TEXAS INSTRUMENTS

TMS320C5x Emulator

Installation Guide

1993 Microprocessor Development Systems

TEXAS INSTRUMENTS

Printed in U.S.A., April 1993
SDS

SPRU125B

TMS320C5x Emulator Installation Guide

April 1993

IMPORTANT NOTICE

Texas Instruments Incorporated (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 current specifications 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.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

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.

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

Copyright © 1993, Texas Instruments Incorporated

Contents

Chapter 1

Installing the Emulator and C Source Debugger With OS/2

This chapter helps you to install the ’C5x emulator board and the C source debugger on a PC system running OS/2. When you complete the installation, turn to the *TMS320C5x C Source Debugger User’s Guide.*

Topic **Page**

1.1 What You'll Need

The following checklists detail items that are shipped with the 'C5x C source debugger and emulator and additional items you'll need to use these tools.

Hardware checklist

☐	**host**	An IBM PC/AT or 100% compatible ISA/EISA-based PC with a hard-disk system and a 1.2-megabyte floppy-disk drive
☐	**memory**	Minimum of 8 megabytes
☐	**display**	Monochrome or color (color recommended)
☐	**slot**	One 16-bit slot
☐	**emulator board power requirements**	Approximately 1 ampere @ 5 volts (5 watts)
☐	**target system**	A board with at least one 'C5x on the emulator scan path
☐	**connector to target system**	14-pin connector (two rows of seven pins)—see Chapter 3 of this book for more information about this connector
☐	**optional hardware**	A Microsoft-compatible mouse
☐		An EGA- or VGA-compatible graphics display card and a large monitor. The debugger has two options that allow you to change the overall size of the debugger display. If you have an EGA- or VGA-compatible graphics card, you can take advantage of some of these larger screen sizes. These larger screen sizes are most effective when used with a large (17" or 19") monitor. (To use a larger screen size, you must invoke the debugger with the appropriate option. For more information about options, refer to the invocation section in Chapter 1, *Overview of a Code Development and Debugging System*, in the *TMS320C5x C Source Debugger User's Guide*.)
☐	**miscellaneous materials**	Blank, formatted disks

Notes:

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)
☐	**software tools**	TMS320 fixed-point family DSP ('C1x/'C2x/'C5x) assembler and linker and TMS320C2x/C5x C compiler
☐	**required files** †	*emurst.exe* resets the 'C5x emulator
☐	†	*board.dat* describes your target system to the debugger
☐	**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 'C5x memory map. If this file isn't present when you first invoke the debugger, then all memory is invalid at first. 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, refer to Chapter 5, *Defining a Memory Map*, in the *TMS320C5x C Source Debugger User's Guide*.
☐	†	*board.cfg* is the default name for the text file that describes your target system to the debugger. If you plan to create target system that contains anything other than a single 'C5x, 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 *Parallel Debug Manager Addendum to the TMS320C4x and TMS320C5x C Source Debugger User's Guides* for more information about creating and translating a board configuration file.
☐	†	*composer.exe* is the utility that translates the board.cfg file to a binary, conditioned format.
☐	†	*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.
☐	†	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 Chapter 9, *Customizing the Debugger Display*, in the *TMS320C5x C Source Debugger User's Guide*.

† Included as part of the debugger package

1.2 Step 1: Installing the Emulator Board in Your PC

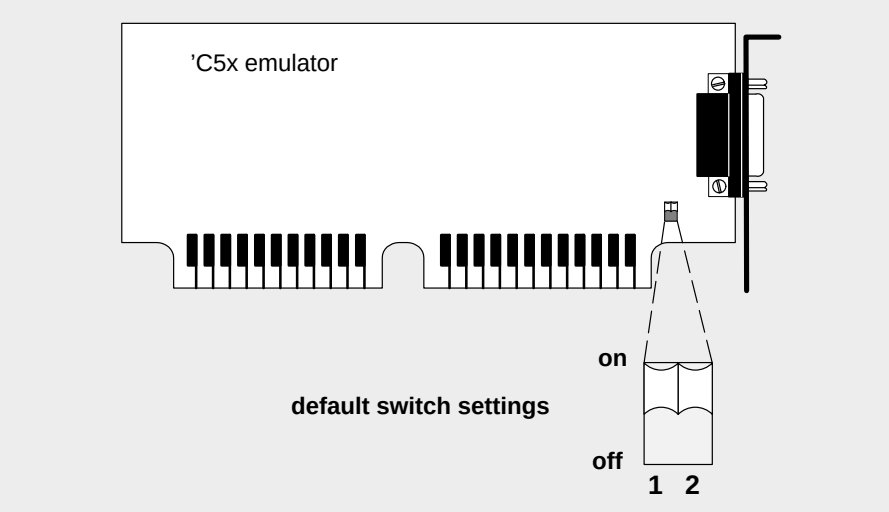
This section contains the hardware installation information for the emulator.

Preparing the emulator board for installation

Before you install the emulator board, 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 emulator board and identifies the switch numbers.

Figure 1–1. Emulator Board I/O Switches



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 'C5x 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.

Table 1–1. Emulator Board Switch Settings

	Address Range	switch # 1	2
default	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	2
⇨		

Setting the emulator board into your PC

After you've prepared the emulator board for installation, follow these steps.

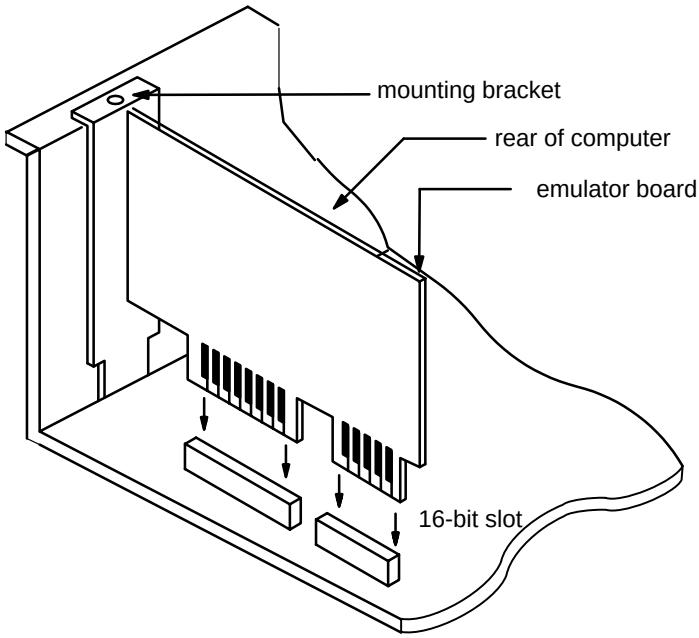
Step 1: Turn off your PC's power. Leave the power cord plugged in so that the computer is properly grounded.

Step 2: Remove the cover of your PC.

Step 3: Remove the mounting bracket from an unused 16-bit slot.

Step 4: Install the emulator board in a 16-bit slot (see Figure 1–2).

Figure 1–2. Emulator Installation



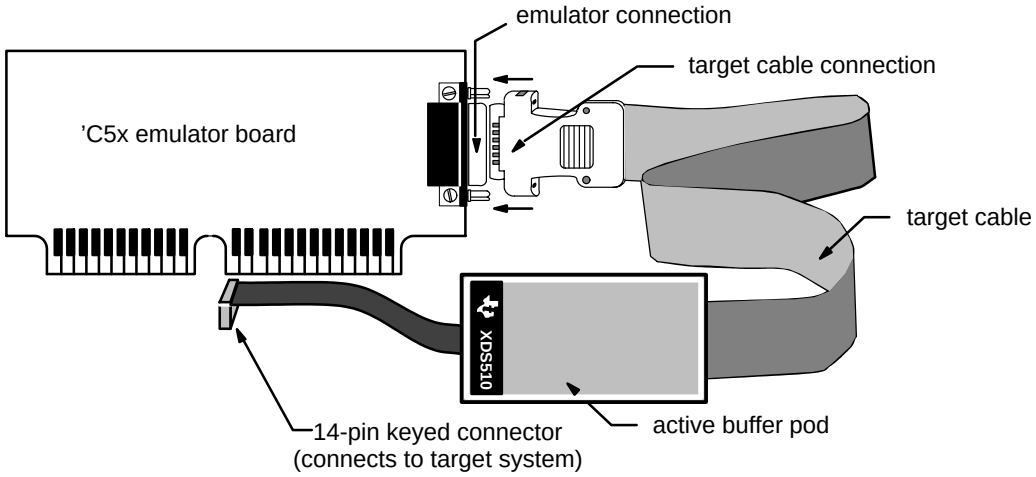
Step 5: Tighten down the mounting bracket.

Step 6: Plug the emulator target cable into the emulator board (see Figure 1–3). The cable is a 25-pin DSUB connector, shaped to ensure proper connection.

Step 7: Replace the PC cover.

Step 8: Turn on the PC's power.

Figure 1–3. Connecting the Target Cable to the 'C5x Emulator Board



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 Emulator to Your Target System

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

Figure 1–4. Typical Setup Using the ’C5x Emulator and Your Target System

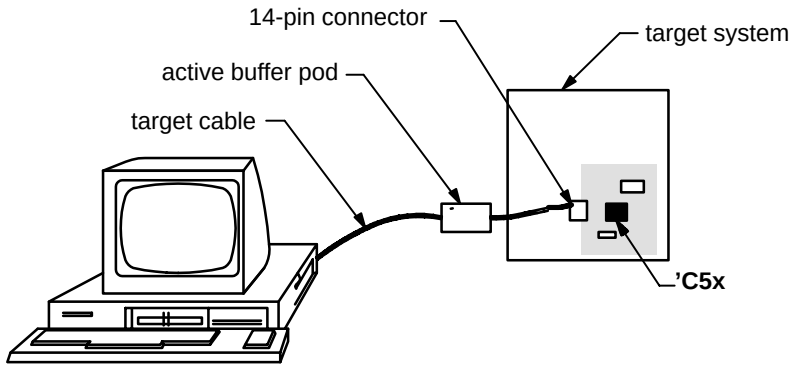
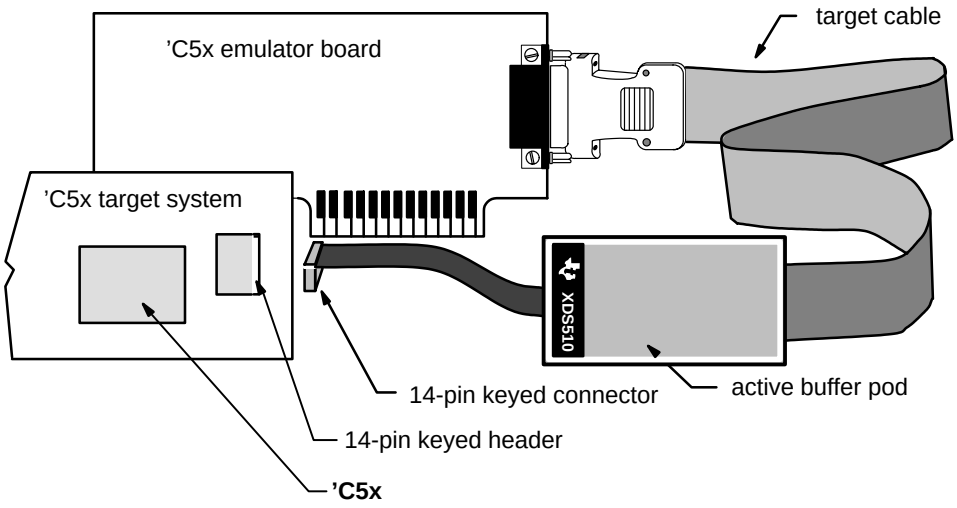


Figure 1–5 shows how you connect the emulator and target cable to your target system. In most cases, the target system will be a ’C5x board of your own design.

Figure 1–5. Connecting the ’C5x Emulator 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 *c5xhll*. This directory will contain the 'C5x C source debugger software. To create this directory, enter:

   MD C:\C5XHLL ↵

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

   COPY A:*.* C:C5XHLL /V ↵

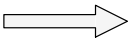
   The OS/2 version of the debugger executable is called emu5xo.exe. If you don't plan to install the DOS or Microsoft Windows version of the debugger software in the c5xhll directory, you can rename the OS/2 version of the executable to emu5x.exe.

4) Include one entry for each 'C5x 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 c5xhll 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 board.



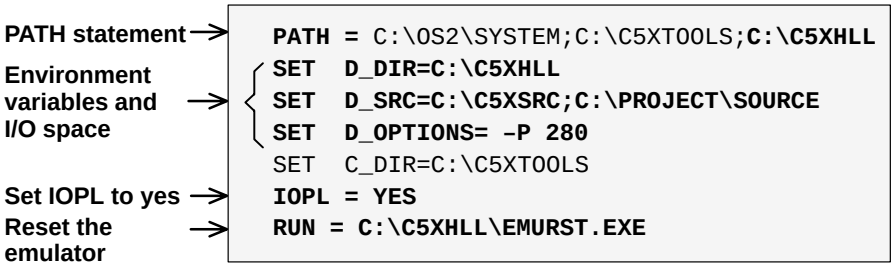
Not only must you do these things before you invoke the debugger for the first time, *you must do them any time you power up or reboot your PC.*

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.

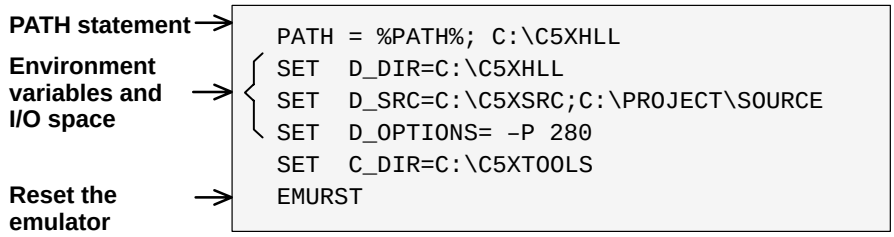
Figure 1–6 *(a)* shows an example of a config.sys file that contains the suggested modifications (highlighted in bold type). Figure 1–6 *(b)* shows a sample batch file that you could create instead of editing the config.sys file. (For the purpose of discussion, assume that this sample file is named initdb.cmd.)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



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



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:\C5XHLL;*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:\c5xhll 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 steps 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 c5xhll directory:

  SET D_DIR=C:\C5XHLL

  (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 'C5x programs were in a directory named *csource* on drive C, the D_SRC setup would be:

  SET D_SRC=C:\CSOURCE

- 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 selected options each time you invoke the debugger. These are the options that you can identify with D_OPTIONS:

–b	–bb	–f *filename*	–i *pathname*
–*n device name*	–p *port address*	–profile	–s
–t *filename*	–v		

  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 Chapter 1, *Overview of a Code Development and Debugging System*, in the *TMS320C5x 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 1–3 lists the I/O nondefault switch setting and the appropriate line that you can add to the config.sys or initdb.cmd file.

Table 1–3. 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:\C5XHLL\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 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 (refer to *Your Switch Settings*, Table 1–2, and *Identifying Nondefault I/O Address Space*, Table 1–3).
  - ■ 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 '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 C50_1. 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: Translate the board configuration file to binary so that the debugger can read it.

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

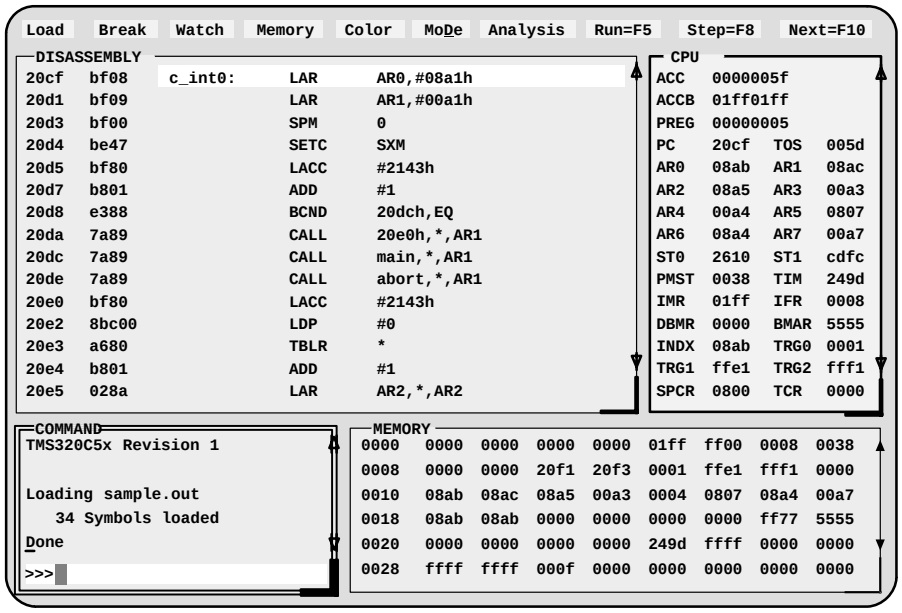
  These steps are described in Section 1.1 of the *PDM Addendum to the TMS320C4x and TMS320C5x C Source Debugger User's Guides*, literature number SPRU094.

1.7 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 c:\C5xhll\sample –n *device name* ↵

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 if it is correctly installed, and re-enter 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. Re-enter 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 re-enter 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 initdb.cmd file, startup.cmd file, or config.sys file.
- ❑ 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 1–3).
  - 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.
- ❑ 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 DOS

This chapter helps you install the ’C5x emulator board and the C source debugger on a PC running MS-DOS or PC-DOS. You can also use the debugger with MS-Windows. When you complete the installation, turn to the *TMS320C5x C Source Debugger User’s Guide*.

Topic **Page**

2.1 What You'll Need

The following checklists detail items that are shipped with the 'C5x C source debugger and emulator and additional items you'll need to use these tools.

Hardware checklist

☐	**host**	An IBM PC/AT or 100% compatible ISA/EISA-based PC with a hard-disk system and a 1.2-megabyte floppy-disk drive
☐	**memory**	Minimum of 4 megabytes
☐	**display**	Monochrome or color (color recommended)
☐	**slot**	One 16-bit slot
☐	**emulator board power requirements**	Approximately 1 ampere @ 5 volts (5 watts)
☐	**target system**	A board with a 'C5x
☐	**connector to target system**	14-pin connector (two rows of seven pins)—see Chapter 3 of this book for more information about this connector
☐	**optional hardware**	A Microsoft-compatible mouse
☐		An EGA- or VGA-compatible graphics display card and a large monitor. The debugger has two options that allow you to change the overall size of the debugger display. If you have an EGA- or VGA-compatible graphics card, you can take advantage of some of these larger screen sizes. These larger screen sizes are most effective when used with a large (17" or 19") monitor. (To use a larger screen size, you must invoke the debugger with an appropriate option. For more information about options, refer to the invocation section in Chapter 1, *Overview of a Code Development and Debugging System*, in the *TMS320C5x C Source Debugger User's Guide*.)
☐	**miscellaneous materials**	Blank, formatted disks

Software checklist

☐	**operating system**	MS-DOS or PC-DOS (version 3.0 or later) Optional: Microsoft Windows (version 3.0 or later)
☐	**software tools**	TMS320 fixed-point family DSP ('C1x/'C2x/'C5x) assembler and linker and TMS320C2x/C5x C compiler
☐	**required file** †	*emurst.exe* resets the 'C5x 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 'C5x memory map. If this file isn't present when you first invoke the debugger, then all memory is invalid at first. 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, refer to Chapter 5, *Defining a Memory Map*, in the *TMS320C5x 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.
☐	†	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 Chapter 9, *Customizing the Debugger Display*, in the *TMS320C5x C Source Debugger User's Guide*.

† Included as part of the debugger package

2.2 Step 1: Installing the Emulator Board in Your PC

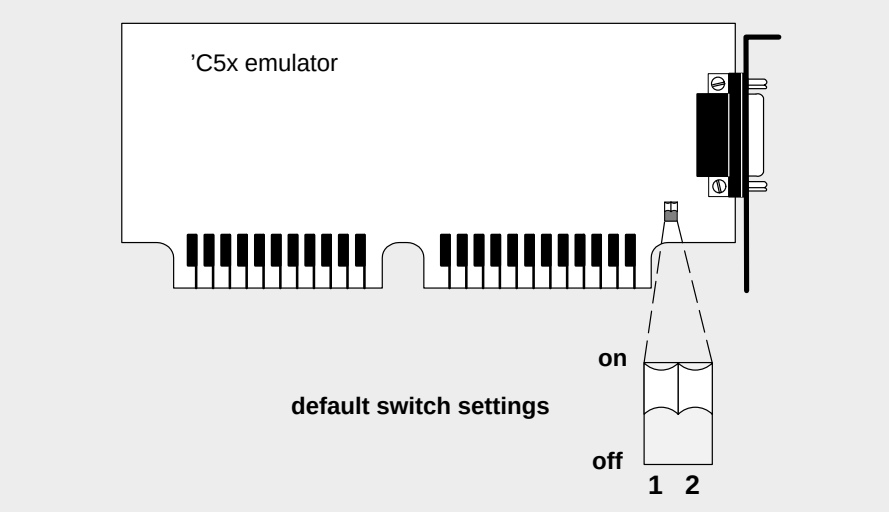
This section contains the hardware installation information for the emulator.

Preparing the emulator board for installation

Before you install the emulator board, 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 emulator and identifies the switch numbers.

Figure 2–1. Emulator Board I/O Switches



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 'C5x 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 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.

Table 2–1. Emulator Board Switch Settings

	Address Range	switch # 1	switch # 2
default	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

Address Range	Switch # 1	Switch # 2

Setting the emulator board into your PC

After you've prepared the emulator board for installation, follow these steps.

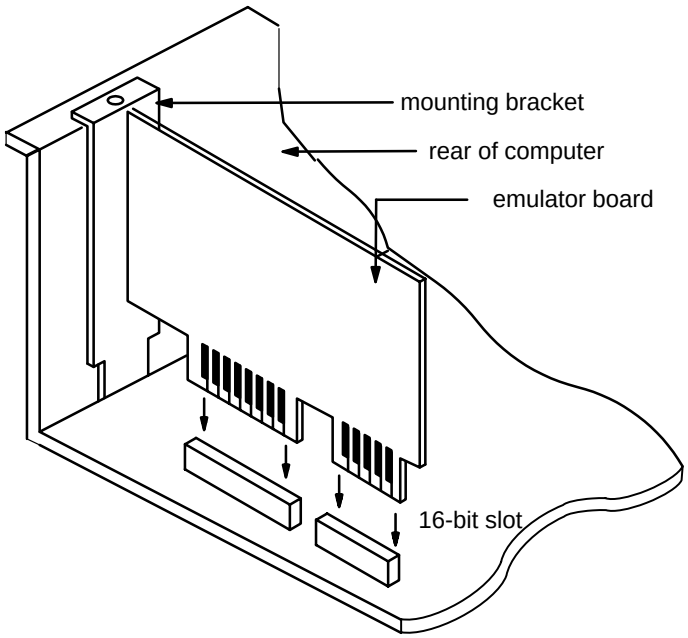
Step 1: Turn off your PC's power. Leave the power cord plugged in so that the computer is properly grounded.

Step 2: Remove the cover of your PC.

Step 3: Remove the mounting bracket from an unused 16-bit slot.

Step 4: Install the emulator board in a 16-bit slot (see Figure 2–2).

Figure 2–2. Emulator Board Installation



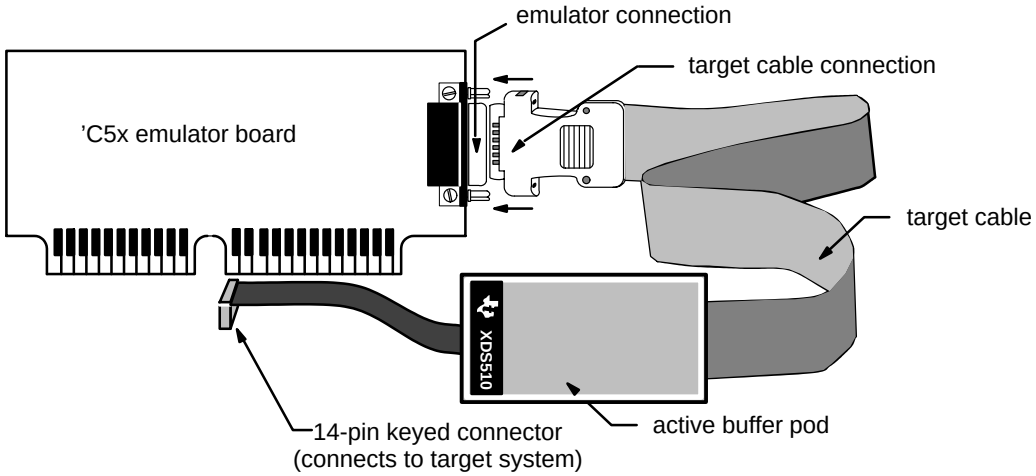
Step 5: Tighten down the mounting bracket.

Step 6: Plug the emulator target cable into the emulator board (see Figure 2–3). The cable is a 25-pin DSUB connector, shaped to ensure proper connection.

Step 7: Replace the PC cover.

Step 8: Turn on the PC's power.

Figure 2–3. Emulator Target Cable and Board



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 Emulator to Your Target System

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

Figure 2–4. Typical Setup Using the ’C5x Emulator and Your Target System

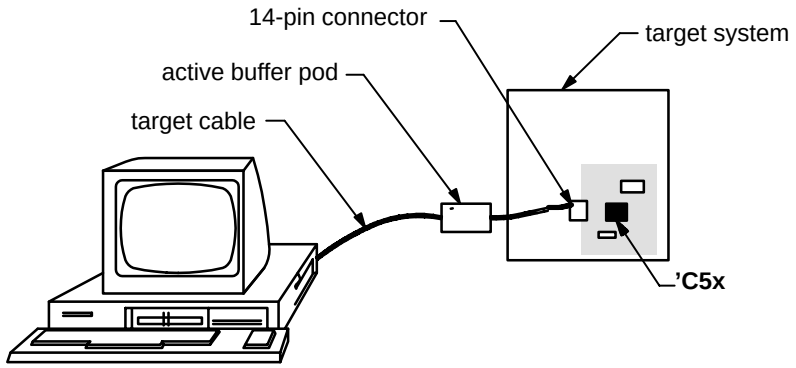
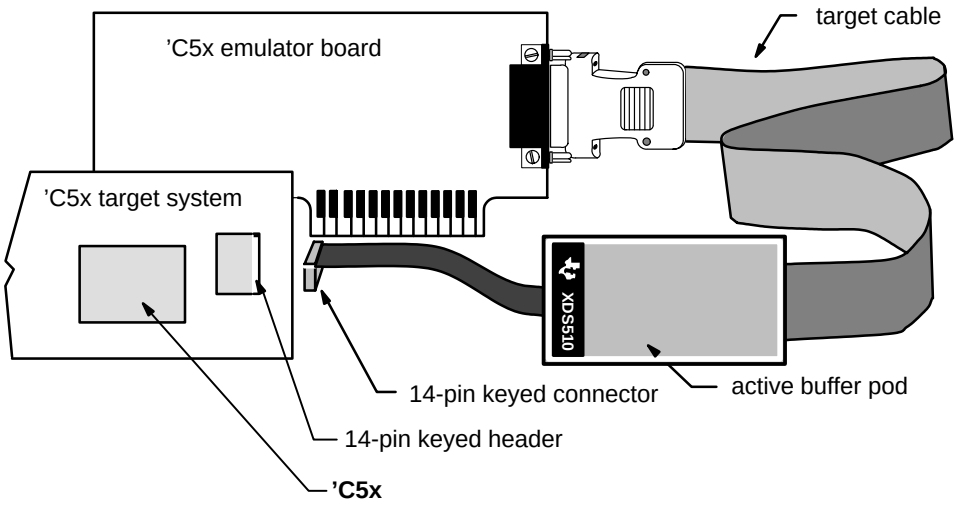


Figure 2–5 shows how you connect the emulator and target cable to your target system. In most cases, the target system will be a ’C5x board of your own design.

Figure 2–5. Connecting the ’C5x Emulator 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 the DOS and/or Microsoft Windows debugger product disk. (If necessary, refer to the DOS manual that came with your computer.)

2) On your hard disk or system disk, create a directory named *c5xhll*. This directory will contain the 'C5x C source debugger software. To create this directory, enter:

   MD C:\C5XHLL ↵

3) Insert either the DOS or Microsoft Windows debugger product disk into drive A. Copy the contents of the disk:

   COPY A:*.* C:\C5XHLL*.* /V ↵

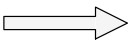
   Repeat this step for the other product disk if you want to be able to run both the DOS and Microsoft Windows versions of the debugger.

   The DOS version of the debugger executable is called emu5x.exe, and the Microsoft Windows version of the debugger executable is called emu5xw.exe. Throughout this document, the executable for the debugger is referred to as simply emu5x.

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 c5xhll 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 board.



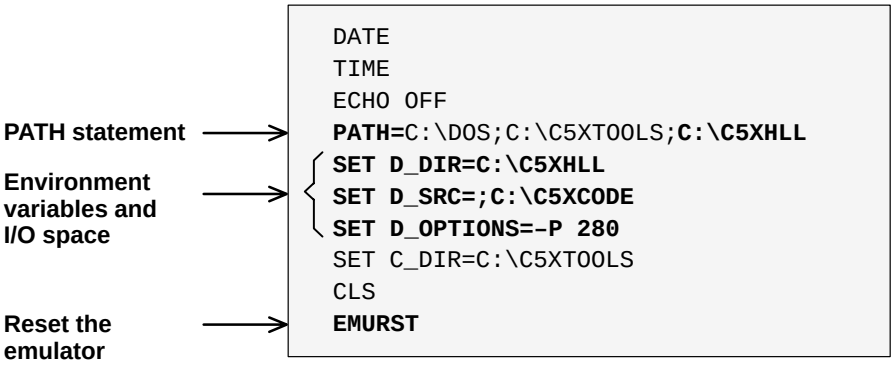
Not only must you do these things before you invoke the debugger for the first time, *you must do them any time you power up or reboot your PC.*

You can accomplish these tasks by entering individual DOS 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.

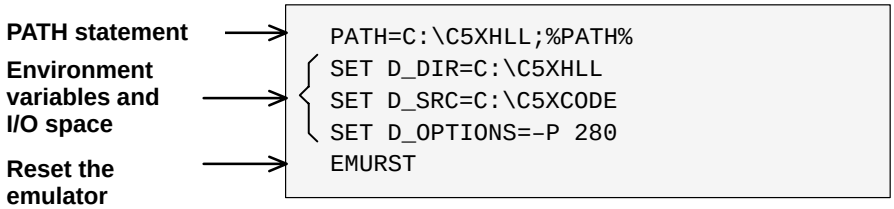
Figure 2–6 *(a)* shows an example of an autoexec.bat file that contains the suggested modifications (highlighted in bold type). Figure 2–6 *(b)* shows a sample batch file that you could create instead of editing the autoexec.bat file (for the purpose of discussion, assume that this sample file is named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

Figure 2–6. DOS-Command Setup for the Debugger

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



(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 Microsoft Windows, invoke initdb.bat *before* entering Microsoft 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:\C5XHLL
```

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:\c5xhll 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:\C5XHLL;%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 steps 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 c5xhll directory:

  ```
  SET D_DIR=C:\C5XHLL
  ```

  (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 'C5x programs were in a directory named *csource* on drive C, the D_SRC setup would be:

  ```
  SET D_SRC=C:\CSOURCE
  ```

- 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. These are the options that you can identify with D_OPTIONS:

–b	–bb	–i *pathname*	–p *port address*
–profile	–s	–t *filename*	–v

  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 Chapter 1, *Overview of a Code Development and Debugging System*, in the *TMS320C5x C Source Debugger User's Guide*.

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 2–3 lists the nondefault I/O switch setting and the appropriate line that you can add to the autoexec.bat or initdb.bat file.

Table 2–3. 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:

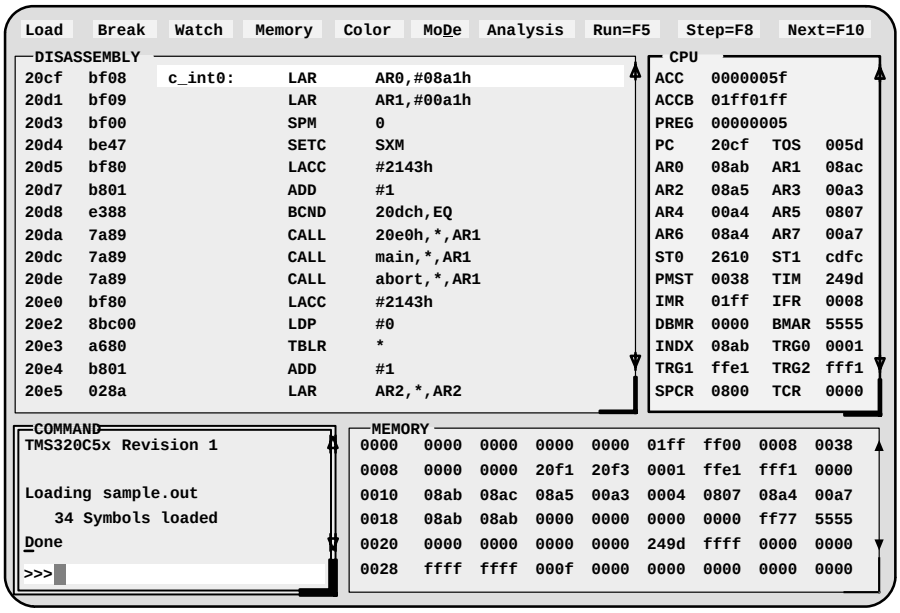
```
emurst
```

2.6 Step 5: Verifying the Installation

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

emu5x c:\c5xhll\sample ↵

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 if it is correctly installed, and re-enter 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. Re-enter 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 re-enter 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 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 port address set correctly:
- ❑ 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.
  - ■ 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 2–3).
  - ■ 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.

After you have checked all of the above, repeat the verification instructions in Section 2.6.

2.7 Using the Debugger With Microsoft Windows

If you're using Microsoft 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. 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 Microsoft Windows in either the standard mode or the 386 enhanced mode to get the best results.

Chapter 3

Specification for Your Target System's Connection to the Emulator

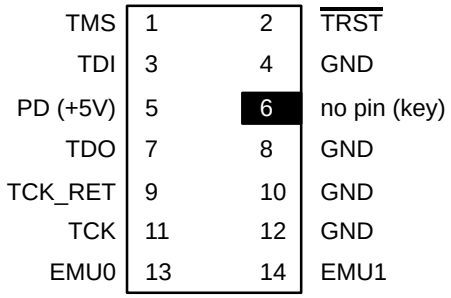
This chapter contains information about connecting your target system to the emulator. Your target system must use a special 14-pin connector for proper communication with the emulator.

Topic **Page**

3.1 Designing Your Target System's Emulator Connector (14-pin Header)

The 'C5x devices support emulation through a dedicated emulation port. This port is a superset of the IEEE 1149.1 (JTAG) standard and is accessed by the emulator. To communicate with the emulator, **your target system must have a 14-pin header** (2 rows of 7 pins) with the connections that are shown in Figure 3–1. Table 3–1 describes the emulation signals.

Figure 3–1. 14-Pin Header Signals and Header Dimensions



Header Dimensions:
Pin-to-pin spacing, 0.100 in. (X,Y)
Pin width, 0.025-in. square post
Pin length, 0.235-in. nominal

Table 3–1. 14-Pin Header Signal Description

Signal	Description	Emulator State	Target State
TMS	JTAG test mode select.	O	I
TDI	JTAG test data input.	O	I
TDO	JTAG test data output.	I	O
TCK	JTAG test clock. TCK is a 10-MHz clock source from the emulation cable pod. This signal can be used to drive the system test clock.	O	I
$\overline{\text{TRST}}$	JTAG test reset.	O	I
EMU0	Emulation pin 0.	I	I/O
EMU1	Emulation pin 1.	I	I/O
PD	Presence detect. Indicates that the emulation cable is connected and that the target is powered up. PD should be tied to +5 volts in the target system.	I	O
TCK_RET	JTAG test clock return. Test clock input to the emulator. May be a buffered or unbuffered version of TCK.	I	O

Note: I = input; O = output

Although you can use other headers, recommended parts include:

straight header, unshrouded	DuPont Connector Systems part number 67996–114
right-angle header, unshrouded	DuPont Connector Systems part number 68405–114

3.2 Bus Protocol

The IEEE 1149.1 specification covers the requirements for JTAG bus slave devices (such as the TMS320C5x devices) and provides certain rules, summarized as follows:

- The TMS/TDI inputs are sampled on the rising edge of the TCK signal of the device.
- The TDO output is clocked from the falling edge of the TCK signal of the device.

When JTAG devices are daisy-chained together, the TDO of one device has approximately a half TCK cycle set up to the next device's TDI signal. This type of timing scheme minimizes race conditions that would occur if both TDO and TDI were timed from the same TCK edge. The penalty for this timing scheme is a reduced TCK frequency.

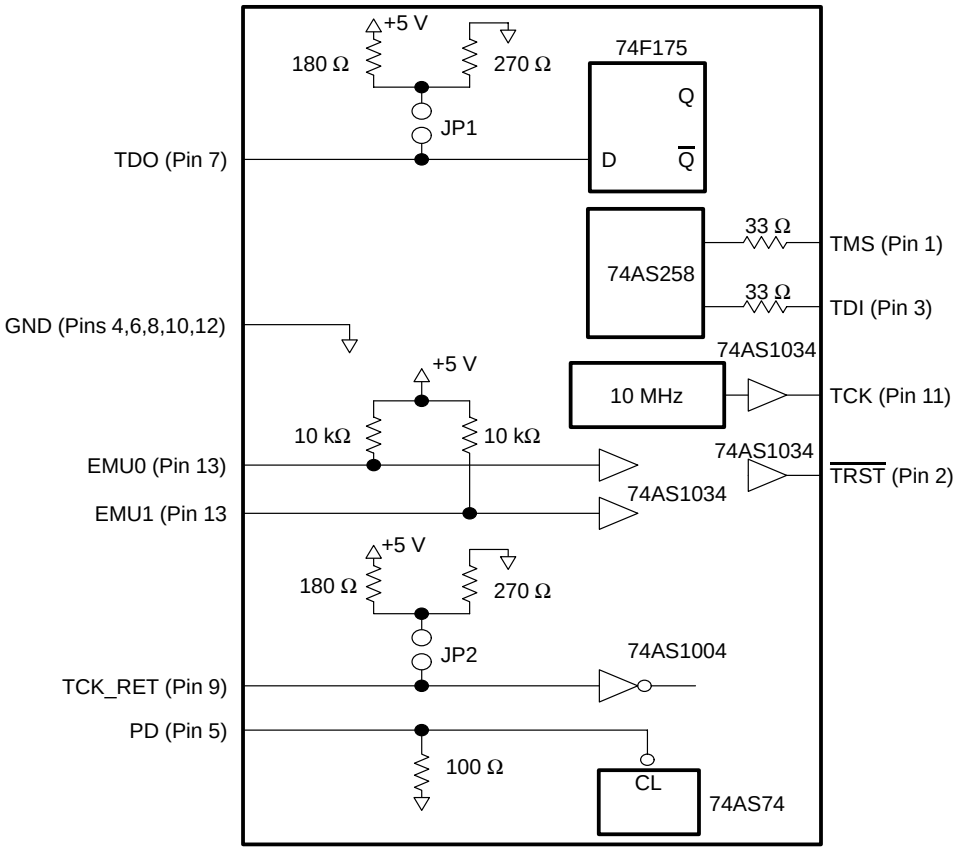
The IEEE 1149.1 specification does not provide rules for JTAG bus master (emulator) devices. Instead, it states that it expects a bus master to provide bus slave compatible timings. The emulator provides timings that meet the bus slave rules and also provides an optional timing mode that allows you to run the emulation at a much higher frequency for improved performance.

3.3 Emulator Cable Pod Logic

Figure 3–2 shows a portion of the emulator cable pod. These are the functional features of the emulator pod:

- Signals TDO and TCK_RET can be parallel-terminated inside the pod if required by the application. The default is that these signals are not terminated.
- Signal TCK is driven with a 74AS1034 device. Because of the high current drive (48 mA I_{OL}/I_{OH}), this signal can be parallel-terminated. If TCK is tied to TCK_RET, then you can use the parallel terminator in the pod.
- Signals TMS and TDI can be generated from the falling edge of TCK_RET, according to the IEEE 1149.1 bus slave device timing rules.They can also be driven from the rising edge of TCK_RET, which allows a higher TCK_RET frequency. The default is to match the IEEE 1149.1 slave device timing rules. This is an emulator software option that can be selected when the emulator is invoked. In general, single-processor applications can benefit from the higher clock frequency. However, in multiprocessing applications, you may wish to use the IEEE 1149.1 bus slave timing mode to minimize emulation system timing constraints.
- Signals TMS and TDI are series-terminated to reduce signal reflections.
- A 10-MHz test clock source is provided. You may also provide your own test clock for greater flexibility.

Figure 3–2. Emulator Pod Interface



3.4 Emulator Cable Pod Signal Timing

Figure 3–3 shows the signal timings for the emulator. Table 3–2 defines the timing parameters for the emulator. The timing parameters are calculated from standard data sheet parts used in the emulator and cable pod. These parameters are for reference only. Texas Instruments does not test or guarantee these timings.

The emulator pod uses TCK_RET as its clock source for internal synchronization. TCK is provided as an optional target system test clock source.

Figure 3–3. Emulator Pod Timings

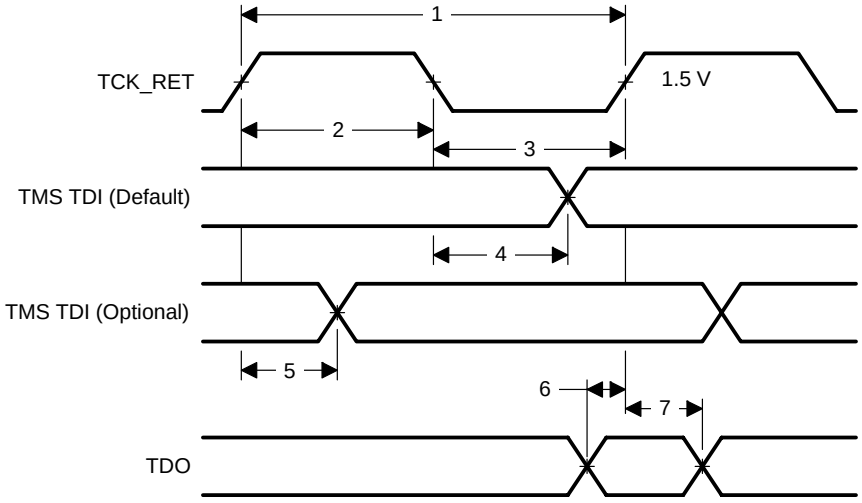


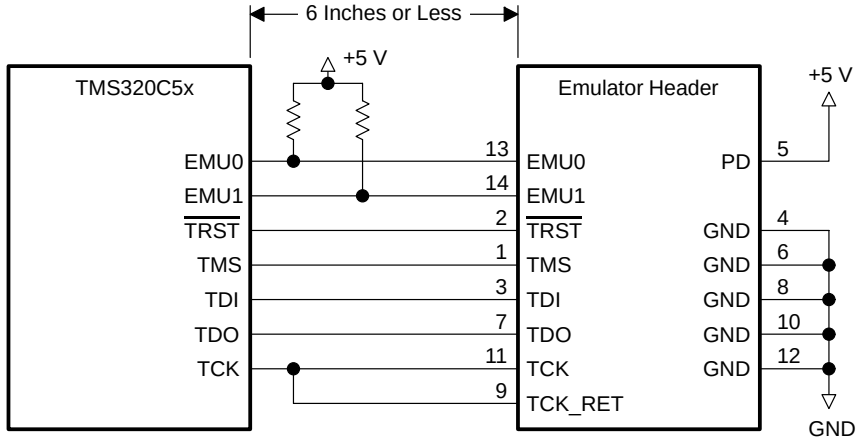
Table 3–2. Emulator Pod Timing Parameters

No.	Reference	Description	Min	Max	Units
1	t_{TCKmin} t_{TCKmax}	TCK_RET period	35	200	ns
2	$t_{TCKhighmin}$	TCK_RET high-pulse duration	15		ns
3	$t_{TCKlowmin}$	TCK_RET low-pulse duration	15		ns
4	$t_{d(XTMXmin)}$ $t_{d(XTMXmax)}$	TMS/TDI valid from TCK_RET low (default timing)	6	20	ns
5	$t_{d(XTMSmin)}$ $t_{d)XTMSmax)}$	TMS/TDI valid from TCK_RET high (optional timing)	7	24	ns
6	$t_{su(XTDOmin)}$	TDO setup time to TCK_RET high	3		ns
7	$t_{hd(XTDOmin)}$	TDO hold time from TCK_RET high	12		ns

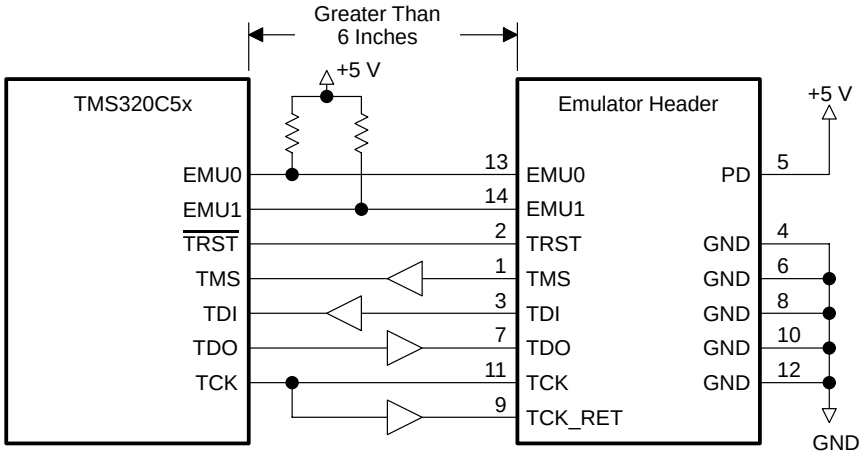
3.5 Buffering Signals Between the Emulator and the Target System

It is extremely important to provide high-quality signals between the emulator and the ’C5x on the target system. If the distance between the emulation header and the ’C5x is greater than 6 inches, the emulation signals must be buffered. The need for signal buffering and placement of the emulation header can be divided into two categories:

- **No signal buffering.** In this situation, the distance between the header and the ’C5x should be no more than 6 inches.



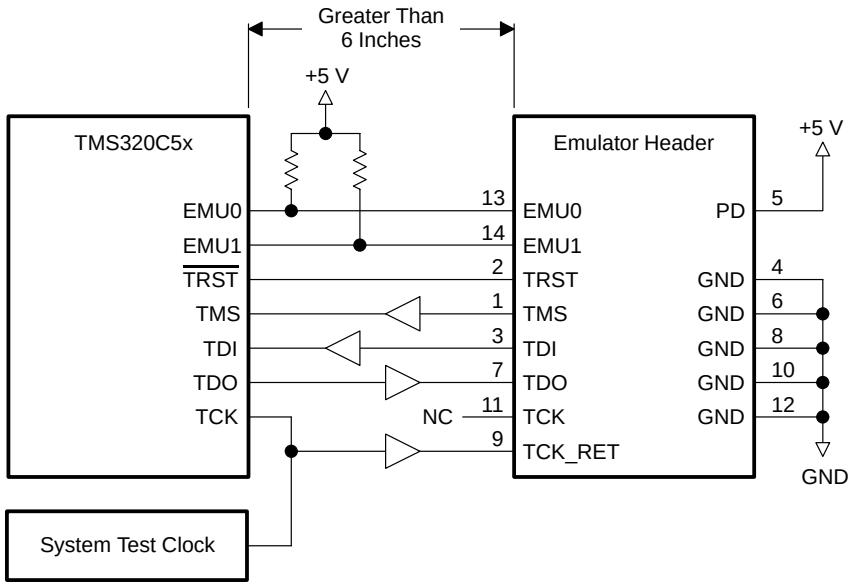
- **Buffered emulation signals.** In this situation, the distance between the emulation header and the ’C5x is greater than 6 inches. The ’C5x emulation signals—TMS, TDI, TDO, and TCK_RET—are buffered through the same package.



- The EMU0 and EMU1 signals must have pullups to 5 volts. The pullup resistor value should be chosen to provide a signal rise of time less than 10 μs. A 4.7-kΩ resistor is suggested for most applications. EMU0 – 1 are I/O pins on the ’C5x; however, they are inputs to the emulator only. In general, these pins are used in multiprocessor systems to provide global run/stop operations.
- It is extremely important to provide high-quality signals, especially on the processor TCK and the emulator TCK_RET signal. In some cases, this may require you to provide special PWB trace routing and to use termination resistors to match the trace impedance. The emulator pod does provide optional internal parallel terminators on the TCK_RET and TDO. TMS and TDI provide fixed series termination.

Figure 3–4 shows an application with the system test clock generated in the target system. In this application, the TCK signal is left unconnected.

Figure 3–4. Target-System Generated Test Clock



There are two benefits to having the target system generate the test clock:

- The emulator provides only a single 10-MHz test clock. If you generate your own test clock, you can set the frequency to match your system requirements.
- In some cases, you may have other devices in your system that require a test clock when the emulator is not connected.

Figure 3–5. Multiprocessor Connections

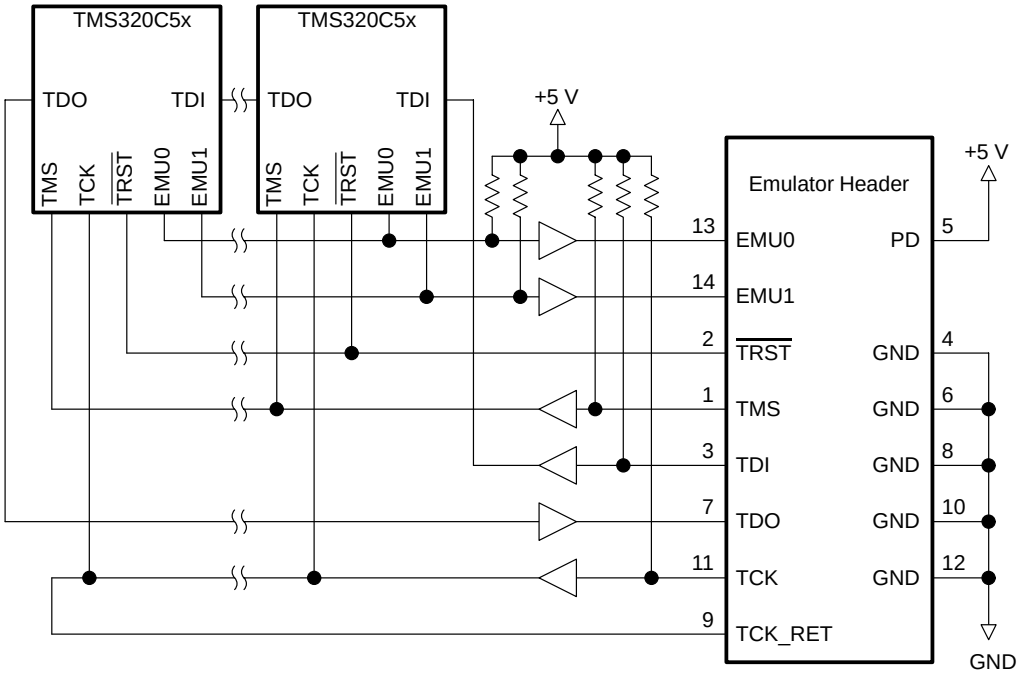


Figure 3–5 shows a typical multiprocessor configuration. This is a daisy-chained configuration (TDO-TDI daisy-chained), which meets the minimum requirements of the IEEE 1149.1 specification. The emulation signals in this example are buffered to isolate the processors from the emulator and provide adequate signal drive for the target system. One of the benefits of a JTAG test interface is that you can generally slow down the test clock to eliminate timing problems. Several key points to multiprocessor support are as follows:

- The processor TMS, TDI, TDO, and TCK should be buffered through the same physical package to better control timing skew.
- The input buffers for TMS, TDI, and TCK should have pullups to 5 volts. This will hold these signals at a known value when the emulator is not connected. A pullup of 4.7 kΩ or greater is suggested.
- Buffering EMU0 and EMU1 is optional but highly recommended to provide isolation. These are not critical signals and do not need to be buffered through the same physical package as TMS, TCK, TDI, and TDO. Unbuffered and buffered signals are shown in Section 3.5 (page 3-7).

3.6 Emulation Timing Calculations

The following are a few examples on how to calculate the emulation timings in your system. For actual target timing parameters, see the appropriate device data sheets.

Assumptions:

$t_{su(TTMS)}$	Target TMS/TDI setup to TCK high	10 ns
$t_{h(TTMS)}$	Target TMS/TDI hold from TCK high	5 ns
$t_{d(TTDO)}$	Target TDO delay from TCK low	15 ns
$t_{d(bufmax)}$	Target buffer delay maximum	10 ns
$t_{d(bufmin)}$	Target buffer delay minimum	1 ns
$t_{(bufskew)}$	Target buffer skew between two devices in the same package: $[t_{d(bufmax)} - t_{d(bufmin)}] \times 0.15$	1.35 ns
$t_{tckfactor}$	Assume a 40/60 duty cycle clock	0.4

Given in Table 3–2 (page 3-6):

$t_{d(XTMSmax)}$	Emulator TMS/TDI delay from TCK_RET low, maximum	20 ns
$t_{d(XTMX)}$	min emulator TMS/TDI delay from TCK_RET low, minimum	6 ns
$t_{d(XTMSmax)}$	Emulator TMS/TDI delay from TCK_RET high, max	24 ns
$t_{d(XTMXmin)}$	Emulator TMS/TDI delay from TCK_RET high, minimum	7 ns
$t_{su(XTDOmin)}$	TDO setup time to emulator TCK_RET high	3 ns

There are two key timing paths to consider in the emulation design:

- ❑ the TCK_RET/TMS/TDI (t_{prdtck_TMS}) path, and
- ❑ the TCK_RET/TDO (t_{prdtck_TDO}) path.

In each case, the worst case path delay is calculated to determine the maximum system test clock frequency.

Case 1: Single processor, direct connection, TMS/TDI timed from TCK_RET low (default timing).

$t_{prdtck\_TMS} = [t_{(d(XTMSmax)} + t_{su(TTMS)}] / t_{tckfactor}$
$= (20 \text{ ns} + 10 \text{ ns}) / 0.4$
$= 75 \text{ ns} \quad (13.3 \text{ MHz})$

$t_{prdtck\_TDO} = [t_{(d(TTDO)} + t_{su(XTDOmin)}] / t_{tckfactor}$
$= (15 \text{ ns} + 3 \text{ ns}) / 0.4$
$= 45 \text{ ns} \quad (22.2 \text{ MHz})$

In this case, the TCK/TMS path is the limiting factor.

Case 2: Single processor, direct connection, TMS/TDI timed from TCK_RET high (optional timing).

$t_{prdtck\_TMS} = t_{d(XTMSmax)} + t_{su(TTMS)}$
$= (24 \text{ ns} + 10 \text{ ns})$
$= 34 \text{ ns (29.4 MHz)}$

$t_{prdtck\_TDO} = [t_{d(TTDO)} + t_{su(XTDOmin)}] / \text{ttckfactor}$
$= (15 + 3) / 0.4$
$= 45 \text{ ns} \quad (22.2 \text{ MHz})$

In this case, the TCK/TDO path is the limiting factor. One other thing to consider in this case is the TMS/TDI hold time. The minimum hold time for the emulator cable pod is 7 ns, which meets the 5-ns hold time of the target device.

Case 3: Single/multiple processor, TMS/TDI buffered input; TCK_RET/TDO buffered output, TMS/TDI timed from TCK_RET high (optional timing).

$t_{prdtck\_TMS} = t_{d(XTMSmax)} + t_{su(TTMS)} + 2t_{d(bufmax)}$
$= 24 \text{ ns} + 10 \text{ ns} + 2 (10)$
$= 54 \text{ ns (18.5 MHz)}$

$$t_{prdtck\_TDO} = \frac{t_{d(TTDO)} + t_{su(XTDOmin)} + t_{bufskew}}{t_{tckfactor}}$$

$= (15 \text{ ns} + 3 \text{ ns} + 1.35 \text{ ns}) / 0.4$
$= 58.4 \text{ ns} \quad (20.7 \text{ MHz})$

In this case, the TCK/TMS path is the limiting factor. The hold time on TMS/TDI is also reduced by the buffer skew (1.35 ns) but still meets the minimum device hold time.

Case 4: Single/multiprocessor, TMS/TDI/TCK buffered input; TDO buffered output, TMS/TDI timed from TCK_RET low (default timing).

$$t_{prdtck\_TMS} = \frac{t_{d(XTMSmax)} + t_{su(TTMS)} + t_{bufskew}}{t_{tckfactor}}$$

$$= (24\text{ ns} + 10\text{ ns} + 1.35\text{ ns}) / 0.4$$

$$= 88.4\text{ ns (11.3 MHz)}$$

$$t_{prdtck\_TDO} = \frac{(tdTTDO + tsuXTDOmin + tdbufmax)}{t_{tckfactor}}$$

$$= (15\text{ ns} + 3\text{ ns} + 10\text{ ns}) / 0.4$$

$$= 70\text{ ns (14.3 MHz)}$$

In this case, the TCK/TMS path is the limiting factor.

In a multiprocessor application, it is necessary to ensure that the EUM0–1 lines can go from a logic low level to a logic high level in less than 10 μs. This can be calculated as follows (remember that t = 5 RC):

$$t_{rise} = 5(R_{pullup} \times N_{devices} \times C_{load\_per\_device})$$

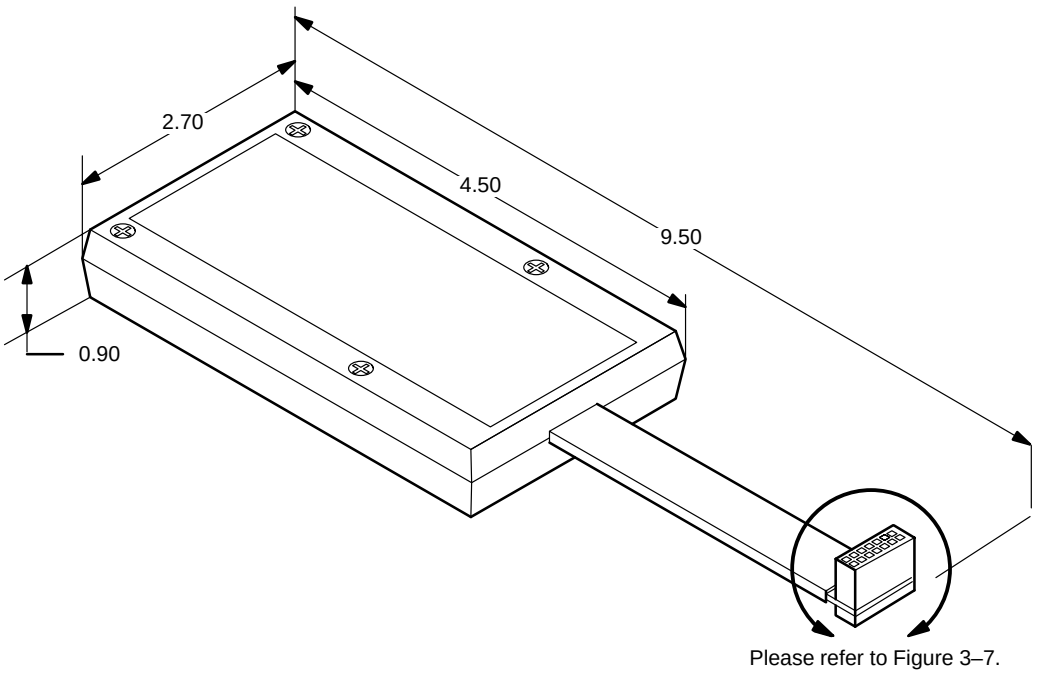
$$= 5(4.7K \times 16 \times 15pF)$$

$$= 5.64\ \mu s$$

3.7 Mechanical Dimensions for the 14-Pin Emulator Connector

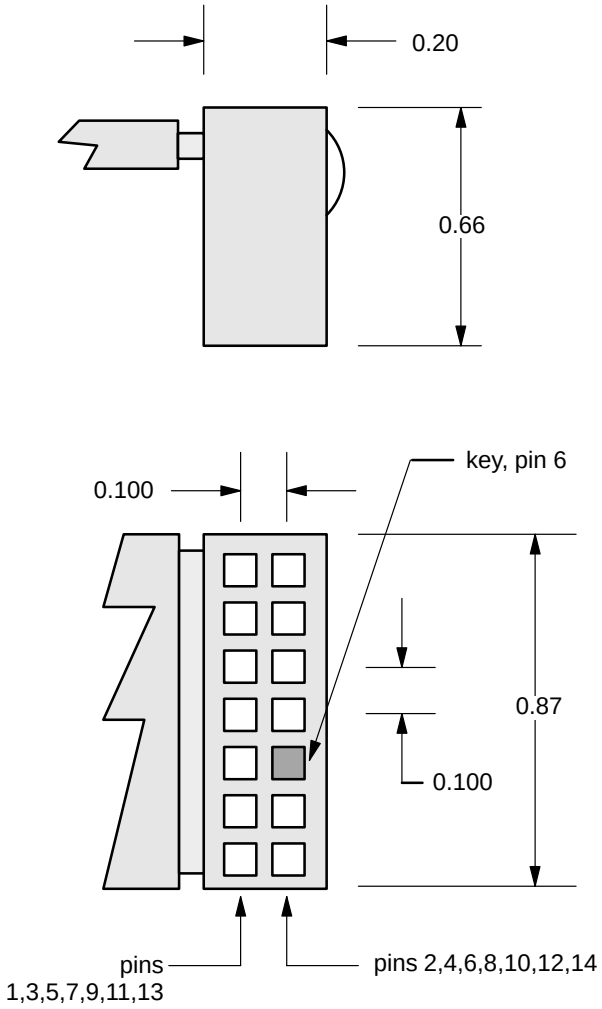
The ’C5x emulator target cable consists of a 3-foot section of jacketed cable, an active cable pod, and a short section of jacketed cable that connects to the target system. The overall cable length is approximately 3 feet10 inches. Figure 3–6 and Figure 3–7 (page 3-14) show the mechanical dimensions for the target cable pod and short cable. Note that the pin-to-pin spacing on the connector is 0.100 inches in both the X and Y planes. The cable pod box is nonconductive plastic with four recessed metal screws.

Figure 3–6. Pod/Connector Dimensions



Note: All dimensions are in inches and are nominal dimensions, unless otherwise specified.

Figure 3–7. 14-Pin Connector Dimensions



Note: All dimensions are in inches and are nominal dimensions, unless otherwise specified.

Index

A

B

C

D

E

L

M

N

O

P

R

S

T

V

X

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.

Copyright © 1997, Texas Instruments Incorporated