

# eCMD Command Line Interface

Version .5

Contact: Chris Engel / Paul Prah

IBM Confidential

## Table of Contents

<b>1 Introduction.....</b>	<b>5</b>
<b>2 Usage Instructions.....</b>	<b>5</b>
2.1 Environment Setup.....	5
2.2 Error Handling.....	5
2.3 Required Input Files.....	5
2.4 Optional Arguments.....	5
<b>3 eCMD Common Commands.....</b>	<b>6</b>
3.1 Common Command Arguments.....	6
3.1.1 Targeting Options.....	6
3.1.2 Data Output Formatting (-o<format>).....	6
3.1.3 Data Input Formatting (-i<format>).....	10
3.1.4 Data Input Bit Modifiers (-b<modifier>).....	10
3.1.5 Command Help (-h).....	10
3.1.6 Trace Options (-trace).....	10
3.1.7 Multiple Command Mode (-stdin).....	11
3.1.8 Quiet Mode (-quiet).....	11
3.2 Chip Display/Alter Commands.....	13
3.2.1 checkrings.....	13
3.2.2 getarray.....	13
3.2.3 getbits.....	14
3.2.4 getcfam.....	16
3.2.5 getlatch.....	17
3.2.6 getringdump.....	18
3.2.7 getscom.....	19
3.2.8 getspy.....	21
3.2.9 gettracearray.....	22
3.2.10 pollscm.....	22
3.2.11 putarray.....	24
3.2.12 putbits.....	24
3.2.13 putcfam.....	25
3.2.14 putlatch.....	26
3.2.15 putpattern.....	27
3.2.16 putscom.....	28
3.2.17 putspy.....	29
3.2.18 ringcache.....	30
3.2.19 sendcmd.....	31
3.3 Processor Functions.....	32
3.3.1 getfpr.....	32
3.3.2 getgpr.....	32

3.3.3	getspr.....	33
3.3.4	putfpr.....	34
3.3.5	putgpr.....	35
3.3.6	putspr.....	35
3.4	Memory Display/Alter Functions.....	37
3.4.1	cacheflush.....	37
3.4.2	getmemdma.....	37
3.4.3	getmemmemctrl.....	38
3.4.4	getmemproc.....	39
3.4.5	putmemdma.....	40
3.4.6	putmemmemctrl.....	40
3.4.7	putmemproc.....	41
3.5	Miscellaneous Commands.....	43
3.5.1	Deconfig.....	43
3.5.2	ecmdquery.....	43
3.5.3	getconfig.....	45
3.5.4	makespsystemcall.....	46
3.5.5	reconfig.....	46
3.5.6	setconfig.....	47
3.6	System Functions.....	49
3.6.1	istep.....	49
3.6.2	setclockspeed.....	49
3.6.3	startclocks.....	50
3.6.4	stopclocks.....	51
3.7	GPIO Commands.....	52
3.7.1	getgpilatch.....	52
3.7.2	getgpiopin.....	52
3.7.3	gpioconfig.....	53
3.7.4	putgpilatch.....	54
3.8	I2C Commands.....	55
3.8.1	geti2c.....	55
3.8.2	i2creset.....	56
3.8.3	puti2c.....	56
3.9	VPD Commands.....	58
3.9.1	getvpdimage.....	58
3.9.2	getvpdkeyword.....	58
3.9.3	putvpdimage.....	59
3.9.4	putvpdkeyword.....	60
3.10	Simulation Commands.....	62
3.10.1	simaet.....	62
3.10.2	simcheckpoint.....	62
3.10.3	simclock.....	62
3.10.4	simecho.....	63

3.10.5	simexit.....	63
3.10.6	simEXPECTFAC.....	64
3.10.7	simexpecttcfac.....	64
3.10.8	simgetcurrentcycle.....	65
3.10.9	simGETFAC.....	65
3.10.10	simGETFACX.....	66
3.10.11	simgettcfac.....	66
3.10.12	simgethierarchy.....	67
3.10.13	siminit.....	67
3.10.14	simPUTFAC.....	68
3.10.15	simPUTFACX.....	68
3.10.16	simputtcfac.....	69
3.10.17	simrestart.....	69
3.10.18	simSTKFAC.....	70
3.10.19	simstktcfac.....	70
3.10.20	simSUBCMD.....	71
3.10.21	simtckinterval.....	71
3.10.22	simUNSTICK.....	72
3.10.23	simunsticktcfac.....	72
<b>4</b>	<b>CIP (Cronus/IP) Extension Commands.....</b>	<b>74</b>
4.1	Processor Functions.....	74
4.1.1	cipbreakpoint.....	74
4.1.2	cipinstruct.....	74
<b>5</b>	<b>CRO (Cronus) Extension Commands.....</b>	<b>76</b>
5.1	Memory Display/Alter Commands.....	76
5.1.1	crodmrandtest.....	76
5.1.2	crogetl2data.....	77
5.1.3	crogetl2dir.....	77
5.1.4	crogetl2.....	78
5.1.5	croputl2data.....	79
5.1.6	croputl2dir.....	80
5.1.7	croputl2.....	80
<b>6</b>	<b>EIP (Eclipz IP) Extension Commands.....</b>	<b>82</b>
6.1	Processor Functions.....	82
6.1.1	eipgetslb.....	82
6.1.2	eipprocleanup.....	82
6.2	Miscellaneous Commands.....	84
6.2.1	crogetconfig.....	84
6.2.2	crosetconfig.....	84

## 1 Introduction

This document has been created using OpenOffice, a copy of the OpenOffice Suite can be obtained from: <http://mcweb.boeblingen.de.ibm.com/OpenOffice/>

This document describes the eCMD command line set. These commands are all written in C code against the eCMD C-API and as such can run against any implementation of the eCMD C-API. Currently this means scripts written to use the eCMD command line will be able to run against GFW for I/P/Z Series or Cronus without any modification.

For the latest list of all available commands please see the eCMD web page at <http://rhea.rchland.ibm.com/eCMD/>.

## 2 Usage Instructions

### 2.1 *Environment Setup*

To run the eCMD command line interface requires a few environment variables be setup prior to executing any commands. The exact method to setup these variables may be different depending on which implementation(plugin) of the C-API you plan on running but will be documented here in the future.

### 2.2 *Error Handling*

All errors encountered running an eCMD command will display a message to the screen and will return a non-zero return code to the calling shell.

### 2.3 *Required Input Files*

eCMD queries all required files (ie scandefs/help text) from the dll that it is using. In the case of IP Series when running on the FSP commands requiring external input files may not run unless a NFS mount is setup to source these files.

### 2.4 *Optional Arguments*

All eCMD optional arguments start with a '-' character, these arguments can be specified in any order on the command line.

## 3 eCMD Common Commands

These are the core command line functions available through the eCMD interface and the syntax of the command. The help text is commented with the text 'Core Common Function' for all commands that are part of the core eCMD subset. Other Series or Cronus specific commands will be specified uniquely as well.

### 3.1 Common Command Arguments

These are common arguments that are supported on most of the eCMD commands.

#### 3.1.1 Targeting Options

Most eCMD functions use the following commands to specify which chip/node/cage you are trying to target in the system. How these options map to physical hardware will be defined by the eCMD team and documented in a separate document for each product.

The valid targeting options:

- -k# (cage)
- -n# (node)
- -s# (slot)
- -p# (position)
- -c# (core)
- -t# (thread)

These options accept the following number strings:

- -p0           Single digit
- -p1,5,10    Comma separated list
- -p2..7       Range of positions
- -p1,2..5,9   Mixture of single and ranges
- -pall         Target all possible configured positions

The -t (thread) argument takes a special option -talive to specify all alive threads.

#### 3.1.2 Data Output Formatting (-o<format>)

The -o argument is used by eCMD to decide how the data should be displayed to the user. The -o argument takes a format string, the available formats are displayed below:

**Left-aligned Hex : -ox**

## eCMD Command Line Interface

```

FORMAT: X
gr      k0:n0:s0:p00:c0      000000000000000000000000
gr      k0:n0:s0:p01:c0      000000000000000000000000
gr      k0:n0:s0:p02:c0      000000000000000000000000

```

### Left-aligned Hex Words : -oxw

```

FORMAT: XW
gr      k0:n0:s0:p00:c0      00000000 00000000 00000000
gr      k0:n0:s0:p01:c0      00000000 00000000 00000000
gr      k0:n0:s0:p02:c0      00000000 00000000 00000000

```

### Left-aligned Hex Word Columns : -oxw2

```

FORMAT: XW2
gr      k0:n0:s0:p00:c0
0: 00000000 00000000
2: 00000000
gr      k0:n0:s0:p01:c0
0: 00000000 00000000
2: 00000000

```

### Right-aligned Hex : -oxr

```

FORMAT: XR
gr      k0:n0:s0:p00:c0      000000000000000000000000
gr      k0:n0:s0:p01:c0      000000000000000000000000
gr      k0:n0:s0:p02:c0      000000000000000000000000

```

## Right-aligned Hex Words : -oxrw

```

FORMAT: XRW
gr      k0:n0:s0:p00:c0      00000000 00000000 00000000
gr      k0:n0:s0:p01:c0      00000000 00000000 00000000
gr      k0:n0:s0:p02:c0      00000000 00000000 00000000

```

## Right-aligned Hex Word Columns : -oxrw2

```

FORMAT: XRW2
gr      k0:n0:s0:p00:c0
0: 00000000 00000000
2: 00000000
gr      k0:n0:s0:p01:c0
0: 00000000 00000000
2: 00000000

```

## Binary : -ob

[illegible]

## Binary Nibbles : -obn

[illegible]

**Binary Nibble Columns : -obn8**

```

FORMAT: BN8
gr      k0:n0:s0:p00:c0

      0          1          2          3
      0123 4567 8901 2345 6789 0123 4567 8901
00: 0000 0000 0000 0000 0000 0000 0000 0000
08: 0000 0000 0000 0000 0000 0000 0000 0000
16: 0000 0000 0000 0000 0000 0000 0000 0000

```

**Binary Words : -obw**

```

FORMAT: BW
gr      k0:n0:s0:p00:c0      00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000

```

**Binary Word Columns : -obw1**

```

FORMAT: BW1
gr      k0:n0:s0:p00:c0

      0          1          2          3
      01234567890123456789012345678901
0: 00000000000000000000000000000000
1: 00000000000000000000000000000000
2: 00000000000000000000000000000000

```

**Simulation Outputs : X-States are simulation states that aren't valid on real hardware, choosing one of the following X-State in a hardware environment will just be equivalent to the binary output.**

**X-State Binary : -obX**

```

FORMAT: BX
gr      k0:n0:s0:p00:c0      00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
gr      k0:n0:s0:p02:c0      00000000000000000000000000000000

```

**X-State Binary Nibbles : -obXn**

```

FORMAT: BXN
gr      k0:n0:s0:p00:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
gr      k0:n0:s0:p01:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
gr      k0:n0:s0:p02:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

```

**X-State Binary Nibble Columns : -obXn8**

```

FORMAT: BXN8
gr      k0:n0:s0:p00:c0

      0          1          2          3
      0123 4567 8901 2345 6789 0123 4567 8901
00: 0000 0000 0000 0000 0000 0000 0000 0000
08: 0000 0000 0000 0000 0000 0000 0000 0000
16: 0000 0000 0000 0000 0000 0000 0000 0000

```



**X-State Binary Words : -obXw**

```

FORMAT: BXW
gr      k0:n0:s0:p00:c0      00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000000

```

**X-State Binary Word Columns : -obXw1**

```

FORMAT: BXW1
gr      k0:n0:s0:p00:c0

      0      1      2      3
01234567890123456789012345678901
0: 00000000000000000000000000000000
1: 00000000000000000000000000000000
2: 00000000000000000000000000000000

```

**Memory Output : -omem**

```

FORMAT: MEM
gr      k0:n0:s0:p00
0000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF
0000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF

```

**Memory Output – Ascii Decode : -omema**

```

FORMAT: MEMA
gr      k0:n0:s0:p00
0000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISisTHEasciiTE]
0000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [XT.....]

```

**Memory Output – Ebcedic Decode : -omeme**

```

FORMAT: MEME
gr      k0:n0:s0:p00
0000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISisTHEebcedic]
0000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [TEXT.....]

```

**Memory Output – D-Card Format : -omemd**

```

FORMAT: MEMD
gr      k0:n0:s0:p00
D 0000000000000100 FEEDBEEFFEEDBEEF 0
D 0000000000000108 FEEDBEEFFEEDBEEF 1
D 0000000000000110 FEEDBEEFFEEDBEEF 0
D 0000000000000118 FEEDBEEFFEEDBEEF 1

```

**Spy Enum Output – Only valid with getspy command : -oenum**

```

FORMAT: ENUM
gr      k0:n0:s0:p00:c0 OFF
gr      k0:n0:s0:p00:c1 ON

```

### 3.1.3 Data Input Formatting (-i<format>)

The -i argument is used by eCMD to determine how to read the data provided by the user.

**Left-aligned Hex : -iX**

**Right-aligned Hex : -iXR**

**Binary : -iB**

**Spy Enum – Only valid with putspy command : -ienum**

### 3.1.4 Data Input Bit Modifiers (-b<modifier>)

The -b argument allows the user to specify a bit operation to perform on the data, this forces eCMD to do a read-modify-write on the data to perform the operation.

**Or : -bor**

Read data from hardware, or in data specified, write data back to hardware.

**And : -band**

Read data from hardware, and with data specified, write data back to hardware.

### 3.1.5 Command Help (-h)

All commands accept the '-h' argument, when specified eCMD will echo back the help text for the command. This text is the same as shown below in this document.

### 3.1.6 Trace Options (-trace)

All commands accept the -trace argument which allows the user to turn on different traces. The format of the trace is common between all major eCMD plugins but the mechanism for displaying the trace may be different. For example Cronus displays traces to stdout in the shell you are running, where as IP GFW writes traces to logs on the FSP.

The trace option syntax is : **-trace=<mode1>[,<mode2>]**

Example : -trace=scan,prcd

Trace Options :

**-trace=scan**

Scan tracing : Displays all ring/scom/spy accesses to the hardware

**-trace=prcd**

Procedure tracing : Displays the procedure trace as defined by the “HW control

procedure” specification.

### 3.1.7 Multiple Command Mode (-stdin)

The -stdin option allows you to specify multiple commands to be run within one execution of the command line client. There are three ways to do this:

#### Single line command :

```
> echo “ecmdquery version ; getscom pu 800000” | $ECMD_EXE -stdin
```

#### Input with a text file :

```
> $ECMD_EXE -stdin < commands.txt
```

#### Where commands.txt is:

```
ecmdquery version;  
getscom pu 800000
```

#### Input from stdin:

```
> $ECMD_EXE -stdin  
<type commands>  
ecmdquery version  
<press Ctrl-D to stop>
```

### 3.1.8 Quiet Mode (-quiet)

Quiet mode turns off some messages that eCMD will display to the screen. Currently the things disabled are the following:

- Command echo (reprint of command run after execution)
- Target message on write operations

Here is an example of the differences:

## eCMD Command Line Interface

```
> putscom pu 800000 0 -pall
p6      k0:n0:s0:p00
ecmd_x86 putscom pu 800000 0 -pall

> putscom pu 800000 0 -pall -quiet
>

> getscom pu 800000
p6      k0:n0:s0:p00      0x0000000000000000
ecmd_x86 getscom pu 800000

> getscom pu 800000 -quiet
p6      k0:n0:s0:p00      0x0000000000000000
>
```

## 3.2 Chip Display/Alter Commands

### 3.2.1 checkrings

#### Syntax:

Syntax: checkrings <ChipSelect> <RingSelect> [-v] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: checks for stuck bits and verifies scanning length by scanning ones and zeros to scan chain.

Parameters:

-----  
ChipSelect Specifies the chip to operate on.

RingSelect Specifies chip ring to operate on. Use "all" for all rings. For a list of available rings, use the query command.

Ex: ecmdquery rings memctrl  
ecmdquery rings pu

-v [optional] Display detailed ring failure data

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----  
Example: checkrings pu all -p0,1 -c0  
checkrings memctrl int -pall

#### Examples:

```
> checkrings test all
Performing 1's test on testring ...
Performing 0's test on testring ...
Performing 1's test on sgxbs ...
Performing 0's test on sgxbs ...
ecmd.exe checkrings test all
```

### 3.2.2 getarray

#### Syntax:

## eCMD Command Line Interface

Syntax: `getarray <ChipSelect> <ArrayName> <ArrayIndex> [NumEntries] [-o<format>]`  
`[-exp <data> [-i<format>]] [-mask <data>]] [-k#] [-n#] [-s#] [-p#]`  
`[-c#]`

ECMD: Core Common Function

Function: Read the specified chip array.

Parameters:

-----  
ChipSelect Chip to read array data from.

ArrayName Name of array to read from.

ArrayIndex Array Index in right aligned hex.

NumEntries[opt] Number of consecutive entries to display  
Address is incremented by 1

-o<format>[opt] Output Format : default 'xl'  
Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect and mask data : default 'xl'  
Run 'ecmdquery formats' to view available formats

-exp [optional] Provide expected data. Returns error if expected != actual. No  
error/no data returned if expected == actual.

-mask [opt] Array data is AND'ed with the mask bits. Only for use with -exp.

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-----  
Examples: > `getarray pu xgpr0 deadbeef80000000 -p0,1 -c1`

## Examples:

### 3.2.3 getbits

#### Syntax:

Syntax: `getbits <ChipSelect> <RingName> <StartPos> <NumBits> [-exp <data>]`  
`[-k#] [-n#] [-s#] [-p#] [-c#] [-o<format>] [-i<format>]`

## eCMD Command Line Interface

```
getbits <ChipSelect> <RingName> -f<filename>          [-k#] [-n#] [-s#] [-p#] [-c#]
```

ECMD:            Core Common Function

Function:        Long scans bits out of a chip's selected ring. (non-destructive)  
Ring is either displayed to screen or written to file specified  
with the -f option.

Parameters:

-----

ChipSelect       Specifies the chip to operate on.

RingName        Specifies chip ring to operate on. For a list of available  
rings, use the ecmdquery command.

                 Ex: ecmdquery rings memctrl  
                     ecmdquery rings pu

StartPos        Specifies starting bit position in Decimal.

NumBits         Specifies number of bits to get from starting position (Decimal)  
Specify the keyword 'end' to fetch from startPos to end of ring.

-exp [optional] Provide expected data. Returns error if expected != actual. No  
error/no data returned if expected == actual.  
Format specified by -i<format>

-o<format>[opt] Specifies the format type of the output : default 'b'  
Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect data : default 'b'  
Run 'ecmdquery formats' to view available formats

-f<filename>[o] Specifies the filename that the ring data should be written to  
Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_ASCII format

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-----

Examples:        > getbits pu stat 0 15  
                 > getbits pu cp\_fxu 0 32 -oxw -exp feedbeef  
                 > getbits memctrl idreg 16 end  
                 > getbits memctrl cp\_fxu -ffxuRingDump.dump

## Examples:

```
> getbits test idreg 0 32
test    k0:n0:s0:p00          idreg(0:31)
0b11111110111011011011111011101111
ecmd.exe getbits test idreg 0 32

> getbits test idreg 0 16 -ox
test    k0:n0:s0:p00          idreg(0:15)
0xFEED
ecmd.exe getbits test idreg 0 16 -ox
```

## 3.2.4 getcfam

### Syntax:

Syntax: getcfam <ChipSelect> <CFAMAddr> [-k#] [-n#] [-s#] [-p#]  
 [-o<format>] [-i<format>] [-exp <data> [-mask <data>]]

ECMD: Core Common Function

Function: Gets CFAM Registers through FSI

Parameters:

-----  
 ChipSelect Chip to get CFAM data from.

CFAMAddr Address in hex.

-exp [optional] Provide expected data. Returns error if expected != actual. No error/no data returned if expected == actual.

-mask [opt] Register data is AND'ed with the mask bits.  
 Only for use with -exp.

-o<format>[opt] Specifies the format type of the output : default 'xl'  
 Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect and mask data : default 'xl'  
 Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----  
 Examples: > getcfam pu 6 -p0,1  
 > getcfam memctrl 800009 -exp feed0000 -mask ffff0000

### Examples:



### 3.2.5 getlatch

#### Syntax:

Syntax: getlatch <ChipSelect> [<RingName>] <LatchName> [<Start> <Numbits>] [-exact | -partial]

[-exp <value>] [-o<format>]  
[-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Gets values for specified latch names in a ring. The latch names in the scandef file are searched for the substring LatchName for a match. Each register containing the pattern-matched substring will be printed to the screen.

With the -exact option, eCMD searches for an exact match, and will return only the first latch that exactly matches (excluding any parentheses). This option also enables searching the scandef with a hash file which greatly increases performance.

The -nocompress flag turns off concatenation of all latches of a register in the scandef and displays on separate lines as they appear in the scandef.

#### Parameters:

```
-----
ChipSelect      Chip to get data from.

RingName  [opt] Specifies chip ring to operate on. For a list of available
              rings, use the ecmdquery command.
              NOTE : If not specified all rings in scandef are searched

              Ex:  ecmdquery rings memctrl
                   ecmdquery rings pu

LatchName      Desired latch to find in the ring. (case insensitive)

Start          [opt] Starting bit position within the latch. Specify with Numbits.

Numbits        [opt] Number of bits to get. Specify along with Start. If out of range,
              and -exact not selected, prints a warning message instead of the
              data. If -exact is selected, immediately returns an error.

-exact         [opt] No pattern matching. Instead, search for exact latch name.
(default)

-partial       [opt] Use pattern matching to find latch name, can be considerably
slower.

-exp [optional] Provide an expected-value as the last argument. Returns error if
              data miscompare, else nothing.
              Format specified by -i<format>

-o<format>[opt] Specifies the format type of both the output and
              the expect-value
              Defaults to 'b' for < 8 bits and 'xl' for >= 8 bits.
              Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect data
```

## eCMD Command Line Interface

Run 'ecmdquery formats' to view available formats

- k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.
- n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.
- s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.
- p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.
- c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----

Examples:

```
getlatch pu cp_abist LATCH0
getlatch pu cp_fpu GCP.FP_FPU.FP_UNIT0.FP_REG -ox -exact -expect
feed -ix
getlatch pu MYLATCH
```

### Examples:

```
> getlatch test sgxbs ACCESS
test    k0:n0:s0:p00
ACCESS.SNPBUF 0b0
ecmd.exe getlatch test sgxbs ACCESS
```

## 3.2.6 getringdump

### Syntax:

Syntax: getringdump <ChipSelect> <RingName1> [<RingName2> ...] [-unsorted]  
[-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Gets values for all latches in the entire scan ring.

NOTE: The entire ring entry from the scandef is read in and then sorted in alphabetical order. Then all registers are pushed together and displayed in 0..n order. To disable this use -unsorted

Parameters:

-----

ChipSelect Chip to get data from.

RingName Specifies one or multiple chip rings to operate on. For a list of available rings, use the ecmdquery command.

Ex: ecmdquery rings memctrl  
ecmdquery rings pu

-unsorted [opt] Don't sort ring dump, display in scandef order

## eCMD Command Line Interface

```
-k# [optional] Specify which cage to act on (0 is default). Specify -kall
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall
to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify
-call to act on all cores.
```

Notes:            Output is binary for latches <= 8 bits in length and xl for > 8.

```
Examples:      getringdump memctrl int
               getringdump pu gps_fuse
               getringdump pu gps_fuse gps_abist cp ras
```

### Examples:

[illegible]

### 3.2.7 getscom

### Syntax:

Syntax: getscom <ChipSelect> <ScanCommAddr> [-v] [-k#] [-n#] [-s#] [-p#] [-c#]  
[-o<format>] [-exp <data>] [-i<format>] [-mask <data>]]

ECMD: Core Common Function

Function: Gets Scan Communications registers.

Parameters:

## eCMD Command Line Interface

```
-----
ChipSelect      Chip to get scancomm data from.

ScanCommAddr    Address in hex.

Start           [opt] Starting bit position within the scom.  Specify with numbits.

Numbits         [opt] Number of bits to display. Specify with Start. If Start and
Numbits         are not specified, start = 0 and numbits is the bitlength of
                  scancomm data.

-i<format>[opt] Specifies the format type of expect and mask data : default 'xl'
                  Run 'ecmdquery formats' to view available formats

-exp [optional] Provide expected data. Returns error if expected != actual. No
                  error/no data returned if expected == actual.

-mask           [opt] Scom data is AND'ed with the mask bits. Only for use with -exp.

-o<format>[opt] Specifies the format type of the output : default 'xl'
                  Run 'ecmdquery formats' to view available formats

-v [optional] Print out Scan Comm bit meaning if available

-k# [optional] Specify which cage to act on (0 is default). Specify -kall
                  to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
                  to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall
                  to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify
                  -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify
                  -call to act on all cores.

-----

Examples:      > getscom pu 6 -p0,1
                  > getscom memctrl 800009 -exp feed000000000001 -mask
ffff00000000ffff
```

## Examples:

```
> getscom test 800000
test   k0:n0:s0:p00      FEEDBEEFAAAAAAAA
ecmd.exe getscom test 800000

> getscom test 800000 -obn8
test   k0:n0:s0:p00
      0      1      2      3
      0123 4567 8901 2345 6789 0123 4567 8901
00: 1111 1110 1110 1101 1011 1110 1110 1111
08: 1010 1010 1010 1010 1010 1010 1010 1010
16: 0000 0000 0000 0000 0000 0000 0000 0000
ecmd.exe getscom test 800000 -obn8
```

### 3.2.8 getspy

#### Syntax:

Syntax: getspy <ChipSelect> <SpyName> [<Start> <Numbits>] [-exp <value>]  
 [-o<format>] [-i<format>] [-v] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Display specified spy, works for edials, idials and aliases.  
 If a spy ecc error is detected all the ecc groupings will be displayed along with a mask showing which bits are in error.

#### Parameters:

```
-----
ChipSelect      Chip to get data from.

SpyName         Desired spy name. (case insensitive)

Start           [opt] Starting bit position within the spy. Specify with Numbits.
                  Only valid with non-enumerated spy's

Numbits         [opt] Number of bits to get. Specify along with Start.
                  Only valid with non-enumerated spy's

-exp [optional] Provides an expected value as the last argument. Returns error
                  only if miscompare.
                  Format specified by -i<format>

-o<format>[opt] Specifies the format type of the output
                  Default format for non-enumerated spys : 'xl'
                  Default format for enumerated spys : 'enum'
                  Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect data
                  Default format for non-enumerated spys : 'xl'
                  Default format for enumerated spys : 'enum'
                  Run 'ecmdquery formats' to view available formats

-v [optional] Enable verbose printing of spy information.
                  Displays all groups of a spy
                  Displays all ECC Checkers

-k# [optional] Specify which cage to act on (0 is default). Specify -kall
                  to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
                  to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall
                  to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify
                  -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify
                  -call to act on all cores.
-----
```

Examples: getspy pu MYALIAS  
 getspy pu REVERSE 16 64 -ox -exp aaaa5555

**Examples:****3.2.9 gettracearray****Syntax:**

Syntax: `gettracearray <ChipSelect> <ArrayName> [<ArrayName2> ...] [-o<format>] [-nostopstart] [-k#] [-n#] [-s#] [-p#] [-c#]`

ECMD: Core Common Function

Function: Read the specified chip trace array.

Parameters:

-----  
ChipSelect Chip to read array data from.

ArrayName Name of array to read from.

-o<format>[opt] Output Format : default 'xl'  
Run 'ecmdquery formats' to view available formats

-nostopstart[o] Don't stop and start the trace arrays while dumping.

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----  
Examples: > `gettracearray pu fbc -p0,1`

**3.2.10 pollscm****Syntax:**

Syntax: `pollscm <ChipSelect> <ScanCommAddr> [-exp <data> [-mask <data>]] [-o<format>] [-i<format>] [-limit #[s|c]] [-interval #[s|c]] [-verbose] [-k#] [-n#] [-s#] [-p#] [-c#]`

ECMD: Core Common Function

Function: Repeatedly gets Scan Communications registers until expected data matches actual data or until polling limit is reached.

Parameters:

## eCMD Command Line Interface

```
-----
ChipSelect      Chip to get scancom data from.

ScanCommAddr    Address in hex.

-exp            [opt] Provide expected data. Returns error if expected != actual. No
                  error/no data returned if expected == actual.

-mask          [opt] Scom data is AND'ed with the mask bits before checking against
                  expected value.

-o<format>[opt] Specifies the format type of the output : default 'xl'
                  Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of expect and mask data : default 'xl'
                  Run 'ecmdquery formats' to view available formats

-limit #       [opt] Max polling number in iterations, seconds, or cycles. To specify
                  in seconds, append an 's' to #. To specify number of cycles for
                  simulation, append a 'c' to #. If limit is not specified,
                  defaults to 1000 iterations. If limit = 0, polls indefinitely.

-interval #    [opt] Time between getscoms. To specify in seconds, append an 's'
                  to #. To specify number of cycles for simulation, append a
                  'c' to #. If interval is not specified it defaults to 5secs.

-verbose       [opt] Prints warning message after each getscom if actual != expected.

-k#            [optional] Specify which cage to act on (0 is default). Specify -kall
                  to act on all cages.

-n#            [optional] Specify which node to act on (0 is default). Specify -nall
                  to act on all nodes.

-s#            [optional] Specify which slot to act on (0 is default). Specify -sall
                  to act on all slots.

-p#            [optional] Specify which chip position to act on (0 is default). Specify
                  -pall to act on all chips.

-c#            [optional] Specify which processor core to act on (0 is default). Specify
                  -call to act on all cores.
-----

Examples:      pollscm pu 800009 -exp feed000000000001 -limit 30s -interval 10s
-verbose -p1
               pollscm pu 800009 -exp feed000000000001 -mask ffff00000000ffff
-limit 10
               pollscm memctrl 400020 -limit 100000c -interval 5000c
```

## Examples:

```
> pollscm test 800000 -exp FEED0000 -limit 5
test   k0:n0:s0:p00:c0:t0 Polling address 800000...
ERROR: (ECMD): Data miscompare occurred at address: 00800000
test   k0:n0:s0:p00:c0:t0 Polling address 800000...
Actual       : FEEDBEEF AAAAAAAA 00000000
Expected     : FEED0000
ecmd.exe pollscm test 800000 -exp FEED0000 -limit 5
```

### 3.2.11 putarray

#### Syntax:

Syntax: putarray <ChipSelect> <ArrayName> <ArrayIndex> <ArrayData> [-i<format>]  
 [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Write the specified data to the specified chip array.

Parameters:

-----  
 ChipSelect Chip to put array data to.

ArrayName Name of array to write to.

ArrayIndex Array Index in right aligned hex.

ArrayData Data to write to array: default "x"  
 Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'  
 Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
 to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
 to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
 to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
 -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
 -call to act on all cores.

-----  
 Examples: > putarray pu xgpr 0 deadbeef80000000 -p0,1 -c1

#### Examples:

### 3.2.12 putbits

#### Syntax:

Syntax: putbits <ChipSelect> <RingName> <StartPos> <Data> [-i<format>] [-b<modifier>]  
 [-k#] [-n#] [-s#] [-p#] [-c#]

putbits <ChipSelect> <RingName> -f<filename> [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function



## eCMD Command Line Interface

Function: Put bits to the specified chip ring. The data either comes from the command line or from the file specified with the -f option.

### Parameters:

---

ChipSelect	Specifies the chip to operate on.
RingName	Specifies chip ring to operate on. For a list of available rings, use the ecmdquery command.  Ex: ecmdquery rings memctrl ecmdquery rings pu
StartPos	Specifies starting bit position in Decimal.
Data	Bits to insert into chip ring. Format specified by -i<format>
-i<format>[opt]	Specifies the format type of input data : default 'b' Run 'ecmdquery formats' to view available formats
-b<mod>[opt]	Bit modifier to apply to current ring data. Run 'ecmdquery formats' to view available modifiers
-f<filename>[o]	Specifies the filename that the ring data should be read from Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_ASCII format
-k# [optional]	Specify which cage to act on (0 is default). Specify -kall to act on all cages.
-n# [optional]	Specify which node to act on (0 is default). Specify -nall to act on all nodes.
-s# [optional]	Specify which slot to act on (0 is default). Specify -sall to act on all slots.
-p# [optional]	Specify which chip position to act on (0 is default). Specify -pall to act on all chips.
-c# [optional]	Specify which processor core to act on (0 is default). Specify -call to act on all cores.

---

Example: > putbits pu int 567 ABAB -ix -p0,1 -c1  
> putbits pu int 23 011X001X -p0 -iX  
> putbits pu int -fintRing.dump

## Examples:

### 3.2.13 putcfam

#### Syntax:

Syntax: putcfam <ChipSelect> <CFAMAddr> [<Start> <Numbits>] <Data> [-i<format>]  
[-b<modifier>] [-k#] [-n#] [-s#] [-p#]

ECMD: Core Common Function

## eCMD Command Line Interface

```
Function:      Write the specified data to the specified chip CFAM register

Parameters:
-----
ChipSelect    Chip to put CFAM data to.

CFAMAddr      Address in right aligned hex.

Start         [opt] Starting bit position within the register.  Specify with numbits.

Numbits       [opt] Number of bits to insert. Specify with Start. If Start and Numbits
                are not specified, start = 0 and numbits is calculated from
                length of data string, rest of cfam register is padded with zeros.

Data          Data to insert into Register.
                Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'
                Run 'ecmdquery formats' to view available formats

-b<mod>[opt]   Bit modifier to apply to current ring data.
                Run 'ecmdquery formats' to view available modifiers

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
                to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
                to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
                to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
                -pall to act on all chips.

-----

Examples:      > putcfam pu 600000 deadbeef  -p0,1
                > putcfam memctrl 2010 001001010110 -ib
                > putcfam l3 40320 00008000 -bor -p12
```

### 3.2.14 putlatch

#### Syntax:

```
Syntax: putlatch <ChipSelect> [<RingName>] <LatchName> [<Start> <Numbits>] <Data>
                [-exact | -partial] [-i<format>] [-b<modifier>] [-k#] [-n#] [-s#] [-p#]
                [-c#]
```

ECMD: Core Common Function

Function: Puts a value for a specified register into a ring. The first register in the scandef file that exactly matches the RegName (not including parenthesis) will be used. If the register is broken into multiple lines, the register lengths are concatenated to form one complete register.

With the -exact option, eCMD searches for an exact match, and will alter only the first latch that exactly matches (excluding any parentheses). This option also enables searching the scandef with a hash file which greatly increases performance.

## eCMD Command Line Interface

```
Parameters:
-----
ChipSelect      Chip to put data to.

RingName        Specifies chip ring to operate on.  For a list of available
                rings, use the ecmdquery command.

                Ex:  ecmdquery rings memctrl
                    ecmdquery rings pu

LatchName       Desired latches to put in the ring.

Start           [opt] Offset at which to begin writing data.  Also specify Numbits.

Numbits         [opt] Number of bits to insert.  If not specified, start = 0 and
                numbits is calculated from the length of the Data string.

Data            Data to be written to the register specified.
                Format specified by -i<format>

-exact          [opt] No pattern matching. Instead, search for exact latch name.
(default)

-partial        [opt] Use pattern matching to find latch name, can be considerably
slower.

-i<format>[opt] Specifies the format type of input data : default 'xl'
                Run 'ecmdquery formats' to view available formats

-b<mod>[opt]    Bit modifier to apply to current ring data.
                Run 'ecmdquery formats' to view available modifiers

-k# [optional]  Specify which cage to act on (0 is default). Specify -kall
                to act on all cages.

-n# [optional]  Specify which node to act on (0 is default). Specify -nall
                to act on all nodes.

-s# [optional]  Specify which slot to act on (0 is default). Specify -sall
                to act on all slots.

-p# [optional]  Specify which chip position to act on (0 is default). Specify
                -pall to act on all chips.

-c# [optional]  Specify which processor core to act on (0 is default). Specify
                -call to act on all cores.

-----

Example:        putlatch pu cp_fpu GCP.PFPU.FP_UNIT0.FPA_LATCH -ix feed
```

### Examples:

## 3.2.15 putpattern

### Syntax:

Syntax: putpattern <ChipSelect> <RingType> <Data> [-i<format>]

## eCMD Command Line Interface

```
                                [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD:          Core Common Function

Function:      Puts a repeated pattern to the entire specified chip ring.

Parameters:
-----
ChipSelect     Specifies the chip to operate on.

RingName       Specifies chip ring to operate on. For a list of available
               rings, use the ecmdquery command.

               Ex:  ecmdquery rings memctrl
                   ecmdquery rings pu

Data           32bit pattern to write.
               Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xr'
               Run 'ecmdquery formats' to view available formats

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
               to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
               to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
               to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
               -pall to act on all chips.

-c#  [optional] Specify which processor core to act on (0 is default). Specify
               -call to act on all cores.

-----

Example:       putpattern pu int FEEDBEEF -p0,1
```

### Examples:

## 3.2.16 putscom

### Syntax:

```
Syntax: putscom <ChipSelect> <ScanCommAddr> [<Start> <Numbits>] <Data> [-i<format>]
               [-b<modifier>] [-k#] [-n#] [-s#] [-p#] [-c#]
```

```
ECMD:          Core Common Function

Function:      Write the specified data to the specified chip using scancom.

Parameters:
-----
ChipSelect     Chip to put scancom data to.

ScanCommAddr   Address in right aligned hex.
```

## eCMD Command Line Interface

Start [opt] Starting bit position within the scom. Specify with numbits.

Numbits [opt] Number of bits to insert. Specify with Start. If Start and Numbits are not specified, start = 0 and numbits is calculated from length of data string, rest of Scom register is padded with zeros.

Data Data to insert into Scom Register.  
Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'  
Run 'ecmdquery formats' to view available formats

-b<mod>[opt] Bit modifier to apply to current ring data.  
Run 'ecmdquery formats' to view available modifiers

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----

Examples: > putscom pu 600000 deadbeef80000000 -p0,1 -c1  
> putscom memctrl 2010 001001010110 -ib  
> putscom 13 40320 0000800000 -bor -p12

## Examples:

### 3.2.17 putspy

#### Syntax:

Syntax: putspy <ChipSelect> <SpyName> [<Start> <Numbits>] <Data> [-i<format>]  
[-b<modifier>] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Write data to a spy. Works with idial, edial and alias spy's.

Parameters:

-----  
ChipSelect Chip to write data to.

SpyName Desired spy name, (case insensitive)

Start [opt] Starting bit position within the spy. Specify with numbits.  
Only valid with non-enumerated spy's

## eCMD Command Line Interface

Numbits [opt] Number of bits to insert. Specify with Start. If Start and Numbits are not specified, start = 0 and numbits is calculated from length of data string.  
Only valid with non-enumerated spy's

Data Data to put into spy, either raw data or enum name.  
Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data  
Default format for non-enumerated spys : 'xl'  
Default format for enumerated spys : 'enum'  
Run 'ecmdquery formats' to view available formats

-b<mod>[opt] Bit modifier to apply to current ring data.  
Run 'ecmdquery formats' to view available modifiers

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----

Examples:        putspy pu MYALIAS -ixr feedbeeffeedbeef  
                 putspy pu EVERYOTHER 16 4 -ib 1010  
                 putspy pu MYEDIAL ENUMVALUE -ienum

## 3.2.18 ringcache

### Syntax:

Syntax: ringcache enable|disable|flush|query

ECMD:            Core Common Function

Function:        Modifies state of internal caching of reads/writes of scan rings

Parameters:

-----

enable	Enables ring caching
disable	Disables ring caching
flush	Flushes ring cache and leaves enabled
query	Displays state of ring cache

-----

## eCMD Command Line Interface

Example:           ringcache enable  
                  ringcache disable

### 3.2.19 sendcmd

#### Syntax:

Syntax: sendcmd <ChipSelect> <ScanInstrCode> <ScanInstrMod> [-v] [-k#] [-n#] [-s#] [-p#]

ECMD:           Core Common Function

Function:       Send a JTAG Instruction to the chip and display  
                 instruction status from previous command

#### Parameters:

-----  
ChipSelect      Chip to send ScanInstrCode to.

ScanInstrCode   Scan instruction code to be sent (in hex).

ScanInstrMod    Scan instruction modifier (for ACCESS/CFAM).

-v   [optional] Verbose mode. Displays the instruction  
                 status in an easy-to-read format.

-k#   [optional] Specify which cage to act on (0 is default). Specify -kall  
                 to act on all cages.

-s#   [optional] Specify which slot to act on (0 is default). Specify -sall  
                 to act on all slots.

-n#   [optional] Specify which node to act on (0 is default). Specify -nall  
                 to act on all nodes.

-p#   [optional] Specify which chip position to act on (0 is default). Specify  
                 -pall to act on all chips.

-----  
Notes:           Only valid with JTAG attached chips

Example:         sendcmd pu 12 C00008 -p0,1

#### Examples:

## 3.3 Processor Functions

### 3.3.1 getfpr

#### Syntax:

Syntax: getfpr <FprStartNum> [<numEntries>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]  
[-o<format>]

ECMD: Core Common Function

Function: Gets Processor Architected FPR (Floating Point Register).

Parameters:

```
-----
FprNum          Fpr Entry to read (Decimal)
numEntries      Specifies number of entries to get from starting entry (Decimal)
-o<format>[opt] Specifies the format type of the output : default 'xl'
                  Run 'ecmdquery formats' to view available formats
-k# [optional]  Specify which cage to act on (0 is default). Specify -kall
                  to act on all cages.
-n# [optional]  Specify which node to act on (0 is default). Specify -nall
                  to act on all nodes.
-s# [optional]  Specify which slot to act on (0 is default). Specify -sall
                  to act on all slots.
-p# [optional]  Specify which chip position to act on (0 is default). Specify
                  -pall to act on all chips.
-c# [optional]  Specify which processor core to act on (0 is default). Specify
                  -call to act on all cores.
-t# [optional]  Specify which processor thread to act on (0 is default). Specify
                  -tall to act on all threads, -talive to act on all alive threads.
-----
```

Examples: > getfpr 6 -p0,1  
> getfpr 0 32 -p10, -t1 -c1

### 3.3.2 getgpr

#### Syntax:

Syntax: getgpr <GprStartNum> [<numEntries>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]  
[-o<format>]

ECMD: Core Common Function

Function: Gets Processor GPR (General Purpose Register).



## eCMD Command Line Interface

### Parameters:

-----

GprNum            Gpr Entry to read (Decimal)

numEntries        Specifies number of entries to get from starting entry (Decimal)

-o<format>[opt] Specifies the format type of the output : default 'xl'  
Run 'ecmdquery formats' to view available formats

-k#   [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n#   [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s#   [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p#   [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c#   [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-t#   [optional] Specify which processor thread to act on (0 is default). Specify  
-tall to act on all threads, -talive to act on all alive threads.

-----

Examples:            > getgpr 6 -p0,1  
                     > getgpr 0 32 -p10, -t1 -c1

## 3.3.3 getspr

### Syntax:

Syntax: getspr <SprName> [<SprName> ...] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]  
                             [-o<format>]

ECMD:                Core Common Function

Function:            Gets Processor SPR (Special Purpose Register).

### Parameters:

-----

SprName            Name of SPR to fetch, multiple SPR's can be listed

-o<format>[opt] Specifies the format type of the output : default 'xl'  
Run 'ecmdquery formats' to view available formats

-k#   [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n#   [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s#   [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

## eCMD Command Line Interface

-p# [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-t# [optional] Specify which processor thread to act on (0 is default). Specify  
-tall to act on all threads, -talive to act on all alive threads.

-----

Examples:           > getspr pu nia msr -p0,1

### 3.3.4 putfpr

#### Syntax:

Syntax: putfpr <FprNum> <Data> [-i<format>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]

ECMD:               Core Common Function

Function:           Write the specified data to a Processor FPR  
                    (Floating Point Register)

Parameters:

-----

FprNum              Fpr Entry to write (Decimal)

Data                Data to insert into FPR.  
                    Format specified by -i<format>

-i<format>[opt]     Specifies the format type of input data : default 'xl'  
                    Run 'ecmdquery formats' to view available formats

-k# [optional]     Specify which cage to act on (0 is default). Specify -kall  
                    to act on all cages.

-n# [optional]     Specify which node to act on (0 is default). Specify -nall  
                    to act on all nodes.

-s# [optional]     Specify which slot to act on (0 is default). Specify -sall  
                    to act on all slots.

-p# [optional]     Specify which chip position to act on (0 is default). Specify  
                    -pall to act on all chips.

-c# [optional]     Specify which processor core to act on (0 is default). Specify  
                    -call to act on all cores.

-t# [optional]     Specify which processor thread to act on (0 is default). Specify  
                    -tall to act on all threads, -talive to act on all alive threads.

-----

Examples:           > putfpr 10 deadbeef80000000 -p0,1 -c1

### 3.3.5 putgpr

#### Syntax:

Syntax: putgpr <GprNum> <Data> [-i<format>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]

ECMD: Core Common Function

Function: Write the specified data to a Processor GPR  
(General Purpose Register)

Parameters:

```
-----
GprNum      Gpr Entry to write (Decimal)

Data         Data to insert into GPR.
             Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'
               Run 'ecmdquery formats' to view available formats

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
               to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
               to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
               to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
               -pall to act on all chips.

-c#  [optional] Specify which processor core to act on (0 is default). Specify
               -call to act on all cores.

-t#  [optional] Specify which processor thread to act on (0 is default). Specify
               -tall to act on all threads, -talive to act on all alive threads.
-----
```

Examples: > putgpr 10 deadbeef80000000 -p0,1 -c1

### 3.3.6 putspr

#### Syntax:

Syntax: putspr <SprName> [<Start> <Numbits>] <Data> [-i<format>]  
[-b<modifier>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]

ECMD: Core Common Function

Function: Write the specified data to a Processor SPR  
(Special Purpose Register).

Parameters:

## eCMD Command Line Interface

```
-----
SprName      Name of SPR to write

Start        [opt] Starting bit position.  Specify with numbits.

Numbits      [opt] Number of bits to insert. Specify with Start. If Start and Numbits
               are not specified, start = 0 and numbits is calculated from
               length of data string, rest of register is padded with zeros.

Data         Data to insert into Register.
               Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'
               Run 'ecmdquery formats' to view available formats

-b<mod>[opt]   Bit modifier to apply to current ring data.
               Run 'ecmdquery formats' to view available modifiers

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
               to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
               to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
               to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
               -pall to act on all chips.

-c#  [optional] Specify which processor core to act on (0 is default). Specify
               -call to act on all cores.

-t#  [optional] Specify which processor thread to act on (0 is default). Specify
               -tall to act on all threads, -talive to act on all alive threads.
-----

Examples:      > putspr nia deadbeef80000000 -p0,1 -c1 -t1
```

## 3.4 Memory Display/Alter Functions

### 3.4.1 cacheflush

#### Syntax:

Syntax: `cacheflush <ChipSelect> <CacheType> [-k#] [-n#] [-s#] [-p#] [-c#]`

ECMD: Core Common Function

Function: Flush a particular cache based on the cache type specified

Parameters:

-----  
ChipSelect Chip to flush the cache on.

CacheType One of L1I / L1D / L2 / L3 / L4

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which core to act on (0 is default). Specify -call to act on all cores.

-----  
Example: `cacheflush pu L1I -p0,1 -call`

### 3.4.2 getmemdma

#### Syntax:

Syntax: `getmemdma <MemAddress> <NumBytes> [-k#] [-n#]  
[-o<format> | -f[d|b]<filename>]`

ECMD: Core Common Function

Function: Display the contents of mainstore using either DMA's or PSI

Parameters:

-----  
MemAddress 64 Bit address to read from (Hex-Right)

NumBytes Number of bytes to fetch (Decimal).

-o<format>[opt] Specifies the format type of the output : default 'mem'  
Not valid with -f option.  
Run 'ecmdquery formats' to view available formats

-fd <filename> Specify full path and filename to file in D-Card format to write

## eCMD Command Line Interface

```
data from system
Not valid with -o option.

-fb <filename> Specify full path and filename to binary file to write data from
system
Not valid with -o option.
Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format

-k# [optional] Specify which cage to act on (0 is default). Specify -kall
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
to act on all nodes.
```

```
-----

Examples:      > getmemdma 1000 128 -p0
                > getmemdma 1000 128 -fb datafile
```

### 3.4.3 getmemmemctrl

#### Syntax:

```
Syntax: getmemmemctrl <MemAddress> <NumBytes> [-k#] [-n#] [-s#] [-p#]
                                             [-o<format> | -f[d|b]<filename>]
```

ECMD: Core Common Function

Function: Display the contents of mainstore using the Memory Controller.  
NOTE : This operation typically is not cache coherent.

#### Parameters:

```
-----
MemAddress      64 Bit address to read from (Hex-Right)
```

```
NumBytes        Number of bytes to fetch (Decimal).
```

```
-o<format>[opt] Specifies the format type of the output : default 'mem'
Not valid with -f option.
Run 'ecmdquery formats' to view available formats
```

```
-fd <filename> Specify full path and filename to file in D-Card format to write
data from system
Not valid with -o option.
```

```
-fb <filename> Specify full path and filename to binary file to write data from
system
Not valid with -o option.
Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format
```

```
-k# [optional] Specify which cage to act on (0 is default). Specify -kall
to act on all cages.
```

```
-n# [optional] Specify which node to act on (0 is default). Specify -nall
to act on all nodes.
```

```
-s# [optional] Specify which slot to act on (0 is default). Specify -sall
to act on all slots.
```

## eCMD Command Line Interface

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----

Examples:        > getmemmemctrl 1000 128 -p0  
                 > getmemmemctrl 1000 128 -fb datafile

### 3.4.4 getmemproc

#### Syntax:

Syntax: getmemproc <MemAddress> <NumBytes> [-k#] [-n#] [-s#] [-p#]  
   [-o<format> | -f[d|b]<filename>]

ECMD:            Core Common Function

Function:        Display the contents of mainstore using the processor

#### Parameters:

-----

MemAddress      64 Bit address to read from (Hex-Right)

NumBytes        Number of bytes to fetch (Decimal).

-o<format>[opt] Specifies the format type of the output : default 'mem'  
                 Not valid with -f option.  
                 Run 'ecmdquery formats' to view available formats

-fd <filename>   Specify full path and filename to file in D-Card format to write  
                 data from system  
                 Not valid with -o option.

-fb <filename>   Specify full path and filename to binary file to write data from  
                 system  
                 Not valid with -o option.  
                 Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_BINARY\_DATA format

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
                 to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
                 to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
                 to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
                 -pall to act on all chips.

-----

Examples:        > getmemproc 1000 128 -p0  
                 > getmemproc 1000 128 -fb datafile

### 3.4.5 putmemdma

#### Syntax:

```
Syntax: putmemdma <MemAddress> <Data> [-i<format>]          [-k#] [-n#] [-s#] [-p#]
        putmemdma <MemAddress> -fb<filename>                [-k#] [-n#] [-s#] [-p#]
        putmemdma -fd<filename>                             [-k#] [-n#] [-s#] [-p#]
```

ECMD: Core Common Function

Function: Write the specified data to mainstore using either DMA's or PSI

Parameters:

---

MemAddress 64 Bit address to write to (Hex-Right). Not valid with -fd option

-fd <filename> Specify full path and filename to file in D-Card format to load to system

-fb <filename> Specify full path and filename to binary file to load to system  
Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_BINARY\_DATA format

Data Data to write into mainstore. Not valid with -f option  
Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'  
Not valid with -f option  
Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

---

Examples: > putmemdma 10000 deadbeef80000000 -p0,1  
> putmemdma 10000 -fb datafile  
> putmemdma -fd dcardfile

### 3.4.6 putmemmemctrl

#### Syntax:

```
Syntax: putmemmemctrl <MemAddress> <Data> [-i<format>]      [-k#] [-n#] [-s#] [-p#]
        putmemmemctrl <MemAddress> -fb<filename>             [-k#] [-n#] [-s#] [-p#]
        putmemmemctrl -fd<filename>                         [-k#] [-n#] [-s#] [-p#]
```

ECMD: Core Common Function

Function: Write the specified data to mainstore using the Memory Controller  
NOTE : This operation typically is not cache coherent.

Parameters:

---

MemAddress 64 Bit address to write to (Hex-Right). Not valid with -fd option



## eCMD Command Line Interface

-fd <filename> Specify full path and filename to file in D-Card format to load to system

-fb <filename> Specify full path and filename to binary file to load to system  
Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_BINARY\_DATA format

Data Data to write into mainstore. Not valid with -f option  
Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'  
Not valid with -f option  
Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----

Examples: > putmemmemctrl 10000 deadbeef80000000 -p0,1  
> putmemmemctrl 10000 -fb datafile  
> putmemmemctrl -fd dcardfile

### 3.4.7 putmemproc

#### Syntax:

Syntax: putmemproc <MemAddress> <Data> [-i<format>] [-k#] [-n#] [-s#] [-p#]  
putmemproc <MemAddress> -fb<filename> [-k#] [-n#] [-s#] [-p#]  
putmemproc -fd<filename> [-k#] [-n#] [-s#] [-p#]

ECMD: Core Common Function

Function: Write the specified data to mainstore using the Processor

#### Parameters:

-----

MemAddress 64 Bit address to write to (Hex-Right). Not valid with -fd option

-fd <filename> Specify full path and filename to file in D-Card format to load to system

-fb <filename> Specify full path and filename to binary file to load to system  
Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_BINARY\_DATA format

Data Data to write into mainstore. Not valid with -f option  
Format specified by -i<format>

-i<format>[opt] Specifies the format type of input data : default 'xl'  
Not valid with -f option

## eCMD Command Line Interface

Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----

Examples:

- > putmemproc 10000 deadbeef80000000 -p0,1
- > putmemproc 10000 -fb datafile
- > putmemproc -fd dcardfile

## 3.5 Miscellaneous Commands

### 3.5.1 Deconfig

#### Syntax:

Syntax: deconfig [<ChipSelect>] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

Function: Deconfigure a target from the system. Can deconfigure cages, nodes, slots, chip positions and cores.

NOTE : It typically requires rerunning isteps to fully disable the target

Parameters:

-----  
ChipSelect[opt] Chip name to deconfigure

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----  
Examples: > deconfig pu -k0 -n1 -p3  
> deconfig -k1 -n2

### 3.5.2 ecmdquery

#### Syntax:

Syntax: ecmdquery <Mode> [Mode Options]

ECMD: Core Common Function

Function: Query information from eCMD

Parameters:

-----  
Mode Query type to perform

Mode Values

rings ChipSelect [ringname] [-k#] [-n#] [-s#] [-p#]  
- Display all rings available (or the selected ring) for chip

scoms ChipSelect [scomaddr] [-k#] [-n#] [-s#] [-p#]

## eCMD Command Line Interface

```
chip
    - Display all scoms available (or the selected scom) for chip

arrays ChipSelect [arrayname] [-k#] [-n#] [-s#] [-p#]
    - Display all arrays available (or the selected array) for

for chip

tracearrays ChipSelect [tracearrayname] [-k#] [-n#] [-s#] [-p#]
    - Display all tr arrays available (or the selected tr array)

spys ChipSelect [spyname] [-k#] [-n#] [-s#] [-p#]
    - Display all spys available (or the selected spy) for chip

formats
    - Display help text for all available input/output formats

chips [-ep] [-dp | -dc | -dt]
    - Display all the chips in the system
    - Use '-ep' to display in an easier to parse format
    - Use -dp to specify position depth display
    - Use -dc to specify core depth display
    - Use -dt to specify trace depth display

chipinfo ChipSelect [-k#] [-n#] [-s#] [-p#]
    - Display info about a particular chip (ex. EC level)

version
    - Display version info about the eCMD Instance you are running
```

-----

```
Example:    ecmdquery rings pu -p0,1
            ecmdquery formats
```

## Examples:

```

> ecmdquery version
=====
Dll Type      : Cronus
Dll Product   : Unknown
Dll Environment : Hardware
Dll Build Date  : Nov 24 2003 14:19:14
Dll Capi Version : .1
=====
ecmd.exe ecmdquery version

> ecmdquery rings test

Available rings for test      k0:n0:s0:p00      ec 0:
Ring Names                    Address      Length      Mask Chkable BroadSide ClockState
-----
idreg                        0x000100      32          N    N          N      UNKNOWN
scancom                     0x000040      64          N    N          N      UNKNOWN
scancomprint                0x000040      64          N    N          N      UNKNOWN
scancomstat                 0x000080      32          N    N          N      UNKNOWN
bypass32                    0x000010      32          N    N          N      UNKNOWN
access_ec                   0x000200      32          N    N          N      UNKNOWN
crcreg                      0x000020      32          N    N          N      UNKNOWN
gp1                          0x001000      32          N    N          N      UNKNOWN
gp2                          0x002000      32          N    N          N      UNKNOWN
gp3                          0x004000      32          N    N          N      UNKNOWN
testring                    0x800003      128         N    Y          N      UNKNOWN
sgxbs                       0x800009      573         N    Y          N      UNKNOWN
ecmd.exe ecmdquery rings test

```

### 3.5.3 getconfig

#### Syntax:

Syntax: getconfig [<ChipSelect>] <ConfigName> [-o<format>] [-k#] [-n#] [-s#] [-p#] [-c#]  
 [-dk | -dn | -ds | -dp | -dc]

ECMD: Core Common Function

Function: Read the specified configuration variable and display to screen

Parameters:

-----  
 ChipSelect[opt] Chip to read data from.

ConfigName Name of configuration variable to fetch

-o<format>[opt] Output Format : default 'xl'. Only valid with numeric data  
 Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
 to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
 to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
 to act on all slots.

## eCMD Command Line Interface

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-dk [optional] Specify cage depth to loop on. When no ChipSelect specified -dk is default

-dn [optional] Specify node depth to loop on.

-ds [optional] Specify slot depth to loop on.

-dp [optional] Specify pos depth to loop on. When ChipSelect specified -dp is default

-dc [optional] Specify core depth to loop on.

-----

Examples:

```
> getconfig SIM_BROADSIDE_MODE -dk
SIM_BROADSIDE_MODE = none
> getconfig pu PLL_TUNE -dp
pu          k0:n0:s0:p00
PLL_TUNE = 3
```

## 3.5.4 makespsystemcall

### Syntax:

Syntax: makespsystemcall "<Command>" [-k#] [-n#]

ECMD: Core Common Function

Function: Run a command on the SE/SP and print the output to stdout.

Parameters:

-----

Command Command to run on the SE/SP. Place command in ""'s to avoid it being parsed.

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-----

Examples:

```
> makespsystemcall "ls"
```

## 3.5.5 reconfig

### Syntax:

Syntax: reconfig [<ChipSelect>] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: Core Common Function

## eCMD Command Line Interface

Function: Reconfigure a previously deconfigured target from the system.  
Can reconfigure cages, nodes, slots, chip positions and cores.

NOTE : It typically requires rerunning isteps to fully enable the target

### Parameters:

-----  
ChipSelect[opt] Chip name to reconfigure

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----  
Examples: > reconfig pu -k0 -n1 -p3  
> reconfig -k1 -n2

## 3.5.6 setconfig

### Syntax:

Syntax: setconfig [<ChipSelect>] <ConfigName> <Value> [-i<format>] [-k#] [-n#] [-s#]  
[-p#] [-c#]  
[-dk | -dn | -ds | -dp | -dc]

ECMD: Core Common Function

Function: Write the specified configuration variable

### Parameters:

-----  
ChipSelect[opt] Chip to write data to.

ConfigName Name of configuration variable to write

Value Value to set configuration variable to

-i<format>[opt] Output Format : default is ascii.  
Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall

## eCMD Command Line Interface

to act on all slots.

- p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.
- c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.
- dk [optional] Specify cage depth to loop on. When no ChipSelect specified -dk is default
- dn [optional] Specify node depth to loop on.
- ds [optional] Specify slot depth to loop on.
- dp [optional] Specify pos depth to loop on. When ChipSelect specified -dp is default
- dc [optional] Specify core depth to loop on.

-----

Examples:        > setconfig SIM\_BROADSIDE\_MODE scan -dk  
                 > setconfig pu PLL\_TUNE 3 -id -dp



## 3.6 System Functions

### 3.6.1 istep

#### Syntax:

```
Syntax: istep
        istep <StepName1>[,<StepName2> ...]
        istep <StartStep>..<EndStep>
        istep -s<StepNumbers>
        istep -i<SkipSteps>
```

ECMD: Core Common Function

Function: Run iSteps on the system. Default is to run all isteps

NOTE: Some plugins support 'istep list' to display all available steps

#### Parameters:

```
-----
<Stepname>      Comma seperated list of names of steps to run
                  (ex 'proc_cfaminit')

<StartStep>     Start Step to run

<EndStep>       Ending Step to run

<StepNum>       Step numbers to run (same format as -p arg) (ex -s0,1..5,10)

<SkipSteps>     Step numbers to NOT run (same format as -p arg) (ex -i0,1..5,10)
-----
```

```
Examples:      istep
                istep proc_cfaminit,proc_scaninit
                istep proc_cfaminit..proc_scominit
                istep -s0,1..5,10,20
                istep -i2,3
                istep list (NOTE: only supported by some plugins)
```

### 3.6.2 setclockspeed

#### Syntax:

```
Syntax: setclockspeed <ClockType> <Speed> [-steer]      [-k#]
```

ECMD: Core Common Function

Function: Change the speed of a clock in the system.

#### Parameters:

```
-----
ClockType      Clock type to program in system:
                - pu_refclock      <-- Processor reference clock
                - memctrl_refclock <-- Memory controller reference clock

Speed          New speed value to program can be specified in mhz (100mhz) or
                in microseconds (10us)
-----
```

## eCMD Command Line Interface

```
-steer [opt]    Steer to new value.  Default is to move in one operation.

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
                    to act on all cages.
```

```
Examples:      > setclockspeed pu_refclock 120mhz
               > setclockspeed memctrl_refclock 10us
```

### 3.6.3 startclocks

### Syntax:

```
Syntax: startclocks                                [-force] [-k#] [-n#] [-s#]
        startclocks -domain <ConvenienceClockDomain> [-force] [-k#] [-n#] [-s#]
        startclocks <ChipSelect> [<ClockDomain>]    [-force] [-k#] [-n#] [-s#] [-p#]
[-c#]
```

ECMD: Core Common Function

Function: Start clocks on a particular domain/chip or the entire system

NOTE : This command typically does not start clocks in a way that the chip will be functional to run instructions. To do that use the `isteps` command to initialize the system.

Parameters:

ChipSelect	Chip name to start clocks on
------------	------------------------------

ClockDomain	Clock domain to start on chip target. Must specify with ChipSelect Names are documented in the scandef for the targetted chip.
-------------	--

`-domain` Specifies we are using a convenience clock domain.

ConvenienceClockDomain The convenience clock domains are documented in the eCMD System specific document for your system type. Must be specified with -domain

-force [opt] Force clocks on regardless of current state

**-k#** [optional] Specify which cage to act on (0 is default). Specify **-kall** to act on all cages.

**-n#** [optional] Specify which node to act on (0 is default). Specify **-nall** to act on all nodes.

**-s#** [optional] Specify which slot to act on (0 is default). Specify **-sall** to act on all slots.

`-p#` [optional] Specify which chip position to act on (0 is default). Specify `-pall` to act on all chips.

`-c#` [optional] Specify which processor core to act on (0 is default). Specify `-call` to act on all cores.

Examples:        > startclocks pu -k0 -n1 -p3

```
> startclocks pu FBC_DOMAIN -k1 -n2
> startclocks
> startclocks -domain ALL_PU_CHIPS
```

## 3.6.4 stopclocks

### Syntax:

```
Syntax: stopclocks [-force] [-k#] [-n#] [-s#]
stopclocks -domain <ConvenienceClockDomain> [-force] [-k#] [-n#] [-s#]
stopclocks <ChipSelect> [<ClockDomain>] [-force] [-k#] [-n#] [-s#] [-p#]
[-c#]
```

ECMD: Core Common Function

Function: Stop clocks on a particular domain/chip or the whole system.

Parameters:

```
-----
ChipSelect      Chip name to stop clocks on

ClockDomain     Clock domain to stop on chip target. Must specify with ChipSelect
Names are documented in the scandef for the targetted chip.

-domain         Specifies we are using a convenience clock domain.

ConvenienceClockDomain The convenience clock domains are documented in the eCMD
System specific document for your system type. Must be specified
with -domain

-force          [opt] Force clocks off regardless of current state

-k#             [optional] Specify which cage to act on (0 is default). Specify -kall
to act on all cages.

-n#             [optional] Specify which node to act on (0 is default). Specify -nall
to act on all nodes.

-s#             [optional] Specify which slot to act on (0 is default). Specify -sall
to act on all slots.

-p#             [optional] Specify which chip position to act on (0 is default). Specify
-pall to act on all chips.

-c#             [optional] Specify which processor core to act on (0 is default). Specify
-call to act on all cores.
-----
```

```
Examples: > stopclocks pu -k0 -n1 -p3
> stopclocks pu FBC_DOMAIN -k1 -n2
> stopclocks
> stopclocks -domain ALL_PU_CHIPS
```

## 3.7 GPIO Commands

### 3.7.1 getgpiolatch

#### Syntax:

Syntax: getgpiolatch <ChipSelect> <EngineId> <Pin> <Mode>  
 [-k#] [-n#] [-s#] [-p#] [-o<format>]

ECMD: Core Common Function

Function: Read data from the specified latch.

Parameters:

```
-----
ChipSelect    Specifies the chip to operate on.

EngineId      Engine number to operate on in decimal

Pin           Pin number to operate on in decimal

Mode          Mode to use on pin.
              Values : IN(Input) OD(Open Drain) OS(Open Source) PP(Push Pull)

-o<format>[opt] Specifies the format type of the output : default 'b'
              Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall
              to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
              to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall
              to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify
              -pall to act on all chips.
-----
```

Examples: > getgpiolatch fsp 0 1 OD

### 3.7.2 getgpiopin

#### Syntax:

Syntax: getgpiopin <ChipSelect> <EngineId> <Pin> [-o<format>] [-k#] [-n#] [-s#] [-p#]  
 getgpiopin <ChipSelect> <EngineId> -mask <MaskValue> [-o<format>] [-i<format>]  
 [-k#] [-n#] [-s#] [-p#]

ECMD: Core Common Function

Function: Read the value of the pin/pins specified

Parameters:

```
-----
ChipSelect    Specifies the chip to operate on.
```

## eCMD Command Line Interface

EngineId            Engine number to operate on in decimal

Pin                Pin number to operate on in decimal

-mask        [opt] To read from multiple pins specify a mask value of pins to read  
Binary input will be treated left-aligned

-o<format>[opt] Specifies the format type of the output : default 'b'  
Run 'ecmdquery formats' to view available formats

-i<format>[opt] Specifies the format type of the input : default 'xl'  
Run 'ecmdquery formats' to view available formats

-k#    [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n#    [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s#    [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p#    [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-----

Examples:            > getgpiopin fsp 0 1  
                      > getgpiopin fsp 0 -mask FFFE0FF7

## 3.7.3 gpioconfig

### Syntax:

Syntax: gpioconfig <ChipSelect> <EngineId> <Pin> <Mode> [-k#] [-n#] [-s#] [-p#]

ECMD:                Core Common Function

Function:            Configures the specified GPIO pin to the specified mode.

Parameters:

-----

ChipSelect           Specifies the chip to operate on.

EngineId             Engine number to operate on in decimal

Pin                  Pin number to operate on in decimal

Mode                 Mode to set on pin.  
Values : IN(Input) OD(Open Drain) OS(Open Source) PP(Push Pull)

-k#    [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n#    [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s#    [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p#    [optional] Specify which chip position to act on (0 is default). Specify

-poll to act on all chips.

Examples:            `> gpioconfig fsp 0 1 IN`

### 3.7.4 putgpiolatch

### Syntax:

```
Syntax: putgpiolatch <ChipSelect> <EngineId> <Pin> <Mode> <Data>
        [-k#] [-n#] [-s#] [-p#] [-i<format>]
putgpiolatch <ChipSelect> <EngineId> -mask <MaskValue> <Mode> <Data>
        [-k#] [-n#] [-s#] [-p#] [-i<format>]
```

ECMD: Core Common Function

Function: Write data to specified latch/latches.

Parameters:

ChipSelect	Specifies the chip to operate on.
------------	-----------------------------------

EngineId	Engine number to operate on in decimal
----------	--

Pin	Pin number to operate on in decimal
-----	-------------------------------------

Data	Data to write to latch(s)
------	---------------------------

Mode                    Mode to use on pin.  
Values : IN(Input) OD(Open Drain) OS(Open Source) PP(Push Pull)

```
-mask      [opt] To write to multiple pins specify a mask value of latches to write
            Binary input will be treated left-aligned
```

**-i<format>[opt]** Specifies the format type of the input : default 'b'  
Run 'ecmdquery formats' to view available formats

**-k#** [optional] Specify which cage to act on (0 is default). Specify **-kall** to act on all cages.

**-n#** [optional] Specify which node to act on (0 is default). Specify **-nall** to act on all nodes.

**-s#** [optional] Specify which slot to act on (0 is default). Specify **-sall** to act on all slots.

`-p#` [optional] Specify which chip position to act on (0 is default). Specify `-pall` to act on all chips.

```
Examples:      > putgpiolatch fsp 0 1 OD 1
               > putgpiolatch fsp 0 1 -mask FFEC0FF7 PP 01280012 -ix
```

## 3.8 I2C Commands

### 3.8.1 geti2c

#### Syntax:

Syntax:

```
WITHOUT OFFSET:
geti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> <Bytes>
      [-k#] [-n#] [-s#] [-p#] [-o<format> | -f<filename>] [-busspeed <speed>]
WITH OFFSET:
geti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> <Bytes> <Offset> <FieldSize>
      [-k#] [-n#] [-s#] [-p#] [-o<format> | -f<filename>] [-busspeed <speed>]
```

ECMD: Core Common Function

Function: Read I2C data from the specified engine/port/device for the specified number of bytes.

Parameters:

```
-----
ChipSelect    Specifies the chip to operate on.

EngineId      Engine number to operate on in decimal

Port          Engine port number to operate on in decimal

SlaveAddress  Slave Device Address in hex

Bytes         Bytes to read from device

Offset        Offset into the slave device

FieldSize     Byte width of the offset

-busspeed [opt] Specifies the bus speed to run i2c in khz : default 400
                Valid values are 400, 100, 50

-o<format>[opt] Specifies the format type of the output : default 'xl'
                Run 'ecmdquery formats' to view available formats

-f<filename>[o] Specifies the filename that the data should be written to
                Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format
                Incase multiple chip positions are specified the target string
will          be postfixed. E.g. filename "datafile" becomes "datafile.k0n0s0p0"

-k# [optional] Specify which cage to act on (0 is default). Specify -kall
                to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall
                to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall
                to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify
                -pall to act on all chips.
-----
```

Examples: > geti2c fsp 0 1 A0 100 -busspeed 100

```
> geti2c fsp 0 1 A0 100 -foutput
> geti2c fsp 0 1 A0 100 2 10
```

## 3.8.2 i2creset

### Syntax:

Syntax: i2creset <ChipSelect> <EngineId> <Port> [-k#] [-n#] [-s#] [-p#]

ECMD: Core Common Function

Function: Resets the specified engine port

Parameters:

-----  
ChipSelect Specifies the chip to operate on.

EngineId Engine number to operate on in decimal

Port Engine port number to operate on in decimal

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----  
Examples: > i2creset fsp 0 1

## 3.8.3 puti2c

### Syntax:

Syntax:

WITHOUT OFFSET:

```
puti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> <Data>
      [-k#] [-n#] [-s#] [-p#] [-i<format>] [-busspeed <speed>]
puti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> -f<filename>
      [-k#] [-n#] [-s#] [-p#] [-busspeed <speed>]
```

WITH OFFSET:

```
puti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> <Data> <Offset> <FieldSize>
      [-k#] [-n#] [-s#] [-p#] [-i<format>] [-busspeed <speed>]
puti2c <ChipSelect> <EngineId> <Port> <SlaveAddress> -f<filename> <Offset>
<FieldSize>
      [-k#] [-n#] [-s#] [-p#] [-busspeed <speed>]
```

ECMD: Core Common Function

Function: Write I2C data to the specified engine/port/device.



## eCMD Command Line Interface

### Parameters:

```
-----
ChipSelect      Specifies the chip to operate on.

EngineId        Engine number to operate on in decimal

Port            Engine port number to operate on in decimal

SlaveAddress    Slave Device Address in hex

Data            Data to write to device

Offset          Offset into the slave device

FieldSize       Byte width of the offset

-busspeed [opt] Specifies the bus speed to run i2c in khz : default 400
                Valid values are 400, 100, 50

-i<format>[opt] Specifies the format type of the input : default 'xl'
                Run 'ecmdquery formats' to view available formats

-f<filename>[o] Specifies the filename that the data should be read from
                Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format

-k# [optional]  Specify which cage to act on (0 is default). Specify -kall
                to act on all cages.

-n# [optional]  Specify which node to act on (0 is default). Specify -nall
                to act on all nodes.

-s# [optional]  Specify which slot to act on (0 is default). Specify -sall
                to act on all slots.

-p# [optional]  Specify which chip position to act on (0 is default). Specify
                -pall to act on all chips.
-----
```

```
Examples:      > puti2c fsp 0 1 A0 FFAABB -busspeed 100
                > puti2c fsp 0 1 A0 -fdata
                > puti2c fsp 0 1 A0 2 10 -fdata
```

## 3.9 VPD Commands

### 3.9.1 getvpdimage

#### Syntax:

```
Syntax: getvpdimage <ChipSelect> <NumBytes> -o<format>          [-k#] [-n#] [-s#] [-p#]
        getvpdimage <ChipSelect> <NumBytes> -fb<filename>      [-k#] [-n#] [-s#] [-p#]
```

ECMD: Core Common Function

Function: Display the contents of a module VPD image.

Parameters:

```
-----
ChipSelect      Chip to get the VPD image from

-o<format>[opt] Specifies the format type of the output : default 'xl'
                  Not valid with -f option.
                  Run 'ecmdquery formats' to view available formats

-fb <filename>   Specify full path and filename to binary file to write data from
                  system
                  Not valid with -o option.
                  Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format
                  Incase multiple chip positions are specified the target string
will
                  be postfixed. E.g. filename "datafile" becomes "datafile.k0n0s0p0"

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
                  to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
                  to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
                  to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
                  -pall to act on all chips.

-----

Examples:      > getvpdimage pu 128 -p0
                > getvpdimage l3cache 128 -fb datafile
```

### 3.9.2 getvpdkeyword

#### Syntax:

```
Syntax: getvpdkeyword <ChipSelect> <RecordName> <Keyword> <NumBytes> -o<format>  [-k#]
        [-n#] [-s#] [-p#]
        getvpdkeyword <ChipSelect> <RecordName> <Keyword> <NumBytes> -fb<filename> [-k#]
        [-n#] [-s#] [-p#]
```

ECMD: Core Common Function

Function: Display the contents of a module VPD record.

## eCMD Command Line Interface

Parameters:  
-----

ChipSelect	Chip to get the module VPD record contents from
RecordName	VPD Record Name (in quotes)
KeyWord	VPD Keyword (in quotes)
-o<format>[opt]	Specifies the format type of the output : default 'xl' Not valid with -f option. Run 'ecmdquery formats' to view available formats
-fb <filename>	Specify full path and filename to binary file to write data from system Not valid with -o option. Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format Incase multiple chip positions are specified the target string will be postfixed. E.g. filename "datafile" becomes "datafile.k0n0s0p0"
-k# [optional]	Specify which cage to act on (0 is default). Specify -kall to act on all cages.
-n# [optional]	Specify which node to act on (0 is default). Specify -nall to act on all nodes.
-s# [optional]	Specify which slot to act on (0 is default). Specify -sall to act on all slots.
-p# [optional]	Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----

Examples:           > getvpdkeyword pu "record" "key" 128 -p0  
                  > getvpdkeyword l3cache "record" "key" 128 -fb datafile

### 3.9.3 putvpdimage

#### Syntax:

Syntax: putvpdimage <ChipSelect> <Data> [-i<format>] [-k#] [-n#] [-s#] [-p#]  
putvpdimage <ChipSelect> -fb<filename>           [-k#] [-n#] [-s#] [-p#]

ECMD:               Core Common Function

Function:           Write the specified data to module VPD

Parameters:  
-----

ChipSelect	Chip to write the VPD image into
Data	Data to write into VPD. Not valid with -f option Format specified by -i<format>
-fb <filename>	Specify full path and filename to binary file to load to system Uses ecmdDataBuffer::ECMD_SAVE_FORMAT_BINARY_DATA format
-i<format>[opt]	Specifies the format type of input data : default 'xl' Not valid with -f option Run 'ecmdquery formats' to view available formats

## eCMD Command Line Interface

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-----

Examples:        > putvpdimage pu deadbeef80000000 -p0,1  
                 > putvpdimage l3cache -fb datafile

### 3.9.4 putvpdkeyword

#### Syntax:

Syntax: putvpdkeyword <ChipSelect> <RecordName> <Keyword> <Data> [-i<format>] [-k#] [-n#]  
         [-s#] [-p#]  
         putvpdkeyword <ChipSelect> <RecordName> <Keyword> -fb<filename>                [-k#] [-n#]  
         [-s#] [-p#]

ECMD:            Core Common Function

Function:        Write the specified data to module VPD keyword

Parameters:

-----  
ChipSelect        Chip to write the VPD data into

RecordName       VPD Record Name (in quotes)

KeyWord           VPD Keyword (in quotes)

Data              Data to write into VPD. Not valid with -f option  
                    Format specified by -i<format>

-fb <filename>    Specify full path and filename to binary file to load to system  
                    Uses ecmdDataBuffer::ECMD\_SAVE\_FORMAT\_BINARY\_DATA format

-i<format>[opt]   Specifies the format type of input data : default 'xl'  
                    Not valid with -f option  
                    Run 'ecmdquery formats' to view available formats

-k# [optional]    Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional]    Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional]    Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional]    Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

## eCMD Command Line Interface

---

Examples:        > putvpdkeyword pu "record" "key" deadbeef80000000 -p0,1  
                 > putvpdkeyword l3cache "record" "key" -fb datafile

## 3.10 Simulation Commands

### 3.10.1 simaet

#### Syntax:

Syntax: `simaet on | off | flush`

ECMD: Core Common Function

Function: Start/Stop Simulation AET logging

Parameters:

-----  
`on` Enable AET

`off` Disable AET

`flush` Flush AET to disk  
-----

Example: `simaet on`  
`simaet off`

#### Examples:

### 3.10.2 simcheckpoint

#### Syntax:

Syntax: `simcheckpoint <checkpoint name>`

ECMD: Core Common Function

Function: Store a checkpoint to the specified file

Parameters:

-----  
`checkpointname` name to store checkpoint under  
-----

Example: `simcheckpoint boot`

#### Examples:

### 3.10.3 simclock

#### Syntax:

## eCMD Command Line Interface

Syntax: `simclock <cycles>`

ECMD: Core Common Function

Function: Clock the simulator

Parameters:

-----  
`cycles`            Number of cycles to clock the simulator  
-----

Example:            `simclock 1000`

### Examples:

## 3.10.4 simecho

### Syntax:

Syntax: `simecho <message>`

ECMD: Core Common Function

Function: Echo a string to stdout as well as sim logs

Parameters:

-----  
`message`           String to echo to sim  
-----

Example:            `simecho "Hello"`

### Examples:

## 3.10.5 simexit

### Syntax:

Syntax: `simexit [<rc> <message>]`

ECMD: Core Common Function

Function: Close down a simulation

Parameters:

-----  
`rc`            [opt] Testcase failure return code to pass to simulation

`message`    [opt] Testcase failure message to pass to simulation

```
-----  
Example:      simexit
```

## Examples:

### 3.10.6 simEXPECTFAC

#### Syntax:

Syntax: `simEXPECTFAC <facname> <data> <length> [<row> <offset>] [-i<format>]`

ECMD: Core Common Function

Function: Perform expect on simulation facility using name

Parameters:

```
-----  
facname      Must be a facility name
```

data Data for expect on facility  
 Format specified by -i<format>

length Bit length of data

row [optional] Facility row

offset [opt] Facility offset

-i<format>[opt] Specifies the format type of input data : default 'xr'  
 Run 'ecmdquery formats' to view available formats

```
-----  
Example:      simEXPECTFAC TITAN.TCKFREQ C 4
```

## Examples:

### 3.10.7 simexpecttcfac

#### Syntax:

Syntax: `simexpecttcfac <facname> <data> [<row> | -subset <startbit> <numbits>]  
-i<format>`

ECMD: Core Common Function

Function: Perform expect on a TCFAC Facility

Parameters:

```
-----  
facname      Must be a facility name
```



## eCMD Command Line Interface

data                    Data for expect  
                      Format specified by -i<format>

row   [optional] Facility row - not valid with -subset

startbit   [opt] Facility offset - not valid with row

numbits    [opt] Number of bits from startbit to read - not valid with row

-i<format>[opt] Specifies the format type of input data : default 'xr'  
                  Run 'ecmdquery formats' to view available formats

-----

Example:                simexpecttcfac TITAN.TCKFREQ F

### Examples:

## 3.10.8 simgetcurrentcycle

### Syntax:

Syntax: simgetcurrentcycle

ECMD:                    Core Common Function

Function:                Retrieve the current cycle count

Parameters:

-----

-----

Example:                simgetcurrentcycle

### Examples:

## 3.10.9 simGETFAC

### Syntax:

Syntax: simGETFAC <facname> <length> [<row> <offset>] [-o<format>]

ECMD:                    Core Common Function

Function:                Read a Simulation Facility using a facility name

Parameters:

-----

facname                Must be a facility name

length                 Bit length of symbol to read

## eCMD Command Line Interface

```
row [optional] Facility row
offset [opt] Facility offset
-o<format>[opt] Specifies the format type of the output : default 'xr'
                  Run 'ecmdquery formats' to view available formats
```

---

Example:           simGETFAC TITAN.TCKFREQ 4

### Examples:

## 3.10.10 simGETFACX

### Syntax:

Syntax: simGETFACX <facname> <length> [<row> <offset>]

ECMD:           Core Common Function

Function:       Read a Simulation Facility using a facility name  
                  Displaying Xstate data. format: 'bX'

#### Parameters:

---

facname         Must be a facility name

length          Bit length of symbol to read

row [optional] Facility row

offset         [opt] Facility offset

---

Example:           simGETFACX TITAN.TCKFREQ 4

### Examples:

## 3.10.11 simgettcfac

### Syntax:

Syntax: simgettcfac <facname> [<row> | -subset <startbit> <numbits>] [-o<format>]

ECMD:           Core Common Function

Function:       Read a TCFAC Facility

Parameters:

---

## eCMD Command Line Interface

facname            Must be a facility name

row   [optional] Facility row - not valid with -subset

startbit   [opt] Facility offset - not valid with row

numbits   [opt] Number of bits from startbit to read - not valid with row

-o<format>[opt] Specifies the format type of the output : default 'xr'  
                  Run 'ecmdquery formats' to view available formats

-----

Example:            simgettcfac TITAN.TCKFREQ

## Examples:

### 3.10.12 simgethierarchy

#### Syntax:

Syntax: simgethierarchy <ChipSelect> [-k#] [-n#] [-s#] [-p#]

ECMD:            Core Common Function

Function:        Retrieve the Simulation hierarchy for a give chip position

Parameters:

-----

ChipSelect       Specifies the chip to operate on.

-k#   [optional] Specify which cage to act on (0 is default). Specify -kall  
                  to act on all cages.

-n#   [optional] Specify which node to act on (0 is default). Specify -nall  
                  to act on all nodes.

-s#   [optional] Specify which slot to act on (0 is default). Specify -sall  
                  to act on all slots.

-p#   [optional] Specify which chip position to act on (0 is default). Specify  
                  -pall to act on all chips.

-----

Example:           > simgethierarchy pu -p2  
                     Model hierarchy for target pu        k0:n0:s0:p00        is :  
                     P0.B.S.LT.EM.

### 3.10.13 siminit

#### Syntax:

Syntax: siminit [<checkpoint>]

ECMD:            Core Common Function

Function: Initialize the simulation

Parameters:

-----  
 checkpoint[opt] Name of checkpoint to load  
 -----

Example:       siminit  
               siminit boot

## Examples:

### 3.10.14 simPUTFAC

#### Syntax:

Syntax: simPUTFAC <facname> <data> <length> [<row> <offset>] [-i<format>]

ECMD: Core Common Function

Function: Write a simulation facility using a name

Parameters:

-----  
 facname       Must be a facility name

data           Data to write to facility  
               Format specified by -i<format>

length         Bit length of symbol to write

row [optional] Facility row

offset [opt] Facility offset

-i<format>[opt] Specifies the format type of input data : default 'xr'  
                   Run 'ecmdquery formats' to view available formats  
 -----

Example:       simPUTFAC TITAN.TCKFREQ C 4

## Examples:

### 3.10.15 simPUTFACX

#### Syntax:

Syntax: simPUTFACX <facname> <data> [<row> <offset>]

ECMD: Core Common Function

## eCMD Command Line Interface

Function:        Write a simulation facility using a name.  
                 Write with Xstate data: format 'bX'

Parameters:

-----  
facname        Must be a facility name  
  
data            X-State Data to write to facility  
  
row    [optional] Facility row  
  
offset        [opt] Facility offset  
  
-----

Example:        simPUTFACX TITAN.TCKFREQ 11XX01

### Examples:

## 3.10.16 simputtcfac

### Syntax:

Syntax: simputtcfac <facname> <data> [<row> <# of rows>] -i<format>

ECMD:            Core Common Function

Function:        Put a TCFAC Facility

Parameters:

-----  
facname        Must be a facility name  
  
data            Data to put  
                 Format specified by -i<format>  
  
row    [optional] Facility row  
  
# of rows [opt] Number of rows to put  
  
-i<format>[opt] Specifies the format type of input data : default 'xr'  
                 Run 'ecmdquery formats' to view available formats  
  
-----

Example:        simputtcfac TITAN.TCKFREQ F

### Examples:

## 3.10.17 simrestart

### Syntax:

## eCMD Command Line Interface

Syntax: `simrestart <checkpoint name>`

ECMD: Core Common Function

Function: Load a checkpoint from the specified file

Parameters:

-----  
`checkpointname` name to load checkpoint from  
-----

Example: `simrestart boot`

### Examples:

## 3.10.18 simSTKFAC

### Syntax:

Syntax: `simSTKFAC <facname> <data> <length> [<row> <offset>] [-i<format>]`

ECMD: Core Common Function

Function: Stick a simulation facility using name

Parameters:

-----  
`facname` Must be a facility name

`data` Data for operation  
Format specified by `-i<format>`

`length` Bit length of data

`row` [optional] Facility row

`offset` [opt] Facility offset

`-i<format>`[opt] Specifies the format type of input data : default 'xr'  
Run 'ecmdquery formats' to view available formats  
-----

Example: `simSTKFAC TITAN.TCKFREQ C 4`

### Examples:

## 3.10.19 simstktcfac

### Syntax:

## eCMD Command Line Interface

Syntax: `simstktcfac <facname> <data> <length> [<row> <# of rows>] -i<format>`

ECMD: Core Common Function

Function: Stick a TCFAC Facility

Parameters:

-----  
facname Must be a facility name

data Data to stick  
Format specified by -i<format>

length Bit length of data

row [optional] Facility row

# of rows [opt] Number of rows to stick

-i<format>[opt] Specifies the format type of input data : default 'xr'  
Run 'ecmdquery formats' to view available formats

-----  
Example: `simstktcfac TITAN.TCKFREQ F 4`

### Examples:

## 3.10.20 simSUBCMD

### Syntax:

Syntax: `simSUBCMD <command>`

ECMD: Core Common Function

Function: Run an rtx SUBCMD

Parameters:

-----  
command rtx command to run

-----  
Example: `simSUBCMD run left`

### Examples:

## 3.10.21 simtckinterval

### Syntax:

Syntax: `simtckinterval <interval>`

## eCMD Command Line Interface

```
ECMD:          Core Common Function

Function:      Adjust the TCK Interval

Parameters:
-----
interval      New Interval
-----

Example:       simtckinterval 18
```

### 3.10.22 simUNSTICK

#### Syntax:

Syntax: simUNSTICK <facname> <length> [<row> <offset>]

```
ECMD:          Core Common Function

Function:      Unstick a Simulation Facility using a name

Parameters:
-----
facname       Must be a facility symbol name

length       Bit length of symbol

row [optional] Facility row

offset [opt] Facility offset
-----

Example:       simUNSTICK TITAN.TCKFREQ 4
```

#### Examples:

### 3.10.23 simunsticktcfac

#### Syntax:

Syntax: simunsticktcfac <facname> [<data> <length> [<row> <# of rows>]] -i<format>

```
ECMD:          Core Common Function

Function:      Unstick a TCFAC Facility

Parameters:
-----
facname       Must be a facility name
```



## eCMD Command Line Interface

data [opt] Data to write with unstick  
Format specified by -i<format>

length [opt] Bit length of data

row [optional] Facility row

# of rows [opt] Number of rows to unstick

-i<format>[opt] Specifies the format type of input data : default 'xr'  
Run 'ecmdquery formats' to view available formats

-----

Example: simunsticketcfac TITAN.TCKFREQ

## Examples:

## 4 CIP (Cronus/IP) Extension Commands

These are functions that are only support by Cronus and GFW on IP Series Systems.

### 4.1 Processor Functions

#### 4.1.1 cipbreakpoint

##### Syntax:

Syntax: `cipbreakpoint set|clear <Type> <Address> [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]`

ECMD: CIP (Cronus/IP) Extension

Function: Set|Clear a processor hardware breakpoint using a real address

##### Parameters:

---

set/clear	Set or Clear the breakpoint
Type	Type of breakpoint to use either (IABR, DABR, CIABR)
Address	64 bit address of breakpoint (Hex-Right)
-k# [optional]	Specify which cage to act on (0 is default). Specify -kall to act on all cages.
-n# [optional]	Specify which node to act on (0 is default). Specify -nall to act on all nodes.
-s# [optional]	Specify which slot to act on (0 is default). Specify -sall to act on all slots.
-p# [optional]	Specify which chip position to act on (0 is default). Specify -pall to act on all chips.
-c# [optional]	Specify which processor core to act on (0 is default). Specify -call to act on all cores.
-t# [optional]	Specify which processor thread to act on (0 is default). Specify -tall to act on all threads, -talive to act on all alive threads.

---

Examples: > `cipbreakpoint set IABR 8000000000FAC230`

#### 4.1.2 cipinstruct

##### Syntax:

Syntax: `cipinstruct start|sreset|stop|step [<steps>] [all] [-v] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]`

ECMD: CIP (Cronus/IP) Extension

Function: Start/Stop/Step Processor instruction execution. Processors are looped upon to perform action they are not performed to all

## eCMD Command Line Interface

processors in sync.

### Parameters:

-----  
start/stop/step/sreset Start, stop, or step the processors.

steps [opt] Number of steps to execute

all [optional] Start/Stop all processors configured in system

-v [optional] Print out IAR after each instruction step

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-t# [optional] Specify which processor thread to act on (0 is default). Specify -tall to act on all threads, -talive to act on all alive threads.

-----  
Examples: > cipinstruct step 10 -p2 -c1  
> cipinstruct start all  
> cipinstruct stop -p3 -c0

## 5 CRO (Cronus) Extension Commands

These are functions that are only supported by Cronus.

### 5.1 Memory Display/Alter Commands

#### 5.1.1 crodmarandtest

##### Syntax:

Syntax: crodmarandtest NumLoops [-f<filename> | StartAddr EndAddr [AccessRest]] [-s#]

```

ECMD:          CRO (Cronus) Extension

Function:      Runs crodmarandtest - Dma/PSI Exerciser

IMPORTANT NOTE:  crodmarandtest is limited to < 4G addresses because of the
FSP

Parameters:
-----
NumLoops      Number of loops to execute. Set to -1 for infinite

-s# [optional]  Max Size of DMA Transfers in Decimal - DEFAULT 32K

-f<> [optional] Address Map file

-proc          Use putMemProc/getMemProc rather than putMemDma/getMemDma

StartAddr     Start of Address range in Hex

EndAddr       End of Address range in Hex - enter 'MAX' to default to
              Maximum memory, See below.

AccessRest[opt] Access Restrictions to this address range
                'W' - Write Only
                'R' - Read Only
                'N' - None - Does Data Compares - DEFAULT
                'AW' - Writes address as data for entire range - see NOTE below
                'AR' - Reads and expects data to be address - see NOTE below
                'AWR' - Writes address as data for entire range, then compares
entire range
                - see NOTE below
-----

Examples:      crodmarandtest 10 0 100000
                crodmarandtest -1 100000 200000 W -s1024
                crodmarandtest -1 100000 MAX R -s4096
                crodmarandtest 10 -fmem.map

MEMORY MAP FILE FORMAT:

#START  END      - PERM  COMMENTS
00000000 00003FFF x R  Interrupt Handlers
00004000 00007FFF x W  Supervisor
0000D000 0000DFFF N A  UNUSED

NOTE: The PERM column is for Access permissions, either R(ead) W(rite) or A(ny)
      The column with a '-' is unused for dmaXer.

```

## eCMD Command Line Interface

The PERM's AW, AR, AWR are not supported in the memory map files  
'#' starts comment lines

-----  
NOTE: For speed reasons, the address written to memory using the 'AW' 'AR' 'AWR'  
tests will be byte swapped in memory. Example: 0xFEEDBEEF will be in memory as  
0xEFBEEDFE.

### 5.1.2 crogetl2data

#### Syntax:

Syntax: crogetl2data Slice CongClass Set [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: CRO (Cronus) Extension

Function: Displays full cache line of L2 cache data.

Note: Data is displayed in hex as follows:

word0 word1 bit65 word2 word3 bit65  
word4 word5 bit65 word6 word7 bit65  
..

Parameters:

-----  
Slice A or B.

CongrClass Congruence class(0:7). Right-aligned hex.

Set Set(0:7) P6 L2 cache is 8-way set associative.

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-----  
Examples: > crogetl2data b 3 1 -p0  
> crogetl2data a 8 8 -p4

## 5.1.3 crogetl2dir

### Syntax:

Syntax: crogetl2dir Slice CongClass Set [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: CRO (Cronus) Extension

Function: Displays an L2 directory entry.

Parameters:

-----  
Slice A or B.

CongrClass Congruence class(0:7). Right-aligned hex.

Set Set(0:7) P6 L2 cache is 8-way set associative.

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----  
Examples: > crogetl2dir b 3 1 -p0  
> crogetl2dir a 8 8 -p4

## 5.1.4 crogetl2

### Syntax:

Syntax: crogetl2 Address [Num] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: CRO (Cronus) Extension

Function: Displays cache lines from Level 2 processor cache. Looks for valid entries in the L2 directory to match the tag id given in the real address.

Parameters:

-----  
Address Real address(0:63) from which to get data. Right-aligned hex.

Num [optional] Decimal number of cache lines to display (1 is default), starting at Address.

## eCMD Command Line Interface

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----

Examples:        > crogetl2 100 3 -p0  
                 > crogetl2 ffb0001f00 -p1

## 5.1.5 croputl2data

### Syntax:

Syntax: croputl2data Slice CongClass Set Data[-k#] [-n#] [-s#] [-p#] [-c#]

ECMD:            CRO (Cronus) Extension

Function:        Writes up to a full cache line of L2 cache data.  
                 Data previously stored in the entry is over-written.

Parameters:

-----  
Slice            A or B.

CongrClass      Congruence class(0:7). Right-aligned hex.

Set             Set(0:7) P6 L2 cache is 8-way set associative.

Data            Words are in hex. Bit65 is 0 or 1. Format is as follows:  
                 word0 word1 b65 word2 word3 b65 [word4 word5 b65...]

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-----

## eCMD Command Line Interface

Examples:           > croputl2data b 3 1 FEEDBEEF FEEDB0B0 1 11223344 55667788 0 -p1  
                  > croputl2data a 8 8 BFBFBFBF DFDFDFDF 1 AFAFAFAF AFAFAFAF 1 -p4

### 5.1.6 croputl2dir

#### Syntax:

Syntax: croputl2dir Slice CongClass Set Data [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD:               CRO (Cronus) Extension

Function:           Writes an L2 directory entry. Data previously stored in the  
entry               is over-written.

Parameters:

-----  
Slice                A or B.

CongrClass       Congruence class(0:7). Right-aligned hex.

Set               Set(0:7) P6 L2 cache is 8-way set associative.

Data               Right-aligned hex, bits(0:37) of the directory entry

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
-pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
-call to act on all cores.

-----  
Examples:           > croputl2dir b 3 1 0FF1238068 -p0  
                  > croputl2dir a 8 8 0FFFF4334C -p4

### 5.1.7 croputl2

#### Syntax:

Syntax: croputl2 Address Set Data [-mesi mesiState] [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD:               CRO (Cronus) Extension

Function:           Replaces up to one cache line in Level 2 processor cache.

Parameters:



## eCMD Command Line Interface

```
-----
Address      Real address(0:63) from which to get data.  Right-aligned hex.

Set          Set(0:7)  P6 L2 cache is 8-way set associative.

Data         Words are in hex.  Bit65 is 0 or 1.  Format is as follows:

              word0 word1 b65 word2 word3 b65 [word4 word5 b65...]

-mesi [optional] Mesi state setting (M, Modified is default):
          I = Invalid
          S = Shared
          SI = SI Shared
          T = Tagged
          TE = Tagged Exclusive
          M = Modified
          ME = Modified Exclusive
          MU = Unsolicited Modified

-k#  [optional] Specify which cage to act on (0 is default). Specify -kall
      to act on all cages.

-n#  [optional] Specify which node to act on (0 is default). Specify -nall
      to act on all nodes.

-s#  [optional] Specify which slot to act on (0 is default). Specify -sall
      to act on all slots.

-p#  [optional] Specify which chip position to act on (0 is default). Specify
      -pall to act on all chips.

-c#  [optional] Specify which processor core to act on (0 is default). Specify
      -call to act on all cores.

-----

Examples:  > croputl2 100 3 FFFFB0B0 AAAABBBB 1 -state ME -p0
           > croputl2 1f0000 7 FFFFB0B0 AAAABBBB 1 FFFFB0B0 AAAABBBB 1
FFFB0B0 AAAABBBB 1 -p1
```

## 6 EIP (Eclipz IP) Extension Commands

These are functions that are only supported by Cronus and GFW on Eclipz IP Series Systems.

### 6.1 Processor Functions

#### 6.1.1 eipgetslb

##### Syntax:

Syntax: eipgetslb <StartEntry> [<numEntries>] [-k#] [-n#] [-s#] [-p#] [-c#] [-t#]

ECMD: EIP (Eclipz/IP) Extension

Function: Gets Segment Lookaside Buffer entry.

Parameters:

StartEntry SLB Entry to read (Decimal)

numEntries Specifies number of entries to get from starting entry (Decimal)

-k# [optional] Specify which cage to act on (0 is default). Specify -kall to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify -call to act on all cores.

-t# [optional] Specify which processor thread to act on (0 is default). Specify -tall to act on all threads, -talive to act on all alive threads.

Notes: Entries are in the range of 0 to 63 for each thread.

Example: > eipgetslb 6 -p0,1 -t1

#### 6.1.2 eipproccleanup

##### Syntax:

Syntax: eipproccleanup [-k#]

ECMD: EIP (Eclipz/IP) Extension

Function: Cleans up processors to enable memory DA after a checkstop

## eCMD Command Line Interface

### Parameters:

-----  
-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
to act on all cages.  
-----

Example:           > eipproccleanup -k0

## 6.2 Miscellaneous Commands

### 6.2.1 crogetconfig

#### Syntax:

Syntax: crogetconfig [<ChipSelect>] <ConfigName> [-o<format>]  
 [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: CRO (Cronus) Extension

Function: Read the specified configuration variable and display to screen

Parameters:

-----  
 ChipSelect[opt] Chip to read data from.

ConfigName Name of configuration variable to fetch

-o<format>[opt] Output Format : default 'xl'. Only valid with numeric data  
 Run 'ecmdquery formats' to view available formats

-k# [optional] Specify which cage to act on (0 is default). Specify -kall  
 to act on all cages.

-n# [optional] Specify which node to act on (0 is default). Specify -nall  
 to act on all nodes.

-s# [optional] Specify which slot to act on (0 is default). Specify -sall  
 to act on all slots.

-p# [optional] Specify which chip position to act on (0 is default). Specify  
 -pall to act on all chips.

-c# [optional] Specify which processor core to act on (0 is default). Specify  
 -call to act on all cores.

-----  
 Examples: > crogetconfig SIM\_BROADSIDE\_MODE  
 SIM\_BROADSIDE\_MODE = none  
 > crogetconfig pu PLL\_TUNE  
 pu k0:n0:s0:p00  
 PLL\_TUNE = 3

### 6.2.2 crosetconfig

#### Syntax:

Syntax: crosetconfig [<ChipSelect>] <ConfigName> <Value> [-i<format>]  
 [-k#] [-n#] [-s#] [-p#] [-c#]

ECMD: CRO (Cronus) Extension

Function: Write the specified configuration variable

Parameters:

-----  
 ChipSelect[opt] Chip to write data to.

## eCMD Command Line Interface

ConfigName      Name of configuration variable to write

Value            Value to set configuration variable to

-i<format>[opt] Output Format : default is ascii.  
                 Run 'ecmdquery formats' to view available formats

-k#    [optional] Specify which cage to act on (0 is default). Specify -kall  
                 to act on all cages.

-n#    [optional] Specify which node to act on (0 is default). Specify -nall  
                 to act on all nodes.

-s#    [optional] Specify which slot to act on (0 is default). Specify -sall  
                 to act on all slots.

-p#    [optional] Specify which chip position to act on (0 is default). Specify  
                 -pall to act on all chips.

-c#    [optional] Specify which processor core to act on (0 is default). Specify  
                 -call to act on all cores.

-----

Examples:        > crosetconfig SIM\_BROADSIDE\_MODE scan  
                 > crosetconfig pu PLL\_TUNE 3 -id