NAME

EGlib – Access EuroGam data acquisition system components

SYNOPSIS

```
#include <EGlib.h>
int EGsetExperimentPath (path)
int EGcreateExperiment (name)
int EGdeleteExperiment (name)
int EGconnectExperiment (name)
int EGdisconnectExperiment ()
int EGsetApparatusPath (path)
int EGattachApparatus (name)
int EGdetachApparatus (name)
int EGauthorise (id, password)
int EGclaimApparatus (name)
int EGfreeApparatus (name)
int EGdisplayResources (stream)
int EGinquireError (buf)
int EGerrno

char *path;
char *name;
char *id;
char *password;
FILE *stream;
char *buf;
```

DESCRIPTION

EGsetExperimentPath () sets the experiment directory for locating capability file names to *path*.

EGcreateExperiment () creates in the current experiment directory an empty capability file called *name* which has no resources.

EGdeleteExperiment () deletes the capability file called *name*. from the current experiment directory.

EGconnectExperiment () resolves *name* to a capability file within the current experiment-path directory and memory-maps it, thus gaining access to the experiment's resources. At most one capability file may be mapped at a time.

EGdisconnectExperiment () unmaps the current capability file (if any) thus denying further access to the experiment's resources.

EGsetApparatusPath () sets the directory for resolving apparatus file names to *path*.

EGattachApparatus () resolves *name* to an apparatus file within the current apparatus-path directory, compiles the resource descriptions therein, and appends them to the current capability file.

EGdetachApparatus () deletes any compiled resource descriptions associated with the apparatus *name* from the current capability file, provided none of the actual resources represented by these descriptions is currently claimed.

EGauthorise () establishes the user *id* and *password* that are be used in claiming resources (such as disc spectrum servers) that are sensitive to file access rights.

EGclaimApparatus () obtains and records capabilities for all the resources represented by compiled descriptions associated with apparatus *name* in the currently mapped capability file.

EGfreeApparatus () returns the capabilities for any claimed resources represented by compiled descriptions associated with apparatus *name* in the currently mapped capability file.

EGdisplayResources () prints on *stream* a formatted dump of all compiled resource descriptions in the current capability file.

EGinquireError () returns in *buf* an explanatory message for the most recently reported error.

All **char** ∗ arguments are null-terminated character strings. There is a limit of 15 characters on all resource, apparatus, and experiment names. The calling program must supply buffers of sufficient size to hold any returned data.

Workstation software interacting with a EuroGam system sees the data acquisition hardware as a collection of named *objects* of various types. These include *register* objects, an abstraction from hardware registers, *spectrum* objects, *event-storage* objects, and others. These objects are managed by various RPC server processes called *resources*, and for purposes of access control, related resources are grouped together into *apparatuses*. To be viable, an experiment may require access to the resources of one or more apparatuses. The individual resources comprising an apparatus are described in an *apparatus file* and are allocated and deallocated together as a unit. Access to resources is controlled by by means of *capabilities* issued by the resources themselves and retained by workstations in an *capability file.* A process with RW access to a capability file can memory-map the file and thus obtain access to the resources described therein. A capability file is thus synonymous with active use of zero or more apparatuses, i.e., an *experiment*.

Each EuroGam object has a full name resembling a Unix filename of the form: /*resource*/*name* or /*role*/*name*. A *role* can be thought of as an alias for a resource name. Roles provide a way of addressing EuroGam objects by function instead of by name. EGlib supports abbreviated object name arguments in which the leading *resource* and *role* components can be omitted by means of a *path*. This is a character string that is prefixed to such arguments if they do not start with a leading '/' character. All objectnames returned by EGlib functions are in the abbreviated form.

RETURN VALUES

On success all functions return 0 except where mentioned above. On failure all functions return -1 and a specific error number can be found in **EGerrno**.

FILES

Apparatus files are in a simple property list format with each resource represented as a list of *keyword-value* pairs. The keywords recognised are **name**, **host**, **role**, **ipaddress**, **port**. Example:

```
((name vxi1) (host nnvd) (role frontend)
 (ipaddress 123456) (port 1023))
((name vxi2) (host nnve) (role frontend)
 (ipaddress 234567) (port 511))
((name vxi3) (host nnvf) (role autofill)
 (ipaddress 16909060) (port 255))
```

SEE ALSO

'Overview of EuroGam Data Acquisition System', EDOC015
'Locating EuroGam Objects', in EDOC086

SYNOPSIS

```
#include <EGlib.h>
int EGsetRegisterPath (path)
int EGreadRegister (register, value)
int EGwriteRegister (register, value)
int EGinitialiseRegister (register)
int EGreadRegisters (pattern, name, value)
int EGreadRegistersMore (name, value)
int EGwriteRegisters (pattern, value)
int EGinitialiseRegisters (pattern)
int EGinquireRegisters (pattern,name)
```

int EGinquireRegistersMore (name)
int EGcreateRegister (register, class)
int EGdestroyRegister (register)
int EGdestroyRegisters (pattern)
int EGgetObject (register, attributes)
int EGsetObject (register, attributes)
int EGgetObjects (pattern, name, attributes)
int EGgetObjectsMore (name, attributes)
int EGsetObjects (pattern, attributes)

char *path;
char *name;
char *register;
char *pattern;
char *value;
char *class;
char *attributes;
char *buf;

DESCRIPTION

EGsetRegisterPath () sets the register-path used to resolve *register* and *pattern* arguments to *path*.

EGreadRegister () returns the current value of the register with name *register*.

EGwriteRegister () sets the value of the register with name *register*.

EGinitialiseRegister () sets the register with name *register* to its default value.

EGreadRegisters returns the *name* and *value* of the first register whose name matches the pattern *pattern*. The function value is 1 if such a register exists and 0 if not.

EGreadRegistersMore returns the *name* and *value* of the next register whose name matches the *pattern* submitted in a prior call of **EGreadRegisters**. The function value is 1 if such a register exists and 0 if not.

EGwriteRegisters () sets each register whose name matches *pattern* to *value*.

EGinitialiseRegisters () sets each register whose name matches *pattern* to its default value.

EGinquireRegisters () returns the *name* of the first register whose name matches *pattern.* The function value is 1 if such a register exists and 0 if not.

EGinquireRegistersMore () returns the *name* of the next register whose name matches the *pattern* submitted in a prior call of **EGinquireRegisters()**. The function value is 1 if such a register exists and 0 if not.

EGcreateRegister () creates a register with name *register* and class *class*.

EGdestroyRegister () destroys the register with name *register*.

EGdestroyRegisters () destroys all registers whose names match *pattern*.

EGgetObject () returns in *attributes* a description of the register with name *register*.

EGsetObject () modifies the description of the register with name *register* according to *attributes*.

EGgetObjects () returns the *name* and *attributes* of the first register whose name matches *pattern*. The function value is 1 if such a register exists and 0 if not.

EGgetObjectsMore returns the *name* and *attributes* of the next register whose name matches the *pattern* submitted in a prior call of **EGgetObjects()**. The function value is 1 if such a register exists and 0 if not.

EGsetObjects () modifies the descriptions of all registers whose names match *pattern* according to *attributes*.

All **char *** arguments are null-terminated character strings. The calling program must supply buffers of sufficient size to hold the returned data.

Each EuroGam object has a full name resembling a Unix filename of the form: /*resource*/*name* or /*role*/*name*. A *role* can be thought of as an alias for a resource name. Roles provide a way of addressing EuroGam objects by function instead of by name. EGlib supports abbreviated *register* and *pattern* arguments in which the leading *resource* and *role* components can be omitted by means of a *register-path*. This is a character string that is prefixed to such arguments if they do not start with a leading '/' character. All register names returned by EGlib functions are in the abbreviated form. Register objects can also be addressed by pattern matching against their names. This allows a single function call to target multiple registers. The patterns supported are:

*	matches any sequence of zero or more characters.
?	matches any single character.
[*m-n***]**	matches any decimal digit sequence representing an integer in the range *m* to *n*.

Register values are passed as printable character strings in a register-specific format, typically decimal integer and floating point numbers, optionally followed in the case of registers representing analogue quantities, by a standard abbreviation for a physical unit of measurement. These strings are generated by **printf()** and can be decoded by **scanf()**. Example:

i=EGsetRegisterPath ("/frontend")
i=EGwriteRegisters ("GE[19-23].FastTriggerSample", "0.75uS")

Each EuroGam register server resource defines a static set of register object *classes*. EGlib provides a way of creating and deleting instances of these classes and of inquiring and modifying particular attributes of individual objects. These operations are resource and register class dependent. Consult the program documentation for particular servers for further details.

RETURN VALUES

On success all functions return 0 except where mentioned above. On failure all functions return -1 and a specific error number can be found in **EGerrno**.

SEE ALSO

'Overview of EuroGam Data Acquisition System', EDOC015
'EuroGam Frontend Register Server', EDOC024
EuroGam Register Server---program documentation

SYNOPSIS

#include <EGlib.h>
int EGsetSpectrumPath (path)
int EGcreate1dSpectrum (name, base, range, layout, type)
int EGcreate2dSpectrum (name, base1, range1, base2, range2, layout, type)
int EGdeleteSpectrum (name)
int EGread1dSpectrum (name, base, range, array, type)
int EGread2dSpectrum (name, base1, range1, base2, range2, array, type)
int EGwrite1dSpectrum (name, base, range, array, type)
int EGwrite2dSpectrum (name, base1, range1, base2, range2, array, type)
int EGreadInformation (name, number, buf)
int EGwriteInformation (name, number, buf)
int EGreadAnnotation (name, dimension, buf)
int EGwriteAnnotation (name, dimension, buf)
int EGreadCalibration (name, dimension, buf)

int EGwriteCalibration (name, dimension, buf)
int EGreadEfficiency (name, dimension, buf)
int EGwriteEfficiency (name, dimension, buf)
int EGinquireSpectrum (name, dim, bases, ranges, layouts, types)
int EGinquireAddress (name, address)
int EGinquireDirectory (name, buf)
int EGinquireDirectoryMore (buf)
int EGsetSpectrumArray (name, number, layout, type)
int EGsetDefaultArray (number)
int EGsetDefaultCompression (ranges)

char *path;
char *name;
char *buf;
int base, range, base1, range1, base2, range2, layout, type;
count *array;
int *dim, *bases, *ranges, *arrays, *layouts, *types;
int number;
int *address

DESCRIPTION

EGsetSpectrumPath () sets the path used to resolve spectrum names to *path*.

EGcreate1dSpectrum () creates a new 1-dimensional spectrum with *range* channels starting in channel *base*. Gives the spectrum default count-array attributes of *layout* and *type*.

EGcreate2dSpectrum () creates a new 2-dimensional spectrum with *range1* channels starting in channel *base1* in the first dimension and *range2* channels starting in channel *base2* in the second dimension. Gives the spectrum default count-array attributes of *layout* and *type*.

EGdeleteSpectrum () destroys spectrum *name*.

EGread1dSpectrum () reads *range* counts starting in channel *base* from the default array of the 1-dimensional spectrum *name* into memory *array* of type *type*.

EGread2dSpectrum () reads a sub-rectangle of *range1* counts in the first dimension by *range2* counts in the second dimension with origin at the channel *(base1, base2)* from the default array of the 2-dimensional spectrum into a memory *array* of type *type*.

EGwrite1dSpectrum () writes *range* counts starting in channel *base* from a memory *array* of type *type* into the default array of the 1-dimensional spectrum *name*.

EGwrite2dSpectrum () writes a sub-rectangle of *range1* counts in the first dimension by *range2* counts in the second dimension with origin at the channel *(base1,*base2) from a memory *array* of type *type* into the default array of the 2-dimensional spectrum *name*.

EGreadInformation () reads information-string *number* of spectrum *name* into the null-terminated string *buf*.

EGwriteInformation () writes the null-terminated string *buf* into information-string *number* of spectrum *name*.

EGreadAnnotation () reads the annotation-string for dimension *dimension* of spectrum *name* into the null-terminated string *buf*.

EGwriteAnnotation () writes the null-terminated string *buf* into the annotation-string for dimension *dimension* of spectrum *name*.

EGreadCalibration () reads the calibration-string for dimension *dimension* of spectrum *name* into the null-terminated string *buf*.

EGwriteCalibration () writes the null-terminated string *buf* into the calibration-string for dimension *dimension* of spectrum *name*.

EGreadEfficiency () reads the efficiency-string for dimension *dimension* of spectrum *name* into the null-terminated string *buf*.

EGwriteEfficiency () writes the null-terminated string *buf* into the efficiency-string for dimension *dimension* of spectrum *name*.

EGinquireSpectrum () for spectrum *name*: returns its dimension in *dim*; returns in *bases[i] (1<=i<=dim)* the base channel of dimension *i*; returns in *ranges[i] (1<=i<=dim)* the range of channels in dimension *i*; returns in *layouts[i] (1<=i<=2)* the layout of count-array *i*; and returns in *types[i] (1<=i<=2)* the type of count-array *i*.

EGinquireAddress () returns in *address* the system address of the base of the default count-array of spectrum *name*.

EGinquireDirectory () returns in *buf* the (directory-relative) name of the first object in directory *name*.

EGinquireDirectoryMore () returns in *buf* the (directory-relative) name of the next object in the directory specified by a prior call of **EGinquireDirectory()**.

EGsetSpectrumArray () sets the count-array *layout* and *type* for count-array *number* of spectrum *name*.

EGsetDefaultArray () determines the *number* of the count-array accessed by subsequent calls of **EGread*Spectrum()** and **EGwrite*Spectrum()**.

EGsetDefaultCompression () determines the compression-range for each dimension in subsequent calls of **EGread*Spectrum()**.

EGlib supports abbreviated spectrum *name* arguments in which the leading *resource* and *role* components can be omitted by means of a *spectrum-path*. This is a character string that is prefixed to such name arguments if they do not start with a leading '/' character. All spectrum names returned by EGlib functions are in the abbreviated form, i.e., relative to the current spectrum-path.

These functions generate RPCs on EuroGam spectrum server processes. The first component of the fully resolved spectrum name selects a resource name or role name which determines the spectrum server that is called. By convention the following role names are used:

hist selects the default EuroGam online histogrammer;

sort selects the EuroGam online event sorter;

disc selects the default disc spectrum server.

Other servers (if they exist) can be selected by using the appropriate resource names. In all cases the corresponding server resource must appear claimed in the process's currently connected capability file.

RETURN VALUES

On success all functions return 0 except where mentioned above. In the case of **EGinquireDirectory()** and **EGinquireDirectoryMore()** the function value is

0 if the directory is empty or contains no further objects,
1 if the object is a spectrum,
2 if the object is a directory, and
3 if the object is of some other type.

On failure all functions return -1 and a specific error number can be found in **EGerrno**.

[what about truncation on read/write spectrum?]

SEE ALSO

'Overview of EuroGam Data Acquisition System', EDOC015
'Unified Spectrum Format', EDOC061
'Spectrum Access', EDOC062

‘Spectrum Access Server Protocol’, EDOC078

BUGS

Provision for accessing event-storage objects is not yet complete.