CHAPTER 3

USING THE MONITOR/DEBUG FIRMWARE

The MC68000 Educational Computer Board has a resident firmware package that provides a self-contained programming and operating environment. The firmware, aptly named "TUTOR", provides the user with monitor/debug, assembly/disassembly, program entry, and I/O control functions. Chapter 3 is a how-to-use description of the TUTOR package, including user interface and the command structure. Chapter 4 provides a detailed discussion of the assembler/disassembler functions called by the TUTOR firmware.

		Page
3.1	WHAT IS TUTOR?	3-3
3.2	OPERATIONAL PROCEDURE	3-4
3.2.1	System Turn-ON	3-4
3.2.2	System Initialization	3-4
3.2.2.1	RESET Button	3-5
3.2.2.2	ABORT Button	3-5
3.2.2.3	User Program	3-5
3.2.3	System Operation	3-5
3.3	TERMINAL CONTROL CHARACTERS	3-6
3.4	COMMAND LINE FORMAT	3-6
3.4.1	Expression as a Parameter	3-7
3.4.2	Address as a Parameter	3-7
3.4.2.1	Address Formats	3-7
3.4.2.2	Offset Registers	3-8
3.4.3	Command Echo Back	3-8
3.5	TUTOR COMMAND SET	3-9
3.5.1	BF - Block Memory Fill	3-11
3.5.2	BM - Block Move	3-12
3.5.3	BR - Breakpoint	3-13
3.5.4	BS - Block of Memory Search	3-14
3.5.5	BT - Block of Memory Test	3-15
3.5.6	DC - Data Conversion	3-16
3.5.7	DF - Display Formatted Registers	3-17
3.5.8	DU - Dump Memory (in S-Record Format)	3-18
3.5.9	GD - Go Direct Execute Program	3-19
3.5.10	GO - Execute Program	3-20
3.5.11	GT - Go Until Breakpoint	3-21
3.5.12	HE - Help	3-22
3.5.13	LO - Load (in S-Record Format)	3-23
3.5.14	MD - Memory Display	3-24
3.5.15	MM - Memory Modify	3-25
3.5.16	MS - Memory Set	3-28
3.5.17	NOBR - Remove Breakpoint	3-29
3.5.18	NOPA - Reset Printer Attach	3-30
3.5.19	OF - Offset	3-31
3.5.20	PA - Printer Attach	3-32
3.5.21	PF - Port Format	3-33
3.5.22	.Rx - Individual Register Display/Change	3-35
3.5.23	TM - Transparent Mode	3-36
3.5.24	TR - Trace	3-38
3.5.25	TT - Trace to Temporary Breakpoint	3-39
3.5.26	VE - Verify (in S-Record Format)	3-40
3.6	COMMAND SUMMARY AND MESSAGES	3-41

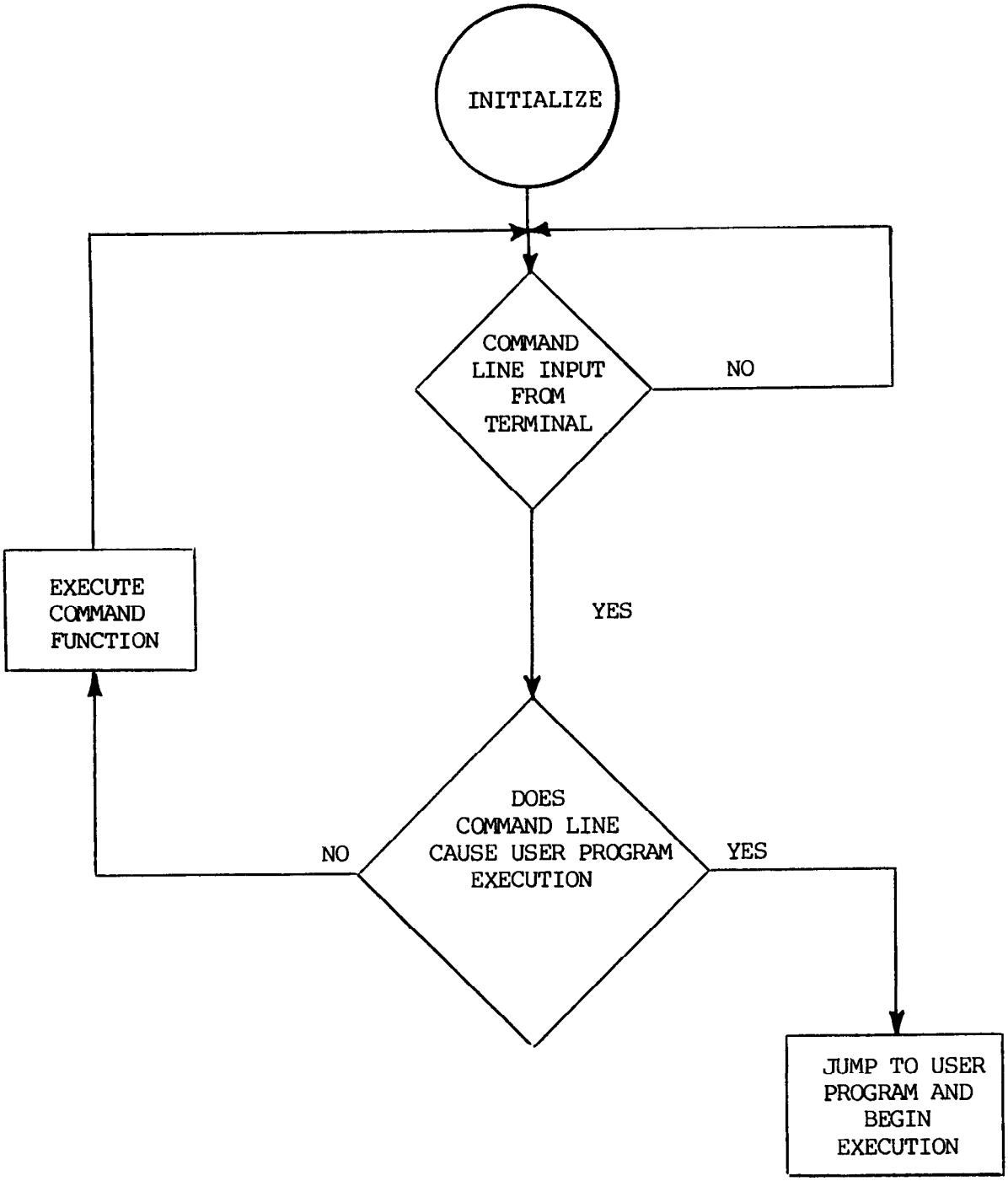

FIGURE 3-1. Flow Diagram of TUTOR Operational Mode

CHAPTER 3

USING THE MONITOR/DEBUG FIRMWARE

3.1 WHAT IS TUTOR?

TUTOR is the resident firmware package for the MC68000 Educational Computer Board. The 16K-byte firmware (stored in two 8Kx8 ROM or EPROM devices) provides a self-contained programming and operating environment. TUTOR interacts with the user through pre-defined commands that are entered via the terminal. The commands fall into four 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 MC68000.

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.

An additional function called the TRAP 14 handler allows the user program to utilize various routines within TUTOR. The TRAP 14 handler is discussed in Chapter 5.

The operational mode of TUTOR is demonstrated in Figure 3-1. After system initialization, the computer waits for a command line input from the user terminal. 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 TUTOR firmware may or may not be re-entered, depending on the discretion of the user. For the alternate case, the command will be executed under control of the TUTOR firmware, and after command completion, the system returns to a waiting condition. During command execution, additional user input may be required, depending on the command function.

The command format and syntax are similar to other Motorola products based on the MC68000. This is done so that a large relearning effort is not required when the user utilizes these other products.

3.2 OPERATIONAL PROCEDURE

CAUTION

POWER SUPPLIES MUST BE TURNED ON AND OFF IN PROPER SEQUENCE TO AVOID DAMAGE TO THE DYNAMIC RAM DEVICES. FOLLOW THE TURN-ON INSTRUCTIONS TO PREVENT PROBLEMS.

System turn-on and initial operation are described in detail in paragraph 2.4. This information is repeated here for convenience and to prevent possible damage.

3.2.1 System Turn-On

To power-up with individual power supplies:

a. All cables should be connected, making sure ground is connected common to all power supplies.

b. Turn on -12.0 Vdc.

c. Turn on +12.0 Vdc.

d. Turn on +5.0 Vdc.

Alternatively, if a single multivoltage supply is used, then:

e. Be sure all voltages are connected prior to power-up.

f. Turn power ON to the board.

3.2.2 System Initialization

The act of powering up the board will initialize the system. The processor is reset and TUTOR is invoked. After initialization, the terminal will print:

```
TUTOR 1.X >
```

where "X" is the revision number of the software.

NOTE

If this response does not appear, system checks may need to be performed as described in paragraph 2.4. The most common problem is that the terminal and board are not set up for matching baud rates. Also, slower terminals such as T.I. 700 series devices can have special requirements, which are discussed in Appendix B.

Other means can be used to re-initialize the Educational Computer Board firmware. These means are discussed in the following paragraphs.

3.2.2.1 RESET Button. RESET is the black button located on the lower edge of the board. Depressing this button causes all processes to terminate, resets the MC68000 processor and MC68230 PI/T, and restarts the TUTOR firmware. Pressing the RESET button should be the appropriate action if all else fails.

3.2.2.2 ABORT Button. ABORT is the red button located next to the RESET button at the lower edge of the board. The abort function causes an interrupt of the present processing (a level 7 interrupt on the MC68000) and gives control to the TUTOR firmware. This action differs from reset in that no processor register or memory contents are changed, the processor and peripherals are not reset, and TUTOR is not restarted. Also, in response to depressing the ABORT button, the contents of the MC68000 internal registers are displayed.

The abort function is most appropriate when software is being debugged. The user can interrupt the processor without destroying the present state of the system.

3.2.2.3 User Program. The user can return control of the system to the firmware by recalling TUTOR via his program. Instructions can be inserted into the user program to call TUTOR via THE TRAP 14 handler. See Chapter 5.

3.2.3 System Operation

After system initialization or return of control to TUTOR, the terminal will print:

```
TUTOR  1.X >
```

and wait for a response.

The user can call any of the commands supported by the firmware. 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.

NOTES

1. The user memory is located at addresses $000900-$007FFF. When first learning the system, the user should restrict his activities to this area of the memory map.

2. 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. This results in the terminal printing out a trap error message and the contents of all MC68000 registers. Control is returned to the TUTOR monitor. A bus trap error also occurs if the system attempts to write to ROM.

3.3 TERMINAL CONTROL CHARACTERS

Several keys are used as command line edit and control functions. It is best to be familiar with these functions before exercising the system. The functions include:

a. Delete (rubout) key or CTRL H - will delete the last character entered on the terminal.

b. CTRL X - will cancel the entire line.

c. CTRL D - will redisplay the entire line.

d. RETURN (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 pushed and held down and then the other key (H, X, D, or W) should be pushed.

These control characters are summarized with the command set in Table 3-2.

3.4 COMMAND LINE FORMAT

The command line format is:

TUTOR 1.X > [NO]<command> [<parameters>] [;<options>]

where:

TUTOR 1.X >	Is the prompt from the educational computer generated by TUTOR.
NO	Is the negative form (opposite) of primitive command.
command	Is the primitive command.
parameters	Are separated by spaces and can be of the form <expression> or <address>.
options	Multiple options may be selected.

NOTES

1. The command line format is defined using special characters which have the following syntactical meanings:

   [ ] Enclose optional fields.

   < > Enclose a syntactical variable.

   These characters are not entered by the user, but are for definition only.

2. Fields are separated by one or more spaces used as a delimiter.

The basic command form consists of the primitive command field and the parameters field, although some primitives do not require parameters. The additional command negation and options fields 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.

3.4.1 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 ($).

3.4.2 Address as a Parameter

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

3.4.2.1 Address Formats.

FORMAT	EXAMPLE	DESCRIPTION
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)

3.4.2.2. Offset Registers. Eight software registers (not actually hardware configured) are used to modify addresses contained in TUTOR 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 3.5.19, and modified by the .RX command, paragraph 3.5.22.

The offset registers are always reset to zero at power-up or by activating the reset button. 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
BR   10+R0
```

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 are set to zero by adding R7 (always zero) to zero.

Example:

```
.R1   8        R1 = 8              Offset R0 is zero, R1 is set to 8
.R0   100      R0 = 100
.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              R0 set to zero
```

3.4.3 Command Echo Back

Most commands that require parameters display back to the user the information entered, but in a physical format so that the user sees the expression or address results. Some error checking is done -- for example, if an address will cause an obvious error, the message INVALID ADDRESS=XXXXXXXX will result on the terminal connected to serial port 1. Refer to Table 3-3 for the error messages and other messages used in TUTOR.

3.5. TUTOR COMMAND SET

Table 3-1 lists the TUTOR commands by type.

TABLE 3-1. TUTOR Commands

COMMAND MNEMONIC	DESCRIPTION	PAGE
MD	Memory Display	3-24
MM, M	Memory Modify	3-25
MS	Memory Set	3-28
.A0 - .A7	Display/Set Address Register	3-35
.D0 - .D7	Display/Set Data Register	3-35
.PC	Display/Set Program Counter	3-35
.SR	Display/Set Status Register	3-35
.SS	Display/Set Supervisor Stack Pointer	3-35
.US	Display/Set User Stack Pointer	3-35
DF	Display Formatted Registers	3-17
OF	Display Offsets	3-31
.R0 - .R6	Display/Set Relative Offset Register	3-31
BF	Block of Memory Fill	3-11
BM	Block of Memory Move	3-12
BT	Block of Memory Test	3-15
BS	Block of Memory Search	3-14
DC	Data Conversion	3-16
BR	Breakpoint Set	3-13
NOBR	Breakpoint Remove	3-29
GO, G	Go	3-20
GT	Go Until Breakpoint	3-21
GD	Go Direct	3-19
TR, T	Trace	3-38
TT	Temporary Breakpoint Trace	3-39
PA	Printer Attach	3-32
NOPA	Reset Printer Attach	3-30
PF	Port Format	3-33
TM	Transparent Mode	3-36
*	Send Message to Port 2	--
HE	Help	3-22
DU	Dump Memory	3-18
LO	Load	3-23
VE	Verify	3-40

Each of the individual commands is described in the following pages. Figure 3-2 shows the general format of the description.

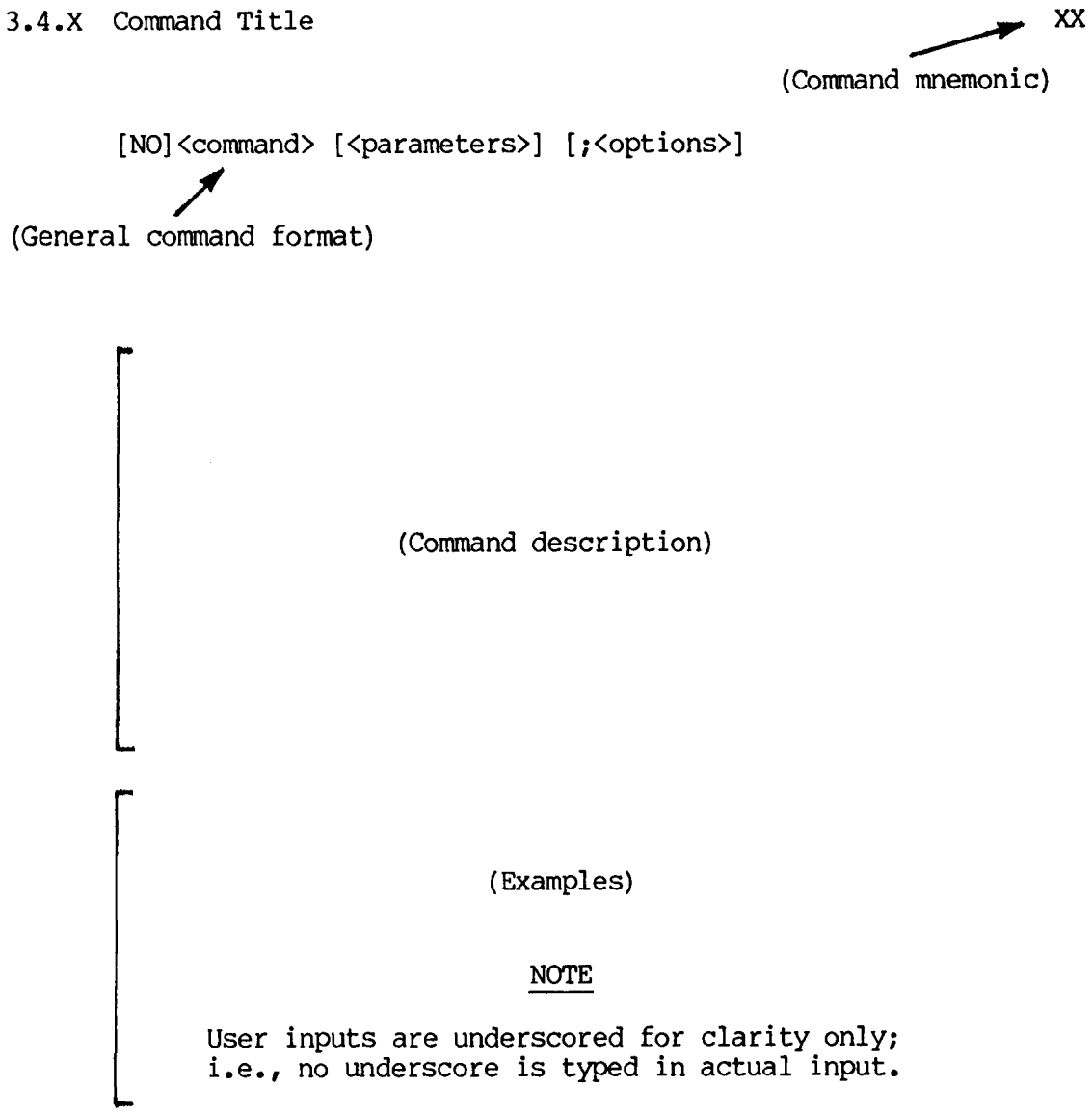

FIGURE 3-2. Command Description Format

BF <address1> <address2> <word>

The BF command fills memory starting with the word boundary (even address) <address1> through <address2>. Both <address1> and <address2> must be even addresses. This command only fills with a word-size (two-byte) data pattern, as specified in hex, octal, decimal, or binary digits. If an entire word size data pattern is not entered, the pattern is right justified and leading zeros are inserted.

EXAMPLE

```
TUTOR  1.X > MD 2004
002004    17 39 2A 33 BF FF 00 9E  41 42 55 CD C4 44 00 98  .9*3?...ABUMDD..

TUTOR  1.X > BF 2004 200A 475A
PHYSICAL ADDRESS=00002004 0000200A

TUTOR  1.X > MD 2004
002004    47 5A 47 5A 47 5A 47 5A  41 42 55 CD C4 44 00 98  GZGZGZGZABUMDD..

TUTOR  1.X > BF 2004 2012 7
PHYSICAL ADDRESS=00002004 00002012

TUTOR  1.X > MD 2004
002004    00 07 00 07 00 07 00 07  00 07 00 07 00 07 00 07  ................

TUTOR  1.X >
```

BM <address1> <address2> <address3>

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

<address1> = beginning address of source memory block
<address2> = ending address of source memory block
<address3> = beginning address of destination memory block

EXAMPLE 1

```
TUTOR  1.X > BM 1800 1900 1860
PHYSICAL ADDRESS=00001800 00001900
PHYSICAL ADDRESS=00001860

TUTOR  1.X >
```

The entire block from $1800 through $1900 is duplicated, starting at $1860.

EXAMPLE 2

```
TUTOR  1.X > MD 1800 10
001800    00 11 22 33 44 55 66 77  88 99 AA BB CC DD EE FF  .."3DUfw..*;L]n.

TUTOR  1.X > BM 1806 1809 1804
PHYSICAL ADDRESS=00001806 00001809
PHYSICAL ADDRESS=00001804

TUTOR  1.X > MD 1800 10
001800    00 11 22 33 66 77 88 99  88 99 AA BB CC DD EE FF  .."3fw....*;L]n.

TUTOR  1.X >
```

3.5.3 Breakpoint BR

BR [<address>[;<count>]]...

When encountered, a breakpoint causes program execution to stop and control to be transferred to TUTOR. 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 user 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 can 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.

The NOBR command is used to eliminate all breakpoints from the breakpoint table.

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

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > BR	Display all breakpoints.
TUTOR 1.X > BR address	Set a breakpoint.
TUTOR 1.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.

See also: GT, NOBR, TT

EXAMPLE

```
TUTOR 1.X > .R4 4000
TUTOR 1.X > BR 1010 2000;5 2040 4000
BREAKPOINTS
001010     001010
002000     002000;5
002040     002040
000000+R4  004000

TUTOR 1.X > NOBR 1010 2040
BREAKPOINTS
002000     002000;5
000000+R4  004000

TUTOR 1.X > NOBR
BREAKPOINTS
TUTOR 1.X >
```

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

The BS command has two modes: 1) literal string search, and 2) 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>. 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 one's. The options supported are:

```
;B   byte
;W   word
;L   long word
```

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

```
TUTOR  1.X > MD 1FF0 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............

TUTOR  1.X > BS 1FF0 200F 'CC'
PHYSICAL ADDRESS=00001FF0 0000200F
002000 'CC'

TUTOR  1.X > BS 1FF0 200F 34 ;W
PHYSICAL ADDRESS=00001FF0 0000200F

TUTOR  1.X > BS 1FF0 200F 03 0F
PHYSICAL ADDRESS=00001FF0 0000200F
002000 43
002001 43

TUTOR  1.X > BS 1000 7FFE 4AFB;W
PHYSICAL ADDRESS=00001000 00007FFE
001000 4AFB

TUTOR  1.X >
```

COMMENT

Successful search for literal string 'CC'.

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

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

Successful search for "leftover" breakpoints.

3.5.5 Block of Memory Test BT

BT <address1> <address2>

The BT command is a destructive test of a block of memory beginning at <address1> through <address2> inclusive (both at word boundaries = even addresses). If this test runs to completion without detecting errors, the memory tested will be set to all zeros.

This command may take several seconds to test large blocks of memory.

If memory problems are found, a message is displayed indicating the address, the data stored, and the data read of the failing memory.

EXAMPLE

```
TUTOR  1.X > BT 5000 5FFE
PHYSICAL ADDRESS=00005000 00005FFE

TUTOR  1.X > BT 6000 6040
FAILED AT 6000 WROTE=FFFE READ=FF00
```

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.

Offsets may be used with the DC command. R0 is used if the offset is not specified.

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

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > DC $data	Convert hexadecimal data into hexadecimal and decimal.
TUTOR 1.X > DC &data	Convert decimal data into hexadecimal and decimal.

EXAMPLE

```
TUTOR 1.X > DC &120
$78=&120

TUTOR 1.X > DC &15+$4-$13
$0=&0

TUTOR 1.X > DC -1000
$FFFFF000=-$1000=-&4096

TUTOR 1.X >

TUTOR 1.X > .R0 1000

TUTOR 1.X > OF
R0=00001000 R1=00000000 R2=00000000 R3=00000000
R4=00000000 R5=00000000 R6=00000000 R7=00000000

TUTOR 1.X > DC 10+10+30
$1050=&4176

TUTOR 1.X > DC 10+10+30+R7
$50=&80
```

DF

The DF command is used to display the MC68000 processor registers. The trace display will be displayed whenever the debugger gains control of the program execution -- i.e., at breakpoints and when tracing.

Note that any single register can be displayed with the .Ax, .Dx, etc., commands.

See also: .Rx (contains forms .Ax, .Dx, etc.)

EXAMPLE

```
TUTOR  1.X > DF
PC=00001000 SR=2700=.S7..... US=00002000 SS=00000F00
D0=FFFFFFFF D1=00000000 D2=00000000 D3=00000000
D4=B0000018 D5=0000003F D6=00000000 D7=00000000
A0=00010040 A1=00000638 A2=00001000 A3=00000542
A4=00000544 A5=0000053A A6=0000053A A7=00000F00
--------------------001000    0C000030              CMP.B    #48,D0
```

NOTE

Any time the registers are displayed, a disassembled line of code is also displayed (see Chapter 4 for format). The instruction located at the address pointed to by the program counter (example, $001000) is disassembled and shown. This is useful for program debugging.

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

The DU command outputs S-records of memory contents from <address1> through <address2> to <port number>. Any optional text is output as part of a header record.

S-records are a standard data format used in transmitting and receiving programs and data. Appendix A discusses them in more detail. As part of the memory dump, any text goes out as an S0 or header record, and transmission ends with an S9 end-of-file record.

The DU command has several options, including dump to Ports 1, 2, 3, and 4. Also a default case of DU dumps to Port 1. The variations include:

COMMAND	PORT #	DESTINATION
DU	Port 1	Terminal
DU1	Port 1	Terminal
DU2	Port 2	Host (modem)
DU3	Port 3	Printer
DU4	Port 4	Audio Cassette

This command does not send control characters to start or stop I/O devices. The offset contained in offset register R0 is added to the starting and ending memory addresses.

See also: LO, VE

EXAMPLE

```
TUTOR 1.X > DU 8800 880F TUTOR 1.X
PHYSICAL ADDRESS=00008800 0000880F
S00C00005455544F5220312E587E
S11388006654BBCE6704610013521E3C004447F813
S9030000FC

TUTOR 1.X > DU1 8800 880F
PHYSICAL ADDRESS=00008800 0000880F
S0030000FC
S11388006654BBCE6704610013521E3C004447F813
S9030000FC

TUTOR 1.X > DU4 7000 70FF                    S-records are dumped to
PHYSICAL ADDRESS=00007000 000070FF           Port 4 (tape recorder)

TUTOR 1.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 (locations $0 through $3FF). If address is not specified, the GD command starts the target program at the address in the PC.

See also: GO, GT

EXAMPLE

```
TUTOR 1.X > GD 2000
PHYSICAL ADDRESS=00002000
```

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

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

a. the target program encounters a breakpoint,

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

c. operator intervention through the RESET or ABORT pushbutton switch.

NOTE

If breakpoints with count are encountered, real time is not achieved. The breakpoint will not stop processing until Count is diminished to zero, but processing overhead is required.

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

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > GO	Begin execution at address in PC.
TUTOR 1.X > GO address	Set PC = address and begin execution at that address.

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

EXAMPLE

```
TUTOR 1.X > MM 2000;L
002000    00002F00 ?3000.

TUTOR 1.X > GO [2000]          Address is memory indirect.
PHYSICAL ADDRESS=00003000
```

3.5.11 Go Until Breakpoint GT

GT <breakpoint address>

The GT command performs the following:

1. sets a temporary breakpoint,
2. sets breakpoints entered by the BR command,
3. sets target program registers as displayed by the DF command,
4. 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.

If the breakpoint address is in the breakpoint table, the message ERROR and the breakpoint table are displayed.

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

EXAMPLE

```
TUTOR  1.X > BR 2010 3000

BREAKPOINTS
002010    002010
003000    003000

TUTOR  1.X > DF
PC=00002000 SR=A704=TS7..Z.. US=FFFFFFFF SS=000007BC
D0=FFFF1230 D1=00101200 D2=FED01210 D3=00000000
D4=FFFF0031 D5=FFFFFF2C D6=00000002 D7=00000000
A0=00010040 A1=FFFFFFFF A2=00000454 A7=0000054E
A4=00009F38 A5=0000053A A6=0000053A A7=000007BC
--------------------002000    0C000030            CMP.B   #48,D0

TUTOR  1.X > GT 2006
PHYSICAL ADDRESS=00002006
PHYSICAL ADDRESS=00002000

TUTOR  1.X > GT 2010                    Temporary breakpoint address $2010 is
PHYSICAL ADDRESS=00002010               already in breakpoint table.
ERROR
002010    002010
003000    003000
```

HE

The HE command gives the user information as to available commands.

EXAMPLE

```
TUTOR  1.X > HE
.PC .SR .US .SS
.D0 .D1 .D2 .D3. D4 .D5 .D6 .D7
.A0 .A1 .A2 .A3 .A4 .A5 .A6 .A7
.R0 .R1 .R2 .R3 .R4 .R5 .R6

BF    BM    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
```

3.5.13 Load (in S-Record Format) LO

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

The LO command moves object data in S-record format from an external device (Port 1, Port 2, or Port 4) to memory. Appendix A discusses S-records in more detail.

The command has the basic forms:

COMMAND	PORT #	SOURCE
LO1	Port 1	Terminal (i.e., terminal with tape drive)
LO	Port 2	Host (modem) - default port
LO2	Port 2	Host (modem)
LO4	Port 4	Audio cassette

The options include:

;-C Ignore the S-record checksum while loading. A checksum is contained in each S-record. If this option is not selected, the received checksum is compared with the calculated checksum. If they do not agree, the message CHKSUM= and the calculated checksum are sent to Port 1. The data is not loaded into memory if the checksums do not agree.

;X Echo data read from the source port onto the Port 1 terminal.

The optional [=text] is used only with Port 2. The text following the "=" is sent to Port 2. In this manner, a message can be sent to Port 2 to start a download, as an example.

A timeout feature is available for Port 2. If the host connected to Port 2 does not respond within approximately 10 seconds, the message TIMEOUT is sent to Port 1 and the LO command is aborted.

Several characteristics of this command are important to note:

a. The offset contained within register R0 is added to the addresses for the data contained within the S-record.

b. Any record not containing an S0, S1, S2, S8, or S9 string is ignored.

c. If an error occurs, causing the system to print out an error message, one or more lines sent during the error message may be ignored. The system cannot be printing and processing incoming data at the same time. To prevent the loss of information, the ECB can send characters to the host to stop and start the transfer of the S-records. Paragraph 4.5.2 describes this feature.

See also: DU, OF, PF, VE

EXAMPLES	COMMENT
TUTOR 1.X > LO ;X=COPY FILE.MX,#CN	Download from Port 2 with echo option.
TUTOR 1.X > LO;X-C=COPY FILE.MX,#	Download from Port 2 without verifying checksum.

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

The MD command is used to display a section of memory beginning at <address> and displaying the number of bytes given as <count>. Two modes are used for the data display -- that is, hex data (with equivalent ASCII) and disassembled form.

The command has the basic forms:

COMMAND	PORT #	DESTINATION
MD	Port 1	Terminal - default Port
MD1	Port 1	Terminal
MD2	Port 2	Host (modem)
MD3	Port 3	Printer

For <count>, the default condition is 16 bytes when no option is specified and one instruction or directive when the disassemble option is used.

Only one option is specified; therefore, only two output forms are used:

1. No option specified -- will display the data in hex and in equivalent ASCII. Data is always displayed in groups of 16 bytes. If the count is not on a 16-byte boundary, the next highest group of 16 will be displayed, unless the count is on a 16-byte boundary plus one, in which case the next highest group of 16 will not be displayed.

   Once 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.

2. ;DI -- invokes the disassembler function. The data is displayed in the disassembled format described in Chapter 4. Included in the instruction display is the address of the opcode, hexadecimal instruction code, instruction mnemonic, and operands. The MD command will display all instructions whose op code is contained within the byte count.

See also: MM, MS

EXAMPLE

```
TUTOR  1.X > MD 1000 12
001000    0C 00 00 30 6D 28 0C 00  00 39 6E 10 02 80 00 00  ...0m(...9n.....
001010    00 0F 11 C0 10 38 1E 3C  00 E4 4E 4E 0C 00 00 41  ...@.8.<.dNN...A

TUTOR  1.X > MD 1000 12 ;DI
001000    0C000030            CMP.B   #48,D0
001004    6D28                BLT.S   $00102E
001006    0C000039            CMP.B   #57,D0
00100A    6E10                BGT.S   $00101C
00100C    02800000000F        AND.L   #15,D0
```

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

The MM command is used to display memory and, as required, modify data or enter new data. The command has two basic forms:

a. Hexadecimal format - The standard form of the MM command displays the address and data at that location. The size option (byte, word, and long word) controls the number of bytes displayed for each address:

OPTION	DESCRIPTION
- (default)	Displays one byte
;W	Displays one word (2 bytes)
;L	Displays one long word (4 bytes)
;O	Displays one byte; accesses only odd address bytes
;V	Displays one byte; accesses only even address bytes
;N	Do not verify; do not read data stored

NOTE: If multiple options are desired, a semicolon (;) must precede each option.

Once entered, the MM command has several submodes of operation that allow modification and verification of data. The subcommands are in the format:

[<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, MS

EXAMPLES

```
TUTOR  1.X > MM 2000;W
002000    2200 ?FFFF
002002    2A07 ?DDDD
002004    60FA ?EEEE^
002002    DDDD ?
002004    EEEE ?
002006    5FFF ?AAAA=
002006    AAAA ?.

TUTOR  1.X > MM 4000;W;N
004000     ?555
004002     ?34.
```

b. ;DI - This option invokes the disassembler/assembler function. The address entered should be the starting address for an instruction (op code) word. The instruction will then be displayed in disassembled form. The disassembled format is described in Chapter 4.

The displayed instruction is followed by a question mark (?) that indicates a new source line may be entered. If a new line is entered, the instruction is immediately assembled, stored, and displayed. To enter a new line, the following format is used:

? <sp> <operation field> <sp> <operand field>(cr)

where:

sp	Is required because no labels are allowed and the format matches the resident assembler.
operation field	Is the MC68000 mnemonic or DC.W directive.
sp	Is a required delimiter.
operand field	Is normally source and destination fields.
cr	Enters new instruction.

The format is discussed in detail in Chapter 4.

Upon entry of the carriage return, the new instruction will overwrite the old instruction and enter the new one. To exit the command, a period (.) is entered immediately after the question mark, followed by a carriage return.

? . cr

NOTE

When inserting new instructions or modifying existing code, the assembler may overwrite the following code. Care must be taken by the programmer to take this factor into account. When moving code, be aware that address vectors may change.

If an error is found in the new instruction, the new line is redisplayed with an "X" immediately under the field suspected of causing a problem in the assembler. The "X" is followed by a question mark to allow re-entry of the corrected souce line.

EXAMPLES

1. TUTOR 1.X > MM 3000;DI
003000 5555 SUBQ.W #2,(A5) ? MOVE.L A0,A1

The assembler overwrites line 3000 and displays:

```
003000   2248                MOVE.L  A0,A1
003002   1211                MOVE.B  (A1),D1 ?
```

2. TUTOR 1.X > MM 3000;DI
```
003000   2248                  MOVE.L   A0,A1 ?
003002   6600E384              BNE.L    $001388 ? MOVE.L A0A1
```

The assembler overwrites line 3002 and displays:

```
003002                         MOVE.L  A0A1
                                       X?
```

An error was found with the operands. The corrected line can now be entered.

MS <address> <data...>

The MS command alters memory by setting data into the address specified. The data can take the form of ASCII string or hexadecimal data. Several strings can be entered; however, size is limited to eight characters.

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > MS 2000 'ABC'	Set memory to ASCII string.
TUTOR 1.X > MS 2003 4445	Set memory to hexadecimal data.
TUTOR 1.X > MS 2005 12345678 12	Size can be up to 8 characters.

See also: MD, MM

EXAMPLE

```
TUTOR  1.X > MD 2000
002000    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................

TUTOR  1.X > MS 2000 'ABC'

TUTOR  1.X > MS 2006 123 123456

TUTOR  1.X > MD 2000
002000    41 42 43 00 00 00 01 23  12 34 56 00 00 00 00 00   ABC....#.4V.....
```

NOBR [<address> <address>....]

The NOBR command is used to remove one or more breakpoints from the internal breakpoint table, and functions as the inverse of the BR command.

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > NOBR	Clear all breakpoints.
TUTOR 1.X > NOBR address	Clear a specific breakpoint.

See also: BR, GT, TT

EXAMPLE

```
TUTOR 1.X > .R3 3000

TUTOR 1.X > OF
R0=00000000 R1=00000000 R2=00000000 R3=00003000
R4=00000000 R5=00000000 R6=00000000 R7=00000000

TUTOR 1.X > BR 2000;5 2030 3000;6 3060

BREAKPOINTS
002000     002000;5
002030     002030
000000+R3  003000;6
000060+R3  003060

TUTOR 1.X > NOBR 3000

BREAKPOINTS
002000     002000;5
002030     002030
000060+R3  003060

TUTOR 1.X > NOBR

BREAKPOINTS

TUTOR 1.X >
```

NOPA

The NOPA command allows the user to detach the line printer from the Port 1 terminal.

See also: PA

OF

The OF command displays the offsets contained within registers R0-R7. To help the user with relocatability and position-independent code, seven general-purpose offsets (.R0-.R6) are provided. Offset .R7 is always zero, which provides a convenient way of zeroing other offsets or 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
BR 10+R0
```

It should also be noted when setting offsets, R0 is added to the expression being entered into the register. To zero a register, use the form:

```
.RX  0+R7          (X = desired register)
```

The form for setting individual registers is given in the Individual Register Display/Change (.Rx) command.

See also: .Rx

EXAMPLE

```
TUTOR  1.X > .R1 1000
TUTOR  1.X > .R3 3300
TUTOR  1.X > .R5 0+R7

TUTOR  1.X > OF
R0=00000000 R1=00001000 R2=00000000 R3=00003300
R4=00000000 R5=00000000 R6=00000000 R7=00000000

TUTOR  1.X > BR D08 1056

BREAKPOINTS
000D08     000D08
000056+R1  001056

TUTOR  1.X > .R0 2000

TUTOR  1.X > BR 10

BREAKPOINTS
000D08     000D08
000056+R1  001056
000010+R0  002010

TUTOR  1.X > MM 1000+R7
000000+R1  0C ?.

TUTOR  1.X >
```

COMMENT

```
Set offset R1.
Set offset R3.
Reset offset R5.

Set offset R0.

Offset R0 is added to the breakpoint
address. Absolute addresses are on
the right. Addresses relative to
the appropriate offset are displayed
on the left. The appropriate offset
is the nearest offset that is less
than or equal to the absolute address.

To access address $1000 with no offset,
offset R7 (always 0) must be added;
otherwise, offset R0 will automatically
be added.
```

PA

The PA command allows the user to attach the line printer so that information sent to the Port 1 terminal will also be printed. (The printer is physically attached to parallel Port 3 of the board. See the initial setup instructions.)

The printer can also be called by the Dump (DU3) and Memory Display (MD3) commands.

If the printer is deselected or not ready, the message PRINTER NOT READY will be sent to Port 1; TUTOR will wait until the printer is ready or the BREAK key is pushed.

See also: NOPA, DU, MD

3.5.21 Port Format PF

PF[<port number>]

The PF command allows the user to display or assign the characteristics of serial I/O Port 1 and Port 2. Each of these ports may be individually programmed for stop bits, character nulls, and carriage return nulls. The baud rates are selected via jumpers (see Chapter 2).

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > PF	Display both Port 1 and Port 2 formats.
TUTOR 1.X > PF1	Change Port 1 format.
TUTOR 1.X > PF2	Change Port 2 format.

Parameters include:

a. FORMAT - This two-character (8-bit) parameter determines the number of stop bits after each byte. Transmission used within TUTOR is 8 bits/byte and restricted to one or two stop bits/bytes. Therefore, for stop bits denoted by FORMAT=, entering:

   15 causes 1 stop bit (default)
   11 causes 2 stop bits

   NOTE: This command alters the control register on the MC6850 ACIA. For more information, see the I/O Chapter 6 and the MC6850 data sheet.

b. CHAR NULL - This parameter is the number of nulls sent after each character (default = 00).

c. CR NULL - This parameter is the number of nulls sent after each carriage return/line feed (line of data). (Default = 00). Hard copy terminals usually require four nulls.

d. OPTIONS - This is the address in RAM where the 6-byte options variable is located. The first and second bytes represent the transfer on and transfer off bytes, respectively, which are used to stop and start the transfer of S-records. (Refer to paragraph 4.5.2.) The third and fourth bytes are used with low baud rate or mechanical terminals to control the display. These bytes are discussed in Appendix B. The last two bytes contain the trailing and exit characters used in the transparent mode of operation (paragraph 3.5.23). All bytes are set to their initial (power up) values when the RESET button is pressed.

EXAMPLES

```
TUTOR 1.X > PF

FORMAT=   15 15
CHAR NULL=00 00
C/R  NULL=00 00
OPTIONS@XXXXXX
```

```
TUTOR  1.X > PF1
FORMAT=   15?11
CHAR NULL=00?
C/R  NULL=00?

TUTOR  1.X > PF2
FORMAT=   15?
CHAR NULL=00?3
C/R  NULL=00?8

TUTOR  1.X >
```

NOTE

TI 700 series terminals should have the following port characteristics:

BAUD RATE	FORMAT	CHAR NULL	C/R NULL
110	11	0	1
150	15	0	1
300	15	0	4
1200	15	3	17
2400	15	7	2F

See Appendix B.

.A0, .A1, .A2, .A3, .A4, .A5, .A6, .A7
.D0, .D1, .D2, .D3, .D4, .D5, .D6, .D7
.PC, .SR, .SS, .US

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

.A0 - .A7	address register
.D0 - .D7	data register
.R0 - .R6	relative offset register (software register)
.PC	program counter
.SR	status register (in the MC68000)
.SS	supervisor stack pointer
.US	user stack pointer

EXAMPLE	COMMENT
TUTOR 1.X > .PC .PC=00001010	Display program counter.
TUTOR 1.X > .A7 1300	Set address register seven.
TUTOR 1.X > .R5 5500	Set relative offset register five.

```
TUTOR 1.X > DF
PC=00001010 SR=2704=.S7..Z.. US=FFFFFFFF SS=00001300
D0=0000D0D0 D1=0000D1D1 D2=0000D2D2 D3=00D3D3D3
D4=D4D4D4D4 D5=000000D5 D6=00000000 D7=00000000
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00001300
--------------------001010    FFFF                DC.W    $FFFF
```

See also: DF, OF

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

The TM command connects the two serial ports of the board together and ignores all input/output between them until the exit character is entered from the terminal attached to serial port 1. The default exit character is CTRL A ($01).

In the transparent mode, the ECB monitors the data transfer only until it sees the exit character; the exit character, therefore, is also transmitted to the host. The ECB must send another character, called the trailing character, to the host to remove or cancel the exit character from the host's buffer. Otherwise, the exit character will still be in the buffer the next time the transparent mode is entered. The default trailing character is CTRL X ($18). Other trailing characters may be selected.

NOTE

In order to enter a trailing character, an exit character must first be entered. Otherwise, the intended trailing character will be interpreted as the exit character.

Some possible exit or trailing characters such as NUL ($00), space ($20), backspace ($08), end of transmission ($04), cancel ($18), line feed ($0A), and carriage return ($0D) cannot be specified as part of the TM command line. These characters are used as separators or control characters or are ignored by the command interpreter. To use these types of characters as exit and trailing characters, the character must be written to RAM using the MM command. The trailing and exit characters are the fifth and sixth bytes, respectively, of the 6-byte options variable described in paragraph 3.5.21.

An asterisk (*) as the first character of the command line means to transmit the rest of the line to the host (port 2).

The modem connected to port 2 should operate at the same baud rate as the terminal connected to port 1.

See also: LO, PF, VE

EXAMPLE	COMMENTS
TUTOR 1.X >	Startup or reset condition.
TUTOR 1.X > TM	Command to enter transparent mode.
TRANSPARENT EXIT=$01 = CTL A	TUTOR prints this; EXIT=$01=CTL A means that in order to exit this mode, the operator must enter CTRL A.

User talks directly to host, uses editor, assembler, etc.

CTRL A	Ends the transparent mode.
TUTOR 1.X >	TUTOR prints this and system is ready for new command.

NOTE: Other exit and trailing characters can be specified. As examples,

```
TUTOR  1.X > TM CTRL R

*TRANSPARENT* EXIT=$12 = CTL R
```

or

```
TUTOR  1.X > TM 7 *              Trailing character=$2A=*

*TRANSPARENT* EXIT=$37 = 7

TUTOR  1.X > PF

FORMAT=   15 15
CHAR NULL=00 00
C/R  NULL=00 00
OPTIONS@0004E6

TUTOR  1.X > MM 4EA              Enter NULL ($00) trailing character directly
0004EA   2A ?0.                  into option byte.

TUTOR  1.X >
```

TR [<count>]
T [<count>]

The TR command executes instructions, one at a time, beginning at the location pointed to by the program counter. After execution of each instruction, the processor registers are displayed, and the instruction pointed to by the program counter is disassembled.

After the trace mode is entered, the prompt includes a colon (i.e., TUTOR 1.X :>). While in this mode, the single character, carriage return, will cause one instruction to be traced. To exit, any command may be entered, followed by a carriage return.

Breakpoints and breakpoint counts are in effect during trace.

Limited tracing can be done within the TUTOR firmware. However, the maximum count is one. Because the stacks are shared by the trace command and the rest of TUTOR, they may become jumbled up when tracing is done in the debugger.

COMMAND FORMAT	DESCRIPTION
TUTOR 1.X > TR	Trace one instruction.
TUTOR 1.X :> T count	Trace "count" (hex) instructions.
TUTOR 1.X :>	Carriage return (CR) executes next instruction.

See also: DF, GO, GT, TT

EXAMPLE

```
TUTOR  1.X > .R6 2000

TUTOR  1.X > .PC 0+R6

TUTOR  1.X > TR 3
PHYSICAL ADDRESS=00002000
PC=00002002 SR=2700=.S7..... US=FFFFFFFF SS=000007BC
D0=0030FF43 D1=0030FF43 D2=0FFFFFFC D3=00000000
D4=FFFFFFFC D5=FFFFFFFC D6=00000002 D7=00000000
A0=00010040 A1=00002004 A2=000007B6 A3=0000053A
A4=00002004 A5=0000053A A6=000007B6 A7=000007BC
--------------------000002+R6 45F82056            LEA.L    $00002056,A2
PC=00002006 SR=2700=.S7..... US=FFFFFFFF SS=000007BC
D0=0030FF43 D1=0030FF43 D2=0FFFFFFC D3=00000000
D4=FFFFFFFC D5=FFFFFFFC D6=00000002 D7=00000000
A0=00010040 A1=00002004 A2=00002056 A3=0000053A
A4=00002004 A5=0000053A A6=000007B6 A7=000007BC
--------------------000006+R6 4EF900008152        JMP      $00008152

.PC within "DEBUGGER".
PC=00008152 SR=2700=.S7..... US=FFFFFFFF SS=000007BC
D0=0030FF43 D1=0030FF43 D2=0FFFFFFC D3=00000000
D4=FFFFFFFC D5=FFFFFFFC D6=00000002 D7=00000000
A0=00010040 A1=00002004 A2=00002056 A3=0000053A
A4=00002004 A5=0000053A A6=000007B6 A7=000007BC
--------------------006152+R6 48B800010406        MOVEM.W  D0,$0406

TUTOR  1.X :>
```

3.5.25 Trace to Temporary Breakpoint TT

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 at the address specified in the program counter (PC) (see TR command).

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 COMMENT

```
TUTOR  1.X > .PC 2000

TUTOR  1.X > TT 2004
PHYSICAL ADDRESS=00002004                  Temporary breakpoint address - $2004
PHYSICAL ADDRESS=00002000                  Execution address - $2000
PC=00002002 SR=2708=.S7.N... US=FFFFFFFF SS=000007BC
D0=BDBC4144 D1=BDBC4144 D2=FFFFFFFF D3=FFFFFFFF
D4=FFFFFFFF D5=FFFFFFFF D6=FFFFFFFF D7=FFFFFFFF
A0=FFFFFFFB A1=FFFFFFFF A2=FFFFFFFF A3=FFFFFFFF
A4=FFFF7FFF A5=FFFFFFFF A6=FFFFFFFF A7=000007BC
--------------------002002    2A07                MOVE.L   D7,D5

AT BREAKPOINT
PC=00002004 SR=2708=.S7.N... US=FFFFFFFF SS=000007BC
D0=BDBC4144 D1=BDBC4144 D2=FFFFFFFF D3=FFFFFFFF
D4=FFFFFFFF D5=FFFFFFFF D6=FFFFFFFF D7=FFFFFFFF
A0=FFFFFFFB A1=FFFFFFFF A2=FFFFFFFF A3=FFFFFFFF
A4=FFFF7FFF A5=FFFFFFFF A6=FFFFFFFF A7=000007BC
--------------------002004    60FA                BRA.S    $002000
```

3.5.26 Verify (in S-Record Format) VE

VE <port number>[;=text]

The VE command verifies the current contents of memory with the object data in S-record format from a device external to memory. The device can be attached to Port 1, Port 2, or Port 4 -- the cassette recorder.

The command has the basic forms:

COMMAND	PORT #	EXTERNAL DEVICE
VE1	Port 1	Terminal (i.e., terminal with tape driver)
VE	Port 2	Host (modem) - default port
VE2	Port 2	Host (modem)
VE4	Port 4	Audio cassette

A timeout feature is present for Port 2. If the host connected to Port 2 does not respond within approximately 10 seconds, the message TIMEOUT will be sent to Port 1 and the VE command will be aborted.

The optional [;=text] is used only with Port 2. The text following the "=" is sent to Port 2. In this manner, the Port 2 device knows what data to send for verification (in S-record format). When a mismatch is found between data in memory and the S-record object data, TUTOR will display the differences. If a display of differences occurs, the record following the record displayed may be lost. To prevent this, the ECB can send characters to the host to stop and start the transfer of the S-records. Paragraph 4.5.2 describes this feature.

Any record not containing an S0, S1, S2, S8, or S9 string is ignored. See Appendix A for information on S-records.

The offset contained in register R0 is added to the addresses of the data contained in each S-record.

See also: DU, LO, PF

EXAMPLE

```
TUTOR  1.X > VE ;=COPY TEST1.MX,#
S1131000-.-.49-.4E-.-.-.-.-.-.-.-.-.-.-.-.-.
ERROR

TUTOR   1.X >

TUTOR  1.X > VE4
S1132000-.-.-.-.-.-.-.-.-.-.21-.-.-.-.-.-.-.
ERROR

TUTOR  1.X >
```

COMMENT

Verify the file TEST1.MX. The record is an S1 record with $13 byte-count starting at address $001000, and there are differences in the third and fifth bytes.

Verify tape file.

TABLE 3-2. TUTOR Commands and Options

COMMAND	DESCRIPTION
BF <address1> <address2> <word>	Block of Memory Fill
BM <address1> <address2> <address3>	Block of Memory Move
BR [<address>[;<count>]]	Breakpoint Set
BS <address1> <address2> <data> [<mask>] [;<option>]	Block of Memory Search; options B, W, L
BT <address1> <address2>	Block of Memory Test
DC <expression>	Data Conversion
DF	Display Formatted Registers
DU[<port number>] <address1> <address2> [<text..>]	Dump Memory (S-records)
GD [<address>]	Go Direct
GO [<address>]	Go
GT <breakpoint address>	Go Until Breakpoint
HE	Help
LO[<port number>] [;<options>] [=text]	Load (S-records); options X, -C
MD[<port number>] <address1> [<count>] [;<option>]	Memory Display; option DI
MM <address> [;<options>]	Memory Modify; options W, L, O, V, N, DI
MS <address> <data...>	Memory Set
NOBR [<address> <address>....]	Breakpoint Remove
NOPA	Reset Printer Attach
OF	Display Offsets
PA	Printer Attach
PF[<port number>]	Port Format
.Rx	Individual Register Display/Change

TABLE 3-2. TUTOR Commands and Options (cont'd)

COMMAND	DESCRIPTION
TM [<exit character>]	Transparent Mode
TR [<count>]	Trace
TT <breakpoint address>	Temporary Breakpoint Trace
VE[<port number>] [=text]	Verify (S-records)
* text....	Send Message to Port 2 (1)
.A0 - .A7 [<expression>]	Display/Set Address Register (2)
.D0 - .D7 [<expression>]	Display/Set Data Register (2)
.R0 - .R6 [<expression>]	Display/Set Relative Offset Register (2)
.PC [<expression>]	Display/Set Program Counter (2)
.SR [<expression>]	Display/Set Status Register (2)
.SS [<expression>]	Display/Set Supervisor Stack Pointer (2)
.US [<expression>]	Display/Set User Stack Pointer (2)
(BREAK)	Abort command
(DEL)	Delete character
(CTRL D)	Redisplay line
(CTRL H)	Delete character
(CTRL W)	Suspend output (3)
(CTRL X)	Cancel command line
(CR)	Process command line

NOTES:

(1) See writeup of TM command.

(2) See writeup of .Rx command.

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

TABLE 3-3. 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
ILLEGAL INSTRUCTION	Instruction used an illegal op-code during program execution
ADDR TRAP ERROR BUS TRAP ERROR	See Traps in MC68000 User's Manual and paragraph 4.3.5.1.
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
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
FAILED AT.. WROTE=.. READ=..	Read or write command failure output by BT
UNDEFINED TRAP 14	Trap function code is not defined
CHKSUM=	Indicates received checksum is incorrect, correct checksum is given
OTHER MESSAGE	**MEANING**
TUTOR 1.X >	TUTOR prompt
TIMEOUT	Displayed if Port 2 does not respond to LO or VE within 10 seconds
FORMAT=	Displayed by PF command
CHAR NULL=	Displayed by PF command
C/R NULL=	Displayed by PF command

TABLE 3-3. Error Messages and Other Messages (cont'd)

ERROR MESSAGE	MEANING
OTHER MESSAGE	**MEANING**
OPTIONS@XXXXXX	Displayed by PF command
TRANSPARENT EXIT=$01=CTL A	Displayed by TM command
SOFTWARE ABORT	Displayed when abort button is pressed
BREAK	BREAK key has been used
AT BREAKPOINT	Indicates program has stopped at breakpoint
BREAKPOINTS	Displayed by BR command
PHYSICAL ADDRESS=	Actual address by command
PC within "DEBUGGER"	Displayed by trace commands