of ASCII, and these are listed below:

a. The uppercase letters A through Z

b. The integers 0 through 9

c. Arithmetic operators: + -

d. Parentheses ( )

e. Characters used as special prefixes:

(pound sign) specifies the immediate form of addressing
$ (dollar sign) specifies a hexadecimal number
& (ampersand) specifies a decimal number
@ (commercial at sign) specifies an octal number
% (percent sign) specifies a binary number
' (apostrophe) specifies an ASCII literal character

f. Five separating characters:

Space
, (comma)
. (period)
/ (slash)
- (dash)

g. The character * (asterisk) indicates current location.

5.2.2 Instruction Summary

Refer to the M68000 16/32-Bit Microprocessor Programmer's Reference Manual for descriptions of the MC68000 instructions and addressing modes.

5.3 ENTERING AND MODIFYING SOURCE PROGRAMS

User programs are entered into the VME/10 RAM using the one-line assembler/ disassembler. The program is entered in assembly language statements on a line-by-line basis. The source code is not saved as it is converted immediately to machine code upon entry. This imposes several restrictions on the type of source line that can be entered.

Symbols and labels, other than the defined instruction mnemonics, are not allowed. The assembler has no means to store the associated values of the symbols and labels in lookup tables. This forces the programmer to use memory addresses and to enter data directly rather than use labels.

Also, editing is accomplished by retyping the entire new source line. Lines can be added or deleted by moving a block of memory data to free up or delete the appropriate number of locations.

In order to describe more clearly the procedures used to enter, modify, and execute a program, a specific example will be described. Figure 5-1 lists a program that converts an ASCII coded number into its hexadecimal equivalent. An ASCII character is in the lowest 8 bits of register D0 when the program is entered. Upon exiting, D0 contains the equivalent hexadecimal digit (0 to F), or an FF if the ASCII character does not correspond to a proper hex number.

```
GETHEX      CMP.B     #$30,D0        IS HEX NO. < 0?
            BLT.S     ERROR          NOT A HEX NO.
            CMP.B     #$39,D0        IS HEX NO. > 9?
            BGT.S     GTHX2
GTHX1       AND.L     #$F,D0         SAVE ONLY LOWER 4 BITS
EXIT        BRA       *              END OF ROUTINE
GTHX2       CMP.B     #$41,D0        IS HEX NO. < 'A'?
            BLT.S     ERROR          NOT A HEX NO.
            CMP.B     #$46,D0        IS HEX NO. > 'F'?
            BGT.S     ERROR          NOT A HEX NO.
            SUB.B     #7,D0          MAKE IT SMALLER -- A=10
            BRA       GTHX1
ERROR       MOVE.L    #$FF,D0        ERROR CODE
            JMP       EXIT
```

NOTE: Converts ASCII digit in lowest 8-bit of register D0 into hex value. Returns equivalent 0-F or FF on error in D0.

FIGURE 5-1. Sample Program to Convert ASCII Digit to Hexadecimal Value

For clarity, Figure 5-1 contains comments and labels. The program as it appears after entry into the VME/10 is shown in Figure 5-3. Figure 5-2 shows the ASCII character set for better understanding of the program.

b7 b6 b5 / Bits						0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	b3	b2	b1	Row	Column Hex	0	1	2	3	4	5	6	7
0	0	0	0	0	0	NUL	DLE	SP	0	@	P		p
0	0	0	1	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	C	FF	FS	,	<	L	\	l	|
1	1	0	1	13	D	CR	GS	-	=	M	]	m	}
1	1	1	0	14	E	SO	RS	.	>	N	^	n	~
1	1	1	1	15	F	SI	US	/	?	O	_	o	DEL

FIGURE 5-2. ASCII Character Set

5.3.1 Invoking the Assembler/Disassembler

The assembler/disassembler is invoked using the ;DI option of the Memory Modify (MM) and Memory Display (MD) commands:

```
MM <address> ;DI
```

where (CR) sequences to next instruction
(.CR) exits command

and

```
MD[<port number>] <address> [<count>];DI
```

The Memory Modify (;DI option) is used for program entry and modification. When this command is used, the memory contents at the specified location are disassembled and displayed, followed by a "?". A new or modified line can be entered if desired.

The disassembled line can be an MC68000 instruction or a DC.W directive. If the disassembler recognizes a valid form of some instruction, the instruction will be returned; if not, the DC.W $xxxx (always hex) is returned. Because the disassembler gives precedence to instructions, a word of data that corresponds to a valid instruction will be returned as the instruction.

For the given example, the program will be entered starting at location $1000:

```
TENbug 2.x > MM 1000;DI
001000    1005                    MOVE.B  D5,D0 ?
```

5.3.2 Entering a Source Line

A new source line is entered immediately following the "?", using the format discussed in paragraph 5.2.1:

```
TENbug 2.x > MM 1000;DI
001000    1005                    MOVE.B  D5,D0 ?  CMP.B #$30,D0
```

When the carriage return is entered terminating the line, the old source line is erased from the terminal screen, the new line is assembled and displayed, and the next instruction in memory is disassembled and displayed:

```
TENbug 2.x > MM 1000;DI
001000    0C000030                CMP.B   #$30,D0
001004    FFFF                    DC.W    $FFFF ?
```

NOTE

If a terminal with a printer only (no CRT) is used, such as a TI 700 series device, the printer will overwrite the previous line. Therefore, a clear printout of the new entry will not be made. This also happens if the printer on port 3 is attached via the PA command.

Another program line can now be entered. Program entry continues in like manner until all lines have been entered. A period is used to exit the MM command.

If an error is encountered during assembly of the new line, the assembler will display the line unassembled with an "X" under the field suspected of causing a problem, or an error message will be displayed. Errors are discussed in paragraph 5.3.5.

5.3.3 Program Entry/Branch and Jump Addresses

Figure 5-3 shows the sample program as it is input to the VME/10 one-line assembler. Notice that the comments and labels used in Figure 5-1 are not allowed; absolute addresses must be used for BRA and JMP instructions.

```
CMP.B    #$30,D0
BLT      *
CMP.B    #$39,D0
BGT      *
AND.L    #$F,D0
BRA      *
CMP.B    #$41,D0
BLT      *
BGT      *
SUB.B    #7,D0
BRA      $100C
MOVE.L   #$FF,D0
JMP      $1012
```

a) First entry

```
CMP.B    #$30,D0
BLT      $1022
CMP.B    #$39,D0
BGT      $1014
AND.L    #$F,D0
BRA      *
CMP.B    #$41,D0
BLT      $1022
BGT      $1022
SUB.B    #7,D0
BRA      $100C
MOVE.L   #$FF,D0
JMP      $1012
```

b) With correct branch addresses

FIGURE 5-3. Sample Program as Entered into VME/10

5.3.3.1 Entering Absolute Addresses. The absolute addresses are probably not known as the program is being entered. For example, when the second line is entered (BLT.S ERROR in Figure 5-1), the user does not know that the branch address (ERROR MOVE.B #$FF,D0) will be $1022. However, the user can instead enter an "*" for branch to self. After the correct address ($1022) is discovered, the second line can be reentered using the correct value. This technique can be used for forward branches and jumps. It is not required for backward branches and jumps, such as the last line of the example, because the required address is already known. If the absolute address is not within the range of a short address, a long address must be specified by appending .L to the mnemonic (BGT.L *).

5.3.3.2 Desired Instruction Form. Care must be taken when entering source lines to ensure that the desired instruction form is entered. If the quick form of the instruction is wanted, it must be specified. For example:

```
005780   203C00000003   MOVE.L   #3,D0     Assembles to the 6-byte instruction.
```

whereas

```
005780   7003           MOVEQ.L  #3,D0     Assembles to the 2-byte instruction.
```

If the PC-relative addressing mode is desired, it must be specified. For example:

```
001000   41F803F0    LEA   $3F0,A0       Assembles $3F0 as an absolute address.
```

whereas

```
001000   41FAF3EE    LEA   $3F0(PC),A0   Assembles $3F0 as a PC-relative address.
```

5.3.3.3 Current Location. To reference a current location in an operand expression, the character "*" (asterisk) can be used. Examples are:

```
007000      6022            BRA      *+$24

007000      6000FFFE        BRA.L    *

007000      60FE            BRA      *
```

5.3.4 Assembler Output/Program Listings

A listing of the program is obtained using the Memory Display (MD) command with the ;DI option. The MD command requires both the starting address and the instruction count to be entered in the command line.

Two techniques can be used to obtain a hard copy of the program using the MD command.

a. The Printer Attach (PA) command is first used to activate the port 4 or 5 printer. A Memory Display (MD) to the terminal will then cause a listing on the terminal and on the printer.

b. An MD4 or MD5 (Memory Display to port 4 or 5) command using the ;DI option will cause a listing on the printer only.

Figure 5-4 shows a listing of the sample program. Note in this example that $D lines are specified in the MD command.

Note also that the listing does not correspond exactly to that of Figure 5-3. As discussed in paragraph 5.2.1.3, the disassembler displays in hexadecimal any number it interprets as an address; all other numbers are displayed in decimal.

```
TENbug 2.x > MD 1000 D;DI
001000    0C000030              CMP.B     #48,D0
001004    6D1C                  BLT.S     $001022
001006    0C000039              CMP.B     #57,D0
00100A    6E08                  BGT.S     $001014
00100C    02800000000F          AND.L     #15,D0
001012    60FE                  BRA.S     $001012
001014    0C000041              CMP.B     #65,D0
001018    6D08                  BLT.S     $001022
00101A    6E06                  BGT.S     $001022
00101C    04000007              SUB.B     #7,D0
001020    60EA                  BRA.S     $00100C
001022    203C000000FF          MOVE.L    #255,D0
001028    4EF81012              JMP       $1012

TENbug 2.x >
```

FIGURE 5-4. Sample Program Listing

5.3.5 Error Conditions and Messages

There are five different conditions that can result in error messages while using the assembler/disassembler. The response to the error condition can be to abort the command (and thus the assembler), or to cause the assembler to ask for a corrected input line. The error conditions are discussed in the following paragraphs and include bus and address error traps, improper characters, numbers which are too large, and assembly errors.

5.3.5.1 Error Traps. Two types of errors are trapped. One form, which produces a bus error trap, may be encountered if a location is accessed where there is no memory. Included in this error type are write cycles to ROM. The second form produces an address error trap. Instructions must always begin on an even address; if not, an address error trap will result. Figure 5-5 shows examples of these conditions.

```
TENbug 2.x > M A00000;DI

2700 0000768A 8008 1105 00A00000 0000 7682 0000 FFFF 0000 6100 0557
1158 0000 00A0 0008 0000 00A0 0002 FFE1 0006 038F 4CD4 0100 0000 4CD4 0000

BUS ERROR TRAP
PC=0000768A SR=2700=.S7..... USP=FFFFFFFF SSP=0000149A VBR=00000000 SFC=2 DFC=7
D0-7 00A00044 01964D20 FFF24D20 00000008 0000B432 00000000 00000000 00000000
A0-7 000016EA 00004EDA 000014D1 0000115C 00A00000 00001158 00001158 0000149A
 PC=00768A    E51C                ROL.B   #2,D4

TENbug 2.x > MM F00000;DI

2700 000076E8 8008 0205 00F00000 0000 8C67 0000 1419 0000 1419 051A
8C67 1280 0000 14DC 14D0 00F0 0001 FFE1 0000 0380 1419 0000 0001 1280 0001

BUS ERROR TRAP
PC=000076E8 SR=2700=.S7..... USP=FFFFFFFF SSP=0000149A VBR=00000000 SFC=2 DFC=7
D0-7 671E8C67 00000001 00000253 00000000 0000001E 0000001E 00000002 00000000
A0-7 00001517 00F00000 000014D1 00001537 00F00000 0000153A 0000153A 0000149A
 PC=0076E8    B400                CMP.B   D0,D2

TENbug 2.x > M 10001;DI

2700 0000768A 800C 1105 00010001 0000 7682 0000 FFFF 0000 6100 0557
1158 0000 0001 0008 0001 0001 0003 FFE1 0006 038F 4CD4 0100 0000 4CD4 0000

BUS ERROR TRAP
PC=0000768A SR=2700=.S7..... USP=FFFFFFFF SSP=0000149A VBR=00000000 SFC=2 DFC=7
D0-7 00010044 01964D20 FFF24D20 00000008 0000B432 00000000 00000000 00000000
A0-7 000016EA 00004EDA 000014D1 0000115C 00010001 00001158 00001158 0000149A
 PC=00768A    E51C                ROL.B   #2,D4
```

FIGURE 5-5. Examples of Error Traps

Also note that bus and address errors also cause display of the exception status from the stack, in hexadecimal characters.

For details on this display, refer to the bus error and address error descriptions in the M68000 16/32-Bit Microprocessor Programmer's Reference Manual.

5.3.5.2 Improper Character. If a character appears in the operand field that does not belong to the class of characters specified or expected, an "X" will be printed beneath the character string suspected of containing the improper character, followed by a "?" to prompt reentry of the line. For example, if a % (percent sign) is used to specify the binary class of characters, only the digits 0 and 1 will be accepted.

```
TENbug 2.x > MM 6000;DI                                  S is not a decimal digit

006000    FFFF      DC.W     $FFFF ?   MOVE.W  #S',D0
006000              MOVE.W   #S',D0
                               X?

TENbug 2.x > MM 6000;DI                                  9 is not an octal digit

006000    FFFF      DC.W     $FFFF ?   ADDA.L  #@974,A6
006000              ADDA.L   #@974,A6
                               X?

TENbug 2.x > MM 6000;DI                                  P is not a decimal digit

006000    FFFF      DC.W     $FFFF ?   JMP  $4000+PC
006000              JMP      $4000+PC
                                     X?
```

FIGURE 5-6. Examples of Improper Characters

5.3.5.3 Number Too Large. Another error type involves numbers which are too large for the MC68000 to handle. Again, an "X" is printed under the number suspected of containing the error, followed by a "?". Figure 5-7 gives an example.

```
TENbug 2.x > MM 4000;DI                              Value is larger than 32 bits

004000    FFFF      DC.W     $FFFF ? LEA.L  $937402110,A7
004000              LEA.L    $937402110,A7
                               X?
```

FIGURE 5-7. Example of a Number Which Is Too Large

5.3.5.4 Assembly Errors. An assembly error can occur due to an invalid opcode, an illegal addressing mode for a particular instruction, a format which is in error (leading space omitted as an example), or a source line which is incorrect in some other way. When the entry as written is not a valid MC68000 instruction, the assembler echoes the source line up to and including the field in which the error probably occurred. It also prints an "X" under the field suspected of containing an error, followed by a "?" to prompt reentry of the line.

The entire line must be reentered in its correct form. If the error has not been corrected or another is encountered, the error indicator will be returned. After all errors have been corrected and the source line represents a valid MC68000 instruction, the line will be assembled. The memory address, machine code, and source code will be displayed and the next line will be disassembled. A period (.) is used to exit the command. Examples of typical errors are shown in Figure 5-8.

```
Example 1    Invalid Opcode

    006700   FFFF       DC.W      $FFFF ?   BEQU.S    $6754
                        BEQU.S
                        X?  BEQ.S $6754
    006700   6752       BEQ.S     $6754

Example 2    Missing Leading Space

    001100   FFFF       DC.W      $FFFF     ?OR.B     D5,(A6)

                        X? OR.B  D5,(A6)
    001100   8B16       OR.B    D5,(A6)

Example 3    Unrecognizable Opcode

    005300   FFFF       DC.W      $FFFF ?   MULSW     52,D3
                        MULSW
                        X?  MULS.W     52,D3
    005300   C7F80034    MULS.W    52,D3

Example 4    Invalid Size Extension

    007200   FFFF       DC.W      $FFFF ?   MOVEQ.B   #2,D1
                        MOVEQ.B   #2,D1
                                       X?  MOVEQ.L   #2,D1
    007200   7202       MOVEQ.L   #2,D1

Example 5    Invalid Addressing Mode

    001500   FFFF       DC.W      $FFFF  ?       ADDQ.B    #7,A0
                        ADDQ.B    #7,A0
                                       X?  ADDQ.B    #7,(A0)
    001500   5E10       ADDQ.B    #7,(A0)

Examples 6 and 7   Branch Address Too Large

    004900   FFFF       DC.W      $FFFF  ?       BRA  $10000
                        BRA       $10000
                                        X?  BRA    $8000
    004900   600036FE   BRA       $8000


    004800   FFFF       DC.W      $FFFF  ?       BRA.S     $7000
                        BRA.S     $7000
                                       X?  BRA.S     $4902
                        BRA.S     $4902
                                       X?  BRA.S     $4860
    004800   605E       BRA.S     $4860
```

FIGURE 5-8. Examples of Assembly Errors

CHAPTER 6

TENbug ROUTINES AVAILABLE TO THE USER

6.1 INTRODUCTION

This chapter describes the TENbug TRAP #15 I/O handler, which allows system calls from user programs. The system calls can be used to access selected functional routines contained within the TENbug firmware, including input and output routines. TRAP #15 may also be used to transfer control to TENbug without performing initialization.

6.2 USER I/O THROUGH TRAP #15

Format in user program:

```
TRAP #15       Call to TENbug trap handler
DC.W $000x     Function being requested (x = function)
```

Valid Functions (refer to paragraph 4.2.26 for port number definitions):

FUNCTION	DESTINATION	DESCRIPTION
0	TENbug	Display format (see DF); then go to TENbug.
1	Console (port 1)	Input line Input parameters: Point A5.L and A6.L both to the start of buffer. Exit conditions: A5.L points to the start of buffer; A6.L points to the end + 1.
2	Console (port 1)	Output line (with CR, LF) Input parameters: Point A5.L to start of string and A6.L to end of string + 1. Exit conditions: None.
3	Host (port 2)	Read line (no echo) Input parameters: Same as Function 1. Exit conditions: Same as Function 1.
4	Host (port 2)	Output line (with CR, LF) Input parameters: Same as Function 2.
5	Printer (port 4)	Output line (with CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
6	Console (port 1)	Output line (no CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.

7	Host (port 2)	Output line (no CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
8	Printer (port 4)	Print line (no CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
9	Host (port 3)	Read line (no echo) Input parameters: Same as Function 1. Exit conditions: Same as Function 1.
A	Host (port 3)	Output line (with CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
B	Host (port 3)	Output line (no CF, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
C	Printer (port 5)	Print line (with CR, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.
D	Printer (port 5)	Print line (no CF, LF) Input parameters: Same as Function 2. Exit conditions: Same as Function 2.

EXAMPLE PROGRAM:

```
                    *
                    *     TEST OF TRAP #15 USER I/O
                    *
      00002000              ORG $2000           PROGRAM STARTS HERE
002000 2E7C00004000 START   MOVE.L #$4000,A7    INITIALIZE STACK
002006 2A7C0000201C         MOVE.L #BUFFER,A5   FIX UP A5 & A6 for I/O
00200C 2C4D                 MOVE.L A5,A6
                    *
00200E 4E4F                 TRAP #15            INPUT BUFFER FROM CONSOLE
002010 0001                 DC.W 1
                    *
002012 4E4F                 TRAP #15            PRINT BUFFER TO CONSOLE
002014 0002                 DC.W 2
                    *
002016 4E4F                 TRAP #15            STOP HERE LIKE BREAKPOINT
002018 0000                 DC.W 0
00201A 60E4
                    *
00201C 0200         BUFFER  DS.L 128            THIS IS THE I/O BUFFER
```

6.3 TENbug SUBROUTINES

A branch table is located at the beginning of ROM to allow use of TENbug subroutines under various operating systems. Note, however, that TRAP #15 support is the standard interface between the user and I/O routines. Use of the branch table should be based upon a complete understanding of the TENbug 2.x source release.

The branch table allows a calling routine to access any of the supported subroutines with a BSR to a long branch located at the start of EPROM. The user will always be able to access these subrouties despite future changes in their locations because the table offset will not change (even though the actual subroutine addresses might).

In most cases, the following subroutines can be entered with a BSR to the address shown for each entry.

$F0000C TRACE - Trace one instruction

No registers are required for linkage except the stack for the RTS.

$F00010 BOCMD - Disk boot entry point

No registers are required for linkage; control is returned to TENbug.

$F00014 ABORTB - Software abort routine

No registers are required for linkage; control is returned to TENbug.

$F00018 CHKBP - Illegal instruction vector

No registers are required for linkage; control is returned to TENbug.

$F0001C CHECKSUM - Calculate or verify checksum

A0	Start of memory
A1	End of memory
CCR	Condition code returned:
Z	(BEQ) Indicates valid checksum

$F00020 DISKR - Disk read routine

A0	Address of hardware (controller/drive)
D0	Number of 256-byte blocks
D1	Winchester byte 5
D2	Memory address
D3	Block address
IPCNUM	RAM variable containing 1-byte controller number
DRVNUM	RAM variable containing 1-byte drive number

$F00024 DISKW - Disk write routine

A0	Address of hardware (controller/drive)
D0	Number of 256-byte blocks
D1	Winchester byte 5
D2	Memory address
D3	Block address
IPCNUM	RAM variable containing 1-byte controller number
DRVNUM	RAM variable containing 1-byte drive number

$F00028 INITVECT - Initialize miscellaneous vectors (#4-#11 and #24-#48)

$F0002C INITHRAM - Initialize specific vectors (bus and address error)

$F00030 INCHS - Input a character from the keyboard

D0	Converted value for key pressed
CCR	Condition codes returned:
Z	(BEQ) Character is displayable
X	(BMI) BREAK entered
C	(BCS) Character is not displayable (i.e., $0-$1F, $80-$FE)

$F00034 OUTCHS - Output a character to the display

D0 must contain the character to be output.

$F00038 POINTRAM - Get RAM pointer

A0 will contain the address of the following RAM variables:

DS.L 1	Starting address of RAM
DS.L 1	Ending address of RAM
DS.L 1	Address of the disk configuration table
DS.L 1	Address of the disk controller/drive variable

$F0003C N/A - Reserved for future use

$F00040 HOTSTART - User request to restart TENbug

No registers are required for linkage; control is returned to TENbug.

$F00044 N/A - Reserved for future use

$F00048 N/A - Reserved for future use

APPENDIX A

SOFTWARE ABORT

If a target program must be stopped with the stack data preserved, the user may press the ABORT pushbutton on the VME/10 chassis operator panel. This will generate a level seven interrupt vector which will interrupt the target program and load the contents of ROM location $138 into the program counter. If the default vector locations have not been overwritten, the console will display SOFTWARE ABORT and the following data will be saved: address registers, data registers, program counter, status register, supervisor stack pointer, user stack pointer, vector base register, destination function code, and source function code.

Remember that TENbug shares resources with the target program under test (refer to paragraph 1.3.2). Therefore, if the target program changes the contents of location $138, this abort feature is lost.

In contrast to the abort feature, the contents of the target supervisor stack pointer, program counter, and status register are lost when the RESET pushbutton is pressed. The RESET feature sets the processor to supervisor state, loads the supervisor stack pointer with the contents of RAM locations 0-$3, and loads the program counter with the contents of RAM locations $4-$7. It also saves the contents of the target registers for display by the Display Format (DF) command.

MOTOROLA

M68KTENBG/A1

AUGUST 1984

ADDENDUM

TO

TENbug DEBUGGING PACKAGE

USER'S MANUAL

This addendum transmits replacement pages for the TENbug Debugging Package User's Manual. Replacement pages have been marked with vertical change bars to indicate revised or new material.

Insert the changed pages attached to this addendum into your M68KTENBG/D2 manual. Make certain that the page you are replacing is removed from your manual. This page of the addendum should be placed after the title page and used as a record page of the changes made to the manual.

Replacement pages provided by this addendum are:

1-1 through 1-6
4-13 through 4-16
4-37 through 4-40
4-43, 4-44, 4-69, 4-70

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

This manual describes the debugging monitor TENbug as it is used in the VME/10 Microcomputer System, hereafter referred to as the VME/10.

1.2 DEFINITION OF TENbug

TENbug is the resident firmware debugging package for the VME/10. The 32K-byte firmware (stored in ROM or EPROM devices) provides a self-contained programming and operating environment. TENbug interacts with the user through predefined commands that are entered via the terminal. The commands fall into five general categories:

a. Commands which allow the user to display or modify memory.

b. Commands which allow the user to display or modify the various internal registers of the MC68010.

c. Commands which allow the user to execute a program under various levels of control.

d. Commands which control access to the various input/output resources on the board.

e. Commands which allow the user to select and test video features and graphics resolution.

An additional function called the TRAP #15 I/O handler allows the user program to utilize various routines within TENbug. The TRAP #15 handler is discussed in Chapter 6.

The operational mode of TENbug is described in Chapter 2.

1.3 TENbug INTERNAL STRUCTURE

1.3.1 Memory Map

The following abbreviated memory map for the VME/10 highlights addresses that might be of particular interest to TENbug users. Refer to the VME/10 Microcomputer System Reference Manual for a complete description of the memory maps for both high- and low-resolution graphics modes.

Note that addresses are assumed to be hexadecimal throughout this manual. In text, numbers may be preceded with a dollar sign ($) for identification as hexadecimal.

RAM LOCATION	FUNCTION
0-3FF	Vectors
400-AFF	Work area and stack for TENbug

SPECIAL LOCATIONS	FUNCTION
F00000-F00007	Area containing initial values for supervisor stack pointer, program counter, and vector base register after cold start
F14000-F14FFF	Area used to define programmable "soft" character set

I/O LOCATION	FUNCTION
F1C1C9	Serial port 2 (host), serial I/O card (optional)
F1C1CB	Serial port 3 (host), serial I/O card (optional)
F1C1E1	Parallel port 1 (printer), parallel I/O card (optional)
F1C1E9	Parallel port 2 (printer), parallel I/O card (optional)
F1C0D1	Base address of RWIN1 Disk Controller

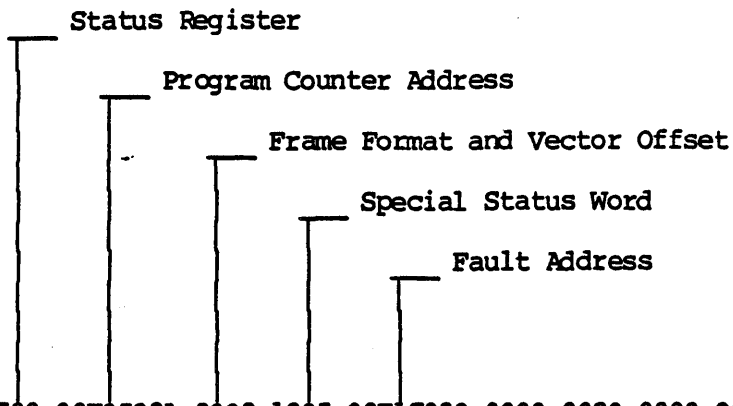
1.3.2 Vectors and Errors

TENbug shares resources with the target program under test -- that is, each affected resource can be used only by TENbug or the target program at any given time.

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle the exception. These vectors are initialized by TENbug in default memory locations 0 through $3FF during a cold- or warm-start sequence (see Chapter 2). If the target program uses any of these locations, the user values must be rewritten following each cold or warm start. If the target program uses any of the following locations, the associated function will be lost to TENbug.

MEMORY LOCATION	TENbug FUNCTION
10-13	Breakpoints (illegal instructions)
24-27	Trace
BC-BF	TRAP #15 user calls to TENbug
138-13B	ABORT pushbutton switch on VME/10 operator panel (refer to Appendix A)

When uninitialized vectors are given control because of exception processing, the console terminal will display a general message indicating the vector offset that was used: Offset Vector $xxx Error Trap. In addition, several of the vectors cause display of appropriate information. (Refer to Appendix B for a list of error messages.) BUS and ADDR error traps also cause display of the exception status from the stack, in hexadecimal characters, as shown in the following example.



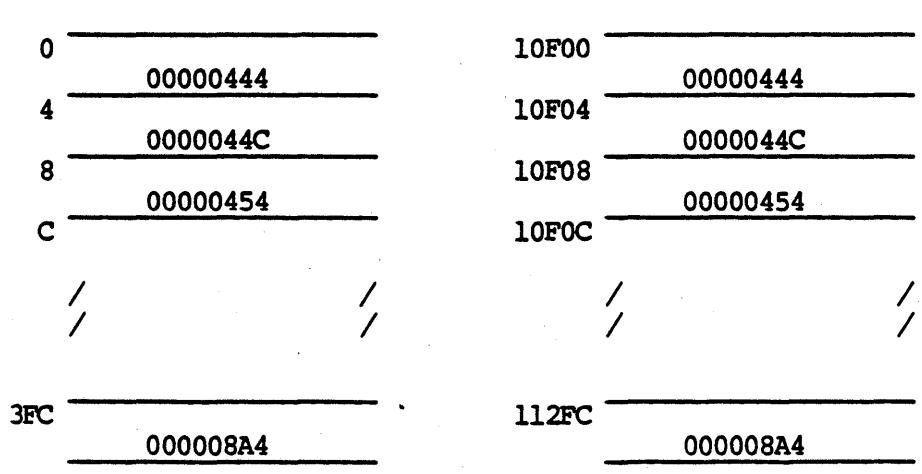

```
2709 00F0533A 8008 1305 00F1C030 0000 0020 0000 20FF 00F0 2007 067A
5338 0000 00F1 2007 C030 00F0 533C FFE1 0000 0394 1E14 0000 0004 1E14 0003

BUS ERROR TRAP
```

For additional information on this display, refer to the bus error, address error, and the reference classification descriptions in the exception processing chapter of the M68000 16/32-Bit Microprocessor Programmer's Reference Manual.

1.3.2.1 Resetting Vector Base Register. The MC68010 processor upon which the VME/10 is based features a Vector Base Register (VBR) which contains the base (starting) address for the VME/10 exception vectors. Exception vectors are located in memory addresses 0 through $3FF relative to the VBR. Upon reset (cold or warm start) of the MC68010, the value of the VBR is set to zero.

TENbug must have control of the exception vectors to function properly. If the user sets the VBR to a value other than its default value of zero, he must also establish a new set of exception vector memory locations for the VBR value. In other words, the user must copy all existing vector memory locations to the same relative location in the new VBR table.

In the following example, the VBR value is changed from 0 to 10F00. Exception vector memory locations must also be copied to this new location. Note that the content of each vector memory location (i.e., the appropriate routine address) remains the same.



1.3.3 Disk I/O

TENbug provides limited support of disk I/O through a Winchester Disk Controller. The commands supported are BH, BO, IOC, IOP, and IOT. BH, BO, and optionally, IOT read the volume ID found in sector 0 of each disk. Pointers to the configuration data located in sector 0 are used to read the configuration sector. The configuration sector contains the data that allows the RWIN1 controller to issue reads to the specific types of drives.

NOTE

A sector is 256 bytes. The disk controller maps physical sectors on various disks into virtual 256-byte sectors at the controller interface.

If a disk read to a Winchester drive fails, the read will be retried twice before generating an error message. The RWIN1 controller will make corrections, if possible, and the disk transfer will continue on from the next sector.

NOTE

ERROR CORRECTION and RETRY were first provided in TENbug 2.1.

The first 256 bytes of the media are the volume ID. Bytes $F8-$FF of the volume ID must contain either the ASCII character string "EXORMACS" or "MOTOROLA"; otherwise, an error message will result. For more information on interpreting the data displayed, see the Winchester Disk Controller User's Manual.

The other information used from the volume ID is:

BYTES	USED FOR
$14-$17	Starting sector address of program to be loaded (via BH, BO).
$18-$19	Number of 256-byte sectors to be loaded.
$1E-$21	Load address (first destination memory byte).
$90-$93	Sector address of media configuration parameters (refer to Appendix C).
$94	Length of configuration area (usually one 256-byte sector).

1.4 TENbug WITH SYSTEM V/68

The following paragraphs list information specific to the use of TENbug with SYSTEM V/68.

1.4.1 Operational Commands

In the following list, commands given in parentheses indicate the key that is to be pressed. Commands not given in parentheses are to be typed as shown.

BH	Boots the operating system from the fixed disk and halts.
BO	Boots the operating system from the fixed disk and gives control to the program loaded.
(BREAK)	Aborts command.
(DEL)	Deletes character.
(CTRL-D)	Redisplays line.
(CTRL-H)	Deletes character.
(CTRL-W)	Suspends output; any character continues output.
(CTRL-X)	Cancels command line.

1.4.2 Debugging Commands

The following commands may be useful for debugging, but should be used only in single-user mode after sync has executed. Use of these commands may result in the need for system reboot.

.A0-.A7	BARS, NOBARS *	HE
.D0-.D7	BF	IOC
.DFC	BM	IOP
.PC	BR, NOBR	IOT
.R0-.R6	BS	MD
.SFC	CH, NOCH	MM
.SR	CRT	MS
.SSP	CS	OF
.USP	DC	PA, NOPA
.VBR	DF	TR
	GD	TT
	GO	
	GR, NOGR	
	GT	

* This command modifies graphics memory and should be used only with an operating system configured to support graphics.

1.4.3 Non-Applicable Commands

The following commands should be used in a stand-alone mode; they should not be used with SYSTEM V/68.

BI	PF
BT	TM
DU	VE
LO	

1.5 REFERENCE MANUALS

Refer to the following documents for more information on the environments in which TENbug is used.

VME/10 Microcomputer System Overview Manual, M68KVSOM

VME/10 Microcomputer System Diagnostics Manual, M68KVSDM

VME/10 Microcomputer System Reference Manual, M68KVSREF

VERSAdos to VME Hardware and Software Configuration User's Manual, MVMEDOS

Winchester Disk Controller User's Manual, M68RWIN1

MVME400 Dual RS-232C Serial Port Module User's Manual, MVME400

MVME410 Dual 16-Bit Parallel Port Module User's Manual, MVME410

M68000 16/32-Bit Microprocessor Programmer's Reference Manual, M68000UM

EXAMPLE

```
TENbug 2.x > .R4 4000

TENbug 2.x > BR 1010 2000;5 2040 4000

BREAKPOINTS
001010     001010
002000     002000;5
002040     002040
000000+R4  004000

TENbug 2.x > NOBR 1010 2040

BREAKPOINTS
002000     002000;5
000000+R4  004000

TENbug 2.x > NOBR

BREAKPOINTS

TENbug 2.x >
```

4.2.9 Block of Memory Search (BS) BS

```
BS <address1> <address2> '<literal string>'
BS <address1> <address2) <data> [<mask>] [1<options>]
```

The BS command has two modes: literal string search and hex data search. Both modes can search memory beginning at <address1> through <address2>, looking for a match. Alternatively, a user can specify that a data search report back only locations that do not match the input data. This alternative search for a mismatch can be particularly useful when searching for suspected faulty memory. For example, a known pattern can be placed into suspect RAM locations, and a BS command with an option to search for a mismatch will display any bad RAM locations.

The literal string mode is initiated if a single quote (') follows <address2>. The ASCII literal string may contain both uppercase and lowercase letters. If a single quote does not follow <address2>, data search mode is assumed. If the optional mask is supplied with a data search, the mask is ANDed to the data found at each address. The data located in the memory is not changed. The masked data is then examined for a match. (The default mask is all 1's.)

Available options for a data search enable a user to specify the data format and whether to search for a match or a mismatch. The options to specify data format are the letters B, W, and L. To specify a mismatch, a minus sign is placed before or after the data format indicator. If there is no minus sign in the options field, a matching search is assumed.

```
;B      Data format is a byte; search for a match.
;-B     Data format is a byte; search for a mismatch.
;W      Data format is a word; search for a match.
;-W     Data format is a word; search for a mismatch.
;L      Data format is a longword; search for a match.
;-L     Data format is a longword; search for a mismatch.
```

The default value for a data search is ;B.

When a search is completed, each address containing data that meets the specified requirements is displayed on the terminal, along with the data located at that address.

To illustrate the searching command, the following examples are provided:

EXAMPLE	COMMENT
TENbug 2.x > MD 10000 40	Show memory to be searched.
010000 A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A %Z%Z%Z%Z%Z%Z%Z%Z	
010010 41 61 20 41 42 61 62 20 41 42 43 61 62 63 20 20 Aa ABab ABCabc	
010020 A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A %Z%Z%Z%Z%Z%Z%Z%Z	
010030 A5 5A A5 5A A5 5A A5 52 A5 52 A5 52 A5 5A A5 5A %Z%Z%Z%R%R%R%Z%Z	
TENbug 2.x > BS 10000 10040 'ab'	Successful search for literal string
Physical Address=00010000 00010040	'ab'.
010015 'ab'	
01001B 'ab'	

```
TENbug 2.x  > BS 10000 10040 43 DF;B
Physical Address=00010000 00010040
01001A 43
01001D 64
```

Successful data search using a mask allowing both lowercase and uppercase ASCII C.

```
TENbug 2.x  > BS 10020 10040 A55A;-W
Physical Address=00010020 00010040
010036 A552
010038 A552
01003A A552
```

Search for any words NOT matching the test pattern.

4.2.10 Block of Memory Test (BT) BT

BT <address1> <address2>

The BT command provides a destructive test of a block of memory. A word boundary (even address) must be given for the starting <address1> and ending <address2> of the block. If the test runs to completion without detecting an error, all memory tested will have been set to zeros.

Execution of this command may take several seconds for large blocks of memory.

When a problem is found in a memory location, the address, the data stored, and the data read are displayed. Control is then returned to TENbug.

See also: BI

EXAMPLE	COMMENT
TENbug 2.x > BT 44000 47FFE PHYSICAL ADDRESS=00044000 00047FFE	Successful memory test; no errors found.
TENbug 2.x > BT 44000 4FFFE PHYSICAL ADDRESS=00044000 0004FFFE FAILED AT 0480FE WROTE=FFFF READ=0000	Unsuccessful memory test; error data is listed.
TENbug 2.x >	

EXAMPLE	COMMENT
TENbug 2.x > IOP	Request physical disk I/O to read a routine.
READ OR WRITE (R/W)=.....R ? (CR)	Use default of read.
MEMORY ADDRESS FOR DISK I/O=.$00000000 ? 1000	Change $0 default to $1000.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? ^	That isn't correct; back up one parameter.
MEMORY ADDRESS FOR DISK I/O=.$00001000 ? A00	Change $1000 to $A00.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? 2	Change default drive 0 to drive 2.
CONTROLLER NUMBER=.....$00 ? (CR)	Use controller 0.
FIRST BLOCK NUMBER=.$00000000 ? 500	Change block number to $500.
NUMBER OF (256 BYTE) BLOCKS=.....$0000 ? 1	Read one 256-byte block.
ARE YOU SURE? (Y/N) ? Y	Last change? Yes, all is ready.
"R" COMPLETE	The read is complete.
TENbug 2.x > MD A00 30	Display data read from disk (only first $30 bytes).

```
00A00    41 F8 10 00 20 3C 00 00  02 FF 11 00 51 C8 FF FC  Ax.. <......QH.:
000A10   60 EE 4E 71 4E 71 4E 71  4E 71 4E 71 4E 71 4E 71  'nNqNqNqNqNqNqNq
000A20   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

TENbug 2.x > M A12;L	Make change to RAM where routine was loaded.
000A12 4E714E71 ? -1	
000A16 4E714E71 ? -1	
000A1A 4E714E71 ? -1	
000A1E 4E710000 ? -1	
000A22 00000000 ? .	
TENbug 2.x > IOP	Request physical disk I/O to write to a disk.
READ OR WRITE (R/W)=.....R ? W	Change to W for write.
MEMORY ADDRESS FOR DISK I/O=.$00000A00 ? (CR)	Use previous address.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$02 ? (CR)	Drive is fine; no change.
CONTROLLER NUMBER=.....$00 ? (CR)	No change.
FIRST BLOCK NUMBER=.$00000500 ? (CR)	No change.
NUMBER OF (256 BYTE) BLOCKS=.....$0001 ? (CR)	No change.
ARE YOU SURE (Y/N) ? Y	Last chance; are you sure? Yes, all is ready.
"W" COMPLETE	Write is complete.
TENbug 2.x >	

IOT [<device>][<controller>]

where:

device — Is a single hexadecimal digit, 0 through 3, specifying the disk to be read. Default value is 0.

controller — Is a single hexadecimal digit, 0 or 1, specifying the RWIN1 controller through which the disk is connected. Default value is 0.

The IOT command allows the user to change the configuration of the RWIN1 controller. If the IOT command is invoked without specifying <device> or <controller>, the command will prompt for required information. If the <device> and/or <controller> are specified in the IOT command line, the current configuration is overwritten with the configuration data located on the disk, without the user having to know and manually enter the parameter information required. The parameters required for correct configuration when the options are not specified depend upon the type of drive, as shown below:

WINCHESTER HARD DISK	5.25-INCH FLOPPY DISK
Drive number	Drive number
Controller number	Controller number
Sector size	Sector size
Number of heads	Number of heads
Number of cylinders	Number of cylinders
Number of sectors per track	Number of sectors per track
	Motorola/IBM format
	Single- or double-sided media
	Single- or double-track density
	Single- or double-data density

The IOT command will present the appropriate questions based upon which drive has been specified. There are four actions that can be taken following a question mark prompt:

? (CR) - Entering a carriage return indicates that the existing value for the current parameter is acceptable; go on to the next parameter.

? . - Entering a period indicates that this execution of the IOC command must be terminated now, without asking for more parameters.

? ^ - Entering a caret symbol indicates that a previous parameter requires a change and will logically back up one parameter each time it is entered (until the first entry is reached, where it will remain until one of the other responses is received).

? <data> - Entering the appropriate data requested (followed by a carriage return or ENTER). Often the parameters are checked for valid options (i.e, Y or N).

Appropriate configuration information for specific disk types is listed in the "Mass Storage" chapter of the VERSAdos to VME Hardware and Software Configuration User's Manual.

EXAMPLE	COMMENT
TENbug 2.x > IOT	Teach RWIN1 controller a new configuration.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$02 ? 00	Select drive 0.
CONTROLLER NUMBER=.....$00 ? (CR)	Use default controller 0.
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ? .	Change of heart ... start over again.
TENbug 2.x > IOT	Invoke the IOT command again.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ? 02	Configure drive 2 (first floppy).
CONTROLLER NUMBER=.....$00 ? (CR)	No change.
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ? (CR)	No change.
NUMBER OF HEADS, (ON THIS DRIVE)=.....$02 ? (CR)	No change.
NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0050 ? (CR)	No change.
SECTORS PER TRACK=.....$10 ? 8	Change from 16 sectors/track to 8.
MOTOROLA/IBM FORMAT (M/I)=.....I ? (CR)	No change.
SINGLE/DOUBLE SIDED MEDIA (S/D)=.....D ? (CR)	No change.
SINGLE/DOUBLE TRACK DENSITY (S/D)=.....D ? (CR)	No change.
SINGLE/DOUBLE DATA DENSITY (S/D)=.....D ? S	Change to single-data density.
TENbug 2.x > IOT	IOT command showing default parameters for hard disk #0. TENbug initializes the RWIN1. No parameters were entered from the menu.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ?	
CONTROLLER NUMBER=.....$00 ?	
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ?	
NUMBER OF HEADS, (ON THIS DRIVE)=.....$01 ?	
NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0001 ?	
SECTORS PER TRACK=.....$00 ?	
TENbug 2.x > IOT 0	IOT command requesting the configuration data from the disk and initializing the RWIN1. No parameters were entered from the menu.
DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.....$00 ?	
CONTROLLER NUMBER=.....$00 ?	
SECTOR SIZE (0=128,1=256,2-512,3-1024)=.....1 ?	
NUMBER OF HEADS, (ON THIS DRIVE)=.....$06 ?	
NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0132 ?	
SECTORS PER TRACK=.....$20 ?	

```
TENbug 2.x > IOT

  DRIVE NUMBER (0&1=FIXED,2&3=FLOPPY)=.......$02 ? 0
                    CONTROLLER NUMBER=.......$00 ? (CR)
SECTOR SIZE(0=128,1=256,2=512,3=1024)=.........1 ? (CR)
     NUMBER OF HEADS, (ON THIS DRIVE)=.......$01 ? (CR)
 NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0001 ? ^
     NUMBER OF HEADS, (ON THIS DRIVE)=.......$01 ? 2
 NUMBER OF CYLINDERS, (ON THIS DRIVE)=.....$0001 ? 132
                   SECTORS PER TRACK=.......$00 ? 20

TENbug 2.x >
```

Invoke IOT to change configuration of hard disk to allow use as a 5-megabyte hard disk.

Oops, passed the number of heads parameter; back up.
Change to 2 heads.
Change to $132 cylinders.
And $20 (32) sectors/track.

NOTE: These parameters are specifically for the 5-megabyte Winchester hard disk. To find out what a particular Winchester hard disk requires for configuration, boot the operating system and, while "inactive" (not updating critical files), press the ABORT button. Then enter an IOT command to see how the boot device is configured. If all is well, enter GO and continue with operating system control.

```
TENbug 2.x > LO ;X=DU A00 A80
```

Enter Load command specifying ;X option (echo S-records to CRT as memory is being loaded).

```
DU A00 A80
PHYSICAL ADDRESS=00000A00 00000A80
```

Notice S0, S1, and S9 records are displayed upon screen. (Refer to warning in command description about timing restrictions with the ;X option.)

```
S0030000FC
S1130A00000102030405FFFF08090A0B0C0D0E0F79
S1130A1010111213141516171819lA1B1C1D1E1F5A
S1130A2020212223242526272829 2A2B2C2D2E2F4A
S1130A3030313233343536373839 3A3B3C3D3E3F3A
S1130A4040414243444546474849 4A4B4C4D4E4F2A
S1130A5050515253545556575859 5A5B5C5D5E5F1A
S1130A6060616263646566676869 6A6B6C6D6E6F0A
S1130A7070717273747576777879 7A7B7C7D7E7FFA
S1040A8080F1
S90300000FC
```

```
TENbug 2.x > MD A00 80
```

Display memory containing downloaded data.

```
000A00   00 01 02 03 04 05 FF FF  08 09 0A 0B 0C 0D 0E 0F  ................
000A10   10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  ................
000A20   20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F   !"#S%&'()*+,-./
000A30   30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
000A40   40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
000A50   50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
000A60   60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  'abcdefghijklmno
000A70   70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}".
```

NOTE

The host system used to create and transmit the S-records was an MC68000 Educational Computer Board (MEX68KECB).

MD[<port number>] <address> [<count>][;<options>]

The MD command displays a portion of memory which begins at <address> and extends for the number of bytes or lines given as <count>. There are two formats that can be requested with the MD command.

a. The dump format begins each line with the starting or next hexadecimal memory address followed by 16 hex bytes per line with the ASCII equivalent shown to the right. The number of lines varies with the <count> entered (or default). There are no partial lines. If the byte count ends in the middle of a line, the complete line is displayed. (Default byte <count> is $10.)

b. The disassembler format provides:

1. The starting or next hexadecimal memory address.

2. The object code displayed in hexadecimal.

3. The M68010 source statement that will assemble into the object code as described in 2. above.

If the operation code is not valid, a "Define Constant" is constructed for one word. Notice that <count> for the disassembler mode is a number of source lines to be disassembled and displayed, not the number of bytes. (Default line <count> is $10.)

Default destination is the console terminal. Specifying MD<port number> allows the output to be directed to another port.

Valid port numbers for this command are:

PORT NUMBER	DESCRIPTION
none	Defaults to TENbug port 1 (VME/10 built-in terminal/keyboard).
1	Specifies TENbug port 1 (VME/10 built-in terminal/keyboard).
2	Specifies TENbug port 2 (MVME400 port 2 - 7201/B).
3	Specifies TENbug port 3 (MVME400 port 1 - 7201/A).
4	Specifies TENbug port 4 (MVME410 port 1 - PIA/A).
5	Specifies TENbug port 5 (MVME410 port 2 - PIA/B).

Options supported are the disassembler and the screen option.

;DI Requests the disassembler option. The <count>, if provided, is a line count (default is $10).

;S Requests the display of a full screen of memory (16 lines of display in either dump or disassembler format). Notice that the default for disassembly is $10 (or 16 decimal) anyway. If the <count> and ;S option are both entered within the same MD command, the ;S option has priority.

All combinations are valid (e.g., ;DIS, ;SDI, ;S DI, ;DI S).

4.3 COMMAND SUMMARY

The commands and options available to TENbug 2.x users are summarized in Table 4-2.

TABLE 4-2. TENbug Command and Option Summary

COMMAND	DESCRIPTION
[NO]BARS	Draw graphics test pattern.
BF <address1> <address2> <pattern>	Block fill.
BH [<device>][,<controller>]	Boot and halt.
BI <address1> <address2>	Block initialize.
BM <address1> <address2> <address3>	Block move.
BO [<device>][,<controller>][,<string>]	Boot operating system.
[NO]BR [<address>[;<count>]]...	Set and remove breakpoints.
BS <address1> <address2> '<literal string>' <address1> <address2> <data>[<mask>][;<option>]	Block search; options ;B ;W ;L ;-B ;-W ;-L.
BT <address1> <address2>	Block test.
[NO]CH [<bits>]	Alter character display map.
CRT	Alter CRT control registers.
CS [<address1>] [<address2>]	Checksum.
DC <expression>	Data conversion/evaluation.
DF	Display formatted registers.
DU[<port number>] <address1> <address2> [<text>]	Dump memory (S-records).
GD [<address>]	Go direct.

TABLE 4-2. TENbug Commands and Option Summary (cont'd)

COMMAND	DESCRIPTION
G[O] [<address>]	Install breakpoints and go.
[NO]GR [<bits>]	Alter graphics display map.
GT <temporary breakpoint address>	Go until address.
HE	Display commands/registers.
IOC	Issue RWIN1 command.
IOP	Issue physical read/write.
IOT [<device>][<controller>]	Teach RWIN1 a configuration.
LO[<port number>] [;<options>] =<text>	Load (S-records).
MD[<port number>] <address> [<count>][;<options>]	Memory display; options ;DI ;S.
M[M] <address>[;<options>]	Memory modify; options ;W ;L ;O ;V ;N ;DI.
MS <address> <data>	Memory set (also ASCII).
OF	Offset register display.
[NO]PA[<port number>]	Printer attach/detach.
PF[<port number>]	Port format.
TM [<exit character> [<trailing character>]]	Transparent mode.
T[R] [<count>]	Trace.
TT <breakpoint address>	Trace until address.
VE[<port number>] [;<options>] =<text>	Verify (S-records).
VM	Toggle video map.

APPENDIX B

TENbug MESSAGES

ERROR MESSAGE	MEANING
PRINTER NOT READY	Printer is not properly connected or cannot receive output.
Error	Error (prefix).
ILLEGAL INSTRUCTION	Instruction used an illegal opcode.
.... Error Trap	See Traps in M68000 16/32-Bit Microprocessor Programmer's Reference Manual.
Offset Vector $xxx Error Trap	Indicates uninitialized vector.
Invalid Option... valid options are... ;DI or ;S	Memory Display command response to invalid option.
IS NOT A HEX DIGIT	Improper character entered in a field that requires a hexadecimal digit.
DATA DID NOT STORE	Data did not go where intended (such as attempting to write to ROM).
Invalid Address	Address too big (1 in bits 24 - 31) or odd for .W or .L (1 in bit 0).
What?	Program does not recognize user's entry.
NOT HEX	Same as IS NOT A HEX DIGIT.
DISK ERROR: Status=xx xx xx xx xx xx xx xx xx xx DISK ERROR: Status=Busy	Refer to the Winchester Disk Controller User's Manual for explanation of the ten status bytes.

OTHER MESSAGE	MEANING
TENbug 2.x >	TENbug 2.x prompt.
Software Abort	Displayed when ABORT button is used.
Break	BREAK key has been used.
At Breakpoint	Indicates program has stopped at breakpoint.
Physical Address	Actual address calculated using parameters and relative offsets.
.PC within "DEBUGGER"	Displayed by trace commands indicating care must be taken while "TESTING" within TENbug (e.g., with breakpoints and STACK).
Booting from: xxxx	Indicates the volume ID of the disk being booted. Message is suppressed if volume ID is null.
Booting from ROM: xxxx	Indicates the name of the routine in ROM that is receiving control during the ROMBOOT procedure.

APPENDIX C

CONFIGURATION AREA

Disks initialized using some systems contain a Volume Identification Block and a Disk Configuration Block in sectors 0 and 1, respectively. TENbug looks for either "EXORMACS" or "MOTOROLA" in locations $F8-$FF of sector 0 to validate the disk. Refer to paragraph 1.3.3 for more information used from the volume ID. TENbug then uses the following parameters from the Disk Configuration Block to access the disk:

Attributes word

Physical sectors per track on media

Number of heads on drive

Number of cylinders on media

Physical sector size of media

Precompensation cylinder number

The complete Disk Configuration Block is shown below:

256-BYTE SECTOR 1 OFFSET	LENGTH IN BYTES	PARAMETER DESCRIPTION
		DEVICE STATUS OR
0	1	CONFIGURATION ERROR CODE
1	1	CHANNEL TYPE
2	1	DEVICE TYPE
3	1	DRIVER CODE
4	2	ATTRIBUTES MASK
6	2	PARAMETERS MASK
8	2	ATTRIBUTES WORD
10($A)	2	SECTOR SIZE
12($C)	4	TOTAL SECTORS
16($10)	4	WRITE TIMEOUT (UNUSED)
20($14)	4	READ TIMEOUT (UNUSED)
24($18)	1	PHYSICAL SECTORS PER TRACK ON MEDIA
25($19)	1	NO. OF HEADS ON DRIVE
26($1A)	2	NO. OF CYLINDERS ON MEDIA
28($1C)	1	INTERLEAVE FACTOR ON MEDIA
29($1D)	1	SPIRAL OFFSET ON MEDIA
30($1E)	2	PHYSICAL SECTOR SIZE OF MEDIA
32($20)	2	PHYSICAL SECTOR SIZE OF DRIVE
34($22)	2	NUMBER OF CYLINDERS ON DRIVE
36($24)	2	PRECOMPENSATION CYLINDER # (usually .5 total cyl)
38($26)	1	PHYSICAL SECTORS PER TRACK ON DRIVE
39($27)	7	RESERVED
40($28)	60($D8)	UNUSED

Disks initialized on some systems may not contain the Volume Identification Block or the Disk Configuration Block. These disks cannot be accessed by TENbug until the locations $F8-$FF of sector 0 are modified to contain either "EXORMACS" or "MOTOROLA". TENbug then uses the following default values to access the disk. These default values will allow access of track 0 for all configurations.

	RWIN1	
PARAMETER DESCRIPTION	FLOPPY DISK	HARD DISK
Attributes word	$0F	$10
Physical sectors per track on media	10	N/A
Number of heads on drive	2	1
Number of cylinders on media	50	1
Physical sector size of media	N/A	N/A
Precompensation cylinder number	N/A	0

The attributes word is defined as:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	MT	SN	DS	MF	TD

MT - Media Type
0 = Floppy disk
1 = Hard disk

SN* - Sector Numbering
0 = Motorola format
1 = IBM format

DS* - Diskette Sides
0 = Single sided
1 = Double sided

MF* - Recording Method
0 = Single data density (FM)
1 = Double data density (MFM)

TD* - Track Density
0 = Single track density (48 TPI)
1 = Double track density (96 TPI)

* Floppy disk attribute only.

APPENDIX D

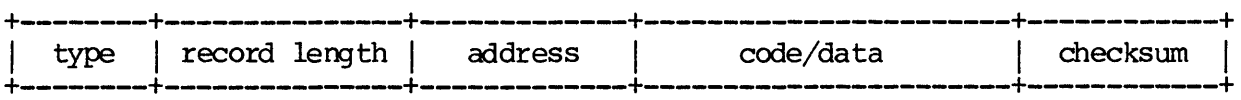
S-RECORD OUTPUT FORMAT

The S-record format for output modules was devised for the purpose of encoding programs or data files in a printable format for transportation between computer systems. The transportation process can thus be visually monitored and the S-records can be more easily edited.

S-RECORD CONTENT

When viewed by the user, S-records are essentially character strings made of several fields which identify the record type, record length, memory address, code/data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first character representing the high-order four bits, and the second the low-order four bits of the byte.

The five fields which comprise an S-record are shown below:



where the fields are composed as follows:

FIELD	PRINTABLE CHARACTERS	CONTENTS
type	2	S-record type -- S0, S1, etc.
record length	2	The count of the character pairs in the record, excluding the type and record length.
address	4, 6, or 8	The 2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
code/data	0-2n	From 0 to n bytes of executable code, memory-loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

When downloading S-records to TENbug, each record must be terminated with a CR. Additionally, an S-record may have an initial field to accommodate other data such as line numbers generated by some time-sharing systems.

Accuracy of transmission is ensured by the record length (byte count) and checksum fields.

S-RECORD TYPES

Eight types of S-records have been defined to accommodate the several needs of the encoding, transportation, and decoding functions. The various Motorola upload, download, and other record transportation control programs, as well as cross assemblers, linkers, and other file-creating or debugging programs, utilize only those S-records which serve the purpose of the program. For specific information on which S-records are supported by a particular program, the user's manual for that program must be consulted.

An S-record-format module may contain S-records of the following types:

S0 The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. Under VERSAdos, the resident linker's IDENT command can be used to designate module name, version number, revision number, and description information which will make up the header record. The address field is normally zeros.

S1 A record containing code/data and the 2-byte address at which the code/data is to reside.

S2 A record containing code/data and the 3-byte address at which the code/data is to reside.

S3 A record containing code/data and the 4-byte address at which the code/data is to reside.

S5 A record containing the number of S1, S2, and S3 records transmitted in a particular block. This count appears in the address field. There is no code/data field.

S7 A termination record for a block of S3 records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.

S8 A termination record for a block of S2 records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.

S9 A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. Under VERSAdos, the resident linker's ENTRY command can be used to specify this address. If not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

Only one termination record is used for each block of S-records. S7 and S8 records are usually used only when control is to be passed to a 3- or 4-byte address. Otherwise, an S9 record is used for termination. Normally, only one header record is used, although it is possible for multiple header records to occur.

CREATION OF S-RECORDS

S-record-format programs may be produced by several dump utilities, debuggers, the VERSAdos resident linkage editor, or several cross assemblers or cross linkers. On VERSAdos systems, the Build Load Module (MBLM) utility allows an executable load module to be built from S-records, and has a counterpart utility in BUILDS, which allows an S-record file to be created from a load module.

Programs are available for downloading or uploading a file in S-record format from a host system to an 8-bit microprocessor-based or a 16-bit microprocessor-based system.

EXAMPLE

Shown below is a typical S-record-format module, as printed or displayed:

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

The module consists of one S0 record, four S1 records, and an S9 record.

The S0 record is comprised of the following character pairs:

S0 — S-record type S0, indicating that it is a header record.

06 — Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.

00
00 — A 4-character, 2-byte address field; zeros in this example.

48
44 — ASCII H, D, and R - "HDR".
52

1B — The checksum.

The first S1 record is explained as follows:

S1 — S-record type S1, indicating that it is a code/data record to be loaded/verified at a 2-byte address.

13 — Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.

00
00 — A 4-character, 2-byte address field; hexadecimal address 0000; where the data which follows is to be loaded.

The next 16 character pairs of the first S1 record are the ASCII bytes of the actual program code/data. In this assembly language example, the hexadecimal opcodes of the program are written in sequence in the code/data fields of the S1 records:

OPCODE	INSTRUCTION	
285F	MOVE.L	(A7)+,A4
245F	MOVE.L	(A7)+,A2
2212	MOVE.L	(A2),D1
226A0004	MOVE.L	4(A2),A1
24290008	MOVE.L	FUNCTION(A1),D2
237C	MOVE.L	#FORCEFUNC,FUNCTION(A1)

. (The balance of this code is continued in the
. code/data fields of the remaining S1 records,
. and stored in memory location 0010, etc.)

2A The checksum of the first S1 record.

The second and third S1 records each also contain $13 (19) character pairs and are ended with checksums 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

The S9 record is explained as follows:

S9 S-record type S9, indicating that it is a termination record.

03 Hexadecimal 03, indicating that three character pairs (3 bytes) follow.

00
00 The address field, zeros.

FC The checksum of the S9 record.

Each printable character in an S-record is encoded in hexadecimal (ASCII in this example) representation of the binary bits which are actually transmitted. For example, the first S1 record above is sent as:

type				length				address								code/data										checksum			
S		1		1		3		0		0		0		0		2		8		5		F		...		2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	...		3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	...		0011	0010	0100	0001

SUGGESTION/PROBLEM REPORT

MICROSYSTEMS

QUALITY • PEOPLE • PERFORMANCE

Motorola welcomes your comments on its products and publications. Please use this form.

To: Motorola Inc.
Microsystems
2900 S. Diablo Way
Tempe, Arizona 85282
Attention: Publications Manager
Maildrop DW164

Product: _____ Manual: _____

COMMENTS: _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please Print

Name _____ Title _____

Company _____ Division _____

Street _____ Mail Drop _____ Phone _____

City _____ State _____ Zip _____

For Additional Motorola Publications
Literature Distribution Center
616 West 24th Street
Tempe, AZ 85282
(602) 994-6561

Four Phase/Motorola Customer Support, Tempe Operations
(800) 528-1908
(602) 438-3100

MOTOROLA

MOTOROLA ***Semiconductor Products Inc.***

P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

17622 PRINTED IN USA (6/84) MPS 4M