QUICK REFERENCE

CONVEX CXdb Quick Reference

Order No. DSW-474

Second Edition
November 1992

CONVEX COMPUTER CORPORATION

CONVEX CXdb Quick Reference

Second Edition

Order No. DSW-474

Printed in the United States of America.

Contents

Commands 1

csd aliases 26

gdb aliases 29

Maryland Windows keyboard functions . . 32

Text scrolling functions 32

Window movement functions 32

Window positioning functions 33

Command window functions 33

Mouse functions 34

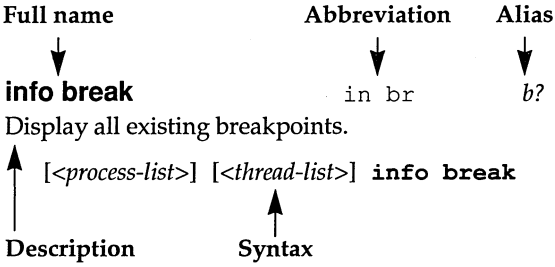
606

Commands

This section lists the following information for each CXdb command:

- Full command name
- Shortest abbreviation for the command
- Default alias for the command
- Brief description of what the command does
- Syntax of the command

The information is presented in the following format:



A

add cmderr

`ad cmde`

Add viewports to cmderr.

add cmderr *<viewport>* [, ...]

add cmdlog

`ad cmdl`

Add viewports to cmdlog.

add cmdlog *<viewport>* [, ...]

add cmdout

`ad cmdo`

Add viewports to cmdout.

add cmdout *<viewport>* [, ...]

add default environment

`ad d e` *denv+*

Add or modify environment variables in the default environment.

add default environment
<environment-variable> = *<string>* [, ...]

add default path

`ad d p` *dp+*

Add directories to the default search path.

add default path *<directory-specifier>* [, ...]

add environment

`ad e` *env+*

Add or modify environment variables in the process environment.

[*<process-list>*] **add environment** *<environment-variable>* = *<string>* [, ...]

add path

`ad p` *p+*

Add directories to the search path.

[*<process-list>*] **add path** *<directory-specifier>* [, ...]

alias

`al`

Define an alias.

alias *<alias-name>* *<string>*

attach

`at`

Debug the image of a running process.

[*<process-list>*] **attach** [*<remote-host>*:] *<process-id>*

B

backtrace

`ba` *bt*

Display the frames of the call stack.

[*<process-list>*] [*<thread-list>*] **backtrace** [*<frame-count>*]

bind

`bin`

Define key bindings for Maryland Windows.

bind *<function-name>* *<key-name>*

break instruction

`bre i` *bi*

Set a breakpoint at an instruction.

[*<process-list>*] [*<thread-list>*] **break instruction** *<language-expression>* [{*<event-handler>*}] [*<debugger-variable>*]

break line

`bre l` *bl*

Set a breakpoint at a source line.

[*<process-list>*] [*<thread-list>*] **`break line`**
<line-specifier> [**{***<event-handler>***}**]
[*<debugger-variable>*]

break routine

`bre r` *br*

Set a breakpoint at the beginning of a routine.

[*<process-list>*] [*<thread-list>*] **`break routine`**
<language-expression> [**{***<event-handler>***}**]
[*<debugger-variable>*]

break source

`bre s` *bs*

Set a breakpoint at a source unit.

[*<process-list>*] [*<thread-list>*] **`break source`**
<source-unit> [**{***<event-handler>***}**] [*<debugger-variable>*]

C

cd

`cd`

Change the console working directory.

`cd` *<directory-specifier>*

clear autocreate

`cl a`

Disable the dynamic creation of source windows.

`clear autocreate`

clear default environment

`cl d e`

Remove all environment variables from the default environment.

`clear default environment`

clear default fixed sched

`cl d f s`

Disable fixed scheduling in the default settings.

`clear default fixed sched`

clear default handler

`cl d h`

Clear the default handler for all eventpoints.

`clear default handler`

clear default remotewd `cl de re`

Clear the default remote working directory.

clear default remotewd

clear echo `cl ec`

Disable echoing of input from initialization files and command files.

clear echo

clear environment `cl en`

Remove all environment variables from the process environment.

[*<process-list>*] **clear environment**

clear fixed sched `cl f s` *cfs*

Disable fixed scheduling in the process settings.

[*<process-list>*] **clear fixed sched**

clear handler `cl h`

Clear the handler for a specified eventpoint.

clear handler *<event-specifier>* [, ...]

clear logging `cl l`

Disable logging for cmdlog.

clear logging

clear noclobber `cl n`

Disable the noclobber option for all viewports.

clear noclobber

clear seq `cl se`

Clear the sequential mode (SEQ) bit.

[*<process-list>*] [*<thread-list>*] **clear seq**

clear sqs `cl sq`

Clear the sequential store enable (SQS) bit.

[*<process-list>*] [*<thread-list>*] **clear sqs**

clear step `cl st`

Reset the stepping granularity to the default setting.

[*<process-list>*] [*<thread-list>*] **clear step**

clear typehandler

`cl t`

Clear the handler for a specified type of eventpoint.

`clear typehandler` *<eventtype-specifier>* [**`,`** ...]

continue

`con` *c*

Continue execution of the process.

[*<process-list>*] [*<thread-list>*] **`continue`** [*<count>*] [**`&`**]

copy

`cop`

Copy a memory region.

[*<process-list>*] [*<thread-list>*] **`copy`** *<source-address>*
[{ **`..`***<ending-address>* | **`:`***<byte-count>* }]
`\;` *<destination-address>*

core

`cor`

Debug the image of a core file or a checkpoint file.

[*<process-list>*] **`core`** [*<remote-host>***`:`**] *<file-name>*

csd

`csd`

Enable compatibility with `csd` debugger commands.

`csd`

cxdb

`cxdb`

Invoke CXdb from the shell.

`cxdb` [**`-a`** *<process-id>*] [**`-csd`**]
[**`-D`** *<directory-specifier>*[**`,`** ...]] [**`-f`** *<file-name>*] [**`-F`**]
[**`-l`**] [**`-nw`**] [**`-nx`**] [**`-sw`** *<geometry>*] [**`-cw`** *<geometry>*]
[**`-hw`** *<geometry>*] [**`-pw`** *<geometry>*]
[**`-fw`** *<geometry>*] [**`-x`** *<command-list>*]
[[**`-e`**] *<file-name>*] [[**`-c`**] *<file-name>*]
[*<x-toolkit-options>*]

D

debug core

`deb c` *dbgc*

Create a process object and debug the image of a core or checkpoint file.

`debug core` *<core-file>* [[*<remote-host>***`:`**]
<executable-file>] [*<debugger-variable>*]

debug exec

`deb e` *dbg*

Create a process object and debug the image of an executable file.

debug exec [<*remote-host*>**:**] <*executable-file*> [<*debugger-variable*>]

debug proc

`deb p` *dbgp*

Create a process object and debug the image from a running process.

debug proc [<*remote-host*>**:**] <*process-id*> [[<*remote-host*>**:**] <*executable-file*>] [<*debugger-variable*>]

detach

`det`

Detach from a process.

[<*process-list*>] **detach**

dirpath

`dir`

Specify an alternate directory path for the CDI data files.

dirpath <*original-directory*> <*alternate-directory*>

disable event

`disab event` *dis*

Disable eventpoints.

disable event <*event-specifier*> [**,** ...]

disable eventtype

`disab eventt`

Disable all eventpoints of the specified type.

[<*process-list*>] **disable eventtype** <*eventtype-specifier*> [**,** ...]

disassemble

`disas`

Display the disassembled code.

[<*process-list*>] [<*thread-list*>] **disassemble** [<*starting-address*> [{ **. .**<*ending-address*> | **:**<*instruction-count*> }]]

display disassembly

`disp d`

Create a disassembly window in CXwindows.

[<*process-list*>] [<*thread-list*>] **display disassembly** [<*address-expression*>] [**\;** <*thread-number*> [**,** ...]]

display examine

`disp e`

Create an examine window in CXwindows.

[<*process-list*>] [<*thread-list*>] **`display examine`**
[<*address-expression*>] [**`\;`** <*thread-number*>]

display file

`disp f`

Display the contents of a file.

[<*process-list*>] **`display file`** <*file-name*>

display routine

`disp r`

Display the source code for a routine.

[<*process-list*>] [<*thread-list*>] **`display routine`**
<*language-expression*> [**`\;`** <*thread-number*> [**`,`** ...]]

display source

`disp so`

Create a new source window to display a source file.

[<*process-list*>] **`display source`** <*file-name*>
[<*thread-number*> [**`,`** ...]]

display stack

`disp st`

Create a stack window in CXwindows.

[<*process-list*>] [<*thread-list*>] **`display stack`**
[<*frame-specifier*>] [**`:`**<*thread-number*>]

E

echo

`ec`

Echo a character string.

`echo[/n]` <*string*> [...]

edit

`ed`

Edit a file.

`edit` [<*file-name*>]

enable event

`ena event` *en*

Enable eventpoints.

`enable event` <*event-specifier*> [**`,`** ...]

enable eventtype `ena eventt`

Enable all eventpoints of the specified type.

[*<process-list>*] **enable eventtype**
<eventtype-specifier> [, ...]

evaluate `eva`

Evaluate a language expression.

[*<process-list>*] [*<thread-list>*]
evaluate *<language-expression>*

event exec `eve e`

Set an eventpoint to watch for an `exec(2)` system call by the process.

[*<process-list>*] **event exec** [{*<event-handler>*}]
[*<debugger-variable>*]

event join `eve j`

Set an eventpoint to trap a thread joining.

[*<process-list>*] **event join** [{*<event-handler>*}]
[*<debugger-variable>*]

event modify `eve m`

Set an eventpoint to watch for a value change within an address range.

[*<process-list>*] [*<thread-list>*]
event modify *<starting-address>*
[{ . .*<ending address>* | :*<byte-count>* }]
[{*<event-handler>*}] [*<debugger-variable>*]

event reached instruction `eve rea i`

Set an eventpoint at an instruction.

[*<process-list>*] [*<thread-list>*]
event reached instruction
<language-expression> [{*<event-handler>*}]
[*<debugger-variable>*]

event reached line `eve rea l`

Set an eventpoint at a source line.

[*<process-list>*] [*<thread-list>*]
event reached line *<line-specifier>*
[{*<event-handler>*}] [*<debugger-variable>*]

event reached routine `eve rea r`

Set an eventpoint at the beginning of a routine.

[*<process-list>*] [*<thread-list>*]
`event reached routine` *<language-expression>*
[**{** *<event-handler>* **}**] [*<debugger-variable>*]

event reached source `eve rea s`

Set an eventpoint at a source unit.

[*<process-list>*] [*<thread-list>*]
`event reached source` *<source-unit>*
[**{** *<event-handler>* **}**] [*<debugger-variable>*]

event relation `eve rel`

Set an eventpoint to watch for an expression to become true.

[*<process-list>*] [*<thread-list>*]
`event relation` *<language-expression>*
[**{** *<event-handler>* **}**] [*<debugger-variable>*]

event signal `eve si`

Set an eventpoint to catch a signal.

[*<process-list>*] **`event signal`** *<signal-specifier>*
[**{** *<event-handler>* **}**] [*<debugger-variable>*]

event spawn `eve sp`

Set an eventpoint to trap the spawning of a thread.

[*<process-list>*] **`event spawn`** [**{** *<event-handler>* **}**]
[*<debugger-variable>*]

examine `exa` *x*

Display a region of memory.

[*<process-list>*] [*<thread-list>*] **`examine`**
[**/**{*<memory-unit>* *<format>* *<fpmode>*}]
<starting-address>
[{ **. .** *<ending-address>* | **:** *<unit-count>*}]

executable `exe`

Specify an executable file for a CDI base and the executable image.

[*<process-list>*] **`executable`** [*<remote-host>*`:`]
<file-name>

F

fill

`fil`

Fill a region of memory with the value of an expression.

[*<process-list>*] [*<thread-list>*] **`fill`** [**`/`***<memory-unit>*] *<starting-address>* [{ **`..`***<ending-address>* | **`:`***<unit-count>* }] **`\;`** *<language-expression>*

find memory backward

`find m b` *fmb*

Find a byte pattern within a memory region.

[*<process-list>*] [*<thread-list>*]
`find memory backward` [**`/`***<memory-unit>*]
<byte-pattern> *<lowest-address>*
{**`..`***<highest-address>* | **`:`***<byte-count>*}

find memory forward

`find m f` *fmf*

Find a byte pattern within a memory region.

[*<process-list>*] [*<thread-list>*] **`find memory forward`**
[**`/`***<memory-unit>*] *<byte-pattern>* *<starting-address>*
{**`..`***<ending-address>* | **`:`***<byte-count>*}

find window backward

`find w b` *fwb*

Find a character string in a source window.

`find window backward` *<string>*
[{ *<window-number>* | *<debugger-variable>* }]

find window forward

`find w f` *fwf*

Find a character string in a source window.

`find window forward` *<string>*
[{ *<window-number>* | *<debugger-variable>* }]

finish

`fini`

Finish executing (step out of) the specified source unit.

[*<process-list>*] [*<thread-list>*] **`finish`**
[*<granularity>*] [**`&`**]

frame

`fr` *f*

Change the current stack frame.

[*<process-list>*] [*<thread-list>*] **`frame`** *<frame-specifier>*

G

gdb

`gdb`

Enable compatibility with `gdb` debugger commands.

gdb

get

`ge`

Restore the contents of memory regions from a file.

[*<process-list>*] [*<thread-list>*] **get** *<file-name>*
[*<starting-address>*
[{ . . *<ending-address>* | : *<byte-count>*}]] [\; ...]

goto address

`g a`

Branch to the specified address.

[*<process-list>*] [*<thread-list>*] **goto address**
<language-expression>

goto line

`g l`

Branch to the specified source line.

[*<process-list>*] [*<thread-list>*] **goto line**
<line-specifier>

goto source

`g s`

Branch to the specified source unit.

[*<process-list>*] [*<thread-list>*] **goto source**
<source-unit>

H

help

`h` `?`

Invoke the CXdb help system.

help [*<string>*]

I

if

`if`

Establish conditional execution of CXdb commands.

if (*<relational-expression>*) *<command-set>*
[**else** *<command-set>*]

info alias

`in al` *i al*

Display aliases.

info alias [*<regular-expression>*]

info args

`in ar` *args*

Display arguments of the current routine.

[*<process-list>*] [*<thread-list>*] **info args**

info bind

`in bi` *i bi*

Display key bindings for Maryland Windows.

info bind [*<key-name>*]

info break

`in br` *b?*

Display all existing breakpoints.

[*<process-list>*] [*<thread-list>*] **info break**

info cregisters

`in cr` *i cr*

Display the communication registers.

[*<process-list>*] **info cregisters**

info cxdb

`in cx` *i cx*

Display the status of the current CXdb session.

info cxdb

info default environment

`in d e` *i d e*

Display all default environment variables.

info default environment [*<regular-expression>*]

info dirpath

`i di`

List alternate CDI directory paths or the names of object files that match a regular expression.

info dirpath [*<regular-expression>*]

info dynamicobject

`in dy` *i dy*

Display memory segments of dynamically loaded objects.

[*<process-list>*] **info dynamicobject**

info environment

`in en` *env?*

Display all process environment variables.

[*<process-list>*] **info environment**
[*<regular-expression>*]

info errno

`in er` *i er*

Display the error message received by the process.

[<*process-list*>] [<*thread-list*>] **`info errno`**

info event

`in event` *e?*

Display the specified eventpoints.

`info event` [<*event-specifier*>] [**`,`** ...]

info eventtype

`in eventt` *et?*

Display all eventpoints of the specified type.

[<*process-list*>] **`info eventtype`** <*eventtype-specifier*> [**`,`** ...]

info expression

`in ex` *describe, whatis*

Display the characteristics of the specified language expression.

[<*process-list*>] [<*thread-list*>] **`info expression`** <*language-expression*>

info formatting

`in fo` *i fo*

Display the settings for memory display formats.

[<*process-list*>] **`info formatting`**

info frame

`in fr` *i fr*

Display a stack frame.

[<*process-list*>] [<*thread-list*>] **`info frame`** [<*frame-specifier*>]

info frame at

`in fr a` *i fr a*

Display the stack frame at the specified address.

[<*process-list*>] [<*thread-list*>] **`info frame at`** <*language-expression*>

info history

`in h` *i h*

Display the CXdb command history.

`info history` [<*command-count*>]

info line

`in li` *i li*

Display the source units on a specified line.

[<*process-list*>] **`info line`** <*line-specifier*>

info locals `in lo` *locals*

Display the local variables of the current routine.

[<*process-list*>] [<*thread-list*>] **info locals**

info macro `in m` *i m*

Display macros.

info macro [<*regular-expression*>]

info objectmap `in o` *i o*

Display the object map.

[<*process-list*>] **info objectmap**

info path `in pa` *p+*

Display the directories in the search path.

[<*process-list*>] **info path**

info process `in pr` *p?*

Display the status of the process.

[<*process-list*>] **info process**

info psw `in ps` *i ps*

Display the processor status word.

[<*process-list*>] [<*thread-list*>] **info psw**

info registers `in r` *i r*

Display the scalar and address registers.

[<*process-list*>] [<*thread-list*>] **info registers**

info scope `in sc` *where*

Display the current scope path.

[<*process-list*>] [<*thread-list*>] **info scope**

info signal `in si` *i si*

Display the settings of the signal actions.

[<*process-list*>] **info signal** [<*signal-specifier*>]
[, ...]

info sourceunit `in so` *i so*

Display the specified source unit.

[<*process-list*>] [<*thread-list*>]
info sourceunit <*source-unit*>

info stack

`in st` *i st*

Display information about the process stack.

[<*process-list*>] [<*thread-list*>] **`info stack`**

info symbols

`in sy` *globals, symbols*

Display program symbols.

[<*process-list*>] **`info symbols`** [<*regular-expression*>]

info threads

`in th` *i th*

Display threads or tasks of the current process.

`[`<*process-list*>**`]`** **`[`**<*thread-list*>**`]`** **`info threads`**

info trace

`in tr` *t?*

Display all tracepoints.

[<*process-list*>] [<*thread-list*>] **`info trace`**

info type

`in ty` *i ty*

Display type definitions.

[<*process-list*>] [<*thread*>] **`info type`**
[<*regular-expression*>]

info vregisters

`in v` *i vr*

Display the vector registers.

[<*process-list*>] [<*thread-list*>] **`info vregisters`**

info watch

`in w` *i w*

Display all watchpoints.

[<*process-list*>] [<*thread-list*>] **`info watch`**

K

kill process

`k p` *k*

Terminate a running process and remove the image being debugged from the process object.

[<*process-list*>] **`kill process`**

L

list `l`

List lines of source code to cmdout.

`list` [[*<file-name>*`:`] *<starting-line>*] [*<number-of-lines>*]

load object `l o`

Load CDI data for an object file that has been dynamically loaded.

[*<process-list>*] **`load object`** *<file-name>* *<text-address>* *<data-address>* *<tdata-address>* *<bss-address>* *<tbss-address>*

M

macro `m`

Define a macro.

`macro` *<name>* [**(***<parameter>*[**`:`***<default>*] [**`, ...`**]**)**] *<string>*

N

next `n`

Step to the next source unit, ignoring subroutine calls.

[*<process-list>*] [*<thread-list>*] **`next`** [*<granularity>*] [*<count>*] [**`&`**]

next instruction `n i` *ni, nexti*

Step to the next instruction, ignoring subroutine calls.

[*<process-list>*] [*<thread-list>*] **`next instruction`** [*<count>*] [**`&`**]

next over `n o` *no*

Step from the current source unit of specified granularity to the next source unit of default granularity, ignoring subroutine calls.

[*<process-list>*] [*<thread-list>*] **`next over`** [*<granularity>*] [**`&`**] [*<count>*]

P

print

`pr` *p*

Evaluate a language expression and print the result.

[*<process-list>*] [*<thread-list>*]
`print``[/[n]`*<format>*`<fpmode>]` *<language-expression>*

put

`pu`

Save the contents of memory to a file for later retrieval.

[*<process-list>*] [*<thread-list>*] **`put`** *<file-name>*
<starting-address> [{`..`*<ending-address>* | `:`*<byte-count>*}] [`\; ...`]

pwd

`pw`

Display the name of the console working directory.

`pwd`

Q

quit

`q`

Exit from CXdb.

`quit`

R

recall

`rec` *!*

Re-execute a previous command.

`recall` [`?`]*<string>*

remove alias

`rem a`

Delete an alias.

`remove alias` *<alias-name>*

remove cmderr

`rem cmde`

Delete viewports from cmderr.

`remove cmderr` *<viewport>* [`, ...`]

remove cmdlog

`rem cmdl`

Delete viewports from cmdlog.

`remove cmdlog` *<viewport>* [`, ...`]

remove cmdout

`rem cmdo`

Delete viewports from cmdout.

remove cmdout *<viewport>* [, ...]

remove default environment

`rem d e` *denv-*

Delete environment variables from the default environment.

remove default environment *<environment-variable>* [, ...]

remove default path

`rem d p` *dp-*

Delete directories from the default search path.

remove default path *<directory-specifier>* [, ...]

remove dirpath

`rem di`

Remove CDI directory paths created with the `dirpath` command.

remove dirpath [*<original-directory>*]

remove environment

`rem en` *env-*

Delete environment variables from the process environment.

[*<process-list>*] **remove environment** *<environment-variable>* [, ...]

remove event

`rem event` *e-*

Delete the specified eventpoints.

remove event *<event-specifier>* [, ...]

remove eventtype

`rem eventt` *et-*

Delete all eventpoints of the specified type.

[*<process-list>*] **remove eventtype** *<eventtype-specifier>* [, ...]

remove macro

`rem m`

Delete a macro.

remove macro *<name>*

remove path

`rem p` *p-*

Delete directories from the process search path.

[*<process-list>*] **remove path** *<directory-specifier>* [, ...]

remove variable

`rem v`

Delete a debugger variable.

`remove variable` *<debugger-variable>*

rerun

`rer` *rr*

Create and execute a new process, using the previous argument list.

[*<process-list>*] **`rerun`** **`[&]`**

resume

`res`

Continue execution of the process from within an eventpoint handler.

`resume`

return

`ret`

Return to the calling routine.

[*<process-list>*] [*<thread-list>*]
`return` *<language-expression>*

run

`ru` *r*

Create and execute a new process.

[*<process-list>*] **`run`** [*<argument-list>*] **`[&]`**

S

set autocreate

`se a`

Enable dynamic creation of source windows.

`set autocreate`

set cmderr

`se cmde`

Clear and redefine the viewport list for cmderr.

`set cmderr` *<viewport>* [, ...]

set cmdlog

`se cmdl`

Clear and redefine the viewport list for cmdlog.

`set cmdlog` *<viewport>* [, ...]

set cmdout

`se cmdo`

Clear and redefine the viewport list for cmdout.

`set cmdout` *<viewport>* [, ...]

set default environment

`se de e` *denv=*

Clear and redefine the environment variables for the default environment.

set default environment
<environment-variable> = *<string>* [, ...]

set default fixed sched

`se de fi s`

Enable fixed scheduling in the default settings.

set default fixed sched

set default format

`se de fo`

Set the default formats for displaying memory.

set default format *<memory-unit>* *<format>*

set default fpmode

`se de fp`

Set the default floating point mode for new processes.

set default fpmode {**ieee** | **native** | **dual**}

set default handler

`se de h`

Set the default handler for eventpoints.

set default handler {*<event-handler>*}

set default memory

`se de m`

Set the default unit size for displaying memory.

set default memory *<memory-unit>*

set default path

`se de pa` *dp=*

Set the default search path.

set default path *<directory-specifier>* [, ...]

set default pshell

`se de ps`

Set the default process shell.

set default pshell {**sh** | **csh**}

set default remotewd

`se de re`

Set the default remote working directory.

set default remotewd *<directory-specifier>*

set default step

`se de s`

Set the default stepping granularity.

set default step *<granularity>*

set directory

`se di`

Set the process working directory.

[*<process-list>*] **`set directory`** *<directory-specifier>*

set echo

`se ec`

Enable echoing of input from command files.

`set echo`

set environment

`se en` *env=*

Clear and redefine the environment variables for the process environment.

[*<process-list>*] **`set environment`**
<environment-variable>=*<string>* **[, ...]**

set evalopts fpmode

`se ev f`

Set the floating point mode for evaluating expressions.

`set evalopts fpmode` {**`ieee`** | **`native`** | **`dual`**}

set evalopts iprecision

`se ev i`

Set the size of integer constants for CXdb.

`set evalopts iprecision` {**`4`** | **`8`**}

set evalopts rprecision

`se ev r`

Set the size of real numbers for CXdb.

`set evalopts rprecision` {**`4`** | **`8`**}

set fixed sched

`se fi s` *sfs*

Enable fixed scheduling in the process settings.

[*<process-list>*] **`set fixed sched`**

set format

`se fo`

Set the formats for displaying memory.

[*<process-list>*] [*<thread-list>*] **`set format`**
<memory-unit> *<format>*

set fpmode

`se fp`

Set the floating point mode for the process.

[*<process-list>*] **`set fpmode`** {**`ieee`** | **`native`**}

set handler `se h`

Set the handler for a specified eventpoint.

set handler *<event-specifier>* [, ...]
{*<event-handler>*}

set ignore `se i`

Set an ignore count for an eventpoint.

set ignore *<ignore-count>* *<event-specifier>* [, ...]

set logging `se l`

Enable logging for cmdlog.

set logging

set memory `se m`

Set the unit size for displaying memory.

[*<process-list>*] [*<thread-list>*]
set memory *<memory-unit>*

set noclobber `se n`

Enable the noclobber option for all viewports.

set noclobber

set path `se pa` *p=*

Set the search path for the process.

[*<process-list>*] **set path** *<directory-specifier>* [, ...]

set printopts maxarray `se pr m`

Set the maximum number of array elements to print.

set printopts maxarray *<number-of-elements>*

set printopts nopadding `se pr n`

Disable padding with leading zeros when printing.

set printopts nopadding

set printopts padding `se pr pa`

Enable padding with leading zeros when printing.

set printopts padding

set printopts precision `se pr pr`

Set the precision used to print floating point numbers.

set printopts precision *<width>*.*<precision>*

set pshell

`se ps`

Set the process shell.

[*<process-list>*] **set pshell** {**sh** | **csh**}

set remotewd

`se r`

Set the remote working directory.

[*<process-list>*] **set remotewd** *<directory-specifier>*

set seq

`se se`

Set the sequential mode (SEQ) bit.

[*<process-list>*] [*<thread-list>*] **set seq**

set shell

`se sh`

Set the type of shell invoked from within CXdb.

set shell {**sh** | **csh** | **tcsh** | **ksh** | **COVUE**}

set signal

se si

Set the actions for the specified signal.

[*<process-list>*] **set signal** *<signal-specifier>*
[{**stop** | **nostop**},] [{**pass** | **nopass**},]
[{**print** | **noprint**}]

set sqs

`se sq`

Set the sequential store enable (SQS) bit.

[*<process-list>*] [*<thread-list>*] **set sqs**

set step

`se st`

Set the stepping granularity.

[*<process-list>*] [*<thread-list>*] **set step** *<granularity>*

set threads

`se th`

Associate windows with particular threads.

set threads *<window>* [, ...]
[*<thread-number>* [, ...]]

set typehandler

`se t`

Define the default handler for all eventpoints of the specified type.

set typehandler *<eventtype-specifier>* [, ...]
{*<event-handler>*}

shell

`sh`

Invoke a shell.

`shell` [**`/`***<shell-specifier>*] [*<shell-commands>*]

signal process

`sig p`

Send a signal to the process.

[*<process-list>*] **`signal process`** *<signal-specifier>* [**`&`**]

signal thread

`sig t`

Send a signal to a specific thread of the process.

[*<process-list>*] *<thread>*
`signal thread` *<signal-specifier>* [**`&`**]

source

`sou`

Execute a CXdb command file.

`source` *<file-name>*

step

`ste`

s

Step to the next source unit.

[*<process-list>*] [*<thread-list>*] **`step`** [*<granularity>*] [*<count>*] [**`&`**]

step instruction

`ste i`

si, stepi

Step to the next instruction.

[*<process-list>*] [*<thread-list>*] **`step instruction`** [*<count>*] [**`&`**]

step over

`ste o`

so

Step from the current source unit of specified granularity to the next source unit of default granularity.

[*<process-list>*] [*<thread-list>*] **`step over`** [*<granularity>*] [*<count>*] [**`&`**]

stop

`sto`

Stop execution of the process.

[*<process-list>*] **`stop`**

T

trace instruction

`t i` *ti*

Set a tracepoint at an instruction.

[*<process-list>*] [*<thread-list>*]
`trace instruction` *<language-expression>*
[**{***<event-handler>***}**] [*<debugger-variable>*]

trace line

`t l` *tl*

Set a tracepoint at a source line.

[*<process-list>*] [*<thread-list>*]
`trace line` *<line-specifier>* [**{***<event-handler>***}**]
[*<debugger-variable>*]

trace routine

`t r` *tr*

Set a tracepoint at the beginning of a routine.

[*<process-list>*] [*<thread-list>*]
`trace routine` *<language-expression>*
[**{***<event-handler>***}**] [*<debugger-variable>*]

trace source

`t s` *ts*

Set a tracepoint at a source unit.

[*<process-list>*] [*<thread-list>*] **`trace source`**
<source-unit> [**{***<event-handler>***}**] [*<debugger-variable>*]

W

watch

`w`

Set a watchpoint to monitor an address range.

[*<process-list>*] [*<thread-list>*]
`watch` *<starting-address>*
[{ **. .** *<ending address>* | **:***<byte-count>*}]
[**{***<event-handler>***}**] [*<debugger-variable>*]

csd aliases

This section lists the most frequently used commands from the `csd` debugger, along with their CXdb equivalents. CXdb includes a set of predefined aliases for the `csd` commands listed here. To use these aliases, first enter the command `csd` at the `(CXdb)` prompt. If you then enter any of the `csd` commands listed in the right column, CXdb executes the associated commands listed in the left column.

If there is no exact CXdb equivalent for a `csd` command, suggested alternatives are listed wherever appropriate.

csd command	CXdb equivalent
`&`	Use `print loc(`*x*`)` or `print &`*x*.
`?`	`find window backward`
`alias`	Use `alias` command.
`assign`	`evaluate`
`call`	`print`
`catch`	Use `set signal` command.
`cregs`	`info cregisters`
`delete`	`remove event`
`down`	Use `frame` command.
`dump`	`backtrace`
`edit`	`edit`
`file`	`display file`
`format`	No equivalent.

`format decimal`	`set format byte dec;`
	`set format half dec;`
	`set format word dec;`
	`set format long dec;`
	`set format quad dec`

`format hex`	`set format byte hex;`
	`set format half hex;`
	`set format word hex;`
	`set format long hex;`
	`set format quad hex`

`fpmode ieee`	`set format ieee`
`fpmode native`	`set fpmode native`
`fpmode auto`	`set fpmode dual`
`func`	`info scope;` Use `backtrace` or `display routine` commands for more information.
`help`	`help`
`ignore`	Use `set signal` command.
`list`	Use `list` command.

csd command	CXdb equivalent
mode	No equivalent.
mode chained	clear seq
mode sequential	set seq
next all	next
nexti	next instruction
print	Use print command.
quit	quit
rerun	Use rerun command.
return	Use return command.
run	Use run command.
regs	info registers
set num_elements =	set printopts maxarray
set precision =	set printopts precision 10
set deref_aaregs	Change format in register window.
set dump_lfmt	Use info commands.
set dumpvregs	Use info vregisters command.
status	info event *
step	step
step all	step
stepi	step instruction
stop	No equivalent.
stop at	break line
stop in	break routine
stop if	event relation
stop threads	event spawn; event join
stopi at	break instruction
thread	info process
threads false	remove eventtype spawn, join
threads true	event spawn; event join
trace threads	event spawn { backtrace 1; echo 'thread spawned'; resume; }; event join { backtrace 1; echo 'thread joined'; resume; };
unalias	remove alias
up	Use up alias.
use	set path
vregs	info vregisters
whatis	info expression
when at	event reached line
when in	event reached routine

csd command	CXdb equivalent
`where`	`backtrace`
`whereis`	`info symbols`
`which`	`info symbols`

gdb aliases

This section lists the most frequently used commands from the gdb debugger, along with their CXdb equivalents. CXdb includes a set of predefined aliases for the gdb commands listed here. To use these aliases, first enter the command gdb at the (CXdb) prompt. If you then enter any of the gdb commands listed in the right column, CXdb executes the associated commands listed in the left column.

If there is no exact CXdb equivalent for a gdb command, suggested alternatives are listed wherever appropriate.

gdb command	CXdb equivalent
add-file	Use load object command.
b	break routine
commands	Use an eventpoint handler.
condition	Use if command within an eventpoint handler.
core-file	core
define	Use alias command.
delete	remove event
directory	add path
disable breakpoints	disable event
document	No equivalent.
down	frame -1
dump-me	Send kill command to CXdb from the shell.
enable breakpoints	enable event
exec-file	executable
forward-search	find window forward
handle	set signal
ignore	Use set ignore command.
info address	info expression
info comm-registers	info cregisters
info directories	info process
info display	info event
info files	info process
info functions	info symbols
info methods	No equivalent.
info sources	info objectmap
info types	info type
info variables	info symbols

gdb command	CXdb equivalent
`jump`	Use `goto line` or `goto address` command.
`list`	Use `display file` command.
`output`	No equivalent.
`printf`	No equivalent.
`printsyms`	`info symbols >`
`ptype`	`info type`
`reverse-search`	`find window backward`
`search`	No equivalent.
`set args`	Specify arguments with `run` or `rerun` command.
`set array-max`	`set printopts maxarray`
`set base`	Use `set format` and `examine`, or `print` with format options.
`set compile off`	No equivalent.
`set compile on`	No equivalent.
`set compiled-breakpoints`	No equivalent.
`set debug-flag`	No equivalent.
`set editing off`	No equivalent.
`set editing on`	No equivalent.
`set history expansion off`	No equivalent.
`set history expansion on`	No equivalent.
`set history file`	`add cmdlog`
`set history size`	No equivalent.
`set history write off`	`clear logging`
`set history write on`	`set logging`
`set parallel fixed`	`set fixed sched`
`set parallel off`	No equivalent.
`set parallel on`	`clear fixed sched`
`set pipeline off`	`set seq`
`set pipeline on`	`clear seq`
`set prettyprint`	No equivalent.
`set unionprint`	No equivalent.
`set prompt`	Done in .Xdefaults file.
`set screensize`	No equivalent.
`set verbose off`	No equivalent.
`set verbose on`	No equivalent.
`symbol-file`	Done automatically as needed.
`tbreak`	Use an eventpoint handler.
`term-status`	No equivalent.

gdb command	CXdb equivalent
`thread`	`info threads`
`tty`	Process interface window created automatically.
`undisplay`	No equivalent.
`unset environment`	`remove environment`
`until`	`finish loop`
`up`	`frame +1`

Maryland Windows keyboard functions

This section lists the keyboard functions used in Maryland Windows. The functions are divided into four groups:

- **Text scrolling functions**—These functions scroll the text inside a window.
- **Window movement functions**—These functions move between windows, close windows, raise and lower windows, resize windows, or move windows.
- **Window positioning functions**—These functions enable you to place a window or resize it.
- **Command window functions**—These functions enable you to scroll text in the CXdb command window or to edit the command line.

Text scrolling functions

Function	Key sequence
left character	**LEFT ARROW, CTRL-b**
right character	**RIGHT ARROW, CTRL-f**
left word	**META-b**
right word	**META-f**
down line	**DOWN ARROW, CTRL-n** (except command window)
up line	**UP ARROW, CTRL-p** (except command window)
down screen	**CTRL-v**
up screen	**META-v**
top of buffer	**META-<**
bottom of buffer	**META->**

Window positioning functions

Function	Key sequence
up	**k, CTRL-p**
down	**j, CTRL-n**
right	**l, META-f**
left	**h, META-b**
anchor window	**SPACEBAR**

Window movement functions

Function	Key sequence
next window	**META-n**
previous window	**META-p**
move window	**META-m**
resize window	**META-z**
lower window	**META-o**
raise window	**META-r**
close window	**META-k** (except command window)

Command window functions

Function	Key sequence
down line	**META-A**
up line	**META-B**
beginning of line	**CTRL-a**
end of line	**CTRL-e**
down history	**CTRL-n**
up history	**CTRL-p**
delete character	**DELETE, CTRL-d**
delete back character	**BACKSPACE, CTRL-h**
transpose characters	**CTRL-t**
delete word	**META-DELETE, META-d**
delete back word	**META-BACKSPACE, META-h**
set mark	**CTRL-@**
exchange point and mark	**CTRL-x**
kill to end of line	**CTRL-k**
kill region	**CTRL-w**
copy region	**META-w**
yank	**CTRL-y**
start over	**CTRL-g**
redrawn screen	**CTRL-l, CTRL-r**
lowercase	**META-l**
uppercase	**META-u**
capitalize	**META-c**
universal argument	**CTRL-u**
argument digit	**META-0...META-9**
newline	**RETURN, LINEFEED, CTRL-j, CTRL-m**

Mouse functions

This section lists the default functions you can perform with the mouse in the source window if you are using CXdb in CXwindows mode. You can change these default mouse functions by modifying the X resource specifications for CXdb. (Refer to the *CONVEX CXdb User's Guide* for more details.)

Key/button	Function
Left button	Text select
Middle button	Text select + `info expression`
Right button	Text select + `print` expression
CTRL-Left button	Source select
CTRL-Middle button	Source select + `info expression`
CTRL-Right button	Source select + `print` expression
META-Left button	Set a breakpoint
META-Middle button	Source select + `info source`
META-Right button	Text select + `print` indirect
CTRL-META-Left button	Run to a location
CTRL-META-Middle button	`info line`
CTRL-META-Right button	Source select + `print` indirect
SHIFT-META-Left button	Set a tracepoint
SHIFT-META-Middle button	(none)
SHIFT-META-Right button	(none)

CONVEX Computer Corporation
Richardson, Texas USA

Document No. 710-015630-001