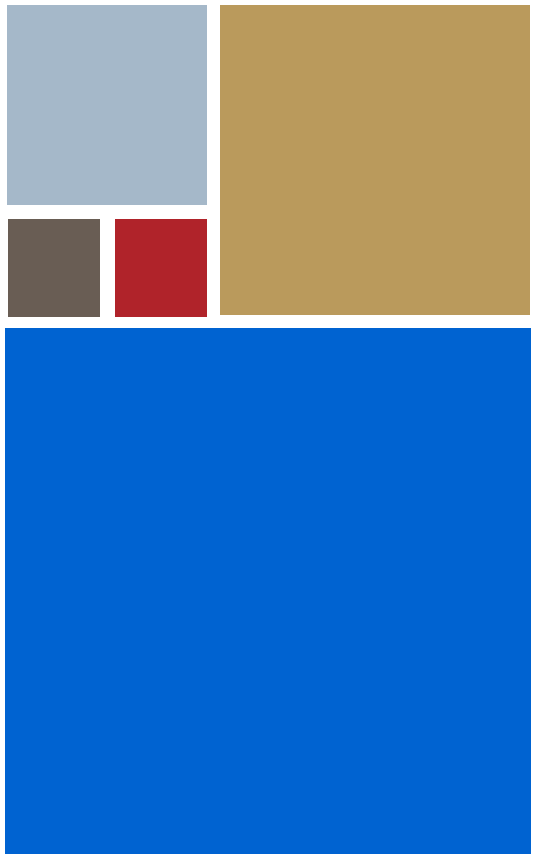


Home

OS-9® for the IBM 405GP 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 Configuring OS-9®

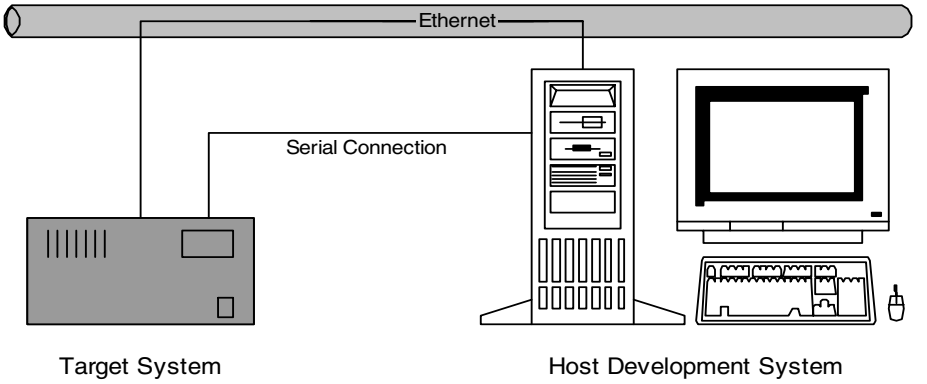
This chapter describes installing and configuring OS-9® on the IBM 405GP Evaluation Board target. It includes the following sections:

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

Development Environment Overview

Figure 1-1 shows a typical development environment for the IBM 405GP board. The components shown include the minimum required to enable OS-9 to run on PowerPC.

Figure 1-1 405GP Development Environment



Requirements and Compatibility

Host Hardware Requirements (PC Compatible)

Your host PC must meet the following minimum requirements:

- 300-400 MB of free disk space (an additional 235MB of free disk space is required to run PersonalJava™ for OS-9)
- an Ethernet network card
- 32MB of RAM (64MB recommended)
- one free serial port

Host Software Requirements (PC Compatible)

Your host PC must have the following applications:

- Windows operating system (95/98/NT/2000/ME are supported)
- a terminal emulation program (such as Hyperterminal, which is provided with Microsoft Windows products)
- a BOOTP server, not supplied by Microware

Target Hardware Requirements

Your IBM 405GP target system requires the following hardware:

- Chassis with IBM 405GP evaluation board and power supply
- RS-232 serial connectors
- VGA monitor, PS/2 keyboard, and PS/2 mouse (optional)
- TVIA IGS5050 PCI Display card (optional)

Target Hardware Setup

The following sections detail how to set up the target board.

Setting the Switches on the Target Board

This section describes any switch settings that must be made on the target board.

Note

Please refer to your ***405GP Reference Board Manual*** for information on hardware preparation and installation, operating instructions, and functional descriptions prior to installing and configuring OS-9 on your 405GP target board.

OS-9 requires the 405GP Reference Board to be strapped with the default settings.

Table 1-1 U52 DIP Switch Settings

Switch Number	Setting
1	On
2	Off
3	Off
4	On
5	On
6	Off
7	Off
8	On

Table 1-2 U53 Switch Settings

Switch Number	Setting
1	Off
2	On
3	Off
4	Off
5	On
6	On
7	On
8	On

Table 1-3 U54 Switch Settings

Switch Number	Setting
1	On
2	On
3	Off
4	On
5	On
6	Off
7	On
8	Off

Table 1-4 U79 Switch Settings

Switch Number	Setting
1	Off
2	Off
3	Off
4	On
5	On
6	On
7	On
8	On

Connecting the Target to the Host

The following sections detail how to connect the target machine to the host machine.

Connecting To the COM Port

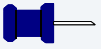
You need a terminal emulation program (such as Hyperterminal) and a serial cable to establish the connection between the host PC and the 405GP target machine.

Step 1. With the target system powered off, use the serial cable to connect the target's COM port to an unused RS-232 COM port on your host PC. You must also connect the target board and your host PC to a network to use bootp (network booting).

Step 2. On the Windows Desktop, select `Start -> Programs -> Accessories -> Hyperterminal`.

Step 3. Click the `Hyperterminal` icon and enter a name for your Hyperterminal session.

Step 4. Select an icon for the Hyperterminal session. A new icon is created with the name of your session associated with it. Click `OK`.



Note

The next time you want to establish the same session, follow the directions in Step 2 and select the icon you created in Step 3.

Step 5. From the **Phone Number** dialog, select **Connect Using** and then select the communications port to be used to connect to the target system. Click `OK`.

Step 6. In the **Port Settings** tab, enter the following settings:

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

Step 7. Click `OK`

Step 8. Go to the Hyperterminal menu and select `File -> Properties`. Click on the `Settings` tab and select the following:

```
Terminal Keys
Emulation = Auto Detect
Backscroll Buffer Lines = 500
```

Step 9. Click `OK`

Step 10. From the Hyperterminal window, select `Call -> Connect` from the pull-down menu to establish your terminal session with the target board. When you are connected, the bottom left of your Hyperterminal screen displays *connected*.

Step 11. Turn on the target board. A power-on banner and menu should appear on the display terminal connected to the board.

Ethernet Connection Only

The target system can also be configured with its own terminal, mouse, and keyboard attached. In this configuration, communication between the host and target is achieved through the Ethernet connection. **Figure 1-2** shows this configuration.

Figure 1-2 Basic 405GP Development System—Ethernet Only

Building the OS-9 Rom Image

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

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

Coreboot

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

Bootfile

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

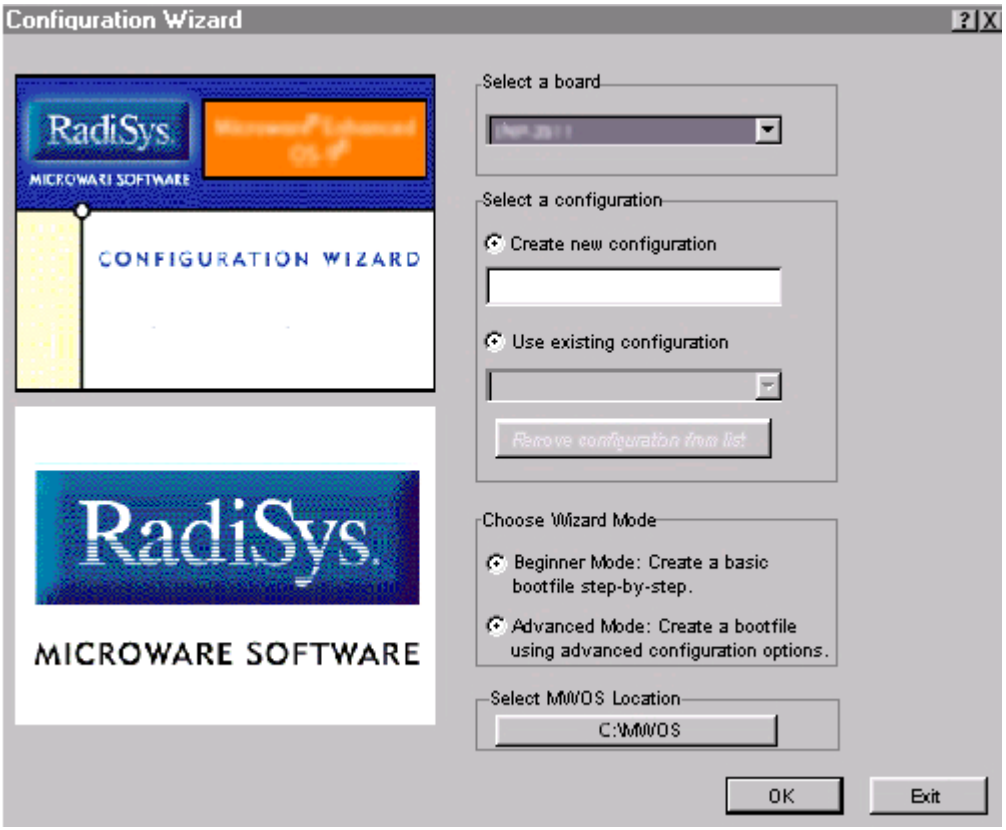
Microware provides a Configuration Wizard to create a coreboot image, a bootfile image, or an entire OS-9 ROM Image. The wizard can also be used to modify an existing image. The Configuration Wizard is automatically installed on your host PC during the OS-9 installation process.

Starting the Configuration Wizard

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

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

Figure 1-3 Configuration Wizard Opening Screen

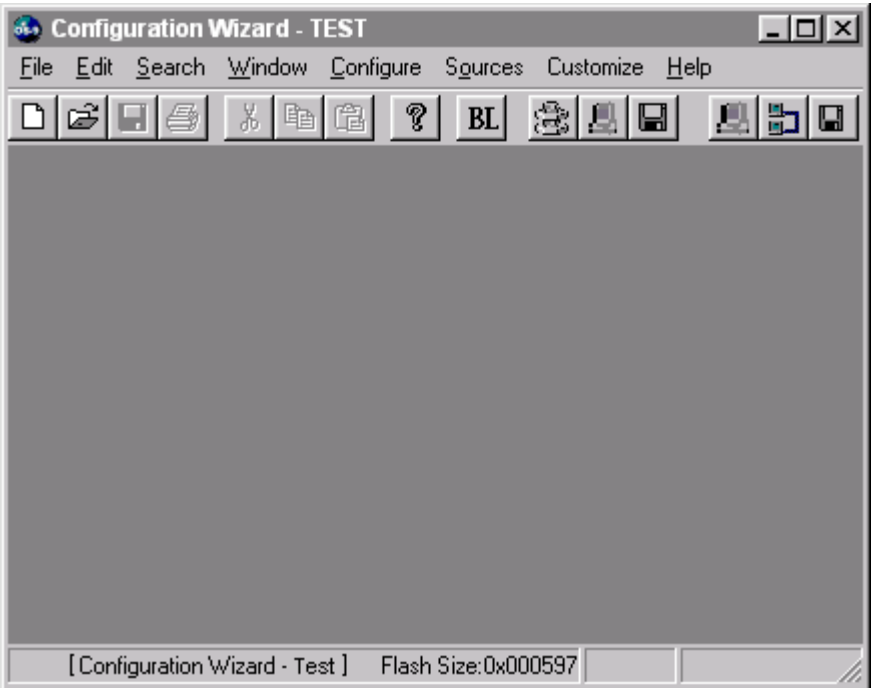


Step 2. Select your target board from the **Select a board** pull-down menu.

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 1-4**.

Figure 1-4 Configuration Wizard Main Window



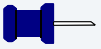
Creating and Configuring the ROM Image

For More Information

The ***OS-9 Device Descriptor and Configuration Module Reference*** manual included on your CD describes each of the OS-9 modules and the various ways that the software can be configured to meet your needs.

The ROM image consists of the coreboot image and the bootfile image. Together, these files comprise the OS-9 operating system.

The Configuration Wizard enables you to choose the contents of your OS-9 implementation. It also enables you to create individual coreboot and bootfile images, or combine them into a single file (the ROM image). The following sections describe how to use the Configuration Wizard to create and configure your OS-9 ROM image.



Note

This section provides an example of an OS-9 ROM image successfully built on a Host PC and transferred to a 405GP target board. You may have to modify your selections depending on your application.

Select System Type

From the Main Configuration window, select `Configure -> Sys -> Select System Type`. For the 405GP target board, use the configuration Wizard's default settings.

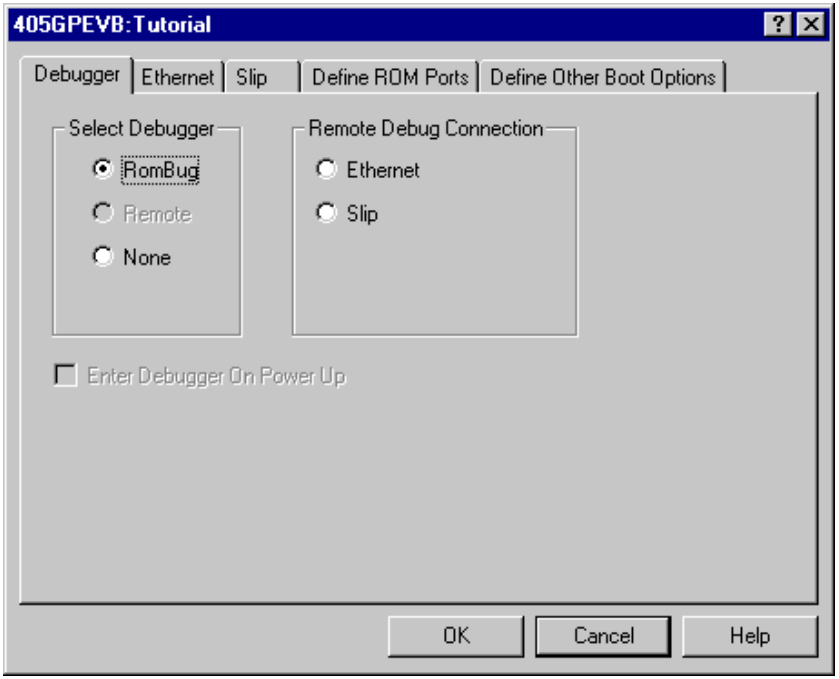
Configure Coreboot Options

To create a new `coreboot` image, use the Configuration Wizard to complete the following steps. Otherwise, continue to the **Configure System Options** section.

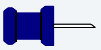
Step 1. From the **Main Configuration** window, select `Configure -> Coreboot -> Main configuration`.

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

Figure 1-5 Coreboot Configuration—Debugger Tab



Step 3. Under Select Debugger, select `RomBug`. This sets the debugging method to a console oriented debugger. Select `None` if you do not want to use a low-level debugger.



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 unless you use `hlproto`. Using `hlproto` is described in the ***Debugging OS-9 Projects*** chapter in the ***Using Hawk™*** manual.

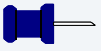
For system state debugging, you must also set the parameters in the Ethernet tab of the coreboot configuration.

Step 4. Select the **Ethernet** tab and configure the settings as appropriate.

For More Information

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

Step 5. Once you have made your settings in the **Network Configuration** dialog, click `OK`.

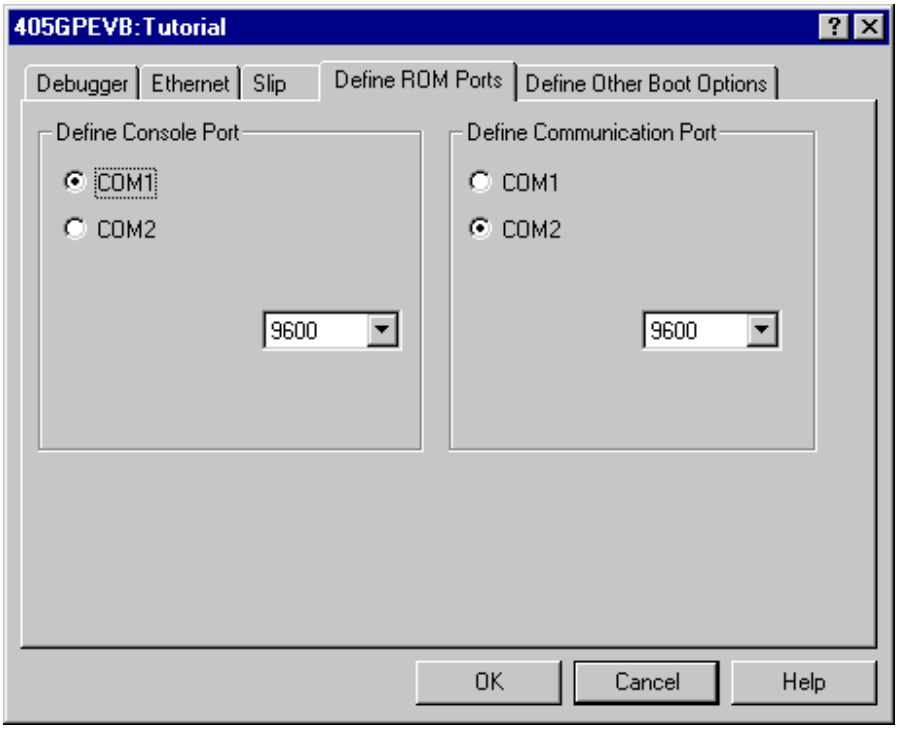


Note

Complete the Ethernet setup information only if you intend to boot your system over a network or if you plan to use system state debugging.

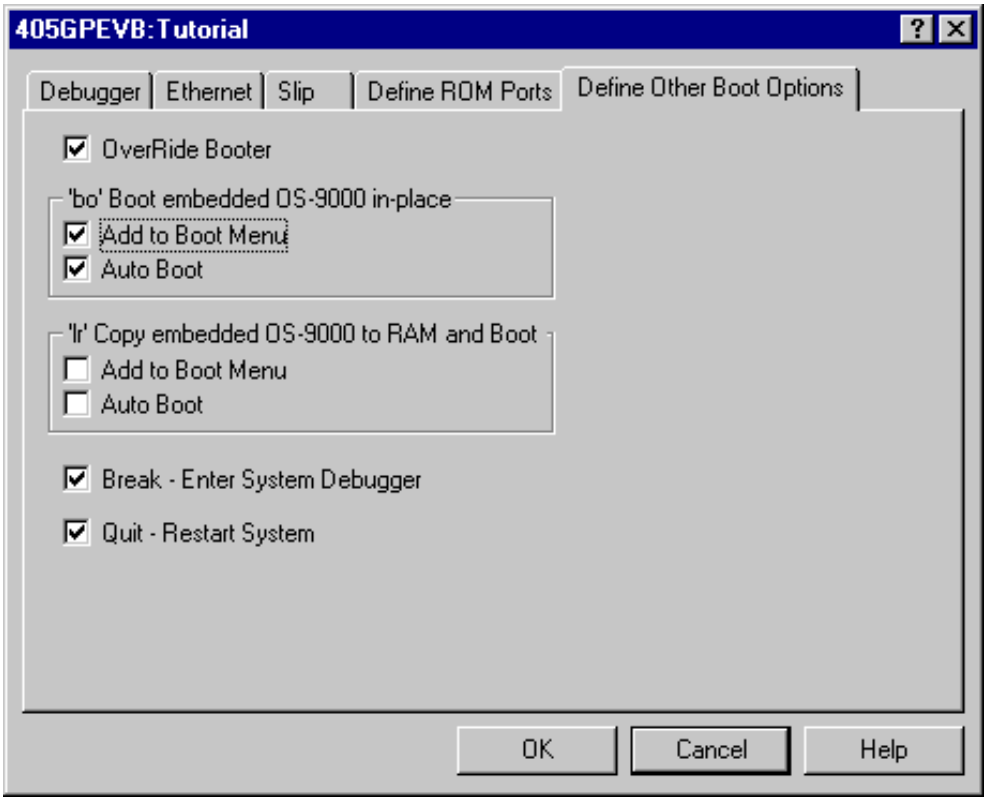
Step 6. Select the **Define ROM Ports** tab. The following window is displayed.

Figure 1-6 Coreboot Configuration—Define ROM Ports Tab



Step 7. Select the **Define Other Boot Options** tab. The following window is displayed.

Figure 1-7 Coreboot Configuration—Define Other Boot Options



Step 8. Select `Break-Enter System Debugger`.

Step 9. Click `OK` and return to the **Main Configuration** window.

Configure System Options

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

Network Configuration

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

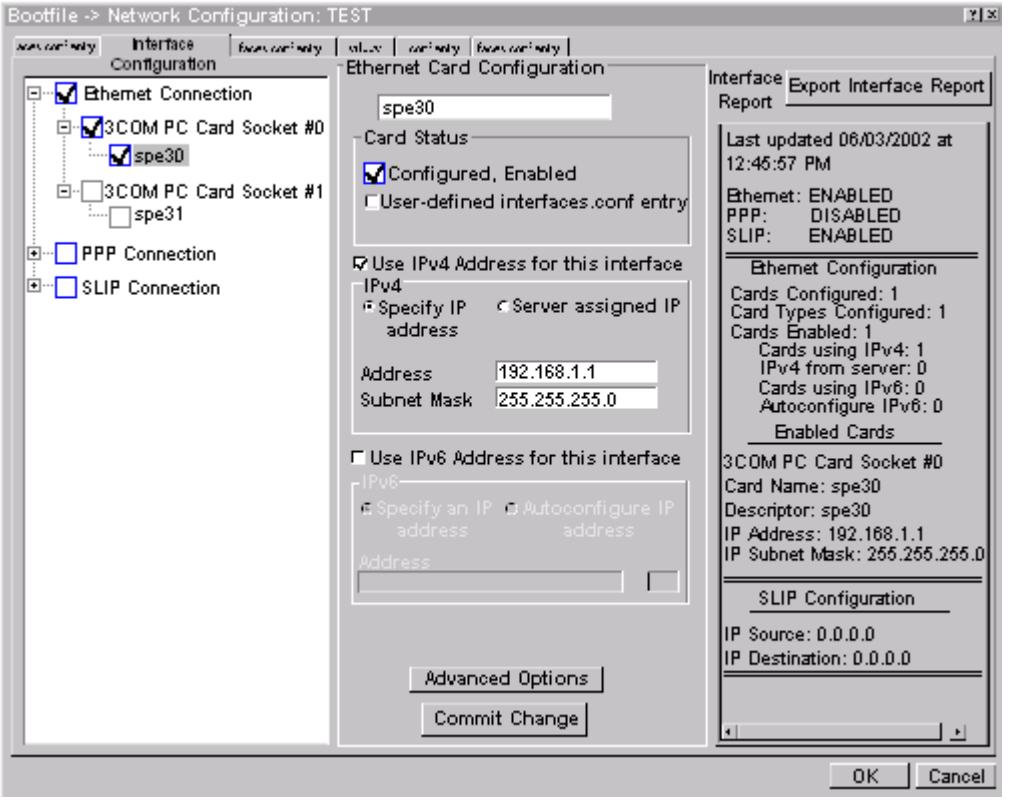
Note

The IP addresses shown in this example are for demonstration only. Contact your network administrator to obtain your IP Setup information.

To configure your network 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 1-8** shows an example of the **Interface Configuration** tab.

Figure 1-8 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.

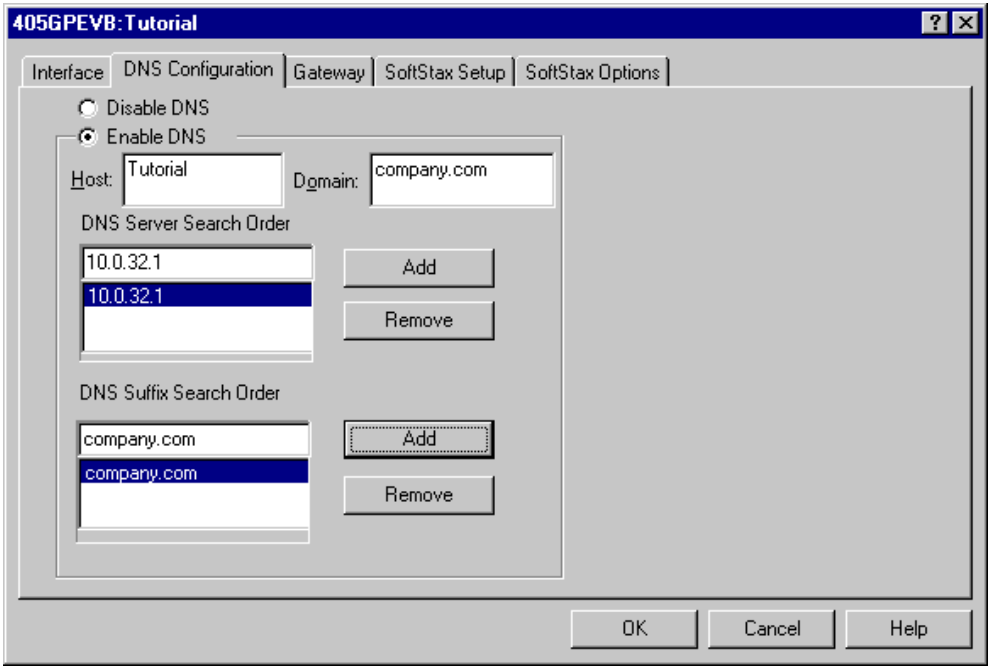
For More Information

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.

Step 3. Select the **DNS Configuration** tab. The following window is displayed. More than one DNS server can be added in this dialog box.

Figure 1-9 Bootfile Configuration—DNS Configuration Tab



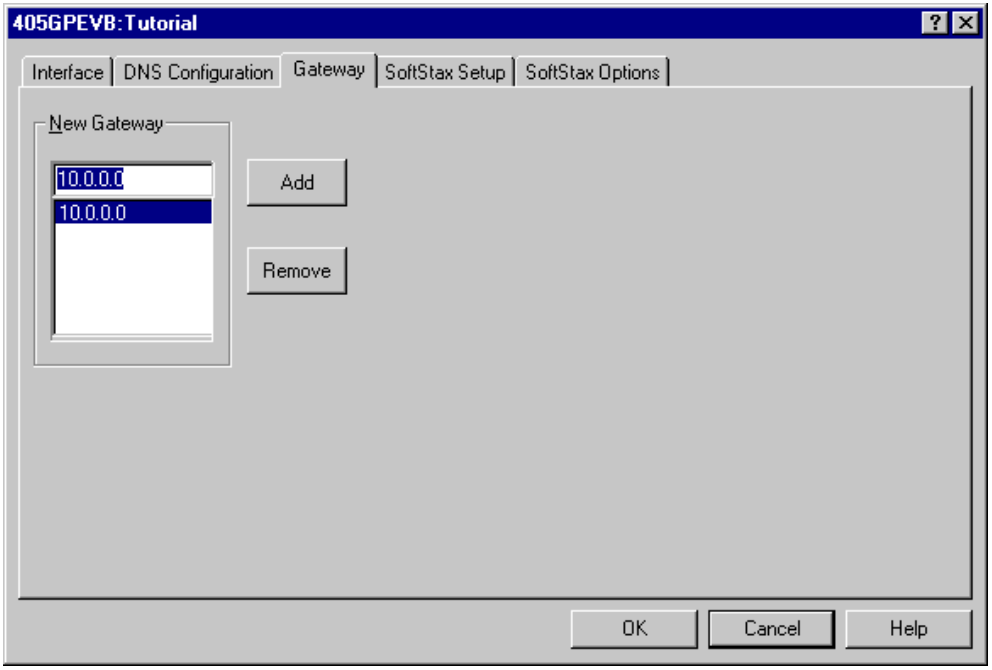
If your network does not use DNS, click `Disable DNS`, and move to the **Gateway** tab.

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

Add DNS IP addresses by clicking on the box directly under **DNS Server Search Order**, typing the IP address, and clicking the `Add` button.

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

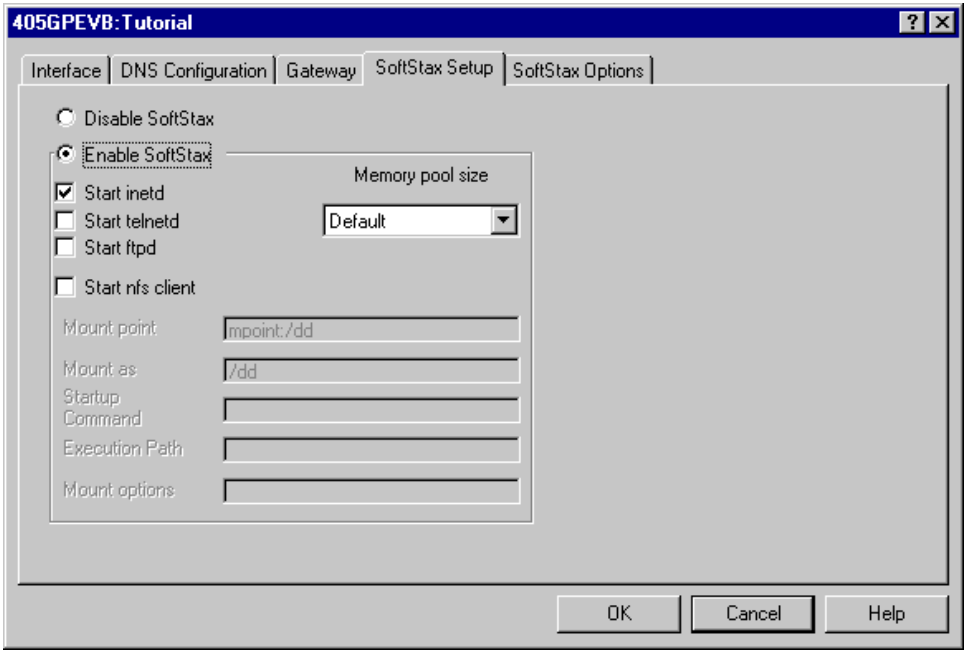
Figure 1-10 Bootfile Configuration—Gateway Tab



Add new gateway address by clicking on the box, typing in the gateway name, and clicking the `Add` button.

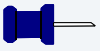
Step 5. Select the **SoftStax® Setup** tab. The following window is displayed.

Figure 1-11 Bootfile Configuration—SoftStax Setup Tab



Step 6. Click `Enable SoftStax`.

The options below represent daemons that can be automatically started if you want to FTP or telnet from a PC to the OS-9 target. **Start NFS Client** enables you to remote mount the target.

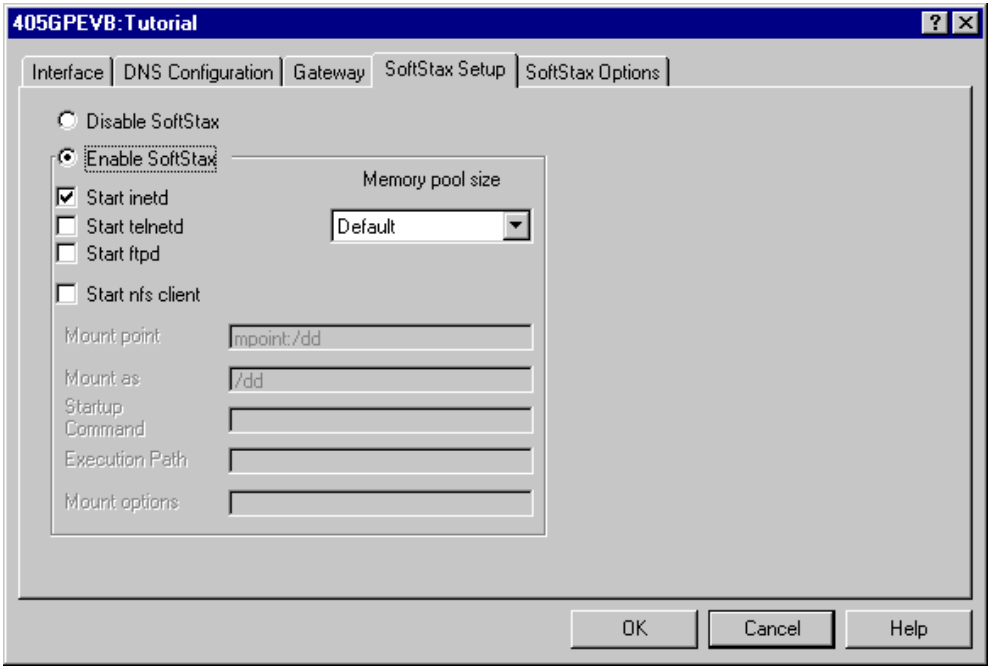


Note

This configuration is set for user state debugging on the target board. For system state Ethernet debugging, select `Disable SoftStax`.

Step 7. Select the **SoftStax Options** tab. The following window is displayed.

Figure 1-12 Bootfile Configuration—SoftStax Options Tab



Step 8. Click OK to return to the **Main Configuration** window.

For More Information

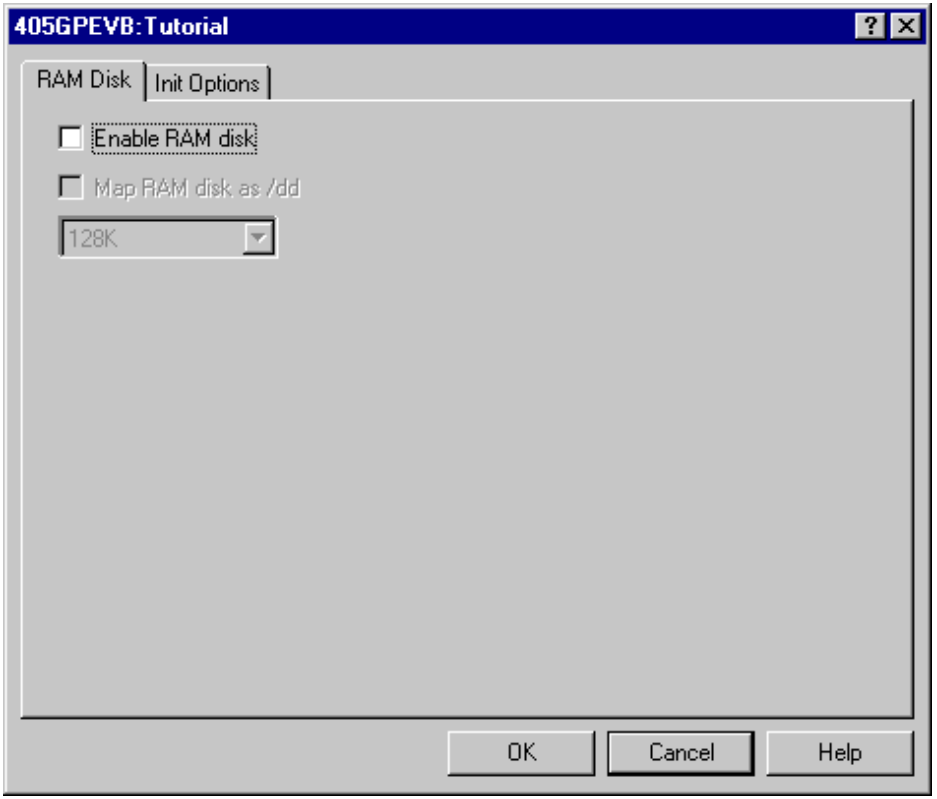
Using LAN Communications has more information about setting your network configuration.

Disk Configuration

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

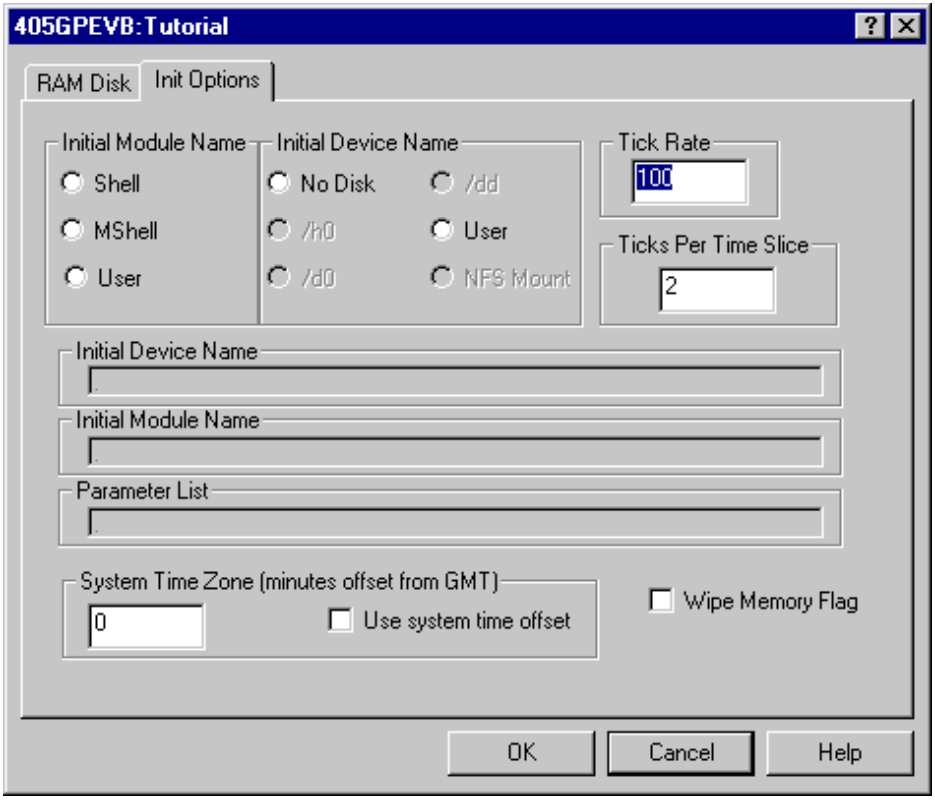
Step 2. Select the **RAM Disk** tab. The following window is displayed. The **RAM Disk** tab enables you to create a RAM disk of any size for loading modules onto the target.

Figure 1-13 Bootfile Configuration—RAM Disk Tab



Step 3. Select the Init Options tab. The following window is displayed. The **Init Options** tab sets the configuration for OS-9 to initialize itself on the target.

Figure 1-14 Bootfile Configuration—Init Options Tab



Step 4. Select the `MShell` option for the initial module name. This causes OS-9 to start a console shell usable from your terminal window. Select `No Disk` in the **Initial Device Name** section.

The tick rate is `100` and ticks per timeslice is set to `2`.

The **Parameter List** box displays the commands that OS-9 executes at system start-up.

Step 5. Click `OK` to return to the **Main Configuration** window.

Build Image

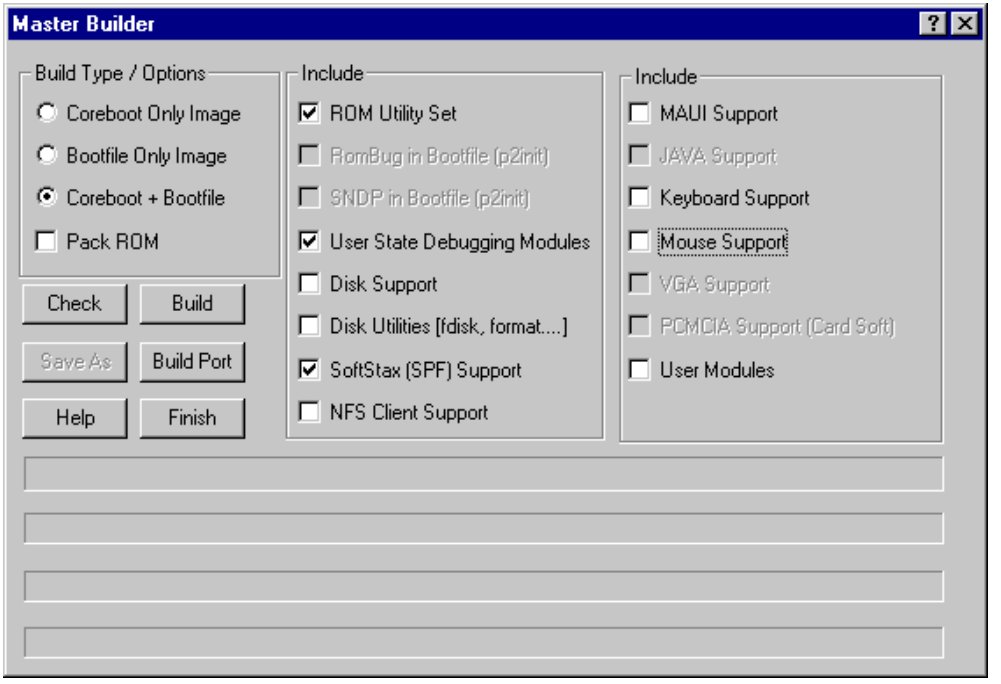
The build process creates a file called `rom` in the following directory on your host system:

```
/mwos/OS9000/403/PORTS/405GPEVB/BOOTS/INSTALL/PORTBOOT/
```

Step 1. Build the `rom` image by selecting `Configure -> Build Image` from the main configuration window.

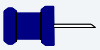
Step 2. Select the `Coreboot + Bootfile` radio button. Disable the `Pack Rom` check-box. Disable `MAUI`, `Keyboard`, and `Mouse` if appropriate. The image shown in **Figure 1-15** is displayed.

Figure 1-15 Master Builder Window-Coreboot Only Image



Step 3. Click on the `Build` button.

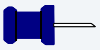
After the image is built, click on the `Finish` button.



Note

This configuration is set for user state Ethernet debugging on the target board. For system state debugging, select `ROMBug in Bootfile (p2init)` and deselect `User State Debugging Modules` under the Include section.

You must also complete the coreboot Ethernet information for system state debugging.



Note

After the `rom` image is built and you are returned to the **Main Configuration** window, you can select `File -> Save Settings` before exiting the Wizard. This saves the settings for your particular configuration.

Transferring the ROM Image to the Target

For the 405GP target board, transferring the ROM Image from the host to the target is done by installing the `rom` file as the bootp boot file for your target. The details of this procedure depend on the bootp server software you are using.

Optional Procedures

Preliminary Testing

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

Step 1. Type `mdir` at the prompt.

`mdir` displays all the modules in memory.

Step 2. Type `procs` at the prompt.

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

Step 3. Test the networking on your system.

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

```
$ ping solkanar
PING solkanar.microware.com (172.16.0.0): 56 data bytes
64 bytes from 172.16.0.0: ttl=128 time=0 ms
```

Step 4. Test `telnet`.

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

```
$ telnet delta
Trying 172.16.0.0...Connected to delta.microware.com.
Escape character is '^]'.
capture closed.

OS-9/68K V3.0.3 Delta VME177 - 68060 98/12/24 14:41:51

User name?: curt
Password:

Process #101 logged on 98/12/24 14:41:56
Welcome!
```

```
*********************************************************
*      WELCOME TO DELTA - THE :OS-9 68K: MACHINE *
```

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

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

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

Chapter 2: Board Specific Reference

This chapter contains information that is specific to the 405GP reference board from IBM. It contains the following sections:

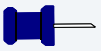
- **Boot Menu Options**
- **Port-Specific Utilities**
- **PowerPC Registers Passed to a New Process**
- **Vector Descriptions for PowerPC 405GP**
- **Configuring Booters**

For More Information

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

Boot Menu Options

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



Note

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

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

```
OS-9 Bootstrap for the PowerPC(tm) (Edition 64)

Now trying to Override autobooters.

Press the spacebar for a booter menu


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

Boot loaded system in-place -------- <bo>
Boot over Ethernet (On-board EMAC) - <eb>
Enter ROM Debugger ----------------- <break>
Restart the System ----------------- <q>

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

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

Table 2-1 Supported Boot Methods

Type of Boot	Description
Ethernet	Boot from over Ethernet from a bootp server (`eb`)
Boot embedded OS-9 in-place	Boot OS-9 from FLASH (`bo`).
Copy embedded OS-9 to RAM and Boot	Copy OS-9 from FLASH (if stored there) to RAM and boot (`lr`).

Port-Specific Utilities

The following port-specific utility is included:

- `pciv`

pciv — PCI Configuration Space View

SYNTAX

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

OPTIONS

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

DESCRIPTION

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

EXAMPLES

When using the `pciv` command with a IBM PowerPC 405GP board, the following information is displayed:

```
$ pciv

BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-------------------------------------------------
000:00:00  1014 0156 0006 2210 060000 01 00 04 01 Bridge Device [S]
```

The `pciv` command in the previous example reports configuration information related to specific hardware attached to the system.

The following are the abbreviations used and their meanings:

BUS - Bus Number

DV - Device Number

FU - Function

VID - Vendor ID

DID - Device ID

CLASS - Class Code

RV - Revision ID

IL - Interrupt Line

IP - Interrupt Pin

[S] - Single function device

[M] - Multiple function device

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

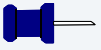
```
$ pciv -a

BUS:DV:FU  VID  DID  CMD  STAT CLASS  RV CS IL IP
-----------------------------------------------
000:00:00  1014 0156 0006 2210 060000 01 00 04 01
(NC) [32-bit] base_addr[1] = 0x00000008  PCI/MEM 0xa0000008 Size = 0x80000000
Bridge Device [S]
```

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

- not prefetchable
- memory type
- address fields
- actual value stored
- type of access
- translated access address used
- size of block

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



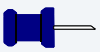
Note

ISA and PCI interrupts cannot be shared.

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) pointer
r13 = constant data pointer
r3  = pointer to fork parameters structure (listed in
                                                F_FORK)
```



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. `r13` is similarly biased. You can usually ignore this bias because the OS-9 linker automatically adjusts for it.

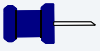
Vector Descriptions for PowerPC 405GP

Table 2-2 Vector Descriptions for PowerPC 405GP

Vector Number	Related OS-9 Call	Assignment
00	None	Reserved
01	F_IRQ	Critical input / Watchdog timer
02	F_STRAP, F_IRQ	Machine check
03	F_STRAP, F_IRQ	Data storage
04	F_STRAP, F_IRQ	Instruction storage
05	F_IRQ (in uicirq)	External interrupt
06	F_STRAP, F_IRQ (in ssm)	Alignment
07	F_STRAP, F_TLINK, F_IRQ (in fpu)	Program
08	None	Reserved
09	F_IRQ	Fixed Interval Timer (FIT)
0A	None	Reserved
0B	None	Reserved
0C	F_SSVC	System call
0D	None	Reserved

Table 2-2 Vector Descriptions for PowerPC 405GP (continued)

Vector Number	Related OS-9 Call	Assignment
0E	None	Reserved
0F	None	Reserved
10	None	Reserved
11	ssm	Implementation dependent data TLB miss
12	ssm	Implementation dependent instruction TLB miss
13 - 1f	None	Reserved
20	None	Debug
21	F_IRQ (in tk403ga)	Programmable Interrupt Timer (PIT)



Note

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

Error Exceptions: vectors 2-4 and 6-7

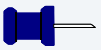
These exceptions are usually considered fatal program errors and unconditionally terminate a user program. If F_DFORK created the process or the process was debug attached with F_DATTACH, then the resources of the erroneous process remain intact and control returns to the parent debugger to allow a post-mortem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

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

If any of these exceptions 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.

OS-9 Vector Mapping

This section contains the vector mappings on the IBM 405GP Reference Board.

The system modules `uicirq` and `fpga405irq` map interrupts coming from UIC and FPGA into the OS-9 vector table according to the following mappings.

Table 2-3 Universal Interrupt Controller Interrupt Vectors

Vector	Source
0x40	COM 1 Serial Port
0x41	COM 2 Serial Port
0x42	Inter-Integrated Circuit (IIC)
0x43	External Master
0x44	PCI
0x45	DMA #0
0x46	DMA #1
0x47	DMA #2
0x48	DMA #3
0x49	Ethernet Wake Up
0x4a	MAL System Error (SERR)
0x4b	MAL Transmit End Of Buffer
0x4c	MAL Receive End Of Buffer
0x4d	MAL Transmit Descriptor Error
0x4e	MAL Receive Descriptor Error
0x4f	Ethernet

Table 2-3 Universal Interrupt Controller Interrupt Vectors

Vector	Source
0x50	PCI System Error
0x51	Error Checking and Correction (ECC) Correctable Error
0x52	PCI Power Management
0x53-0x58	Reserved
0x59	External IRQ #0 (FPGA)
0x5a	External IRQ #1 (FPGA)
0x5b	External IRQ #2 (IrDA)
0x5c	External IRQ #3 (PCI Slot #3)
0x5d	External IRQ #4 (PCI Slot #2)
0x5e	External IRQ #5 (PCI Slot #1)
0x5f	External IRQ #6 (PCI Slot #0)

The individual interrupts from the FPGA are mapped into the range 0x60-0x64. The following table describes the mapping:

Table 2-4 FPGA Interrupt Vectors

Vector	Source
0x60	PS/2 Mouse
0x61	PS/2 Keyboard

Table 2-4 FPGA Interrupt Vectors (continued)

Vector	Source
0x62	IrDA
0x63	Expansion Interface
0x64	Critical Interrupt Signal

Configuring Booters

The following booters are available for the 405GP target platforms. The abbreviated name and configuration parameters for the booters are listed with recommended values (if any).

Note

The 405GP booters are located in `coreboot.ml`.

Table 2-5 405GP Booters

Booter	Description	Recommended Values
`llbootp`	Standard BOOTP booter	
	Abbreviated name:	`"eb"`

	Configuration parameters	`"driver=ll405gp"`
		`"bootfile=os9boot"`
		`"maxbootptry=8"`

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 IBM 405GP target platform. These modules can be found in the following directory:

`MWOS/OS9000/403/PORTS/405GPEVB/CMDS/BOOTOBJS/ROM`

Configuration Modules

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

Console Drivers

`io16550`	provides console services for the 16550 UARTs on the 405GP Reference Board.

Debugging Modules

`usedebug`	is a debugger configuration module.

Ethernet Driver

`ll405gp`	provides network driver services for the on-board EMAC/MAL.

System Modules and Files

`initext`	is a user-customizable system initialization module.
`portmenu`	retrieves a list of configured booter names from the ROM `cnfgdata` module.
`romcore`	provides bootstrap code.
`romstart`	resets vectors.
`evbstart`	is a binary header that tells the IBM Ethernet boot loader where to load the boot.

Timer Modules

`tbtimer`	provides polling timer services using the `tblo` and `tbhi` registers in the 405GP processors.

High-Level System Modules

The following OS-9 system modules are tailored specifically for your 405GP platform. Unless otherwise specified, each module can be found in a file of the same name in the following directory:

`<MWOS>/OS9000/403/PORTS/405GPEVB/CMDS/BOOTOBJS`

Pseudo Vectoring Modules

`uicirq`	remaps the various interrupts on vector 5 to those on vectors 0x40 to 0x5f.
`fpga405irq`	remaps the various interrupts on vectors 0x59 and 0x5a to those on vectors 0x60 to 0x64.

Real Time Clock Driver

`rtc1x43`	provides OS-9 access to the real time clock.

Ticker

`tk403ga`	provides the system ticker based on the Programmable Interval Timer.

Shared Libraries

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

Serial and Console Drivers

`sc16550`	provides support for the 16550 UART serial port. The descriptors provided for this driver are named `t0, t1, term_t0,` and `term_t1.` They are located in the following directory: `DESC/SC16550`
`scp87303`	provides serial port support.

PS/2 Mouse and Keyboard Driver

`sc8042k`	is a keyboard and mouse driver used by MAUI.

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 A-1 Common System Modules List

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

Table A-1 Common System Modules List (continued)

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

Table A-1 Common System Modules List (continued)

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

Table A-1 Common System Modules List (continued)

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