

eCMD Command Line Interface

Version .3

Contact: Chris Engel / Paul Prah

IBM Confidential

Table of Contents

1 Introduction.....	4
2 Usage Instructions.....	4
2.1 Environment Setup.....	4
2.2 Error Handling.....	4
2.3 Required Input Files.....	4
2.4 Optional Arguments.....	4
3 eCMD Common Commands.....	5
3.1 Common Command Arguments.....	5
3.1.1 Targeting Options.....	5
3.1.2 Data Output Formatting (-o<format>).....	5
3.1.3 Data Input Formatting (-i<format>).....	8
3.1.4 Data Input Bit Modifiers (-b<modifier>).....	8
3.2 Command Help (-h).....	8
3.3 Chip Display/Alter Commands.....	9
3.3.1 checkrings.....	9
3.3.2 getarray.....	9
3.3.3 getbits.....	10
3.3.4 getcfam.....	11
3.3.5 getlatch.....	12
3.3.6 getringdump.....	13
3.3.7 getscom.....	15
3.3.8 getspy.....	16
3.3.9 pollscom.....	17
3.3.10 putarray.....	18
3.3.11 putbits.....	19
3.3.12 putcfam.....	20
3.3.13 putlatch.....	20
3.3.14 putpattern.....	21
3.3.15 putscom.....	22
3.3.16 putspy.....	23
3.3.17 sendcmd.....	24
3.4 Processor Functions.....	26
3.4.1 breakpoint.....	26
3.4.2 getfpr.....	26
3.4.3 getgpr.....	27
3.4.4 getspr.....	28
3.4.5 instruct.....	28
3.4.6 putfpr.....	29

3.4.7	putgpr.....	30
3.4.8	putspr.....	30
3.5	Memory Display/Alter Functions.....	32
3.5.1	getmemdma.....	32
3.5.2	getmemmemctrl.....	32
3.5.3	getmemproc.....	33
3.5.4	putmemdma.....	33
3.5.5	putmemmemctrl.....	34
3.5.6	putmemproc.....	35
3.6	Miscellaneous Commands.....	36
3.6.1	ecmdquery.....	36
3.7	Simulation Commands.....	37
3.7.1	simaet.....	37
3.7.2	simcheckpoint.....	38
3.7.3	simclock.....	38
3.7.4	simecho.....	38
3.7.5	simexit.....	39
3.7.6	simEXPECTFAC.....	39
3.7.7	simexpecttcfac.....	40
3.7.8	simgetcurrentcycle.....	40
3.7.9	simGETFAC.....	41
3.7.10	simGETFACX.....	41
3.7.11	simgettcfac.....	42
3.7.12	siminit.....	42
3.7.13	simPUTFAC.....	42
3.7.14	simPUTFACX.....	43
3.7.15	simputtcfac.....	44
3.7.16	simrestart.....	44
3.7.17	simSTKFAC.....	44
3.7.18	simstktcfac.....	45
3.7.19	simSUBCMD.....	46
3.7.20	simtckinterval.....	46
3.7.21	simUNSTICK.....	46
3.7.22	simunsticktcfac.....	47

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      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      000000000 000000000 000000000
gr      k0:n0:s0:p01:c0      000000000 000000000 000000000
gr      k0:n0:s0:p02:c0      000000000 000000000 000000000

```

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: 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

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
000000000000000000000000000000 00000000000000000000000000000000
gr      k0:n0:s0:p01:c0      00000000000000000000000000000000
000000000000000000000000000000 00000000000000000000000000000000
```

Binary Word Columns : -oBW1

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

      0          1          2          3
      01234567890123456789012345678901
0: 0000000000000000000000000000000000000000000000000000000000000000
1: 0000000000000000000000000000000000000000000000000000000000000000
2: 0000000000000000000000000000000000000000000000000000000000000000
```

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 sgxbs ...
Performing 0's test on sgxbs ...
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 '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:      > 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. 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

-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)
0b111111101110110110111101110111
ecmd.exe getbits test idreg 0 32

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

3.3.4 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:

eCMD Command Line Interface

```
-----
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:

3.3.5 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 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.
```

eCMD Command Line Interface

```
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.
                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
                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.PFPU.FP_UNIT0.FP_REG -ox -exact -compress
-expect feed -ix
```

Examples:

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

3.3.6 getringdump

Syntax:

Syntax: getringdump <ChipSelect> <RingName1> [<RingName2> ...]

eCMD Command Line Interface

[-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 <code>ecmdquery</code> 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:

```
> getringdump test sgxbs
test    k0:n0:s0:p00
*****
* ECMD Dump scan ring contents, Tue Nov 25 12:58:44 2003
* Position 0:0, test sgxbs Ring
* Chip EC 9999
* Ring length: 573 bits
USE_GB_X.CHANNELINL2(0:150) 0xAAAA000000000000000000000000000000000000000000000000000000000000
USE_ENT_IOS.ERR_SET_GX_BOUNDARY.RECEIVER_LATCH.L2 0b0
USE_ENT_IOS.ERR_SET_GX_BOUNDARY.DRIVER_LATCH.L2 0b0
USE_ENT_IOS.ERR_SET_GX_BOUNDARY.ENABLE_LATCH.L2 0b0
USE_ENT_IOS.SYSTEM_ERR_BOUNDARY.RECEIVER_LATCH.L2 0b0
USE_ENT_IOS.SYSTEM_ERR_BOUNDARY.DRIVER_LATCH.L2 0b0
USE_ENT_IOS.SYSTEM_ERR_BOUNDARY.ENABLE_LATCH.L2 0b0
.....
USE_ENT_IOS.DBUGX_OUT_123_BOUNDARY.MXM.BDYRML.L2Q 0b0
USE_ENT_IOS.DBUGX_OUT_123_BOUNDARY.MXM.BYDML.L2Q 0b0
USE_ENT_IOS.DBUGX_OUT_123_BOUNDARY.MXM.BDYEM.L2Q 0b0
USE_GB_X.CHANNELINL2(151:247) 0x0000000000000000000000000000000000000000000000000000000000000000
ACCESS.SNPBUF 0b0
ecmd.exe getringdump test sgxbs
```

3.3.7 getscom

Syntax:

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

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] 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

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

3.3.8 getspy

Syntax:

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

-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:

3.3.9 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:

```
-----
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.

-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.
                  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.

-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:      pollscm pu 800009 -exp feed000000000001 -limit 30s -interval 10s
-verbose -pl   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
```

3.3.10 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 'x1'
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 xgpr0 deadbeef80000000 -p0,1 -c1

Examples:**3.3.11 putbits****Syntax:**

Syntax: putbits <ChipSelect> <RingName> <StartPos> <Data> [-i<format>] [-b<modifier>]
 [-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.
 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

-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

Examples:

3.3.12 putcfam

Syntax:

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

ECMD: Core Common Function

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 'x1'
 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

3.3.13 putlatch

Syntax:

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

eCMD Command Line Interface

concatenated to form one complete register.

Parameters:

ChipSelect	Chip to put data to.
RingName	Specifies chip ring to operate on. For a list of available rings, use the <code>ecmdquery</code> command.
	Ex: <code>ecmdquery rings memctrl</code> <code>ecmdquery rings pu</code>
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 <code>-i<format></code>
<code>-i<format></code>	[opt] Specifies the format type of input data : default 'xl' Run ' <code>ecmdquery formats</code> ' to view available formats
<code>-k#</code>	[optional] Specify which cage to act on (0 is default). Specify <code>-kall</code> to act on all cages.
<code>-n#</code>	[optional] Specify which node to act on (0 is default). Specify <code>-nall</code> to act on all nodes.
<code>-s#</code>	[optional] Specify which slot to act on (0 is default). Specify <code>-sall</code> to act on all slots.
<code>-p#</code>	[optional] Specify which chip position to act on (0 is default). Specify <code>-pall</code> to act on all chips.
<code>-c#</code>	[optional] Specify which processor core to act on (0 is default). Specify <code>-call</code> to act on all cores.

Example: `putlatch pu cp fpu GCP.PFPU.FP UNIT0.FPA LATCH -ix feed`

Examples:

3.3.14 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

eCMD Command Line Interface

```

                                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:

3.3.15 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.
                                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.
```

eCMD Command Line Interface

```
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:

3.3.16 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

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
```

eCMD Command Line Interface

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

Examples:

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

eCMD Command Line Interface

Example: sendcmd pu 12 C00008 -p0,1

Examples:

3.4 Processor Functions

3.4.1 breakpoint

Syntax:

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

```

ECMD:          Core Common Function

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:      > breakpoint set IABR 8000000000FAC230

```

3.4.2 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)

```

IBM Confidential

Page 27

04/16/04 11:26:39 am

Syntax:

IBM Confidential

Page 27

04/16/04 11:26:39 am

IBM Confidential

Page 27

04/16/04 11:26:39 am

IBM Confidential

Page 27

04/16/04 11:26:39 am

IBM Confidential

Page 27

04/16/04 11:26:39 am

IBM Confidential

eCMD Command Line Interface

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

3.4.4 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.  
  
-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 -p0,1
```

3.4.5 instruct

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

ECMD: Core Common Function

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

Parameters:

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

eCMD Command Line Interface

steps [opt] Number of steps to execute

-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: > instruct step 10 -p2 -c1

3.4.6 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.

eCMD Command Line Interface

-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

3.4.7 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

3.4.8 putspr

Syntax:

Syntax: putspr <SprName> [<Start> <Numbits>] <Data> [-i<format>]

eCMD Command Line Interface

```

                                [-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:
-----
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
```

3.5 Memory Display/Alter Functions

3.5.1 getmemdma

Syntax:

Syntax: getmemdma <MemAddress> <NumBytes> [-k#] [-n#] [-o<format>]

```

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'
                  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:       > getmemdma 1000 128 -p0
    
```

3.5.2 getmemmemctrl

Syntax:

Syntax: getmemmemctrl <MemAddress> <NumBytes> [-k#] [-n#] [-s#] [-p#] [-o<format>]

```

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'
                  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.

Examples: > getmemmemctrl 1000 128 -p0

3.5.3 getmemproc

Syntax:

Syntax: putmemproc <MemAddress> <Data> | -f[d|b]<filename> [-i<format>]
[-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)

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

-fb <filename> Specify full path and filename to binary file to load

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: > putmemproc 10000 deadbeef80000000 -p0,1

3.5.4 putmemdma

Syntax:

Syntax: putmemdma <MemAddress> <Data> | -f[d|b]<filename> [-i<format>] [-k#] [-n#]

ECMD: Core Common Function

eCMD Command Line Interface

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

Parameters:
-----
MemAddress     64 Bit address to write to (Hex-Right)

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

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
```

3.5.5 putmemmemctrl

Syntax:

```
Syntax: putmemmemctrl <MemAddress> <Data> | -f[d|b]<filename> [-i<format>]
                                             [-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)

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

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.
```

eCMD Command Line Interface

-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

3.5.6 putmemproc

Syntax:

Syntax: putmemproc <MemAddress> <Data> | -f[d|b]<filename> [-i<format>]
[-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)

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

-fb <filename> Specify full path and filename to binary file to load

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: > putmemproc 10000 deadbeef80000000 -p0,1

3.6 Miscellaneous Commands

3.6.1 ecmdquery

Syntax:

Syntax: `ecmdquery <Mode> [Mode Options]`

ECMD:	Core Common Function
Function:	Query information from eCMD
Parameters:	

Mode	Query type to perform
Mode Values	
	<code>rings ChipSelect [-k#] [-n#] [-s#] [-p#]</code> - Display all rings available for chip
	<code>formats</code> - Display help text for all available input/output formats
	<code>chips [-ep]</code> - Display all the chips in the system - Use '-ep' to display in an easier to parse format
	<code>chipinfo ChipSelect [-k#] [-n#] [-s#] [-p#]</code> - Display info about a particular chip (ex. EC level)
	<code>version</code> - Display version info about the eCMD Instance you are running

Example:	<code>ecmdquery rings pu -p0,1</code> <code>ecmdquery formats</code>

Examples:

eCMD Command Line Interface

```
> 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
```

3.7 Simulation Commands

3.7.1 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:

3.7.2 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:

3.7.3 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:

3.7.4 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:

3.7.5 simexit

Syntax:

Syntax: simexit

```
ECMD:          Core Common Function  
Function:       Close down a simulation  
Parameters:-----  
-----
```

```
Example:      simexit
```

Examples:

3.7.6 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:

3.7.7 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

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:

3.7.8 simgetcurrentcycle

Syntax:

Syntax: `simgetcurrentcycle`

ECMD: Core Common Function

Function: Retrieve the current cycle count

Parameters:

Example: `simgetcurrentcycle`

Examples:

3.7.9 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

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:

3.7.10 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: 'X'

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:

3.7.11 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'
Run 'ecmdquery formats' to view available formats

Example: `simgettcfac TITAN.TCKFREQ`

Examples:

3.7.12 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:

3.7.13 simPUTFAC

Syntax:

eCMD Command Line Interface

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 read

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:

3.7.14 simPUTFACX

Syntax:

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

ECMD: Core Common Function

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

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:

3.7.15 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:

3.7.16 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:

3.7.17 simSTKFAC

Syntax:

eCMD Command Line Interface

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:

3.7.18 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
Format specified by -i<format>

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

Examples:

3.7.19 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:

3.7.20 simtckinterval

Syntax:

Syntax: simtckinterval <interval>

ECMD: Core Common Function

Function: Adjust the TCK Interval

Parameters:

interval New Interval

Example: simtckinterval 18

3.7.21 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

eCMD Command Line Interface

row [optional] Facility row
offset [opt] Facility offset

Example: simUNSTICK TITAN.TCKFREQ 4

Examples:

3.7.22 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
 Format specified by -i<format>

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: simunsticktcfac TITAN.TCKFREQ F

Examples: