OS-9 for MVME Board Guide

Version 3.2

www.radisys.com
World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

RadiSys Microware Communications Software Division, Inc.
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Revision A
December 2001

Copyright and publication information

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

Table of Contents

Chapter 1: Installing and Configuring OS-9 **7**

8 Development Environment Overview
9 Requirements and Compatibility
9 Host Hardware Requirements (PC Compatible)
9 Host Software Requirements (PC Compatible)
9 Target Hardware Requirements
10 Target Hardware Setup
11 Connecting the Target to the Host
15 Building the OS-9 ROM Image
15 Coreboot
15 Bootfile
16 Using the Configuration Wizard
18 Creating and Configuring the ROM Image
18 Select System Type
19 Configure Coreboot Options
21 Configure System Options
22 Network Configuration
24 Disk Configuration
26 Build Image
27 Transferring the ROM Image to the Target
27 Configure the TFTP Server
28 Boot the Target from an Ethernet Network
30 Creating a Startup File
31 Example Startup File
33 Optional Procedures
33 Preliminary Testing
34 Booting Your Reference Board from Flash
39 Disk Booting RBF

Chapter 2: Board Specific Reference 43

44 Boot Menu Options
46 Vector Descriptions for PowerPC 603/604
48 Error Exceptions: vectors 2-4 and 6-7
48 Vectored Interrupts: vector 5
49 User Trap Handlers: vector 7
49 System Calls: vector 12
50 Configuring Booters
53 Port Specific Utilities
63 PowerPC™ Registers Passed to a New Process

Appendix A: Board Specific Modules 65

66 Low-Level System Modules
66 Configuration Modules
66 Console Drivers
66 Debugging Module
67 Ethernet Driver
67 SCSI Host Adapter Support Booter Module
67 System Modules
67 Timer Module
68 High-Level System Modules
68 Interrupt Controllers
69 Real Time Clock Driver
69 Ticker
69 Abort Handler
70 Shared Libraries
70 Serial and Console Drivers
72 Parallel Driver
72 Data Disk Drivers
73 SCSI support
74 Common Modules List

Appendix B: Partitioning and Formatting Your Hard Drive **79**

80 Partitioning Your Hard Drive
85 Formatting Your Hard Drive
87 OS-9 Partitioning Options
87 Create OS-9 Partition (1)
87 Set Active Partition (2)
87 Delete Partition (3)
88 Display Partition Information (4)
88 Change Extended DOS Partition to OS-9 Partition (5)

Product Discrepancy Report **89**

Chapter 1: Installing and Configuring OS-9

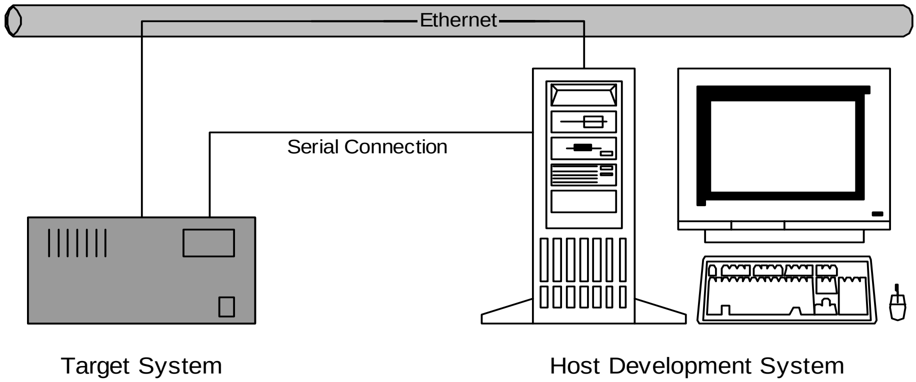
This chapter describes installing and configuring OS-9 on the following Motorola MVME reference boards: MVME1603, MVME1604, MVME2303; MVME2304; MVME2603; MVME2604; MVME2700; MVME3603; MVME3604. The following sections are included:

- **Development Environment Overview**
- **Requirements and Compatibility**
- **Target Hardware Setup**
- **Connecting the Target to the Host**
- **Building the OS-9 ROM Image**
- **Transferring the ROM Image to the Target**
- **Creating a Startup File**
- **Optional Procedures**

Development Environment Overview

Figure 1-1 shows a typical development environment for the MVME boards. The components shown include the minimum required to enable OS-9 to run on the supported boards.

Figure 1-1 MVME Development Environment



Requirements and Compatibility

Host Hardware Requirements (PC Compatible)

Your host PC must meet the following minimum requirements:

- Windows 95, ME, 2000, or NT
- 300-400 MB of free disk space
- An Ethernet network card
- 32MB of RAM
- one free serial port

Host Software Requirements (PC Compatible)

Your host PC must have the following applications

- A terminal emulation program (such as Hyperterminal, which comes with Microsoft Windows 95 and Windows NT).
- `TFTPSERVERPro` server application for downloading the OS-9 ROM image to the MVME target. This application is included with Enhanced OS-9 for PowerPC and must be loaded onto your host PC during the CD-ROM installation process.

Target Hardware Requirements

Your reference board requires the following hardware:

- Enclosure or chassis with power supply
- A RS-232 null modem serial cable
- MVME712/761 transition module (Ethernet and serial connections)
- Disk drives

Target Hardware Setup

You must modify the jumper settings for Flash. When programming the Flash system, you must have the Flash bank B (1MB) area enabled. This enables programming of the Flash bank A (4MB or 8MB) section.

For More Information

Refer to the appropriate ***Installation and Use*** and ***Programmer's Guide*** documents from Motorola for more information about programming the Flash system on your reference board. You can access these documents directly from your browser by opening the following url:

http://www.mcg.mot.com/literature/locator.html

Connecting the Target to the Host

This section describes connecting the target board to the host PC via serial and Ethernet connections.

Complete the following steps to connect the target to the host:

Step 1. Use an RS-232 null modem cable to connect the target to the serial port of your host system. Depending on your host system, you may need either a straight or reversed serial cable.

Step 2. With the target system powered off, connect the serial cable to the COM1 port on the reference board. On the MVME, COM1 is labeled **SERIAL PORT1/CONSOLE** or **COM1** on the transition module. You must also connect the host and target systems to a network to use TFTP.

Step 3. Connect the other end of the serial cable to the desired communication (COM) port on the host system.

Step 4. On the Windows desktop, click on the `Start` button and select `Programs -> Accessories -> Hyperterminal`.

Step 5. Double-click the `Hyper Terminal` icon and enter a name for your Hyperterminal session.

Step 6. Select an icon for the new Hyperterminal session. A new icon is created with the name of your session associated with it. You can select this icon the next time you establish a Hyperterminal session.

Step 7. Click `OK`.

Step 8. From the **Phone Number** dialog, select `Connect Using` and then select the communications port to be used to connect to the target system. Click `OK`.

Step 9. In the **Port Settings** tab, enter the following settings:

```
Bits per second = 9600
Data Bits = 8
Parity = None
Stop bits = 1
Flow control = XOn/XOff
```

Step 10. Click `OK.`

Step 11. From the Hyperterminal window, select `Call -> Connect` from the pull-down menu to establish your terminal session with the target board. When you are connected, the bottom left of your Hyperterminal screen displays *connected*.

Step 12. Turn on the target system. A power-on banner and `PPC1-Bug>` prompt should appear on the display terminal.

Note

If your target system already has an OS-9 ROM image installed, you can get a `PPC1-Bug>` prompt by pressing the `Esc` key during the target system bootup. You can then rebuild the ROM image as desired.

Step 13. From the PPC1-Bug> prompt, type `niot` and configure the target board to receive the file as follows:

`PPC1-Bug>niot`	
`Controller LUN =00?`	
`Device LUN     =00?`	
`Node Control Memory Address =00FA0000?`	should not need to change this
`Client IP Address =182.52.109.68?`	IP address of the reference board
`Server IP Address =182.52.109.53?`	IP address of the machine with tftp boot server
`Subnet IP Address Mask =255.255.255.0?`	fill in as required
`Broadcast IP Address =255.255.255.255?`	fill in as required
`Gateway IP Address =0.0.0.0?`	fill in as required
`Boot File Name ("NULL" for None) =rom?`	name of image to load in tftpboot directory
`Argument File Name ("NULL" for None) =?`	
`Boot File Load Address =00080000?`	load address; must be 0x80000
`Boot File Execution Address =00080000?`	execution address; must be 0x80000
`Boot File Execution Delay =00000000?`	no delay required
`Boot File Length =00000000?`	get length automatically
`Boot File Byte Offset =00000000?`	
`BOOTP/RARP Request Retry =00?`	
`TFTP/ARP Request Retry =00?`	
`Trace Character Buffer Address =00000000?`	
`BOOTP/RARP Request Control: Always/When-Needed (A/W)=W?`	
`BOOTP/RARP Reply Update Control: Yes/No (Y/N) =Y?`	
`Update non-volatile RAM (Y/N) Y`	

Note

The MVME has Ethernet built into the transition module. You must complete this step to configure the board to work on an Ethernet network.

To properly complete the configuration, get the following information from your network administrator:

Table 1-1 System Administrator Input

Information Needed	Information Used for this Tutorial
IP Address and Host Name	____________________
Broadcast IP Address	____________________
Subnet Mask	____________________
Network Domain	____________________
DNS IP Addresses	____________________
Gateway IP Addresses	____________________

Building the OS-9 ROM Image

The OS-9 ROM Image is a set of files and modules that collectively make up the OS-9 operating system. The specific ROM Image contents can vary from system to system depending on hardware capabilities and user requirements.

To simplify the process of loading and testing OS-9, the ROM Image is generally divided into two parts: the low-level image, called `coreboot`, and the high-level image, called `bootfile`.

Coreboot

The coreboot image is generally responsible for initializing hardware devices and locating the high-level (or bootfile) image as specified by its configuration. For example from a FLASH part, a harddisk, or Ethernet. It is also responsible for building basic structures based on the image it finds and passing control to the kernel to bring up the OS-9 system.

Bootfile

The bootfile image contains the kernel and other high-level modules (initialization module, file managers, drivers, descriptors, applications). The image is loaded into memory based on the device you select from the boot menu. The bootfile image normally brings up an OS-9 shell prompt, but can be configured to automatically start an application.

Microware provides a Configuration Wizard to create a coreboot image, a bootfile image, or an entire OS-9 ROM Image. The wizard can also be used to modify an existing image. The Configuration Wizard is automatically installed on your host PC during the Enhanced OS-9 installation process.

Using the Configuration Wizard

Motorola MVME reference boards enable you to boot from a number of devices, including those shown below:

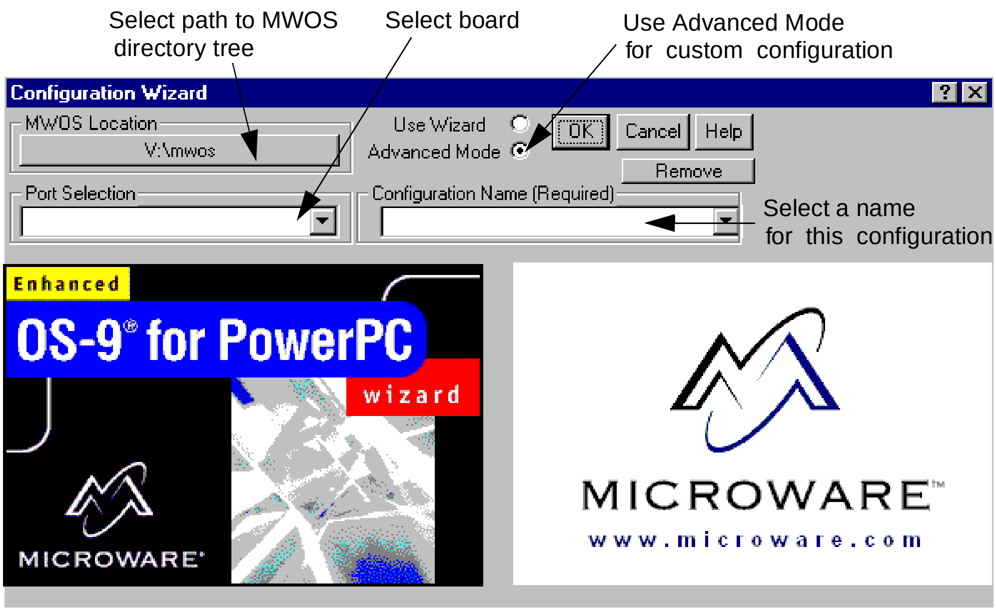
- Flash ROM
- SCSI floppy
- SCSI hard disk
- SCSI tape
- floppy disk
- Ethernet (you will have to supply your own BOOTP server)

Regardless of what device you use for booting, the basic booting process is the same. You need to create a ROM image using the OS-9 Configuration Wizard and then place the ROM image on the boot device.

This section describes using the Configuration Wizard to build the OS-9 ROM image. To use the Wizard, perform the following steps:

Step 1. On the Windows desktop, select `Start` --> `Programs` --> `Microware` --> `Enhanced OS-9 for PowerPC` --> `Microware Configuration Wizard`. You should see the following opening screen.

Figure 1-2 The Configuration Wizard Opening Window



Step 2. Select the path where the MWOS directory structure can be located by clicking in the MWOS location button.

Step 3. Select the target board from the **Port Selection** pull-down menu.

Step 4. Select a name for your configuration in the **Configuration Name** field. Your settings are saved. This enables you to modify the ROM image incrementally, without having to reselect every option for each change.

Creating and Configuring the ROM Image

This section describes how to use the Configuration Wizard to create and configure your OS-9 ROM image.

Note

For the 8XX target boards, the coreboot and bootfile images are combined into one image, called ROM. The rom image is then transferred from the host PC to the target board.

Note

This section provides an example of an OS-9 ROM image successfully built on a Host PC and transferred to an 823FADS target board. You may have to modify your selections depending on your application.

Select System Type

Configure system type options by selecting `Configure` -> `Sys` -> `Select System Type` from the Main Configuration window.

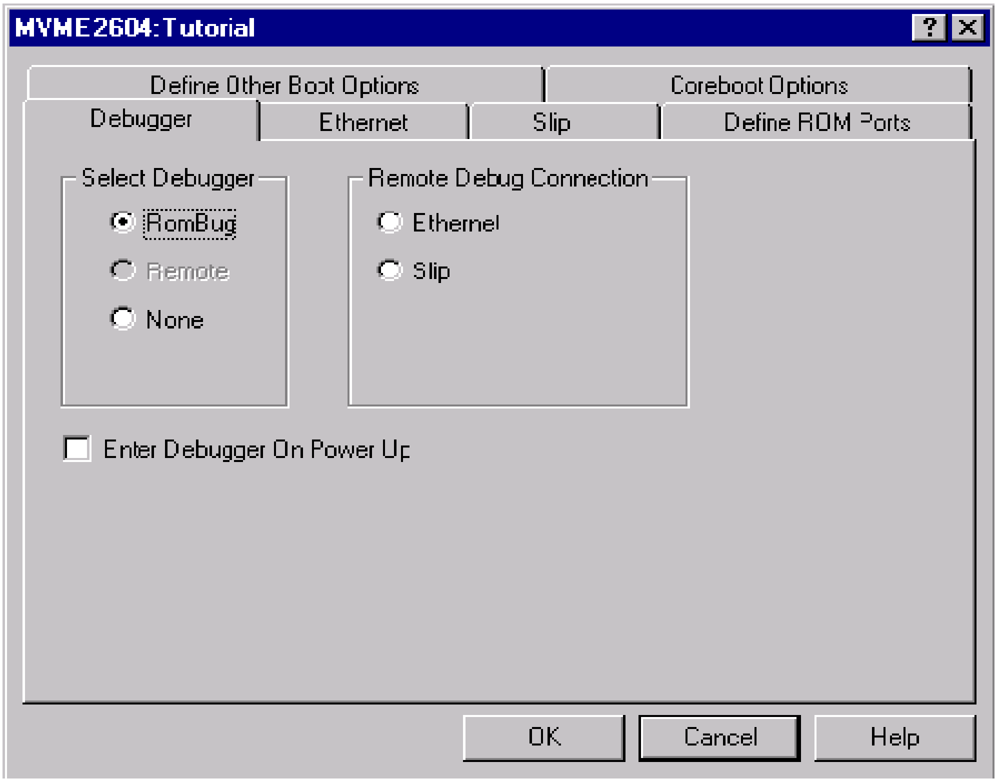
For the 823FADS target board, you can bypass this option and use the default settings.

Configure Coreboot Options

Step 1. From the Main Configuration window, select `Configure` -> `Coreboot` -> `Main configuration`.

Step 2. Select the **Debugger** tab. The following window is displayed.

Figure 1-3 Coreboot Configuration—Debugger Tab



Step 3. Under **Select Debugger**, select `RomBug`. This sets Ethernet as the method for user state debugging. Select `None` if you do not want to debug your program.

Note

To perform system state debugging, select `Ethernet` under **Remote Debug Connection**. If you set Ethernet as the method for system state debugging, you will not be able to perform user state debugging via Ethernet.

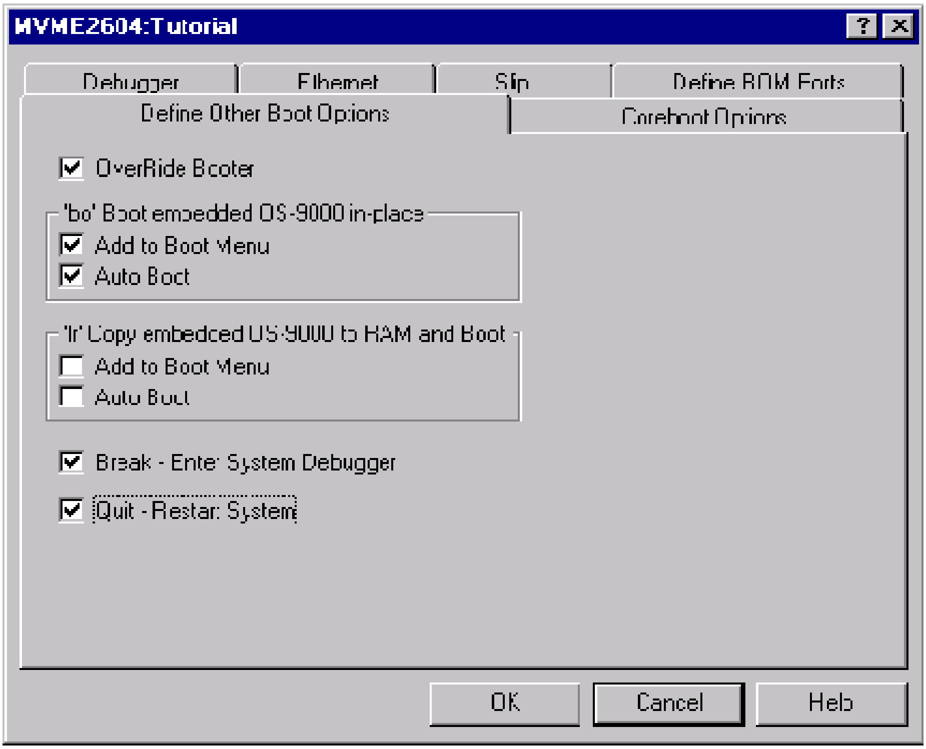
For system state debugging, you must also set the parameters in the **Ethernet** tab of the coreboot configuration.

Note

The addresses shown are for demonstration only. Contact your network administrator to obtain your setup information.

Step 4. Select Define Other Boot Options. The following window is displayed.

Figure 1-4 Coreboot Configuration—Define Other Boot Options Tab



Step 5. Select Break-Enter System Debugger.

Step 6. Click OK and return to the **Main Configuration** window.

Configure System Options

Configure system options by selecting Configure -> Bootfile -> Configure System Options from the **Main Configuration** window. You can bypass this option and use the default settings.

Network Configuration

To use the target board across a network, you must enable the Ethernet network settings. The **IP Address**, **DNS Configuration**, and **Gateway** tabs of the network configuration are similar to the **TCP/IP Properties** window in Windows.

Step 1. From the **Main Configuration** window, select `Configure` -> `Bootfile` -> `Network Configuration`.

Step 2. Select the `IP Address` tab.

Step 3. Click the `Specify an IP Address` button.

- Enter your IP address.
- Enter your broadcast address.
- Enter the subnet mask.
- Set the pulldown menu at the bottom of the window to `DEC21140-AUI`.

Step 4. Select the `DNS Configuration` tab.

More than one DNS server can be added in this dialog box. If your network does not use DNS, click `Disable DNS`, and move to the **Gateway** tab.

If you have DNS available, click `Enable DNS` and type your host name and domain.

Note

You add DNS IP addresses by clicking on the box directly under **DNS Server Search Order** and typing the IP address. Click the Add button when complete.

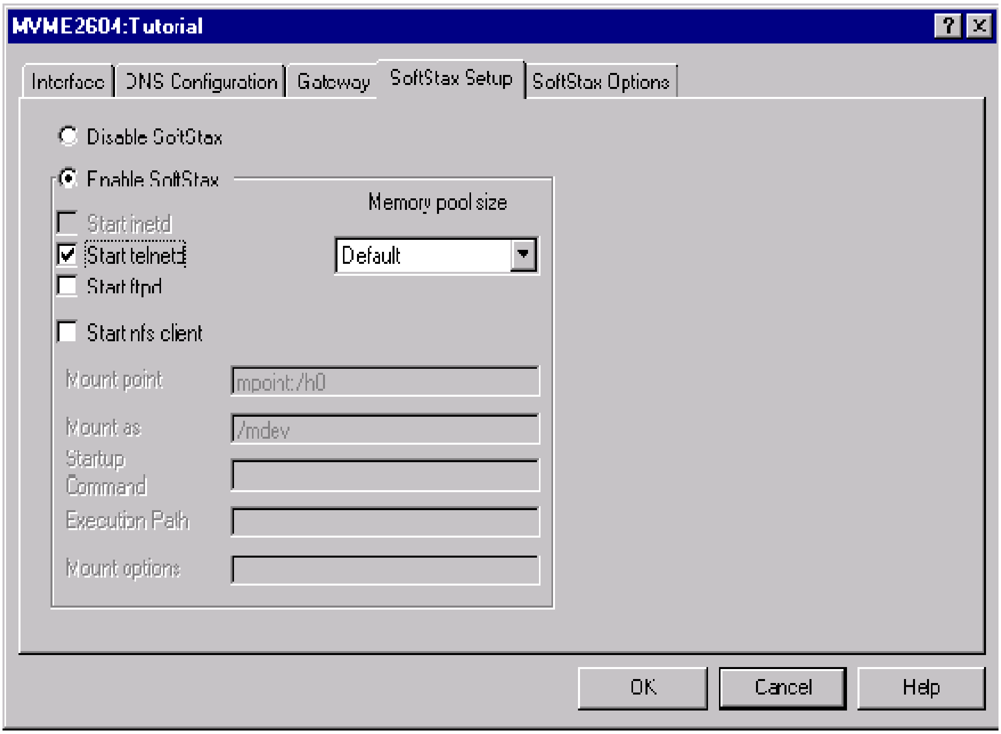
More than one DNS server can be added by repeating these steps.

Step 5. Select the **Gateway** tab. Add new gateway addresses by clicking on the box and typing in the gateway name. Click the Add button when complete.

Step 6. Select the `SoftStax Setup` tab. The following window is displayed.

The options below represent daemons that can be automatically started if you want to FTP or telnet from a PC to the OS-9 target. **Start NFS Client** enables you to remote mount the target. For this demonstration, you will telnet to the target and establish a sender window and a receiver window.

Figure 1-5 Bootfile Configuration—SoftStax Setup Tab



Step 7. Click `Enable SoftStax`.

Step 8. Click `Start telnetd`. (The only checked box on this tab should be the **Start telnetd** box.)

Step 9. Click `OK`.

Step 10. Select the **SoftStax Options** tab.

The **SoftStax Options** tab enables you to include networking utilities in the ROM image. By default, `ftp`, `hostname`, `ping`, and `netstat` are included. You can add other utilities as desired.

Step 11. Click `OK` at the bottom of the **Network Configuration** menu to complete network configuration and return to the **Main Configuration** window.

Disk Configuration

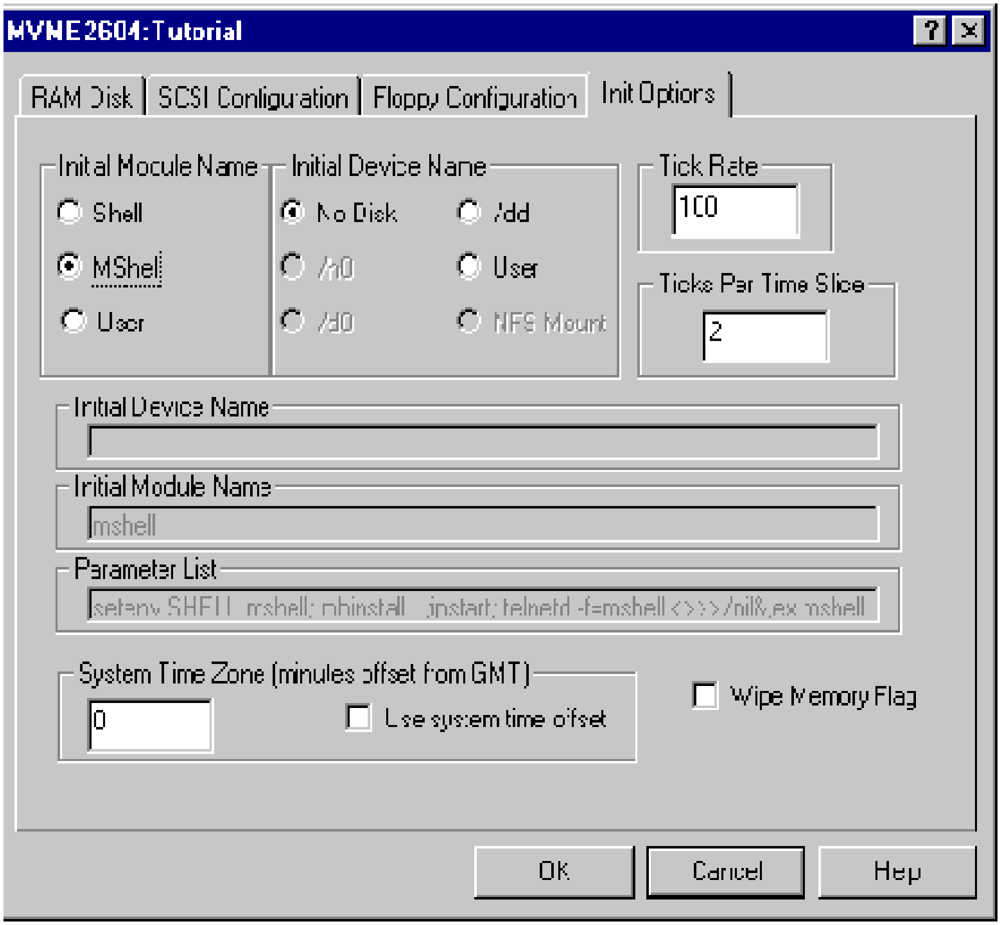
Step 1. From the Main Configuration window, select `Configure` -> `Bootfile` -> `Disk Configuration`.

The Disk Configuration options include the following tabs:

- The **RAM Disk** tab enables you to create a RAM disk of any size for loading modules onto the target.
- The **SCSI Configuration** tab enables you to configure SCSI drives for the target.
- The **Floppy Configuration** tab enables you to configure a floppy drive for the target.
- The **Init Options** tab sets the configuration for OS-9 to initialize itself on the target.

Step 2. Select the **Init Options** tab. The following window is displayed.

Figure 1-6 Bootfile Configuration—Init Options Tab



Step 3. Select the Mshell option for the initial module name. This causes OS-9 to start a console shell usable from your terminal window. **Initial Device Name** should be selected as No Disk.

The tick rate is 100 and ticks per timeslice is set to 2. If you look at the **Parameter List** box, you see the commands that OS-9 executes upon system start-up.

Step 4. Click OK to return to the **Main Configuration** window.

Build Image

Complete the following steps to build the target board image.

Step 1. From the Main Configuration window, select `Configure -> Build Image`. The **Master Builder** window appears.

Step 2. Select the **Coreboot + Bootfile** option.

Step 3. Select the `ROM Utility Set`, `User State Debugging Modules`, and the `SoftStax (SPF) Support` boxes under the **Include** options.

Step 4. Click `Build`. It should display progress information and show the statistics of the image just created.

Step 5. Click `Save As`. The `rom` file is created in the following directory:

```
MWOS/OS9000/603/PORTS/MOTRAVEN/BOOTS/INSTALL/PORTBOOT
```

Step 6. Click `Save`. The Master Builder window is displayed. At this point you can either close the Configuration Wizard or leave it open for use in the **Booting Your Reference Board from Flash** section. If you choose to close it, you can save your configuration settings for later use.

Transferring the ROM Image to the Target

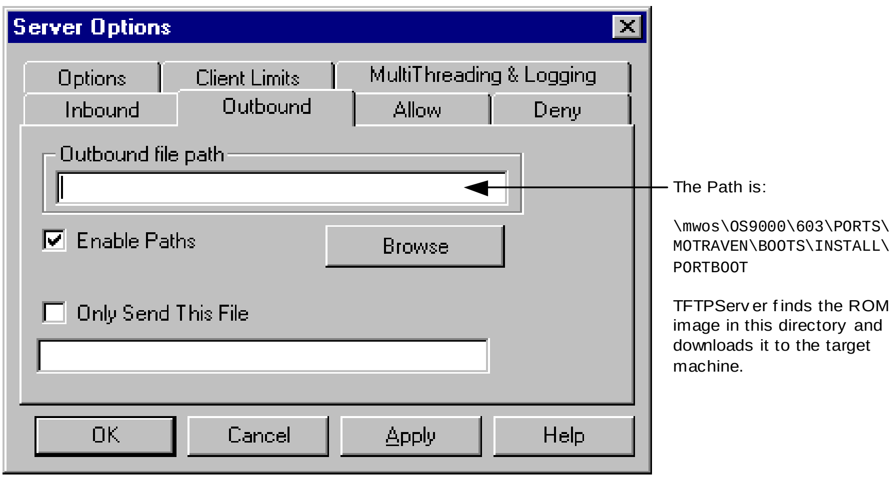
Configure the TFTP Server

`TFTPServer32` is the Trivial File Transfer Protocol (TFTP) server utility that must be installed on your PC host from the Enhanced OS-9 for PowerPC CD. This is the tool you will use to transfer the ROM image from your host system to the target system. Perform the following steps to configure the TFTP server:

Step 1. On the Windows desktop click `Start -> Programs -> TFTPServer -> TFTPServer32`.

Step 2. Select `System -> Setup` and click the `Outbound` tab. Indicate the path to where the ROM image is located in the Outbound File Path box.

Figure 1-7 TFTP Server Options Window



Step 3. Use default settings for all other settings.

Step 4. Apply the changes and click OK to exit TFTP Server Pro.

Boot the Target from an Ethernet Network

The MVME has Ethernet built into the transition module. Use the following procedure to set up the board to work on an Ethernet network.

Step 1. Check that your Ethernet network connection is operational.

On your host desktop, click on the Network Neighborhood icon. If you can see other computers (or at least your own) on the network your Ethernet connection is functional.

Step 2. From the host system, bring up your Hyperterminal session as described in **Connecting the Target to the Host**.

Note

Although not required, you can use the env command to set up the nbo option as an autobooter. At the PPC1-Bug> prompt, type env. Your screen should display the following.

```
PPC1Bug> env
Network Auto Boot Enable [Y/N]            = Y?
Network Auto Boot at power-up only [Y/N] = Y?
Network Auto Boot Controller LUN          = 00?
Network Auto Boot Device LUN              = 00?
Network Auto Boot Abort Delay             = 5?
Network Auto Boot Configuration Parameters Offset (NVRAM) = 00001000?
```

Step 3. At the `PPC1-Bug>` prompt type **nbo**. This command transfers the ROM image from the host system to the target system and boots the target. Your screen should display the following:

```
PPC1-Bug>nbo
Network Booting from: DEC21140, Controller 0, Device 0
Device Name: /pci@80000000/pci1011,9@e,0:0,0
Loading: rom

Client IP Address      = 172.16.4.108
Server IP Address      = 172.16.4.56
Gateway IP Address     = 172.16.1.254
Subnet IP Address Mask = 255.255.0.0
Boot File Name         = rom
Argument File Name     =

Network Boot File load in progress... To abort hit <BREAK>

Bytes Received =&1652544, Bytes Loaded =&1652544
Bytes/Second   =&206568, Elapsed Time =8 Second(s)


OS-9000 Bootstrap for the PowerPC(tm)

Now trying to Override autobooters.
Now trying to Boot embedded OS-9000 in-place.
Now searching memory ($000b7e20 - $0021373f) for an OS-9000 Kernel...

An OS-9000 kernel was found at $000b7e20
A valid OS-9000 bootfile was found.
+3
$
```

Your target system should now display the $ OS-9 prompt.

Creating a Startup File

When the Configuration Wizard is set to use a hard drive, or another fixed drive such as a PC Flash Card, as the default device, it automatically sets up the init module to call the `startup` file in the SYS directory in the target (For example: `/h0/SYS/startup`, `/mhc1/SYS/startup`). However, this directory and file will not exist until you create it. To create the startup file, complete the following steps:

Step 1. Create a SYS directory on the target machine where the `startup` file will reside (for example: `makdir /h0/SYS`, `makdir /dd/SYS`).

Step 2. On the host machine, navigate to the following directory:

`MWOS/OS9000/SRC/SYS`

In this directory, you will see several files. The files related to this section are listed below:

- `motd`: Message of the day file
- `password`: User/password file
- `termcap`: Terminal description file
- `startup`: Startup file

Step 3. Transfer all files to the newly created SYS directory on the target machine. (You can use Kermit, or FTP in ASCII mode to transfer these files.)

Step 4. Since the files are still in DOS format, you will be required to convert them into the OS-9 format with the `cudo` utility. The following command is an example:

```
cudo -cdo password
```

This will convert the `password` file from DOS to OS-9 format.

For More Information

For a complete description of all the `cudo` command options, refer to the ***Utilities Reference Manual*** located on the Enhanced OS-9 CD.

Step 5. Since the command lines in the startup file are system-dependent, it may be necessary to modify this file to fit your system configuration. It is recommended that you modify the file before transferring it to the target machine.

Example Startup File

Below is the example startup file as it appears in the `MWOS/OS9000/SRC/SYS` directory:

```
-tnxnp
tmode -w=1 nopause
*
*OS-9 - Version 3.0
*Copyright 2001 by Microware Systems Corporation
*The commands in this file are highly system dependent and
*should be modified by the user.
*
*setime </term                ;* start system clock
setime -s                     ;* start system clock
link mshell csl               ;* make "mshell" and "csl" stay in memory
* iniz r0 h0 d0 t1 p1 term    ;* initialize devices
* load utils                  ;* make some utilities stay in memory
* tsmon /term /t1 &           ;* start other terminals
list sys/motd
setenv TERM vt100
tmode -w=1 pause
mshell<>>>/term -l&
```

For More Information

Refer to the **Making a Startup File** section in Chapter 9 of the ***Using OS-9*** manual for more information on startup files.

Optional Procedures

The following section provides optional procedures you can perform after installing and configuring OS-9 on your board.

Preliminary Testing

Once you have established an OS-9 prompt on your target system, you can perform the following procedures to test your system:

Step 1. Type `mdir` at the prompt.

`mdir` displays all the modules in memory.

Step 2. Type `procs` at the prompt.

`procs` displays the processes currently running in the system.

Step 3. Test the networking on your system.

Select a host on the Ethernet network and run the `ping` utility. The following display shows a successful `ping` to a machine called solkanar.

```
$ ping solkanar
PING solkanar.microware.com (172.16.2.51): 56 data bytes
64 bytes from 172.16.2.51: ttl=128 time=0 ms
```

Step 4. Test `telnet`.

Select a host machine that allows telnet access and try the OS-9 `telnet` utility. The following display shows a successful `telnet` to a machine called delta.

```
$ telnet delta
Trying 172.16.1.40...Connected to delta.microware.com.
Escape character is '^]'.
capture closed.
```

```
OS-9/68K V3.0.3 Delta VME177 - 68060 98/12/24 14:41:51
User name?: curt
Password:
Process #101 logged on 98/12/24 14:41:56
Welcome!
************************************************************
*      WELCOME TO DELTA - THE :OS-9 68K: MACHINE *
```

Step 5. Test telnet from your host PC to the reference board.

From the Windows Start menu, select Run and type `telnet` <`hostname`> and click `OK`. A telnet window should display with a $ prompt. Type `mdir` from the prompt. You should see the same module listing as on the serial console port.

You have now created your OS-9 boot image and established network connectivity with your OS-9 target system.

Booting Your Reference Board from Flash

Once the ROM image is built and loaded onto the target system, you can copy the ROM image to Flash memory or to a disk. This enables you to boot the target without using a network. This section describes booting the target from Flash or a disk.

To boot the target system from Flash, you must return to the configuration wizard and rebuild the ROM image.

WARNING

Follow the steps below carefully. During this procedure it is possible to overwrite the manufacturer's original Flash image. In this event, you will be required to return the hardware to the manufacturer.

Step 1. Open the OS-9 configuration wizard. Be sure to start with the same ROM image that you built in the **Building the OS-9 ROM Image** section.

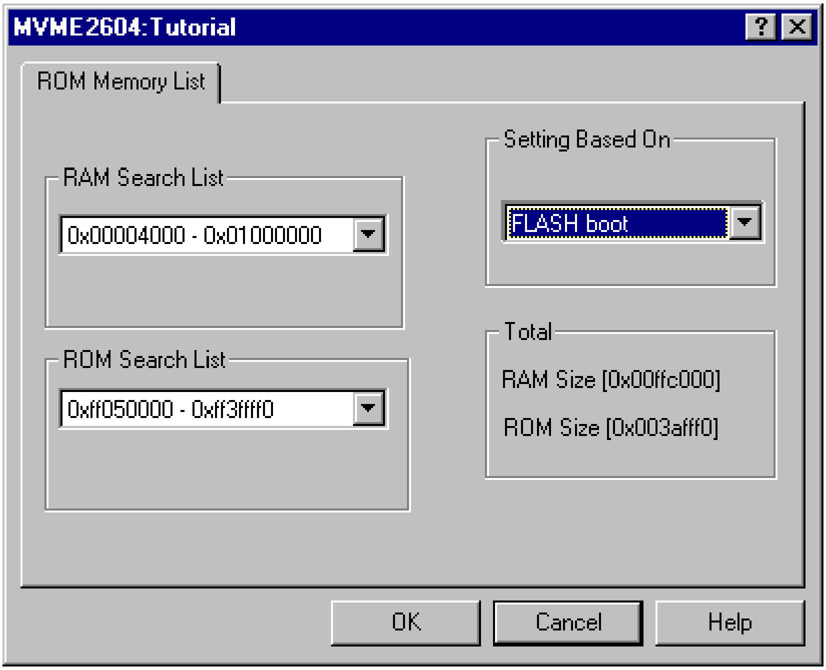
Step 2. Configure Flash booting options.

- Select `Configure -> Sys -> Select System Type` from the Main Configuration window.
- Select the **ROM Memory List** tab.
- Select Flash Boot from the **Settings Based On** pulldown menu. **Figure 1-8** shows this configuration.

Note

This example uses the MVME2604 as the reference board.

Figure 1-8 ROM Memory List

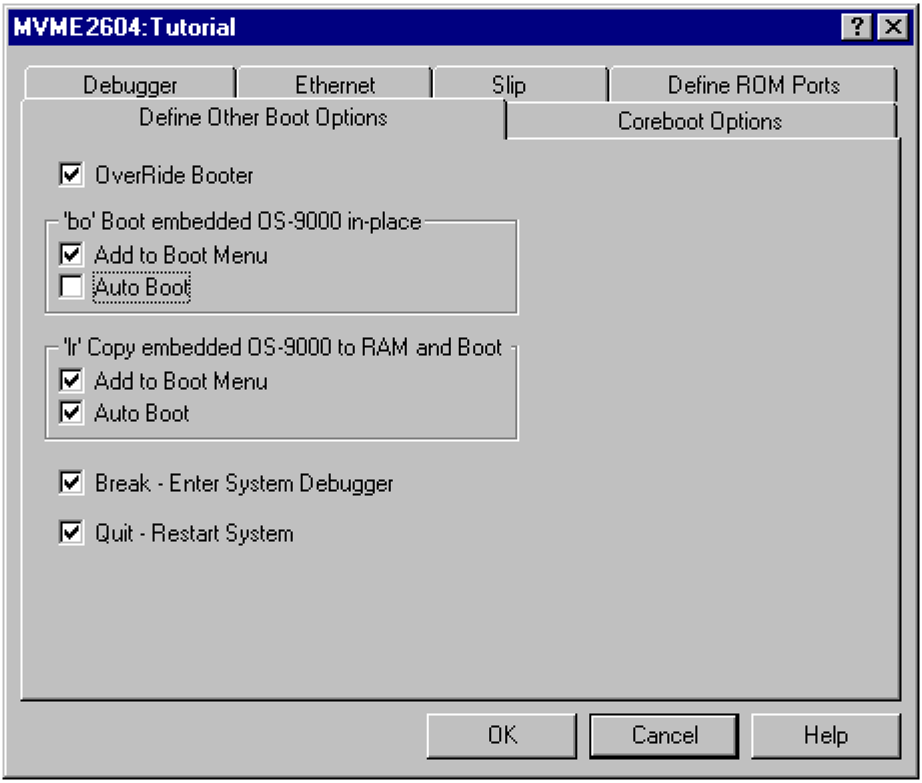


Step 3. Set the `lr` option.

The `lr` option moves the boot image modules from Flash to RAM before booting begins. This is optional. However, the Flash device is very slow and using the `lr` option is highly recommended.

- Select `Configure -> Coreboot -> Main Configuration` from the **Main Configuration** window.
- Select the **Define Other Boot Options** tab.
- Configure the tab according to **Figure 1-9**.

Figure 1-9 Setting the `lr` Option



Step 4. Rebuild the ROM image.

- From the **Main Configuration** window, select `Configure -> Build Image`. The **Master Builder** window appears.
- Do not change the settings
- Click `Build`. Progress information is displayed and the statistics of the image just created are shown.
- Click `Save As` to save the image. The file rom is saved in the following directory:

  `MWOS/OS9000/603/PORTS/MOTRAVEN/BOOTS/INSTALL/PORTBOOT.`

Step 5. From the host system, bring up your Hyperterminal session as described in **Connecting the Target to the Host**.

- At the `PPC1-Bug>` prompt, type the `niot` command as described in **Connecting the Target to the Host**
- At the `PPC1Bug>` prompt, type the `niop` command to transfer the ROM image from the host system to the target system. Your screen should display the following:

```
PPC1-Bug>niop
Controller LUN =00?
Device LUN     =00?
Get/Put        =G?
File Name      =? rom
Memory Address =00004000?
Length         =00000000?
Byte Offset    =00000000?

Bytes Received =&1652584, Bytes Loaded =&1652584
Bytes/Second   =&236083, Elapsed Time =7 Second(s)
```

Step 6. At the `PPC1-Bug>` prompt, type `pflash` to program the ROM image into the target system's Flash memory.

WARNING

Make sure the jumper settings for your board are correct. The memory at 0xff000000 must be the 4MB or 8MB FLASH image not the 1MB image where PPC1Bug is located. Failure to set up the board correctly can cause the PPC1Bug image to be erased resulting in a non-working board.

Step 7. Adjust the number of bytes received to a block boundary.

```
PPC1-Bug>pflash 4000:1937f0 ff000000;b
Source Starting/Ending Addresses      =00004000/001977EF
Destination Starting/Ending Addresses =FF000000/FF1937EF
Number of Effective Bytes             =001937F0 (&1652720)

Program FLASH Memory (Y/N)? y
```

Note

If the last two digits in HEX are less than FO, change them to FO. If the last two digits are greater than FO, add 100_{16} to that number and change the last two digits to FO. Following is an example:

```
&1909180 = 0x1D21BC
round = 0x1D21F0
```

The image should now be in the 0xff000000 section.

Step 8. From the PPC1-Bug> prompt, type env. This indicates to PPC1-Bug where the ROM image is located.

```
PPC1Bug> env
ROM Boot Enable [Y/N]            = Y?
ROM Boot at power-up only [Y/N]  = N?
ROM Boot Abort Delay             = 1?
ROM Boot Direct Starting Address = FF000260?
ROM Boot Direct Ending Address   = FF000260?
```

The above sequence will set up the system to autoboot using the ROM image. You can also use the `rb` command from the `PPC1-Bug>` to boot the target from ROM.

Note

The `coreboot` file can be placed in Flash without the `bootfile` file. This can be desirable if disk booting or eb BOOTP booting. You must supply your own BOOTP server.

Disk Booting RBF

Once you have booted your system from an Ethernet Network and configured your SCSI hard drive, you can use the following procedure to transfer the coreboot and bootfile images to the target machine.

Note

A method for transferring the ROM image using TFTP is described in the **Transferring the ROM Image to the Target** section.

For More Information

Refer to Appendix B: Partitioning and Formatting Your Hard Drive for hard drive formatting and partitioning procedures.

Step 1. At the $ prompt (the OS-9 prompt), create the ROM image by typing the following commands:

```
bootgen -el=coreboot /hs01fmt
```

This command places the TYPE41 boot image on SCSI hard drive.

```
bootgen /hs01fmt bootfile -nb400
```

This command places the high-level boot image on the system disk.

Note

The bootfile and coreboot file are located at:

`<drive>:\MWOS\OS9000\603\PORTS\MOTRAVEN\BOOTS\INSTALL\PORTBOOT.`

Step 2. At the PPC1-Bug> prompt, type `pboot 0` to boot the target system. Your screen should display similar to the following:

```
Booting from: NCR53C810, Controller 0, Drive 0
Loading: Operating System

IPL Loaded at: $01F30000
Residual-Data Located at: $01F84000


OS-9000 Bootstrap for the PowerPC(tm)

Now trying to Override autobooters.

BOOTING PROCEDURES AVAILABLE ---- <INPUT>
Scan SCSI devices --------------- <ioi>
Boot FDC floppy ----------------- <fd>
Boot from PC-Floppy ------------- <pf>
Boot from Viper tape drive ------ <vs>
Boot over Ethernet -------------- <eb>
Boot from SCSI(SCCS) hard drive - <hs>
Boot embedded OS-9000 in-place -- <bo>
Kermit download ----------------- <ker>
PCI View Utility ---------------- <pciv>
Enter system debugger ----------- <break>
Restart the System -------------- <q>
```

Step 3. Select a boot method from the above menu. In this case, enter `hs`.

Your screen should display similar to the following:

```
Symbios 53C810 @ 0x81000000 SELFID (07) MAXCNT (0x01000000)
ID (00) LUN (00) SI (00) EI (03) LSNOFFS (00000804)

Checking Partitions         : 0
Volume Name                 : os9000
FD bootfile block offset    : 0x00000997

Booting from partition
Reading Bootfile......

Boot Address                : 0x00300000
Boot Size                   : 0x00180a00

OS-9000 kernel was found.
A valid OS-9000 bootfile was found.
+3
$
```

Chapter 2: Board Specific Reference

This chapter contains information that is specific to the Motorola MVME reference boards. It contains the following sections:

- **Boot Menu Options**
- **Vector Descriptions for PowerPC 603/604**
- **Configuring Booters**
- **Port Specific Utilities**
- **PowerPC™ Registers Passed to a New Process**

For More Information

For general information on porting OS-9, see the ***OS-9 Porting Guide***.

Boot Menu Options

You select your boot device menu options using the configuration wizard. For each boot device option, you can select whether you want it to be displayed on a boot menu, set up to autoboot, or both. The autoboot option enables the device selected to automatically boot up the high-level bootfile, bypassing the boot device menu.

Note

When using the Configuration Wizard, you should select only one device for autoboot on your system.

Following is an example of the Boot Menu displayed in the terminal emulation window (using `Hyperterminal`):

```
OS-9000 Bootstrap for the PowerPC(tm)

Now trying to Override autobooters.

BOOTING PROCEDURES AVAILABLE ----- <INPUT>

Scan SCSI devices ---------------- <ioi>
Boot FDC floppy ------------------ <fd>
Boot from PC-Floppy -------------- <pf>
Boot from Teac SCSI floppy drive - <fs>
Boot from SCSI PC-Floppy --------- <pfs>
Boot from Viper tape drive ------- <vs>
Boot over Ethernet --------------- <eb>
Boot from SCSI(SCCS) hard drive -- <hs>
Boot embedded OS-9000 in-place --- <bo>
Enter system debugger ------------ <break>
Restart the System --------------- <q>

Select a boot method from the above menu:
```

What you select for boot options in the configuration wizard determines what modules are included in the coreboot image. **Table 2-1** lists some of the supported boot devices for OS-9:

Table 2-1 Supported Boot Methods

Type of Boot	Description
Boot from RBF hard disk	Boot from a standard SCSI hard disk (`hs`).
Floppy Disk	Boot from floppy disk. You must select if the floppy is controlled by a Random Block File System (RBF) (`fd` or `fs`) or PC File System (`pf` or `pfs`).
Boot embedded OS-9 in-place	Boot OS-9 from FLASH (`bo`).
Copy embedded OS-9 to RAM and Boot	Copy OS-9 from FLASH (if stored there) to RAM and boot (`lr`).

Vector Descriptions for PowerPC 603/604

Table 2-2 Vector Descriptions for PowerPC 603/604

Vector Number	Related OS-9 Call	Assignment
0	None	Reserved
1	F_IRQ	System reset
2	F_STRAP, F_IRQ	Machine check
3	F_STRAP, F_IRQ	Data access
4	F_STRAP, F_IRQ	Instruction access
5	F_IRQ	External interrupt
6	F_STRAP, F_IRQ	Alignment
7	F_STRAP, F_TLINK, F_IRQ	Program
8	F_IRQ	Floating-point unavailable
9	F_IRQ	Decrementer
10	None	Reserved
11	None	Reserved
12	F_SSVC	System call
13	None	Trace
14	None	Reserved

Table 2-2 Vector Descriptions for PowerPC 603/604 (continued)

Vector Number	Related OS-9 Call	Assignment
15	None	Reserved
	F_IRQ	Performance monitoring interrupt (604e)
16	None	Instruction translation miss
	None	Reserved (604e)
17	None	Data load translation miss
	None	Reserved (604e)
18	None	Data store translation miss
	None	Reserved (604e)
19	F_IRQ	Instruction address breakpoint
20	F_IRQ	System management interrupt
21-47	None	Reserved

Note

The vector numbers in Table 2-2 are logical vector numbers. The actual processor vectors can be computed by multiplying the logical vector number by 256.

Error Exceptions: vectors 2-4 and 6-7

These exceptions are usually considered fatal program errors and unconditionally terminate a user program. If `F_DFORK` create the process or the process was debug attached with `F_DATTACH`, then the resources of the erroneous process remain intact and control returns to the parent debugger to allow a postmortem examination.

A user process may use the `F_STRAP` system call to install an exception handler to catch the errors and recover from the exceptional condition. When a recoverable exception occurs, the process' exception handler installed with the `F_STRAP` system call is executed with a pointer to the process' normal static data and the current stack pointer. Also, the process' exception handler will receive as parameters the vector number of the error, the program instruction counter of where the error occurred, and the fault address of the error if applicable. The exception handler must decide whether and where to continue execution. Programs written in the C language may use the `setjmp` and `longjmp` library routines to properly recover from the erroneous condition.

If any of these exception occur in system state during a system call made by the process due to the process passing bad data to the kernel, the process' exception handler is not called. Instead, the appropriate vector error is returned from the system call.

Vectored Interrupts: vector 5

In general, the PowerPC processor family uses a single interrupt vector for all external interrupts. However, most systems supporting the PowerPC family use additional external logic to support more powerful nested interrupt facilities. Hence, the vector numbers used by OS-9000 device drivers are usually logical vectors outside of the range of the hardware vectors listed above. The device drivers install their interrupt service routines, via the `F_IRQ` system call, on the logical vector and the kernel's dispatch code uses the external logic vector to identify the source of the interrupt and call the associated interrupt service routine. Interrupt service routines are executed in system state without an associated current process.

Note

The F_IRQ system call may also be used to install exception handlers on some non-hardware interrupt vectors. The above table lists the exceptions that may be monitored using the F_IRQ facility. The installed exception handler is called just like any other interrupt service routine when the associated exception occurs.

User Trap Handlers: vector 7

This vector is used for dispatching user code into system state trap handlers. The vector provides a mechanism for programs to switch states and dispatch to a subroutine module to execute code in system state.

System Calls: vector 12

This vector is used for service call dispatching to the OS-9 operating system as well as user services installed using the F_SSVC service request.

Configuring Booters

The following booters are available for the MVME2603/MVME2604 and MVME3603/MVME3604 target platforms. The abbreviated name and configuration parameters for the booters are listed with recommended values (if any).

Table 2-3 MVME260X/360X Booters

Booter	Description	Recommended Values
`fdc765`	Standard floppy disk booter	
	Abbreviated name:	`"fd"`

	Configuration parameters:	`"port=0x800003f0"`
		`"lun=0"`
		`"si=0"`
		`"ei=3"`

`fsboot`	TEAC SCSI floppy disk booter	
	Abbreviated name:	`"fs"`

	Configuration parameters:	`"port=0xff000000"`
		`"device=ncr8xx"`
		`"id=6"`
		`"si=0"`
		`"ei=3"`

Table 2-3 MVME260X/360X Booters

Booter	Description	Recommended Values
`hsboot`	SCSI hard disk booter	
	Abbreviated name:	"hs"
	Configuration parameters:	"port=0xff000000" "device=ncr8xx" "id=<default scsi ID>" "si=0" "ei=3" "lsnoffs=2052"
`ide`	Standard IDE hard disk booter	
	Abbreviated name:	"ide"
	Configuration parameters:	"port=0x800001f0" "si=0" "ei=3" "lsnoffs=2052"
`llbootp`	Standard BOOTP booter	
	Abbreviated name:	"eb"
	Configuration parameters	"driver=ll21040"

Table 2-3 MVME260X/360X Booters

Booter	Description	Recommended Values
`romboot`	Embedded system booter	
	Abbreviated name:	`"ro" (reconfigured to "bo" and "lr")`
	Configuration parameters:	`<none>`
`vsboot`	SCSI tape booter	
	Abbreviated name:	`"vs"`

	Configuration parameters:	`"port=0xff000000"`
		`"device=ncr8xx"`
		`"id=4"`

`ioi`	SCSI Bus diagnostic tool	
	Abbreviated name:	`"ioi"`

	Configuration parameters:	`"port=0xff000000"`
		`"device=ncr8xx"`
		`"reset=1"`

Port Specific Utilities

The following port specific utilities are included:

- `dmppci`
- `mouse`
- `pciv`
- `setpci`
- `testpci`

`dmppci` Show PCI Information

SYNTAX

```
dmppci <bus_number> <device_number>
        <function_number> {<size>}
```

OPTIONS

`-?`	Display help

DESCRIPTION

`dmppci` displays PCI configuration information that is not normally available by other means, except programming, using the PCI library.

EXAMPLE

```
$ dmppci 0 11 1 0x40
    PCI DUMP Bus:0 Dev:11 Func:1 Size:64
    -----------------------------------
VID  DID  CMD  STAT CLASS  RV CS IL IP LT HT BI MG ML SVID SDID
---  ---- ---- ---- -----  -- -- -- -- -- -- -- -- -- ---- ----
10ad 0105 0005 0280 01018f 05 08 0e 01 00 80 00 02 28 0000 0000

BASE[0]  BASE[1]  BASE[2]  BASE[3]  BASE[4]  BASE[5]  CIS_P    EXROM
-------- -------- -------- -------- -------- -------- -------- --------
01000321 01000331 01000329 01000335 01000301 01000311 00000000 00000000

Offset 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
------------------------------------------------------
0000   ad 10 05 01 05 00 80 02 05 8f 01 01 08 00 80 00
0010   21 03 00 01 31 03 00 01 29 03 00 01 35 03 00 01
0020   01 03 00 01 11 03 00 01 00 00 00 00 00 00 00 00
0030   00 00 00 00 00 00 00 00 00 00 00 00 0e 01 02 28
```

mouse

Show Mouse Library Functions

SYNTAX

```
mouse <opts>
```

OPTIONS

`-?`	Display help
`-s`	Slow mouse
`-f`	Fast mouse
`-r[n]`	Set resolution to n
`-p[n]`	Set sample rate to n
`-c[n]`	Set scale factor to n

DESCRIPTION

`mouse` displays mouse status information.

EXAMPLE

```
$ mouse
Opening device /m0
status = 0x08, x =   4, y =   0
status = 0x08, x =   6, y =   0
status = 0x08, x =   7, y =   1
status = 0x08, x =   7, y =   1
status = 0x08, x =   8, y =   1
status = 0x08, x =   7, y =   0
status = 0x28, x =   7, y = 255  Y Negative
status = 0x28, x =   7, y = 254  Y Negative
status = 0x28, x =   5, y = 254  Y Negative
status = 0x08, x =   2, y =   0
status = 0x28, x =   1, y = 255  Y Negative
status = 0x08, x =   2, y =   0
status = 0x28, x =   0, y = 255  Y Negative
status = 0x08, x =   1, y =   0
status = 0x09, x =   0, y =   0  Left Button
status = 0x08, x =   0, y =   0
status = 0x0a, x =   0, y =   0  Right Button
status = 0x08, x =   0, y =   0
```

pciv — PCI Configuration Space View

SYNTAX

```
pciv [<opts>]
```

OPTIONS

`-?`	Display help.
`-a`	Display base address information and size.
`-r`	Display PCI routing information.

DESCRIPTION

The `pciv` utility allows visual indication of the status of the PCIbus. This utility is port dependent.

EXAMPLES

When using the `pciv` command with a Motorola PowerPC board, the following information is displayed:

```
$ pciv

PowerPC 603 Configuration Report

Model:  Ultra PowerPC

Board Configuration Reports
[Z85230 ESCC] [PMC] [Graphics] [Ethernet] [SCSI]

BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:00:00  1057 0001 0106 2080 060000 24 00 00 00 MPC105
000:11:00  8086 0484 000f 0200 000000 84 00 00 00 PCI/ISA Bridge
000:12:00  1000 0001 0007 0200 010000 02 00 0b 01 NCR53C810 SCSI
000:14:00  1011 0002 0007 0280 020000 23 00 09 01 DECchip 21040
000:15:00  1013 00a8 0000 0000 030000 8e 00 0b 01 GD5434 Graphics
```

The following configuration registers apply to these DEV columns:

- 12 - NCR53C810 Configuration Register
- 14 - DECchip 21040 Configuration Register
- 15 - GD5434 Configuration Register

The `pciv` command in the previous example reports configuration information related to specific hardware attached to the system. The MVME2600 and MVME3600 series are specific about the PCI devices located on the main board. For this reason, the information displayed is not generic in format.

```
DETAIL OF BASIC VIEW:
   BUS       : Bus Number
   DEV       : Device Number
   VID       : Vendor ID
   DID       : Device ID
   CLASS     : Class Code
   RV        : Revision ID
   IL        : Interrupt Line
   IP        : Interrupt Pin
   [S]       : Single function device
   [M]       : Multiple function device
```

When the `-a` option is used address information is also displayed as well as the size of the device blocks being used. All six address PCI address entries are scanned.

```
(C) [32-bit] base_addr[0] = 0x3efefe81  PCI/IO
                  0xbefefe80 Size = 0x00000080
```

The fields in the previous example are, from left to right, as follows:

- prefetchable
- memory type
- address fields
- actual value stored
- type of access
- translated access address used (shown on second line)
- size of block (shown on second line)

When the `-r` option is used, PCI-specific information related to PCI interrupt routing is displayed. If an ISA BRIDGE controller is found in the system, the routing information is used. The use of ISA devices and PCI devices in the same system requires interrupts to be routed either to ISA or PCI devices. Since ISA devices employ edge-triggered interrupts and PCI use devices use level interrupts, the `EDGE/LEVEL` control information is also displayed. If an interrupt is shown as `LEVEL` with a PCI route associated with it, no ISA card can use that interrupt. This command also shows the system interrupt mask from the interrupt controller.

Note

ISA and PCI interrupts cannot be shared.

setpci

Set PCI Value

SYNTAX

```
setpci <bus> <dev> <func> <offset> <size{bwd}>
<value>
```

OPTIONS

`-?`	Display help

DESCRIPTION

The `setpci` utility sets PCI configuration information that is not normally available by other means other than programming using the PCI library. The `setpci` utility may also be used to read a single location in PCI space. Parameters include:

`<bus>`	= PCI Bus Number 0..255
`<dev>`	= PCI Device Number 0..32
`<func>`	= PCI Function Number 0..7
`<offset>`	= Offset value (ie. command register offset = 4)
`<size>`	= Size b=byte w=word d=dword
`<value>`	= The value to write in write mode. If no value is included, the utility is in read mode.

EXAMPLES

```
$ setpci 0 19 0 0x14 d

PCI READ MODE
-------------

PCI Value.....0x3bfedd00 (dword) READ

PCI Bus.........0x00
PCI Device......0x13
PCI Function....0x00
PCI Offset....0x0014

$ setpci 0 19 0 0x14 d 0x1234500

PCI WRITE MODE
--------------

PCI Value.....0x01234500 (dword) WRITE

PCI Bus.........0x00
PCI Device......0x13
PCI Function....0x00
PCI Offset....0x0014
$
$ setpci 0 19 0 0x14 d

PCI READ MODE
-------------

PCI Value.....0x01234500 (dword) READ

PCI Bus.........0x00
PCI Device......0x13
PCI Function....0x00
PCI Offset....0x0014
```

testpci — Test PCI Value

SYNTAX

```
testpci
```

OPTIONS

`-?`	Display help

DESCRIPTION

The `testpci` utility tests all PCI library functions. To use this utility, you must have a graphics card in the system. This utility shows how the PCI library calls can be used.

EXAMPLE

```
$ testpci
Test PCI Library Calls Edition 2
_pci_search_device ........................ok....
_pci_next_device ..........................ok....
_pci_get_config_data ......................ok....
_pci_find_device ..........................ok....
_pci_find_class_code ......................ok....
_pci_read_configuration_byte ..............ok....
_pci_read_configuration_word ..............ok....
_pci_read_configuration_dword .............ok....
_pci_write_configuration_byte .............ok....
_pci_write_configuration_word .............ok....
_pci_write_configuration_dword ............ok....
_pci_get_irq_pin ..........................ok....
_pci_get_irq_line .........................ok....
_pci_set_irq_line .........................ok....
PCI LIBRARY TEST CONTAINS NO ERRORS.
```

PowerPC™ Registers Passed to a New Process

The following PowerPC registers are passed to a new process (all other registers are zero):

```
r1  = stack pointer
r2  = static storage (data area) base pointer
r3  = points to fork parameters structure (listed in
                                            f_fork)
r13 = points to the constant data of code area of the
                                            module
```

Note

r2 is always biased by the amount specified in the `m_dbias` field of the program module header which allows object programs to access a larger amount of data using indexed addressing. You can usually ignore this bias because the OS-9 linker automatically adjusts for it.

Appendix A: Board Specific Modules

This appendix contains lists of high and low-level modules. The following sections are included:

- **Low-Level System Modules**
- **High-Level System Modules**
- **Common Modules List**

Low-Level System Modules

The following low-level system modules are tailored specifically for the MVME target platforms. These modules can be found in the following directory:

`MWOS/OS9000/603/PORTS/MOTRAVEN/CMDS/BOOTOBJS/ROM`

Configuration Modules

`cnfgdata`	provides low-level configuration data including configuration of a serial console.
`cnfgfunc`	retrieves configuration parameters from the `cnfgdata` module.
`commcnfg`	retrieves the name of the low-level auxiliary communication port driver from the `cnfgdata` module.
`conscnfg`	retrieves the name of the low-level console driver from the `cnfgdata` module.

Console Drivers

`io16550`	provides console services for the external 16550 serial ports.
`io8042`	provides console services for the VGA display and keyboard interface (when available).
`io85x30`	provides console services for the 82530 serial ports (when available).

Debugging Module

`usedebug`	is a debugger configuration module.

Ethernet Driver

`ll21040`	provides network driver services for the DEC 21040 Ethernet port.

SCSI Host Adapter Support Booter Module

`ncr8xx`	provides the booter subsystem with SCSI host adapter services for both the NCR 53C810 and 53C825 interfaces.

System Modules

`ide`	is a low-level IDE booter module.
`initext`	is a user-customizable system initialization module.
`portmenu`	retrieves a list of configured booter names from the ROM `cnfgdata` module.
`romcore`	gives bootstrap code.
`rpciv`	shows information about devices on the PCI bus.

Timer Module

`swi8timr`	provides polling timer services with a software loop self-calibrated from the 8259-like timer.

High-Level System Modules

The following OS-9 system modules are tailored specifically for MVME series platforms. Unless otherwise specified, each module can be found in the following directory:

`MWOS/OS9000/603/PORTS/MOTRAVEN/CMDS/BOOTOBJS`

Interrupt Controllers

These modules provide extensions to the vectors module by mapping the single interrupt generated by an interrupt controller into a range of pseudo vectors which are recognized by OS-9 as extensions to the base CPU exception vectors.

`picirq` provides interrupt acknowledge and dispatching support for the nested 8259 interrupt controllers on the MVME2603/2604 series platforms. Maps the nested PIC interrupts 0-15 to OS-9 pseudo vectors 64-79 ($40-$4f).

The `picirq` module used in the sample boots is located in the file also called `picirq`. It provides slightly lower performance, but allows use of the last set of BAT registers for ISA memory access. This is the default configuration, as it supports a wider range of platforms.

`universeirq`	provides interrupt acknowledge and dispatch support for the Tundra Universe (CA91C042) chip implemented on the MVME2603/2604 series of CPU boards. Use this module together with the proper `picirq` module, if you require access to VME interrupts on one of these platforms. `universeirq` maps VME interrupts 64-255 to OS-9 pseudo-vectors 64-255 ($40-$ff).
`ravenirq`	provides interrupt acknowledge and dispatch support.

Real Time Clock Driver

`rtc48t18`	provides OS-9 access to the M48T18 BBRAM real time clock. In this release, `rtc48t18` is the name of the ticker regardless of the CPU in use on your platform. This is likely to change in a future release.

Ticker

`tk8253`	provides the system ticker through the Intel 8253 programmable interval timer.

Abort Handler

`abort`	provides handler for the abort interrupt which calls into the system-state debugger. If no system state debugger is configured, the system will perform a soft reset.

Shared Libraries

`picsub`	provides interrupt enable and disable routines to handle platform-specific interrupt controller issues for device drivers. This module is called by all drivers and should be included in your `bootfile`.

Serial and Console Drivers

`sc16550`	provides support for the external 16550 serial ports. This driver is used to drive the console over the com1 port in the sample boots provided in the package. The descriptors provided for this driver are named `t1`, `t2`, `term_t1`, and `term_t2` and are located in the following directory: `MWOS/OS9000/603/PORTS/MOTRAVEN/ CMDS/BOOTOBJS/DESC/SC16550`
`sc85x30`	provides support for the 82530 serial ports (when available). The descriptors provided for this driver are named `t3`, `t4`, `term_3`, and `term_4` and are located in the following directory: `MWOS/OS9000/603/PORTS/MOTRAVEN/ CMDS/BOOTOBJS/DESC/SC85X30`
`sc8042`	provides unified support for the i8042 keyboard and VGA monitor output device (when available). The descriptors for this device are named `t0` and `term` and are located in the following directory: `MWOS/OS9000/603/PORTS/MOTRAVEN/ CMDS/BOOTOBJS/DESC/SC8042`

To configure your monitor as the high-level console, change the reference to the `term` device descriptor in the boot list used to build your system to point to this file instead of the 16550 `term` descriptor.

`sc8042k` — provides unified support for the i8042 keyboard and input device (mouse).

The descriptors provided for this driver are named `k0`, `kx`, and `m0` are located in files stored in the following directory:

MWOS/OS9000/603/PORTS/MOTRAVEN>/CMDS/BOOTOBJS/DESC/SC8042K

`sc8042m` — provides unified support for the multiple windowing version of the SC8042, keyboard, and graphics support in text mode using a standard VGA card and monitor.

The descriptors provided for this driver are named `term`, `mterm0`, `mterm1`, `mterm2`, and `mterm3`. For an explanation of the language versions available, see the previous note. The descriptors are located in files stored in the following directory:

MWOS/OS9000/603/PORTS/MOTRAVEN/CMDS/BOOTOBJS/DESC/SC8042M

Note

For each of the sc8042 keyboard descriptors, several language versions are provided including: French, United Kingdom, German, and Norwegian. The different language descriptors are named according to the same rules as shown in the example for the French i8042 keyboard descriptor: `k0_fr`.

Parallel Driver

`scp87303` provides support for the 87303 parallel port. The descriptor provided for this driver is named `p.lp1` and is located in the following directory:

```
MWOS/OS9000/603/PORTS/MOTRAVEN/
CMDS/BOOTOBJS/DESC/SCP87303
```

Data Disk Drivers

`rb765` is a device driver for floppy drive.

`rb1003` Provides support for IDE and EIDE drives up to 4GB. Many descriptors are provided for use with this driver. Among the descriptors provided are several modules named `h0` and `dd`. These descriptors are contained in files of unique names and located in the following directory:

```
MWOS/OS9000/603/PORTS/MOTRAVEN/
CMDS/BOOTOBJS/DESC/RB1003
```

SCSI support

The high-level SCSI command set drivers `rbsccs`, `rbteac`, and `sbscsi` are available to support the use of SCSI disk and tape devices in the following directory:

`MWOS/OS9000/PPC/CMDS/BOOTOBJS`

`scsi8xx`	provides SCSI host adapter services for both the NCR 53C810 and 53C825 interfaces. In this release, `scsi8xx` is the name of the ticker regardless of the CPU in use on your platform. This is likely to change in a future release.

Common Modules List

The following low-level system modules provide generic services for OS9000 modular ROM. They are located in the following directory:

`MWOS/OS9000/PPC/CMDS/BOOTOBJS/ROM`

Table 2-4 Common System Modules List

Module	Description
`bootsys`	provides booter services.
`console`	provides high-level I/O hooks into low-level console serial driver.
`dbgentry`	provides hooks to low-level debugger server.
`dbgserv`	is a debugger server module.
`excption`	is a service module.
`fdc765`	provides PC style floppy support.
`fdman`	is a target-independent booter support module providing general booting services for RBF file systems.
`flboot`	is a SCSI floptical drive disk booter.
`flshcach`	provides the cache flushing routine.
`fsboot`	is a SCSI TEAC floppy disk drive booter.
`hlproto`	allows user-state debugging.
`hsboot`	is a SCSI hard disk drive booter.

Table 2-4 Common System Modules List (continued)

Module	Description
`ide`	provides target-specific standard IDE support, including PCMCIA ATA PC cards.
`iovcons`	is a hardware independent virtual console driver that provides a telnetd-like interface to the low-level system console.
`llbootp`	is a target-independent BOOTP protocol booter module.
`llip`	is a target-independent internet protocol module.
`llkermit`	is a kermit booter (serial down loader).
`llslip`	is a target-independent serial line internet protocol module. This modules uses the auxiliary communications port driver to perform serial I/O
`lltcp`	is a target-independent transmission control protocol module.
`lludp`	is a target-independent user datagram protocol modules.
`notify`	coordinates use of low-level I/O drivers in system and user-state debugging.
`override`	enables overriding of the autobooter. If the space bar is pressed within three seconds after booting the target, a boot menu is displayed. Otherwise, booting proceeds with the first autobooter.

Table 2-4 Common System Modules List (continued)

Module	Description
`parser`	parses key fields from the `cnfgdata` module and the user parameter fields.
`pcman`	is a target-independent booter support module providing general booting services for PCF file systems (PC FAT file systems).
`protoman`	is a target-independent protocol module manager. This module provides the initial communication entry points into the protocol module stack.
`restart`	restarts boot process.
`romboot`	locates the OS-9 bootfile in ROM, FLASH, NVRAM.
`rombreak`	enables break option from the boot menu.
`rombug`	is a debugger client module.
`scsiman`	is a target-independent booter support module that provides general SCSI command protocol services
`sndp`	is a target-independent system-state network debugging protocol module. This module acts as a debugging client on the target, invoking the services of dbgserv to perform debug tasks.
`srecord`	receives a Motorola S-record format file from the communications port and loads it into memory.
`swtimer`	is a software timer.

Table 2-4 Common System Modules List (continued)

Module	Description
`tsboot`	is a SCSI TEAC tape driver booter.
`type41`	is a primary partition type.
`vcons`	provides the console terminal pathlist.
`vsboot`	is a SCSI archive viper tape drive booter.

Appendix B: Partitioning and Formatting Your Hard Drive

This appendix explains how to partition and format your hard drive with one primary partition on your target system.

Partitioning Your Hard Drive

This section explains how to partition your hard drive using the `fdisk` command. The `fdisk` command displays and alters the partition table. You should format your hard drive after you have partitioned it.

Note

Although OS-9 can be used without disk partitions, the use of partitions is strongly recommended, even if only one partition is used. You cannot perform hard disk booting if you do not partition your hard disk.

Note

OS-9 uses extended type41 partitions using the Random Block File Manager (RBF) file system.The `fdisk` utility used to create partitions allows a maximum of four primary partitions to be created. For information on how to create more than one primary partition, refer to the ***Utilities Reference Manual***, located on the ***Enhanced OS-9*** CD.

To create a partition on your target system, use the following steps:

Step 1. Familiarize yourself with the `fdisk` command options and their uses, as listed in **Table B-1**.

Table B-1 `fdisk` Command Options

Option	Description
`-a [=] <num>`	Makes partition <num> the active partition.
`-d [=] <dev>`	Examines/changes device. Default = `/hc`.
`-c`	Forces terminal mode (cursers off).
`-e`	Includes partition information in display mode.
`-s`	Displays the partition table.

Step 2. At the OS-9 prompt, type `tmode nopause`. This allows you to view the entire `fdisk options` window after step 3.

Step 3. Create a partition using the `fdisk` utility. You must refer to the SCSI raw drive when using `fdisk`. The following descriptors are available when booting.

```
hs0fmt<----- SCSI ID 0
```

```
hs1fmt<----- SCSI ID 1
```

For example, to partition SCSI ID 1, you would enter the following command at the OS-9 prompt:

```
fdisk -d=/hs1fmt -e
```

Use the `-i` option to clear existing partitions from the board.

Note

You can determine the appropriate description of your SCSI driver from Wizard by selecting `Configure -> Bootfile -> Disk Configuration -> SCSI Configuration` tab.

Note

For a complete explanation of related device descriptors, see the ***OS-9 Porting Guide***.

Step 4. The following partitioning options display:

```
1. Create OS-9000 partition
2. Set Active Partition
3. Delete partition
4. Display partition information
5. Change extended DOS partition to OS-9000 partition
```

Note

If your hard drive already has a partition you want to delete, select 3.

For More Information

Refer to **OS-9 Partitioning Options** later in this Appendix for more information on how to delete a partition.

Step 5. Select `1. Create OS-9000 Partition`. A prompt appears asking you for the size of the partition you want (in cylinders). The default, shown in brackets, is the maximum amount of cylinders available for your partition on the hard drive. (You may have to hit `<return>` to view all the information).

Note

If you currently have a partition on the drive (such as DOS), the default size is the total number of remaining cylinders.

```
            Display Partition Information
Current fixed disk device: /hcfmt@
Partition  Status      Type    Start     End    Size

Enter the partition size in cylinders: [ 1022]
```

Note

It is important to note that one cylinder does not necessarily reflect 1MB. Enter the number of cylinders to allocate for the partition, not the number of bytes.

Step 6. The system determines the maximum amount of cylinders and uses this as the default selection.

If you want the partition to be a portion of the total number of cylinders, enter this number of cylinders instead.

Step 7. Hit `<return>`

Step 8. The following is displayed:

```
1. OS9000/386 type partition
2. Extended Type 41 partition

select partition type (1,2)...............:  [  ]
```

Step 9. Type 2 for Extended type 41 partition

Step 10. When the partitioning has completed, the display shows the display partition information screen:

```
1. Create OS-9000 partition
2. Set Active Partition
3. Delete partition
4. Display partition information
5. Change extended DOS partition to OS-9000 partition
```

Step 11. Hit `<esc>`

Step 12. The partitioning is now complete. To exit the `fdisk` utility and save the partition to the hard drive, hit the `<esc>` key. The following question is displayed:

```
Want to save new partition information (y/n)?
```

Step 13. Type `Y` to save the partition information to disk. You return to the OS-9 prompt.

Step 14. Move on to **Formatting Your Hard Drive**.

Formatting Your Hard Drive

Before you format your hard drive, make sure that it is partitioned correctly. See **Partitioning Your Hard Drive** in this Appendix for information on how to perform this task. This section explains how to format your hard drive using the `format` command.

For More Information

For a complete description of all the `format` command options, refer to the ***Utilities Reference Manual*** located on the ***Enhanced OS-9*** CD.

Step 1. Format the partitions using the correct descriptor for your hard drive. Descriptor options include the following:

```
hs01fmt---->SCSI ID=0 Partition = 1
hs02fmt---->SCSI ID=0 Partition = 2
hs03fmt---->SCSI ID=0 Partition = 3
hs04fmt---->SCSI ID=0 Partition = 4
hs11fmt---->SCSI ID=1 Partition = 1
hs12fmt---->SCSI ID=1 Partition = 2
hs13fmt---->SCSI ID=1 Partition = 3
hs14fmt---->SCSI ID=1 Partition = 4
hs51fmt---->SCSI ID=5 Partition = 1
hs52fmt---->SCSI ID=5 Partition = 2
hs53fmt---->SCSI ID=5 Partition = 3
hs54fmt---->SCSI ID=5 Partition = 4
```

Step 2. Enter the command `format /hs01fmt -np -nv -r -vOS9000` to format the hard drive. The following table shows the format specified device options.

Table B-2 Format Specified Device Options

`-be`	create big endian fs (ie: PPC)
`-bo=<num>`	use block offset of <num>
`-c`	enable command/interactive mode
`-dd`	double density disk
`-ds`	double sided disk
`-h=<num>`	disk has <num> heads
`-i=<num>`	use interleave of <num>
`-le`	create little endian (ie: x86, ARM)
`-m=<num>`	put bitmap at block <num>
`-np`	no physical format
`-nv`	no physical verify
`-o`	do interleave optimization
`-r`	assume ready (don't ask)
`-s=<num>`	use spiral skew of <num>
`-sd`	single density disk

Table B-2 Format Specified Device Options (continued)

`-ss`	single sided disk
`-to=<num>`	use track offset of <num>
`-t=<num>`	disk has <num> tracks
`-v=<name>`	set volume name to <name>
`-?`	print this help message

Step 3. Your hard drive is now partitioned and formatted, and the OS-9 prompt returns.

OS-9 Partitioning Options

Create OS-9 Partition (1)

Creates OS-9 partitions. When partitions are created, you are prompted for the size of the partition in terms of cylinders.

Set Active Partition (2)

Specifies which partition is bootable. If DOS is set as the active partition and the system is reset, then DOS loads. To allow OS-9 to boot, you must use the DOS version of `fdisk` to set the OS-9 partition to active. If a boot manager is used, then set the Boot Manager as active.

Delete Partition (3)

Deletes partitions. Use the delete option with care. Extended partitions may include any logical drives associated with them.

Display Partition Information (4)

Displays the partition tables. If the -e option is used, additional information about the partition tables displays.

The extended/additional information includes:

Table B-3 Display Partition -e Option

	Explanation
st	Start-flag (if 80 drive is startable)
s_head	Start head (byte)
s_cyl_blk	Start Cylinder block (word)
type	Partition type (word)
e_head	End head (byte)
e_cyl_blk	End cylinder block (word)
s_blk	Start block (LBA) (long-word)
size	Size of block (LBA) (long-word)

Change Extended DOS Partition to OS-9 Partition (5)

Converts an extended partition to an OS-9 partition. Extended partitions may include logical drives.

Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:____

Company:_______________________________________

Phone:____

Fax:_____________________Email:__________________

Product Name:

Description of Problem:

____

____

____

____

____

____

____

____

____

____

Host Platform_____________________________________

Target Platform____________________________________

RadiSys.
MICROWARE SOFTWARE