Power Hawk Series 600
Console Reference Manual

(Title formerly “Motorola Single Board Computer (SBC) Console Reference Manual”)

CONCURRENT COMPUTER CORPORATION

0830050-040
March 2001

CAUTIONARY NOTICE

While the manufacturer has attempted to detail in this manual all areas of possible danger to personnel in conjunction with the use of this equipment, personnel should use caution when installing, checking out, operating and servicing this equipment, especially when power is on. As with all electronic equipment, care should be taken to avoid electrical shock in all circuits where substantial currents or voltages may be present either through design or short circuit. Caution should be observed in hoisting equipment, especially regarding large structures during installation.

The Manufacturer is specifically not liable for any damage or injury, arising out of a worker's failure to follow the instructions contained in this manual, or in his failure to exercise due care and caution in the installation, operation, checkout and service of this equipment.

PROPRIETARY DATA

This document, the design contained herein, the detail and invention are considered proprietary to Concurrent Computer Corporation. As the property of Concurrent Computer Corporation it shall be used only for reference, contract or proposal work by this corporation, or for field repair of Concurrent products by Concurrent Computer Corporation service personnel, customers, or end users.

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Computer Corporation makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, FL 33069. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Printed in U. S. A.

Revision History:	Level:	Effective With:
Original Release - January 1996	000	Product Release
Previous Release - March 1999	030	MCP750 Support Release
Current Release - June 2001	040	Update to Series 600

Preface

Scope of Manual

This manual describes the console for Concurrent Computer Corporation's Power Hawk Series 600 systems and gives the information necessary to use it to debug the particular system. The following systems are covered in this manual:

System Platform	Motherboard Type	Number of Processors	Form Factor
Power Hawk 620	MVME2600	1	VME 6U
Power Hawk 640	MVME4600	2	VME 6U
PowerStack II/III	MTX	1 or 2	PC

Structure of Manual

This manual consists of a title page, this preface, a master table of contents, three chapters, three local tables of contents for the chapters, two appendixes, and an index. A brief description of the chapters and appendixes follows:

- Chapter 1 explains where the console fits in a system and describes the hardware of the console.
- Chapter 2 describes what occurs during system initialization and the console interface.
- Chapter 3 contains an alphabetical listing of the console debugging commands. Each command listing contains the purpose of the command, its syntax, an explanation of the command parameters, and examples of the command syntax and usage.
- Appendix A is a quick reference guide that lists the console commands and their meanings, as well as an explanation of the command parameters.
- Appendix B lists the possible error codes that may appear executing console commands. There is also a short description of the error and a possible cure to the problem.

The index has an alphabetical list of all paragraph formats, character formats, cross reference formats, table formats, and variables.

Syntax Notation

The following notation is used throughout this guide:

italic	Books, reference cards, and items that the user must specify appear in italic type. Special terms may also appear in italic.
bold	User input appears in bold type and must be entered exactly as shown. Names of directories, files, and commands also appear in bold type.
`list`	Operating system and program output such as prompts and messages and listings of files and programs appears in list type.
[]	Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such option or arguments.

Vendor Documentation

To access the manuals listed below, please visit the Motorola on-line documentation site at: "**www.mcg.mot.com/literature**".

Manual Name	Motorola Pubs No.	2600 Series	4600 Series	MTX Series
MVME2600 Series Single Board Computer Installation and Use	V2600A/IH	X	-	-
MVME4600 Series VME Processor Module Installation and Use	V4600A/IH	-	X	-
MTX Series Single Board Computer Installation and Use	MTXA/IH1	-	-	X
PPCBug Firmware Package User's Manual - Parts 1 & 2	PPCBUGA1/UM & PPCBUGA2/UM	-	X	X
MVME712M Transition Module and P2 Adapter Board Installation and Use Manual	MVME712MA/IH	X	X	-
MVME761Transition Module Installation and Use	VME761A/IH	X	X	-
PowerPC Open Firmware User's Manual -Volumes 1 & 2	PPCOFWA1/UM & PPCOFWA2/UM	X	-	-
MTX Series Single Board Computer Programmer's Reference Guide	MTXA/PG	-	-	X

Contents

1
Introduction to the Console

1
Introduction to the Console

Overview of Console

The console for the Series 600 system allows the operator to initialize the system and perform certain diagnostic procedures. An overview of this product is provided in the following paragraphs.

The console software package is included as a part of the system bootstrap process in order to provide the following:

- an operator's interface..
- knowledge of the PowerMAX OS™ file systems on the hard disks and tapes.

The Series 600 system systems normally begin execution in the internal **PPCBug** routine. **PPCBug** will then autoboot the Series 600 system console off of the appropriate boot media. The console is provided in a loadable format as the first file on all bootable media: diskettes, tapes and hard disk systems. The **PPCBug** internal ROM bootstrap code reads the console into memory, where it then relocates itself to higher memory locations and begins execution.

The exact mode of operation depends upon the operator action during the start-up. If the operator interrupts the boot sequence, a console prompt will be output and console commands may be used for debugging or system start-up as described later in this manual. If no keys are depressed system bootstrap is fully automatic and the PowerMAX OS kernel will be brought up to the multi-user system level specified in the **/etc/inittab** file.

Console Terminal Selection

The Series 600 system console code will operate over the internal serial port 1 on all systems.

When either of the internal serial ports are utilized for the console, any regular ASCII terminal with keyboard operating at 9600 baud (8-bits per character with no parity) may be connected. No specific terminal model is required. When these terminals are used as the console the native Series 600 system keyboard (if present) is disabled.

When more than one device is available, the Series 600 system console initialization code allows the operator to select the specific console device as described in the next chapter.

2
Startup

2
Startup

System Initialization

System initialization can be separated into three distinct areas: **PPCBug** Initialization, Console Initialization, and System Boot.

These occur as described in the following paragraphs.

PPCBug Initialization

The **PPCBug** package is the Series 600 system Self Test and Boot Loader package. It is installed in EPROM on the processor board. **PPCBug** is used to power-up and configure the processor.

PPCBug should normally be set to 'Autoboot' the Operating System (OS) off of the system disk. This is enabled via the **PPCBug** '**ENV'** command. Refer to the Series 600 system document *PPC1Bug Diagnostics Manual*, Publication Number PPC1DIAA/UM for further details. In this case the boot sequence will appear similar to the following:

```
PPC1 Debugger/Diagnostics Release Version 3.2 - 05/01/97 RM01
COLD Start

Local Memory Found =04000000 (&67108864)

MPU Clock Speed =200Mhz

BUS Clock Speed =67Mhz

WARNING: Keyboard Not Connected

Reset Vector Location   : ROM Bank A
Mezzanine Configuration : Single-MPU
Current 60X-Bus Master  : MPU0
Idle MPU(s)             : NONE

System Memory: 64MB, ECC Enabled (ECC-Memory Detected)
L2Cache:       256KB

SelfTest/Boots about to Begin... Press <BREAK> at anytime to Abort ALL

AutoBoot about to Begin... Press <ESC> to Bypass, <SPC> to Continue

Booting from: NCR53C825, Controller 0, Drive 0
Device Name : /pci@80000000/pci1000,3@c,0/harddisk@0,0
Loading: Operating System

IPL Loaded at: $03EF0000

Residual-Data Located at: $03F7F000
```

As shipped from the factory, 'Autoboot' option may not be enabled, in which case the program will stop with an input prompt similar to the following:

```
PPC1-Bug>
```

At this time the following command should be used to boot the console and operating system off of the appropriate boot media.

PPC1-Bug>pboot CLUN, **DLUN**

Where **CLUN** is the logical unit number of the controller supporting the boot device and **DLUN** is the logical unit number of the boot device.

The operator can acquire the values of **CLUN** and **DLUN** by using the **ioi** command as follows:

PPC1-Bug>ioi

```
I/O Inquiry Status:

CLUN  DLUN  CNTRL-TYPE  DADDR  DTYPE  RM   Inquiry-Data
0     0     NCR53C825   0      $00    N    SEAGATE  ST32550N         0019
0     10    NCR53C825   1      $00    N    SEAGATE  ST32550N         0019
0     20    NCR53C825   2      $00    N    SEAGATE  ST34572N         0784
0     50    NCR53C825   5      $01    Y    ARCHIVE  Python 28388-XXX 5.72
0     60    NCR53C825   6      $05    Y    PLEXTOR   CD-ROM   PX-40TW1.00
1     0     PC8477      0      $00    Y    <None>
```

Example - Boot from tape device:

Based upon the above inquiry status results, to boot from the ARCHIVE Python 28388 tape device on the NCR53C825 controller, the operator enters the following command:

PPC1-Bug>**pboot 0,50**

Example - Boot from CD-ROM device:

Based upon the above inquiry status results, to boot from the Plextor CD-ROM device on the NCR53C825 controller, the operator enters the following command:

PPC1-Bug>**pboot 0,60**

Console Initialization

When **PPCBug** loads the OS off of the boot media, it is actually loading the PowerMAX OS Console software package described in this document. Console initialization consists of determining the console device and selection of debug or system boot modes. **PPCBug** normally operates on an ASCII terminal connected to the COM1 port.

The PowerMAX OS Console startup sequence appears as follows:

```
PowerMAX OS Console (yymmdd-5.1)
Type '#' to cancel boot
```

At this time the console waits ten seconds in case the pound sign (**#**) is entered. An exclamation point (**!**) entered automatically starts the boot sequence without waiting the full ten seconds. If, however, the operator enters the pound sign (**#**) during those ten seconds, the system boot procedure is cancelled and the #> prompt is displayed. When the #>

prompt is displayed, any command described in Chapter 3 of this manual may be entered. Of particular importance is the **fb** command which causes the boot program to execute and load system programs, and the **pboot** register, which specifies the boot options.

System Boot

If the console system boot procedure has not been cancelled, either via entry of the pound sign (**#**) or the processor board jumper, the system boot mode is entered. The sequence is as follows:

```
PowerMAX OS Console (yymmdd-4.3)
Type '#' to cancel boot
Set Run Mode
CPU 0
..............
dsk(0,0,0,0)/.
Initialize VME
dsk(0,0,0,0)/boot
PowerMAX OS Boot Loader
Boot
:/stand/unix
2981244+602511+1801957 start 0x4000
symbol table loaded
```

Console Interface

After the boot program is loaded into memory, the processor begins executing code from memory (the boot program code). Control is passed back to the console during halt, reboot and breakpoint operations. This section describes how and when control passes from the console to a program in main memory, and from that program back to the console.

System Entry to Console

Any entry from a program to the console is performed via exceptions. These exceptions consist of breakpoint, trace, halt, and error. Upon entry to the console, the current context, system and user registers, and operating modes are saved and the `#>` prompt is output. Commands described in Chapter 3 of this manual may then be input. Control is also transferred to the console if the operator enters the sequence **<CR>~i** at the system console while the UNIX®[1] kernel is in operation.

Control may be returned to the executing program by entering the **r** (Run) command. Note that if File (**f**) commands are used, it is no longer possible to return to the operating system at the point it entered the console.

1. UNIX is a registered trademark of the Open Group.

Reset Button

All MVME x604 systems contain a **Reset** button on the front of the unit. Depressing this button causes a processor reset which will re-enter the **PPCBug** package described earlier. Note that this reset does NOT reset the devices on the VMEbus. The PowerMAX OS Console **<CR>~b** sequence may be used to reset the VMEbus and return to the **PPCBug** package.

Abort Button

The MVME x604 systems contain an **Abort** button on the front of the unit. This button is normally used to interrupt the executing program. If this button is depressed while the PowerMAX OS is running, the PowerMAX OS Console debug mode is entered as described in the following chapter. Entering a **r** command followed by a period (**.**) at this time returns control to the running system.

3
Console Debugging Commands

3
Console Debugging Commands

Summary of Commands

A summary of the console command set is shown in Table 3-1. The console supports many debugging tasks such as those listed below.

- Resetting and configuring the hardware.
- Examining and changing the contents of registers and memory.
- Setting and honoring breakpoints.

When the console is ready for a new command, the following prompt is displayed:

#>

The **Help** command symbol (**?**), if used as a command line entry (i.e.,**? <CR>**), displays the list of available commands that are shown in Table 3-1. If information is only needed for a specific command, that command symbol would be entered on the command line. For example, if the Examine/Change Memory command were entered following the **?** symbol (i.e. #>**?e <CR>**), the screen would only display information related to that specific command.

Syntax Conventions

The following conventions are used in the command syntaxes:

<a> – **a** is mandatory

[a] – **a** is optional

a|b – either **a** or **b** but not both. The **a** option or **b** option can be used with the command but the **a** option cannot be used along with the **b** option. Note that there may be a string of **OR** options (i.e., **a|b|c|d|e**) in this case you can only have one option, either **a** or **b** or **c** etc.

Table 3-1. Console Debugging Commands - Summary

Command	Definition	See Page No.
a	ASCII Dump	3-8
b	List Breakpoints	3-11
b	Set Breakpoints	3-12
bk	Clear Breakpoints	3-13
c	Copy Memory	3-14
d	Display Memory in Hexadecimal	3-16
di	Disassemble Memory	3-19
e	Examine/Change Memory	3-20
fb	Boot Operating System	3-22
fc	Display Directory	3-23
fd	Display/Set Default Device	3-24
fh	Display Mounted File Systems	3-26
fr	Load and Execute a Program	3-27
g	General Register Display/Modify	3-28
i	Initialize Memory to Value (Fill)	3-30
m	Memory Test	3-32
o	Global Command Options	3-33
p	Processor Register Display/Modify	3-34
qa	Query Address	3-37
qb	Query Backplane	3-38
qs	Query Stack	3-39
qv	Query Virtual Address	3-40
qy	Query Current Boot Options	3-41
r	Execute Run	3-42
ra	Execute Run to Address	3-43
rd	Run Without Breakpoints	3-44
rn	Run to Next Instruction	3-45
rr	Run to Return Address	3-46
s	Search Memory for Data	3-47
sr	Search Memory Range for Data	3-50
tc	Configure Console Device	3-51

Table 3-1. Console Debugging Commands - Summary (Cont.)

Command	Definition	See Page No.
td	Configure CPU Down (multiprocessor SBCs only)	3-53
tk	Configure Keyboard Device	3-54
tu	Configure CPU Up (multiprocessor SBCs only)	3-56
tv	Configure Video Device	3-57
w	Write Data to Memory	3-59
y	Initialize Boot Options/Flags	3-60
z	Single–Step Processor	3-61
?	Help Command	3-62

Some options are specified by a dash (**–**) followed by the option character. Command, options, and data must be entered in lower case. In this manual, parameters which must be entered are enclosed in < >. Optional parameters are enclosed in brackets **[]**. Optional parameters include such items as ending addresses for display commands. In general, the command syntax is shown below:

command –options start_address ending_address data

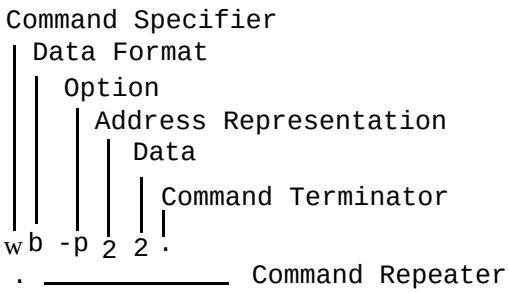
OR

command –options start_address:byte_count data

Most commands are terminated in one of two ways: by typing a period or by entering a carriage return **<CR>**. If a command is terminated via a period, the command executes immediately and then displays the prompt. If the command is terminated via a carriage return the command executes and then allows the use of one of the repeater commands. Repeater commands are discussed later in this chapter under the *Command Manipulators* heading.

Command Format

Although there is no format common to all the commands described in this chapter, most of the commands have one or more of the features listed in the sample command shown below.



Command Specifier

Table 3-1 briefly described each of the console debugging commands. These are the basic commands without their optional parameters.

Data Formats

The range of values for formatted data:

b Formatted as a byte – transfers data via eight–bit transfers,

w Formatted as a word (two bytes) – transfers data via 16–bit transfers, or

l Formatted as longwords (four bytes) – transferred in a single 32–bit transfer.

Global Options

Some commands have options, which are preceded by dashes. The following global options are available on most commands.

–b<n> Specify program base address. The base address, represented by the address value *n*, is added to all addresses entered from the command line. The address value *n* is zero by default. Numeric address values are discussed later in this chapter.

–r<n> Execute command <*n*> times (0 = infinite times). If this option is set the command executes <*n*> times. If <*n*> is zero, the command executes until aborted with a **CTRL–C**.

Numeric Values

A numeric value may be entered in any of the following formats.

'cccc' ASCII value.

hex digits Hexadecimal number.

$ Value of last entered address parameter.

% Contents of the program counter of the default processor.

%regname Contents of the specified processor register of the default processor.

BnBnBn Hex value of all the specified bits added together (e.g., **B2B0 = 5)**.

[\]symbol Console or program symbol (operating system or diagnostic) name. Leading backslash required when a symbol doesn't contain a leading underscore (_).

Address Value

An address may be entered in any of the following formats.

numeric value	Physical address by default. If the **o+v** option is set, and virtual memory is enabled then the address defaults to virtual; otherwise, the address is physical.
[numeric value]	A physical address is specified by enclosing a numeric value within square brackets.
(numeric value)	A virtual address is specified by enclosing a numeric value within parenthesis. The SDR0 and SDR1 registers for the processor contain the address of the translation tables.
***numeric value:size**	An indirect address is specified by placing an asterisk (*****) before a numeric value. Note that specifying indirection is valid only for memory reference options. The optional size parameter specifies the size of the indirect memory reference and must be in the range 1 through 4.
***[numeric value]:size**	Indirect physical address. The optional size parameter specifies the size of the indirect memory reference and it must be in the range 1 through 4.
***(numeric value):size**	Indirect virtual address. The optional size parameter specifies the size of the indirect memory reference and it must be in the range 1 through 4.

Command Manipulators

There are two categories of command manipulators: terminators and repeaters. Most commands can be terminated (exited) in one of two ways: by pressing the **<CR>** key or by typing a period (**.**).

If a command line is terminated by typing just a period, the command executes immediately and then the prompt is displayed, sometimes on the same line as the command results. Note that typing another period after the command has terminated causes that command to repeat.

If a command line is terminated by pressing the **<CR>** key, the command executes and then allows a repeat of the command (or a version of the command) via one of the following **repeaters.** (Note that not all repeaters are valid for all commands.).

–	When a dash (**–**) is used as a repeater the current data is displayed in ascii and as binary bits (e.g. **B26 B5 B0**). Note that this repeater is only valid for the **e**, **g**, and **p** commands.
<n><CR>	Change address to <*n*>.
@	Keep same address.
<SP>	Increment address to next page.
<CR>	Increment address to next line.

/ Decrement address to previous page.

. Repeat or exit current command.

Any command can be aborted by typing **CTRL–C**. This action causes a soft reset of the console. Any commands typed but not yet executed are ignored.

The following example shows the effect of the various command terminators.

```
 #>db 0:10<CR>
00000000 36 03 63 38 53 60 50 41 17 C0 FF EE D0 C0 02 37,
00000010 73 20 0C 0D EE FF 0C 71 14 05 06 35 83 36 30 63 /

00000000 36 03 63 38 53 60 50 41 17 C0 FF EE D0 C0 02 37 @
00000000 36 03 63 38 53 60 50 41 17 C0 FF EE D0 C0 02 37 <SP>
00000010 73 20 0C 0D EE FF 0C 71 14 05 06 35 83 36 30 63 <CR>
00000020 45 33 07 01 00 AC DC FE 98 48 42 43 16 41 44 FF 70<CR>
00000070 FF 44 14 61 34 24 84 89 EF CD CA 00 10 70 33 54.
 #>
```

Command Editing

Table 3-2 lists the character sequences that you may enter to edit the commands discussed in this chapter.

Table 3-2. Console Special Key Functions

Press	Function
DEL	Delete the last character typed.
CTRL–C	A soft reset to the console that aborts the command in progress.
CTRL–H	Delete the last character typed.
CTRL–O	Flush output characters until next ^O, error, or input prompt occurs.
CTRL–Q	XOFF – Stops Scrolling.
CTRL–P	Re-display the current input line.
CTRL–S	XON – Starts Scrolling.
CTRL–U	Delete entire input line.
<CR>~h	Halt the processor and enter console mode.
<CR>~i	Halt the processor. (This character sequence was added for compatibility purposes only and, for the PowerMAX OS computer systems, performs the same function as the **<CR>~h** sequence.)
<CR>~k	Enter PowerMAX OS kernel debugger.
<CR>~b	Reset system and reload console.

CAUTION

When in the user mode keystroke strings which are identical to the **<CR>~** commands are recognized by the system as console commands and, therefore, could cause adverse effects to other users on the system. (i.e., The keystroke sequence **<CR>~h** halts the system even though it appears in a user application.)

Console Commands

The remaining part of this chapter describes each of the console commands, with one or more examples of each command.

a **ASCII DUMP** **a**

Purpose: This command displays a portion of memory beginning at the specified location. The displayed data is in ASCII format and grouped by byte (**b**), word (**w**), or longword (**l**). This command has options (preceded by dashes) which are listed below. For a more detailed description of the options refer to the options paragraph in this chapter.

Syntax: **a[format][–b<n>][start_address [end_address]]**

a[format][–b<n>] [start_address [:byte_count]]

format Determines whether the data appears in byte, word, or longword [**b**, **w,** or **l**] format. Defaults to **b**.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The hexadecimal address at which the operation starts. The default value is 0.

end_address The hexadecimal address at which the operation ends.

:byte_count Number (in hexadecimal) of bytes displayed. The default is a page (256 bytes). Note that if you specify word format, **byte_count** should be a multiple of two. If you specify longword format, **byte_count** should be a multiple of four.

repeaters See the command manipulators paragraph for explanation.

Examples: The following are valid commands.

a B0 Displays a page of data starting at location **B0.**

ab 0 Displays a page of data starting at location **0.**

al 100:10. Displays the right–most byte of each of the four longwords of data starting at location 100. In other words, it displays the byte of data at memory locations 103, 107, 10B, and 10F.

aw 0. Displays right–most byte in each word starting at location 0.

a0. Displays from byte 0.

ab 0 10. Displays contents of addresses 0 through 10.

a **ASCII DUMP (Continued)** **a**

Sample ASCII dumps are shown below.

ASCII Dump by Byte with an Initial Address of 0

```
#>ab 0.
  00000000      H  C-  1  .  .  .  .  .  .  .  .  .  .  .  +  u
  00000010      .  .  .  .  .  .  .  .  .  .  .  T  [  .  .  .
  00000020      .  .  .     .  .  .  .  .  .  .  d  .  R  C  S  .
  00000030      .  .  .  0  .  .  .  .  .  .  .  .  .  2  .  o  .
  00000040      .  .  .  @  .  .  .  .  .  .  .  .  .  .  .  .  .
  00000050      .  .  .  P  .  .  .  .  .  .  .  .  .  .  .  +  .
  00000060      .  .  .  \  .  .  .  .  .  .  .  .  .  .  .  +  .
  00000070      .  .  .  p  .  .  .  .  .  .  .  .  .  .  .  +  .
  00000080      .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  +  .
  00000090      .  .  .  .  .  .  .  .  .  .  .  T  [  .  .  +  .
  000000A0      .  .  .  .  .  .  .  .  .  .  .  d  .  .  .  +  .
  000000B0      .  .  .  0  .  .  .  .  .  .  .  .  .  .  .  +  .
  000000C0      .  .  .  @  .  .  .  .  .  .  .  .  .  .  .  +  .
  000000D0      .  .  .  P  .  .  .  .  .  .  .  .  .  .  1  9  7
  000000E0      .  .  .  \  .  .  .  .  .  .  .  .  .  .  2  8  9
  000000F0      .  .  .  p  .  .  .  .  .  .  .  .  .  .  o  .  4
```

ASCII Dump of Right–Most Byte in Each Word — Initial Address of 0

```
#>aw 0.
 00000000       C  1  .  .  .  .  .  u
 00000010       .  .  .  .  .  [  .  .
 00000020       .     .  .  .  .  C  .
 00000030       .  0  .  .  .  .  .  .
 00000040       .  @  .  .  .  .  .  .
 00000050       .  P  .  .  .  .  .  .
 00000060       .  \  .  .  .  .  .  .
 00000070       .  p  .  .  .  .  .  .
 00000080       .  .  .  .  .  .  .  .
 00000090       .  .  .  .  .  [  .  .
 000000A0       .     .  .  .  .  .  .
 000000B0       .  0  .  .  .  .  .  .
 000000C0       .  @  .  .  .  .  .  .
 000000D0       .  P  .  .  .  .  1  7
 000000E0       .  \  .  .  .  .  2  9
 000000F0       .  p  .  .  .  .  o  4
```

a **ASCII DUMP (Continued)** **a**

ASCII Dump by Longword

```
 #>al 0.
00000000        1  .  .  u
00000010        .  .  [  .
00000020        .  .  .  .
00000030        0  .  .  .
00000040        @  .  .  .
00000050        P  .  .  .
00000060        \  .  .  .
00000070        p  .  .  .
00000080        .  .  .  .
00000090        .  .  [  .
000000A0        .  .  .  .
000000B0        0  .  .  .
000000C0        @  .  .  .
000000D0        P  .  .  7
000000E0        \  .  .  9
000000F0        p  .  .  4
```

ASCII Dump in Various Formats

```
#>ab 0 :10.
00000000      H  C  P  1  .  .  .  .  .  .  .  .  .  .  +  u
#>ab 0 :4.
00000000      H  C  P  1
#>aw 0 :4.
00000000      C  1
#>al 0 :4.
00000000      1
#>ab 0 f.
00000000      H  C  P  1  .  .  .  .  .  .  .  .  .  .  +  u
```

b **LIST BREAKPOINTS** **b**

Purpose: This command lists breakpoints for all of the processors.

Function: Some of the breakpoint commands have options (preceded by dashes) which are listed below. For a more detailed description of the options refer to the options paragraph in this chapter. Up to eight breakpoint entries are kept in an internal break address table.

Syntax: **b**

A sample list breakpoint command is shown below.

```
#>b.
00 00001000 CPU physical
01 00002000 CPU physical
```

b **SET BREAKPOINTS** **b**

Purpose: This command sets breakpoints for all or one of the processors.

Function: When the processor hits a breakpoint, the console removes the breakpoints from memory before accepting any commands. Some of the breakpoint commands have options (preceded by dashes) which are listed below. For a more detailed description of the options refer to the options paragraph in this chapter.

Up to eight breakpoint entries are kept in an internal break address table. Overflow of the break address table generates an error message. When a program begins executing the system enters the breakpoints into the code.

Syntax: **b [–a][–o][–b<n>]<address>**

–a Immediately inserts all breakpoints into memory.

–o Breakpoint is temporary. Temporary breakpoints are removed once they are hit.

–b<n> Specify program base address. The base address, represented by the address value *n*, is added to all addresses entered from the command line. The address value *n* is zero by default. Numeric address values are discussed later in this chapter.

address The address to which a breakpoint is assigned. If you want to get a breakpoint at a processor address enter that particular address after the **b** command. If the **address** is already defined, an error message appears on the screen. If the address cannot be written, an error is generated.

Examples: The following are valid commands.

b1000. Set breakpoint at 0x1000

b2000. Set breakpoint at 0x2000

b.or b<CR> Displays breakpoint

Sample set breakpoint commands are shown below.

Set Breakpoints at Memory Locations 1000 and 2000

```
 #>b1000.  #>b2000.
```

Set Duplicate Breakpoint

```
#>b BFF00000.

#>b BFF00000.

error 0007: duplicate breakpoint
```

bk **CLEAR BREAKPOINTS** **bk**

Purpose: This command clears (removes) breakpoints for all of the processors.

Function: Some of the breakpoint commands have options (preceded by dashes) which are listed below. For a more detailed description of the options refer to the options paragraph in this chapter.

Syntax: **bk <address>|<all>**

address The address to which a breakpoint is assigned. If you want to clear a breakpoint at a processor address enter that particular address after the **bk** command.

Examples: The following are valid commands.

bk1000. Remove breakpoint at 0x1000.

bk all Removes all breakpoints.

A sample breakpoint command is shown below.

Remove Breakpoint at Memory Location 1000.

#>**bk1000.**

#>**b.**

```
01 00002000 CPU physical
```

c **COPY MEMORY** **c**

Purpose: This command moves the data located at the **source_start_address** through **source_end_address** (inclusive) to the locations starting at the **destination_start_address**. This command also moves the data located at the **source_start_address** to the locations starting at the **destination_start_address** for the number of bytes specified in the **byte_count**. This command has options (preceded by dashes) which are listed below. For a more detailed description of the options refer to the options paragraph in this chapter.

Note: When virtual addressing is used, translation is performed in 'data' space.

Syntax: **c[format][–b<n>]<source_start_address> <source_end_address> <destination_start_address>**

c[format][–b<n>] <source_start_address><:byte_count> <destination_start_address>

format Determines whether the data is to be copied in byte, word, or longword [**b**, **w**, or **l**] format. The default format is longword.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

source_start_address
This is the address at which the memory to be copied starts.

source_end_address
This is the address at which the memory to be copied stops.

destination_start_address
This is the destination address.

:byte_count Number (in hexadecimal) of bytes copied. Note that if you specify word format, **byte_count** should be a multiple of two. If you specify longword format, **byte_count** should be a multiple of four.

Examples: The following are valid commands.

c B0 C0 D0 Moves data at locations **B0** through **C0** to location **D0**.

cb 0 C0 D0 Moves values at locations **0** through **C0** to location **D0**.

c **COPY MEMORY(Continued)** **c**

Sample copy commands are shown below.

Find out what values are at location 1000.

```
#>d1000 :10.
00001000 ABFF0000 44700004 3322000F F000000F
```

Move values byte by byte between 0 and 400 to 1000.

```
#>cb0 400 1000. #> d1000:10.
00001000 00000000 00000004 00000008 0000000C
```

Move values word by word between 1000 and 1400 to 2000.

```
#>cw1000 1400 2000. #> d2000:10.
00002000 00000000 00000004 00000008 0000000C

#>cw2000:400 3000. #>d3000:10.
00003000 00000000 00000004 00000008 0000000C
```

d **DISPLAY MEMORY IN HEXADECIMAL** **d**

Purpose: This command displays a portion of memory beginning at the specified location. The displayed data is in hexadecimal format and grouped by byte, word, or longword.

Note: When virtual addressing is used, translation is performed in 'data' space.

Syntax: **d[format][–b<n>][start_address [end_address]]**

d[format][–b<n>][start_address[:byte_count]]

format Determines whether the data is displayed in byte, word, or longword [**b**, **w**, or **l**] format. The default value is **w** in console mode and **l** in processor mode (**o+p**).

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The hexadecimal address at which the operation starts. The default value is the last s**tart_address** specified.

end_address The hexadecimal address at which the operation ends.

:byte_count Number of bytes displayed. The default is a page (256 bytes). Note that if you specify word format, **byte_count** should be a multiple of two. If you specify longword format, **byte_count** should be a multiple of four.

repeaters See the command manipulators paragraph for explanation.

Examples: The following are valid commands.

d Displays a page of data starting at location 0 in longword format (assumes processor mode).

d B0 Displays a page of data starting at location **B0.**

db 0 Displays a page of data starting at location **0** in byte format.

dw 0 4 Displays the first three words.

d DISPLAY MEMORY IN HEXADECIMAL (Continued) d

A sample of a hexadecimal memory display is shown below.

HEXADECIMAL DISPLAY BY BYTE STARTING AT ADDRESS 0

```
#>db 0.
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AB 75
00000010 00 00 00 10 00 00 00 00 1B 9D D4 5B 00 0C 00 02
00000020 00 00 00 20 00 00 00 00 1B 9D E4 0C 52 43 53 00
00000030 00 00 00 30 00 00 00 00 00 00 00 00 32 2E 6F 00
00000040 00 00 00 40 00 00 00 00 00 00 00 00 00 14 00 08
00000050 00 00 00 50 00 00 00 00 00 00 00 00 00 00 AB 8D
00000060 00 00 00 60 00 00 00 00 00 00 00 00 00 00 AB 8E
00000070 00 00 00 70 00 00 00 00 00 00 00 00 00 00 AB 8F
00000080 00 00 00 80 00 00 00 00 00 00 00 00 00 00 AB 90
00000090 00 00 00 90 00 00 00 00 1B 9D D4 5B 00 00 AB 84
000000A0 00 00 00 A0 00 00 00 00 1B 9D E4 0C 00 00 AB 92
000000B0 00 00 00 B0 00 00 00 00 00 00 00 00 00 00 AB 93
000000C0 00 00 00 C0 00 00 00 00 00 00 00 00 00 00 AB 94
000000D0 00 00 00 D0 00 00 00 00 00 00 00 00 00 31 39 37
000000E0 00 00 00 E0 00 00 00 00 00 00 00 00 00 32 38 39
000000F0 00 00 00 F0 00 00 00 00 00 00 00 00 2E 6F 00 34
```

HEXADECIMAL DISPLAY BY WORD STARTING AT ADDRESS 1000

```
#>dw 1000.
00001000 0000 0000 0000 0000 0000 0000 0000 0000
00001010 0000 0000 0000 0000 0000 0000 1B9E 0E4C
00001020 0000 0000 0000 0000 0000 0000 1B9E 0E4C
00001030 0000 0000 0000 0000 0000 0000 0000 0000
00001040 0000 0000 0000 0000 0000 0000 0000 0000
00001050 0000 0000 0000 0000 0000 0000 0000 0000
00001060 0000 0000 0000 0000 0000 0000 0000 0000
00001070 0000 0000 0000 0000 0000 0000 0000 0000
00001080 0000 0000 0000 0000 0000 0000 0000 0000
00001090 0000 0000 0000 0000 0000 0000 1B9D CEBB
000010A0 0000 0000 0000 0000 0000 0000 1B9E 0E4B
000010B0 0000 0000 0000 0000 0000 0000 0000 0000
000010C0 0000 0000 0000 0000 0000 0000 0000 0000
000010D0 0000 0000 0000 0000 0000 0000 0000 0000
000010E0 0000 0000 0000 0000 0000 0000 0000 0000
000010F0 0000 0000 0000 0000 0000 0000 0000 0000
```

d DISPLAY MEMORY IN HEXADECIMAL (Continued) d

HEX DISPLAY STARTING AT ADDRESS 1000 — NO DATA SIZE SPECIFIED

#>**d1000.** <——— *Defaults to longword*

```
00001000 00000000 00000000 00000000 00000000
00001010 00000000 00000000 00000000 1B9E0E4C
00001020 00000000 00000000 00000000 1B9E0E4C
00001030 00000000 00000000 00000000 00000000
00001040 00000000 00000000 00000000 00000000
00001050 00000000 00000000 00000000 00000000
00001060 00000000 00000000 00000000 00000000
00001070 00000000 00000000 00000000 00000000
00001080 00000000 00000000 00000000 00000000
00001090 00000000 00000000 00000000 1B9DCEBB
000010A0 00000000 00000000 00000000 1B9E0E4B
000010B0 00000000 00000000 00000000 00000000
000010C0 00000000 00000000 00000000 00000000
000010D0 00000000 00000000 00000000 00000000
000010E0 00000000 00000000 00000000 00000000
000010F0 00000000 00000000 00000000 00000000
```

COMPARE ASCII DISPLAY TO HEXADECIMAL DISPLAY

#>**wl 0 48 43 50 31.** #>**ab 0 :10.** <— *Write hexadecimal data to memory.*

```
00000000 .  .  .  H  .  .  .  C  .  .  .  P  .  .  .  1
#>al 0 :10.
00000000 H  C  P  1
#>db 0 :10.
00000000 00 00 00 48 00 00 00 43 00 00 00 50 00 00 00 31
```

di **DISASSEMBLE MEMORY** **di**

Purpose: This command disassembles instructions beginning at the specified address. Note that when virtual addressing is used, translation is performed in 'instruction' space.

Syntax: **di[–b<n>][start_address[:byte_count]]**

di[–b<n>][start_address [end_address]]

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The hexadecimal address at which the operation starts. The default value is the last **start_address** specified.

end_address The hexadecimal address at which the operation ends.

:byte_count Number of bytes displayed. The default is 16 longwords (64 bytes).

Sample disassembly commands are shown below. Note that the symbol table must be loaded (bit 7 of register pboot i.e., :#>pboot 80.) to obtain the symbols at boot time.

#>di %pc–10.

```
VirtualPhysicalLabel andOp CodesInstructions
AddressAddressOffset


000187b4 [000187b4] halt+34          70c31010 andi. r3,r6,1010
000187b8 [000187b8] halt+38          7c600124 mtmsr r3
000187bc [000187bc] halt+3c          4c00012c isync
000187c0 [000187c0] halt+40          4ea00421 bctrl
000187c4 [000187c4] halt+44        % 7cc00124 mtmsr r6
000187c8 [000187c8] halt+48          4c00012c isync
000187cc [000187cc] halt+4c          7ca803a6 mtlr r5
000187d0 [000187d0] halt+50          4ea00020 blr
000187d4 [000187d4] halt+54          48000004 b halt+0x58
000187d8 [000187d8] halt+58          48000004 b halt+0x5c
000187dc [000187dc] halt+5c          48000004 b consbkpt
000187e0 [000187e0] consbkpt         80801ff0 lwz r4,0x1ff0(r0)
000187e4 [000187e4] consbkpt+4       2c040000 cmpwi crf0,r4,0
000187e8 [000187e8] consbkpt+8       4082000c bne- crf0,consbkpt+0x14
000187ec [000187ec] consbkpt+c       38600001 li r3,1
000187f0 [000187f0] consbkpt+10      4ea00020 blr
```

Note that% implies the program counter of the default processor and * implies break.

e **EXAMINE/CHANGE MEMORY** **e**

Purpose: This command displays a byte, word, or longword of memory beginning at the specified memory location. This command can also change the data at that location and subsequent locations via the data specified. The format of the data written is controlled by the format and command options specified.

Syntax: **e[format][–b<n>][start_address[data]]**

format Determines whether the data is displayed in byte, word, or longword [**b**, **w**, or **l**] format. The default value is **w** in console mode and **l** in processor mode.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The hexadecimal address at which the operation starts. The default value is 0.

data The new value to be entered at **start_address**.

repeaters See the command manipulators paragraph for explanation.

Examples: The following are valid commands.

e B0. Displays a longword of data starting at location **B0**.

eb 0. Displays the byte of data at location **0**.

ew0 5. Displays the current word of data at location 0 and then changes the contents to 5.

Sample examine and change commands are shown below.

EXAMINE ONE BYTE AT ADDRESS 0

```
#>eb 0.
00000000 00
```

EXAMINE ONE WORD AT ADDRESS 0

```
#>ew 0.
00000000 0000
```

EXAMINE MEMORY STARTING AT ADDRESS 0 – NO DATA SIZE SPECIFIED

```
#>e0 <CR><—— Defaults to longword.
00000000 00000000,<—— The comma shows next longword.
00000004 AB0007FF.<—— The period terminates command.
```

e **EXAMINE/CHANGE MEMORY (Continued)** **e**

EXAMINE WORD WITH VIRTUAL ADDRESS SPECIFIED

#>**e (BFFF8000) <CR>**
BFFF800 [00018000] 00000000.

DEPOSIT A LONGWORD IN MEMORY AND VERIFY THAT THE VALUE WAS STORED

#>**e10 <CR>**
00000010 7FFAB001 **50** <—— *The user enters 50 and the console writes 50 to location 10 and verifies that the value is actually stored at 10.*
00000014 00000432
#>**e10 <CR>**
00000010 00000050.<—— *The user enters a period.*

DEPOSIT A WORD OF DATA WITH THE BYTE PARAMETER

#>eb 10 <CR>
00000010 00 **123@**<—— *The console displays an error message*
error 0009: memory doesn't match
0000010 **23.**<—— *The period terminates the command.*

fb **BOOT OPERATING SYSTEM** **fb**

Purpose: This command boots the operating system (OS) of the computer. The contents of processor boot register determine how the system boots. Table 3-3 lists the possible values for **pboot** and the effect of those values on the boot process.

Note: The values can be added together. For example, a value of 1 in **pboot** causes the system to prompt for the boot OS file, a value of 2 automatically boots the user into single–user mode, and a value of 3 boots into single–user mode and prompts for the boot file. To change the value of processor boot register, use the **p** command.

Table 3-3. Effect of pboot on Boot Process

pboot Value (Hex)	Effected File	Effect on Boot Process
0		Boots automatically without option.
1	/boot	Requests file name for boot. Asks user to specify the program to load.
2	unix	Boots /OS to single–user mode.
4	unix	Do not synchronize before reboot.
8	unix	Do not reboot, just halt
20	unix	Reset system console terminal
80	unix	Debug option (load symbol table).
100	unix	Load and then halt twice, once before enabling Virtual Memory and once after enabling Virtual Memory.
400	unix	Load then halt in kernel debugger
800	unix	Do not initialize kernel debugger.

Syntax: **fb [-c<*n*>][-q]**

-c Specifies the processor <***n***> on which the **fb** command is to run on. If none specified, defaults to processor 0.

-q Option **-q** (quick). If set, only one attempt is made to find the **/boot** or specified file system.

A sample system boot listing is shown below.

```
 #>fb <CR>

 dsk(3,0,0,0)/.
Initialize VME
 dsk(3,0,0,0)/boot

 Boot
: unix
 747336+61360+597388 start 0x4000
```

fc **DISPLAY DIRECTORY** **fc**

Purpose: This command lists the contents of the specified directory.

Note: Never append a period to this command. After the command, you must press **<CR>.** (Periods are valid syntax in pathnames.)

Syntax: **fc [dir_name]**

dir_name A directory name. If the **m** option of the **o** command is set (**o+m**) you must provide the device on which the directory is located.

Examples: The following are valid commands.

fc/usr/d <CR>

fc/ <CR>

A sample root directory listing is shown below.

```
#>fc / <CR>
.
.               lost+found      dev             bin
usr             etc             hdcwcs          boot
stand_ls        stand_cat       unix            profile
tmp             unixPREV        unixBACKUP      fastboot
```

fd **DISPLAY/SET THE DEFAULT DEVICE** **fd**

Purpose: This command sets or displays the default device.

Note: Never append a period to this command. After the command, you must press a **<CR>.** (Periods are valid syntax in parameters.)

Syntax: **fd [-l][-s][dev]**

-l List the logical device table. If this option is entered, do not enter **dev**. This option displays all of the available boot devices (tapes, and disks) along with the logical device numbers. See examples below.

-s Saves the device specified in NVRAM as the default boot device. On subsequent boots, if the system is allowed to boot automatically, or the operator does **not** enter an **fd** command, the device saved in NVRAM is used. Any entry of **fd** will cause the default to be **not** used until the next boot. (See examples using the **-s** option on the next page.)

dev The device that is to be chosen as the default device. Two formats are available depending upon the number of fields in **dev.** The two field version is either **dsk(d,p)** or **mt(d,p)** where **d** is a logical device number and p is the partition number (0 through 6). Logical device numbers always run from 0 to n and correspond to the available boot devices found by the system during a search of all available SCSI controllers. The table of logical device numbers may be displayed via the **-l** option. The second format for **dev** provides an absolute hardware address and is input as **dsk(c,u,p,b)** or **mt(c,u,p,b)** where **c** is the controller number within the particular bus, **u** is the drive ID, **p** is the partition number (0 through 6) and **b** is the bus number (0 is normally the internal PCI bus, 1 is the VME bus and 2 is the IDE bus). To use the absolute mode, all four fields must always be entered. If you do not specify a device, the console assumes the two field version and selects **dsk(0,0).**

Examples: The following are valid commands.

fd mt(0) Set the default device to the first tape device found on the various SCSI bus controllers in the system. Partition zero is selected by default.

fd dsk(0,0,0,0) Set the default device to the disk on SCSI ID 0 of the internal PCI bus SCSI controller.

fd dsk(0,0,2,0) Set default to ‘usr’ partition.

`fd -l` List the available devices and logical device numbers:

```
>fd -l
.............
fd              disk                                    tape
0   (0,0,x,0) SEAGATE ST15150N
1   (0,3,x,0) SEAGATE ST19171W
2   (0,4,x,0) SEAGATE ST34573W
3   (0,6,x,0) PLEXTOR CD-ROM PX-40TW
```

Note: CD-ROM while listed, is not a boot device.

Examples using the `-s` option:

The following are valid commands using the **`-s`** option.

`fd -s` with no parameters specified, clears the default device

`fd -s dsk(1)` causes the second disk listed under **`fd-l`** to be used as the default boot device on subsequent system boots.

Example of changing the default disk from drive 0 to drive 1:

```
#0>fd
.............................
dsk(0,0,0,0)
#0>fd -s dsk(1)
Update NVRAM (Y/N) ? y|
NVRAM updated
```

Example of reverting back to drive 0 (default):

```
#0>fd -s
Clearing default boot device.
Update NVRAM (Y/N) ? y
NVRAM updated.
```

fh **DISPLAY MOUNTED FILE SYSTEMS** **fh**

Purpose: This command gives the default input device.

Note: Never append a period to this command. After the command, you must press **<CR>.** (Periods are valid syntax in parameters.)

Syntax: **fh**

A sample display from the **fh** command is shown below.

```
#>fh <CR>
Default:        dsk (5,0,0,0)
```

fr **LOAD AND EXECUTE A PROGRAM** **fr**

Purpose: This command loads and executes a program. The command can be followed by an optional list of arguments that are to be passed to the program.

Note: Never append a period to this command. After the command, you must press **<CR>.** (Periods are valid syntax in parameters.)

Syntax: **fr<spec>[base] [-c<*n*>]**

spec A file specification, in the following format: **[dev] pathname**. This file contains the file to be loaded. If the **m** option of the **o** command is set (**o+m**), you must provide the device on which the directory is located via a **fd** command or by specifying **dsk/**.

base The address into which the program is loaded. Programs are loaded and run at 0x2000 as a default. If, however, you specify a **base,** the load address and start address are set to the **base** value.

-c Specifies the processor <***n***> on which the **fr** command is to run on. If none specified, defaults to processor 0.

Example: A sample load the boot program and boot the system sequence is shown below.

#>**fr/boot** <**CR**>

```
 Boot
: /stand/unix
Target 0 found: IBM      0661371
3198180+606544+1805440 start 0x2000
symbol table loaded

PowerMAX OS Release 4.3
```

g **GENERAL REGISTER DISPLAY/MODIFY** **g**

Purpose: This command displays and/or modifies the contents of the 40 general–purpose registers of the default processor as shown in Table 3-4. If no parameters are specified, this command displays all of the general purpose registers (e.g. pc, r0 through r3, etc.). If a register name with no **data** parameter is specified, the contents of that specific register is displayed. If the **data** parameter is included, the console changes the value in the register. Subsequent registers can be modified by specifying new data for that particular register. To display the registers, the processor must be halted. After the processor is halted, the data displayed is that obtained at the last processor halt.

Note: This command is similar to the **p** command.

Syntax: **g <register_name >[data]**

Table 3-4. General–Purpose Registers

REGISTER N	ACRONYM	TYPE
Program Counter	pc	R/W
Machine Status Register	msr	R/W
Register 0	r0	R/W
Register 1	r1	R/W
.	.	.
.	.	,
.	.	.
Register 31	r31	R/W
Condition Register	cr	R/W
Link register	lr	R/W
Count Register	ctr	R/W
Extension Register	xer	R/W
System Processor Level	spl	R/W

register_name The general–purpose register to be examined or changed.

data The new hexadecimal value to be entered at **register_name.**

repeaters See the command manipulators paragraph for explanation.

g GENERAL REGISTER DISPLAY/MODIFY (Continued) g

Examples: The following are valid commands.

gpc. Display contents of the program counter.

gr1. Display contents of data register r1.

g. Displays contents of all general registers.

An examine all general register values example is shown below.

```
#>g
pc  = 000187C4    msr = 00001010     cr = 48800000    spl = 00000061
r0  = 0000F084     r1 = FFD040B8     r2 = 00000000     r3 = 00001010
r4  = 00002000     r5 = 0021E554     r6 = 00009032     r7 = C22DE4F5
r8  = 00000001     r9 = 00000001    r10 = 002F2D19    r11 = 002F2D19
r12 = C22DE475    r13 = 0021E140    r14 = 002F2D19    r15 = 00000069
r16 = 003EFE48    r17 = 00000069    r18 = 00000069    r19 = 00000000
r20 = 00000001    r21 = FFD041FA    r22 = 00000001    r23 = 00000061
24  = 0000006C    r25 = 00009032    r26 = 00000000    r27 = 00000094
r28 = DEADBEEF    r29 = 00000020    r30 = 81818181    r31 = 0BADC0DE
lr  = 000187C4    ctr = 01FC7ED8    xer = 00000004     mq = 00000000
```

An examine and change register values example is shown below.

```
#>gr1 <CR>       <——Displays contents of register r1.
r1 = C002F7F2,   <——Entering the comma displays r2.
r2 = 00000000,   <——Entering the comma displays r3.
r3 = 81A40001    <——Entering the period finishes command.
#>gr2 23.81A40001<——Change contents of r2 to 23.
#>gr1 <CR>       <——Display register r1.
r1 = C002F7F2,
r2 = 00000000,
r3 = 00000023.
```

i INITIALIZE MEMORY TO VALUE (FILL) i

Purpose: This command writes the **data** into all locations between the **start_address** and **end_address.** The format of the data is controlled via the options entered. Memory addresses before 0x6000 are used by the console and should not be initialized.

Note: When virtual addressing is used, translation is performed in 'data' space.

Syntax: **i[format][-q][-b<n>]<start_address><end_address> [fill_value]**

i[format][-q][-b<n>]<start_address>:<byte_count> [fill_value]

format	Determines whether the data is displayed in byte, word, or long-word [**b**, **w**, or **l**] format. The default value is **w** in console mode and **l** in processor mode.
-q	Error checking is suspended for this command, resulting in faster command execution.
-b<n>	Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).
start_address	The address at which the loading of memory starts. This address may not be a virtual address.
end_address	The address at which the loading of memory ends. If the **end_address** is not supplied or is a location before the **start_address**, you get a syntax error. This address may not be a virtual address.
:byte_count	Number (in hexadecimal) of bytes initialized. Note that if you specify word format, **byte_count** should be a multiple of two. If you specify longword format, **byte_count** should be a multiple of four.
fill_value	The hexadecimal word that is loaded into each memory location. The fill value defaults to zero.

Examples: The following are valid commands.

i10 20 10101010.	Loads each longword from 10 to 20 with the hexadecimal word 10101010.
ib10 20 F.	Loads each byte from 10 to 20 with the value F.

i INITIALIZE MEMORY TO VALUE (FILL) (Continued) i

Sample memory initialization procedures are shown below.

INITIALIZE MEMORY BETWEEN ADDRESSES 1000 AND 2000

```
#>i1000 2000 10101010. #0>d1000 :10.
```
<— Defaults to longword hexadecimal display starting at address 1000.

```
00001000 10101010 10101010 10101010 10101010
```

INITIALIZE MEMORY BETWEEN ADDRESSES 1000 AND 2000 (LOAD A WORD)

```
#>iw1000 2000 ff. #>d1000:10.
```

```
00001000 00FF00FF 00FF00FF 00FF00FF 00FF00FF
```

INITIALIZE MEMORY BETWEEN ADDRESSES 10 AND 20 (LOAD A BYTE)

```
#>ib10 20 f. #>d10:10.
```

```
00000010 0F0F0F0F 0F0F0F0F 0F0F0F0F 0F0F0F0F
```

m **MEMORY TEST** **m**

Purpose: This command performs a combination of tests (ones, zeroes, and unique address) that check memory between the **start_address** and **end_address.** Any errors which occur appear listed on the console screen.

Note: When virtual addressing is used, translation is performed in 'data' space.

Syntax: **m[format][-b<n>]]<start_address><end_address>**

m[format][-b<n>]<start_address>:<byte_count>

format: Determines whether the data is displayed in byte, word, or longword [**b, w,** or **l**] format.

-b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The first address tested. This address may not be a virtual address.

end_address The last address tested. This address may not be a virtual address.

:byte_count Number (in hexadecimal) of bytes tested. Note that if you specify word format, **byte_count** should be a multiple of two. If you specify longword format, **byte_count** should be a multiple of four.

Example: The following is a valid command.

m1000 2000.

o GLOBAL COMMAND OPTIONS o

Purpose: This command sets conditions under which the console operates. These conditions are stored as options in an options word.

Syntax: **o [+|–][m][r][v][–b<n>]**

+ or – A plus (**+**) adds and a minus (**–**) removes the specified options from the options word, effectively enabling or disabling that option. If you do not use a plus or minus, the command sets the options word to the options specified. The options are the conditions under which the console operates. If you do not specify any options, the console displays the current options.

m Disables automatic translation and 'mount' of directory names to the corresponding file system devices. These devices allow system files to be available from the console across all system disks by system–wide pathnames.

r Enable Reset Button. Front panel reset button which returns control to the console. If disabled, front panel reset button resets machine.

v Defaults to virtual addresses whenever virtual memory is enabled. Brackets or parentheses may be used to override the default address mode.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

Examples: The following are valid commands.

o. Display options that are set.

o+r Add r option.

o–vm Disable v and m options.

Sample set and remove options commands are shown below.

#>**o.** <——*Display current options*.
```
mr -b 0
```

#>**o+v.**<—— *Add v option.*
```
o. mrv -b 0
```

p PROCESSOR REGISTER DISPLAY/MODIFY p

Purpose: This command displays and changes the contents of the 44 processor registers shown in Table 3-5. If no parameters are specified, this command displays all of the processor registers. If a register name with no data is given, the contents of the specified register is displayed. If you specify the **data** parameter, the console changes the value in the register to the value specified. Processor registers and their attributes were discussed under the processor registers paragraph in this manual. If the target processor is running when this command is issued, the console momentarily halts the processor. This action provides a "snapshot" of the processor register values.

Note: This command is similar to the **g** command.

Syntax: **p<register_name> [data]]**

register_name The processor register to be examined or changed. The value of the **register_name** is the symbolic register name.

data The hexadecimal data to be placed in the processor register.

repeaters See the command manipulators paragraph for explanation.

Examples: The following are valid commands.

p. Displays all processor registers.

pboot Displays contents of processor boot register For more information on the processor register refer to the **fb** (Boot Operating System) command.

dar. Displays contents of the Data Address Register.

An examine all processor register values example is shown below.

```
#>p.

 dsisr = 0A000000    dar = 3003B288   sdr1 = 01E0000F  fpscr = 000000D0
   sr0 = 20000000    sr1 = 20DE2989    sr2 = 20DE298A    sr3 = 20DE298B
   sr4 = 20DE298C    sr5 = 20DE298D    sr6 = 20DE298E    sr7 = 20DE298F
   sr8 = 20DE2990    sr9 = 20DE2991   sr10 = 20DE2992   sr11 = 20DE2993
  sr12 = 20DE2994   sr13 = 20DE2995   sr14 = 20000001   sr15 = 20000002
 sprg0 = 002F0000  sprg1 = 20DEE9DE  sprg2 = FFD05000  sprg3 = FFD04400
ibat0u = 0000007E ibat1u = 00000000 ibat2u = 00000000 ibat3u = 00000000
ibat0l = 00000003 ibat1l = 00000000 ibat2l = 00000000 ibat3l = 00000000
dbat0u = 0020003E dbat1u = 00000000 dbat2u = 00000000 dbat3u = 00000000
dbat0l = 00200012 dbat1l = 00000000 dbat2l = 00000000 dbat3l = 00000000
  dabr = DEADBEEF   iabr = 00000000   hid0 = 0000C084   boot = 00000082
```

Sample commands that examine and change the processor registers are shown below .

Examine the contents of processor register dar

```
#>p dar <CR>
dar = 00002000,
#>
```

Table 3-5. Processor Registers Accessed via p Command

REGISTER NAME	ACRONYM	TYPE
Boot Register	boot	R/W
Data Storage Interrupt Status Register	dsisr	R/W
Data Address Register	dar	R/W
Floating Point Status Register	fpscr	R/W
Segment Register 0	sr0	R/W
Segment Register 1	sr1	R/W
Segment Register 2	sr2	R/W
Segment Register 3	sr3	R/W
Segment Register 4	sr4	R/W
Segment Register 5	sr5	R/W
Segment Register 6	sr6	R/W
Segment Register 7	sr7	R/W
Segment Register 8	sr8	R/W
Segment Register 9	sr9	R/W
Segment Register 10	sr10	R/W
Segment Register 11	sr11	R/W
Segment Register 12	sr12	R/W
Segment Register 13	sr13	R/W
Segment Register 14	sr14	R/W
Segment Register 15	sr15	R/W
Storage Description Register 1	sdr1	R/W
Special Register G0	sprg0	R/W
Special Register G1	sprg1	R/W
Special Register G2	sprg2	R/W
Special Register G3	sprg3	R/W
Instruction Batch Register 0 Upper	ibat0u	R/W
Instruction Batch Register 0 Lower	ibat0l	R/W
Instruction Batch Register 1 Upper	ibat1u	R/W
Instruction Batch Register 1 Lower	ibat1l	R/W
Instruction Batch Register 2 Upper	ibat2u	R/W
Instruction Batch Register 2 Lower	ibat2l	R/W
Instruction Batch Register 3 Upper	ibat3u	R/W
Instruction Batch Register 3 Lower	ibat3l	R/W
Data Batch Register 0 Upper	dbat0u	R/W
Data Batch Register 0 Lower	dbat0l	R/W
Data Batch Register 1 Upper	dbat1u	R/W
Data Batch Register 1 Lower	dbat1l	R/W
Data Batch Register 2 Upper	dbat2u	R/W
Data Batch Register 2 Lower	dbat2l	R/W
Data Batch Register 3 Upper	dbat3u	R/W
Data Batch Register 3 Lower	dbat3l	R/W
Data Address Breakpoint Register	dabr	R/W
Instruction Address Breakpoint Register	iabr	R/W
Hardware Implementation Dependent Reg 0	hid0	R/W

p PROCESSOR REGISTER DISPLAY/MODIFY (Continued) P

CHANGE THE CONTENTS OF PROCESSOR BOOT REGISTER

```
#>p boot 180. 00000000
#>p boot.
boot = 00000180
```

qa **QUERY ADDRESS** **qa**

Purpose: This command allows either the symbolic name of a specified address or the address of a specified symbolic name to be queried. The symbols table must have been previously loaded by setting bit 7 in the pboot register (e.g. **pboot 80.**) and issuing a **fb** command.

Syntax: **qa<address>**

address The address for which a symbol name is to be displayed.

Example: The following are valid commands.

```
#>qa C0066000 <CR>
hdioctl+2A0 (C0065D60+2A0)

#>qa \hdioctl <CR>
 hdioctl (C0065D60)
```

qb **QUERY BACKPLANE** **qb**

Purpose: This command displays processor status information.

Syntax: **qb**

A sample display from the **qb** command is shown below.

```
#> qb
cpu      up       run
00       y         n
```

qs **QUERY STACK** **qs**

Purpose: This command permits stack status to be queried. The stack status includes the stack pointer and program counter.

Note: Only the KERNEL stack can be displayed.

Syntax: **qs**

Examples: The following is a valid command.

#>**qs.**

```
      --------- KERNEL STACK ---------

         _cnputs() at C00639E6(_cnputs+6)
BFFFE9E2 _cnproc() at C0063908(_cnproc+200)
BFFFEA46 _ttwrite() at C006ACDA(_ttwrite+31E)
BFFFEA8A _raw_rw() at C0050416(_raw_rw+526)
BFFFEACA _write() at C002C680(_write+140)
BFFFEB66 _syscall() at C004F2D2(_syscall+1F6)
BFFFEBB2 _Xtrap0() at C000C58E(_Xtrap0+1E)
```

qv **QUERY VIRTUAL ADDRESS** **qv**

Purpose: This command decodes and prints a virtual address.

Syntax: **qv <virtual address>**

virtual address The virtual address in question.

Examples: The following are valid commands.

NOTE

Page tables should be loaded before expecting complete translations. This action may be accomplished via loading UNIX.

```
#>qv 187b4
** Fnd in ibat0 u=0x0000007e, l=0x00000003   (000187b4)    pp=11
 Vaddr = 0x000187b4    SID=0x0
**1-PTE(0) @ 0x01e00600 = 80000000 00018010   (000187b4)    pp=00
  1-PTE(1) @ 0x01e00608 = 00000000 00000000
  1-PTE(2) @ 0x01e00610 = 00000000 00000000
  1-PTE(3) @ 0x01e00618 = 00000000 00000000
  1-PTE(4) @ 0x01e00620 = 00000000 00000000
  1-PTE(5) @ 0x01e00628 = 00000000 00000000
  1-PTE(6) @ 0x01e00630 = 00000000 00000000
  1-PTE(7) @ 0x01e00638 = 00000000 00000000
  2-PTE(0) @ 0x01eff9c0 = 00000000 00000000
  2-PTE(1) @ 0x01eff9c8 = 00000000 00000000
  2-PTE(2) @ 0x01eff9d0 = 00000000 00000000
  2-PTE(3) @ 0x01eff9d8 = 00000000 00000000
  2-PTE(4) @ 0x01eff9e0 = 00000000 00000000
  2-PTE(5) @ 0x01eff9e8 = 00000000 00000000
  2-PTE(6) @ 0x01eff9f0 = 00000000 00000000
  2-PTE(7) @ 0x01eff9f8 = 00000000 00000000

#>qv a00000
 Vaddr = 0x00a00000    SID=0x0
  1-PTE(0) @ 0x01e28000 = 00000000 00000000
  1-PTE(1) @ 0x01e28008 = 00000000 00000000
  1-PTE(2) @ 0x01e28010 = 00000000 00000000
  1-PTE(3) @ 0x01e28018 = 00000000 00000000
  1-PTE(4) @ 0x01e28020 = 00000000 00000000
  1-PTE(5) @ 0x01e28028 = 00000000 00000000
  1-PTE(6) @ 0x01e28030 = 00000000 00000000
  1-PTE(7) @ 0x01e28038 = 00000000 00000000
  2-PTE(0) @ 0x01ed7fc0 = 00000000 00000000
  2-PTE(1) @ 0x01ed7fc8 = 00000000 00000000
  2-PTE(2) @ 0x01ed7fd0 = 00000000 00000000
  2-PTE(3) @ 0x01ed7fd8 = 00000000 00000000
  2-PTE(4) @ 0x01ed7fe0 = 00000000 00000000
  2-PTE(5) @ 0x01ed7fe8 = 00000000 00000000
  2-PTE(6) @ 0x01ed7ff0 = 00000000 00000000
  2-PTE(7) @ 0x01ed7ff8 = 00000000 00000000
```

qy **QUERY BOOT OPTIONS** **qy**

Purpose: This command displays processor cache status

Syntax: **qy**

Examples: The following is a valid command.

```
#>y0.
#>qy.

                                CPU 0

001 primary data cache:          y
002 primary instruction cache:   y
004 branch history table:        y
008 L2 Cache (256K)              y
010 L2 Cache Copyback            y
```

r **EXECUTE RUN** **r**

Purpose: This command starts the processor executing code. The initial program counter is either specified by the starting address or is taken to be the current value of the program counter.

Function: The **r** command inserts breakpoints and starts the processor executing at the **[start_address]**. If **[start_address]** is not specified, use the current program counter value as the starting address.

Syntax: **r[start_address]**

start_address The address the processor jumps to. If you do not specify a **start_address,** the value of the program counter is used.

Example: The following is a valid command.

r100,0,0,1 Puts the value 1 into r3 and runs the program at memory location 100.

Additional examples of the run command are shown below.

#>**b \cnputs** *(breakpoint at _cnputs)*
#>**r**

```
Processor 0 breakpoint <CR>
C0083DE0 [00083DE0] \cnputs %*67FFF040 subu r31,r31,0x40
```

ra **EXECUTE RUN TO ADDRESS** **ra**

Purpose: This command starts the processor executing code. The initial program counter value is taken to be the current value of the program counter.

Function: The **ra** command creates a temporary breakpoint at **<address>**, inserts breakpoints, and starts the processor executing from current program counter.

Syntax: **ra <address>**

address The address of the application program the processor runs to.

Example: The following is a valid command.

#>**ra \cnrint2ecx** *(run to address of _cnrint2ecx)*

```
CPU 0 breakpoint
C008251C [0008251C] \cnrint2ecx %67FFF0480 subu r31,r31,0x480
#>
```

rd **RUN WITHOUT BREAKPOINTS** **rd**

Purpose: This command starts the processor executing code. The initial program counter is either specified by the starting address or is taken to be the current value of the program counter.

Function: The **rd** command starts the processor executing at [**start_address**] without inserting breakpoints. If [**start_address**] is not specified use the current program counter value as the starting address.

Syntax: **rd [start_address]**

start_address The address of the application program the processor jumps to. If you do not specify a **start_address,** the value of the program counter is used.

Example: The following is a valid command.

#> **rd** Resumes execution of the program.

rn **RUN TO NEXT INSTRUCTION** **rn**

Purpose: This command starts the processor executing code. The initial program counter is either specified by the starting address or is taken to be the current value of the program counter.

Function: The **rn** command creates a temporary breakpoint at the address following the current instruction, insert breakpoints, and starts the processor executing at **[start_address]**. If **[start_address]** is not specified use the current program counter value as the starting address.

Syntax: **rn[start_address]**

start_address The address of the application program the processor jumps to. If you do not specify a **start_address,** the value of the program counter is used.

Example: The following is a valid command.

```
#> rn
CPU 1 breakpoint
C0082520 [00082520]    \cnrint2ecx+4%    21DF0020    st.d
r14,r31,0x20
#>
```

rr **RUN TO RETURN ADDRESS** **rr**

Purpose: This command starts the processor(s) executing code. The initial program counter is either specified by the starting address or is taken to be the current value of the program counter.

Function: The **rr** command creates a temporary breakpoint at the return address of the current C procedure, inserts breakpoints, and starts the processor(s) executing at **[start_address]**. If **[start_address]** is not specified use the current program counter value as the starting address.

Warning: In order to have this instruction function properly you must have executed the first (link) instruction.

Syntax: rr [start_address]

start_address The address of the application program the processor jumps to. If you do not specify a **start_address,** the value of the program counter is used.

Example: The following is a valid command.

```
#> rr
breakpoint: C00908F4

CPU 1 breakpoint
C00908F4 [000908F4] _lock_driv_sema+A4% F440580F or r2,r0,0x14
```

s **SEARCH MEMORY FOR DATA** **s**

Purpose: This command displays a page (256 bytes) of memory in hexadecimal beginning at **start_address.** If the search routine locates the requested **pattern** in this page, it encloses the **pattern** with asterisks. Otherwise, it indicates that there is no match.

Syntax: **s[format][–b<n>]<start_address><pattern>[mask]**

format Determines whether the data to be searched is in byte, word, or longword [**b**, **w**, or **l**] format. If you specify a byte format and have a longword pattern, the routine searches memory but does not find a match.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The address at which the search starts.

pattern The pattern for which memory is searched. The pattern can be a byte, word, or longword.

mask The bit mask, which is a hexadecimal value that determines the part of each longword to be compared with the **pattern.** The mask can be any hexadecimal value from 00000000 to FFFFFFFF. Default is FFFFFFFF. Bit 1 sets the mask.

Examples: The following are valid commands.

s0 70 Search for the pattern 0x70 starting at address 0.

sb0 4 Search for a byte with pattern 4 starting at address 0.

s SEARCH MEMORY FOR DATA (Continued) s

Sample search procedures are shown below.

SEARCH FOR THE PATTERN F0 STARTING AT ADDRESS 0

```
#>i 0 1000 0. #>w 4FC f0.
#>s 0 F0 <CR>
No match @ 00000000,
No match @ 00000100,
No match @ 00000200,
No match @ 00000300,

00000400    00000000    00000000    00000000    00000000
00000410    00000000    00000000    00000000    00000000
00000420    00000000    00000000    00000000    00000000
00000430    00000000    00000000    00000000    00000000
00000440    00000000    00000000    00000000    00000000
00000450    00000000    00000000    00000000    00000000
00000460    00000000    00000000    00000000    00000000
00000470    00000000    00000000    00000000    00000000
00000480    00000000    00000000    00000000    00000000
00000490    00000000    00000000    00000000    00000000
000004A0    00000000    00000000    00000000    00000000
000004B0    00000000    00000000    00000000    00000000
000004C0    00000000    00000000    00000000    00000000
000004D0    00000000    00000000    00000000    00000000
000004E0    00000000    00000000    00000000    00000000
000004F0    00000000    00000000    00000000   *000000F0*
                                                (Pattern match)
```

SEARCH FOR THE PATTERN 70 STARTING AT ADDRESS 0

```
 #>s0 70.

00000000     00000000     00000004     00000008     0000000C
00000010     00000010     00000014     00000018     0000001C
00000020     00000020     00000024     00000028     0000002C
00000030     00000030     00000034     00000038     0000003C
00000040     00000040     00000044     00000048     0000004C
00000050     00000050     00000054     00000058     0000005C
00000060     00000060     00000064     00000068     0000006C
00000070    *00000070*    00000074     00000078     0000007C
00000080     00000080     00000084     00000088     0000008C
00000090     00000090     00000094     00000098     0000009C
000000A0     000000A0     000000A4     000000A8     000000AC
000000B0     000000B0     000000B4     000000B8     000000BC
000000C0     000000C0     000000C4     000000C8     000000CC
000000D0     000000D0     000000D4     000000D8     000000DC
000000E0     000000E0     000000E4     000000E8     000000EC
000000F0     000000F0     000000F4     000000F8     000000FC
```

s SEARCH MEMORY FOR DATA (Continued) s

SEARCH FOR A BYTE WITH PATTERN 4 STARTING AT ADDRESS 0

#>**sB0 4**.

```
00000000   00 00 00 00 00 00 00*04*00 00 00 08 00 00 00 0C
00000010   00 00 00 10 00 00 00 14 00 00 00 18 00 00 00 1C
00000020   00 00 00 20 00 00 00 24 00 00 00 28 00 00 00 2C
00000030   00 00 00 30 00 00 00 34 00 00 00 38 00 00 00 3C
00000040   00 00 00 40 00 00 00 44 00 00 00 48 00 00 00 4C
00000050   00 00 00 50 00 00 00 54 00 00 00 58 00 00 00 5C
00000060   00 00 00 60 00 00 00 64 00 00 00 68 00 00 00 6C
00000070   00 00 00 70 00 00 00 74 00 00 00 78 00 00 00 7C
00000080   00 00 00 80 00 00 00 84 00 00 00 88 00 00 00 8C
00000090   00 00 00 90 00 00 00 94 00 00 00 98 00 00 00 9C
000000A0   00 00 00 A0 00 00 00 A4 00 00 00 A8 00 00 00 AC
000000B0   00 00 00 B0 00 00 00 B4 00 00 00 B8 00 00 00 BC
000000C0   00 00 00 C0 00 00 00 C4 00 00 00 C8 00 00 00 CC
000000D0   00 00 00 D0 00 00 00 D4 00 00 00 D8 00 00 00 DC
000000E0   00 00 00 E0 00 00 00 E4 00 00 00 E8 00 00 00 EC
000000F0   00 00 00 F0 00 00 00 F4 00 00 00 F8 00 00 00 FC
```

sr **SEARCH MEMORY RANGE FOR DATA** **sr**

Purpose: This command searches a range of memory and displays the addresses of all those locations that contained a matching pattern.

Note: When virtual addressing is used, address translation is performed in 'data' space.

Syntax: **sr[format][–b<n>]<start_address><end_address> <pattern>[mask]**

sr[format][–b<n>]<start_address>:<byte_count> pattern> [mask]

format Determines whether the data to be searched is in byte, word, or longword [**b**, **w**, or **l**] format. If you specify a byte format and have a longword pattern, the routine searches memory but does not find a match.

–b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line *(*<*n*> is zero by default).

start_address The address at which the search starts.

end_address The hexadecimal address at which the operation ends.

:byte_count Number (in hexadecimal) of bytes searched. Note that if you specify word format, **byte_count** should be a multiple of two. If you specify **longword format**, **byte_count** should be a multiple of four **pattern.** The pattern for which memory is searched. The pattern can be a byte, word, or longword.

mask The bit mask, which is a hexadecimal value that determines the part of each longword to be compared with the pattern. The mask can be any hexadecimal value from 00000000 to FFFFFFFF. Default is FFFFFFFF. Bit 1 sets the mask.

Examples: The following are valid commands.

sr0 1000 70 Search for the pattern 70 at addresses 0 through 1000.

srb0 100 4 Search for a byte with pattern 4 at addresses 0 through 100.

An example search for a byte with a pattern F0 starting at address 0 is shown below.

```
#>i 0 1000 0.
#>w 4FC F0.
#>sr0 500 F0.
000004FC 000000F0
```

tc **CONFIGURE CONSOLE DEVICE** **tc**

Purpose: This command is used to select a device to be used as the system console.

Function: In the absence of either a keyboard or VGA compatible display device, the default console device of the Motorola SBC system is assigned to serial port 0 (a.k.a **com1**). If the system detects an equipped VGA device (either builtin or a VGA compatible expansion card) AND a working keyboard, then the default console device is normally assigned to the keyboard/display combination.

The **tc** command allows an operator to modify this default behavior and select a specific system console device. This setting can optionally be recorded in a non-volatile RAM area and such that it will persist across subsequent system boots.

Syntax: tc [-w] [-r] [<device>]

-w Write to NVRAM without prompting the operator.

-r Reinitialize console TTY subsystem without prompting the operator. This enables the changed settings immediately.

<device> One of

serial0	Select first serial port
serial1	Select second serial port
kd	Select keyboard/VGA display
default	Restore default behavior
com1 and **com2**	May be used as aliases for **serial0** and **serial1** respectively.

If <device> is omitted, the currently selected console device is printed.

Examples: The following are valid commands.

Print currently selected console device:

```
#>tc
default
```

Force console device to the first serial port:

```
#>tc com1
Update NVRAM (Y/N) ? y
Reinitialize TTY subsystem (Y/N) ? y
#>
```

Force console device to keyboard/display:

```
#>tc -w kd
Reinitialize TTY subsystem (Y/N) ? y
#>
```

tc **CONFIGURE CONSOLE DEVICE (Continued)** **tc**

Restore default console device selection algorithm:

```
#>tc -r -w default
#>
```

td **CONFIGURE CPU DOWN** **td**

Purpose: This command is used on multiprocessor SBCs to mark down the specified CPUs. The **qb** command may be used to display the result.

Syntax: td <cpu list>

Example: The following is a valid command.

td 1 Disable processor 1.

tk **CONFIGURE KEYBOARD DEVICE** **tk**

Purpose: This command is used to select type of keyboard attached to the system.

Function: By default, systems expect a PS/2 or PC/AT style keyboard to be attached via the builtin PC compatible (Intel 8042AH) keyboard interface.

To use a serial interfaced keyboard instead, the operator must specify the port, keyboard type, and serial line parameters of the requested device.

Modified settings can optionally be recorded in a non-volatile RAM area and such that they will persist across subsequent system boots.

Syntax: tk [-r] [-w] [<intfc> [<intfc parms>]]

-w Write to NVRAM without prompting the operator.

-r Reinitialize console TTY subsystem without prompting the operator. This enables modified settings immediately.

<intfc> One of

- **builtin** Builtin PC style keyboard interface
- **serial** Select keyboard attached via standard RS-232 422 serial port
- **default** Restore default keyboard interface.

<intfc parms> Optional parameters that server to further identify keyboard and its interface. The list of possible parameters depends upon the selected keyboard interface.

Builtin (PC-style) keyboard interface parameters are:

mode=[XT/AT]
type=[query|<kbd ID byte>]
codeset=[default|1|2|3|]

Serial port keyboard interface parameters are:

port=[<serial port number>|com1|com2]
baud=[1200|2400|4800|9600]
csz=[8|7]
par=[N|E|O|M|S]
stop=[1|1.5|2]
type=[<kbd ID byte>|101key|84key]
codeset=[default|1|2|3]

If **<intfc>** is omitted, the currently selected keyboard interface and type are printed.

tk CONFIGURE KEYBOARD DEVICE (Continued) tk

Examples: The following are valid commands.

Print keyboard configuration:

```
#>tk
Interface:     builtin
Mode:          XT
Keyboard type: ab (101/102 key)
Scancode set:  default (2)
```

Configure keyboard on serial port 1 (**com2**):

```
#>tk serial port=com2 baud=1200
Update NVRAM (Y/N) ? y
Reinitialize TTY subsystem (Y/N) ? y
#>
```

Print serial keyboard settings:

```
#>tk
Interface:     serial
Port:          1 (com2)
Baud:          1200
Char size:     8
Parity:        None
Stop bits:     1
Keyboard type: 0 (AT2SER converter)
Scancode set:  default (2)
#>
```

Restore default keyboard settings:

```
#>tk -r -w default
#>
```

tu **CONFIGURE CPU UP** **tu**

Purpose: This command is used on multiprocessor SBCs to mark up CPUs. The **qb** command may be used to display the result.

Syntax: **tu all**

tu <cpu list>

all Enables all the boards and all of the processors.

Example: The following is a valid command.

tu 1 Enable CPU 1.

tv **CONFIGURE VIDEO DEVICE** **tv**

Purpose: This command selects the PCI display device to be used for system output. It also establishes default video mode, colors, and cursor type.

Function: A Power Hawk or PowerStack II system may be equipped with more than one VGA compatible device. By default, the VGA compatible expansion card in the lowest numbered PCI/PMC slot number will be selected and enabled if equipped. If no VGA expansion card is present, then the builtin VGA device is selected, if equipped.

The **tv** command allows the system operator to explicitly select which equipped VGA compatible device should be used as the output device when the system is configured for a keyboard/display console.

This command also allows the operator to override the default video mode, colors, and cursor type used by the text mode VGA driver.

Modified settings can optionally be recorded in a non-volatile RAM area. When this occurs, those modified settings will persist across subsequent system boots.

Syntax: **tv [-w] [-r] [device=<dev>] [mode=<mode>] [bg=<bg>] [fg=<fg>] [cursor=<cursor>] [default]**

-w Write to NVRAM without prompting the operator.

-r Reinitialize console TTY subsystem without prompting the operator. This action enables modified settings immediately.

device=<dev> Select a specific VGA device to enable by its PCI bus, device, and function number.

Although there are numerous supported aliases, **<dev>** is actually a comma separated tuple composed of the PCI bus, device, and function number of a specific VGA device. If only two numbers are supplied, then the function number is assumed to be zero and if only a single number is supplied, then the bus and function numbers are both taken as zero.

A number of aliases can be used including **builtin**, **slot1**, **slot2, slot3** representing the physical connection points of PCI devices. The keyword “default” restores the default selection mechanism.

mode=<mode> Select new VGA/SVGA video mode. **<mode>** can be a number from 1 through 27, or its symbolic equivalent (e.g. *color80x25*). See **display(7)** manpage for details of the various supported video modes.

bg=<bg> Select a new background color. Possible values of **<bg>** are black, blue, green, cyan, red, magenta, brown, white, gray, light blue, light green, light cyan, light red, light magenta, yellow, or bright white.

tv **CONFIGURE VIDEO DEVICE (Continued)** **tv**

fg=<fg> Select a new foreground color. See the **bg=<bg>** command for list of possible colors.

cursor=<cursor> Select VGA cursor type. Possible values for **<cursor>** are **uline**, **block**, or **none**.

default If this keyword is present video device parameters are resorted to their default settings.

If no arguments are provided, the currently selected video display device and settings are printed.

Examples: The following are valid commands.

Print current video device and settings:

```
#>tv
VGA device: Requested: default, Actual: PCI/PMC slot1
video mode:  24
fg color:    white (0x7)
bg color:    black (0x0)
cursor type: underline
#>
```

Select builtin VGA instead of expansion card:

```
#>tv device=builtin
Update NVRAM (Y/N) ? y
Reinitialize TTY subsystem (Y/N) ? y
#>
```

Select VGA compatible device in PCI/PMC slot 2:

```
#>tv -w device=slot2
Reinitialize TTY subsystem (Y/N) ? y
#>
```

Select black text on a cyan background with a block cursor:

```
#>tv fg=black bg=cyan cursor=block
Update NVRAM (Y/N) ? y
Reinitialize TTY subsystem (Y/N) ? y
#>
```

Restore default VGA colors:

```
#>tv -r -w fg=default bg=default
#>
```

Restore all video device defaults:

```
#>tv -r -w default
#>
```

w **WRITE DATA TO MEMORY** **w**

Purpose: This command writes the specified hexadecimal data to memory beginning at the **start_address**. The format of the data written is controlled by the options used.

Note: When virtual addressing is used, translation is performed in 'data' space.

Syntax: **w[format][-b<n>]<start_address><data0> [data1]...**

format Determines whether the data is written in byte, word, or longword [**b**, **w**, or **l**] format. Default is **w** in console mode, in processor mode.

-b<n> Specifies program base address. The base address <*n*> is added to all addresses entered from the command line (<*n*> is zero by default).

start_address The hexadecimal address at which the writing starts.

data0, data1 The data to be written to memory. The data must be hexadecimal. Note that multiple data locations can be specified.

Examples: The following are valid commands.

wb0 2. Writes a 2 to byte 0 of memory.

w10 3. Writes a 3 to longword 0 of memory.

Sample write commands are shown below.

WRITE BYTES TO MEMORY STARTING AT ADDRESS 0

```
#>wb0 1 2 3 4 5 6 7 8 9 a. #>d0 10.

00000000 01020304 05060708 090A0000 00000000

00000010 00000000
```

WRITE WORDS TO MEMORY STARTING AT ADDRESS 0

```
#>ww0 1 2 3 4 5 6 7 8 9 a. #>d0 10.

00000000 00010002 00030004 00050006 00070008

00000010 0009000A
```

y **INITIALIZE** **y**

Purpose: This command initializes the CPUs and processor storage and selects the boot options. The flag bits for the **y** command are shown in Table 3-6.

Syntax: **y flags**

flags A numerical value which is the sum of the flag values shown in Table 3-6. below. This command turns off selected portions of the processor cache.

Sample initialization commands are shown below.

#0>**y <CR>** Initializes processor with all caches enabled.

Table 3-6. y Command Flag Bits

Bit	Flag	Effect
B0	001	Disable primary data cache
B1	002	Disable primary instruction cache
B2	004	Disable branch history table
B3	008	Disable L2 Cache
B4	010	Disable L2 Copyback Mode

z SINGLE–STEP PROCESSOR z

Purpose: This command single–steps a single processor one instruction at a time. Breakpoints are inserted into memory. Any pending interrupts are executed before returning control to the console.

Syntax: **`z [address]`**

`address` The address of the instruction to single step. If you omit the **`address`**, the value of the program counter is used.

Examples: The following are valid commands.

```
#>z
CPU 0 single step
C0082520 [00082520] _cnrint2ecx+4% 21DF0020 st.d r14,r31,0x20
(Note that C0082520 is the next instruction to execute.)

#>z
CPU 0 single step
C0082524 [00082524] _cnrint2ecx+8% 221F0028 st.d r16,r31,0x28

#>z
CPU 0 single step
C0082528 [00082528] _cnrint2ecx+C% 225F0030 st.d r18,r31,0x30
```

? HELP COMMAND ?

Purpose: The help command displays a basic list of all the console commands. You can obtain more information about a command by following the question mark (**?**) with the command letter or another **?**. The help command is available immediately following power–up. Examples of the help command are shown below. The **?** command displays help and/or global dash options.

Syntax: **?** List of commands.

?<command> Help on one command.

– Help on global options.

?? Help on parameters.

Example: The following is a valid command.

```
a(scii dump)
as(semble)
b(reakpoint)
c(opy memory)
d(isplay memory in hex)
di(sassemble)
e(xamine/change memory)
f(ile) b(oot) l(oad) r(un) s(cript)
g(eneral register display/modify)
i(nitialize memory to value)
m(emory test)
o(ptions)
p(rocessor register display/modify)
q(uery)
r(un)
s(earch memory)
w(rite data to memory)
z(single step)                    for more help type??


 #>?a
 a(scii dump)
 a[b|w|l][start_address [end_address]]
 a[b|w|l][start_address [:byte_count]]

#>???
(help)
```

? HELP COMMAND (Continued) ?

An expression can be one or more numeric values separated by the arithmetic operators: plus(+) or minus(–).

```
numeric value
    hex digits – hexadecimal number
    $          – last address value
    %          – contents of program counter
    %regname   – contents of processor register
   ‘regname    – address of processor register
    [\]symbol  – address of symbol
    BnBn       – binary bits
    ’n’        – ascii value

address value
    value         – address
    [value]       – physical address
    (value)       – virtual address
    *value:size   – indirect address
    *[value]:size – indirect physical address
    *(value):size – indirect virtual address

?<cmd> – help on <cmd>
?–     – help on command options

 #>?–
 –(command options)

 –r<n> – execute’n’ times (0 = infinite times)
```

A
Command Summary

The following is an alphabetical list of the console commands along with their definition and syntax.

–A–

ASCII Dump

a[format][–b<n>] [start_address [end_address]]
a[format][–b<n>] [start_address [:byte_count]]

–B–

Breakpoints (List)

b

Breakpoints (Set)

b [–a] [–o] [–b<n>] <address>

Breakpoints (Clear)

bk <address> | <all>

–C–

Copy Memory

c[format] [–b<n>] <source_start_address> <source_end_address>
<destination_start_address>
c[format] [–b<n>] <source_start_address> <:byte_count> <destination_start_address>

–D–

Display Memory in Hexadecimal

d[format][–b<n>] [start_address [end_address]]
d[format] [–b<n>] [start_address [:byte_count]]

Disassemble Memory

di [–b<n>][start_address [end_address]]
di [–b<n>]] [start_address [:byte_count]]

–E–

Examine/Change Memory

e[format] [–b<n>] [start_address [data]]

–F–

Boot Operating System

fb

Display Directory

fc [dir_name]

Display/Set the Default Device

fd [-l] [dev]

Display Mounted File Systems

fh

Load and Execute a Program

fr <spec> [base]

–G–

General Register Display/Modify

g <register_name> [data]

–I–

Initialize Memory to Value (Fill)

i[format] [–q] [–b<n>] <start_address> <end_address> [fill_value]
i[format] [–q] [–b<n>] <start_address>: <byte_count> [fill_value]

–M–

Memory Test

m[format] [–b<n>] <start_address> <end_address>
m[format][–b<n>] <start_address>:<byte_count>

–O–

Global Command Options

o[+|–] [m] [r][v] [–b<n>]

–P–

Processor Register Display/Modify

p<register_name> [data]

–Q–

Query Address

qa <address>

Query Backplane

qb

Query Stack

qs

Query Virtual Address

qv <virtual address>

Query Boot Options

qy

–R–

Run (Execute)

r [start_address]

Run (Execute to Address)

ra <address>

Run Without Breakpoints

rd [start_address]

Run to Next Instruction

rn [start_address]

Run to Return Address

rr [start_address]

–S–

Search Memory for Data

s[format] [–b<n>] <start_address> <pattern> [mask]

Search Memory Range for Data

sr[format] [–b<n>] <start_address> <end_address> <pattern> [mask]
sr[format] [–b<n>] <start_address>:<byte_count> <pattern> [mask]

-T-

Configure Console Device

tc [-w] [-r] [<device>]

Configure CPU Down (Multiprocesor SBCs Only)

td <cpu list>

Configure Keyboard Device

tk [-r] [-w] [<intfc> [<intfc parms>]]

Configure CPU Up (Multiprocesor SBCs Only)

tu <cpu list>

Configure Video Device

tv [-w] [-r] [device=<dev>] [mode=<mode>][bg=<bg>] [fg=<fg>]
[cursor=<cursor>] [default]

–W–

Write Data to Memory

w[format] [–b<n>]<start_address> <data0> [data1]

-Y-

Initialize

y flags

–Z–

Single–Step Processor

z [address]

–?–

Help Command

? [–t<n>] (List of commands)
? [–t<n>] <command> (Help on one command)
?– [–t<n>] (Help on global options)
? [–t<n>] (Help on parameters)

The following is a list of the console commands by function.

REGISTER AND MEMORY MANIPULATION

ASCII Dump

a[format][–b<n>][start_address [end_address]]
a[format][–b<n>][start_address [:byte_count]]

Copy Memory

c[format][–b<n>] <source_start_address> <source_end_address>
<destination_start_address>
c[format] [–b<n>] <source_start_address> <:byte_count> <destination_start_address>

Display Memory in Hexadecimal

d[format] [–b<n>] [start_address [end_address]]
d[format] [–b<n>] [start_address [:byte_count]]

Disassemble Memory

di [–b<n>] [start_address [end_address]]
di [–b<n>] [start_address [:byte_count]]

Examine/Change Memory

e[format] [–b<n>] [start_address [data]]

General Register Display/Modify

g <register_name> [data]

Initialize Memory to Value (Fill)

i[format] [–q] [–b<n>] <start_address> <end_address> [fill_value]
i[format] [–q] [–b<n>] <start_address>: <byte_count> [fill_value]

Memory Test

m[format] [–b<n>] <start_address> <end_address>
m[format][–b<n>] <start_address>:<byte_count>

Processor Register Display/Modify

p <register_name> [data]

Search Memory for Data

s[format] [–b<n>] <start_address> <pattern> [mask]

Search Memory Range for Data

sr[format][–b<n>] <start_address> <end_address> <pattern> [mask]
sr[format][–b<n>] <start_address>:<byte_count> <pattern> [mask]

Write Data to Memory

w[format] [–b<n>] <start_address> <data0> [data1]

FILE OPERATIONS

Boot Operating System

fb

Display Directory

fc [dir_name]

Display/Set the Default Device

fd [-l] [dev]

Display Mounted File Systems

fh

Load and Execute a Program

fr<spec> [base]

EXECUTION

List Breakpoints

b

Set Breakpoints

b [–a] [–o] [–b<n>] <address>

Clear Breakpoints

bk <address> | <all>

Execute Run

r[start_address]

Execute Run To Address

ra <address>

Run Without Breakpoints

rd [start_address]

Run To Next Instruction

rn [start_address]

Run to Return Address

rr [start_address]

Single–Step Processor

z [address]

HELP

Help Command

? [–t<n>]	(List of commands)
? [–t<n>]<command>	(Help on one command)
?– [–t<n>]	(Help on global options)
?? [–t<n>]	(Help on parameters)

MISCELLANEOUS

Global Command Options

o [+|–][m] [r] [v][–b<n>]

Query Address

qa<address>

Query Backplane

qb

Query Stack

qs

Query Virtual Address

qv<virtual address>

Query Boot Options

qy

Configure Console Device

tc [-w] [-r] [<device>]

Configure CPU Down (Multiprocessor SBCs Only)

td <cpu list>

Configure Keyboard Device

tk [-r] [-w] [<intfc> [<intfc parms>]]

Configure CPU Up (Multiprocessor SBCs Only)

tu <cpu list>

Configure Video Device

tv [-w] [-r] [device=<dev>] [mode=<mode>][bg=<bg>] [fg=<fg>]
[cursor=<cursor>] [default]

Initialize

y flags

Table A-1 lists the command parameters, range and their definitions.

Table A-1. Command Parameter Definitions

Parameter	Range	Comment
address		Can be any valid physical or virtual address (if the **o+v** option is set), including the device address.
base		The address into which the program is loaded. Default is 2000
byte_count		Number of bytes displayed.
data	[00000000–FFFFFFFF]	Data to be passed to the program. This data is format dependent.
dev	mt(c,u,p,b) dsk(c,u,p,b) where c = slot number; u=unit on controller ‘c’; p=partition number (0–7); b=bus number (0)	The device that is used by the command.
dir_name		The directory name.

Table A-1. Command Parameter Definitions (Cont.)

Parameter	Range	Comment
destination_start_ address		The address where the destination is started
end_address		The address at which the operation stops.
fill_value	[00000000–FFFFFFFF]	The value that is loaded into each memory location.
format	[**b**, **w**, or **l**]	The amount of bits that the data appears in. Formats are byte, word, or longword (**b**, **w**, or **l**)
mask		The bit mask is a hexadecimal value that determines which part of each longword is to be compared with the pattern.
options	**–b<n>, –p,–o, –c<n>, –r<n>, –t<n>, –y<n>, –a, –x, –k, –u, –d<n>, –e, –g**	The conditions the console operates under.
pattern	[00000000–FFFFFFFF]	The pattern for which memory is searched.
register_name	pc, msr, cr, spl, r0– r31, lr, ctr, xer, mq, tid, dsisr, dar, fpscr, sr0-sr15, sdr0, sdr1, eim0, eim1, eis0, eis1, 23	The name of a register.
spec		A file specification, in the following format: **[dev] pathname**. This file contains the file to be loaded. If the **m** option of the **o** command is set **(o+m)**, you must provide the device on which the directory is located via a **fd** command or by specifying **dsk/**.
start_address		The address at which the operation starts.

B
Error Codes

The following is a numerical list of the console error codes that may appear on the screen whenever a console command is executed and an error is detected.

Debugger Error Codes

error 0001: syntax error

The command entered contained a syntax error. Use the help command to obtain the correct syntax (e.g., **?d**).

error 0002: undefined symbol

The symbol name used is not defined or the symbols are not loaded. If trying to reference a processor symbol, ensure that the console is in the processor mode (**o+p**) and the processor symbol table has been loaded. To load processor symbols, bit 7 of the pboot register must be set (e.g., **pboot 80.**) before issuing the **fb** or **fr** command.

error 0003: starting address must be less than ending address

When specifying an address range, the second address must be greater than the first address. To specify a byte count instead of an ending address use a colon '**:**' (e.g., **d 100:10**).

error 0004: illegal CPU number

The processor number used is not a valid processor. Ensure that the processor number is a processor and the processor is marked 'up'. The **qb** command may be used to query the current configuration.

error 0005: invalid stack frame

The **rr** command requires that a valid stack frame exists to enable a return address to be extracted. Ensure that the processor has executed the link instruction of the current C procedure.

error 0006: date/time format: y/m/d h:m:s

When setting the time of century clock, the format of the date and time was incorrect. The correct format is: *y/m/d h:m:s.*

error 0007: duplicate breakpoint

An attempt was made to set a breakpoint at an address that already contained a breakpoint. To correct this situation, remove the old breakpoint first with a **bk** command.

error 0008: breakpoint table full

An attempt was made to set a breakpoint when eight breakpoints already exist. A maximum of eight breakpoints may be set at any one time. To correct this situation remove an existing breakpoint before setting the new one.

error 0009: memory doesn't match

While using the **e** command, data read does not match data written. The **e** command always verifies that it can read any data that it writes.

error 000B: illegal option

An illegal option letter was specified on the **o** command. A list of legal options may be displayed by typing **?o**.

error 000C: illegal register

The register name specified after the '%' was not a valid register name. Use the **g** and **p** command to obtain a complete list of valid register names. If trying to reference a processor register, ensure that the console is in the processor mode (**o+p**).

error 000D: no symbol for address

An address was specified on a command that did not correspond to a symbol name. Ensure that the correct mode processor (**o+p**) or **CP** (**o–p**) is set. Also ensure that the symbol table is loaded. To load processor symbols, bit 7 of the pboot register must be set (e.g., **pboot 80**.) before issuing the **fb** or **fr** command.

error 000F: illegal option '–n'

An illegal option **n** was used on a command. Use the help command to obtain a list of legal options for the command.

error 0010: option '–n' requires an argument

The option n requires an argument. Ensure that there is no space between the option letter and the argument (e.g., **–c3** is correct, **–c 3** is wrong). Use the help command to get a list of legal options for the command.

error 0011: console locked

An attempt was made to use the console when it was disabled at the control panel switch. Before using the console, it must be enabled at the control panel switch.

error 0012: unable to access memory using backplane

The console is unable to access system memory using the backplane. Most likely system memory is not functional. Use the **m** command to verify system memory.

error 0013: failed to load/boot

The **fb** command was unable to load the program "boot" from the default boot device. Ensure that the correct boot device is selected with the **fd** command.

error 0014: low tocc battery

When accessing the time of century clock (TOCC), the console detected a low battery. To correct this problem, replace the TOCC battery.

error 001B: bad device or pathname

An invalid device name or pathname was specified on one of the **f** commands. Ensure that the pathname starts with a slash **/**. Use the help command **?f** to verify the correct syntax is being used. The device must be either **dsk(c,u,p,b)** or **mt(c,u,p,b)** (e.g., **fd dsk(0), fd mt(0)**.

Where: **c** – controller number
u – unit number (optional)
p – partition number (optional)
b – bus number (optional) (0=primary).

error 0020: read failed, offset x

A disk or tape read at byte offset *x* failed. To correct this problem, try a different disk or tape drive.

error 0021: open failed

The open of a disk or tape failed.

error 0022: n in open

An illegal bus was specified on a **f** command. Use a **0** to specify the primary bus, and a **1** for the secondary bus. Ensure that the bus exists. Use the **qb** command to query the system configuration.

error 0022: n in open

A device type other than ‘dsk’ or ‘mt’ was used.

error 0023: not a directory

Either an I/O error occurred or the device does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command.

error 0027: CPU x failed to single step

The console single steps a processor by setting the trace bit in the **sr** register and running that particular processor. If the processor does not take a trace exception within one second, the single step fails. This error most frequently occurs with the **z** command, but it can also occur with the **r** command. The **r** command uses single stepping to skip over breakpoints.

error 0029: a: expected n, actual m

A memory test detected an error at address a. The value *n* was written to memory and the value *m* was read back.

error 002A: All CPUs must be halted

An **f** command was attempted when one or more processors where running. All of the processors must be halted with the **h** command before initiating an **f** command.

error 002E: invalid memory destination

An internal error occurred in the console which caused it to perform an invalid memory reference. Reset the console with **<CR>~b** and retry the command.

error 002F: register 'n' is read only

An attempt was made to modify processor register *n*. Register *n* is a read only register and may not be modified.

error 0030: CPU must be running

A command was attempted that expected a running processor. If a **tu** command (applicable to multiprocessor SBCs only) was being attempted use the **r** command instead.

Console Debugger Error Codes

error 0201: boot script missing

A script by the name 'boot' should always exist. The 'boot' script gets executed at power up and should contain commands to boot the operating system. The default 'boot' script contains the command **fb**.

error 0202: slot n is not valid

A command referenced slot *n* that was either empty or did not contain a board of the proper type. Use the **qb** command to display the hardware configuration of the system.

error 020F: invalid segment descriptor, vaddr=n

While translating virtual address *n* to a physical address, the console referenced an invalid segment descriptor in system memory. Ensure that the crp and segment descriptor are valid.

error 0210: page not in memory, vaddr=n

While translating virtual address *n* to a physical address, the console detected that the page containing the virtual address was not in system memory. Since the page is not in memory the data in this page is not accessible to the console. Ensure that the virtual address is within the bounds of the memory allocated to be accessed by the console.

error 0264: CPU n marked down

For multiprocessor SBCs only. Processor *n* was marked down due to either a **td** command or the detection of an error. The up/down status of a processor can be checked via the **qb** command.

error 0265: CPU n is not valid

A command referenced a processor that does not exist. Use the **qb** command to display the system hardware configuration.

error 0267: vaddr (n) is supervisor protected

An attempt was made to perform a virtual address translation in user space, and that address was marked as supervisor – protected.

error 0268: (batc) probe operation failed on CPU n

The console instructed a secondary processor to perform a memory management unit probe operation to check for BATC valid translation at a given address. No response was received.

error 0280: CPU n failed to acknowledge DCB request

The console could not communicate with processor *n*.

error 0281: CPU n failed to set DCB done bit

Processor *n* did not complete a console request.

I/O Error Codes

error 0601: null path

An internal console error occurred when the console was opening a file. Reset the console with **<CR>~b** and retry the command.

error 0602: file not found

An attempt was made to open a file that does not exist. Ensure that a valid pathname was specified. Use the **fc** command to verify that the file exists. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0603: block number negative

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0604: block number overflow

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0605: indirect block number void

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0606: block number void

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0607: not a directory

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0608: zero length directory

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 060E: cannot write files

An attempt was made to write a file. The console does not support device writes. This error code should not occur under normal operating conditions and therefore it indicates an operator error. Reset the console with **<CR>~b** and retry the command.

error 060F: no more file slots

An internal console error occurred while opening a file. Reset the console with **<CR>~b** and retry the command. If error still occurs, suspect a corrupted file.

error 0610: no more disk buffers

An internal console error occurred while allocating a disk buffer. This error code should not occur under normal operating conditions and therefore it indicates an operator error. Reset the console with **<CR>~b** and retry the command.

error 0611: super block read error

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0612: read error

An I/O read error occurred. Previous message should indicate type of error.

error 0613: zero length directory record

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0614: bad magic number in super block

The device being read does not contain a valid file system. Verify that the disk or tape contains valid data and retry the command. Reset system via the **fd** command to ensure that the media is properly partitioned. Suspect corrupt file, rerun from a back–up file.

error 0615: CP system device unavailable, retrying

The console was unable to open the boot device during a **fb** command. The console attempts 12 retries then stops trying. Ensure that at least one Generic Disk (GD) disk controller exists in the primary I/O bus and the disk is spun up and ready.

640 Series Console Errors

These error codes (0640 through 064F) apply to the Concurrent SCSI Adapters, Generic Disk (GD), and Generic Tape (GT) devices, and the definition of the symptom will reflect which controller is displaying the error code. For example, if **error 0645: NCR Controller not found** is displayed on the console terminal, then the NCR controller is the source of the error code. However, if the generic disk controller is indicating this same error: **error 0645: GD:Controller not found** is displayed on the console terminal.

error 0640: ___: No error

This error can only result from an internal hardware or software error.

error 0641: ___: Interface not configured

This error results from selecting a disk or tape device that is not valid for this machine type. Use the **fd** command to select an appropriate device type.

error 0642: ___: Invalid Command

This error can only result from an internal hardware or software error.

error 0643: ___: Unsupported command

This error can only result from an internal hardware or software error.

error 0644: ___: Bad device specification

This error results from selecting a disk or tape device that is not valid. Use the **fd** command to select an appropriate device type.

error 0645: ___: Controller not found

This error results when an invalid controller number has been provided as part of a device specification. Use the **fd** command to select an appropriate controller number (0 through 9).

error 0646: ___: Device not found

This error results when the device was not found at the specified hardware address. Use the **fd** command to select a new device address.

error 0647: ___: Device type mismatch

This error results when the device specification referred to a disk (tape) when the real device found at that address was a tape (disk). Ensure that the correct address was used with the **fd** command.

error 0648: ___: Controller timed out

This error results when a I/O controller (NCR or IS) or device malfunctions, or an internal software error occurs. Suspect the I/O controller or device malfunction. Reset the system and retry the command.

error 0649: ___: Controller reports fatal error

This error results when a I/O controller (NCR or IS) or device malfunctions, or an internal software error occurs. Suspect the I/O controller or device malfunction. Reset the system and retry the command.

error 064A: ___: Unrecovered device error

This error results when a I/O controller (NCR or IS) or device malfunctions, or an internal software error occurs. Suspect the I/O controller or device malfunction. Reset the system and retry the command.

error 064B: ___: Device not ready

This type of error occurs when the device is off–line, disk is not up to speed, or malfunctioning. Ensure that the device is off–line and operational. Reset the system and retry command.

error 064C: ___: Unit attention condition

This type of error occurs for an unexpected SCSI device/bus reset, device power loss, or media change. Ensure that the device is on–line and operational. Reset the system and retry command.

error 064D: ___: Device hit a filemark

This type of error occurs for an unexpected filemark or the End–Of–Valid–Data indicator was hit during tape operations. Probable tape read error. Reset the system and retry the command. If subsequent attempts also fail, secure a new tape and retry command. If, with a new tape, the attempt fails, suspect the device is malfunctioning and should be replaced.

error 064E: ___: Device reports end–of–medium

This type of error occurs for an unexpected filemark or the End–Of–Valid–Data indicator was hit during tape operations. Probable tape read error. Reset the system and retry the command. If subsequent attempts also fail, secure a new tape and retry command. If, with a new tape, the attempt fails, suspect the device is malfunctioning and should be replaced.

error 064F: ___: Device busy

This error occurs when an unexpected “BUSY” condition is reported by the SCSI device. Reset the system and retry the command. If the error condition still exists, suspect a device malfunction and replace suspected device.

650 Series Console Errors

These error codes (0650 through 065A) apply to the SCSI Adapter and the definition of the symptom will reflect which controller is displaying the error code. For example, if **error 0650: Bad NCR module id** is displayed on the console terminal then the NCR controller is the source of the error code.

error 0650: Bad ___ module id

A probe for a controller returned a bad module id code. Ensure that a controller exists in the slot being probed. If a controller exists suspect the controller.

error 0651: Bad ___ bus no

An open of a device was attempted with a bad bus number. Ensure that a valid bus number (**0** = primary) is specified on the **fd** command.

error 0652: Bad ___ slot no

An open of a device was attempted with a bad slot number. Ensure that a valid slot number (2 through 9) is specified on the **fd** command.

error 0653: Bad ___ ctrl no

An open of a device was attempted with a bad controller number. Ensure that a valid controller number (2 through 9) is specified on the **fd** command.

error 0654: Bad ___ unit no

An open of a device was attempted with a bad drive number. Ensure that a valid unit number (0 through 7) is specified on the **fd** command.

error 0655: Bad ___ partition no

An open of a device was attempted with a bad partition number. Ensure that a valid partition number (0 through 7) is specified on the **fd** command.

error 0657: ___: SCSI request sense failed

This type of error occurs when a NCR or SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device or controller malfunction. Attempt to run the I/O diagnostic programs.

error 0658: ___: SCSI inquiry failed

This type of error occurs when a NCR or SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device or controller malfunction. Attempt to run the I/O diagnostic programs.

error 0659: ___: SCSI test unit ready failed

This type of error occurs when a NCR or SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device or controller malfunction. Attempt to run the I/O diagnostic programs.

error 065A: ___: SCSI load tape command failed

This type of error occurs when a NCR or SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device or controller malfunction. Attempt to run the I/O diagnostic programs.

6B0 Series Console Errors

These error codes (06B0 through 06B4) apply to the Generic Disk (GD) and Generic Tape (GT) devices, and the definition of the symptom will reflect which controller is displaying the error code. For example, if **error 06B0:GT:Interface not found** is displayed on the console terminal, the generic tape controller is the source of the error code. However, if the generic disk controller is indicating this same error: **error 06B0: GD: Interface not found** is displayed on the console terminal.

error 06B0: ___: Interface not found

This error can only result from an internal hardware or software error.

error 06B1: ___: Device not initialized

This error can only result from an internal hardware or software error.

error 06B2: ___: Read failed

This error results when a I/O controller or device malfunctions, or an internal software error occurs. Suspect the I/O controller or device malfunction. Reset the system and retry the command. If the operation still fails, ensure that the I/O controller and device are all the current revision and run the diagnostic programs to validate the hardware.

error 06B3: ___: Write unsupported

This error can only result from an internal hardware or software error.

error 06B4: ___: Bad request size

This error can only result from an internal hardware or software error.

error 06C0: GD: Can't read disk status

This error results when a I/O controller or device malfunctions, or an internal software error occurs. Suspect the I/O controller or device malfunction. Reset the system and retry the command. If the operation still fails, ensure that the I/O controller or device are all the current revision and run the diagnostic programs to validate the hardware.

error 06C2: GD: Drive off-line

An I/O request to the GD controller returned drive off–line status. Ensure that the drive is on–line and retry the command.

error 06C3: GD: Can't read geometry block

Either an I/O error occurred or the disk has not been formatted. Verify that the disk has been properly formatted. Suspect a medium fault. Restore file and reformat.

error 06C4: GD: Bad geometry block header

Either an I/O error occurred or the disk has not been formatted. Verify that the disk has been properly formatted. Suspect a medium fault. Restore file and reformat.

error 06C5: GD: Bad geometry block checksum

Either an I/O error occurred or the disk has not been formatted. Verify that the disk has been properly formatted. Suspect a medium fault. Restore file and reformat.

error 06C6: GD: Null partition

A null length partition was specified on the **fd** command. To correct this situation, select a different partition with the **fd dsk(n,n,partition no.)** command and retry the command. Suspect media.

error 06D0: GT: Seek failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D1: GT: Load command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D2: GT: Unload command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D3: GT: Rewind command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D4: GT: Space fwd file command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D5: GT: Space back rec command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D6: GT: Space fwd rec command failed

This type of error occurs when a SCSI device is malfunctioning. Reset the system and retry the command. If the error still occurs, suspect a device malfunction. Attempt to run the I/O diagnostic programs and replace board(s) indicated.

error 06D7: GT: Cannot seek to partition

This type of error occurs for an unexpected filemark or the End–Of–Valid–Data indicator was hit during tape operations. Probable tape read error. Reset the system and retry the command. If subsequent attempts also fail, secure a new tape and retry command. If, with a new tape, the attempt fails, suspect the device is malfunctioning and should be replaced.

error 0972: CPU n failed interrupt reset: RHAC = X

Processor *n* could not reset all interrupt requests.

error 0973: CPU n IGA configuration ram xxx: expected xx received xx

Processor *n* interrupt configuration RAM failed test.

error 0974: CPU n IGA level decode programming failed

Processor *n* failed the level decode programming.

error 0975: CPU n IGA ipl decode programming failed

Processor *n* failed the ipl decode programming.

error 0976: CPU n IGA vector table programming failed

Processor *n* failed the vector table programming.

error 0980: CPU n exception: vector x ("description") epsr x exip x enip x

Processor *n* had an unexpected exception, a register dump follows to aid in debugging

error 0981: CPU n cannot be disabled

Processor *n* can not be disabled via normal procedures.

error 0983: Cannot single-step across a trap/rte instruction, pc = x

A trap or rte instruction at pc location x cannot be single-stepped.

0984: exception x occurred while processing exception y

Another exception *x* occurred during the processing of exception *y*.

error 0990: board in slot n failed to report board configuration

The processor board in slot *n* did not power up/reset correctly.

error 0991: CPU n failed to report status after reset

Processor *n* did not report its error status after a reset.

error 0992: CPU n reported error code x (description)

Processor *n* reported error *x* which is described here.

error 0993: Global memory x failed RAM test

The global memory *x* failed RAM test during reset, and is currently not being used by the console.

error 0999: Backplane reset aborted

An error condition was detected which caused the reset of the backplane to be aborted.

Index

Spine for 1/2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up