

eCMD Command Line Interface

Version .1

Contact: Chris Engel / Hans-Joachim Hartmann

IBM Confidential

Table of Contents

1 Introduction.....	4
2 Usage Instructions.....	4
3 eCMD Common Commands.....	4
3.1 Common Command Arguments.....	4
3.1.1 Data Output Formatting (-o<format>).....	4
3.1.2 Data Input Formatting (-i<format>).....	6
3.2 Chip Display/Alter Commands.....	8
3.2.1 checkrings.....	8
3.2.2 getbits.....	8
3.2.3 getlatch.....	9
3.2.4 getringdump.....	10
3.2.5 getscom.....	11
3.2.6 getspy.....	12
3.2.7 pollscom.....	13
3.2.8 putbits.....	14
3.2.9 putlatch.....	15
3.2.10 putpattern.....	16
3.2.11 putscom.....	16
3.2.12 putspy.....	17
3.2.13 sendcmd.....	18
3.3 Miscellaneous Commands.....	20
3.3.1 ecmdquery.....	20
3.4 Simulation Commands.....	20
3.4.1 simaet.....	20
3.4.2 simcheckpoint.....	21
3.4.3 simclock.....	21
3.4.4 simecho.....	21
3.4.5 simexit.....	22
3.4.6 simEXPECTFAC.....	22
3.4.7 simEXPECTFACS.....	23
3.4.8 simexpecttcfac.....	23
3.4.9 simgetcurrentcycle.....	24
3.4.10 simGETFAC.....	24
3.4.11 simGETFACS.....	25
3.4.12 simGETFACX.....	25
3.4.13 simgettcfac.....	26
3.4.14 siminit.....	26
3.4.15 simPUTFAC.....	27

eCMD Command Line Interface

3.4.16	simPUTFACS.....	27
3.4.17	simPUTFACX.....	28
3.4.18	simputtcfac.....	28
3.4.19	simrestart.....	29
3.4.20	simSTKFAC.....	29
3.4.21	simSTKFACS.....	30
3.4.22	simstktcfac.....	30
3.4.23	simSUBCMD.....	31
3.4.24	simsymbol.....	31
3.4.25	simUNSTICK.....	31
3.4.26	simUNSTICKS.....	32
3.4.27	simunsticktcfac.....	32

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/>

2 Usage Instructions

3 eCMD Common Commands

These are the core command line functions available through the eCMD interface and the syntax of the command.

3.1 Common Command Arguments

3.1.1 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

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

Left-aligned Hex Words : -oXW

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

Left-aligned Hex Word Columns : -oXW2

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

Right-aligned Hex : -oXR

eCMD Command Line Interface

```

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

```

Right-aligned Hex Words : -oXRW

```

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

```

Right-aligned Hex Word Columns : -oRXW2

```

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

```

Binary : -oB

[illegible]

Binary Nibbles : -oBN

```

FORMAT: BN
gr      k0:n0:p00:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
gr      k0:n0:p01:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
gr      k0:n0:p02:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

```

Binary Nibble Columns : -oBN8

```

FORMAT: BN8
gr      k0:n0:p00:c0

```

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

Binary Words : -oBW

[illegible]

Binary Word Columns : -oBW1

eCMD Command Line Interface

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

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

Memory Output : -omem

```
FORMAT: MEM
gr      k0:n0:p00
00000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF
00000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF
```

Memory Output – Ascii Decode : -omema

```
FORMAT: MEMA
gr      k0:n0:p00
00000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISistHEasciITE]
00000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [XT.....]
```

Memory Output – Ebcedic Decode : -omeme

```
FORMAT: MEME
gr      k0:n0:p00
00000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISistHEebcedic]
00000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [TEXT.....]
```

Memory Output – Dcard Format : -odcard

```
FORMAT: DCARD
gr      k0:n0:p00
D 00000000000000100 FEEDBEEFFEEDBEEF 0
D 00000000000000108 FEEDBEEFFEEDBEEF 1
D 00000000000000110 FEEDBEEFFEEDBEEF 0
D 00000000000000118 FEEDBEEFFEEDBEEF 1
```

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

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

3.1.2 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

3.2 Chip Display/Alter Commands

3.2.1 checkrings

Syntax:

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

```

ECMD:          Core Common Function

Function:      checks for stuck bits and verifies scanring 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

-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.

-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:

3.2.2 getbits

Syntax:

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

```

ECMD:          Core Common Function

Function:      Long scans bits out of a chip's selected ring. (non-destructive)

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

RingName       Specifies chip ring to operate on.  For a list of available

```


eCMD Command Line Interface

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 'all' 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.

-o<format>[opt] Specifies the format type of both the output and the expect-value: default 'b'

-X [optional] For simulation use only. Must be used to display any Xstates data in ring.

-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.

-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 all
```

Examples:

3.2.3 getlatch

Syntax:

Syntax: `getlatch <ChipSelect> <RingName> <LatchName> [<Start> <Numbits>] [-exact]`
`[-compress] [-exp <value>]`
`[-o<format>] [-k#] [-n#] [-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). The -compress flag searches past the first match for more matches and concatenates data if the register is broken into separate lines.

Parameters:

eCMD Command Line Interface

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

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 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,
data.           and -exact not selected, prints a warning message instead of the
                If -exact is selected, immediately return an error.

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

-compress       [opt] Displays as a single line any registers that are broken up
                into multiple lines in the scandef file.

-exp [optional] Provide an expected-value as the last argument. Returns error if
                data miscompare, else nothing. Automatically turns on -nop.

-o<format>[opt] Specifies the format type of both the output and
                the expect-value
                Defaults to binary for < 8 bits and hex for >= 8 bits.

-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.

-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:       getreg pu cp_abist LATCH0
                getreg pu cp_fpu GCP.PFPU.FP_UNIT0.FP_REG -ox -exact -compress
-expect feed
```

Examples:

3.2.4 getringdump

Syntax:

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

ECMD:          Core Common Function

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

Parameters:
```

eCMD Command Line Interface

```
-----
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 smi
                       ecmdquery rings pu

-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.

-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 hex for > 8.

Examples:       getringdump memctrl int
                  getringdump pu gps_fuse
                  getringdump pu gps_fuse gps_abist cp_ras
```

Examples:

3.2.5 getscom

Syntax:

```
Syntax: getscom <ChipSelect> <ScanCommAddr> [-v] [-k#] [-n#] [-p#] [-c#] [-o<format>]
          [-exp <Bits0-31> <Bits32-63> <Bits64-96>] [-mask <Bits0-31> <Bits32-63>
          <Bits64-96>]]
```

```
ECMD:          Core Common Function

Function:       Gets Scan Communications registers.

Parameters:
-----
ChipSelect      Chip to get scancomm data from.

ScanCommAddr    Address in hex.

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

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

-o<format>[opt] Output Format : default xw - See doc for details

-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.
```

eCMD Command Line Interface

```

-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 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 feed0000 00000001 0 -mask ffff0000
0000ffff 0

```

Examples:

3.2.6 getspy

Syntax:

Syntax: `getspy <ChipSelect> <SpyName> [<Start> <Numbits>] [-exp <value>]`
`[-o<format>] [-k#] [-n#] [-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

-only -exp [optional] Provides an expected value as the last argument. Returns error
                     if miscompare. Expected value string is read depending
                     on format flag.

-o<format>[opt]     Specifies the format type of both the output and
                     the expect-value. (default: hex-left)
                     For enums use -oenum

-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.

-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

```

eCMD Command Line Interface

-call to act on all cores.

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

Examples:

3.2.7 pollscm

Syntax:

Syntax: pollscm <ChipSelect> <ScanCommAddr> [-exp <Bits0-31> <Bits32-63> <Bits64-96>
[-mask <Bits0-31> <Bits32-63> <Bits64-96>]]
 [-limit #[s|c]] [-interval #[s|c]] [-verbose] [-k#] [-n#]
[-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:

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*] Scm data is AND'ed with the mask bits before checking against
 expected value.

-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.
 If limit = 0 and -interval is not specified, the interval defaults
 to 5 seconds.

-interval # [*opt*] Time between getscoms. To specify in seconds, append an 's'
 to #. To specify number of cycles for simulation, append a
 'c' to # (number of cycles must be > 1000). If -limit is not
 specified with -interval, the limit defaults to ~240 seconds
 or 1 million cycles, depending on how -interval is specified.
 If neither -limit or -interval are specified, limit defaults to
 60 seconds and interval defaults to 5 seconds.

-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.

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

-pall

eCMD Command Line Interface

```
to act on all chips.

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

Note:          If used, -interval and -limit must come after -exp.

Examples:      pollscm spinnaker 800009 -exp feed0000 00000001 0 -limit 30s
-interval 10s -verbose -p1
               pollscm spinnaker 800009 -exp feed0000 00000001 0 -mask ffff0000
0000ffff 0 -limit 10
               pollscm outrigger 400020 -limit 100000c -interval 5000c
```

Examples:

3.2.8 putbits

Syntax:

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

```
ECMD:          Core Common Function

Function:       Put bits to the specified chip ring.

Paramaters:
-----
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.  Default is binary.

-i<format>[opt] Specifies the format type of input 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.

-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 -x -p0,1 -c1
```

```
putbits pu int 23 011X001X -p0
```

Examples:

3.2.9 putlatch

Syntax:

```
Syntax: putlatch <ChipSelect> <RingName> <LatchName> [<Start> <Numbits>] <Data>
        [-i<format>] [-k#] [-n#] [-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 successive lines, the register lengths are concatenated to form one complete register.

Parmeters:

```
-----
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 depends on
               format flag (default hex-left).

-i<format>[opt] Specifies the format type of input 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.

-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:      putreg pu cp_fpu GCP.PFP0.FP_UNIT0.FPA_LATCH -ix feed
```

Examples:

3.2.10 putpattern

Syntax:

Syntax: putpattern <ChipSelect> <RingType> <Data> [-i<format>] [-k#] [-n#] [-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. (default: hex-right)

-i<format>[opt] Specifies the format type of input 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.

-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:

3.2.11 putscom

Syntax:

Syntax: putscom <ChipSelect> <ScanCommAddr> <Bits0-31> <Bits32-63> <Bits64-96> [-and | -or] [-k#] [-n#] [-p#] [-c#]

ECMD: Core Common Function

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

Parameters:

eCMD Command Line Interface

```
-----
ChipSelect      Chip to put scancom data to.

ScanCommAddr    Address in right aligned hex.

Bits0-31        Data consists of 8 hex numbers, right aligned.
Bits32-63       Data consists of 8 hex numbers, right aligned.
Bits64-96       Data consists of 8 hex numbers, right aligned.

-and [optional] Input data will be AND'ed with current scom data.
-or  [optional] Input data will be OR'ed with current scom 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.

-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:          Leading zeros are not necessary.

Examples:       > putscom pu 600000 deadbeef 80000000 0 -p0,1 -c1
                > putscom memctrl 2010 00800 488 0
                > putscom 13 40320 00008000 0 0 -or -p12
```

Examples:

3.2.12 putspy

Syntax:

Syntax: putspy <ChipSelect> <SpyName> [<Start> <Numbits>] <Data>
[-i<format>] [-k#] [-n#] [-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 alias.  Specify with numbits.
                Only valid with non-enumerated spy's

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.
```

eCMD Command Line Interface

```
                                Only valid with non-enumerated spy's

Data                            Data to put into spy, either raw data or enum name.

-i<format>[opt]                Specifies the format type of the input data (default: hex-left)
                                For enums use -ienum

-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.

-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:                       putalias pu MYALIAS -ixr feedbeeffeedbeef
                                putalias pu EVERYOTHER 16 4 -ib 1010
                                putalias pu MYEDIAL ENUMVALUE
```

Examples:

3.2.13 sendcmd

Syntax:

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

```
ECMD:                           Core Common Function

Function:                        Send a JTAG Instruction to the chip

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.

-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:                          Leading zeros ARE NECESSARY if the command is not a full
```

eCMD Command Line Interface

16 bits (e.g. ACCESS)
Only valid with JTAG attached chips

Example: sendcmd pu 12 C00008 -p0,1

Examples:

3.3 *Miscellaneous Commands*

3.3.1 **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 [-k#] [-n#] [-p#] - Display all rings available for chip

Example:	ecmdquery rings pu -p0,1

Examples:

3.4 *Simulation Commands*

3.4.1 **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:

3.4.2 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:

3.4.3 simclock

Syntax:

Syntax: `simclock <cycles>`

ECMD: Core Common Function

Function: Clock the simulator

Parameters:

`cycles` Number of cycles to clock the simulator

Example: `simclock 1000`

Examples:

3.4.4 simecho

Syntax:

Syntax: `simecho <message>`

ECMD: Core Common Function

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

Parameters:

eCMD Command Line Interface

message String to echo to sim

Example: simecho "Hello"

Examples:

3.4.5 simexit

Syntax:

Syntax: simexit

ECMD: Core Common Function

Function: Close down a simulation

Parameters:

Example: simexit

Examples:

3.4.6 simEXPECTFAC

Syntax:

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

ECMD: Core Common Function

Function: Perform expect on simulation facility using a symbol

Parameters:

symbol Must be a facility symbol

data Data for expect on facility

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"

Example: simEXPECTFAC 100 C 4

Examples:

3.4.7 simEXPECTFACS

Syntax:

Syntax: `simEXPECTFACS <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

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"

Example: `simEXPECTFACS TITAN.TCKFREQ C 4`

Examples:

3.4.8 simexpecttcfac

Syntax:

Syntax: `simexpecttcfac <facname> <data> [<row>] -i<format>`

ECMD: Core Common Function

Function: Perform expect on a TCFAC Facility

Parameters:

facname Must be a facility name

data Data for expect

row [optional] Facility row

-i<format>[opt] Specifies the format type of the input data: default 'xr'

eCMD Command Line Interface

Example: simexpecttcfac TITAN.TCKFREQ F

Examples:

3.4.9 simgetcurrentcycle

Syntax:

Syntax: simgetcurrentcycle

ECMD: Core Common Function

Function: Retrieve the current cycle count

Parameters:

Example: simgetcurrentcycle

Examples:

3.4.10 simGETFAC

Syntax:

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

ECMD: Core Common Function

Function: Read a Simulation Facility using a symbol

Parameters:

symbol Must be a facility symbol number - hex right aligned

length Bit length of symbol to read

row [optional] Facility row

offset [opt] Facility offset

-o<format>[opt] Specifies the format type of the output: default 'xr'

Example: simGETFAC 100 4

Examples:

3.4.11 simGETFACS

Syntax:

Syntax: `simGETFACS <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

row [optional] Facility row

offset [opt] Facility offset

-o<format>[opt] Specifies the format type of the output: default 'xr'

Example: `simGETFACS TITAN.TCKFREQ 4`

Examples:

3.4.12 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: "b"

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:

3.4.13 simgettcfac

Syntax:

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

ECMD: Core Common Function

Function: Read a TCFAC Facility

Parameters:

`facname` Must be a facility name

`row` [optional] Facility row

`startbit` [opt] Facility offset

`numbits` [opt] Number of bits from startbit to read

`-o<format>`[opt] Specifies the format type of the output: default 'xr'

Example: `simgettcfac TITAN.TCKFREQ`

Examples:

3.4.14 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:

3.4.15 simPUTFAC

Syntax:

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

ECMD: Core Common Function

Function: Write a simulation facility using a symbol

Parameters:

symbol Must be a facility symbol

data Data to write to facility

length Bit length of symbol to read

row [optional] Facility row

offset [opt] Facility offset

-i<format>[opt] Specifies the format type of input data. default: "xr"

Example: `simPUTFAC 100 C 4`

Examples:

3.4.16 simPUTFACS

Syntax:

Syntax: `simPUTFACS <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

length Bit length of symbol to read

row [optional] Facility row

offset [opt] Facility offset

-i<format>[opt] Specifies the format type of input data. default: "xr"

Example: `simPUTFACS TITAN.TCKFREQ C 4`

Examples:

3.4.17 simPUTFACX

Syntax:

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

ECMD: Core Common Function

Function: Write a simulation facility using a name.
Write with Xstate data: format "b"

Parameters:

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

Example: `simPUTFACX TITAN.TCKFREQ 11XX01`

Examples:

3.4.18 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
row [optional] Facility row
of rows [opt] Number of rows to put
-i<format>[opt] Specifies the format type of the input data: default 'xr'

Example: `simputtcfac TITAN.TCKFREQ F`

Examples:

3.4.19 simrestart

Syntax:

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:

3.4.20 simSTKFAC

Syntax:

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

ECMD: Core Common Function

Function: Stick a simulation facility using a symbol

Parameters:

`symbol` Must be a facility symbol

`data` Data for operation

`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"

Example: `simSTKFAC 100 C 4`

Examples:

3.4.21 simSTKFACS

Syntax:

Syntax: `simSTKFACS <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

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"

Example: `simSTKFACS TITAN.TCKFREQ C 4`

Examples:

3.4.22 simstktcfac

Syntax:

Syntax: `simstktcfac <facname> <data> [<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

row [optional] Facility row

of rows [opt] Number of rows to stick

-i<format>[opt] Specifies the format type of the input data: default 'xr'

Example: `simstktcfac TITAN.TCKFREQ F`

Examples:

3.4.23 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:

3.4.24 simsymbol

Syntax:

Syntax: simsymbol <facname>

ECMD: Core Common Function

Function: Look up the symbol id of a facility

Parameters:

facname Facility name to look up symbol

Example: simsymbol TITAN.TCKFREQ

Examples:

3.4.25 simUNSTICK

Syntax:

eCMD Command Line Interface

Syntax: `simUNSTICK <symbol> <length> [<row> <offset>]`

ECMD: Core Common Function

Function: Unstick a Simulation Facility using a symbol

Parameters:

symbol Must be a facility symbol number - hex right aligned

length Bit length of symbol

row [optional] Facility row

offset [opt] Facility offset

Example: `simUNSTICK 100 4`

Examples:

3.4.26 simUNSTICKS

Syntax:

Syntax: `simUNSTICKS <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: `simUNSTICKS 100 4`

Examples:

3.4.27 simunsticktcfac

Syntax:

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

eCMD Command Line Interface

```
ECMD:           Core Common Function

Function:       Unstick a TCFAC Facility

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

data           Data to write with unstick

row [optional] Facility row

# of rows [opt] Number of rows to unstick

-i<format>[opt] Specifies the format type of the input data: default 'xr'

-----

Example:       simunsticktcfac TITAN.TCKFREQ F
```

Examples: