Application Report
SPRA439D–August 2008

Emulation Fundamentals for TI's DSP Solutions

Chuck Brokish Senior Member, Technical Staff *Field Design and Applications*

ABSTRACT

In software development, perhaps the most critical, yet least predicitible stage in the process is debugging. Of the many factors that come into play when debugging software applications, time is of utmost importance. The time required to set up and debug a software application can have major impacts on time-to-market, meeting customer expectations, and ultimately the financial impact of a well-developed product.

This paper will explain the fundamentals of how the emulation logic and emulation tools work together with the TI digital signal processors. By understanding the fundamentals of emulation, you will be able to accelerate the process of setting up and performing software debug, as well as aid in troubleshooting potential problems in the debugging setup.

A detailed explanation of the setup of the emulator hardware systems for single and multi-processor applications, along with a discussion of how the system components interact during debug will be discussed in the sections to follow. Also included is a troubleshooting guide to assist in common setup problems.

Topic **Page**

Trademarks

Code Composer Studio is a trademark of Texas Instruments.

Windows is a trademark of other.

1 Introduction

Texas Instruments' debuggers allow extensive visibility into the processors, their registers, and the application's software. This visibility enables the software engineer to:

- understand what changes are taking place inside of the processor as the application executes
- set breakpoints in the application, based on hardware signal values or software locations within the application

At these breakpoints, you can:

- understand the state of the processor and data and determine if the application is still operating correctly
- perform benchmarking (timing analysis) and profiling (CPU loading) of the application software within the emulator

Additionally, multiprocessor debug allows you to:

- debug software on several processors at the same time
- provide a method of stopping one or multiple processors based on a condition set in another processor
- capture the entire system state at the time in question

These capabilities, along with many more within the Texas Instruments debuggers, can greatly reduce debugging time in the software development cycle.

1.1 Hardware System Requirements

To understand the workings of the emulation logic on TI processors, it is first necessary to understand the basic components of emulation. The following hardware components are required to properly debug the target system:

- A personal computer (refer to your CCS manual for minimum requirements)
- Windows™2000 SP4 or Windows™ XP SP2
- Code Composer Studio™(appropriate version)
- Emulator (XDS560, XDS510 or XDS560/510 class emulator)
- Emulation interface pod (XDS560T, XDS560, XDS510 or XDS560/510 class emulation pod)
- Target (not required if using TI simulator)

1.2 Hardware Setup

When using the TI debugger for software debugging on a hardware platform, it is necessary to perform a few setup procedures to ensure that the target processor works properly with the XDS510, XDS560, or third-party emulation hardware.

The emulation setup is comprised of two tools: the emulator (i.e., XDS510 or XDS560) which controls the information flow to and from the target, and the debugger, which is the user interface to this information. Beyond the emulation setup is the target processor. TI uses several common interface connections between the target and the emulation HW. The protocol used within the emulation logic of TI processors is the Joint Test Action Group (JTAG).

This section covers the basics of ensuring proper communication setup between the emulation hardware and the target processor.

1.3 General Understandings

Debug of the hardware is performed by stopping the core to enable information to be scanned into and out of the device via the JTAG header. This information is transferred serially through the JTAG port following the IEEE 1149.1 JTAG specifications. It is important to understand that this debug method is near real-time, but is intrusive, as it may require that the core be halted to scan the information.

While the connection to the JTAG header may be the same, the scan chains used for emulation purposes are different from those used for boundary scan. Internal to the processor, there are various serial scan chains through which the information can be scanned. The emulator card controls which scan chain is used, and what information is contained in each scan chain. Traditionally referred to as the Scan Manager, this function assumes the task of controlling all information scanned into and out of the various processors in the scan chain. Further, it directs this information to and from the various debugger windows.

When using a personal computer (PC), the JTAG connection to the target processor can be made through several options including a PCI card (Figure 1), ISA card, (Figure 2), USB, PCMCIA, Ethernet, or many other options. This information regarding the scan control is available to the PC host via dynamically linked libraries (DLLs).

Additional information regarding physical connection of the XDS510 and XDS560 debugger interfaces is available in the documentation that is shipped with the product. Other helpful information, can be found at other locations within Texas Instruments, such as the TI Bulletin Board and the TI Internet Homepage, http://www.ti.com.

TI third parties are making other emulation interfaces available, including PCMCIA, USB, and Ethernet connections. With these connection methods, scan management is performed by the CPU of the host computer and an external processor, respectively.

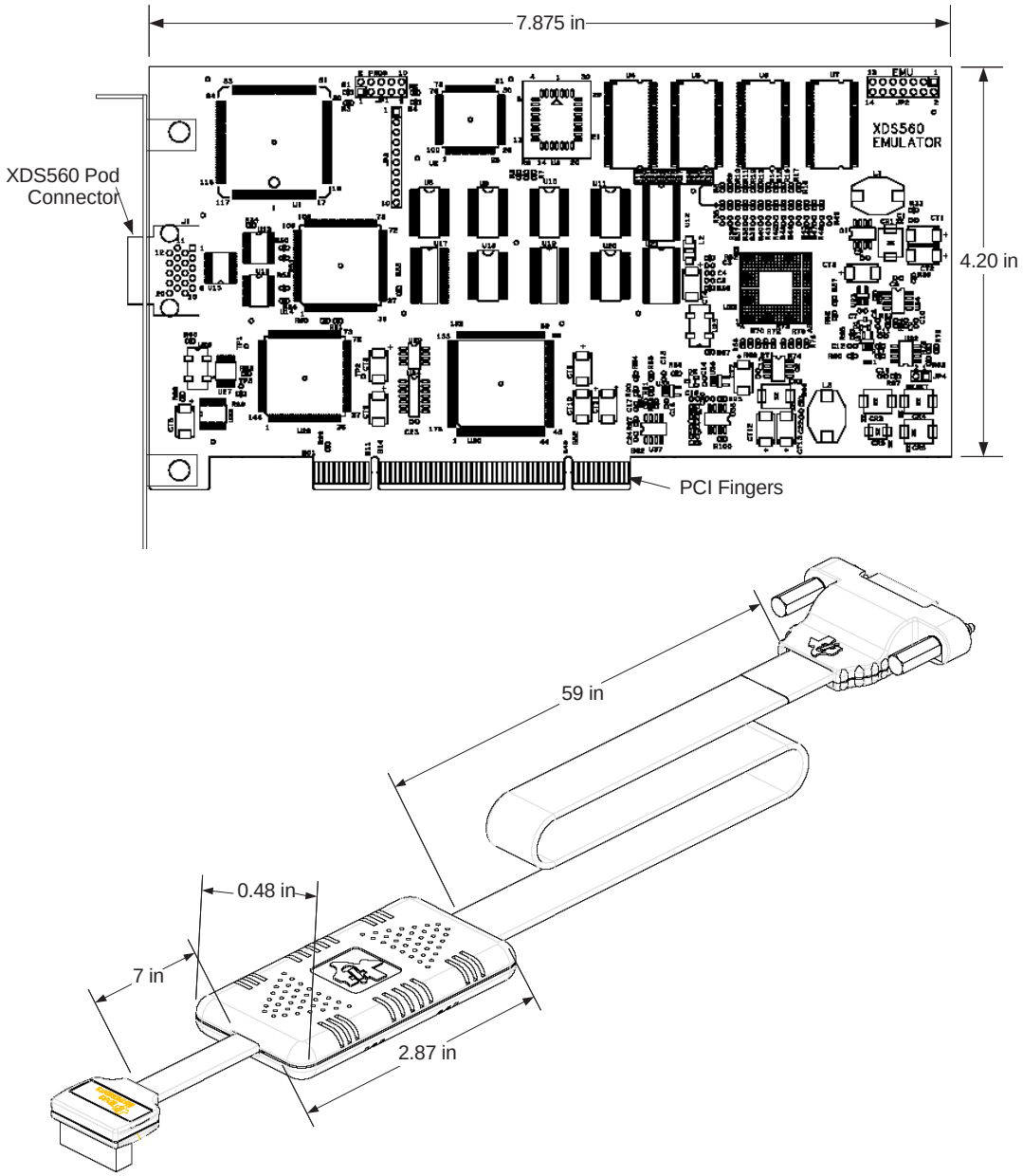

Figure 1. Connecting the XDS560 PCI card to the Target System

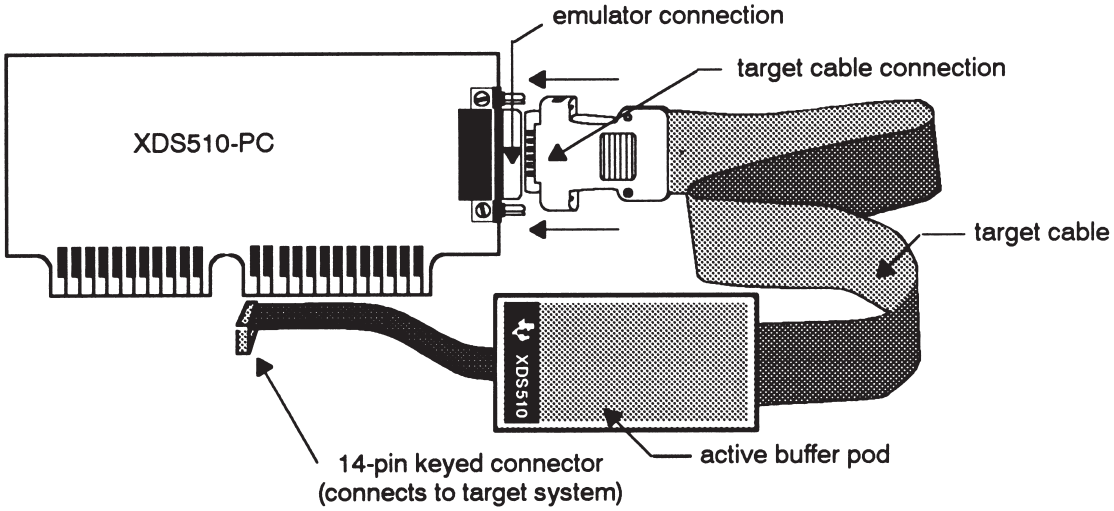

Figure 2. Connecting the XDS520 ISA card to the Target system

1.4 Signal Descriptions

There are five signals and one ground used for JTAG connections to control the JTAG test access port (TAP) controller, as defined by the IEEE 1149.1 standard. These signals are shown in Table 1.

Table 1. Standard TAP Controller JTAG

Signal Name	Descriptions	Emulator Signal Type	Target Signal Type
TRST	Test Reset	O	I
TMS	Test Mode Select	O	I
TDI	Test Data Input	O	I
TDO	Test Data Output	I	O
TCK	Test Clock (clock provided by the XDS510 or XDS560 pod)	O	I
GND	Ground		

I = Input; O = Output

TI also uses two other signals provided on the JTAG header as a method of confirming proper connection of the emulation hardware (Table 2).

The Target Voltage Detect (formally Presence Detect) signal and the Test Clock Return are used to confirm the presence of the target device and the proper functioning of signals to and from the XDS510 or XDS560 emulators. Their functionality will be further explained when EMURST and troubleshooting hardware problems are discussed.

Documentation on some of the newer TI devices refer to signals named ET0, ET1, and TVD. These signals are identical to the traditional signals named EMU0, EMU1, and PD, respectively; however, their names have been changed to better reflect their use within TI's latest emulation solutions.

Presence Detect (PD) has been renamed Target Voltage Detect to indicate that, not only is it used to detect the presence of a target being connected to the JTAG header, but it is also used to detect the level of the voltage on the JTAG signals for that device. On the newer emulation hardware, the JTAG port is able to adapt the voltage to match the voltage of the target. This provides a means of interfacing with legacy devices, which operated at a higher voltage (up to 5V) as well as the newer devices, whose JTAG signals are capable of operating down to the CPU voltage of 0.8V and below.

The TCK_RET signal is used to clock the logic within the emulator. It is not typically a signal which is actually present on the chip being debugged. Rather, it is a representation of the TCK signal supplied to the target board, but routed through the target board such that it can accurately represent any delays or signal effects observed by other JTAG signals.

If the other JTAG signals are buffered (as shown in Figure 4), then the TCK pin should be buffered as well before being brought out on the TCK_RET pin. This ensures that the clock within the pod is properly synchronized with the associated signals which it is clocking internal to the pod.

However, on devices that include an ARM core of version ARM926 and above, the TCK_RET signal must actually come from a dedicated pin of the chip itself. This is because ARM Ltd varies from the IEEE 1149.1 specification and makes the JTAG clock dependent on the CPU clock. Therefore, the RTCK signal from the ARM core is actually in internal synchronization with both the TCK signal from the JTAG header as well as the CPU clock.

Table 2. Connection Verification JTAG Signals

Signal Names	Decriptions	Emulator Signal Type	Target Signal Type
TCK_RET (or RTCK)	Test Clock Return (buffered or unbuffered connection of TCK coming back from the target board)	I	Signals connected to target board; not to the chip
	Fro devices using ARM core revision 926 and above, RTCK is brought ut of a dedicated pin on the device.	I	O
TDV or PD_(V_{cc})	Target Voltage Detect	I	O
I = Input; O = Output			

Texas Instruments adds two more signals to the JTAG header , which are used for advanced emulation capability. These signals, shown in Table 3, provide the capability to perform benchmarking, software profiling, and multi-processor emulation with inter-processor breakpoint capabilities

Table 3. TI Advanced Emulation JTAG Signals

Signal Name	Description	Emulator Signal Type	Target Signal Type
EMU0 (or ET0)	Emulation Pin 0 (or Emulation Test 0)	I	I/O
EMU1 (or ET1)	Emulation Pin 1 (or Emulation Test 1)	I	I/O
I = Input; O=Output			

The use of the EMU signals is detailed in the **C-Source Debugger User's Guide**. The EMU signals are used by the emulator to provide:

- Clocking information when performing
  - software benchmarking
  - software profiling

The EMU signals are also used with the ANALYSIS window of the debugger to assist in multiprocessor debugging and serve a sglobal breakpoints for multiprocessor setups. On the newer devices, the EMU (or ET) signals provide data streams for high-speed real-time data exchange (RTDX).

Setup of these signals is accomplished using the SET plug-in or in the case of multiprocessor debugging, is done within the parallel debug manager (PDM) which will be discussed more in Section 3.5.

Table 4. Comprehensive list of Emulation Test Pins

Emulation Header	Signal Name	Description	Emulator or Signal Type	Target Signal Type
14p, 20p, 60p[1]	EMU0 (or ET0)	Emulation Pin 0 (or Emulation Test 0)	I/O	I/O
14p, 20p, 60p[1]	EMU1 (or ET1)	Emulation Pin 0 (or Emulation Test 1)	I/O	I/O
20p, 60p[2],[3]	EMU2 (or ET2)	Emulation Pin 0 (or Emulation Test 2)	I/O	I/O
20p, 60p[2],[3]	EMU3 (or ET3)	Emulation Pin 0 (or Emulation Test 3)	I/O	I/O
20p, 60p[2],[3]	EMU4 (or ET4)	Emulation Pin 0 (or Emulation Test 4)	I/O	I/O
60p[3]	EMU5 (or ET5)	Emulation Pin 0 (or Emulation Test 5)	I	O
60p[3]	EMU6 (or ET6)	Emulation Pin 0 (or Emulation Test 6)	I	O
60p[3]	EMU7 (or ET7)	Emulation Pin 0 (or Emulation Test 7)	I	O
60p[3]	EMU8 (or ET8)	Emulation Pin 0 (or Emulation Test 8)	I	O
60p[3]	EMU9 (or ET9)	Emulation Pin 0 (or Emulation Test 9)	I	O
60p[3]	EMU10 (or ET10)	Emulation Pin 0 (or Emulation Test 10)	I	O
60p[3]	EMU11 (or ET11)	Emulation Pin 0 (or Emulation Test 11)	I	O
60p[3]	EMU12 (or ET12)	Emulation Pin 0 (or Emulation Test 12)	I	O
60p[3]	EMU13 (or ET13)	Emulation Pin 0 (or Emulation Test 13)	I	O
60p[3]	EMU14 (or ET14)	Emulation Pin 0 (or Emulation Test 14)	I	O
60p[3]	EMU15 (or ET15)	Emulation Pin 0 (or Emulation Test 15)	I	O
60p[3]	EMU16 (or ET16)	Emulation Pin 0 (or Emulation Test 16)	I	O
60p[3]	EMU17 (or ET17)	Emulation Pin 0 (or Emulation Test 17)	I	O

(1) Can be used for basic emulation, HS-RTDX, and Trace (currently)
(2) Can be used for HS-RTDX or Trace (currently)
(3) Can be used for Trace only (currently)

Table 4. Comprehensive list of Emulation Test Pins (continued)

Emulation Header	Signal Name	Description	Emulator or Signal Type	Target Signal Type
60p[(3)]	EMU18 (or ET18)	Emulation Pin 0 (or Emulation Test 18)	I	O
14p refers to the tradition 14 pin [7x2] header with 0.1" pitch; 20p refers to TI's compact 20 pin [2x10] header with 0.1" x 0.050" pitch; 60p refers to TI's 60 pin [4x15] header with 0.25" x 0.050" pitch				

Most EMU signals are bi-directional (both input and output) at the target. They can be used as an output and driven low by a device as a result of breakpoint conditions being met. They can also be used as inputs, and monitored in the debugging logic. This allows one core to set the EMU signal, and another device (or devices) to break as a result.

Figure 3 demonstrates the common connection of these signals on the user's hardware design. If the processor is six inches or more from the JTAG header, there could be degradation of the signal due to loading of the signal path. In this case, you should use buffers to drive the signals and maintain a suitable signal quality to ensure proper emulation (Figure 4). It should be noted that proper buffer selection is as important as proper signal routing on the target.

For devices that include TCK_RET as an output from the target, be sure to use that signal, and do not simply buffer the TCK input signal (Figure 5).

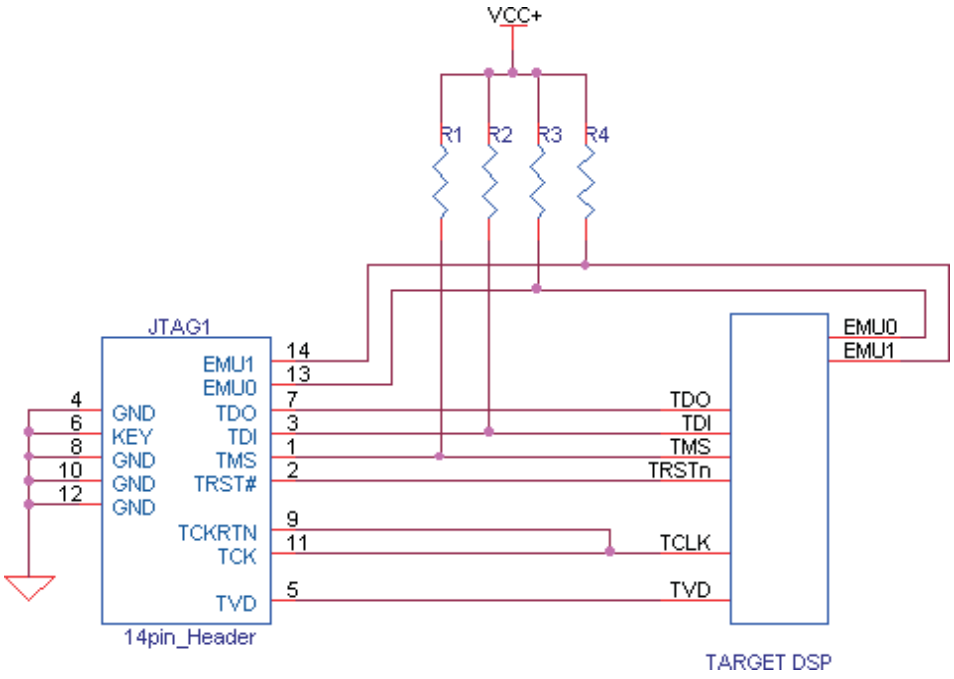

Figure 3. Emulator Connections Without Signal Buffering

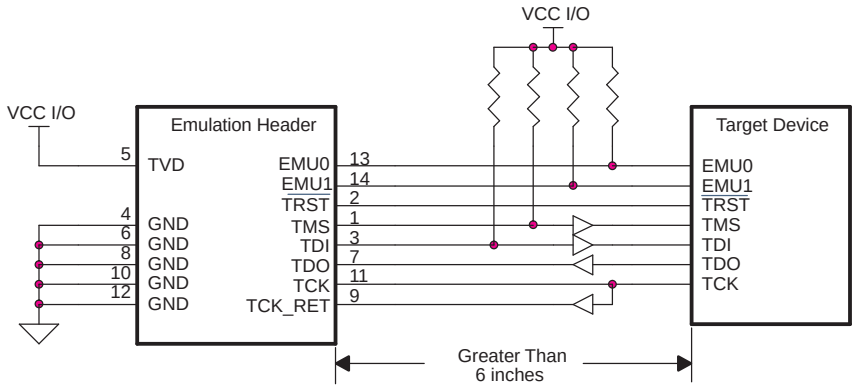

Figure 4. Emulator Connections with Signal Buffering

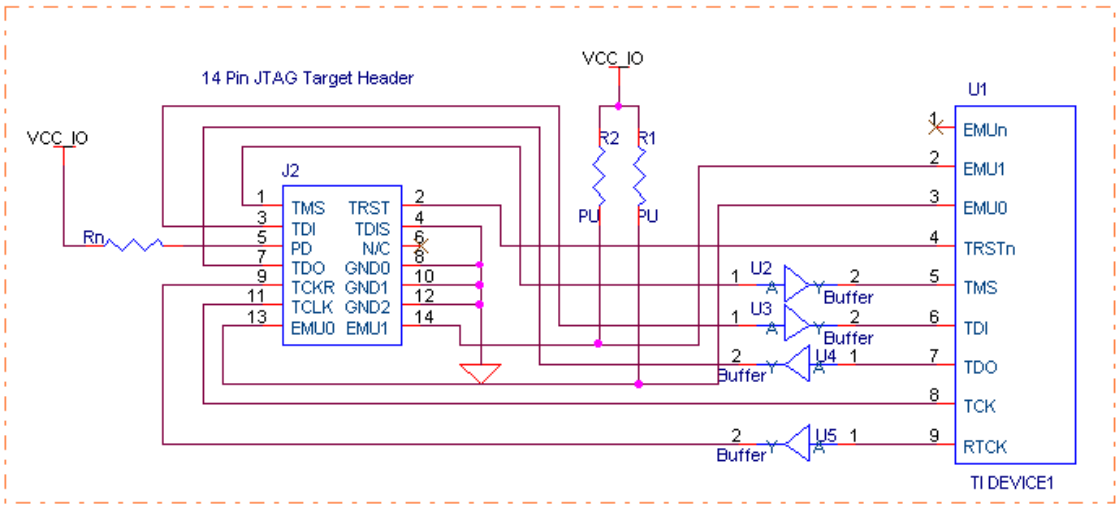

Figure 5. Emulator Connections on Devices Which Provide TCK RET

Texas Instruments allows for the use of a standard 14-pin keyed header for all of their JTAG devices.

TMS	1	2	TRST#
TDI	3	4	TDIS
TVD	5	6	N/C
TDO	7	8	GND
TCK	9	10	GND
TCKRTN	11	12	GND
EMU0	13	14	EMU1

Figure 6. Standard 14-pin Keyed JTAG Header

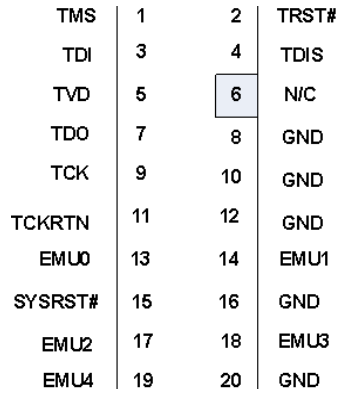

Figure 7. Compact 20-pin TI Keyed JTAG Header

Table 5. 60-Pin TI Advance Emulation Header

60-Pin Header Interface Pin Out				
Column	A	B	C	D
Row				
1	GND[1]	ID0	ID2	NC[2]
2	GND	TMS	EMU18[3]	GND
3	GND	EMU17[3]	$\overline{\text{TRST}}$	GND
4	GND	TDI	EMU16[3]	GND
5	GND	EMU14[3]	EMU15[3]	GND
6	GND	EMU12[3]	EMU13[3]	GND
7	GND	TDO	EMU11	GND
8	Type 0	TVD	TCLKRTN	Type 1
9	GND	EMU9	EMU10	GND
10	GND	EMU7	EMU8	GND
11	GND	EMU5	EMU6	GND
12	GND	TCLK	EMU4	GND
13	GND	EMU2	EMU3	GND
14	GND	EMU0	EMU1	GND
15	$\overline{\text{TGTRST}}$	ID1	ID3	GND

(1) On some target applications, detection of the emulation pod is required. It is recommended that when this condition exists, pin A1 on the 60-pin header be used. Instead of grounding pin A1 on the target board, a pullup resistor greater than or equal to 10K should be connected to the pin. when the emulation pod header is connected, pin A1 will be grounded.
(2) Pin D1 is intended for future expansion and should be left unconnected.
(3) The 60-pin header provides 19 EMU pins for advance emulation features. Not all devices will contain this number of EMU pins. Connect only the EMU pins present on the target DSP to the corresponding EMU pin on the header.

Figure 6, Figure 7, and Table 5 illustrate the signal pin out for all current TI emulation header interfaces. Refer to specific header documents listed in the reference section of this document.

2 Software Setup

The software on your computer must be set up to recognize the devices that are physically located within the JTAG scan chain. The tool used to perform this function is called Code Composer Studio™ Setup. It is installed on your desktop as an icon and executes a CC_Setup.exe program. This section explains how to designate the specific target[s] that reside within your system.

2.1 Board Configuration Information

The information regarding the scan chain is entered through the use of Code Composer Studio Setup (CC_Setup). This program allows you to enter unique names for the target processors, or use the default names to simply indicate which processor[s] is in the scan chain.

When starting CC_Setup, a three-paneled screen appears which allows you to enter information regarding the board System Configuration, available board/simulator types, and driver installation as shown in Figure 8. Several boards are already defined, and you have the option of defining your own custom boards. You can also save them for reloading later, and share them with colleagues.

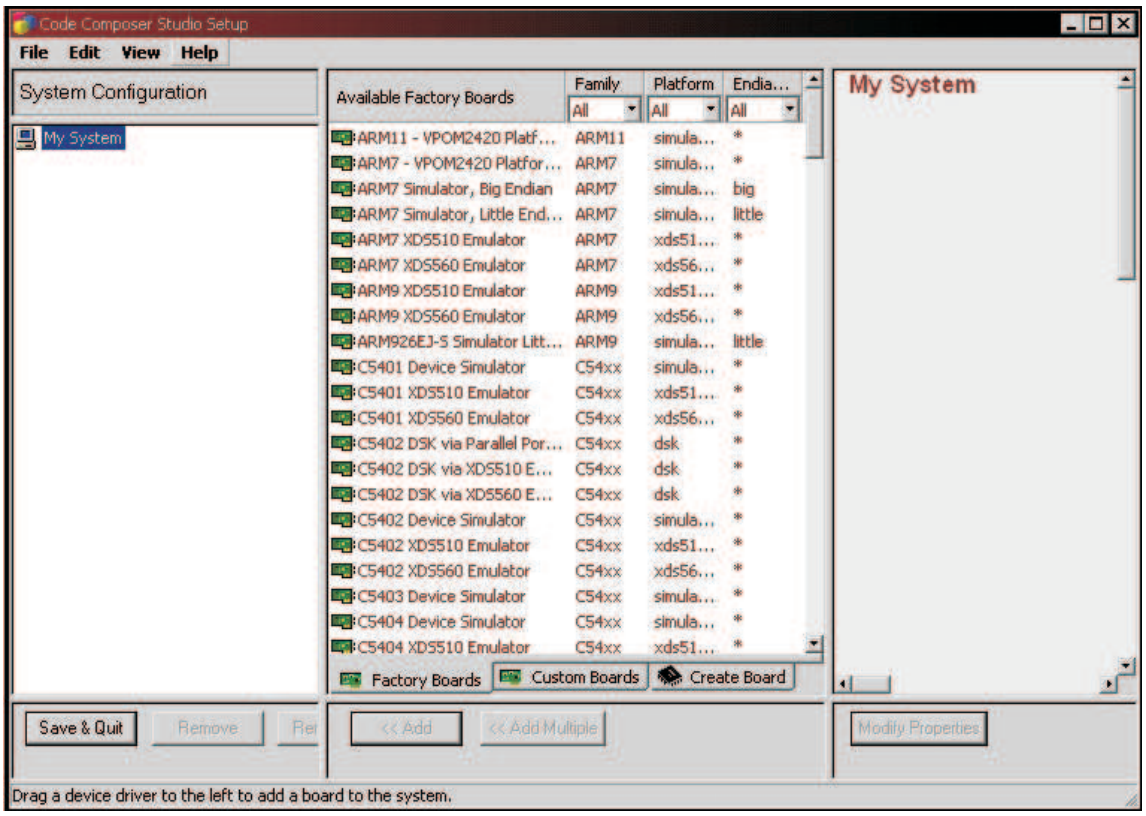

Figure 8. Startup Screen for CCS Setup

2.2 Single Processor Debug

If only a single device is included in the scan chain, the signals are connected as shown in Figure 3 and Figure 4. In this case, the debugger can be set up by specifying the device family for the target processor, and then providing information regarding the physical connection of the emulation hardware.

It is also possible to set up the software such that you can debug multiple targets; this will be discussed further in the next section.

When specifying the target device, you should be able to find a descriptor for the device family and connection method from the center column. For this example, there is a TMS320C5510 DSP device being connected via the XDS560 hardware. You will start by dragging the appropriate Connection hardware from the center column to the left column and dropping it under "My System."

By selecting and dragging an available device from the center column into the left-hand column, you can begin the process of describing your system configuration as shown in Figure 9.

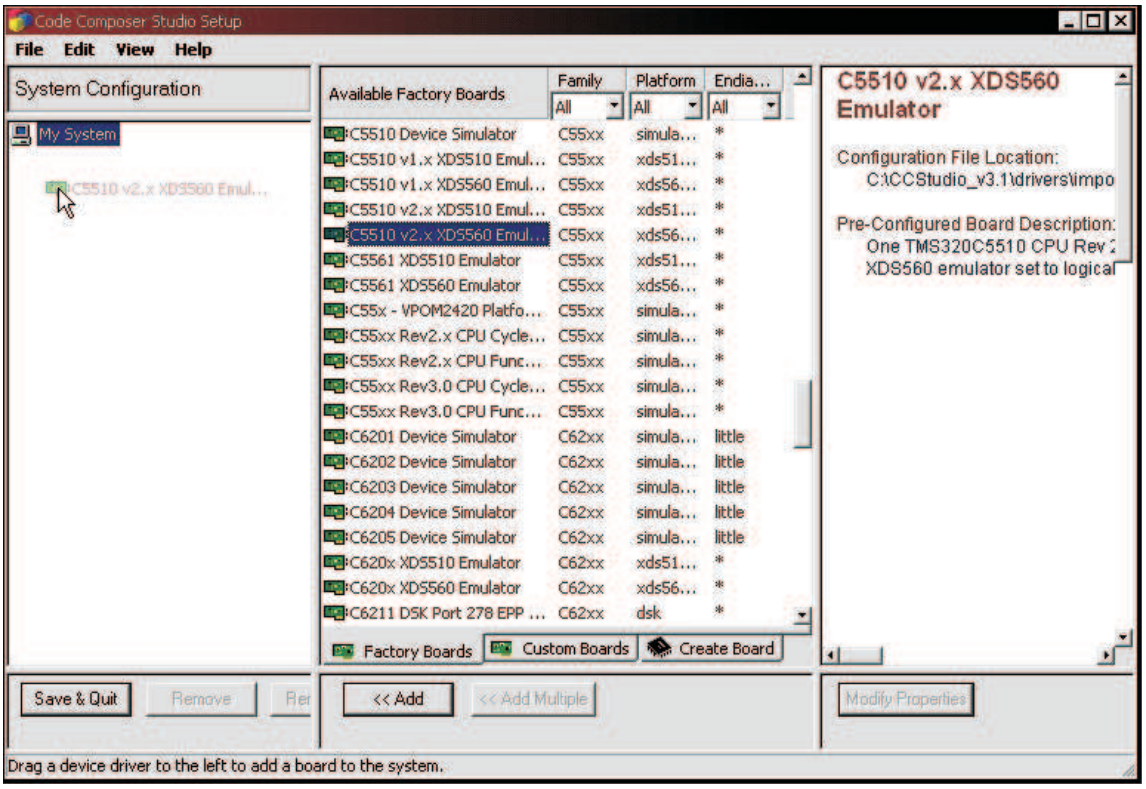

Figure 9. Drag and Drop Processor/Emulator Selection

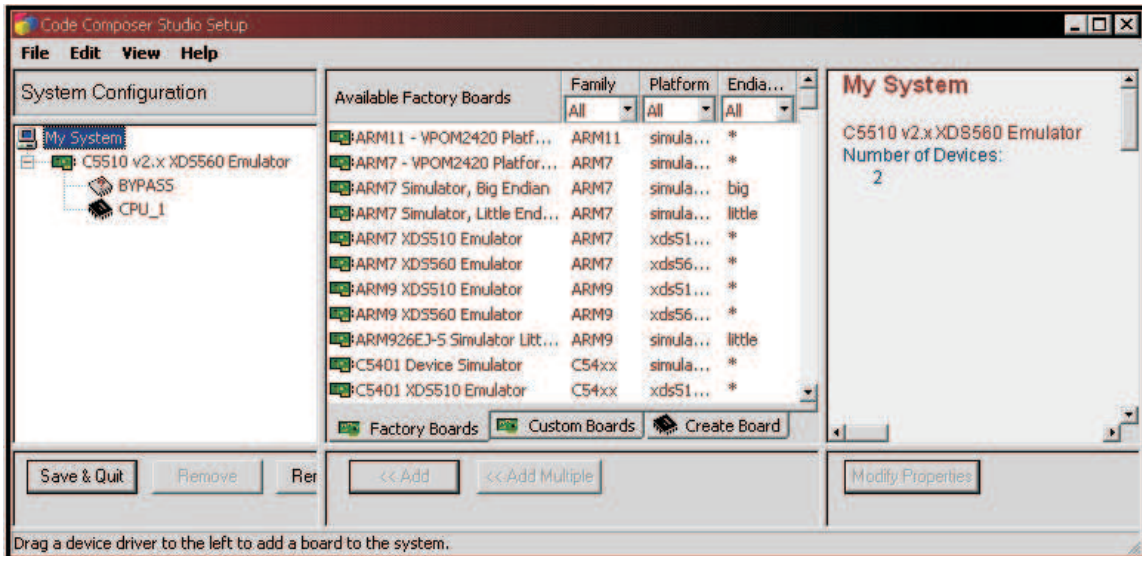

Figure 10. Result of Drag and Drop Processor/Emulator Selection

Figure 10 illustrates the resulting configuration after successfully dragging and dropping the appropriate processor/emulator combination into the left pane or System Configuration window.

It may be necessary to fine-tune the properties of your configuration. To accomplish this, right-mouse click on the target/emulator in the System Configuration window and select Properties.

Figure 11. Adjusting System Configuration Properties

A dialog box will pop up to allow for more detailed configuration. First is the description of the Board Name. You can use the default, or select your own (Figure 11).

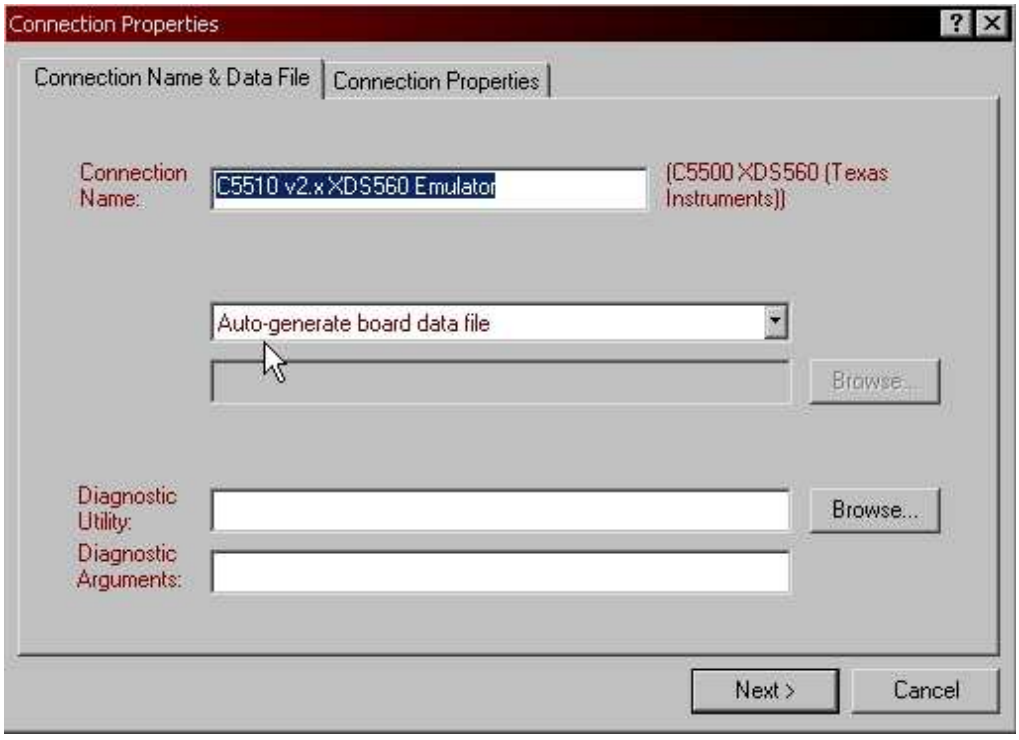

Figure 12. Adjusting the Connection Properties

Next, specify the physical address that your emulation hardware is connected within your host computer. When using the XDS510, the default address for TI hardware is address 0x240 (Figure 12).

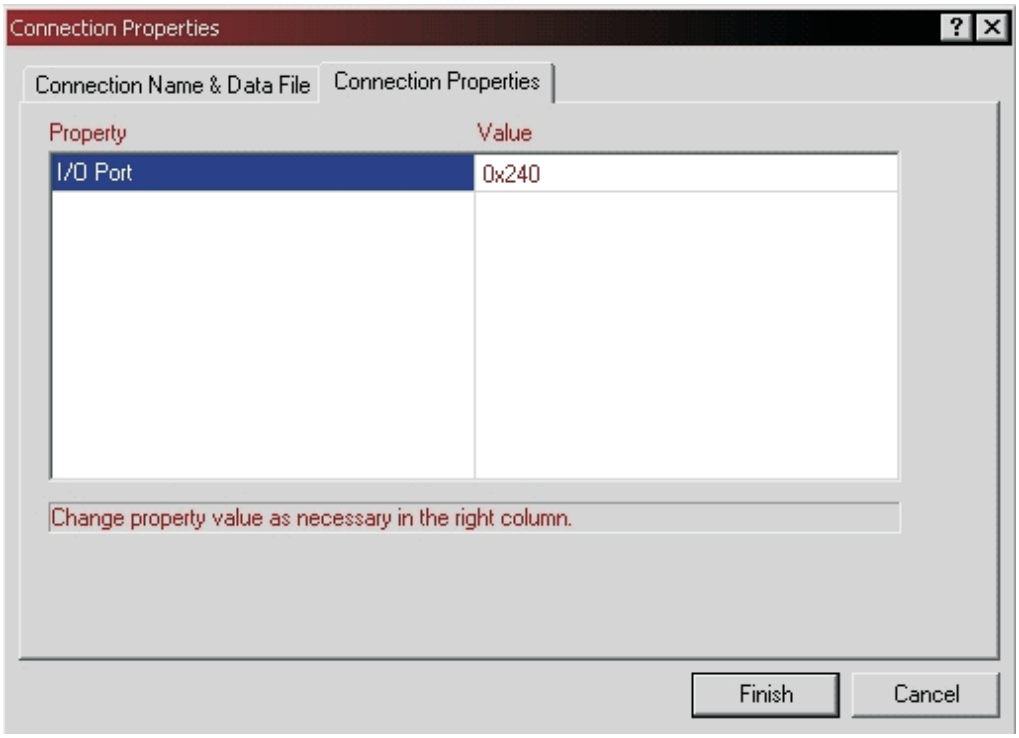

Figure 13. Defining a Connection to the XDS510

When using the XDS560 emulation hardware, the plug-and-play software of the PC operating system automatically detects the default address for the hardware (Figure 14). Within the board properties of the XDS560, you also have the option of selecting the clock speed you want to use for the TCK signal, or allow the hardware to select the optimal clock speed for your setup.

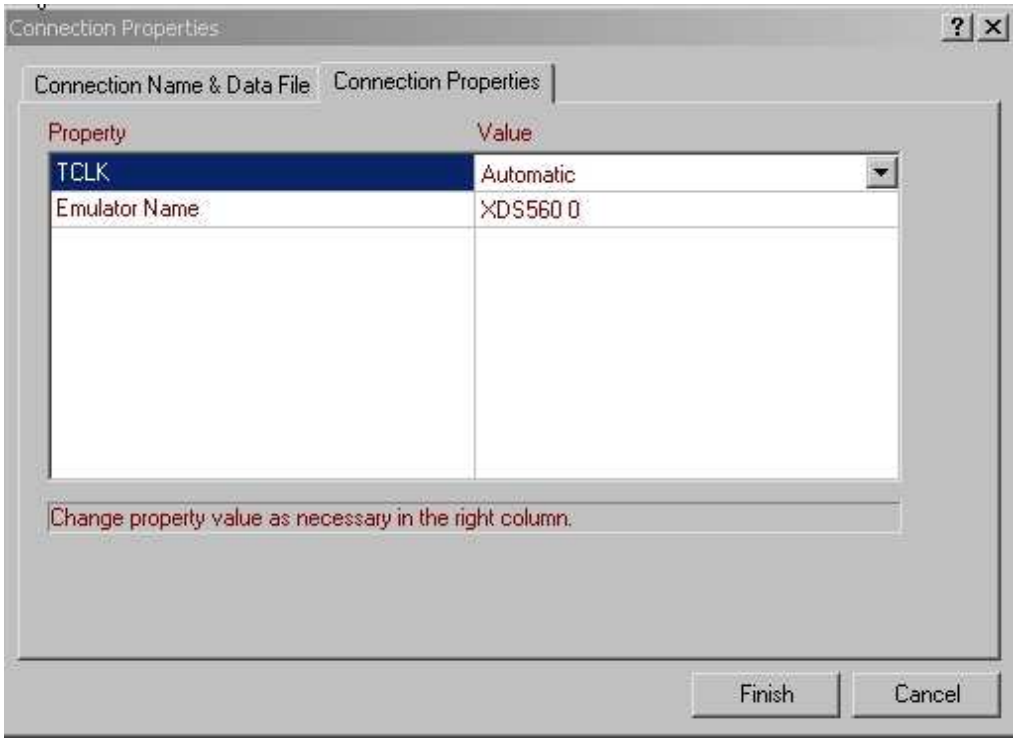

Figure 14. Defining a Connection to the XDS560

Finally, you can select a GEL file that you want to run when Code Composer Studio opens (Figure 15). The General Extension Language (GEL) is an interpretive language similar to C that lets you create functions to extend Code Composer Studio's usefulness.

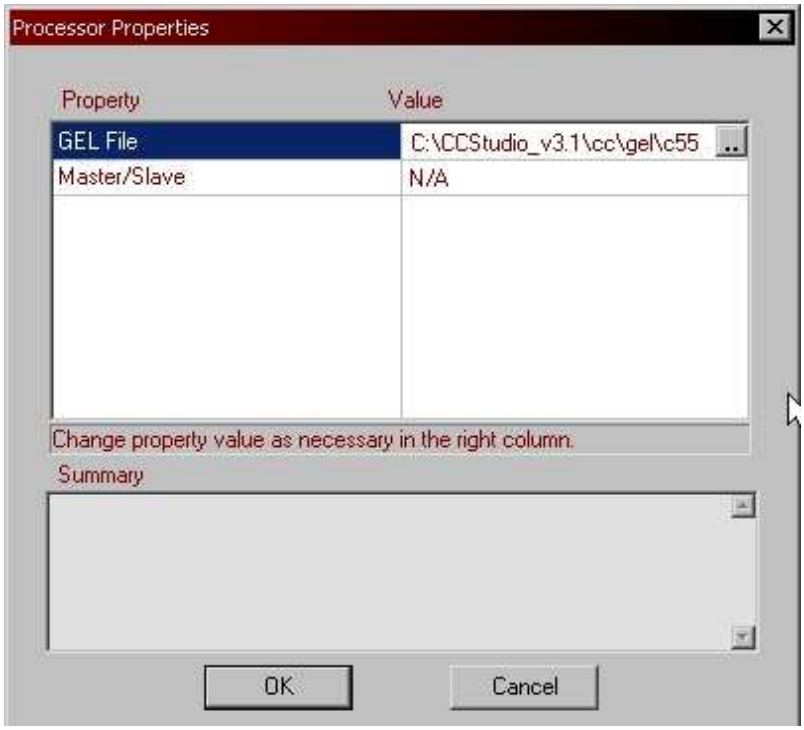

Figure 15. Defining a GEL File be Used When Opening CCS

When finished, click the Finish button, and you will see the board defined showing the emulation connection hardware and target processor in the left window pane, and specific details of the target connection in the right window pane (Figure 15).

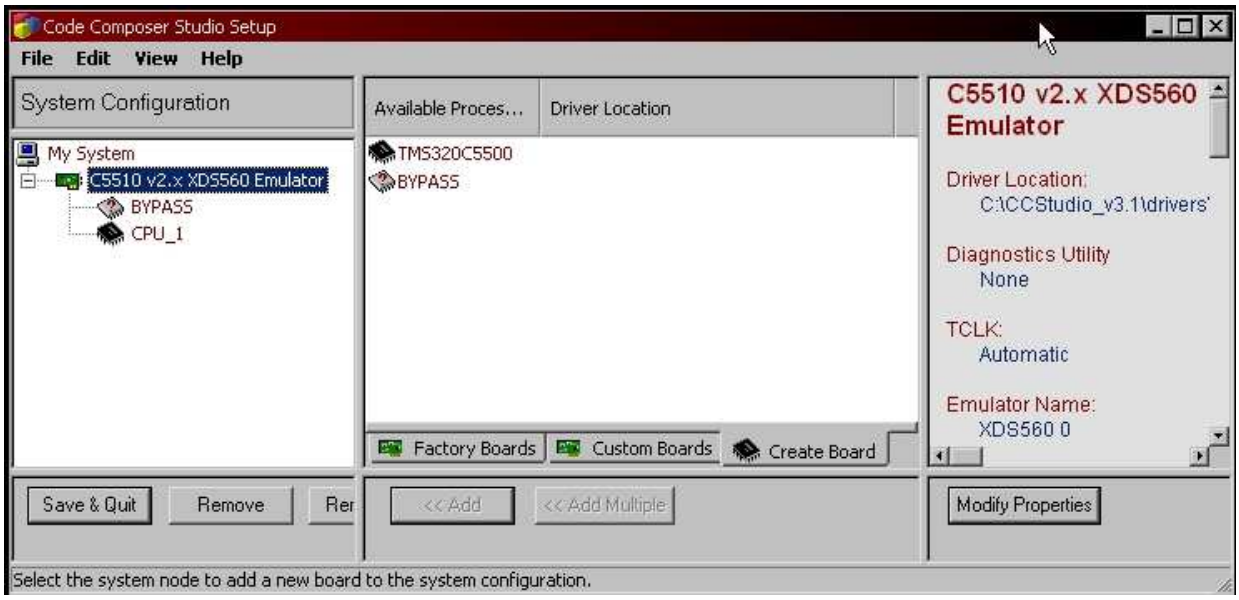

Figure 16. Viewing the System Configuration within CC_Setup

2.3 Multiprocessor Debug

There are multiple methods of connecting scan chains to your emulator/header depending on which emulator/emulation pod configuration is used. A basic rule to remember is that all data must be scanned serially through all devices as shown in Figure 17 to Figure 19.

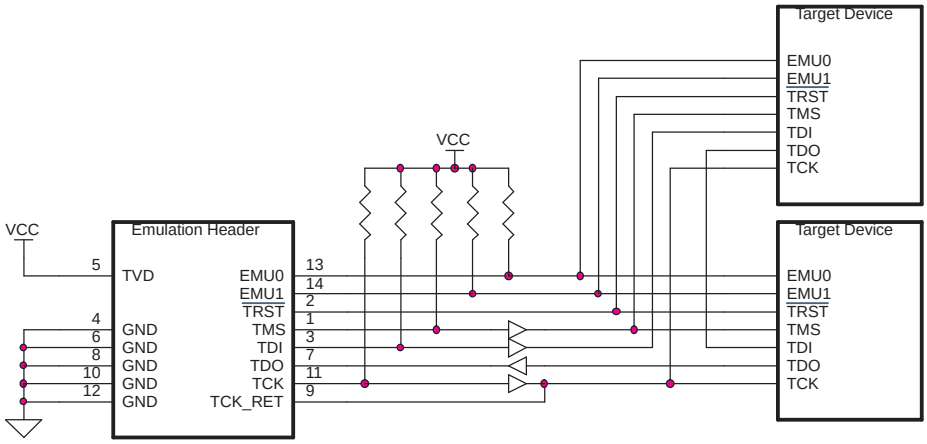

Figure 17. Emulator Connections for Multiprocessor Systems

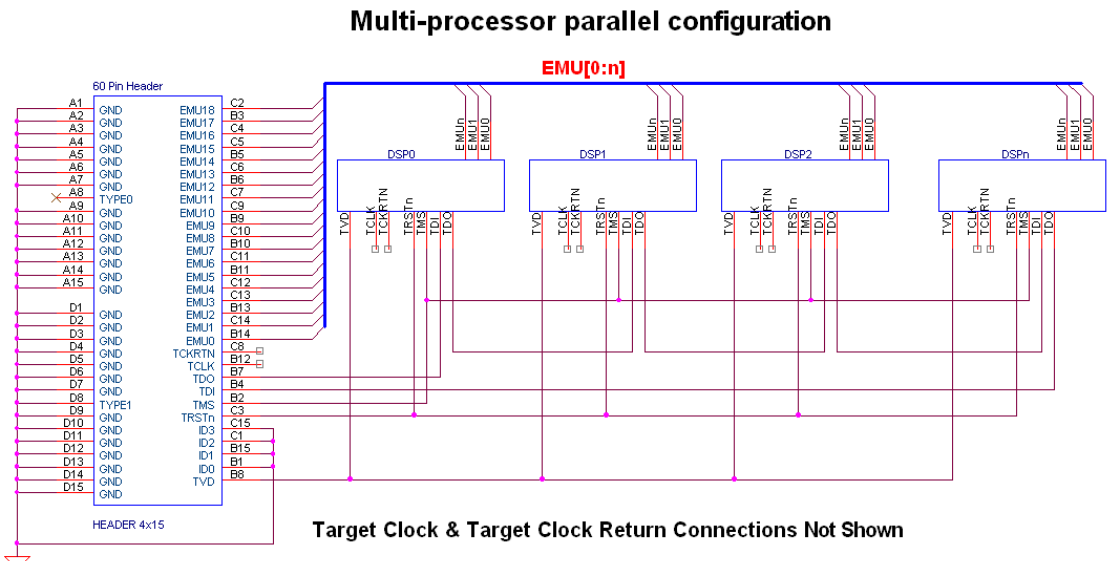

Figure 18. Emulator Connections (Parallel Config)

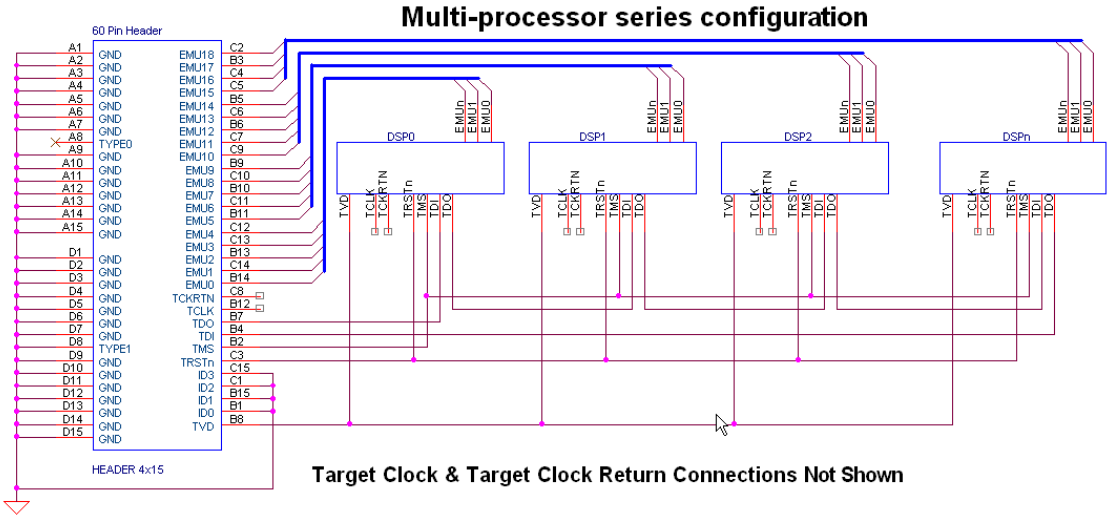

Figure 19. Emulator Connections (Serial Config)

As discussed in Section 1.3, the scan manager controls the bits that are scanned into and out of the scan chain such that the information is directed to the correct target processor, as well as data coming out being directed to the proper debugger session. For this reason, it is imperative that the scan manager know exactly which devices are in the scan chain and in what order they appear. Devices should be entered in the order in which they are physically located within the JTAG scan chain (from TDI to TDO).

To begin, start from the Create Board tab of CCS_Setup (Figure 20). Then, simply drag the emulator connection from those available in the center pane, and drop it into the left pane. This time, give it a unique target Name (Figure 21) and define the connection.

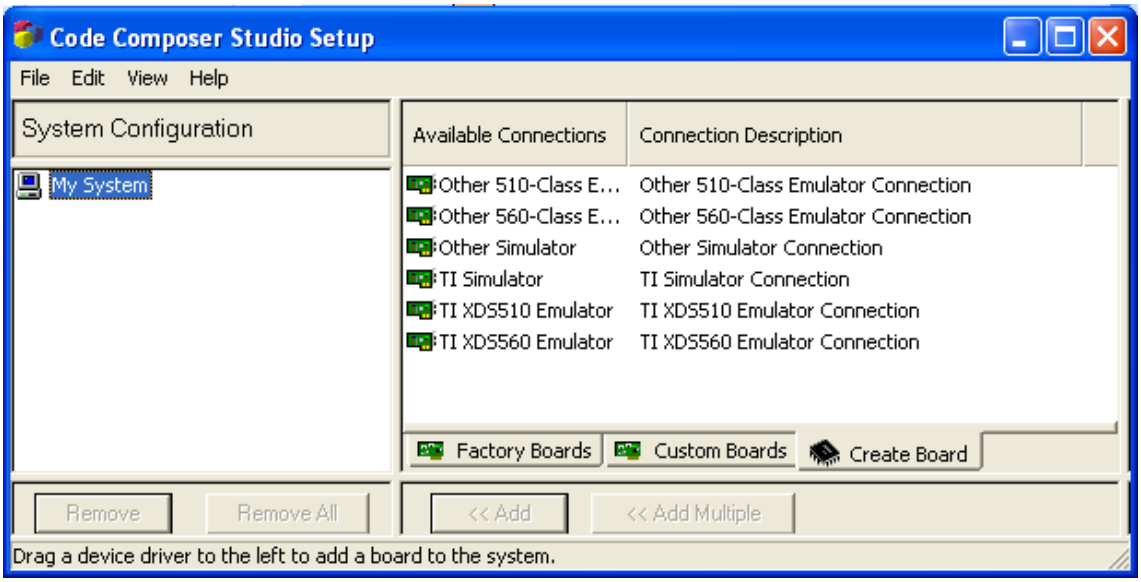

Figure 20. Creating a Board Configuration

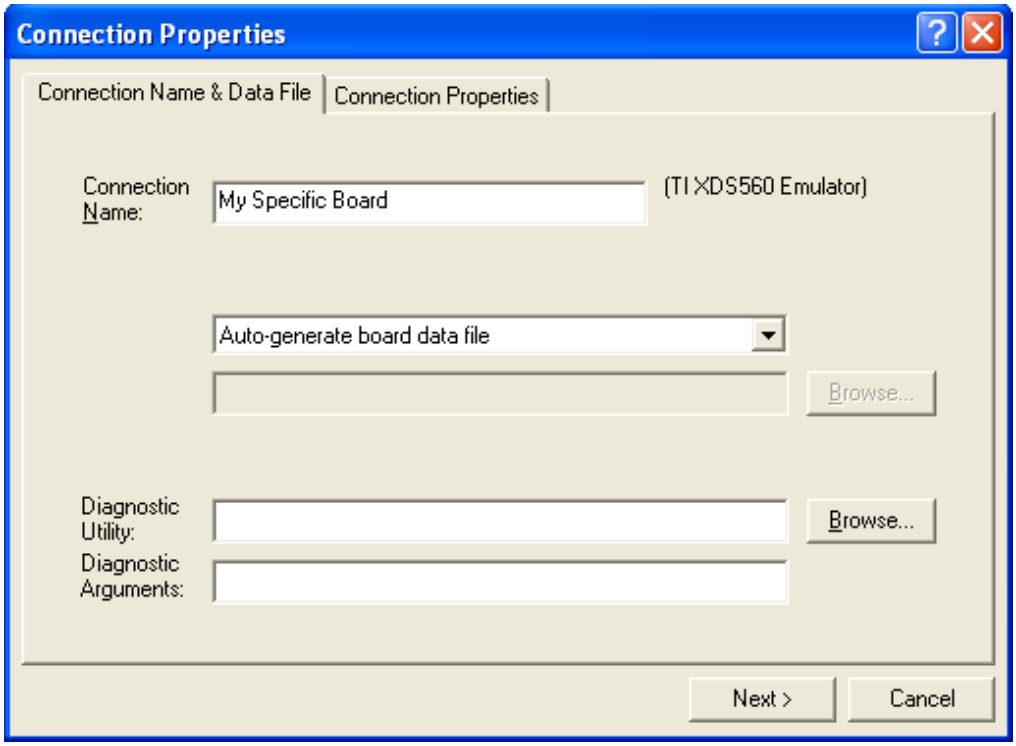

Figure 21. Setting up the Connection to the Target Board

Now, drag a specific target processor from the center pane into the left pane (Figure 21). You will get a dialog box showing the Device Properties.

2.3.1 Homogeneous Multiprocessor Debug

When multiple identical processors are used in the same scan chain, it is referred to as homogeneous multiprocessor debug. In this type of debug, the devices must have the same TI definition for that processor, but each processor must have a unique name. So, if you intend to use multiple devices from the same processor family, they each require a unique name. You can accept the default suffix appended to the processor name (Figure 23), or you can name them yourself, (such as the function that it serves).

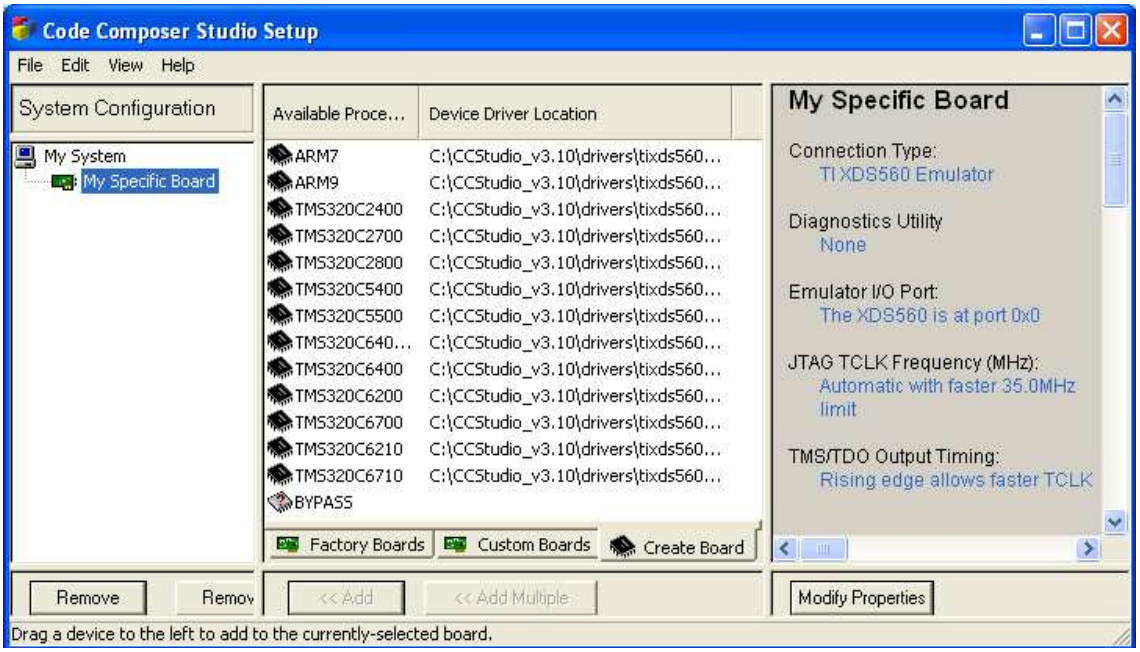

Figure 22. Selecting Available Processors When Creating a Board Setup

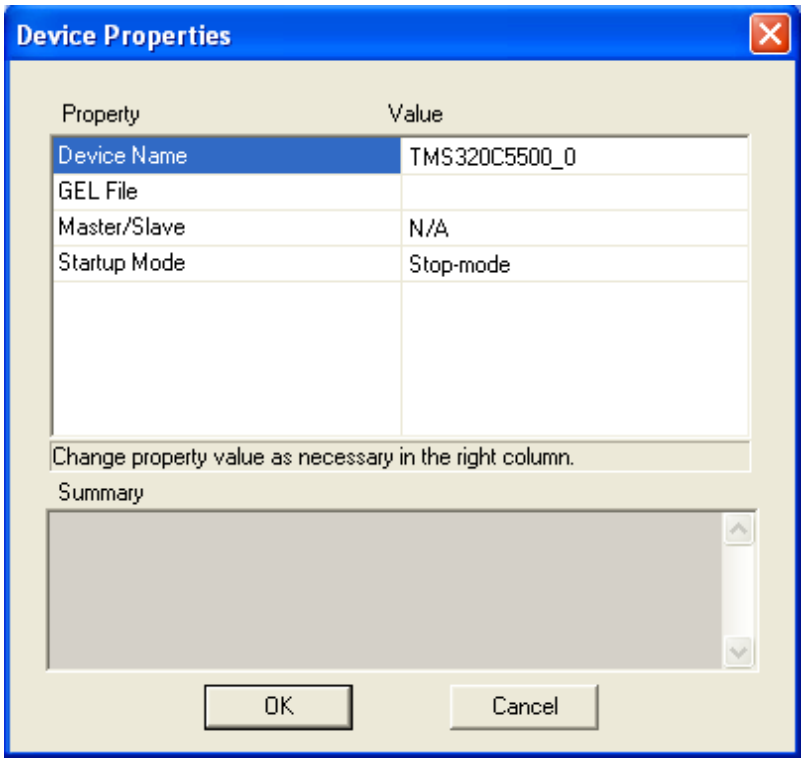

Figure 23. Naming Each Processor in a Board Setup

Repeat this process for each processor in the system, until your board is fully defined. Figure 23 shows a board defined with multiple C55x digital signal processors, using the default processor name.

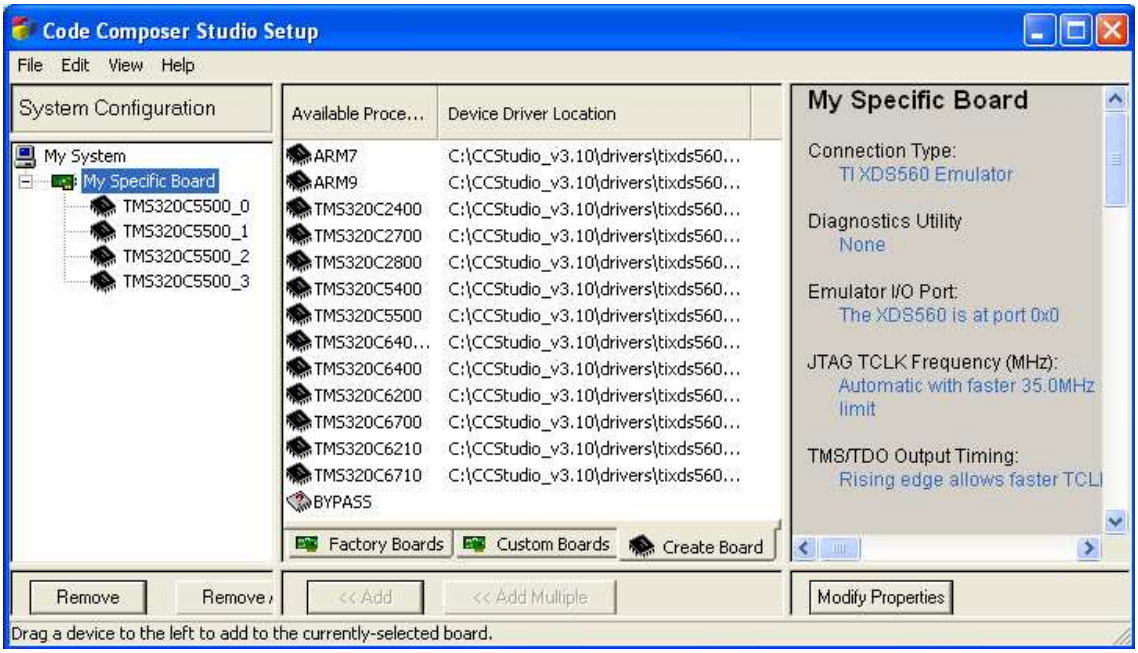

Figure 24. Homogeneous Multiprocessor Definition

2.3.2 Heterogeneous Multiprocessor Debug

When multiple processors are used in the same scan chain, but are from different processor families, it is referred to as heterogeneous multiprocessor debug. When setting up the system for heterogeneous debug, the specific CPU target that you want to debug must be included within the Available Processor Type list in CCSetup (Figure 25).

If the TI processors used in your heterogeneous processor system are not listed as being supported with any of the available drivers, please contact Texas Instruments Technical Support for more assistance, as you may need an update to Code Composer Studio IDE to support your specific target CPU.

Again, you can accept the default name, or you can create a unique name for it. Figure 25 shows a system defined with a TMS320C6200 DSP (serving as an imaging engine) and an ARM9 microcontroller (serving as a general-purpose controller), in which the processors are named according to their function.

Figure 25. Heterogeneous Multiprocessor Definition

2.4 Scanning Through Non-Debug JTAG Devices

It is also possible to scan information through the JTAG ports of devices that are not being debugged via the TI emulator. This condition arises if a customer wants to perform boundary scan on devices within their target board. Boundary scan is performed using the JTAG header as well, but uses different software than is used for TI's emulation.

When devices are included in the JTAG scan path, but are not being debugged via JTAG emulation, it is possible to scan the emulation information through these devices by putting them in BYPASS mode. It is first necessary to understand how large their JTAG instruction registers are.

Figure 26 shows an example of a design in which devices exist in the JTAG scan path that are not being emulated, and are placed in BYPASS mode.

Figure 26. Specifying NON-Emulation Devices in Bypass Mode

In this illustration, the bypass device, BYPASS_0, has a JTAG instruction register length of 8-bits. This information is passed on to the scan manager so that the JTAG registers can be set up to bypass through this device. Note also, that the first chip in the scan chain (in this case, Boundry Scan Dev) also shows up with a "?", indicating that the emulation software knows nothing of the device other than its JTAG instruction register length.

2.5 Multiprocessor Debug on Devices Which Provide TCK_RET as an Output

Special considerations must be made regarding the JTAG clock when one of the devices in the scan chain provides TCK_RET as an output. As mentioned earlier, ARM Ltd varies from the IEEE-1149.1 specification, and makes the JTAG clock dependent on the CPU clock. While this is done for legitimate design reasons, it does place an extra burden on the debug considerations, and requires special care in routing the JTAG clock.

Within the CPU clock dependency, ARM Ltd gates the TCK pin going into the device, to restrict its speed with respect to the CPU. As a result, for every clock edge going into the device on TCK pin, there may not be a corresponding TCK_RET signal coming out of the device. Likewise, the data being scanned out of the ARM core does not change on every TCK cycle, but rather on every TCK_RET cycle.

At a high level, the TCK_RET signal from the target devices which provide it, will always be a clock at or below the speed of TCK. If the TCK pin were to be supplied from the JTAG header to all devices, but the TDO and TDI lines were still connected in series, the rate at which each device was scanning data would not match the clock. Because of this, the TCK_RET signal must then be used as an input to all other devices in the scan chain, to ensure that the data rate through the scan chain matches the clock.

You can think of this as a queue, in which data may not get clocked on every input clock cycle (TCK), but may be halted occasionally by the ARM core, which may use multiple TCK cycles to scan a single bit. In this queue, you must then stall all of the other devices, to ensure that they sample data only when new data is available. The entire scan chain must slow down to the rate of its slowest member.

With these considerations, you can then clock data through all of the devices on the board, but must connect the JTAG clocks as shown in Figure 27. In this case, the TCK_RET signal gets used as the master clock for scanning data through all of the devices in the scan chain.

Note: On such systems, if the CPU clock is not present on the ARM core that generates TCK_RET, there will be no TCK_RET, and all JTAG activity will halt - likely resulting in a debugger error. This can restrict debug options when attempting to test Idle, Sleep, and other low-power modes, which may require the ARM core's clock to be removed.

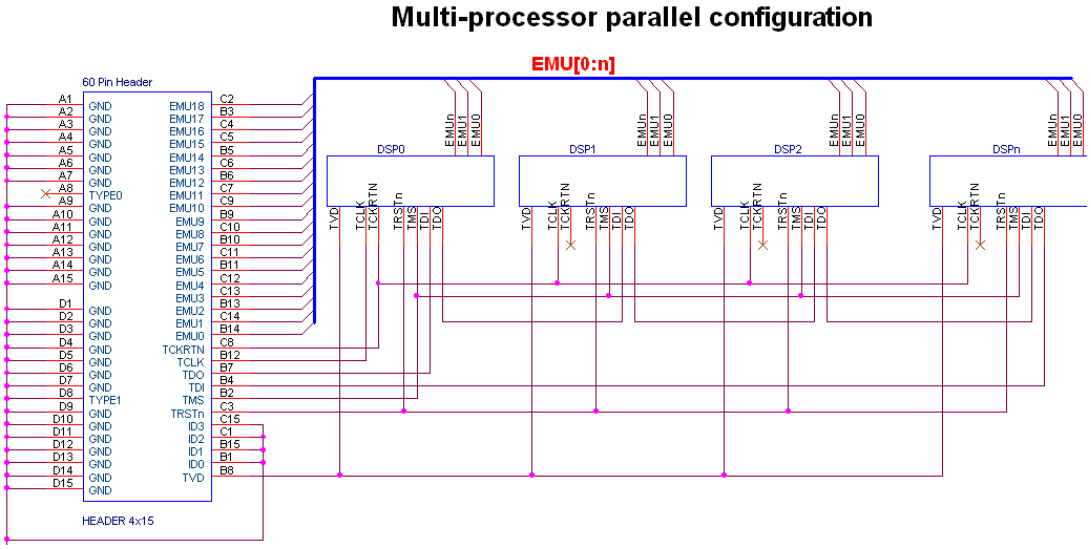

Figure 27. Emulator Connections for Systems in which TCK_RET is supplied by the Target Device

If you are using a system with multiple devices, in which more than one device provides a TCK_RET signal, then special considerations must be made to accommodate the clocking requirements. In this case, you cannot allow the TCK_RET from more than one device to be the master of the scan chain. So we must connect external “voting logic” circuitry (Figure 27)to act as a clock arbitrator.

3 Establishing Communication with Your Emulation Hardware and Target

Once you have installed your emulator hardware and set up your board configuration, it is time to establish communication with the hardware from your host computer. This section will explain some of the files used to confirm connectivity of the hardware.

3.1 Hardware Setup

Before executing any files, the following steps are important:

- Set up your emulator according to the directions that were included in shipping.
- Connect the JTAG header from the emulator pod to the target.
- Connect the emulator pod to your host computer by the appropriate means (PCI card, ISA card, USB Port, etc.), depending on which emulator you are using

3.2 Physical I/O Space of Your Emulation Hardware

The next step in establishing communication is to be sure to confirm the physical address of the emulator on your host computer.

- The XDS560 installation makes use of the plug-n-play capabilities in most PCs today. With this capability, the Code Composer Studio software is able to detect the address of the hardware without having to explicitly define the address.
- If you are using an ISA slot in a PC, the XDS510 will require 32 bytes of IO space on the PC, and comes shipped with a default setting of 0x0240 – 0x025F. If this space is already taken within your PC, you can set the board to be addressed elsewhere in the I/O space (0x0280, 0x0320, 0x0340)

- The XDS510 ISA card will require changing switches on the hardware if the default location is not available on the host computer.

3.3 Invoking Emulation Reset

You are now able to invoke the emulation-reset software. This software resets the internal emulation debugging logic as well as several other functions. Depending on which debugging software you are using, the emulation-reset software may have a different name. The program that ships with Code Composer Studio is XDSReset.exe, while the older debuggers included the software as EMURST.exe. Among the functions performed by the emulation-reset software are:

1. Check for the correct I/O address of the emulator hardware
2. Verify that there is no debugger currently running in multi-processor mode
3. Reset the Test Bus Controller
4. Check if power exists on the Target Voltage Detect (or Presence Detect) pin. If not, pull TRST_ high and generate error message “CANNOT DETECT TARGET POWER”. If TVD has power, then pull TRST_ low and then high.
5. Put the device in Test Logic Reset (TLR).
6. Check if the device is in TLR. If not, create the error message “XDS510 RESET FAILED.”

In most cases, this setup goes smoothly, and you are now ready to begin debugging your software by starting the emulator

3.4 Invoking the Emulation Debugger

When invoking the emulation debugger, you simply need to double-click the Code Composer Studio icon – which should execute CC_App.exe. There are several parameters that can be passed when invoking CC_App, including specification of a desired GEL-file, and other options. But most of these options can be specified from within CC_Setup, as explained earlier. Please refer to the User’s Guide that shipped with your debugging software for more information on these options.

3.5 Using the Parallel Debug Manager

When debugging systems have more than one TI device defined, you will be required to use the Parallel Debug Manager (PDM), which provides a method of synchronous debugging of your multiprocessor application. If you have configured a multiprocessor system within CC_Setup, the PDM will automatically be invoked when you start CC_App.

After the PDM is invoked, you can spawn emulator sessions from within the PDM environment similar to the way they are invoked from the icon in a single-processor system configuration (Figure 28). The PDM has provisions to allow for grouping the processors within the JTAG scan path to provide debugging commands to a select set of processors, without all of the processors being affected.

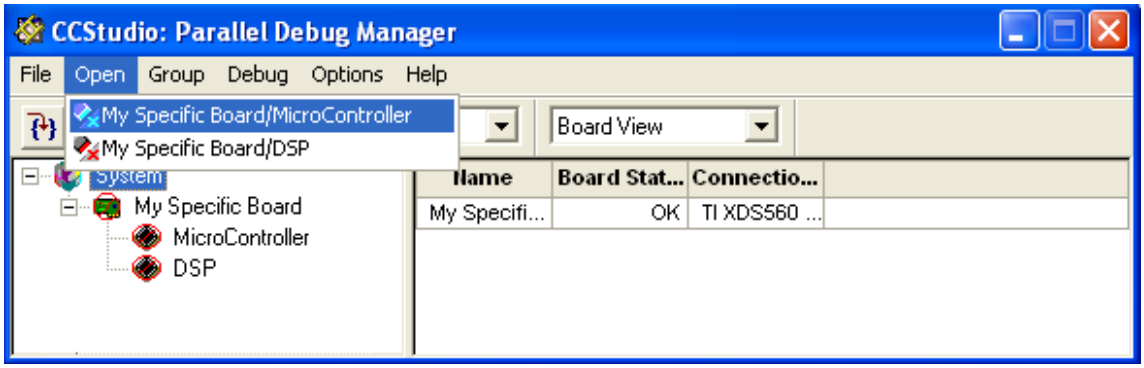

Figure 28. Opening Debug Sessions from the Parallel Debug Manager

The PDM will also allow single-point control that can start and stop several processors synchronously. Synchronous debug includes the ability to synchronously start, stop, and step the processors (Figure 29). It also includes the ability to perform global breakpoints through the use of the EMU signals as detailed in Section 1.4.

Figure 29. Parallel Debug Manager

Additional details on the Parallel Debug Manager are available in the C Source Debugger User's Guide for the specific processor you are using.

4 Troubleshooting Emulation Setup Errors

This section will provide solutions to emulation setup errors, as well as provide a chart to verify connectivity.

4.1 Emulation Reset Errors

If you run into errors when invoking the emulation reset software, first check the set up of the physical connection to the host computer and make sure that power is supplied to the target. Also make sure that the signals on the board are buffered if the device is more that 6 inches from the JTAG header, as shown in Figure 4.

You may also want to inspect the TCK_RET signal using an oscilloscope, which will provide more detail regarding the overall signal quality. A weak signal on TCK_RET may indicate that buffers are needed in your system to accommodate long paths. The absence of TCK_RET or TVD indicates a connectivity problem within your setup.

Before invoking the emulator:

- Note the settings on your physical XDS510 board regarding the I/O space that it will be addressed
- Set the I/O space address accordingly within Code Composer Setup
- If you are using a parallel port debugger, make sure that you set the I/O address correctly and that the parallel port on your PC is set up to support the mode which you intend to use it (i.e. SPP4, SPP8, ECP). Please refer to the manufacturer's instructions on parallel port setup.

Errors in the emulation hardware address will result in the error message: "CANNOT DETECT TARGET POWER". Make sure to inspect that you have set up the XDS510 correctly before looking for errors on your target.

When resetting an XDS560 board, there are several errors that can be displayed. This hardware checks connectivity to the target board at several locations including:

- The emulation board within the PC
- The board's connection to the JTAG cable
- The cable's connection to the target

Errors in the above connections will result in the messages as covered in Table 6 as well as thoroughly detailed in the XDS560 Troubleshooting Guide.

4.2 Emulator Errors

Occasionally, a target will pass emulation-reset, but yield errors when invoking the emulator. We will address some of these errors and possible setup conditions that might yield the errors.

Among the most common problems encountered when starting the emulator are simply system configuration errors. System configuration errors will cause a variety of errors. For example, if the devices that are defined within CC_Setup are not the same, or even in the wrong order as those on the physical target, you may see data errors. These errors may include all zeros, all ones (or F's if viewing HEX data), or repetitive bit patterns throughout registers and memory displays.

The emulator is still sometimes able to start with an incorrect board configuration. However, it will not scan out the correct information. This is due to the fact that the scan manager is distributing the information coming out of the TDO pin to the corresponding debugger window. If the device ordering in the system configuration is incorrect, the bit-stream coming out of TDO will be incorrectly broken up and the debugger window will not get the correct information for the target device you are intending to debug.

If it appears that the board configuration is correct within CC_Setup and the bits being displayed in the emulator are all zeros or all ones, the physical connection of the JTAG signals should be investigated. A solder short across a JTAG header can cause signals to be shorted and give erroneous information at the TDO pin. Make sure to test each of the JTAG signals described in Table 1, Table 2, and Table 3.

On some of the newer TI devices, you will see documentation referring to signals called ET0, ET1, and TVD. These signals are identical to the traditional signals called EMU0, EMU1, and PD, respectively. However, there names have been changed to better reflect their use within TI's emulation solutions.

Presence detect (PD) has been re-named target voltage detect to indicate that, not only is it used to detect the presence of a target being connected to the JTAG header, but it is also used to detect the level of the voltage on the JTAG signals for that device. On the newer emulation hardware, the JTAG pod is able to adapt the voltage to match the voltage of the target. This provides a means of interfacing with legacy devices, which operated at a higher voltage (up to 5V) as well as the newer devices, whose JTAG signals are capable of operating down to the CPU voltage of 0.8V.

Monitor each signal using an oscilloscope to determine if it is at the appropriate level, or if it is changing as it should be. (WARNING: Selection of scope and scope probe may impact your measurements, due to probe capacitance).

Table 6 shows the standard signals included on the JTAG header that should be investigated and what level should be expected at each pin, depending on the mode of operation.

Table 6. JTAG Signal Activity for Connectivity Verification

JTAG Signals	Level	Conditions
TVD (PD)	Hi	Vcc or I/O voltage (if different than CPU) of the device if target board has power
TCK	Both	10.368 MHz square wave clock signal coming from the JTAG pod on the XDS510. Variable clocks possible on XDS560, and other third party emulation HW.
TCK_RET	Both	Derived from TCK. If this signal is dirty or attenuated, buffers should be used on the JTAG signals as shown in Figure 4. If this signal is not present, make sure that TCK is present. If this signal is not present on devices that provide TCK_RET from the target, make sure that the cores on the device are powered, clocked and released from reset.
TMS	Both	Controls the state machine, so it should change every time another debug operation is performed
	Low	When the device is in Run Test/Idle mode
	Hi	When the device is set in Test Logic Reset state
TRST		Active-Low signal: Emulation logic should be out of reset to perform emulation. NOTE that if this signal is not pulled down internally on the specific device you are using, this signal should have a pull-down resistor on it to ensure that the emulation logic is RESET when the emulation hardware is not attached.

Table 6. JTAG Signal Activity for Connectivity Verification (continued)

JTAG Signals	Level	Conditions
TDI	Both	This signal is the information that is being scanned into the target from the emulator pod. It should change levels as data is scanned through the target
TDO	Both	This signal is reading internal information from the target and should be changing to reflect the data that is being scanned out
GND	Low	This signal should be clean and zero volts
EMU[n]	Hi	At startup, these signals are typically HI, as they are used to indicate global breakpoints as well as being used for benchmarking and profiling. NOTE that if these signals are not pulled up internally on the specific device you are using, you should place pull-up resistors on them to ensure that they are inactive when the emulation hardware is not attached. On some devices, they can affect the operational mode of the device if they are not pulled up.
	Low	If the device has hit a global breakpoint which has caused a CPU halt for debug, the EMU signals may still be low to indicate that the break conditions have been met.
	Both	If the device is actively being debugged and you are performing such functions as Benchmarking, Profiling, or RTDX, the EMU signals may be toggling to transmit data to the emulation hardware board.

By viewing the signals shown above on an oscilloscope, you can determine if the signals are changing as they should during normal emulation and emulation start-up. If you cannot get your emulator started due to errors, make sure to monitor these signals at the time of invoking the emulator to determine if there is ever any activity on them.

5 Troubleshooting Guide

In the event that your emulator tools cause errors that you do not understand, we have provided a table allowing you to determine some of the commonly found errors in the emulation setup. Look in the first column to determine the error you are seeing and then track down the possible errors listed in column two.

Table 7. Troubleshooting Guide for Emulation Errors

Problem	Possible Solution
Error Message: “CANNOT DETECT TARGET POWER”	• Ensure that the target has proper voltage supplied to it • Check that the emulator board is securely installed • Check that the cabling is securely connected between the emulator and the target • Make sure that the port address of the emulator hardware is set correctly • Make sure that the emulator hardware setting for hte port address correspond to that entered within CC_Setup • Make sure you don't have another I/O device conflicting for the same I/O space on yoru host computer
Error Message: “CANNOT INITIALIZE THE TARGET!!”	• Make sure that the target has proper voltage supplied to it • Make sure the processor is not in RESET • Make sure that the port address of the emulator hardware is set correctly (as explained abive) • check that the cabling is securely connected between the emulator and the target • Make sure you have correctly entered the board information within CC_Setup • Run the Emulation Reset software before invoking the emulator to verify that emulation logic is in proper state
Error Message: “Processor access timeout”	• External device may be holding the hardware HOLD signal • Processor may be waiting for READY signal from external peripheral

Table 7. Troubleshooting Guide for Emulation Errors (continued)

Problem	Possible Solution
Data in the debugger screen displays all zeros or all F's	• Check the solder connections on the JTAG header for short circuits. Check the board schematics to insure proper routing of the JTAG signals • Make sure you are using the correct board configuration within CC_Setup
Data in the debugger screen displays repetitive bit-patterns	• Make sure you are using the correct board configuration file
Data in the debugger screen displays random bit-patterns where specific data is expected	• Check for dirty signals on the JTAG header • Check memory map definition with that in the debugger initialization file to ensure that there is actually memory present
Error Condition: Code Composer Studio halts at splash screen. (Windows 98 only)	• Check that the emulator board is securely installed. • Check that the cabling is securely connected between the emulator and the target.
Error Message: Can not Initialize Target CPU: SC_ERR_CTL_CBL_BREAK_NEAR <-182>	• The cable pod is not connected or is improperly connected at the back of the PC. Verify that cable is connected and that the pod connector's screws are tightened into the XDS560 PCI bracket.
Error Message: Can not Initialize Target CPU: SC_ERR_CTL_CBL_BREAK_FAR <-183>	• The target header from the XDS560 pod is not connected or is improperly connected to a target board. • The cable pod is not connected or is improperly connected at the back of the PC. Verify that cable is connected and that the pod connector's screws are tightened into the XDS560 PCI bracket. " On some PENTEK boards, the target emulation header uses the "Target Disconnect" pin of the XDS560. Verify that pin 4 of the target board emulation header is grounded. If not, then add a wire-wrap wire from pin 4 to pin 8 of the target emulation header. Refer the XDS560 Technical Reference Manual for additional information.
Error Message: Can not Initialize Target CPU: SC_ERR_NO_TRG_POWER <-180>	• Make sure that the target has proper voltage supplied to it. • Check that the emulator board is securely installed • Check that the cabling is securely connected between the emulator and the target. • Make sure that the XDS560 hardware settings for the port address

6 Other Debugging Diagnostic Tools

XDSPROBE .exe is a utility that ships with Code Composer Studio. It is capable of exercising the JTAG scan path to test for connectivity to the target device as well as determining if device[s] match what is defined within the board configuration file from CC_Setup.

6.1 XDS Probe

c\There are far more capabilities to this tool which are defined within the on-line documentation within Code Composer Studio. The tool can be found in the \ti\cc\bin directory. The help manual can be viewed by typing "xdsprobe -vh" in a DOS shell at the command prompt while in the..\ti\cc\bin directory. XDSProbe is currently being shipped with CCS as a legacy utility and will not be updated in future releases

6.2 DBGJTAG

DBGJTAG.exe is another utility that ships with newer versions of Code Composer Studio. DBGJTAG is a similar utility to XDSProbe but also includes additional functionality and features. DBGJTAG will ultimately replace XDSProbe. The help manual can be viewed by typing "dbgjtag -H manual" in a DOS shell at the command prompt while in the..\ti\cc\bin directory.

6.3 Third-Party Tools

Texas Instruments has an extensive Third Party network that develops tools for the TI DSP Solutions. Occasionally, these companies will create their own tools to aid in verifying connectivity of their emulation tools. These tools can be used as additional aids to what is described in this document. Support of these tools as well as questions regarding their use should be directed through the third party.

6.4 Bulletinboard Tools

Texas Instruments also occasionally develops special tools to perform specific tasks and places them on the TI Electronic Bulletin Board for free access. These tools can be downloaded and used at the user's discretion. These tools are provided for free, and as such, are not supported. Among these tools on the Bulletin Board is a tool called XDS_DIAG.EXE. This tool is an older tool, which performs similar functions to XDSPROBE.

6.5 Future Tools Development

Texas Instruments is dedicated to the development of industry leading development and debug tools. As such, TI is continually developing new capabilities, which allow customers increased flexibility and visibility in software development. Updates to these and other tools may be available upon occasion via the Update Advisor within Code Composer Studio. Be sure to check for updates occasionally. Keep in touch with your local distributor and/or TI representative to stay abreast of the newest tools and technology available. You can also monitor the Bulletin Board and Internet Homepage for DSP Solutions to find new capabilities as TI announces additional developments

7 References

1. *Testability Primer* (SSYA002)
2. *JTAG/MPSD Emulation Technical Reference* (SPDU079)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265