EDOC087

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Eurogam Register Server - C library

Edition 1.0
July 1991

NSF Software Systems Group
UK Science and Engineering Research Council
Daresbury Laboratory

Introduction

This document describes the C interface library for the Eurogam Register Server (ERS) forming part of the NSF data acquisition system.

The library implements the ERS specification described in the document *Eurogam Register Server* (EDOC034.)

In order to use the library, programs should include the header file *ers.h* and should be linked with the libraries *libers.a*.

Summary

The library consists of a number of C procedure calls, each one corresponding to a procedure of the Eurogam Register Server, as described in the *Eurogam Register Server* document. In addition there are miscellaneous routine for setting up and message printing.

The library expects the ERS server to use the UDP port specified in EDOC034. It does not call the server's portmapper to discover the port.

Description

Returns

The eurogam register server calls will return the value OK (0) or a positive value, if sucessfull, or ERROR (-1) if not. In the case of ERROR the procedure status (see EDC034) is copied to the global error status *errno*. A short message describing the error condition can be obtained by calling *ersPerror*.

Eurogam Register Server Calls

int ersNullProc ()

This routine does no work. It calls the server's null procedure. This is a procedure that is made available by all RPC servers to allow server testing and timing.

int ersClaimResource (client, resource, capability)

CLIENT *client;

char *resource;

CAP *capability;

This routine allocates the resource specified by the string *resource* for exclusive use of the client to the supplied . The resourse will generally have the same name as the host.

The server generated capability is copied to the user supplied buffer pointed to by *Capability* . This will act as opaque access token for subsequent procedure requests.

Returns OK or ERROR.

int ersFreeResource (client, capability, resource)

CLIENT *client;

char *capability;

char *resource;

This routine frees the resource specified by the string *resource* .

Returns OK or ERROR.

int ersDefineRegister (client, capability, resource, reg, class)

CLIENT *client;

char *capability;

char *resource;

char *reg;

char *class;

This routine defines a new register with name *reg* of type *class*.

Returns OK or ERROR.

int ersUndefineRegister (client, capability, resource, reg)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

This routine removes the register with name *reg*.

Returns OK or ERROR.

int ersReadRegister (client, capability, resource, reg, value)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

char *value;

This routine reads the register *reg*. The result is copied to the user supplied buffer *value*, which should be large enough to store the expected result.

Number of bytes copied or ERROR.

int ersWriteRegister (client, capability, resource, reg, length, value)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

int length;

char *value;

This routine writes the buffer *value* to the register *reg*. The number of bytes to be written should be specified by *length*.

Returns OK or ERROR.

int ersInitialiseRegister (client, capability, resource, reg)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

This routine initialises the register *reg*.

Returns OK or ERROR.

int ersInquireRegisters (client, capability, resource, pattern, cookie, count, list)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

int cookie;

int count;

REG_LIST **list;

This routine fetches a null terminated list of register names matching *pattern* with a total size of up to *count* bytes. The routine will set the user supplied pointer, pointed to by *List*.

Returns end of data, TRUE (1) or FALSE (0), or ERROR.

int ersReadRegisters (client, capability, resource, pattern, cookie, count, list)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

int cookie;

int count;

VAL_LIST **list;

This routine fetches a null terminated list of register values from registers with names matching *pattern*.

Returns end of data or ERROR.

int ersWriteRegisters (client, capability, resource, pattern, length, value)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

int length;

char *value;

This routine writes the buffer *value* to all the registers with names matching *pattern*.

Returns OK or ERROR.

int ersInitialiseRegisters (client, capability, resource, pattern)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

This routine initialises all the registers with names matching *pattern*.

Returns OK or ERROR.

int ersReadAttribute (client, capability, resource, reg, value)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

char *value;

This routine reads the attributes of the register *reg*. The result is copied to the user supplied buffer *value*, which should be large enough to store the expected result.

Returns number of bytes copied or ERROR.

int ersWriteAttribute (client, capability, resource, reg, length, value)

CLIENT *client;

CAP *capability;

char *resource;

char *reg;

int length;

char *value;

This routine writes the buffer *value* to the attributes of register *reg*. The number of bytes to be written should be specified by *length*.

Returns OK or ERROR.

int ersUndefineRegisters (client, capability, resource, pattern)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

This routine removes all registers with names matching *pattern*.

Returns OK or ERROR.

int ersReadAttributes (client, capability, resource, pattern, cookie, count, list)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

int cookie;

int count;

VAL_LIST **list;

This routine fetches a null terminated list of register attribute values from registers with names matching *pattern*

Returns end of data or ERROR.

int ersWriteAttributes (client, capability, resource, pattern, length, value)

CLIENT *client;

CAP *capability;

char *resource;

char *pattern;

int length;

char *value;

This routine writes the buffer *value* to the attributes of all the registers with names matching *pattern*.

Returns OK or ERROR.

Miscellaneous Routines

CLIENT *ersInitialise (host)

char *host;

This routine initialises the C library to use the register server provided by the system identified by *host*. The routine returns a CLIENT pointer which should be supplied to subsequent ERS calls.

char *ersPerror (string)

char *string;

This routine produces a short error message on the standard error describing the error referred to by *errno*. If *string* is not a NULL pointer and does not point to a null string, the string it points to is printed, followed by a colon, followed by a space, followed by the message. If *string* is a NULL pointer or points to a null string, just the message is printed, followed by a NEWLINE. The message consists of the words “ERS error “ followed by *errno*, followed by a NEWLINE, followed by a description of the error. *Errno* is expected to be an error that was previously returned by one of the NTS calls described above.

char *ersVersion ()

This routine returns the current library version string.