Home

OS-9® for MBX Board Guide

Version 4.7

RadiSys®
THE POWER OF WE

Copyright and publication information

This manual reflects version 4.7 of Microware OS-9. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

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®

This chapter describes installing and configuring OS-9® on the following Motorola MBX reference boards: MBX821 and MBX860. The following sections are included:

- **Development Environment**
- **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**
- **Writing the OS-9 Image to Flash Memory**
- **Configuring the MBX Environment Parameters**
- **Booting the MBX Reference Board**
- **Creating a Startup File**
- **Optional Procedures**

Note

Please refer to your *MBX Series Embedded Controller Installation and Use* manual for information on hardware preparation and installation, operating instructions, and functional descriptions prior to installing and configuring OS-9 on your MBX reference board. You must also know whether your MBX reference board is an entry level or standard level board. This determines the minimum amount of Flash memory available for loading OS-9 and its utilities on the board.

WARNING

To safely connect an LCD screen to your MBX target system, you need to integrate an additional circuit for applying power to the LCD panel. Failure to do so could result in damage to the LCD screen. If you have not integrated this circuit into your MBX system, *do not* keep the panel connected to the MBX board during the boot sequence. Apply power to the panel only after the internal LCD controller clock starts and turn the power off when the controller clock stops. If you are not sure whether you have the recommended additional circuitry, you can include the MAUI® support modules in the build, using the Configuration Wizard. After you have programmed the Flash memory on your target, type the following initialization command at the OS-9 prompt:

```
iniz /gfx
```

This command initializes the LCD panel.

> **Note**
> You do not need an LCD display terminal unless you plan to use MAUI or Java in your OS-9 system.

Development Environment

Figure 1-1 shows a typical development environment for the MBX reference board. The components shown include the minimum required to enable OS-9 to run on the MBX821 and MBX860 boards.

Figure 1-1 MBX Development Environment

Target System   Host Development System
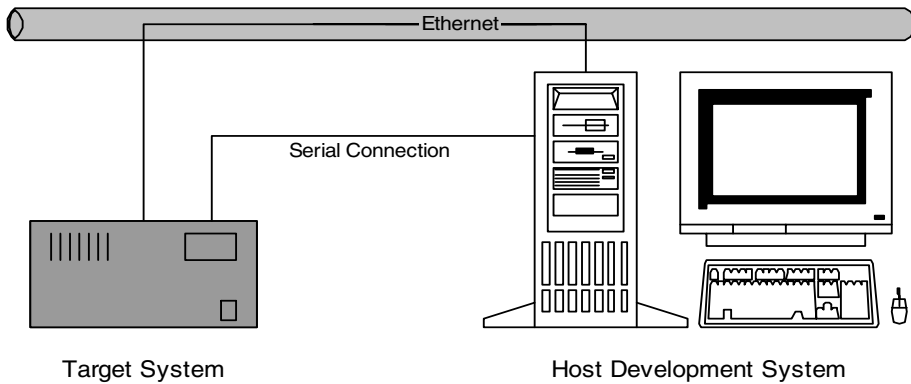
Requirements and Compatibility

Host Hardware Requirements (PC Compatible)

Your host PC must meet the following minimum requirements:

- Windows 95, 98, ME, 2000, or NT
- 300-400 MB of free disk space (an additional 235MB of free disk space is required to run PersonalJava for OS-9)
- an Ethernet network card
- 16MB of RAM (32MB is recommended)
- 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 MBX target

  This application is included with Microware OS-9 for PowerPC and must be loaded onto your host PC during the CD-ROM installation process.

Target Hardware Requirements

Your MBX target system requires the following hardware:

- enclosure or chassis with power supply

- display terminal

- Ethernet 10BaseT and connecting cables

- RS-232 serial connectors

- disk drives and other I/O devices and their appropriate connecting cables

- RAM memory must be single bank EDO DIMMs, 60 nanosecond, 4K refresh, 3.3V, 168pin

- LCD screen, keyboard, and mouse are optional

PersonalJava Hardware Requirements

Your target must have the following to run PersonalJava for OS-9:

- MBX with 16MB of RAM

- 4MB of FLASH (Boot)

- LCD Display (optional)

Target Hardware Setup

The following section details how to set up your target hardware.

Setting the Switches on the Target Board

The board configuration for the J4 jumper on the MBX board enables you to select either the on-board Flash memory (32-bit) or the socketed Flash chip in the 32-pin socket XU2 (8-bit) as the boot ROM for the MBX board.

Table 1-1 provides the attributes for the J4 jumper.

Table 1-1 Jumper J4 Attributes

J4	Boot Device	Bit Width	ROM Address	Flash Address
1-2	ROM	8	FE000000	FC000000
2-3	Flash	32	FC000000	FE000000

Note

If you have an entry level 860 or 821 processor board (the model number of the board ends in -001, or -001A), the minimum size of the 32-bit on-board Flash memory is 2MB. If you have a standard 860 or 821 board, the minimum size of the 32-bit on-board Flash memory is 4MB. If you can not determine the model number of the board, see if the board has hard disk and floppy support. If it does, it is a standard board.

The on-board Motorola monitor/debugger, EPPCBug, resides in the Flash chips. To ensure that you do not erase EPPCBug when reprogramming the Flash memory for the OS-9 boot image, please follow these instructions:

Step 1.   With the power switch OFF, set jumper J3 to pins 1-2. This prevents inadvertently overwriting the Flash memory used in the boot ROM.

Step 2.   Set J4 to pins 1-2 so the socketed Flash chip is used as the boot ROM device.

WARNING

If you programmed the on-board Flash without first setting the J4 jumper to 1-2, you have erased the Motorola EPPCBug utility. The result is the board can not boot because it does not find the EPPCBug boot utility.

To correct this problem, restore the J4 jumper to 2-3 and the J3 jumper to 2-3 and type:

```
EPPCBug> pflash fe000000: <size> fc000000;b
```

This restores EPPCBug back to the on-board Flash. The <size> variable should agree with the amount of Flash previously programmed and should be entered in decimal format (for example, enter &4194296 to program 4 MB of Flash memory on the MBX target system).

Next set the J3 jumper to 1-2 and the J4 jumper to 1-2. Pflash should work fine on the downloaded image at fc000000. Use the following Program Flash Memory command (pflash) to program the OS-9 boot image on the on-board flash:

```
EPPCbug> pflash 4000: <size> fc000000;b
```

More In
fo More
Informatio
n More Inf
ormation M
ore Inform
ation More

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 the documents directly from your browser by opening the following url:

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

Connecting the Target to the Host

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

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. 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 `Hyperterminal` 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 = None
```

Step 10. Click `OK`.

Step 11.   From the Hyperterminal window, select `File/Properties`. Click on the **Settings** tab and select the following options:

`Terminal Keys`

`Emulation = Auto Detect`

`Backscroll Buffer Lines = 500`

Step 12.   Click `OK`.

Step 13.   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 14.   Turn on the target system. A power-on banner and `EPPC-Bug>` prompt should appear on the display terminal connected to the board.

Note

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

Step 15.   Configure `EPPC-Bug` (the MBX debugger).

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

Table 1-2 System Administrator Input

Information Needed	Information Used for this Tutorial
IP Address and Host Name	_____
Broadcast IP Address	_____
Subnet Mask	_____

Table 1-2 System Administrator Input (continued)

Information Needed	Information Used for this Tutorial
Network Domain	_____
DNS IP Addresses	_____
Gateway IP Addresses	_____

This information is also applied in the **Network Configuration** section.

Step 16. At the `EPPC-Bug>` prompt, enter `vpd` and then `<return>`. This is the Display Vital Product Data command.

Step 17. Write down the Ethernet address that is displayed. This is later used as the MAC address in the Configuration Wizard.

Step 18. At the `EPPC-Bug>` prompt, enter `niot` and then `<return>`. This is the Network I/O Teach command for configuring network parameters.

Step 19. Configure the network parameters as follows:

Note

The `Node Memory Control Address` is different depending upon the MBX board you are using.

```
Controller LUN =20? 20 <return>
Device LUN =00? <return>
Node Memory Control Address =003c8000? <return>
Client IP Address =0.0.0.0? <MBX board IP address> <return>
Server IP Address =0.0.0.0? <development system IP address> <return>
Subnet IP Address Mask =0.0.0.0? <MBX board subnet mask> <return>
Broadcast IP Address =0.0.0.0? <subnet broadcast IP> <return>
Gateway IP Address =0.0.0.0? <subnet gateway IP, if any> <return>
Boot File Name ("NULL" for none) =?   <return>
Argument File Name ("NULL" for None) =?   <return>
Boot File Length =00000000? <return>
Boot File Byte Offset =00000000? <return>
```

```
BOOTP/RARP Request Retry =00? <return>
TFTP/ARP Request Retry =00? <return>
Trace Character Buffer Address =00000000? <return>
BOOTP/RARP Request Control: Always/When-Needed (A/W)=W? <return>
BOOTP/RARP Reply Update Control: Yes/No (Y/N) =Y? <return>
Update Non-Volatile Memory (Y/N)? y <return>
```

For More Information

Refer to the ***Motorola EPPCBug Firmware Package User's Manual*** for detailed information on the EPPCBug commands.

Step 20. Check that your Ethernet network connection for your host PC 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.

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 OS-9 installation process.

Starting the Configuration Wizard

The Configuration Wizard is the application used to build the coreboot, bootfile, or ROM image. To start the Configuration Wizard, perform the following steps:

Step 1.   From the Windows desktop, select Start -> RadiSys -> Microware OS-9 for <product> -> Configuration Wizard. You should see the following opening screen:

Figure 1-2 Configuration Wizard Opening Screen
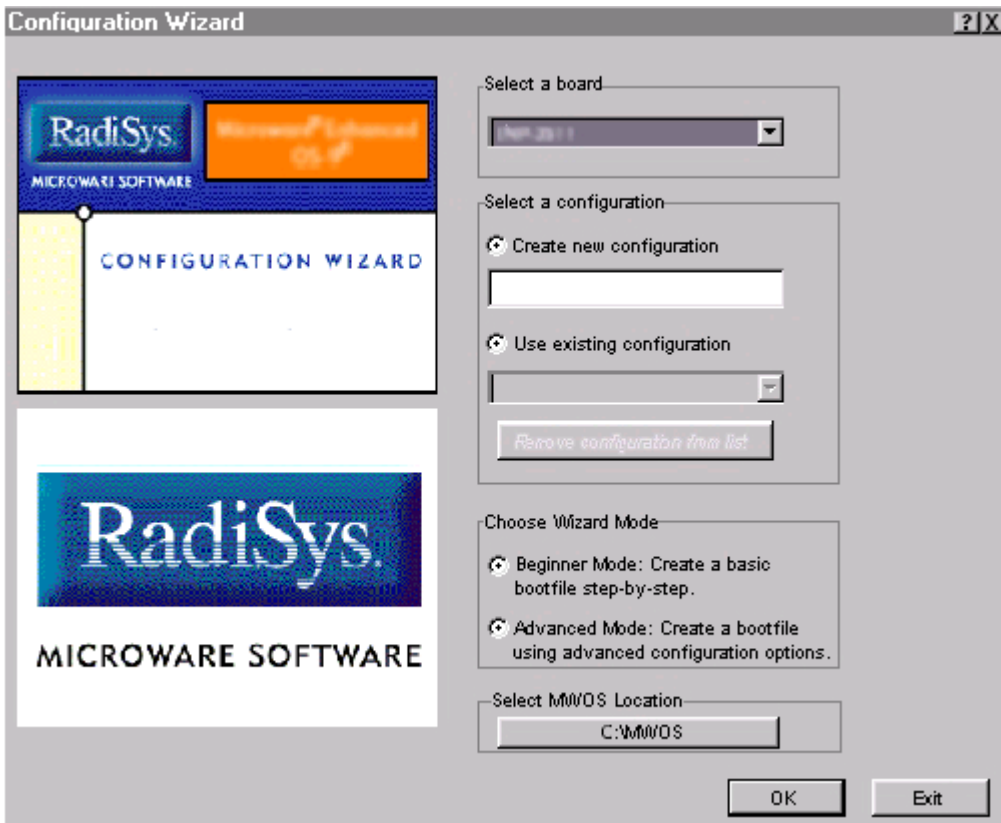
Step 2.   Select your target board from the **Select a board** pull-down menu.

Step 3.   Select the `Create new configuration` radio button from the
          Select a configuration menu and type in the name you want to give
          your ROM image in the supplied text box. This names your new
          configuration, which can later be accessed by selecting the **Use
          existing configuration** pull down menu.

Step 4.   Select the `Advanced Mode` radio button from the **Choose Wizard
          Mode** field and click `OK`. The Wizard's main window is displayed. This is
          the dialog from which you will proceed to build your image. An example
          is shown in **Figure 1-3**.

Figure 1-3 Configuration Wizard Main Window

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

This section provides an example of an OS-9 ROM image successfully built on a Host PC and transferred to an MBX860 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 MBX target board, you must know the size of available Flash memory on the target board. This setting is configured by selecting the **ROM Memory List** tab and activating the **Setting Based On** pull-down menu. This is shown in **Figure 1-4**.

Figure 1-4 Configuring Flash Memory
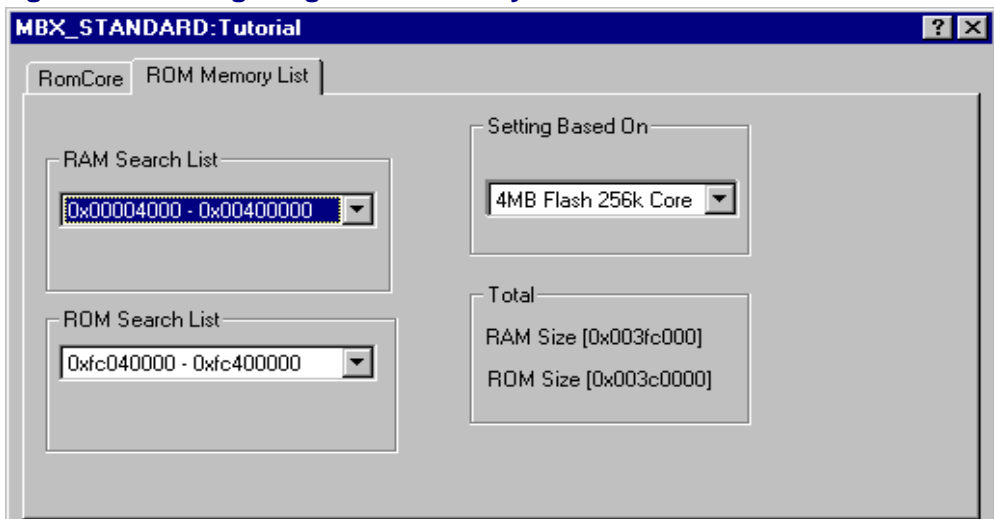
Configure Coreboot Options

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

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

Figure 1-5 Coreboot Configuration—Debugger Tab
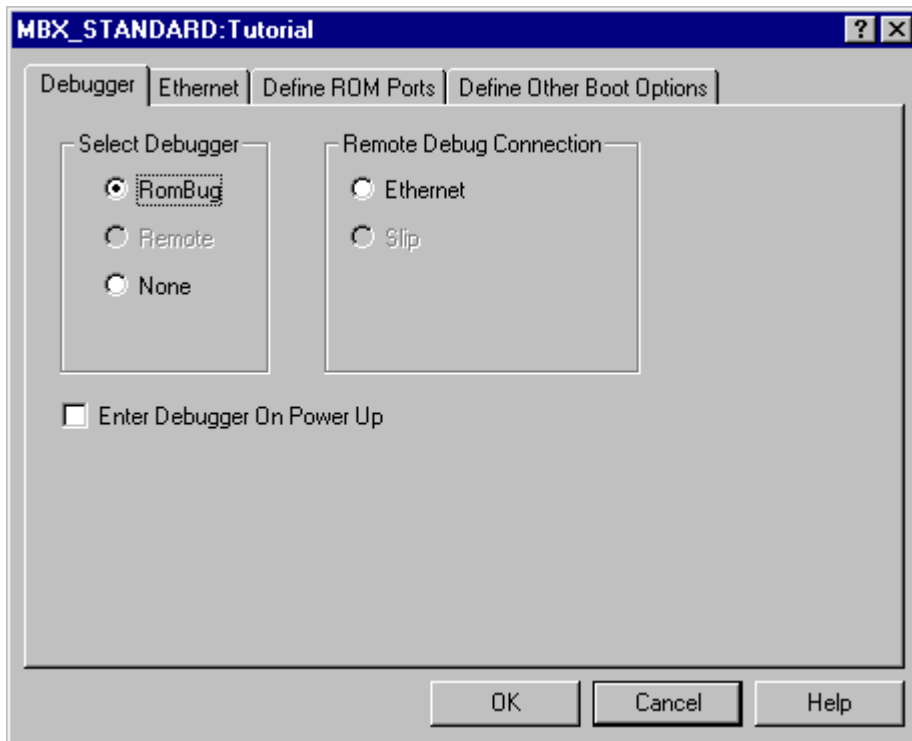
Step 3. Under Select Debugger, select `RomBug`. This sets Ethernet as the method for user state debugging. Select `None` if you do not want to debug your program.

> **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.

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

Configure System Options

When you select `Configure -> Bootfile -> Configure System Options` the **System Options** window appears. This window contains the **Define /term Port** tab, the **Bootfile Options** tab and the **MAUI Options** tab. You can use the default settings for your selections.

Network Configuration

To configure your network, complete the following steps:

Step 1.   If you want to use the target board across a network, you will need to configure the Ethernet settings within the Configuration Wizard. To do this, select `Configure -> Bootfile -> Network Configuration` from the Wizard's main menu.

Step 2.   From the **Network Configuration** dialog, select the `Interface Configuration` tab. From here you can select and enable the interface. For example, you can select the appropriate Ethernet card from the list of options on the left and specify whether you would like to enable IPv4 or IPv6 addressing. **Figure 1-6** shows an example of the **Interface Configuration** tab.

Figure 1-6 Bootfile -> Network Configuration -> Interface Configuration
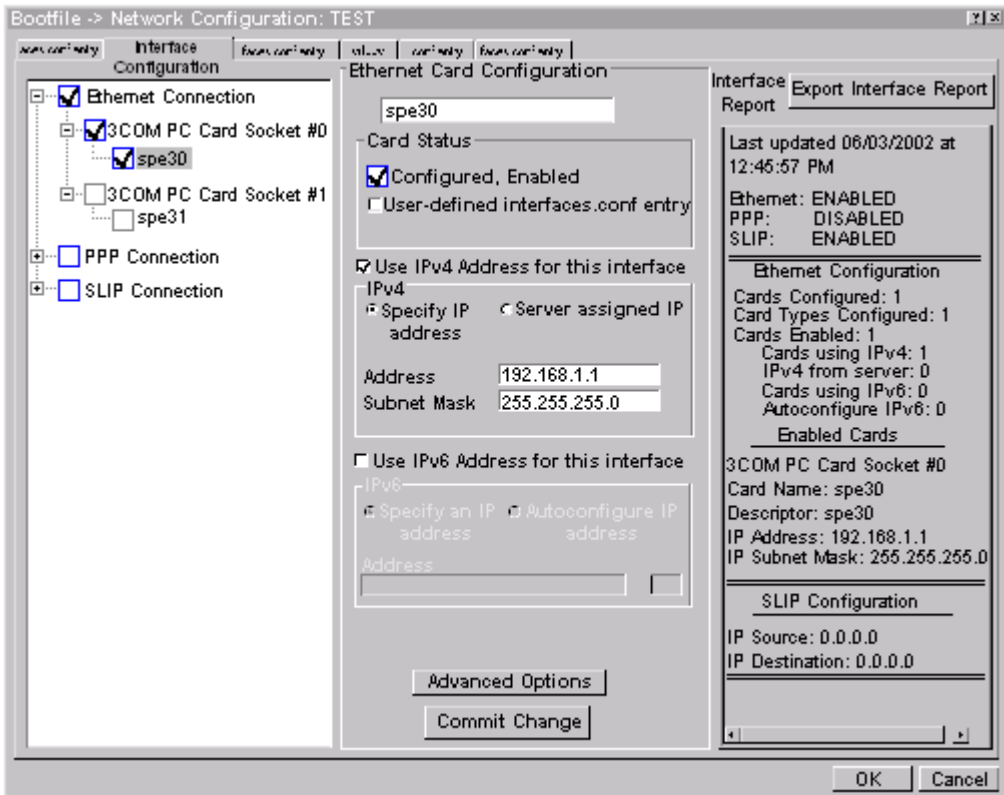
For More Information

To learn more about IPv4 and IPv6 functionalities, refer to the ***Using LAN Communications*** manual, included with this product CD.

For More Information

Contact your system administrator if you do not know the network values for your board.

Step 3.   Once you have made your settings in the **Network Configuration** dialog, click `OK`.

Step 4.   Set configuration for the Domain Name Server (DNS).

- 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.

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

Note

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 window in **Figure 1-7** 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 1-7 Bootfile Configuration—SoftStax Setup Tab
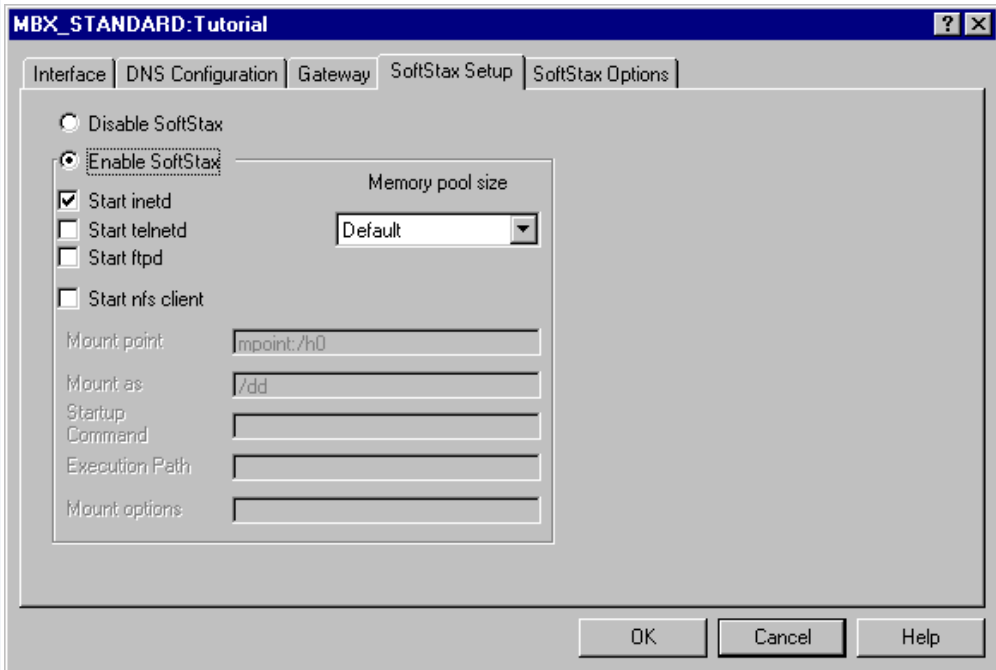
Step 7.   Click Enable SoftStax.

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 following window is displayed.

Figure 1-8 Bootfile Configuration—Disk Configuration Interface
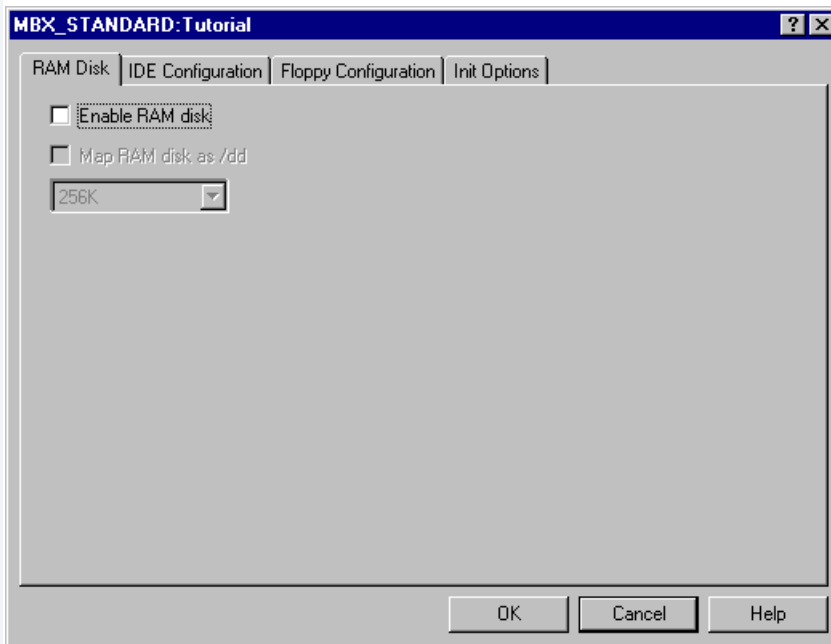
The **Disk Configuration** window contains the following tabs:

- The **RAM Disk** tab enables you to create a RAM disk of any size for loading modules onto the target.

- The **IDE Configuration** tab enables you to configure various 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` tab. The following window is displayed.

Figure 1-9 Bootfile Configuration—Init Options Tab
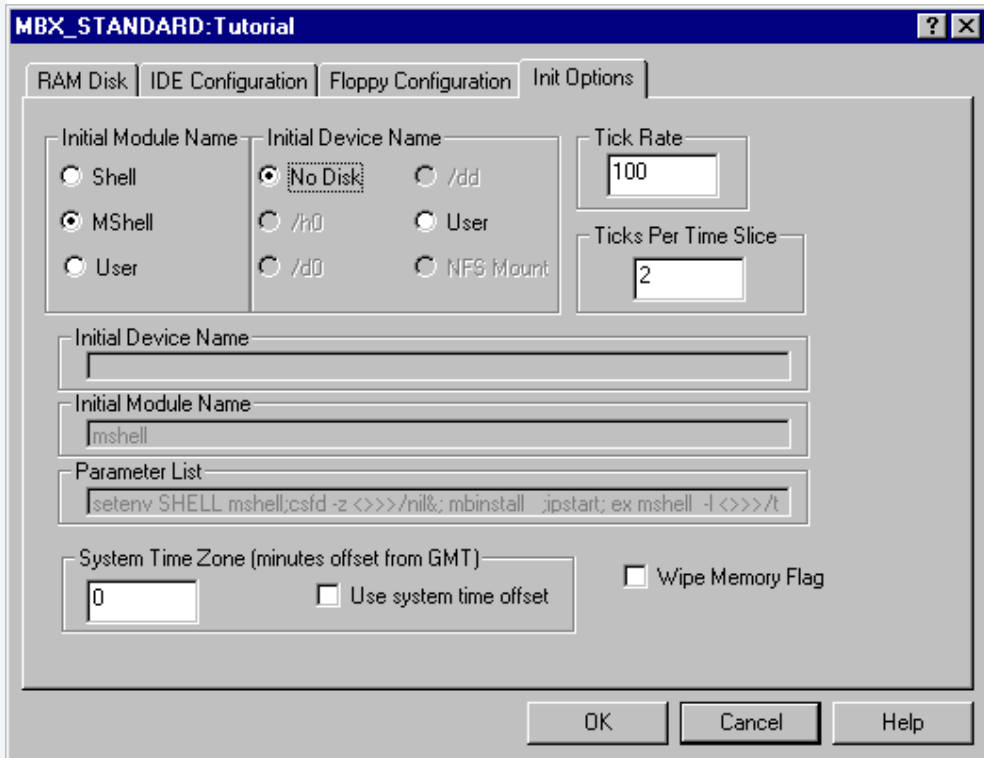
- Select the `Mshell` option for the initial module name. This causes OS-9 to start a console shell usable from your terminal window. Select No Disk in the **Initial Device Name** section.

- 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 3.   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 and rom.s files are created in the following directory:

`MWOS/OS9000/821/PORTS/MBX8XX/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 later use. If you choose to close, you can save your configuration settings.

Transferring the ROM Image to the Target

`TFTPServer32` is the Trivial File Transfer Protocol (TFTP) server utility that must be installed on your PC host from the Microware 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 1-10 TFTP Server Options Window
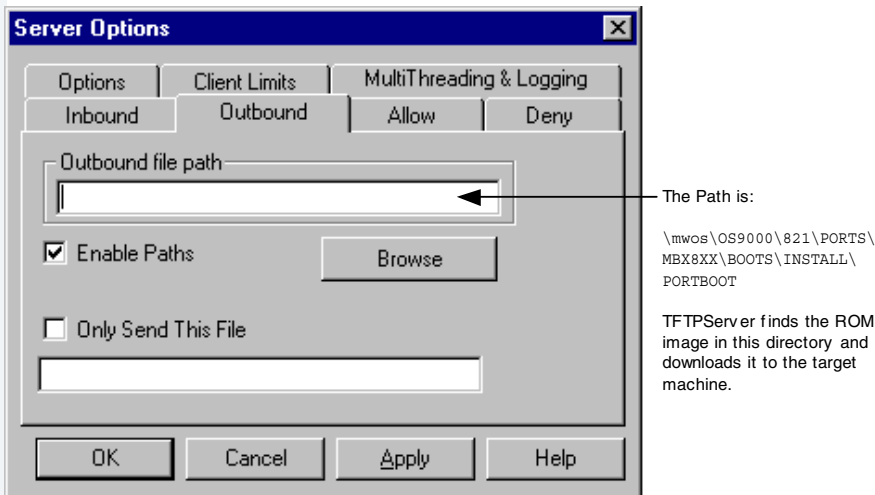
Step 3.   Use default settings for all other settings.

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

Step 5.   At the `EPPC-Bug>` prompt on the target, enter the Network I/O Physical command:

```
niop
<return>
```

> **Note**
>
> The `NIOP` command enables you to get files from the supported Ethernet network interfaces and put files to the supported Ethernet network interfaces. When invoked, this command goes into an interactive mode, prompting you for all parameters necessary to carry out the command. This command uses the TFTP protocol to perform the file transfer.

Step 6.   Configure the parameters as follows:

```
Controller LUN =00? 20 <return>
Device LUN =00? <return>
Get/Put =G? <return>
File Name =? rom <return>
Memory Address =00004000? <return>
Length =00000000? <return>
Byte Offset =00000000? <return>
```

> **Note**
>
> The transfer can take a minute or more depending on your network conditions. If you are using `TFTPServer32`, you will see a log entry reporting a successful transfer. If the utility appears to be hung or showing no progress, verify that your server IP address is correct.

Writing the OS-9 Image to Flash Memory

After completing the previous stage, you should see the following displayed in the Hyperterminal window:

```
Bytes received = &4194296 Bytes Loaded = &4194296
Bytes/Second &?????, Elapsed Time = ? second(s)
```

- The `bytes received = &4194296` is the size of Flash you selected in the `ROM Memory List` dialog in the Wizard. This number varies, depending upon the devices and options you selected in the wizard. For example, the `bytes received = &2097144` could also be possible.

- The `bytes/second (&?????)` varies, depending on the speed of your TFTPServer.

To write the OS-9 image to flash memory, complete the following steps:

Step 1.   At the `EPPC-Bug>` prompt on the target, enter the Program Flash Memory command. Remember to use the `bytes received = <&number>` as seen in the Hyperterminal window after the ROM file has been downloaded to your MBX target system:

```
pflash 4000:&4194296 fc000000;b
<return>
Program Flash Memory <y/n>
y
<return>
```

⚠️ **WARNING**

If you are writing a smaller FLASH than the previous FLASH, modules may remain in FLASH from the previous write. The process for eliminating stale modules from FLASH is similar to the process for programming a real image into FLASH, except that the image programmed is all `F`s (or `1`s) which is identical to an erased FLASH.

To write `F`s into FLASH, pick an area of RAM to be used as a buffer, fill that buffer with `F`s, then program the FLASH with the buffer image.

Type the following for a 2MB FLASH at the EPPC-Bug prompt:

```
EPPC-Bug> bf 4000:80000 ffffffff
```

where 80000 is the size in hex longwords

You see the following displayed on your screen:

```
Effective address: 00004000
Effective count  : &2097152    (for a 2MB FLASH)
```

Type the following at the EPPC-Bug prompt:

```
EPPC-Bug> pflash 4000:200000 fc000000;b
```

where `200000` is the size and `fc000000` is the FLASH destination address

This is best done before writing a desired image to FLASH. By carefully selecting the size and FLASH destination address, a portion of the FLASH can be erased, retaining a desired portion of the FLASH image.

Step 2.   Wait for the Flash programming to complete.

Step 3.   At the `EPPC-Bug>` prompt on the target, enter the **Reset** command:

```
Reset
<return>
Cold/Warm Reset [C,W] = C? <return>
Execute Local SCSI Bus Reset [Y,N] = N? Y <return>
Execute MPC8xx Reset = [Y,N] = N? Y <return>
```

Selecting `Y` to `Execute MPC8xx Reset` resets the system and returns you to the `EPPC-Bug>` prompt.

Configuring the MBX Environment Parameters

EPPC Debugger is an on-board monitor/debugger residing in the Flash chips. Its functions include booting and resetting the system, initializing a request, displaying and modifying configuration variables, running self-tests and diagnostics, and updating firmware ROM. Access the EPPC Debugger using a terminal program (such as Hyperterminal).

Step 1. At the `EPPC-Bug>` prompt, enter the **Set Environment** command:

`env`, then `<return>`.

Step 2. Configure the parameters as follows:

```
Probe System for Supported I/O Controllers [Y/N] = Y? <return>
Local SCSI Bus Reset on Debugger Startup [Y/N] = N? <return>
PCI Interrupts Route Control Registers(PIRQ0/1/2/3)=0A0B0E0F? <return>
Firmware Command Buffer Offset = 000002C8? <return>
Firmware Command Buffer Size = 00000200? <return>
Firmware Command Buffer Delay = ? 5000 <return>
Program Intermediate Load Address = 00200000? <return>
Binary Program Load Address = 00080000? <return>
Binary Program Execution Offset = 00000100? <return>
Primary Network Controller LUN = 20? <return>
Primary Network Device LUN = 00? <return>
Firmware Command Buffer:
['NULL' terminates entry]? go fc000100 <return>
 NULL <return> (Note: the NULL entry is case sensitive)
```

Note

Typing `go fc000100` `<enter>` `NULL` `<enter>` at the `Firmware Command Buffer` prompt enables the board to automatically boot to the OS-9 boot menu. Otherwise, you need to type `go fc000100` each time the system boots.

```
Update Non-Volatile Memory (Y/N) Y <return>
Reset Local System (CPU)[Y/N}? Y <return>
```

For More Information

Refer to the ***Motorola EPPCBug Firmware Package User's Manual*** for more information on how to configure the MBX system using EPPCBug.

Booting the MBX Reference Board

Note

If you typed `go fc000100` in the **Firmware Command Buffer** prompt in the **Configuring the MBX Environment Parameters** section, then selecting `Y` to `Execute MPC8xx Reset` resets the system and boots to the OS-9 menu as shown in step 2 below.

To boot the MBX Reference Board, complete the following steps:

Step 1.   At the `EPPC-Bug>` prompt, type the following command:

`EPPC-Bug> go fc000100`

Note

The `GO` command initiates target code execution. All previously set breakpoints are enabled. If an address is specified, it is placed in the target IP. Execution starts at the target IP address.

Step 2.   A similar boot menu appears:

```
OS-9000 Bootstrap for the PowerPC(tm)
BOOTING PROCEDURES AVAILABLE ---------- <INPUT>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot - <lr>
Boot from PC Card -------------------- <pc>
Boot from (RBF) Floppy disk ---------- <fd>
Boot from PC Floppy disk ------------- <pf>
Boot from (RBF) Hard Disk ------------ <hd>
PCI View Utility --------------------- <pciv>
```

```
Enter ROM Debugger -------------------- <break>
Restart the System ------------------- <q>
Select a boot method from the above menu:
```

Note

The boot menu can have different selections, depending upon your selections using the Wizard.

Step 3.   Type the BO command to select booting OS-9 in-place. The OS-9 prompt appears.

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 Microware 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

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 example 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 example 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.

Chapter 2: Board Specific Reference

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

- **Boot Menu Options**
- **Vector Descriptions for PowerPC MPC821**
- **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*.

RadiSys.

MICROWARE SOFTWARE

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 MPC821

Table 2-2 Vector Descriptions for PowerPC 821

Vector Number	Related OS-9 Call	Assignment
00	None	Reserved
01	F_IRQ	System reset
02	F_STRAP, F_IRQ	Machine check
03	F_STRAP, F_IRQ	Data access
04	F_STRAP, F_IRQ	Instruction access
05	F_IRQ (in siuirq)	External interrupt
06	F_STRAP, F_IRQ (in ssm)	Alignment
07	F_STRAP, F_TLINK, F_IRQ	Program
08	None	Floating-point unavailable
09	F_IRQ (in tkdec)	Decrementer
0A	None	Reserved
0B	None	Reserved
0C	F_SSVC	System call
0D	None	Trace

Table 2-2 Vector Descriptions for PowerPC 821 (continued)

Vector Number	Related OS-9 Call	Assignment
0E	None	Floating point assist
0F	None	Reserved
10	fpu	Implementation dependent software emulation
11	ssm	Implementation dependent instruction TLB miss
12	ssm	Implementation dependent data TLB miss
13	ssm	Implementation dependent instruction TLB error
14	ssm	Implementation dependent data TLB error
21-27	None	Reserved
28	None	Implementation dependent data breakpoint
29	None	Implementation dependent instruction
30	None	Implementation dependent peripheral breakpoint
31	None	Implementation dependent non-maskable development port

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 created the process or the process had debug attached with F_DATTACH, then the resources of the erroneous process remain intact and control returns to the parent debugger to allow a post-mortem 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-9 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. In addition, 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.

OS-9 Vector Mapping

This section contains the vector mappings and dual-port RAM mappings for the 821/860 processors.

The system modules `siuirq` and `cpicirq` map interrupts coming from the SIU and CPM into the OS-9 vector table according to the following mappings.

SIU (System Interface Unit) vectors are mapped starting at vector 0x40 in the order shown in Table 12-1 of the **MPC821 User's Manual,** and as shown in the following table.

Table 2-3 System Interface Unit Vectors

Vector	Source
0x40	IRQ0
0x41	Level 0
0x42	IRQ1
0x43	Level 1
0x44	IRQ2
0x45	Level 2
0x46	IRQ3
0x47	Level 3
0x48	IRQ4
0x49	Level 4 (CPIC)
0x4a	IRQ5

Table 2-3 System Interface Unit Vectors (continued)

Vector	Source
0x4b	Level 5
0x4c	IRQ6
0x4d	Level 6
0x4e	IRQ7
0x4f	Level 7

CPM (Communications Processor Module) vectors are mapped starting at vector 0x50 in the order shown in Table 16-43 of the *MPC821 User's Manual*, and as shown in the following table.

Table 2-4 Communications Processor Module Vectors

Vector	Source
0x50	Error
0x51	Parallel I/O--PC4
0x52	Parallel I/O--PC5
0x53	SMC2/PIP
0x54	SMC1
0x55	SPI
0x56	Parallel I/O--PC6
0x57	Timer 4

Table 2-4 Communications Processor Module Vectors (continued)

Vector	Source
0x58	Reserved
0x59	Parallel I/O--PC7
0x5a	Parallel I/O--PC8
0x5b	Parallel I/O--PC9
0x5c	Timer 3
0x5d	Reserved
0x5e	Parallel I/O--PC10
0x5f	Parallel I/O--PC11
0x60	I2C
0x61	RISC Timer Table
0x62	Timer 2
0x63	Reserved
0x64	IDMA2
0x65	IDMA1
0x66	SDMA Channel Bus Error
0x67	Parallel I/O--PC12
0x68	Parallel I/O--PC13

Table 2-4 Communications Processor Module Vectors (continued)

Vector	Source
0x69	Timer 1
0x6a	Parallel I/O--PC14
0x6b	SCC4
0x6c	SCC3
0x6d	SCC2
0x6e	SCC1
0x6f	Parallel I/O--PC15

Dual-port RAM Mapping

The 821 and 860 processors include 5120 bytes of dual-port RAM for buffer descriptor and microcode use. Since the high- and low-level drivers both use this area and must agree on their usage of it, the following locations have been reserved for the following uses:

Table 2-5 Dual Port RAM Use Map

Offset into DPRAM	Use
0x0 - 0x0f	SCC1
0x10 - 0x1f	SCC2
0x20 - 0x2f	SCC3
0x30 - 0x3f	SCC4

Table 2-5 Dual Port RAM Use Map (continued)

Offset into DPRAM	Use
0x40 - 0x4f	SMC1
0x50 - 0x5f	SMC2
0x60 - 0xff	reserved
0x100 - 0x17f	Ethernet
0x180 - 0x200	reserved

Configuring Booters

The following booters are available for the MPC8xx target platforms. The abbreviated name and configuration parameters for the booters are listed with recommended values (if any).

Note

The MPC8xx booters are located in `coreboot.ml`

Table 2-6 MPC8xx Booters

Booter	Description	Recommended Values
fdc765	Standard floppy disk booter	
	Abbreviated name:	`"fd"`
	Configuration parameters:	`"port=0x800003f0"`
		`"lun=0"`
		`"si=0"`
		`"ei=3"`

Table 2-6 MPC8xx Booters (continued)

Booter	Description	Recommended Values
`fsboot`	TEAC SCSI floppy disk booter	
	Abbreviated name:	`"fs"`
	Configuration parameters:	`"port=0xff000000"`
		`"device=ncr8xx"`
		`"id=6"`
		`"si=0"`
		`"ei=3"`
`hsboot`	SCSI hard disk booter	
	Abbreviated name:	`"hs"`
	Configuration parameters:	`"port=0xff000000"`
		`"device=ncr8xx"`
		`"id=<default scsi ID>"`
		`"si=0"`
		`"ei=3"`
		`"lsnoffs=2052"`

Table 2-6 MPC8xx Booters (continued)

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

Table 2-6 MPC8xx Booters (continued)

Booter	Description	Recommended Values
vsboot	SCSI tape booter	
	Abbreviated name:	"vs"
	Configuration parameters:	"port=0xff000000"
		"device=ncr8xx"
		"id=4"

Port Specific Utilities

The following port specific utilities are included:

- `dmppci`

- `mouse`

- `pciv`

- `setpci`

- `testpci`

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 MBX821/860 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
```

OS-9 for the MBX Board Guide

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
```

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-9000 linker automatically adjusts for it.

Appendix A: Board Specific Modules

This chapter describes the modules specifically written for the target board. It includes the following sections:

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

Low-Level System Modules

The following low-level system modules are tailored specifically for the MBX reference platform. The functionality of many of these modules can be altered through changes to the configuration data modules (`cnfgdata`). These modules are located in the following directory:

`MWOS/OS9000/821/PORTS/MBX8XX/CMDS/BOOTOBJS/ROM`

cnfgdata	is a data module containing configuration parameters.
cnfgfunc	retrieves configuration parameters from the `cnfgdata` module.
conscnfg	retrieves the name of the low-level auxiliary communication port driver from the cnfgdata module.
harddisk	allows booting from hard disk.
	This module uses `ide` and `fdman`.
iocpm	is a driver for SCC/SMC serial port.
llquicc	provides network driver services for Ethernet port.
pccard	allows booting from a PC Card ATA device.
	This module uses `ide` and `pcman`.
portmenu	retrieves a list of configured booter names from the ROM cnfgdata module.
qspan	provides PCI bus configuration.
rpciv	is a `pciv` psuedo-booter (PCI bus scanner) module.
tbtimer	provides polling timer services using `tblo` and `tbhi` registers.
usedebug	is a debugger configuration module.
w83c553f	is a PCI bus configuration module.

High-Level System Modules

The following OS-9 system modules are tailored specifically for the MBX reference board. Unless otherwise specified, each module is located in a file of the same name in the following directory:

`MWOS/OS9000/821/PORTS/MBX8XX/CMDS/BOOTOBJS`

`cpicirq`	is an interrupt dispatch module.
`llqd`	disables llquicc.
`pcisub`	is a PCI handling module.
`picirq`	provides interrupt controller support.
`picsub`	is a PIC handling module.
`rb1003`	provides support for IDE and EIDE drives up to 4GB.
	Many descriptors are provided for use with this driver. Among these provided descriptors are several modules named h0 and dd.
`rb765`	is a real-time clock module.
`rtc821`	is a real-time clock module.
`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.
`sc8042k`	is a keyboard/mouse driver.
`sccpm`	is a high-level driver for 8xx SCCI and SMCI.
`scp87303`	is a 1284 parallel port driver (use with 37C672).

`siuirq`	is an inerrupt dispatch module.
`tk821pit`	provides the system ticker based on the SIU periodic interrupt timer.
`tkcpm`	provides the system ticker based on the CPM general purpose timer.
`tkdec`	provides the system ticker based on the PowerPC decrementer.

Common System Modules List

The low-level system modules shown in **Table 2-7** provide generic services for OS9000 Modular ROM. They are located in the following directory:

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

Table 2-7 Typical Coreboot Image Contents

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	provides general booting services for RBF file systems.
flboot	is a SCSI floptical 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 2-7 Typical Coreboot Image Contents (continued)

Module	Description
ide	provides target-specific standard IDE support, including PCMCIA ATA PC cards.
iovcons	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 module.
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 2-7 Typical Coreboot Image Contents (continued)

Module	Description
parser	parses key fields from the cnfgdata module and the user parameter fields.
pcman	provides 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	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 2-7 Typical Coreboot Image Contents (continued)

Module	Description
tsboot	is a SCSI TEAC tape drive 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.

RadiSys.

MICROWARE SOFTWARE

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 *Microware 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.

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

Step 2. 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 the 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 3. 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 item three.

For More Information

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

Step 4.   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 5.   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 6.   Hit `<return>`

Step 7.   The following is displayed:

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

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

Step 8.   Type `2` for Extended type 41 partition

Step 9.   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 10.   Hit `<esc>`

Step 11.   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 12.   Type `Y` to save the partition information to disk. You return to the OS-9 prompt.

Step 13.   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 ***Microware OS-9*** CD.

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

```
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 option.

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
`-ss`	single sided disk

Table B-2 Format Specified Device Options (continued)

`-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.