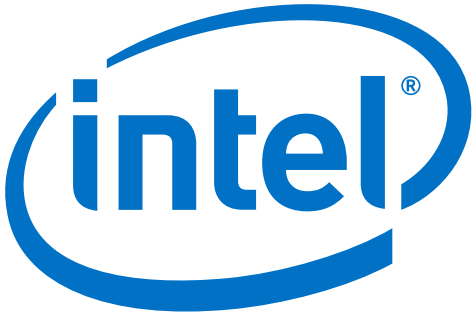

Movidius™

moviDebug

Manual

Version 00.90.0 / 2018-05-29

Table of Contents

Copyright and Proprietary Information Notice

Movidius Ltd.
1730 South El Camino Real, Suite 200
San Mateo, CA 94402
http://www.movidius.com/

1. Introduction

Movidius Debugger (moviDebug2) is an application that is used to test and debug Movidius code on different platforms.

1.1. Conventions used in this document

General moviDebug2 command format is:

```
command ?subcommand?... ?-optionalSwitch [optionalArgument]?... ?optionalParameters?...
```

The moviDebug2 commands in the present document are presented in a `Fixed width font`.

The parameters of a moviDebug2 command are separated by whitespace. (Note: For passing arguments containing spaces, please consult the Tcl/Tk documentation.)

The arguments preceded by `-` are actual switches accepted by the commands.

The parameters not preceded by `-` are not always actual keywords. Consult the command's parameter description and examples for clarification.

The `{ }` denote a set of possible choices, `|` separates the alternatives.

The `? ?` notation marks an optional parameter.

The `[ ]` notation marks an optional part of a(n optional) parameter.

The `...` notation means that the argument can be repeated multiple times

2. Overview

2.1. Features

The Movidius Debugger is based on the TCF Agent framework provided by the Eclipse environment. The TCF support is used for integration with the Eclipse IDE and to implement many of the debugger commands.

The debugger shell also provides a fully featured Tcl 8.6 command line interface.

Tcl Wiki (http://wiki.tcl.tk/)

Eclipse TCF Wiki (https://wiki.eclipse.org/TCF)

2.2. Shell Command line and scripting

The moviDebug2 command line interface provides the necessary commands for loading, running ad debugging programs. It is based on Tcl 8.6 scripting, so any Tcl specific command is available at the command line.

Built in support is also available for

- **tk**, the Tcl standard widget toolkit (version 8.6) with `ttk` support.
- **tcllib**, the Tcl standard library (version 1.18).

None of the above special packages are internally used by the debugger DLL/SO component's scripts.

MoviDebug2 uses and exposes *TclOO*, the official Tcl object-oriented extension built into the Tcl 8.6 interpreter.

2.3. Startup switches

The moviDebug2 command line format is:

```
moviDebug2 [OPTION...] [--] [FILE...]
```

The switches are presented in the table below:

Table 1. **Available switches**

Switch	Description

`--server[-host] HOST`
`-serverIP:HOST`
`-srvIP:HOST`
	Specify host to connect to.
	
	Defaults to `localhost`.

Switch	Description

`--[server-]port PORT`
`-serverPort:PORT`
`-srvPort:PORT`
	Specify TCP port of server to connect to.
	
	If not specified (or set to zero), the debugger will try to connect to port 30000 (first, and then to port 30001) at initialization.

`--chip-version CHIPVERSION` `-cv:CHIPVERSION`	**Specify chip version.** e.g: `ma2100`, `ma2150`, `ma2x5x`, `ma2450`, `ma2480`. Affects default DDR initialisation ELF file. Affects which Tcl script from the `cv` directory will be `source`-d. Note: if the server reports an incompatible chip version (e.g ma2x8x instead of ma2450), it will override this parameter.
`--verbose[=VERBOSITY]` `-verbose[:VERBOSITY]`	**Enable verbose mode.** Enables verbose TCF log and CLI output. `VERBOSITY` should be one of `debug`, `verbose`, `on`, `off`, `true`, `false`, `info`, `warning`, `error`, `silent`
`--proxy[-port] PORT` `-proxy[Port]:PORT`	**Enable moviDebug protocol proxy.** Allows connecting a second debugger and forwards its requests to the connected server.
`--no-tcf` `-noTCF`	**Disable TCF agent.** Skips initializing the TCF agent. Its services will be unavailable for the session.
`--no-tcf-discovery` `-no[Tcf]Discovery`	**Disable TCF agent discovery service** Disables the TCF Agent Discovery Service. This service runs on UDP and enables automatic discovery of agents on the local network.
`--tcf[-transport][-url] URL` `-tcf[Transport][Url]:URL`	**Set TCF agent transport URL.** Sets listen hostname for TCF agent. This effectively selects the interface on which to listen for incoming connections. e.g `--tcf-transport-url TCP:127.0.0.1:1234` will bind only to local host port 1234. **Notes:** — the URL prefix `PIPE:` can be used to bind to Named Pipes. (Windows only.) — the URL prefix `UNIX:` can be used to bind to Unix domain sockets. (Not available for Windows.)

Switch	Description
`--tcf[-listen]-host HOST` `-tcf[Listen]Host:HOST`	**Set TCF agent to listen on IP address or hostname.** Sets listen hostname for TCF agent. This effectively selects the interface on which to listen for incoming connections. For example, `--tcf-listen-host localhost` would prevent connections from other computers. (Also available as `--tcf[-listen]-local[host]`).
`--tcf[-listen]-port PORT` `-tcf[Listen]Port:PORT`	**Set TCF agent listen port.** Sets listen port for TCF agent. This allows distinguishing between and connecting to separate instances of moviDebug from Eclipse.
`--no-uart` `-noU{art|ART}`	**Disable UART initialisation and polling.** Equivalent with --tcl-var mdbg::UART=off Does not initialise UART registers and don't poll UART output. **Note:** UART polling is implemented as chip version dependent Tcl scripts.
`--no-[core-]state-update`	**Disable periodic core state polling.** Periodic state update can be controlled using the `coreconfig stateupdate` command. Manual state update can be forced on cores/groups with the `state -update` command. **Note:** some operations such as `halt`, `breset`, `step` or `cont` will force state updates in the model.
`--no-rtems[-threads]`	**Disable RTEMS threads support**
`--no-leon-rt`	**Disable support for LEON RT core**
`--single-leon-rt`	**Only support LEON RT as single-core**
`--no-leon-nn`	**Disable support for LEON NN core**
`--no-shaves`	**Disable support for SHAVE cores**
`--no-ucnn` `--no-shave-nn`	**Disable support for NN microcores (ucNN)** aka. ShaveNN cores.
`--no-shave-vector-registers`	**Disable support for SHAVE vector registers**
`--use-shave-l2-cache-as-ddr`	**Load DDR sections into Direct-Mode initialized Shave L2 Cache**
`--no-mdbg[-dll|-lib]-init` `-noMDbg[Dll|Lib]Init`	**Don't initialize the moviDebug2 library.** Skips the moviDebug shared library initialization. `mdbg::dll::init` will need to be called manually from Tcl.

Switch	Description
`--no-tcl` `-noTcl`	**Don't use Tcl.** Does not load the Tcl shared library. The moviDebug2 shared library will be loaded/ initialized directly. The Tcl shell and scripting features will be unavailable.
`--[change-]dir[ectory] DIR` `-C DIR`	**Change to directory.** Change to directory DIR before parsing the rest of the arguments. **Notes:** — *This option has immediate effects during argument parsing, which means it will affect subsequent file name arguments!* — The directory must exist.
`--log[-file] FILE` `-log[File]:FILE`	**Log console activity to file.** Logs all console I/O into given file for later inspection.
`--[tcl-]init FILE` `-[tcl]Init:FILE`	**Additional Tcl shell initialisation script(s).** Allows running additional Tcl scripts to initialise the Tcl environment. The option name needs to be given for each file separately. The script(s) run after the moviDebug2 shared library initialization (unless `--no-mdbg-dll-init` was also specified). **Note:** *Any Tcl error encountered during the execution of these scripts will cause the program to exit, even if interactive mode (see below) is enabled!*
`--[tcl-]script FILE` `FILE...`	**Run moviDebug2 Tcl script(s) (instead of interactive shell).** Run Tcl script(s) instead of interactive session (unless `--interactive` also specified) The option name needs to be given for each file separately. The script(s) run after the Tcl shell initialization. **Note:** *Any Tcl error encountered during the execution of these scripts will stop further script execution!*
`--interactive` `-i`	**Don't exit after non-init scripts finished.** Runs the interactive console after the (non-init) scripts finish. Normally the program will exit if (non-init) scripts were specified in its command line. This option will prevent that behaviour and let the interactive console run even in this case.

Switch	Description
`--no-color` `-noColor`	**Disable coloured output.** Forces the debugger NOT to use ANSI terminal colours (even if `MV_DBG2_COLOR` is set to *true*).
`--[force-]color` `-[force]Color`	**Forces coloured output.** Forces the debugger to use ANSI terminal colours (even if `MV_DBG2_COLOR` is set to *false*).
`--stdio`	**Force standard I/O console.** Forces the debugger NOT to use the ANSI terminal/WinConsole driver.
`--ddr-init[-elf] PATH_TO_ELF` `-ddrInit:PATH_TO_ELF`	**Alternate DDR initialisation ELF file path.** Specify an alternate ELF file for DDR initialisation. At DDR initialisation, the debugger will run this file instead of the chip-version-dependent standard DDR initialisation executable. *No error checking is performed.*
`--[tcl-]var VAR=VALUE` `-D:VAR=VALUE`	**Define Tcl variable.**
`--server-messages VERBOSITY`	**Set server messages verbosity** Select minimum log level of server-sent asynchronous messages that will be displayed. VERBOSITY must be one of `silent`, `debug`, `verbose`, `info`, `warning`, `error`. The default is `silent`. **Note:** *messages below the* `dll::loglevel` *setting (available in Tcl) will not be displayed even if this switch enables them!*
`--[custom-]client-[busy-]message MESSAGE`	**Set custom "client busy" message** The message will be sent to other clients of the server while it's busy with this client. Useful in scenarios where multiple moviDebug2 clients share a scarce moviDebugServer resource. **Note:** the message text can be set or changed later in the Tcl CLI using the `mvproto::setClientBusyMessage` command.
`--version` `-version`	**Show version and exit.**
`--help`	**Show help and exit.**

2.4. Environment variables

The system environment variables affecting runtime behaviour are presented in the table below:

Table 2. **Environment variables**

Variable name	Description
MV_DBG2_SERVER_STATS	**Enable server communication statistics.** The feature is off by default. Setting the value to 1 or true enables it. The statistics will be printed at the end of the debugging session.
MV_DBG2_SERVER_DEBUG_LOG	**Enable/disable server interface debug logging.** The feature is off by default. Setting the value to 1 or true enables it. **Notes:** - the log level needs to be set to debug for the messages to be visible - e.g. by dll::loglevel debug command - e.g. by -verbose:debug or --verbose=debug command line switch - e.g. by MV_DBG2_LOGLEVEL=DEBUG environment variable
MV_DBG2_REGISTER_BLOCK_CACHE	**Disable or enable the register block cache.** Setting the value to 0 or false disables the feature, setting 1 or true enables it. When enabled, internal accesses of the cores' contiguous registers will be done in burst blocks instead of individual reads. The values are cached temporarily. **Note:** *this feature is automatically enabled for moviUsbJtag interface.*
MV_DBG2_COLOR	**Control use of ANSI terminal colors** Setting this value to 0 or false will disable use of terminal colors by default. Setting it to 1 or true will enable use of terminal colors by default. This is also the default behaviour. **Note:** *this environment setting can be overridden by the* --[force-]color *or* --no-color *switches.*
MV_DBG2_LOGLEVEL	**Control default moviDebug2 DLL verbosity** Should be one of debug, verbose, info, warning, error, silent Notes: - this environment setting can be overridden by the --verbose[:VERBOSITY] switch. - this environment setting can be changed by the mdbg::dll::loglevel command.
MV_DBG2_SERVER_LOGLEVEL	**Control default server messages verbosity** Should be one of debug, verbose, info, warning, error, silent **Notes:** - this environment setting can be overridden by the --server-messages switch. - this environment setting can be changed by the mdbg::dll::srvloglevel command.
MV_DBG2_VCS_SAVE_DIR	**Control VCS Hooks save file directory** Default is the current working directory of the application

Variable name	Description
MV_DBG2_VCS_LOAD_DIR	**Control VCS Hooks load file directory** Default is the current working directory of the application

2.5. Command line auto-completion

The command line supports auto-completion of commands, sub-commands, register names and variables.

2.6. Quick start guide

For the first-time user, the default values for the command line switches provide for a quick and easy start. Thus, there is no need for any switches to be used in order to get started. They are useful only for specific configurations that more advanced users may consider.

The debugger will try to connect by default to the local host via port 30000 (default moviSim port). If this fails, the debugger will try port 30001 next (default port for the moviDebugServer).

Once a connection is made successfully, the debugger will identify the platform it has connected to and print that information for the user.

The command prompt is printed, giving the user the opportunity to issue various commands, which are described in the other sections of this document.

The most commonly used commands are:

- `help ?command?...` — display a list of commands if used with no argument or the help for a specific (sub)command.
- `breset` — reset the board.
- `startupcore LOS` — set the startup core to Leon OS.
- `loadfile ELF` — load the specified ELF file.
- `breakpoint add -file FILE -line LINE` — add a breakpoint in the specified FILE at the specified LINE.
- `mget {address|variable}` — get the value of an address / variable.
- `target {LOS|LRT|Sn}` — set current active context to LOS or LRT or Shave *n*.
- `callstack` — display the callstack for the selected context.

For a more detailed description of the commands, the full list of commands and their options is described in the following chapters of this document.

3. Command reference

3.1. alias — Define command shortcut.

Synopsis

```
alias name command ?args...?
```

Description

Useful, autocomplete-friendly version of the `interp alias` command.

Lets the user define their own shortcuts to commands and command-argument combinations.

Note: For more complicated situations, use the Tcl built-in `proc` to define procedures .

Examples

Simplify typing:

```
% alias c continue
```

```
% alias i step into -line
```

```
% alias s step over
```

```
% alias r step return
```

```
% alias l listsource -2 +3
```

```
% alias k callstack
```

Define a jtag-only `mget` command:

```
% alias jget mget -jtag
```

- this is legal afterwards:

```
% jget -sym mvConsoleTxQueue
```

Define a printf-like command:

```
% alias printf apply {{args} {puts [format {*}$args]}}
```

- allows for

```
% printf "Address is 0x%08x" $addr
```

Arguments

Synopsis	Description
`name`	**Name of alias**
`command`	**Target command to refer to.**
`args`	**Extra argument of target command**

3.2. breakpoint — Manage breakpoints.

Synopsis

```
breakpoint {list|add|modify|remove|enable|disable|messages|ls|change|configure|insert|rm|delete} ?args?...
```

Description

The command allows management of all types of TCF breakpoints/watchpoints and their properties.

This includes SOFTWARE and HARDWARE breakpoints and INSTRUCTION, DATA READ and DATA WRITE breakpoints.

The following subcommands are supported:

```
SUBCOMMAND, VARIANTS        DESCRIPTION
list, ls                    List breakpoints.
add, insert                 Add new breakpoint.
modify, change, configure   Change breakpoint properties.
remove, delete, rm          Remove breakpoint(s).
enable                      Enable breakpoint(s).
disable                     Disable breakpoint(s).
messages                    Control display of breakpoint status
                            messages.
```

Aliases

```
ALIAS       COMMAND
bp          breakpoint
lsbp        breakpoint list
```

Notes

- breakpoints can be added even if no file has been loaded.
- the breakpoint PROPERTIES (definitions) determine the way the breakpoints will be physically PLANTED.
- breakpoints are re-evaluated when the memory map (symbol information) of a target changes.
- the breakpoint STATUS gives information about the physical INSTANCEs of the PLANTED breakpoints.
- breakpoints can be associated with particular TARGETs, or can exist without target.
- Eclipse creates breakpoints without targets, the CLI adds the current target by default.
- breakpoint definitions are bound to the TCF client who creates them. This means breakpoints set from within the Eclipse GUI and the Tcl CLI should not be modified by the other. Also they will automatically be removed when the client (e.g Eclipse IDE) disconnects.
- multiple breakpoint definitions can trigger the same point
- a single breakpoint definition can trigger multiple points
- to preserve resources, only the first address of a multi-point hardware breakpoint definition will be planted.

Examples

Add a software breakpoint in main.cpp file at line 50:

```
% breakpoint add -type software -file main.cpp -line 50
```

or:

```
% bp add main.cpp:50
```

Add a hardware breakpoint in main.cpp file at line 60:

```
% breakpoint add -file main.cpp -line 60 -type hardware
```

Display all the breakpoints:

```
% breakpoint list
```

Change the breakpoint with ID #1 to hardware:

```
% breakpoint modify #1 -type hardware
```

Delete the breakpoint with ID #2:

```
% breakpoint remove #2
```

Run to POSIX_Init (remove breakpoint after hit):

```
% breakpoint add -temporary -location POSIX_Init; run -wait
```

Easily run to line 60 of current file:

```
% bp add -temp :60; contw
```

3.2.1. breakpoint add, breakpoint insert — Add breakpoint.

Synopsis

```
breakpoint add ?-quiet? ?-id id? ?-type {auto|software|hardware}? ?-location location? ?-file file? ?-line line? ?-column column? ?-target targetList?... ?-no-target? ?-condition condExpr? ?-ignorecount ignoreCount? ?-execute? ?-read? ?-write? ?-change? ?-size bpSize? ?-temporary? ?-skip-prologue? ?-stopgroup stopGroupTargetList? ?-maskvalue maskValue? ?-mask mask? ?-inverted? ?-enabled? ?-disabled? [file]:line ?addressOrSymbolName?
```

```
breakpoint insert ?-quiet? ?-id id? ?-type {auto|software|hardware}? ?-location location? ?-file file? ?-line line?
?-column column? ?-target targetList?... ?-no-target? ?-condition condExpr? ?-ignorecount ignoreCount? ?-execute? ?-
read? ?-write? ?-change? ?-size bpSize? ?-temporary? ?-skip-prologue? ?-stopgroup stopGroupTargetList? ?-maskvalue
maskValue? ?-mask mask? ?-inverted? ?-enabled? ?-disabled? [file]:line ?addressOrSymbolName?
```

Arguments

Synopsis	Description
`-quiet`	**Hide feedback** Temporarily disables messages about breakpoints being added/removed/changed during the call. This effectively will disable feedback about the breakpoint in question.
`-id id`	**ID of breakpoint.** If missing, the CLI will generate an ID automatically. This ID will have the form #<number> and should be unique.
`-type {auto|software|hardware}`	**Breakpoint type (software, hardware, auto).**
`-location location`	**Breakpoint location.** The breakpoint location is either a symbol or an adddress. Cannot be used together with `-file`, `-line` and `-column`.
`-file file`	**Breakpoint file.** The file where the breakpoint is located. Requires `-line`. Can't be used together with -location.
`-line line`	**Breakpoint line.** The line in the file where the breakpoint is located. Requires `-file`.
`-column column`	**Breakpoint column.** The column in the line in the file where the breakpoint is located. Requires `-line`
`-target targetList...`	**Specify breakpoint target.** Affects which targets the breakpoint will be set to. If missing when adding breakpoints, the current target is used. The `-no-target` option can specifiy that the breakpoint is available for all the targets. Can be specified multiple times or as a list of targets.
`-no-target`	**Make breakpoint available on all targets.**
`-condition condExpr`	**Breakpoint condition.** Expression to be evaluated by TCF before reporting the target to be stopped in breakpoint. If the expression evaluates to false, the framework will resume the target automatically.

Synopsis	Description
`-ignorecount ignoreCount`	**Breakpoint ignore count.** Number of times the breakpoint has to be ignored. That is, TCF will resume the target this number of times automatically.
`-execute`	**Instruction breakpoint.** This is the default breakpoint type.
`-read`	**Data read breakpoint.** Can be combined with `-write` and `-change`.
`-write`	**Data write breakpoint.** Can be combined with `-read` and `-change`.
`-change`	**Data change breakpoint.** Reserved.
`-size bpSize`	**Size of the breakpoint in bytes.** Used to specify break range. NOTE: not all targets support all sizes of breakpoints.
`-temporary`	**Temporary breakpoint.** Causes the breakpoint to be deleted after first hit.
`-skip-prologue`	**Skip function prologue.** Causes the execution engine to skip the function prologue when the breakpoint is being hit.
`-stopgroup stopGroupTargetList`	**Stop other targets.** Specify the targets which need to be stopped when this breakpoint is hit. The argument should be a Tcl list of target names. NOTE: If the stop group does not contain the target of the breakpoint, the debugger will resume it after hitting the breakpoint.
`-maskvalue maskValue`	**Breakpoint mask value.** A breakpoint can be qualified with a mask value which may be further refined with a mask.
`-mask mask`	**Breakpoint mask.** see `-maskvalue`
`-inverted`	**Invert breakpoint range.** Break if PC out of range. Not supported yet.
`-enabled`	**Enable breakpoint.** This is the default when adding new breakpoints.

Synopsis	Description
`-disabled`	**Disable breakpoint.** The breakpoints added via CLI are enabled by default. This switch prevents that. Disables the breakpoint at change.
`[file]:line`	**Source location** This argument was added for convenience. It is functionally equivalent with the parameters of the `-file` and `-line` arguments. Therefore it cannot be used in conjunction with those. In case the `file` part is empty, it is inferred from the current PC of the breakpoint target.
`addressOrSymbolName`	**Address or symbol name.** This argument was added for convenience. It is functionally equivalent with the parameter of the `-location` argument. Therefore it cannot be used in conjunction with that.

3.2.2. breakpoint change, breakpoint configure, breakpoint modify — Change breakpoint.

Synopsis

```
breakpoint change id ?-quiet? ?-type {auto|software|hardware}? ?-location location? ?-file file? ?-line line? ?-column column? ?-target targetList?... ?-no-target? ?-condition condExpr? ?-ignorecount ignoreCount? ?-execute? ?-read? ?-write? ?-change? ?-size bpSize? ?-temporary? ?-skip-prologue? ?-stopgroup stopGroupTargetList? ?-maskvalue maskValue? ?-mask mask? ?-inverted? ?-enabled? ?-disabled? ?-no-type? ?-no-location? ?-no-file? ?-no-line? ?-no-column? ?-no-condition? ?-no-ignorecount? ?-no-access-mode? ?-no-size? ?-not-temporary? ?-dont-skip-prologue? ?-no-stopgroup? ?-no-maskvalue? ?-no-mask? ?-not-inverted?
```

```
breakpoint configure id ?-quiet? ?-type {auto|software|hardware}? ?-location location? ?-file file? ?-line line? ?-column column? ?-target targetList?... ?-no-target? ?-condition condExpr? ?-ignorecount ignoreCount? ?-execute? ?-read? ?-write? ?-change? ?-size bpSize? ?-temporary? ?-skip-prologue? ?-stopgroup stopGroupTargetList? ?-maskvalue maskValue? ?-mask mask? ?-inverted? ?-enabled? ?-disabled? ?-no-type? ?-no-location? ?-no-file? ?-no-line? ?-no-column? ?-no-condition? ?-no-ignorecount? ?-no-access-mode? ?-no-size? ?-not-temporary? ?-dont-skip-prologue? ?-no-stopgroup? ?-no-maskvalue? ?-no-mask? ?-not-inverted?
```

```
breakpoint modify id ?-quiet? ?-type {auto|software|hardware}? ?-location location? ?-file file? ?-line line? ?-column column? ?-target targetList?... ?-no-target? ?-condition condExpr? ?-ignorecount ignoreCount? ?-execute? ?-read? ?-write? ?-change? ?-size bpSize? ?-temporary? ?-skip-prologue? ?-stopgroup stopGroupTargetList? ?-maskvalue maskValue? ?-mask mask? ?-inverted? ?-enabled? ?-disabled? ?-no-type? ?-no-location? ?-no-file? ?-no-line? ?-no-column? ?-no-condition? ?-no-ignorecount? ?-no-access-mode? ?-no-size? ?-not-temporary? ?-dont-skip-prologue? ?-no-stopgroup? ?-no-maskvalue? ?-no-mask? ?-not-inverted?
```

Arguments

Synopsis	Description
`id`	**ID of breakpoint to change.**
`-quiet`	**Hide feedback** Temporarily disables messages about breakpoints being added/removed/changed during the call. This effectively will disable feedback about the breakpoint in question.
`-type {auto|software|hardware}`	**Breakpoint type (software, hardware, auto).**
`-location location`	**Breakpoint location.** The breakpoint location is either a symbol or an adddress. Cannot be used together with `-file`, `-line` and `-column`.
`-file file`	**Breakpoint file.** The file where the breakpoint is located. Requires `-line`. Can't be used together with -location.
`-line line`	**Breakpoint line.** The line in the file where the breakpoint is located. Requires `-file`.
`-column column`	**Breakpoint column.** The column in the line in the file where the breakpoint is located. Requires `-line`
`-target targetList...`	**Specify breakpoint target.** Affects which targets the breakpoint will be set to. If missing when adding breakpoints, the current target is used. The `-no-target` option can specifiy that the breakpoint is available for all the targets. Can be specified multiple times or as a list of targets.
`-no-target`	**Make breakpoint available on all targets.**
`-condition condExpr`	**Breakpoint condition.** Expression to be evaluated by TCF before reporting the target to be stopped in breakpoint. If the expression evaluates to false, the framework will resume the target automatically.
`-ignorecount ignoreCount`	**Breakpoint ignore count.** Number of times the breakpoint has to be ignored. That is, TCF will resume the target this number of times automatically.
`-execute`	**Instruction breakpoint.** This is the default breakpoint type.
`-read`	**Data read breakpoint.** Can be combined with `-write` and `-change`.

Synopsis	Description
`-write`	**Data write breakpoint.** Can be combined with `-read` and `-change`.
`-change`	**Data change breakpoint.** Reserved.
`-size bpSize`	**Size of the breakpoint in bytes.** Used to specify break range. NOTE: not all targets support all sizes of breakpoints.
`-temporary`	**Temporary breakpoint.** Causes the breakpoint to be deleted after first hit.
`-skip-prologue`	**Skip function prologue.** Causes the execution engine to skip the function prologue when the breakpoint is being hit.
`-stopgroup stopGroupTargetList`	**Stop other targets.** Specify the targets which need to be stopped when this breakpoint is hit. The argument should be a Tcl list of target names. NOTE: If the stop group does not contain the target of the breakpoint, the debugger will resume it after hitting the breakpoint.
`-maskvalue maskValue`	**Breakpoint mask value.** A breakpoint can be qualified with a mask value which may be further refined with a mask.
`-mask mask`	**Breakpoint mask.** see `-maskvalue`
`-inverted`	**Invert breakpoint range.** Break if PC out of range. Not supported yet.
`-enabled`	**Enable breakpoint.** This is the default when adding new breakpoints.
`-disabled`	**Disable breakpoint.** The breakpoints added via CLI are enabled by default. This switch prevents that. Disables the breakpoint at change.
`-no-type`	**Reset breakpoint type.**
`-no-location`	**Reset breakpoint location.**
`-no-file`	**Reset breakpoint file.**
`-no-line`	**Reset breakpoint line.**
`-no-column`	**Reset breakpoint column.**

Synopsis	Description
`-no-condition`	**Reset breakpoint condition.**
`-no-ignorecount`	**Reset breakpoint ignore count.**
`-no-access-mode`	**Reset breakpoint access mode.** Resets access mode specified previously by the `-execute`, `-read`, `-write` or `-change` flags.
`-no-size`	**Reset breakpoint size.**
`-not-temporary`	**Make breakpoint permanent.** Cancels the effect of the `-temporary` flag.
`-dont-skip-prologue`	**Don't skip prologue** Cancels the effect of `-skip-prologue` flag.
`-no-stopgroup`	**Reset breakpoint stop group.**
`-no-maskvalue`	**Reset breakpoint mask value.**
`-no-mask`	**Reset breakpoint mask.**
`-not-inverted`	**Reset breakpoint inverted.** Cancels the effect of the `-inverted` flag.

3.2.3. breakpoint delete, breakpoint remove, breakpoint rm — Remove breakpoint(s).

Synopsis

```
breakpoint delete ?-quiet? ?-all? ?args?...
```

```
breakpoint remove ?-quiet? ?-all? ?args?...
```

```
breakpoint rm ?-quiet? ?-all? ?args?...
```

Arguments

Synopsis	Description
`-quiet`	**Hide feedback** Temporarily disables messages about breakpoints being added/removed/changed during the call. This effectively will disable feedback about the breakpoint in question.
`-all`	**All breakpoints.**

Synopsis	Description
`args...`	**Breakpoint ID(s) to consider.** Multiple IDs can to be given as consecutive arguments.

3.2.4. breakpoint disable — Disable breakpoint(s).

Synopsis

```
breakpoint disable ?-quiet? ?-all? ?args?...
```

Arguments

Synopsis	Description
`-quiet`	**Hide feedback** Temporarily disables messages about breakpoints being added/removed/changed during the call. This effectively will disable feedback about the breakpoint in question.
`-all`	**All breakpoints.**
`args...`	**Breakpoint ID(s) to consider.** Multiple IDs can to be given as consecutive arguments.

3.2.5. breakpoint enable — Enable breakpoint(s).

Synopsis

```
breakpoint enable ?-quiet? ?-all? ?args?...
```

Arguments

Synopsis	Description
`-quiet`	**Hide feedback** Temporarily disables messages about breakpoints being added/removed/changed during the call. This effectively will disable feedback about the breakpoint in question.
`-all`	**All breakpoints.**
`args...`	**Breakpoint ID(s) to consider.** Multiple IDs can to be given as consecutive arguments.

3.2.6. breakpoint list, breakpoint ls — List breakpoints.

Synopsis

```
breakpoint list ?-quiet? ?-no-status? ?-ids? ?args?...
```

```
breakpoint ls ?-quiet? ?-no-status? ?-ids? ?args?...
```

Arguments

Synopsis	Description
`-quiet`	**Return dictionary instead of printing to stdout.**
`-no-status`	**Omit status information**
`-ids`	**Return list of IDs.**
`args...`	**Argument list of breakpoint IDs to print information about.** If none specified, all breakpoints are to be shown.

3.2.7. breakpoint messages — Control display of breakpoint status messages.

Synopsis

```
breakpoint messages ?{off|on|verbose|show|hide}?
?{add|remove|change|plant|unplant|error|error.all|condition|hit}?...
```

Description

Return the list of currently displayed event messages.

The following modes are supported:

```
off     - no status messages displayed
on      - enable essential status messages
verbose - all messages displayed
```

```
show|hide ?add|remove|change|plant|unplant|error|condition|hit?...
        - show or hide individual message categories
```

Arguments

Synopsis	Description
`{off|on|verbose|show|hide}`	**Display mode.**

3.3. breset — Board reset.

Synopsis

```
breset ?-timeout timeout? ?-async?
```

Description

The command will try to reset the board to its initial state.

- If TCF is available, the corresponding TCF Platform service command will be called. It waits then for the main core of the platform to be suspended.
- If TCF is not available, a hand-coded sequence will be executed to reset the board.

Notes

- Resetting the board is done differently for moviSim and moviDebugServer.

Examples

Reset the board:

```
% breset
```

Arguments

Synopsis	Description
`-timeout timeout`	**Timeout (ms).** Raise error if main core does not enter into valid suspended state in the specified amount of time. The command will wait forever if this argument is not specified. Useful for scripting.
`-async`	**Asynchronous operation** Don't wait for the main core.

3.4. callstack — Get call stack for current target.

Synopsis

```
callstack ?-target target? ?-quiet? ?-start start? ?-count levels? ?target? ?count?
```

Description

Display a table of the call stack for target.

Numeric columns include the Instruction Pointer, the Frame Pointer and the Return Address.

Each row's instruction pointer is interpreted to display symbol and line number information.

Aliases

```
ALIAS       COMMAND
cs          callstack
```

Notes

A more traditional display format can be attained using:

```
% state -stack
```

Examples

Display call stack for current core:

```
% callstack
```

Display 3 levels from LOS call stack starting with the second level:

```
% callstack  -target LOS -start 1 -count 3
```

Display top 5 levels of call stack for current core:

```
% callstack 5
```

Arguments

Synopsis	Description
`-quiet`	**Don't print, return Tcl list**
`-start start`	**First level.**
`-count levels`	**Count of levels.** Default is 10 levels.

3.5. cont — Continue execution of target.

Synopsis

```
cont ?-wait? ?-async? ?-timeout timeout? ?-all? ?[-target] target?
```

Description

Resumes normal execution of target (a.k.a continue).

Optionally it can be asynchronous, which means that it will send the Resume command to the target but it will not wait for the target(s) to really resume.

At also can wait until the target or suspends again or a breakpoint is hit by another target.

See `-wait` and `-async` switches.

Aliases

```
ALIAS               FULL FORM
contw               cont -wait
```

Examples

Resume execution, wait until current core stops or a breakpoint is hit:

```
% contw
```

Resumes execution of the Leon RT:

```
% cont LRT
```

Arguments

Synopsis	Description
`-wait`	**Wait for target to suspend again.** Incompatible with `-async`.
`-async`	**Do not wait for target to resume.** Normally the command will wait for the target to resume. This wait can be bypassed, the command becoming asynchronous. Incompatible with synchronisation options `-wait` and `-timeout`.
`-timeout timeout`	**Raise error on timeout (ms).** Causes an error to be raised if the timeout period elapses before normal return. The normal return condition is: * target Suspended — with the `-wait` flag present. * target Resumed otherwise. Incompatible with `-async`.

Synopsis	Description
`-all`	**Continue execution of all suspended targets (asynchronous).** This is essentially an asynchronous operation, so the `-async` option is implied. Cannot be combined with options incompatible with `-async`, such as `-wait` or `-timeout`.
`[-target] target`	**Specify target.** If missing, the current target is used, except for the `-all` option. The `-target` option name can be missing if this is the last argument.

3.6. coreconfig — Configure core settings

Synopsis

```
coreconfig ?{safety|stateupdate|cacheaccess|help}? ?args?...
```

Description

Configure core settings

Arguments

Type "help coreconfig safety" to get help on arguments.

Examples

Do not allow any memory operations while running:

```
% coreconfig safety strict
```

Make an exception to allow disassembly from uncached memory:

```
% coreconfig safety -dasm bypass
```

We still want to stop the core when setting the breakpoints:

```
% coreconfig safety -breakpoint safe
```

Specify all three at once for LeonOS:

```
% coreconfig safety -target LOS -default strict -breakpoint safe -dasm bypass
```

Enable live RTEMS thread list update (from L2 cache data):

```
% coreconfig safety LOS -l1dread bypass -l2read unsafe
```

```
% coreconfig safety LRT -l1dread bypass -l2read unsafe
```

Restore default TCF behaviour:

```
% coreconfig safety LOS default
```

3.6.1. coreconfig cacheaccess — Get/set cache access policy

Synopsis

```
coreconfig cacheacess ?[-target] target? ?general | options...?
```

Description

Cache access policy configuration.

The cache access policy determines the cache coherency management during different memory operations of the TCF model.

The following policies are implemented:

- Precise (default): Cache memory is accessed via debug/diagnostic access. Write operations update all the data present in all the cache levels. Read operations retrieve data from nearest cache.
- Simple: caches are flushed before read and write, invalidated after write.
- Bypass: caches are bypassed. Uncached memory is accessed.

The following scenarios are supported:

- Default: applies to all operations
- Level 1: applies to Level 1 Instruction and Data
- Level 1 Data: applies to Read and Write
- Level 1 Data Read
- Level 1 Data Write
- Level 1 Instruction: applies to Read and Write
- Level 1 Instruction Read
- Level 1 Instruction Write
- Level 2: applies to Read and Write

- Level 2 Read
- Level 2 Write

Notes

- The `general` setting clears all previous settings.
- The default general setting is `precise`.
- The rule hierarchy is that the more generic settings provide the defaults to more specific settings. For example, the setting for `Level 1` is the default for `Level1 Instruction` which is the default for `Level 1 Instruction Read`.
- Cache access policy only affects the TCF memory model operations, that is, the following are not affected: PIPE I/O, UART polling, `mget -jtag`, `mset -jtag`, `mdump -jtag`, `loadfile`, `savefile`.
- Cache access policy does NOT affect how memory is accessed while the target is seen as RUNNING. See `coreconfig safety` for that.

Examples

Disable cache support entirely. Access only uncached data:

```
% coreconfig cacheaccess A bypass
```

Use simple level 1 instruction cache management for all LEON cores

```
% coreconfig cacheaccess LALL -l1i simple
```

Use simple level 2 cache management for all SHAVE cores

```
% coreconfig cacheaccess SALL -l2 simple
```

Restore default behaviour to all cores

```
% coreconfig cacheaccess A default
```

Arguments

Synopsis	Description
`[-target] target`	**Specify target.** If missing, the current target is used.
`-general {precise|simple|bypass|default}`	**General policy. Overwrites all previous settings.**
`-default {precise|simple|bypass|default}`	**Default policy**
`-l1 {precise|simple|bypass|default}`	**Level 1 cache access policy**

Synopsis	Description
`-l1i {precise|simple|bypass|default}`	**Level 1 Instruction cache access policy**
`-l1iread {precise|simple|bypass|default}`	**Level 1 Instruction cache read access policy**
`-l1iwrite {precise|simple|bypass|default}`	**Level 1 Instruction cache write access policy**
`-l1d {precise|simple|bypass|default}`	**Level 1 Data cache access policy**
`-l1dread {precise|simple|bypass|default}`	**Level 1 Data cache read access policy**
`-l1dwrite {precise|simple|bypass|default}`	**Level 1 Data cache write access policy**
`-l2 {precise|simple|bypass|default}`	**Level 2 cache access policy**
`-l2read {precise|simple|bypass|default}`	**Level 2 cache read access policy**
`-l2write {precise|simple|bypass|default}`	**Level 2 cache write access policy**
`-quiet`	**Return instead of printing.**
`{precise|simple|bypass|default}`	**Same as "-general"**

3.6.2. coreconfig help — Get help

Synopsis

```
coreconfig help ?args?...
```

3.6.3. coreconfig safety — Get/set core safety

Synopsis

```
coreconfig safety ?[-target] target? ?general | options...?
```

Description

Core safety policy configuration.

The core safety policy determines what should happen when different memory operations are requested from a running core.

The following policies are implemented:

- Safe (default): let TCF handle safety by always interrupting the core's execution. Notify user and resume automatically.
- Strict: do not stop the core, return memory access failure
- Bypass: do not stop the core, bypass the cache memories and access uncached data.
- Unsafe: do not stop the core, act as if it were stopped.

The following scenarios are supported:

- MemGet: Data Read
- MemSet: Data Write (implies reading)
- Dasm: Instruction Read
- Breakpoint: Instruction Write (implies reading)
- Level 1 Data Cache Read
- Level 1 Data Cache Write
- Level 1 Instruction Cache Read
- Level 1 Instruction Cache Write
- Level 2 Cache Read
- Level 2 Cache Write

A fallback mechanism is implemented to imply `default` policies per operation:

```
Default > MemGet > MemSet
MemGet  > L1 Data Read   > L1 Data Write
MemSet  > L1 Data Write
```

```
Default > Dasm > Breakpoint
```

```
Dasm       > L1 Instruction Read > L1 Instruction Write
Breakpoint > L1 Instruction Read > L1 Instruction Write
```

```
Default > L2 Cache Read > L2 Cache Write
```

Notes

- If Level 1 and Level 2 Cache Data Read policy becomes Unsafe or Bypass then the RTEMS thread list will be updated during runtime.
- Cores in `No clock`, `No power`, `No Debug Support` and `Reset` state are considered as running by the TCF framework.

Examples

Type "help coreconfig" to see examples.

Arguments

Synopsis	Description
`[-target] target`	**Specify target.** If missing, the current target is used.

Synopsis	Description
-general {safe|strict|bypass|unsafe|default}	General safety. Overwrites all previous settings
-default {safe|strict|bypass|unsafe|default}	Default Safety
-mget {safe|strict|bypass|unsafe|default}	Data Memory Read Safety
-mset {safe|strict|bypass|unsafe|default}	Data Memory Write Safety
-dasm {safe|strict|bypass|unsafe|default}	Instruction Memory Read Safety
-breakpoint {safe|strict|bypass|unsafe|default}	Instruction Memory Write Safety
-l1dread {safe|strict|bypass|unsafe|default}	Level 1 Data Cache Read safety
-l1dwrite {safe|strict|bypass|unsafe|default}	Level 1 Data Cache Write safety
-l1iread {safe|strict|bypass|unsafe|default}	Level 1 Instruction Cache Read safety
-l1iwrite {safe|strict|bypass|unsafe|default}	Level 1 Instruction Cache Read safety
-l2read {safe|strict|bypass|unsafe|default}	Level 2 Cache Read Safety
-l2write {safe|strict|bypass|unsafe|default}	Level 2 Cache Write Safety
-quiet	Return instead of printing.
{safe|strict|bypass|unsafe|default}	Same as "-general"

3.6.4. coreconfig stateupdate — Enable / Disable Core State Update

Synopsis

```
coreconfig stateupdate [-target] target ?-interval interval? ?-quiet? ?-nocomplain? ?enabled? ?interval?
```

Description

Control or query the automatic core state update.

By default all cores are monitored.

Examples

Disable state polling for current core (group):

```
% coreconfig stateupdate off
```

Disable state polling for LRT:

```
% coreconfig stateupdate LRT off
```

Disable state polling for All cores:

```
% coreconfig stateupdate A off
```

Enable state polling for Shave Cores:

```
% coreconfig stateupdate -target SALL true
```

Enable state polling for Leon Cores:

```
% coreconfig stateupdate LALL on
```

Enable state polling for Shave Cores, set base interval to 1 second:

```
% coreconfig stateupdate SALL on 1000
```

Set fast state polling interval for LOS

```
% coreconfig stateupdate LOS -interval 50
```

Arguments

Synopsis	Description
`[-target] target`	**Specify target.** If missing, the current target is used.
`-interval interval`	**Specify base update interval in ms.** The base update interval is the time between core state updates when the core state is Suspended. Unpowered/Unclocked state is updated 10x slower and Running state is updated 10x faster. `default` is 100ms
`-quiet`	**Return instead of printing.**
`-nocomplain`	**Don't throw error if target does not exist.**
`enabled`	**set state (optional)** A Tcl boolean value is expected, e.g. `0`, `1`, `false`, `true`, `on`, `off` etc. If missing, current state will be returned.
`interval`	**Same as -interval argument** This parameter was added for convenience.

3.7. cpr — CPR management.

Synopsis

```
cpr ?{enable|disable|list|sys|aux|pll|island|setSysFreq|auxClockSet|reset|boot|summary|grep|help}? ?args?...
```

Description

Control over CPR module.

The following subcommands are supported:

```
SUBCOMMAND, VARIANTS                          DESCRIPTION
```

```
cpr enable                                    Enables given clock.
cpr disable                                   Disables given clock.
cpr list <css|mss|sipp|upa|aux>               List all clock ids by
                                              group.
cpr sys                                       Display system clock
                                              status.
cpr aux                                       Display Aux Clock
                                              Status.
cpr pll                                       Display State of pll.
cpr island <status|enable|disable>            Power Island manipulation.
cpr setSysFreq <freqMhz>                      Use PLL0 and sys divider
                                              to achieve freqMhz.
cpr auxClockSet                               Setup CPR_IO_CLK1 from
                                              source REFCLK1 with
                                              8/16 divider.
cpr reset <show|list|listautodeasserting>     Show sticky resets
                                              /list all reset signals.
cpr reset <assert|deassert|pulse>             Alter a reset signal by
                                              constant name or by
                                              domain and number.
cpr boot <getMode|listModes>                  Either display currently
                                              latched boot mode, or
                                              list.
cpr summary                                   Display a summary of CPR
                                              clock and PLL status.
cpr help                                      Display help for CPR
                                              command.
```

Aliases

```
ALIAS       COMMAND
mclk        cpr
```

Examples

To enable CSS_I2C0 clock:

```
% cpr enable   CSS_I2C0
```

To enable CSS clock #20:

```
% cpr enable   css 20
```

To enable all myriad2 clocks:

```
% cpr enable   all
```

To disable CSS_I2C0 clock:

```
% cpr disable  CSS_I2C0
```

To disable CSS clock #20:

```
% cpr disable  css 20
```

To list all clock ids by group:

```
% cpr list css
```

To display system clock status:

```
% cpr sys
```

To display Aux Clock Status:

```
% cpr aux
```

To display State of PLL 0:

```
% cpr pll 0
```

To display status of the power islands:

```
% cpr island status
```

To achieve freqMhz by using PLL0 and sys divider:

```
% cpr setSysFreq <freqMhz>
```

To setup CPR_IO_CLK1 from source REFCLK1 with 8/16 divider:

```
% cpr auxClockSet AUX_GPIO1 REFCLK1 8 16
```

To list all reset signals:

```
% cpr reset list
```

To alter a reset signal by constant name:

```
% cpr reset assert CSS_I2C0
```

To alter a reset signal by domain and number:

```
% cpr reset pulse css 10
```

To display currently latched boot mode:

```
% cpr boot status
```

Display a table of the possible GPIO configured boot modes:

```
% cpr boot listModes
```

To display a summary of CPR clock and PLL status:

```
% cpr summary
```

3.7.1. cpr aux — Display Aux Clock Status

Synopsis

```
cpr aux
```

3.7.2. cpr auxClockSet — Setup CPR_IO_CLK1 from source REFCLK1 with 8/16 divider

Synopsis

```
cpr auxClockSet clk src numerator denominator ?enable?
```

3.7.3. cpr boot — Either display currently latched boot mode, or list all boot modes

Synopsis

```
cpr boot {getMode|listModes|status}
```

3.7.4. cpr disable — Disable CPR.

Synopsis

```
cpr disable ?args?...
```

3.7.5. cpr enable — Enable CPR.

Synopsis

```
cpr enable ?args?...
```

3.7.6. cpr grep — Search for clock name by regular expression

Synopsis

```
cpr grep ?queryRegex?
```

Description

Only implemented for MA2x8x.

3.7.7. cpr help, gpio help, mutex help —

Synopsis

```
cpr help
```

```
gpio help
```

```
mutex help
```

3.7.8. cpr island — Power Island manipulation

Synopsis

```
cpr island {status|enable|disable} ?islandNum?
```

Arguments

Synopsis	Description
islandNum	**Specify the island number.**

3.7.9. cpr list — Lists all clock ids by group

Synopsis

```
cpr list subcmd
```

Arguments

Synopsis	Description
subcmd	**Groups of clock (css, mss, sipp, upa, aux... etc).**

3.7.10. cpr pll — Display State of pll

Synopsis

```
cpr pll {0|1|all}
```

3.7.11. cpr reset — Manage reset bits

Synopsis

```
cpr reset {show|list|listautodeasserting|assert|deassert|pulse} ?args?...
```

Description

- shows sticky resets
- lists all reset signals
- alters a reset signal by constant name or by domain and number

3.7.12. cpr setSysFreq — Use PLL0 and sys divider to achieve freqMhz

Synopsis

```
cpr setSysFreq freqMhz
```

Arguments

Synopsis	Description
freqMhz	**Specify the freqMhz**

3.7.13. cpr summary — Display a summary of CPR clock and PLL status

Synopsis

```
cpr summary
```

3.7.14. cpr sys — Displays system clock status

Synopsis

```
cpr sys
```

3.8. dasm — Disassemble instructions.

Synopsis

```
dasm ?-address address? ?-count count? ?-target target? ?-quiet? ?-opcodes?
```

Description

The command is used to disassemble instruction opcodes at a particular address.

- It uses the current program counter as default starting address.
- The number of disassembled instructions is 10 by default .

This behaviour can be customised with switches.

Notes

- The disassembled instructions will be shown from the target's point of view, including the contents of the L1/L2 cache.
- Software breakpoint instructions are not visible in the disassembly.

Examples

Disassemble instructions (10, starting from PC):

```
% dasm
```

Disassemble instructions starting from 0x70000000:

```
% dasm 0x70000000
```

```
% dasm -address 0x70000000
```

Disassemble 15 instructions starting from 0x70000000:

```
% dasm 0x70000000 15
```

```
% dasm -address 0x70000000 -count 15
```

Disassemble instructions starting from `main`:

```
% dasm main
```

Disassemble 15 instructions (from current PC):

```
% dasm 15
```

```
% dasm -count 15
```

Disassemble 15 Leon OS instructions starting from 0x70000000:

```
% dasm -count 15 -target LOS -address 0x70000000
```

```
% dasm -t LOS 0x70000000 15
```

Arguments

Synopsis	Description
`-address address`	**Disassemble instructions starting from address** By default the current PC (program counter) is used.
`-count count`	**Number of instructions** The default is 10.
`-target target`	**Select target for disassembly.** If not specified, uses the current target.
`-quiet`	**Dont' print/format; return list of address/instruction pairs** Normally the command prints its result and returns nothing. This switch causes the output to be omitted and returned as a Tcl list of dictionaries.
`-opcodes`	**Show opcodes.** Show opcodes in addition to decoded instruction.

3.9. ddrinit — Initialise DDR.

Synopsis

```
ddrinit
```

Description

The command uses the TCF Platform service's ddrInit command to initialise the DDR component of the target.

Notes

- DDR init is not required for moviSim.
- access to uninitialised DDR addresses can cause the target's bus(ses) to lock up, causing undefined behaviour (JTAG FlowControlError and such).
- DDR initialisation is also carried out automatically when loading ELF files having code/data/BSS sections in the DDR.
- it is **not** recommended to initialise the DDR **after** having loaded an ELF executable on the target.
- DDR initialisation is done by loading and executing a separate, platform-dependent ELF. These files can be found in the "common/moviDebug/ddrinit" directory of the moviTools distribution. The DDR initialisation file can be custom overridden by a command-line switch of the moviDebug2 executable.

Examples

Initialise the DDR:

```
% ddrinit
```

3.10. displaystate — Control context state display

Synopsis

```
displaystate ?{on|off|verbose}?
```

Description

Control the display of console messages on state change.

Examples

Enable essential messages:

```
% displaystate on
```

Disable all messages:

```
% displaystate off
```

Enable all messages:

```
% displaystate verbose
```

Query current level:

```
% displaystate
```

Arguments

Synopsis	Description
`{on|off|verbose}`	**Verbosity level** Possible values: * `on` - Enable essential messages. * `off` - Disable all messages. * `verbose` - Enable all messages This parameter is optional. When missing, the command will return the current verbosity level.

3.11. eval% — Pseudo-interactive script evaluation

Synopsis

```
::mdbg::eval% script ?args...?
```

Description

This facility was added to provide for non-interactive script debugging.

The executed script is echoed to the standard output

along with a virtual prompt showing the current target.

The result of the command is returned to the caller

but it's also echoed back to the standard output.

The namespace path is temporarily altered to include the `::mdbg` namespace.

This allows better access to moviDebug commands within scripts.

Examples

Sample script (example.tcl)

```
# Set up convenient shortcut for command
interp alias {} % {} ::mdbg::eval%
```

```
    % breset
    % loadfile $elf
    # Set a breakpoint, store value
    set bpid [
        % breakpoint add -location main
    ]
    % runw
    % breakpoint remove $bpid
    % cont -wait
    # Avoid substitution with braces:
    % {
        breset; loadfile $elf; run; wait
    }
Run command:
```

```
    moviDebug2 -D:elf=output/001_HelloWorldLeon.elf --script example.tcl
Output:
```

```
...
P0:AA% breset
...
P0:AA% loadfile output/001_HelloWorldLeon.elf
P0:ALOS
...
P0:ALOS% breakpoint add -location main
#1
...
P0:ALOS% runw
...
P0:ALOS% breakpoint remove #1
...
P0:ALOS% cont -wait
...
P0:AA% breset; loadfile $elf; run; wait
...
```

3.12. getexitcode — Gets the exit code for the application

Synopsis

```
getexitcode
```

Description

The command gets the exit code for the application.

It works if current target is a Leon Core.

Aliases

- getpc
- getip
- programcounter

3.13. getprogramcounter — Get Program Counter.

Synopsis

```
getprogramcounter ?core?
```

Description

Get the Program Counter's value for target (a.k.a Instruction Pointer).

Target should be a Myriad Core.

Notes

This command is deprecated.

Use "state -pc" instead.

Examples

Return the value of PC register of Leon OS:

```
% getprogramcounter LOS
```

Return the value of IP register of Shave 0:

```
% getprogramcounter S0
```

Arguments

Synopsis	Description
`core`	**Target (core).** If missing, the current target is used.

3.14. gpio — GPIO pin state management

Synopsis

```
gpio {status|listmodes|findpins|set|toggle|configure|get|getraw|help} ?args?...
```

Description

Display state of, and configure the General-purpose input/output (GPIO) pins of the MA2x5x chip.

```
SUBCOMMAND, VARIANTS    DESCRIPTION
```

```
gpio status             Show GPIO pin status.
gpio set                Set output value in direct mode.
gpio configure          Configure a GPIO pin/range.
gpio listmodes          List all modes.
gpio get                Get the input value in direct mode.
gpio getraw             Display the raw status
gpio findpins           Show all GPIO pins starting with
                        'regExpression'
gpio toggle             Invert the output value
gpio help
```

Examples

Show the current state of all GPIOs:

```
% gpio status
```

Show the current state of gpio 22 to 35:

```
% gpio status 22..35
```

Set GPIO 19 high

```
% gpio set 19 1
```

Display the full gpio mode selection table:

```
% gpio get 12
```

Show the raw state of gpio 12 pin:

```
% gpio getraw 12
```

Show the state of the input cell for GPIO 12:

```
% gpio listmodes 12
```

Show the gpio pins which have a mode containing the string scl:

```
% gpio findpins scl
```

Advanced pin configurations:

Configure gpio 3 in mode 0x7 as an output with an output value of 1 (equivalent of gpio set 3 1):

```
% gpio configure 3 -mode 7 -out 1 -direction out
```

Configure gpios 3 to 7 in mode 0x7 as an output with an output value of 1 (equivalent of gpio set 3 1):

```
% gpio configure 3..7 -mode 7 -out 1 -direction out
```

Configure gpio 40 to 45 to output 1 with drive strenth 12 mA; voltage 1.8 V; Weak pullup enabled; Schmitt trigger on; wakeup disabled; and reciever enabled:

```
% gpio configure 40..45 -mode 7 -out 1 -direction out -drive 12 -voltage 1.8 -res PUP -schmitt on -wakeEn 0 -ren 1
```

3.14.1. gpio configure — Configure a GPIO pin.

Synopsis

```
gpio configure ?gpioNumber[..gpioNumber]? ?-mode {0|1|2|3|4|5|6|7}? ?-direction {in|out}? ?-drive {2|4|8|12}? ?-
voltage {1.8|3.3}? ?-res {BUS_KEEPER|PUP|PDOWN|NO_PULL}? ?-schmitt {on|off}? ?-wakeEn {0|1}? ?-ren {0|1}? ?-local
{0|1}? ?-out {0|1}? ?-data_inv {0|1}? ?-oe_inv {on|off}? ?-slew {fast|slow}? ?-json json?
```

Description

Configures one or more of the parameters for GPIO pin (mode, direction, drive, voltage, res, schmitt, wakeEn, ren, out value).

Examples

To configure the parameters of GPIO 40

```
% gpio configure 40 -mode 7 -out 1 -direction out -drive 4 -voltage 1.8 -res PUP -schmitt on -wakeEn 0 -ren 1 -local
1
```

To configure the parameters of the GPIOs pins between 40 and 45

```
% gpio configure 40..45 -mode 7 -out 1 -direction out -drive 12 -voltage 1.8 -res PUP -schmitt on -wakeEn 0 -ren 1
```

Arguments

Synopsis	Description
`gpioNumber[..gpioNumber]`	**Gpio number or range**
`-json json`	**Use JSON data** Set options from a JSON structure similar to an item of the status array. Replaces the functionality of the other options and as such it cannot be used together with them.

3.14.2. gpio findpins — Show all GPIO pins starting with 'regExpression'

Synopsis

```
gpio findpins ?-quiet? ?-json? regExpression
```

Description

Display all Gpio pins which had the mode starting with 'regExpression'

Examples

To display all Gpio pins which had the mode starting with 'cam_'

```
% gpio findpins ^cam_
```

Arguments

Synopsis	Description
`-quiet`	**Don't display; return list of dictionaries.** Normally the subcommand displays a table to the standard output and will leave no result in the Tcl interpreter. This flag inhibits the display and causes the command to return a result usable for further processing by Tcl scripts. WARNING: Format of returned data is not definitive and subject to change without notice.
`-json`	**Don't display; return JSON data.** Same as `-quiet` but the output format is JSON instead of Tcl dictionaries. WARNING: The format of the returned data is not 100% definitive. Cannot be used with `-quiet`.

3.14.3. gpio get — Get the input value in direct mode.

Synopsis

```
gpio get gpioNumber
```

Description

Configures the GPIO pin to input in direct mode and get the input value.

Examples

To configure the GPIO 40 to input in direct mode and get the input value.

```
% gpio get 40
```

Arguments

Synopsis	Description
gpioNumber	**Gpio number**

3.14.4. gpio getraw — Display the raw status

Synopsis

```
gpio getraw gpioNumber
```

Description

Read the raw status bit for the GPIO without changing its configuration.

Examples

To display raw status of GPIO 2

```
% gpio getraw 2
```

Arguments

Synopsis	Description
gpioNumber	**Gpio number**

3.14.5. gpio listmodes — List all modes

Synopsis

```
gpio listmodes ?-quiet? ?-json? ?gpioNumber[..gpioNumber]?
```

Description

Display all modes for the GPIO pin.

Examples

To display the modes for GPIO 0.

```
% gpio listmodes 0
```

To display the modes for GPIOs between 40 and 45

```
% gpio listmodes 40..45
```

Arguments

Synopsis	Description
`-quiet`	**Don't display; return list of dictionaries.** Normally the subcommand displays a table to the standard output and will leave no result in the Tcl interpreter. This flag inhibits the display and causes the command to return a result usable for further processing by Tcl scripts. WARNING: Format of returned data is not definitive and subject to change without notice.
`-json`	**Don't display; return JSON data.** Same as `-quiet` but the output format is JSON instead of Tcl dictionaries. WARNING: The format of the returned data is not 100% definitive. Cannot be used with `-quiet`.
`gpioNumber[..gpioNumber]`	**Gpio number or range**

3.14.6. gpio set — Set output value in direct mode.

Synopsis

```
gpio set ?gpioNumber[..gpioNumber]? newValue
```

Description

Configures the GPIO pin to output in direct mode and set the output value.

Examples

To configure the GPIO 40 to output in direct mode and set the output value to 1.

```
% gpio set 40 1
```

To configure the GPIOs between 40 and 45 to output in direct mode and set the output value to 0.

```
% gpio set 40..43 0
```

Arguments

Synopsis	Description
`gpioNumber[..gpioNumber]`	**Gpio number or range**

3.14.7. gpio status — Show GPIO pin status.

Synopsis

```
gpio status ?-quiet? ?-json? ?gpioNumber[..gpioNumber]?
```

Description

Displays the status of GPIO pin configuration.

Examples

Display state of all pins:

```
% gpio status
```

Display status of the first GPIO pin:

```
% gpio status 0
```

To display status for pins from 0 to 10 (inclusive):

```
% gpio status 0..10
```

Arguments

Synopsis	Description
`-quiet`	**Don't display; return list of dictionaries.** Normally the subcommand displays a table to the standard output and will leave no result in the Tcl interpreter. This flag inhibits the display and causes the command to return a result usable for further processing by Tcl scripts. WARNING: Format of returned data is not definitive and subject to change without notice.
`-json`	**Don't display; return JSON data.** Same as `-quiet` but the output format is JSON instead of Tcl dictionaries. WARNING: The format of the returned data is not 100% definitive. Cannot be used with `-quiet`.
`gpioNumber[..gpioNumber]`	**Gpio number or range**

3.14.8. gpio toggle — Invert the output value

Synopsis

```
gpio toggle ?gpioNumber[..gpioNumber]?
```

Description

The behaviour is:

```
if (gpio configured as output in mode 0x7)
    gpio set gpionNum inverse(previous gpioOutputSetting)
else
    gpio set gpioNum inverse(current gpioRawValue)
```

Examples

To invert the output value for GPIO 2

```
% gpio toggle 2
```

To invert the output value for GPIOs between 40 and 45

```
% gpio toggle 40..45
```

Arguments

Synopsis	Description
gpioNumber[..gpioNumber]	**Gpio number or range**

3.15. halt — Break execution of target(s).

Synopsis

```
halt -a[ll] ?-target target? ?-async? ?-timeout timeout? ?-temporary ?script? ? ?-script script? ?target?
```

Description

The command sends the Suspend command to the specified target (or the current one).

Optionally it can be asynchronous, which means that it will not wait for the target(s) to really suspend.

Aliases

```
ALIAS   FULL FORM
bn      halt
```

Examples

Suspend all contexts. Wait until they stop:

```
% halt -all
```

Suspend Leon OS:

```
% halt LOS
```

Safely refresh and then print the list of running RTEMS threads:

```
% halt -temporary { state -children [target]:RTEMS }
```

Arguments

Synopsis	Description
`-a[ll]`	**Suspend all contexts.** Iterates all running TCF RunControl contexts and send the `suspend` command to them. Waits until none of them is running. Can be given a timeout. **Notes** * recently changed to be synchronous by default.
`-target target`	**Select target to halt.** See `target` below.
`-async`	**Do not wait for the target to suspend.** Cannot be used with `-timeout`.
`-timeout timeout`	**Throw exception if `timeout` milliseconds elapse before suspending the core.** Normally the operation waits indefinitely. This switch allows the operation to have a timeout. This is especially useful for scripted scenarios. Cannot be used with `-async`.
`-temporary ?script?`	**Resume immediately after halt** This is useful to get a refresh on the list of currenty running RTEMS threads. Optionally can execute `script` while halted. The `script` parameter cannot be given when the `-script` option is also present.
`-script script`	**Script to run while temporarily halted.** Requires the `-temporary` option. Cannot be used when the `script` parameter is given to the `-temporary` option.
`target`	**Specify target of command.** If missing, the command uses the current target. Same as argument of `-target` option.

3.16. help — Show help

Synopsis

```
help ?-short? ?-version? ?command? ?subcommand? ?args?...
```

Arguments

Synopsis	Description
`-short`	**Only show short descriptions.**
`-version`	**Show moviDebug library version.**
`command`	**Show help for specified command.** If missing, all simple commands will be listed.
`subcommand`	**Show help for subcommand of `command` argument.** Can be glob pattern, e.g *.
`args...`	**Extra arguments. Ignored.** This allows the user to prepend `help` to a full command line, without requiring to erase the extra provided arguments. In the future the switches might be analysed to provide more specific information.

3.17. hist — Execution history.

Synopsis

```
hist ?-target target? ?-count count? ?-quiet? ?-pc? ?-opcodes? ?-dont-decode? ?target?
```

Description

Displays execution history for selected target.

Note

For SHAVEs the last item is the current instruction.

Examples

Displays history from current target:

```
% hist
```

Displays the last 5 instruntion from current target:

```
% hist 5
```

Displays the last 10 instructions and their opcode from LOS:

```
% hist -target LOS -count 10 -opcode
```

Arguments

Synopsis	Description
`-target target`	**Select target.** Default is current target.
`-count count`	**Maximum count of instructions.**
`-quiet`	**Don't display, just return values.**
`-pc`	**Only show program counter.**
`-opcodes`	**Show opcodes.**
`-dont-decode`	**Omit extra column for decoded instructions.** Used when `-opcodes` also specified.
`target`	**Same as -target.** Added for convenience. Cannot be used together with `-target`

3.18. jtag — Direct JTAG interface.

Synopsis

```
jtag  subcommand ?args?...
```

Description

Provides interface to various JTAG operations.

See `help jtag *` for list of subcommands.

3.18.1. jtag change32 — Change word (32-bit).

Synopsis

```
jtag change32 address operator expr...
```

Description

Combine read-modify-write operation in a single command.

Examples

Enable the LEON OS and RT DSU by setting the corresponding bits in the register:

```
% jtag change32 CPR_GEN_CTRL_ADR |= 1 << 15 | 1 << 17
```

Disable the LEON OS and RT DSU by clearing the same bits:

```
% jtag change32 CPR_GEN_CTRL_ADR &= ~(1 << 15 | 1 << 17)
```

Arguments

Synopsis	Description
`address`	**address or register name +offset**
`operator`	**modify operator** One of: `|=`, `&=`, `^=`, `+=`, `-=`, `*=`, `/=`, `<<=`, `>>=`
`expr...`	**value or expression to apply** Can be a numeric value or any Tcl `expr` expression.

3.18.2. jtag fillBlock — Fill block.

Synopsis

```
jtag fillBlock address size pattern
```

3.18.3. jtag formatBlock — Format+write arguments.

Synopsis

```
jtag formatBlock format address size arguments...
```

3.18.4. jtag get32 — Read word (32bit).

Synopsis

```
jtag get32 address
```

Arguments

Synopsis	Description
`address`	**address or register name +offset**

3.18.5. jtag get64 — Read double word (64-bit).

Synopsis

```
jtag get64 address
```

Arguments

Synopsis	Description
address	**address or register name +offset**

3.18.6. jtag getBlock — Read (and scan) block.

Synopsis

```
jtag getBlock ?-scan binFmt? address size
```

Arguments

Synopsis	Description
?-scan binFmt?	**Format specifier compatible with Tcl built-in `binary format`**

3.18.7. jtag getBurst32 — Burst read 32-bit words.

Synopsis

```
jtag getBurst32 address numWords32
```

3.18.8. jtag ir — Send JTAG instruction register value.

Synopsis

```
jtag ir number
```

3.18.9. jtag lock — Lock JTAG access during script

Synopsis

```
jtag lock script
```

Description

Prevent other threads from accessing the JTAG interface during given script.

Normally the TCF thread will periodically access the JTAG interface. This can be prevented by moving select operations inside a `jtag lock` block.

Notes

- Scripts should be short and mostly using `jtag` subcommands!
- UART polling and pipe processing are performed on the same thread, so either disable UART and remove all pipes or avoid any Tcl event handling inside the script. That is, don't call `update`, `wwait`, `wait`.
- Invoking TCF-related commands within the script will cause the system to lock up because the TCF thread will probably be hung waiting on the JTAG lock held by this thread.
- Do not use the `mget` command in the script (except maybe `mget -jtag -reg XXXX`), this command will talk to TCF and it also does Tcl event processing while doing so.
- Use `jtag get32`, `jtag set32` and `jtag change32` commands to access registers
- Use the simple `after` command if you want to introduce delays inside the lock script.

Examples

A kind a software reset implemented manually:

```
jtag lock {
    # Set SRST first physical low then physical high
    foreach state {low high} {
        jtag pins -srst $state
    }
    # Wait 50 ms without event processing
    after 50
    # Send IR sequence
    foreach cmd {0x1 0xe 0xe} {
        jtag ir $cmd
    }
    # Enable Leon OS and RT DSU
    jtag change32 CPR_GEN_CTRL_ADR |= 1<<15 | 1 <<17
}
```

Arguments

Synopsis	Description
`script`	**Block of commands to execute within exclusive access block.**

3.18.10. jtag pins — Set JTAG pins state.

Synopsis

```
jtag pins ?-trst high|low|toggle? ?-srst high|low|toggle? ?-led high|low|toggle?
```

Arguments

Synopsis	Description
`?-trst high|low|toggle?`	**Change TRST pin state.**
`?-srst high|low|toggle?`	**Change SRST pin state.**
`?-led high|low|toggle?`	**Change LED pin state.**

3.18.11. jtag readBlock — Read block.

Synopsis

```
jtag readBlock address size
```

3.18.12. jtag scan — Scan JTAG register bits.

Synopsis

```
jtag scan dr|ir count pattern ...
```

Description

Shifts bits of JTAG register in and out.

The bits can be specified in groups of either binary or hexadecimal format.

The concatenated patterns will be shifted in least significant bit first.

The result will be in exactly the same format as the input.

Arguments

Synopsis	Description
`dr|ir`	**JTAG register.**
`count`	**Total bit count.** This value is checked against the bit-length of the pattern list provided. An error is generated on mismatch.

Synopsis	Description
`pattern ?...?`	**Bit patterns.** One or more binary or hexadecimal bit patterns. (Given as separate arguments). Binary patterns must begin with `0b`, hex patterns with `0x`. The number of digits in the pattern will determine the number of bits. (One hex digit equals four bits.) The big endian bit patterns are concatenated and shifted in least significant bit first. The result will be a list of patterns having the same format.

3.18.13. jtag scanBlock — Read+scan block into variables.

Synopsis

```
jtag scanBlock format address size varnames...
```

3.18.14. jtag set32 — Write word (32bit).

Synopsis

```
jtag set32 address value32
```

Arguments

Synopsis	Description
`address`	**address or register name +offset**
`value32`	**32-bit integer value**

3.18.15. jtag set64 — Write double word (64-bit).

Synopsis

```
jtag set64 address value64
```

Arguments

Synopsis	Description
`address`	**address or register name +offset**

Synopsis	Description
value64	**64-bit integer value**

3.18.16. jtag setBlock — (Format data and) write block.

Synopsis

```
jtag setBlock ?-format binFmt? address data
```

Arguments

Synopsis	Description
?-format binFmt?	**Format specifier compatible with Tcl built-in `binary format`**

3.18.17. jtag setBurst32 — Burst write 32-bit words.

Synopsis

```
jtag setBurst32 address listOfWords
```

3.18.18. jtag writeBlock — Write block.

Synopsis

```
jtag writeBlock address data
```

3.19. linenumbers — Show line number information at address or source location

Synopsis

```
linenumbers ?-s[how-source]? ?-addr addr? ?-size size? ?-file file? ?-line line? ?-list-files ?pattern(1)? ? ?-no-dirs? ?-column column? ?-value? ?-quiet? ?-target target? ?-frame frame? ?-display? ?pattern(2)?
```

Description

Shows line number information as present in .debug_lines section.

Either an address or a source file/line combination should be provided.

Aliases

```
ALIAS      FULL FORM
linfo      linenumbers
lnum       linenumbers
```

Examples

Show line number debug information for the current instruction

```
% linenumbers
```

Show line number debug information for the address 0x70181328:

```
% linenumbers -addr 0x70181328
```

Show line number debug information for the return address

```
% linenumbers -frame 1
```

Show line number debug information for the current instruction of Shave 0

```
% linenumbers -target S0
```

Show line number debug information for line 42 of main.cpp:

```
% linenumbers -file main.cpp -line 42
```

Return the address mapped to line 42 of main.cpp:

```
% linenumbers -file main.cpp -line 42 -value
```

List all known C++ source files having lines mapped to Shave 0:

```
% linenumbers -target S0 -list-files *.cpp
```

Arguments

Synopsis	Description
`-s[how-source]`	**List actual source lines if possible.**

Synopsis	Description
`-addr addr`	**Start address** Retrieve line number information at address.
`-size size`	**Size of code range at address** Retrieve line number information for all addresses in range, starting from `-addr`, sized `-size`. Default is 1.
`-file file`	**File name** Retrieve line number information for file. Requires `-line` to be specified as well. Cannot be used together with `-addr`.
`-line line`	**Line number** Retrieve line number information at line. Requires `-file` to be specified as well. Cannot be used together with `-addr`.
`-list-files ?pattern(1)?`	**Retrieve list of source files** `pattern() is a glob-style pattern to filter file names. e.g .c` Only one of the `pattern (*)` arguments can be given at a time. Cannot be used together with `-file`, `-addr`, `-s[how-source]`, `-value` and `-display`
`-no-dirs`	**Omit base directory from listing** Requires `-list-files`. Causes to display or return only source file names, not their base directories.
`-column column`	**Column.** Retrieve line number information for this column.
`-value`	**Do not display. Return first value as result.** Returns either a numeric address, or a {filepath line} pair.
`-quiet`	**Do not display. Return TCF structure.**
`-target target`	**Select target for operation** Default is current target.
`-frame frame`	**Select stack frame for operation** Use current program counter on specified frame as address. If missing, the top frame (actual current PC) is used. Cannot be used together with `-addr`, `-file` or `-list-files`.

Synopsis	Description
`-display`	**When "-value" requested, also do a full display.** This is the default behaviour, except when `-value` is also specified.
`pattern(2)`	**File name pattern for list-files.** Provided for convenience. Requires `-list-files`. Only one of the `pattern(*)` arguments can be given at a time.

3.20. listsource — List source code.

Synopsis

```
listsource ?-target target? ?-frame frame? ?-file fileName? ?-start startLine? ?-end endLine? ?fileName? ?[-
]startLine? ?[+/-]endLine?
```

Description

Display source code listing for the specified file

or the current target's current source file.

Aliases

```
ALIAS   FULL FORM
lsrc    listsource
```

Examples

List the source code between line 20 and 40 from current source file:

```
% listsource 20 40
```

```
% listsource -start 20 -end 40
```

List the code for the current line and the next 5 lines:

```
% listsource +5
```

```
% listsource -end +5
```

List the code for the 5 lines before, and the current source line:

```
% listsource -start -5
```

```
% listsource -5
```

List the code for the 2 lines before, the current line, and the next 3 lines:

```
% listsource -start -2 -end +2
```

```
% listsource -2 +2
```

List the all the source code for the file:

```
% listsource -file d:/mdk/examples/HowTo/Cpp/cppshave/leon/main.cpp
```

```
% listsource d:/mdk/examples/HowTo/Cpp/cppshave/leon/main.cpp
```

List the source code for lines 10 to 40 of file:

```
% listsource -start 10 -end 40 -file leon/main.cpp
```

```
% listsource -start 10 -end 40 leon/main.cpp
```

```
% listsource -file leon/main.cpp -start 10 -end 40
```

```
% listsource -file leon/main.cpp 10 40
```

```
% listsource leon/main.cpp 10 40
```

List the source code from line 20 to the end of file:

```
% listsource -file leon/main.cpp -start 20 -end end
```

```
% listsource leon/main.cpp 20 end
```

Arguments

Synopsis	Description
`-target target`	**Select target for operation** Default is current target. Cannot be used with `fileName`.
`-frame frame`	**Select stack frame for operation** Use current program counter on specified frame as address. If missing, the top frame (actual current PC) is used. Cannot be used with `fileName`.
`-file fileName`	+ See `fileName` below.
`-start startLine`	+ See `[-]startLine` below.
`-end endLine`	+ See `[-]endLine` below.
`fileName`	**Source file name.** If missing, will use current target's current source file.
`[-]startLine`	**Index of first line.** It can be determined automatically for the current source file. The default is 1. A `-` prefix makes it a value relative to current source line.
`[+/-]endLine`	**Index of last line (abs/rel).** A `+` or `-` prefix makes it a relative number. `end` is treated specially and it means the last line in the file. It can be determined automatically for the current source file. The default is +0 when fileName is not present.

3.21. loadfile — File load/verify/run operations.

Synopsis

```
loadfile ?-elf? ?-verify? ?-symbols-only? ?-startupcore startupCore? ?-quiet? ?-binary? ?-hex? ?-chan? ?-address
baseAddress? ?-noload? ?-async? ?-breset? ?-ddrinit? ?-run? ?-wait? ?-stopatmain? ?-timeout timeout? ?-
[B[D]][R[S][W]]? fileNameOrChannel
```

Description

Allows loading files into the target platform.

The loaded ELF file can also be started right away and the command can be run synchronously or asynchronously.

- Synchronous operation (default): The command will not normally exit until the target suspends/starts execution.
- Asynchronous operation (`-async` flag): the command returns immediately without waiting for its effect to become active.

Notes

- Currenty only ELF and BIN loading is implemented.
- Loading always happens using JTAG interface, cache coherency is not managed.
- The startup core needs to be specified before ELF loading, because the entry point is set during this command.
- Main startup core get started up / reset in the Myriad2 TCF model automatically.
- If the ELF contains Shave window address settings, the corresponding shaves are started up.
- Breakpoints always get reevaluated when the symbol information becomes (un)available.
- `loadfile -run` is implemented more efficiently than `load` followed by `run`.

Aliases

```
ALIAS           COMMAND
loadelf         loadfile -elf
loadandrun      loadfile -elf -run
verify          loadfile -verify -noload
loadandverify   loadfile -verify
loadsym         loadfile -elf -symbols-only
```

Examples

Load my ELF file:

```
% loadfile myElfFile.elf
```

```
% loadfile d:/mdk/examples/HowTo/Cpp/cppshave/output/cppshave.elf
```

```
% loadfile {d:\mdk\examples\HowTo\Cpp\cppshave\output\cppshave.elf}
```

Load and verify my ELF file:

```
% loadfile -verify output/myElfFile.elf
```

Load binary data to address 0x70000000:

```
% loadfile -address 0x70000000 myFile.bin
```

Reset board, initialise DDR, load myElfFile.elf, verify, run, wait for stop:

```
% loadfile -elf -breset -ddrinit -verify -run -wait myElfFile.elf
```

Board Reset, DDR Init, Load, Run, Wait:

```
% loadelf -BDRW myElfFile.elf
```

Only load symbols information and memory maps from myElfFile.elf:

```
% loadfile -elf -symbols-only myElfFile.elf
```

Arguments

Synopsis	Description
`-elf`	**Treat `fileNameOrChannel` as name of ELF file.** Normally the file type can be detected from its extension. This switch also enables a lot of extra flags: `-breset`, `-ddrinit`, `-runw`, `-[B[D]][R[W]]`
`-verify`	**Verify loaded bytes.** Verifies loaded data. Returns error on failure
`-symbols-only`	**Only load symbol information.** This switch allows the user to load symbol information for the target, without loading the program/data bytes of an ELF file, effectively attaching to a running application. Incompatible with any file type other than ELF.
`-startupcore startupCore`	**Set startup core.** Select a core name to use as startup for Elf file Incompatible with any file type other than ELF.

Synopsis	Description
`-quiet`	**Don't display section names during load** Supresses messages during load.
`-binary`	**Treat `fileNameOrChannel` as name of file containing raw binary data.** Requires `-address` to be specified. Incompatible with other file type switches: `-elf`, `-hex`, `-chan`.
`-hex`	**Treat `fileNameOrChannel` as name of file containing HEX data.** HEX format is Movidius-specific. *!! NOT IMPLEMENTED !!* Incompatible with other file type switches: `-elf`, `-binary`, `-chan`.
`-chan`	**Treat `fileNameOrChannel` as name of Tcl channel.** *!! NOT IMPLEMENTED !!* Allows loading binary data directly from open Tcl channel. (e.g. pipe, socket) Incompatible with other file type switches: `-elf`, `-binary`, `-hex`.
`-address baseAddress`	**Specify base address for loading file data.** Address, symbol or register name. Especially for `-binary` and `-channel` types. It's also theoretically possible to load/reloc ELF .so files at `baseAddress`. This is not supported currently.
`-noload`	**Bypass loading of bytes on verify.** Allows verification of bytes without loading the data first. Requires `-verify` flag.
`-size loadSize`	**Override size of loaded data.** Allows overriding the size of raw bytes loaded. Normally the whole file/channel data is loaded (read until EOF). Requires `-chan` or `-binary` flag to be specified.
`-offset fileOffset`	**Override source offset of loaded data.** Allows overriding the size of raw bytes loaded. Normally data is read from beginning of file or current read offset of channel. This switch will cause a (forward) seek in the source channel (if possible) or skips the number of bytes equal to offset. Negative numbers should cause a negative seek if possible on the channel. Requires `-chan` or `-binary` flag to be specified.
`-async`	**Asynchronous operation.** For ELF file type, the load/-run can be done asynchronously, that means that the command will return immediately, not waiting for the target's state change. Incompatible with synchronisation switches `-wait`, `-runw`, `-timeout`. Incompatible with `-elf` flags `-breset`, `-ddrinit`, `-[B[D]][R[W]]`.

Synopsis	Description
`-breset`	**Reset board before load.** Issues `breset` command before loading. Requires `-elf` switch or `loadelf` alias.
`-ddrinit`	**Initialise DDR before load.** Issues `ddrinit` command before loading. Requires `-elf` switch or `loadelf` alias.
`-run`	**Run loaded elf.** Works even without specifying `-elf` flag. Ignored if file name not recognized as ELF. Synchronous by default. Can be executed asynchronously. Incompatible with other file type switches: `-binary`, `-hex`, `-chan`.
`-wait`	**Wait for the application to suspend.** The command will not return until the application stops for any reason. Requires `-run` flag.
`-runw`	**-run -wait**
`-stopatmain`	**Stop at main function** Works even without specifying `-elf` flag. For a POSIX RTEMS application the `main` function is the `POSIX_Init` function of the startup core. For non-POSIX RTEMS application the `main` function is the `Init` function of the startup core.
`-timeout timeout`	**Timeout (ms).** Raises error if command does not return normally in the specified number of milliseconds. Incompatible with `-async`.
`-[B[D]][R[S][W]]`	**[-breset [-ddrinit]] [-run [-stopatmain] [-wait]]** Short versions of long options. Not all combinations are allowed and the order of the characters is important. Short and long version of same option cannot be both specified. Incompatible with same other options as long version.
`fileNameOrChannel`	**File name or channel.** Specifies a file name or a Tcl channel name. This is a mandatory argument. What it means depends on the presence of `-elf`, `-binary`, `-hex`, `-chan` flags. If none of these are present, the CLI will try to determine the meaning of the name. NOTE: There is no default case for this detection logic.

3.22. mdump — Dump memory in a canonical hexdump-like format.

Synopsis

```
mdump ?-target target? ?-frame frame? ?-jtag? ?-mem memTarget? location count
```

Description

Display specified number of values from specified address in a human readable format.

Examples

Display 10 32-bit words from DDR in hexdump-like format (hexadecimal little endian 32-bit words + ASCII):

```
% mdump 0x70000000 10
```

Arguments

Synopsis	Description
`-target target`	**Operate on specific target.** If missing, current target is used.
`-frame frame`	**Specify reference stack frame.** Search for variable(s) on stack frame context #`frame`. Default is 0 (top frame) for `-var`, otherwise uses non-frame context.
`-jtag`	**Use direct JTAG interface for memory I/O.** This flag forces the command to bypass TCF's memory context infrastructure to do memory I/O. JTAG I/O is done directly using the `jtag` command which will use direct protocol commands from the `::mvproto` namespace. It is useful in certain scenarios where cache coherence is not a problem. Incompatible with `-mem` argument.
`-mem memTarget`	**Target for memory I/O.** Normally the memory operations go through the target context's associated memory context. This argument allows specifying a different context to do I/O through. Incompatible with `-jtag` flag.
`location`	**Memory location - address, register, symbol, variable or expression.** Start Address from where values are being displayed. The type will be tried to be detected automatically.

Synopsis	Description
`count`	**The number of words being displayed.** Must be an integer.

3.23. memmap — Show memory map for target

Synopsis

```
memmap ?-target? target ?-quiet? ?-first? ?-name pattern? ?-not-name notpattern? ?-xname regex? ?-not-xname
notregex? ?-bss? ?-not-bss? ?-exec? ?-not-exec? ?-read? ?-not-read? ?-write? ?-not-write? ?-at addr? ?-not-at
notaddr? ?target(2)?
```

Description

Display memory map for specified target or current target, as seen by TCF.

Examples

Display memory map for Leon OS:

```
% memmap LOS
```

Show all `.bss` sections for the entire application:

```
% memmap A -bss
```

Show all executable sections for the entire application:

```
% memmap A -exec
```

Show all non-BSS data sections for the entire application:

```
% memmap -not-bss -target A -not-exec
```

Show all Shave 0 code sections

```
% memmap -exec -target S0
```

Show the current core's section containing `POSIX_Init`:

```
% memmap -at POSIX_Init
```

Show Leon RT specific sections of the application (name starting with `.lrt.`):

```
% memmap A -name .lrt.*
```

Show '.text' and '.data' sections in the application (specified by regular expression):

```
% memmap A -xname {\.(text|data)$}
```

Arguments

Synopsis	Description
`?-target? target`	**Target name** If missing, will use current target.
`-quiet`	**Return instead of print** The result will be a list of dictionaries containing the section information. NOTE: Not all fields will be present in each list item.
`-name pattern`	**Only sections matching `pattern`**
`-not-name notpattern`	**Exclude sections matching `notpattern`**
`-xname regex`	**Only sections matching `regex` regular expression**
`-not-xname notregex`	**Exclude sections matching `notregex` regular expression**
`-bss`	**Only BSS sections**
`-not-bss`	**Exclude BSS sections**
`-exec`	**Only executable sections**
`-not-exec`	**Exclude executable sections**
`-read`	**Only readable sections**
`-not-read`	**Exclude readable sections**
`-write`	**Only writable sections**
`-not-write`	**Exclude writable sections**
`-at addr`	**Return sections containing address, symbol or register**
`-not-at notaddr`	**Exclude sections containing address, symbol or register**
`target(2)`	**Target name. Added for convenience** Same as `-target` option.

3.24. mfill — Fill memory location with pattern.

Synopsis

```
mfill ?location(1)? ?-size size? ?-zero? ?-offset srcSliceOffset? ?-length srcSliceLength? ?-format tclBinaryFormat?
?-target target? ?-frame frame? ?-jtag? ?-mem memTarget? ?location(3)? ?data?...
```

Description

Fill value of specified address, register, symbol, variable or expression with pattern.

Examples

Fill an array with the byte value 255:

```
% mfill my255Array -format c 255
```

Fill an array with the Ascii pattern `Hello!`:

```
% mfill myHelloArray "Hello!"
```

Fill an array with the pattern `0xCAFEBABE`:

```
% mfill myCafeArray -format i 0xCAFEBABE
```

Fill an array with the 4 byte pattern `0xDEAD`:

```
% mfill myDeadArray -format s 0xDEAD
```

Fill the first 256KiB of DDR with zeros, using the JTAG interface (faster):

```
% mfill -jtag -reg DDR_BASE_ADR -size [expr 256*1024] -zero
```

Fill the same area with the pattern `Hello!\\r\\n`:

```
% mfill -jtag -reg DDR_BASE_ADR -size [expr 256*1204] -format a*cc Hello! 13 10
```

Arguments

Synopsis	Description
`location(1)`	**Memory location - address, register, symbol, variable or expression (before switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` options is also given.
`-addr ?location(2)?`	**Treat `location()` as numeric address.*** One and only one of the `location(*)` arguments must be provided.
`-reg ?location(2)?`	**Treat `location()` as register name.*** One and only one of the `location(*)` arguments must be provided.
`-sym ?location(2)?`	**Treat `location()` as (global) symbol.*** One and only one of the `location(*)` arguments must be provided.
`-var ?location(2)?`	**Treat `location()` as (local) variable.*** One and only one of the `location(*)` arguments must be provided.
`-expr ?location(2)?`	**Treat `location()` as expression.*** One and only one of the `location(*)` arguments must be provided.
`-id ?location(2)?`	**Treat `location()` as TCF Symbol ID*** One and only one of the `location(*)` arguments must be provided.
`-size size`	**Specifiy fill size** Default is to fill the entire `location`.
`-zero`	**Fill with zero** Data is zeros. Cannot be used with `-offset`, `-length`, `-format` and `-data`
`-offset srcSliceOffset`	**Take data from offset `srcSliceOffset` of input.** Default is to take from beginning of the binary formatted input.
`-length srcSliceLength`	**Take only `srcSliceLength` bytes from input.** Default is to take all of the binary formatted input from ?srcSliceOffset? to end (see `-offset`).

Synopsis	Description
`-format tclBinaryFormat`	**Format non-binary input argument(s).** Normally the input data should already be in raw binary format. This switch allows formatting non-binary data using Tcl's built-in `binary format` command. The `tclBinaryFormat` argument is the same as for the aforementioned comand's `format` argument. The formatted binary input will be optionally sliced according to the `-offset` and `-length` arguments and used as fill pattern. Tcl `binary format` specfiers quick reference: `a` : Stores a byte string in the output string. Every character is taken as modulo 256 (i.e. the low byte of every character is used, and the high byte discarded) `A` : Same as `a`, except that spaces are used for padding instead of nulls. `b` : Stores a string of binary digits in low-to-high order within each byte in the output string. `B` : Same as `b` except that the bits are stored in high-to-low order within each byte. `H` : Stores a string of count hexadecimal digits in high-to-low within each byte in the output string. `h` : Same as `H` except that the digits are stored in low-to-high order within each byte. `c` : Stores one or more 8-bit integer values in the output string. `s` : Same as `c` except that it stores one or more 16-bit integers in little-endian byte order in the output string. `S`: Same as `s` except that it stores one or more 16-bit integers in big-endian byte order in the output string. `t`: (mnemonically `tiny`) same as `s` and `S` except that it stores the 16-bit integers in the output string in the native byte order of the machine. `i`: Same as `c` except that it stores one or more 32-bit integers in little-endian byte order in the output string. `I`: Same as `i` except that it stores one or more one or more 32-bit integers in big-endian byte order in the output string. `w`: Same as `c` except that it stores one or more 64-bit integers in little-endian byte order in the output string. `W`: Same as `w` except that it stores one or more one or more 64-bit integers in big-endian byte order in the output string. `m`: (mnemonically the mirror of `w`) is the same as `w` and `W` except that it stores the 64-bit integers in the output string in the native byte order of the machine where the Tcl script is running. `f`: Same as `c` except that it stores one or more one or more single-precision floating point numbers in the machine's native representation in the output string. `r`: (mnemonically `real`) same as `f` except that it stores the single-precision floating point numbers in little-endian order. `R`: Same as `r` except that it stores the single-precision floating point numbers in big-endian order. `d`: Same as `f` except that it stores one or more one or more double-precision floating point numbers in the machine's native representation in the output string. `q`: (mnemonically the mirror of `d`) is the same as `d` except that it stores the double-precision floating point numbers in little-endian order. `Q`: Same as `q` except that it stores the double-precision floating point numbers in big-endian order. `x`: Stores count `null` bytes in the output string. This type does not consume an argument. `X`: Moves the cursor back count bytes in the output string. If count is *

Synopsis	Description
`-target target`	**Operate on specific target.** If missing, current target is used.
`-frame frame`	**Specify reference stack frame.** Search for variable(s) on stack frame context #`frame`. Default is 0 (top frame) for `-var`, otherwise uses non-frame context.
`-jtag`	**Use direct JTAG interface for memory I/O.** This flag forces the command to bypass TCF's memory context infrastructure to do memory I/O. JTAG I/O is done directly using the `jtag` command which will use direct protocol commands from the `::mvproto` namespace. It is useful in certain scenarios where cache coherence is not a problem. Incompatible with `-mem` argument.
`-mem memTarget`	**Target for memory I/O.** Normally the memory operations go through the target context's associated memory context. This argument allows specifying a different context to do I/O through. Incompatible with `-jtag` flag.
`location(3)`	**Memory location - address, register, symbol. variable or expression (after switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` flags is also given.
`data...`	**Input value(s).** Normally it will be a binary/ascii string value, unless the `-format` option is also present. In this case there can be multiple arguments, but at least one must be present. See: `-format` option.

3.25. mget — Get value from memory location.

Synopsis

```
mget ?location(1)? ?-type typeclass? ?-size size? ?-value? ?-binary? ?-scan tclBinaryFormat? ?-asciiz? ?-count
count? ?-[struct]depth? ?-target target? ?-frame frame? ?-jtag? ?-mem memTarget? ?location(3)? ?count?
```

Description

Display or return value of specified address, register, symbol, variable or expression.

Aliases

```
ALIAS       COMMAND
mgetv       mget -value
mgetb       mget -binary
mgets       mget -asciiz
```

Examples

Display 32-bit unsigned value at some address:

```
% mget 0x14400010
```

Display the value of the LRT PC register:

```
% mget PC -target LRT
```

Display the value of the local variable called PC on LRT:

```
% mget -target LRT -var PC
```

Display the value of the global symbol called PC as the LRT sees it:

```
% mget -target LRT -sym PC
```

Display the values of 32 input registers of LRT as signed integers, starting from I0:

```
% mget I0 -reg -type int -count 32 -target LRT
```

Display the V0 register value of Shave 0 (will show four rows of values):

```
% mget V0 -target S0 -type float
```

Get PC register value and the value of the next 2 registers of LRT:

```
% mget pc -target lrt -count 3
```

Get the value of the I0 register as two half-precision floating point values:

```
% mget i0 -type float -size 2 -count 2
```

Return 8 double precision and 16 single precision LEON floating point register values (little endian) as a list of two lists, first element has 8 values, second has 16 values:

```
% mget f0 -size [expr 8*8+16*4] -binary -scan q8r16
```

Arguments

Synopsis	Description
`location(1)`	**Memory location - address, register, symbol, variable or expression (before switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` options is also given.
`-addr ?location(2)?`	**Treat `location()` as numeric address.*** One and only one of the `location(*)` arguments must be provided.
`-reg ?location(2)?`	**Treat `location()` as register name.*** One and only one of the `location(*)` arguments must be provided.
`-sym ?location(2)?`	**Treat `location()` as (global) symbol.*** One and only one of the `location(*)` arguments must be provided.
`-var ?location(2)?`	**Treat `location()` as (local) variable.*** One and only one of the `location(*)` arguments must be provided.
`-expr ?location(2)?`	**Treat `location()` as expression.*** One and only one of the `location(*)` arguments must be provided.
`-id ?location(2)?`	**Treat `location()` as TCF Symbol ID*** One and only one of the `location(*)` arguments must be provided.
`-type typeclass`	**Force type class for value.** Affects display/returned value (get) or binary format of scanned string value (set). The following type classes are recognized: int - signed integer 1, 2, 4, 8 bytes hex - unsigned hexadecimal integer 1, 2, 4, 8 bytes unsigned - unsigned decimal integer 1, 2, 4, 8 bytes float - floating point 2, 4, 8 bytes pointer - pointer 4, 8 bytes binary - raw binary string, any size asciiz - null terminated ASCII string or string pointer

Synopsis	Description
`-size size`	**Specifiy size of location.** This is usually used in conjuction with the `-type` argument, to override the detected size of memory location. NOTE: not all -type(class) / -size combinations are legal, see `-type`.
`-endian {little|big}`	**Force endiannes of value.** Only little-endian is supported.
`-value`	**Return value.** Only return the (formatted/structured) value of the memory location. NOTE: Supresses display by default, but this can be reenabled by the `-full` flag. Structures and arrays may be returned as Tcl dictionary/list hierarchies. Incompatible with `-quiet` flag.
`-binary`	**-type binary -value** Return binary value directly. Cannot be used in conjuction with `-type` and whatever the `-value` switch is incompatible with (e.g. `-quiet`).

Synopsis	Description
`-scan tclBinaryFormat`	**Scan binary value.** Allows scanning non-binary data using Tcl's built-in `binary scan` command. The `tclBinaryFormat` argument is passed along as `binary scan's` `format` argument. NOTE A list will be returned if the `binary scan` command would have required more than one variable as its remaining arguments. This switch requires the use of the `-binary` switch. Tcl `binary scan` specifiers quick reference: `a`: The data is a byte string. 'A': This form is the same as `a`, except trailing blanks and nulls are stripped from the scanned value before it is stored. `b`: The data is turned into a string of count binary digits in low-to-high order represented as a sequence of "1" and "0" characters. `B`: Same as `b`, except the bits are taken in high-to-low order within each byte `H`: The data is turned into a string of hexadecimal digits in high-to-low order. `h`: Same as `H`, except the digits are taken in reverse (low-to-high) order within each byte `c`: The data is turned into 8-bit signed integers and stored as a list. `s`: The data is interpreted as (a list of) 16-bit signed integers represented in little-endian byte order. `S`: Same as `s` except that the data is interpreted as count 16-bit signed integers represented in big-endian byte order. `t`: The data is interpreted as 16-bit signed integers represented in the native byte order. `i`: The data is interpreted as 32-bit signed integers represented in little-endian byte order. `I`: Same as `i` except that the data is interpreted as count 32-bit signed integers represented in big-endian byte order. For example, `n`: The data is interpreted as 32-bit signed integers represented in the native byte order of the machine running the Tcl script. `w`: The data is interpreted as 64-bit signed integers represented in little-endian byte order. `W`: Same as `w` except that the data is interpreted as count 64-bit signed integers represented in big-endian byte order. `m`: The data is interpreted as 64-bit signed integers represented in the native byte order of the machine running the Tcl script. `f`: The data is interpreted as single-precision floating point numbers in the machine's native representation. `r`: Same as `f` except that the data is interpreted as count single-precision floating point number in little-endian order. `R`: Same as `f` except that the data is interpreted as count single-precision floating point number in big-endian order. `d`: Same as `f` except that the data is interpreted as count double-precision floating point numbers in the machine's native representation. `q`: Same as `d` except that the data is interpreted as count double-precision floating point number in little-endian order. `Q`: Same as `d` except that the data is interpreted as count double-precision floating point number in big-endian order. `x`: Moves the cursor forward `count` bytes in string. If `count` is * or is

Synopsis	Description
`-asciiz`	**-type asciiz -value** Return value as decoded ASCII string. Location will be treated as if start address of a null terminated string, EXCEPT when it's detected to be a pointer, in this case it will be dereferenced. Location is read until the null terminator or the limit is reached. The `-size` option limits the size of the returned string. Cannot be used in conjuction with `-binary` switch.
`-count count`	**Number of items beginning at location.** Default is 1. For scalars and structures, if specified, `count-1` items following the specified location will also be retrieved. For arrays, it will retrieve the first `count` elements of the array. NOTES: Causes the return value to be a list! Some local vector register names are configured to set this option automatically! `-structdepth` is adjusted to at least 1 when this is specified for an array variable. Cannot be used together with final `count` argument.
`-[struct]depth`	**Structure fields/Array items recursion limit.** Default is 1. Determines how deeply recursively the display of a structure/array should be performed. NOTE: ALL OF THE DATA is retrieved first and this only affects the display!
`-quiet`	**Don't print anything. Return data as Tcl structure (list/dictionary).** Normally the command shows a nice tabular display of the information and does not return a value. This can be suppressed and the machine-readable format will be returned instead. NOTES: This option is for advanced cases and internal reuse. The `-value` flag is better suited for typical use-cases. Specifying the `-count` option makes the return value a list of dictionaries, even when `count` is 1. This also applies for some vector registers where's an implicit count.

Synopsis	Description
`-full`	**Display full information.** This is the default behaviour, so this switch only needs to be explicitly given when `-value` is also present. This option is for scripting, when value is needed but the table should be also displayed.
`-A`	**-value -addr**
`-R`	**-value -reg**
`-S`	**-value -sym**
`-V`	**-value -var**
`-X`	**-value -expr**
`-target target`	**Operate on specific target.** If missing, current target is used.
`-frame frame`	**Specify reference stack frame.** Search for variable(s) on stack frame context #`frame`. Default is 0 (top frame) for `-var`, otherwise uses non-frame context.
`-jtag`	**Use direct JTAG interface for memory I/O.** This flag forces the command to bypass TCF's memory context infrastructure to do memory I/O. JTAG I/O is done directly using the `jtag` command which will use direct protocol commands from the `::mvproto` namespace. It is useful in certain scenarios where cache coherence is not a problem. Incompatible with `-mem` argument.
`-mem memTarget`	**Target for memory I/O.** Normally the memory operations go through the target context's associated memory context. This argument allows specifying a different context to do I/O through. Incompatible with `-jtag` flag.
`location(3)`	**Memory location - address, register, symbol. variable or expression (after switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` flags is also given.
`count`	**Number of values beginning with address.** Same as `-count` option. Added for convenience. Cannot be used together with `-count`.

3.26. mget/i — Interactive `mget` session

Synopsis

```
mget/i ?options?...
```

Description

Start an interactive `mget` session.

The standard input is read for locations separated by `;` or newline.

Empty input or end-of-file (Ctrl-D) ends the session.

Empty locations are ignored.

`mget` is performed for each location separately, using the base options and additional parameters

Text after C-style `//` comments modify the additional parameters for `mget`.

Text after C-style `///` comments sets the base options and clears any additional parameters.

Basic autocomplete is avaiable in the interactive `mget` shell.

Tcl event dispatch is blocked while user input is being entered!

Arguments

Synopsis	Description
`?options?...`	**Any option available for `mget` except `location()`*** These will be the base options for the interactive `mget` session.

3.27. mset — Set value to memory location.

Synopsis

```
mset ?location(1)? ?-type typeclass? ?-size size? ?-list? ?-dict? ?-binary? ?-target target? ?-frame frame? ?-jtag?
?-mem memTarget? ?location(3)? what
```

Description

Set value of specified address, register, symbol, variable or expression.

Examples

Set a 32-bit word representing the value 10 at address:

```
% mset 0x14400010 10
```

Set PC register value of LRT:

```
% mset -target LRT -reg PC 0x76000000
```

Set the variable named `PC` variable tLRT:

```
% mset -target LRT -var PC  10
```

Set the value of the global symbol named `PC`, as if performed by the LeonRT (updating the L1+L2 cache):

```
% mset -target LRT -sym PC 15
```

Set athe IEEE 32-bit float representation of 4.3 in the I0 register:

```
% mset -reg i0 -type float 4.3
```

Arguments

Synopsis	Description
`location(1)`	**Memory location - address, register, symbol, variable or expression (before switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` options is also given.
`-addr ?location(2)?`	**Treat `location()`** as numeric address.* One and only one of the `location(*)` arguments must be provided.
`-reg ?location(2)?`	**Treat `location()`** as register name.* One and only one of the `location(*)` arguments must be provided.
`-sym ?location(2)?`	**Treat `location()`** as (global) symbol.* One and only one of the `location(*)` arguments must be provided.
`-var ?location(2)?`	**Treat `location()`** as (local) variable.* One and only one of the `location(*)` arguments must be provided.
`-expr ?location(2)?`	**Treat `location()`** as expression.* One and only one of the `location(*)` arguments must be provided.

Synopsis	Description
`-id ?location(2)?`	**Treat `location()` as TCF Symbol ID*** One and only one of the `location(*)` arguments must be provided.
`-type typeclass`	**Force type class for value.** Affects display/returned value (get) or binary format of scanned string value (set). The following type classes are recognized: int - signed integer 1, 2, 4, 8 bytes hex - unsigned hexadecimal integer 1, 2, 4, 8 bytes unsigned - unsigned decimal integer 1, 2, 4, 8 bytes float - floating point 2, 4, 8 bytes pointer - pointer 4, 8 bytes binary - raw binary string, any size asciiz - null terminated ASCII string or string pointer
`-size size`	**Specifiy size of location.** This is usually used in conjuction with the `-type` argument, to override the detected size of memory location. NOTE: not all -type(class) / -size combinations are legal, see `-type`.
`-endian {little|big}`	**Force endiannes of value.** Only little-endian is supported.
`-binary`	**-type binary -size <length of `what`>** Treat `what` as binary data and write all of it at the specified location. Cannot be used in conjuction with `-type` and `-size`
`-list`	**Set array from list.** *!! NOT IMPLEMENTED !!* Takes each list element from supplied input and sets the array's elements individually.
`-dict`	**Set array/structure from list/dictionary.** *!! NOT IMPLEMENTED !!* Takes each dictionary key/value item from supplied input and sets the structure fields individually.
`-target target`	**Operate on specific target.** If missing, current target is used.
`-frame frame`	**Specify reference stack frame.** Search for variable(s) on stack frame context #`frame`. Default is 0 (top frame) for `-var`, otherwise uses non-frame context.

Synopsis	Description
`-jtag`	**Use direct JTAG interface for memory I/O.** This flag forces the command to bypass TCF's memory context infrastructure to do memory I/O. JTAG I/O is done directly using the `jtag` command which will use direct protocol commands from the `::mvproto` namespace. It is useful in certain scenarios where cache coherence is not a problem. Incompatible with `-mem` argument.
`-mem memTarget`	**Target for memory I/O.** Normally the memory operations go through the target context's associated memory context. This argument allows specifying a different context to do I/O through. Incompatible with `-jtag` flag.
`location(3)`	**Memory location - address, register, symbol. variable or expression (after switches).** One and only one of the `location(*)` arguments must be provided. The type will be tried to be detected automatically, unless one of `-addr`, `-reg`, `-sym`, `-var`, `-expr` flags is also given.
`what`	**Input value.** Its form should be compatible with the memory location's type.

3.28. mutex — Hardware mutex operations.

Synopsis

```
mutex {status|lock|unlock|help} ?args?...
```

Description

Control over hardware mutexes.

The following subcommands are supported:

```
SUBCOMMAND                                  DESCRIPTION
```

```
mutex status                                Display the status of all mutexes.
mutex lock <mutexNumber> <core>             Lock a mutex.
mutex unlock <mutexNumber> <core>           Unlock a mutex.
```

Examples

Show the current state of all mutexes:

```
% mutex status
```

Lock the mutex 0 on LOS:

```
% mutex lock 0 LOS
```

Unlock the mutex 0 on LOS:

```
% mutex unlock 0 LOS
```

3.28.1. mutex lock — Lock mutex.

Synopsis

```
mutex lock index ?target?
```

Description

Lock mutex.

Examples

Lock the mutex 0 through Shave 0:

```
% mutex lock 0 S0
```

Arguments

Synopsis	Description
index	**Mutex index.**

3.28.2. mutex status — Get the status of mutex.

Synopsis

```
mutex status ?-quiet? ?index[..index]?
```

Description

Get the status of mutex.

Examples

Show the current state of all mutexes:

```
% mutex status
```

Arguments

Synopsis	Description
`-quiet`	**Don't display; return list of dictionaries.** Normally the subcommand displays a table to the standard output and will leave no result in the Tcl interpreter. This flag inhibits the display and causes the command to return a result usable for further processing by Tcl scripts.
`index[..index]`	**Mutex index or index range.**

3.28.3. mutex unlock — Unlock mutex.

Synopsis

```
mutex unlock index ?target?
```

Description

Unlock mutex.

Examples

Unlock the mutex 0 through Shave 0:

```
% mutex unlock 0 S0
```

Arguments

Synopsis	Description
`index`	**Mutex index.**

3.29. pipe — Debug pipe management.

Synopsis

```
pipe {create|configure|puts|send|peek|recv|flush|delete|status|info|list|reset|interval} ?args?...
```

Description

Manipulate debug pipes.

The pipe structure is a circular buffer in the application like the following:

```
#define CON_Q_ELEMENTS (8) /* or any size */
#define CON_Q_SIZE     (CON_Q_ELEMENTS + 1)
```

```
typedef struct {
    volatile u32  canaryStart;   // Used to detect corruption of queue
    volatile int  in;            // write index
    volatile int  out;           // read index
    volatile int  queueSize;
    volatile u32  canaryEnd;     // Used to detect corruption of queue
    volatile u8   buffer[CON_Q_SIZE];
} tyMvConsoleQueue;
```

```
    tyMvConsoleQueue mvConsoleTxQueue =
    {
        .canaryStart = 0x11223344,
        .in          = 0,
        .out         = 0,
        .queueSize   = CON_Q_SIZE,
        .canaryEnd   = 0xAABBCCDD,
    };
At most two of these queue structures in the target's memory can be associated with a DEBUG PIPE object, one READ
QUEUE and one WRITE QUEUE.
```

Each DEBUG PIPE has it's own `pipeName` which identifies the object.

The queue structures need not be adjacent, their addresses can be associated individually, and also by symbolic name. The DEBUG PIPE object need not be always associated with a queue.

The memory location occupied by these unidirectional queues will be monitored and can be used to achieve bi-directional comunication between the debugger and the target.

3.29.1. pipe configure — Configure pipe.

Synopsis

```
pipe configure pipeName ?-stdout? ?-stderr? ?-readaddr readAddr? ?-readsym readSymName? ?-writeaddr writeAddr? ?-writesym writeSymName? ?-in inFile? ?-out outFile? ?-append? ?-prefix? ?-target target? ?-source sourceChannel? ?-dest destChannel? ?-channel twoWayChannel? ?-tcp tcpPort?
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.
-stdout	Redirect to stdout.
-stderr	Redirect tot stderr.
-readaddr readAddr	Set read queue address.
-readsym readSymName	Set read queue address from symbol name.
-writeaddr writeAddr	Set write queue address.
-writesym writeSymName	Set write queue address from symbol name.
-in inFile	Redirect from input file.
-out outFile	Redirect to output file.
-append	Open output file in append mode.
-prefix	Set prefix for each printed line.
-target target	Use target for memory I/O
-source sourceChannel	Redirect from source Tcl channel.
-dest destChannel	Redirect to dest Tcl channel
-channel twoWayChannel	Redirect to+from channel.
-tcp tcpPort	Open TCP port and redirect bidirectionally.

3.29.2. pipe create — Create pipe.

Synopsis

```
pipe create pipeName ?-stdout? ?-stderr? ?-readaddr readAddr? ?-readsym readSymName? ?-writeaddr writeAddr? ?-writesym writeSymName? ?-in inFile? ?-out outFile? ?-append? ?-prefix? ?-target target? ?-source sourceChannel? ?-dest destChannel? ?-channel twoWayChannel? ?-tcp tcpPort?
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must not already exist
-stdout	Redirect to stdout.
-stderr	Redirect tot stderr.
-readaddr readAddr	Set read queue address.
-readsym readSymName	Set read queue address from symbol name.
-writeaddr writeAddr	Set write queue address.

Synopsis	Description
`-writesym writeSymName`	**Set write queue address from symbol name.**
`-in inFile`	**Redirect from input file.**
`-out outFile`	**Redirect to output file.**
`-append`	**Open output file in append mode.**
`-prefix`	**Set prefix for each printed line.**
`-target target`	**Use `target` for memory I/O**
`-source sourceChannel`	**Redirect from source Tcl channel.**
`-dest destChannel`	**Redirect to dest Tcl channel**
`-channel twoWayChannel`	**Redirect to+from channel.**
`-tcp tcpPort`	**Open TCP port and redirect bidirectionally.**

3.29.3. pipe delete — Delete pipe.

Synopsis

```
pipe delete pipeName
```

Arguments

Synopsis	Description
`pipeName`	**Name of pipe. Must already exist.**

3.29.4. pipe flush — Flush pipe contents

Synopsis

```
pipe flush pipeName
```

Description

Reads data from pipe until there's nothing more to read and transfers it according to the pipe's read configuration.

Notes

- Tcl Event processing occurs during call.
- May never stop if the Myriad application is continuously producing output.
- Control-C can interrupt this in interactive mode.

Arguments

Synopsis	Description
pipeName	**Name of pipe. Must already exist.**

3.29.5. pipe info — Pipe information.

Synopsis

```
pipe info pipeName ?{-stdout|-stderr|-readaddr|-readsym|-writeaddr|-writesym|-in|-out|-append|-prefix|-target|-source|-dest|-channel|-tcp}?...
```

Description

NOT IMPLEMENTED

Arguments

Synopsis	Description
pipeName	**Name of pipe. Must already exist.**

3.29.6. pipe interval — Set global pipe update intervals (ms).

Synopsis

```
pipe interval ?-busy busyInterval? ?-idle idleInterval?
```

3.29.7. pipe list — List all pipes.

Synopsis

```
pipe list
```

3.29.8. pipe peek — Peek available bytes in pipe.

Synopsis

```
pipe peek pipeName ?-scan binFmt? ?-line? ?maxCount?
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.

3.29.9. pipe puts — Put string to pipe.

Synopsis

```
pipe puts pipeName ?-nonewline? string
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.

3.29.10. pipe recv — Receive data from pipe.

Synopsis

```
pipe recv pipeName ?-scan binFmt? ?-line? ?maxCount?
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.

3.29.11. pipe reset — Reset all pipes.

Synopsis

```
pipe reset
```

3.29.12. pipe send — Send binary data to pipe.

Synopsis

```
pipe send pipeName ?-format binFmt? data
```

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.

3.29.13. pipe status — Pipe status.

Synopsis

```
pipe status pipeName
```

Description

NOT IMPLEMENTED

Arguments

Synopsis	Description
pipeName	Name of pipe. Must already exist.

3.30. registers — TCF Registers

Synopsis

```
registers ?{list|get}? ?args?...
```

Description

Type "help registers list" for more help.

3.30.1. registers get — Get value of register

Synopsis

```
registers get ?-target target? ?-frame frame? name
```

Arguments

Synopsis	Description
-target target	Select target.
-frame frame	Select stack frame
name	Name of register

3.30.2. registers list — List TCF registers

Synopsis

```
registers list ?-target target? ?-frame frame? ?-all? ?-bits? ?-quiet? ?-no-read? ?-force-read? ?prefix?
```

Arguments

Synopsis	Description
`-target target`	**Select target.**
`-frame frame`	**Select stack frame**
`-all`	**Show all registers for context.**
`-bits`	**Show bitfields.**
`-quiet`	**Don't display; return list of dictionaries.** Normally the subcommand displays a table to the standard output and will leave no result in the Tcl interpreter. This flag inhibits the display and causes the command to return a result usable for further processing by Tcl scripts.
`-no-read`	**Don't read values**
`-force-read`	**Force read values** Normally the subcommand will not read values which are marked as `ReadOnce`, because reading them would destroy their value. This flag forces read on those values as well.
`prefix`	**Name prefix glob pattern.**

3.31. repeat — Repeat command or script.

Synopsis

```
repeat count command ?args...?
```

Description

Allows repeating a command or script several times.

Usually there's no real need to do this but it can be useful when migrating MoviDebug Classic scrips.

Examples

Do ten line steps, printing the current line after each one:

```
% repeat 10 step into -line
```

Note: Usually it's faster to do this without intermediary feedback:

```
% step into -line -count 10
```

Do the same line steps, but pause a second between them:

```
% repeat 10 {step -line; wait -ms 1000}
```

Display the current PC, function+offset, file and line for a running core, once per second:

```
# Interrupt with Control-C.
# Note: "forever" is always greater than any number so this will work.
```

```
% repeat forever { puts "[target]: [state -pc -sym -file -line]"; wait -ms 1000 }
```

Arguments

Synopsis	Description
`count`	**Number of iterations.**
`command`	**Command or script to repeat.**
`args`	**Arguments to `command`.**

3.32. run — Run application.

Synopsis

```
run ?-wait? ?-async? ?-timeout timeout?
```

Description

Starts execution of the loaded application.

This can be performed either synchronously or asynchronously.

- The synchronous mode — enabled by default — permits waiting for the application to be stopped, using a the `-wait` flag.

A timeout can be provided to give a time limit to the waiting mechanism.

Exceeding the time limit will raise an error.

Notes

- The command is semantically equivalent to the following:

```
% cont [startupcore]
```

- This means that issuing the command twice — without `loadfile` — will NOT run the application again from the beginning.
- In interactive mode the command is also interruptible by the Control-C key combination.

Aliases

```
ALIAS           FULL FORM
runw            run -wait
runandwait      run -wait
```

Examples

Run application (wait until the core starts running):

```
% run
```

Run application, wait until execution stops:

```
% runw
```

Arguments

Synopsis	Description
`-wait`	**Wait for application to stop.** This flag will cause the command to wait until the application stops. The causes for stopping can be: * application termination * breakpoint was encountered on any other core. Incompatible with the `-async` option.
`-async`	**Return immediately.** Suppresses the default wait mechanism. Incompatible with the `-wait` and `-timeout` options.
`-timeout timeout`	**Raise error on timeout (ms).** Causes the command to raise an error if it does not return normally after the specified number of milliseconds. This timepout is also valid when the `-wait` option is not provided. Incompatible with the `-async` option.

3.33. savefile — Save memory to file.

Synopsis

```
savefile ?-append? ?-mem memTarget? ?-hex? ?-chan? fileNameOrChannel address ?size?
```

Description

Saves a memory region to file.

Notes

- To achieve top performance, data is normally saved using the direct JTAG interface. Coherency is not managed.
- In case when cache coherency is an issue, the `-mem` option forces the operation to be executed using the cache-aware memory model of a specific target.

Warning: Accessing large amounts of data through the Leon/Shave TCF memory model is very slow!

Examples

Save 100 bytes from 0x70000000 in myFile.txt:

```
% savefile myFile.txt 0x70000000 100
```

Append 1 KiB of data from 0x70000000 to myFile.txt:

```
% savefile -append myFile.txt 0x70000000 1024
```

Save some uncached data from the DDR (this is actually not so slow):

```
% savefile -mem A uncached_ddr.txt DDR_BASE_ADR [expr 64 * 1024]
```

Save the contents of the DDR as Shave 0 sees it through the Level 1 and Level 2 caches (reeealy slow):

```
% savefile -mem S0 myFile.txt DDR_BASE_ADR [expr 64 * 1024]
```

Save the contents of the myGlobalBuffer symbol to disk (auto-detected size):

```
% savefile myFile.txt myGlobalBuffer
```

Arguments

Synopsis	Description
`-append`	**Writes to the end of the file.**
`-mem memTarget`	**Read data using TCF** Use the TCF memory interface of `memTarget`. Might get really slow for larger datasets.
`-hex`	**Save to HEX file named `fileNameOrChannel`** HEX format is Movidius-specific. *!! NOT IMPLEMENTED !!* Incompatible with other file type switches: `-chan`.
`-chan`	**Treat `fileNameOrChannel` as name of Tcl channel.** Allows saving binary data directly to open Tcl channel. (e.g. Tcl pipe, socket) Incompatible with other file type switches: `-hex`.
`fileNameOrChannel`	**File name or channel.** Specifies a file name or a Tcl channel name. This is a mandatory argument.
`address`	**Address to save file data from.** Address, symbol or register name. This is a mandatory argument.
`size`	**Size for saved data.** If missing, the size of the symbol will be used.

3.34. startupcore — Set/get the startup core for subsequent load/run operations

Synopsis

```
startupcore core
```

Description

Set/get the main core for subsequent load/run operations.

- This will determine the core which will receive the elf file's entry point.
- The `run` command is equivalent to `cont [startupcore]`.

Examples

Set Leon RT to be the main core:

```
% startupcore LRT
```

3.35. state — Processor and thread state.

Synopsis

```
state -a[ll] ?-reg? ?-F? ?-asr? ?-target target? ?-recursive ?parent? ? ?-children ?parent? ? ?-quiet? ?-isrunning? ?-issuspended? ?-isbp? ?-bpids? ?-exited? ?-pc? ?-sym? ?-function? ?-offset? ?-basedir? ?-file? ?-line? ?-column? ?-suspended-only? ?-stack? ?-update? ?-async? ?target?
```

Description

Display and/or return TCF RunControl context states.

Aliases

- ps

Examples

Display state of the current core:

```
% state
```

Display state of the all contexts (cores + threads):

```
% state -all
```

Display state of Leon OS:

```
% state -target LOS
```

```
% state LOS
```

Display state of the current core:

```
% state
```

Display the values of the most important Leon OS registers:

```
% state -target LOS -reg
```

Return the values of the same LOS registers as Tcl dictionary:

```
% state -target LOS -reg -quiet
```

Return the PC value of LOS:

```
% state -target LOS -pc
```

Return 1 when the application exited:

```
% state -exited -target [startupcore]
```

Return 1 when LRT is running, 0 otherwise:

```
% state -isrunning LRT
```

Return current file and line:

```
% state -file -line
```

Return function name+offset at current PC:

```
% state -sym
```

Something very similar to the MoviDebug Classic `rtemsthreads` command:

```
% state -children [target]:RTEMS -stack
```

Arguments

Synopsis	Description
`-a[ll]`	**All targets.**
`-reg`	**Get register values for target.** Target should be LEON or SHAVE core.
`-F`	**Also get float registers.** Target should be LEON core.
`-asr`	**Also get ASR(17, 24..31) registers.** Target should be LEON core.
`-target target`	**Specific target.**

Synopsis	Description
`-recursive ?parent?`	**Target and its children.**
`-children ?parent?`	**Target's immediate children.**
`-quiet`	**Don't print. Return value.**
`-isrunning`	**Return logical true if target is running.**
`-issuspended`	**Return logical true if target is suspended.**
`-isbp`	**Return logical true if target is suspended in breakpoint.**
`-bpids`	**Return list of breakpoint IDs the target is currently suspended at.**
`-exited`	**Return logical true if target has finished execution.**
`-pc`	**Get Program Counter for target.**
`-sym`	**Get symbol name+offset at current PC**
`-function`	**Get symbol name at current PC without offset (current function)**
`-offset`	**Get offset of current PC relative to current symbol name (function)**
`-basedir`	**Get base directory of current file at PC**
`-file`	**Get current file (name relative to base directory)**
`-line`	**Get current line**
`-column`	**Get current column**
`-suspended-only`	**Only suspended targets.**
`-stack`	**Show call stack for suspended target(s)**
`-update`	**Force a core state update on target or target group.** The state update is done synchronously unless the `-async` option is given. This is useful if the automatic core state monitoring is disabled.
`-async`	**Asynchronous update** Requires the `-update` option. Does not wait for the update to be finished. Causes the function to return and print nothing. Because of this it cannot be used together with many options.
`target`	**Target (after switches).** Same as -target argument. Incompatible with `-all` and `-target`. Added for convenience to allow simple use-cases like `ps LOS`

3.36. step — Step-by-step execution.

Synopsis

```
step ?{into|out|return|over|range}? ?args?...
```

Description

Executing the (current) target in a stepwise fashion.

This can be done either synchronously or asynchronously.

Subcommands/aliases

The following modes/aliases are supported:

```
FULL FORM            ALIAS(ES)        DESCRIPTION
```

```
step into -asm       stepasm          Execute a single assembly
                     stepinstr        instruction.
```

```
step into -line      stepline         Execute source code line.
                                      Steps into function.
```

```
step over -asm       stepoverasm      Execute single assembly
                     stepoverinstr    instruction or function
                                      call.
step over -line      stepover         Execute source code line,
                     stepoverline     including function call in
                                      line.
step over -range                      Executes code in range,
                                      including function calls.
step out             stepout          Steps out of current
                                      function.
step range                            Execute until control goes
                                      out of specified range.
```

Examples

Executes an instruction on the Leon OS:

```
% step into -asm -target LOS
```

Alternatively:

```
% target LOS
```

```
% step
```

Execute a source code line on the LeonOS (if line information is available):

```
% step over -line -target LOS
```

Alternatively:

```
% target LOS
```

```
% stepover
```

Make the LOS step out of the current function:

```
% step out -target LOS
```

3.36.1. step into — Step into.

Synopsis

```
step into ?-target target? ?-count count? ?-timeout timeout? ?-async? ?-quiet? ?-asm? ?-line?
```

Description

Stepwise execution of target, entering called function(s).

Arguments

Synopsis	Description
`-target target`	**Operate on specified target.** If missing, current target is used.

Synopsis	Description
`-count count`	**Repeat step `count` times.** If specified, the step is performed in the background `count` number of times. The target is not reported as being stopped until all the steps are performed. If missing, the step is performed only once.
`-timeout timeout`	**Timeout (ms).** Raises an error if the stepping operation does not finish in time. Incompatible with `-async` option.
`-async`	**Return immediately.** Bypasses synchronisation with target state changes. Incompatible with `-timeout` argument.
`-quiet`	**Quiet** Does not try to display source information after completing high level step.
`-asm`	**Instruction step.** This is the default mode of operation. Incompatible with `-line`.
`-line`	**Source line step.** Incompatible with `-asm`.

3.36.2. step out, step return — Step out.

Synopsis

```
step out ?-target target? ?-count count? ?-timeout timeout? ?-async? ?-quiet?
```

```
step return ?-target target? ?-count count? ?-timeout timeout? ?-async? ?-quiet?
```

Description

Resumes execution until control reaches to the point where the current function has been called.

Notes

- For correct behaviour of this function, the debugger will try to employ hardware breakpoints in the target.
- If there are no more hardware breakpoints available it will try to do the stepping using software breakpoints. This might cause unexpected behaviour if the return instruction is in the core's instruction pipeline.

Make sure you have a free hardware breakpoint in your SHAVE before stepping out.

Arguments

Synopsis	Description
`-target target`	**Operate on specified target.** If missing, current target is used.
`-count count`	**Repeat step `count` times.** If specified, the step is performed in the background `count` number of times. The target is not reported as being stopped until all the steps are performed. If missing, the step is performed only once.
`-timeout timeout`	**Timeout (ms).** Raises an error if the stepping operation does not finish in time. Incompatible with `-async` option.
`-async`	**Return immediately.** Bypasses synchronisation with target state changes. Incompatible with `-timeout` argument.
`-quiet`	**Quiet** Does not try to display source information after completing high level step.

3.36.3. step over — Step over.

Synopsis

```
step over ?-target target? ?-count count? ?-timeout timeout? ?-async? ?-quiet? ?-asm? ?-line? ?-range low high?
```

Description

Stepwise execution, not entering function calls.

Arguments

Synopsis	Description
`-target target`	**Operate on specified target.** If missing, current target is used.

Synopsis	Description
`-count count`	**Repeat step `count` times.** If specified, the step is performed in the background `count` number of times. The target is not reported as being stopped until all the steps are performed. If missing, the step is performed only once.
`-timeout timeout`	**Timeout (ms).** Raises an error if the stepping operation does not finish in time. Incompatible with `-async` option.
`-async`	**Return immediately.** Bypasses synchronisation with target state changes. Incompatible with `-timeout` argument.
`-quiet`	**Quiet** Does not try to display source information after completing high level step.
`-asm`	**Step over instruction.** Runs the target to the next instruction. Function calls are considered to be the same instruction. Incompatible with `-line` and `-range`.
`-line`	**Step over source line.** This is the default mode of operation. Runs the target to the next source line. Function calls are considered to be the same line. Incompatible with `-asm` and `-range`.
`-range low high`	**Range step.** Runs the target until control goes out of the specified range. The range is defined from `low` to `high - 1`. A function call within the range is considered to be part of the range. Incompatible with `-asm` and `-line` options.

3.36.4. step range — Step within range.

Synopsis

```
step range ?-target target? ?-count count? ?-timeout timeout? ?-async? ?-quiet? low high
```

Description

Resumes execution of target until control reaches out of the given range, for any reason.

The range starts from the `low` address and ends just before the `high` address.

Arguments

Synopsis	Description
`-target target`	**Operate on specified target.** If missing, current target is used.
`-count count`	**Repeat step `count` times.** If specified, the step is performed in the background `count` number of times. The target is not reported as being stopped until all the steps are performed. If missing, the step is performed only once.
`-timeout timeout`	**Timeout (ms).** Raises an error if the stepping operation does not finish in time. Incompatible with `-async` option.
`-async`	**Return immediately.** Bypasses synchronisation with target state changes. Incompatible with `-timeout` argument.
`-quiet`	**Quiet** Does not try to display source information after completing high level step.
`low`	**Low PC address.**
`high`	**High PC address.**

3.37. sym — Retrieve symbol information.

Synopsis

```
sym {list|locals|addr|at|inrange|size|exists} ?args?...
```

Description

List or query information related to symbols.

Aliases

```
ALIAS       COMMAND
symbol      sym
symaddr     sym addr
locals      sym locals -quiet
```

Examples

List ELF symbols starting with the letter `m`, with details:

```
% sym list -full m*
```

Get the address of `main`:

```
% sym addr main
```

Display a table of local variables (and their values):

```
% sym locals
```

Get the name of the current function + offset:

```
% sym at -addr [state -pc] -quiet -first
```

or:

```
% state -sym
```

3.37.1. sym addr — Get address of symbol.

Synopsis

```
sym addr ?-target target? ?-nocomplain? name
```

Arguments

Synopsis	Description
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.
`-nocomplain`	**Don't throw error when symbol is not found**

3.37.2. sym at — Get symbol(s) at address.

Synopsis

```
sym at ?-quiet? ?-addr address? ?-target target? ?-nocomplain? ?-first? ?address?
```

Arguments

Synopsis	Description
`-quiet`	**Don't print result, just return value.**
`-addr address`	**Address** Specify address. Cannot be used together with last `address` argument
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.
`-nocomplain`	**Don't throw error when symbol is not found**
`-first`	**Only return first matching symbol name.**

3.37.3. sym exists — Return logical true if symbol exists.

Synopsis

```
sym exists ?-target target? name
```

Arguments

Synopsis	Description
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.

3.37.4. sym inrange — List all symbols in address range.

Synopsis

```
sym inrange ?-quiet? ?-target target? low high
```

Arguments

Synopsis	Description
`-quiet`	**Don't print result, just return value.**
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.

3.37.5. sym list — List symbols.

Synopsis

```
sym list ?-quiet? ?-target target? ?-full? ?pattern?
```

Description

List ELF symbols matching optional pattern.

Arguments

Synopsis	Description
`-quiet`	**Don't print result, just return value.**

Synopsis	Description
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.
`-full`	**Full symbol information.**
`pattern`	**Filter pattern.**

3.37.6. sym locals — Get local variables

Synopsis

```
sym locals ?-quiet? ?-target target? ?-frame frame?
```

Arguments

Synopsis	Description
`-quiet`	**Don't print result, just return value.**
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.
`-frame frame`	**Select frame.** Default is top frame (i.e. 0).

3.37.7. sym size — Get size of symbol.

Synopsis

```
sym size ?-target target? ?-nocomplain? name
```

Arguments

Synopsis	Description
`-target target`	**Select target.** Default is current target. Cannot be used with `-context`.
`-context contextId`	**Target context ID.** Same as `-target` but always expects canonical context identifier. Cannot be used with `-target`.
`-nocomplain`	**Don't throw error when symbol is not found**

3.38. target — Get or set current target

Synopsis

```
target ?core?
```

Description

The current target determines the default context for most of the debugger commands.

Its canonical name (prefixed with `P<processor#>:{A|S|F|V|?}`) is displayed in the command prompt.

The command recognizes the bare unprefixed core identifier and the canonical name.

When called without arguments, it will return the canonical current target name.

Aliases

```
ALIAS       FULL FORM
t           target
```

Examples

Set the current target to the Leon OS:

```
% target LOS
```

Dump 1K of uncached DDR memory:

```
% target A
```

```
% mdump DDR_BASE_ADR [expr 1024/4]
```

Arguments

Synopsis	Description
?core?	**target (core) identifier** e.g LOS, LRT, S0, S1

3.39. targetid — Get canonical TCF context ID of (current) target

Synopsis

```
targetid ?core?
```

Description

The canonical target name is the always prefixed with P<processor#>:{A|S|F|V|?})

The command recognizes the bare unprefixed core identifier and the canonical name.

When called without arguments, it will return the current target's canonical name.

Note

Contrary to the target command, this will not change the current target.

Examples

Get the canonical name of the root context

```
% targetid A
```

Get the canonical name of the Leon RT

```
% targetid LOS
```

Arguments

Synopsis	Description
?core?	**target (core) or group identifier** e.g A, LOS, LRT, S0, S1, LALL, LOS:RTEMS, etc

3.40. uart — UART management.

Synopsis

```
uart {on|off|handler|tcp|status|flush|silent|interval|prefix} ?args?...
```

Description

Control over UART communication.

Notes

- polling is ON by default.
- polling is implemented entirely in Tcl
- polling is done differently for moviDebugServer and moviSim
- register setup happens during moviDebug2 startup and during board reset IFF the polling is configured ON beforehand.
- If your application does not use UART for printing, you will need to disable the UART register setup.
    1. from the moviDebug2 command line: using the `--no-uart` option.
    2. issuing the `uart off` and `breset` commands in the CLI

Examples

Enable UART polling:

```
% uart on
```

Disable UART polling:

```
% uart off
```

Get UART polling status:

```
% uart status
```

Flush collected UART data (adding a newline forcefully):

```
% uart flush
```

Listen on port 12345 and send UART output to clients:

```
% uart tcp -port 12345
```

Same as before, but also wait for the first client to connect before continuing. Raise an error if noone

connects in 60 seconds:

```
% uart tcp -port 12345 -wait -timeout [expr 60 * 1000]
```

*Type "help uart *" for subcommand list.*

Type "help uart" + subcommand for more details.

3.40.1. uart flush — Flush UART data.

Synopsis

```
uart flush
```

3.40.2. uart handler — Add/remove UART handler.

Synopsis

```
uart handler {add|cancel|reset} ?args?...
```

Description

Set or cancel UART data handler callback scripts.

NOTE	Handler callbacks will receive terminating newline as well.

Examples

Process UART output using custom command:

```
% proc CustomPrint {prefix line} { puts -nonewline stderr "$prefix $line" }
```

```
% uart handler add CustomPrint "myUART:"
```

Process UART output using `apply` and lambda expression:

```
% uart handler add apply {line { puts "<< [string range $line 0 end-1] >>" } ::}
```

Append UART output lines (without last newline) to the global list `UartOutput`:

```
% uart handler add { lappend ::UartOutput [string range %0 0 end-1] }
```

Append UART output lines to the global string `UartOutput`:

```
% uart handler add append UartOutput
```

Save UART output to some file:

```
% set FileHandle [open "UartOutput.txt" w]
```

```
% uart handler add puts -nonewline $FileHandle
```

Remove added handler:

```
% set handlerID [uart handler add MyCustomHandlerProc]
```

```
% # ...
```

```
% uart handler cancel $handlerID
```

Remove all added handlers:

```
% uart handler reset
```

Get the quoted UART prefix:

```
% format {"%s"} [uart prefix]
```

3.40.2.1. uart handler add — Add handler command.

Synopsis

```
uart handler add command ?args?...
```

Description

Returns ID.

Type "help uart handler" for examples

Arguments

Synopsis	Description
`command`	**Handler command.**
`args...`	**Extra arguments to command before data argument.**

3.40.2.2. uart handler cancel — Cancel handler.

Synopsis

```
uart handler cancel id
```

Description

Type "help uart handler" for examples.

Arguments

Synopsis	Description
`id`	**Handler ID.**

3.40.2.3. uart handler reset — Clear all UART handlers.

Synopsis

```
uart handler reset
```

3.40.3. uart interval — Control uart polling interval

Synopsis

```
uart interval ?-idle idle? ?-busy busy?
```

3.40.4. uart off — Disable UART.

Synopsis

```
uart off
```

3.40.5. uart on — Enable UART.

Synopsis

```
uart on
```

3.40.6. uart prefix — Get or set the UART message prefix

Synopsis

```
uart prefix ?prefix?
```

Description

By default the UART messages are prefixed by the string `_UART: _`.

This subcommand allows changing that prefix to an arbitrary string.

When called without arguments, it will return the current value of the prefix.

Notes

- No separators are added to the prefix string so they must be added to the argument string when needed.
- Enclose arguments containing spaces in braces or double quotes.
- Alternatively spaces can be escaped (using the backslash character).

3.40.7. uart silent — Control display of UART messages

Synopsis

```
uart silent ?state?
```

Arguments

Synopsis	Description
`state`	**Message suppression state.** If missing, current state will be returned.

3.40.8. uart status — Get UART status (on/off).

Synopsis

```
uart status
```

3.40.9. uart tcp — Provide UART output on TCP.

Synopsis

```
uart tcp ?-port port? ?-off? ?-wait? ?-timeout timeout?
```

Arguments

Synopsis	Description
`-port port`	**Specify port.**
`-off`	**Turn off server.**
`-wait`	**Wait for first client.**
`-timeout timeout`	**Wait timeout (ms).**

3.41. wait — Wait a time period or event(s).

Synopsis

```
wait ?-ms ms? ?-target target? ?-service TcfService? ?-timeout timeout? ?-suspended? ?-resumed? ?-running? ?-not-
running? ?-anybp? ?-event TcfEvent?... ?-filter-script filterBody? ?-filter-args filterArgs? ?-filter-params
filterParams? ?-after triggerScript? ?targetOrMs?
```

Description

Allows waiting a specified number of milliseconds or certain events.

Waiting for events can be time-limited.

Tcl events are processed during the wait.

Interruptible using the Control+C key combination in interactive mode.

Examples

Wait for 3000 ms:

```
% wait -ms 3000
```

Wait for 3000 ms for the Shave0 to suspend. Raise error on timeout:

```
% wait -target S0 -suspended -timeout 3000
```

Wait for 3000 ms for the Leon RT to suspend. Also stop if a breakpoint is hit on another core. Raise error on timeout:

```
% wait -target LRT -suspended -anybp -timeout 3000
```

Run application and wait until the LRT starts:

```
% wait -target LRT -resumed -after {run}
```

Arguments

Synopsis	Description
`-ms ms`	**Milliseconds to wait.** If provided, causes the command to wait unconditionally, until the specified number of milliseconds pass.
`-target target`	**Target to wait for.** Specifiy the target of which state is conditioning the wait. If missing, the current target will be considered. This is normally used in conjuction one of the `-suspended`, `resumed`, `-running` or `-not-running` options.
`-service TcfService`	**TCF Service.** This is used for advanced use-cases. The specified TCF Service's events will be monitored. Used together with `-event`, `-filter-\\...`
`-timeout timeout`	**Raise error on timeout (ms).** Causes the command to raise an error if specified time elapses without normal return.
`-suspended`	**Wait on target suspend.** This is the default behaviour when waiting on target. Return condition is that the target is in Suspended state.
`-resumed`	**Wait on target resumed.** Return condition is that the target resumes execution from the Suspended or state. It can also mean that the target loses its clock or power.

Synopsis	Description
`-running`	**Wait on target is running.** Return condition is that the target is running (i.e. it has clock, power, and it's not Suspended)
`-not-running`	**Wait on target not running.** Return condition is that the target is **not** running.
`-anybp`	**Wait on any breakpoint hit.** Requires `-suspended` switch to be present. Triggers return also on any other breakpoint being hit.
`-event TcfEvent...`	**TCF Event.** Requires the `-service` argument. Restricts the monitored events of the aforementioned service. Can be given multiple times to specify multiple events. The default is to monitor all events of the service (*).
`-filter-script filterBody`	**TCF event filter script.** Specifies a script to evaluate when one of the monitored events occurs. Should return logical true to signal that the wait condition has been satisfied. (i.e. the wait is over and the command can return) The script body will receive arguments consisting of: * values specified by `-filter-args` argument. * event name * event arguments (depend on actual event) Default implementation is to return true, i.e. the first event will satisfy the wait condition.
`-filter-args filterArgs`	**Value(s) transmitted as first parameter(s) of filter script.** `filterArgs` is a list of values. Each one of the items will become a separate argument of the filter script.
`-filter-params filterParams`	**Formal parameters of filter script.** The order is the following: * argument names for each value in the `-filter-args` list * parameter receiving TCF event name * event parameters of the TCF event. The default is `args` which takes in all the actual parameters as a list.

Synopsis	Description
`-after triggerScript`	**Run trigger script.** Allows specifing a script which will be executed **after** the event monitoring has been set up, on the caller's stack frame. This script is usually asynchronous, and it is expected to cause the wait condition to be satisfied during execution or at a later time.
`targetOrMs`	**Target or milliseconds.** This is a combination of the `-ms` and `-target` argument. It is for convenience to allow syntax such as: `wait 1000` - wait a second `wait LOS` - wait for LeonOS to suspend. It cannot be provided if one of `-ms` and `-target` is already present. It is also incompatible with the advanced options `-service`, `-event`, `-filter-script`, `-filter-args`, `-filter-params`.

3.42. vcshooks::log — Configure VCS Hooks Logging

Synopsis

```
vcshooks::log {on|off} ?fileName? ?fileOpenMode?
```

Description

Configure the display and logging ouptut of VCS Hooks functionality.

Examples

Disable VCS Hooks display:

```
% vcshooks::log off
```

Enable VCS Hooks display on screen only:

```
% vcshooks::log on
```

Enable VCS Hooks display on screen and into newly created file:

```
% vcshooks::log on "vcshooks.log"
```

Enable VCS hooks display on screen and append into file:

```
% vcshooks::log on "vcshooks.log" "a"
```