Application Report
SPRA758A - May 2002

Using xdsprobe with the XDS560 and XDS510

Roland Hoar and Michael Dunn *Software Development Systems*

ABSTRACT

This application report familiarizes the reader with the xdsprobe utility. This utility may be used to troubleshoot Code Composer Studio™ 2.00 and 2.10 initialization problems that normally manifest themselves as an error message that indicates that the target DSP cannot be initialized. This problem may be caused by incorrect configuration or hardware problems in the JTAG scan path. The xdsprobe command line utility may be used to analyse, verify, and solve JTAG scan path problems in single processor or multiprocessor targets. This utility may be used with four types of emulation controller: TI's XDS510 and XDS560 emulation controllers; those third party emulation controllers or DSP boards that were developed using TI's 510-class Sourceless-EPK; the third party 560-class emulation controllers or DSP boards that were developed using TI's Source-EPK. Some of the common hardware and software problems that may be exposed are broken or unreliable JTAG scan paths, lack of EMU0/1 pull-up resistors, incorrect link delay parameters, circuit board layout errors and schematic mistakes and erroneous configuration files.

Contents

Figures

Introduction

There are situations where CCS (Code Composer Studio 2.00 and 2.10) is unable to start-up because it is unable to successfully communicate with the target DSP or ARM micro-controller. If unsuccessful it will display one of several messages referring to the inability of CCS to initialise the device. This failure may be due to communication problems related to the Test Bus Controller (TBC) or the JTAG scan-path that provide the physical connection between CCS and the device. The command line utility named xdsprobe that is provided with CCS may be used to test communication between CCS and the TBC, and also to test the integrity of the JTAG scan-path between the TBC and target DSP or ARM micro-controller.

The xdsprobe utility is located in the <root>:\ti\cc\bin directory of the disk-drive on which CCS is installed. It can be used with TI emulators (XDS510 or XDS560), TI DSP boards (C6711DSK or C5402DSK or C5510EVM), and the many third party products (both emulators and DSP boards) that were developed using TI's Sourceless EPK for the SN74ACT8990 TBC device.

The xdsprobe utility was originally developed as an engineering tool to facilitate the development of the physical connection between the TBC and the target DSP's and ARM micro-controllers. The feature set of xdsprobe has been much expanded to support the development of the emulators themselves and is also being extended into testing communication within target DSP's and ARM micro-controllers. It remains an engineering tool for expert users and some of the options may not be useful or necessary in a field environment.

The xdsprobe utility is actually one of a set of four utilities—the others are xdsbuild, xdserror and xdsreset. All have built-in manuals or user guides that are output by using the `–v –h' pair of command line options. The surprisingly large users guide for xdsprobe is included in this. The xdsbuild companion utility also has an equally large users guide for board configuration files—it is output by using the `–v –i' pair of command-line options. The users guide for board configuration files is also included in this report.

This report covers only the basic options needed to investigate CCS configuration files, communication between CCS and the TBC, and communication over the JTAG scan-path between the TBC and TBC and target DSP or ARM micro-controller. A step by step sequence for troubleshooting will be described in the following sections. Each step should be successfully completed prior to continuing to the next step.

The version of xdsprobe used to generate the examples in this report is 3.03.12. It is the first version to support the XDS560 emulator as well as the XDS510 emulator and Sourceless EPK. It is included as part of the XDS560 update to CCS 2.1.

The Sourceless EPK for 510-class emulators and DSP boards is available as a standard product with part number EPKTMDX3240B51. It requires use of TI's SN74ACT8990 Test Bus Controller device to ensure compatibility with the standard 510-class DSP and ARM drivers provided with Code Composer Studio 2.00 and 2.10.

Background

Prior to using xdsprobe it may be useful to have a high level understanding of the relationship of the relevant software components that are included with CCS. See Figure 1, Figure 2 and Figure 3 for visual representation of the relationship of the software components in XDS510 emulators, Sourceless EPK based solutions, and XDS560 emulators.

Figure 1 shows the relationship between the xdsprobe utility software and the DSP target when the emulator is TI's XDS510 emulator supported by the standard 510-class DSP drivers provided with CCS.

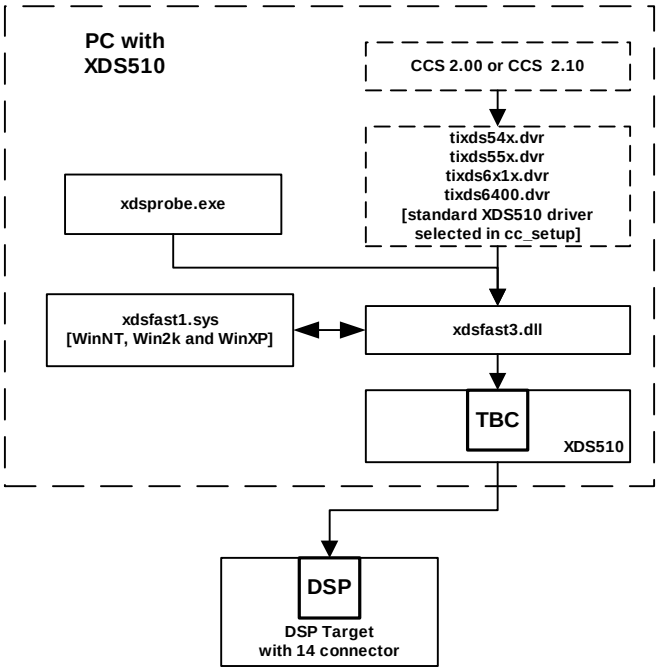

Figure 1. Block Diagram – XDS510 Emulator Configuration

Figure 2 shows TI's C6711 DSK as an example of the relationship between the xdsprobe utility software and a the DSP target when the product is developed with TI's Sourceless-EPK and supported by the standard 510-class DSP drivers provided with CCS. The authors of this report have used xdsprobe with the following Sourceless-EPK based products:

- TI's C6711 Parallel Port DSK.
- TI's C5402 Parallel Port DSK.
- TI's C5510 PCI-bus EVM.
- Blackhawk DSP's USB 1.0 and USB 2.0 Emulators.
- DSP Research's "FleXDS" PCI Emulator.
- Innovative Integration's "Code Hammer" PCI Emulator.
- Kane Computing's "Predator" PCMCIA Emulator.

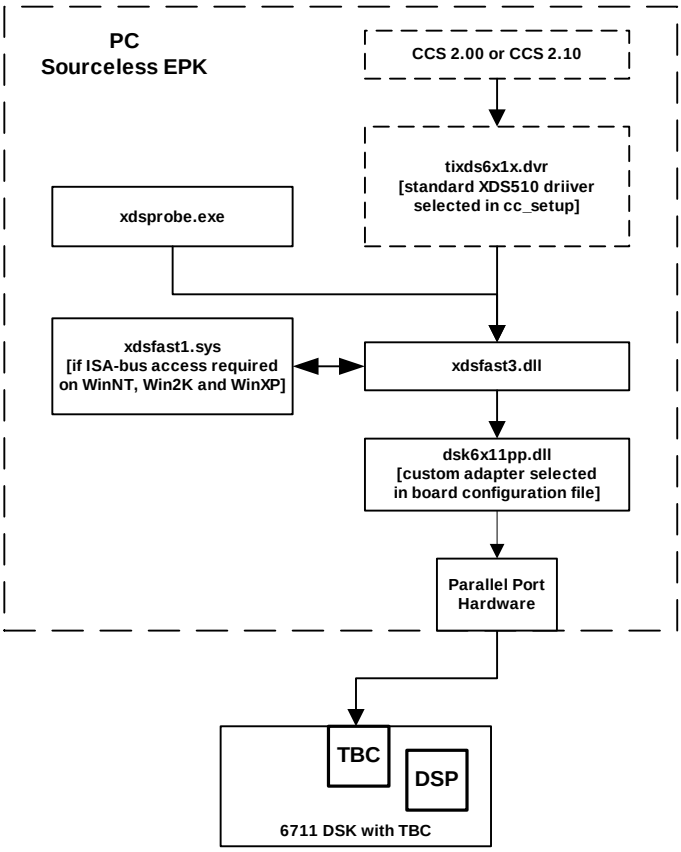

Figure 2. Block Diagram – XDS510 Sourceless-EPK Configuration

Figure 3 shows the relationship between the xdsprobe utility software and a the DSP target when the emulator is a TI XDS560 emulator supported by the standard 560-class DSP drivers provided with CCS.

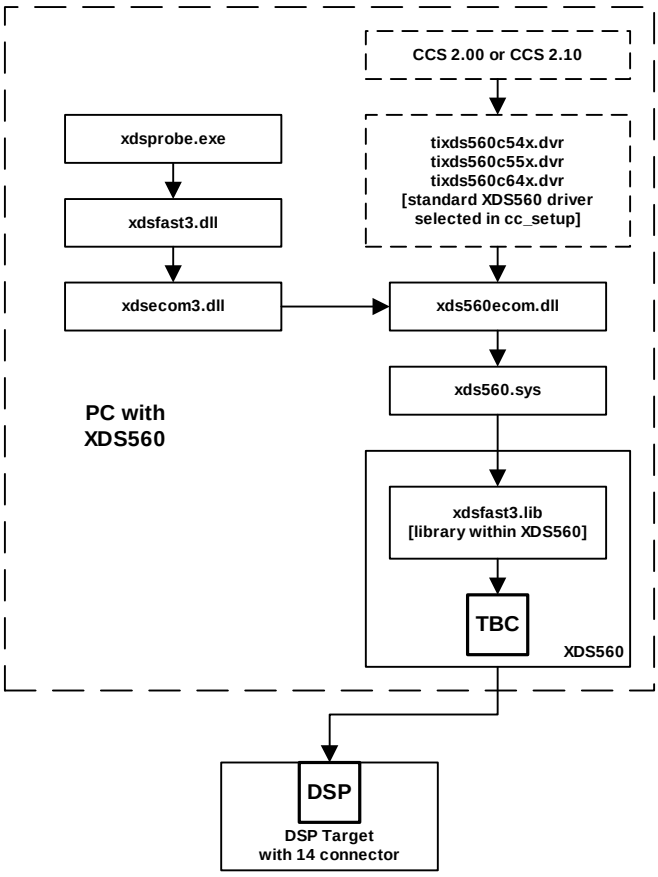

Figure 3. Block Diagram – XDS560 Emulator Configuration

Step 1 – Familiarization and Setup

Getting Help

The manual or users guide for xdsprobe is included in this report. It is generated by opening a DOS Shell or Command Prompt in the <root>:\ti\cc\bin directory of the disk-drive on which CCS is installed, and then running this xdsprobe command:

```
xdsprobe –h -v > probe-help.txt
```

The manual or users guide for board configuration files is included in this report. It is also generated by opening a DOS Shell or Command Prompt in the <root>:\ti\cc\bin directory of the disk-drive on which CCS is installed, and then running this xdsbuild command:

```
xdsbuild –i -v > board-config.txt
```

Specifying Device Address (–p)

The '–p' option is used by xdsprobe to specify the same physical or logical device address that has to be provided to CCS Setup via its Board Properties dialog. If this option is not specified then the default address 0x0 will be used, which is fine for TI's XDS510 and XDS560 emulators, but may not be appropriate for products based on the Sourceless EPK.

The XDS510 uses 4 physical addresses of 0x240, 0x280, 0x320, 0x340 depending on the position of a DIP switch on the XDS510. When xdsprobe operates an XDS510 the logical addresses 0x0, 0x1, 0x2, 0x3 are also aliased to those physical addresses.

The XDS560 simply uses logical addresses of 0x0, 0x1, 0x2, 0x3, depending on the number of XDS560's installed.

Emulators and DSP boards based on the Sourceless EPK use either physical or logical addresses depending on the product. The appropriate values will be provided in the product documentation or help files. When xdsprobe is used with TI's C6711DSK and C5402DSK the physical addresses 0x378, 0x278 and 0x3BC are commonly used depending on the configuration of the PC's parallel-ports. When xdsprobe is used with Blackhawk DSP's USB emulator then the logical addresses 0x0, 0x1, 0x2, 0x3 … are used depending on the number of emulators installed. When xdsprobe is used with DSP Research's FleXDS PCI emulator then the logical addresses 0x1, 0x2, 0x3, 0x4 are used depending on the number of emulators installed. That emulator also accepts the numerical values of XDS510 physical addresses (0x240, 0x280, 0x320, 0x340) and aliases them to its logical addresses (0x1, 0x2, 0x3, 0x4).

Using Configuration Files (–f)

CCS and its ancestor applications have always used scan-path descriptions in board configuration files, as a method of describing the type and ordering of DSP devices and ARM micro-controllers to the application. The CCS Setup application is now used to generate the board configuration file named ccBrd0.dat in the <root>:\ti\cc\bin\BrdDat directory of the disk-drive on which CCS is installed. Those scan-path descriptions are used by CCS itself and also used by the advanced options of xdsprobe.

CCS 2.00 began using configuration variables in board configuration files, as an alternative to the use of the operating system's own environment variables. Those configuration variables are used to select the adapter library used by CCS and xdsprobe when operating on Sourceless EPK based products, and used by CCS and xdsprobe to select the operation of XDS560 emulators instead of XDS510 emulators.

The board configuration file used by xdsprobe is selected by its `-f' option. If the `-f' option of is not used to select a configuration file then the name 'board.cfg' will be used. This file already exists in <root>:\ti\cc\bin directory, but is intended for use by the companion xdsreset utility to reset XDS510 emulators.

The xdsprobe utility may use the board configuration file generated by the CCS Setup application and used by CCS, or it may use a board configuration file created by the user. We will only cover enough of the concept of configuration files to use basic features of xdsprobe.

Configuration files may have either a `.cfg' or `.dat' filename extension, and are now always plain text files. The `-f' option uses the `.cfg' suffix as its default filename extension. The binary versions of board configuration files used by the ancestor applications of CCS are obsolete.

The following example of xdsprobe uses the board configuration file created by CCS Setup to reset an XDS510 or XDS560 as selected by CCS Setup, and using the default address value.

```
xdsprobe –f BrdDat\ccBrd0.dat –r
```

The following example of xdsprobe uses the board configuration file created by CCS Setup to reset whichever Sourceless EPK based product is selected by CCS Setup, but requires an appropriate address value to be provided.

```
xdsprobe–f BrdDat\ccBrd0.dat –r –p [address]
```

Figure 4 shows an example configuration file for the C6711 DSK. Figure 5 and Figure 6 show example XDS510 and XDS560 configuration files generated by CCS Setup. In all cases the lines beginning with semicolons are extra comments that have been added.

```
;CFG-2.0
; The prior line is the label required in all configuration files.

; The following line names the Sourceless EPK adapter library.
[POD_DRVR] 'dsk6x11pp.dll'
; The following line is a configuration variable for the adapter – use EPP mode.
[POD_MODE] 1
; The following line specifies the parallel-port address used by the adapter.
[POD_PORT] 0x378

; The following lines are configuration variables disabled by being commented out.
;[POD_LOG_MAIN] 'pod_main.log'
;[POD_LOG_FUNCTION] 'pod_func.log'
;[POD_LOG_ADAPTER] 'pod_adapt.log'

; The following is the CPU name.
"cpu_a" TI320C6711
```

Figure 4. Configuration File for C6711 DSK with Added Comments

```
;CFG-2.0
; The prior line is the label required in all configuration files.

; The following lines are CPU names.
"cpu_0"   TI320C671x
"cpu_1"   TI320C620x
```

Figure 5. XDS510 Configuration File for C6711+C6201 with Added Comments

```
;CFG-2.0
; The prior line is the label required in all configuration files.

; The following lines are CPU names
"cpu_0"   TI320C671x
"cpu_1"   TI320C620x

; The following line selects an XDS560 instead of an XDS510.
[UNIFY_ECOMMODE] YES

; The following two lines control the PLL generating the XDS560's TCLK.
[UNIFY_TCLK_PROGRAM]   'AUTOMATIC'
```

Figure 6. [UNIFY_TCLK_FREQUENCY] 'EXCHANGE'XDS560 Configuration File for C6711+C6201 with Added Comments

Using Configuration Files to Allow Use of Slow TCLK's

When CCS or xdsprobe are used with an XDS560 that is connected to a target system generating its own external TCLK, the XDS560 and its software will automatically detect the presence of an external TCLK, measure the TCLK frequency, and then make any necessary adjustments to their operating mode. The frequency of the external TCLK can be anywhere between 1000Hz and at least 35MHz. The user does not have to make any adjustments.

When CCS or xdsprobe are used with an XDS510 that is connected to a target system generating its own external TCLK, then if the frequency is less than half that of the built-in 10.368MHz oscillator, the user must place configuration variables in the board configuration file to ensure the XDS510 and its software make the correct adjustments to their operating mode. The adjustments enable the XDS510 and its software to operate in "slow clock mode" which supports frequencies down to 1000Hz. To enable the XDS510's slow clock operation at 80KHz the following variables must be added to the board configuration file:

```
; Enable the XDS510 slow clock operating mode.
[unify_slowclk] yes
; Select the XDS510 slow clock frequency value.
[unify_slowfrq] 80000
```

When CCS or xdsprobe are used with third party emulators (based on the Sourceless-EPK) that use slow internal or external TCLK frequencies, then configuration variables for operation with slow clock frequencies may be required, depending on the design of the product.

When CCS or xdsprobe are used with DSK's, EVM's and other products with embedded TBC's, that are based on the Sourceless-EPK then the TCLK has a fixed source and frequency. The configuration variables for operation with slow clock frequencies are not required.

Using Configuration Files to Select Programmable TCLK's

When CCS or xdsprobe are used with an XDS510 or Sourceless EPK product, it is not currently possible to program or measure the TCLK frequency. Some Sourceless EPK based products do allow manual TCLK frequency selection using jumpers or switches.

When CCS or xdsprobe are used with an XDS560 that is connected to a target system using the emulator's internal clock, the XDS560 automatically detects the absence of an external TCLK, selects a TCLK frequency based on configuration variables in the board configuration file, makes any necessary adjustments to its operating mode, and then performs tests to validate the scan-path reliability at that frequency. The frequency of that internal TCLK is based on configuration variables placed in the board configuration file either by CCS Setup when using CCS, or directly by the user when using xdsprobe. The frequency can be anywhere from 500KHz up to 50MHz (though CCS Setup restricts the upper limit to the 35MHz acceptable to HS-RTDX opertions). There are three frequency selection methods corresponding to the options available for CCS in CCS Setup:

Inserting the following configure variable in the board configuration file will instruct the XDS560 to use the "legacy" frequency" (10.368MHz):

```
[unify_tclk_program] 'legacy'
```

Inserting the following two variables in the file will instruct the instruct the XDS560 to perform an "automatic" frequency selection, with an upper limit of the maximum HS-RTDX frequency (currently 35.0MHz):

```
[unify_tclk_program]   'automatic'
[unify_tclk_frequency] 'exchange'
```

Inserting the following two variables in the file will instruct the XDS560 to apply a "user defined" frequency (such as 15.5MHz):

```
[unify_tclk_program]   'specific'
[unify_tclk_frequency] '15.5'
```

These three frequency selection methods have many other variations that are detailed in the board configuration users guide in the section detailing these specific variables.

Avoiding Use of Configuration Files (-F)

The use of board configuration files with xdsprobe can be avoided by using the new `-F' option to explicitly request that xdsprobe should communicate with an XDS510 emulator, an XDS560 emulator, or an explicitly selected Sourceless EPK based product. The `-f' option can also be used with the new `-F' option (to provide scan-path descriptions and configuration variables), but its use is not required. Also, if the `-F' option is used without the `-f' option, then the default `board.cfg' filename will not be selected.

The following shows xdsprobe resetting an XDS510 emulator using the default address value.

```
xdsprobe –F –r
```

The following is an example of xdsprobe resetting an XDS560 emulator using the default address value. The XDS560 is selected because its driver is explicitly named using the new option.

```
xdsprobe –F xds560.sys –r
```

The following examples of xdsprobe reset some of TI's own Sourceless EPK based products at the selected addresses. The product is selected because its adapter library is explicitly named using the new option.

```
xdsprobe –F dsk6x11pp.dll –r –p 0x378    ← TI's Parallel-Port C6711 DSP Starter Kit

xdsprobe –F dsk5402pp.dll –r –p 0x378  ← TI's Parallel-Port C5402 DSP Starter Kit

xdsprobe –F pci5510.dll –r –p 0x0    ← TI's PCI bus C5510 DSP Target Board
```

The following examples of xdsprobe reset a third-party Sourceless EPK based product at the selected address. The product is selected because its adapter library is explicitly named using the new option.

```
xdsprobe –F podflexds.dll –r –p 0x100  ← DSP Research's FleXDS PCI emulator

xdsprobe –F mdpjtag3.dll –r –p 0x0   ← Blackhawk DSP's USB 2.0 JTAG emulator

xdsprobe –F iipcipod.dll –r –p 0x0 ← Innovative Integration's Code Hammer PCI emulator
```

JTAG TCLK and the XDS560 versus the XDS510

The XDS510 emulator and all the currently available Sourceless EPK products have fixed or manually selected TCLK frequencies. For example, the XDS510 uses 10.386MHz, the C6711 DSK uses 25 Mhz and the C5402 DSK uses 20 MHz. The xdsprobe utility is unable to modify or measure the TCLK frequency used by the XDS510 emulator, or the currently available Sourceless EPK products.

The XDS560 emulator has both a software programmed PLL for providing an internal TCLK source, and also a software programmed multiple-mode counter for measuring external TCLK sources. The PLL can generate any frequency between 500 Khz and 50 MHz to an accuracy of 1/128 and can generate whole and half megahertz frequencies exactly. The multiple-mode counter can measure any frequency between 1000 Hz and 50 MHz to an accuracy of 1/128. The xdsprobe utility is able to fully utilise both the internal TCLK programming and external TCLK measurement features of the XDS560 emulator. It has two new options that allow the user to investigate the programming and measurement of internal and external TCLK sources.

The xdsprobe `–k' option is used to extract the record maintained by the XDS560—about how it has programmed and measured TCLK based on instructions from CCS Setup or xdsprobe via board configuration files. The following command prints the history resulting from the board configuration file received from CC Setup.

```
Xdsprobe –f BrdDat\ccBrd0.dat –r –p [address] -k
```

The xdsprobe `-d' option is used to investigate the reliability of the scan-path over a range of TCLK frequencies. The option takes a pair of comma separated sub-arguments representing frequencies between 500 KHz and 50 MHz. It performs reliability tests on the scan-path at exact TCLK frequencies, and repeatedly performs those tests using frequency increments of 1/64, so as to test TCLK over multiple power-of-two frequency ranges between the two limits. The results of these tests are output as maps of scan-test success and failure. Their analysis can provide detailed information about the reliability of the scan-paths. The following custom board configuration file and command print a map of scan-tests between 0.5 MHz and 48.0 MHz.

```
;cfg-2.0
;This custom file is named `ez-startup.cfg'.
;It helps easy xdsprobe start-up on systems with low quality scan-paths.

;This selects the XDS560.
[unify_ecommode] yes

;These request the XDS560 to use a 500KHz TCLK at start-up.
[unify_tclk_program]   'specific'
[unify_tclk_frequency] '0.5'

;This request the XDS560 to not test the scan-path at start-up.
[unify_tclk_scantest]  'nothing'
```

```
xdsprobe –f ez-startup.cfg –r –p [address] –d1,32
```

Error Reporting by the XDS560 versus the XDS510

The XDS510 emulator and all the currently available Sourceless EPK products have limited capabilities (by comparison with the XDS560) for diagnosing and reporting startup problems to xdsprobe. For example when CCS or xdsprobe operate an XDS510, they are unable to distinguish cable-break, power-loss and dead-clock errors, reporting them all as the power loss error titled `SC_ERR_CTL_NO_TRG_POWER'

The XDS560 emulator has extensive capabilities for diagnosing and reporting startup problems to xdsprobe. For example when CCS or xdsprobe operate an XDS510, they are able to distinguish emulator cable-breaks at both ends of the emulator cable, target power-loss and JTAG dead-clock errors, reporting them all as uniquely titled errors:

```
The target system has lost its power:
  SC_ERR_CTL_NO_TRG_POWER
The target system has lost its clock:
  SC_ERR_CTL_NO_TRG_CLOCK
The target cable is disconnected at the host:
  SC_ERR_CTL_CBL_BREAK_NEAR
The target cable is disconnected at the target:
  SC_ERR_CTL_CBL_BREAK_FAR
```

```
---[An error has occurred and this utility has aborted]---------

This error is generated by TI's USCIF driver.

The error's value is: `-183'.
The error's title is: `SC_ERR_CTL_CBL_BREAK_FAR'.

The error's explanation is:
The controller has detected a far-away cable break.
The user must connect the cable/pod to the target.
```

Figure 7. The XD560 Error Report When the 14-pin Header is Disconnected From the Target

In addition the XDS560 emulator also performs tests on the reliability of the scan-path as part of its extensive TCLK programming and measurement capabilities. When these tests fail it will provide uniquely titled errors to indicate that the JTAG controller cannot reliably communicate with the target DSP's and micro-controllers at that TCLK frequency:

```
The external frequency failed the scan-path test
  SC_ERR_TST_EXTERNAL
The user selected internal legacy frequency failed the scan-path test
  SC_ERR_TST_LEGACY
The user selected internal specific frequency failed the scan-path test
  SC_ERR_TST_SPECIFIC
The user selected internal auto-range frequency failed the scan-path test
  SC_ERR_TST_AUTOMATIC
```

```
---[An error has occurred and this utility has aborted]---------

This error is generated by TI's USCIF driver.

The error's value is: `-291'.
The error's title is: `SC_ERR_TST_AUTOMATIC'.

The error's explanation is:
The utility or debugger has requested the PLL generating the JTAG
clock to automatically select a frequency by using an auto-ranging
algorithm. The built-in scan-path test has repeatedly failed.
This indicates that the auto-ranging algorithm cannot find
a frequency that will allow the JTAG controller to reliably
communicate with the target DSP's and micro-controllers.
Try using the legacy frequency or a known good specific frequency.
```

Figure 8. The XD560 Error Report When the Automatic Frequency Selection Fails.

Step 2 – Applying an Emulation Reset (–r)

You can generate an emulation reset using the `–r' option of xdsprobe. The effect of this option is exactly the same as obtained by the companion xdsreset utility. If no options of xdsprobe other than `–p', `–f', `–F' are used then the `–r' option is not essential because xdsprobe then behaves as a synonym for xdsreset. The examples in the prior description of the `–f' and `–F' options show how the `–r' option can be used to reset specific emulators or target boards.

What does an emulation reset do?

An emulation reset accesses several TBC registers and if no problems are detected the TBC is thoroughly initialised and a JTAG TRST pulse is output to the target DSP or ARM micro-controller. If the DSP is configured correctly, and EMU0 and EMU1 are pulled high, the target DSP or ARM micro-controller is placed in emulation mode. This will allow the emulation logic to be controlled from the JTAG interface. The `–r' option does not validate the integrity or reliability of the JTAG scan-path.

Step 3 – Verifying the TBC Functionality (–y used with –c and –o)

You can test a TBC with the `–y' option of xdsprobe. This option may be used without the related `–c' and `–o' options to test the TBC just once.

The tests may be repeated continuously (or until a key is pressed) by using the `–y –c' combination. The tests may be repeated a specific number of times (or until a key is pressed) by using the `–y –c number' combination.

```
xdsprobe –F xds560.sys –p 0 -y       ← Test a TBC in an XDS560 emulator just once.

xdsprobe –F xds560.sys –p 0 -y –c    ← Test a TBC in an XDS560 emulator continuously.

xdsprobe –F xds560.sys –p 0 -y –c 64 ← Test a TBC in an XDS560 emulator 64 times.
```

Test outputs may be reformatted for saving in a file by also using the `–o filename' option. The following example tests a TBC within an XDS560 continuously (or until a key press or major error occurs) and saves the reformatted results in a file. It was used during emulator development to test for rapid and correct detection of fatal cable-break, power-loss and dead-clock situations.

```
xdsprobe –F xds560.sys –p 0 -y –c –o results.txt
```

What does the '–y' option do?

This option runs a series of tests on the TBC. The tests are independent of the communication channel between CCS and the TBC and independent of the JTAG scan-path between the TBC and any target DSP or ARM micro-controller. This in turn will verify that the TBC in the emulator or DSP board (depending on your configuration) is functioning correctly. There is no output from the TBC to the target DSP or ARM micro-controller. The `–y' option does not validate the integrity or reliability of the JTAG scan-path.

Step 4 – Verifying Scan Path Integrity (–i used with –o)

The `–i' option is used to test the integrity of the JTAG scan-path that is daisy-chained between the TBC, target DSP's and ARM micro-controllers, and any other devices that are included in the JTAG scan path.

This integrity test outputs a large amount of text, and the output will still likely scroll off the visible region of the command window. The results need to be carefully scrutinised, and can be redirected to a file by also using the `–o filename' option. For example:

```
xdsprobe –F xds560.sys –p 0 –i –o results.txt  ←  Test integrity of XDS560's scan-path

xdsprobe –F –p 0 –i –o results.txt           ←  Test integrity of XDS510's scan-path
```

What does the '–i' option do?

The initial operation that is performed on the scan-path is to apply standard JTAG Instructions and apply appropriate data patterns of 'all-ones' and 'all-zeroes' so as to measure the total length of all JTAG instruction registers (IR) and total length of all JTAG bypass registers (DR). The results of these tests are used by xdsprobe to decide if the scan-path has a stuck-fault (implying the scan-path is shorted to either the power or ground voltages). The results are also compared by xdsprobe with its built in knowledge of valid IR and DR lengths for TI's devices. It can decide if the results represent a valid single device scan-path, valid heterogeneous scan-path (multiple and different devices), valid homogeneous multiprocessor device scan-path (multiple but similar devices), or it can decide the results are invalid. The integrity test run on a target consisting of a C6201 DSP with an 8-bit IR and a C6711 DSP with a 46-bit IR should give the following result.

```
The test for the JTAG IR instruction scan-path length succeeded.
The JTAG IR instruction scan-path length is 54 bits.

The test for the JTAG DR bypass scan-path length succeeded.
The JTAG DR bypass scan-path length is 2 bits.

The scan-path appears to consist of 2 devices.
```

After the scan path length has been determined, a series of fixed 32-bit data patterns are applied to the scan-path so as to verify data can be transmitted and received without error. If any of these tests pass then the scan-path is probably complete. If only some of these tests fail there is a timing related problem in the scan-path. Successful completion of all tests ensures a scan-path that provides consistent communication with the target DSP's and ARM micro-controllers. The remaining concern is reliability, which is discussed in the next section.

Step 5 – Verifying Scan Path Reliability (–g used with –c and –o)

The `–g pattern' option is used to test the reliability of the JTAG scan-path that is daisy-chained between the TBC, target DSP's and ARM micro-controllers, and any other devices that are included in the JTAG scan path. This `-g pattern' option differs from the `–i' option in that `–g' applies a single data pattern to the scan path that may be specified by the user. The data pattern may be repeated continuously (or until a key is pressed) by using the `–g pattern –c' combination. The data pattern may be repeated a specific number of times (or until a key is pressed) by using the `–g pattern –c number' combination.

This reliability test outputs less text than the integrity test, but still the output likely scrolls off the visible region of the command window. The results still need to be carefully scrutinised and can be redirected to a file by using the `–o filename' option.

The reliability of the scan-path may be verified by applying a single 32-bit data pattern many times or continuously. This allows the user to observe the success or failure over a period of time, and also to probe the scan-path electrically during the test. For example:

```
xdsprobe –F xds560.sys –p 0 –g –c 65536  ←  Do 64K XDS560 scan-path reliability tests.

xdsprobe –F xds560.sys –p 0 –g –c      ←  Do continuous XDS560 scan-path tests.

xdsprobe –F –p 0 –g –c 65536         ←  Do 64K XDS510 scan-path reliability tests.

xdsprobe –F –p 0 –g –c             ←  Do continuous XDS510 scan-path tests.
```

What does the '–g' option do?

The initial operation performed on the scan-path during the reliability test is identical to that for the integrity test. It measures the total length of all JTAG instruction registers (IR) and total length of all JTAG bypass registers (DR). The results are used by to decide the general state of the scan-path and compared with built in knowledge of valid IR and DR lengths for TI's devices.

After the scan path length has been determined, a single 32-bit data pattern is applied to the scan-path so as to verify data can be transmitted and received without error. The first eight errors that are detected are reported in detail, but do not terminate the scan-path test. A continuously updated summary of the total number test performed and the number of failed tests will be displayed. The default value of the 32-bit data pattern is 0x5533CCAA. An alternative value can be entered with the `–g pattern' option. For example:

```
xdsprobe –F xds560.sys –p 0 –g 0xaaaaaaaa –c 1024  ←  Do 1024 XDS560 scan-path
                                                       reliability tests with
                                                       alternating ones and zeros.

xdsprobe –F –p 0 –g 0xaaaaaaaa –c 1024  ←  Do 1024 XDS510 scan-path reliability
                                           tests with alternating ones and zeros.
```

Since the reliability test continues even when errors are reported, then the `–g pattern –c' combination can be used to continuously inject a fixed pattern in the scan-path that can then be electrically probed to determine with an oscilloscope the location of failures.

Step 6 – Verifying TCLK Frequency and Source (–k used with –r and –f)

The xdsprobe `–k' option is used to print the record maintained by the XDS560 about how it has programmed and measured TCLK based on instructions from CCS Setup or xdsprobe via board configuration files. The following command prints the history resulting from the board configuration file received from CC Setup.

```
Xdsprobe –f BrdDat\ccBrd0.dat –r –p [address] -k
```

What does the '–k' option do?

When the `–k' option of xdsprobe is applied to XDS510 emulators and Sourceless-EPK based products then it outputs the following standard text to report the lack of TCLK programming and measurement capability:

```
---[The log-file for the PLL that may generate the JTAG TCLK]---

There is no hardware for programming the JTAG TCLK frequency.

---[Measure the source and frequency of the JTAG TCLK]----------

There is no hardware for measuring the JTAG TCLK frequency.
```

Figure 9. The XD510 Boilerplate Text Output by the –k Option

When the `–k' option of xdsprobe is applied to XDS560 emulators then it outputs the following standard text as an explanation of the results that follow:

```
The hexa-decimal `YZ' values are commands given to the PLL
while testing for external JTAG TCLK's and while searching
for the optimal frequency of internal JTAG TCLK's.

The `MHz' indicates the frequency programmed into the PLL.
The `Words' indicates the block length in the scan-test.
The `Result' indicates the results of the scan-test.
The `Description' indicates the purpose of the scan-test.

The `Results' of the `-d' and `-k' options have the same meaning:
The `successful' shows a test succeeded.
The `not tested' shows a test was not done.
The `bad value' shows a test failed with a bad value.
The `bad command' shows a test failed with a bad command.
The `bad frqncy' shows a test failed with a bad frequency.
```

Figure 10. The XD560 Boilerplate Text Output by the –k Option

When the `–k' option of xdsprobe is applied to XDS560 emulators connected to a C5421 target system, and the board configuration file has been prepared by CCS Setup to automatically select the TCLK frequency, then xdsprobe produces the following output showing the frequencies tested:

```
---[The log-file for the PLL that may generate the JTAG TCLK]---

The IR/DR scan-path tests used blocks of up-to 512 32-bit words
A total of 36 frequencies have been tested.

  Test   Y    Z    MHz  Words  Results     Description
  ----  ---  ---  ----  -----  -------     -----------
    1   -01  00   0.50   none  not tested  apply low frequency
    2   -01  00   0.50   none  not tested  detect internal clock
    3   -01  09   0.57   none  not tested  detect internal clock
    4   -01  00   0.50    128  successful  auto step initial
    5   -01  0C   0.59    128  successful  auto step delta
    6   -01  1A   0.70    128  successful  auto step delta
    7   -01  2A   0.83    128  successful  auto step delta
    8   -01  3D   0.98    128  successful  auto step delta
    9    00  0A   1.16    128  successful  auto step delta
   10    00  17   1.36    128  successful  auto step delta
   11    00  27   1.61    128  successful  auto step delta
   12    00  3A   1.91    128  successful  auto step delta
   13    01  08   2.25    128  successful  auto step delta
   14    01  15   2.66    128  successful  auto step delta
   15    01  24   3.13    128  successful  auto step delta
   16    01  36   3.69    128  successful  auto step delta
   17    02  06   4.38    128  successful  auto step delta
   18    02  13   5.19    128  successful  auto step delta
   19    02  22   6.13    128  successful  auto step delta
   20    02  34   7.25    128  successful  auto step delta
   21    03  04   8.50    128  successful  auto step delta
   22    03  10  10.00    128  successful  auto step delta
   23    03  1F  11.88    128  successful  auto step delta
   24    03  30  14.00    128  successful  auto step delta
   25    04  02  16.50    128  successful  auto step delta
   26    04  0E  19.50    128  successful  auto step delta
   27    04  1C  23.00    128  successful  auto step delta
   28    04  2D  27.25    128  bad value   auto step delta
   29    04  0D  19.25    512  successful  auto power initial
   30    04  1D  23.25    512  successful  auto power delta
   31    04  25  25.25    512  successful  auto power delta
   32    04  29  26.25    512  successful  auto power delta
   33    04  2B  26.75    512  successful  auto power delta
   34    04  2C  27.00    512  successful  auto power delta
   35    04  2C  27.00    512  successful  auto power delta
   36    04  21  24.25    512  successful  auto margin initial

---[Measure the source and frequency of the JTAG TCLK]----------

The JTAG TCLK is generated from the internal PLL.
The frequency of the JTAG TCLK is measured as 24.2MHz.
```

Figure 11. The XD560 –k Output from C5421 Test Board with Automatic Selection of TCLK Frequency

The following examples show xdsprobe's output when a specific frequency of 30MHz is requested with a C6711 DSK and when the legacy frequency is requested on a C5402 DSK, and also the output when using a VC33 target system with an external TCLK source.

```
---[The log-file for the PLL that may generate the JTAG TCLK]---

  Test   Y    Z    MHz  Words  Results     Description
  ----  ---  ---  ----  -----  -------     -----------
    1   -01  00   0.50   none  not tested  apply low frequency
    2   -01  00   0.50   none  not tested  detect internal clock
    3   -01  09   0.57   none  not tested  detect internal clock
    4    04  38  30.00    512  successful  apply explicit clock

---[Measure the source and frequency of the JTAG TCLK]----------

The JTAG TCLK is generated from the internal PLL.
The frequency of the JTAG TCLK is measured as 30.0MHz.
```

Figure 12. The XD560 –k Output from C6711 DSK with 30MHz Specific TCLK Frequency

```
---[The log-file for the PLL that may generate the JTAG TCLK]---

The IR/DR scan-path tests used blocks of up-to 512 32-bit words
A total of 4 frequencies have been tested.

  Test   Y    Z    MHz  Words  Results     Description
  ----  ---  ---  ----  -----  -------     -----------
    1   -01  00   0.50   none  not tested  apply low frequency
    2   -01  00   0.50   none  not tested  detect internal clock
    3   -01  09   0.57   none  not tested  detect internal clock
    4    03  13  10.38    512  successful  apply explicit clock

---[Measure the source and frequency of the JTAG TCLK]----------

The JTAG TCLK is generated from the internal PLL.
The frequency of the JTAG TCLK is measured as 10.4MHz.
```

Figure 13. The XD560 –k Output from C5402 DSK with 10.368 Legacy TCLK Frequency

```
---[The log-file for the PLL that may generate the JTAG TCLK]---

The IR/DR scan-path tests used blocks of up-to 512 32-bit words
A total of 3 frequencies have been tested.

  Test   Y    Z    MHz  Words  Results     Description
  ----  ---  ---  ----  -----  -------     -----------
    1   -01  00   0.50   none  not tested  apply low frequency
    2   -01  00   0.50   none  not tested  detect external clock
    3   -01  00   0.50    512  successful  test external clock

---[Measure the source and frequency of the JTAG TCLK]----------

The JTAG TCLK is generated from an external source.
The frequency of the JTAG TCLK is measured as 8.19MHz.
```

Figure 14. The XD560 –k Output from VC33 Board with 8.2MHz External TCLK Frequency

Step 7 – Verifying TCLK Quality (–d used with –f)

The XDS510 emulator and all currently available Sourceless EPK products have internal TCLK sources with relatively low frequencies and relatively slow rise and fall times. This means that the quality of the JTAG signal routing to the target DSP or ARM micro-controller has limited impact on CCS's ability to reliably communicate with the system.

The XDS560 emulator has an internal TCLK source with relatively high frequencies (up-to 50MHz) and relatively fast rise and fall times. This means that the quality of the JTAG signal routing to the target DSP or ARM micro-controller has significant impact on the ability of CCS to reliably communicate with the system.

The following command prints the scan-path quality map from 4.0MHz to 48MHz resulting from the board configuration file received from CC Setup.

```
Xdsprobe –f BrdDat\ccBrd0.dat –r –p [address] –d4,48
```

What does the '–d' option do?

When the `–d' option of xdsprobe is applied to XDS510 emulators and Sourceless-EPK based products then it outputs the following standard text to report the lack of TCLK programming and measurement capability:

```
---[Perform the scan-path double test (frequency and data)]-----

There is no hardware for programming the JTAG TCLK frequency.
```

Figure 15. The XD510 Boilerplate Text Output by the –d Option

When the `–d' option of xdsprobe is applied to XDS560 emulators then it outputs the following standard text as an explanation of the results that follow:

```
The hardware does support the scan-path double test.
The JTAG TCLK is generated from the internal PLL.

This scan-path test uses blocks of 512 32-bit words.

The `Results' of the `-d' and `-k' options have the same meaning:
The character `O' shows a test succeeded.
The character `-' shows a test was not done.
The character `X' shows a test failed with a bad value.
The character `#' shows a test failed with a bad command.
The character `%' shows a test failed with a bad frequency.

The characters `[]' and `{}' show the impact of configure
variables in the board configuration file on the maximum test
frequency and the frequency actually chosen for normal operation.
If equal they are marked with `[]' else they are marked with `{}'.
```

Figure 16. The XD560 Boilerplate Text Output by the –k Option

When the `–d4,48' option of xdsprobe is applied to XDS560 emulators connected to a scan-path with DSP's on two separate boards, and the board configuration file was prepared by CCS Setup to automatically select the TCLK frequency, then xdsprobe produces a map showing a simple transition from successful to failed frequencies, suggesting a clean TCLK signal:

```
---[Perform the scan-path double test (frequency and data)]-----

    Base (MHz)  Step (KHz)  Results

     4.0        62.5        O  O  O  O  O  O  O  O
     4.5        62.5        O  O  O  O  O  O  O  O
     5.0        62.5        O  O  O  O  O  O  O  O
     5.5        62.5        O  O  O  O  O  O  O  O
     6.0        62.5        O  O  O  O  O  O  O  O
     6.5        62.5        O  O  O  O  O  O  O  O
     7.0        62.5        O  O  O  O  O  O  O  O
     7.5        62.5        O  O  O  O  O  O  O  O

    Base (MHz)  Step (KHz)  Results

     8.0        125.0       O  O  O  O  O  O  O  O
     9.0        125.0       O  O  O  O  O  O  O  O
    10.0        125.0       O  O  O  O  O  O  O  O
    11.0        125.0       O  O  O  O  O  O  O  O
    12.0        125.0       O  O  O  O  O  O  O  O
    13.0        125.0       O  O  O  O  O  O  O  O
    14.0        125.0       O  O  O  O  O  O  O  O
    15.0        125.0       O  O  O  O  O  O  O  O

    Base (MHz)  Step (KHz)  Results

    16.0        250.0       O  O  O  O  O  O  O  O
    18.0        250.0       O  O  O  O  O {O} O  O
    20.0        250.0       O  X  X  O  X  O  O  X
    22.0        250.0       X  X  X  X  X  X  X  X
    24.0        250.0       X  X  X  X  X  X  X  X
    26.0        250.0       X  X  X  X  X {X} X  X
    28.0        250.0       X  X  X  X  X  X  X  X
    30.0        250.0       X  X  X  X  X  X  X  X

    Base (MHz)  Step (KHz)  Results

    32.0        500.0       X  X  X  X  X  X  X  X
    36.0        500.0       X  X  X  X  X  X  X  X
    40.0        500.0       X  X  X  X  X  X  X  X
    44.0        500.0       X  X  X  X  X  X  X  X

The scan-path double test (frequency and data) has finished.
```

Figure 17. The XD560 –d Output from C6711+C5402 DSK with Automatic Selection of TCLK Frequency

When the `–d4,48' option of xdsprobe is applied to XDS560 emulators connected to a scan-path with DSPs on two separate boards, and the board configuration file was prepared by CCS Setup to automatically select the TCLK frequency, then xdsprobe produces a map showing bands of successful and failed scan-tests, suggesting severe ringing on the TCLK signal:

```
---[Perform the scan-path double test (frequency and data)]-----

    Base (MHz)  Step (KHz)  Results

     4.0        62.5        O  O  O  O  O  O  O  O
     4.5        62.5        O  O  O  O  O  O  O  O
     5.0        62.5        O  O  O  O  O  O  O  O
     5.5        62.5        O  O  O  O  O  O  O  O
     6.0        62.5        O  O  O  O  O  O  O  O
     6.5        62.5        O  O  O  O  O  O  O  O
     7.0        62.5        O  O  O  O  O  O  O  O
     7.5        62.5        O  O  O  O  O  O  O  O

    Base (MHz)  Step (KHz)  Results

     8.0        125.0       O  O  O  O  O  O  O  O
     9.0        125.0       O  O  O  O  O  O  O  O
    10.0        125.0       O  O  O  O  O  O  O  O
    11.0        125.0       O  O  O  O  O  O  O  O
    12.0        125.0       O  O  O  O  O  O  O  O
    13.0        125.0       O  O  O  O  O  O  O  O
    14.0        125.0       O  O  O  O  O  O  O  O
    15.0        125.0       O  O  O  O  O  O  O  O

    Base (MHz)  Step (KHz)  Results

    16.0        250.0       O  O  O  O  O  O  O  O
    18.0        250.0       O  O  O  O  O  O  O  O
    20.0        250.0       O  O  O  O  O  O  O  O
    22.0        250.0       O  O  O  O  O  O {O} O
    24.0        250.0       O  O  O  O  O  O  O  O
    26.0        250.0       O  X  X  X  X {X} X  X
    28.0        250.0       X  X  X  X  O  O  O  O
    30.0        250.0       O  O  O  O  O  O  O  O

    Base (MHz)  Step (KHz)  Results

    32.0        500.0       O  O  O  O  O  O  O  O
    36.0        500.0       O  O  O  O  O  O  O  O
    40.0        500.0       O  O  X  X  X  X  X  X
    44.0        500.0       X  X  X  X  X  X  X  X

The scan-path double test (frequency and data) has finished.
```

Figure 18. The XD560 –d Output from C5510+C5402 DSK with Automatic Selection of TCLK Frequency

Troubleshooting

Failures may be divided into two major groups, TBC access failures and JTAG scan-path failures. TBC access failures must be resolved before attempting JTAG scan-path tests.

Error Reporting

If xdsprobe fails and reports an error then the user should carefully read its explanation. The XDS560 emulator has error reporting that is more accurate than the XDS510 emulator or the Sourceless EPK based products. In particular the XDS560 can provide accurate cable-break, power-loss and dead clock errors, plus it performs scan-path tests at startup that have their own specific errors.

TBC Access Failures

The most likely causes of TBC access failures, in order of probability are: configuration or device address error (possibly just user error); custom adapter library error (possibly a software bug in a Sourceless EPK product); hardware failure (probably ESD damage or electrical overload)

Scan-Paths that are Broken

Use the `–g pattern –c' to continuously inject a fixed pattern in the scan-path that can then be electrically probed with an oscilloscope to determine the location of stuck-faults.

Scan-Paths with Burst Errors

If examination of the errors reported by the `–g –c' continuous reliability test reveals their length equals the scan-path length, then double clocking of the TBC or target devices by EMI generated spikes in the JTAG Test Clock signal (TCLK) may be suspected. This occurs more often in low-humidity environments and in applications with externally sourced TCLKs or low TCLK frequencies. The XDS510 and XDS560 emulators remain high speed products when low TCLK frequencies are used, and that signal should be shielded and carefully routed.

Scan-Paths with Wrong IR Length but Correct DR Length

The correct DR length indicates that the JTAG scan-path is most likely configured correctly, but the DSP is not in emulation mode. When this occurs the most common symptom is the IR length is shorter than expected, actually 4 or 8 instead of an expected 8 or 46. The common causes of this problem are: EMU0 and EMU1 signals are not pulled high or processor test (reserved) pins incorrectly configured.

Scan-Paths with Wrong IR Length and Wrong DR Length

If both the IR and DR lengths are incorrect by the same amount then a link-delay error may be suspected. For example a system with a C6711 DSP and a C6201 DSP reporting lengths of 53 and 3 instead of 52 and 2. This symptom usually occurs in Sourceless EPK based products. It indicates the wrong 'Link Delay' parameter was used by the adapter library. The link-delay represents shift register bits (flip-flops) in the scan-path that are clocked in every JTAG state instead of just in the JTAG shift states. The XDS510 has 4 of these bits in its board and pod—its link-delay is 4. The XDS560 has 5 of these bits in its board and pod—its link-delay is 5. All products with a TBC connected directly to a DSP have a link-delay of 0 (for example the C6711 DSK and C5402 DSK).

The User's Guide for XDSPROBE

This listing is output by the command `xdsprobe –v –h'

Synopsys

```
xdsprobe [-vh]
         [-e name] [-o [file name]] [-c [number]]
         [-n name] [-f file] [-F program/adapter]
         [-r] [-R [data file]]
         [-m [letters]] [-j title]
         [-d [limit],[limit]] [-d [center]] [-k]
         [-b] [-i] [-g [data]] [-y]  [-Y]
         [-a] [-s] [-z]
```

General

```
This XDSPROBE utility is much more sophisticated
than the archaic XDS_DIAG utility that it replaces.

It is Copyright (c) 1990-2002 by Texas Instruments Incorporated.
It is version `3.3.12' that was compiled on `Apr  8 2002'.
It is compatible with Release-3 of the Unified-SCIF stack.

This is one of a set of four command-line utilities
that are distributed with the Unified-SCIF stack:

  xdsreset - this is used to reset scan-controllers and scan-paths.
  xdsprobe - this is used to probe (test) scan-controllers and scan-paths.
  xdserror - this is used to obtain explainations of error numbers.
  xdsbuild - this is used to build and explain scan-path descriptions.

All these utilities have -vh' verbose help options that print full manuals.
The xdsbuild utility also has a `-vi' verbose information option that prints
documentation about board configuration files and configuration variables.

This utility operates on controllers indirectly via the Unified-SCIF
stack (the mid-level software interface for controller operation
and path management) instead of directly on hardware registers.

This XDSPROBE utility is compatible with any operating system or
product supported by the Unified-SCIF stack, either using internal code,
or via an emulator program, or via an emulator adapter built with the
SourceLess-EPK. The list of operating systems includes Win95/98/ME and
WinNT/2K/XP.  The list of products includes ISA-bus XDS510's, PCI-bus
XDS560's, parallel-port DSK's, PCI-bus EVM's, PCI-bus Target Board's,
QuickTurn boxes, and third-party USB, PCI, VME, PCMCIA,
Ethernet and TCP/IP products!
```

Description

```
This XDSPROBE utility is used to develop 510 and 560 class
products and also to test the scan-controllers and scan-paths
that connect them to target systems.  It provides:

This utility has extended scan-controller tests [-y] and embedded memory
tests [-Y] that may be run a fixed number of times or continuously.

This utility has basic scan-path tests [-b -i] with fixed data patterns.
It also has an extended scan-path test [-g] with user selected data patterns
that may be run a fixed number of times or continuously. These tests
will also measure scan-path lengths, and then interpret the values
to detect stuck-faults and link-delay errors, and compare the values
against known valid combinations of JTAG IR path-lengths and JTAG DR
path-lengths for homogeneous systems (with multiple similar devices)
and heterogeneous systems with dis-similar devices.

This utility has a basic tool [-j] for applying a specific JTAG state,
or a sequence of JTAG states to a scan-path and target system.

This utility has a basic tool [-m] for manually applying `SMG_acquire()'
and `SMG_release() commands to the Unified-SCIF stack and thus
potentially suspending and resuming tools such as Code Composer
Studio when it is using some classes of emulator products.

This utility has an advanced tool [-d] for performing combined PLL
programming and scan-path tests over the full frequency range of any
PLL based TCLK source in 560 class and advanced 510 class products.
This tool can be used to detect signal quality problems in the
target system or the emulator's cable and pod; and errors in
the emulators PLL based TCLK source.  Its use is especially
valuable in multiprocessor target systems where the JTAG
scan-path is routed between multiple processors.

This utility has an advanced tool [-k] for extracting the hidden
log-files of PLL programming and scan-path test results that are
stored in 560 class and advanced 510 class products to record
the actions performed by the emulator when selecting a TCLK
frequency based on configure variable values placed in the
board configuration file by Code Composer Studio or the user.

This utility also provides the complete functionality of the
XDSRESET utility.  If the only options used are those available
in that utility, then XDSPROBE behaves as a synonym for it.
It extends that utility by allowing for FPGA's related to the
scan-controller to be initialised using external data from a
file as an alternative to the use of built-in data by XDSRESET.
```

```
On systems that support advanced error detection (such as all 560 class
products and advanced 510 class products) it will also provide distinct
error messages for each of several common interconnect problems between
the scan-controller and target system: lost target power;
dead JTAG clocks; cable breaks near the target;
and cable breaks near the scan-controller.

The major options listed below are the significant functions implemented
by this utility.  The minor options listed below provide extra information
required by (or used to modify) those functions.  The user typically
selects a single major option and one or several minor options.
```

Minor Options

The [–h] or [/?]Option

```
Request the output of brief help or the whole manual.

The verbose option may be used together with this option.
It results in a whole manual instead of brief help.
```

The [–v] or [/v] Option

```
Request the output of verbose help or verbose information.

The output of the help option becomes this manual.
The output of the other options is detailed information
about the actions being performed by that option.
```

The [–p address] Option

```
Provide the io port or board index of an emulator, which may be
a classic XDS510 emulator, a XDS510 class controller requiring an
emulator adapter (such as a DSK or EVM), or a XDS560 class
controller requiring an emulator program.

The addresses have up-to 8 hex-digits with an optional `0x' prefix.
The default address is now 0x0 for all products.

Multiple `-p' options may be used to selected multiple
controllers in the given order of the `-p' options.

The classic XDS510 class controllers actually use addresses 0x240,
0x280, 0x320 and 0x340, but the XDS510 specific code now aliases
0x0-0x3 to those addresses for compatibility with the new default.

The parallel-port DSK products actually use addresses
0x378, 0x278, and 0x3BC, but the DSK specific code now aliases
0x0-0x2 to those addresses for compatibility with the new default.
```

The [–e name] Option

```
Provide a suffix for the name of a variable in a version two board
configuration file. The variable is used to provide extra command
line options, or all of the command line options. When the suffix
is not provided then the default name "PROBE_OPTIONS" will be used.
When it is provided then "PROBE_" plus the suffix will be used.

When this option is not used then all such variables will be ignored.
The obsolete -x option is no-longer required to ensure this.

The options and sub-arguments given by the value of this variable
must have the same syntax as when used on an real command line.
The sub-arguments must be quoted if they contain spaces.

This option allows multiple pre-defined options to be embedded
in the board configuration file, and for one of them to be
selected by the sub-argument to the option.  For example:

  use `-e' to select [probe_options]

  use `-etest1' to select [probe_test1]
  use `-etest2' to select [probe_test2]
  use `-etest3' to select [probe_test3]

These variables also allow appropriate builds of the utility
to be used in environments where C language IO is available,
but the command-line interface is not convenient (a Windows
short-cut) or does not exist (embedded systems).
```

The [–o [output file]] Option

```
Redirect the standard output of this utility and append it to a file.
The default extension of `.txt' is implied if none is specified.

If no file name is provided then the standard output of
this utility will simply be passed to the console.

This option always causes the suppression of backspace characters
normally used to generate scrolling test scores in utilities.

This option does not affect the standard error output of this
utility which is used only to report invalid arguments.

The `-o filename' option can be used together with the `-c number'
option available in this utility when it is run from a script to
append the results of multiple tests to a single output file.
```

The [–c [number]] Option

```
Repeat an operation a fixed number of times or continuously.

If this option is used together with specific major options then
that option will repeat a fixed number of times or continuously.
A running count of the status of the major option is displayed.

If the decimal number is absent or is zero then the operation is
repeated continuously until terminated by the user pressing any key.
If the decimal number is non-zero then the operation is repeated
that many times, or until terminated by the user pressing any key.

The primary uses of this option are with the `-g' given-data tests,
the `-y' scan-controller tests and the `-Y' embedded memory tests.
```

The [–n name] Option

```
Provide the name of the processor device for the `-a' option.
The default processor name is simply `target'.
```

The [–f [board configuration file]] Option

```
Provide the name of the version two format board configuration file.
The default extension of `.cfg' is implied if none is specified.

When the related `-F' option is not used then board configuration files
are compulsory.  So if the `-f' option is also not used then the default
board configuration file named `board.cfg' must be present.  When the
related `-F' option is used then board configuration files are optional.
So if the `-f' option is not used, then an empty scan-path description
and no extra configuration variables are implied.

The documentation of the version two format of board configuration files,
and the details of scan-path description and configuration variables
is generated by the `-vi' verbose information option of XDSBUILD.

The minimum content for the board configuration files needed for TI's
classic XDS510 emulator is one line with an eight character comment.
The minimum content for the XDS510 class products designed using
the Sourceless-EPK is two lines. The extra line uses a [POD_DRVR]
variable to name the emulator adapter that operates the hardware.

The minimum content for the text-formatted board configuration files
needed for TI's XDS560 emulator and other XDS560 class products is
two lines.  The extra line uses a [UNIFY_ECOM_DRVR] variable to name
the emulator program that is loaded into the hardware to operate it.

The TI ISA-bus XDS510 emulator requires at least:
  ;CFG-2.0

The TI PCI bus XDS560 emulator requires at least:
  ;CFG-2.0
  [UNIFY_ECOM_DRVR] 'xds560.out'

The TI Parallel-Port `6711 DSP Starter Kit' require at least:
  ;CFG-2.0
  [POD_DRVR] 'dsk6x11pp.dll'

The TI PCI bus `5510 DSP Target Board' requires at least:
  ;CFG-2.0
  [POD_DRVR] 'pci5510.dll'

The DSP Research `FleXDS PCI emulator' requires at least:
  ;CFG-2.0
  [POD_DRVR] 'podflexds.dll'

The Blackhawk `USB 2.0 JTAG emulator' from EWA Inc requires at least:
  ;CFG-2.0
  [POD_DRVR] 'mdpjtag3.dll'

The Innovative Integration `Code Hammer PCI emulator' requires at least:
  ;CFG-2.0
  [POD_DRVR] 'iipcipod.dll'
```

```
Products that communicate with specific DSP's on the JTAG scan-path
will also require one or more lines of scan-path description in the file.
Newer products such as TI's Code Composer Studio will add this automatically.
Older products such as TI's C Source Debugger require it added manually
```

The [–F [emulator program/adapter file]] Option

```
This option provides a simple alternative to the `-f' option in
situations where no scan-path description and no extra configure
variables are required.  In situations where those items are
required it often allows the emulator specific items to be
excluded from the `-f' options board configuration file.

This option provides the file name of the ECOM based emulator program
(with a `.out' suffix) used to operate a XDS560 class controller,
or the SEPK based emulator adapter (with a `.dll' suffix) used
to operate a XDS510 class controller.  If no file name is
used then TI's classic XDS510 emulator is operated directly.

If the `-F' option is not used to specify the file name of a required
XDS510 class emulator adapter or XDS560 class emulator program then
that file name must be provided with [POD_DRVR] and [UNIFY_ECOM_DRVR]
variables respectively in the board configuration file.

When the `-F' option is missing then the `-f' option's configuration
files are compulsory.  In this case if the `-f' option is also
missing then the default board configuration file
named `board.cfg' must be present.

When the `-F' option is used then the `-f' option's configuration
files are optional.  In this case if the `-f' option is missing
then an empty scan-path description and no extra
configuration variables are implied.

The TI ISA-bus XDS510 emulator can be operated using:
  `xdsprobe -F ...'

The TI PCI bus XDS560 emulator can be operated using:
  `xdsprobe -Fxds560.out ...'

The TI Parallel-Port `6711 DSP Starter Kit' can be operated using:
  `xdsprobe -Fdsk6x11pp.dll ...'

The TI PCI bus `5510 DSP Target Board' can be operated using:
  `xdsprobe -Fpci5510.dll ...'

The DSP Research `FleXDS PCI emulator' can be operated using:
  `xdsprobe -Fpodflexds.dll ...'

The Blackhawk `USB 2.0 JTAG emulator' from EWA Inc can be operated using:
  `xdsprobe -Fmdpjtag3.dll ...'

The Innovative Integration `Code Hammer PCI emulator' can be operated using:
  `xdsprobe -Fiipcipod.dll ...'
```

Major Options

The [–r] Option

```
The controllers selected by the one or more `-p' options will
be reset (loaded, re-programmed, initialised and configured).

The controllers will enter the JTAG 'Test-Logic-Reset'
state and then the TRST signal will be pulsed.

When none of the other major options are
selected then this option is implied.

Systems such as all 560 class products and some 510 class products
are constructed using programmable FPGA's.  The data for the FPGA
is always compiled into the 560 class emulator programs, such as
`xds560.out', and may be compiled into 510 class emulator adapters.
This option will load the emulator program/adapter and re-program
the FPGA, then initialise and configure the controller.
```

The [–R [data file]] Option

```
The controllers selected by the one or more `-p' options will
be reset (loaded, re-programmed, initialised and configured).

The controllers will enter the JTAG 'Test-Logic-Reset'
state and then the TRST signal will be pulsed.

When this option is used without a sub-argument then its behaviour is
identical to the `-r' option.  It will use the FPGA data compiled
into the emulator program/adapter to re-program the FPGA.

When this option is used with a sub-argument naming an FPGA data
file then it will use the data in that file to re-program the FPGA,
instead of the data compiled into the emulator program/adapter.
The intention of this option is to improve productivity during
emulator development, by allowing multiple changes to the FPGA
to be evaluated without repeated re-compiles of the binaries.

This option already supports the common Altera FPGA data file
formats used when developing the XDS560 and XDS5100 emulators.
These are the "Raw Binary File" format with an `.rbf' suffix
and the "Tabular Text File" format with a `.ttf' suffix.
A third format is `C' language source files from which the first
set of array initialisation values, delimited by a pair of `{}'
will be interpreted as if using the "Tabular Text File" format.

Third parties using the latest Sourceless-EPK revision can
extend the supported file formats by writing code to
implement the pod_program_fpga() function that is
accessed with the POD_CMD_PROGRAM_FPGA command.
```

The [–m [letters]] Option

```
The controller will be managed by single letter commands.

`o' - Open the controller and get a new handle.
`a' - Acquire the controller by using the handle.
`r' - Release the controller by using the handle.
`c' - Close the controller by using the handle.
`n' - Do nothing to the controller.
`t' - Terminate the commands.

A series of letters such as `oarcnnoarct' is also valid.
Using acquire or release when closed is bad.

The letter(s) can be entered either on the command-line
or interactively in response to status messages provided
by XDSPROBE in response to the use of this option.

If a controller is opened and then acquired by use of the
command `a' then any USCIF3 compatible debugger or utility
that is using the controller will be suspended until the
controller is released by use of the `r' command.
```

The [–j title] Option

```
The controller will enter the JTAG state whose title
is given by the sub-argument of the `-j' option.

Multiple `-j' options may be used to selected multiple
JTAG states in the given order of the `-j' options.

The titles are case insensitive.  Both the colloquial
short names and the canonical long names are accepted:

`reset'  or `Test-Logic-Reset'
`idle'   or `Run-Test/Idle'
`dscan'  or `Shift-DR'
`iscan'  or `Shift-IR'
`dpause' or `Pause-DR'
`ipause' or `Pause-IR'

The TRST signal will also be pulsed if `reset' is used.
The TDO signal will be all-ones if `dscan' or `iscan' are used.

Adjacent pairs of identical `dpause' or `ipause' titles result
in movement through the inner loop of the JTAG state diagram:

dpause -> dexit1 -> dupdate -> dcapture -> dexit2 -> dpause
ipause -> iexit1 -> iupdate -> icapture -> iexit2 -> ipause
```

The [–d [limit],[limit]] and [–d [center]] Options

```
Perform the double (data and frequency) test on the JTAG IR and DR.
This option only works on systems with a PLL programmable JTAG TCLK
such as all 560 class products and advanced 510 class products.

The double test has been used to detect the symptoms of insidious and
intermittent errors during in the design of target systems and emulators:
signal quality problems in the target system; signal quality problems
in the emulator's cable and pod; inappropriate choices of reference
oscillator for the programmable PLL; obscure race conditions and timing
errors in the PLL programming logic; inaccurate results from the frequency
measurement logic.  The production implementation of the XDS560 emulator
and its cable and pod is known to repeatidly pass these tests perfectly
from 0.5Mhz to near 50.0MHz. Any new errors found are likely due to
signal quality problems in the target system's scan-path.

The double test consists of using the PLL programmable JTAG TCLK to
sweep through a range of frequencies while performing a scan-test at
each frequency.  The range is either between two limit frequencies
or around a center frequency.  The values `limit' and `center' are
decimal numbers between 0.5 and 100.0 that are commonly (but not
always) whole and half integer values with one decimal place.

The tests always sweep from the lower to the higher frequency.
The tests can be terminated by the user pressing any key.

The results of the sweep through a range of frequencies is displayed in
a table format.  The result of the scan-test at each JTAG TCLK frequency
is indicated by one of five flags (similar to the `-k' option):

  `O' -> This character shows a test succeeded.
  `-' -> This character shows a test was not done.
  `X' -> This character shows a test failed with a bad data value
   due to a mismatch between the send and receive data.
  `#' -> This character shows a test failed with a bad command
   due to the controller state being corrupted.
  `%' -> This character shows a test failed with a bad frequency value
   due to a mismatch between the programmed and measured values.

Note that though PLL programming has a resolution of just 1/64,
the frequency measurements are accurate to better than 1/256.
```

```
When the characters `[]' and `{}' enclose any of the five flags they
show the impact of the [unify_tclk_program] and [unify_tclk_frequency]
variables in the board configuration file on the maximum frequency
tested for scan-path operation and the frequency finally selected.
If equal they are marked with `[]', if not they are marked with `{}'.
These variables are inserted either by Code Composer Studio's setup
utility or by the user manually editing that file. The full explanation
of the use of these two configuration variables is included in
the text-format document generated by the `-vi' verbose
information option of XDSBUILD.

The limits of the sweeps through ranges of frequencies are based on the
PLL's programmability. These are the low but usable 0.5MHz and the high
and thoroughly unusable 100.0MHz.  The practical upper limit for XDS560
class products using its production cable and pod is actually 50.0MHz.

The sweeps around a specified center frequency are also based on
the PLL's programmability.  Currently it is eight PLL steps
below the center frequency and eight PLL steps above the
center frequency.  Sixty-four PLL steps double a frequency.

A `-d' without any values and comma will perform the double
test between the default frequencies (0.5Mhz and 50.0MHz).

A `-d 8.0,32.0' will perform the double test
over the range between 8.0MHz and 32.0 MHz.
A `-d 15.0' with a single frequency will perform the
double test over a small range centered around 15.0MHz.

A `-d ,4.0' will perform the double test from the default (0.5Mhz)
start frequency up to the named 4.0 MHz finish frequency.
A `-d 25.0,' will perform the double test from the named 25.0MHz
start frequency up to the default (50.0MHz) finish frequency.

Before the double test is used it may be necessary to temporarily
replace the PLL programming specification in the board configuration
file if that specification is causing failures. The failures can be
viewed in the output of the `-k' option. The original values can be
disabled by inserting a semi-colon in their first column.
The safest possible replacement values are:

  [UNIFY_TCLK_PROGRAM]   'SPECIFIC'
  [UNIFY_TCLK_FREQUENCY] '0.5'
  [UNIFY_TCLK_SCANTEST]  'NOTHING'
```

The [–k] Option

```
Report the JTAG TCLK selection diagnostics. This option only works on
systems with a PLL programmable JTAG TCLK such as all 560 class
products and advanced 510 class products.

When Code Composer Studio or a utility is launched the USCIF software
must select an appropriate JTAG TCLK frequency.  This is done based on
configuration variables placed in the board configuration file either
by Code Composer Studio or by the user. The algorithm used to select
the JTAG TCLK frequency will first check if the programmable PLL
or an external source generates the JTAG TCLK, second it will
measure the scan-path length, and finally it will perform tests
at one or more frequencies to check for any errors when scanning
data through the JTAG IR and DR scan-paths at those frequencies.

The algorithm used to select the JTAG TCLK frequency maintains
an internal record of the frequency choices and test results
(the JTAG TCLK selection diagnostics) during its execution.
These JTAG TCLK selection diagnostics are displayed in a table format.

  The values in the `YZ' column are the commands programmed into the
  PLL so as to output the frequency value in the `MHz' column.

  The values in the `Words' column are the length in 32-bit words
  of any scan-path test performed at the programmed frequency:
    `none', 128 (4k bits), 512 (16k bits), 2048 (64k bits)

  The values in the `Result' column are the result of any scan-path
  test performed at the programmed frequency indicated by one of
  five expressions (similar to the `-d' option):

  `successful' -> This expression shows a test succeeded.
  `not tested' -> This expression shows a test was not done.
  `bad value'     This expression shows a test failed with
                  a bad data value due to a mismatch
                  between the send and receive data.
  `bad command'-> This expression shows a test failed with
                  a bad command due to the controller
                  state being corrupted.
  `bad frqncy' -> This expression shows a test failed with a bad
                  frequency value due to a mismatch between the
                  programmed and measured frequency values.

  Note that though PLL programming has a resolution of just 1/64,
  the frequency measurements are accurate to better than 1/256.
```

```
  The values in the `Description' column are the purpose of any
  scan-path test preformed at the programmed frequency or
  the result of any frequency measurement:

  `detect external clock' -> The frequency tests detected an external clock.
  `detect internal clock' -> The frequency tests suggest an internal clock.
  `apply low frequency'   -> The minimum TCLK frequency is programmed.
  `apply explicit clock'  -> A specific TCLK frequency is programmed.
  `auto step initial'     -> The initial step-upwards frequency is programmed.
  `auto step delta'       -> A higher step-upwards frequency is programmed.
  `auto power initial'    -> The initial optimisation frequency is programmed.
  `auto power delta'      -> A modified optimisation frequency is programmed.
  `auto margin initial'   -> The initial safety-margin frequency is
programmed.
  `auto margin delta'     -> A lower safety-margin frequency is programmed.

  The algorithm used to select the JTAG TCLK frequency is controlled by
  up-to five configuration variables in the board configuration file.

  [UNIFY_TCLK_PROGRAM]
  [UNIFY_TCLK_FREQUENCY]
  [UNIFY_TCLK_SCANTEST]
  [UNIFY_TCLK_LOOPCOUNT]
  [UNIFY_TCLK_REFERENCE]

Only the first two configuration variables are inserted by Code Composer
Studio's setup utility.  All five can be inserted by the user  manually
editing that file.  The complete explanation of the use of these
configuration variables is included in the text-format document
generated by the `-vi' verbose information option of XDSBUILD.

Code Composer Studio's setup utility uses only three of the many possible
combinations of [unify_tclk_program] and [unify_tclk_frequency].

To use the legacy XDS510 frequency (always 10.368MHz).

  [UNIFY_TCLK_PROGRAM] 'LEGACY'

To use an automatically selected frequency with an upper limit
of the maximum HS-RTDX frequency (currently 35.0MHz).

  [UNIFY_TCLK_PROGRAM]   'AUTOMATIC'
  [UNIFY_TCLK_FREQUENCY] 'EXCHANGE'

To use a specific (user-selected) frequency (such as 15.5MHz)

  [UNIFY_TCLK_PROGRAM]   'SPECIFIC'
  [UNIFY_TCLK_FREQUENCY] '15.5'
```

The [–b] Option

```
Perform the broken path test on the scan-path.

Two blocks of fixed 32-bit data values are scanned through
the JTAG IR instruction register and JTAG DR bypass
register so-as to check their integrity.

The values used are 0x00000000 and 0xFFFFFFFF.
They were chosen to verify that there are
no stuck-fault errors in the two paths.

In each test the first eight errors detected
will each result in one line of explanation.
After the eighth error the test will be mute.
```

The [–i] Option

```
Perform the integrity test on the scan-path.

The length of the JTAG IR instruction register and JTAG DR
bypass register are first measured, then four blocks of fixed
32-bit data values are scanned through the two paths to check
their integrity.

Two of the blocks use the values 0xFE03E0E2 and its inverse
0x01FC1F1D. These values were chosen because they are good
at detecting alignment errors that shift the data by 1-31 bits.
These errors occur with adapters having bad link-delay values.

Two of the blocks use the values 0x55330F0F and its inverse
0xAACCF0F0. These values were chosen because they are good at
generating at-speed errors with excessively fast JTAG clocks.

In each test the first eight errors detected
will each result in one line of explanation.
After the eighth error the test will be mute.
```

The [–g [data]] Option

Perform the given data test on the scan-path.

The length of the JTAG IR instruction register and JTAG DR bypass register are first measured, then blocks of data are scanned through the JTAG IR and DR paths to check their integrity.

The optional sub-argument is the given data value which must be in 32-bit hexadecimal format. Its new default value of 0x5533CCAA is chosen for the trivial reasons of leading and trailing zeros, single and double bits patterns, and its 16-bit symmetry.

If the initial attempt to measure the JTAG register lengths fails then the given data will still be scanned through the scan-paths. This allows the use of the -g and -c options together so-as to inject a user selected value into a possibly broken scan-path while the location of the failure is diagnosed with an oscilloscope or logic analyser.

In each test the first eight errors detected will each result in one line of explanation. After the eighth error the test will be mute.

If the -c option is used together with this option then the tests will repeat a fixed number of times, or repeat continuously until terminated by the user pressing any key. A running count of total tests and failed tests is displayed.

Blocks of the given 32-bit data value are scanned through the JTAG IR path to check its integrity until any key is pressed, at which time blocks of the given 32-bit data value are scanned through the JTAG DR paths to check its integrity.

If the tests are repeated continuously then the given data test will also exercise USCIF3's two major scan commands (SC_CMD_DO_STATUS_3 and SC_CMD_SCAN_FULL_3) by alternating between two implementations.

A less obvious but valuable use of both the `-g' and `-y' options together with the continuous option `-c' is to evaluate an emulators ability to handle target-cable disconnects, target power losses and target JTAG clock failures by causing those failures to occur while running the `-c -g' and `-c -y' options, and then also checking that the utility can reliably restart after the problem is corrected.

The [–y] Option

```
Perform the scan-controller hardware tests.

These verify that the scan-controller hardware is operating
correctly at the chosen test clock and local clock frequencies.

The hardware tests involve register accesses, buffer accesses,
parameter and pointer accesses, local memory accesses and
also run a variety of scan commands in data loop-back mode.

In each test the first eight errors detected will each result in one
line of explanation.  After the eighth error the test will be mute.

The verbose option may be used together with this option.
It will provide one line of explanation for every check,
irrespective of it finding a success or failure.
This may result in surprising quantities of text.

If the -c option is used together with this option then the tests
will repeat a fixed number of times, or repeat continuously until
terminated by the user pressing any key.  A running count
of total tests and failed tests is displayed.

A less obvious but valuable use of both the `-g' and `-y' options
together with the continuous option `-c' is to evaluate an emulators
ability to handle target-cable disconnects, target power losses and
target JTAG clock failures by causing those failures to occur while
running the `-c -g' and `-c -y' options, and then also checking that
the utility can reliably restart after the problem is corrected.
```

The [–Y] Option

```
Perform the embedded memory hardware tests.

These verify that the local-memory hardware is operating
correctly at the chosen local clock frequency.

These tests are still under development and they
will be extended in later releases of this utility.

Earlier releases of this utility had `-d/-l' options for memory
download and upload tests. They have been merged into this option.

If the -c option is used together with this option then the tests
will repeat a fixed number of times, or repeat continuously until
terminated by the user pressing any key.  A running count
of total tests and failed tests is displayed.
```

The [–a] Option

```
Perform the analysis test on the scan-path.

The length of the JTAG IR instruction register and JTAG DR
bypass register are first measured, then the devices on the
scan-path are interrogated to confirm that they match the
description provided in the board configuration file.

The board configuration file used in the analysis test must
be generated by Code Composer Studio or the XDSBUILD utility.

The analysis test is still under development.
```

The [–s] Option

```
Show the configure variable values that affect the Unified-SCIF.
```

The [–z] Option

```
Perform the zero-client test on USCIF3.

These verify that the acquire and release function built
into the Unified-SCIF are operating correctly.

These tests are still under development and they
will be extended in later releases of this utility.

If the -c option is used together with this option then the tests
will repeat a fixed number of times, or repeat continuously until
terminated by the user pressing any key.  A running count
of total tests and failed tests is displayed.
```

Examples

```
Some examples of the usage of this XDSPROBE utility are:

`xdsprobe -h'           <- List the brief help
`xdsprobe /?'           <-   using just a few paragraphs.

`xdsprobe -vh | more'   <- List the verbose help by
`xdsprobe /? /v | more' <-   providing this whole manual!

`xdsprobe [options] > xdsprobe.txt' <- Save everything in a text file.

The following examples will need a default version
two board configuration file named 'board.cfg':

`xdsprobe'
   Reset an emulator selected by [POD_DRVR] or [UNIFY_ECOM_DRVR]
   variables in the board configuration file using default addresses.
   If the variables don't exist then reset a classic XDS510 emulator.
   In all three cases use default addresses and provide brief comments.

`xdsprobe -p0x280 -v'
   Reset a classic XDS510 emulator at port 0x280 with verbose comments.

`xdsprobe -p320 -p340'
   Reset a pair of classic XDS510 emulator's at port 0x320 and 0x340.

The following examples do not require any board configuration file:

`xdsprobe -F -v'
   Reset a classic XDS510 emulator with verbose comments.

`xdsprobe -F xds560.out -R fpgadata.ttf -v'
   Load the FPGA in an XDS560 class emulator from an external file.

`xdsprobe -F xds560.out -d8,32'
   Do data/frequency tests between 8 and 32MHz on a XDS560 class emulator.

`xdsprobe -F mdpjtag3.dll -c64 -y'
   Do 64 controller tests on a USB 2.0 emulator from BlackHawk.

`xdsprobe -F podflexds.dll -c256 -g'
   Do 256 scan-path tests on a FleXDS PCI emulator from DSP Research.
```

The User’s Guide for Board Config’ Files

This listing is output by the command `xdsbuild –v –i’

The Contents

The Introduction

The version 2 format of board configuration files is understood by CCS 2.x and the USCIF3 software it uses to operate the Test Bus Controllers in emulators, DSK's and EVM's. When composing these files the very first line in the file is the standard label. Following this are comments, scan path description and configure variable description that may be blocked in any order, or interleaved in any order.

The board configuration file may consist of as little as a label line, or a label line plus configure variable description, or a label plus scan-path description, or all three.

Here is an example 6-line file without any scan-path description or configure variables. It is adequate to reset an XDS510 emulator with the XDSRESET utility.

```
;cfg-2.0
;
;Use this file to reset an XDS510 with XDSRESET.
;No configure variables or scan path description is needed.
;
;The end of the file
```

Here is an example 7-line file without any scan-path description or configure variables. It is adequate to reset an XDS560 emulator with the XDSRESET utility.

```
;cfg-2.0
;
;Use this file to reset an XDS560 with XDSRESET.
;No configure variables or scan path description is needed.
[unify_ecommode] yes
;
;The end of the file
```

Here is an example 12-line file for an XDS560 emulator with a scan-path consisting of a C54x DSP and an ARM micro-controller along with three configure variables.

```
;cfg-2.0
;
;The configure variables.
[unify_ecommode]        yes        ; Use an XDS560 class emulator.
[unify_tclk_program]   'automatic' ; Select TCLK frequency by auto-ranging.
[unify_tclk_frequency] 'exchange'  ; Do not exceed maximum HS-RTDX frequency.
;
;The scan path description.
"cpu_a" TI320C54xx
"cpu_b" TIARM9xx
;
;The end of the file
```

The Label

The very first line of a board configuration text file using the version 2 format must start with the following eight characters:

```
;cfg-2.0
```

The label is case insensitive. The semicolon marks the line as a comment, the `cfg' indicates a board configuration file, and the `2.0' indicates version 2.0 of the file format.

The Comments

The semi-colon and anything following it is interpreted as a comment in board configuration files. For example:

```
;This is a line of comment text.
```

Comments may be used to conveniently suppress configure variables without deleting their text, by inserting a semi-colon at the first column of the line.

```
;This variable is ignored until the semi-colon is deleted.
;[unify_noamble] yes
```

The Scan Path Description

The set of acceptable class ID's (device family names) used in the scan-path descriptions of board configuration files are chosen to fully distinguish the supported device families both by the user readable name and the corresponding internal class number. Please refer to the next section for the class ID's.

The scan-path is modeled as an ordered list of devices. Each device in the scan-path description is represented by a unique name and a class ID (device family name), and the list will proceed from the device closest to the TBC's JTAG TDI pin to the device closest to the TBC's JTAG TDO pin. The device names are case insensitive. Each device name is limited to a maximum of eight alphanumeric or underscore characters with the first character always being alphabetic. The device name is adorned with double quotes and the class ID is a pre-defined alphanumeric symbol.

For a bypass device, we use BYPASSnnn as the class ID where nnn specifies the number of bits in the IR in hexadecimal notation:

```
"bypass_name" BYPASSnnn
```

For most other devices, only device names and device classes need to be described in the text file as follows.

```
"device_name" CLASS_ID
```

For SPL devices we need to use SPL as the class ID. We also need to describe the four secondary emulation scan paths connected to it using exactly four sets of open and close braces:

```
"spl_name" SPL
{
  "device_name" CLASS_ID
}
{
  "device_name" CLASS_ID
}
{
  "device_name" CLASS_ID
}
{
  "device_name" CLASS_ID
}
```

For ASP devices, we will use ASPaaa as the class ID where aaa specifies the three digits hexadecimal address of the ASP devices. We also need to describe the one secondary emulation scan path connected to it using exactly one set of open and close braces:

```
"asp_name"    ASPaaa
{
  "device_name" CLASS_ID
}
```

The Scan Path Description's Class Identifiers

```
Here is the list of the acceptable class ID's (device family names)
for use in version 2 format board configuration files.
The device family names cannot be abbreviated.

GENERICnnn   Any device with a JTAG IR length of 0x1-0x400.
             These device families do need custom device drivers.

BYPASSnnn    Any device with a JTAG IR length of 0x1-0x400.
ASPaaa       The `8996 Addressable Scan Port (ASP) devices.
SPL          The `8997 Scan Path Linker (SPL) devices.
DBM          The `8994 Digital Bus Monitor (DBM) devices.
             These device families don't need device drivers.

TI370P08xxx - The Prism 8-bit micro-controller devices
TI370P16xxx - The Prism 16-bit micro-controller devices

TI320VC3x   - The VC3X DSP devices with an 8-bit JTAG IR
TI320C4x    - The C4X DSP devices with an 8-bit JTAG IR
TI320C5x    - The C5X DSP devices with an 8-bit JTAG IR
TI320C8x    - The C8X DSP devices with an 8-bit JTAG IR

TIARM7xx    - The ARM-7 embedded micros with a 4-bit JTAG IR
TIARM9xx    - The ARM-9 embedded micros with a 4-bit JTAG IR

TI320C20x   - The C2000 DSP devices with an 8-bit JTAG IR
TI320C24x   - The C2000 DSP devices with an 8-bit JTAG IR
TI320C27xx  - The C2000 DSP devices with a 38-bit JTAG IR
TI320C28xx  - The C2000 DSP devices with a 38-bit JTAG IR

TI320C54x   - The C5000 DSP devices with an 8-bit JTAG IR
TI320C54xx  - The C5000 DSP devices with an 8-bit JTAG IR
TI320C55xx  - The C5000 DSP devices with a 38-bit JTAG IR

TI320C620x  - The C6000 fixed-point DSP's with an 8-bit JTAG IR
TI320C670x  - The C6000 float-point DSP's with an 8-bit JTAG IR
TI320C621x  - The C6000 fixed-point DSP's with a 46-bit JTAG IR
TI320C671x  - The C6000 float-point DSP's with a 46-bit JTAG IR
TI320C64xx  - The C6000 DSP devices with a 38-bit JTAG IR

The triple n's represent up-to that many hexadecimal
digits for JTAG IR lengths.  The triple a's represent
exactly that many hexadecimal digits for address values.
The single, double and triple x's represent a literal 'x' or
exactly that many hexadecimal digits for a device identifier.
```

The Configure Variables

The configure variables are used for internal development purposes or external product release purposes. They replace the environment variables used by WIN32 environments and the external variables specified used by Workstation box and Thunderstorm card environments in older software.

An example of configuration variables being used for internal development purposes is configuring USCIF3 for the XDS510 to operate at slow TCLK mode and to operate with non-standard link-delays. An example of configuration variables being used for external product release purposes is providing the correct adapter driver information for a C6211 DSK or C5510 EVM. All this run-time configuration is specified in the board configuration file. Software developers can specify their own configuration variables and use the appropriate USCIF3 API functions to fetch the variable from a board configuration file that has been loaded into memory.

The Names Of Configure Variables

In the board configuration file, we can specify the name and value for any relevant configuration variables. Each configuration variable must consist of a unique name. The variable name is case insensitive and is limited to a maximum of thirty-two alphanumeric or underscore characters, with the first character always being an alphabetic character. Each configuration variable name must be wrapped by an open square bracket "[" and a close square bracket "]" to distinguish it from the device names in the scan path description.

The syntax for describing configuration variables is as follows:

```
[name] value
```

Examples of the possible types of value are as follows:

```
[boolean_name]  yes (or no)  ; Boolean value
[hexa_name]     0x12345678   ; Hexadecimal number
[octal_name]    012345678    ; Octal number
[decimal_name]  12345678     ; Decimal number
[string_name]  'hello world' ; String value representing some text
[string_name]  '3.141592653' ; String value representing a number
```

The Values Of Configure Variables

```
The configuration variables can have three data types:

  boolean type, numeric type, string type.

Configure variables that are boolean have values that are
unadorned. The acceptable case-insensitive synonyms
for the two values NO and YES are:

  zero, no, not, low, false, negate, negative
  one, yes, high, true, truth, assert

Configure variables that are numerical have values that are unadorned.
They may be of three case-insensitive types that are
internally represented as 32-bit values:

  hexadecimal numbers - prefix with "0x" or "0X",
  octal numbers - prefix with "0",
  decimal numbers - no prefix

Configure variables that are strings are adorned with single quotes.
They can consist of any characters except the single quote character,
the new line character, the carriage return character and the string
terminating character "\0".  The length of the string cannot
exceed 256 characters, which does not include the single
quotes around the string. For example:

  'fall', 'rise', 'legacy', '10.368'

Experience has shown that the most user-friendly configure
variables are those that are text strings used to represent
both numerical and string values. This requires more complex
software to interpret values, but more importantly allows those
values to be more self-explanatory and thus more user friendly.

Experience has also shown that the least user-friendly configure
variables are sets of related boolean variables that require
obscure prioritisation rules when several are used simultaneously.
```

The Currently Supported Configure Variables

The variables documented here are those supported by release 3.3.11 and later of USCIF3 and its utilities for the XDS510 and XDS560. This also corresponds to the first release of software supporting the XDS560 with Code Composer Studio 2.x.

The default values of these variables are interpreted by the software to be appropriate for the emulator class (XDS510, XDS560 or XDS5100) and the version of the emulator's FPGA, cable and pod.

The currently supported configuration variables are:

- [unify_ecom_drvr] and [unify_ecom_port]
- [unify_ecommode]
- [unify_linkdly]
- [unify_dlymode] and [unify_dlysize]
- [unify_tdoedge] and [unify_tdiedge]
- [unify_tbconly] and [unify_exlalso]
- [unify_logmode] and [unify_logfile]
- [unify_tclk_program] and [unify_tclk_frequency]
- [unify_pll_program] and [unify_pll_frequency]
- [unify_tclk_reference]
- [unify_tclk_loopcount]
- [unify_tclk_scantest]
- [unify_size_scantest]
- [unify_size_irpath] and [unify_size_drpath]
- [unify_bigscan] and [unify_noamble]
- [unify_clkmode]
- [unify_slowclk], [unify_slowfrq], [unify_fastclk] and [unify_autoclk]
- [pod_drvr] and [pod_port]
- [pod_logmode], [pod_logfile] and [pod_logtype]
- [pod_blkmode]
- [reset_option], [probe_option], [error_option] and [build_option]

The [unify_ecom_drvr] and [unify_ecom_port] Variables

```
These two variables are used by the XDS560 software
to load and address a program OUT file.

The variable [unify_ecom_drvr] has a string value that is the file name
of the program OUT loaded by the XDS560 software to operate
a TI or third-party XDS560 class emulator.

 'xds560.out' - The program file for the XDS560.

The variable [unify_ecom_port] has a numeric value that is the port address
that the XDS560 software to be passed to the program OUT instead of
the port address provided by the debugger or utility software.

  0x01234567 - Up to eight hexadecimal digits that over-ride the port address.
```

The [unify_ecommode] Variable

```
This variable is considered out-of-date and replaced by [unify_ecom_drvr]
and [unify_ecom_port].  It is still supported by the XDS560 software
in case of accidental use with early board configuration files.
It is ignored if the newer variables are present.

This older variable provided limited flexibility with just two options:
1. Select a classic XDS510 emulator or a 510
   class emulator using an emulator adapter.
2. Select a XDS560 emulator using an emulator adapter named `xds560.out'.

The boolean variable [unify_ecommode] can be set to YES or NO.

  no  - This implies selection of 510 hardware and software.
  yes - This implies selection of 560 hardware and software.

To modify the default selection of USCIF communication with
an XDS150 class product or XDS560 class product
use the boolean values NO and YES:

To communicate with 510 hardware and software:

 [unify_ecommode] no

To communicate with 560 hardware and software:

 [unify_ecommode] yes
```

The [unify_linkdly] Variable

```
This string variable matches the emulator to its cable and target
by compensating for delays between the timing of the JTAG
output signals (TMS and TDO) and the JTAG input signal TDI.

This variable is often used by users of the Sourceless-EPK and
Quickturn hardware to modify the link-delay caused by the cable
and pod connecting the Test Bus Controller and target.

The variable [unify_linkdly] has a string variable with two options:

 'default'  - Use the default link-delay (currently '4').
 '0' - '31' - Use the specified link-delay value.

The XDS560 software will modify the link-delay value to match
the TDO and TDI timing. The XDS510 software currently
does not do that, but may do so at some later date.

Examples:

When using an XDS510 with Quickturn hardware
the link-delay is sometimes increased by one TCLK:

 [unify_linkdly]  '5'

When using an embedded TBC or TBC-XL that is directly
connected to a target device the link-delay is zero:

 [unify_linkdly]  '0'
```

The [unify_dlymode] and [unify_dlysize] Variables

```
These two variables are considered out-of-date and replaced by
[unify_linkdly].  They are still supported by the software in case
of accidental use.  They are ignored if the newer variable is present.

The boolean variable [unify_dlymode] was used to choose
The use of the default value (no) or an explicit value (yes).
The numeric variable [unify_dlysize] was used to provide the explicit value.

Examples:

When using an XDS510 with Quickturn hardware
the link-delay is sometimes increased by one TCLK:

 [unify_dlymode]  YES
 [unify_dlysize]  5

When using an embedded TBC or TBC-XL that is directly
connected to a target device the link-delay is zero:

 [unify_dlymode]  YES
 [unify_dlysize]  0
```

The [unify_tdoedge] and [unify_tdiedge] Variables

```
These string variables select the timing of the TDO and TDI
JTAG data pins on the 14-pin header of the XDS510 and XDS560,
relative to the rising and falling edge of TCLK.

To modify the default timing of the TMS/TDO and TDI pins
use the string values 'default', 'fall' and 'rise'.
These variables use the honest JTAG naming convention where
TDO is the data output pin from the header or device
and TDI is the data input pin to the header or device.

Currently the default timing of both pins is 'rise' to
maximise the usable TCLK (JTAG Test Clock) frequency.

To output TDO on the default edge of TCLK:
 [unify_tdoedge] 'default'

To output TDO on the rising edge of TCLK:
 [unify_tdoedge] 'rise'

To output TDO on the falling edge of TCLK:
 [unify_tdoedge] 'fall'

To output TDI on the default edge of TCLK:
 [unify_tdiedge] 'default'

To output TDI on the rising edge of TCLK:
 [unify_tdiedge] 'rise'

To output TDI on the falling edge of TCLK:
 [unify_tdiedge] 'fall'

Currently the variable [unify_tdiedge] is ignored by the
XDS510 and XDS560 because its use has been found to have
little or no effect on the maximum usable TCLK frequency.

The XDS560 software will modify the link-delay value to match
the TDO and TDI timing. The XDS510 software currently
does not do that, but may do so at some later date.

The documentation for our Test Bus Controllers uses the standard JTAG
naming convention with the data output pin of the controller and
targets all named TDO and the data input pin of the controller and
targets all named TDI, so that a TDO pin always connects to a TDI pin.

The documentation for TI's 14-pin emulator header uses a naming convention
with the data output pin named TDI and the data input pin named TDO,
to match the target device pin names to which they are connected.
```

The [unify_tbconly] and [unify_exlalso] Variables

```
These boolean variables select the operating mode of the
Test Bus Controller in the XDS510 and XDS560 products.

To modify the default configuration of the Test Bus
Controller use the boolean values NO and YES:

If the (XDS510 class) emulator's default is TBC-XL
mode then to operate in its default mode:

 [unify_tbconly]  no

To operate in TBC mode if supported by the emulator:

 [unify_tbconly]  yes

If the (XDS560 class) emulator's default is TBC
mode then to operate in its default mode:

 [unify_exlalso]  no

To operate in TBC-XL mode if supported by the emulator:

 [unify_exlalso]  yes
```

The [unify_logmode] and [unify_logfile] Variables

```
These two variables are used to generate text formatted log-files
when using the Win32 build (xdsfast3.dll) of the USCIF3 software.
They are used to aid in the development of PTI software and USCIF3
utilities by configuring the generation of text formatted
log-files that document the interaction of the xdsfast3.dll
with the PTI software and USCIF3 utilities.

The release builds of the xdsfast3.dll currently exclude support
for these two variables (the corresponding UNIFY_LOG_MODE
build opion is 0).  Engineers desiring to use these two
variables must request an updated build.

The variable [pod_logmode] has a boolean value:

  no     - Disable the generation of a adapter log-file.
  yes    - Enable the generation of an adapter log-file.

The variable [pod_logfile] has a string value:

 'Log-File.txt' - The file-name of the USCIF3 log-file.
```

The [unify_tclk_program] and [unify_tclk_frequency] Variables

```
These variables select the frequency of the
TCLK JTAG clock pin in XDS560 products.

First note that currently the maximum frequencies compatible
with XDS560 JTAG is 50MHz, and with XDS560 HS-RTDX is 35MHz.
In many existing systems the maximum frequency will be limited by the
number of target devices and the quality of the JTAG signal paths.

There are ten choices for programming the XDS560's PLL that are
selected by using the string variable named [unify_tclk_program].
Nine of the choices are explicitly selected by the user and the
tenth ('external') is normally automatically decided by the software
testing the connection between the 14-pin header and the target device.

The variable [unify_tclk_program] has a string value with ten options:

 'external'  - Use an external clock source.
 'default'   - Use the default clock frequency.
 'reference' - Use the PLL reference frequency.
 'minimum'   - Use the minimum internal frequency (currently 0.5MHz).
 'legacy'    - Use the legacy XDS510 frequency (always 10.368MHz).
 'standard'  - Use the standard TBC frequency (currently 30MHz).
 'exchange'  - Use the default HS-RTDX frequency (currently 35MHz).
 'maximum'   - Use the maximum internal frequency (currently 50MHz).
 'specific'  - Use a specific (user-selected) frequency.
 'automatic' - Use an automatically selected frequency.

The last two of these choices for programming the XDS560's PLL
require a frequency value. It is provided by a second configure
variable named [unify_tclk_frequency] and is parsed as a string.

In the case of [unify_tclk_program] being 'specific' the
value of [unify_tclk_frequency] is the actual
frequency the XDS560 will use.

In the case of [unify_tclk_program] being `automatic' the value
of [unify_tclk_frequency] is the maximum frequency the
XDS560's automatic auto-ranging algorithm will use.
They string value of [unify_tclk_frequency] can represent the decimal
frequency in MHz, in the range from 0.50MHz to at least 49.5MHz.
If the value is a whole or half mega-hertz value entered with
a trailing '.0' or '.5' then the exact frequency will be output.
Other values are output with an accuracy of about 0.7%.
```

```
The string value of [unify_tclk_frequency] can also have a
limited set of non-numeric values so as to allow USCIF
to pick the corresponding numeric value.

The variable [unify_tclk_frequency] has a string value with seven options:

 'default'        - Use the default clock.
 'minimum'        - Use the minimum internal frequency (currently 0.5MHz).
 'legacy'         - Use the legacy 510 frequency (always 10.368MHz).
 'standard'       - Use the standard TBC frequency (currently 30MHz).
 'exchange'       - Use the default HS-RTDX frequency (currently 35MHz).
 'maximum'        - Use the maximum internal frequency (currently 50MHz).
 '1.0' - '49'     - Use exact whole megahertz frequencies.
 '0.5' - '49.5'   - Use exact half megahertz frequencies.
 '0.75' - '49.75' - Use near (<1%) and exact quarter megahertz frequencies.

Examples:

To request the emulator to use the legacy XDS510 frequency:

 [unify_tclk_program]   'legacy'

To request the emulator to use the standard TBC frequency:

 [unify_tclk_program]   'standard'

To request the emulator to use the default HS-RTDX frequency:

 [unify_tclk_program]   'exchange'

To request the emulator to auto-range up-to the HS-RTDX limit:

 [unify_tclk_program]   'automatic'
 [unify_tclk_frequency] 'exchange'

To request the emulator to auto-range up-to 5.0MHz:

 [unify_tclk_program]   'automatic'
 [unify_tclk_frequency] '5.0'

To request the emulator to use a specific frequency of 13.75MHz:

 [unify_tclk_program]   'specific'
 [unify_tclk_frequency] '13.75'

To request the emulator to use the reference oscillator
from its PLL, which is 1MHz, 2MHz, 4MHz, 5MHz or 10MHz.

 [unify_tclk_program]   'reference'
```

The [unify_pll_program] and [unify_pll_frequency] Variables

```
These two variables are considered out-of-date and replaced by
[unify_tclk_program] and [unify_tclk_frequency].  They are
still supported by the XDS560 software in case of accidental
use with early board configuration files. They are ignored
if the newer variables are present.

These older variables provided limited flexibility with just three options:
1. Program the PLL with the XDS510 legacy frequency.
2. Auto-range the PLL with an upper limit of the HS-RTDX maximum frequency.
3. Program the PLL with half and whole megahertz frequencies.

The boolean variable [unify_pll_program] can be set to YES or NO.

  no  - This implies program the PLL to use the legacy
        XDS510 TCLK frequency which is 10.368MHz.
  yes - This implies program the PLL to auto-range the TCLK
        frequency or program the PLL to a specific TCLK frequency.

The numeric variable [unify_pll_frequency] can have values from 0 to 100.

  0          - This selects auto-ranging.
  1, .., 100 - These select specific frequencies in `half-megahertz' steps,
               thus 1, 2, .., 99, 100 are 0.5MHz, 1.0MHz, .., 49.5MHz, 50MHz.

Examples:

To request the emulator to use the XDS510 legacy frequency of 10.368MHz:

 [unify_tclk_program]   no

To request the emulator to auto-range the TCLK frequency,
with an upper limit of the HS-RTDX maximum:

 [unify_tclk_program]   yes
 [unify_tclk_frequency] 0 (autoranging selected by zero value)

To request the emulator to use a specific whole or half megahertz
frequency represented by a number that is its double.

 [unify_tclk_program]   yes
 [unify_tclk_frequency] 70 (specific frequency is this number divided by 2)
```

The [unify_tclk_reference] Variable

```
This string variable instructs the software in XDS560 products to generate
PLL programming commands for a specific reference oscillator frequency.
The PLL generates the JTAG clock TCLK in XDS560 products.

There are five options used to instruct the XDS560 software
to match its PLL programming commands to specific
reference oscillator frequency.

The variable [unify_tclk_reference] has a string value with six options:

 'default' - Assume the default PLL reference oscillator.
 '1.0'     - Assume a 1.0MHz PLL reference oscillator.
 '2.0'     - Assume a 2.0MHz PLL reference oscillator.
 '4.0'     - Assume a 4.0MHz PLL reference oscillator.
 '5.0'     - Assume a 5.0MHz PLL reference oscillator.
 '10.0'    - Assume a 10.0MHz PLL reference oscillator.

As an example, to instruct the emulator to generate PLL
programming commands to match a 2.0MHz reference:

 [unify_tclk_reference] '2.0'
```

The [unify_tclk_loopcount] Variable

```
This string variable modifies the scan-path test used by software in XDS560
products to validate the selection of 'automatic' JTAG Clock TCLK
frequencies that were selected by [unify_tclk_program]
and [unify_tclk_frequency].

It instructs the emulator to modify its auto-ranging `loop-count'.
This is the maximum number of frequencies and scan-tests
which will be tried by the auto-ranging algorithm.

The variable [unify_tclk_loopcount] is a string with two options:

 'default'   - Use the default loop-count (currently '64').
 '2' - '256' - Use the explicit loop-count value.

The effect of these options are logged by the 560 software and
reported by XDSPROBE's -k option, so as to allow investigation
of the success and failure of of auto-ranging scan-tests.

When investigating auto-ranging failures on multiprocessor
systems it may be necessary to increase the loop-count:

 [unify_tclk_loopcount] '128'
```

The [unify_tclk_scantest] Variable

```
This string variable modifies the scan-path test used by XDS560 products
to validate the selection of JTAG Clock TCLK frequencies that were
selected by [unify_tclk_program] and [unify_tclk_frequency].

It instructs the emulator to skip the scan-test entirely or to use
the scan-test and decide to continue or surrender when it fails.

The variable [unify_tclk_scantest] has a string value with four options:

 'default'   - Use the default scan-test option (currently 'surrender').
 'nothing'   - Do nothing - skip the scan-test.
 'continue'  - Do scan-test, log result, continue after fail.
 'surrender' - Do scan-test, log result, surrender after fail.

The effect of these options are logged by the 560 software
and reported by XDSPROBE's -k option, so as to allow
investigation of the success and failure of these scan-tests.

As an example, when the scan-path is corrupted or broken it may
be necessary to request the emulator to skip the scan test:

 [unify_tclk_scantest]  'nothing'
```

The [unify_size_scantest] Variable

```
This string variable modifies the length of the scan-path tests used
by XDSPROBE for its `-b', `-i', `-g' options, and also used by XDS560
products to validate the selection of JTAG Clock TCLK frequencies that
were selected by the [unify_tclk_program] and [unify_tclk_frequency]
variables. All of these tests are specific to the standard
JTAG IR instruction scan-path and JTAG DR bypass scan-path.

This variable instructs the emulator modify the default length
from 16K bits to the given length, which is currently limited
to between 1K bits and 512K bits.

The length of the scan-path tests must be greater than the total length
of the JTAG IR instruction registers and JTAG DR bypass registers for
these operations to succeed.  The length of the scan-path tests can
have a noticeable effect on the speed of these operations at the
low TCLK frequencies used by Quickturn and IKOS hardware.

For values near or above 65536 bits to be effective
the [unify_bigscan] variable must also be set to YES.
This may become automatic in a later release.

The variable [unify_size_scantest] has a string value with four options:

 'default'         - Do scan-tests of the default length
 'shortest'        - Do scan-tests of the shortest length
 'longest'         - Do scan-tests of the longest length.
 '1024' - '524288' - Do scan-tests with a specific length.

As an example, when operating with a multiprocessor system with 1024
C55xx or C64xx devices, the scan-path tests must be longer than 1024
Icemaker IR registers (38 bits each) or DR registers (1 bit each):

 [unify_size_scantest]  '40000'
```

The [unify_size_irpath] and [unify_size_drpath] Variables

```
These string variables modify the JTAG IR instruction scan-path
and JTAG DR bypass scan-path size measurements used by
XDSPROBE for its `-b', `-i', `-g' options.

Similar measurements are also used by XDS560 products as part of
validating the selection of JTAG Clock TCLK frequencies that were
selected by [unify_tclk_program] and [unify_tclk_frequency].

These variables instruct the emulator to either measure the
total size of all the registers on the IR or DR scan-paths,
or to use the values provided by these variables.

The variables [unify_size_irpath] and [unify_size_irpath]
have string values with three options:

 'default'      - Do the default operation (currently measure).
 'measure'      - Measure the size of the IR instruction scan-path.
 '0' - '524288' - Specify the size of the DR bypass scan-path.
```

The [unify_bigscan] Variable

```
This boolean variable is used by XDS510 and XDS560 software
to enable big scans over 65536 bits in length and accept
the corresponding small reduction in performance.

The default operation improves performance by programming
only one of the two counter registers in the controller.
This limits the size of the scans to just below 65536 bits.

It may be used to aid in debugging new silicon by allowing
the XDS510 and XDS560 to be used to access proprietary
internal scan-paths that are much bigger than the
relatively short public scan-paths.

The variable [unify_bigscan] has a boolean value with two options:

  no  - Use only short scans and obtain improved performance.
  yes - Enable big scans over 65536 bits in length.

This variable affects all IR and DR scans, unlike the
related [unify_size_scantest] variable that affects
only scan-path tests specific to the standard JTAG IR
instruction scan-path and JTAG DR bypass scan-path.
```

The [unify_noamble] Variable

This boolean variable is used by XDS510 and XDS560 software to disable the internally calculated scan-path pre-ambles and post-ambles that select a single target device in a multi-processor system.

It may be used to aid in debugging problems with multiprocessor systems by allowing them to be reproduced by applying the board configuration file for a multiprocessor system to a system with a single target device.

The variable [unify_noamble] has a boolean value with two options:

```
no  - Use the internally calculated pre-amble and post-amble.
yes - Use the value of zero for the pre-amble and post-amble.
```

The [unify_clkmode] Variable

This string variable is used by XDS510 and XDS560 software to configure the normal operating mode of the software so as to improve performance at high TCLK frequencies and to enable operation at TCLK frequencies down to 1KHz.

This variable is commonly used with the Sourceless-EPK, and with Quickturn and IKOS hardware via the XDS510, because of its need for mauual configuration. This variable is hopefully never required with the self configuring XDS560.

The `slow clock mode' is used by XDS510 and XDS560 to enable operation at low TCLK frequencies down to just 1KHz by instructing the software to poll the ready signal on the Test Bus Controller. In this mode the default low frequency limit is 20000Hz. Operation can be extended down to just 1000Hz by explicitly specifying the actual low frequency limit. This information is used to modify internal timeout values, so as to ensure long JTAG scans at low TCLK frequencies are not interpreted as a hung scan-controller.

The `fast clock mode' is used by XDS510 and XDS560 to improve performance at high TCLK frequencies by instructing the software to skip certain polling operations on the Test Bus Controller.

The `automatic clock mode' is used to instruct XDS560 software only to make an automatic selection of the appropriate mode based on its measurement of the TCLK frequency.

The variable [unify_clkmode] has a string value with six options:

```
 'default'   - Use the default clock mode option.
 'automatic' - Use the automatic clock mode option.
 'fast'      - Use the fast clock mode option.
 'normal'    - Use the normal clock mode option.
 'slow'      - Use the slow clock mode option with
               the default low frequency limit.
 '1000'      - Use the slow clock mode option with
               an explicit low frequency limit.
```

When it is not defined the XDS510 and XDS560 use the default clock mode, which the XDS510 interprets as 'normal' and the XDS560 interprets as 'automatic'.

The [unify_slowclk], [unify_slowfrq], [unify_fastclk] and [unify_autoclk] Variables

```
These three boolean variables are considered out-of-date and replaced
by [unify_clkmode]. They are still supported by the software due to
their popularity. They are ignored if the newer variable is present.

The variable [unify_slowclk] has a boolean value with two options:

  no  - This implies the target clock is normal.
        USCIF3 will optimise for standard performance.
  yes - This implies the target clock is slow. USCIF3 will optimise
        for slow clock operation at the expense of performance.
        When used together with [unify_slowfrq], this choice
        works with low TCLK frequencies down to just 1KHz.

The variable [unify_slowfrq] has a numerical value.
It specifies the lowest TCLK frequency supported by [unify_slowclk].

  default - If not used then the lowest frequency
            supported by [unify_slowclk] is 20000Hz.
  number -  If used then it specifies in Hz the lowest frequency
            supported by [unify_slowclk].  This value is used to
            modify internal timeout values, so as to ensure long
            JTAG scans at low TCLK frequencies are not interpreted
            as a hung scan-controller. It has been tested at
            frequencies as low as 1000Hz.

The variable [unify_fastclk] has a boolean value with two options:

  no  - This implies the target clock is normal.
        USCIF3 will optimise for standard performance.
  yes - This implies the target clock is fast. USCIF3 will optimise
        for faster TCLK frequencies and have better performance.

The variable [unify_autoclk] has a boolean value with two options:

  no  - This implies operate in normal clock mode regardless
        of the clock frequency programmed into the PLL in the system
        unless instructed by the other two configuration variables.
  yes - This implies automatically configure itself to operate in
        slow clock/normal clock/fast clock mode based on the
        clock frequency programmed into the PLL in the system.

They have an obscure prioritisation scheme, but [unify_slowfrq]
alone does imply [unify_slowclk], and they will both override
the others when several are simultaneously defined.

When none are defined the XDS510 default clock mode is normal
clock and the XDS560 default clock mode is automatic.
```

The [pod_drvr] and [pod_port] Variables

```
These two variables are used by the XDS510 software to load and
address an adapter DLL file that was developed using the Sourceless-EPK.

The variable [pod_drvr] has a string value that is the file name
of the adapter DLL loaded by the XDS510 software to operate
a TI or third-party product instead of an XDS510.

 'oldtool.dll'   - The normal-mode XDS510 adapter in the USCIF archive.
 'newtool.dll'   - The block-mode XDS510 adapter in the USCIF archive.
 'pod510.dll'    - The normal-mode XDS510 adapter in the Sourceless-EPK.
 'pod510blk.dll' - The block-mode XDS510 adapter in the Sourceless-EPK.

 'dsk6x11pp.dll' - The adapter for TI's 6211 and 6711 Parallel Port DSK's.
 'pci5510.dll'   - The adapter for TI's 5510 PCI Target Board.
 'mdpjtag3.dll'  - The adapter for Black-Hawk DSP's USB 2.0 emulator.
 'iipcipod.dll'  - The adapter for Innovative Integration's PCI emulator.
 'podflexds.dll' - The adapter for DSP-Research's FleXDS PCI emulator.
 'flexdscli.dll' - The adapter for DSP-Research's FleXDS TCP/IP emulator.

The variable [pod_port] has a numeric value that is the port address
that the XDS510 software to be passed to the adapter DLL instead
of the port address provided by the debugger or utility software.

  0x01234567 - Up to eight hexadecimal digits that over-ride the port address.
```

The [pod_logmode], [pod_logfile] and [pod_logtype] Variables

```
These three variables are used to generate text formatted log-files when
using the XDS510 software and the Sourceless-EPK to develop adapter DLL's.
They are used to aid in the development of those adapter DLL's by
configuring the generation of text formatted log-files that document
the interaction of the XDS510 software and those adapter DLL's.

The variable [pod_logmode] has a boolean value:

  no     - Disable the generation of a adapter log-file.
  yes    - Enable the generation of an adapter log-file.

The variable [pod_logfile] has a string value:

 'Log-File.txt' - The file-name of the Sourceless-EPK adapter log-file.

The variable [pod_logtype] has a string value:

 'short' - Generate a short format (brief) adapter log-file.
 'long'  - Generate a long format (verbose) adapter log-file.
```

The [pod_blkmode] Variable

This boolean variable is used with adapters designed using the XDS510 software and Sourceless-EPK that are capable of operating in both normal-mode and block-mode. It is used to switch the adapter from using the default normal-mode over to block-mode.

The variable [unify_blkmode] has a boolean value with two options:

```
no  - Configure the adapter for normal-mode operation.
yes - Configure the adapter for block-mode operation.
```

The [reset_option], [probe_option], [error_option] and [build_option] Variables

These string variables are used when the `-e' option is used with the XDSRESET, XDSPROBE, XDSERROR and XDSBUILD utilities. The variables are used to provide extra command line options or all of the command line options for these utilities.

If the `-e' option has a sub-argument then it provides a suffix to the variable name. This allows multiple pre-defined options to be embedded in the board configuration file, and for one of them to be selected by the sub-argument to the option. For example:

```
use `-e' with XDSPROBE to select [probe_option]

use `-etest1' with XDSPROBE to select [probe_option_test1]
use `-etest2' with XDSPROBE to select [probe_option_test2]
use `-etest3' with XDSPROBE to select [probe_option_test3]
```

These variables also allow appropriate builds of the utility to be used in environments where C language IO is available, but the command-line interface is not convenient (a Windows short-cut) or does not exist (embedded systems).

Detailed information about the use of these variables is provided by the manuals for these utilities. The manuals are printed by the `-vh' option of these commands.

Appendix – The Terms and Definitions

510-class

This term refers to TI's ISA-bus XDS510 emulator and to those products developed using TI's Sourceless-EPK for 510-class products.

560-class

This term refers to TI's PCI-bus XDS560 emulator and to those products developed using TI's Source-EPK for 560-class products.

Sourceless EPK

The TI Sourceless Emulation Porting Kit for 510-class products (TI part number EPKTMDX3240B51) allows TI customers or third parties to expand the emulation capability of Code Composer Studio. This may be implemented without any TI product specific knowledge or technology and may take the form of a general emulation controller that supports many target systems or a DSP board that contains a Test Bus Controller. All implementations must contain a TI TBC (SN74ACT8990) or TBC compatible device. The Sourceless EPK allows developers to use the transport mechanism or bus of their choice to access the TBC. This allows Code Composer Studio to be used in more environments than are directly provided by TI. It requires use of TI's SN74ACT8990 Test Bus Controller to ensure compatibility with the DSP and ARM drivers provided with Code Composer Studio 2.00 and 2.10

TBC

This term normally refers to the SN74ACT8990 Test Bus Controller that is used in many TI EVMs, DSKs and is required for the Sourceless EPK. This is a parallel access device used by TI's emulation software to control the serial JTAG link and one or more TI DSPs.

TCLK

This term refers to the JTAG Test Clock signal that is used to clock the scan-path connecting the TBC and target DSP or ARM micro-controller.

CCS

This term refers to both the 2.00 and 2.10 releases of Code Composer Studio.

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 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.

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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265