EDOC088

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Eurogam Register Server - Commands

Edition 1.0
July 1991

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

Introduction

This document describes the UNIX command interface to the Eurogam Register Server (ERS) forming part of the NSF data acquisition system. The commands call the functions described in the document *Eurogam Register Server — C library* (EDOC087)

Each time a command is invoked the host, capability and register name are stored in an information file (/usr/tmp/ers_info by default). Subsequent commands will assume these values if they are not specified on the command line.

Commands

ersclaim [–f file] [host]

Claim the resource *host* for the exclusive use of the user. The returned capability is stored in the information file.

ersfree [–f file] [–c capability] [host]

Free the resorce *host* to be used by other users. If *host* is not specified the latest entry in the information file is assumed.

ersdefine [–f file] [–c capability] register type [host]

Define a new register of class *type*.

ersundef [–f file] [–c capability] [register [host]]

Remove a register or registers from the resource *host*.

ersread [–f file] [–c capability] [–k cookie] [–n count] [register [host]] [–a | –b]

Print register value(s).

erswrite [–f file] [–c capability] [–n count] [register [host]] –a asci_data | –b binary_data

Write data to register(s).

ersinit [–f file] [–c capability] [register [host]]

Initialise register(s).

ersinquire [–f file] [–c capability] [–k cookie] [–n count] [register [host]]

Print a list of register names.

ersrda [–f file] [–c capability] [–k cookie] [–n count] [register [host]] [–a | –b]

Print register attribute value(s).

erswta [–f file] [–c capability] [–n count] [register [host]] –a asci_data | –b binary_data

Write data to register attribute(s).

Options

This section describes the options available for the commands above. Users who do not wish to supply all the arguments or to supply them out of order, can do so by specifying them as options.

–a asci_data This option specifies that the command will interpret the rest of the arguments as data in asci format. This option should be the last one.

–b binary_data This option specifies that the command will interpret the rest of the arguments as binary data in hex format. This option should be the last one.

–c capability This option specifies capability string. The capability is stored in the information file when a resource is claimed so it is not recomended to use this option.

–f file This option specifies the name of the information file to use.

–h host This option specifies the host or resource name. If not specified the command will assume the last host entered on the information file.

–k cookie This option specifies cookie value for use when returning lists of registers.

–n count This option specifies the number of bytes of data following the –a or –b options, or the number of bytes to use for lists of registers.

–r register This option specifies the register name. If not specified the command will assume the last register entered in the information file for the requested host. If the register name contains wild cards **,?,[]* then the command will call the equivalent function. eg.*readRegisters()* instead of *readRegister()*. The unix shell will try to expand the wild cards so the name must be enclosed in quotes.

–t type This option specifies the register type or class.

–u If this option is specified the command will display a list of options and quit.

Examples

ersclaim nnvb will claim the resource *nnvb*.

ersfree –f myfile will free the last resource entered in the file *myfile*.

ersdefine –h nnvx a.register aclass will define a register to be known by the resource *nnvx* with name *a.register* and class *aclass*.

ersundef will remove the last register entered in the default information file.

ersread this.register –b will read the register *this.register* and print its value in binary format.

erswrite that.register –b 1 020304 0aff will write 6 bytes of data (values 1,2,3,4,10 and 255) to the register *that.register*.

erswrite "these*registers" nnvx some data strings will write the value *some data strings* to the registers matching the pattern *these*registers*.

ersrda –k 92 –r "*" –h nnvx will print the attributes from all the registers known to the resource *nnvx* starting from cookie *92*.

erswta –n 3 –b 01 02 03 "reg[1,2]" will write 3 bytes of data to the attributes of registers *reg1* and *reg2*.