# B

# The ROM Debugger

This appendix documents the debug version of the ROM debugger. A new ROM debugger, RomBug, is also available. RomBug is documented in the ***OS-9 ROM Debugger's User's Manual***.

## The ROM Debugger

The ROM debugger is an optional part of the Professional OS-9 package. The ROM debugger is not a conventional program because you cannot invoke it from the command line. Assuming the ROM debugger is present and enabled, it is invoked in the following situations:

- When the machine is turned on and the UseDebug routine in the sysinit.a file returns the Zero flag of the CCR as false.
- When the abort signal (auto vector level 7) is encountered.
- When a bus error, address error, illegal instruction error, or trace exception is encountered.

## Overview of Debugger Functions

The ROM debugger loads and tests OS-9 and I/O drivers. The debugger's command set allows you to analyze programs by tracing, single instruction stepping, and breakpointing. It can disassemble instructions as the instructions are traced or stepped, or as a block. Commands can also examine or alter memory or CPU registers.

Depending on the type of debugger options selected, you can communicate with the host system as a terminal and download programs into RAM for testing via the communications link.

The debugger accepts command lines from the console. The command lines consist of a command code followed by a return key. Use the backspace (<control>H) and line delete (<control>X) keys to correct errors. If the system's ioxxx.a files resemble the standard Microware versions, you can use xoff (<control>S) and xon (<control>Q) to suspend and resume output, respectively.

## Expressions and Register Names

In some commands, the debugger can accept number expressions, shown as <num> or <len>, and address expressions, shown as <addr>. Both types of expressions have the same syntax. Expressions can be numbers or a combination of numbers, operators, and register names. Expressions are evaluated from left to right without priority, unless parentheses are used.

All numbers are assumed to be hexadecimal unless preceded with a pound sign character (#). For example, 200 is interpreted as a hex number but #200 is interpreted as a decimal number.

Register names consist of a period (.) followed by the usual assembly language name. For example, .a3 refers to register A3. Register names can be used freely in expressions.

The operators recognized by the debugger are:

| | | |
|---|---|---|
| One-Operand Operators: | - | negative |
| | ~ | bit-by-bit NOT |
| Two-Operand Operators: | + | add |
| | - | subtract |
| | * | multiply |
| | / | divide |
| | > | shift right; A>B shifts A right B bits |
| | < | shift left; A<B shifts A left B bits |
| | & | bit-by-bit AND |
| | \| | bit-by-bit OR |
| | ^ | bit-by-bit XOR (exclusive OR) |

The debugger has a special internal register called the ***relocation register***. This is used to add a constant offset to addresses. You can change the value of the relocation register at any time using the .r command.

**NOTE:** The display mode and change memory mode commands for the debugger are not affected by the relocation register offset.

It is often convenient to set the relocation register to the beginning physical address of a program code section or data area. Subsequent address commands can use the same offsets printed on assembler listings. Any address expression consisting of only a number (no operator) automatically has the relocation value added. The value of the relocation register can also be used in expressions or changed by referring to the register as ".r".

## The Debugger and Traps

Many of the debugger functions are implemented using traps. In particular, breakpoints cannot be used in ROM because breakpoints work by transparently replacing the existing instruction opcodes with opcodes that cause an “illegal instruction” trap. Obviously, instructions in ROM cannot be replaced in this manner.

The e command alternately enables and disables the debugger. When the debugger is enabled, it handles all address errors, bus errors, illegal instructions, and trace traps. The level 7 autovector trap is also reserved for use with an optional abort switch. Before the OS-9 kernel is started, traps not deliberately set by the debugger cause appropriate diagnostic messages to be displayed on the console.

When the debugger is disabled, all traps are passed to OS-9. To run the OS-9 user debug utility after the system is up, the ROM debugger must be disabled.

## Breakpoints and Caching

The debugger uses traps to set up breakpoints. Consequently, systems such as 68020 systems that contain instruction caches may have occasional problems with missed breakpoints. This occurs when a breakpoint is set at an instruction location that is currently cached. The debugger sets an “illegal instruction” trap in the code location specified, but the CPU executes the cached version of the instruction, causing the breakpoint to be missed.

To avoid this problem, Microware recommends that *all* cache resources for the system be disabled while using the ROM debugger, if possible.

## The Talk-Through Command

The second communications port on the target system is used for communications with the host system to provide download and talk-through functions. You can make versions of the debugger that omit the download or both download and talk-through functions in order to save ROM space.

Talk-through mode makes the debugger transparently pass data between the target system's terminal and the communications port to the host system. This effectively makes the target system terminal act as a host system terminal. The target system's terminal can be used to edit, assemble, etc., on the host system. This eliminates the need for two terminals. Use the following command to enter talk-through mode:

**tm <EscChar>**

This mode is exited when the specified escape character is typed.

Obviously, you should select the escape character carefully so it will not be the same as one used in normal communications with the host. Infrequently used characters such as the tilde (~) are recommended.

## The Download Command

The download command passes a command to the host system which causes it to send program data to the target system via the communications link. The program is loaded into RAM memory.

The program must be in the industry-standard Motorola S-record format. Only S1, S2, S3, S7, S8, and S9 record formats are recognized. The binex utility must be used to convert the OS-9 linker output from its normal binary format to S-record format. binex is a standard utility on professional OS-9 systems, licensed OS-9 distribution packages, and Port Paks. A Unix version is included in the distribution packages for VAX systems.

**NOTE:** Refer to the ***OS-9 Utilities*** section for more information on the binex utility.

The S-record format has data records that include a ***load address*** that specifies where to load the program in memory. OS-9 programs are position-independent, so the load address always starts at address zero. As S-records are received, the load addresses are added to the debugger's relocation register value to determine the actual address in RAM where the program is stored.

**NOTE:** You must download all program modules before OS-9 is executed for the first time. Otherwise, the modules will not be found by the search.

The relocation register to the area of RAM reserved for downloaded code in the boot.a special search table must be set. The two versions of the download command are:

| Name | Description |
|---|---|

| | |
|---|---|
| l <HostCmd> | Downloads data in Motorola S-record format. <HostCmd> is sent to the host as a command line to trigger the download. I/O delay must be set in register .d0 before the download. The load addresses are displayed every 512 bytes. |
| le <HostCmd> | Same as l <HostCmd> except received S-records are also displayed on the console instead of load addresses. |

The <HostCmd> sent to the host is the command required to dump the S-record file. For OS-9 hosts, the screen pause must be turned off using the tmode nopause command. A sample download command for an OS-9 host system is:

```
.r f1000 l binex objs/boot320
```

A sample download command for a Unix host system is:

```
l cat s.rec.file
```

The debugger transmits the command string to the host and then expects the host to begin transmitting S-records. The download ends when an S9 type record is received.

Sometimes the target system cannot keep up with a sustained high data rate when downloading. Therefore, the debugger sends xon and xoff to the host for flow control. If the host system does not respond immediately to xoff, you must set up a buffering delay count in register .d0 before using the download commands. A value of 20 works well in most cases with a data link running at 9600 baud, but you may have to experiment with this value as it is dependent on a combination of characteristics of the host system:

- xoff response lag time
- The target system CPU speed
- The baud rate

If the download command seems to hang up, a <control>E character aborts the download and also sends an abort signal to the host system. This may happen if the I/O buffer delay is not large enough or if the OS-9 host's screen pause is on.

Downloading using these commands should only be attempted after a hardware reset or after a debugger rst command. Otherwise, stack/data conflicts may occur within OS-9 and may produce strange results.

If you are debugging only one module, the module should be kept in a different file than the main OS-9 download file. This allows the main OS-9 code already in memory to be used and only the new version of the module will have to be downloaded. This will save a considerable amount of time. The rst command must be used first.

## Basic Debugger Commands

| Command | Description |
|---|---|
| b | Display addresses of all breakpoints. |
| b <addr> | Set breakpoint at <addr>. <addr> is relative to the default relocation register. |
| c <mod> <addr> | Enter change memory mode starting at <addr>. <mod> applies until change mode is exited. The default data length is one byte (8 bits). The <mod> options are:<br><br>w R/W words (16 bits)<br>l R/W long words (32 bits)<br>n No read for match or print<br>m No reread for match test<br>o R/W odd addresses, bytes only<br>v R/W even addresses, bytes only<br><br>Change mode commands are:<br><br><CR> Move to next location<br><num> Store new value, reread, verify match, move to next location<br>- Move to previous location<br>+ Move to next location<br>. Exit mode<br><br>Commands may be strung together. For example, the following command changes one location and then exits change mode:<br><br>**c .a5+3c FF .** |
| d <addr> [<len>] | Enter memory display mode beginning at <addr>. Contents of memory are displayed in hex and ASCII. If <len> is not specified, 256 bytes are displayed. Display mode commands are:<br><br><CR> Display next <len> bytes<br>. Exit mode<br>other Exit mode, interpret as command<br><br>**NOTE**: The display mode and change memory mode commands for the debugger are not affected by the relocation register offset. |

| Command | Description |
|---|---|
| di <addr> [<len>] | Disassemble and display <len> instructions beginning at <addr>. If <len> is not specified, 20 instructions are displayed. Disassemble mode commands are:<br><br><CR> Display next <len> bytes<br>. Exit mode<br>other Exit mode, interpret as command |
| e | Enable/disable debugger. |
| g | Execute program starting at <PC>. |
| g <addr> | Execute program starting at <addr>.<br><br>**NOTE:** If the program is stopped at a breakpoint, it is necessary to trace one instruction before using the g command. |
| k <addr> | Kill (remove) breakpoint located at <addr>. <addr> is relative to the default relocation register. |
| k* | Kill all breakpoints. |
| l<hostcmd> | Download data in Motorola S-record format. <hostcmd> is sent to the host as a command line to trigger the download. I/O delay must be set in .d0 before the download. The load addresses are displayed every 512 bytes. |
| le<hostcmd> | Same as l<hostcmd>, except received S-records are displayed instead of load addresses. |
| rst | Reset system; PC = Initial PC, SSP = Initial SSP, and SR = Supervisor state interrupts masked are set to level 7. This allows a g command to restart the system. |
| t <num> | Enter trace mode and trace <num> instructions. Trace mode commands are:<br><br><CR> Trace <num> more instructions<br>. Exit trace mode<br>other Exit trace mode, interpret as command |
| tm <EscChar> | Enter talk-through mode. This mode is exited when the specified escape character is typed. |

| Command | Description |
|---|---|
| . | Display all registers. |
| .<reg> <num> | Set register <reg> to value <num>. |
| .pc <addr> | Set program counter to <addr>. |

|  |  |
|---|---|
| .r <num> | Set relocation register to <num>. |

***End of Appendix B***