

# ***TMS320C5x Emulator Getting Started Guide***

***Release 7.40***

Literature Number: SPRU196  
Manufacturing Part Number: D415006-9741 revision \*  
September 1996



## **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.

## Preface

# Read This First

---

---

---

### ***About This Manual***

The *TMS320C5x Emulator Getting Started Guide* tells you how to install release 7.40 of the TMS320C5x emulator debugging tools on your system. It also does the following:

- ☐ Tells you how to set environment variables for parameters that you use often
- ☐ Gives helpful information for troubleshooting
- ☐ Gives information on new or changed features for this release of the emulator

### ***How to Use This Manual***

The goal of this book is to get you started using the emulator specifically designed for the TMS320C5x. Following are the topics covered in this getting started guide:

For information about ...	Refer to ...
Installing the debugger software, setting environment variables, and verifying the installation on DOS and Windows™ 3.x systems	Chapter 1
Installing the debugger software, setting environment variables, and verifying the installation on OS/2™ systems	Chapter 2
Installing the debugger software, setting environment variables, and verifying the installation on a SPARCstation™	Chapter 3
Installing the debugger software, setting environment variables, and verifying the installation on an HP system	Chapter 4
Enabling extended addressing	Chapter 5
Release enhancements	Chapter 6
Using extended addressing with the debugger	Chapter 7
Troubleshooting	Appendix A

## Notational Conventions

- ❑ The TMS320C50, TMS320C51, TMS320C52, and TMS320C53 devices are referred to as 'C5x.
- ❑ Program listings, program examples, and interactive displays are shown in a *special typeface* similar to a typewriter's. Examples use a **bold version** of the special typeface for emphasis; interactive displays use a **bold version** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

Here is an example of a command that you might enter:

```
cd /cdrom/hp
```

- ❑ In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a command syntax:

```
set PATH=C:\pathname1; pathname2
```

*set* is the command. This command has one parameter, indicated by *pathname*.

- ❑ Square brackets ( [ and ] ) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of a command that has optional parameters:

```
emurst [options]
```

This command allows you to specify one or more options.

## **Related Documentation From Texas Instruments**

The following books describe the TMS320C5x devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

**XDS51x Emulator Installation Guide** (literature number SPNU070) describes the installation of the XDS510™, XDS510PP™, and XDS510WS™ emulator controllers. The installation of the XDS511™ emulator is also described.

**TMS320C5x C Source Debugger User's Guide** (literature number SPRU055) tells you how to invoke the 'C5x emulator, EVM, and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

**TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide** (literature number SPRU018) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the 'C1x, 'C2x, 'C2xx, and 'C5x generations of devices.

**TMS320C2x/C2xx/C5x Optimizing C Compiler User's Guide** (literature number SPRU024) describes the 'C2x/C2xx/C5x C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C2x, 'C2xx, and 'C5x generations of devices.

**TMS320C5x User's Guide** (literature number SPRU056) describes the 'C5x 16-bit, fixed-point, general-purpose digital signal processors. Covered are its architecture, internal register structure, instruction set, pipeline, specifications, DMA, I/O ports, and on-chip peripherals.

### ***FCC 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***

HP-UX, HP 9000 Series 700, and PA-RISC are trademarks of Hewlett-Packard Company.

IBM, OS/2, OS/2 Warp, PC, and PC-DOS are trademarks of International Business Machines Corp.

Microsoft, MS-DOS, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

OpenWindows, SunOS, and Solaris are trademarks of Sun Microsystems, Inc.

Pentium is a trademark of Intel Corporation.

SPARCstation is a trademark of SPARC International, Inc., but licensed exclusively to Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X Window System is a trademark of the Massachusetts Institute of Technology.

**If You Need Assistance**

<b>If you want to . . .</b>	<b>Contact Texas Instruments at . . .</b>
Visit TI online	World Wide Web: <a href="http://www.ti.com">http://www.ti.com</a>
Receive general information or assistance	World Wide Web: <a href="http://www.ti.com/sc/docs/pic/home.htm">http://www.ti.com/sc/docs/pic/home.htm</a> North America, South America: (214) 644-5580 Europe, Middle East, Africa Dutch: 33-1-3070-1166 English: 33-1-3070-1165 French: 33-1-3070-1164 Italian: 33-1-3070-1167 German: 33-1-3070-1168 Japan (Japanese or English) Domestic toll-free: 0120-81-0026 International: 81-3-3457-0972 or 81-3-3457-0976 Korea (Korean or English): 82-2-551-2804 Taiwan (Chinese or English): 886-2-3771450
Ask questions about Digital Signal Processor (DSP) product operation or report suspected problems	Hotline: (713) 274-2320 Fax: (713) 274-2324 Fax Europe: +33-1-3070-1032 Email: <a href="mailto:dsph@ti.com">dsph@ti.com</a> World Wide Web: <a href="http://www.ti.com/dsps">http://www.ti.com/dsps</a> BBS North America: (713) 274-2323 8-N-1 BBS Europe: +44-2-3422-3248 320 BBS Online: <a href="ftp://ti.com/mirrors/tms320bbs">ftp.ti.com/mirrors/tms320bbs</a> (192.94.94.53)
Request tool updates	Software: (214) 638-0333 Software fax: (214) 638-7742 Hardware: (713) 274-2285
Order Texas Instruments documentation (see Note 1)	Literature Response Center: (800) 477-8924
Make suggestions about or report errors in documentation (see Note 2)	Email: <a href="mailto:comments@books.sc.ti.com">comments@books.sc.ti.com</a> Mail: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251-1443

- Notes:**
- 1) The literature number for the book is required; see the center of the title page.
  - 2) Please mention the full title of the book, the literature number from the lower-right corner of the back cover, and the publication date from the spine or front cover.

# Contents

---

---

---

<b>1</b>	<b>Installing the Debugger With DOS or Windows 3.x</b>	<b>1-1</b>
	<i>Lists the hardware and software you'll need to install the emulator board and C source debugger; provides installation instructions for PC systems running MS-DOS, PC-DOS, or Windows.</i>	
1.1	System Requirements	1-2
	Hardware checklist	1-2
	Software checklist	1-3
1.2	Step 1: Installing the XDS510 Emulator Controller	1-5
1.3	Step 2: Installing the Debugger Software	1-5
	Installing the debugger on DOS systems	1-5
	Installing the debugger on Windows 3.x systems	1-5
1.4	Step 3: Setting Up the Debugger Environment	1-6
	Identifying the directory that contains the executable files (PATH statement)	1-6
	Identifying alternate directories for the debugger (D_DIR)	1-7
	Identifying directories that contain source files (D_SRC)	1-7
	Setting default debugger options (D_OPTIONS)	1-7
1.5	Step 4: Resetting the Emulator	1-9
1.6	Step 5: Describing Your Target System to the Debugger	1-10
1.7	Step 6: Verifying the Installation	1-11
	Installation error messages	1-12
1.8	Using the Debugger With Windows 3.x	1-14
<b>2</b>	<b>Installing the Debugger With OS/2</b>	<b>2-1</b>
	<i>Lists the hardware and software you'll need to install the emulator board and C source debugger; provides installation instructions for PC systems running OS/2.</i>	
2.1	System Requirements	2-2
	Hardware checklist	2-2
	Software checklist	2-3
2.2	Step 1: Installing the XDS510 Emulator Controller	2-5
2.3	Step 2: Installing the Debugger Software	2-5
2.4	Step 3: Setting Up the Debugger Environment	2-6
	Identifying the directory that contains the executable files (PATH statement)	2-6
	Identifying alternate directories for the debugger (D_DIR)	2-7
	Identifying directories that contain source files (D_SRC)	2-7
	Setting default debugger options (D_OPTIONS)	2-7
	Setting the IOPL option	2-8



2.5	Step 4: Resetting the Emulator .....	2-9
2.6	Step 5: Describing Your Target System to the Debugger .....	2-10
2.7	Step 6: Verifying the Installation .....	2-11
	Installation error messages .....	2-12
<b>3</b>	<b>Installing the Debugger on a SPARCstation .....</b>	<b>3-1</b>
	<i>Lists the hardware and software you'll need to install the workstation emulator and C source debugger; provides installation instructions for SPARCstations running SunOS.</i>	
3.1	System Requirements .....	3-2
	Hardware checklist .....	3-2
	Software checklist .....	3-2
3.2	Step 1: Installing the XDS510WS Emulator Controller .....	3-4
3.3	Step 2: Installing the Debugger Software .....	3-4
	Mounting the CD-ROM .....	3-4
	Copying the files .....	3-5
	Unmounting the CD-ROM .....	3-5
3.4	Step 3: Ensuring That the Emulator Supports the Debugger .....	3-6
3.5	Step 4: Describing Your Target System to the Debugger .....	3-7
3.6	Step 5: Setting Up the Debugger Environment .....	3-8
	Modifying the path statement .....	3-8
	Setting up the environment variables .....	3-8
	Invoking the new or modified .cshrc file .....	3-10
3.7	Step 6: Verifying the Installation .....	3-11
3.8	Using the Debugger With the X Window System .....	3-12
	Using the keyboard's special keys .....	3-12
	Changing the debugger font .....	3-13
	Color mappings on monochrome screens .....	3-13
<b>4</b>	<b>Installing the Debugger on an HP Workstation .....</b>	<b>4-1</b>
	<i>Lists the hardware and software you'll need to install the workstation emulator and C source debugger; provides installation instructions for HP Workstations running HP-UX.</i>	
4.1	System Requirements .....	4-2
	Hardware checklist .....	4-2
	Software checklist .....	4-2
4.2	Step 1: Installing the XDS510WS Emulator Controller .....	4-4
4.3	Step 2: Installing the Debugger Software .....	4-4
	Mounting the CD-ROM .....	4-4
	Copying the files .....	4-5
	Unmounting the CD-ROM .....	4-5
4.4	Step 3: Ensuring That the Emulator Supports the Debugger .....	4-6
4.5	Step 4: Describing Your Target System to the Debugger .....	4-7
4.6	Step 5: Setting Up the Debugger Environment .....	4-8
	Modifying the path statement .....	4-8
	Setting up the environment variables .....	4-8
	Invoking the new or modified .cshrc file .....	4-10

4.7	Step 6: Verifying the Installation .....	4-11
4.8	Using the Debugger With the X Window System .....	4-12
	Using the keyboard's special keys .....	4-12
	Changing the debugger font .....	4-13
	Color mappings on monochrome screens .....	4-13
<b>5</b>	<b>Enabling Extended Addressing .....</b>	<b>5-1</b>
	<i>Describes how to use extended addressing with the TMS320C5x.</i>	
5.1	Understanding the Use of Extended Addressing .....	5-2
	About extended addressing .....	5-2
	Sample extended memory system .....	5-3
5.2	Building an Extended Memory System .....	5-4
	Adding memory and logic .....	5-4
	Creating registers .....	5-5
	Updating or creating source files .....	5-5
	Updating or creating a linker command file .....	5-6
<b>6</b>	<b>Release Notes .....</b>	<b>6-1</b>
	<i>Details the features added or changed for this release.</i>	
	COFF version 2 .....	6-1
	Multiple MEMORY windows .....	6-1
	Multiple WATCH windows .....	6-2
	New debugger commands .....	6-3
	Extended addressing .....	6-4
<b>7</b>	<b>Using the Debugger With Extended Addressing .....</b>	<b>7-1</b>
	<i>Describes extended addressing and how to use it with the emulator version of the TMS320C5x debugger.</i>	
7.1	Understanding the Use of Extended Addressing .....	7-2
	About extended addressing .....	7-2
	Sample extended memory system .....	7-3
7.2	Setting Up Extended Addressing .....	7-4
	Describing your extended memory configuration to the debugger .....	7-4
	Enabling extended addressing .....	7-5
7.3	Debugging With Extended Addressing .....	7-6
	Registers associated with extended addressing: PMR, DMR, and EPC .....	7-6
	New expression syntax .....	7-7
	How extended addressing affects symbols .....	7-8
	Using 16-bit expressions with 32-bit extended addressing .....	7-9
	Hardware breakpoints and extended addressing .....	7-9

<b>A</b>	<b>Troubleshooting .....</b>	<b>A-1</b>
	<i>Describes problems that you may encounter while installing and using the emulator on your workstation and gives suggestions for resolving those problems.</i>	
A.1	Problems When Booting Your Workstation .....	A-2
A.2	Problems With Multiple Emulators on SunOS .....	A-2
A.3	Problems When Resetting the Emulator .....	A-3
A.4	Problems When Invoking the Debugger .....	A-5
A.5	Additional Emulator and Debugger Problems .....	A-7

# Installing the Debugger With DOS or Windows 3.x

---

---

---

This chapter helps you install the C source debugger on a PC™ running DOS or Windows. When you complete the installation, see the *TMS320C5x C Source Debugger User's Guide* for instructions on using the debugger.

To install the XDS510 emulator controller, see the *XDS51x Emulator Installation Guide*.

Topic	Page
1.1 System Requirements .....	1-2
1.2 Step 1: Installing the XDS510 Emulator Controller .....	1-5
1.3 Step 2: Installing the Debugger Software .....	1-5
1.4 Step 3: Setting Up the Debugger Environment .....	1-6
1.5 Step 4: Resetting the Emulator .....	1-9
1.6 Step 5: Describing Your Target System to the Debugger .....	1-10
1.7 Step 6: Verifying the Installation .....	1-11
1.8 Using the Debugger With Windows 3.x .....	1-14

## 1.1 System Requirements

To install and use the 'C5x emulator and C source debugger, you need the items listed in the following hardware and software checklists.

### **Hardware checklist**

<input type="checkbox"/>	<b>System</b>	80386-, 80486-, or Pentium™-based PC with a 100% compatible ISA/EISA card slot
<input type="checkbox"/>	<b>Memory</b>	4–16 Mbytes of free memory is recommended to ensure optimum performance
<input type="checkbox"/>	<b>Disk space</b>	10 Mbytes available disk space for executables and libraries
<input type="checkbox"/>	<b>Display</b>	Monochrome or color monitor (color recommended)
<input type="checkbox"/>	<b>Required hardware</b>	CD-ROM drive
<input type="checkbox"/>	<b>Optional hardware</b>	Microsoft-compatible mouse
<input type="checkbox"/>		EGA- or VGA-compatible graphics display card and a large (17 or 19 inch) monitor. The debugger has two options that allow you to change the overall size of the debugger display. To use a larger screen size, you must invoke the debugger with the appropriate option. For more information about options, see the invocation section in the <i>Overview of a Code Development and Debugging System</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i> .
<input type="checkbox"/>	<b>XDS510 emulator controller</b>	XDS510 or XDS510PP emulator controller

#### **Note:**

The speed at which your system operates depends on the amount of RAM available on your PC and the number of debuggers running simultaneously.

**Software checklist**

<input type="checkbox"/>	<b>Operating system</b>	MS-DOS™ or PC-DOS™ (version 3.0 or later) Optional: Windows 3.x
<input type="checkbox"/>	<b>Software tools</b>	TMS320C1x/C2x/C2xx/C5x DSP assembler and linker Optional: TMS320C2x/C2xx/C5x C compiler
<input type="checkbox"/>	<b>Required files included with the debugger package</b>	The debugger executable file for your system:  <i>emu5x.exe</i> is for DOS systems. <i>emu5xw.exe</i> is for Windows 3.x systems with one device. <i>emu5xwm.exe</i> is for Windows 3.x systems with multiple devices. You can use only one device at a time.
<input type="checkbox"/>		<i>emurst</i> resets the emulator.
<input type="checkbox"/>		<i>board.dat</i> describes your target system to the debugger in terms of what devices are on the emulation scan path. The <i>board.dat</i> file included in the debugger package is for a target board with one TMS320C5x named <i>cpu_a</i> .
<input type="checkbox"/>	<b>Optional files included with the debugger package</b>	<i>emuinit.cmd</i> is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C5x memory map. When you first start using the debugger, this memory map should be sufficient for your needs. Later, you may want to define your own memory map. For information about setting up your own memory map, see the <i>Defining a Memory Map</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i> .
<input type="checkbox"/>		<i>composer.exe</i> is the utility that translates the <i>board.cfg</i> file to a binary, conditioned format. For the emulator to initialize properly, you must create a new <i>board.dat</i> file with this release of the composer.
<input type="checkbox"/>		<i>board.cfg</i> is a text file used to describe your target system in terms of what devices are on the emulation scan path. The <i>board.cfg</i> file included in the debugger package is for a target board with one TMS320C54x named <i>cpu_a</i> .
<input type="checkbox"/>		<i>init.clr</i> is a general-purpose screen configuration file. If this file isn't present when you invoke the debugger, the debugger uses the default screen configuration.
<input type="checkbox"/>		<i>init.25</i> , <i>init.43</i> , and <i>init.50</i> have been provided for basic 80×25, 80×43, and 80×50 screen sizes, respectively. The <i>init.clr</i> file brings up the debugger in 80×25 mode. To bring the debugger up in another mode, copy one of the <i>init.xx</i> files to the <i>init.clr</i> file.



The default configuration is for color monitors; an additional file, *mono.clr*, can be used for monochrome monitors. When you first start to use the debugger, the default screen configuration should be sufficient for your needs. Later, you may want to define your own custom configuration.

For information about these files and about setting up your own screen configuration, see the *Customizing the Debugger Display* chapter in the *TMS320C5x C Source Debugger User's Guide*.

## 1.2 Step 1: Installing the XDS510 Emulator Controller

Before installing the 'C5x debugger software, you must install an XDS510 emulator controller: the XDS510 or XDS510PP.

Follow the instructions in the *XDS51x Emulator Installation Guide* to install the XDS510 emulator controller.



## 1.3 Step 2: Installing the Debugger Software

This section explains the process of installing the debugger software on a hard-disk system for two different operating systems:

- ☐ DOS
- ☐ Windows 3.x

### ***Installing the debugger on DOS systems***

To install the debugger on a DOS system, follow these steps:

- 1) Insert the debugger CD-ROM into your CD-ROM drive.
- 2) Change to the CD-ROM drive (replace d with the letter of your CD-ROM drive):  
`d: `
- 3) Enter the following command:  
`install `
- 4) Follow the on-screen instructions.

### ***Installing the debugger on Windows 3.x systems***

To install the debugger on a Windows 3.x system, follow these steps:

- 1) Insert the debugger CD-ROM into your CD-ROM drive.
- 2) Start Windows 3.x.
- 3) From the File menu, select Run.
- 4) In the dialog box, enter the following command (replace d with the letter of your CD-ROM drive):  
`d:\setup.exe`
- 5) Click on OK.
- 6) Follow the on-screen instructions.



## 1.4 Step 3: Setting Up the Debugger Environment

You can define *environment variables* that set certain debugger parameters you normally use. An environment variable is a system symbol that you define and assign to a string. When you use environment variables, default values are set, making each individual invocation of the debugger simpler because these parameters are automatically specified.

The debugger uses environment variables for finding or obtaining certain types of information. By default, the installation program sets up these environment variables:

```
SET PATH=C:\c5xh11;%PATH%
SET D_DIR=C:\c5xh11
SET D_SRC=C:\c5xh11
SET D_OPTIONS=option1 option2...
```

D\_OPTIONS is set to the options you enter when prompted. If you do not enter any options, then D\_OPTIONS is not set.

If you choose not to have the environment variables set up automatically, you can modify your autoexec.bat file to include the SET commands above.

The remainder of this section describes these environment variables and other variables that you can define.

### ***Identifying the directory that contains the executable files (PATH statement)***

You must include the emulator directory in your PATH statement. This allows you to specify the debugger executable without specifying the name of the directory that contains the executable file.

- ☐ If you modify your autoexec.bat file to change the path information, add the following to the end of the PATH statement:

**;C:\c5xh11**

- ☐ If you are creating a batch file, use this format:

**SET PATH=C:\c5xh11;%PATH%**

(Be careful not to precede the equal sign with a space.)

The addition of **;%PATH%** ensures that this PATH statement does not undo the PATH statements in any other batch files (including the autoexec.bat file).

### ***Identifying alternate directories for the debugger (D\_DIR)***

The debugger uses the D\_DIR environment variable to name alternative directories that contain auxiliary files (emurst, emuinit.cmd, etc.) that the debugger needs. The command for assigning the environment variable is as follows:

**SET D\_DIR=C:\c5xhll**

(Be careful not to precede the equal sign with a space.)

### ***Identifying directories that contain source files (D\_SRC)***

The debugger uses the D\_SRC environment variable to name directories that contain program source files. The command for assigning the environment variable is as follows:

**SET D\_SRC=pathname1; pathname2 . . .**

(Be careful not to precede the equal sign with a space.)

The *pathnames* are directories that contain program source files. You can separate pathnames with a semicolon or with blanks.

### ***Setting default debugger options (D\_OPTIONS)***

You might find it useful to set default debugger options using the D\_OPTIONS environment variable. When you use the D\_OPTIONS environment variable, the debugger uses the options and/or input filenames that you name with D\_OPTIONS every time you run the debugger. The command for assigning the environment variable is as follows:

**SET D\_OPTIONS=[object filename] [debugger options]**

(Be careful not to precede the equal sign with a space.)

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
–f <i>filename</i>	Identify a new board configuration file
–i <i>pathname</i>	Identify additional directories
–n <i>processor name</i>	Identify processor for debugging
–p <i>port address</i>	Identify the port address
–profile	Enter profiling environment
–s	Load the symbol table only
–t <i>filename</i>	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 *TMS320C5x C Source Debugger User's Guide*.

## 1.5 Step 4: Resetting the Emulator

You must reset the emulator *before* invoking the debugger. Reset can occur only after you have powered up the target board. You can reset the emulator in one of these ways:

- ☐ Add the emurst command to the autoexec.bat file in the following format:

**emurst [-x] [-p *number*]**

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

The `-p` option *number* represents the device driver number.

If the following message appears after the emulator is reset, you have a hardware error:

CANNOT DETECT TARGET POWER

One of several problems may cause this error message to appear. Follow the suggestions listed below and restart your PC. Check:

- ☐ Is the emulator board installed snugly?
- ☐ Is the cable connecting your emulator and target system loose?
- ☐ Is the target power on?
- ☐ Is your target board getting the correct voltage?
- ☐ Is your emulator scan path uninterrupted?
- ☐ Is your port address set correctly?
  - Check to be sure the `-p` option of the `D_OPTIONS` environment variable matches the I/O address defined by your switch settings. For information about the switch settings, see the *XDS51x Emulator Installation Guide*.
  - Check to see if you have a conflict in address space with another bus setting. If you have a conflict, change the switches on your board to one of the alternate settings. Modify the `-p` option of the `D_OPTIONS` environment variable to reflect the change in your switch settings.

## 1.6 Step 5: Describing Your Target System to the Debugger

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

- ☐ If you're using an emulation scan path that contains only one 'C5x and no other devices, you can use the *board.dat* file that comes with the 'C5x emulator kit. This file describes to the debugger the single 'C5x in the scan path and gives the 'C5x 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 Section 1.7.
- ☐ If you want to use a different name for the target device, you must follow these steps:
  - 1) Create the board configuration file.
  - 2) Translate the board configuration file to binary so that the debugger can read it.
  - 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 *TMS320C5x C Source Debugger User's Guide*.

## 1.7 Step 6: Verifying the Installation

To ensure that you have correctly installed the emulator and debugger software, enter one of these commands:

- ☐ For DOS systems, enter this command from the DOS prompt:

```
emu5x c:\c5xh11\sample
```

- ☐ For Windows systems, choose one of these commands:

- ☒ If you have one device in your scan path, enter this command from the system prompt:

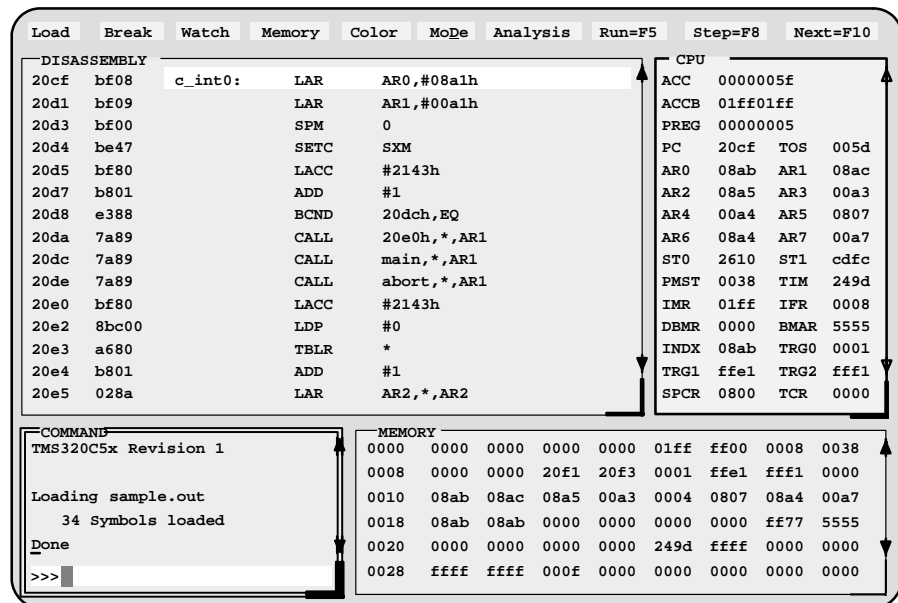
```
emu5xw c:\c5xh11\sample
```

- ☒ If you have more than one device in your scan path, enter this command from the system prompt:

```
emu5xwm c:\c5xh11\sample -n cpu_a
```

Note that while the emu5xwm executable supports more than one device on the target board, you can use only one device at a time.

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 see a display and the lines of code show ADD instructions, your emulator board may not be installed snugly. Check your board to see whether it is correctly installed, and reenter the command above.

- ☐ If you see a display and the lines of code say *Invalid address* or the fields in the MEMORY window are shown in red, the debugger may not be able to find the emunit.cmd file. Check for the file in the directories specified by the D\_SRC environment variable or ensure that the file is in the current directory. Reenter the command above.
- ☐ If you don't see a display, 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.

### **Installation error messages**

While invoking the debugger, you may see the following message:

CANNOT INITIALIZE THE TARGET SYSTEM ! !  
- Check I/O configuration  
- Check cabling and target power

One or several of the following conditions may be the cause:

- ☐ Is the target power on?
- ☐ Is the emulator board installed snugly?
- ☐ Is the device installed snugly?
- ☐ Is the cable connecting your emulator and target system loose?
- ☐ Is your target board getting the correct voltage?
- ☐ Is your emulator scan path interrupted? For DOS systems, only one device is allowed in the scan path. Check the connections; either they are not connected, or they are connected improperly.
- ☐ Did you use the -n option? Or was it used with an incorrect device name? You must supply a valid device name with the -n option.
- ☐ After you powered up the target board, did you execute the emurst.exe command? Check your autoexec.bat file. This command must be executed *after* you powered up the target board.

- ☐ Did you use the `-p` option? Is your port address set correctly?
  - Check to be sure the `-p` option used with the `D_OPTIONS` environment variable matches the I/O address defined by your switch settings. For information about the switch settings, see the *XDS51x Emulator Installation Guide*.
  - Check to see whether you have a conflict in address space with another bus setting. If you have a conflict, change the switches on your board to one of the alternate settings. Modify the `-p` option of the `D_OPTIONS` environment variable to reflect the change in your switch settings.
- ☐ Is the `board.dat` file in the current directory or in a directory specified by `D_DIR`?

After you have checked all of the above, repeat the verification instructions on page 1-11.



## **1.8 Using the Debugger With Windows 3.x**

If you're using Windows 3.x, you can freely move or resize the debugger display on the screen. If the resized display is bigger than the debugger requires, the extra space is not used. If the resized display is smaller than required, the display is clipped. Note that when the display is clipped, it can't be scrolled.

You should run Windows 3.x in either the standard mode or the 386 enhanced mode to get the best results.

# Installing the Debugger With OS/2

---

---

---

This chapter helps you to install the C source debugger on a PC system running OS/2. When you complete the installation, see the *TMS320C5x C Source Debugger User's Guide* for instructions on using the debugger.

To install the XDS510 emulator controller, see the *XDS51x Emulator Installation Guide*.

Topic	Page
2.1 System Requirements .....	2-2
2.2 Step 1: Installing the XDS510 Emulator Controller .....	2-5
2.3 Step 2: Installing the Debugger Software .....	2-5
2.4 Step 3: Setting Up the Debugger Environment .....	2-6
2.5 Step 4: Resetting the Emulator .....	2-9
2.6 Step 5: Describing Your Target System to the Debugger .....	2-10
2.7 Step 6: Verifying the Installation .....	2-11

## 2.1 System Requirements

To install and use the 'C5x emulator and C source debugger, you need the items listed in the following hardware and software checklists.

### *Hardware checklist*

<input type="checkbox"/>	<b>System</b>	80386-, 80486-, or Pentium-based PC with a 100% compatible ISA/EISA card slot
<input type="checkbox"/>	<b>Memory</b>	Minimum of 8 Mbytes
<input type="checkbox"/>	<b>Disk space</b>	10 Mbytes available disk space for executables and libraries
<input type="checkbox"/>	<b>Display</b>	Monochrome or color (color recommended)
<input type="checkbox"/>	<b>Required hardware</b>	CD-ROM drive
<input type="checkbox"/>	<b>Optional hardware</b>	Microsoft-compatible mouse
<input type="checkbox"/>		EGA- or VGA-compatible graphics display card and a large monitor (17 or 19 inch). The debugger has two options that allow you to change the overall size of the debugger display. To use a larger screen size, you must invoke the debugger with the appropriate option. For more information about options, see the invocation section in the <i>Overview of a Code Development and Debugging System</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i> .
<input type="checkbox"/>	<b>XDS510 emulator controller</b>	XDS510 or XDS510PP emulator controller

**Note:**

The speed at which your system operates depends on the amount of RAM available on your PC and the number of debuggers running simultaneously.

**Software checklist**

<input type="checkbox"/>	<b>Operating system</b>	OS/2 version 1.1 or later
<input type="checkbox"/>	<b>Software tools</b>	TMS320C1x/C2x/C2xx/C5x DSP assembler and linker Optional: TMS320C2x/C2xx/C5x C compiler
<input type="checkbox"/>	<b>Required files included with the debugger package</b>	<i>emu5xo.exe</i> is the debugger executable file.
<input type="checkbox"/>		<i>emurst.exe</i> resets the emulator.
<input type="checkbox"/>		<i>board.dat</i> describes your target board to the 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 TMS320C5x named <i>cpu_a</i> .
<input type="checkbox"/>	<b>Optional files</b>	<i>emuinit.cmd</i> is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C5x memory map. When you first start using the debugger, this memory map should be sufficient for your needs. Later, you may want to define your own memory map. For information about setting up your own memory map, see the <i>Defining a Memory Map</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i> .
<input type="checkbox"/>		<i>composer.exe</i> is the utility that translates the board.cfg file to a binary, conditioned format. For the emulator to initialize properly, you must create a new board.dat file with this release of the composer.
<input type="checkbox"/>		<i>board.cfg</i> is a text file used to describe your target system 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 TMS320C5x named <i>cpu_a</i> .
<input type="checkbox"/>		<i>init.clr</i> 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.
<input type="checkbox"/>		<i>init.25</i> , <i>init.43</i> , and <i>init.50</i> have been provided for basic 80×25, 80×43, and 80×50 screen sizes, respectively. The <i>init.clr</i> file brings up the debugger in 80×25 mode. To bring the debugger up in another mode, copy one of the <i>init.xx</i> files to the <i>init.clr</i> file.



The default configuration is for color monitors; an additional file, *mono.clr*, can be used with monochrome monitors. When you first start to use the debugger, the default screen configuration should be sufficient for your needs. Later, you may want to define your own custom configuration.

For information about these files and about setting up your own screen configuration, see the *Customizing the Debugger Display* chapter in the *TMS320C5x C Source Debugger User's Guide*.



## 2.2 Step 1: Installing the XDS510 Emulator Controller

Before installing the 'C5x debugger software, you must install an XDS510 emulator controller: the XDS510 or XDS510PP.

Follow the instructions in the *XDS51x Emulator Installation Guide* to install the XDS510 emulator controller.

## 2.3 Step 2: Installing the Debugger Software

To install the debugger on an OS/2 system, follow these steps:

- 1) Insert the debugger CD-ROM into your CD-ROM drive.
- 2) Start a DOS-Windows session.
- 3) Change to the CD-ROM drive (replace d with the letter of your CD-ROM drive):  
`d: `
- 4) Enter the following command:  
`install `
- 5) Follow the on-screen instructions.

## 2.4 Step 3: Setting Up the Debugger Environment

You can define *environment variables* that set certain debugger parameters you normally use. An environment variable is a system symbol that you define and assign to a string. When you use environment variables, default values are set, making each individual invocation of the debugger simpler because these parameters are automatically specified.

The debugger uses environment variables for finding or obtaining certain types of information. By default, the installation program sets up these environment variables:

```
SET PATH=C:\c5xh11;%PATH%
SET D_DIR=C:\c5xh11
SET D_SRC=C:\c5xh11
SET D_OPTIONS=option1 option2 ...
```

D\_OPTIONS is set to the options you enter when prompted. If you do not enter any options, then D\_OPTIONS is not set.

If you choose not to have the environment variables set up automatically, you can modify your config.sys file to include the SET commands above.

The remainder of this section describes these environment variables and other variables that you can define.

### ***Identifying the directory that contains the executable files (PATH statement)***

You must include the emulator directory in your PATH statement. This allows you to specify the debugger executable without specifying the name of the directory that contains the executable file.

- ☐ If you modify your config.sys file to change the path information, add the following to the end of the PATH statement:

**`;C:\c5xh11`**

- ☐ If you are creating a batch file, use this format:

**`SET PATH=C:\c5xh11;%PATH%`**

(Be careful not to precede the equal sign with a space.)

The addition of **`;%PATH%`** ensures that this PATH statement does not undo the PATH statements in any other batch files (including the config.sys file).

### ***Identifying alternate directories for the debugger (D\_DIR)***

The debugger uses the D\_DIR environment variable to name alternative directories that contain auxiliary files (emurst, emuinit.cmd, etc.) that the debugger needs. The command for assigning the environment variable is as follows:

**SET D\_DIR=C:\c5xhll**

(Be careful not to precede the equal sign with a space.)

### ***Identifying directories that contain source files (D\_SRC)***

The debugger uses the D\_SRC environment variable to name directories that contain program source files. The command for assigning the environment variable is as follows:

**SET D\_SRC=pathname1; pathname2 . . .**

(Be careful not to precede the equal sign with a space.)

The *pathnames* are directories that contain program source files. You can separate pathnames with a semicolon or with blanks.

### ***Setting default debugger options (D\_OPTIONS)***

You might find it useful to set default debugger options using the D\_OPTIONS environment variable. When you use the D\_OPTIONS environment variable, the debugger uses the options and/or input filenames that you name with D\_OPTIONS every time you run the debugger. The command for assigning the environment variable is as follows:

**SET D\_OPTIONS=[object filename] [debugger options]**

(Be careful not to precede the equal sign with a space.)

This tells the debugger to load the specified object file and use the specified options each time you invoke the debugger. Table 2–1 lists the options that you can identify with D\_OPTIONS.



Table 2–1. Options for Use With D\_OPTIONS

Option	Brief Description
-b[b]	Select the screen size
-f <i>filename</i>	Identify a new board configuration file
-i <i>pathname</i>	Identify additional directories
-n <i>processor name</i>	Identify processor for debugging
-p <i>port address</i>	Identify the port address
-profile	Enter profiling environment
-s	Load the symbol table only
-t <i>filename</i>	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 *TMS320C5x C Source Debugger User's Guide*.

### Setting the IOPL option

You must override the default IOPL setting. IOPL is an OS/2-specific option that prevents you from accessing your emulator. To override the default setting, set IOPL to YES by adding the following line to your config.sys file:

```
IOPL=YES
```

#### Note:

You must set the IOPL option in the config.sys file; you cannot access it in any other way.

## 2.5 Step 4: Resetting the Emulator

You must reset the emulator *before* invoking the debugger. Reset can occur only after you have powered up the target board. You can reset the emulator in one of these ways:

- ☐ Add the following line to the end of your config.sys file:

```
RUN = C:\C5XHLL\EMURST.EXE
```

### Note:

If you reset the emulator using the RUN command in your config.sys file, emurst will *not* display an error message when trying to reset the emulator while a debugger is running. In addition, executing emurst in this manner will not produce any standard messages.

- ☐ Create or modify a file called C:\startup.cmd that contains the emurst command in the following format:

```
emurst [-x] [-p number]
```

The `-x` option tells the emurst utility to ignore any options specified with the D\_OPTIONS environment variable.

The `-p` option *number* represents the device driver number.

### Note:

If a debugger is running, emurst will not reset the emulator. The debugger displays the message, "RESET DISALLOWED : DEBUGGER RUNNING".

If the following message appears after the emulator is reset, you have a hardware error:

```
CANNOT DETECT TARGET POWER
```

One of several problems may cause this error message to appear. Follow the suggestions listed below and restart your PC. Check:

- ☐ Is the emulator board installed snugly?
- ☐ Is the cable connecting your emulator and target system loose?
- ☐ Is the target power on?
- ☐ Is your target board getting the correct voltage?
- ☐ Is your emulator scan path uninterrupted?

- ☐ Is your port address set correctly?
  - Check to be sure the `-p` option of the `D_OPTIONS` environment variable matches the I/O address defined by your switch settings. For information about the switch settings, see the XDS510 installation instructions in the *XDS51x Emulator Installation Guide*.
  - Check to see if you have a conflict in address space with another bus setting. If you have a conflict, change the switches on your board to one of the alternate settings. Modify the `-p` option of the `D_OPTIONS` environment variable to reflect the change in your switch settings.

## 2.6 Step 5: Describing Your Target System to the Debugger

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

- ☐ If you're using an emulation scan path that contains only one 'C5x and no other devices, you can use the *board.dat* file that comes with the 'C5x emulator kit. This file describes to the debugger the single 'C5x in the scan path and gives the 'C5x 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 Section 2.7.
- ☐ If you plan to use a different target system, you must follow these steps:
  - 1) Create the board configuration file.
  - 2) Translate the board configuration file to binary so that the debugger can read it.
  - 3) Specify the configuration file when invoking the debugger.

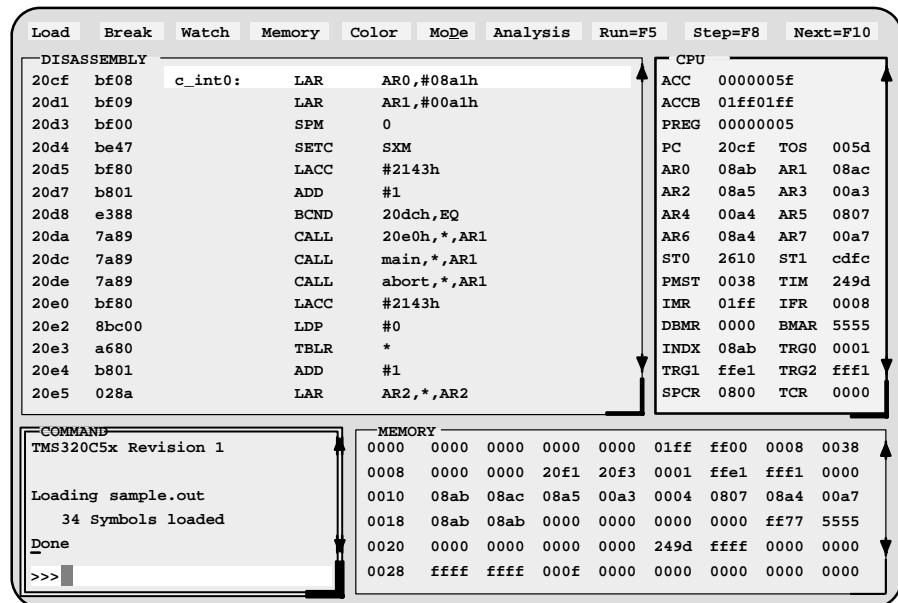
These steps are described in the *Describing Your Target System to the Debugger* appendix in the *TMS320C5x C Source Debugger User's Guide*.

## 2.7 Step 6: Verifying the Installation

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

```
emu5xo c:\C5xh11\sample -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 see a display and the lines of code show ADD instructions, your emulator board may not be installed snugly. Check your board to see whether it is correctly installed, and reenter the command above.
- ☐ If you see a display and the lines of code say *Invalid address* or the fields in the MEMORY window are shown in red, the debugger may not be able to find the `emuinit.cmd` file. Check for the file in the directories specified by the `D_SRC` environment variable or ensure that the file is in the current directory. Reenter the command above.
- ☐ 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.

## Installation error messages

While invoking the debugger, you may see the following message:

CANNOT INITIALIZE TARGET SYSTEM ! !  
- Check I/O configuration  
- Check cabling and target power

One of several of the following conditions may be the cause; check:

- ☐ Is the target power on?
- ☐ Is the emulator board installed snugly?
- ☐ Is the device installed snugly?
- ☐ Is the cable connecting your emulator and target system loose?
- ☐ Is your target board getting the correct voltage?
- ☐ Is your emulator scan path interrupted? One or more devices on the emulator scan path may have been removed. Check the connections; either they are not connected, or they are connected improperly.
- ☐ Did you use the `-n` option? Or was it used with an incorrect device name? You must supply a valid device name with the `-n` option.
- ☐ After you powered up the target board, did you execute the `emurst.exe` command? Check your `startup.cmd` file or `config.sys` file. This command must be executed *after* you powered up the target board.
- ☐ Did you use the `-p` option? Is your port address set correctly?
  - Check to be sure the `-p` option of the `D_OPTIONS` environment variable matches the I/O address defined by your switch settings. For information about the switch settings, see the *XDS51x Emulator Installation Guide*.
  - Check to see whether you have a conflict in address space with another bus setting. If you have a conflict, change the switches on your board to one of the alternate settings. Modify the `-p` option of the `D_OPTIONS` environment variable to reflect the change in your switch settings.
- ☐ Is the `board.dat` file in the current directory or in a directory specified by `D_DIR`?

**Note:**

If the debugger suddenly quits or behaves erratically during the debugging process, the board.dat file may contain incorrect information in the correct format. See Section 2.6, *Step 5: Describing Your Target System to the Debugger*, page 2-10 for more information.

After you have checked all of the above, repeat the verification instructions on page 2-11.

# Installing the Debugger on a SPARCstation

This chapter helps you to install the TMS320C5x C source debugger on a SPARCstation running OpenWindows™ under SunOS™ version 4.1.1 (or higher). After completing the installation, see the *TMS320C5x C Source Debugger User's Guide* for instructions on using the debugger.

To install the XDS510WS emulator controller, see the *XDS51x Emulator Installation Guide*.

Topic	Page
3.1 System Requirements .....	3-2
3.2 Step 1: Installing the XDS510WS Emulator Controller .....	3-4
3.3 Step 2: Installing the Debugger Software .....	3-4
3.4 Step 3: Ensuring That the Emulator Supports the Debugger .....	3-6
3.5 Step 4: Describing Your Target System to the Debugger .....	3-7
3.6 Step 5: Setting Up the Debugger Environment .....	3-8
3.7 Step 6: Verifying the Installation .....	3-11
3.8 Using the Debugger With the X Window System .....	3-12

### 3.1 System Requirements

To install and use the 'C5x emulator and C source debugger, you need the items listed in the following hardware and software checklists.

#### Hardware checklist

<input type="checkbox"/>	<b>System</b>	SPARCstation or 100% compatible system
<input type="checkbox"/>	<b>Display</b>	Monochrome or color (color recommended)
<input type="checkbox"/>	<b>CD-ROM drive</b>	CD-ROM drive
<input type="checkbox"/>	<b>XDS510 emulator controller</b>	XDS510WS emulator controller

#### Software checklist

<input type="checkbox"/>	<b>Operating system</b>	SunOS version 4.1.3 (or higher) or SunOS version 5.x (also known as Solaris 2.x) using an X Window System-based window manager, such as OpenWindows version 3.0 (or higher).
<input type="checkbox"/>	<b>Root privileges</b>	If you are running SunOS 4.1.x, 5.0, or 5.1, you <i>must</i> have root privileges to mount and unmount the CD-ROM. If you don't, get help from your system administrator.
<input type="checkbox"/>	<b>Software tools</b>	TMS320C1x/C2x/C2xx/C5x DSP assembler and linker Optional: TMS320C2x/C5x DSP C compiler
<input type="checkbox"/>	<b>Required files included with debugger package</b>	<i>emu5x</i> is the debugger executable file.
<input type="checkbox"/>		<i>c5x510ws.out</i> is the executable portion of the debugger that runs on the emulator.
<input type="checkbox"/>		<i>emurst</i> resets the emulator and downloads <i>c5x510ws.out</i> to the emulator.
<input type="checkbox"/>		<i>board.dat</i> describes your target board to your debugger in terms of what devices are in the emulation scan path. The <i>board.dat</i> file included in the debugger package is for a target board with one TMS320C5x named <i>cpu_a</i> .



- |                          |  |  |
|--------------------------|--|--|
| <input type="checkbox"/> | <b>Optional files included with debugger package</b> | <p><i>emuinit.cmd</i> is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C5x memory map. When you first start using the debugger, this memory map should be sufficient for your needs. Later, you may want to define your own memory map. For information about setting up your own memory map, see the <i>Defining a Memory Map</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i>.</p>                                  |
| <input type="checkbox"/> |  | <p><i>composer</i> is the utility that translates the board.cfg file to a binary, conditioned format. For the emulator to initialize properly, you must create a new board.dat file with this release of the composer.</p>   |
| <input type="checkbox"/> |  | <p><i>board.cfg</i> is a text file used to describe your target system 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 TMS320C5x named <i>cpu_a</i>.</p>   |
| <input type="checkbox"/> |  | <p><i>init.clr</i> 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.</p>  |
| <input type="checkbox"/> |  | <p><i>init.25</i>, <i>init.43</i>, and <i>init.50</i> have been provided for basic 80×25, 80×43, and 80×50 screen sizes, respectively. The <i>init.clr</i> file brings up the debugger in 80×25 mode. To bring the debugger up in another mode, copy one of the <i>init.xx</i> files to the <i>init.clr</i> file.</p>  |
| <input type="checkbox"/> |  | <p>The default configuration is for color monitors; an additional file, <i>mono.clr</i>, can be used with monochrome monitors. When you first start to use the debugger, the default screen configuration should be sufficient for your needs. Later, you may want to define your own custom configuration.</p> <p>For information about these files and about setting up your own screen configuration, see the <i>Customizing the Debugger Display</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i>.</p> |

### 3.2 Step 1: Installing the XDS510WS Emulator Controller

Before installing the 'C5x debugger software, you must install an XDS510WS emulator controller.

Follow the instructions in the *XDS51x Emulator Installation Guide* to install the XDS510WS emulator controller.

### 3.3 Step 2: 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 CD-ROM. To install the emulator software, you must mount the CD-ROM, copy the files, and unmount the CD-ROM.

**Note:**

If you are running SunOS 4.1.x, 5.0, or 5.1, you *must* have root privileges to mount or unmount the CD-ROM. If you don't, get help from your system administrator.

#### ***Mounting the CD-ROM***

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

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

```
mount -rt hsfs /dev/sr0 /cdrom
exit
cd /cdrom/sparc
```

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

```
mount -rF hsfs /dev/sr0 /cdrom
exit
cd /cdrom/cdrom0/sparc
```

- ☐ 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/sparc
```

- 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/sparc
```

### ***Copying the files***

After you mount 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 *c5xhl* on your hard disk. To create this directory, enter:

```
mkdir /your_pathname/c5xhl
```

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

```
cp -r * /your_pathname/c5xhl
```

### ***Unmounting the CD-ROM***

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

- If you have a 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
```

### 3.4 Step 3: Ensuring That 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 debugger 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.

The -p option *number* represents the device driver number you defined in the EMULATOR configuration file (see the *XDS51x Emulator Installation Guide*), and *pathname-filename* is the location and name of the c5x510ws.out file. Specify the number in hexadecimal.

If the default, *rsd4a*, is the device driver for the emulator, you can omit the -p option.

---

**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. (This number is *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.

### 3.5 Step 4: 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 'C5x and no other devices, you can use the *board.dat* file that comes with the 'C5x emulator kit. This file describes to the debugger the single 'C5x in the scan path and gives the 'C5x 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 Section 3.6.
- ☐ If you plan to use a different target system, you must follow these steps:
  - 1) Create the board configuration file.
  - 2) Translate the board configuration file to binary so that the debugger can read it.
  - 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 *TMS320C5x C Source Debugger User's Guide*.

### 3.6 Step 5: Setting Up the Debugger Environment

To ensure that your debugger works correctly, you must:

- ☐ Modify the path shell variable to identify the c5xhll directory
- ☐ Set up the environment variables that you want to use
- ☐ Reinitialize your shell

#### **Modifying the path statement**

You must include the debugger directory in your shell path. To do this, you need to modify your shell configuration file in your home directory (for example, the .cshrc file for a C shell). Include the pathname to your c5xhll directory in your path. The following statement is an example of what a typical path-variable definition looks like:

```
set path = ( . /bin /usr/ucb /usr/contrib/bin /usr/bin \  
/usr/openwin/bin)
```

The following is an example of a modified path variable. The part of the path that is boldface is an example of a pathname that identifies the c5xhll directory:

```
set path = ( . /bin /usr/ucb /usr/contrib/bin /usr/bin \  
/usr/openwin/bin /user/fred/c5xhll)
```

#### **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. You can set up the environment variables in your shell configuration file. The debugger uses four environment variables, named D\_DIR, D\_SRC, D\_OPTIONS, and DISPLAY:

- ☐ Set up the D\_DIR environment variable to identify the c5xhll directory. The general format for doing this is:

```
setenv D_DIR "/user/fred/c5xhll"
```

(Be sure to enclose the directory name within quotes.)

This directory contains auxiliary files (such as emuinit.cmd) 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 access from the debugger. The general format for doing this is:

```
setenv D_SRC "pathname1; pathname2..."
```

(Be sure to enclose the path names within one set of quotes.)

For example, if your programs were in a directory named /user/fred/c5xcode, the D\_SRC setup would be:

```
setenv D_SRC "/user/fred/c5xcode"
```

- You can use several options when you invoke the debugger. If you use the same options repeatedly, it's convenient to specify them with D\_OPTIONS. The general format for doing this is:

**setenv D\_OPTIONS "[object filename] [debugger options]"**

(Be sure to enclose the options and filenames within one set of quotes.)

This tells the debugger to load the specified object file and use the specified options each time you invoke the debugger. These are the options that you can identify with D\_OPTIONS:

Option	Brief Description
-b[b]	Select the screen size
-d <i>machine name</i>	Display debugger on different machine
-f <i>filename</i>	Identify a new board configuration file
-i <i>pathname</i>	Identify additional directories
-n <i>processor name</i>	Identify processor for debugging
-p <i>port address</i>	Identify the port address
-profile	Enter profiling environment
-s	Load the symbol table only
-t <i>filename</i>	Identify a new initialization file
-v	Load without the symbol table

Note that you can override D\_OPTIONS by invoking the debugger or emurst with the -x option.

For more information about options, see the invocation section in the *Overview of a Code Development and Debugging System* chapter in the *TMS320C5x 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 from the one the debugger is running on. The general format for doing this is:

**setenv DISPLAY "machine name"**

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

**setenv DISPLAY barney:0** 


You can also display the debugger on a different machine by using the -d option when invoking the debugger.

**emu5x -d barney:0** 

For more information about using the debugger under the X Window system, see Section 3.8, *Using the Debugger With the X Window System*, page 3-12.

### ***Invoking the new or modified .cshrc file***

When you modify your shell configuration file, you must ensure that the changes are made to your current session. For example, if you are using a C shell, use this command to reread the .cshrc file:

**source ~/.cshrc** 

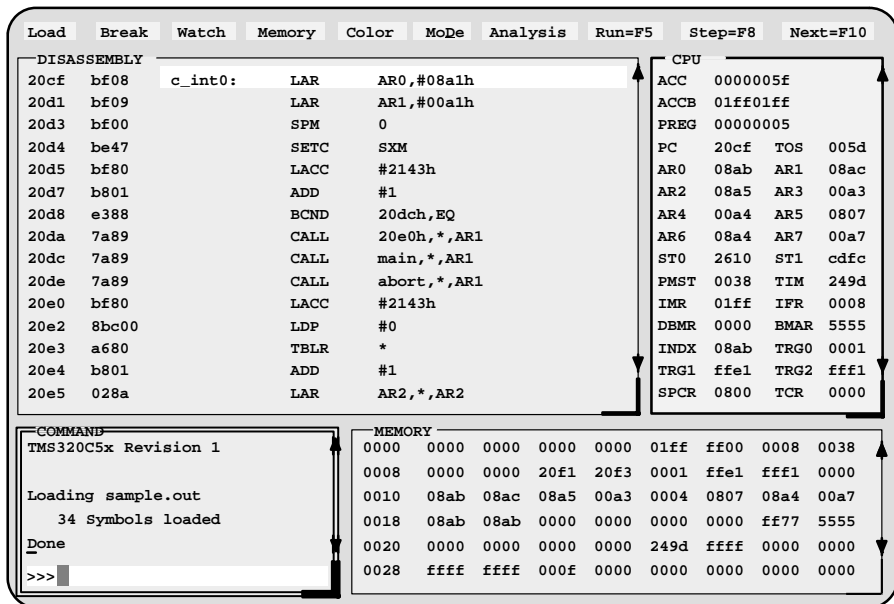


### 3.7 Step 6: Verifying the Installation

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

```
emu5x sample -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.
- ☐ If you continue to experience problems, see Appendix A, *Troubleshooting*.

### 3.8 Using the Debugger With the X Window System

If you're using the X Window System to run the 'C5x 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, have these special symbols on separate keys. Other keyboards, such as the Sun Type 4 keyboard, do not have the special keys, but the functions are available.

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.

Debugger Key Needed	Keysym for That Function
(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 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 map a Sun Type 4 keyboard to the keys listed above:

```
key code      keysym
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 'C5x debugger fonts to Courier, add the following line to the `.Xdefaults` file:

```
emu5x*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, , you can use a monochrome monitor. The following table shows the color mappings for monochrome screens:

<b>Color</b>	<b>Appearance on Monochrome Screen</b>
black	black
blue	black
green	white
cyan	white
red	black
magenta	black
yellow	white
white	white

# Installing the Debugger on an HP Workstation

This chapter helps you to install the TMS320C5x C source debugger on an HP 9000 Series 700™ PA-RISC™ computer with HP-UX™ 9.0x. After completing the installation, see the *TMS320C5x C Source Debugger User's Guide* for instructions on using the debugger.

To install the XDS510WS emulator controller, see the *XDS51x Emulator Installation Guide*.

Topic	Page
4.1 System Requirements .....	4-2
4.2 Step 1: Installing the XDS510WS Emulator Controller .....	4-4
4.3 Step 2: Installing the Debugger Software .....	4-4
4.4 Step 3: Ensuring That the Emulator Supports the Debugger .....	4-6
4.5 Step 4: Describing Your Target System to the Debugger .....	4-7
4.6 Step 5: Setting Up the Debugger Environment .....	4-8
4.7 Step 6: Verifying the Installation .....	4-11
4.8 Using the Debugger With the X Window System .....	4-12

## 4.1 System Requirements

To install and use the 'C5x emulator and C source debugger, you need the items listed in the following hardware and software checklists.

### Hardware checklist

- |                          |                            |   |
|--------------------------|----------------------------|---|
| <input type="checkbox"/> | <b>System</b>              | HP 9000 Series 700 PA-RISC computer     |
| <input type="checkbox"/> | <b>Display</b>             | Monochrome or color (color recommended) |
| <input type="checkbox"/> | <b>CD-ROM drive</b>        | CD-ROM drive                            |
| <input type="checkbox"/> | <b>XDS51x emulator kit</b> | XDS510WS emulator                       |

### Software checklist

- |                          |  |   |
|--------------------------|--|---|
| <input type="checkbox"/> | <b>Operating system</b>                              | HP-UX version 9.0x or higher  |
| <input type="checkbox"/> | <b>Root privileges</b>                               | You <i>must</i> have root privileges to mount and unmount the CD-ROM. If you don't, get help from your system administrator.  |
| <input type="checkbox"/> | <b>Software tools</b>                                | TMS320C1x/C2x/C2xx/C5x DSP assembler and linker<br>Optional: TMS320C2x/C2xx/C5x C compiler  |
| <input type="checkbox"/> | <b>Required files included with debugger package</b> | <i>emu5x</i> is the debugger executable file.   |
| <input type="checkbox"/> |  | <i>c5x510ws.out</i> is the executable portion of the debugger that runs on the emulator.  |
| <input type="checkbox"/> |  | <i>emurst</i> resets the emulator and downloads <i>c5x510ws.out</i> to the emulator.  |
| <input type="checkbox"/> |  | <i>board.dat</i> describes your target board to your debugger in terms of what devices are on the emulation scan path. The <i>board.dat</i> file included in the debugger package is for a target board with one TMS320C5x named <i>cpu_a</i> . |

- |                          |  |   |
|--------------------------|--|---|
| <input type="checkbox"/> | <b>Optional files included with debugger package</b> | <i>emuinit.cmd</i> is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C5x memory map. When you first start using the debugger, this memory map should be sufficient for your needs. Later, you may want to define your own memory map. For information about setting up your own memory map, see the <i>Defining a Memory Map</i> chapter in the <i>TMS320C5x C Source Debugger User's Guide</i> . |
| <input type="checkbox"/> |  | <i>composer</i> is the utility that translates the board.cfg file to a binary, conditioned format. For the emulator to initialize properly, you must create a new board.dat file with this release of the composer.   |
| <input type="checkbox"/> |  | <i>board.cfg</i> is a text file used to describe your target system 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 TMS320C5x named <i>cpu_a</i> .  |
| <input type="checkbox"/> |  | <i>init.clr</i> 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.  |
| <input type="checkbox"/> |  | <i>init.25</i> , <i>init.43</i> , and <i>init.50</i> have been provided for basic 80×25, 80×43, and 80×50 screen sizes, respectively. The <i>init.clr</i> file brings up the debugger in 80×25 mode. To bring the debugger up in another mode, copy one of the <i>init.xx</i> files to the <i>init.clr</i> file.  |
| <input type="checkbox"/> |  | The default configuration is for color monitors; an additional file, <i>mono.clr</i> , can be used for monochrome monitors. When you first start to use the debugger, the default screen configuration should be sufficient for your needs. Later, you may want to define your own custom configuration.  |
- For information about these files and about setting up your own screen configuration, see the *Customizing the Debugger Display* chapter in the *TMS320C5x C Source Debugger User's Guide*.

## 4.2 Step 1: Installing the XDS510WS Emulator Controller

Before installing the 'C5x debugger software, you must install an XDS510WS emulator controller.

Follow the instructions in the *XDS51x Emulator Installation Guide* to install the XDS510WS emulator controller.

## 4.3 Step 2: 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.

### Note:

You *must* have root privileges to mount or unmount the CD-ROM. If you don't, get help from your system administrator.

### Mounting the CD-ROM

As root, you can mount the CD-ROM using the UNIX mount command or the SAM (system administration manager):

- ☐ To use the UNIX mount command, enter:

```
mount -rt cdfs /dev/dsk/your_cdrom_device /cdrom
exit
```

Make the hp directory on the CD-ROM the current directory. For example, if the CD-ROM is mounted at /cdrom, enter:

```
cd /cdrom/hp
```

- ☐ To use SAM to mount the CD-ROM, see *System Administration Tasks*, the HP documentation about SAM, for instructions.

### ***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 *c5xhl1* on your hard disk. To create this directory, enter:

```
mkdir /your_pathname/c5xhl1
```

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

```
cp -r * /your_pathname/c5xhl1
```

### ***Unmounting the CD-ROM***

You must unmount the CD-ROM after copying the files. As root, enter:

```
cd  
umount /cdrom  
exit
```



#### 4.4 Step 3: Ensuring That 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 debugger 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.

The -p option *number* represents the SCSI controller number concatenated to the SCSI ID number of the XDS510WS (for example, the default would be 2014), and *pathname-filename* is the location and name of the c5x510ws.out file.

If the default, 4, is the SCSI ID for the emulator and if the emulator is on the default SCSI bus, you can omit the -p option.

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

## 4.5 Step 4: 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 'C5x and no other devices, you can use the *board.dat* file that comes with the 'C5x emulator kit. This file describes to the debugger the single 'C5x in the scan path and gives the 'C5x 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 Section 4.6.
- ☐ If you plan to use a different target system, you must follow these steps:
  - 1) Create the board configuration file.
  - 2) Translate the board configuration file to binary so that the debugger can read it.
  - 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 *TMS320C5x C Source Debugger User's Guide*.

## 4.6 Step 5: Setting Up the Debugger Environment

To ensure that your debugger works correctly, you must:

- ☐ Modify the path shell variable to identify the c5xhll directory
- ☐ Set up the environment variables that you want to use
- ☐ Reinitialize your shell

### **Modifying the path statement**

You must include the debugger directory in your shell path. To do this, you need to modify your shell configuration file in your home directory (for example, the .cshrc file for a C shell). Include the pathname to your c5xhll directory in your path. The following statement is an example of what a typical path-variable definition looks like:

```
set path = ( . /bin /usr/ucb /usr/contrib/bin /usr/bin \
/usr/openwin/bin)
```

The following is an example of a modified path variable. The part of the path that is boldface is an example of a pathname that identifies the c5xhll directory:

```
set path ( . /bin /usr/ucb /usr/contrib/bin /usr/bin \
/usr/openwin/bin /user/fred/c5xhll)
```

### **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. You can set up the environment variables in your shell configuration file. The debugger uses four environment variables, named D\_DIR, D\_SRC, D\_OPTIONS, and DISPLAY:

- ☐ Set up the D\_DIR environment variable to identify the c5xhll directory. The general format for doing this is:

```
setenv D_DIR "/user/fred/c5xhll"
```

(Be sure to enclose the directory name within quotes.)

This directory contains auxiliary files (such as emuinit.cmd) 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 "pathname1;pathname2;..."
```

(Be sure to enclose the path names within one set of quotes.)

For example, if your programs were in a directory named /user/fred/c5xcode, the D\_SRC setup would be:

```
setenv D_SRC "/user/fred/c5xcode"
```

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

**setenv D\_OPTIONS "[object filename] [debugger options]"**

(Be sure to enclose the options and filenames within one set of quotes.)

This tells the debugger to load the specified object file and use the specified options each time you invoke the debugger. These are the options that you can identify with `D_OPTIONS`:

Option	Brief Description
<code>-b[b]</code>	Select the screen size
<code>-c</code>	Clear the .bss section
<code>-f filename</code>	Identify a new board configuration file
<code>-i pathname</code>	Identify additional directories
<code>-me</code>	Select little-endian format
<code>-min</code>	Select the minimal debugging mode
<code>-n processor name</code>	Identify processor for debugging
<code>-p port address</code>	Identify the port address
<code>-s</code>	Load the symbol table only
<code>-t filename</code>	Identify a new initialization file
<code>-v</code>	Load without the symbol table

Note that you can override `D_OPTIONS` by invoking the debugger or `emurst` 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 *TMS320C5x 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 from the one the debugger is running on. The general format for doing this is:

**setenv DISPLAY "machine name"**

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

```
setenv DISPLAY barney:0
```

You can also display the debugger on a different machine by using the -d option when invoking the debugger.

```
emu5x -d barney:0
```

For more information about using the debugger under the X Window system, see Section 4.8, *Using the Debugger With the X Window System*.

### ***Invoking the new or modified .cshrc file***

When you modify your shell configuration file, you must ensure that the changes are made to your current session. For example, if you are using a C shell, use this command to reread the .cshrc file:

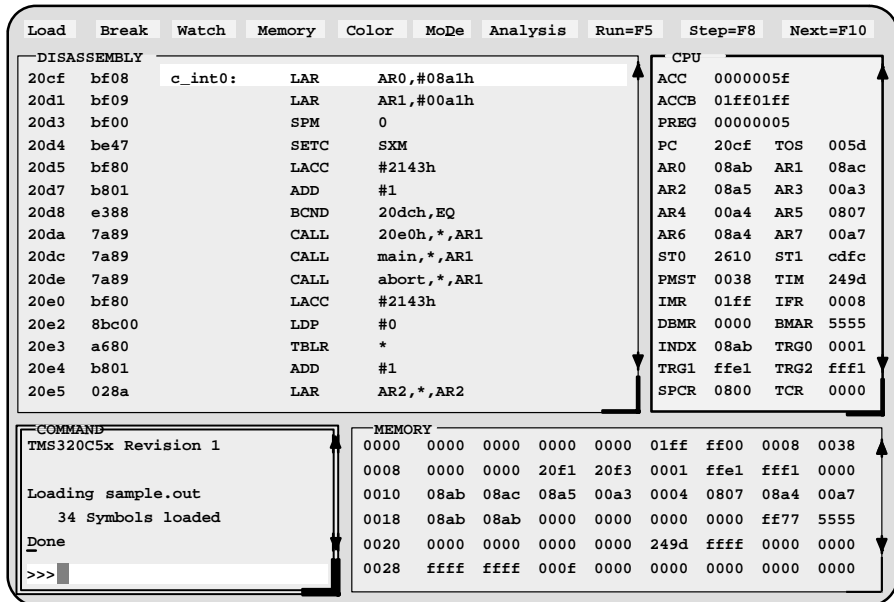
```
source ~/.cshrc
```

## 4.7 Step 6: Verifying the Installation

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

```
emu5x sample -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.
- ☐ If you continue to experience problems, see Appendix A, *Troubleshooting*.

## 4.8 Using the Debugger With the X Window System

If you're using the X Window System to run the 'C5x 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, have these special symbols on separate keys. Other keyboards, such as the Sun Type 4 keyboard, do not have the special keys, but the functions are available.

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.

Debugger Key Needed	Keysym for That Function
(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 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 map a Sun Type 4 keyboard to the keys listed above:

```
      key code      keysym
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 'C5x debugger fonts to Courier, add the following line to the `.Xdefaults` file:

```
emu5x*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, , you can use a monochrome monitor. The following table shows the color mappings for monochrome screens:

<b>Color</b>	<b>Appearance on Monochrome Screen</b>
black	black
blue	black
green	white
cyan	white
red	black
magenta	black
yellow	white
white	white



# Enabling Extended Addressing

---

---

---

The TMS320C5x is limited to 64K bytes of address space in program, data, and I/O space. Some applications require access to memory beyond the 64K limit for program and data space. By adding memory and additional logic to your target system, you can extend the memory of your system. The emulator version of the debugger includes an extended addressing feature that enables the 'C5x to access this extended memory when you debug your system.

If you use extended addressing, the debugger supports access to instructions and data stored in extended memory.

This chapter defines extended addressing and describes what you need to do to your target system, your source files, and your debugger to enable extended addressing for the TMS320C5x.

Topic	Page
5.1 Understanding the Use of Extended Addressing .....	5-2
5.2 Building an Extended Memory System .....	5-4

## 5.1 Understanding the Use of Extended Addressing

With the extended addressing feature, you have the ability to add additional memory to your target system for use by the 'C5x.

### ***About extended addressing***

Once you modify your target system to include extended memory, you can use the debugger to access the extended memory. To determine which memory location you want to access, the debugger must have a unique address.

In an extended memory system, you logically split the 16-bit 'C5x address space into two pieces. The low-ordered addresses are common or *unmapped* memory. The high-ordered addresses are extended or *mapped* memory. The address that defines the boundary between unmapped and mapped memory, the *mapped start address*, is defined based on the external registers and memory that you add to your target system.

You can add and define multiple banks of physical memory that overlay each other in a single, mapped address range. The use of extended pages of memory extends the native 16-bit address space. Because the 16-bit address space is extended, the debugger uses addresses that are 32 bits wide. The 32-bit address is composed of the following:

- ☐ The 16 most significant bits (MSBs) represent the extended page number. In your source code, you store this number in one of the following registers:
  - The program mapper register (PMR) stores the extended page number for extended program-memory pages. The value that you use in the PMR is the same extended-page number that you use in the linker command file.
  - The data mapper register (DMR) stores the extended page number for extended data-memory pages. The value that you use in the DMR is the same extended-page number that you use in the linker command file.

You create the PMR or DMR when you build your extended memory system.

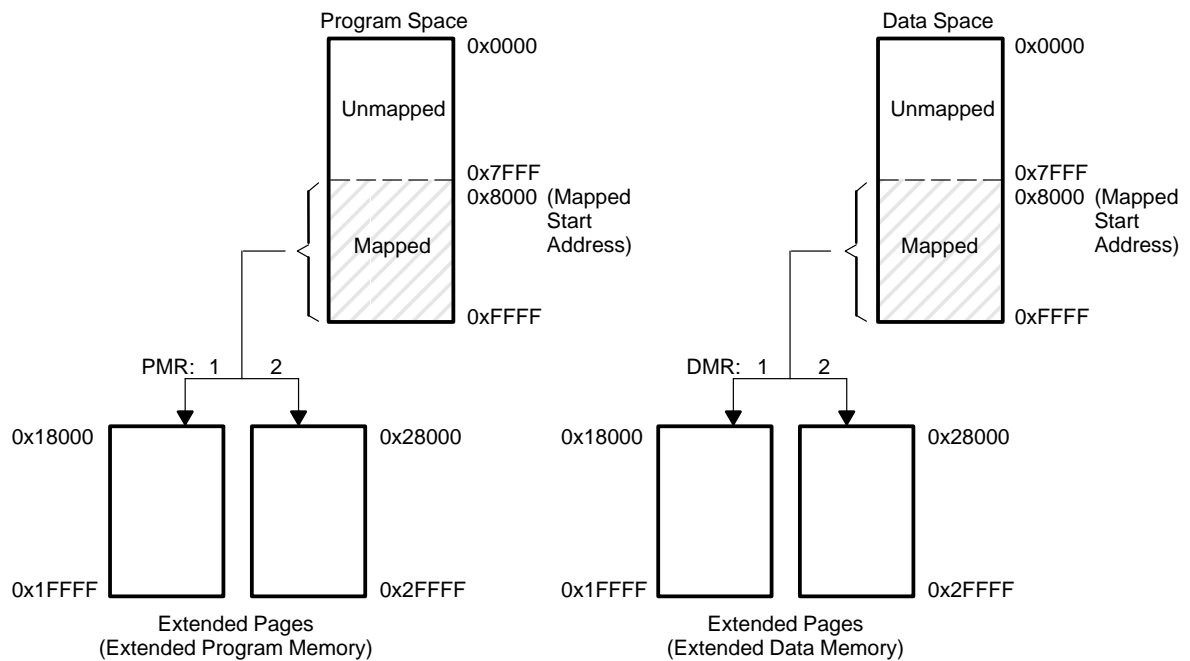
- ☐ The 16 least significant bits (LSBs) represent the native 16-bit address for the symbol.

### Sample extended memory system

Figure 5–1 illustrates a sample extended memory system. In this example, the program and data space are allocated in the same manner:

- ❑ The unmapped memory region is from 0x0000 to 0x7FFF.
- ❑ The mapped memory region is from 0x8000 to 0xFFFF.
- ❑ Two extended pages extend the mapped address space.
- ❑ The mapper register (PMR or DMR) qualifies the mapped address, creating a unique address for each location in each extended page.

Figure 5–1. Sample Extended Memory System



## 5.2 Building an Extended Memory System

To use the emulator with extended addressing, you must do the following:

Step	Action	See Page
1	Add memory and its associated logic to your target system.	5-4
2	Create registers on your target system.	5-5
3	Update or create source files that take advantage of extended memory.	5-5
4	Update or create a linker command file that takes advantage of extended memory.	5-6
5	Describe and enable extended addressing in the debugger.	7-4

### ***Adding memory and logic***

When you build your target system, add memory and the logic for the memory as you normally would. The following rules apply when adding extended memory:

- ☐ The amount of memory you can add is limited by the number of high-order addresses (up to 16 bits) and the size of the mapped memory region (some amount you determine, but less than 64K).
- ☐ Memory is divided into extended pages.
- ☐ You must supply memory decode logic that expands the 16-bit address into a 32-bit address.
- ☐ The decode logic determines the mapped start address.
- ☐ You assign a number to each extended page. You load this number into a register, which becomes the high-order portion of a 32-bit address used to access an extended address (see the *Creating registers* subsection, page 5-5, for information about these registers).
- ☐ The maximum size for an extended page is the size of the mapped memory region of the 'C5x address space.  
  
For example, in Figure 5–1 the maximum size of an extended page can be 0x8000 bytes since that is the size of the mapped address space.
- ☐ You cannot define all of your 'C5x address space as mapped, and you can only have one mapped area per program or data memory space.
- ☐ Memory addresses cannot be more than 32 bits. The high-order portion is up to 16 bits, and the low-order portion is 16 bits.

### ***Creating registers***

When you build your target system, add one or two external registers. Add the registers as you would normally add a register. You add one or both of the following registers:

- ☐ Program mapper register (PMR). The PMR is used to extend program memory.
- ☐ The data mapper register (DMR). The DMR is used to extend data memory.

The following rules apply when building the registers:

- ☐ They must be in the mapped memory region of the 'C5x address space.
- ☐ They must be mapped into an unmapped data space or an I/O space.
- ☐ They must be read and write enabled.
- ☐ They can be up to 16 bits wide.
- ☐ The size of a register determines the number of extended pages since each extended page must have a number associated with it. For example, if your register is 8 bits wide, you can have 256 extended pages since an 8-bit register can contain addresses 0x00 to 0xFF.

### ***Updating or creating source files***

To read and write to extended memory in your source code, you must do the following:

- 1) Load the extended page number for the memory you want to access into a mapper register (PMR or DMR). This is the 16 MSBs of the extended address.
- 2) Use the normal 16-bit I/O instructions with the 16 LSBs of the extended address. This identifies the memory that you want to access in the extended page.

The linker and assembler automatically concatenate the extended page number and the 16-bit address together to obtain the 32-bit address needed to access extended memory.

### **Updating or creating a linker command file**

You must define your extended memory system to the linker by using the linker MEMORY directive in a linker command file. This section describes how to define extended memory addresses in a linker command file. For more details about linker command files, see the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide*.

The only difference between specifying a normal address and an extended address is to add an extended page number to a 16-bit address in the MEMORY directive. All other conventions for a linker command file are the same.

Example 5–1 shows a sample linker command file for use with an extended memory system.

- ☐ Base defines the unmapped region of the 'C5x program space. The region starts at 0x1000 and has a length of 0x2000.
- ☐ Ext0 defines the mapped memory region of the 'C5x program space. The region starts at 0xC000 and has a length of 0x4000. The PMR must be set to 0 to access this memory.
- ☐ Ext1, Ext2, Ext3, and Ext4 define extended pages for program space. When accessed, these pages overlay the mapped memory region defined by Ext0. Each extended page starts at 0xC000 and has a length of 0x4000, although they do not have to take up the entire mapped memory region.

The number preceding each starting address is the extended page number. You load this number into a mapper register to identify the section of extended memory you want to access. This entire number is used as the extended memory address.

Ext1 has a starting address of 0x0001 C000, Ext2 has a starting address of 0x0002 C000, and so on. The number you assign to the extended page is arbitrary, but it cannot be larger than the mapper register you load it into.

- ☐ No extended memory is defined for the 'C5x data space.
- ☐ All other aspects of the linker command file are the same as defined in the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide*.

*Example 5–1. Sample Linker Command File*

```

/* Input filenames      */
callall.obj
ext0.obj
ext1.obj
ext4.obj

/* Options              */
-o callall.out
-m callall.map
-c
-l rts50.lib
-stack 30
MEMORY
{
/* Program memory      */
PAGE 0 :
    Base: origin = 1000h , length = 2000h
    Ext0: origin = 0c000h , length = 4000h
    Ext1: origin = 1c000h , length = 4000h
    Ext2: origin = 2c000h , length = 4000h
    Ext3: origin = 3c000h , length = 4000h
    Ext4: origin = 4c000h , length = 4000h
/* Data memory         */
PAGE 1 :
    Scratch_RAM : origin = 3000h , length = 1000h
}
/* SECTIONS directive */
SECTIONS
{
    .text :          > Base          PAGE = 0
    .text0 :         { ext0.obj(.text) } > Ext0 PAGE 0
    .text1 :         { ext1.obj(.text) } > Ext1 PAGE 0
    .text4 :         { ext4.obj(.text) } > Ext4 PAGE 0
}

```

Once you set up your extended address system, you are ready to use it. To use it with the 'C5x debugger, you must describe your memory system to the debugger and enable extended addressing. You accomplish these tasks by using the EXT\_ADDR\_DEF and EXT\_ADDR debugger commands.

Additionally, some of the debugger displays are altered when using extended addressing. For information about using the debugger with extended addressing, see Chapter 7, *Using the Debugger With Extended Addressing*.

# Release Notes

---

---

---

---

Release 7.40 of the TMS320C5x debugger contains general enhancements as well as enhancements specific to the 'C5x emulator. The following sections list these enhancements.

### ***COFF version 2***

This release supports an expanded object file format called COFF2. The debugger is capable of handling COFF object files developed with assembly language tools using the COFF0, COFF1, or COFF2 formats.

### ***Multiple MEMORY windows***

You can now open more than four MEMORY windows. The MEM command has a new optional *window name* parameter. When you open an additional MEMORY window, the debugger appends the *window name* to the MEMORY window label. You can create as many MEMORY windows as you need. The basic syntax for the MEM command is:

**mem** *expression* [, *display format*] [, *window name*]

You can use the MEM command to display a different memory range in a window. The *window name* parameter is optional if you are displaying a different memory range in the default MEMORY window. Use the *window name* parameter when you want to display a new memory range in one of the additional MEMORY windows.



## Multiple WATCH windows

You can now access multiple WATCH windows. Use the *window name* parameter as described for each WATCH window command.

- ❑ The WA command has a new optional *window name* parameter. When you open an additional WATCH window, the debugger appends the *window name* to the WATCH window label. You can create as many WATCH windows as you need. The basic syntax for the WA command is:

**wa** *expression* [, [*label*] [, [*display format*] [, *window name*] ] ]

If you omit the *window name* parameter, the debugger displays the expression in the default WATCH window (labeled WATCH).

- ❑ The WD command deletes a specific item from the WATCH window. The WD command's *index number* parameter must correspond to one of the watch indexes listed in the WATCH window. The optional *window name* parameter is used to specify a particular WATCH window. The basic syntax for the WD command is:

**wd** *index number* [, *window name*]

- ❑ The WR command deletes all items from a WATCH window and closes the window.

- To close the default WATCH window, enter:

**wr**

- To close one of the additional WATCH windows, use this syntax:

**wr** *windowname*

- To close all WATCH windows, enter:

**wr \***

## New debugger commands

The debugger supports the following new commands on all platforms.

<b>cd, chdir</b>	<i>Change Directory</i>
<b>Syntax</b>	<b>cd</b> [ <i>directory name</i> ] <b>chdir</b> [ <i>directory name</i> ]
<b>Menu selection</b>	none
<b>Environments</b>	<input checked="" type="checkbox"/> basic debugger <input type="checkbox"/> PDM <input checked="" type="checkbox"/> profiling
<b>Description</b>	<p>The CD or CHDIR command changes the current working directory from within the debugger. You can use relative pathnames as part of the <i>directory name</i>. If you don't use a <i>directory name</i>, the CD command displays the name of the current directory. Note that this command can affect any other command whose parameter is a filename, such as the FILE, LOAD, and TAKE commands, when used with the USE command. You can also use the CD command to change the current drive. For example:</p> <pre>cd c: cd d:\csource cd c:\c5xh11</pre>
<b>dir</b>	<i>List Directory Contents</i>
<b>Syntax</b>	<b>dir</b> [ <i>directory name</i> ]
<b>Menu selection</b>	none
<b>Environments</b>	<input checked="" type="checkbox"/> basic debugger <input type="checkbox"/> PDM <input checked="" type="checkbox"/> profiling
<b>Description</b>	<p>The DIR command displays a directory listing in the display area of the COMMAND window. If you use the optional <i>directory name</i> parameter, the debugger displays a list of the specified directory's contents. If you don't use the a <i>directory name</i>, the debugger lists the contents of the current directory.</p> <p>You can list only files that match a specific format within a directory by using the asterisk (*) wildcard character. If the <i>directory name</i> ends in a partial file-name with an asterisk, the debugger lists only the files which match the wildcard string. For example, to list every file in the home directory that has a .cmd extension, you would enter:</p> <pre>DIR /home/* .cmd</pre>

### ***Extended addressing***

The 'C5x debugger supports mapped extended memory. You can define a segmented memory architecture that goes beyond the 64K byte limit imposed by the 16-bit address bus. You can use extended addressing with program address space or data address space.

See Chapter 5, *Enabling Extended Addressing*, for information on setting up and enabling extended addressing. See Chapter 7, *Using Extended Addressing With the Debugger*, for more information on implementing extended addressing with the 'C5x debugger.

# Using the Debugger With Extended Addressing



The TMS320C5x is limited to 64K bytes of address space in program, data, and I/O space. Some applications require access to memory beyond the 64K limit for program and data space. By adding memory and additional logic to your target system, you can *extend* the memory of your system. The emulator version of the debugger includes an extended addressing feature that enables the 'C5x to access this extended memory when you debug your system.

This chapter defines extended addressing and describes what you need to do in your debugger to enable extended addressing for the TMS320C5x.

Topic	Page
7.1 Understanding the Use of Extended Addressing .....	7-2
7.2 Setting Up Extended Addressing .....	7-4
7.3 Debugging With Extended Addressing .....	7-6

## 7.1 Understanding the Use of Extended Addressing

With the extended addressing feature, you have the ability to add additional memory to your target system for use by the 'C5x.

### ***About extended addressing***

Once you modify your target system to include extended memory, you can use the debugger to access the extended memory. To determine which memory location you want to access, the debugger must have a unique address.

In an extended memory system, you logically split the 16-bit 'C5x address space into two pieces. The low-ordered addresses are common or *unmapped* memory. The high-ordered addresses are extended or *mapped* memory. The address that defines the boundary between unmapped and mapped memory, the *mapped start address*, is defined based on the external registers and memory that you add to your target system.

You can add and define multiple banks of physical memory that overlay each other in a single, mapped address range. The use of extended pages of memory extends the native 16-bit address space. Because the 16-bit address space is extended, the debugger uses addresses that are 32 bits wide. The 32-bit address is composed of the following:

- ☐ The 16 most significant bits (MSBs) represent the extended page number. In your source code, you store this number in one of the following registers:
  - The program mapper register (PMR) stores the extended page number for extended program-memory pages. The value that you use in the PMR is the same extended-page number that you use in the linker command file.
  - The data mapper register (DMR) stores the extended page number for extended data-memory pages. The value that you use in the DMR is the same extended-page number that you use in the linker command file.

You create the PMR or DMR when you build your extended memory system.

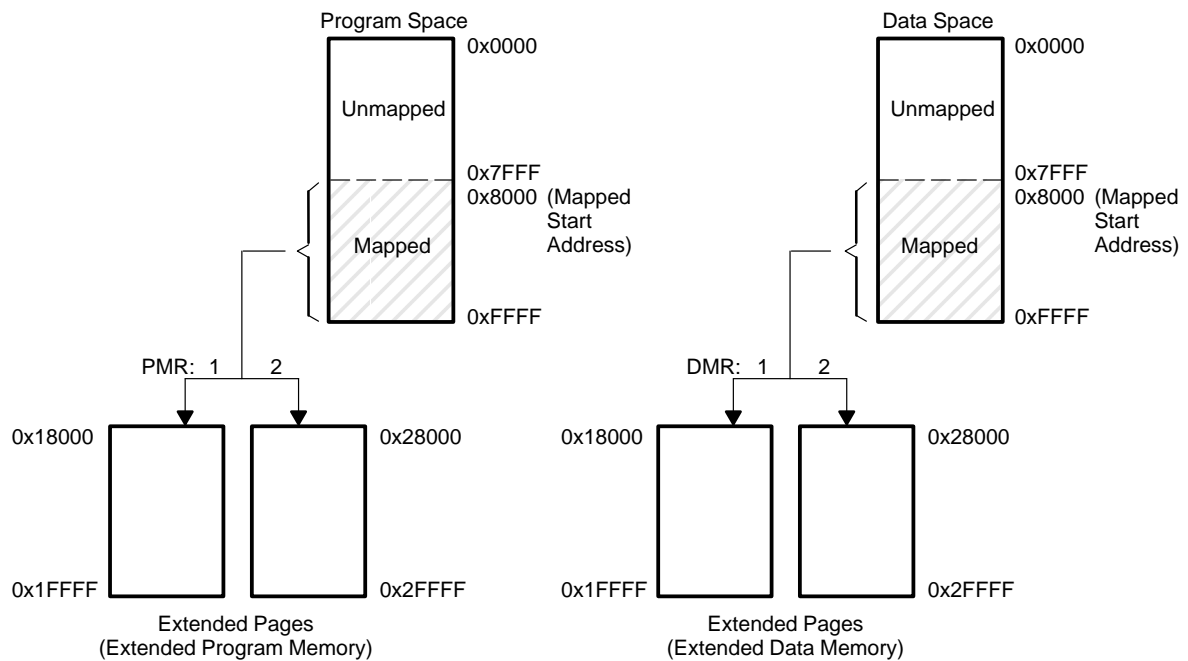
- ☐ The 16 least significant bits (LSBs) represent the native 16-bit address for the symbol.

### Sample extended memory system

Figure 7–1 illustrates a sample extended memory system. In this example, the program and data space are allocated in the same manner:

- ❑ The unmapped memory region is from 0x0000 to 0x7FFF.
- ❑ The mapped memory region is from 0x8000 to 0xFFFF.
- ❑ Two extended pages extend the mapped address space.
- ❑ The mapper register (PMR or DMR) qualifies the mapped address, creating a unique address for each location in each extended page.

Figure 7–1. Sample Extended Memory System



## 7.2 Setting Up Extended Addressing

Before you can use extended memory, you must build a target system that includes extended memory, including hardware that assigns an extended page to an extended memory region. Chapter 5, *Enabling Extended Addressing*, describes how to set up your target system with extended memory.

Once you have built your extended memory system, you must set up the debugger to use extended addressing by doing the following:

- ☐ Describe your extended memory configuration to the debugger
- ☐ Enable extended addressing

This section describes how to perform these tasks.

### *Describing your extended memory configuration to the debugger*

You must describe to the debugger your extended memory configuration and where to look for the PMR or DMR register. To do so, use the `EXT_ADDR_DEF` command. The syntax for this command is:

**ext\_addr\_def** *map start* [{@**prog** | @**data**}], *reg addr* [{@**prog** | @**data** | @**io**}], *mask*

- ☐ The *map start* parameter defines the beginning of the mapped memory range. By default, the *map start* parameter is treated as a program-memory address. However, you can follow it with @**prog** to identify program memory or with @**data** to identify data memory.
- ☐ The *reg addr* parameter defines the location of the mapper register (PMR or DMR).
  - If you are defining program memory, use the address for the PMR.
  - If you are defining data memory, use the address for the DMR.

By default, the *reg addr* parameter is treated as a data-memory address. However, you can follow it with @**prog** to identify program memory, with @**data** to identify data memory, or with @**io** to identify I/O space.

- ☐ The *mask* parameter is a bit mask that represents the size of the PMR or DMR.

For example, if you designed an extended memory system that begins mapping at address 0x8000 in program memory, with the 8-bit PMR located at 0x0100 in I/O space, you would enter:

**ext\_addr\_def** 0x8000@prog, 0x0100@io, 0xff

To avoid entering the `EXT_ADDR_DEF` command each time you invoke the debugger, you can modify your `emuinit.cmd` file to include the command. The debugger reads and executes the commands in the `emuinit.cmd` file each time you invoke the debugger.

### ***Enabling extended addressing***

Before you load your target code, you must enable extended addressing. To do so, use the EXT\_ADDR ON command. The syntax for this command is:

**ext\_addr {on | off}**

You cannot use this command before you define your extended memory configuration with the EXT\_ADDR\_DEF command.

To avoid entering the EXT\_ADDR command each time you invoke the debugger, you can modify your emuinit.cmd file to include the command.

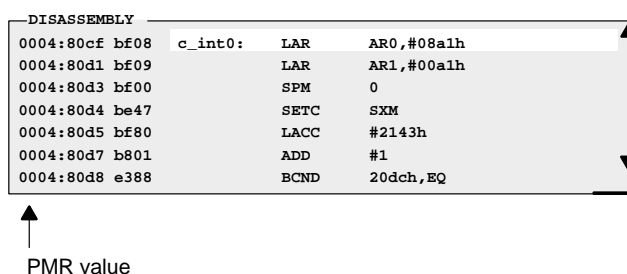


### 7.3 Debugging With Extended Addressing

Once you have set up the debugger for extended addressing (as described in Section 7.2, *Setting Up Extended Memory*, on page 7-4), you can use the debugger to access data or code stored in the extended memory of your system.

When the debugger loader loads a section of code that contains extended addresses, the loader places the addresses in the proper overlays automatically. The debugger also changes the display of the DISASSEMBLY or MEMORY window to show extended addresses. It uses the PMR or DMR value (which contains the extended page number) as a prefix to the address. The DISASSEMBLY window in Figure 7-2 shows addresses that have been accessed from extended memory. The debugger used the PMR value (4) as a prefix to the addresses in extended memory.

Figure 7-2. The DISASSEMBLY Window With Extended Addressing In Use



#### Note:

The extended-page number that is shown in the DISASSEMBLY or MEMORY window does not always represent the value currently stored in the PMR or DMR. However, while stepping through the code, the PMR and the extended-page number are the same.

#### Registers associated with extended addressing: PMR, DMR, and EPC

When you turn extended addressing on (using the EXT\_ADDR ON command), the debugger adds the following registers to the CPU window:

- ☐ One of the following mapper registers:
  - The program mapper register (PMR) displays the current value of the external PMR that you created in your target system.
  - The data mapper register (DMR) displays the current value of the external DMR that you created in your target system.
- ☐ The extended program counter (EPC). The EPC is 32 bits wide and represents the PMR or DMR value concatenated with the PC value.

You can use these registers in expressions. For example, this command sets the PMR to 4 and the program counter (PC) to 0x8000:

```
? EPC= 0x48000
```

This DASM command causes the DISASSEMBLY window to display the current extended PC location:

```
DASM EPC
```

When you turn extended addressing off (using the EXT\_ADDR OFF command), the CPU window no longer displays the PMR, DMR, and EPC registers, and you cannot use these registers in expressions.

### ***New expression syntax***

When you enter one of the commands listed in Table 7–1, you can use an address as one of the command parameters. You can append a suffix to the address to specify whether the address is in program memory (@prog), data memory (@data), or I/O space (@io). When extended addressing is enabled, you can use two new suffixes with the commands in Table 7–1:

- ☐ The @prog16 suffix identifies an address in extended program memory. The debugger uses the current value in the PMR to qualify the address that you enter.
- ☐ The @data16 suffix identifies an address in extended data memory. The debugger uses the current value in the DMR to qualify the address that you enter.

*Table 7–1. Commands That Use the @prog16 and @data16 Suffixes*

Command	Command
? (evaluate expression)	EVAL
ADDR	MEM
DASM	WA
DISP	

**How extended addressing affects symbols**

When you use the debugger with symbol names, the debugger uses the entire extended address associated with the symbol.

When you use a pointer in an expression, the debugger uses the current PMR or DMR value to determine the pointer value. Likewise, if you use the contents of a register as an address, the debugger uses the current PMR or DMR value to determine the correct address. For example, if the PC=0x8000 and the PMR=2 and you enter:

? \*PC

The debugger returns the value at location 0x28000.

Table 7–2 shows typical commands that you could use with the debugger and how the results of the commands are affected by extended addressing.

*Table 7–2. Sample Commands and Results Using Extended Addressing*

If you enter this...	The result is...
<code>dasm 0x8000</code>	Disassembly starting at 0:8000, regardless of the PMR value
<code>dasm 0x8000@prog16</code>	Disassembly starting at x:8000, where x represents the current PMR value
<code>dasm label0</code>	Where label0 is a label located at 0x8000, the result is disassembly starting at 0:8000, regardless of the PMR value
<code>dasm label4</code>	Where label4 is a label located at 0x48000, the result is disassembly starting at 4:8000, regardless of the PMR value
<code>dasm pc</code>	Where PC=0x8000, the result is disassembly starting at x:8000, where x represents the current PMR value
<code>dasm ptr</code>	Where ptr is a VOID* pointing to 0x8000, the result is disassembly starting at x:8000, where x represents the current PMR value
<code>dasm (ptr+1)</code>	Where ptr is a VOID* pointing to 0x8000, the result is disassembly starting at x:8001, where x represents the current PMR value

**Note:**

In the DISASSEMBLY window, addresses associated with symbols are shown as 16-bit addresses. Moreover, if a symbol represents an extended address, you cannot see the entire 32-bit address in the DISASSEMBLY window. However, when you use the symbol name in an expression, the debugger accesses the correct address.

***Using 16-bit expressions with 32-bit extended addressing***

Extended addressing allows you to use 32-bit addresses to reference locations in extended memory. When you use the debugger and specify a 16-bit expression, the debugger uses the following algorithm to determine which memory location to access:

- 1) If you use the @prog16 suffix, the debugger uses the current PMR value as a prefix to the address that you entered. Likewise, if you use the @data16 suffix, the debugger uses the current DMR value as a prefix to the address that you entered.
- 2) If you use a pointer in an expression, the debugger uses the current PMR or DMR value as a prefix to the pointer value, as applicable.
- 3) If you use the contents of a register as an address, the debugger uses the PMR or DMR value as a prefix to the address, as applicable.
- 4) If none of the above is true, the debugger uses 0 as a prefix to the address. This applies to constants and program labels.

***Hardware breakpoints and extended addressing***

Do not use extended addressing when you are using hardware breakpoints. A hardware breakpoint set at a particular address causes the debugger to stop at that address not only for the native 'C5x memory but for every extended page defined by your extended memory. This might confuse the emulator and can cause unexpected results.

# Troubleshooting

---

---

---

This chapter describes some common problems you may encounter while using your emulator or debugger on your workstation.

Topic	Page
A.1 Problems When Booting Your Workstation .....	A-2
A.2 Problems With Multiple Emulators on SunOS .....	A-2
A.3 Problems When Resetting the Emulator .....	A-3
A.4 Problems When Invoking the Debugger .....	A-5
A.5 Additional Emulator and Debugger Problems .....	A-7

## A.1 Problems When Booting Your Workstation

After installing your emulator, problems may occur when you attempt to boot your workstation. The following are typical problems and suggestions to help resolve these problems:

- ☐ Your workstation will not boot when connected to your emulator, even if your emulator is not turned on.
  - 1) Be sure that all of your SCSI cables are connected securely and that the SCSI bus is terminated properly.
  - 2) Remove any unnecessary SCSI devices from the bus.
  - 3) Make sure that the total length of the SCSI bus is less than six meters, including the section of the bus within the SPARCstation 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.

For more information, see the *XDS51x Emulator Installation Guide*.

## A.2 Problems With Multiple Emulators on SunOS

When you have multiple emulators running constantly, they may consume most of the CPU time. To overcome this situation, run the emulators at a lower priority using the UNIX nice command. See the man page entry on nice for more information.

### A.3 Problems When Resetting the Emulator

After you power up the emulator and the workstation, if you have the following problems while 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 `c5x510ws.out` file. You can specify it at this prompt or you can reexecute `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 the `XDS510WS` installation instructions in the *XDS51x Emulator Installation Guide* for more information).
- 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 to 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 SPARCstation, which causes `<m>` to change. Use the corrected configuration file to build the currently executing kernel (see the `XDS510WS` installation instructions in the *XDS51x Emulator Installation Guide* for more information).

- 5) You haven't turned on the XDS510WS, or it hasn't completed its self test. 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 c5x510ws.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 located where emurst can find it.
- 3) The file that you specified to emurst isn't appropriate for this use. Use the c5x510ws.out file that is included with the debugger software.



## A.4 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 emu5x 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 3.4, *Step 3: Ensuring That the Emulator Supports the Debugger*, on page 3-6 for SPARCstations and Section 4.4, *Step 3: Ensuring That the Emulator Supports the Debugger*, on page 4-6, for HP workstations). 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 c5x510ws.out file,  
and # refers to the file /dev/rsd#a, which is associated with the emula-  
tor 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 from 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. (For more information on the `-p` option, see Section 3.4, *Step 3: Ensuring That the Emulator Supports the Debugger*, on page 3-6, for SPARCstations and Section 4.4, *Step 3: Ensuring That the Emulator Supports the Debugger*, on page 4-6, for HP workstations.)
- 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, reexecute 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 cpu_name` debugger option.
- The debugger couldn't find the `cpu_name` that you specified with the `-n` option in your board configuration file.

Reexecute the debugger with the `-n` debugger option, specifying the name of a 'C5x device 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 that your emulation cable is firmly attached both to the XDS510WS and to your target system.
- 7) Make sure that 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 message at the operating-system command line when trying to execute the `emu5x` command:

```
emu5x: 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 "`emu5x*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.

---

## A.5 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 emu5x or emurst under SunOS 4.1.x, refer to the following explanations.

These messages are status messages, not error messages.

❑ `<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 SPARCstation in a certain amount of time. This can be caused by one of these conditions:

- 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.

❑ `<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 to be 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.

❑ <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.

❑ 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.

---

# Index

@data16 suffix 7-7

@prog16 suffix 7-7

## A

arrow keys

for HP workstations 4-12

for SPARCstations 3-12

autoexec.bat file 1-6 to 1-8

## B

batch files

autoexec.bat 1-6 to 1-8

board.cfg

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

board.dat

for DOS 1-3

for HP workstations 4-2

for OS/2 2-3

for SPARCstations 3-2

for Windows 3.x 1-3

config.sys 2-6 to 2-8

.cshrc

for HP workstations 4-8 to 4-10

for SPARCstations 3-8 to 3-10

emuinit.cmd

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

emurst

for DOS 1-9

for HP workstations 4-6

batch files, emurst (continued)

for OS/2 2-9

for SPARCstations 3-6

for Windows 3.x 1-9

init.clr

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

initialization

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

mono.clr

for DOS 1-4

for HP workstations 4-3

for OS/2 2-4

for SPARCstations 3-3

for Windows 3.x 1-4

screen sizes

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

startup.cmd 2-9

board.cfg file

for DOS 1-3

for HP workstations 4-3

for OS/2 2-3

for SPARCstations 3-3

for Windows 3.x 1-3

board.dat file

defined

for DOS 1-3

for HP workstations 4-2

for OS/2 2-3

board.dat file, defined (continued)  
     *for SPARCstations* 3-2  
     *for Windows 3.x* 1-3  
 describing target system to debugger  
     *for DOS* 1-10  
     *for HP workstations* 4-7  
     *for OS/2* 2-10  
     *for SPARCstations* 3-7  
     *for Windows 3.x* 1-10  
 error messages  
     *for DOS* 1-13  
     *for OS/2* 2-12  
     *for Windows 3.x* 1-13  
 booting, problems A-2

## C

c5x510ws.out file  
     *for HP workstations* 4-2 4-6  
     *for SPARCstations* 3-2 3-6  
 c5xhll directory  
     *for DOS* 1-6  
     *for HP workstations* 4-5, 4-8  
     *for OS/2* 2-6  
     *for SPARCstations* 3-5, 3-8  
     *for Windows 3.x* 1-6  
 CD-ROM  
   mounting  
     *for HP workstations* 4-4  
     *for SPARCstations* 3-4  
   requirements  
     *for DOS* 1-2  
     *for HP workstations* 4-2  
     *for OS/2* 2-2  
     *for SPARCstations* 3-2  
     *for Windows 3.x* 1-2  
   retrieving files from  
     *for DOS* 1-5  
     *for HP workstations* 4-5  
     *for OS/2* 2-5  
     *for SPARCstations* 3-5  
     *for Windows 3.x* 1-5  
   unmounting  
     *for HP workstations* 4-5  
     *for SPARCstations* 3-5  
 CHDIR (CD) command 6-3  
 chmod UNIX command A-3

COFF2 6-1  
 colors, mapping with X Windows  
     *for HP workstations* 4-13  
     *for SPARCstations* 3-13  
 composer utility  
     *for DOS* 1-3  
     *for HP workstations* 4-3  
     *for OS/2* 2-3  
     *for SPARCstations* 3-3  
     *for Windows 3.x* 1-3  
 config.sys file  
     setting the IOPL option 2-8  
     setting up environment variables 2-6 to 2-8  
 configuration file, board.dat  
     *for DOS* 1-3, 1-10  
     *for HP workstations* 4-2 4-7  
     *for OS/2* 2-3, 2-10  
     *for SPARCstations* 3-2 3-7  
     *for Windows 3.x* 1-3, 1-10  
 .cshrc file  
     *for HP workstations* 4-8  
     *for SPARCstations* 3-8 to 3-10  
     invoking  
       *for HP workstations* 4-10  
       *for SPARCstations* 3-10  
 current directory, changing 6-3  
 customizing the display  
     *for HP workstations* 4-13  
     *for SPARCstations* 3-13  
 init.clr file  
     *for DOS* 1-3  
     *for HP workstations* 4-3  
     *for OS/2* 2-3  
     *for SPARCstations* 3-3  
     *for Windows 3.x* 1-3  
 mono.clr file  
     *for DOS* 1-4  
     *for HP workstations* 4-3  
     *for OS/2* 2-4  
     *for SPARCstations* 3-3  
     *for Windows 3.x* 1-4  
 screen sizes  
     *for DOS* 1-3  
     *for HP workstations* 4-3  
     *for OS/2* 2-3  
     *for SPARCstations* 3-3  
     *for Windows 3.x* 1-3

# D

- D\_DIR environment variable
  - for DOS 1-7
  - for HP workstations 4-8
  - for OS/2 2-7
  - for SPARCstations 3-8
  - for Windows 3.x 1-7
- D\_OPTIONS environment variable
  - for DOS 1-7
  - for HP workstations 4-9
  - for OS/2 2-7
  - for SPARCstations 3-9
  - for Windows 3.x 1-7
  - troubleshooting A-6
- D\_SRC environment variable
  - for DOS 1-7
  - for HP workstations 4-8
  - for OS/2 2-7
  - for SPARCstations 3-8
  - for Windows 3.x 1-7
- data mapper register (DMR)
  - defined 7-2
  - describe to debugger 7-4
  - in CPU window 7-6
- debugger
  - communicating with your target system
    - for DOS 1-10
    - for HP workstations 4-7
    - for OS/2 2-10
    - for SPARCstations 3-7
    - for Windows 3.x 1-10
  - displaying on a different machine
    - for HP workstations 4-10
    - for SPARCstations 3-10
  - environment setup
    - for DOS 1-6 to 1-8
    - for HP workstations 4-8 to 4-10
    - for OS/2 2-6 to 2-8
    - for SPARCstations 3-8 to 3-10
    - for Windows 3.x 1-6 to 1-8
  - font changes
    - for HP workstations 4-13
    - for SPARCstations 3-13
  - installation
    - for DOS 1-1 to 1-14
    - for HP workstations 4-1 to 4-13
    - for OS/2 2-1 to 2-13
    - for SPARCstations 3-1 to 3-13
  - debugger, installation (continued)
    - for Windows 3.x 1-1 to 1-14
    - troubleshooting A-5 to A-6
  - invoking
    - for DOS 1-11
    - for HP workstations 4-11
    - for OS/2 2-11
    - for SPARCstations 3-11
    - for Windows 3.x 1-11
  - options
    - for DOS 1-7
    - for HP workstations 4-9
    - for OS/2 2-7
    - for SPARCstations 3-9
    - for Windows 3.x 1-7
  - troubleshooting A-5 to A-6, A-7 to A-8
  - using the X Window System
    - for HP workstations 4-12 to 4-13
    - for SPARCstations 3-12 to 3-13
  - using with Windows 3.x 1-14
- default
  - memory map
    - for DOS 1-3
    - for HP workstations 4-3
    - for OS/2 2-3
    - for SPARCstations 3-3
    - for Windows 3.x 1-3
  - screen configuration file
    - for DOS 1-4
    - for HP workstations 4-3
    - for OS/2 2-4
    - for SPARCstations 3-3
    - for Windows 3.x 1-4
- DIR command 6-3
- directories
  - auxiliary files
    - for DOS 1-7
    - for HP workstations 4-8
    - for OS/2 2-7
    - for SPARCstations 3-8
    - for Windows 3.x 1-7
  - c5xhll directory
    - for DOS 1-6
    - for HP workstations 4-5, 4-8
    - for OS/2 2-6
    - for SPARCstations 3-5, 3-8
    - for Windows 3.x 1-6
  - changing current directory 6-3

- directories (continued)
  - debugger software
    - for DOS 1-6
    - for HP workstations 4-5, 4-8
    - for OS/2 2-6
    - for SPARCstations 3-5, 3-8
    - for Windows 3.x 1-6
  - identifying additional source directories
    - for DOS 1-7
    - for HP workstations 4-8
    - for OS/2 2-7
    - for SPARCstations 3-8
    - for Windows 3.x 1-7
  - listing contents of current directory 6-3
  - relative pathnames 6-3
- display, font changes
  - for HP workstations 4-13
  - for SPARCstations 3-13
- DISPLAY environment variable
  - for HP workstations 4-10
  - for SPARCstations 3-10
- display requirements
  - for DOS 1-2
  - for HP workstations 4-2
  - for OS/2 2-2
  - for SPARCstations 3-2
  - for Windows 3.x 1-2
- DOS
  - CD-ROM requirements 1-2
  - display requirements 1-2
  - emulator controller requirements 1-2
  - graphics card requirements 1-2
  - hardware requirements 1-2
  - host system 1-2
  - installation
    - debugger software 1-5
    - emulator controller 1-5
    - environment variables 1-6 to 1-8
    - error messages 1-12
    - verifying 1-11
  - memory requirements 1-2
  - mouse requirements 1-2
  - operating system 1-3
  - resetting emulator 1-9
  - setting up debugger environment 1-6 to 1-8
  - software requirements 1-3
  - target system 1-3
  - using Windows 3.x 1-14

- downloading code
  - for HP workstations 4-6
  - for SPARCstations 3-6
- driver file
  - selecting 3-6
  - troubleshooting A-3, A-5

## E

- emu5x command
  - for DOS 1-3
  - for HP workstations 4-2
  - for SPARCstations 3-2
  - troubleshooting A-5 to A-6, A-7 to A-8
  - verifying the installation
    - for DOS 1-11
    - for HP workstations 4-11
    - for SPARCstations 3-11
- emu5xo command
  - for OS/2 2-3
  - verifying the installation 2-11
- emu5xw command
  - for Windows 3.x 1-3
  - verifying the installation 1-11
- emu5xwm command
  - for Windows 3.x 1-3
  - verifying the installation 1-11
- emuinit.cmd file
  - for DOS 1-3
  - for HP workstations 4-3
  - for OS/2 2-3
  - for SPARCstations 3-3
  - for Windows 3.x 1-3
- emulator
  - additional tools
    - for DOS 1-3
    - for HP workstations 4-2
    - for OS/2 2-3
    - for SPARCstations 3-2
    - for Windows 3.x 1-3
- board.cfg file
  - for DOS 1-3
  - for HP workstations 4-3
  - for OS/2 2-3
  - for SPARCstations 3-3
  - for Windows 3.x 1-3
- board.dat file
  - for DOS 1-3
  - for HP workstations 4-2



- emulator, board.dat file (continued)
  - for OS/2 2-3
  - for SPARCstations 3-2
  - for Windows 3.x 1-3
- booting problems A-2
- debugger environment
  - for DOS 1-6 to 1-8
  - for HP workstations 4-8 to 4-10
  - for OS/2 2-6 to 2-8
  - for Windows 3.x 1-6 to 1-8
  - SPARCstation 3-8 to 3-10
- debugger installation
  - for DOS 1-1 to 1-14
  - for HP workstations 4-1 to 4-13
  - for OS/2 2-1 to 2-13
  - for SPARCstations 3-1 to 3-13
  - for Windows 3.x 1-1 to 1-14
- driver file access A-3
- hardware requirements
  - for DOS 1-2
  - for HP workstations 4-2
  - for OS/2 2-2
  - for SPARCstations 3-2
  - for Windows 3.x 1-2
- host system
  - for DOS 1-2
  - for HP workstations 4-2
  - for OS/2 2-2
  - for SPARCstations 3-2
  - for Windows 3.x 1-2
- installation. *See XDS51x Emulator Installation Guide*
- memory, default map
  - for DOS 1-3
  - for HP workstations 4-3
  - for OS/2 2-3
  - for SPARCstations 3-3
  - for Windows 3.x 1-3
- operating system
  - for DOS 1-3
  - for HP workstations 4-2
  - for OS/2 2-3
  - for SPARCstations 3-2
  - for Windows 3.x 1-3
- requirements
  - for DOS. *See DOS*
  - for HP workstations. *See HP workstations*
  - for OS/2. *See OS/2*
  - for SPARCstations. *See SPARCstations*
  - for Windows 3.x. *See Windows 3.x*
- emulator (continued)
  - resetting
    - for DOS 1-9
    - for OS/2 2-9
    - for Windows 3.x 1-9
    - problems A-3 to A-4, A-7 to A-8
  - screen configuration files
    - for DOS 1-3
    - for HP workstations 4-3
    - for OS/2 2-3
    - for SPARCstations 3-3
    - for Windows 3.x 1-3
  - software requirements
    - for DOS 1-3
    - for HP workstations 4-2 to 4-3
    - for OS/2 2-3
    - for SPARCstations 3-2
    - for Windows 3.x 1-3
  - troubleshooting A-1 to A-8
- emulator controller, requirements
  - for DOS 1-2
  - for HP workstations 4-2
  - for OS/2 2-2
  - for SPARCstations 3-2
  - for Windows 3.x 1-2
- EMULATOR file, troubleshooting A-3
- emurst command
  - for DOS 1-3, 1-9
  - for HP workstations 4-2 4-6
  - for OS/2 2-3, 2-9
  - for SPARCstations 3-2 3-6
  - for Windows 3.x 1-3
  - specifying parameters A-3
  - syntax
    - for DOS 1-9
    - for HP workstations 4-6
    - for OS/2 2-9
    - for SPARCstations 3-6
    - for Windows 3.x 1-9
  - troubleshooting A-3 to A-4, A-7 to A-8
  - when invoking the debugger A-5
- end key
  - for HP workstations 4-12
  - for SPARCstations 3-12
- environment variables
  - D\_DIR
    - for DOS 1-7
    - for HP workstations 4-8
    - for OS/2 2-7

environment variables, D\_DIR (continued)  
     for SPARCstations 3-8  
     for Windows 3.x 1-7

D\_OPTIONS  
     for DOS 1-7  
     for HP workstations 4-9  
     for OS/2 2-7  
     for SPARCstations 3-9  
     for Windows 3.x 1-7

D\_SRC  
     for DOS 1-7  
     for HP workstations 4-8  
     for OS/2 2-7  
     for SPARCstations 3-8  
     for Windows 3.x 1-7

DISPLAY  
     for HP workstations 4-10  
     for SPARCstations 3-10

displaying the debugger on a different machine  
     for HP workstations 4-10  
     for SPARCstations 3-10

for DOS 1-6 to 1-8  
     for HP workstations 4-8 to 4-10  
     for OS/2 2-6 to 2-8  
     for SPARCstations 3-8 to 3-10  
     for Windows 3.x 1-6 to 1-8

error messages, installation  
     for DOS 1-12  
     for OS/2 2-12  
     for Windows 3.x 1-12

EXT\_ADDR command 7-5

EXT\_ADDR\_DEF command 7-4

extended addressing 6-4, 7-1 to 7-10  
     @data16 suffix 7-7  
     @prog16 suffix 7-7  
     16-bit expressions with 32-bit addressing 7-9  
     affect on symbols 7-8  
     building system 5-4 to 5-8  
     data mapper register (DMR)  
         defined 7-2  
         describe to debugger 7-4  
         in CPU window 7-6  
     debugging with extended addressing 7-6 to 7-10  
     described 5-2  
     describing memory configuration 7-4  
     enabling 7-5  
     extended program counter (EPC) 7-6  
     hardware breakpoints 7-9  
     introduction 5-1

extended addressing (continued)  
     native address 7-2  
     overlay number 7-2  
     program mapper register (PMR)  
         defined 7-2  
         describe to debugger 7-4  
         in CPU window 7-6  
     sample commands and results 7-8  
     sample extended memory architecture 7-3  
     sample system 5-3  
     setting up mapped extended memory 7-4 to 7-5  
     understanding extended memory architecture 7-2 to 7-3

extended memory architecture  
     See also extended addressing  
     debugging with extended addressing 7-6 to 7-10  
     setting up 7-4 to 7-5  
     understanding 7-2 to 7-3

extended program counter (EPC) 7-6

## F

-f debugger option, troubleshooting A-5

FILE command, changing the current directory 6-3

font changes  
     for HP workstations 4-13  
     for SPARCstations 3-13

font file, troubleshooting A-6

function key mapping  
     for HP workstations 4-12  
     for SPARCstations 3-12

## G

graphics card requirements  
     for DOS 1-2  
     for OS/2 2-2  
     for Windows 3.x 1-2

## H

hardware breakpoints, extended addressing 7-9

hardware checklist  
     for DOS 1-2  
     for HP workstations 4-2  
     for OS/2 2-2  
     for SPARCstations 3-2  
     for Windows 3.x 1-2

hardware installation. *See XDS51x Emulator Installation Guide*

home key

- for HP workstations 4-12
- for SPARCstations 3-12

host system

- for DOS 1-2
- for HP workstations 4-2
- for OS/2 2-2
- for SPARCstations 3-2
- for Windows 3.x 1-2

HP workstations

- CD-ROM requirements 4-2
- display requirements 4-2
- emulator controller requirements 4-2
- hardware requirements 4-2
- host system 4-2
- installation
  - debugger software* 4-4
  - emulator controller* 4-4
  - verifying* 4-11
- operating system 4-2
- setting up debugger environment 4-8 to 4-10
- software requirements 4-2 to 4-3
- target system 4-3

## I

init.clr file

- for DOS 1-3
- for HP workstations 4-3
- for OS/2 2-3
- for SPARCstations 3-3
- for Windows 3.x 1-3

initialization batch files, emuinit.cmd

- for DOS 1-3
- for HP workstations 4-3
- for OS/2 2-3
- for SPARCstations 3-3
- for Windows 3.x 1-3

insert key

- for HP workstations 4-12
- for SPARCstations 3-12

installation

- debugger software
  - for DOS* 1-5
  - for HP workstations* 4-4
  - for OS/2* 2-5

installation, debugger software (continued)

- for SPARCstations* 3-4
- for Windows 3.x* 1-5

emulator. *See XDS51x Emulator Installation Guide*

emulator controller

- for DOS* 1-5
- for HP workstations* 4-4
- for OS/2* 2-5
- for SPARCstations* 3-4
- for Windows 3.x* 1-5

error messages

- for DOS* 1-12
- for OS/2* 2-12
- for Windows 3.x* 1-12

troubleshooting A-1 to A-8

verifying

- for DOS* 1-11
- for HP workstations* 4-11
- for OS/2* 2-11
- for SPARCstations* 3-11
- for Windows 3.x* 1-11

invoking

- .cshrc file
  - for HP workstations* 4-10
  - for SPARCstations* 3-10
- debugger
  - for DOS* 1-11
  - for HP workstations* 4-11
  - for OS/2* 2-11
  - for SPARCstations* 3-11
  - for Windows 3.x* 1-11
  - troubleshooting* A-5 to A-6

IOPL, setting 2-8

ipcrm UNIX command A-3

ipcs UNIX command A-3

IPCSEMAPHORE option A-3

## K

keyboard, mapping keys

- for HP workstations 4-12
- for SPARCstations 3-12

keys, special keys with the X Window System

- for HP workstations 4-12
- for SPARCstations 3-12

keysym label

- for HP workstations 4-12
- for SPARCstations 3-12

**L**

## labels, keysym

- for HP workstations 4-12
- for SPARCstations 3-12

## LED lights, after emurst

- for HP workstations 4-6
- for SPARCstations 3-6

## linker command file, enabling extended addressing 5-6 to 5-8

**M**

## mapping keys for use with X Windows

- for HP workstations 4-12
- for SPARCstations 3-12

## MEM command 6-1

## memory

- adding to device 5-4

## default map

- for DOS 1-3
- for HP workstations 4-3
- for OS/2 2-3
- for SPARCstations 3-3
- for Windows 3.x 1-3

## extended addressing

- debugging with* 7-6 to 7-10
- setting up* 7-4 to 7-5
- understanding* 7-2 to 7-3

## extended architecture, sample memory architecture 7-3

## requirements

- for DOS 1-2
- for OS/2 2-2
- for Windows 3.x 1-2

## memory mapping

## emuinit.cmd file

- for DOS 1-3
- for HP workstations 4-3
- for OS/2 2-3
- for SPARCstations 3-3
- for Windows 3.x 1-3

- extended addressing. *See* extended addressing with extended addressing 7-2

## MEMORY window 6-1

## messages, installation errors

- for DOS 1-12
- for OS/2 2-12
- for Windows 3.x 1-12

## modifying, current directory 6-3

## mono.clr file

- for DOS 1-4
- for HP workstations 4-3
- for OS/2 2-4
- for SPARCstations 3-3
- for Windows 3.x 1-4

## monochrome monitors, color mapping with X Windows

- for HP workstations 4-13
- for SPARCstations 3-13

## mouse, requirements

- for DOS 1-2
- for OS/2 2-2
- for Windows 3.x 1-2

**N**

## -n debugger option

- for DOS 1-11
- for HP workstations 4-11
- for OS/2 2-11
- for SPARCstations 3-11
- for Windows 3.x 1-11
- troubleshooting A-6

**O**

## OpenWindows, finding the font file A-6

## operating system

- for DOS 1-3
- for HP workstations 4-2
- for OS/2 2-3
- for SPARCstations 3-2
- for Windows 3.x 1-3

## optional files

- for DOS 1-3
- for HP workstations 4-3
- for OS/2 2-3
- for SPARCstations 3-3
- for Windows 3.x 1-3

## options, IPCSEMAPHORE A-3

- OS/2
  - CD-ROM requirements 2-2
  - display requirements 2-2
  - emulator controller requirements 2-2
  - graphics card requirements 2-2
  - hardware requirements 2-2
  - host system 2-2
  - installation
    - debugger software* 2-5
    - emulator controller* 2-5
    - environment variables* 2-6 to 2-8
    - error messages* 2-12
    - verifying* 2-11
  - memory requirements 2-2
  - mouse requirements 2-2
  - operating system 2-3
  - resetting emulator 2-9
  - setting up debugger environment 2-6 to 2-8
  - software requirements 2-3
  - target system 2-3
- overlay number 7-2

## P

- p debugger option
  - selecting the driver file 3-6
    - troubleshooting* A-3, A-5
  - using with emurst
    - for DOS* 1-9
    - for HP workstations* 4-6
    - for OS/2* 2-9
    - for SPARCstations* 3-6
    - for Windows 3.x* 1-9
  - with D\_OPTIONS environment variable
    - for DOS* 1-9, 1-13
    - for OS/2* 2-10, 2-12
    - for Windows 3.x* 1-9, 1-13
- page-down key
  - for HP workstations 4-12
  - for SPARCstations 3-12
- page-up key
  - for HP workstations 4-12
  - for SPARCstations 3-12
- PATH statement
  - for DOS 1-6
  - for OS/2 2-6
  - for Windows 3.x 1-6

- path statement
  - for HP workstations 4-8
  - for SPARCstations 3-8
- permissions, root access
  - for HP workstations 4-2
  - for SPARCstations 3-2
- port address
  - for DOS 1-9, 1-13
  - for OS/2 2-10, 2-12
  - for Windows 3.x 1-9, 1-13
- program mapper register (PMR)
  - defined 7-2
  - describe to debugger 7-4
  - in CPU window 7-6

## R

- RAM speed
  - for DOS 1-2
  - for OS/2 2-2
  - for Windows 3.x 1-2
- registers, adding to device 5-5
- relative pathnames 6-3
- required files
  - for DOS 1-3
  - for HP workstations 4-2
  - for OS/2 2-3
  - for SPARCstations 3-2
  - for Windows 3.x 1-3
- required tools
  - for DOS 1-3
  - for HP workstations 4-2
  - for OS/2 2-3
  - for SPARCstations 3-2
  - for Windows 3.x 1-3
- resetting
  - emurst command
    - for DOS* 1-9
    - for OS/2* 2-9
    - for Windows 3.x* 1-9
  - emurst file, troubleshooting A-3 to A-4, A-7 to A-8
- retrieving files from CD-ROM
  - for DOS 1-5
  - for HP workstations 4-5
  - for OS/2 2-5
  - for SPARCstations 3-5
  - for Windows 3.x 1-5

root privileges  
     for HP workstations 4-2  
     for SPARCstations 3-2

## S

software checklist  
     for DOS 1-3  
     for HP workstations 4-2 to 4-3  
     for OS/2 2-3  
     for SPARCstations 3-2  
     for Windows 3.x 1-3

source files, enabling extended memory 5-5

SPARCstations  
     CD-ROM requirements 3-2  
     display requirements 3-2  
     emulator controller requirements 3-2  
     hardware requirements 3-2  
     host system 3-2  
     installation  
         debugger software 3-4  
         emulator controller 3-4  
         verifying 3-11  
     operating system 3-2  
     setting up debugger environment 3-8 to 3-10  
     software requirements 3-2  
     target system 3-2

special keys X Window System  
     for HP workstations 4-12  
     for SPARCstations 3-12

startup.cmd file 2-9

symbols, affect of extended addressing 7-8

system commands  
     CD command 6-3  
     DIR command 6-3

## T

target system  
     board.dat file  
         for DOS 1-3  
         for HP workstations 4-2  
         for OS/2 2-3  
         for SPARCstations 3-2  
         for Windows 3.x 1-3  
     describing to the debugger  
         for DOS 1-3, 1-10  
         for HP workstations 4-3, 4-7

target system, describing to the debugger  
     (continued)  
         for OS/2 2-3, 2-10  
         for SPARCstations 3-3, 3-7  
         for Windows 3.x 1-3, 1-10

troubleshooting A-1 to A-8  
     when booting workstation A-2  
     when invoking the debugger A-5 to A-6  
     when resetting emulator A-3 to A-4, A-7 to A-8  
     when using the debugger A-7 to A-8

## U

utilities  
     xev  
         for HP workstations 4-12  
         for SPARCstations 3-12  
     xmodmap  
         for HP workstations 4-12  
         for SPARCstations 3-12  
     xrdb  
         for HP workstations 4-13  
         for SPARCstations 3-13

## V

verifying, installation  
     for DOS 1-11  
     for HP workstations 4-11  
     for OS/2 2-11  
     for SPARCstations 3-11  
     for Windows 3.x 1-11  
     troubleshooting A-5

## W

WA command 6-2

WATCH window 6-2

WD command 6-2

window name parameter  
     MEMORY window 6-1  
     WATCH window 6-2

Windows 3.x  
     CD-ROM requirements 1-2  
     display requirements 1-2  
     emulator controller requirements 1-2  
     graphics card requirements 1-2  
     hardware requirements 1-2  
     host system 1-2

## Windows 3.x (continued)

## installation

- debugger software* 1-5
- emulator controller* 1-5
- environment variables* 1-6 to 1-8
- error messages* 1-12
- verifying* 1-11

memory requirements 1-2

mouse requirements 1-2

operating system 1-3

resetting emulator 1-9

setting up debugger environment 1-6 to 1-8

software requirements 1-3

target system 1-3

using with the debugger 1-14

workstation, problems when booting A-2

WR command 6-2



## -x debugger option

for DOS 1-8

for HP workstations 4-9

for OS/2 2-8

for SPARCstations 3-9

for Windows 3.x 1-8

using with emurst

*for DOS* 1-9

## -x debugger option, using with emurst (continued)

*for HP workstations* 4-6*for OS/2* 2-9*for SPARCstations* 3-6*for Windows 3.x* 1-9

## X Window System

## color mapping

*for HP workstations* 4-13*for SPARCstations* 3-13

## displaying debugger on a different machine

*for HP workstations* 4-10*for SPARCstations* 3-10

## special keys

*for HP workstations* 4-12*for SPARCstations* 3-12

## using with the debugger

*for HP workstations* 4-12 to 4-13*for SPARCstations* 3-12 to 3-13

## .Xdefaults file

*for HP workstations* 4-13*for SPARCstations* 3-13

## xev utility

*for HP workstations* 4-12*for SPARCstations* 3-12

## xmodmap utility

*for HP workstations* 4-12*for SPARCstations* 3-12

## xrdb utility

*for HP workstations* 4-13*for SPARCstations* 3-13