OS-9 for MIPS Atlas TI 4KC Core Board Guide

Version 3.2

www.radisys.com

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

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

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

Revision A
December 2001

Copyright and publication information

This manual reflects version 3.2 of Enhanced OS-9 for MIPS32.

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.

December 2001

Table of Contents

Chapter 1: Installing and Configuring OS-9

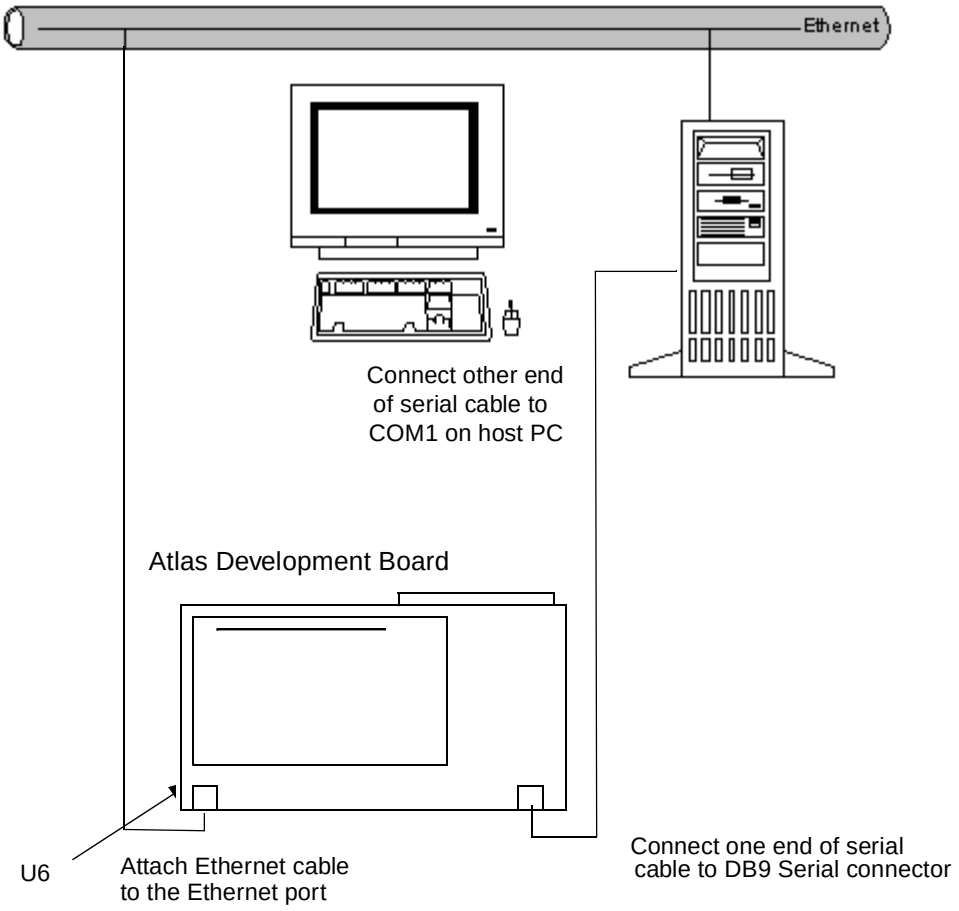
This chapter describes installing and configuring OS-9 on the MIPS Atlas™ board. It includes the following sections:

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

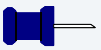
Development Environment Overview

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

Figure 1-1 MIPS Atlas™ Development Environment



Requirements and Compatibility



Note

Before you begin, install the ***Enhanced OS-9 for MIPS CD-ROM*** on your host PC.

Host Hardware Requirements (PC Compatible)

Your host PC must have the following minimum hardware characteristics:

- 32MB of RAM
- an Ethernet network card

Host Software Requirements (PC Compatible)

Your host PC must have the following software installed:

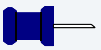
- Windows 95, 98, ME, 2000, or NT

Target Hardware Requirements

Your MIPS evaluation board requires the following hardware:

- a power supply
- an RS-232 null modem serial cable (for serial console)
- an Ethernet cable (for down-loading programs to the board)

Target and Host Setup



Note

Before installing and configuring OS-9 on your MIPS evaluation board, refer to the hardware documentation for information on hardware setup.

Jumper and Switch Settings

The factory default setting for the DIP switches will work with OS-9. If you need to change any of the jumper or switch settings, please refer to the ***Atlas™ Developer's Kit Getting Started Manual***.

Installing the TFTP Server

This section details the steps involved with setting up the Walusoft TFTP server on your host machine. If you choose to use another TFTP server in place of the Walusoft TFTP server, be certain to follow its directions and specifications.

To set up the Walusoft TFTP server on your host machine, complete the following steps:

Step 1. Load the product CD into the host machine's CD ROM drive and let the CD autorun. The installer's dialog box appears.

Step 2. Select `Walusoft TFTPServer32Pro` from the installer's menu and follow the installer's directions to load the TFTP server on the host machine.

Step 3. Start TFTPServer32Pro by selecting `Start` -> `Programs` -> `TFTPServer` -> `TFTPServer32`. The Walusoft TFTP Server32 Pro splash screen appears. You will need to set up the path to the outbound folder.

Step 4. Select `System` -> `Setup` to display the **Server Options** dialog box.

Step 5. Click on the **Outbound** tab and enter the following path in the **Outbound file path** text box.
`<drive>:\mwos\OS9000\MIPS32\PORTS\ATLAS_4KC\BOOTS\INSTALL\PORTBOOT\`

Step 6. Click `OK` to accept the path. The TFTP server is now configured for use with the Configuration Wizard and OS-9.

Step 7. Minimize the TFTP server window to let the server run in the background.

Connecting the Target to the Host

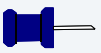
Connecting the Atlas™ board to your host PC involves attaching the power, serial, and Ethernet cables to the reference board. Once you have the board connected, you can use the serial console in Hawk to verify the serial connection.

Attaching the Cables

Step 1. Attach an Ethernet cable to the Ethernet connector on the Atlas™ board.

Step 2. Connect a serial cable to the DB9 serial connector on the Atlas™ board for the serial console.

Step 3. Connect the other end of the serial cable to COM1 on the host PC. Depending on your PC system, you may need either a straight or a reversed serial cable to make this connection.



Note

A standard null modem cable may not work. You may need a cable that has transmit and receive swapped. For the Atlas™ board, be sure the hardware signals RTS/CTS, DSR and DTR are connected correctly. Its default baud rate is 38400.

Step 4. Following the instructions in the ***Atlas™ Developer's Kit Getting Started Manual***, attach a PC power supply.

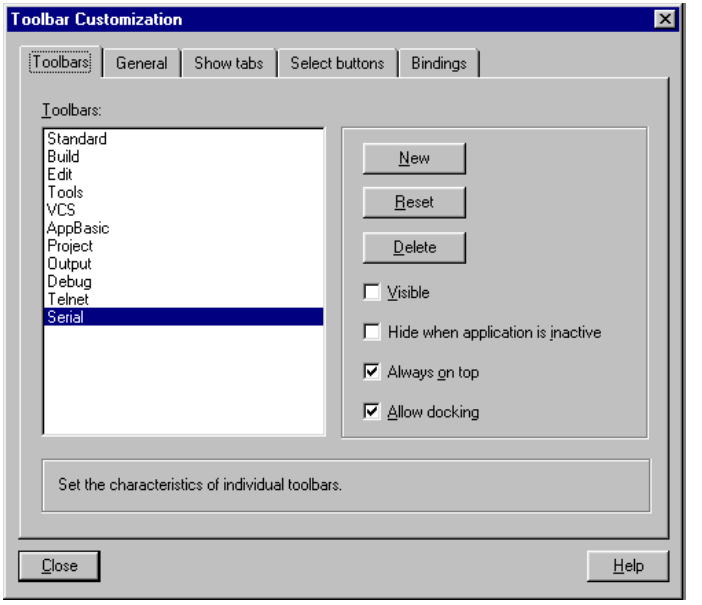
Booting to the Boot Menu

You may want to boot to the MIPS YAMON monitor prompt to verify that your serial cable is connected properly. By default, the MIPS YAMON monitor runs at a baud rate of 38400.

Step 1. From the desktop, click `Start` and select `Programs` -> `Microware` -> `Enhanced OS-9 for MIPS v3.0` -> `Microware Hawk IDE` to start the Microware Hawk IDE.

Step 2. If the **Serial** console window is not open, it can be opened from the **Toolbar Customization** dialog (shown in **Figure 1-2**). (Select `Customize` -> `Toolbars...` to open the **Toolbar Customization** dialog box.)

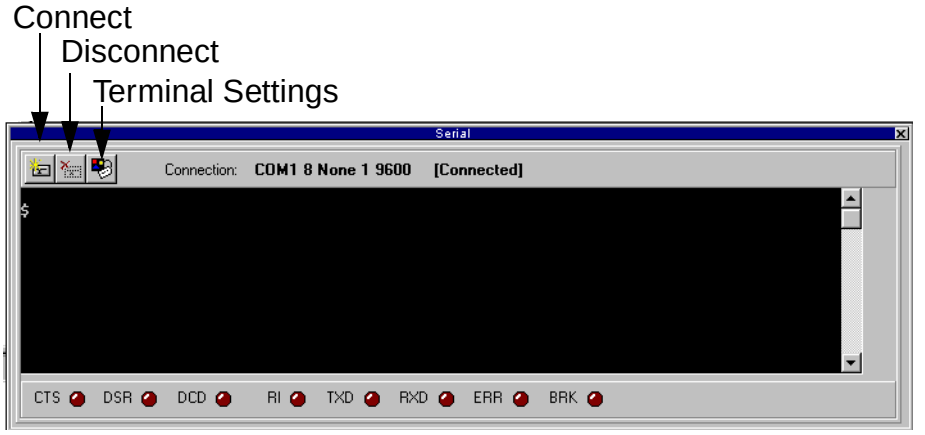
Figure 1-2 Toolbar Customization dialog box



Step 3. Once the **Toolbar Customization** dialog box is open, select `Serial` in the **Toolbars** list box.

Step 4. Click the `Visible` check box, then click the `Close` button. The **Serial** console window opens. (The **Serial** window can be seen in **Figure 1-3**.)

Figure 1-3 Hawk Serial Console Window



Step 5. Once you have the serial console window open, click on the `Connect` button in the upper left corner of the serial console window. The **Com Port Options** dialog box appears.

Step 6. Click on the `OK` button because the default settings are correct. The message *[Not Connected]* should change to *[Connected]*.

Step 7. Apply power to the board. The YAMON monitor boots the board. The display will look similar to the following figure.

Figure 1-4 IDT SIM initial screen

```
YAMON ROM Monitor, Revision 01.01.
Copyright (c) 1999-2000 MIPS Technologies, Inc. - All Rights Reserved.

For a list of available commands, type 'help'.

Compilation time =            Jan 31 2000  16:07:08
Board type/revision =         0x00 (Atlas) / 0x00
Core board type/revision =    0x01 (CoreLV) / 0x00
FPGA revision =               0x0007
MAC address =                 00.d0.a0.00.00.27
Board S/N =                   0000000039
PCI bus frequency =           33 MHz
Processor ID/revision =       0x80 (MIPS 4Kc) / 0x01
Endianness =                  Big
CPU frequency =               80 MHz
CPU bus frequency =           40 MHz
Flash memory size =           36 MByte
RAM size =                    32 MByte
First free RAM address =      0x80066628

YAMON>
```

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 or Ethernet network. It is also responsible for building basic structures based on the image it finds and passing control to the kernel to bring up the OS-9 system.

Bootfile

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

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

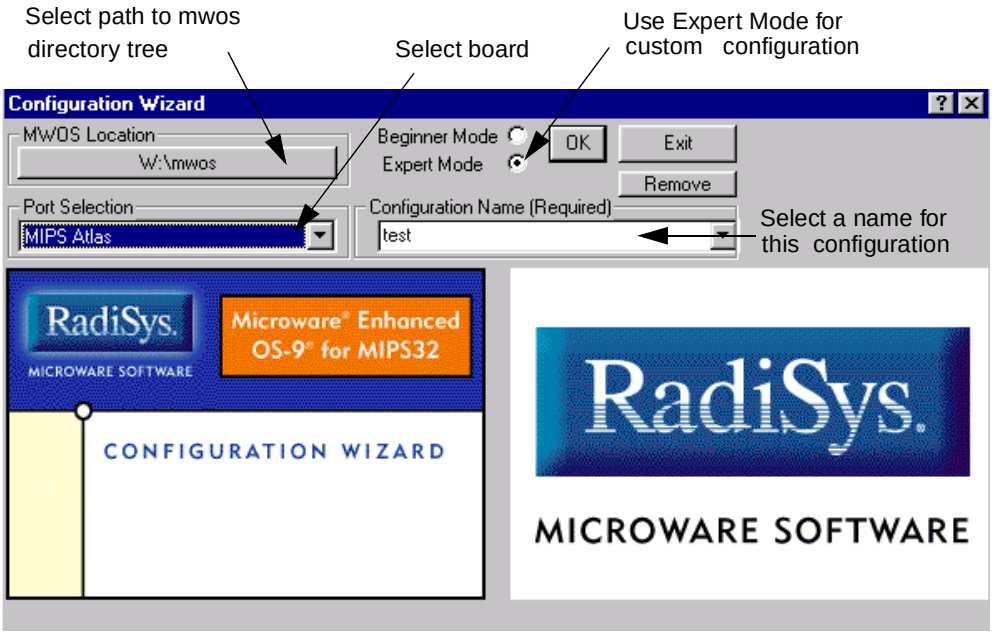
Using the Configuration Wizard

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

Step 1. Click the `Start` button on the Windows desktop.

Step 2. On the Windows desktop, select `Start` --> `Programs` --> `Microware` --> `Enhanced OS-9 for MIPS v3.0` --> `Microware Configuration Wizard`. You should see the following opening screen.

Figure 1-5 Configuration Wizard Opening Screen



Step 3. Select the path where the MWOS directory structure is located by clicking the **MWOS Location** button.

Step 4. Select the target board from the **Port Selection** pull-down menu.

Step 5. Select a name for your configuration in the **Configuration Name** field. This enables you to modify and save your settings for the ROM image incrementally, without having to reselect every option for each change.

Step 6. Select `Expert Mode` and click `OK`. The main configuration window is displayed. Expert mode enables you to make more detailed and specific choices about what modules are included in your 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.

Configuring Coreboot Options

Most of the default options in the dialogs that control the configuration of the bootfile are correct. There are a few functions, such as Ethernet, that need additional information in order to be configured correctly. To set up the coreboot image, complete the following steps.

Step 1. From the menu bar, select `Configure` -> `Select System Type...`.

Step 2. Click on the **RomCore** tab and verify the **Boot using YAMON** romcore type option button is selected.

Step 3. Click on the **ROM Memory List** tab and verify that the **Setting Based On** drop-down list box has either `default` or `YAMON boot` selected.

Step 4. Click `OK` to save any changes.

Step 5. To set up the coreboot networking, select `Configure` -> `Coreboot` -> `Main Configuration` from the menu bar.

Step 6. Click on the **Debugger** tab. Make sure `Ethernet` is selected in the **Remote Debug Connection** area and `Remote` is selected in the **Select Debugger** area. Remote debugging is enabled so that system-state debugging can be performed in Hawk.

Step 7. Click on the `Ethernet` tab and enter the Ethernet address information in the address text boxes. For most situations you will need to fill out the following text boxes:

- **IP Address**
- **IP Broadcast**
- **Subnet Mask**
- **IP Gateway**

If you are uncertain of the values for these text boxes, contact your system administrator.

Step 8. Click `OK` to close the window.

Configuring Bootfile Options

Most of the default options in the dialogs that control the configuration of the bootfile are correct. There are a few functions, such as Ethernet, that need additional information in order to be configured correctly. To configure your bootfile options, complete the following steps:

Step 1. To configure the Ethernet function, select `Configure` -> `Bootfile` -> `Network Configuration..` from the menu bar.

Step 2. Click on the `Interface` tab.

Step 3. Click on `Ethernet Connection` in the **Select Interface** list.

Step 4. Make sure that **Specify an IP Address** is selected and the address information in the IP address text boxes is correct. They should have been copied from the coreboot Ethernet dialog box. If the information is not correct, correct it now.

Step 5. Select the `Ethernet` check box in the **Enable Interface** area.

Step 6. Make sure the name of the Ethernet controller chip is displayed in the combo box located to the right of the **Enable Interface** area. If it is not visible, the Ethernet modules will not be included in the build.

Step 7. Click on the `SoftStax Setup` tab, and select `Enable SoftStax`.

Step 8. Click `OK` to close the dialog box.

Step 9. Select `Configure` -> `Bootfile` -> `Disk Configuration..` from the menu bar and verify that the default settings in the dialog box are acceptable to you.

Step 10. Leave the other default settings alone and select `Configure` -> `Build Image..` from the menu bar to display the **Master Builder** window.

Step 11. Select the following check boxes as they are appropriate to your setup:

- SoftStax (SPF) Support
- User State Debugging Modules
- If you are using a RAM disk, select `Disk Support`.
- If you are using a RAM disk, select `Disk Utilities`.

Step 12. Click `Coreboot + Bootfile` and click `Build`. This will build the ROM image that can be burned into flash memory. The name of the file containing the ROM image is `rom.S`. It is in the Motorola S-record format. The file rom.s is located in `mwos\OS9000\MIPS32\PORTS\ATLAS_4KC\BOOTS\INSTALL\PORTBOOT`. This directory was specified as the outgoing directory when the TFTP server was set up.

Step 13. Click `Finish` and then select `File` -> `Save Settings` to save the configuration.

Step 14. Select `File` -> `Exit` to quit from the Configuration Wizard.

Transferring the ROM Image to the Target

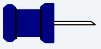
In the previous section, you built a ROM image. To load this ROM image onto the target board, complete the following steps:

Step 1. The networking environment variables on the Atlas™ board need to be set up before you use it for the first time. The `ipaddr`, `subnetmask`, `bootfile`, and `bootserver` environment variables need to be set with the `set` command. To set these variables, type the following commands at the YAMON prompt:

```
set ipaddr <Atlas™ board's IP address>
set subnetmask <xxx.xxx.xxx.xxx>
set bootfile rom.S
set bootserver <IP address of the TFTP server>
```

Download the OS-9 ROM image, `rom.S`, to the Atlas™ board using the following load command at the YAMON prompt:

```
load
```



Note

It will about a minute to download the entire image.

Note

If you have trouble with downloading the image, be sure the following items are correct:

- The environment variables on the board are set to correct values.
- The load command was correctly entered (use of spaces is correct).
- The path to the Outbound folder in your TFTP server is correct.
- The Ethernet connector is plugged in and the board has power applied to it.

Step 2. Enter the following command to start OS-9.

```
go 80070000
```

Step 3. To be able to use Hawk to load and debug your applications, you need to start the debugging daemons. Type the following command to start the debugging daemons:

```
spfndpd<>>>/nil&
```

Optional Procedures

The following sections describe procedures you may perform once you have installed and configured OS-9.

Burning the OS-9 ROM Image into FLASH memory

The Wizard can be used to build an OS-9 ROM image which is burned into the Atlas™ board's 4 megabyte of bootable FLASH memory. This is explained in the following steps.

Step 1. Build a YAMON OS-9 ROM image with networking enabled as explained previously in **Configuring Bootfile Options** on page 17.

Step 2. Select `Configure` -> `Bootfile` -> `Network Configuration..` to display the networking dialog box. Make sure FTP is included in the boot by clicking on the **SoftStax Options** tab and verifying that the **ftp** check box is selected.

Step 3. Select `Configure` -> `Bootfile` ->`Configure System Options..` and click on the **Bootfile Options** tab. Verify that the **pflash** button is checked to include the pflash utility. This will include an 8 megabyte RAM disk called **r1**.

Step 4. Use YAMON to boot the OS-9 ROM image.

Step 5. Use the Wizard to build a flashable boot. Select `Configure` -> `Select System Type` to display the **Select System Type..** window, and check the `OS-9 Image in flash` button under the **RomCore** tab. Under the **ROM Memory List** tab and in the **Setting Based On** box, select `OS-9 Image in flash`. `Click OK`. Select further Wizard options as needed. For example, it would be good to include the `lr` option which is found on the **Define Other Boot Options** tab in the **configure**->**coreboot**->**main configuration** dialog box.

Step 6. Build the OS-9 ROM image.

Step 7. In order to transfer the OS-9 ROM image, initialize the `r1` descriptor with the `iniz` command and change directory to `r1` (`chd r1`). Use `ftp` to transfer the ROM file to the Atlas board. To flash the image, type `pflash -ri -f=rom`. Then wait until it is finished.

Step 8. Reset the board.

Further OS-9 ROM images can be transferred with FTP and flashed as needed. A common practice is to flash a coreboot only with the Ethernet boot (eb boot) option. Then the wizard can be used to build bootfiles only. The bootfiles can be moved to bootp servers. The Atlas can be booted then using bootp. This allows easy changes to the bootfile without flashing all the time.

If it is desired to go back to YAMON after OS-9 has been flashed, use the standard YAMON flashing procedure of transferring the YAMON image over the parallel port.

Building with Makefiles

Building boots with makefiles allows you greater control over which modules are included in the boot. For the Atlas™ reference board, there is a directory in which boots can be made. This is shown below:

```
MWOS\OS9000\MIPS32\PORTS\ATLAS_4KC\BOOTS\SYSTEMS\PORTBOOT
```

By altering the various `.ml` files within this directory, specific boots can be made. These `.ml` files are described in a following section called **Using Makefiles**.

romcore vs. romcore_eprom

There are two romcore files for the ATLAS port. The `romcore` file is used for YAMON boots, and the `romcore_eprom` file is used for OS-9 flash boots. The `coreboot.ml` file can be changed to select the romcore file for the desired boot method.

Makefile Network Option

By default the makefile in the PORTBOOT directory will not include networking. However, by setting the NETWORK macro definition to TRUE, the networking modules will be included in the bootfile. In addition, be sure the IP and MAC addresses for the board are setup correctly to avoid network problems.

Using Makefiles

When using a makefile to build boots, three bootlist files are used to include the modules for booting. These bootlist files can be edited in order to include or not include modules needed for your system. These bootlist files are located in ATLAS_4KC\BOOTS\SYSTEMS\PORTBOOT. They are defined as follows:

`coreboot.ml`	used to make the low-level boot (called `coreboot`) When using this file, the `romcore` file must be input first, followed by the `initext` file. These two files are not OS-9 modules. `romcore` is the raw code needed to bring the hardware to a known stable state, while `initext` is a way for users to extend the low level `sysinit` code without changing `sysinit.c` or remaking `romcore`. Use `romcore` for YAMON boots and `romcore_eprom` for OS-9 FLASH boots. The rest of the files included with `coreboot.ml` are actual OS-9 modules. Low-level booters and debuggers can be added or removed. In addition, the low-level Ethernet and IP stack can be uncommented in order to provide `bootp` booting. Low-level Ethernet or low-level SLIP can also provide system state debugging through Hawk.

Note

Only OEM licensees have the ability to make romcore. BLS licensees do not have this ability.

`bootfile.ml`	used to create the high-level boot (called `bootfile`) This file contains all of the modules needed to produce an OS-9 system. This includes the kernel, system protection, cache control, file managers, and drivers and descriptors. Also included are various utilities and application programs. Not included with this file are networking modules. Additional modules can be included or excluded where appropriate.
`spf_mods.ml`	contains the SoftStax modules and network utilities These modules are simply merged into the end of the bootfile created from the `bootfile.ml` bootlist.

Making Network Configuration Changes

To configure the network parameters for SoftStax and Ethernet, two files need to be changed and two makefiles need to be run. To do this, complete the following steps:

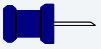
Step 1. Navigate to `MWOS\OS9000\MIPS32\PORTS\ATLAS_4KC\SPF\ETC` directory and open the `interfaces.conf` file.

Step 2. From the `interfaces.conf` file, fill in the correct IP address, broadcast address, and netmask values. You can also supply the host name in this area as well.

Step 3. Save the file.

Step 4. Once you have saved the file, run the makefile in the directory listed in step one. This will make the appropriate `inetdb` and `inetdb2` modules.

Step 5. Once this is done, go to the `MWOS\OS9000\MIPS32\PORTS\ATLAS_4KC\SPF\SPSA9730` directory and run the `spfdesc.mak` makefile. This will create the `spsa0` descriptor. At this point networking is configured for SoftStax.



Note

Because the Configuration Wizard configures the network in its own manner, if you are using it to configure network parameters, the above changes are not needed. However, if you choose to make the above changes, the Wizard will remain unaffected.

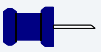
Low Level Network Configuration Changes

To configure the low-level Ethernet parameters, one file needs to be altered and one makefile needs to be run. To do this, complete the following steps:

Step 1. Navigate to the `MWOS\OS9000\MIPS32\PORTS\ATLAS_4KC\ROM\CNFGDATA` directory and open the `config.des` file.

Step 2. In the `config.des` file, you will need to correctly define the macros for the IP, broadcast, subnet, and MAC addresses.

Step 3. Run the makefile in the directory listed in step one and a new `cnfgdata` module will be created. A coreboot can now be created with this configuration.



Note

Because the Configuration Wizard configures the network in its own manner, if you are using it to configure Ethernet parameters, the above changes are not needed. However, if you choose to make the above changes, the Wizard will remain unaffected.

Chapter 2: Board Specific Reference

This chapter contains porting information specific to the MIPS board. It includes the following sections:

- **The Fastboot Enhancement**
- **OS-9 Vector Mappings**

The Fastboot Enhancement

The Fastboot enhancements to OS-9 were added to address the needs of embedded systems that require faster system bootstrap performance. The Fastboot concept exists to inform OS-9 that the defined configuration is static and valid. This eliminate the dynamic search OS-9 usually performs during the bootstrap process. It also allows the system to perform for a minimal amount of runtime configuration. As a result, a significant increase in bootstrap speed is achieved.

Overview

The Fastboot enhancement consists of a set of flags that control the bootstrap process. Each flag informs some portion of the bootstrap code of a particular assumption, and that the associated bootstrap functionality should be omitted.

One important feature of the Fastboot enhancement is the ability of the flags to become dynamically altered during the bootstrap process. For example, the bootstrap code might be configured to query dip switch settings, respond to device interrupts, or respond to the presence of specific resources that indicate different bootstrap requirements.

Another important feature of the Fastboot enhancement is its versatility. The enhancement's versatility allows for special considerations under a variety of circumstances. This can be useful in a system in which most resources are known, static, and functional, but whose additional validation is required during bootstrap for a particular instance (such as a resource failure).

Implementation Overview

The Fastboot configuration flags have been implemented as a set of bit fields. One 32-bit field has been dedicated for bootstrap configuration. This four-byte field is contained within a set of data structures shared by

the kernel and the ModRom sub-components. Hence, the field is available for modification and inspection by the entire set of system modules (both high-level and low-level).

Currently, there are six-bit flags defined, with eight bits reserved for user-definable bootstrap functionality. The reserved user-definable bits are the high-order eight bits (31-24). This leaves bits available for future enhancements. The currently defined bits and their associated bootstrap functionality are listed in the following sections.

B_QUICKVAL

The `B_QUICKVAL` bit indicates that only the module headers of modules in ROM are to be validated during the memory module search phase. Limiting validation in this manner will omit the CRC check on modules, which may save you a considerable amount of time. For example, if a system has many modules in ROM, in which access time is typically longer than it is in RAM, omitting the CRC check will drastically decrease the bootstrap time. Furthermore, since it is rare that data corruption will occur in ROM, omitting the CRC check is a safe option.

In addition, the `B_OKRAM` bit instructs the low-level and high-level systems to accept their respective RAM definitions without verification. Normally, the system probes memory during bootstrap based on the defined RAM parameters. This method allows system designers to specify a possible range of RAM the system will validate upon startup; thus, the system can accommodate varying amounts of RAM. However, in an embedded system (where the RAM limits are usually statically defined and presumed to be functional) there is no need to validate the defined RAM list. Bootstrap time is saved by assuming that the RAM definition is accurate.

B_OKROM

The `B_OKROM` bit causes acceptance of the ROM definition without probing for ROM. This configuration option behaves similarly to the `B_OKRAM` option with the exception that it applies to the acceptance of the ROM definition.

B_1STINIT

The B_1STINIT bit causes acceptance of the first init module found during cold-start. By default, the kernel searches the entire ROM list passed up by the ModRom for init modules before it takes the init module with the highest revision number. Using the B_1STINIT in a statically defined system omits the extended init module search, which can save a considerable amount of time.

B_NOIRQMASK

The B_NOIRQMASK bit instructs the entire bootstrap system to not mask interrupts for the duration of the bootstrap process. Normally, the ModRom code and the kernel cold-start mask interrupts for the duration of the system startup. However, in systems with a well-defined interrupt system (systems that are calmed by the sysinit hardware initialization code) and a requirement to respond to an installed interrupt handler during startup, this option can be used. Its implementation will prevent the ModRom and kernel cold-start from disabling interrupts. (This is useful in power-sensitive systems that need to respond to “power-failure” oriented interrupts.)

Note

Some portions of the system may still mask interrupts for short periods during the execution of critical sections.

B_NOPARITY

If the RAM probing operation has not been omitted, the B_NOPARITY bit causes the system to not perform parity initialization of the RAM. Parity initialization occurs during the RAM probe phase. The B_NOPARITY option is useful for systems that either require no parity initialization or only require it for “power-on” reset conditions. Systems that only require parity initialization for initial power-on reset conditions can dynamically use this option to prevent parity initialization for subsequent “non-power-on” reset conditions.

Implementation Details

This section describes the compile-time and runtime methods by which you can control the bootstrap speed of your system.

Compile-time Configuration

The compile-time configuration of the bootstrap is provided by a pre-defined macro, `BOOT_CONFIG`, which is used to set the initial bit-field values of the bootstrap flags. You can redefine the macro for recompilation to create a new bootstrap configuration. The new, over-riding value of the macro should be established as a redefinition of the macro in the `rom_cnfg.h` header file or a macro definition parameter in the compilation command.

The `rom_cnfg.h` header file is one of the main files used to configure the ModRom system. It contains many of the specific configuration details of the low-level system. Below is an example of how you can redefine the bootstrap configuration of your system using the `BOOT_CONFIG` macro in the `rom_cnfg.h` header file:

```
#define BOOT_CONFIG (B_OKRAM + B_OKROM + B_QUICKVAL)
```

Below is an alternate example showing the default definition as a compile switch in the compilation command in the makefile:

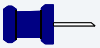
```
SPEC_COPTS = -dNEWINFO –dNOPARITYINIT
–dBOOT_CONFIG=0x7
```

This redefinition of the `BOOT_CONFIG` macro results in a bootstrap method, which accepts the RAM and ROM definitions without verification. It also validates modules solely on the correctness of their module headers.

Runtime Configuration

The default bootstrap configuration can be overridden at runtime by changing the `rinf->os->boot_config` variable from either a low-level P2 module or from the `sysinit2()`function of the

sysinit.c file. The runtime code can query jumper or other hardware settings to determine which user-defined bootstrap procedure should be used. An example P2 module is shown below.



Note

If the override is performed in the sysinit2() function, the effect is not realized until after the low-level system memory searches have been performed. This means that any runtime override of the default settings pertaining to the memory search must be done from the code in the P2 module code.

```
#define NEWINFO
#include <rom.h>
#include <types.h>
#include <const.h>
#include <errno.h>
#include <romerrno.h>
#include <p2lib.h>

error_code p2start(Rominfo rinf, u_char *glbls)
{
   /* if switch or jumper setting is set… */
   if (switch_or_jumper == SET) {
      /* force checking of ROM and RAM lists */
      rinf->os->boot_config &= ~(B_OKROM+B_OKRAM);
   }
   return SUCCESS;
}
```

OS-9 Vector Mappings

This section contains the vector mappings for the Atlas™ board.

Table 2-1 shows the OS9 IRQ assignment for the target board.

Table 2-1 IRQ Assignments

OS9 IRQ #	MIPS Function
0x40	On-board interrupt controller
0x45	CPU internal counter
0x48	16550 serial controller
0x49	Timer 0
0x4C	Real time Clock
0x4D	Core card low priority interrupt
0x4E	Core card high priority interrupt
0x50	Compact PCI IntA#
0x51	Compact PCI IntB#
0x52	Compact PCI IntC#
0x53	Compact PCI IntD#
0x54	Enum# signal from PCI backplane
0x55	DEG# signal from PCI backplane
0x56	Indicates ATX PSU about to fail

Table 2-1 IRQ Assignments (continued)

OS9 IRQ #	MIPS Function
0x57	Primary PCI INTA# (PCI slot only)
0x58	Primary PCI INTB# (includes all SAA9730 devices)
0x59	Primary PCI INTC# (includes SCSI)
0x5A	Primary PCI INTD# (PCI slot only)
0x5B	Primary PCI SER R# signal

Appendix A: Board Specific Modules

This chapter describes the modules specifically written for the MIPS 79S334 boards. It includes the following sections:

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

Low-Level System Modules

The following low-level system modules are tailored specifically for the Atlas™ board. They are located in the following directory:

`MWOS/OS9000/MIPS32/PORTS/ATLAS_4KC/CMDS/BOOTOBJS/ROM`

`cnfgdata`	contains low-level configuration data
`cnfgfunc`	provides access services to the `cnfgdata`
`commcnfg`	inits communication port defined in `cnfgdata`
`conscnfg`	inits console port defined in `cnfgdata`
`initext`	user-customizable system initialization module
`io16550`	ROM based serial IO driver
`llsa9730`	Low-level Ethernet ROM driver
`portmenu`	inits booters defined in the `cnfgdata`
`romcore`	bootstrap code
`tmratlas`	ROM timer services
`usedebug`	debugger configuration module

High-Level System Modules

The following OS-9 system modules are tailored specifically for the MIPS Atlas™ board. Unless otherwise specified, each module is located in the following directory:

`MWOS/OS9000/MIPS32/PORTS/ATLAS_4KC/CMDS/BOOTOBJS`

`counter`	Dummy IRQ handler that handles MIPS 32 counter interrupts
`sc16550`	Serial driver for the 16550 UART
`pcirq`	Provide interrupt acknowledge and dispatching support for the 32334 on-chip interrupt controller.
`rtc1687`	Real time clock module for IC87 real time clock device
`rtcinit`	Setup module for the real time clock device
`sc9730`	Serial driver for SAA9730 serial device
`tkatlas`	System clock module
`vectmips32`	Vector module for MIPS 32

Common System Modules List

The following low-level system modules provide generic services for OS9000 Modular ROM. Your board port may or may not have these modules depending on your reference board's capabilities.They are located in the following directory:

`MWOS/OS9000/MIPS32/CMDS/BOOTOBJS/ROM`

`bootsys`	provides booter registration services
`console`	provides console services
`dbgentry`	inits debugger entry point for system use
`dbgserv`	provides debugger services
`excption`	provides low-level exception services
`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 low-level cache management services
`fsboot`	is a SCSI TEAC floppy disk drive booter
`hlproto`	provides user level code access to protoman
`hsboot`	is a SCSI hard disk drive booter
`ide`	provides target-specific standard IDE support, including PCMCIA ATA PC cards
`llbootp`	provides `bootp` services
`llip`	provides low-level IP services

`llkermit`	provides a booter that uses kermit protocol
`llslip`	provides low-level SLIP services
`lltcp`	provides low-level TCP services
`lludp`	provides low-level UDP services
`notify`	provides state change information for use with LL and HL drivers
`override`	provides a booter that allows a choice between menu and auto booters
`parser`	provides argument parsing services
`pcman`	provides a booter that reads MS-DOS file system
`protoman`	provides a protocol management module
`restart`	provides a booter that causes a soft reboot of the system
`romboot`	provides a booter that allows booting from ROM
`rombreak`	provides a booter that calls the installed debugger
`rombug`	provides a low-level system debugger
`scsiman`	is a target-independent booter support module that provides general SCSI command protocol services
`sndp`	provides low-level system debug protocol
`srecord`	provides a booter that accepts S-Records
`swtimer`	provides timer services via software loops
`tsboot`	is a SCSI TEAC tape drive booter

`type41`	is a primary partition type
`vsboot`	is a SCSI archive viper tape drive booter

Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:_____

Company:_____

Phone:_____

Fax:_____Email:_____

Product Name:_____

Description of Problem:

_____

_____

_____

_____

_____

_____

_____

_____

_____

Host Platform_____

Target Platform_____

RadiSys.

MICROWARE SOFTWARE