Open GENIE GCL for Instruments

Draft Version, July 2007

Generated by Doxygen 1.6.1

Sun May 9 00:54:00 2010

Contents

1 Open GENIE GCL for Instruments **1**

1.1 Introduction . . . 1

1.2 Topics Covered . . . 1

1.3 Additional Information . . . 2

2 Graphics and Data Display **2**

3 Instrument Scripting and Control **2**

4 The Open GENIE language **2**

4.1 Introduction . . . 2

5 Open GENIE Doxygen Syntax **3**

5.1 Introduction . . . 3

5.2 Syntax . . . 3

6 "Data access and File I/O" **4**

7 Module Documentation **4**

7.1 Base Primitives . . . 4

7.1.1 Detailed Description . . . 4

7.1.2 Function Documentation . . . 4

7.2 Operating system interacion . . . 10

7.3 Output operations . . . 10

7.3.1 Detailed Description . . . 11

7.3.2 Function Documentation . . . 11

7.4 Input operations . . . 14

7.4.1 Detailed Description . . . 14

7.4.2 Function Documentation . . . 14

7.5 Data type convertion. . . . 14

7.5.1 Detailed Description . . . 15

7.5.2 Function Documentation . . . 15

7.6 Commands for data display. . . . 15

7.6.1 Detailed Description . . . 16

7.6.2 Function Documentation . . . 16

7.7 Miscellaneous commands . . . 17

7.7.1 Function Documentation . . . 17

7.8 Instrument Control. 17
7.8.1 Detailed Description 18
7.9 Setup for instrument control. 18
7.9.1 Detailed Description 18
7.9.2 Function Documentation 18
7.10 Data Acquisition Control. 19
7.10.1 Detailed Description 20
7.10.2 Function Documentation 20
7.11 High level instrument control. 34
7.11.1 Detailed Description 34
7.11.2 Function Documentation 34
7.12 Low Level LabVIEW Control. 36
7.12.1 Detailed Description 37
7.12.2 Function Documentation 37

8 Example Documentation **41**
8.1 abort_handler.gcl 41
8.2 cphs100.gcl 42
8.3 drange_2_2.gcl 42
8.4 hardware_periods.gcl 42
8.5 paths.gcl 42
8.6 pg002.gcl 42
8.7 set_pressure.gcl 42
8.8 simple.gcl 42
8.9 sxd_scan.gcl 42

1 Open GENIE GCL for Instruments

1.1 Introduction

If you are already familiar with Open GENIE syntax, you should only need to read the description of the **Open GENIE Doxygen syntax** (p. 3) which will explain how these generated pages relate to the usual Open GENIE syntax. If you are new to Open GENIE, you should first read the **Open GENIE language** (p. 2).

1.2 Topics Covered

The main topics covered here will be:

- **The Open GENIE language** (p. 2)

- **Open GENIE syntax in Doxygen** (p. 3)
- **Instrument Control and Scripting** (p. 2)
- **Data access and File I/O** (p. 4)
- **Graphics and Data Display** (p. 2)

In addition a full list of commands and their descriptions is included in the manual

1.3 Additional Information

The latest version of this manual is available in PDF format on the web at `http://download.opengenie.org/doc/pdf/manual_brief.pdf` and a searchable html version is browsable at `http://download.opengenie.org/doc/html_brief/`

2 Graphics and Data Display

Details about scripting and control stuff **Commands used for data display** (p. 15)

3 Instrument Scripting and Control

Groups of individual commands can be combined together for easy execution or to create new commands.

These combinations are usually called script files.

Script files are loaded into GENIE via the **LOAD** (p. 7) command. If the file contains a list of commands, these are executed sequentially as the script is loaded. If the file contains definitions for new commands using the PROCEDURE / ENDPROCEDURE keywords then these new commands are compiled into GENIE memory and made available for execution.

Several example files of scripts made from **instrument control commands** (p. 17) are available via the **Example** tab above.

- The **simple** (p. **??**) script just uses **CSET** (p. 34) and **WAITFOR** (p. 33)
- The more advanced **sxd_scan** (p. **??**) script writes a custom PROCEDURE to control a VI using **SETLABVIEWVAR** (p. 39)
- **abort_handler::gcl** (p. **??**) is an example of using **SET_ABORT_HANDLER** (p. 9) to control Ctrl-C behaviour in scripts

4 The Open GENIE language

4.1 Introduction

The GENIE command style is very similar to the VMS operating system i.e.

```
COMMAND/QUALIFIER value1 value2 keyword1=value3 keyword2=value4
```

For example

```
CHANGE title="new title" user="new user"
CSET /WAIT temp1=4.0
```

There are a few important differences from VMS however. Character strings must always be included in double quotation marks i.e. "string"; this is to distinguish them from words or function names that form part of the GENIE language and so make interpreting the language easier to program.

For example, on VMS you could type

```
CHANGE title something ! Works on old VMS system only
```

This was possible because a separate program interpreted each command: with the GENIE processor we need to tell it that title and something are not variable names. Being more explicit in this way allows more complicated expressions to be written in the language. For example in GENIE you can change both the title and user in one command as shown above.

In moving commands from VMS to the PC, command will be similar except

- **COMMAND NAME VALUE** will usually become **COMMAND NAME=VALUE**
- Any string values must be quoted i.e. **COMMAND NAME="StringValue"**

5 Open GENIE Doxygen Syntax

5.1 Introduction

We are now using Doxygen to document the Open GENIE GCL language. As Doxygen does not understand GCL directly, it is first pre-processed into C language like code which is then parsed by Doxygen. A side effect of this is that the documentation produced for the commands is written out in a C function like syntax, but it is easy enough to relate this back to more familiar GCL syntax.

5.2 Syntax

In OpenGENIE a command qualifier and a boolean parameter are equivalent, so if a COMMAND can take two qualifier /QUAL1 and /QUAL2 then writing

```
COMMAND /QUAL1
```

is equivalent to

```
COMMAND QUAL1=$TRUE QUAL2=$FALSE
```

In the function documentation, qualifiers are listed as if they were function arguments and are given the data type Qualifier. For example see **ISC_SETUP** (p. 18). If the parameters for a function must always be of a specific type e.g. Integer or String, this is also listed. If the parameter may be of differnt type depending on the usage of the command, then its type will be listed as Generic and you will need to refer to the documentation for valid choices.

All open GENIE procedures can be called in command or function syntax i.e.

```
COMMAND /QUAL1 4.0
```

or

```
RESULT = COMMAND:QUAL1(4.0)
```

In the documentation only the function syntax is referred to, but that does not mean the command syntax cannot be used.

6 "Data access and File I/O"

stuff about data access

7 Module Documentation

7.1 Base Primitives

Low level GENIE operations.

Functions

- **BUILD_ARRAY** (Generic V1, Generic V2, Generic V3, Generic V4, Generic V5, Generic V6, Generic V7, Generic V8, Qualifier DCOM)
- **DEFINED** (Generic VAR)
- **DIMENSIONALITY** (Generic ARRAY)
- **DIMENSIONS** (Integer DIM1, Integer DIM2, Integer DIM3, Integer DIM4, Integer DIM5, Integer DIM6, Integer DIM7, Integer DIM8, Integer DIM9, Integer DIM10)
- **EXISTS** (String VAR, Qualifier PROC)
- **FIELDS** (Integer NUMFIELDS)
- **FILL** (Array ARRAY, Generic VALUE, Generic STEP, Qualifier GEOMETRIC)
- **IS_A** (Generic VAR, String TYPESTRING)
- **LOAD** (String FILENAME)
- **NOW** (Qualifier SEC, Qualifier SYSTEM, Qualifier MILLISEC, Qualifier ISO, Qualifier ISOBASIC)
- **SAVE** (String FILENAME)
- **SEARCHPATH** (String NAME, String PATH1, String PATH2, String PATH3, String PATH4, String PATH5, Stringarray PATHS, Qualifier APPEND, Qualifier PREPEND, Qualifier DEFAULT, Qualifier SHOWALL)
- **SET_ABORT_HANDLER** (String PROC, Qualifier DEFAULT)
- **SLEEP** (Integer TIME, Qualifier MILLISEC)
- **TIME_FROM_ISOTIME** (Stringarray ISOTIME)
- **TRANSLATE_PATH** (String PATH, Qualifier WRITE, Qualifier VERBOSE, Qualifier NOMESSAGE)

7.1.1 Detailed Description

Low level GENIE operations.

7.1.2 Function Documentation

7.1.2.1 BUILD_ARRAY (Generic *V1*, Generic *V2*, Generic *V3*, Generic *V4*, Generic *V5*, Generic *V6*, Generic *V7*, Generic *V8*, Qualifier *DCOM*)

Build an array from parameter list.

Parameters:

← ***V1*** 1st value

← ***V2*** 2nd Value

← ***DCOM*** (optional) Build a dcom array

Returns:

Array of values, Workspacearray for *DCOM*

Warning:

all parameters must be of the same data type unless *DCOM* specified

7.1.2.2 DEFINED (Generic *VAR*)

Check if a variable is defined.

Parameters:

← ***VAR*** Variable to check

Returns:

Boolean $TRUE or $FALSE

7.1.2.3 DIMENSIONALITY (Generic *ARRAY*)

Return dimensionality of an Array.

Parameters:

← ***ARRAY*** Input Array object

Returns:

Integerarray containing Array dimensions

See also:

DIMENSIONS (p. 5)

7.1.2.4 DIMENSIONS (Integer *DIM1*, Integer *DIM2*, Integer *DIM3*, Integer *DIM4*, Integer *DIM5*, Integer *DIM6*, Integer *DIM7*, Integer *DIM8*, Integer *DIM9*, Integer *DIM10*)

Create an Array of a given size, but unspecified type. The Array will take on a type as soon as one of its elements is assigned to.

Parameters:

← ***DIM1*** First array dimension

← ***DIM2*** Second array dimension

← ***DIM3*** Third array dimension

Returns:

Allocated, but uninitialised, Array object

See also:

FILL (p. 7)

7.1.2.5 EXISTS (String *VAR*, Qualifier *PROC*)

Check if a global variable of a given name exists. Note that you cannot type **EXISTS** (p. 6):VAR("w1.x") as w1.x is not a variable. Instead you need to split the operation into two parts and use **EXISTS** (p. 6)("w1") to check for the workspace and FIELD_PRESENT("w1","x") to check for the workspace field.

Parameters:

VAR Object name to check

PROC Treat name as a PROCEDURE name rather than a variable name

Returns:

Boolean $TRUE or $FALSE

See also:

FIELD_PRESENT

7.1.2.6 FIELDS (Integer *NUMFIELDS*)

Create a Workspace object of the default or specified size. A workspace will automatically grow in size as more items are added, but it is more efficient to specify the correct size at creation.

Parameters:

← ***NUMFIELDS*** (optional) Initial number of field entries to create

Returns:

Empty Workspace object

7.1.2.7 FILL (Array *ARRAY*, Generic *VALUE*, Generic *STEP*, Qualifier *GEOMETRIC*)

Fill a Array with items of one type.

Parameters:

← ***ARRAY*** Array to fill
← ***VALUE*** value to assign to each element
← ***STEP*** (optional) Increment *VALUE* by this for each subsequent array index
← ***GEOMETRIC*** (optional) Fill array geometrically e.g. multiply rather than add *STEP*

7.1.2.8 IS_A (Generic *VAR*, String *TYPESTRING*)

Test if a variable is of a given type.

Parameters:

← ***VAR*** The variable to test
← ***TYPESTRING*** Type to check for (boolean, real, integer, string, realarray, integerarray, interval)

Returns:

Boolean

7.1.2.9 LOAD (String *FILENAME*)

Load files containing GENIE comamnds into memory.

Parameters:

← ***FILENAME*** file containing commands/procedures

7.1.2.10 NOW (Qualifier *SEC*, Qualifier *SYSTEM*, Qualifier *MILLISEC*, Qualifier *ISO*, Qualifier *ISOBASIC*)

Print current time.

Parameters:

← ***SEC*** Time in seconds since midnight
← ***SYSTEM*** System time (number of seconds since 1-JAN-1970
← ***MILLISEC*** millisecond time
← ***ISO*** ISO8601 extended format (YYYY-MM-DDTHH:MM:SS)
← ***ISOBASIC*** ISO8601 basic format (YYYYMMDDTHHMMSS)

Returns:

time

7.1.2.11 SAVE (String *FILENAME*)

Save a snapshot of a session. The snapshot contains all variable values.

Parameters:

← ***FILENAME*** Name of output image file

7.1.2.12 SEARCHPATH (String *NAME*, String *PATH1*, String *PATH2*, String *PATH3*, String *PATH4*, String *PATH5*, Stringarray *PATHS*, Qualifier *APPEND*, Qualifier *PREPEND*, Qualifier *DEFAULT*, Qualifier *SHOWALL*)

Define a search path (like a VMS logical name).

Parameters:

← ***NAME*** name of path to define

← ***PATH1*** First directory/path to associate with *NAME*

← ***PATH2*** Second directory/path to associate with *NAME*

← ***PATHS*** array of paths to associate with *NAME*; alternative to specifying PATH1, PATH2 etc. *NAME*

← ***DEFAULT*** modify the default path (known as $PATH)

← ***SHOWALL*** List contents of path *NAME*

← ***APPEND*** append values to path *NAME*

← ***PREPEND*** prepend values to path *NAME*

Returns:

Stringarray of paths in *NAME*

Searchpaths are lists of directories and allow files to be located easily via the syntax "SOME_-NAME:file.dat" The path is expanded automatically by commands such as **GET** (p. 14) using the **TRANSLATE_PATH** (p. 10) command, which can also be used separately to test paths. Searchpaths can reference other searchpaths and are translated recursively. If you wish to reference another searchpath remember to add a : or it will be treated as a normal directory name. For example

- SEARCHPATH "place1" "c:\\place1" "c:\\place1_old"
- SEARCHPATH "mydata" "place1:" "c:\\data2" "$archive:"

There are some special pre-defined searchpaths:

- $DISKDIR is the directory specified by the **SET** (p. **??**)/DISK and **SET** (p. **??**)/DIR commands
- $ARCHIVE is the ISIS data archive ("\\isis\inst$" etc. and "data.isis.rl.ac.uk")
- $PATH is the directories used for filename generated by the **ASSIGN** (p. **??**) command

filenames generated by **ASSIGN** (p. **??**) will be named "$PATH:file.ext". The default value of $PATH is to search the current directory, $DISKDIR and finally $ARCHIVE. Use **SEARCHPATH** (p. 8)/DEFAULT to set the value of $PATH. To see the contents of all paths use **SEARCHPATH** (p. 8)/SHOWALL or for individul paths

- PRINTN SEARCHPATH("mydata") to show path "mydata"
- PRINTN SEARCHPATH:DEFAULT() to show $PATH
- SEARCHPATH/DEFAULT "c:\\data" "$diskdir:" "$archive:"

See also:

TRANSLATE_PATH (p. 10)

7.1.2.13 SET_ABORT_HANDLER (String *PROC*, Qualifier *DEFAULT*)

Define a procedure to be called in the event of a Control-C being received. This procedure is called before a user prompt is returned to the command line and additional control-C presses are ignored during its execution. The procedure remains the active handler until it is called, after which it will need to be readded by calling **SET_ABORT_HANDLER** (p. 9) again. It is possible for the handler to re-register itself while it is executing if this behaviour is desired.

Parameters:

← ***PROC*** name of procedure to call

← ***DEFAULT*** reset handler to the default (which does nothing additional)

7.1.2.14 SLEEP (Integer *TIME*, Qualifier *MILLISEC*)

Wait a specified amount of real time before continuing.

Parameters:

← ***TIME*** Length of time to wait (in seconds unless */MILISEC* is specified)

← ***MILLISEC*** Interpret *TIME* as milli-seconds rather than seconds

7.1.2.15 TIME_FROM_ISOTIME (Stringarray *ISOTIME*)

Convert an array of ISO format time string into a plottable value.

Parameters:

← ***ISOTIME*** Stringarray of time values

Returns:

Realarray of time values

7.1.2.16 TRANSLATE_PATH (String *PATH*, Qualifier *WRITE*, Qualifier *VERBOSE*, Qualifier *NOMESSAGE*)

Translate a path containing logical names.

Parameters:

← ***PATH*** path to translate

← ***WRITE*** check for write access to file. Default is for read access.

← ***VERBOSE*** produce more output of path translation.

Returns:

translated path

This command translates paths of the form "SOME_NAME:file.dat". If SOME_NAME has been defined as a searchpath using the **SEARCHPATH** (p. 8) command then it is translated accordingly.

See also:

SEARCHPATH (p. 8)

7.2 Operating system interacion

These command interafct with the operating system and return information back to GENIE. These command interafct with the operating system and return information back to GENIE.

7.3 Output operations

Commands to output data from GENIE to e.g.

Functions

- **CFN** (Generic P1, Integer NUMBER, String NAME, Qualifier DISKDIR)
- **FILETYPE** (String FILE)
- **PRINT** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20)
- **PRINTE** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20, Qualifier NOLOCATE)
- **PRINTEN** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20, Qualifier NOLOCATE)
- **PRINTI** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20)
- **PRINTIN** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20)

- **PRINTN** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Generic P10, Generic P11, Generic P12, Generic P13, Generic P14, Generic P15, Generic P16, Generic P17, Generic P18, Generic P19, Generic P20)

7.3.1 Detailed Description

Commands to output data from GENIE to e.g. Files

7.3.2 Function Documentation

7.3.2.1 CFN (Generic *P1*, Integer *NUMBER*, String *NAME*, Qualifier *DISKDIR*)

Construct a full filename from various information.

Parameters:

← ***P1*** Either a NAME or NUMBER (see below)

← ***NUMBER*** A run number

← ***NAME*** A file name

← ***DISKDIR*** Append $DISKDIR rather than $PATH to names

Returns:

Full file name

7.3.2.2 FILETYPE (String *FILE*)

Determine the type of a file.

Parameters:

← ***FILE*** Name of file

Returns:

File type as string

7.3.2.3 PRINT (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*)

The **PRINT** (p. 11) and **PRINTN** (p. 13) commands differ only in that **PRINTN** (p. 13) adds a trailing newline where **PRINT** (p. 11) does not. They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

Parameters:

← ***P1*** First item to print (any object type)

← ***P2*** Second item to print (any object type)

See also:

PRINTN (p. 13) **PRINTEN** (p. 12)

7.3.2.4 PRINTE (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*, Qualifier *NOLOCATE*)

Print an error (usually red) message to the screen. The **PRINTE** (p. 12) and **PRINTEN** (p. 12) commands differ only in that **PRINTEN** (p. 12) adds a trailing newline where **PRINTE** (p. 12) does not. They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

Parameters:

← ***P1*** First item to print (any object type)

← ***P2*** Second item to print (any object type)

See also:

PRINTN (p. 13) **PRINTIN** (p. 13)

7.3.2.5 PRINTEN (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*, Qualifier *NOLOCATE*)

Parameters:

P1 Print an error (usually red) message to the screen with a trailing newline. Print an error (usually red) message to the screen.

The **PRINTE** (p. 12) and **PRINTEN** (p. 12) commands differ only in that **PRINTEN** (p. 12) adds a trailing newline where **PRINTE** (p. 12) does not. They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

7.3.2.6 PRINTI (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*)

Print an informational (usually blue) message to the screen. The **PRINTI** (p. 12) and **PRINTIN** (p. 13) commands differ only in that **PRINTIN** (p. 13) adds a trailing newline where **PRINTI** (p. 12) does not. They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

Parameters:

← ***P1*** First item to print (any object type)

← ***P2*** Second item to print (any object type)

Warning:

The message will not appear on the screen if informational message printing has been disabled with the **TOGGLE** (p. 17) command.

See also:

PRINTN (p. 13) **PRINTEN** (p. 12)

7.3.2.7 PRINTIN (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*)

Print an informational (usually blue) message to the screen with a trailing newline. Print an informational (usually blue) message to the screen.

The **PRINTI** (p. 12) and **PRINTIN** (p. 13) commands differ only in that **PRINTIN** (p. 13) adds a trailing newline where **PRINTI** (p. 12) does not. They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

Parameters:

← ***P1*** First item to print (any object type)

← ***P2*** Second item to print (any object type)

Warning:

The message will not appear on the screen if informational message printing has been disabled with the **TOGGLE** (p. 17) command.

See also:

PRINTN (p. 13) **PRINTEN** (p. 12)

7.3.2.8 PRINTN (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Generic *P10*, Generic *P11*, Generic *P12*, Generic *P13*, Generic *P14*, Generic *P15*, Generic *P16*, Generic *P17*, Generic *P18*, Generic *P19*, Generic *P20*)

Print something to the screen with a trailing newline. The **PRINT** (p. 11) and **PRINTN** (p. 13) commands differ only in that **PRINTN** (p. 13) adds a trailing newline where **PRINT** (p. 11) does not.

They both accept up to 20 parameters labeled P1 to P20, but only P1 and P2 are documented below.

Parameters:

← ***P1*** First item to print (any object type)

← ***P2*** Second item to print (any object type)

See also:

PRINTN (p. 13) **PRINTEN** (p. 12)

7.4 Input operations

Commands to input data into GENIE from e.g.

Functions

- **GET** (Generic ITEM, String FILE, Generic ITEM_ARGS, String HOST, Qualifier ARRAY, Qualifier REMOTE, Qualifier NEWICP)

7.4.1 Detailed Description

Commands to input data into GENIE from e.g. Files

7.4.2 Function Documentation

7.4.2.1 GET (Generic *ITEM*, String *FILE*, Generic *ITEM_ARGS*, String *HOST*, Qualifier *ARRAY*, Qualifier *REMOTE*, Qualifier *NEWICP*)

Read a spectrum from a data source.

Parameters:

← ***ITEM*** Item to read (either a String or an Integer)

← ***FILE*** Source to read item from

← ***ITEM_ARGS*** ???

← ***HOST*** Host to read item from (default: localhost)

← ***ARRAY*** Return a Workspacearray for multiple spectra rather than a multi-dimensional y array

← ***REMOTE*** Use remote access via DCOM

← ***NEWICP*** Connect to a New style ICP program rather than a remote GENIE

7.5 Data type convertion.

Functions for converting one variable into another type.

Functions

- **AS_STRING** (Generic VAR, String FORMAT)
- **FLOAT_AS_STRING** (Real VAR, Integer DP)

7.5.1 Detailed Description

Functions for converting one variable into another type.

7.5.2 Function Documentation

7.5.2.1 AS_STRING (Generic *VAR*, String *FORMAT*)

Convert a variable into a String.

Parameters:

← ***VAR*** Variable to convert

← ***FORMAT*** (optional) printf style format string

Returns:

converted value

See also:

FLOAT_TO_STRING

7.5.2.2 FLOAT_AS_STRING (Real *VAR*, Integer *DP*)

Convert a Real variable into a String with a given precision.

Parameters:

← ***VAR*** Variable to convert

← ***DP*** Number of decimal places to keep

Returns:

VAR convered to a String

See also:

AS_STRING (p. 15)

7.6 Commands for data display.

This group contains all commands used for data display.

Functions

- **ALTER** (Generic P1, Generic P2, Generic P3, Generic P4, Real XMIN, Real XMAX, Real YMIN, Real YMAX, Qualifier BINNING, Qualifier MARKERS, Qualifier PLOT, Qualifier DEVICE, Qualifier HARDCOPY, Qualifier TEXTHEIGHT, Qualifier FONT, Qualifier LINETYPE, Qualifier TEXTCOLOUR, Qualifier LINECOLOUR, Qualifier PLOTCOLOUR, Qualifier SIZE, Qualifier

LINEWIDTH, Qualifier MARKERSIZE, Qualifier STATUS, Qualifier BACKGROUND, Qualifier FOREGROUND, Qualifier CURSORFORMAT)

- **DISPLAY** (Generic P1, Generic P2, Generic P3, Generic P4, Generic P5, Generic P6, Generic P7, Generic P8, Generic P9, Workspace WKSP, Real XMIN, Real XMAX, Real YMIN, Real YMAX, Colour PLOTCOLOUR, Real LINEWIDTH, Colour TEXTCOLOUR, Real TEXTHEIGHT, Real MIN, Real MAX, String TWODLABEL, Colour LINECOLOUR, Qualifier MARKERS, Qualifier LINE, Qualifier ERRORS, Qualifier HISTOGRAM, Qualifier SQRT, Qualifier LOG, Qualifier LINEAR, Qualifier SMOOTH, Qualifier RAW)
- **PLOT** (Workspace WKSP, Integer BINNING, Qualifier HISTOGRAM, Qualifier LINE, Qualifier MARKERS, Qualifier ERRORS)

7.6.1 Detailed Description

This group contains all commands used for data display.

7.6.2 Function Documentation

7.6.2.1 ALTER (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Real *XMIN*, Real *XMAX*, Real *YMIN*, Real *YMAX*, Qualifier *BINNING*, Qualifier *MARKERS*, Qualifier *PLOT*, Qualifier *DEVICE*, Qualifier *HARDCOPY*, Qualifier *TEXTHEIGHT*, Qualifier *FONT*, Qualifier *LINETYPE*, Qualifier *TEXTCOLOUR*, Qualifier *LINECOLOUR*, Qualifier *PLOTCOLOUR*, Qualifier *SIZE*, Qualifier *LINEWIDTH*, Qualifier *MARKERSIZE*, Qualifier *STATUS*, Qualifier *BACKGROUND*, Qualifier *FOREGROUND*, Qualifier *CURSORFORMAT*)

Change defaults used by **PLOT()** (p. 17) and **DISPLAY()** (p. 16) commands.

Parameters:

← ***BINNING*** Change default bin grouping used for plotting

← ***FONT*** Change default Font

← ***PLOT*** Plot

7.6.2.2 DISPLAY (Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, Generic *P5*, Generic *P6*, Generic *P7*, Generic *P8*, Generic *P9*, Workspace *WKSP*, Real *XMIN*, Real *XMAX*, Real *YMIN*, Real *YMAX*, Colour *PLOTCOLOUR*, Real *LINEWIDTH*, Colour *TEXTCOLOUR*, Real *TEXTHEIGHT*, Real *MIN*, Real *MAX*, String *TWODLABEL*, Colour *LINECOLOUR*, Qualifier *MARKERS*, Qualifier *LINE*, Qualifier *ERRORS*, Qualifier *HISTOGRAM*, Qualifier *SQRT*, Qualifier *LOG*, Qualifier *LINEAR*, Qualifier *SMOOTH*, Qualifier *RAW*)

Display a Workspace on the screen.

See also:

PLOT (p. 17)

7.6.2.3 PLOT (Workspace *WKSP*, Integer *BINNING*, Qualifier *HISTOGRAM*, Qualifier *LINE*, Qualifier *MARKERS*, Qualifier *ERRORS*)

Over Plot a Workspace on the screen.

Parameters:

← ***WKSP*** Input Workspace

← ***BINNING*** Bin grouping to apply to data

← ***HISTOGRAM*** plot a histogram

← ***MARKERS*** plot markers of the type defined by **ALTER** (p. 16)/MARKERS

See also:

DISPLAY (p. 16)

7.7 Miscellaneous commands

Functions

- **TOGGLE** (Boolean VAL, Qualifier STATUS, Qualifier LOGX, Qualifier LOGY, Qualifier CLEAR, Qualifier MODE, Qualifier CHAR, Qualifier HEADER, Qualifier RX, Qualifier RY, Qualifier INFO, Qualifier GRATICULE, Qualifier ON, Qualifier OFF, Qualifier NOMESSAGE, Qualifier DEBUG)

7.7.1 Function Documentation

7.7.1.1 TOGGLE (Boolean *VAL*, Qualifier *STATUS*, Qualifier *LOGX*, Qualifier *LOGY*, Qualifier *CLEAR*, Qualifier *MODE*, Qualifier *CHAR*, Qualifier *HEADER*, Qualifier *RX*, Qualifier *RY*, Qualifier *INFO*, Qualifier *GRATICULE*, Qualifier *ON*, Qualifier *OFF*, Qualifier *NOMESSAGE*, Qualifier *DEBUG*)

Change behaviour of various commands.

Parameters:

← ***LOGX*** Toggle between linear and logarithmic X axes

← ***ON*** Force option to be enabled rather than toggle the value

7.8 Instrument Control.

This group contains all commands used for instrument control.

Modules

- **Setup for instrument control.**
- **Data Acquisition Control.**
- **High level instrument control.**
- **Low Level LabVIEW Control.**

7.8.1 Detailed Description

This group contains all commands used for instrument control. This includes commands for control of the Data Acquisition system and of the LabVIEW VIs that interact with the Sample Environment.

7.9 Setup for instrument control.

Commands in this group configure setting within the running genie session to specify the names of Sample Environment blocks, DAE type etc.

Functions

- **GETBLOCKS** ()
- **ISC_SETUP** (String INSTRUMENT, Integer MODE, Qualifier NEWDAE, Qualifier REMOTE, Qualifier SCRIPT, Qualifier DIRECT)

7.9.1 Detailed Description

Commands in this group configure setting within the running genie session to specify the names of Sample Environment blocks, DAE type etc. The command usually only needs to be executed once per session or when something changes (e.g. new sample environment equipment is added)

7.9.2 Function Documentation

7.9.2.1 GETBLOCKS ()

Update GENIE session with sample environment block information on SECI. The command re-reads the list of sample environment parameters from SECI and must be re-run if a block is removed/added to SECI.

See also:

ISC_SETUP (p. 18) **Example**

```
GetBlocks
```

7.9.2.2 ISC_SETUP (String *INSTRUMENT*, Integer *MODE*, Qualifier *NEWDAE*, Qualifier *REMOTE*, Qualifier *SCRIPT*, Qualifier *DIRECT*)

Setup a GENIE session for instrument control operations. This command is passed the name of the instrument that you would like to control and does the necessary initialisations for talking to it. You only need to execute this command once per GENIE session and, on the instrument itself, it will have been done for you already in the instrument GENIEINIT. This command also calls the **GETBLOCKS** (p. 18) command to initialise **CSET** (p. 34) and **CSHOW** (p. 35).

Parameters:

← ***INSTRUMENT*** Instrument name to connect to and control

← ***MODE*** (not used at present)

← ***NEWDAE*** Talk to a new style Muon DAE (default: old sytle neutron DAE)

← ***DIRECT*** (not used at present)

← ***REMOTE*** (not used at present)

← ***SCRIPT*** (not used at present)

See also:

GETBLOCKS (p. 18) **CSET** (p. 34) **CSHOW** (p. 35) **Example**

```
ISC_SETUP/NEWDAE "NDXSURF"
```

7.10 Data Acquisition Control.

These commands control data acquisition.

Functions

- **ABORT** (Qualifier QUIET)
- **ABORT_POSTCMD** (Qualifier QUIET)
- **ABORT_PRECMD** (Qualifier QUIET)
- **AUTOUPDATE** (Real DUMP_INTERVAL)
- **BEAMLINE_PAR** (String PAR, Workspace EXTRAPARAMS, Qualifier SET, Qualifier GET, Qualifier REMOVE, Qualifier LIST)
- **BEGIN** (Integer PERIOD, String MEASUREMENT_ID, String MEASUREMENT_TYPE, String MEASUREMENT_SUBID, String SAMPLE_ID, Qualifier WAIT, Qualifier QUIET, Qualifier PAUSED)
- **BEGIN_POSTCMD** (Integer RUN_NUMBER, Qualifier QUIET)
- **BEGIN_PRECMD** (Qualifier QUIET)
- **CHANGE** (String TITLE, String USER, Integer PERIOD, Integer RBNO, Real AOI, Real PHI, Integer NPERIODS, Real THICKNESS, String SAMPLE_NAME)
- **CHANGE_FINISH** ()
- **CHANGE_MONITOR** (Integer SPEC, Real TLOW, Real THIGH)
- **CHANGE_PERIODS** (Integer NSOFT, Integer SEQUENCES, Integer OUTPUT_DELAY, String FILE, Integer FRAMES, Integer OUTPUT, String LABEL, Qualifier HARD_INT, Qualifier HARD_EXT, Qualifier SOFT, Qualifier PFILE, Qualifier PERIOD1, Qualifier PERIOD2, Qualifier PERIOD3, Qualifier PERIOD4, Qualifier PERIOD5, Qualifier PERIOD6, Qualifier PERIOD7, Qualifier PERIOD8, Qualifier ALLPERIODS, Qualifier DAQ, Qualifier DWELL, Qualifier UNUSED)
- **CHANGE_START** ()
- **CHANGE_SYNC** (Qualifier ISIS, Qualifier INTERNAL, Qualifier SMP)
- **CHANGE_TABLES** (String WIRING, String DETECTOR, String SPECTRA)
- **CHANGE_TCB** (Real TLOW, Real THIGH, Real TSTEP, String FILE, Qualifier LOG, Qualifier TCBFILE, Qualifier RANGE1, Qualifier RANGE2, Qualifier RANGE3, Qualifier RANGE4, Qualifier RANGE5, Qualifier REGIME1, Qualifier REGIME2)
- **CHANGE_VARS** (Real AOI, Real PHI, Integer NPERIODS)
- **CHANGE_VETOS** (Qualifier CLEARALL, Qualifier SMP, Qualifier NOSMP, Qualifier TS2, Qualifier NOTS2, Qualifier HZ50, Qualifier NOHZ50, Qualifier EXT0, Qualifier NOEXT0, Qualifier EXT1, Qualifier NOEXT1, Qualifier EXT2, Qualifier NOEXT2, Qualifier EXT3, Qualifier NOEXT3)
- **DAE_WRITE** (Integer CARD_ID, Integer CARD_ADDRESS, Integer VALUE, Qualifier W16)

- **END** (Qualifier IMMEDIATE, Qualifier QUIET)
- **END_POSTCMD** (Integer RUN_NUMBER, Qualifier IMMEDIATE, Qualifier QUIET)
- **END_PRECMD** (Qualifier IMMEDIATE, Qualifier QUIET)
- **GETFRAMES** (String HOST, Qualifier ALLPERIODS)
- **GETMEVENTS** (String HOST)
- **GETPERIOD** (String HOST)
- **GETRUNNUMBER** ()
- **GETRUNSTATE** (Qualifier VI)
- **GETTOTALCOUNTS** (String HOST)
- **GETUAMPS** (String HOST, Qualifier TS1MIDNIGHT, Qualifier TS2MIDNIGHT, Qualifier ALLPERIODS)
- **LOADSCRIPT** (String SCRIPT, String DIR)
- **PAUSE** (Qualifier IMMEDIATE, Qualifier QUIET)
- **PAUSE_POSTCMD** (Qualifier QUIET)
- **PAUSE_PRECMD** (Qualifier QUIET)
- **RECOVER** ()
- **RESUME** (Qualifier QUIET)
- **RESUME_POSTCMD** (Qualifier QUIET)
- **RESUME_PRECMD** (Qualifier QUIET)
- **SAMPLE_PAR** (String PAR, Workspace EXTRAPARAMS, Qualifier SET, Qualifier GET, Qualifier REMOVE, Qualifier LIST)
- **SET_FERMI_VETO** (Real DELAY, Real WIDTH, Qualifier ON, Qualifier OFF)
- **SETSCRIPTDIR** (String DIRNAME)
- **SETSCRIPTNAME** (String NAME)
- **SNAPSHOT_CRPT** (String FILE, Qualifier NOUPDATE, Qualifier NOPAUSE)
- **STORE** ()
- **SUM_ALL_DAE_MEMORY** ()
- **SUM_ALL_SPECTRA** ()
- **UPDATE** ()
- **UPDATESTORE** (Qualifier NOPAUSE)
- **USER_DETAILS** (String PAR, Integer RBNO, String NAME, String INSTITUTE, String ROLE, Qualifier SET, Qualifier GET, Qualifier REMOVE)
- **WAITFOR** (String BLOCK, Generic VALUE, Generic RANGE, Generic LOWLIMIT, Generic HIGHLIMIT, Generic EXTRAPARAMS, Qualifier ALL, Qualifier ANY)
- **WAITFORDASHRUNSTATE** (String STATE, Integer MAXWAIT, Qualifier FALSE, Qualifier TRUE)

7.10.1 Detailed Description

These commands control data acquisition. They are used to start and stop collection as well as generate the final raw data file.

7.10.2 Function Documentation

7.10.2.1 ABORT (Qualifier *QUIET*)

Abort a run. The instrument is returned to the SETUP state, but data is not lost until the next **BEGIN** (p. 22). If you typed **ABORT** (p. 20) in error, use the **RECOVER** (p. 30) command to return the instrument to a RUNNING state before typing **END** (p. 27)

7.10.2.2 ABORT_POSTCMD (Qualifier *QUIET*)

Called by **ABORT** (p. 20) immedately after data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.3 ABORT_PRECMD (Qualifier *QUIET*)

Called by **ABORT** (p. 20) immedately before data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.4 AUTOUPDATE (Real *DUMP_INTERVAL*)

Start an autoupdate loop. This command is usually ran in a second genie process and causes that process to wait until the specified number of uAmp.hour has been reached at which point an **UPDATESTORE** (p. 32) is executed.

Parameters:

← ***DUMP_INTERVAL*** How man uAmp.hour to regularly dump

7.10.2.5 BEAMLINE_PAR (String *PAR*, Workspace *EXTRAPARAMS*, Qualifier *SET*, Qualifier *GET*, Qualifier *REMOVE*, Qualifier *LIST*)

Set a value on trhe Beamline Parameters VI.

Parameters:

← ***PAR*** Name of parameter to set or get

← ***SET*** set/write the value

← ***GET*** get/read the value

← ***REMOVE*** remove the value

← ***LIST*** list all values on the VI

See also:

SAMPLE_PAR (p. 31), **USER_DETAILS** (p. 33)

Returns:

values

7.10.2.6 BEGIN (Integer *PERIOD*, String *MEASUREMENT_ID*, String *MEASUREMENT_TYPE*, String *MEASUREMENT_SUBID*, String *SAMPLE_ID*, Qualifier *WAIT*, Qualifier *QUIET*, Qualifier *PAUSED*)

Start A RUN. The acquisition memory is cleared and data collection is started. Before counting begins the procedure **BEGIN_PRECMD** (p. 22) is called to allow per instrument configuration.

Parameters:

← ***PERIOD*** (optional) start the run in a period number other than 1

← ***WAIT*** (optional) Wait 120 seconds, abort and then begin again

← ***QUIET*** (optional) print less output to screen

← ***MEASUREMENT_ID*** (optional) measurement_id to associate with run - see **MEASUREMENT** (p. **??**)

← ***MEASUREMENT_SUBID*** (optional) measurement_subid to associate with run - see **MEASUREMENT** (p. **??**)

← ***MEASUREMENT_TYPE*** (optional) measurement type to associate with run - see **MEASUREMENT** (p. **??**)

← ***SAMPLE_ID*** (optional) sample identifier to associate with run - see **MEASUREMENT** (p. **??**)

Returns:

run number of started run

Precondition:

The instrument is in a SETUP state

See also:

MEASUREMENT (p. **??**)

7.10.2.7 BEGIN_POSTCMD (Integer *RUN_NUMBER*, Qualifier *QUIET*)

Called by **BEGIN** (p. 22) immedately after data collection is actually started. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.8 BEGIN_PRECMD (Qualifier *QUIET*)

Called by **BEGIN** (p. 22) immedately before data collection is actually started. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.9 CHANGE (String *TITLE*, String *USER*, Integer *PERIOD*, Integer *RBNO*, Real *AOI*, Real *PHI*, Integer *NPERIODS*, Real *THICKNESS*, String *SAMPLE_NAME*)

Change values on the instrument.

Parameters:

← ***TITLE*** Run title
← ***SAMPLE_NAME*** Sample name
← ***USER*** User name
← ***PERIOD*** curent data acquisition period
← ***RBNO*** RB (experiment) number
← ***AOI*** Angle of incidence (reflectomters)
← ***PHI*** Sample angle PHI
← ***NPERIODS*** Number of periods

Warning:

NPERIODS is only available in SETUP and *PERIOD* in a non-running state.

7.10.2.10 CHANGE_FINISH ()

End a block change operation started by **CHANGE_START** (p. 24). Block change operations must be enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.11 CHANGE_MONITOR (Integer *SPEC*, Real *TLOW*, Real *THIGH*)

Change monitor as part of a block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands.

Parameters:

← ***SPEC***
← ***TLOW***
← ***THIGH***

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.12 CHANGE_PERIODS (Integer *NSOFT*, Integer *SEQUENCES*, Integer *OUTPUT_DELAY*, String *FILE*, Integer *FRAMES*, Integer *OUTPUT*, String *LABEL*, Qualifier *HARD_INT*, Qualifier *HARD_EXT*, Qualifier *SOFT*, Qualifier *PFILE*, Qualifier *PERIOD1*, Qualifier *PERIOD2*, Qualifier *PERIOD3*, Qualifier *PERIOD4*, Qualifier *PERIOD5*, Qualifier *PERIOD6*, Qualifier *PERIOD7*, Qualifier *PERIOD8*, Qualifier *ALLPERIODS*, Qualifier *DAQ*, Qualifier *DWELL*, Qualifier *UNUSED*)

Change period setup as part of a scripted block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands

Parameters:

← ***HARD_INT*** Hardware periods where a period change is initiated by the DAE after a specified number of frames

← ***HARD_EXT*** Hardware periods where period changes are initialted by an external source

← ***SOFT*** Software periods where period is changed by user via a **CHANGE** (p. 22) PERIOD= command

← ***PERIOD1*** Apply parameters such as *DAQ*, *FRAMES* etc. to first hardware period

← ***ALLPERIODS*** Apply parameters such as *DAQ*, *FRAMES* etc. to all periods

← ***NSOFT*** set the number of software periods (*SOFT* only)

← ***SEQUENCES*** set number of hardware period sequences to do (*HARD_INT* only)

← ***OUTPUT_DELAY*** define the position within a frame (ms) at which a *HARD_INT* period change takes place

← ***DAQ*** The period selected by *PERIOD1* or *ALLPERIODS* will be a Data Acquisitoon period

← ***DWELL*** The period selected by *PERIOD1* or *ALLPERIODS* will be a Dwell (non Data Acquisition) period

← ***UNUSED*** The period selected by *PERIOD1* or *ALLPERIODS* will not be used

← ***FRAMES*** Specify the number of good frames to count in *PERIOD1* or *ALLPERIODS* (*HARD_INT* only)

← ***OUTPUT*** Specify the output signal for *PERIOD1* or *ALLPERIODS* (*HARD_INT* and *HARD_EXT* only)

← ***LABEL*** Specify the display label for *PERIOD1* or *ALLPERIODS* (*HARD_INT* and *HARD_EXT* only)

← ***PFILE*** Use file specified by *FILE* to read period information from (*HARD_INT* and *HARD_EXT* only). Deafult: use *PERIOD1* etc. specified values. a periods file just contains line of the form "DAQ 100 1 per1" for "DAQ period, 100 frames, output 1, label per1"

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.13 CHANGE_START ()

start a scripted block change operation. A block change operation is completed when **CHANGE_FINISH** (p. 23) is called. Between these two calls a sequence of other commands such as **CHANGE_TABLES** (p. 25) can be called.

Precondition:

The instrument is in a SETUP state

7.10.2.14 CHANGE_SYNC (Qualifier *ISIS*, Qualifier *INTERNAL*, Qualifier *SMP*)

Change DAE timing (sync) source as part of a block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands.

Parameters:

← ***INTERNAL*** use internal (test clock) frame sync

← ***ISIS*** Use ISIS as the timing source

← ***SMP*** Use SMP (the chopper)

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.15 CHANGE_TABLES (String *WIRING*, String *DETECTOR*, String *SPECTRA*)

Change tables as part of a block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands.

Parameters:

← ***WIRING*** New wiring table

← ***DETECTOR*** New detector table

← ***SPECTRA*** New spectra table

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.16 CHANGE_TCB (Real *TLOW*, Real *THIGH*, Real *TSTEP*, String *FILE*, Qualifier *LOG*, Qualifier *TCBFILE*, Qualifier *RANGE1*, Qualifier *RANGE2*, Qualifier *RANGE3*, Qualifier *RANGE4*, Qualifier *RANGE5*, Qualifier *REGIME1*, Qualifier *REGIME2*)

Change time channels as part of a scripted block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands

Parameters:

← ***TLOW***

← ***THIGH***

← ***TSTEP***

← ***FILE*** File to load TCB values from if TCBFILE qualifier used (default: c:/labview modules/-dae/tcb.dat)

← ***LOG*** Use logarithmic rather than linear binning for this range

← ***TCBFILE*** Load values from data in argument

- FILE

← ***RANGE1*** The values given correspond to the first time range

← ***RANGE2*** The values given correspond to the second time range

← ***RANGE3***

← ***RANGE4***

← ***RANGE5***

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.17 CHANGE_VARS (Real *AOI*, Real *PHI*, Integer *NPERIODS*)

Change other variables as part of a block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands.

Parameters:

← ***AOI*** Angle of incidence (reflectomters)

← ***PHI*** Sample angle PHI

← ***NPERIODS*** Number of periods

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.18 CHANGE_VETOS (Qualifier *CLEARALL*, Qualifier *SMP*, Qualifier *NOSMP*, Qualifier *TS2*, Qualifier *NOTS2*, Qualifier *HZ50*, Qualifier *NOHZ50*, Qualifier *EXT0*, Qualifier *NOEXT0*, Qualifier *EXT1*, Qualifier *NOEXT1*, Qualifier *EXT2*, Qualifier *NOEXT2*, Qualifier *EXT3*, Qualifier *NOEXT3*)

Change vetos as part of a block change operation. Block change operations are enclosed by **CHANGE_START** (p. 24) and **CHANGE_FINISH** (p. 23) commands. The default behaviour is to leave the veto setting unchanged if it is not specified, so you need to explicitly use e.g. /NOSMP to always disable the SMP veto

Parameters:

← ***SMP*** enable SMP veto

← ***NOSMP*** disable SMP veto

← ***TS2*** enable TS2 pulse veto

← ***NOTS2*** disable TS2 pulse veto

← ***HZ50*** enable ISIS not at 50 Hz veto

← ***NOHZ50*** disable ISIS not at 50 Hz veto

← ***SMP*** enable SMP veto

← ***NOSMP*** disable SMP veto

← ***EXT0*** enable external veto 0

← ***NOEXT0*** disable external veto 0

← ***EXT1*** enable external veto 1

← ***NOEXT1*** disable external veto 1

← ***EXT2*** enable external veto 2

← ***NOEXT2*** disable external veto 2

← ***EXT3*** enable external veto 3

← ***NOEXT3*** disable external veto 3

← ***CLEARALL*** clear all vetos first (can be used in combination with others)

Precondition:

A previous **CHANGE_START** (p. 24) command been issued

7.10.2.19 DAE_WRITE (Integer *CARD_ID*, Integer *CARD_ADDRESS*, Integer *VALUE*, Qualifier *W16*)

write a value to a location on a DAE acquisition card e.g. reset detector card period size - write 0 to address 8 and 12

Parameters:

← ***W16*** Write a 16bit value (default: 32bit)

← ***CARD_ID*** ID (crate) of acquisition card

← ***CARD_ADDRESS*** Address of location on card to reference

← ***VALUE*** Value to write

7.10.2.20 END (Qualifier *IMMEDIATE*, Qualifier *QUIET*)

End a run. Data collection is stopped, a raw data file is created and the instrument returns to the SETUP state. Before counting finishes the procedure **END_PRECMD** (p. 27) is called to allow per instrument configuration. All values specified in a **MEASUREMENT** (p. **??**) are cleared on an END

Returns:

run number of ended run

See also:

MEASUREMENT (p. **??**), **BEGIN** (p. 22)

7.10.2.21 END_POSTCMD (Integer *RUN_NUMBER*, Qualifier *IMMEDIATE*, Qualifier *QUIET*)

Called by **END** (p. 27) immedately after data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.22 END_PRECMD (Qualifier *IMMEDIATE*, Qualifier *QUIET*)

Called by **END** (p. 27) immedately before data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.23 GETFRAMES (String *HOST*, Qualifier *ALLPERIODS*)

return current good frames.

Returns:

FRAMES

Parameters:

← ***ALLPERIODS*** Return a Workspace with two arrays *raw* and *processed* containing the per-period values

7.10.2.24 GETMEVENTS (String *HOST*)

return total counts for all detectors connected.

Parameters:

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

MEVENTS Number of events in millions as a float

7.10.2.25 GETPERIOD (String *HOST*)

return current period number.

Returns:

PERIOD number

7.10.2.26 GETRUNNUMBER ()

Returns what current run number of the instrument as an Integer value.

Returns:

Instrument run number as Integer

7.10.2.27 GETRUNSTATE (Qualifier *VI*)

Returns what status of the instrument as an Integer value. You can use RUN_STATUS(code) to get a readable String run state.

See also:

RUN_STATUS

Returns:

Instrument status as Integer

Return values:

SETUP

RUNNING

PAUSED

WAITING

Parameters:

← ***VI*** (optional) read run state from the DAE VI and not the control program. You would only want to do this if you were about to interact with the VI itself and needed to make sure it was in the correct state.

7.10.2.28 GETTOTALCOUNTS (String *HOST*)

return total counts for all detectors connected.

Parameters:

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

COUNTS Total number of counts

7.10.2.29 GETUAMPS (String *HOST*, Qualifier *TS1MIDNIGHT*, Qualifier *TS2MIDNIGHT*, Qualifier *ALLPERIODS*)

return good Microamp hours.

Parameters:

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

← ***TS1MIDNIGHT*** Return total TS1 uamps to target since midnight

← ***TS2MIDNIGHT*** Return total TS2 uamps to target since midnight

← ***ALLPERIODS*** Return a Workspace with two arrays *raw* and *processed* containing the per-period values

Returns:

UAMPS

7.10.2.30 LOADSCRIPT (String *SCRIPT*, String *DIR*)

Load a script file into memory and name it. This loads an old style script file, which has a .isc extension and is like a GENIE PROCEDURE except that is is missing the PROCEDURE and ENDPROCEDURE commands. The file is loaded and a command is defined in GENIE which is the same as the filename; this name can then be used to execute the script. If the file does not exist, it is searched for in the directory specified by the **SETSCRIPTDIR** (p. 31) command.

Parameters:

← ***SCRIPT*** name of script to run

← ***DIR*** (not used at present)

7.10.2.31 PAUSE (Qualifier *IMMEDIATE*, Qualifier *QUIET*)

PAUSE a run. Data collection is suspended until a subsequent **RESUME** (p. 20).

7.10.2.32 PAUSE_POSTCMD (Qualifier *QUIET*)

Called by **PAUSE** (p. 30) immedately after data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.33 PAUSE_PRECMD (Qualifier *QUIET*)

Called by **PAUSE** (p. 30) immedately before data collection is actually stopped. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.34 RECOVER ()

Recover from an unintentional **ABORT** (p. 20) command. This must be typed before the next **BEGIN** (p. 22).

7.10.2.35 RESUME_POSTCMD (Qualifier *QUIET*)

Called by **RESUME** (p. 20) immedately after data collection is actually started. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.36 RESUME_PRECMD (Qualifier *QUIET*)

Called by **RESUME** (p. 20) immedately before data collection is actually started. The default implementation does nothing and can be changed on an instrument by instrument basis.

7.10.2.37 SAMPLE_PAR (String *PAR*, Workspace *EXTRAPARAMS*, Qualifier *SET*, Qualifier *GET*, Qualifier *REMOVE*, Qualifier *LIST*)

Read or write a value on the Sample Parameters VI.

Parameters:

← ***PAR*** Name of parameter to set or get

← ***SET*** set/write the value

← ***GET*** get/read the value

← ***REMOVE*** remove the value

← ***LIST*** list all values on the VI

See also:

BEAMLINE_PAR (p. 21), **USER_DETAILS** (p. 33)

Returns:

values

7.10.2.38 SET_FERMI_VETO (Real *DELAY*, Real *WIDTH*, Qualifier *ON*, Qualifier *OFF*)

set fermi chopper veto parameters. This is only valid on instruments with such a device

Parameters:

← ***DELAY*** Delay in microseconds

← ***WIDTH*** Width in microseconds

← ***ON*** Enable veto

← ***OFF*** Disable Veto

7.10.2.39 SETSCRIPTDIR (String *DIRNAME*)

Define a default search directory for **LOADSCRIPT** (p. 30).

Parameters:

← ***DIRNAME*** Directory to search

7.10.2.40 SETSCRIPTNAME (String *NAME*)

Display the name of a script on the dashboard.

Parameters:

← ***NAME*** Name of script to display on dashboard

7.10.2.41 SNAPSHOT_CRPT (String *FILE*, Qualifier *NOUPDATE*, Qualifier *NOPAUSE*)

Update CRPT and save to a named file. Unless *NOUPDATE* is specified data is first loaded from the electronics into the computer memory as if an **UPDATE** (p. 32) had been issued. The data and other parameters are then written to disk.

See also:

UPDATESTORE (p. 32)

7.10.2.42 STORE ()

Store the CRPT. Data loaded into memory by a previous **UPDATE** (p. 32) command is now written to disk.

See also:

UPDATESTORE (p. 32)

7.10.2.43 UPDATE ()

Update CRPT. Data is loaded from the electronics into the computer memory, but is not written to disk until a **STORE** (p. 32) command is executed.

Warning:

Data is still collected during the update - if you want a true consistent snapshot you should first execute the command between a **PAUSE** (p. 30) and **RESUME** (p. 20)

See also:

UPDATESTORE (p. 32)

7.10.2.44 UPDATESTORE (Qualifier *NOPAUSE*)

Perform an **UPDATE** (p. 32) and **STORE** (p. 32) in a combined operation. This is slightly more efficeint than doing the two separately.

Parameters:

← ***NOPAUSE*** Do not pause collection while doing update and store

7.10.2.45 USER_DETAILS (String *PAR*, Integer *RBNO*, String *NAME*, String *INSTITUTE*, String *ROLE*, Qualifier *SET*, Qualifier *GET*, Qualifier *REMOVE*)

Set a value on the User Details VI.

Parameters:

← ***PAR*** Name of parameter to set or get
← ***RBNO*** set the RB number
← ***NAME*** set the user name
← ***INSTITUTE*** set the user institute
← ***ROLE*** set the user role (PI or Contact)
← ***PAR*** Name of parameter to set or get
← ***SET*** set/write the value
← ***GET*** get/read the value
← ***REMOVE*** remove the value

Returns:

values

See also:

BEAMLINE_PAR (p. 21), **SAMPLE_PAR** (p. 31)

7.10.2.46 WAITFOR (String *BLOCK*, Generic *VALUE*, Generic *RANGE*, Generic *LOWLIMIT*, Generic *HIGHLIMIT*, Generic *EXTRAPARAMS*, Qualifier *ALL*, Qualifier *ANY*)

Wait for a specified condition before proceeding further. The command allows script execution to be suspended until e.g. the specified number of uAmp.hour has been reached or some other condition. Multiple conditions can be specified on the command and then /ALL or /ANY will indicate if one of all of these needs to be met.

Parameters:

← ***ALL*** Wait for all conditions
← ***ANY*** Wait for any condition
← ***RANGE*** An Interval structure indicating the range of values to wait for
← ***LOWLIMIT*** Low limit of range
← ***HIGHLIMIT*** High limit of range
← ***EXTRAPARAMS*** NAME=VALUE list of blocks to wait for. NAME can be any current SECI block or the following pre-defined named
- SECONDS real time
- FRAMES good frames
- UAMPS microamps
- TIME a Interval containing hours:minutes

7.10.2.47 WAITFORDASHRUNSTATE (String *STATE*, Integer *MAXWAIT*, Qualifier *FALSE*, Qualifier *TRUE*)

Wait for a particular instrument run state.

Parameters:

← ***STATE*** Run state to wait for

← ***TRUE*** wait for the specified run state to be entered

← ***FALSE*** wait for any state other than the specified one

See also:

GETRUNSTATE (p. 28)

7.11 High level instrument control.

These commands sample environment equipment in a high level, general way and allow setting of parameters via configured SECI blocks.

Functions

- **CSET** (String BLOCK, Generic VALUE, Generic RANGE, Generic LOWLIMIT, Generic HIGHLIMIT, Real SPREAD, Real RETRY, Integer NRETRY, Generic EXTRAPARAMS, Qualifier WAIT, Qualifier CONTROL, Qualifier NOCONTROL)
- **CSHOW** (String BLOCK, Qualifier ALL)
- **GETSE** (String BLOCK, Integer RUN_NUMBER, String FILE)
- **SEPLOT** (String BLOCK, Integer RUN_NUMBER, String FILE, Qualifier PLOTAV)

7.11.1 Detailed Description

These commands sample environment equipment in a high level, general way and allow setting of parameters via configured SECI blocks. **CSET** (p. 34) and **CSHOW** (p. 35) are good examples of such commands. To access values on LabVIEW VIs directly you need to use the **Low level LabVIEW commands** (p. 36) instead

7.11.2 Function Documentation

7.11.2.1 CSET (String *BLOCK*, Generic *VALUE*, Generic *RANGE*, Generic *LOWLIMIT*, Generic *HIGHLIMIT*, Real *SPREAD*, Real *RETRY*, Integer *NRETRY*, Generic *EXTRAPARAMS*, Qualifier *WAIT*, Qualifier *CONTROL*, Qualifier *NOCONTROL*)

Set a sample environment variable value. The command is passed the name of a SECI defined sample environment parameter and its desired value in a NAME=VALUE style. It is possible to set more than one value at a time, and all the values will be set before any buttons are pressed. The order of setting will not be the same as specified on the command line though.

Parameters:

← ***RANGE*** Range of values allowed for WAIT or CONTROL specified as LOWER:UPPER

← ***LOWLIMIT***

← ***HIGHLIMIT***

← ***RETRY*** Used with /WAIT to specify a retry interval (seconds) for resending setpoint

← ***NRETRY*** Used with *RETRY* to specify a maximum numnber of attemps

← ***SPREAD*** Fractional spread of values allowed for WAIT or CONTROL - 0.1 means 10%

← ***EXTRAPARAMS*** Additional NAME=VALUE arguments

← ***WAIT***

← ***CONTROL*** Enable run control on this block

← ***NOCONTROL*** Disable run control on this block

See also:

CSHOW (p. 35) **GETBLOCKS** (p. 18)

Example

```
CSET TEMP1=10.0 SPREAD=0.1
```

7.11.2.2 CSHOW (String *BLOCK*, Qualifier *ALL*)

show a sample environemtn variable value

Parameters:

← ***BLOCK*** Name of SECI defined block to read

← ***ALL*** List all blocks and their values to the screen

Returns:

Generic value of block; undefined if /ALL is specified

7.11.2.3 GETSE (String *BLOCK*, Integer *RUN_NUMBER*, String *FILE*)

Read a set of sample environment values from a log file into a Workspace. The values can then be plotted using the **DISPLAY** (p. 16) and **PLOT** (p. 17) commands. To read and plot in one command use **SEPLOT** (p. 36).

Either a full path to the filename can be given or both the name of the block and the run_number. In the latter case the filename is constructed from the current instrument, disk and directory settings.

Parameters:

← ***BLOCK*** Name of the block to read

← ***RUN_NUMBER*** Run number to read block from

← ***FILE*** Full name of a log file to read

Returns:

Workspace with data values

See also:

SEPLOT (p. 36)

7.11.2.4 SEPLOT (String *BLOCK*, Integer *RUN_NUMBER*, String *FILE*, Qualifier *PLOTAV*)

Plot a set of sample environment values from a log file. Either a full path to the filename can be given or both the name of the block and the run_number. In the latter case the filename is constructed from the current instrument, disk and directory settings. The comamnd uses **GETSE** (p. 35) to read the data and **DISPLAY** (p. 16) to plot it

See also:

GETSE (p. 35) **DISPLAY** (p. 16)

Parameters:

← ***BLOCK*** Name of the block to read

← ***RUN_NUMBER*** Run number to read block from

← ***FILE*** Full name of a log file to read

← ***PLOTAV*** Also plot a line showing the average Y value on the graph

7.12 Low Level LabVIEW Control.

These commands act directly on labVIEW VIs and do not use the blocks configured in SECI and displayed on the dashboard, unlike the **high level commands** (p. 34).

Functions

- **CALLLABVIEW** (String VI, Stringarray NAMES, Array VALUES, String HOST, Qualifier NOREF, Qualifier REENTRANT)
- **CALLVIMETHOD** (String VI, String METHOD, Generic P1, Generic P2, Generic P3, Generic P4, String HOST)
- **CLOSEVIFRONTPANEL** (String VI, String HOST)
- **GETLABVIEWVAR** (String VI, String VARNAME, String HOST)
- **HIDEVI** (String VI, String HOST)
- **PUSHLABVIEWBUTTON** (String VI, String BUTTONNAME, String HOST, Qualifier WAIT)
- **SETLABVIEWVAR** (String VI, String VARNAME, Generic VALUE, String HOST, Qualifier NOSIGNAL)
- **SETLABVIEWVARSIGNAL** (String VI, String VARNAME, Generic VALUE, String HOST, Qualifier NOREF)
- **STARTVI** (String VI, String HOST, Qualifier WAIT)
- **STOPVI** (String VI, String HOST)
- **UNHIDEVI** (String VI, String HOST)
- **WAITFORBOOLEAN** (String VI, String BOOL, String HOST, Qualifier FALSE, Qualifier TRUE)
- **WAITFORBUTTON** (String VI, String BUTTON, String HOST)

7.12.1 Detailed Description

These commands act directly on labVIEW VIs and do not use the blocks configured in SECI and displayed on the dashboard, unlike the **high level commands** (p. 34). They commands are usually used to produce custom commands where more complexity than is allowed by e.g. **CSET** (p. 34) is needed.

7.12.2 Function Documentation

7.12.2.1 CALLLABVIEW (String *VI*, Stringarray *NAMES*, Array *VALUES*, String *HOST*, Qualifier *NOREF*, Qualifier *REENTRANT*)

Call a LabVIEW VI as a Sub VI. This command is passed the name of the LabVIEW VI to call and a set of (name, value) parameters. It will call the VI and not return until processing is complete.

If the vi supports reentran execution, this is enabled

Parameters:

← ***VI*** Name of VI to call

← ***NAMES*** Labels of Variables on VI front panel

← ***VALUES*** Values of Variables on VI front panel, usually created with **BUILD_ARRAY** (p. 4)

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

← ***NOREF*** (optional) Do not pass variables to labview by reference

← ***REENTRANT*** (optional) Always open Vi for re-rentrant execution (default is to check if this is possible)

Returns:

VI call result

Warning:

The VI should probably be enabled for re-rentrant operations

See also:

SETLABVIEWVAR (p. 39) **BUILD_ARRAY** (p. 4)

7.12.2.2 CALLVIMETHOD (String *VI*, String *METHOD*, Generic *P1*, Generic *P2*, Generic *P3*, Generic *P4*, String *HOST*)

Call a method on a LabVIEW VI. This command is passed the name of the LabVIEW VI and the method to call

Parameters:

← ***VI*** Name of VI to read

← ***METHOD*** method to call

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.3 CLOSEVIFRONTPANEL (String *VI*, String *HOST*)

Close a LabVIEW VI front panel. This command is passed the name of the LabVIEW VI whose front panel you wish to close

Parameters:

← ***VI*** Name of VI to read
← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.4 GETLABVIEWVAR (String *VI*, String *VARNAME*, String *HOST*)

Read a variable from a LabVIEW VI. This command is passed the name of the LabVIEW VI and the label for the variable on the front panel.It will return the value of that variable.

Parameters:

← ***VI*** Name of VI to read
← ***VARNAME*** Label of Variable on VI front panel
← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

See also:

SETLABVIEWVAR (p. 39) **Example**

```
VAL=GetLabVIEWVar("Eurothem.vi", "Some Control label")
```

7.12.2.5 HIDEVI (String *VI*, String *HOST*)

Hide a LabVIEW VI front panel. This command is passed the name of the LabVIEW VI whose front panel you wish to close

Parameters:

← ***VI*** Name of VI to read
← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.6 PUSHLABVIEWBUTTON (String *VI*, String *BUTTONNAME*, String *HOST*, Qualifier *WAIT*)

Push a button on a LabVIEW VI. This command is passed the name of the LabVIEW VI and the label for the button. It will press the button and (optionally) wait for the button to pop back up again.

Parameters:

← ***VI*** name of LabVIEW VI

← ***BUTTONNAME*** label of Button on VI

← ***HOST*** (optional) name of computer to connect to (default: That specified by **ISC_SETUP** (p. 18))

← ***WAIT*** Wait for button to pop back up before returning from procedure (default: NOWAIT)

See also:

SETLABVIEWVAR() (p. 39) **Example**

```
PushLabVIEWButton "Eurothem.vi" "Read Value"
```

7.12.2.7 SETLABVIEWVAR (String *VI*, String *VARNAME*, Generic *VALUE*, String *HOST*, Qualifier *NOSIGNAL*)

Set a variable on a LabVIEW VI front panel. This command is passed the name of the LabVIEW VI and the label for the variable on the front panel. It will set the variable to the supplied value.

Parameters:

← ***VI*** Name of VI to read

← ***VARNAME*** Label of Variable on VI front panel

← ***VALUE*** Value to set

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

← ***NOSIGNAL*** Do not use **SETLABVIEWVARSIGNAL** (p. 39) for set even when $ISC_-LABVIEWSIGNAL is defined

See also:

GETLABVIEWVAR() (p. 38) **Example**

```
SetLabVIEWVar "Eurother.vi" "Temperature" 55.0
```

7.12.2.8 SETLABVIEWVARSIGNAL (String *VI*, String *VARNAME*, Generic *VALUE*, String *HOST*, Qualifier *NOREF*)

Set a variable on a LabVIEW VI front panel and trigger an event. This command is passed the name of the LabVIEW VI and the label for the variable on the front panel. It will set the variable to the supplied value. The call uses an intermediate VI to implement the "set value with signalling" mechanism and so trigger a LabVIEW event.

Parameters:

← ***VI*** Name of VI to read

← ***VARNAME*** Label of Variable on VI front panel

← ***VALUE*** Value to set

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

See also:

SETLABVIEWVAR (p. 39)

7.12.2.9 STARTVI (String *VI*, String *HOST*, Qualifier *WAIT*)

Start a LabVIEW VI. This command is passed the name of the LabVIEW VI to start

Parameters:

← ***VI*** Name of VI to read

← ***WAIT*** Wait for VI to stop executing once it is started (default is not to wait)

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.10 STOPVI (String *VI*, String *HOST*)

Stop a LabVIEW VI. This command is passed the name of the LabVIEW VI to stop

Parameters:

← ***VI*** Name of VI to read

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.11 UNHIDEVI (String *VI*, String *HOST*)

Unhide a LabVIEW VI front panel. This command is passed the name of the LabVIEW VI whose front panel you wish to close

Parameters:

← ***VI*** Name of VI to read

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

Returns:

Generic variable containing value of VI control/indicator

7.12.2.12 WAITFORBOOLEAN (String *VI*, String *BOOL*, String *HOST*, Qualifier *FALSE*, Qualifier *TRUE*)

Wait for a Labview VI boolean variable to become either true or false. This command is passed the name of the LabVIEW VI to call and a set of (name, value) parameters. It will call the VI and not return until processing is complete.

Parameters:

← ***VI*** name of labview VI

← ***BOOL*** label for boolean variable

← ***TRUE*** wait for value to become TRUE

← ***FALSE*** wait for value to become FALSE (default: TRUE)

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

See also:

WAITFORBUTTON (p. 41)

7.12.2.13 WAITFORBUTTON (String *VI*, String *BUTTON*, String *HOST*)

Wait for a Labview VI button to pop back up. This command is passed the name of the LabVIEW VI to call and a set of (name, value) parameters. It will poll the VI and not return until processing is complete.

Parameters:

← ***VI*** name of labview VI

← ***BUTTON*** label for button

← ***HOST*** (optional) Remote HOST to connect to (default: as specified by **ISC_SETUP** (p. 18))

See also:

WAITFORBOOLEAN (p. 41)

8 Example Documentation

8.1 abort_handler.gcl

How to enable a Ctrl-C handler for a script using **SET_ABORT_HANDLER** (p. 9).

8.2 cphs100.gcl

A simple script used to calculate and set chopper phases.

8.3 drange_2_2.gcl

The IRIS drange command.

8.4 hardware_periods.gcl

Hardware periods example.

8.5 paths.gcl

Usage of the **SEARCHPATH** (p. 8) and **TRANSLATE_PATH** (p. 10) commands.

8.6 pg002.gcl

The IRIS PG002 command.

8.7 set_pressure.gcl

The PEARL/GEM SET_PRESSURE command.

8.8 simple.gcl

A very basic script that just uses **CSET** (p. 34) and **WAITFOR** (p. 33).

8.9 sxd_scan.gcl

A more complicated script that defines a custom PROCEDURE. This script also controls a VI using **SETLABVIEWVAR** (p. 39).

Index

ABORT
    Acquisition, 20
ABORT_POSTCMD
    Acquisition, 20
ABORT_PRECMD
    Acquisition, 20
Acquisition
    ABORT, 20
    ABORT_POSTCMD, 20
    ABORT_PRECMD, 20
    AUTOUPDATE, 20
    BEAMLINE_PAR, 20
    BEGIN, 21
    BEGIN_POSTCMD, 21
    BEGIN_PRECMD, 21
    CHANGE, 22
    CHANGE_FINISH, 22
    CHANGE_MONITOR, 22
    CHANGE_PERIODS, 23
    CHANGE_START, 23
    CHANGE_SYNC, 24
    CHANGE_TABLES, 24
    CHANGE_TCB, 24
    CHANGE_VARS, 25
    CHANGE_VETOS, 25
    DAE_WRITE, 26
    END, 26
    END_POSTCMD, 26
    END_PRECMD, 27
    GETFRAMES, 27
    GETMEVENTS, 27
    GETPERIOD, 27
    GETRUNNUMBER, 27
    GETRUNSTATE, 28
    GETTOTALCOUNTS, 28
    GETUAMPS, 28
    LOADSCRIPT, 29
    PAUSE, 29
    PAUSE_POSTCMD, 29
    PAUSE_PRECMD, 29
    RECOVER, 29
    RESUME_POSTCMD, 30
    RESUME_PRECMD, 30
    SAMPLE_PAR, 30
    SET_FERMI_VETO, 30
    SETSCRIPTDIR, 31
    SETSCRIPTNAME, 31
    SNAPSHOT_CRPT, 31
    STORE, 31
    UPDATE, 31
    UPDATESTORE, 32
    USER_DETAILS, 32
    WAITFOR, 32
    WAITFORDASHRUNSTATE, 33
ALTER
    Graphics, 15
AS_STRING
    Convertion, 14
AUTOUPDATE
    Acquisition, 20

Base Primitives, 3
BEAMLINE_PAR
    Acquisition, 20
BEGIN
    Acquisition, 21
BEGIN_POSTCMD
    Acquisition, 21
BEGIN_PRECMD
    Acquisition, 21
BUILD_ARRAY
    Primitives, 4

CALLLABVIEW
    LabVIEW, 36
CALLVIMETHOD
    LabVIEW, 37
CFN
    Output, 10
CHANGE
    Acquisition, 22
CHANGE_FINISH
    Acquisition, 22
CHANGE_MONITOR
    Acquisition, 22
CHANGE_PERIODS
    Acquisition, 23
CHANGE_START
    Acquisition, 23
CHANGE_SYNC
    Acquisition, 24
CHANGE_TABLES
    Acquisition, 24
CHANGE_TCB
    Acquisition, 24
CHANGE_VARS
    Acquisition, 25
CHANGE_VETOS
    Acquisition, 25
CLOSEVIFRONTPANEL
    LabVIEW, 37

Commands for data display., 15
Control
    CSET, 34
    CSHOW, 34
    GETSE, 35
    SEPLOT, 35
Convertion
    AS_STRING, 14
    FLOAT_AS_STRING, 14
CSET
    Control, 34
CSHOW
    Control, 34

DAE_WRITE
    Acquisition, 26
Data Acquisition Control., 18
Data type convertion., 14
DEFINED
    Primitives, 4
DIMENSIONALITY
    Primitives, 4
DIMENSIONS
    Primitives, 5
DISPLAY
    Graphics, 15

END
    Acquisition, 26
END_POSTCMD
    Acquisition, 26
END_PRECMD
    Acquisition, 27
EXISTS
    Primitives, 5

FIELDS
    Primitives, 6
FILETYPE
    Output, 10
FILL
    Primitives, 6
FLOAT_AS_STRING
    Convertion, 14

GET
    Input, 13
GETBLOCKS
    Setup, 17
GETFRAMES
    Acquisition, 27
GETLABVIEWVAR
    LabVIEW, 37
GETMEVENTS
    Acquisition, 27
GETPERIOD
    Acquisition, 27
GETRUNNUMBER
    Acquisition, 27
GETRUNSTATE
    Acquisition, 28
GETSE
    Control, 35
GETTOTALCOUNTS
    Acquisition, 28
GETUAMPS
    Acquisition, 28
Graphics
    ALTER, 15
    DISPLAY, 15
    PLOT, 16

HIDEVI
    LabVIEW, 38
High level instrument control., 33

Input
    GET, 13
Input operations, 13
Instrument Control., 17
IS_A
    Primitives, 6
ISC_SETUP
    Setup, 18

LabVIEW
    CALLLABVIEW, 36
    CALLVIMETHOD, 37
    CLOSEVIFRONTPANEL, 37
    GETLABVIEWVAR, 37
    HIDEVI, 38
    PUSHLABVIEWBUTTON, 38
    SETLABVIEWVAR, 38
    SETLABVIEWVARSIGNAL, 39
    STARTVI, 39
    STOPVI, 40
    UNHIDEVI, 40
    WAITFORBOOLEAN, 40
    WAITFORBUTTON, 41
LOAD
    Primitives, 6
LOADSCRIPT
    Acquisition, 29
Low Level LabVIEW Control., 36

Miscellaneous
    TOGGLE, 16
Miscellaneous commands, 16

NOW
Primitives, 6

Operating system interacion, 9
Output
CFN, 10
FILETYPE, 10
PRINT, 10
PRINTE, 11
PRINTEN, 11
PRINTI, 11
PRINTIN, 12
PRINTN, 12
Output operations, 9

PAUSE
Acquisition, 29
PAUSE_POSTCMD
Acquisition, 29
PAUSE_PRECMD
Acquisition, 29
PLOT
Graphics, 16
Primitives
BUILD_ARRAY, 4
DEFINED, 4
DIMENSIONALITY, 4
DIMENSIONS, 5
EXISTS, 5
FIELDS, 6
FILL, 6
IS_A, 6
LOAD, 6
NOW, 6
SAVE, 7
SEARCHPATH, 7
SET_ABORT_HANDLER, 8
SLEEP, 8
TIME_FROM_ISOTIME, 8
TRANSLATE_PATH, 9
PRINT
Output, 10
PRINTE
Output, 11
PRINTEN
Output, 11
PRINTI
Output, 11
PRINTIN
Output, 12
PRINTN
Output, 12
PUSHLABVIEWBUTTON
LabVIEW, 38

RECOVER
Acquisition, 29
RESUME_POSTCMD
Acquisition, 30
RESUME_PRECMD
Acquisition, 30

SAMPLE_PAR
Acquisition, 30
SAVE
Primitives, 7
SEARCHPATH
Primitives, 7
SEPLOT
Control, 35
SET_ABORT_HANDLER
Primitives, 8
SET_FERMI_VETO
Acquisition, 30
SETLABVIEWVAR
LabVIEW, 38
SETLABVIEWVARSIGNAL
LabVIEW, 39
SETSCRIPTDIR
Acquisition, 31
SETSCRIPTNAME
Acquisition, 31
Setup
GETBLOCKS, 17
ISC_SETUP, 18
Setup for instrument control., 17
SLEEP
Primitives, 8
SNAPSHOT_CRPT
Acquisition, 31
STARTVI
LabVIEW, 39
STOPVI
LabVIEW, 40
STORE
Acquisition, 31

TIME_FROM_ISOTIME
Primitives, 8
TOGGLE
Miscellaneous, 16
TRANSLATE_PATH
Primitives, 9

UNHIDEVI
LabVIEW, 40
UPDATE
Acquisition, 31
UPDATESTORE

Acquisition, 32
USER_DETAILS
Acquisition, 32

WAITFOR
Acquisition, 32
WAITFORBOOLEAN
LabVIEW, 40
WAITFORBUTTON
LabVIEW, 41
WAITFORDASHRUNSTATE
Acquisition, 33