TMS320C2xx PC Emulator Installation Guide

SPRU152
September 1995

IMPORTANT NOTICE

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

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

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

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

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

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

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

WARNING

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

TRADEMARKS

MS-DOS and MS-Windows are registered trademarks of Microsoft Corp.

Pentium is a trademark of Intel Corporation.

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

Contents

Figures

Tables

Chapter 1

Installing the Emulator and C Source Debugger With OS/2

This chapter helps you with the following:

- ❑ Installing the XDS510-PC (the emulation controller interface hardware) in a PC.
- ❑ Installing the TMS320C2xx C source debugger on a PC running OS/2™.

When you complete the installation, turn to the *TMS320C2xx C Source Debugger User's Guide.*

Topic **Page**

1.1 What You'll Need

The following checklists detail items that are shipped with the 'C2xx C source debugger and XDS510-PC emulator and additional items you'll need to use these tools.

Hardware checklist

☐	**host**	A 32-bit 386-based PC (or higher) or a Pentium™ PC with an ISA/EISA bus and a 1.44-megabyte floppy-disk drive
☐	**memory**	Minimum of 8 megabytes
☐	**display**	Monochrome or color (color recommended)
☐	**slot**	One 16-bit slot
☐	**XDS510-PC power requirements**	Approximately 1 ampere @ 5 volts (5 watts)
☐	**target system**	A board with a 'C2xx and a IEEE 1149.1 standard interface
☐	**connector to target system**	14-pin connector (two rows of seven pins)—for more information about this connector, refer to the *JTAG/MPSD Emulation Technical Reference*.
☐	**optional hardware**	A Microsoft-compatible mouse
☐		An EGA- or VGA-compatible graphics display card and a large monitor.
☐	**miscellaneous materials**	Blank, formatted disks

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

☐	**operating system**	OS/2 (version 1.1 or later, version 2.x recommended)
☐	**software tools**	TMS320C1x/C2x/C2xx/C5x assembler and linker Optional: TMS320C1x/C2x/C2xx/C5x C compiler
☐	**required files** †	*emu2xxo.exe* is the executable file that invokes the C source debugger for the emulator.

† Included as part of the debugger package

- ☐ † *board.dat* describes your target system to the debugger.
- ☐ † *board.cfg* is the default name for the text file that describes your target system to the debugger. If you plan to create a target system that contains anything other than a single 'C2xx, you must make a text file that describes the target system. Once you have created the file, you must translate it to a binary, conditioned format so that the debugger can understand it; this reformatted file is called board.dat.

  Refer to the *Describing Your Target System to the Debugger* appendix in the *TMS320C2xx C Source Debugger User's Guide* for more information.
- ☐ † *composer.exe* is the utility that translates the board.cfg file to a binary, conditioned format.
- ☐ † *emurst.exe* resets the 'C2xx emulator

optional files

- ☐ † *emuinit.cmd* is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C209 memory map. If this file isn't present when you first invoke the debugger, then all memory is invalid at first. You *must* modify this file to match your target system's memory map. For information about setting up your own memory map, refer to the *Defining a Memory Map* chapter in the *TMS320C2xx C Source Debugger User's Guide*.
- ☐ † *init.clr* 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.
- ☐ † *init.25* and *init.50* have been provided for basic 80x25 and 80x50 screen sizes, respectively. The init.clr file brings up the debugger in 80x25 mode. To bring the debugger up in another mode, copy one of the init.xx files to the init.clr file.
- ☐ † The default configuration is for color monitors; an additional file, *mono.clr,* can be used with monochrome monitors. When you first invoke 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 the screen configuration files and about setting up your own screen configuration, refer to the *Customizing the Debugger Display* chapter in the *TMS320C2xx C Source Debugger User's Guide*.

† Included as part of the debugger package

1.2 Step 1: Installing the XDS510-PC in Your PC

This section contains the hardware installation information for the XDS510-PC emulator.

Preparing the XDS510-PC for installation

Before you install the XDS510-PC, you must be sure that the board's switches are set to correctly identify the I/O space that the board can use.The emulator uses 32 bytes of the PC I/O space; two switches on the board identify this space.

Figure 1–1 shows where these switches are on the XDS510-PC and identifies the switch numbers.

Switches are shipped in the default settings shown here and are listed in Table 1–1. If you use an I/O space that differs from the default, change the switch settings. Table 1–1 also shows alternate settings.

In most cases, you can leave the switch settings in the default position. However, you must ensure that the emulator I/O space does not conflict with other bus settings. For example, if you've installed a bus mouse in your system, you may not be able to use the default switch settings for the I/O address space—the mouse might use this space. Refer to your PC technical reference manual and your other hardware-board manuals to see if there are any I/O space conflicts. If you find a conflict, use one of the alternate settings shown in Table 1–1.

Figure 1–1. XDS510-PC I/O Switches

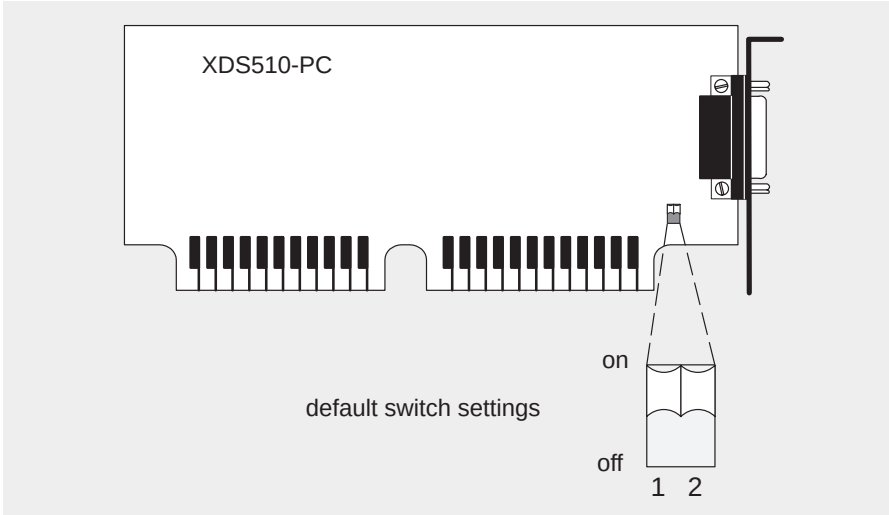


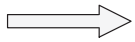
Table 1–1. XDS510-PC Switch Settings

Address Range	Switch # 1	Switch # 2
0x0240–0x025F	on	on
0x0280–0x029F	on	off
0x0320–0x033F	off	on
0x0340–0x035F	off	off

Some of the other installation steps require you to know which switch settings you used. If you reset the I/O switches, note the modified settings here for later reference.

Table 1–2. Your Switch Settings

Address Range	Switch # 1	Switch # 2



1) **To minimize the risk of electric shock and fire hazard, be sure that all major components that you interface with Texas Instruments devices are limited in energy and certified by one or more of the following agencies: UV, CSA, VDE, or TUV.**

2) **To minimize the risk of personal injury, *always* turn off the power to your PC and unplug the power cord before installing the XDS510-PC.**

Setting the XDS510-PC into your PC

After you've prepared the XDS510-PC for installation, follow these steps.

1) Turn off the power to the PC, and unplug the power cord.
2) Remove the cover of your PC.
3) Remove the mounting bracket from an unused 16-bit slot.
4) Install the XDS510-PC in a 16-bit slot (see Figure 1–2).
5) Tighten down the mounting bracket.
6) Plug the target cable into the XDS510-PC (see Figure 1–3). The cable is a 25-pin DSUB connector, shaped to ensure proper connection.
7) Replace the PC cover.
8) Plug in the power cord, and turn on the power to the PC.

Figure 1–2. XDS510-PC Installation

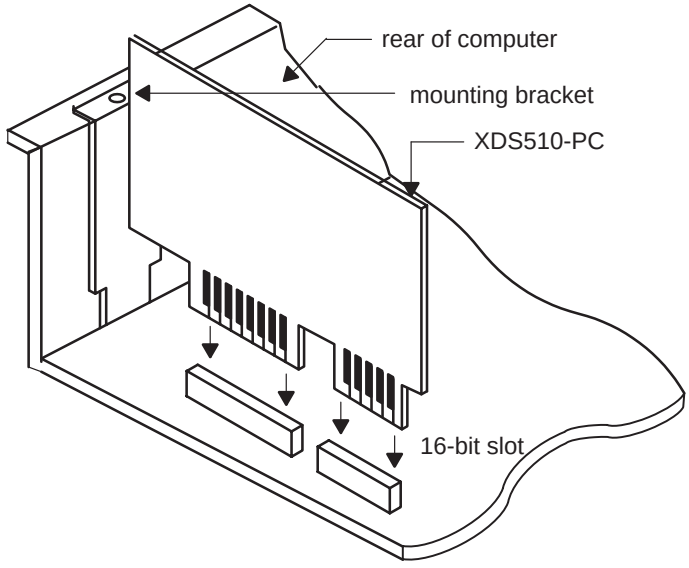
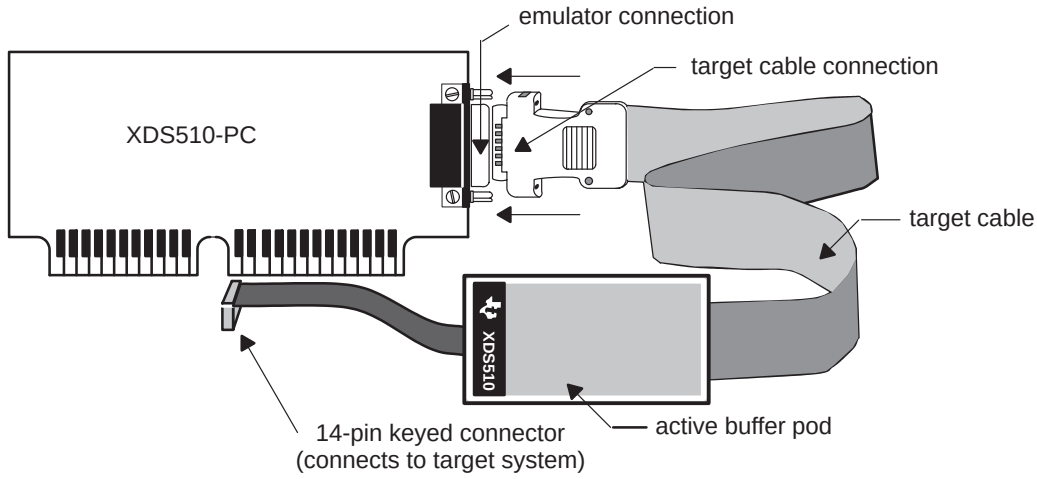


Figure 1–3. Connecting the Target Cable to the XDS510-PC



Don’t connect or disconnect the target cable while the PC is powered up.

Be very careful with the target cable connectors. Connect them gently; forcing the connectors into position may damage them.

Remember, the connector is keyed. Be sure to connect the cable so that the key fits into its slot.

1.3 Step 2: Connecting the XDS510-PC to Your Target System

Figure 1–4 shows a typical setup using the XDS510-PC emulator, target cable, and your target system.

Figure 1–5 shows how you connect the XDS510-PC emulator and target cable to your target system. In most cases, the target system will be a ’C2xx board of your own design.

Figure 1–4. Typical Setup Using the XDS510-PC and Your Target System

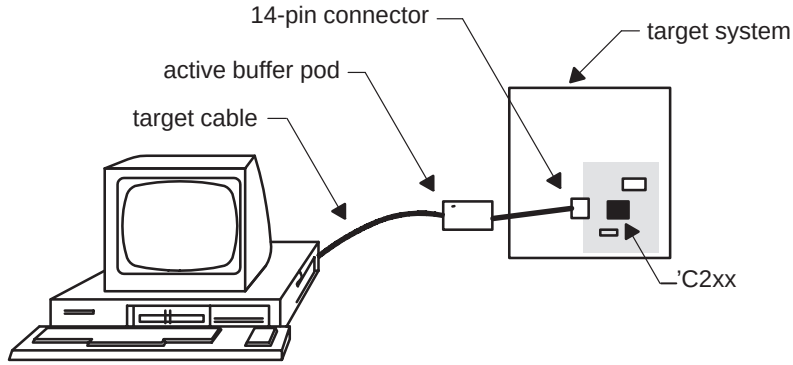
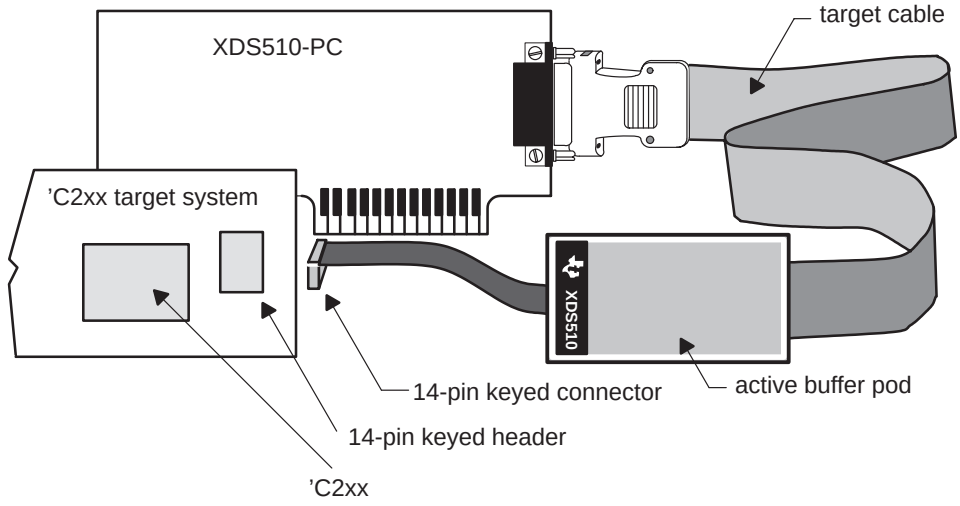


Figure 1–5. Connecting the XDS510-PC to Your Target System



1.4 Step 3: Installing the Debugger Software

This section explains the process of installing the debugger software on a hard-disk system.

1) Make a backup copy of the OS/2 debugger product disk. (If necessary, refer to the OS/2 manual that came with your computer.)

2) On your hard disk or system disk, create a directory named *c2xxhll*. This directory will contain the 'C2xx C source debugger software. To create this directory, enter:

   MD C:\C2XXHLL ↵

3) Insert the OS/2 debugger product disk into drive A. Copy the contents of the disk:

   XCOPY /V /S A:*.* C:\C2XXHLL ↵

   The OS/2 version of the debugger executable is called emu2xxo.exe. If you don't plan to install the MS-Windows version of the debugger software in the c2xxhll directory, you can rename the OS/2 version of the executable to emu2xx.exe.

4) Include one entry for each 'C2xx debugger on your scan path in either your *Start Programs* menu or a *Group* menu. (Refer to your OS/2 manual for instructions on adding a new program to your Start Programs or Group menu.)

1.5 Step 4: Setting Up the Debugger Environment

To ensure that your debugger works correctly, you must:

- ❑ Modify the PATH statement to identify the c2xxhll directory.
- ❑ Define environment variables so that the debugger can find the files it needs.
- ❑ Identify any nondefault I/O space used by the emulator.
- ❑ Set the IOPL option. (This can be done *only* in your config.sys file.)
- ❑ Reset the emulator.

Not only must you do these things before you invoke the debugger for the first time, *you must do them any time you power up or reboot your PC.*

You can accomplish most of these tasks by entering individual OS/2 commands, but it's simpler to put the commands in your config.sys file or a separate batch file.

Figure 1–6 *(a)* shows an example of a config.sys file that contains the suggested modifications (highlighted in bold type). Figure 1–6 *(b)* shows a sample batch file that you could create instead of editing the config.sys file. (For the purpose of discussion, assume that this sample file is named initdb.cmd.) The subsections following the figure explain these modifications.

Figure 1–6. OS/2 Command Setup for the Debugger

(a) Sample config.sys file to use with the debugger and emulator

PATH statement →
Environment variables and I/O space →
Set IOPL to yes →
Reset the emulator →

```
PATH = C:\OS2\SYSTEM;C:\C2XXTOOLS;C:\C2XXHLL
SET  D_DIR=C:\C2XXHLL
SET  D_SRC=C:\C2XXCODE;C:\PROJECT\SOURCE
SET  D_OPTIONS= –P 280
SET  C_DIR=C:\C2XXTOOLS
IOPL = YES
RUN = C:\C2XXHLL\EMURST.EXE
```

(b) Sample initdb.cmd file to use with the debugger and emulator

PATH statement →
Environment variables and I/O space →
Reset the emulator →

```
PATH = %PATH%; C:\C2XXHLL
SET  D_DIR=C:\C2XXHLL
SET  D_SRC=C:\C2XXCODE;C:\PROJECT\SOURCE
SET  D_OPTIONS= –P 280
SET  C_DIR=C:\C2XXTOOLS
EMURST
```

Invoking the new or modified batch file

- If you modify the config.sys file, be sure to invoke it before invoking the debugger for the first time. To invoke this file, reboot your PC.
- If you create an initdb.cmd file, you must invoke it before invoking the debugger for the first time. After that, you'll need to invoke initdb.cmd for each session in which you want to use the debugger. To invoke this file, enter:

  INITDB ↵

Modifying the PATH statement

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

PATH=C:\C2XXHLL;*path2***;***path3***;. . .**

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

- ❑ If you are modifying your config.sys file and it already contains a PATH statement, simply include **;C:\c2xxhll** at the end of the statement, as shown in Figure 1–6 *(a)*.
- ❑ If you are creating an initdb.cmd file, include **%path%;** at the front of your PATH statement, as shown in Figure 1–6 *(b)*.

Note:

Creating an *initdb.cmd* file to modify your path statement has its limitations. The new path statement is active only within the window in which you invoked initdb.cmd. Ideally, your path statement should be set in your config.sys file.

Setting up the environment variables

An environment variable is a special system symbol that the debugger uses for finding or obtaining certain types of information. The debugger uses three environment variables named D_DIR, D_SRC, and D_OPTIONS. The next three bullets tell you how to set up these environment variables. The format for doing this is the same for both the config.sys and initdb.cmd files.

- ❑ Set up the D_DIR environment variable to identify the c2xxhll directory:

  SET D_DIR=C:\C2XXHLL

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

  These directories contain 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 code. The general format for doing this is:

  SET D_SRC=*pathname*$_1$*;pathname*$_2$*...*

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

  For example, if your 'C2xx programs were in a directory named *c2xxcode* on drive C, the D_SRC setup would be:

  SET D_SRC=C:\C2XXCODE

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

  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 1–3 lists the options that you can identify with D_OPTIONS.

Table 1–3. Options for Use With D_OPTIONS

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

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

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

Note:

Setting environment variables in the *initdb.cmd* file has its limitations. The new environment variables are active only within the window in which you invoked initdb.cmd. Ideally, your environment variables should be set in your config.sys file.

Identifying the correct I/O switches

Refer to your entries in Table 1–2 (page 1-5). If you didn't modify the I/O switches, you can skip this step.

If you modified the I/O switch settings, you must use the debugger's –p option to identify the I/O space that the emulator is using. You can do this each time you invoke the debugger, or you can specify this information by using the D_OPTIONS environment variable. Table 1–4 lists the I/O nondefault switch setting and the appropriate line that you can add to the config.sys or initdb.cmd file.

Table 1–4. Identifying Nondefault I/O Address Space

	switch #		Add this line to the
Address Range	**1**	**2**	**batch file**
0x0280–0x029F	on	off	`SET D_OPTIONS=–p 280`
0x0320–0x033F	off	on	`SET D_OPTIONS=–p 320`
0x0340–0x035F	off	off	`SET D_OPTIONS=–p 340`

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.

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 three ways:

- ❑ Add the following line to the end of your config.sys file (as shown in Figure 1–6 *(a)* on page 1-10):

  RUN = C:\C2XXHLL\EMURST.EXE

Note:

If you reset the emulator using the RUN command in your config.sys file (see Figure 1–6 *(a)* on page 1-10.), 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.

- ❑ If you created an initdb.cmd file, add the following line to the end of the file (as shown in Figure 1–6 *(b)* on page 1-10):

  EMURST

- ❑ Create or modify a file called C:\startup.cmd that contains the following line:

  EMURST

Note:

If a debugger is running, emurst will not reset the emulator. The debugger will display 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 XDS510-PC installed snugly?
- ❑ Is the cable connecting your XDS510-PC 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 (refer to *Your Switch Settings*, Table 1–2, and *Identifying Nondefault I/O Address Space*, Table 1–4).
  - 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 in Table 1–1. 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 'C2xx and no other devices, you can use the *board.dat* file that comes with the 'C2xx emulator kit. This file describes to the debugger the single 'C2xx in the scan path and gives the 'C2xx the name CPU_A. Since the debugger automatically looks for a file called board.dat in the current directory and in the directories specified with the D_DIR environment variable, you don't need to create your own board configuration file. Go to the next page.
- ❑ If you plan to use a different target system, you must follow these steps:

  Step 1: Create the board configuration file.

  Step 2: Use the composer utility to translate the board configuration file to binary so that the debugger can read it.

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

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

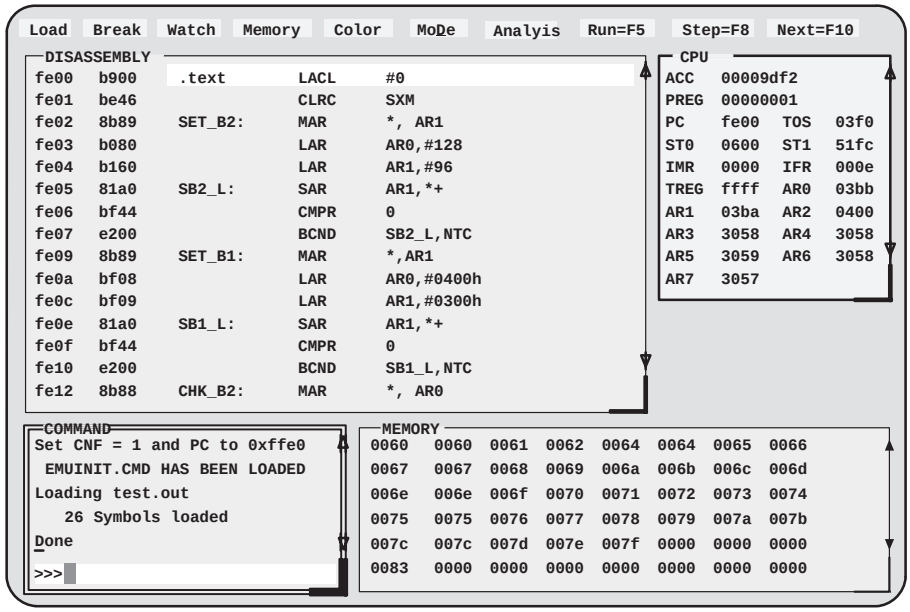
1.7 Step 6: Verifying the Installation

To ensure that you have correctly installed the XDS510-PC emulator and debugger software, enter this command at the system prompt:

EMU2XX C:\C2XXHLL\test –N *DEVICE NAME* ↵

Note that **emu2xx** refers to the emu2xx and emu2xxo executables.

You should see a display similar to this one:



- ❑ If you see a display similar to this one, you have correctly installed your XDS510-PC emulator and debugger.
- ❑ If you see a display and the lines of code show ADD instructions, your XDS510-PC may not be installed snugly. Check your board to see if 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 XDS510-PC installed snugly?
- ❑ Is the device installed snugly?
- ❑ Is the cable connecting your XDS510-PC 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 initdb.cmd file, startup.cmd file, or config.sys file.
- ❑ Is your PATH statement set correctly?
- ❑ 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 (refer to *Your Switch Settings*, Table 1–2, and *Identifying Nondefault I/O Address Space*, Table 1–4).
  - 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 in Table 1–1. Modify the –p option of the D_OPTIONS environment variable to reflect the change in your switch settings.

- ❑ Was the –f option specified on the command line or in the D_OPTIONS environment variable (where the default file specified by the –f option is board.dat) specifying a file that the debugger can't find?
  - If you didn't provide *any* path information with the filename, the debugger couldn't find the file in the current directory or in any of the directories listed in the D_DIR environment variable.
  - If you didn't provide the *correct* path information, re-execute the debugger, specifying the correct pathname and filename for the board configuration file.
- ❑ Is the board.dat file in the current directory or in a directory specified by D_DIR?
- ❑ Did the compose utility successfully create the board.dat file?

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 1.6 (page 1-15) for more information.

After you have checked all of the above, repeat the verification instructions in Section 1.7.

Chapter 2

Installing the Emulator and C Source Debugger With MS-Windows

This chapter helps you with the following:

- ❑ Installing the XDS510-PC (the emulation controller interface hardware) in a PC.
- ❑ Installing the TMS320C2xx C source debugger on a PC running MS-Windows™.

When you complete the installation, turn to the *TMS320C2xx C Source Debugger User's Guide.*

Topic **Page**

2.1 What You'll Need

The following checklists detail items that are shipped with the 'C2xx C source debugger and XDS510-PC emulator and additional items you'll need to use these tools

Hardware checklist

☐	**host**	A 32-bit 386-based PC (or higher) or a Pentium PC with an ISA/EISA bus and a 1.44-megabyte floppy-disk drive
☐	**memory**	Minimum of 8 megabytes
☐	**display**	Monochrome or color (color recommended)
☐	**slot**	One 16-bit slot
☐	**XDS510-PC power requirements**	Approximately 1 ampere @ 5 volts (5 watts)
☐	**target system**	A board with a 'C2xx device
☐	**connector to target system**	14-pin connector (two rows of seven pins)—for more information about this connector, refer to the *JTAG/MPSD Emulation Technical Reference*.
☐	**optional hardware**	A Microsoft-compatible mouse
☐		An EGA- or VGA-compatible graphics display card and a large monitor.
☐	**miscellaneous materials**	Blank, formatted disks

Software checklist

☐	**operating system**	MS-Windows (version 3.1 or later) or MS-Windows for Workgroups (version 3.11 or later)
☐	**software tools**	TMS320C1x/C2x/C2xx/C5x assembler and linker Optional: TMS320C1x/C2x/C2xx/C5x C compiler
☐	**required files** †	*emu2xxwm.exe* is the executable file that invokes the C source debugger for the emulator.
☐	†	*emu2xxdm.dll* is a dynamically linked library (DLL) that the debugger (emu2xxwm.exe) uses. This library is loaded automatically when you invoke the debugger. This file must remain in the same directory as the debugger executable.
☐	†	*smg501w.dll* is a DLL that the debugger uses. This file must remain in the same directory as the debugger executable.

† Included as part of the debugger package

- ☐ † *emu2xxw.exe* is a faster version of emu2xxwm.exe for target boards that include only one 'C2xx on the emulation scan path. This file does not work with the XDS511.
- ☐ † *emu2xxd.dll* is a DLL that the debugger (emu2xxw.exe) uses. This file is for use with target boards that include only one 'C2xx on the emulation scan path. This file must remain in the same directory as the debugger executable. There is no DLL corresponding to smg510w.dll when the target board includes only one 'C2xx on the emulation scan path.
- ☐ † *board.dat* describes your target system to the debugger. This file is required only when you are using emu2xxwm.exe.
- ☐ † *board.cfg* is the default name for the text file that describes your target system to the debugger. If you plan to create a target system that contains anything other than a single 'C2xx, you must make a text file that describes the target system. Once you have created the file, you must translate it to a binary, conditioned format so that the debugger can understand it; this reformatted file is called board.dat.

  Refer to the *Describing Your Target System to the Debugger* appendix in the *TMS320C2xx C Source Debugger User's Guide* for more information.
- ☐ † *composer.exe* is the utility that translates the board.cfg file to a binary, conditioned format. This file is required only when you are using emu2xxwm.exe.
- ☐ † *emurst.exe* resets the 'C2xx emulator

optional files

- ☐ † *emuinit.cmd* is a general-purpose batch file that contains debugger commands. The version of this file that's shipped with the debugger defines a 'C209 memory map. If this file isn't present when you first invoke the debugger, then all memory is invalid at first. You *must* modify this file to match your target system's memory map. For information about setting up your own memory map, refer to the *Defining a Memory Map* chapter in the *TMS320C2xx C Source Debugger User's Guide*.
- ☐ † *init.clr* 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.
- ☐ † *init.25* and *init.50* have been provided for basic 80x25 and 80x50 screen sizes, respectively. The init.clr file brings up the debugger in 80x25 mode. To bring the debugger up in another mode, copy one of the init.xx files to the init.clr file.

† Included as part of the debugger package

☐ † 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, refer to the *Customizing the Debugger Display* chapter in the *TMS320C2xx C Source Debugger User's Guide*.

† Included as part of the debugger package

2.2 Step 1: Installing the XDS510-PC in Your PC

This section contains the hardware installation information for the XDS510-PC emulator.

Preparing the XDS510-PC for installation

Before you install the XDS510-PC, you must be sure that the board's switches are set to correctly identify the I/O space that the board can use.The emulator uses 32 bytes of the PC I/O space; two switches on the board identify this space.

Figure 2–1 shows where these switches are on the XDS510-PC and identifies the switch numbers.

Switches are shipped in the default settings shown here and are listed in Table 2–1. If you use an I/O space that differs from the default, change the switch settings. Table 2–1 shows alternate settings.

In most cases, you can leave the switch settings in the default position. However, you must ensure that the 'C2xx emulator I/O space does not conflict with other bus settings. For example, if you've installed a bus mouse in your system, you may not be able to use the default switch settings for the I/O address space—the mouse might use this space. Refer to your PC technical reference manual and your other hardware-board manuals to see if there are any I/O space conflicts. If you find a conflict, use one of the alternate settings shown in Table 2–1.

Figure 2–1. XDS510-PC I/O Switches

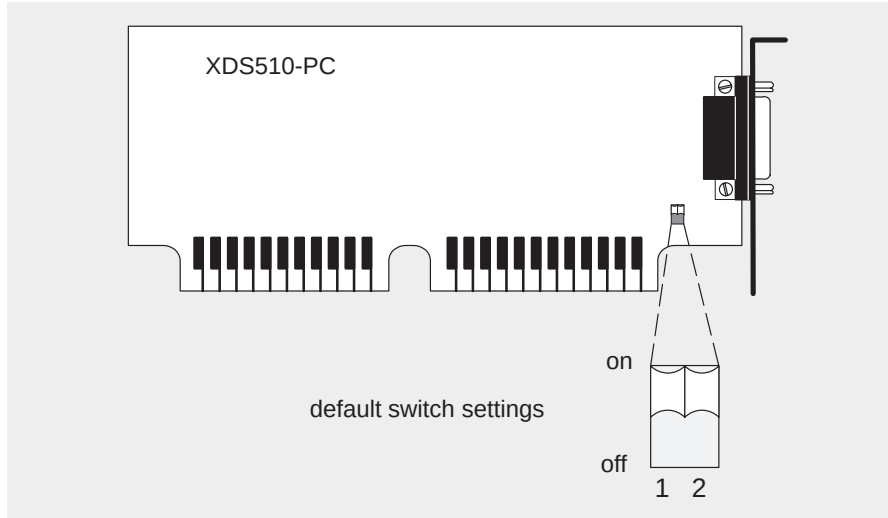


Table 2–1. XDS510-PC Switch Settings

	Switch #	
Address Range	**1**	**2**
0x0240–0x025F	on	on
0x0280–0x029F	on	off
0x0320–0x033F	off	on
0x0340–0x035F	off	off

Some of the other installation steps require you to know which switch settings you used. If you reset the I/O switches, note the modified settings here for later reference.

Table 2–2. Your Switch Settings

	Switch #	
Address Range	**1**	**2**

WARNING

1) **To minimize the risk of electric shock and fire hazard, be sure that all major components that you interface with Texas Instruments devices are limited in energy and certified by one or more of the following agencies: UV, CSA, VDE, or TUV.**
2) **To minimize the risk of personal injury, *always* turn off the power to your PC and unplug the power cord before installing the XDS510-PC.**

Setting the XDS510-PC into your PC

After you've prepared the XDS510-PC for installation, follow these steps.

1) Turn off the power to the PC, and unplug the power cord.
2) Remove the cover of your PC.
3) Remove the mounting bracket from an unused 16-bit slot.
4) Install the XDS510-PC in a 16-bit slot (see Figure 2–2).
5) Tighten down the mounting bracket.
6) Plug the target cable into the XDS510-PC (see Figure 2–3). The cable is a 25-pin DSUB connector, shaped to ensure proper connection.
7) Replace the PC cover.
8) Plug in the power cord, and turn on the power to the PC.

Figure 2–2. XDS510-PC Installation

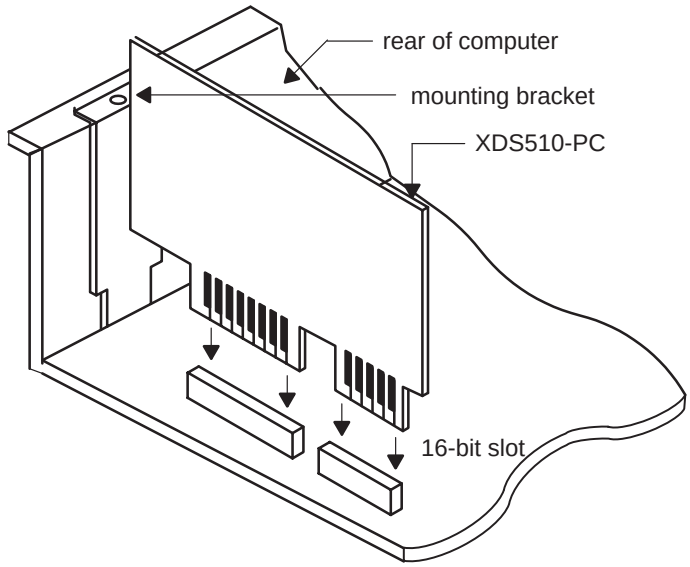
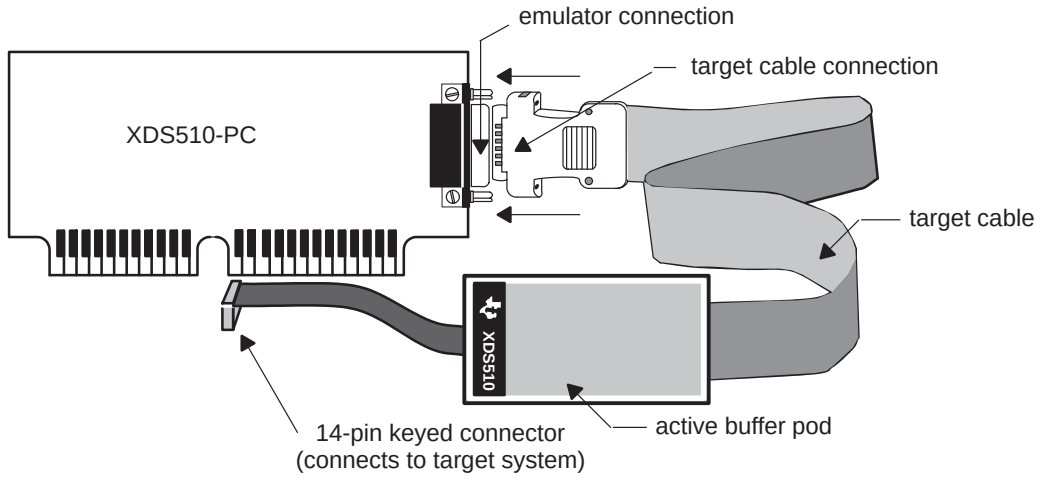


Figure 2–3. Connecting the Target Cable to the XDS510-PC



Don’t connect or disconnect the target cable while the PC is powered up.

Be very careful with the target cable connectors. Connect them gently; forcing the connectors into position may damage them.

Remember, the connector is keyed. Be sure to connect the cable so that the key fits into its slot.

2.3 Step 2: Connecting the XDS510-PC to Your Target System

Figure 2–4 shows a typical setup using the XDS510-PC emulator, target cable, and your target system.

Figure 2–5 shows how you connect the XDS510-PC emulator and target cable to your target system. In most cases, the target system will be a ’C2xx board of your own design.

Figure 2–4. Typical Setup Using the XDS510-PC and Your Target System

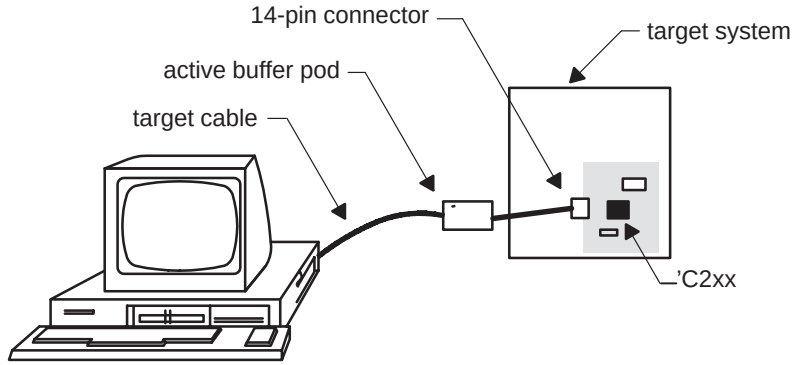
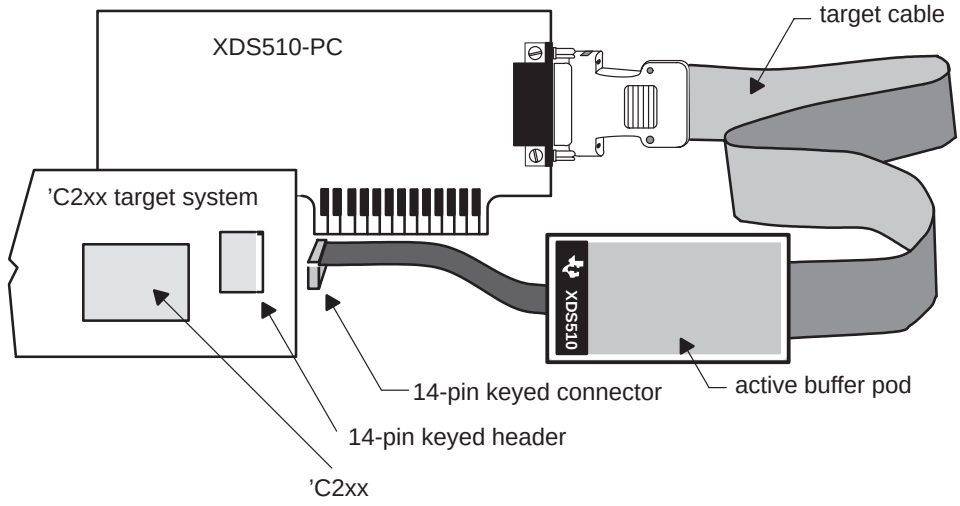


Figure 2–5. Connecting the XDS510-PC to Your Target System



2.4 Step 3: Installing the Debugger Software

This section explains the process of installing the debugger software on a hard-disk system.

1) Make a backup copy of the MS-Windows debugger product disk. (If necessary, refer to the manual that came with your computer.)

2) On your hard disk or system disk, create a directory named *c2xxhll*. This directory will contain the 'C2xx C source debugger software. To create this directory, enter:

   MD C:\C2XXHLL ↵

3) Insert the debugger product disk into drive A. Copy the contents of the disk:

   XCOPY /V /S A:*.* C:\C2XXHLL ↵

   There are two MS-Windows debugger executables: emu2xxwm.exe (for target boards, such as the XDS511, with multiple devices in the emulator scan path) and emu2xxw.exe (for target boards with only one 'C2xx in the emulation scan path). Throughout this document, the executable for the debugger is referred to as simply emu2xx.

4) You may want to create an icon to make it easier to invoke the debugger from within the MS-Windows environment. To install the debugger in MS-Windows, create a new program item (refer to your MS-Windows manual for details). While creating a new program item, type **emu2xxwm.exe** or **emu2xxw.exe** as the command line and include any additional parameters you want.

Notes:

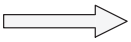
1) When using MS-Windows, you can freely move or resize the debugger display on the screen. If the resized display is bigger than the debugger requires, the extra space is not used, and the display reverts back to the original size. If the resized display is smaller than required, the display is clipped. Note that when the display is clipped, it can't be scrolled.

2) You should run MS-Windows in either the standard mode or the 386 enhanced mode to get the best results.

2.5 Step 4: Setting Up the Debugger Environment

To ensure that your debugger works correctly, you must:

- ❑ Modify the PATH statement to identify the c2xxhll directory.
- ❑ Define environment variables so that the debugger can find the files it needs.
- ❑ Identify any nondefault I/O space used by the emulator.
- ❑ Reset the emulator.



Not only must you do these things before you invoke the debugger for the first time, *you must do them any time you power up or reboot your PC.*

You can accomplish these tasks by entering individual commands, but it's simpler to put the commands in a batch file. You can edit your systems autoexec.bat file; in some cases, modifying the autoexec may interfere with other applications running on your PC. So, if you prefer, you can create a separate batch file that performs these tasks.

Figure 2–6 *(a)* shows an example of an autoexec.bat file that contains the suggested modifications (highlighted in bold type). Figure 2–6 *(b)* shows a sample batch file that you could create instead of editing the autoexec.bat file (for the purpose of discussion, assume that this sample file is named *initdb.bat*). The subsections following the figure explain these modifications.

Figure 2–6. Command Setup for the Debugger

(a) Sample autoexec.bat file to use with the debugger and emulator

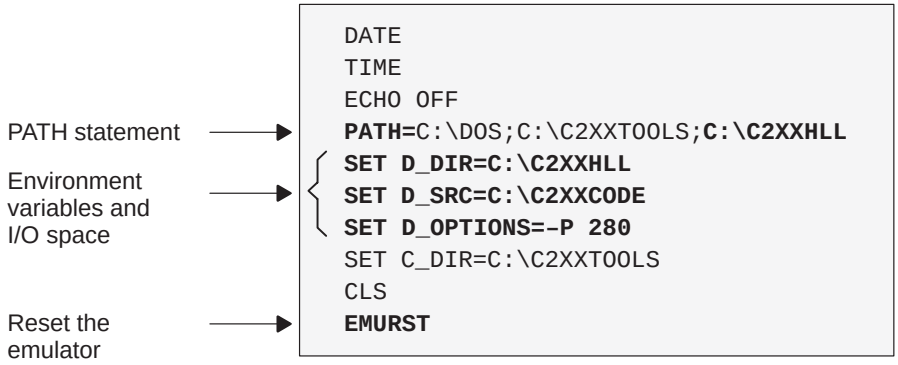
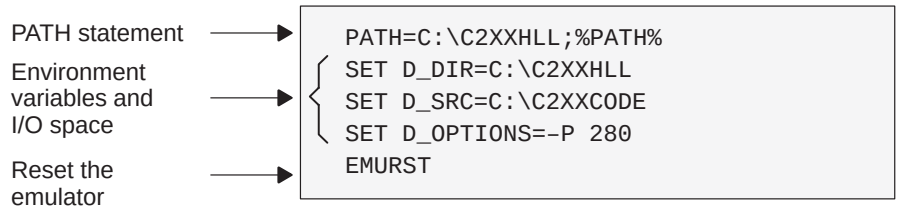


Figure 2–6. Command Setup for the Debugger (Continued)

(b) Sample initdb.bat file to use with the debugger and emulator



Invoking the new or modified batch file

- If you modify the autoexec.bat file, be sure to invoke it before invoking the debugger for the first time. To invoke this file, enter:

  AUTOEXEC ↵

- If you create an initdb.bat file, you must invoke it before invoking the debugger for the first time. If you are using MS-Windows, invoke initdb.bat *before* entering MS-Windows. You'll need to invoke initdb.bat any time that you power up or reboot your PC. To invoke this file, enter:

  INITDB ↵

Modifying the PATH statement

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

```
PATH=C:\C2XXHLL
```

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

- If you are modifying an autoexec that already contains a PATH statement, simply include **;C:\c2xxhll** at the end of the statement, as shown in Figure 2–6 *(a)*.

- If you are creating an initdb.bat file, use a different format for the PATH statement, as shown in Figure 2–6 *(b)*:

  ```
  PATH=C:\C2XXHLL;%PATH%
  ```

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

Setting up the environment variables

An environment variable is a special system symbol that the debugger uses for finding or obtaining certain types of information. The debugger uses three environment variables named D_DIR, D_SRC, and D_OPTIONS. The next three bullets tell you how to set up these environment variables. The format for doing this is the same for both the autoexec.bat and initdb.bat files.

- Set up the D_DIR environment variable to identify the c2xxhll directory:

  SET D_DIR=C:\C2XXHLL

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

  This directory contains auxiliary files (emurst, emuinit.cmd, etc.) that the debugger needs.

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

  SET D_SRC=*pathname*$_1$*;pathname*$_2$*...*

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

  For example, if your 'C2xx programs were in a directory named *c2xxcode* on drive C, the D_SRC setup would be:

  SET D_SRC=C:\C2XXCODE

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

  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 2–3 lists the options that you can identify with D_OPTIONS.

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

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

Table 2–3. Options for Use With D_OPTIONS

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

Identifying the correct I/O switches

Refer to your entries in Table 2–2 (page 2-5). If you didn't modify the I/O switches, skip this step.

If you modified the I/O switch settings, you must use the debugger's –p option to identify the I/O space that the emulator is using. You can do this each time you invoke the debugger, or you can specify this information by using the D_OPTIONS environment variable. Table 2–4 lists the nondefault I/O switch setting and the appropriate line that you can add to the autoexec.bat or initdb.bat file.

Table 2–4. Identifying Nondefault I/O Address Space

	switch #		Add this line to the
Address Range	1	2	batch file
0x0280–0x029F	on	off	`SET D_OPTIONS=–p 280`
0x0320–0x033F	off	on	`SET D_OPTIONS=–p 320`
0x0340–0x035F	off	off	`SET D_OPTIONS=–p 340`

Resetting the emulator

To reset the emulator, add this line to the autoexec.bat or initdb.bat file:

```
emurst
```

Note:

The emurst program resets the ’C2xx emulator only; it doesn’t apply reset to the ’C2xx device on your target board. If you cycle the power to your target board, you **must** execute emurst before you invoke the debugger.

2.6 Step 5: Describing Your Target System to the Debugger

If you are using the emu2xxwm.exe executable, the debugger supports the presence of multiple devices on the emulation scan path. 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 ’C2xx and no other devices, you can use the *board.dat* file that comes with the ’C2xx emulator kit. This file describes to the debugger the single ’C2xx in the scan path and gives the ’C2xx the name CPU_A. Since the debugger automatically looks for a file called board.dat in the current directory and in the directories specified with the D_DIR environment variable, you don’t need to create your own board configuration file. Go to the next page.

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

  Step 1: Create the board configuration file.

  Step 2: Use the composer utility to translate the board configuration file to binary so that the debugger can read it.

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

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

2.7 Step 6: Verifying the Emulator Device Driver Installation

The emulator device driver enables the debugger to communicate with the emulator. To ensure that the emulator driver is installed correctly, follow these steps:

1) Turn off the power to the ’C2xx target board.

2) Connect the JTAG cable from your PC to the ’C2xx target board.

3) Turn on the power to the ’C2xx target board.

4) If a hard RESET is not generated automatically after power-up, apply a hard RESET to the ’C2xx device.

5) From the command prompt in a MS-DOS™ window, enter:

   emurst ↵

You should see a message similar to the following:

```
XDS510 IS RESET, HARDWARE VERSION 3
```

- ❑ If this message appears, you have correctly installed the emulator driver. Turn to Section 2.8, *Step 7: Verifying the Emulator and Debugger Installation*.

- ❑ If you see this message:

  ```
  CANNOT DETECT TARGET POWER
  ```

  follow the troubleshooting tips in the following section.

Troubleshooting

- ❑ The target power is not on or the cables are not connected.

  Be sure that power is applied to the target board and that the JTAG cable is connected firmly.

- ❑ The I/O switch settings on the XDS510-PC don’t match the value specified with the –p option.

  If you are using nondefault I/O switch settings, you must specify the corresponding –p option in either the D_OPTIONS environment variable or on the command line when you invoke the debugger.

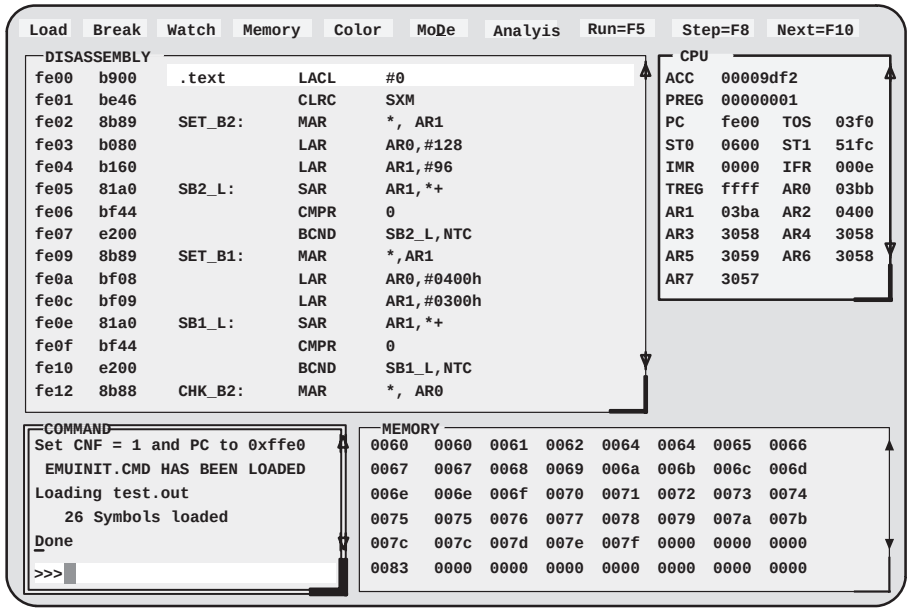
2.8 Step 7: Verifying the Debugger and Emulator Installation

To ensure that you have correctly installed the XDS510-PC emulator and debugger software, enter this command from the MS-Windows program manager:

EMU2XX C:\C2XXHLL\TEST –N CPU_A ↵

Note that **emu2xx** refers to the emu2xx, emu2xxw, and emu2xxwm executables.

You should see a display similar to this one:



- ❑ If you see a display similar to this one, you have correctly installed your XDS510-PC emulator and debugger.
- ❑ If you see a display and the lines of code show ADD instructions, your XDS510-PC may not be installed snugly. Check your board to see if 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 THE 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 XDS510-PC installed snugly?
- ❑ Is the device installed snugly?
- ❑ Is the cable connecting your XDS510-PC 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 if you are using emu2xxwm.exe.
- ❑ Does the emurst command appear at the end of either your autoexec.bat or initdb.bat file? This command must be executed *after* you powered up the target board.
- ❑ Is your PATH statement set correctly?
- ❑ 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 (refer to *Your Switch Settings*, Table 2–2, and *Identifying Nondefault I/O Address Space*, Table 2–4).
  - 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 in Table 2–1. Modify the –p option of the D_OPTIONS environment variable to reflect the change in your switch settings.

- ❑ Was the –f option specified on the command line or in the D_OPTIONS environment variable (where the default file specified by the –f option is board.dat) specifying a file that the debugger can't find?
  - ■ If you didn't provide *any* path information with the filename, the debugger couldn't find the file in the current directory or in any of the directories listed in the D_DIR environment variable.
  - ■ If you didn't provide the *correct* path information, re-execute the debugger, specifying the correct pathname and filename for the board configuration file.
- ❑ Is the board.dat file in the current directory or in a directory specified by D_DIR?
- ❑ Did the compose utility successfully create the board.dat file?

After you have checked all of the above, repeat the verification instructions in Section 2.8.

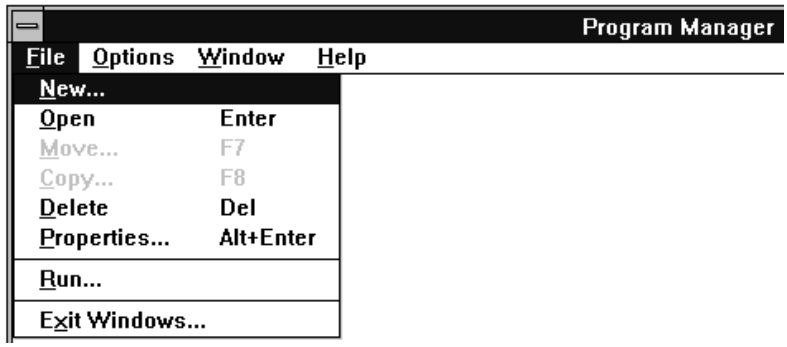
2.9 Setting Up Icons for the Windows Program Manager

There are two ways to invoke the debugger. You are already familiar with the first method; you can invoke the debugger from the MS-Windows program manager:

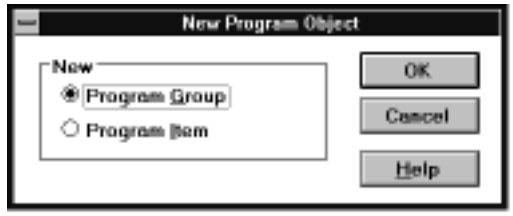
```
emu2xx –n cpu_a ↵
```

An easier way to invoke the debugger is to establish a program group window that contains an icon for the 'C2xx debugger. This allows you to invoke a debugger by simply double-clicking on its icon.

Step 1: From the Windows Program Manager, select File → New.

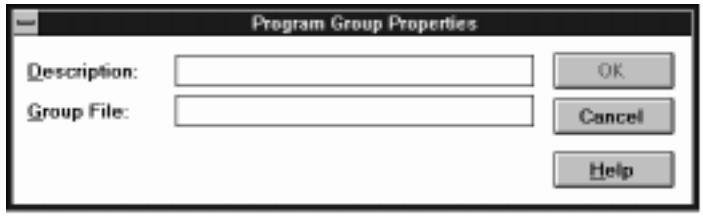


This displays the New Program Object dialog box:



Step 2: Select Program Group, then click on the OK button.

This displays a dialog box for you to enter the group name:



Step 3: Enter *TMS320C2xx Debugger* as the Description, then click on the OK button.

This displays an empty program group named TMS320C2xx Debugger.

Now you can assign program items (debugger icons) to the TMS320C2xx Debugger program group. Be sure that the TMS320C2xx Debugger program group is active. If it is not highlighted (active), move your mouse into its window and click.

Step 4: While the TMS320C2xx Debugger program group is active, from the Windows Program Manager, select File → New.

This displays the New Program Object dialog box.

Step 5: Select the Program Item option button, then click on the OK button.

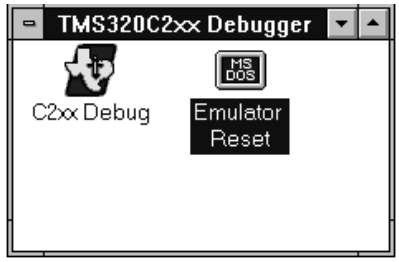
This displays the Program Item Properties dialog box. Enter the following information for the 'C2xx debugger:

After entering this information, click on the OK button. This displays a Texas Instruments icon in your TMS320C2xx Debugger program group.

Step 6: You can also put emurst into the TMS320C2xx Debugger program group. The program item descriptions and command lines is:

Description	Command Line
EMURST	c:\c2xxhll\emurst –p 240

After entering the emurst program item, the TMS320C2xx Debugger program group should look like the following:



Step 7: Invoke the debugger by double-clicking on its icon.

Chapter 3

Release Notes

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

Topic **Page**

3.1 Differences Between the ’C5x and ’C2xx Debuggers

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

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

3.2 New Features of the ’C2xx Debugger

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

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

Usage of arrow keys in the COMMAND window

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

Notes:

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

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

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

Minimal debugging mode

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

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

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

The running indicator

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