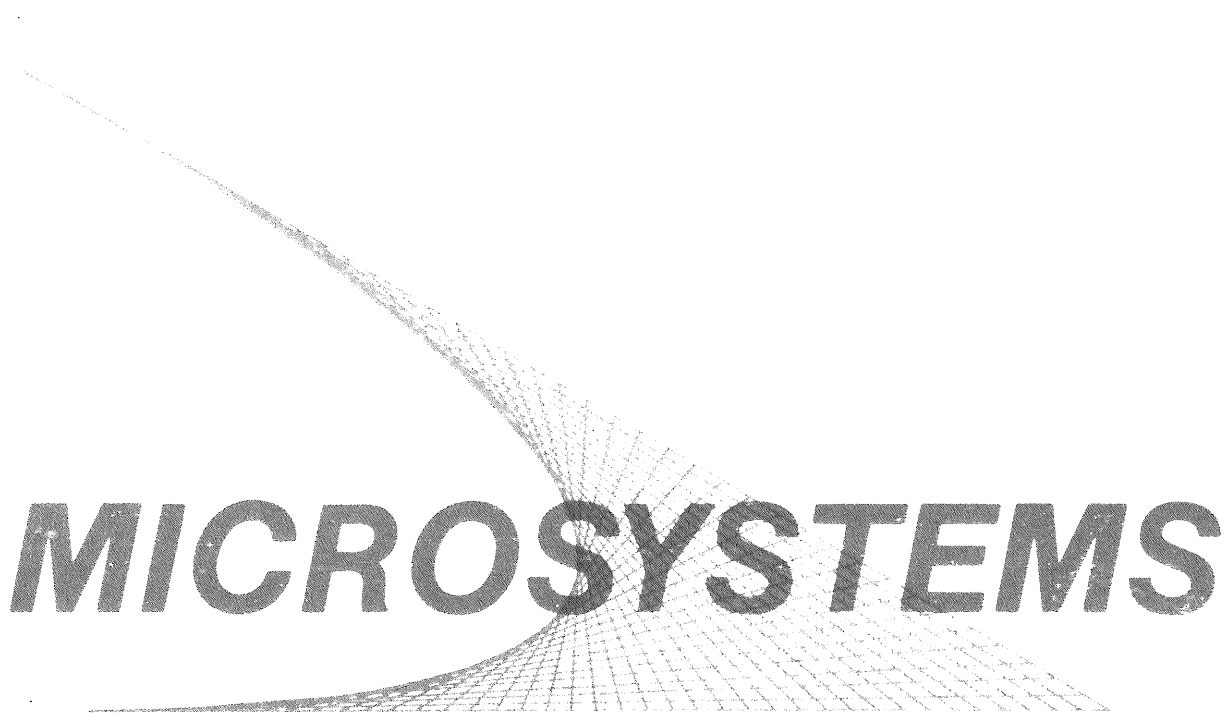
TENBUG USERS MANUAL

MOTOROLA

M68KTENBG/D1

TENbug Debugging Package
User's Manual



QUALITY • PEOPLE • PERFORMANCE

M68KTENBG/D1

SEPTEMBER 1983

TENbug DEBUGGING PACKAGE

USER'S MANUAL

First Edition

TABLE OF CONTENTS

		Page
CHAPTER 1	GENERAL INFORMATION	
1.1	INTRODUCTION	1-1
1.2	DEFINITION OF TENbug	1-1
1.3	TENbug INTERNAL STRUCTURE	1-1
1.3.1	Memory Map	1-1
1.3.2	Vectors and Errors	1-2
1.3.2.1	Resetting Vector Base Register	1-3
1.3.3	Disk I/O	1-4
1.4	REFERENCE MANUALS	1-4
CHAPTER 2	TENbug OPERATING PROCEDURE	
2.1	INTRODUCTION	2-1
2.2	CHASSIS CONTROL SWITCHES	2-1
2.3	TERMINAL CONTROL CHARACTERS	2-2
2.4	ENTERING TENbug DURING SYSTEM POWER-UP (COLD START)	2-2
2.4.1	Cold Start without MVME400 Module	2-2
2.4.2	Cold Start with MVME400 Module	2-4
2.5	ENTERING TENbug VIA SIMULATED COLD START	2-4
2.6	ENTERING TENbug WITHOUT DESTROYING MEMORY CONTENTS (WARM START)	2-5
2.7	TENbug COMMAND OPERATION	2-5
2.8	WHEN TENbug PROMPT FAILS TO APPEAR	2-5
CHAPTER 3	COMMAND LINE FORMAT	
3.1	INTRODUCTION	3-1
3.2	TENbug COMMAND LINE FORMAT	3-2
3.2.1	Expression as a Parameter	3-2
3.2.2	Address as a Parameter	3-2
3.2.2.1	Address Formats	3-3
3.2.2.2	Offset Registers	3-3
3.3	COMMAND VERIFICATION	3-4
CHAPTER 4	COMMAND SET	
4.1	INTRODUCTION	4-1
4.2	TENbug COMMANDS	4-2
4.2.1	Display/Set Register	4-3
4.2.2	Bootstrap Halt (BH)	4-4
4.2.3	Bootstrap Operating System (BO)	4-5
4.2.4	Breakpoint Set and Remove (BR and NOBR)	4-8
4.2.5	Display Formatted Registers (DF)	4-10
4.2.6	Execute Program (G)	4-11
4.2.7	Go Direct Execute Program (GD)	4-12
4.2.8	Go Until Breakpoint (GT)	4-13
4.2.9	Memory Modify (M)	4-14
4.2.10	Memory Display (MD)	4-15

TABLE OF CONTENTS (cont'd)

		Page
4.2.11	Display Offsets (OF)	4-17
4.2.12	Printer Attach and Detach (PA and NOPA)	4-18
4.2.13	Trace (T)	4-19
4.2.14	Trace to Temporary Breakpoint (TT)	4-21
4.2.15	Video Map (VM)	4-22
4.3	COMMAND SUMMARY	4-23
CHAPTER 5	TENbug ROUTINES AVAILABLE TO THE USER	
5.1	INTRODUCTION	5-1
5.2	USER I/O THROUGH TRAP #15	5-1
APPENDIX A	SOFTWARE ABORT	A-1
APPENDIX B	ERROR MESSAGES AND OTHER MESSAGES	B-1
APPENDIX C	CONFIGURATION AREA	C-1

LIST OF ILLUSTRATIONS

FIGURE 2-1.	Flow Diagram of VME/10 Cold Start	2-3
2-2.	Flow Diagram of TENbug Operational Mode	2-6

LIST OF TABLES

TABLE 4-1.	TENbug Commands by Type	4-1
4-2.	TENbug Command and Option Summary	4-23

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 16K-byte firmware (stored in two 8Kx8 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 video 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 5.

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 Overview Manual for a complete description of the memory maps for both high- and normal-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

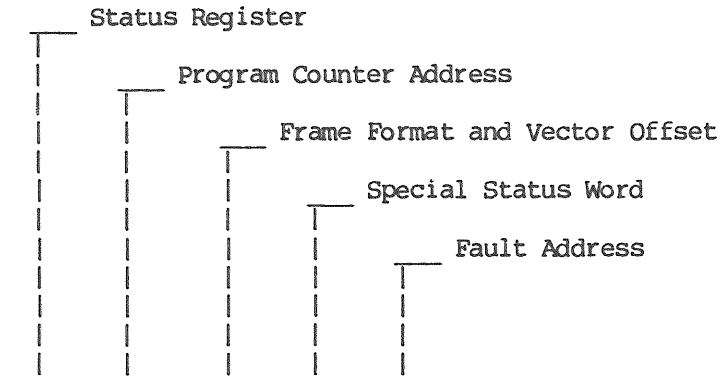
1.3.2 Vectors and Errors

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 (see Appendix A)

The vectors with default memory locations of $80 through $3FF cause a ???? TRAP ERROR 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 trap errors also cause display of the exception status from the stack, in hexadecimal characters, as shown in the following example.



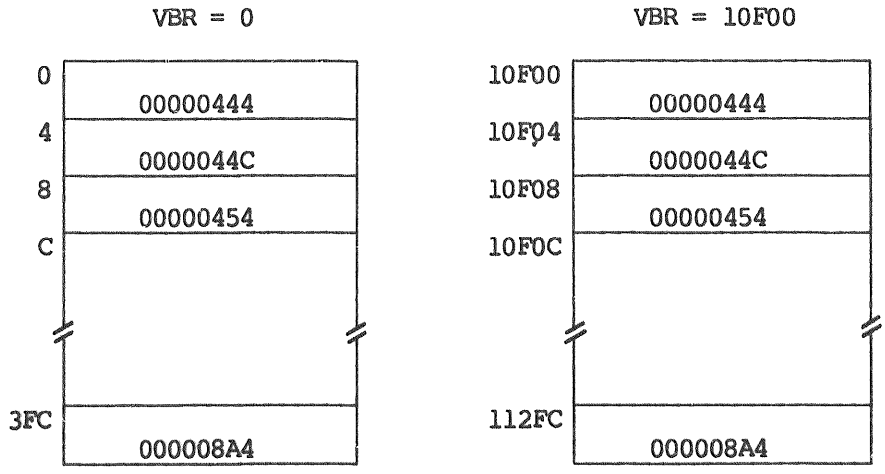

```
2709 00F0533A 8008 1305 00F1C030 0000 0020 0000 20FF 00F0 2007 067A
5338 0000 00F1 2007 C030 00F0 533C FFE1 0000 0394 1E14 0000 0004 1E14 0003

BUS TRAP ERROR
```

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 MC68010 16-Bit Virtual Memory Microprocessor product specification handbook.

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 and BO. Each of these commands does a read of the volume ID found on VERSAdos sector 0.

The first 256K bytes of the media are the volume ID. Bytes $F8-$FF of the volume ID must contain the ASCII character string "EXORMACS"; otherwise, an error message will result and the terminal will print the contents of the MC68010 registers. The D4, D5, and D6 registers contain status bytes from the disk controller. For more information on interpreting the contents of these registers, see the Winchester Disk Controller User's Manual.

The other information used from the volume ID is:

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

1.4 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 Hardware Manual, M68KVSHM

Winchester Disk Controller User's Manual, M68KWIN1

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

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

MC68010 16-Bit Virtual Memory Microprocessor product specification handbook, ADI-942

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 2.4 through 2.6.

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 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 position, VME/10 enters TENbug. Also, when the key switch is in the locked position, the front panel pushbutton switches RESET and ABORT 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 2.6 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 '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 2.5.

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 other 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 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.

2.4.1 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 x.y >
```

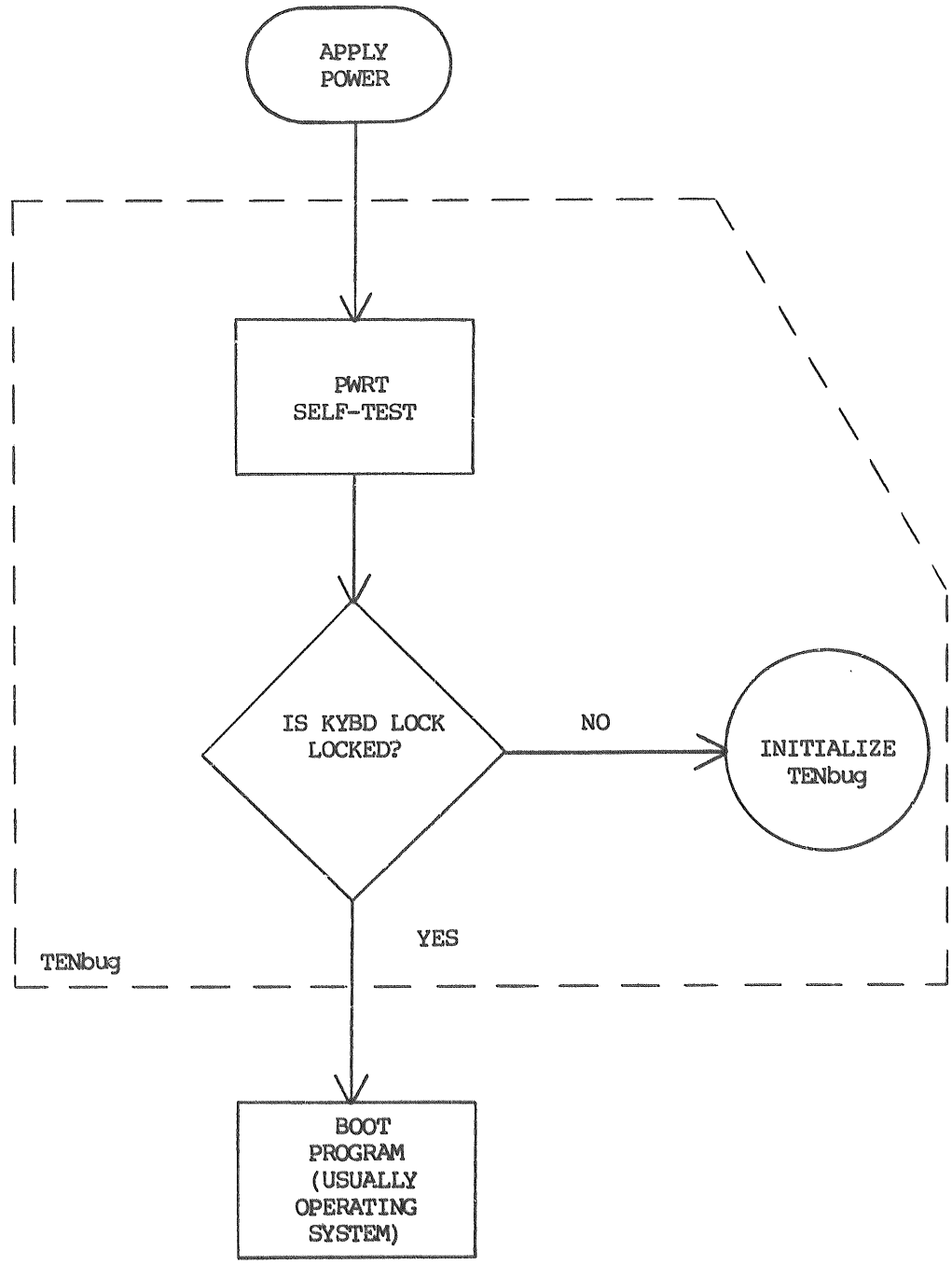


FIGURE 2-1. Flow Diagram of VME/10 Cold Start

2.4.2 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 x.y >
```

2.5 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 2.4.1.

e. When an MVME400 module is present in the VME card cage, go to step c. of paragraph 2.4.2.

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.

2.6 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-$3 into the supervisor stack, and the contents of $4-$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 2.4.1.

d. When an MVME400 module is present in the VME card cage, go to step c. of paragraph 2.4.2.

2.7 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.

2.8 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.

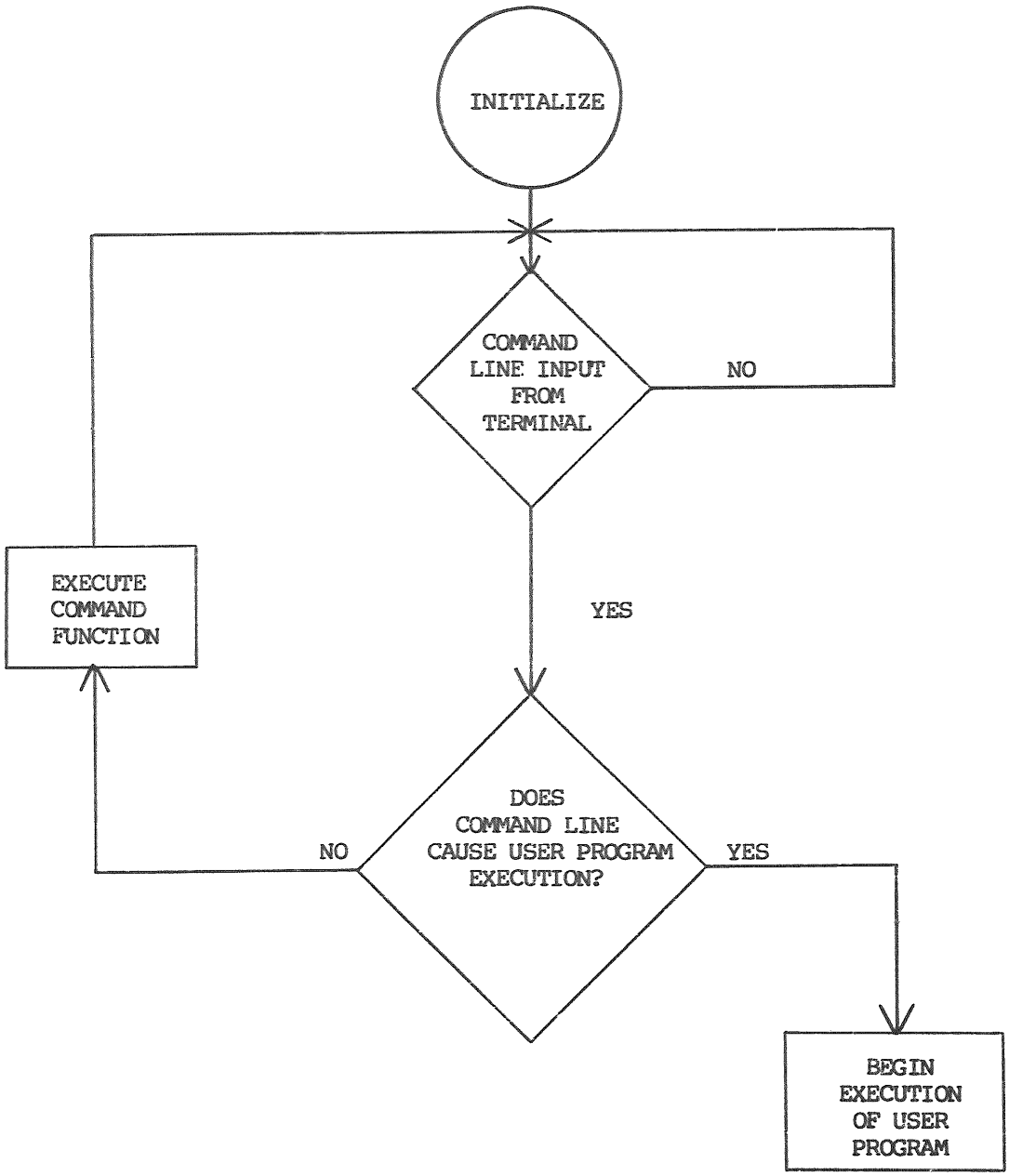


FIGURE 2-2. Flow Diagram of TENbug Operational Mode

COMMAND FORMAT

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 meta-symbols, which are used for definition only and are not entered by the user. These meta-symbols 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 x.y > [NO]<command>[<port number>] [<parameters>] [;<options>]

where:

TENBUG x.y >	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	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-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 (NOTE: offset zero is added).
expression+offset	130+R5	Absolute address plus offset five (not an assembler-accepted syntax).
expression+offset	150+R7	Absolute address (NOTE: offset 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 4.2.11, 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, then 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.

COMMAND SET

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.

TABLE 4-1. TENbug Commands by Type

COMMAND MNEMONIC	DESCRIPTON
MD	Memory display
MM	Memory modify
.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
.USP	Display/set user stack pointer
.VBR	Display/set vector base register
DF	Display formatted registers
.R0-.R6	Display/set relative offset register
OF	Display offsets
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
BH	Bootstrap halt
BO	Bootstrap operating system
VM	Video map

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.

.<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 x.y > .PC .PC=00001010	Display program counter.
TENbug x.y > .A7 1300	Set address register seven.
TENbug x.y > .R5 5500	Set relative offset register five.

```
TENbug x.y > DF
PC=00001010 SR=2700=.S7..... USP=0000C19E SSP=00001300 VBR=00000000 SFC=2 DFC=2
D0-7 00000000 00000000 00003048 4D453455 00000000 00000020 00000000 0007FFFE
A0-7 00F1C0D1 00001694 0000065C 00F01D72 00F0120C 00000538 00000538 00001300
  PC=001010
```

See also: DF, OF

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 - Boot Halt from default drive 0, default controller 0.

```
TENbug x.y > BH
PC=00001694 SR=2700=.S7..... USP=0000C19E SSP=00040E00 VBR=00000000 SFC=2 DFC=2
D0-7 00000000 00000000 00003048 4D453455 00000000 00000020 00000000 0007FFFE
A0-7 00F1C0D1 00001694 0000065C 00F01D72 00F0120C 00000538 00000538 00040E00
  PC=001694

TENbug x.y >
```

NOTE

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

```
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. Volume ID locations $F8-$FF are read to ensure that they contain the string "EXORMACS".

c. The location of the program to be loaded and its destination in memory are identified by examining volume ID locations as shown:

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

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

LOCATIONS	CONTENTS
$90-$93	VERSAdos sector address of the media configuration parameters (normally VERSAdos sector 1)
$94	Length of the configuration area (normally one VERSAdos 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. See Appendix C for additional information.

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

f. The status register is updated to reflect supervisor mode and interrupt level seven.

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

h. 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 in 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, M68KVSF.

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

EXAMPLES	COMMENTS
TENbug x.y > BO	Boot from the first hard disk (default device 0) on the RWIN1 (default controller 0).
TENbug x.y > BO 1	Boot from the second hard disk (device 1) on the RWIN1 (default controller 0).

4.2.4 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 (see 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 x.y > BR	Display all breakpoints.
TENbug x.y > BR <address>	Set a breakpoint.
TENbug x.y > 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 x.y > NOBR	Clear all breakpoints.
TENbug x.y > NOBR <address>	Clear a specific breakpoint.

See also: GT, TT

EXAMPLE

```
TENbug x.y > .R4 4000

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

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

TENbug x.y > NOBR 1010 2040

BREAKPOINTS
002000     002000;5
000000+R4  004000

TENbug x.y > NOBR

BREAKPOINTS

TENbug x.y >
```

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, etc., commands. See the display/set register command (.<register>), and the OF command.

EXAMPLE

```
TENbug x.y > DF
PC=00F02C9E SR=2704=.S7..Z.. USP=FFFFFFFF SSP=000007C4 VBR=00000000 SFC=2 DFC=2
D0-7 00300030 00000804 00000000 00000000 4D505520 00000020 00000000 00000000
A0-7 00F1A031 00F0133C 00F008AA 00000458 0000049A 00000536 00000536 000007C4
  PC=F02C9E

TENbug x.y > .R1 3000

TENbug x.y > .PC 40000

TENbug x.y > .SS C00

TENbug x.y > DF
PC=00040000 SR=2704=.S7..Z.. USP=FFFFFFFF SSP=00000C00 VBR=00000000 SFC=2 DFC=2
D0-7 00300030 00000804 00000000 00000000 4D505520 00000020 00000000 00000000
A0-7 00F1A031 00F0133C 00F008AA 00000458 0000049A 00000536 00000536 00000C00
  PC=03D000+R1

TENbug x.y >
```

G[O] [<address>]

The G (or GO) command causes the target program to execute (free run in real time) until:

a. The target program encounters a breakpoint,

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

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

NOTE

The execution will not be in real time if breakpoints with <count> are encountered.

The G sequence starts by tracing one instruction, setting any breakpoints, and then free running.

COMMAND FORMAT	DESCRIPTION
TENbug x.y > G	Begin execution at address in PC.
TENbug x.y > GO <address>	Set PC = address and begin execution at that address.

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

EXAMPLE

```
(Listing of program in memory at location 000900)
000900    1018                MOVE.B   (A0)+,D0
000902    0C000000            CMP.B    #0,D0
000906    67F8                BEQ.S    $000900
000908    4E75                RTS

TENbug x.y > BR 900 908

BREAKPOINTS
000900    000900
000908    000908

TENbug x.y > G 900
PHYSICAL ADDRESS=00000900

AT BREAKPOINT
PC=00000908 SR=2700=.S7..... USP=0000C19E SSP=00000C00 VBR=00000000 SFC=2 DFC=2
D0-7 00000020 00000000 00003048 4D453455 00000000 00000020 00000000 0007FFFE
A0-7 0000160D 00001694 0000065C 00F01D72 00F0120C 00000538 00000538 00000C00
  PC=000908

TENbug x.y >
```

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 000900)
000900    1018                  MOVE.B   (A0)+,D0
000902    0C000000              CMP.B    #0,D0
000906    66F8                  BNE.S    $000900
000908    4E75                  RTS

TENbug x.y > BR 900 908

BREAKPOINTS
000900    000900
000908    000908

TENbug x.y > G 900
PHYSICAL ADDRESS=00000900

AT BREAKPOINT
PC=00000900 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=000900

TENbug x.y > GD 900
PHYSICAL ADDRESS=00000900
```

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 000900)
000900    1018                MOVE.B   (A0)+,D0
000902    0C000000            CMP.B    #0,D0
000906    66F8                BNE.S    $000900
000908    4E75                RTS

TENbug x.y > BR 900 908

BREAKPOINTS
000900    000900
000908    000908

TENbug x.y > .PC 900

TENbug x.y > GT 906
PHYSICAL ADDRESS=00000906
PHYSICAL ADDRESS=00000900

AT BREAKPOINT
PC=00000906 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=000906

TENbug x.y > BR

BREAKPOINTS
000900    000900
000908    000908

TENbug x.y >
```

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

The function of the 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 long word (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 long word (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.

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.

See also: MD

EXAMPLES

```
TENbug x.y > M 1000 ;L
00001000 00000000 ? 200=
00001000 00000200 ? .

TENbug x.y > MM 40000 ;W;N
00040000 ? 555
00040002 ? 34.
```

```
MD[<port number>] <address> [<count>]
MDS <address> [<count>]
```

The MD command displays a portion of memory which begins at <address> and extends for the number of bytes given as <count>. If <count> is not specified, the default is $10 bytes or, if MDS is used, the default number of bytes is $100 (a section). The display is in hex and in ASCII characters.

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

Valid port numbers for this command are:

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 1 - 7201/A)
3	specifies TENbug port 3 (MVME400 port 2 - 7201/B)
4	specifies TENbug port 4 (MVME410 port 1 - PIA/A)
5	specifies TENbug port 5 (MVME410 port 2 - PIA/B)

After the MD command is entered, it will continue with the next 16 lines of output each time a carriage return (CR) is entered. Any other command exits MD and enters the new command.

See also: MM

EXAMPLES

```
TENbug x.y > MD 900 3F
000900    20 3C 00 00 43 43 24 19  D6 82 0C 82 7F FF FF FF   <..CC$.V........
000910    67 02 60 08 06 83 00 00  00 01 60 E4 04 83 00 00  G.@.........@D....
000920    00 01 60 DC 00 00 00 00  00 00 00 00 00 00 00 00  ..@\............
000930    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................

TENbug x.y > MD 0 FF
000000    00 00 08 04 00 F0 0F CC  00 F0 00 40 00 F0 00 4A  .....p.L.p.@.p.J
000010    00 F0 24 CE 00 00 1A 84  00 00 1A 86 00 00 1A 88  .p$N............
000020    00 00 1A 8A 00 00 1A 94  00 00 1A 8C 00 00 1A BE  ................
000030    00 00 18 60 00 00 18 60  00 00 18 60 00 00 18 60  ...'...'...'...'
000040    00 00 18 60 00 00 18 60  00 00 18 60 00 00 18 60  ...'...'...'...'
000050    00 00 18 60 00 00 18 60  00 00 18 60 00 00 18 60  ...'...'...'...'
000060    00 00 19 90 00 00 18 60  00 00 18 60 00 00 18 60  ...'...'...'...'
000070    00 00 18 60 00 00 18 60  00 00 18 60 00 00 18 60  ...'...'...'...'
000080    00 00 0F CE 00 00 10 96  00 00 1A 1E 00 00 1A 20  ...N............
000090    00 00 1A 22 00 00 1A 24  00 00 1A 26 00 00 1A 28  ...".../...&...(
0000A0    00 00 1A 2A 00 00 1A 2C  00 00 1A 2E 00 00 1A 30  ...*...,........0
0000B0    00 00 1A 32 00 00 1A 34  00 00 1A 36 00 F0 3C 82  ...2...4...6.p<.
0000C0    00 00 1A 3A 00 00 1A 3C  00 00 1A 3E 00 00 1A 40  ...:...<...>...@
0000D0    00 00 1A 42 00 00 1A 44  00 00 1A 46 00 00 1A 48  ...B...D...F...H
0000E0    00 00 1A 4A 00 00 1A 4C  00 00 1A 4E 00 00 1A 50  ...J...L...N...P
0000F0    00 00 1A 52 00 00 1A 54  00 00 1A 56 00 00 1A 58  ...R...T...V...X

TENbug x.y >
```

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 x.y > .R1 1000                                    Set offset R1.

TENbug x.y > .R3 33000

TENbug x.y > .R4 440000

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

TENbug x.y > .R6 -1

TENbug x.y > OF                                          Display offsets.
R0=00000000 R1=00001000 R2=00000000 R3=00033000
R4=00440000 R5=00000000 R6=FFFFFFFF R7=00000000

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

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

TENbug x.y > MM 10+R7                    R7 is always 0, so it overrides R0.
000010 00 ?.

TENbug x.y >
```

NOTE

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.

4.2.12 Printer Attach and Detach (PA and NOPA)

```
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 module, 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:

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.

```
EXAMPLE                         COMMENT

TENbug x.y > PA5

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

See also: MD

T[R] [<count>]

The 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 x.y :>). While in this mode, typing the single character (CR) 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 (TRAP, etc.).

COMMAND FORMAT	DESCRIPTION
TENbug x.y > T	Trace one instruction.
TENbug x.y :> TR <count>	Trace <count> instructions.
TENbug x.y :> (CR)	Carriage return (CR) executes next instruction.
TENbug x.y :> .	Typing a period (.) followed by a (CR) exits trace mode.

NOTE

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

See also: DF, GO, GT, TT

EXAMPLES

```
TENbug x.y > .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 :> T 2
PHYSICAL ADDRESS=00002002
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
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 x.y :> .

TENbug x.y >
```

TT <breakpoint address>

The TT command performs the following:

1. Sets a temporary breakpoint at the address specified.
2. Starts program execution in the trace mode.
3. Traces until any breakpoint with a zero count is encountered.
4. Resets the temporary breakpoint.

The temporary breakpoint is not displayed by the BR command.

See also: DF, GO, GT, TR

EXAMPLE

```
TENbug x.y > .PC 2000

TENbug x.y > 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 x.y :> .

TENbug x.y >
```

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 normal (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 x.y >
```

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

```
TENbug x.y 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 normal and high resolution modes appear in the VME/10 Microcomputer System Overview Manual.

EXAMPLE

```
TENbug x.y > VM

TENbug x.y m> VM

TENbug x.y >
```

4.3 COMMAND SUMMARY

Table 4-2 summarizes the commands and options available to TENbug users.

TABLE 4-2. TENbug Command and Option Summary

COMMAND	DESCRIPTION
BH [<device>][,<controller>]	Bootstrap halt
BO [<device>][,<controller>][,<string>]	Bootstrap operating system
[NO]BR [<address>[;<count>]]	Breakpoint set (and remove)
DF	Display formatted registers
G[O] [<address>]	Go
GD [<address>]	Go direct
GT <temporary breakpoint address>	Go until breakpoint
M[M] <address>[;<options>]	Memory modify; options W, L, O, V, N
MD[<port number>] <address> [<count>]	Memory display
OF	Display offsets
[NO]PA[<port number>]	Printer attach (and detach)
T[R] [<count>]	Trace
TT <breakpoint address>	Temporary breakpoint trace
VM	Video map
.A0-.A7 [<expression>]	Display/set address register (1)
.D0-.D7 [<expression>]	Display/set data register (1)
.DFC [<expression>]	Display/set destination function code (1)
.R0-.R6 [<expression>]	Display/set relative offset register (1)
.PC [<expression>]	Display/set program counter (1)
.SFC [<expression>]	Display/set source function code (1)
.SR [<expression>]	Display/set status register (1)
.SSP [<expression>]	Display/set supervisor stack pointer (1)

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

COMMAND		DESCRIPTION
.USP	[<expression>]	Display/set user stack pointer (1)
.VBR	[<expression>]	Display/set vector base register (1)
(BREAK)		Abort command
(DEL)		Delete character
(CTRL D)		Redisplay line
(CTRL H)		Delete character
(CTRL W)		Suspend output (2)
(CTRL X)		Cancel command line
(CR)		Process command line

NOTES:

(1) See description of .<register> command.

(2) When CTRL W is used, the output display can be continued by entering any character.

CHAPTER 5

TENbug ROUTINES AVAILABLE TO THE USER

5.1 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.

5.2 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 4.2.10 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
```

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 (see 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.

APPENDIX B

ERROR MESSAGES AND OTHER MESSAGES

ERROR MESSAGE	MEANING
PRINTER NOT READY	Printer is not properly connected or cannot receive output
SYNTAX ERROR	Error in command line
ERROR	Error (prefix)
ILLEGAL INSTRUCTION	Instruction used an illegal op-code
.... TRAP ERROR	See Traps in MC68010 16-Bit Virtual Memory Microprocessor product specification handbook
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	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

OTHER MESSAGE	MEANING
TENbug x.y >	TENbug 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 by command
.PC within "DEBUGGER"	Displayed by trace commands

APPENDIX C

CONFIGURATION AREA

Disks initialized using VERSAdos contain a Volume Identification Block and a Disk Configuration Block in VERSAdos sectors 0 and 1, respectively. TENbug looks for the string "EXORMACS" in locations $F8-$FF of sector 0 to validate the disk. 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:

VERSAdos 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($A)	2	VERSADOS SECTOR SIZE
12($C)	4	TOTAL VERSADOS 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 systems without VERSAdos 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 VERSAdos sector 0 are modified to contain the string "EXORMACS". TENbug then uses the following default values to access the disk. These default values will allow access of track 0 for all configurations.

PARAMETER DESCRIPTION	RWIN1 FLOPPY DISK	RWIN1 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	4F	1
Physical sector size of media	N/A	N/A
Precompensation cylinder number	28	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.