CoreSight™ Access Tool (CSAT)

User Guide

CSAT Version 2.6.0 or later

ARM

Web Address
http://www.arm.com

Contents

1 ABOUT THIS DOCUMENT

1.1 Change control

1.1.1 Current status and anticipated changes

Initial Draft – external document from original internal guide. Applicable to version 1.1.5 of CSAT.

Update to coincide with release of CSAT 1.2.0 alongside RVI 3.3.

Update for release of CSAT 2.0.7 with trace command set 2.0.1.1; use with DSTREAM / RVI firmware 4.3.0 and later, and VSTREAM client 3.3c and later.

Document number change to ARM-EPM-051792. Update for release of CSAT 2.6.0, trace and VSTREAM command sets. Use with DSTREAM / RVI firmware 4.8.0 and later, VSTREAM client 4.14a and later.

1.2 References

This document refers to the following documents.

Ref	Doc No	Author(s)	Title
1	ARM DUI 0481E	ARM	DSTREAM, Setting Up the Hardware
2	ARM DUI 0515E	ARM	RVI and RVT, Setting Up the Hardware
3	ARM DUI 0498E	ARM	DSTREAM and RVI, Using the Debug Hardware Configuration Utility
4	DS165-PRDC-011261	ARM	VSTREAM, Client User Guide

1.3 Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
Debug Control Unitls	Hardware (or software virtual equivalent) that creates a connection between the debugger and the target system. Drives the target connection – JTAG, SWD, ‘virtual’ as required.
Trace Capture Unit	Hardware (or software virtual equivalent) that captures the trace output from the target via a trace connector.
CSAT	CoreSight ^{TM} Access Tool
DSTREAM	Hardware DSTREAM Debug Control and Trace Capture Unit. Connects to Hardware JTAG or Serial Wire connections.
VSTREAM (client)	Virtual Debug Control Unit and Trace Capture Unit. Connects to virtual RTL Simulation or Emulation targets.
Virtual Target	A target system running in an RTL simulation or emulation environment, connecting using the VSTREAM transactor.
RVI	RealView ICE Debug control unit. (no longer manufactured)
RVT2	RealView Trace Capture unit for RVI. (no longer manufactured)

2 SCOPE

This document describes the commands and usage of the CoreSight Access Tool version 2.6.0 or later. This tool is designed for the testing of CoreSight systems using DSTREAM hardware or the VSTREAM virtual connection, attached to the test system.

The trace capture capability in the DSTREAM hardware, or the VSTREAM client is also required if testing involves evaluation of trace output via a CoreSight TPIU / trace connector. Trace captured on the target system such as in the CoreSight ETB or ETR does not require a trace capture unit.

It is assumed that the reader is familiar with the use of the DSTREAM / RVI products (see Ref 1, 2), the debug hardware configuration utilities (Ref 3), and the RVT2 trace capture unit (Ref 2)

VSTREAM user information can be obtained from the VSTREAM client user guide (Ref 4).

2.1 DSTREAM, VSTREAM and RVI/RVT compatibility

CSAT version 2.6.0 (and later) is delivered as part of DS-5 debugger versions 5.17 and later. Previous versions of CSAT were shipped with earlier DS-5 products.

The CSAT tool is designed to work closely with the DSTREAM hardware, and installed firmware versions, from version 4.0 onwards.

CSAT also connects with the VSTREAM client software, version 3.3c and later, but VSTREAM client control requires use with VSTREAM client version 4.14, and the VSTREAM client command extension DLL.

See the **`csat_changelog.txt`** that is supplied with CSAT for updated compatibility, enhancement and bug-fix information.

3 INTRODUCTION

The CoreSight ™ Access Tool (CSAT) program is a console based test program for validating CoreSight systems. The tool contains a number of commands to allow access to a CoreSight system, using the DSTREAM and VSTREAM debug control units, and associated trace capture functionality where required. CSAT has sufficient commands to control both debug control units, in additional to accessing CoreSight components.

The summary of the available commands uses the following syntax. Most commands have a long and short format. These are shown as follows:

```
COMMAND (CMD) opt1 [opt2]  opt3 | opt4 opt5=param1{,param2}*
```

The options use '**[ ]**'for optional, and '**|**' for exclusive alternatives, '**{ }**' to bracket alternative sets.

'*****' means repeat as required.

For example, the DAP memory read command uses **dpmemread** and **dmr**.

3.1 Getting Started

CSAT has versions for both Linux (**csat**) and Windows (**csat.exe**).

First ensure that the debug control unit (DSTREAM) has the JTAG cable / Serial wire connected to the target you intend to use during the session and power up the unit. If using VSTREAM, ensure that the client is running and connected to the virtual target running in RTL simulation or emulation.

3.1.1 Installation

CSAT ships with a number of key files and DLLs for full functionality. These are present in the CSAT install directory. DS-5 ships with a windows console or linux shell with the paths set correctly for the CSAT installation.

At start-up CSAT will attempt to load the file **cscompdata.dat,** first searching the local working directory for this file. If it is not present here, then CSAT will examine the following environment variables to try to determine the install directory:-

ARM_CSAT_PATH	- may be set by user.
ARM_RVI_TOOLS	- set by the DS-5 installation environment.

If CSAT is installed into a common network location then users should set the ARM_CSAT_PATH environment variable to the installed network location, as well as any PATH environment variable to enable CSAT to function correctly.

3.1.2 Command Line Options

```
csat [<start_up_batch_file>] [-cmdset <cmdset_name> <cmdset_dll_filename>]*
```

<start_up_batch_file> : this optional parameter is the name of a file containing CSAT commands that will be run when CSAT starts. See the '**batch**' command for further information about these command files. This parameter will always be the first parameter on the CSAT command line if present.

-cmdset : an optional parameter that loads an additional command set into CSAT. The command set is referenced within CSAT using the **cmdset_name** parameter. If this parameter is longer than 5 characters then a long and short form of the command will be formed using the 1st five characters for the short form command.

Additional command sets can only be loaded at CSAT start-up, via the command line. The command set is loaded before any start-up batch file is executed, therefore commands from the command set may be used in the start-up batch file.

Run **CSAT** with desired options to see the sign on and command prompt. Note that the sign on indicates a successful load of the **cscompdata.dat** file, and the successful load of a command set.

e.g :-

```
csat –cmdset vstrm_control vstrm_cmd_dll.dll

###############################################
#  CSAT -   CoreSight Access Tool      v2.6.0 #
#                                             #
#       [with Trace Commands v 2.0.1.1]       #
#                                             #
#        Copyright 2007-2013 ARM Limited      #
###############################################

CoreSight Component Data file read successfully.

Loading command set vstrm_control
Loaded Command Set DLL for command 'vstrm'
VSTREAM control command set : Version 1.0

%>
```

3.1.2.1 VSTREAM command set

At version 2.6.0 of CSAT there is a single additional command set available – the VSTREAM client control command set. This allows CSAT to connect to a running VSTREAM client and control trace capture and other features in the VSTREAM client. The following CSAT command line will load the VSTREAM command DLL into CSAT:-

```
csat –cmdset vstrm_control vstrm_cmd_dll.dll
```

This is the recommended command line to use the CSAT examples supplied with VSTREAM.

Within CSAT command files or at the CSAT command prompt the VSTREAM commands are referenced using the syntax:-

```
vstrm  <vstrm_command> <vstrm_command_options>
```

The VSTREAM command set uses the same commands as the VSTREAM_terminal application – see the VSTREAM Client User Guide for full details on the commands available.

3.1.3 Connecting to the Debug Control Unit

First CSAT must be connected to the DSTREAM / VSTREAM debug control unit.

The command **connect** (or **con** is the short version) can be used to connect either via TCP or USB (if connected by USB). With TCP use the debug control unit IP address, or the host name (this depends on your local networks DNS server – if the hostname connection fails try IP address).

e.g.

```
con TCP:Koyaanisqatsi
con TCP:192.168.23.240
con USB
```

The same command is used to connect to the VSTREAM client, although if the VSTREAM client and CSAT are running on the same PC then the following localhost command works.

```
con TCP:localhost
```

3.1.4 Setting the Target Scan Chain

Next the debug control unit needs to get information about devices on the scan chain. This can be done in one of two ways.

1) The scan chain can be directly specified.

2) The debug control unit can auto-detect the chain, which succeeds only if all the devices on the chain are recognised.

The JTAG clock frequency must also be set at this time. The chain command is used to set this information.

3.1.4.1 Directly specified Scan Chain

The most common command used is to connect to a CoreSight System with a single DAP on the scanchain.

```
chain dev=ARMCS-DP clk=10000000
```

When using the VSTREAM client, the clock parameter may still be supplied as part of the command but will be ignored. The scan chain is simulation by the VSTREAM client to appear similar to a hardware scanchain used by DSTREAM.

Note: CoreSight components and Cortex cores will never appear in the scan chain set up by these commands. These are attached to the CoreSight DAP.

3.1.4.2 Auto-detected Scan Chain

This example autodetects an adaptively clocked system:-

```
chain dev=auto clk=A
```

A typical response to this is:

```
Jtag clock set to 50000000A
ID:0 ARMCS-DP
```

This shows a single DAP devices on the scan chain, a DAP (ARMCS-DP) at scan chain index 0

3.1.4.3 Serial Wire Connections

The serial wire connection to the DAP requires that the scan chain is directly specified and the debug control unit is switched to serial wire mode. A chain with a single DAP and adaptive clocking is used:-

```
chain dev=ARMCS-DP clk=A
```

The target connection probe is then switched into Serial wire mode using box configuration items (see 4.23 below):-

```
cfb SWJEnable 1
cfb LVDSProbeMode 2
```

These configuration items will also ensure that the target is switched into SW mode.

3.1.5 Connect to theTarget DAP

Finally, the target DAP must be opened. This is achieved by using the devopen (dvo) command.
The parameter to this command is the index on the scan chain. The chain is zero indexed from the first device specified in the chain command. If auto-detecting, the indexes for the devices are printed as part of the result.

In the above examples the DAP is the first and only device on the scan chain – at index 0, so to open the DAP we use the command:-

```
dvo 0
```

All CSAT commands are now available for use – most will not work until a target device is open.

Advanced systems may have more than one DAP – each of these can be opened simultaneously using the 'dvo' command, however only one DAP remains "active" as the target of any subsequent commands.

3.1.6 Start-up Command file, minimum requirements

Any start up command file will be required to connect to debug unit, build a scan chain and connect to the device as a minimum requirement – otherwise any subsequent commands in the file will fail. A typical start-up file is:-

```
# a startup script
# A single line comment
echo Running my startup script

con TCP:Koyaanisqatsi   # an inline comment
chain dev=auto clk=A
dvo 0
```

Following this, additional commands could set up components in the system for use in subsequent command files or on the command line.

```
# component addresses
v7dbg.0 baseaddr 0x80001000
ptm.0   baseaddr 0x80002000
v7dbg.1 baseaddr 0x80004000
ptm.1   baseaddr 0x80005000
v7dbg.2 baseaddr 0x80010000
etm.0   baseaddr 0x80011000

# alias for easier reference
alias v7dbg.0 a9_0      # 1st A9 core
alias v7dbg.1 a9_1      # 2nd A9 core
alias v7dbg.2 r7        # R7 core

r7 rr didr        # read DIDRs  - check connection
a9_0 rr didr
a9_1 rr didr
```

3.1.7 Trace Capture Control

The DSTREAM / RVT2 trace capture units can be controlled using the **`trace`** command. This command has a number of functions, such as **`open`**, **`start`**, **`stop`**, **`read`** and **`close`**, which are used to initiate tracing and extract trace data from the unit. Previously captured trace using DS-5 can be extracted from the trace buffer using CSAT.

Trace control of the VSTREAM client is achieved using the optional VSTREAM command set. This must be loaded at CSAT start-up. This has similar control functionality to the trace commands for DSTREAM. See the VSTREAM client user guide for further information.

3.1.8 CSAT Exit Codes

CSAT can return a number of codes on exit.

Code	Description
-1	Internal C error. Memory allocation error or some other system issue.
0	Normal Program termination
1	Batch command execution error forced exit. Occurs if a command from batch file is executed and results in an error, and the **`batch_on_error`** option is set to **`exit.`** See (4.19)
2	A start-up batch file could not be found, or contained a syntax error on parsing. No commands run, CSAT exits with this code.

3.2 Command Summary

The following tables summarize the commands available within CSAT. These are grouped by functionality.

3.2.1 Connection and Configuration Commands

Command	Short Name	Description	Ref
rviscan	**rvs**	Scan for Debug Control Units (DSTREAM, VSTREAM, RVI)	4.1, p17
connect	**con**	Connect to a debug control unit via TCP or USB.	4.2, p17
options	**opt**	Set or print default CSAT options.	4.3, p17
chain	**chn**	Set or auto-detect scan chain and set JTAG clock frequency.	4.4, p19
cfgbox	**cfb**	Configure the Debug Control Unit.	4.5, p21
devopen	**dvo**	Open a connection to a target device on the scan chain.	4.6, p21
dap_chain_index	**dix**	Set the active DAP on the scanchain as target for subsequent commands.	4.7, p21
dapenum	**dpe**	Enumerate the AP modules on the DAP.	4.8, p21
cfgtplate	**cfg**	Configure the target device handler template.	4.13, p22
devclose	**dvc**	Close the connection to a target device.	4.9, p22
disconnect	**dcn**	Disconnect from Debug Control Unit. (closes all target connections)	4.10, p22
exit	**---**	Exit from CSAT.(closes target connections and disconnect from debug control unit)	4.11, p22

3.2.2 Register Commands

Command	Short Name	Description	Ref
dpregread	**drr**	Read a DAP register (AP or DP register)	4.14, p24
dpregwrite	**drw**	Write a DAP register (AP or DP register)	4.14, p24

3.2.3 Memory Commands

Command	Short Name	Description	Ref
dpmemread	**dmr**	Read memory using the DAP via an AP or arch v7 core.	4.15, p25
dpmemwrite	**dmw**	Write memory using the DAP via an AP or arch v7 core.	4.16, p27
dpmemfill	**dmf**	Fill memory using the DAP via an AP or arch v7 core.	4.17, p28
dpfload	**dfl**	Load memory from a file (bin or elf) via an AP or arch v7 core.	4.18, p28
dpfsave	**dfs**	Save memory data to binary file via an AP or arch v7 core.	4.19, p29
axictrl	**axi**	Set or display AXI-AP default control values.	4.15.3.1, p26

3.2.4 CoreSight Component Commands

CoreSight component commands allow components to be manipulated by component and register name, rather than memory operation, allowing easier use of the CSAT tool, and more readable command files.

The available CoreSight components are defined by reading in the **`cscompdata.dat`** file at CSAT start-up.

Command	Description	Ref
cscomp	Set the CS register component default parameters. List the available components.	5.2, p33
alias	Set a string to alias a CS component instance.	5.6, p40

Component commands have the form:-
`<component_instance | alias_name> <sub_command> <parameters>*`

Components come in two classes:

- CoreSight register components – these have a common set of sub commands
- Extended CS Components – these have enhanced component specific sub commands in addition to the normal register component commands.

See section 5 for a reference for these commands.

3.2.5 Miscellaneous Commands

Command	Short Name	Description	Ref
log	**---**	Control command logging.	4.12, p22

echo	**---**	Echo a string (in command file).	4.21, p30
reset	**rst**	Reset the system, JTAG TAP controller or via control register.	4.20, p29
batch	**---**	Run a CSAT command file.	4.23, p30
system	**:**	Run an external system command.	4.22, p30

3.2.6 Trace Commands

Trace control commands are used to control DSTREAM / RVT2 trace capture unit, and upload trace after capture. Trace buffer upload can also take place from CSAT if another tool has been used to enable and control trace capture.

Trace commands are of the form:-

```
<trace_prefix>  <trace_command> [<parameters>]*
```

The trace prefix is either '**`trace`**' or the short form '**`trc`**'. See 4.24 - Trace Commands.

The table shows the trace commands available:-

Trace Command	Description	Ref
help	Print a list of available trace commands.	6, p41
open	Open a trace connection – initialise the trace command handler.	6.2, p41
identify	Extract version information from trace capture unit.	6.4, p42
close	Close the trace connection.	6.3, p42
getconfigitems	Get supported trace configuration items.	6.13, p45
getconfigitem	Get the value of a trace configuration item.	6.13, p45
describemode	Describe trace mode.	6.12, p44
getmodes	Get available trace operational modes.	6.12, p44
getmode	Get current trace operational mode.	6.12, p44
getstate	Get Current capture unit state.	6.5, p42
setmode	Set current trace operational mode.	6.12, p44
setconfigitem	Set value of a configuration item.	6.13, p45
commitconfig	Commit current configuration to trace capture unit.	6.13, p45
async	Switch async monitoring on or off.	6.14, p48
start	Start trace capture.	6.7, p43
stop	Stop trace capture.	6.8, p43
reset	Reset trace capture functionality.	6.6, p43
read	Read some trace data from capture buffer	6.9, p43
readusb	Read some trace data from capture buffer via USB for faster upload.	6.9.2, p44
dumpbuffer	Write text file with capture buffer contents.	6.10, p44
dumpbin	Write capture buffer contents to binary file.	6.11, p44

3.2.7 Deprecated Commands

Command	Short Name	Description	Ref
runtest	**rnt**	Run a test on the system using ARM validation IP (not present in a released product).	4.26, p32
abort	**---**	Abort current command.	4.25, p32
abortall	**---**	Abort all commands.	4.25, p32

3.2.8 Command Availability

Most commands require a connection to a DAP device, otherwise an error will occur. During the connection sequence, the availability of commands is described below:-

Commands	Availability
rviscan, log, connect, exit	Anytime
chain, options, cfgbox, devopen, trace open, disconnect	After **`connect`**
All other device commands	After **`devopen`**
All other trace commands	Atter **`trace open`**

4 CSAT COMMAND REFERENCE

4.1 Scan for Debug Control Units

`rviscan (rvs)`

Scans for RVI, DSTREAM and VSTREAM debug control units on the network and connected via USB. Scans for 10 seconds.

```
%>rviscan
RVIScan started...
>> TCP:arm03965xp          10.33.0.118       VSTREAM ; 0 active connections
>> TCP:DS-noisyfan         10.33.0.108       DSTREAM ; 0 active connections
>> TCP:DS0000000556        10.33.0.211       DSTREAM ; 0 active connections
>> TCP:DS-Haystacks        10.33.0.161       DSTREAM ; 0 active connections
>> TCP:rvi-dave            10.33.0.226       RVI     ; 0 active connections
>> TCP:DS-Daedalus         10.33.0.150       DSTREAM ; 2 active connections
>> TCP:DS-Enterprise       10.33.0.179       DSTREAM ; 0 active connections
RVIScan : Scan complete.
%>
```

Note: The scan tool only searches for run control units that are connected to your local network, so units that are connected to separate subnets will not be found. Consequently, if you want to connect to an debug control unit on a separate network, you must ensure that you know the IP address of the unit on that network.

4.2 Connect to Hardware Debug unit

`connect (con) TCP:<hostname> | TCP:<ip address> | USB | USB:<serial_no>`

Connect to a debug contol unit. Specify TCP or USB. USB attaches to the first USB DSTREAM/RVI found on the local system, USB:<serial_no> attaches to a USB unit with the given number. The TCP option can specify either the hostname or the IP address of your debug control unit.

The VSTREAM client cannot be connected over USB, but TCP:localhost may be used if the VSTREAM client and CSAT are running on same system.

e.g:-

```
con TCP:MIKEL
con TCP:192.168.23.172
con USB
con USB:109330168
con TCP:localhost
```

4.3 Options Command

`options (opt) [<option_name> <option_value>]`

The option command allows default parameters to be changed from the initial program defaults to user defined defaults. The <option_name> parameter is the name of the option to set. The <option_value> is dependent on the option being set. Running the options command with no name displays all the current options values.

For example:

```
%>opt
Current Default Option Settings:-
```

```
memfile = MEMDMP.BIN
tracefile = TRACEDMP.TXT
tracebinfile = TRCDMPBIN.TRB
memprot = 0x43
autopowerup = true
test_timeout = 0x000493E0
test_baseaddr = 0x12004000
test_termval = 0x00000004
test_emptyval = 0xF1F00000
test_apidx = 0x00
test_resetlo_ondone = false
batch_on_error = halt
no_async_trace_msgs = 0x00
no_async_rvi_msgs = 0x00
retry_conn = 1
use_dl_fill = true
```

4.3.1 memfile option

The `memfile` option defines the default file used when loading and saving binary memory files. See **`dpfload`** and **`dpfsave`** commands.

4.3.2 tracefile and tracebinfile options

The `tracefile` option defines the default file used when dumping trace data to a text file on the host system. See the **`trace dumpbuffer`** command. The `tracebinfile` option defines the default filename used when dumping a binary trace data file with the **`trace dumpbin`** command.

4.3.3 memprot option

The `memprot` option, is used when accessing memory through a memory bus connected to a Mem-AP – such as the AHB-AP or APB-AP. These AP types have access control bits as the most significant byte of the AP Control – Status Word (CSW) register. The `memprot` option value is an 8 bit value which is used by memory commands using a Mem-AP to set the MS-byte of the AP CSW register. Dependent on the AP in use, only certain bits in this value are significant. See the CoreSight documentation for further details.

The `memprot` value defaults to 0x43, which is the default value for the MS byte in the AHB-AP CSW register.

All memory commands (**`dfl, dfs, dmr, dmw`**) use the `memprot` value when accessing memory.

4.3.4 autopowerup option

The `autopowerup` option controls if the ARMCS-DP handler attempts to use the power control bits in the DPACC.CSW register to power up the system when the device open (**dvo**) command is used, and power down the system on device close (**dvc**). Defaults to true. Set to false ***before*** connect to prevent any register operations on the DAP during connect.

4.3.5 batch_on_error Option

This option controls the action of CSAT when an error occurs executing a set of commands from a batch file, either the start-up batch file, or as a result of a batch command. There are three possible values for the batch_on_error option

`halt` - this is the default. Results in all remaining commands from the batch file being cancelled and control returns to the command line.

`cont` - The error is ignored. The remaining commands are executed.

`exit` - All remaining commands are cancelled and CSAT terminates as though an exit command had been executed.

```
%> options batch_on_error exit
```

4.3.6 no_async_trace_msgs and no_async_rvi_msgs Options

These options suppress the asynchronous messages that appear from the trace capture and debug control units respectively.

Useful during test runs, as the timing of the appearance of these messages can be unpredictable, and result in a different text log.

4.3.7 retry_conn Option

Determines how many times CSAT attempts to connect to the requested target.

4.3.8 use_dl_fill Option

The file load operations can use template download and fill commands to increase efficiency of download operations – on version 3.3.x firmware and above. This option available on **CSAT 1.2.0** and above.

If connected to an older firmware version, then this option automatically sets itself to false as soon as an invalid download or fill command is attempted and revert to the older file load mechanism.

4.3.9 test_* options (deprecated)

The test_* options control the default values and operation of the 'runtest' command. See this command for more information.

`test_timeout` - the timeout before the runtest command aborts.

```
%> options test_timeout 300000
```

`test_baseaddr` - the address of the Tube FIFO register to monitor

```
%> options test_baseaddr 0x12004000
```

`test_apidx` - the AP index of the DAP AP that the Tube FIFO is attached to.

```
%> options test_apidx 0
```

`test_termval` - FIFO read value runtest uses to determine test is complete

```
%> options test_termval 0x00000004
```

`test_emptyval` - FIFO read value runtest uses to determine FIFO is empty

```
%> options test_emptyval 0xF1F00000
```

`test_resetlo_ondone` - controls if reset is asserted when test completes.

```
%> options test_resetlo_ondone false
```

4.4 Set JTAG Scan Chain

```
chain (chn) [dev=auto | dev=? | dev=DEVICE_NAME{,DEVICE_NAME}*]  [clk=<FreqHz> |
clk=A]
```

`dev=auto` auto-detects the scan chain.

`dev=?` displays the current setup of the chain.

`dev=DEVICE_NAME{,DEVICE_NAME}*` directly sets the scan chain.

`clk` sets the JTAG clock speed, either a frequency in Hz, for example `clk=10000000` for 10MHz, or A for adaptive clocking.

This command sets up the scan chain. This will either be a JTAG scan chain with one or more devices or a single device serial wire when connecting to DSTREAM. The scan chain is emulated as a JTAG scan chain with VSTREAM.

Each DEVICE_NAME represents a component on the scan chain; the DEVICE_NAME is the name of the component handler in the DSTREAM or VSTREAM firmware. These component handlers are also referred to as “templates”.

The name used for the DAP component handler is ARMCS-DP. This will appear once in the scan chain for each DAP that appears in the system.

This setup may not be required if connecting CSAT as a second connection where another client such as DS-5 is currently connected.

Auto-detected Scan Chain

This example auto-detects an adaptively clocked system:-

```
chain dev=auto clk=A
```

A typical response to this is:

```
Jtag clock set to 50000000A
ID:0 ARMCS-DP
```

This shows a single DAP devices on the scan chain, a DAP (ARMCS-DP) at scan chain ID 0.

If there are other devices on the scan chain that the auto-detect algorithm recognises then these may appear in the list:-

```
Jtag clock set to 50000000A
ID:0 ARMCS-DP
ID:1 ARM1176JSF
```

Directly Specified Scan Chain

It is important to specify all the devices on the JTAG scan chain. Devices that are not CoreSight DAP may be specified using the UNKNOWN_<X> syntax as a placeholder. This allows the debug control unit to put the unknown item into bypass, while accessing the known items correctly.

Alternatively a known device may be specified as part of the chain. See 8.1 for a list of known devices that may be used or may appear on auto-detect.

Set a known JTAG chain, at 20MHz JTAG clock, with a CoreSight DAP and an unknown device, IR length 4.

e.g. **`%> chain dev=ARMCS-DP,UNKNOWN_4 clk=2000000`**

The next example sets the chain for a system that cannot be auto detected, and set the JTAG clock to 10MHz. This sets two target devices, an ARMCS-DP device at scan chain position (ID) 0, and an unknown device at position 1.

```
chain dev=ARMCS-DP,UNKNOWN_4 clk=10000000
```

The UNKNOWN_<X> device defines the JTAG IR length on the scan chain.

Multiple DAPs and unknown devices may be specified on the scan chain.

Note: CoreSight components and Cortex cores will never appear in the scan chain set up by the auto-detect or directly specified scan chain commands. These are attached to the CoreSight DAP.

Existing Scan Chain

An existing scan chain is discovered by using the **`dev=?`** as the command option. In this case it is possible that CoreSight components may appear as part of the scan chain, associated with an ARMCS-DP component. This is an extended scan chain set up by the DS-5 debugger. CSAT must be connected only to the ARMCS-DP components in this scan chain.

4.5 Configure Debug Control Unit

`cfgbox (cfb) <item_name> [<item_value>]`

The debug control unit has a number of configurable items which affect the operation of the unit and certain functions such as reset. Most of these do not affect the operation of CSAT, other than the reset timing parameters which are used when system reset is requested.

A list of some of the configurable items which could be used in CSAT is shown below:

Name	Description
ResetOperation	Overridden by the CSAT reset command
ResetHoldTime	Time in milliseconds system reset is asserted (if complete cycle)
PostResetDelay	Time in milliseconds before further operations on template after reset.
LVDSProbeMode	Alter probe to switch to SW connection mode.
SWJEnable	Switch target into SW connection mode.

A full list of available configuration items for the debug control unit can be obtained by reading the special “CONFIG_ITEMS” configuration item:-

```
%> cfb CONFIG_ITEMS
```

lists all the box items. See the configuration tools documentation (Ref 3) for further information.

4.6 Target Device open

`devopen (dvo) <devid>`

Open a DAP device on the scan chain. The `<devid>` is the ID displayed by the chain command, or the zero based index of the device in a directly specified chain. Only DAP devices can be opened. Multiple DAP devices may be opened, but only one device will be the active device as a target for commands. When opening multiple devices the last opened device will be the active device. See the **`dap_chain_index`** command (4.6) to control the active DAP device.

e.g. **`%> dvo 0`**

4.7 Target Device – Select Active DAP Chain Index

`dap_chain_index (dix) <devid>`

Select the currently active DAP device. <devid> is the scan chain. All subsequent commands will be sent to the active device, unless a CoreSight component has been tied to a specific DAP index.

4.8 Enumerate DAP

`dapenum (dpe)`

Scans all AP locations and build list of available APs on the active DAP. Numbers shown are the AP index numbers to use in commands which require an AP index.

```
%>dapenum
Enumerated 3 APs
  0 : AHB-AP
  1 : APB-AP
  2 : JTAG-AP
```

4.9 Target Device close

`devclose (dvc) [<devid>]`

Close the specified device if <devid> provided, otherwise close the currently active DAP device.

4.10 Disconnect

`disconnect (dcn)`

Disconnect from RVI. Executes **`devclose`** if currently attached to a device.

4.11 Exit

`exit`

Close CSAT program. Executes **`disconnect`** if still connected to a debug control unit, and also closes any trace connection.

4.12 Logging

`log  on | off | <filename>`

Logging is on by default, and all commands are logged to CSAT.LOG.

Logging can be turned on or off, or a new log file name for the session may be specified as <filename>.

CSAT.LOG is overwritten each time CSAT is started.

4.13 Configure ARMCS-DP Component Handler

`cfgtplate (cfg) <item_name> [<item_value>]`

The component handler “template” – ARMCS-DP – used with the DAP component may require additional configuration before or during use. This command enables that configuration.

`<item_name>` is the name of the configuration item.

`<item_value>` is the string value. If <item_value> is omitted then the current value for the item is displayed. The **`cfg`** command operates on the active DAP connected using the **`dvo`** command. Where there are multiple DAPs in a system, each DAP instance has a separate attached ARMCS-DP component handler instance with a set of configuration items. Configuration must occur for each instance in a multi-DAP system.

A full list of available configuration items for the template can be obtained by reading the special “CONFIG_ITEMS” configuration item:-

```
%> cfg CONFIG_ITEMS
```

lists all the template configuration items. See the configuration tool documentation (Ref 3) for further information.

4.13.1 Configuration Items for the ARMCS-DP template

There are three configuration items that may be of use when configuring the ARMCS-DP component for use with CSAT. These are RESET_REG_DATA, IGNORE_BUS_BUSY and USE_TB_RST_OFFSETS.

The remaining items are used by other commands or not appropriate for alteration using CSAT and must not be altered. The list below shows the items at time of writing this manual, but it is possible that these could be different in later versions of the DSTREAM firmware:-
DCOMMS_CHANNEL_ID, MEMORY_ACCESS_AP, DAP_CONFIG_INFO, V7DBG_CONFIG, DAP_POWER_UP, IGNORE_POWERUP_FAIL and AXI_CTRL_DEFS

RESET_REG_DATA
This sets the parameters for the reset register should such an element present on the system. The value is a string of comma separated values of the format:

```
<ap_no>,<base_addr>,<assert_value>,<deassert_value>,<mask>,<hprot_rst_regs>
```

All values are in hex.
`<ap_no>` = AP number for the memory mapped reset register logic.
`<base_addr>` = 32 bit base address value for register set containing the reset register.
`<assert_value>` = 32 bit data value when asserting reset.
`<deassert_value>` = 32 bit value when de-asserting reset.
`<mask>` = 32 bit mask of bits not to be altered in the register. (set bit to 1 = do not allow change to bit value.).
<hprot_rst_regs> = 8 bit value for the HProt value used when accessing the reset register values.

e.g. read then set the parameters:

```
%>cfg RESET_REG_DATA 1,80003094,4,0,4,80
Configuration item : RESET_REG_DATA set.
%>cfg RESET_REG_DATA
Configuration item : RESET_REG_DATA is : 01,80003094,00000004,00000000,00000004,80
```

If only the first n parameters are supplied when setting, the rest remain the same:

```
%>cfg RESET_REG_DATA 1,80003098
Configuration item : RESET_REG_DATA set.
%>cfg RESET_REG_DATA
Configuration item : RESET_REG_DATA is : 01,80003098,00000004,00000000,00000004,80
```

IGNORE_BUS_BUSY
The template checks the AP status register bits (6 & 7) to establish if a transaction is in progress, and that the bus is enabled, before undertaking a memory transaction across the bus connected to the AP. If a transaction is in progress (bit 7 = 0) or the AP is disabled (bit 6 = 0) then the operation is rejected. This is seen in CSAT as an error in a memory command. This configuration item when set to ‘1’ forces the memory operations to ignore the status of these bits. The example sets the configuration item.
e.g. `%> cfg IGNORE_BUS_BUSY 1`
The item is normally set to ‘0’

USE_TB_RST_OFFSETS (deprecated: only if ARM validation IP is present on prototype system)

These are parameters for the Validation Trickbox reset scheduler (only applicable if the trickbox IP has been built into the system).

e.g.
```
%>cfg USE_TB_RST_OFFSETS
Configuration item : USE_TB_RST_OFFSETS is : 01,00000520,00000018
```

<use_tb_sched> = 1 if trickbox reset scheduling in use, 0 if not.

<reset_reg_offset> = 0x520, offset from <base_addr> for the reset register.

<reset_sched_offset> = 0x018, offset from < base_addr> for the schedule register.

By default, the DAP template implements a **`ctrl`** register reset (see 4.21) assuming a validation configuration with a reset control register at 0x520 offset from the base address of the register set, and a reset request register 0x018 offset from the base. This is the configuration of the current validation trickbox.

If reset without the trickbox is required, or offset values are different then this config item should be changed. To disable the trickbox reset and use a normal register reset:
```
%>cfg USE_TB_RST_OFFSETS 0
Configuration item : USE_TB_RST_OFFSETS set.
```

The remaining config items are used as part of other CSAT commands or are not relevant when connected with CSAT and must not be altered:-

DCOMMS_CHANNEL_ID, MEMORY_ACCESS_AP, DAP_CONFIG_INFO, V7DBG_CONFIG, DAP_POWER_UP and IGNORE_POWERUP_FAIL.

These items exist at the time of writing, but may vary with versions of debug control unit.

4.14 DAP register read and write

```
dpregread (drr) <base name>.<index/name>
dpregwrite (drw) <base name>.<index/name> <data>
```

Read or write a DAP register. Register names are based on the names used in the CoreSight Architecture Manual. See also 8.2 - DAP Register Programmers Model Diagram.

DP Registers

Base name is **`DPACC`** or **`DP`**. Valid indexes are:-

`CSW` or **`1`**	: control status word.
`ADDR` or 2	: AP select address register.
`RDBUFF` or **`3`**	: Read buffer.

E.g:-
```
drr dp.csw             # read the DP control status word
drr dp.1               # read the DP control status word
drr dp.addr            # read the DP AP select address registers
drw dp.addr 0x010000F0  # select the AP at AP index 0x01, AP register block 0xF0.
```

AP Registers

Base name is APACC or AP. For a simple access the indexes 0 – 3 are valid.

<ins>*Simple access*</ins> requires selecting the AP index and register block using the dp.addr register then reading the relevant AP register.

e.g.
```
drw dp.addr 0x01000000  # select AP at AP index 0x01, AP register block 0x00
```

```
drr ap.0                 # read the AP control status register. (@ ap index 0x1)
drw ap.1                 # write the AP transfer address register.
drw dp.addr 0x00000000  # select AP at AP index 0x00, AP register block 0x00
drr ap.0                 # read the AP control status register. (@ ap index 0x0)
```

Enhanced access is available for the 1st eight registers in any of the 256 possible APs. This access type sets the **dp.addr** automatically according to the AP base name used.

The enhanced basename command has the form:-

<register_op> AP<n>.<index/name> [<data>]

In this case <n> represents AP index 0 – 255.

The valid names and indexes are:-

Name	Index	Description
CSW	0	Control Status Word
TAR	1	Transfer Address Register (bits 31:0)
TARH	2	Transfer Address Register (bits 63:32) – 64 bit addressing AP only such as the AXI-AP
DRW	3	Data Read / Write
BD0	4	Banked Data Read/Write 0
BD1	5	Banked Data Read/Write 1
BD2	6	Banked Data Read/Write 2
BD3	7	Banked Data Read/Write 3

e.g.

```
drr ap1.csw # read the AP control status register. (@ ap index 0x1)
drr ap0.csw # read the AP control status register. (@ ap index 0x0)
```

After an enhanced AP access command the `dp.addr` register will remain at the value set for the command.

Register operations are combined to access memory locations on the memory bus attached to the AP.

e.g.

```
drr AP0.CSW
drw ap0.csw 0x43800022        # set the CSW for 32 bit incrementing access
drw AP0.TAR 0x8100            # set the access address to 0x8100
drw AP0.DRW 0x12345678        # write 0x12345678 @ 0x8100
drw ap0.drw 0x87654321        # write 0x87654321 @ 0x8104
```

4.15 DAP memory read

dpmemread (dmr) <ap_no> {L|l:}<address>[.{b|B|h|H|l|L}] <no. items>

Read some memory.

<ap_no> is the AP index (0 – 255), or is set to **v7dbg** to access memory via the currently active v7dbg core. See the **v7dbg** command and CoreSight component commands for options to configure this support.

{L|l:}<address>[.{b|B|h|H|l|L}] is the address to access, along with data access size and address size information.

`<no. items>` defines the number of elements of the given size.

4.15.1 Memory Command Address Syntax

`{L|l:}<address>[.{b|B|h|H|l|L}]`

The prefix L: or l: is used to indicate a 64 bit address. The top 32 bits of the supplied address will be written to the TARH AP register before the rest of the memory operation is completed. This value will remain in the TARH for all subsequent memory operations on the AP. 64 bit addressing is not possible if using the v7dbg option for the <ap_no>

Access size defaults to word access size (32 bit). A suffix of .b, .B, selected byte access, .h or .H chooses half word access size and .l or .L chooses 64 bit access size. Addresses must be aligned to element access size.

Note: if a long address is used for a Mem-AP that does not support 64 bit addressing then the memory operation will occur on the 32 bit address only.

4.15.2 64 bit access limitation – 32 bit boundary wrapping.

The 64 addressing in CSAT wraps on the 32 bit boundaries correctly, as the operation for the high word of the address is independent of the low 32 bits.

`DMR 0 L:0x1000FFFFFFFC 4` will read 4 words from:

`0x1000FFFFFFFC, 0x100000000000, 0x100000000004, 0x100000000008.`

4.15.3 Additional Memory access configuration.

For any AP indexed memory access (not v7dbg) the current **`options memprot`** value will be applied to the most significant byte of the AP Control Status Word. This affects the type and privilege of the memory access. See the DAP AP documentation for more information on valid values.

If the indexed AP is an AXI AP then the current **`axi`** command values can set additional bits in the control status world

4.15.3.1 AXI AP configuration command

`axi [domain <val>] [ACE enable|disable]`

Domain is a 2 bit value occupying bits [14:13] in the AXI-AP CSW.
ACE enable is a 1 bit value occupying bit [12] in the AXI-AP CSW.

Run `axi` with no parameters to get current values.
Set values by using either or both optional keywords

e.g.

```
axi domain 0x2
axi ace enable
axi ace disable domain 0x3
```

AXI values will not be applied to AP types that are not AXI-AP.

4.15.4 Memory Read Examples

Example reads 4 words from address location 0x8100.

e.g. **%> dmr 0 0x8100 4**

```
0x00008100 : 0x12601E30 0x88E0A9CA 0x1AC23260 0x34A6FB79
```

Example reads 7 bytes from address location 0x9101.

Note: CSAT always displays from word aligned addresses, so the location not read is shown as ‘----‘.

e.g. **%> dmr 0 0x9101.B 7**

```
0x00009100 : ---- 0x1E 0xEF 0xB2 0x52 0x87 0x4E 0x49
```

Example reads 3 half words from address location 0x8042 using the v7 debug core.

e.g. **%> dmr v7dbg 0x8042.h 3**

```
0x00008040 : ------ 0xF8A5 0xAE3D 0xA251
```

Example to configure and use AXI-AP (assuming @ AP index 0x2) for memory reads

```
options memprot 0x46          # set AXI-AP appropriate CSW MSB value
axi domain  0x3               # set AXI domain value
DMR 2 L:0x100020003000 4      # high word TAR = 0x1000
DMR 2 0x4010.l 2              # high word TAR unchanged @ 0x1000
```

This reads 4 x 32 bit words from 64 bit address 0x100020003000, and the 2 x 64 bit words from 64 bit address 0x100000004010.

4.16 DAP memory write

dpmemwrite (dmw) <ap_no> {L|l:}<address>[.{b|B|h|H|l|L}] <data> [<data>]*

Write some memory.

<ap_no> and <address> syntax, and any additional Mem-AP configuration as described above in the DAP memory read command (4.15, 4.15.1, 4.15.2 and 4.15.2.1).

<data> in decimal or 0x hex format. Access size defaults to 32 bit word. This can be set to byte, halfword or 64bit long word accesses using the suffixes on the address value.

Data is interpreted according to access size. **0x00** is a single byte, half word, word or long word. The value **0x123456** is interpreted as 1 word / long word **0x00123456**, 2 half words **0x3456**, **0x0012** or 3 bytes, **0x56, 0x34, 0x12**.

v7dbg routes the memory command via the v7 debug support on the core. See the **v7dbg** command for options to configure this support. This is limited to 32 bit or smaller accesses only.

4.16.1 DAP Write Examples

Example writes 4 x32 bit words, 0x00000001, 0x00003444, 0x00000004 and 0x00000007 to address location 0x8000.

%> dmw 0 0x8000 1 0x3444 4 7

The same data as bytes writes 5 bytes to 0x8000, 0x01, 0x44, 0x34, 0x04, 0x07.

```
%> dmw 0 0x8000.B 1 0x3444 4 7
```

Memory write via v7 debug:

```
%> dmw v7dbg 0x8000 0x12345678 0x87654321
```

Memory write of 2 x 64 bit values 0x0000000000000203 and 0x0000000000000233 to 0x022220202000.

```
%> DMW 0 L:0x22220202000.l 0x203 0x233
```

4.17 DAP Memory Fill

`dpmemfill (dmf) <ap_no> <address>[.{b|B|h|H}] <pattern> <iterations>`

This is a new command available with CSAT 1.2.x and RVI firmware 3.3.x. It is not available if the **use_dl_fill** option is set to false.

The command fills memory according to <ap_no> and <address>, defined as per the **`dpmemwrite`** command.

Pattern is a decimal or hex value of up to 32 bits, used according to the size parameter on the address. The iterations value determines how many times the pattern is repeated. The number of bytes actually filled depends on both the size and number of iterations.

e.g. %> **`dmf 1 0x0000 0xEAFFFFFE 8`**

fills the first 8 words at address 0x0000 with the pattern 0xEAFFFFFE.

e.g. **`%> dmf v7dbg 0x8000.b 0x123 7`**

fills the first 7 bytes at address 0x8000 with the pattern 0x23.

Note: The address command syntax does not directly support 64 bit AXI-AP addressing. This can be achieved by setting the AXI-AP TARH register before executing the command.

e.g. Fill of 16 words of 0x11223344 to 0x100080007000.

```
drw ap0.tarh 0x1000
dmf 0 0x80007000 0x11223344 16
```

4.18 DAP File load

`dpfload (dfl) <ap_no> [memlog] { elf [execload] } | { bin <addr> } [filename]`

Load either a binary or an elf file into the target memory.

<ap_no> is the AP index (0 – 255) used to access the memory, or can be v7dbg to route the memory access via v7 debug.

The ***elf*** or ***bin*** parameters control the type of file loaded. The elf file contains information on memory locations to load.

When loading a binary file, an <addr> parameter is supplied to define the start address for the load. If no filename is supplied then the default **`MEMDMP.BIN`** is used. This default filename can be altered using the **`options memfile`** command.

The ***memlog*** option is used to force CSAT to log the addresses that the load process used.

The ***execload*** option is used in the elf files to force the execute time addresses to be used for load, rather than the default load time addresses. Sometimes used if executable does not contain scatter-loading code.

`v7dbg` routes the memory command via the v7 debug support on the core. See the **`v7dbg`** command for options to configure this support.

e.g. **`%> dfl 0 bin 0x8000 mydata.bin`**

Load `mydata.bin` into the memory accessed by AP0, at location 0x8000.

e.g. **`%> dfl v7dbg elf myprog.axf`**
Load elf program `myprog.axf` into the memory accessed via v7 debug.

The memory access privileges may be affected by the `memprot` option. See the **`options memprot`** command.

Note: The address command syntax in this command does not directly support 64 bit AXI-AP addressing when loading a file to a binary location. This can be achieved by setting the AXI-AP TARH register before executing the command.

4.19 DAP File save

`dpfsave <ap_no> <addr> <nWords> [filename]`

Save a block of memory to a file. The memory and length are defined by the <ap_no> (AP index / `v7dbg`), <addr> (memory address) and <nWords> (number of 32bit words) parameters. If no filename is supplied then the default **`MEMDMP.BIN`** is used. This default filename can be altered using the **`options memfile`** command.
The memory access privileges may be affected by the `memprot` option. See the **`options memprot`** command.
`v7dbg` routes the memory command via the v7 debug support on the core. See the **`v7dbg`** command for options to configure this support.

Note: The address command syntax in this command does not directly support 64 bit AXI-AP addressing when saving a file data. This can be achieved by setting the AXI-AP TARH register before executing the command.

4.20 Reset Target

`reset (rst) {sys &| ctrl [lo | hi] } | { tap  &| dapabort }`

Reset the target system. Options for system, control register, tap or dapabort resets.

`sys`	Drive nSRST line on the JTAG connector. Can be used with hi, lo and ctrl.
`ctrl`	Use the configured reset control register. Can be used with hi, lo and ctrl.
`tap`	Drive the nTRST line on the JTAG connector. Can be used with dapabort.
`dapabort`	Set the DAP abort command. Can be used with tap.
`hi`	De-assert reset level. Can be used with sys and ctrl.
`lo`	Assert reset level. Can be used with sys and ctrl.

System reset can be timed, according to box settings. Alternatively, a system or control register reset level can be set. **`reset sys lo`** asserts system reset, **`reset sys hi`** de-asserts system reset.

If the DAP template is connected then the parameters supplied in the **`cfgtplate`** command described above is used to implement the reset if a control register reset is used. **`reset dapabort`** does not cause the nTRST line to be used on the tap controller, but forces the ABORT instruction code into the JTAG-DP IR register.

It is not possible to use the sys, ctrl options with the tap, dapabort options.

4.20.1 Control Register Reset

The following algorithm is used for the control register reset. <name> represent configuration parameter details. (see 4.22.1 for config parameter details).

Reset Assert():

```
rstregaddr = <base_addr>
if ( <use_tb_sched> = 1)
       rstregaddr  += < reset_reg_offset>
rstregval = read ( rstregaddr )
rstregval &= <mask>
rstregval |= (<assert_value> & ~<mask>)
write (rstregaddr, rstregval)
if ( <use_tb_sched> = 1)
       write (<base_addr> + <reset_sched_offset>, 0)
```

Reset De-assert():

```
rstregaddr = <base_addr>
if ( <use_tb_sched> = 1)
       rstregaddr  += < reset_reg_offset>
rstregval = read ( rstregaddr )
rstregval &= <mask>
rstregval |= (<deassert_value> & ~<mask>)
write (rstregaddr, rstregval)
```

4.21 Echo String

`echo <some string>`

Prints <some string> to the display. Useful in batch files.

4.22 System Command

CSAT can execute an operating system command using the **`system`** or **`:`** commands.

e.g. 'dir' command of local file system

```
%>: dir
 Volume in drive C is WINDOWS XP
 Volume Serial Number is A457-1CD4

 Directory of C:\ProgFiles\CSAT

20/12/2007  13:37    <DIR>          .
20/12/2007  13:37    <DIR>          ..
11/11/2005  11:57            40,960 boost_thread-vc71-mt-1_31.dll
20/12/2007  13:45           229,376 csat.exe
20/12/2007  14:12             4,221 CSAT.LOG
20/12/2007  11:25            53,248 rvicomms.dll
20/12/2007  14:06           217,088 rvt_cli_cmds.dll
25/07/2007  15:20             2,604 startup.txt
               6 File(s)        564,106 bytes
               3 Dir(s)  15,380,971,520 bytes free
%>
```

Note: results of these external commands do not appear in the CSAT log file.

4.23 Batch Commands

`batch <filename>`

Run a batch of commands from a file. File format is a text file with lines which can contain the following:

- Any valid command. Batch files cannot call the **`batch`** command.
- A blank line.
- A line with a # character. The # and anything after it on the line are treated as a comment. Comments can appear after commands, or on an otherwise blank line.

e.g. `%>` **`batch c:\startup.txt`**

Batch files are parsed completely to create a list of executable commands, which are only executed after the file processing is complete. If a file contains a syntax error, then none of the commands in the file are executed.

Once the batch file has been parsed and a command list created then the commands are executed. Action on errors during batch command execution are described below (see section 4.19.2).

4.23.1 Batch File as a start-up command line parameter

`csat` can be started with a batch command file as a parameter. This is a start-up batch file.

e.g. `%>` **`csat c:\startup.txt`**

This has the same effect as entering '`batch c:\startup.txt`' as the first command at the command prompt. If the startup.txt file cannot be found, or contains syntax errors then no commands are run, and CSAT exits with an error code.
Once the start-up batch file has been parsed and a command list created then the commands are executed. Action on errors during batch command execution are described below (see section 4.19.2).

4.23.2 Execution Errors when running Batch File Commands

The operation of CSAT when a command in a batch file fails with an error is controlled by the batch_on_error option. This is set as follows:

```
%> options batch_on_error exit
```

Possible values are:

`halt` - this is the default. Results in all remaining commands from the batch file being cancelled and control returns to the command line.

`cont` - The error is ignored. The remaining commands are executed.

`exit` - All remaining commands are cancelled and CSAT terminates as though an exit command had been executed with return code 1. (See 3.1.1 for CSAT return codes.)

4.24 Trace Commands

`trace (trc) <function> [options]`

This command has a set of functions used to control the the trace capture functionality in the DSTREAM or the RVT2 trace capture unit when attached to the RVI debug control unit. The functions available are described in the trace command reference below.
The command line

```
%> trace help
```

lists possible trace functions.

4.25 Abort Commands

```
abort
abortall
```

The **`abort`** and **`abortall`** commands cancel the current and current plus all remaining commands respectively. The current command is aborted only if it operates asynchronously. This currently applies only to the **`runtest`** command.

4.26 Runtest (deprecated)

```
runtest  (rnt) [timeout=<milliseconds>] [termval=<val>]  [apidx=<ap_no>]
[emptyval=<val>]  [baseaddr=<addr>]
```

Runtest has a specific set of operations, designed for validation of core on a test FPGA with appropriate additional validation hardware. It first de-asserts system reset, and then start monitoring the supplied address as the Tube FIFO. If a timeout occurs or the termval is detected, then the test command terminates, asserting system reset.

All parameters are optional, default values are used for any not supplied as follows:

```
apidx       = 0
termval     = 4
emptyval    = 0xF1F00000
timeout     = 300000
baseAddr    = 0x12500000
```

5 CORESIGHT COMPONENT COMMAND REFERENCE

5.1 Common Features

CSAT version 2.0.0 and above introduce a set of commands based on CoreSight components available on the debug bus. These commands simplify accessing and controlling CoreSight systems.

All CoreSight components supported have a set of standard register access sub-commands including those to read and write component registers, or set the component base address. Multiple instances of a component type are supported.

Component commands which have support for only the standard register sub-commands are referred to as register components. Some CoreSight component types have additional sub-commands, and are referred to as extended components.

The available CoreSight components are defined by reading in the **`cscompdata.dat`** file at CSAT start-up.

5.1.1 Component Command Format

Component commands take the form:-

`<name>.<n> <sub_command> [<options>*]`

<name> - The name of the component.

<n> - the component instance in the range 0 – 39. Components instances allow the user to support multiple components of the same type on more complex CoreSight systems. Using the command name without an instance suffix references instance .0 initially. This can be changed using the 'def' sub command.

`<sub_command>` - either one of the standard register sub-commands, or an extended sub-command in component commands that support this feature.

`<options>` - 0 or more sub command options.

5.2 cscomp Command.

```
cscomp
cscomp def_apidx <val>
cscomp def_apctrl <val>
```

The **`cscomp`** command controls the default settings for all the supported CoreSight component commands. Running the command without any options displays the current settings, and the available CoreSight components.

```
e.g. %>cscomp
cscomp : Settings:-
    def_apidx   0x01
    def_apctrl  0x82

cscomp : Available components:-
    cti
    etb
    etm
    itm
    pmua9
    ptm
    tfun
    tpiu
    v7dbg
```

The **`def_apidx`** option allows the setting of the default apidx for all the components, unless this has been overridden by the individual component setting. <val> should be in the range 0 – 255, but is typically 0 for a system with a DAPLite, or 1 for a system with a DAP.

The **`def_apctrl`** option sets the value used in bits 31:24 of the control-status register in the AP used to address the debug bus. For an APB-AP this should normally be set to 0x80.

5.3 Standard Register Sub-Commands

5.3.1 Supported CoreSight Components

Currently the following components support the standard register sub-commands:-

```
    cti,  etb, etm, etm4, itm, pmua9, ptm, tfun, tpiu, v7dbg, stm, cm3, cm3_dwt,
cm3_etm, cm3_tpiu, v8cs, v8cti, v8pmu.
```

The current list of supported components can be found using the **`cscomp`** command.

The following components support additional extended sub-commands:-

```
    etb, v7dbg.
```

The component data file **`cscompdata.dat`** defines all the supported components which have the standard set of register sub-commands. This file defines the register names for the component, the offset from the component base address, and the access permissions for the register. Users can add their own CoreSight component definitions to this file if desired. This file is read into CSAT when the program starts.

Extended commands for components are not defined in this file, but are built into the CSAT product.

5.3.2 View Current Settings

`<component>`

Type the name of the component instance to view the current settings for the component.

```
%>ptm
ptm.0 Settings:-
        DAP dev chain idx = 0
        baseaddr 0x80002000
        apidx    0x01
        apctrl   0x80
```

Extended components may show additional settings

```
%>v7dbg
v7dbg.0 Settings:-
        DAP dev chain idx = curr
        baseaddr 0x80001000
        apidx    0x01
        apctrl   0x80
        fastwordaccess   false
        ACTIVE
```

5.3.3 Set Base Address (baseaddr)

`<component> baseaddr <address_value>`

This command must be used first to ‘activate’ the component instance. No sub-commands that access registers associated with the component can be used until the component instance has had its base address on the debug bus defined. Base address value is supplied as a 32 bit hex number.

e.g. **`%> ptm.3 baseaddr 0x80104000`**

5.3.4 List Defined Registers (rlist)

`<component> rlist`

This command lists the registers and register groups defined for the component in the component data file. . Registers are grouped in the component data file broadly by function.

e.g **`%> tfun rlist`**

```
List of valid register groups and names:-
group name : csmgmt
        imcr,ctagsr,ctagcr,lar,lsr,authsr,
group name : devid
        devid,devtypeid,peripid4,peripid5,peripid6,peripid7,peripid0,peripid1,
        peripid2,peripid3,compid0,compid1,compid2,compid3,
group name : integ
        itatbdata0,itatbctr2,itatbctr1,itatbctr0,
group name : prog
        ctrl,pcr,
```

5.3.5 Set Instance as Default (def)

`<component> def`

This sets the defined instance as the one used for the component command when an instance suffix is not used.

e.g **`%> etm.5 def`**

Instance `etm.5` is used whenever the `etm` command is used. At startup, the default for each component is set to instance .0.

5.3.6 Register Read (rr)

`<component> rr <offset>|<reg_name>|<group_name>| all`

Component registers can be read by offset from the base address of the component, by register name, as a group of registers, or all registers in a component may be read. Registers are grouped in the component data file broadly by function.

`<offset>` - referencing the register by offset means that the read is always attempted, even if the register is write only. The offset reference can also be used to access locations that are not defined as register names. Offset in range 0x000 – 0xFFC.

`<reg_name>` - referencing the register by name reads the register at the offset defined for the name in the component data file. If the access for the register name is defined as write only then the read is not attempted.

`<group_name>` - all the registers in the group are read and displayed. If any register in the group is defined as write-only access then the read for that register is not attempted, and value marked as !!!!!!!!. The operation then continues with the next register in the group.

`<all>` - displays all defined registers with same semantics as using a group name.

e.g **`%> etm.1 rr 0x120`**

`%> ptm rr cr`

```
%> v7dbg rr devid
```

5.3.7 Register Write (rw)

`<component> rw <offset>|<reg_name> <value>`

Writes value to the register referenced by <offset> or <reg_name>.

`<offset>` - referencing the register by offset means that the write is always attempted, even if the register is read only. The offset reference can also be used to access locations that are not defined as register names. Offset in range 0x000 – 0xFFC.

`<reg_name>` - referencing the register by name writes the register at the offset defined for the name in the component data file. If the access for the register name is defined as read only then the write is not attempted.

e.g

```
%> etm.1 rw 0x020 0x012403404
%> ptm rw cr 0x0D400
```

5.3.8 Register Read-Modify-Write (rmw)

`<component> rmw <offset>|<reg_name> <reg_write_value> <mask>`

This command allows a register to be modified on a bitfield basis. The register is read, the mask and value used to determine which bits are altered. A ‘1’ in the mask value indicates a bit that is allowed to be changed, ‘0’ indicates a bit to remain unaltered. The following logic is used to write the register

```
reg_final_value := (reg_read_value & ~mask) |  (reg_write_value & mask)
```

`<offset>` - referencing the register by offset means that the write is always attempted, even if the register is read only. The offset reference can also be used to access locations that are not defined as register names. Offset in range 0x000 – 0xFFC.

`<reg_name>` - referencing the register by name writes the register at the offset defined for the name in the component data file. If the access for the register name is defined as read only then the write is not attempted.

e.g **`%> etm.1 rmw 0x020 0x002 0x0F`**

```
%> ptm rmw cr 0x400 0x400
```

5.3.9 Set AP Index for Component (apidx)

`<component> apidx <val>`

`<val>` in range 0 – 255.

Normally it is assumed that all CoreSight components are on a single debug bus, attached to an AP in the DAP, the AP index of which is defined in the `cscomp` command using the def_apidx command. If a design has components connected to a different AP, then this command can be used to override the global value.

5.3.10 Set DAP index for component (dix)

`<component> dix [<val> | curr]`

<val> is a valid DAP chain index.

This command can be used to tie a component instance to a specific DAP chain index. By default a component will execute register commands on the currently active DAP index. Use the **`curr`** option to restore this default operation.

When a command for a DAP tied component is executed, the DAP index is temporarily changed for the duration of the command, then restored to leave the global DAP chain index value unchanged.

e.g.

```
dvo 0                        # open DAP at index 0
dvo 1                        # open DAP at index 1 (this now global index)
dix 0                        # global index DAP 0
v7dbg.0 dix 1                # this core on DAP 1
v7dbg.1 baseaddr 0x80001000  # set core base address
v7dbg.0 baseaddr 0x80001000  # set core base address

v7dbg.0 rr didr        # read debug ID reg, DAP1, addr 0x80001000
v7dbg.1 rr didr        # read debug ID reg, DAP0, addr 0x80001000
```

Note: the DAP chain index is checked for validity and connection at time of register access command execution.

5.3.11 Command Help

`<component> help | ?`

The component commands can all print a summary of the available sub-commands and options. For component commands with extended sub-commands these are also listed.

```
%>etb help
etb <subcommand> [<options>]*
subcommands:-
        rr <offset>|<reg_name>|<group_name>|all  -  read register or group of
registers
        rw   <offset>|<reg_name> <value>  - write register by offset or name
        rmw  <offset>|<reg_name> <value> <mask>  -  read-modify-write register by
offset or name
        baseaddr <addr_val>   - set base address for this instance
        apidx  <val>   - override default AP idx for this instance
        rlist  - list valid group and register names
        def  - make this instance the default for unqualified name
        dumpbin [forceall] [<filename>] -  dump trace buffer to data file -
optionally supply filename, force all buffer download.
        read [<start_rec>] [<num_recs>] - display trace data from the etb buffer
        status - current state of etb trace buffer.
        start [<mode>] - start the etb tracing, optionally define mode.
        stop - stop the etb tracing.
```

5.4 Extended Commands : ETB component

5.4.1 ETB Current Status (status)

`etb status`

Displays information on the state of the trace buffer, and if the ETB is collecting trace or stopped. Displays details of status and FFSR regs, along with the number of records captured if stopped.

e.g.
```
%>etb status
etb.0 Status : Trace Disabled
    Buffer Size = 2048 records, 8192 Bytes
    STS Reg 0x0000000C:- FtEmpty:AcqComp:Not Triggered:Not Full
    FFSR Reg 0x2:- FtStopped:Not FlInProg

    1172 Trace Records Captured
%>
```

5.4.2 ETB Start Trace Collection (start)

`etb start [<mode>]`

This starts the ETB capturing trace. The default is to use normal mode if the `<mode>` option is omitted. At start of trace, the RAM write pointer (RWP) is set to 0 – the start of the ETB RAM buffer, and the mode bits in the formatter and flush control register (FFCR) are set to 'normal', or the supplied user mode. Other bits in this register are unaffected.

`<mode>` can by omitted or one of:-

`normal`	– normal mode for formatter.
`cont`	– continuous mode for formatter.
`bypass`	– formatter off.
`asis`	– leave mode as-is, do not alter the ffcr.

Note: if the mode bits in the FFCR are altered by this command, they are not restored on stopping trace collection.

5.4.3 ETB Stop Trace Collection (stop)

`etb stop`

Halts trace collection by the ETB.

5.4.4 ETB Read Trace Buffer (read)

`etb read <start_rec> <num_recs>`

Read and display on screen <num_rec> records from the trace buffer starting at <start_rec> offset from the oldest/first record in the buffer.

```
%>etb read 0 64
Rec(00000000) : 0x80000813 0xEC600000 0x7C8892D8 0x38223344
Rec(00000004) : 0x05F8A810 0x04220BCE 0x08C808E0 0x09C008E0
Rec(00000008) : 0xD008C808 0xC808C008 0xC008D008 0x0008C808
Rec(00000012) : 0x08C008D0 0x08D008C8 0x08C808C0 0x00C008D0
Rec(00000016) : 0xD008C808 0xC808C008 0xC008D008 0x0008C808
Rec(00000020) : 0x08C008D0 0x08D008C8 0x08C808C0 0x00C008D0
Rec(00000024) : 0xD008C808 0xC808C008 0xC008D008 0x0008C808
Rec(00000028) : 0x08C008D0 0x08D008C8 0x08C808C0 0x00C008D0
Rec(00000032) : 0x08C80813 0x08C008D0 0x08D008C8 0x01C808C0
Rec(00000036) : 0xC008D008 0xD008C808 0xC808C008 0x0008D008
Rec(00000040) : 0x08C808C0 0x08C008D0 0x08D008C8 0x00C808C0
Rec(00000044) : 0xC008D008 0xD008C808 0xC808C008 0x0008D008
Rec(00000048) : 0x08C808C0 0x08C008D0 0x08D008C8 0x00C808C0
Rec(00000052) : 0xC008D008 0xD008C808 0xC808C008 0x0008D008
Rec(00000056) : 0x08C808C0 0x08C008D0 0x08D008C8 0x00C808C0
Rec(00000060) : 0xC008D008 0xD008C808 0xC808C008 0x0008D008
%>
```

5.4.5 ETB Dump Trace Buffer to file (dumpbin)

`etb dumpbin [fsync] [forceall]  [<file_name>]`

This command dumps all the trace samples from the ETB buffer into a disk file, for offline analysis by a suitable tool.

`fsync` – pre-pend a tpiu frame sync packet to the data – this allows the trace analysis tool to auto-sync with the data in the buffer.

`forceall` – download the entire buffer, irrespective of how many records have been captured.

`<file_name>` - name of file for download. Default is **`etb_buffer.trb`**.

```
%>etb dumpbin fsync mya9_dump.trb

Uploading 1172 records from ETB memory buffer
Writing 1172 trace records to file mya9_dump.trb
```

5.5 Extended Commands: v7dbg component

The v7dbg component supports all the standard register sub commands and defines has 2 additional sub-commands that control the settings and operation of the component. These settings affect the dap memory access commands `dmr, dmw, dmf, dfl,` and `dfs`, when the **`v7dbg`** option is used as part of these commands.

The sub-commands are `active` and `fastwordaccess`.

Typing the command v7dbg command without any sub commands causes the current settings to be shown.

```
%>v7dbg
v7dbg.0 Settings:-
        baseaddr 0x80110000
        apidx    0x01
        apctrl   0x80
        fastwordaccess   true
        ACTIVE
```

By default the `v7dbg` command without a suffix refers to the `v7dbg.0` instance. The `apctrl` parameter is set by the `cscomp` command `def_apctrl`. (See section 5.2).

5.5.1 v7dbg – active

The 'active' setting determines which instance of v7dbg is used when memory accesses with a v7dbg option are executed. For example in the 3 core system described below:-

```
%> v7dbg.0
v7dbg.0 Settings:-
        baseaddr 0x80110000
        apidx    0x01
        apctrl   0x80
        fastwordaccess   true
        ACTIVE
%> v7dbg.1
v7dbg.1 Settings:-
        baseaddr 0x80112000
        apidx    0x01
        apctrl   0x80
        fastwordaccess   true
        Inactive (active instance : v7dbg.0)
%> v7dbg.2
v7dbg.2 Settings:-
        baseaddr 0x80114000
        apidx    0x01
        apctrl   0x80
        fastwordaccess   true
        Inactive (active instance : v7dbg.0)
```

The `v7dbg.0` instance is marked as the active instance. Thus if a memory read command such as '**`dmr v7dbg 0x1000 128`**' is issued, then the v7dbg.0 instance is used.

***Note**: the dmr command **cannot** use the v7dbg.<n> syntax to select an instance.*

5.5.2 v7dbg fastwordaccess

'**`fastwordaccess`**' can be true or false. By default this is set to true for using the maximum efficiency during memory operations. Only set this to false if a fault is suspected in the core debug hardware. **`fastwordaccess`** causes the DSCR DCC transfer mode bits to be set to 'fast' for the duration of the memory operation.

e.g. %> `v7dbg.5 active`

```
%> v7dbg.1 fastwordaccess false
```

5.5.3 v7dbg – previous command syntax

Prior to the introduction of the CoreSight components, v7dbg was a standalone command that set up the v7dbg options for the memory access commands using the v7dbg option. The previous syntax is still usable for legacy scripts and will operate of the default **`v7dbg.0`** instance.

`v7dbg [apidx=<N>] [baseaddr=<0xNNNN>] [fastwordaccess={true|false}]`

5.6 Alias Command

```
alias <command_name> <alias_name>
```

The alias command creates a new command in the command tree with the name <alias_name>. When this command is executed, the <alias_name> is substituted in the command line with <command_name>.

The alias command is not specifically a CoreSight component command, but is useful in the CoreSight context in describing multi-core systems, making scripts easier to read for the user. For example, if two v7dbg cores are in the system one A9 and one A5, the following commands could be used:-

```
v7dbg.0 baseaddr 0x80001000
v7dbg.1 baseaddr 0x80007000
alias v7dbg.0 a9
alias v7dbg.1 a5
```

Now,

```
A5 rr dscr
```

is interpreted as

```
v7dbg.1 rr dscr
```

6 TRACE COMMAND REFERENCE

6.1 Trace Support in CSAT

CSAT has a set of trace commands which allow the control of trace and extraction of trace data from the RealView Trace 2 (RVT2) trace capture unit and the DSTREAM debug and trace unit.

VSTREAM does not currently have a virtual trace capture capability.

'**`trace help`**' lists the available trace functions:

```
%>trace help
Help for RVT CLI
================

Command Set version : 2.0.1.0
List of Commands:-

async   Switch async monitoring on or off.
close   Close connection to RVT unit
commitconfig   Commit current configuration.
describemode   Describe trace mode.
dumpbin   Dump trace buffer to a binary file.
dumpbuffer   Dump trace buffer to a file.
getconfigitem   Get the value of a config item.
getconfigitems   Get supported configuration items.
getmode   Get current trace operational mode.
getmodes   Get available trace operational modes.
getstate   Get Current RVT state.
help   Print a list of available commands.
identify   Identify versions on attached RVT unit.
open   Open connection to RVT unit
read   Read data
readusb   Read data via usb
reset   Reset RVT.
setconfigitem   Set value of a configuration item.
setmode   Set current trace operational mode.
signals   Get trace signal settings from RVC file and store in configuration.
start   Start tracing.
stop   Stop tracing.
For individual command help type <cmd> help

%>
```

The 'help' command may be used at any time.

6.2 Trace Open

`trace open [TCP:<hostname> | TCP:<ip address> | USB]`

Open a connection to the RVT2 trace capture unit or the trace capture component within DSTREAM, using the same connection syntax as connecting to the debug control unit. The connection option can be omitted, in which case the same option as used to connect to the debug control unit is used. The open function must be used before any other of the trace functions except '**`trace async`**'.

e.g. **`%> trc open`**

Connect to the RVT2 unit attached to the currently connected RVI unit, or the trace components for the currently connected DSTREAM unit.

e.g. **`%> trace open TCP:192.168.23.123`**

Connect to RVT unit by TCP address.

Opening trace normally results in asynchronous notification messages being sent and displayed.

```
%>
Async Notification of RVT internal state change...
 TriggerState = Not Triggered
 BufferState = Trace Buffer Empty, Stopped

%>
Async Notification of RVT internal configuration change...
```

To suppress these messages both on open and later in the trace session, use the ‘**`trace async`**’ function.

6.3 Trace Close

`trace close`

This function closes the connection to the trace capture unit.

6.4 Trace Identify

`trace identify`

This obtains hardware and software version information from the attached trace capture unit.

e.g.

```
%>trace identify

RVT Product Name: RealView Trace
RVT Serial Number: 1064
Firmware revision: 0x03010000
Hardware revision: 0x02000201
PRB Serial Number: 0
PRB HW Rev: 0x00000000
FPGA Image Name: rvt2x.bin
FPGA Image Rev: 0x000004AD
%>
```

The MS byte of ‘hardware revision’ indicates the version of the trace capture unit – in the above example RVT2 (0x02……..).

6.5 Trace Getstate

`trace getstate`

This function gets the current trace state from the trace capture unit. This indicates if the unit is tracing or stopped, and the amount of records captured if stopped, along with the trigger position if triggered. The format differs between RVT2 and DSTREAM.

e.g. for RVT2

```
%>trace getstate
GetState Command Response

 Trigger State : Triggered
 Buffer State : Trace Buffer Empty, Stopped
 Current Trace Write Pointer : 74565
 Trigger Pointer : 0
```

e.g. for DSTREAM

```
Hardware is DSTREAM
Current DSTREAM state...
        lastFrameWritten = not set
        triggerFrame = not set
        status = 0x00006800

         BUFFER_FULL(0)       CLOCK_AT_EDGE(0)        USB_PATH_ERR(0)    INVAL_PTR_UPDATE(0)
          VALID_TPKT(0)             ALIGNED(0)       FIFO_OVERFLOW(0)         T_CLOCK_ERR(1)
               READY(1)      FIFO_UNDERFLOW(0)          FIFO_EMPTY(1)          RESYNC_ERR(0)
         TRIGGER_ERR(0)      DATA_IN_BUFFER(0)          TARGET_ERR(0)        USB_ACTIVITY(0)
       BUFFER_RD_ERR(0)          DONE_DELAY(0)                 ERR(0)      BUFFER_WRAPPED(0)
       TRACE_STOPPED(0)       TRIGGER_FOUND(0)
```

6.6 Trace Reset

`trace reset`

This function resets the trace capture unit.

6.7 Trace Start

`trace start`

This function starts the trace unit capturing trace.

6.8 Trace Stop

`trace stop`

This function stops the trace unit capturing trace.

6.9 Trace Read

`trace read <start index> <record count> [{extract [withts]} | raw]`

This function allows the display of a block of trace data from the trace capture unit. The command line above is the complete set of options, but differing options and combinations are valid for differing versions of trace capture unit.

Read commands are useful for checking for the presence of trace data, and examining small amounts, but the '**`dumpbuffer`**' and '**`dumpbin`**' trace commands are more useful for collecting all the data in a buffer for analysis by other tools.

6.9.1 RVT2 Read Commands

`trace read <start index> <record count> raw`

Only the read raw command is valid for RVT2 hardware.

With RVT2, when timestamps are not needed, some of the timestamp area is used to store trace data.

e.g.

Data generated on short run without timestamps

```
%>trace read 0 6 raw
TD00000000: 00080a4c 41000808 080a000a 00000000
TD00000001: 00080a00 0a00080a 080a0008 00000000
```

```
TD00000002: 00080a00 0a00080a 080a0008 00000000
TD00000003: 00080a00 0a00080a 080a0008 00000000
TD00000004: 00080a00 0a00080a 080a0008 00000000
TD00000005: 00080a00 0a00080a 080a0008 00000000
```

Data generated on same run with timestamps

```
%>trace read 0 14 raw
TD00000000: 00080a4c 00000808 007ae92e 00000000
TD00000001: 00080100 00000808 007ae960 00000000
TD00000002: 00080800 00000808 007ae992 00000000
TD00000003: 00080800 00000808 007ae9c4 00000000
TD00000004: 00080800 00000808 007ae9f6 00000000
TD00000005: 00080800 00000808 007aea28 00000000
TD00000006: 00080800 00000808 007aea5a 00000000
TD00000007: 00080800 00000808 007aea8c 00000000
TD00000008: 00080800 00000808 007aeabe 00000000
TD00000009: 00080800 00000808 007aeaf0 00000000
TD0000000a: 00080800 00000808 007aeb22 00000000
TD0000000b: 00080800 00000808 007aeb54 00000000
TD0000000c: 00080800 00000808 007aeb86 00000000
TD0000000d: 07080800 00000808 00dca8a1 00010000
```

6.9.2 DSTREAM Read Commands

DSTREAM can also read using the USB port to speed up data transfer.

`trace readusb <start index> <record count> raw`

6.10 Trace Dumpbuffer

`trace dumpbuffer <filename>`

This function dumps the entire contents of the trace buffer on the trace capture unit into the named file `<filename>` supplied. The default filename name ‘`TRACEDMP.TXT`’ is used if `<filename>` is omitted. This default can be altered using the **`options tracefile`** command.

6.11 Trace Dumpbin

`trace dumpbin <filename>`

Dumps the entire contents of the trace buffer into a binary format file.

The default filename name `‘TRCDMPBIN.TRB’` is used if `<filename>` is omitted. This default can be altered using the **`options tracebinfile`** command.

These files may be input by other tools to analyse the trace data collected.

6.12 Trace Unit Data Capture and Storage Modes

`trace describemode <mode name>` ***: get more info about mode***.

`trace getmode` ***: get current mode for unit***

`trace getmodes` ***: get list of available modes for the unit***

`trace setmode <mode name>` ***: set mode on unit***

Data capture modes may be described as ‘classic’ or ‘continuous’. Classic mode uses the trace TRACECTL pin, along with other pins to determine if a valid trace sample is being presented to the trace probe inputs. This mode may be used with single trace source targets, or CoreSight trace systems that use a TPIU either in Normal or Bypass modes. Continuous mode is designed exclusively to capture trace from a TPIU in Continuous mode.

Data storage modes describe what happens to the data once it has been captured. 'Store and forward' storage modes place trace data into RAM on the capture device until the trace operation is complete. A client can then access the data from the storage RAM.

CSAT can extract trace data from the store and forward modes using either the '**`trace read`**' or '**`trace dumpbuffer`**' and '**`trace dumpbin`**' commands.

The table below shows the names of the modes and the RVT/DSTREAM versions that support them.

Name	Description	RVT2	DSTREAM
`classic`	Classic store and forward mode	Y	Y
`continuous`	TPIU continuous mode, store and forward storage.	Y	Y

6.13 Trace Configuration Item Commands

```
trace getconfigitems
trace getconfigitem [-v] <config item name>
trace setconfigitem <config item name> <config item value>
trace commitconfig
```

The trace configuration items set up the trace capture unit to capture trace. There are a number of options, which depend on the capture and storage modes, and also the capabilities of the trace capture unit.

The '**`getconfigitems`**' command lists valid configuration items for the current mode and capture device.

The '**`getconfigitem`**' command gets the value, and if the '–v' option is used, additional information about a config item.

e.g.

```
%>trace getconfigitem -v ETM_PROTOCOL
Name    : ETM_PROTOCOL
Type    : TPA_TYPE_ENUM
Size    : 4
Max     : 0x00000003
Min     : 0x00000000
EnumStr:-
ETMv1
ETMv3
ETMv1tov3

DispStr: ETM Protocol
HelpStr: The ETM Protocol version
Avail  : true
RdOnly : false
Value: 0x00000001 {ETMv3}
```

This config item is an enum type, so has string and integer values that are equivalent.

It may be set using either value, using the '**`setconfigitem`**' command:

```
%>trace setconfigitem ETM_PROTOCOL 2
%>trace commitconfig
%>trace getconfigitem ETM_PROTOCOL
ETM_PROTOCOL=0x00000002 {ETMv1tov3}
%>trace setconfigitem ETM_PROTOCOL ETMv3
%>trace commitconfig
```

```
%>
```

Note: a ‘**`commitconfig`**’ command is required to force the trace capture unit to validate the configuration. This is to allow a number of values to be set, before doing a ‘**`commitconfig`**’ to validate the set. When setting certain config items, it is possible to temporarily have an illegal combination due to the one at a time nature of setting the items, therefore the combination is only validated once ‘**`commitconfig`**’ is sent, which may be after a number of individual items.
If an illegal combination is detected by the ‘**`commitconfig`**‘ command, then an error is returned and none of the config items are set.

Note: before a config item is committed, attempting to read the item results in the original value being read back, not the value about to be committed.

The following sub-sections describe the configuration items, which modes and trace capture unit versions they appear in, and possible values where appropriate.
The modes and capture unit versions are abbreviated as follows:

RVT2X	-	RVT2 classic store and forward mode.
RVT2C	-	RVT2 continuous store and forward mode.
DST	-	DSTREAM – all modes

6.13.1 PORT_WIDTH

Item Name : PORT_WIDTH

Valid In : All versions and modes.

Description: The bit width of the connected ETM or TPIU port. Classic modes have discrete sets of values, continuous is variable from 1 to max port width for capture device. Port widths greater than 16 bit, require RVT2 using a 32 bit trace probe.

Type : Integer

Values : RVT2X, RVT2XS – 4,8,16,32 bits; RVT2C, RVT2CS – 1 to 32 bits; DST 1-16 bits

6.13.2 PORT_MODE

Item Name : PORT_MODE

Valid In : Any classic mode.

Description : The current setting on the ETM if the data is normal or multiplexed across the output pins.

Type : Enum

Values : Mode_Normal, Mode_Mux;

6.13.3 CLOCK_MODE

Item Name : CLOCK_MODE

Valid In : Any classic mode.

Description : The clock mode used for data capture – either single edged – SDR, or double edged – DDR.

Type : Enum

Values : Clock_SDR, Clock_DDR

6.13.4 ETM_PROTOCOL

Item Name : ETM_PROTOCOL

Valid In : Any classic mode

Description : information for the capture device on the ETM protocol being captured. Allows rejection of disabled trace cycles

Type : Enum

Values : RVT2X, RVT2C - ETMv1, ETMv3, ETMv1Tov3; DST - ETMv1, ETMv3

6.13.5 PACKING

Item Name : PACKING

Valid In : All modes

Description : How the data is packed into the trace data ram. RVT2 uses a maximum or minimal packing scheme. DSTREAM does not vary packing

Type : boolean

Values : RVT2 - true – maximum packing, false – minimum packing.

6.13.6 SET_TIMESTAMPS

Item Name : SET_TIMESTAMPS

Valid In : All modes in RVT2

Description : Determines if timestamps are generated and recorded by the RVT unit.

Type : boolean

Values : true – timestamps in use; false – no timestamps being used.

6.13.7 TRIGGER_POSITION

Item Name : TRIGGER_POSITION

Valid In : All modes – RVT2 only.

Description : determines the trigger position in the capture buffer, in terms of a percentage of the buffer.

Type : integer

Values : 1 – 100

6.13.8 TRACE_DEPTH

Item Name : TRACE_DEPTH

Valid In : Any mode. Read only in DSTREAM

Description : How large a buffer to used when tracing data. May be used to artificially reduce the captured data size in RVT2.

Type : integer.

Values : depends on size of buffer.

6.13.9 SYNC_STRIP

Item Name : SYNC_STRIP

Valid In : Any continuous mode

Description : Determines if the contiguous TPIU frame syncs, or half frame syncs, are stripped from the data before it is stored.

Type : Enum – RVT2C, RVT2CS;

Values : RVT2 – KeepAll, StripHalf; StripFull.

Unused in DSTREAM.

6.13.10 CLOCK_EDGE

Item Name : CLOCK_EDGE

Valid In : Any classic mode. (visible but unused in RVT2 continuous modes)

Description : Determines the clock edge to capture data if using SDR clocking.

Type : enum

Values : SDR_Rising – capture on rising edge; SDR_Falling – capture on falling edge.

6.13.11 POST_TRIGGER_FILL

Item Name : POST_TRIGGER_FILL

Valid In : RVT2 none streaming modes.

Description : continue to fill buffer to the end after a trigger operation, even if this exceeds buffer depth.
Type : boolean
Values : true / false

6.13.12 The DELAY_TRACE_xxx set

The config items DELAY_TRACE_CLOCK, and DELAY_TRACE_SIGNAL_1 to DELAY_TRACE_SIGNAL_36, allow the user to tune the line delays in the trace data capture system in RVT2. See ref 1 for further information.

6.13.13 Diagnostic Config Items

The following RVT2 items are for diagnostic and test use only – do not use.
TPIU_DEBUG.
FPGA_DIAGNOSTICS.
The following DSTREAM item is also for diagnostic use only
USB_ENUM_ID.

6.13.14 DSTREAM Trace Capture Parameters

The amount of trace captured, and trigger position is controlled by a group of config items in DSTREAM.

6.13.14.1 BUFFER_FULL_MARKER.

Integer number of 512 byte trace lines to capture before buffer is full. Limits the buffer size

6.13.14.2 TRIGGER_RUN_LENGTH

Integer number of trace lines to capture after a trigger is detected.

6.13.14.3 STOP_AFTER_TRIGGER

Boolean controlling if the trace capture stops after a trigger is detected. See the TRIGGER_RUN_LENGTH element for the number of frames captured after the trigger, and before the capture is halted.

6.13.14.4 WRAP_ON_FULL

Boolean controlling if the trace capture stops when full, or wraps and starts overwriting oldest data until a stop condition is met.

6.14 Trace Async

```
trace async {on | off }
```

This command enables or disables the display of the asynchronous status messages from RVT within CSAT. These occur if the state of trace capture changes, or if configuration changes. The information given in the async state change message is the same as that printed by the ‘**`getstate`**’ function.

7 EXAMPLES

7.1 Example session log

```
%%con TCP:Koyaanisqatsi
Connected to:ARM RealView ICE
Base H/W: V1 Rev C-01
TurboTAP Rev: 1.10
Firmware: 7.2.0, Build 12
%%chain dev=auto clk=A
Jtag clock set to 50000000A
ID:0 ARMCS-DP
ID:1 ETMBUF
ID:2 ARM1136JF-S
%%dmr DPACC.CSW
dpmemread : Insufficient command parameters.
%%drr DPACC.CSW
Command failed - no connection.
%%dvo 0
Open connection to device ID : 0x00000F0F, version 0x00000006
Msg returned with RVMOpenConn: CoreSight DAP template
%%drr DPACC.CSW
DPACC.CSW [2081] => 0xF0000000
%%dmr 0 0x8000 8
0x00008000 : 0x86A942AF 0x6A41717B 0x1BF3500C 0xE1AC9832
0x00008010 : 0xD07FBD57 0xD50D030F 0xFA736D84 0x2409FF10
%%dmw 0 0x8000 1 2 3 4 5 6 7 8
Wrote 8 words to memory via AP0
%%dmr 0 0x8000 8
0x00008000 : 0x00000001 0x00000002 0x00000003 0x00000004
0x00008010 : 0x00000005 0x00000006 0x00000007 0x00000008
%%drr AP0.CSW
AP0.CSW [1800] => 0x43800062
%%drr AP0.0
AP0.0 [1800] => 0x43800062
%%drr APACC.0
APACC.0 [2000] => 0x43800062
%%exit
```

7.2 Example Batch Command File

```
# a startup script

# A single line comment
echo Running my startup script
```

```
con TCP:Koyaanisqatsi   # an inline comment
chain dev=auto clk=A
dvo 0
```

7.3 Example Setup Script Using Coresight Component Commands

This setup script is used with a 3 core A9 on a PBXA9 board.

```
# CSAT 2.0 - PB-A9 component setup

# components all on APb-AP - apidx 1
cscomp def_apidx 1

####  3 a9 cores

# core 0
#
v7dbg.0 baseaddr 0x80110000
pmua9.0 baseaddr 0x80111000
cti.0   baseaddr 0x80118000
ptm.0   baseaddr 0x8011c000

alias v7dbg.0   a9_0
alias pmua9.0   pmu_0
alias cti.0     cti_0
alias ptm.0     ptm_0


# core 1
#
v7dbg.1 baseaddr 0x80112000
pmua9.1 baseaddr 0x80113000
cti.1   baseaddr 0x80119000
ptm.1   baseaddr 0x8011d000

alias v7dbg.1   a9_1
alias pmua9.1   pmu_1
alias cti.1     cti_1
alias ptm.1     ptm_1

# core 2
#
v7dbg.2 baseaddr 0x80114000
pmua9.2 baseaddr 0x80115000
cti.2   baseaddr 0x8011a000
ptm.2   baseaddr 0x8011e000

alias v7dbg.2   a9_2
alias pmua9.2   pmu_2
alias cti.2     cti_2
alias ptm.2     ptm_2


# common cs comps
etb.0  baseaddr 0x80001000
```

```
cti.4  baseaddr 0x80002000
tpiu.0 baseaddr 0x80003000
tfun.0 baseaddr 0x80004000
itm.0  baseaddr 0x80005000

alias cti.4   cscti
alias etb.0   csetb
alias tpiu.0  cstpiu
alias tfun.0  cstfun
alias itm.0   csitm
```

Once the script is run then the aliases can be used to control components – e.g. halt core 0

```
a9_0 rr prsr
a9_0 rr prsr  # read prsr to clear down sticky power down bit
a9_0 rr dscr # read dscr
a9_0 rmw dscr 0x00004000 0x00004000  # write dscr halt mode debugging bit
a9_0 rw prcr 0x1  # request core to halt
a9_0 rr dscr # read dscr – check it has halted
```

7.4 Example Script - Setting PTM to Trace All

```
# setup ptm to trace all - default ptm instance

# print ptm settings
ptm

# print current control register
ptm rr cr

# write power up and programming bit on
ptm rw cr 0x400

# check programming bit is on
ptm rr sr
ptm rr cr

# write the trace control regs for trace all.
ptm rw tecr1 0x01000000
ptm rw teevr 0x6f

# write the trace id to 9
ptm rw traceidr 9

# done – now need to clear the program bit and trace will start.
```

8 APPENDIX

8.1 Processor names that may appear in a JTAG scan chain.

The following processor template names may appear on a scan chain that is already set, during auto-detection, or can be used when defining a scan chain. These cannot be opened using the CSAT dvo command.

Arm 7 Family : ARM7EJ-S, ARM710T, ARM720T, ARM720T_r4, ARM740T, ARM7TDMI, ARM7TDMI-S, ARM7TDMI_r4, ARM7TDMI-S_r4.

Arm 9 Family : ARM966E-S, ARM940T, ARM946E-S, ARM926EJ-S, ARM9SMP-Eval, ARM925T, ARM922T, ARM920T, ARM9E-S, ARM9EJ-S, ARM9TDMI.

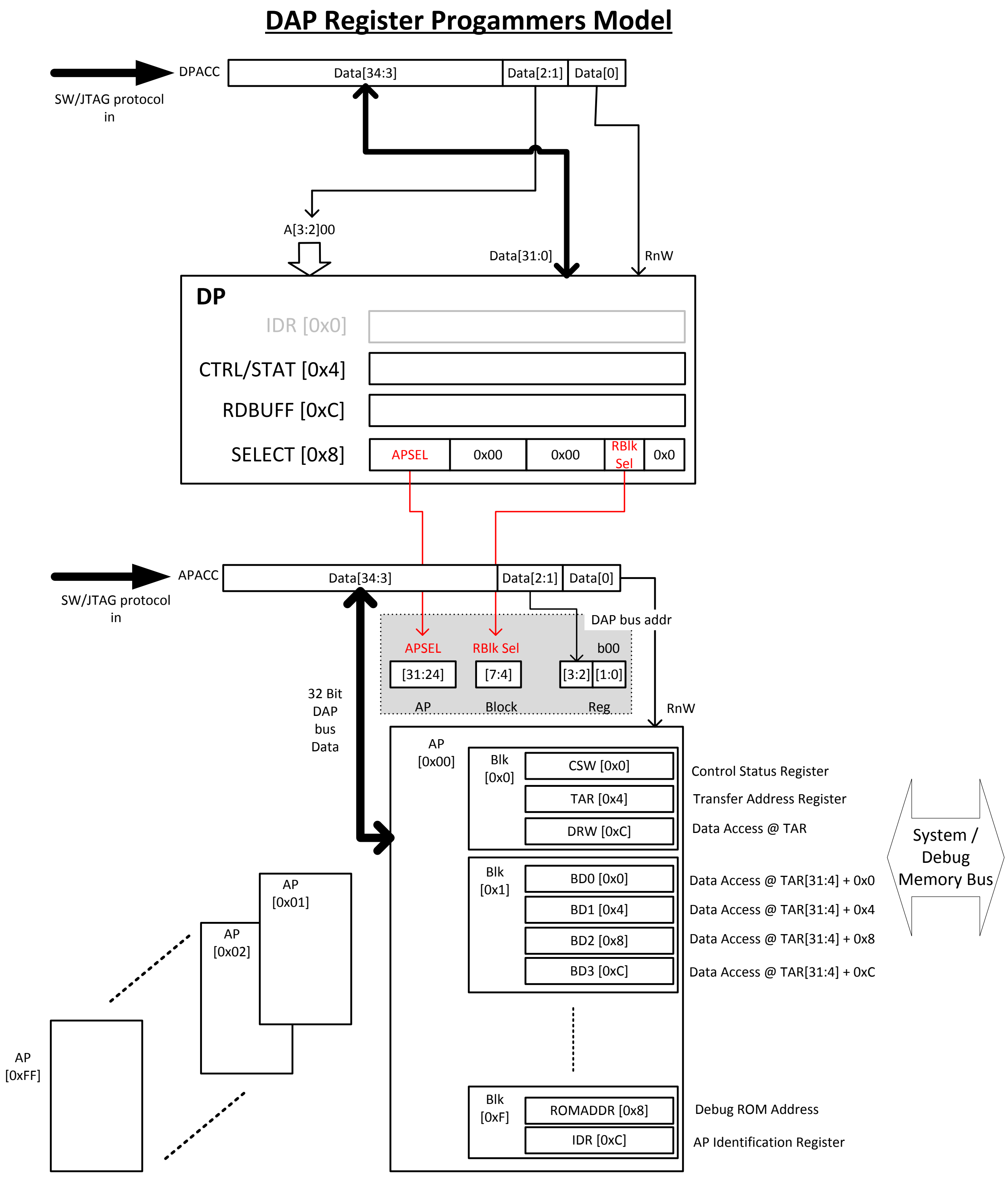
Arm 10 Family : ARM1020E, ARM1022E, ARM10200E, ARM10220E, ARM1026EJ-S

Arm 11 Family : ARM1136J-S, ARM1136JF-S, ARM1156-S, ARM1156F-S, ARM1176JZF-S

Secure Core : SC100D, SC200D

Arm Misc : ETMBUF, MPCore,

8.2 DAP Register Programmers Model Diagram


DAP Register Progammers Model
DPACC
SW/JTAG protocol in
Data[34:3]
Data[2:1]
Data[0]
A[3:2]00
Data[31:0]
RnW
DP
IDR [0x0]
CTRL/STAT [0x4]
RDBUFF [0xC]
SELECT [0x8]
APSEL
0x00
0x00
RBlk Sel
0x0
APACC
SW/JTAG protocol in
Data[34:3]
Data[2:1]
Data[0]
DAP bus addr
APSEL
RBlk Sel
b00
[31:24]
[7:4]
[3:2]
[1:0]
AP
Block
Reg
RnW
32 Bit DAP bus Data
AP [0x00]
Blk [0x0]
CSW [0x0]
TAR [0x4]
DRW [0xC]
Control Status Register
Transfer Address Register
Data Access @ TAR
Blk [0x1]
BD0 [0x0]
BD1 [0x4]
BD2 [0x8]
BD3 [0xC]
Data Access @ TAR[31:4] + 0x0
Data Access @ TAR[31:4] + 0x4
Data Access @ TAR[31:4] + 0x8
Data Access @ TAR[31:4] + 0xC
System / Debug Memory Bus
AP [0x01]
AP [0x02]
AP [0xFF]
Blk [0xF]
ROMADDR [0x8]
IDR [0xC]
Debug ROM Address
AP Identification Register