Home

OS-9® for MP5 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.

Table of Contents

Chapter 1: Installing and Running OS-9® for PowerPC

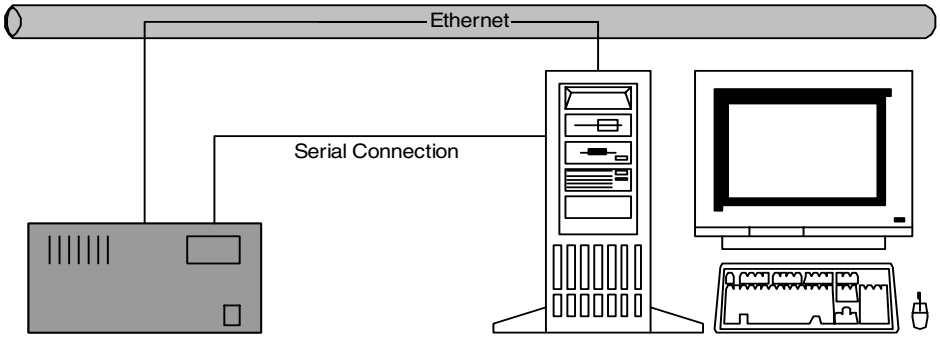
This chapter describes installing and configuring OS-9® on the MP5 target board. It includes the following sections:

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

Development Environment Overview

Figure 1-1 shows a typical development environment for the MP5 board. The components shown include the minimum required to enable OS-9 to run on the MP5 board.

Figure 1-1 MP5 Development Environment Overview



Requirements and Compatibility

Host Hardware Requirements (PC Compatible)

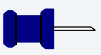
The host PC must have the following minimum hardware characteristics:

- Windows 95, 98, ME, 2000 (SP2 or higher), or NT (SP4 or higher)
- 300-400 MB of free disk space
- An ethernet card
- 16MB of RAM (32MB is recommended)
- one free serial port and one free parallel port

Host Software Requirements (PC Compatible)

The host PC must have the following applications:

- EBDS Debugger application for downloading the OS-9 ROM image to the target
- a terminal emulation program



Note

The examples in this document use the terminal emulation program, Hyperterminal, which ships with all Windows operating systems.

Target Hardware Requirements

Your target board requires the following hardware:

- a power supply (running at 12 V)
- an EBDS Lite Debugger parallel cable
- an RS-232 null modem serial cable

Connecting the Target to the Host

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

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

Step 3. Connect one end of the serial cable to the COM1 port on the target. On the MP5, COM1 is labeled **Ser. 19200** on the board. Connect the other end of the serial cable to the desired communication (COM) port on the host system.

Step 4. Connect the EBDS parallel cable to the target for ROM transfer. Connect the other end of the EBDS Lite cable to the appropriate connector.

Step 5. Apply power to the board and start the `EBDS Debugger`. You should now be able to access the registers for the board.

Building the OS-9 ROM Image

Overview

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

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

Coreboot

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

Bootfile

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

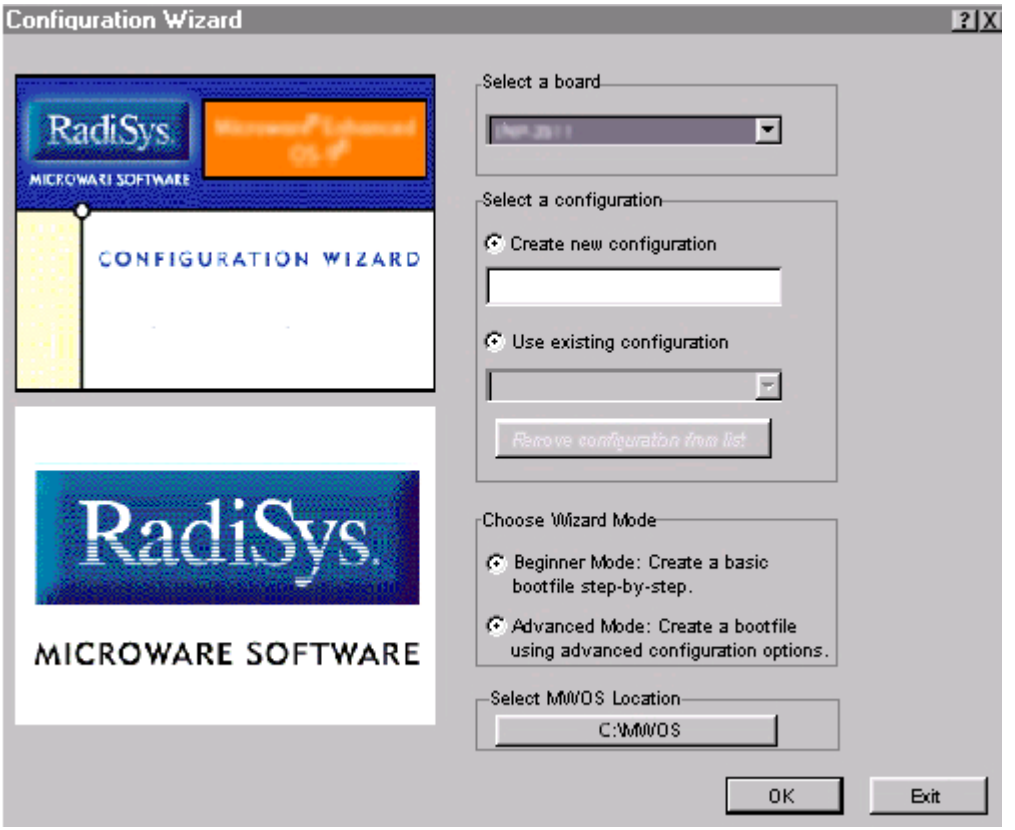
Microware provides a Configuration Wizard to create a coreboot image, a bootfile image, or an entire OS-9 ROM image. The Wizard can also be used to modify an existing image. The Configuration Wizard is automatically installed on your host PC during the OS-9 installation process.

Starting the Configuration Wizard

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

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

Figure 1-2 Configuration Wizard Opening Screen

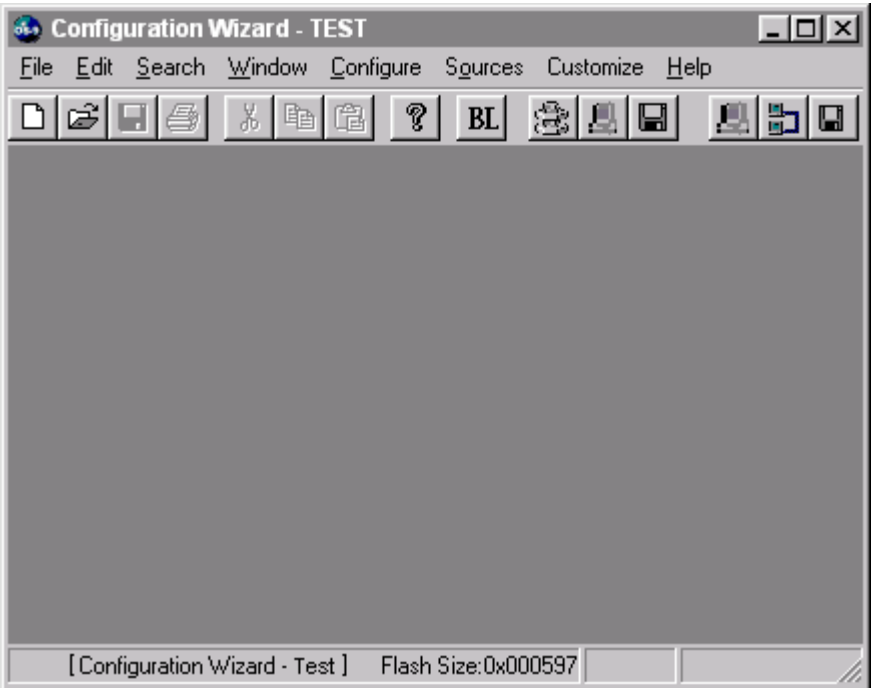


Step 2. Select your target board from the **Select a board** pull-down menu.

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

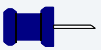
Step 4. Select the `Advanced Mode` radio button from the **Choose Wizard Mode** field and click `OK`**.** The Wizard's main window is displayed. This is the dialog from which you will proceed to build your image. An example is shown in **Figure 1-3**.

Figure 1-3 Configuration Wizard Main Window



Creating the ROM Image

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



Note

The following section provides an example of an OS-9 ROM image successfully built on a host PC and transferred to the MP5 target board. You may have to modify your selections depending on your application.

Select System Type

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

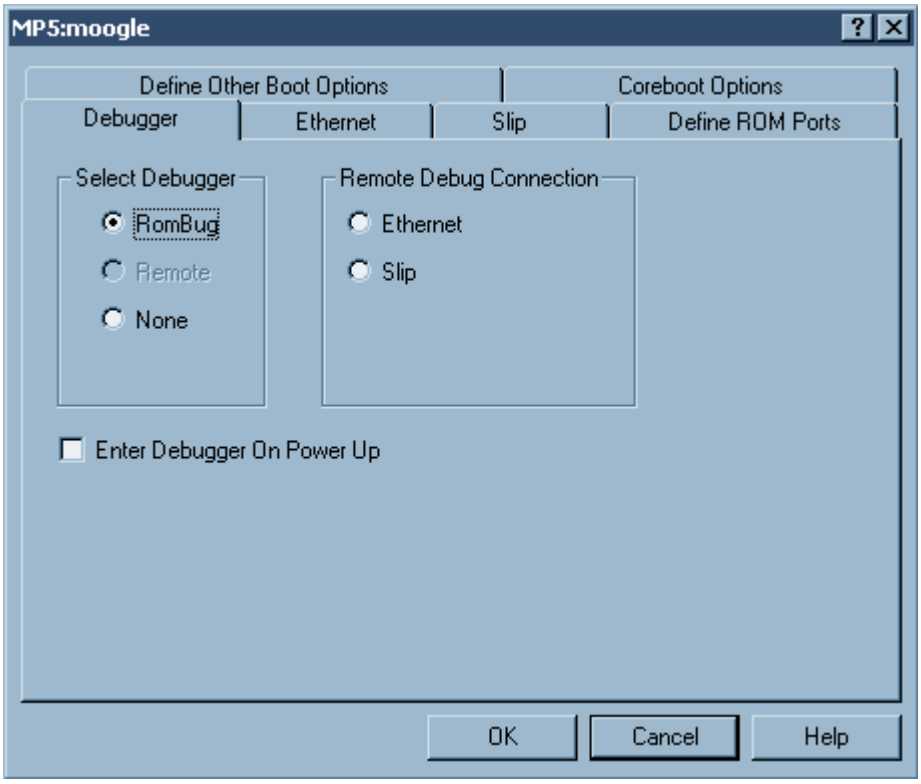
For the MP5 target board, you can bypass this option and use the default settings.

Configure Coreboot Options

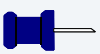
Step 1. From the Main Configuration window, select `Configure` -> `Coreboot` -> `Main configuration`.

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

Figure 1-4 Coreboot Configuration - Debugger Tab



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

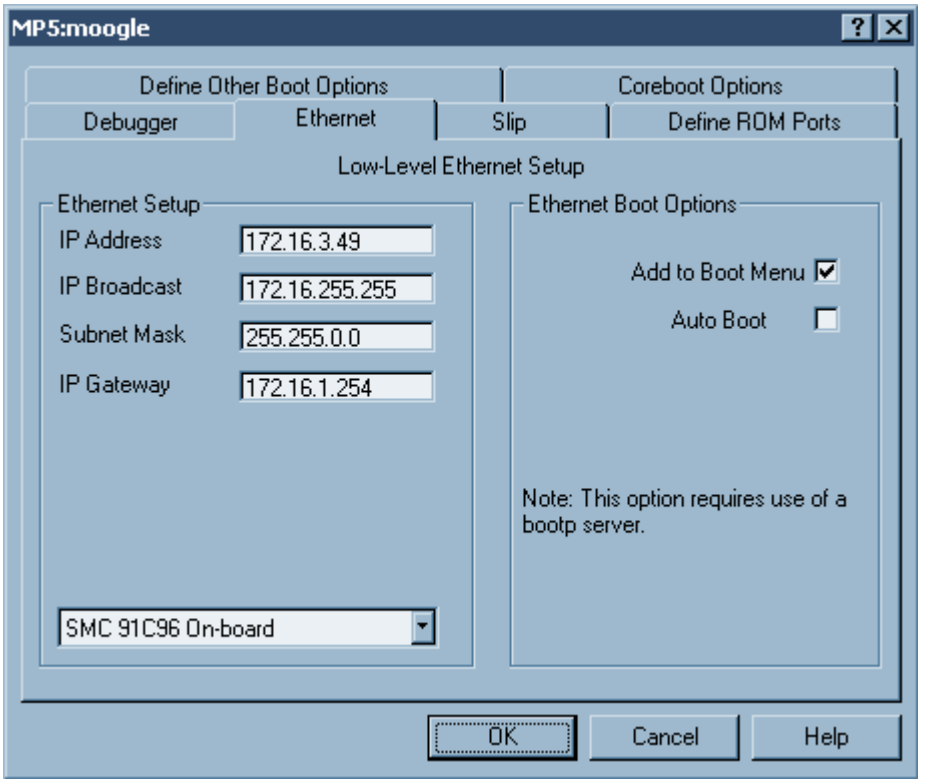


Note

To perform system state debugging, select `Ethernet` under **Remote Debug Connection**. If you set Ethernet as the method for system state debugging, you will not be able to perform user state debugging via Ethernet. In addition, to perform system state debugging, you must also set the parameters in the **Ethernet** tab of the coreboot main configuration window.

Step 4. Select the `Ethernet` tab. The following window is displayed:

Figure 1-5 Coreboot Configuration - Ethernet Tab



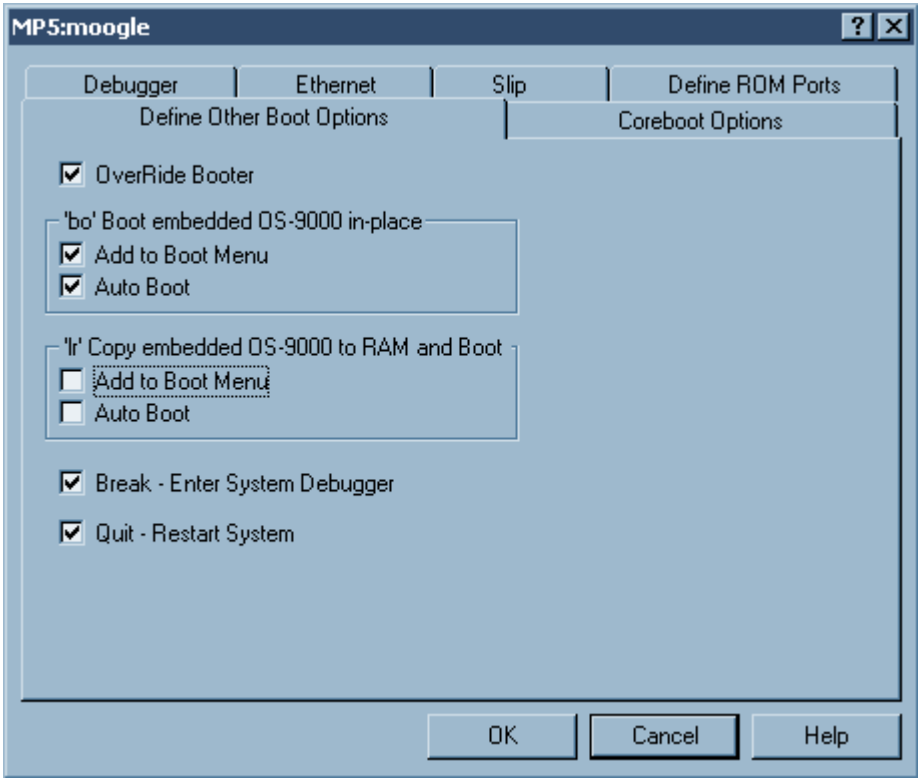
Select `Add to Boot Menu` under **Ethernet Boot Options** to include the TFTP boot capability.

WARNING

Due to ROM size limitation, you are not able to put both ROMBug and low-level ethernet driver together on the same coreboot file.

Step 5. Select the `Define Other Boot Options` tab. The following window is displayed:

Figure 1-6 Coreboot Configuration - Define Other Boot Options Tab



Step 6. Select `Break - Enter System Debugger`.

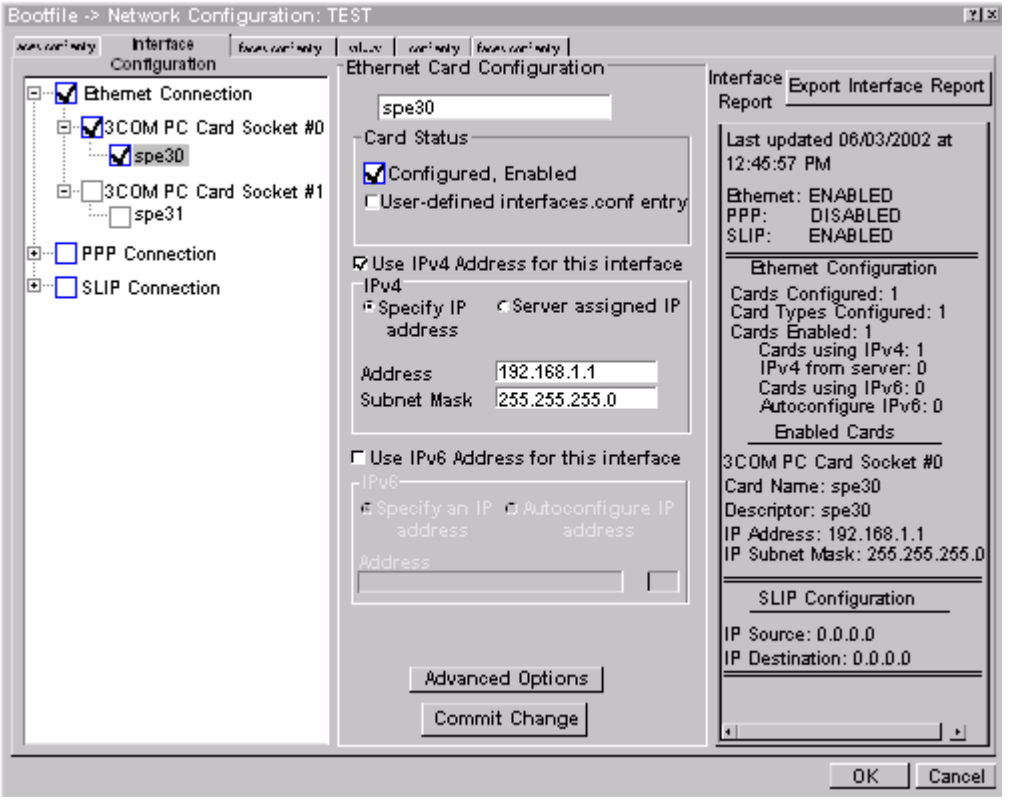
Step 7. Click `OK` and return to the Main Configuration window.

Configure Bootfile Options

In configuring the bootfile options, it is necessary to configure the network and disk settings. To modify these settings, complete the following steps:

Step 1. From the **Network Configuration** dialog, select the `Interface Configuration` tab. From here you can select and enable the interface. For example, you can select the appropriate Ethernet card from the list of options on the left and specify whether you would like to enable IPv4 or IPv6 addressing. **Figure 1-7** shows an example of the **Interface Configuration** tab.

Figure 1-7 Bootfile -> Network Configuration -> Interface Configuration



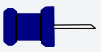
For More Information

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

For More Information

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

Step 2. Once you have made your settings in the **Network Configuration** dialog, click `OK`.



Note

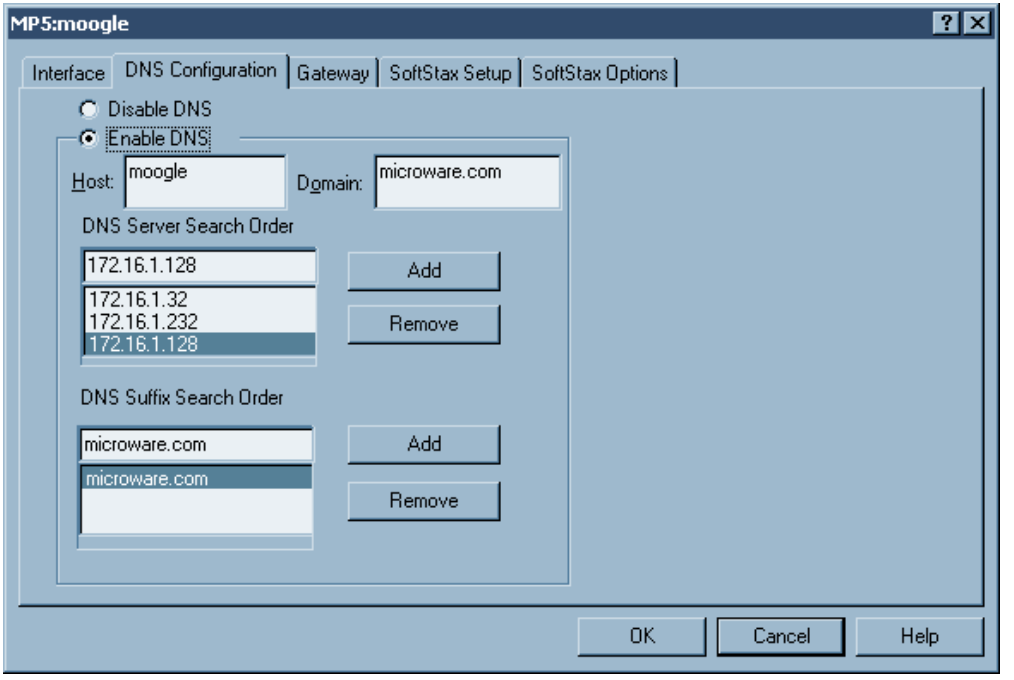
The addresses shown are for demonstration only. Contact your network administrator to obtain your IP Setup information. The MP5 board does not have a hardware assigned MAC address.

Step 3. To use the target board across a network, you must enable the Ethernet network settings. Click the `Specify an IP Address` button, then complete the following information:

- enter your IP address
- enter your broadcast address
- enter the subnet mask
- set the pulldown menu at the bottom of the window to `SMC 91C96 On-board`

Step 4. Select the **DNS Configuration** tab. The following window is displayed. More than one DNS server can be added in this dialog box.

Figure 1-8 Bootfile Configuration - DNS Configuration Tab



If your network does not use DNS, click `Disable DNS`, and move to the **Gateway** tab. If you have DNS available, click `Enable DNS` and type your host name and domain.

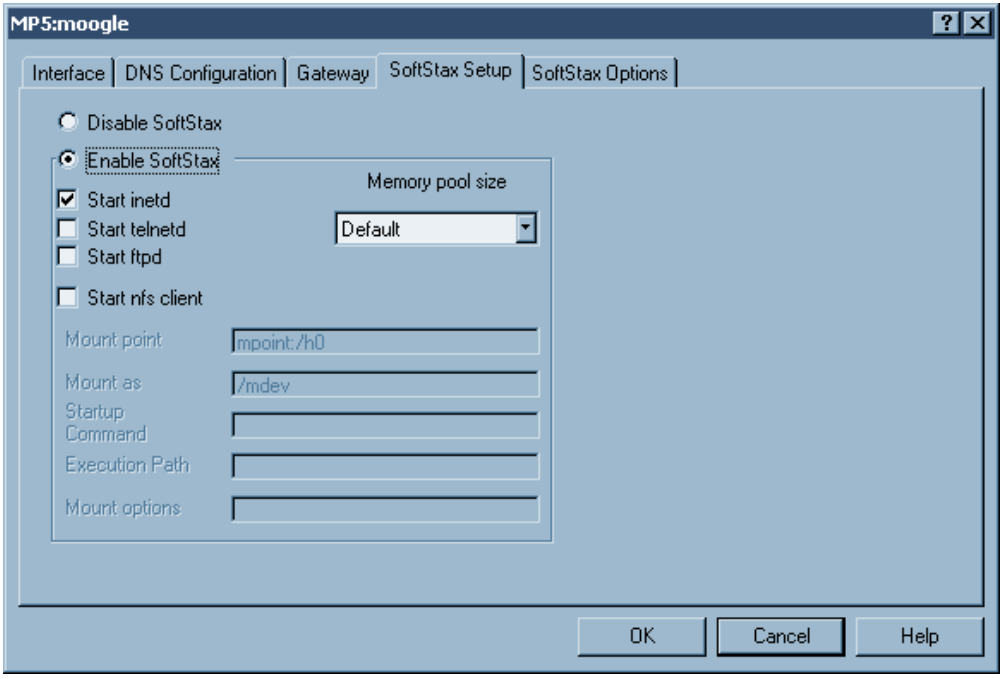
Note

You add DNS IP addresses by clicking on the box directly under `DNS Server Search Order` and typing the IP address. Click the `Add` button when complete. More than one DNS server can be added by repeating these steps.

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

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

Figure 1-9 Bootfile Configuration - SoftStax Setup Tab



The options above represent daemons that can be automatically started if you want to FTP, Telnet or both from a PC to the OS-9 target. Clicking on Start NFS Client enables you to remote mount the target. For this demonstration, you will telnet to the target and establish a Sender and a Receiver window.

Step 7. Click Enable SoftStax.

Step 8. Click Start inetd.

Step 9. Click OK.

Step 10. Select the SoftStax Options tab.

The SoftStax Options tab enables you to include networking utilities in the ROM image. By default, none of the utilities are included because of ROM size limitation.

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

Step 12. From the Main Configuration window, select Configure -> Bootfile -> Disk Configuration. The **Disk Configuration** window appears.

The Disk Configuration window contains the following tabs:

- RAM Disk

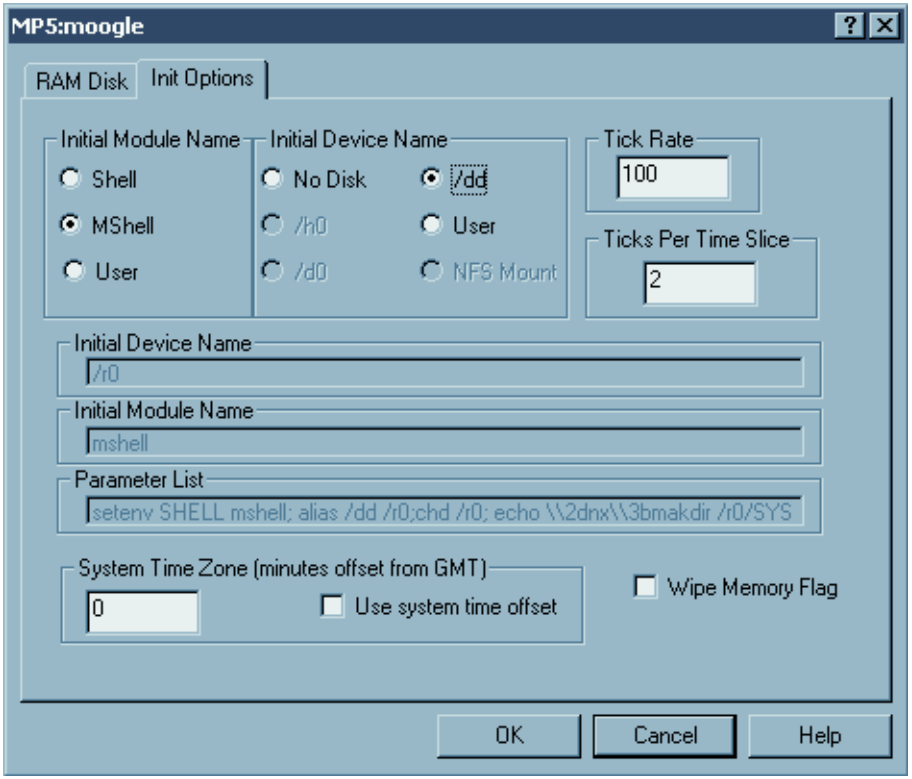
  This tab enables you to create a RAM disk of any size for loading modules onto the target.

- Init Options

  This tab sets the configuration for OS-9 to initialize itself on the target.

Step 13. Select the `Init Options` tab. The following window is displayed:

Figure 1-10 Bootfile Configuration - Init Options Tab



- Select the `MShell` option for the initial module name. This causes OS-9 to start a console shell usable from your terminal window. Select `/dd` in the `Initial Device Name` section.
- The tick rate is `100` and ticks per timeslice is set to `2`. If you look at the `Parameter List` box, you see the commands that OS-9 executes upon system start-up.

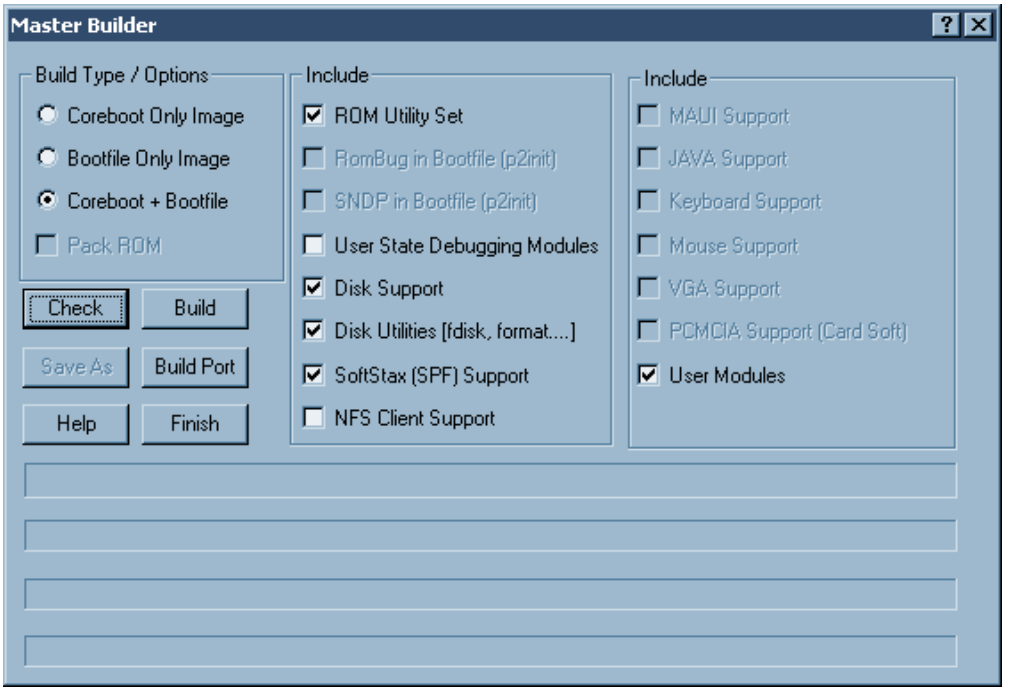
Step 14. Click `OK` to return to the Main Configuration window.

Building the Image

Complete the following steps to build the target board image.

Step 1. From the Main Configuration window, select `Configure` -> `Build Image`. The Master Builder window appears. (**Figure 1-11** shows the Master Builder window configuration.)

Figure 1-11 Master Builder Window



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

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

Step 4. Click `Build`. It should display progress information and show the statistics of the image just created. The `rom` and `rom2` file is created in the following directory:

```
MWOS/OS9000/555/PORTS/MP5/BOOTS/INSTALL/PORTBOOT
```

Note

Since the Flash region on the MP5 is not contiguous, the the OS-9 ROM image needs to be split into `rom` and `rom2`.

Transferring the ROM Image to the Target

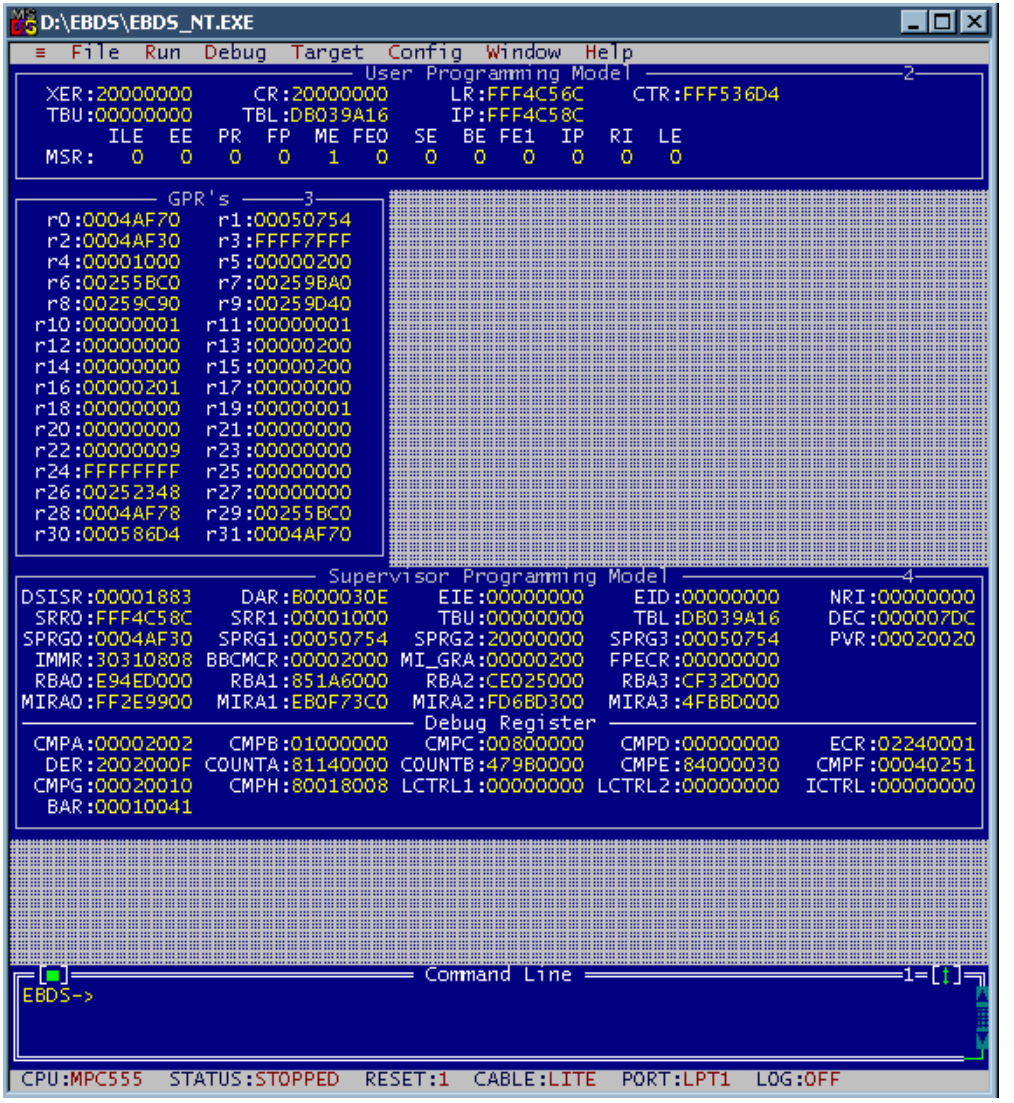
Configure the EBDS Debugger

EBDS Debugger is the debugging utility that must be installed on your PC host. It is used to transfer the ROM image from your host to the target system. Perform the following steps to install and configure the EBDS Debugger:

Step 1. Copy the EBDS software by extracting the `ebds.zip` file to a directory (The default is `C:\`). Run `reg_ebdsdrv` to install it, then restart the PC to incorporate the changes.

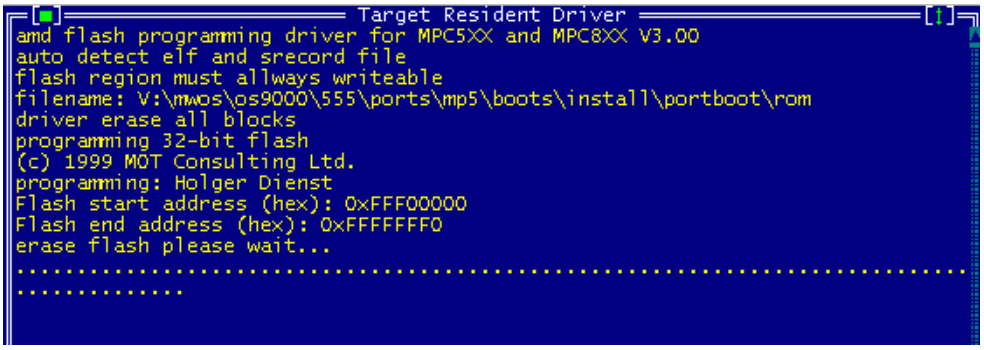
Step 2. Connect the target board to the host PC using the EBDS Lite Debugger parallel cable. Run the EBDS Debugger by typing `ebds_nt`. You should now see the list of hardware registers of the MP5.

Figure 1-12 EBDS Debugger Main Window



Step 3. On the Command Line, type `do oakfl1.do` to flash the first image to ROM. When given the flash prompt, type `0xfff00000` as the start address and `0xffffffff0` as the end address. The flash process starts from here.

Figure 1-13 EBDS Debugger Flash Utility Window



Step 4. After the flash process is complete, return to the command line. Type `do oakfl2.do` to flash the second image to ROM. When given the prompt, type `0xffe00000` as the start address and `0xffefffff0` as the end address. The second image will then be flashed to ROM.

WARNING

Both images must be flashed to the ROM, otherwise, the target will not run properly.

Step 5. Disconnect the debugger cable and reboot the target.

Step 6. Use Hyperterminal to communicate with target.

Creating a Startup File

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

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

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

`MWOS/OS9000/SRC/SYS`

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

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

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

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

```
cudo -cdo password
```

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

For More Information

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

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

Example Startup File

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

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

For More Information

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

Optional Procedures

Preliminary Testing

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

Step 1. Type `mdir` at the prompt. This displays all of the modules in memory.

Step 2. Type `procs` at the prompt. This displays the processes currently running in the system.

Step 3. Test the networking on your system by selecting a host on the Ethernet network and running the `ping` utility. The following display shows a successful `ping` to a machine called `visor`:

```
$ ping visor
PING visor.microware.com (172.16.4.162): 56 data bytes
64 bytes from 172.16.2.51: ttl=128 time=0 ms
```

Step 4. Test `telnet`.

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

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

OS-9/68K V3.0.3 Delta VME177 - 68060 00/01/31 17:00:00
User name?: kupo
Password:
Process #101 logged on 01/03/18 18:00:00
Welcome!
************************************************************
*      WELCOME TO DELTA - THE :OS-9 68K: MACHINE *
```

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

From the Windows **Start** menu, select `Run` and type the following command:

`telnet <hostname>`

Step 6. Click `OK`. A telnet window should display with a `$` prompt. Type `mdir` from the prompt. You should see the same module listing as you see on the serial console port.

Chapter 2: Board Specific Reference

This chapter contains information that is specific to the MP5 reference boards. It contains the following sections:

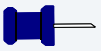
- **Boot Menu Options**
- **Vector Descriptions for PowerPC™ 555**
- **PowerPC™ Registers Passed to a New Process**

For More Information

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

Boot Menu Options

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



Note

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

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

```
OS-9 Bootstrap for the PowerPC(tm)

Now trying to Override autobooters.

BOOTING PROCEDURES AVAILABLE ------- <INPUT>

Boot Embedded OS-9 in-place -------- <bo>
Copy Embedded OS-9 to RAM and Boot - <lr>
Boot over the Ethernet ------------- <eb>
Boot via. Kermit Download ---------- <ker>
Enter System Debugger -------------- <break>
Restart the System ----------------- <q>

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

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

Table 2-1 Supported Boot Methods

Type of Boot	Description
Boot from the Ethernet	boot from a BootP server (`eb`)
Boot embedded OS-9 in-place	boot OS-9 from Flash ROM (`bo`)
Copy embedded OS-9 to RAM and Boot	copy OS-9 from ROM (if stored there) to RAM and boot (`lr`)

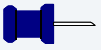
Vector Descriptions for PowerPC™ 555

Table 2-2 Vector Descriptions for PowerPC™ 555

Vector Number	Assignment
0	Reserved
1	System reset
2	Machine check
3	Reserved
4	Reserved
5	External interrupt
6	Alignment
7	Program
8	Floating-point unavailable
9	Decrementer
10	Reserved
11	Reserved
12	System call
13	Trace
14	Floating Point Assist

Table 2-2 Vector Descriptions for PowerPC™ 555

Vector Number	Assignment
15	Reserved
16	Software emulation error
17	Reserved
18	Reserved
19	Instruction protection error
20	Data protection error
28	Data breakpoint
29	Instruction breakpoint
30	Maskable external breakpoint
31	Non-maskable external breakpoint



Note

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

Error Exceptions: Vectors 2, 6 and 7

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

A user process may use the `F_STRAP` system call to install an exception handler to catch the errors and recover from the exceptional condition. When a recoverable exception occurs, the process' exception handler installed with the `F_STRAP` system call is executed with a pointer to the process' normal static data and the current stack pointer.

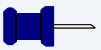
In addition, the process' exception handler will receive - as parameters - the vector number of the error, the program instruction counter of where the error occurred, and the fault address of the error if applicable. The exception handler must decide whether and where to continue execution. Programs written in the C language may use the `setjmp` and `longjmp` library routines to properly recover from the erroneous condition.

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

Vectored Interrupts: Vector 5

In general, the PowerPC processor family uses a single interrupt vector for all external interrupts. However, most systems supporting the PowerPC family use additional external logic to support more powerful nested interrupt facilities. Hence, the vector numbers used by OS-9 device drivers are usually logical vectors outside of the range of the hardware vectors listed above.

The device drivers install their interrupt service routines via the F_IRQ system call on the logical vector, and the kernel's dispatch code uses the external logic vector to identify the source of the interrupt and call the associated interrupt service routine. Interrupt service routines are executed in system-state without an associated current process.



Note

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

User Trap Handlers: Vector 7

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

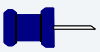
System Calls: Vector 12

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

PowerPC™ Registers Passed to a New Process

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

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



Note

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

Appendix A: Board Specific Modules

This chapter contains an overview of the board-specific low-level system modules and the high-level system modules. Each listing includes a brief description. The following sections are included:

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

Low-Level System Modules

The following low-level system modules are tailored specifically for the MP5 target platforms. These modules can be found in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/ROM`

Configuration Modules

`cnfgdata`	Provides low-level configuration data including configuration of a serial console
`cnfgfunc`	Retrieves configuration parameters from the `cnfgdata` module
`commcnfg`	Retrieves the name of the low-level auxiliary communication port driver from the `cnfgdata` module
`conscnfg`	Retrieves the name of the low-level console driver from the `cnfgdata` module
`initext`	User-customizable system initialization module

Console Driver

`iosci55a`	Provides console services for the built-in SCI serial port

Ethernet Driver

`ll91c94`	Provides network driver services for the SMC91C96 on-board ethernet chip

System Modules

`portmenu`	Retrieves a list of configured booter names from the ROM `cnfgdata` module
`romcore`	Bootstrap code
`romstart`	Provides the ROMStart prefix code

Timer Module

`tbtimer`	Provides polling timer services using `tblo` and `tbhi` registers

Debugging Module

`usedebug`	Debugger configuration module

High-Level System Modules

The following OS-9 system modules are tailored specifically for MP5 series platforms. Unless otherwise specified, each module can be found in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS`

Interrupt Controllers

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

`siu5irq`	Provides interrupt acknowledge and dispatching support for the SIU interrupt controllers on the PowerPC 555 series platforms. This also maps the nested PIC interrupts 0-15 to OS-9 pseudo vectors 64-88 (0x40-0x58)

Ticker

`tk555`	Provides the system ticker through the SIU built-in timer

Abort Handler

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

Shared Libraries

`picsub` — Provides interrupt enable and disable routines to handle platform-specific interrupt controller issues for device drivers

This module is called by all drivers and **should** be included in your `bootfile`.

I/O and Disk Descriptors

`pipe` — Pipeman descriptor that provides a RAM-based FIFO, which can be used for process communication. It is located in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/DESC`

`r0` — RBF descriptor that provides access to a RAM disk. A variant called `r0.dd` is available if you want to call the RAM disk `/dd`. These descriptors are located in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/DESC/RAM`

Serial and Console Drivers

`sc555` — Provides support for the internal SCI serial module.

The descriptors for this driver, named `t1` and `term_t1`, are located in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/DESC/SC555`

`scllio` — Provides support for the low-level polled serial driver.

The descriptors for this driver, named `term` and `vcons`, are located in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/DESC/SCLLIO`

Ethernet Driver

`sc91c94`

Provides support for SMC91C96 on-board ethernet chip.

The descriptor for this driver, named `spsmc0`, is located in the following directory:

`MWOS/OS9000/555/PORTS/MP5/CMDS/BOOTOBJS/SPF`

Common System Modules List

The following low-level system modules provide generic services for OS-9 modular ROM. They are located in the following directory:

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

Table 2-3 Common System Modules List

Module	Description
`bootsys`	Provides booter services
`console`	Provides high-level I/O hooks into low-level console serial driver.
`dbgentry`	Provides hooks to low-level debugger server
`dbgserv`	Debugger server module
`excption`	Service module
`fdc765*`	Provides PC-style floppy support
`fdman*`	Provides general booting services for RBF file systems
`flboot*`	Provides SCSI floppy disk booter
`flshcach`	Provides the cache flushing routine
`fsboot*`	Provides TEAC SCSI floppy disk booter
`hlproto`	Allows user-state debugging
`hsboot*`	Provides SCSI hard disk booter

Table 2-3 Common System Modules List (continued)

Module	Description
ide*	Provides target-specific standard IDE support, including PCMCIA ATA PC cards
iovcons*	Provides a virtual console driver that provides a Telnet daemon (telnetd)-like interface to the low-level system console
llbootp	Provides low-level BootP services
llip	Provides low-level (Internet Protocol) IP services
llkermit	Provides a Kermit protocol booter
llslip	Provides low-level Serial Line over IP (SLIP) services
lltcp	Provides low-level Transmission Control Protocol (TCP) services
lludp	Provides low-level User Datagram Protocol (UDP) services
notify	Coordinates the use of low-level I/O drivers in system and user-state debugging
override	Enables overriding of the auto booter If Spacebar is pressed within three seconds of booting the target, a boot menu is displayed. Otherwise, booting proceeds with the first auto booter.
parser	Parses key fields from the cnfgdata module and the user parameter fields

Table 2-3 Common System Modules List (continued)

Module	Description
pcman*	Provides booter support module providing general booting services for PCF file systems (PC FAT file systems)
protoman	Provide a protocol management services. This module provides the initial communication entry points into the protocol module stack.
restart	Restarts boot process
romboot	Locates the OS-9 bootfile in ROM or Flash part
rombreak	Enables break option from the boot menu
rombug	Debugger client module
scsiman*	Provides booter support module that provides general SCSI command protocol services
sndp	Provides System-state Network Debugging Protocol (SNDP) services This module acts as a debugging client on the target, invoking the services of dbgserv to perform debug tasks.
srecord*	Receives a Motorola S-record format file from the serial port and loads it into memory
swtimer*	Software timer
tsboot*	Provides SCSI TEAC tape drive booter
type41*	Primary partition type

Table 2-3 Common System Modules List (continued)

Module	Description
`vcons`	Provides console terminal pathlist
`vsboot`*	Provides SCSI tape booter

* Not applicable to the MP5 board