Home

OS-9® for x86 Board Guide

Version 4.7

RadiSys
THE POWER OF WE

www.radisys.com

Revision A • July 2006

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.

Contents

1 Installing and Configuring OS-9®

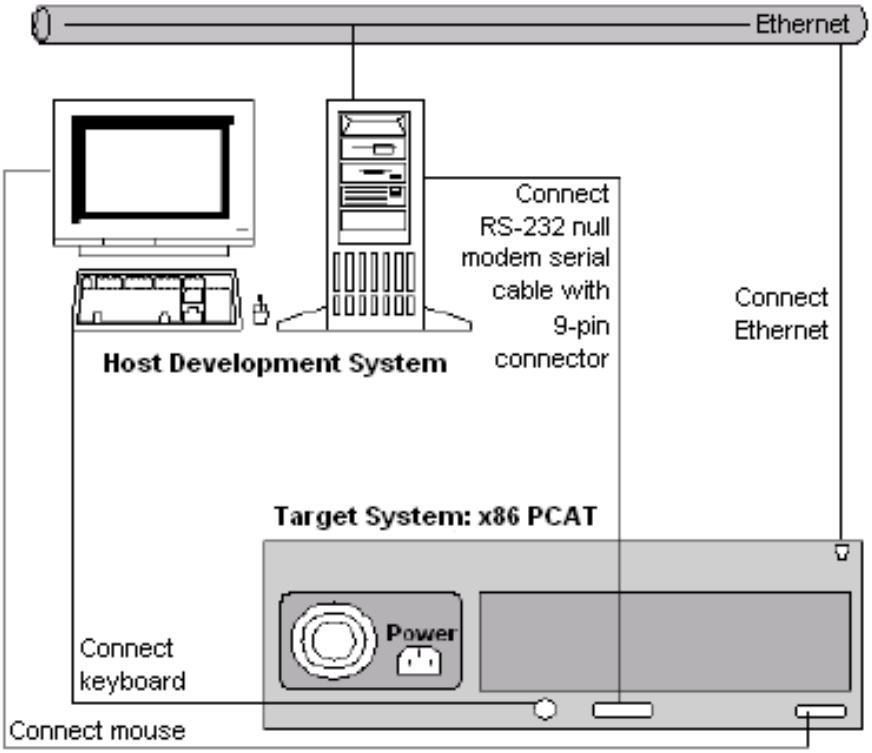
This chapter describes installing and configuring Microware OS-9® for x86/Pentium on the PCAT, MediaGX, and STPC targets.

Even though much of the information in this manual is cast in context of the PCAT port, this information applies to the MediaGX and STPC ports as well. If a piece of information is relevant only to the PCAT, MediaGX, or STPC port, that piece of information is labeled as such.

Development Environment Overview

Figure 1-1 shows a typical development environment for the x86 board.

Figure 1-1. x86 Development Environment



Requirements and Compatibility

Before you begin, install Microware OS-9 for x86/Pentium onto your host PC.

Host Hardware Requirements (PC Compatible)

The host PC must have the following minimum hardware characteristics:

- the recommended amount of RAM for the host operating system
- 250 – 350 MB free disk space
- CD-ROM drive
- network card (required when using Microware's Hawk™ to debug applications on the target computer)

Host Software Requirements (PC Compatible)

The host PC must have the following software installed:

- Microware OS-9
- Microsoft Windows 2000 or XP

Target Hardware Requirements

The following is required of your target hardware:

- compatible target hardware, such as PCAT, MediaGX, or STPC board
- monitor (may be removed once OS-9 is installed)
- IDE, SCSI, or other storage device
- floppy drive (may be removed once OS-9 is installed)
- network connection to the host computer (required for initial configuration and when using Microware's Hawk to debug applications)
- keyboard or serial connection to the Windows host computer (may be removed once OS-9 is installed)

Target Hardware Setup

Supported Devices

The following sections list the supported devices for Microware OS-9 for x86/Pentium.

Refer to Appendix B, Configuring Hardware Devices of this document for detailed information on configuring and troubleshooting specific devices with OS-9.

Ethernet Controllers

To complete development work, you will need an ISA, PCI or PCMCIA network card supported by OS-9. Driver support is also included for network cards from the following manufacturers:

- 3Com PCI series

3C900-TPO	10Base-T TPO NIC
3C905-TX	10/100 Base-TX (RJ-45)
3C905-T4	10/100 Base-T4 (RJ-45)
3C900B-CMB	10Base-T/10Base-2/AUI Combo
3C900B-TPO	10Base-T TPO NIC
3C905B-TX	10/100 Base-TX NIC
3CSOHO100-TX	10/100 Base-TX NIC

9006	10Base-T/10Base-2/TPC
9058	10/100 COMBO Deluxe board
9200	Tornado NIC
9800	10/100 Base-TX NIC(Python-H)
9805	10/100 Base-TX NIC(Python-T)

- 3Com ISA Etherlink III
  (includes the 509B part)
- 3Com Etherlink III PC Card
  Etherlink III 3C589D
- DEC 21140
  Asante' Fast 10/100
  NETGEAR FA310
  SMC EtherPower 10/100 - SMC9332DST
  SMC EtherPower 10/100 - SMC9334BDT/SC (Dual)
- Intel® Pro/100 Series
  82558
  82559
  82559ER
- Intel® Pro/1000 Series
  82540
  82541
- Realtek RTL8139A
  8139 on board
- SMC 91C94/96
  Versalogic board
- LAN79C961/AM79C973
  WinSystems PC104+ card (driver is used with vmware)

- NE2000 (PCI)

  Compex RL2000

  Holtek HT80232

  Holtek HT80229

  KTI ET32P2

  NetVin NV5000SC

  RealTek RTL-8029 and RL8139

  SureCom NE34

  Via 82C926 and 92C926

  Winbond 89C940

  Winbond w89C940

- NE2000 (ISA)

  PCI drivers can be used with ISA cards. The interrupt vector and port address will have to be changed.

  ACCTON - EN166X MPX 2 Ethernet

  D-LINK DE-220PCT - 10Mbps Combo 16-Bit Ethernet ISA Adapter

  ZF netDisplay

- NE2000 PCMCIA (PC Card)

  Socket LP-E, NE2000 clone (83902 core)

- NETGEAR FA311/FA312
- Cirrus Logic CS8900 (ISA)

  No system-state debugging available for this card.

For some Ethernet cards, the I/O base address and interrupt settings must be configured on the card to match the settings used by OS-9. A setup disk, provided with the network card, may be needed to configure the card to the correct settings. The default settings for an NE2000 card are I/O Base 0x340 and IRQ 9.

MAUI® VGA Support

- MediaGX onchip (`gx_mediagx`) * **MediaGX only**
- MediaGX onchip (high color `gx_mediagxh`) * **MediaGX only**
- Generic VGA mode 13 (320x200x8bpp)
- Generic VGA mode 12 & "X" (640x480x4bpp & 360x480x8bpp)
- Cirrus Alpine Series - CL-GD5434, CL-GD5480 etc. (up to 1024x768x24bpp) * **PCAT only**
- Standard VESA (INT 10h) driver (`gx_vesa`)
- Linear mode VESA (INT 10h) driver (`gx_vesal`)

- High color VESA (INT 10h) driver (`gx_vesah`)
- Liner mode, high color VESA (INT 10h) driver (`gx_vesalh`)
- ISA banked

Sequential Device Support

- VGA Graphics / Keyboard
- Serial Mouse
- PS2 Mouse
- 16550 Serial
- Digiboard
- Hostess i
- LavaPort-Quad (Lava PCI QUAD Card)

  One card support enabled, remove # in front of the lines for the other three cards in the `bootfile.ml` file to enable them .
- Risicom (STPC only)
- Parallel Printer

Physical Disk Media

- IDE Standard
- PCMCIA IDE
- PCAT-style Floppy
- Symbios 810,810A,825,825A and 875 PCI SCSI controllers—Wide, Ultra and Ultra Wide
- DiskOnChip
- Diamond FirePort20 and FirePort40—Wide, Ultra and Ultra Wide
- Adaptec 1540/1542 ISA
- Adaptec 2940, 2940U and 2940UW

System Devices

- Real Time Clock

Additional Devices

- PPP and SLIP

CMOS Settings

It may be necessary to modify the BIOS settings in CMOS to boot from a hard disk. Configure the board with the correct settings for the attached peripherals. The boot sequence should try floppy first, and then the IDE hard drive.

SIMM Modules

OS-9 will run with as little as 2 MB of RAM; however, it may be more convenient to install additional memory when developing and testing graphic intensive applications.

Refer to the **Hardware Installation** section of your hardware manual when installing the SIMM modules.

Connecting the Target to the Host

Connecting the PCAT Target

To connect an PCAT board to the host, complete the following steps:

Step 1. Connect the target system to a power supply. Make sure the power switch is in the OFF position.

Step 2. Connect the target system to an Ethernet network (required for initial configuration and when using Microware's Hawk to debug applications).

Step 3. Connect the target system to the host system using an RS-232 null modem serial cable with 9-pin connectors.

Step 4. Connect the target system to a monitor and keyboard (may be removed once OS-9 is installed).

Connecting the STPC Target

To connect an STPC board to the host, complete the following steps:

Step 1. Connect the target system to a power supply. Make sure the power switch is in the OFF position.

Step 2. Connect the floppy drive to the mother board, using an FDC1 board connector.

Step 3. Connect the IDE drive to the mother board using an IDE1 board connector.

Step 4. Connect a VGA monitor to the VA1 connector on the board. (You can connect an NTSC monitor to the P2 connector on the board instead.)

Step 5. Connect a PS/2-style keyboard and mouse to the KB1 connectors on the board.

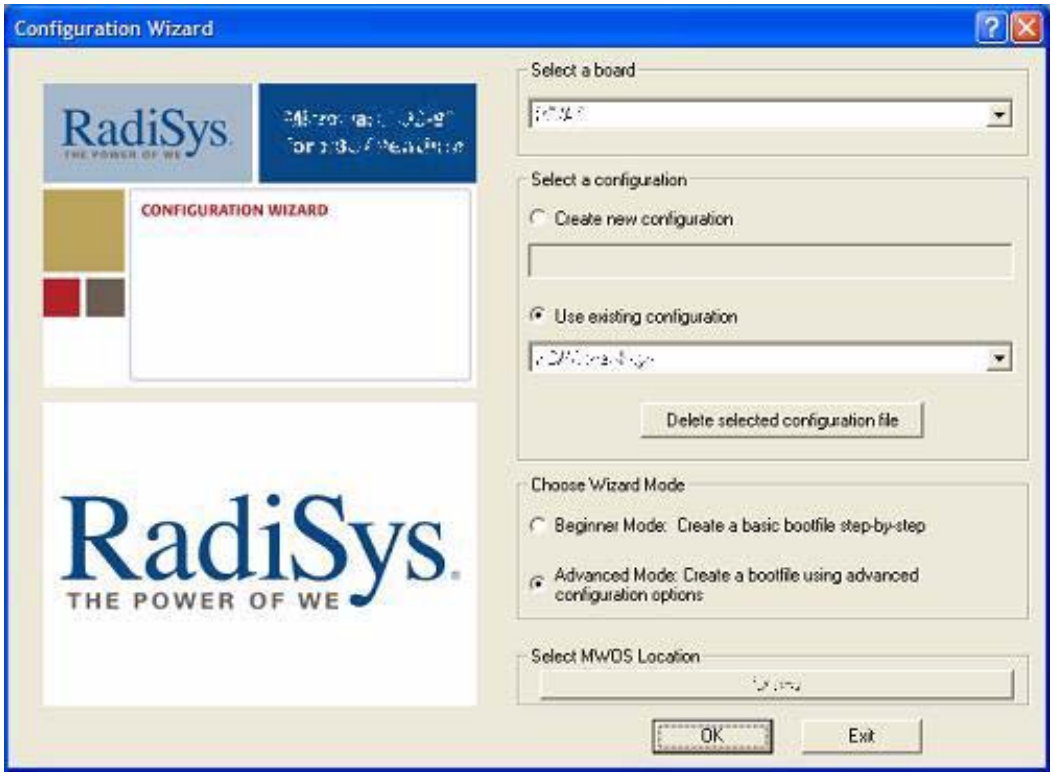
Building the Bootfile Image

The following sections detail the preferable method for building the bootfile image. This preferrable method includes building a bootable floppy disk using Microware's Configuration Wizard.

Starting the Configuration Wizard

Step 1. From the Windows desktop, select `Start` -> `RadiSys` -> `Microware OS-9 for <target>` -> `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.

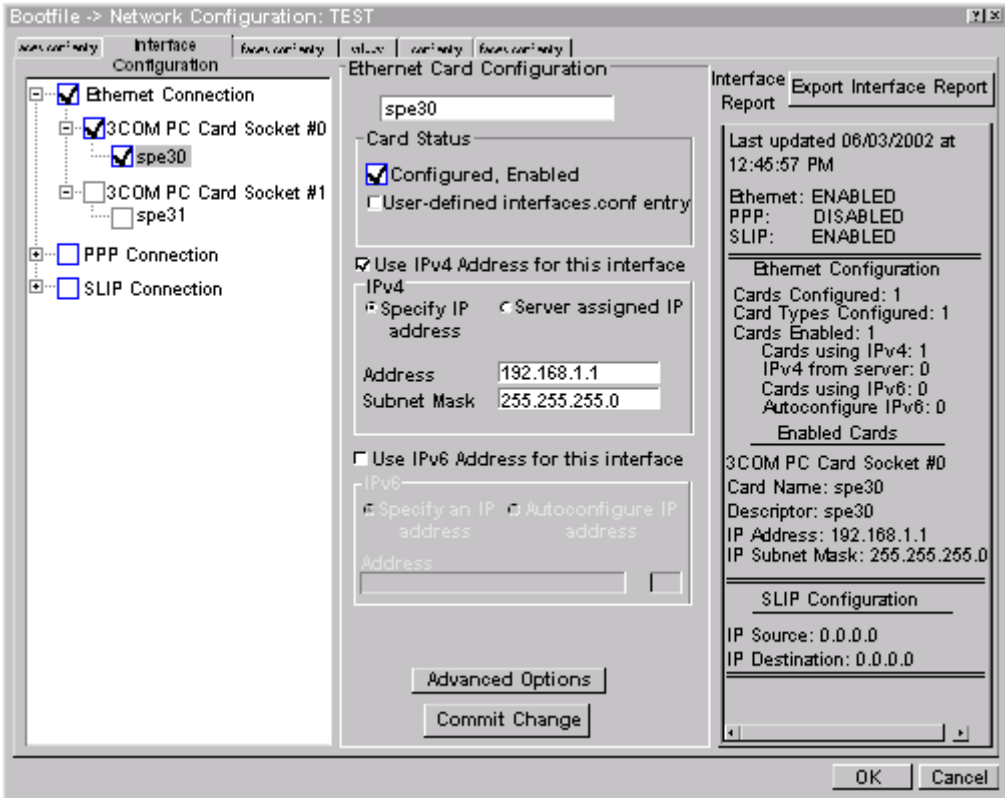
Step 3. Select the `Create new configuration` radio button from the **Select a configuration** menu and type in the name you want to give your ROM image in the supplied text box. This names your new configuration, which can later be accessed by selecting the **Use existing configuration** pull down menu.

Step 4. Select `Beginner Mode` and click `OK`. You are ready to begin preparing your floppy disk for the build.

Building the Bootable Floppy Disk

Once you have opened the Configuration Wizard in Beginner mode, the **Network Interface** dialog appears (shown in Figure 1-3). This window is where you will begin configuring your floppy disk for the build.

Figure 1-3. Bootfile -> Network Configuration -> Interface Configuration



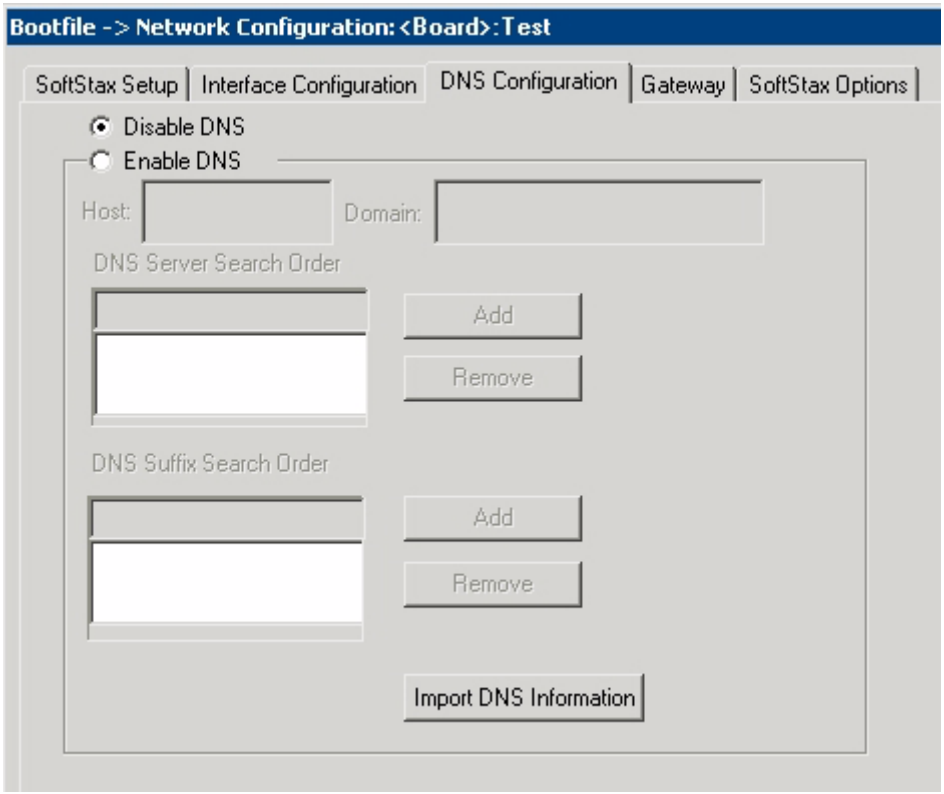
- To learn more about IPv4 and IPv6 functionalities, refer to the *Using LAN Communications* manual.
- Contact your system administrator if you do not know the network values for your board.

Step 1. Select your network adaptor model by scrolling though the choices under the Ethernet Connection area. If you do not want networking enabled, uncheck the **Ethernet Connection** box. Click `Next`.

To select a different type of interface, check the appropriate connection box and select the device and descriptor.

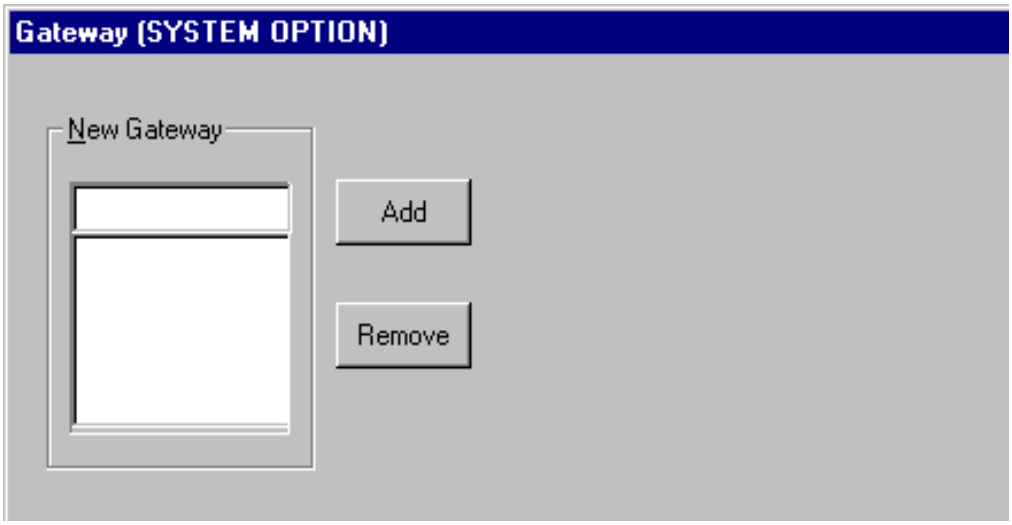
Step 2. The DNS Configuration window appears (shown in Figure 1-4). Select the `Enable DNS` radio button and fill in the appropriate values for your network. If you do not want DNS enabled, simply select the **Disable DNS** radio button. Click `Next`.

Figure 1-4. DNS Configuration Window



Step 3. The **Gateway** window appears (shown in Figure 1-5). Add in the appropriate information for your network. Click `Next`.

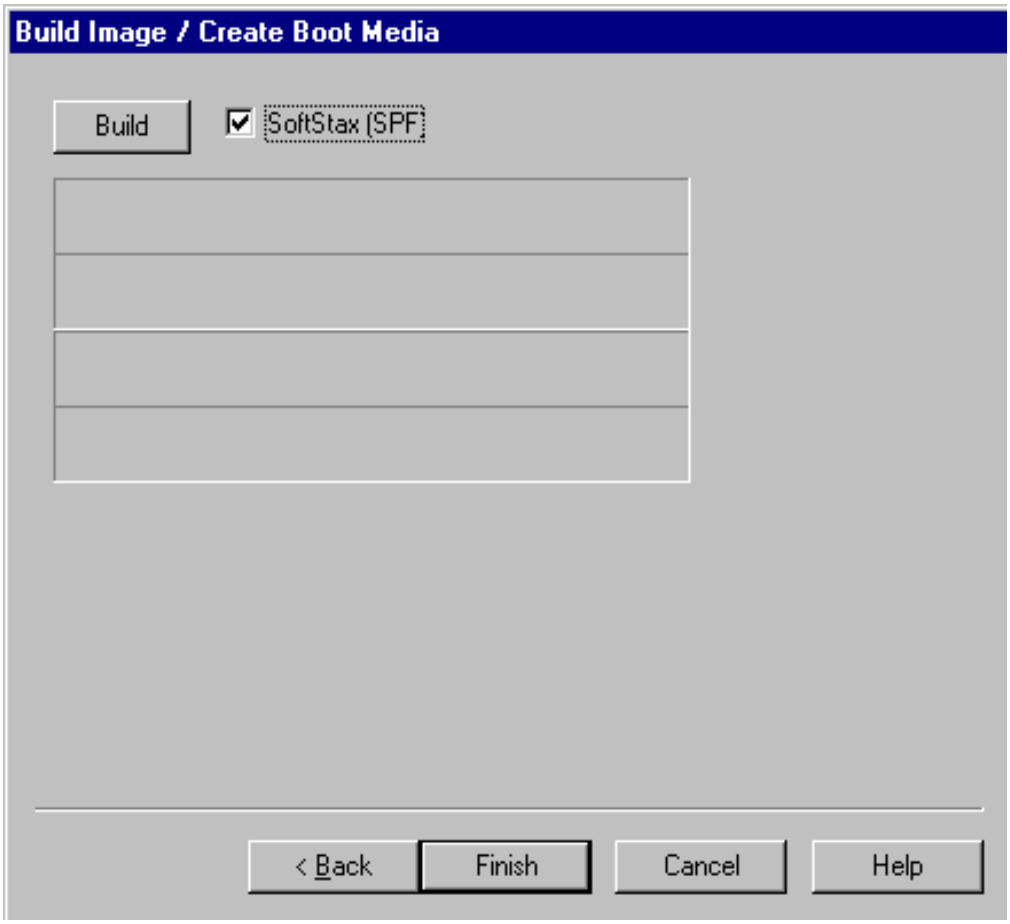
Figure 1-5. Gateway Window



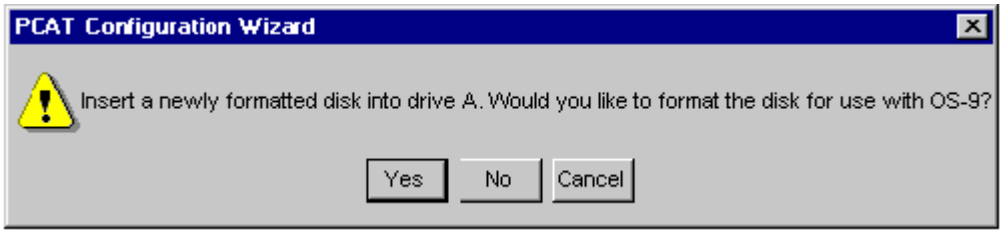
Contact your system administrator if you do not know the appropriate gateway values for your system.

Step 4. The **Build Image/Create Boot Media** window appears (shown in Figure 1-6). Select the **SoftStax®** check box if you would like SoftStax (networking) enabled and click `Build`.

Figure 1-6. Build Image Window



Step 5. When the build is complete, the following dialog box appears:



Insert a new floppy disk into your computer's floppy drive and click `Yes` to format it for OS-9 and copy the boot image.

Step 6. Click `Finish`. A dialog appears prompting you to save the file. The newly created boot floppy can be used to bring OS-9 up on the x86 target hardware. The default OS-9 console is the target's monitor.

If you enabled networking, you should be able to telnet into the target from your host PC. To do this, select `Run` from the Windows start menu and type the command `telnet <target hardware>`. The login user name is `super`; the password is `user`.

Building an OS-9 Image on the Target

This section describes using the floppy you built in the previous sections on the target machine.

Preparing the Hard Disk

The newly created boot floppy may be used to format a local hard disk with the OS-9 file system. A network connection between the OS-9 target machine and the Windows host computer may be used to load the OS-9 system files onto the hard disk.

Partitioning the Drive

Complete the following steps to partition the hard drive:

Step 1. Once the bootfile is read, the OS-9 console prompt appears ($). From the prompt, run `fdisk` by typing the following command: `fdisk -d=/hcfmt -e`

Step 2. The **Fdisk Options** menu appears. An example of this menu is shown below.

```
Current fixed disk device: /hc<n>fmt@
Choose operation to execute:
1.Create OS-9000 partition
2.Set active partition
3.Delete partition
4.Display partition information
5.Change Extended-Dos partition to OS-9 partition
6.Write master boot record (MBR)
```

It may be necessary to make room for the OS-9 partition. To do this, select `3` and delete an existing partition.

Step 3. Select `1` to create the OS-9 partition. The partition information is displayed. The **Enter the partition size in cylinders:** [ ] prompt appears at the bottom of the screen. Press `Enter` to accept the size.

Step 4. The following partition type options are displayed:

```
1. OS-9/386 type partition
2. Extended Type 41 partition
```

Select `1` and press `Enter`. The partition information is displayed again. Press `Esc` to return to the **Fdisk Options** menu.

Step 5. The next step is to make the partition active. Once returned to the **Fdisk Options** menu, select `2` to set the active partition. Press `Enter`.

Step 6. The partition information is displayed once again. When prompted, select the number that corresponds to the partition you would like to make active and press `Enter`.

Step 7. The partition information displays your new information. Press `Esc` to return to the Fdisk Options menu. At the menu, you can do one of two things:

- If the disk you are using is new and contains no other operating systems, proceed through step ten.
- If at least one other operating systems exists on your disk, proceed directly to the section, Formatting the Hard Drive.

Step 8. If the disk you are using contains no other operating system, you will need to write master boot record (MBR) to the disk. To do this, select `6` from the Fdisk Options menu.

Step 9. The display information prompts you to select the partition on which you wish to write the MBR. Once you have done this, press `Enter`. The Fdisk Options menu is displayed. Press `Esc` to exit this menu.

Step 10. The want to save new partition information prompt appears. Press `y` to save the partition information and press `Enter`. The partition information is written to the drive.

Formatting the Hard Drive

To format the hard drive, complete the following steps:

Step 1. Create the OS-9 RBF file system by running `format` from the console:

```
format /hc<n>fmt
```

where `<n>` is the partition number you created with `fdisk`.

Press `Enter`.

Step 2. The disk format utility parameters display. At the prompt, select `y` if you are ready to begin performing the selected partition. Press `Enter` to continue.

Step 3. From here you are prompted to enter the following information, respectively:

- physical format
- disk name
- physical verify

It is not typically necessary to complete a physical format and verify.

After entering the appropriate information, your formatting information is displayed and the OS-9 prompt returns.

FTP to the Target

This section discusses how to transfer the required files from the host to the target machine. There are two options for performing this transfer:

- Option 1. Copy the .tar archive to the target before transferring the files.
- Option 2. Do not copy the .tar archive to the target before transferring the files. This option is recommended for systems with limited disk space.

The tar archives are found in the RESIDENT directory on the product CD.

Option 1

Step 1. On the host machine, open the command prompt window.

Step 2. From the command prompt, move to the directory which contains the diskcache file. (This file is located in MWOS\OS9000\80386\CMDS.)

Step 3. Begin the FTP session with the target machine by typing the following command:

```
ftp <target>
```

Step 4. To log in to the system, type the appropriate username and password. (The username is super. The password is user.)

Step 5. At the next ftp prompt, type the following command to set binary mode: bin

Step 6. Move to the target hard drive directory by typing the following command:

```
cd hc<n>fmt
```

Step 7. To place the tar utility onto the target system, type the following command:

```
put tar
```

Step 8. To place the diskcache file into the target system, type the following command:

```
put diskcache
```

Step 9. Change the local directory to the RESIDENT directory of the installation CD-ROM:\

```
/cd Z:\RESIDENT
```

where Z: is your CD-ROM drive letter

Step 10. To place the MWOS directory tree into the target system, type the following command:

```
put mw86.tar
```

If disk space is limited, then you may wish to download mw86sm.tar in place of mw86.tar. mw86sm.tar includes a reduced set of commands and descriptors. The file mw86sm.tar can be loaded in a disk space smaller than eight megabytes.

Step 11. To exit the program, type quit.

Step 12. To turn on disk cache support from the OS-9 console, type the following command:

```
$ chd /hc<n>fmt ; load -d diskcache ; diskcache -e /hc<n>fmt=1024k
```

Step 13. Expand the system files by typing the following command:

```
$ load -d tar ; tmode nopause ; tar xvpf mw86.tar
```

The disk is now formatted and the OS-9 system files have been copied to disk.

To save space, delete the .tar file after extracting its contents.

Option 2

The following steps discuss the procedure for transferring required files from the host to the target machine without copying the tar archive to the target first. The tar archives are found in the directory named RESIDENT on the product CD.

Step 1. On the host machine, open the command prompt window.

Step 2. From the command prompt, move to the directory which contains the diskcache file. (This file is located in `MWOS\OS9000\80386\CMDS`.)

Step 3. Begin the FTP session with the target machine by typing the following command:

```
ftp <target>
```

Step 4. To log in to the system, type the appropriate username and password. (The username is `super`. The password is `user`.)

Step 5. At the next `ftp` prompt, type the following command to set binary mode: `bin`

Step 6. Move to the target hard drive directory by typing the following command:

```
cd hc<n>fmt
```

Step 7. Type the following command:

```
put tar
```

Step 8. To place the `diskcache` file into the target system, type the following command:

```
put diskcache
```

Step 9. Change the local directory to the `RESIDENT` directory of the installation CD-ROM.

```
/cd Z:\RESIDENT
```

where Z: is your CD-ROM drive letter

Step 10. Type the following command to send the tar archive into a pipe on the target. This command will start and then wait for you to complete this procedure's steps on the target:

```
send mw86.tar /pipe/mw86.tar
```

Step 11. On the target, change directories to the partition that will hold the contents of the tar archive:

```
chd /hc<n>fmt
```

Step 12. Extract the contents of the tar to the partition with the following commands:

```
load -d diskcashe
tmode nopause
diskcache -e /hc<n>fmt=1024k
tar -xvpf /pipe/mw86.tar
```

The tar archive is transferred from the host and extracted onto the hard drive. You will see a series of hash marks display on the host's monitor while the transfer is in progress.

If disk space is limited, you can download `mw86sm.tar` instead of `mw86.tar`. `mw86sm.tar` includes a reduced set of commands and descriptors. The file `mw86sm.tar` can be loaded in a disk space smaller than eight megabytes.

Step 13. To exit the FTP program from the host, type `quit`.

Step 14. The disk is now formatted and the OS-9 system files have been copied to disk.

Advanced Configurations

The following steps detail configuring the target system to boot from a local hard disk and to moving the OS-9 console to a serial port. This might be beneficial if you wish to remove the monitor or use it specifically for graphic applications.

Step 1. Open the Configuration Wizard in Advanced Mode. The Main Configuration window is displayed.

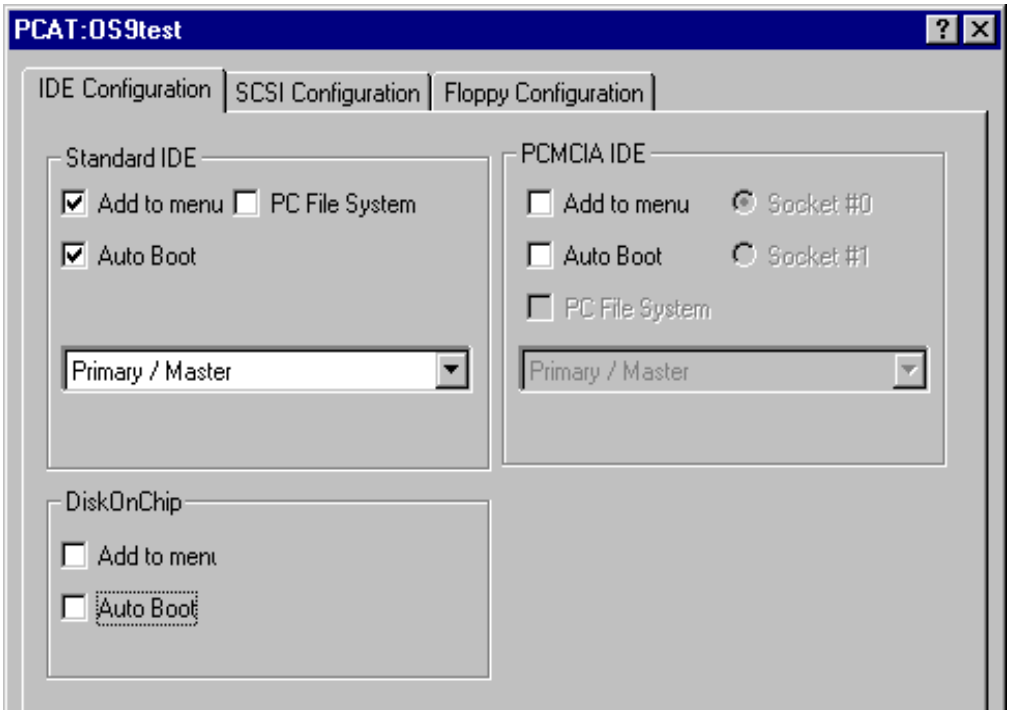
Step 2. Select `Configure` -> `Bootfile` -> `Configure System Options` from the menu.

Step 3. If you plan on using the monitor and keyboard as the OS-9 system console, leave the VGA/Keyboard radio button checked. Otherwise, click on the `COM1` radio button on the Define /term Port tab. This moves the high-level console to serial port one.

Step 4. Verify that the baud rate is set to `9600`. Click `OK`.

Step 5. Select `Configure` -> `Coreboot` -> `Disk Configuration`. The following window appears:

Figure 1-7. Disk Configuration Window-IDE Configuration tab

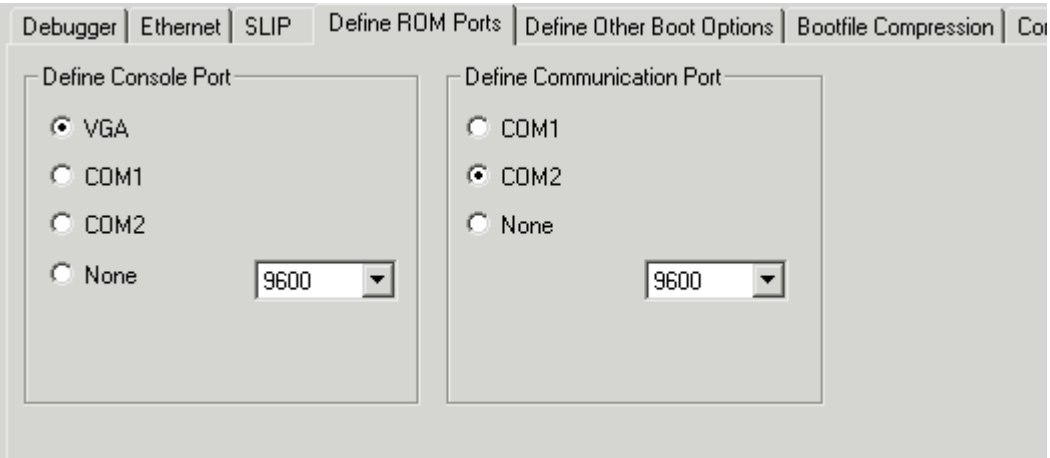


Step 6. Click the `Auto Boot` check box for Standard IDE and click `OK`.

Step 7. Select `Configure` -> `Coreboot` -> `Main Configuration` from the menu.

Step 8. Select the `Define ROM Ports` tab. The following window appears:

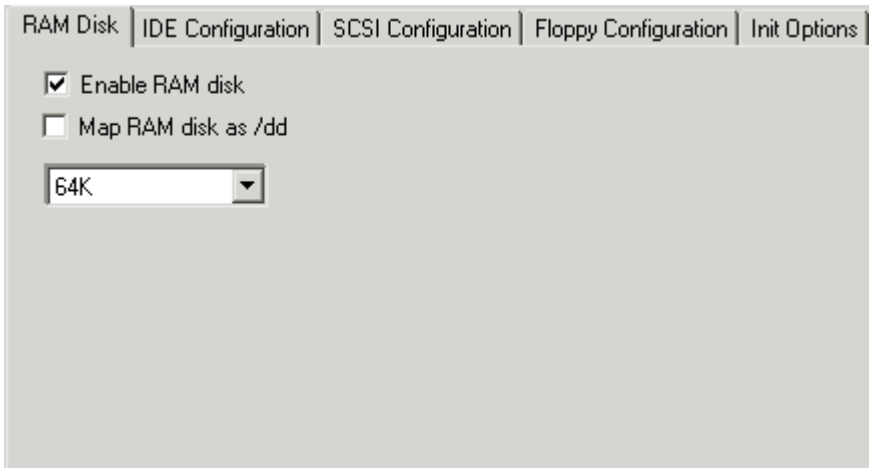
Figure 1-8. Define ROM Ports



Step 9. If you plan on using the monitor and keyboard as the OS-9 system console, leave the **VGA/Keyboard** radio button checked. Otherwise, click on the `COM1` radio button on the **Define Console Port** and **Define Communication Port** areas. This moves the high-level console to serial port one. Click `OK`.

Step 10. Select `Configure` -> `Bootfile` -> `Disk Configuration` from the menu.

Figure 1-9. RAM Disk

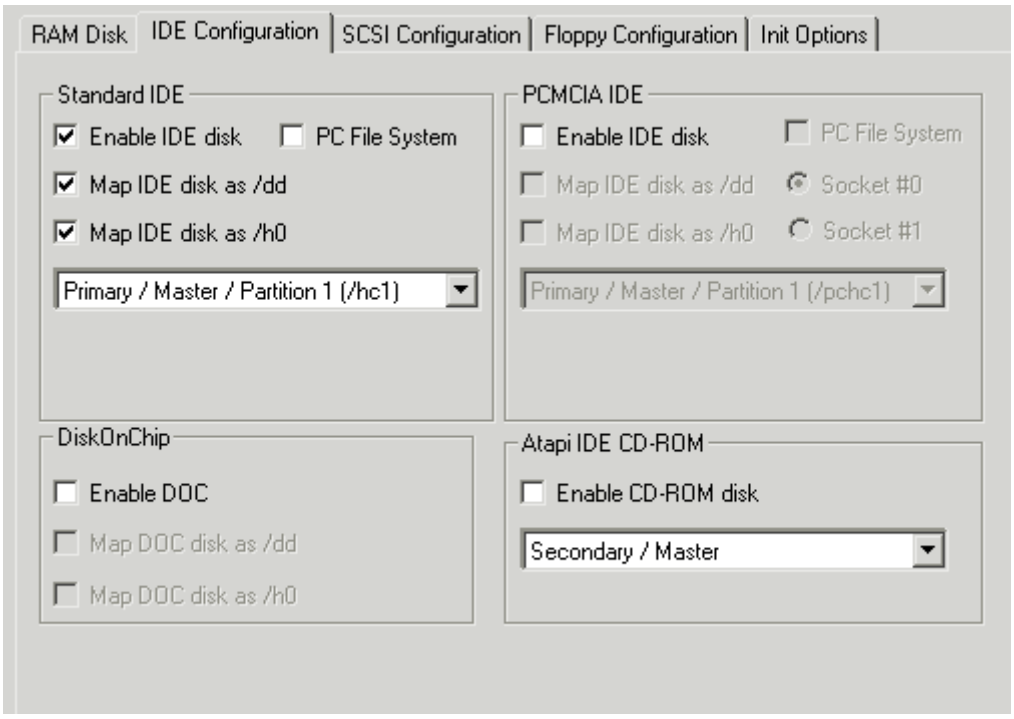


Step 11. From the **RAM Disk** tab, verify that the **Enable RAM disk** box is checked and other boxes are unchecked.

Step 12. Select the RAM disk size from the drop down list box. Use of a RAM disk is optional, and you may disable it by unchecking the **Enable RAM disk** box. If enabled, the RAM disk may be accessed as `/r0` on the target system.

Step 13. Select the **IDE Configuration** tab. The following window displays:

Figure 1-10. Disk Configuration Window-IDE Configuration tab



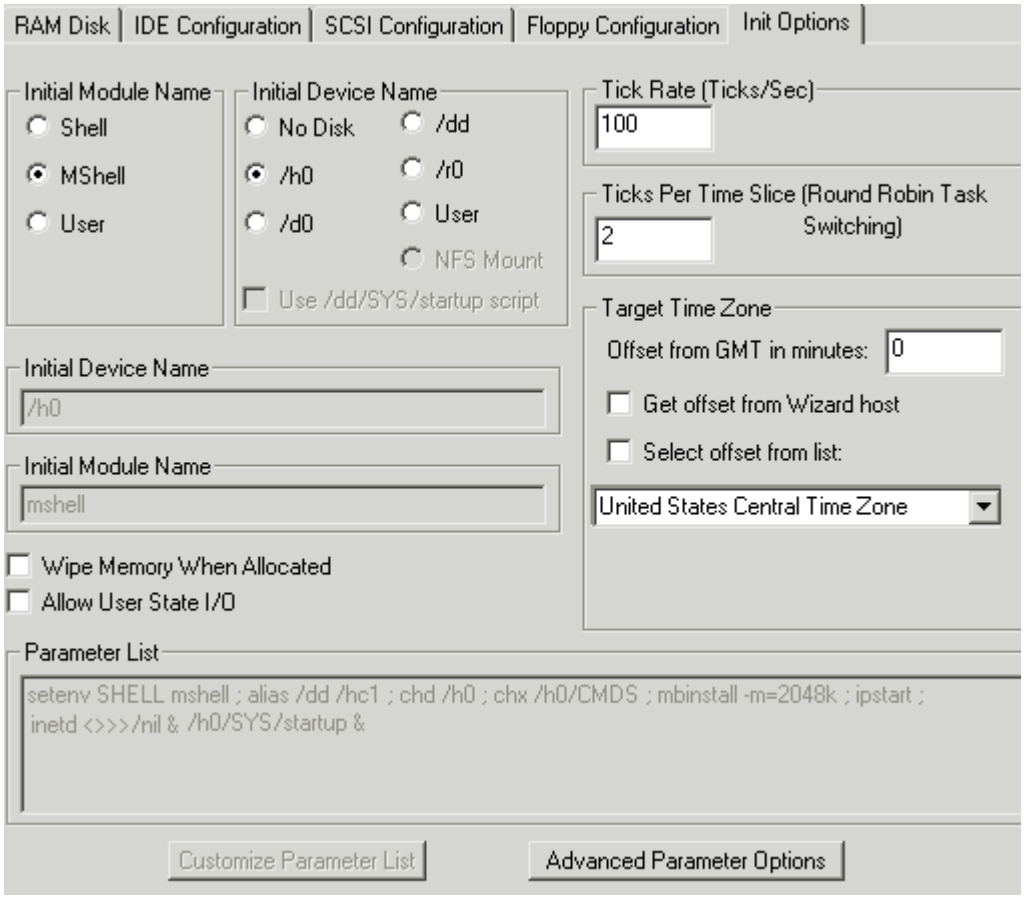
Step 14. Click the **Enable IDE disk**, **Map IDE disk as /dd**, and **Map IDE disk as /h0** check boxes to enable them.

The standard IDE hard disk will be accessed as device `/hc1` from the OS-9 console. The same device may also be accessed as `/h0` or `/dd`.

An IDE CD-ROM drive may be attached to the target system and accessed as device `/cd0`. The CD-ROM must be the master device on the second IDE channel.

Step 15. Click on the **Init Options** tab. The following window appears:

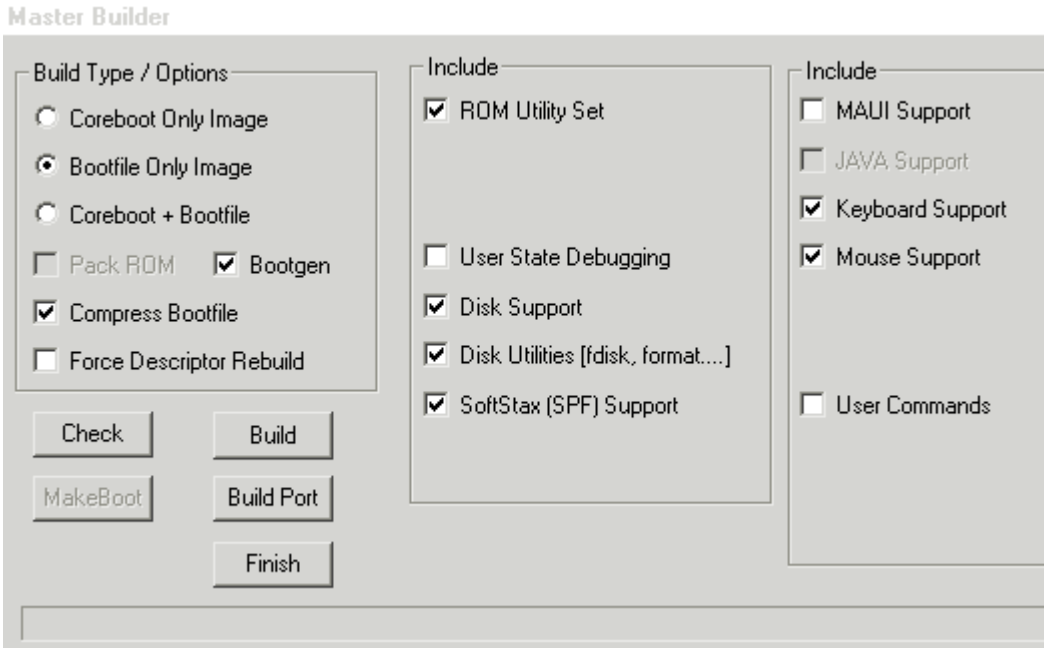
Figure 1-11. Disk Configuration Window-Init Options tab



Step 16. Select the `/h0` radio button to use the as the initial device. Click `OK`.

Step 17. Select `Configure` -> `Build Image` from the menu. The following window appears:

Figure 1-12. Master Builder Window



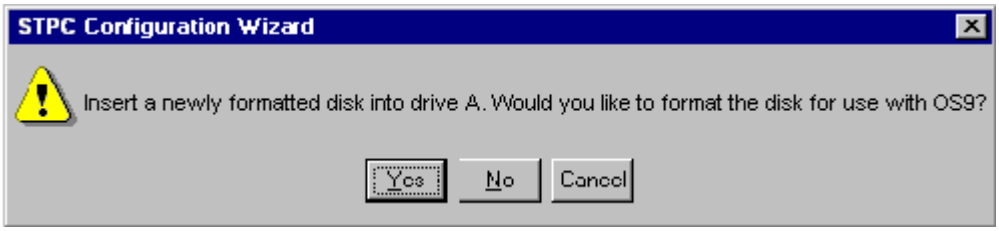
Step 18. Verify that the following options are checked:

- Bootgen
- Coreboot + Bootfile
- ROM Utility Set
- Disk Support
- Disk Utilities [fdisk, format...]
- SoftStax (SPF) Support Modules
- Keyboard Support
- Mouse Support (Enables support for a PS/2 style mouse)

Select the **User-State Debugging Modules** check box to include the Hawk debugging modules on the target system. Alternately, you may load and run the modules from the hard disk on the target.

Step 19. Click `Build` to create the OS-9 boot image.

Step 20. Click the `MakeBoot` button once it is enabled. The following dialog box appears:



Step 21. Insert a new floppy disk into your computer's floppy drive and click `Yes` to format it for OS-9 and copy the boot image.

Step 22. Once the boot image has been written to floppy, select `Finish`.

Step 23. Save your changes and exit the Configuration Wizard by selecting `File` -> `Exit`.

If you enabled networking, you should be able to telnet into the target from your host PC. To do this, select `Run` from the Windows start menu and type the following command:

```
telnet <target hostname>
```

The login user name is `super`; the password is `user`.

Configuring the Hard Drive

This section finishes the hard disk configuration by using the OS-9 `bootgen` utility to install a boot image onto the hard disk. The hard disk will be made bootable using the OS-9 boot image created in the previous sections.

Step 1. Change directories to your hard disk by typing the following command:

```
chd /hc<n>fmt
```

Step 2. Turn disk caching off by typing the following command at the OS-9 console:

```
$ diskcache -d /hc<n>fmt
```

Step 3. To perform a `bootgen` command on the new system, type the following command at the OS-9 console:

```
$ bootgen /hc<n>fmt -i=/d0/iplhdnoq -l=/d0/firstboot /d0/sysboot -nb400
```

Step 4. Remove the floppy from the drive and reboot the system. The system should boot from the hard disk, with the OS-9 system prompt appearing on the console.

Optional Procedures

The following sections discuss optional procedures you may perform once you have set up and configured the x86 board.

Setting Up OS-9 for BOOTP

A feature of Microware OS-9 for x86/Pentium is the ability to boot over Ethernet (BOOTP) using the `eb` option from the OS-9 boot menu. BOOTP is useful when a boot image is too big to fit on a single floppy. Often, users may place the initial "coreboot image" on one floppy and the "bootfile image" on a second floppy, but this method is not ideal. If MAUI and networking are selected, the boot image may be too large to fit even using this method.

OS-9 may boot from any RFC-compliant BOOTP server. Another option is to set up an OS-9 machine as a BOOTP server.

OS-9 BOOTP SERVER

OS-9 is a good choice for a BOOTP server. The `mw86.tar` file included with Microware OS-9 for x86/Pentium will install the `/h0/TFTPBOOT/bootptab` file to the

target system. The bootptab file may be edited to include any target information required.

Figure 1-13. Example entry for bootptab

```
#
# Example entry for bootptab:
#
ast:hd=/h0/tftpboot:ht=ethernet:ha=00609788CECE:ip=10.0.0.27
:sm=255.255.255.0:bs=auto:bf=bootfile:
#
```

Step 1. Use ftp to transfer a bootfile to the OS-9 BOOTP server. Make sure to place it in the /h0/TFTPBOOT directory.

Step 2. To allow bootp requests to gain access to the bootfile you must set public access, type the following command:

```
$ attr -prgr /h0/TFTPBOOT/bootfile
```

Step 3. Next, start the BOOTP server:

```
$ tftpd <>>>/nil&
$ bootpd /h0/TFTPBOOT/bootptab <>>>/nil&
```

You may run bootpd in the foreground, if desired, with the following command:

```
$ bootpd -d /h0/TFTPBOOT/bootptab
```

The -d option will allow you to see the BOOTP request as it happens.

Step 4. Next, perform a BOOTP operation to the target using the eb option in the initial OS-9 boot menu.

Executing Commands with the Wizard

One feature in the Wizard is the ability to execute commands at different phases of the build process. you can for example, perform an ftp command from the bootfile to the BOOTP server, using the Wizard. The following example will show what you need to do to set up the Wizard to perform this command to the created bootfile when the build is finished.

For PCAT:

Step 1. Edit the pcat.ini file located in the following directory:

```
mwos/OS9000/80386/PORTS/PCAT/BOOTS/INSTALL/INI
```

For MediaGX:

Edit the mediaGX.ini file located in the following directory:

```
mwos/OS9000/80386/PORTS/MEDIAGX/BOOTS/INSTALL/INI
```

Step 2. Look at the end of the file for the EXEC_AFTER_BUILD example code.

The BOOTP server IP address is located in the "PARAMS" string above.

Step 3. Next, create a `ftp.lst` file in `BOOTS\INSTALL\PORTBOOT`. Include the name and password required to login to the bootp server machine.

2 Board-Specific Reference

This chapter contains porting information that is specific to the x86 board.

x86 Utilities

The following sections describe utilities specifically written for x86.

abort

The `abort` utility is a p2module that may be used to allow the system to enter debug state once a non-maskable interrupt (NMI) is generated.

Usage:

```
$ p2init abort
```

cachechk

The `cachechk` utility may be used to verify L2 cache is working on a given system.

```
(Super)[/h0/>] cachechk

Memory Block  Transfer Speed  Access Time     Chart
------------  --------------  ------------    -----
      256     292.90 MB/s     3.41 ns/byte    ###
      512     301.04 MB/s     3.32 ns/byte    ###
     1024     305.30 MB/s     3.28 ns/byte    ###
     2048     307.10 MB/s     3.26 ns/byte    ###
     4096     307.63 MB/s     3.25 ns/byte    ###
     8192     307.10 MB/s     3.26 ns/byte    ###
    16384     301.30 MB/s     3.32 ns/byte    ###
    32768     176.19 MB/s     5.68 ns/byte    ######
    65536     174.72 MB/s     5.72 ns/byte    ######
   131072     173.43 MB/s     5.77 ns/byte    ######
   262144     164.04 MB/s     6.10 ns/byte    ######
   524288     153.46 MB/s     6.52 ns/byte    #######
  1048576      96.58 MB/s    10.35 ns/byte    ##########
  2097152      84.34 MB/s    11.86 ns/byte    ############
```

In the case above, there is an L1 cache size of 16K and an L2 cache size of 512K.

dmppci

The `dmppci` utility may be used to examine PCI configuration space.

Usage

```
(Super)[/h0/>] dmppci -?
Syntax: dmppci <bus_number> <device_number> <function_number> {<size>}
Function: dump PCI configuration space.
Options:
  none.
(Super)[/h0/>] dmppci 0 4 0

    PCI DUMP Bus:0 Dev:4 Func:0 Size:64
    -----------------------------------
VID  DID  CMD  STAT CLASS  RV CS IL IP LT HT BI MG ML SVID SDID
---  ---- ---- ---- -----  -- -- -- -- -- -- -- -- -- ---- ----
1013 00d6 0007 00a0 030000 03 00 0a 01 40 00 00 10 10 1013 8000
BASE[0]  BASE[1]  BASE[2]  BASE[3]  BASE[4]  BASE[5]  CIS_P    EXROM
-------- -------- -------- -------- -------- -------- -------- --------
e0000000 e2100000 00000000 00000000 00000000 00000000 00000000 00000000
Offset 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
-------------------------------------------------------
0000   13 10 d6 00 07 00 a0 00 03 00 00 03 00 40 00 00
0010   00 00 00 e0 00 00 10 e2 00 00 00 00 00 00 00 00
0020   00 00 00 00 00 00 00 00 00 00 00 00 13 10 00 80
0030   00 00 00 00 00 00 00 00 00 00 00 00 0a 01 10 10
```

gimmeio

`gimmeio` is an example trap handler and test program that demonstrates how to allow I/O port access in user-state programs. The `MWOS/OS9000/80386/PORTS/<BOARD>/UTILS/GIMMEIO` directory contains both the test program and trap handler source code.

`tcall.c`	OS-9/x86 trap handler source file
`thandler.c`	OS-9/x86 trap handler source file
`trapc.a`	OS-9/x86 trap handler source file
`ttest.c`	example test program source code
`makefile`	makefile for creating test program and trap handler
`<BOARD>/CMDS/gimmeio`	system-state trap handler module
`<BOARD>/CMDS/ttest`	user-state test program
`<BOARD>/LIB/gimmeio.l`	`gimmeio` trap handler library

`ttest.c` is a example of how to call the trap handler in order to be granted access to performing I/O in user-state.

In order for a user-state program to be granted I/O port access by GIMMEIO, the user-state program module must be supervisor state (owned by 0.x).

Although the GIMMEIO trap handler was required as of v2.1 of OS-9 for x86, you may now allow I/O access system wide if desired, by selecting Allow User-State I/O in the Configuration Wizard's Init Options dialog box.

loop

The loop command may be used to create repetitive commands.

Usage: loop -?

Usage: loop [-t] [-n<count>] [-m] [-s<count>] [<prog>] [..<progx>]

-t	reports time used
-x	displayed if error show value
-i	do not exit on errors
-n	loop count
-s	sleep for count
-m	sleep count is in milliseconds : default is seconds

Example

Create a file and test for the file removal. Check once every two seconds.

```
Super)[/h0/>] copy SYS/startup -w=/r0
copying SYS/startup to /r0/startup
(Super)[/h0/>] loop -t -x -s2 "dir /r0/startup >>>/nil"
 98/11/08 22:52:25  up for:  0 days  0 hours  0 minutes 0 seconds
Wait 2 Seconds
 98/11/08 22:52:27  up for:  0 days  0 hours  0 minutes 2 seconds
Wait 2 Seconds
 98/11/08 22:52:29  up for:  0 days  0 hours  0 minutes 4 seconds
Wait 2 Seconds
 98/11/08 22:52:31  up for:  0 days  0 hours  0 minutes 6 seconds

Wait 2 Seconds
 98/11/08 22:52:33  up for:  0 days  0 hours  0 minutes 8 seconds
Wait 2 Seconds
000:216 (E_PNNF)   File not found.
Error #000:216 (E_PNNF)   File not found.
 The pathlist does not lead to any known file.
```

mouse

The `mouse` utility is provide as a example of how to access the mouse from user programs. Source is included.

```
(Super)[/r0/>] mouse
Opening device /m0
status = 0x18, x = 255, y =   0  X Negative
status = 0x18, x = 253, y =   0  X Negative
status = 0x18, x = 253, y =   1  X Negative
status = 0x18, x = 251, y =   0  X Negative
status = 0x18, x = 252, y =   1  X Negative
status = 0x18, x = 250, y =   0  X Negative
status = 0x18, x = 250, y =   1  X Negative
status = 0x18, x = 251, y =   0  X Negative
status = 0x18, x = 252, y =   0  X Negative
status = 0x18, x = 252, y =   0  X Negative
status = 0x18, x = 254, y =   0  X Negative
status = 0x18, x = 254, y =   0  X Negative
status = 0x08, x =   2, y =   0
status = 0x08, x =   3, y =   0
status = 0x08, x =   4, y =   0
```

pciv

The `pciv` utility allows viewing all PCI devices in the system.

Usage: pciv -?

pciv- PCI Configuration Space browser.

Options:

-a	show base address info and size
-r	show PCI routing information
?	display help

```
(Super)[/h0/>] pciv

BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-----------------------------------------------
000:00:00  8086 1250 0006 2200 060000 03 00 00 00 Bridge Device [S]
000:02:00  1011 0022 0107 0280 060400 03 08 00 00 Bridge Device [S]
000:03:00  8086 1229 0007 0290 020000 04 08 0a 01 Network Controller [S]
000:04:00  1013 00d6 0007 00a0 030000 03 00 0a 01 Display Controller [S]
000:07:00  8086 7000 000f 0280 060100 01 00 00 00 Bridge Device [M]
```

```
000:07:01  8086 7010 0005 0280 010180 00 00 00 00 Mass Storage Controller
[S]
001:13:00  10b7 9000 0007 0200 020000 00 00 0a 01 Network Controller [S]


(Super)[/h0/>] pciv -a


BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:00:00  8086 1250 0006 2200 060000 03 00 00 00
Bridge Device [S]
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:02:00  1011 0022 0107 0280 060400 03 08 00 00
(NC) base_addr[2] = 0x40010100  PCI/IO  0x40010100 Size = 0x00000004
(C)  [32-bit] base_addr[3] = 0x2280e1e1  PCI/IO  0x2280e1e0 Size =
0x00000010
(C)  [32-bit] base_addr[4] = 0xdff0d800  PCI/MEM 0xdff0d800 Size =
0x00000010
(C)  [32-bit] base_addr[5] = 0xaff1a801  PCI/IO  0xaff1a800 Size =
0x00000010
Bridge Device [S]
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:03:00  8086 1229 0007 0290 020000 04 08 0a 01
(NC) [32-bit] base_addr[0] = 0xe2110008  PCI/MEM 0xe2110008 Size =
0x00001000
(C)  [32-bit] base_addr[1] = 0x00006001  PCI/IO  0x00006000 Size =
0x00000020
(C)  [32-bit] base_addr[2] = 0xe2000000  PCI/MEM 0xe2000000 Size =
0x00100000
Network Controller [S]
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:04:00  1013 00d6 0007 00a0 030000 03 00 0a 01
(C)  [32-bit] base_addr[0] = 0xe0000000  PCI/MEM 0xe0000000 Size =
0x02000000
(C)  [32-bit] base_addr[1] = 0xe2100000  PCI/MEM 0xe2100000 Size =
0x00010000
Display Controller [S]
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:07:00  8086 7000 000f 0280 060100 01 00 00 00
Bridge Device [M]
```

```
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:07:01  8086 7010 0005 0280 010180 00 00 00 00
(C)  [32-bit] base_addr[4] = 0x0000f001  PCI/IO  0x0000f000 Size =
0x00000010
Mass Storage Controller [S]
BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
001:13:00  10b7 9000 0007 0200 020000 00 00 0a 01
(C)  [32-bit] base_addr[0] = 0x0000e001  PCI/IO  0x0000e000 Size =
0x00000040
Network Controller [S]


(Super)[/h0/>] pciv -r


ELCR-EDGE/LEVEL CONTROL REGISTER


INT CNTRL-1 - [0x000004d0] = 0x00
INT00 INT01 INT02 INT03 INT04 INT05 INT06 INT07
edge  edge  edge  edge  edge  edge  edge  edge
INT CNTRL-2 - [0x000004d1] = 0x04
INT08 INT09 INT10 INT11 INT12 INT13 INT14 INT15
edge  edge  level edge  edge  edge  edge  edge
INTERRUPT CONTROLLER STATUS [PIC-8259]
OCW1 - OPERATIONAL CONTROL WORD 1 REGISTER
INT CNTRL-1 - [0x00000021] = 0xf8
INT00 INT01 INT02 INT03 INT04 INT05 INT06 INT07
on    on    on    off   off   off   off   off
INT CNTRL-1 - [0x000000a1] = 0xbb
INT08 INT09 INT10 INT11 INT12 INT13 INT14 INT15
off   off   on    off   off   off   on    off
```

pcmcia

The `pcmcia` utility provides a means to insert and remove PCMCIA devices. Source is provided for `pcmcia` so you may add support for their own cards.

Usage: pcmcia -?

Syntax: pcmcia [<opts>]

Function: initialize PCMCIA socket

Options:

-s=socket	socket [default all sockets]
-d	deinitialize socket(s)
-i	initialize socket(s)
-v	verbose mode
-x	dump CIS/Config information
-?	print this help message

```
(Super)[/r0/>] pcmcia -iv
MICROWARE PCMCIA SOCKET SERVICES
i82365sl step B PCMCIA type controller
socket #1 occupied [0xff]
v1_Major = 4 v1_Minor 1
Manufacture Name String = EXP
Additional Info String = CD+GAME
Product Name String = C1
IDE Base 0x00000360 : Vector 0

(Super)[/r0/>] chd /pcmhe1
 (Super)[/pcmhe1/>] free
"pcmhe1"
Capacity: 43967 blocks, 1373.968 Mbytes
Free: 28681 blocks, 896.281 Mbytes
Largest Free Block: 13036 blocks, 407.375 Mbytes

(Super)[/r0/>] pcmcia -d -s=1

socket1:  occupied
It is now save to remove the card is socket #01

MWOS/OS9000/80386/PORTS/<BOARD>/ROM/config.des

#define LLCIS_PORT"cbase=0xd4000"
#define LLCIS_PARAMS"verbose=1 fixed=1"
#define IDE_CIS_PARAMS"ide0=0x320,0 ide1=0x360,0"
#define ETH_CIS_PARAMS"3com=0x340,3"
#define SERIAL_CIS_PARAMS"com=0x340,10"
```

The PCMCIA SOCKET SERVICES require a VADEM 465 or similar controller.

- i82365sl step A
- i82365sl step B
- VLSI 82C146

Early versions of this chip will only work with one socket due to chip bug.

- IBM
- Vadem
- Cirrus CLPD67xx

The PCMCIA SOCKET SERVICES do not use interrupts and for IDE based devices no interrupts are used by default. If the PCMCIA device does not work check what is reported during the boot process. The type of PCMCIA controller as well as the device information is displayed. It may be that another device is using the memory at 0xd4000. If this is the case change the value in `config.des`. The Wizard will use this value next time you create a boot image.

pinfo

The `pinfo` utility may be used to access DOS extended partitions by providing the required information to create descriptors. The extended partitions are displayed as well if the partition may be used with OS-9. Note that RadiSys currently only supports this utility with IDE devices. SCSI devices are not supported. You may create or modify a existing descriptor with the values shown in the Extended partition section. The LUN and LSNOFFS fields should reflect the values shown.

```
Super)[/h0/>] pinfo -?
Syntax:   pinfo {</device>}
Function: show disk partition information
Options:none.

(Super)[/h0/>] pinfo /hcfmt@
================ Primary  Partitions =====================

Partition   LUN  LSNOFFS     Par_Type                 FMGR
-----------------------------------------------------------
01           01  0x00000000  OTHER                    NA
02           02  0x00000000  OS/2 Boot Manager        NA
03           03  0x00000000  DOS Extended             NA
04           04  0x00000000  OS-9000                  RBF

================ Extended Partitions =====================

Partition   LUN  LSNOFFS     Par_Type                 FMGR
-----------------------------------------------------------
01           02  0x0036c180  Linux native             NA
```

setpci

The `setpci` utility may be used to change PCI information in the PCI configuration space. You may also use `setpci` to examine information in the PCI configuration space. This should help you develop PCI drivers and applications.

Usage

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

Function

Set/Read PCI configuration space.

Options

All command line arguments are required, save that the presence or absence of <value> indicates whether to read or write the specified information.

<bus>PCI bus number (0..255)

<dev>PCI device number (0..32)

<func>PCI function number (0..7)

<offset>offset value (e.g. 4 for command register offset)

<size>size (b = byte, w = word, d = dword)

<value>if present, the specified value will be written; if not present, the value will be read

Example

```
$ setpci 1 13 0 0x10 d
PCI READ MODE
-------------
PCI Value.....0x0000e001 (dword) READ
PCI Bus.........0x01
PCI Device......0x0d
PCI Function....0x00
PCI Offset....0x0010
```

symbios_info

When using the Symbios SCSI controller family of cards, you may use this utility to see how drives in the system have been configured.

The `symbios_info` utility provides a simple means to determine how the current SCSI drive parameters have been utilized. Symbios 810-875 controllers are supported.

The device should be inized prior to using this utility. On initial access of any device, the information is stored in the SCSI internal threads. The `Symbios_info` function will examine the thread information.

Due to the nature of the `Symbios_info` utility changes to the Symbios driver may cause this program to fail. The `Symbios_info` utility should be re-compiled anytime the driver changes. Although the `Symbios_info` utility is mainly used to see how the drive in use is set up, advanced information is also included to help determine any problems with using SCSI drives. Most problems with SCSI are normally termination related. As newer drives become available, expect to see problems that require software related changes.

Syntax

symbios_info [<opts>]

Options

-sshow information

-rshow registers

-dshow DSP information

-tshow time information

Examples

In the basic information mode, `symbios_info` displays the interrupt vector information, the type of Symbios controller found, and the negotiation information. For synchronous negotiations, the drive requested time information as well as the actual negotiated time used is displayed.

```
$ iniz hs02; dir /hs02; symbios_info

PC-AT Compatible 80386  OS-9000 V2.2 for Intel x86
 vector  ($)   prior  drivstat   irq svc    driver
-------------  -----  ---------  ---------  ------

    74  ($4a)     10  $00f90fb4  $0013e6f1  scsi8xx

Symbios 53C875 [Symbios Device ID = 0x0f]

[00] [0c:0f] final [0f:0f] ULTRA WIDE SCSI 33.3 MB/s (60 ns, offset 15)
```

The show information option will display the current thread states.

```
$ iniz hs02; dir /hs02; Symbios_info -s
PC-AT Compatible 80386  OS-9000 V2.2 for Intel x86

 vector  ($)   prior  drivstat   irq svc    driver
```

```
------------  -----  ---------  ---------  ------
    74  ($4a)     10  $00f90fb4  $0013e6f1  scsi8xx

Dump Threads.... lst @ 0x00f866a0

lst->wakes = 0
chip free
id=00 Ent_WHICHPHASE Thread has completed operation
CMD[2a00000fffffffa80300000100 CMD_STATUS[00000000]
MSGI[00000000] MSGO[c00103010f0f0c100000000000000000]
lth->synctried = Yes
lth->widetried = Yes
lth->processid 00000003
lth->thread_sem is free
lth->xferflags 0000002d SCSI_ATN SCSI_SYNC SCSI_WIDE SCSI_ULTRA
lst->sbclmaster 00000098 lth->sbclmask 00000098 sxfr 2f scntl3 9d
id=01 Thread not in use
id=02 Thread not in use
id=03 Thread not in use
id=04 Thread not in use
id=05 Thread not in use
id=06 Thread not in use
id=07 Ent_WAITFORRESELECT SCSI SELFID
id=08 Thread not in use
id=09 Thread not in use
id=0a Thread not in use
id=0b Thread not in use
id=0c Thread not in use
id=0d Thread not in use
id=0e Thread not in use
id=0f Thread not in use
```

The show registers option will show the current Symbios internal registers. Not all registers are displayed, only registers that are safe to display. Use this option with care. The SCSI bus should be idle when using this option.

```
$ iniz hs02; dir /hs02; symbios_info -r
PC-AT Compatible 80386  OS-9000 V2.2 for Intel x86

 vector  ($)   prior  drivstat   irq svc    driver
------------  -----  ---------  ---------  ------
    74  ($4a)     10  $00f90fb4  $0013e6f1  scsi8xx
```

```
Location      Value         Register  Status
-------------------------------------------
0xe8001000    [d0]          SCNTL0    ARB1 ARB0 WATN
0xe8001001    [00]          SCNTL1
0xe8001002    [00]          SCNTL2
0xe8001003    [55]          SCNTL3    SCF2 SCF0 CCF2 CCF0
0xe800100a    [80]            SCID    RES
0xe8001005    [00]           SXFER
0xe8001006    [07]            SDID    ENC2 ENC1 ENC0
0xe8001007    [0f]           GPREG    GPIO3 GPIO2 GPIO1 GPIO0
0xe8001008    [00]            SFBR
0xe8001009    [00]            SOCL
0xe800100a    [80]            SSID    VAL
0xe800100b    [00]            SBCL
0xe800100d    [00]          SSTAT0
0xe800100e    [0f]          SSTAT1    SDPOL MSG C/D I/O
0xe800100f    [0a]          SSTAT2    SPL1 LDSC
0xe8001010    [0000058f]       DSA
0xe8001014    [00]           ISTAT
0xe8001018    [00]          CTEST0
0xe8001019    [f0]          CTEST1    FMT3 FMT2 FMT1 FMT0
0xe800101a    [35]          CTEST2    CIO CM TEOP DACK
0xe800101b    [31]          CTEST3    V1 V0 WRIE
0xe800101c    [b2ac61c9]      TEMP
0xe8001020    [00]           DFIFO
0xe8001021    [00]          CTEST4
0xe8001022    [24]          CTEST5    DFS BL2
0xe8001024    [00f86a68]       DBC
0xe8001027    [54]            DCMD
0xe8001028    [00240000]      DNAD
0xe800102c    [00000008]       DSP
0xe8001030    [0000058f]      DSPS
0xe8001034    [00]        SCRATCH0
0xe8001035    [00]        SCRATCH1
0xe8001036    [80]        SCRATCH2
0xe8001037    [00]        SCRATCH3
0xe8001038    [8e]           DMODE    BL1 ERL ERMP BOF
0xe8001039    [25]            DIEN    BF SIR IID
0xe800103a    [00]             SBR
0xe800103b    [81]           DCNTL    CLSE COM
```

```
0xe800103c    [b2ac61c9]     ADDER
0xe8001040    [8f]           SIEN0    M/A SGE UDC RST PAR
0xe8001041    [05]           SIEN1    ST0 HTH
0xe8001044    [11]           SLPAR
0xe8001046    [70]          MACNTL    TYP2 TYP1 TYP0
0xe8001047    [0f]          GPCNTL    GPIO3 GPIO2 GPIO1 GPIO0
0xe8001048    [0e]          STIME0    SEL3 SEL2 SEL1
0xe8001049    [00]          STIME1
0xe800104a    [80]         RESPID0
0xe800104b    [00]         RESPID1
0xe800104c    [77]          STEST0    SSAID2 SSAID1 SSAID0 ART SOZ SOM
0xe800104d    [0c]          STEST1    DBLEN DBLSEL
0xe800104e    [00]          STEST2
0xe800104f    [80]          STEST3    TE
0xe8001058    [00]            SBDL
0xe8001059    [00]           SBDL1
0xe800105c    [ffffffff]  SCRATCHB
```

The show DSP option displays the current Symbios scripts location; it is useful when and if the SCSI bus locks. The information obtained will help to deal with drives that appear to have problems. If a SCSI drive appears to hang, you can load the `symbios_info` utility and run it after the hang to see the state of the scripts.

Tech support can use this information to determine what the drive is doing or not doing. The section of the dump shown may be compared to the v53c810.lst file.

```
$ iniz hs02; dir /hs02; symbios_info -d
PC-AT Compatible 80386  OS-9000 V2.2 for Intel x86

 vector  ($)   prior  drivstat   irq svc    driver
-------------  -----  ---------  ---------  ------
    74  ($4a)     10  $00f90fb4  $0013e6f1  scsi8xx

Script dsp @ 0xe8002018

00000018: 80880000 000002c4
00000020: 74011000 00000000
00000028: 808c0010 00000028
00000030: 741a4000 00000000
00000038: 808c0040 00000008
00000040: 80880000 fffffb8
00000048: 98080000 00000090
00000050: 80880000 0000028c
```

The show time option will show the current Symbios setup for the controller used.

```
$ iniz hs02; dir /hs02; symbios_info -t
PC-AT Compatible 80386  OS-9000 V2.2 for Intel x86

 vector  ($)   prior  drivstat   irq svc    driver
-------------  -----  ---------  ---------  ------
    74  ($4a)     10  $00f90fb4  $0013e6f1  scsi8xx
Symbios 53C875 [Symbios Device ID = 0x0f]

[00] [0c:0f] final [0f:0f] ULTRA WIDE SCSI 33.3 MB/s (60 ns, offset 15)

Driver is PCI I/O mapped

Symbios Clock     [0x00000050] (80)
Core Clock        [0x00000014] (20)
Min Period        [0x0000000c] (12)
Max Offset        [0x00000010] (16)
I/O Base          [0x0000e400]
Memory Base       [0xe8001000]
RAM Base          [0xe8002000]
Script            [0xe8002000] size (1548)
Selfid            [0x00000007]
Irq Level         [0x00000000]
Irq Vector        [0x0000004a]
Irq Priorty       [0x0000000a]

SCSI controller supports SCSI Wide 16

Special Features:

Clock Doubler Enabled
SCSI Large FIFO enabled size = 536
Burst Rate = 128
Burst Op Code Fetch Enabled
PCI Read Line Enabled
PCI Read Multiple Enabled
Write and Invalidate Enabled
PCI Cache Line Size Enabled
```

testpci

The `testpci` utility provides a means to test the PCI library calls. Source is provide so that you have examples of all of the PCI calls available. See the <Bold><links> PCI Configuration Information section for information on the PCI call usage.

Example

```
$ testpci
Test PCI Library Calls Edition 3
_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.
```

vidbios

The `vidbios` utility shows how to use the INT10h trap handler. You may either use the `vidbios` utility or incorporate the functionality in their own programs by studying the code in

`MWOS/OS9000/80386/PORTS/<BOARD>/UTILS/VIDBIOS.`

The `vidbios` utility allows setting specific video mode using INT10h on video cards. Some video cards may not function correctly with the `vidbios` utility due to the protected nature of OS-9.

Usage

vidbios [<options>]

Function

Make 16-bit int10h video BIOS call

Options

One or more of the following options must be specified. Options default to a value of zero if not specified.

```
-eax=0xhhhhhhhhvalue to load into eax for BIOS call
-ebx=0xhhhhhhhhvalue to load into ebx for BIOS call
-ecx=0xhhhhhhhhvalue to load into ecx for BIOS call
-edx=0xhhhhhhhhvalue to load into edx for BIOS call
-ebp=0xhhhhhhhhvalue to load into ebp for BIOS call
-esi=0xhhhhhhhhvalue to load into esi for BIOS call
-edi=0xhhhhhhhhvalue to load into edi for BIOS call
-r print register state after BIOS call
```

ROM Utilities and Special Booters

llkermit

The llkermit ROM booter allows booting OS-9 over serial using Kermit Protocol. You must select llkermit in the ROM options when creating the boot image. Once the menu is displayed type `ker`. You should now be able to send the image on the communications port.

llcis

The llcis ROM sub-booter allows PCMCIA devices to be initialized for use.

The PCMCIA utility shares the same configuration information as the llcis sub-booter.

```
MWOS/OS9000/80386/PORTS/<BOARD>/ROM/config.des
#define LLCIS_PORTcbase=0xd4000"
#define LLCIS_PARAMS"verbose=1 fixed=1"
#define IDE_CIS_PARAMS"ide0=0x320,0 ide1=0x360,0"
#define ETH_CIS_PARAMS"3com=0x340,3"
#define SERIAL_CIS_PARAMS"com=0x340,10"
```

The PCMCIA SOCKET SERVICES require a VADEM 465 or similar controller.

- i82365sl step A
- i82365sl step B
- VLSI 82C146 - Note. Early versions of this chip will only work with one socket due to chip bug.
- IBM
- Vadem
- Cirrus CLPD67xx

The PCMCIA SOCKET SERVICES do not use interrupts, and for IDE based devices, no interrupts are used by default. If the PCMCIA device does not work check what is reported during the boot process. The type of PCMCIA controller as well as the device information is displayed. It may be that another device is using the memory at 0xd4000. If this is the case change the value in config.des. The Wizard will use this value next time you create a boot image.

rpciv

ROM based version of the `pciv` utility. The `rpciv` utility is provided for debugging purposes before the system boots.

MAUI Graphics Support

This section details information for using MAUI (Multimedia Application User Interface) for the x86 board.

Getting Started

To start MAUI from the OS-9 console, complete the following steps:

Step 1. From the command prompt, navigate to the `/h0/SYS` directory.

Step 2. At the prompt, type the following command:

```
loadmaui
```

To verify that MAUI is running, try executing one of the demo programs, such as `fdraw` or `fcopy`, from the OS-9 console.

Configuring the Display

The following code fragments, from the `loadmaui` file, configure OS-9 for Generic VGA mode 13 graphics support. Video mode 13 works with most every graphics card, but does not provide the best resolution. You may want to comment out the mode 13 driver by placing an asterisk in front of each line, and uncomment one of the other video drivers such as the generic VESA driver or the ISA Bank driver. On the OS-9 target system you may edit the `loadmaui` file using the umacs editor.

```
*
* Graphics card selections.
*
* Note: The cdb default is PS2 mouse. To use serial mouse
* select the "_s" version.
*
* MAUI port - Generic VGA mode 13 ( 320x200x8bpp )
Remove the leading asterisk from one cdb_ file, vga and gx_vga files to
enable Generic VGA mode 13 video.
*
load -d CMDS/BOOTOBJS/MAUI/cdb_vga - PS/2 mouse
```

```
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_s - Serial mouse
load -d CMDS/BOOTOBJS/MAUI/vga
load -d CMDS/BOOTOBJS/MAUI/gx_vga
*
* MAUI port - Generic VGA mode 12 & "X" ( 640x480x4bpp & 360x480x8bpp )
Remove the leading asterisk from one cdb_ file and the vga_ext and
gx_vga_ext files to enable Generic VGA mode 12 video.
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_ext - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_ext_s - Serial mouse
*load -d CMDS/BOOTOBJS/MAUI/vga_ext
*load -d CMDS/BOOTOBJS/MAUI/gx_vga_ext
*
* MAUI port - CL-GD5434 ( up to 1024x768x24bpp )
Remove the leading asterisk from one cdb_ file and the gfx and gx_cl543
files to enable graphics support for the Cirrus Logic 5434

*load -d CMDS/BOOTOBJS/MAUI/cdb - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_s - Serial mouse
*load -d CMDS/BOOTOBJS/MAUI/gfx
*load -d CMDS/BOOTOBJS/MAUI/gx_cl543
*
*
*******************************************************
* VESA driver - uses INT 10h calls
Remove the leading asterisk from the following files to enable VESA driver
support

* the CDB determines which drivers are used. Pick one
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_vesa - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_vesa_s - Serial mouse
*
* The graphics descriptor.
*load -d CMDS/BOOTOBJS/MAUI/vesa - Must uncomment this line to use the
VESA driver
*
* The graphics driver. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/gx_vesa - Normal VESA driver
*load -d CMDS/BOOTOBJS/MAUI/gx_vesal - Linear mode VESA
*load -d CMDS/BOOTOBJS/MAUI/gx_vesah - 15 bit color support
```

```
*load -d CMDS/BOOTOBJS/MAUI/gx_vesalh - Linear mode, 15 bit color
*******************************************************
* ISAbank driver. Banked mode driver uses a data module
* the CDB determines which drivers are used. Pick one
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_svga - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_svga_s - Serial mouse
*
* The graphics descriptor.
*load -d CMDS/BOOTOBJS/MAUI/svgab - uncomment to use the ISA bank driver
*
* The graphics driver. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank1 - 1024x768 default
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank6 - 640x480 default
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank8 - 800x600 default
*
* The data module. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/ibcl5422 - Cirrus Logic 5422 ISA
*load -d CMDS/BOOTOBJS/MAUI/ibcl5428 - Cirrus Logic 5428 VESA LB
*load -d CMDS/BOOTOBJS/MAUI/ibcl5429 - Cirrus Logic 5429 VESA LB
*load -d CMDS/BOOTOBJS/MAUI/ibct65548ts110cs - Toshbia 110CS laptop
*load -d CMDS/BOOTOBJS/MAUI/ibct65550ts205cds - Toshbia 205cds laptop
*load -d CMDS/BOOTOBJS/MAUI/ibct65550ts205cdsvga - Toshbia 205cds laptop
with VGA monitor
*load -d CMDS/BOOTOBJS/MAUI/ibtlet4000 - Tseng Labs ET4000 ISA
*******************************************************

* End of bootlist
```

The `gx_vesa` driver comes in four different modules.

`gx_vesa`	This has 640x480, 800x600, 1024x768, and 1280x1024 support at 256 colors. The VESA BIOS is asked what modes are supported. The BIOS should stop the driver from setting any modes that can't be displayed. This driver will use a linear display buffer if the VESA BIOS is version 2.0 or greater and tells the driver that linear buffers are supported.
`gx_vesal`	This only works on cards with BIOS's that support linear mode. It is about 3 times faster than gx_vesa on supported hardware.

gx_vesah	This adds support for 15 bit high color modes. It looks for the highest resolution high color mode. The color depth table is separate from the resolution table.
gx_vesalh	This is a linear buffer only with high color support.

The descriptor is vesa and there are two cdb modules. cdb_vesa and cdb_vesa_s. cdb_vesa is set up for a bus mouse and cdb_vesa_s is set up for a serial mouse.

The gx_isabank driver comes in three modules depending on what default resolution you want. gx_isabank1 has a default of 1024x768, gx_isabank6 has a default of 640x480 and gx_isabank8 has a default of 800x600 all at 256 colors. The gx_isabank driver uses a data module to tell it how to talk to different hardware. The data modules included are listed below:

- ibcl5422 cirrus logic 5422 ISA card
- ibcl5428 cirrus logic 5428 VESA LB card
- ibcl5429 cirrus logic 5429 VESA LB card
- ibct65548ts110cs Toshiba laptop
- 110cs ibct65550ts205cds Toshiba laptop
- 205cds ibct65550ts205cdsvga Toshiba laptop
- 205cds with external VGA monitor
- ibtlet4000 Tseng labs ET4000 ISA card

The descriptor is svgab and the cdb's are cdb_svgab and cdb_svga_s. cdb_svga is for a bus mouse and cdb_svga_s is for a serial mouse.

All modules are in the following directory:
MWOS/OS9000/80386/PORTS/<BOARD>/CMDS/BOOTOBJS/MAUI

Configuring the Display for the STPC Board

OS-9 for STPC supports 256 colors with 640 x 480, 800 x 600 or 1024 x 768 resolutions. You can set this resolution by editing the loadmaui script and finding the STPC sections at the end of the file. You can also uncomment the code block for the desired resolution by removing the leading asterisks. The unused resolution must be commented out.

The STPC evaluation board may be configured to use the NTSC television output in place of the normal SVGA output by removing the leading asterisks from the NTSC section of the loadmaui file. The unused SVGA resolutions must be commented out. The video output must also be changed in the BIOS CMOS settings.

The following code fragments, from the loadmaui file, configure the display for 640 x 480 resolution.

```
* STPC - Gloria Board
* The gd_???? is for the client version and
* the gd_????_co is for the consumer version
* of the cpu.
* MAUI port - STPC at 640x480 256 color
```

```
*
load -d CMDS/BOOTOBJS/MAUI/cdb_stpc
load -d CMDS/BOOTOBJS/MAUI/svga_stpc
load -d CMDS/BOOTOBJS/MAUI/gx_stpc6
load -d CMDS/BOOTOBJS/MAUI/gd_stpc
*load -d CMDS/BOOTOBJS/MAUI/gd_stpc_co
*
* MAUI port - STPC at 800x600 256 color
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_stpc
*load -d CMDS/BOOTOBJS/MAUI/svga_stpc
*load -d CMDS/BOOTOBJS/MAUI/gx_stpc8
*load -d CMDS/BOOTOBJS/MAUI/gd_stpc
*load -d CMDS/BOOTOBJS/MAUI/gd_stpc_co
*
* MAUI port - STPC at 1024x768 256 color
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_stpc
*load -d CMDS/BOOTOBJS/MAUI/svga_stpc
*load -d CMDS/BOOTOBJS/MAUI/gx_stpc1
*load -d CMDS/BOOTOBJS/MAUI/gd_stpc
*load -d CMDS/BOOTOBJS/MAUI/gd_stpc_co
*
* MAUI port - STPC at 640x480 256 color NTSC output
*
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_stpc
*load -d CMDS/BOOTOBJS/MAUI/svga_stpc
*load -d CMDS/BOOTOBJS/MAUI/gx_stpc6
*load -d CMDS/BOOTOBJS/MAUI/gd_stpcntsc
*load -d CMDS/BOOTOBJS/MAUI/gd_stpcntsc_co
*
* End of bootlist
```

In the above example, the file `cdb_stpc` is the control data block for the mouse. `Svga_stpc` is the descriptor file for the graphics display. `Gx_stpc8` is the 800 x 600 resolution driver, `gx_stpc6` is the 640 x 480 resolution driver and `gx_stpc1` is the 1024 x 768 resolution driver. `Gd_stpc` and `Gd_stpcntsc` are data modules used by the graphic driver.

Using Cross-Hosted Utilities

The following utilities may be executed on the Windows system to access an OS-9 formatted floppy (RBF). You may use the cross hosted utilities in much the same way you do from OS-9. Note: The Wizard uses the Cross Hosted Utilities when creating boot media.

Table 2-1. Cross-Hosted Utilities

Command	Executable	Description
`os9cmp /d0/a /d0/b`	`os9cmp.exe`	compare files
`os9deldir /d0/MYDIR`	`os9deldir.exe`	delete directory
`os9dump /d0 or os9dump/d0@`	`os9dump.exe`	dump files
`os9merge /d0/a >/d0/b`	`os9merge.exe`	merge files
`os9touch /d0/sam`	`os9touch.exe`	touch file
`os9format /d0`	`os9format.exe`	format media
`os9attr -epege /d0/module`	`os9attr.exe`	change attribute
`os9bootgen /d0 -i=iplfd -l=coreboot`	`os9bootgen.exe`	make bootable
`os9chown 1.0 /d0/file`	`os9chown.exe`	change owner
`os9copy myfile -w= /d0`	`os9copy.exe`	copy file
`os9dcheck /d0`	`os9dcheck.exe`	check disk
`os9del /d0/sam`	`os9del.exe`	delete file
`os9dir -e /d0`	`os9dir.exe`	show directory
`os9free /d0`	`os9free.exe`	show free space
`os9list /d0/sys/password`	`os9list.exe`	list file
`os9makdir /d0/SYS /d0/CMDS`	`os9makdir.exe`	make directory
`os9mv /d0/CAT/toy /d0/DOG/toy`	`os9mv.exe`	move file
`os9rename /d0/sam /d0/fred`	`os9rename.exe`	rename file

See *OS-9 Utilities* for more information on cross-hosted utilities.

PCI Configuration Information

By default the PCI system will search up to seven buses. On newer motherboards, PCI slot devices are not bus zero. The maximum bus number may be changed in the following directory:

`MWOS/OS9000/80386/PORTS/<BOARD>/systype.h`

The PCI library must be re-made as well in the following directory:
`MWOS/OS9000/80386/PORTS/<BOARD>/PCILIB`

Running `os9make` from this directory will re-create a new PCI library. You must also re-make any drivers that require the new changes.

```
MWOS/OS9000/80386/PORTS/<BOARD>/systype.h
#defineISA_IOBASE0x00000000/* ISA Base Address */
#definePCI_CNF_ADR0x00000CF8/* PCI Configuration Address */
```

```
#definePCI_DATA_ADR0x00000CFC/* PCI Data Address */
#definePCI_IO_BASEISA_IOBASE/* PCI I/O Base */
#definePCI_MEM_BASE0x00000000/* PCI Memory Base */
#defineMAX_PCI_BUS_NUMBER7/* Max PCI BUS Number */
```

PCI Library User Guide

The following functions are contained in the PCI library, `pcilib.l`.

`pcilib.l` is compiled as port-specific. For example, for the PC-AT port, this library is located in `'MWOS/OS9000/80386/<BOARD>/LIB/pcilib.l'`.

_pci_search_device()

`_pci_search_device()` provides a means to check whether PCI devices are available in the system. If the system supports PCI devices and at least one PCI device is found, `_pci_search_device()` will return SUCCESS; otherwise it returns NO_DEVICE.

Syntax

```
#include <pcicnfg.h>
error_code _pci_search_device(PCI_config_stat stat);
```

Attributes

State: System

Header Files

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
pci_config_stat stat;
if (_pci_search_device(&stat) == NO_DEVICE) {
printf("There is no PCI devices on this machine.");
return EXIT_FAILURE;
}
return EXIT_SUCCESS;
}
```

_pci_next_device()

`_pci_next_device()` will find the next PCI device starting at the current bus_number and device_number in the `PCI_config_stat` structure pointed at by the incoming parameter stat. If another PCI device is found, the status returned is SUCCESS, and the fields:

- `bus_number`
- `device_number`
- `function_number`
- `vendor_id`
- `device_id`
- `rev_class`

in the structure stat points to will reflect the proper values for the device found. If no PCI next device is found, then `_pci_next_device()` will return `NO_DEVICE`.

Syntax

```
#include <pcicnfg.h>
error_code _pci_next_device(PCI_config_stat stat);
```

Attributes

State: System

Header Files

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
pci_config_stat stat;
stat.bus_number = 0;
stat.device_number = 0;
if (_pci_next_device(&stat) == NO_DEVICE) {
printf("There are no more PCI devices on this machine.");
return EXIT_FAILURE;
} else {
printf("Next device at bus:%d device%d\n",
stat.bus_number, stat.device_number);
}
return EXIT_SUCCESS;
}
```

pci_get_config_data()

`pci_get_config_data()` provides a simple means to obtain the PCI standard information for a given PCI device.

Many PCI devices include additional information after the standard configuration block. To access it one must use `pci_read_configuration()`. For information on the information returned, refer to the `pci_config_reg` structure in `pcicnfg.h`.

Syntax

```
#include <pcicnfg.h>

error_code pci_get_config_data(u_int32 bus, u_int32 device, u_int32 func,
PCI_config_reg cnfg);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, device;
pci_config_reg cnfg;
PCI_config_reg cp = &cnfg;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
pci_get_config_data(bus, device, 0L, &cnfg);
printf("\n");
printf("BUS:DEV VID DID CLASS RV IL IP\n");
printf("--------------------------------\n");
printf("%03d:%02d %04x %04x %06x %02x %02x %02x ",
bus, device,
cp->vendor_id, cp->device_id,
(cp->rev_class>>8)&0xffffff, cp->rev_class & 0xff,
cp->irq_line, cp->irq_pin );
return EXIT_SUCCESS;
}
```

pci_find_device()

`pci_find_device function()` will search the PCI bus for a device with the same `vendor_id` and `device_id` passed. If the index is nonzero, then the device found is based on the index. For example, if index is equal to one, then the second card found with the same `vendor_id` and `device_id` on a match is returned.

If a PCI device is found then `pci_find_device()` will return SUCCESS and the bus number and device number will be stored where the bus and dev arguments point respectively. The upper three bits of the device number specify the function number for multi-function devices.

If no PCI device is found, the `pci_find_device()` function will return NO_DEVICE.

Syntax

```
#include <pcicnfg.h>
error_code pci_find_device(u_int32 vender_id,
u_int32 device_id, u_int32 index,
u_int8 *bus, u_int8 *dev);
```

Attributes

State:System

Header Files

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>


main()
{
u_int8 bus, dev;
u_int32 index = 0;

if (pci_find_device(PCI_VENDOR_ID_NCR,
PCI_DEVICE_NCR53C810, index, &bus, &dev) == SUCCESS)
{
printf("NCR53C810 found at bus:%d device:%d function:%d\n",
bus, dev & 0x1f, dev >> 5);
}
return EXIT_SUCCESS;
}
```

pci_find_class_code()

The `pci_find_class_code()` function will search the PCI bus for a device with the same `class_code` as the one passed. If the index is nonzero, then the device found is based on the index. For example, if index is equal to one then the second card found with the same `class_code` on a match is returned.

If such a PCI device is found, then `pci_find_class_code()` will return SUCCESS and store the bus number and device number in the objects pointed at by the bus and dev parameters respectively. The upper three bits of the device number specify the function number for multi-function devices.

If no PCI device is found, `pci_find_device()` will return `NO_DEVICE`.

Syntax

```
#include <pcicnfg.h>
error_code pci_find_class_code( u_int32 class_code,
u_int32 device_index, u_int8 *bus, u_int8 *dev);
```

Attributes

State: System

Header Files

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>
#define NETWORK_ATM_CONTROLLER 0x020300
main()
{
u_int8 bus, dev;
u_int32 index = 0;
if (pci_find_class_code(NETWORK_ATM_CONTROLLER,
index, &bus, &dev) == SUCCESS)
{
printf("device at bus:%02d dev:%02d func:%02d\n",
bus, dev&0x1f, dev>>5);
}
return EXIT_SUCCESS;
}
```

pci_read_configuration_byte()

`pci_read_configuration_byte()` will return the PCI configuration byte value for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
u_int8 pci_read_configuration_byte(u_int32 bus, u_int32 dev,
u_int32 func, u_int32 index);
```

Attributes

State: System

Header Files

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int8 irqline;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
irqline = pci_read_configuration_byte(bus, device,
func, offsetof(pci_config_reg, irq_line));
printf("PCI irq line = %d\n", irqline);
return EXIT_SUCCESS;
}
```

pci_read_configuration_word()

`pci_read_configuration_word()` will return the PCI configuration word value for the PCI device at `bus` bus number, `dev` device number, `func` function number, `index` offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
u_int16 pci_read_configuration_word(u_int32 bus, u_int32 dev,
u_int32 func, u_int32 index);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int16 vend_id;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
vend_id = pci_read_configuration_word(bus, device,
func, offsetof(pci_config_reg, vendor_id));
printf("PCI vendor id = 0x%04x\n", vendor_id);
return EXIT_SUCCESS;
}
```

pci_read_configuration_dword()

`pci_read_configuration_dword()` function will return the PCI configuration dword value for the PCI device at `bus` bus number, `dev` device number, `func` function number, `index` offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
u_int32 pci_read_configuration_dword(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <systype.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int32 hdware;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
/* Get PCI I/O Port Address */
hdware = pci_read_configuration_dword(bus, dev, 0,
offsetof(pci_config_reg,base_addrs[0]));
/* mask address and add PCI Area Offset */
hdware = (hdware & ~1) + PCI_IO_BASE;
printf("PCI device at 0x%08x\n", hdware);
return EXIT_SUCCESS;
}
```

pci_write_configuration_byte()

`pci_write_configuration_byte()` writes to the PCI configuration space the byte value `val` for the PCI device at `bus` bus number, `dev` device number, `func` function number, `index` offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
error_code pci_write_configuration_byte(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int8 val);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
u_int8 bus, dev, func;
u_int8 cache_siz;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
cache_siz = 4; /* cache line size */
error = pci_write_configuration_byte(bus, dev, func,
offsetof(pci_config_reg, cache_line_siz), cache_siz);
return error;
}
```

pci_write_configuration_word()

`pci_write_configuration_word function()` writes to the PCI configuration space the word value `val` for the PCI device at `bus` bus number, `dev` device number, `func` function number, `index` offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
error_code pci_write_configuration_word(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int16 val);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
u_int8 bus, dev, func;
u_int16 cmd;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
cmd = 7; /* set device to allow bus master */
error = pci_write_configuration_word(bus, dev, func,
offsetof(pci_config_reg, command_reg), cmd);
return error;
}
```

pci_write_configuration_dword()

`pci_write_configuration_dword()` writes to the PCI configuration space the dword value `val` for the PCI device at `bus` bus number, `dev` device number, `func` function number, `index` offset into the configuration space.

Syntax

```
#include <pcicnfg.h>
error_code pci_write_configuration_dword(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int32 val);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
u_int8 bus, dev, func;
u_int32 value;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
value = 0xffffffff; /* get size info from device */
error = pci_write_configuration_dword(bus, dev, func,
offsetof(pci_config_reg, base_addrs[0]), value);
return error;
}
```

pci_get_irq_pin()

`pci_get_irq_pin()` returns the status of the IRQ pin on a given PCI device at `bus` bus number, `dev` device number, `func` function number.

Syntax

```
#include <pcicnfg.h>
u_int8 pci_get_irq_pin(u_int8 bus, u_int8 dev, u_int8 func);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int8 irqpin;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
irqpin = pci_get_irq_pin(bus, device, func);
printf("IRQ PIN = %d\n", irqpin);
return EXIT_SUCCESS;
}
```

pci_get_irq_line()

`pci_get_irq_line()` returns the status of the IRQ line on a given PCI device at `bus` bus number, `dev` device number, `func` function number.

Syntax

```
#include <pcicnfg.h>
u_int8 pci_get_irq_line(u_int8 bus, u_int8 dev, u_int8 func);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>



main()
{
u_int8 bus, dev, func;
u_int8 irqline;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
irqline = pci_get_irq_line(bus, device, func);
printf("IRQ LINE = %d\n", irqline);
return EXIT_SUCCESS;
}
```

pci_set_irq_line()

`pci_set_irq_line()` sets the IRQ line on a given PCI device at `bus` bus number, `dev` device number, `func` function number.

Syntax

```
#include <pcicnfg.h>
error_code pci_set_irq_line(u_int8 bus, u_int8 dev,
u_int8 func, u_int8 irqvect);
```

Attributes

State: System

Header File

`MWOS/SRC/DEFS/HW/pcicnfg.h`

Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int8 irqline;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
irqline = 9; /* IRQ LINE = vector 9 */
pci_set_irq_line(bus, device, func, irqline);
return EXIT_SUCCESS;
}
```

A Board-Specific Modules

This chapter describes the modules specifically written for the target board.

Low-Level System Modules

The following low-level system modules are tailored specifically for x86 platforms. The functionality of many of these modules can be altered through changes to the configuration data module (`cnfgdata`). These modules are located in the following directory:

`MWOS/OS9000/80386/PORTS/<BOARD>/CMDS/BOOTOBJS/ROM`

`llfa311`	Ethernet driver that supports autosense of 10/100BaseT Full/Half Duplex
`lle509`	Ethernet driver that supports the low-level 3Com ISA bus driver
`llpro100`	Ethernet driver for the Intel PRO/100 series card
`ll79C961`	Ethernet driver for the LAN79C961/AM79C973 cards
`ll8139`	Ethernet driver for the RealTek RL8139 card
`llne2000`	Ethernet driver for the NE2000 card
`llcis`	P2module that includes PCMCIA socket services
`ll21040`	Ethernet driver for the NETGEAR FA310-TX card
`ll91C94`	Ethernet driver for the SMC 91C94/96 cards
`lle509`	Ethernet driver for 3Com PCI card
`cnfgdata`	data module containing the configuration parameters * STPC only
`romcore`	system initialization * STPC only

High-Level System Modules

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

`MWOS/OS9000/80386/PORTS/<BOARD>/CMDS/BOOTOBJS`

MAUI Support

Modules in the PCAT Port Directory

`MWOS/OS9000/80386/PORTS/<BOARD>/CMDS/BOOTOBJS/MAUI`

`cdb`	`cdb_s`	`mp_xtkbd`	`ts205cdsvga`
`cdb_vesa`	`cdb_vesa_s`	`ibcl5422`	`ibtlet4000`
`cdb_vga`	`cdb_vga_s`	`ibcl5428`	`mp_kybrd`
`cdb_svga`	`cdb_svga_s`	`ibcl5429`	`mp_bsptr`
`gx_vga`	`gx_vesa`	`ibct65548`	`ibct65550`
`gx_vesah`	`gx_vesal`	`ts110cs`	
`gx_vesalh`	`vga_ext`	`ibct65550`	
`vga_ext_s`	`mp_msptr`	`ts205cds`	

Modules in the MEDIAGX Port Directory

MWOS/OS9000-80386/PORTS/MEDIAGX/CMDS/BOOTOBJS/MAUI

gx_mediagx

gx_mediagxh

Modules in the STPC Port Directory

MWOS/OS9000-80386/PORTS/STPC/CMDS/BOOTOBJS/MAUI

cdb_stpc	gd_stpc
gd_stpc_co	gd_stpcntsc
gd_stpcntsc_co	gx_stpc1
gx_stpc6	gx_stpc8
svga_stpc	

PersonalJava™ Support

pjruntime

pjruntime_g

Sequential Device Support

sc8042

serial mouse

p2mouse

sc16550

Parallel Driver

scp87303

aha1540

aic7870

Ticker

tk8253

Abort Handler

abort

Parallel Support

scsi8xx

Common System Modules List

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

`MWOS/OS9000/80386/PORTS/<BOARD>/CMDS/BOOTOBJS/ROM`

Table A-1. Typical Coreboot Image Contents

Module	Description
`cnfgdata`	data module containing the configuration parameters
`cnfgfunc`	retrieves configuration parameters from the cnfgdata boot data module
`commcnfg`	retrieves the name of the low-level driver to use for the auxiliary communications port from the configuration module
`conscnfg`	retrieves name of the low-level driver to use for the console from the configuration data module
`initext`	provides modular functional extension to the `sysinit1()` and `sysinit2()` routines
`io16550`	serial IO driver
`io8042`	provides support for ROM P2 modules
`ll1540`	SCSI driver for AHA1540
`ll7870`	SCSI driver for AIC-7870
`ll83790`	Ethernet driver for ROM P2 modules
`ncr8xx`	SCSI driver for NCR53C810/825
`portmenu`	boot system support module
`romcore`	system initialization
`rpciv`	ROM utility
`swi8timr`	software timer
`usedebug`	retrieves the flag from the configuration data module indicating whether or not the debugger is called during system startup

B Configuring Hardware Devices

This appendix contains detailed information for configuring and troubleshooting specific devices with OS-9.

Ethernet Controllers

Some Network Interface Cards require that a setup disk, included with the card, is ran before the card is installed in a system running OS-9.

The setup disk is required for configuring the connection type for cards which support multiple interfaces, such as connections for 10Base-T, 10Base-2 or AUI. The setup disk may also be needed to configure the card for a specific interrupt or I/O address.

3Com PCI

3C900B-TPO - 10Base-T TPO NIC

3C900B-CMB - 10Base-T/10Base-2/AUI Combo

3C905-T4 - 10/100 Base-T4 (RJ-45) - 3C905-T4 Fast Etherlink XL

3C905B-TX - 10/100Base-TX NIC

3CSOHO100-TX - 10/100 Base-TX NIC - Office Connect 10/100

3C900-TPO - 10Base-T TPO NIC

System-State Debugging - Supported

Default Settings

PORTADDR	NA
IRQVECTOR	NA
CONNTYPE	INF_EXT /* Auto */

Solving Configuration Issues

Connection Type

The default connection type is set to INF_EXT (auto). For the 3Com ISA card, this implies the card setup program has been used and has setup the card connection type. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following:

1. Use the 3Com setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.
3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

```
(Super)[/h0/sys/>] netstat -in
NameMtu   NetworkAddressIpktsIerrsOpktsOerrsColl
lo0 1536  <Link> 0      0000
lo0 1536  127    127.0.0.100000
enet01500 <Link> 00.00.C0.91.4F.96551103500
enet01500 182.52.109182.52.109.2555103500
```

Modifying the OS-9 Descriptor

Edit the file MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf_desc.h, looking in the "#ifdef spe30_pci" section for CONNTYPE, which you should set to the appropriate value from the following list:

INF_AUI = AUI Connection type

INF_BNC = BNC connection type

INF_UPT = 10BaseT (RJ45)

INF_EXT = Use same connection type determined in 3Com setup program

```
/*

 * From spf_desc.h
*/

/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */

#define CONNTYPEINF_EXT
```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509 directory and type os9make -f=spfdesc.mak.

Next time you run the Wizard, it will use the new descriptor.

Boomerang

The source code for the spe509 driver includes “#if defined(BOOMERANG)” sections to allow only including support for the newer 3Com PCI based cards. Each card is now defined in a constant table and as such the driver makefile used must be modified to include both the define for “BOOMERANG” and the compiler option “-c” to force constant code data.

```
/* spfdrvr.mak - add the following define and compiler option */

DEFINES = -c -dBOOMERANG

/* spfdesc.mak - add the following define */
MACROS = -dBOOMERANG
```

DMA

To allow support for the newer 3Com “B” based cards, DMA support with ring buffers has been added. The size of the ring buffers may be set in the “spf_desc.h” file.

```
#define RX_RING_CNT32/* Number of buffers in BOOMERANG recv ring */
#define TX_RING_CNT16/* Number of buffers in BOOMERANG xmit ring */
```

Time-Out Options

To allow support with switches and slow hubs the time-out for checking for link beat has been increased. This change effects 3Com NON-B parts as well as PCMCIA CARDS using UTP connections. The default time-out prior to this change was 750ms. Most switches take two to three seconds to sync. A loop count has been added.

```
/* When a connection type is tried will wait for the time
 * specified in LINK_BEAT_ITER and LINK_BEAT_SLEEP_TIME.
 * This should address the problem with not being able to work
 * with switches. Most switches take 2 to 3 seconds, we will wait up to
 * 5.25 seconds (192/256ths)*7. */


#define LINK_BEAT_ITER 7
#define LINK_BEAT_SLEEP_TIME 0x800000c0 /* 192/256ths of a second (750 ms)
*/
```

The `PCIV` utility may be used to examine a network card. This utility displays vendor and device ID's for each installed PCI device. To find out if your card has been tested with OS-9, run the `pciv` command and look at the vendor and device IDs. The vendor ID should be 0x10B7 for all 3Com network cards. Network cards with the following device IDs have been tested with OS-9 drivers:

3Com 3C5090x5900
3Com 3C900-TPO0x9000
3Com 3C9000x9001
3Com 3C900B-TPO0x9004
3Com 3C900B-CMB0x9005
3Com 3C905-T40x9051(2)
3Com 3C905B-TX0x9055(1)
3Com 3CSOHO100-TX0x7646(1)

Support for the following cards is included with the driverp; however, these cards were not tested prior to the release.

3Com 3C905-TX0x9050
3Com 10/100 COMBO Deluxe0x9058
3Com 10Base-T/10Base-2/TPC0x9006
3Com 10Base-FL NIC0x900A
3Com 100Base-FX NIC0x905A
3Com Tornado NIC0x9200
3Com 10/100 Base-TX NIC (Python-H)0x9800
3Com 10/100 Base-TX NIC (Python-T)0x9805

- 100BaseT support is included for 3C905B-TX and 3CSOHO100-TX.
- The 3C905-T4 has been tested with 10BaseT only.

3Com ISA

3Com ISA EtherLink III

System-State Debugging

Supported

Default Settings

PORTADDR	0x340/* IO port for ISA */
IRQVECTOR	0x43/* IRQ vector */
CONNTYPE	INF_EXT/* Auto */

Solving Configuration Issues

Connection Type

The default connection type is set to INF_EXT (auto). For the 3Com ISA card, this implies the card setup program has been used and has setup the card connection type. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following.

1. Use the 3Com setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.
3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ3. In this case you have the following options.

1. Disable the COM2 serial port from the BIOS to allow IRQ3 to function with this card.
2. Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

If it seems like you should be getting interrupts this can be tested.

Use the command irqs to see a list of interrupts, e.g.:

```
(Super)[/h0/sys/>] irqs
```

```
PC-AT Compatible 80386  OS9 For Embedded Systems


 vector  ($)   prior  drivstat   irq svc    driver  dev list
-------------  -----  ---------  ---------  ------  ---------
     7  ($07)     10  $0003c444  $0010f7b4  fpu      <na>
    14  ($0e)      1  $0003c3a4  $00110113  vectors  <na>
    64  ($40)     10  $00ff40b0  $0011098f  tk8253   <na>
    65  ($41)     10  $00ffa680  $00120582  sc8042m  <na>
    65  ($41)     10  $00e85db0  $00120582  sc8042m  <na>
    65  ($41)     10  $00e84a40  $00120582  sc8042m  <na>
    65  ($41)     10  $00e82980  $00120582  sc8042m  <na>
    74  ($4a)      1  $00ff02d0  $001f9504  spe509   <na>
    78  ($4e)     10  $00ff4f30  $00137906  rb1003   <na>
```

In this case, you can go into RomBug by typing break and placing a breakpoint at the ISR.

```
$ break
RomBug: b 1f9504
RomBug: g
```

and then pinging a machine on the net:

```
$  ping 182.52.109.13
```

(using the actual address of another machine on the network, rather than the one shown above).

If interrupts are running you should be presented a Rombug prompt at the breakpoint address. You can type g to see if you get another interrupt or k to kill the breakpoint.

Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case, the OS-9 command netstat -in will not show the card as available.

The following netstat example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in


Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs
Coll
lo0   1536  <Link>                             0     0        0     0     0
lo0   1536  127           127.0.0.1            0     0        0     0     0
enet0 1500  <Link>      00.00.C0.91.4F.96      55   110       35     0     0
enet0 1500  182.52.109    182.52.109.25        55   110       35     0     0
```

Modifying the OS-9 Descriptor

Edit the file MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf_desc.h and look for the #ifdef spe30_isa section.

Change the fields below as required.

INF_AUI = AUI Connection type

INF_BNC = BNC connection type

INF_UTP = 10BaseT (RJ45)

INF_EXT = Use same connection type determined in 3Com setup program

```
/*

 * From spf_desc.h

*/

#define PORTADDR0x340/* IO port for ISA*/
#define IRQVECTOR0x43/* IRQ vector                          */
/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */
#define CONNTYPEINF_EXT
```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509 directory and type:

```
C:> os9make -f=spfdesc.mak
```

Next time you run the Wizard the new descriptor will be used.

BNC is not supported for the 3Com 10/100 Combo cards.

Low-level System Changes

If system-state debugging is used, you must change the low-level system by modifying the following lines from the file

MWOS/OS9000/80386/PORTS/<BOARD>/ROM/CNFGDATA/config.des:

```
#define LLE509_PORT_ADDRESS 0x340
#define LLE509_IF_VECTOR 0x43
```

as required by the system. For example, for IRQ10, here are the changes required.

```
#define LLE509_PORT_ADDRESS 0x340
#define LLE509_IF_VECTOR 0x4a
```

The Wizard will automatically re-make the cnfgdata module.

3Com PCMCIA

When making bootfile only images care should be taken to make sure PCMCIA support is enabled in the low-level 'coreboot' system if PCMCIA devices are to be employed once the system is booted.

3Com EtherLink III PC CARD

3Com Megahertz LAN (3CCE589ET) - 10 Mbps LAN PC Card

System-State Debugging

Supported

Default Settings

PORTADDR	0x340/* IO port for ISA */
IRQVECTOR	0x43/* IRQ vector */
CONNTYPE	INF_EXT/* Auto */

Solving Configuration Issues

Connection Type

The default connection type is set to INF_EXT (auto). For the 3Com PCMCIA card this implies the card will detect the connection type used. If desired the connection type may be forced. To force the connection type the descriptor must be changed.

Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ3. In this case you have the following options.

1. Disable the COM2 serial port from the BIOS to allow IRQ3 to function with this card.
2. Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed. Also the PCMCIA socket services setup must be changed to assign the new interrupt to the PCMCIA Ethernet Card.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

Use the command irqs to see a list of interrupts.

```
(Super)[/h0/sys/>] irqs
```

```
 vector  ($)   prior  drivstat   irq svc    driver  dev list
-------------  -----  ---------  ---------  ------  ---------
     7  ($07)     10  $0003c444  $0010f7b4  fpu      <na>
    14  ($0e)      1  $0003c3a4  $00110113  vectors  <na>
    64  ($40)     10  $00ff40b0  $0011098f  tk8253   <na>
    65  ($41)     10  $00ffa680  $00120582  sc8042m  <na>
    65  ($41)     10  $00e85db0  $00120582  sc8042m  <na>
    65  ($41)     10  $00e84a40  $00120582  sc8042m  <na>
    65  ($41)     10  $00e82980  $00120582  sc8042m  <na>
    74  ($4a)      1  $00ff02d0  $001f9504  spe509   <na>
    78  ($4e)     10  $00ff4f30  $00137906  rb1003   <na>
```

In the case above, you can go into RomBug by typing break and placing a break at the ISR.

```
$ break


RomBug: b 1f9504
RomBug: g
```

and then pinging a machine on the net.

```
$  ping 182.52.109.13
```

(Using the actual address of another machine o the network, rather than the one shown above.)

If interrupts are running you should be presented a Rombug prompt at the breakpoint address. You can type g to see if you get another interrupt or k to kill the breakpoint.

Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case the OS-9 command netstat -in will not show the card as available.

```
(Super)[/h0/sys/>] netstat -in


Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs
Coll
lo0   1536  <Link>                             0     0        0     0     0
lo0   1536  127           127.0.0.1            0     0        0     0     0
enet0 1500  <Link>      00.00.C0.91.4F.96      55   110       35     0     0
enet0 1500  182.52.109    182.52.109.25        55   110       35     0     0
```

Modifying the OS-9 Descriptor

Edit the file `MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf_desc.h`

Look for the `#ifdef spe30_isa` section and change the PORTADDR, IRQVECTOR, and CONNTYPE as required.

The permissible values for CONNTYPE are:

INF_AUI = AUI Connection type

INF_BNC = BNC connection type

INF_UPT = 10BaseT (RJ45)

INF_EXT = Probe connection type

```
/*
 * From spf_desc.h
*/


#define PORTADDR           0x340/* IO port for ISA*/
#define IRQVECTOR          0x43/* IRQ vector                        */
/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */
#define CONNTYPE           INF_EXT
```

Finally, remake the descriptor by changing to the `MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509` directory and typing

```
os9make -f=spfdesc.mak.
```

Low-level System Changes

System-State debugging requires a change to the low-level system, as well as the PCMCIA socket services information. This is controlled by the contents of the file `MWOS/OS9000/80386/PORTS/<BOARD>/ROM/cnfgdata.des`.

Find the following lines:

```
#define LLE509_PORT_ADDRESS0x340
#define LLE509_IF_VECTOR   0x43
#define ETH_CIS_PARAMS     "3com=0x340,3"
```

The above port addresses and/or IRQ information should be changed as required by the system. For IRQ 10, here are the changes required:

```
#define LLE509_PORT_ADDRESS0x340
#define LLE509_IF_VECTOR   0x4a
#define ETH_CIS_PARAMS     "3com=0x340,10"
```

The Wizard will automatically re-make the cnfgdata module.

DEC 21140

Intra Server DE504-BA (Quad)

Asante' Fast 10/100

D-Link DFE-500TX ProFast 10/100 Adapter

System-State Debugging

Supported

Default Settings

```
PORTADDR                    NA
IRQVECTOR                   NA
CONNTYPE                    INF_UTP
```

Solving Configuration Issues

Connection Type

The default connection type is set to INF_AUI.

The following netstat example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in

Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs
Coll
lo0   1536  <Link>                            0     0        0     0     0
lo0   1536  127           127.0.0.1           0     0        0     0     0
enet0 1500  <Link>      00.00.C0.91.4F.96     55   110       35     0     0
enet0 1500  182.52.109    182.52.109.25       55   110       35     0     0
```

Modifying the OS-9 Descriptor

Edit the file `MWOS/OS9000/80386/PORTS/PCAT/SPF/SP21140/DEFS/spf_desc.h`, changing the definition of CONNTYPE as required.

Possible values for CONNTYPE:

```
*

 * From spf_desc.h

*/

/*

 * Interface/connection type

 * Common values:

 *
 * INF_UTP      ==    MII_10MB == 10Mb/s 21140
 * INF_AUI      ==    SRL_10MB == Conventional 10Mb/s 21140
```

```
 * INF_UTP100   ==    MII_100MBTX == MII 100Mb/s 21140
 * INF_FX100    ==    MII_100MBFX == MII 100Mb/s 21140
 * INF_MII10    ==    MII_10MB == 10Mb/s 21140
 * INF_MII100   ==    MII_100MB == MII 100Mb/s 21140
 *


 * Note: Not all common values will work. Below are common
 * values used for different cards supported. Much work at
 * driver level still remains to allow auto and NWay support.
 * Support for DEC21143 may be added in the future once the
 * NWay support is added.
 *
 *  Intra Server DE504-BA (Quad)
 *     10BaseT = INF_UTP  ( note: preliminary release support for 21143.
No 100BaseT support)
 *
 * Asante' Fast 10/100
 *     10BaseT = INF_UTP
 * 100BaseT = INF_MII100
 *
 *  D-Link DFE-500TX ProFast 10/100 Adapter
 *
 *    10BaseT = INF_UTP
 *   10BaseT = INF_MII10
 * 100BaseT = INF_MII100
 *
*/


#define CONNTYPE    INF_UTP
```

Finally, remake the descriptor: change to the `MWOS/OS9000/80386/PORTS/PCAT/SPF/SP21140` directory and type:

```
os9make -f=spfdesc.mak
```

You have now created a new descriptor. Next time you run the Wizard, it will use the new descriptor.

Adding support for Dual and Quad Channel Cards

The descriptors for the additional Ethernet ports must be added. Edit the spf.ml file in the `MWOS/OS9000/80386/PORTS/<BOARD>/BOOTS/INSTALL/PORTBOOT` directory. Find the entry for `spde0`. Add `spde1` for a dual card or `spde1`, `spde2` and `spde3` for a quad card.

Next edit the <board>.ini file located in `MWOS/OS9000/80386/PORTS/<BOARD>/BOOTS/INSTALL/INI` directory. Look for the ETHER_OPTION_ string and add the entries as required. You must specify the Ethernet information for all extra Ethernet ports used.

The following example adds the three extra Ethernet ports for a quad card.

ETHER_OPTION_2=enet1 address 112.16.1.237 broadcast 112.16.255.255 netmask 255.255.000.000 binding /spde1/enet

ETHER_OPTION_3=enet2 address 122.16.1.237 broadcast 122.16.255.255 netmask 255.255.000.000 binding /spde2/enet

ETHER_OPTION_4=enet3 address 132.16.1.237 broadcast 132.16.255.255 netmask 255.255.000.000 binding /spde3/enet

Once the boot image is created you may boot OS-9 and use `netstat` to see that all cards are active and ready for use. You should see entries for `enet0`, `enet1`, `enet2` and `enet3` if you are using a quad card.

AM79C961 & AM79C73A

System-State Debugging

Supported

Default Settings

PORTADDR	0x300
IRQVECTOR	NA
CONNTYPE	NA

Solving Configuration Issues

The AM79C961A driver is designed to work in systems where DMA BUS MASTER mode is employed with respect to the AM79C961 or AM79C973 interfaces.

The AM79C961A driver is PLUG & PLAY. Only the base address should be defined to allow multiple card usage.

Modifying the OS-9 Descriptor

1. Edit the file. `MWOS/OS9000/80386/PORTS/PCAT/SPF/SP79C961/DEFS/spf_desc.h`, changing the line defining PORTADDR, which reads `#define PORTADDR 0x300 /* Base address of hardware */`, to give PORTADDR the desired value.
2. Next re-make the descriptor: change to the `MWOS/OS9000/80386/PORTS/PCAT/SPF/SP79C961` directory and type the command `os9make -f=spfdesc.mak`

You have now created a new descriptor. The next time you run the Wizard, it will use the new descriptor.

NE2000

ZF NetDisplay

ACCTON - EN166X MPX 2 Ethernet

D-LINK DE-220PCT - 10Mbps Combo 16-Bit Ethernet ISA Adapter

Compex - ReadyLink 2000 - PCI 32-bit

System-State Debugging

Supported

Default Settings

PORTADDR	0x340/* IO port for ISA */
IRQVECTOR	0x49/* IRQ vector */
CONNTYPE	INF_EXT/* Auto */

Board Setup Issues

ZF NetDisplay

use <CDROM>:\Drivers\Ethernet\Realtek\RSET8019.EXE"

to determine the IO address and IRQ required.

IO=0x340 VECTOR=0x49 is typical. Settings are system dependent.

ACCTON - EN166X MPX 2 Ethernet

use "1step" program located on the setup disk to set card to

jumpered "ne2000" mode.

IO=0x300 VECTOR=0x43 is typical. Settings are system dependent.

D-LINK DE-220PCT - 10Mbps Combo 16-Bit Ethernet ISA Adapter

Use "setup" program located on the setup disk "A:\SETUP\setup.exe" to setup the card. Disable PNP and setup Interrupt and I/O base address.

Compex - ReadyLink 2000 - PCI 32-bit

Just plug and go. Multiple cards may be used by using the

PCI Specific Settings listed below.

PCI Settings

When using multiple NE2000 PCI cards in a system you may force the driver to use a specific slot or card number for the device being used. PCIINDEX may be used to specify the card instance to be used. Keep in mind the PCIINDEX method is based on a first found basis, so moving cards in the system will change the configuration used.

You may also use the PCIBUS and PCIDEV to force the use of the device to a specific slot. To find out the current PCIBUS and PCIDEV values use the OS-9 command pciv.

```
/*
 * PCI Specific Settings
*/
```

```
#define PCIINDEX        0x00/* 0 picks first card */
#define PCIBUS          0x00/* 0 indicates to search */
#define PCIDEV          0x00/* 0 indicates to search */
```

Connection Type

The default connection type is set by either the configuration setup program that came with the card or by hardware jumpers employed. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following.

1. Use the NE2000 setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.
3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

Interrupt Conflict Options

Another problem may be the interrupt used. The default interrupt is IRQ9. In this case you have the following options.

1.Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

Use the command irqs to see a list of interrupts.

```
(Super)[/h0/sys/>] irqs
```

```
 vector  ($)   prior  drivstat   irq svc    driver  dev list
------------  -----  ---------  ---------  ------  ---------
     7  ($07)     10  $0003c444  $0010f7b4  fpu      <na>
    14  ($0e)      1  $0003c3a4  $00110113  vectors  <na>
    64  ($40)     10  $00ff40b0  $0011098f  tk8253   <na>
    65  ($41)     10  $00ffa680  $00120582  sc8042m  <na>
    65  ($41)     10  $00e85db0  $00120582  sc8042m  <na>
    65  ($41)     10  $00e84a40  $00120582  sc8042m  <na>
    65  ($41)     10  $00e82980  $00120582  sc8042m  <na>
    74  ($49)      1  $00ff02d0  $001f9504  spne2000 <na>
    78  ($4e)     10  $00ff4f30  $00137906  rb1003   <na>
```

In the case above, you can go into RomBug by typing break and placing a breakpoint at the ISR.

```
$ break
RomBug: b 1f9504
RomBug: g
```

and then pinging a machine on the net:

```
$  ping 182.52.109.13
```

(using the actual address of another machine on the network, rather than the one shown above).

If interrupts are running you should be presented a Rombug prompt at the breakpoint address. You can type g to see if you get another interrupt or k to kill the breakpoint.

Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case, the OS-9 command netstat -in will not show the card as available.

The following netstat example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in

Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs
Coll
lo0   1536  <Link>                             0     0        0     0     0
lo0   1536  127           127.0.0.1            0     0        0     0     0
enet0 1500  <Link>      00.00.C0.91.4F.96      55   110       35     0     0
enet0 1500  182.52.109    182.52.109.25        55   110       35     0     0
```

Modifying the OS-9 Descriptor

Edit the file `MWOS/OS9000/80386/PORTS/PCAT/SPF/NE2000/DEFS/spf_desc.h`.

Change the fields below as required.

```
/* From spf_desc.h */

#define PORTADDR        0x00000340/* Base address of hardware */
#define VECTOR          0x49/* Port vector */

/* PCI Specific Settings */

#define PCIINDEX        0x00/* 0 picks first card */
#define PCIBUS          0x00/* 0 indicates to search */
#define PCIDEV          0x00/* 0 indicates to search */
```

Finally, remake the descriptor: change to the `MWOS/OS9000/80386/PORTS/PCAT/SPF/NE2000` directory and type:

```
C:> os9make -f=spfdesc.mak
```

Next time you run the Wizard the new descriptor will be used.

Low-level System Changes

If system-state debugging is used, you must change the low-level system by modifying the following lines from the file

`MWOS/OS9000/80386/PORTS/<BOARD>/ROM/cnfgdata.des`:

```
#define LLNE2000_PORT_ADDRESS       0x340
#define LLNE2000_IF_VECTOR          0x49
```

as required by the system. For example, for IRQ10, here are the changes required.

```
#define LLNE2000_PORT_ADDRESS       0x340
#define LLNE2000_IF_VECTOR          0x4a
```

The Wizard will automatically re-make the cnfgdata module.

NE2000 PCMCIA

Realtek RTL-8029	0x10ec	0x8029
Winbond 89C940	0x1050	0x0940
Winbond w89C940	0x1050	0x5a5a
KTI ET32P2	0x8e2e	0x3000
NetVin NV5000SC	0x4a14	0x5000
Via 82C926	0x1106	0x0926
SureCom NE34	0x10bd	0x0e34
Holtek HT80232	0x12c3	0x0058
Holtek HT80229	0x12c3	0x5598

Cirrus Logic CS8900

The OS-9 `sp8900` software driver provides support for the Cirrus Logic CS8900a Ethernet Controller. This allows the device to be used as part of an OS-9 SoftStax network implementation.

The Cirrus Logic CS8900a provides single chip support for IEEE 802.3 Ethernet. It has a direct ISA bus interface and is therefore commonly found in PC-AT type environments.

The OS-9 `sp8900` driver takes advantage, where appropriate, of the Plug and Play capability of the CS8900a device. This reduces the time taken to configure the CS8900a for use within an OS-9 environment.

System-State Debugging

Not Supported

Hardware Configuration

The CS8900a should be supplied with an MSDOS hosted configuration program. This should be used to pre-configure the device for use. This program assumes the CS8900 a has the associated EEPROM as recommended. This EEPROM is used to store the CS8900 a configuration parameters. At this time OS-9 will only support devices that have this configuration.

Using the Setup Program

Before using the setup program, you should determine the network adaptor's IO address and Interrupt level. The cs8900a has a limited number of possible combinations, these should be chosen with care. As a default OS-9 will assume IO port 0x300 and IRQ Level 10. It is also important to note that OS-9 drives the device using the PC-AT I/O Bus for ALL operations. Therefore shared memory should be disabled for OS-9 operation.

Having selected the correct choices you may run the setup program and configure the CS8900a accordingly.

If the device was supplied without a configuration utility it will be necessary to obtain this from the vendor or try the cirrus logic Web site:
`http://www.cirrus.com/drivers/`

The setup program also incorporates a self test utility that may be used to confirm correct operation of the device before proceeding.

OS-9 Software Configuration

Configuring PnP Firmware

The OS-9 `sp8900` driver will use the PnP (Plug and Play) capability of the cs8900a. This will only be used if it is enabled in the OS-9 device descriptor. When enabled the OS-9 driver will search all possible I/O locations for a cs8900a device. If found, the first one, starting at the lowest valid I/O address, will be used. The software will confirm that the EEPROM is present. The OS-9 driver extracts the necessary configuration details from this device and initializes the cs8900a.

Configuring OS-9 Descriptors

The OS-9 device descriptor allows you to override the PnP default configuration. At this time only a subset of all the possible configuration parameters may be overridden. To change the PnP values the following fields must be modified. This should be performed using a text editor and the OS-9 tools provided within the Microware Hawk package.

Once modified the descriptor should be regenerated and tested.

Device Descriptor Fields

The standard device descriptor is as follows. This file may be found in

```
 .../MWOS/OS9000/<processor>/PORTS/<port>/SPF/SP8900/DEFS/spf_desc.h
#define SPF_DIR_NONE0xFF
#define SPF_DIR_IN0x00
#define SPF_DIR_OUT0x01

#include <SPF/item.h>

#ifdef spcs0

/***    Device Descriptor for SPF 8900 ethernet driver      */
#define PNPON   1 /* do plug and play */
#define PNPOFF  0 /* Use descriptor values ( see manual )    */

/*****************************************************************
**
*  User configuration defines
/*****************************************************************
**
*  Port configuration defines                                    */

/* Macros that initialize device descriptor common fields */
```

```
/* 300/320/340/360 */

#define PORTADDR          0x300/* Base address of hardware */
#define LUN               0x7F/* logical unit number      */

#define VECTOR            0x4a/* Port vector              */
#define PRIORITY          8/* IRQ polling priority     */
#define IRQLEVEL          0/* Port IRQ Level           */
#define PNP8900           PNPON/* Do plug and play ( Normal setting ) */

#define TB486COMPATTRUE
--------------------*/
```

Any information (not shown) beyond this point MUST not be changed.

User Configurable Fields

The following fields are user configurable.

Field Name	Default Value	Possible Values
PORTADDR		0x3000x200..0x360
VECTOR		0x4a0x45,0x4a,0x4b,0x4c
PRIORITY		0x080..255
PNP8900		PNPONPNPON or PNPOFF
TB486COMPAT*		TRUETRUE or FALSE

*note: The tb486 board is a special case and this flag should be set false for any other board type.

Generating a New Device Descriptor

Having located and edited the field as desired the new device descriptor may be generated with the following steps

Change directory to: ../MWOS/OS9000/80386/PORTS/PCAT/SPF/SP8900

Enter the command: os9make -f=sppfdesc.mak -u MOPTS=-u

The new descriptor will be built.

OS-9 SP8900 Components

The complete driver consists of two OS9 load modules. You should refer to the appropriate Microware manual for further information concerning system configuration. The SP8900 component files are:

SP8900 -- cs8900a Ethernet Driver

spcs0 -- cs8900 Device Descriptor

NETGEAR

FA311

FA312

The high-level driver for these cards is SPFA311. The low-level driver is LLFA311.

NETGEAR FA311 and FA312 support autosense of 10/100BaseT Full/Half Duplex and contain multicasting support.

LAN

LAN79C961

LANAM79C973

The low-level driver for this card is LL79C961. The high-level driver is SP79C961.

Multicasting support is available for the SP79C961.

Realtek

RL8139

The low-level driver for this card is LL8139. The high-level driver is SP8139.

SMC

SMC91C94

SMC91C96

The low-level driver for this card is LLC91C94. The high-level driver is SP91C94.

Intel PRO/100 Series

82557

82558

82559

The low-level driver for this card is LLPRO100. The high-level driver is SPPRO100.

Intel PRO/1000 Series

82540

82541

The LLPRO1000 low-level driver has not been ported to x86 boards but sources are provided. The high-level driver is SPPRO1000.

Sequential Device Support

VGA Graphics / Keyboard

VGA support is provided using standard VGA graphics screen and keyboard. Most PC based systems use VGA keyboard as the default device for user input. While this is not required for OS-9 based systems it is a convenient way to initially setup systems for use with OS-9.

During the development of MAUI user applications, a serial console may be the preferred method since the text based console may interfere with the graphics application on the same device.

MULTI-TERM is a feature of the VGA Graphics/Keyboard console driver which provides up to four virtual screens. If you are a console user, you may switch between screens by pressing an alternate function key combination, such as <Alt> <F1>, <Alt> <F2>, <Alt><F3> or <Alt><F4>. MULTI-TERM may be started automatically in the /h0/sys/startup file or manually from the console by executing the following commands:

```
$ mshell -l <>>>/mterm1&
$ mshell -l <>>>/mterm2&
$ mshell -l <>>>/mterm3&
```

VGA TERMINAL Descriptors Notes

/mterm0	Multi-term descriptor 0
/mterm1	Multi-term descriptor 1
/mterm2	Multi-term descriptor 2
/mterm3	Multi-term descriptor 3

The following optional settings apply to the VGA/Keyboard console:

#define DS_ROMBREAK	1/* Enter RomBug - Shift PrintScreen. */
#define DS_RESTART	1/* Reset System - Ctrl/Alt/Del. 0=disabled */
#define DS_NUM_LOCK	1/* Keyboard Number lock 0=off 1=on */
#define DS_SHIFT_LOCK	0/* Keyboard Caps lock 0=off 1=on */

To change these options, edit the file `MWOS/OS9000/80386/PORTS/PCAT/SCF/SC8042M/confg.des`. Find the sections as outlined above. Change as desired. Then, change to the `MWOS/OS9000/80386/PORTS/PCAT/SCF/SC8042M/DRVR` directory and type `os9make`.

Language Support Options

To change the language support for the keyboard use the advanced mode from the Wizard and select **BOOTFILE OPTIONS** tab. Select the language desired.

`MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\term0`

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm0
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm1
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm2
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm3
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\term0_fr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm0_fr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm1_fr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm2_fr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm3_fr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\term0_gr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm0_gr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm1_gr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm2_gr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm3_gr
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\term0_nw
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm0_nw
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm1_nw
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm2_nw
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm3_nw
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\term0_uk
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm0_uk
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm1_uk
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm2_uk
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC8042M\mterm3_uk
```

Serial Mouse

Configuration modules for a Serial Mouse is included in the system image when the Mouse option is not selected in the Configuration Wizard's Master Builder screen. Serial mouse support is only included when sc16550 support is enabled in the Configuration Wizards BOOTFILE OPTIONS dialog box.

The default port is COM1. The `MWOS/OS9000/80386/PORTS/<BOARD>/BOOTS/INSTALL/PORTBOOT/bootfile.ml` file may be changed to allow a different port to be used.

Default (Serial Mouse configured using COM1)

```
* [OPTION4 && !MOUSE] serial mouse
*
../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t1
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t2
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t3
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t4
```

Changed to use COM3

```
* [OPTION4 && !MOUSE] serial mouse
*
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t1
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t2
../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t3
*../../../CMDS/BOOTOBJS/DESC/SC16550/m0_t4
```

PS2 Mouse

PS2 mouse support is automatically included when the Mouse option is selected from the Configuration Wizard's Master Build screen.

16550 Serial

Standard PC type serial ports are supported. By default, four descriptors are available, but you may add more as needed.

Use of COM1 and COM2 are standard on PC based systems. COM3 and COM4 are not. Since COM1 and COM2 use IRQ3 and IRQ4, most systems will not allow COM3 and COM4 to also use IRQ3 and or IRQ4. The main reason for this is that IRQ3 and IRQ4 are normally edge based interrupts, and the 16550 is normally implemented in a edge based configuration. Therefore, anytime COM3 and or COM4 are used, you must determine the interrupt vector to use for these ports.

To change the vector you must edit the `systype.h` file located in the port directory.

```
MWOS/OS9000/80386/PORTS/<BOARD>/systype.h
#define T1BASE_165500x000003f8/* SC16550 port 1 */
#define T1VECT_165500x44/* IRQ 4 */
#define T1PRI_165505/* Priority */

#define T2BASE_165500x000002f8/* SC16550 port 2 */
#define T2VECT_165500x43/* IRQ 3 */
#define T2PRI_165505/* Priority */

#define T3BASE_165500x000003e8/* SC16550 port 3 */

#define T3VECT_165500x44/* IRQ 4 */
#define T3PRI_1655010/* Priority */

#define T4BASE_165500x000002e8/* SC16550 port 4 */
#define T4VECT_165500x43/* IRQ 3 */
#define T4PRI_1655010/* Priority */
```

Making the Descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory: `MWOS/OS9000/80386/PORTS/PCAT/SCF/SC16550/DESC`

Type `os9make`; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\term1
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\t1
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\term2
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\t2
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\term3
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\t3
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\term4
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\t4
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\ps
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\m0_t1
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\m0_t2
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\m0_t3
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SC16550\m0_t4
```

Digiboard

Support for the Digiboard intelligent serial card is included by selecting the Digiboard option in the Configuration Wizard's Bootfile Options dialog box.

To change the vector you must edit the `systype.h` file located in the port directory.

`MWOS/OS9000/80386/PORTS/<BOARD>/systype.h`

```
#define DIGIPORT    0xe0       /* port address of DIGI board status reg. */
#define DIGILEVEL   0x45         /* 16450 keyboard controller */
#define DIGIVECTOR  DIGILEVEL    /* irq vector same as irq level */

#define T10PORT     0x320        /* t10 onboard port address */
#define T11PORT     0x328        /* t11 onboard port address */
#define T12PORT     0x330        /* t12 onboard port address */
#define T13PORT     0x338        /* t13 onboard port address */
#define T14PORT     0x340        /* t14 onboard port address */
#define T15PORT     0x348        /* t15 onboard port address */
#define T16PORT     0x350        /* t16 onboard port address */
#define T17PORT     0x358        /* t17 onboard port address */
```

Making the Descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory: `MWOS/OS9000/80386/PORTS/PCAT/SCF/SCPC8/DESC`

Type `os9make`; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t10
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t11
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t12
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t13
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t14
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t15
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t16
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCPC8\t17
```

HostessI

Support for the HostessI intelligent serial card is included by selecting the HostessI option in the Configuration Wizard's Bootfile Options dialog box.

To change the vector you must edit the `systype.h` file located in the port directory.

```
MWOS/OS9000/80386/PORTS/<BOARD>/systype.h
#define HS_PORT            0x00000218/* Hostess i board. serial adapter
                           board */
#define HS_VECT            0x4f/* IRQ 15 */
#define HS_BOARDMEM        0xd0000/* onboard memory place in the system
                           address space */
#define HS_NBLINES         16/* number lines on the board (8/16) */
/* Old board doesn't permit 16 bits mode. */
#define HS_BUSSIZE         8/* size of the bus the board uses (8/16) */
```

Making the Descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory: `MWOS/OS9000/80386/PORTS/PCAT/SCF/SCHOST/DESC`

Type `os9make` the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t40
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t41
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t42
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t43
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t44
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t45
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t46
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t47
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t48
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t49
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t50
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t51
```

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t52
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t53
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t54
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCHOST\t55
```

Risicom

Support for the Risicom8 intelligent serial card is included.

To change the vector you must edit the `systype.h` file located in the port directory.

```
MWOS/OS9000/80386/PORTS/PCAT/systype.h
#define RC8BASE            0x00000220/* Riscom8 serial port adapter */
#define RC8VECT            0x45/* IRQ 5 */
```

Making the Descriptors

Once the `systype.h` file has been updated the new descriptors may be created.

Change to directory: `MWOS/OS9000/80386/PORTS/PCAT/SCF/SCPC8/DESC`

Type `os9make`; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t20
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t21
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t22
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t23
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t24
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t25
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t26
MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCRISCOM\t27
```

The Risicom board has not been verified with this driver for this release.

Parallel Printer

Standard PC style printer support is included.

To change the vector or port address you must edit the `systype.h` file located in the port directory.

```
MWOS/OS9000/80386/PORTS/<BOARD>/systype.h
#define PLEVEL0x47/* scp87303 parallel port */
#define PVECTPLEVEL/* irq vector same as irq level */
#define LPT1BASE           0x000003bc/* base address of first parallel
                           port */
```

```
#define LPT2BASE          0x00000378/* base address of second parallel
                          port */
#define LPT3BASE          0x00000278/* base address of third parallel
                          port */
```

Making the Descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory: `MWOS/OS9000/80386/PORTS/PCAT/SCF/SCP87303/DESC`

Type `os9make`; the following descriptors will be made:

`MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCP87303\p.lp1`

`MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCP87303\p.lp2`

`MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\SCP87303\p.lp3`

Physical Disk Media

IDE Standard

Support for IDE based devices, including standard IDE based hard disk. Primary and secondary controllers with master and slave drive support. On some embedded systems Compact Flash supported devices may be used as if they are standard PC AT based devices.

Benefits

- Supports large media (8.5GB maximum).
- PIO mode three supported.
- PC File system supported including long filenames (FAT32 is not supported). Boot support (requires OS-9 coreboot load).
- Native RBF file system supported. Full boot support including IPL boot technology.

The standard configuration assumes the primary controller is located at 0x1f0 with IRQ 14 and secondary controller at 0x170 with IRQ 15. You may, however, change these values as needed to suit the target. The values are based on the contents of the files `MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des` and `MWOS/OS9000/80386/PORTS/<BOARD>/systype.h`.

The pertinent lines in `MWOS/OS9000/80386/PORTS/<BOARD>/systype.h` are shown below:

```
#if defined(RB1003_SPEC_IO_ADDRESS) /* PCMCIA */

#defineBASE_RB1003_PRI 0x00000320/* IDE controller port addr */
#defineVECT_RB1003_PRI0x0/* IDE controller vector */
#defineBASE_RB1003_SEC0x00000360/* IDE 2nd controller port */
#defineVECT_RB1003_SEC0x0/* IDE 2nd controller vector */

#else

#defineBASE_RB1003_PRI0x000001f0/* IDE controller port addr */
#defineVECT_RB1003_PRI0x4e/* IDE controller vector */
#defineBASE_RB1003_SEC0x00000170/* IDE 2nd controller port */

#defineVECT_RB1003_SEC0x4f/* IDE 2nd controller vector */

#endif
```

while in `MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des` the portion of interest is shown below:

```
/ * Device specific defines
 *
 * ds_idetype = IDE interface type
 *              IDE_TYPE_STANDARD
 *              IDE_TYPE_PCI
 *              IDE_TYPE_PCMCIA
 *
 * ds_polled =  IDE_POLLED
 *              IDE_INTERRUPTS
 *
 * ds_altstat = HD_DEFAULT_ALTSTAT (Standard IDE offset)

 *        HD_PCMCIA_ALTSTAT (PCMCIA IDE offset)
 *
 * ds_timeout = Drive ready timeout in seconds.
```

```
 *        IDE specification allows for up to 30 seconds. Allow the max.
 *      Users are free to reduce this amount if desired. PCMCIA IDE FLASH
 *        type cards require a few miliseconds. Rotating devices requires
 *        more time.

#define IDE_TYPE_STANDARD 0

#define IDE_TYPE_PCI 1
#define IDE_TYPE_PCMCIA 2

#define IDE_INTERRUPTS 0
#define IDE_POLLED 1

#define HD_DEFAULT_ALTSTAT 0x0206
#define HD_PCMCIA_ALTSTAT 0xe


init dev_specific {

#if defined(RB1003_SPEC_IO_ADDRESS)
   ds_idetype = IDE_TYPE_PCMCIA;
   ds_polled =  IDE_POLLED;
   ds_altstat = HD_PCMCIA_ALTSTAT;
   ds_timeout = 30;

#else

   ds_idetype = IDE_TYPE_STANDARD;
   ds_polled =  IDE_INTERRUPTS;
   ds_altstat = HD_DEFAULT_ALTSTAT;
   ds_timeout = 30;

#endif
);
```

Since OS-9 does not require the BIOS to use IDE it is possible on some systems to use IDE without interrupts. Keep in mind that on some systems disabling the IDE from the BIOS also disables the IDE controller as well.

Drive time-out may also fail on drives that are extremely old. If you are having problems using drives that are less than 540MB you may want to disable the time-out. This can be done by setting time-out value to zero in config.des and re-making the descriptors and boot image.

Using IDE in PCI Mode

Support is included to support IDE devices as PCI specific devices. PCI based IDE support is not automatic and may not work on some PCI bridges. The `rb1003` driver must be re-made with the following changes to the makefile.

```
PCILIB          =       -l=$(PORT)/LIB/pcilib.l

LIB             =       $(PICLIB) $(PCILIB) \

                        $(CPULIB) $(CLIB) $(P2LIB) $(OS_LIB) $(SYS)

SPEC_COPTS      =      -a -c -r -t=0 -bepg -dNEWINFO $(PICISR) $(IRQMASK) \

                        -dPCI
```

In this case PCILIB has been added; in addition, PCI has been defined in the SPEC_COPTS section. On some systems that use both primary and secondary controllers that allow level interrupt to be set and used in PCI standard method, you can save one interrupt vector. You must also set the device type to PCI in the config.des file shown above. You must have the sources for RB1003 for the ability to make this change using the cross hosted utilities.

If the PCI bridge does not work in PCI mode you can modify the RB1003 init code as need for the PCI bridge device used. The sources are located in `MWOS/OS9000/SRC/IO/RBF/DRVR/RB1003`, and are included with the Embedded Systems package.

Use of IDE in PCI mode adds about 2K to the driver size.

RBF

OS-9 RBF native file system may be used on any IDE drive. For more information see BootGen and `IDE Descriptors`.

PCF

A PC style file system is also supported. If access to partitions other then the primary are required you may use the pinfo utility to obtain the information required to create specific device descriptors. For more information see `IDE Descriptors`. You may select the PCF file system as the boot media.

For example. If the drive is FAT you may place the bootfile image on the root. Make sure it is called `os9kboot`. Next, create a CMDS and SYS directory at the root level. Copy whatever CMDS you need to the CMDS directory. Create a startup and or password file as needed. This method allows you to use the same partition as Windows95 or NT when you actually run OS-9.

Prepare Windows95/NT based system for use with OS-9.

```
md C:\CMDS
md C:\SYS


copy MWOS\OS9000\80386\CMDS\* C:\CMDS


copy MWOS\OS9000\80386\PORTS\<BOARD>\BOOTS\INSTALL\SYS\MSHELL\startup
C:\SYS


copy MWOS\OS9000\80386\PORTS\<BOARD>\BOOTS\INSTALL\SYS\MSHELL\password
C:\SYS


cd C:\SYS


cudo -cdo startup
cudo -cdo password
```

Although RBF is the preferred file system for use with OS-9 the convenience of using FAT file systems should be taken into consideration when deciding how you want to setup your system.

Special Note

In the following example the IDE device for /h0 and /dd is set for IDE primary partition four.

If the init dialog is set to /h0 the following is generated. In this case you also have SoftStax SPF enabled.

```
setenv SHELL mshell; alias /dd /hc4;chd /h0 ; chx /h0/cmds;mbinstall;


   ipstart;inetd <>>>/nil&;/h0/sys/startup &\n
```

If the init dialog is set to /dd the following is generated. In this case you also have SPF enabled.

```
setenv SHELL mshell; alias /dd /hc4;chd /dd ; chx /dd/cmds;mbinstall ;


          ipstart;inetd <>>>/nil&;;/dd/sys/startup &\n
```

In both cases the script file on hc4 in sys/startup will be executed. When building systems this file must exist, but does not have to contain data. The following commands suffice to create the expected directory and file:

```
$ makdir /hc4/SYS
```

```
$ touch /hc4/startup
```

It is usually best to create the initial boot image to not use /h0. /dd should be set for RAM disk. This will allow downloading the TAR images. Next setup the final boot image and select /h0 as initial device name.

Descriptors

Refer to `IDE Descriptors` for information on descriptor naming conventions. The descriptors for RB1003 are located in `MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RB1003`. Also the `RB1003` driver is located in `MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS`.

ROM BOOTING

If changes to the IDE addresses of time-out values are employed, then the ROM boot system may also require changes.

```
MWOS/OS9000/80386/PORTS/<BOARD>/ROM/cnfgdata.des
```

Find the following sections:

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=30"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=30"
```

To remove time-out for example you could change the above to:

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=0"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=0"
```

Or you could make the time-out shorter. IDE specification indicates you can wait up to 30 seconds.

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=5"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=2"
```

Advanced Notes

Some embedded systems include support for Compact Flash, which looks like a standard IDE device. In these cases, you may decide that RBF is the file system of choice, since you can boot the embedded board with no other boot devices installed. Compact Flash devices will work in PCMCIA systems with a carrier, so that you can use a standard PC with PCMCIA support to build up the PCMCIA disk. Once the disk is built, you can then remove the Compact Flash from the carrier and place it in the target system for use.

PCMCIA IDE

Microware PCMCIA socket services are included with all PCMCIA selections. When making bootfile only images care should be taken to make sure PCMCIA support is enabled in the low-level 'coreboot' system if PCMCIA devices are to be employed once the system is booted.

Support for IDE based devices including standard PCMCIA IDE based hard disk.

Benefits

- Supports large media(8.5GB maximum).
- PIO mode three supported.
- PC File system supported including long filenames. Boot support (requires OS-9 coreboot load).
- Native RBF file system supported. Full boot support including IPL boot technology (PCMCIA BIOS BOOT support required if this option is used).
- Requires no interrupts. Interrupts are optional.

The standard configuration assumes socket #0 is mapped to 0x320 and socket #1 is mapped to 0x360. The default configuration does not use interrupts. You may however enable interrupts if desired.

Example (Enable interrupts on PCMCIA device in socket #0 only - IRQ5 used)

```
/*

  * MWOS/OS9000/80386/PORTS/<BOARD>/systype.h file.

*/

#defineBASE_RB1003_PRI0x00000320/* IDE controller port addr */
#defineVECT_RB1003_PRI0x45/* IDE controller vector */

/*
  * MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des
*/

    ds_idetype = IDE_TYPE_PCMCIA;
    ds_polled = IDE_INTERRUPTS;
    ds_altstat = HD_PCMCIA_ALTSTAT;
    ds_timeout = 30;

MWOS/OS9000/80386/PORTS/<BOARD>/ROM/cnfgdata.des

#define IDE_CIS_PARAMS "ide0=0x320,5 ide1=0x360,0"
```

Once the changes are made change to the MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/DESC directory and type os9make. The changes to `MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des` are automatically taken care of next time you run the Wizard.

Changes to the default values are based on the `MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des` file as well as the `MWOS/OS9000/80386/PORTS/<BOARD>systype.h` file.

```
MWOS/OS9000/80386/PORTS/<BOARD>systype.h
#if defined(RB1003_SPEC_IO_ADDRESS) /* PCMCIA */

#defineBASE_RB1003_PRI0x00000320/* IDE controller port addr */
#defineVECT_RB1003_PRI0x0 /* IDE controller vector */
#defineBASE_RB1003_SEC0x00000360/* IDE 2nd controller port */
#defineVECT_RB1003_SEC0x0/* IDE 2nd controller vector */

#else

#defineBASE_RB1003_PRI0x000001f0/* IDE controller port addr */
#defineVECT_RB1003_PRI0x4e/* IDE controller vector */

#defineBASE_RB1003_SEC0x00000170/* IDE 2nd controller port */
#defineVECT_RB1003_SEC0x4f/* IDE 2nd controller vector */

#endif

MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des

/*
 * Device specific defines
 *
 * ds_idetype = IDE interface type
 *              IDE_TYPE_STANDARD
 *              IDE_TYPE_PCI
 *              IDE_TYPE_PCMCIA
 *
 * ds_polled =  IDE_POLLED
 *              IDE_INTERRUPTS
 *
 * ds_altstat = HD_DEFAULT_ALTSTAT (Standard IDE offset)

 *              HD_PCMCIA_ALTSTAT (PCMCIA IDE offset)
 *
 * ds_timeout = Drive ready timeout in seconds.
 *              IDE specification allows for up to 30 seconds. Allow max.
 *             Users are free to reduce this amount if desired. PCMCIA IDE
 *              FLASH type cards require a few miliseconds. Rotating
 *              devices require more time. */
```

```
#defineIDE_TYPE_STANDARD0

#defineIDE_TYPE_PCI1
#defineIDE_TYPE_PCMCIA2

#defineIDE_INTERRUPTS0
#defineIDE_POLLED1

#defineHD_DEFAULT_ALTSTAT0x0206
#defineHD_PCMCIA_ALTSTAT0xe

init dev_specific {

#if defined(RB1003_SPEC_IO_ADDRESS)
    ds_idetype = IDE_TYPE_PCMCIA;
    ds_polled =  IDE_POLLED;
    ds_altstat = HD_PCMCIA_ALTSTAT;
    ds_timeout = 30;

#else

    ds_idetype = IDE_TYPE_STANDARD;
    ds_polled =  IDE_INTERRUPTS;
    ds_altstat = HD_DEFAULT_ALTSTAT;
    ds_timeout = 30;


#endif
};
```

RBF

OS-9 RBF native file system may be used on any IDE drive including PCMCIA devices. For more information see BootGen and IDE DESCRIPTORS. When using RBF with PCMCIA only OS-9 will be able to access the media. When running FDISK on PCMCIA media, be sure to write down the ID type. You will need this value if you decide to later restore the media for use with DOS/ Windows. fdisk -d=/pchcfmt -s will show the type. If you need to restore the PCMCIA IDE card for use with DOS/Windows you must restore the ID type. If you have PCMCIA support at the DOS level you may be able to use FDISK. If not you can use Linux to change the ID type. you may add this feature to OS-9 fdisk in the future but be warned: once the device is changed to RBF if you do not have the tools then this disk will have to stay RBF.

PCF

PC style file system is also supported. For more information see `IDE Descriptors`.

You may select the PCF file system as the boot media.

For example, if the drive is FAT, you may place the bootfile image on the root. Make sure it is called os9kboot. Next create a CMDS and SYS directory at the root level. Copy whatever CMDS you need to the CMDS directory. Create a startup and or password file as needed. This method allows you to use the same partition as Windows95 or NT when you actually run OS-9.

Prepare Windows95/NT based system for use with OS-9.

```
md C:\CMDS


md C:\SYS


copy MWOS\OS9000\80386\CMDS\* C:\CMDS


copy MWOS\OS9000\80386\PORTS\<BOARD>\BOOTS\INSTALL\SYS\MSHELL\startup
C:\SYS


copy MWOS\OS9000\80386\PORTS\<BOARD>\BOOTS\INSTALL\SYS\MSHELL\password
C:\SYS


cd C:\SYS


cudo -cdo startup


cudo -cdo password
```

Although RBF is the preferred file system for use with OS-9 the convenience of using FAT file systems should be taken into consideration when deciding how you want to setup your system.

Special Note

In the following example the IDE device for `/h0` and `/dd` is set for PCMCIA IDE using socket #0.

If the init dialog is set to /h0 the following is generated. In this case you also have SoftStax SPF enabled.

```
setenv SHELL mshell; alias /dd /pcmhc1;chd /h0 ; chx /h0/cmds;mbinstall ;


        ipstart;inetd <>>>/nil&;/h0/sys/startup &\n
```

If the init dialog is set to /dd the following is generated. In this case you also have SoftStax SPF enabled.

```
setenv SHELL shell; alias /dd /pcmhc1;chd /dd ; chx /dd/cmds;mbinstall ;
          ipstart;inetd <>>>/nil&;/dd/sys/startup &\n
```

In both cases above the script file on hc4 in sys/startup will be executed. When building systems this file must exist but does not have to contain any data. To create the needed directory and file, the following commands suffice:

```
$ makdir /pcmhc1/SYS
$ touch /pcmhc1/SYS/startup
```

It is usually best to create the initial boot image to not use /h0. /dd should be set for RAM disk. This will allow downloading the TAR images. Next setup the final boot image and select /h0 if as initial device name.

Descriptors

Refer to `IDE Descriptors` for information on descriptor naming conventions. The descriptors for RB1003 are located in `MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RB1003`. Also the RB1003 driver is located in `MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS`.

ROM BOOTING

If changes to the IDE addresses of time-out values are employed then the ROM boot system may also require changes.

```
MWOS/OS9000/80386/PORTS/<BOARD>/ROM/cnfgdata.des
```

Find the following sections:

```
#define IDE_CIS_PARAMS "ide0=0x320,0 ide1=0x360,0"


#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=30 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=30
altstat=0xe"
```

To remove time-out for example you could change the above to:

```
#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=0 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=0
altstat=0xe"
```

Or you could make the time-out shorter. IDE specification indicates you should wait up to 30 seconds.

```
#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=5 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=2
altstat=0xe"
```

To explain the definition of IDE_CIS_PARAMS in detail: "ide0=0x320,5 ide1=0x360,0" indicates that IDE0 (socket 0) has a base address of 0x320 and uses IRQ 5, while IDE1 (socket 1) has a base address of 0x360 and uses no interrupt.

Advanced Notes

Some embedded systems support Compact Flash, which looks like a standard IDE device. In these cases, RBF may be the file system of choice, since the embedded board can boot with no other boot devices installed. Compact Flash devices work in PCMCIA systems with a carrier, so that a standard PC can be used with PCMCIA support to build up the PCMCIA disk. Once the disk is built, the Compact Flash can then be removed from the carrier and placed in the target system for use.

IDE Descriptors

For Standard IDE devices the devices are referenced as shown in the following table.

Standard IDE - RBF Descriptors

/hcfmt	IDE primary master - Entire disk
/hc1fmt	IDE primary master - Primary partition #1
/hc2fmt	IDE primary master - Primary partition #2
/hc3fmt	IDE primary master - Primary partition #3
/hc4fmt	IDE primary master - Primary partition #4
/hdfmt	IDE primary slave - Entire disk
/hd1fmt	IDE primary slave - Primary partition #1
/hd2fmt	IDE primary slave - Primary partition #2
/hd3fmt	IDE primary slave - Primary partition #3
/hd4fmt	IDE primary slave - Primary partition #4
/hefmt	IDE secondary master - Entire disk
/he1fmt	IDE secondary master - Primary partition #1
/he2fmt	IDE secondary master - Primary partition #2
/he3fmt	IDE secondary master - Primary partition #3
/he4fmt	IDE secondary master - Primary partition #4
/hffmt	IDE secondary slave - Entire disk
/hf1fmt	IDE secondary slave - Primary partition #1
/hf2fmt	IDE secondary slave - Primary partition #2
/hf3fmt	IDE secondary slave - Primary partition #3
/hf4fmt	IDE secondary slave - Primary partition #4

Standard IDE - PCF Descriptors

/mhc1	IDE primary master - Primary partition #1
/mhc2	IDE primary master - Primary partition #2
/mhc3	IDE primary master - Primary partition #3

/mhc4	IDE primary master - Primary partition #4
/mhd1	IDE primary slave - Primary partition #1
/mhd2	IDE primary slave - Primary partition #2
/mhd3	IDE primary slave - Primary partition #3
/mhd4	IDE primary slave - Primary partition #4
/mhe1	IDE secondary master - Primary partition #1
/mhe2	IDE secondary master - Primary partition #2
/mhe3	IDE secondary master - Primary partition #3
/mhe4	IDE secondary master - Primary partition #4
/mhf1	IDE secondary slave - Primary partition #1
/mhf2	IDE secondary slave - Primary partition #2
/mhf3	IDE secondary slave - Primary partition #3
/mhf4	IDE secondary slave - Primary partition #4

CDROM IDE Descriptors

`/cd0IDE` secondary master

PCMCIA IDE - RBF Descriptors

/pchcfmt	PCMCIA IDE Socket #0 - Entire disk
/pchc1fmt	PCMCIA IDE Socket #0 - Primary partition #1
/pchefmt	PCMCIA IDE Socket #1 - Entire disk
/pche1fmt	PCMCIA IDE Socket #1 - Primary partition #1

PCMCIA IDE - PCF Descriptors

/pcmhc1	PCMCIA IDE Socket #0 - Primary partition #1
/pcmhe1	PCMCIA IDE Socket #1 - Primary partition #1

The descriptors for IDE are automatically included when using the Wizard. You may also access the descriptors in the MWOS directory structure at:
`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RB1003`

DiskOnChip

Overview

M-Systems' DiskOnChip™ (DOC) is a generation of single-chip flash disks. The DOC device contains built-in firmware that provides full, hard disk emulation and allows the DiskOnChip to operate as a boot device.

When used under OS-9, the DiskOnChip is managed by a TrueFFS™, technology-based device driver, attached to the standard OS-9 file system (RBF) or to a DOS compatible file system (PCF). In addition, a native RBF file system is supported, including full-boot support with IPL boot technology.

The following sections are intended for systems integrators designing with the DiskOnChip 2000, DiskOnChip Millennium or DiskOnChip DIMM and describe how to use the DiskOnChip as a bootable data storage device under the OS-9 operating system.

In the following sections, the term *DiskOnChip* is used to described the aforementioned DiskOnChip family of products.

Low- and High-Level Boot Support

- Low-Level Support

```
OS9000/80386/CMDS/BOOTOBJS/ROM/doc
```

- High-Level Support

```
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/rbdoc
```

Required Software

- MS-DOS boot floppy
- M-Systems DOC utilities disk (v5.14)
- OS-9 for x86 boot disk with low- and high-level DOC support

In setting up your host machine for the DOC boot, be certain the BIOS settings are set such that any hard disks are disabled; be sure the system is booting from floppy before it is booting from a hard disk. An example setting may look like the example below:

```
PC BIOS settings
DISK A:          1.44 MB, 3 ½
DISK B:          Not Installed
IDE Adapter 0 Master: None
IDE Adapter 0 Slave: None
IDE Adapter 1 Master: None
IDE Adapter 1 Slave: None
Boot Sequence:   A: then C:=
```

Descriptors Used by the Configuration Wizard

```
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RBDOC/dochcfmt
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RBDOC/dochc1
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RBDOC/dochc1fmt
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RBDOC/dochc1.h0
```

Additional descriptors are provided in the following location:

```
OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RBDOC
```

Please refer to DiskOnChip Descriptors section for information on the use of these other descriptors.

Formatting DiskOnChip for DOS

By default, the DiskOnChip firmware installs the DOC as an additional disk in the system. This allows you to boot an operating system from the DOC on a diskless machine. If your machine contains other hard disks, but you still want to boot from the DOC, you will need to specify the DOC as default.

In order to install the DOC as the first drive, boot your target system into MS-DOS and enter the following command:

```
DUPDATE /WIN:{address} /S:DOC121.EXB /FIRST
```

- `{address}` is the base address of the DOC (such as `D000` or `D400`).
- `121` in the file `DOC121.EXB` represents the firmware version. The actual firmware version may be greater (such as `DOC122.EXB`).

The `DUPDATE` utility and firmware files are provided with the DiskOnChip ISA evaluation board available from M-Systems.

The default base address for the M-System's evaluation board is `D000h`. Refer to the documentation included with your hardware for the base address and board jumper settings.

If you do not need to access additional hard disk(s) under OS-9, you may also disable them in the CMOS setup. In this case, the DOS command above is not necessary.

In some cases it is useful to prevent the DiskOnChip firmware from installing at boot time. You can achieve this by typing the following DOS command:

```
DUPDATE /WIN:{address} /S:DOC2.FFF
```

Step 1. Boot to a DOS floppy disk, then insert the M-Systems utility disk. Format the drive with M-Systems `dformat` utility and update the firmware to version 5.1.4:

```
dformat /win:d800 /s:doc514.exb /first

When notified that all data on DiskOnChip will be destroyed.  Continue?
Reply Y.
```

Step 2. When done formatting the device with `dformat`, reboot the machine with a DOS floppy.

Step 3. Transfer the DOS systems files onto DiskOnChip.

```
Example: If DOC is c:, type sys c:.
```

Step 4. To verify that the system boots properly, remove any floppy disks and reboot your system. If a DOS prompt appears, the transfer was successful.

Building a DiskOnChip Image for OS-9

This section takes you through the process for building a boot image with OS-9 Configuration Wizard, including instructions for configuring both low- and high-level images (`coreboot + bootfile`).

The DiskOnChip distribution is made up of two primary modules:

- `doc`: the OS-9 low-level booter module
- `rbdoc`: the OS-9 device driver for DiskOnChip

The DiskOnChip device appears as a disk drive to the high-level system and is accessed using the RBF descriptor, `/dochc1`.

To build the DiskOnChip image, complete the following steps:

Step 1. From your host machine, select `Start -> Programs -> RadiSys -> Configuration Wizard` to open the Configuration Wizard.

Step 2. From the Wizard's opening screen, complete the following steps:

1. Select your target from the **Select a Board** list.
2. Select the Create a new Configuration radio button and type `DiskOnChip` for the configuration name in the box provided.
3. Be sure the **Advanced Mode** radio button is selected and click `OK` to proceed into the Wizard.

Step 3. From the Wizard's main menu, select `Configure -> Coreboot -> Disk Configuration`. The `IDE Configuration` tab should display.

Step 4. In the IDE Configuration tab, select the `Add to menu` and `Auto Boot` check boxes in the DiskOnChip area. This will include modules into your coreboot file that allow the system to boot from the DOC device. Click the `OK` button to exit the Disk Configuration dialog.

This step makes the DOC device the default boot media. Note that if an OS-9 boot image is not found on the DOC, OS-9 will attempt to boot from the floppy disk.

Step 5. From the Wizard's main menu, select `Configure -> Bootfile -> Disk Configuration`. Select the `IDE Configuration` tab.

Step 6. In the DiskOnChip area of the IDE Configuration tab, select the `Enable DOC` check box. This will include modules into your bootfile that allow you to format the DOC device.

Step 7. Select the `Init Options` tab.

Do not make the DOC device the default disk in the high-level configuration.

Step 8. Click `OK` to exit the Disk Configuration dialog.

Step 9. From the main menu, select `Configure -> Build Image`. The Master Builder dialog appears.

Step 10. In the Master Builder dialog, select the `Coreboot + Bootfile` radio button. Then, select the `Check` button to verify that your image will fit on the device. When you are satisfied with the size and contents of your image, select the `Build` button. The build may take a few minutes.

Step 11. Once the build process is finished, select the `MakeBoot` button and follow the prompts to create a bootable OS-9 floppy.

Step 12. Click the `Finish` button to close the Master Builder dialog.

Formatting the OS-9 Partition

Once you have built the DiskOnChip image in the Configuration Wizard, you will need to create and format the partition for OS-9. To do this, complete the following steps:

Step 1. Boot your PC with the OS-9 boot disk you created in the previous section.

Step 2. From the OS-9 prompt, run `fdisk` on the device by typing the following command:

```
fdisk -d=/dochcfmt
```

Step 3. The `fdisk` menu should display. From here, select `option 4` to display the partition; the only partition on DOC should be the "FAT-12 DOS" partition.

It is important that you are able to see the "DOS FAT-12" partition first when you start `fdisk` and select `option 4` to display the partitions; if you do not see the DOS partition, the DiskOnChip boot will not work. It is also important that you do not use `option 6` to write the master boot record (MBR). This will destroy information needed to boot the DOC device.

Step 4. Select `Esc` to return to the `fdisk` menu, then select `option 5` to change the partition to the OS-9 partition.

Step 5. Specify the DOS partition and select `1` to convert it to an OS-9/80386 partition.

Step 6. Select `Esc` to return to the main menu, then select `option 2` to set the partition as “active.”

Step 7. Specify the partition, then select `Esc` to return to the `fdisk` menu.

Step 8. Select `Esc` to exit the `fdisk` menu and select `y` to save your changes.

To format the partition, type the following command at the OS-9 prompt:

```
format /dochc1fmt -nv -np -r -v
```

Step 9. Place your boot on the device by typing the following command:

```
bootgen /dochc1fmt -i=/d0/iplhdnoq -l=/d0/firstboot /d0/sysboot -nb1024
```

If you do want the DOC device to be the default disk (`/dd`), create a new boot disk with this option before doing the `bootgen` command. You may also have to create a `SYS` directory with a password and startup file and a `SYS` directory if you want to make DOC your default disk.

Step 10. Boot the host machine with the OS-9 floppy.

Step 11. Initialize the DOC device by typing the following command:

```
chd /d0
iniz /dochcfmt
```

Step 12. Remove any floppy disks from the floppy drive and reboot your target. The OS-9 IPL message will appear briefly, followed by the message:

```
OS-9000/x86 Bootstrap
Now trying to Override autobooters.
```

At this point, the floppy booter will fail because there is no floppy disk in the drive. This will cause the DiskOnChip booter to read the OS-9 bootfile from the DiskOnChip device and the system to boot to a shell prompt.

DiskOnChip Descriptors

In the `MWOS\OS9000\80386\PORTS\PCAT\CMDS\BOOTOBJS\DESC\RBDOC` directory there are numerous device descriptors for both RBF and PCF filesystems. Note that the table below omits descriptors with the filename extension .h0 - these files are also present, and contain device descriptors with the canonical name h0, useful for systems whose main disk unit will be a DiskOnChip device.

DiskOnChip - RBF Descriptors

/dochcfmt	DiskOnChip Device 0 - Entire disk
/dochc1fmt	DiskOnChip Device 0 - Primary partition #1
/dochc2fmt	DiskOnChip Device 0 - Primary partition #2
/dochc3fmt	DiskOnChip Device 0 - Primary partition #3
/dochc4fmt	DiskOnChip Device 0 - Primary partition #4
/dochdfmt	DiskOnChip Device 1 - Entire disk
/dochd1fmt	DiskOnChip Device 1 - Primary partition #1
/dochd2fmt	DiskOnChip Device 1 - Primary partition #2
/dochd3fmt	DiskOnChip Device 1 - Primary partition #3
/dochd4fmt	DiskOnChip Device 1 - Primary partition #4
/dochefmt	DiskOnChip Device 2 - Entire disk
/doche1fmt	DiskOnChip Device 2 - Primary partition #1
/doche2fmt	DiskOnChip Device 2 - Primary partition #2
/doche3fmt	DiskOnChip Device 2 - Primary partition #3
/doche4fmt	DiskOnChip Device 2 - Primary partition #4
/dochffmt	DiskOnChip Device 3 - Entire disk
/dochf1fmt	DiskOnChip Device 3 - Primary partition #1
/dochf2fmt	DiskOnChip Device 3 - Primary partition #2
/dochf3fmt	DiskOnChip Device 3 - Primary partition #3

/dochf4fmt	DiskOnChip Device 3 - Primary partition #4

DiskOnChip - PCF Descriptors

/docmhcfmt	DiskOnChip Device 0 - Entire disk
/docmhc1fmt	DiskOnChip Device 0 - Primary partition #1
/docmhc2fmt	DiskOnChip Device 0 - Primary partition #2
/docmhc3fmt	DiskOnChip Device 0 - Primary partition #3
/docmhc4fmt	DiskOnChip Device 0 - Primary partition #4
/docmhdfmt	DiskOnChip Device 1 - Entire disk
/docmhd1fmt	DiskOnChip Device 1 - Primary partition #1
/docmhd2fmt	DiskOnChip Device 1 - Primary partition #2
/docmhd3fmt	DiskOnChip Device 1 - Primary partition #3
/docmhd4fmt	DiskOnChip Device 1 - Primary partition #4
/docmhefmt	DiskOnChip Device 2 - Entire disk
/docmhe1fmt	DiskOnChip Device 2 - Primary partition #1
/docmhe2fmt	DiskOnChip Device 2 - Primary partition #2
/docmhe3fmt	DiskOnChip Device 2 - Primary partition #3
/docmhe4fmt	DiskOnChip Device 2 - Primary partition #4
/docmhffmt	DiskOnChip Device 3 - Entire disk
/docmhf1fmt	DiskOnChip Device 3 - Primary partition #1
/docmhf2fmt	DiskOnChip Device 3 - Primary partition #2
/docmhf3fmt	DiskOnChip Device 3 - Primary partition #3
/docmhf4fmt	DiskOnChip Device 3 - Primary partition #4

PCAT-Style Floppy

Standard floppy support is provided using the RB765 driver. /d0 may be used to access RBF native file system. /md0 may be used to access PC style floppy devices.

Floppy Descriptors

Floppy - RBF Descriptors

/d0	Floppy drive A:

Floppy - PCF Descriptors

/md0 Floppy drive A:

When using the Wizard the descriptors for floppy devices are automatically included. You may also access the descriptors in the MWOS directory structure:
`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/RB765`

Symbios 810,810A,825,825A and 875 PCI SCSI Controllers

Benefits

- Wide support
- Ultra FAST20 support
- SCRIPTS RAM support (able to run scripts from on-chip RAM)
- Large FIFO enabled
- Increased burst rates to 128 where supported
- Special PCI cache features enabled
- PCI IO Mode selectable (PCI I/O or PCI Memory)

The SCRIPTS RAM support is currently only available on OS-9 x86 based systems. Requires non translation of PCI memory. To use SCRIPTS RAM support include the "-dSCRIPTS_RAM" in the compile line when making the driver.

Instruction prefetch is not enabled by default. Maximum burst rate and large fifo's are enabled.

By default the Microware Symbios driver will use the PCI I/O model. To speed up transfers, especially on x86 platforms, the memory mode may be used. In the PCI memory mode no in/out instructions are used. For the x86 platform this removes the CPU related waits added by the use of "inc", "outc" etc. If you want to run the driver in PCI Memory mode the driver may be recompiled with the "-dPCI_IO_MAPPED" flag removed from the
`MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile`

IO_MAPPED=-dPCI_IO_MAPPED

To use memory model change to:

IO_MAPPED=# -dPCI_IO_MAPPED

The default has changed to IO_MAPPED for x86 due to problems on PCAT-based motherboards.

Prior to this release the following Symbios devices were supported the following devices:

Number of devices supported (2)

`DEVICEWIDEULTRA1 ULTRA2 FIFO_SIZEBURST`

```
------------------------------------------------------------------
Symbios 53c810N/AN/AN/A 6416

Symbios 53c825NoN/AN/A  8816
```

This release adds the following:

Number of devices supported (12)

```
DEVICEWIDEULTRA1 ULTRA2 FIFO_SIZEBURST
-------------------------------------------------------------
Symbios 53c810N/AN/AN/A 6416
Symbios 53c810APN/AN/AN/A6416 (1)
Symbios 53c815N/AN/AN/A 6416 (1)
Symbios 53c820YesN/AN/A 8816 (1)
Symbios 53c825YesN/AN/A 8816
Symbios 53c825AYesN/AN/A536128

Symbios 53c875YesYESN/A 536128
Diamond FirePort20YesN/AN/A536128 (825A)
Diamond FirePort40YesYESN/A536128 (875)
Symbios 53c860YesYESN/A 536128 (1)
Symbios 53c885YesYESN/A 536128 (1)
Symbios 53c895YesYESYES 536128 (1,2)
Symbios 53c896YesYESYES 536128 (1,2)
```

(1) Support is included but untested.

(2) Support for 895 and 896 is only available with out ULTRA support. The 160Mhz clock will be enabled on a future release.

The 895 and 896 have not been tested.

[Symbios 53C810]

[Symbios 53C810A]

Device supports burst op code fetch

Device supports instruction prefetch

Device supports Cache Line Size and Cache Commands

[Symbios 53C810ALV] * same as 810

Device supports burst op code fetch

Device supports instruction prefetch

Device supports Cache Line Size and Cache Commands

[Symbios 53C815]

Device supports burst op code fetch

[Symbios 53C825]

Device supports Wide SCSI data transfers

Device supports burst op code fetch
[Symbios 53C825A]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
[Symbios 53C860]
Device supports burst op code fetch
Device supports instruction prefetch
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers
[Symbios 53C875]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers
[Symbios 53C885]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers
Device supports Clock Doubler
[Symbios 53C895]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers (Not supported yet)

Device supports Clock Doubler (Not supported yet)

Device supports Fast-40 transfers (Not supported yet)

[Symbios 53C896]

Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers (Not supported yet)

Device supports Clock Doubler (Not supported yet)

Device supports Fast-40 transfers (Not supported yet)

Using Ultra Fast20 and Wide support.

Controller Dependency

For FAST20 support the controller must support FAST20.

Device Descriptors

To use a device with disconnect, wide, synchronous data transfer, and FAST20 Ultra the following should be added to the device descriptor entry in "systype.h". Be sure to re-make the descriptors.

```
#define SCSIOPTS SCSI_ATN|SCSI_SYNC|SCSI_WIDE|SCSI_ULTRA
```

Optionally you may use EditMod to change the SCSIOPTS field. For SYNC and ATN the SCSIOPTS value is "5".

Using Multiple SCSI Controllers

It is possible to use multiple SCSI controllers with the Symbios family of controllers.

The port address is used to specify the card to use.

PortAddress format.

[0xff] [device] [index] [SCSI_ID]

device = device number. Use PCIV to discover index to match. This is system dependent and slot dependent.

Index = you may instead use index to specify the index of the card found. Zero indicates first card, one indicates second card, etc.

The same address information may be used from the OS-9 boot menu to access additional SCSI controllers, e.g.:

: hs port=0xff000100 id=3 ? Boot from second SCSI controller SCSI ID=3

Creating Driver-Specific Versions

By default, the Symbios scsi8xx driver will look for any Symbios SCSI card based on table usage. You may however re-compile the driver to only look for the card desired.

MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

PCI_DEV_ID=# -dSYMBIOS_DEVICE_ID=0xf

Remove the # and specify the ID required.

Driver name: scsi8xx

Rom driver name: ncr8xx

Diamond FirePort20 and FirePort40

Benefits

- Wide support
- Ultra FAST20 support
- SCRIPTS RAM support (able to run scripts from on chip RAM) (1)
- Large FIFO enabled
- Increased burst rates to 128 where supported
- Special PCI cache features enabled (2)
- PCI IO Mode selectable (PCI I/O or PCI Memory) (3)

Additional Notes

1. The SCRIPTS RAM support is currently only available on OS-9, x86 based systems. Requires non translation of PCI memory. To use SCRIPTS RAM support include the "-dSCRIPTS_RAM" in the compile line when making the driver.
2. Instruction prefetch is not enabled by default. Maximum burst rate and large fifo's are enabled.
3. By default the Microware Symbios driver will use the PCI I/O model. To speed up transfers especially on x86 platforms the memory module may be used. In the PCI memory mode no in/out instructions are used. For the x86 platform this removes the CPU related waits added by the use of "inc", "outc" etc... If you would like to run the driver in PCI Memory mode the driver may be recompiled with the "-dPCI_IO_MAPPED" flag removed.

```
MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

IO_MAPPED=-dPCI_IO_MAPPED
```

To use memory model change to:

```
IO_MAPPED=# -dPCI_IO_MAPPED
```

The default has changed to IO_MAPPED for x86 due to problems on PCAT-based motherboards.

Prior to this release the following Symbios devices were supported:

Number of devices supported (2)

```
DEVICEWIDEULTRA1 ULTRA2 FIFO_SIZEBURST
-------------------------------------------------------------
Symbios 53c810N/AN/AN/A 6416
Symbios 53c825NoN/AN/A  8816
```

This release adds the following:

Number of devices supported (12)

```
DEVICEWIDEULTRA1 ULTRA2 FIFO_SIZEBURST
-------------------------------------------------------------
Symbios 53c810N/AN/AN/A 6416
Symbios 53c810APN/AN/AN/A6416 (1)
Symbios 53c815N/AN/AN/A 6416 (1)
Symbios 53c820YesN/AN/A 8816 (1)
Symbios 53c825YesN/AN/A 8816
Symbios 53c825AYesN/AN/A536128

Symbios 53c875YesYESN/A 536128
Diamond FirePort20YesN/AN/A536128 (825A)
Diamond FirePort40YesYESN/A536128 (875)
Symbios 53c860YesYESN/A 536128 (1)
Symbios 53c885YesYESN/A 536128 (1)
Symbios 53c895YesYESYES 536128 (1,2)
Symbios 53c896YesYESYES 536128 (1,2)
```

1. Support is included but untested.
2. Support for 895 and 896 is only available with out ULTRA support. The 160Mhz clock will be enabled on a future release. Note the 895 and 896 have not been tested.

[Symbios 53C810]

[Symbios 53C810A]

Device supports burst op code fetch

Device supports instruction prefetch

Device supports Cache Line Size and Cache Commands
[Symbios 53C810ALV] * same as 810
Device supports burst op code fetch
Device supports instruction prefetch
Device supports Cache Line Size and Cache Commands
[Symbios 53C815]
Device supports burst op code fetch
[Symbios 53C825]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
[Symbios 53C825A]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
[Symbios 53C860]
Device supports burst op code fetch
Device supports instruction prefetch
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers
[Symbios 53C875]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands
Device supports Fast-20 transfers
[Symbios 53C885]
Device supports Wide SCSI data transfers
Device supports burst op code fetch
Device supports instruction prefetch
Device has Scripts RAM
Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers

Device supports Clock Doubler

[Symbios 53C895]

Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers (Not supported yet)

Device supports Clock Doubler (Not supported yet)

Device supports Fast-40 transfers (Not supported yet)

[Symbios 53C896]

Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers (Not supported yet)

Device supports Clock Doubler (Not supported yet)

Device supports Fast-40 transfers (Not supported yet)

Using Ultra Fast20 and Wide support.

Controller Dependency

For FAST20 support the controller must support FAST20.

Device Descriptors

To use a device with disconnect, wide, synchronous data transfer, and FAST20 Ultra the following should be added to the device descriptor entry in "systype.h". Be sure to re-make the descriptors.

```
#define SCSIOPTS SCSI_ATN|SCSI_SYNC|SCSI_WIDE|SCSI_ULTRA
```

Optionally you may use EditMod to change the SCSIOPTS field. For SYNC and ATN the SCSIOPTS value is "5".

Using Multiple SCSI Controllers

It is possible to use multiple SCSI controllers with the Symbios family of controllers.

The port address is used to specify the card to use.

PortAddress format.

[0xff] [device] [index] [SCSI_ID]

device = device number. Use PCIV to discover index to match. This is system dependent and slot dependent.

Index = you may instead use index to specify the index of the card found. Zero indicates first card, one indicates second card, and so on.

The same address information may be used from the OS-9 boot menu to access additional SCSI controllers, e.g.:

: hs port=0xff000100 id=3 ? Boot from second SCSI controller SCSI ID=3

Creating Driver-Specific Versions

By default, the Symbios scsi8xx driver will look for any Symbios SCSI card based on table usage. You may however re-compile the driver to only look for the card desired.

`MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile`

PCI_DEV_ID=# -dSYMBIOS_DEVICE_ID=0xf

Remove the # and specify the ID required.

Driver name: scsi8xx

Rom driver name: ncr8xx

Adaptec 1540/1542 ISA

Support for Adaptec 1540 series is provided, this includes 1540, 1542 and 1542CP. The driver probes the DMA channel used, but the port address and interrupt are fixed. If the vector does not match the card, a Bad Mode error is returned. You may set up the descriptor to use vector zero, which forces the driver to use what the card reports.

```
#define                 BASE_AHA15400x00000330
#define                 VECT_AHA15400x4b
```

Driver name: aha1540

Rom driver name: ll1540

Adaptec 2940, 2940U and 2940UW

Support for Adaptec PCI series AHA2940, 2940U and 2940UW is provided. Only one SCSI controller of this type is allowed.

Driver name: aic7870

Rom driver name: ll7870

SCSI HARD - RBF Descriptors

/hs<id>fmtid= SCSI ID (1-f) - Entire disk

/hs<id><part>fmtid= SCSI ID (1-f) part= partition

/hs<id><part>id= SCSI ID (1-f) part= partition

SCSI HARD - PCF Descriptors

/mhs<id><part>id= SCSI ID (1-f) part= partition

SCSI FLOPPY - RBF Descriptors

d<id>_3.d0id= SCSI ID (1-f) mapped as drive d0

SCSI FLOPPY - PCF Descriptors

md<id>_3.d0id= SCSI ID (1-f) mapped as drive md0

SCSI TAPE Descriptors

/mt<id>id= SCSI ID (1-f)

SCSI CDROM Descriptors

/cd0SCSI ID is set to 5

When using the Wizard, the descriptors for SCSI are automatically included or created as needed for the SCSI controller selected. You may also access the descriptors in the MWOS directory structure:

`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/SCSI8XX`

`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/AHA1540`

`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBJS/DESC/AIC7870`

System Devices

Real Time Clock

Real-time clock (RTC) devices with battery backup enable the system clock to be set without operator intervention. The bootfile options dialog in the Configuration Wizard may be used to include one of two possible real-time clock drivers.

The local time driver assumes that the time stored in the RTC device is local time. This option maintains compatibility when another O.S. is installed on the same machine.

The GMT driver assumes that the time stored in the RTC is Greenwich Mean Time.

The driver communicates with the OS-9 kernel using GMT, with the System Time Zone field in the init module converting between GMT and local time. Refer to the Configuration Wizard Init Options dialog for information on setting the system time zone.

Additional Devices

PPP and SLIP

You can use the Wizard to configure and use both PPP and SLIP. You may select any or all of Ethernet, PPP, or SLIP. When using PPP or SLIP, the SPF options must be enabled. You can do this from the SPF/Options tab by selecting either SLIP or PPP or both. When you do this, make sure SPF is checked when building the boot image. If Ethernet is not desired, select `None` for the Ethernet controller name.

PPP Setup

Set up PPP by completing the following steps:

Step 1. Edit the `<board>.ini` file in the following directory:

`MWOS\OS900\80386\PORTS\<BOARD>\BOOTS\INSTALL\INI`

Search for `ETHER_OPTION_1`.

Step 2. By default, the PPP setup will obtain the address from the server. If desired this may be changed.

ETHER_OPTION_1=ppp0 binding /ipcp0 iff_pointopoint

Step 3. Make sure PPP is selected in the SPF/Options tab.

Step 4. Go into the Wizard and select `Enable SoftStax`. Build the boot.

SLIP Setup

Set up SLIP by completing the following steps:

Step 1. Edit the `<board>.ini` in the following directory:

`MWOS\OS900\80386\PORTS\<BOARD>\BOOTS\INSTALL\INI`

Search for `ETHER_OPTION_0`.

Step 2. Setup SLIP as required.

ETHER_OPTION_0=slip0 address 10.0.0.1 destaddr 10.0.0.2 binding /spsl0

Step 3. Make sure SLIP is selected in the SPF/Options tab.

Step 4. Go into the Wizard and select `Enable SoftStax`. Build the boot.