EDOC062

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Spectrum Access

Edition 1.1
Sept 1991

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

1 Introduction

The document *EDOC061* defines a common format for spectra to be used for both on-line and off-line use. This document describes a set of procedures which may be used to access these spectra. Analysis programs using these procedures can access the spectrum data in a consistent way which is independant of the location of the spectra. The procedures hide all detail of the format of storage from the analysis programs which have no requirement to know, for example, the format of the spectrum header.

Since in general the analysis programs using these procedures will be executing in a workstation which is remote from the actual location of the spectra data and connected via a local area network (ethernet) features are provided in the access procedures intended to allow the volume of data needed to be read across the network to be kept to a minimum. The interface procedures will use the RPC mechanism to access the remote spectrum server. The Spectrum Access Program Protocol is described in the document *EDOC078*.

The spectrum format allows for multi dimensional data arrays. However most spectra are in practice of dimension one or two and many existing analysis programs have been written for fixed dimensional data. Hence interface procedures are provided both for the general case of n dimensional spectra and for the specific cases of 1 and 2 dimensional spectra.

2 Document History

Version 1.0 July 1991 - First version of this document in the document server.
Version 1.1 Sept 1991 - Corresponding to Spectrum Format V2.1 and Access Protocol V1.1.

3 Server Location

The spectrum to be accessed is defined throughout this document by a character string **pathname** which identifies both the server and the spectrum within that server. On-line servers have only a one level name structure and hence **pathname** always refers to either the server (authorisation, create and read names) or to a spectrum. For off-line servers **pathname** refers to the server (authorisation), a disc directory (create and read names) or a disc file which contains a spectrum.

The general format of **pathname** is /server_name/local_name where server_name is either the name of an on-line spectrum server (e.g. **hist** or **sort**) or of an off-line server for disc based spectra (e.g. **disc** which is the default disc spectrum server or **nnsa** which is a specific disc spectrum server) and local_name identifies the spectrum within the server and has a format which is server dependant.

For example: A spectrum in the on-line histogrammer may have the **pathname** /hist/ge26 and one in the on-line sorter may have the **pathname** /sort/adc20_gate2.

4 Procedure Specifications

In all procedures which have a parameter **name** then if the character string **name** starts with a leading '/' character it is an absolute name and the character string **pathname** is just the contents of the character string **name**. In all other cases **name** is relative to the path defined by the procedure EGsetSpectrumpath and the character string **pathname** is formed by concatenating the character string **path** and the character string **name**.

Throughout the procedure specifications parameters in normal type (eg. call) are supplied to the procedure and parameters in bold type (eg. **reply**) are returned by the procedure.

Throughout the procedure specifications **name** is a character string parameter.

Throughout this document the possible values of the parameter **type** are as given in *EDOC061*.

type	
0	8 bit unsigned integer
1	8 bit signed integer
2	16 bit unsigned integer
3	16 bit signed integer
4	32 bit unsigned integer
5	32 bit signed integer
6	32 bit IEEE floating point

Throughout this document the possible values of the parameter **layout** are as given in *EDOC061*.

layout	
0	histogram/full maxtrix
1	half matrix

4.1 Define Path

EGsetSpectrumPath(path)

Allows a current path to be defined by the character string parameter *path*.

4.2 Access Authorisation

Certain servers (for example off-line disc servers) will require authorisation information before access is granted.

EGauthorise(name,id,password)

For the spectrum server identified by **pathname** an access identifier in the character string parameter *id* and associated password in the character string parameter *password* are supplied.

4.3 Spectrum Create

Create a new spectrum in the server and directory identified by **pathname** (off-line servers) or in the server identified by **pathname** (on-line servers).

4.3.1 nD spectrum create

EGcreateSpectrum(name,dimension,base,range,layout,type)

dimension is an integer which contains the number of dimensions in the spectrum.

base is an integer array having at least *dimension* elements which contains the number of the first channel for dimensions 1 to *dimension* of the spectrum and *range* is an integer array having at least *dimension* elements which contains the number of channels for dimensions 1 to *dimension* of the spectrum.

layout is an integer which indicates the format of the spectrum.

type is an integer which indicates the format of each data item in the spectrum.

layout and *type* may be left undefined by setting to the value -1. They must however be defined by a subsequent call on procedure EGsetSpectrumArray before any attempt to write to the spectrum is made.

The error spectrum is undefined by this procedure but may be defined by a subsequent call on procedure EGsetSpectrumArray.

4.3.2 1D spectrum create

EGcreate1dSpectrum(name,base,range,type)

base is an integer containing the number of the first channel in the spectrum and *range* is an integer containing the number of channels in the spectrum.

type is an integer which indicates the format of each data item in the spectrum.

type may be left undefined by setting to the value -1. It must however be defined by a subsequent call on procedure EGsetSpectrumArray before any attempt to write to the spectrum is made.

The error spectrum is undefined by this procedure but may be defined by a subsequent call on procedure EGsetSpectrumArray.

4.3.3 2D spectrum create

EGcreate2dSpectrum(name,base1,range1,base2,range2, layout,type)

base1 is an integer containing the number of the first channel in dimension 1 of the spectrum and *range1* is an integer containing the number of channels in dimension 1 of the spectrum.

base2 is an integer containing the number of the first channel in dimension 2 of the spectrum and *range2* is an integer containing the number of channels in dimension 2 of the spectrum.

layout is an integer which indicates the format of the spectrum.

type is an integer which indicates the format of each data item in the spectrum.

layout and *type* may be left undefined by setting to the value -1. They must however be defined by a subsequent call on procedure EGsetSpectrumArray before any attempt to write to the spectrum is made.

The error spectrum is undefined by this procedure but may be defined by a subsequent call on procedure EGsetSpectrumArray.

4.4 Set Spectrum Array

Allows the array layout and type to be defined or redefined.

EGsetSpectrumArray(name,number,layout,type)

number is 1 to define the spectrum data array and 2 to define the error spectrum data array.

layout is an integer which indicates the format of the spectrum.

type is an integer which indicates the format of each data item in the spectrum.

4.5 Locate Spectrum

Locate an existing spectrum identified by **pathname**. Enables the existance of a spectrum to be tested for and the result returned.

EGlocateSpectrum(name)

4.6 Delete Spectrum

Delete the spectrum identified by **pathname**.

EGdeleteSpectrum(name)

4.7 Define Default Array

Defines the number of the data array to be accessed by subsequent calls of the read spectrum and write spectrum procedures.

EGsetDefaultArray(number)

number is 1 to access the spectrum data array and 2 to access the error spectrum data array. By default the spectrum data array is accessed.

4.8 Read Spectrum Counts

Read spectrum data. An area of the spectrum may be fetched which is of a size (normally greater) different from the array into which the data is to be returned. In this case the original data will be summed and scaled in order to generate a histogram of the requested size and type.

4.8.1 Define Data Scaling

Defines the range for each dimension in subsequent calls of EGread*Spectrum.

EGsetDefaultScale(dimension,size)

dimension is an integer which contains the number of dimensions in the spectrum.

size is an integer array having at least *dimension* elements which contains the number of elements for dimensions 1 to *dimension* of *array* which is supplied in subsequent calls of read spectrum procedures.

By default data scaling does not occur. An element of *size* set to zero implies no scaling in that dimension.

The effect of this procedure may be reset by a call with the elements of *size* set to zero.

4.8.2 nD Spectrum Data Read

EGreadSpectrum(name,dimension,base,range,**array**, type)

dimension is an integer which contains the number of dimensions in the spectrum.

base is an integer array having at least *dimension* elements which contains the base channel for dimensions 1 to *dimension* to be used by the read request and *range* is an integer array having at least *dimension* elements which contains the number of channels for dimensions 1 to *dimension* required by the read request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is returned in *array* with the spectrum counts stored with the last dimension varying fastest (the standard C ordering).

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

If the elements of *size* have been defined by a previous call on procedure EGsetDefaultScale and are different from the elements of *range* then the spectrum data requested by *base* and *range* will be scaled to fit the space defined by *size*.

4.8.3 1D Spectrum Read

EGread1dSpectrum(name,base,range,**array**,type)

base is an integer which contains the base channel to be used by the read request and *range* is an integer which contains the number of channels required by the read request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is returned in *array*.

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

If *size* has been defined by a previous call on procedure EGsetDefaultScale and is different from *range* then the spectrum data requested by *base* and *range* will be scaled to fit the space defined by *size*.

Example - channels 0 to 3000 of a spectrum are to be read and returned as 16 bit unsigned integers
EGread1dSpectrum(name,0,3000,array,2)

Example - channels 0 to 4095 of a spectrum are to be read and returned as 16 bit unsigned integers in a 512 element array
EGsetDefaultScale(512)

EGread1dSpectrum(name,0,4095,array,2)

4.8.4 2D Spectrum Read

EGread2dSpectrum(name,base1,range1,base2,range2,**array**, type) *base1* and *base2* are integers which contains the base channels for dimensions 1 and 2 to be used by the read request and *range1* and *range2* are integers which contains the number of channels in dimensions 1 and 2 required by the read request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is returned in *array* with the spectrum counts stored with the last dimension varying fastest (the standard C ordering).

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

If the elements of *size* have been defined by a previous call on procedure EGsetDefaultScale and are different from *range** then the spectrum data requested by *base** and *range** will be scaled to fit the space defined by *size*.

Example - a 4K by 4K 2D matrix is to be returned as 16 bit signed integers.

EGread2dSpectrum(name,0,4096,0,4096,array,3)

4.9 Write Spectrum Counts

4.9.1 nD Spectrum Data Write

EGwriteSpectrum(name,dimension,base,range,array,type)

dimension is an integer which contains the number of dimensions in the spectrum.

base is an integer array having at least *dimension* elements which contains the base channel for dimensions 1 to *dimension* to be used by the write request and *range* is an integer array having at least *dimension* elements which contains the number of channels for dimensions 1 to *dimension* supplied by the write request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is supplied in *array* with the spectrum counts stored with the last dimension varying fastest (the standard C ordering).

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

4.9.2 1D Spectrum Write

EGwrite1dSpectrum(name,base,range,array,type) *base* is an integer which contains the base channel to be used by the write request and *range* is an integer which contains the number of channels supplied by the write request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is supplied in *array*.

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

4.9.3 2D Spectrum Write

EGwrite2dSpectrum(name,base1,range1,base2,range2,array, type)

base1 and *base2* are integers which contains the base channels for dimensions 1 and 2 to be used by the write request and *range1* and *range2* are integers which contains the number of channels in dimensions 1 and 2 supplied by the write request.

The *base* and *range* arguments must define a region of counts lying entirely within the spectrum.

The data is supplied in *array* with the spectrum counts stored with the last dimension varying fastest (the standard C ordering).

type is an integer which indicates the format of each data item in *array*. *type* need not be the same as the type of the counts held in the spectrum. The appropiate sign extension or truncation will take place automatically.

4.10 Read Header String

The spectrum format allows for up to 16 information strings to be held for each spectrum plus one annotation, one calibration and one efficiency string per dimension.

4.10.1 Read Information String

EGreadInformation(name,number,**string**)

number is an integer which contains the number of the information string to be returned.

The data is returned in the character string parameter **string**.

Certain of the 16 available information strings are by convention used to hold specific types of data associated with the spectrum. For convenience specific procedures are available to access these.

EGreadTitle(name,**string**)
equivalent to EGreadInformation with number equal 1

EGreadExpt(name,**string**)

equivalent to EGreadInformation with number equal 2

EGreadRun(name,**string**)
equivalent to EGreadInformation with number equal 3

EGreadComment(name,**string**)
equivalent to EGreadInformation with number equal 4

4.10.2 Read Annotation String

EGreadAnnotation(name,number,**string**)

number is an integer which contains the spectrum dimension associated with the annotation string to be returned.
The data is returned in the character string parameter **string**.

4.10.3 Read Calibration String

EGreadCalibration(name,number,**string**)

number is an integer which contains the spectrum dimension associated with the calibration string to be returned.
The data is returned in the character string parameter **string**.

4.10.4 Read Efficiency String

EGreadEfficiency(name,number,**string**)

number is an integer which contains the spectrum dimension associated with the efficiency string to be returned.
The data is returned in the character string parameter **string**.

4.11 Write Header String

4.11.1 Write Information String

EGwriteInformation(name,number,string)

number is an integer which contains the number of the information string to be written.
The data is supplied in the character string parameter **string**.
Certain of the 16 available information strings are by convention used to hold specific types of data associated with the spectrum. For convenience specific procedures are available to access these.

EGwriteTitle(name,string)
equivalent to EGwriteInformation with number equal 1

EGwriteExpt(name,string)
equivalent to EGwriteInformation with number equal 2

EGwriteRun(name,string)
equivalent to EGwriteInformation with number equal 3

EGwriteComment(name,string)
equivalent to EGwriteInformation with number equal 4

4.11.2 Write Annotation String

EGwriteAnnotation(name,number,string)

number is an integer which contains the spectrum dimension associated with the annotation string to be written.

The data is supplied in the character string parameter **string**.

4.11.3 Write Calibration String

EGwriteCalibration(name,number,string)

number is an integer which contains the spectrum dimension associated with the calibration string to be written.

The data is supplied in the character string parameter **string**.

4.11.4 Write Efficiency String

EGwriteEfficiency(name,number,string)

number is an integer which contains the spectrum dimension associated with the efficiency string to be written.

The data is supplied in the character string parameter **string**.

4.12 Inquire Spectrum Attributes

Obtain miscellaneous information about the spectrum.

4.12.1 nD Inquire Attributes

EGinquireSpectrum(name,**dimension**,**base**,**range**, **layout**, **type**)

dimension is an integer which is returned with the number of dimensions in the spectrum.

base is an integer array having at least **dimension** elements which is returned with the number of the first channel for dimensions 1 to **dimension** of the spectrum and **range** is an integer array having at least **dimension** elements which is returned with the number of channels for dimensions 1 to **dimension** of the spectrum.

layout is an integer array having 2 elements which is returned with the format of the data arrays.

type is an integer array having 2 elements which is returned with the format of each data item in the data arrays.

4.12.2 1D Inquire Attributes

EGinquire1dSpectrum(name,**base**,**range**, **type1**, **type2**)

base is an integer which is returned with the number of the first channel in the spectrum and **range** is an integer which is returned with the number of channels in the spectrum.

type1 is an integer returned which indicates the format of each data item in the spectrum and **type2** is an integer returned which indicates the format of each data item in the error spectrum. If the error spectrum is undefined the value -1 will be returned.

4.12.3 2D Inquire Attributes

EGinquire2dSpectrum(name,**base1**,**range1**,**base2**,**range2**,**layout1**, **layout2**, **type1**,**type2**)

base1 is an integer which is returned with the number of the first channel in dimension 1 of the spectrum and **range1** is an integer which is returned with the number of channels in dimension 1 of the spectrum.

base2 is an integer which is returned with the number of the first channel in dimension 2 of the spectrum and **range2** is an integer which is returned with the number of channels in dimension 2 of the spectrum.

layout1 is an integer returned which indicates the format of the spectrum and **layout2** is an integer returned which indicates the format of the error spectrum. If the error spectrum is undefined the value -1 will be returned.

type1 is an integer returned which indicates the format of each data item in the spectrum and **type2** is an integer returned which indicates the format of each data item in the error spectrum. If the error spectrum is undefined the value -1 will be returned.

4.13 Miscellaneous

Available with on-line servers only. These procedures are intended for use only by data aquisition software which requires direct access to the spectrum counts space and are not expected to be of general interest.

4.13.1 Obtain VME address of data array

Returns the VME address of the base of the data array associated with the spectrum.

EGinquireAddress(name,**address**)

For the spectrum identified by *pathname* returns the VME address in **address**.

4.14 Read Names

Obtain the next name from the location identified by **pathname**. For on-line spectra **pathname** is just the name of a server. For off-line spectra **pathname** should identify a disc directory.

EGinquireDirectory(name,**string**)

The first name is returned in the character string parameter **string**.

EGinquireDirectoryMore(**string**)

The next name is returned in the character string parameter **string**.
Successive calls to this procedure allow the names of all objects within the defined directory or server to be obtained.
The procedures EGinquireDirectory and EGinquireDirectoryMore return an integer which indicates the type of the name returned.

value	
0	the name refers to a spectrum
1	the name refers to a directory
2	the name refers to a something else
3	the directory is empty or the end of the directory has been reached

4.15 Generate Error Message

Converts the error number returned by a procedure into a meaningfull text message.

spgenerrmess(**text**)

Converts EGerrno into a text string which is returned in the character string parameter **text**.

5 Return_Codes

To be supplied

6 Language Bindings

The interface procedures support bindings for the C and Fortran languages.

The procedure names are valid for C. While not valid for ANSI Fortran 77 they are valid for Fortran 90 and are acceptable for Sun Fortran. The procedure names given in this document may thus be used by both Fortran and C bindings within the Eurogam environment.

Throughout the procedure specifications variable length text parameters are identified as *character string parameters*. For the C binding such parameters should be of type *char* and the text string should follow the C convention and be terminated by a null character. For procedures which return a text string the space reserved by the caller should be large enough to hold the text string plus the terminating null character. For the Fortran binding such parameters should be of type *character* and be padded as necessary with trailing space characters.

For Fortran use the procedures should be invoked as subroutines.

7 Implementation

To be supplied

8 Further Study

read spectrum projections (slicing)
scaling options for read
debugging (and other) options