Home

OS-9® for Toshiba JMR-TX3927 Board Guide

Version 4.7

RadiSys

THE POWER OF WE

www.radisys.com

Revision A • July 2006

Contents

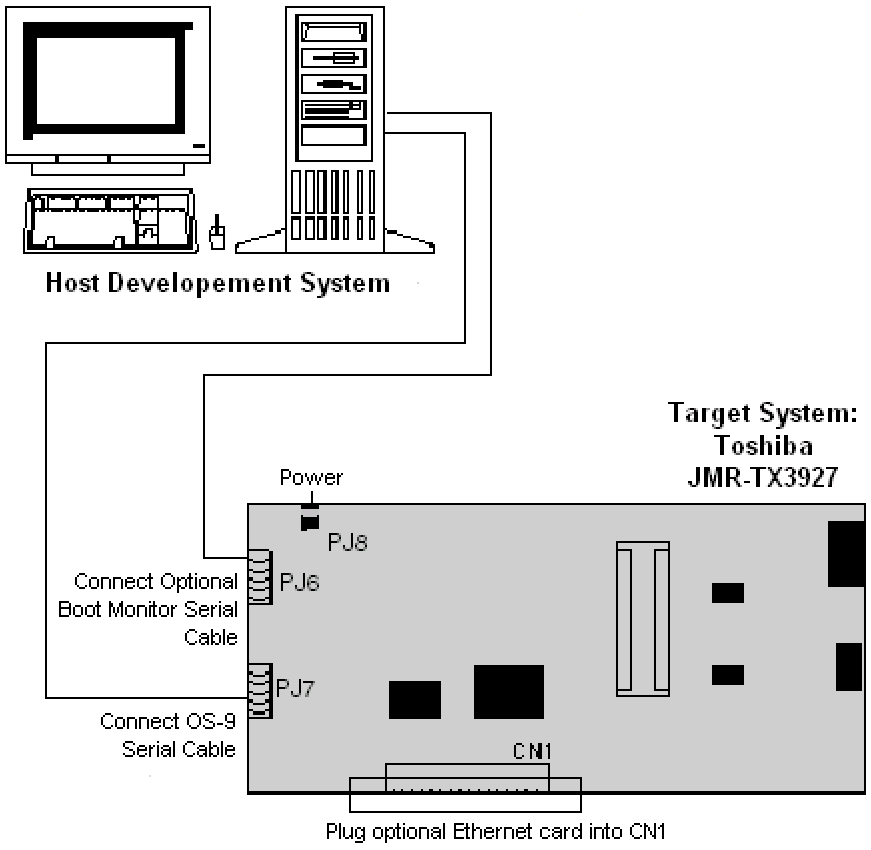
1 Installing and Configuring OS-9®

This chapter describes installing and configuring OS-9® on the Toshiba JMR-TX3927 board. It includes the following sections:

Development Environment Overview

Figure 1-1 shows a typical development environment for the Toshiba JMR-TX3927 board. The components shown include the minimum required to enable OS-9 to run on the board.

Figure 1-1. Toshiba JMR-TX3927 Development Environment


Host Developement System
Target System: Toshiba JMR-TX3927
Power
PJ8
PJ6
Connect Optional Boot Monitor Serial Cable
PJ7
Connect OS-9 Serial Cable
CN1
Plug optional Ethernet card into CN1

Requirements and Compatibility

Before you begin, install the *Microware OS-9 for MIPS 3000 CD-ROM* on your host PC.

Host Hardware Requirements (PC Compatible)

Your host PC must have the following minimum hardware characteristics:

- 32MB of RAM

Host Software Requirements (PC Compatible)

Your host PC must have the following software installed:

- Microware OS-9
- Windows 95, 98, ME, 2000, or NT

Target Hardware Requirements

Your JMR-TX3927 board requires the following hardware:

- a power supply
- an RS-232 null modem serial cable (for serial console)

In addition to the above requirements, you may want to have a second RS-232 null modem serial cable and an Ethernet cable and card. (Supported Ethernet cards include the Intel PRO/100 and PCI NE2000 compatible cards.)

Software Compatibility

Microware OS-9 for MIPS is compatible with the following Microware software:

- Hawk™
- SoftStax®
- LAN Communications

Target and Host Setup

Before installing and configuring OS-9 on your JMR-TX3927 evaluation board, refer to the hardware documentation for information on hardware setup.

Settings

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

Switch Configuration to Run OS-9

S2.2 set to OFF	TLBs enabled
S5.1 set to ON	CE0 connected to FLASH bank 1

S5.2 set to OFF	CE1 not connected to FLASH bank 1
S5.3 set to OFF	CE0 not connected to FLASH bank 2
S5.4 set to OFF	CE1 not connected to FLASH bank 2
S5.5 set to OFF	CE0 not connected to FLASH bank 3
S5.6 set to OFF	CE1 not connected to FLASH bank 3
S5.7 set to OFF	CE0 not connected to EPROM bank
S5.8 set to OFF	boot with 32-bit bus

Switch Configuration to Run HCP5 Monitor

S2.2 set to OFF or ON	TLBS enabled or disabled
S5.1 set to OFF	CE0 not connected to FLASH bank 1
S5.2 set to ON	CE1 connected to FLASH bank 1
S5.3 set to OFF	CE0 not connected to FLASH bank 2
S5.4 set to OFF	CE1 not connected to FLASH bank 2
S5.5 set to ON	CE0 connected to FLASH bank 3
S5.6 set to OFF	CE1 not connected to FLASH bank 3
S5.7 set to OFF	CE0 not connected to EPROM bank
S5.8 set to ON	boot with 16-bit bus

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, be certain to follow its directions and specifications.

A TFTP server is optional for the JMR-TX3927 board.

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

1. Load the product CD into the host machine's CD-ROM drive and allow the CD autorun. The installer's dialog box appears.
2. Select `Walusoft TFTPServer32Pro` from the installer menu and follow the directions to load the server onto your host machine.
3. Start TTFTPServer32Pro by selecting `Start -> Programs -> TFTPServer-TFTPServer32`. The Walusoft TFTP Server32 Pro splash screen appears. From here you will need to set the path to the outbound folder.
4. Select `System -> Setup` to display the Server Options dialog box.
5. Select the `Outbound` tab and enter the following path in the Outbound file path text box:

```
<DRIVE>:\MWOS\OS9000\MIPS3000\PORTS\JMRTX3927\BOOTS\INSTA
LL\PORTBOOT
```

6. Click `OK` to accept the path. The TFTP server is now configured for use with the Configuration Wizard and OS-9.
7. Minimize the TFTP server window to let the server run in the background.

Connecting the Target to the Host

Connecting the JMR-TX3927 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

To attach the cables, complete the following steps:

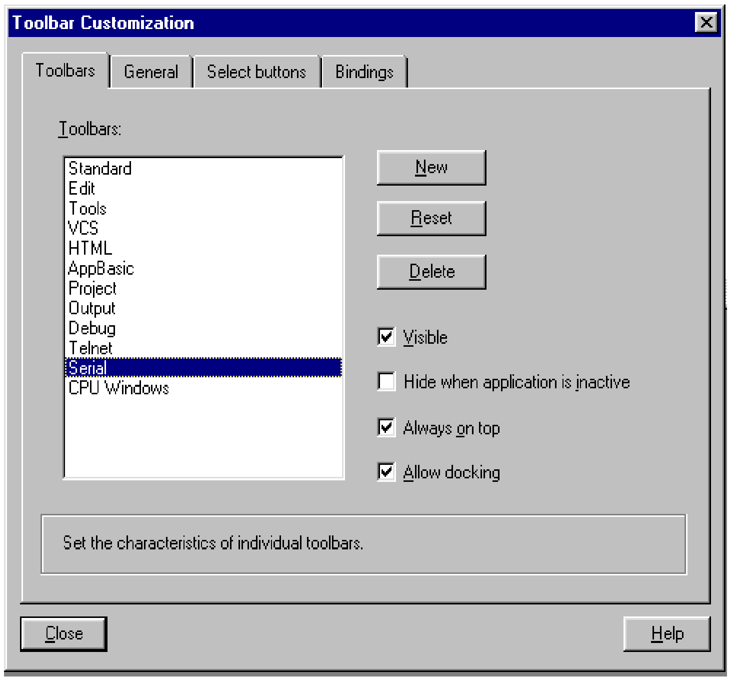
1. Attach the power connector to PJ8.
2. Attach the serial cable to PJ6 (RS-232 C-1). (The HCP5 monitor is on PJ6; it will be used to load the flash image.)
3. Connect the optional serial cable to PJ7. (The default OS-9 console is on PJ7.) If you are only using one cable, you will need to switch it after the flash image is installed.
4. If you are using an Ethernet card, plug it into CN1.

Booting to the Boot Menu

It will be necessary to boot to the HCP5 prompt in order to verify that your serial cable is connected properly. To do this, complete the following steps:

1. From the desktop, click Start and select `RadiSys -> Microware OS-9 for MIPS3000 -> Hawk IDE` to start Hawk.
2. If the Serial console window is not open, it can be opened from the Toolbar Customization dialog (shown in Figure 1-2).
   (Select `Tools -> Customize -> Toolbars` to open the Toolbar Customization dialog.)

Figure 1-2. Toolbar Customization dialog



Toolbars:

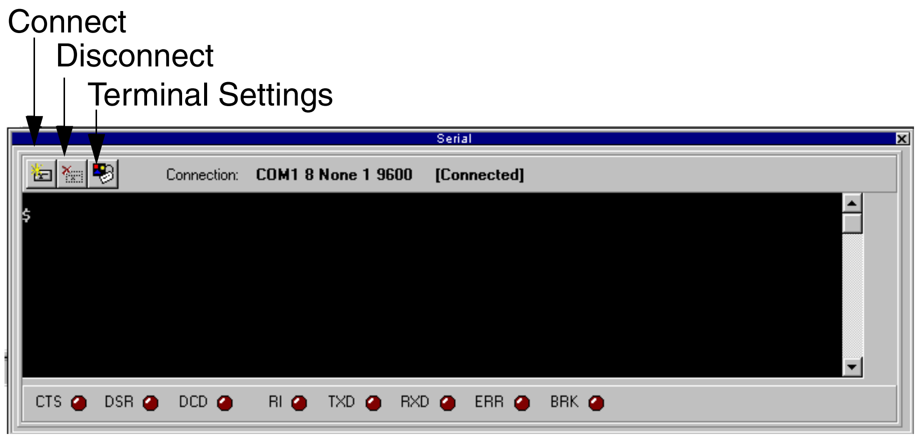
Standard
Edit
Tools
VCS
HTML
AppBasic
Project
Output
Debug
Telnet
Serial
CPU Windows

[✓] Visible
[ ] Hide when application is inactive
[✓] Always on top
[✓] Allow docking

Set the characteristics of individual toolbars.

3. Once the dialog is open, select `Serial` in the Toolbars list box.
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



Connection: COM1 8 None 1 9600 [Connected]

```
$
```

5. Once you have the Serial Console window open, click the `Connect` button in the upper left corner of the window. The Com Port Options dialog box appears.
6. Click the `OK` button. The message *[Not Connected]* should change to *[Connected]*.
7. Apply power to the board. The HCP5 boots the board.

Building the OS-9 ROM Image

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

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

Coreboot

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

Bootfile

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

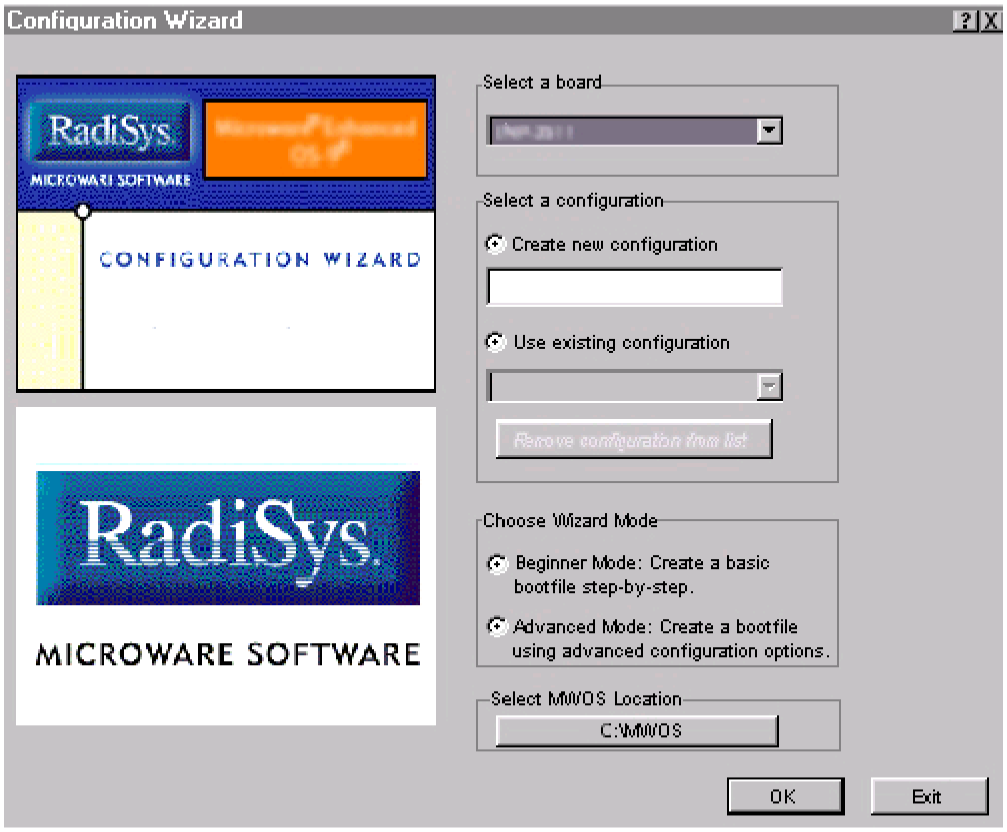
Microware provides a Configuration Wizard to create a coreboot image, a bootfile image, or an entire OS-9 ROM Image. The wizard can also be used to modify an existing image. The Configuration Wizard is automatically installed on your host PC during the OS-9 installation process.

Starting the Configuration Wizard

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

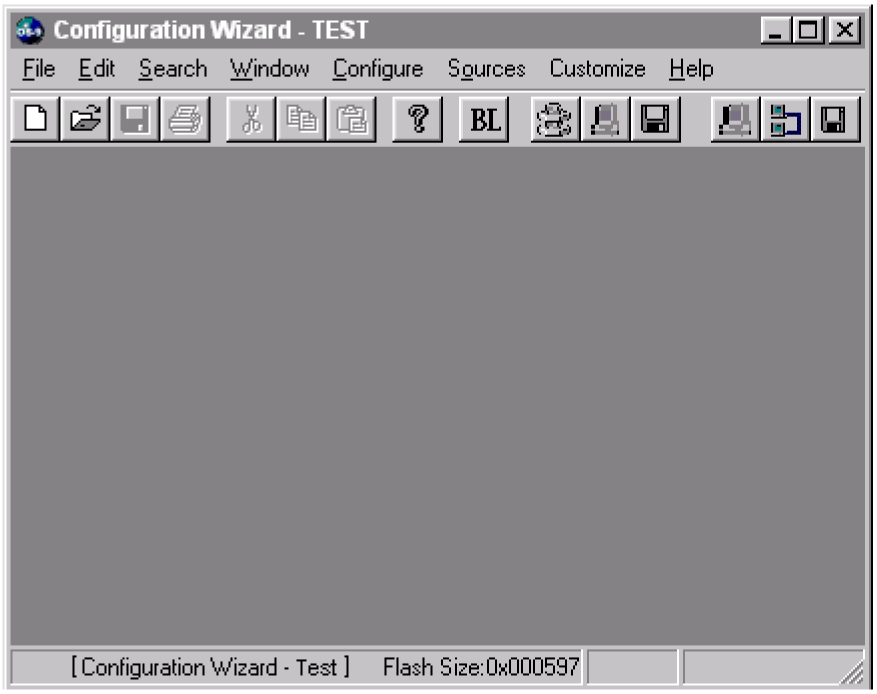
1. From the Windows desktop, select `Start` -> `RadiSys -> Microware OS-9 for <product> -> Configuration Wizard`. You should see the following opening screen:

Figure 1-4. Configuration Wizard Opening Screen



2. Select your target board from the Select a board pull-down menu.
3. Select the `Create new configuration` radio button from the Select a configuration menu and type in the name you want to give your ROM image in the supplied text box. This names your new configuration, which can later be accessed by selecting the Use existing configuration pull down menu.
4. Select the `Advanced Mode` radio button from the Choose Wizard Mode field and click `OK`. The Wizard's main window is displayed. This is the dialog from which you will proceed to build your image. An example is shown in Figure 1-5.

Figure 1-5. Configuration Wizard Main Window


Configuration Wizard - TEST
File Edit Search Window Configure Sources Customize Help
BL
[Configuration Wizard - Test]
Flash Size:0x000597

Configuring Coreboot and Bootfile Options

The only steps necessary in configuring the coreboot and bootfile options involve filling in the Ethernet information. All other default settings in the Configuration Wizard are correct for the JMR-TX3927 board.

If you do not have Ethernet installed on your system, you may skip the sections on Ethernet setup and proceed directly to the Building the Coreboot + Bootfile Image section.

Ethernet Configuration (Coreboot)

To configure the Ethernet settings from the Configuration Wizard, complete the following steps:

1. From the Main Configuration window, select `Configure` -> `Coreboot` -> `Main configuration`.
2. Select the `Ethernet` tab.
3. Enter the Ethernet address information in the address text boxes. This includes the following information:
    - IP Address
    - IP Broadcast
    - Subnet Mask
    - IP Gateway

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

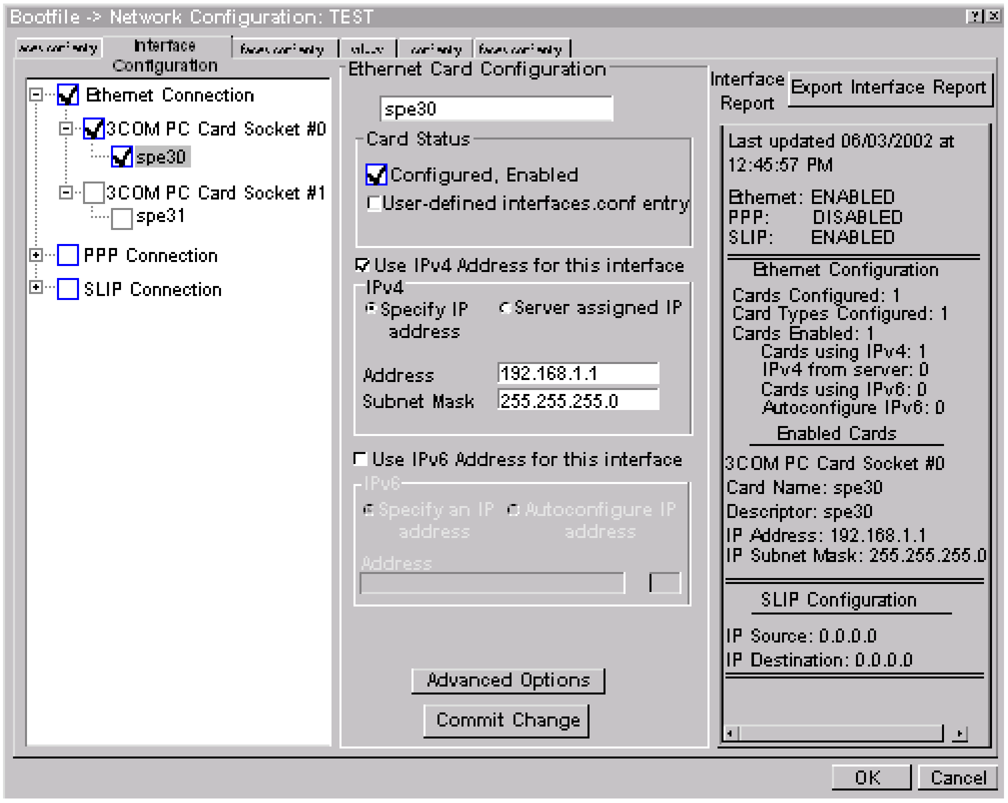
4. Select the appropriate Ethernet card in the drop down menu box at the bottom of the screen. (This should be either NE2000 or Intel PRO 100.)
5. Check the Add to Boot Menu check box under the Ethernet Boot Options.
6. Click `OK` to close the window.

Ethernet Configuration (Bootfile)

To configure the Ethernet settings from the Configuration Wizard, complete the following steps:

1. If you want to use the target board across a network, you will need to configure the Ethernet settings within the Configuration Wizard. To do this, select `Configure -> Bootfile -> Network Configuration` from the Wizard's main menu.
2. From the Network Configuration dialog, select the `Interface Configuration` tab. From here you can select and enable the interface. For example, you can select the appropriate Ethernet card from the list of options on the left and specify whether you would like to enable IPv4 or IPv6 addressing. Figure 1-6 shows an example of the Interface Configuration tab.

Figure 1-6. Bootfile -> Network Configuration -> Interface Configuration



Bootfile -> Network Configuration: TEST

Interface Configuration

- ☑ Ethernet Connection
  - ☑ 3COM PC Card Socket #0
    - ☑ spe30
  - ☐ 3COM PC Card Socket #1
    - ☐ spe31
- ☐ PPP Connection
- ☐ SLIP Connection

Ethernet Card Configuration

spe30

Card Status
☑ Configured, Enabled
☐ User-defined interfaces.conf entry

☑ Use IPv4 Address for this interface
IPv4
[✓ Specify IP address] Server assigned IP
Address 192.168.1.1
Subnet Mask 255.255.255.0

☐ Use IPv6 Address for this interface
IPv6
[✓ Specify an IP address] Autoconfigure IP address
Address

Advanced Options
Commit Change

Interface Report
Export Interface Report

```
Last updated 06/03/2002 at
12:45:57 PM

Ethernet: ENABLED
PPP:      DISABLED
SLIP:     ENABLED

Ethernet Configuration
Cards Configured: 1
Card Types Configured: 1
Cards Enabled: 1
    Cards using IPv4: 1
    IPv4 from server: 0
    Cards using IPv6: 0
    Autoconfigure IPv6: 0
Enabled Cards
3COM PC Card Socket #0
Card Name: spe30
Descriptor: spe30
IP Address: 192.168.1.1
IP Subnet Mask: 255.255.255.0

SLIP Configuration
IP Source: 0.0.0.0
IP Destination: 0.0.0.0
```

OK Cancel

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

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

3. Select the SoftStax Setup tab.
4. From the SoftStax Setup tab, select the `Enable SoftStax` radio button.
5. Click `OK` to close the dialog box.

Building the Coreboot + Bootfile Image

Once your coreboot and bootfile settings have been properly configured, complete the following steps to build the ROM image:

1. Select `Configure -> Build Image`. The Master Builder window is displayed.
2. From the Master Builder window, select the check boxes appropriate for your setup:
    - If you are using a RAM disk, select the Disk Support and Disk Utilities check boxes.
    - If you have Ethernet setup for your system, select the SoftStax (SPF) Support and User State Debugging Modules check boxes.
3. Select the `Coreboot + Bootfile` radio button.
4. Click `Build`. This builds the ROM image that can later be burned into flash memory. The name of this ROM image is `rom.S`. The file containing the image is in Motorola S-record format.
5. Click `Finish`.
6. Select `File -> Save Settings` to save the configuration.
7. Select `File -> Exit` to quit 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:

1. On switch bank S5, set switches 2, 5, and 8 in the `ON` position. All other switches should remain in the `OFF` position.
2. The serial cable should be connected to PJ6 with a setting of 38400 8N1. Once you apply power, the HCP5 prompt should appear.
3. At the HCP5 prompt, type `o`. The screen should display the following information:

```
HCP5? o
Option menu
1:flash copy16 2:Flash test  3:flash copy32 4:bootwt 5:115K-BPS
a:isatest
b:pcitest
```

```
c:akiruno

d:hcp5_b27

Option command ?
```

4. At the Option command, type `4`. The following message appears:

```
Flash Memory breaks when executes this function. Are you sure? [y}?
```

5. Type `y` and hit Enter. The following message appears:

```
ROM Write

flash ID check

ID be000000  a00055a0  setup-ok

ROM-BUS=32 1CHIP=16Bit 4 4 MBM29PL160BD aaaa-be000000 80000
1f80000 2

ROM ADR-bed00000 c0000000

load Srecord file
```

6. From the Hyperterminal window, transfer the `rom.S` file from the following directory to the target:

   ```
   MWOS/OS9000/MIPS3000/PORTS/JMRTX3927/BOOTS/INSTALL/
   PORTBOOT
   ```

   This transfer should take awhile to complete. When it is finished, you will need to wait awhile longer for the flash to complete and the HCP5 prompt to appear again. Once the flash is complete, the screen should display the following information:

```
e c0000000 bfc00000 m=003fffff add=00000000 size=00400000
a0100000 hi=c0000000   low=bfc00000

FL adr=bec00000  flash OK

FL adr=bec80000  flash OK

FL adr=bed00000  flash OK

FL adr=bed80000 flash OK

FL adr=bed00000  flash OK

FL adr=bed80000  flash OK

FL adr=bed00000  flash OK

FL adr=bed80000  flash OK

ROM write END

HCP5?
```

7. Power the system off.
8. On switch bank S5, set switch 1 to the `ON` position. All others should remain in `OFF` position. In addition, if you have not already done so, turn switch 2 on switch bank S2 to the `OFF` position. OS-9 needs the TLBs enabled in order to run.

The serial cable should now be connected to PJ7 with a setting of 9600 8N1. Once you apply power, the OS-9 prompt should appear.

Optional Procedures

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

Building with Makefiles

Building boots with makefiles allows you greater control over which modules are included in the boot. For the Toshiba JMR-TX3927 reference board, the directory in which boots can be made is listed below:

`MWOS/OS9000/MIPS3000/PORTS/JMRTX3927/BOOTS/SYSTEMS/PORTBOOT`

Makefile Network Option

Networking is not included in a bootfile by default. To include the networking modules in the bootfile, set the network macro definition to `TRUE` in the file named `makefile`. In addition, be certain that the IP address for the board is setup correctly; this helps 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 to include or exclude modules required for the system.

These bootfile lists are located in `<PORTS>/BOOTS/SYSTEMS/PORTBOOT`, and 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`. 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, IP stack, and SCSI system can be uncommented in order to provide `bootp` booting and/or SCSI booting. Low-level Ethernet or low-level SLIP can also provide system state debugging through Hawk.
`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, one file needs to be altered and one makefile needs to be run. To do this, complete the following steps:

1. Navigate to the `MWOS\OS9000\MIPS3000\PORTS\JMRTX3927\SPF\ETC` directory and open the `interfaces.conf` file.
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.
3. Save the file.

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

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:

1. Navigate to the `MWOS\OS9000\MIPS3000\PORTS\JMRTX3927\ROM\CNFGDATA` directory and open the `config.des` file.
2. From the `config.des` file, you will need to correctly define the macros for the IP address, broadcast, subnet, and mac.
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.

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.

•

2 Board Specific Reference

This chapter contains porting information specific to the Toshiba JMR-TX3927 board. It includes the following sections:

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.)

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_config.h` header file or 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. Below is an example of how you can redefine the bootstrap configuration of your system using the `BOOT_CONFIG` macro in the `rom_config.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.

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;
}
```

A Board Specific Modules

This chapter describes the modules specifically written for the Toshiba JMR-TX3927 board. It includes the following sections:

Low-Level System Modules

The following low-level system modules are tailored specifically for the Toshiba JMR-TX3927 board. They are located in the following directory:

`MWOS/OS9000/MIPS3000/PORTS/JMRTX3927/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
`io3927`	low-level serial driver for serial ports
`llne2000`	low-level Ethernet driver module
`llpro100`	low-level Ethernet driver module
`portmenu`	inits booters defined in the `cnfgdata`
`romcore`	bootstrap code
`tmr3927`	simulated low-level timer module
`usedebug`	debugger configuration module

High-Level System Modules

The following OS-9 system modules are tailored specifically for the Toshiba JMR-TX3927 board. Unless otherwise specified, each module is located in the following directory:

`MWOS/OS9000/MIPS3000/PORTS/JMRTX3927/CMDS/BOOTOBJS`

PIC Support

Module	
`picsub`	programmable interrupt controller handling module

Ticker (System Clock) Support

Module	
`tk3927`	system ticker module

Module	
`hcsub`	provides a high speed timer interface used by the HawkEye profiler

Real Time Clock

`rtc1742`	real-time clock module

Serial Support

Module

`sc3927`	serial port driver

Descriptors /term1 /t1

term1 and t1 are assigned to SIO0. The connector for SIO0 is labeled as PJ7.

Driver Name: `sc3927`

Default Baud Rate: 9600

Default Parity: None

Default Data Bits: 8

To use it: Select `SIO0` in the Configuration Wizard.

Descriptors /term2 /t2

term2 and t2 are assigned to the SIO1. The connector for SIO1 is labeled as PJ6.

Driver Name: `sc3927`

Default Baud Rate 9600

Default Parity: None

Default Data Bits: 8

To use it: Select `SIO1` in the Configuration Wizard.

Baud Rates

The following OS-9 baud rates are supported by the `sc3927` driver:

Table A-1. Supported SC85x30 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 `sc3927` driver:

Table A-2. SC85x30 Baud Rates Not Supported

31250	56000	57600	64000	115200

Common System Modules List

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

`MWOS/OS9000/MIPS3000/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
`flshcach`	provides low-level cache management services
`hlproto`	provides user level code access to protoman
`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
`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
`sndp`	provides low-level system debug protocol
`srecord`	provides a booter that accepts S-Records
`swtimer`	provides timer services via software loops