OS-9 for SuperH 7709SE01/ 7709ASE01 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 SuperH.
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

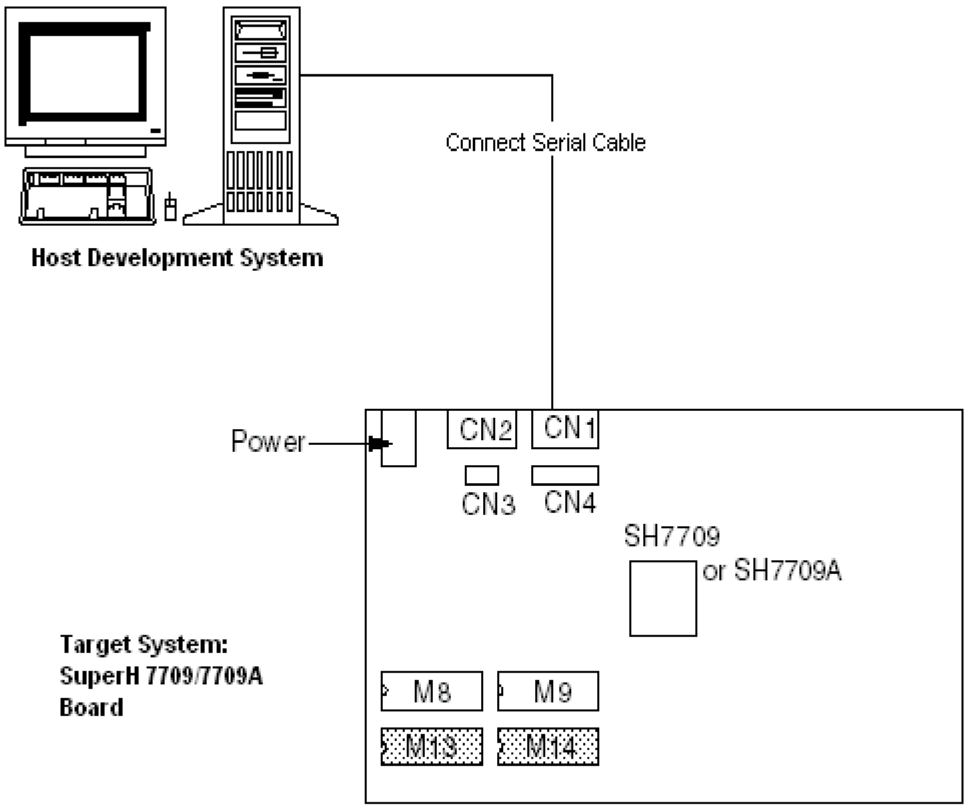
Chapter 1: Installing and Configuring OS-9

This chapter details the procedures for creating an OS-9 bootfile for the SuperH evaluation board. The following sections are included:

Development Environment Overview

Figure 1-1 shows a typical development environment for the SuperH board. The components include the minimum required to enable OS-9 to run on the SuperH 7709 and 7709A board.

Figure 1-1 SH7709/7709A Development Environment



Requirements and Compatibility

Host Hardware Requirements (PC Compatible)

Your host PC should have the following minimum hardware characteristics:

- a minimum of 32MB of free disk space (an additional 235MB of free disk space is required to run Java for OS-9)
- an Ethernet network card
- a PCMCIA card reader/writer
- the recommended amount of RAM for the host operating system

Host Software Requirements (PC Compatible)

- Windows 95, 98, ME, 2000, or NT
- Enhanced OS-9 for SH-3

Target Hardware Requirements

Your SuperH evaluation board requires the following hardware:

- enclosure or chassis with power supply
- an RS-232 null modem serial cable
- VGA display and serial mouse

Java Hardware Requirements

Your SuperH evaluation board must have the following to run Java for OS-9:

- 16MB of RAM
- 4MB of FLASH (Boot)
- VGA display and serial mouse

Software Compatibility

Enhanced OS-9 for SH-3 is compatible with the following software:

- Microware OS-9 for SH-3
- Microware Hawk
- Microware SoftStax

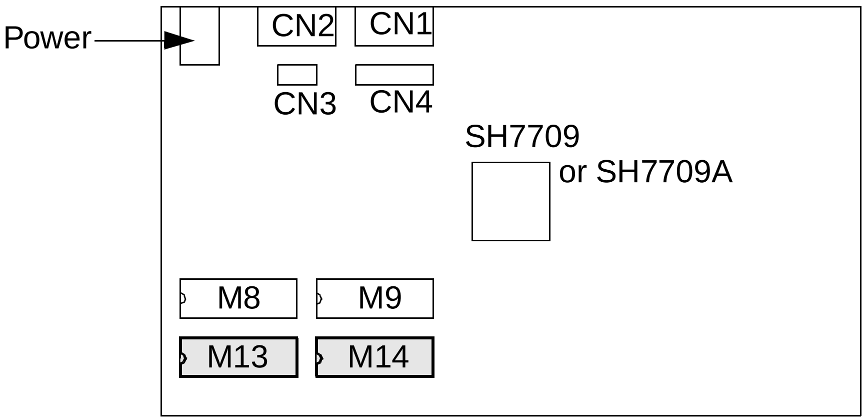
Target Hardware Setup

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

Installing the EPROMs

The first stage in configuring your SuperH evaluation board is to install the two pre-loaded EPROM devices that have been included in your ***Enhanced OS-9 for SuperH*** package. These devices include a coreboot system that has been pre-configured to get your board up and running quickly. Install the EPROM devices in sockets M13 (HIGH) and M14(LOW).

Figure 1-2 EPROM locations on the SuperH Evaluation Board



Note

If you need to program a new coreboot image into the flash devices on the SuperH evaluation board instead of using the coreboot image supplied on the EPROMs, see **Placing a Coreboot Image in FLASH Memory** on page 26.

Settings

The factory default settings for the DIP switches and jumpers may not work with OS-9. Be sure the DIP jumpers and switches agree with the following settings:

Jumpers on the SuperH 7709 Board

- J1 has pins 1 and 2 connected. (The default RTC power is from main power source--not CNB.)
- J2 has pins 1 and 2 open.
- J3 has pins 1 and 2 connected.
- J4 has pins 2 and 3 connected.
- J5 has pins 2 and 3 connected.
- J6 has all pins open (no jumpers).
- J7 as all pins open (no jumpers).
- J8 has pins 1 and 2 connected.
- J9 has pins 1 and 2 connected.
- J10 has pins 1 and 2 connected.
- J11 has pins 2 and 3 connected.
- J12 has pins 1 and 2 connected.

Jumpers on the HY7709CHK-I/O Board (default)

- J1 - all installed
- J22/3 Open - 1/4 Closed
- J32/3 Open - 1/4 Closed
- J4 - 1-2
- SW1 - All Open (Off)

External clock for serial ports

If you want to use an external clock the serial ports, you can select the frequency by setting J3 and J2.

J3 (SCI0)

Pins 1-4 shorted, Pins 2-3 shorted614.4 KHz (38.4 Kbps)

Pins 1-4 open, Pins 2-3 shorted307.2 KHz (19.2 Kbps)

Pins 1-4 shorted, Pins 2-3 open153.6 Khz (9.6 Kbps)

Pins 1-4 open, Pins 2-3 open 78.6 KHz (4.8 Kbps)

Switches

The switch settings described are not the factory defaults. They are switch settings that must be used to ensure OS-9 properly runs on the reference board. If a switch setting is not specified, use the factory default setting.

- SW3 has switches 1, 2, 3, 6 set to ONSwitch 6 on SW3 is factory set to OFF (little endian mode). It needs to be set to ON (big endian mode).
- SW4 has switches 1, 2 set to ON(Selects EPROM, Normal mode (No ICE) note: sw3 and sw4 are not used by OS-9 for baud rate).
- SW5 has switches 5, 6 set to ON.
- SW6 has switches 2,4,6 set to ONSelects the last byte of the MAC Address of the board.
- SW7 is not applicable (Reserved).
- SW8 is not applicable (Reserved).
- SW9 is not applicable (Reserved).

Connecting the Target to the Host

Note

Please refer to the Hitachi documentation for information on hardware preparation and installation, operating instructions, and functional descriptions prior to installing OS-9 on your SuperH evaluation board.

Note

The factory default settings for the DIP switches and jumpers may not work with OS-9. Be sure the DIP switches and jumpers agree with the settings in the **Settings** section.

Connecting the SuperH Evaluation Board to your host PC is a two-part process. The first part is to attach the serial cable between the board and the host PC. The second part is to set up the communication program. In this case, the communication program is HyperTerminal. HyperTerminal is supplied with Windows 95/98 on your PC.

Establishing a Serial Connection

Step 1. Connect the serial cable to the connector CN1 on the SuperH evaluation board.

Step 2. Connect the other end of the serial cable to one of the COM ports 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

If you do not know what type of serial cable your machine uses, try a reversed cable first. If the connection fails (no boot messages appear in the communication program's window), then try a straight serial cable.

Set Up HyperTerminal for Windows

Step 1. **For Windows 95/98:** From the `Start` button, select `Programs -> Accessories -> HyperTerminal` to open the HyperTerminal folder. Double-click the `Hyperterm` icon to start HyperTerminal.

For Windows NT: From the `Start` button, select `Programs -> Accessories -> HyperTerminal->Hyperterminal` to start Hyperterminal.

Step 2. Enter a name for your HyperTerminal session in the **Name** text box of the **Connection Description** dialog box.

Step 3. Select an icon for the new HyperTerminal session.

Step 4. **For Windows 95/98:** Click `OK`. The **Phone Number** dialog box appears.
For Windows NT: Click `OK`. The **Connect To** dialog box appears.

Step 5. **For Windows 95/98:** In the **Phone Number** dialog box, go to the **Connect Using** drop-down combo box and select the communications (COM) port that is connected to the SuperH evaluation board.

For Windows NT: In the **Connect To** dialog box, go to the **Connect Using** drop-down combo box and select the communications (COM) port that is connected to your SuperH evaluation board.

Step 6. Click `OK`. The **COM# Properties** dialog box appears (# represents the number of your chosen COM port, such as COM1).

Step 7. In the **Port Settings** tab, enter the settings as indicated in the following list:

- Bits per second: `9600`
- Data bits: `8`
- Parity: `None`
- Stop Bits: `1`
- Flow Control: `Xon/Xoff`

Step 8. Click `OK`. You are now connected to the SuperH Evaluation Board. The bottom left of your HyperTerminal screen will display the word *connected.*

Step 9. Turn on the SuperH evaluation board. A boot menu similar to the one in the following illustration will appear after boot messages are displayed.

Figure 1-3 OS-9 Boot Menu

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found

Now trying to Override autobooters

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

Boot from PCMCIA PCCARD ----------------
<pcm_pc>
Boot embedded OS-9000 in-place --------- <bo>
Copy embedded OS-9000 to RAM and boot -- <lr>
Enter system debugger ------------------
```

Now that you have connected your host system to the evaluation board, you will need to build a bootfile and place it on a PCMCIA IDE card.

Building the Bootfile Image

The following sections detail how to build the bootfile image using the Configuration Wizard.

Using the Configuration Wizard

The OS-9 coreboot image in the supplied EPROMs allows for booting from PCMCIA IDE cards. To boot from the PCMCIA IDE card, you need to create an OS-9 boot image in the PCMCIA card with the Configuration Wizard. The Configuration Wizard is a special purpose utility that simplifies the task of building an OS-9 ROM image for your SuperH evaluation board. The Configuration Wizard was installed on your host PC when you installed the Enhanced OS-9 for SuperH package.

Note

The SanDisk 4MB and 20MB PCMCIA ATA IDE cards do not work correctly with PCMCIA IDE interrupts. To make sure that all cards (including the SanDisk cards) work with OS-9, the default mode for accessing the PCMCIA IDE cards is set to polled mode. If you need to enable PCMCIA IDE interrupts, see **Enabling PCMCIA IDE Interrupts** on page 52.From the Windows 95/98 or Windows NT desktop, click `Start`. From the Windows 95/98 or Windows NT desktop, click `Start`.

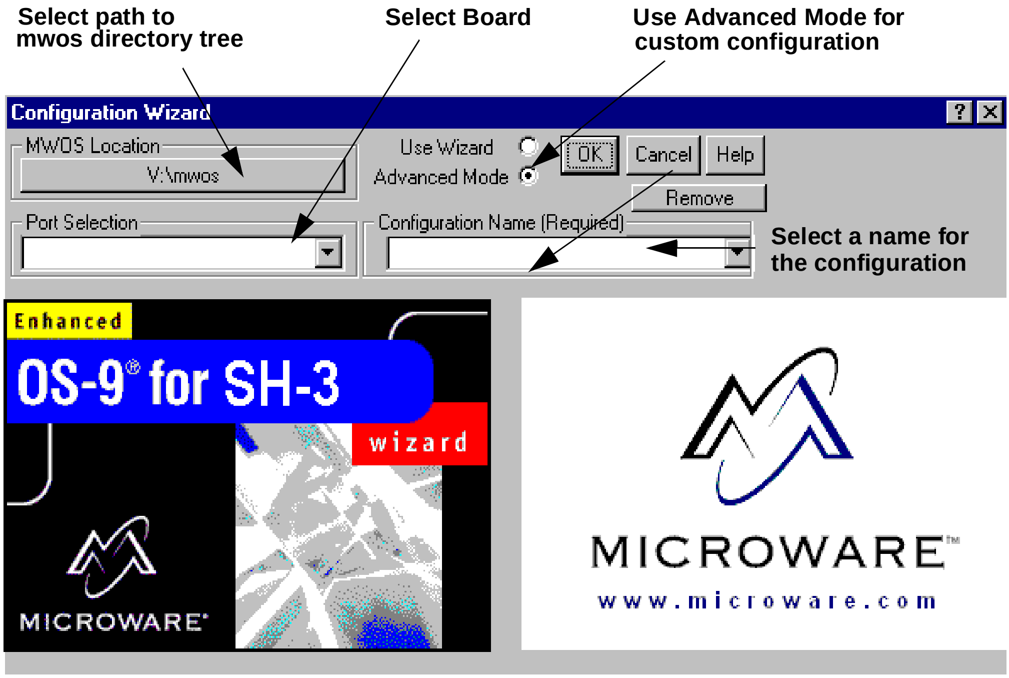
Note

The procedures in this section assume you want to set up your reference board for user state application development using Hawk and an Ethernet connection.From the Windows 95/98 or Windows NT desktop, click `Start`.From the Windows 95/98 or Windows NT desktop, click `Start`.

To use the Configuration Wizard, perform the following steps:

Step 1. Select `Start` -> `Programs` -> `Microware` -> `Enhanced OS9 for SH-3 v3.0.3` -> `Microware Configuration Wizard`. The window below appears:

Figure 1-4 SuperH Configuration Wizard



Step 2. Select the path where the MWOS directory structure can be located by clicking the MWOS location button.

Step 3. Select the target board from the **Port Selection** pull-down menu.

Step 4. Select a name for your configuration in the **Configuration Name** field. Your settings will be saved. This enables you to modify the ROM image incrementally, without having to reselect every option for each change.

Step 5. Select `Advanced Mode` and click `OK`. The Main Configuration window is displayed. Advanced mode enables you to make more detailed and specific choices about what modules are included in your ROM image.

Building the Bootfile Image

Once in the Advanced mode of the Configuration Wizard, build a bootfile image by completing the following steps:

Step 1. To enable the Ethernet function, click on the `System Network Configuration` button.

Step 2. Click on the `Interface` tab.

Step 3. Click on `Ethernet Connection` under the **Select Interface** option.

Step 4. Select `Specify an IP address` and enter the address information in the boxes below. For most situations you will need to fill out the following text boxes:

- `IP Address`
- `IP Broadcast Address`
- `Subnet Mask`
- `MAC Address` for your SuperH board

If you are unsure of these values, contact your system administrator.

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

Step 6. Make sure the Ethernet controller chip is displayed in the box under the note about the MAC address. 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 window.

Step 9. To enable the PCMCIA IDE function, click on the `Disk Configuration` button that is found in the bootfile Configuration button group.

Step 10. Click on the `IDE Configuration` tab and select `Enable IDE Disk`.

Step 11. Click `OK` to close the dialog box.

Step 12. Leave the other default settings alone and click the `Build Images` button to display the **Master Builder** window.

Step 13. Select the following check boxes:

- `Disk Support`
- `Disk Utilities [fdisk, format...]`
- `SoftStax (SPF) Support Modules`
- `User State Debugging Modules`

Step 14. Click `Bootfile Only Image` under **Build Type/ Options**, then click `Build`. This will build the bootfile image that can be placed on the PCMCIA IDE card.

Step 15. Insert the PCMCIA IDE card into the PCMCIA slot of your host computer.

Step 16. Click `Save As` to save the bootfile to the root directory of the PCMCIA IDE card. Name the bootfile os9kboot.

Step 17. Click `Finish` and then select `File` -> `Exit` to quit from Configuration Wizard. A dialog box appears and asks if you want to save your changes.

Step 18. Click `Yes` to save the changes, or click `No` to delete the changes. The Configuration Wizard program closes the dialog box and exits.

Step 19. Make sure the power to the board is turned off.

WARNING

If you insert a PCMCIA card into the PCMCIA socket of the SuperH evaluation board with power applied to the board, you will damage the PCMCIA card.

Step 20. Remove the PCMCIA IDE card from the host computer.

Step 21. Position the PCMCIA card so the end with the PCMCIA female connector is facing the PCMCIA socket and the label on the top of the card is facing down.

Step 22. Slide the card into the socket until the card snaps onto the connector pins and the eject button pops out.

Step 23. Apply power to the board. The following boot menu appears.

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found in socket 00
Now trying to Override autobooters

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

Boot over Ethernet (amd7990) ---------- <eb>
Boot from PCMCIA PCCARD --------------- <pcm_pc>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot-- <lr>
Kermit download ----------------------- <ker>
Enter system debugger ----------------- <break>
Restart the System -------------------- <q>

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

Step 24. Type `pcm_pc` to select `Boot from PCMCIA PCCARD`. The SuperH evaluation board will boot from the IDE PCMCIA card. You will see the shell prompt “$” at the end of the boot process.

Step 25. Type the following commands to start the NDP Server and the NDP I/O Server:

spfndpd <>>>/nil&
ndpio<>>>/nil&

The system is ready for you to start user state application development with Microware Hawk. See ***Getting Started with Microware Hawk*** to get oriented with Microware Hawk.

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 Enhanced OS-9 CD.

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

Example Startup File

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

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

For More Information

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

Chapter 2: Optional Procedures

The following sections detail the optional procedures you may wish to perform once you have installed and configured OS-9.

These procedures involve customizing the coreboot image. The main reason for changing the coreboot image is to take advantage of ROM Ethernet services such as System State Debugging. The System State Debugging limitation occurs because the IP address used in the EPROM image is set to `0.0.0.0`. If you want System State Debugging, you must create a new version of the coreboot image with an IP address assigned to the board.

Note

If you are only doing User State Debugging under SoftStax, changing the coreboot image is not necessary.

The following sections are included:

- **Placing a Coreboot Image in FLASH Memory**
- **Placing a Coreboot + Bootfile Image into FLASH Memory**
- **Making a Coreboot Image with an EPROM Programmer**
- **Making a Coreboot + Bootfile Image with an EPROM Programmer**

Placing a Coreboot Image in FLASH Memory

To put a coreboot image onto the SuperH board, you will have to build the image, embed it in a bootfile, and program it into flash memory.

Building the Coreboot Image

Step 1. Click `Start` on the Windows 95/98 or Windows NT desktop.

Step 2. Select `Programs -> Microware -> Enhanced OS-9 for SH-3 v3.0.3 -> Microware Configuration Wizard`. The following window appears:

Figure 2-1 Configuration Wizard

Step 3. Type a name in the **Configuration Name** text box. This allows you to save your Wizard settings for future reference.

Note
For subsequent uses of a configuration, the Configuration Wizard automatically adds the processor family to the beginning of the configuration name. Do not attempt to modify this portion of the name.

Step 4. Click `Advanced Mode` and click `OK`. The **SuperH Configuration Wizard** window appears.

Step 5. Click the `Main Configuration` button. A window titled **SH3:<configuration name>** appears.

Step 6. Click on the `Ethernet` tab and under the **Ethernet Setup** option enter the address information in the text boxes. For most situations you will need to fill out the following boxes:

- `IP Address`
- `IP Broadcast`
- `Subnet Mask`
- `MAC Address` for your SuperH board
- `IP Gateway`
  Your system administrator will know which values to enter in these boxes.

Step 7. Click `OK` to close the window.

Step 8. Click `Build Images` to display the **Master Builder** window.

Step 9. Click `Coreboot Only Image` setting and click `Build`. The coreboot image (pflashcore) is built and saved in the following directory: MWOS\0S9000\SH3\PORTS\SH7709\BOOTS\INSTALL\PORTBOOT

Step 10. Click Finish to dismiss the **Master Builder** window.

Embedding the Coreboot Image in a Bootfile

Step 1. In the SuperH Configuration Wizard window, click `Configure System Options`. A window named **SH3:<your configuration name>** appears.

Step 2. Click on the `Bootfile Options` tab.

Step 3. Select `PF-CORE`. This will include the file named `pflashcore` in the new bootfile.

Step 4. If there are any other bootfile options you want active at this time, select them as well. The Configuration Wizard help system explains the options. To access the help, click the question mark in the upper right corner of the option's dialog box and then select the option you want explained.

Step 5. Click `OK` to close the window.

Step 6. Click `Build Images` to open the **Master Builder** window.

Step 7. Click `Bootfile Only Image` under the **Build Type/Options**, then click `Build`. The bootfile image (bootfile) is built and saved in the following directory:
MWOS\0S9000\SH3\PORTS\SH7709\BOOTS\INSTALL\PORTBOOT
The **Save As** button is enabled when the build is completed.

Step 8. Save the bootfile to the root directory of the PCMCIA IDE card. Use the name os9kboot.

Step 9. Click `Finish` to close the **Master Builder** window, and select `File -> Save Settings` to save the configuration.

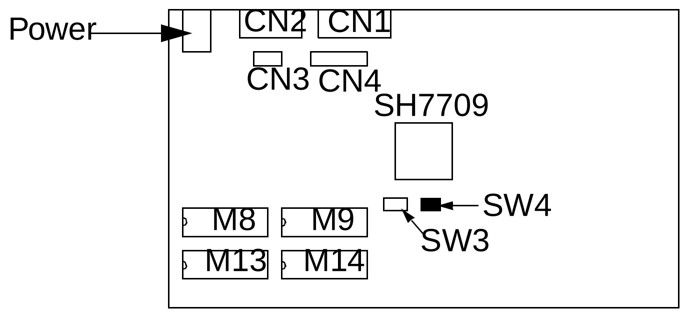
Step 10. Select `File -> Exit` to quit from the Wizard.

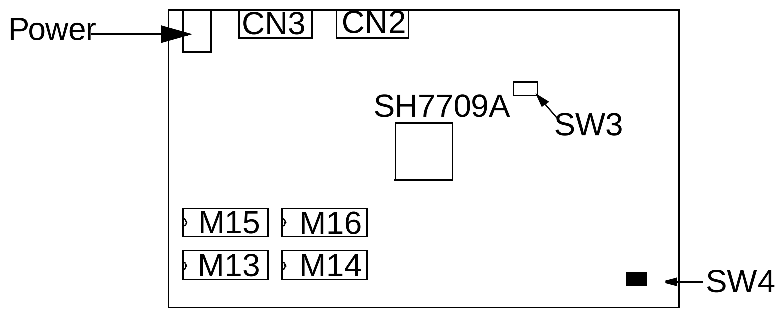
Programming the Coreboot Image into FLASH memory

Step 1. Remove power from the SuperH evaluation board.

Step 2. Locate the four-switch dip switch labeled SW4 on the SuperH evaluation board.

Figure 2-2 Location of Switch 4





Step 3. **SH7709:** Set switch SW4-1(switch 1 on SW4) to the ON position. This tells the system to boot from the EPROM instead of the flash memory.

SH7709A: Set switch SW4-3(switch 3 on SW4) to the ON position. This tells the system to boot from the EPROM instead of the flash memory.

Step 4. Remove the PCMCIA IDE card containing `os9kboot` from the PC host and insert the card into the PCMCIA socket on the SuperH board.

Step 5. Start **HyperTerminal**.

Step 6. Apply power to the SuperH evaluation board. The SuperH evaluation board will boot to the following boot menu.

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found in socket 00
Now trying to Override autobooters

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

Boot over Ethernet (amd7990) ---------- <eb>
Boot from PCMCIA PCCARD --------------- <pcm_pc>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot-- <lr>
Kermit download ----------------------- <ker>
Enter system debugger ----------------- <break>
Restart the System -------------------- <q>

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

Step 7. Type `pcm_pc` to finish booting with the bootfile on the PCMCIA IDE card. The new bootfile containing the coreboot image is now loaded into the SuperH evaluation board's RAM memory. You are ready to load the coreboot image into flash ROM.

Step 8. At the shell prompt ($), type the following command: `pflash`. This command erases flash memory, loads the new coreboot image into the flash memory and verifies the contents of the flash memory.

Step 9. When you get the shell prompt ($) again, remove power from the SuperH evaluation board.

Step 10. **SH7709:** Set switch 4-1 to the OFF position.

SH7709A: Set switch 4-3 to the OFF position.

Step 11. Reboot the system. The SuperH board is now using the new coreboot image in flash memory.

Now the system is ready for you to start system state and user state application development with Microware Hawk. See ***Getting Started with Microware Hawk*** to get oriented with Microware Hawk.

Placing a Coreboot + Bootfile Image into FLASH Memory

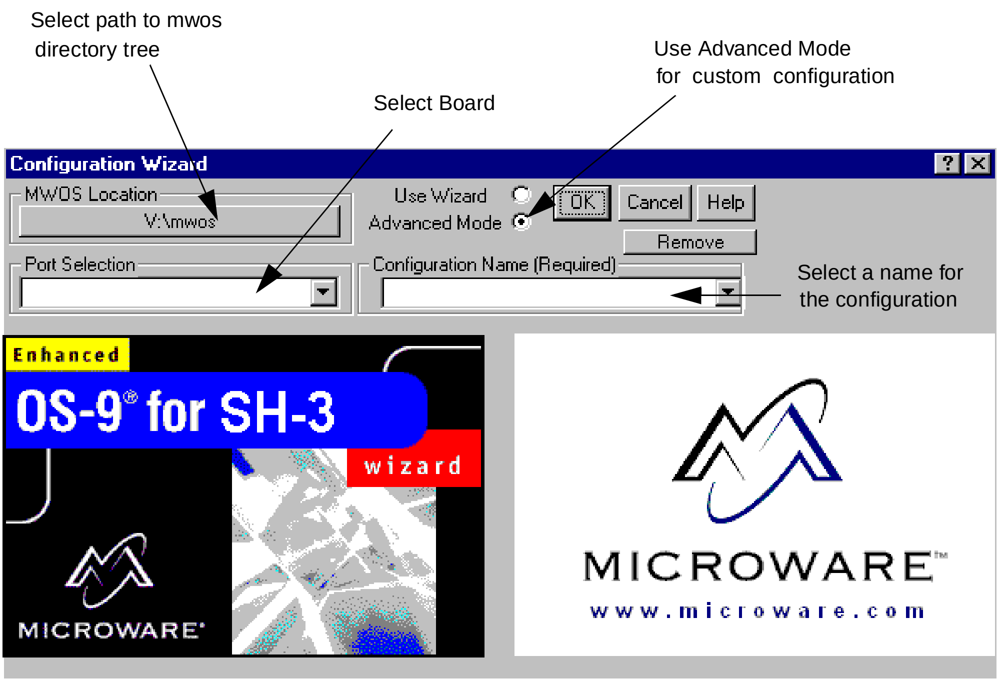
To put a Coreboot+Bootfile image onto the SuperH board, you will have to build the image, embed it in another bootfile to transfer it to the board, and store it in flash memory.

Building the Coreboot+Bootfile Image

Step 1. Click the `Start` button on the Windows 95/98 or Windows NT desktop.

Step 2. Select `Programs` -> `Microware` -> `Enhanced OS-9 for SH-3 v3.0.3` -> `Microware Configuration Wizard`. The following window appears:

Figure 2-3 SuperH Configuration Wizard



Step 3. Type a name in the **Configuration Name** text box. This allows you to save your wizard settings for future reference.

Note

For subsequent uses of a configuration, Configuration Wizard automatically adds the processor family to the beginning of the configuration name. Do not attempt to modify this portion of the name.

Step 4. Click `Advanced Mode` and click `OK`. The **SuperH Configuration Wizard** window appears.

Step 5. Click the `Main Configuration` button. A window titled **SH3:<configuration name>** appears.

Step 6. Click on the `Ethernet` tab and under the **Ethernet Setup** option enter the address information in the text boxes. For most situations you will need to fill out the following boxes:

- `IP Address`
- `IP Broadcast`
- `Subnet Mask`
- `MAC Address` for your SuperH board
- `IP Gateway`

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

Step 7. Click `OK` to close the window.

Step 8. Select any other coreboot or bootfile options you want included in your coreboot + bootfile image.

Step 9. Click `Build Images` to display the **Master Builder** window.

Step 10. Click `Coreboot + Bootfile` under the **Build Type/Options**, then click `Build`. The coreboot+bootfile image (`pflashrom`) is built and saved in the following directory: `MWOS\OS9000\SH3\PORTS\SH7709\BOOTS\INSTALL\PORTBOOT`

Step 11. Click `Finish` to dismiss the **Master Builder** window.

Embedding the Coreboot+Bootfile Image in a Bootfile

Step 1. In the **Configuration Wizard** window, click `Configure System Options`. A window named **SH3:<your configuration name>** appears.

Step 2. Click on the **Bootfile Options** tab.

Step 3. Click `PF-ROM`. This box includes the file named rom (which contains the coreboot + bootfile image) in the new bootfile as a data module.

Step 4. If there are any other bootfile options you want active at this time, select them as well.

Step 5. Click `OK` to close the window.

Step 6. Click `Build Images` to open the **Master Builder** window.

Step 7. Click `Bootfile Only Image` under **Build Type/ Options**, then click `Build`. The bootfile+coreboot image (bootfile) is built and saved in the following directory.
`MWOS\0S9000\SH3\PORTS\SH7709\BOOTS\INSTALL\PORTBOOT`
The **Save As** button is enabled when the build is completed.

Step 8. Save the bootfile to the root directory of the PCMCIA IDE card. Use the name os9kboot.

Step 9. Click `Finish` to close the **Master Builder** screen, and select `File -> Save Settings` to save the configuration.

Step 10. Select `File -> Exit` to quit from the Wizard.

Programming the Coreboot+Bootfile Image into FLASH memory

Step 1. Remove power from the SuperH evaluation board.

Step 2. Locate the dip-switch labled SW4 on the SuperH evaluation board.

Figure 2-4 Location of Switch 4 (SH7709)

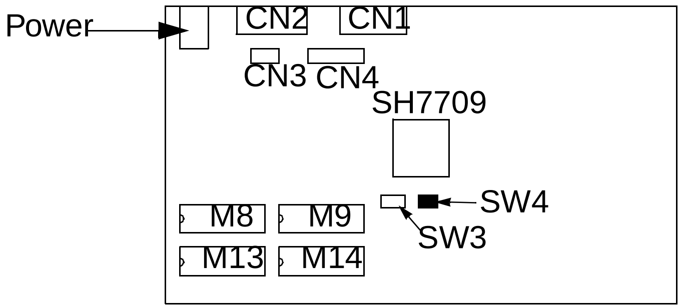
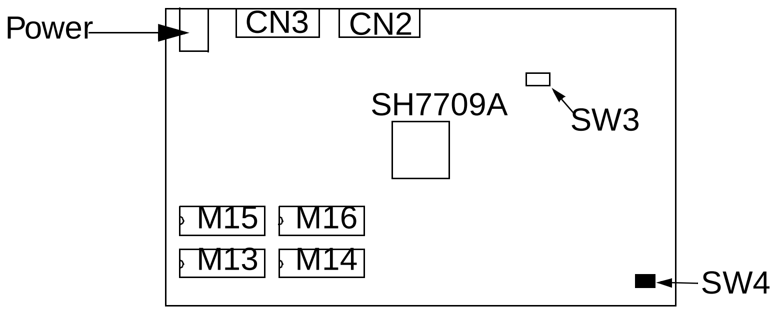


Figure 2-5 Location Switch 4 (SH7709A)



Step 3. **SH7709:** Set switch SW4-1 (switch 1 on SW4) to the ON position. This tells the system to boot from the EPROM instead of the flash memory.

SH7709A: Set switch SW4-3 (switch 3 on SW4) to the ON position. This tells the system to boot from the EPROM instead of the flash memory.

Step 4. Remove the PCMCIA IDE card from the PC host and insert the card into the PCMCIA socket on the SuperH board.

Step 5. Start **Hyperterminal**.

Step 6. Apply power to the board. The SuperH evaluation board will boot to the following boot menu.

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found in socket 00
Now trying to Override autobooters

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

Boot over Ethernet (amd7990) ---------- <eb>
Boot from PCMCIA PCCARD --------------- <pcm_pc>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot-- <lr>
Kermit download ----------------------- <ker>
Enter system debugger ----------------- <break>
Restart the System -------------------- <q>

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

Step 7. Type `pcm_pc` to finish booting with the bootfile on the IDE. The new bootfile containing the coreboot+bootfile image is now loaded into the SuperH evaluation board's RAM memory. You are ready to load the coreboot+bootfile image into flash ROM.

Step 8. At the shell prompt ($), type the following command: `pflash`. The command erases flash memory, loads the new coreboot+bootfile into the flash memory and verifies the contents of the flash memory.

Step 9. When you get the shell prompt ($) again, turn off the SuperH evaluation board.

Step 10. **SH7709:** Set switch SW4-1 to the OFF position.

SH7709A: Set switch SW4-3 to the OFF position.

Step 11. Reboot the system. The SuperH board is now using the new coreboot + bootfile image in flash memory.

Now the system is ready for you to start system state and user state application development with Microware Hawk. See ***Getting Started with Microware Hawk*** to get oriented with Microware Hawk.

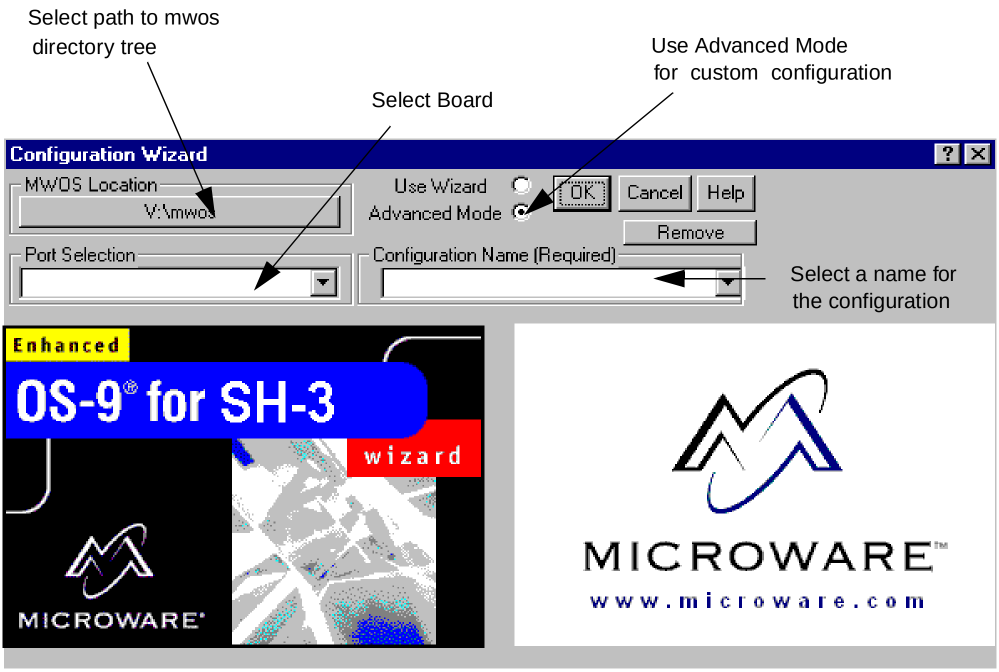
Making a Coreboot Image with an EPROM Programmer

This section tells you how to create the coreboot image up to the point where you will transfer the file to your EPROM programmer. Refer to the instructions for your EPROM programmer to learn how to program the new coreboot image into the EPROMS.

Step 1. Click the `Start` button on the Windows 95/98 or Windows NT desktop.

Step 2. Select `Programs -> Microware -> Enhanced OS-9 for SH-3 v3.0.3 -> Microware Configuration Wizard`. The following window appears:

Figure 2-6 SuperH Configuration Wizard



Step 3. Give the boot image a name in the **Configuration Name** text box. This allows you to save your wizard settings for future reference.

Note
For subsequent uses of a configuration, Configuration Wizard automatically adds the processor family to the beginning of the configuration name. Do not attempt to modify this portion of the name.

Step 4. Select `Advanced Mode` and click `OK`. The **SuperH Configuration Wizard** window appears.

Step 5. Click the `Main Configuration` button. A window titled **SH3:<configuration name>** appears.

Step 6. Click on the `Ethernet` tab and under the **Ethernet Setup** option enter the address information in the text boxes. For most situations you will need to fill out the following boxes:

- `IP Address`
- `IP Broadcast`
- `Subnet Mask`
- `IP Gateway`
- `MAC Address` for your SuperH board

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

Step 7. Click `OK` to close the window.

Step 8. Select any other coreboot options you want included in your coreboot image.

Step 9. Click `Build Images` to display the **Master Builder** window.

Step 10. Click `Coreboot Only Image` and click `Build`.

Step 11. Click `Save As` to save the coreboot image to a directory of your choosing. The default file name is coreboot.

Step 12. Click `Finish` to close the **Master Builder** window, and select `File -> Save Settings` to save the configuration.

Step 13. Select `File -> Exit` to quit from the Wizard.

Step 14. Transfer the coreboot image to the EPROMs with the EPROM programmer. You will need to follow the documentation for the EPROM programmer to complete this step.

Step 15. With the power to the board turned off, insert the EPROMs into the SuperH board.

Step 16. Set switch 4-1 (switch 1 on SW4) to the ON position so the board will boot from the EPROMs.

Step 17. Turn on power to the board. The SuperH evaluation board will boot to the boot menu.

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found in socket 00
Now trying to Override autobooters

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

Boot over Ethernet (amd7990) ---------- <eb>
Boot from PCMCIA PCCARD --------------- <pcm_pc>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot-- <lr>
Kermit download ----------------------- <ker>
Enter system debugger ----------------- <break>
Restart the System -------------------- <q>

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

Step 18. Select the booting method you want to use to boot the system to the dollar sign shell prompt ($).

Now the system is ready for you to start system state and user state application development with Microware Hawk. See ***Getting Started with Microware Hawk*** to get oriented with Microware Hawk.

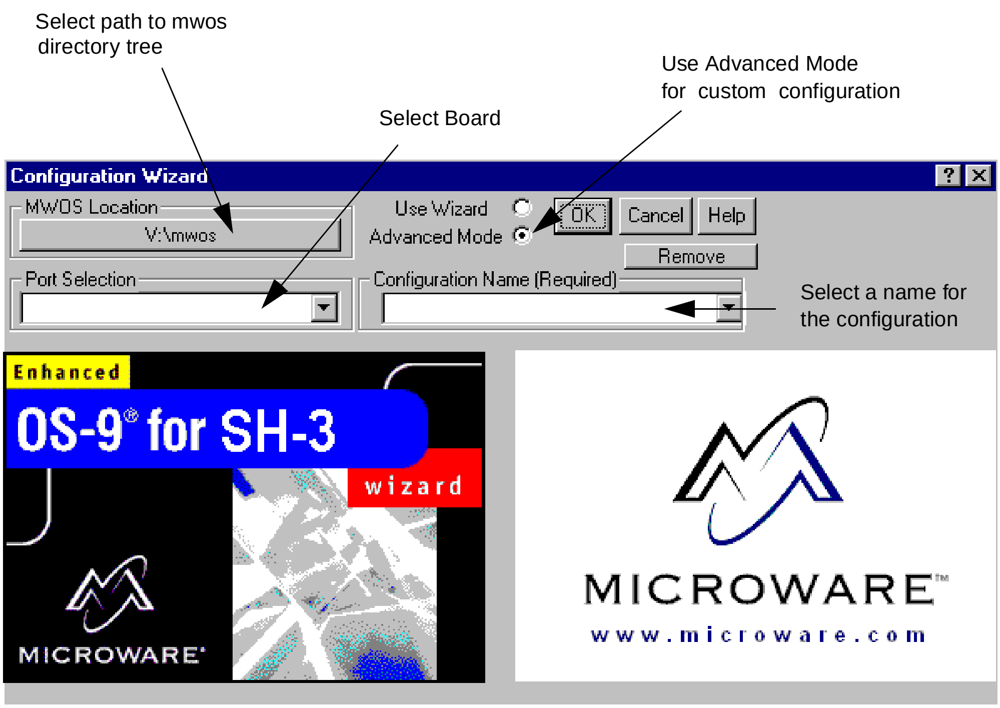
Making a Coreboot + Bootfile Image with an EPROM Programmer

We will tell you how to create the Coreboot+Bootfile image up to the point where you will transfer the file to your EPROM programmer. You will have to refer to your EPROM programmer's instructions to learn how to program the new Coreboot+Bootfile image into the EPROMS.

Step 1. From the Windows 95/98 or Windows NT desktop, click **Start**.

Step 2. Select `Programs -> Microware -> Enhanced OS-9 for SH-3 v3.0.3 -> Microware Configuration Wizard`. The following window appears:

Figure 2-7 SuperH Configuration Wizard



Step 3. Give the boot image a name in the **Configuration Name** text box. This allows you to save your wizard settings for future reference.

Note

For subsequent uses of a configuration, Configuration Wizard automatically adds the processor family to the beginning of the configuration name. Do not attempt to modify this portion of the name.

Step 4. Select Advanced Mode and click `OK`. The SuperH Configuration Wizard window appears.

Step 5. Click the `Main Configuration` button. A window titled **SH3:<configuration name>** appears.

Step 6. Click on the `Ethernet` tab and under the **Ethernet Setup** option enter the address information in the text boxes. For most situations you will need to fill out the following boxes:

- `IP Address`
- `IP Broadcast`
- `Subnet Mask`
- `IP Gateway`
- `MAC Address` for your SuperH board

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

Step 7. Click `OK` to close the window.

Step 8. Click on the **Bootfile Configuration** or the **Coreboot Configuration** buttons, and select the coreboot and bootfile options you want included in your coreboot + bootfile image.

Step 9. Click `Build Images` to display the **Master Builder** window.

Step 10. Click `Coreboot+Bootfile Image` and click `Build`.

Step 11. Make sure the combined coreboot+bootfile is not larger than your available EPROM memory. If it is too big, you will have to complete one of the following three actions:

- select the Pack ROM option
- turn off some of the bootfile options
- select the Pack ROM option and deselect some bootfile options

Step 12. Click `Save As` to save the coreboot+bootfile image to a directory of your choosing. If you do not have that directory on the drive, you can create it.

Step 13. Click `Finish` to close the **Master Builder** window.

Step 14. Select `File -> Save Settings` and `File -> Exit` to close Configuration Wizard.

Step 15. Transfer the coreboot+bootfile image to the EPROMS with the EPROM programmer. You will need to follow the documentation for the EPROM programmer to complete this step.

Step 16. With the power to the board turned off, insert the EPROMS into the SuperH board.

Step 17. **SH7709:** Set switch 4-1 (switch 1 on SW4) to the ON position so the board will boot from the EPROMS.

SH7709A: Set switch 4-3 (switch 3 on SW4) to the ON position so the board will boot from the EPROMS.

Step 18. Turn on power to the board. The SuperH evaluation board will boot to the boot menu.

```
OS-9000 Bootstrap for the SuperH

ATA IDE disk found in socket 00
Now trying to Override autobooters

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

Boot over Ethernet (amd7990) ---------- <eb>
Boot from PCMCIA PCCARD --------------- <pcm_pc>
Boot embedded OS-9000 in-place -------- <bo>
Copy embedded OS-9000 to RAM and boot-- <lr>
Kermit download ----------------------- <ker>
Enter system debugger ----------------- <break>
Restart the System -------------------- <q>

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

Step 19. Select the booting method you want to use to boot the system to the dollar sign shell prompt.

Now the system is ready for you to start system state and user state application development with Microware Hawk. See ***Getting Started with Microware Hawk*** to get oriented with Microware Hawk.

Chapter 3: Board Specific Reference

This chapter explains how to speed up boot times and enable PC Card interrupts. The chapter includes the following sections:

- **Boot Menu Options**
- **The Fastboot Enhancement**
- **Enabling PCMCIA IDE Interrupts**

Boot Menu Options

You select your boot device menu options using 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 Configuration Wizard, we recommend that you select only one device for autoboot on your system.

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

```
OS-9000 Bootstrap for the SuperH

MICROWARE PCMCIA SOCKET SERVICES
ATA IDE disk found.

Now trying to Override autobooters.

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

Boot over Ethernet (DP83902A) --------------- <eb>
Boot from PCMCIA PCCARD --------------------- <pcm_pc>
Boot embedded OS-9000 in place -------------- <bo>
Copy embedded OS-9000 to RAM and boot ------- <lr>
Enter system debugger ----------------------- <break>
Restart the system -------------------------- <q>
```

What you select for boot options in Configuration Wizard determines what modules are included in the coreboot image. The supported boot device for Enhanced OS-9 for SuperH is a PCMCIA card.

The Fastboot Enhancement

The Fastboot enhancements to OS-9 were added to address the needs of embedded systems that require faster system bootstrap performance than what is normally seen. OS-9's normal bootstrap performance can be mostly attributed to its flexibility. OS-9 can handle many different runtime configurations to which it dynamically adjusts during the bootstrap process.

The Fastboot concept consists of informing OS-9 that the defined configuration is static and valid. These assumptions eliminate the dynamic searching OS-9 normally performs during the bootstrap process and allow the system to perform 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 that a particular assumption can be made and that the associated bootstrap functionality should be omitted.

One very important feature of the Fastboot enhancement is that not only can the control flags be statically defined when the embedded system is initially configured, but they may also be dynamically altered during the bootstrap process itself. For example, the bootstrap code could be configured to query dip switch settings, respond to device interrupts, or respond to the presence of specific resources which would indicate different bootstrap requirements.

Also, the Fastboot enhancement's versatility allows for special considerations under certain circumstances. This versatility would be useful in a system where normally all resources are known, static and functional, but additional validation is required during bootstrap for a particular instance such as a resource failure. The low-level bootstrap code could respond to some form of user input that would inform it that additional checking and system verification is desired.

Implementation Overview

The Fastboot configuration flags have been implemented as a set of bit fields. An entire 32-bit field has been dedicated for bootstrap configuration. This four-byte field is contained within the set of data structures shared by the ModRom sub-components and the kernel. Hence, the field is available for modification and inspection by the entire set of system modules (high-level and low-level). Currently, there are just 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 below:

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. This causes the CRC check on modules to be omitted. This option is potentially a large time saver due to the complexity and expense of CRC generation. If a system has many modules in ROM, where access time is typically longer than RAM, omitting the CRC check on the modules will drastically decrease the bootstrap time. It is fairly rare that corruption of data occurs in ROM. Therefore, omitting CRC checking will usually be a safe option.

B_OKRAM

The B_OKRAM bit informs both the low-level and high-level systems that they should accept their respective RAM definitions without verification. Normally, the system probes memory during bootstrap based on the defined RAM parameters. This allows system designers to specify a possible RAM range which the system will validate upon startup. Thus the system can accommodate varying amounts of RAM. But 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 just like the *B_OKRAM* option except 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 accepts and uses the `init` module with the highest revision number. In a statically defined system, a good deal of time can be saved by using this option to omit the extended `init` module search.

B_NOIRQMASK

The B_NOIRQMASK bit informs the entire bootstrap system that it should 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. But some systems that have a well defined interrupt system (i.e. completely calmed by the `sysinit` hardware initialization code) and also have a requirement to respond to an installed interrupt handler during system startup can enable this option to prevent the ModRom and the kernel cold-start from disabling interrupts. This is particularly 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 at all or systems that 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 users can control the bootstrap speed of their 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. Users can redefine the macro for recompilation to create a new bootstrap configuration. The new over-riding value of the macro should be established by redefining the macro in the `rom_config.h` header file or as a macro definition parameter in the compilation command.

The `rom_config.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. Here is an example of how a user can redefine the bootstrap configuration of their system using the `BOOT_CONFIG` macro in the `rom_config.h` header file:

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

And here 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 would result in a bootstrap method which would accept the RAM and ROM definitions as they are without verification, and also validate 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 what 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;
```

Enabling PCMCIA IDE Interrupts

Due to a problem with losing interrupts when using certain PCMCIA IDE cards with the SuperH (SH7709 or SH7709A) board, the default configuration of OS-9 has been set to polled mode for accessing PCMCIA IDE type devices.

The following PCMCIA IDE cards are known to have problems with interrupts:

- the SanDisk PCMCIA PC CARD ATA 4MB card
- the SanDisk PCMCIA PC CARD ATA 20MB card

The following PCMCIA IDE cards have not shown any problems with interrupts:

- the Viking PCMCIA PC CARD ATA 12MB card
- the EXP Disk Traveler HDG-1.4GB card
- the Maxtor Hard Card series

All of the above cards (including the SanDisk cards) will work with polled mode. If you need to enable interrupts for use with your applications, you will need to follow the steps outlined in **Enabling PCMCIA IDE Interrupts on the SuperH**.

Before You Start

You need to test to see if your PCMCIA IDE card will work with PCMCIA interrupts enabled. If the following sequence of three commands work, then you can safely enable interrupts on your system.

```
$chd/mhc1
$save kernel
$ident kernel
```

Enabling PCMCIA IDE Interrupts on the SuperH

To enable interrupts on PCMCIA IDE devices the Microware Socket Services and device descriptors must be updated.

The Microware PCMCIA Socket Services are included in a p2module called `llcis` as well as in the `pcmcia` utility's module. Both of these modules should be compiled with interrupts enabled to use PCMCIA IDE interrupts.

Updating the `llcis` module

First you need to update the makefile for the `llcis` module.

Step 1. Change to the `LLCIS` directory. The `LLCIS` directory is found in the following path: `MWOS\OS9000\SH3\PORTS\SH7709\ROM\LLCIS`. `LLCIS` contains a file named `makefile`.

Step 2. Using a text editor, open `makefile`.

Step 3. Remove the '#' character from the following line:
`SPEC_COPTS = -dSINGLE_SOCKET # -dUSE_IRQ`

Step 4. Type `os9make` from the `LLCIS` directory to build a new `llcis` module.

Updating the PCMCIA utility

After you update the makefile for the `llcis` module, you need to update the makefile for the PCMCIA utility. The path to the PCMCIA utility's makefile is as follows:
`MWOS\OS9000\SH3\PORTS\SH7709\UTILS\PCMCIA\makefile`.

Step 1. Change to the PCMCIA directory.

Step 2. Using a text editor, open the file named `makefile`.

Step 3. Remove the '#' character from the following line:
`SPEC_COPTS =   -dSINGLE_SOCKET -k # -dUSE_IRQ`

Step 4. Type `os9make` from the PCMCIA directory to build a new `pcmcia` module.

Updating the RBF/PCF PCMCIA IDE device descriptors

After you update the modules `llcis` and `pcmcia`, you need to update the PCMCIA IDE device descriptors. The PCMCIA IDE device descriptors are found in the `config.des` file. The path to the `config.des` file is as follows:
`MWOS\OS9000\SH3\PORTS\SH7709\RBF\RB1003\config.des`

Step 1. Change to the `RB1003` directory.

Step 2. Using a text editor, open the file named `config.des`.

Step 3. Find the following section of code in the file:

```
init dev_specific {
    ds_idetype = IDE_TYPE_PCMCIA;
    ds_polled = IDE_POLLED;
    ds_altstat = HD_ALTSTAT;
    ds_timeout = 30;
};
```

Step 4. Change `IDE_POLLED` to `IDE_INTERRUPTS` in the following line:
`ds_polled = IDE_POLLED;`

Step 5. Change to the following directory:
`MWOS\OS9000\SH3\PORTS\SH7709\RBF\RB1003\DESC` directory

Step 6. Type `os9make`. This will build the RBF descriptors.

You have now enabled OS-9 to use PCMCIA IDE interrupts with the SH7709. Now you can create a new build using Configuration Wizard.

Note

If you are only creating a bootfile image, select `llcis` from the **Bootfile Options** tab in the wizard. This will allow the `llcis` **Microware Socket Services** to start after the system is booted. When using this option, you do not need to re-create the FLASH coreboot image.

Appendix A: Board Specific Modules

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

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

Low-Level System Modules

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

`MWOS/OS9000/SH3/PORTS/SH7709/CMDS/BOOTOBJS/ROM`

`cnfgdata`	standard `cnfgdata` module
`cnfgfunc`	configuration function module
`commcnfg`	communications port configuration module
`conscnfg`	console port configuration module
`io16550`	ROM based serial IO driver
`iosh7708`	ROM based serial IO driver
`iosh7709`	ROM based serial IO driver
`ll83902`	low-level ethernet ROM driver
`llcis`	SH7709 PCMCIA ROM services
	PCMCIA IDE and 3COM Ethernet support
`portmenu`	boot system support module
`sh3timer`	ROM timer services
`usedebug`	use debug support module
	allows system to enter ROMbug or `SNDP` on power-up if desired

High-Level System Modules

The following OS-9 system modules are tailored specifically for the SuperH 7709 board. Each module is located in a file of the same name in the following directory:

`MWOS/OS900/SH3/PORTS/7709/CMDS/BOOTOBJS`

`abort`	aborts the switch handler
`pwrext`	power management extension
`pwrplcy`	decides and initializes power states

PCMCIA Support for IDE Type Devices

Module

`rb1003`

Descriptors

`/hc1`	PCMCIA RBF type device RAW *
`/hc1.dd`	PCMCIA RBF type device, partition #1 *
`/hc1fmt`	PCMCIA RBF type device, partition #1 format enabled *
`/mhc1`	PCMCIA PC file system type device
`/mhc1.dd`	PCMCIA PC file system type device

Note

The Configuration Wizard does not support configuration of PCMCIA IDE card for use with RBF. For items marked with an * the PC file system is assumed.

Real Time Clock

Module

rtc7709

Ticker (System Clock) Support

Module

`tk7709`

Serial Support

Module

`sc7708`

Descriptors /term /t3

t3 is assigned to the sc7708. The connector is located on the HY7709CHK-I/0 expansion board. It is labeled as `RS232 Ch0 CN2`.

t3: serial port #3

Default Baud Rate: 9600

Default Parity: None

Default Data Bits: 8

To use it select the following: **7708p1 in the Configuration Wizard**.

Module

`sc7709`

Descriptors /term /t1

t1 is assigned to the sc7709. The connector is located at the rear of the board near the Ethernet connector. It is labeled as `CN1 SH7709 SCI`.

t1: serial port #1

Default Baud Rate: 9600

Default Parity: None

Default Data Bits: 8

Software/Hardware/Auto handshaking is supported.

To use it select the following: **7709p1 in the Configuration Wizard**.

Descriptors /t2

t2 is assigned to the sc7709. The connector is located on the HY7709CHK-I/O expansion board. It is labeled as `RS232 Ch1 CN3`.

t2: serial port #2

Default Baud Rate 9600

Default Parity: None

Default Data Bits: 8

To use it select the following: **7709p2 in the Configuration Wizard**.

Baud Rates

The following OS-9 baud rates are supported by the SC7709 driver:

Table 3-1 Supported SC7709 Baud Rates

50	75	110	134.5	150	300
600	1200	1800	2000	2400	3600
4800	7200	9600	19200	38400	

The following OS-9 baud rates are not supported by the SC7709 driver:

Table 3-2 SC7709 Baud Rates Not Supported

31250	56000	57600	64000	11520 0

Module

`sc16550`

Descriptors /term /t3

t3 is assigned to the SMC 37C935 16550 (compatible UART). The connector is located on the side of the board near the Ethernet connector. It is labeled as `CN3 COM1`.

t3: serial port #3

Default Baud Rate: 9600

Default Parity: None

Default Data Bits: 8

To use it: Select `16550 p1` in the Configuration Wizard.

Module

`sc16550`

Descriptors /term /t4

t4 is assigned to the SMC 37C935 16550 (compatible UART). The connector for t4 is located on an expansion board.

t4: serial port #4

Default Baud Rate: 9600

Default Parity: None

Default Data Bits: 8

To use it: Select `16550 p2` in the Configuration Wizard.

Baud Rates

The following OS-9 baud rates are supported by the sc16550 driver:

Table 3-3 Supported sc16550 Baud Rates

50	75	110	134.5	150	300
600	1200	1800	2000	2400	3600
4800	7200	9600	19200	38400	

The following OS-9 baud rates are not supported by the sc16550 driver:

Table 3-4 sc16550 Baud Rates Not Supported

31250	56000	57600	64000	115200

scsi710

Power Management Support

Module

sysif

Common System Modules List

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

`MWOS/OS9000/SH3/CMDS/BOOTOBJS/ROM`

`bootsys`	provides booter registration services.
`console`	provides console services.
`dbgentry`	inits debugger entry point for system use.
`dbserv`	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 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 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.

Appendix B: The SuperH Modules

This chapter is an overview of the **OS-9 for Embedded Systems** boot image and its components. Using Wizard eliminates the need to have an in-depth understanding of how to create and update an OS-9 boot image just to get started. This chapter explains the types of boot images created by Wizard and lists the OS-9 modules that are available for **OS-9 for Embedded Systems (SH7709)**.

Enhanced OS-9 Directory Map

The source code, example code, makefiles, and modules for Enhanced OS-9 are located in the directory structure as shown in **Figure 3-1**, **Figure 3-2** and **Figure 3-3**.

Figure 3-1 Enhanced OS-9 for SuperH Directories

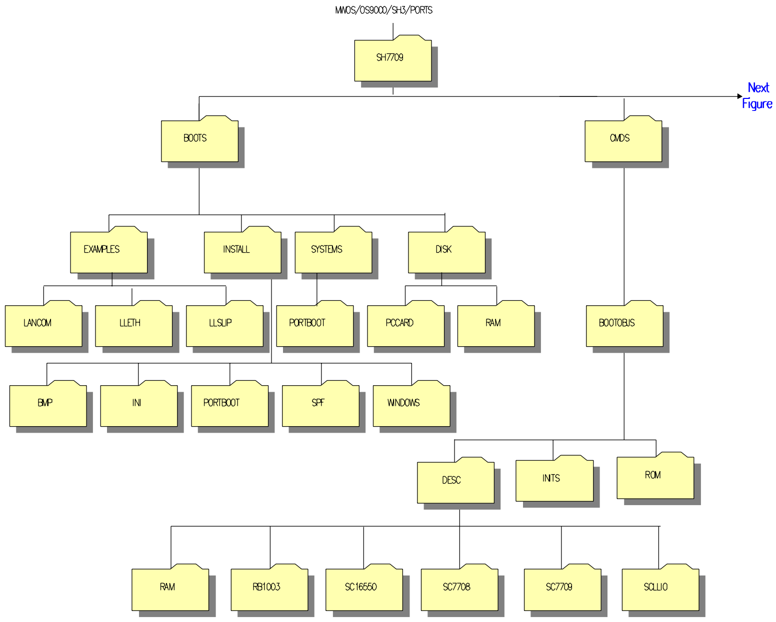


Figure 3-2 Enhanced OS-9 for SuperH Directories (continued)

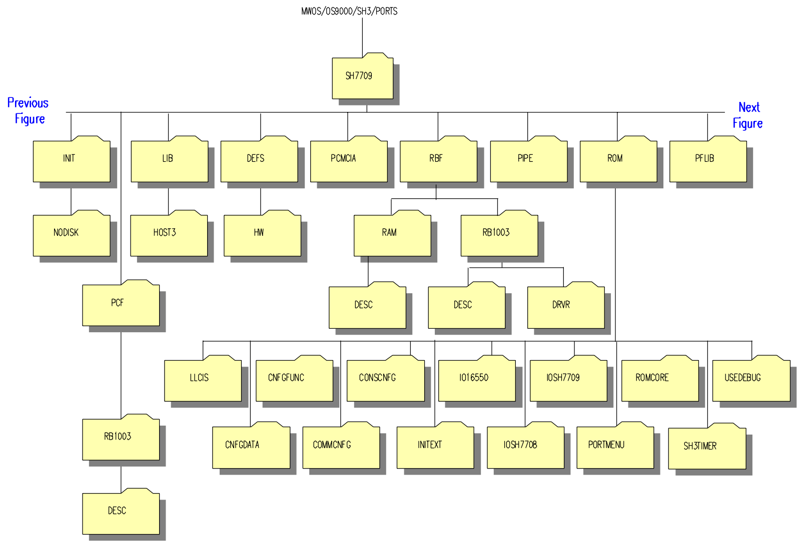
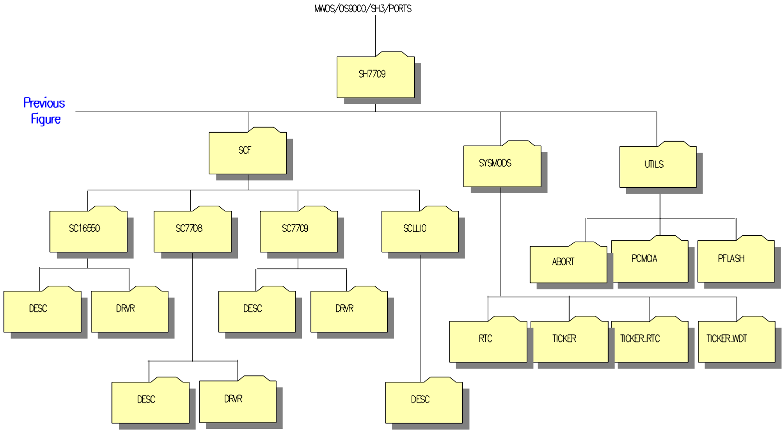


Figure 3-3 Enhanced OS-9 for SuperH Directories (continued)



Coreboot Module List

Table B-1 lists all of the coreboot modules available for the SH7709SE01 and SH7709ASE. The list is organized alphabetically. The modules do not necessarily load in this order, and each module is not necessarily used in every build.

Table B-1 Coreboot Image Modules

Module	Description
`bootsys`	module that provides booter services
`cnfgdata`	data module containing configuration parameters
`cnfgfunc`	module that retrieves configuration parameters from the `cnfgdata` module
`commcnfg`	module that retrieves the name of the low-level auxiliary communication port driver from the `cnfgdata` module
`conscnfg`	module that retrieves the name of the low-level console driver from the `cnfgdata` module
`console`	provides high-level I/O hooks into low-level console serial driver
`dbgentry`	debugger entry glue module
`dbgserv`	debugger server module
`excption`	exception services module

Table B-1 Coreboot Image Modules (continued)

Module	Description
`fdman`	RBF (Random Block File) floppy drive manager (RBF is the native OS-9 file system)
`flshcach`	module that provides the cache flushing routine
`ide`	low-level IDE booter module
`initext`	user-customizable system initialization module
`iosh7709`	low-level serial driver for SH7709 serial ports
`iosh7708`	low-level serial driver for SH7708 serial ports
`io16550`	low-level serial driver for com1/com2 serial ports
`ll21040`	low-level Ethernet driver module
`llbootp`	low-level BOOTP booter module
`llcis`	low-level PCMCIA configuration information service module
`llip`	low-level IP protocol module
`llkermit`	low-level Kermit protocol module
`llslip`	low-level SLIP protocol module

Table B-1 Coreboot Image Modules (continued)

Module	Description
`lltcp`	low-level TCP protocol module
`lludp`	low-level UDP protocol module
`notify`	module that coordinates use of low-level I/O drivers in system and user-state debugging
`override`	Target-independent booter module which enables overriding of the autobooter. If the space bar is pressed within three seconds after the booting the target, a boot menu is displayed. Otherwise, booting proceeds with the first autobooter.
`parser`	`parser` is called by the booters to parse the key fields from the `cnfgdata` module and the user input (user parameter fields) during system boot
`pcman`	PCF (PC File) floppy drive and partition logical translation
`portmenu`	retrieves a list of configured booter names from the ROM `cnfgdata` module
`protoman`	low-level protocol manager module
`restart`	system reset/restart module
`romboot`	booter module that locates the OS-9 bootfile in ROM, FLASH, or NVRAM

Table B-1 Coreboot Image Modules (continued)

Module	Description
`rombreak`	booter module that enables the break option in the boot menu (used to enter the debugger module)
`rombug`	RomBug debug client module
`romcore`	bootstrap code
`sh3timer`	simulated low-level timer module
`sndp`	FasTrak system-state debug client module
`usedebug`	debugger configuration module

Bootfile Module List

Table B-2 lists all of the bootfile modules available for the SH7709. The list is organized alphabetically. The modules do not necessarily load in this order, and each module is not necessarily used in every build.

Table B-2 Bootfile Image Modules

Module, Device Driver, File Manager or Descriptor	Description
`abort`	abort switch handler
`alias`	utility that assigns an alternate name to a device pathlist
`attr`	utility that examines or changes the security attributes (`<permissions>`) of the specified file(s).
`bootgen`	utility that builds and links a bootstrap file
`build`	utility that builds a text file from standard input
`cache`	module that enables the data cache
`configurer`	OS-9 initialization module
`copy`	Utility that copies data from one `file` to 136`another file`
`csl`	C shared library module

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`del`	utility that deletes the specified files
`deldir`	utility that deletes the specified data directory and the files (and subdirectories) it contains
`dir`	utility that displays a formatted list of file names from the specified directory
`dsave`	utility that copies a directory and its contents to another location
`fdisk`	utility that makes RBF (Random Block File Manager) disk partitions (not required for PCF IDE PC Cards)
`format`	utility that initializes the RBF (Random Block File Manager) file structure on a disk device (not required for PCF IDE PC Cards)
`fpu`	Floating point module for systems with math co-processor
`free`	utility that displays free space remaining on a mass-storage device

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`hc1`	hard disk device descriptor (partition 1)
`hc1.h0`	hc1 as the startup device (/h0)
`hc1fmt`	hc1 with formatting enabled
`hlproto`	protoman interface trap module for user-state connections
`iniz`	utility that initializes and link the device to the system
`ioman`	handles all I/O requests
`irqs`	utility that displays a list of the system's IRQ polling table
`kernel`	OS-9 kernel
`list`	utility that displays text lines from the specified path or paths (typically a file or files) to standard output.
`llcis`	low-level PCMCIA configuration information service module
`load`	utility that loads one or more specified modules into memory

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`makdir`	utility that creates a new directory
`mdir`	utility that lists a module directory
`mhc1`	device descriptor for PCMCIA IDE drive 0, partition 1 (for socket 0)
`mhc1.h0`	mhc1 descriptor as startup device (/h0)
`mshell`	an expanded command interpreter (it can be used in place of `shell`)
`ndpio`	user-state remote debugger module for use with spf (network daemon protocol for the I/O handler)
`nil`	device descriptor
`null`	device driver
`pcf`	PC File manager (MS-DOS devices)

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`pcmcia`	PCMCIA (PC Card) socket control manager pcmcia command (function:initialize SA-1100 PCMCIA socket Options: -i initialize socket(s) -d de-initialize socket(s) -v verbose mode -x dump CIS/Config information -s = socket: socket [default is all sockets] -? Print help message note: the -s option is not used in single socket systems.
pd	utility that shows the path from the root directory to the current data directory.

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`pflash`	utility that clears and programs flash memory on the target Options: -a erase all unlocked blocks -v verify only -m = <name> data module name default=‘romimg’ -?Õp = <addr> Flash start address -? display help message
`pflashcore`	module containing coreboot image
`pflashrom`	module containing a combined coreboot and bootfile image
`pipe`	pipe descriptor
`pipeman`	file manager for pipes
`pwrman`	power management module
`pwrplcy`	power management module (contains platform specific code)
`procs`	utility that shows the current process list

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`r0`	device descriptor for the Random Block File (RBF) RAM disk
`r0.dd`	device descriptor for the Random Block File (RBF) RAM disk as the default device
`ram`	device driver for ramdisk
`rb1003`	device driver for IDE hard drives
`rbf`	Random Block File (RBF) manager (OS-9 file system devices)
`RomBug`	RomBug debugger client module
`rtc7709`	real time clock module
`save`	utility that copies the specified module(s) from memory into the current data directory as files
`sc16550`	serial driver for the 16550 serial ports
`sc7708`	serial driver for the 7708 serial ports
`sc7709`	serial driver for the 7709 serial ports

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`scf`	file manager for Sequential Character File (SCF) devices
`shell`	the default command interpreter
`sleep`	utility that puts a running process to sleep for a specified amount of time
`sndp`	system-state debugging client
`spfndpd`	user-state remote debugger module for use with spf (network debugger protocol daemon)
`spfndpdc`	user-state remote debugger module for use with spf (network debugger protocol daemon for the server)
`ssm`	MMU module that provides processes with address space protection
`sysif`	power management module (provides a system specific interface to hardware components that do not have a device driver interface to OS-9)
`t1`	device descriptor for the 7709 port 1

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`t1_auto`	device descriptor for the 7709 port 1 (automatic CTS/RTS)
`t1_hw`	device descriptor for the 7709 port 1 (hardware flow control)
`t2`	device descriptor for 7709 port 2
`t3`	device descriptor for the 7708 port 1
`t4`	device descriptor for the 16550 port 1
`t5`	device descriptor for the 16550 port 2
`term1`	device descriptor for using the low-level console for high-level I/O through 7709 port 1
`term1_auto`	device descriptor for using the low-level console for high-level I/O through 7709 port 1 (automatic CTS/RTS)
`term1_hw`	device descriptor for using the low-level console for high-level I/O through 7709 port 1 (hardware flow control)

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`term2`	device descriptor for using the low-level console for high-level I/O through 7709 port 2
`term3`	device descriptor for using the low-level console for high-level I/O through the 7708 port 1
`term4`	device descriptor for using the low-level console for high-level I/O through the 16550 port 1
`term5`	device descriptor for using the low-level console for high-level I/O through the 16550 port 2
`tk7709`	system clock module
`transh3`	translation module for SH3
`undpd`	low-level user-state remote debugger module (network debugger protocol daemon)
`undpdc`	low-level user-state remote debugger module (network debugger protocol daemon for the server)
`vectsh3`	vector module for SH3

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager or Descriptor	Description
`bootsys`	module that provides booter services
`cnfgdata`	data module containing configuration parameters
`commcnfg`	module that retrieves the name of the low-level auxiliary communication port driver from the `cnfgdata` module
`conscnfg`	module that retrieves the name of the low-level console driver from the `cnfgdata` module
`console`	provides high-level I/O hooks into low-level console serial driver
`dbgentry`	module that provides hooks to low-level debugger server
`dbgserv`	debugger server module
`excption`	exception services module

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager,or Descriptor	Descriptor
`fdman`	RBF (Random Block File) floppy drive manager (RBF is the native OS-9 file system)
`flshcach`	module that provides the cache flushing routine
`ide`	low-level IDE booter module
`initext`	user-customizable system initialization module
`iosh7709`	low-level serial driver for SH7709 serial ports
`iosh7708`	low-level serial driver for SH7708 serial ports
`io16550`	low-level serial driver for com1/com2 serial ports
`ll21040`	low-level Ethernet driver module
`llbootp`	low-level BOOTP booter module
`llcis`	low-level PCMCIA configuration information service module
`llip`	low-level IP protocol module

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager,or Descriptor	Descriptor
`llkermit`	low-level Kermit protocol module
`llslip`	low-level SLIP protocol module
`lltcp`	low-level TCP protocol module
`lludp`	low-level UDP protocol module
`notify`	module that coordinates use of low-level I/O drivers in system and user-state debugging
`override`	Target-independent booter module which enables overriding of the autobooter. If the space bar is pressed within three seconds after the booting the target, a boot menu is displayed. Otherwise, booting proceeds with the first autobooter.
`parser`	`parser` is called by the booters to parse the key fields from the `cnfgdata` module and the user input (user parameter fields) during system boot
`pcman`	PCF (PC File) floppy drive manager

Table B-2 Bootfile Image Modules (continued)

Module, Device Driver, File Manager,or Descriptor	Descriptor
`portmenu`	retrieves a list of configured booter names from the ROM `cnfgdata` module
`protoman`	low-level protocol manager module
`restart`	booter module that restarts boot process
`romboot`	booter module that locates the OS-9 bootfile in ROM, FLASH, or NVRAM
`rombreak`	booter module that enables the break option in the boot menu (used to enter the debugger module)
`rombug`	RomBug debugger client module
`romcore`	bootstrap code
`sh3timer`	simulated low-level timer module
`sndp`	system state debug client module
`usedebug`	debugger configuration module

Path Descriptions

Table B-3 lists all of the coreboot module paths for the SH7709. The list is organized alphabetically. The modules do not necessarily load in this order, and each module is not necessarily used in every build. Table

Table B-3 Coreboot Paths

Module	Description
bootsys	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
cnfgdata	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
cnfgfunc	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOJS\ROM
commcnfg	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
conscnfg	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
console	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
dbgentry	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
dbgserv	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
excption	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-3 Coreboot Paths (continued)

Module	Description
fdman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
flshcach	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
ide	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
initext	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
iosh7709	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
iosh7708	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
io16550	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
llbootp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
llcis	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
llip	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
llkermit	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-3 Coreboot Paths (continued)

Module	Description
llslip	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
lltcp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
lludp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
notify	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
override	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
parser	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
pcman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
portmenu	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
protoman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
restart	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
romboot	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-3 Coreboot Paths (continued)

Module	Description
rombreak	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
rombug	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
romcore	C:\Mwos\OS9000\SH3\PORTS\BOOTOBJS\ROM
sh3timer	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
sndp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
usedebug	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM

B-4 lists all of the bootfile module paths for the SH7709 and Sh7709A. The list is organized alphabetically. The modules do not necessarily load in this order, and each module is not necessarily used in every build.

Table B-4 Bootfile Paths

Module, Device Driver, File Manager or Descriptor	Description
abort	C:\Mwos\OS9000\SH3\PORTS \SH7709\CMDS\BOOTOBJS
alias	C:\Mwos\OS9000\SH3\CMDS\ NOCSL
attr	C:\Mwos\OS9000\SH3\CMDS\ NOCSL
bootgen	C:\Mwos\OS9000\SH3\CMDS\ NOCSL
build	C:\Mwos\OS9000\SH3\CMDS\ NOCSL
cache	C:\Mwos\OS9000\SH3\CMDS\ BOOTOBJS
configurer	C:\Mwos\OS9000\SH3\PORTS \SH7709\ROM\CNFGDATA
copy	C:\Mwos\OS9000\SH3\CMDS
csl	C:\Mwos\OS9000\SH3\CMDS
del	C:\Mwos\OS9000\SH3\CMDS

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
deldir	C:\Mwos\OS9000\SH3\CMDS\NOCSL
dir	C:\Mwos\OS9000\SH3\CMDS
dsave	C:\Mwos\OS9000\SH3\CMDS\NOCSL
fdisk	C:\Mwos\OS9000\SH3\CMDS\NOCSL
format	C:\Mwos\OS9000\SH3\CMDS\NOCSL
free	C:\Mwos\OS9000\SH3\CMDS\NOCSL
hc1fmt	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\RB1003
hlproto	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
iniz	C:\Mwos\OS9000\SH3\CMDS
ioman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
irqs	C:\Mwos\OS9000\SH3\CMDS\NOCSL

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
kernel	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
list	C:\Mwos\OS9000\SH3\CMDS\NOCSL
llcis	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
load	C:\Mwos\OS9000\SH3\CMDS
makdir	C:\Mwos\OS9000\SH3\CMDS\NOCSL
mdir	C:\Mwos\OS9000\SH3\CMDS
mhc1	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\RB1003
mhc1.h0	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\RB1003
mshell	C:\Mwos\OS9000\SH3\CMDS\NOCSL
ndpio	C:\Mwos\OS9000\SH3\CMDS
nil	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
null	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
pcf	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
pcmcia	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS
pd	C:\Mwos\OS9000\SH3\CMDS
pflash	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS
pipe	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC
pipeman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
pwrman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
pwrplcy	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
procs	C:\Mwos\OS9000\SH3\CMDS\NOCSL

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
r0	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\RAM
r0.dd	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\RAM
ram	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTS\DISK
rb1003	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
rbf	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
RomBug	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
rtc7709	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
save	C:\Mwos\OS9000\SH3\CMDS\NOCSL
sc16550	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
sc7708	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
sc7709	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
scf	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
shell	C:\Mwos\OS9000\SH3\CMDS
sleep	C:\Mwos\OS9000\SH3\CMDS
sndp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
spfndpd	C:\Mwos\OS9000\SH3\CMDS
spfndpdc	C:\Mwos\OS9000\SH3\CMDS
ssm	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
sysif	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
t1	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
t1_auto	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
t1_hw	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
t2	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
t3	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77008
t4	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC16550
t5	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC16550
term1	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
term1_auto	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
term1_hw	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
term2	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77009
term3	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC77008
term4	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC16550
term5	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\DESC\SC16550
tk7709	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS
transh3	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
undpd	C:\Mwos\OS9000\SH3\CMDS
undpdc	C:\Mwos\OS9000\SH3\CMDS
vectsh3	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS
bootsys	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
cnfgdata	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
commcnfg	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
conscnfg	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
console	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
dbgentry	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
dbgserv	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
excption	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
fdman	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
flshcach	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
ide	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
initext	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
iosh7709	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
iosh7708	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
io16550	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
llbootp	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
llcis	C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM
llip	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM
llkermit	C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
`llslip`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`lltcp`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`lludp`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`notify`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`override`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`parser`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`pcman`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`portmenu`	`C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM`
`protoman`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`restart`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`

Table B-4 Bootfile Paths (continued)

Module, Device Driver, File Manager or Descriptor	Description
`romboot`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`rombreak`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`rombug`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`romcore`	`C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM`
`sh3timer`	`C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM`
`sndp`	`C:\Mwos\OS9000\SH3\CMDS\BOOTOBJS\ROM`
`usedebug`	`C:\Mwos\OS9000\SH3\PORTS\SH7709\CMDS\BOOTOBJS\ROM`

Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:____

Company:_______________________________________

Phone:____

Fax:____________________Email:____________________

Product Name:
Enhanced OS-9 for Super H 7709SE01/7709ASE01 Board Guide

Description of Problem:

Host
Platform____

Target
Platform____

RadiSys.
MICROWARE SOFTWARE