

eCMD Command Line Interface

Version .2

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 getarray.....	8
3.2.3 getbits.....	9
3.2.4 getlatch.....	10
3.2.5 getringdump.....	11
3.2.6 getscom.....	12
3.2.7 getspy.....	13
3.2.8 pollscom.....	14
3.2.9 putarray.....	15
3.2.10 putbits.....	16
3.2.11 putlatch.....	17
3.2.12 putpattern.....	18
3.2.13 putscom.....	18
3.2.14 putspy.....	19
3.2.15 sendcmd.....	20
3.3 Miscellaneous Commands.....	22
3.3.1 ecmdquery.....	22
3.4 Simulation Commands.....	23
3.4.1 simaet.....	23
3.4.2 simcheckpoint.....	23
3.4.3 simclock.....	23
3.4.4 simecho.....	24
3.4.5 simexit.....	24
3.4.6 simEXPECTFAC.....	25
3.4.7 simEXPECTFACS.....	25
3.4.8 simexpecttcfac.....	26
3.4.9 simgetcurrentcycle.....	26
3.4.10 simGETFAC.....	27
3.4.11 simGETFACS.....	27
3.4.12 simGETFACX.....	28
3.4.13 simgettcfac.....	28

eCMD Command Line Interface

3.4.14	siminit.....	29
3.4.15	simPUTFAC.....	29
3.4.16	simPUTFACS.....	30
3.4.17	simPUTFACX.....	30
3.4.18	simputtcfac.....	31
3.4.19	simrestart.....	31
3.4.20	simSTKFAC.....	32
3.4.21	simSTKFACS.....	32
3.4.22	simstktcfac.....	33
3.4.23	simSUBCMD.....	33
3.4.24	simsymbol.....	34
3.4.25	simUNSTICK.....	34
3.4.26	simUNSTICKS.....	35
3.4.27	simunsticktcfac.....	35

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.

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 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      00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
gr      k0:n0:s0:p02:c0      00000000000000000000000000000000
```

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      00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
gr      k0:n0:s0:p02:c0      00000000000000000000000000000000
```

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 : -oRXW2

```
FORMAT: RXW2
gr      k0:n0:s0:p00:c0
0: 00000000 00000000
2: 00000000
gr      k0:n0:s0:p01:c0
0: 00000000 00000000
2: 00000000
```

Binary : -oB

```
FORMAT: B
gr      k0:n0:s0:p00:c0      0000000000000000000000000000000000000000000000000000000000000000
gr      k0:n0:s0:p01:c0      0000000000000000000000000000000000000000000000000000000000000000
gr      k0:n0:s0:p02:c0      0000000000000000000000000000000000000000000000000000000000000000
```

Binary Nibbles : -oBN

```
FORMAT: BN
gr      k0:n0:s0:p00:c0      0000 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 0000
gr      k0:n0:s0:p02:c0      0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Binary Nibble Columns : -oBN8

eCMD Command Line Interface

```
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
```

Memory Output : -omem

```
FORMAT: MEM
gr      k0:n0:s0:p00
000000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF
000000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF
```

Memory Output – Ascii Decode : -omema

```
FORMAT: MEMA
gr      k0:n0:s0:p00
000000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISisTHEasciITE]
000000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [XT.....]
```

Memory Output – Ebcedic Decode : -omeme

```
FORMAT: MEME
gr      k0:n0:s0:p00
000000000000000100: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [THISisTHEebcedic]
000000000000000110: FEEDBEEF FEEDBEEF FEEDBEEF FEEDBEEF [TEXT.....]
```

Memory Output – Dcard Format : -odcard

```
FORMAT: DCARD
gr      k0:n0:s0:p00
D 000000000000000100 FEEDBEEFFEEDBEEF 0
D 000000000000000108 FEEDBEEFFEEDBEEF 1
D 000000000000000110 FEEDBEEFFEEDBEEF 0
D 000000000000000118 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.

3.2 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.

3.3 Chip Display/Alter Commands

3.3.1 checkrings

Syntax:

Syntax: checkrings <ChipSelect> <RingSelect> [-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

-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 sgxbbs ...
Performing 0's test on sgxbbs ...
ecmd.exe checkrings test all
```

3.3.2 getarray

Syntax:

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

ECMD: Core Common Function

eCMD Command Line Interface

```
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 x - See doc for details

-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 deadbeef800000000 -p0,1 -c1
```

Examples:

3.3.3 getbits

Syntax:

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

eCMD Command Line Interface

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. <data> is left aligned hex

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

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

-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.

-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 all

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
```

3.3.4 getlatch

Syntax:

Syntax: getlatch <ChipSelect> <RingName> <LatchName> [<Start> <Numbits>] [-exact] [-nocompress] [-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). The -compress flag searches past the first match

eCMD Command Line Interface

for

more matches and concatenates data if the register is broken into separate lines.

Parameters:

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.
 NOTE : Not allowed when using -nocompress

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 return an error.
 NOTE : Not allowed when using -nocompress

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

-nocompress[opt] Displays Latches as they are broken up in the scandef.

-exp [optional] Provide an expected-value as the last argument. Returns error if data miscompare, else nothing. Data is hex left aligned

-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.

-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.PFP.U.FP_UNIT0.FP_REG -ox -exact -compress

-expect feed

Examples:

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

3.3.5 getringdump

Syntax:

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

ECMD: Core Common Function

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

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 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.

-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 hex for > 8.

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

Examples:


```

                                -call to act on all cores.

-----

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

```

```
> getscom test 800000
test    k0:n0:s0:p00          FEEDBEEF  AAAAAAAA  00000000
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
```

Syntax:

```

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. 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.

```

eCMD Command Line Interface

-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:

3.3.8 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#] [-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:

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.

eCMD Command Line Interface

```
-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.
-----

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:

```
> 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 AAAAAAAAA 00000000
Expected    : FEED0000
ecmd.exe pollscm test 800000 -exp FEED0000 -limit 5
```

3.3.9 putarray

Syntax:

```
Syntax: putarray <ChipSelect> <ArrayName> <ArrayIndex> <ArrayData> [-i<format>]
[-b<modifier>]
               [-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"

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

-b<mod>[opt] Bit modifier to apply to current scom data.

-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.

-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 xgpr0 deadbeef80000000 -p0,1 -c1

Examples:

3.3.10 putbits

Syntax:

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

ECMD: Core Common Function

Function: Put bits to the 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

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

-b<mod>[opt] Bit modifier to apply to 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.

-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

-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:

3.3.11 putlatch

Syntax:

Syntax: putlatch <ChipSelect> <RingName> <LatchName> [<Start> <Numbits>] <Data>
 [-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
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

-b<mod>[opt] Bit modifier to apply to 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.

-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.

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

Examples:

3.3.12 putpattern

Syntax:

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

-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:

3.3.13 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.

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. Default is Hex-Left.

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

-b<mod>[opt] Bit modifier to apply to 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.

-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:

3.3.14 putspy

Syntax:

Syntax: putspy <ChipSelect> <SpyName> [<Start> <Numbits>] <Data>

eCMD Command Line Interface

```
[-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

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.

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

-b<mod>[opt]   Bit modifier to apply to 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.

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

Examples:

3.3.15 sendcmd

Syntax:

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

ECMD:          Core Common Function

Function:      Send a JTAG Instruction to the chip
```

eCMD Command Line Interface

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:          Leading zeros ARE NECESSARY if the command is not a full
                16 bits (e.g. ACCESS)
                Only valid with JTAG attached chips
```

```
Example:        sendcmd pu 12 C00008 -p0,1
```

Examples:

3.4 Miscellaneous Commands

3.4.1 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#] [-s#] [-p#]
- Display all rings available for chip

Example: `ecmdquery rings pu -p0,1`

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 Simulation Commands

3.5.1 simaet

Syntax:

eCMD Command Line Interface

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:

3.5.2 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:

3.5.3 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:

3.5.4 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:

3.5.5 simexit

Syntax:

Syntax: simexit

ECMD: Core Common Function

Function: Close down a simulation

Parameters:

Example: simexit

Examples:

3.5.6 simEXPECTFAC

Syntax:

eCMD Command Line Interface

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:

3.5.7 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:

3.5.8 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

data Data for expect

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 the input data: default 'xr'

Example: simexpecttcfac TITAN.TCKFREQ F

Examples:

3.5.9 simgetcurrentcycle

Syntax:

Syntax: simgetcurrentcycle

ECMD: Core Common Function

Function: Retrieve the current cycle count

Parameters:

Example: simgetcurrentcycle

Examples:

3.5.10 simGETFAC

Syntax:

eCMD Command Line Interface

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:

3.5.11 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:

3.5.12 simGETFACX

Syntax:

eCMD Command Line Interface

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:

3.5.13 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 - 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'

Example: `simgettcfac TITAN.TCKFREQ`

Examples:

3.5.14 siminit

Syntax:

eCMD Command Line Interface

Syntax: `siminit [<checkpoint>]`

ECMD: Core Common Function

Function: Initialize the simulation

Parameters:

`checkpoint[opt]` Name of checkpoint to load

Example: `siminit`
`siminit boot`

Examples:

3.5.15 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 : Default "xr"

`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.

Example: `simPUTFAC 100 C 4`

Examples:

3.5.16 simPUTFACS

Syntax:

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

ECMD: Core Common Function

Function: Write a simulation facility using a name

eCMD Command Line Interface

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:

3.5.17 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:

3.5.18 simputtcfac

Syntax:

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

ECMD: Core Common Function

eCMD Command Line Interface

```
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:

3.5.19 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:

3.5.20 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
```

eCMD Command Line Interface

```
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:

3.5.21 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:

3.5.22 simstktcfac

Syntax:

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

```
ECMD:           Core Common Function
Function:        Stick a TCFAC Facility
Parameters:
```

eCMD Command Line Interface

```
-----
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:

3.5.23 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:

3.5.24 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:

3.5.25 simUNSTICK

Syntax:

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:

3.5.26 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:

3.5.27 simunsticktcfac

Syntax:

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

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: