Emulation Fundamentals for TI's DSP Solutions

APPLICATION REPORT: SPRA439

Charles W. Brokish
Member, Group Technical Staff
Field Design and Applications
March 1998

TEXAS INSTRUMENTS

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 application 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

Contents

Figures

Tables

Emulation Fundamentals for TI's DSP Solutions

Abstract

In software development, perhaps the most critical, yet least predictable stage in the process is debugging. Many factors come into play when debugging software applications. Among these factors, time is of utmost importance. The TI emulator is useful when integrating the hardware and software portions of the user's application. 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 which captures the market share.

Texas Instruments' debuggers allow extensive visibility into the processors, their registers, and the application's software. This visibility allows the software engineer to understand what changes are taking place inside of the processor as the application executes. The software engineer can set breakpoints in the application based on hardware signal values or software locations within the application. At these breakpoints, the user can understand the state of the processor and data and determine if their application is still operating correctly. They can also perform benchmarking(timing analysis) and profiling(CPU loading) of their application software within the emulator.

Additionally, multi-processor debug can allow the user to debug software on several processors at the same time and provide a method of stopping one or multiple processors based on a condition set in another processor – allowing the user to capture the entire system state at the time in question.

The capabilities mentioned, and many more within the Texas Instruments debuggers can greatly reduce debugging time in the software development cycle.

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. This knowledge will also aid in trouble shooting potential problems in your debugging setup.

This paper will explain the setup of the XDS510 systems for single and multi-processor applications. It will also explain how the system components interact during debug. A trouble-shooting guide will be provided to assist in commonly found set-up problems.

1. Hardware Setup

In order to understand the workings of the emulation logic on the TI processors, it is first necessary to understand what emulation consists of. This section will cover setup of the hardware into your host operating system.

1.1 Introduction

When using the TI debugger for software debugging on a hardware platform, it is necessary to perform a few setup procedures to ensure proper working of the target processor with the XDS510.

The emulation setup is comprised of two tools - the XDS510(emulator) 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. The emulation logic within the TI processors uses the JTAG (Joint Test Action Group) standard connection, in procuring the debug information from within the processor.

This section is intended to cover the basics of ensuring proper communication setup between the emulation hardware and the target processor.

1.2 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[1]. It is important to understand that this debug method is near real-time, but is intrusive, as it requires that the core be halted to scan the information.

[1] IEEE Std 1149.1 (JTAG) Testability Primer, TI Literature #SSYA002C, 1997

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. Internally to the processor, there are various serial scan chains in which the information can be scanned into and out of. The control of which scan chain is used and what information is contained in each scan chain, is performed by a microprocessor. This “scan manager” has the task of controlling this information as it is scanned to and from the various processors in the scan chain and directing it to and from the various debugger windows.

The host of the XDS510 acts as the scan manager as it controls the delivery of the scan information to and from the target and the debugger window. If the operating system is a PC (Personal Computer) and the JTAG connection is being made through an ISA card (Figure 1) or XDS510PP (Figure 2), the host of the XDS510 is the microprocessor in the PC (i.e., the Pentium® processor).

However, if the JTAG connection is through another hardware interface or if the operating system is a UNIX workstation (Figure 3), the JTAG connection utilizes a separate processor to control the scan information. Whether the host CPU, or a separate processor control the JTAG scan information, they need to be supplied with information regarding the devices included in the scan chain.

Figure 1. Connecting the XDS510 ISA card to Your Target System

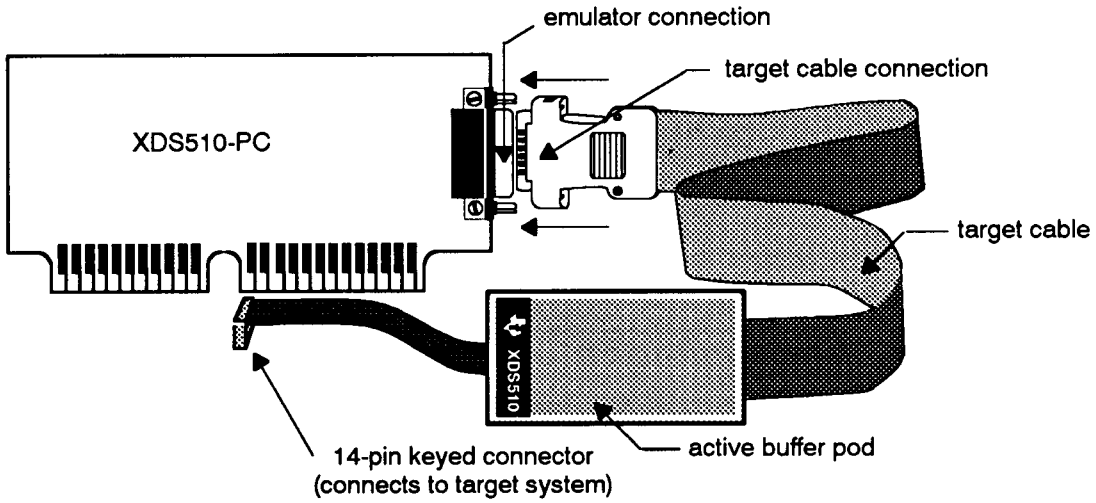


Figure 2. Connecting the XDS510-PP to Your Target System

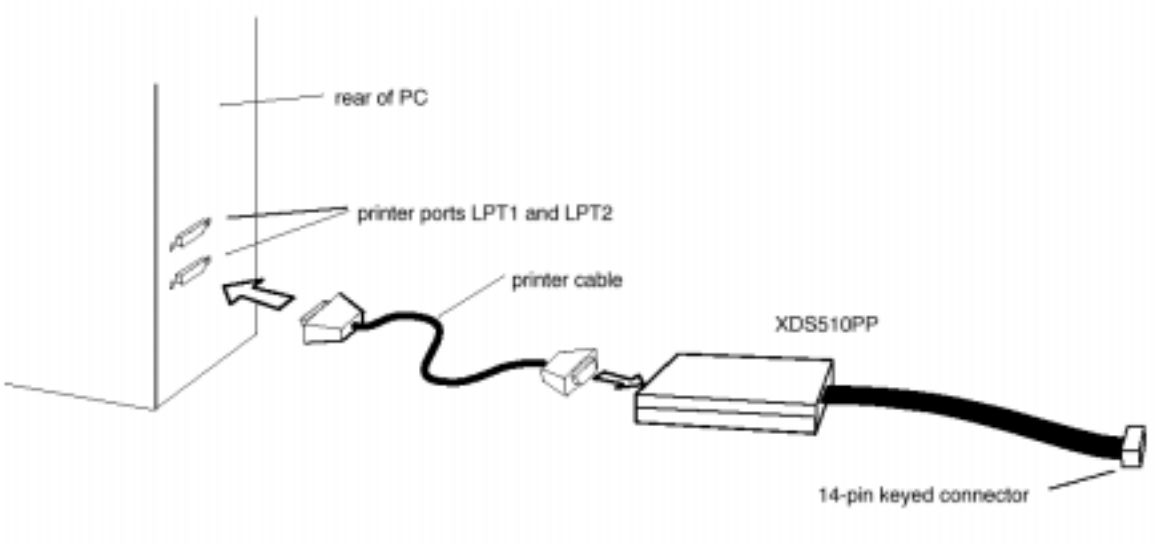
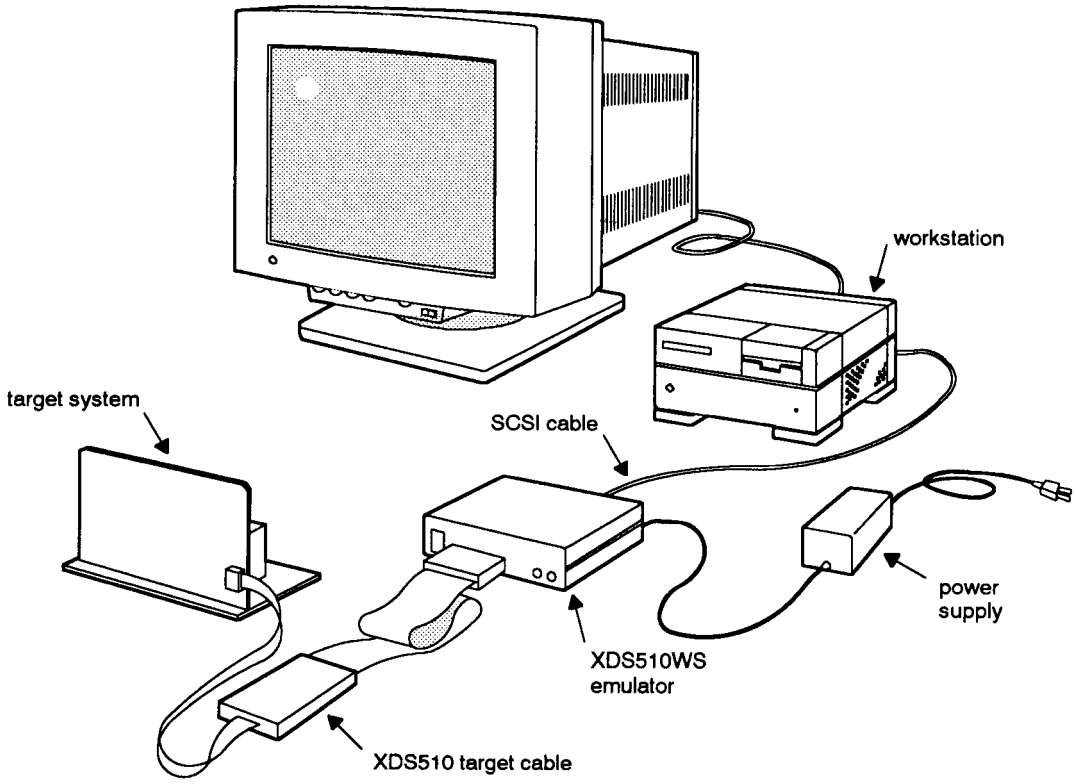


Figure 3. Connecting the XDS510-WS to Your Target System



This information regarding the scan control is available to the PC host via Dynamically Linked Libraries, otherwise known as DLL files. This information is passed to the XDS510-WS in the form of a separate file when performing emulation reset. We'll discuss this utility more in section 3.3.

Additional information regarding physical connection of the XDS510 debugger interfaces is available in the documentation that is shipped with the product. Other helpful information[2, 3] can be found at other locations within Texas Instruments, such as the TI Bulletin Board and the Internet Homepage.

TI third parties are making other XDS510 interfaces available, including PCMCIA and Ethernet connections[4]. In these connections, scan management is performed by the CPU of the host computer and an external processor, respectively.

1.3 Signal Descriptions

There are five standard signals and 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. 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 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 pod. We will discuss their functionality further when we discuss EMURST and trouble-shooting hardware problems.

Table 1. Standard TAP Controller JTAG Signals

Signal Name	Description	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 (10 MHz clock provided by the XDS510 pod)	O	I
GND	Ground		

* I = Input, O = Output

[2] XDS51x Emulator Installation Guide, TI Literature #SPNU070A, 1996
[3] JTAG/MPSD Emulation Technical Reference, TI Literature #SPDU079A, 1994
[4] White Mountain DSP, MTN-510/LT & Trek-510, http://www.wmdsp.com/

Table 2. Connection Verification JTAG Signals

Signal Name	Description	Emulator Signal Type*	Target Signal Type*
TCK_RET	Test Clock Return (Buffered or un-buffered connection of TCK coming back from the target)	I	O
PD(V_{cc})	Presence Detect	I	O

* I = Input, O = Output

Additionally, 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	Emulation Pin 0	I	I/O
EMU1	Emulation Pin 1	I	I/O

* I = Input, O = Output

The use of the EMU signals is detailed in your C-Source Debugger User's Guide. They are used by the XDS510 to perform clocking capabilities when performing software benchmarking and software profiling. They are also used with the ANALYSIS window of the debugger to assist in multi-processor debugging.

The EMU signals are 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 4 demonstrates the common connection of these signals on the user's hardware design. If the processor is 6 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 5).

Figure 4. Emulator Connections Without Signal Buffering

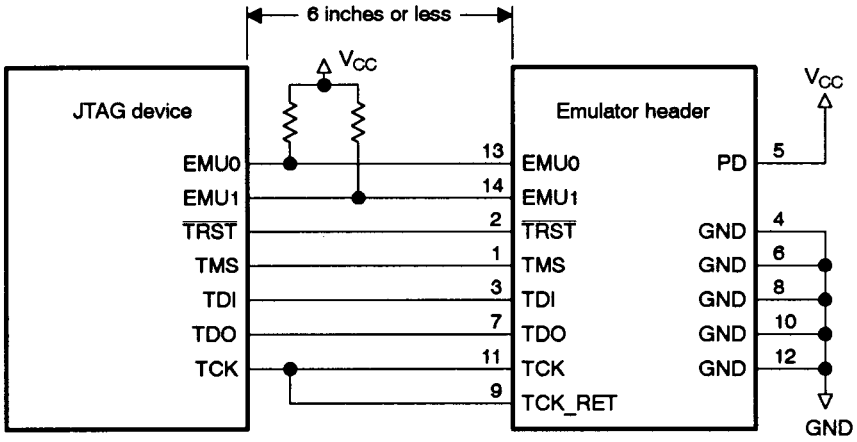
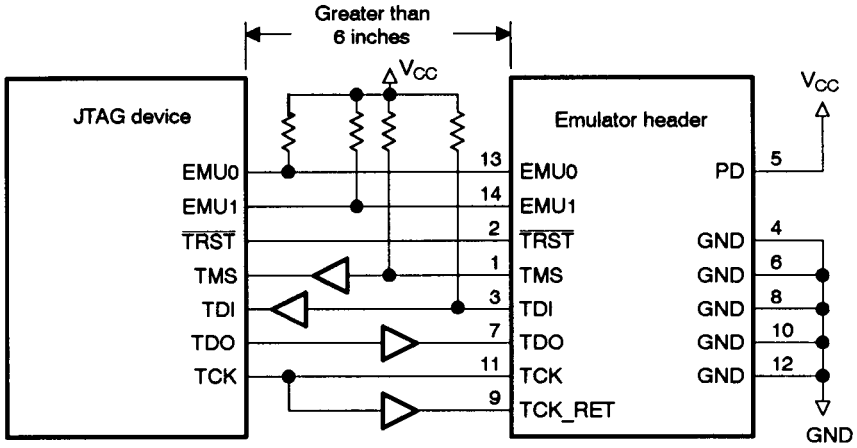


Figure 5. Emulator Connections With Signal Buffering



2. Software Setup

2.1 Single Processor Debug

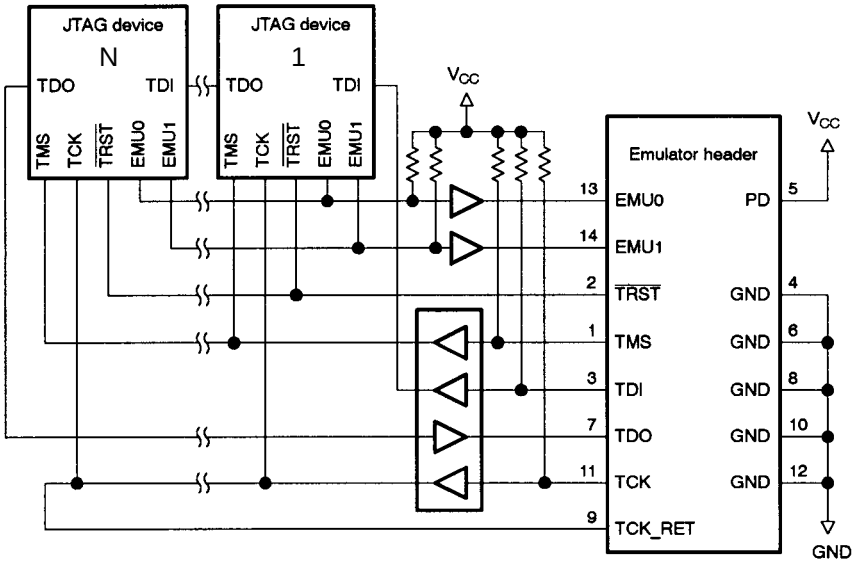
If only a single device is included in the scan chain, the signals are connected as shown in Figure 4 and Figure 5. Most of the TI target-specific software debuggers (Emulators) have traditionally had an executable with a name such as EMU5XX.EXE (TMS320C54x Emulator). This name indicates that it is an Emulator and the family of the target device, such as the C5xx family. Also becoming more common are filenames such as EMU5XXW.EXE, in which the "W" indicates that it is a Windows program. It still has the same appearance, but is able to be run as a Windows application rather than as a DOS application.

Additionally, executable filenames can take on the appearance of EMU5XXWM.EXE, in which the "M" indicates that multiple sessions can be run concurrently for multi-processor applications. We will discuss this further in the next section.

2.2 Multi-Processor Debug

When multiple devices are included in the scan chain, the data needs to be scanned serially through all of the devices and the signals are connected as shown in Figure 6.

Figure 6. Emulator Connections for Multi-Processor Systems



As discussed in section 1.2, 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 what order they are in.

2.3 Board Configuration Information

The information regarding the scan chain is entered in a file called BOARD.CFG. This file is a text file containing a user-defined name of the processor and the TI-defined name for that processor. The user-defined name can be anything that the user wants to call the device such as “dsp1” or “vocoder”. While the TI-defined name for that processor is very specific, such as TI320C5xx, as shown in Figure 7.

Figure 7. Sample BOARD.CFG files for Single Processor Systems

BOARD.CFG

```
“dsp1”    TI320C5xx
```

- or -

BOARD.CFG

```
“vocoder”    TI320C5xx
```

When multiple processors of the same type are used in the same scan chain, each processor must have a unique user-defined name, but must have the same TI-defined name for that processor as shown in Figure 8.

Figure 8. Sample BOARD.CFG files for Multi-Processor Systems

BOARD.CFG

```
“dsp1”    TI320C5xx
“dsp2”    TI320C5xx
```

- or -

BOARD.CFG

```
“vocoder”    TI320C5xx
“modem”      TI320C5xx
```

It should be noted that the order in which the devices are listed in the BOARD.CFG file is the **opposite** order of which they are physically connected in the scan chain. This is due to the fact that the emulator views them in the order the bits come out of the scan chain.

If one views the scan chain as a serial shift register, the **first** bits to come out of the scan chain are those of the **last** device in the chain. Therefore, if there were two TMS320C54x devices in the scan chain shown in Figure 6, and the BOARD.CFG file used was that shown on the right side of Figure 8, the “modem” device would be located first on the JTAG scan chain (device #1). Note that the “first” device in this case is the device that is connected to TDI of the JTAG header. TDO of the first device is then connected to TDI of the next, and so on for subsequent devices.

When invoking the emulator, it is necessary to express **which** processor you wish to debug. (This is *not required* when performing single-processor debug, but is good practice.) The method used to distinguish which device is being debugged is the –n parameter. This parameter is used as follows…

```
EMU5XX –n dsp2  <other options>
```

Note that the name of the device is the user-defined name, but is not in quotes.

Similarly, when performing emulation on systems that have multiple processors, and are not all of the same family, the information for all of the devices is contained in a single BOARD.CFG file. Figure 9 shows an example configuration file used for a system with a TMS320C54x DSP and ARM micro-controller in the same JTAG scan chain.

Figure 9. Sample BOARD.CFG file for Multi-Processor System with Separate Processor Families

BOARD.CFG

```
“DSP”    TI320C5xx
“Micro”     TI470R1x
```

The information contained in the BOARD.CFG file is in ASCII format and must be converted into a binary format for use by the XDS510 scan manager. The tool used to perform this modification is called COMPOSER.EXE. COMPOSER is shipped with the emulator software.

To run COMPOSER, you can simply run it without any additional parameters. When run in this manner, it expects to find a file called BOARD.CFG in the same directory and creates an output file called BOARD.DAT.

Alternately, you can run this program with a different input filename by including that name as a run-time parameter. The input file must still contain the same information as described in this section of the document.

You can also specify a different output filename when using COMPOSER as well. But it should be noted that this will require using the –f <filename> option when invoking the emulation tools.

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 emulated, 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 10 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 10. Sample BOARD.CFG file for Non-Emulation devices in BYPASS mode

BOARD.CFG

```
"BSD"     BYPASS04
"DSP"     TI320C5xx
"Micro"   TI470R1x
"other"   BYPASS08
```

In this case, the bypass devices have a JTAG Instruction Register which are 4-bits and 8-bits long. This information is passed on to the scan manager by appending the length of the IR to the word "BYPASS" when defining the target-specific device name.

3. Establishing Communication With Your XDS510

Once you have installed your XDS510 and set up your board configuration file, 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, make sure that your XDS510 is set up according to the directions that were included in shipping. Make sure that the 14-pin JTAG header is connected from the XDS510 pod to the target and that the XDS510 pod is connected to your host computer by the appropriate means(ISA card, Parallel Port, SCSI bus, etc), depending on which XDS510 emulator you are using.

3.2 Parameters and Environment Variables

Be sure to confirm the physical address of the XDS510 on your host computer. 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)[5].

Similarly, if you are using an XDS510-PP, the default address will be 0x0378. And the default SCSI address on a XDS510-WS is address 4. The XDS510 ISA card and XDS510-WS will require changing switches on the hardware if the default location is not available on the host computer.

When invoking the various emulator executables, it will be necessary to pass the address of the XDS510 as a parameter if it is not the default setting. On the PC, this can be accomplished by using the -p parameter with the appropriate address. For example,

EMU5XX –p280 <additional parameters>

…indicates that the XDS510 is located at I/O space 0x0280.

This information can also be passed in the form of environment variables. For example, if the address being used is not the default, you can set the new address as an environment variable using the “SET D_OPTIONS” command in your AUTOEXEC.BAT file.

[5] XDS51x Emulator Installation Guide, TI Literature #SPNU070A, 1996

```
SET D_OPTIONS=-p280
```

If you do not intend to use D_OPTIONS, it is advised that before attempting to establish communication with your XDS510 hardware, you make sure that D_OPTIONS is not already set as an environment variable.

You may also bypass an environment variable settings by using the –x option when invoking the emulation tools.

3.3 Invoking EMURST

You are now able to invoke EMURST.EXE. EMURST is an executable file which resets the internal emulation debugging logic as well as several other functions. An example invocation of the emulation reset tool is:

```
EMURST –p 280
```

Among the functions performed by EMURST are:

1) Check if parameters have been passed (-p / -x options)
2) Check for the correct I/O address of the XDS510
3) Verify that there is no debugger currently running in multi-processor mode (OS/2)
4) Reset the Test Bus Controller
5) Check if power exists on the Presence Detect pin. If not, take TRST_ high and generate error message “CANNOT DETECT TARGET POWER”. If PD, then take TRST_ low and then high.
6) Put the device in Test Logic Reset(TLR).
7) Check if the device is in TLR. If not, create the error message “XDS510 RESET FAILED”
8) If external processor is performing the scan management (XDS510-WS), then download the specified .OUT file for the scan manager.

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

3.4 Invoking the Emulator

When invoking the emulator, you simply need to enter the name of the executable and any parameters that need to be set. Table 4 shows a list of the parameters that can optionally be included in the command line when invoking the emulator. Please refer to the C Source Debugger User's Guide for your processor for additional information on the debugger options.

Table 4. Summary of Emulation Debugger Options

Option	Description
-b[b]	Select the screen size as 80 characters by 43 lines or 80 characters by 50 lines.
-bw, -bl	Set the screen size to user-defined size (not available on all platforms)
-f *filename*	Identify a specific board configuration file
-i *pathname*	Specify additional directories
-n *processor name*	Specify which processor is be debugged
-p *port address*	Identify the port address of the XDS510
-profile	Enter the debugger in profiling mode
-s	Load the symbol table only (used when the code already exists in ROM)
-t *filename*	Indicate a specific Emulator Initialization file
-v	Load code without the symbol table
-x	Ignore D_OPTIONS environment variable

A typical command line to invoke the emulator might look like…

EMU5XXW –p 280 –n modem –t MODM_INI.CMD

If using the BOARD.CFG from Figure 8, you would have to run COMPOSER to obtain the resulting BOARD.DAT. The command line shown would imply that the XDS510 was located at address 0x0280 of the computer's I/O space, that we want to use MODM_INI.CMD as the emulator initialization file, and that we want to debug the DSP performing modem functions.

3.5 Invoking Multiple Emulator Sessions

Similarly, we can invoke multiple emulator sessions to debug several processors on the scan chain at the same time. We need to make sure that the emulator executable that we are using supports multiple instantiations. The multiple sessions could be started as follows…

EMU5XXWM –p 280 –n modem –t MODM_INI.CMD

…followed by

EMU5XXWM –p 280 –n vocoder –t VOC_INI.CMD

This would start two emulator sessions in parallel – each allowing **independent** visibility into the application code running on the separate processors. These emulator sessions are not tied together in software and either processor can be started and stopped independently. And the user can quit and restart emulator sessions independently.

The information scanned into and out of the single JTAG scan path gets separated and directed to and from the separate processors and debugger sessions by the scan management software and the XDS510 hardware.

3.6 Using the Parallel Debug Manager

At times it may be advantageous or necessary to debug systems simultaneously. In this case you may need the systems to be able to start and stop synchronously. The Parallel Debug Manager provides a method of synchronous debugging of your multiprocessor application.

After invoking the PDM, you can spawn emulator sessions from within the PDM environment similar to the way they are invoked from a command line. This will allow single-point control of starting and stopping of several processors synchronously. PDM has provisions to allow for grouping of the processors within the JTAG scan path to provide debugging commands to a select set of processors without all of the processors being affected.

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. Trouble-Shooting Emulation Set-up Errors

4.1 EMURST Errors

If you have run into errors when invoking EMURST, 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 5.

You may also want to inspect the TCK_RET signal on an oscilloscope. This will indicate the integrity of the signals on the board in terms of cleanliness as well as magnitude. 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 PD indicates a connectivity problem within your setup.

As discussed in section 3.2, before invoking the emulator you should…

- Make sure to note the settings on your physical XDS510 board regarding the I/O space that it will be addressed.
    - Make sure to set any environment variables accordingly, if you choose.
    - Or, make sure to clear (or ignore using the –x option) any pre-existing environment variables if you do not plan to use them.
- If you do not use environment variables regarding the physical address of the XDS510, make sure to include the address as a command-line option when invoking the emulator using the –p parameter.

Errors in the XDS510 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.

4.2 Emulator Errors

Occasionally, a target will pass EMURST, 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 environment variables or board configuration file errors. Environment variable errors were discussed in section 4.1, but will also cause errors when invoking the emulator if not corrected.

Board configuration file errors will cause a variety of errors. Among these are all zeros, all ones (or F's if viewing HEX data), or repetitive bit patterns throughout registers and memory displays.

The emulator is still typically able to start with an incorrect board configuration file. 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 information in the BOARD.DAT file 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.

When investigating possible errors in the board configuration, make sure that COMPOSER has been run on the BOARD.CFG file. Also, make sure that the corresponding BOARD.DAT file is in the source directory or the startup directory of the emulator.

If it appears that the board configuration file is correct, 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 and Table 2. Monitor each signal on an oscilloscope to determine if it is at the appropriate level or it is changing as it should.

Table 5 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 5. JTAG Signal Activity for Connectivity Verification

JTAG Signal	Level	Conditions
PD	Hi	V_{cc} if target board has power
TCK	Both	10.368 MHz square wave clock signal coming from the JTAG pod
TCK_RET	Both	Duplicate of TCK. If this signal is dirty or attenuated, buffers should be used on the JTAG signals as shown in Figure 5
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	Hi	Active-Low signal: Emulation logic should be out of reset to perform emulation
TDI	Both	This signal is the information that is being scanned into the target from the XDS510 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

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. Trouble-Shooting 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 6. Trouble-Shooting Guide for Emulation Errors

Problem	Possible Solution

Error Message: "CANNOT DETECT TARGET POWER"	• 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 port address of the XDS510 is set correctly
	  • Make sure that the XDS510 hardware settings for the port address correspond to those used when invoking the emulator software
	  • Make sure that a preset D_OPTIONS environment variable is not conflicting with desired port address
	  • Make sure you don't have another I/O device conflicting for the same I/O space on your 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 XDS510 is set correctly(as explained above) • Check that the cabling is securely connected between the emulator and the target • Make sure you are using the correct board configuration file • Run EMURST 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
Error Message: Cannot find file: board.dat	• Make sure that the **board.dat** file is in the operating directory or the executable source directory
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 file
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 • Make sure that the memory map is turned on in the debugger

6. Other Debugging Diagnostic Tools

6.1 Third Party Tools

Texas Instruments has an extensive Third Party network that develops tools for the TI DSP Solutions[6]. Occasionally, these companies will create their own tools to aid in verifying connectivity of the XDS510 for emulation. 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.

6.2 Bulletin Board 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 can be a very effective tool in confirming connectivity of the XDS510, confirming the devices present on the JTAG scan chain, and verifying the correctness of the board configuration file (BOARD.DAT).

The XDS_DIAG utility performs the following functions:

1) Checks the D_OPTIONS environment variable and reports settings
2) Detects previous Power Loss conditions
3) Toggles TRST
4) Resets Scan Controller
5) Checks and reports the Scan Controller revision
6) Checks the scan configuration file (BOARD.DAT) and displays information about the expected devices on the scan path (such as device ID, part number, Manufacturer ID, etc.). It also reports if no BOARD.DAT file was found.
7) Verifies the scan path integrity
8) Verify the core revisions for the various devices on the scan path

[6] http://www.ti.com/sc/docs/dsps/develop/3party.htm

9) Continuously scan a pattern through the scan path, allowing the user to hook up a logic analyzer or oscilloscope to verify the TDI and TDO signals.

The XDS_DIAG utility can be found on the TI Bulletin Board, and can be accessed at the following address via a web-browser:

ftp://ftp.ti.com/pub/tms320bbs/dsputils/

6.3 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.

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 to find new capabilities as TI announces additional developments.