TEXAS INSTRUMENTS

TMS320C2xx Workstation Emulator

Installation Guide

1995 Microprocessor Development Systems

[redacted]

Texas Instruments

Printed in U.S.A., September 1995
D412004-9741 revision*

SPRU153

TMS320C2xx Workstation Emulator Installation Guide

SPRU153
September 1995

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

TRADEMARKS

Contents

Figures

Chapter 1

Installing the Emulator and C Source Debugger

This chapter helps you to install the TMS320C2xx emulator, C source debugger, and the parallel debug manager (PDM) on a SPARCstation running OpenWindows™ under SunOS™ version 4.1.x or 5.x (SunOS version 5.x is also known as Solaris™ version 2.x). After completing the installation, turn to the *TMS320C2xx C Source Debugger User's Guide*.

Topic **Page**

1.1 What You'll Need

The following checklists describe items that are shipped with your emulator and any additional items you'll need to use this tool.

Hardware checklist

☐	**host**	A SPARCstation or 100% compatible system.
☐	**display**	Monochrome or color (color recommended).
☐	**interface to host**	A SCSI bus controller with at least one free SCSI identifier (refer to page 1-7 for more information on locating a free SCSI ID).
☐	**power supply**	The external power supply for the emulator.
☐	**emulator**	An XDS510WS emulator.
☐	**SCSI cable**	A SCSI cable used for connecting the emulator to your SPARCstation.
☐	**SCSI terminator**	A SCSI bus terminator if your emulator is at the end of the SCSI chain. Refer to page 1-10 for more information on the SCSI terminator.
☐	**emulation cable**	A cable that connects the emulator to your target system.
☐	**target system**	A board with at least one 'C2xx on the emulation scan path.
☐	**connector to target system**	A 14-pin connector (two rows of seven pins)-for more information about this connector, refer to the *JTAG/MPSD Emulation Technical Reference*.
☐	**required hardware**	CD-ROM drive.
☐	**optional hardware**	A mouse.
☐	**root privileges**	You must have root privileges to configure your SPARCstation and to mount and unmount the CD-ROM.

Software checklist

☐	**operating system**	OpenWindows version 3.0 (or higher) running under SunOS version 4.1.3 (or higher). If you're using SunOS 5.x (also known as Solaris 2.x), you must have the Binary Compatibility Package (BCP) installed; if you don't, get your system administrator's help.
☐	**software tools**	TMS320C1x/C2x/C2xx/C5x assembler and linker. Optional: TMS320C1x/C2x/C2xx/C5x C compiler
☐	**required files**	† *emu2xx* is the executable file that invokes the C source debugger for the emulator.
☐		† *c2xx510ws.out* is the executable portion of the debugger that runs on the emulator.
☐		† *emurst* resets the emulator and downloads c2xx510ws.out to the emulator.
☐		† *board.dat* describes your target board to your debugger in terms of what devices are on the emulation scan path. The board.dat file included in the debugger package is for a target board with one 'C2xx named *CPU_A*.
☐	**optional files**	† *emuinit.cmd* is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C209 memory map. If this file isn't present when you first invoke the debugger, then all memory is invalid at first. You *must* modify this file to match your target system's memory map. For information about setting up your own memory map, refer to the *Defining a Memory Map* chapter in the *TMS320C2xx C Source Debugger User's Guide*.
☐		† The *composer* utility allows you to convert your text board configuration file (*board.cfg*) into a format the debugger can read (*board.dat*). For the emulator to initialize properly, you must create a new board.dat file with this release of the composer.
☐		† *board.cfg* is a text file used to describe your target board in terms of what devices are on the emulation scan path. The board.cfg file included in the debugger package is for a target board with one 'C2xx named *CPU_A*.
☐		† *init.clr* is a general-purpose screen configuration file. If this file isn't present when you invoke the debugger, the debugger uses a default screen configuration.

† Included as part of the debugger package

☐ † *mono.clr* is a screen configuration file designed especially for monochrome monitors; the default is for color monitors.

For more information about these files and about setting up your own screen configuration, refer to the *Customizing the Debugger Display* chapter in the *TMS320C2xx C Source Debugger User's Guide*.

† Included as part of the debugger package

WARNING

1) **To minimize the risk of electric shock and fire hazard, be sure that all major components that you interface with Texas Instruments devices are limited in energy and certified by one or more of the following agencies: UV, CSA, VDE, or TUV.**
2) **Turn the power off before you connect components and cables.**
3) **Never disconnect or reconnect any cables or other hardware devices while the emulator is turned on.**
4) **Be sure all devices on the SCSI bus, including your workstation, are turned off before you connect the emulator to your workstation.**

1.2 Step 1: Connecting the Emulator to Your Workstation

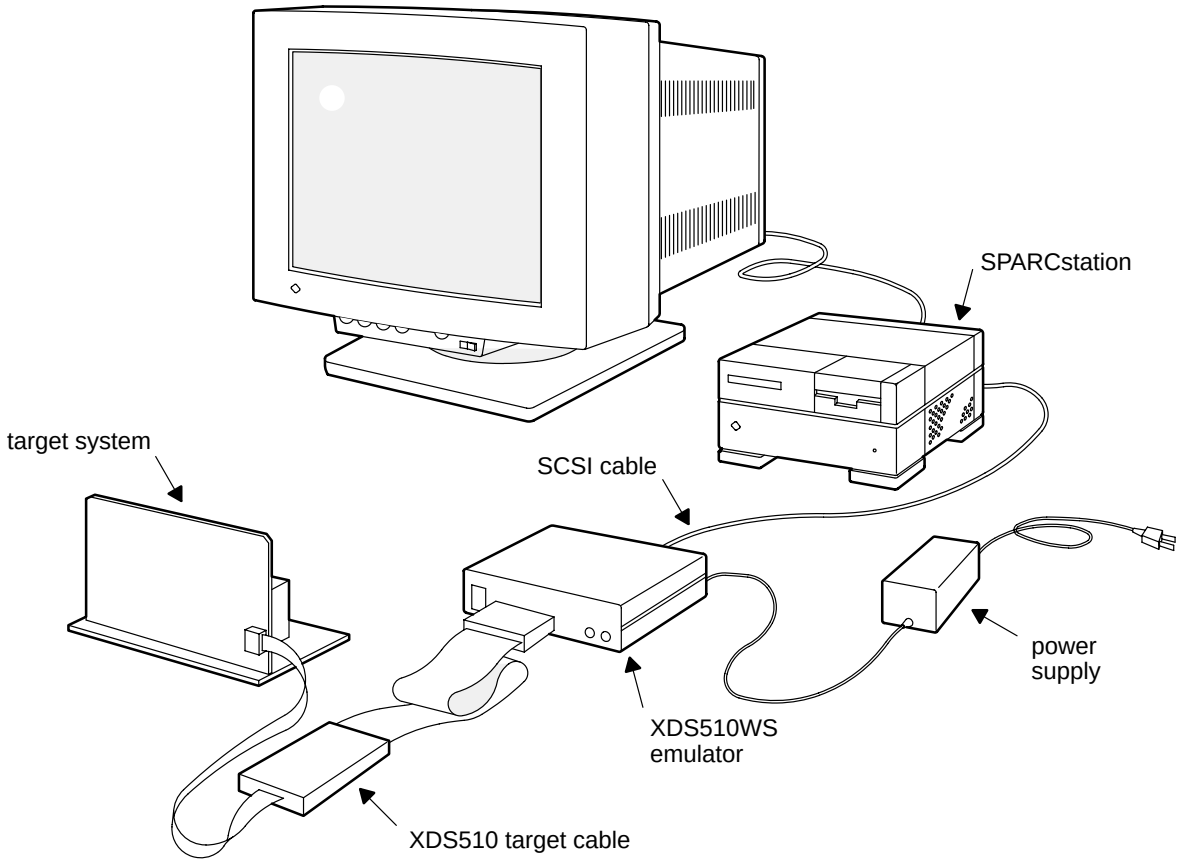
This section contains hardware installation information for the emulator. You *must* have root access to the host machine you intend to connect to the emulator. If you do not, contact your system administrator.

Figure 1–1 shows a typical setup using the emulator, target cable, and your workstation.

WARNING

Turn the power off before you connect components and cables.

Figure 1–1. Typical Setup of the Emulator on Your Workstation



Before you attach the emulator to your workstation, be sure the emulator is working properly. To do this, connect the power supply to your emulator and plug in the power supply (refer to Figure 1–2).

Turn on the emulator. When the first LED light from the left is lit, the emulator power is on. If this light does not come on, check your power connections and restart the emulator.

Refer to Figure 1–3 to locate the LED lights on the front of the emulator. Looking at your emulator, you should notice that the sixth LED from the left is on; this indicates that the emulator is running through a self test. Your emulator is ready and running properly, once the first, second, and fifth indicator lights from the left are on, and the sixth LED from the left is off.

If the first, second, and fifth indicator lights from the left do not come on, something is wrong with the emulator. Recheck your connections and turn the emulator off and on a second time. If the fifth indicator is still not on, shut off the emulator and contact the hotline.

Figure 1–2. Rear View of the XDS510WS Emulator

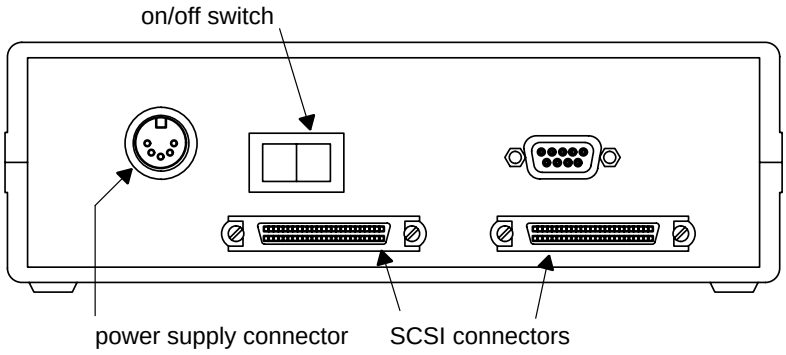
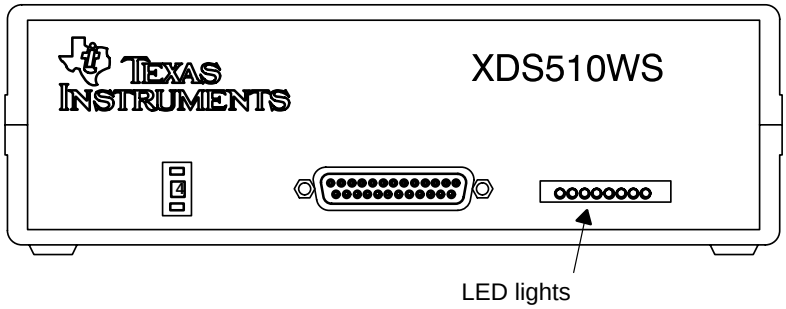


Figure 1–3. Front View of the XDS510WS Emulator



Locating a SCSI bus with an unused identifier

Each SCSI controller in your workstation has its own SCSI bus, and a workstation usually has only one SCSI controller (unless you have added additional controller cards). A single bus can support up to eight different devices, including the SPARCstation, each uniquely numbered 0 through 7, with the higher priority devices assigned to the larger SCSI ID numbers. Your SPARCstation is SCSI ID 7 by default. CD ROMs are ID 6 by default, and tape cartridges are usually ID 2. Your emulator uses SCSI ID 4 by default. If, however, SCSI ID 4 is already used, you must change the emulator's ID to one that is not used.

To get a list of the used SCSI IDs on your workstation, follow these steps:

Step 1: Enter the following command as *root* to get the PROM prompt:

```
halt ↵
```

Step 2: If you receive the following message:

```
Program terminated
Type b(boot), c(continue), or n(new command mode)
>
```

Type **n** ↵.

Step 3: After you receive the following message:

```
Type help for more information
ok
```

Type **probe–scsi** ↵ (or, if you have multiple SCSI controllers, use **probe–scsi–all** ↵).

You should see a list of used SCSI IDs scroll on your screen; it should look similar to the following message:

```
Target 3
   Unit 0 disk SEAGATE ST1480 SUN Copyright (c) 1992
   Seagate all rights reserved 0000
ok
```

The number following the word *Target* represents the currently used SCSI IDs. In the above message, SCSI ID number 3 is taken.

Note that the SPARC's SCSI ID is stored in the PROM environment variable *scsi_initiator_id* and can be viewed at this point when you type **printenv.**

Setting the SCSI ID on your emulator

If your SPARCstation is already using SCSI ID 4 (see the previous section on locating SCSI IDs), then you must change the SCSI ID on your emulator.

Before resetting the emulator's SCSI ID, be sure the emulator is not turned on.

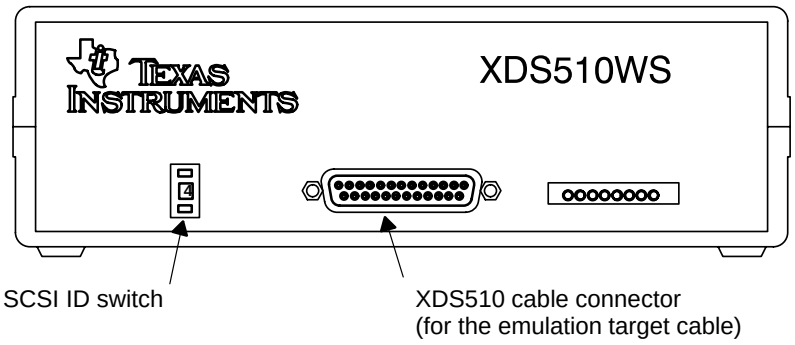
Never disconnect or reconnect any cables or other hardware devices while the emulator is turned on.

Your emulator's SCSI ID is controlled by a switch on the front panel of the emulator. Refer to Figure 1–4 for the location of this switch.

This switch can be in any one of ten positions, 0 through 9; however, do not use settings 8 and 9. (The emulator uses only the three least significant bits of the switch number; therefore, a setting of 8 would set the SCSI ID to 0, and a setting of 9 would set the SCSI ID to 1.)

When you've finished resetting the emulator's SCSI ID, you can connect the emulation target cable to the front of the emulator.

Figure 1–4. Front View of the XDS510WS Emulator



Adding the emulator onto the SCSI bus

The SCSI bus is a chain with two distinct ends; it is not a loop. Although there may be SCSI devices within your host, the visible chain begins at the host and ends at one of the external SCSI devices. You can connect the emulator into the SCSI bus anywhere along this chain; however, it's best to place the emulator where you can easily connect it to your target system. The emulator's indicator lights should be visible and the power switch readily accessible.

CAUTION

Be sure all devices on the SCSI bus, including your workstation, are turned off before you connect the emulator to your workstation.

Connect the SCSI cable to the back of your workstation (see Figure 1–5); you can use either one of the SCSI connectors. (Refer to Figure 1–6 for location of the SCSI connectors.)

Figure 1–5. Connecting the Emulator to Your Workstation

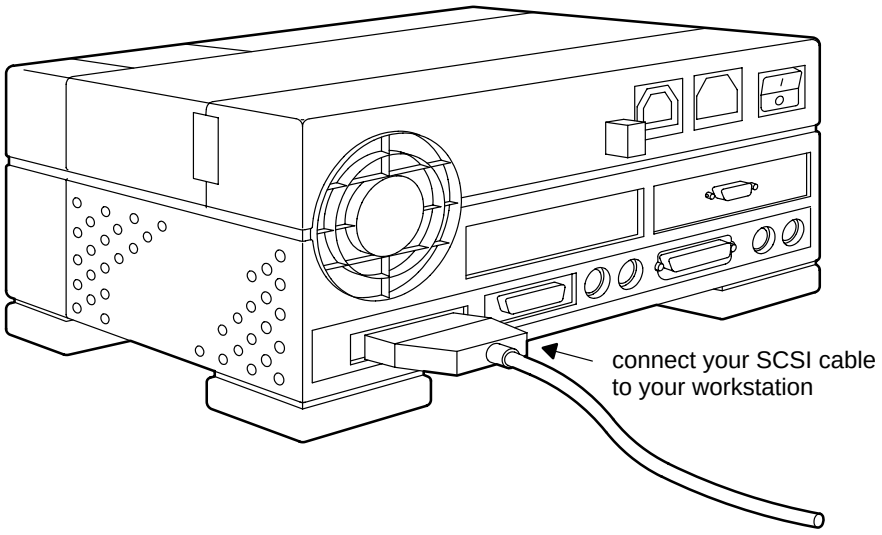
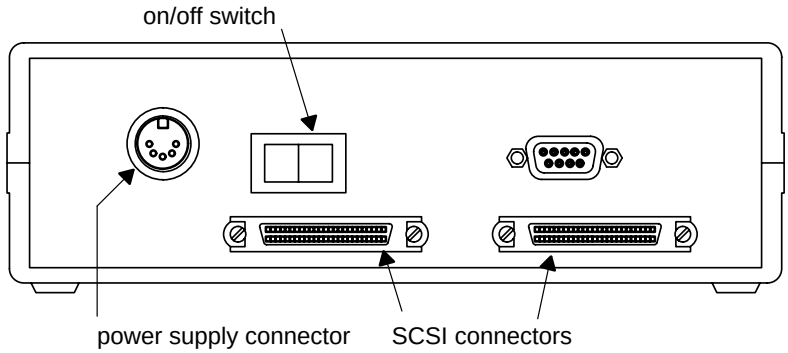


Figure 1–6. Rear View of the XDS510WS Emulator



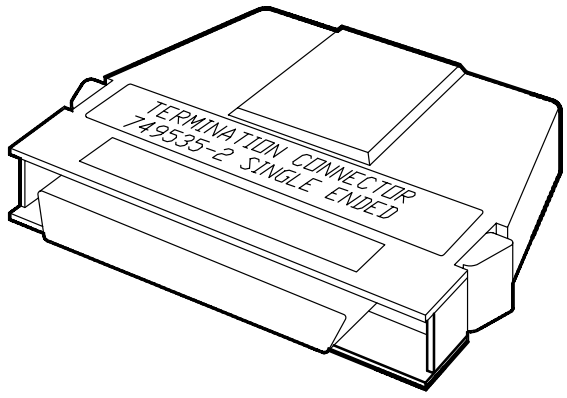
Terminating the SCSI bus

You *must* terminate the SCSI bus at each end of its chain to reduce signal noise. The device farthest on the chain from the host (your workstation) should be terminated; terminating intervening devices can cause intermittent errors in the SCSI bus.

If your emulator is at the end of a SCSI bus, you must terminate your emulator by connecting the external terminator (see Figure 1–7) to the unused SCSI connector on the back of your emulator.

Power up the external SCSI devices (including the emulator) before turning on your workstation.

Figure 1–7. The External Terminator for SCSI Bus Termination



1.3 Step 2: Setting Up Your Workstation to Recognize the Emulator

To continue this installation, you *must* have root access on your workstation; if you do not, contact your system administrator.

This step varies, depending on which version of the operating system you are using:

- **If you have SunOS 5.x**, as the root user, enter:

  `halt` ↵

  With the XDS510WS properly connected and powered up, reboot your SPARCstation with the PROM command, *boot –r.* Once the system comes back up, execute the following command as the root user:

  `/usr/sbin/disks` ↵

  Once you have entered these commands, you can skip to Section 1.4.

- **If you have SunOS 4.1.x**, you must complete the instructions in this section to set up your workstation.

Modifying your workstation's configuration file

Once you have set up your emulation hardware, you must modify your workstation's configuration file to allow the debugger to access the emulator. The name of the configuration file used by your workstation normally appears in parentheses following your SunOS version number when you boot or log in to your system. In Example 1–1, the configuration file is called GENERIC.

Example 1–1. Locating the Name of Your Configuration File

```
Last login: Mon Mar 15 09:40:13 on console
SunOS Release 4.1.1 (GENERIC)#1: Mon Feb 1 09:00:07 CST 1993
You have mail.
```

To change this configuration file, complete these ten steps:

1) Switch directories to find the configuration file. To do this, enter the following command:

   `cd /usr/kvm/sys/sun4/conf` ↵

   If this command does not work, replace **sun4** with either **sun4c** or **sun4m**, depending on the type of workstation:

Machine type	Use directory name
SPARCstation 1, 1+, or 2	sun4c
SPARCstation 5, 10, 20, xxx–MP	sun4m

Note:

If the specified directory does not exist or doesn't contain the specified configuration file, then your system was probably installed without modification privileges. Contact your system administrator for help.

2) Copy the current configuration file to a file called EMULATOR:

   cp *filename* **EMULATOR** ⏎

   Replace *filename* with the name of the current configuration file (see Example 1–1).

3) Edit the EMULATOR file. You can use any editor you are familiar with, but for the purpose of discussion, the vi editor is used; to use this editor, enter:

   vi EMULATOR ⏎

   Your EMULATOR file should look similar to Example 1–2. While in the vi editor, search for:

   - *ident* and replace the string following it with **"EMULATOR"**.
   - *options IPCSEMAPHORE* to be sure it exists in your configuration file and is not a comment; comments are preceded by the # symbol.
   - *options IPCSHMEM* to be sure it exists in your configuration file and is not a comment.
   - *options IPCMESSAGE* to be sure it exists in your configuration file and is not a comment.
   - *target # lun 0*, where # is the SCSI ID for the emulator (4 by default; refer to Section 1.2). Be sure the entry is set up as a disk; to do this, make sure *tape st4*, for example, is changed to *disk sd4*.

   Any other references to the driver that you have chosen (**sd4** is the default) should be turned into comments.

Note:

When you execute the debugger or emurst and you use the –p debugger option, you are referring to the sd# in the configuration file and to the associated rsd#a file. (These numbers are **not necessarily** the same as the SCSI ID number, but it can be.)

Example 1–2 shows a correctly modified EMULATOR file. Notice that modifications are highlighted and shown in bold face type. Lines preceded by # are comments and are ignored; you do not have to edit them. However, for consistency, these lines are modified.

Example 1–2.Setting Up the EMULATOR Configuration File

```
#
# @(#) GENERIC from master 1.28 90/09/21 SMI
#
# This config file describes an generic Sun–4c kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun–4c CPU types.
# There is little to be gained by removing support for particular
# CPUs, so you might as well leave them all in.
#
machine             ”sun4c”
cpu                 ”SUN4C_60”    # Sun–4/60
#
# Name this kernel EMULATOR.
#
ident         ”EMULATOR”
.
.
.
#
# The following options are for various System V IPC facilities.
# Most standard software does not need them, although they are
# used by SunGKS and some third–party software.
#
options IPCMESSAGE  # System V IPC message facility
options IPCSEMAPHORE# System V IPC semaphore facility
options IPCSHMEM    # System V IPC shared memory facility
.
.
.
scsibus0 at esp     # declare first scsi bus
  disk sd0 at scsibus0 target 3 lun 0   # first hard SCSI disk
  disk sd1 at scsibus0 target 1 lun 0   # second hard SCSI disk
  disk sd2 at scsibus0 target 2 lun 0   # third hard SCSI disk
  disk sd3 at scsibus0 target 0 lun 0   # fourth hard SCSI disk
  disk sd4 at scsibus0 target 4 lun 0   # XDS510WS emulator
  tape st1 at scsibus0 target 5 lun 0   # second SCSI tape
  disk sr0 at scsibus0 target 6 lun 0   # CD–ROM device

scsibus1 at esp     # declare second scsi bus
  #disk sd4 at scsibus1 target 3 lun 0  # fifth hard SCSI disk
  disk sd5 at scsibus1 target 1 lun 0   # sixth hard SCSI disk
  disk sd6 at scsibus1 target 2 lun 0   # seventh hard SCSI disk
  disk sd7 at scsibus1 target 0 lun 0   # eighth hard SCSI disk
  tape st2 at scsibus1 target 4 lun 0   # third SCSI tape
  tape st3 at scsibus1 target 5 lun 0   # fourth SCSI tape
  disk sr1 at scsibus1 target 6 lun 0   # 2nd CD–ROM device
```

4) When you have finished editing your EMULATOR file, save the file and exit the vi editor by pressing SHIFT Z Z.

5) Now, you must create the EMULATOR directory. To do this, enter:

   config EMULATOR ↵

6) Once you have created the EMULATOR directory, change your current directory to your newly created directory by entering:

   cd ../EMULATOR ↵

7) Now you must compile the new kernel described by your configuration file. To do this, enter:

   make ↵

8) Save the old kernel file (vmunix) so that you can easily revert to it; enter:

   mv /vmunix /vmunix.orig ↵

9) To move the new kernel file into use, enter:

   cp vmunix / ↵

10) You are now ready to reboot your workstation; enter:

    shutdown –r now ↵

When you log onto your workstation, you should notice the name EMULATOR appear in parentheses as shown in Example 1–3.

If EMULATOR does not appear in parentheses as you are rebooting your workstation, then your emulator may not be installed properly. Go back through these ten steps and be sure that you have followed each step correctly.

Example 1–3. Running Your Modified Configuration File

```
Last login: Mon Mar 15 09:40:13 on console
SunOS Release 4.1.1 (EMULATOR)#1: Mon Feb 1 09:00:07 CST 1993
You have mail.
```

1.4 Step 3: Allowing the Debugger to Access the Emulator

The debugger accesses the emulator by reading from and writing to the device driver you defined in the EMULATOR configuration file. As a result, to execute the debugger, you must have read and write permissions on the driver file.

This step varies, depending on which version of the operating system you are using:

- **If you have SunOS 5.x**, nothing further is required after taking the actions in the first bullet of Section 1.3. To confirm proper operation, execute the following command.

  `ls –l /dev/rsd*a` ⏎

  If rsd#a is not listed, return to the first bullet in Section 1.3. If rsd#a is listed with permissions other than lrwxrwxrwx, change them as shown below for SunOS 4.1.x users.

- **If you have SunOS 4.1.x**, as the root user, enter the following command, replacing # with the device driver number of the emulator (4 by default):

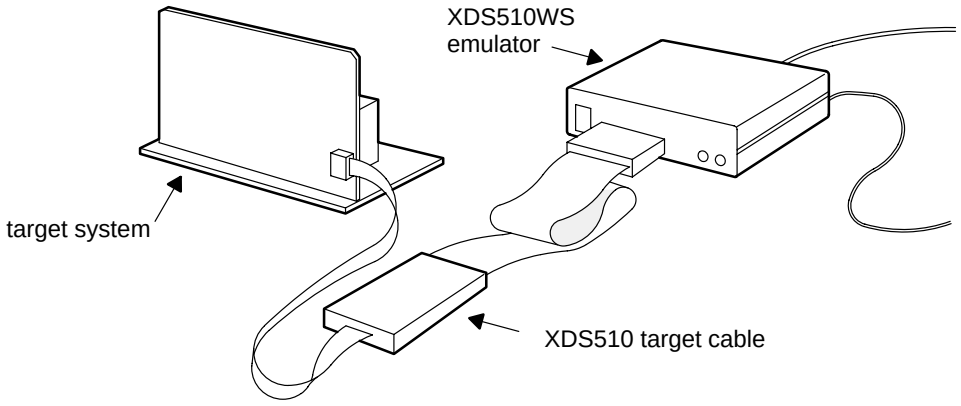
  `chmod a+rw /dev/rsd#a` ⏎

  This enables the debugger to access the emulator *without* root privileges.

1.5 Step 4: Connecting the Emulator to Your Target System

In most cases, the target system is a board of your own design. It should have the appropriate emulation header, as described in the hardware requirements (refer to page 1-2). To correctly connect the target cable to your target board, make sure the key is aligned properly; then, slowly and firmly, attach the cable (see Figure 1–8).

Figure 1–8. Connecting the Emulator to Your Target System



1.6 Step 5: Installing the Debugger Software

This section explains the process of installing the debugger software on your hard-disk system. The software package is shipped on a CD-ROM. To install the software, you must mount the CD-ROM, copy the files, and unmount the CD-ROM.

Mounting the CD-ROM

The steps to mount the CD-ROM vary according to your operating-system version:

- If you have SunOS 4.1.x, as root, load the CD-ROM into the drive and enter the following from a command shell:

  ```
  mount –rt hsfs /dev/sr0 /cdrom ↵
  exit ↵
  cd /cdrom ↵
  ```

- If you have SunOS 5.0 or 5.1, as root, load the CD-ROM into the drive and enter the following from a command shell:

  ```
  mount –rF hsfs /dev/sr0 /cdrom ↵
  exit ↵
  cd /cdrom/cdrom0 ↵
  ```

- If you have SunOS 5.2 or higher:

  - If your CD-ROM drive is already attached, load the CD-ROM into the drive and enter the following from a command shell:

    ```
    cd /cdrom/cdrom0 ↵
    ```

  - If you do not have a CD-ROM drive attached, you must shut down your system to the PROM level, attach the CD-ROM drive, and enter the following:

    ```
    boot –r ↵
    ```

    After you log into your system, load the CD-ROM into the drive and enter the following from a command shell:

    ```
    cd /cdrom/cdrom0 ↵
    ```

Copying the files

After you've mounted the CD-ROM, you must create the directory that will contain the debugger software and copy the software to that directory.

1) Create a directory named *c2xxhll* on your hard disk. To create this directory, enter:

   mkdir c2xxhll ↵

2) Copy the files from the CD-ROM to your hard-disk system:

   cp –r * c2xxhll ↵

Unmounting the CD-ROM

You must unmount the CD-ROM after copying the files.

- If you have SunOS 4.1.x, 5.0, or 5.1, as root, enter the following from a command shell:

  cd ↵
  umount /cdrom ↵
  eject /dev/sr0 ↵
  exit ↵

- If you have SunOS 5.2 or higher, enter the following from a command shell:

  cd ↵
  eject ↵

1.7 Step 6: Making Sure the Emulator Supports the Debugger

The ROM code for the emulator does not contain the information necessary to debug a processor; that code must be downloaded from the host. This makes it easier to upgrade the emulation software. The *emurst* program downloads the necessary code for proper emulation.

To run this program, enter the emurst command in the following format:

emurst [**–x**] [**–p** *number*] *pathname–filename*

The –x option tells the emurst utility to ignore any options specified with the D_OPTIONS environment variable.

Number represents the device driver number you defined in the EMULATOR configuration file (refer to Section 1.3), and *pathname–filename* is the location and name of the c2xx510ws.out file.

You can omit the –p option if the default, *rsd4a*, is the device driver for the emulator.

Note:

When you execute the debugger or emurst and you use the –p debugger option, you are referring to the sd# in the configuration file and to the associated rsd#a file. (These numbers are **not necessarily** the same as the SCSI ID number, but it can be.)

You can be sure that emurst succeeded when only the first and second LEDs from the left are on.

1.8 Step 7: Describing Your Target System to the Debugger

In order for the debugger to understand how you have configured your target system, you must supply the target configuration information in a file for the debugger to read.

- ❑ If you're using an emulation scan path that contains only one 'C2xx and no other devices, you can use the *board.dat* file that comes with the 'C2xx emulator kit. This file describes to the debugger the single 'C2xx in the scan path and gives the 'C2xx the name CPU_A. Since the debugger automatically looks for a file called board.dat in the current directory and in the directories specified with the D_DIR environment variable, you don't need to create your own board configuration file. Go to the next section.

- ❑ If you plan to use a different target system, you must follow these steps:

  Step 1: Create the board configuration file.

  Step 2: Translate the board configuration file to binary so that the debugger can read it.

  Step 3: Specify the configuration file when invoking the debugger.

  These steps are described in the *Describing Your Target System to the Debugger* appendix in the *TMS320C2xx C Source Debugger User's Guide*.

1.9 Step 8: Setting Up the Debugger Environment

To ensure that your debugger works correctly, you must:

- ❑ Modify the PATH statement to identify the c2xxhll directory.

- ❑ Define environment variables so that the debugger can find the files it needs.

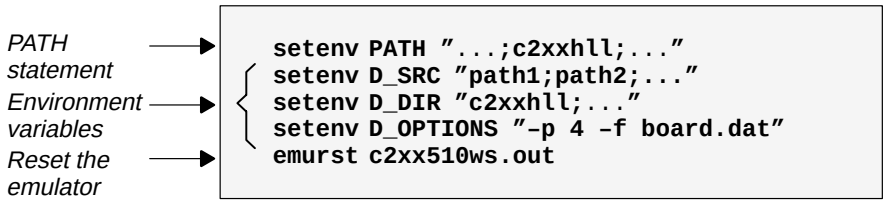
- ❑ Identify any nondefault device driver used by the emulator.

- ❑ Reset the emulator.

You can accomplish most of these tasks by entering individual commands, but it's simpler to put the commands in your .cshrc file.

Figure 1–9 shows an example of a .cshrc file that contains the suggested modifications (highlighted in bold type). The subsections following the figure explain these modifications.

Figure 1–9. Command Setup for the Debugger



Modifying the PATH statement

Define a path to the debugger directory. The general format for doing this is:

```
setenv PATH "...;c2xxhll;..."
```

This allows you to invoke the debugger without specifying the name of the directory that contains the debugger executable file.

Setting up the environment variables

An environment variable is a special system symbol that the debugger uses for finding or obtaining certain types of information. The debugger uses five environment variables named D_DIR, D_SRC, D_OPTIONS, DISPLAY (X Window System™ only), and LD_LIBRARY_PATH. The next five bullets tell you how to set up these environment variables. The format for doing this is the same for either the .cshrc file or the command line.

- Set up the D_DIR environment variable to identify the emulator directories:

  ```
  setenv D_DIR "c2xxhll"
  ```

  These directories contain auxiliary files (init.cmd, init.clr, etc.) that the debugger needs.

- Set up the D_SRC environment variable to identify any directories that contain program source files that you'll want to look at while you're debugging source code. The general format for doing this is:

  ```
  setenv D_SRC "path1;path2;..."
  ```

  For example, if your programs were in a directory named *csource*, the D_SRC setup would be:

  ```
  setenv D_SRC "csource"
  ```

- You can use several options when you invoke the debugger. If you use the same options repeatedly, it's more convenient to specify them by using D_OPTIONS. The general format for doing this is:

  setenv D_OPTIONS [*object filename*] [*debugger options*]

  This tells the debugger to load the specified object file and use the specified options each time you invoke the debugger. Table 1–1 lists the options that you can identify with D_OPTIONS.

Table 1–1. Options for Use With D_OPTIONS

Option	Brief Description
–b[b]	Select the screen size
–c	Clear the .bss section
–d *machinename*	Display the debugger on different machine
–f *filename*	Identify a new board configuration file
–i *pathname*	Identify additional directories
–min	Select the minimal debugging mode
–n *processor name*	Identify processor for debugging
–p *driver number*	Identify the device driver number
–s	Load the symbol table only
–t *filename*	Identify a new initialization file
–v	Load without the symbol table

Note that you can override D_OPTIONS by invoking the debugger with the –x option.

For more information about options, see the invocation instructions in the *Overview of a Code Development and Debugging System* chapter in the *TMS320C2xx C Source Debugger User's Guide*.

- If you are using the X Window System, you can use the DISPLAY environment variable to display the debugger on a different machine than the one the debugger is running on. The general format for doing this is:

  setenv DISPLAY "*machinename***"**

  For example, if you are running the debugger on a machine called opie and you want the 'C2xx debugger display to appear on a machine called barney, the DISPLAY setup would be:

  ```
  setenv DISPLAY barney:0
  ```

For more information about using the debugger under the X Window System, refer to Section 1.11, *Using the Debugger With the X Window System*.

- The LD_LIBRARY_PATH environment variable allows you to specify pathnames to the OpenWindows X Window library. If you are unable to invoke the debugger or use the emulator properly, be sure that the LD_LIBRARY_PATH environment variable is set to the appropriate pathnames. For example:

  setenv LD_LIBRARY_PATH "/usr/openwin/lib: /usr/lib"

Invoking the new or modified .cshrc file

If you create or modify your .cshrc file, you must invoke that file before invoking the debugger for the first time. To do so, enter:

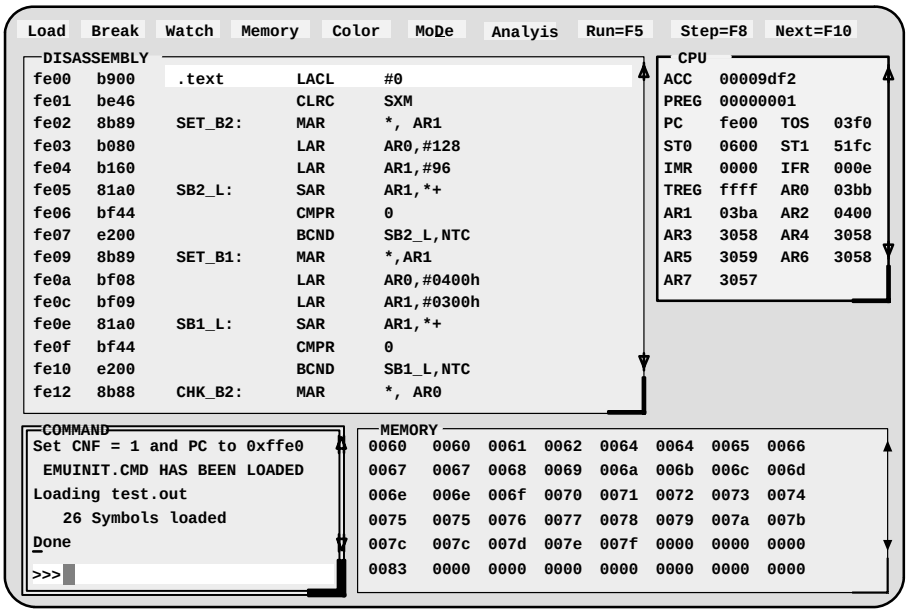
source .cshrc ⏎

1.10 Step 9: Verifying the Installation

To ensure that you have correctly installed the emulator and debugger software, enter this command at the system prompt:

emu2xx test –n CPU_A ↵

You should see a display similar to this one:



- If you see a display similar to this one, you have correctly installed your emulator and debugger.
- If you don't see a display, then your debugger or board may not be installed properly. Go back through the installation instructions and be sure that you have followed each step correctly; then reenter the command above.

1.11 Using the Debugger With the X Window System

If you're using the X Window System to run the debugger, you need to know about the keyboard's special keys, the debugger fonts, and using the debugger on a monochrome monitor.

Using the keyboard's special keys

The debugger uses some special keys that you can map differently from your particular keyboard. Some keyboards, such as the Sun Type 5 keyboard, may have these special symbols on separate keys. Other keyboards, such as the Sun Type 4 keyboard, do not have the special keys.

The special keys that the debugger uses are shown in the following table with their corresponding keysym. A **keysym** is a label that interprets a keystroke; it allows you to modify the action of a key on the keyboard.

Key	Keysym
F1 to F10	F1 to F10
PAGE UP	Prior
PAGE DOWN	Next
HOME	Home
END	End
INSERT	Insert
→	Right
←	Left
↑	Up
↓	Down

Use the X utility xev to check the keysyms that are associated with your keyboard. If you need to change the keysym definitions, use the xmodmap utility. For example, you could create a file that contains the following commands and use that file with xmodmap to change a Sun Type 4 keyboard to match the keys listed above:

```
keysym R13    = End
keysym Down   = Down
keysym F35    = Next
keysym Left   = Left
keysym Right  = Right
keysym F27    = Home
keysym Up     = Up
keysym F29    = Prior
keysym Insert = Insert
```

Refer to your X Window System documentation for more information about using xev and xmodmap.

Changing the debugger font

You can change the font of the debugger screen by using the xrdb utility and modifying the .Xdefaults file in your root directory. For example, to change the fonts of the debugger to Courier, add the following line to the .Xdefaults file:

```
emu2xx*font:courier
```

For more information about using xrdb to change the font, refer to your X Window System documentation.

Color mappings on monochrome screens

Although a color monitor is recommended (and necessary for the graphic display features), the following table shows the color mappings for monochrome screens:

Color	Appearance on Monochrome Screen
black	black
blue	black
green	white
cyan	white
red	black
magenta	black
yellow	white
white	white

Chapter 2

Troubleshooting

This chapter describes some common problems you may encounter while using your emulator or debugger on your workstation. You should be familiar with the procedures described in Chapter 1 before trying to troubleshoot problems with the XDS510WS and its software.

Topic **Page**

2.1 Problems When Booting Your Workstation

After installing your emulator, problems may occur when you attempt to boot your workstation. The following suggestions resolve many of these problems:

- ❑ Your workstation will not boot when connected to your emulator, even if your emulator is not turned on.

  1) Be sure all of your SCSI cables are connected securely and the SCSI bus is terminated properly (see *Terminating the SCSI bus* on page 1-10).

  2) Remove any unnecessary SCSI devices from the bus.

  3) Make sure the total length of the SCSI bus is less than six meters, including the section of the bus within the SPARC chassis.

- ❑ Your workstation will boot when the emulator is turned off but will not boot when the emulator is turned on.

  Your emulator's SCSI ID conflicts with the SCSI ID of another device on the SCSI bus. Go back through the instructions on page 1-7.

2.2 Problems When Resetting the Emulator

After you power up the emulator and the workstation, if you have the following problems attempting to reset the emulator, implement the applicable solutions:

- When you execute the emurst command, you receive this message:

  ```
  emurst file [.out]:
  ```

  You forgot to specify the *pathname–filename* of the c2xx510ws.out file. You can specify it at this prompt or re-execute emurst with *pathname–filename* specified on the command line.

- When executing the emurst command, you receive this message:

  ```
  >> can't initialize the target system
  ```

  1) You haven't set the IPCSEMAPHORE option to allow the debugger to access the emulator. Be sure that the configuration file, EMULATOR, has the options line set correctly, without comments. Then, use the corrected configuration file to build the currently executing kernel (see Section 1.3, *Step 2: Setting Up Your Workstation to Recognize the Emulator,* on page 1-11).

  2) There are too many current semaphores on the system. Clean up the unused semaphores by using the **ipcs –st** and **ipcrm** utilities, and try to execute emurst again.

  3) You may not have permission to access the driver file you specified with the –p debugger option. Normally, you specify the –p debugger option on the command line or in the D_OPTIONS environment variable. Remember, if you haven't specifically reset the driver file number to another number, the default is 4. Have the root user execute the following command and try to execute emurst again.

     chmod a+rw /dev/rsd#a ↵

  4) The driver file you specified with the –p debugger option is not correctly associated with your emulator in your configuration file. Make sure your configuration file contains a line similar to this:

     ```
     disk sd# at scsibus<m> target <s> lun 0
     ```

     where # is the device driver number, <s> is the SCSI ID of the XDS510WS you set with the switch at the front of the XDS510WS. The <m> is zero(0), unless the XDS510WS is connected to a second SCSI bus that you added to your SPARC, which will cause <m> to change. Use the corrected configuration file to build the currently executing kernel (see Section 1.3, *Step 2: Setting Up Your Workstation to Recognize the Emulator,* on page 1-11).

5) You haven't turned on the XDS510WS, or it hasn't completed its self tests. Turn on the XDS510WS and wait for the self test to complete successfully before executing emurst. The self test has completed, once the sixth LED from the left is off and the first, second, and fifth LEDs from the left are on.

- When executing the emurst command, you receive this message:

  ```
  >> error loading file
  ```

  1) The emurst utility can't find the c2xx510ws.out file as specified. If you didn't specify the *pathname–filename* with an extension as part of the name, the emurst utility appends the default extension *.out* to the name.

  2) If you didn't provide path information (just the filename), emurst searches first in the current directory and then in all of the directories specified in the D_DIR environment variable before returning this error. Make sure the correct file is somewhere emurst can find it.

  3) The file that you specified to emurst isn't appropriate for this use. Use the c2xx510ws.out file that is included with the debugger software.

2.3 Problems When Invoking the Debugger

If you encounter these problems when you invoke the debugger, the suggested solutions may resolve the problems:

❑ You receive the following message when executing the emu2xx command:

```
CANNOT INITIALIZE THE TARGET !!
– Check I/O configuration
– Check cabling and target power
```

1) The emurst command didn't successfully execute before you tried to invoke the debugger. Execute emurst (see Section 1.7, *Step 6: Making Sure the Emulator Supports the Debugger,* on page 1-18). The emurst has completed successfully if you see your command prompt after this message:

   ```
   EMURST for XDS510WS loading <pathname-filename> at #
   ```

   where <pathname-filename> is the location of the c2xx510ws.out file, and # refers to the file /dev/rsd#a, which is associated with the emulator in the configuration file, EMULATOR. Also, you can be sure that emurst succeeded when only the first and second LEDs from the left are on.

2) The –p debugger option that you entered on the command line or in the D_OPTIONS environment variable specifies a different driver file than the one used by emurst. Remember, if you haven't specifically reset the driver file number to another number, the default is 4. Use the same –p option that you used when you executed emurst. (Refer to Section 1.7 on page 1-18 for more information on the –p option).

3) The –f debugger option you specified on the command line or in the D_OPTIONS environment variable (where the default file specified by the –f option is board.dat) specifies a file that the debugger can't find.

   - If you didn't provide *any* path information with the filename, the debugger couldn't find the file in the current directory or in any of the directories listed in the D_DIR environment variable.
   - If you didn't provide the *correct* path information, re-execute the debugger, specifying the correct pathname and filename for the board configuration file.

4) One of these two problems could exist:

   - You didn't specify **–n** *processor name* debugger option.
   - The debugger couldn't find the *processor name* that you specified with the **–n** option in your board configuration file.

   Re-execute the debugger with the –n debugger option, specifying the name of a *processor name* from the board configuration file.

5) You may not have described your target system correctly in the board configuration file that you specified with the –f debugger option on the command line or in the D_OPTIONS environment variable. Review your board configuration file and correctly describe the target system.

6) Make sure your emulation cable is firmly attached both to the XDS510WS and to your target system.

7) Make sure your target system is receiving sufficient power at the required voltage to allow all devices on the board to work properly.

❑ You receive the following messages at the operating-system command line when trying to execute the emu2xx command:

```
emu2xx: display :0.0 doesn't know font 7x14
```

The default font file that the debugger uses (7x14.ff) couldn't be found by OpenWindows. OpenWindows searches for these font files in the directories specified in the FONTPATH environment variable. To correct the problem, do one of the following:

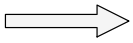
- Add the font file 7x14.ff to a directory defined in the FONTPATH environment variable.
- Add to the .Xdefaults file in your home directory the line "emu2xx*font: *GoodFontName*", where *GoodFontName* is the name of a font that OpenWindows can find.
- Copy a valid font file onto 7x14.ff.

Note:

The operating-system window provides operating-system messages. These messages differ from the error messages that you may see in the COMMAND window of the debugger.

2.4 Additional Emulator and Debugger Problems

The operating-system window displays operating-system messages. These messages differ from the error messages that you may see in the COMMAND window of the debugger. If you receive one of these operating-system messages while executing the emurst or the debugger, refer to the following explanations.



Note:

For each of the following four bulleted items (❑), note that the messages are *status messages, **not** error messages*.

❑ In your operating-system window, you receive the following message while executing emu2xx or emurst under SunOS 4.1.x:

```
<date> <time> <hostname> vmunix: sd<n>: disk not
responding to selection
```

or under SunOS 5.x:

```
WARNING: /sbus@1,f8000000/esp@0,800000/sd@<n>,0(sd<n>):
      disk not responding to selection
```

The XDS510WS didn't respond to the SPARC in a certain amount of time. This can be caused by several different things:

- The XDS510WS isn't powered
- The XDS510WS is executing its self test
- The XDS510WS is executing a lengthy debugger command such as a large memory-fill

❑ In your operating-system window, you receive the following message while executing mpemu, ppemu, or emurst under SunOS 4.1.x:

```
<date> <time> <hostname> vmunix: sd<n>: offline
```

or under SunOS 5.x:

```
WARNING /sbus@1,f8000000/esp@0,800000/sd@<n>,0(sd<n>):
      offline
```

The SPARCstation is unable to select the XDS510WS after several attempts and therefore considers the emulator offline. This message can be generated during large memory-fill instructions and should **not** be considered an error by itself or in combination with the preceding message. The debugger automatically corrects for this situation, unless a major error has taken place, in which case, the debugger eventually returns an error message in the COMMAND window of the debugger.

- In your operating-system window, you receive the following message while executing mpemu, ppemu, or emurst under SunOS 4.1.x:

  ```
  <date> <time> <hostname> vmunix: sd<n>: disk okay
  ```

  or under SunOS 5.x:

  ```
  WARNING: /sbus@1,f8000000/esp@0,800000/sd@<n>,0(sd<n>):
        disk okay
  ```

  The SPARCstation has reconnected with the XDS510WS after the XDS510WS didn't respond to the selection. When the debugger recovers from the *offline* condition (described in the previous bulleted item), one of the two messages shown above is written to the operating-system window.

- In your operating-system window, you receive the following message while executing mpemu, ppemu, or emurst under SunOS4.1.x:

  ```
  sd<n> at esp0 target <p> lun 0
  sd<n>: Vendor 'TI–ASP', product 'XDS510–WS_Rev.*', 130
  512 byte blocks
  <date> <time> <hostname> vmunix: sd<n>: corrupt label –
  wrong magic number
  ```

  If the emulator has been inactive on the bus since the SPARCstation's last attempt to access it, the XDS510WS returns to an active status on the bus. The above message informs you of this *new* SCSI device.

Note:

Since the SPARCstation interprets the emulator as a SCSI disk, the SPARCstation expects it to be formatted. When the SPARCstation first finds that the new device isn't formatted, it produces the corrupt label message.

Chapter 3

Interpreting the XDS510WS LEDs

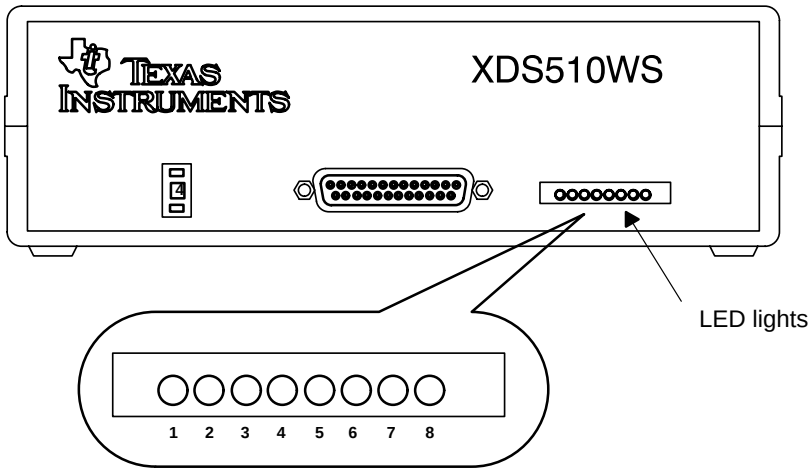
The XDS510WS emulator provides status information about the operation of the emulator through eight light-emitting diodes (LEDs) on the front panel of the emulator chassis.

Topic **Page**

3.1 XDS510WS LEDs

On the front of the XDS510WS is a small panel of LEDs that provide status information during the operation of the emulator (refer to Figure 3–1).

Figure 3–1. XDS510WS LEDs



The LEDs are numbered from left to right, starting with LED 1 through LED 8. The three LED conditions are:

Meaning		LED Symbol
Off	=	○
On	=	●
Intermittent†	=	◐

† Intermittently on and off; no steady state

3.2 Power Indicator: LED 1

LED 1 is on whenever the system is plugged in and switched on. If LED 1 doesn't come on, you should:

1) Ensure that the power supply is firmly plugged into a proper outlet.

2) Check to see that the power supply cable is firmly plugged into the XDS510WS.

3) Check to see that the XDS510WS is switched on.

3.3 Power-Loss Indicator: LED 2

When LED 2 is on, the XDS510WS has detected a power loss on the target system.

Note:

After you apply power to the target, this LED remains on until you invoke a debugger.

When you invoke the debugger, if LED 2 fails to go off and the debugger fails to start, you should ensure that the emulation cable is firmly and correctly attached to both the XDS510WS and the target. Also, check to see that the target is turned on and powered sufficiently. Additionally, check to see whether the target was designed to provide V_{CC} to the emulation header pin, PD.

Once LED 2 has gone off, if it comes on during your debugging session, the target system has lost power.

3.4 Emulation-Instruction Indicator: LED 3

LED 3 is on whenever the XDS510WS is executing an emulation instruction. Normally, you shouldn't notice the sporadic on state of this LED.

Occasionally, when you're performing a time-consuming emulation command such as a large FILL, LED 3 and LED 1 will be the only LEDs on. If LED 3 stays on for too long (greater than five minutes), there is a problem. To continue working, exit the debugger, cycle the power on the XDS510WS, and begin again.

3.5 Error/Status Indicators: LEDs 4, 5, and 6

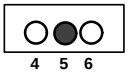
LEDs 4, 5, and 6 indicate error messages and signify the state of the emulator.

When you first power up the XDS510WS and immediately after you execute an emurst command, the emulator performs a self-test. LEDs 4 and 5 will be off, and LED 6 will be on to indicate that the self-test is being performed:

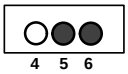
Note:

The self-test should take only a few seconds; moreover, if these three LEDs show this pattern for more than a minute, something is wrong:

If the self-test completes successfully, LEDs 4 and 6 will be off, and LED 5 will be on:



If these three LEDs show this pattern:



there has been a communications error. These errors are generally not serious, but if you can't continue without intervention, cycle the power on the XDS510WS, reexecute emurst, and restart the debugger.

3.6 SCSI-Transfer Indicators: LEDs 7 and 8

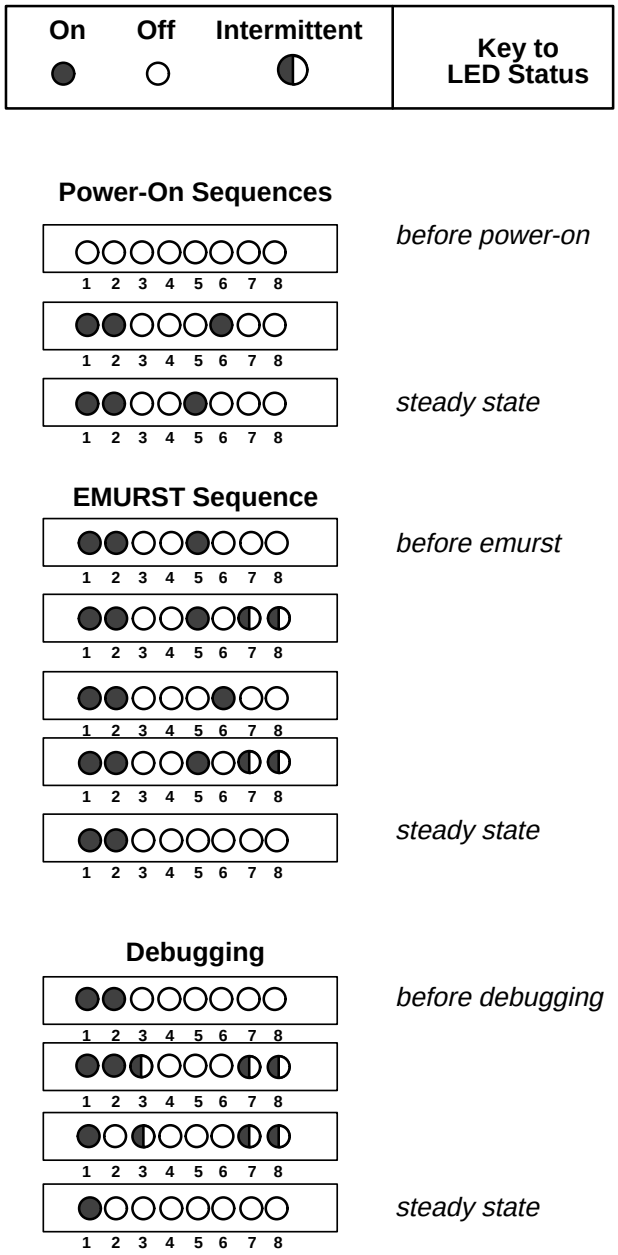
LEDs 7 and 8 indicate that a SCSI transfer is in progress with the emulator. If the debugger seems to hang and the LEDs become fixed (not flashing) in any pattern other than 7 and 8 off as shown below:

there is probably a problem. You can cycle the power on the XDS510WS, reexecute emurst, and restart the debugger.

3.7 XDS510WS LED Interpretation

Figure 3–2 shows the standard LED sequences. These patterns allow you to understand quickly the operational status of the emulator and its functions.

Figure 3–2. Standard LED Sequences



Chapter 4

Release Notes

If you have used the ’C5x debugging tools, you will find some differences between the ’C5x version of the debugger and the ’C2xx version. This chapter provides an overview of the differences and describes new features of the ’C2xx debugger.

Topic **Page**

4.1 Differences Between the ’C5x and ’C2xx Debuggers

The ’C2xx version of the emulator does not have all of the analysis features included in the ’C5x version. You cannot use the RUNB (run benchmark) command or the analysis count events features with the ’C2xx version of the emulator. In addition, the hardware breakpoint features in the ’C2xx emulator are reduced—you can break only on program accesses and low levels on the EMU0/1 pins.

There is no profiler for the ’C2xx emulator version of the debugger.

4.2 New Features of the ’C2xx Debugger

The ’C2xx debugger has three new features that are not included in the ’C5x debugger:

- Usage of arrow keys in the COMMAND window
- Minimal debugging mode
- The running indicator

Usage of arrow keys in the COMMAND window

When a field is selected for editing, the ← and → keys move the cursor within the field. You can use CONTROL ← or CONTROL → moves to the next field, except when the COMMAND window is active; in this case, the cursor moves to the beginning of the preceding or next word.

Notes:

1) When the COMMAND window is not active, you cannot use the arrow keys to move through or edit text on the command line.

2) Typing a command doesn’t make the COMMAND window the active window.

3) If you press ↵ when the cursor is in the middle of text, the debugger truncates the input text at the point where you press ↵. This is not only true when entering text in the COMMAND window but also when editing fields in other windows. Likewise, if you use CONTROL ← or CONTROL → to move to the beginning of the previous or next word, the debugger truncates the input text at the point where you press that key combination.

Minimal debugging mode

By default, the debugger automatically displays whatever code is currently running—assembly language or C. A DISASSEMBLY window, CPU register window, MEMORY window, CALLS window, or more windows may be displayed. The 'C2xx debugger has a *minimal* debugging mode that displays the COMMAND window, WATCH window, and DISP window only. The WATCH and DISP windows are displayed only if you cause them to display (by entering the WA or DISP commands).

Minimal mode allows you to query the target system without displaying any additional information. You can display the contents of CPU registers, memory addresses, or symbols within the COMMAND window by using the WA, DISP, and ?/EVAL commands. You can use any of the standard debugger commands in the COMMAND window. If you use the C, ASM, or MIX commands, the debugging mode changes to the auto mode, assembly mode, or mixed mode, respectively. To return to minimal mode, use the MINIMAL command.

You can also specify the minimal mode by choosing Minimal from the Mode menu or by using the –min option when invoking the debugger.

The running indicator

When the debugger causes the processor to run code, the debugger displays *Running...* in the title of the COMMAND window. This occurs when you use the RUN, GO, or RET command; the single-stepping commands do not change the title of the COMMAND window.

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.