A diagnostic utility for TI and 3rd-Party JTAG hardware

The user's guide to

DBGJTAG

```
       20-pin                         14-pin                         14-pin
      TI header                      TI header                      TI header
    (14 signals)                   (10 signals)                    (8 signals)

      ╔═══════╗                      ╔═══════╗                      ╔═══════╗
 TMS  o║1     2║o nTRST        TMS  o║1     2║o nTRST        TMS  o║1     2║o nTRST
 TDI  o║3     4║i TDIS         TDI  o║3     4║i TDIS         TDI  o║3     4║x NC
 TVD  i║5     6║x ╫╫╫╫         TVD  i║5     6║x ╫╫╫╫         TVD  i║5     6║x ╫╫╫╫
 TDO  i║7     8║x GND0         TDO  i║7     8║x GND0         TDO  i║7     8║x GND0
TCLKR i║9    10║x GND1       TCLKR  i║9    10║x GND1          NC  x║9    10║x GND1
TCLKO o║11   12║x GND2       TCLKO  o║11   12║x GND2       TCLKO  o║11   12║x GND2
 EMU0 b║13   14║b EMU1        EMU0  i║13   14║i EMU1        EMU0  o║13   14║o EMU1
nSRST o║15   16║x GND3               ╚═══════╝                      ╚═══════╝
 EMU2 b║17   18║b EMU3
 EMU4 b║19   20║x GND4        GNDx   These are the three to nine signal grounds.
      ╚═══════╝                 ╫   These are the header orientation keys.

                                i   Input signals from the target system.
                                o   Output signals to the target system.
                                b   These are bi-directional signals.
                                x   These are not signals.
```

Texas Instruments H. Roland Hoar rhoar@ti.com

This document describes the benefits
and features of the following software:

DBGJTAG and USCIF35 Version 35.32.0.0

This document has the following format:

Page height / width	9.25 / 7.5 inches
Text height / width	8.0 / 5.5 inches
Header	The section number & name, page number
Footer	None
Level 1 headings	Arial 14pt bold & numbered
Level 2 headings	Arial 12pt bold & numbered
Level 3 headings	Arial 12pt bold & numbered
Level 4 headings	Arial 10pt bold text only
Proportional text	Times New Roman 10pt
Typewriter text	Courier New 9pt

This document has the following history:

Version 0.1	1st Draft	7th May 2006
...	...	...
Version 0.2	13th Draft	15th May 2008

Table of Contents

Outstanding Issues

1.3.2 Examples Requiring an Emulator

The list of examples at the end of this section is not yet done.

There are no examples for the informal major options: -M, -A and -I.

 3.3.1 The Introduction and 5.3 The Selection of TCLKO Frequency

The frequency selection description is moved to an appendices section.

The text in both areas must be reworked.

3.7 Test the Emulator with [-T]

This section is only half done.

3.8 Test the Cable and Pod with [-U]

This section is only half done.

4.1 The Miscellaneous Actions with [-M]

This entire section is only half done.

Can little-used commands be deleted?

Can individual commands be relocated to other options?

4.2 The Experimental Device Analysis with [-A]

The implementation was just updated to support routers - the description should be updated.

Do a top-down design – document a useful prototype [-A] command and then implement it.

4.3 The Experimental Interactive Shell with [-I]

The old implementation of this command has just been deleted.

Do a top-down design – document a useful prototype [-I] command and then implement it.

 5.3 The Selection of TCLKO Frequency

This section has just been relocated from 3.3 Select the Frequency with [-F].

The beginning of each of those sections should be compared during
review to ensure their content is appropriate for the partitioning.

5.5.2 The Examples

This section lacks Kamal's examples of support for CS/DAP.

1 The Introduction

1.1 The Foreword

1.1.1 The Format

This users guide adopts the smaller width and height layout used by O'Reilly, Prentice Hall and Addison Wesley. It allows PDF versions to be displayed full size on a monitor side-by-side with application software. It also allows printed versions to be shelved and transported with those other reference works. The layout also has pages with brief header text and no footer text.

The introductory section is ordered by benefits (what the user wants to do) and then details the corresponding features (that are used to obtain those benefits). The main section is a description of the minor and major commands and their parameter lists. The appendices are a description of all the reference material required by the introductory and main sections. Those two sections are shortened by deliberately locating all reference material in the appendices:

- 5.1 The Standard Emulator Signals and Cables
- 5.2 The Frequency of TCLKO and TCLKR
- 5.3 The Selection of TCLKO Frequency
- 5.4 The Format of Board Config' Files
- 5.5 The Examples of Board Config' Files
- 5.6 The Standard Board Config' Variables
- 5.7 The XDSRESET and XDSPROBE Utilities
- 5.8 The Design of the DBGJTAG Utility

The result is that the introduction is rather short instead of long, the main text is long instead of very long, and the appendices are long and useful – they may be read as stand-alone descriptions of specific topics involving the DBGJTAG utility and USCIF35 software.

1.1.2 The Contributors

The DBGJTAG utility was first developed by Roland Hoar.

The UNIX ports of DBGJTAG and USCIF35 were first developed by Ed Fewell.

The ACT8990 TBC tests used by the [-T control]
command were first developed by Yihan Jiang.

The ECOM interface for USCIF35 that connects DBGJTAG
with 560-class emulators was first developed by Huimin Xu.

The DSP and ARM analysis routines used by the [-A scanpath] command
are derived from work done by TI's DSP and ARM emulation driver teams.

Many of the board config' file examples were provided by Huimin Xu and Kamal Nehal.

1.1.3 The Copyright

This document and the DBGJTAG, XDSPROBE and XDSRESET utilities and USCIF35 software are Copyright (c) 1990-2006 by Texas Instruments Incorporated.

1.2 The Overview

1.2.1 The Utilities

There are currently three utilities distributed with the USCIF35 software stack. All three are implemented as command-line parsers mounted on a shared library. The DBGJTAG utility has a full-feature parser that accesses all features of the library. The XDSRESET and XDSPROBE utilities are now simple parsers that access a few features of the library so as to emulate their original stand-alone implementations.

DBGJTAG	A new utility used to develop, test and evaluate TI and 3rd party hardware that uses JTAG scan-controllers, cables, routers and targets.
XDSPROBE	An old utility used to reset and test JTAG scan-controllers and targets.
XDSRESET	An old utility used to reset JTAG scan-controllers.

The DBGJTAG utility supports all of the capabilities of both XDSRESET and XDSPROBE. The development of those older utilities has been discontinued. The older utilities are still provided to allow scripts that use their command-line options to remain compatible with new software releases. Users should adopt DBGJTAG not only for its advanced features, but also to perform the simple resets and tests supported by the old utilities. An appendix to this user guide details the upgrade from XDSRESET and XDSPROBE options to the equivalent DBGJTAG options.

This manual is a description of the benefits and features of the current version of DBGJTAG. The effort of describing the utility in a logical and user-friendly manner led the author to re-work several features. This manual is an accurate guide to that version, and likely a good guide to later versions. If the user has an older version of DBGJTAG and USCIF35 installed then they should be aware that the current version is known to be compatible across releases 2.2 - 3.3 of the CC Setup and CC Studio suite.

1.2.2 The Benefits

This utility is used to develop, test and evaluate the USCIF35 software stack and TI and 3rd party hardware that use JTAG scan-controllers, cables, routers and devices:

- Hardware tests on emulators are supported by the [-T] and [-U] options.
- Scan tests on routers and devices are supported by the[-S] and [-R] options,
- Frequency tests on routers and devices are supported by the [-F] and [-G] options.
- White-box tests on software components are supported by [-F], [-B] and [-E] options.

This utility has many benefits provided by a large set of command-line options. The following descriptions allow the user to browse the benefits before wading through the commands and parameters in the description of the command-line options.

Portability

The DBGJTAG utility and the USCIF35 software stack are available as builds for the following environments: Windows (Intel), Solaris (Sparc), Linux (32-bit & 64-bit – Intel), Mac OS X (Power-PC & Intel - universal binary)

Harmless default behaviour

The default action, when no command-line options are used, does not touch any emulator or scan-path hardware. The default action is to print the version information plus brief help about the supported commands and the availability of this user guide.

Built-in brief help for all commands

All of the major options have a '-X help' command that retrieves brief help listing the commands and parameters available for that option. At a later date that help may be extended to describing the acceptable parameter values. This help is intended to provide users who have already read this manual with reminders of the commands and parameters.

Accurate reporting of common errors

On systems that support advanced error detection (such as all 560-class products and advanced 510-class products) this utility will provide accurate and 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.

Selecting board config' files

This utility has a lower-case [-f] option that selects a board config' file using either the modern format or the legacy format. The file may contain optional config' variables and an optional scan-path description. The config' variables are name-value pairs that provide a general purpose mechanism to customise the behaviour of the USCIF35 software stack. The scan-path description details the naming and arrangement of routers and devices on the scan-path for use by commands that access the scan-path. The best known examples of board config' files are generated by CC Setup and used by CC Studio. Their location and name is such that '-f BrdDat/ccBrd0.dat' is likely sufficient to have DBGJTAG use the same file and emulator hardware as the CC Setup and CC Studio suite.

Selecting 100/510-class and 560-class emulators

This utility has a [-X adapter] command to select 100/510-class hardware and a [-X program] command to select 560-class hardware. The lower-case [-d] and [-p] options are synonyms for a subset of of features in the [-X] option.

Resetting 100/510-class and 560-class emulators

This utility has a [-Y reset] command that resets scan-controllers and operates the nTRST JTAG reset signal, and a [-Y system] command that operates the nSRST system reset signal. The lower-case [-r], [-r -v] and [-s] options are synonyms for specific features of these commands.

Configuring cables

This utility has a [-Y emupins] command that configures the boot-modes of the EMU signals in the cable, and a [-Y jtagpins] command that configures the timing and loop-back of the JTAG signals in the cable. The full capabilities of these commands can also be accessed by variables in the [-f] option's board config' file, which empowers any product (such as the CC Setup and CC Studio suite) that can generate and accept board config' files.

Emulator and cable FPGA/CPLD components

This utility has a [-D] option that is a tool for translating FPGA/CPLD data and programming those components, which are used to implement emulators and their cables.

Measuring JTAG IR and DR path-lengths

This utility has a [-S pathlength] command that measures the lengths of the JTAG instruction register and JTAG data register, of routers and devices on the scan-path. This test will also interpret the results to detect stuck-faults and link-delay errors, and compare the values against known valid combinations of JTAG instruction register and data register lengths. This comparison comprehends both homogeneous systems with multiple similar devices, and heterogeneous systems with multiple dis-similar devices. This command has surprising value when used in combination with the [-R] option described below, because path-length values can provide confirmation that a specific sub-path or a specific set of sub-paths have been accessed.

Standard JTAG IR and DR test-patterns

This utility has [-S brokenpath] and [-S integrity] commands for performing basic scan-path tests using fixed data patterns. They may be run once, repeated a fixed number of times, or run continuously.

Custom JTAG IR and DR test-patterns

This utility has a [-S givendata] command for flexible scan-path tests with user selected test patterns. It may be run once, repeated a fixed number of times, or run continuously. The continuous method will report data errors, but not terminate operation, allowing it to be used as a pattern generator when investigating faulty target systems.

Static operation of JTAG routers

This utility has a [-R routelist] command that is typically used in combination with the [-f] and [-S] or [-G] options. It provides a method of accessing a fixed sub-path, or set of sub-paths, through the routers and devices described in the board config' file specified by the [-f] option. The [-S] or [-G] options are then used to apply a test to that specific scan-path.

Dynamic operation of JTAG routers

This utility has [-R matrixlist] command that is also used with the [-f] and [-S] or [-G] options. It provides a method of accessing a dynamic sequence of sub-paths, or sets of sub-paths, through the routers and devices described in the board config' file specified by the [-f] option. The [-S] or [-G] options are then used to test that sequence of scan-paths.

Selecting a TCLK frequency

This utility has a [-F clock] command that controls the frequency of the JTAG TCLK signal (generated by a programmable PLL) used by all 560-class emulators and advanced 510-class emulators. It supports both the selection of specific TCLK frequencies and the use of an auto-ranging algorithm to select the optimum TCLK frequency for best performance. The full capabilities of this command can also be accessed by variables in the [-f] option's board config' file, which empowers any product (such as the CC Setup and CC Studio suite) that can generate and accept board config' files. This command also has a [whitebox=yes] parameter that executes a copy of the command in the utility instead of the one loaded into the emulator hardware.

Retrieving a TCLK programming log-file

This utility has a [-F inform] command for retrieving the internal log-file of PLL programming actions and scan-path test results that are stored in 560-class and advanced 510-class products. Those log-files record the actions performed by the emulator and measurements taken by the emulator, when selecting a TCLK frequency based on the [-F clock] command or variables placed in the [-f] option's board config' file.

Combined TCLK frequency and scan-path tests

This utility has a [-G range] command that is an advanced tool for performing combined JTAG TCLK frequency and scan-path tests over the full range of the programmable PLL in 560-class and advanced 510-class products. The frequency tests both select the TCLKO output frequency and measure the TCLKR return frequency.

Evaluate signal quality problems in the target system

This utility has a [-G range] command that can be used to evaluate problems in the target system, or the emulator's cable and pod; or the emulators own TCLK generation and measurement. Its use is especially valuable in that not only the occurrence of failures, but also the patterns of success/failure, indicate subtle problems such as incorrect termination of JTAG signals.

Analyse routers and devices on the JTAG scan-path

This utility has an experimental [-A scanpath] command that will compare the scan-path described in a board config' file with the actual routers and devices connected to the emulator.

Test JTAG scan-controller hardware

This utility has a [-T control] command that applies standard tests to the scan-controller hardware. They may be run once, repeated a fixed number of times, or run continuously.

Operate external test hardware

This utility has a [-U test460] command that operates the ‘Test460’ hardware and its embedded firmware. The Test460 consists of three JTAG operated components - cross-bar, clock switch and error generator. The crossbar connects an emulator under test to a choice of one internal target (within the Test460) and up-to four external targets. The switch drives an emulator under test with the emulator’s own clock, or a wide-range programmable clock source generated within the Test460. The error generator stimulates an emulator under test with lost target power; dead JTAG clocks and cable breaks near the target. An extra JTAG connector on the Test460 allows the cross-bar, clock switch and error generator to be controlled by the [-U test460] command applied to an independent emulator.

Parse and generate board config’ files

This utility has a [-B] option that is used to perform white-box testing of the parsing and generation of board config’ files in either the modern or legacy format. This option does not require an emulator, routers or devices to be installed. The [-B parse] command simply parses the file and creates an equivalent data structure, reporting any errors that are found during these operations. The [-B legacy] and [-B modern] commands will additionally use the data structure to generate board config’ files in the requested format. The parsing and generation of these files is based on characteristics of individual devices and families of devices that are stored in alias and family config’ files specified by additional parameters.

Test serial and hierarchical scan-path management

This utility has a [-B test] command that is used to perform white-box testing of the USCIF35 software stack’s management of serial and hierarchical scan-paths. This option does not require an emulator, routers or devices to be installed. This command runs connect/disconnect, select/de-select and preamble/post-amble tests based on the scan-path description from a board config’ file, while bypassing the low-level control routines of the hierarchical and serial management software that would access the emulator.

Apply config’ variables at the command line

This utility has a [-V apply] command that is used to directly apply one or more arbitrary config’ variable names and values at the command-line instead of applying them indirectly as items in the [-f] option’s board config’ file.

Retrieve merged config’ variable values

This utility has a [-V merge] command that retrieves the final set of config’ variables that are applied by the utility after merging variables from the board config’ file, command-line options that generate variables, and the [-V apply] command that explicitly uses variables.

Explain error numbers

This utility has [-E single] command that retrieves the error title and text strings associated with any of the (greater than three hundred) error numbers in the USCIF35 software stack that is used by the DBGJTAG utility, and the CC Studio and CC Setup suites.

Test error reporting

This utility has [-E range] command that used to perform white-box testing of the error reporting mechanism. This option does not require an emulator, routers or devices to be installed. This command retrieves error title and text strings associated with ranges of error numbers in the USCIF35 software stack. It does so by using the same interface as does the CC Setup and CC Studio suite.

Parse and generate error config' files

This utility has a [-E] option that is used to perform white-box testing of the parsing and generation of error config' files. This command does not require an emulator, routers or devices to be installed. The [-E parse] command simply parses the file and creates an equivalent data structure, reporting any errors that are found during these operations. The [-E errors] command with the [filetype=config] parameters and [filetype=header] parameters will additionally use the data structure to generate error config' files and 'C' language error header files.

1.2.3 The Syntax

The user tolerant syntax

White-space can be used (or not) around any sub-argument, command, parameter name, or parameter value. White-space can also be used (or not) around any meta-character (the comma, equal, plus and minus characters) other than the option itself.

The minor options

These are single lower case letters without built-in help. Most minor options have equivalent major options that provide option-specific brief help.

- The minor options have the following generic syntax where the character 'z' is used as a place-holder for a minor option:

  ```
  [-z] | [-z value]
  ```

- Some minor options never have a sub-argument:

  ```
  [-r -v]
  ```

- Any minor option with a sub-argument always requires it:

  ```
  [-d driver -p address]
  ```

The major options

These are single upper case letters and always require a sub-argument. The sub-argument is a comma-separated list of items. The first item is a 'command' word and the following items are 'parameter=value' pairs. The command and parameters may be abbreviated to four letters.

- The major options have the following generic syntax where the character 'Z' is used as a place-holder for a major option:

  ```
  [-Z command,parameter1=value1,parameter2=value2,...]
  ```

- The major options always have an option-specific brief help command:

```
[-R help]
```

- The path-length measurement command typically has no parameters:

```
[-S pathlength]
```

- The given-data scan-test command typically has one parameter:

```
[-S givendata,repeat=number]
```

- The clock frequency command requires several parameters in any order:

```
[-F clock,program=value,beginning=value,frequency=value]
```

- The route-list command has parameter values that are lists delimited by '+' characters:

```
[-F routelist,devices=item1+item2+item3+item4]
```

1.3 The Examples

The following examples use the [-d] and [-p] options to select the emulator. . The 'xds100pp.dll' adapter for the TI XDS100pp emulator uses ports 0/1/2 as aliases for physical ports 0x378/278/3BC to select one of three emulators. The 'xds510.dll' adapter for the TI XDS510 emulator uses ports 0/1/2/3 as aliases for physical ports 0x240/280/320/340 to select one of four emulators. The 'xds560.out' program for the TI XDS560 emulator uses ports 0/1/2/3 to select one of four emulators.

The [-f] option allows a board config' file to be used to select either: a 100/510-class emulator named by the sepk.pod_drvr and sepk.pod_port variables; or a 560-class emulator named by the uscif.ecom_drvr and uscif.ecom_port variables. The values provided by the [-f] option are over-written by those provided by the [-d] and [-p] options.

The compatible Blackhawk 510-class and 560-class emulators are used by substituting their adapter/program files in the values of the '..._drvr' variables and [-d] options, and substituting their port addresses in the values of the '..._port' variables and [-p] options.

1.3.1 Examples Without an Emulator

Request the brief help

```
dbgjtag -h
```

Request information about an error number

Request the error title and explanation corresponding to a specific error number.
Disconnecting the emulator cable near its JTAG header causes error number -183:

```
dbgjtag -E single,number=183
```

Request information about many error numbers

Request the error text corresponding to every error number.
The current range of DBGJTAG and USCIF35 error numbers is -100 to -999:

```
dbgjtag -E range,lowest=100,highest=999
```

Generate the ‘C’ language header file error mnemonics

Request the generation of the ‘C’ language header file including nmemonics for every error number listed in the standard error config’ file used by USCIF35:

```
dbgjtag -E errors,,filetype=header,
           input=xdserror.cfg,output=xdserror.h
```

Parse a board config’ file and report any errors

Request the parsing and conversion of a board config’ file to an equivalent data structure, reporting the occurrence of any errors found. The named board config’ file is the one used by the CC Setup and CC Studio suite. The named family and alias config’ files are those distributed with DBGJTAG and USCIF35:

```
dbgjtag -B parse,input=BrdDat\ccBrd0.dat,
           family=xdsfamily.cfg,alias=xdsalias.cfg,
           verbose=yes
```

The board config’ file named ‘BrdDat\ccBrd0.dat’ is generated by the CC Setup utility that is a component of every CC Studio installation. An alternative is any file in ‘5.5.2 The Examples’.

Test or convert a board config’ file with family and alias files

Request the generation of a board config’ file in a specific format. A data structure is created using the contents of the input, family and alias files, and then used to create the output file. The input file is potentially: any file used to test file parsing and generation; any user written file to support silicon development; or any file created by the CC Setup and CC Studio suite. The named family and alias files are those distributed with DBGJTAG and the USCIF35 software.

The following example outputs a file in the legacy format which does not support routers and sub-paths. Only two of the eight board config’ file descriptions of ‘5.5.2 The Examples’ meet that restriction. Both ‘A simple board config’ file for OMAP1710’ and ‘A simple board config’ file for Janus’ are acceptable as input files replacing the placeholder name ‘without_subpaths.cfg’.

```
dbgjtag -B legacy,
           output=legacy.cfg,
           input=without_subpaths.cfg,
           family=xdsfamily.cfg,alias=xdsalias.cfg
```

The following example outputs a file in the modern format which does support routers and sub-paths. All eight of the board config’ file descriptions of ‘5.5.2 The Examples’ are acceptable as input files replacing the placeholder name ‘maybe_subpaths.cfg’.

```
dbgjtag -B modern,
           output=modern.cfg,
           input=maybe_subpaths.cfg,
           family=xdsfamily.cfg,alias=xdsalias.cfg,
           crushed=yes,exclude=yes,indent=yes
```

Test scan-path management with a board config’ file

Request the testing of the hierarchical and serial scan-path management within the USCIF35 software against a specific scan-path description by applying DBGJTAG’s built-in test suite.

```
dbgjtag -B test,
           input=planets_complex.cfg,output=results.txt,
           family=xdsfamily.cfg,alias=xdsalias.cfg,
           verbose=yes
```

The board config’ file descriptions of ‘5.5.2 The Examples’ show ‘A complex board config’ file for Sedna-Neptune3G’ with a file named ‘planets_complex.cfg’. It has two routers each having six sub-paths, each with a different device. The second level router is located on an extra sub-path of the first level router which is the only device located on the main scan-path.

Apply an arbitrary set of configure variables

Apply an arbitrary set of configure variables.

```
-V apply,name1=value1,name2=value2,...
```

List the merged configure variable values from all sources

List the variables and values that will be passed to the emulator after the command line and board configure file have been merged.

```
-V merge
```

Request a xxx

Generate a C source file from a FPGA/CPLD data file.

```
-D source,type=title,version=number,
          input=filename.ttf,output=filename.c,
          verbose=boolean
```

Generate a raw binary file from a FPGA/CPLD data file.

```
-D binary,input=filename.ttf,output=filename.rbf,
          verbose=boolean
```

Generate a encoded/decoded data file using a chosen algorithm.

```
-D crush,input=filename.txt,output=filename.txt,
         action=huffman/crc32,
         direction=encode/decode,
         verbose=boolean
```

1.3.2 Examples Requiring an Emulator

Request a verbose controller reset with the [-d -p] options

Request a verbose controller reset,
selecting it with the [-d -p] options and resetting with the [-r -v] options:

```
dbgjtag -d xds560.out -p 0 -rv

dbgjtag -d xds510.dll -p 0 -rv

dbgjtag -d xds510.dll -p 0x240 -rv

dbgjtag -d xds100pp.dll -p 0 -rv

dbgjtag -d xds100pp.dll -p 0x378 -rv
```

Request a verbose controller reset with the [-X -Y] options

Request a verbose controller reset,
selecting with the [-X] option and resetting with the [-Y] option:

```
dbgjtag -X program,driver=xds560.out,ioport=0
        -Y reset,logfile=yes

dbgjtag -X adapter,driver=xds510.dll,ioport=0
        -Y reset,logfile=yes

dbgjtag -X adapter,driver=xds100pp.dll,ioport=0
        -Y reset,logfile=yes
```

Request a system reset via a revision-D cable

Request a system reset from an emulator with a Revision-D cable,
selecting it with the [-d -p] options and resetting with the [-Y] option:

```
dbgjtag -d xds560.out -p 0
        -Y system,signal=pulse
```

Request a boot-mode via a revision-D cable

Request specific values of JTAG nTRST boot-mode and power nTVD boot-mode
values for the EMU1-0 pins from an emulator with a Revision-D cable.
This will be used in combination with the -F, -S and -R option:

```
dbgjtag -d xds560.out -p 0
        -Y emupins,jtagboot=01,powerboot=10
```

Request JTAG loop-back and JTAG timing on a revision-D cable

Request a specific configuration of the JTAG pins (loop-back TDO / TDI and falling-edge TMS / TDO) from an emulator with a Revision-D cable. This will be used in combination with the -F, -S and -R option:

```
dbgjtag -d xds560.out -p 0
        -Y jtagpins,loopback=total,tdoedge=fall
```

Request a specific TCLK frequency selection

Request a specific TCLK frequency with an exact value:

```
dbgjtag -d xds560.out -p 0
        -F clock,program=specific,frequency=10.368Mhz
```

Request an automatic TCLK frequency selection

Request an automatic TCLK frequency, with the default lower limit of 500Khz and an explicit upper limit compatible with HS-RTDX:

```
dbgjtag -d xds560.out -p 0
        -F clock,program=automatic,frequency=35.0Mhz
```

Request an adaptive TCLK frequency on a revision-D cable

Request an adaptive TCLK frequency, with the maximum frequency applied to the emulator synchroniser:

```
dbgjtag -d xds560.out -p 0
        -F clock,program=adaptive,frequency=48.0Mhz
```

Retrieve the TCLK programming log-file

Retrieve the TCLK log-file as an enhancement of either the prior three examples:

```
dbgjtag -d xds560.out -p 0
        -F clock,...
        -F inform,logfile=yes
```

Request a combined frequency and scan-path test

Request a combined frequency/scan-path test between two frequencies:

```
dbgjtag -d xds560.out -p 0
        -G range,lowest=2.5MHz,highest=40.0Mhz
```

Request a simple scan-path length measurement

Request a simple path-length measurement on the default scan-path:

```
dbgjtag -d xds560.out -p 0
        -S pathlength
```

Request a simple scan-path broken-path test

Request a simple broken scan-path test on the default scan-path:

```
dbgjtag -d xds560.out -p 0
        -S brokenpath
```

Request a simple scan-path integrity test

Request a simple scan-path integrity test on the default scan-path:

```
dbgjtag -d xds560.out -p 0
        -S integrity
```

Request a simple scan-path test with a custom data value

Request scan-path tests, on the default scan-path,
repeated 256 times with a literal value of 32-bit data:

```
dbgjtag -d xds560.out -p 0
        -S givendata,repeat=256,literal=0xcc3355aa
```

Request a simple scan-path test with a custom data value

Request scan-path tests, on the default scan-path,
repeated forever with a short value expanded to 32-bits of data:

```
dbgjtag -d xds560.out -p 0
        -S givendata,repeat=forever,expand=0xa
```

Request a scan-path length measurement and test on named sub-paths

Request a path-length measurement on DR/IR scan-paths described
in a board config' file that selects devices named on the command-line:

```
dbgjtag -d xds560.out -p 0
        -f omap2420.cfg
        -S pathlength
        -R routelist,subpaths=sub_3+sub_0
```

The board config' file descriptions of '5.5.2 The Examples' show the file named 'omap_2420.cfg' with four sub-paths. Only the two sub-paths named on the command line are accessed. Those sub-paths access an arm7 and an arm11 device. The lengths of the router and two ARM's will be measured. This example can be modified to use the [-S brokenpath / integrity / givendata] commands shown in the prior 'simple scan-path test' examples.

Request a scan-path length measurement and test on named devices

Request a path-length measurement on a route of DR/IR scan-paths described
in a board config' file that selects sub-paths named on the command-line:

```
dbgjtag -d xds560.out -p 0
        -f planets_simple.cfg
        -S pathlength
        -R routelist,devices=arm7+arm9+arm11
```

The board config' file descriptions of '5.5.2 The Examples' show A simple board config' file for Sedna-Neptune3G with a file named 'planets_simple.cfg'. It has two routers having five sub-paths, each having a different device. Only the sub-paths for the three devices named on the command line are accessed. The lengths of the two routers and three ARM's will be measured. This example can be modified to use the [-S brokenpath / integrity / givendata] commands shown in the prior 'simple scan-path test' examples.

Request a scan-path length measurement and test on router descendants

Request a path-length measurement on a route of DR/IR scan-paths described in
a board config' file that selects descendants of a router named on the command-line:

```
dbgjtag -d xds560.out -p 0
        -f tomahawk.cfg
        -S pathlength
        -R routelist,hierarchy=descendant,devices=icepick
```

The board config' file descriptions of '5.5.2 The Examples' show the file named 'tomahawk.cfg' with a router having six sub-paths, each having one c64plus device. All the descendant sub-paths of the router named on the command line are accessed. The lengths of the router and all six DSP's will be measured. This example can be modified to use the [-S brokenpath / integrity / givendata] commands shown in the prior 'simple scan-path test' examples.

Request a xxx

Select a matrix of DR/IR scan-paths from a board config' file.

```
dbgjtag -d xds560.out -p 0
        -f scan-path.cfg
        -R matrixlist,devices=list,subpaths=list,
           method=string,hierarchy=string,
           reorder=string,repeat=number,
           noscan=boolean,verbose=boolean
```

Request a xxx

Do not program or query any of the FPGA/CPLD's.

```
-D nothing
```

Query the named FPGA/CPLD to report the version of its VHDL code.

```
-D version,type=title,notouch=boolean,verbose=boolean
```

Erase the named FPGA/CPLD.

```
-D erase,type=title,notouch=boolean,verbose=boolean
```

Program the named FPGA/CPLD using the default internal VHDL code.

```
-D internal,type=title,notouch=boolean,verbose=boolean
```

Program the named FPGA/CPLD using VHDL from source/binary/data files.

```
       -D external,type=title,data=filename.c/rbf/ttf,
                    notouch=boolean,verbose=boolean
```

Request a scan-controller test

Request a scan-controller test repeated 256 times:

```
       dbgjtag -d xds510.dll -p 0
               -T control,repeat=256
```

Request a xxx

Test a cable and pod using the Quad Router hardware with Test460 VHDL.

```
       dbgjtag -d xds560.out -p 0
               -U test460,
                  pathselect=string,
                  frequency=value,
                  sendclock=string,
                  inputclock=cable(s),
                  returnclock=cable,
                  erasefail=boolean,
                  typefail=string,
                  waitfail=string,
                  whenfail=string,
                  delayfail=number,
                  occurfail=number,
                  emu0pin=cable(s),
                  emu1pin=cable(s),
                  switchport=string,
                  verbose=boolean
```

Request a xxx

Minor options

```
       dbgjtag -f

       dbgjtag -e

       dbgjtag -o

       dbgjtag -s

       dbgjtag -w
```

2 The Minor Options

The minor command-line options are represented by single lower case letters and their syntax is detailed in the section '1.2.3 The Syntax'. They either lack a sub-argument, or always require a sub-argument. The minor command line options are implemented for pragmatic reasons rather than absolute necessity. They provide:

- A measure of familiarity to users of XDSRESET and XDSPROBE:
- Industry standard access to brief help: [-h]
- Accessory features that assist major commands: [-d -p -f -e -o -w]
- Shortcuts to selected major commands and parameters: [-r -v -s]

2.1 *Obtain Brief Help with [-h]*

This option has no sub-argument. It is the standard option for brief help on modern operating systems and the likely first choice of an inquisitive user. It retrieves the version number and the build date, plus a brief list of the minor and major options, and the file name of this users guide.

`[-h]`	Request brief help.

This option also has the following characteristics: it does not touch hardware; it does not require an emulator to be installed; it overrides other options on the command-line; it is implied if there are no options at all on the command-line.

2.2 Reset the Emulator with [-r] and [-v]

These options have no sub-arguments. They are used to reset the emulator's JTAG scan-controller, initialise the emulator's cable control logic, and reset the JTAG interface. The target connector will be placed in the JTAG Test-Logic-Reset state and the active-low JTAG nTRST signal will be asserted for several milliseconds.

`[-r]`	Request a controller reset.
`[-v]`	Request verbose output when controller is opened or reset. If not used then those actions are silent.

If the emulator is a 560-class product or advanced 510-class product then its JTAG TCLK signal will be programmed to the default frequency, which may be specified by a config' variable, and its internal FPGA's will be programmed with the default VHDL, built into the emulator program itself. In the case of TI's XDS560 product, the frequency is 500kHz and the program is 'xds560.out'.

The [-r] and [-v] options are actually synonyms for features of the [-Y reset] command:

`[-r]`	`[-Y reset,logfile=no]`	Request silent reset.
`[-rv]`	`[-Y reset,logfile=yes]`	Request verbose reset.

2.3 *Select the Emulator with [-d] and [-p]*

The [-d] option requires a filename as its sub-argument. It is used to select an emulator by providing the name of the emulation adapter (with a *.dll extension) required by 100/510-class emulators, or the name of the emulation program (with a *.out filename extension) required by 560-class emulators. The classes of emulator require different communication methods and the filename extension that indicates the method to be used.

The [-p] option requires a decimal or hexa-decimal port address as its sub-argument. It is used to select one of several similar emulators installed in the same host by providing the location of the emulator.

The [-d] and [-p] options provide simple alternatives to the [-f] option which requires emulator selection to be made by the values of config' variables placed in a board config' file. If the [-d] and [-p] options are not used to provide the name and port address then they must be provided with sepk.pod_drvr and sepk.pod_port and uscif.ecom_drvr and uscif.ecom_port variables respectively in the board config' file selected by the [-f] option. When the [-d] and [-p] options are used then the variables are ignored.

The driver names 'xds100pp.dll', 'xds510.dll' and 'xds560.out' are used by the TI XDS100pp, XDS510 and XDS560 emulators. If filename extensions are not provided then the utility will choose by searching the available driver files. Thus the XDS100pp can be selected by [-d xds100pp], the XDS510 by [-d xds510], and the XDS560 by [-d xds560].

The port addresses 0x0 - 0x3 are used by the TI XDS560 emulator. The port addresses 0x240, 0x280, 0x320 and 0x340, are used by the TI XDS510 emulator, however its adapter accepts 0x0 - 0x3 as aliases for those addresses. The port addresses 0x378, 0x278, 0x3BC, are used by the TI XDS100pp emulator, however its adapter accepts 0x0 - 0x2 as aliases for those addresses.

The [-d] and [-p] options are actually synonyms for features of the [-X reset] command:

```
[-X adapter,driver=xds100pp.dll,ioport=0]

[-X adapter,driver=xds510.dll,ioport=0]

[-X program,driver=xds560.out,ioport=0]
```

These examples using [-d] and [-p] are equivalent to those for the [-f] option:

The TI 'XDS100pp parallel-port emulator' can be operated using:

```
dbgjtag -d xds100pp.dll -p 0x0 ...
```

The TI 'XDS510 ISA-bus emulator' can be operated using:

```
dbgjtag -d xds510.dll -p 0x0 ...
```

The TI 'XDS560 PCI-bus emulator' can be operated using:

```
dbgjtag -d xds560.out -p 0x0 ...
```

The Blackhawk 'USB510 bus-powered JTAG emulator' can be operated using:

```
dbgjtag -d bhjtag3.dll -p 0x0 ...
```

The Blackhawk 'USB560 bus-powered JTAG emulator' can be operated using:

```
dbgjtag -d bh560ubp.out -p 0x0 ...
```

2.4 *Select the Board Config' File with [-f]*

This option requires a filename as its sub-argument. It is used to provide the name of the modern or legacy format board config' file. The default extension of '.cfg' is implied if none is specified. When the related [-d] and [-p] options are not used then board config' files are compulsory. When those options are used then board config' files are optional. The documentation of the file formats, with details of scan-path descriptions and config' variables are in an appendix of this user's guide.

The content needed for 100/510-class emulators is the variables sepk.pod_drvr and sepk.pod_port to name the emulator's adapter and address. The content needed for 560-class emulators is the variables uscif.ecom_drvr and uscif.ecom_port to name the emulator's program and address. These examples using [-f] are equivalent to those for the [-d] and [-p] options:

The TI XDS100pp parallel-port emulators can be operated using:

```
# config version=3.5
$ sepk
  pod_drvr = xds100pp.dll
  pod_port = 0x0
$ /
# /
```

The TI XDS510 ISA-bus emulators can be operated using:

```
# config version=3.5
$ sepk
  pod_drvr = xds510.dll
  pod_port = 0x0
$ /
# /
```

The TI XDS560 PCI-bus emulators can be operated using:

```
# config version=3.5
$ uscif
  ecom_drvr = xds560.out
  ecom_port = 0x0
$ /
# /
```

The Blackhawk USB510 bus-powered JTAG emulators can be operated using:

```
# config version=3.5
$ sepk
  pod_drvr = bhjtag3.dll
  pod_port = 0x0
$ /
# /
```

The Blackhawk USB560 bus-powered JTAG emulators can be operated using:

```
# config version=3.5
$ uscif
  ecom_drvr = bh560ubp.out
  ecom_port = 0x0
$ /
# /
```

2.5 *Get Options in Config' Variables with [-e]*

This option requires a config' variable name as its sub-argument.

It is used to provide the name of a variable in a board config' file. The variable is used to provide extra command line options, or all of the command line options. When this option is not used then that variable will be ignored.

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 config' file, and for just one to be selected by this option.

use [-e uscif.test_1] to select uscif.test_1

use [-e uscif.test_2] to select uscif.test_2

use [-e uscif.test_3] to select uscif.test_3

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

2.6 *Redirect Standard Output with [-o]*

This option has an optional sub-argument whose value is a filename.

This option is used to 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 rolling busy flags and rolling test scores on the console output. This option does not affect the standard error output of this utility which is used only to report invalid arguments.

2.7 *Reset the Target System with [-s]*

This option has a sub-argument whose value has four choices.

This option is used to reset the target system by pulsing the nSRST signal available on the Revision-D cable just once. It skips all actions that involve touching emulator hardware. The nSRST signal is accessed without touching the controller in the emulator and without modifying the JTAG state. The choices for this option are:

`nothing`	Do nothing to the system reset signal, its output value does not change.
`assert`	Assert the system reset signal, its active low so outputs a low voltage.
`negate`	Negate the system reset signal, its active low so outputs a high voltage.
`pulse`	Pulse the system reset signal, invert then wait 500us then restore.

2.8 Don’t Touch Hardware with [-w]

This option has no sub-argument. It is used to skip all actions that involve touching emulator hardware. These actions include those that open, call and close emulator control functions. The primary purpose of this option is to improve the testing of the utility itself.

3 The Major Options

The major command-line options are represented by single upper case letters and their syntax is detailed in the section '1.2.3 The Syntax'. They always require a sub-argument. The sub-argument consists of a list of a command and zero or more parameters separated by comma characters. The command is a single word. The parameters are a name and a value separated by an equal characters. The minor command line options provide access to all of the features of the utility from simple emulator selection and resets to complex scan-path frequency and routing tests.

3.1 *Select the Emulator with [-X]*

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
        -X help
```

Select a 560-class emulator by program with optional 'out' extension.

```
        -X program,driver=filename.out,ioport=address
```

Select a 100/510-class emulator by adapter with optional 'dll' extension.

```
        -X adapter,driver=filename.dll,ioport=address
```

Select a 100/510-class or 560-class emulator via a remote machine and service.

```
        -X remote,nethost=name,netport=number
```

Select a 100/510-class or 560-class emulator using config' variables.

```
        -X variable,configure=filename.cfg
```

3.1.1 The Introduction

The basics of emulator selection

The selection of the emulator by the `[-X]` command is from the user's perspective probably the least interesting decision made by the DBGJTAG utility. However it is also a key decision for those other`[-Y, -F, -G, -S, -R, -T, -U]` commands that access an emulator and perform the actions of real interest from the user's perspective. The `[-X]` command supports the selection of 100/510-class and 560-class emulators on the local machine. The `[-X]` command also provides a facility for selecting an emulator on a remote host. Currently that functionality is only supported by a limited number of emulator programs and adapters.
The selection of the emulator by the `[-X]` major command
is the equivalent of selection of the emulator by:

- The `[-d / -p]` minor commands.

- The `uscif.ecom_drvr / uscif.ecom_port` variables.
- The `sepk.pod_drvr / sepk.pod_port` variables.

The selection of a remote host by the `[-X]` major command
is the equivalent of selection of remote host by:

- The `sepk.net_host` / `sepk.net_port` variables.

The `[-X program / adapter]` commands are used to select both 100/510-class and 560-class emulators on the local computer. This direct selection from the command-line over-rides any attempt at selection based on variables in a board config' file.

The `[-X remote]` command extends the `[-X program / adapter]` commands. It is used to select 100/510-class and 560-class emulators on a remote computer. This direct selection from the command-line over-rides any attempt at selection based on variables in a board config' file. Currently this functionality is only supported by a limited number of emulator programs and adapters.

The `[-X variable]` command uses variables in a board config' file. The variables select 100/510-class and 560-class emulators on the local computer or a remote computer. Those variables provide information equivalent to the `[-X program / adapter]` and `[-X remote]` commands of the DBGJTAG utility.

3.1.2 The Commands

The emulator selection [-X PROGRAM] command

This command is used to select 560-class emulators on the local computer. This direct selection from the command-line over-rides any attempt at selection based on variables in a board config' file.

The emulator selection [-X ADAPTER] command

This command is used to select 100/510-class emulators on the local computer. This direct selection from the command-line over-rides any attempt at selection based on variables in a board config' file.

The emulator selection [-X REMOTE] command

This command extends the `[-X program / adapter]` commands. It is used to select 100/510-class and 560-class emulators on a remote computer. This direct selection from the command-line over-rides any attempt at selection based on variables in a board config' file. Currently this functionality is only supported by a limited number of emulator programs and adapters.

The emulator selection [-X VARIABLE] command

This command uses variables in a board config' file. The variables select 100/510-class and 560-class emulators on the local computer or a remote computer. Those variables provide information equivalent to the `[-X program / adapter]` and `[-X` `remote]` commands of the DBGJTAG utility.

3.1.3 The Parameters

The DRIVER parameter

This parameter is the name of the emulator program or adapter that provides access to the emulator. The [-d] option and the `uscif.ecom_drvr` and `sepk.pod_drvr` variables access the same functionality as this parameter. The former variable is used to access 560-class emulators. The latter variable is used to access 100/510-class emulators. All three have similar values.

The emulator adapter file names are libraries for 100/510-class emulators developed using TI's 'XDS510 Sourceless EPK'. They connect the CC Setup and CC Studio suite, and DBGJTAG to 100/510-class emulators. Some well known adapter file names are:

`xds100pp.dll`	The TI XDS100pp parallel-port 100-class emulator.
`xds510.dll`	The TI XDS510 ISA-bus 510-class emulator.
`xdsbhpci.dll`	The BH PCI-bus emulator operated 510-class.
`bhjtag3.dll`	The BH USB bus powered 510-class emulator.

The emulator program file names are libraries for 560-class emulators developed using TI's 'XDS560 Sourceless EPK'. They connect the CC Setup and CC Studio suite, and DBGJTAG to 560-class emulators. Some well known program file names are:

`xds560.out`	The TI PCI-bus XDS560 emulator.
`bh560pci.out`	The BH PCI-bus emulator operated 560-class.
`bh560ubp.out`	The BH USB bus powered 560-class emulator.
`bh560usbm.out`	The BH USB mains power 560-class emulator.

The IOPORT parameter

This parameter is the value of the emulator index or I/O port that distinguishes each emulator of similar type. The [-p] option and the `uscif.ecom_port` and `sepk.pod_port` variables access the same functionality as this parameter. The former variable is used to access 560-class emulators. The latter variable is used to access 100/510-class emulators. All three have similar values.

The simple 100/510-class emulators use hardware specific I/O port addresses with obscure numerical values. The software for TI XDS510 emulators supports aliases (0, 1, 2, and 3) for the I/O port addresses (0x240, 0x280, 0x320 and 0x340). The software for TI XDS100pp emulators supports aliases (0, 1, and 2) for the I/O port addresses (0x378, 0x278 and 3BC).

0x378 0x278 3BC	The TI XDS100pp parallel-port 100-class emulator.
0x240 0x280 0x320 0x340	The TI XDS510 ISA-bus 510-class emulator.
`0 1 2 3`	The BH PCI-bus emulator operated 510-class.
`0 1 2 3`	The BH USB bus powered 510-class emulator.

The sophisticated 560-class emulators use simple indexes
to distinguish multiple emulators of the same type.

`0 1 2 3`	The TI PCI-bus XDS560 emulator.
`0 1 2 3`	The BH PCI-bus emulator operated 560-class.
`0 1 2 3`	The BH USB bus powered 560-class emulator.
. `0 1 2 3`	The BH USB mains power 560-class emulator.

The NETHOST parameter

This parameter is the network name of the remote computer on which a TCP or UDP service supports client connections to an emulator. The `sepk.net_host` variable accesses the same functionality as this parameter and has similar values. Currently this functionality is only supported by a limited number of emulator programs and adapters.

- The network name can be a short locally recognised name that is a single string.
- The network name can also be a fully qualified domain name (FQDN),
  which is multiple strings separated by single dots characters.
- The network name can also be an internet protocol (IP) address, |which is
  four small decimal numbers from 0 to 255 that are separated by dot characters.

The NETPORT parameter

This parameter is the network port on the remote computer at which a TCP or UDP service supports client connections to an emulator. The `sepk.net_port` variable accesses the same functionality as this parameter and has similar values. Currently this functionality is only supported by a limited number of emulator programs and adapters.

- The network port is the TCP or UDP port number,
  which is a relatively small decimal number from 0 to 65535

The CONFIGURE parameter

The [-f] option accesses the same functionality as this parameter.

This parameter is the name of the board config' file that provides config' variables for emulator selection, and possibly also config' variables and a scan-path description for other purposes. The file can be hand-written or generated by CC Setup utility. These files typically have '.cfg' and '.dat' file name extensions. The file generated by CC Setup is named 'BrdDat\ccBrd0.dat'.

3.2 *Configure the Emulator with [-Y]*

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-Y help
```

Reset the target system via its nSRST signal.

```
        -Y system,signal=string
```

Reset an emulator and its JTAG interface via its nTRST signal.

```
        -Y reset,logfile=boolean
```

Select the bootmode and default output values for the EMU1-0 pins.

```
        -Y emupins,jtagboot=string,
                   powerboot=string,
                   emuoutput=string
```

Select the configuration of the JTAG TMS, TDO and TDI pins.

```
        -Y jtagpins,loopback=string,
                    linkdelay=string
                    tdoedge=string,
                    tdiedge=string
```

3.2.1 The Introduction

The basics of emulator configuration

After the selection of the emulator by the [-X] command, the configuration of the emulator by the `[-Y]` command is the next decision made by the DBGJTAG utility. This configuration prepares the emulator and target system for the `[-F, -G, -S, -R, -T, -U]` commands that perform the actions of interest to the user. The `[-Y]` command supports the configuration of 100/510-class and 560-class emulators. However the nature of the configuration is dependent on the cable electronics attached to the emulator or embedded within the emulator. The emulator signals and cables are detailed in the section '5.1 The Standard Emulator Signals and Cables'. The Revision-D cable supports the most flexible configuration, the Revision-B cable supports a more limited configuration, and the XDS510 and XDS100 cables supports the most restricted configuration.

The `[-Y system / reset]` command is used to reset 100/510-class and 560-class emulators and their target systems. This direct selection from the command-line over-rides any selection based on variables in a board config' file.

The `[-Y emupins]` command is used to configure target system boot-modes that are output on the EMU1-0 pins when rising-edges occur on the nTRST and TVD signals. This direct selection from the command-line over-rides any selection based on variables in a board config' file.

The `[-Y jtagpins]` command is used to configure the loop-back and timing of the five JTAG signals: TMS, TDO / TDI and TCLKO / TCLKR. The loop-back can be individually applied to the pairs of clock and data signals, or applied as a total loop-back that also allows the cable to remain operational when disconnected from the target system. The timing can be controlled both as the clock periods of delay, of the TDI input signal verses the TMS / TDO output signals, and the timing of the TMS / TDO output signals against the rising / falling edge of TCLKR.

3.2.2 The Commands

The emulator configuration [-Y SYSTEM] command

This command is used to reset the target system. The nSRST signal available on the Revision-D cable is pulsed just once. It skips all actions that involve touching scan-controller hardware. The nSRST signal is accessed without touching the JTAG controller in the emulator and without modifying the JTAG state.

The emulator configuration [-Y RESET] command

This command is used to reset the emulator. Reset means the emulator will be re-programmed and initialised. The JTAG controller will enter the JTAG 'Test-Logic-Reset' state. The nTRST signal available on the XDS100 and XDS510 cables, and the Revision-B, C and D cables is pulsed just once.

All 560-class emulators and advanced 510-class emulators are constructed using programmable FPGA's. The data for the FPGA is compiled into 560-class emulator programs, 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 and cable.

The emulator configuration [-Y EMUPINS] command

This command is used to configure the boot-mode and default behaviour of the EMU1-0 signals and their response to changes in the nTRST (JTAG reset) output signal and TVD (target voltage detect) input signal.

The configuration of the EMU1-0 signals is supported only by the Revision-D cable. All other cables for 510- and 560-class emulators have fixed input-only EMU1-0 signals. The 100-class emulators have no EMU1-0 signals.

The `jtagboot=string` parameter configures the EMU1-0 signals to remain high-impedance or to output a value (with respectable set-up and hold times) when a rising edge occurs on the nTRST signal. This two-bit value is used by many target devices to select alternative scan-path configurations, or configure power management, with the all-ones value as the default selected by pull-up resistors when the emulator cable is disconnected.

The `powerboot=string` parameter configures the EMU1-0 signals to remain high-impedance or to output a value (with respectable set-up and hold times) when a rising edge occurs on the TVD signal. This two-bit value is used by many target devices to select alternative scan-path configurations, or configure power management, with the all-ones value as the default selected by pull-up resistors when the emulator cable is disconnected.

The `emuoutput=string` parameter configures the EMU1-0 signals to remain high-impedance or to output a value when neither boot-mode is being applied.

The emulator configuration [-Y JTAGPINS] command

This command is used to configure the loop-back feature of the TCLKO / TCLKR and TDO / TDI signals, the pipeline feature of the TMS / TDO and TDI pins, and the timing feature of the TMS / TDO and TDI signals.

The configuration of the loop-back feature is supported only by the Revision-D cable. The configuration of the link-delay and signal timing is supported only by all cables.

The loop-back feature of the TCLKO / TCLKR and TDO / TDI signals is applied at the header. This parameter selects loop-back of the header's TDO / TDI signals and loop-back of the TCLKO / TCLKR signals. When total loop-back of both signals is selected, then the header's TVD power-loss detect and TDIS cable-break detect signals are also disabled. This enables loop-back operation even when the target is powered-off or the header is dis-connected.

The link-delay feature of the TMS / TDO and TDI signals is applied at the scan-controller to compensate for the pipe-line in the cable and target system through which the TMS / TDO and TDI signals are clocked. The maximum link-delay value is dependent on the scan-controller design. No scan-controllers yet support values higher than 31.

The propagation and timing feature of the TMS / TDO and TDI signals is applied at the header. The propagation of the output and the sampling of the input can be selectively done on the rising or falling edges of the clock signal. The default choices are those that allow the highest possible clock frequency to be achieved.

3.2.3 The Parameters

The SIGNAL parameter

The [-s] minor command accesses the same functionality as this parameter.

This parameter is used to reset the target system by pulsing the nSRST signal available on the Revision-D cable just once. It skips all actions that involve touching emulator hardware. The nSRST signal is accessed without touching the controller in the emulator and without modifying the JTAG state. This parameter has string values whose default is marked bold.

``**n**`othing`	Do nothing to the system reset signal, its output value does not change.
`assert`	Assert the system reset signal, its active low so outputs a low voltage.
`negate`	Negate the system reset signal, its active low so outputs a high voltage.
`pulse`	Pulse the system reset signal, invert then wait 500us then restore.

The LOGFILE parameter

The [-r] and [-v] minor commands accesses the same functionality as this parameter.

This parameter is used to reset the emulator's JTAG scan-controller, initialise the emulator's cable control logic, and reset the JTAG interface. The target connector will be placed in the JTAG Test-Logic-Reset state and the active-low JTAG nTRST signal will be asserted for several milliseconds.

If the emulator is a 560-class product or advanced 510-class product then its JTAG TCLK signal will be programmed to the default frequency, which may be specified by

[-F clock,beginning=hertz], and its internal FPGA's will be programmed with the default VHDL built into the emulator program itself. In the case of TI's XDS560 product, the frequency is 500kHz and the program is 'xds560.out'. This parameter has boolean values whose default is marked bold.

`no` (**n**o)	Request a silent reset. This is similar to [-r].
`yes`	Request a verbose reset. This is similar to [-r -v].

The JTAGBOOT parameter

This command-line parameter is equivalent to these config' variables:

```
uscif.jtagboot_mode=string
uscif.jtagboot_value=string
```

The former has the values `disable` and `enable`.
The latter has similar values to this parameter.

This parameter selects the 'JTAG nTRST boot-mode' applied to a target system. It represents two-bit values that are output on the EMU1-0 pins in response to specific conditions. It is supported by 560-class and 510-class emulators with Revision-D cables. This parameter has string and numerical values whose default is marked bold.

d`isable / hiz`	Both EMU1-0 pins remain hi-z.
`00`	Both EMU1-0 pins are low.
`01`	EMU1 is low, EMU0 is high.
`10`	EMU1 is high, EMU0 is low.
`11`	Both EMU1-0 pins are high.

The two-bit value is output on the pins when the header's nTRST signal is asserted. The value is sampled by the target system on the following rising edge of nTRST as it is negated. The value continues to be output on the pins after the rising edge until USCIF35 software disables their output buffers.

The POWERBOOT parameter

This command-line parameter is equivalent to these config' variables:

```
uscif.powerboot_mode=string
uscif.powerboot_value=string
```

The former has the values `disable` and `enable`.
The latter has similar values to this parameter.

This parameter selects the 'Power-on-Reset boot-mode' applied to a target system. It represents two-bit values that are output on the EMU1-0 pins in response to specific conditions. It is supported by 560-class and 510-class emulators with Revision-D cables. This parameter has string and numerical values whose default is marked bold.

d`isable / hiz`	Both EMU1-0 pins remain hi-z.

`00`	Both EMU1-0 pins are low.
`01`	EMU1 is low, EMU0 is high.
`10`	EMU1 is high, EMU0 is low.
`11`	Both EMU1-0 pins are high.

The two-bit value is output on the pins when the header's TVD signal is negated. The value is sampled by the target system on the following rising edge of TVD as it is asserted. The value continues to be output on the pins after the rising edge until USCIF35 software disables their output buffers.

The EMUOUTPUT parameter

This command-line parameter is equivalent to these config' variables:

```
uscif.emuoutput_mode=string
uscif.emuoutput_value=string
```

The former has the values `disable` and `enable`.
The latter has similar values to this parameter.

This parameter selects the 'default or background values' applied to a target system when no specific boot-mode conditions occur. It represents two-bit values that are output on the EMU1-0 pins in response to specific conditions. It is supported by 560-class and 510-class emulators with Revision-D cables. This parameter has string and numerical values whose default is marked bold.

d`isable / hiz`	Both EMU1-0 pins remain hi-z.
`00`	Both EMU1-0 pins are low.
`01`	EMU1 is low, EMU0 is high.
`10`	EMU1 is high, EMU0 is low.
`11`	Both EMU1-0 pins are high.

The two-bit value is default value output on the pins when neither the
'JTAG nTRST boot-mode' nor the 'Power-on-Reset boot-mode'
is being applied to a target system.

The LOOPBACK parameter

This command-line parameter is equivalent to these config' variables:

```
uscif.loopback_mode=string
uscif.loopback_value=string
```

The former has the values `disable` and `enable`.
The latter has similar values to this parameter.

This parameter is used only by 560-class and 510-class emulators with Revision-D cables. They configure loop-back of the header's TDO / TDI signals and TCLKO / TCLKR signals. This parameter has string values whose default is marked bold.

disable	The loopback of clock and/or data is disabled.
data	The loopback of TDO / TDI is enabled.
clock	The loopback of TCLKO / TCLKR is enabled.
total	The loopback of both data and clock is enabled and the detection of power-loss and cable-break is disabled.

When loop-back of both sets of signals is selected then both the header's TVD power-loss detect and TDIS cable-break detect signals are also disabled. This enables emulator testing when the target is powered-off or the header is dis-connected.

The LINKDELAY parameter

This command-line parameter is equivalent to these config' variables:

```
uscif.linkdelay_mode=string

uscif.linkdelay_value=string
```

The former has the values `disable` and `enable`.
The latter has similar values to this parameter.

This parameter is used by all 100/510-class and 560-class emulators. It provides the link-delay values applied to the scan-controller that compensates for the pipe-line in the cable through which the TMS / TDO and TDI signals are clocked. The maximum link-delay value depends on the scan-controller design. No scan-controllers support values higher than 31. This parameter has string and numerical values whose default is marked bold.

disable	The cable's designed link delay value will be applied.
automatic	The link delay value will be calculated automatically.
0 - 31	The explicit link delay value will be applied.

The TDOEDGE and TDIEDGE parameters

These command-line parameters are equivalent to these config' variables:

```
uscif.tdoedge=string

uscif.tdiedge=string
```

The former parameter is used by all 100/510-class and 560-class emulators. The latter parameter is currently ignored by all cables. These parameters select the timing of the TMS / TDO and TDI signals, relative to the rising and falling edges of TCLKR signal on the header of the emulator. The USCIF35 software stack may slightly adjust the link-delay to match the chosen timing. These parameters have string values whose defaults are marked bold.

The `[-Y jtagpins,tdoedge=string]` parameter has these options:

fall	Output TDO on the falling TCLKR edge.
rise	Output TDO on the rising TCLKR edge.

The `[-Y jtagpins,tdiedge=string]` parameter has these options:

`fall`	Input TDI on the falling TCLKR edge.
`r``ise`	Input TDI on the rising TCLKR edge.

These variables use the JTAG naming convention where TMS / TDO are output pins from the header to the target system and TDI is the input pin on the header from the target system. This is different from the TI and ARM emulator header documentation that names the data output pin as TDI and the data input pin as TDO.

The choice of clock edge can have a significant impact on the maximum clock frequency and thus the performance of utilities, loaders and debuggers. The default is rising edge timing because that achieves 39.0MHz operation with many target systems, while the use of falling edge operation achieves only 25.5MHz operation with the same target systems.

3.3 Select the Frequency with [-F]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-F help
```

Select primary options for TCLK frequency selection.

```
-F clock,program=value,
         beginning=hertz,frequency=hertz,
         testmode=value
```

Select secondary options for TCLK frequency selection.

```
-F pll,reference=value,overflow=value,
       initial=boolean,terminal=boolean
```

Select tertiary options for TCLK frequency selection.

```
-F length,bits=number,irsize=value,drsize=value
```

Request extra information about TCLK frequency selection.

```
-F inform,logfile=boolean,whitebox=boolean
```

3.3.1 The Introduction

The basics of frequency selection

The [-F clock / pll / length / inform] commands of the DBGJTAG utility provide access to the emulator hardware and software features that generate the JTAG output clock (TCLKO) and measure the JTAG return clock (TCLKR). These commands allow the user to choose any of the algorithms for frequency selection that are supported by the emulator and its cable, and provide options to modify the algorithms' behaviour by over-riding measurements made during their execution. The commands also allow the user to access the log-file of decisions made and actions performed by the software when executing the algorithm.

These commands can access two versions of the JTAG TCLKO frequency selection algorithm. The default black-box version is embedded within the emulator and able to access the generation and measurement hardware directly. It is used by tools running emulation drivers on DSP and ARM target devices. The white-box version runs within this utility and accesses the hardware indirectly. It is used for developing and validating the algorithms themselves.

These commands require the emulator to have a JTAG TCLKO output driven by a programmable PLL and a JTAG TCLKR input monitored by a frequency measurement counter. These features are available on all 560-class products and advanced 510-class products.

The core parameters are the [-F clock] command's 'program' and 'frequency' parameters, and the [-F inform] command's 'logifile' and 'whitebox' parameters. The former two choose the TCLKO selection algorithm and frequency, the latter retrieves the log-file of actions taken and decisions made when applying the algorithm, plus chooses the version of the algorithm.

The following example is useful for investigating the maximum performance of the electrical connection between the JTAG controller and target devices. It chooses automatic frequency selection with a 64MHz limit and retrieval of the resulting log-file:

```
-F clock,program=automatic,
         frequency=64MHz -F inform,logfile=yes
```

The following example duplicates the typical configuration of emulation drivers on DSP and ARM target devices. It chooses automatic frequency selection with a 35MHz limit and no retrieval of the log-file.

```
-F clock,program=automatic,
         frequency=35MHz -F inform,logfile=no
```

The config' variables for frequency selection

The parameters for the [-F clock / pll / length] commands are implemented with memory-mapped config' variables instead of with traditional function arguments. These commands can also be entered using the generic [-V apply] command that allows any config' variable to be entered from the command line. The core of these commands can execute both within the emulator (black-box) and on the host computer (white-box). The values of these memory-mapped variables will over-ride any similar variables input from the [-f] command's board config' file.

The parameters for the [-F inform] command are implemented with traditional function arguments. That command is only required to execute on the host computer's USCIF35 software for both 560-class and 510-class products.

The application of the variables associated with the [-F clock / pll / length] commands is restricted to the TCLK generation and measurement functions that are used by the [-F] and [-f] options, and by the start-up of emulation drivers for DSP and ARM target devices. The full details of the config' variables are in the first three sub-sections of '5.6 The Standard Board Config' Variables'.

For the [-F clock] command 'The primary frequency variables' and default values are:

```
uscif.tclk_program               automatic
uscif.tclk_beginning             500KHz
uscif.tclk_frequency             64.0MHz
uscif.tclk_testmode              surrender
```

For the [-F pll] command 'The secondary frequency variables' and default values are:

```
uscif.tclk_overflow          failure
uscif.tclk_reference         default
uscif.tclk_initial           true
uscif.tclk_terminal          false
```

For the [-F length] command 'The tertiary frequency variables' and default values are:

```
uscif.tclk_scan_bits         16768
uscif.tclk_path_irsize       measure
uscif.tclk_path_drsize       measure
```

The details of frequency selection

The full details of frequency selection are described in '5.3 The Selection of TCLKO Frequency'.

3.3.2 The Commands

The frequency selection [-F CLOCK] command

This command accesses the primary configuration features of the algorithms used to select the JTAG TCLKO frequency. Its parameters select the choice of algorithm, starting frequency and target frequency, and provide an override for decisions based on the results of scan-tests used by some algorithms. Each of the parameters have matching config' variable that accesses the same feature from the board config' file. The command-line parameters override the board config' variables.

The program=value parameter (and its equivalent variable uscif.tclk_program) chooses the algorithm used to select the JTAG TCLKO frequency generated via a programmable PLL.

The beginning=hertz and the frequency=hertz parameters (and their equivalent variables uscif.tclk_beginning and uscif.tclk_frequency) provide the starting frequency and target frequency for the algorithms used to select the JTAG TCLKO frequency generated via a programmable PLL.

The testmode=value parameter (and its equivalent variable uscif.tclk_testmode) modifies the scan-path test that is used to choose the JTAG clock frequency used by XDS560 products.

The frequency selection [-F PLL] command

This command accesses the secondary configuration features of the algorithms used to select the JTAG TCLKO frequency. Its parameters select the reference clock frequency for the PLL that generates TCLKO, plus choose how overflows of the storage for the embedded log-file are to be handled, plus choose if the TCLKO is to be initialised at start-up and re-initialised at shut-down. Each of the parameters have matching config' variable that accesses the same feature from the board config' file. The command-line parameters override the board config' variables.

The reference=value parameter (and its equivalent variable uscif.tclk_reference) modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the reference oscillator frequency for the PLL.

The overflow=value parameter (and its equivalent variable uscif.tclk_overflow) modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the handling of log-file over-flows when recording PLL actions, as a gross error or a harmless event.

The initial=boolean and the terminal=boolean parameters (and their equivalent variables uscif.tclk_initial and uscif.tclk_terminal) modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. The former parameter optionally initialises the PLL just after the USCIF35 software stack is first opened. The latter parameter optionally re-initialises the PLL just before the USCIF35 software stack is last closed.

The frequency selection [-F LENGTH] command

This command accesses the tertiary configuration features of the algorithms used to select the JTAG TCLKO frequency. Its parameters override the default length of scan-tests used by some algorithms, plus provide overrides values for the IR and DR register path-length measurements required by some algorithms. Each of the parameters have matching config' variable that accesses the same feature from the board config' file. The command-line parameters override the board config' variables.

The bits=number parameter (and its equivalent variable uscif.tclk_scan_bits) selects the length of the scan-path tests used to validate the frequency of the JTAG clock for XDS560 products. Those tests are chosen by the algorithms selected by the program=value parameter.

The irsize=value and the drsize=value parameters (and their equivalent variables uscif.tclk_path_irsize and uscif.tclk_path_drsize) modify the scan-path size measurements used to calibrate scan-path tests. They instruct the emulator to either measure the sizes of the JTAG IR instruction scan-path and JTAG DR bypass scan-paths, or to use explicit values. The scan-path tests are themselves used to validate the frequency of the JTAG clock for XDS560 products.

The frequency selection [-F INFORM] command

This command accesses two diagnostic features for the algorithms used to select the JTAG TCLKO frequency. One feature provides detailed information about the measurements made and actions taken by the algorithms. The other feature allows the algorithms to be exercised as a host implementation or embedded implementation.

The whitebox=boolean parameter selects use of one of the two instances of the JTAG TCLKO frequency selection algorithm. One instance is the default (black-box) version that runs embedded within the emulator and accesses the generation and measurement hardware directly. It is used by tools running emulation drivers on DSP and ARM target devices. The other instance is the (white-box) version that runs within this utility and accesses the hardware indirectly - it is used for developing and validating the algorithms themselves.

The inform=boolean parameter selects the retrieval and printing of the internal log-file of actions taken and decisions made when applying the JTAG TCLKO frequency selection algorithms. The log-file is formatted as an eight column table. The columns in order are:

`Test`	The index of the action or scan-path test.
`Size`	The length in words of any scan-path test.
`Y`	The PLL programming major coordinate: 1MHz * (2**Y).
`Z`	The PLL programming minor coordinate: (1 + Z/64).

`Mhz`	The intended PLL frequency expressed as a number and suffix.
`Flag`	The result expressed as a character plus parenthesis or braces.
`Result`	The result expressed as a short text string.
`Description`	The purpose of the test expressed as a long text string.

The values in the ‘Size’ column are the length in 32-bit words
of any scan-path test performed at the programmed frequency:

`none`	No scan-path test.
`128`	4k bit long scan-path test.
`512`	16k bit long scan-path test.
`2048`	64k bit long scan-path test.

The values in the ‘Y’ and ‘Z’ columns are the commands programmed into the PLL so as to output the frequency value in the ‘MHz’ column. The values are related by the expression ‘1MHz * (2**Y) * (1 + Z/64)’. Note that though PLL programming for the output TCLKO has a resolution of just 1/64, the frequency measurements for the input TCLKR are accurate to better than 1/256.

The values in the ‘MHz’ column indicate the frequency output by the PLL.
The characters in the ‘Flag’ column and the values in
the ‘Result’ column indicate the evaluation of the scan-test.

`0`	`successful`	This marks a test that succeeded.
`-`	`not tested`	This marks a test that was not done.
`?`	`bad value`	This marks a test that failed with a bad data value due to a mismatch between the send and receive data.
`#`	`bad command`	This marks a test that failed with a bad command due to the controller state being corrupted.
`%`	`bad frqncy`	This marks a test that failed with a bad frequency due to a mismatch between the programmed and measured values.

The values in the ‘Flag’ column are also marked by the character pairs ‘[ ]’ and ‘{ }’ when auto-ranged frequency or specific frequency configurations are used. If the maximum test frequency and the frequency finally chosen for operation are equal then they are marked with ‘[ ]’, else they are marked with ‘{ }’.

The values in the ‘Description’ column indicate either the purpose of any scan-path test performed at the programmed frequency, or the result of any frequency measurement.

The following values relate to initial scan-tests:

`isit external clock`	The frequency tests detect an external clock.
`isit internal clock`	The frequency tests suggest an internal clock.
`test external clock`	This test now assumes an external clock.

`test internal clock`	This test now assumes an internal clock.
`measure path length`	The IR and DR lengths are measured.
`apply beginning clock`	The beginning frequency is programmed.
`apply explicit clock`	A single specific frequency is used.

The following values relate to auto-ranging scan-tests:

`auto step initial`	The initial step-up frequency is applied.
`auto step delta`	A higher step-up frequency is applied.
`auto power initial`	The initial optimised frequency is applied.
`auto power delta`	A modified optimised frequency is applied.
`auto margin initial`	The initial margin frequency is applied.
`auto margin delta`	A lower margin frequency is applied.

3.3.3 The Parameters

The PROGRAM parameter

This command-line parameter is equivalent to this config' variable:

```
uscif.tclk_program=value
```

This parameter selects the method used by XDS560 products to choose the JTAG clock frequency. All of the tests used by those algorithms are restricted to the JTAG IR instruction scan-path and JTAG DR bypass scan-path. This parameter has string values whose default is marked bold.

`nothing`	Don't touch the PLL at all.
`external`	Assume an external clock source. This is used by the the USCIF35 software stack as an over-ride value if it detects TCLKR has an external source.
`reference`	Use the PLL reference frequency.
`specific`	Use a specific user-selected frequency. The parameter `[-F clock,frequency=hertz]` is the actual frequency. Its default is `legacy`.
`a``utomatic`	Use an automatic frequency selection. The parameter `[-F clock,beginning=hertz]` is the lower frequency limit. Its default is `superlow` or `minimum`, the cable's lowest supported fixed frequency. The parameter `[-F clock,frequency=hertz]` is the upper frequency limit. Its default is `maximum`.
`adaptive`	Use an adaptive frequency selection. The parameter `[-F clock,frequency=hertz]` is the upper frequency limit. Its default is `adaptive`,

	the cable's highest supported adaptive frequency. Supported only by the XDS560 Revision-D cables.
`inverter`	Use an inverter frequency selection. No restriction on the upper frequency limit. Supported only by the XDS560 Revision-D cables.

The BEGINNING / FREQUENCY parameters

These command-line parameters are equivalent to these config' variables:

```
uscif.tclk_beginning=hertz
```

```
uscif.tclk_frequency=hertz
```

These parameters select the frequency limits used by the parameter `[-F clock,program=value]` when choosing the JTAG clock frequency used by XDS560 products. Their default values depend on the value of that other parameter.

These parameters have string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix '5.2 The Frequency of TCLKO and TCLKR'.

The TESTMODE parameter

This command-line parameter is equivalent to this config' variable:

```
uscif.tclk_testmode=value
```

This parameter modifies the scan-path test that is used to choose the JTAG clock frequency used by XDS560 products. Those frequencies are selected by the parameters `[-F clock,program=value,beginning=hertz,frequency=hertz]`. It instructs the emulator to skip the scan-test entirely or to use the scan-test and decide to continue or surrender if it fails. This parameter has string values whose default is marked bold.

`nothing`	Do nothing - skip the scan-test.
`continue`	Do scan-test, log result, continue after fail.
`s``urrender`	Do scan-test, log result, surrender after fail.

The REFERENCE parameter

This command-line parameter is equivalent to this config' variable:

```
uscif.tclk_reference=value
```

This parameter modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the reference oscillator frequency for the PLL. This parameter has string and numerical values whose default is marked bold.

`d``efault`	Assume the default reference oscillator for the cable.
`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.

The OVERFLOW parameter

This command-line parameter is equivalent to this config' variable:

```
uscif.tclk_overflow=value
```

This parameter modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the handling of log-file over-flows when recording PLL actions, as a gross error or a harmless event. This parameter has string values whose default is marked bold.

`failure`	The PLL log-file generates an error when full.
`s``ilent`	The PLL log-file is silent when full.

The INITIAL / TERMINAL parameters

These command-line parameters are equivalent to these config' variables:

```
uscif.tclk_initial=boolean
uscif.tclk_terminal=boolean
```

These parameters modify the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. The `[-F pll,initial=boolean]` parameter optionally initialises the PLL just after the USCIF35 software stack is first opened. The `[-F pll,terminal=boolean]` parameter optionally re-initialises the PLL just before the USCIF35 software stack is last closed. They are supported by XDS560 emulators with Revision-B, C and D cables. These parameters have string values whose defaults are marked bold.

The `[-F pll,initial=boolean]` parameter has two options.

`false`	Do nothing.
`t``rue`	Apply the [-F clock,`beginning=hertz]` frequency.

The `[-F pll,terminal=boolean]` parameter has two options.

`f``alse`	Do nothing.
`true`	Apply the `[-F clock,beginning=hertz]` frequency.

The BITS parameter

This command-line parameter is equivalent to this config' variable:

```
uscif.tclk_scan_bits=number
```

This parameter selects the length of the scan-path tests used to validate the frequency of the JTAG clock for XDS560 products. Those tests are chosen by the algorithms selected by the `[-F clock,program=value]` parameter. The length of the tests can be varied between 1k bits and 512k bits. The tests can have a noticeable effect on the speed of those algorithms at the low TCLK frequencies used by hardware and software simulations, in which case the tests may be reduced in length.

This parameter has string and numerical values whose default is marked bold.

`shortest`	Use IR/DR can-tests that are 1k bits long.
m`edium`	Use IR/DR scan-tests that are 16k bits long.
`longest`	Use IR/DR scan-tests that are 512k bits long.
`1024 – 524288`	Use IR/DR scan-tests that are a specific length.

The IRSIZE / DRSIZE parameters

These command-line parameters are equivalent to these config' variables:

```
uscif.tclk_path_irsize=value

uscif.tclk_path_drsize=value
```

These parameters modify the scan-path size measurements used to calibrate scan-path tests. They instruct the emulator to either measure the sizes of the JTAG IR instruction scan-path and JTAG DR bypass scan-paths, or to use explicit values. The scan-path tests are themselves used to validate the frequency of the JTAG clock for XDS560 products. Those tests are chosen by the algorithms selected by the `[-F clock,program=value]` parameter. The size measurements can have a noticeable effect on the speed of those algorithms at the low TCLK frequencies used by hardware and software simulations, in which case the sizes may be given explicitly. This parameter has string and numerical values whose default is marked bold.

m`easure`	Measure the size of the IR/DR scan-paths.
`0 – 524288`	Specify the size of the IR/DR scan-paths.

The WHITEBOX parameter

The `[-F inform,whitebox=boolean]` parameter selects use of one of the two instances of the JTAG TCLKO frequency selection algorithm. One instance is the default (black-box) version that runs embedded within the emulator and accesses the generation and measurement hardware directly. It is used by tools running emulation drivers on DSP and ARM target devices. The other instance is the (white-box) version that runs within this utility and accesses the hardware indirectly - it is used for developing and validating the algorithms themselves. This parameter has boolean values whose default is marked bold.

f`alse`	Use the embedded (black-box) algorithm.
`true`	Use the host-based (white-box) algorithm.

The LOGFILE parameter

The `[-F inform,logfile=boolean]` parameter selects the retrieval and printing of the internal log-file of actions taken and decisions made when applying the JTAG TCLKO frequency selection algorithm. This parameter has boolean values whose default is marked bold.

f`alse`	Do not retrieve and print the log-file for the

algorithm.

```
        true
```
Retrieve and print the log-file for the algorithm.

3.4 Scan-Path DR/IR Tests with [-S]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
        -S help
```

Measure the length of the DR/IR scan-paths.

```
        -S pathlength,bits=number,
                      irsize=number,drsize=number,
                      ignore=boolean,analyse=boolean
```

Test the DR/IR scan-paths for breaks.

```
        -S brokenpath,bits=number,repeat=number
```

Test the DR/IR scan-paths with fixed data values.

```
        -S integrity,bits=number,repeat=number
```

Test the DR/IR scan-paths with user-specified literal data values.

```
        -S givendata,literal=value,bits=number,repeat=number,
                     testcase=number,testwidth=string
```

Test the DR/IR scan-paths with user-specified expanded data values.

```
        -S givendata,expand=value,bits=number,repeat=number,
                     testcase=number,testwidth=string
```

3.4.1 The Introduction

The basics of the scan-path tests

The [-S] command provides a set of tests for measuring the length and evaluating the reliability of the JTAG IR instruction registers and JTAG DR bypass registers. The path-length test has an optional 'analyse=boolean' parameter for evaluating its results. The three scan-path reliability tests now all support optional 'bits=number' parameters that allow their length of the test itself to be selected. All three also support optional 'repeat=number' parameters that allow them to be repeated a fixed number of times or for ever.

The [-S pathlength] command performs scan-tests that measure the lengths of the JTAG IR instruction registers and JTAG DR bypass registers.

The [-S brokenpath] command uses fixed data values. It performs scan-tests that check for the symptoms of breaks or stuck-faults in the JTAG IR instruction registers and JTAG DR bypass registers.

The [-S integrity] command uses fixed data values. It performs scan-tests that check for the symptoms of alignment errors and speed limitations in the JTAG IR instruction registers and JTAG DR bypass registers.

The [-S givendata] command accepts user specified data values. It performs scan-tests that are intended as general purpose scan-tests for the JTAG IR instruction registers and JTAG DR bypass registers.

3.4.2 The flow-chart of the scan-path tests

The scan-path tests are implemented in two phases.

Phase #1 Measure the path-length.

Phase #2 Perform the scan-tests.

The phase #1 of the scan-path tests

```
  START PHASE #1
MEASURE THE PATHLENGTH
        |
        |
        V
+-----------------+
|     select      |
|    the clock    |
|    frequency    |
+-----------------+
        |
        |
        V
+-----------------+
|    measure      |
|      the        |
|   path-length   |
+-----------------+
        |
        |
        V
  /-------------\
 /    verbose    \ NOT
 \       ?       /------>|
  \-------------/        |
       YES               |
        |                |
        |                |
        V                |
+-----------------+      |
| analyse/report  |      |
|      the        |      |
|   path-length   |      |
+-----------------+      |
        |                V
        |<----------------
        |
        V
      FINISH
```

The phase #2 of the scan-path tests

```
      START PHASE #2
    PERFORM THE SCAN-TEST
            |
            |
            V
    +------------------+
    |     select       |
    |    the clock     |
    |    frequency     |
    +------------------+
            |
            |
            V
    +------------------+
    |    measure       |
    |      the         |
    |   path-length    |
    +------------------+
            |
            |<-----------------
            |                 ^
            V                 |
    +------------------+      |
    | brokenpath or    |      |
    | integrity or     |      |
    | given-data test  |      |
    +------------------+      |
            |                 |
            |                 |
            V                 |
      /-------------\         |
     /    repeat     \ YES    |
     \       ?       /------->|
      \-------------/
           NOT
            |
            |
            V
          FINISH
```

3.4.3 The Commands

The scan-path test [-S PATHLENGTH] command

This command performs scan-tests that measure the lengths of the JTAG IR instruction registers and JTAG DR bypass registers. If the 'analyse' parameter is used then it will also analyse and report on the values that have been measured.

The following summary of path-length values
for common TI devices may be useful.

The JTAG DR bypass register length:

1 bit	The industry standard for all JTAG devices.

The JTAG IR instruction register lengths:

2 bits	The Icepick-A/B TI router devices.
4 bits	The ARM7 or ARM9 TI micro-controllers.
5 bits	The ARM11 TI micro-controllers.
6 bits	The Icepick-C TI router devices.
8 bits	The older ‘Classic’ TI DSP devices: C4x, C5x, C8x, C20xx, C24xx
8 bits	The newer TI ‘Classic’ TI DSP devices: VC33, C54xx, C620x, C670x
38 bits	The ‘IceMaker’ TI DSP devices: C27xx, C28xx, C55xx, C55plus, C64xx, C64plus
46 bits	The mixed ‘Classic/IceMaker’ TI DSP devices: C621x, C622x, C671x, C672x

The scan-path test [-S BROKENPATH] command

This command uses fixed data values. It performs scan-tests that check for the symptoms of breaks or stuck-faults in the JTAG IR instruction registers and JTAG DR bypass registers.

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

These tests use the two data values 0x00000000 and 0xFFFFFFFF.

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 parameter ‘repeat=number’ is used to repeat the tests 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 parameter ‘bits=number’ is used to modify the length of the tests. The default length is 16k bits.

The scan-path test [-S INTEGRITY] command

This command uses fixed data values. It performs scan-tests that check for the symptoms of alignment errors and speed limitations in the JTAG IR instruction registers and JTAG DR bypass registers.

The length of the JTAG IR instruction registers and JTAG DR bypass registers are first measured, then four blocks of fixed 32-bit data values are scanned through both scan-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.

Two of the blocks use the values 0x5533CCAA and its inverse 0xAACC3355. 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 parameter ‘repeat=number’ is used to repeat the tests 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 parameter ‘bits=number’ is used to modify the length of the tests. The default length is 16k bits.

The scan-path test [-S GIVENDATA] command

This command accepts accepts user specified data values. It performs scan-tests that are intended as a general purpose scan-test for the JTAG IR instruction registers and JTAG DR bypass registers.

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

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 ‘repeat=number’ parameter to inject a user selected value into a possibly broken scan-path while the location of the failure is traced with a logic analyser or oscilloscope.

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 optional parameters ‘literal=value’ and ‘expand=value’ are the given data value which must be in 32-bit hexadecimal format. Their default value of 0x5533CCAA was chosen for the trivial reasons of leading and trailing zeros, single and double bits patterns, and 16-bit symmetry.

The parameter ‘repeat=number’ is used to repeat the tests 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.

If the tests are repeated then 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 until any key is pressed.

If the tests are repeated continuously then the given data test will also exercise USCIF35’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 this command 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 this command, and then also checking that the utility can reliably restart after the problem is corrected.

The parameter ‘bits=number’ is used to modify the length of the tests. The default length is 16k bits.

3.4.4 The Parameters

The BITS parameter

The ‘bits’ parameter represents the number of bits used in each scan-test. A shorter bit count will result in a faster but less thorough scan-test. A longer bit count will result in a slower but more thorough scan-test. For some software releases the bit count value may also be rounded off to the nearest multiple of 32 or 1024 bits. The bit count must be longer than the scan-path length.

This parameter has string and numerical values whose default is marked bold.

`0 / none`	Do not use a scan-test.
`shortest`	Use IR/DR can-tests that are 1k bits long.
`m``edium`	Use IR/DR scan-tests that are 16k bits long.
`longest`	Use IR/DR scan-tests that are 512k bits long.
`1024 - 524288`	Use IR/DR scan-tests that are a specific length.

The IRSIZE parameter

This parameter has string and numerical values whose default is marked bold.

`m``easure`	Measure the size of the DR scan-paths.
`0 - 524288`	Specify the size of the DR scan-paths.

The DRSIZE parameter

This parameter has string and numerical values whose default is marked bold.

`m``easure`	Measure the size of the DR scan-paths.
`0 - 524288`	Specify the size of the DR scan-paths.

The IGNORE parameter

This parameter is used to choose the response to path-length measurement errors.

This parameter has boolean values whose default is marked bold.

`n``o`	Measurement errors are considered fatal
`yes`	Measurement errors are ignored, and the following scan-test commands are performed

The ANALYSE parameter

This parameter is interpreted as boolean value that is used to apply or skip the analysis of results obtained from path-length measurement of the JTAG IR instruction and JTAG DR bypass scan-paths. Both the JTAG standard itself and the implementation of the JTAG interface on DSP and ARM devices allow a number of inferences to be made based on the numerical values of the scan-paths. This parameter has boolean values whose default is marked bold.

`n`o (n bold)	Skip the analysis of path-length measurements.
`yes`	Apply the analysis of path-length measurements.

The REPEAT parameter

This parameter represents the number of times the test is applied. A running count of the status of the major option is displayed. The repetition of the test will be halted when any key is pressed.

If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key:

This parameter has numerical and string values whose default is marked bold.

`0`	The test is performed continuously.
`1`	The test is performed just once.
`2 - n`	The test is performed this many times.

Some choices have synonyms:

`default`	The test is performed just once.
`single`	The test is performed just once.
`forever`	The test is performed continuously.

The repeat value of ‘0’ and synonym ‘forever’ are interpreted as a request to continuously repeat the test until any key is pressed, even if errors are found in received data values. This provides a tool for injecting a data value into the scan path so that locations on the scan-path can be monitored for matching/mismatching data values.

The LITERAL parameter

This parameter is interpreted as a (8 hex digit) 32-bit data value that is used for the scan-test. If the full 8 hex digits are not provided then the missing digits are interpreted as zeros. This parameter has numerical values whose default is marked bold.

0`x5533CCAA`	
`0x1`	`0x00000001`
`0x12`	`0x00000012`
`0x123`	`0x00000123`
`0x1234`	`0x00001234`

The EXPAND parameter

This parameter is interpreted as a (8 hex digit) 32-bit data value that is used for the scan-test. If the full 8 hex digits are not provided then the missing digits are interpreted as duplicates. This parameter has numerical values whose default is marked bold.

0x5533CCAA	
0x1	0x11111111
0x12	0x12121212
0x123	0x23123123
0x1234	0x12341234

Some choices have synonyms:

default	0x5533CCAA
zeros	0x00000000
ones	0xFFFFFFFF
align	0xFE03E0E2
speed	0x5533CCAA
~align	0x01FC1F1D
~speed	0xAACC3355

The TESTCASE parameter

This parameter selects the test-case that is used for the scan-test. The test-cases all use the same 32-bit scan-data values, but apply them using different USCIF35 API's and TBC commands. This provides a tool for testing the USCIF35 API's and TBC commands relating to scan-data. Currently there are ten test-cases.

If this parameter has the value 10 then everyone of the test-cases will be performed in sequential order as the scan-test is repeated. If this parameter is 0 - 9 then a single test-case is applied as the scan-test is repeated:

This parameter has string and numerical values whose default is marked bold.

10	Every test-case is used sequentially
0 - 9	A single test-case is used repeatedly

Some choices have synonyms:

default	Every test-case is used sequentially
everyone	Every test-case is used sequentially

The TESTWIDTH parameter

This parameter selects the integer type that is used for the scan-test. The test-cases all use the same 32-bit scan-data values, but pass them to and from USCIF35 API's and TBC commands using arrays of specific integer types. This provides a tool for testing their implementation on different operating systems that handle integer types as little-endian and big-endian values.

This parameter has string values whose default is marked bold.

long	The TBC commands use arrays of 32-bit values.
`short`	The TBC commands use arrays of 16-bit values.
`byte`	The TBC commands use arrays of 8-bit values.

3.5 Scan-Path Frequency Tests with [-G]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-G help
```

Sweep around a center frequency performing scan-path tests.

```
-G single,middle=frequency,
           after=frequency,measure=boolean,
           bits=number,repeat=number
```

Sweep across a range of frequencies performing scan-path tests.

```
-G range,lowest=frequency,highest=frequency,
          after=frequency,measure=boolean,
          bits=number,repeat=number
```

3.5.1 The Introduction

The basics of the scan & frequency tests

The [-G single / range] commands of the DBGJTAG utility require the system to have a JTAG TCLKO output driven by a programmable PLL and a JTAG TCLKR input monitored by a frequency measurement counter. This is true for all 560-class products and advanced 510-class products. The details of the format of frequency values and the supported range of frequency values is detailed in '5.2 The Frequency of TCLKO and TCLKR'.

The choice of the [-G single] or [-G range] command selects between investigating the effects of narrow or wide ranges of JTAG TCLKO frequencies. The tests can perform JTAG IR/DR scan-tests with fixed data patterns, to validate the JTAG scan-path connection between the emulator's TDO and TDI pins. The tests can also measure the JTAG TCLKR input frequency, to validate the software and programmable PLL hardware that generates the JTAG TCLKO.

The 560-class emulators are known to pass these tests repeatedly without error, from a low 0.5Mhz up-to near 48.0MHz for the Revision-B and C cables with the 14-pin header, from an ultra-low 488Hz up-to 48.0MHz for the Revision-D cables with the 20-pin header. Any new errors found are more likely due to target specific signal quality problems.

The [-G single / range] commands have previously 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.
- Incorrect choice of reference oscillator for the programmable PLL.
- Obscure race conditions and timing errors in the PLL programming logic.
- Frequency dependent clock domain errors in the Nano-TBC VHDL.
- Inaccurate results from the frequency measurement logic.

The 'measure=boolean' parameter of the [-G single / range] commands is used to enable validation, at every chosen JTAG TCLK frequency, of the software and programmable PLL hardware by measuring the JTAG TCLKR input. The values 'false' and 'true' disable and enable measurement.

The 'bits=number' parameter of the [-G single / range] commands is used to enable validation, at every chosen JTAG TCLK frequency, of the JTAG scan-path connection between the emulator's TDO and TDI pins. The value 'none' disables validation, the values 'shortest' / 'medium' enable validation by selecting a short quick scan-test and the medium slower scan-test.

The 'repeat=number' parameter of the [-G single / range] commands is used to repeat the entire scan & frequency test one or more times. If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key.

The details of the scan & frequency tests

The [-G single / range] commands perform tests whose purpose is to investigate the effect of narrow or wide range of JTAG TCLKO frequencies. The narrow range of frequencies is specified by the 'middle=frequency' parameter of the [-G single] command. The wide range of frequencies is specified by the 'lowest=frequency' and 'highest=frequency' parameters of the [-G range] command.

The results of the sweep through a range of frequencies is displayed an a table format that marks test results in a style similar to that used by the PLL programming log-file.

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

The values are also marked by the character pairs '[ ]' and '{ }' when auto-ranged frequency or specific frequency configurations are used with the [-G single / range] options. If the maximum test frequency and the final frequency chosen for normal operation are equal then they are marked with '[ ]', else they are marked with '{ }'.

These character pairs show the impact of the ‘uscif.tclk_program’, ‘uscif.tclk_beginning’ and ‘uscif.tclk_frequency’ variables in the board config’ file, or the impact of the [-F clock] options, on the maximum frequency tested for scan-path operation and the frequency finally selected.

Before the test is used it may be necessary to apply a safe frequency value. These tests are preceded by the DBGJTAG’s default auto-ranging scan-tests. If the target system has electrical problems then scan-test failures may prevent the narrow and wide frequency tests being performed. Those failures can often be avoided by configuring DBGJTAG to be less intrusive:

```
-F clock,program=specific,
         beginning=0.5,
         frequency=0.5,
         testmode=nothing
```

After the test is used it may be necessary to apply a safe frequency value. These tests always sweep from a lower to a higher frequency. If the higher frequency exceeds the designed limits of the target system then that system may enter a corrupted state. This situation may be alleviated following the sweep by applying a much lower frequency with the ‘after=frequency’ parameter.

3.5.2 The Commands

The scan & frequency tests [-G SINGLE] command

The [-G single] command selects a narrow frequency range to perform scan and frequency tests with the single ‘middle’ parameter specifying the center point. The [-G single / range] commands are otherwise similar and described in the prior sections ‘The basics of the scan & frequency tests’ and ‘The details of the scan & frequency tests’.

The scan & frequency tests [-G RANGE] command

The [-G range] command selects a wide frequency range to perform scan and frequency tests with the pair of ‘lowest’ and ‘highest’ parameters specifying the lower and upper limits. The [-G single / range] commands are otherwise similar and described in the prior sections ‘The basics of the scan & frequency tests’ and ‘The details of the scan & frequency tests’.

3.5.3 The Parameters

The MIDDLE parameter

The ‘middle’ parameter chooses the frequency around which the effect of a narrow range of JTAG TCLKO frequencies will be tested. The lowest and highest frequencies tested will be 12.5% below and above the middle value. The accuracy of the JTAG TCLKO generation and JTAG TCLKR measurement is sufficient that frequency octaves are be swept using 64 steps. Thus the narrow range involves 16 frequency values. This parameter has string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix ‘5.2 The Frequency of TCLKO and TCLKR’.

The LOWEST / HIGHEST parameters

The ‘lowest’ parameter chooses the lowest frequency above which the effects of a wide range of JTAG TCLKO frequencies will be tested. It is used by the [-G Range] command together with the ‘highest’ parameter. The ‘highest’ parameter chooses the highest frequency below which the effects of a wide range of JTAG TCLKO frequencies will be tested. It is used by the [-G Range] command together with the ‘lowest’ parameter. These parameters have string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix ‘5.2 The Frequency of TCLKO and TCLKR’.

The AFTER parameter

The ‘after’ parameter chooses a final frequency that will be applied after the effects of a narrow or wide range of JTAG TCLKO frequencies are tested. These tests always sweep from a lower to a higher frequency. If the higher frequency exceeds the designed limits of the target system then that system may enter a corrupted state. This situation can often be corrected by applying a much lower frequency following the sweep. This parameter has string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix ‘5.2 The Frequency of TCLKO and TCLKR’.

The MEASURE Parameter

The ‘measure’ parameter chooses if the JTAG TCLKR input frequency is to be measured whenever the JTAG TCLKO frequency is generated. This parameter has boolean values whose default is marked bold.

`**t**rue`	Do measure the input frequency.
`false`	Do not measure the input frequency.

The BITS Parameter

The ‘bits’ parameter represents the number of bits used in each scan-test. A shorter bit count will result in a faster but less thorough scan-test. A longer bit count will result in a slower but more thorough scan-test. For some software releases the bit count value may also be rounded off to the nearest multiple of 32 or 1024 bits. The bit count must be longer than the scan-path length.

This parameter has string and numerical values whose default is marked bold.

`0 / none`	Do not use a scan-test.
`shortest`	Use IR/DR can-tests that are 1k bits long.
`**m**edium`	Use IR/DR scan-tests that are 16k bits long.
`longest`	Use IR/DR scan-tests that are 512k bits long.
`1024 - 524288`	Use IR/DR scan-tests that are a specific length.

The REPEAT Parameter

This parameter represents the number of times the test is applied. A running count of the status of the major option is displayed. The repetition of the test will be halted when any key is pressed.

If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key.

This parameter has numerical and string values whose default is marked bold.

`0`	The test is performed continuously.
`1`	The test is performed just once.
`2 - n`	The test is performed this many times.

Some choices have synonyms:

`default`	The test is performed just once.
`single`	The test is performed just once.
`forever`	The test is performed continuously.

3.6 Scan-Path Routing Tests with [-R]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of ‘name=value’ pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-R help
```

Apply a standard test suite to the router management software.

```
-R softlist,input=filename,output=filename,
            family=filename,alias=filename,
            verbose=boolean
```

Select a route of DR/IR scan-paths from a board config’ file.

```
-R routelist,devices=list,subpaths=list,
             method=string,hierarchy=string,
             noscan=boolean,verbose=boolean
```

Select a matrix of DR/IR scan-paths from a board config’ file.

```
-R matrixlist,devices=list,subpaths=list,
              method=string,hierarchy=string,
              reorder=string,repeat=number,
              noscan=boolean,verbose=boolean
```

3.6.1 The Introduction

The basics of the route and matrix commands

The [-S pathlength / brokenpath / integrity / givendata] commands of the DBGJTAG utility are used alone, or with a board config' file, to test the length and reliability of the JTAG scan-path connection between the emulator's TDO and TDI pins.

The [-R routelist / matrixlist] commands of the DBGJTAG utility extend the [-S pathlength / brokenpath / integrity / givendata] commands. They are used to test the length and reliability of any JTAG scan-path connection that can be created between the emulator's TDO and TDI pins by directly or indirectly selecting one or more sub-paths to be included in the connection between the emulator's TDO and TDI pins.

The 'subpath=list' and 'device=list' parameters of the [-R routelist / matrixlist] options are used to describe JTAG scan-path connections by specifying the names of sub-paths, or the names of devices and routers that are located on those sub-paths.

The [-f] option of the DBGJTAG utility allows a board config' file to be specified. The modern board config' file format supported by USCIF35 and its DBGJTAG utility allows any type of JTAG scan-path to be accurately described.

The modern board config' file format and the USCIF35 API's that read that format allow user defined names to be assigned to both the sub-paths, and the devices and routers. These are the exact same names specified in the 'subpath=list' and 'device=list' parameters of the [-R routelist / matrixlist] options.

The basics of route and matrix selection

JTAG scan-paths can be simple, containing only un-switched devices. Such scan-paths consist of a single chain of devices with shared TCLK clock and TMS control signals input to each device and a chain of TDO and TDI data signals connecting each prior device to each following device. The diagram below shows a simple scan-path with six devices.

```
TCLK ------+--------+--------+--------+--------+--------+---> TCLK
TMS ----+--+-----+--+-----+--+-----+--+-----+--+-----+--+---> TMS
        |  |     |  |     |  |     |  |     |  |     |  |
        |  |     |  |     |  |     |  |     |  |     |  |
        V  V     V  V     V  V     V  V     V  V     V  V
       +----+   +----+   +----+   +----+   +----+   +----+
       |    |   |    |   |    |   |    |   |    |   |    |
TDI -->|    |-->|    |-->|    |-->|    |-->|    |-->|    |--> TDO
       |    |   |    |   |    |   |    |   |    |   |    |
       +----+   +----+   +----+   +----+   +----+   +----+
```

JTAG scan-paths can also be routed, containing multiple child scan-paths connected by router devices to parent scan-paths. The router devices provide multiple sets of TCLK, TMS, TDO and TDI signals, one set for each of its child scan-paths and one set for its parent scan-path. The router devices can select and de-select one, several or all of its child scan-paths, allowing them to be selectively accessed through the router, from the parent scan-path TCLK, TMS, TDO and TDI signals. The diagram below shows a single router with four sub-paths, two of which each have two devices, and two of which each have three devices.

```
        +------+
        |      |-->----------------------------+
        |      |       +-----+        +-----+  |
        |      |--<----|     |--------|     |--+
        |      |       +-----+        +-----+
        |      |
        |      |-->-------------------------------------------+
TCLK -->|      |       +-----+        +-----+        +-----+  |
TMS  -->|      |--<----|     |--------|     |--------|     |--+
TDI  -->|      |       +-----+        +-----+        +-----+
TDO  <--|      |
        |      |-->-------------------------------------------+
        |      |       +-----+        +-----+        +-----+  |
        |      |--<----|     |--------|     |--------|     |--+
        |      |       +-----+        +-----+        +-----+
        |      |
        |      |-->----------------------------+
        |      |       +-----+        +-----+  |
        |      |--<----|     |--------|     |--+
        +------+       +-----+        +-----+
```

JTAG scan-paths can also be fully hierarchical, containing both un-switched devices and routers that provide access to one or more child scan-paths that themselves can contain both un-switched devices and routers. This enables the creation of hierarchical systems with a single root scan-path and multiple mixed scan-paths connected in a descending parent-child relationship by router devices. The diagram below shows three routers and fifteen un-switched devices connected in a root path and five sub-paths. The root path has two routers, R1 and R2, which have one and two (child) sub-paths respectively. The lower sub-path of R2 contains a third router R3, which itself has two additional (grand-child) sub-paths.

```
        +------+    +-----+        +-----+       +--------+
TDI ->--|  R1  |----| US1 |--------| US2 |-------|   R2   |-->- TDO
        +------+    +-----+        +-----+       +--------+
           ||                                      ||  ||
           |+-->----------------------------+      ||  ||
           |        +-----+        +-----+  |      ||  ||
           +---<----| US3 |--------| US4 |--+      ||  ||
                    +-----+        +-----+         ||  ||
                                                   ||  ||
     +------------------------------------------<--+|  ||
     |  +-----+     +-----+        +-----+          |  ||
     +--| US5 |-----| US6 |--------| US7 |------>---+  ||
        +-----+     +-----+        +-----+             ||
                                                       ||
  +-------------------------------------------------<--+|
  |  +--------+     +-----+        +-----+              |
  +--|   R3   |-----| US8 |--------| US9 |---------->---+
     +--------+     +-----+        +-----+
       ||  ||
       ||  |+-->-------------------------------------------+
       ||  |       +------+       +------+       +------+  |
       ||  +---<---| US10 |-------| US11 |-------| US12 |--+
       ||          +------+       +------+       +------+
       ||
       |+-->-----------------------------------------------+
       |           +------+       +------+       +------+  |
       +---<-------| US13 |-------| US14 |-------| US15 |--+
                   +------+       +------+       +------+
```

The following explanation of how scan-path structure relates to the [-R routelist / matrixlist] options is almost literally just the text of the introduction to the option in reverse order.

The [-f] option of the DBGJTAG utility allows a board config' file to be specified. The modern board config' file format supported by USCIF35 and its DBGJTAG utility allows any type of JTAG scan-path to be accurately described.

The [-S pathlength / brokenpath / integrity / givendata] commands of the DBGJTAG utility are used alone, or with a board config' file, to test the length and reliability of the JTAG scan-path connection between the emulator's TDO and TDI pins.

The modern board config' file format and the USCIF35 API's that read that format allow user defined names to be assigned to both the sub-paths, and the devices and routers.

The 'subpath=list' and 'device=list' parameters of the [-R routelist / matrixlist] options are used to describe JTAG scan-path connections by specifying the names of sub-paths, or the names of devices and routers that are located on those sub-paths. These are the exact same names specified in the 'subpath=list' and 'device=list' parameters of the [-R routelist / matrixlist] options.

The [-R routelist / matrixlist] commands of the DBGJTAG utility extend the [-S pathlength / brokenpath / integrity / givendata] commands. They are used to test the length and reliability of any JTAG scan-path connection that can be created between the emulator's TDO and TDI pins by directly or indirectly selecting one or more sub-paths to be included in the connection between the emulator's TDO and TDI pins.

3.6.2 The Commands

The scan-path test [-R SOFTLIST] command

This command parses the ‘input’, ‘family’ and ‘alias’ board config’ files, then uses the information to load a set of data structures, performs tests using those data structures, and then generates the ‘output’ test results file.

When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files.

The scan-path test [-R ROUTELIST] command

The [-R routelist] command does not perform any scan-tests. It is used to select the sub-paths, in addition to the main path, that are accessed during scan-path tests selected by other commands. If this command is not used then only the main path will be accessed by those other commands.

The command allows the user to provide a single list of devices and sub-paths that are used to select sub-paths. This route-list consists of a series of names separated by ‘+’ characters. The names match objects in the board configure file selected by the [-f] option. To describe devices and sub-paths in a board configure file the modern text format must be used.

When the name of a device is included in the device list then the sub-path on which that device resides will be selected. When the name of a sub-path is included in the sub-path list then that specific sub-path will be selected.

This command has a ‘method’ parameter that chooses between single and plural methods of sub-path selection/de-selection.

This command has a ‘hierarchy’ parameter that expands the reference names into a limited or fully hierarchical list of devices and sub-paths accessed through routers related to the original devices and sub-paths names.

This command has a ‘verbose’ parameter that allows the user to request a report on the interpretation of the lists of routers, sub-paths and devices based on the given board configure file.

The scan-path test [-R ROUTELIST] flow-chart

The [-R routelist] command is implemented in four major phases. The first phase uses the route-list to create the JTAG scan-path connection between the emulator's TDO and TDI pins. The second phase performs a path-length measurement. The third phase performs scan-path tests on the connection. The fourth phase uses the route-list to delete the JTAG scan-path connection between the emulator's TDO and TDI pins.

The phase #1 and #2 of routelist selection

```
        START PHASE #1
         CREATE ROUTE
             |
             |
             V
    +------------------+
    |     select       | <<== -F clock
    |    the clock     |
    |    frequency     |
    +------------------+
             |
             |
             V
    +------------------+
    |     select       | <<== -R routelist,devices/subpaths=list
    |    sub-paths     |
    |  on route-list   |
    +------------------+
             |
             |
             V
       START PHASE #2
      PATH-LENGTH TEST
             |
             |
             V
    +------------------+
    |    measure       | <<== -S pathlength
    |      the         |
    |  path-length     |
    +------------------+
             |
             |
             V
```

The phase #3 and #4 of routelist selection

```
   START PHASE #3
   SCAN-PATH TEST
        |
        |<----------------
        V                ^
+-----------------+      |
|     do the      |      | <<== -S brokenpath/integrity/givendata
|    scan-path    |      |
|      test       |      |
+-----------------+      |
        |                |
        |                |
        V                |
  /-------------\        |
 /    repeat     \ YES   | <<== -S givendata,repeat=number
 \      ?        /------>|
  \-------------/
       NOT
        |
        |
        V
   START PHASE #4
    DELETE ROUTE
        |
        |
        V
+-----------------+
|    de-select    | <<== -R routelist,devices/subpaths=list
|    sub-paths    |
|  on route-list  |
+-----------------+
        |
        |
        V
     FINISH
```

The scan-path test [-R ROUTELIST] phases

This command can be used with a [-S pathlength] command together with a [-S brokenpath / integrity / givendata] command. In this case all four phases will be performed: create route, path-length test, scan-path test, delete route.

This command can be used without a [-S pathlength] command but with a [-S brokenpath / integrity / givendata] command. In this case the four phases will still be performed. This is because the path-length values are required to calibrate the scan-path tests, however their values will not be reported.

This command can be used with only a [-S pathlength] command and without a [-S brokenpath / integrity / givendata] command. In this case only three phases will be performed: create route, path-length test, delete route.

This command can be used without any [-S pathlength / brokenpath / integrity / givendata] commands. In this case only two phases will be performed: create route, delete route.

COMMAND PHASES	without [-S pathlength]	using [-S pathlength]
without [-S scantest]	**1 4**	**1 2 4**
using [-S scantest]	**1 2* 3 4**	**1 2 3 4**
scantest : This refers to any brokenpath, integrity or givendata tests. 2* : This second phase uses an implicit pathlength test, to calibrate the scantest.		

The scan-path test [-R MATRIXLIST] command

The [-R matrixlist] command does not perform any scan-tests. It is used to select the sub-paths, in addition to the main path, that are accessed during scan-path tests selected by other commands. If this command is not used then only the main path will be accessed by those other commands.

The command allows the user to provide multiple lists of devices and sub-paths that are used to select sub-paths. Each route-list consists of a series of names separated by '+' characters. The overall matrix-list consists of a series of route-lists separated by ':' characters. The names match objects in the board configure file selected by the [-f] option. To describe devices and sub-paths in a board configure file the modern text format must be used.

When the name of a device is included in the device list then the sub-path on which that device resides will be selected. When the name of a sub-path is included in the sub-path list then that specific sub-path will be selected.

This command has a 'method' parameter that chooses between single and plural methods of sub-path selection/de-selection.

This command has a 'hierarchy' parameter that expands the reference names into a limited or fully hierarchical list of devices and sub-paths accessed through routers related to the original devices and sub-paths names.

This command has a 'reorder' parameter that is used to choose the order in which each route-list within a matrix-list is applied to create scan-path connections between the emulator's TDO and TDI pins.

This command has a 'repeat' parameter that is used to choose the number of times in which each route-list within a matrix-list is applied to create, test and delete scan-path connections between the emulator's TDO and TDI pins.

This command has a 'verbose' parameter that allows the user to request a report on the interpretation of the lists of routers, sub-paths and devices based on the given board configure file.

The scan-path test [-R MATRIXLIST] flow-chart

The [-R matrixlist] command is implemented in three major phases. The first phase performs a router-switch test on every route-list in the matrix list, but does not perform any path-length measurement or scan-path test. The second phase performs path-length measurements on every route-list in the matrix list, but does not perform any scan-path test. The third phase performs scan-path tests on every route-list in the matrix list.

The phase #1 of matrixlist selection

```
        START PHASE #1
      ROUTER-SWITCH TEST
             |
             |
             V
     +-----------------+
     |     select      | <<== -F clock
     |    the clock    |
     |    frequency    |
     +-----------------+
             |
------------>|
^            V
|    +-----------------+
|    |     select      | <<== -R matrixlist,devices/subpaths=list
|    |    sub-paths    |
|    |  on route-list  |
|    +-----------------+
|            |
|            |
|            V
|    +-----------------+
|    |    de-select    | <<== -R matrixlist,devices/subpaths=list
|    |    sub-paths    |
|    |  on route-list  |
|    +-----------------+
|            |
|            |
|            V
|       /-----------\
|      /  pick next  \
| YES /  route-list   \
|<----\  from matrix  /
       \     ?       /
        \-----------/
            NOT
             |
             |
             V
```

The phase #2 of matrixlist selection

```
        START PHASE #2
       PATH-LENGTH TEST
             |
------------>|
^            V
|    +-----------------+
|    |     select      | <<== -R matrixlist,devices/subpaths=list
|    |    sub-paths    |
|    |  on route-list  |
|    +-----------------+
|            |
|            |
|            V
|    +-----------------+
|    |    measure      | <<== -S pathlength
|    |      the        |
|    |  path-length    |
|    +-----------------+
|            |
|            |
|            V
|    +-----------------+
|    |    de-select    | <<== -R matrixlist,devices/subpaths=list
|    |    sub-paths    |
|    |  on route-list  |
|    +-----------------+
|            |
|            |
|            V
|       /-----------\
|      /  pick next  \
| YES /  route-list   \
|<----\  from matrix  /
       \     ?       /
        \-----------/
            NOT
             |
             |
             V
```

The phase #3 of matrixlist selection

```
       START PHASE #3
       SCAN-PATH TEST
             |
------------>|<-------------------
^            V                   ^
|    +-----------------+         |
|    |     select      |         | <<==
|    |    sub-paths    |         |   -R matrixlist,devices/subpaths=list
|    |  on route-list  |         |
|    +-----------------+         |
|            |                   |
|            |<--------------    |
|            V              ^    |
|    +-----------------+    |    |
|    |     do the      |    |    | <<==
|    |    scan-path    |    |    |   -S brokenpath/integrity/givendata
|    |      test       |    |    |
|    +-----------------+    |    |
|            |              |    |
|            |              |    |
|            V              |    |
|      /-------------\      |    |
|     /    repeat     \ YES |    | <<==
|     \      ?        /---->|    |   -S givendata,repeat=number
|      \-------------/           |
|           NOT                  |
|            |                   |
|            |                   |
|            V                   |
|    +-----------------+         |
|    |    de-select    |         | <<==
|    |    sub-paths    |         |   -R matrixlist,devices/subpaths=list
|    |  on route-list  |         |
|    +-----------------+         |
|            |                   |
|            |                   |
|            V                   |
|       /-----------\            |
|      /  pick next  \           |
| YES /  route-list   \          |
|<----\  from matrix  /          |
|      \      ?      /           |
|       \-----------/            |
|           NOT                  |
|            |                   |
|            |                   |
|            V                   |
       /-------------\           | <<==
      /    repeat     \ YES      |   -R matrixlist,repeat=number
      \      ?        /--------->|
       \-------------/
            NOT
             |
             |
             V
           FINISH
```

The scan-path test [-R MATRIXLIST] phases

This command can be used with a [-S pathlength] command together with a [-S brokenpath / integrity / givendata] command. In this case all three phases will be performed: router-switch test, then path-length test, then scan-path test.

This command can be used without a [-S pathlength] command but with a [-S brokenpath / integrity / givendata] command. In this case the three phases will still be performed. This is because the path-length values are required to calibrate the scan-path tests, however their values will not be reported.

This command can be used with only a [-S pathlength] command and without a [-S brokenpath / integrity / givendata] command. The first phase and its router-switch test will performed. The second phase and its path-length measurement will also be performed. The third phase and its scan-path test will be skipped.

This command can also be used without any [-S pathlength / brokenpath / integrity / givendata] commands. The first phase and its router-switch test will performed. The second phase and its path-length measurement will be skipped. The third phase will be performed without any scan-path test, leaving only the core sub-path select and de-select actions required for creating JTAG scan-path connections between the emulator's TDO and TDI pins. All of the [-R matrixlist] parameters remain effective during the third phase, enabling thorough testing of router select and de-select actions.

COMMAND PHASES	without [-S pathlength]	using [-S pathlength]
without [-S scantest]	**1 3***	**1 2**
using [-S scantest]	**1 2* 3**	**1 2 3**

```
scantest : This refers to any brokenpath,
           integrity or givendata tests.

 2* : This second phase uses an implicit
      pathlength test, to calibrate the scantest.

 3* : This third phase uses only the router
      select/deselect, without any scantest.
```

3.6.3 The Parameters

The INPUT parameter

This parameter selects the name of the board config' file that is used to source variable and scan-path information for the utility. The 'input' file is processed after the 'family' and 'alias' files.

If this parameter is not provided then the default file 'input.txt' will be read.
If this parameter is used with the value '0' then no file will be read.

The OUTPUT parameter

This parameter selects the name of the board config' file that is output by the utility. The format of the output board config' file is selected by the choice of the 'legacy', 'modern' or 'xml' commands. The content of the output board config' file is influenced by the 'indent', 'comment', 'xplicit' and 'crushed' parameters.

If this parameter is not provided then the default file 'output.txt' will be written.
If this parameter is used with the value '0' then no file will be written.

The FAMILY parameter

This parameter selects the name of an extra board config' file that is used to source family information for the utility. The 'family' file is processed before the 'input' file.

If this parameter is not provided then the default file 'xdsfamily.cfg' will be read.
If this parameter is used with the value '0' then no file will be read.

The ALIAS parameter

This parameter selects the name of an extra board config' file that is used to source alias information for the utility. The 'alias' file is processed before the 'input' file.

If this parameter is not provided then the default file 'xdsalias.cfg' will be read.
If this parameter is used with the value '0' then no file will be read.

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

The DEVICES parameter

This parameter is shared by both of the `[-R routelist / matrixlist]` commands. It may be a single name or a route-list, which is a list of one or more names separated by '`+`' characters, or it may be a matrix-list, which is a list of one or more route-lists separated by ':' characters. Each name is a single device or a single router. The names are taken from the board configure file selected by the `[-f]` option.

The `[-R routelist]` command accepts only single names and route-lists.
The `[-R matrixlist]` command accepts single names, route-lists and matrix-lists.

The list of names in a route-list imply a scan-path connection between the emulator's TDO and TDI pins that passes through the devices and routers identified by those names. The scan-path connection is created and deleted by an appropriate (and possibly hierarchical) selection and de-selection of routers sub-paths.

The list of route-lists in a matrix-list describe multiple scan-path connections between the emulator's TDO and TDI pins, each of which pass through the devices and routers identified by those route-lists. The multiple scan-path connections are created and deleted one after the other by appropriate (and possibly hierarchical) selection and de-selection of routers sub-paths.

The SUBPATHS parameter

This parameter is shared by both of the `[-R routelist / matrixlist]` commands. It may be a single name or a route-list, which is a list of one or more names separated by '+' characters, or it may be a matrix-list, which is a list of one or more route-lists separated by ':' characters. Each name is a single sub-path. The names are taken from the board configure file selected by the `[-f]` option.

The `[-R routelist]` command accepts only single names and route-lists.
The `[-R matrixlist]` command accepts single names, route-lists and matrix-lists.

The list of names in a route-list describe a scan-path connection between the emulator's TDO and TDI pins that passes through the sub-paths identified by those names. The scan-path connection is created and deleted by an appropriate (and possibly hierarchical) selection and de-selection of routers sub-paths.

The list of route-lists in a matrix-list imply multiple scan-path connections between the emulator's TDO and TDI pins, each of which pass through the sub-paths identified by those route-lists. The multiple scan-path connections are created and deleted one after the other by appropriate (and possibly hierarchical) selection and de-selection of routers sub-paths.

The METHOD parameter

This parameter is used to choose between single and plural methods of sub-path selection/de-selection. Both methods create the exact same scan-path connection between the emulator's TDO and TDI pins. The difference is only the manner in which the operation are performed on each router.

This parameter has string values whose default is marked bold.

`**p**lural`	Select and de-select items on a route-list all together.
`single`	Select and de-select items on a route-list one at a time.

The HIERARCHY parameter

This parameter expands the devices and sub-paths selected by the route-lists lists provided by the ‘devices’ and ‘sub-paths’ parameters.

Using a device name that is a router with the ‘children’ parameter value is equivalent to also adding any devices on its sub-paths to the list the names. Using a device name that is a router with the ‘descendant’ parameter value is equivalent to also adding any devices on its sub-paths to the list of names, and then recursing this search through any routers found on the sub-paths. Using a sub-path name with the ‘children’ and ‘descendant’ parameter values is equivalent to using the device names of every router on that sub-path.

This parameter has string values whose default is marked bold.

`**s**ibling`	Access only the named items.
`children`	Access named items and their children.
`descendant`	Access named items plus their descendants.

The REORDER parameter

This parameter is used to choose the order in which each route-list within a matrix-list is applied to create scan-path connections between the emulator’s TDO and TDI pins.

These choices provide both trivial and complex modifications of the matrix-list without re-writing it. The forward and reverse choices apply the route-lists in forward and reverse order. The combine choice expands the number of route-lists in the matrix-list to ensure every combination of two route-lists are applied. The shuffle choice shuffles the route-lists in the matrix-list to allow random permutations of the route-lists to be applied. The permute choice expands the number of route-lists in the matrix-list to ensure every permutation of those route-lists are applied.

When each route-list has only single device the matrix-list describes a sequence of single device select / de-select actions similar to those used by DSP and ARM drivers. In this situation the combine choice expands the route-list to create every possible sequence of single device select / de-select actions. This is equivalent to an exhaustive test of every possible interaction between DSP and ARM drivers.

A shuffle request with fewer than four route-lists in the matrix-list will be converted to a permute request. Only shuffle requests with four or greater route-lists have a performance advantage over permute requests.

This parameter has string values whose default is marked bold.

`**f**orward`	Apply route-lists in forward order - n items.
`reverse`	Apply route-lists in reverse order - n items.
`combine`	Apply route-lists in every combination of two.
`shuffle`	Apply route-lists in random permutation.
`permute`	Apply route-lists in every permutation.

The REPEAT parameter

This parameter represents the number of times the test is applied. A running count of the status of the major option is displayed. The repetition of the test will be halted when any key is pressed.

If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key.

This parameter has numerical and string values whose default is marked bold.

`0`	The test is performed continuously.
`1`	The test is performed just once.
`2 - n`	The test is performed this many times.

Some choices have synonyms:

`default`	The test is performed just once.
`single`	The test is performed just once.
`forever`	The test is performed continuously.

The NOSCAN parameter

This parameter is used to disable the low-level mini-drivers that implement the actual sub-path select/de-select operations for each router family. This option is valuable when developing both the [-R routelist / matrixlist] commands and the hierarchical routing support in the underlying scan-path management software. It allows software development to be done with target systems that lack actual router hardware.

This parameter has boolean values whose default is marked bold.

no	Enable the low-level mini-drivers.
yes	Disable the low-level mini-drivers.

3.7 Test the Emulator with [-T]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of ‘name=value’ pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-T help
```

Test the emulator control a repeated number of times.

```
-T control,repeat=number,verbose=boolean
```

Test the emulator memory a repeated number of times.

```
-T memory,repeat=number,verbose=boolean
```

Test the emulator hardware via its observe registers.

```
        -T observe
```

3.7.1 The Introduction

The basics of the emulator tests

The details of the emulator tests

The repeated tests are halted when any key is pressed. If a count of 1 - n is used then the test repeats that often. If a count of 0 is used then the test repeats forever. The string ‘single’ also represents the repeat count of 1. The string ‘forever’ also represents the repeat count of 0. The default repeat count is ‘1’.

3.7.2 The Commands

The emulator test [-T CONTROL] command

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 the [-y] option 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 emulator test [-T MEMORY] command

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] and [-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 emulator test [-T OBSERVE] command

3.7.3 The Parameters

The REPEAT parameter

This parameter represents the number of times the test is applied. A running count of the status of the major option is displayed. The repetition of the test will be halted when any key is pressed.

If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key.

This parameter has numerical and string values whose default is marked bold.

`0`	The test is performed continuously.
`1`	The test is performed just once.
`2 - n`	The test is performed this many times.

Some choices have synonyms:

`default`	The test is performed just once.
`single`	The test is performed just once.
`forever`	The test is performed continuously.

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

3.8 *Test the Cable and Pod with [-U]*

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-U help
```

Test a cable and pod using the Quad Router hardware with Test460 VHDL.

```
-U test460,pathselect=string,
           frequency=value,
           sendclock=string,
           inputclock=cable(s),
           returnclock=cable,
           erasefail=boolean,
           typefail=string,
           waitfail=string,
           whenfail=string,
           delayfail=number,
           occurfail=number,
           emu0pin=cable(s),
           emu1pin=cable(s),
           switchport=string,
           verbose=boolean
```

3.8.1 The Introduction

The basics of cable and pod tests

The details of cable and pod tests

3.8.2 The Commands

The cable and pod tests [-U TEST460] command

3.8.3 The Parameters

The PATHSELECT parameter

This parameter is only useful for emulators connected to the test-port. It provides limited scan-path routing (with relatively high speed) through the main-port of the Quad-Router without adding any JTAG TAP to that main path. There are four external cables and one internal target that can be individually selected. The external cables can also also be selected together as a single longer scan-path. The external cables and internal target can also be bypassed to produce a loop-back connection that is also the default. The choices are:

`loopback`	Select the internal loop-back connection.
`A:cable`	Select the external cable A.
`B:cable`	Select the external cable B.

C:cable	Select the external cable C.
D:cable	Select the external cable D.
internal	Select the internal target only.
external	Select all four external cables.

The FREQUENCY parameter

This parameter is only useful for emulators connected to the test-port. It is interpreted as a (8 hex digit) 32-bit data value for the frequency that the local PLL is programmed to output for possible use by these parameters:

‘SendClock’, ‘InputClock’ and ‘ReturnClock’

This parameter has string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix ‘5.2 The Frequency of TCLKO and TCLKR’.

The SENDCLOCK parameter

This parameter is only useful for emulators connected to the test-port. It selects the common clock used for outputs to the four external cables. It can either be the main port’s TCLK input, or the locally generated TCLK output by on the PLL on the quad-router, or the fixed reference clock used by that same programmable clock generator. The default is marked bold. The choices are:

reference	Use the local fixed reference TCLK
programme	Use the local programmable TCLK
hostclock	Use the host emulator TCLK input on the main-port

The INPUTCLOCK parameter

This parameter is only useful for emulators connected to the test-port. It operates the multiplexers in each 60-pin header of the four external cables. The multiplexers select the clock used to sample inputs from those four external cables. The default for each of the cables is the clock returned from the corresponding target. The alternative for each of the cables is to use the common clock selected by the ‘SendClock’ parameter. The choices are a ‘+’ separated list of one or more of:

unique	Use the unique target clocks for each of the cables.
A:cable	Use the common send clock for target A.
B:cable	Use the common send clock for target B.
C:cable	Use the common send clock for target C.
D:cable	Use the common send clock for target D.

The RETURNCLOCK parameter

This parameter is only useful for emulators connected to the test-port. It selects the TCLK returned via the main-port. It can be any one of the ‘InputClock’ used to sample the four external cables. The default is marked bold. The choices are just one of:

`A:cable`	Use the clock returned from cable A.
`B:cable`	Use the clock returned from cable B.
`C:cable`	Use the clock returned from cable C.
`D:cable`	Use the clock returned from cable D.

The ERASEFAIL parameter

This parameter is only useful for emulators connected to the test-port. It overrides all other '`...Fail`' parameters. It removes any existing error that is being asserted, and erases the error configuration. The choices are:

`false`	The error configuration is not altered.
`true`	The error configuration and any outstanding error are erased.

The TYPEFAIL parameter

This parameter is only useful for emulators connected to the test-port. It selects the type of error that will be generated on the main-port. The choices are:

`nothing`	The failure type will be nothing.
`cablebreak`	The failure type will be a cable-break.
`powerloss`	The failure type will be a power-loss.
`deadclock`	The failure type will be a dead-clock.
`headerswap`	The failure type will be a swap-header.
`stuckfault`	The failure type will be a stuck-fault.

The WAITFAIL parameter

This parameter is only useful for emulators connected to the test-port. It selects if the generation of the error will wait for a trigger on the main port. The choices are:

`nothing`	The failure waits for nothing.
`trigger`	The failure waits until trigger.

The WHENFAIL parameter

This parameter is only useful for emulators connected to the test-port. It selects the event or JTAG state that triggers the generation of an error on the main-port. The choices are:

`runsignal`	The failure occurs when run signal is asserted.
`mainidle`	The failure occurs when main port is in idle state.
`mainscan`	The failure occurs when main port is in scan state.
`mainend`	The failure occurs when main port is in end state.

The DELAYFAIL parameter

This parameter is only useful for emulators connected to the test-port. It selects the number of clocks delay before the generation of an error on the main-port. This parameter is interpreted as a hexa-decimal or decimal data value limited to an acceptable range. The choices are:

`0 - 65535`	Require literal clock delay.

The OCCURFAIL parameter

This parameter is only useful for emulators connected to the test-port. It selects the number of occurrences before the generation of an error on the main-port. This parameter is interpreted as a hexa-decimal or decimal data value limited to an acceptable range. The choices are:

`0 - 65535`	Require literal occurrences.

The EMU0PIN / EMU1PIN parameters

This parameter is only useful for emulators connected to the test-port. It enables bi-directional multiplexers that route signals on the EMU0 / EMU1 pin between the host-port and the four cables. The choices are a '+' separated list of:

`none`	Enable none of the bidir-muxes for the pin.
`A:cable`	Enable bidir-mux on cable A for the pin.
`B:cable`	Enable bidir-mux on cable B for the pin.
`C:cable`	Enable bidir-mux on cable C for the pin.
`D:cable`	Enable bidir-mux on cable D for the pin.

The SWITCHPORT parameter

This parameter is only useful for emulators connected to the main-port. It requests the emulator to generate sequences using a shadow protocol during non-scan states. The main-port of the quad-router interprets the sequence as a command to switch the functions of internal main-port and internal test-port between the physical main-port and physical test-port. The choices are:

`normal`	The physical main/test ports connect the main/test ports.
`switch`	The physical main/test ports connect the test/main ports.

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

3.9 Handle FPGA/CPLD Data with [-D]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of ‘name=value’ pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-D help
```

Do not program or query any of the FPGA/CPLD’s.

```
-D nothing
```

Query the named FPGA/CPLD to report the version of its VHDL code.

```
-D version,type=title,
           notouch=boolean,verbose=boolean
```

Erase the named FPGA/CPLD.

```
-D erase,type=title,
         notouch=boolean,verbose=boolean
```

Program the named FPGA/CPLD using the default internal VHDL code.

```
-D internal,type=title,
            notouch=boolean,
            verbose=boolean
```

Program the named FPGA/CPLD using VHDL from source/binary/data files.

```
-D external,type=title,
            data=filename.c/rbf/ttf,
            notouch=boolean,
            verbose=boolean
```

Generate a ‘C’ language source file from a FPGA/CPLD data file.

```
-D source,type=title,
          version=number,
          input=filename.ttf,
          output=filename.c,
          verbose=boolean
```

Generate a raw binary file from a FPGA/CPLD data file.

```
-D binary,input=filename.ttf,
          output=filename.rbf,
          verbose=boolean
```

Generate a encoded/decoded data file using a chosen algorithm.

```
-D crush,input=filename.txt,
         output=filename.txt,
         action=huffman/crc32,
         direction=encode/decode,
         verbose=boolean
```

3.9.1 The Introduction

The basics of handling FPGA/CPLD data

The [-D nothing / version / erase / internal / external] commands of the DBGJTAG utility provide tools for programming the FPGA/CPLD components that are used to construct emulator hardware. The data used to program the FPGA/CPLD components is compiled from VHDL source code by development tools provided by the FPGA/CPLD manufacturer.

The [-D source / binary / crush] commands of the DBGJTAG utility provide tools for converting FPGA/CPLD data output by the development tools provided by the FPGA/CPLD manufacturer. to formats appropriate for building emulator programs and adapters with embedded FPGA/CPLD programming capabilities.

3.9.2 The Commands

The FPGA/CPLD data [-D NOTHING] command

This command is used to instruct the utility to not implicitly or explicitly load any FPGA/CPLD's with new internal or external VHDL data. The hardware will instead operate with the existing VHDL data already programmed into the FPGA/CPLD. This command is typically used to support other command-line operations by suppressing any potential modification of previously loaded external FPGA/CPLD data by an unwanted re-loading of internal data.

The FPGA/CPLD data [-D VERSION] command

This command retrieves the version of the VHDL data currently programmed
into the FPGA/CPLD. The VHDL data is not erased or re-programmed.

The FPGA/CPLD data [-D ERASE] command

This command erases the the VHDL data currently programmed into the FPGA/CPLD. This is the same erasure that automatically occurs before re-programming with internal or external VHDL data.

The FPGA/CPLD Data [-D INTERNAL] command

This command is used to re-program the FPGA/CPLD with the emulator's built-in VHDL data. This is the same VHDL data that is used by the [-r] option when the emulator is reset by this utility.

The FPGA/CPLD data [-D EXTERNAL] command

This command is used to re-program the FPGA/CPLD with VHDL data read from source/binary files or a VHDL data file instead of the default built-in VHDL data. This capability simplifies emulator development by avoiding re-builds of the emulator program or adapter.

The FPGA/CPLD data [-D SOURCE] command

This command is is a file formatting operation that does not require the [-d] and [-p] options, nor access to an emulator. This command converts FPGA/CPLD data to a 'C' language source code file. The file contains FPGA/CPLD data, size and compression information, plus a check-sum. The file is used to build an emulator program with updated built-in default FPGA/CPLD data. This capability speeds emulator development when the emulator program is rebuilt.

- The default input filename extension is 'ttf'.
- The default output filename extension is 'c'.

The FPGA/CPLD data [-D BINARY] command

This command is is a file formatting operation that does not require the [-d] and [-p] options, nor access to an emulator. This command converts FPGA/CPLD data to a raw binary data file. The raw binary data file is is used in testing emulator software. This capability speeds emulator development when software is first built.

- The default input filename extension is 'ttf'.
- The default output filename extension is 'rbf'.

The FPGA/CPLD data [-D CRUSH] command

This command is is a file formatting operation that does not require the [-d] and [-p] options, nor access to an emulator. This command encodes/decodes data using the Huffman algorithms or CRC32 algorithms. The resulting data is used in testing emulator software. This capability speeds emulator development when software is first built.

- The default input filename extension is 'txt'.
- The default output filename extension is 'txt'.

3.9.3 The Parameters

The TYPE parameter

This parameter is used by the 'version', 'erase' and 'internal' and 'external' programming commands. It selects which FPGA/CPLD within the emulator or its cable is to be selected or accessed.

This parameter has string values whose default is marked bold.

`**e**mulator`	Select the controller for standard emulators.
`multipod`	Select the multi-purpose pod for standard emulators.
`client`	Select the controller for trace-enabled emulators.
`server`	Select the trace-pod for trace-enabled emulators.

The DATA parameter

This parameter is used by the ‘external’ file programming commands. It names the VHDL data file to be used when programming an FPGA/CPLD using external data instead of the default data built in to the emulator. The default filename extension is ‘ttf’. The supported file formats are:

- ‘C’ language source code with a ‘c’ extension,
- Raw Binary File format with a ‘rbf’ extension,
- Tabular Text File format with a ‘ttf’ extension.

The NOTOUCH parameter

This parameter is used by the ‘version’, ‘erase’ and ‘internal’ and ‘external’ programming commands. It instructs that the emulator program or adapter is to avoid touching (reading or writing) hardware registers after the FPGA/CPLD is programmed.

This parameter has boolean values whose default is marked bold.

`no` (**n**o)	It is OK to touch hardware registers after the FPGA/CPLD is programmed.
`yes`	Do not touch hardware registers after the FPGA/CPLD is programmed.

The VERSION parameter

This parameter is an up-to 32-bit value that is used by the ‘source’ file conversion command. The ‘source’ command outputs ‘C’ language source code with an optional ‘c’ extension. This parameter, input as decimal or hexa-decimal, provides a method of embedding the FPGA/CPLD version number in that file. Note that the all-zeros value is reserved to represent un-programmed FPGA/CPLD’s.

This parameter has numerical values whose default is marked bold.

`0x0 - `**`0`**`xFFFFFFFF`	The version number of the FPGA/CPLD.

The INPUT parameter

This parameter names the input VHDL data file to be used by the ‘source’, ‘binary’ and ‘crush’ file conversion commands.

- The ‘source’ command inputs files with the Tabular Text File format and an optional ‘ttf’ extension.
- The ‘binary’ command inputs files with the Tabular Text File format and an optional ‘ttf’ extension.
- The ‘crush’ command inputs text files with an optional ‘txt’ extension.

The OUTPUT parameter

This parameter names the output VHDL data file to be generated by the ‘source’, ‘binary’ and ‘crush’ file conversion commands.

- The ‘source’ command outputs ‘C’ language source code with an optional ‘c’ extension.
- The ‘binary’ command outputs the Raw Binary File format with an optional ‘rbf’ extension.
- The ‘crush’ command outputs text files with an optional ‘txt’ extension.

The ACTION parameter

This parameter names the action to be performed on the VHDL data by the ‘crush’ file conversion command.

`huffman`	Encode or decode the data using Huffman’s algorithm.
`crc32`	Encode or decode a CRC-32 value using the data.

The DIRECTION parameter

This parameter names the direction of the action to be performed on the VHDL data by the ‘crush’ file conversion command.

`encode`	Encode using the Huffman/CRC-32 algorithm.
`decode`	Decode using the Huffman/CRC-32 algorithm.

The VERBOSE parameter

This parameter is used to request additional information from the command. This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies ‘`yes`’.

3.10 Handle Board Config’ Files with [-B]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of ‘name=value’ pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-B help
```

Simply parse a board config’ file for correctness.

```
-B parse,input=filename,family=filename,
         alias=filename,verbose=boolean
```

Input a board config' file and translate to the legacy text format.

```
-B legacy,input=filename,output=filename,
          family=filename,alias=filename,
          verbose=boolean,
          indent=boolean,
          comment=boolean
```

Input a board config' file and translate to the modern text format.

```
-B modern,input=filename,output=filename,
          family=filename,alias=filename,
          verbose=boolean,
          xplicit=boolean,
          crushed=boolean,
          exclude=boolean
          indent=boolean,
          comment=boolean
```

Input a board config' file and translate to the verbose XML format.

```
-B xml,input=filename,output=filename,
       family=filename,alias=filename,
       verbose=boolean,
       xplicit=boolean,
       crushed=boolean,
       exclude=boolean
       indent=boolean,
       comment=boolean
```

3.10.1 The Introduction

The basics of handling board config' files

The `[-B parse / legacy / modern / XML]` commands of the DBGJTAG utility provide tools used to read and write board config' files in several formats. They provide a stand-alone tool to test the API's and data structures in the USCIF35 software stack that are used to implement those operations.

This utility fully supports the legacy version 2.0 format of board config' files used prior to release 2.4 of Code Composer Studio. It can read and write the config' variables and scan-path descriptions on both input and output.

This utility also fully supports the modern version 3.5 format of board config' files first used by release 2.4 of Code Composer Studio. It can read and write the config' variables and scan-path descriptions on both input and output.

This utility also has optional support for the experimental XML format board config' files. When that support is included in builds of this utility it can read and write the config' variables and scan-path descriptions on both input and output.

3.10.2 The Commands

The board config' file [-B PARSE] command

This command parses the 'input', 'family' and 'alias' board config' files. When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files.

The board config' file [-B LEGACY] command

This command parses the 'input', 'family' and 'alias' board config' files, then uses the information to load a set of data structures, and then generates the 'output' board config' file using the legacy format.

When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files.

The content of the 'output' board config' file is influenced by the 'indent', 'comment', 'xplicit' and 'crushed' parameters.

The board config' file [-B MODERN] command

This command parses the 'input', 'family' and 'alias' board config' files, then uses the information to load a set of data structures, and then generates the 'output' board config' file using the the modern format.

When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files.

The content of the 'output' board config' file is influenced by the 'indent', 'comment', 'xplicit' and 'crushed' parameters.

The board config' file [-B XML] command

This command parses the 'input', 'family' and 'alias' board config' files, then uses the information to load a set of data structures, and then generates the 'output' board config' file using the experimental XML format.

When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files.

The content of the 'output' board config' file is influenced by the 'indent', 'comment', 'xplicit' and 'crushed' parameters.

3.10.3 The Parameters

These commands have input, family and alias parameters for loading board configuration files. The input parameter is used to load the familiar content of such files: the variable descriptions and scan-path description containing devices, routers and sub-paths. The family and alias parameters are used to load the standard family and alias descriptions that support the scan-path description.

This arrangement of multiple file for specific purposes is only a matter of convenience. A single file containing the complete set of descriptions may be loaded using only the input parameter.

The INPUT parameter

This parameter selects the name of the board config' file that is used to source variable and scan-path information for the utility. The 'input' file is processed after the 'family' and 'alias' files.

If this parameter is not provided then the default file 'input.txt' will be read.
If this parameter is used with the value '0' then no file will be read.

The OUTPUT parameter

This parameter selects the name of the board config' file that is output by the utility. The format of the output board config' file is selected by the choice of the 'legacy', 'modern' or 'xml' commands. The content of the output board config' file is influenced by the 'indent', 'comment', 'xplicit' and 'crushed' parameters.

If this parameter is not provided then the default file 'output.txt' will be written.
If this parameter is used with the value '0' then no file will be written.

The FAMILY parameter

This parameter selects the name of an extra board config' file that is used to source family information for the utility. The 'family' file is processed before the 'input' file.

If this parameter is not provided then the default file 'xdsfamily.cfg' will be read.
If this parameter is used with the value '0' then no file will be read.

The ALIAS parameter

This parameter selects the name of an extra board config' file that is used to source alias information for the utility. The 'alias' file is processed before the 'input' file.

If this parameter is not provided then the default file 'xdsalias.cfg' will be read.
If this parameter is used with the value '0' then no file will be read.

The INDENT parameter

This parameter controls the amount of indentation for legibility included in the output file. If the default value of 'no' is used, then a slightly smaller output file is created, because no indents will be included before each description. If the value used is 'yes', then a slightly larger file is created, because one or more indents will be included before each variable, package, family, device and sub-path description.

This parameter has boolean values whose default is marked bold.

no	Output without any indents.
yes	Output with indents for legibility.

The COMMENT parameter

This parameter controls the amount of explanatory comments included in the output file. If the default value of 'no' is used, then a smaller output file is created, because no comments will be included with the descriptions. If the value used is 'yes', then a larger file is created, because comments will be provided for each variable, package, family, device and sub-path description.

This parameter has boolean values whose default is marked bold.

no	Output without any comments.
yes	Output with lots of comments.

The XPLICT parameter

This parameter controls the number of sub-path parameters included in the output file. If the default value of 'no' is used, then a smaller output file is created, because only the essential parameters will be included in sub-path descriptions. If the value used is 'yes', then a larger file is created, because more parameters (including 'parent' and 'devices') will be appended.

This parameter has boolean values whose default is marked bold.

no	Less output with essential sub-path parameters.
yes	More output with every sub-path parameters.

The CRUSHED parameter

This parameter controls the number of package, family and alias descriptions included in the output file. If the default value of 'no' is used, then a larger output file is created, because all of the package and family descriptions are included in the output file. If the value used is 'yes', then a smaller file is created, because only descriptions required by the scan-path description (devices, routers and sub-paths) are included in the output file.

This parameter has boolean values whose default is marked bold.

no	Output every package, family and alias description.
yes	Output only essential package, family and alias descriptions.

The EXCLUDE parameter

This parameter controls the number of package, family and alias descriptions included in the output file. If the default value of 'no' is used, then a larger output file is created, because some or all of the descriptions are included in the output file, as chosen by the other 'crushed' parameter. If the value used is 'yes', then a smaller file is created, because none of the descriptions required by the scan-path description (devices, routers and sub-paths) are included in the output file.

This parameter has boolean values whose default is marked bold.

no	Output some package, family and alias description.
yes	Output no package, family and alias descriptions.

The VERBOSE parameter

This parameter does NOT affect the content of the output file, instead it affects the content of the standard output that is directed to the console.

If the default value of 'no' is used, then little information is output to the console. If the value used is 'yes', then line by line information is output to the console as the three files are parsed and the one file is written. If a syntax error is found when parsing files then the line number of the syntax error will be reported.

This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

3.11 Handle Config' Variables with [-V]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-V help
```

List the configure variables and values that will be passed to the emulator.

```
-V merge
```

Apply an arbitrary set of configure variables and values.

```
-V apply,name1=value1,name2=value2,...
```

3.11.1 The Introduction

The basics of handling config' variable

The [-V merge] command of the DBGJTAG utility is used for listing the config' variables and values that will be passed to the emulator after the command line and board config' file have been merged.

The [-V apply] command of the DBGJTAG utility is used for creating and deleting any config' variable directly from the command-line. The value applied from the command line will override or suppress an occurrence of the variable provided in board config' files.

3.11.2 The Commands

The handling config' variable [-V MERGE] command

The [-V merge] command is a tool for listing the variables and values that will be passed to the emulator after the command line and board config' file have been merged.

The handling config' variable [-V APPLY] command

The `[-V apply]` command is a tool for creating and deleting any config' variable directly from the command-line. Those variables can affect:

- Any board config' file input with the [-f] option.
- Any board config' files input and output with the [-B] option.
- Any DBGJTAG emulator or scan-path operation whose behaviour can be modified by config' variables.

Any parameter with a value that is used with the `[-V apply]` command will be interpreted as specifying a config' variable to be created or over-written using the parameter's name and value. Any parameter lacking a value that is used with the `[-V apply]` command will specify a config' variable to be deleted. This command is unusual in that the DBGJTAG supports arbitrarily named parameters both with and without values, and also supports up-to 64 instances of arbitrarily named parameters.

The variable names should be in the format used by modern format of board config' files, with a section name and a variable name. If the section name is missing (which means lacks a '.' character to separate the section and name) then a default section convenient for USCIF35 and SEPK developers will be applied.

- Here are two parameters with full section and variable names.

  ```
  -V apply,section.string=abcd,section.number=0123
  ```

  Here are the resulting config' variables:

  ```
  $ section
      string=abcd
      number=0123
  $ /
  ```

- Here are two parameters without sections, and variable names that lack any special prefix. The default '`uscif`' section name is implied.

  ```
  -V apply,tclk_string=abcd,tclk_number=0123
  ```

  Here are the resulting config' variables:

  ```
  $ uscif
      tclk_string=wxyz
      tclk_number=3210
  $ /
  ```

- Here are two parameters without sections, and variable names that use the '`pod_`' prefix. The default '`sepk`' section name is implied.

  ```
  -V apply,pod_string=abcd,pod_number=0123
  ```

  Here are the resulting config' variables:

  ```
  $ sepk
  ```

```
            pod_string=wxyz
            pod_number=3210
        $ /
```

- Here are two parameters without sections, and variable names that use the '`net_`' prefix. The default '`sepk`' section name is implied.

  ```
  -V apply,net_host=127.0.0.1,net_port=4567
  ```

  Here are the resulting config' variables:

  ```
  $ sepk
      net_host=127.0.0.1
      net_port=4567
  $ /
  ```

3.11.3 The Parameters

The [-V apply] command and [-V merge] command
of the DBGJTAG utility have no defined parameters.

3.12 *Handle Error Numbers with [-E]*

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-E help
```

Explain a single error number.

```
-E single,number=value,
          verbose=boolean
```

Explain a range of error numbers.

```
-E range,lowest=value,highest=value,
         verbose=boolean
```

Simply parse an error description file for correctness.

```
-E parse,input=filename,verbose=boolean
```

Input an error description file, and output the header or description.

```
-E errors,input=filename,output=filename,
          filetype=string,verbose=boolean,
          indent=boolean,comment=boolean
```

3.12.1 The Introduction

The basics of handling error numbers

The [-E] option provides a set of commands for handling error numbers and error config' files.

The detailed description of the error numbers, titles and descriptions referenced by the USCIF35 software stack, its parent utilities, and its child adapters is stored in the error config' file named 'xdserror.cfg' that is shipped with each binary release of USCIF35.

The detailed description of the error comments and mnemonics used when building the USCIF35 binaries is the C' language header file 'xdsfast3.h' that is also shipped with the Sourceless-EPK's used to develop products using the USCIF35 software stack. The 'xdsfast3.h' header file is derived from the 'xdserror.cfg' config' file.

The [-E single / range] commands print the titles and explanations associated with a single error number, or with a range of error numbers between the specified limits. The error numbers are those are returned by the USCIF35 software stack, its parent utilities, and its child adapters.

The [-E single / range] commands are implemented in a (hopefully) intuitive manner. The vast majority of error numbers are negative. These commands interpret both positive and negative values as references to error numbers with the same absolute value. When such errors exist then the details of both will be reported. When only the negative errors exists then that alone will be reported. When any of the three trivial error numbers are selected then all three will be reported.

The [-E parse] command parses error config' files. When used with the verbose parameter this utility will provide information about syntax errors found when parsing the files

The [-E errors] command parses the 'input' error config' files, then uses the information to load a set of data structures, and then generates the 'output' error config' file using the modern format, or the 'C' language error header file.

3.12.2 The Commands

The handling error numbers [-E SINGLE] command

This command prints the titles and explanations associated with a single error number. The error numbers are those are returned by the USCIF35 software stack, its parent utilities, and its child adapters.

The [-E single=0] command will list the error title and description for the
SC_ERR_NONE value which actually indicates that no error has occurred.

The handling error numbers [-E RANGE] command

This command prints the titles and explanations associated with a range of error numbers between the specified limits. The error numbers are those are returned by the USCIF35 software stack, its parent utilities, and its child adapters.

The [-E range=1,999] command will list all of the errors generated by the USCIF35 software stack, its parent utilities, and its child adapters. The listing sweeps from more positive to more negative error numbers, and can be terminated by the user pressing any key.

The handling error numbers [-E PARSE] command

This command parses the ‘input’ error config’ files.

When used with the verbose parameter this utility will provide
information about syntax errors found when parsing the files.

The handling error numbers [-E ERRORS] command

This command parses the ‘input’ error config’ files, then uses the information to load a set of data structures, and then generates the ‘output’ error config’ file using the modern format, or the ‘C’ language error header file.

When used with the verbose parameter this utility will provide
information about syntax errors found when parsing the files.

The content of the ‘output’ error config’ file is influenced
by the ‘indent’ and ‘comment’ parameters.

3.12.3 The Parameters

The NUMBER parameter

This parameter selects the single error number whose title and explanation are to be retrieved for the [-E single] command. The error numbers all have negative values, however any positive values input via this parameter will be automatically and silently negated.

The LOWEST / HIGHEST parameters

These two parameters select the range of error numbers whose titles and explanations are to be retrieved for the[-E range] command. The error numbers all have negative values, however any positive values input via these parameters will be automatically and silently negated.

The INPUT parameter

This parameter selects the name of the error config’ file in the modern format that is
used to source error information for the for the [-E parse] and [-E errors] commands.

If this parameter is not provided then the default file ‘input.txt’ will be read.
If this parameter is used with the value ‘0’ then no file will be read.

The OUTPUT parameter

This parameter selects the name of the error config’ file or error header file that is output by the the[-E errors] command. The format of the error config’ file is the modern format. The content of the output board config’ file is influenced by the ‘indent’ and ‘comment’ parameters.

If this parameter is not provided then the default file ‘output.txt’ will be written.
If this parameter is used with the value ‘0’ then no file will be written.

The FILETYPE parameter

This parameter selects the type of output file to be generated by the [-E errors] command. The output file can be an error config' file in the modern format containing the complete error descriptions associated with every USCIF35 error number, including the each error number's title, origin and offset, plus its single-line brief error text and the full explanation. The output file can also be a 'C' language header file containing every USCIF35 error number listed as a single-line comment and a single-line mnemonic and numerical value.

This parameter has string values whose default is marked bold.

config	Output an error configuration file.
header	Output an error header file.

The INDENT parameter

This parameter controls the amount of indentation for legibility included in the output file. If the default value of 'no' is used, then a slightly smaller output file is created, because no indents will be included before each description. If the value used is 'yes', then a slightly larger file is created, because one or more indents will be included before each variable, package, family, device and sub-path description.

This parameter has boolean values whose default is marked bold.

no	Output without any indents.
yes	Output with indents for legibility.

The COMMENT parameter

This parameter controls the amount of explanatory comments included in the output file. If the default value of 'no' is used, then a smaller output file is created, because no comments will be included with the descriptions. If the value used is 'yes', then a larger file is created, because comments will be provided for each variable, package, family, device and sub-path description.

This parameter has boolean values whose default is marked bold.

no	Output without any comments.
yes	Output with lots of comments.

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

4 The Informal Options

The informal command-line options are actually a sub-set of the major command-line options. They are documented separately for two reasons:

- They are new and experimental options whose features may change. Their documentation is incomplete and may be inaccurate.
- They are old and little-used options that may be re-designed or deleted.

4.1 *The Miscellaneous Actions with [-M]*

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-M help
```

Do USCIF35 software open/acquire/release/close management tests.

```
-M manage,string=value
```

Do USCIF35 software zero-client tests.

```
-M zero,repeat=number
```

Do JTAG state command stuff with the TBC hardware.

```
-M jtag,goto=states,repeat=number,verbose=boolean
```

4.1.1 The Introduction

The basics of the miscellaneous actions

The details of the miscellaneous actions

4.1.2 The Commands

The acquire-release [-M MANAGE] command

The basic access to the scan-management component of USCIF35 can be exercised by single letter commands or a string of letters such as 'oarcnnoarct'. The letters can be entered either on the command-line or interactively in response to status messages provided by this utility.

If USCIF35 is opened and the controller acquired by use of the 'a' command then any compatible debugger or utility that is using USCIF35 will be suspended until the controller is released by use of the 'r' command. It is invalid to acquire or release when USCIF35 is closed.

The zero-client [-M ZERO] command

Perform the zero-client test on USCIF35.

These tests verify that the acquire and release function built into the USCIF35 are operating correctly.

If the 'repeat=number' parameter is used 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 JTAG state [-M JTAG] command

The controller will transition through the series of JTAG states listed by the value of the 'states' parameter. The choices are:

```
trstpin, reset, idle, pausedr, pauseir,
idle2pausedr, idle2pauseir
```

The use of 'trstpin' or 'reset' results in a transition to the 'Test-Logic-Reset' state. The nTRST signal will also be pulsed if 'trstpin' is used.

The use of 'idle', 'pausedr' or 'pauseir' results in a transition to the 'Run-Test/Idle' or 'DR-Pause' or 'IR-Pause' states.

The use of 'idle2pausedr' or 'idle2pauseir' results in a transition to the 'Run-Test/Idle' state for a single TCLK followed by a transition to the 'DR-Pause' or 'IR-Pause' states.

The use of 'pausedr' followed by 'pausedr' results in a transition to the 'Pause-DR' state and then through the inner loop of the JTAG state diagram to the 'Pause-DR' state again:

DR-Pause → DR-Exit1 → DR-Update → DR-Capture → DR-Exit2 → DR-Pause

The use of 'pauseir' followed by 'pauseir' results in a transition to the 'Pause-IR' state and then through the inner loop of the JTAG state diagram to the 'Pause-IR' state again:

IR-Pause → IR-Exit1 → IR-Update → IR-Capture → IR-Exit2 → IR-Pause

4.1.3 The Parameters

The STRING parameter

This parameter can have values that are one of six letters, or a sequence of those letters. The letters exercise commands that provide basic access to the scan-management component of USCIF35. This parameter has string values constructed from the following letters:

`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.

The REPEAT parameter

If this parameter has the value zero then the operation is repeated continuously until terminated by the user pressing any key. If this parameter is non-zero then the operation is repeated that many times, or until terminated by the user pressing any key:

This parameter has numerical and string values whose default value is marked bold.

`0`	The test is performed continuously.
`1`	The test is performed just once.
`2 - n`	The test is performed this many times.

Some choices have synonyms:

`default`	The test is performed just once.
`single`	The test is performed just once.
`forever`	The test is performed continuously.

The GOTO parameter

The value of this parameter a list of JTAG state transitions represented by one or more of the following case-insensitive items separated by '+' characters:

`trstpin`	Go to the JTAG Test-Logic-Reset state and pulse the nTRST pin.
`reset`	Go to the JTAG Test-Logic-Reset state.
`idle`	Go to the JTAG Run-Test/Idle state.
`pausedr`	Go to the JTAG Pause DR state.
`pauseir`	Go to the JTAG Pause IR state.
`idle2pausedr`	Go via Run-Test/Idle to the JTAG Pause DR state.
`idle2pauseir`	Go via Run-Test/Idle to the JTAG Pause IR state.

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

4.2 The Experimental Device Analysis with [-A]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-A help
```

Analyse a scan-path and examine a specific target.

```
-A scanpath,device=name,verbose=boolean
```

4.2.1 The Introduction

The basics of the scan-path analysis

The details of the scan-path analysis

4.2.2 The Commands

The scan-path analysis [-A SCANPATH] command

Each device on the scan-path is individually selected based on the the description provided in the board configuration file. For each device the total length of the JTAG IR instruction registers and JTAG DR bypass registers are first measured and compared with the expected value. For each device a customised interrogation routine is applied to confirm that it matches the description provided in the board configuration file.

4.2.3 The Parameters

The DEVICE parameter

The VERBOSE parameter

This parameter is used to request additional information from the command.
This parameter has boolean values whose default is marked bold.

no	Request brief information.
yes	Request verbose information.

If only the name is used, without a value, then it implies '`yes`'.

4.3 The Experimental Interactive Shell with [-I]

This option has a sub-argument that is a comma separated list of a command and zero or more parameters. The parameters consist of 'name=value' pairs that may be listed in any order. The command and the parameter names may be shortened to just 4 characters.

Print the brief hints for experts.

```
-I help
```

Launch the interactive shell.

```
-I luascript,start=filename,finish=filename
```

4.3.1 The Introduction

The basics of the interactive shell

The details of the interactive shell

4.3.2 The Commands

The interactive shell [-I LUAJTAG] command

4.3.3 The Parameters

The START parameter

The FINISH parameter

5 The Appendices

These appendices contain all secondary material related to the DBGJTAG utility:

- The Standard Emulator Signals and Cables.
- The Frequency of TCLKO and TCLKR.
- The Format of Board Config' Files.
- The Standard Board Config' Variables.
- The XDSRESET and XDSPROBE Utilities.
- The Design of the DBGJTAG Utility.

5.1 The Standard Emulator Signals and Cables

5.1.1 The Signals for 560- and 100/510-Class Emulators

The newer TI XDS560 emulators have Revision-D cables that use the 20-pin 14-signal header shown in the diagrams below and use its full set of signals. The older TI XDS560 emulators have Revision-B cables that use the 14-pin 10-signal header shown in the diagrams below and use its full set of signals. The TI XDS510 emulators have cables that use the 14-pin 10-signal header and replace the TDIS signal with an additional ground signal. The TI XDS100 emulators have cables that use the basic 14-pin 8-signal header.

```
         20-pin                      14-pin                      14-pin
       TI header                   TI header                   TI header
      (14 signals)                (10 signals)                 (8 signals)

       ╔═══════╗                  ╔═══════╗                   ╔═══════╗
  TMS o║1     2║o nTRST      TMS o║1     2║o nTRST       TMS o║1     2║o nTRST
  TDI o║3     4║i TDIS       TDI o║3     4║i TDIS        TDI o║3     4║x NC
  TVD i║5     6║x ╫╫╫╫╫      TVD i║5     6║x ╫╫╫╫╫       TVD i║5     6║x ╫╫╫╫╫
  TDO i║7     8║x GND0       TDO i║7     8║x GND0        TDO i║7     8║x GND0
TCLKR i║9    10║x GND1     TCLKR i║9    10║x GND1         NC x║9    10║x GND1
TCLKO o║11   12║x GND2     TCLKO o║11   12║x GND2      TCLKO o║11   12║x GND2
 EMU0 b║13   14║b EMU1      EMU0 i║13   14║i EMU1       EMU0 o║13   14║o EMU1
nSRST o║15   16║x GND3            ╚═══════╝                   ╚═══════╝
 EMU2 b║17   18║b EMU3
 EMU4 b║19   20║x GND4      GNDx   These are the three to nine signal grounds.
       ╚═══════╝               ╫    These are the header orientation keys.

                               i    Input signals from the target system.
                               o    Output signals to the target system.
                               b    These are bi-directional signals.
                               x    These are not signals.
```

The TMS, TDO and TDI signals are the true JTAG control and data signals.
The nTRST and nSRST signals are the active-low JTAG reset and system reset.

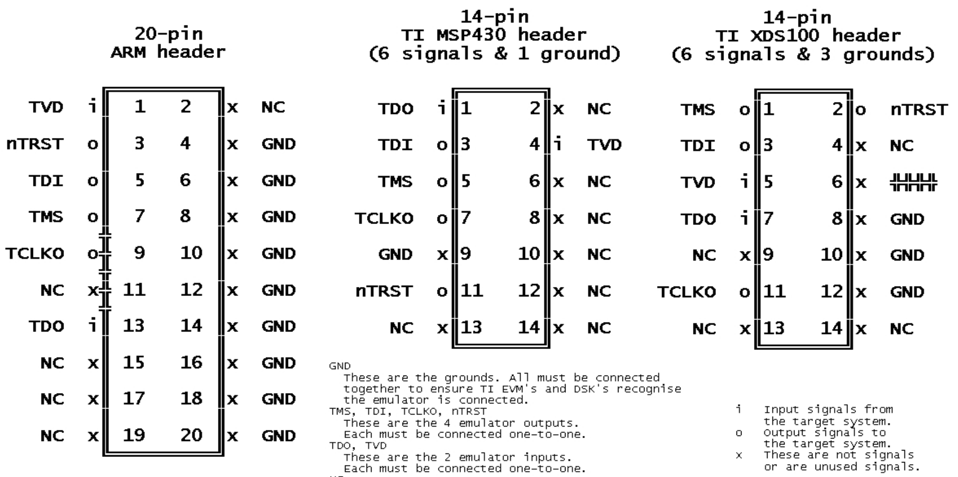
The TCLKO signal is the JTAG clock output from the emulator to the target system. It is controlled via a programmable PLL. The TCLKR signal is the JTAG clock returned from the target system to the emulator. It is the controlling clock for the emulator. It may be the TCLKO looped at the target system, or it may be a clock source in the target system.

The TDIS, TVD and TCLKR signals along with appropriate cable circuitry allow full feature emulators to accurately report errors due to emulator cable-breaks, target power-loss and dead return clocks. All the 560-class emulators and advanced 510-class emulators accurately report these errors, some less sophisticated 510-class emulators detect but do not distinguish these errors. The 100-class emulators are intended to be ultra low-cost and make no attempt to detect or distinguish these errors.

The EMU1-0 signals are the non-JTAG bi-directional and
input only signals for boot-mode support and debug support.

5.1.2 The Signals on Other Hardware

Software support is available for re-using the TI MSP430PIF hardware and the Amontec JTAGKey and JTAGKey-Tiny hardware as limited capability XDS100 emulators, without EMU pins and boot-mode support. The MSP430PIF hardware has a parallel-port interface with 14-pin MSP430 header. The Amontec hardware has USB interfaces and 20-pin ARM headers. The following diagrams illustrate how the header signals have to be re-wired for us as an XDS100 emulator.



```
GND
  These are the grounds. All must be connected
  together to ensure TI EVM's and DSK's recognise
  the emulator is connected.
TMS, TDI, TCLKO, nTRST
  These are the 4 emulator outputs.
  Each must be connected one-to-one.
TDO, TVD
  These are the 2 emulator inputs.
  Each must be connected one-to-one.
NC
  These are the unused signals.
  They must not be connected to anything.
╫╫╫╫
  This is the header orientation key.
  It must not be connected to anything.
```

```
i   Input signals from
    the target system.
o   Output signals to
    the target system.
x   These are not signals
    or are unused signals.
```

5.1.3 The Revision-D Cable for 560- and 510-Class Emulators

The Revision-D cable has a 20-pin header. It has external and internal implementations. The external implementation has a 69" long cable using micro-coax for signal transmission, with its electronics packaged in a credit-card size in-line pod. The internal implementation has the cable electronics embedded in the emulator, with a short 8" long cable. The external cable is compatible with all TI and Blackhawk 560-class emulators that support external cables, and some 3rd party 510-class emulators. The internal cable is supported by some newer Blackhawk 560-class emulators.

Cable Feature	Revision-D support	DBGJTAG command
accurate error detection	power-loss, dead-clock, cable-break	
TDIS signal	yes	
nTRST signal	yes	-Y reset
nSRST signal	yes	-Y system,signal=pulse
EMU1-0 signals	bi-directional	
nTRST boot-mode on EMU1-0	yes	-Y emupins, jtagboot=11/10/01/00
TVD boot-mode on EMU1-0	yes	-Y emupins, powerboot=11/10/01/00
TCLKO signal	yes	
TCLKR signal	yes	
adaptive TCLKO/TCLKR	yes	-F clock,program=adaptive
programmable TCLKO	488Hz to 64.0MHz	-F clock,program=automatic
measurable TCLKR	488Hz to 64.0MHz	-F inform,logfile=yes
TMS/TDO timing	rising and falling edge	-Y jtagpins,tdoedge=rise/fall
TDI timing	rising edge only	
clock loop-back	yes	-Y jtagpins,loopback=clock
data loop-back	yes	-Y jtagpins,loopback=data
total loop-back	yes - cable may be disconnected	-Y jtagpins,loopback=total

The Revision-C cable was similar to the Revision-D cable.
It used a 14-pin header and lacked the nSRST signal.
It had a limited life-span and no production use.

5.1.4 The Revision-B Cable for 560- and 510-Class Emulators

The Revision-B cable has a 14-pin header. The Revision-B cable only has an external implementation. That implementation has a 69" long cable using micro-coax for signal transmission, with its electronics packaged in a credit-card size in-line pod. The external cable is compatible with all TI and Blackhawk 560-class emulators that support external cables, and some 3rd party 510-class emulators.

Cable Feature	Revision-B support	DBGJTAG command
accurate error detection	power-loss, dead-clock, cable-break	
TDIS signal	yes	
nTRST signal	yes	-Y reset
nSRST signal	no	
EMU1-0 signals	input-only	
nTRST boot-mode on EMU1-0	no	
TVD boot-mode on EMU1-0	no	
TCLKO signal	yes	
TCLKR signal	yes	
adaptive TCLKO/TCLKR	no	
programmable TCLKO	500KHz to 64.0MHz	-F clock,program=automatic
measurable TCLKR	488Hz to 64.0MHz	-F inform,logfile=yes
TMS/TDO timing	rising and falling edge	-Y jtagpins,tdoedge=rise/fall
TDI timing	rising edge only	
clock loop-back	no	
data loop-back	no	
total loop-back	no	

The Revision-A cable was similar to the Revision-B cable.
It used a different reference frequency for the programmable TCLKO.
It had a limited life-span and no production use.

5.1.5 The 510-Class Emulator Cables

The XDS510 cable may have external and internal implementations.
There is no standard for the physical format of the cable.

Cable Feature	XDS510 support	DBGJTAG command
accurate error detection	no	
TDIS signal	no	
nTRST signal	yes	-Y reset
nSRST signal	no	
EMU1-0 signals	input-only	
nTRST boot-mode on EMU1-0	no	
TVD boot-mode on EMU1-0	no	
TCLKO signal	yes	
TCLKR signal	yes	
adaptive TCLKO/TCLKR	no	
programmable TCLKO	fixed frequency, 10.368 or 12.0 MHz	
measurable TCLKR	no	
TMS/TDO timing	rising and falling edge	-Y jtagpins,tdoedge=rise/fall
TDI timing	rising edge only	
clock loop-back	no	
data loop-back	no	
total loop-back	no	

The external cables vary between 36” and 72” and are implemented using a variety of cables technologies, SCSI, fire-wire (IEEE 1394), parallel-port (IEEE 1284) and plain old shielded ribbon cable. The external cables typically have their electronics packaged in a (large by current standards) in-line pod. The internal implementations have the cable electronics embedded in the emulator itself with a short 8” long cable.

5.1.6 The 100-Class Emulator Cables

The XDS100pp cable may have external and internal implementations.
There is no standard for the physical format of the cable.

Cable Feature	**XDS100pp support**	**DBGJTAG command**
accurate error detection	no	
TDIS signal	no	
nTRST signal	yes	-Y reset
nSRST signal	no	
EMU1-0 signals	no	
nTRST boot-mode on EMU1-0	no	
TVD boot-mode on EMU1-0	no	
TCLKO signal	yes	
TCLKR signal	no	
adaptive TCLKO/TCLKR	no	
programmable TCLKO	software generated, 50 to 200 KHz	
measurable TCLKR	no	
TMS/TDO timing	rising and falling edge	-Y jtagpins,tdoedge=rise/fall
TDI timing	rising edge only	
clock loop-back	no	
data loop-back	no	
total loop-back	no	

5.2 The Frequency of TCLKO and TCLKR

5.2.1 The Format of Frequency Values

The USCIF35 software stack supports a standard interpretation of text frequency values. Both the command-line parameters of DBGJTAG and the config' variables supported by the modern and legacy formats of board config' files conform to that standard.

In summary the text frequency values are expressed, as integers or real numbers, either un-adorned or with case-insensitive 'Hz', 'kHz' or 'MHz' suffixes. The un-adorned numbers will be interpreted as mega-hertz values. The parser explicitly checks for simple half and whole number values (0.5, 1.0, 1.5, 2.0,...) because the programmable PLL's used for TCLKO generates those values especially accurately. The design of cable software and hardware is such that other values are output with an accuracy of about 0.7%.

In detail the interpretation of values is as follows:

- The values are mnemonic strings or numbers with optional suffixes.
- The values are case-insensitive.

The interpretation of numerical values is as follows:

- The numbers may be integer or real values.
- Numbers without a suffix are mega-hertz frequencies.
- Numbers with a 'MHz' suffix are mega-hertz frequencies.
- Numbers with a 'KHz' suffix are kilo-hertz frequencies.
- Numbers with a 'Hz' suffix are hertz frequencies.

Some numerical examples are:

• `10.368`	A number without a suffix is mega-hertz frequency.
• `10.368MHz`	A number with a mega-hertz suffix.
• `500KHz`	A number with a kilo-hertz suffix.
• `500000Hz`	A number with a hertz suffix.

Some mnemonic examples are:

• `superlow`	The super-low minimum frequency - 488Hz.
• `minimum`	The minimum frequency - 500KHz.
• `legacy`	The standard XDS510 frequency - 10.368MHz.
• `exchange`	The standard XDS560 frequency - 35MHz.
• `adaptive`	The adaptive frequency - 48MHz.
• `maximum`	The maximum frequency - 64MHz.

5.2.2 The Range of Frequency Values

The range of TCLKO frequencies supported by the emulator and cable hardware, is 500Khz to 64MHz for the Revision-B and C cables, 488Hz to 64MHz for the Revision-D cables, 10.368MHz for the legacy cable for 510-class emulators, and typically 100KHz for the 100-class emulators.

The limits and resolution of the ranges of frequencies is based on the programmable PLL driving the JTAG TCLKO output and the frequency measurement logic monitoring the JTAG TCLKR input. The software and programmable PLL used by 560-class cables to generate the JTAG TCLKO output clock frequency has a resolution of 1 part in 64 or about 1.6% over most of its operating range. The software and frequency measurement hardware used by 560-class cables to measure the JTAG TCLKR input clock frequency has a resolution of 1 part in 128 or about 0.8% over most of its operating range. Thus accuracy of the combined generation and measurement allows each frequency octave to be supported in 64 discrete steps:

The Revision-B and C cables with the 14-pin header can generate TCLKO at 448 distinct frequencies over 7 octaves, from a low 500Khz up-to 64.0MHz. The octaves are:

```
500.0kHz      1.000MHz
2.000MHz      4.000MHz
8.000MHz      16.00MHz
32.00MHz      64.00MHz
```

The Revision-D cables with the 20-pin header can generate TCLKO at 1088 distinct frequencies over 17 octaves, from an ultra-low 488Hz up-to 64.0MHz. The octaves are:

```
488.0Hz       976.0Hz
1.953kHz      3.906kHz
7.812kHz      15.63kHz
31.25kHz      62.50kHz
125.0kHz      250.0kHz
500.0kHz      1.000MHz
2.000MHz      4.000MHz
8.000MHz      16.00MHz
32.00MHz      64.00MHz
```

The 560-class cables can measure TCLKR from an ultra-low 488Hz up-to 64.0MHz, and potentially operate over the same frequency range. The cable for 510-class emulators with the 14-pin header can generate TCLKO only at the fixed 10.368MHz for TCLKO. It cannot measure TCLKR at all, and to ensure reliable operation below half that frequency requires the use of the uscif.slowclk and uscif.slowfrq variables described in '5.4.8 The USCIF.SLOWCLK / SLOWFRQ variables' to explicitly tell the USCIF software the actual TCLKR frequency value.

Note that these statements apply only to frequency generation and measurement. Any JTAG IR/DR scan-tests performed at ultra-low frequencies will require large amounts of time. Any JTAG IR/DR scan-tests performed at frequencies over 32Mhz will approach the frequency limits of firstly the target systems and secondly the 560-class cable.

5.3 The Selection of TCLKO Frequency

5.3.1 The Introduction to Frequency Selection

The methods of frequency selection

The DBGJTAG utility and USCIF35 software support three methods of configuring the frequency of the JTAG output clock (TCLKO). These methods all provide the same functionality. The first method is the eleven ‘`uscif.tclk_`’ config’ variables – they are input from the board config’ file output by CC Setup for use by CC Studio's DSP and ARM emulation drivers - or input from the [-f] command of the DBGJTAG utility. The second method is the [-V apply] command of the DBGJTAG utility - it can be used to explicitly apply the same config’ variables on the command line. The third method is the [-F clock / pll / length] commands of the DBGJTAG utility - its parameters are automatically converted to memory mapped expressions of the config’ variables.

These methods require the emulator to have a JTAG output clock (TCLKO) driven by a programmable PLL and a JTAG return clock (TCLKR) monitored by a frequency measurement counter. These features are available on all 560-class products and advanced 510-class products. The 100-class emulators cannot control their TCLKO frequency and do not support a TCLKR signal.

These methods provide access to the emulator hardware and software features that generate TCLKO and measure TCLKR. They allow the user to choose any of the algorithms for frequency selection that are supported by the emulator and its cable, and provide options to modify the algorithm’s behaviour by over-riding measurements made during their execution.

The full details of the config’ variables are in the first three
sub-sections of ‘5.6 The Standard Board Config’ Variables’.

For the [-F clock] command ‘The primary frequency variables’ and default values are:

```
uscif.tclk_program                automatic
uscif.tclk_beginning              500KHz
uscif.tclk_frequency              64.0MHz
uscif.tclk_testmode               surrender
```

For the [-F pll] command ‘The secondary frequency variables’ and default values are:

```
uscif.tclk_overflow               failure
uscif.tclk_reference              default
uscif.tclk_initial                true
uscif.tclk_terminal               false
```

For the [-F length] command ‘The tertiary frequency variables’ and default values are:

```
uscif.tclk_scan_bits              16768
uscif.tclk_path_irsize            measure
uscif.tclk_path_drsize            measure
```

The target connection and frequency selection

The feature set and range of operation of the scan-path frequency selection is dependent on the connection between the JTAG controller in the emulator and the DSP and ARM target devices. In the case of embedded JTAG controllers and 100/510-class emulators it is typically a ‘dumb’ fixed frequency connection that cannot respond to any feature of the [-F] commands. In the case of 560-class emulators with an external cable/pod then the connection will respond to most or all options of the [-F] commands, with older revisions of the cable/pod supporting fewer options.

The XDS560 Revision-D cables with 20-pin headers have all features of the [-F] option including the newest: 1) use of the two adaptive clocking modes between 48.0MHz and 500KHz and 2) use of the specific and automatic modes between 64.0MHz and down past 500Khz to an ultra-low 488Hz for TCLKO generated by the cable.

The XDS560 Revision-B cables with 14-pin headers have a feature sub-set that: 1) excludes use of the the two adaptive clocking modes and 2) restricts use of the specific and automatic clocking modes to operation between 64.0MHz and 500KHz for TCLKO generated by the cable.

The XDS560 Trace cables with 60-pin headers have a feature sub-set that: 1) excludes use of the the two adaptive clocking modes and 2) includes use of the specific and automatic modes between 64.0MHz and down past 500Khz to an ultra-low 488Hz for TCLKO generated by the cable.

The emulation drivers and frequency selection

Test and debug tools that run emulation drivers on DSP and ARM target devices are reliant on variables in their board config’ files to control the frequency selection features of the emulator hardware and software. Those files also provide the scan-path description required by the emulation drivers. The current practice for frequency selection is to include explicit values only for the uscif.tclk_program and uscif.tclk_frequency config’ variables and accept the implied default values for the other variables. However any of the eleven frequency selection config’ variables may be included to tailor the operation of the emulator’s scan-path frequency selection hardware and software to the needs of the emulation driver.

The uscif.tclk_beginning and uscif.tclk_frequency config’ variables have string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix ‘5.2 The Frequency of TCLKO and TCLKR’.

Here are command-line and configuration variable selections
for adaptive clocking at its maximum synchronisation frequency:

```
  -F clock,program=adaptive,frequency=48.0mhz

  $ uscif
    tclk_program   = adaptive
    tclk_frequency = 48
  $ /
```

Here are command-line and configuration variable selections
using numeric values for the fixed ‘legacy’ frequency:

```
  -F clock,program=specific,frequency=10.368

  $ uscif
    tclk_program   = specific
    tclk_frequency = 10.368mhz
  $ /
```

Here are command-line and configuration variable selections
using mnemonics for the fixed ‘legacy’ frequency:

```
-F clock,program=specific,frequency=legacy
```

```
$ uscif
  tclk_program   = specific
  tclk_frequency = legacy
$ /
```

Here are command-line and configuration variable selections for
the automatic frequency selection over a very wide range:

```
-F clock,program=automatic,
         beginning=50khz,frequency=64mhz
```

```
$ uscif
  tclk_program   = automatic
  tclk_beginning = 50.0KHz
  tclk_frequency = 64.0MHz
$ /
```

Here are command-line and configuration variable selections for
the automatic frequency selection over the 0.5MHz to 35MHz range:

```
-F clock,program=automatic,frequency=exchange
```

```
$ uscif
  tclk_program   = automatic
  tclk_frequency = exchange
$ /
```

5.3.2 The Flow-Chart of Frequency Selection

The frequency selection algorithm is implemented in four major phases
that are influenced by the [-F clock / pll / length] commands:

Phase #1A	This checks for TCLKO programming and TCLKR measurement hardware.
Phase #1B	This checks for the TCLKR being controlled by the emulator's internal TCLKO or by the target system's own external clock.
Phase #2	This performs JTAG IR and DR path-length measurement as preparation for this command's own scan-tests.
Phase #3	This performs configuration for external TCLKR sources, or specific or adaptive TCLKO frequencies for the emulator.
Phase #4A	This performs the initial quick configuration for automatically selected TCLKO frequencies.
Phase #4B	This performs the precision configuration for automatically selected TCLKO frequencies.
Phase #4C	This performs the final safety margin configuration for automatically selected TCLKO frequencies.

The phase #1A of frequency selection

```
  START PHASE #1A
CHECK HARDWARE RESOURCES
        |
        |
        V
+---------------+ The ‘default’ choice is a variation of ‘automatic’.
|   interpret   | The ‘inverter’ choice is a variation of ‘adaptive’.
|   the TCLKO   | The ‘reference’, ‘superlow’, ‘minimum’, ‘legacy’,
|    choice     | ‘standard’, ‘exchange’ and ‘maximum’ choices
+---------------+ are variations of ‘specific’.
        |
        |
        V
  /-----------\
 /    check    \
/    the TCLKO  \ NO
\ choice is OK  /----> Failure (bad choice)
 \     ?       /
  \-----------/
    YES |  The legal clock choices are: ‘nothing’,
        |   ‘external’, ‘automatic’, ‘adaptive’ and ‘specific’.
        V
  /-----------\          /-----------\
 /    check    \        /  check the  \
/  the TCLKO is \ NO   /  TCLKO choice \ NO
\ programmable  /----> \ is ‘specific’ /----> Failure
 \     ?       /        \     ?       /     (bad choice)
  \-----------/          \-----------/
    YES |                  YES |
        |<----------------------+
        V
  /-----------\       Cannot detect external
 /    check    \      TCLKO, must rely on user
/  the TCLKR is \ NO  selecting ‘external’ choice.
\  measurable   /-------------------->|
 \     ?       /                      |
  \-----------/                       |
    YES |   Can detect external       |
        |   TCLKO and override any    |
        |   choice with ‘external’.   |
        |<----------------------------+
        V
+----------------+
|   initialise   | The log-file will be used to record
|  the log-file  | every action in this flow-chart.
+----------------+
        |
        |
        V
  /-----------\
 /    is the   \
/  TCLKO choice \ YES
\  ‘external’   /----> Skip Phase #1B, go to Phase #2,
 \     ?       /         and assume clock is external.
  \-----------/
     NO |
        |
        V
  /-----------\
 /    is the   \
/  TCLKO choice \ YES
\  ‘adaptive’   /----> Skip Phase #1B, go to Phase #2,
 \     ?       /         and assume clock is adaptive.
  \-----------/
     NO |
        |
        V
```

The phase #1B of frequency selection

```
  START PHASE #1B
DETECT TCLKR SOURCE
        |
        |
        V
+---------------+
|   select the  |
|   beginning   |
|   frequency   |
+---------------+
        |
        |
        V                                     /-----------\
+---------------+    +---------------+       /  are TCLKR  \
|    program    |    |    measure    |      /   and TCLKO   \ NO
|   the TCLKO   |--->|   the TCLKR   |---> /   frequencies  /---->|
|   frequency   |    |   frequency   |     \ within 1/16th /      |
+---------------+    +---------------+      \      ?      /       |
                                             \-----------/        |
                                                   YES            |
                        The TCLKR/TCLKO match - the |             |
+---------------+       clock may be internal or a  |             |
|  increase the |       similar external frequency. |             |
|   frequency   |<----------------------------------+             |
|    by 1/8th   |                                                 |
+---------------+                                                 |
        |                                                         |
        |                                                         |
        V                                     /-----------\       |
+---------------+    +---------------+       /  are TCLKR  \      |
|    program    |    |    measure    |      /   and TCLKO   \ NO  |
|   the TCLKO   |--->|   the TCLKR   |---> /   frequencies  /---->|
|   frequency   |    |   frequency   |     \ within 1/16th /      |
+---------------+    +---------------+      \      ?      /       |
                                             \-----------/        |
                                                   YES            |
               The TCLKR/TCLKO match - the clock is |             |
               internal and the choices ‘specific’, |             |
            ‘automatic’ and ‘adaptive’ may be used. |             |
        |<------------------------------------------+             |
        |                                                         |
        |<--------------------------------------------------------+
        |                        The TCLK is external and the clock
        |                        choice is replaced with ‘external’.
        V
```

The phase #2 of frequency selection

```
  START PHASE #2
PATH-LENGTH TEST
        |
        |
        V
+---------------+      +---------------+
|   select the  |      |    program    |
|   beginning   |----->|   the TCLKO   |
|   frequency   |      |   frequency   |
+---------------+      +---------------+
                              |
                              |
        +---------------------+
        |
        |
        V
  /-----------\                               /----------\
 / is the JTAG \      +---------------+      /    was     \
/   IR register \ NO  |    measure    |     /  measurement \ NO
\ length  known /---->|  the JTAG IR  |---> \ successful   /-----+
 \      ?      /      |  path length  |      \     ?      /      |
  \-----------/      +---------------+       \----------/       |
       YES                                      YES            V
        |                                          |          Failure
        |                                          |      (IR measurement)
        |<----------------------------------------+
        |
        V
  /-----------\                               /----------\
 / is the JTAG \      +---------------+      /    was     \
/   DR register \ NO  |    measure    |     /  measurement \ NO
\ length  known /---->|  the JTAG DR  |---> \ successful   /-----+
 \      ?      /      |  path length  |      \     ?      /      |
  \-----------/      +---------------+       \----------/       |
       YES                                      YES            V
        |                                          |          Failure
        |                                          |      (DR measurement)
        |<----------------------------------------+
        |
        |
        V
```

The phase #3 of frequency selection

```
  START PHASE #3
CONFIGURE EXTERNAL, SPECIFIC AND ADAPTIVE CLOCKS
        |
        |
        V
  /-----------\                               /----------\
 /    is the   \      +---------------+      /   was the  \
/  TCKO choice  \ YES |  perform the  |     /   scan test  \ NO
\  ‘external’   /---->|  JTAG IR/DR   |---> \  successful  /-----+
 \     ?       /      |   scan test   |      \     ?      /      |
  \-----------/       +---------------+       \----------/       |
        NO                                         YES           V
        |                                           |         Failure
        |                                           |       (scan-test)
        |                                  Success V
        |                      (configured for ‘external’ clock)
        |
        |
        V
   /----------\                             +-----------------+
  /   is the   \      +---------------+    |    configure    |
 /  TCKO choice \ YES |   fetch the   |    |   hardware for  |
/ ‘specific’ or /---->|   frequency   |--->|  the ‘specific’ |
\  ‘adaptive’  /      |     value     |    |  or ‘adaptive’  |
 \    ?       /       +---------------+    |     frequency   |
  \----------/                              +-----------------+
        NO                                          |
        |                                           |
        |                                           V
        |        +----------------------------------
        |        |
        |        |
        |        |                            /----------\
        |        |    +---------------+      /   was the  \
        |        |    |  perform the  |     /   scan test  \ NO
        |        +--->|  JTAG IR/DR   |---> \  successful  /-----+
        |             |   scan test   |      \     ?      /      |
        |             +---------------+       \----------/       |
        |                                          YES           V
        |                                           |         Failure
        |                                           |       (scan-test)
        |                                  Success V
        V                 (configured for ‘specific’/‘adaptive’ clock)
```

The phase #4A of frequency selection

```
  START PHASE #4A
CONFIGURE AUTOMATIC CLOCK
(QUICK TEST TO ESTIMATE HIGHEST GOOD FREQUENCY)
        |
        |
        V
  /------------\
 /    is the    \
/   TCKO choice  \ NO
\   'automatic'  /----> Failure (bad choice)
 \      ?       /
  \------------/
    YES |
        |
        V
+------------------+       +------------------+
|    fetch the     |       |    fetch the     |
|     minimum      |------>|     maximum      |
| frequency value  |       | frequency value  |
+------------------+       +------------------+
                                   |
+------------------+               |
|    select the    |               |
|     minimum      |<--------------+
| frequency value  |
+------------------+
        |
        |
        V
+------------------+
|     program      |
|    the TCLKO     |<-------------+
|    frequency     |              |
+------------------+              |
        |                         |
        |                         |
        V                         |
+------------------+      +------------------+
|   perform the    |      |  increment the   |
|   JTAG IR/DR     |      |    frequency     |
|    scan test     |      |      by 1/4      |
+------------------+      +------------------+
        |                         ^
        |                         |
        V                     YES |
  /------------\           /-------------\
 /    was the   \         /     was the   \
/    scan test   \ YES   /    frequency <  \
\   successful   /---->  \ (maximum + 1/4) /
 \      ?       /         \      ?        /
  \------------/           \-------------/
     NO |                     NO |
        |                        |
        V                        V Skip Phase #4B, go to Phase #4C.
  /------------\
 /    was the   \
/   frequency == \ YES
\    minimum     /--------------> Failure (scan-test)
 \      ?       /
  \------------/
     NO |
        |
        V
```

The phase #4B of frequency selection

```
  START PHASE #4B
CONFIGURE AUTOMATIC CLOCK
(SLOW PRECISION TEST FOR HIGHEST GOOD FREQUENCY)
        |
        |
        V
+-----------------+       +-----------------+
|    fetch the    |       |    calculate    |
|      upper      |------>|    the lower    |
| frequency value |       | frequency value |
+-----------------+       +-----------------+
                                  |
                                  |
                                  V
+-----------------+       +-----------------+
|    program      |       |    calculate    |
|   the TCLKO     |<------|    the middle   |<---+
|   frequency     |       | frequency value |    |
+-----------------+       +-----------------+    |
        |                                        |
        |                                        |
        V                                        |
+-----------------+                              |
|   perform the   |                              |
|   JTAG IR/DR    |                              |
|    scan test    |                              |
+-----------------+                              |
        |                                        |
        |                                        |
        V                                        |
  /------------\         +-----------------+     |
 /    was the   \        |    increase     |     |
/    scan test   \ YES   |    the lower    |     |
\   successful   /-----> |    frequency    |     |
 \       ?      /        |     value       |     |
  \------------/         +-----------------+     |
        NO                         |             |
        |                          |             |
        |                          |             |
        V                          V             |
+-----------------+         /------------\       |
|    decrease     |        /  upper/lower \      |
|    the upper    |       /   frequencies  \ NO  |
|    frequency    |-----> \     equal      /---->|
|     value       |        \      ?       /
+-----------------+         \------------/
                                 YES
                                  |
                                  |
                                  V
        +--------------------------
        |
        |
        V
```

The phase #4C of frequency selection

```
  START PHASE #4C
CONFIGURE AUTOMATIC CLOCK
(FINAL LONG TEST TO APPLY SAFETY MARGIN)
        |
        |
        V
+-----------------+      +-----------------+
|    fetch the    |      |    fetch the    |
|     minimum     |----->|     current     |
| frequency value |      | frequency value |
+-----------------+      +-----------------+
                                |
                                |
                                |
+-----------------+             |
|    reduce the   |             V
| frequency value |<-------------
|      by 1/8     |             ^
+-----------------+             |
        |                       |
        |                       |
        V                       |
+-----------------+             |
|     program     |             |
|    the TCLKO    |             |
|    frequency    |             |
+-----------------+             |
        |                       |
        |                       |
        V                       |
+-----------------+             |
|   perform the   |             |
|   JTAG IR/DR    |             |
|    scan test    |             |
+-----------------+             |
        |                       |
        |                       |
        V                      YES
  /------------\          /------------\
 /    was the   \        /   frequency  \
/    scan test   \ NO   /        >        \ NO
\   successful   /----> \    minimum     /-----+
 \      ?       /        \      ?       /      |
  \------------/          \------------/       |
       YES                                     V
        |                                  Failure (auto-range)
        |
        V
  Success
 (configured for ‘automatic’ clock)
```

5.4 The Format of Board Config' Files

The version 3.5 format (also called the modern format) is the standard text format for USCIF35. That format supports a large set of features:

- The use of config' variables for run-time software configuration.
- The description of scan-paths containing devices.
- The description of scan-paths also containing routers and sub-paths.
- The description of the characteristics of families of devices and routers.
- The definition of textual and numerical aliases for family names.
- The definition of brief and verbose error strings associated with error numbers.

The version 2.0 format (also called the legacy format) is depreciated, but still supported to ensure USCIF35 provides full compatibility when installed in CC Setup / CC Studio release 2.2 or later. That format supports a small set of features:

- The use of config' variables for run-time software configuration.
- The description of scan-paths containing devices.

Both formats have these common attributes:

- The first line in the file is a label describing the file format.
- The comments are started by a semi-colon / hash and finished by a line-ending.
- The blank and line-ending characters are mere white-space outside comments.
- The order of config' variables is not significant.
- The syntax of config' variables does not enforce any type distinctions.
- The order of devices and sub-paths in the scan-path description is significant.
- The device and family names in the scan-path
  description may have up-to sixteen characters.
- The family names in the scan-path description may be any of the textual and
  numerical values in the standard family and alias configuration files.

The board config' text files typically have *.cfg and *.dat suffixes.
The choice of suffix no longer has any significance.

The CC Setup and CC Studio software suites use files with *.dat suffixes.
A typical filename is BrdDat\ccBrd0.dat.

The USCIF35 software uses files with *.cfg suffixes.
A typical filenames are xdserror.cfg, xdsfamily.cfg and xdsalias.cfg.

5.4.1 The Two Example Files

The current builds of USCIF35 support both the legacy and modern formats.
The single restriction is that only the modern format can describe scan-paths with routers.

The file may have a label alone or with either config' variables or scan-path description. Here is an example of equivalent files using the legacy format (left) and modern format (right) with all three - a label plus config' variables and scan path description:

```
;cfg-2.0                              # config version=3.5

                                      $ uscif
; Select the XDS560.                    # Select the XDS560.
[unify_ecom_port]  0x0                  ecom_port = 0x0
[unify_ecom_drvr] 'xds560.out'          ecom_drvr = xds560.out
                                      $ /

                                      $ uscif
; Request auto-ranging TCLK.            # Request auto-ranging TCLK.
[unify_tclk_program]   'specific'       tclk_program   = specific
[unify_tclk_frequency] '35.0'           tclk_frequency = 35.0
                                      $ /

; Describe a 4 device scan-path.      # Describe a 4 device scan-path.
; 3 are named, 1 is bypassed.         # 2 are named and 1 is bypassed.
"cpu_0" TI470R1x                      @ cpu_0 family=ARM7xx
"cpu_1" TI320C54xx                    @ cpu_1 family=TMS320C54xx
"cpu_2" TI320C64xx                    @ cpu_2 family=TMS320C64xx
"cpu_3" BYPASS02E                     @ cpu_3 family=BYPASS irbits=46

; The end of the file.                # The end of the file.
                                      # /
```

5.4.2 The Character Set and White-Space

The board configure file can include any of the 8-bit ANSI/ISO printing characters and control characters with values 0x20 – 0xFF.

The board configure file cannot include any of the 8-bit ANSI/ISO control characters with values 0x00 – 0x1F, except those interpreted as white-space. This restriction applies to every token in the board configure file, including the comments and the values of variables.

The board configure file can include several 8-bit ANSI/ISO control characters (in addition to the blank character) that are interpreted as white-space. The characters are:

ASCII Name	ASCII Title	Hexadecimal Value	Keyboard escape sequence	C Language escape sequence
horizontal-tab	HT	0x09	^I	\t
line-feed	LF	0x0A	^J	\n
vertical-tab	VT	0x0B	^K	\v
form-feed	FF	0x0C	^L	\f
carriage-return	CR	0x0D	^M	\r

Note the line-feed is the standard new-line character for Unix and `C' language,
and the form-feed is the standard new-page character for Unix and `C' language.

5.4.3 The Scan-Path Ordering

The ordering of device descriptions indicates the relative position of devices in the main-path or sub-paths. The following description is deliberately expressed in words to avoid confusion due to the contradictory naming convention for JTAG data-in and data-out signals that is used by some JTAG connectors:

- The device whose data reaches the emulator first is listed first in the board configuration file. This device is a pre-amble for all others. This device has the lowest device ID.
- The device whose data reaches the emulator last is listed last in the board configuration file. This device is a post-amble for all others. This device has the highest device ID.

The ordering of sub-path descriptions does not indicate the relative position of sub-paths in the main-path or sub-paths. The relative position of sub-paths is dependent on the interpretation of sub-path addresses by the device that is the parent router of the sub-paths, that interpretation is implied by the family name of the parent router.

5.4.4 The Values of Variables

The syntax of config' variables makes no distinction between naked (un-quoted) and quoted variable values, other than that values containing white-space require quotes. The legacy files require single-quotes, the modern files require double-quotes. The variable values commonly indicate boolean, numerical, frequency and string values.

The type of the value is NOT enforced by the syntax of the file format, and not when the variable is read by the parser, but instead by USCIF35 or application software at run-time as appropriate for the usage of the variable, at which time USCIF35 has API support for interpreting the values as one or more of four data types:

•	String	A value that is used as-is.
•	Boolean	A value that is case-insensitive YES/NO or TRUE/FALSE.
•	Boolean	A value that is numeric non-zero/zero.
•	Numeric	A value with a 0x prefix followed by up-to eight hex digits.
•	Numeric	A value with no prefix followed by up-to nine decimal digits.
•	Frequency	A real number frequency with an implied MHZ suffix.
•	Frequency	A real number frequency with an explicit case-insensitive MHZ, KHZ or HZ suffix.

The string values 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 string values must be quoted if they are strings that contain spaces. The length of the string should not exceed 256, which does not include the quotes around the string. The unusually large lengths support file-names and path-names.

For boolean values, we use case-insensitive YES/NO and TRUE/FALSE, or numerical non-zero/zero as the values.

For numeric values, we can use "0x" followed by eight case-insensitive hex digits as a hexadecimal number, or nine decimal digits for a decimal number.

For frequency values a real number with no suffix represents a MHz value. The frequency values my also be followed by case-insensitive MHZ, KHZ and HZ values. The frequency values are typically used to represent frequencies from a low of 488Hz (11 octaves below 1Mhz) to a high of 64MHz (6 octaves above 1Mhz), such as the 'legacy' frequency of 10.368Mhz and the 'exchange' frequency of 35.0Mhz.

5.4.5 Translating Configure Variables

The rules for translation of configure variable names between the legacy text format and the modern text format are based on these assumptions:

- Variable names in legacy text files do not use '.' characters, but do have at least one '_' character, and none are the first or last character.
- Variable names in modern text files have exactly one '.' character, which separates the 'section' and 'variable' components of the name, and it is not the first or last character.

The translation of names from legacy text format to modern text format uses the following rules:

- Search for the first '_' in the legacy variable name and replace it with a '.' character. The preceding text will be used as the section name name. The succeeding text will be used as the variable name.
- If the text up to the first '_' was 'UNIFY_' then replace it with 'USCIF'.
- If the text up to the first '_' was 'POD_' then replace it with 'SEPK.POD_'.

The result will be that all of USCIF35's variables with the 'UNIFY' prefix will be in the 'USCIF' section of modern text files, and all of the Sourceless EPK's 'POD' variables will appear in the 'SEPK' section of modern text files. Here is an example of variable translation between the legacy format (left) and modern format (right):

```
;cfg-2.0                             # config version=3.5

                                     $ sepk
[pod_port]  0x0                        pod_port = 0x0
[pod_drvr] 'xds510.dll'                pod_drvr = xds510.dll
                                     $ /

                                     $ uscif
[unify_ecom_port]       0x0            ecom_port = 0x0
[unify_ecom_drvr]      'xds560.out'    ecom_drvr = xds560.out
[unify_tclk_program]   'specific'      tclk_program   = specific
[unify_tclk_frequency] '35.0'          tclk_frequency = 35.0
                                     $ /

                                     $ router
[router_cheat_family]  'icepick'       cheat_family   = icepick
```

```
[router_cheat_subpaths] 7            cheat_subpaths = 7
[router_cheat_customs]  0x0047       cheat_customs  = 0x0047
[router_cheat_defaults] 0x0003       cheat_defaults = 0x0003
                                   $ /

; The end of the file.             # The end of the file.
                                   # /
```

5.4.6 Translating Files with DBGJTAG

The DBGJTAG utility has a [-B] option that provides commands for parsing and translating board configuration files. The parsing capability can be used to test files for correct syntax. The translating capability can be used convert files between formats.

These commands have input, family and alias parameters for loading board configuration files. The input parameter is used to load the traditional content of these files: the device, sub-path and variable descriptions. The family and alias parameters are used to load standard family and alias descriptions. If a parameter is not applied then the default standard file will be assumed. This arrangement of multiple filenames is just a matter of convenience. A single file containing the complete set of descriptions may be loaded using only the input parameter. If a parameter is applied with the value '0' then no file will be used.

The user can obtain help from DBGJTAG on this topic:

```
    dbgjtag -B help
```

This example simply parses a board configure file and reports on its correctness or errors:

```
    dbgjtag -B parse,input=file1,output=file2,
               family=file3,alias=file4,
               verbose=boolean
```

This example inputs a board configure file and translates to the legacy text format:

```
    dbgjtag -B legacy,input=file1,output=file2,
               family=file3,alias=file4,
               indent=boolean,comment=boolean,verbose=boolean
```

This example inputs a board configure file and translates to the modern text format:

```
    dbgjtag -B modern,input=file1,output=file2,
               family=file3,alias=file4,
               indent=boolean,comment=boolean,verbose=boolean
```

The indent option is used to control indenting in the output file:

```
    indent=no/yes    ←  The default behaviour is no indenting.
```

The comment option is used to control comments in the output file:

```
    comment=no/yes   ←  The default behaviour is no comments.
```

The verbose option in is used to control extra information on standard output:

```
    verbose= no/yes  ←  The default behaviour
                         is no extra information.
    verbose          ←  Request extra information.
```

5.4.7 The Legacy Format

The legacy label and comments

The first item in the board config' file must be the label. The label indicates the format of the file and the revision number of the format. This is the label:

```
;cfg-2.0
```

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

The comments are allowed anywhere in the file. A comment is started with a semicolon and continues to the end of the line.

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

The configure variables may be suppressed without deleting their text, by simply inserting a semi-colon at the start of the line.

The legacy device descriptions

The legacy format does not support scan-paths with routers and sub-paths. The only devices that can be accessed are those on the main scan-path that proceeds from the device closest to the emulator's data input to the device closest to the emulator's data output.

The device description has a unique device name plus a family name or alias value. Those names are case insensitive and limited to 16 characters. The device name must be wrapped by double quotes and the other name must be one of the pre-defined names or values. For non-bypass devices the device description is:

```
"device_name" family_name
```

For bypass devices the pseudo-family name BYPASSxxx is used. The suffix is a 3-digit raw hexa-decimal number without a 0x label that specifies the device's IR register length:

```
"bypass_name" BYPASSxxx
```

The legacy variable descriptions

The legacy format requires the names of config' variables to concatenate their section and variable identifiers with an intervening underscore character. The variable name must be wrapped by an **[** open bracket and a **]** close bracket. Variable names are case insensitive and limited to 33 characters with at least one underscore character.

Here are some pretend variable descriptions in the legacy format that show the variety of possible values, and match the similar examples in the later description of the modern format:

```
[itsa_boolean]  YES        ; Boolean value
[itsa_hexa]     0x5533CCAA ; Hex 32-bit data value
[itsa_decimal]  20000KHz   ; Decimal 20KHz frequency
[itsa_exchange] 35.0MHz    ; Decimal MHz frequency
[itsa_510class] xds510.dll ; String adapter filename
[itsa_560class] xds560.out ; String program filename
```

5.4.8 The Modern Format

This description of the modern format references the example files in '5.5.2 The Examples'.

A summary of the modern format

- The syntax uses identifiers, numbers, meta-characters and operators.
  The rules ensure their definitions do not overlap and that they are easily distinguished.
- The family, device, sub-path, section, variable and error names are identifiers.
- The first character of an identifier must be alphabetic
  and the others alphanumeric or underscore.
- The upper/lower case of characters in an identifier is not significant.
- The length of identifiers is currently limited to 16 characters.
- The seven single characters **% ~ @ & $ ^ !** are reserved
  as meta-characters used to initiate and terminate records.
- The single character / is reserved to terminate a series of records.
  It is used as a pseudo-identifier.
- The single character # is reserved to initiate comments.
  It is used as a pseudo white-space.
- The single character **n** is used to terminate comments.
  It is used as a white-space.
- The upper/lower case of values used in variable records is significant.
  They may represent case-sensitive path-names in some file systems.
- A number is an unsigned 32-bit integer that can be decimal,
  or hexa-decimal if preceded by '0x' or '0X'.
- The three single characters = and **.** and **-** are reserved as operators.
- The two sets of paired characters **[ ]** and **" "** are reserved as operators.

The syntax allows family, device, sub-path, section, variable and error descriptions to occur in a single file or in separate files. The existing USCIF35 software releases locate the standard family, alias and error descriptions in separate files named 'xdsfamily.cfg', 'xdsalias.cfg' and 'xdserror.cfg' that are distributed with the releases. The application specific variable descriptions and scan-path descriptions of devices and sub-paths are generated by CC Setup as a single file typically named 'BrdDat\ccBrd0.dat'.

The label and comments

The first item in the board config' file must be the label. The label indicates the format of the file and the revision number of the format. Typically it occupies the first line, however the syntax is white-space friendly, so the label may be spread across up-to five lines. This is a label:

```
# config version=3.5
```

Although the label has the syntax of a comment, its grammar is similar to that of a record. The `# config` is similar to a record's 'meta-character plus title' token pair. The `version=3.5` is similar to a record's 'name=value' token triplet.

The result is that label actually has the same case-insensitivity and flexibility regarding white-space as has a true record.

The board config' file can include comments. Except for the first line, comments are ignored by software that uses the file. Comments are information provided for the engineer reading the file.

Comments begin with the **#** hash-mark meta-character. Comments may be inserted anywhere that white-space is permitted. Anything between the hash-mark and the end-of-line will be ignored.

The following pair of comments are near the beginning and end
of the example file 'A simple board config' file for OMAP1710'.

```
# Select an XDS560 emulator and choose a 10.368MHz TCLK.
# This is the end of the file.
```

The records and their parameters

The board config' file contains a series of records. Each record consists of tokens separated by white-space (space, tabs and end-of-line characters). The handling of white-space is clean – both tabs and end-of-line characters are accepted anywhere that white-space is valid.

Each record begins with a token pair, which is a meta-character and an identifier, or a meta-character and the / character as a pseudo-identifier. Each record contains zero or more token triplets which are 'name=value' parameters where the 'name' is an identifier. White-space may occur within these pairs and triplets. White-space must occur between these pairs and triplets.

The example below shows that the description of one or more variables requires two records.

```
$ uscif
  ecom_port = 0x0
  ecom_drvr = xds560.out
$ /
```

The example below shows that the description of each device requires one record.

```
@ cpu_0 family = arm7xx
@ cpu_1 family = tms320c54xx
```

The ordering of multiple records in a board config' file may or may not be significant. The ordering of device records represents the ordering of devices in the main-path or sub-path. The ordering of family, alias, sub-path, variable and error records is not significant. The overall ordering must be family records first, then alias records referencing the family descriptions, and finally device records referencing the alias and family descriptions.

The meta-character plus title pairs

The start of a record is one of the following seven meta-characters.

```
%  ~  @  &  $  ^  !
```

The choice of meta-character indicates the purpose of the record. It is followed by a title that is an identifier which uniquely names that record. They may be separated by optional white-space. The meta-characters provide independent name-spaces for the record titles.

`% family`	A record that is a family description.
`~ alias`	A record that is an alias description.
`@ device`	A record that is a device description.
`& subpath`	A record that is a sub-path description.
`$ section`	A record that is a variable section description.
`^ offset`	A record that is an error offset description.
`! title`	A record that is an error title description.

The name plus value parameters

The core of a record is zero or more parameters consisting of 'name=value' token triplets. The name may be followed by white-space. The value may be preceded by optional white-space.

The name is an identifier that uniquely names the parameter within the record. The = operator is simply a separator between the name and value. The value may be an identifier, or a naked string without white-space or a double-quoted string including white-space.

The meta-character plus slash pairs

The end of a series of one or more records may be indicated by a record that is simply a token pair consisting of a meta-character plus the / character as a pseudo-identifier. They may be separated by optional white-space. The value of the meta-character is the same as that used in the first record. The / character which is an otherwise invalid identifier is used instead of a record title.

`% /`	End a record that is a family description.
`~ /`	End a record that is an alias description.
`@ /`	Not used - The end of a device description is marked by the start of any following record.
`& /`	End a record that is a sub-path description.
`$ /`	End a record that is a variable section description.
`^ /`	End a record that is an error offset description.
`! /`	End a record that is an error title description.

The family descriptions

A family description begins with a **%** meta-character followed by optional white-space and the identifier for a family name. The family names are case insensitive and limited to 16 characters. The contents of a family description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the parameters in the list is not significant. A family description ends with a **%** meta-character and a / character.

The lines below show how a family description includes the list of family characteristics.

```
% family_name
    name = value
    name = value
% /
```

The currently defined parameter names that are used in family descriptions are:

`drbits`	The length of the JTAG Bypass Register of the device.
`irbits`	The length of the JTAG Instruction Register of the device.
`onceend`	The length of the end delay parameter after a single scan.
`manyend`	The length of the end delay parameter after a repeat scan.
`fixnumber`	The number of fixed memory spaces needed for the device.
`fixedsize`	The size of fixed memory spaces allocated for the device.
`scanpaths`	The actual number of IR/DR scan-paths inside the device.
`routerflags`	The flags for router properties of the device.
`routerpaths`	The maximum number of router sub-paths of the device.

The parameters in family descriptions may have both textual and numerical values. The current the parameters only use numerical values. The syntax permits future use of parameters that have string or boolean or frequency values.

The family descriptions are used to describe device families that consist of single processors, such as the TMS320C55xx and TMS320C64xx families, and the ARM7xx and ARM9xx families. The purpose of the family name that is the title of the family description is to allow device descriptions to reference family characteristics with 'family=string' parameters. The family description lists the family name and the characteristics of the family that are required for the operation of the USCIF35 software running on the TBC and Nano-TBC scan-controller hardware. Some of the characteristics are familiar to engineers developing the emulation drivers that use USCIF35 software, and other characteristics are familiar to engineers developing scan-controller hardware.

The family descriptions example

This example is a sub-set of the standard descriptions from the 'xdsfamily.cfg' file used by the current USCIF35 release:

```
# config version=3.5
% arm9xx
    drbits=1 irbits=4
    onceend=0 manyend=0
    fixnumber=8 fixedsize=0x1400
    scanpaths=16 unused=0
    routerflags=0 routerpaths=0
% /
% arm11xx
    drbits=1 irbits=5
    onceend=0 manyend=0
    fixnumber=8 fixedsize=0x1400
    scanpaths=16 unused=0
    routerflags=0 routerpaths=0
% /
% icepick_c
    drbits=1 irbits=6
    onceend=0 manyend=0
    fixnumber=0 fixedsize=0
    scanpaths=4 unused=0
    routerflags=0x0151 routerpaths=32
% /
% tms320c64plus
    drbits=1 irbits=38
    onceend=0 manyend=16
    fixnumber=16 fixedsize=0x1400
    scanpaths=16 unused=0
    routerflags=0 routerpaths=0
% /
% tms320c621x
    drbits=1 irbits=46
    onceend=0 manyend=8
    fixnumber=16 fixedsize=0x1400
    scanpaths=16 unused=0
    routerflags=0 routerpaths=0
% /
# This is the end of the file.
```

The alias descriptions

An alias description begins with a ~ meta-character followed by optional white-space and the identifier for a family name. The family names are case insensitive and limited to 16 characters. The contents of an alias description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the parameters in the list is not significant. An alias description ends with a ~ meta-character and a / character.

The lines below show how an alias description includes the list of alias characteristics.

```
~ alias_name
    name = value
    name = value
~ /
```

The currently defined parameter names that are used in alias descriptions are:

`type`	The type of the alias – currently can only be 'family'.
`modern`	The default alias value for use with modern config' files.
`legacy`	The default alias value for use with legacy config' files.
`alias`	The 0, 1 or more alternative aliases may be textual or numerical values.

The parameters in alias descriptions may have both textual and numerical values.
The current parameters do use both types of values.

The alias descriptions are used to describe alternative names for families. The purpose of the family name that is the title of the alias description is to allow device descriptions to reference its alias values instead of the family name with 'family=string' parameters.

The alias description lists the specific default aliases used by the USCIF35 software when generating legacy and modern config' files as the 'modern=string' or 'legacy=number' parameters. If no default alias is defined for those circumstances then the value "undefined" indicates that the name provided as the title of the description will be used.

The alias description also lists zero or more 'alias=string' or 'alias=number' values that are accepted as equivalent to the family name when parsing legacy and modern board config' files. Those alternative aliases may be textual values such as the project names used during emulation driver development, or the otherwise obsolete names required for backwards compatibility with older software releases. Those alternative aliases may be also be numerical values such as the 'processor ID' used by CC Setup and CC Studio.

This means that the 'family=value' parameter in device descriptions can use surprising values. The example below is a valid scan-path description of three similar tms320c64plus devices:

```
@ cpu_0 family=tms320c64plus
@ cpu_1 family=joule
@ cpu_2 family=0x50019348
```

The alias descriptions example

This example is a sub-set of the standard descriptions from the 'xdsalias.cfg' file used by the current USCIF35 release:

```
# config version=3.5
~ arm9xx
    type   = family
    modern = undefined
    legacy = tiarm9xx
    alias  = tiarm925
    alias  = ti470r2x
    alias  = tms470r2
    alias  = tms470r9
    alias  = tms470r925
    alias  = 0x75800930
    alias  = 0x75800bf8
~ /
~ arm11xx
    type   = family
    modern = undefined
    legacy = tiarm11xx
    alias  = ti470r3
    alias  = tms470r3
    alias  = tms470r11
    alias  = tms470r1136
    alias  = 0x75800D20
~ /
~ icepick_c
    type   = family
    modern = undefined
    legacy = undefined
    alias  = icepickc
    alias  = 0x3c000805
~ /
~ tms320c64plus
```

```
        type   = family
        modern = undefined
        legacy = undefined
        alias  = joule
        alias  = 0x50019348
        alias  = 0x500191b8
    ~ /
    ~ tms320c621x
        type   = family
        modern = undefined
        legacy = ti320c621x
        alias  = ti320c61x
        alias  = tms320c6211
        alias  = tms320c621
        alias  = 0x50018b28
        alias  = 0x5001bf28
    ~ /
    # This is the end of the file.
```

The device descriptions

A device description begins with a @ meta-character followed by optional white-space and the identifier for a device name. The device names are case insensitive and limited to 16 characters. The contents of a device description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the parameters in the list is not significant. A device description ends with the meta-character and identifier that start any following record.

The device description is used to describe a single device on the main scan-path or a sub-path. Those devices may be ordinary devices or may be scan-path routers that have sub-paths associated with them. The description lists the device name and the characteristics of that device that that contribute to the connectivity of the main scan-path, routers and sub-paths that proceed from the device closest to the emulator's data input to the device closest to the emulator's data output. The characteristics are represented as the parameters consisting of 'name=value' triplets.

The lines below show how a device description includes the list of device characteristics.

```
@ device_name
    name = value
    name = value
```

The standard parameter names that are used in device descriptions are:

family	The family name or a textual or numerical alias for the family name. This family name must match a family description. This aliases must match the alias description for the family.
drbits	The length of the JTAG Bypass Register of the device. This overrides the value from the family description.
irbits	The length of the JTAG Instruction Register of the device. This overrides the value from the family description.
subpaths	The number of sub-paths controlled by the device. A non-zero value implies the device is a router.

The extra parameter names that are optionally used in device descriptions are:

`parent`	The explicit name of the parent sub-path of this device. This is normally not used as it is implied by the hierarchy of records in board config' files using the modern format.
`partner`	An arbitrary device name that is used by some clients.
`address`	An arbitrary 32-bit integer that is used by some clients.
`identify`	An arbitrary 32-bit integer that is used by some clients.

The parameters in device descriptions may have both textual and numerical values.

The device descriptions are used to describe main scan-paths and sub-paths that consist of one or more ordinary devices and routers, such as the TMS320C55xx and TMS320C64xx devices, the ARM7xx and ARM9xx devices, and the ICEPICK_A, ICEPICK_B and ICEPICK_C routers. The author of the scan-path description specifies the device name and later uses it together with the entire scan-path description in the board config' file to operate the emulation driver associated with that device or the utility accessing that device via the USCIF35 software.

The ordering of devices on their parent scan-path is explicitly indicated by the relative position of device records on a main scan-path or sub-path.

The board config' file examples in '5.5.2 The Examples' show a simple scan-path with three different devices (including a bypass device) in A simple board config' file for OMAP1710, plus a simple scan-path with six identical devices in A simple board config' file for Janus.

The sub-path descriptions

A sub-path description begins with a **&** meta-character followed by optional white-space and the identifier for a sub-path name. The sub-path names are case insensitive and limited to 16 characters. The contents of a sub-path description is a list of zero or more parameters consisting of 'name=value' triplets . The ordering of the parameters in the list is not significant. A sub-path description ends with the start of a following sub-path description, or ends with a **&** meta-character and a / character.

The lines below show how a sub-path description includes both its list of sub-path characteristics and the list of device descriptions for the devices located on the sub-path.

```
& subpath_name
    address = number
    default = boolean
    custom  = boolean
      @  device_name ...
      ...
      @  device_name ...
& /
```

The currently defined parameter names that are used in sub-path descriptions are:

`address`	The address of the sub-path – the numerical value used by USCIF35 software to select and de-select the sub-path via the parent router.

`default`	The default selection indicates if the sub-path is selected before start-up of the USCIF35 software and to be restored to that state when at the shut-down of the USCIF35 software.
`custom`	The custom selection indicates if the sub-path is to be continuously selected during the operation of the USCIF35 software.

The extra parameter names that are optionally used in some sub-path descriptions are:

`parent`	The explicit name of the parent router of this sub-path. This is normally not used as it is implied by the hierarchy of records in board config' files using the modern format.
`devices`	The explicit number of the child devices on this sub-path. This is normally not used as it is implied by the hierarchy of records in board config' files using the modern format.
`pseudo`	An arbitrary boolean value that is used by some clients.

The parameters in sub-path descriptions may have both textual and numerical values.

The sub-path descriptions are used to describe sub-paths that consist of one or more ordinary devices and routers, such as the TMS320C55xx and TMS320C64xx devices, the ARM7xx and ARM9xx devices, and the ICEPICK_A, ICEPICK_B and ICEPICK_C routers. The author of the scan-path description specifies the sub-path name and later uses it together with the entire scan-path description in the board config' file to operate the emulation driver associated with devices on that sub-path or the utility accessing that sub-path via the USCIF35 software.

The ordering of sub-paths on their parent scan-path is not indicated by the relative position of sub-path records. It is indirectly indicated by the values of the address parameters and the family name of the parent router and depends on the design of that router.

The board config' file examples in '5.5.2 The Examples' show four simple sub-paths each with one different devices (including a bypass device) in A simple board config' file for OMAP2420, plus six simple sub-paths each with one similar device in A simple board config' file for Tomahawk.

The variable descriptions

A variable description begins with a **$** meta-character followed by optional white-space and the identifier for a section name. The section names are case insensitive and limited to 16 characters. The contents of a variable description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the parameters in the list is not significant. A variable description ends with a **$** meta-character and a / character.

The identifier for a parameter name is case insensitive and limited to 16 characters. The parameter value may or may-not be enclosed in the **"** double-quote characters. If the parameter value contains white-space then it must be double-quoted.

Throughout this document the section name and parameter name are together considered a config' variable name when the two are concatenated with an intervening **.** character.

Here is a variable description where the parameter name
is followed by its naked value without white-space.

```
$ section_name
    variable_name = naked-value
$ /
```

Here is a variable description where the parameter name is followed by its quoted value that may or may not contain white-space.

```
$ section_name
    variable_name = "quoted value"
$ /
```

The parameter names used in variable descriptions are somewhat different from the parameter names for other (family, alias, device, sub-path and error title) descriptions in that there is no permanent definition of the names and their purposes. They vary between USCIF35 software releases, generally increasing in number and flexibility across successive releases, and are used to customise the behaviour of the USCIF35 software. The currently implemented standard config' variables are described in '5.6 The Standard Board Config' Variables'.

Here are some pretend variable descriptions in the modern format that show the variety of possible values, and match the similar examples in the earlier description of the legacy format:

```
$ itsa
    boolean  = YES        # Boolean value
    hexa     = 0x5533CCAA # Hex 32-bit data value
    decimal  = 20000KHz   # Decimal 20KHz frequency
    exchange = 35.0MHz    # Decimal MHz frequency
    510class = xds510.dll # String adapter filename
    560class = xds560.out # String program filename
$ /
```

The board config' file examples in '5.5.2 The Examples' show eight board config' files with config' variables. They use four config' variables with '`uscif`' section names and two config' variables with '`sepk`' section names.

The error title description

An error title description begins with a **!** meta-character followed by optional white-space and the identifier for an error title. The error titles are case insensitive and limited to 32 characters. The contents of an error title description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the items in the list is not significant. An error title description ends with a **!** meta-character and a / character.

The lines below show how an error title description includes the details of the error characteristics.

```
! error_title
    name = value
    name = value
! /
```

The currently defined parameter names that are used in error title descriptions are:

`origin`	The type of the error – currently can only be 'uscif'.
`offset`	The block of error mnemonics with which this error is associated. This string matches a parameter in an error offset description that provides the numerical value of the block.

`number`	The relative value of this error compared to the offset value. This number is added to the offset value to obtain the error number.
`alias`	An alias for this error title description. It is used to append mnemonics for aliases to the 'C' language header file
`brief`	The brief one-line of text explaining this specific error. It is used to generate a comment for the error mnemonic in the 'C' language header file. If it has the value 'undefined' then the first line of the multi-line explanation will be used instead.
`explain`	The thorough many-lines of text explaining this specific error. It is used to generate the full explanation of the error.

The parameters in error title descriptions may have both textual and numerical values. The current parameters do use both types of values. The values of the brief and explain parameters are always quoted because they contain white-space. The many-lines of text for the value of the explain parameter are individually quoted.

The error title descriptions are used to describe the 'C' language mnemonics and the explanatory text associated with error numbers for the USCIF35 software. The purpose of the title of the error title description is to represent the 'C' language mnemonic associated with the error number that is used to generate the error header file. The offset and number are numerical values that are summed to generate the actual error number. The offset is a text string whose equivalent numerical values are described in a separate error offset description. The alias is an alternative mnemonic with the same value and interpretation – typically is represents an obsolete mnemonic no longer used by the USCIF35 software but still present in emulation drivers or utilities.

The error offset description

An error offset description begins with a ^ meta-character followed by optional white-space and the identifier for an error offset. The error titles are case insensitive and limited to 16 characters. The contents of an error title description is a list of zero or more parameters consisting of 'name=value' triplets. The ordering of the items in the list is not significant. An error title description ends with a ^ meta-character and a / character.

The lines below show how an error title description includes the details of the error characteristics.

```
^ error_offset
    name = value
    name = value
^ /
```

The currently defined parameter name that is used in error title descriptions is:

`origin`	The type of the error – currently can only be 'uscif'.

The parameters in error offset descriptions may have both textual and numerical values. The current parameters do use both types of values.

The parameter names (other than 'origin') used in error offset descriptions are somewhat different from the parameter names for other (family, alias, device, sub-path and error title) descriptions in that they are not fixed. They are the acceptable values of the 'offset' parameters from the error title descriptions. The numerical values of the parameters are the base values of blocks of error numbers represented by the error descriptions using the corresponding 'offset' parameter value.

The error descriptions example

This example is a sub-set of the standard descriptions from the 'xdserror.cfg' file used by the current USCIF35 release:

```
# config version=3.5
^ offset
  origin     = uscif
  offset_ctl = -180
^ /
! sc_err_ctl_no_trg_power
  origin  = uscif
  offset  = offset_ctl
  number  = 0
  alias   = undefined
  brief   =
    "The target system has lost its power supply."
  explain =
    "The controller has detected a target power loss."
    "The user must turn-on or connect the power supply."
! /
! sc_err_ctl_no_trg_clock
  origin  = uscif
  offset  = offset_ctl
  number  = -1
  alias   = undefined
  brief   =
    "The target system has lost its JTAG clock."
  explain =
    "The controller has detected a dead JTAG clock."
    "The user must turn-on or connect the JTAG clock."
! /
! sc_err_ctl_cbl_break_near
  origin  = uscif
  offset  = offset_ctl
  number  = -2
  alias   = undefined
  brief   =
    "The cable is disconnected near-to the controller."
  explain =
    "The controller has detected a cable break near-to it."
    "The user must connect the cable/pod to the controller."
! /
! sc_err_ctl_cbl_break_far
  origin  = uscif
  offset  = offset_ctl
  number  = -3
  alias   = undefined
  brief   =
    "The cable is disconnected far-from the controller."
  explain =
    "The controller has detected a cable break far-from it."
    "The user must connect the cable/pod to the target."
! /
# This is the end of the file.
```

5.5 The Examples of Board Config' Files

5.5.1 The Introduction

The purpose of the examples

These examples are included for several purposes:

- To provide stand-alone examples with commentary about their content and comparison that is intended to be read by the user of this guide.
- To be referenced by the usage examples for DBGJTAG in both '1.3.1 Examples Without an Emulator' and '1.3.2 Examples Requiring an Emulator'.
- To be referenced by the explanation of the board config' file's format in '5.4.8 The Modern Format'.

If matching target systems are available, then these examples can be selected with the [-f] option and used to:

- Select the emulator via the config' variable descriptions.
- Apply scan tests to routers and devices via the[-S] and [-R] options.
- Apply frequency tests to routers and devices via the [-F] and [-G] options.

The default behaviour these examples are selected with the [-f] option can be modified:

- The selection of the emulator and its port address by config' variables can be over-ridden by the [-d] and [-p] options.
- The generation of the TCLKO frequency by config' variables can be over-ridden by the [-F clock,program=string,frequency=string] option.
- The detection of the TCLKR frequency can be over-ridden by [-V apply,slowclk=yes,slowfrq=hertz] where 'hertz' is the numerical frequency value.

These examples can also be used by some commands without the [-f] option:

- Apply white-box tests to software components within the USCIF35 software and DBGJTAG via the [-B] option.

The types of the examples

All of the examples are real – they are used by engineers working on the development of emulation drivers and DSP/ARM devices. Both simple and complex examples are provided for the DaVinci and Sedna-Neptune3G devices.

The simple examples have scan-path descriptions that provide information necessary to running a debugger or validation tests with device-specific emulation drivers. Their content is similar to the board config' files created by CC Setup and used by CC Studio. The simple examples show scan-path descriptions both without routers (only a main scan-path and no sub-paths) and with routers (a main-scan-path, one or more routers and sub-paths).

The complex examples have scan-path descriptions that provide much additional information to support the initial development of both emulation drivers and the DSP/ARM devices themselves. The complex examples all involve routers and sub-paths.

The examples are:

- A simple board config' file for OMAP1610/1710
- A simple board config' file for OMAP2420
- A simple board config' file for Janus
- A simple board config' file for Tomahawk
- A simple board config' file for DaVinci
- A complex board config' file for DaVinci
- A simple board config' file for Sedna-Neptune3G
- A complex board config' file for Sedna-Neptune3G

These examples use the modern format defined in '5.4.8 The Modern Format'. The config' variables are described using variable records from 'The variable descriptions'. The simple and complex scan-paths are described using the device and sub-path records from 'The device descriptions' and the 'The sub-path descriptions'.

5.5.2 The Examples

A simple board config' file for OMAP1710

This is an example file named named 'omap1710.cfg'. It has three devices located directly on the main scan-path. There is one DSP and one ARM whose families are known to the USCIF35 software. The device names are simply 'c55' and 'arm9'. The family of the other device is described as 'bypass' because it is unfamiliar to the software.

This file should be compared with 'A simple board config' file for OMAP2420' which also has DSP, ARM and bypass devices, but located on separate sub-paths of a router.

```
# config version=3.5
# Select a TI PCI-bus XDS560 emulator.
# Choose a fixed 10.368MHz TCLK.
$ uscif
  ecom_drvr=xds560.out
  ecom_port=0
  tclk_program=specific
  tclk_frequency=legacy
$ /
# The OMAP1610/1710 device has a DSP and an ARM.
@ c55 family=tms320c55xx
@ arm9 family=arm9xx
@ bypass family=bypass irbits=8
# This is the end of the file.
# /
```

A simple board config' file for OMAP2420

This is an example file named 'omap2420.cfg'. It has four devices located on four separate sub-paths of a router. The router is the only device located on the main scan-path. The devices are a DSP and two ARM's whose families are known to the USCIF35 software. The device names are simply 'c55' and 'arm7' and 'arm11'. The family of the other device is described as 'bypass' because it is unfamiliar to the software.

This file should be compared with 'A simple board config' file for OMAP1710' which also has DSP, ARM and bypass devices, but located directly on the main scan-path.

```
# config version=3.5
# Select a TI ISA-bus XDS510 emulator.
$ sepk
  pod_drvr=xds510.dll
  pod_port=0
$ /
# The IcePick-B behaviour is customised for reliable operation.
$ router
  skip_polling=not
  idle_delay=10
$ /
# The OMAP2420 device has an IcePick-B
# router plus one DSP and two ARM's.
@ icepick family=icepick_b subpaths=4
  & sub_3 address=6 default=no custom=no
    @ arm7 family=arm7xx
  & sub_2 address=2 default=no custom=no
    @ tms320c55xx family=tms320c55xx
  & sub_1 address=1 default=yes custom=no
    @ etb family=bypass irbits=4
  & sub_0 address=0 default=yes custom=no
    @ arm11 family=arm11xx
  & /
# This is the end of the file.
# /
```

A simple board config' file for Janus

This is an example file named 'janus.cfg'. It has six devices located directly on the main scan-path. The six devices are identical DSP's whose family is known to the USCIF35 software. The device names are simply indexed as 'dsp_0' to 'dsp_5'.

This file should be compared with another example that also has six identical devices: 'A simple board config' file for Tomahawk'. That example differs in that each device is located on a separate sub-path of a router.

```
# config version=3.5
# Select a Blackhawk USB 560-class bus-power emulator.
# Choose a fixed 10.368MHz TCLK.
$ uscif
    ecom_drvr=bh560ubp.out
    ecom_port=0
    tclk_program=specific
    tclk_frequency=legacy
$ /
# The Janus device has six DSP's.
@ dsp_5 family=tms320c55xx
@ dsp_4 family=tms320c55xx
@ dsp_3 family=tms320c55xx
@ dsp_2 family=tms320c55xx
@ dsp_1 family=tms320c55xx
@ dsp_0 family=tms320c55xx
# This is the end of the file.
# /
```

A simple board config' file for Tomahawk

This is an example file named 'tomahawk.cfg'. It has six devices located on six separate sub-paths of a router. The router is the only device located on the main scan-path. The six devices are identical DSP's whose family is known to the USCIF35 software. The device names are simply indexed as 'dsp_0' to 'dsp_5'.

This file should be compared with another example that also has six identical devices: 'A simple board config' file for Janus'. That example differs in that every device is located directly on the main scan-path.

```
# config version=3.5
# Select a Blackhawk USB 510-class bus-power emulator.
$ sepk
    pod_drvr=bhjtag3.dll
    pod_port=0
$ /
# The Tomahawk device has one IcePick-C router plus six DSP's.
@ icepick family=icepick_c subpaths=6
  & sub_5 address=21 default=no
    @ dsp_5 family=tms320c64plus
  & sub_4 address=20 default=no
    @ dsp_4 family=tms320c64plus
  & sub_3 address=19 default=no
    @ dsp_3 family=tms320c64plus
  & sub_2 address=18 default=no
    @ dsp_2 family=tms320c64plus
  & sub_1 address=17 default=no
    @ dsp_1 family=tms320c64plus
  & sub_0 address=16 default=no
    @ dsp_0 family=tms320c64plus
  & /
# This is the end of the file.
# /
```

A simple board config' file for DaVinci

This is an example file named 'davinci_simple.cfg'. It has two devices located on two separate sub-paths of a router. The router is the only device located on the main scan-path. The devices are a DSP and an ARM with families known to the USCIF35 software. The device names are simply 'c64plus' and 'arm9'.

This file should be compared with the complex example of the exact same DaVinci device: 'A complex board config' file for DaVinci'. That example differs in that it also describes eight additional emulation and test modules located on eight other sub-paths.

```
# config version=3.5
# Select a TI PCI-bus XDS560 emulator.
# Choose a fixed 10.368MHz TCLK.
$ uscif
  ecom_drvr=xds560.out
  ecom_port=0
  tclk_program=specific
  tclk_frequency=legacy
$ /
# The DaVinci device has an IcePick-C
# router plus one DSP and one ARM.
@ router family=icepick_c subpaths=2
  & dsp address=18 default=no custom=no
    @ c64 family=tms320c64plus
  & arm address=16 default=no custom=no
    @ arm9 family=arm9xx
& /
# This is the end of the file.
# /
```

A complex board config' file for DaVinci

This is a example file named 'davinci_complex.cfg'. It has ten devices located on ten separate sub-paths of a router. The router is the only device located on the main scan-path. Two of the devices are a DSP and an ARM with families known to the USCIF35 software. The device names are simply 'c64plus' and 'arm9'. The other devices are emulation and test modules required for software development and silicon test. Their families are described as 'bypass' because they are unfamiliar to the software.

This file should be compared with the simple example of the exact same DaVinci device: 'A simple board config' file for DaVinci'. That example differs in that it only describes the DSP and ARM devices located on two separate sub-paths of the router.

```
# config version=3.5
# Select a TI ISA-bus XDS510 emulator.
$ sepk
  pod_drvr=xds510.dll
  pod_port=0
$ /
# The DaVinci device has an IcePick-C
# router plus one DSP and one ARM.
@ router family=icepick_c subpaths=10
  # These are the debug subpaths.
  & sub_9 address=19 default=no
    @ fake family=bypass irbits=8
  & sub_8 address=18 default=no
    @ c64 family=tms320c64plus
  & sub_7 address=17 default=no
    @ etb family=bypass irbits=4
  & sub_6 address=16 default=no
    @ arm9 family=arm9xx
  # These are the test subpaths.
  & sub_5 address=5 default=no
    @ plldft family=bypass irbits=8
  & sub_4 address=4 default=no
    @ pllcatscan family=bypass irbits=7
  & sub_3 address=3 default=no
    @ secmodule family=bypass irbits=1
  & sub_2 address=2 default=no
    @ gemp1500 family=bypass irbits=18
  & sub_1 address=1 default=no
    @ dftp1500 family=bypass irbits=50
  & sub_0 address=0 default=no
    @ efuse family=bypass irbits=4
  & /
# This is the end of the file.
# /
```

A simple board config' file for Sedna-Neptune3G

This is an example file named 'planets_simple.cfg'. It has five devices located on five separate sub-paths of two routers. The routers each support two and three devices. The routers are connected hierarchically. The lower router is located on an extra sub-path of the higher router which is the only device located on the main scan-path. Two of the devices are DSP's and three are ARM's with families known to the USCIF35 software. The device names are descriptive: 'c64', 'c55', 'arm11', 'arm9' and 'arm7'.

This file should be compared with the complex example of the exact same Sedna-Neptune3G device: 'A complex board config' file for Sedna-Neptune3G'. That example differs in that it also describes seven additional emulation and test modules located on seven other sub-paths.

```
# config version=3.5
# Select a Blackhawk USB 560-class bus-power emulator.
# Choose a fixed 10.368MHz TCLK.
$ uscif
    ecom_drvr=bh560ubp.out
    ecom_port=0
    tclk_program=specific
    tclk_frequency=legacy
$ /
# The level one Sedna chip has an IcePick-B router.
# The level two Neptune-3G chip also has an IcePick-B router.
# The entire device has two DSP's and three ARM's.
@ level_1 family=icepick_b subpaths=3
  & sub_10 address=0 default=yes
    @ arm11 family=arm11xx
  & sub_16 address=6 default=no
    @ c64 family=tms320c64plus
  & sub_11 address=1 default=no
    @ level_2 family=icepick_b subpaths=3
      & sub_20 address=0 default=yes
        @ arm9 family=arm9xx
      & sub_21 address=1 default=no
        @ c55 family=tms320c55xx
      & sub_24 address=4 default=no
        @ arm7 family=arm7xx
      & /
    & /
# This is the end of the file.
# /
```

A complex board config' file for Sedna-Neptune3G

This is an example file named 'planets_complex.cfg'. It has twelve devices located on twelve separate sub-paths of two routers. Each router supports six devices. The routers are connected hierarchically. The lower router is located on an sub-path of the higher router which is the only device located on the main scan-path. Two of the devices are DSP's and three are ARM's with families known to the USCIF35 software. The device names are descriptive: 'c64', 'c55', 'arm11', 'arm9' and 'arm7'. The other devices are emulation and test modules for software development and silicon test. Their families are unfamiliar to the software and described as 'bypass'.

This file should be compared with the simple example of the exact same Sedna-Neptune3G device: 'A simple board config' file for Sedna-Neptune3G'. That example differs in that it only describes the DSP and ARM devices located on five separate sub-paths of the router.

```
# config version=3.5
# Select a Blackhawk USB 510-class bus-power emulator.
$ sepk
  pod_drvr=bhjtag3.dll
  pod_port=0
$ /
# The level one Sedna chip has an IcePick-B router.
# The level two Neptune-3G chip also has an IcePick-B router.
# The entire device has two DSP's and three ARM's.
@ level_1 family=icepick_b subpaths=7
  & sub_10 address=0 default=yes # ARM1136, ICEcrusher11, ETM11, PSA.
    @ arm11 family=arm11xx
  & sub_12 address=2 default=yes
    @ etb family=bypass irbits=4
  & sub_13 address=3 default=no  # Prod' test chip-level TAP.
    @ sedna_chiptap family=bypass irbits=8
  & sub_14 address=4 default=no  # Device config' e-Fuse.
    @ sedna_efuse family=bypass irbits=4
  & sub_15 address=5 default=no
    @ xti family=bypass irbits=3
  & sub_16 address=6 default=no  # The IVA2 device.
    @ c64 family=tms320c64plus
  & sub_11 address=1 default=no
    @ level_2 family=icepick_b subpaths=6
      & sub_20 address=0 default=yes # ARM9, E2ICE, ETM9.
        @ arm9 family=arm9xx
      & sub_21 address=1 default=no  # The UMA2.6 device.
        @ c55 family=tms320c55xx
      & sub_22 address=2 default=no  # Device config' e-Fuse.
        @ n3g_efuse family=bypass irbits=4
      & sub_23 address=3 default=no  # Prod' test chip-level TAP.
        @ n3g_chiptap family=bypass irbits=8
      & sub_24 address=4 default=no
        @ arm7 family=arm7xx
      & sub_25 address=5 default=no  # No JTAG BYPASS instruction!
        @ nec family=bypass irbits=8
      & /
   & /
# This is the end of the file.
# /
```

5.6 *The Standard Board Config' Variables*

The variables documented here are those supported by USCIF35 and its utilities for 510- and 560-class emulators. The default values of these variables may depend on the emulator being 510- or 560-class, the version of the scan-controller, and the revision of the emulator cable.

5.6.1 The Primary Frequency Variables

These variables select the frequency of the JTAG clock for XDS560 products. They are also used to implement the parameters of the `[-F clock]` command, whose detailed description in this manual is thus also a description of these variables.

The USCIF.TCLK_PROGRAM variable

This config' variable is equivalent to this command-line parameter:

```
[-F clock,program=value]
```

This variable selects the method used by XDS560 products to choose the JTAG clock frequency. All of the tests used by those algorithms are restricted to the JTAG IR instruction scan-path and JTAG DR bypass scan-path. This variable has string values whose defaults are marked bold.

`nothing`	Don't touch the PLL at all.
`external`	Assume an external clock source. This is used by the the USCIF35 software stack as an over-ride value if it detects TCLKR has an external source.
`reference`	Use the PLL reference frequency.
`specific`	Use a specific user-selected frequency. The config' variable `uscif.tclk_frequency` is the actual frequency. Its default is `legacy`.
a`utomatic`	Use an automatic frequency selection. The config' variable `uscif.tclk_beginning` is the lower frequency limit. Its default is `superlow` or `minimum`, the cable's lowest supported fixed frequency. The config' variable `uscif.tclk_frequency` is the upper frequency limit. Its default is `maximum`.
`adaptive`	Use an adaptive frequency selection. The config' variable `uscif.tclk_frequency` is the upper frequency limit. Its default is `adaptive`, the cable's highest supported adaptive frequency. Supported only by the XDS560 Revision-D cables.
`inverter`	Use an inverter frequency selection. No restriction on the upper frequency limit. Supported only by the XDS560 Revision-D cables.

The USCIF.TCLK_BEGINNING / FREQUENCY variables

These config' variables are equivalent to these command-line parameters:

```
[-F clock,beginning=hertz,frequency=hertz]
```

These variables select the frequency limits used by the config' variable `uscif.tclk_program` when choosing the JTAG clock frequency used by XDS560 products. Their default values depend on the value of that other variable.

These variables have string values, and numerical values with optional suffixes, that represent literal frequencies as defined in the appendix '5.2 The Frequency of TCLKO and TCLKR'.

The USCIF.TCLK_TESTMODE variable

This config' variable is equivalent to this command-line parameter:

```
[-F clock,testmode=value]
```

This variable modifies the scan-path test that is used to choose the JTAG clock frequency used by XDS560 products. Those frequencies are selected by the config' variables `uscif.tclk_program`, and `uscif.tclk_beginning` / `frequency`. It instructs the emulator to skip the scan-test entirely or to use the scan-test and decide to continue or surrender if it fails. This variable has string values whose defaults are marked bold.

`nothing`	Do nothing - skip the scan-test.
`continue`	Do scan-test, log result, continue after fail.
`s``urrender`	Do scan-test, log result, surrender after fail.

5.6.2 The Secondary Frequency Variables

These variables select the frequency of the JTAG clock for XDS560 products. They are also used to implement the parameters of the `[-F pll]` command, whose detailed description in this manual is thus also a description of these variables.

The USCIF.TCLK_REFERENCE variable

The USCIF.TCLK_OVERFLOW variable

The USCIF.TCLK_INITIAL / TERMINAL variable

The USCIF.TCLK_REFERENCE variable

This variable is equivalent to this command-line parameter:

```
[-F pll,reference=value]
```

This variable modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the reference oscillator frequency for the PLL. This variable has string values whose defaults are marked bold.

`d``efault`	Assume the default reference oscillator for the cable.
`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.

The USCIF.TCLK_OVERFLOW variable

This variable is equivalent to this command-line parameter:

```
[-F pll,overflow=value]
```

This variable modifies the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. It chooses the handling of log-file over-flows when recording PLL actions, as a gross error or a harmless event. This variable has string values whose defaults are marked bold.

`failure`	The PLL log-file generates an error when full.
s`ilent`	The PLL log-file is silent when full.

The USCIF.TCLK_INITIAL / TERMINAL variable

These variables are equivalent to these command-line parameters:

```
[-F pll,initial=boolean,terminal=boolean]
```

These variables modify the programming of the PLL used to generate JTAG clock frequencies in XDS560 products. The `uscif.tclk_initial` variable optionally initialises the PLL just after the USCIF35 software stack is first opened. The `uscif.tclk_terminal` variable optionally re-initialises the PLL just before the USCIF35 software stack is last closed. They are supported by XDS560 emulators with Revision-B, C and D cables. These variables have boolean values whose defaults are marked bold.

The `uscif.tclk_initial` variable has two options.

`false`	Do nothing.
t`rue`	Apply the `uscif.tclk_beginning` frequency.

The `uscif.tclk_terminal` variable has two options.

f`alse`	Do nothing.
`true`	Apply the `uscif.tclk_beginning` frequency.

5.6.3 The Tertiary Frequency Variables

These variables select the frequency of the JTAG clock for XDS560 products. They are also used to implement the parameters of the `[-F length]` command, whose detailed description in this manual is thus also a description of these variables.

The USCIF.TCLK_SCAN_BITS variable

This variable is equivalent to this command-line parameter:

```
[-F length,bits=number]
```

This variable selects the length of the scan-path tests used to validate the frequency of the JTAG clock for XDS560 products. Those tests are chosen by the algorithms selected by the `uscif.tclk_program` variable. The length of the tests can be varied between 1k bits and 512k bits. The tests can have a noticeable effect on the speed of those algorithms at the low TCLK frequencies used by hardware and software simulations, in which case the tests may be reduced in length.

This parameter has string and numerical values whose default is marked bold.

`shortest`	Use IR/DR can-tests that are 1k bits long.
m`edium`	Use IR/DR scan-tests that are 16k bits long.
`longest`	Use IR/DR scan-tests that are 512k bits long.
`1024 - 524288`	Use IR/DR scan-tests that are a specific length.

The USCIF.TCLK_PATH_IRSIZE / DRSIZE variables

These variables are equivalent to these command-line parameters:

```
[-F length,irsize=value,drsize=value]
```

These variables modify the scan-path size measurements used to calibrate scan-path tests. They instruct the emulator to either measure the sizes of the JTAG IR instruction scan-path and JTAG DR bypass scan-paths, or to use explicit values. The scan-path tests are themselves used to validate the frequency of the JTAG clock for XDS560 products. Those tests are chosen by the algorithms selected by the `uscif.tclk_program` variable. The size measurements can have a noticeable effect on the speed of those algorithms at the low TCLK frequencies used by hardware and software simulations, in which case the sizes may be given explicitly. These variables have string and numerical values whose defaults are marked bold.

m`easure`	Measure the size of the IR/DR scan-paths.
`0 - 524288`	Specify the size of the IR/DR scan-paths.

5.6.4 The EMU pins Variables

These variables select the boot-mode and output values of the EMU1-0 pins. They are also used to implement the parameters of the `[-Y emupins]` command, whose detailed description in this manual is thus also a description of these variables.

The USCIF.JTAGBOOT_MODE / VALUE variables

The USCIF.POWERBOOT_MODE / VALUE variables

The USCIF.EMUOUTPUT_MODE / VALUE variables

The USCIF.JTAGBOOT_MODE / VALUE variables

These variables are equivalent to this command-line parameter:

```
[-Y emupins,jtagboot=string]
```

These variables select the 'JTAG nTRST boot-mode' applied to a target system. They represent 2-bit values that are output on the EMU1-0 pins in response to specific conditions. They are supported by 560-class and 510-class emulators with Revision-D cables. These variables have string and numerical values whose defaults are marked bold.

The `uscif.jtagboot_mode` variable has these options:

disable	Both EMU1-0 pins remain hi-z.
enable	The EMU1-0 pins are controlled by `uscif.jtagboot_value`.

The `uscif.jtagboot_value` variable has these options:

disable / hiz	Both EMU1-0 pins remain hi-z.
00	Both EMU1-0 pins are low.
01	EMU1 is low, EMU0 is high.
10	EMU1 is high, EMU0 is low.
11	Both EMU1-0 pins are high.

The 2-bit value is output on the pins when the header's nTRST signal is asserted. The value is sampled by the target system on the following rising edge of nTRST as it is negated. The value continues to be output on the pins after the rising edge until USCIF35 software disables their output buffers.

The USCIF.POWERBOOT_MODE / VALUE variables

These variables are equivalent to this command-line parameter:

```
[-Y emupins,powerboot=string]
```

These variables select the 'Power-on-Reset boot-mode' applied to a target system. They represent 2-bit values that are output on the EMU1-0 pins in response to specific conditions. They are supported by 560-class and 510-class emulators with Revision-D cables. These variables have string and numerical values whose defaults are marked bold.

The `uscif.powerboot_mode` variable has these options:

disable	Both EMU1-0 pins remain hi-z.
enable	The EMU1-0 pins are controlled by `uscif.powerboot_value`.

The `uscif.powerboot_value` variable has these options:

disable / hiz	Both EMU1-0 pins remain hi-z.
00	Both EMU1-0 pins are low.
01	EMU1 is low, EMU0 is high.
10	EMU1 is high, EMU0 is low.
11	Both EMU1-0 pins are high.

The 2-bit value is output on the pins when the header's TVD signal is negated. The value is sampled by the target system on the following rising edge of TVD as it is asserted. The value continues to be output on the pins after the rising edge until USCIF35 software disables their output buffers.

The USCIF.EMUOUTPUT_MODE / VALUE variables

These variables are equivalent to this command-line parameter:

```
[-Y emupins,emuoutput=string]
```

These variables select the 'default or background values' applied to a target system when no specific boot-mode conditions occur. They represent 2-bit values that are output on the EMU1-0 pins in response to specific conditions. They are supported by 560-class and 510-class emulators with Revision-D cables. These variables have string and numerical values whose defaults are marked bold.

The `uscif.emuoutput_mode` variable has these options:

`**d**isable`	Both EMU1-0 pins remain hi-z.
`enable`	The EMU1-0 pins are controlled by `uscif.emuoutput_value`.

The `uscif.emuoutput_value` variable has these options:

`**d**isable / hiz`	Both EMU1-0 pins remain hi-z.
`00`	Both EMU1-0 pins are low.
`01`	EMU1 is low, EMU0 is high.
`10`	EMU1 is high, EMU0 is low.
`11`	Both EMU1-0 pins are high.

The two-bit value is default value output on the pins when neither the 'JTAG nTRST boot-mode' nor the 'Power-on-Reset boot-mode' is being applied to a target system.

5.6.5 The JTAG pins Variables

These variables select the configuration of the JTAG pins – TCLKO / TCLKR, TMS / TDO and TDI. They are also used to implement the parameters of the `[-Y jtagpins]` command, whose detailed description in this manual is thus also a description of these variables.

The USCIF.LOOPBACK_MODE / VALUE variables

The USCIF.LINKDELAY_MODE / VALUE variables

The USCIF.TDOEDGE and USCIF.TDIEDGE variables

The USCIF.LOOPBACK_MODE / VALUE variables

These variables are equivalent to this command-line parameter:

```
[-Y jtagpins,loopback=string]
```

These variables are used only by 560-class and 510-class emulators with Revision-D cables. They configure loop-back of the header's TDO / TDI signals and TCLKO / TCLKR signals. These variables have string values whose defaults are marked bold.

The `uscif.loopback_mode` variable has these options:

`**d**isable`	The loopback value will be ignored.
`enable`	The loopback value will be applied.

The `uscif.loopback_value` variable has these options:

`**d**isable`	The loopback of clock and/or data is disabled.
`data`	The loopback of TDO / TDI is enabled.
`clock`	The loopback of TCLKO / TCLKR is enabled.
`total`	The loopback of both data and clock is enabled and the detection of power-loss and cable-break is disabled.

When loop-back of both sets of signals is selected then both the header's TVD power-loss detect and TDIS cable-break detect signals are also disabled. This enables emulator testing when the target is powered-off or the header is dis-connected.

The USCIF.LINKDELAY_MODE / VALUE variables

These variables are equivalent to this command-line parameter:

```
[-Y jtagpins,linkdelay=string]
```

These variables are used by all 100/510-class and 560-class emulators. They provides the link-delay values applied to the scan-controller that compensates for the pipe-line in the cable through which the TMS / TDO and TDI signals are clocked. The maximum link-delay value depends on the scan-controller design. No scan-controllers support values higher than 31. These variables have string and numerical values whose defaults are marked bold.

The `uscif.linkdelay_mode` variable has these options:

`**d**isable`	The link delay value will be ignored.
`enable`	The link delay value will be applied.

The `uscif.linkdelay_value` variable has these options:

`**d**isable`	The cable's designed link delay value will be applied.
`automatic`	The link delay value will be calculated automatically.
`0 - 31`	The explicit link delay value will be applied.

The USCIF.TDOEDGE and USCIF.TDIEDGE variables

These variables are equivalent to these command-line parameters:

```
[-Y jtagpins,tdoedge=string,tdiedge=string]
```

The former variable is used by all 100/510-class and 560-class emulators. The latter variable is currently ignored by all emulators. These variables select the timing of the TMS / TDO and TDI signals, relative to the rising and falling edges of TCLKR signal on the header of the emulator. The USCIF35 software stack may slightly adjust the link-delay to match the chosen timing. These variables have boolean values whose defaults are marked bold.

The `uscif.tdoedge` variable has these options:

fall	Output TDO on the falling TCLKR edge.
rise	Output TDO on the rising TCLKR edge.

The `uscif.tdiedge` variable has these options:

fall	Input TDI on the falling TCLKR edge.
rise	Input TDI on the rising TCLKR edge.

These variables use the JTAG naming convention where TMS / TDO are output pins from the header to the target system and TDI is the input pin on the header from the target system. This is different from the TI and ARM emulator header documentation that names the data output pin as TDI and the data input pin as TDO.

The choice of clock edge can have a significant impact on the maximum clock frequency and thus the performance of utilities, loaders and debuggers. The default is rising edge timing because that achieves 39.0MHz operation with many target systems, while the use of falling edge operation achieves only 25.5MHz operation with the same target systems.

5.6.6 The Scan Command Variables

There is currently only a single variable that affects scan-commands:

The USCIF.NOAMBLE variable

The USCIF.NOAMBLE variable

This variable is used to disable the scan-path pre-ambles and post-ambles that are calculated by USCIF35 software and used to select a single target device in a multi-processor system. This variable has boolean values whose default is marked bold.

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

This variable 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 that actually only has a single target device.

5.6.7 The Error Analysis Variables

There is currently only a single variable that affects error-analysis:

The USCIF.NOCHECK variable

The USCIF.NOCHECK variable

This variable is used to disable the internal double-check function that USCIF35 software uses to evaluate raw error values and replace them with more informative errors. This variable has a boolean values whose default is marked bold.

no	Report the error value diagnosed by the double-check function.
yes	Report the raw error values without any double-check.

This variable may be used to aid in debugging problems with target systems by suppressing the masking of raw error values by the double-check function. The most common effect of that function is to replace scan-controller buffer access timeout errors with the cable-break, power-loss and dead-clock errors that can be their root cause.

5.6.8 The Clock Attribute Variables

These variables modify the behaviour of the USCIF35 software stack so as to optimise or enable operation at relatively high and low TCLK frequencies. They are not explicitly associated with any of the DBGJTAG commands or parameters. However they can be applied from the command-line using the [-V apply] command.

The USCIF.OKSTALL variable

The USCIF.SLOWCLK / SLOWFRQ variables

The USCIF.FASTCLK variable

The prioritisation scheme for these variables gives successively higher priority to variables that configure successively lower frequency operation. The `uscif.okstall` variable overrides the other three variables, which are then ignored. The `uscif.slowfrq` variable implies slow-clock is enabled even if `uscif.slowclk` is not asserted, and overrides the `uscif.fastclk` variable, which is than ignored. The `uscif.fastclk` variable is effective only if the other variables are not used at all,

The USCIF.OKSTALL variable

This variable is used to disable the generation of scan-controller timeout errors and dead-clock errors when a stall occurs in the TCLKR signal from external clock sources. This variable may be used to enable operation of emulators with both simulated and physical target systems that temporarily stall the TCLKR signal or reduce the TCLKR frequency.

This variable is used by both 510-class emulators and 560-class emulators. This variable should be used cautiously. When enabled, the USCIF35 software stack, and any software tools mounted on that stack, will also stall in response to dead-clocks that result from cable breaks and power-losses, in addition to dead-clocks resulting from a stall of the TCLKR signal. This is not user-friendly behaviour. This variable has boolean values whose default is marked bold.

no	Report scan-controller timeout errors and dead-clock errors.
yes	Allow TCLKR to stall indefinitely without reporting any error.

The USCIF.SLOWCLK / SLOWFRQ variables

These variables enable the reliable operation at relatively low TCLKR frequencies, of emulators that cannot measure frequency. They instruct the USCIF35 software stack to apply additional polling operations to the scan-controller hardware.

These variables are used by 100/510-class emulators. The actual frequency value depends on the design of each emulator product. For the XDS100pp emulator these variables are always required because of its low 100KHz operating frequency. For the XDS510 emulator these variables are required below 5.0MHz, which is half that product's standard 10.368Mhz TCLK. All 560-class emulators are able to measure the TCLKR frequency, and automatically configure themselves for reliable operation without user intervention. However 560-class emulators still allow these variables to override their frequency measurements, as a debug tool for the USCIF35 software stack.

When `uscif.slowclk` is asserted the default value of `uscif.slowfrq` is 20000, ensuring reliable operation down to 20KHz. If reliable operation is required at lower frequencies then an explicit frequency value must be applied to `uscif.slowfrq`. These variables have been tested with `uscif.slowclk` asserted and values of `uscif.slowfrq` down to 488, supporting reliable operation down to 488Hz.

These variables have boolean, string and numerical values whose defaults are marked bold.

The variable `uscif.slowclk` has these options:

no	Disable reliable operation at relatively low TCLK frequencies.
yes	Enable reliable operation at relatively low TCLK frequencies. This may occur at the expense of performance.

The variable `uscif.slowfrq` has these options:

default	The lowest frequency reliable operation is 20000Hz.
>= 488	The lowest frequency reliable operation is given as an integer representing a literal value in units of Hertz.

The USCIF.FASTCLK variable

This variable improves the performance at relatively high TCLKR frequencies, of emulators that cannot measure frequency. It instructs the USCIF35 software stack to skip certain polling operations on the scan-controller hardware.

This variable is used by 510-class emulators. The actual value is dependant on the design of each emulator product. For the XDS510 emulator this variable is effective above 15.0MHz, which is one and a half times that product's standard 10.368Mhz TCLK. All 560-class emulators are able to measure the TCLKR frequency, and also do not require frequency dependent optimisations to achieve acceptable performance. This variable has boolean values whose default is marked bold.

no	Skip optimisations that require high TCLK frequencies.
yes	Enable optimisations that require relatively high TCLK frequencies and improve performance.

5.6.9 The Emulator Selection Variables

These variables select the emulator itself. They are also used to implement the parameters of the `[-X program]`, `[-X adapter]` and `[-X network]` commands, whose detailed description in this manual is thus also a description of these variables.

The USCIF.ECOM_DRVR / PORT variables

These variables are equivalent to the command-line parameters

```
[-X program,driver=filename.out,ioport=address].
```

These variables are used to select an emulator or other product that was developed using the XDS560 Sourceless-EPK. A different pair of variables are used to enable communication with 510-class products.

These variables have string and numerical values whose defaults are marked bold.

The variable `uscif.ecom_drvr` has a string value that is the file-name of the program, with a *.out extension, that is loaded by the USCIF35 software stack to operate a 560-class emulator.

xds560.out	The program for TI's PCI-bus XDS560 emulator.
bh560ubp.out	The program for Blackhawk's bus-powered USB 560-class emulator.

The variable `uscif.ecom_port` has a 32-bit decimal or hexa-decimal value that is the port address of the emulator.

0x0	The typical port address for the first emulator of a particular type installed.

The SEPK.POD_DRVR / PORT variables

These variables are equivalent to the command-line parameters:

```
[-X adapter,driver=filename.dll,ioport=address].
```

These variables are used to select an emulator or other product that was developed using the XDS510 Sourceless-EPK. A different pair of variables are used to enable communication with 560-class products.

These variables have string and numerical values whose defaults are marked bold.

The variable `sepk.pod_drvr` has a string value that is the file-name of the adapter, with a *.dll extension, that is loaded by the USCIF35 software stack to operate a 510-class emulator.

xds100p p.dll	The adapter for TI's parallel-port XDS100pp emulator.
xds510.dll	The adapter for TI's ISA-bus XDS510 emulator.

bh`pjtag3.dll`	The adapter for Blackhawk's bus-powered USB 510-class emulator.

The variable `uscif.sepk_port` has a 32-bit decimal or hexa-
decimal value that is the port address of the emulator.

`0x0`	The typical port address for the first emulator of a particular type installed.

The SEPK.NET_HOST / PORT variables

These variables are equivalent to the command-line parameters:

[-X remote,nethost=`name,netport=number`].

Currently this functionality is only supported by a
limited number of emulator programs and adapters.

The variable `sepk.net_host` is the network name of the remote computer on which a TCP or UDP service supports client connections to an emulator. The `sepk.net_host` variable accesses the same functionality as this parameter and has similar values.

- The network name can be a short locally recognised name that is a single string.
- The network name can also be a fully qualified domain name (FQDN),
  which is multiple strings separated by single dots characters.
- The network name can also be an internet protocol (IP) address, |which is
  four small decimal numbers from 0 to 255 that are separated by dot characters.

The variable `sepk.net_port` is the network port on the remote computer at which a TCP or UDP service supports client connections to an emulator. The `sepk.net_port` variable accesses the same functionality as this parameter and has similar values.

- The network port is the TCP or UDP port number,
  which is a relatively small decimal number from 0 to 65535

5.6.10 The Emulator Communication Variables

These variables modify the behaviour of the USCIF35 software stack when operating adapters for 510-Sourceless-EPK products. They are not explicitly associated with any of the DBGJTAG commands or parameters. However they can be applied from the command-line using the [-V apply] command.

The SEPK.POD_NOSPLASH variable

This variable is used with adapters designed using the XDS510 Sourceless-EPK that generate product specific splash-dialogs at startup. The variable is used to suppress the splash-dialog in development and test environments where it is considered a distraction rather than a benefit. The `bhjtag3.dll` adapter for the Blackhawk USB510 emulator supports this variable. This variable has boolean values whose default is marked bold.

no	Enable the display of the splash-dialog at start-up.
yes	Disable the display of the splash-dialog at start-up.

The SEPK.POD_BLKMODE variable

This variable is used with adapters designed using the XDS510 Sourceless-EPK that are capable of operating in both normal-mode and block-mode. The `xds510.dll` adapter for the TI XDS510 emulator supports this variable. This variable has boolean values whose default is marked bold.

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

The SEPK.POD_LOGMODE / LOGFILE / LOGTYPE variables

These variables are used with adapters designed using the XDS510 Sourceless-EPK. They are used to aid in the development of those adapters by recording the interaction of the USCIF35 software stack with the adapters as text a formatted log-file. The `xds510.dll` adapter for the TI XDS510 emulator supports these variables.

These variables have boolean and string values whose defaults are marked bold.

The variable `sepk.pod_logmode` has boolean values whose default is marked bold.

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

The variable `sepk.pod_logfile` has string values whose default is '`./sepk-logfile.txt`'.

filename	The file-name of the Sourceless-EPK adapter log-file.

The variable `sepk.pod_logtype` has string values whose default is marked bold.

brief	Generate a brief adapter log-file.
verbose	Generate a verbose adapter log-file.

5.6.11 The Router Configuration Variables

These variables modify the behaviour of the USCIF35 software stack when operating routers. These variables are not explicitly associated with any of the DBGJTAG commands or parameters other than their indirect effects on the [-R routelist] and [-R matrixlist] commands. However they can be applied from the command-line using the [-V apply] command.

The variables for router support are:

The ROUTER.CONTROL_SCAN variable

The ROUTER.SUBPATH_DEFAULT / CUSTOM variables

The variables for only Icepick-A, B, C routers.

The ROUTER.CONTROL_SCAN variable

This variable is used to modify the behaviour of all routers.

This variable is used to disable / enable the sub-path descriptions in a board config' file scan-path description. It is typically used to support testing of board config' files with hierarchical scan-path descriptions on physical scan-paths that do not have routers. This variable has boolean values whose default is marked bold.

`no`	Disable the sub-paths of all routers.
`y``es`	Enable the sub-paths of all routers.

The ROUTER.SUBPATH_DEFAULT / CUSTOM variables

This variable is used to modify the behaviour of all routers.

These variables are used to ignore or apply the default and custom sub-path selections that may be included in each sub-path description in a board config' file scan-path description. They are typically used to temporarily ignore all such selections without editing every sub-path description.

These variables have boolean values whose defaults are marked bold.

The variable `router.subpath_default` has these options:

`no`	Ignore the default sub-path selections of all router sub-paths.
`y``es`	Apply the default sub-path selections of all router sub-paths.

The variable `router.subpath_custom` has these options:

`no`	Ignore the custom sub-path selections of all router sub-paths.
`y``es`	Apply the custom sub-path selections of all router sub-paths.

The ROUTER.IDLE_DELAY variable

This variable is used to modify the behaviour of only Icepick-A, B and C routers.

This variable is used to apply a delay of a number of TCLK periods when each sub-path select / de-select is completed. The delay occurs in the JTAG Run-Test/Idle state that marks that completion. It is typically used to add additional delay to compensate for differences in behaviour that occur between router simulation and router hardware. This variable has numeric values whose default is `10`.

`0 - 65535`	Wait for a number of TCLK periods after select / de-select.

The ROUTER.SKIP_POLLING variable

This variable is used to modify the behaviour of only Icepick- A, B and C routers.

This variable is used to enable or disable the polling of router status values accessed via the JTAG IR/DR shift states during each sub-path select / de-select operation. It is typically used to enable or disable this status polling so as to compensate for differences in behaviour that occur between router simulation and router hardware. This variable has boolean values whose default is marked bold.

n`o`	Enable polling of router status values during select / de-select.
`yes`	Disable polling of router status values during select / de-select.

The ROUTER.RGST_SUPPORT variable

This variable is used to modify the behaviour of only Icepick-A and B routers.

This variable is used to disable or enable the use of the four `router.sys_opn / cls` and `router.pin_opn / cls` variables. Those four register variables have default values of 0 which is likely never appropriate and this variable is used to explicitly suppress those variables. This variable has boolean values whose default is marked bold.

n`o`	The four register variables are not used.
`yes`	The four register variables are used.

The ROUTER.RGST_SYS_OPN / CLS variables

These variables are used to modify the behaviour of only Icepick-A and B routers.

The two variables `router.sys_opn / cls` are used to load values
into the System Control register of the router at start-up and shut-down.

The two variables have numeric values, most likely they will be entered
as 16- or 32-bit hexadecimal values since they represent register values.
The default values are `0`, which are likely never appropriate.

`0x0 - 0xFFFFFFFF`	The value applied to the System Control register.

The ROUTER.RGST_PIN_OPN / CLS variables

These variables are used to modify the behaviour of only Icepick-A and B routers.

The two variables `router.pin_opn / cls` are used to load values
into the Pin Manager register of the router at start-up and shut-down.

The two variables have numeric values, most likely they will be entered as 16- or 32-bit hexadecimal values since they represent register values. The default values are `0`, which are likely never appropriate.

`0x0 - 0xFFFFFFFF`	The value applied to the Pin Manager register.

The ROUTER.PRIVATE_INST variable

This variable is used to modify the behaviour of only Icepick-C routers.

This variable is used to enable or disable the application of the private instruction command to the router at start-up and shut-down. This variable has boolean values whose default is marked bold.

`n``o`	The private instruction will not be applied to the router.
`yes`	The private instruction will be applied to the router.

The ROUTER.ADAPTIVE_TCLK variable

This variable is used to modify the behaviour of only Icepick-C routers.

This variable is used to enable or disable the application of the adaptive clocking flag to the router at start-up and shut-down. This variable has boolean values whose default is marked bold.

`n``o`	The router will not use adaptive clocking.
`yes`	The router will use adaptive clocking.

5.6.12 The Two-Pin JTAG Variables

These variables modify the behaviour of the USCIF35 software stack to support the IEEE 1149.7 Draft Standard for “advanced JTAG” or “two-pin JTAG”. They configure support for operating the Debug and Test Interface (DTI) between a Debug and Test Controller (DTC) and Debug and Test Target (DTT). These variables are not explicitly associated with any of the DBGJTAG commands or parameters other than their indirect effects on the `[-F, -S, -R]` commands. However they can be applied from the command-line using the `[-V apply]` command.

The variables for advanced JTAG support are:

The USCIF.JTAG_VERSION variable

The USCIF.JTAG_ADAPTER variable

The USCIF.JTAG_DELTA variable

The USCIF.JTAG_EDGE variable

The USCIF.JTAG_FORMAT variable

The USCIF.JTAG_VERSION variable

This variable is used to indicate which version of the software support for aJTAG should be used. During the development of the aJTAG hardware the USCIF35 software support is expected to evolve, with the addition, modification and deletion of its config’ variables.

The alpha1 and alpha2 versions of aJTAG software support this variable.
This variable has string values whose default is marked bold.

`**u**nknown`	The USCIF support for aJTAG is not specified
`alpha1`	The 1st version of USCIF support for aJTAG.
`alpha2`	The 2nd version of USCIF support for aJTAG.

The USCIF.JTAG_ADAPTER variable

This variable is used to indicate if the hardware support for aJTAG is installed
between the emulator and target system. Also if it is installed then should
the 1194.1 protocol or the the 1194.7 protocol be used.

The alpha1 and alpha2 versions of aJTAG software support this variable.
This variable has string values whose default is marked bold.

remove	The aJTAG hardware is not present in the connection.
disable	The hardware is present but the 1194.1 protocol is used.
enable	The hardware is present and the 1194.7 protocol is used.

The USCIF.JTAG_DELTA variable

This variable is used to indicate if the hardware support
for aJTAG has a fixed link-delay inherent in its design.

The alpha1 and alpha2 versions of aJTAG software support this variable.
This variable has string and numerical values whose default is marked bold.

none	The aJTAG hardware adds no link-delay.
0 - 31	The aJTAG hardware has its own fixed link-delay.

The USCIF.JTAG_EDGE variable

This variable is used to indicate if the hardware support for aJTAG should be
configured to sample the TMSC input signal on the rising or falling edge of TCLK.

Only the alpha2 version of aJTAG software supports this variable.
This variable has string values whose default is marked bold.

rise	The TMSC input is sampled on the TCLK rising edge.
fall	The TMSC input is sampled on the TCLK falling edge.

The USCIF.JTAG_FORMAT variable

This variable is used to choose which of the JTAG and aJTAG scan formats should be used.

The alpha1 and alpha2 versions of aJTAG software support this variable.
This variable has string values (with synonyms) whose default is marked bold.

disable	Use the standard 1149.1 protocol.

`jscan0/jtag0`	Use the aJTAG jtag scan mode #0.
`oscan1/optimise1`	Use the aJTAG optimised scan mode #1.
`oscan2/optimise2`	Use the aJTAG optimised scan mode #2.
`oscan3/optimise3`	Use the aJTAG optimised scan mode #3.
`oscan5/optimise5`	Use the aJTAG optimised scan mode #5.
`oscan6/optimise6`	Use the aJTAG optimised scan mode #6.
`oscan7/optimise7`	Use the aJTAG optimised scan mode #7.
`mscan/multiple`	Use the aJTAG mutli-device scan mode.

5.7 *The XDSRESET and XDSPROBE Utilities*

The documentation for XDSRESET and XDSPROBE is now expressed in terms of the equivalent DBGJTAG features, so as to avoid duplicate documentation, and to encourage users to adopt DBGJTAG, and to abandon XDSRESET and XDSPROBE.

5.7.1 The Synopsis

XDSRESET is a utility used to reset scan-controllers and scan-paths.

```
xdsreset     [-h] [-v] [-r] [-s signal]
             [-o [file]] [-f file]
             [-F program/adapter] [-p address]
```

XDSPROBE is a utility used to test scan-controllers and scan-paths.

```
xdsprobe     [-h] [-v] [-r] [-s signal]
             [-o [file]] [-f file]
             [-F program/adapter] [-p address]
             [-b] [-i] [-g [data]] [-y] [-c [number]]
             [-d [limit],[limit]] [-d [center]] [-k]
             [-a] [-n name]
```

5.7.2 The Introduction

The DBGJTAG utility with its new-style command-line options replaces both XDSRESET and XDSPROBE. However the now depreciated XDSRESET and XDSPROBE utilities are still provided to allow test scripts and batch files that use old-style command-line options to remain compatible with new USCIF35 releases. Those depreciated utilities are implemented (or more accurately – emulated) as simple command-line parsers that use a limited set of the features available in the core library of DBGJTAG.

The XDSRESET and XDSPROBE utilities have default option values that can result in surprising behaviour. These utilities apply the [-r] reset command option if no major options are chosen. They apply the [-f board.cfg] if no board config' file is selected, and that default file typically selects an ISA-bus XDS510 emulator. They also apply the [-F xds510.dll] and [-p 0x240] emulator selection options for an ISA-bus XDS510 if no emulator is selected. The defaults are applied even if no such hardware is installed! The remnants of the non-plug-and-play ISA-bus in modern PC's will not report an error when software attempts to touch non-existent hardware.

The DBGJTAG utility is implemented without defaults so as to avoid such surprises. Its default behaviour is much improved over XDSRESET and XDSPROBE. The DBGJTAG utility reports an error instead of applying obscure default values for essential parameters that are not explicitly used on the command-line. The DBGJTAG utility used with no arguments simply prints the brief help, and does not touch any emulator hardware.

5.7.3 The Minor Options

The [-h] and [-?] option

This XDSRESET and XDSPROBE option requests brief help.

```
xdsprobe -h
```

It is similar to this DBGJTAG command:

```
dbgjtag  -h
```

The [-r] and [-v] option

This XDSRESET and XDSPROBE verbose option
initialises the emulator and pulses the nTRST signal.

```
xdsprobe -F xds560 -p 0 -rv
```

It is similar to this DBGJTAG command:

```
dbgjtag  -d xds560 -p 0 -rv
```

The [-s signal] option

This XDSRESET and XDSPROBE option pulses and operates the nSRST signal.

```
xdsprobe -F xds560 -p 0
         -s assert/negate/toggle
```

It is similar to this DBGJTAG command:

```
dbgjtag  -d xds560 -p 0
         -s assert/negate/toggle
```

The [-F program/adapter] option

This XDSRESET and XDSPROBE option provides
the name of an emulator program or adapter.

```
xdsprobe -rv -p 0 -F program.out/adapter.dll]
```

It is similar to this DBGJTAG command:

```
dbgjtag  -rv -p 0 -d program.out/adapter.dll]
```

The [-p address] option

This XDSRESET and XDSPROBE option provides the address of an emulator:

```
       xdsprobe -rv -F xds560 -p address
```

It is similar to this DBGJTAG command:

```
       dbgjtag  -rv -d xds560 -p address
```

The [-f filename] option

This XDSRESET and XDSPROBE option provides the name of a board config file:

```
       xdsprobe -rv -f filename
```

It is similar to this DBGJTAG command:

```
       dbgjtag  -rv -f filename
```

The [-o filename] option

This XDSRESET and XDSPROBE option provides the name of an output file.

```
       xdsprobe -rv -f filename -o filename
```

It is similar to this DBGJTAG command:

```
       dbgjtag  -rv -f filename -o filename
```

5.7.4 The Major Options

The [-b] option

This XDSPROBE option requests the broken-path scan-test.

```
       xdsprobe -Fxds560 -p0 -b
```

It is similar to this DBGJTAG command:

```
       dbgjtag  -dxds560 -p0
                -S pathlength
                -S brokenpath
```

The [-i] option

This XDSPROBE option requests the integrity scan-test.

```
       xdsprobe -Fxds560 -p0 -i
```

It is similar to this DBGJTAG command:

```
       dbgjtag  -dxds560 -p0
                -S pathlength
                -S integrity
```

The [-g value] and [-c number] options

These XDSPROBE options request the given-data scan-test.

```
       xdsprobe -Fxds560 -p0 -g data -c number
```

They are similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0
         -S pathlength
         -S givendata,literal=data,repeat=number
```

The [-y] and [-c number] options

These XDSPROBE options request the scan-controller test.

```
xdsprobe -Fxds560 -p0 -y -c number
```

They are similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0 -T control,repeat=number
```

The [-d value] option

This XDSPROBE option requests the combined frequency and scan test:

```
xdsprobe -Fxds560 -p0 -d freq
```

It is similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0 -G single,center=freq
```

The [-d value1,value2] option

This XDSPROBE option requests the combined frequency and scan test:

```
xdsprobe -Fxds560 -p0
         -d freq1,freq2
```

It is similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0
         -G range,lowest=freq1,highest=freq2
```

The [-k] option

This XDSPROBE option requests the TCLK programming log-file:

```
xdsprobe -Fxds560 -p0 -k
```

It is similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0 -F inform,logfile=yes
```

The [-a] and [-n name] options

These XDSPROBE options request the experimental device analysis.

```
xdsprobe -Fxds560 -p0
         -a -n name
```

They are similar to this DBGJTAG command:

```
dbgjtag  -dxds560 -p0
         -A scanpath,device=name
```

5.8 The Design of the DBGJTAG Utility

5.8.1 The Design Requirements

This section lists the requirements considered during the design of the utility.

The utility and handbook are not intended for novice users

The users of the utility and its manual are expected to have JTAG and emulation experience.

The utility may be a multi-purpose utility

The accumulation of multiple major commands in a single utility is accepted.
The need for the user to type long command-line options is accepted.

The utility shall not surprise the user

The utility shall not touch hardware when used without command-line options.
The utility shall not apply default values to command-line options.

The utility shall not be unnecessarily verbose

The utility shall output brief text as its default behaviour.
The utility shall not output boiler-plate (fixed) text.

The utility shall provide limited built-in help

The complete help shall be provided in a handbook.
The use of the utility without any options, shall generate generic brief help.
The minor help option '-h', shall generate generic brief help.
The major help commands '-X help' shall generate option-specific brief help.
The utility shall not incorporated any other help mechanism.

The lower case command-line options are to be 'minor commands'

The selection of the emulator, the selection of the config' file,
and the application of resets to the emulator and target system.

The upper case command-line options are to be 'major commands'

The selection of significant actions applied to
the emulator, the scan-paths and the target devices.

The utility shall implement 'policy and mechanism'

The utility shall be a command-line parser/driver and a core library.
The utility shall permit other parsers and graphical interfaces to use the library.

5.8.2 The Design Issues

This section lists the issues exposed during the design of the utility.

The utility shall be modified over time to comprehend these issues.

The utility lacks explicit support for its own testing

The utility does not output status and error values in a fixed summary
format appropriate for the automated testing of the utility itself.

The utility lacks thorough support for a GUI interface

The utility does not output status and error values using a call-back
mechanism appropriate for the use of a GUI instead of a CLI.

The utility outputs status and error values for the CLI interface directly from
the shared library. This not compatible with the use of a GUI instead of a CLI.

The command-line parser uses a single data-structures

The utility uses a single large data-structure to store all parsed command-line options.
The utility uses a single large data-structure to store the shared library status and results.
The resulting two large data-structures are difficult to maintain.

The utility should used multiple option specific
data-structures nested within a parent data-structure.

The utility and USCIF library lack text-formatted log-files

The binary-formatted log-files output by the USCIF library (example TCLKO programming)
to the utility are difficult to debug, difficult to automatically test, and time-consuming to enhance.

The USCIF library should use text-formatted log-files unless it is a clear performance issue.

The utility lacks a satisfactory scan-path analysis command

The original analysis command has been upgraded to comprehend scan-paths
with routers. However the result still does not do a satisfactory
amount of scan-path analysis. The issue is being investigated.

The utility lacks a satisfactory scripting interface

The original scripting interface based on the old PDM parser has been abandoned
because the cost to support was too high compared to the benefits. An alternative
open-source scripting interface is being investigated.

This is the last page of “The User’s Guide To DBGJTAG”.